ฐานข้อมูลเอ็กซ์เอ็มแอล

XML DATABASE

กนกวรรณ ตันติพาณิชย์กูล

กิตติศักดิ์ จิรภิวงศ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2550

ปริญญานิพนธ์ปีการศึกษา 2550

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ฐานข้อมูลเอ็กซ์เอ็มแอล

XML DATABASE

ผู้จัดทำ

1. นางสาวกนกวรรณ ตันติพาณิชย์กูล รหัสนักศึกษา 47010005

2. นายกิตติศักดิ์ จิรภิวงศ์ รหัสนักศึกษา 47010050

อาจารย์ที่ปรึกษา

(รศ.ดร.ศุภมิตร จิตตะยโศธร)

ฐานข้อมูลเอ็กซ์เอ็มแอล

นางสาวกนกวรรณ ตันติพาณิชย์กูล 47010005
นายกิตติศักดิ์ จิรภิวงศ์ 47010050
รศ.ดร.ศุภมิตร จิตตะยโศธร อาจารย์ที่ปรึกษา
ปีการศึกษา 2550

บทคัดย่อ

เทคโนโลยี XML นั้นเป็นเทคโนโลยีที่กำลังได้รับความสนใจในอินเตอร์เน็ตเป็นอย่างมาก ซึ่งมีประโยชน์ในการพรรณนาข้อมูล จึงทำให้ในอินเตอร์เน็ตมีข้อมูลที่เก็บอยู่ในรูป XML เป็นจำนวนมาก และมีการพัฒนาเทคโนโลยีที่จะมาสนับสนุน XML ให้มีความสามารถที่ดีขึ้น ไม่ว่าจะเป็น XSL, XSLT, XPath, XQuery, XLink, XPointer, DTD และ XML schema นอกจากนั้นเทคโนโลยี RDF ซึ่งเขียนโดย XML นั้นมีความน่าสนใจ เนื่องจาก RDF กำหนดรูปแบบในการอธิบายเนื้อหาข้อมูลให้อยู่ในรูปแบบเป็นมาตรฐานเดียวกัน และ RDF ยังมี RDFS, OWL และ SPARQL เข้ามาสนับสนุนอีกด้วย

งานวิจัยชิ้นนี้ชี้ให้เห็นการนำเสนอการมอง XML และ RDF เป็นฐานข้อมูล และสามารถนำเอกสาร XML และ RDF จำนวนมากในอินเตอร์เน็ตมาใช้ประโยชน์ได้อย่างไรบ้าง ทำอย่างไรจึงใช้ประโยชน์จากเอกสาร XML และ RDF ได้ที่มีอยู่ในระบบอินเตอร์เน็ตได้ในฐานะฐานข้อมูลชนิดหนึ่ง

เมื่อเรามองเอกสาร XML ในรูปแบบฐานข้อมูลแล้ว สามารถใช้ XQuery ในการ query ข้อมูลในเอกสาร XML หลายๆ เอกสารจากหลายๆ แหล่งข้อมูลในระบบอินเตอร์เน็ตได้ และเมื่อเรามองเอกสาร RDF ในรูปแบบฐานข้อมูลแล้ว สามารถใช้ SPARQL ในการ query ข้อมูลในเอกสาร RDF และ OWL จากหลายๆ แหล่งข้อมูลในระบบอินเตอร์เน็ตได้ง่ายกว่าการ query โดยใช้ XQuery เนื่องจาก RDF และ OWL มี ontology และความสัมพันธ์ของข้อมูลเข้ามาเกี่ยวข้อง การ query เอกสาร XML ด้วย XQuery และการ query เอกสาร RDF และ OWL ด้วย SPARQL ดังกล่าวนั้น เปรียบได้กับการใช้ SQL ในการ query ข้อมูลใน relational database นั่นเอง นั่นหมายความว่าอินเตอร์เน็ตเปรียบเสมือนฐานข้อมูลขนาดใหญ่ที่สุดในโลกที่รวบรวมข้อมูลต่างๆ ไว้ ทุกคนสามารถเข้ามาใช้ประโยชน์ได้อย่างเต็มที่

XML Database

Ms. Kanokwan	Tantipanichkul	47010005
Mr. Kitisak	Chirapiwong	47010050
Assoc.Prof.Dr. Suphamit	Chittayasothorn	Advisor

Academic Year 2007

ABSTRACT

XML technology has become to be on spot lately on Internet society. One of its interesting properties was known to be able to describe the essence of data which stored as a tree. Undoubtedly, XML plays the major roles in today's Internet data storage. The supporting technologies of XML have been developing for since, results in XSL, XSLT, XPATH, XQuery, XLink, XPointer, DTD and XML schema. Moreover, RDF technology is interesting because it define a form to describe data and standardize the data. The supporting technologies of RDF have been developing for since, results in RDFS, OWL and SPARQL.

This project aimed to propose, firstly the new perspective of viewing XML and RDF as a database and secondly, answering these questions on "How to make use of the scattered XML and RDF documents on the Internet?" and "How to make use of them in the view of relational database?"

Principle lies on once we choose to see the XML document in the different view, as the database view, we can use the supporting technology, for instance XQuery, to get information from various XML sources on the Internet and SPARQL, to get information from various RDF and OWL sources on the Internet that is easier than using XQuery because RDF has an ontology and the relationship of the data. In the other words, once viewing Internet as the gigantic database, querying the XML as its elements and querying the RDF and OWL as its elements are similar to use the traditional SQL statement query the result out of its source. Eventually, everyone could find interesting in its use.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ดีด้วยคำแนะนำ คำปรึกษา และคอยดูแลจากหลายๆ ฝ่าย ด้วยกัน โดยเฉพาะอย่างยิ่งอาจารย์ที่ปรึกษาที่คอยให้ความเอาใจใส่ ให้คำแนะนำ และความช่วยเหลือเสมอมา คือ รศ.ดร.ศุภมิตร จิตตะยโศธร ซึ่งต้องกราบขอบพระคุณเป็นอย่างสูง

ขอขอบคุณภาควิชาวิศวกรรมคอมพิวเตอร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังที่ได้จัดเตรียมสิ่งอำนวยความสะดวก เพื่อให้การดำเนินงานเป็นไปด้วยความสะดวกรวดเร็ว โดยเฉพาะอินเตอร์เน็ตความเร็วสูงที่มีให้บริการสำหรับการค้นคว้าหาความรู้ต่างๆ ซึ่งท้ายที่สุดแล้วได้ประกอบกันเป็นส่วนหนึ่งของโครงงานนี้

ขอขอบคุณพี่ๆ และเพื่อนๆ ที่คอยสร้างความครึกครื้นเผื่อผ่อนคลายจากความเครียดในการทำงาน และเป็นกำลังใจให้เสมอมา

และที่ขาดไม่ได้เลยต้องขอขอบคุณบุคคลที่มีความสำคัญที่สุดในชีวิต นั่นคือบิดามารดาและครอบครัวอันเป็นที่เคารพรัก ซึ่งได้ให้การอบรมเลี้ยงดูเป็นอย่างดี พร้อมกับให้โอกาสในการศึกษาอย่างเต็มที่ อีกทั้งยังให้กำลังใจ มอบความรักให้เสมอมา

กนกวรรณ ตันติพาณิชย์กูล
กิตติศักดิ์ จิรภิวงศ์

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญตาราง (ต่อ)

ตารางที่ **หน้า**

สารบัญรูป

หน้า

สารบัญรูป (ต่อ)

หน้า

สารบัญรูป (ต่อ)

หน้า

สารบัญรูป (ต่อ)

หน้า

สารบัญรูป (ต่อ)

หน้า

สารบัญรูป (ต่อ)

หน้า

สารบัญรูป (ต่อ)

หน้า

สารบัญรูป (ต่อ)

หน้า

สารบัญรูป (ต่อ)

หน้า

สารบัญรูป (ต่อ)

หน้า

สารบัญรูป (ต่อ)

หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาของปัญหา

เนื่องจากในปัจจุบันอินเตอร์เน็ตเข้ามามีบทบาทในชีวิตประจำวันของมนุษย์มากขึ้น ในบริษัทหรือองค์กรต่างๆ มีการสื่อสารกันโดยใช้อินเตอร์เน็ต และยังมีเทคโนโลยีใหม่ๆ เช่น ภาษา XML, เทคโนโลยี RDF และ OWL เกิดขึ้น โดยเราสามารถใช้เทคโนโลยีเหล่านี้เป็นสื่อกลางในการอธิบายข้อมูลที่มีอยู่ได้ในรูปแบบของเอกสาร ทำให้เกิดความคิดที่จะมองเอกสารที่ใช้เทคโนโลยีดังกล่าวในรูปแบบของฐานข้อมูลกันเป็นอย่างมากในวงการนักวิชาการ

จะดีแค่ไหนถ้าหากเราต้องการทราบผลข้อมูลใดๆ แล้วเราสามารถ query เอกสาร XML หรือ RDF ที่มีจำนวนมากมายในอินเตอร์เน็ต แล้วได้ผลลัพธ์ของการ query ออกมาเป็นเอกสาร XML หรือ RDF ฉบับเดียวที่มีเนื้อหาครบถ้วน หรืออาจใช้ ontology เพื่อเป็นตัวช่วยในการ query ข้อมูลได้ง่ายขึ้น

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 เพื่อศึกษา XML และ RDF ในฐานะฐานข้อมูล

1.2.2 เพื่อศึกษาภาษา XQuery และความสามารถของภาษา XQuery เมื่อเทียบกับ SQL

1.2.3 เพื่อศึกษาภาษา SPARQL

1.2.4 เพื่อศึกษา ontology ของข้อมูล และวิธีการสร้าง ontology และความสามารถของภาษา SPARQL ในการ query เอกสาร OWL

1.2.5 เพื่อศึกษาขีดความสามารถของ DBMS ที่เกี่ยวข้องกับ XML

1.2.6 เพื่อทดลองใช้ความสามารถในการจัดการ XML

1.3 ประโยชน์ที่คาดว่าจะได้รับ

1.3.1 ได้รับความรู้ ความเข้าใจเกี่ยวกับ XML และ RDF ในฐานะฐานข้อมูล

1.3.2 ได้รับความรู้ ความเข้าใจเกี่ยวกับภาษา XQuery รวมทั้งสามารถบอกข้อดีข้อเสียและความแตกต่างของ XQuery กับ SQL ได้

1.3.3 ได้รับความรู้ ความเข้าใจเกี่ยวกับภาษา SPARQL ในการ query เอกสาร RDF และ OWL

1.3.4 มีความรู้เกี่ยวกับ ontology ของข้อมูล และวิธีการสร้าง ontology และความสามารถของภาษา SPARQL ในการ query เอกสาร OWL

1.3.5 ได้รับความรู้ ความเข้าใจเกี่ยวกับขีดความสามารถของ DBMS ที่เกี่ยวข้องกับ XML

1.3.6 สามารถจัดการ XML และ RDF ได้

1.4 ขอบเขตของโครงงาน

ศึกษาเทคโนโลยีของภาษา XML และ XQuery แล้วนำข้อมูลที่ได้จากการศึกษามาทำการทดลอง query ข้อมูลที่มีโครงสร้างแบบ relational database และข้อมูลที่ไม่ใช่ relational database นั่นคือแบบ hierarchy เพื่อให้เห็นจริงในเรื่องของความสามารถของการใช้ XQuery โดยสร้างเอกสาร XML 2 อย่าง คือ เอกสาร XML ที่ข้อมูลต่างๆ อยู่ในลักษณะ flatted file และเอกสาร XML ที่ข้อมูลต่างๆ อยู่ในรูปแบบของ Hierarchy เมื่อเทียบกับการใช้ SQL ในการ query relational database จากนั้นนำความรู้ที่ได้จากการทดลองมาหาข้อดีข้อเสียและความแตกต่างของ XQuery กับ SQL

ต่อจากนั้นจึงศึกษาเกี่ยวกับการค้นหาเอกสาร XML ที่อยู่บนอินเตอร์เน็ตโดยใช้การ query ที่เราเขียนขึ้นเองที่คอมพิวเตอร์เครื่องหนึ่งที่ต่อกับเครือข่ายอินเตอร์เน็ตอยู่ และวิธีการ query ให้ได้มาซึ่งเอกสาร XML ที่อยู่บนอินเตอร์เน็ตนั้น ต่อมาศึกษาเทคโนโลยีที่เกี่ยวกับ XML ของ DBMS ต่าง ๆ เช่น Oracle, DB2 หรือ SQL Server เป็นต้น และเปรียบเทียบเพื่อหาความคล้ายและแตกต่างกันของเทคโนโลยีที่เกี่ยวกับ XML ของ DBMS ต่างๆ นั้น

ต่อมาจึงศึกษาเทคโนโลยีของ RDF และ SPARQL จากนั้นทดลองสร้างไฟล์ RDF แล้วใช้ SPARQL ในการ query เอกสาร RDF เพื่อเป็นการทดลองศึกษาความสามารถของ SPARQL และศึกษาความสามารถของ RDF ที่เพิ่มขึ้นมาจาก XML

จากนั้นจึงศึกษา ontology ของเอกสาร RDF แล้วทดลองแลกเปลี่ยน ontology ระหว่างกัน และใช้ SPARQL ทดลอง query ข้อมูลในเอกสาร ontology นั้นๆ

1.5 วิธีการดำเนินงาน

1.5.1 ศึกษา XML ในฐานะฐานข้อมูล

1.5.2 ศึกษาภาษา XQuery และความสามารถของภาษา XQuery เมื่อเทียบกับ SQL

1.5.3 ศึกษาขีดความสามารถของ DBMS ที่เกี่ยวข้องกับ XML

1.5.4 ทดลองใช้ความสามารถในการจัดการ XML

1.5.5 ศึกษา RDF ในฐานะฐานข้อมูล

1.5.6 ศึกษาภาษา SPARQL และความสามารถของภาษา SPARQL ในการ query เอกสาร RDF

1.5.7 ทดลองสร้างเอกสาร RDF และ ใช้ SPARQL ในการ query เอกสารนั้น

1.5.8 ศึกษา Web Ontology Language (OWL)

1.5.9 ทดลองสร้าง ontology และแลกเปลี่ยน ontology ระหว่างกัน และใช้ SPARQL ทดลอง query ข้อมูลที่เป็นส่วน ontology

1.6 ส่วนประกอบของปริญญานิพนธ์

ปริญญานิพนธ์ฉบับนี้มีเนื้อหาทั้งหมด 4 บท ได้แก่

เนื้อหาในบทที่ 1 กล่าวถึงความเป็นมาของปัญหา วัตถุประสงค์ของโครงงาน ประโยชน์ที่คาดว่าจะได้รับ ขอบเขตของโครงงาน วิธีการดำเนินงาน และส่วนประกอบของรายงานฉบับนี้

เนื้อหาในบทที่ 2 กล่าวถึงทฤษฎีที่เกี่ยวข้องกับ XML ที่นำมาใช้สำหรับโครงงานนี้ ซึ่งมีเนื้อหาประกอบด้วย ทฤษฎีพื้นฐานของภาษา XML, ความถูกต้องของเอกสาร XML, การกำหนดโครงสร้างเอกสารแบบ DTD และ XML schema, การประยุกต์ใช้งานภาษา XML, XPath, XQuery, XQuery core และแบบจำลองกระบวนการทำงานของ XQuery (XQuery processing model)

เนื้อหาในบทที่ 3 กล่าวถึงทฤษฎีที่เกี่ยวข้องกับ RDF, RDF Schema (RDFS), OWL (Web Ontology Language) และ SPARQL

เนื้อหาในบทที่ 4 กล่าวถึงการทดลองทั้ง 3 การทดลอง คือ การทดลอง query เอกสาร XML โดยใช้ภาษา XQuery การทดลอง query เอกสาร RDF โดยใช้ภาษา SPARQL และการทดลองสร้าง ontology และแลกเปลี่ยน ontology ระหว่างกัน แล้วใช้ SPARQL เพื่อ query ข้อมูลใน ontology นั้น โดยในการทดลอง query เอกสาร XML โดยใช้ภาษา XQuery นั้น ได้มีการทดลอง query เพื่อเปรียบเทียบความสามารถของ XQuery เมื่อเทียบกับ SQL ด้วย

เนื้อหาในบทที่ 5 เป็นบทวิจารณ์และสรุป ซึ่งกล่าวถึงบทสรุปของโครงงาน วิจารณ์สิ่งที่ได้รับจากโครงงาน ปัญหาอุปสรรคและแนวทางแก้ไข และข้อเสนอแนะสำหรับเป็นแนวทางในการพัฒนาต่อ

บทที่ 2
ทฤษฎีที่เกี่ยวข้องกับ XML

บทนี้จะกล่าวถึงทฤษฎีที่เกี่ยวข้องกับ XML ที่นำมาใช้สำหรับโครงงานนี้ ซึ่งมีเนื้อหาประกอบด้วย ทฤษฎีพื้นฐานของภาษา XML, ความถูกต้องของเอกสาร XML, การกำหนดโครงสร้างเอกสารแบบ DTD และ XML schema, การประยุกต์ใช้งานภาษา XML, XPath, XQuery, XQuery core และแบบจำลองกระบวนการทำงานของ XQuery (XQuery processing model)

2.1 ทฤษฎีพื้นฐานของภาษา XML

2.1.1 ประวัติความเป็นมาของภาษา XML

ภาษา XML เป็นภาษาที่ถูกเผยแพร่โดยสมาคม W3C (World Wide Web Consortium) จุดประสงค์เพื่อสร้างภาษาที่สามารถนิยามข้อมูลได้ (data definition) XML จัดเป็นภาษาที่อยู่ในตระกูลของภาษาที่ใช้อธิบายและ/หรือให้คำจำกัดความของข้อมูล (XML: Extensible Markup Language) ภาษา XML เป็นภาษาที่มีรากฐานมาจากภาษา GML (Generalize Markup Language) และต่อมาได้มีการพัฒนาภาษา SGML (Standard Generalize Markup Language) แต่ภาษา SGML มีข้อจำกัดในการใช้งานจึงได้มีการพัฒนาภาษา HTML (Hyper Text Markup Language) สำหรับใช้ในการแสดงผลข้อมูลบนเวปไซต์ ต่อมาได้มีการนำข้อดีของภาษา SGML และ HTML มาพัฒนาเป็นภาษา XML ซึ่งเป็นเทคโนโลยีที่เกี่ยวพันกับการกำหนดลักษณะ และโครงสร้างของข้อมูล ซึ่งเป็นข้อมูลที่สามารถอธิบายความหมายของตัวข้อมูลเองได้ (Self-Described Data) โดยอาศัยแท็กที่ผู้ใช้กำหนดขึ้นเพื่อนิยามข้อมูล ซึ่งมีประโยชน์สำหรับการนำมาวางรูปแบบเอกสาร ทำให้สามารถนำมาเป็นรูปแบบในการแลกเปลี่ยนเอกสารระหว่างหน่วยงาน หรือองค์กรได้โดยไม่ต้องกังวลเรื่องความแตกต่างของรูปแบบข้อมูล จุดเด่นของภาษา XML ที่ได้รับความนิยมและแพร่หลายในปัจจุบันคือเป็นภาษาที่แสดงผลได้หลายแพลตฟอร์ม (platform independent)

2.1.2 โครงสร้างเอกสาร XML

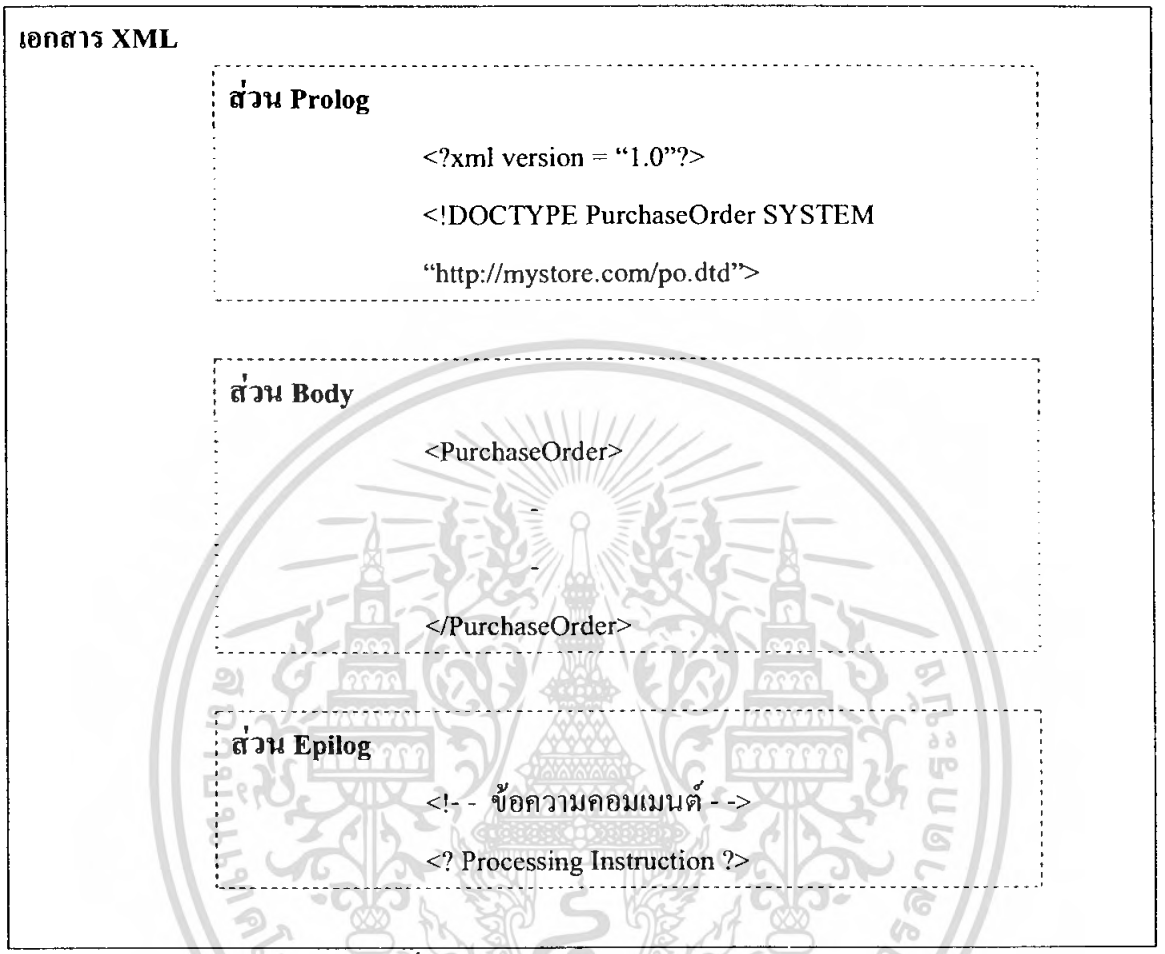
เอกสาร XML มีโครงสร้างประกอบด้วย 3 ส่วนหลักๆ ดังนี้

Prolog ส่วนนี้ประกอบด้วย 2 ส่วนย่อย คือ XML declaration และ document type declaration

Body เป็นส่วนของเนื้อหาเอกสารซึ่งประกอบด้วยแท็ก (tag) ที่นิยามข้อความหรือข้อมูล และข้อมูลภายในแท็ก

Epilog คือส่วนที่เป็นข้อความของคอมเมนต์ (comment) และ PI (Processing Instuction) การแสดงตำแหน่งของ Epilog ที่แสดงในรูปที่ 2.1 แสดงในส่วนท้ายของเอกสาร เพื่อต้องการแยก

เอกสารออกเป็นสัดส่วนชัดเจน แต่ในทางปฏิบัติจริงนั้น ส่วนของคอมเมนต์จะสอดแทรกอยู่ตามเนื้อหาเอกสาร



รูปที่ 2.1 โครงสร้างของเอกสาร XML

โครงสร้างของเอกสาร XML มีข้อมูลที่สำคัญที่ต้องให้ความสนใจ 4 ประเภท คือ การประกาศเอกสาร XML (XML declaration) การบอกประเภทของเอกสาร XML (document type declaration) ข้อความคอมเมนต์ (comment) และการแทรกส่วนประมวลผล PI (Processing Instruction)

2.1.2.1 การประกาศเอกสาร XML (XML declaration)

การประกาศเอกสาร XML เป็นการประกาศให้ทราบว่าเอกสารนี้เป็นเอกสาร XML ไม่ใช่ไฟล์ข้อความธรรมดา นอกจากเอกสาร XML แล้ว เอกสารภาษาอื่นๆ ที่มีรากฐานมาจากภาษา XML ก็ต้องมีการประกาศความเป็นภาษานั้นๆ ด้วย เช่น ภาษา XHTML, ภาษา WML ที่ใช้กับ WAP (Wireless Application Protocol) รูปแบบการประกาศเอกสาร XML แสดงดังรูปที่ 2.2

```
<?xml version = "1.0" encoding = "ชุดอักษร" standalone = "yes|no"?>
```

รูปที่ 2.2 รูปแบบการประกาศเอกสาร XML

ตัวอักษรทุกตัวของแต่ละแอตทริบิวต์ในรูปแบบการประกาศเอกสาร XML จะต้องเป็นตัวพิมพ์เล็ก (lowercase) ห้ามเป็นตัวอักษรพิมพ์ใหญ่ (uppercase) และแต่ละแอตทริบิวต์ที่ปรากฏในรูปที่ 2.2 ต้องเรียงลำดับกันตามที่แสดงเท่านั้น ความหมายของแต่ละแอตทริบิวต์มีรายละเอียดดังนี้

version เป็นแอตทริบิวต์ที่ต้องระบุไว้เสมอ เพื่อบ่งบอกรุ่นของ XML รูปแบบที่แสดงดังรูปที่ 2.2 เป็น XML เวอร์ชัน 1.0 ในอนาคตอาจเปลี่ยนแปลงได้

encoding เป็นแอตทริบิวต์ตัวเลือก (optional) ที่ระบุเมื่อจำเป็นต้องใช้เท่านั้น โดยบ่งบอกชุดรหัสอักษรที่ใช้ในเอกสาร XML เพื่อให้ตัวแปลเอกสาร (XML parser) สำหรับค่าปกติคือ utf-8

standalone เป็นแอตทริบิวต์ตัวเลือก (optional) ที่ระบุเมื่อจำเป็นต้องใช้เท่านั้น แอตทริบิวต์นี้จะบอกให้รู้ว่าเอกสาร XML ขึ้นอยู่กับเอกสารอื่นหรือไม่ ซึ่งมีค่าที่เป็นไปได้เพียงค่าเดียว คือ yes หรือ no ถ้าเป็น yes หมายถึง เอกสาร XML นั้นไม่ขึ้นกับเอกสารอื่น แต่ถ้าเป็น no หมายถึง เอกสาร XML นั้นขึ้นกับเอกสารอื่น เช่น เอกสารการกำหนดโครงสร้างของ XML แบบ DTD หรือ XML Schema ซึ่งรายละเอียดของการกำหนดโครงสร้างของเอกสาร XML จะกล่าวถึงในหัวข้อ 2.3 สำหรับค่าปกติของแอตทริบิวต์นี้คือ no

2.1.2.2 การบอกประเภทของเอกสาร XML (document type declaration)

การบอกประเภทของเอกสาร XML คือการประกาศชนิดของเอกสาร XML และไฟล์ของ DTD ที่กำหนดโครงสร้างของเอกสาร ตัวอย่างแสดงได้ดังรูปที่ 2.3

```
<?xml version = "1.0" encoding = "ชุดอักษร" standalone =
"yes|no"?>
```

รูปที่ 2.3 ตัวอย่างการประกาศเอกสาร XML และไฟล์ DTD

ตัวอย่างในรูปที่ 2.3 ประกาศให้ทราบว่าเอกสาร XML นี้มีอิลิเมนต์ PurchaseOrder เป็นรูตอิลิเมนต์ และมีไฟล์ DTD ชื่อ po.dtd อยู่ที่ http://mystore.com/po.dtd สำหรับคีย์เวิร์ด SYSTEM ต้องระบุไว้เพื่อให้ทราบว่าเป็นการอ้างอิงตำแหน่งของไฟล์ด้วย URL ในกรณีที่อ้างอิงตำแหน่งของไฟล์แบบ Local จะต้องใช้คีย์เวิร์ดเป็น PUBLIC เช่น ไฟล์ที่อยู่ในเครื่องคอมพิวเตอร์เดียวกัน

การประกาศประเภทของเอกสาร XML สามารถประกาศได้ 2 แบบ แบบที่หนึ่งประกาศแบบภายใน (Internal DTD) เป็นการประกาศโดยการแทรก DTD ในเอกสาร XML แบบที่สองประกาศแบบภายนอก (External DTD) เป็นการประกาศโดยการแยกไฟล์ออกมาต่างหาก ตัวอย่างในรูปที่ 2.3 เป็นการประกาศแบบภายนอก เพราะแยก DTD เป็นไฟล์ต่างหาก คือ po.dtd

การบอกประเภทของเอกสาร XML (document type declaration) ต่างกับการกำหนดโครงสร้างของ XML แบบ DTD (Document Type Definition) เนื่องด้วยเพราะ DTD เป็นส่วนหนึ่งของการบอกประเภทของเอกสาร XML

2.1.2.3 ข้อความคอมเมนต์ (comment)

การเขียนข้อความที่เป็นคอมเมนต์ของเอกสาร XML ข้อความจะอยู่ภายในเครื่องหมาย <!-- และ -->

ข้อห้ามของการเขียนข้อความคอมเมนต์มีข้อกำหนดดังนี้

1 ห้ามเขียนข้อความคอมเมนต์ส่วนบนของเอกสาร เพราะถูกกำหนดไว้ให้กับการบอกประเภทของเอกสาร XML

2 ห้ามแทรกข้อความคอมเมนต์อยู่ในแท็ก ตัวอย่างแสดงดังรูปที่ 2.4

```
<Element <!- - ข้อความคอมเมนต์ - -> > Value </Element>
```

รูปที่ 2.4 ตัวอย่างข้อห้ามการแทรกข้อความคอมเมนต์ในแท็ก

3 ห้ามมีเครื่องหมายลบ 2 ตัวติดกัน (--) ซึ่งจะทำให้มีลักษณะเหมือนกับเครื่องหมายเปิด-ปิดของคอมเมนต์ เพราะจะทำให้ตัวแปลภาษา XML ทำงานสับสนและเกิดข้อผิดพลาดได้

ประโยชน์ของคอมเมนต์โดยพื้นฐานก็คือ เป็นข้อความสำหรับให้ใครอ่านก็ได้ อาจเป็นข้อความเพื่อเน้นย้ำ แนะนำ หรือเตือนความจำ เป็นต้น โดยที่ตัวแปลภาษา XML จะละเว้นข้อความคอมเมนต์ไว้ ไม่แปลเหมือนข้อความอื่นๆ ในเอกสาร XML และสามารถอาศัยคุณสมบัติของคอมเมนต์มาช่วยตัดงานบางส่วนออกไปชั่วคราวโดยไม่ต้องลบทิ้ง เพื่อมิให้ตัวแปลภาษา XML แปลงานในส่วนนั้น

2.1.2.4 การแทรกส่วนประมวลผลด้วย Processing Instruction: PI

การแทรกส่วนประมวลผลด้วย Processing Instruction เป็นส่วนที่โปรแกรมตัวแปลภาษา XML จะต้องแปลหรือประมวลผล ข้อความส่วนที่เป็น Processing Instruction ขึ้นต้นด้วยเครื่องหมาย <? และลงท้ายด้วย ?> รูปที่ 2.5 แสดงตัวอย่างการแทรกส่วนประมวลผลด้วย Processing Instruction

```
<? xml-stylesheet type = "text/css" href = "mystyle.css" ?>
```

รูปที่ 2.5 ตัวอย่างการแทรกส่วนประมวลผลด้วย Processing Instruction

รูปที่ 2.5 เป็นตัวอย่างการแทรกส่วนประมวลผลด้วย Processing Instruction โดย Processing Instruction จะบอกโปรแกรมตัวแปลภาษา XML ที่ฝังอยู่ใน Internet Explorer ให้ใช้สไตล์ชีทจากไฟล์ชื่อ mystyle.css มากำหนดรูปแบบการแสดงผลเอกสาร XML ทางเบราว์เซอร์

2.1.3 กฎพื้นฐานของภาษา XML

กฎของภาษา XML เบื้องต้น ได้แก่ กฎไวยากรณ์ของอิลิเมนต์ (element) โดยอิลิเมนต์มีความหมายครอบคลุมตั้งแต่ตัวแท็ก (tag) และข้อมูลที่อยู่ภายในแท็ก ขณะที่แท็กหมายถึงส่วนที่เป็นแท็กจริงๆ ไม่รวมข้อมูลภายในแท็กนั้น ไวยากรณ์ที่เกี่ยวกับอิลิเมนต์ของภาษา XML ประกอบด้วย

1 เอกสาร XML หนึ่งๆ จะมีรูตอิลิเมนต์ (root element) ได้เพียงรูตเดียว ซึ่งจะทำหน้าที่คลุมอิลิเมนต์อื่นๆ ทั้งหมด รูปที่ 2.6 ตัวอย่าง <Paper> รูตอิลิเมนต์

```
<Paper>
    <title> XML Database </title>
    <Author> Kanokwan and Kittisak </Author>
</Paper>
```

รูปที่ 2.6 ตัวอย่าง <Paper> รูตอิลิเมนต์

2 แท็กเปิดและแท็กปิดต้องเหมือนกัน แตกต่างกันเพียงแท็กปิดต้องมีเครื่องหมายแสลช (/) นำหน้าชื่อแท็ก

3 กระระบุแท็กห้ามมีการเหลื่อมกัน (overlap) หมายถึงแท็กที่เปิดก่อนต้องปิดทีหลัง รูปที่ 2.7 แสดงตัวอย่างการระบุแท็กที่เหลื่อมกัน

```
<book><title> XML Database </book></title>
```

รูปที่ 2.7 ตัวอย่างการระบุแท็กที่เหลื่อมกัน

4 ชื่อแท็กมีคุณสมบัติเป็น case-sensitive คือตัวอักษรพิมพ์เล็ก ตัวอักษรพิมพ์ใหญ่จะมีความแตกต่างกัน เช่น แท็ก <City> แท็ก <CITY> และแท็ก <city> จะเป็นคนละแท็กกัน

5 แท็กว่าง (empty tag) หมายถึงแท็กที่ไม่มีข้อมูลอยู่ภายในแท็ก รูปแบบการเขียนแท็กว่างเขียนได้ 2 รูปแบบ แบบที่หนึ่ง <title/> แบบที่สอง <title></title>

6 ค่าของแอตทริบิวต์ต้องอยู่ในเครื่องหมายคำพูดแบบดับเบิลโควต (") หรือซิงเกิลโควต (') อย่างใดอย่างหนึ่ง เช่น <book ISBN = "974-86631-0-3" /> จากตัวอย่างนี้ยังแสดงให้เห็นว่าอิลิเมนต์ book เป็นอิลิเมนต์ว่างหรือแท็กว่างซึ่งมีอิลิเมนต์ได้เหมือนกับอิลิเมนต์หรือแท็กปกติทุกอย่าง

กรณีที่ในค่าของแอตทริบิวต์มีเครื่องหมายคำพูดแบบดับเบิลโควตหรือซิงเกิลโควตอยู่ด้วย จะต้องหลีกเลี่ยงการใช้เครื่องหมายที่แตกต่างกัน เช่น <link Onclick = " location.href = 'main.xml' "> หรือ <link Onclick = 'location.href = "main.xml" '>

7 คำสงวนในภาษา XML ประกอบด้วย 5 อักขระ ได้แก่ <, &, >, " และ ' ซึ่งอักขระทั้ง 5 ตัวนี้ ถูกสงวนไว้เพื่อใช้เป็นส่วนประกอบตามโครงสร้างของภาษา ดังนั้นหากเนื้อความเอกสารจำเป็นต้องมีอักขระที่เป็นคำสงวน ก็จะต้องระบุเป็นชุดอักษรพิเศษ (entity reference) แทนคำสงวนทั้ง 5 ตัว การระบุชุดอักษรพิเศษแทนคำสงวนทั้ง 5 ตัว แสดงได้ดังตารางที่ 2.1

ตารางที่ 2.1 คำสงวนและอักษรพิเศษ

คำสงวน	อักษรพิเศษ
<	<
&	&
>	>
"	"
'	'

ตัวอย่างการใช้อักษรพิเศษ เช่น หากต้องการให้ข้อความ คุกกี้ราคา < 25 บาททุกชิ้นที่ S&P. อยู่ในแท็ก <promotion> ก็จะต้องเขียนดังรูปที่ 2.8

```
<promotion> คุกกี้ราคา &lt; 25 บาททุกชิ้นที่ S&amp;P. </promotion>
```

รูปที่ 2.8 ตัวอย่างการใช้อักษรพิเศษ

8 การตั้งชื่อแท็กมีหลักเกณฑ์ดังนี้

- ชื่อแท็กต้องขึ้นต้นด้วยตัวอักษรหรือเครื่องหมายอันเดอร์สกอร์ (_) เท่านั้น
- อักษรตัวถัดไปต้องเป็นตัวอักษร ตัวเลข เครื่องหมายจุด (.) เครื่องหมายยัติภังค์ (-) เครื่องหมายอันเดอร์สกอร์ (_) หรือเครื่องหมายโคลอน (:) เท่านั้น แต่สำหรับเครื่องหมายโคลอนจะมีปัญหาเรื่องเนมสเปซ ดังนั้นควรจะหลีกเลี่ยงการใช้
- ชื่อแท็กมีคุณสมบัติ case sensitive ดังได้กล่าวไว้ในไวยากรณ์ที่ 4
- อักษร 3 ตัวแรกของแท็กห้ามเป็นค่า XML ทั้งตัวเล็กและตัวใหญ่

ตัวอย่างชื่อแท็กที่ถูกต้องแสดงดังรูปที่ 2.9 และตัวอย่างแท็กที่ไม่ถูกต้องแสดงได้ดังตารางที่ 2.2

```
<example />
<_example />
<example_ex />
<example_ex-ex />
<_example.ex />
<example.ex />
<example1 />
<example_2 />
```

รูปที่ 2.9 ตัวอย่างชื่อแท็กที่ถูกต้อง

ตารางที่ 2.2 ตัวอย่างแท็กที่ไม่ถูกต้อง

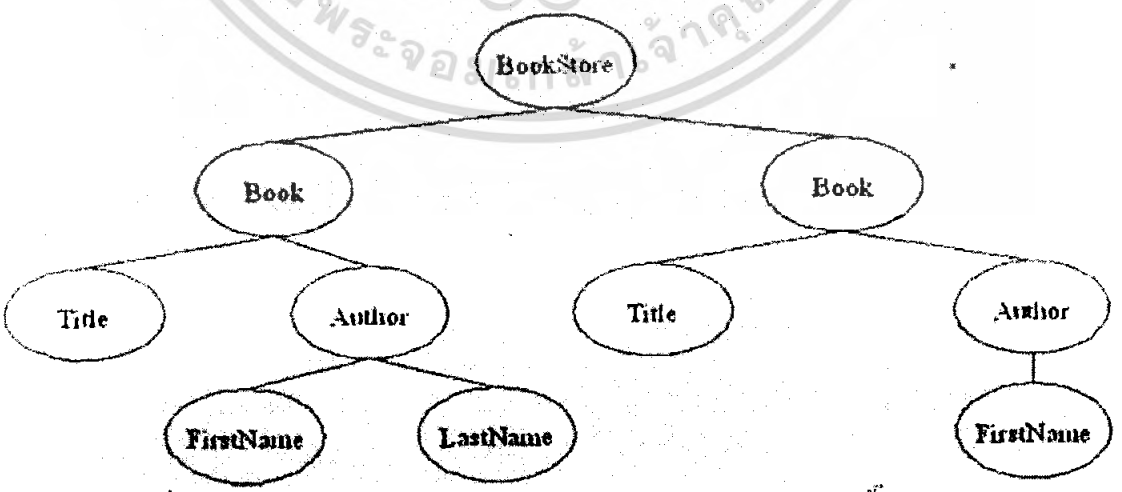
ชื่อแท็กที่ไม่ถูกต้อง	คำอธิบายสาเหตุที่ไม่ถูกต้อง
<-example />	ขึ้นต้นด้วยเครื่องหมายยัติภังค์
<1example />	ขึ้นต้นด้วยตัวเลข
<.example />	ขึ้นต้นด้วยเครื่องหมายจุด
<:example />	ขึ้นต้นด้วยเครื่องหมายโคลอน
<?example />	ขึ้นต้นด้วยเครื่องหมายคำถาม
<example? />	มีเครื่องหมายคำถาม
<example* />	ขึ้นต้นด้วยเครื่องหมายดอกจัน
<xmlexample />	ขึ้นต้นด้วย xml
<XMLexample />	ขึ้นต้นด้วย XML
<Xmlexample />	ขึ้นต้นด้วย Xml

จากโครงสร้างเอกสาร XML และกฎพื้นฐานของภาษา XML ดังอธิบายข้างต้น สามารถยกตัวอย่างเพื่อประกอบความเข้าใจ ดังนี้

```
<?xml version="1.0"?>
<BookStore>
    <Book category='Computer'>
        <Title>XML for Beginner</Title>
        <Author>
            <FirstName>John</FirstName>
            <LastName>Smith</LastName>
        </Author>
    </Book>
    <Book category='Math'>
        <Title>Calculus I</Title>
        <Author>
            <FirstName>James</FirstName>
        </Author>
    </Book>
</BookStore>
```

รูปที่ 2.10 ตัวอย่างเอกสาร XML sample.xml

จากรูปที่ 2.10 สามารถนำเอกสาร XML sample.xml มาแสดงในลักษณะของลำดับชั้นต้นไม้ได้ดังรูปที่ 2.11



รูปที่ 2.11 โครงสร้างเอกสาร XML sample.xml ในลักษณะลำดับชั้นต้นไม้

2.1.4 มาตรฐานที่เกี่ยวข้องกับภาษา XML

มาตรฐานเสริมศักยภาพของการทำงานของ XML ที่ถูกเสนอต่อ W3C จำนวนมาก ได้แก่ XSL, Xlink และ Xpointer ซึ่งต่างก็เป็นมาตรฐานที่ถูกนำเสนอเพื่อสนับสนุน XML ในด้านของสไตล์ชีท (style sheet) การทำไฮเปอร์ลิงค์ (hyperlinks) และคุณสมบัติด้านอื่นๆ นอกจากนั้นยังมีมาตรฐานอีกหลายรายการที่ได้รับการปรับปรุงสนับสนุน XML

มาตรฐานต่างๆ ที่เกี่ยวข้องกับ XML ที่ได้ถูกเสนอให้ W3C รับรอง ตัวอย่างเช่น

2.1.4.1 เนมสเปซ (namespace)

เนมสเปซทำหน้าที่ช่วยในการกำหนดขอบเขตให้กับชื่อของอิลิเมนต์และแอตทริบิวต์ เพื่อหลีกเลี่ยงการถูกเรียกใช้ซ้ำกันระหว่างเอกสาร XML ที่เกิดจากแหล่งกำเนิดต่างกัน เนมสเปซแต่ละชุดเป็นการรวมของชื่อต่างๆ ของอิลิเมนต์และแอตทริบิวต์ของ XML ที่ถูกระบุพร้อมกับ URI (Uniform Resource Identifier) ที่เป็นแหล่งกำเนิดของเอกสาร XML นั้น

2.1.4.2 XSL (Extensible Stylesheet Language)

XSL เป็นภาษาในการกำหนดสไตล์ชีทให้กับข้อมูล XML (XML data) สำหรับการนำเสนอในเวปเบราว์เซอร์หรือสื่อนำอื่นๆ XSL มีศักยภาพสูงกว่า CSS (Cascading Stylesheet) ที่ใช้เป็นสไตล์ชีทสำหรับ HTML มาก

2.1.4.3 XLink

XLink มีพื้นฐานการทำงานมาจาก HyTime และ XLink เป็นการปรับปรุงการเชื่อมโยงใน HTML โดยสนับสนุนการเชื่อมโยงแบบสองทาง (bi-direction) แบบหนึ่งต่อกลุ่ม (one-to-many) และแบบแยกตามประเภท (typed links) เช่น การอ้างอิงเชิงอรรถ (footnote) และอภิธานศัพท์ (glossary) เป็นต้น

2.1.4.4 XPointer

XPointer มีพื้นฐานการทำงานมาจาก Text Encoding Intiative (TEI) XPointer อนุญาตให้สามารถชี้ไปที่จุดหรือตำแหน่งใดก็ได้ในเอกส่รเป้าหมาย โดยไม่ต้องกำหนดการเชื่อมโยงแบบ HTML ไว้ล่วงหน้า

2.1.4.5 RDF (Resource Description Framework)

RDF เป็นเมตะเดตา (metadata) สำหรับเอกสาร XML คล้ายกับเมตะเดตาแท็ก (metadata tag) ของ HTML เช่น ผู้แต่ง ลิขสิทธิ์ และวันที่จัดพิมพ์ เป็นต้น

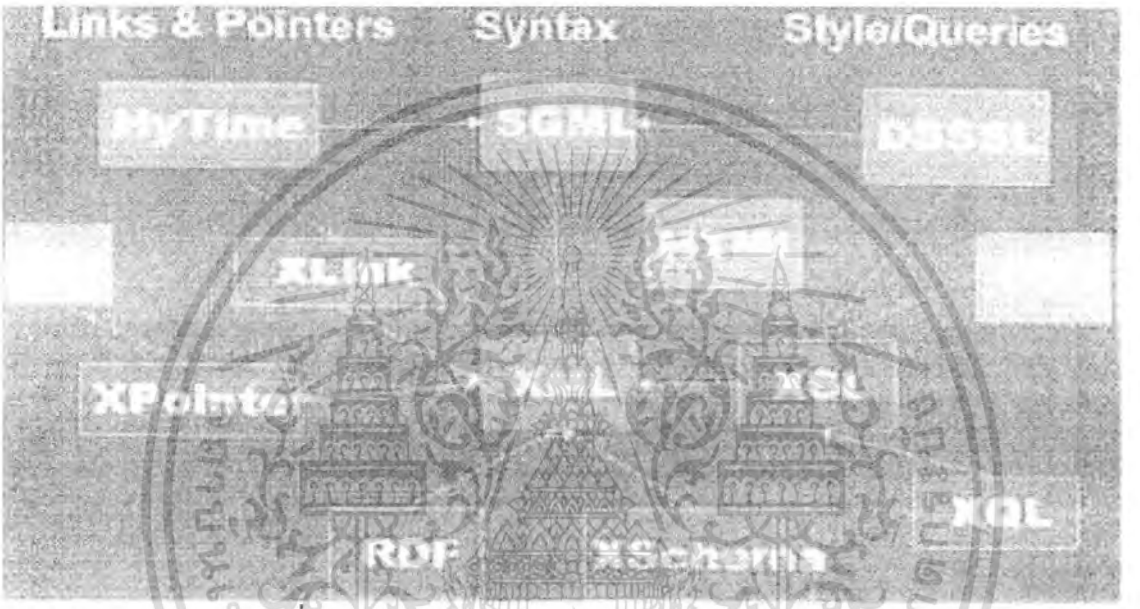
2.1.4.6 XSchema

XSchema เข้ามาแทนที่ DTD สำหรับการกำหนดประเภทข้อมูลของ XML XSchema เป็นการประยุกต์หลายเทคโนโลยีเข้าด้วยกัน ได้แก่ XData, SOX, DCD (Document Content Description) และ DDML

2.1.4.7 XQL (XML Query Language)

XQL เป็นส่วนขยายเพิ่มเติมของ XSL สำหรับใช้อ้างถึงและกลั่นกรองส่วนที่เป็นอิลิเมนต์และแท็กของเอกสาร XML XQL ช่วยกำหนดรูปแบบที่ชัดเจน เข้าใจง่ายในการเข้าถึงอิลิเมนต์ที่เฉพาะเจาะจง และสำหรับการค้นหาโหนด (node) ที่มีคุณลักษณะพิเศษต่างๆ

ความสัมพันธ์ระหว่าง 7 มาตรฐานของ XML แสดงได้ดังรูปที่ 2.12



รูปที่ 2.12 ความสัมพันธ์ระหว่าง 7 มาตรฐานของ XML

นอกจากมาตรฐานทั้ง 7 ของ XML ที่กล่าวมานั้น ยังมีการกำหนดมาตรฐานด้าน Application Programming Interface (API) เพื่อที่จะให้แอพลิเคชันสามารถเข้าถึงเอกสาร XML ได้ทันที เพื่อเอกสาร XML เข้าสู่กระบวนการ parsing โดยมี API 2 กลุ่มที่ได้รับความนิยม ได้แก่

2.1.4.8 DOM (Document Object Model)

DOM เป็น tree-based API ที่ทำการประมวลผลโครงสร้างเอกสาร XML ให้เป็นโครงสร้างแบบต้นไม้ที่ประกอบไปด้วยลำต้น กิ่งก้านสาขา และใบ เพื่อให้แอพลิเคชันสามารถเข้าหาจุดต่างๆ ของโครงสร้างต้นไม้นั้นได้ โดยที่ขอบเขตจะถูกจำกัดโดยหน่วยความจำที่เรียกใช้ได้ในขณะนั้น นั่นคือ DOM เป็นกระบวนการดึงข้อมูลในเอกสาร XML มาใช้งานใน Application โดยมีหลักการคือ จะนำข้อมูลจากเอกสาร XML ทั้งเอกสารมาวางเป็นโครงสร้างลักษณะลำดับชั้นต้นไม้ในหน่วยความจำของเครื่องคอมพิวเตอร์ แล้วใช้วิธีการท่องไปในต้นไม้เพื่อดึงข้อมูลมา ซึ่งวิธีการโหลดข้อมูลทั้ง

เอกสาร XML เข้ามาไว้ในหน่วยความจำนั้นทำให้เกิดเป็นข้อเสียอย่างหนึ่งของ DOM เพราะจะกลายเป็นข้อจำกัดในเรื่องของหน่วยความจำในกรณีที่เอกสาร XML มีขนาดใหญ่มากๆ ก็จะต้องเปลืองเนื้อที่ในหน่วยความจำมาก และอาจจะไม่มีเนื้อที่เหลือพอที่จะทำงาน

2.1.4.9 SAX (Simple API for XML)

SAX เป็น event-based API ที่รายงานตามเหตุการณ์ของ parsing เช่น จุดเริ่มต้นและจุดสิ้นสุดของอิลิเมนต์ต่างๆ ไปให้แอพลิเคชันผ่านทางการร้องขอ โดยปกติจะไม่มีการสร้างโครงสร้างต้นไม้ขึ้นมา เนื่องจากการทำงานเป็นแบบ event-based API การเข้าถึงเอกสาร XML จึงทำได้ง่าย ไม่ซับซ้อน และที่สำคัญผู้ใช้สามารถทำ parsing เอกสารที่มีขนาดใหญ่มากกว่าปริมาณหน่วยความจำที่เรียกใช้ได้ และผู้ใช้สามารถสร้างดครงสร้างข้อมูลของเอกสารอยู่ในรูปแบบที่ตนเองต้องการได้

2.1.5 ประโยชน์ของภาษา XML

ประโยชน์ของ XML ที่สำคัญคือการเป็นแม่แบบหรือต้นแบบในการนิยามข้อมูลเพื่อใช้ในงานต่างๆ ซึ่งสามารถแบ่งรายละเอียดได้ดังนี้

1 ใช้สำหรับสร้างข้อมูลที่สามารถอธิบายความหมายของตัวเองได้ (self-describe data)

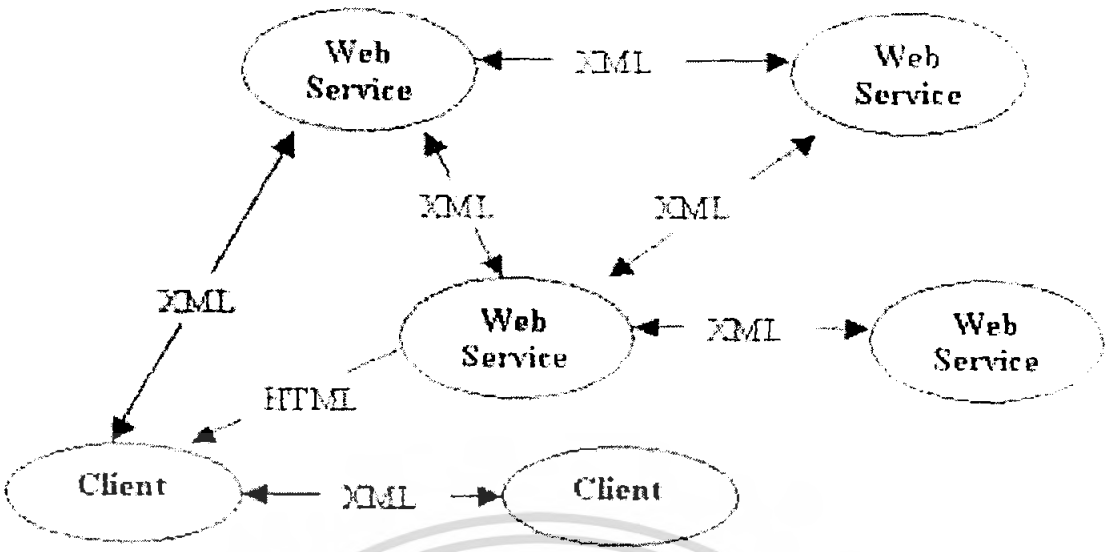
จากความสามารถในการสร้างแท็กขึ้นมาอธิบายข้อมูลที่อยู่ภายในแท็ก ทำให้ข้อมูลนั้นมีความหมายอยู่ในตัวมันเอง เป็นข้อมูลที่สามารถเขียนโปรแกรมมาดึงข้อมูลไปใช้งานได้โดยง่าย และแม้นมนุษย์ก็สามารถอ่านได้ ดังนั้นจึงสามารถกล่าวได้ว่าเอกสาร XML มีคุณสมบัติครบทั้งแบบ machine readable และแบบ human readable

2 ใช้สำหรับการแลกเปลี่ยนข้อมูล (data exchange)

เนื่องด้วยเอกสาร XML มีลักษณะเป็นไฟล์ข้อความธรรมดา ดังนั้นจึงทำให้เอกสาร XML เป็นภาษากลางที่สามารถใช้ได้ในทุกแพลตฟอร์ม (platform) เช่น แพลตฟอร์มวินโดว์ส ยูนิกซ์ หรืออื่นๆ ดังนั้นจึงทำให้เอกสาร XML มีความสามารถในการแลกเปลี่ยนเอกสารข้ามแพลตฟอร์มกันได้

3 เป็นรูปแบบข้อความในการสื่อสาร (messaging format) ระหว่างแอพลิเคชันหรือโปรแกรม

เป็นแนวคิดที่กำลังได้รับความนิยมเป็นอย่างมากสำหรับประโยชน์ของ XML ข้อนี้เนื่องด้วยตั้งแต่ปี ค.ศ. 2000 เป็นต้นมา แนวคิดนี้เป็นแนวคิด web service ที่บริษัทไมโครซอฟต์เรียกว่า .NET ซึ่งแนวความคิดนี้ก็คือการเตรียมซอฟแวร์สำหรับให้บริการทำงานบางอย่างอยู่ในเซิร์ฟเวอร์เครื่องใดเครื่องหนึ่งภายในเครือข่ายอินเตอร์เน็ต โดยผู้ใช้งานทั่วไปสามารถเรียกใช้บริการนั้นได้ ดังรูปที่ 2.13 ซึ่งจะเห็นได้ว่า XML เป็นรูปแบบการสื่อสารระหว่างองค์ประกอบต่างๆ ตามแนวคิดของ web service



รูปที่ 2.13 ข้อความที่ใช้สื่อสารระหว่างองค์ประกอบของ web service มีรูปแบบเป็น XML

4 ประโยชน์ในเชิงเทคโนโลยีอินเตอร์เน็ตและการพัฒนาเวป

ประโยชน์ในเชิงเทคโนโลยีอินเตอร์เน็ตและการพัฒนาเวปในเมืองไทยยังไม่เห็นประโยชน์มากนัก เพราะลักษณะงานยังไม่มีความจำเป็นให้นำ XML ไปพัฒนา แต่งานบางอย่าง เช่น การพัฒนาเวป สามารถที่จะนำ XML มาช่วยเพิ่มประสิทธิภาพได้

5 เป็นรากฐานของภาษาใหม่ๆ ในการพัฒนาเวป

ภาษาใหม่ๆ ในการพัฒนาเวป เช่น ภาษา XHTML, MathML, VML, WML เป็นต้น

6 ใช้ในแวดวงธุรกิจแบบ B2B (Business to Business)

ในกรณีนี้จะต้องใช้ภาษาเฉพาะอย่าง เช่น cXML (Commerce XML), xCML (XML Common Business Language) เป็นต้น

และในต่างประเทศมีโครงการนำ XML มาใช้แทนระบบ EDI (Electronic Data Interchange) ซึ่งเป็นการแลกเปลี่ยนเอกสารอิเล็กทรอนิกส์ระหว่างองค์กร

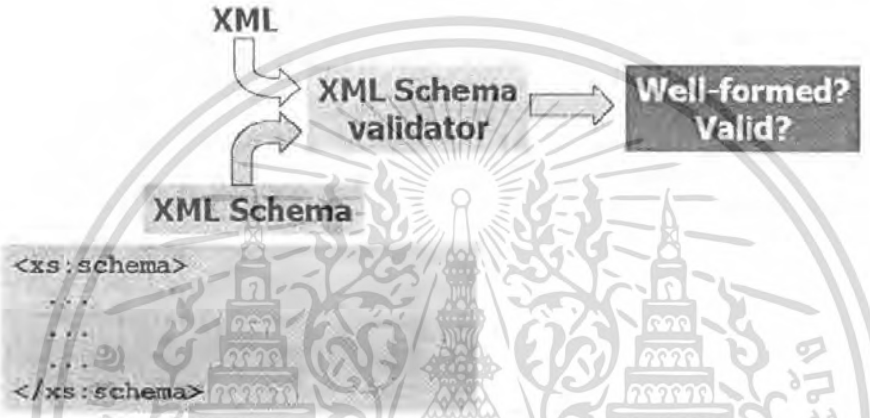
2.2 ความถูกต้องของเอกสาร XML

อันเนื่องมาจากการที่ภาษา XML ยอมให้สร้างแท็กขึ้นมาเองได้ ดังนั้นจึงจำเป็นต้องมีมาตรฐานที่ใช้วัดความถูกต้องของเอกสาร XML ขึ้นมา มี 2 ประเภท ได้แก่

2.2.1 Well-formed XML คือ เอกสาร XML ที่มีโครงสร้างถูกต้อง โดยมีลักษณะดังที่อธิบายในหัวข้อกฎพื้นฐานของภาษา XML ซึ่งจะเห็นว่า Well-formed XML นั้น ไม่ได้บอกว่าข้อมูลมีความถูกต้องและตรงตามความต้องการที่จะนำไปใช้หรือไม่ จึงต้องมีการ Validation ดังจะกล่าวต่อไปนี้

2.2.2 Valid XML คือ เอกสาร XML ที่มีโครงสร้างข้อมูลถูกต้องและครบถ้วน ซึ่งใช้ XML Schema หรือ DTD (Document Type Definition) ในการอธิบายตัวข้อมูล เช่น องค์ประกอบของข้อมูล, ชนิดของข้อมูล และอื่น ๆ เพื่อนำมาทำการ Validation เอกสาร XML ดังตัวอย่างในรูปที่ 2.14 และ 2.15

```
<location>
  <latitude>8.9</latitude>
  <longitude>60.5</longitude>
  <altitude>2000</altitude>
</location>
```



รูปที่ 2.14 การตรวจสอบความถูกต้องในเอกสาร XML

ในรูปที่ 2.15 จะเห็นได้ว่า

- ข้อมูลใน XML # 1 นั้นไม่ครบถ้วน
- ข้อมูลใน XML # 2 เท่านั้นที่อธิบาย <location> ได้ครบถ้วนและถูกต้อง
- ข้อมูล XML ทั้งสองต่างเป็น Well-formed XML แต่ XML # 2 เป็น "Valid XML data"

1
```
<location>
  <latitude>8.9</latitude>
  <longitude>60.5</longitude>
</location>
```

2
```
<location>
  <latitude>8.9</latitude>
  <longitude>60.5</longitude>
  <altitude>2000</altitude>
</location>
```

รูปที่ 2.15 เอกสาร XML ที่เป็น Well-formed XML และ Valid XML

2.3 การกำหนดโครงสร้างเอกสาร XML แบบ DTD และ XML Schema

การกำหนดโครงสร้างหรือสกีมา (schema) ของเอกสาร XML เป็นการระบุว่าเอกสาร XML มีโครงสร้างอย่างไร มีอิลิเมนต์ มีแอตทริบิวต์อะไรบ้าง และมีชนิดข้อมูลของอิลิเมนต์ แอตทริบิวต์อย่างไร เอกสาร XML ที่ถูกกฎพื้นฐานนั้นเรียกว่าเอกสาร XML ที่มีคุณสมบัติ well-formed แต่เอกสาร XML ที่มีความน่าเชื่อถือนั้นต้องเป็นเอกสาร XML ที่มีการกำหนดโครงสร้างเอกสาร XML ที่มีการอ้างอิงโครงสร้างเป็นเอกสาร XML ที่มีคุณสมบัติ valid การกำหนดโครงสร้างของเอกสาร XML นั้นมีประโยชน์สำคัญ 2 ประการ คือ

1 เป็นการกำหนดรูปแบบโครงสร้างของเอกสาร XML หรือเรียกว่าแบบจำลองข้อมูล (data model) ซึ่งเป็นการกำหนดว่าเอกสาร XML มีแท็กอะไรบ้าง มีแอตทริบิวต์อะไรบ้าง ค่าของแอตทริบิวต์เป็นอะไรบ้าง เป็นต้น

2 เป็นสัญญาระหว่างองค์กรที่ใช้เอกสาร XML ในการแลกเปลี่ยนข้อมูล เพื่อให้รูปแบบของเอกสาร XML ที่ใช้มีรูปแบบที่ตรงกัน และเพื่อความถูกต้องในการนำเอกสาร XML ไปใช้งาน

การกำหนดโครงสร้างของเอกสาร XML ทำได้หลายวิธี เช่น DCD (Document Content Description), RELAX/TREX, BizTalk, XDR (XML-Data Reduced), DTD และ XML Schema เป็นต้น แต่ที่กล่าวถึงในโครงงานนี้มี 2 วิธี คือ แบบ DTD และแบบ XML Schema ซึ่งจะกล่าวในหัวข้อ 2.3.1 และ 2.3.2 ตามลำดับ ซึ่งทั้ง 2 แบบนี้ถูกเผยแพร่โดยสมาคม W3C

2.3.1 การกำหนดโครงสร้างเอกสาร XML แบบ DTD

การวางโครงสร้างของเอกสาร XML แบบ DTD นั้นสืบทอดมาจากภาษา SGML ปัจจุบันมีแนวโน้มของการใช้งานน้อยลง แต่ก็ควรศึกษาเพื่อเป็นพื้นฐานในการออกแบบโครงสร้างของเอกสาร XML เพราะมีบางเทคโนโลยียังใช้งาน DTD อยู่บ้าง เช่น เทคโนโลยี WAP (Wireless Application Protocol) ซึ่งเป็นการเรียกดูข้อมูลอินเตอร์เน็ตผ่านโทรศัพท์มือถือ ถูกสร้างด้วยภาษา WML (Wireless Markup Language) เป็นภาษาที่อยู่ในตระกูลของภาษา XML แต่มีการกำหนดโครงสร้างแบบ DTD

2.3.1.1 ลักษณะของการกำหนดโครงสร้างเอกสาร XML แบบ DTD

ลักษณะของการกำหนดโครงสร้างเอกสาร XML แบบ DTD แสดงดังรูปที่ 2.16

```
<!ELEMENT BookCataloge (Book)*>
<!ELEMENT Book
(Title,Author,Date,ISBN,Publisher)*>
<!ELEMENT Title(#PCDATA)>
<!ELEMENT Author(#PCDATA)>
<!ELEMENT Date(#PCDATA)>
<!ELEMENT ISBN(#PCDATA)>
<!ELEMENT Publisher(#PCDATA)>
```

รูปที่ 2.16 ลักษณะของเอกสาร DTD

จากการกำหนดโครงสร้างของเอกสาร XML ในรูปที่ 2.16 ทำให้เราสามารถสร้างเอกสาร XML ที่อ้างอิงกับโครงสร้างได้ดังรูปที่ 2.17

```
<?xml version = "1.0">
<BookCataloge>
        <Book>
                <Title>Conceptual Schema Relational Database Design</Title>
                <Author>Terry Helpin</Author>
                <Date>1 Febuary 1995</Date>
                <ISBN>0 13 355702 2</ISBN>
                <Publisher>Hall of Australia</Publisher>
        </Book>
</BookCataloge>
```

รูปที่ 2.17 เอกสาร XML ที่อ้างอิงกับโครงสร้างแบบ DTD (รูปที่ 2.16)

2.3.1.2 การประกาศโครงสร้างเอกสารแบบ DTD

สามารถทำได้สองวิธีคือ การประกาศแบบภายใน (Internal DTD) และการประกาศแบบภายนอก (External DTD) การประกาศแบบภายในนั้นจะประกาศไว้ในส่วนของ Document Type Declaration ซึ่งเป็นส่วนของเอกสาร XML การประกาศโครงสร้างของเอกสาร DTD แบบภายในแสดงได้ดังรูปที่ 2.18 (บริเวณที่เป็นตัวหนา)

```
<?xml version = "1.0">
<!DOCTYPE BookCataloge [
        <!ELEMENT BookCataloge (Book)*>
        <!ELEMENT Book (Title,Author,Date,ISBN,Publisher)*>
        <!ELEMENT Title(#PCDATA)>
        <!ELEMENT Author(#PCDATA)>
        <!ELEMENT Date(#PCDATA)>
        <!ELEMENT ISBN(#PCDATA)>
        <!ELEMENT Publisher(#PCDATA)>
]>
<BookCataloge>
        <Book>
                <Titile>Conceptual Schema Relational Database Design</Titile>
                <Author>Terry Helpin</Author>
                <Date>1 Febuary 1995</Date>
                <ISBN>0 13 355702 2</ISBN>
                <Publisher>Hall of Australia</Publisher>
        </Book>
</BookCataloge>
```

รูปที่ 2.18 การประกาศโครงสร้างเอกสาร DTD แบบภายใน

การประกาศโครงสร้างเอกสาร DTD แบบภายนอกจะกระทำโดยแยกเป็นไฟล์ต่างหาก โดยไม่รวมกับส่วนของเอกสารเอ็กซ์แอล เพียงแต่มีการอ้างอิงถึงเท่านั้น ไฟล์ของโครงสร้างเอกสารจะมีนามสกุล (.dtd) แสดงดังรูปที่ 2.19

```
<?xml version = "1.0">
<!DOCTYPE BookCataloge SYSTEM "bookcataloge.dtd">
<BookCataloge>
        <Book>
                <Titile>Conceptual Schema Relational Database Design</Titile>
                <Author>Terry Helpin</Author>
                <Date>1 Febuary 1995</Date>
                <ISBN>0 13 355702 2</ISBN>
                <Publisher>Hall of Australia</Publisher>
        </Book>
</BookCataloge>
```

รูปที่ 2.19 การประกาศโครงสร้างเอกสาร DTD แบบภายนอก

2.3.1.3 การประกาศ อิลิเมนต์ และ แอตทริบิวต์

การกำหนดโครงสร้างของเอกสาร XML แบบ DTD สิ่งที่ผู้กำหนดจำเป็นต้องทราบมีอยู่สองประเภทหลักๆ คือการประกาศ อิลิเมนต์ และแอตทริบิวต์

1 การประกาศอิลิเมนต์

ภายในเอกสาร XML จะประกอบด้วยอิลิเมนต์จำนวนมากตั้งแต่รูตอิลิเมนต์และอิลิเมนต์ต่างๆ ประกอบกันเป็นเอกสาร XML ซึ่งการประกาศอิลิเมนต์จึงถือว่าเป็นสิ่งสำคัญที่จะต้องกล่าวถึง รูปแบบการประกาศอิลิเมนต์แสดงดังรูปที่ 2.20

```
<!ELEMENT ชื่ออิลิเมนต์        (เนื้อหาของอิลิเมนต์ )>
```

รูปที่ 2.20 รูปแบบการประกาศอิลิเมนต์

ชื่ออิลิเมนต์นั้นมีข้อกำหนดตามที่กล่าวไว้ในหัวข้อกฎพื้นฐานของ XML ส่วนเนื้อหาของ XML มีโอกาศเป็นไปได้คือ เนื้อหาของอิลิเมนต์ประกอบด้วยอิลิเมนต์อื่นๆ เนื้อหาอิลิเมนต์ประกอบด้วยข้อความปกติ และเนื้อหาของอิลิเมนต์อาจไม่มีอะไรอยู่เลย (เป็นแท็กว่าง) หรือในอิลิเมนต์จะประกอบด้วยอิลิเมนต์ปนอยู่กับข้อความก็ได้จากที่กล่าวมาสามารถแสดงตัวอย่างได้ดังรูปที่ 2.21 – 2.28

```
<!ELEMENT ชื่ออิลิเมนต์ (อิลิเมนต์1,อิลิเมนต์2,อิลิเมนต์3,...,อิลิเมนต์ n)>
```

รูปที่ 2.21 รูปแบบการประกาศประกาศอิลิเมนต์ที่ภายในประกอบด้วยอิลิเมนต์

```
<Book>
        <Titile>Conceptual Schema Relational Database Design</Titile>
        <Author>Terry Helpin</Author>
        <Date>1 Febuary 1995</Date>
        <ISBN>0 13 355702 2</ISBN>
        <Publisher>Hall of Australia</Publisher>
</Book>
```

รูปที่ 2.22 อิลิเมนต์ที่ประกอบด้วยอิลิเมนต์อื่นๆ อยู่ภายใน

จากรูปที่ 2.22 จะแสดงให้เห็นว่าอิลิเมนต์ Book จะประกอบด้วยอิลิเมนต์อื่นๆ ภายในเช่น Title, Authour, Date, ISBN และ Publisher

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

รูปที่ 2.23 รูปแบบการประกาศอิลิเมนต์ที่ภายในประกอบด้วยข้อความ

รูปที่ 2.23 แสดงรูปแบบการประกาศอิลิเมนต์ที่ภายในประกอบด้วยข้อความธรรมดา โดยที่ PCDATA (Parsed Character Data) เป็นชนิดของข้อมูลของโครงสร้างของเอกสาร XML แบบ DTD

```
<Title>Conceptual Schema Relational Database Design</Title>
```

รูปที่ 2.24 อิลิเมนต์ที่ภายในประกอบด้วยข้อความ

```
<!ELEMENT ชื่ออิลิเมนต์ EMPTY>
```

รูปที่ 2.25 รูปแบบการประกาศอิลิเมนต์ว่าง

```
<Student ID="47010005">
```

รูปที่ 2.26 อิลิเมนต์ที่ไม่มีอะไรอยู่ หรือแท็กว่าง

รูปที่ 2.25 อิลิเมนต์ที่ภายในไม่มีข้อความ ไม่มีอิลิเมนต์ มีเพียงแอตทริบิวต์ อิลิเมนต์ลักษณะนี้เป็นอิลิเมนต์ว่าง

```
<!ELEMENT ชื่ออิลิเมนต์ (#PCDATA,อิลิเมนต์1,อิลิเมนต์2,อิลิเมนต์3,...,อิลิเมนต์ n)>
```

รูปที่ 2.27 รูปแบบการประกาศอิลิเมนต์ปนกับข้อความ

```
<Book>
        This is essential book for conceptual database design
        <Titile>Conceptual Schema Relational Database Design</Titile>
</Book>
```

รูปที่ 2.28 อิลิเมนต์ที่ปนอยู่กับข้อความ

นอกจากที่กล่าวมาแล้วรูปแบบการประกาศอิลิเมนต์ยังมีอีกหลายรูปแบบ เช่น รูปแบบการประกาศอิลิเมนต์แบบให้เลือกด้วยเครื่องหมายไปป์ (|) และรูปแบบการประกาศอิลิเมนต์แบบระบุด้วย (+, *, ?) แสดงดังรูปที่ 2.29 และ 2.30 ตามลำดับ

```
<!ELEMENT ชื่ออิลิเมนต์ (อิลิเมนต์1|อิลิเมนต์2|อิลิเมนต์3)>
```

รูปที่ 2.29 รูปแบบการประกาศอิลิเมนต์แบบเลือก

การใช้เครื่องหมายไปป์ (|) คั่นระหว่างอิลิเมนต์หมายความว่าให้เลือกเพียงอิลิเมนต์อิลิเมนต์หนึ่งเท่านั้น

```
<!ELEMENT Book (ISBN,Title,Author+,Editor*,(Review|Example)?)>
```

รูปที่ 2.30 รูปแบบการประกาศอิลิเมนต์แบบระบุ

จากรูปที่ 2.30 สามารถอธิบายได้ดังนี้ เมื่ออิลิเมนต์ Book หมายถึงรูตอิลิเมนต์ อิลิเมนต์ ISBN หมายถึงที่สามารถปรากฏได้เพียงอิลิเมนต์เดียว และต้องมีปรากฏในเอกสารไม่มีไม่ได้ อิลิเมนต์ Title หมายถึงอิลิเมนต์ที่ปรากฏต่อจากอิลิเมนต์ ISBN มีได้เพียงอิลิเมนต์เดียวและไม่มีไม่ได้ อิลิเมนต์ Author หมายถึงอิลิเมนต์ที่ปรากฏต่อจาก Title มีได้ตั้งแต่ 1 อิลิเมนต์ขึ้นไป และต้องมีปรากฏไม่มีไม่ได้ อิลิเมนต์ Editor หมายถึงอิลิเมนต์ที่ปรากฏต่อจาก Author มีได้ตั้งแต่ 1 อิลิเมนต์ขึ้นไปหรือไม่มีปรากฏ

เลยก็ได้ อิลิเมนต์ Review หรือ Example เป็นอิลิเมนต์ที่ปรากฏต่อจาก Editor จะมีหรือไม่ก็ได้ ถ้ามีจะปรากฏได้เพียงอิลิเมนต์ใดอิลิเมนต์หนึ่งเท่านั้น

2 การประการแอตทริบิวต์

การประการแอตทริบิวต์มีความสำคัญไม่น้อยไปกว่าการประกาศอิลิเมนต์ เนื่องจากเอกสาร XML โดยทั่วไปจะประกอบด้วยข้อมูลที่ถูกเก็บอยู่ภายในอิลิเมนต์ และข้อมูลที่บรรจุอยู่ภายในแอตทริบิวต์แล้วแต่ผู้ออกแบบโครงสร้างของเอกสารจะเป็นผู้กำหนด รูปแบบการประกาศแอตทริบิวต์แสดงได้ดังรูปที่ 2.31 ตัวอย่างการประกาศแอตทริบิวต์แสดงดังรูปที่ 2.32

```
<!ATTLIST ชื่ออิลิเมนต์ ชื่อแอตทริบิวต์ ชนิดข้อมูลของแอตทริบิวต์
(#REQUIRED|#IMPLIED|#FIXED ค่าปกติของแอตทริบิวต์)>
```

รูปที่ 2.31 รูปแบบการประกาศแอตทริบิวต์

เมื่อชื่ออิลิเมนต์หมายถึง ชื่อของอิลิเมนต์ที่มีแอตทริบิวต์ปรากฏอยู่ ชื่อแอตทริบิวต์หมายถึงชื่อของแอตทริบิวต์ที่ปรากฏ ชนิดข้อมูลของแอตทริบิวต์ หมายถึงชนิดของข้อมูลที่กำหนดซึ่งชนิดของข้อมูลถูกกำหนดโดยสมาคม W3C #REQUIRED หมายถึงแอตทริบิวต์นั้นจำเป็นต้องมีปรากฏ #IMPLIED หมายถึงแอตทริบิวต์นั้นจะมีปรากฏหรือไม่ก็ได้ #FIXED ค่าปกติของแอตทริบิวต์ หมายถึงเมื่อมีการระบุคีย์เวิร์ดนี้จะต้องระบุค่าปกติต่อท้ายเสมอ

```
<!ATTLIST Student ID CDATA #REQUIRED)>
```

รูปที่ 2.32 ตัวอย่างการประกาศแอตทริบิวต์

จากรูปที่ 2.32 สามารถอธิบายรายละเอียดได้ดังนี้ เมื่อ Student หมายถึงชื่อของอิลิเมนต์ ID หมายถึงชื่อของแอตทริบิวต์ CDATA หมายถึงชนิดของข้อมูล #REQUIRED หมายถึงการระบุว่าต้องมีแอตทริบิวต์นี้ปรากฏทุกครั้ง และการกำหนดอิลิเมนต์แสดงดังรูปที่ 2.26

2.3.2 การกำหนดโครงสร้างของเอกสาร XML แบบ XML Schema

การกำหนดโครงสร้างของเอกสาร XML แบบ DTD ที่กล่าวมาแล้วนั้นเป็นรูปแบบของการกำหนดโครงสร้างของเอกสารของภาษา SGML ยังมีข้อด้วยที่ไม่เหมาะสมกับภาษา XML หลายประการเช่น มีชนิดของข้อมูลน้อย และมีรูปแบบการเขียน (Syntax) ที่ไม่เหมือนกับภาษา XML เป็นต้น ทำให้ต้องใช้ตัวแปรภาษาคนละตัวกับภาษา XML ด้วยเหตุผลที่กล่าวมาข้างต้น

ทางสมาคม W3C จึงได้ออกมาตรฐานการกำหนดโครงสร้างของเอกสาร XML แบบเอ็กซ์เอ็มแอลสกีมา (XML Schema) มาตรฐานของเอ็กซ์เอ็มแอลสกีมาแบ่งออกเป็น 3 ส่วนคือ ส่วนแรกเป็นข้อมูลเบื้องต้น (Primer) ส่วนที่สองเป็นส่วนของโครงสร้าง (Structure) และส่วนที่สามเป็นชนิดของข้อมูล (Datatypes)

2.3.2.1 ลักษณะของเอ็กซ์เอ็มแอลสกีมา

เพื่อให้เห็นภาพลักษณ์ของเอ็กซ์เอ็มแอลสกีมาที่ชัดเจน จะอธิบายการกำหนดโครงสร้างแบบ DTD ควบคู่ไปกับเอ็กซ์เอ็มแอลสกีมา แสดงดังรูปที่ 2.33 และ 2.34 ตามลำดับ

```
<!ELEMENT BookStore(Book)*>
<!ELEMENT Book(Title,Author,Data,ISBN,Publisher)>
<!ELEMENT Title(#PCDATA)>
<!ELEMENT Author(#PCDATA)>
<!ELEMENT Data(#PCDATA)>
<!ELEMENT ISBN(#PCDATA)>
<!ELEMENT Publisher(#PCDATA)>
```

รูปที่ 2.33 การกำหนดโครงสร้างแบบ DTD

```
<?XML version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.books.org"
            xmlns="http://www.books.org"
            elementFormDefault="qualified">
    <xsd:element name="BookStore">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element ref="Book" minOccurs="1" maxOccurs="unbounded"/>
                </xsd:sequence>
            </xsd:complexType>
    </xsd:element>
    <xsd:element name="Book">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element ref="Title" minOccurs="1" maxOccurs="unbounded"/>
                    <xsd:element ref="Author" minOccurs="1" maxOccurs="unbounded"/>
                    <xsd:element ref="Date" minOccurs="1" maxOccurs="unbounded"/>
                    <xsd:element ref="ISBN" minOccurs="1" maxOccurs="unbounded"/>
                    <xsd:element ref="Publisher" minOccurs="1"
maxOccurs="unbounded"/>
                </xsd:sequence>
            </xsd:complexType>
    </xsd:element>
    <xsd:element name="Title" type="xsd:string">
    <xsd:element name="Author" type="xsd:string">
    <xsd:element name="Date" type="xsd:string">
    <xsd:element name="ISBN" type="xsd:string">
    <xsd:element name="Publisher" type="xsd:string">
</xsd:schema>
```

รูปที่ 2.34 การกำหนดโครงสร้างแบบ XML Schema

เมื่อเปรียบเทียบการกำหนดโครงสร้างของเอกสารแบบ DTD ในรูปที่ 2.33 กับการกำหนดโครงสร้างแบบ XML Schema ในรูปที่ 2.34 แล้วจะเห็นได้ว่าการกำหนดโครงสร้างแบบเอ็กซ์เอ็มแอลสกีมานั้นมีความซับซ้อนกว่าแบบ DTD พอสมควรการกำหนดอิลิเมนต์ต่างๆ ในเอ็กซ์เอ็มแอลสกีมาจะกล่าวถึงในหัวข้อถัดไป

2.3.2.2 ประเภทอิลิเมนต์ของโครงสร้างแบบ XML Schema

การกำหนดโครงสร้างของเอกสาร XML แบบเอ็กซ์เอ็มแอลสกีมาที่แสดงในรูปที่ 2.34 จะประกอบด้วยการกำหนดอิลิเมนต์ต่างๆ จำนวนมาก อิลิเมนต์ที่กล่าวถึงในเอ็กซ์เอ็มแอลสกีมาแบ่งออกเป็นสองชนิดคือ อิลิเมนต์แบบธรรมดา (Simple Type) และอิลิเมนต์แบบซับซ้อน (Complex Type)

อิลิเมนต์แบบธรรดาคืออิลิเมนต์ที่มีข้อมูลภายในเป็นพื้นฐาน เช่น สตริง ตัวเลข และวันที่เป็นต้น ชนิดข้อมูลของอิลิเมนต์แบบธรรมดาถูกกำหนดในมาตรฐานเอ็กซ์เอ็มแอลสกีมา ส่วนที่3นอกจากนี้แล้วแอตทริบิวต์ก็จัดว่าเป็นอิลิเมนต์แบบธรรมดาเช่นกัน รูปแบบการประกาศอิลิเมนต์แบบธรรมดาแสดงดังรูปที่ 2.35 และตัวอย่างการประกาศอิลิเมนต์แบบธรรมดาแสดงดังรูปที่ 2.36

```
<xsd:element name="ชื่ออิลิเมนต์" type="ชื่อชนิดข้อมูล">
```

รูปที่ 2.35 รูปแบบการประกาศอิลิเมนต์แบบธรรมดา

```
<xsd:element name="Title" type="xsd:string">
```

รูปที่ 2.36 ตัวอย่างการประกาศอิลิเมนต์แบบธรรมดา

อิลิเมนต์แบบซับซ้อนคืออิลิเมนต์ที่มีข้อมูลภายในเป็นอิลิเมนต์ หรืออิลิเมนต์ที่มีแอตทริบิวต์ปรากฏอยู่ รูปแบบการประกาศอิลิเมนต์แบบซับซ้อนแสดงดังรูปที่ 2.37 และตัวอย่างของการประกาศอิลิเมนต์แบบซับซ้อนแสดงดังรูปที่ 2.38

```
<xsd:element name="ชื่ออิลิเมนต์">
        <xsd:complexType>
                [<xsd:sequence>,<xsd:choice>]
                        <element1>
                        <element2>
                        <element3>
                          .
                          .
                          .
                        <elementn>
                [</xsd:sequence>,</xsd:choice>]
        </xsd:complexType>
</xsd:element>
```

รูปที่ 2.37 รูปแบบการประกาศอิลิเมนต์แบบซับซ้อน

```
<xsd:ement name="Book">
        <xsd:complexType>
                <xsd:sequence>
                        <xsd:element name="Title" type="xsd:string" />
                        <xsd:element name="Author" type="xsd:string" />
                        <xsd:element name="Publisher type="xsd:string" />
                [</xsd:sequence>,</xsd:choice>]
        </xsd:complexType>
</xsd:element>
```

รูปที่ 2.38 ตัวอย่างการประกาศอิลิเมนต์แบบซับซ้อน

คีย์เวิร์ด "<xsd:sequence>" xsd (XML Schema Definition) หมายถึงการบ่งบอกว่าเป็นเอกสารเอ็กซ์เอ็มแอลสกีมา และไฟล์ข้อมูลของเอ็กซ์เอ็มแอลสกีมามีนามสกุล(.xsd) sequence หมายถึง การเรียงลำดับของอิลิเมนต์ลูกที่อยู่ภายในอิลิเมนต์ซับซ้อน และอิลิเมนต์ที่ปรากฏในเอกสาร XML ที่อ้างถึงต้องเรียงลำดับตามที่กำหนด และคีย์เวิร์ดอีกตัวคือ

(<xsd:choice>) หมายถึงการระบุว่าอิลิเมนต์ลูกตัวใดตัวหนึ่งที่ปรากฏได้ในเอกสาร XML ที่อ้างถึง

การกำหนดโครงสร้างของเอกสาร XML แบบเอ็กซ์เอ็มแอลสกีมาสามารถกำหนดจำนวนอิลิเมนต์ที่สามารถปรากฏได้ในเอกสาร XML ด้วยการใช้แอตทริบิวต์ minOccurs เป็นแอตทริบิวต์ที่ระบุจำนวนต่ำสุดที่จำนวนอิลิเมนต์สามารถปรากฏได้ค่าที่ระบุส่วนมาก 0 กับ 1 และmaxOccurs เป็นแอตทริบิวต์ที่ระบุจำนวนสูงสุดที่จำนวนอิลิเมนต์ที่สามารถปรากฏได้ในเอกสาร XML สามารถระบุค่าได้หลายค่าตั้งแต่ 1 จนถึง "unbounded" การระบุค่าสูงสุดเป็น unbounded หมายความว่าสามารถระบุค่าของอิลิเมนต์นั้นได้โดยไม่จำกัดจำนวน ในกรณีที่เราไม่ระบุแอตทริบิวต์ minOccurs และ maxOccurs ก็จะถือเอาค่าปกติมากำหนดจำนวนอิลิเมนต์ คำปกติมีค่าเป็น 1 หมายความว่าจำนวนอิลิเมนต์ปรากฏได้ต่ำสุดและสูงสุดเท่ากับ 1 และต้องมีอิลิเมนต์ปรากฏถ้าไม่มีจะเกิดข้อผิดพลาด (Error) ตัวอย่างการใช้แอตทริบิวต์ minOccurs และ maxOccurs แสดงในรูปที่ 2.29 รูปแบบการประกาศอิลิเมนต์ของการกำหนดโครงสร้างเอกสารแบบเอ็กซ์เอ็มแอลสกีมายังมีอีกหลายรูปแบบ สามารถหาข้อมูลอ้างอิงได้ที่มาตรฐานเอ็กซ์เอ็มแอลสกีมาของ W3C ในส่วนที่ 2

2.3.2.3 การประกาศแอตทริบิวต์

รูปแบบการประกาศแอตทริบิวต์ของการกำหนดโครงสร้างของเอกสาร XML แบบเอ็กซ์เอ็มแอลสกีมา มีรูปแบบที่เหมือนกับรูปแบบการประกาศอิลิเมนต์แบบธรรมดา เพราะแอตทริบิวต์จัดอยู่ในอิลิเมนต์แบบธรรมดา รูปแบบการประกาศแอตทริบิวต์แสดงดังรูปที่ 2.39

```
<xsd:attribute name="ชื่อแอตทริบิวต์" type="ชื่อชนิดข้อมูล" [use default fixed ค่าปกติ] />
```

รูปที่ 2.39 รูปแบบการประกาศแอตทริบิวต์

เมื่อชื่อแอตทริบิวต์ หมายถึงชื่อของแอตทริบิวต์ ชื่อชนิดข้อมูล หมายถึงชื่อชนิดของข้อมูลเช่น สตริง นัมเบอร์ และวันที่ เป็นต้น นอกจากนี้ยังมีแอตทริบิวต์อื่นๆ ให้เลือก คือ แอตทริบิวต์ use หมายถึง การระบุระดับความจำเป็นของแอตทริบิวต์นี้ว่าต้องมีหรือไม่ ค่าของแอตทริบิวต์นี้มี 3 ค่า คือ required (จำเป็นต้องมี) option (มีหรือไม่ก็ได้) และ prohibited(ไม่ต้องมี) แอตทริบิวต์ default ใช้กำหนดค่าปกติของแอตทริบิวต์ แอตทริบิวต์ fixed ใช้กำหนดค่าของแอตทริบิวต์ แอตทริบิวต์ทั้ง 3 คือ use, default และ fixed จะมีหรือไม่ก็ได้เนื่องจากเป็นแอตทริบิวต์ทางเลือก ตัวอย่างการประกาศแอตทริบิวต์แสดงดังรูปที่ 2.40

```
<xsd:attribute name=”country” type=”xsd:string” fixed “TH” />
```

รูปที่ 2.40 ตัวอย่างการประกาศแอตทริบิวต์

2.3.3 ความแตกต่างระหว่าง DTD และ XML Schema

การกำหนดโครงสร้างของเอกสาร XML โดยมใช้ DTD และ XML Schema มีความแตกต่างกันหลายประการมีรายละเอียดดังนี้

- XML Schema มีรูปแบบการเขียน (Syntax) เหมือนกับเอกสาร XML ทำให้สามารถใช้ตัวแปลภาษาตัวเขวกันได้ แต่ DTD มีรูป แบบการเขียนที่ต่างจากเอกสาร XML ทำให้ต้องมีตัวแปลภาษาสำหรับ DTD แยกต่างหากเกิดความสิ้นเปลืองและความยากลำบากในการใช้งาน
- XML Schema สนับสนุนการใช้เนมสเปส (Namespaces) ให้สารามารถใช้แท็กซ์ที่มีชื่อเหมือนกันในเอกสารเดียวกันแต่มีความหมายแตกต่างกัน สำหรับการกำหนดโครงสร้างแบบ DTD นั้นไม่สามารถทำได้
- XML Schema มีชนิดของข้อมูลมากกว่า DTD และชื่อชนิดของข้อมูลมีชื่อเรียกที่ใช้กันทั่วไปในการเขียนโปรแกรมเช่น สตริง นัมเบอร์ เป็นต้น สำหรับการกำหนดโครงสร้างแบบ DTD นั้นมีชื่อเรียกชนิดของข้อมูลที่ไม่คุ้นเคยเช่น PCDATA, NMTOKENS เป็นต้น
- การกำหนดโครงสร้างของเอกสาร XML แบบ XML Schema สามารถกำหนดลำดับของอิลิเมนต์ที่ปรากฏในเอกสารเอ็กซ์เอ็มแอลสกีมาได้
- การกำหนดโครงสร้างของเอกสาร XML แบบ XML Schema สามารถกำหนดรูปแบบชนิดของข้อมูลตามที่ผู้ใช้ต้องการได้เช่น เบอร์โทรศัพท์มีรูปแบบ “dd-ddd-dddd” หมายถึงรูปแบบของเบอร์โทรศัพท์ที่มีตัวเลขข้างหน้า 2 หลักตามด้วยขีดและเลขอีก 3 หลักตามด้วยขีด และเลขอีก 4 หลักเป็นต้น

2.4 การประยุกต์ใช้งานภาษา XML

ปัจจุบันมีการนำภาษา XML ไปประยุกต์ใช้งานหลายด้าน พอจะสรุปเป็นหัวข้อหลักๆ ได้ดังต่อไปนี้

2.4.1 ใช้สำหรับแยกข้อมูลจากภาษา HTML

การใช้ภาษา HTML แสดงผลข้อมูลบนบราวเซอร์ข้อมูลที่จะแสดงถูกเก็บในภาษา HTML แต่เราสามารถใช้ภาษา XML ร่วมกับ HTML ได้โดยใช้ HTML มีบทบาทสำคัญในการแสดงผลและจัดตำแหน่งการแสดงผล ส่วนภาษา XML จะเป็นส่วนเก็บข้อมูลที่จะแสดงแยก

เก็บในไฟล์ (.xml) ต่างหาก ข้อมูล XML จะปรากฏเป็นกลุ่มๆ ใน ภาษา HTML หรือเรียกว่า "Data Islands"

2.4.2 ใช้สำหรับแลกเปลี่ยนข้อมูลระหว่างแอปพลิเคชั่น

การเก็บข้อมูลในระบบคอมพิวเตอร์ และฐานข้อมูลในปัจจุบันมีหลากหลายรูปแบบทำให้เป็นปัญหานี้คือต้องหารูปแบบของข้อมูลที่ไม่ขึ้นอยู่กับแพลตฟอร์ม และสามารถอ่านได้หลายๆ แอปพิเคชั่น ซึ่งตรงกับคุณสมบัติของภาษา XML เพราะข้อมูลในเอกสาร XML มีลักษณะเป็นไฟล์ข้อมูลธรรมดาที่อ่านได้ทั้งมนุษย์ และโปรแกรมแอปพลิเคชั่น

2.4.3 ใช้สำหรับเก็บข้อมูล

ภาษา XML สามารถนำไปประยุกต์ใช้ในการเก็บข้อมูลทั้งในรูปของไฟล์ข้อมูล และการเก็บข้อมูลลงฐานข้อมูล นอกจากนี้ยังมีแอปพลิเคชั่นที่สามารถค้นหาข้อมูล XML มาแสดงผลในแพลทฟอร์มต่างๆ ได้

2.4.4 ใช้สำหรับการสร้างภาษาใหม่

เนื่องจากภาษา XML เป็นภาษาที่สามารถกำหนดโครงสร้างขึ้นเองได้ จึงมีการนำภาษา XML มาเป็นแม่แบบในการสร้างภาษาใหม่ขึ้นมาใช้งานกับแพลตฟอร์มต่างๆ เช่น WML (Wireless Markup Language) ใช้สำหรับอุปกรณ์แบบพกพาเช่นโทรศัพท์มือถือเป็นต้น

2.4.5 ใช้สำหรับส่งข้อมูลระหว่างองค์กรธุรกิจ (B2B:Business to Business)

ภาษา XML เป็นภาษาที่สามารถกำหนดโครงสร้างของเอกสารที่สร้างขึ้นได้ ดังนั้นเมื่อองค์กรต้องการแลกเปลี่ยนข้อมูลกันจึงต้องมีการกำหนดโครงสร้างที่เหมือนกันจึงจำทำให้เอกสารที่แลกเปลี่ยนกันนั้นมีความถูกต้องและมีความน่าเชื่อถือสูง

2.5 XPath

Xpath คือภาษาที่ใช้ค้นหาข้อมูลภายในเอกสาร XML โดยเราใช้ XPath เพื่อระบุ Element หรือ Attribute ในเอกสาร XML ที่ต้องการเดินทางไป Xpath ใช้ Path expression ในการเลือก Node หรือเซตของ Node ในเอกสาร XML ซึ่ง Path expression เหล่านี้มีรูปแบบการใช้เหมือนกับ Path expression ในระบบไฟล์ของคอมพิวเตอร์ นอกจากนี้ XPath ยังมี Build-in function ให้เลือกใช้กว่า 100 ฟังก์ชัน ฟังก์ชันเหล่านี้ ได้แก่ ฟังก์ชันเกี่ยวกับค่าของ String ค่าของตัวเลข การเปรียบเทียบวันที่และเวลา เป็นต้น นอกจากนี้ XPath ยังเป็นมาตรฐานของ W3C อีกด้วย

2.5.1 คำศัพท์ต่างๆ ใน XPath

2.5.1.1 Node

ใน Xpath มี Node ทั้งหมด 7 ชนิด คือ Element, Attribute, ข้อความ, Namespace, คำสั่งในการดำเนินการ (processing-instruction), ข้อความอธิบาย (comment) และ Node เอกสาร (root node) เอกสาร XML อยู่ในรูปของโครงสร้างต้นไม้ โดยเราเรียกรากของต้นไม้ว่า Node เอกสาร หรือ root node

จากเอกสาร XML ดังรูปที่ 2.41 นี้

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<bookstore>

<book category="COOKING">
  <title lang="en">Everyday Italian</title>
  <author>Giada De Laurentiis</author>
  <year>2005</year>
  <price>30.00</price>
</book>

<book category="CHILDREN">
  <title lang="en">Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>

<book category="WEB">
  <title lang="en">XQuery Kick Start</title>
  <author>James McGovern</author>
  <author>Per Bothner</author>
  <author>Kurt Cagle</author>
  <author>James Linn</author>
  <author>Vaidyanathan Nagarajan</author>
  <year>2003</year>
  <price>49.99</price>
</book>

<book category="WEB">
  <title lang="en">Learning XML</title>
  <author>Erik T. Ray</author>
  <year>2003</year>
  <price>39.95</price>
</book>

</bookstore>
```

รูปที่ 2.41 เอกสาร XML ที่ใช้ประกอบการอธิบายเรื่อง XPath

จะเห็นตัวอย่างของ Node ในเอกสาร XML ข้างต้น ดังรูปที่ 2.42 นี้

```
<bookstore>  (document node)

<author>J K. Rowling</author>  (element node)

lang="en"  (attribute node)
```

รูปที่ 2.42 ตัวอย่างของ Node ในเอกสาร XML ที่ใช้ประกอบการอธิบายเรื่อง XPath

2.5.1.2 Atomic Value

Atomic value คือ Node ที่ไม่มีลูกหรือพ่อแม่

ตัวอย่างของ Atomic value ดังรูปที่ 2.43

```
J K. Rowling

"en"
```

รูปที่ 2.43 ตัวอย่างของ Atomic Value ในเอกสาร XML ที่ใช้ประกอบการอธิบายเรื่อง XPath

2.5.1.3 Item

Item คือ Atomic value หรือ Node

2.5.2 ความสัมพันธ์ระหว่าง Node

```
<book>
  <title>Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>
```

รูปที่ 2.44 ตัวอย่างของความสัมพันธ์ระหว่าง Node แบบที่ 1

2.5.2.1 พ่อแม่ (Parent)

Element และ Attribute แต่ละตัวมีพ่อแม่ได้แค่ตัวเดียว

ตัวอย่างในรูปที่ 2.44 Element ที่ชื่อว่า book เป็นพ่อแม่ของ title, author, year และ price

2.5.2.2 ลูก (Children)

Element node หนึ่งๆ อาจมีลูกได้ตั้งแต่ศูนย์ตัวขึ้นไป

ตัวอย่างในรูปที่ 2.44 ที่ชื่อว่า title, author, year และ price เป็นลูกของ Element ที่ชื่อว่า book

2.5.2.3 พี่น้อง (Sibling)

Node ที่มีพ่อแม่เดียวกัน

ตัวอย่างในรูปที่ 2.44 Element ที่ชื่อว่า title, author, year และ price เป็นพี่น้องกัน

```
<bookstore>

<book>
  <title>Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>

</bookstore>
```

รูปที่ 2.45 ตัวอย่างของความสัมพันธ์ระหว่าง Node แบบที่ 2

2.5.2.4 บรรพบุรุษ (Ancestor)

พ่อแม่ของ Node นั้นๆ รวมถึงปู่ย่าตายายขึ้นๆ ไป

ตัวอย่างในรูปที่ 2.45 บรรพบุรุษของ Element ที่ชื่อว่า title คือ Element ที่ชื่อว่า book และ bookstore

2.5.2.5 ผู้สืบสกุล (Descendant)

ลูกๆ ของ Node นั้นๆ รวมถึงหลานๆ ลงไป

ตัวอย่างในรูปที่ 2.45 ผู้สืบสกุลของ Element ที่ชื่อว่า bookstore คือ Element ที่ชื่อว่า book, title, author, year และ price

2.5.3 การเลือก Node

XPath ใช้ Path expression ในการเลือก Node หรือเซตของ Node ในเอกสาร XML โดยการเลือก Node จะเลือกเป็นขั้นๆ ลงไป Path expression ที่สำคัญๆ มีดังนี้

ตารางที่ 2.3 Path expression ที่สำคัญ

Expression	คำอธิบาย
ชื่อ Node	เลือก Node ลูกทั้งหมดที่มีชื่อตามที่ระบุ
/	เลือก Node จาก Root node
//	เลือก Node ในเอกสารจาก Node ปัจจุบันที่ตรงกับที่ระบุ โดยไม่สนใจตำแหน่งที่อยู่ของ Node นั้น
.	เลือก Node ปัจจุบัน
..	เลือกพ่อแม่ของ Node ปัจจุบัน
@	เลือก Attribute

ตัวอย่างของการเลือก Node ดูได้จากตารางที่ 2.4

ตารางที่ 2.4 ตัวอย่างของการเลือก Node

Path Expression	ผลที่ได้
Bookstore	เลือก Node ที่เป็นลูกของ Element ทั้งหมดที่ชื่อ bookstore
/ bookstore	เลือก Root element ที่ชื่อว่า bookstore **หมายเหตุ** เส้นทางขึ้นต้นด้วยเครื่องหมาย "/" แสดงถึงเส้นทางที่เป็น Absolute path ไปยัง Element ที่ต้องการ
Bookstore/book	เลือก Element ที่ชื่อ book ทั้งหมดที่เป็นลูกของ bookstore
//book	เลือก Element ที่ชื่อ book ทั้งหมดโดยไม่สนใจว่าจะมี Element ที่ชื่อ book อยู่ที่ใดในเอกสารบ้าง
Bookstore//book	เลือก Element ที่ชื่อ book ทั้งหมดที่เป็นผู้สืบสกุลของ Element ที่ชื่อ bookstore โดยไม่สนใจว่าจะมี Element ที่ชื่อ book อยู่ที่ใดภายใต้ Element ที่ชื่อ bookstore บ้าง
//@lang	เลือก Attribute ทั้งหมดที่ชื่อ lang

2.5.3.1 Predicate

เราจะใช้ Predicate เพื่อหา Node ที่ระบุเฉพาะเจาะจงลงไป หรือใช้หา Node ที่มีค่าที่ระบุเฉพาะเจาะจง โดย Predicate จะระบุอยู่ในวงเล็บก้ามปูเสมอ ต่อไปนี้จะโชว์ตัวอย่าง Path expression ที่มี Predicate เป็นส่วนประกอบ

ตารางที่ 2.5 ตัวอย่าง Path expression ที่มี Predicate เป็นส่วนประกอบ

Path Expression	ผลที่ได้
/bookstore/book[0]	เลือก Element ตัวแรกที่ชื่อ book ที่เป็นลูกของ Element ที่ชื่อ bookstore หมายเหตุ ใน IE5 เป็นต้นมา Element ตัวแรกจะใช้ [0] แต่ตามมาตรฐานของ W3C จะใช้ [1]
/bookstore/book[last()]	เลือก Element ตัวสุดท้ายที่ชื่อ book ที่เป็นลูกของ Element ที่ชื่อ bookstore
/bookstore/book[position()<3]	เลือก Element 2 ตัวแรกที่ชื่อ book ที่เป็นลูกของ Element ที่ชื่อ bookstore
//title[@lang]	เลือก Element ทั้งหมดที่ชื่อ title ที่มี Attribute ชื่อ lang
//title[@lang='eng']	เลือก Element ทั้งหมดที่ชื่อ title ที่มี Attribute ชื่อ lang ที่มีค่า eng
/bookstore/book[price>35.00]	เลือก Element ทั้งหมดที่ชื่อ book ที่เป็นลูกของ Element ที่ชื่อ bookstore ที่มีค่าของ Element ที่ชื่อ price สูงกว่า 35.00
/bookstore/book[price>35.00]/title	เลือก Element ทั้งหมดที่ชื่อ title ที่เป็นลูกของ Element ที่ชื่อ book ซึ่งเป็นลูกของ Element ที่ชื่อ bookstore อีกทีหนึ่งที่มีค่าของ Element ที่ชื่อ price สูงกว่า 35.00

2.5.3.2 การเลือก Node ที่ไม่รู้จัก

เราสามารถใช้ XPath wildcard ในการเลือก Node ที่ไม่รู้จักได้

ตารางที่ 2.6 การเลือก Node ที่ไม่รู้จักโดยใช้ XPath wildcard

Wildcard	คำอธิบาย
*	ใช้เลือก Element node ใดๆ
@*	ใช้เลือก Attribute node ใดๆ
Node()	ใช้เลือก Node ชนิดใดๆ ก็ได้

ตัวอย่าง Path expression ที่ใช้ XPath wildcard มีดังนี้

ตารางที่ 2.7 ตัวอย่าง Path expression ที่ใช้ XPath wildcard

Path Expression	ผลที่ได้
/bookstore/*	เลือก Element ทั้งหมดที่เป็นลูกของ Element ที่ชื่อ bookstore
//*	เลือก Element ทั้งหมดในเอกสาร
//title[@*]	เลือก Element ที่ชื่อ title ทั้งหมดที่มี Attribute ค่าใดๆ

2.5.3.3 การเลือกหลายเส้นทาง

เราสามารถเลือกเส้นทางหลายๆ เส้นใน XPath expression ได้โดยใช้ | Operator ซึ่งตัวอย่างของการใช้งานเป็นดังตารางที่ 2.8

ตารางที่ 2.8 ตัวอย่าง Path expression ที่ใช้ | Operator ในการเลือกหลายเส้นทาง

Path Expression	ผลที่ได้
//book/title | //book/price	เลือก Element ที่ชื่อ title และ price ทั้งหมดที่เป็นลูกของ Element ที่ชื่อ book
//title | //price	เลือก Element ที่ชื่อ title และ price ทั้งหมดในเอกสาร
/bookstore/book/title | //price	เลือก Element ที่ชื่อ title ทั้งหมดที่เป็นลูกของ Element ที่ชื่อ book ซึ่งเป็นลูกของ Element ที่ชื่อ bookstore อีกทีหนึ่ง และ เลือก Element ที่ชื่อ price ทั้งหมดในเอกสาร

2.5.4 แกนของ XPath

แกนของ XPath ใช้กำหนดความสัมพันธ์ของเซตของ Node กับ Node ปัจจุบัน

ตารางที่ 2.9 แกนของ XPath

ชื่อแกน	ผลที่ได้
Ancestor	เลือกบรรพบุรุษทั้งหมดของ Node ปัจจุบัน
ancestor-or-self	เลือกบรรพบุรุษทั้งหมดของ Node ปัจจุบันและ เลือก Node ปัจจุบันด้วย
Attribute	เลือก Attribute ทั้งหมดของ Node ปัจจุบัน
Child	เลือกลูกทั้งหมดของ Node ปัจจุบัน
Descendant	เลือกผู้สืบสกุลทั้งหมดของ Node ปัจจุบัน
descendant-or-self	เลือกผู้สืบสกุลทั้งหมดของ Node ปัจจุบันและ เลือก Node ปัจจุบันด้วย
Following	เลือกทุกๆ อย่างในเอกสารที่อยู่หลังแท็กปิดของ Node ปัจจุบัน
following-sibling	เลือกพี่น้องทั้งหมดที่อยู่หลัง Node ปัจจุบัน
Namespace	เลือก Namespace node ทั้งหมดของ Node ปัจจุบัน
Parent	เลือกพ่อแม่ของ Node ปัจจุบัน
Preceding	เลือกทุกๆ อย่างในเอกสารที่อยู่ก่อนหน้าแท็กเปิดของ Node ปัจจุบัน
preceding-sibling	เลือกพี่น้องทั้งหมดที่อยู่ก่อนหน้า Node ปัจจุบัน
Self	เลือก Node ปัจจุบัน

2.5.5 ที่อยู่ของ Path expression

ที่อยู่ของ Path สามารถเป็นได้ 2 แบบ คือ Absolute หรือ Relative

ที่อยู่ของ Path แบบ Absolute จะเริ่มต้นด้วยเครื่องหมาย / ซึ่งที่อยู่ของ Path แบบ Relative จะไม่ได้ขึ้นต้นด้วยเครื่องหมาย / นี้ ทั้ง 2 กรณีนี้ ที่อยู่ของ Path จะประกอบด้วยขั้นตอนตั้งแต่ 1 ขั้นขึ้นไป โดยแยกแต่ละขั้นตอนด้วยเครื่องหมาย /

```
An absolute location path:

/step/step/...

A relative location path:

step/step/...
```

รูปที่ 2.46 ที่อยู่ของ Path แบบ Absolute และ Relative

แต่ละขั้นตอนจะหาค่ากับ Node ต่างๆ ในเซตของ Node ปัจจุบัน
ขั้นตอนประกอบด้วย

- แกน (กำหนดความสัมพันธ์ของต้นไม้ระหว่าง Node ที่เลือกกับ Node ปัจจุบัน)
- Node test (ระบุ Node ข้างในแกน)
- Predicate ตั้งแต่ศูนย์ตัวขึ้นไป (ระบุเซตของ Node ที่เลือกมาแล้วให้เฉพาะเจาะจงลงไป)

Syntax สำหรับขั้นตอนของที่อยู่คือ *ชื่อแกน::nodetest[predicate]*
ตัวอย่างของขั้นตอนของที่อยู่ดังแสดงต่อไปนี้

ตารางที่ 2.10 ตัวอย่างของขั้นตอนของที่อยู่

ตัวอย่าง	ผลที่ได้
child::book	เลือก Node ที่ชื่อว่า book ทั้งหมดที่เป็นลูกของ Node ปัจจุบัน
attribute::lang	เลือก Attribute ที่ชื่อว่า lang ทั้งหมดของ Node ปัจจุบัน
child::*	เลือกลูกของ Node ปัจจุบันทั้งหมด
attribute::*	เลือก Attribute ของ Node ปัจจุบันทั้งหมด
child::text()	เลือก Node ลูกที่เป็นข้อความของ Node ปัจจุบันทั้งหมด
child::node()	เลือก Node ลูกของ Node ปัจจุบันทั้งหมด
descendant::book	เลือก book ที่เป็นผู้สืบสกุลของ Node ปัจจุบันทั้งหมด
ancestor::book	เลือก book ที่เป็นบรรพบุรุษของ Node ปัจจุบันทั้งหมด
ancestor-or-self::book	เลือก book ที่เป็นบรรพบุรุษของ Node ปัจจุบันทั้งหมด และถ้า Node ปัจจุบันเป็น book เลือก Node ปัจจุบันด้วย
child::*/child::price	เลือก price ที่เป็นของหลานของ Node ปัจจุบันทั้งหมด

2.5.6 XPath Operator

ตารางที่ 2.11 แสดง Operator ที่สามารถใช้ได้ใน XPath expression

ตารางที่ 2.11 Operator ที่สามารถใช้ได้ใน XPath expression

Operator	คำอธิบาย	ตัวอย่าง	ค่าที่ Return
|	คำนวณ 2 เซตของ Node	//book | //cd	Return เซตของ Node ที่เป็น Element ที่ชื่อ book และ cd ทั้งหมด
+	การบวก	6 + 4	10
-	การลบ	6 – 4	2
*	การคูณ	6 * 4	24
Div	การหาร	8 div 4	2
=	เท่ากับ	price=9.80	จริง ถ้าราคาเท่ากับ 9.80 เท็จ ถ้าราคาเท่ากับ 9.90
!=	ไม่เท่ากับ	price!=9.80	จริง ถ้าราคาเท่ากับ 9.90 เท็จ ถ้าราคาเท่ากับ 9.80
<	น้อยกว่า	price<9.80	จริง ถ้าราคาเท่ากับ 9.00 เท็จ ถ้าราคาเท่ากับ 9.80
<=	น้อยกว่าหรือเท่ากับ	price<=9.80	จริง ถ้าราคาเท่ากับ 9.00 เท็จ ถ้าราคาเท่ากับ 9.90
>	มากกว่า	price>9.80	จริง ถ้าราคาเท่ากับ 9.90 เท็จ ถ้าราคาเท่ากับ 9.80
>=	มากกว่าหรือเท่ากับ	price>=9.80	จริง ถ้าราคาเท่ากับ 9.90 เท็จ ถ้าราคาเท่ากับ 9.70
Or	หรือ	price=9.80 or price=9.70	จริง ถ้าราคาเท่ากับ 9.80 เท็จ ถ้าราคาเท่ากับ 9.50
And	และ	price>9.00 and price<9.90	จริง ถ้าราคาเท่ากับ 9.80 เท็จ ถ้าราคาเท่ากับ 8.50
Mod	หารเอาเศษ	5 mod 2	1

2.6 XQuery

XQuery คือภาษาที่ใช้ค้นหาข้อมูลภายในเอกสาร XML ถูกคิดค้นและพัฒนาโดยองค์กร W3C เพื่อให้เป็นมาตรฐานสำหรับการค้นหาข้อมูลภายในเอกสาร XML โดย XQuery เป็น Functional Language ที่ผู้ใช้สามารถอ่านแล้วเข้าใจ (Human-readable) และผลลัพธ์จะยังคงอยู่ใน

รูปแบบตามโครงสร้างของ XML ด้วยเช่นกัน สำหรับการเขียนชุดคำสั่งหนึ่งๆ ในการค้นหาข้อมูลนั้นจะอยู่ในรูปของ Expression ซึ่ง Expression ที่เป็นส่วนประกอบใน XQuery มีดังนี้ รูปแบบการเข้าถึงข้อมูล (Path Expression) ไวยากรณ์ FLWOR การใช้ Operator และฟังก์ชัน Build-in การใช้เงื่อนไขในการแสดงผลลัพธ์ (Condition Expression) และการใช้เงื่อนไขในการจำกัดจำนวนข้อมูล (Quantified Expression) พร้อมกันนี้ XQuery ยังมีความสามารถในการค้นหาข้อมูลจากเอกสาร XML มากกว่า 1 เอกสารในการค้นหาแต่ละครั้งโดยที่เอกสาร XML เหล่านั้นไม่จำเป็นต้องถูกเก็บไว้ในที่เก็บข้อมูลที่เดียวกัน และความสามารถในการสร้างฟังก์ชันขึ้นใช้งานเองได้ที่นอกเหนือจากฟังก์ชัน Build-in

2.6.1 รูปแบบการเข้าถึงข้อมูล

XQuery มีวิธีการชี้ตำแหน่งของส่วนต่างๆ ในเอกสาร XML โดยอิงตามหลักไวยากรณ์ของ Xpath ซึ่งยังคงมองเอกสาร XML ในลักษณะของลำดับชั้นต้นไม้ดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 2.1 การเขียนนิพจน์สำหรับการเข้าถึงข้อมูล Author

/BookStore/Book/Author คำอธิบาย: เข้าถึงข้อมูลของ Author Element ทั้งหมดที่อยู่ภายใต้ Book Element และ Book Element ก็ต้องอยู่ภายใต้ BookStore Element

2.6.2 ไวยากรณ์ FLWOR

ไวยากรณ์ที่ใช้ในการค้นหาข้อมูลของ XQuery นั้นพยายามลอกเลียนแบบการทำงานมาจาก SQL ให้มากที่สุด เพื่อให้สามารถเรียนรู้ได้ง่าย ไวยากรณ์ที่กล่าวมานี้เรียกว่า FLWOR ซึ่งย่อมาจาก for, let, where, order by และ return สำหรับการค้นหาข้อมูลในเอกสาร XML นั้น for และ/หรือ let จะใช้ในการกำหนดเอกสาร XML ที่ต้องการค้นหาข้อมูล และเป็นการชี้ตำแหน่งเริ่มต้น where จะใช้สำหรับการกรองข้อมูลที่ต้องการค้นหา order by จะใช้สำหรับกำหนดการเรียงลำดับของข้อมูลผลลัพธ์ และ return จะใช้ในการกำหนอรูปแบบของผลลัพธ์ตามโครงสร้างของ XML ใน XQuery นั้น ให้ความสำคัญกับตัวอักษรเล็ก-ใหญ่เหมือนกับการกำหนดชื่อแท็กใน XML ดังนั้นจึงต้องมีการกำหนดรูปแบบของการใช้คำสั่งใน XQuery ซึ่งจะใช้ตัวอักษรตัวเล็กในทุกๆ คำสั่งของ XQuery

2.6.2.1 for และ let

for ใช้สำหรับกำหนดตัวแปร และ/หรือชุดของตัวแปร ซึ่งในแต่ละชุดคำสั่งของ for นั้นจะคืนค่าออกมาเป็นกลุ่มของโหนด และค่าผลลัพธ์ในแต่ละโหนด (รวมถึงโหนดลูกหลานของโหนดดังกล่าวด้วย) ที่ได้มาก็จะถูกเก็บไว้ในตัวแปรที่กำหนด และมีการวนซ้ำในการคืนค่าของผลลัพธ์ด้วย

let ใช้สำหรับกำหนดตัวแปร และ/หรือชุดของตัวแปรเหมือนกับ for พร้อมกับการคืนค่าข้อมูลก็เหมือนกับ for ด้วยเช่นกัน ซึ่งถ้ามองดูแบบผิวเผินแล้วการทำงานของ let นั้นจะคล้ายกับการทำงานของ for แต่ในความเป็นจริงแล้วทั้งสองยังคงมีสิ่งที่แตกต่างกันอยู่คือ let จะไม่มีการวนซ้ำในการคืนค่าของผลลัพธ์

ถึงแม้ว่าการทำงานของ for และ let จะคล้ายคลึงกันมาก แต่อย่างไรแล้วทั้งสองยังคงมีความแตกต่างกันในเรื่องของการคืนค่าผลลัพธ์ เพื่อให้เข้าใจถึงความแตกต่างของ for และ let จะแสดงในตัวอย่างต่อไปนี้

ตัวอย่างที่ 2.2 การใช้ for และการคืนค่าผลลัพธ์

```
for $s in (<one/>,<two/>,<three/>)
return <out> { $s } </out>
ผลลัพธ์ที่ได้คือ
<out>
    <one/>
</out>
<out>
    <two/>
</out>
<out>
    <three/>
</out>
```

ตัวอย่างที่ 2.3 การใช้ let และการคืนค่าผลลัพธ์

```
let $s :=(<one/>,<two/>,<three/>)
return <out> { $s } </out>
ผลลัพธ์ที่ได้คือ
<out>
    <one/>
    <two/>
    <three/>
</out>
```

2.6.2.2 where

where ใช้สำหรับกำหนดเงื่อนไขในการกรองข้อมูลที่ต้องการจากที่ได้กำหนดไว้ในตัวแปร (จากการใช้ for และ/หรือ let) สำหรับการคืนค่าของการใช้ where นี้จะเป็นในลักษณะของบูลีน การทำงานในส่วนนี้เปรียบเทียบได้กับการใช้ where ใน SQL นั่นเอง ดังแสดงในตัวอย่างต่อไปนี้

ตัวอย่างที่ 2.4 ค้นหาชื่อหนังสือจากเอกสาร books.xml โดยหนังสือเล่มนั้นจะต้องมาจากสำนักพิมพ์ Harper and Row

```
for $b in doc("books.xml") //book
where $b/pubinfo/publisher = "Harper and Row"
return $b/title
ผลลัพธ์ที่ได้คือ
<title>Harold and the Purple Crayon</title>
<title>Harold's Fairy Tale</title>
```

2.6.2.3 order by

order by ใช้ในการกำหนดการเรียงลำดับของข้อมูล และ/หรือชุดของข้อมูล และการใช้งานของ order by นั้นจะต้องถูกกำหนดไว้ก่อนการใช้ return การทำงานในส่วนนี้เปรียบเทียบได้กับการใช้ order by ใน SQL แต่ความพิเศษของ order by ใน XQuery คือสามารถเรียงลำดับข้อมูลที่ไม่มีอยู่ในผลลัพธ์ได้ ดังแสดงในตัวอย่างต่อไปนี้

ตัวอย่างที่ 2.5 ค้นหาชื่อหนังสือจากเอกสาร books.xml โดยหนังสือเล่มนั้นจะต้องมาจากสำนักพิมพ์ Harper and Row และผลลัพธ์ที่ได้จะต้องเรียงลำดับตามปีที่พิมพ์ของหนังสือเล่มนั้น ซึ่งในผลลัพธ์ไม่ต้องแสดงปีที่พิมพ์หนังสือ

```
for $b in doc("books.xml") //book
where $b/pubinfo/publisher = "Harper and Row"
order by $b/pubinfo/year
return $b/title
ผลลัพธ์ที่ได้คือ
<title>Harold and the Purple Crayon</title>
<title>Harold's Fairy Tale</title>
```

2.6.2.4 return

return ใช้ในการกำหนดรูปแบบการแสดงผลของผลลัพธ์ ซึ่ง Element ที่ปรากฏในผลลัพธ์นั้นไม่จำเป็นต้องเป็น Element ที่มีอยู่ในเอกสาร XML ที่ทำการค้นหาก็ได้ และถ้าก่อนหน้า return ไม่ปรากฏ order by อยู่ก่อน การเรียงลำดับของข้อมูลในผลลัพธ์จะเป็นไปตามลำดับในเอกสาร XML ที่ใช้ในการค้นหา หรือเป็นไปตามลำดับที่ได้ถูกกำหนดไว้จากการใช้ชุดคำสั่ง for ดังแสดงในตัวอย่างต่อไปนี้

ตัวอย่างที่ 2.6 ค้นหาชื่อหนังสือจากเอกสาร books.xml โดยหนังสือเล่มนั้นจะต้องมาจากสำนักพิมพ์ Harper and Row และมีการกำหนดโครงสร้างของผลลัพธ์ โดยสร้าง Element ใหม่ที่ไม่มีอยู่ในเอกสาร books.xml

```
for $b in doc("books.xml") //book
where $b/pubinfo/pubisher = "Harper and Row"
return <HR_book> { $b/title } <HR_book>
ผลลัพธ์ที่ได้คือ
<HR_book>
    <title>Harold and the Purple Crayon</title>
</HR_book>
<HR_book>
    <title>Harold's Fairy Tale</title>
</HR_book>
```

2.6.2.5 Operator และฟังก์ชัน Build-in

XQuery มีการกำหนด Operator และฟังก์ชัน Build-in ให้ใช้งาน ซึ่งประกอบไปด้วย

- Operator ทางคณิตศาสตร์ ได้แก่ เครื่องหมายบวก ลบ เป็นต้น
- Operator สำหรับการเปรียบเทียบ ได้แก่ เครื่องหมายมากกว่า น้อยกว่า เป็นต้น
- Operator ในเชิงตรรกะ ได้แก่ and, or เป็นต้น

สำหรับฟังก์ชัน Build-in ที่มีให้ใช้ใน XQuery มีมากกว่า 100 ฟังก์ชัน ซึ่งฟังก์ชันเหล่านี้เป็นฟังก์ชันเกี่ยวกับ String, ตัวเลข, การเปรียบเทียบวันที่และเวลา และอื่นๆ โดย Default prefix ของNamespace ฟังก์ชันคือ fn: ดันนั้นในการเรียกใช้ฟังก์ชัน เราไม่จำเป็นต้องใส่ prefix fn: นำหน้าก็ได้ตัวอย่างของฟังก์ชัน Build-in เช่น

- ฟังก์ชันการหาค่าเฉลี่ย (avg)
- ฟังก์ชันการหาผลรวม (sum)
- ฟังก์ชันการนับจำนวน (count)
- ฟังก์ชันการหาค่าสูงสุด (max)
- ฟังก์ชันการหาค่าต่ำสุด (min)
- ฟังก์ชันสำหรับการอ้างถึงเอกสาร XML ที่ต้องการค้นหาข้อมูล (doc)
- ฟังก์ชันสำหรับการนำแสดงเฉพาะข้อมูลภายในแท็ก (text)
- ฟังก์ชันการตรวจสอบค่าว่าง (empty)
- ฟังก์ชันการตรวจสอบค่าที่มีอยู่ (exists)
- ฟังก์ชันการหาวันที่จากวันเวลา (day-from-dateTime)

ตัวอย่างที่ 2.7 ค้นหาราคาหนังสือที่แพงที่สุดจากเอกสาร books.xml

let $b :=doc("data/books.xml") //book

return <HR_book> { **max($b/pubinfo/price)** } </HR_book>

ผลลัพธ์ที่ได้คือ

<HR_book>15.45</HR_book>

การเรียกใช้ฟังก์ชันสามารถเรียกใช้ได้ในที่ต่อไปนี้

- เรียกใช้ฟังก์ชันใน Element เช่น

```
<name>{uppercase($booktitle)}</name>
```

รูปที่ 2.47 ตัวอย่างการเรียกใช้ฟังก์ชันใน Element

- เรียกใช้ฟังก์ชันใน Predicate ของ Path expression

```
doc("books.xml")/bookstore/book[substring(title,1,5)='Harry']
```

รูปที่ 2.48 ตัวอย่างการเรียกใช้ฟังก์ชันใน Predicate ของ Path expression

- เรียกใช้ฟังก์ชันใน let clause

```
let $name := (substring($booktitle,1,4))
```

รูปที่ 2.49 ตัวอย่างการเรียกใช้ฟังก์ชันใน let clause

2.6.3 การกำหนดเงื่อนไขสำหรับการแสดงผลของผลลัพธ์

การกำหนดเงื่อนไขสำหรับการแสดงผลของผลลัพธ์จะถูกใช้ภายใน return เพื่อให้กำหนดเงื่อนไขในการแสดงค่าผลลัพธ์ ซึ่งจะต้องอยู่ในรูปแบบของ if...then...else ดังแสดงในตัวอย่างต่อไปนี้

ตัวอย่างที่ 2.8 ค้นหาชื่อหนังสือจากเอกสาร books.xml โดยในผลลัพธ์จะแสดงปีที่พิมพ์หนังสือถ้าหนังสือเล่มนั้นพิมพ์หลังจากปี 1955 แต่จะแสดงราคาหนังสือถ้าหนังสือเล่มนั้นพิมพ์ก่อน หรือพิมพ์ในปี 1955

```
for $b in doc("books.xml") //book
return <HR_book> {
    $b/title,
    if ($b/pubinfo/year > "1955") then $b/pubinfo/year
    else $b/pubinfo/price
    } </HR_book>
```

ผลลัพธ์ที่ได้คือ

```
<HR_book>
    <title>Harold and the Purple Crayon</title>
    <price>4.76</price>
</HR_book>
<HR_book>
    <title>Harold's Fairy Tale</title>
    <year>1956</year>
</HR_book>
<HR_book>
    <title>Rise Up Singinng</title>
    <year>1988</year>
</HR_book>
```

2.6.4 การสร้างฟังก์ชัน

นอกจากฟังก์ชัน Build-in แล้ว XQuery ยังเปิดให้ผู้ใช้สามารถสร้างฟังก์ชันขึ้นใช้งานเองได้ โดยในการสร้างฟังก์ชันหนึ่งๆ นั้นจะต้องเป็นไปตาม Syntax ดังนี้

```
declare function prefix:function_name($parameter AS datatype)
  AS returnDatatype
{

(: ...function code here... :)

};
```

รูปที่ 2.50 Syntax ของการสร้างฟังก์ชัน

ในการใช้ฟังก์ชันที่สร้างขึ้นเองนั้นต้อง

- ใช้ keyword ว่า declare function
- ชื่อของฟังก์ชันต้องมี Prefix
- โดยส่วนใหญ่แล้ว ชนิดข้อมูลของ Parameter จะเหมือนกับชนิดข้อมูลที่มีการระบุแล้วใน XML Schema
- ส่วนเนื้อหาของฟังก์ชันต้องอยู่โดยวงเล็บปีกกาเปิดและปิด

ตัวอย่างของฟังก์ชันที่สร้างขึ้นเองซึ่งประกาศไว้ใน Query แสดงดังรูปที่ 2.51

```
declare function local:minPrice(
  $price as xs:decimal?,
  $discount as xs:decimal?)
  AS xs:decimal?
{
let $disc := ($price * $discount) div 100
return ($price - $disc)
};

(: Below is an example of how to call the function above :)

<minPrice>{local:minPrice($book/price, $book/discount)}</minPrice>
```

รูปที่ 2.51 ตัวอย่างของฟังก์ชันที่สร้างขึ้นเอง

2.6.5 การกำหนดเงื่อนไขในการจำกัดจำนวนข้อมูล

การใช้เงื่อนไขประเภทนี้ใช้สำหรับการค้นหาข้อมูลในลักษณะดังนี้ คัดเลือกข้อมูลที่มีเพียงบาง Element ตรงกับเงื่อนไขที่ต้องการ หรือคัดเลือกข้อมูลที่ทุกๆ Element จะต้องตรงกับ

เงื่อนไขที่ต้องการ โดยใช้คำสั่ง some หรือ every ตามลำดับ สำหรับการใช้งานด้วยตำสั่งทั้งสองนั้นจะต้องตามด้วยคำสั่ง satisfies ดังแสดงในตัวอย่างต่อไปนี้

ตัวอย่างที่ 2.9 ค้นหาชื่อหนังสือจากเอกสาร books.xml โดยที่จะต้องมีหนังสืออย่างน้อย 1 เล่มที่พิมพ์หลังจากปี 1970

```
let $b in doc("books.xml") //book
where some $y in $b//year satisfies $y > "1970"
return $b/title
ผลลัพธ์ได้คือ
<title>Harold and the Purple Crayon</title>
<title>Harold's Fairy Tale</title>
<title>Rise Up Singing</title>
```

ตัวอย่างที่ 2.10 ต้นหาชื่อหนังสือจากเอกสาร books.xml โดยที่หนังสือทุกๆ เล่มจะต้องพิมพ์หลังจากปี 1955

```
let $b in doc("books.xml") //book
where every $y in $b//year satisfies $y > "1955"
return $b/title
ผลลัพธ์ที่ได้คือ ไม่มีคำตอบสำหรับการค้นหานี้
```

2.6.6 การค้นหาข้อมูลภายในเอกสาร XML ที่มีมากกว่า 1 เอกสาร

ในกรณีที่ต้องทำการค้นหาข้อมูลจากเอกสาร XML 2 เอกสารขึ้นไป เพื่อนำผลลัพธ์ที่ได้จากทั้งสองเอกสารมารวมกันเป็นเอกสารผลลัพธ์เดียว โดยการรวมผลลัพธ์จากทั้งสองเอกสารนั้นจะต้องมีการกำหนดเงื่อนไขเพื่อให้เอกสารทั้งสองอ้างอิงถึงในสิ่งเดียวกัน ซึ่งสามารถเปรียบเทียบได้กับการค้นหาข้อมูลจาก 2 ตารางขึ้นไป (Join) ใน SQL นั่นเอง ดังแสดงในตัวอย่างต่อไปนี้

ตัวอย่างที่ 2.11 ค้นหาชื่อหนังสือ และชื่อผู้แต่งจากเอกสาร books.xml รวมกับการค้นหาคำวิจารณ์หนังสือเล่มนั้นจากเอกสาร reviews.xml

```
for $b in doc("books.xml") //book,
    $r in doc("reviews.xml") //book
where $b/title = $r/title
return <HR_book> {
        $b/title,
        $b/author,
        $r/review
    } </HR_book>
```

ผลลัพธ์ที่ได้คือ

```
<HR_book>
        <title>Rise Up Singing</title>
        <author>
            <lastname>Blood</lastname>
            <firstname>Peter</firstname>
        </author>
        <author>
            <lastname>Patterson</lastname>
            <firsnme>Annie</firstname>
        </suthor>
        <review>This is great!</review>
<HR_book>
```

2.6.7 การทำงานกับ Attribute

ในกรณีที่เอกสาร XML ที่ต้องการค้นหามี Attribute เป็นส่วนประกอบหนึ่งในเอกสาร ด้วยนั้น สามารถนำมาใช้เป็นข้อมูลหนึ่งในการกำหนดเงื่อนไข และการแสดงผลได้ดังนี้

2.6.7.1 การกำหนดเงื่อนไขด้วย Attribute

XQuery สามารถค้นหาข้อมูลโดยมีการกำหนดเงื่อนไขในการกรองข้อมูลด้วย Attribute ดังแสดงในตัวอย่างต่อไปนี้

ตัวอย่างที่ 2.12 ค้นหาชื่อหนังสือจากเอกสาร books.xml โดยหนังสือเล่มนั้นจะต้องมี Attribute number มากกว่า 2000

```
for $b in doc("books.xml") //book
where $b/@number > "2000"
return <HR_book> { $b/title } </HR_book>
ผลลัพธ์ที่ได้คือ
<HR_bool>
    <title>Rise Up Singing</title>
</HR_book>
```

2.6.7.2 การแสดง Attribute ในผลลัพธ์

XQuery สามารถแสดง Attribute ใน Element ที่สร้างใหม่ของการแสดงผลลัพธ์ ดังแสดงในตัวอย่างต่อไปนี้

ตัวอย่างที่ 2.13 ค้นหาชื่อหนังสือจากเอกสาร books.xml โดยกำหนดโครงสร้างของผลลัพธ์โดยสร้าง Element ใหม่ที่ไม่มีอยู่ในเอกสาร และภายใน Element ใหม่นี้แสดงข้อมูลของ Attribute number ด้วย

```
for $b in doc("books.xml") //book
return
<HR_book> { $b/@number }
    { $b/title }
</HR_book>
ผลลัพธ์ที่ได้คือ
<HR_book number="1001">
    <title>Harold and the Purple Crayon</title>
</HR_book>
<HR_book number="1002">
    <title>Harold's Fairy Tale</title>
</HR_book>
<HR_book number="2001">
    <title>Rise Up Singing</title>
</HR_book>
```

2.6.8 การแสดงผลการ Query ในรูปแบบ HTML

อ้างอิงการเอกสาร XML ในรูปที่ 2.41 เมื่อเขียน XQuery FLWOR expression ดังรูปข้างล่างนี้

```
for $x in doc("books.xml")/bookstore/book/title
order by $x
return $x
```

รูปที่ 2.52 XQuery FLWOR expression

Expression รูปด้านบนนี้จะทำการเลือก Element ที่ชื่อ title ที่อยู่ภายใต้ Element ที่ชื่อ book ที่อยู่ภายใต้ Element ที่ชื่อ bookstore มาทั้งหมด และ Return Element ที่ชื่อ title เรียงตามลำดับอักษร

ถ้าต้องการ List รายการ title ของ book ใน bookstore ในรูปแบบ HTML เราจะเพิ่มแท็ก <ul> และ <li> เข้าไปใน FLWOR expression ดังรูปที่ 2.53

```
<ul>
{
for $x in doc("books.xml")/bookstore/book/title
order by $x
return <li>{$x}</li>
}
</ul>
```

รูปที่ 2.53 FLWOR expression ที่แสดงผลในรูปแบบ HTML

เมื่อเพิ่มแท็ก <ul> และ <li> เข้าไปใน FLWOR expression แล้ว จะได้ผลลัพธ์ดังนี้

```
<ul>
<li><title lang="en">Everyday Italian</title></li>
<li><title lang="en">Harry Potter</title></li>
<li><title lang="en">Learning XML</title></li>
<li><title lang="en">XQuery Kick Start</title></li>
</ul>
```

รูปที่ 2.54 การ List รายการ title ของ book ใน bookstore ในรูปแบบ HTML

ถ้าเราไม่ต้องการให้แสดง Element ที่ชื่อ title ออกมา แต่ต้องการให้แสดงแค่ข้อมูลที่อยู่ระหว่าง Element ที่ชื่อ title ให้เพิ่ม data () ดังรูปที่ 2.55

```
<ul>
{
for $x in doc("books.xml")/bookstore/book/title
order by $x
return <li>{data($x)}</li>
}
</ul>
```

รูปที่ 2.55 FLWOR expression ที่แสดงผลเฉพาะส่วนที่เป็นข้อมูลในรูปแบบ HTML

ผลลัพธ์จากการเพิ่ม data() จะได้ดังรูปนี้

```
<ul>
<li>Everyday Italian</li>
<li>Harry Potter</li>
<li>Learning XML</li>
<li>XQuery Kick Start</li>
</ul>
```

รูปที่ 2.56 การ List เฉพาะส่วนที่เป็นข้อมูลของ title ของ book ใน bookstore ในรูปแบบ HTML

2.7 XQuery Core

XQuery Core เป็นรูปแบบภาษาในเชิงฟังก์ชันที่มีพื้นฐานบนพีชคณิต ซึ่ง XQuery Expression ที่ได้รับการตรวจสอบว่าถูกต้องตามไวยากรณ์แล้วจะถูกแปลงให้อยู่ในรูปของ XQuery Core เพื่อนำไปประมวลหาผลลัพธ์ต่อไป โดยรูปแบบของ XQuery Core ก็คือ NestedmFor-Loop ดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 2.14 การแปลง Expression การเข้าถึงข้อมูลให้เป็น XQuery Core

```
/bib/book
แปลงเป็น XQuery Core ได้ดังนี้
FOR $v1 IN /bib RETURN
    FOR $v2 IN NODES ($v1) RETURN
        TYPESWITCH ($v2) AS $v3
            CASE ELEMENT book {ANYTYPE}
                RETURN $v3
            DEFAULT RETURN ( )
คำอธิบาย
ลูปนอกสุดจะเป็นการวนซ้ำทุกๆ โหนดใน /bib ด้วยตัวแปร $v1 ซึ่งจากเอกสาร books.xml จะเห็น
ว่า /bib นั้นเป็น Root Element อยู่แล้ว เพราะฉะนั้นจะเป็นการวนลูปเพียงแค่ครั้งเดียว และสำหรับ
ลูปในจะเป็นการวนซ้ำภายใน $v1 อีกทีเพื่อหาว่ามี Element ใดใน $v1 ที่มี Element เป็น book
```

จากตัวอย่างข้างต้นจะเป็นตัวอย่างง่ายๆ ในการแปลง Expression การเข้าถึงข้อมูลของ XQuery ให้อยู่ในรูปของ XQuery Core แต่นอกจากการใช้ Nested For-Loop นั้นยังมีการใช้ If...Then...Else มาเสริมในส่วนของการแปลง XQuery Expression ด้วย ซึ่งจะถูกใช้ในการกรองข้อมูลด้วย WHERE ดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 2.15 การแปลง XQuery Expression ให้อยู่ในรูปของ Xauery Core

```
for $b in doc("books.xml")/bib/book,
    $r in doc("reviews.xml")/reviews/book
where $b/title = $r/title
return <HR_book> { $b/title,$b/author,$r/review } <HR_book>
```

แปลงเป็น XQuery Core ได้ดังนี้

```
FOR $b in NODES (doc("books.xml")/bib) RETURN
    FOR $r in NODES (doc("reviews.xml")/reviews) RETURN
        NOT (EMPTY(
            FOR $v1 IN $b/title RETURN
                FOR $v2 IN $r/title RETURN
                    IF EQ($v1,$v2) THEN
                        ELEMENT book { $b/title,$b/author,Sr/review }
                    ELSE ( ) ) )
```

2.8 แบบจำลองกระบวนการทำงานของ XQuery (XQuery Processing Model)

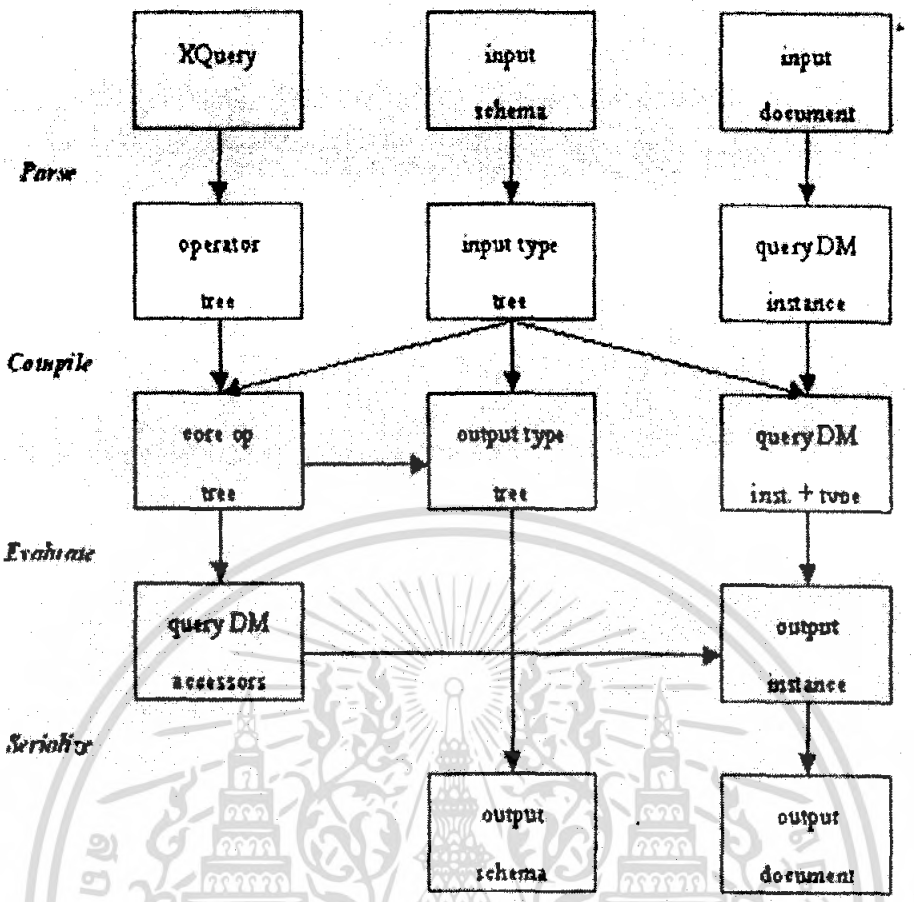
XQuery Processing Model ประกอบด้วย 4 ขั้นตอน ซึ่งในแต่ละขั้นตอนจะนำผลลัพธ์ที่ได้จากขั้นตอนก่อนหน้ามาเป็นอินพุต เพื่อนำมาเข้าสู่กระบวนการทำงานเพื่อให้ได้เอาท์พุตไปทำงานในขั้นตอนต่อไป ดังแสดงในรูปที่ 2.57 ซึ่งรายละเอียดในแต่ละขั้นตอนมีดังนี้

2.8.1 Parse มีหน้าที่ในการตรวจสอบความถูกต้องของ XQuery Expression, XML Schema และ XML Document หลังจากตรวจสอบความถูกต้องเรียบร้อยแล้ว จะสร้างโครงสร้าง Operator ในรูปแบบต้นไม้ สร้างโครงสร้างเอกสารในรูปแบบต้นไม้ และสร้างโครงสร้างข้อมูลเอกสารในรูปแบบต้นไม้ตามลำดับ

2.8.2 Compile มีหน้าที่แปลง XQuery Expression ให้อยู่ในรูปของ XQuery Core และนำโครงสร้างเอกสารกับเอกสาร XML มาตรวจสอบความถูกต้องร่วมกัน

2.8.3 Evaluate มีหน้าที่นำ XQuery Core มาประมวลผลเพื่อค้นหาข้อมูลผลลัพธ์ และสร้างโครงสร้างผลลัพธ์

2.8.4 Serialize มีหน้าที่นำข้อมูลผลลัพธ์ และโครงสร้างผลลัพธ์มาสร้างเป็นเอกสาร XML



รูปที่ 2.57 แบบจำลองการทำงานของ XQuery

บทที่ 3
ทฤษฎีที่เกี่ยวข้องกับ RDF

3.1 Resource Description Framework (RDF)

W3C ได้สร้างมาตรฐาน RDF ขึ้นมาเพื่อให้เป็นมาตรฐานในการสร้าง Metadata ให้กับทรัพยากรที่มีอยู่มากมายบน World Wide Web เพราะ Internet ในปัจจุบันมีการเจริญเติบโตที่รวดเร็ว ทำให้ข้อมูลที่มีอยู่นั้นมากมายจนเกินที่กำลังของมนุษย์จะเข้าไปค้นหาข้อมูลที่ต้องการท่ามกลางที่มีอยู่มากมายนั้นได้ ซึ่ง RDF จะทำให้สิ่งที่มีอยู่มากมายนั้นมีคำอธิบายไปในตัว ทำให้สามารถจัดการได้อย่างอัตโนมัติไม่ต้องให้คนมาจัดการกับทรัพยากรเหล่านี้ด้วยตนเอง

3.1.1 ประวัติของ RDF

ในเดือนตุลาคม ปี 1997 ได้เริ่มมีการร่างข้อกำหนดของ RDF ขึ้นมา

ในเดือนพฤศจิกายน ปี 1997 ได้มีการแนะนำ RDF Metadata

ในเดือนกุมภาพันธ์ ปี 1999 ข้อกำหนดของ RDF ในเรื่องของโครงสร้าง และ Syntax ก็ได้ออกมา

ในเดือนสิงหาคม ปี 1999 RDF Interest Group ได้ก่อตั้งขึ้นมา

ในเดือนมีนาคม ปี 2000 RDF Schema Specification 1.0 ได้ถูกเผยแพร่สู่สาธารณะ

3.1.2 Metadata คือ อะไร

Metadata คือ ข้อมูลที่ใช้อธิบายข้อมูลหนึ่ง ซึ่งในที่นี้ก็เจาะจงกับเนื้อหาที่อยู่บน Web นั้นก็คือ link หรือ URI ต่างๆ นั่นเอง ขอบเขตของ Metadata กับ Data นั่นไม่อาจจะแบ่งแยกได้อย่างชัดเจน ขึ้นอยู่กับ Application ที่ใช้งานว่าจะมองเห็นเป็นอะไร ซึ่งในบางกรณีก็อาจจะมองเป็นได้ทั้งสองอย่างพร้อมกัน

3.1.3 แนวคิดในการออกแบบ RDF

RDF ถูกสร้างขึ้นมาโดยมีแนวคิดดังนี้

1 มีโครงสร้างที่ไม่ซับซ้อน

2 having formal semantics and provable inference

3 สามารถใช้ไวยากรณ์ของ extensible URI-based ได้

4 ใช้ syntax ของ XML เป็นพื้นฐาน

5 รองรับการใช้งานประเภทข้อมูลแบบ XML Schema

6 สามารถให้ใครก็ได้สร้าง Statement ให้กับทรัพยากรใดๆ ก็ได้

3.1.4 RDF ประกอบด้วยอะไร

1 Resource คือ URI เช่น http://www.google.co.th/, urn:isbn:1930110111

2 Property คือ สิ่งที่ใช้อธิบาย Resource เช่น Title, Name

3 Statement คือ องค์ประกอบรวมของ Resource, Property และค่าของ Property

3.1.5 ประเภทข้อมูลที่มีอยู่ใน RDF

ประเภทข้อมูลที่ใช้กันมีอยู่ 2 อย่างหลักๆ ได้แก่

1 Resource ซึ่งก็คือ URI ต่างๆ นั่นเอง

2 Literal ก็คือ Datatype ต่างๆ เช่น ข้อความ ค่าความจริง หรือตัวเลข ซึ่งจะถูกกำหนดตาม XML Schema

3.1.6 ประโยชน์ของ RDF

1 ทำให้สามารถจัดการทรัพยากรที่มีอยู่ได้ง่ายขึ้น โดยสามารถใช้ระบบอัตโนมัติในการจัดการทรัพยากรแทนมนุษย์ได้ เช่น แทนที่จะใช้คนมานั่งเปิดเว็บดูทีละเว็บแล้วก็ค่อยจัดกลุ่มเว็บ ก็เปลี่ยนมาใช้โปรแกรมมาอ่านข้อมูลส่วนที่เป็น RDF แล้วก็วิเคราะห์ว่าเว็บไซต์นี้เป็นแบบใด ซึ่งทำให้การทำงานเร็วขึ้นและเป็นการลดภาระที่จะให้มนุษย์จัดการเองให้น้อยลง

2 ทำให้สามารถใช้ทรัพยากรร่วมกันระหว่าง Application ได้ เพราะ RDF ได้มีการกำหนดมาตรฐานในการจัดการกับข้อมูลในแต่ละด้านขึ้น เช่น Dublin Core เป็นรูปแบบที่ใช้ในการทำ metadata กับหนังสือ ซึ่งก็จะทำให้โปรแกรมค้นหาหนังสือต่างๆ สามารถทำงานกับฐานข้อมูลร่วมกันได้ หรือจะมาตรฐาน RSS (RDF Site Summary) ที่เป็นมาตรฐานในการแลกเปลี่ยนข่าวสารระหว่างกัน ซึ่งในปัจจุบันก็มีโปรแกรมที่สามารถอ่าน RSS ได้เป็นจำนวนมาก เป็นต้น นอกจากนี้ เรายังสามารถสร้างรูปแบบของ RDF ขึ้นมาได้เองอีกด้วย

3 เป็นมาตรฐานที่ถูกสร้างขึ้นมาเพื่อสนับสนุนแนวคิดของ Semantic Web ให้เป็นจริง

3.1.7 การเขียน RDF

สามารถเขียนได้ 2 แบบ คือ

1 RDF/XML หรือแบบมาตรฐาน เป็นแบบที่ใช้พื้นฐานมาจาก XML

```
<rdf:RDF
  xmlns:FOAF="http://xmlns.com/foaf/0.1/"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rev="http://amk.ca/xml/review/1.0#">
  <!-- Implies rdf:type property is rev:Review -->
  <rev:Review rdf:about="http://example.com/rev1">
    <rev:subject rdf:resource="urn:isbn:1930110111"/>
  </rev:Review>
  <rdf:Description rdf:about="http://example.com/author/0042">
    <FOAF:firstName>Bob</FOAF:firstName>
    <FOAF:homepage rdf:resource="http://www.snee.com/bob/"/>
    <FOAF:pastProject rdf:resource="urn:isbn:1930110111"/>
    <FOAF:surname>DuCharme</FOAF:surname>
  </rdf:Description>
</rdf:RDF>
```

รูปที่ 3.1 การเขียน RDF แบบมาตรฐาน

2 Notation-3 หรือ N3 ซึ่งจะอ่านและเขียนง่ายกว่าแบบมาตรฐาน RDF/XML

```
@prefix rev: <http://amk.ca/xml/review/1.0#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix FOAF: <http://xmlns.com/foaf/0.1/> .
<http://example.com/author/0042>
   FOAF:firstName "Bob";
   FOAF:surname "DuCharme";
   FOAF:homepage <http://www.snee.com/bob/>;
   FOAF:pastProject <urn:isbn:1930110111> .
<http://example.com/rev1> rev:subject [
  = <urn:isbn:1930110111>;
  dc:title "XSLT Quickly";
  dc:creator <http://example.com/author/0042>;
  dc:publisher "Manning" ] .
```

รูปที่ 3.2 การเขียน RDF แบบ Notation-3

ในการเขียน RDF แต่ละครั้งจะต้อง Load XML Namespace จาก http://www.w3.org/1999/02/22-rdf-syntax-ns# ทุกครั้งโดย Namespace นี้จะเก็บ Syntax ที่ใช้ใน RDF แบบมาตรฐานแล้วหลังจากนั้นก็จะ Load Namespace ที่ต้องการจะใช้งาน ซึ่งในปัจจุบันก็มี Namespace หลายตัวให้เลือกใช้ ได้แก่

1 Dublin Core ใช้กันมากในงานห้องสมุด

- Namespace url: http://purl.org/dc/elements/1.1/
- Describes book
- Properties: title, creator, publisher, subject, identifier

2 FOAF (Friend-of-a-friend)

- Namespace url: http://xmlns.com/foaf/0.1/
- Describes people
- Classes: Person
- Properties: name, interest, mbox, schoolHomepage, workplaceHomepage

3 DOAP (Description of a Project)

- Namespace url: http://usefulinc.com/ns/doap#
- Describes open source projects

- Classes: Project, Repository
- Properties: name, homepage, mailing-list, license, maintainer

Namespace เหล่านี้มีการใช้งานจริงอยู่ตาม Internet ซึ่งทำให้เมื่อใช้งาน Namespace เหล่านี้แล้ว Application ที่สร้างมาสำหรับใช้กับ Namespace เหล่านี้ก็จะใช้ได้กับงานของเราได้ด้วย และสามารถที่จะทำให้ Resource ที่เรามีอยู่สามารถใช้งานร่วมกันผู้อื่นได้อีกด้วย แต่ถ้าไม่พอใจ Namespace ที่มีอยู่แล้ว ต้องการสร้างใหม่ขึ้นมาเองก็สามารถที่จะทำได้

3.1.8 RDF Graph คือ อะไร

รูปที่ 3.3 RDF Graph

คือ แผนภาพที่ใช้แสดงโครงสร้างของ RDF ที่ใช้งานอยู่ โดยส่วนประกอบของ RDF Graph มีอยู่ 3 อย่างคือ

1 Subject คือ หัวข้อหรือ Resource ที่ต้องการจะอธิบาย โดย Resource ที่ต้องการอธิบายจะต้องเป็น URI เท่านั้น เช่น http://www.ku.ac.th/ หรือจะเป็น Blank node ก็ได้

2 Predicate หรือ Property คือ คุณลักษณะของ Resource อย่างเช่น ที่อยู่ รหัสไปษณีย์

3 Object คือค่าของ Property เช่น ที่อยู่ก็จะมีค่าเป็น Kasetsart U. เป็นต้น หรือ อาจจะเป็น Resource ที่เป็น URI ก็ได้และอาจจะเป็น Blank node ก็ได้

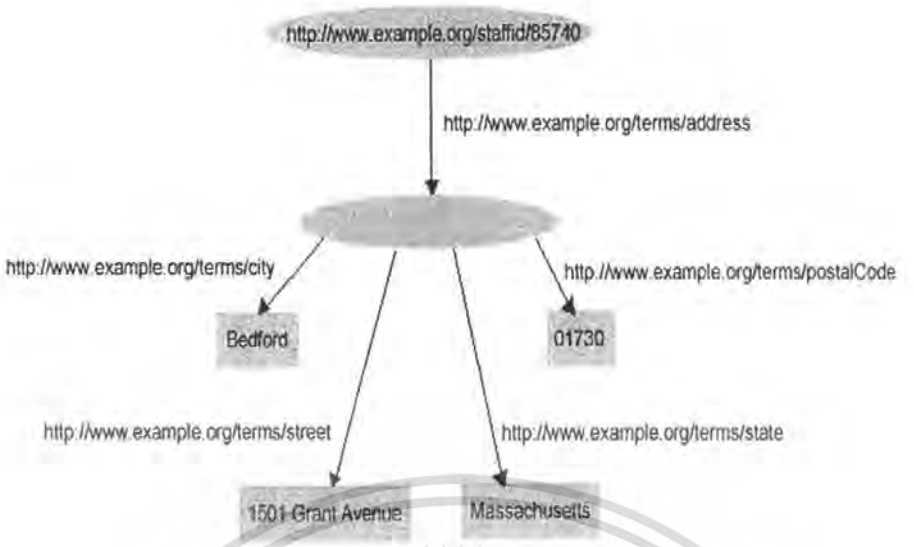
3.1.9 การเขียนกราฟ

1 Predicate เป็นลูกศรที่ชี้จาก Subject ไปยัง Object ที่ต้องการดังรูปข้างต้น

2 Datatype แบบ Resource หรือ URI จะใช้วงรี

3 ส่วน Datatype แบบ Literal จะใช้รูปสี่เหลี่ยมผืนผ้า

4 ฉะนั้น Subject ทุกๆ อัน จะต้องใช้รูปวงรีอย่างแน่นอน เพราะ Subject ต้องเป็น URI เท่านั้น



รูปที่ 3.4 แผนภาพกราฟ

จาก http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/

รูปข้างบนนี้สามารถอธิบายเราได้ดังนี้

1. http://www.example.org/staffid/85740 เป็น Subject ที่เราต้องการอธิบาย
2. http://www.example.org/staffid/85740 มี Property ที่ได้กำหนดไว้ที่ http://www.example.org/terms/address
3. http://www.example.org/terms/address นั้นมี property แยกย่อยไปอีก 4 อัน คือ
   a. http://www.example.org/terms/city ซึ่งเป็นชื่อของเมืองซึ่งจากรูปก็จะได้ Object เป็น Bedford
   b. http://www.example.org/terms/street ซึ่งเป็นชื่อของถนนซึ่งจากรูปก็จะได้ Object เป็น 1501 Grant Avenue
   c. http://www.example.org/terms/state ซึ่งเป็นชื่อของรัฐซึ่งจากรูปก็จะได้ Object เป็น Massachusetts
   d. http://www.example.org/terms/postalCode ซึ่งเป็นรหัสไปรษณีย์ซึ่งจากรูปก็จะได้ Object เป็น 01730

จากแผนภาพสามารถแปลงเป็นโค๊ด RDF/XML ได้ดังนี้

```
<?xml version="1.0"?>
<rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:staff="http://www.example.org/terms/">
        <rdf:Description rdf:about="http://www.example.org/staffid/85740">
                <staff:address rdf:NodeID="address1"/>
        </rdf:Description>
        <rdf:Description rdf:NodeID="address1">
                        <staff:city>Bedford</staff:city>
                        <staff:street>1501 Grant Avenue</staff:street>
                        <staff:state>Massachusetts</staff:state>
                        <staff:postalCode>01730</staff:postalCode>
        </rdf:Description>
</rdf:RDF>
```

รูปที่ 3.5 RDF/XML แบบที่ 1 ที่ได้จากรูปที่ 3.4

หรือ

```
<?xml version="1.0"?>
<rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:staff="http://www.example.org/terms/">
        <rdf:Description rdf:about="http://www.example.org/staffid/85740">
                <staff:address>
                        <staff:city>Bedford</staff:city>
                        <staff:street>1501 Grant Avenue</staff:street>
                        <staff:state>Massachusetts</staff:state>
                        <staff:postalCode>01730</staff:postalCode>
                </staff:address>
        </rdf:Description>
</rdf:RDF>
```

รูปที่ 3.6 RDF/XML แบบที่ 2 ที่ได้จากรูปที่ 3.4

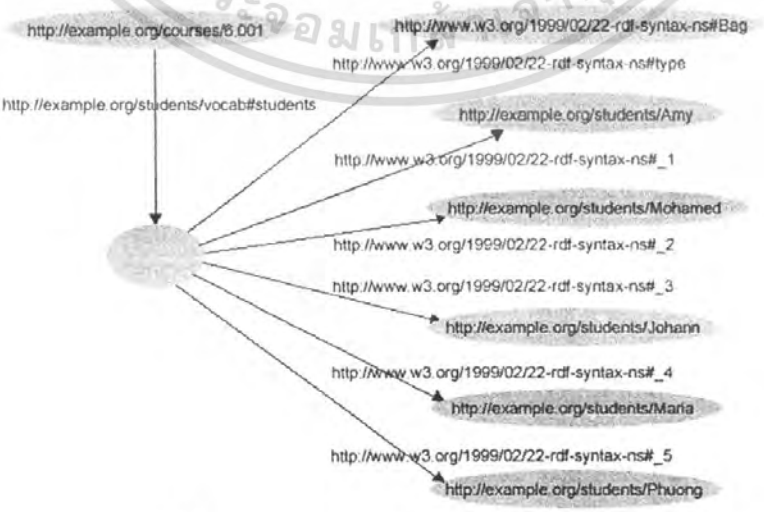
3.1.10 Container

RDF Containers คือ Resource ที่มีการรวมหลายๆ Object เข้าด้วยกันให้กลายเป็นกลุ่มๆ หนึ่ง เช่น กลุ่มของ นักเรียน มีอยู่ 3 ประเภทด้วยกัน คือ

1 Bag (rdf:Bag) เป็นการจัดกลุ่มแบบธรรมดา โดยไม่มีการเรียงลำดับและอาจจะมีการซ้ำซ้อนกันก็ได้

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:s="http://example.org/students/vocab#">
  <rdf:Description rdf:about="http://example.org/courses/6.001">
    <s:students>
      <rdf:Bag>
        <rdf:li rdf:resource="http://example.org/students/Amy"/>
        <rdf:li rdf:resource="http://example.org/students/Mohamed"/>
        <rdf:li rdf:resource="http://example.org/students/Johann"/>
      </rdf:Bag>
    </s:students>
  </rdf:Description>
</rdf:RDF>
```

รูปที่3.7 ตัวอย่างการใช้ RDF Containers



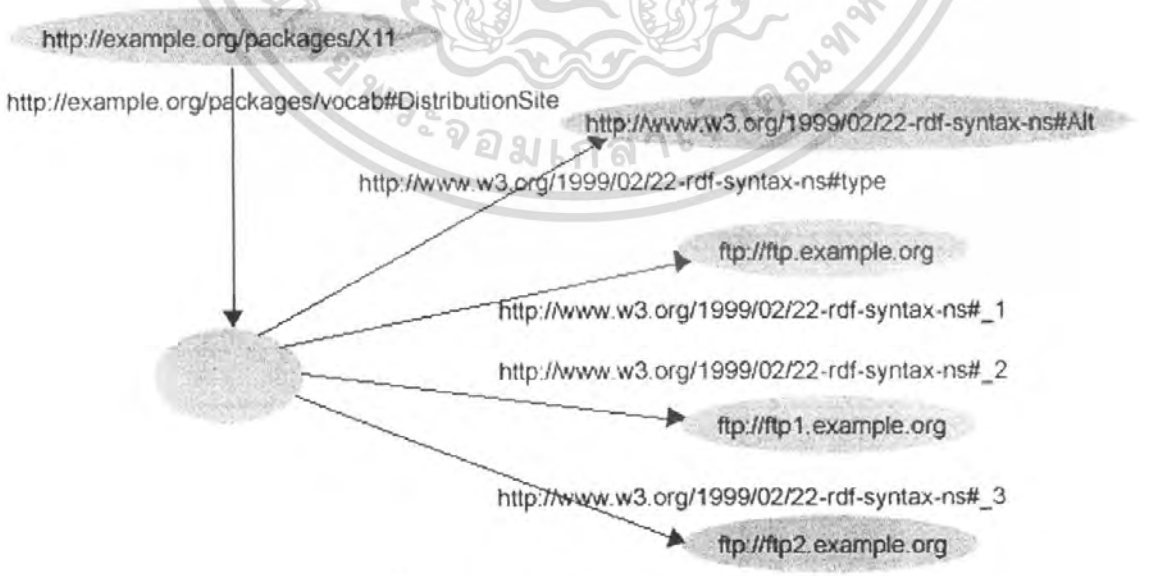
รูปที่ 3.8 แผนภาพการใช้ RDF Containers

2 Sequential (rdf:Seq) เป็นการรวมกลุ่มที่มีลักษณะคล้ายกับ Bag รวมไปถึงการใช้งานก็จะเหมือน Bag เพียงแต่ภายในกลุ่มใน rdf:seq จะมีการจัดเรียงค่าด้วย

3 Alternative (rdf:Alt) เป็นรูปแบบของ Container ที่ค่าในตัวจะไม่มีค่าใดเลยที่ซ้ำกัน

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:s="http://example.org/packages/vocab#">
  <rdf:Description rdf:about="http://example.org/packages/X11">
    <s:DistributionSite>
      <rdf:Alt>
        <rdf:li rdf:resource="ftp://ftp.example.org"/>
        <rdf:li rdf:resource="ftp://ftp1.example.org"/>
        <rdf:li rdf:resource="ftp://ftp2.example.org"/>
      </rdf:Alt>
    </s:DistributionSite>
  </rdf:Description>
</rdf:RDF>
```

รูปที่ 3.9 ตัวอย่างการใช้ Alternative



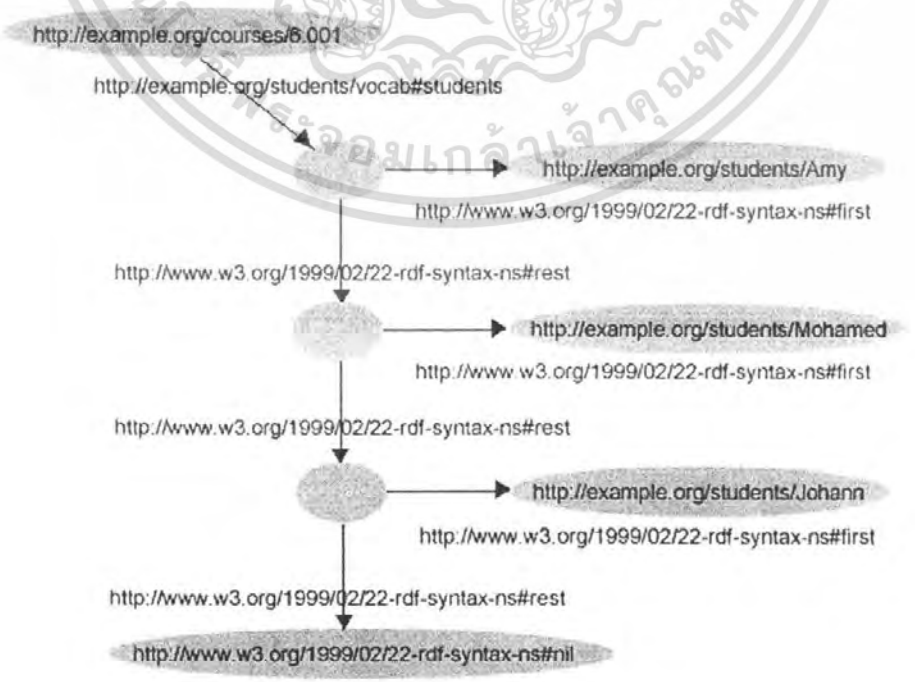
รูปที่ 3.10 แผนภาพการใช้ Alternative

3.1.11 Collection

นอกจากจะมี Container ให้ใช้แล้ว RDF ยังมี Collection ให้ใช้ โดย Collection จะมีข้อแตกต่างจาก Container ตรงที่ Collection จะมีสมาชิกที่เลือกเท่านั้น ขณะที่ Container นั้นจะสนแต่ว่า ตัวมันมีอะไรบ้างเท่านั้น ไม่ได้นึกถึงว่าสมาชิกของมันจะมีอะไรหรือไม่

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:s="http://example.org/students/vocab#">
  <rdf:Description rdf:about="http://example.org/courses/6.001">
    <s:students rdf:parseType="Collection">
      <rdf:Description rdf:about="http://example.org/students/Amy"/>
      <rdf:Description rdf:about="http://example.org/students/Mohamed"/>
      <rdf:Description rdf:about="http://example.org/students/Johann"/>
    </s:students>
  </rdf:Description>
</rdf:RDF>
```

รูปที่ 3.11 ตัวอย่างการใช้ Collection



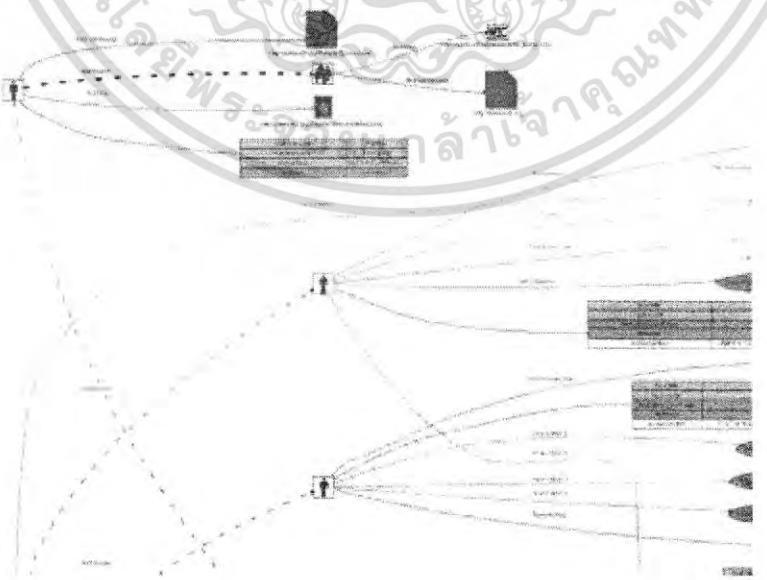
รูปที่ 3.12 แผนภาพการใช้ Collection

3.1.12 GSS (Graph Style Sheet)

เป็นการสร้างรูปแบบของ Graph โดยมีจุดประสงค์เพื่อทำให้กราฟดูมีความชัดเจนมากขึ้น มีความคล้ายคลึงกับ CSS ที่ใช้ใน HTML โดยสามารถใส่สี เปลี่ยนแปลง font เปลี่ยนสีของเส้น เปลี่ยนลักษณะของเส้น ทำการรวมหลายๆ property กลายเป็นตาราง เปลี่ยนลักษณะของกรอบ และใส่รูปแบบ Bitmap ลงไป

ตัวอย่าง

รูปที่ 3.13 กราฟก่อนทำการใช้ GSS

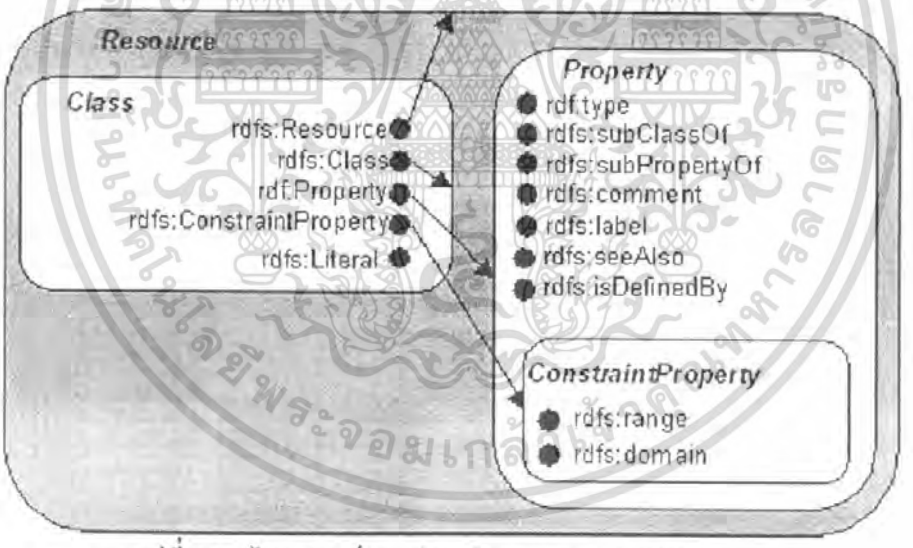


รูปที่ 3.14 กราฟหลังการใช้ GSS

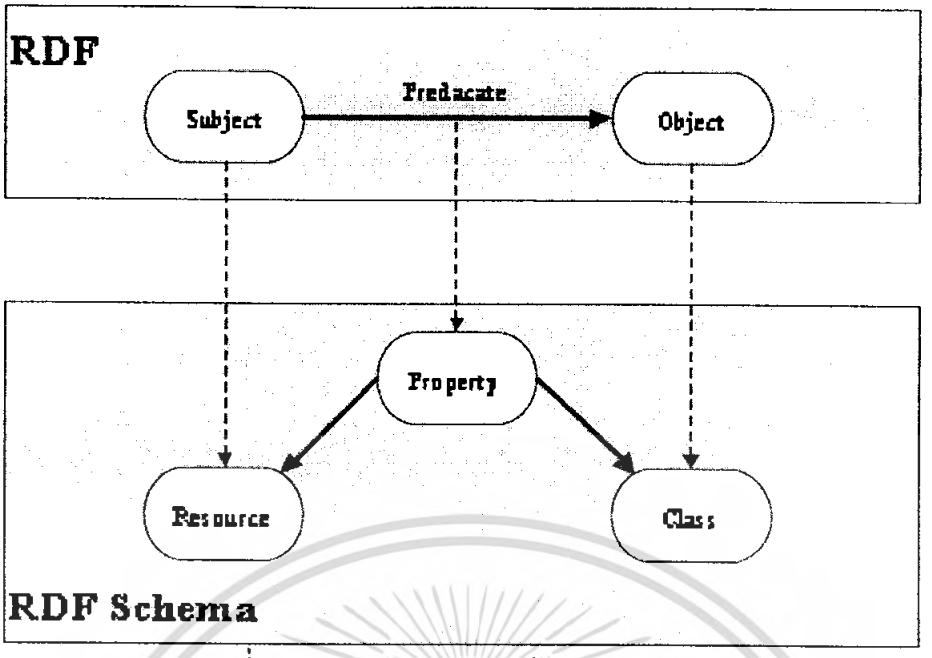
จะเห็นได้ว่าเมื่อมีการใช้ GSS เข้ามาช่วย สามารถทำให้กราฟที่ได้นั้นดูง่ายขึ้น จากเดิมที่มีแต่เส้นสีเขียวลากไปลากมาเหมือนกันหมด ก็สามารถทำเป็นเส้นประเพื่อเน้นถึงว่าเป็นการอ้างอิงไปสู่โหนดอื่นๆ หรือ property ที่มีเยอะแยะก็สามารถรวมเป็นตารางหนึ่งตาราง และนอกจากนี้ยังมีการใช้ภาพมาแทนกรอบธรรมดาทำให้สามารถสื่อความหมายได้ดีขึ้น

3.2 RDF Schema (RDFS)

RDF Schema เป็นการอธิบายโครงสร้างของ Metadata ในการอธิบาย Resource โดยกำหนด Vocabulary เพื่ออธิบายโครงสร้างของ Metadata ซึ่งประกอบไปด้วย คุณสมบัติและค่าของคุณสมบัติ โดยอธิบายขยายให้อยู่ในรูปของโหนดที่เป็น Property และ Class โดยสามารถสืบทอดเป็น SubProperty และ SubClass ได้ ตามลำดับ โดยส่วนประกอบของ RDFS ประกอบด้วยส่วนที่ใช้ในการนิยามคลาส และส่วนที่ใช้ในการนิยามคุณลักษณะ และส่วนที่เป็นข้อจำกัดของข้อมูล สามารถอธิบายได้ดังรูปที่ 3.15 และในการอธิบายโครงสร้างของ Metadata จะสามารถอธิบายได้ดังรูปที่ 3.16



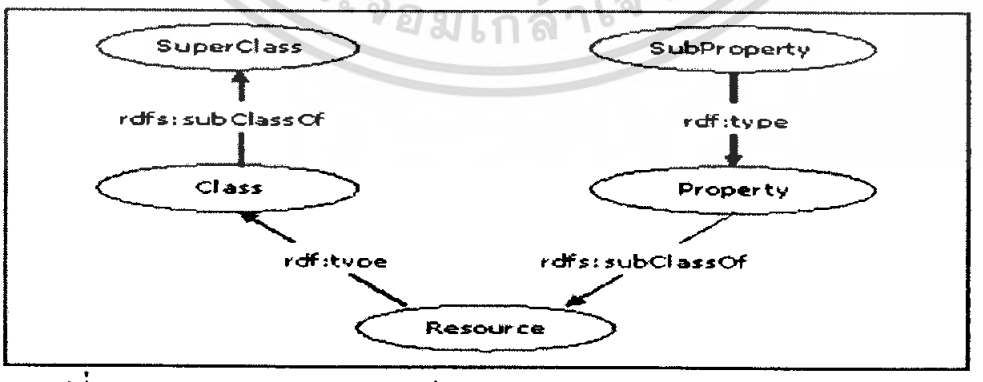
รูปที่ 3.15 ไวยากรณ์และส่วนประกอบของ RDF Schema



รูปที่ 3.16 การอธิบายโครงสร้างของ Metadata

จากรูปที่ 3.16 เป็นการนำโครงสร้าง Metadata ในการอธิบายข้อมูล ซึ่งประกอบไปด้วยคุณสมบัติและค่าของคุณสมบัติมากำหนดให้เป็นโหนดเพื่ออธิบาย Resource โดยส่วน Predicate ซึ่งเป็นคุณสมบัติจะถูกกำหนดมาเป็นโหนด Property และส่วนของ Object ซึ่งเป็นค่าของคุณสมบัติจะถูกกำหนดให้เป็นโหนดที่เป็น Class โดยทั้งโหนด Property และ Class จะสามารถสืบทอดเป็น SubProperty และ SubClass ตามลำดับ

เมื่อทำการอธิบายส่วนของ Metadata ให้กลายเป็นโหนด Property และโหนด Class รวมถึงมีการสืบทอดเป็น SubClass และ SubProperty จากนั้นจะมีการจัดวางความสัมพันธ์ระหว่างโหนดในรูปแบบของ RDF Schema ซึ่งมีความสัมพันธ์กันดังรูปที่ 3.17

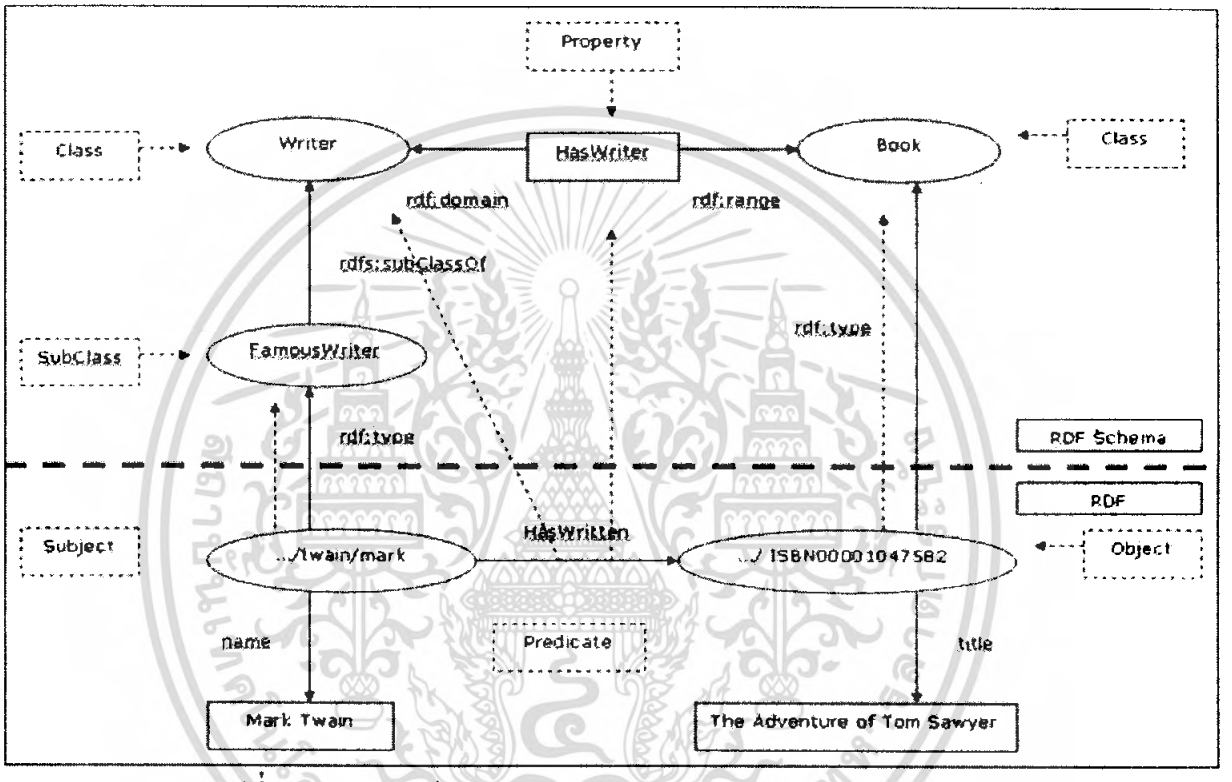


รูปที่ 3.17 การอธิบายความสัมพันธ์ของ Resource ในรูปแบบของ RDF Schema

จากรูปที่ 3.17 แสดงให้เห็นถึงการจัดวางความสัมพันธ์ของแต่ละโหนดในรูปแบบของ RDF Schema โดยโหนด Resource กับโหนด Class มีความสัมพันธ์เป็น rdf:type, โหนด Resource

กับโหนด Property มีความสัมพันธ์กันเป็น rdfs:subClassOf, โหนด Class และ SubClass มีความสัมพันธ์กันเป็น rdfs:subClassOf และ โหนด Property และ SubProperty มีความสัมพันธ์กันเป็น rdf:type

ในการสร้าง RDF Schema นั้นจะกำ หนด Vocabulary เพิ่มเข้าไปเพื่ออธิบายโครงสร้างของ Metadata ที่อธิบายในโครงสร้าง RDF เพื่อให้กลายเป็นข้อมูลที่มีความหมายขึ้นมาดังตัวอย่างข้อมูลในรูปที่ 3.18



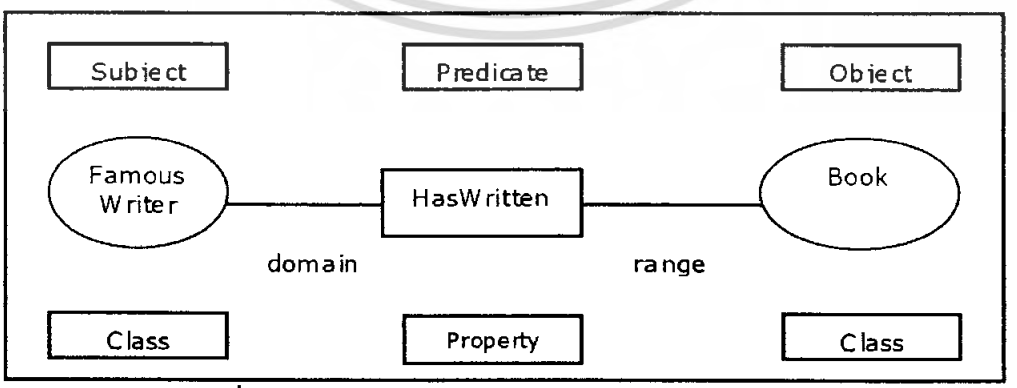
รูปที่ 3.18 ตัวอย่างข้อมูลในการกำหนดส่วนของ RDF Schema

จากรูปที่ 3.18 จะประกอบไปด้วย 2 ส่วน (แบ่งโดยเส้นจุดเส้นประ) ส่วนแรกเป็นส่วนการอธิบายข้อมูลโดยแทนข้อมูลในโครงสร้าง RDF ซึ่งอยู่ในส่วนล่าง เพื่ออธิบายว่า Resource ที่มี URI เป็น .../twain/mark มีคุณสมบัติเป็น hasWritten และมีค่าของคุณสมบัติเป็น Resource ../ISBN00001047582 และมีการอธิบายอีกว่า Resource ที่มี URI เป็น .../twain/mark นั้นมีคุณสมบัติเป็น name ซึ่งมีค่าของคุณสมบัติเป็นส่วน Literal ที่มีค่าเป็น "Mark Twain" และ Resource ../ISBN00001047582 มี title เป็นคุณสมบัติ และมีค่าของคุณสมบัติเป็นส่วน Literal ที่มีค่าเป็น "The Adventure of Tom Sawyer "

จะเห็นได้ว่าจากการอธิบายในส่วนล่างซึ่งเป็นการอธิบายข้อมูลโดยแทนข้อมูลในโครงสร้างการอธิบายข้อมูล RDF เพื่ออธิบายว่า Resource ที่มี URI เป็น .../twain/mark มีคุณสมบัติและค่าของคุณสมบัติที่สัมพันธ์กับสิ่งใด แต่สิ่งเหล่านี้ไม่ได้บ่งบอกถึงความหมายที่ Resource นั้น

เป็นอยู่โดยแท้จริง ดังนั้นจึงต้องมีการอธิบายในส่วนของ RDF Schema เพิ่มเข้าไปเพื่อทำ ให้ข้อมูลประกอบไปด้วยความหมายขึ้นมา

ในการอธิบายในส่วนของ RDF Schema นั้น จะทำการกำหนดส่วนที่เรียกว่า Vocabulary เพื่ออธิบายขยายส่วนของโครงสร้าง Metadata ออกไป โดยเริ่มจากส่วนของ Subject นั่นคือ ส่วนของ Resource .../twain/mark ซึ่งสามารถอธิบายโดยทำให้ Resource อยู่ในรูปของ Class FamousWriter ตามลูกศรเส้นประ และตามหลักการของ RDF Schema ระหว่าง Resource และ Class จะมีความสัมพันธ์กันเป็น rdf:type ต่อมาพิจารณาในส่วนของ Predicate HasWritten ซึ่งเป็นคุณสมบัติ สามารถอธิบายโดยทำให้คุณสมบัติอยู่ในรูปของ Class Writer และ Property HasWritten ตามลูกศรเส้นประ และส่วนสุดท้ายคือส่วนของ Object ซึ่งคือส่วนของ Resource .../ISBN00001047582 ซึ่งสามารถอธิบายโดยทำให้ Resource อยู่ในรูปของ Class Book ตามลูกศรเส้นประ ซึ่งก็จะมีความสัมพันธ์กันเป็น rdf:type กับส่วนของ Object ดังนั้น Class และ Property ทั้งหมดที่เกิดขึ้นจากการอธิบายโครงสร้าง Metadata คือ Class จะมีทั้งหมด 3 Class คือ Class FamousWriter , Class Writter และ Class Book ส่วน Property จะมี 1 Property คือ Property HasWritter โดยในการจัดวางความสัมพันธ์นั้น Class FamousWriter จะกลายเป็น SubClass ของ Class Writer ซึ่งตามหลักการของ RDF Schema จะมีความสัมพันธ์กันเป็น rdfs:subClassOf ส่วน Property HasWriter จะถูกจัดวางให้อยู่ระหว่าง Class Writer และ Class Book โดย Property HasWriter จะถูกกำหนดให้ใช้ Domain และ Range ซึ่งจะถูกใช้ในกรณีที่จะอธิบายความสัมพันธ์ของ Class 2 Class ซึ่งถูกคั่นด้วยโหนด Property ซึ่งมีลักษณะคล้ายกับโครงสร้างของ RDF ที่ประกอบไปด้วย Subject, Predicate และ Object โดยโหนด Class ด้านหนึ่งเป็นส่วนของ Subject และ Class และ Class อีกด้านหนึ่งเป็นส่วนของ Object และ มีโหนด Property ที่ถูกคั่นอยู่จะเป็นส่วนของ Predicate ในโครงสร้าง RDF ซึ่งสามารถอธิบายความสัมพันธ์ระหว่างโหนดได้ดังรูปที่ 3.19



รูปที่ 3.19 การอธิบายความสัมพันธ์ของ Class 2 Classes

จากรูปที่ 3.19 domain จะถูกใช้เพื่อบอกความสัมพันธ์ระหว่างโหนด Property และ Class ที่เป็น Resource ที่ถูกอธิบายหรือในส่วนของ Subject จากในตัวอย่างคือ Resource (FamousWriter) เป็น class มีความสัมพันธ์กับ Property (hasWritten) ส่วน range จะถูกใช้ เพื่อบอกความสัมพันธ์ระหว่าง โหนด Property ไปยัง Class ที่เป็นค่าของคุณสมบัติหรือในส่วนของ Object จากในตัวอย่างคือ Object (Book) ซึ่งเป็น Class มีความสัมพันธ์ กับ Property (hasWritten) จากกราฟในรูปที่ 5 เราสามารถนำไปเขียนเป็น RDF Document ได้ดังรูปที่ 3.20

```
<?xml version="1.0"?>
<rdf:RDF
        xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs ="http://www.w3.org/2000/01/rdf-schema#"
        xmlns:s ="http://www.bookonline.com"
<rdf:Description rdf:about="http://www.famouswriters.org/twain/mark">
        <s:hasName>Mark Twain</s:hasName>
        <s:hasWritten rdf:resource="http://www.books.org/ISBN0001047582">
        <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#FamousWriter">
</rdf:Description>
<rdf:Description rdf:about="http://www.books.org/ISBN0001047582">
        <s:title>The Adventure of Tom Sawyer</s:title>
        <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Book">
</rdf:Description>
<rdf:Description rdf:about=" http://www.w3.org/2000/01/rdf-schema#FamousWriter">
        <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Writer">
</rdf:Description>
<rdf:Description rdf:about="http://www.movieworld.com/schema#hasWritten">
        <rdfs:domain rdf:resource=" http://www.w3.org/2000/01/rdf-schema#Writer "/>
        <rdfs:range rdf:resource=" http://www.w3.org/2000/01/rdf-schema#Book "/>
</rdf:Description>

</rdf:RDF>
```

รูปที่ 3.20 RDF Document ของกราฟในรูปที่ 3.19

จากรูปที่ 3.20 ส่วนของเอกสารที่อยู่ในกรอบสี่เหลี่ยมเส้นประจะเป็นส่วนของการกำหนด Vocabulary เพื่ออธิบายโครงสร้างของ Metadata ในส่วนของ RDF Schema ซึ่งจะบ่งบอกถึงความหมายของข้อมูล ลักษณะเฉพาะ และความสัมพันธ์ระหว่างกลุ่มของคุณสมบัติ และกลายเป็นข้อมูลที่มีความหมาย

3.3 OWL (Web Ontology Language)

OWL (Web Ontology Language) เป็นภาษาที่ใช้ในการอธิบายและยกตัวอย่าง Web ontology ซึ่ง Ontology เป็นเทอมที่ยืมมาจากหลักปรัชญาทางวิทยาศาสตร์เกี่ยวกับการบรรยายลักษณะของสิ่งที่มีอยู่จริงในโลก และสิ่งเหล่านั้นสัมพันธ์กันอย่างไร โดย OWL ontology อาจจะรวมไปถึงการบรรยายเกี่ยวกับ class เช่น ลักษณะและตัวอย่างของ class นั้นๆ ยกตัวอย่าง ontology

ในส่วนของ OWL formal semantics จะระบุรายละเอียดเกี่ยวกับวิธีการ derive เพื่อให้ได้ผลลัพธ์ทางตรรกะ (logical)

3.3.1 The Species of OWL (ชนิดของ OWL)

ภาษา OWL แบ่งเป็น 3 ชนิด ตามลักษณะการออกแบบ เพื่อการใช้งาน โดยระบุรายละเอียดของกลุ่มอุปกรณ์และผู้ใช้ ดังนี้

1. OWL Lite สนับสนุนต่อผู้ใช้จำนวนมากที่ต้องการแบ่งหมวดหมู่ตามลำดับชั้น (classification hierarchy) และง่ายต่อการจำกัดลักษณะเฉพาะ ตัวอย่างเช่น ในขณะที่ OWL Lite สนับสนุน cardinality constraints มันจะอนุญาตให้ค่าของ cardinality เป็น 0 หรือ 1 เท่านั้น ซึ่ง การเตรียม tool ที่ support กับ OWL Lite จะง่ายกว่าการทำงานโดยใช้ลักษณะความสัมพันธ์ และจัดเตรียมวิธีการที่รวดเร็ว และเทคนิคอื่น ๆ เกี่ยวกับการแบ่งประเภท
2. OWL DL สนับสนุนต่อผู้ใช้ที่ต้องการความชัดเจนสูงสุดโดยไม่มีการสูญเสียในการดำเนินการที่สมบูรณ์ และการตัดสินใจของระบบการให้เหตุผล โดย OWL DL รวมไปถึงภาษา OWL ที่สร้างด้วยการกำหนดขอบเขต เช่น การแบ่งประเภท ดังนั้น OWL DL จึงเป็นการตั้งชื่อที่เหมาะสม สอดคล้องกับการอธิบายทางตรรกะ (description logics) โดยในส่วนของการวิจัยที่ได้มีการศึกษารายละเอียดนั้นสามารถตัดสินใจได้ด้วย first order logic ซึ่งจะสามารถกล่าวได้ว่า OWL DL นั้น ออกแบบมาเพื่อให้ support กับการอธิบายทางตรรกะในส่วนของธุรกิจที่มีอยู่ และมีลักษณะในการดำเนินการที่ถูกต้องสำหรับระบบการให้เหตุผล
3. OWL Full มุ่งไปที่ผู้ใช้ที่ต้องการความชัดเจนสูงสุด และความไม่มีกฎเกณฑ์เกี่ยวกับ syntax ของ RDF กับการที่ไม่มีการรับประกันในการดำเนินการ โดย OWL Full จะยอมให้ใช้ ontology ขยายความหมายในการอธิบายกลุ่มคำศัพท์ ซึ่งไม่น่าเป็นไปได้ว่า ซอฟต์แวร์ที่ใช้ในการให้เหตุผลจะสามารถ support กับทุกลักษณะของ OWL Full

แต่ละส่วนย่อยของภาษาเป็นขอบเขตของพื้นฐานที่เกิดขึ้นมาก่อน เกิดขึ้นมาจากคำถามที่ว่า อะไรที่ทำให้สามารถแสดงออกมาได้อย่างถูกต้อง และอะไรที่สามารถสรุปเหตุผลนั้นได้ การติดตามเซตของความสัมพันธ์ที่ได้ไม่สามารถทำกลับกันได้

ทุกๆ กฎของ OWL Lite เป็นส่วนหนึ่งของ OWL DL

ทุกๆ กฎของ OWL DL เป็นส่วนหนึ่งของ OWL Full

ทุกข้อสรุปที่เป็นเหตุเป็นผลของ OWL Lite เป็นข้อสรุปที่เป็นเหตุเป็นผลของ OWL DL ด้วย

ทุกข้อสรุปที่เป็นเหตุเป็นผลของ OWL DL เป็นข้อสรุปที่เป็นเหตุเป็นผลของ OWL Full ด้วยเช่นกัน

ผู้พัฒนา ontology ที่นำ OWL มาใช้ควรจะคำนึงถึงชนิดที่ดีที่สุดเหมาะสมกับความต้องการ โดยการเลือกระหว่าง OWL Lite และ OWL DL จะขึ้นอยู่กับขอบเขตที่ผู้ใช้ต้องการเกี่ยวกับการกำหนดลักษณะภายใต้เงื่อนไขของ OWL DL การให้เหตุผลโดย OWL Lite จะมีลักษณะการดำเนินการที่ต้องการ การให้เหตุผลโดย OWL DL จะใช้กับกรณีที่มีความซับซ้อน ส่วนการเลือกระหว่าง OWL DL และ OWL Full โดยทั่วไปจะขึ้นอยู่กับขอบเขตที่ผู้ใช้ต้องการเกี่ยวกับ meta-modeling ที่สะดวกของ RDF Schema

ผู้ใช้เปลี่ยนจาก RDF มาเป็น OWL DL หรือ OWL Lite นั้นต้องการการรับรองว่าเอกสาร RDF เดิมนั้นทำตามเงื่อนไขที่กำหนดโดย OWL DL หรือ OWL Lite

เมื่อเรากล่าวถึงเพียง OWL DL หรือ OWL Full เท่านั้น มันจะชัดเจนได้โดยใช้ OWL DL

3.3.2 Structure of the Document (โครงสร้างของเอกสาร)

ในลำดับการเตรียมกลุ่มตัวอย่างที่ถูกต้องตั้งแต่เริ่ม เรามีการสร้างสิ่งที่เป็น wine และ food นี่เป็น OWL DL ontology ซึ่งบางครั้งการโต้แย้งหาเหตุผลของเราจะมุ่งจุดสนใจไปที่ประสิทธิภาพของ OWL Full และมันจะชัดเจนมากขึ้น wine และ food ontology มีลักษณะสำคัญที่ดัดแปลงมาจากส่วนสำคัญของ DAML ontology library กับประวัติศาสตร์ที่ยาวนานโดยดั้งเดิมนั้นพัฒนาโดย McGuinness ซึ่งเป็นตัวอย่าง CLASSIC description logic , การขยายเพื่อการสอนเกี่ยวกับ description logic และ เป็นการขยายเพื่อการสอนเกี่ยวกับ ontology

3.3.3 The Structure of Ontologies (โครงสร้างของ Ontologies)

OWL เป็นส่วนประกอบของกิจกรรมของ Semantic Web ซึ่งมีจุดมุ่งหมายในการสร้างทรัพยากรของ Web ให้สามารถเข้าไปใช้ได้อย่างรวดเร็วเป็นกระบวนกระแบบอัตโนมัติ โดยการเพิ่มข้อมูลเกี่ยวกับทรัพยากรที่บรรยายหรือจัดเตรียมเนื้อหาของ Web ซึ่งโดยปกติแล้ว Semantic Web เป็น web ที่มีความกระจายของแหล่งข้อมูล OWL ต้องยอมให้มีการจัดเก็บรวบรวมข้อมูลจากแหล่งข้อมูลที่กระจายอยู่นั้น นี่เป็นส่วนที่ยอมให้ ontology มีความสัมพันธ์กัน รวมทั้งมีการนำข้อมูลเข้าจาก ontology อื่น ๆ อย่างชัดเจนด้วย

นอกจากนี้ OWL ทำให้เกิดการเปิดเผยข้อสันนิษฐานของโลก ซึ่งนี่เป็นการบรรยายเกี่ยวกับทรัพยากรที่ไม่ได้จำกัดให้มีเพียงไฟล์เดียวหรือขอบเขตเดียว ขณะที่คลาส C1 อาจจะถูกอธิบายไว้ตั้งแต่ดั้งเดิมใน ontology O1 มันสามารถถูกขยายเพิ่มขึ้นใน ontology อื่นๆได้ ซึ่งผลที่ได้จากการเพิ่มเติมการอ้างถึงเกี่ยวกับ C1 นั้นเป็น monotonic ซึ่งข้อมูลใหม่ไม่สามารถลดลงหรือถอยกลับไปเป็นข้อมูลก่อนหน้านั้นได้ ข้อมูลใหม่สามารถขัดแย้งกับของเก่าได้ แต่ความจริงและการถ่ายทอดนั้นสามารถเพิ่มได้อย่างเดียว ไม่เคยลบออกไป

ความเป็นไปได้ของความขัดแย้งนั้น ๆเป็นสิ่งหนึ่งที่ผู้ออกแบบ ontology ต้องการที่จะเข้าไปทำการพิจารณา และมันเป็นผลที่คาดหวังว่า tool นั้นจะ support ในการช่วยสืบค้นข้อเท็จจริง

การเขียน ontology ที่สามารถอธิบายความชัดเจน และใช้โดย software agents เราต้องการ syntax และ formal semantics สำหรับ OWL ซึ่ง OWL เป็นกลุ่มคำศัพท์ที่ขยาย RDF Semantics ของ RDF โดย OWL semantics จะอธิบายใน OWL Web Ontology Language Semantics and Abstract Syntax

3.3.3.1 Namespaces

ก่อนที่เราจะสามารถใช้เซตของเทอม เราต้องการที่แสดงความถูกต้องในการกำหนดกลุ่มคำศัพท์ที่จะใช้ มาตรฐานของส่วนประกอบแรกของ ontology ประกอบด้วยเซตของ การประกาศ XML namespace ซึ่งประกอบด้วย tag เปิด rdf:RDF ซึ่งมีวิธีการระบุอย่างชัดเจน และวางไว้ในส่วนของการนำเสนอ ontology ซึ่งจะทำให้เข้าใจได้ง่ายขึ้น ซึ่ง OWL ontology ดั้งเดิมนั้นเริ่มต้นด้วยการประกาศ namespace(namespace declaration) โดยมีลักษณะดังต่อไปนี้ แน่นอนว่า URIs ของการกำหนด ontologies โดยปกติจะไม่ได้อ้างอิงถึง w3.org เสมอไป

```
<rdf:RDF
   xmlns     ="http://www.w3.org/TR/2004/REC-owl-guide-20040210/wine#"
   xmlns:vin ="http://www.w3.org/TR/2004/REC-owl-guide-20040210/wine#"
   xml:base  ="http://www.w3.org/TR/2004/REC-owl-guide-20040210/wine#"
   xmlns:food="http://www.w3.org/TR/2004/REC-owl-guide-20040210/food#"
   xmlns:owl ="http://www.w3.org/2002/07/owl#"
   xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:xsd ="http://www.w3.org/2001/XMLSchema#">
```

รูปที่ 3.21 namespace

การประกาศในสองส่วนแรกจะระบุชื่อ namespace ที่สัมพันธ์กับ ontology ส่วนแรกจะทำการกำหนด default namespace โดยเริ่มจากชื่อที่ไม่มี prefix อ้างอิงไปยัง ontology ปัจจุบัน ส่วนที่สองจะระบุชื่อ namespace ของ ontology ปัจจุบันด้วยprefix vin:

ส่วนที่สามจะระบุ URI เป้าหมายสำหรับ document (ซึ่งจะแสดงตัวอย่างในด้านล่าง) ส่วนที่สี่จะระบุชื่อ namespace ที่ support กับ food ontology ด้วย prefix food:

ส่วนที่ห้าการประกาศ namespace ที่บอกว่าอยู่ใน document โดย elements ที่นำหน้าด้วย owl: ซึ่งควรจะเข้าใจการอ้างอิงไปยังสิ่งที่ต้องการจะดึงจาก namespace ที่เรียก http://www.w3.org/2002/07/owl# นี่เป็นการประกาศ OWL โดยทั่วไป ซึ่งใช้แนะนำเกี่ยวกับ OWL vocabulary

OWL ขึ้นอยู่กับการสร้างคำอธิบายโดย RDF, RDFS, และ XML Schema datatypes ในเอกสารนี้ rdf: เป็น prefix อ้างอิงไปยังสิ่งที่ต้องการจะดึงจาก namespace ที่เรียก http://www.w3.org/1999/02/22-rdf-syntax-ns# สองการประกาศ namespace ถัดไปนั้นสร้าง statements คล้ายกันเกี่ยวกับ RDF Schema (rdfs:) และ XML Schema datatype (xsd:) namespaces

เป็นส่วนที่ช่วยในการเขียน URLs ที่ยาวมากที่มักจะใช้ประโยชน์บ่อยๆ ให้สามารถใช้ได้โดยเตรียมเซตของ entity definitions ในการประกาศประเภท document (DOCTYPE) นำหน้า ontology definitions การกำหนดชื่อด้วยการประกาศ namespace มีความสำคัญเป็นเพียงส่วนหนึ่งของ XML tags เท่านั้น ค่าของ Attribute ไม่ใช่ namespace แต่ใน OWL เรามักจะอ้างอิงถึงตัวบอกลักษณะ ontology โดยใช้ค่าของ attribute อยู่บ่อยๆ ซึ่งสามารถเขียนดึงในรูปแบบที่ขยายเต็มที่ ดังตัวอย่าง "http://www.w3.org/TR/2004/REC-owl-guide-20040210/wine#merlot" การย่อสามารถกำหนดโดยใช้ ENTITY definition ดังตัวอย่าง

```
<!DOCTYPE rdf:RDF [
  <!ENTITY vin  "http://www.w3.org/TR/2004/REC-owl-guide-20040210/wine#" >
  <!ENTITY food "http://www.w3.org/TR/2004/REC-owl-guide-20040210/food#" > ]>
```

รูปที่ 3.22 การใช้งาน ENTITY definition

หลังจากการประกาศ ENTITY คู่นี้ เราสามารถเขียนค่า "&vin;merlot" และ มันจะขยาย ไปยัง http://www.w3.org/TR/2004/REC-owl-guide-20040210/wine#merlot

บางครั้งสิ่งที่สำคัญคือ การประกาศ namespace rdf:RDF สามารถเปลี่ยนแปลงการประกาศ entity ได้ง่ายขึ้น ซึ่งจะเพิ่มส่วนประกอบของ ontology เข้าไป ดังนี้

```
<rdf:RDF
  xmlns    ="&vin;"
  xmlns:vin ="&vin;"
  xml:base  ="&vin;"
  xmlns:food="&food;"
  xmlns:owl ="http://www.w3.org/2002/07/owl#"
  xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:xsd ="http://www.w3.org/2001/XMLSchema#">
```

รูปที่ 3.23 การเพิ่มส่วนประกอบของ ontology เข้าไป

3.3.3.2 Ontology Headers

ในการสร้าง namespaces หนึ่ง โดยทั่วไปเราจะประกอบด้วยการรวบรวมการอ้างอิงเกี่ยวกับคลุ่มของ ontology ภายใต้ tag owl:Ontology ซึ่ง tag นี้สนับสนุนสนุนการ

วิเคราะห์เฉพาะส่วนงานที่เป็น comment, version control และการรวมของ ontology อื่นๆ

```
<owl:Ontology rdf:about="">
 <rdfs:comment>An example OWL ontology</rdfs:comment>
 <owl:priorVersion rdf:resource="http://www.w3.org/TR/2003/PR-owl-guide-20031215/wine"/>
 <owl:imports rdf:resource="http://www.w3.org/TR/2004/REC-owl-guide-20040210/food"/>
 <rdfs:label>Wine Ontology</rdfs:label>
 ...
```

รูปที่ 3.24 Ontology Headers

เราใช้ '...' เพื่อแสดงว่ามีการเพิ่มเติมข้อความที่ไม่พิจารณาถึงเป้าหมายของตัวอย่างนี้

element owl:Ontology เป็นส่วนที่รวบรวม OWL meta-data จำนวนมากสำหรับ document ซึ่งมันไม่ได้รับรองว่า document บรรยาย ontology ในการสืบทอดตามเหตุผลในบางกลุ่มนั้น ontology ไม่ได้มีอยู่อันเดียว แต่มี classes และ properties กำหนด domaim เมื่อใช้ OWL อธิบายการรวบรวมของข้อมูลตัวอย่าง โดย tag owl:Ontology อาจต้องการ

คำสั่งในการบันทึกข้อมูล version และเก็บคำอธิบายของเอกสารไว้ ดังนั้นใน OWL เทอมของ ontology จะมีการขยายเพื่อรวมข้อมูลตัวอย่าง ดังที่เห็นในข้างต้น

attribute rdf:about เป็นชื่อหรือการอ้างอิงถึงontology ที่ค่าของ attribute เป็น "" เป็นกรณีมาตรฐาน ที่ว่าชื่อของ ontology เป็นพื้นฐานของ URI ของ element owl:Ontology เป็นตัวอย่าง URI ของ document ที่บรรจุ ontology ข้อยกเว้น ที่มีการขึ้นต้นด้วยใช้ xml:base อาจจะเป็นการกำหนด URI เริ่มต้นสำหรับ element เพื่อบางอย่างที่ไม่ใช่ URI ของ document ปัจจุบัน

rdfs:comment เป็นการจัดเตรียม ส่วนที่ต้องการให้ชัดเจน เพื่อใช้ให้คำจำกัดความของ ontology

owl:priorVersion เป็น tag มาตรฐาน ซึ่งมุ่งที่จะเตรียมการทำงานของ version control systems กับontologies ส่วนการทำ Ontology versioning จะกล่าวในส่วนต่อไป

owl:imports ใช้สำหรับโครงสร้างในการ include-style ซึ่งowl:importsใช้ argument ตัวเดียว โดยกำหนดด้วย attribute rdf:resource

การนำ ontology อื่นๆ เข้ามา การอ้างอิงถึง ontology ใด จะอ้างจาก ontology นั้นไปยัง ontology ปัจจุบัน

Properties ที่ใช้ให้คำจำกัดความนั้นควรจะประกาศโดยใช้ owl:AnnotationProperty ตัวอย่าง เช่น <owl:AnnotationProperty rdf:about="&dc;creator" />

OWL นั้นมีหลายวิธีการที่จะโยง ontology ปัจจุบัน และนำ ontology อื่นเข้ามารวมกัน ซึ่งจะกล่าวใน ontology mapping

เรามักใช้ rdfs:label เพื่อสนับสนุนคำอธิบายที่เป็นภาษาธรรมชาติสำหรับ ontology ของเราเอง

การกำหนด ontology header จะปิดด้วย tag </owl:Ontology>

ในเบื้องต้น ปัจจุบันการกำหนดให้รู้ว่าสิ้นสุด ontology ทำโดยปิดด้วย tag </rdf:RDF>

3.3.3.3 Data Aggregation and Privacy (การรวมกลุ่มของข้อมูล และข้อมูลที่อยู่เดี่ยวๆ)

OWL สามารถที่จะแสดงข้อมูล ontology เกี่ยวกับตัวอย่างที่ปรากฏในหลายๆ document ที่ support การเชื่อมข้อมูลจากหลายๆ แหล่งด้วยวิธีการที่มีลักการแน่นอน โดยรากฐานของ semantics จะ support กับการอ้างอิงถึงข้อมูลที่อาจจะให้ผลที่รวดเร็ว โดยเฉพาะความสามารถในการแสดงความเท่ากัน โดยใช้ owl:sameAs สามารถใช้ในลักษณะที่บอกความแตกต่างของแต่ละข้อมูลได้เหมาะสม ซึ่งเป็นการแสดงความเท่ากันตามความเป็นจริง โดย Owl:InverseFunctionalProperty สามารถใช้เชื่อมข้อมูลที่อยู่เดี่ยวๆ

เข้าด้วยกัน ยกตัวอย่างเช่น ถ้าลักษณะเช่น "SocialSecurityNumber" เป็น owl:InverseFunctionalProperty เมื่อทำการแยกเป็นข้อมูลเดี่ยวๆ สองครั้ง สามารถทำให้เห็นว่ามีพื้นฐานค่าของ property นั้นๆ เหมือนกัน เมื่อแยกเป็นข้อมูลเดี่ยวๆ ด้วยวิธีการเช่นเดียวกันนั้น ข้อมูลนั้นจากต่างแหล่งกันสามารถรวมกันได้ การรวมกลุ่มนี้สามารถใช้ตัดสินความจริงได้ ซึ่งนั่นไม่ได้เป็นการแสดงจากแหล่งใดแหล่งหนึ่งโดยตรง

ความสามารถของ Semantic Web ในการเชื่อมโยงข้อมูลจากหลายๆ แหล่งเป็นลักษณะที่ต้องการ และมีประโยชน์ได้ผลมาก ซึ่งสามารถใช้ได้ในหลายๆ application อย่างไรก็ตามประสิทธิภาพในการรวมข้อมูลจากหลายแหล่งนั้น รวมกับผลการอนุมานของ OWL มีความเป็นไปได้อย่างมาก ผู้ใช้ OWL ควรเตรียมสิ่งเกี่ยวข้องที่เป็นไปได้ของข้อมูลที่อยู่เดี่ยวๆ การอธิบายรายละเอียดของวิธีการแก้ปัญหาความปลอดภัยเป็นการพิจารณาถึงขอบเขตสำหรับ Working Group

3.3.4 Basic Elements (อิลิเมนต์พื้นฐาน)

elements ส่วนใหญ่ของ OWL ontology เกี่ยวข้องกับ classes, properties, instances ของ classes, และ ความสัมพันธ์ระหว่าง instances นี้ ในส่วนนี้จะแสดงส่วนประกอบของภาษาซึ่งเป็นส่วนประกอบที่สำคัญเพื่อแนะนำถึง elements

3.3.4.1 Simple Classes and Individuals (คลาสอย่างง่าย และข้อมูลเดี่ยวๆ)

มีการใช้ ontology หลายครั้ง ที่จะขึ้นอยู่กับความสามารถในการให้เหตุผลเกี่ยวกับข้อมูลที่อยู่เดี่ยวๆ ในลำดับที่จะทำนี้อยู่ในรูปแบบวิธีการที่เป็นประโยชน์ ที่เราจำเป็นต้องมีวิธีการระบุ classes ที่อยู่เดี่ยวๆ และ properties ที่สืบทอดคุณสมบัติความสัมพันธ์ของ class เราสามารถอ้างถึง properties ที่ระบุเกี่ยวกับข้อมูลที่อยู่เดี่ยวๆ นั้นได้ แต่อำนาจของ ontology ส่วนใหญ่มาจากการให้เหตุผลเกี่ยวกับพื้นฐานของ class (class-based reasoning) บางครั้งเราต้องการที่จะเน้นเกี่ยวกับความแตกต่างระหว่าง class กับ object และ class กับเซตที่บรรจุ elements ต่างๆ เรากล่าวได้ว่า เซตของข้อมูลเดี่ยวๆ เป็นสมาชิกของ class นี่เป็นการขยายความเกี่ยวกับ class

1 Simple Named Classes (การตั้งชื่อคลาสอย่างง่าย)

Class, rdfs:subClassOf

ยกตัวอย่าง class ได้แก่

- owl:Thing หมายถึง ทุกข้อมูลเดี่ยว ๆในโลกของ OWL (OWL world) เป็นสมาชิกของคลาส นี้

- owl:Nothing หมายถึง คลาสว่าง (empty class)

ตัวอย่าง โดเมน wines เราสร้างคลาสของรากของต้นไม้ (three root classes) ได้แก่ Winery, Region, และ ConsumableThing ดังนี้

```
<owl:Class rdf:ID="Winery"/>
<owl:Class rdf:ID="Region"/>
<owl:Class rdf:ID="ConsumableThing"/>
```

รูปที่ 3.25 โดเมน wines

ตัวอย่าง การอธิบายคลาส Wine ซึ่ง Wine เป็น PotableLiquid เรากำหนด Pasta เป็น EdibleThing ด้วย ดัง code ต่อไปนี้

2 Individuals (ข้อมูลเดี่ยวๆ)

ในการเพิ่มเติมเข้าไปในคลาส เราต้องการที่จะสามารถอธิบายสมาชิกของคลาสได้ โดยทั่วไปเรามักคิดเกี่ยวกับข้อมูลเดี่ยว ๆใน universe ของสิ่งต่างๆ ซึ่งข้อมูลเดี่ยวๆ นั้นเป็นไปได้น้อยที่สุดที่จะนำเข้ามาโดยการประกาศให้เป็นสมาชิกของคลาส

ยกตัวอย่าง เช่น <Region rdf:ID="CentralCoastRegion" />

ต่อไปนี้เป็นการอธิบายความหมายของตัวอย่างด้านบน

```
<owl:Thing rdf:ID="CentralCoastRegion" />

<owl:Thing rdf:about="#CentralCoastRegion">
  <rdf:type rdf:resource="#Region"/>
</owl:Thing>
```

รูปที่ 3.26 การอธิบายความหมายของตัวอย่างด้านบน

rdf:type เป็น RDF property ที่โยงข้อมูลเดี่ยวๆ ไปยังคลาส ว่าเป็นสมาชิกของคลาสนั้นๆ

- ตัวอย่าง Grapes ถูกกำหนดอยู่ใน food ontology ดังนี้

```
<owl:Class rdf:ID="Grape">
  ...
</owl:Class>
```

รูปที่ 3.27 ตัวอย่าง Grapes ถูกกำหนดอยู่ใน food ontology

- และตัวอย่าง wine ontology เรามีดังนี้

```
<owl:Class rdf:ID="WineGrape">
  <rdfs:subClassOf rdf:resource="&food;Grape" />
</owl:Class>

<WineGrape rdf:ID="CabernetSauvignonGrape" />
```

รูปที่ 3.28 ตัวอย่าง wine ontology

3 Design for Use (การออกแบบสำหรับใช้งาน)

นี่เป็นส่วนที่สำคัญในความแตกต่างระหว่าง class และ ข้อมูลเดี่ยวๆ ในOWL โดย class นั้นจะชัดเจนที่ชื่อและการรวบรวบ properties ที่บรรยายเซตของข้อมูลเดี่ยวๆ ซึ่งข้อมูลเดี่ยวๆ นั้นเป็นสมาชิกของเซต ดังนั้น class ควรจะมีลักษณะเช่นเดียวกันกับกลุ่มของสิ่งที่มีอยู่ในธรรมชาติ ที่อยู่ใน domain ที่สนใจที่จะบรรยาย

ในการสร้าง ontology มีลักษณะพิเศษที่กล่าวถึงอยู่บ่อยๆ 2 วิธี ได้แก่

- Levels of representation (ระดับของการแทนค่า)
- Subclass vs. instance (คลาสย่อยกับส่วนที่ได้มาจากคลาส)

3.3.4.2 Simple Properties

โลกของคลาสและข้อมูลย่อยๆ จะไม่น่าสนใจมากนัก ถ้าเราสามารถกำหนดเทคนิคในการจัดแบ่งกลุ่มได้ โดยมีลักษณะ (Properties) ที่ทำให้เกิดการอ้างอิงถึงความจริงโดยทั่วไปเกี่ยวกับสมาชิกของคลาส และระบุความจริงเกี่ยวกับข้อมูลต่างๆ

1 Defining Properties (การกำหนด Properties)

ObjectProperty, DatatypeProperty, rdfs:subPropertyOf, rdfs:domain, rdfs:range

- datatype properties เป็นความสัมพันธ์ระหว่าง instances ของ classes และ RDF literals และ XML Schema datatypes

- object properties เป็นความสัมพันธ์ระหว่าง instances ของสอง classes ซึ่ง object properties ไม่ได้เจตนาจะแสดงถึงการติดต่อกับ RDF term rdf:object

2. Properties and Datatypes

เราจะจำแนก properties โดยขึ้นอยู่กับความสัมพันธ์ระหว่างข้อมูลกับข้อมูล (object properties) หรือ ข้อมูลกับ datatypes (datatype properties) ซึ่ง Datatype

properties อาจจะเป็นขอบเขตที่คลอบคลุม RDF literals หรือ การกำหนด simple types ที่สอดคล้องกับ XML Schema datatypes

OWL มักใช้ built-in XML Schema datatypes เป็นส่วนมาก การอ้างอิงถึง datatypes เหล่านี้ทำโดยใช้ URI อ้างอิงสำหรับ datatype คือ http://www.w3.org/2001/XMLSchema ซึ่ง datatypes ดังต่อไปนี้ได้เสนอไว้สำหรับใช้กับ OWL

Datatypes ด้านบน เพิ่ม rdfs:Literal จาก built-in OWL datatypes เข้าไป จะ support กับ xsd:integer และ xsd:string datatypes

built-in XML Schema datatypes อื่นๆ อาจจะใช้ใน OWL Full แต่การเตือนจะระบุในเอกสาร OWL Semantics and Abstract Syntax

```
<owl:Class rdf:ID="VintageYear" />

<owl:DatatypeProperty rdf:ID="yearValue">
  <rdfs:domain rdf:resource="#VintageYear" />
  <rdfs:range  rdf:resource="&xsd;positiveInteger"/>
</owl:DatatypeProperty>
```

รูปที่ 3.29 Properties

Property yearValue สัมพันธ์กับ VintageYears โดยเป็นค่า integer เราจะกล่าวว่า property hasVintageYear เป็นสัมพันธ์ของ Vintage กับ VintageYear

3 Properties of Individuals

อันดับแรกเราจะอธิบายถึง Region และ Winery individuals และเราจะกำหนด wine ก่อน ซึ่งเป็น Cabernet Sauvignon ดังนี้

```
<Region rdf:ID="SantaCruzMountainsRegion">
  <locatedIn rdf:resource="#CaliforniaRegion" />
</Region>

<Winery rdf:ID="SantaCruzMountainVineyard" />

<CabernetSauvignon
  rdf:ID="SantaCruzMountainVineyardCabernetSauvignon" >
  <locatedIn   rdf:resource="#SantaCruzMountainsRegion"/>
  <hasMaker    rdf:resource="#SantaCruzMountainVineyard" />
</CabernetSauvignon>
```

รูปที่ 3.30 อธิบายถึง Region และ Winery individuals

Datatype properties สามารถเพิ่มเข้าไปในข้อมูลในรูปแบบที่คล้ายกัน ดังที่เราอธิบาย instance ของ VintageYear และโยงไประบุค่าของประเภท &xsd:positiveInteger ดังต่อไปนี้

```
<VintageYear rdf:ID="Year1998">
  <yearValue rdf:datatype="&xsd;positiveInteger">1998</yearValue>
</VintageYear>
```

รูปที่ 3.31 อธิบาย instance ของ VintageYear และโยงไประบุค่าของประเภท &xsd:positiveInteger

3.3.4.3 Property Characteristics (ลักษณะของ properties)

ในส่วนถัดไปเป็นส่วนที่อธิบายวิธีการใช้เพื่อระบุ properties ต่อไป เป็นไปได้ที่จะระบุลักษณะของ properties (property characteristics) ซึ่งมีวิธีการที่มีประสิทธิภาพสำหรับปรับปรุงการให้เหตุผลเกี่ยวกับ property ดังนี้

1 TransitiveProperty

ถ้า property P ถูกระบุให้เป็น transitive แล้ว สำหรับ x, y, และ z ใดๆ จะเป็นดังนี้ คือ

P(x,y) and P(y,z) implies P(x,z)

2 SymmetricProperty

ถ้า property P เป็น symmetric แล้ว สำหรับ x และ y ใดๆ จะเป็นดังนี้ คือ

P(x,y) iff P(y,x)

3 FunctionalProperty

ถ้า property P เป็น functional แล้ว สำหรับ x, y, และ z ทั้งหมดจะเป็นดังนี้ คือ

P(x,y) and P(x,z) implies y = z

4 inverseOf

ถ้า property P1 เป็น owl:inverse ของ P2 แล้ว สำหรับ x และ y ทั้งหมดจะเป็นดังนี้ คือ

P1(x,y) iff P2(y,x)

5 InverseFunctionalProperty

ถ้า property P เป็น InverseFunctional แล้ว สำหรับ x, y, และ z ทั้งหมดจะเป็นดังนี้ คือ

P(y,x) and P(z,x) implies y = z

3.3.4.4 Property Restrictions (การกำหนด Property)

การเพิ่มเติมที่จะกำหนดลักษณะเฉพาะของ Property เป็นไปได้ที่จะจำกัดขอบเขตของ property ในการระบุถ้อยคำในหลากหลายวิธีการ เราจะดำเนินการในการกำหนด property ด้วยวิธีการบรรยายหลากหลายรูปแบบดังจะกล่าวต่อไปด้านล่าง ซึ่งสามารถใช้อยู่ภายในส่วนของ owl:Restriction โดย element owl:onProperty จะบอกถึงการกำหนด property

1 allValuesFrom, someValuesFrom

```
<owl:Class rdf:ID="Wine">
 <rdfs:subClassOf rdf:resource="&food;PotableLiquid" />
 ...
 <rdfs:subClassOf>
  <owl:Restriction>
   <owl:onProperty rdf:resource="#hasMaker" />
   <owl:allValuesFrom rdf:resource="#Winery" />
  </owl:Restriction>
 </rdfs:subClassOf>
 ...
</owl:Class>
```

รูปที่ 3.32 การใช้ allValuesFrom

```
<owl:Class rdf:ID="Wine">
 <rdfs:subClassOf rdf:resource="&food;PotableLiquid" />
 <rdfs:subClassOf>
  <owl:Restriction>
   <owl:onProperty rdf:resource="#hasMaker" />
   <owl:someValuesFrom rdf:resource="#Winery" />
  </owl:Restriction>
 </rdfs:subClassOf>
 ...
</owl:Class>
```

รูปที่ 3.33 การใช้ someValuesFrom

2 Cardinality

```
<owl:Class rdf:ID="Vintage">
 <rdfs:subClassOf>
  <owl:Restriction>
   <owl:onProperty rdf:resource="#hasVintageYear"/>
   <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
  </owl:Restriction>
 </rdfs:subClassOf>
</owl:Class>
```

รูปที่ 3.34 การใช้ cardinality

3 hasValue [OWL DL]

```
<owl:Class rdf:ID="Burgundy">
  ...
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:onProperty rdf:resource="#hasSugar" />
    <owl:hasValue rdf:resource="#Dry" />
   </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

รูปที่ 3.35 การใช้ hasValue

3.3.5 Ontology Mapping

ในคำสั่งสำหรับ ontologies ที่ทำให้มีผลมากที่สุดนั้น พวกเขาต้องการแบ่งเป็นหลายส่วนอย่างแพร่หลาย ในคำสั่งที่จะช่วยลดการใช้ความพยายามทางปัญญา รวมทั้งการพัฒนา ontology พวกเขาต้องการที่จะนำกลับมาใช้ใหม่ ในทางที่ดีที่สุดของทุกสิ่งที่เป็นไปได้ในโลก พวกเขาต้องการที่จะสงบ ตัวอย่าง คุณสามารถนำ date ontology จากแหล่งหนึ่งมาใช้ และนำ physical location ontology จากแหล่งอื่นมา แล้วขยายความคิดของ location ให้ประกอบด้วยระยะเวลาในระหว่างที่มันอยู่

นี่เป็นสิ่งที่สำคัญที่จะต้องเข้าใจว่าความพยายามในการพัฒนา ontology ส่วนมาก เป็นความใส่ใจที่จะนำหลายๆ คลาส และหลายๆ property มาเกี่ยวข้องสัมพันธ์กัน ด้วยวิธีการที่จะทำให้มีความเกี่ยวข้องกันมากขึ้น เราต้องการการอ้างอิงที่ง่าย ๆไม่ซับซ้อนเกี่ยวกับจำนวนสมาชิกของคลาสให้มีอย่างกว้างขวาง และทำให้มีความเกี่ยวข้องกันที่เป็นประโยชน์ นี่เป็นส่วนที่ยากของการพัฒนา ontology ถ้าคุณสามารถพบ ontology ที่มีอยู่ ซึ่งครอบคลุม เคยมีการใช้ และกลั่นกรองมาแล้ว นั่นทำให้รู้สึกเข้าใจและนำมันมาใช้ได้เลย

มันจะเป็นการท้าทายที่จะผสม ontology ที่มีการรวบรวมไว้ ซึ่ง tool ที่สนับสนุนเกือบจะมีความแน่นอน เป็นความต้องการที่จะรักษาความสอดคล้องไว้

3.3.5.1 Equivalence between Classes and Properties (ความเท่ากันระหว่าง Classes และ Properties)

equivalentClass, equivalentProperty เป็นการโยงเซตของส่วนประกอบของ ontology ต่างๆ เข้าด้วยกันเป็นส่วนที่สาม ซึ่งมักจะมีประโยชน์ที่จะสามารถแสดงลักษณะเฉพาะของ class หรือ property ใน ontology หนึ่งว่าเท่ากันกับ class หรือ property ในอีก ontology หนึ่ง

```
<owl:Class rdf:ID="Wine">
 <owl:equivalentClass rdf:resource="&vin;Wine"/>
</owl:Class>
```

รูปที่ 3.36 การใช้ equivalentClass

ส่วนในการโยง properties เข้าด้วยกันในรูปแบบที่คล้ายกันนี้ เราจะใช้ owl:equivalentProperty

3.3.5.2 Identity between Individuals (ความเหมือนกันระหว่างข้อมูล)

sameAs วิธีการนี้คล้ายกับวิธีที่ใช้สำหรับคลาส แต่ประกาศสองข้อมูลให้เหมือนกัน

```
<Wine rdf:ID="MikesFavoriteWine">
 <owl:sameAs rdf:resource="#StGenevieveTexasWhite" />
</Wine>
```

รูปที่ 3.37 การใช้ sameAs

และมี hasMaker เป็นฟังก์ชันที่ใช้ในการทำให้ข้อมูลไม่ขัดแย้งกัน

```
<owl:Thing rdf:about="#BancroftChardonnay">
  <hasMaker rdf:resource="#Bancroft" />
  <hasMaker rdf:resource="#Beringer" />
</owl:Thing>
```

รูปที่ 3.38 การใช้ hasMaker

นอกจากปัญหาความขัดแย้งของข้อมูลใน ontology แล้ว มันหมายความอย่างง่ายก็คือ Bancroft = Beringer

3.3.5.3 Different Individuals (ข้อมูลที่แตกต่างกัน)

differentFrom, AllDifferent วิธีการนี้ให้ผลตรงข้ามกับการใช้ sameAs

```
<WineSugar rdf:ID="Dry" />

<WineSugar rdf:ID="Sweet">
  <owl:differentFrom rdf:resource="#Dry"/>
</WineSugar>

<WineSugar rdf:ID="OffDry">
  <owl:differentFrom rdf:resource="#Dry"/>
  <owl:differentFrom rdf:resource="#Sweet"/>
</WineSugar>
```

รูปที่ 3.39 การใช้ differentFrom

```
<owl:AllDifferent>
  <owl:distinctMembers rdf:parseType="Collection">
    <vin:WineColor rdf:about="#Red" />
    <vin:WineColor rdf:about="#White" />
    <vin:WineColor rdf:about="#Rose" />
  </owl:distinctMembers>
</owl:AllDifferent>
```

รูปที่ 3.40 การใช้ AllDifferent

หมายเหตุ: owl:distinctMembers นั้นสามารถใช้ในการรวมร่วมกับ owl:AllDifferent เท่านั้น

3.3.6 Complex Classes [OWL DL] (คลาสที่ซับซ้อน)

OWL มีการเพิ่มเติม constructors กับรูปแบบของคลาส โดย constructors เหล่านี้สามารถใช้เพื่อสร้างคลาสที่เรียกว่า class expressions ซึ่ง OWL สนับสนุนเซตของ operations พื้นฐาน กล่าวคือ union, intersection และ complement โดยใช้ชื่อ owl:unionOf, owl:intersectionOf, และ owl:complementOf ตามลำดับ

หมายเหตุ : class expressions นั้น เป็นกลุ่มที่ไม่มีความต้องการในการตั้งชื่อสำหรับทุกๆ คลาส ซึ่งจะพิจารณาการใช้กลุ่มของ operations เพื่อสร้างคลาสที่ซับซ้อน (Complex Classes) จากคลาสที่ไม่ระบุชื่อ หรือคลาสกับค่าที่กำหนด

3.3.6.1 Set Operators

intersectionOf, unionOf, complementOf

OWL มีวิธีการที่จะจัดการ class extensions โดยใช้ เซตของ operations พื้นฐาน ดังต่อไปนี้

1 Intersection [some uses of OWL DL]

```
<owl:Class rdf:ID="WhiteWine">
 <owl:intersectionOf rdf:parseType="Collection">
  <owl:Class rdf:about="#Wine" />
  <owl:Restriction>
   <owl:onProperty rdf:resource="#hasColor" />
   <owl:hasValue rdf:resource="#White" />
  </owl:Restriction>
 </owl:intersectionOf>
</owl:Class>
```

รูปที่ 3.41 การใช้ประโยค intersectionOf

2 Union [OWL DL]

ตัวอย่างต่อไปนี้แสดงการใช้ประโยค unionOf ซึ่งจะใช้คล้ายกับประโยค intersectionOf

```
<owl:Class rdf:ID="Fruit">
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#SweetFruit" />
    <owl:Class rdf:about="#NonSweetFruit" />
  </owl:unionOf>
</owl:Class>
```

รูปที่ 3.42 การใช้ประโยค unionOf

เมื่อเสร็จสมบูรณ์แล้วจะได้ผลซึ่งมีความหมายเหมือนกับการใช้ union สามารถเขียนอยู่ในรูปแบบที่แตกต่าง ดังนี้

```
<owl:Class rdf:ID="Fruit">
  <rdfs:subClassOf rdf:resource="#SweetFruit" />
  <rdfs:subClassOf rdf:resource="#NonSweetFruit" />
</owl:Class>
```

รูปที่ 3.43 เขียนอยู่ในรูปแบบที่แตกต่าง

จะกล่าวได้ว่า instances ของ Fruit เป็น subset ของ intersection ของ sweet fruit และ non-sweet fruit เราจะคาดได้ว่าผลที่ได้เป็นเซตว่าง

3 Complement [OWL DL]

ประโยค complementOf จะเลือกข้อมูลทั้งหมดจาก domain ของที่ บรรยาย ที่ไม่เป็นส่วนหนึ่งของคลาสนั้น ซึ่งจะใช้เป็นประจำในการอ้างถึงกลุ่มข้อมูลขนาดใหญ่มาก ดังตัวอย่างนี้

```
<owl:Class rdf:ID="ConsumableThing" />

<owl:Class rdf:ID="NonConsumableThing">
  <owl:complementOf rdf:resource="#ConsumableThing" />
</owl:Class>
```

รูปที่ 3.44 ประโยค complementOf

คลาสของ NonConsumableThing ประกอบด้วยสมาชิกของคลาส โดยที่ข้อมูลทั้งหมดไม่เป็นส่วนหนึ่งที่จะทำการขยาย ConsumableThing ซึ่งเซตนี้ประกอบ ด้วย Wines, Regions เป็นต้น นี่เป็นเซตของส่วนต่างระหว่าง owl:Thing และ ConsumableThing เพราะฉะนั้นรูปแบบที่ใช้เป็นตัวอย่างสำหรับ complementOf ที่ใช้อยู่ร่วมกับกลุ่มของ operators อื่นๆ เป็นดังนี้

```
<owl:Class rdf:ID="NonFrenchWine">
 <owl:intersectionOf rdf:parseType="Collection">
  <owl:Class rdf:about="#Wine"/>
  <owl:Class>
   <owl:complementOf>
    <owl:Restriction>
     <owl:onProperty rdf:resource="#locatedIn" />
     <owl:hasValue rdf:resource="#FrenchRegion" />
    </owl:Restriction>
   </owl:complementOf>
  </owl:Class>
 </owl:intersectionOf>
</owl:Class>
```

รูปที่ 3.45 complementOf ที่ใช้ อยู่ร่วมกับกลุ่มของ operators อื่นๆ

3.3.6.2 Enumerated Classes

oneOf [OWL DL]

OWL มีวิธีการที่จะระบุรายละเอียดคลาส via โดยการนับสมาชิกของคลาสโดยตรง จะทำโดยใช้ประโยค oneOf จุดเด่นคือเป็นคำนิยามที่สมบูรณ์ในการระบุ class extension ดังนั้นไม่มีข้อมูลอื่นที่สามารถประกาศเป็นส่วนหนึ่งในคลาสได้

การกำหนดคลาส WineColor ดังต่อไปนี้ สมาชิกของคลาสเป็นข้อมูลเดี่ยวๆ ได้แก่ White, Rose, และ Red

```
<owl:Class rdf:ID="WineColor">
 <rdfs:subClassOf rdf:resource="#WineDescriptor"/>
 <owl:oneOf rdf:parseType="Collection">
  <owl:Thing rdf:about="#White"/>
  <owl:Thing rdf:about="#Rose"/>
  <owl:Thing rdf:about="#Red"/>
 </owl:oneOf>
</owl:Class>
```

รูปที่ 3.46 การกำหนดคลาส WineColor

สิ่งแรกที่ต้องเข้าใจในส่วนนี้เป็นส่วนที่ไม่ใช่ข้อมูลอื่นที่สามารถให้ผล WineColor ที่ต้องการ เนื่องจาก class มีการกำหนดด้วยการนับแต่ละ element ของประโยค oneOf ต้องเป็นข้อมูลเดี่ยวๆ ที่ประกาศไว้ โดย ข้อมูลเหล่านั้นเป็นส่วนหนึ่งของบางคลาส ในตัวอย่างข้างต้นถูกอ้างด้วยชื่อ เราใช้ owl:Thing ซึ่งเป็นความคิดเก่าง่ายๆ ในการแนะนำการอ้างอิงในทางเลือกที่ดีที่สุด เราสามารถอ้างอิงถึง elements ของเซตโดยขึ้นอยู่กับการระบุประเภทของ elements นั้นๆ เช่น WineColor ระบุโดย

```
<owl:Class rdf:ID="WineColor">
 <rdfs:subClassOf rdf:resource="#WineDescriptor"/>
 <owl:oneOf rdf:parseType="Collection">
  <WineColor rdf:about="#White" />
  <WineColor rdf:about="#Rose" />
  <WineColor rdf:about="#Red" />
 </owl:oneOf>
</owl:Class>
```

รูปที่ 3.47 การอ้างอิงถึง elements ของเซตโดยขึ้นอยู่กับการระบุประเภทของ elements นั้นๆ

ในกรณีอื่นที่มีลักษณะการบรรยายข้อมูลที่ซับซ้อนมากกว่า ก็มีผลต่อ elements ของประโยค oneOf ด้วย ดังตัวอย่างต่อไปนี้

```
<WineColor rdf:about="#White">
  <rdfs:label>White</rdfs:label>
</WineColor>
```

รูปที่ 3.48 ลักษณะการบรรยายข้อมูลที่ซับซ้อนมากกว่ามีผลต่อ elements ของประโยค oneOf

3.3.6.3 Disjoint Classes

disjointWith [OWL DL]

การแยกส่วนของเซตของคลาสสามารถแสดงโดยการใช้ constructor owl:disjointWith ดังตัวอย่างต่อไปนี้

```
<owl:Class rdf:ID="Pasta">
  <rdfs:subClassOf rdf:resource="#EdibleThing"/>
  <owl:disjointWith rdf:resource="#Meat"/>
  <owl:disjointWith rdf:resource="#Fowl"/>
  <owl:disjointWith rdf:resource="#Seafood"/>
  <owl:disjointWith rdf:resource="#Dessert"/>
  <owl:disjointWith rdf:resource="#Fruit"/>
</owl:Class>
```

รูปที่ 3.49 การแยกส่วนของเซตของคลาสโดยการใช้ constructor owl:disjointWith

3.3.7 Ontology Versioning

Ontologies คล้าย software เพราะสามารถ maintain ได้ และสามารถเปลี่ยนแปลงได้ตลอดเวลา โดยในelement owl:Ontology จะสามารถเชื่อมโยงไปยัง version ของ ontology ก่อนหน้าได้โดยใช้ property owl:priorVersion ในการเชื่อมโยงเพื่อใช้งาน version เก่าของ ontology นั้นได้ ซึ่งมีรูปแบบ ดังตัวอย่างต่อไปนี้

```
<owl:Ontology rdf:about="">
 ...
 <owl:priorVersion rdf:resource="http://www.w3.org/TR/2003/CR-owl-guide-20030818/wine"/>
 ...
</owl:Ontology>
```

รูปที่3.50 Ontology Versioning

3.3.8 Usage Examples

ส่วนนี้เป็นตัวอย่างการใช้ซึ่งจะอธิบายบางตัวอย่างในการใช้ใน domain ของ wine ได้แก่

3.3.8.1 Wine Portal

ซึ่งศึกษาได้จาก Wine-Portal.com

3.3.8.2 Wine Agent

ซึ่งศึกษาได้จาก wine agent

3.4 SPARQL

3.4.1 เขียนการสืบค้นแบบง่ายๆ

ส่วนใหญ่แล้วรูปแบบการ Query ของ SPARQL จะเป็นรูปแบบสามระดับ หรือจะเรียกทั่วๆ ไปว่า รูปแบบกราฟพื้นฐาน โดยที่รูปแบบสามระดับจะคล้ายกับ RDF สามส่วน อันประกอบไปด้วย subject, predicate and object ซึ่งสามารถแทนเป็นตัวแปรได้

ในตัวอย่างนี้จะแสดงให้เห็นว่า SPARQL query หาหัวข้อ (title) ของหนังสือจากกราฟข้อมูล ในการ query จะแบ่งเป็นสองส่วน คือ SELECT จะประกอบไปด้วยตัวแปรเพื่อใช้ในการผลลัพธ์ และส่วน WHERE ซึ่งจะคล้ายกับรูปแบบกราฟพื้นฐาน ในตัวอย่างนี้จะมีตัวแปรเดียวคือ (?title) ซึ่งวางอยู่ในตำแหน่ง object

```
<http://example.org/book/book1> <http://purl.org/dc/elements/1.1/title> "SPARQL Tutorial" .
```

รูปที่ 3.51 ข้อมูลที่ใช้ในการ query

```
SELECT ?title
WHERE
{
<http://example.org/book/book1> <http://purl.org/dc/elements/1.1/title> ?title .
}
```

รูปที่ 3.52 การ query

การสืบค้นข้างบนจะได้ผลลัพธ์ออกมา 1 ผลลัพธ์

Title
"SPARQL Tutorial"

รูปที่ 3.53 ผลลัพธ์การ query

3.4.2 Multiple Matches การสืบค้นที่มีข้อมูลมากกว่าหนึ่ง

ผลลัพธ์ของการสืบค้นคือคำตอบของลำดับซึ่งตอบสนองความต้องการ ในทางรูปแบบของกราฟข้อมูล จะสืบค้นการสืบค้นที่มีข้อมูลตั้งแต่ 0, 1 หรือมากกว่า 5

```
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .

_:a  foaf:name   "Johnny Lee Outlaw" .
_:a  foaf:mbox   <mailto:jlow@example.com> .
_:b  foaf:name   "Peter Goodguy" .
_:b  foaf:mbox   <mailto:peter@example.org> .
_:c  foaf:mbox   <mailto:carol@example.org> .
```

รูปที่ 3.54 ข้อมูลที่ใช้ในการ query

```
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox
WHERE
  { ?x foaf:name ?name .
    ?x foaf:mbox ?mbox }
```

รูปที่ 3.55 การ query

name	Mbox
"Johnny Lee Outlaw"	<mailto:jlow@example.com>
"Peter Goodguy"	<mailto:peter@example.org>

รูปที่ 3.56 ผลลัพธ์การ query

ผลลัพธ์ที่ได้แต่ละอัน เลือกจากตัวแปรที่อยู่ในขอบเขตของรูปแบบการสืบค้น RDF ข้อมูลนั้น ความเป็นไปได้ของผลลัพธ์ที่จะให้ได้อยู่บนตัวอย่างข้างบน และสองซับเซ็ตต่อไปนี้จะเป็นข้อมูลที่อยู่ภายใต้เงื่อนไขการสืบค้น

```
_:a foaf:name  "Johnny Lee Outlaw" .
_:a foaf:box   <mailto:jlow@example.com> .
_:b foaf:name  "Peter Goodguy" .
_:b foaf:box   <mailto:peter@example.org> .
```

รูปที่ 3.57 รูปแบบการสืบค้นกราฟพื้นฐาน

นี้เป็นรูปแบบการสืบค้นกราฟพื้นฐาน ซึ่งตัวแปรที่ใช้ในการสืบค้นจะต้องมีรูปแบบเข้าข่ายกับทุกๆ เงื่อนไข

3.4.3 การ macth ข้อมูล RDF ที่เป็น Literals

เมื่อข้อมูลเป็นดังเนื้อหา RDF ที่เป็น Literals ดังต่อไปนี้

```
@prefix dt:  <http://example.org/datatype#> .
@prefix ns:  <http://example.org/ns#> .
@prefix :    <http://example.org/ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

:x  ns:p  "cat"@en .
:y  ns:p  "42"^^xsd:integer .
:z  ns:p  "abc"^^dt:specialDatatype .
```

รูปที่ 3.58 ข้อมูลเป็น RDF ที่เป็น Literals

โดยที่ "cat"@en 8nv RDF literal ที่ประกอบไปด้วย “cat” แมว และภาษาอังกฤษ, "42"^^xsd:integer เป็นชนิดข้อมูลของ literal ประเภท http://www.w3.org/2001/XMLSchema#integer และ "abc"^^dt:specialDatatype เป็นชนิดข้อมูลของ literal ประเภท http://example.org/datatype#specialDatatype. โดยที่ข้อมูล RDF คือกราฟที่จะเป็นตัวอย่างในหัวข้อ 3.3.4-3.4.6

3.4.4 การ match ข้อมูล literal ด้วย Language Tags

Language Tags ใน SPARQL โดยใช้ @ ในการสืบค้นต่อไปนี้จะไม่ออก เพราะ "cat" ไม่ใช่ "cat"@en

```
SELECT ?v WHERE { ?v ?p "cat" }
```

รูปที่ 3.59 การ query

v

รูปที่ 3.60 ผลลัพธ์การ query

แต่ถ้าสืบค้นตามตัวอย่างข้างล่างจะพบผลลัพธ์อยู่ในตัวแปร v คือจะอยู่ในขอบเขต :x เพราะว่าใช้ Language Tagsพิเศษ และเข้ากับข้อมูล

```
SELECT ?v WHERE { ?v ?p "cat"@en }
```

รูปที่ 3.61 การ query

v
<http://example.org/ns#x>

รูปที่ 3.62 ผลลัพธ์การ query

3.4.5 การ Match Literals ที่เป็นตัวเลข

ตัวเลขใน SPARQL จะเป็นการแสดงถึงชนิด RDF literal โดยมีชนิดข้อมูลเป็น xsd:integer สำหรับตัวอย่างนี้ สามารถเขียนสั้นๆ ได้เป็น 42 แทน "42"^^<http://www.w3.org/2001/XMLSchema#integer>.

รูปแบบการสืบค้นต่อไปนี้จะทำให้ตัวแปร v เก็บค่า :y

```
SELECT ?v WHERE { ?v ?p 42 }
```

รูปที่ 3.63 การ query

v
<http://example.org/ns#y>

รูปที่ 3.64 ผลลัพธ์การ query

3.4.6 การ Match Literals ที่เป็นชนิดที่กำหนดเอง

ตามการสืบค้นต่อไปนี้ผลลัพธ์ที่ได้ตัวแปล v จะเก็บ :z ในการประมวรการสืบค้นจะเข้าใจชนิดข้อมูลเพราะชนิดมาจาก IRI ซึ่ง match เข้ากับ literal

```
SELECT ?v WHERE { ?v ?p "abc"^^<http://example.org/datatype#specialDatatype> }
```

รูปที่ 3.65 การ query

v
<http://example.org/ns#z>

รูปที่ 3.66 ผลลัพธ์การ query

3.4.7ป้ายโหนดว่างในผลลัพธ์การสืบค้น

ผลลัพธ์ของการสืบค้นสามารถเป็นโหนดว่างได้ โหนดว่างในตัวอย่างต่อไปนี้กำหนดให้เป็น "_:"

โหนดว่างจะมีขอบเขตของผลลัพธ์ หรือสำหรับรูปแบบการสืบค้นแบบ CONSTRUCT ผลลัพธ์ที่ได้จะเป็นกราฟ ในตัวอย่างเราจะใช้โหนดว่างเป็นป้าย

```
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .
_:a  foaf:name   "Alice" .
_:b  foaf:name   "Bob" .
```

รูปที่ 3.67 ข้อมูลที่ใช้ในการ query

ผลลัพธ์ข้างบนนี้จะให้ค่าของป้ายโหนดว่างที่แตกต่างกัน เพราะโหนดว่างที่แสดงใน RDF เทอมในค่าแตกต่างกัน

```
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>
SELECT ?x ?name
WHERE  { ?x foaf:name ?name }
```

รูปที่ 3.68 การ query

x	name
_:c	"Alice"
_:d	"Bob"

รูปที่ 3.69 ผลลัพธ์การ query

โดยจะเห็นผลลัพธ์ที่ได้จะไม่เหมือนกับที่เก็บไว้ และสืบค้นอีกครั้งผลลัพธ์ก็จะไม่เหมือนเดิม

x	name
_:r	"Alice"
_:s	"Bob"

รูปที่ 3.70 ผลลัพธ์การ query เมื่อ query อีกครั้งหนึ่ง

ในการเขียน application ไม่ควรเขียนด้วยโหนดว่าง

3.4.8 การสร้างกราฟ RDF

ใน SELECT จะเป็นรูปแบบของ การคืนค่าตัวแปลที่เราสร้างขึ้น ส่วนใน CONSTRUCT จะคืนค่าในรูปของกราฟ RDF โดยที่กราฟที่สร้างจะใช้รูปแบบพื้นฐานของ กราฟ RDF ที่เรา query

```
@prefix org:    <http://example.com/ns#> .

_:a  org:employeeName   "Alice" .
_:a  org:employeeId     12345 .

_:b  org:employeeName   "Bob" .
_:b  org:employeeId     67890 .
```

รูปที่ 3.71 ข้อมูลที่ใช้ในการ query

```
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
PREFIX org:    <http://example.com/ns#>

CONSTRUCT { ?x foaf:name ?name }
WHERE  { ?x org:employeeName ?name }
```

รูปที่ 3.72 การ query

```
@prefix org: <http://example.com/ns#> .


_:x foaf:name "Alice" .
_:y foaf:name "Bob" .
```

รูปที่ 3.73 ผลลัพธ์การ query

เราสามารถเข้าให้อยู่ในรูป RDF/XML ได้ดังนี้

```
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:foaf="http://xmlns.com/foaf/0.1/"
   >
  <rdf:Description>
   <foaf:name>Alice</foaf:name>
  </rdf:Description>
  <rdf:Description>
   <foaf:name>Bob</foaf:name>
  </rdf:Description>
</rdf:RDF>
```

รูปที่ 3.74 เอกสาร RDF ในรูป RDF/XML

3.4.9 RDF Term Constraints (Informative)

การประมวรรูปแบบการmatch กราฟ RDF จะให้ผลลัพธ์ที่เป็นลำดับ โดยเก็บเป็น set อยู่ในตัวแปรของ RDF เทอม SPARQL FILTERs จะกรองข้อเฉพาะที่ถูกต้องหรือที่เราต้องการ

ในหัวข้อนี้จะแสดงให้เห็นถึงพื้นฐานในการใช้ FILTERs โดยจะให้กราฟข้อมูลร่วมกันในอีกสามหัวข้อต่อไป

```
@prefix dc:  <http://purl.org/dc/elements/1.1/> .
@prefix :    <http://example.org/book/> .
@prefix ns:  <http://example.org/ns#> .
:book1  dc:title  "SPARQL Tutorial" .
:book1  ns:price  42 .
:book2  dc:title  "The Semantic Web" .
:book2  ns:price  23 .
```

รูปที่ 3.75 ข้อมูลที่ใช้ในการ query

3.4.10 Restricting the Values of Strings

ฟังก์ชัน SPARQL FILTER จะใช้ ฟังก์ชัน regex ในการทดสอบ RDF literals ว่าเข้ากับรูปแบบ string ที่ต้องการหรือไม่

```
PREFIX  dc:  <http://purl.org/dc/elements/1.1/>
SELECT  ?title
WHERE   {       ?x dc:title ?title
                FILTER regex(?title, "^SPARQL") }
```

รูปที่ 3.76 การ query

title
"SPARQL Tutorial"

รูปที่ 3.77 ผลลัพธ์การ query

Regular expression ถ้าต้องการไม่ให้สนใจตัวเล็กตัวใหญ่ ให้ใช้ flag "i"

```
PREFIX  dc:  <http://purl.org/dc/elements/1.1/>
SELECT  ?title
WHERE   { ?x dc:title ?title
        FILTER regex(?title, "^SPARQL") }
```

รูปที่ 3.78 การ query

```
PREFIX  dc:  <http://purl.org/dc/elements/1.1/>
SELECT  ?title
WHERE   { ?x dc:title ?title
          FILTER regex(?title, "web", "i" ) }
```

รูปที่ 3.79 การ query

title
"The Semantic Web"

รูปที่ 3.80 ผลลัพธ์การ query

regular expression language จะประกาศเอาไว้ใน http://www.w3.org/TR/xpath-functions/#regex-syntax

3.4.11การกรอง ข้อมูลที่เป็นทศนิยม

SPARQL FILTERs สามารถกรองข้อมูลที่เป็นเลขคณิตได้

```
PREFIX  dc:  <http://purl.org/dc/elements/1.1/>
PREFIX  ns:  <http://example.org/ns#>
SELECT  ?title ?price
WHERE   { ?x ns:price ?price .
          FILTER (?price < 30.5)
          ?x dc:title ?title . }
```

รูปที่ 3.81 การ query

Title	price
"The Semantic Web"	23

รูปที่ 3.82 ผลลัพธ์การ query

โดยที่ตัวแปลราคาจะออกแต่ของ :book2 เท่านั้นเพราะมีราคามากกว่า 30.5 ตามที่ตัวกรองต้องการ

3.4.12รูปแบบของ constraints อื่นๆ

SPARQL สนับสนุน xsd:string, xsd:boolean และ xsd:dateTime

3.4.13 Predicate-Object Lists

ในรูปแบบสามส่วน subject สามารถใช้ได้มากกว่าหนึ่งครั้งใน รูปแบบ สามส่วนครั้งต่อไปโดย ใช้ ";"

```
?x  foaf:name  ?name ;
    foaf:mbox  ?mbox .
```

รูปที่ 3.83 Predicate-Object Lists

ซึ่งรูปแบบนี้จะเหมือนกันรูปแบบต่อไปนี้

```
?x  foaf:name  ?name .
?x  foaf:mbox  ?mbox .
```

รูปที่ 3.84 อีกรูปแบบหนึ่งของการเขียนเมื่อเทียบกับรูปที่ 3.83

3.4.14 Object Lists

ถ้าจะใช้ subject และ predicate ร่วมกัน สามารถแยก objects ด้วย "," ได้

```
?x foaf:nick "Alice" , "Alice_" .
```

รูปที่ 3.85 Object Lists

ซึ่งรูปแบบนี้จะเหมือนกันรูปแบบต่อไปนี้

```
?x  foaf:nick  "Alice" .
?x  foaf:nick  "Alice_" .
```

รูปที่ 3.86 อีกรูปแบบหนึ่งของการเขียนเมื่อเทียบกับรูปที่ 3.85

ถ้าเกิดว่ามี predicate-object ต่างกัน สามารถเขียนแบบนี้ได้

```
?x  foaf:name ?name ; foaf:nick  "Alice" , "Alice_" .
```

รูปที่ 3.87 Object Lists เมื่อมี predicate-object ต่างกัน

มีค่าเท่ากับ

```
?x  foaf:name  ?name .
?x  foaf:nick  "Alice" .
?x  foaf:nick  "Alice_" .
```

รูปที่ 3.88 อีกรูปแบบหนึ่งของการเขียนเมื่อเทียบกับรูปที่ 3.87

3.4.15 การสืบค้นข้อมูลที่เป็น RDF Collections

RDF collections จะสามารถเขียนในรูปแบบของสามส่วนโดยการใช้รูปแบบ "(element1 element2 ...)" รูปแบบ "()" คือทางเลือกตัวหนึ่งของ IRI http://www.w3.org/1999/02/22-rdf-syntax-ns#nil. ตัวอย่างใช้Collections สามารถelements (1 ?x 3 4)จะมีการแบ่งให้ใช้โหนดว่างใน collection โหนดว่างจะเป็นsubject ของ Collections ต่อไป โหนดว่างจะไม่เหมือนเดิมในการสืบค้นแต่ละครั้ง

```
(1 ?x 3 4) :p "w" .
```

รูปที่ 3.89 RDF collections

```
        _:b0  rdf:first  1 ;
              rdf:rest   _:b1 .
_:b1  rdf:first  ?x ;
              rdf:rest   _:b2 .
_:b2  rdf:first  3 ;
              rdf:rest   _:b3 .
_:b3  rdf:first  4 ;
              rdf:rest   rdf:nil .
_:b0  :p        "w" .
```

รูปที่ 3.90 อีกรูปแบบหนึ่งของการเขียนเมื่อเทียบกับรูปที่ 3.89

หรือRDF collectionsสามารถใช้รูปแบบอื่นร่วมได้เช่น

```
(1 [:p :q] ( 2 ) ) .
```

รูปที่ 3.91 RDF collections ที่ใช้รูปแบบอื่นร่วม

```
        _:b0  rdf:first  1 ;
              rdf:rest   _:b1 .
_:b1  rdf:first  _:b2 .
_:b2  :p         :q .
_:b1  rdf:rest   _:b3 .
_:b3  rdf:first  _:b4 .
_:b4  rdf:first  2 ;
              rdf:rest   rdf:nil .
_:b3  rdf:rest   rdf:nil .
```

รูปที่ 3.92 อีกรูปแบบหนึ่งของการเขียนเมื่อเทียบกับรูปที่ 3.91

Grammar rules:

[40]	*Collection*	::=	'(' GraphNode+ ')'
[92]	*NIL*	::=	'(' WS* ')'

รูปที่ 3.93 Grammar rules

3.4.16 rdf:type

keyword. "a" จะแทน predicate http://www.w3.org/1999/02/22-rdf-syntax-ns#type โดยที่ keywordต้องเป็นตัวเล็กเท่านั้น

```
?x  a  :Class1 .
  [ a :appClass ] :p "v" .
```

รูปที่ 3.94 keyword "a"

```
?x   rdf:type  :Class1 .
 _:b0  rdf:type  :appClass .
 _:b0  :p       "v" .
```

รูปที่ 3.95 syntactic sugar

3.4.17 Group Graph Patterns

ในการสืบค้นของ SPARQL กลุ่มของรูปแบบกราฟสามารถกำนดขอบเขตได้ด้วย {} สำหรับตัวอย่างต่อไปนี้จะเป็นการสืบค้นที่มีกลุ่มของกราฟเพียงกลุ่มด้วย

```
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox
WHERE  {    ?x foaf:name ?name .
            ?x foaf:mbox ?mbox . }
```

รูปที่ 3.96 Group Graph Patterns

การสืบค้นดังตัวอย่างข้างบนจะเป็นการสืบค้นที่ใช้กันอยู่ แต่สำหรับตัวอย่างข้างล่างนี้จะเป็นรูปแบบของการใช้กลุ่มของกราฟสองกลุ่ม ตัวอย่างข้างล่างนี้จะมีโครงสร้างของรูปแบบการสืบค้นที่แตกต่างกัน แต่ยังได้ผลลัพธ์ที่เหมือนกับการสืบค้นข้างบน

```
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox
       WHERE  { { ?x foaf:name ?name . }
    { ?x foaf:mbox ?mbox . } }
```

รูปที่ 3.97 การใช้กลุ่มของกราฟสองกลุ่ม

3.4.18 Empty Group Pattern

รูปแบบกราฟจะเป็นแบบนี้:

```
{ }
```

รูปที่ 3.98 รูปแบบ Empty Group Pattern

ตัวอย่างการ match ต่อไปนี้จะไม่สร้างอะไรในตัวแปลเลย

```
SELECT ?x
WHERE {}
```

รูปที่ 3.99 ตัวอย่างการ match

ในการสืบค้นนี้ตัวแปล x จะไม่ได้ถูกสร้างขึ้นมาเลย

3.4.19 Scope of Filters

keyword FILTER จะมีข้อจำกัด ว่าจะใช้ต่อท้ายดรงกราฟส่วนไหนดี ซึ่งรูปแบบข้างล่างการสืบค้นทั้งหมดให้ผลลัพธ์เหมือนกัน

```
{  ?x foaf:name ?name .
   ?x foaf:mbox ?mbox .
   FILTER regex(?name, "Smith") }
```

รูปที่ 3.100 keyword FILTER แบบที่ 1

```
{  FILTER regex(?name, "Smith")
   ?x foaf:name ?name .
   ?x foaf:mbox ?mbox .}
```

รูปที่ 3.101 keyword FILTER แบบที่ 2

```
{  ?x foaf:name ?name .
   FILTER regex(?name, "Smith")
   ?x foaf:mbox ?mbox .}
```

รูปที่ 3.102 keyword FILTER แบบที่ 3

3.4.20 Group Graph Pattern Examples

```
{  ?x foaf:name ?name .
   ?x foaf:mbox ?mbox . }
```

รูปที่ 3.103 กราฟของพื้นฐานรูปแบบการสืบค้น

นี้คือกราฟของพื้นฐานรูปแบบการสืบค้น ส่วนพื้นฐานรูปแบบการสืบค้นแบบนี้เงื่อนไขเป็นดังนี้

```
{  ?x foaf:name ?name . FILTER regex(?name, "Smith")
   ?x foaf:mbox ?mbox .}
```

รูปที่ 3.104 พื้นฐานรูปแบบการสืบค้นและการกรองกลุ่มเดียว

นี้คือกลุ่มของพื้นฐานรูปแบบการสืบค้นและการกรองเพียงกลุ่มเดียว ต่อไปนี้เป็นพื้นฐานการสืบค้นสองกลุ่ม

```
{   ?x foaf:name ?name .
   {}
    ?x   foaf:mbox   ?mbox
.}
```

รูปที่ 3.105 พื้นฐานการสืบค้นสองกลุ่ม

ตัวอย่างนี้จะมีสามกลุ่ม โดยที่กลุ่มแรกเป็นรูปแบบการสืบค้นปกติ กลุ่มที่สองเป็นกลุ่มว่าง กลุ่มที่สามก็เป็นรูปแบบการสืบค้นปกติเช่นกัน

3.4.21 Optional Pattern Matching

คือจะมีก็ได้ไม่มีก็ได้ ให้แสดงหมด

```
@prefix foaf:     <http://xmlns.com/foaf/0.1/> .
@prefix rdf:      <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

_:a  rdf:type      foaf:Person .
_:a  foaf:name     "Alice" .
_:a  foaf:mbox     <mailto:alice@example.com> .
_:a  foaf:mbox     <mailto:alice@work.example> .

_:b  rdf:type      foaf:Person .
_:b  foaf:name     "Bob" .
```

รูปที่ 3.106 ข้อมูลที่ใช้ในการ query

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox
WHERE  { ?x foaf:name  ?name .
         OPTIONAL { ?x  foaf:mbox  ?mbox } }
```

รูปที่ 3.107 การ query

ด้วยข้อมูลดังกล่าว สามารถสืบค้นได้ผลลัพธ์ดังต่อไปนี้:

name	mbox
"Alice"	<mailto:alice@example.com>
"Alice"	<mailto:alice@work.example>
"Bob"	

รูปที่ 3.108 ผลลัพธ์การ query

โดยจะเห็นว่าจะไม่มีค่าใน mbox ในคำตอบของ name ที่ชื่อ Bob

ในการสืบค้นจะหาชื่อของคนจากข้อมูล ถ้าโครงสร้างแบบสามส่วนมี mbox เป็น predicate และบาง subject มีเนื้อหาของ object ในตัวอย่างนี้โครงสร้างแบบสามส่วนตัวแรกจะ

ให้ข้อมูลที่ตรงกับเงื่อนไข แต่ในทั่วไปถ้าเงื่อนไขอยู่ในส่วนของ OPTIONAL เงื่อนไขนั้นจะมีหรือไม่มีก็ได้ ก็จะให้ผลลัพธ์ออกมาเหมือนกัน

3.4.22 Constraints in Optional Pattern Matching

```
@prefix dc:   <http://purl.org/dc/elements/1.1/> .
@prefix :     <http://example.org/book/> .
@prefix ns:   <http://example.org/ns#> .

:book1  dc:title  "SPARQL Tutorial" .
:book1  ns:price  42 .
:book2  dc:title  "The Semantic Web" .
:book2  ns:price  23 .
```

รูปที่ 3.109 ข้อมูลที่ใช้ในการ query

```
PREFIX  dc:  <http://purl.org/dc/elements/1.1/>
PREFIX  ns:  <http://example.org/ns#>
SELECT  ?title ?price
WHERE   { ?x dc:title ?title .
          OPTIONAL { ?x ns:price ?price . FILTER (?price < 30) }}
```

รูปที่ 3.110 การ query

Title	price
"SPARQL Tutorial"	
"The Semantic Web"	23

รูปที่ 3.111 ผลลัพธ์การ query

จะไม่แสดงราคาของหนังสือที่มีหัวข้อว่า "SPARQL Tutorial" เพราะว่ารูปแบบกราฟ optional ไม่มีผลลัพธ์ที่นำไปสู่การเรียกตัวแปร ราคา "price"

3.4.23 Multiple Optional Graph Patterns

รูปแบบของกราฟคือการเรียกซ้ำ(recursively) อย่างชัดเจน โดยที่รูปแบบของกราฟอาจจะไม่มี Optional เลยหรือจะมีมากหลายตัว หรือการสืบค้นทั้งหมดจะเป็น Optional ก็ได้ ในตัวอย่างนี้จะมีส่วน Optional สองส่วน

```
@prefix foaf:      <http://xmlns.com/foaf/0.1/> .
_:a  foaf:name     "Alice" .
_:a  foaf:homepage <http://work.example.org/alice/> .
_:b  foaf:name     "Bob" .
_:b  foaf:mbox     <mailto:bob@work.example> .
```

รูปที่ 3.112 ข้อมูลที่ใช้ในการ query

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox ?hpage
WHERE  { ?x foaf:name  ?name .
         OPTIONAL { ?x foaf:mbox ?mbox } .
         OPTIONAL { ?x foaf:homepage ?hpage }}
```

รูปที่ 3.113 การ query

Name	mbox	hpage
"Alice"		<http://work.example.org/alice/>
"Bob"	<mailto:bob@work.example>	

รูปที่ 3.114 ผลลัพธ์การ query

3.4.24 Matching Alternatives

SPARQL ได้จัดเตรียมวิธีที่จำรวมหนึ่งในตัวเลือกหลายๆ ที่อยู่ในกราฟมาให้พร้อมแล้ว ดังนั้นจึงสามารถหาหนึ่งในตัวเลือกหลายๆ หรือพบทั้งหมด

รูปแบบของตัวเลือกนั้นจะให้ คำว่า UNION ในการรวม

```
@prefix dc10:  <http://purl.org/dc/elements/1.0/> .
@prefix dc11:  <http://purl.org/dc/elements/1.1/> .
_:a  dc10:title    "SPARQL Query Language Tutorial" .
_:a  dc10:creator  "Alice" .
_:b  dc11:title    "SPARQL Protocol Tutorial" .
_:b  dc11:creator  "Bob" .
_:c  dc10:title    "SPARQL" .
_:c  dc11:title    "SPARQL (updated)" .
```

รูปที่ 3.115 ข้อมูลที่ใช้ในการ query

```
PREFIX dc10:  <http://purl.org/dc/elements/1.0/>
PREFIX dc11:  <http://purl.org/dc/elements/1.1/>
SELECT ?title
WHERE  { { ?book dc10:title  ?title } UNION { ?book dc11:title  ?title } }
```

รูปที่ 3.116 การ query

Title
"SPARQL Protocol Tutorial"
"SPARQL"
"SPARQL (updated)"
"SPARQL Query Language Tutorial"

รูปที่ 3.117 ผลลัพธ์การ query

ในการสืบค้นจะพบหัวข้อของหนังสือในข้อมูล โดยที่หัวข้อมีปรากฏใน Dublin Core **แต่เกิดเรา สืบค้นด้วยชื่อตัวแปรที่แตกต่างกัน**

```
PREFIX dc10:  <http://purl.org/dc/elements/1.0/>
PREFIX dc11:  <http://purl.org/dc/elements/1.1/>
SELECT ?x ?y
WHERE  { { ?book dc10:title ?x } UNION { ?book dc11:title  ?y } }
```

รูปที่ 3.118 การ query

X	Y
	"SPARQL (updated)"
	"SPARQL Protocol Tutorial"
"SPARQL"	
"SPARQL Query Language Tutorial"	

รูปที่ 3.119 ผลลัพธ์การ query

นี้เป็นผลลัพธ์ที่ได้การสืบค้น โดยที่ตัวแปร x ซึ่งจะอยู่ในขอบเขตทางซ้ายของ การรวมด้วย UNION จะเห็นว่ามีกราฟที่ไม่เข้าเงื่อนไขด้วย

ในการรวมกราฟด้วย UNION เราสามารถรวมกราฟได้มากกว่าหนึ่งรูปแบบสามส่วน

```
PREFIX dc10:  <http://purl.org/dc/elements/1.0/>
PREFIX dc11:  <http://purl.org/dc/elements/1.1/>

SELECT ?title ?author
WHERE  { { ?book dc10:title ?title .  ?book dc10:creator ?author }
       UNION
       { ?book dc11:title ?title .  ?book dc11:creator ?author } }
```

รูปที่ 3.120 การ query

author	title
"Alice"	"SPARQL Protocol Tutorial"
"Bob"	"SPARQL Query Language Tutorial"

รูปที่ 3.121 ผลลัพธ์การ query

ในการสืบค้นนี้ใช้หัวข้อของหนังสือเท่านั้น แต่ถ้ามีหัวข้อกับผู้สร้างสามารถดูรูปแบบได้จากเวอร์ชันของ Dublin Core

[25]	*GroupOrUnionGraphPattern*	::=	GroupGraphPattern ('UNION' GroupGraphPattern)*

รูปที่ 3.122 Grammar rule

3.4.25. Examples of RDF Datasets

RDF Datasets เป็นการจับกลุ่มของกราฟที่เหมือนกันจากหลายๆ แหล่งมารวมกันจะมีส่วนประกอบสองส่วน

- มีข้อมูลที่เป็นกราฟเริ่มต้น ที่เก็บข้อมูลเกี่ยวกับชื่อกราฟที่เกี่ยวข้องกับ datasets นั้นๆ ไว้ทั้งหมด
- ข้อมูลของชื่อกราฟที่อยู่ในกราฟเริ่มต้น

```
# Default graph
@prefix dc: <http://purl.org/dc/elements/1.1/> .
<http://example.org/bob>    dc:publisher  "Bob" .
<http://example.org/alice>  dc:publisher  "Alice" .
```

รูปที่ 3.123 ตัวอย่าง RDF Datasets

```
# Named graph: http://example.org/bob
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
_:a foaf:name "Bob" .
_:a foaf:mbox <mailto:bob@oldcorp.example.org>
.
```

รูปที่ 3.124 ตัวอย่าง RDF Datasets

```
# Named graph: http://example.org/alice
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
_:a foaf:name "Alice" .
_:a foaf:mbox <mailto:alice@work.example.org> .
```

รูปที่ 3.125 ตัวอย่าง RDF Datasets

3.4.26 Specifying the Default Graph

FROM เป็นสิ่งที่มีเนื้อหาเป็น IRI ซึ่งจะระบุที่อยู่ของกราฟเริ่มต้น นี้แต่จะไม่ได้ระบุถึงกราฟ named อื่นๆ

```
# Default graph (stored at http://example.org/foaf/aliceFoaf)
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
_:a foaf:name "Alice" .
_:a foaf:mbox <mailto:alice@work.example> .
```

รูปที่ 3.126 ข้อมูลที่ใช้ในการ query

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name
FROM <http://example.org/foaf/aliceFoaf>
WHERE { ?x foaf:name ?name }
```

รูปที่ 3.127 การ query

name
"Alice"

รูปที่ 3.128 ผลลัพธ์การ query

3.4.27 Specifying Named Graphs

การสืบค้นสามารถสนับสนุน IRIs ของกราฟ named ใน RDF Dataset ได้โดยการใช้ FROM NAMED แล้วตามด้วย IRI นั้นๆ

```
# Graph: http://example.org/bob
@prefix                                    foaf:
<http://xmlns.com/foaf/0.1/> .
_:a foaf:name "Bob" .
_:a                                        foaf:mbox
<mailto:bob@oldcorp.example.org> .
```

รูปที่ 3.129 Specifying Named Graphs

```
# Graph: http://example.org/alice
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
_:a foaf:name "Alice" .
_:a foaf:mbox <mailto:alice@work.example> .
```

รูปที่ 3.130 Specifying Named Graphs

```
...
FROM NAMED <http://example.org/alice>
FROM NAMED <http://example.org/bob>
...
```

รูปที่ 3.131 FROM NAMED

FROM NAMED เป็นไวยากรณ์ที่สนับสนุน IRI เพื่อที่จะประสานกราฟ แต่ระหว่าง IRI และกราฟใน RDF dataset เป็นแบบมีทิศทาง

3.4.28 Combining FROM and FROM NAMED

จาก FROM และ FROM NAMED สามารถใช้สืบค้นได้ดังนี้

```
# Default graph (stored at http://example.org/dft.ttl)
@prefix dc: <http://purl.org/dc/elements/1.1/> .

<http://example.org/bob>    dc:publisher  "Bob Hacker" .
<http://example.org/alice>  dc:publisher  "Alice Hacker" .
```

รูปที่ 3.132 Combining FROM and FROM NAMED

```
# Named graph: http://example.org/bob
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

_:a foaf:name "Bob" .
_:a foaf:mbox <mailto:bob@oldcorp.example.org> .
```

รูปที่ 3.133 Combining FROM and FROM NAMED

```
# Named graph: http://example.org/alice
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
_:a foaf:name "Alice" .
_:a foaf:mbox <mailto:alice@work.example.org> .
```

รูปที่ 3.134 Combining FROM and FROM NAMED

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT ?who ?g ?mbox
FROM <http://example.org/dft.ttl>
FROM NAMED <http://example.org/alice>
FROM NAMED <http://example.org/bob>
WHERE{      ?g dc:publisher ?who .
            GRAPH ?g { ?x foaf:mbox ?mbox  }
```

รูปที่ 3.135 Combining FROM and FROM NAMED

การ query นี้จะใช้ กราฟเริ่มต้น และกราฟ named ทั้งสองกราฟ โดยใช้ คำ GRAPH

3.4.29 Accessing Graph Names

เป็นการ query ในแต่ละ กราฟ named ในdataset และมีตัวแปล src ในการเก็น IRIs ที่อยู่ใน FROM NAMED มาที่ละค่า

```
# Named graph: http://example.org/foaf/aliceFoaf
@prefix  foaf:    <http://xmlns.com/foaf/0.1/> .
@prefix  rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix  rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
_:a  foaf:name    "Alice" .
_:a  foaf:mbox    <mailto:alice@work.example> .
_:a  foaf:knows   _:b .

_:b  foaf:name    "Bob" .
_:b  foaf:mbox    <mailto:bob@work.example> .
_:b  foaf:nick    "Bobby" .
_:b  rdfs:seeAlso <http://example.org/foaf/bobFoaf> .

<http://example.org/foaf/bobFoaf>
    rdf:type      foaf:PersonalProfileDocument .
```

รูปที่ 3.136 ข้อมูลที่ใช้ในการ query

```
# Named graph: http://example.org/foaf/bobFoaf
@prefix  foaf:    <http://xmlns.com/foaf/0.1/> .
@prefix  rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix  rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
:z  foaf:mbox    <mailto:bob@work.example> .
_:z  rdfs:seeAlso  <http://example.org/foaf/bobFoaf> .
_:z  foaf:nick    "Robert" .
<http://example.org/foaf/bobFoaf>
        rdf:type     foaf:PersonalProfileDocument .
```

รูปที่ 3.137 ข้อมูลที่ใช้ในการ query

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?src ?bobNick
FROM NAMED <http://example.org/foaf/aliceFoaf>
FROM NAMED <http://example.org/foaf/bobFoaf>
WHERE  {
          GRAPH ?src
          { ?x foaf:mbox <mailto:bob@work.example> .
           ?x foaf:nick ?bobNick  }}
```

รูปที่ 3.138 การ query

ผลลัพธ์ที่ได้ชื่อของกราฟ named และชื่อเล่น

src	bobNick
<http://example.org/foaf/aliceFoaf>	"Bobby"
<http://example.org/foaf/bobFoaf>	"Robert"

รูปที่ 3.139 ผลลัพธ์การ query

3.4.30 Restricting by Graph IRI

นอกจากนี้แล้วสามารถกรองให้กำจัดลงได้โดยให้แสดงแต่ชื่อเล่นของกราฟ http://example.org/foaf/bobFoaf. เท่านั้น

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX data: <http://example.org/foaf/>

SELECT ?nick
FROM NAMED <http://example.org/foaf/aliceFoaf>
FROM NAMED <http://example.org/foaf/bobFoaf>
WHERE{
   GRAPH data:bobFoaf {
     ?x foaf:mbox <mailto:bob@work.example> .
     ?x foaf:nick ?nick }}
```

รูปที่ 3.140 Restricting by Graph IRI

ผลลัพธ์ที่ได้มีดังนี้:

nick
"Robert"

รูปที่ 3.141 ผลลัพธ์การ query

3.4.31 Restricting Possible Graph IRIs

นอกจากนี้แล้วยังสามารถผสมทั้งสองส่วนเข้าด้วยกันได้ คือให้จำกัดขอบเขตของกราฟผสมกับไม่จำกัดได้ดังนี้

```
PREFIX data: <http://example.org/foaf/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?mbox ?nick ?ppd
FROM NAMED <http://example.org/foaf/aliceFoaf>
FROM NAMED <http://example.org/foaf/bobFoaf>
WHERE{
 GRAPH data:aliceFoaf{
  ?alice foaf:mbox <mailto:alice@work.example> ;
       foaf:knows ?whom .
  ?whom  foaf:mbox ?mbox ;
       rdfs:seeAlso ?ppd .
  ?ppd  a foaf:PersonalProfileDocument . } .
 GRAPH ?ppd {
   ?w foaf:mbox ?mbox ;
     foaf:nick ?nick}}
```

รูปที่ 3.142 Restricting Possible Graph IRIs

Mbox	nick	ppd
<mailto:bob@work.example>	"Robert"	<http://example.org/foaf/bobFoaf>

รูปที่ 3.143 ผลลัพธ์การ query

3.4.32 Named and Default Graphs

การประกาศ Named ใน Default Graphs โดยที่ไม่ต้องใช้ FROM NAMED

```
# Default graph
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix g:  <tag:example.org,2005-06-06:> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

g:graph1 dc:publisher "Bob" .
g:graph1 dc:date "2004-12-06"^^xsd:date .

g:graph2 dc:publisher "Bob" .
g:graph2 dc:date "2005-01-10"^^xsd:date .
```

รูปที่ 3.144 ข้อมูลที่ใช้ในการ query

```
# Graph: locally allocated IRI: tag:example.org,2005-06-06:graph1
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

_:a foaf:name "Alice" .
_:a foaf:mbox <mailto:alice@work.example> .

_:b foaf:name "Bob" .
_:b foaf:mbox <mailto:bob@oldcorp.example.org> .
```

รูปที่ 3.145 ข้อมูลที่ใช้ในการ query

```
# Graph: locally allocated IRI: tag:example.org,2005-06-06:graph2
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

_:a foaf:name "Alice" .
_:a foaf:mbox <mailto:alice@work.example> .

_:b foaf:name "Bob" .
_:b foaf:mbox <mailto:bob@newcorp.example.org> .
```

รูปที่ 3.146 ข้อมูลที่ใช้ในการ query

ในการ query จะหาอีเมล์ และแสดงชื่อและข้อมูลวันที่ออกมา

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dc:   <http://purl.org/dc/elements/1.1/>

SELECT ?name ?mbox ?date
WHERE{  ?g dc:publisher ?name ;
        dc:date ?date .
   GRAPH ?g
     { ?person foaf:name ?name ; foaf:mbox ?mbox }}
```

รูปที่ 3.147 การ query

ผลลัพธ์ที่ได้จะแสดง อีเมล์ของ "Bob" และวันที่

Name	***mbox***	**date**
"Bob"	<mailto:bob@oldcorp.example.org>	"2004-12-06"^^xsd:date
"Bob"	<mailto:bob@newcorp.example.org>	"2005-01-10"^^xsd:date

รูปที่ 3.148 ผลลัพธ์การ query

3.4.33 ORDER BY

```
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>

SELECT ?name
WHERE { ?x foaf:name ?name }
ORDER BY ?name
```

รูปที่ 3.149 ORDER BY

เรียงตามตัวแปร ?name จากน้อยไปมาก

```
PREFIX    :   <http://example.org/ns#>
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
PREFIX xsd:    <http://www.w3.org/2001/XMLSchema#>

SELECT ?name
WHERE { ?x foaf:name ?name ; :empId ?emp }
ORDER BY DESC(?emp)
```

รูปที่ 3.150 ORDER BY

เรียงตามตัวแปร ?emp จากมากไปน้อย

```
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>

SELECT ?name
WHERE { ?x foaf:name ?name ; :empId ?emp }
ORDER BY ?name DESC(?emp)
```

รูปที่ 3.151 ORDER BY

เรียงตามตัวแปร ?name จากน้อยไปมากก่อน ถ้าเท่ากันจึงเรียงตามตัวแปร ?emp จากมากไปน้อย

3.4.34 Projection

การประมวลจะเป็นลำดับ คือเมื่อใช้ตัวแปรหนึ่งจาก โซลูชั่นหรือคำตอบหนึ่ง โซลูชั่นที่สองก็จะสามารถดึงค่าจากโซลูชั่นที่ 1 ไปใช้ได้

```
@prefix foaf:      <http://xmlns.com/foaf/0.1/> .

_:a  foaf:name     "Alice" .
_:a  foaf:mbox     <mailto:alice@work.example> .

_:b  foaf:name     "Bob" .
_:b  foaf:mbox     <mailto:bob@work.example> .
```

รูปที่ 3.152 ข้อมูลที่ใช้ในการ query

```
PREFIX foaf:     <http://xmlns.com/foaf/0.1/>
SELECT ?name
WHERE
{ ?x foaf:name ?name }
```

รูปที่ 3.153 การ query

name
"Bob"
"Alice"

รูปที่ 3.154 ผลลัพธ์การ query

3.4.35 Duplicate Solutions

โซลูชั่นที่เป็นลำดับ ถ้าไม่มี DISTINCT หรือ REDUCEDจะทำให้คำตอบซ้ำกันได้

```
@prefix  foaf:  <http://xmlns.com/foaf/0.1/> .

_:x   foaf:name   "Alice" .
_:x   foaf:mbox   <mailto:alice@example.com> .

_:y   foaf:name   "Alice" .
_:y   foaf:mbox   <mailto:asmith@example.com> .

_:z   foaf:name   "Alice" .
_:z   foaf:mbox   <mailto:alice.smith@example.com> .
```

รูปที่ 3.155 ข้อมูลที่ใช้ในการ query

```
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
SELECT ?name WHERE { ?x foaf:name ?name }
```

รูปที่ 3.156 การ query

Name
"Alice"
"Alice"
"Alice"

รูปที่ 3.157 ผลลัพธ์การ query

3.4.36 DISTINCT

ถ้าใช้ มอด'ดิไฟเออะ(modifier) ชื่อ DISTINCTจะทำให้ได้เพียงค่าเดียว

```
PREFIX foaf:    <http://xmlns.com/foaf/0.1/>
SELECT DISTINCT ?name WHERE { ?x foaf:name ?name }
```

รูปที่ 3.158 การ query

Name
"Alice"

รูปที่ 3.159 ผลลัพธ์การ query

3.4.37 REDUCED

ถ้าใช้ มอด'ดิไฟเออะ(modifier) ชื่อ REDUCED จะทำให้ลดค่าเดียวที่ซ้ำลง

```
PREFIX foaf:    <http://xmlns.com/foaf/0.1/>
SELECT REDUCED ?name WHERE { ?x foaf:name ?name }
```

รูปที่ 3.160 การ query

จะมีค่าของโซลูชั่นที่ 1, 2 (แสดงให้ที่นี้) หรือ 3

name
"Alice"
"Alice"

รูปที่ 3.161 ผลลัพธ์การ query

3.4.38 OFFSET

คือการจะให้แสดงผลของคำตอบเริ่มตั้งแต่ตัวที่เท่าไร ส่วนใหญ่จะใช่คู่กับ ORDER BY.

เช่น คำตอบหาที่ลำดับที่สอง

```
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>

SELECT  ?name
WHERE   { ?x foaf:name ?name }
ORDER BY ?name
LIMIT   5
OFFSET  10
```

รูปที่ 3.162 OFFSET

3.4.39 LIMIT

คือการกำหนดกำจัดจำนวนการแสดงผล

```
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>

SELECT ?name
WHERE { ?x foaf:name ?name }
LIMIT 20
```

รูปที่ 3.163 LIMIT

ถ้ากำหนดให้ LIMIT ก็จะไม่แสดงผลอะไรเลย

3.4.40 Query Forms SELECT

SELECT จะแสดงผล ตัวแปรที่สร้างขึ้น ถ้าใช้ SELECT * จะเป็นการแสดงผลตัวแปรทุกตัว

```
@prefix  foaf:  <http://xmlns.com/foaf/0.1/> .

_:a   foaf:name   "Alice" .
_:a   foaf:knows  _:b .
_:a   foaf:knows  _:c .

_:b   foaf:name   "Bob" .

_:c   foaf:name   "Clare" .
_:c   foaf:nick   "CT" .
```

รูปที่ 3.164 ข้อมูลที่ใช้ในการ query

```
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
SELECT ?nameX ?nameY ?nickY
WHERE
 { ?x foaf:knows ?y ;
    foaf:name ?nameX .
  ?y foaf:name ?nameY .
  OPTIONAL { ?y foaf:nick ?nickY }
 }
```

รูปที่ 3.165 การ query

nameX	nameY	nickY
"Alice"	"Bob"	
"Alice"	"Clare"	"CT"

รูปที่ 3.166 ผลลัพธ์การ query

แต่ผลลัพธ์ที่ได้จะส่งกลับมาในรูป XML Format ดังตัวอย่างต่อไปนี้

```
<?xml version="1.0"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#">
 <head>
  <variable name="nameX"/>
  <variable name="nameY"/>
  <variable name="nickY"/>
 </head>
 <results>
  <result>
   <binding name="nameX">
    <literal>Alice</literal>
   </binding>
   <binding name="nameY">
    <literal>Bob</literal>
   </binding>
  </result>
  <result>
   <binding name="nameX">
    <literal>Alice</literal>
   </binding>
   <binding name="nameY">
    <literal>Clare</literal>
   </binding>
   <binding name="nickY">
    <literal>CT</literal>
   </binding>
  </result>
 </results>
</sparql>
```

รูปที่ 3.167 ผลลัพธ์การ query ในรูป XML Format

Grammar rule:

[5]	*SelectQuery*	::=	'SELECT' ('DISTINCT' | 'REDUCED')? (Var+ | '*') DatasetClause* WhereClause SolutionModifier

รูปที่ 3.168 Grammar rule

3.4.41 Query Forms CONSTRUCT

เป็นรูปแบบที่ให้ผลลัพธ์ออกมาเป็นกราฟได้

```
@prefix  foaf:  <http://xmlns.com/foaf/0.1/> .

_:a    foaf:name   "Alice" .
_:a    foaf:mbox   <mailto:alice@example.org> .
```

รูปที่ 3.169 ข้อมูลที่ใช้ในการ query

```
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
PREFIX vcard:  <http://www.w3.org/2001/vcard-rdf/3.0#>
CONSTRUCT   { <http://example.org/person#Alice> vcard:FN ?name }
WHERE       { ?x foaf:name ?name }
```

รูปที่ 3.170 การ query

```
@prefix vcard: <http://www.w3.org/2001/vcard-rdf/3.0#> .
<http://example.org/person#Alice> vcard:FN "Alice" .
```

รูปที่ 3.171 ผลลัพธ์การ query

3.4.42 Accessing Graphs in the RDF Dataset

เป็นการผสมระหว่าง Graphs และ CONSTRUCT

```
PREFIX  dc: <http://purl.org/dc/elements/1.1/>
PREFIX app: <http://example.org/ns#>
CONSTRUCT { ?s ?p ?o }
WHERE{
  GRAPH ?g { ?s ?p ?o } .
  { ?g dc:publisher <http://www.w3.org/> } .
  { ?g dc:date ?date } .
  FILTER ( app:customDate(?date) > "2005-02-28T00:00:00Z"^^xsd:dateTime ) .}
```

รูปที่ 3.172 Accessing Graphs in the RDF Dataset

Grammar rule:

[6]	*ConstructQuery*	::=	'CONSTRUCT' ConstructTemplate DatasetClause* WhereClause SolutionModifier

รูปที่ 3.173 Grammar rule

3.4.43 Query Forms ASK

เป็นการถามว่ามีกราฟ ตามรูปแบบที่กำหนดหรือไหม จะตอบเพียง yes และ no

```
@prefix foaf:      <http://xmlns.com/foaf/0.1/> .

_:a  foaf:name     "Alice" .
_:a  foaf:homepage  <http://work.example.org/alice/> .

_:b  foaf:name     "Bob" .
_:b  foaf:mbox     <mailto:bob@work.example> .
```

รูปที่ 3.174 ข้อมูลที่ใช้ในการ query

```
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
ASK  { ?x foaf:name  "Alice" }
```

รูปที่ 3.175 การ query

```
yes
```

รูปที่ 3.176 ผลลัพธ์การ query

ผลลัพธ์ที่อยู่ในรูป XML Format

```
<?xml version="1.0"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#">
  <head></head>
  <results>
    <boolean>true</boolean>
  </results>
</sparql>
```

รูปที่ 3.177 ผลลัพธ์การ query ในรูป XML Format

3.4.44 DESCRIBE (Informative)

เป็นรูปแบบการคืนผลลัพธ์ที่เป็น RDF graph เพียงตัวเดียว ที่บรรจุเนื้อหาข้อมูล RDF ที่เกี่ยวข้อง

```
PREFIX ent:  <http://org.example.com/employees#>
DESCRIBE ?x WHERE { ?x ent:employeeId "1234" }
```

รูปที่ 3.178 DESCRIBE (Informative)

```
@prefix foaf:   <http://xmlns.com/foaf/0.1/> .
@prefix vcard:  <http://www.w3.org/2001/vcard-rdf/3.0> .
@prefix exOrg:  <http://org.example.com/employees#> .
@prefix rdf:    <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl:    <http://www.w3.org/2002/07/owl#>

_:a    exOrg:employeeId    "1234" ;

       foaf:mbox_sha1sum   "ABCD1234" ;
       vcard:N
       [ vcard:Family      "Smith" ;
         vcard:Given       "John" ] .

foaf:mbox_sha1sum rdf:type owl:InverseFunctionalProperty.
```

รูปที่ 3.179 DESCRIBE (Informative)

Grammar rule:

[7]	*DescribeQuery*	::=	'DESCRIBE' (VarOrIRIref+ | '*') DatasetClause* WhereClause? SolutionModifier

รูปที่ 3.180 Grammar rule

3.4.45 Operand Data Types

สามารถใช้ ชนิดข้อมูลที่ประกาศใน XML Schema ได้ไม่ว่าจะเป็น

- xsd:integer
- xsd:decimal
- xsd:float
- xsd:double
- xsd:string
- xsd:boolean
- xsd:dateTime

- xsd:nonPositiveInteger
- xsd:negativeInteger
- xsd:long
- xsd:int
- xsd:short
- xsd:byte
- xsd:nonNegativeInteger
- xsd:unsignedLong
- xsd:unsignedInt
- xsd:unsignedShort
- xsd:unsignedByte
- xsd:positiveInteger

3.4.46 Operator Mapping

ยังสามารถใช้ Operatorที่ประกาศไว้ใน XML Schemaได้ดังต่อไปนี้

ตารางที่ 3.1 SPARQL Unary Operators

Operator	Type(A)	Function	Result type
XQuery Unary Operators			
! A	xsd:boolean (EBV)	fn:not(A)	xsd:boolean
+ A	numeric	op:numeric-unary-plus(A)	numeric
- A	numeric	op:numeric-unary-minus(A)	numeric
SPARQL Tests, defined in section 11.4			
BOUND(A)	variable	bound(A)	xsd:boolean
isIRI(A) **isURI(A)**	RDF term	isIRI(A)	xsd:boolean
isBLANK(A)	RDF term	isBlank(A)	xsd:boolean
isLITERAL(A)	RDF term	isLiteral(A)	xsd:boolean
SPARQL Accessors, defined in section 11.4			
STR(A)	literal	str(A)	simple literal
STR(A)	IRI	str(A)	simple literal
LANG(A)	literal	lang(A)	simple literal
DATATYPE(A)	typed literal	datatype(A)	IRI
DATATYPE(A)	simple literal	datatype(A)	IRI

ตารางที่ 3.2 SPARQL Binary Operators

Operator	Type(A)	Type(B)	Function	Result type
Logical Connectives, defined in section 11.4				
A || B	xsd:boolean (EBV)	xsd:boolean (EBV)	logical-or(A, B)	xsd:boolean
A && B	xsd:boolean (EBV)	xsd:boolean (EBV)	logical-and(A, B)	xsd:boolean
XPath Tests				
A = B	numeric	numeric	op:numeric-equal(A, B)	xsd:boolean
A = B	simple literal	simple literal	op:numeric-equal(fn:compare(A, B), 0)	xsd:boolean
A = B	xsd:string	xsd:string	op:numeric-equal(fn:compare(STR(A), STR(B)), 0)	xsd:boolean
A = B	xsd:boolean	xsd:boolean	op:boolean-equal(A, B)	xsd:boolean
A = B	xsd:dateTime	xsd:dateTime	op:dateTime-equal(A, B)	xsd:boolean
A != B	numeric	numeric	fn:not(op:numeric-equal(A, B))	xsd:boolean
A != B	simple literal	simple literal	fn:not(op:numeric-equal(fn:compare(A, B), 0))	xsd:boolean
A != B	xsd:string	xsd:string	fn:not(op:numeric-equal(fn:compare(STR(A), STR(B)), 0))	xsd:boolean
A != B	xsd:boolean	xsd:boolean	fn:not(op:boolean-equal(A, B))	xsd:boolean
A != B	xsd:dateTime	xsd:dateTime	fn:not(op:dateTime-equal(A, B))	xsd:boolean
A < B	numeric	numeric	op:numeric-less-than(A, B)	xsd:boolean
A < B	simple literal	simple literal	op:numeric-equal(fn:compare(A, B), -1)	xsd:boolean
A < B	xsd:string	xsd:string	op:numeric-equal(fn:compare(STR(A), STR(B)), -1)	xsd:boolean
A < B	xsd:boolean	xsd:boolean	op:boolean-less-than(A, B)	xsd:boolean
A < B	xsd:dateTime	xsd:dateTime	op:dateTime-less-than(A, B)	xsd:boolean
A > B	numeric	numeric	op:numeric-greater-than(A, B)	xsd:boolean
A > B	simple literal	simple literal	op:numeric-equal(fn:compare(A, B), 1)	xsd:boolean
A > B	xsd:string	xsd:string	op:numeric-equal(fn:compare(STR(A), STR(B)), 1)	xsd:boolean
A > B	xsd:boolean	xsd:boolean	op:boolean-greater-than(A, B)	xsd:boolean
A > B	xsd:dateTime	xsd:dateTime	op:dateTime-greater-than(A, B)	xsd:boolean
A <= B	numeric	numeric	fn:not(op:numeric-less-than(A, B), op:numeric-equal(A, B))	xsd:boolean
A <= B	simple literal	simple literal	fn:not(op:numeric-equal(fn:compare(A, B), 1))	xsd:boolean
A <= B	xsd:string	xsd:string	fn:not(op:numeric-equal(fn:compare(STR(A), STR(B)), 1))	xsd:boolean
A <= B	xsd:boolean	xsd:boolean	fn:not(op:boolean-greater-than(A, B))	xsd:boolean
A <= B	xsd:dateTime	xsd:dateTime	fn:not(op:dateTime-greater-than(A, B))	xsd:boolean
A >= B	numeric	numeric	fn:not(op:numeric-less-than(A, B)), op:numeric-equal(A, B))	xsd:boolean
A >= B	simple literal	simple literal	fn:not(op:numeric-equal(fn:compare(A, B), -1))	xsd:boolean
A >= B	xsd:string	xsd:string	fn:not(op:numeric-equal(fn:compare(STR(A), STR(B)), -1))	xsd:boolean
A >= B	xsd:boolean	xsd:boolean	fn:not(op:boolean-less-than(A, B))	xsd:boolean
A >= B	xsd:dateTime	xsd:dateTime	fn:not(op:dateTime-less-than(A, B))	xsd:boolean
XPath Arithmetic				
A * B	numeric	numeric	op:numeric-multiply(A, B)	numeric
A / B	numeric	numeric	op:numeric-divide(A, B)	numeric, but xsd:decimal if both operands are xsd:integer
A + B	numeric	numeric	op:numeric-add(A, B)	numeric
A - B	numeric	numeric	op:numeric-subtract(A, B)	numeric
SPARQL Tests, defined in section 11.4				
A = B	RDF term	RDF term	RDFterm-equal(A, B)	xsd:boolean
A != B	RDF term	RDF term	fn:not(RDFterm-equal(A, B))	xsd:boolean
sameTERM(A)	RDF term	RDF term	sameTerm(A, B)	xsd:boolean
langMATCHES(A, B)	simple literal	simple literal	langMatches(A, B)	xsd:boolean
REGEX(STRING, PATTERN)	simple literal	simple literal	fn:match-er(STRING, PATTERN)	xsd:boolean

ตารางที่ 3.3 SPARQL Trinary Operators

Operator	Type(A)	Type(B)	Type(C)	Function	Result type
SPARQL Tests, defined in section 11.4					
REGEX(STRING, PATTERN, FLAGS)	simple literal	simple literal	simple literal	fn:matches(STRING, PATTERN, FLAGS)	xsd:boolean

3.4.47 bound

```
xsd:boolean  BOUND (variable var)
```

รูปที่ 3.181 bound

จะคืนค่า true ถ้าตัวแปรนั้นมีค่า และ คืนค่า false เมื่อ ตัวแปรนั้นไม่มีค่า

```
@prefix foaf:      <http://xmlns.com/foaf/0.1/> .
@prefix dc:        <http://purl.org/dc/elements/1.1/> .
@prefix xsd:       <http://www.w3.org/2001/XMLSchema#> .

_:a  foaf:givenName  "Alice".

_:b  foaf:givenName  "Bob" .
_:b  dc:date       "2005-04-04T04:04:04Z"^^xsd:dateTime .
```

รูปที่ 3.182 ข้อมูลที่ใช้ในการ query

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dc:   <http://purl.org/dc/elements/1.1/>
PREFIX xsd:   <http://www.w3.org/2001/XMLSchema#>
SELECT ?name
 WHERE { ?x foaf:givenName  ?givenName .
       OPTIONAL { ?x dc:date ?date } .
       FILTER ( bound(?date) ) }
```

รูปที่ 3.183 การ query

givenName
"Bob"

รูปที่ 3.184 ผลลัพธ์การ query

อีกตัวอย่างหนึ่ง

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dc:   <http://purl.org/dc/elements/1.1/>
SELECT ?name
 WHERE { ?x foaf:givenName  ?name .
       OPTIONAL { ?x dc:date ?date } .
       FILTER (!bound(?date)) }
```

รูปที่ 3.185 การ query

Name
"Alice"

รูปที่ 3.186 ผลลัพธ์การ query

3.4.48 isIRI

```
xsd:boolean   ISIRI (RDF term term)
xsd:boolean   ISURI (RDF term term)
```

รูปที่ 3.187 isIRI

เป็นฟังก์ชันตรวจสอบว่ามีค่าเป็น IRI หรือไหม

```
@prefix foaf:     <http://xmlns.com/foaf/0.1/> .

_:a  foaf:name      "Alice".
_:a  foaf:mbox      <mailto:alice@work.example> .

_:b  foaf:name      "Bob" .
_:b  foaf:mbox      "bob@work.example" .
```

รูปที่ 3.188 ข้อมูลที่ใช้ในการ query

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox
 WHERE { ?x foaf:name  ?name ;
         foaf:mbox  ?mbox .
       FILTER isIRI(?mbox) }
```

รูปที่ 3.189 การ query

name	Mbox
"Alice"	<mailto:alice@work.example>

รูปที่ 3.190 ผลลัพธ์การ query

3.4.49 isBlank

```
xsd:boolean   ISBLANK (RDF term term)
```

รูปที่ 3.191 isBlank

เป็นฟังก์ชันตรวจสอบว่ามีค่าเป็นโหนดว่างหรือไม่

```
@prefix a:      <http://www.w3.org/2000/10/annotation-ns#> .
@prefix dc:     <http://purl.org/dc/elements/1.1/> .
@prefix foaf:   <http://xmlns.com/foaf/0.1/> .

_:a  a:annotates  <http://www.w3.org/TR/rdf-sparql-query/> .
_:a  dc:creator   "Alice B. Toeclips" .

_:b  a:annotates  <http://www.w3.org/TR/rdf-sparql-query/> .
_:b  dc:creator   _:c .
_:c  foaf:given   "Bob".
_:c  foaf:family  "Smith".
```

รูปที่ 3.192 ข้อมูลที่ใช้ในการ query

```
PREFIX a:    <http://www.w3.org/2000/10/annotation-ns#>
PREFIX dc:   <http://purl.org/dc/elements/1.1/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?given ?family
 WHERE { ?annot  a:annotates <http://www.w3.org/TR/rdf-sparql-query/> .
      ?annot  dc:creator  ?c .
      OPTIONAL { ?c  foaf:given  ?given ; foaf:family  ?family } .
      FILTER isBlank(?c)}
```

รูปที่ 3.193 การ query

given	Family
"Bob"	"Smith"

รูปที่ 3.194 ผลลัพธ์การ query

3.4.50 isLiteral

xsd:boolean ISLITERAL (RDF term term)

รูปที่ 3.195 isLiteral

เป็นฟังก์ชันตรวจสอบว่ามีค่าเป็น Literal หรือไม่

3.4.51 str

simple literal STR (literal ltrl)
simple literal STR (IRI rsrc)

รูปที่ 3.196 str

เป็นฟังก์ชันตรวจสอบว่ามีค่าเป็น string หรือไม่

3.4.52 lang

simple literal LANG (literal ltrl)

รูปที่ 3.197 lang

เป็นฟังก์ชันถามภาษา

3.4.53 datatype

IRI DATATYPE (typed literal typedLit)
IRI DATATYPE (simple literal simpleLit)

รูปที่ 3.198 datatype

เป็นฟังก์ชันถามชนิดข้อมูล

3.4.54 logical-or

xsd:boolean xsd:boolean left || xsd:boolean right

รูปที่ 3.199 logical-or

3.4.55 logical-and

xsd:boolean xsd:boolean left && xsd:boolean right

รูปที่ 3.200 logical-and

3.4.56 langMatches

ให้แสดงข้อมูลที่เป็นภาษาที่ต้องการ

```
@prefix dc:     <http://purl.org/dc/elements/1.1/> .

_:a  dc:title       "That Seventies Show"@en .
_:a  dc:title       "Cette Série des Années Soixante-dix"@fr .
_:a  dc:title       "Cette Série des Années Septante"@fr-BE .
_:b  dc:title       "Il Buono, il Bruto, il Cattivo" .
```

รูปที่ 3.201 ข้อมูลที่ใช้ในการ query

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT ?title
 WHERE { ?x dc:title  "That Seventies Show"@en ;
          dc:title  ?title .
        FILTER langMatches( lang(?title), "FR" ) }
```

รูปที่ 3.202 การ query

Title
"Cette Série des Années Soixante-dix"@fr
"Cette Série des Années Septante"@fr-BE

รูปที่ 3.203 ผลลัพธ์การ query

3.4.57 regex

```
xsd:boolean   REGEX (simple literal text, simple literal pattern)
xsd:boolean   REGEX (simple literal text, simple literal pattern, simple literal flags)
```

รูปที่ 3.204 regex

ตรวจสอบรูปแบบตาม regular expression ได้

```
@prefix foaf:      <http://xmlns.com/foaf/0.1/> .

_:a  foaf:name     "Alice".
_:b  foaf:name     "Bob" .
```

รูปที่ 3.205 regex

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name
 WHERE { ?x foaf:name  ?name
       FILTER regex(?name, "^ali", "i") }
```

รูปที่ 3.206 การ query

Name
"Alice"

รูปที่ 3.207 ผลลัพธ์การ query

3.4.58 Constructor Functions

SPARQL สามารถใช้ constructor functions จาก Xpath ได้ เช่น

bool = xsd:boolean

dbl = xsd:double

flt = xsd:float

dec = xsd:decimal

int = xsd:integer

dT = xsd:dateTime

str = xsd:string

IRI = IRI

ltrl = simple literal

3.4.59 Extensible Value Testing

ดึงฟังก์ชันที่มีโปรแกรมเมอร์คนอื่นเขียนมาใช้ได้

```
xsd:boolean   func:even (numeric value)
```

รูปที่ 3.208 การดึงฟังก์ชันที่มีโปรแกรมเมอร์คนอื่นเขียนมาใช้

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX func: <http://example.org/functions#>
SELECT ?name ?id
WHERE { ?x foaf:name  ?name ;
        func:empId   ?id .
      FILTER (func:even(?id)) }
```

รูปที่ 3.209 การ query

บทที่ 4

การทดลอง

โครงงานนี้จะแบ่งการทดลองออกเป็นสามตอนดังนี้

1 XML การทดลอง Query เอกสาร xml โดยใช้ภาษา XQuery

2 RDF การทดลอง สร้าง และ Query เอกสาร RDF โดยใช้API Jena ใน Java และภาษา SPARQL

3 OWL การทดลอง สร้าง, แลกเปลี่ยน และ Query ไฟล์ Ontology (.owl) และไฟล์ RDF โดยใช้โปรแกรม Protégé

4.1 XML

จะเป็นการสร้างเอกสาร XML ขึ้นมา แล้วทดลอง Query โดยใช้ภาษา XQuery

4.1.1 จุดประสงค์การทดลอง

4.1.1.1 เพื่อศึกษาความสามารถของภาษา XQuery ว่าสามารถ Query ข้อมูลที่มีโครงสร้างแบบ Relational Database ได้ทุกอย่างตามที่ SQL มีหรือไม่ และถ้าสามารถ Query ได้จะมีความคล้ายคลึงหรือแตกต่างกันอย่างไร

4.1.1.2 เพื่อศึกษาความสามารถของภาษา XQuery ว่าสามารถ Query ข้อมูลที่ไม่ใช่โครงสร้างแบบ Relational Database เช่น Hierarchy ได้หรือไม่ และถ้าสามารถ Query ได้จะมีความคล้ายคลึงหรือแตกต่างกันอย่างไรกับการ Query ในข้อที่ 4.1.1

4.1.2 ขั้นตอนการทดลอง

4.1.2.1 นำตัวอย่างข้อมูล PRESIDENT จากวิชา 01072127 DATABASE SYSTEMS มาแปลเป็นเอกสาร XML สองโครงสร้าง

โครงสร้างแบบที่หนึ่ง จะแยกตารางละหนึ่งเอกสาร XML รวมเป็นทั้งหมด 7 เอกสาร XMLคือ

TABLE_PRESIDENT.xml,TABLE_PRES_MARRIAGE.xml,TABLE_PRES_HOBBY.xml, TABLE_ADMINISTRATION.xml, TABLE_ADMIN_PR_VP.xml, TABLE_STATE.xml, TABLE_ELECTION.xml

โครงสร้างแบบที่สอง จะมีเพียงเอกสาร XML เพียงเอกสารเดียวคือ PRESIDENT.xml โดยที่จะมีลักษณะเป็นระดับชั้น หลักคือจะจัดให้ข้อมูลที่เกี่ยวกับ PRESIDENT ที่ในแบบแรกจะแยกตาราง เช่น PRES_MARRIAGE, PRES_HOBBY, ADMINISTRATION, ADMIN_PR_VP มารวมไว้ใน PRESIDENT ที่เกี่ยวข้อง และแต่ละ PRESIDENT จะแยกต่าง

พรรคการเมืองอีกครั้งหนึ่ง ที่ไม่เกี่ยวคือ STATE และ ELECTION จะแยกเป็นโหนดภายนอกต่างหาก

ดังนั้นจึงมีสามโหนดใหญ่ คือ PARTYS, TABLE_STATE และ TABLE_ELECTION โครงสร้างของเอกสารโดยละเอียดสามารถดูได้จากหัวข้อโครงสร้างต้นไม้ของเอกสาร XML ที่ใช้ทดลอง

4.1.2.2 จากนั้น Query แบบโจทย์แบบเรียน เริ่มบทที่ 5 ถึง บทที่ 14

4.1.2.3 วิเคราห์หาความเหมือนและแตกต่างของ SQL, XQuery แบบที่ 1 และ XQuery แบบที่ 2

4.1.3 โครงสร้างต้นไม้ของเอกสาร XML ที่ใช้ทดลอง

4.1.3.1 แบบแรก แยกตารางละหนึ่งเอกสาร XML รวมเป็นทั้งหมด 7 เอกสาร XML

```
<TABLE_PRESIDENT>
        <PRESIDENT>
                <PRES_NAME>
                <BIRTH_YR>
                <YRS_SERV>
                <DEATH_AGE>
                <PARTY>
                <STATE_BORN>
```

รูปที่ 4.1 โครงสร้าง Element ในไฟล์ TABLE_PRESIDENT.xml

```
<TABLE_PRES_MARRIAGE>
        <PRES_MARRIAGE>
                <PRES_NAME>
                <SPOUSE_NAME>
                <PR_AGE>
                <SR_AGE>
                <NR_CHILDREN>
                <MAR_YEAR>
```

รูปที่ 4.2 โครงสร้าง Element ในไฟล์ TABLE_PRES_MARRIAGE.xml

```
<TABLE_PRES_HOBBY>
        <PRES_HOBBY>
                <PRES_NAME>
                <HOBBY>
```

รูปที่ 4.3 โครงสร้าง Element ในไฟล์ TABLE_PRES_HOBBY.xml

```
<TABLE_ADMINISTRATION>
          <ADMINISTRATION>

                <ADMIN_NR>

                <PRES_NAME>

              <YEAR_INAUGURATED>
```

รูปที่ 4.4 โครงสร้าง Element ในไฟล์ TABLE_ADMINISTRATION.xml

```
<TABLE_ADMIN_PR_VP>
        <ADMIN_PR_VP>
                <ADMIN_NR>
                <PRES_NAME>
                <VICE_PRES_NAME>
```

รูปที่ 4.5 โครงสร้าง Element ในไฟล์ TABLE_ADMIN_PR_VP.xml

```
<TABLE_STATE>
        <STATE>
                <STATE_NAME>
                <ADMIN_ENTERED>
                <YEAR_ENTERED>
```

รูปที่ 4.6 โครงสร้าง Element ในไฟล์ TABLE_STATE.xml

```
<TABLE_ELECTION>
        <ELECTION>
                <ELECTION_YEAR>
                <CANDIDATE>
                <VOTES>
                <WINNER_LOSER_INDIC>
```

รูปที่ 4.7 โครงสร้าง Element ในไฟล์ TABLE_ELECTION.xml

4.1.3.2 แบบที่สอง ซึ่งมีลักษณะเป็นลำดับชั้น

```
<ROOT>
        <PARTYS>
                <PARTY>
                        <PARTY_NAME>
                        <PRESIDENT>
```

<PRES_NAME>

<BIRTH_YR>

<DEATH_AGE>

<STATE_BORN>

<HOBBY>

<PRES_MARRIAGE>

<SPOUSE_NAME>

<PR_AGE>

<SR_AGE>

<NR_CHILDREN>

<MAR_YEAR>

<ADMINISTRATION>

<ADMIN_NR>

<YEAR_INAUGURATED>

<ADMIN_PR_VP>

<ADMIN_NR>

<VICE_PRES_NAME>

<TABLE_STATE>

<STATE>

```
                    <STATE_NAME>
                    <ADMIN_ENTERED>
                    <YEAR_ENTERED>
        <TABLE_ELECTION>
            <ELECTION>
                    <ELECTION_YEAR>
                    <CANDIDATE>
                    <VOTES>
                    <WINNER_LOSER_INDIC>
```

รูปที่ 4.8 โครงสร้าง Element ในไฟล์ PRESIDENT.xml

4.1.4 ตัวอย่างการทดลอง

4.1.4.1 การเลือกทั้งหมดหรือการเลือกคอลัมน์บางรายการจากตาราง (Selecting All or Particular Columns from One Table)

EX.ให้แสดงรายชื่อ ปีเกิด, อายุเมื่อถึงอสัญกรรม (ถ้ามี), จำนวนปีที่ตำแหน่ง รัฐที่ท่านเกิด และพรรคการเมืองที่สังกัดจากตาราง recent presidents (เปลี่ยนลำดับของการ SELECT)

โค๊ดการ Query

SQL
SELECT PRES_NAME,BIRTH_YR,DEATH_AGE,YRS_SERV, - STATE_BORN,PARTY- FROM RECENT_PRESIDENTS

XQuery แบบที่ 1

```
for $i in doc("TABLE_PRESIDENT.xml")/TABLE_PRESIDENT/PRESIDENT
return   <TR>
                <TD>{ $i/PRES_NAME }</TD>
                <TD>{$i/BIRTH_YR}</TD>
                <TD>{ $i/DEATH_AGE }</TD>
                <TD>{ $i/YRS_SERV }</TD>
                <TD>{ $i/STATE_BORN }</TD>
                <TD>{ $i/PARTY }</TD>
        </TR>
```

XQuery แบบที่ 2

```
for $i in doc("PRESIDENT.xml")//PRESIDENT
return   <TR>
                <TD>{ $i/PRES_NAME }</TD>
                <TD>{$i/BIRTH_YR}</TD>
                <TD>{ $i/DEATH_AGE }</TD>
                <TD>{ $i/YRS_SERV }</TD>
                <TD>{ $i/STATE_BORN }</TD>
                <TD>{ $i/../PARTY_NAME }</TD>
        </TR>
```

ตารางที่ 4.1 เปรียบเทียบ SQL กับ XQuery แบบที่ 1 ในตัวอย่างที่ 4.1.4.1

SQL เทียบกับ XQuery แบบที่ 1	
สิ่งที่เหมือนกัน	**สิ่งที่แตกต่างกัน**
- FROM ใน SQL คล้ายกับ For ใน XQuery เพราะเลือกว่าจะนำข้อมูลจากที่ใดมาประมวลมวลเหมือนกัน - SELECT ใน SQL คล้ายกับ return ใน XQuery เพราะ เป็นเลือกว่าจะเอาอะไรมาแสดง	- For ใน XQuery จะชี้ไปได้ลึกว่า FROM เพราะ FROM บอกว่าเอามาจากตารางใด For จะอ้างไปลึกไปถึงอีลิเมนได้ (เทียบได้กับ Columns) -ใน XQuery จะเน้นการอ้างถึงข้อมูลด้วยตัวแปรเป็นหลัก

ตารางที่ 4.2 เปรียบเทียบ XQuery ทั้ง 2 แบบ ในตัวอย่างที่ 4.1.4.1

เทียบ XQuery ทั้งสองแบบ	
สิ่งที่เหมือนกัน	สิ่งที่แตกต่างกัน
-ส่วนใหญ่จะเหมือนกัน	- ชื่อPARTY เป็นโหนดที่อยู่บน$i จึงตรงใช้.. อ้างโดยแม่ก่อน

ผลลัพธ์

SQL

รูปที่ 4.9 ผลลัพธ์ ของ SQL ใน ตัวอย่างที่ 4.1.4.1

XQuery แบบที่ 1

PRES_NAME	BIRTH_YR	DEATH_AGE	YRS_SERV	STATE_BORN	PARTY
Washington G	1732	67	7	Virginia	Federalist
Adams J	1735	90	4	Massachusetts	Federalist
Jefferson T	1743	83	8	Virginia	Demo-Rep
Madison J	1751	85	8	Virginia	Demo-Rep
Monroe J	1758	73	8	Virginia	Demo-Rep
Adams J Q	1767	80	4	Massachusetts	Demo-Rep
Jackson A	1767	78	8	South Carolina	Democratic
Van Buren M	1782	79	4	New York	Democratic
Harrison W H	1773	68	0	Virginia	Whig
Tyler J	1790	71	3	Virginia	Whig
Polk J K	1795	53	4	North Carolina	Democratic
Taylor Z	1784	65	1	Virginia	Whig
Fillmore M	1800	74	2	New York	Whig
Pierce F	1804	64	4	New Hampshire	Democratic
Buchanan J	1791	77	4	Pennsylvania	Democratic
Lincoln A	1809	56	4	Kentucky	Republican
Johnson A	1808	66	3	North Carolina	Democratic
Grant U S	1822	63	8	Ohio	Republican
Hayes R B	1822	70	4	Ohio	Republican
Garfield J A	1831	49	0	Ohio	Republican
Arthur C A	1830	56	3	Vermont	Republican
Cleveland G	1837	71	8	New Jersey	Democratic
Harrison B	1833	67	4	Ohio	Republican
McKinley W	1843	58	4	Ohio	Republican
Roosevelt T	1858	60	7	New York	Republican
Taft W H	1857	72	4	Ohio	Republican
Wilson W	1856	67	8	Virginia	Democratic
Harding W G	1865	57	2	Ohio	Republican
Coolidge C	1872	60	5	Vermont	Republican
Hoover H C	1874	90	4	Iowa	Republican
Roosevelt F D	1882	63	12	New York	Democratic
Truman H S	1884	88	7	Missouri	Democratic
Eisenhower D D	1890	79	8	Texas	Republican
Kennedy J F	1917	46	2	Massachusetts	Democratic
Johnson L B	1908	65	5	Texas	Democratic
Nixon R M	1913	?	5	California	Republican
Ford G R	1913	?	2	Nebraska	Republican
Carter J E	1924	?	4	Georgia	Democratic
Reagan R	1911	?	3	Illinois	Republican

รูปที่ 4.10 ผลลัพธ์ ของ XQuery แบบที่ 1 ใน ตัวอย่างที่ 4.1.4.1

XQuery แบบที่ 2

PRES_NAME	BIRTH_YR	DEATH_AGE	YRS_SERV	STATE_BORN	PARTY
Washington G	1732	67	7	Virginia	Federalist
Adams J	1735	90	4	Massachusetts	Federalist
Jefferson T	1743	83	8	Virginia	Demo-Rep
Madison J	1751	85	8	Virginia	Demo-Rep
Monroe J	1758	73	8	Virginia	Demo-Rep
Adams J Q	1767	80	4	Massachusetts	Demo-Rep
Jackson A	1767	78	8	South Carolina	Democratic
Van Buren M	1782	79	4	New York	Democratic
Harrison W H	1773	68	0	Virginia	Whig
Tyler J	1790	71	3	Virginia	Whig
Polk J K	1795	53	4	North Carolina	Democratic
Taylor Z	1784	65	1	Virginia	Whig
Fillmore M	1800	74	2	New York	Whig
Pierce F	1804	64	4	New Hampshire	Democratic
Buchanan J	1791	77	4	Pennsylvania	Democratic
Lincoln A	1809	56	4	Kentucky	Republican
Johnson A	1808	66	3	North Carolina	Democratic
Grant U S	1822	63	8	Ohio	Republican
Hayes R B	1822	70	4	Ohio	Republican
Garfield J A	1831	49	0	Ohio	Republican
Arthur C A	1830	56	3	Vermont	Republican
Cleveland G	1837	71	8	New Jersey	Democratic
Harrison B	1833	67	4	Ohio	Republican
McKinley W	1843	58	4	Ohio	Republican
Roosevelt T	1858	60	7	New York	Republican
Taft W H	1857	72	4	Ohio	Republican
Wilson W	1856	67	8	Virginia	Democratic
Harding W G	1865	57	2	Ohio	Republican
Coolidge C	1872	60	5	Vermont	Republican
Hoover H C	1874	90	4	Iowa	Republican
Roosevelt F D	1882	63	12	New York	Democratic
Truman H S	1884	88	7	Missouri	Democratic
Eisenhower D D	1890	79	8	Texas	Republican
Kennedy J F	1917	46	2	Massachusetts	Democratic
Johnson L B	1908	65	5	Texas	Democratic
Nixon R M	1913	?	5	California	Republican
Ford G R	1913	?	2	Nebraska	Republican
Carter J E	1924	?	4	Georgia	Democratic
Reagan R	1911	?	3	Illinois	Republican

รูปที่ 4.11 ผลลัพธ์ ของ XQuery แบบที่ 2 ใน ตัวอย่างที่ 4.1.4.1

4.1.4.2 การเลือกแบบระบุแถวในตาราง (Selecting Specified Rows of One Table)

EX. ให้แสดงละเอียดจากตาราง PRESIDENT เกี่ยวกับประธานาธิบดีทุกท่านที่เกิดในรัฐ Texas หรือท่านอยู่พรรค Republican แต่ไม่ได้เกิดในรัฐ California

โค๊ดการ Query

SQL

```
SELECT      *-
FROM        RECENT_PRESIDENTS-
WHERE       STATE_BORN = 'Texas' OR (PARTY = 'Republican' AND NOT
STATE_BORN = 'California')
```

XQuery แบบที่ 1

```
for $i in doc("TABLE_PRESIDENT.xml")/TABLE_PRESIDENT/PRESIDENT
where $i/STATE_BORN = 'Texas' or ($i/PARTY = 'Republican' and $i/STATE_BORN !=
'California')
return   <TR>
                <TD>{ $i/PRES_NAME }</TD>
                <TD>{$i/BIRTH_YR}</TD>
                <TD>{ $i/YRS_SERV }</TD>
                <TD>{ $i/DEATH_AGE }</TD>
                <TD>{ $i/PARTY }</TD>
                <TD>{ $i/STATE_BORN }</TD> </TR>
```

XQuery แบบที่ 2

```
for $i in doc("TABLE_PRESIDENT.xml")/TABLE_PRESIDENT/PRESIDENT
where $i/STATE_BORN = 'Texas' or ($i/PARTY = 'Republican' and $i/STATE_BORN !=
'California')
return   <TR>
                <TD>{ $i/PRES_NAME }</TD>
                <TD>{$i/BIRTH_YR}</TD>
                <TD>{ $i/YRS_SERV }</TD>
                <TD>{ $i/DEATH_AGE }</TD>
                <TD>{ $i/../PARTY_NAME }</TD>
                <TD>{ $i/STATE_BORN }</TD>
        </TR>
```

ตารางที่ 4.3 เปรียบเทียบ SQL กับ XQuery แบบที่ 1 ในตัวอย่างที่ 4.1.4.2

SQL เทียบกับ XQuery แบบที่ 1	
สิ่งที่เหมือนกัน	**สิ่งที่แตกต่างกัน**
- มีเครื่องหมาย =, !=,or,and,	- XQuery ไม่มี not แต่มีฟังก์ชัน not() แทน

ผลลัพธ์

SQL

PRES_NAME	BIRTH_YR	YRS_SERV	DEATH_AGE	PARTY	STATE_BORN
Lincoln A	1809	4	56	Republican	Kentucky
Grant U S	1822	8	63	Republican	Ohio
Hayes R B	1822	4	70	Republican	Ohio
Garfield J A	1831	0	49	Republican	Ohio
Arthur C A	1830	3	56	Republican	Vermont
Harrison B	1833	4	67	Republican	Ohio
McKinley W	1843	4	58	Republican	Ohio
Roosevelt T	1858	7	60	Republican	New York
Taft W H	1857	4	72	Republican	Ohio
Harding W G	1865	2	57	Republican	Ohio
Coolidge C	1872	5	60	Republican	Vermont
Hoover H C	1874	4	90	Republican	Iowa
Eisenhower D D	1890	8	72	Republican	Texas
Johnson L B	1908	5	65	Democratic	Texas
Ford G R	1913	2	?	Republican	Nebraska
Reagan R	1911	3	?	Republican	Illinois

* END OF RESULT ***** 16 ROWS DISPLAYED *****

รูปที่ 4.12 ผลลัพธ์ ของ SQL ใน ตัวอย่างที่ 4.1.4.2

XQuery แบบที่ 1

PRES_NAME	BIRTH_YR	YRS_SERV	DEATH_AGE	PARTY	STATE_BORN
Lincoln A	1809	4	56	Republican	Kentucky
Grant U S	1822	8	63	Republican	Ohio
Hayes R B	1822	4	70	Republican	Ohio
Garfield J A	1831	0	49	Republican	Ohio
Arthur C A	1830	3	56	Republican	Vermont
Harrison B	1833	4	67	Republican	Ohio
McKinley W	1843	4	58	Republican	Ohio
Roosevelt T	1858	7	60	Republican	New York
Taft W H	1857	4	72	Republican	Ohio
Harding W G	1865	2	57	Republican	Ohio
Coolidge C	1872	5	60	Republican	Vermont
Hoover H C	1874	4	90	Republican	Iowa
Eisenhower D D	1890	8	79	Republican	Texas
Johnson L B	1908	5	65	Democratic	Texas
Ford G R	1913	2	?	Republican	Nebraska
Reagan R	1911	3	?	Republican	Illinois

รูปที่ 4.13 ผลลัพธ์ ของ XQuery แบบที่ 1 ใน ตัวอย่างที่ 4.1.4.2

XQuery แบบที่ 2

PRES_NAME	BIRTH_YR	YRS_SERV	DEATH_AGE	PARTY	STATE_BORN
Lincoln A	1809	4	56	Republican	Kentucky
Grant U S	1822	8	63	Republican	Ohio
Hayes R B	1822	4	70	Republican	Ohio
Garfield J A	1831	0	49	Republican	Ohio
Arthur C A	1830	3	56	Republican	Vermont
Harrison B	1833	4	67	Republican	Ohio
McKinley W	1843	4	58	Republican	Ohio
Roosevelt T	1858	7	60	Republican	New York
Taft W H	1857	4	72	Republican	Ohio
Harding W G	1865	2	57	Republican	Ohio
Coolidge C	1872	5	60	Republican	Vermont
Hoover H C	1874	4	90	Republican	Iowa
Eisenhower D D	1890	8	79	Republican	Texas
Johnson L B	1908	5	65	Democratic	Texas
Ford G R	1913	2	?	Republican	Nebraska
Reagan R	1911	3	?	Republican	Illinois

รูปที่ 4.14 ผลลัพธ์ ของ XQuery แบบที่ 2 ใน ตัวอย่างที่ 4.1.4.2

4.1.4.3 ฟังก์ชันที่ทั่วไปที่มีให้เรียกใช้ (Built-in Functions)

EX. ให้หาว่ามีประธานาธิบดีที่สังกัดพรรค Republican เป็นจำนวนเท่าไร

โค๊ดการ Query

SQL	
SELECT	MIN(PARTY), COUNT(*)-
FROM	PRESIDENTS-
WHERE	PARTY='Republican'

XQuery แบบที่ 1
let $i := doc("TABLE_PRESIDENT.xml")/TABLE_PRESIDENT/PRESIDENT[PARTY = 'Republican']/PARTY return <TR> <TD>{ max(for $string in $i return string($string))}</TD> <TD>{ count($i)}</TD> </TR>

XQuery แบบที่ 2
let $i := doc("PRESIDENT.xml")//PRESIDENT[PARTY = 'Republican']/PARTY return <TR> <TD>{ distinct-values($i)}</TD> <TD>{ count($i)}</TD> </TR>

ตารางที่ 4.4 เปรียบเทียบ SQL กับ XQuery แบบที่ 1 ในตัวอย่างที่ 4.1.4.3

SQL เทียบกับ XQuery แบบที่ 1	
สิ่งที่เหมือนกัน	**สิ่งที่แตกต่างกัน**
-มีฟังก์ชัน min, max,count เหมือนกัน -let จะคล้าย from เพราะเลือกว่าจะนำข้อมูลจากที่ใดมาประมวลมวลเหมือนกัน	-XQuery จะใช้ min, max,count ต้องใช้คู่กับ let -let จะคล้าย for แต่จะให้ค่าคืนเป็นชุดเดียวกัน -max, min ใน XQuery จะใช้ยากกว่าเพราะทำงานกับค่าคงที

ตารางที่ 4.5 เปรียบเทียบ XQuery ทั้ง 2 แบบในตัวอย่างที่ 4.1.4.3

เทียบ XQuery ทั้งสองแบบ	
สิ่งที่เหมือนกัน	**สิ่งที่แตกต่างกัน**
	-แบบสองใช้ distinct-values จะง่ายกว่า

ผลลัพธ์

SQL


MIN(PARTY) COUNT(*)
Republican 16

รูปที่ 4.15 ผลลัพธ์ ของ SQL ใน ตัวอย่างที่ 4.1.4.3

XQuery แบบที่ 1
MIN(PARTY) AVG(*) Republican 16

รูปที่ 4.16 ผลลัพธ์ ของ XQuery แบบที่ 1 ใน ตัวอย่างที่ 4.1.4.3

XQuery แบบที่ 2
MIN(PARTY) AVG(*) Republican 16

รูปที่ 4.17 ผลลัพธ์ ของ XQuery แบบที่ 2 ใน ตัวอย่างที่ 4.1.4.3

4.1.4.4 การคำนวณ (Calculation)

EX. ให้แสดงอายุเฉลี่ยเมื่อถึงอสัญกรรมของประธานาธิบดี (โดยไม่นับท่านที่ยังไม่ถึงอสัญกรรม)

โค๊ดการ Query

SQL	
SELECT	SUM(DEATH_AGE)/COUNT(*)-
FROM	PRESIDENT -
WHERE	DEATH_AGE IS NOT NULL

XQuery แบบที่ 1
let $i := doc("TABLE_PRESIDENT.xml")//PRESIDENT[DEATH_AGE != '?'] return <TR> <TD>{ sum($i/DEATH_AGE) div count($i)}</TD> </TR>

XQuery แบบที่ 2

```
let $i := doc("PRESIDENT.xml")//PRESIDENT[DEATH_AGE != '?']
return   <TR>
                <TD>{ sum($i/DEATH_AGE) div count($i)}</TD>
         </TR>
```

ผลลัพธ์

SQL

รูปที่ 4.18 ผลลัพธ์ ของ SQL ใน ตัวอย่างที่ 4.4.1.4

XQuery แบบที่ 1

SUM(DEATH_AGE)/COUNT(*)
68.85714285714286

รูปที่ 4.19 ผลลัพธ์ ของ XQuery แบบที่ 1ใน ตัวอย่างที่ 4.4.1.4

XQuery แบบที่ 2

SUM(DEATH_AGE)/COUNT(*)
68.85714285714286

รูปที่ 4.20 ผลลัพธ์ ของ XQuery แบบที่ 2ใน ตัวอย่างที่ 4.4.1.4

4.1.4.5 การจัดกลุ่มตามลักษณะหน้าตา (The Grouping Feature)

EX. ให้แสดงจำนวนบุตรของการแต่งงานครั้งที่มากที่สุด มากกว่า จำนวนบุตรของการแต่งงานครั้งที่น้อยที่สุด อยู่สองคนหรือมากกว่า ของประธานาธิบดีที่แต่งงานมากกว่าหนึ่งครั้ง

โค๊ดการ Query

SQL
SELECT PRES_NAME,MAX(NR_CHILDREN),MIN(NR_CHILDREN)- FROM PRES_MARRIAGE- GROUP BY PRES_NAME HAVING COUNT(*) >= 2 AND MAX(NR_CHILDREN) >= MIN(NR_CHILDREN)+2

XQuery แบบที่ 1
for $i in distinct-values(doc("TABLE_PRES_MARRIAGE.xml")//PRES_NAME) let $j := doc("TABLE_PRES_MARRIAGE.xml")//PRES_MARRIAGE[PRES_NAME = $i] where (count($j) >= 2) and (max($j/NR_CHILDREN) >= min($j/NR_CHILDREN)+2) return <TR> <TD> | {$i} </TD> <TD> \| {max($j/NR_CHILDREN)} </TD> <TD> | {min($j/NR_CHILDREN)} | </TD> </TR>

XQuery แบบที่ 2
for $i in distinct-values(doc("PRESIDENT.xml")//PRES_NAME) let $j := doc("PRESIDENT.xml")//PRESIDENT[PRES_NAME = $i]//PRES_MARRIAGE where (count($j) >= 2) and (max($j/NR_CHILDREN) >= min($j/NR_CHILDREN)+2) return <TR> <TD> | {$i} </TD> <TD> \| {max($j/NR_CHILDREN)} </TD> <TD> | {min($j/NR_CHILDREN)} | </TD> </TR>

ผลลัพธ์

SQL


รูปที่ 4.21 ผลลัพธ์ ของ SQL ใน ตัวอย่างที่ 4.4.1.5

XQuery แบบที่ 1
| PRES_NAME | MAX(NR_CHILDREN) | MIN(NR_CHILDREN) | | Fillmore M | 2 | 0 | | Roosevelt T | 5 | 1 | | Wilson W | 3 | 0 |

รูปที่ 4.22 ผลลัพธ์ ของ XQuery แบบที่ 1 ใน ตัวอย่างที่ 4.4.1.5

XQuery แบบที่ 2
| PRES_NAME | MAX(NR_CHILDREN) | MIN(NR_CHILDREN) | | Wilson W | 3 | 0 | | Fillmore M | 2 | 0 | | Roosevelt T | 5 | 1 |

รูปที่ 4.23 ผลลัพธ์ ของ XQuery แบบที่ 2 ใน ตัวอย่างที่ 4.4.1.5

4.1.4.6 การเลือกคอลัมน์หรือแถวจากหลาย ๆ ตาราง (Selecting Columns and Rows From Several Tables: Joining)

EX. ให้แสดงละเอียด เกี่ยวกับประธานาธิบดีที่แต่งงานก่อนอายุ 20 หรือ ภรรยาอายุก่อน 18

โค๊ดการ Query

SQL
SELECT PRESIDENT.PRES_NAME, BIRTH_YR, PR_AGE,SP_AGE, SPOUSE_NAME- FROM PRESIDENT,PRES_MARRIAGE- WHERE PRESIDENT.PRES_NAME = PRES_MARRIAGE.PRES_NAME- AND (PR_AGE < 20 OR SP_AGE < 18)- ORDER BY PR_AGE

XQuery แบบที่ 1

```
for $i in doc("TABLE_PRESIDENT.xml")//PRESIDENT
for $j in
doc("TABLE_PRES_MARRIAGE.xml")//PRES_MARRIAGE[PRES_NAME=$i/PRES_NAME]
where (number($j/PR_AGE) < 20) or (number($j/SR_AGE) < 18)
order by number($j/PR_AGE)
return   <TR>
                <TD> | {$i/PRES_NAME} </TD>
                <TD> | {$i/BIRTH_YR} </TD>
                <TD> | {$j/PR_AGE } </TD>
                <TD> | {$j/SR_AGE } </TD>
                <TD> | {$j/SPOUSE_NAME} </TD>
        </TR>
```

XQuery แบบที่ 2

```
for $i in doc("PRESIDENT.xml")/ /PRES_MARRIAGE
where (number($j/PR_AGE) < 20) or (number($j/SR_AGE) < 18)
order by number($j/PR_AGE)
return   <TR>
                <TD> | {$i/../PRES_NAME} </TD>
                <TD> | {$i/../BIRTH_YR} </TD>
                <TD> | {$i/PR_AGE } </TD>
                <TD> | {$i/SR_AGE } </TD>
                <TD> | {$i/SPOUSE_NAME} </TD>
        </TR>
```

ตารางที่ 4.6 เปรียบเทียบ XQuery แบบทั้ง 2 แบบในตัวอย่างที่ 4.4.1.6

เทียบ XQuery ทั้งสองแบบ	
สิ่งที่เหมือนกัน	สิ่งที่แตกต่างกัน
	-โครงสร้างแบบที่สองสามารถลดการ for ลดได้ เพราะ PRES_MARRIAGE เรียบตาม PRESIDENT อยู่แล้ว

ผลลัพธ์

SQL

รูปที่ 4.24 ผลลัพธ์ ของ SQL ใน ตัวอย่างที่ 4.4.1.6

XQuery แบบที่ 1

PRES_NAME	BIRTH_YR	PR_AGE	SR_AGE	SPOUSE_NAME
Johnson A	1808	18	16	McCardle E
Monroe J	1758	27	17	Kortright E

รูปที่ 4.25 ผลลัพธ์ ของ XQuery แบบที่ 1 ใน ตัวอย่างที่ 4.4.1.6

XQuery แบบที่ 2

PRES_NAME	BIRTH_YR	PR_AGE	SR_AGE	SPOUSE_NAME
Johnson A	1808	18	16	McCardle E
Monroe J	1758	27	17	Kortright E

รูปที่ 4.26 ผลลัพธ์ ของ XQuery แบบที่ 2 ใน ตัวอย่างที่ 4.4.1.6

4.1.4.7 Subquerie

EX. ให้แสดงรายชื่อและอายุเมื่อถึงอสัญกรรม ของประธานาธิบดีที่ถึงอสัญกรรมอายุน้อยที่สุด

โค๊ดการ Query

SQL	
SELECT	PRES_NAME, DEATH_AGE-
FROM	PRESIDENT-
WHERE	DEATH_AGE = -
(SELECT	MIN(DEATH_AGE)-
FROM	PRESIDENT)

XQuery แบบที่ 1

```
for $i in doc("TABLE_PRESIDENT.xml")//PRESIDENT
where number($i/DEATH_AGE) =
min(doc("TABLE_PRESIDENT.xml")//PRESIDENT[DEATH_AGE !=
    "?"]/DEATH_AGE)
return   <TR>
                <TD><B> | {$i/PRES_NAME}   </B></TD>
                <TD><B> | {$i/DEATH_AGE} | </B></TD>
         </TR>
```

XQuery แบบที่ 2

```
for $i in doc("PRESIDENT.xml")//PRESIDENT
where number($i/DEATH_AGE) = min(doc("PRESIDENT.xml")//PRESIDENT[DEATH_AGE
!= "?"]/DEATH_AGE)
return   <TR>
                <TD><B> | {$i/PRES_NAME}  </B></TD>
                <TD><B> | {$i/DEATH_AGE} |</B></TD>
         </TR>
```

ตารางที่ 4.7 เปรียบเทียบ SQL กับ XQuery แบบที่ 1 ในตัวอย่างที่ 4.4.1.7

SQL เทียบกับ XQuery แบบที่ 1	
สิ่งที่เหมือนกัน	**สิ่งที่แตกต่างกัน**
	-ไม่จำเป็นต้องใช่ subqueries ก็ได้ สามารถกรองด้วย xpath ได้

ผลลัพธ์

SQL


รูปที่ 4.27 ผลลัพธ์ ของ SQL ใน ตัวอย่างที่ 4.4.1.7

XQuery แบบที่ 1
| PRES_NAME | DEATH_AGE | | Kennedy J F | 46 |

รูปที่ 4.28 ผลลัพธ์ ของ XQuery แบบที่ 1 ใน ตัวอย่างที่ 4.4.1.7

XQuery แบบที่ 2
| PRES_NAME | DEATH_AGE | | Kennedy J F | 46 |

รูปที่ 4.29 ผลลัพธ์ ของ XQuery แบบที่ 2 ใน ตัวอย่างที่ 4.4.1.7

4.1.4.8 การใช้มากกว่าหนึ่งครั้งจากตารางเดียวกัน (Use of More Than One Copy of a table)

EX. ให้แสดงประธานาธิบดีที่เกิดในปีเดียวกันในอยู่ในแถวเดียวกัน โดยที่เป็นคนละคนกันและข้อมูลไม่ซ้ำกัน (คู่ไม่ซ้ำกัน)

โค๊ดการ Query

SQL	
SELECT	T1.PRES_NAME,T1.BIRTH_YR,T2.PRES_NAME,T2.BIRTH_YR-
FROM	PRESIDENT T1,PRESIDENT T2-
WHERE	T1.BIRTH_YR=T2.BIRTH_YR AND T1.PRES_NAME <
T2.PRES_NAME	

XQuery แบบที่ 1

```
for $i in doc("TABLE_PRESIDENT.xml")//PRESIDENT
for $j in doc("TABLE_PRESIDENT.xml")//PRESIDENT
where ($i/BIRTH_YR = $j/BIRTH_YR) and ($i/PRES_NAME < $j/PRES_NAME )
return   <TR>
                <TD><B>{$i/PRES_NAME}</B></TD>
                <TD><B>{$i/BIRTH_YR}</B></TD>
                <TD><B>{$j/PRES_NAME}</B></TD>
                <TD><B>{$j/BIRTH_YR}</B></TD>
         </TR>
```

XQuery แบบที่ 2

```
for $i in doc("PRESIDENT.xml")//PRESIDENT
for $j in doc("PRESIDENT.xml")//PRESIDENT
where ($i/BIRTH_YR = $j/BIRTH_YR) and ($i/PRES_NAME < $j/PRES_NAME )
return   <TR>
                <TD><B>{$i/PRES_NAME}</B></TD>
                <TD><B>{$i/BIRTH_YR}</B></TD>
                <TD><B>{$j/PRES_NAME}</B></TD>
                <TD><B>{$j/BIRTH_YR}</B></TD>
         </TR>
```

ตารางที่ 4.8 เปรียบเทียบ SQL กับ XQuery แบบที่ 1 ในตัวอย่างที่ 4.4.1.8

SQL เทียบกับ XQuery แบบที่ 1	
สิ่งที่เหมือนกัน	**สิ่งที่แตกต่างกัน**
- Use of More Than One Copy of a table คล้ายกันมากเพราะใช้หลักอ้างตัวแปรอยู่แล้ว	

ผลลัพธ์

SQL


รูปที่ 4.30 ผลลัพธ์ ของ SQL ใน ตัวอย่างที่ 4.4.1.8

XQuery แบบที่ 1			
PRES_NAME	BIRTH_YR	PRES_NAME	BIRTH_YR
Adams J Q	1767	Jackson A	1767
Grant U S	1822	Hayes R B	1822
Ford G R	1913	Nixon R M	1913

รูปที่ 4.31 ผลลัพธ์ ของ XQuery แบบที่ 1ใน ตัวอย่างที่ 4.4.1.8

XQuery แบบที่ 2			
PRES_NAME	BIRTH_YR	PRES_NAME	BIRTH_YR
Adams J Q	1767	Jackson A	1767
Grant U S	1822	Hayes R B	1822
Ford G R	1913	Nixon R M	1913

รูปที่ 4.32 ผลลัพธ์ ของ XQuery แบบที่ 2ใน ตัวอย่างที่ 4.4.1.8

4.1.4.9 การ Subquery ซ้อนเป็นลำดับชั้น (Correlated Subqueries.)

EX. ให้แสดงรายชื่อประธานาธิบดีทุกท่านเป็นประธานาธิบดีครั้งแรกก่อนอายุ 45

โค๊ดการ Query

SQL	
SELECT	PRES_NAME,BIRTH_YR-
FROM	PRESIDENT-
WHERE	BIRTH_YR + 45 >-
(SELECT	MIN(YEAR_INAUGURATED)-
FROM	ADMINISTRATION-
WHERE	ADMINISTRATION.PRES_NAME = PRESIDENT.PRES_NAME)

XQuery แบบที่ 1

```
for $i in doc("TABLE_PRESIDENT.xml")//PRESIDENT
 where $i/BIRTH_YR + 45 > min(
        let $j :=
doc("TABLE_ADMINISTRATION.xml")//ADMINISTRATION[PRES_NAME=$i/PRES_NAME]
        return $j/YEAR_INAUGURATED
 )
return   <TR>
                <TD><B>{$i/PRES_NAME}</B></TD>
                <TD><B>{$i/BIRTH_YR}</B></TD>
        </TR>
```

XQuery แบบที่ 2

```
for $i in doc("PRESIDENT.xml")//PRESIDENT
 where $i/BIRTH_YR + 45 > min(
        for $x in doc("PRESIDENT.xml")//PRESIDENT[PRES_NAME=$i/PRES_NAME]
        let $j := $x//ADMINISTRATION
        return $j/YEAR_INAUGURATED
 )
return   <TR>
                <TD><B>{$i/PRES_NAME}</B></TD>
                <TD><B>{$i/BIRTH_YR}</B></TD>
        </TR>
```

ตารางที่ 4.9 เปรียบเทียบ SQL กับ XQuery แบบที่ 1 ในตัวอย่างที่ 4.4.1.9

SQL เทียบกับ XQuery แบบที่ 1	
สิ่งที่เหมือนกัน	**สิ่งที่แตกต่างกัน**
-ใช้หลักตัวแปรอ้างนอก subqueries ได้	

ผลลัพธ์

SQL


รูปที่ 4.33 ผลลัพธ์ ของ SQL ใน ตัวอย่างที่ 4.4.1.9

XQuery แบบที่ 1
| PRES_NAME | BIRTH_YR Roosevelt T 1858 Kennedy J F 1917

รูปที่ 4.34 ผลลัพธ์ ของ XQuery แบบที่ 1 ใน ตัวอย่างที่ 4.4.1.9

XQuery แบบที่ 2
| PRES_NAME | BIRTH_YR Kennedy J F 1917 Roosevelt T 1858

รูปที่ 4.35 ผลลัพธ์ ของ XQuery แบบที่ 2 ใน ตัวอย่างที่ 4.4.1.9

4.1.4.10 Subquery โดยการทดสอบการมีชีวิตอยู่ (Subqueries with Test for Existence)

EX. ให้แสดงรายชื่อประธานาธิบดีและอายุเมื่อถึงอสัญกรรม เฉพาะประธานาธิบดีที่ไม่เคยแต่งงานแล้ว

โค๊ดการ Query

SQL	
SELECT	PRES_NAME,DEATH_AGE-
FROM	PRESIDENT-
WHERE	NOT EXISTS-
(SELECT	*-
FROM	PRES_MARRIAGE-
WHERE	PRES_MARRIAGE.PRES_NAME=PRESIDENT.PRES_NAME)

XQuery แบบที่ 1
for $x in doc("TABLE_PRESIDENT.xml")//PRESIDENT where not (exists(doc("TABLE_PRES_MARRIAGE.xml")//PRES_MARRIAGE[PRES_NAME=$x/PRES_NAME])) return <TR> <TD><B>{$x/PRES_NAME }</B></TD> <TD><B>{$x/DEATH_AGE }</B></TD> </TR>

XQuery แบบที่ 2
for $x in doc("PRESIDENT.xml")//PRESIDENT where not (exists(doc("PRESIDENT.xml")//PRESIDENT[PRES_NAME=$x/PRES_NAME]//PRES_MARRIAGE)) return <TR> <TD><B>{$x/PRES_NAME }</B></TD> <TD><B>{$x/DEATH_AGE }</B></TD> </TR>

ตารางที่ 4.10 เปรียบเทียบ SQL กับ XQuery แบบที่ 1 ในตัวอย่างที่ 4.4.1.10

SQL เทียบกับ XQuery แบบที่ 1	
สิ่งที่เหมือนกัน	**สิ่งที่แตกต่างกัน**
	-สามารถลด subqueries ได้ด้วย xpath

ตารางที่ 4.11 เปรียบเทียบ XQuery ทั้ง 2 แบบในตัวอย่างที่ 4.4.1.10

เทียบ XQuery ทั้งสองแบบ	
สิ่งที่เหมือนกัน	**สิ่งที่แตกต่างกัน**
	-เปลี่ยนตำแหน่งการกรอง PRES_NAME เท่านั้นเอง

ผลลัพธ์

SQL
PRES_NAME DEATH_AGE Buchanan J 77

รูปที่ 4.36 ผลลัพธ์ ของ SQL ใน ตัวอย่างที่ 4.4.1.10

XQuery แบบที่ 1
| PRES_NAME | DEATH_AGE | Buchanan J 77

รูปที่ 4.37 ผลลัพธ์ ของ XQuery แบบที่ 1 ใน ตัวอย่างที่ 4.4.1.10

XQuery แบบที่ 2
| PRES_NAME | DEATH_AGE | Buchanan J 77

รูปที่ 4.38 ผลลัพธ์ ของ XQuery แบบที่ 2 ใน ตัวอย่างที่ 4.4.1.10

4.1.5 สรุป SQL เทียบกับ XQuery

ตารางที่ 4.12 สรุป SQL เทียบกับ XQuery

สิ่งที่เหมือนกัน	สิ่งที่แตกต่างกัน
-FROM ใน SQL คล้ายกับ For และ Letใน XQuery เพราะเลือกว่าจะนำข้อมูลจากที่ใดมาประมวลมวลเหมือนกัน	- For ใน XQuery จะชี้ไปได้ลึกว่า FROM เพราะ FROM บอกว่าเอามาจากตารางใด For จะอ้างไปลึกไปถึงอีลิเมนได้ (เทียบได้กับ Columns)
-SELECT ใน SQL คล้ายกับ return ใน XQuery เพราะ เป็นเลือกว่าจะเอาอะไรมาแสดง	-ใน XQuery จะเน้นการอ้างถึงข้อมูลด้วยตัวแปรเป็นหลัก
-มี ORDER BY เหมือนกัน และไม่เขียนอะไรเพิ่มเป็นการ เรียงจากน้อยไปมาก	-SQL ถ้าจะเรียงจากมากไปน้อย ใช้ DESC ส่วน XQuery ใช้ descending
- ข้อมูลที่ได้จาก for และ SELECT มีโอกาสซ้ำกันได้	-อยู่คนละทำตำแหน่ง คือ DISTINCT จะอยู่ใน SELECT (ส่วนแสดงผล) แต่ distinct-valuesจะอยู่ที่ for (การเลือกไม่ใช่การแสดงผล)
-SQL มี DISTINCT, XQuery มี distinct-values ซึ่งใช้กำจัดค่าที่ซ้ำออกไปได้	- ไม่เท่ากับใน SQL ใช้ ^=, ไม่เท่ากับใน XQuery ใช้ !=

สิ่งที่เหมือนกัน	สิ่งที่แตกต่างกัน
-มี where และ เครื่องหมายเปรียบเทียบเหมือนกัน	- XQuery ไม่มี not แต่มีฟังก์ชัน not() แทน
- มีเครื่องหมาย =, !=,>,<,>=,<=,or,and,+,-,*,(แต่หารใช้ div)	- คำสั่ง in ( ) ใน SQL XQuery ใช้ = () แทนได้
-มีฟังก์ชัน min, max, count, sum เหมือนกัน	-XQuery จะใช้ min, max,count ต้องใช้คู่กับ let
-ใน where สามารถใช้การคำนวณ บวก, ลบ,คูณ, หารได้เหมือนกัน	-let จะคล้าย for แต่จะให้ค่าคืนเป็นชุดเดียวกัน
- Use of More Than One Copy of a table คล้ายกันมากเพราะใช้หลักอ้างตัวแปรอยู่แล้ว	-max, min ใน XQuery จะใช้ยากกว่าเพราะทำงานกับค่าคงที
-ใช้หลักตัวแปรอ้างนอก subqueries ได้	-XQuery ไม่ใช้การหารเป็น / เพราะไปซ้ำกับการ Xpath
-มี EXISTSเหมือนกัน	- SQL จะมีปัญหาเรื่องทศนิยมและจำนวนเต็ม แต่ XQueryจะคิดเป็นทศนิยมอย่าง ถ้าอย่างให้ output ออกมาเหมือนจำเป็นต้องใช้ฟังก์ชัน xs:integer()
	- XQuery ไม่มี GROUP BY ต้องใช้ for ช่วยในการทำ GROUP BY
	- XQuery จะ join ตาราง ใช้หลัก ตัวแปร และ for แต่ถ้าจะ join สามตารางก็ต้องมีสาม for แต่ SQL มี FROM ครั้งเดียว
	-บางคิวรีไม่จำเป็นต้องใช่ subqueries ก็ได้ สามารถกรองด้วย xpath แทนได้

4.2 RDF

การทดลอง สร้าง และ Query เอกสาร RDF โดยใช้API Jena ใน Java และภาษา SPARQL

4.2.1 จุดประสงค์การทดลอง

4.2.1.1 เพื่อศึกษาความสามารถของ RDF ว่าสามารถออกแบบในระดับแนวความคิดได้ โดยการนำตัวอย่าง มาสร้างเป็นเอกสาร RDF

4.2.1.2 เพื่อศึกษาความสามารถของ SPARQL ว่าสามารถสืบค้นข้อมูลระดับแนวความคิดได้หรือไม่

4.2.2 ขั้นตอนการทดลอง

4.2.2.1 จากโจทย์ ในการประชุมขององค์กรศาสนานานาชาติ องค์กรต้องการเก็บข้อมูลการประชุม โดยได้ออกรายงานมาสองฉบับ ฉบับแรกเป็นรายงานเกี่ยวกับข้อมูลของศาสนาต่าง ๆ ส่วนในฉบับที่สองเกี่ยวข้องกับผู้พูดและหัวข้อที่พูด องค์กรศาสนานานาชาติยังต้องการให้ใช้รหัสศาสนา และรหัสพระคัมภีร์

ตารางที่ 4.13 รายงานที่ 1

RELIGION_NAME	TEACHER	ADHERENT(MILLIONS)	TEXTS
Christianity	Christ	400	Old Testament New Testament
Budhism	Budha	300	Sutra
Muslim	Mahamed	400	Koran
Hinduism	Khrisna	350	Upanishad Pakavat Kita

ตารางที่ 4.14 รายงานที่ 2

SPEAKER	TOPIC	NO.SESSIONS	TEXT READ
John	Christianity Hinduism	2 1	Pakavat Kita Koran Old Testament
Peter	Budhism	2	Koran
Muslim	Budhism Christianity	3 1	Old Testament New Testament Sutra

4.2.2.2 จากรายงานสามารถออกแบบเป็น Triples of the Data Model และ Graph of the data model ได้ดังนี้

ตารางที่ 4.15 Triples of the Data Model

Subject	Predicate	Object
http://CaseStudy/RELIGION/Christianity	http://www.kmitl.ac.th/~s7010050/relationship#HadRCODE	"R001"
http://CaseStudy/RELIGION/Christianity	http://www.kmitl.ac.th/~s7010050/relationship#HadTEACHER	"Christ"
http://CaseStudy/RELIGION/Christianity	http://www.kmitl.ac.th/~s7010050/relationship#HadADHERENT	"400"
http://CaseStudy/RELIGION/Christianity	http://www.kmitl.ac.th/~s7010050/relationship#HadTEXTS	http://CaseStudy/TEXTS/OldTestament
http://CaseStudy/RELIGION/Christianity	http://www.kmitl.ac.th/~s7010050/relationship#HadTEXTS	http://CaseStudy/TEXTS/NewTestament
http://CaseStudy/RELIGION/Christianity	http://www.kmitl.ac.th/~s7010050/relationship#IsSpoken	http://CaseStudy/SPEAK/JohnChristianity
http://CaseStudy/RELIGION/Christianity	http://www.kmitl.ac.th/~s7010050/relationship#IsSpoken	http://CaseStudy/SPEAK/ChandraChristianity
http://CaseStudy/RELIGION/Budhism	http://www.kmitl.ac.th/~s7010050/relationship#HadRCODE	"R002"
http://CaseStudy/RELIGION/Budhism	http://www.kmitl.ac.th/~s7010050/relationship#HadTEACHER	"Budha"
http://CaseStudy/RELIGION/Budhism	http://www.kmitl.ac.th/~s7010050/relationship#HadADHERENT	"300"
http://CaseStudy/RELIGION/Budhism	http://www.kmitl.ac.th/~s7010050/relationship#HadTEXTS	http://CaseStudy/TEXTS/Sutra
http://CaseStudy/RELIGION/Budhism	http://www.kmitl.ac.th/~s7010050/relationship#IsSpoken	http://CaseStudy/SPEAK/PeterBudhism
http://CaseStudy/RELIGION/Budhism	http://www.kmitl.ac.th/~s7010050/relationship#IsSpoken	http://CaseStudy/SPEAK/ChandraBudhism
http://CaseStudy/RELIGION/Muslim	http://www.kmitl.ac.th/~s7010050/relationship#HadRCODE	"R003"
http://CaseStudy/RELIGION/Muslim	http://www.kmitl.ac.th/~s7010050/relationship#HadTEACHER	"Mahamed"

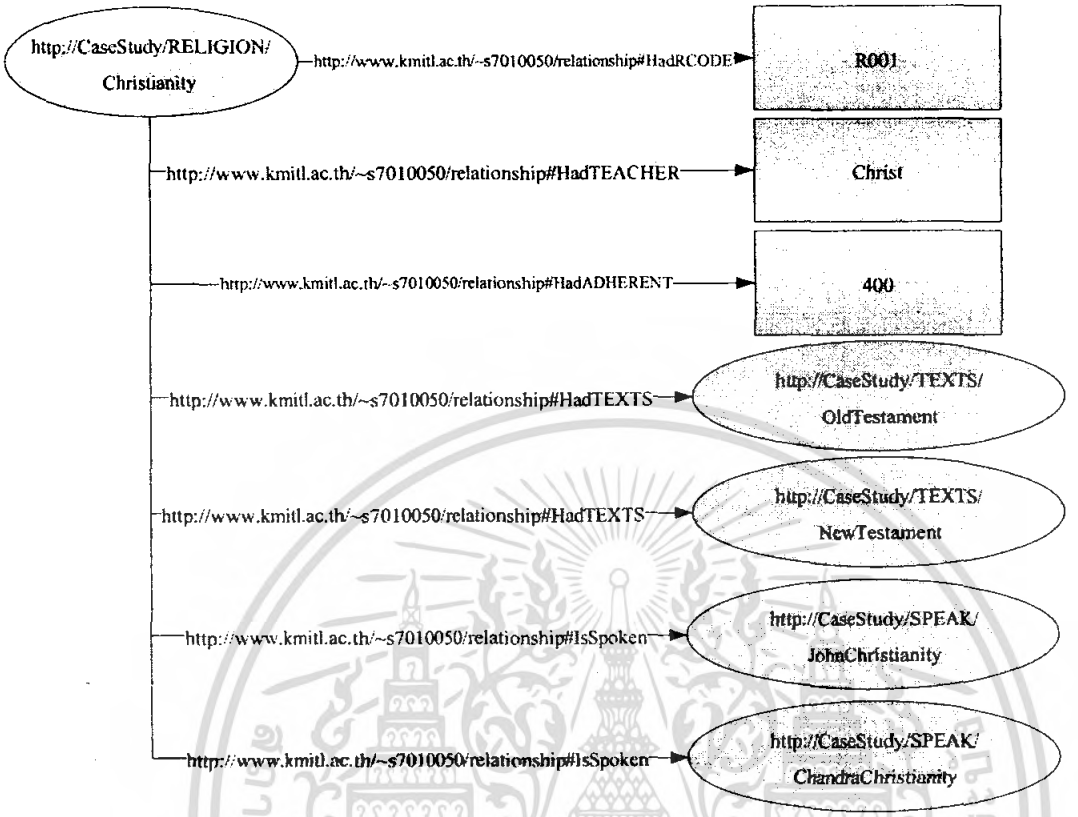
http://CaseStudy/RELIGION/Muslim	http://www.kmitl.ac.th/~s7010050/relationship#HadADHERENT	"400"
http://CaseStudy/RELIGION/Muslim	http://www.kmitl.ac.th/~s7010050/relationship#HadTEXTS	http://CaseStudy/TEXTS/Koran
http://CaseStudy/RELIGION/Hinduism	http://www.kmitl.ac.th/~s7010050/relationship#HadRCODE	"R004"
http://CaseStudy/RELIGION/Hinduism	http://www.kmitl.ac.th/~s7010050/relationship#HadTEACHER	"Khrisna"
http://CaseStudy/RELIGION/Hinduism	http://www.kmitl.ac.th/~s7010050/relationship#HadADHERENT	"350"
http://CaseStudy/RELIGION/Hinduism	http://www.kmitl.ac.th/~s7010050/relationship#HadTEXTS	http://CaseStudy/TEXTS/Upanishad
http://CaseStudy/RELIGION/Hinduism	http://www.kmitl.ac.th/~s7010050/relationship#HadTEXTS	http://CaseStudy/TEXTS/PakavatKita
http://CaseStudy/RELIGION/Hinduism	http://www.kmitl.ac.th/~s7010050/relationship#IsSpoken	http://CaseStudy/SPEAK/JohnHinduism
http://CaseStudy/TEXTS/OldTestament	http://www.kmitl.ac.th/~s7010050/relationship#HadTXCODE	"TX001"
http://CaseStudy/TEXTS/OldTestament	http://www.kmitl.ac.th/~s7010050/relationship#HadTXNAME	"Old Testaments"
http://CaseStudy/TEXTS/OldTestament	http://www.kmitl.ac.th/~s7010050/relationship#BelongTo	http://CaseStudy/RELIGION/Christianity
http://CaseStudy/TEXTS/OldTestament	http://www.kmitl.ac.th/~s7010050/relationship#IsReadBy	http://CaseStudy/SPEAKER/John
http://CaseStudy/TEXTS/OldTestament	http://www.kmitl.ac.th/~s7010050/relationship#IsReadBy	http://CaseStudy/SPEAKER/Chandra
http://CaseStudy/TEXTS/NewTestament	http://www.kmitl.ac.th/~s7010050/relationship#HadTXCODE	"TX002"
http://CaseStudy/TEXTS/NewTestament	http://www.kmitl.ac.th/~s7010050/relationship#HadTXNAME	"New Testament"
http://CaseStudy/TEXTS/NewTestament	http://www.kmitl.ac.th/~s7010050/relationship#BelongTo	http://CaseStudy/RELIGION/Christianity
http://CaseStudy/TEXTS/NewTestament	http://www.kmitl.ac.th/~s7010050/relationship#IsReadBy	http://CaseStudy/SPEAKER/Chandra

http://CaseStudy/TEXTS/Sutra	http://www.kmitl.ac.th/~s7010050/relationship#HadTXCODE	"TX003"
http://CaseStudy/TEXTS/Sutra	http://www.kmitl.ac.th/~s7010050/relationship#HadTXNAME	"Sutra"
http://CaseStudy/TEXTS/Sutra	http://www.kmitl.ac.th/~s7010050/relationship#BelongTo	http://CaseStudy/RELIGION/Budhism
http://CaseStudy/TEXTS/Sutra	http://www.kmitl.ac.th/~s7010050/relationship#IsReadBy	http://CaseStudy/SPEAKER/Chandra
http://CaseStudy/TEXTS/Koran	http://www.kmitl.ac.th/~s7010050/relationship#HadTXCODE	"TX004"
http://CaseStudy/TEXTS/Koran	http://www.kmitl.ac.th/~s7010050/relationship#HadTXNAME	"Koran"
http://CaseStudy/TEXTS/Koran	http://www.kmitl.ac.th/~s7010050/relationship#BelongTo	http://CaseStudy/RELIGION/Muslim
http://CaseStudy/TEXTS/Koran	http://www.kmitl.ac.th/~s7010050/relationship#IsReadBy	http://CaseStudy/SPEAKER/John
http://CaseStudy/TEXTS/Koran	http://www.kmitl.ac.th/~s7010050/relationship#IsReadBy	http://CaseStudy/SPEAKER/Peter
http://CaseStudy/TEXTS/Upanishad	http://www.kmitl.ac.th/~s7010050/relationship#HadTXCODE	"TX005"
http://CaseStudy/TEXTS/Upanishad	http://www.kmitl.ac.th/~s7010050/relationship#HadTXNAME	"Upanishad"
http://CaseStudy/TEXTS/Upanishad	http://www.kmitl.ac.th/~s7010050/relationship#BelongTo	http://CaseStudy/RELIGION/Hinduism
http://CaseStudy/TEXTS/PakavatKita	http://www.kmitl.ac.th/~s7010050/relationship#HadTXCODE	"TX006"
http://CaseStudy/TEXTS/PakavatKita	http://www.kmitl.ac.th/~s7010050/relationship#HadTXNAME	"Pakavat Kita"
http://CaseStudy/TEXTS/PakavatKita	http://www.kmitl.ac.th/~s7010050/relationship#BelongTo	http://CaseStudy/RELIGION/Hinduism
http://CaseStudy/TEXTS/PakavatKita	http://www.kmitl.ac.th/~s7010050/relationship#IsReadBy	http://CaseStudy/SPEAKER/John
http://CaseStudy/SPEAKER/John	http://www.kmitl.ac.th/~s7010050/relationship#HadSNAME	"John"

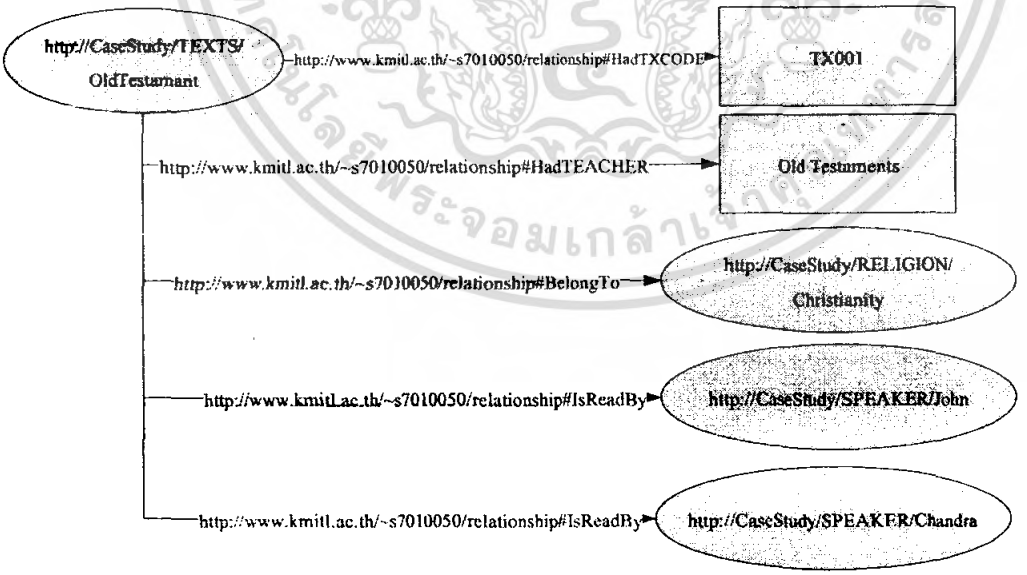
http://CaseStudy/SPEAKER/John	http://www.kmitl.ac.th/~s7010050/relationship#Read	http://CaseStudy/TEXTS/PakavatKita
http://CaseStudy/SPEAKER/John	http://www.kmitl.ac.th/~s7010050/relationship#Read	http://CaseStudy/TEXTS/Koran
http://CaseStudy/SPEAKER/John	http://www.kmitl.ac.th/~s7010050/relationship#Read	http://CaseStudy/TEXTS/OldTestament
http://CaseStudy/SPEAKER/John	http://www.kmitl.ac.th/~s7010050/relationship#Speak	http://CaseStudy/SPEAK/JohnChristianity
http://CaseStudy/SPEAKER/John	http://www.kmitl.ac.th/~s7010050/relationship#Speak	http://CaseStudy/SPEAK/JohnHinduism
http://CaseStudy/SPEAKER/Peter	http://www.kmitl.ac.th/~s7010050/relationship#HadSNAME	"Peter"
http://CaseStudy/SPEAKER/Peter	http://www.kmitl.ac.th/~s7010050/relationship#Read	http://CaseStudy/TEXTS/Koran
http://CaseStudy/SPEAKER/Peter	http://www.kmitl.ac.th/~s7010050/relationship#Speak	http://CaseStudy/SPEAK/PeterBudhism
http://CaseStudy/SPEAKER/Chandra	http://www.kmitl.ac.th/~s7010050/relationship#HadSNAME	Chandra
http://CaseStudy/SPEAKER/Chandra	http://www.kmitl.ac.th/~s7010050/relationship#Read	http://CaseStudy/TEXTS/OldTestament
http://CaseStudy/SPEAKER/Chandra	http://www.kmitl.ac.th/~s7010050/relationship#Read	http://CaseStudy/TEXTS/NewTestament
http://CaseStudy/SPEAKER/Chandra	http://www.kmitl.ac.th/~s7010050/relationship#Read	http://CaseStudy/TEXTS/Koran
http://CaseStudy/SPEAKER/Chandra	http://www.kmitl.ac.th/~s7010050/relationship#Speak	http://CaseStudy/SPEAK/ChandraBudhism
http://CaseStudy/SPEAKER/Chandra	http://www.kmitl.ac.th/~s7010050/relationship#Speak	http://CaseStudy/SPEAK/ChandraChristianity
http://CaseStudy/SPEAK/JohnChristianity	http://www.kmitl.ac.th/~s7010050/relationship#About_RELIGION	http://CaseStudy/RELIGION/Christianity
http://CaseStudy/SPEAK/JohnChristianity	http://www.kmitl.ac.th/~s7010050/relationship#By_SPEAKER	http://CaseStudy/SPEAKER/John
http://CaseStudy/SPEAK/JohnChristianity	http://www.kmitl.ac.th/~s7010050/relationship#HadNOSESSION	"2"

http://CaseStudy/SPEAK/JohnHinduism	http://www.kmitl.ac.th/~s7010050/relationship#About_RELIGION	http://CaseStudy/RELIGION/Hinduism
http://CaseStudy/SPEAK/JohnHinduism	http://www.kmitl.ac.th/~s7010050/relationship#By_SPEAKER	http://CaseStudy/SPEAKER/John
http://CaseStudy/SPEAK/JohnHinduism	http://www.kmitl.ac.th/~s7010050/relationship#HadNOSESSION	"1"
http://CaseStudy/SPEAK/PeterBudhism	http://www.kmitl.ac.th/~s7010050/relationship#About_RELIGION	http://CaseStudy/RELIGION/Budhism
http://CaseStudy/SPEAK/PeterBudhism	http://www.kmitl.ac.th/~s7010050/relationship#By_SPEAKER	http://CaseStudy/SPEAKER/Peter
http://CaseStudy/SPEAK/PeterBudhism	http://www.kmitl.ac.th/~s7010050/relationship#HadNOSESSION	"2"
http://CaseStudy/SPEAK/ChandraBudhism	http://www.kmitl.ac.th/~s7010050/relationship#About_RELIGION	http://CaseStudy/RELIGION/Budhism
http://CaseStudy/SPEAK/ChandraBudhism	http://www.kmitl.ac.th/~s7010050/relationship#By_SPEAKER	http://CaseStudy/SPEAKER/Chandra
http://CaseStudy/SPEAK/ChandraBudhism	http://www.kmitl.ac.th/~s7010050/relationship#HadNOSESSION	"3"
http://CaseStudy/SPEAK/ChandraChristianity	http://www.kmitl.ac.th/~s7010050/relationship#About_RELIGION	http://CaseStudy/RELIGION/Christianity
http://CaseStudy/SPEAK/ChandraChristianity	http://www.kmitl.ac.th/~s7010050/relationship#By_SPEAKER	http://CaseStudy/SPEAKER/Chandra
http://CaseStudy/SPEAK/ChandraChristianity	http://www.kmitl.ac.th/~s7010050/relationship#HadNOSESSION	"1"

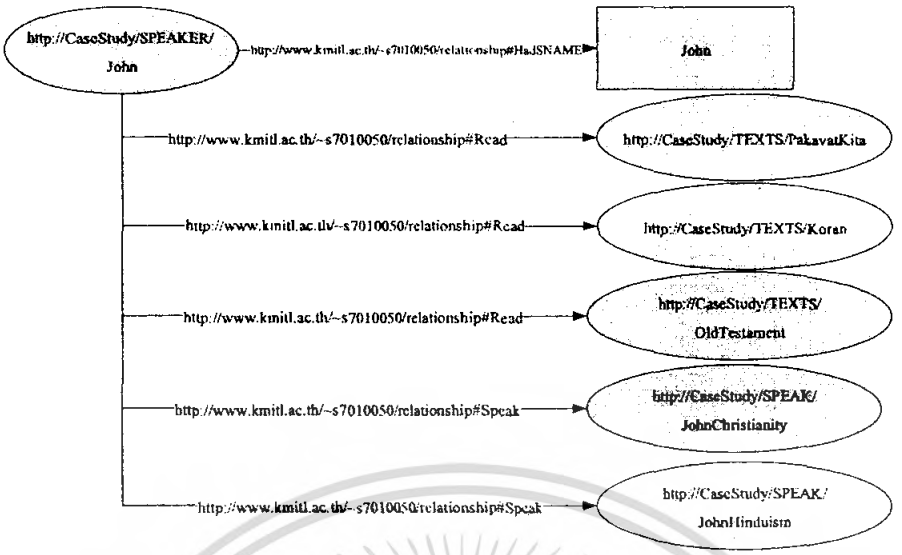
ตัวอย่าง Graph of the data model



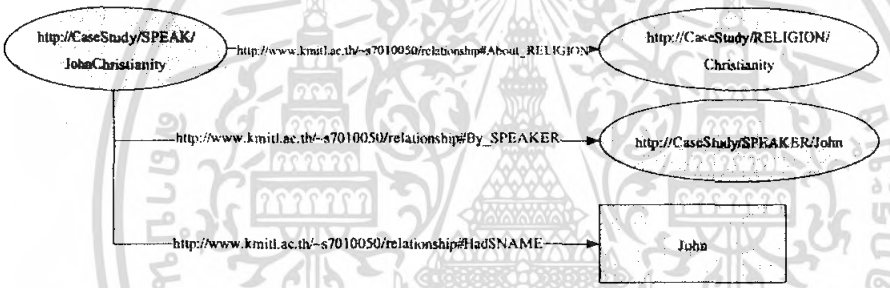
รูปที่ 4.39 Graph of the data model ของ ศาสนา Christianity



รูปที่ 4.40 Graph of the data model ของ Texts ชื่อ Old Testamant



รูปที่ 4.41 Graph of the data model ของ ผู้พูด ชื่อ John



รูปที่ 4.42 Graph of the data model ของ การพูด

4.2.2.3 ต่อมาเป็นการสร้าง RDF ตามที่ได้ออกแบบมา ในการทดลองนี้เราจะเขียนด้วยภาษา Java และใช้ API jena ในการช่วยสร้าง ไฟล์ RDF

-Model model = ModelFactory.createDefaultModel(); //เป็นการสร้าง obj ของ คลาส Model ขึ้นมาใหม่เพื่อใช่สร้าง

-RDFPropertyHadRCODE= model.createProperty("http://www.kmitl.ac.th/~s7010050/ relationship#/RELIGION/","HadRCODE"); //สร้าง Property ที่ต้องการใช้ขึ้นมา

-Resource Christianity = model.createResource(CaseStudyUri+"RELIGION/"+ "Christianity"); //สร้าง Resource ที่ต้องการใช้ขึ้นมา

-Christianity.addProperty(HadRCODE,"R001"); // เพิ่ม Property ที่เกี่ยวข้องลงใน Resource ที่ต้องการ

-model.write(new FileOutputStream("CaseStudyModel.rdf"),"RDF/XML"); // save ลงไฟล์ RDFได้

```
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:j.0="http://www.kmitl.ac.th/~s7010050/relationship/SPEAK/"
   xmlns:j.1="http://www.kmitl.ac.th/~s7010050/relationship/SPEAKER/"
   xmlns:j.2="http://www.kmitl.ac.th/~s7010050/relationship/TEXTS/"
   xmlns:j.3="http://www.kmitl.ac.th/~s7010050/relationship/RELIGION/" >
  <rdf:Description rdf:about="http://CaseStudy/TEXTS/NewTestamant">
   <j.2:IsReadBy rdf:resource="http://CaseStudy/SPEAKER/Chandra"/>
   <j.2:BelongTo rdf:resource="http://CaseStudy/RELIGION/Christianity"/>
   <j.2:HadTXNAME>New Testamant</j.2:HadTXNAME>
   <j.2:HadTXCODE>TX002</j.2:HadTXCODE>
  </rdf:Description>
  <rdf:Description
rdf:about="http://CaseStudy/SPEAK/ChandraChristianity">
   <j.0:HadNOSESSION>1</j.0:HadNOSESSION>
   <j.0:By_SPEAKER
rdf:resource="http://CaseStudy/SPEAKER/Chandra"/>
   <j.0:About_RELIGION
rdf:resource="http://CaseStudy/RELIGION/Christianity"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://CaseStudy/SPEAKER/Chandra">
   <j.1:Speak
rdf:resource="http://CaseStudy/SPEAK/ChandraChristianity"/>
   <j.1:Speak rdf:resource="http://CaseStudy/SPEAK/ChandraBudhism"/>
   <j.1:Read rdf:resource="http://CaseStudy/TEXTS/Sutra"/>
   <j.1:Read rdf:resource="http://CaseStudy/TEXTS/NewTestamant"/>
   <j.1:Read rdf:resource="http://CaseStudy/TEXTS/OldTestamant"/>
   <j.1:HadSNAME>Chandra</j.1:HadSNAME>
  </rdf:Description>
```

รูปที่ 4.43 ตัวอย่าง ไฟล์ RDF ที่ได้

4.2.2.4 สร้างไฟล์ Query ภาษา SPARQL ลง ไฟล์ เช่น cs1.rq

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIXRELIGION: <http://www.kmitl.ac.th/~s7010050/relationship/RELIGION/>
PREFIX TEXTS: <http://www.kmitl.ac.th/~s7010050/relationship/TEXTS/>
PREFIX SPEAKER: <http://www.kmitl.ac.th/~s7010050/relationship/SPEAKER/>
PREFIX SPEAK: <http://www.kmitl.ac.th/~s7010050/relationship/SPEAK/>

SELECT *
WHERE
 {
   ?resource SPEAK:HadNOSESSION ?NOSESSION .
   FILTER(xsd:integer(?NOSESSION) >= 2 )
 }
```

รูปที่ 4.44 ตัวอย่าง ไฟล์ Query ชื่อ cs1.rq

4.2.2.5 อ่านไฟล์ Query ขึ้นมาใส่ String

```
FileInputStream fin = new FileInputStream("cs1.rq");
        BufferedInputStream bin = new
BufferedInputStream(fin);
        DataInputStream din = new DataInputStream(bin);
        String s;
        while((s = din.readLine()) != null)
        {
         //System.out.println(s) ;
         queryString +=s+'\n';
        }
```

รูปที่ 4.45 รูปโค๊ดการอ่านไฟล์ Query

4.2.2.6 Query

```
Query query = QueryFactory.create(queryString) ;
QueryExecution qe = QueryExecutionFactory.create(query, model);
ResultSet results = qe.execSelect();
```

รูปที่ 4.46 รูปโค๊ดการ Query

4.2.2.7 แสดงผลลัพธ์ที่ได้

```
ResultSetFormatter.out(System.out, results, query);
```

รูปที่ 4.47 รูปโค๊ดการแสดงผลลัพธ์

```
 1 PREFIX  xsd: <http://www.w3.org/2001/XMLSchema#>
 2 PREFIX  RELIGION: <http://www.kmitl.ac.th/~s7010050/relationship/RELIGION/>
 3 PREFIX  SPEAK: <http://www.kmitl.ac.th/~s7010050/relationship/SPEAK/>
 4 PREFIX  TEXTS: <http://www.kmitl.ac.th/~s7010050/relationship/TEXTS/>
 5 PREFIX  SPEAKER: <http://www.kmitl.ac.th/~s7010050/relationship/SPEAKER/>
 6
 7 SELECT  *
 8 WHERE
 9   { ?resource  SPEAK:HadNOSESSION  ?NOSESSION .
10     FILTER ( xsd:integer(?NOSESSION) >= 2 )
11   }

Result:
-------------------------------------------------------------
| resource                                          | NOSESSION |
=============================================================
| <http://CaseStudy/SPEAK/JohnChristianity>         | "2"       |
| <http://CaseStudy/SPEAK/ChandraBudhism>           | "3"       |
| <http://CaseStudy/SPEAK/PeterBudhism>             | "2"       |
-------------------------------------------------------------
Press any key to continue...
```

รูปที่ 4.48 รูปการแสดงผลลัพธ์

4.2.2.8 นอกจากนี้ยังสามารถ Query ไฟล์ OWL ที่อยู่บน Internet ได้

4.3 OWL

การทดลองสร้าง, แลกเปลี่ยน และ Query ไฟล์ Ontology (.owl) และไฟล์ RDF โดยใช้โปรแกรม Protégé ตัวอย่าง Ontology จะเป็นเรื่องเกี่ยวกับ Pizza

4.3.1 จุดประสงค์การทดลอง

4.3.1.1 เพื่อศึกษาโครงสร้าง และ ทดลองสร้าง Ontology (.owl) ขึ้นมา

4.3.1.2 เพื่อทดลองใช้โปรแกรมตรวจจะการผิด Ontology และแก้ไขได้

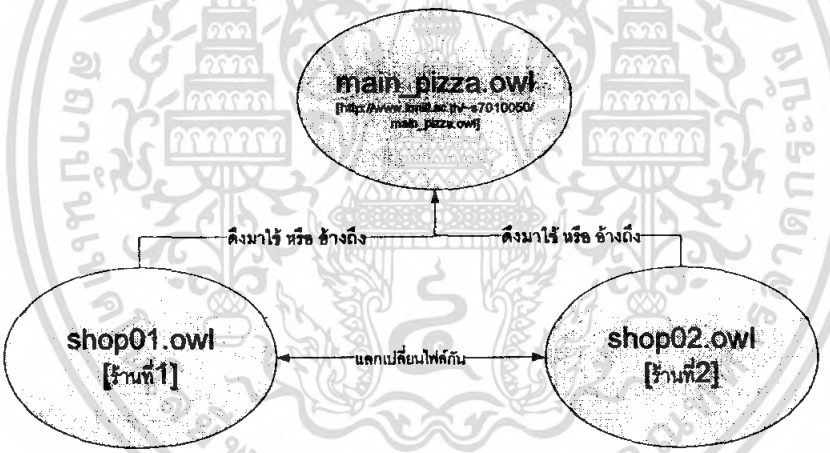
4.3.1.3 เพื่อศึกษาความสามารถของ SPARQL ว่าสามารถสืบค้นข้อมูล Ontology และ RDF ทีอยู่ในไฟล์ .owlได้หรือไม่

4.3.1.4 เพื่อทดลองว่า ถ้ามีร้านค้าสองร้าน (ร้านประเภทเดียว) ซึ่งอาจจะมี Ontology ของตัวเอง จะสามารถแลกเปลี่ยนข้อมูลกันได้อย่างไร โดยใช้ Ontology (.owl) เข้ามาช่วย

4.3.2 ขั้นตอนการทดลอง

4.3.2.1 ออกแบบ

โดยให้ทั้งสองร้านใช้ Ontology กลาง ร่วมกัน โดยที่ Ontology กลางจะแบ่งประเภท Pizza ออกเป็น ชื่อPizza และประเภทPizza ซึ่งชื่อPizza แต่ละชนิดจะประกอบไปด้วย ขอบหรือฐาน Pizza รวมกับหน้า Pizza สรุปคือ Ontology กลาง จะออกแบบ Ontology ของ Pizzaให้ระดับหนึ่ง แล้วแต่ละร้านก็ดึงหรืออ้างถึง Ontology กลาง และmap Ontology ของร้านตัวเองแล้วเพิ่มลงไปเป็น Ontology เวอร์ชันของร้านตัวเอง (shop01.owl, shop02.owl) อาจจะเพิ่ม RDF ลงไปด้วย หลังจากนั้นก็แลกเปลี่ยนไฟล์ .owl กัน แต่ละร้านก็นำมา update ถ้า update แล้วสามารถ อ่าน Ontology ด้วย โปรแกรม Protégé ก็แปลว่าร้านแต่ละร้านก็สามารถแลกเปลี่ยนข้อมูล Ontology กันได้



รูปที่ 4.49 การออกแบบไฟล์ OWL

4.3.2.2 สร้างไฟล์ Ontology หลัก

โดยตั้งชื่อว่า main_pizza.owl โดยจะมี class owl:Thing เป็นหลัก(ในowl class ทุก class ต้องสืบทอดมาจาก class owl:Thing) และมี class ลูกสอง class คือ DomainConcep (จะเก็บข้อมูลหลัก ๆเช่น Pizza), ValuePartition(จะเป็นพวกความเผ็ดหรือด้านคุณภาพ) ใน DomainConcep จะมีfoodเป็น class ลูกและมี country ที่เทียบเท่า ในfood จะมี IceCream, Pizza, PizzaBase, PizzaTopping ในPizza จะมี ชื่อPizza (NamePizza) และประเภท Pizza อื่น เช่น Pizzaเนื้อ (MeatyPizza), Pizzaผัก (VegetarianPizza) และใน NamePizza ก็จะมีชื่อPizza ต่าง เช่น American ซึ่งจะมีส่วนประกอบเป็น PizzaBase และ PizzaTopping เป็น

หลัก จากนั้นนำไฟล์ที่ได้ไปวางไว้ใน Internet

(http://www.kmitl.ac.th/~s7010050/main_pizza.owl)

รูปที่ 4.50 Class OWL

4.3.2.3 สร้าง shop01.owl

โดยให้ส่วน<owl:Ontology >ประกาศเป็นดังนี้

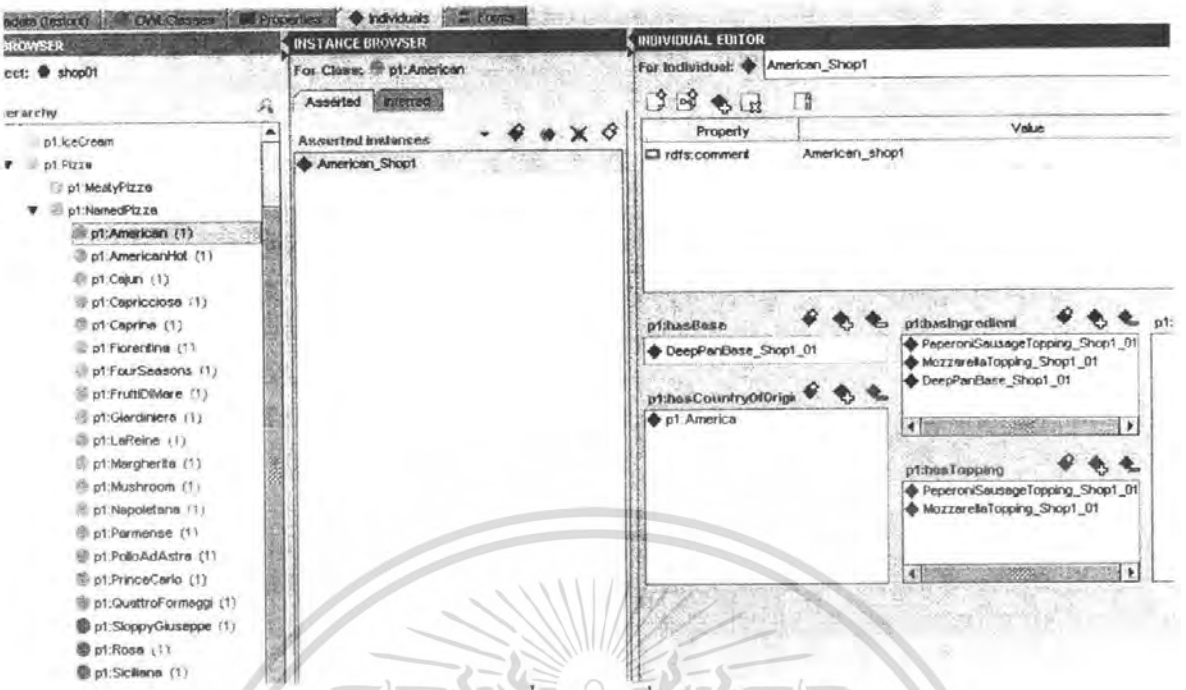
```
<owl:Ontology rdf:about="">
        <owl:imports rdf:resource="http://www.kmitl.ac.th/~s7010050/main_pizza.owl"/>
        <owl:imports rdf:resource="http://www.owl-ontologies.com/assert.owl"/>
</owl:Ontology>
```

รูปที่ 4.51 ประกาศส่วนหัวของ OWL

เพียงเท่านี้ก็สามารถดึง Ontology กลางมาใช้ได้แล้ว หลังจากนี้ก็ทำการ map Ontology ของร้านตัวเองแล้วเพิ่มลงไปเป็น Ontology เวอร์ชันของร้านตัวเอง เช่นอาจจะ เพิ่มหน้า pizza ใหม่ ๆ, ประเภท Pizza, Pizzabase, PizzaTopping ของร้านตัวเองลงไป

ตารางที่ 4.16 ตารางอธิบาย Class ที่เพิ่มเติมของร้านทั้งสองร้าน

ร้านที่ 1	ร้านที่ 2
▼ p1:NamedPizza p1:American p1:AmericanHot p1:Cajun p1:Capricciosa p1:Caprina p1:Fiorentina p1:FourSeasons p1:FruttiDiMare p1:Giardiniera p1:LaReine p1:Margherita p1:Mushroom p1:Napoletana p1:Parmense p1:PolloAdAstra p1:PrinceCarlo p1:QuattroFormaggi p1:SloppyGiuseppe p1:Rosa p1:Siciliana p1:SpicyPizza p1:VegetarianPizza p1:SpicyPizzaEquivalent p1:VegetarianPizzaEquivalent1 p1:CheeseyPizza p1:NonVegetarianPizza	▼ p1:NamedPizza p1:American p1:AmericanHot p1:Cajun p1:Capricciosa p1:Caprina p1:Fiorentina p1:FourSeasons p1:FruttiDiMare p1:Giardiniera p1:LaReine p1:Margherita p1:Mushroom p1:Napoletana p1:Parmense p1:PolloAdAstra p1:PrinceCarlo p1:QuattroFormaggi p1:Veneziana p1:Soho p1:UnclosedPizza p1:SpicyPizza p1:VegetarianPizza p1:ThinAndCrispyPizza p1:VegetarianPizzaEquivalent2 p1:InterestingPizza p1:RealItalianPizza
- เพิ่ม NamedPizza ชื่อ p1:SloppyGiuseppe, p1:Rosa,p1:Siciliana - เพิ่ม ประเภท Pizza ได้แก่ p1:SpicyPizzaEquivalent, p1:VegetarianPizzaEquivalentI, p1:CheeseyPizza, p1:NonVegetarianPizza - จะสังเกตเห็นว่า class ที่มีสีเข้ม คือที่เพิ่มเข้าไปใหม่ - ส่วนที่มีจาง คือ class ที่อยู่ใน main_pizza.owl	- เพิ่ม NamedPizza ชื่อ p1:Veneziana, p1:Soho, p1:UnclosedPizza - เพิ่ม ประเภท Pizza ได้แก่ p1:ThinAndCrispyPizza, p1:VegetarianPizzaEquivalent2, p1:InterestingPizza, p1:RealItalianPizza - จะสังเกตเห็นว่า class ที่มีสีเข้ม คือที่เพิ่มเข้าไปใ ส่วนที่มีจาง คือ class ที่อยู่ใน main_pizza.owl



รูปที่ 4.52 รูปที่ได้จากโปรแกรม

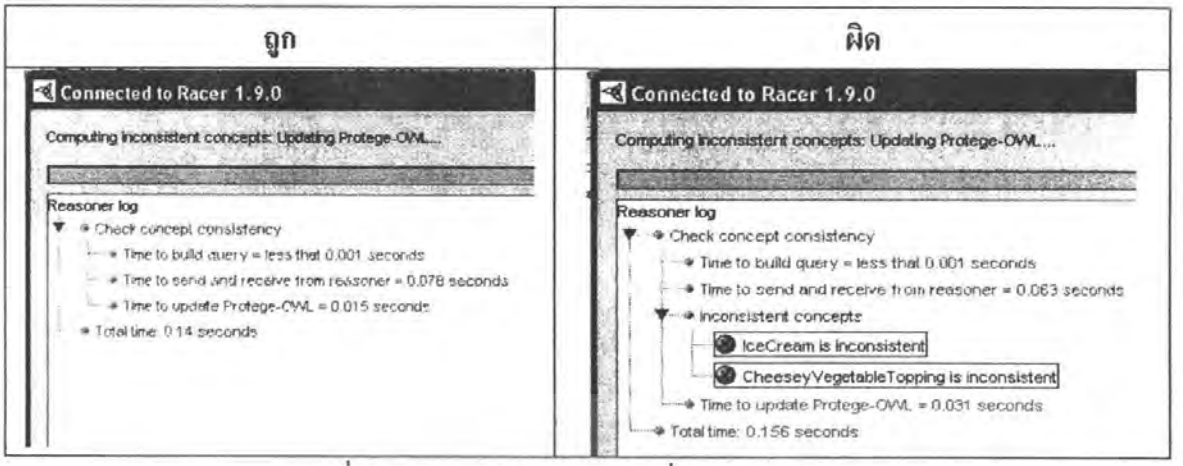
นอกจากนี้แล้วร้านที่ 1 เพิ่ม Instance (ซึ่งจะสร้าง code เป็น RDF) เข้าไปให้ p1:NamedPizza แต่ละตัว ส่วนร้านที่ 2 ก็จะคล้าย ๆ กัน

```
<p1:American rdf:ID="American_Shop1">
  <p1:hasTopping>
   <p1:PeperoniSausageTopping rdf:ID="PeperoniSausageTopping_Shop1_01">
    <p1:isToppingOf rdf:resource="#American_Shop1"/>
    <p1:hasSpiciness rdf:resource="#Medium_Shop1"/>
   </p1:PeperoniSausageTopping>
  </p1:hasTopping>
  <p1:hasTopping>
   <p1:MozzarellaTopping rdf:ID="MozzarellaTopping_Shop1_01">
    <p1:isToppingOf rdf:resource="#American_Shop1"/>
    <p1:hasSpiciness rdf:resource="#Mild_Shop1"/>
   </p1:MozzarellaTopping>
  </p1:hasTopping>
  <p1:hasBase>
   <p1:DeepPanBase rdf:ID="DeepPanBase_Shop1_01">
    <p1:isBaseOf rdf:resource="#American_Shop1"/>
   </p1:DeepPanBase>
  </p1:hasBase>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >American_shop1</rdfs:comment>
 </p1:American>
```

รูปที่ 4.53 รูปโค๊ด Instance เพิ่มมา

4.3.2.4 สามารถใช้โปรแกรม RacerPro เพื่อตรวจสอบ Ontology

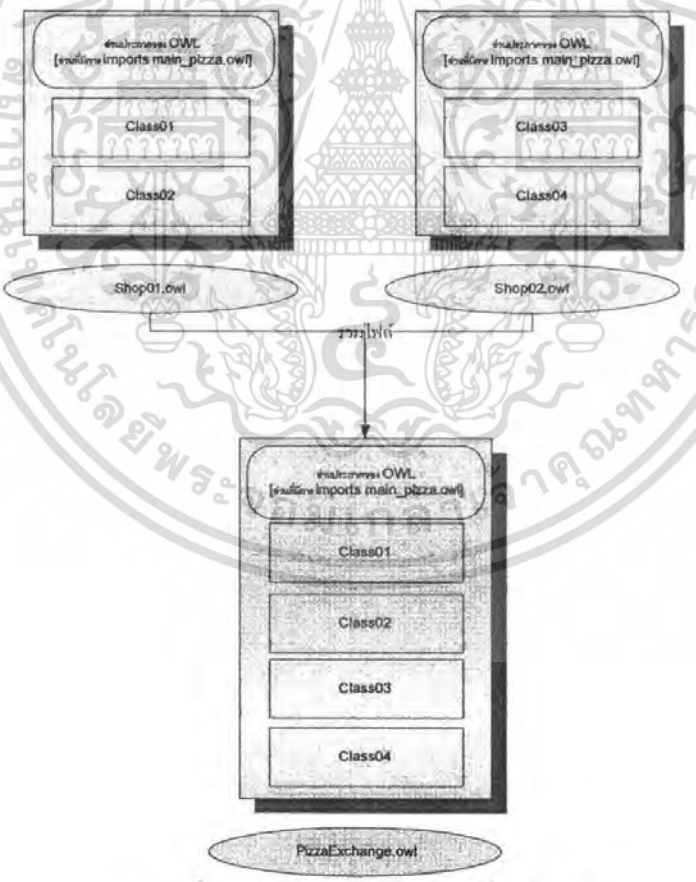
ว่าถูกต้องหรือสอดคล้อง กันหรือไหม โดยคลิกที่ หรือ เมนู owl >> Check consistency



รูปที่ 4.54 รูปการใช้ RacerPro เพื่อตรวจสอบ Ontology

4.3.2.5 แลกเปลี่ยนไฟล์ .owl กัน

แล้วเขียนโปรแกรม java นำ ไฟล์ shop01.owl, shop02.owl มารวมกัน ตามภาพต่อไปนี้



รูปที่ 4.55 รูปหลักการแลกเปลี่ยนไฟล์

4.3.2.6 ผลลัพธ์ที่ได้

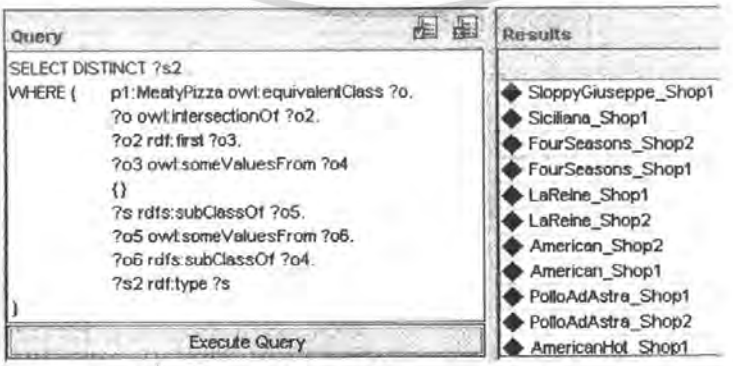
ผลลัพธ์ที่ได้ คือได้ไฟล์ PizzaExchange.owl ซึ่งเปิดด้วยโปรแกรม Protégé จะได้รูปต่อไปนี้

ตารางที่ 4.17 ตารางแสดง Class และ Instances

class	Instances
p1:Mushroom p1:Napoletana p1:Parmense p1:PolloAdAstra p1:PrinceCarlo p1:QuattroFormaggi p1:SloppyGiuseppe p1:Rosa p1:Siciliana p1:Veneziana p1:Soho p1:UnclosedPizza p1:SpicyPizza p1:VegetarianPizza p1:SpicyPizzaEquivalent p1:VegetarianPizzaEquivalent1 p1:CheeseyPizza p1:NonVegetarianPizza p1:ThinAndCrispyPizza p1:VegetarianPizzaEquivalent2 p1:InterestingPizza p1:RealItalianPizza	▼ p1:NamedPizza p1:American (2) p1:AmericanHot (2) p1:Cajun (2) p1:Capricciosa (2) p1:Caprina (2) p1:Fiorentina (2) p1:FourSeasons (2) p1:FruttiDiMare (2) p1:Giardiniera (2) p1:LaReine (2) p1:Margherita (2) p1:Mushroom (2) p1:Napoletana (2) p1:Parmense (2) p1:PolloAdAstra (2) p1:PrinceCarlo (2) p1:QuattroFormaggi (2) p1:SloppyGiuseppe (1) p1:Rosa (1) p1:Siciliana (1) p1:Veneziana (1) p1:Soho (1) p1:UnclosedPizza (1)

4.3.2.7 Query ด้วย SPARQL

สามารถ ใช้ภาษา SPARQL Query ข้อมูล Ontology และ Instances (ซึ่งเป็น RDF ที่อยู่ในไฟล์ .owl) ได้ ตามเมนู owl >> Open Sparql Query panel



รูปที่ 4.56 ตัวอย่างการใช้ SPARQL Query ข้อมูล

4.3.2.8 อ่านไฟล์ที่อยู่ใน Internet

นอกจากนี้แล้วสามารถ เขียนโปรแกรม Java ให้อ่านไฟล์ที่อยู่ใน Internet ได้โดยเพิ่ม code ส่วนต่อไปนี้

```
URL url01 = new URL("http://www.kmitl.ac.th/~s7010050/shop01.owl");
URL url02 = new URL("http://www.kmitl.ac.th/~s7010050/shop02.owl");
BufferedReader buff = new BufferedReader(new
InputStreamReader(url01.openStream()));
BufferedReader buff2 = new BufferedReader(new
InputStreamReader(url02.openStream()));
```

รูปที่ 4.57 โปรแกรม Java ให้อ่านไฟล์ที่อยู่ใน Internet

บทที่ 5
บทวิจารณ์และสรุป

5.1 บทสรุป

5.1.1 บทสรุปโดยภาพรวม

ในมุมมองของฐานข้อมูลแล้ว จะกล่าวได้ว่าเอกสาร XML นั้นก็คือฐานข้อมูล XML นั่นเอง ซึ่งสามารถ query ข้อมูลในเอกสาร XML ได้โดยใช้เทคโนโลยี XPath ซึ่งเป็นภาษาที่ใช้ค้นหาจุดเริ่มต้นของข้อมูลภายในเอกสาร XML ต่อจากนั้นจึงใช้ XQuery ที่มีไวยากรณ์ FLWOR เข้าช่วยในการค้นหาข้อมูลที่ต้องการภายในเอกสาร XML อีกทีหนึ่ง

โดยปกติแล้วเอกสาร XML ไม่จำเป็นต้องใช้การกำหนดโครงสร้างเอกสารแบบ DTD หรือ XML schema ก็สามารถใช้งานได้ ถ้าเอกสาร XML นั้นมีคุณสมบัติ well-formed อยู่แล้ว แต่ถ้ามีการกำหนดโครงสร้างเอกสารแบบ DTD หรือ XML schema เพิ่มเติม นั่นคือมีการเพิ่มให้เอกสาร XML มีคุณสมบัติ valid เพิ่มเติมแล้ว จะช่วยให้รูปแบบโครงสร้างของเอกสาร XML ของเอกสาร XML หลายๆ เอกสารที่มีข้อมูลเกี่ยวข้องกันอยู่ในรูปแบบเดียวกัน นั่นคือเป็นการกำหนดว่าเอกสาร XML มีแท็กอะไรบ้าง มีแอตทริบิวต์อะไรบ้าง ค่าของแอตทริบิวต์เป็นอะไรบ้าง เป็นต้น และยังเป็นสัญญาระหว่างองค์กรที่ใช้เอกสาร XML ในการแลกเปลี่ยนข้อมูลเพื่อให้รูปแบบของเอกสาร XML ที่ใช้มีรูปแบบที่ตรงกัน และเพื่อความถูกต้องในการนำเอกสาร XML ไปใช้งานด้วย

เนื่องจากเอกสาร XML เป็นเอกสารที่ถูกสร้างโดยอนุญาตให้ผู้เขียนเอกสารกำหนดรูปแบบ และเนื้อหาเองได้ ต่อมาจึงได้มีเทคโนโลยี RDF (Resource Description Framework) เพื่อกำหนดรูปแบบในการอธิบายเนื้อหาข้อมูลเอกสารให้อยู่ในรูปแบบเป็นมาตรฐานเดียวกัน โดยเทคโนโลยี RDF นี้ มีความสามารถในเรื่องต่างๆ 3 เรื่องที่เหนือกว่า XML คือ

1 เทคโนโลยี RDF มี semantic ในการอธิบายข้อมูลที่อยู่ภายในเอกสาร ซึ่ง XML นั้นไม่มี

2 สามารถใช้ RDFS (RDF Schema) มาเสริมเทคโนโลยี RDF ได้ ทำให้เอกสารสามารถนำเสนอความสัมพันธ์ระหว่างข้อมูลต่างๆ ที่มีความสัมพันธ์เกี่ยวข้องกันภายในเอกสาร RDF ได้

3 เทคโนโลยี RDF มี ontology ซึ่งอยู่ในรูปแบบของ OWL ในการบัญญัติศัพท์ต่างๆ ให้ผู้ใช้เอกสาร RDF ที่ใช้ ontology เดียวกันเข้าใจตรงกันได้ มีประโยชน์ในกรณีที่ถ้ามีเอกสาร RDF ที่เก็บข้อมูลประเภทเดียวกันอยู่หลายๆ เอกสาร ซึ่งเก็บอยู่ในที่ต่างๆ กัน

จะสามารถจัดการเอกสาร RDF เหล่านั้นให้อยู่ในรูปแบบ ontology เดียวกัน ต่อจากนั้นก็จะสามารถใช้ข้อมูลในเอกสาร RDF ต่างๆ ได้เข้าใจตรงกัน

เราสามารถ query ข้อมูลในเอกสาร RDF ได้โดยใช้ภาษา SPARQL ซึ่งทำได้ง่ายกว่าการ query เอกสาร XML ด้วยภาษา XPath ร่วมกับ XQuery เนื่องจากความสามารถที่เหนือกว่าของ RDF ดังกล่าว 3 ข้อนั่นเอง

5.1.2 เทคโนโลยี XML ในมุมมองของฐานข้อมูล

5.1.2.1 โครงสร้างข้อมูล (data structure)

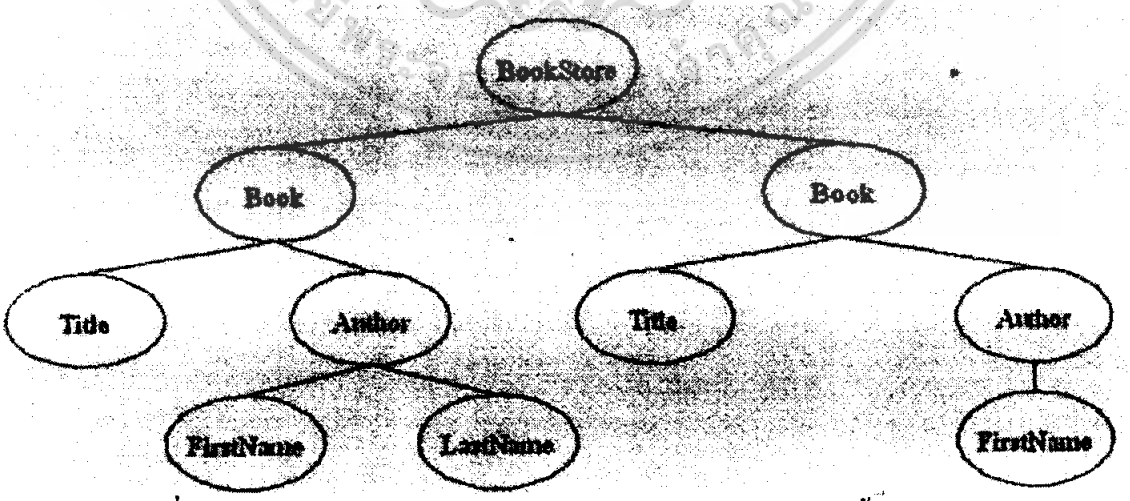
XML มีโครงสร้างข้อมูลเป็นลำดับชั้นต้นไม้ (Hierarchy) และมีความสัมพันธ์เป็นแบบพ่อลูก คือ พ่อ (parent) 1 คนมีลูก (child) ได้หลายคน แต่ลูกมีพ่อได้คนเดียว (นั่นคือเป็นความสัมพันธ์แบบ 1 to Many) หรือแบบพ่อคนเดียวมีลูก 1 คน (นั่นคือเป็นความสัมพันธ์แบบ 1 to 1) และจากความสัมพันธ์แบบ 1 to Many ของ XML ทำให้ XML สามารถรองรับการออกแบบข้อมูลที่มีลักษณะเป็น Repeating Group (ข้อมูลชนิดเดียวกันแต่มีหลายค่า) ได้อีกด้วย ดังตัวอย่างที่ 1

ถึงแม้ XML จะมีโครงสร้างเป็นลำดับชั้นต้นไม้ (Hierarchy) ซึ่งไม่มีความสัมพันธ์ของข้อมูลเป็นแบบลูกมีพ่อได้หลายคน แต่ XML ก็สามารถรองรับความสัมพันธ์แบบ Many to Many ได้เพราะมี DTD หรือ XML Schema มาช่วย เช่น ชนิดของ attribute ที่ชื่อ ID จะคล้าย Primary Key ใน relational database ส่วน IDREF และ IDREFS จะคล้ายกับ Foreign Key ใน relational database

```
<?xml version="1.0"?>
<BookStore>
    <Book category='Computer'>
        <Title>XML for Beginner</Title>
        <Author>
            <FirstName>John</FirstName>
            <LastName>Smith</LastName>
        </Author>
    </Book>
    <Book category='Math'>
        <Title>Calculus I</Title>
        <Author>
            <FirstName>James</FirstName>
        </Author>
    </Book>
</BookStore>
```

รูปที่ 5.1 ตัวอย่างเอกสาร XML sample.xml

จากรูปที่ 5.1 สามารถนำเอกสาร XML sample.xml มาแสดงในลักษณะของลำดับชั้นต้นไม้ได้ดังรูปที่ 5.2



รูปที่ 5.2 โครงสร้างเอกสาร XML sample.xml ในลักษณะลำดับชั้นต้นไม้

ตัวอย่างที่ 1 การเก็บข้อมูลของหนังสือ ซึ่งอาจจะมีผู้แต่งหลายคน และแต่ละเล่มอาจจะมีไม่เท่ากันได้ สามารถออกแบบได้ดังนี้

```
<หนังสือ>
        <ชื่อหนังสือ>Database System Concepts</ชื่อหนังสือ>
        <พิมพ์ครั้งที่>Fifth</พิมพ์ครั้งที่>
        <ผู้ประพันธ์>
            <ชื่อผู้ประพันธ์>Abraham Silberschatz</ชื่อผู้ประพันธ์>
            <ชื่อผู้ประพันธ์>Henry F. Korth</ชื่อผู้ประพันธ์>
            <ชื่อผู้ประพันธ์>S. Sudarshan</ชื่อผู้ประพันธ์>
        </ผู้ประพันธ์>
        <สำนักพิมพ์>McGRAW-HILL INTERNATIONAL EDITION</สำนักพิมพ์>
</หนังสือ>
```

5.1.2.2 ความถูกต้องของข้อมูล (data integrity)

Data integrity ของเอกสาร XML คือ คุณสมบัติ well-formed ของเอกสาร XML นั่นเอง โดยคุณสมบัติ well-formed นั้นสามารถดูได้ในบทที่ 2 รวมทั้ง element ลูกสามารถมี element พ่อได้เพียง element เดียว และ element หนึ่งๆ จะมีAttributesหรือไม่มีก็ได้ ถ้ามีสามารถมีได้มากกว่าหนึ่งAttributes

5.1.2.3 Data manipulation language

ภาษาที่ใช้ในการสืบค้นเอกสาร XML คือภาษา XQuery ซึ่งมีความสามารถดังต่อไปนี้

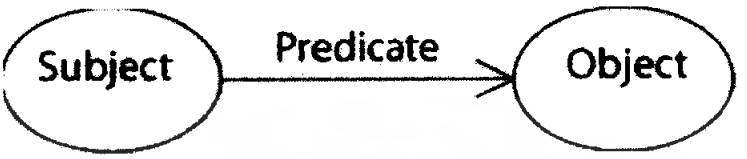
- สามารถเลือกแสดงผล Elements และAttributes ทั้งหมดหรือบางส่วนของเอกสาร XML ได้
- สามารถเลือกแสดงผล Elements และAttributes ตามเงื่อนไขที่ต้องการได้
- มี Functions ต่างๆ ให้เลือก ใช้มากมาย เช่น AVG, SUM ,MIN ,MAX ,COUNT เป็นต้น
- สามารถ คำนวณตัวเลขได้ เช่น +, -, *, div
- สามารถกรองข้อมูลเป็นกลุ่มๆ แม้จะไม่มีคำสั่ง Group By ให้เรียกใช้ก็ตาม
- สามารถนำเอกสารสองเอกสารมา Join กันได้
- สามารถทำ subquery ได้
- สามารถใช้เอกสารหนึ่งๆ ได้มากกว่า 1 copy

- สามารถทำ Correlated Subqueries (อ้างตารางนอก Sub queries) ได้
- สามารถตรวจสอบความมีอยู่จริงของ Subqueries โดยใช้ exists()

5.1.3 เทคโนโลยี RDF ในมุมมองของฐานข้อมูล

5.1.3.1 โครงสร้างข้อมูล (data structure)

RDF จะมีโครงสร้างของข้อมูลเป็น Graph ดังรูป



รูปที่ 5.3 โครงสร้างข้อมูลของ RDF

ส่วนประกอบของ RDF Graph มี 3 ส่วน คือ

1 Subject คือ หัวข้อหรือ Resource ที่ต้องการจะอธิบาย โดย Resource ที่ต้องการอธิบายจะต้องเป็น URI เท่านั้น เช่น http://www.kmitl.ac.th/ หรือจะเป็น Blank node ก็ได้

2 Predicate หรือ Property คือ คุณลักษณะของ Resource อย่างเช่น ที่อยู่ รหัสไปษณีย์

3 Object คือค่าของ Property เช่น ที่อยู่ก็จะมีค่าเป็นKMITL (Datatype แบบ Literal) เป็นต้น หรือ อาจจะเป็น Resource ที่เป็น URI ก็ได้และอาจจะเป็น Blank node ก็ได้ (Datatype แบบ Resource)

5.1.3.2 ความถูกต้องของข้อมูล (data integrity)

- Resource ต้องเป็น URI เท่านั้น
- Subject สามารถเป็นเป็น Resource หรือ Blank node ก็ได้
- Object เป็นได้สามอย่างคือ Literal, Resource และ Blank node
- Literal จะไม่มี Predicate หรือ Property
- Predicate หรือ Property จะมีทิศทางจาก Subject ไป Object เสมอ
- จะต้องมี<rdf:RDF> เป็น Root element
- Graph หนึ่งๆ ซึ่งประกอบไปด้วย Subject, Property, Object ต้องอยู่ใน <rdf:Description>
- Subject ต้องเป็นค่าของ Attribute ชื่อ rdf:about
- Property ต้องเป็น element ลูก ของ element ที่ชื่อ <rdf:Description>
- Object ที่เป็น Literal เทียบได้กับ PCDATA ที่อยู่ใน element ของ Property นั้นๆ ในเอกสาร XML เช่น <dc:title>World Wide Web Consortium</dc:title>

- Object ที่เป็น Resource คือค่าของ Attribute ชื่อ rdf:resource ที่อยู่ใน element ของ Property นั้น ๆ เช่น <rdf:li rdf:resource="ftp://ftp.example.org"/>
- Blank node จะใช้เมื่อ ใน element ของ Property มี element ลูกอีก เช่น <rdf:Alt> หรือ element ของ Property มี Attribute เป็น rdf:parseType="Collection"

The original RDF/XML document

```
1: <?xml version="1.0"?>
2:
3: <rdf:RDF
4: xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
5: xmlns:cd="http://recshop.fake/cd#">
6:
7: <rdf:Description
8: rdf:about="http://recshop.fake/cd/Beatles">
9: <cd:artist rdf:parseType="Collection">
10: <rdf:Description rdf:about="http://recshop.fake/cd/Beatles/George"/>
11: <rdf:Description rdf:about="http://recshop.fake/cd/Beatles/John"/>
12: <rdf:Description rdf:about="http://recshop.fake/cd/Beatles/Paul"/>
13: <rdf:Description rdf:about="http://recshop.fake/cd/Beatles/Ringo"/>
14: </cd:artist>
15: </rdf:Description>
16:
17: </rdf:RDF>
18:
```

Triples of the Data Model Back to Validator Input

Number	Subject	Predicate	Object
1	http://recshop.fake/cd/Beatles	http://recshop.fake/cd#artist	genid:A21470
2	genid:A21470	http://www.w3.org/1999/02/22-rdf-syntax-ns#first	http://recshop.fake/cd/Beatles/George
3	genid:A21470	http://www.w3.org/1999/02/22-rdf-syntax-ns#rest	genid:A21471
4	genid:A21471	http://www.w3.org/1999/02/22-rdf-syntax-ns#first	http://recshop.fake/cd/Beatles/John
5	genid:A21471	http://www.w3.org/1999/02/22-rdf-syntax-ns#rest	genid:A21472
6	genid:A21472	http://www.w3.org/1999/02/22-rdf-syntax-ns#first	http://recshop.fake/cd/Beatles/Paul
7	genid:A21472	http://www.w3.org/1999/02/22-rdf-syntax-ns#rest	genid:A21473
8	genid:A21473	http://www.w3.org/1999/02/22-rdf-syntax-ns#first	http://recshop.fake/cd/Beatles/Ringo
9	genid:A21473	http://www.w3.org/1999/02/22-rdf-syntax-ns#rest	http://www.w3.org/1999/02/22-rdf-syntax-ns#nil

รูปที่ 5.4 ตัวอย่าง blank node ที่เกิดจาก Collection

The original RDF/XML document

```
1: <?xml version="1.0"?>
2: <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3:          xmlns:s="http://example.org/packages/vocab#">
4:    <rdf:Description rdf:about="http://example.org/packages/X11">
5:       <s:DistributionSite>
6:          <rdf:Alt>
7:             <rdf:li rdf:resource="ftp://ftp.example.org"/>
8:             <rdf:li rdf:resource="ftp://ftp1.example.org"/>
9:             <rdf:li rdf:resource="ftp://ftp2.example.org"/>
10:         </rdf:Alt>
11:      </s:DistributionSite>
12:   </rdf:Description>
13: </rdf:RDF>
14:
15:
```

Triples of the Data Model Validator Input

Number	Subject	Predicate	Object
1	genid:A21488	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/1999/02/22-rdf-syntax-ns#Alt
2	http://example.org/packages/X11	http://example.org/packages/vocab#DistributionSite	genid:A21488
3	genid:A21488	http://www.w3.org/1999/02/22-rdf-syntax-ns#_1	ftp://ftp.example.org
4	genid:A21488	http://www.w3.org/1999/02/22-rdf-syntax-ns#_2	ftp://ftp1.example.org
5	genid:A21488	http://www.w3.org/1999/02/22-rdf-syntax-ns#_3	ftp://ftp2.example.org

รูปที่ 5.5 ตัวอย่าง blank node ที่เกิดจาก <rdf:Alt>

5.1.3.3 Data manipulation language

ภาษาที่ใช้ในการสืบค้น RDF คือภาษา SPARQL ซึ่งมีความสามารถดังต่อไปนี้

- สามารถเลือกแสดงผลโดยใช้หลักการของ RDF กราฟ คือ Subject, Object, Predicate
- สามารถเลือกแสดงผล ได้ตามเงื่อนไขที่ต้องการได้ เพราะมี FILTER() ในการกรอง
- สามารถ คำนวณตัวเลขได้ เช่น +, -, *, / หรือ เปรียบเทียบ (>,<,=,!=) หรือ Logical (||,&&) ได้ตาม ชนิดข้อมูลที่ประกาศใน XML Schema
- สามารถกรองข้อมูลเป็นกลุ่มๆ โดยใช้หลักการของ Empty Group Pattern และ Projection ช่วยได้
- สามารถนำเอกสารสองเอกสารมา Join กันได้ โดยการใช้ Named, Empty Group Pattern, Projection
- ไม่สามารถทำ Subqueries ได้
- สามารถใช้เอกสารหนึ่งๆ ได้มากกว่า 1 copy โดย Empty Group Pattern ช่วย
- ไม่สามารถทำ Correlated Subqueries (อ้างตารางนอก Subqueries) ได้ เพราะไม่มี Subqueries

- สามารถ bound ตรวจสอบได้ว่ามีตามเงื่อนไขใหม่ จะคล้ายๆ exists() ใน XQuery
- มี ORDER BY ที่ใช้ในการเรียงผลลัพธ์
- มี Constructor Functions ต่างๆ ให้
- มี offset และ limit ในการกำหนดจำนวนการแสดงผลได้
- มีรูปแบบการแสดง หลายอย่าง เช่น SELECT, CONSTRUCT, ASK, DESCRIBE
- มีฟังก์ชันในการตรวจว่าเป็นค่าอะไรบ้างเช่น isIRI, isBlank, isLiteral, lang เป็นต้น

5.1.4 ความสามารถของเอกสาร RDF ที่เหนือกว่าเอกสาร XML

เอกสาร RDF มีความสามารถที่เหนือกว่าเอกสาร XML อยู่ 3 ประเด็น ดังนี้

1 เทคโนโลยี RDF มี semantic ในการอธิบายข้อมูลที่อยู่ถายในเอกสาร ซึ่ง XML นั้นไม่มี

2 สามารถใช้ RDFS (RDF Schema) มาเสริมเทคโนโลยี RDF ได้ ทำให้เอกสารสามารถนำเสนอความสัมพันธ์ระหว่างข้อมูลต่างๆ ที่มีความสัมพันธ์เกี่ยวข้องกันภายในเอกสาร RDF ได้

3 เทคโนโลยี RDF มี ontology ซึ่งอยู่ในรูปแบบของ OWL ในการบัญญัติศัพท์ต่างๆ ให้ผู้ใช้เอกสาร RDF ที่ใช้ ontology เดียวกันเข้าใจตรงกันได้ มีประโยชน์ในกรณีที่ ถ้ามีเอกสาร RDF ที่เก็บข้อมูลประเภทเดียวกันอยู่หลายๆ เอกสาร ซึ่งเก็บอยู่ในที่ต่างๆ กัน จะสามารถจัดการเอกสาร RDF เหล่านั้นให้อยู่ในรูปแบบ ontology เดียวกัน ต่อจากนั้นก็จะสามารถใช้ข้อมูลในเอกสาร RDF ต่างๆ ได้เข้าใจตรงกัน

5.1.5 ความสามารถของภาษา SPARQL และ XQuery

- SPARQLมี productivity มากกว่า XQuery ในการ Query เอกสาร RDF เพราะ SPARQL มีความง่ายในการใช้งานมากกว่า XQuery เนื่องจากสามารถ query เอกสาร RDF ได้โดยใช้ syntax สั้นๆ เพียงไม่กี่บรรทัดก็จะได้ผลลัพธ์เหมือนกับการใช้ XQuery ในการ query เอกสาร XML ซึ่งมีความยาว และยุ่งกว่ามาก แต่SPARQL มีข้อจำกัดที่ คือ ไม่สามารถทำ subquery ได้ จึงจะต้องใช้ XQuery มาช่วยโดยการทำ XQuery มา query เอกสาร RDF แทนการใช้ SPARQL เนื่องจากเอกสาร RDF ก็เขียนโดยใช้ syntax XML เหมือนกัน
- การ query โดยใช้ SPARQL นั้น คนที่ทำการ query แค่รู้ว่าต้องการ query อะไรเท่านั้นก็สามารถ query ข้อมูลได้เลย ซึ่งต่างจาก XQuery ที่จำเป็นต้องรู้โครงสร้างข้อมูลก่อน
- SPARQL สามารถ Query ข้อมูลที่มี Ontology ได้ ทำให้การ query โดยใช้ SPARQL เป็นภาษาธรรมชาติ นั่นคือใกล้เคียงกับภาษามนุษย์มากกว่าการใช้ XQuery

- SPARQL สามารถถามถึงคุณสมบัติที่เกี่ยวกับ RDF เช่น isIRI, isBlank, isLiteral, lang เป็นต้นได้ ในขณะที่ XQueryไม่มี คุณสมบัติ เหล่านี้
- SPARQL คำนวณหรือเปรียบเทียบชนิดข้อมูลที่ไม่ใช่ primitive type ได้ เช่น ข้อมูลชนิดวันที่ ในขณะที่ XQuery ทำไม่ได้
- มีการแบ่งส่วนการ Query โครงสร้างข้อมูลและส่วนข้อมูลอย่างชัดเจน
  - ?x ns:price ?price . -> Query โครงสร้างข้อมูล
  - FILTER (?price < 30.5) -> Query ข้อมูล
  - แต่ XQuery จะใช้ Xpath ในการ Query โครงสร้างข้อมูล และ Query ข้อมูลบางส่วนเท่าที่ Xpath สามารถทำได้
- การ query โดยใช้ SPARQL นั้นมีความง่ายกว่าการใช้ XQuery เนื่องจากผู้ใช้ไม่จำเป็นต้องมีประสบการณ์ในการ query มาก่อน และไม่จำเป็นต้องรู้ไวยากรณ์ระดับยาก ก็สามารถ query ข้อมูลได้
- SPARQL เหมาะกับการออกรายงาน เพราะมีรูปแบบการแสดงผลที่หลายอย่าง เช่น SELECT, CONSTRUCT, ASK, DESCRIBE ส่วน XQuery มีรูปแบบการแสดงผลเพียงอย่างเดียวคือ return

5.1.6 เปรียบเทียบความสามารถของภาษา SQL และภาษา XQuery

จากการทดลอง query เอกสาร XML ด้วยภาษา XQuery ได้ข้อสรุป ดังนี้

1 ภาษา XQuery มีความสามารถอย่างน้อยเท่ากับภาษา SQL เพราะ XQuery สามารถ query ข้อมูลได้อย่างที่ SQL ทำได้

2 ภาษา XQuery มีคุณสมบัติเป็น Relational Complete เพราะการที่ภาษาใดจะมีคุณสมบัติเป็น Relational Complete ต้องมีความสามารถอย่างน้อยเทียบเท่า Relational Algebra หรือ Relational Calculus เนื่องจากภาษา SQL มีคุณสมบัติเป็น Relational Complete ดังนั้นเมื่อภาษา XQuery มีความสามารถเทียบเท่ากับภาษา SQL ดังกล่าว ก็แปลว่าภาษา XQuery มีความสามารถอย่างน้อยเทียบเท่า Relational Algebra หรือ Relational Calculus ด้วย ทำให้ภาษา XQuery มีคุณสมบัติเป็น Relational Complete ตามไปด้วย

3 ภาษา XQuery สามารถ query ข้อมูลที่มีโครงสร้างเป็นลำดับชั้น (hierarchy) ได้ ซึ่งภาษา SQL ไม่สามารถทำได้

5.2 วิจารณ์สิ่งที่ได้จากโครงงาน

5.2.1 รู้จัก XML และ เทคโนโลยีของ XML เช่น XSL, DTD, XML Schema, DOM, XPath และ XQuery เป็นต้น

5.2.2 รู้จักเทคโนโลยี RDF, RDFS, OWL และ SPARQL

5.2.3 สามารถ query ข้อมูลในเอกสาร XML ด้วย XQuery และ XPath ได้

5.2.4 สามารถ query ข้อมูลในเอกสาร RDF ด้วย SPARQL ได้โดยใช้ Jena API

5.2.5 สามารถ query ข้อมูลใน OWL ด้วย SPARQL

5.2.6 สามารถเปรียบเทียบความแตกต่างระหว่าง SQL และ XQuery ได้

5.2.7 สามารถสร้าง แลกเปลี่ยน และ query ไฟล์ OWL และไฟล์ RDF ได้โดยใช้โปรแกรม Protege

5.2.8 รู้จักคุณประโยชน์ของเทคโนโลยีของ XML ในปัจจุบัน

5.2.9 รู้จักการเก็บข้อมูลประเภท XML และเทคโนโลยีที่เกี่ยวข้องกับ XML ใน DBMS ต่างๆ

5.3 ปัญหาอุปสรรคและแนวทางแก้ไข

5.3.1 XML มีเทคโนโลยีที่เกี่ยวข้องมากมายจึงจำเป็นต้องใช้เวลาในการศึกษา สามารถแก้ไขได้โดยให้เวลากับการทำงานนี้อย่างมาก

5.3.2 โปรแกรมที่ใช้ในการแปล XQuery ไปเป็นผลลัพธ์ ไม่มี debugger มาให้

5.3.3 XQuery เป็นเทคโนโลยีที่กำลังอยู่ในการพัฒนาอาจมีการเปลี่ยนแปลงได้

5.3.4 เอกสารศึกษาส่วนเป็นภาษาอังกฤษ และยากต่อการทำความเข้าใจ สามารถแก้ไขได้โดยให้เวลากับการทำงานนี้อย่างมาก และพยายามพูดคุยกันเพื่อความเข้าใจที่ตรงกัน

5.3.5 การ query เอกสารข้าม Internet บางครั้งไม่สำเร็จหรือมีปัญหา สามารถแก้ไขได้โดยศึกษาจากแหล่งข้อมูลอื่นเพิ่มเติม

5.4 แนวทางการพัฒนาต่อ

5.4.1 ศึกษาและทดลองใช้ agent เป็นตัวแทนในการไปเยี่ยมชมเวปในอินเตอร์เน็ตที่ใช้ XML หรือ RDF และ query ข้อมูลที่ต้องการออกมา

5.4.2 ศึกษาการ create, insert, update และ delete ข้อมูลในเอกสาร XML และ RDF

5.4.3 สร้างโปรแกรมที่เก็บข้อมูล ความสัมพันธ์ระหว่างข้อมูล และ ontology จากนั้นสามารถทำการ create, insert, update และ delete ได้

บรรณานุกรม

ขนิษฐสรณ์ เวียรศิลป์. 2546. "การพัฒนาโปรแกรมสำหรับการค้นหาข้อมูลภายในเอกสาร XML ด้วยไวยากรณ์ Xquery ผ่านเว็บ." วิทยานิพนธ์วิทยาศาสตรบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง.

เดวิด ฮันเตอร์. 2545. **คัมภีร์การใช้ XML ฉบับสมบูรณ์.** แปลโดย สุวัฒนา สุขสมจินตน์. กรุงเทพฯ : สำนักพิมพ์ซีเอ็ดยูเคชั่น.

ปรีชาพล เหมะสุดนธ์. 2549. "การพัฒนาโปรแกรมสำหรับการค้นหาข้อมูลภายในเอกสารเอ็กซ์เอ็มแอลด้วยเอ็กซ์คิวรีบนด็อตเน็ตเฟรมเวิร์ก." วิทยานิพนธ์วิทยาศาสตรบัณฑิต สาขาเทคโนโลยีสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง.

ศุภชัย สมพานิช. 2544. **เข้าใจและใช้งานภาษา XML ฉบับโปรแกรมเมอร์.** นนทบุรี : สำนักพิมพ์ อินโฟเพรส.

สำนักวิทยบริหารมหาวิทยาลัยสงขลานครินทร์. 2550. **XML มาตรฐานใหม่ Web Page.** [Online]. Available : http://oas.psu.ac.th/techno/article/xml.pdf.

สุพิทย์ กาญจนพันธุ์. 2550. **WEB 3.0.** [Online]. Available : http://61.19.158.36/?page_id=15.

202.28.94.51. 2551. **ความหมายบนฐานของ RDF.** [Online]. Available : http://202.28.94.51/users/web/Nappakun/Thesis07-08-05/19-06-05/%E0%B8%81%E0%B8%B2%E0%B8%A3%E0%B8%88%E0%B8%B1%E0%B8%94%E0%B9%80%E0%B8%81%E0%B9%87%E0%B8%9A%E0%B9%81%E0%B8%A5%E0%B8%B0%E0%B8%81%E0%B8%B2%E0%B8%A3%E0%B8%AA%E0%B8%B7%E0%B8%9A%E0%B8%84%E0%B9%89%E0%B8%99%E0%B8%A0%E0%B8%B2%E0%B8%9E%E0%B9%80%E0%B8%8A%E0%B8%B4%E0%B8%87%E0%B8%84%E0%B8%A7%E0%B8%B2%E0%B8%A1%E0%B8%AB%E0%B8%A1%E0%B8%B2%E0%B8%A2%E0%B8%9A%E0%B8%99%E0%B8%90%E0%B8%B2%E0%B8%99%E0%B8%82%E0%B8%AD%E0%B8%87%20RDF_5.doc.

202.28.94.50. 2551. **Semantic Search for Learning Object.** [Online]. Available : 202.28.94.50/Lab/siil/resource/Semantic%20Search%20for%20Learning%20Object.doc.

Abraham Silberschatz, Henry F. Korth and S. Sudarshan. 2549. **Database System Concepts.** New York : McGrawHILL.

Burapha University. 2550. **An Overview of XML.** [Online].

Available : http://staff.buu.ac.th/~seree/xml/310418_1_XML.ppt.

Burapha University. 2550. **cs473581.** [Online].

Available : http://cvs.buu.ac.th/~cs473581/knowledge-base/47033581.doc.

Burapha University. 2550. **viewvc.** [Online]. Available : http://cvs.buu.ac.th/cgi-bin/viewvc.cgi/f50353/cs473206/47033206.odt?view=co

Cmsmartsoft. 2550. **XML คืออะไร.** [Online].

Available : http://202.28.249.241/~mitm2003/phpBB2/download.php?id=1107 &sid=afa04059de97f56ab695f55623c52f75.

Co-ode. 2550. **Pizza Ontologyv1.5.** [Online].

Available : http://www.co-ode.org/ontologies/pizza/2007/02/12//.

Codenotes. 2544. DTD **Attribute Types.** [Online].

Available : http://www.codenotes.com/articles/articleAction.aspx?articleID=159.

Cs.man.ac.uk. 2551. **ISWC2003.** [Online].

Available : www.cs.man.ac.uk/~horrocks/ISWC2003/Tutorial/examples.pdf.

Dajobe. 2551. **Introduction to RDF Query with SPARQL.** [Online].

Available : http://www.dajobe.org/talks/200603-sparql-stanford/.

DSstar. 2550. **XML's REALITY CHECK: DATABASE MANAGEMENT.** [Online].

Available : http://www.taborcommunications.com/dsstar/02/0212/103910.html.

Exzilla. 2550. **Howto: SYS.XMLType.** [Online].

Available : http://www.exzilla.net/docs/xmltype/HOWTO-xmlType.php.

Frank van Harmelen. 2546. **Frank van Harmelen Knowledge Representation and Reasoning Group.** [Online].

Available : http://www.cs.vu.nl/~frankh/postscript/OntoHandbook03OWL.pdf.

Gavin Powell. 2550. **Beginning XML databases.** Indiana : Wiley Publishing.

Ict. 2551. **DTD and XML Schema.** [Online]. Available : http://www.ict.pyo.nu.ac.th/sathienh/DTD%20and%20XML%20Schema.pdf.

Ilrt. 2551. **Ontologies-SIMILE.** [Online].

Available : http://www.ilrt.org/discovery/2000/11/lux/.

Ilrt. 2551. **RDF:Ontologies and Metadata.** [Online].

Available : http://www.ilrt.org/discovery/2000/11/lux/.

Information and Communication Technology Center, Ministry of Public Health. 2550. **ความเป็นมาของ XML.** [Online]. Available : http://ict.moph.go.th/news/xml.php.

Jena. 2551. **Jena – A Semantic Web Framework for Java.** [Online]. Available : http://jena.sourceforge.net/.

Jena. 2551. **SPARQL Tutorial.** [Online]. Available : http://jena.sourceforge.net/ARQ/Tutorial/.

Jim Melton and Stephen Buxton. 2549. **Querying XML : XQuery, XPath, and SQL/XML in Context.** Amsterdam : Elsevier.

Kasetsart University. 2550. **xml เพื่อสร้างเอกสารบนอินเทอร์เน็ต.** [Online]. Available : http://www.ku.ac.th/magazine_online/xml.html.

Kevin Williams. 2000. **Professional XML Databases.** Birmingham : Wrox Press.

Kku. 2551. **DTD.** [Online]. Available : http://gear.kku.ac.th/~krunapon/courses/178375/slides/dtd.pdf.

Kmitnb. 2551. **project_old.** [Online]. Available : sailom4.cs.kmitnb.ac.th/project/project_old/project247/present/folder.2004-11-29.1299376961/ONTOLOGY.DOC.

Librdf. 2551. **Redland Rasqal RDF Query Demonstration.** [Online]. Available : http://librdf.org/querys.

Nappakun. 2548. **ตัวแบบการอธิบายข้อมูลรูปภาพสำหรับการ.** [Online]. Available : 202.28.94.51/users/web/Nappakun/Thesis07-08-05/ตัวแบบการอธิบายข้อมูลรูปภาพสำหรับการ09-08-05.ppt.

NECTEC*PEDIA. 2550. **Extension Markup Language (ความรู้ที่ได้รับจากการบรรยาย).** [Online]. Available : http://wiki.nectec.or.th/setec/Pub/Report_XML_by_warm.

Ngamnij. 2550. **Workshop.** [Online]. Available : http://202.28.94.51/users/ngamnij/322735/Resource/Workshop-2550.pdf.

Pantip. 2549. **ความรู้ทั่วไปเกียวกับ XML.** [Online]. Available : http://www.pantip.com/tech/developer/topic/DX2185555/DX2185555.html.

Sripatum-University. 2551. **alumni.** [Online]. Available : http://alumni.spu.ac.th/mallika/msit9/member/New%20Folder/New%20Folder%20(2)/Metadata.pdf.

Thaixml. 2550. **essentials.** [Online]. Available : http://www.thaixml.com/essentials.

Tla.or.th. 2551. **conferance2549.** [Online]. Available : tla.or.th/conferance2549.doc.

Univ-lyon1. 2550. **RDF Tutorial.** [Online].

Available : http://bat710.univ-lyon1.fr/~champin/rdf-tutorial/rdf-tutorial.html.

W3C. 2457. **OWL Web Ontology Language Guide.** [Online].

Available : http://www.w3.org/TR/2004/REC-owl-guide-20040210/.

W3C. 2457. **OWL Web Ontology Language Overview.** [Online].

Available : http://www.w3.org/TR/2004/REC-owl-features-20040210/.

W3C. 2551. **SPARQL Query Language for RDF.** [Online].

Available : http://www.w3.org/TR/rdf-sparql-query/#sparqlSyntax.

W3C. 2550. **W3C XML Query (XQuery).** [Online].

Available : http://www.w3.org/XML/Query.

W3schools. 2550. **RDF Tutorial.** [Online].

Available : http://www.w3schools.com/rdf/default.asp.

W3schools. 2550. **Semantic Web Tutorial.** [Online].

Available : http://alumni.spu.ac.th/mallika/msit9/member/New%20Folder/New %20Folder%20(2) /Metadata.pdf.

W3schools. 2550. **Semantic Web Tutorial.** [Online].

Available : http://www.w3schools.com/semweb/default.asp.

W3schools. 2550. **XML Tutorial.** [Online]. Available : http://www.w3schools.com.

Wikipedia. 2551. **เอกซ์เอ็มแอล.** [Online]. Available : http://th.wikipedia.org/wiki/XML.

Wikipedia. 2551. **SPARQL.** [Online]. Available : http://en.wikipedia.org/wiki/SPARQL.

Wikipedia. 2551. **Web Ontology Language.** [Online].

Available : http://en.wikipedia.org/wiki/Web_Ontology_Language.

Wikipedia. 2550. **XML.** [Online]. Available : http://en.wikipedia.org/wiki/XML.

Xml.com. 2551. **introducing-sparql-querying-semantic-web-tutorial.** [Online].

Available : http://www.xml.com/pub/a/2005/11/16/introducing-sparql-querying-semantic-web-tutorial.html?page=1.

Xulplanet. 2551. **rdfquery.** [Online].

Available : http://www.xulplanet.com/tutorials/mozsdk/rdfquery.php.