XTI: เทคนิคการทำดัชนีแบบเข้ารหัสบิตแมปสำหรับการสอบถามข้อมูลในเอกสาร XML

XTI: A Bitmap Encoding Indexing Technique for Querying XML Documents

วรารัตน์ จักรหวัด

Wararat Jakawat

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญา
วิทยาศาสตรมหาบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์
มหาวิทยาลัยสงขลานครินทร์

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science in Computer Science
Prince of Songkla University
2553

ชื่อวิทยานิพนธ์ XTI: เทคนิคการทำดัชนีแบบเข้ารหัสบิตแมปสำหรับการสอบถามข้อมูลในเอกสาร XML

ผู้เขียน นางสาววรารัตน์ จักรหวัด

สาขาวิชา วิทยาการคอมพิวเตอร์

อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	**คณะกรรมการสอบ**
…….....	……..ประธานกรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.ศิริรัตน์ วณิชโยบล)	(ผู้ช่วยศาสตราจารย์ ดร.สุขุมาล กิติสิน)
	……..กรรมการ
	(ผู้ช่วยศาสตราจารย์ ดร.ศิริรัตน์ วณิชโยบล)
	……..กรรมการ
	(ดร.ลัดดา ปรีชาวีรกุล)
	……..กรรมการ
	(ผู้ช่วยศาสตราจารย์ ดร.ภัทร อัยรักษ์)

บัณฑิตวิทยาลัย มหาวิทยาลัยสงขลานครินทร์ อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษา ตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์

……..

(รองศาสตราจารย์ ดร.เกริกชัย ทองหนู)

คณบดีบัณฑิตวิทยาลัย

ชื่อวิทยานิพนธ์	XTI: เทคนิคการทำดัชนีแบบเข้ารหัสบิตแมปสำหรับการสอบถามข้อมูลในเอกสาร XML
ผู้เขียน	นางสาววรารัตน์ จักรหวัด
สาขาวิชา	วิทยาการคอมพิวเตอร์
ปีการศึกษา	2552

บทคัดย่อ

ในปัจจุบันนี้ การนำเสนอข้อมูลในระบบอินเตอร์เน็ต ส่วนใหญ่จะอยู่ในรูปแบบของเอกสาร XML เพราะ XML เป็นมาตรฐานที่ใช้กันแพร่หลายในการแลกเปลี่ยนข้อมูลในระบบอินเตอร์เน็ต เพื่อที่จะเพิ่มประสิทธิภาพในการค้นหาข้อมูล วิทยานิพนธ์นี้ได้นำเสนอเทคนิคการทำดัชนี สำหรับการค้นหาข้อมูลในเอกสาร XML โดยจะทำการลงรหัสให้กับแต่ละอิลิเมนต์และแอททริบิวต์ ซึ่งได้นำ แนวคิดของ Huffman Coding มาประยุกต์ใช้สำหรับสร้างรหัสซึ่งจะช่วยลดเวลาในการประมวลผลการสอบถามและลดพื้นที่สำหรับจัดเก็บดัชนีและใช้ตำแหน่งเริ่มต้นและ สิ้นสุดในการเข้าถึงข้อมูลในเอกสาร XML โดยเรียกเทคนิคนี้ว่า XTI (XML Tree Index) จากผลการศึกษาแสดงให้เห็นว่าดัชนี XTI ช่วยลดพื้นที่สำหรับจัดเก็บดัชนีและเวลาในการประมวลผลการสอบถามได้

Thesis Title	XTI: An Bitmap Encoding Indexing Technique for Querying XML Documents
Author	Miss Wararat Jakawat
Major Program	Computer Science
Academic Year	2009

ABSTRACT

Recently, data on the internet is in the form XML Documents because XML is a standard for Internet data exchange. We proposed a new efficient indexing technique to search data in XML Documents. We apply Huffman Coding to encode each element and attribute, leading to use less space and save query processing time. Moreover, we use start_offset and end_offset to access data in XML Documents. The new indexing technique is called XTI (XML Tree Index). The study results confirm that our technique can reduce space and save query processing time.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จได้ด้วยความช่วยเหลือและสนับสนุนจากบุคคลหลายฝ่าย ผู้วิจัยรู้สึกซาบซึ้งและขอกราบขอบพระคุณอย่างสูง คือ

ผู้ช่วยศาสตราจารย์ ดร.ศิริรัตน์ วณิชโยบล อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่กรุณาให้คำปรึกษาแนะนำ และช่วยเหลือในการแก้ปัญหาต่าง ๆ ให้แก่ผู้วิจัยเสมอมา พร้อมทั้งตรวจทานและแก้ไขวิทยานิพนธ์ให้แก่ผู้วิจัย

ผู้ช่วยศาสตราจารย์ ดร.สุขุมาล กิติสิน ประธานกรรมการสอบวิทยานิพนธ์ ที่กรุณาช่วยตรวจทานและแก้ไขวิทยานิพนธ์ให้มีความสมบูรณ์

ดร.ลัดดา ปรีชาวีรกุล กรรมการในการสอบวิทยานิพนธ์ที่กรุณาให้ข้อเสนอแนะในการทำวิจัย รวมทั้งตรวจทานแก้ไขวิทยานิพนธ์

ผู้ช่วยศาสตราจารย์ ดร.ภัทร อัยรักษ์ กรรมการในการสอบวิทยานิพนธ์ ที่กรุณาตรวจทานแก้ไขวิทยานิพนธ์

อาจารย์ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยสงขลานครินทร์ทุกท่านที่ให้ความรู้ทางด้านวิชาการ ซึ่งสามารถนำมาใช้ในการทำวิทยานิพนธ์ได้อย่างดียิ่ง

เจ้าหน้าที่ภาควิชาวิทยาการคอมพิวเตอร์ และเจ้าหน้าที่บัณฑิตวิทยาลัยทุกท่านที่ให้ความช่วยเหลือ และอำนวยความสะดวกเกี่ยวกับเอกสารต่าง ๆ

เพื่อน ๆ พี่ ๆ และน้อง ๆ ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ที่ให้คำปรึกษา และช่วยเหลือในการทำวิทยานิพนธ์

คุณพ่อ คุณแม่ และน้องชาย ที่ให้การสนับสนุนคอยเป็นห่วงสุขภาพและให้กำลังใจแก่ผู้วิจัยมาโดยตลอด

ผู้วิจัยขอขอบคุณทุกท่านเป็นอย่างสูงมา ณ โอกาสนี้

วรารัตน์ จักรหวัด

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

รายการตาราง

รายการภาพประกอบ

รายการภาพประกอบ (ต่อ)

รายการภาพประกอบ (ต่อ)

บทที่ 1

บทนำ

1.1 ความเป็นมา

ภาษา XML (Extensible Markup Language) (W3C, 2009) เป็นมาตรฐานที่ใช้กันแพร่หลายในการแลกเปลี่ยนข้อมูลในระบบอินเตอร์เน็ตเป็นภาษาที่ยืดหยุ่นต่อข้อมูลที่มีลักษณะแบบกึ่งโครงสร้าง (semi-structure) และเป็นตัวกลางในการแปลงข้อมูลที่อยู่ในลักษณะการเก็บในรูปแบบที่แตกต่างกันทำให้ระบบต่าง ๆสามารถเข้าถึงและใช้งานข้อมูลร่วมกันได้ (interoperability) ระบบที่นำ XML มาใช้งาน เช่น ระบบ E-commerce เป็นต้น ดังนั้นการทำให้การค้นหาข้อมูลมีประสิทธิภาพ ทำให้ผู้ใช้งานระบบมีความพอใจเป็นงานสำคัญและส่งผลดีต่อระบบ

วิธีการจัดเก็บข้อมูลของเอกสาร XML มี 3 วิธี (Tain et al., 2002; Ozgur et al., 2006; Liu and Murthy, 2009) คือ

1) เก็บข้อมูลของเอกสาร XML เป็นแบบ Files ซึ่งใช้เวลาในการค้นหามาก ถ้ามีการทำดัชนีจะช่วยลดเวลาในการค้นหาลงได้ แต่ต้องเพิ่มพื้นที่สำหรับจัดเก็บดัชนี

2) เก็บข้อมูลของเอกสาร XML ด้วย RDBMS (Relational Database Management System) วิธีนี้จะต้องแปลงข้อมูลจากเอกสาร XML เป็นตารางและใช้เทคนิคการทำดัชนีใน RDBMS ในการค้นหาข้อมูล ถึงแม้ว่า RDBMS จะเป็นฐานข้อมูลที่มีใช้กันมาตั้งแต่ในอดีต แต่การแปลงข้อมูลเอกสาร XML และ การสอบถาม ทำให้เกิด Overhead และข้อมูลที่เก็บในตารางต้องแยกเป็นส่วนๆ ทำให้ต้องการจำนวนการ join มากทำให้ทำงานช้า

3) เก็บข้อมูลของเอกสาร XML ด้วย XML Database ซึ่งเป็นฐานข้อมูลที่ออกแบบมาสำหรับเก็บข้อมูลกึ่งโครงสร้างโดยเฉพาะ จัดเก็บเอกสาร XML โดยไม่ต้องทราบถึงโครงสร้างของเอกสารทำให้จัดเก็บได้ง่าย

ในงานวิจัยนี้มุ่งประเด็นไปที่การค้นหาข้อมูลในเอกสาร XML ที่จัดเก็บในรูปแบบ Files เพราะเอกสาร XML มีลักษณะเป็นโครงสร้างแบบต้นไม้อยู่แล้ว โดยจะสร้างดัชนีจากเอกสาร XML โดยจะทำการลงรหัสให้กับแต่ละอิลิเมนต์และแอททริบิวต์ ซึ่งได้นำ แนวคิด

ของ Huffman Coding มาประยุกต์ใช้สำหรับสร้างรหัสซึ่งจะช่วยลดเวลาในการประมวลผลการสอบถามและลดพื้นที่สำหรับจัดเก็บดัชนีและใช้ตำแหน่งเริ่มต้นและ สิ้นสุดในการเข้าถึงข้อมูลในเอกสาร XML

1.2 การตรวจเอกสารและงานวิจัยที่เกี่ยวข้อง

1.2.1 การตรวจเอกสาร

การจัดเก็บข้อมูลด้วยเอกสาร XML เป็นที่นิยมใช้กัน เนื่องจาก XML เป็นมาตรฐานในการแลกเปลี่ยนข้อมูล ดังนั้นการเพิ่มประสิทธิภาพในการค้นหาข้อมูลนั้นเป็นสิ่งสำคัญ โดยเฉพาะอย่างยิ่งการเข้าถึงข้อมูลจะต้องเป็นไปอย่างรวดเร็ว ซึ่งวิธีหนึ่งที่นิยมใช้ในการเพิ่มประสิทธิภาพ คือ การทำดัชนี ในงานวิจัยที่ผ่านมามีการคิดค้นเทคนิคการทำดัชนีสำหรับเอกสาร XML ที่เก็บอยู่ในรูปแบบ Files ขึ้นมาหลายวิธี โดยมีการพัฒนาประสิทธิภาพทั้งในด้านการใช้พื้นที่และเวลาให้ดีขึ้นเรื่อยๆ

1.2.2 งานวิจัยที่เกี่ยวข้อง

A New Query Processing Technique for XML Based on Signature

งานวิจัยนี้ (Park and Kim, 2001) กล่าวถึงเทคนิคการทำดัชนีแบบ Signature เป็นการแทนค่าข้อมูลแต่ละแท็กในเอกสาร XML ด้วยบิต เพื่อลดพื้นที่สำหรับจัดเก็บดัชนีและเพิ่มความเร็วในการค้นหาข้อมูล เพราะสามารถดำเนินการระดับบิต เช่น AND และ OR เป็นต้น และมีงานวิจัย (Chung et al., 2003; Huang and Wang, 2008) ได้ปรับปรุงเทคนิคการทำดัชนีที่อยู่บนหลักการของดัชนีแบบ Signature เพื่อให้มีประสิทธิภาพเพิ่มขึ้น

The Design and Performance Evaluation of Alternative XML Storage Strategies

งานวิจัยนี้ (Tain et al., 2002) กล่าวถึงวิธีการจัดเก็บเอกสาร XML ทั้ง 3 รูปแบบ คือ 1) แบบ Files 2) XML Database และ 3) RDBMS และได้แสดงผลการเปรียบเทียบประสิทธิภาพในเรื่องของเวลาในการสร้างเอกสาร XML ใหม่ และเวลาในการสอบถามข้อมูล ซึ่งพบว่า การจัดเก็บเอกสาร XML ในรูปแบบ Files ไม่ต้องเสียเวลาในส่วนนี้ ส่วนเรื่องของเวลาในการสอบถามนั้นพบว่า การจัดเก็บเอกสาร XML ในรูปแบบ Files ให้ผลไม่ค่อยดี เนื่องจากยังไม่มีการทำดัชนี

An Efficient XML Query Processing Based on Combining T-Bitmap and Index Techniques และ Indexing XML Data Stored in a Relational Database

งานวิจัยนี้ (Pal et al., 2004; Huang and Wang, 2008) มีการกล่าวถึงรูปแบบของการสอบถามในเอกสาร XML ซึ่งสรุปได้ 3 รูปแบบ คือ 1) การค้นหาตามแกน parent-child เป็นการค้นหาเริ่มจากโหนดราก ในแต่ละขั้นตอนจะท่องต้นไม้ไปตามแกน parent-child 2) การค้นหาตามแกน ancestor-descendant เป็นการสอบถามที่ไม่ต้องเริ่มจากโหนดราก และ 3) การค้นหาที่มีการระบุเงื่อนไข

Answering XML Queries Using Path-Based Indexes: A Survey

งานวิจัยนี้ (Wong et al., 2006) กล่าวถึงเทคนิคการทำดัชนีสำหรับเอกสาร XML ที่เก็บในรูปแบบ Files และได้มีการวิเคราะห์ประสิทธิภาพของแต่ละเทคนิค ตัวอย่างเทคนิคการทำดัชนี เช่น DataGuide 1-Index 2-Index และ Index fabric เป็นต้น ซึ่งเทคนิคเหล่านี้มีประสิทธิภาพในบางเรื่องแต่ไม่สามารถครอบคลุมได้ทุกการสอบถาม รวมถึงยังใช้พื้นที่ในการจัดเก็บมากอีกด้วย

A Decade of XML Data Management: An Industrial Experience from Oracle

งานวิจัยนี้ (Liu and Murthy, 2009) กล่าวถึงวิธีการจัดเก็บเอกสาร XML ในรูปแบบของ RDBMS โดยได้กล่าวไว้ว่า RDBMS ไม่เหมาะกับการจัดเก็บเอกสาร XML เนื่องจากยังมีปัญหาในเรื่องของ Back Tracking

1.3 วัตถุประสงค์

1.3.1 เพื่อวิเคราะห์และออกแบบเทคนิคการเพิ่มประสิทธิภาพการทำดัชนีสำหรับเอกสาร XML เพื่อให้มีประสิทธิภาพทั้งในด้านการใช้พื้นที่ที่ใช้ในการจัดเก็บดัชนีและเวลาที่ใช้ในการสอบถาม

1.3.2 เพื่อพัฒนาเทคนิคการเพิ่มประสิทธิภาพการทำดัชนีสำหรับเอกสาร XML เพื่อให้มีประสิทธิภาพทั้งในด้านการใช้พื้นที่ที่ใช้ในการจัดเก็บดัชนีและเวลาที่ใช้ในการสอบถาม

1.4 วิธีการดำเนินการวิจัย

1. ศึกษาแนวคิดที่เกี่ยวข้องกับเอกสาร XML และเทคนิคการทำดัชนีสำหรับเอกสาร XML ที่เคยมีมา

2. วิเคราะห์และออกแบบเทคนิคใหม่ในการทำดัชนี สำหรับเอกสาร XML ซึ่งดัชนีใหม่ที่ได้นี้เรียกว่า XTI (XML Tree Index)

3. กำหนดรูปแบบการประเมินประสิทธิภาพของดัชนี XTI ที่คิดค้นใหม่ กับดัชนีเปรียบเทียบที่เคยมีมา โดยทำการประเมินประสิทธิภาพทั้งในด้านการใช้พื้นที่สำหรับจัดเก็บดัชนีและเวลาที่ใช้ในการสอบถาม ซึ่งมีขั้นตอนดังต่อไปนี้

3.1 ศึกษาข้อมูลสำหรับใช้ทดสอบ ซึ่งเป็นข้อมูลมาตรฐานจาก Benchmark (Schmidt et al., 2002)

3.2 ติดตั้งโปรแกรมสำหรับรันผลการสร้างข้อมูลทดสอบ บนระบบปฏิบัติการ Window XP Professional

3.3 จัดเตรียมข้อมูลสำหรับทดสอบ โดยการเลือกข้อมูลเอกสาร XML ที่ factor 0.02

3.4 ออกแบบขั้นตอนวิธีในการสร้างดัชนี XTI และดัชนีเปรียบเทียบ

3.5 พัฒนาโปรแกรมเพื่อสร้างดัชนีตามขั้นตอนที่ได้ออกแบบไว้ในข้อ 3.4 โดยใช้ตัวแปลภาษาจาวา

4. ออกแบบขั้นตอนวิธีการสอบถามบนดัชนี XTI และดัชนีเปรียบเทียบ

5. พัฒนาโปรแกรมเพื่อสอบถามข้อมูลบนดัชนี XTI และดัชนีเปรียบเทียบตามที่ได้ออกแบบไว้ในข้อ 4 โดยใช้ตัวแปลภาษาจาวา

6. ประเมินประสิทธิภาพการใช้พื้นที่ในการจัดเก็บดัชนี (Space) และเวลาในการสอบถาม (Time)

7. วิเคราะห์และสรุปผลการประเมินประสิทธิภาพ

1.5 ขอบเขตงานวิจัย

วิเคราะห์ ออกแบบ และพัฒนาเทคนิคการเพิ่มประสิทธิภาพการทำดัชนีสำหรับเอกสาร XML เพื่อให้มีประสิทธิภาพทั้งในด้านการใช้พื้นที่ที่ใช้จัดเก็บดัชนีและเวลาที่ใช้ในการสอบถาม

ประเมินประสิทธิภาพของการใช้พื้นที่ในการจัดเก็บดัชนี และเวลาในการสอบถามข้อมูล

1.6 ขั้นตอนการดำเนินงาน

1. ศึกษางานวิจัยและเอกสารที่เกี่ยวข้องและกำหนดขอบเขตของปัญหา
2. วิเคราะห์และออกแบบเทคนิค
3. กำหนดรูปแบบการประเมินประสิทธิภาพ
4. ศึกษาและวิเคราะห์หาเครื่องมือสำหรับประเมินประสิทธิภาพ
5. พัฒนาดัชนีตามที่ได้ออกแบบไว้
6. ประเมินประสิทธิภาพดัชนีที่สร้างขึ้นใหม่กับดัชนีเปรียบเทียบ
7. สรุปและวิเคราะห์ผลการประเมินเปรียบเทียบดัชนี
8. เขียนบทความนำเสนอในที่ประชุมวิชาการ/วารสาร
9. จัดทำเอกสารประกอบการวิจัย

1.7 ระยะเวลาดำเนินงานและแผนดำเนินงาน

ระยะเวลาดำเนินงาน มกราคม 2552 – เมษายน 2553

ตารางที่ 1-1 ระยะเวลาดำเนินการ

ขั้นตอนการดำเนินการ	เดือน															
	2552												2553			
	1	2	3	4	5	6	7	8	9	10	11	12	1	2	3	4
1. ศึกษางานวิจัยและเอกสารที่เกี่ยวข้อง																
2. วิเคราะห์และออกแบบเทคนิค																
3. กำหนดรูปแบบการประเมินประสิทธิภาพ																
4. ศึกษาเครื่องมือสำหรับใช้ประเมินประสิทธิภาพ																
5. พัฒนาดัชนีตามที่ได้ออกแบบไว้																

6. ประเมินประสิทธิภาพกับดัชนีเปรียบเทียบ																
7. เขียนบทความตีพิมพ์ในที่ประชุมวิชาการ/วารสาร																
8. ตีพิมพ์																
9. จัดทำเอกสารประกอบการวิจัย																

1.8 เครื่องมือและอุปกรณ์

1.8.1 สถานที่

ห้องปฏิบัติการวิจัยเทคโนโลยีระบบสารสนเทศและการประยุกต์ (CS207) ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยสงขลานครินทร์

1.8.2 เครื่องมือที่ใช้

ด้านฮาร์ดแวร์

1. เครื่องคอมพิวเตอร์จำนวน 1 เครื่อง รุ่น Intel® Core™2 Quad ระบบปฏิบัติการ Window XP Professional
2. หน่วยความจำขนาด 1 GB
3. ฮาร์ดดิสก์ ความจุ 320 GB

ด้านซอฟต์แวร์

1. ระบบปฏิบัติการ Window XP Professional
2. ตัวแปลภาษาจาวา สำหรับใช้ในการพัฒนาโปรแกรมเพื่อวัดประสิทธิภาพของเวลาที่ใช้ในการสอบถามข้อมูล

1.9 ประโยชน์ที่คาดว่าจะได้รับ

ได้เทคนิคการเพิ่มประสิทธิภาพดัชนีสำหรับเอกสาร XML ที่มีประสิทธิภาพทั้งในด้านการใช้พื้นที่ที่ใช้ในการจัดเก็บดัชนีและเวลาที่ใช้ในการสอบถาม

บทที่ 2

ทฤษฎีที่เกี่ยวข้อง

ในบทนี้จะกล่าวถึงทฤษฎีต่างๆ ประกอบด้วย ภาษา XML (Extensible Markup Language) การจัดเก็บเอกสาร XML สำหรับทำดัชนี แนวคิดของ Huffman Coding และ การค้นหาข้อมูลในต้นไม้ (Tree)

2.1 ภาษา XML (Extensible Markup Language)

ส่วนนี้จะกล่าวถึงประวัติความเป็นมา ความหมาย ส่วนประกอบ รูปแบบของ XPath และประโยชน์ของภาษา XML

2.1.1 ความเป็นมาของภาษา XML

ในปี ค.ศ.1967 มีการนำเสนอแนวคิดของภาษา Markup (สุธี พงศาสกุลชัย, 2007) ที่ใช้ในการแยกส่วนโครงสร้างของเอกสารจากส่วนแสดงผล เพื่อนำเอกสารไปใช้งานอื่นๆ นอกจากส่วนแสดงผล ต่อมาในปี ค.ศ.1969 บริษัท IBM ได้คิดค้นภาษา GML (Generalized Markup Language) เพื่อใช้ประมวลผลข้อความ โดยแนวคิดของภาษานี้ คือ เพื่อให้นักพัฒนาคิดค้นกลุ่มแท็ก (Tag) ที่ใช้อธิบายเอกสารขึ้นเองได้ แต่ยังไม่มีมาตรฐาน จึงมีการคิดค้นภาษาใหม่คือ SGML (Standard Generalized Markup Language) เป็นการรวบรวมมาตรฐานต่างๆ ไว้ แต่ต้องมีเอกสารที่เรียกว่า DTD (Document Type Definition) เพื่อใช้นิยามโครงสร้างของเอกสาร ทำให้ภาษา SGML ยากต่อการเรียนรู้และไม่ได้รับความนิยม

ในปี 1980 มีการกำเนิดภาษา HTML (Hypertext Markup Language) ซึ่งมีแนวคิดพื้นฐานมาจาก SGML โดยมีวัตถุประสงค์เพื่อใช้ในการแสดงผลข้อมูลบนอินเตอร์เน็ตผ่านโปรแกรมเบราว์เซอร์เท่านั้น ซึ่งยากในการดึงส่วนของข้อมูลที่ต้องการออกมา ทำให้มีการคิดค้นภาษาใหม่ที่เรียกว่า XML (Extensible Markup Language) ในปี 1996 ภายใต้การดูแลของ W3C (World Wide Web Consortium) โดยจะนำเอาข้อดีของภาษา SGML ในการนิยามข้อมูล และความสามารถในการใช้งานผ่านระบบอินเตอร์เน็ตของภาษา HTML เข้าไว้ด้วยกัน

2.1.2 ความหมายของภาษา XML

XML เป็นภาษา Markup ที่ใช้วิธีการระบุเนื้อหาและจัดรูปแบบข้อมูลด้วยไฟล์

ข้อความ โดยได้รับการออกแบบเพื่อใช้แท็กในการอธิบายความหมายของข้อมูลผู้ใช้สามารถกำหนดแท็กที่ใช้งานได้ตามต้องการ ทำให้ XML มีความยืดหยุ่นสามารถใช้งานได้หลากหลายและถูกนำมาใช้เป็นสื่อกลางในการแลกเปลี่ยนข้อมูลข่าวสารผ่านระบบอินเตอร์เน็ต

2.1.3 ส่วนประกอบของเอกสาร XML

1) แท็ก (Tag)

เป็นส่วนประกอบสำคัญของภาษา Markup การกำหนดแท็กเริ่มต้น (Start Tag) ชื่อแท็กอยู่ภายใต้เครื่องหมาย ”<” และ ”>” เช่น <product> ส่วนการกำหนดแท็กสิ้นสุด (End Tag) อยู่ภายใต้เครื่องหมาย ”</” และ ”>” เช่น </product> และตั้งแต่แท็กเริ่มต้นไปถึงแท็กสิ้นสุดจะถูกเรียกว่า อิลิเมนต์ (Element) ตัวอย่างแสดงดังภาพประกอบ 2-1 อธิบายได้ดังนี้

<product>	เป็นแท็กเริ่มต้น
</product>	เป็นแท็กสิ้นสุด
<product> TV </product>	เป็นอิลิเมนต์

```
<product> TV </product>
```

ภาพประกอบ 2-1 ตัวอย่างอิลิเมนต์ของ XML

2) แอททริบิวต์ (Attribute)

การระบุคุณสมบัติให้กับอิลิเมนต์ ใช้เพื่ออธิบายส่วนเพิ่มเติมให้กับแต่ละอิลิเมนต์ โดยมีรูปแบบการกำหนดดังนี้

<element attribute = “value” | ‘value’> text </element>

ตัวอย่างการกำหนดค่าของแอททริบิวต์แสดงดังภาพประกอบ 2-2

```
<product id = “001”> TV </product>
หรือ
<product id = ‘001’> TV </product>
```

ภาพประกอบ 2-2 ตัวอย่างการกำหนดค่าแอททริบิวต์

3) Entity

เป็นกลุ่มของอักขระที่ถูกกำหนดความหมายไว้แล้ว โดย XML Parser จะประมวลผลกลุ่มอักขระนั้นๆ แล้วส่งค่าออกมาเป็นผลลัพธ์ สำหรับ Entity ของ XML มีอยู่ด้วยกัน 2 ประเภท ดังนี้

1.) Entity ที่กำหนดโดย W3C ทำให้ XML สามารถประมวลผล Entity นั้นๆได้อย่างถูกต้อง โดย Entity ที่สำคัญมีดังนี้

< มีค่าเท่ากับ <

&gt มีค่าเท่ากับ >

& มีค่าเท่ากับ &

" มีค่าเท่ากับ “

' มีค่าเท่ากับ ‘

2.) Entity ที่ผู้ใช้กำหนดเอง ผู้พัฒนาสามารถกำหนด Entity ขึ้นใช้เองได้ แต่ต้องมีการประกาศความหมายของ Entity ไว้ในส่วนของ Document Type Definition (DTD) ก่อน

2.1.4 กฎพื้นฐานในการเขียน XML (Well-Formed)

1. ทุกอิลิเมนต์ของ XML จะต้องประกอบด้วยแท็กเริ่มต้นและแท็กสิ้นสุด โดยทั้งสองแท็กจะต้องมีชื่อเหมือนกัน

2. การกำหนดชื่อแท็กจะคำนึงถึง Case Sensitive คือตัวอักษรพิมพ์ใหญ่พิมพ์เล็กมีความหมายแตกต่างกัน

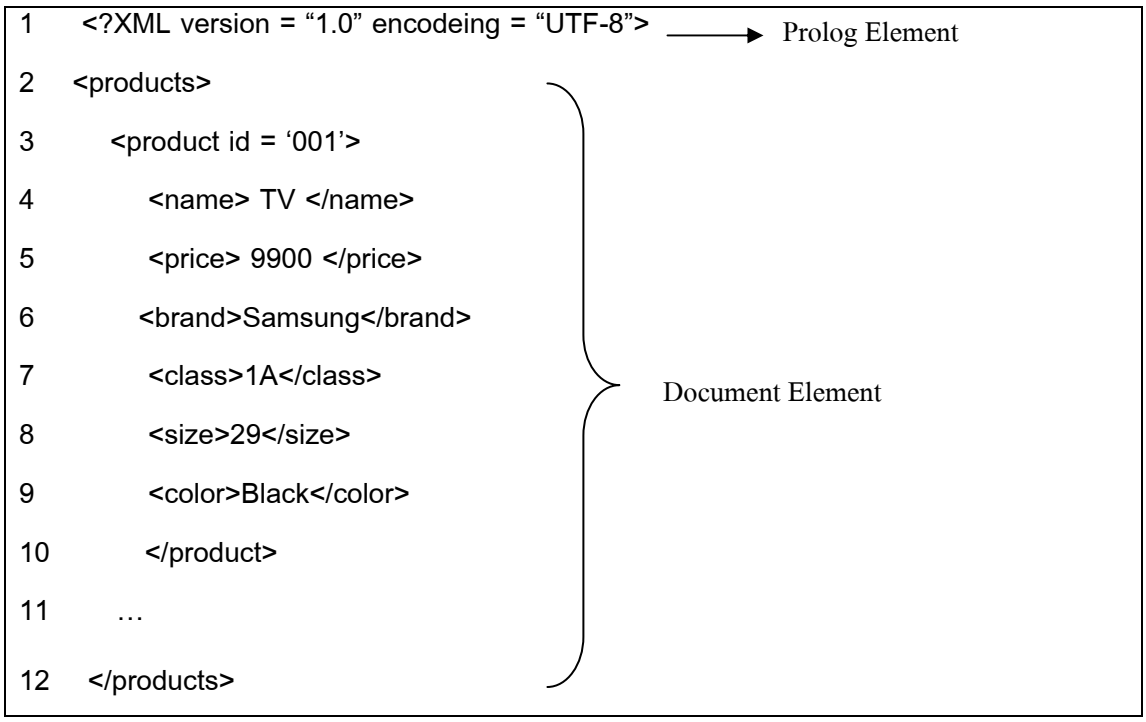
3. ทุกเอกสาร XML ต้องมี Root Element และมีได้เพียงหนึ่ง Root เท่านั้น

4. อิลิเมนต์ของ XML ต้องซ้อนกันเป็นลำดับ เช่น <product><name>…</name></product>

5. XML จะไม่ตัดส่วนที่เป็น White Space ในข้อความออก เช่น การเว้นวรรค เป็นต้น

6. การตั้งชื่ออิลิเมนต์ของเอกสาร XML สามารถใช้อักขระ ตัวเลข และอักขระพิเศษได้ ยกเว้นเครื่องหมาย & และไม่สามารถใช้ตัวเลขหรือตัวอักขระพิเศษนำหน้าชื่อของอิลิเมนต์ นอกจากนี้ยังห้ามเว้นช่องว่างระหว่างชื่ออิลิเมนต์ด้วย

ตัวอย่างเอกสาร XML แสดงดังภาพประกอบ 2-3



ภาพประกอบ 2-3 ตัวอย่างเอกสาร XML

จากภาพประกอบ 2-3 บรรทัดที่ 1 คือส่วน Prolog Element เป็นส่วนที่ใช้ประกาศเอกสาร XML ที่บรรจุส่วนต่างๆ ได้แก่ เวอร์ชั่น หมายเหตุ และ DTD เป็นต้น

ส่วนบรรทัดที่ 2-12 คือส่วนของ Document Element เป็นอิลิเมนต์ที่ประกอบด้วยอิลิเมนต์ย่อยซ้อนกันอยู่เป็นลำดับ

2.1.5 การกำหนดโครงสร้างเอกสาร XML

เป็นการกำหนดรูปแบบโครงสร้างของเอกสาร XML ว่าประกอบด้วยอิลิเมนต์อะไรบ้าง มีลำดับการใช้งานอย่างไรและข้อมูลที่อิลิเมนต์ต่าง ๆ เก็บไว้คืออะไร ซึ่งเป็นส่วนสำคัญที่ใช้สำหรับตรวจสอบความถูกต้องของข้อมูลภายในเอกสาร XML โดยวิธีกำหนดโครงสร้างเอกสาร XML ที่นิยมมีดังนี้ (W3Schools, 2008)

1. DTD (Document Type Definition) เป็นข้อกำหนดและกฎเกณฑ์ของเอกสาร XML มีองค์ประกอบหลักๆ คือ อิลิเมนต์ แอททริบิวต์ Entities PCDATA (Parsed Character Data) และ CDATA (Character Data) ภาพประกอบ 2-4 เป็น DTD ของเอกสาร XML ในภาพประกอบ 2-3

```
<!ELEMENT products (product, id, name, price, brand, class, size, color)>
<!ELEMENT product EMPTY>
<!ATTLIST product id CDATA “001”>
<!ELEMENT name (#PCDATA)>
<!ELEMENT price (#PCDATA)>
<!ELEMENT brand (#PCDATA)>
<!ELEMENT class (#PCDATA)>
<!ELEMENT size (#PCDATA)>
<!ELEMENT color (#PCDATA)>
```

ภาพประกอบ 2-4 ตัวอย่าง DTD

2. XML Schema การนิยามโครงสร้างของ DTD มีข้อจำกัดหลายอย่าง ทำให้ยากต่อการเรียนรู้และนำไปใช้งาน ดังนั้น W3C ได้กำหนดมาตรฐานใหม่สำหรับการกำหนดโครงสร้างเอกสาร XML เรียกว่า XML Schema ช่วยให้จัดการโครงสร้างของเอกสาร XML ได้ดีขึ้น เนื่องจากเขียนด้วย ภาษา XML จากเอกสาร XML ภาพประกอบ 2-5 แสดง XML Schema ของเอกสาร XML ในภาพประกอบ 2-3 และในตารางที่ 2-1 แสดงการเปรียบเทียบ DTD และ XML Schema

```
<?xml version="1.0"?>
<xs:schema
xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name=“products”>
   <xs:complexType>
      <xs:sequence>
         <xs:element name="product"/>
            <xs:complexType>
               <xs:attribute name="id" type="xs:string"/>
            <xs:/complexType>
         <xs:/element>
         <xs:element name="name" type="xs:string"/>
         <xs:element name="price" type="xs:integer"/>
         <xs:element name="brand" type="xs:string"/>
         <xs:element name="class" type="xs:string"/>
         <xs:element name="size" type="xs:string"/>
         <xs:element name="color" type="xs:string"/>
      <xs:/sequence>
   <xs:/ complexType>
<xs:/element>

<xs:/schema>
```

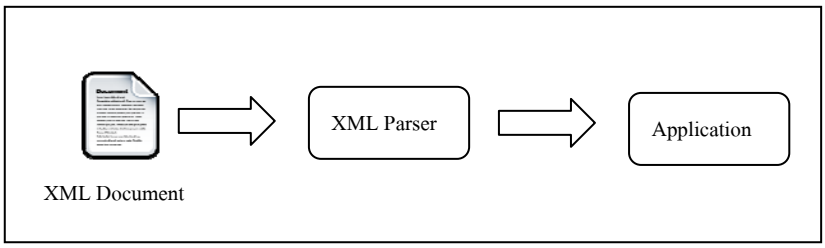
ภาพประกอบ 2-5 ตัวอย่าง XML Schema

ตารางที่ 2-1 เปรียบเทียบ DTD กับ XML Schema

DTD	XML Schema
- รองรับชนิดข้อมูลได้น้อยกว่า XML Schema - ไม่ได้อยู่ภายใต้ไวยากรณ์ของ XML	- รองรับชนิดข้อมูลได้มากกว่า DTD - อยู่ภายใต้ไวยากรณ์ของ XML - กำหนดรูปแบบข้อมูลเองได้ - กำหนดจำนวนและลำดับของอิลิเมนต์ลูกได้ - รองรับการใช้ Namespace

2.1.6 XML Parser

คือ ตัวแปรภาษา XML ทำหน้าที่อ่านและแปลความหมาย วิเคราะห์โครงสร้าง รวมถึงตรวจสอบความถูกต้องของเอกสาร XML โดยกระบวนการทำงานของ Parser แสดงดังภาพประกอบ 2-6

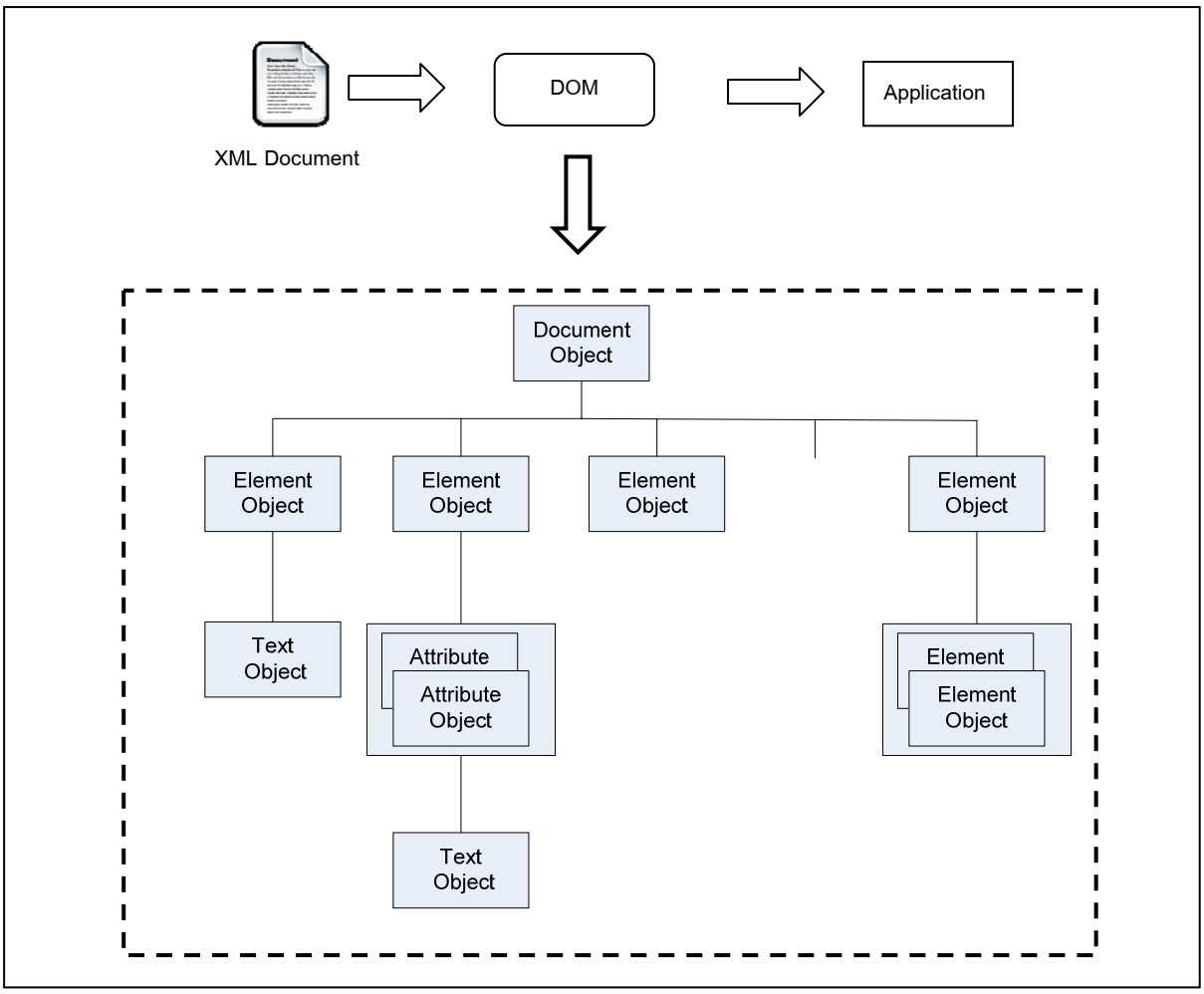


ภาพประกอบ 2-6 กระบวนการทำงานของ XML Parser

XML Parser (Benz and Durant, 2003; Silberschatz et al., 2006; เดวิด ฮันเตอร์, 2002) สามารถจำแนกตามวิธีการสำรวจเนื้อหาของเอกสารออกเป็น 2 ชนิด คือ DOM (Document Object Model) ซึ่งเป็นแบบ Tree-based Parser และ SAX (Simple API for XML) เป็นแบบ Event-driven Parser

1. DOM (Document Object Model)

DOM ได้รับการรับรองเป็นมาตรฐานโดย W3C วิธีการทำงานของ DOM จะอ่านเอกสาร XML ทั้งหมดสร้างเป็นโครงสร้างต้นไม้ (Tree) ในหน่วยความจำของคอมพิวเตอร์ที่ประกอบด้วยอิลิเมนต์หรือแอททริบิวต์ต่างๆ การเข้าถึงข้อมูลใช้วิธีการเดินเข้าถึง (Traverse) ต้นไม้ และจะมองเอกสารเป็นอ็อบเจ็กต์ ซึ่งโครงสร้างการทำงานของ DOM แสดงภาพประกอบที่ 2-7



ภาพประกอบ 2-7 กระบวนการทำงานของ DOM

ข้อดีของ DOM

(1) สามารถทำงานได้ดีเมื่อจะเข้าถึงโหนดต่าง ๆแบบสุ่มและเขียนชุดคำสั่งได้ง่าย

(2) สนับสนุนการอ่านและ เขียนข้อมูลลงไฟล์

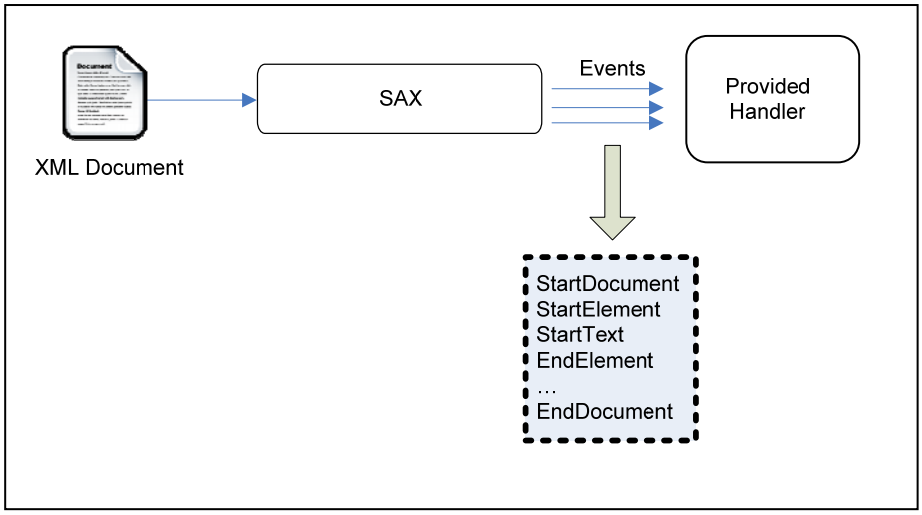
(3) อ่านเอกสาร XML เพียงครั้งเดียว

ข้อเสียของ DOM

(1) ต้องใช้หน่วยความจำเป็นจำนวนมาก

2. SAX (Simple API for XML)

หลักการทำงานของ SAX เป็นแบบ Event – driven คือจะไม่โหลดข้อมูลทั้งหมดในเอกสาร XML เข้ามาในหน่วยความจำ แต่จะแปลความหมายของเหตุการณ์ที่เกิดขึ้นเป็นหลัก เมื่อ Parser อ่านข้อมูลจากเอกสาร XML ในแต่ละครั้งจะจดจำโครงสร้างไวยากรณ์ของเอกสาร XML ไว้ (จะสร้าง Event ขึ้นตามข้อมูลในเอกสาร XML เช่น StartElement และ EndElement เป็นต้น) และตอบสนองต่อเหตุการณ์ที่อ่านได้ ด้วยวิธีการที่กำหนดไว้ แสดงภาพประกอบ 2-8



ภาพประกอบ 2-8 กระบวนการทำงานของ SAX

ข้อดีของ SAX

(1) การกระทำกับ XML อย่างง่ายเช่น การอ่าน

(2) ใช้หน่วยความจำน้อย

(3) ทำงานได้ดีกับเอกสาร XML ไฟล์ที่มีขนาดใหญ่

(4) เข้าถึงข้อมูลเฉพาะที่ต้องการทำให้ทำงานเร็ว

ข้อเสียของ SAX

(1) ไม่เหมาะสำหรับการจัดการโครงสร้าง XML

(2) ไม่เหมาะกับการเขียนข้อมูลลง XML

(3) ต้องอ่านเอกสาร XML ทุกครั้งที่ต้องการเข้าถึงข้อมูล

2.1.7 XPath (XML Path Language)

XPath (W3C, 2009) เป็นภาษาที่ใช้ในการสอบถามข้อมูลโดยการระบุตำแหน่งของโหนดหรืออิลิเมนต์ในเอกสาร XML ใช้ path expression ในการเข้าถึงข้อมูล สัญลักษณ์ที่ใช้ใน XPath แสดงได้ดังตารางที่ 2-2

<u>สัญลักษณ์ที่ใช้ใน XPATH</u>

สัญลักษณ์ที่ใช้ใน XPath สามารถแสดงได้ดังตารางที่ 2-2

ตารางที่ 2-2 สัญลักษณ์ที่ใช้ใน XPath

Expression	Description
Node name	การเอ่ยชื่อลอยๆ หมายถึง การเข้าถึงโหนดลูกทุกตัวของโหนดที่มีชื่อปรากฏ
/	หากขึ้นต้นด้วยเครื่องหมาย / แล้วตามด้วยชื่อโหนด หมายถึง การอ้างจากตำแหน่งอ้างอิงที่เป็นโหนดราก หรือ root element
//	การขึ้นต้นด้วยเครื่องหมาย // นำหน้าชื่อโหนด หมายถึง การอ้างโหนดโดยไม่สนใจว่าข้อมูลจะอยู่ที่ใดใน XML
.	การขึ้นต้นด้วยเครื่องหมาย . นำหน้าชื่อหรือการอ้างอิง หมายถึง โหนดปัจจุบัน
..	การขึ้นต้นด้วยเครื่องหมาย .. หรือใช้ .. หมายถึง โหนดพ่อแม่ของโหนดปัจจุบัน
@	เข้าถึงสมาชิกที่เป็นแอตทริบิวต์ของอิลิเมนต์

<u>รูปแบบการสอบถาม (Query)</u>

การสอบถามข้อมูลในเอกสาร XML อยู่ในรูปของ XPath สามารถแบ่งได้ 2 แบบดังนี้

1) Absolute Path Expression (APE)

เป็นการสอบถามที่เริ่มจากโหนดราก ในแต่ละขั้นตอนจะท่องต้นไม้ไปตามแกนพ่อแม่ – ลูก (Parent – Child Axis) เช่น products/product/id

2) Regular Path Expression (RPE)

เป็นการสอบถามที่ไม่ต้องเริ่มจากโหนดราก สามารถท่องต้นไม้ไปตามแกนต่าง ๆได้ เช่น //product/name

ตัวอย่าง XPath

แสดงตัวอย่างของ XPath จากเอกสาร XML ในภาพประกอบ 2-3 ดังนี้

- ต้องการหาสินค้าทั้งหมด

XPath : /products

- ต้องการหารหัสสินค้าของสินค้าทั้งหมด

XPath : /products/product/id

- ต้องการหาชื่อของสินค้าทั้งหมด

XPath : //product/name

- ต้องการหาราคาสินค้าที่มีรหัสสินค้าเป็น 001

XPath : //product/@id = '001'/price

2.1.8 ประโยชน์จาก XML

ประโยชน์ของ XML ยกตัวอย่างได้ดังนี้

- self-describe data: ใช้สำหรับสร้างข้อมูลที่สามารถอธิบายเนื้อหาเองได้ ทำให้ง่ายในการจัดการ

- data exchange: ใช้สำหรับการแลกเปลี่ยนข้อมูล

- messaging format: กำหนดรูปแบบข้อความในการสื่อสาร ระหว่างแอพพลิเคชั่นหรือโปรแกรม เช่นโพรโตคอล SOAP เป็นต้น

- ใช้สำหรับการเข้าถึงระบบข้อมูลขนาดใหญ่ เช่น ใช้กับระบบเครือข่ายในองค์กร หรืออินเตอร์เน็ตเพื่อดูข้อมูลหรือเรียกใช้ข้อมูลที่ให้การแสดงผลทางหน้าจอที่รวดเร็ว

- XML จะเกิดความสะดวกในระบบพาณิชย์อิเล็กทรอนิกส์ ในระบบการค้าอิเล็กทรอนิกส์ ข้อมูลจะถูกเก็บอยู่ในรูปแบบของเอกสาร XML ทั้งหมด

- XML สนับสนุนการทำงานกับ UNICODE และผสมได้หลากหลายภาษา

- การพัฒนา XML Processor ทำให้สามารถดึงเอกสาร XML มาใช้งานได้ง่าย และใช้ร่วมกับโปรแกรมประยุกต์อื่นได้ง่าย เช่น โปรแกรม DB2, Oracle, SAP เป็นต้น

- XML ช่วยทำให้เกิดการรับส่งข้อมูลแบบ Electronic Data Interchange โดยทำให้แนวทางการเชื่อมโยงและสร้างความเป็นเอกสาร หรือมาตรฐานระหว่างองค์กร

- นำไปประยุกต์ใช้ในการดำเนินกิจกรรมบนเครือข่าย เช่น eBusiness, EDI, eCommerce, การจัดการ Supply chain, Demand chain management การดำเนินการแบบ Intranet และ Web base application

2.2 การจัดเก็บข้อมูลของเอกสาร XML สำหรับทำดัชนี

การจัดเก็บข้อมูลของเอกสาร XML เพื่อทำดัชนีมี 3 วิธี (Tain et al., 2002; Ozgur et al., 2006; Liu and Murthy, 2009)

1) Files

วิธีนี้เป็นการเก็บเอกสาร XML ทั่วไป (Goldman and Widom, 1997; Milo and Suciu, 1999; Cooper et al., 2001; Park and Kim, 2001; Chung et al., 2003, Huang and Wang, 2008)

ข้อดี

(1) เป็นวิธีที่สามารถสร้างได้ง่าย

ข้อเสีย

(1) การค้นหาข้อมูลทำได้ยากและใช้เวลานาน จึงต้องใช้การทำดัชนีช่วยในการค้นหาข้อมูล แต่ต้องเพิ่มพื้นที่สำหรับจัดเก็บดัชนี

2) RDBMS

ข้อมูลในเอกสาร XML ถูกแปลงเป็นตารางในฐานข้อมูล (Seo et al., 2003; Ozgur and Gundem, 2006; Jiang et al., 2002; Pal et al., 2004)

ข้อดี

(1) ทำให้ระบบฐานข้อมูลที่มีมายาวนานได้ใช้ประโยชน์

(2) เป็นที่นิยมใช้ในหลายองค์กร

(3) เก็บข้อมูลโดยไม่ต้องใช้ DTD

ข้อเสีย

(1) เสียเวลาในการแปลงข้อมูลและการสอบถาม

(2) ข้อมูลถูกแยกเป็นส่วนของตารางเล็กๆ ทำให้เพิ่มพื้นที่การเก็บและต้องใช้จำนวนการ join ที่มากขึ้นทำให้ช้าในการค้นหา

(3) สิ้นเปลืองทรัพยากร (เพิ่มค่า I/O Cost)

3) XML Database

เป็นฐานข้อมูลที่ออกแบบมาสำหรับเก็บข้อมูลกึ่งโครงสร้างโดยเฉพาะ มีการเก็บข้อมูลใช้พื้นฐานของโครงสร้างแบบต้นไม้ ในการเก็บเอกสาร XML ไม่จำเป็นต้องใช้โครงสร้างเอกสาร (McHugh et al., 1997; Bancihon st al., 1988)

ข้อดี

(1) สนับสนุนการทำงานของ DOM

(2) เนื่องจาก XML มีลักษณะเป็นโครงสร้างแบบต้นไม้อยู่แล้วทำให้สะดวกในการจัดการกับเอกสาร XML

ข้อเสีย

- สิ้นเปลืองเนื้อที่ในการเก็บข้อมูลและต้องใช้ทรัพยากรในการทำงานมาก

2.3 แนวคิดของ Huffman Coding

Huffman Coding (Anany, 2007; Huffman, 1952) เป็นอัลกอริทึมที่พัฒนาขึ้นโดย Dr. Huffman ในปี 1952 ซึ่งเป็นอัลกอริทึมที่นิยมใช้สำหรับบีบอัดข้อมูล ซึ่งผลที่ได้จากการบีบอัดข้อมูลด้วยอัลกอริทึมนี้อยู่ในรูปแบบของ Lossless Compression โดยมีหลักการคือ จะใช้ตารางที่บรรจุ Codeword ที่ได้สร้างขึ้นมาแทนที่ข้อมูลในเอกสาร บางครั้งอาจจะเรียกตารางนี้ว่า Dictionary เพื่อใช้ในการเข้าถึงข้อมูลในเอกสาร สำหรับการถอดรหัสก็ใช้ตารางนี้เช่นเดียวกัน

Huffman Coding มีสิทธิภาพดีมากในเรื่องของการบีบอัดข้อมูล และ เพิ่มความเร็วในการประมวลผลการค้นหา นอกจากนี้ยังช่วยลดขนาดของเนื้อที่ในการเก็บข้อมูลได้อีกด้วย (Hashemian, 1995) การสร้าง Huffman Coding มี 2 แบบ คือ 1) แบบ fixed-length ซึ่งวิธีนี้จะมีความยาวของรหัสที่ได้ไม่เกิน $log_2 n$ สมมุติให้มีข้อมูลทั้งหมด n ตัว 2) แบบไม่ fixed-length

หลักการของ Huffman Coding สามารถอธิบายได้เป็น 3 ขั้นตอน ประกอบด้วย ข้อมูลนำเข้า ผลลัพธ์ และเป้าหมายของอัลกอริทึม

2.3.1 ข้อมูลนำเข้า (Input)

กำหนดให้ A เป็นเซต (Set) ของอักขระ ที่ไม่เป็นเซตว่าง และมีจำนวนข้อมูล n ตัว และกำหนดให้ W เป็นเซตของน้ำหนักของอักขระแต่ละตัวในเซต A ตั้งแต่ a_1 จนถึง a_n ซึ่งสามารถเขียนให้อยู่ในรูปสมการได้ดังนี้

$$A = \{a_1, a_2, ..., a_n\}$$
$$W = \{w_1, w_2, ..., w_n\}$$

โดยที่ค่าสมาชิกแต่ละตัวของเซต W นั้นจะต้องมีค่าเป็นจำนวนจริง และสมาชิกในทั้งสองเซตจะต้องมีค่าเป็น n เท่ากัน สามารถเขียนเป็นสมการได้ดังนี้

$$w_i = weight(a_i) \qquad \text{for } 1 \le i \le n$$

2.3.2 ผลลัพธ์ (Output)

สำหรับผลลัพธ์ที่ได้จากอัลกอริทึมนี้ จะต้องหาเซตของ $C(A,W)$ ซึ่งประกอบด้วย C_1 ถึง C_n และเรียกสมาชิกแต่ละตัวว่า Codeword ซึ่งสามารถเขียนเป็นสมการได้ดังนี้

$$C(A,W) = \{c_1, c_2, c_3, ..., c_i, ..., c_n\} \quad \text{for } 1 \le i \le n$$

2.3.3 เป้าหมายของอัลกอริทึม

กำหนดให้ $L(C)$ เป็นผลรวมของน้ำหนักสมาชิกทุกตัวในเซต W คูณกับความยาวของ Codeword ในแต่ละตัว เป้าหมายของอัลกอริทึมคือ เพื่อที่จะต้องการหาค่า $L(C)$ ที่น้อยที่สุดเท่าที่จะทำได้ สามารถเขียนเป็นสูตรในการหา $L(C)$ ได้ดังนี้

$$L(C) = \sum_{i=1}^{n} (W_i \times length(C_i))$$

ตัวอย่างเช่น ถ้ามีข้อมูลชุดหนึ่งประกอบด้วยอักขระ 5 ตัว คือ A, B, C, D และ _ ซึ่งมีค่าความน่าจะเป็นในการพบอักขระดังกล่าวในเอกสารเป็น 35% 10% 20% 20% และ

15% ตามลำดับ เมื่อใช้ Huffman Coding ในการแทนที่ข้อมูลแต่ละตัวทั้งสองวิธีสามารถแสดงได้ดังตารางที่ 2-3

ตารางที่ 2-3 ตัวอย่างการใช้งาน Huffman Coding

อินพุต	อักขระ	A	B	C	D	_	
	ค่าน้ำหนัก	35	10	20	20	15	=100
เอาท์พุต	Codeword : non fixed length	11	100	00	01	101	
	ความยาว Codeword (บิต)	2	3	2	2	3	
	Codeword :fixed length	000	001	010	011	100	
	ความยาว Codeword (บิต)	3	3	3	3	3	

จากตาราง จะคำนวณเนื้อที่ เมื่อนำอัลกอริทึม Huffman Coding มาใช้ในการแทนที่ข้อมูล จะใช้เนื้อที่ในการเก็บดังนี้

แบบ non fixed length

$$L(C) = (35 \times 2) + (10 \times 3) + (20 \times 2) + (20 \times 2) + (15 \times 3)$$

$$L(C) = 225$$

แบบ fixed length

$$L(C) = (35 + 10 + 20 + 20 + 15) \times 3$$

$$L(C) = 300$$

ดังนั้นจะได้ว่า non fixed length ใช้เนื้อที่ในการเก็บข้อมูลเพียง 225 บิต และแบบ fixed length จะใช้เนื้อที่ในการเก็บข้อมูลเพียง 300 บิต

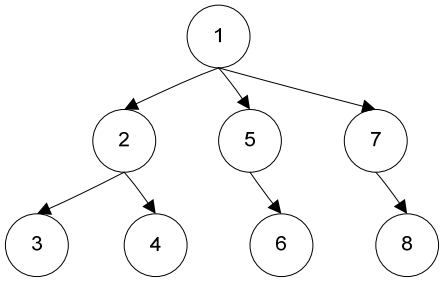
Huffman Coding นั้นจะไม่มีวิธีการทำที่ตายตัว แต่ส่วนมากจะใช้ต้นไม้ทวิภาค (Binary Tree) ในการสร้างเพื่อหา $C(A,W)$ ที่เหมาะสมที่สุด เพราะว่า Binary Tree นั้นมีการทำงานและการค้นหาได้รวดเร็ว

2.4 วิธีการเข้าถึงข้อมูลในรูปแบบของโครงสร้างต้นไม้

การเข้าถึงข้อมูลในรูปแบบของโครงสร้างแบบต้นไม้ วิธีที่นิยมใช้ส่วนใหญ่มี 2 วิธี คือ การค้นหาแบบลึกก่อน (Depth First Search: DFS) และ การค้นหาแบบกว้างก่อน (Breadth First Search: BFS)

2.4.1 การค้นหาแบบลึกก่อน (Depth First Search: DFS)

การค้นหาแบบลึกก่อน (Anany, 2007) เป็นการค้นหาในแนวลึกก่อน โดยจะทำการค้นหาจากโครงสร้างต้นไม้ โดยเริ่มต้นจากโหนดราก (Root node) ที่อยู่บนสุดแล้วท่องลงมาในระดับลึกสุดจนถึงโหนดล่างสุด (Leaf node) จากนั้นให้ย้อนกลับมาที่จุดสูงสุดของกิ่งเดียวกัน ถ้ายังมีกิ่งที่แยกออกไปและยังไม่ได้เดิน ก็ให้เริ่มเดินไปจนถึงระดับล่างสุดอีก ทำเช่นนี้ไปเรื่อยๆ จบพบโหนดที่ต้องการหรือสำรวจครบทุกโหนด กำหนดให้หมายเลขที่อยู่ในโหนดเป็นลำดับการเดิน ซึ่งสามารถแสดงได้ดังภาพประกอบ 2-9

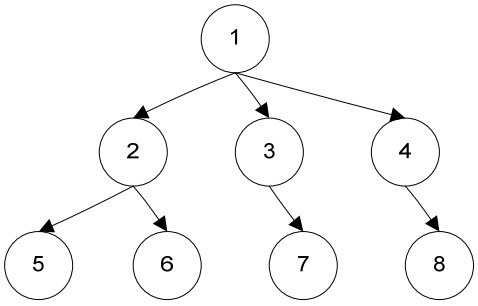


ภาพประกอบ 2-9 ลำดับการเดินแบบ DFS

การค้นหาแบบลึกก่อนสามารถนำมาใช้กับข้อมูลที่มีลักษณะโครงสร้างแบบกราฟได้ด้วย แต่สำหรับการเดินบนกราฟนั้นจะไม่มี โหนดในระดับลึกสุดดังนั้นการเดินบนกราฟจึงมีวิธีการเดินใหม่ คือ เริ่มจากโหนดเริ่มต้น (จะต้องมีการกำหนดไว้ก่อน) จากนั้นให้นำโหนดที่ติดกันกับโหนดปัจจุบัน โดยเก็บโหนดที่ยังไม่ได้เดินผ่านและยังไม่มีใน Stack ไว้ เมื่อหมดแล้วให้ pop ตัวบนสุดออกมาสำรวจ แล้วนำโหนดที่อยู่ข้างเคียงทั้งหมดที่ยังไม่ได้เดินผ่านมาต่อท้าย Stack แล้ว pop ตัวบนสุดออกมา ทำเช่นนี้ไปเรื่อยๆจนพบโหนดที่ต้องการ หรือเดินหมดทุกโหนดในกราฟแล้ว

2.3.4 การค้นหาแบบกว้างก่อน (Breadth First Search: BFS)

การค้นหาแบบกว้างก่อน (Anany, 2007) เป็นการค้นหาทีละระดับของโครงสร้างแบบต้นไม้โดยเริ่มจากโหนดราก (ระดับที่ 0) แล้วลงมาที่ระดับที่ 1 จากซ้ายไปขวาเสร็จแล้วไประดับที่ 2 จากซ้ายไปขวาเช่นกัน ทำเช่นนี้ไปเรื่อยๆ จนพบโหนดที่ต้องการ กำหนดให้หมายเลขที่อยู่ในโหนดเป็นลำดับการเดิน ซึ่งแสดงได้ดังภาพประกอบ 2-10



ภาพประกอบ 2-10 ลำดับการเดินแบบ BFS

การค้นหาแบบกว้างก่อน สามารถนำมาใช้กับโครงสร้างข้อมูลแบบกราฟได้ โดยจะอาศัยโครงสร้างข้อมูลแบบคิว(Queue) มาช่วย การท่องกราฟจะเริ่มจากสำรวจที่โหนดเริ่มต้น แล้วนำโหนดข้างเคียงมาเก็บไว้ในคิว เมื่อสำรวจโหนดเริ่มต้นเสร็จ ให้นำข้อมูลในคิวออกมาสำรวจ แล้วนำโหนดข้างเคียงที่ยังไม่ได้สำรวจและไม่ได้อยู่ในคิวใส่คิวไว้ ทำเช่นนี้ไปเรื่อย ๆ จนพบโหนดที่ต้องการ หรือเมื่อสำรวจครบทุกโหนด

ในตารางที่ 2-4 แสดงการเปรียบเทียบ การค้นหาแบบลึกก่อนและแบบกว้างก่อน

ตารางที่ 2-4 เปรียบเทียบ DFS และ BFS

DFS	BFS
ใช้หน่วยความจำน้อยกว่า เพราะจะเก็บสถานะในเส้นทางค้นหาปัจจุบันไว้เท่านั้น เมื่อไปเส้นทางอื่นเส้นทางที่ผ่านมาก็ไม่จำเป็นต้องเก็บ	ใช้หน่วยความจำมาก เพราะต้องเก็บสถานะไว้ทุกตัวเพื่อหาเส้นทางจากสถานะเริ่มต้นไปหาคำตอบ
ถ้าคำตอบอยู่ระดับความลึกจะช่วยให้พบคำตอบโดยไม่ต้องทำการค้นหามาก	ถ้าคำตอบอยู่ระดับที่ความลึกไม่มาก ก็จะมีโอกาสจะค้นหาพบได้เร็ว
เมื่อพบคำตอบแล้วไม่รับประกันได้ว่าเส้นทางนั้นเป็นเส้นทางที่สั้นที่สุด	เมื่อพบคำตอบแล้วสามารถรับประกันได้ว่าเป็นเส้นทางที่สั้นที่สุด

2.4 การดำเนินการระดับบิต (Bitwise-Operation)

2.4.1 ตัวดำเนินการ (Operator)

ตัวดำเนินการ (Schildt, 1990) หมายถึง เครื่องหมายที่ใช้ในการกำหนดกรรมวิธีทางคณิตศาสตร์ ซึ่งเป็นการเปรียบเทียบระหว่างข้อมูล 2 ตัว ที่เรียกว่าตัวถูกดำเนินการ (Operand) โดยอาจมีค่าเป็นตัวเลข ข้อความ ค่าคงที่ หรือตัวแปรต่างๆ เป็นต้น

2.4.2 ตัวดำเนินการระดับบิต (Logical Operator)

ตัวดำเนินการระดับบิต เป็นการดำเนินการเชิงตรรกะในระดับบิต โดยจะใช้มุมมองในแบบเลขฐานสองมาจัดการกับข้อมูล นั่นคือข้อมูลตัวเลขนั้นจะถูกแปลงเป็นเลขฐานสองในหน่วยความจำในขณะที่มีการดำเนินการเชิงตรรกะในระดับบิต ซึ่งแต่ละตำแหน่งของบิตมีค่าที่เป็นไปได้อยู่สองค่า คือ หากตำแหน่งบิตนั้นมีค่าเป็น 1 จะหมายถึงมีค่าเป็นจริง และหากตำแหน่งบิตนั้นมีค่าเป็น 0 จะหมายถึงมีค่าเป็นเท็จ โดยหากกำหนดตัวแปร *x* มีค่าเป็น 01001 และตัวแปร *y* มีค่าเป็น 10111 สามารถอธิบายการทำงานของตัวดำเนินการระดับบิตได้ดังนี้

1) ตัวดำเนินการระดับบิต AND (Bitwise AND Operator) เขียนแทนด้วยเครื่องหมาย “*x* AND *y*” หมายถึง การเทียบบิตแบบ AND ระหว่าง *x* กับ *y* จะได้ค่าเป็น 1 ถ้า *x* และ *y* ไม่เป็น 0 ทั้งคู่ และจะได้ค่าเป็น 0 กรณีเดียวคือ ทั้ง *x* และ *y* ต้องเป็น 0 ทั้งคู่ ตัวอย่างเช่น “01001 & 10111” ให้ผลลัพธ์เป็น 00001

2) ตัวดำเนินการระดับบิต OR (Bitwise Inclusive OR Operator) เขียนแทนด้วยเครื่องหมาย “*x* OR *y*” หมายถึง การเทียบบิตแบบ OR ระหว่าง *x* กับ *y* จะได้ค่า 1 ถ้า x และ *y* ต้องเป็น 1 ทั้งคู่ และจะได้ค่า 0 ก็ต่อเมื่อ *x* หรือ *y* มีค่าเป็น 0 ตัวอย่างเช่น “01001 OR 10111” ให้ผลลัพธ์เป็น 11111

3) ตัวดำเนินการระดับบิต XOR (Bitwise Exclusive OR Operator) เขียนแทนด้วยเครื่องหมาย “*x* XOR *y*” หมายถึง การเทียบบิตแบบ XOR ระหว่าง *x* กับ *y* จะได้ค่า 1 เมื่อ *x* และ *y* ต้องมีค่าเป็น 1 หรือ 0 เหมือนกัน แต่ถ้าค่าต่างกันจะได้ค่าเป็น 0 ตัวอย่างเช่น “01001 XOR 10111” ให้ผลลัพธ์ 11110

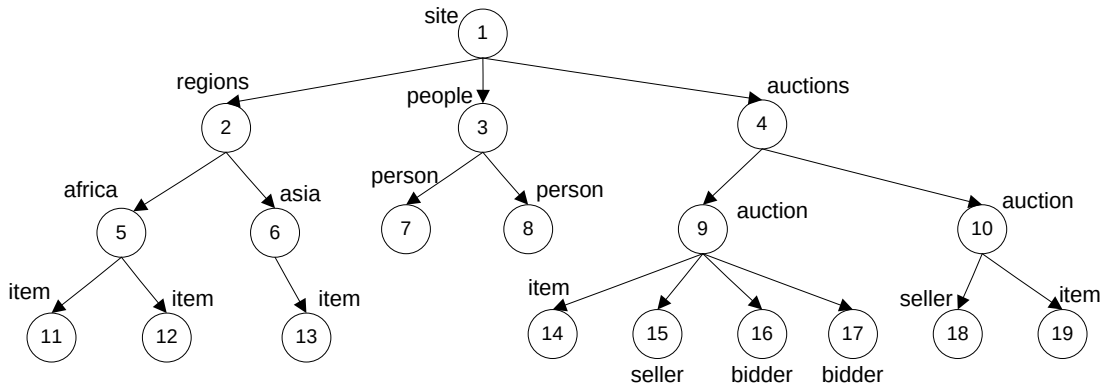
4) ตัวดำเนินการระดับบิต NOT (Bitwise NOT Operator) เขียนแทนด้วยเครื่องหมาย “~*x*” หมายถึง การสลับค่าบิตให้ *x* จะให้ผลลัพธ์ของบิตมีค่าตรงข้ามจากค่าเดิม ตัวอย่างเช่น ~01001 กลายเป็น 10110

บทที่ 3

เทคนิคการทำดัชนีสำหรับเอกสาร XML ที่เก็บในรูปแบบ Files

ในงานวิจัยที่ผ่านมามีการนำเสนอเทคนิคการทำดัชนีสำหรับเอกสาร XML ที่จัดเก็บอยู่ในรูปแบบ Files ไว้หลายวิธี ซึ่งสามารถสรุปได้เป็น 2 แบบ คือ 1) สร้างดัชนีโดยเก็บเป็นข้อความ (String) ซึ่งใช้การเปรียบเทียบแบบ String Matching และ 2) สร้างดัชนีโดยเก็บในรูปแบบของบิต (Signature) และใช้การดำเนินการในระดับบิต (Bitwise Operation) เพื่อความสะดวกในการอธิบายต่อไป จึงมีการกำหนดความหมายของคำศัพท์ดังต่อไปนี้

Label Path: สมมุติให้ *o* เป็นโหนดในต้นไม้ จะได้ว่า label path ของโหนด *o* คือ ลำดับของ labels ตั้งแต่ 1 ขึ้นไป โดยจะแบ่งแยกแต่ละ label ด้วย "." หรือถ้าเป็นในการสอบถามของ XML แยกแต่ละ label ด้วย "/" หรือ "//" จากภาพประกอบ 3-1 สามารถแสดงตัวอย่าง label path คือ africa.item (africa/item) และ asia.item (asia/item) เป็น label path ของโหนดที่ 2 (regions)

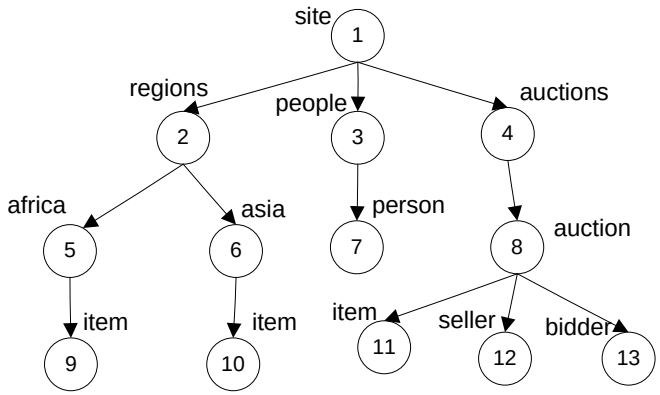


ภาพประกอบ 3-1 XML Tree

3.1 DataGuide

เทคนิคการทำดัชนีแบบ DataGuide (Goldman and Widom, 1997) เป็นการทำดัชนีในรูปของแบบที่เรียกว่า label path ในการค้นหาใช้การเปรียบเทียบแบบ path matching จะไม่ใช้การเปรียบเทียบแบบ keyword matching ในการสร้างดัชนีนั้นจะเก็บ path ทั้งหมดในแต่ละโหนดไว้และ path ที่ซ้ำกันจะเก็บเพียงครั้งเดียว DataGuide

จากภาพประกอบ 3-2 แสดง DataGuide ของ XML Tree ในภาพประกอบ 3-1 ตัวอย่างเช่น ต้องการค้นหา /site/people/person ซึ่งอยู่ในตำแหน่งที่ 1 3 และ 7 ตามลำดับ เมื่อค้นหาไปถึงโหนดที่ 7 แล้วจะพบว่าในโหนดที่ 7 นี้จะอ้างถึง โหนดที่ 7 และ 8 ใน XML Tree ของภาพประกอบ 3-1 ทำให้สามารถได้คำตอบทั้งหมดที่ต้องการ



ภาพประกอบ 3-2 DataGuide

ข้อดีของ DataGuide

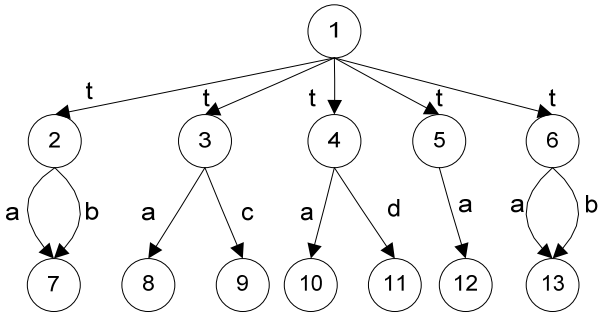
1. ช่วยลดเวลาการเข้าถึงโหนด เพราะโครงสร้างของ DataGuide จะมีจำนวนโหนดน้อยกว่าโครงสร้างของ XML Tree
2. สร้างได้ง่ายและครอบคลุมทุกการสอบถามเพราะยังเก็บข้อมูลทั้งหมดไว้

ข้อเสียของ DataGuide

1. การค้นหาจำเป็นต้องเริ่มจากโหนดรากเสมอ ซึ่งใช้เวลามากทำให้ไม่มีประสิทธิภาพ
2. ใช้พื้นที่ในการเก็บข้อมูลมาก

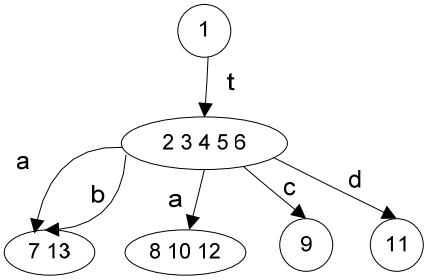
3.2 1-Index

เทคนิคการทำดัชนีแบบ 1-Index (Milo and Suciu, 1999) เป็นการรวมกลุ่มโหนดที่มีเส้นทางเหมือนกันเข้าไว้ด้วยกันเพื่อลดพื้นที่ในการจัดเก็บดัชนีแต่ยังมีปัญหาคล้าย ๆกับ DataGuide เพื่อให้ง่ายสำหรับการอธิบาย จะสร้าง XML Tree ใหม่แสดงในภาพประกอบ 3-3 เนื่องจาก XML Tree ในภาพประกอบ 3-1 ซับซ้อนเกินไป



ภาพประกอบ 3-3 XML Tree

จากภาพประกอบ 3-3 แสดง XML Tree ที่สร้างขึ้นใหม่ และในภาพประกอบ 3-4 แสดง 1-Index ที่สร้างสำหรับ XML Tree ในภาพประกอบ 3-3 โดยเส้นทางที่เหมือนกันจะไว้ด้วยกัน เช่น จากโหนดที่ 1 มี t ที่เดินไปยังโหนดที่ 2 3 4 5 และ 6 จึงนำมารวมไว้ด้วยกันที่โหนดเดียว



ภาพประกอบ 3-4 1-Index

ข้อดีของ 1-Index

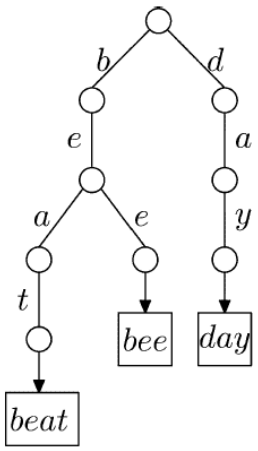
- ลดพื้นที่การจัดเก็บดัชนีได้ดีกว่า DataGuide

ข้อเสียของ 1-Index

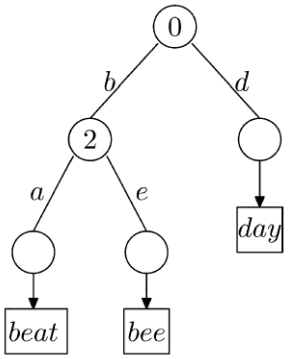
- การค้นหาข้อมูลต้องเริ่มจากโหนดรากเหมือน DataGuide ยังต้องใช้เวลามาก จึงไม่มีประสิทธิภาพ

3.3 Index Fabric

วิธีนี้นำแนวคิดของ Patricia Tree มาช่วยสำหรับทำดัชนี ซึ่งแนวคิดของ Patricia Tree ในภาพประกอบ 3-5 แสดงตัวอย่างของดัชนี ที่มีคำว่า beat bee และ day หลังจากใช้แนวคิดของ Patricia Tree แสดงในภาพประกอบ 3-6 ซึ่งช่วยลดโครงสร้างต้นไม้ได้ ทำให้ต้นไม้มีความลึก (Depth) น้อยลงกว่าเดิม สำหรับการค้นหาข้อมูลจะใช้ตำแหน่งตัวอักษรเป็นคีย์ในการค้นหา



ภาพประกอบ 3-5 ตัวอย่างของดัชนีของคำว่า beat bee และ day



ภาพประกอบ 3-6 Patricia Trie

เนื่องจากการสอบถามในเอกสาร XML เป็นแบบข้อความที่เป็น path สำหรับการนำมาใช้นั้น ต้องมีการสร้างคล้ายๆกับตาราง Mapping ซึ่งจะเรียกว่า designator dictionary เพื่อทำการกำหนดค่ากับแต่ละโหนดที่แตกต่างกัน จากภาพประกอบ 3-1 นำมาสร้างเป็น designator dictionary ได้ดังภาพประกอบ 3-7

Tagname	Designator
site	α
regions	θ
africa	β
asia	γ
item	δ
people	η
person	π
auctions	υ
auction	χ
seller	μ
bidder	ψ

ภาพประกอบ 3-7 designator dictionary

ในการสร้าง Index Fabric (Cooper et al., 2001) จะนำแนวคิดพื้นฐานของ Patricia Trie มาสร้างเป็น sub-tree ย่อยๆ โดยแยกเป็น layer ตามแนว horizontal ถ้าใน layer

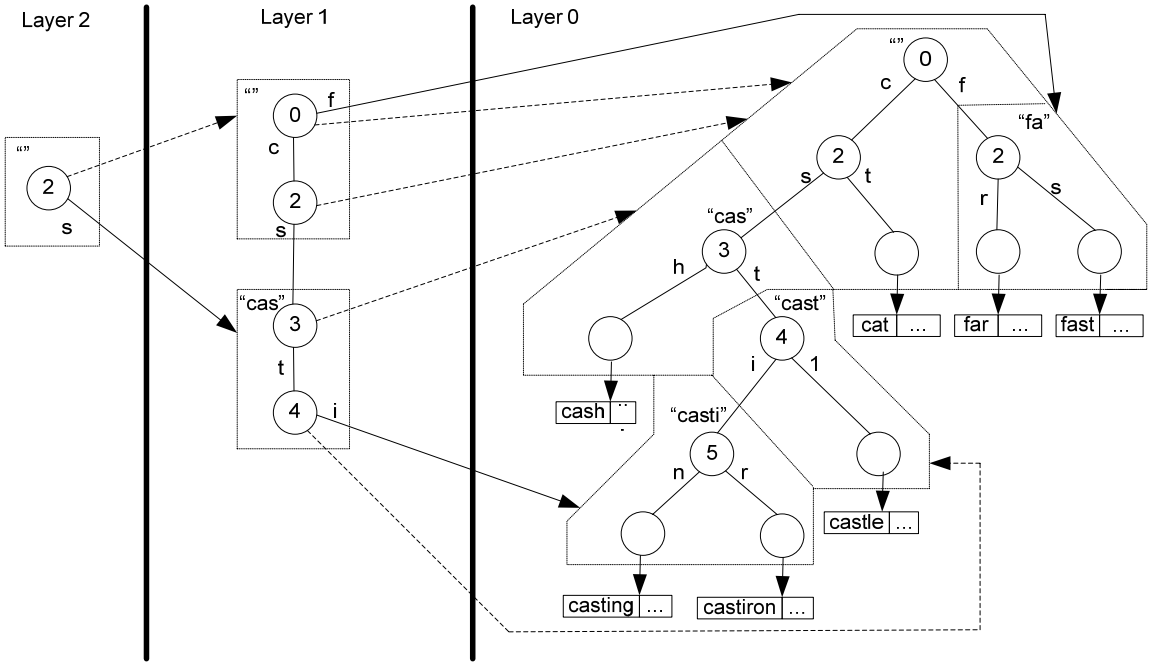
ทางขวายังมีข้อมูลมากจะสร้างอีก layer เพื่อให้ข้อมูลน้อยลง จากภาพประกอบ 3-1 นำมาสร้างดัชนีแบบ Index Fabric ได้ดังภาพประกอบ 3-8 ซึ่งจะได้ Index Fabric ที่มี 3 layer

สำหรับการค้นหาข้อมูล จะเริ่มจากโหนดรากที่อยู่ layer ทางซ้ายสุดก่อนแล้วตรวจสอบไปเรื่อยๆ ตัวอย่างเช่น นำการสอบถามของเอกสาร XML มาแปลงเป็นสัญลักษณ์ตาม designator dictionary ได้ดังนี้

การสอบถาม : /site/people/person

แปลงเป็น : $\alpha\eta\pi$

เริ่มค้นหาจาก layer ที่ 2 ดูว่าจะต้องเริ่มตรวจสอบที่ตำแหน่งที่เท่าไร ดูจากเลขที่อยู่ในโหนด จากการสอบถามมีตัวแรกเป็น α ซึ่งไม่ใช่ θ จึงเดินไปตามทางเส้นประใน layer ที่ 1 จากนั้นก็ทำเช่นนี้ไปเรื่อยๆ จนถึง sub-tree ที่ต้องการ



ภาพประกอบ 3-8 Index Fabric

ข้อดีของ Index Fabric

1. ช่วยเพิ่มประสิทธิภาพสำหรับการสอบถามที่อยู่รูปแบบแกน parent-child เท่านั้น
2. ช่วยลดเวลาในการประมวลผลการสอบถามได้ดีกว่า DataGuide

ข้อเสียของ Index Fabric

1. ใช้พื้นที่ในการเก็บข้อมูลมากเพราะต้องแบ่ง layer
2. เสียเวลาในการแปลงการสอบถามและคำตอบ

ในตารางที่ 3-1 แสดงการเปรียบเทียบดัชนีทั้ง 4 แบบสำหรับการค้นหาข้อมูลในเอกสาร XML โดยกำหนดให้ n เป็นจำนวนโหนดทั้งหมด และให้ m เป็นจำนวนเส้นทางระหว่างโหนด (Edges) ทั้งหมด

ตารางที่ 3-1 เปรียบเทียบดัชนี 3 แบบ

ดัชนี	พื้นที่สำหรับเก็บดัชนี	เวลาในการค้นหาข้อมูล (ลักษณะการเปรียบเทียบ)
DataGuide	$Exp(m)$	String matching (ทุกโหนด)
1-Index	m	String matching (ทุกโหนด)
Index Fabric	n	String matching (บางโหนด)

จากตารางที่ 3-1 เทคนิคการทำดัชนีทั้ง 3 แบบจะใช้การค้นหาข้อมูลแบบ String matching ซึ่งทำให้เสียเวลาในการประมวลผล จึงได้มีงานวิจัยได้พัฒนาเพื่อเพิ่มประสิทธิภาพการค้นหาข้อมูลโดยการแทนข้อมูลแต่ละโหนดด้วย Bit String เรียกว่า Signature เพราะสามารถดำเนินการระบิต เช่น AND หรือ OR ได้ทำให้การค้นหาข้อมูลมีประสิทธิภาพมากขึ้น

3.5 แนวคิดของดัชนีแบบ Signature

การสร้างดัชนีแบบ Signature เป็นการแทนค่าแต่ละโหนด (อิลิเมนต์ แอททริบิวต์ และ Value) ของโครงสร้างต้นไม้ด้วย Bit String การทำดัชนีแบบ Signature ถูกนำมาใช้อย่าง

แพร่หลายสำหรับการค้นหาข้อมูลในเอกสาร XML ซึ่งมีแนวคิดในการสร้าง คือ 1) สร้าง Mapping Table 2) นำค่าที่ได้จาก Mapping Table มาสร้าง Signature แสดงได้ดังนี้

1) สร้าง Mapping Table

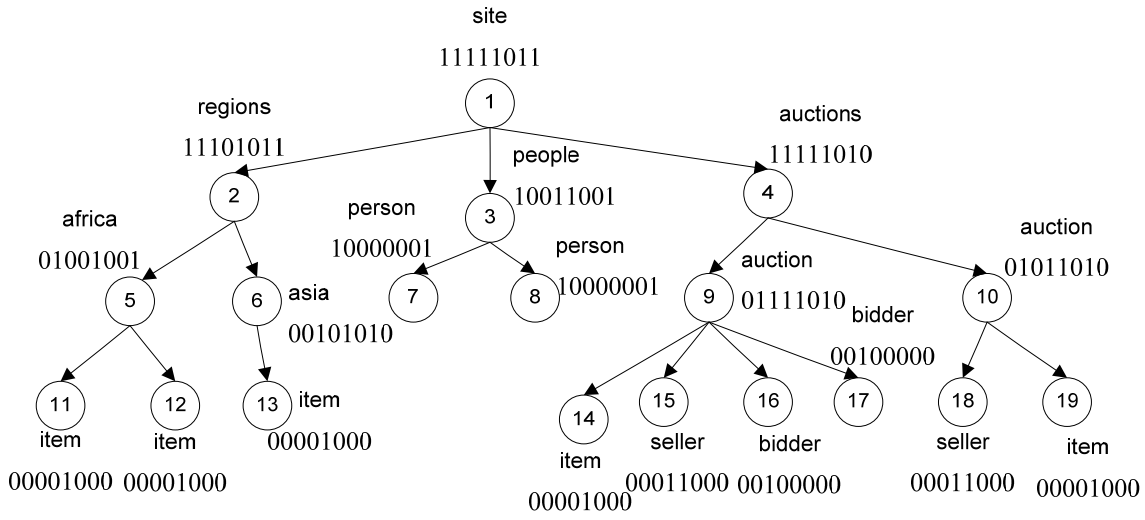
จากภาพประกอบ 3-1 นำมาสร้าง Mapping Table แสดงในตารางที่ 3-1 ซึ่งสร้างด้วย Hash Function ในการสร้าง Hash Function นั้นอาจจะมีลักษณะที่แตกต่างกัน

2) สร้าง Signature

สำหรับการสร้าง Signature จะนำรหัสที่ได้จาก Mapping Table มาดำเนินการตรรกะ OR ระหว่างรหัสประจำโหนดกับรหัสของโหนดลูกหลานทั้งหมด ในภาพประกอบ 3-9 แสดง Signature ของภาพประกอบ 3-1 ตัวอย่างเช่น โหนด Africa มีค่ารหัสจาก Mapping Table เป็น 01000001 และมีโหนดลูกสองโหนดคือ โหนด item มีค่ารหัสจาก Mapping Table เป็น 00001000 ดังนั้น Signature ของโหนด africa มีค่าเป็น 01001001

ตารางที่ 3-2 Mapping Table

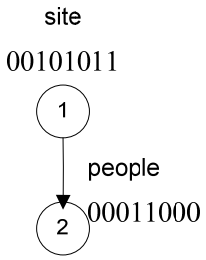
ชื่อโหนด	รหัส
site	00000011
regions	10001000
africa	01000001
asia	00100010
item	00001000
people	00011000
person	10000001
auctions	11000000
auction	01000010
seller	00011000
bidder	00100000



ภาพประกอบ 3-9 Signature

สำหรับการค้นหาข้อมูลนั้นจะใช้วิธีแบบ DFS (ซึ่งได้กล่าวแล้วในบทที่ 3) ในการเปรียบเทียบใช้การดำเนินการตรรกะ AND ขั้นตอนการค้นหามีดังนี้

1. กำหนดให้ การสอบถามเป็น /site/people จะต้องแปลงการสอบถามให้เป็นต้นไม้การสอบถาม (Query Tree) ได้ดังภาพประกอบ 3-10



ภาพประกอบ 3-10 ต้นไม้การสอบถาม (Query tree)

2. จากนั้นเริ่มทำการค้นหาที่โหนดแรกของต้นไม้การสอบถาม กับโหนดแรกของต้นไม้ในภาพประกอบ 3-10 โดยนำ Signature มาดำเนินการกตรรกะ AND ได้ดังนี้ 00101011 AND 11111011 = 00101011 ซึ่งผลลัพธ์ที่ได้มีค่าเท่ากับ Signature ของต้นไม้การสอบถามแสดง

ว่าใช่โหนดที่ต้องการ ต่อมาเลื่อนมาที่โหนดลูกของต้นไม้การสอบถามคือ โหนด people มี Signature เป็น 00101000 และเลื่อนไปที่โหนดลูกโหนดแรกในต้นไม้ในภาพประกอบ 3-9 คือ โหนด regions มี Signature เป็น 11101011 และนำ Signature มาดำเนินการตรรกะ AND กัน ผลลัพธ์ที่ได้คือ 00001000 (00011000 AND 11101011) ซึ่งไม่เท่ากับ Signature ของโหนดในต้นไม้การสอบถาม จึงไม่จำเป็นต้องค้นหาโหนดลูกของต้นไม้ในภาพประกอบ 3-10 แต่จะเลื่อนไปยังโหนดถัดไปทางขวาคือ โหนด people มี Signature เป็น 10011001 นำมาดำเนินการตรรกะ AND กับโหนดเดิมในต้นไม้การสอบถามต่อ ผลลัพธ์ที่ได้ คือ 00011000 (00011000 AND 10011001) ซึ่งเท่ากับ Signature ของโหนดที่กำลังหาอยู่

มีงานวิจัยที่ได้นำเสนอและพัฒนาประสิทธิภาพของการทำดัชนีสำหรับเอกสาร XML ที่อยู่บนหลักการของดัชนีแบบ Signature เช่น

งานวิจัยของ Park และคณะ ในปี 2001

ใช้ Hash Function ในการสร้าง Mapping Table กับทุกโหนด และการสร้าง Signature ใช้การดำเนินการทางตรรกะ OR ค่าจาก Mapping Table ของตัวเองกับโหนดลูกหลาน

ข้อดี

(1) ในการค้นหาข้อมูลลดการเข้าถึงโหนดที่ไม่จำเป็นได้

ข้อเสีย

(1) ใช้พื้นที่ในการเก็บดัชนีมาก

(2) การเปรียบเทียบใช้ทุกบิต

งานวิจัยของ Chung และ คณะ ในปี 2003

ใช้ Hash Function ในการสร้าง Mapping Table แต่การสร้าง Signature นั้นมี 2 แบบ คือ 1) Horizontal Signature (HS) เป็น Signature แต่ละระดับชั้นเกิดจากการใช้ การดำเนินการตรรกะ OR เช่นกัน 2) Vertical Signature (VS) เป็น Signature แต่ละ path

คำนิยามของ Horizontal Signature และ Vertical Signature มีดังนี้ กำหนดให้ *h* เป็นความสูงของต้นไม้ ให้โหนดรากเป็นระดับที่ 1 (1st-level) และมีโหนดใบ (Leaf node) เป็นระดับที่ h (*h*th-level) และให้ *D* แทนเอกสาร XML จะได้ว่า $N_D(a,b)$ แสดงถึง โหนดที่ b ในระดับที่ a ดังนั้นจะได้ว่า $N_D(h,n^{(h-1)})$ เป็นโหนดใบทางขวาสุด

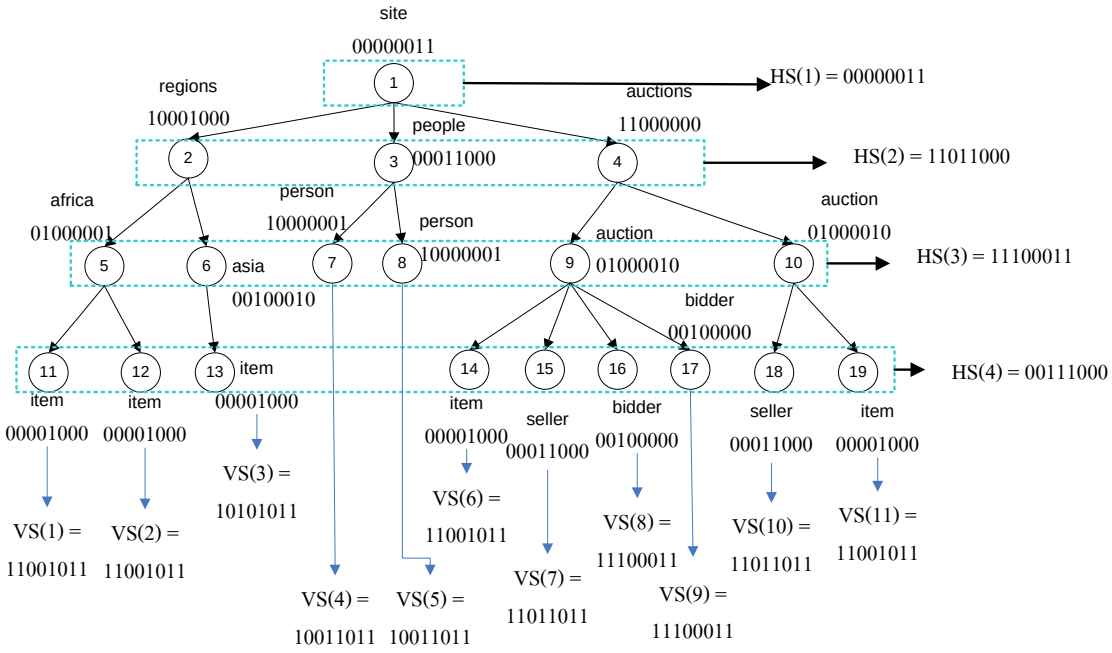
คำนิยาม Horizontal Signature (HS) : กำหนดให้ *i* เป็นระดับตามแนว Horizontal ให้ *D* เป็นเอกสาร XML จะได้ว่า $HS_D(i)$ แสดงถึง signature ในระดับที่ *i* ตามแนว Horizontal และกำหนดให้ "|" แทนการดำเนินตรรกะ OR เขียนเป็นสูตรได้ดังนี้

$$HS_D(i) = S_D(i,1) \mid S_D(i,2) \mid ... \mid S_D(i,n^{(i-1)})$$

คำนิยาม Vertical Signature (VS) : กำหนดให้ *k* เป็นลำดับที่ของเส้นทาง (Path) จะได้ว่า $VS_D(k)$ เป็น Signature ตามแนว Vertical ของเส้นทางลำดับที่ *k* กำหนดให้ *w1* และ *w2* เป็นตำแหน่งของโหนดในเส้นทางที่ k สามารถเขียนเป็นสูตรได้ดังนี้

$$VS_D(k) = S_D(1,w1) \mid S_D(2,w2) \mid ... \mid S_D(h,k)$$

ในภาพประกอบ 3-11 แสดง Horizontal Signature และ Vertical Signature ของ XML Tree ในภาพประกอบ 3-1 สำหรับการค้นหาข้อมูล นำการสอบถามแปลงเป็นต้นไม้การสอบถามในลักษณะเดียวกับต้นไม้ที่ทำเป็นดัชนี จากนั้นเริ่มค้นหาตามแนว Horizontal ครบทุกโหนด แล้วค่อยค้นหาตามแนว Vertical โดยใช้การดำเนินตรรกะ AND



ภาพประกอบ 3-11 Horizontal Signature และ Vertical Signature ของ XML Tree ใน

ข้อดี

(1) ในการค้นหาข้อมูลลดการเข้าถึงโหนดที่ไม่จำเป็นได้

(2) ลดพื้นที่ในการเก็บ

ข้อเสีย

(3) ความยาวของ Signature ที่ได้ยาวมาก

(4) การเปรียบเทียบต้องใช้ทุกบิต

งาน Huang และคณะในปี 2008

งานวิจัยนี้มีการสร้างดัชนี 3 ดัชนี คือ 1) Storage Tree ในแต่ละโหนดจะประกอบด้วย Signature และค่า containing code (ลำดับการท่องต้นไม้แบบ DFS ประกอบด้วย start และ end ซึ่งเป็นลำดับเริ่มต้นและสิ้นสุดในการท่องผ่านแต่ละโหนด) ตัวอย่างเช่น จากภาพประกอบ 3-2 โหนดที่ 2 มี containing code เป็น (2,10) และ โหนดที่ 5 มี containing code

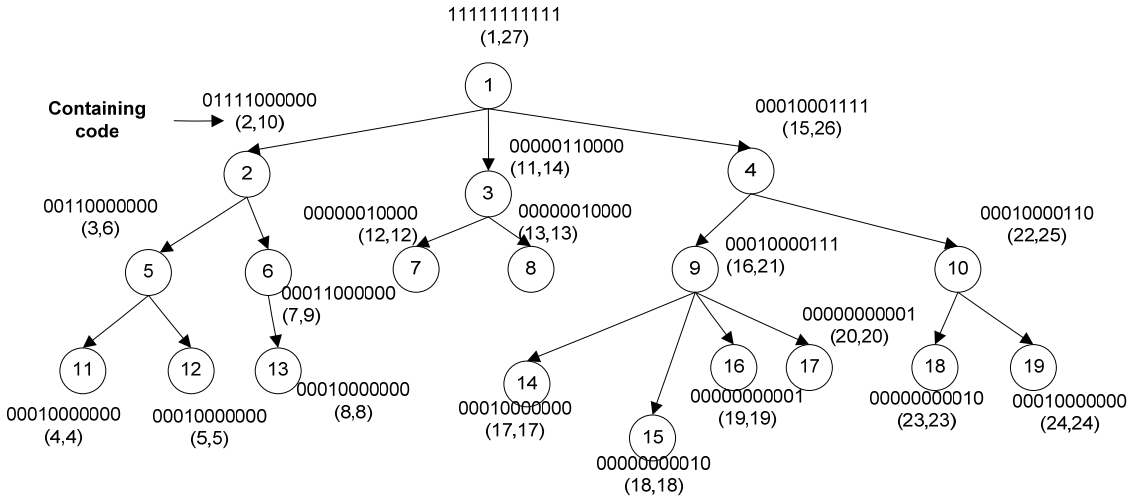
เป็น (3,6) เป็นต้น ซึ่งนำมาใช้สำหรับค้นหาการสอบถามตามแกน parent-child 2) TagIndex สร้างด้วยขั้นตอนวิธีแบบ B+ tree และ 3) ValueIndex สร้างด้วยขั้นตอนวิธีแบบ B+ tree

สำหรับแนวคิดการสร้างรหัสใน Mapping Table เฉพาะโหนดที่เป็นอิลิเมนต์และแอททริบิวต์เท่านั้น ซึ่งแนวคิดการสร้าง Mapping Table ใช้ความยาวของ Signature เท่ากับจำนวนโหนดที่เป็นอิลิเมนต์และแอททริบิวต์ที่แตกต่างกันและกำหนดให้บิต 1 มีเพียง 1 ตำแหน่งเท่านั้นในค่าของ Signature และอยู่ในตำแหน่งที่ไม่ซ้ำกันเป็นลำดับตามการท่องต้นไม้แบบ DFS (Depth First Search) ซึ่งในตารางที่ 3-3 แสดงรหัสในตาราง Mapping Table ของ XML Tree ในภาพประกอบ 3-1

ตารางที่ 3-3 รหัสใน Mapping Table

ชื่อโหนด	รหัส
site	10000000000
regions	01000000000
africa	00100000000
item	00010000000
asia	00001000000
people	00000100000
person	00000010000
auctions	00000001000
auction	00000000100
seller	00000000010
bidder	00000000001

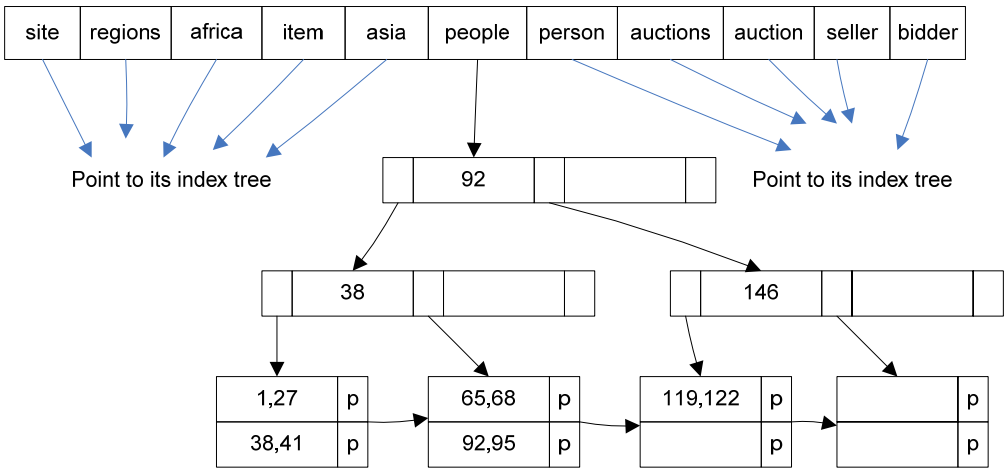
การสร้าง Signature คือ รหัสที่เกิดจากการดำเนินการตรรกะ OR ค่ารหัสจากใน Mapping Table ของแต่ละโหนดกับโหนดลูกหลานที่มีทั้งหมด จะได้ Storage Tree ดังแสดงในภาพประกอบ 3-12



ภาพประกอบ 3-12 Storage Tree

ส่วน TagIndex ใช้ขั้นตอนวิธีแบบ B+ tree สร้างโดยนำโหนดทั้งหมดที่เป็นอิลิเมนต์และแอททริบิวต์มาสร้าง ใช้สำหรับค้นหาการสอบถามที่มีลักษณะแบบ anscestor-descendant ซึ่งโหนดภายในของต้นไม้มีโครงสร้างเป็น [start] และโหนดภายนอก มีโครงสร้างประกอบด้วย [start, end, pointer] โดยที่ pointer เป็นตำแหน่งที่อยู่ของข้อมูล ในแต่ละเป็นอิลิเมนต์และแอททริบิวต์ที่แตกต่างกันจะมี TagIndex ที่แตกต่างกัน แสดงดังภาพประกอบ 3-13

ส่วน ValueIndex ใช้ขั้นตอนวิธีแบบ B+ tree สร้างโดยนำโหนดทั้งหมดที่เป็น Value มาใช้สร้างและใช้ containingg code ของแต่โหนดเป็นคีย์สำหรับค้นหา ดัชนีนี้ใช้สำหรับการค้นหาการสอบถามที่มีการระบุเงื่อนไขมาโดยเฉพาะ ซึ่งโครงสร้างของโหนดภายนอกประกอบด่วย [ข้อความ, start, pointer] โดยที่ pointer เป็นตำแหน่งที่อยู่ของข้อมูล



ภาพประกอบ 2-13 TagIndex ของโหนด people

ข้อดี

(1) ลดจำนวนโหนดที่ต้องเข้าถึง

(2) ช่วยให้การค้นหาเร็วขึ้นสำหรับการสอบถามเป็นไปตามแกน Ancestor – Descendant

(3) ลดความยาวของ Signature

ข้อเสีย

(1) ในการเก็บข้อมูลของ B+tree ต้องเสีย Overhead ที่ไม่จำเป็นสำหรับเก็บ Address ของข้อมูลจริง

ตารางที่ 3-4 แสดงการเปรียบเทียบประสิทธิภาพดัชนีแบบ Signature ทั้ง 3 แบบที่ได้กล่าวมา ซึ่งพบว่าดัชนีของ Huang และคณะ ใช้พื้นที่ในการเก็บดัชนีน้อยที่สุดในการเก็บดัชนีในโครงสร้างแบบต้นไม้และช่วยลดเวลาในการประมวลผลการสอบถามตามแกน Ancestor – Descendant ได้เพราะมีดัชนี TagIndex ช่วยสำหรับค้นหา

ตารางที่ 3-4 เปรียบเทียบดัชนีแบบ Signature

ดัชนี	ลักษณะการเปรียบเทียบ			
	ความยาวของ Signature	การสร้าง Mapping Table	การค้นหา (จำนวนบิตที่ใช้ในการเปรียบเทียบ)	จำนวนดัชนี
Park (Park and Kim, 2001)	จำนวนโหนดทั้งหมด	Hash Function	ทุกบิต	1
Chung (Chung et al, 2003)	จำนวนโหนดทั้งหมด	Hash Function	ทุกบิต	1
Huang (Huang and Wang, 2008)	จำนวนอิลิเมนต์ + จำนวนแอททริบิวต์	กำหนดให้บิต 1 มีค่าเพียง 1 บิต ตามลำดับที่ไม่ซ้ำกัน	ทุกบิต	3

ตารางที่ 3-5 แสดงการเปรียบเทียบดัชนี 2 แบบ คือ 1) สร้างดัชนีโดยเก็บในรูปแบบข้อความ (String) ซึ่งใช้การเปรียบเทียบแบบ String Matching และ 2) สร้างดัชนีโดยเก็บในรูปแบบของบิต (Signature) และใช้การเปรียบเทียบในระดับบิต (Bitwise Operation) ซึ่งพบว่าดัชนีแบบ Signature ใช้พื้นที่สำหรับจัดเก็บข้อมูลและเวลาในการประมวลผลการสอบถามที่น้อยกว่า

ตารางที่ 3-5 เปรียบเทียบดัชนี 2 แบบ คือ 1) สร้างดัชนีโดยเก็บในรูปแบบข้อความ (String) และ 2) ดัชนีโดยเก็บในรูปแบบของบิต (Signature)

ดัชนี	ลักษณะการเปรียบเทียบ	
	ลักษณะการเก็บข้อมูล	ลักษณะการเปรียบเทียบ
1) ดัชนีโดยเก็บในรูปแบบข้อความ (String)	String	String
2) ดัชนีโดยเก็บในรูปแบบของบิต (Signature)	บิต	Bitwise Operation

บทที่ 4

เทคนิคการทำดัชนีแบบ XTI

ในงานวิจัยที่เกี่ยวกับการทำดัชนีบนเอกสาร XML ที่ผ่านมานั้น เห็นว่ามีการพัฒนาขั้นตอนการสร้างต่อๆ กันมาเพื่อให้ได้ดัชนีที่มีประสิทธิภาพมากกว่าเดิมทั้งในเรื่องของพื้นที่ที่ใช้ในการเก็บดัชนีและเวลาที่ใช้ในการสอบถาม ดัชนีแบบ Signature ช่วยลดพื้นที่ในการจัดเก็บดัชนีและเวลาในการสอบถามข้อมูลได้ และมีผู้คิดค้นและพัฒนาประสิทธิภาพของดัชนีแบบ Signature อยู่เรื่อยๆ

ในงานวิจัยนี้ ผู้วิจัยนำเสนอเทคนิคการทำดัชนีแบบใหม่ ซึ่งใช้พื้นที่ในการจัดเก็บดัชนีและเวลาในการประมวลผลการสอบถามน้อยกว่า คือ นำแนวคิดของ Huffman Coding (Anany, 2007; Huffman, 1952) มาประยุกต์ใช้สำหรับสร้างรหัสของแต่ละโหนดใน XML Tree โดยมีความยาวของรหัสเท่ากัน และให้ทุกโหนดมีน้ำหนักเท่ากันในการค้นหา ซึ่งเรียกดัชนีนี้ว่า XTI (XML Tree Index)

4.1 การคิดค้นดัชนีแบบ XTI

จากการศึกษางานวิจัยที่ผ่านมาจะเห็นว่า เทคนิคการทำดัชนีสำหรับเอกสาร XML ที่อยู่ในรูปแบบ Files นั้น ตัวอย่างเช่น DataGuide 1-Index และ Index frabic ยังคงมีปัญหาเรื่องพื้นที่ในการจัดเก็บดัชนีเพราะเก็บข้อมูลเป็นข้อความ (String) นอกจากนี้ยังใช้เวลาในการค้นหาข้อมูลมากอีกด้วย เนื่องจากใช้การเปรียบเทียบแบบ String Matching ต่อมาจึงได้มีการคิดค้นเทคนิคการทำดัชนีแบบ Signature ซึ่งช่วยลดพื้นที่การจัดเก็บดัชนีและเวลาในการสอบถามได้ดี เพราะเก็บข้อมูลในรูปของ Bit String และใช้การดำเนินการทางตรรกะ เช่น AND หรือ OR ในการเปรียบเทียบเพื่อค้นหาข้อมูล แต่ยังมีปัญหาในเรื่องรูปแบบการลงรหัสที่ใช้แทนข้อมูลแต่ละโหนด ซึ่งในงานวิจัยที่ผ่านมายังใช้ประโยชน์แต่ละบิตไม่คุ้มค่า ดังนั้นในงานวิจัยนี้จึงได้คิดค้นดัชนีเรียกว่า XML Tree Index (XTI) ซึ่งนำหลักการของ Huffman Coding มาประยุกต์ใช้สำหรับสร้างรหัสของแต่ละโหนด ทำให้ทุกรหัสถูกใช้อย่างคุ้มค่ามากที่สุด นอกจากนี้แล้วยังมีการใช้ Start_offset และ End_offset ซึ่งเป็นตำแหน่งเริ่มต้นและสิ้นสุดของแท็กต่างๆ ในเอกสาร XML โดยจะใช้ค่านี้สำหรับการเข้าถึงข้อมูลในเอกสาร XML ทำให้ผลลัพธ์ที่ได้อยู่ในรูปแบบของเอกสาร XML สามารถนำไปใช้ประโยชน์ต่อได้

จากหลักการของ Huffman Coding จะได้สูตรทั่วไปในการคำนวณหาความยาวของรหัสได้ตามสมการที่ 1

ความยาวของรหัส $= \lceil \log_2 n \rceil$ (1)

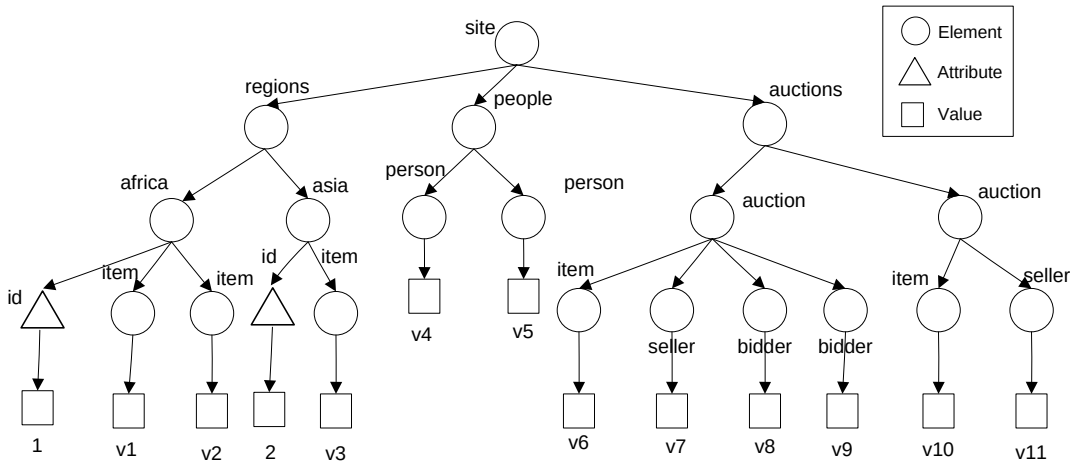
เมื่อ n แทนจำนวนของโหนดใน XML Tree ที่เป็นอิลิเมนต์และแอททริบิวต์ที่แตกต่างกัน

4.2 ขั้นตอนการสร้างดัชนีแบบ XTI

ในส่วนนี้จะใช้ตัวอย่างจากเอกสาร XML ในภาพประกอบ 4-1 ซึ่งสามารถเขียนเป็น XML Tree ได้ดังแสดงในภาพประกอบ 4-2

```
<site>
    <regions>
        <africa id = '1'>
            <item>v1</item>
            <item>v2</item>
        </africa>
        <asia id = '2'>
        <item>v3</item>
        </asia>
    </regions>
    <people>
        <person>v4</person>
        <person>v5</person>
    </people>
    <auctions>
        <auction>
            <item>v6</item>
            <seller>v7</seller>
            <bidder>v8</bidder>
            <bidder>v9</bidder>
        </auction>
        <auction>
            <item>v10</item>
            <seller>v11</seller>
        </auction>
    </auctions>
</site>
```

ภาพประกอบ 4-1 ตัวอย่างเอกสาร XML



ภาพประกอบ 4-2 ตัวอย่างXML Tree แปลงจากเอกสาร XML ในภาพประกอบ 4-1

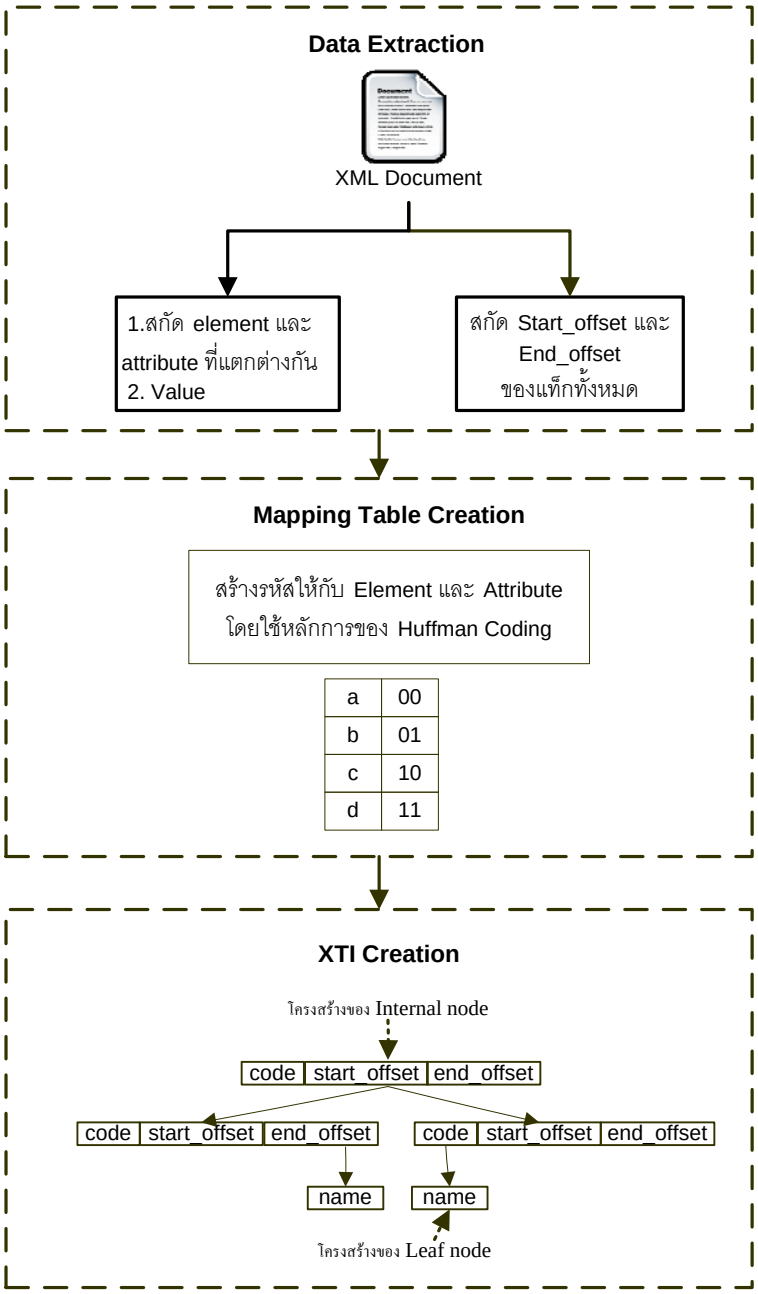
ขั้นตอนการสร้างดัชนีแบบ XTI ประกอบด้วย 3 ขั้นตอนหลัก คือ ขั้นตอนที่ 1 Data Extraction ขั้นตอนที่ 2 Mapping Table Creation และขั้นตอนที่ 3 XTI Creation แสดงดังภาพประกอบ 4-3 โดยแต่ละขั้นตอนมีรายละเอียดดังต่อไปนี้

ขั้นตอนที่ 1: Data Extraction

ในขั้นตอนนี้เป็นการอ่านเอกสาร XML เพื่อทำการสกัดข้อมูล แบ่งเป็น 2 กลุ่ม คือ 1) แท็กทั้งหมดที่แตกต่างกันในเอกสาร XML ตารางที่ 4-1 แสดงข้อมูลที่ได้จากตัวอย่าง XML Tree ซึ่งได้จำนวนของอิลิเมนต์และแอททริบิวต์ที่แตกต่างกันทั้งหมด 12 ตัว และ 2) Start_offset และ End_offset ของแท็กจากเอกสาร XML ตารางที่ 4-2 แสดง Start_offset และ End_offset ทั้งหมดที่ได้จากเอกสาร XML ในภาพประกอบ 4-1 สำหรับ XML Tree นั้นจะนำมาใช้ในกระบวนการ XTI Creation

ตารางที่ 4-1 ข้อมูลที่ได้จากตัวอย่าง XML Tree

อิลิเมนต์	site, regions, africa, asia, item, people, person, auctions, auction,
แอททริบิวต์	id
Value	v1,v2,v3,v4,v5,v6,v7,v8,v9,v10,v11,1,2



ภาพประกอบ 4-3 ขั้นตอนการสร้างดัชนีแบบ XTI

ตารางที่ 4-2 Start_offset และ End_offset

ชื่ออิลิเมนต์และแอททริบิวต์	Start_offset และ End_offset	Value	Start_offset และ End_offset
site	1,27	1	3,3
regions	2,10	v1	4,4
africa	3,6	v2	5,5
asia	7,9	2	7,7
item	4,4;5,5;17,17;24,24	v3	8,8
people	11,14	v4	12,12
id	3,3; 7,7	v5	13,13
person	12,12;13,13	v6	17,17
auctions	15,26	v7	18,18
auction	16,21;22,25	v8	19,19
seller	18,18;23,23	v9	20,20
bidder	19,19; 20,20	v10	23,23
		v11	24,24

ขั้นตอนที่ 2 : Mapping Table Creation

ในขั้นตอนนี้จะนำอิลิเมนต์และแอททริบิวต์ที่ได้จากตารางที่ 4-1 สร้างเป็นรหัสโดยใช้ Huffman Coding ซึ่งแสดงดังตารางที่ 4-3 ก่อนสร้างรหัสต้องคำนวณความยาวของรหัสได้เท่ากับ $\lceil \log_2 12 \rceil = 4$

ตารางที่ 4-3 รหัสของแต่ละอิลิเมนต์และแอททริบิวต์

ชื่ออิลิเมนต์และแอททริบิวต์	รหัส
site	0000
regions	0001
africa	0010
id	0011
item	0100
asia	0101
people	0110
person	0111
auctions	1000
auction	1001
seller	1010
bidder	1011

ขั้นตอนที่ 3: XTI Creation

ในขั้นตอนนี้จะสร้าง XML Tree Index แบบใหม่ซึ่งในแต่ละโหนดของ Tree มีโครงสร้างข้อมูลแตกต่างกันตามชนิดของโหนด นั่นคือระหว่างโหนดภายใน (Internal Node) หรือโหนดภายนอก (External Node หรือ Leaf Node)

1) กำหนดโครงสร้างข้อมูลของแต่ละ Internal Node ของ XML Tree ซึ่งแสดงในภาพประกอบ 4-4 (ก) โดยประกอบด้วย 3 fields ดังนี้

- code หมายถึง รหัสที่กำหนดให้แก่โหนดอิลิเมนต์หรือแอททริบิวต์ โดยใช้แนวคิดของ Huffman Coding ในการสร้างรหัสให้ทุกโหนดที่ลงรหัสมีน้ำหนักเท่ากัน
- start_offset หมายถึง ตำแหน่งเริ่มต้นของอิลิเมนต์หรือแอททริบิวต์
- end_offset หมายถึง ตำแหน่งสิ้นสุดของอิลิเมนต์หรือแอททริบิวต์

2) สำหรับโครงสร้างข้อมูลที่ Leaf Node ของ XML Tree จะประกอบด้วย field เดียวเท่านั้นคือ name หมายถึงค่าข้อมูล ดังแสดงโครงสร้างในภาพประกอบ 4-4 (ข) และในภาพประกอบ 4-5 แสดง XTI ที่ได้จากเอกสาร XML ในภาพประกอบ 4-2

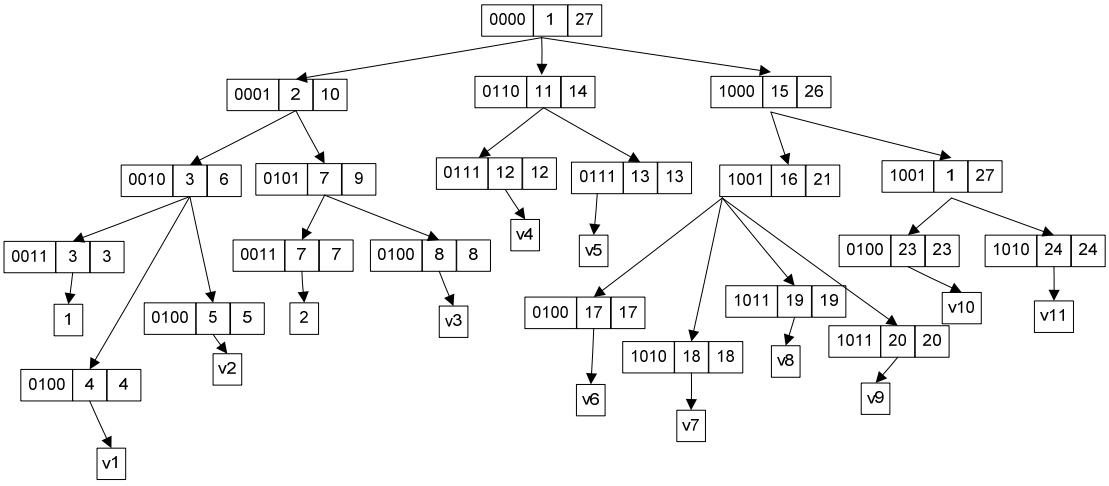
code	start_offset	end_offset

(ก) โครงสร้างของ Internal node

name

(ข) โครงสร้างของ Leaf node

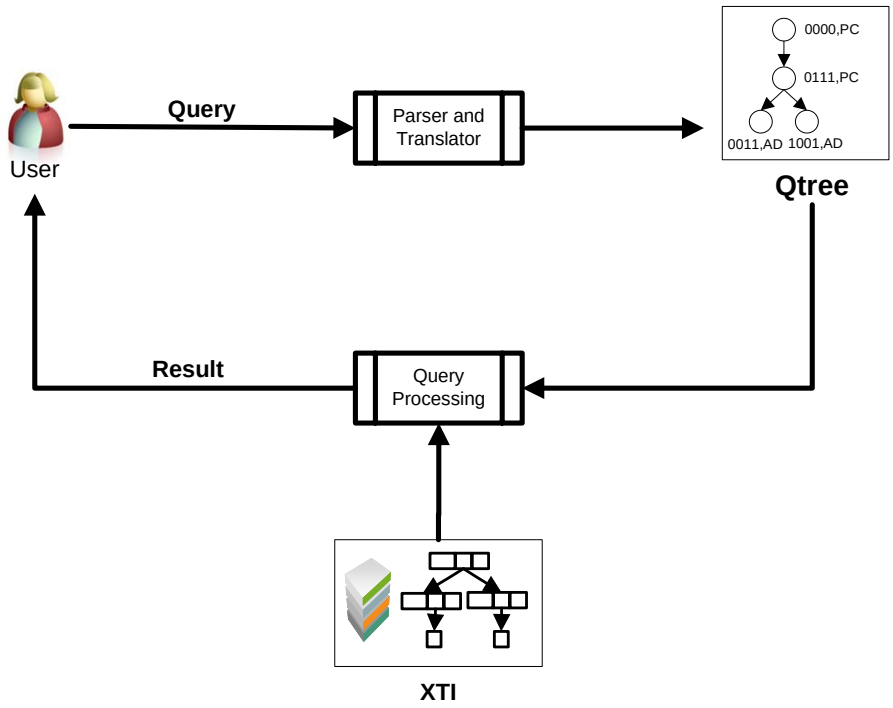
ภาพประกอบ 4-4 โครงสร้างข้อมูลของ Internal node และ Leaf node



ภาพประกอบ 4-5 XTI ของเอกสาร XML ในภาพประกอบ 4-2

4.3 การประมวลผลการสอบถาม (Querying Processing)

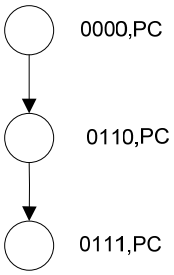
ในส่วนนี้กล่าวถึงขั้นตอนการประมวลผลการสอบถาม แสดงขั้นตอนการทำงานได้ดังภาพประกอบ 4-6 ซึ่งมีรายละเอียดดังนี้



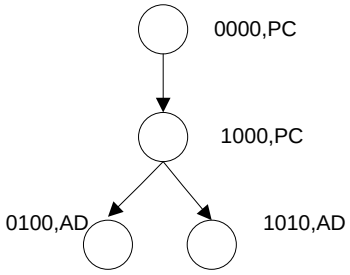
ภาพประกอบ 4-6 ขั้นตอนการประมวลผลการสอบถาม

1. แปลงการสอบถามให้อยู่ในรูปของต้นไม้การสอบถาม (Query Tree : Qtree) โดยแต่ละโหนดของ Qtree มีโครงสร้างข้อมูลที่ประกอบด้วยสอง fields ดังนี้

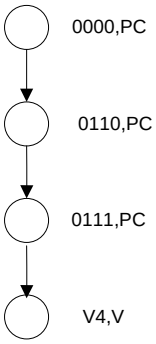
- code เป็นรหัสของโหนดอิลิเมนต์หรือแอททริบิวต์ดังตารางที่ 4-3 สำหรับโหนดที่เป็น Value นั้นตรงส่วนนี้ใช้เป็นค่าข้อมูลของแต่ละโหนด
- type เป็นชนิดแกน (Axis) ของแต่ละโหนดที่ใช้สำหรับวิเคราะห์เพื่อทำการค้นหา มี 3 แบบคือ 1) parent-child axis (PC) 2) ancestor-descendant axis (AD) และ 3) Value (V) ในภาพประกอบ 4-7 4-8 และ 4-9 แสดงQuery Tree ของการสอบถาม /site/people/person /site/auctions//item/[seller] และ /site/people/person = ‘v4’ ตามลำดับ



ภาพประกอบ 4-7 Query Tree ของการสอบถาม /site/people/person



ภาพประกอบ 4-8 Query Tree ของการสอบถาม /site/auctions//item/[seller]



ภาพประกอบ 4-9 Query Tree ของการสอบถาม /site/people/person = ‘v4’

2. ทำการค้นหา Qtree ใน XTI แบบ DFS (Depth First Search) โดยเปรียบเทียบทีละโหนดของ Qtree กับโหนดของ XTI เริ่มจากโหนดรากของทั้งสองจน Leaf Node ของ Qtree โดยใช้ขั้นตอนวิธีการค้นหาข้อมูลในภาพประกอบ 4-10 อธิบายได้ดังนี้

2.1 ตรวจสอบในแต่ละโหนดของ Qtree โดยจะตรวจสอบดูว่าฟิลด์ type มีชนิดแกนเป็นแบบไหน

- ถ้าเป็น PC ให้ทำการดำเนินการตรรกะ XOR ระหว่าง code ของ Qtree กับ code ของโหนดปัจจุบันใน XTI (ดูบรรทัดที่ 3-4)
- ถ้าเป็น AD คำนวณจำนวนโหนดลูกของโหนดใน XTI ที่โหนดปัจจุบัน จากนั้นให้ข้ามโหนดนั้นไป 1 ระดับ แล้วทำการดำเนินการตรรกะ XOR ระหว่าง code ของ Qtree กับ code ของโหนดลูกของโหนดปัจจุบันใน XTI จนหมดทุกโหนด (ดูบรรทัดที่ 5-7)
- ถ้าเป็น V ให้ทำการเปรียบเทียบกับค่าข้อมูลใน Qtree กับ ใน XTI (ดูบรรทัดที่ 8-9)

2.2 ถ้าผลลัพธ์ที่ได้มีค่าเท่ากับ code ของโหนดใน Qtree ให้ขยับไปยังโหนดลูกของ Qtree พร้อมกับขยับไปยังโหนดลูกของ XTI แล้วกลับไปทำข้อ 2.1 แต่ถ้าผลลัพธ์ที่ได้มี

ค่าไม่เท่ากับ code ของ Qtree ให้กลับไปทำข้อ 2 โดยไม่ต้องขยับโหนดใน Qtree แต่ให้ขยับโหนดใน XTI

2.3 ถ้าถึง Leaf Node ของ Qtree ให้ทำการดึงค่า start_offset และ end_offset จากโหนดใน XTI ออกมา

ตัวอย่างการสอบถามข้อมูลจากการสอบถาม /site/auctions//item/[seller] นำมาสร้างเป็น Qtree ได้ดังภาพประกอบ 4-8 และในภาพประกอบ 4-11 แสดงขั้นตอนการค้นหาข้อมูลของการสอบถามใน XTI ซึ่งสามารถอธิบายได้ดังนี้

```
QueryPrecessing
// q = Qtree, t = XTI
2 while q and t is not empty
3    Case : q.type = PC
4       FindOffset(q,t)
5    Case : q.type = AD
6       t ← getDecendant(t) //advance one level
7       FindOffset(q, t)
8    Case : q.type = V
9       FindOffset(q, t)
```

```
FindOffset(q,t)
1 if (q.code XOR t.code = 0)
2     if (q is leaf node)
3             OutputOffset(t)
4     else
5        t← get node by DFS
6        q← get node by DFS
7        QueryProcessing(q,t)
8 else
9      t← get node by BFS
10      FindOffset(q,t)
```

ภาพประกอบ 4-10 ขั้นตอนวิธีการค้นหาข้อมูล

1. เนื่องจากโหนดแรกใน Qtree มี Type เป็น PC จากนั้นนำ code ของโหนดใน Qtree ซึ่งมี code เป็น 0000 มาดำเนินการตรรกะ XOR กับโหนดแรกใน XTI ได้เลยซึ่งมี code เป็น 0000 ผลลัพธ์ที่ได้มีค่าเท่ากับ 0 แสดงว่าใช่โหนดที่ต้องการ และคำนวณจำนวนโหนดลูกของโหนดใน Qtree และ XTI ได้เป็น 1 และ 3 ตามลำดับ

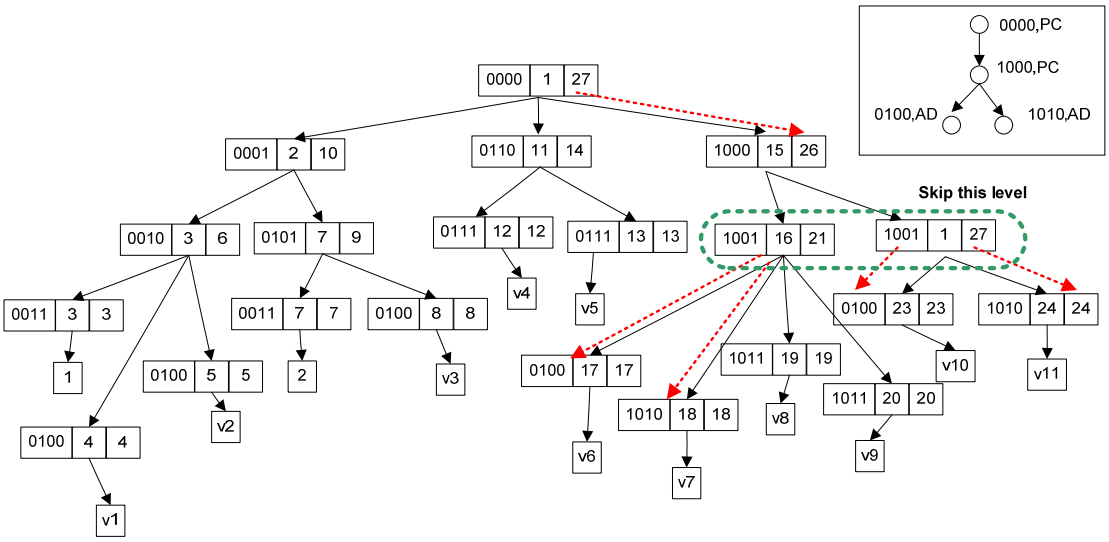
2. จากนั้นเปรียบเทียบโหนดลูกโหนดแรกของ Qtree กับโหนดลูกโหนดแรกของ XTI ซึ่งได้ผลลัพธ์เป็น 1001 (1000 XOR 0001) ซึ่งมีค่าไม่เท่ากับ 0 แสดงว่ายังไม่ใช่โหนดที่ต้องการ ในกรณีนี้ไม่ต้องขยับไปยังโหนดถัดไปใน Qtree แต่ให้ขยับไปยังโหนดถัดไปใน XTI แทน แล้วนำ code มาเปรียบเทียบกับ code ของโหนดใน Qtree โหนดเดิม ซึ่งผลลัพธ์ที่ได้เป็น 1110 (1000 XOR 0110) ซึ่งมีค่าไม่เท่ากับ 0 แสดงว่ายังไม่ใช่โหนดที่ต้องการ และให้ขยับไปยังโหนดถัดไปใน XTI อีกโหนด นำ code มาเปรียบเทียบกันใหม่ได้ผลลัพธ์เป็น 0 (1000 XOR 1000) แสดงว่าใช่โหนดที่ต้องการ จากนั้นก็ให้คำนวณจำนวนโหนดลูกของ Qtree และ XTI ได้เป็น 2 และ 2 ตามลำดับ

3. เริ่มที่โหนดลูกโหนดแรกใน Qtree ซึ่งมีรหัสเป็น 0100 แต่เนื่องจากว่ามี type เป็น AD ในการเปรียบเทียบกับโหนดใน XTI นั้นจะต้องข้ามไปหนึ่งระดับในโหนดปัจจุบัน ซึ่งใน XTI มีโหนดปัจจุบันคือ โหนดที่มีรหัส 1001 จากนั้นจะไปคำนวณจำนวนโหนดลูกของโหนดนี้ (จำนวนโหนดหลานของโหนดใน XTI ปัจจุบัน)ได้เท่ากับ 4 เริ่มเปรียบเทียบกับโหนดหลานตัวแรกมีรหัสเป็น 0100 ผลลัพธ์ที่ได้เป็น 0 (0100 XOR 0100) แสดงว่าใช่โหนดที่ต้องการ และโหนดใน Qtree นี้เป็น Leaf Node ดังนั้นจะได้ค่า Start_offset และ End_offset ของข้อมูล

4. ต่อจากนั้นตรวจสอบโหนดลูกที่คำนวณได้ในข้อ 2 พบว่ายังมีโหนดอีกใน Qtree จึงขยับไปที่โหนดถัดไปทั้งใน Qtree และ XTI แล้วนำรหัสทั้งสองมาเปรียบเทียบกันได้ผลลัพธ์เป็น 0 (1010 XOR 1010) แสดงว่าใช่โหนดที่ต้องการ และโหนดใน Qtree นี้เป็น Leaf Node ดังนั้นจะได้ค่า Start_offset และ End_offset ของข้อมูล

5. ตรวจสอบโหนดลูกที่คำนวณได้ในข้อ 2 พบว่าจำนวนโหนดลูกใน Qtree นั้นหมดแล้ว ถึงแม้ว่าในจำนวนโหนดลูกใน XTI ยังไม่หมดก็หยุดการทำงานไม่ต้องค้นหาต่อในส่วนของโหนดหลานของโหนดลูกโหนดที่ 1 ในโหนด XTI ปัจจุบัน

6. จากนั้นกลับมาที่โหนดลูกตัวที่สองใน XTI ที่คำนวณได้ในข้อ 2 ในกรณีนี้จะเริ่มค้นหาใหม่จากโหนดลูกทั้งหมดใน Qtree ทำเช่นนี้ไปเรื่อยๆจนหมดโหนดใน XTI



ภาพประกอบ 4-11 ตัวอย่างการค้นหาข้อมูล

4.4 ข้อดีของดัชนีแบบ XTI

ดัชนีแบบ XTI ใช้พื้นที่ในการจัดเก็บดัชนีน้อยกว่าดัชนีแบบ Signature ที่ได้กล่าวมาแล้วในบทที่ 3 เพราะนำหลักการของ Huffman Coding มาใช้แทนค่าข้อมูลในแต่ละโหนดทำให้มีการใช้บิตทุกบิตได้อย่างคุ้มค่ามากที่สุด และมีการสร้างดัชนีเพียงอย่างเดียวที่สามารถทำการค้นหาข้อมูลได้จากทุกการสอบถามที่มีลักษณะที่แตกต่างกัน นอกจากนี้คำตอบที่ได้จากการสอบถามอยู่ในรูปของเอกสาร XML ทำให้สะดวกในการนำไปใช้งานต่อ

4.5 ข้อด้อยของดัชนีแบบ XTI

ดัชนีแบบ XTI ไม่สามารถช่วยลดการเข้าถึงโหนดที่ไม่จำเป็นได้ ถ้าเอกสาร XML มีลักษณะโครงสร้างที่แตกต่างกันมาก ดังนั้นดัชนีแบบ XTI จึงเหมาะกับการค้นหาข้อมูลในเอกสาร XML ที่มีลักษณะโครงสร้างที่เหมือนกัน ถึงแม้ว่าดัชนีแบบ XTI จะใช้การดำเนินการตรรกะ XOR ซึ่งมีการทำงานที่ช้ากว่า AND แต่เนื่องจากการเก็บข้อมูลโดยนำหลักการ Huffman Coding ทำให้ลดจำนวนบิตลงได้มาก ทำให้การทำงานตรงนี้ไม่มีผลมากนัก

ถ้าเป็นการค้นหาข้อมูลที่อยู่ในระดับที่ลึกมากจนถึงระดับของ Leaf node และโหนดนั้นอยู่ทางขวาสุดของโครงสร้างต้นไม้ ดัชนี XTI ทำงานได้ไม่ดี เพราะต้องเสียเวลาในการค้นหาทุกโหนด

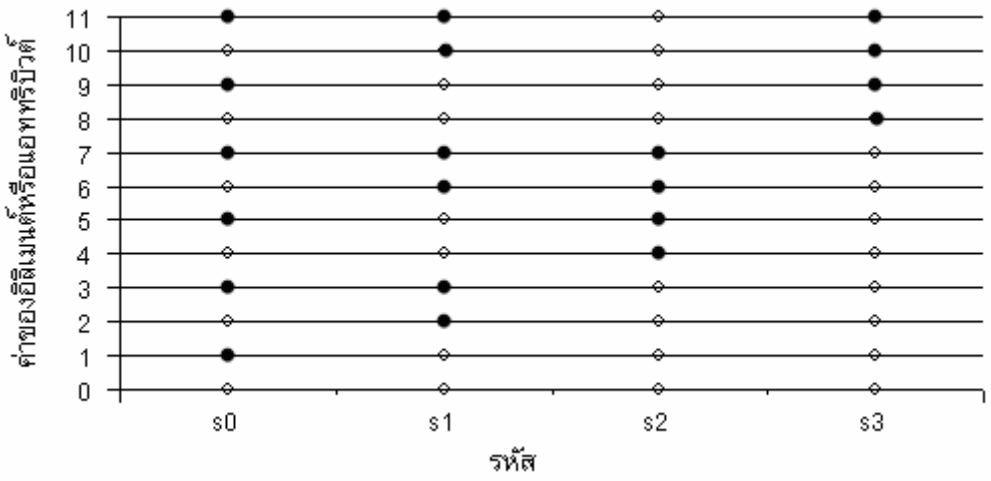
บทที่ 5

การวิเคราะห์และผลการทดลอง

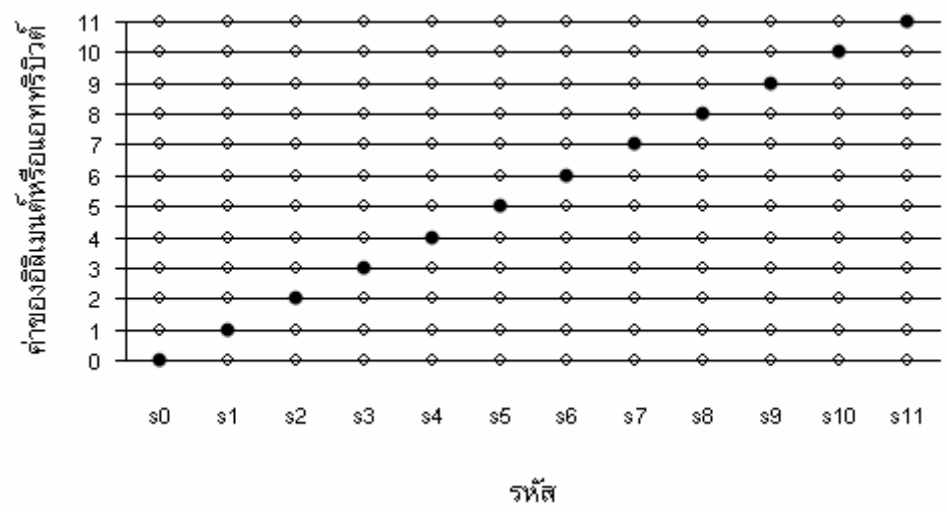
สำหรับในบทนี้จะกล่าวถึงการวิเคราะห์เปรียบเทียบค่าใช้จ่ายเกี่ยวกับพื้นที่ในการจัดเก็บดัชนี ค่าใช้จ่ายเกี่ยวกับเวลาในการประมวลผลการสอบถามข้อมูล และประสิทธิภาพในแง่ Space-Time Trade-off (การแลกเปลี่ยนระหว่างประสิทธิภาพของพื้นที่กับเวลา) ของดัชนีแบบ XTI (XML Tree Index) กับดัชนีแบบ Signature ของ Huang และ Wang (Huang and Wang, 2008) จากบทที่ 3 พบว่าดัชนีแบบ Signature มีประสิทธิภาพดีกว่าดัชนีที่เก็บในรูปแบบของข้อความ และดัชนีของ Huang และ Wang เป็นดัชนีแบบ Signature ที่มีประสิทธิภาพดีที่สุด

5.1 ค่าใช้จ่ายเกี่ยวกับพื้นที่ที่ใช้ในการจัดเก็บดัชนี

จากที่ได้กล่าวมาแล้วในบทที่ 3 และ 4 จะเห็นว่าดัชนีแบบ XTI และดัชนีแบบ Signature ของ Huang และ Wang มีการลงรหัสข้อมูลในแต่ละโหนดที่แตกต่างกัน พิจารณารูปแบบการลงรหัสที่แตกต่างกันของดัชนีทั้งสองแบบดังภาพประกอบ 5-1 และ 5-2 ตามลำดับ



ภาพประกอบ 5-1 แผนภาพการลงรหัสดัชนี XTI
เมื่อ จำนวนอิลิเมนต์และแอททริบิวต์ (n) เท่ากับ 12



ภาพประกอบ 5-2 แผนภาพการลงรหัสดัชนีของ Huang และ Wang
เมื่อ จำนวนอิลิเมนต์และแอททริบิวต์ (n) เท่ากับ 12

จากภาพประกอบ 5-1 และ 5-2 การทำดัชนีแบบ XTI ใช้พื้นที่น้อยกว่าดัชนีของ Huang และ Wang คือ กำหนดให้ จำนวนอิลิเมนต์และแอททริบิวต์ที่แตกต่างกัน (n) เท่ากับ 12 จะเห็นได้ว่าดัชนี XTI ใช้จำนวนบิตในการลงรหัสเพียง $\log_2 12 = 4$ บิต นั่นคือ S_0, S_1, S_2 และ S_4 ส่วนดัชนีของ Huang และ Wang ใช้จำนวนบิตในการลงรหัสทั้งหมดถึง 12 บิต นั่นคือ $S_1, S_2, S_3, ..., S_{11}$ ซึ่งเท่ากับจำนวนอิลิเมนต์และแอททริบิวต์ที่แตกต่างกัน

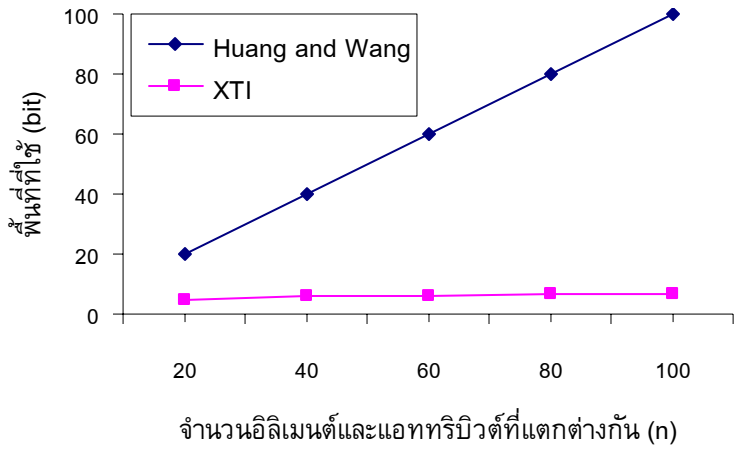
ต่อไปจะพิจารณาเปรียบเทียบพื้นที่ที่ใช้ในการจัดเก็บดัชนี XTI กับดัชนีของ Huang และ Wang ดังตารางที่ 5-1

ตารางที่ 5-1 พื้นที่ที่ใช้ในการจัดเก็บดัชนี XTI กับดัชนีของ Huang และ Wang เมื่อ n คือ จำนวนอิลิเมนต์และแอททริบิวต์ที่แตกต่างกัน และ m คือจำนวนอิลิเมนต์และแอททริบิวต์ทั้งหมด

ชนิดของดัชนี	พื้นที่ที่ใช้ในการจัดเก็บดัชนี (บิต)
ดัชนี XTI	nm
ดัชนีของ Huang และ Wang	$(\log_2 n)m$

จากตารางที่ 5-1 จะเห็นได้ว่า สำหรับการทำดัชนี XTI กับดัชนีของ Huang และ Wang นั้นถ้าใช้จำนวนอิลิเมนต์และแอททริบิวต์ทั้งหมด m โหนด พื้นที่ที่ใช้ในการจัดเก็บดัชนีทั้งสองแบบจะแปรผันตรงกับค่าจำนวนอิลิเมนต์และแอททริบิวต์ที่แตกต่างกัน (n) ที่นำมาใช้ทำดัชนี

ต่อไปเป็นการเปรียบเทียบให้เห็นถึงการใช้พื้นที่ในการจัดเก็บดัชนี XTI กับดัชนีของ Huang และ Wang โดยพิจารณาจากกราฟดังภาพประกอบ 5-3



ภาพประกอบ 5-3 กราฟแสดงการเปรียบเทียบพื้นที่ที่ใช้ในการสร้างดัชนี XTI กับดัชนีของ Huang และ Wang เมื่อจำนวนอิลิเมนต์และแอททริบิวต์ที่แตกต่างกัน

พิจารณาภาพประกอบ 5-3 แสดงความสัมพันธ์ระหว่างพื้นที่ที่ใช้ในการสร้างดัชนีกับจำนวนอิลิเมนต์และแอททริบิวต์ที่แตกต่างกันของดัชนี XTI กับดัชนีของ Huang และ Wang ซึ่งมีจำนวนอิลิเมนต์และแอททริบิวต์ที่แตกต่างกัน จะเห็นได้ว่าดัชนี XTI ใช้พื้นที่ในการเก็บดัชนีน้อยกว่าดัชนีของ Huang และ Wang

5.2 ค่าใช้จ่ายเกี่ยวกับเวลาที่ใช้ในการประมวลผลการสอบถาม

ดัชนี XTI กับดัชนีของ Huang และ Wang มีรูปแบบการลงรหัสที่แตกต่างกัน นอกจากจะทำให้มีพื้นที่ที่เก็บดัชนีแตกต่างกันแล้ว ยังทำให้ขั้นตอนการประมวลผลการสอบถามแตกต่างกันด้วย เป็นผลให้เวลาในการสอบถามแตกต่างกัน ถ้ารหัสที่ใช้มีความยาวมากต้องใช้เวลามากในการดำเนินการตรรกะระหว่างบิต พิจารณาจำนวนบิตของรหัสที่ใช้และวิธีการดำเนินการตรรกะระหว่างดัชนี XTI กับดัชนีของ Huang และ Wang ดังตารางที่ 5-2

ตารางที่ 5-2 จำนวนบิตของรหัสที่ใช้และวิธีการดำเนินการตรรกะระหว่างดัชนี XTI กับดัชนีของ Huang และ Wang

ชนิดของดัชนี	จำนวนบิตของรหัสที่ใช้	วิธีการดำเนินการตรรกะในการเปรียบเทียบ 1 โหนด
ดัชนี XTI	$\log_2 n$	XOR
ดัชนีของ Huang และ Wang	n	AND, เปรียบเทียบบิต

จากตารางที่ 5-2 จะเห็นว่าในการสอบถามข้อมูลนั้นดัชนี XTI ใช้การดำเนินการตรรกะ XOR เพียง 1 ครั้งในการค้นหาข้อมูล 1 โหนดและมีจำนวนบิตของรหัสที่ใช้เพียงแค่ $\log_2 n$ บิตเท่านั้น ส่วนดัชนีของ Huang และ Wang ในการค้นหาข้อมูล 1 โหนดต้องใช้การดำเนินการตรรกะ AND และ การเปรียบเทียบบิตอย่างละ 1 ครั้ง รวมถึงยังมีจำนวนของรหัสที่ใช้ถึง n บิต ซึ่งกรณีนี้ดัชนี XTI ใช้เวลาในการเปรียบเทียบ 1 โหนดน้อยกว่าดัชนีของ Huang และ Wang

ที่กล่าวมานี้เป็นการเปรียบเทียบค่าใช้จ่ายเกี่ยวกับเวลาที่ใช้ในการประมวลผลการสอบถามด้วยการวิเคราะห์ทางทฤษฎีโดยพิจารณาจากวิธีการการเปรียบเทียบใน 1 โหนดตามวิธีของดัชนีทั้งสอง ต่อไปจะทำการเปรียบเทียบค่าใช้จ่ายเกี่ยวกับเวลาที่ใช้ในการสอบถามจากผลการทดลอง โดยทำการทดลองบนเครื่องคอมพิวเตอร์รุ่น Intel® Core™2 Quad ระบบปฏิบัติการ Window XP Professional หน่วยความจำ 2 GB ฮาร์ดดิสก์ 320 GB และใช้ตัวแปลภาษาจาวา ในการเขียนโปรแกรม

ข้อมูลที่ใช้ในการทดสอบเป็นข้อมูลมาตรฐานจาก XMark XML Benchmark (A. Schmidt et al., 2002) โดยข้อมูลที่นำมาสร้างใช้ factor ที่ 0.02 และใช้การสอบถามทั้งหมด 6 การสอบถาม ที่มีลักษณะตามที่ได้อธิบายไปแล้วในบทที่ 2 แสดงดังตารางที่ 5-1 ประกอบด้วย Q1 Q2 และ Q3 เป็นการสอบถามตามแกน Parent-Child ส่วน Q4และ Q5 เป็นการสอบถามตามแกน Ancestor-Descendant และ Q6 เป็นการสอบถามที่มีการระบุเงื่อนไข

ตารางที่ 5-3 การสอบถาม

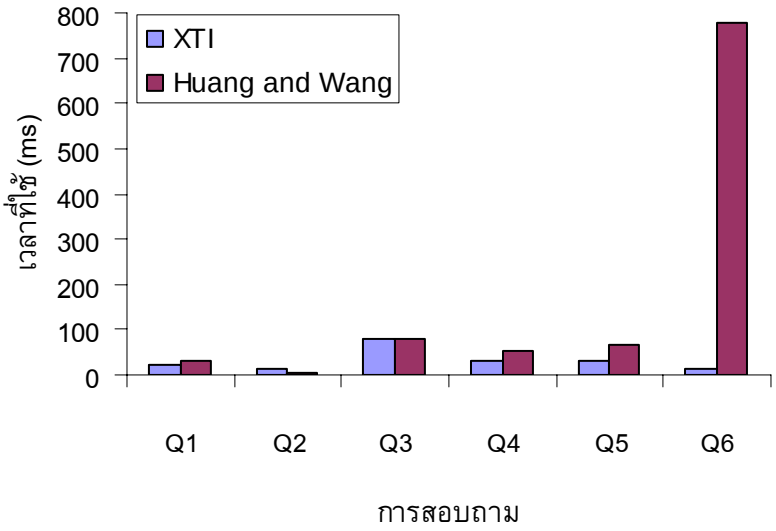
Q1	/site/regions/asia/item/mailbox
Q2	/site/closed_auctions/closed_auction/annotation/happiness
Q3	/site/people/person/id/[name]
Q4	/site/closed_auctions/closed_auction//author
Q5	/site/regions/europe/item/mailbox//to/[date]
Q6	/site/regions/europe/item/location= ‘United States’/[name]

บันทึกผลการทดลอง ดำเนินการรันโปรแกรมเพื่อการสอบถามทีละการสอบถาม ซึ่งดำเนินการซ้ำทั้งหมด 3 ครั้ง ดังผลการทดลองในตารางที่ 5-4

ตารางที่ 5-4 เวลาที่ใช้ในการสอบถามของดัชนี XTI กับดัชนีของ Huang และ Wang

การสอบถาม	**ครั้งที่**	**เวลาที่ใช้ในการสอบถาม (วินาที)**	
		ดัชนี XTI	**ดัชนีของ Huang และ Wang**
Q1	1	21	31
	2	20	31
	3	21	31
	ค่าเฉลี่ย	**20.67**	**31.00**
Q2	1	15	0
	2	16	15
	3	15	0
	ค่าเฉลี่ย	**15.33**	**5.00**
Q3	1	78	78
	2	78	78
	3	78	79
	ค่าเฉลี่ย	**78.00**	**78.33**
Q4	1	32	62
	2	32	16
	3	35	77
	ค่าเฉลี่ย	**33.00**	**51.33**
Q5	1	31	63
	2	31	78
	3	31	63
	ค่าเฉลี่ย	**31.00**	**68.00**
Q6	1	15	671
	2	16	925
	3	15	733
	ค่าเฉลี่ย	**15.33**	**776.33**

จากตารางที่ 5-4 นำเวลาที่ใช้ในการประมวลผลการสอบถามเฉลี่ยแต่ละการสอบถามมาเขียนกราฟได้ดังภาพประกอบ 5-4



ภาพประกอบ 5-4 กราฟเปรียบเทียบเวลาที่ใช้ในการสอบถามของดัชนี XTI กับดัชนีของ Huang และ Wang

จากภาพประกอบ 5-4 เห็นได้ว่าดัชนี XTI ใช้เวลาในการประมวลผลการสอบถามได้ดีเกือบทุกการสอบถาม โดยเฉพาะ Q6 ซึ่งเป็นการสอบถามที่มีการระบุเงื่อนไขดัชนีของ Huang และ Wang ต้องใช้ ValueIndex ซึ่งเป็นดัชนีแบบ B+tree ช่วยในการค้นหาข้อมูล แต่สำหรับดัชนี XTI ใช้การเปรียบเทียบแบบ String ในโหนดของต้นไม้ได้เลย ส่วน Q4 และ Q5 เป็นการสอบถามตามแกน Ancestor-Descendant ซึ่งดัชนีของ Huang และ Wang ต้องใช้ TagIndex ซึ่งเป็นดัชนีแบบ B+tree ช่วยในการค้นหาข้อมูล ทำให้เวลาในการประมวลผลการสอบถามช้ากว่าดัชนี XTI เพราะใช้เพียงดัชนีเดียว แต่ใช้การกระโดดข้ามไปหนึ่งระดับในการค้นหา

สำหรับ Q1 Q2 และ Q3 นั้น เป็นการสอบถามอยู่ในรูปแบบทั่วไปคือ เป็นการสอบถามตามแกน Parent-Child เห็นได้ว่าเวลาที่ได้ไม่แตกต่างกันมากนัก โดยเฉพาะ Q3 เวลาที่ใช้ใกล้เคียงกัน เป็นเพราะ Q3 เป็นการสอบถามที่ต้องค้นหาไปในระดับลึกที่ลึกมาก ไม่เหมือนกับ Q1 ส่วนเวลาในการสอบถาม Q2 นั้น เห็นได้ว่าดัชนี XTI ใช้เวลามากกว่าดัชนีของ Huang และ Wang เนื่องจากการสอบถามใน Q2 นี้เป็นการสอบถามในระดับลึกสุดของต้นไม้และโหนดที่ต้องการคำตอบนั้นอยู่ทางขวาสุด และจากการตรวจสอบเอกสาร XML ข้อมูลในส่วน

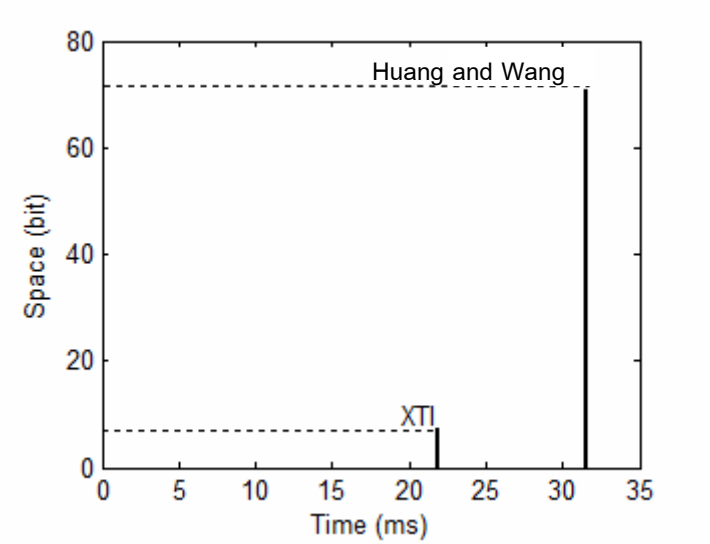
ที่ต้องการคำตอบคือโหนด happiness นั้น ไม่เป็นโหนดลูกของโหนด annotation ทุกโหนดที่มี ซึ่งจากตรงนี้เป็นข้อด้อยของดัชนี XTI ซึ่งไม่สามารถกรองโหนดที่ไม่จำเป็นออกไปก่อนได้

5.3 ประสิทธิภาพในแง่ Space-Time Trade-off

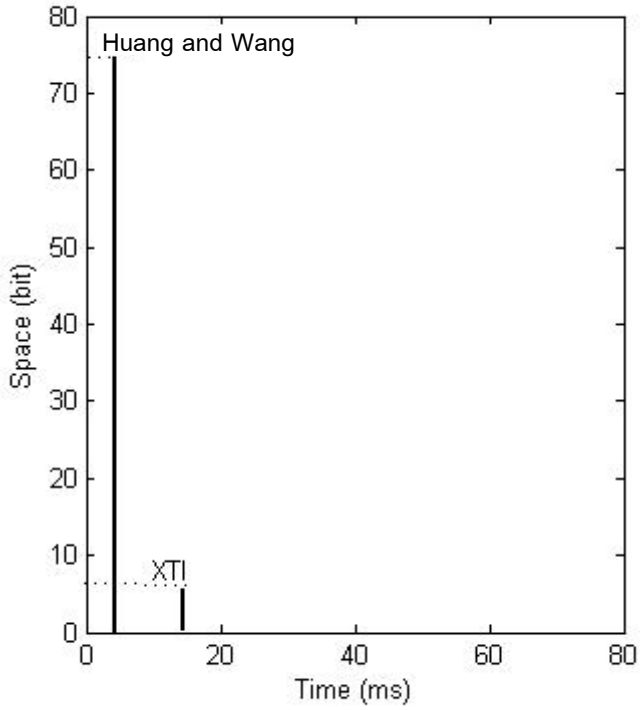
จากหัวข้อ 5.1 และ 5.2 จะเห็นว่า ดัชนี XTI มีประสิทธิภาพในเรื่องของพื้นที่สำหรับจัดเก็บดัชนีและเวลาที่ใช้ในการประมวลผลการสอบถาม ซึ่งขึ้นอยู่กับรูปแบบการสอบถามและการจัดโครงสร้างข้อมูลในเอกสาร XML และมีอีกแนวทางหนึ่งที่เป็นไปได้ในการวิเคราะห์ประสิทธิภาพ คือ การเปรียบเทียบเรื่องการแลกเปลี่ยนระหว่างประสิทธิภาพของพื้นที่กับเวลา หรือเรียกว่า ประสิทธิภาพในแง่ Space-Time Trade-off อย่างเช่น ยอมให้มีการใช้พื้นที่มากขึ้นเพื่อให้เวลาในการสอบถามน้อยลง หรือยอมให้มีการใช้เวลามากขึ้นเพื่อให้มีการใช้พื้นที่น้อยลง เป็นต้น

ในหัวข้อนี้ จะดำเนินการวิเคราะห์ประสิทธิภาพในแง่ Space-Time Trade-off ของดัชนี XTI กับดัชนีของ Huang และ Wang แต่ละการสอบถาม (Q1-Q6) โดยพิจารณาจากพื้นที่ใต้กราฟที่แสดงความสัมพันธ์ระหว่างพื้นที่ที่ใช้ (ความยาวของรหัส) กับเวลาที่ใช้การประมวลผลการสอบถามแต่ละการสอบถาม ดังภาพประกอบ 5-5, 5-6, 5-7, 5-8, 5-9 และ 5-10

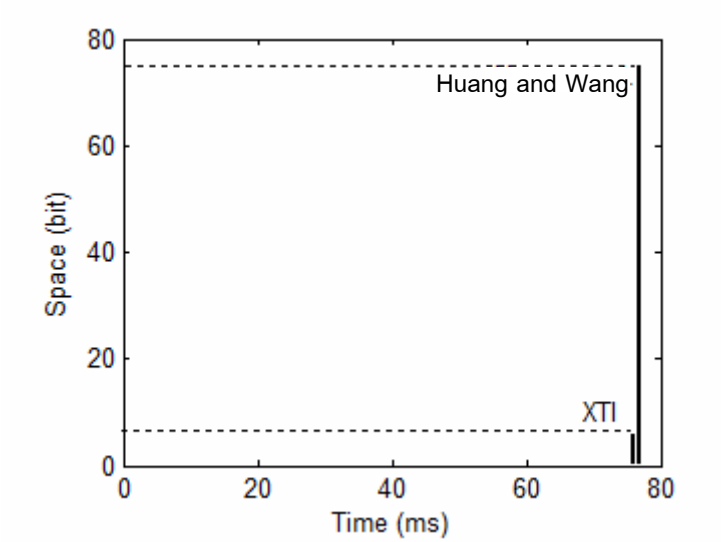
ภาพประกอบ 5-5, 5-6, 5-7, 5-8, 5-9 และ 5-10 เมื่อพิจารณาพื้นที่ใต้กราฟของดัชนี XTI กับดัชนีของ Huang และ Wang (Huang and Wang, 2008) พบว่า พื้นที่ใต้กราฟของดัชนีของ Huang และ Wangมีลักษณะเป็นรูปแบบสี่เหลี่ยมที่มีความสูงมาก เนื่องจากใช้พื้นที่สำหรับจัดเก็บดัชนีมาก จึงทำให้พื้นที่ใต้กราฟมาก ส่วนพื้นที่ใต้กราฟของดัชนี XTI มีความกว้างของฐานและความสูงน้อยกว่า เมื่อเปรียบเทียบพื้นที่ใต้กราฟของดัชนีทั้งสอบแบบจากภาพประกอบ 5-5, 5-6, 5-7, 5-8, 5-9 และ 5-10 จะเห็นได้ว่า พื้นที่ใต้กราฟของดัชนี XTI น้อยกว่าดัชนีของ Huang และ Wang หมายความว่า ดัชนี XTI มีประสิทธิภาพในแง่ Space-Time Trade-off ดีกว่า



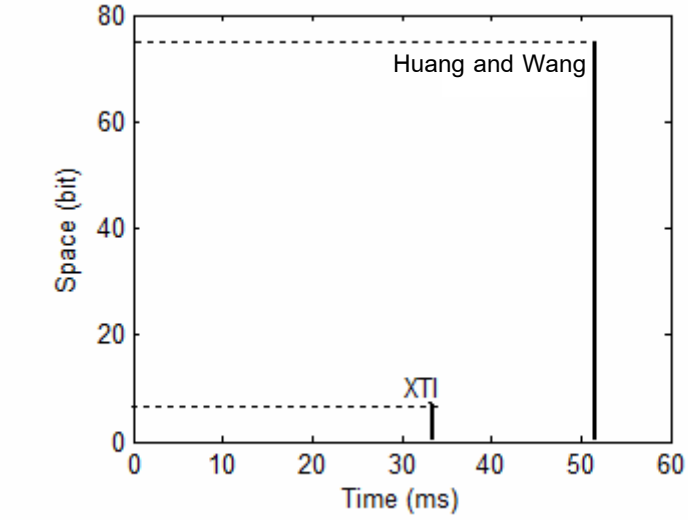
ภาพประกอบ 5-5 กราฟแสดงความสัมพันธ์ระหว่างพื้นที่ที่ใช้ (ความยาวของรหัส) กับเวลาที่ใช้การประมวลผลการสอบถามของแต่ละการสอบถามของดัชนี XTI กับดัชนีของ Huang และ Wang ในการสอบถาม Q1



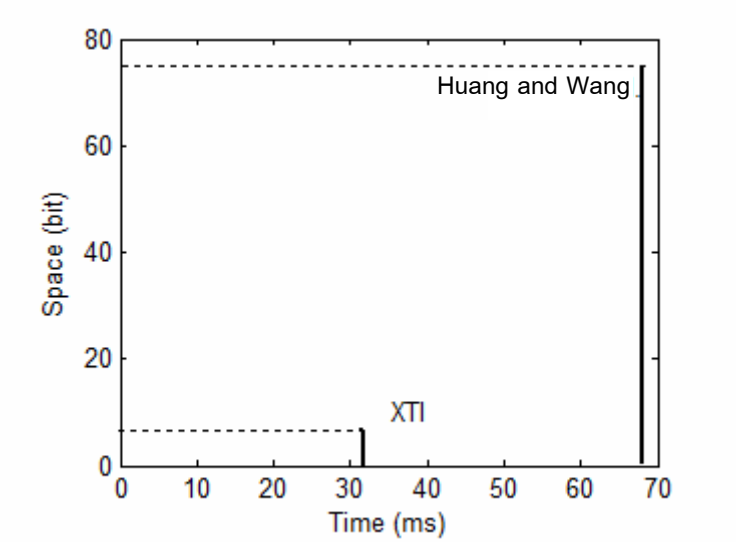
ภาพประกอบ 5-6 กราฟแสดงความสัมพันธ์ระหว่างพื้นที่ที่ใช้ (ความยาวของรหัส) กับเวลาที่ใช้การประมวลผลการสอบถามของแต่ละการสอบถามของดัชนี XTI กับดัชนีของ Huang และ Wang ในการสอบถาม Q2



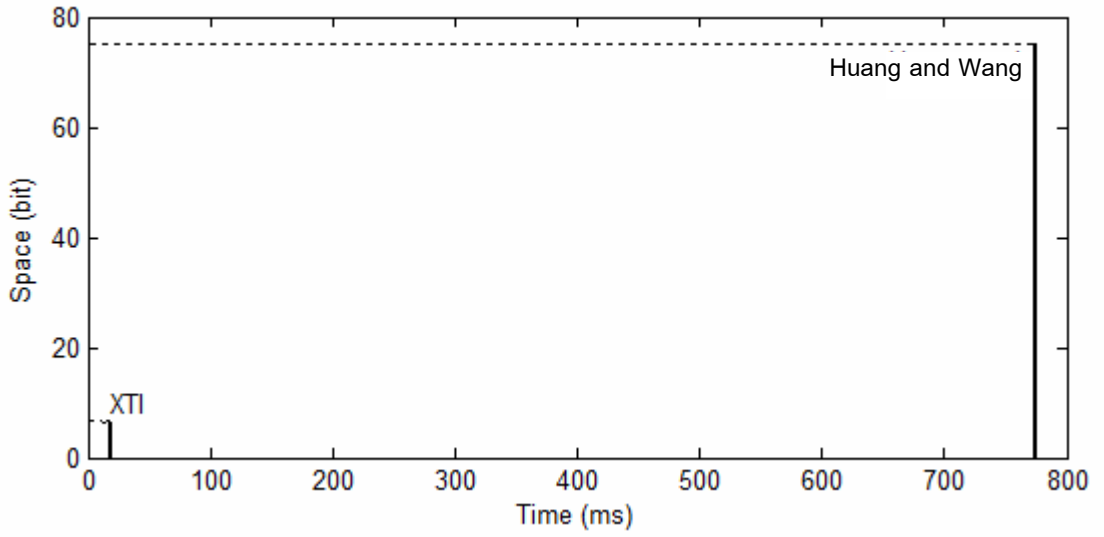
ภาพประกอบ 5-7 กราฟแสดงความสัมพันธ์ระหว่างพื้นที่ที่ใช้ (ความยาวของรหัส) กับเวลาที่ใช้การประมวลผลการสอบถามของแต่ละการสอบถามของดัชนี XTI กับดัชนีของ Huang และ Wang ในการสอบถาม Q3



ภาพประกอบ 5-8 กราฟแสดงความสัมพันธ์ระหว่างพื้นที่ที่ใช้ (ความยาวของรหัส) กับเวลาที่ใช้การประมวลผลการสอบถามของแต่ละการสอบถามของดัชนี XTI กับดัชนีของ Huang และ Wang ในการสอบถาม Q4



ภาพประกอบ 5-9 กราฟแสดงความสัมพันธ์ระหว่างพื้นที่ที่ใช้ (ความยาวของรหัส) กับเวลาที่ใช้การประมวลผลการสอบถามของแต่ละการสอบถามของดัชนี XTI กับดัชนีของ Huang และ Wang ในการสอบถาม Q5



ภาพประกอบ 5-10 กราฟแสดงความสัมพันธ์ระหว่างพื้นที่ที่ใช้ (ความยาวของรหัส) กับเวลาที่ใช้การประมวลผลการสอบถามของแต่ละการสอบถามของดัชนี XTI กับดัชนีของ Huang และ Wang ในการสอบถาม Q6

บทที่ 6

บทสรุปและข้อเสนอแนะ

งานวิทยานิพนธ์นี้ ได้นำเสนอเทคนิคการเพิ่มประสิทธิภาพการทำดัชนีสำหรับเอกสาร XML ที่เก็บอยู่ในรูปแบบไฟล์ โดยนำหลักการของ Huffman Coding มาประยุกต์ใช้สำหรับการลงรหัสให้กับแต่ละโหนด นอกจากนี้แล้วยังมีการใช้ Start_offset และ End_offset ซึ่งเป็นตำแหน่งเริ่มต้นและสิ้นสุดของแท็กต่างๆในเอกสาร XML โดยจะใช้ค่านี้สำหรับการเข้าถึงข้อมูลในเอกสาร XML โดยมีขั้นตอนการเพิ่มประสิทธิภาพ 3 ขั้นตอนหลักคือ ขั้นตอนที่ 1 Data Extraction (สกัดข้อมูล แบ่งเป็น 2 กลุ่ม คือ 1) อิลิเมนต์และแอททริบิวต์ทั้งหมดที่แตกต่างกันในเอกสาร XML และ 2) Start_offset และ End_offset ของแท็กจากเอกสาร XML) ขั้นตอนที่ 2 Mapping Table (นำอิลิเมนต์และแอททริบิวต์ที่ได้ สร้างเป็นรหัสโดยใช้ Huffman Coding) Creation และขั้นตอนที่ 3 XTI Creation (จะสร้าง XML Tree Index แบบใหม่) ผลลัพธ์ที่ได้ช่วยลดเวลาในการสอบถามข้อมูลและคำตอบที่ได้อยู่ในรูปของเอกสาร XML ซึ่งสามารถนำไปใช้ต่อได้ง่าย ซึ่งงานวิทยานิพนธ์นี้ได้รับการตีพิมพ์ผลงานในระดับประเทศ ดังนี้

เรื่อง การเพิ่มประสิทธิภาพการค้นหาข้อมูลในเอกสาร XML ของแคตตาล็อกสินค้าอิเล็กทรอนิกส์ ตีพิมพ์ใน The 6th Joint Conference on Computer Science and Software Engineering (JCSSE2009) จัดที่จังหวัดภูเก็ต ประเทศไทย วันที่ 13-15 พฤษภาคม 2552 ดังแสดงในภาคผนวก ข

เรื่อง เทคนิคการเพิ่มประสิทธิภาพการทำดัชนีสำหรับเอกสาร XML ตีพิมพ์ใน The 7th Joint Conference on Computer Science and Software Engineering (JCSSE2010) จัดที่จังหวัดกรุงเทพมหานคร ประเทศไทย วันที่ 12-14 พฤษภาคม 2553 ดังแสดงในภาคผนวก ค

6.1 บทสรุป

เทคนิคการทำดัชนีสำหรับเอกสาร XML ได้รับความนิยมใช้ในปัจจุบัน เนื่องจากภาษา XML ได้กลายเป็นมาตรฐานที่ใช้กันแพร่หลายในการแลกเปลี่ยนข้อมูลในระบบอินเตอร์เน็ต (Internet) เนื่องจากการทำดัชนีช่วยให้การค้นหาข้อมูลได้เร็วขึ้น โดยที่ผ่านมามีการพัฒนาขั้นตอนการสร้างดัชนีสำหรับเอกสาร XML ต่อๆกันมา เพื่อให้สอดคล้องกับความต้องการและมีประสิทธิภาพยิ่งขึ้น

ในอดีตที่ผ่านมามีเทคนิคการทำดัชนีสำหรับเอกสาร XML ที่จัดเก็บในรูปแบบไฟล์ที่น่าสนใจ จากการศึกษาเทคนิคการทำดัชนีสำหรับเอกสาร XML ที่จัดเก็บในรูปแบบไฟล์ ซึ่งทำดัชนีโดยใช้โครงสร้างแบบต้นไม้ สามารถแบ่งได้เป็น 2 กลุ่ม ได้แก่ การทำดัชนีโดยเก็บข้อมูลเป็นข้อความ (String) ในแต่ละโหนดของต้นไม้ และการทำดัชนีเก็บข้อมูลเป็นรหัส (Signature) ในแต่ละโหนดของต้นไม้ สำหรับงานวิทยานิพนธ์นี้ สนใจเทคนิคการทำดัชนีสำหรับเอกสาร XML ที่จัดเก็บในรูปแบบไฟล์ โดยใช้วิธีการลงรหัสให้กับแต่ละโหนดในโครงสร้างต้นไม้ ได้แก่ ดัชนีของ Park and Kim (Park and Kim, 2001) ดัชนีของ Chung และคณะ (Chung et al, 2003) และดัชนีของ Huang และ Wang (Huang and Wang, 2008) ซึ่งพบว่างานวิจัยของ Huang และWang สามารถลดพื้นที่ในการจัดเก็บดัชนี (ความยาวของรหัสที่ใช้แทนแต่ละโหนดในโครงสร้างต้นไม้) ทำให้เวลาในการสอบถามข้อมูลมีประสิทธิภาพ

ในงานวิทยานิพนธ์นี้ ได้นำเสนอเทคนิคการทำดัชนีแบบใหม่สำหรับการค้นหาข้อมูลในเอกสาร XML ซึ่งสร้างเพียงดัชนีเดียวเท่านั้นเรียกว่า XTI (XML Tree Index) โดยจะทำการลงรหัสให้กับอิลิเมนต์และแอททริบิวต์ ซึ่งนำแนวคิดของ Huffman Coding มาใช้ สำหรับสร้างรหัส และใช้ตำแหน่งเริ่มต้นและสิ้นสุดในการ เข้าถึงข้อมูลในเอกสาร XML เพื่อลดพื้นที่สำหรับเก็บดัชนีและเพิ่มความเร็วในการ ประมวลผลการสอบถาม ซึ่งสามารถพิจารณาการเปรียบเทียบลักษณะที่สำคัญของดัชนี XTI กับดัชนีของ Huang และคณะ ได้ดังตารางที่ 6-1

ตารางที่ 6-1 ลักษณะที่สำคัญของดัชนี XTI กับดัชนีของ Huang และ Wang (n แทนจำนวนอิลิเมนต์และแอททริบิวต์ที่แตกต่างกัน)

ลักษณะที่สำคัญ	ดัชนี XTI	ดัชนีของ Huang และคณะ
จำนวนบิตทั้งหมดที่ใช้ในแต่ละอิลิเมนต์และแอททริบิวต์ (พื้นที่)	$\log_2 n$	n
จำนวนครั้งในการดำเนินตรรกะสำหรับการเปรียบเทียบค่าที่เท่ากับ 1 โหนด	1XOR	1AND, 1 เปรียบเทียบบิต
การเข้าถึงโหนดที่ไม่จำเป็น	ลดได้	ลดไม่ได้

จากตารางที่ 6-1 จะเห็นได้ว่า ดัชนี XTI และดัชนีของ Huang และ Wang มีลักษณะบางประการที่แตกต่างกัน ส่งผลให้มีข้อดีและข้อจำกัดที่แตกต่างกันด้วย ซึ่งมีลักษณะดังนี้

ดัชนี XTI มีการสร้างเพียงดัชนีเดียวที่เป็นโครงสร้างแบบต้นไม้ ที่สามารถตอบคำถามได้ทุกแบบตามที่ได้อธิบายไปแล้วในบทที่ 2 ซึ่งข้อมูลในแต่ละโหนดที่ไม่ใช่ Leaf node ของต้นไม้ประกอบด้วย 3 fields คือ 1) code เป็นรหัสที่ได้จากการนำหลักการของ Huffman Coding มาประยุกต์ใช้ ซึ่งจะใช้จำนวนบิตในการแทนแต่ละอิลิเมนต์และแอททริบิวต์เพียงแค่ $\log_2 n$ 2) start_offset เป็นตำแหน่งเริ่มต้นของแต่ละแท็กในเอกสาร XML และ 3) end_offset เป็นตำแหน่งสิ้นสุดของแต่ละแท็กในเอกสาร XML ซึ่ง start_offset และ end_offset นี้ช่วยในการเข้าถึงข้อมูลในเอกสาร XML สำหรับโหนดที่เป็น Leaf node จะมี 1 field คือ name เป็นชื่อของโหนด การค้นหาข้อมูลใช้การดำเนินการตรรกะ XOR เพียง 1 ครั้งทำให้ใช้เวลาน้อยในเปรียบเทียบข้อมูลค่าที่เท่ากัน แต่มีข้อด้อยคือ ไม่สามารถลดการเข้าถึงโหนดที่ไม่จำเป็นได้

ส่วนดัชนีของ Huang และ Wang (Huang and Wang, 2008) มีการสร้างดัชนีทั้งหมด 3 ดัชนี คือ 1) Storage Tree เป็นดัชนีในรูปแบบโครงสร้างต้นไม้ ใช้ในการค้นหาข้อมูลตามแกน parent-child ข้อมูลในแต่ละโหนดของโครงสร้างต้นไม้ที่ไม่ใช่ Leaf node ประกอบด้วย 3 fields คือ Signature เป็นรหัสให้กับแต่ละโหนดที่เป็นอิลิเมนต์และแอททริบิวต์ โดยความยาวของรหัสที่ได้เท่ากัน n ส่วนอีก 2 fields เป็นลำดับการเข้าถึงในแต่ละโหนดแบบ Depth First Search ซึ่งเรียกว่า Containing Code จะใช้เป็นตัวเชื่อมในการไปดึงข้อมูลจากอีก 2 ดัชนี สำหรับโหนดที่เป็น Leaf node มีแต่ค่า Containing Code เท่านั้น 2) TagIndex ใช้สำหรับค้นหาตามแกน ancestor-descendant ซึ่งเป็นดัชนีที่สร้างด้วยขั้นตอนวิธีแบบ B+ tree โดยนำแท็กทั้งหมดที่เป็นอิลิเมนต์และแอททริบิวต์มาสร้าง โดย 1 แท็กของอิลิเมนต์และแอททริ-บิวต์จะมี TagIndex เป็นของตัวเอง และ 3) ValueIndex ใช้สำหรับการค้นหาเมื่อมีการระบุเงื่อนไข ซึ่งเป็นดัชนีที่สร้างด้วยขั้นตอนวิธีแบบ B+ tree นำค่าที่เป็น Value ทั้งหมดมาสร้างเป็นดัชนี 1 ดัชนี การค้นหาข้อมูลใช้การดำเนินการตรรกะ AND และ การเปรียบเทียบบิต สำหรับการเปรียบเทียบข้อมูลค่าที่เท่ากัน ทำให้ใช้เวลามากกว่าดัชนี XTI แต่วิธีนี้สามารถลดการเข้าถึงโหนดที่ไม่จำเป็นได้

สำหรับการคิดค้นดัชนี XTI มีวัตถุประสงค์เพื่อลดพื้นที่ในการจัดเก็บและเวลาในการประมวลผลการสอบถาม โดยดัชนี XTI นี้เหมาะกับการนำไปใช้กับเอกสาร XML ที่มีโครงสร้างของอิลิเมนต์ย่อยเหมือนๆ กัน

6.2 ข้อเสนอแนะและงานในอนาคต

1. ในการลงรหัสให้กับแต่ละอิลิเมนต์และแอททริบิวต์ นอกจากการใช้แนวคิดของ Huffman Coding แบบ fixed length แล้วอาจจะใช้แบบ non fixed length ได้อีกด้วย เพื่อที่จะลดจำนวนรหัสที่ใช้แทนข้อมูลในแต่ละโหนดลงอีก จะใช้การนับความถี่ของโหนดที่พบในเอกสาร XML เพื่อใช้เป็นเงื่อนไขในการลงรหัส และอาจจะใช้เทคนิคทาง Data Mining

อย่างเช่น Clustering หรือ Association Rules มาช่วยในการพิจารณา เพื่อให้ได้การลงรหัสสำหรับแต่ละโหนดที่มีประสิทธิภาพมากยิ่งขึ้น

2. เนื่องจากการทำดัชนีแบบ XTI ยังไม่สามารถลดการเข้าถึงโหนดที่ไม่จำเป็นได้ แต่เนื่องจากใช้จำนวนบิตในการเปรียบเทียบน้อยกว่าดัชนีของ Huang และ Wang ทำให้เวลาในการประมวลการสอบถามดีกว่า ดังนั้นสำหรับงานในอนาคตจะเป็นวิธีการสร้างดัชนีสำหรับเอกสาร XML ให้มีการเข้ารหัสที่มีประสิทธิภาพยิ่งขึ้น และลดการเข้าโหนดที่ไม่จำเป็นลงได้

บรรณานุกรม

เดวิด ฮันเตอร์. 2002. คัมภีร์การใช้ XML ฉบับสมบูรณ์. ซีเอ็ดยูเคชั่น: กรุงเทพฯ.

สุธี พงศาสกุลชัย. 2007. การพัฒนาระบบด้วยสถาปัตยกรรมเชิงบริการเทคโนโลยีบน Web Service. เคทีพี คอมพ์ แอนด์ คอนซัลท์ จำกัด: กรุงเทพฯ.

Schmidt, A., Waas, F., Kersten, M., Carey, M.J., Mano, I. and Busse, R. 2002. XMark: A Benchmark for XML Data Management. Proceedings of the 28th VLDB Conference. Hong Kong, China.

Levitin A. 2007. Introducton to The Design and Analysis of algorithms Second Edition. Pearson Education, Inc.:Untited States of America.

Silberschatz, A., Horth, H.F. and Sudarahan, S. 2006. Database Syatem Concepts Fifth Edition.

Ozgur, A. and Gundem,T.I. 2006. Efficient indexing technique for XML-based electronic product catalogs. DEEC 2005. Vol.5. pp.66-77.

Benz, B. and Durant, J.R. 2003. XML Programming Bible. Wiley Publishing, Inc.: New York.

Cooper, B.F., Sample, N., Franklin, M.J., Hjaltason, G.R., and Shadmon, M. 2001. A Fast Index for Semistructured Data. Proceeding of the 27th VLDB Conference. Roma, Italy.

Seo, C., Lee. S.W., Kim, H. J. 2003. An efficient inverted index technique for XML documents using RDBMS. Information and Software Technology. January 2003. Vol.5. pp.11-22.

Huffman, D.A. 1952. A Method for the Construction of Minimum-Redundancy Codes. Proceeding of The I.R.E. December 1952.

Bancihon, F., Barbedette, G., Benzaken, V., Delobel, C., Gamerman, S., Lecluse, C., Pfeffer, P., Richard, P., and Velez, F. 1988. The design and implementation of O2, an object-oriented database system, in: K. Dittrich (Ed.). Proceedings of the Second International Workshop on Object-oriented Database.

Tain, F., Dewit, D.J., Chen, J. and Zhang, C. 2002. The Design and Performance Evaluation of Alternative XML Storage Strategies. ACM SIGMOD Record. March 2002. Vol.31, No.1.

Jiang, H., Lu, H., Wang, W., and Yu, J.X. 2002. XParent: An Efficient RDBMS-Based XML Database System. 18th International Conference on Data Engineering (ICDE'02). pp.0335.

Schildt, H. 1990. C++ by Herbert Schildt. Published by Osborne McGraw-Hill: Untited States of America.

McHugh, J., Abiteboul, S., Goldman, R., Quass, D., and Widom, J. 1997. Lore: A database management system for semi structured data. SIGMOD Record 26 (3). pp.54-66.

Wang, K.F., Yu, J.X. and Tang, N. 2006. Answering XML Queries Using Path-Based Indexes: A Survey. Vol.9, pp.277-299.

Goldman, R. and Widom, J. 1997. DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases. Proceedings of the 23rd International Conference on Very Large Data Bases. pp.436-445.

Hashemian, R. 1995. Memory Efficient and High-Speed Search Huffman Coding. IEEE Transaction on Communications. October 1995. Vol.43, No.10.

Pal, S., Cseri, I., Seeliger, O., Scheller, G., Giakoumakis, L., Zolotov, V., Giakoumakis, L., and Zolotov, V. 2004. Indexing XML Data Stored in a Relational Database. Proceedings of the 30th VLDB Conference.Toronto, Canada.

Park, S. and Kim, H.J. 2001. A New Query Processing Technique for XML Based on Signature. Proceedings of the 7th International Conference on Database Systems for Advanced Applications. pp.22.

Milo, T. and Suciu, D. 1999. Index Structure for Path Expressions. In:Proceedings of ICDT.

Chung, Y.D., Kim, J.W. and Kim, M.H. 2003. Efficient preprocessing of XML queries using structured signatures. pp.257-264.

Huang, Y.F. and Wang, S.H. 2008. An Efficient XML Query Processing Based on Combining T-Bitmap and Index Techniques. ISCC 2008. pp.858-863.

W3Schools. 2008. http://www.w3schools.com (accessed 30/11/2008).

W3C. 2009. XML Path Language (XPath) Version 1.0. http://www.w3.org/TR/xpath (accessed 10/11/2009).

Liu, Z.H. and Murthy, R. 2009. A Decade of XML Management: An Industrial Experience Report from Oracle. ICDE 2009. pp.1351-1362.

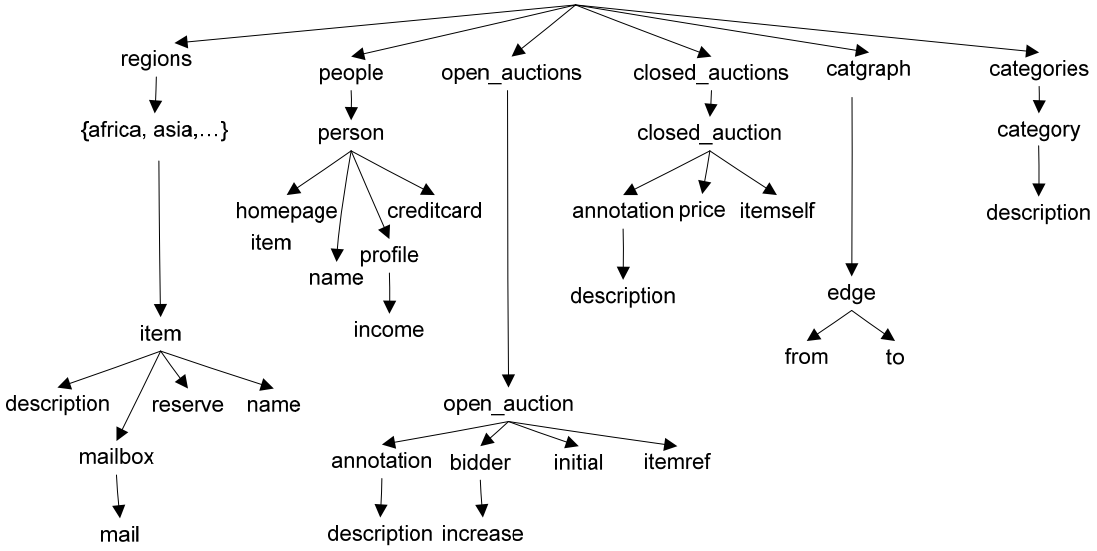
ภาคผนวก

ภาคผนวก ก.

ก.1 การเตรียมข้อมูลเพื่อใช้ในการทดลอง

เอกสาร XML ที่ใช้ในการทดลองเป็นเอกสาร XML จาก XML Benchmark (XMark – An XML Benchmark Project, 2001) ซึ่งเอกสาร XML มีขนาด 100 MB (factor 1.0) เพื่อความสะดวกในการทดสอบ จะสร้างเอกสาร XML ใหม่ โดยใช้ XMLGen จาก XMark โดยใช้ factor ที่ 0.02 จะได้เอกสาร XML ที่มีขนาด 2.4 MB

โครงสร้างของเอกสาร XML จาก XML Benchmark แสดงได้ดังภาพประกอบ ก-1



ภาพประกอบ ก-1 โครงสร้างของเอกสาร XML จาก XML Benchmark (Schmidt et al., 2002)

ก.2 การเขียนโปรแกรมเพื่อทดลองการสอบถามของดัชนี XTI กับ ดัชนีของ Huang และคณะ

ในการเขียนโปรแกรมเพื่อทดลองเป็นการเขียนโปรแกรมด้วยตัวแปรภาษาจาวา (JAVA Complier) โดยมีการกำหนด ตัวแปร ฟังก์ชัน และไฟล์ที่สำคัญดังนี้

- **ตัวแปร (variable)**

expr	ใช้สำหรับเก็บการสอบถามทั้งหมด
rootnode	เก็บข้อมูลของต้นไม้จากเอกสาร XML
mainnode	เก็บข้อมูลของต้นไม้การสอบถาม
head	เก็บข้อมูลของต้นไม้เป็นดัชนีแบบ XTI
XMLData	เก็บข้อมูลเอกสาร XML
t	เป็น pointer ชี้ไปยังข้อมูลใน XTI
q	เป็น pointer ชี้ไปยังข้อมูลในต้นไม้การสอบถาม (Query tree : Qtree)

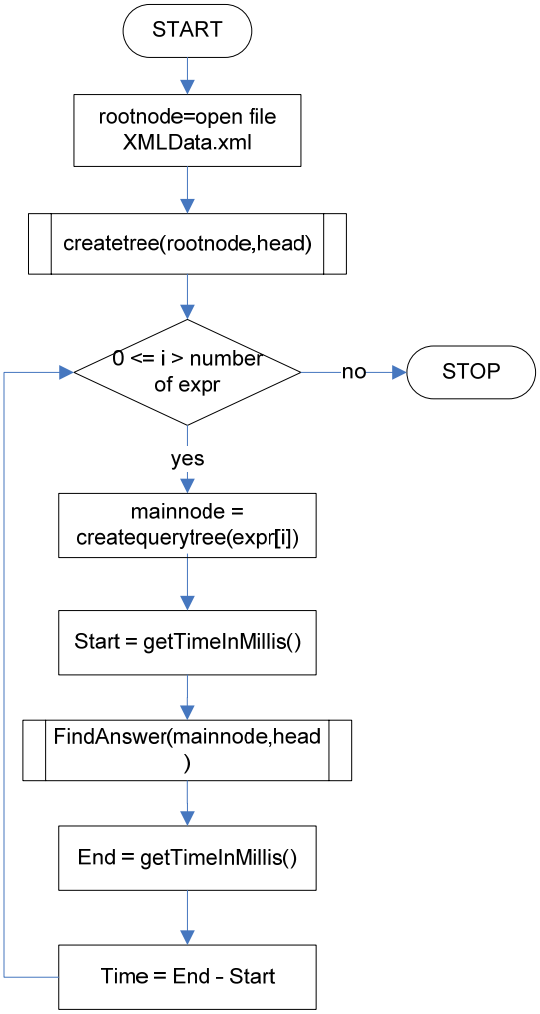
- **ฟังก์ชัน (function)**

createtree(rootnode,head)	ฟังก์ชันสำหรับสร้างดัชนี XTI
createquerytree(expr)	ฟังก์ชันสำหรับแปลงการสอบถามให้เป็นต้นไม้การสอบถาม
FindAnswer(mainnode)	ฟังก์ชันสำหรับค้นหาคำตอบ
ComputeChildsTree()	ฟังก์ชันสำหรับนับจำนวนโหนดลูกของโหนดปัจจุบันในดัชนี XTI
ComputeChildsQueryTree()	ฟังก์ชันสำหรับนับจำนวนโหนดลูกของโหนดปัจจุบันในต้นไม้การสอบถาม
PrintOutput()	ฟังก์ชันสำหรับพิมพ์คำตอบเป็นเอกสาร XML
ReadXML()	ฟังก์ชันเก็บข้อมูลเอกสาร XML ไว้ในตัวแปร XMLData

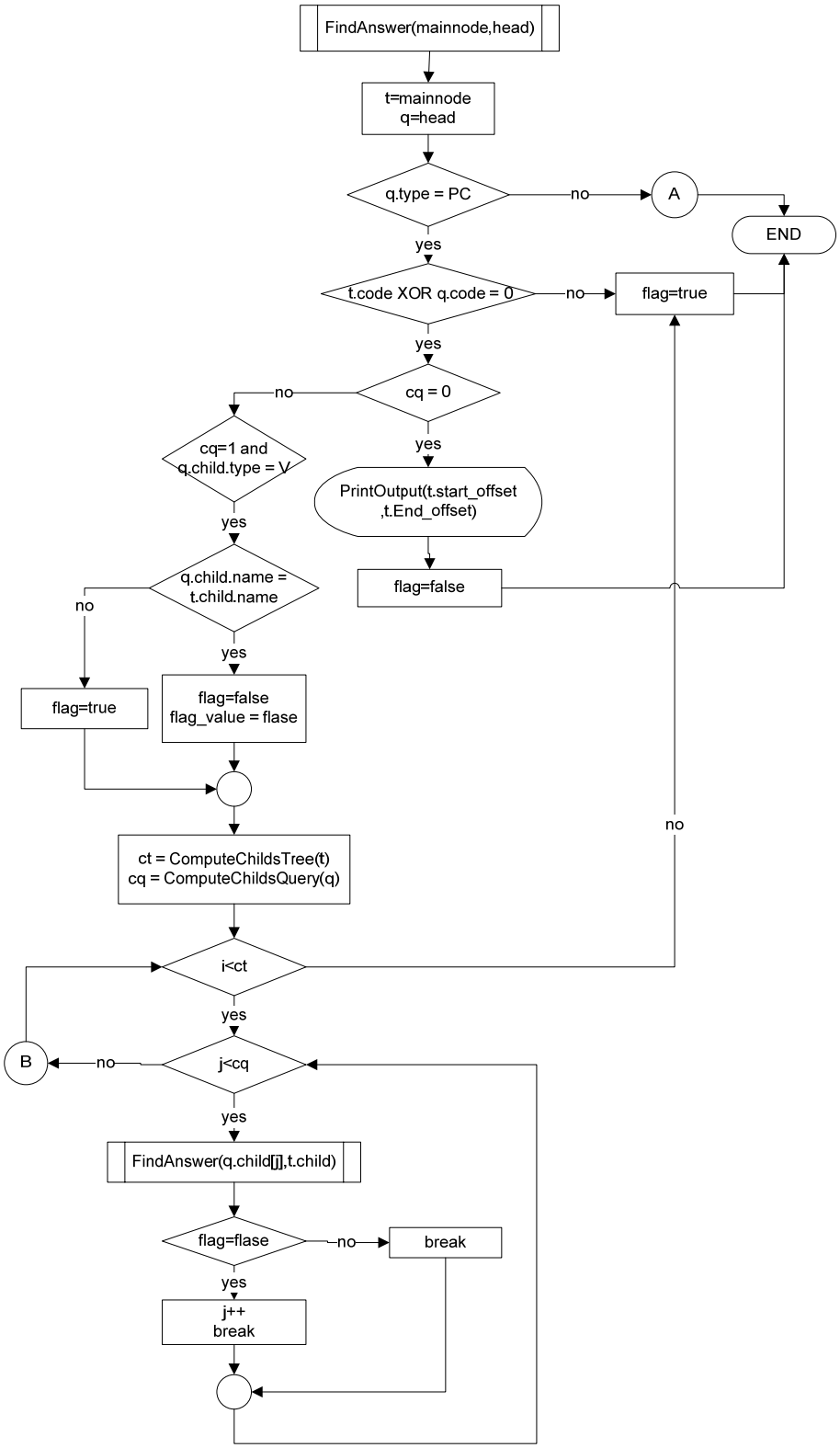
- **ไฟล์ (file)**

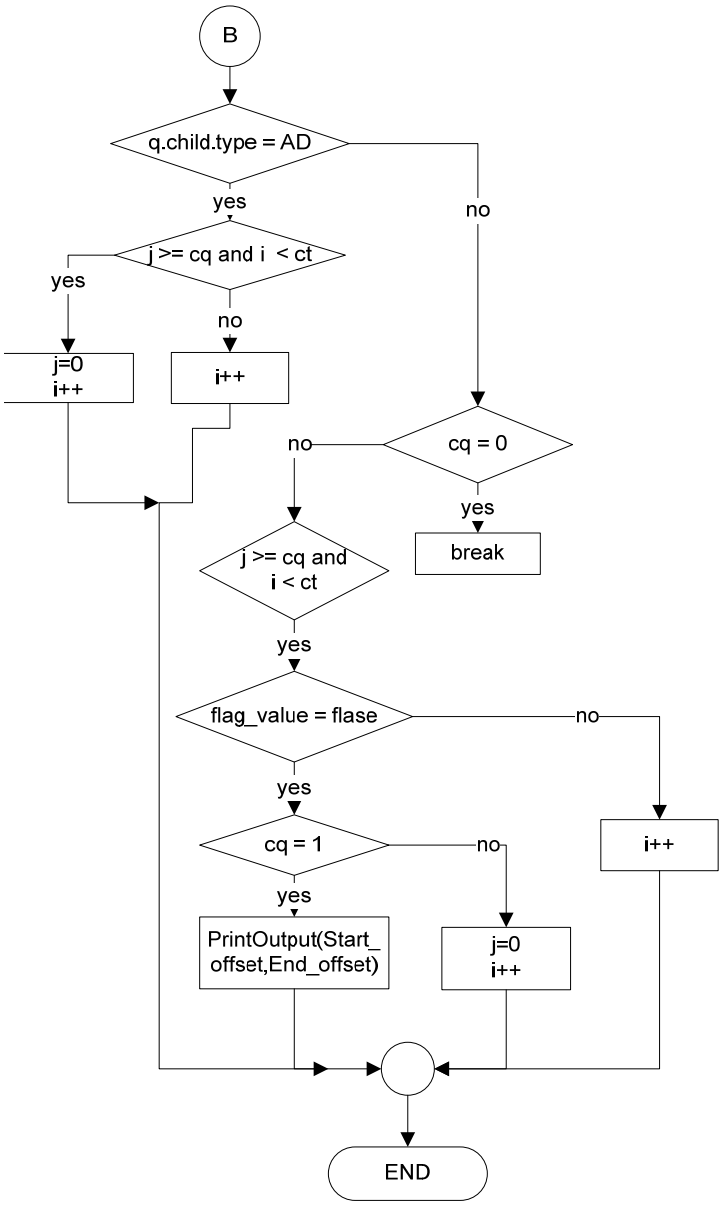
XMLData.xml	เป็นไฟล์เอกสาร XML ที่ใช้ทดสอบ

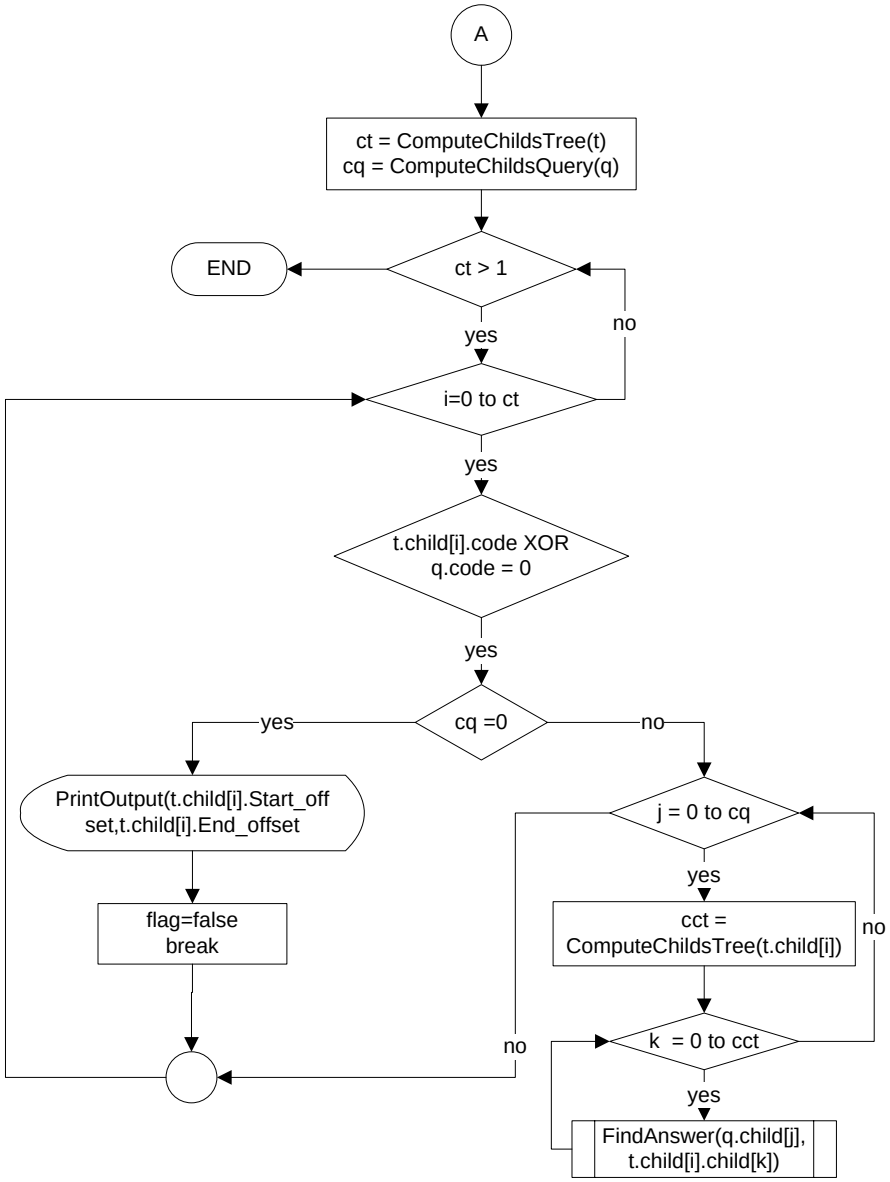
ก.2.1 ขั้นตอนวิธีการทำงานหลักของการสอบถามของดัชนี XTI กับ ดัชนีของ Huang และคณะ


START
rootnode=open file XMLData.xml
createtree(rootnode,head)
0 <= i > number of expr
no
STOP
yes
mainnode = createquerytree(expr[i])
Start = getTimeInMillis()
FindAnswer(mainnode,head)
End = getTimeInMillis()
Time = End - Start

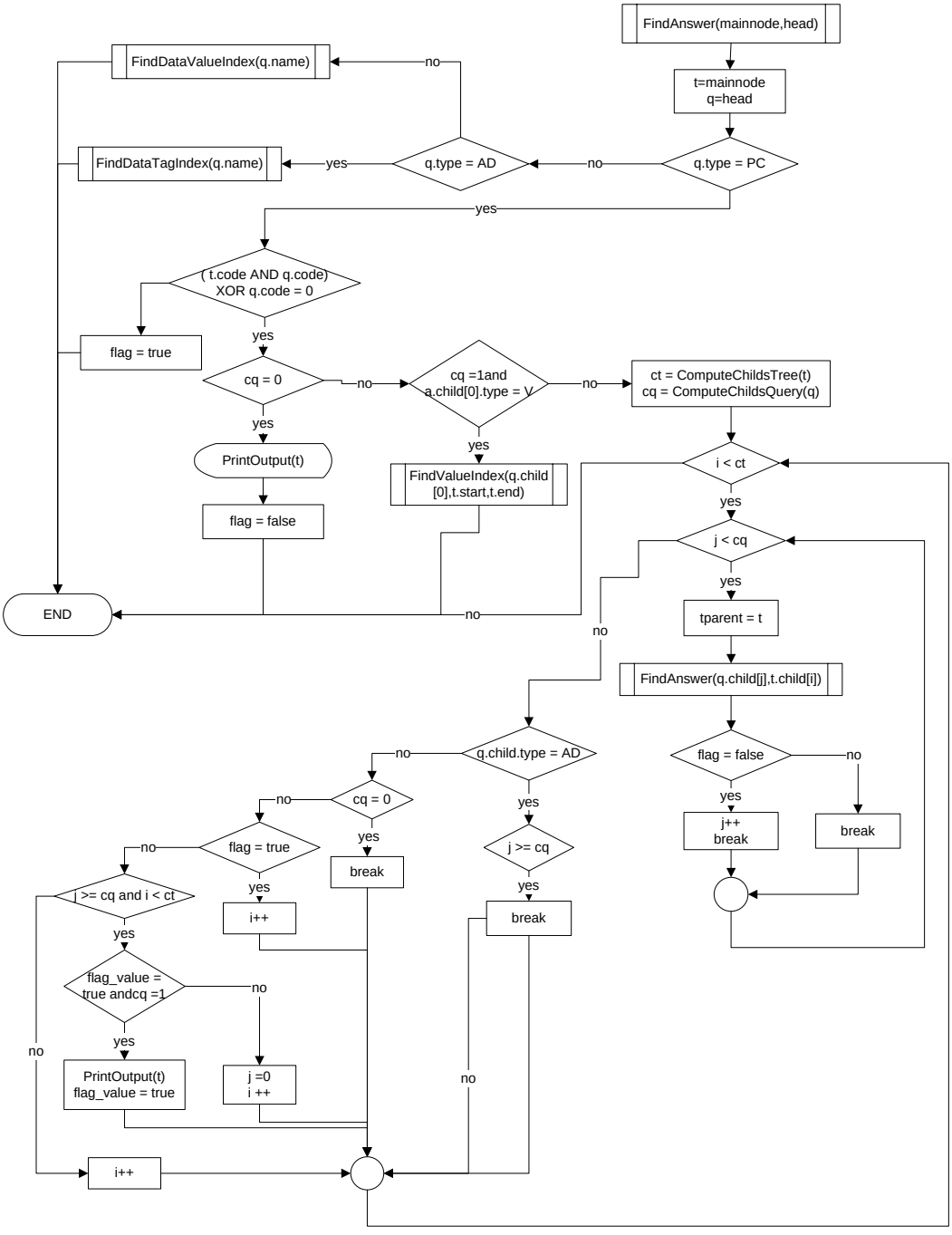
ก.2.2 ขั้นตอนวิธีการสอบถามของดัชนี XTI




B
q.child.type = AD
yes
no
j >= cq and i < ct
yes
no
j=0
i++
i++
no
cq = 0
yes
break
j >= cq and
i < ct
yes
flag_value = flase
no
yes
i++
cq = 1
no
yes
PrintOutput(Start_
offset,End_offset)
j=0
i++
END


A
ct = ComputeChildsTree(t)
cq = ComputeChildsQuery(q)
END
ct > 1
no
yes
i=0 to ct
yes
t.child[i].code XOR
q.code = 0
yes
cq =0
yes
no
PrintOutput(t.child[i].Start_off
set,t.child[i].End_offset
flag=false
break
j = 0 to cq
yes
cct =
ComputeChildsTree(t.child[i])
no
k = 0 to cct
yes
FindAnswer(q.child[j],
t.child[i].child[k])
no

ก.2.3 ขั้นตอนวิธีการสอบถามของดัชนีของ Huang และคณะ



ภาคผนวก ข

ผลงานตีพิมพ์

เรื่อง	การเพิ่มประสิทธิภาพการค้นหาข้อมูลในเอกสาร XML ของแคตตาล็อกสินค้าอิเล็กทรอนิกส์
งานประชุมวิชาการ	The 6th Joint Conference on Computer Science and Software Engineering (JCSSE2009)
สถานที่	จังหวัดภูเก็ต ประเทศไทย
วันที่	13-15 พฤษภาคม 2552

การเพิ่มประสิทธิภาพการค้นหาข้อมูลในเอกสาร XML ของแคตตาล็อกสินค้าอิเล็กทรอนิกส์
Improvement Querying XML-based Electronic Product Catalog

วรารัตน์ จักรหวัด, ศิริรัตน์ วณิชโยบล และ ปรีชา วงศ์หิรัญเดชา
ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยสงขลานครินทร์
อ.หาดใหญ่ จ.สงขลา 90112
Email: {s5110220070, sirirut.v, preecha.v}@psu.ac.th

บทคัดย่อ

ในปัจจุบันนี้การทำธุรกิจบนเว็บหรือ e-commerce เป็นที่แพร่หลาย ข้อมูลสินค้าที่นำเสนอเก็บในแคตตาล็อกอิเล็กทรอนิกส์ ซึ่งส่วนใหญ่จะอยู่ในรูปแบบของเอกสาร XML เพราะ XML เป็นมาตรฐานที่ใช้กันแพร่หลายในการแลกเปลี่ยนข้อมูลในระบบอินเตอร์เน็ต การจะทำให้การค้นหาข้อมูลสินค้าในแคตตาล็อกมีประสิทธิภาพ ทำให้ลูกค้ามีความพอใจเป็นงานสำคัญและส่งผลดีต่อบริษัท

บทความนี้ได้นำเสนอเทคนิคการทำดัชนีสำหรับการค้นหาข้อมูลในเอกสาร XML ของแคตตาล็อกสินค้าอิเล็กทรอนิกส์ โดยการเก็บค่าของอิลิเมนต์ ค่าแอททริบิวต์ และตำแหน่งเริ่มต้นสิ้นสุดในการเข้าถึงเอกสาร XML และอาศัยคุณสมบัติการสร้างดัชนีที่คอลัมน์ของ RDBMS สุดท้ายได้เสนอการวิเคราะห์และเปรียบเทียบประสิทธิภาพกับเทคนิคการทำดัชนีที่นำเสนอก่อนหน้า

คำสำคัญ : indexing, XML

Abstract

Recently, e-commerce business has become pervasive. Product data is stored in an electronic product catalog which is in the form XML documents because XML is a standard for Internet data exchange. Efficient searching of data on the catalog is important for customer satisfaction and company profit.

In this paper, we present indexing technique for searching in a XML-based electronic product catalog. The technique stores values of element, attribute, start_offset and end_offset in RDBMS and use indexing technique on column. Our comparative study show that the performance of the proposed technique is better than the techniques presented earlier.

1. บทนำ

ในปัจจุบันนี้ XML (Extensible Markup Language) เป็นมาตรฐานที่ใช้กันแพร่หลายในการแลกเปลี่ยนข้อมูลในระบบอินเตอร์เน็ต XML พัฒนามาจาก SGML ซึ่งเหมือนกับ HTML แต่ต่างกันตรงที่ HTML ถูกสร้างมาเพื่อแสดงผลข้อมูลผ่านเว็บเท่านั้น ซึ่งไม่สะดวกในการดึงส่วนของข้อมูลออกมา แต่สำหรับ XML นั้นเป็นภาษาที่ยืดหยุ่นต่อข้อมูลที่มีลักษณะแบบกึ่งโครงสร้าง (semi-structure) และเป็นตัวกลางในการแปลงข้อมูลที่อยู่ในลักษณะการเก็บในรูปแบบที่แตกต่างกันทำให้ระบบต่างๆสามารถเข้าถึงและใช้งานข้อมูลร่วมกันได้ (interoperability)

การทำธุรกิจบนเว็บหรือ e-commerce เป็นที่นิยมกันมากและข้อมูลของสินค้าที่นำเสนอให้ลูกค้าเก็บอยู่ในรูปแบบที่เรียกว่า แคตตาล็อกสินค้าอิเล็กทรอนิกส์ (Electronic Product Catalogs: ePC) ความเร็วในการค้นหารายละเอียดสินค้าต่างๆ ใน ePC ตามความต้องการของลูกค้าเป็นตัวแปรสำคัญอันหนึ่งที่ทำให้ยอดขายของระบบ e-commerce เพิ่มมากขึ้น เพราะความนิยมและประโยชน์ของ XML จึงได้มีการสร้าง ePC ให้อยู่ในรูปแบบเอกสาร

XML และจัดทำดัชนีสำหรับการสอบถามข้อมูลในเอกสาร XML เพื่อเพิ่มความเร็วในการสอบถามข้อมูลสินค้า

เพื่อเพิ่มประสิทธิภาพในการค้นหาข้อมูลในเอกสาร XML ได้มีการจัดเก็บเอกสาร XML สองแบบ [1,2] คือ 1) จัดเก็บในฐานข้อมูลที่ออกแบบมาเฉพาะสำหรับข้อมูลแบบกึ่งโครงสร้าง และ 2) จัดเก็บในฐานข้อมูลโครงสร้างเชิงสัมพันธ์ (RDBMS) โดยในแบบแรกจะสร้างดัชนีของเอกสาร XML ในรูปแบบของโครงสร้างต้นไม้ (tree) ซึ่งสะดวกในการสร้างแต่สิ้นเปลืองเนื้อที่ในหน่วยความจำ เพราะในการสอบถามแต่ละครั้งโครงสร้างต้นไม้ต้องถูกนำเข้า (load) มาในหน่วยความจำทั้งหมด ทำให้ประสิทธิภาพของการสอบถามจะลดลงมาก ถ้าข้อมูลที่เก็บมีจำนวนมากเพราะจะทำให้เสียเวลาในการค้นหาข้อมูล เนื่องจากต้นไม้มีขนาดใหญ่และการค้นหาต้องเริ่มที่โหนดราก (root node) ของต้นไม้เสมอ ส่วนแบบที่สองเป็นการแปลงโครงสร้างเอกสาร XML (XML Schema) ไปเป็นโครงสร้างตารางใน RDBMS ซึ่งเป็นวิธีที่ดี [2] และใช้คุณสมบัติของ RDBMS ในการค้นหาข้อมูล ข้อดีของวิธีการนี้คือสามารถใช้เทคนิคการทำดัชนีบนตารางใน RDBMS เพื่อความรวดเร็วในการค้นหาข้อมูล และเนื่องจาก RDBMS ได้รับการพัฒนามายาวนานปัจจุบันจึงเป็นที่นิยมใช้ในองค์กรสำหรับเก็บข้อมูลเอกสาร XML

ในบทความนี้นำเสนอโครงสร้างและเทคนิคการทำดัชนีแบบใหม่เพื่อเพิ่มประสิทธิภาพในการค้นหาข้อมูลในเอกสาร XML ของแคตตาล็อกสินค้าอิเล็กทรอนิกส์ โดยการประยุกต์ใช้ RDBMS และใช้คุณสมบัติการสร้างดัชนีที่คอลัมน์ในการค้นหาทำให้เร็วกว่าเทคนิคการทำดัชนีที่เคยมีมา

เอกสารนี้ประกอบด้วย 6 ส่วน โดยส่วนที่ 2 อธิบายถึงทฤษฎีที่เกี่ยวข้อง ส่วนที่ 3 จะกล่าวถึงรูปแบบของการสอบถามในแคตตาล็อกสินค้าอิเล็กทรอนิกส์และงานวิจัยที่เกี่ยวข้องส่วนที่ 4 เป็นการนำเสนอโครงสร้างและเทคนิคการทำดัชนีแบบใหม่เพื่อเพิ่มประสิทธิภาพในการค้นหาข้อมูลในแคตตาล็อกสินค้าอิเล็กทรอนิกส์ ส่วนที่ 5 เป็นการวิเคราะห์และประเมินค่าการสอบถามในแคตตาล็อกสินค้าอิเล็กทรอนิกส์ และส่วนที่ 6 เป็นบทสรุป

2. ทฤษฎีที่เกี่ยวข้อง

ในส่วนนี้เป็นการกล่าวถึง 1) XML 2) DTDs และ XML Schema และ 3) XML Query Languages อย่างสังเขป

2.1 XML (Extensible Markup Language)

ภาษา XML เป็นวิธีการระบุเนื้อหาโดยใช้แท็ก (tag) ในการอธิบายความหมายของข้อมูลที่ผู้ใช้สามารถกำหนดแท็กที่ใช้งานได้เอง เอกสาร XML ประกอบด้วยแท็กต่างๆ คือแท็กเริ่มต้นจะอยู่ภายใต้เครื่องหมาย < และ > ส่วนแท็กสิ้นสุดจะอยู่ภายใต้เครื่องหมาย </ และ > สำหรับแท็กเริ่มต้นไปจนถึงแท็กสิ้นสุดจะถูกเรียกว่าอิลิเมนต์ (element) แต่ละอิลิเมนต์อาจจะอยู่ในรูปแบบของแอทริบิวต์ (attribute) หรืออิลิเมนต์ย่อย (sub-element) แอทริบิวต์จะระบุคุณสมบัติให้กับอิลิเมนต์เพื่ออธิบายส่วนเพิ่มเติม ดังตัวอย่างรูปที่ 1 แสดงตัวอย่างเอกสาร XML ของข้อมูล ePC ซึ่งประกอบด้วยอิลิเมนต์ catalog เป็นราก (root) ซึ่งมีอิลิเมนต์ย่อย category มีแอททริบิวต์ name ในอิลิเมนต์ย่อย category ประกอบด้วยอิลิเมนต์ย่อยอีกชั้นคือ product ส่วนรายละเอียดภาษา XML สามารถดูได้จาก [3, 4]

2.2 DTDs และ XML Schema

ในภาษา XML มีการใช้งาน Document Type Descriptors (DTDs) สำหรับกำหนดรูปแบบ (ข้อกำหนด) ของอิลิเมนต์หรือแอททริบิวต์ที่อธิบายถึงโครงสร้างของเอกสาร XML โดยมีไวยากรณ์ คือ <!ELEMENT ตามด้วยชื่อของส่วนประกอบ ตามด้วยคำอธิบายของส่วนประกอบ

XML Schemas [7] สร้างขึ้นมาเพื่อขยาย DTDs อยู่ภายใต้ไวยากรณ์ของ XML โดยสามารถสร้างตารางใน RDBMS ให้เหมาะกับเอกสาร XML ด้วย XML Schema รูปที่ 2 และ 3 แสดงตัวอย่าง DTDs และ XML Schema ของเอกสาร XML ในรูปที่ 1 ตามลำดับ

```
<catalog>
   <category name = “TV”>
    <product>
      <id>001</id>
      <name>LCD</name>
      <brand>Samsung</brand>
      <price>19000</price>
      <class>LA-40A330J1</class >
      <size>29</size >
    </product>
        . . .
   </category>
        . . .
</catalog>
```

รูปที่ 1: ตัวอย่างเอกสาร XML ของข้อมูล ePC

```
<!ELEMENT catalog(category)>
<!ELEMENT category(name,product)>
<!ELEMENT name(#PCDATA)>
<!ELEMENT product(id,name*,brand,price,class,size)>
<!ELEMENT id(#PCDATA)>
<!ELEMENT name(#PCDATA)>
<!ELEMENT brand(#PCDATA)>
<!ELEMENT price(#PCDATA)>
<!ELEMENT class(#PCDATA)>
<!ELEMENT size(#PCDATA)>
```

รูปที่ 2: ตัวอย่าง DTDs

```
<xs:schema>
  <xs:element name=”catalog”>
    <xs:complexType>
    <xs:sequence>
      <xs:element name=“category”>
        <xs:complexType>
        <xs:sequence>
       <xs:element name=“name” type=“xs:string”>
          <xs:element name= “product”>
            <xs:complexType>
            <xs:sequence>
          <xs:element name= “id” type=“xs:string”/>
          <xs:element name= “name”  type=“xs:string”/>
          <xs:element name= “brand” type=“xs:string”/>
          <xs:element name= “price” type=“xs:integer”/>
          <xs:element name= “class” type=“xs:string”/>
          <xs:element name= “size” type=“xs:integer”/>
            </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

รูปที่ 3: ตัวอย่าง XML Schema

2.3 XML Query Languages

XML Query Languages เป็นภาษาสำหรับค้นหาข้อมูลในเอกสาร XML เช่น XML-QL [8] และ Lorel [9] เป็นต้น ทุกภาษาจะใช้ กลไกของ XPath [10] เช่น XML-QL มีรูปแบบไวยากรณ์คล้ายกับ SQL ดังแสดงในรูปที่ 4

```
SELECT P.price
FROM product P
WHERE P.name = “LCD TV”
```

รูปที่ 4: ตัวอย่าง XML-QL

3. งานวิจัยที่เกี่ยวข้อง

ในส่วนนี้จะกล่าวถึงรูปแบบของการสอบถามข้อมูลในเอกสาร XML ของ ePC และงานวิจัยที่เกี่ยวข้อง

3.1 รูปแบบของการสอบถามในแคตตาล็อกสินค้าอิเล็กทรอนิกส์

รูปแบบการสอบถามสินค้าในแคตตาล็อกแบ่งเป็น 4 แบบ [5] ดังรายละเอียดข้างล่าง

- รูปแบบที่ 1 เป็นการสอบถามที่ลูกค้าต้องการค้นหาสินค้าที่มีคุณสมบัติตามที่ระบุ เช่น

  Q1: /catalog/category[@name = TV’]/product[name = ‘LCD’]
- รูปแบบที่ 2 เป็นการสอบถามที่ลูกค้าต้องการค้นหาค่าของคุณสมบัติตามชนิดของสินค้า เช่น

  Q2: /catalog/category[@name = ‘TV]/product/name
- รูปแบบที่ 3 เป็นการสอบถามที่ลูกค้าต้องการค้นหาทุกสินค้าในแต่ละประเภทที่ระบุ เช่น

  Q3: /catalog/category[@name = ‘TV’]
- รูปแบบที่ 4 เป็นการสอบถามที่ลูกค้าระบุสินค้ามาให้และต้องการรู้อีกคุณสมบัติหนึ่งของสินค้านั้น เช่น

  Q4: /catalog/category[@name = ‘TV]/product[name = ‘LCD’]/price

3.2 งานวิจัยที่เกี่ยวข้อง

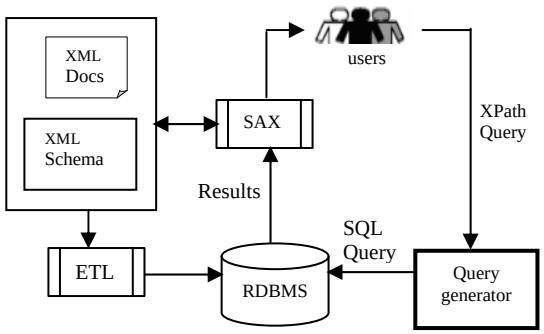
มีงานวิจัยจำนวนมากได้เสนอเทคนิคการทำดัชนีกับเอกสาร XML เช่น 1-indexes และ 2-indexes [11] เทคนิคนี้จะเริ่มหาข้อมูลตั้งแต่โหนดรากเสมอ มีการนำเสนอการแก้ปัญหานี้คือใช้ Bindex และ Lindex [12] เทคนิคการทำดัชนีแบบ Inverted เป็นที่นิยม [13] วิธีนี้จะแยกเก็บค่าของอิลิเมนต์และแอททริบิวต์ในแต่ละตาราง แม้ว่าวิธีนี้ใช้เพียง 2 ตาราง (ตารางเก็บค่าอิลิเมนต์และตารางเก็บค่าแอททริบิวต์) อย่างไรก็ตามถ้าค้นหาข้อมูลจำเป็นต้อง join ตารางทั้งสองนี้เพื่อให้ได้ path ที่ต้องการ ดังตัวอย่างเช่น /catalog/category[@name = TV'] เพื่อที่จะหาคำตอบต้อง join 3 ครั้ง คือ ครั้งที่ 1 ได้ /catalog/category ครั้งที่ 2 ได้/catalog/category[@name] และครั้งที่ 3 ได้ /catalog/category[@name = TV'] ต่อมามีการแก้ปัญหาโดยนำเสนอ 4-index [13] งานวิจัยที่กล่าวมาเป็นการเสนอเทคนิคดัชนีกับเอกสาร XML ทั่วๆไป แต่ Ozgur และคณะในปี 2006 เสนอดัชนีกับเอกสาร XML ที่เป็น ePC โดยใช้ 4 ตารางในการทำดัชนีสำหรับใช้กับเอกสาร XML ของแคตตาล็อกสินค้า คือ 1) ตาราง CategoryIndex จัดเก็บ path ของประเภทสินค้าที่มีในแคตตาล็อก เช่น /catalog/category[@name = 'TV'] และตำแหน่งเริ่มต้นและสิ้นสุด 2) ตาราง PathIndex จัดเก็บ path ของแอททริบิวต์ที่มีในแต่ละอิลิเมนต์ เช่น /catalog/category[@name = 'TV']/product/name 3) ตาราง ValueIndex จัดเก็บค่าของแอททริบิวต์ และ 4) ตาราง ProductIndex จัดเก็บค่าเริ่มต้นและสิ้นสุดของอิลิเมนต์ product แต่ละตัวในเอกสาร XML ข้อเสียของเทคนิคการทำดัชนีแบบนี้คือการเก็บ path ของอิลิเมนต์และแอททริบิวต์ทั้งหมดส่งผลให้การประมวลผลในการค้นหาข้อมูลล่าช้าเนื่องจากต้อง load ข้อมูลทั้งตารางมาทำการเปรียบเทียบกับ XPath

4. เทคนิคการทำดัชนีแบบใหม่เพื่อเพิ่มประสิทธิภาพในการค้นหาข้อมูลในเอกสาร XML ของแคตตาล็อกสินค้าอิเล็กทรอนิกส์

ในส่วนนี้นำเสนอโครงสร้างการทำงานและเทคนิคการทำดัชนีสำหรับการค้นหาข้อมูลในเอกสาร XML ของแคตตาล็อกสินค้าอิเล็กทรอนิกส์ (ePC) โดยอาศัยคุณสมบัติของ RDBMS กล่าวคือแทนที่จะเก็บ path ของเอกสาร XML เหมือนงานวิจัยอื่น เทคนิคนี้จะทำการเก็บค่าของอิลิเมนต์ ค่าของแอททริบิวต์และขอบเขตในเอกสาร XML และใช้คุณสมบัติการทำดัชนีใน RDBMS ทำการค้นหาข้อมูลที่ต้องการในเอกสาร XML จากนั้นใช้ SAX ในการประมวลผล [6]

โครงสร้างการทำงานของระบบ แสดงในรูปที่ 5 ประกอบด้วยส่วนประกอบหลัก ๆ ดังนี้

- เอกสาร XML และ XML Schema ของข้อมูล ePC
- ETL (Extract Transformation Load) ทำหน้าที่ดึงข้อมูล เปลี่ยนแปลง และโหลดข้อมูลจากเอกสาร XML และ XML Schema หลังจากนั้นทำการสร้างโครงสร้างของฐานข้อมูล อัตโนมัติใน RDBMS
- Query generator เป็นกระบวนการเปลี่ยนการสอบถามจากผู้ใช้ (users) ซึ่งในรูปของ XPath [10] ให้เป็นคำสั่ง SQL เพื่อการค้นหาข้อมูลในส่วน RDBMS
- SAX (Simple API for XML) ทำหน้าที่นำผลลัพธ์ที่ได้จาก RDBMS ไปค้นหาข้อมูลในเอกสาร XML
- RDBMS เป็นระบบจัดการฐานข้อมูลเชิงสัมพันธ์



รูปที่ 5 โครงสร้างการทำงาน

บทความนี้เสนอเทคนิคการทำดัชนีกับข้อมูลในเอกสาร XML โดยใช้ RDBMS มีแนวคิดคือเก็บค่าของอิลิเมนต์ ค่าแอททริบิวต์และค่าเริ่มต้นสิ้นสุดทำให้ไม่ต้องอ่านเอกสารทั้งหมด โดยมีการใช้ 2 ตารางคือ Category Index Table และ Product Table มีรายละเอียดดังนี้

1. ตารางดัชนีประเภทสินค้า (Category Index Table)

ตาราง Category Index Table เก็บค่าของ category (แอททริบิวต์ของ category) เช่น โทรทัศน์ เครื่องเล่นวิดีโอ เครื่องปรับอากาศ เป็นต้น ทั้งนี้เพราะว่าในการสอบถามข้อมูลใน ePC นั้นบ่อยครั้งที่ลูกค้าจะค้นหาข้อมูลสินค้าตามประเภทสินค้า ดังนั้นการทำดัชนีของประเภทสินค้า (category index) จึงมีความสำคัญ

ตารางที่ 5 แสดงตัวอย่างตาราง Category Index Table ของรูปที่ 1 ซึ่งประกอบด้วยค่าของ category แต่ละ category ข้อมูลประเภทสินค้าแต่ละประเภทและตำแหน่งเริ่มต้นและตำแหน่งสิ้นสุดของอิลิเมนต์ category นั้นในเอกสาร XML

2. ตารางรายละเอียดสินค้า (Product Table)

เป็นตาราง Product Table เก็บค่าของอิลิเมนต์นย่อยและตำแหน่งเริ่มต้นสิ้นสุด ตัวอย่าง Product Table ของอิลิเมนต์ย่อยของรูปที่ 1 แสดงในตารางที่ 6

ในรูปที่ 5 เอกสาร XML และ XML Schema จะผ่านกระบวนการ ETL เข้ามาใน RDBMS เมื่อผู้ใช้ทำการสอบถาม XPath Query ถูกสร้างขึ้นมา จากนั้นผ่าน Query generator ซึ่งเปลี่ยน XPath Query เป็น SQL Query เพื่อค้นหาข้อมูลใน RDBMS ผลลัพธ์ (Results) ที่ได้จะเป็นค่าตำแหน่งเริ่มต้นและสิ้นสุด ของอิลิเมนท์หรือแอททริบิวต์ที่อยู่ในเอกสาร XML เพื่อทำการส่งต่อให้ส่วน SAX ดึงข้อมูลจากเอกสาร XML

ตารางที่ 5: Category Index Table

name	start_ category	end_ category
TV	2	106
Air	111	163

5. การวิเคราะห์และประเมินค่าการสอบถาม

ในส่วนนี้จะเป็นการวิเคราะห์และประเมินค่าการสอบถามเอกสาร XML ของ ePC 4 รูปแบบ ดังอธิบายไว้ในหัวข้อที่ 3.1

5.1 ประเมินค่าการสอบถามรูปแบบที่ 1

ตัวอย่างเช่น /catalog/category[@name='TV']/product [name = 'LCD'] กล่าวคือลูกค้าต้องการหา TV ชนิด LCD ถ้าใช้เทคนิคการทำดัชนีก่อนหน้า [5] จะต้องใช้การดำเนินการ join 2 ครั้ง (PathIndex กับ ValueIndex และ ValueIndex กับ ProductIndex) แต่ไม่มีการดำเนินการ join ถ้าใช้เทคนิคดัชนีที่งานวิจัยนี้นำเสนอเพราะสามารถได้ผลลัพธ์จากการสอบถามจาก Product Table

5.2 ประเมินค่าการสอบถามรูปแบบที่ 2

ตัวอย่างเช่น /catalog/catalog/category[@name = 'TV']/product/name กล่าวคือลูกค้าต้องการหาชื่อสินค้าทั้งหมดที่เป็น TV ถ้าใช้เทคนิคการทำดัชนีก่อนหน้า [5] จะต้องมีการดำเนินการ join 1 ครั้ง (Pathindex กับ ValueIndex) แต่ไม่มีการดำเนินการ join เกิดขึ้นถ้าใช้เทคนิคดัชนีที่งานวิจัยนี้นำเสนอเพราะสามารถได้คำตอบจากการสอบถามจาก Product Table

5.3 ประเมินค่าการสอบถามรูปแบบที่ 3

ตัวอย่างเช่น /catalog/category[@name = 'TV'] กล่าวคือลูกค้าต้องการหาสินค้าทั้งหมดที่เป็น TV แสดงรายละเอียดทุกอย่างที่มี ถ้าใช้เทคนิคการทำดัชนีก่อนหน้า [5] และเทคนิคดัชนีที่งานวิจัยนี้นำเสนอไม่มีการดำเนินการ join เกิดขึ้น แต่การค้นหาข้อมูลในตารางของวิธีที่นำเสนอก่อนหน้า [5] ต้องใช้การเปรียบเทียบข้อความ (string matching) ทำให้การทำงานช้าเนื่องจากต้องโหลดข้อมูลทั้งตารางมาเก็บไว้ในหน่วยความจำ ส่วนวิธีที่นำเสนอนี้สามารถใช้คุณสมบัติการสร้างดัชนีของ RDBMS ในการสร้างดัชนีที่คอลัมน์เพื่อการค้นหาที่มีประสิทธิภาพ

ตารางที่ 6: Product Table

id	name	brand	price	class	size	start_ product	end_ product	category_ name
001	LCD	Sumsang	19000	LA-40A330J1	29	3	10	TV
002	Nation	Philips	39000	AN-422N	-	11	18	Air

5.4 ประเมินค่าการสอบถามรูปแบบที่ 4

ตัวอย่างเช่น /catalog/category[@name = 'TV]/product [name = 'LCD']/price กล่าวคือลูกค้าต้องการหาราคา LCD เป็น TV ถ้าใช้เทคนิคการทำดัชนีก่อนหน้า [5] จะมีการดำเนินการ join 2 ครั้ง (PathIndex กับ ValueIndex และ PathIndex กับ ValueIndex) แต่ไม่มีการดำเนินการ join เกิดขึ้นถ้าใช้เทคนิคดัชนีที่งานวิจัยนี้นำเสนอเพราะสามารถได้ผลลัพธ์จากการสอบถามจาก Product Table

จากการวิเคราะห์และประเมินค่าการสอบถามทั้ง 4 รูปแบบ จะเห็นได้ว่าถ้าใช้เทคนิคดัชนีที่งานวิจัยนี้นำเสนอผลลัพธ์ของการสอบถามทั้ง 4 รูปแบบไม่ต้องมีการดำเนินการ join เกิดขึ้น ซึ่งเป็นการลดการประมวลผลการสอบถามอย่างมีประสิทธิภาพ

6. บทสรุป

ในระบบธุรกิจการทำให้ลูกค้ามีความพอใจถือเป็นงานสำคัญ ดังนั้นการเพิ่มประสิทธิภาพในการค้นหาข้อมูลในแคตตาล็อกสินค้าให้เร็วขึ้นทำให้เกิดผลดีต่อบริษัท การสร้างแคตตาล็อกสินค้าในรูปแบบเอกสาร XML เป็นที่นิยมเพราะมีความเป็นมาตรฐานและยืดหยุ่นในการออกแบบ ในบทความนี้ได้นำเสนอเทคนิคการทำดัชนีโดยแปลงโครงสร้างและลักษณะข้อมูลของเอกสาร XML ให้เป็นข้อมูลเก็บใน RDBMS เป็นการเพิ่มประสิทธิภาพการสอบถามจากนั้นอาศัยคุณสมบัติของ RDBMS คือใช้การทำดัชนีที่คอลัมน์ในตารางเพื่อทำการค้นหาค่าตำแหน่งเริ่มต้นและสิ้นสุดของข้อมูลที่จะอ่านในเอกสาร XML ทำให้ไม่ต้องอ่านเอกสาร XML ทั้งหมดและได้ทำการเปรียบเทียบกับเทคนิคที่นำเสนอก่อนหน้า

7. เอกสารอ้างอิง

[1] H. Jiang, H. Lu, W. Wang, and J. X. Yu, "XParent: An Efficient RDBMS-Based XML Database System", 2002.

[2] J. Shanmugasundaram, K. Tufte, and G. He, "Relational Databases for Querying XML Documents: Limitations and Opportunities", Proceedings of the 25th VLDB Conference, 1999.

[3] R. Cover, "The SGML/XML Web Page", http://www.oasis-open.org/cover/xml.html.

[4] T. Bray, J. Paoli, C. M. Sperberg-McQueen, "Extensible Markup Language (XML) 1.0", Available from: http://www.w3.org/TR/REC-xml.

[5] A. Ozgur, T. I. Gundem, "Efficient indexing technique for XML-based electronic product catalogs", Electronic Commerce Research and Applications, 2006.

[6] W. Jakawat, S. Vanichayobon, and P. Vonghirandecha "Improvement Querying XML-based Electronic Product Catalog using RDBMS", Technical Report, Computer Science Department, Faculty of Science, Prince of Songkla University, Thailand, 2008.

[7] Microsoft Corporation, XML Schema, http://www.microsoft.com/xml/schema/reference/star.asp.

[8] Deutsch, M. Fernandez, D. Florescu, A. Levy, D. Suciu, "XML-QL: A Query Language for XML", http://www.w3.org/TR/NOTE-xml-ql.

[9] S. Abiteboul, D. Quass, J. McHugh, J. Widom, J. Wiener, "The Lorel Query Language for Semistructured Data", International Journal on Digital Libraries, 1(1), pp. 68-88, April 1997.

[10] W3C Recommendation, XML Path Language (Xpath) 1.0, Available from: http://www.w3.org/TR/xpath, 1999.

[11] T. Milo, D. Suciu, "Index structures for path expressions", in: Proceedings of the International Conference on Database Theory, 1999, pp. 277–295.

[12] J. McHugh, S. Abiteboul, R. Goldman, D. Quass, J. Widom, "Lore: A database management system for semi structured data", SIGMOD Record 26 (3), 1997, pp. 54–66.

[13] C. Seoa,*, S. W. Leeb, H .J .Kima, "An efficient inverted index technique for XML documents using RDBMS", Information and Software Technology 45, 2003, pp. 11–22.

[14] C. Zhang, J. Naughton, D. DeWitt, Q. Luo, G. Lohman. "On supporting containment queries in relational database management systems", in: Proceedings of the ACM SIGMOD International Conference on the Management of Data, 2001.

ภาคผนวก ค

ผลงานตีพิมพ์

เรื่อง	เทคนิคการเพิ่มประสิทธิภาพการทำดัชนีสำหรับเอกสาร XML
งานประชุมวิชาการ	The 7th Joint Conference on Computer Science and Software Engineering (JCSSE2010)
สถานที่	จังหวัดกรุงเทพมหานคร ประเทศไทย
วันที่	12-14 พฤษภาคม 2553

เทคนิคการเพิ่มประสิทธิภาพการทำดัชนีสำหรับเอกสาร XML
An Efficiency Indexing Technique for XML Documents

วรารัตน์ จักรหวัด และ ศิริรัตน์ วณิชโยบล
ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยสงขลานครินทร์
อ.หาดใหญ่ จ.สงขลา 90112
Email: {s5110220070, sirirut.v}@psu.ac.th

บทคัดย่อ

ในปัจจุบันนี้ การนำเสนอข้อมูลในระบบอินเตอร์เน็ตส่วนใหญ่จะอยู่ในรูปแบบของเอกสาร XML เพราะ XML เป็นมาตรฐานที่ใช้กันแพร่หลายในการแลกเปลี่ยนข้อมูลในระบบอินเตอร์เน็ต เพื่อที่จะเพิ่มประสิทธิภาพในการค้นหาข้อมูล ในบทความนี้ได้นำเสนอเทคนิคการทำดัชนีสำหรับการค้นหาข้อมูลในเอกสาร XML โดยจะทำการลงรหัสให้กับแต่ละอิลิเมนต์และแอททริบิวต์ ซึ่งได้นำแนวคิดของ Huffman Coding มาประยุกต์ใช้สำหรับสร้างรหัส ซึ่งจะช่วยลดเวลาในการประมวลผลการสอบถามและลดพื้นที่สำหรับจัดเก็บดัชนีและใช้ตำแหน่งเริ่มต้นและสิ้นสุดในการเข้าถึงข้อมูลในเอกสาร XML สุดท้ายได้เสนอการวิเคราะห์และเปรียบเทียบกับดัชนีที่เสนอก่อนหน้า

คำสำคัญ: indexing, XML

Abstract

Recently, data on the Internet is in the form XML Documents because XML is a standard for Internet data exchange. In the paper, we proposed a new efficient indexing technique to search data in XML Documents. We apply Huffman Coding to encode each element and attribute, leading to use less space and save processing time. Moreover, we use start_offset and end_offset to access data in XML Documents. Our Comparative study shows that the performance of the proposed technique is better than the techniques presented earlier.

Key Words: indexing, XML

1. บทนำ

ในปัจจุบันนี้ XML (Extensible Markup Language) เป็นมาตรฐานที่ใช้กันแพร่หลายในการแลกเปลี่ยนข้อมูลในระบบอินเตอร์เน็ต XML พัฒนามาจาก SGML ซึ่งเหมือนกับ HTML แต่ต่างกันตรงที่ HTML ถูกสร้างมาเพื่อแสดงผลข้อมูลผ่านเว็บเท่านั้น ซึ่งไม่สะดวกในการดึงส่วนของข้อมูลออกมา แต่สำหรับ XML นั้นเป็นภาษาที่ยืดหยุ่นต่อข้อมูลที่มีลักษณะแบบกึ่งโครงสร้าง (Semi-structure) และเป็นตัวกลางในการแปลงข้อมูลที่อยู่ในลักษณะการเก็บในรูปแบบที่แตกต่างกันทำให้ระบบต่างๆสามารถเข้าถึงและใช้งานข้อมูลร่วมกันได้ (Interoperability)

ตัวอย่างระบบงานที่นิยมใช้เอกสาร XML ในการเก็บข้อมูล เช่น การทำธุรกิจบนเว็บหรือ E-commerce ดังนั้นการเพิ่มความเร็วในการค้นหาข้อมูลต่างๆ จะส่งผลดีต่อระบบงานนั้น เพื่อเพิ่มประสิทธิภาพในการค้นหาข้อมูลในเอกสาร XML ได้มีการทำการจัดเก็บเอกสาร XML อยู่ใน 3 รูปแบบ [1,2,3] คือ 1) จัดเก็บในรูปไฟล์ (File) เป็นวิธีที่สะดวกในการสร้าง และมีการสร้างไฟล์ สำหรับเก็บดัชนีเพื่อเพิ่มความเร็วในการค้นหา 2) จัดเก็บในฐานข้อมูลที่

ออกแบบมาเฉพาะสำหรับ XML (XML Database) ซึ่งต้องนำเอกสาร XML จัดเก็บในฐานข้อมูล เป็นวิธีที่ใช้งานสะดวก และ 3) จัดเก็บในฐานข้อมูลโครงสร้างเชิงสัมพันธ์ (RDBMS) จะเป็นการแปลง (Transform) โครงสร้างเอกสาร XML (XML Schema) ไปเป็นโครงสร้างตารางใน RDBMS วิธีนี้เสียเวลาในการแปลงข้อมูลในเอกสาร XML และการสอบถาม ถึงแม้ว่า RDBMS ได้รับการพัฒนามายาวนานปัจจุบันจึงเป็นที่นิยมใช้ในองค์กร แต่ในการเก็บข้อมูลนั้น ถูกแยกเป็นตารางเล็กๆ ทำให้เพิ่มพื้นที่การจัดเก็บ และในการค้นหาต้องใช้การดำเนินการ join ที่มากขึ้น ทำให้การประมวลผลการสอบถามไม่มีประสิทธิภาพ

ในบทความนี้มุ่งไปที่การจัดเก็บเอกสาร XML ในรูปไฟล์ (ตามรูปแบบที่ 1) และนำเสนอเทคนิคการทำดัชนีแบบใหม่เพื่อเพิ่มประสิทธิภาพในการค้นหาข้อมูลในเอกสาร XML

เอกสารนี้ประกอบด้วย 6 ส่วน โดยส่วนที่ 2 อธิบายถึงทฤษฎีที่เกี่ยวข้อง ส่วนที่ 3 จะกล่าวถึงงานวิจัยที่เกี่ยวข้องกับการทำดัชนีบนเอกสาร XML ส่วนที่ 4 เป็นการนำเสนอเทคนิคการทำดัชนี XTI ส่วนที่ 5 เป็นการวิเคราะห์ประสิทธิภาพของดัชนี และสุดท้ายส่วนที่ 6 เป็นบทสรุป

2. ทฤษฎีที่เกี่ยวข้อง

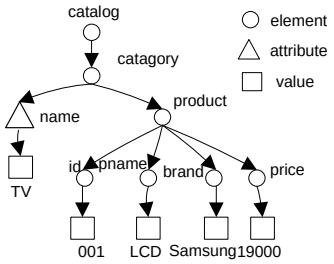
ในส่วนนี้จะเป็นการกล่าวถึง 1) ภาษา XML และ 2) XPath อย่างสังเขป

2.1 XML (Extensible Markup Language)

ภาษา XML เป็นวิธีการระบุเนื้อหาโดยใช้แท็ก (tag) ในการอธิบายความหมายของข้อมูลที่ผู้ใช้สามารถกำหนดแท็กที่ใช้งานได้เอง เอกสาร XML ประกอบด้วยแท็กต่างๆ คือแท็กเริ่มต้นจะอยู่ภายใต้เครื่องหมาย < และ > ส่วนแท็กสิ้นสุดจะอยู่ภายใต้เครื่องหมาย </ และ > ในเอกสาร XML ประกอบด้วย 3 ส่วน คือ 1) อิลิเมนต์ (Element) เป็นส่วนของแท็กเริ่มต้นไปจนถึงแท็กสิ้นสุด 2) แอทริบิวต์ (Attribute) เป็นการระบุคุณสมบัติให้กับอิลิเมนต์เพื่ออธิบายส่วนเพิ่มเติม และ 3) ค่าข้อมูล (Value) เป็นข้อมูลที่เก็บในเอกสาร XML รูปที่ 1 แสดงตัวอย่างเอกสาร XML และ XML Tree Model ประกอบด้วยอิลิเมนต์ catalog โหนดราก ซึ่งมีอิลิเมนต์ category (มีแอททริบิวต์ชื่อ name) ภายในประกอบด้วยอิลิเมนต์อีกชั้นคือ product และใน product มีอิลิเมนต์ต่างๆ เช่น id ซึ่งมีค่าข้อมูลเท่ากับ 001 รายละเอียดของภาษา XML สามารถดูได้จาก [4,5]

```
<catalog>
  <category name = "TV">
   <product>
     <id>001</id>
     <pname>LCD</pname>
     <brand>Samsung</brand>
     <price>19000</price>
   </product>
  </category>
</catalog>
```

(a) ตัวอย่างเอกสาร



(b) ตัวอย่าง XML Tree Model

รูปที่ 1 ตัวอย่างเอกสาร XML และ XML Tree Model

2.2 XPath

XPath [6] เป็นภาษาที่ใช้ในการสอบถามข้อมูลโดยการระบุตำแหน่งของโหนดหรืออิลิเมนต์ในเอกสาร XML ใช้ path expression ในการเข้าถึงข้อมูล ตัวอย่างเช่น /catalog/category[@name = TV']/product[pname = 'LCD'] เป็นการสอบถามรายละเอียดทั้งหมดของ TV ที่เป็น LCD

สำหรับรูปแบบการสอบถามข้อมูลในเอกสาร XML แบ่งออกได้เป็น 3 รูปแบบ [7, 14]

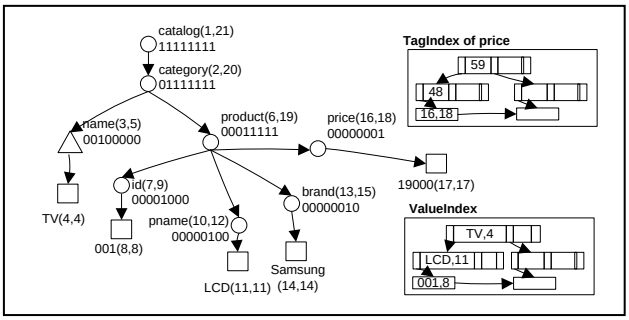
- การค้นหาตามแกน parent-child เป็นการค้นหาที่เริ่มต้นการค้นหาตั้งแต่โหนดรากของต้นไม้ และจะท่องต้นไม้ทีละระดับไปตามแกน parent-child เช่น catalog/category/product/pname
- การค้นหาตามแกน ancestor-descendant เป็นการสอบถามที่ไม่ต้องไปทีละระดับของต้นไม้ เช่น catalog/category//pname หรือไม่ต้องเริ่มจากโหนดราก เช่น //category//pname

- การค้นหาที่มีการระบุเงื่อนไข เช่น catalog/category/product/pname = ‘LCD’/[price]

3. งานวิจัยที่เกี่ยวข้อง

มีงานวิจัยจำนวนมากได้เสนอเทคนิคการทำดัชนีกับเอกสาร XML ที่เก็บอยู่ในรูปแบบไฟล์ ซึ่งนิยมทำดัชนีใน XML Tree เช่น 1-index และ 2-index [8] เทคนิคนี้จะทำการค้นหาข้อมูลตั้งแต่โหนดรากเสมอ ทำให้เสียเวลามาก ดังนั้น Jason และ คณะ[9] ได้นำเสนอเทคนิคแบบ Bindex และ Lindex เพื่อช่วยลดเวลาในการค้นหาตั้งแต่โหนดราก แต่ใช้พื้นที่ในการเก็บจำนวนมาก ต่อมา Brian และคณะ [10] ได้นำแนวคิดของ Paricia tree มาใช้สำหรับทำดัชนี เรียกว่า Index Fabric ซึ่งมีประสิทธิภาพสำหรับการสอบถามที่อยู่ในรูปแบบ parent - child

งานวิจัย [7, 11, 12] ได้นำเสนอการทำดัชนีบนโครงสร้างต้นไม้ด้วยวิธีการลงรหัสเรียกว่า Signature วิธีนี้ช่วยลดเวลาในการประมวลผลการสอบถามได้ เพราะใช้การดำเนินการตรรกะ (Boolean operation) เช่น AND, OR ในระดับบิตทำให้ลดการเข้าถึงโหนดที่ไม่จำเป็นได้ งานวิจัยของ Sangwon และคณะ [11] และงานวิจัยของ Chung และคณะ [12] ใช้ Hash function ในการสร้าง Signature ให้กับทุกโหนด ทำให้ Signature ที่ได้มีความยาวมาก มีข้อเสียคือกรณีที่เป็นการค้นหาตามแกน ancestor – descendant ไม่สามารถลดจำนวนโหนดที่ต้องเข้าถึงได้ เพื่อแก้ปัญหานี้ ในปี 2008 Huang และคณะ [7] นำเสนอเทคนิคการค้นหาข้อมูลในเอกสาร XML ด้วยการใช้ Signature ร่วมกับอีก 2 ดัชนีซึ่งมีโครงสร้างแบบ B^+Tree เรียกว่าดัชนี TagIndex และ ValueIndex ซึ่งดัชนีทั้ง 2 นี้จะใช้ containing code (ลำดับการท่อง tree แบบ DFS) ใช้เป็น key ในการค้นหา งานวิจัยนี้สร้าง Signature ความยาว *n* บิต (*n* แทนจำนวนอิลิเมนต์และแอททริบิวต์ทั้งหมด) ในทุก ๆ โหนดอิลิเมนต์และแอททริบิวต์ ก่อนสร้าง Signature ต้องกำหนดตำแหน่งที่เป็นบิต 1 ให้แต่ละโหนดอิลิเมนต์/แอททริบิวต์ โดยให้แต่ละบิตแทนแต่ละค่าของอิลิเมนต์หรือแอททริบิวต์ ตัวอย่างเช่น เอกสาร XML ในรูปที่ 1 มีจำนวนอิลิเมนต์และแอททริบิวต์เท่ากับ 8 ดังนั้น แต่ละโหนดอิลิเมนต์/แอททริบิวต์จะถูกแทนด้วย 8 บิต บิตที่ 1 แทนcatalog ดังนั้นรหัส 10000000 จะถูกกำหนดในโหนด catalog บิตที่ 2 แทน category รหัส 01000000 จะถูกกำหนดในโหนด category บิตที่ 4 แทน product รหัส 00010000 จะถูกกำหนดในโหนด product เป็นต้น การสร้าง Signature ของแต่ละโหนดอิลิเมนต์/แอ-ททริบิวต์ต้องดำเนินการตรรกะ OR ระหว่างรหัสของโหนดตัวเองกับโหนดลูกหลาน เช่น product มีรหัสเป็น 00011111 ซึ่งได้จากการดำเนินการตรรกะรหัสจาก 5 โหนด กล่าวคือ โหนด product (00010000) โหนด id (00001000) โหนด pname (00000100) โหนด brand (00000010) และโหนด price (00000001) ในรูปที่ 2 แสดง XML Storage Model TagIndex และ ValueIndex จากเอกสาร XML ในรูปที่ 1



รูปที่ 2 XML Storage Model TagIndex และ ValueIndex

ถึงแม้ว่า Signature จะช่วยให้การประมวลผลเร็วขึ้นแต่เพื่อที่จะปรับปรุงประสิทธิภาพการค้นหาข้อมูลสำหรับเอกสาร XML บทความนี้ได้นำเสนอเทคนิคการทำดัชนี XTI โดยใช้จำนวนบิตในการลงรหัสให้กับอิลิเมนต์และแอททริบิวต์น้อยกว่างานวิจัยที่เคยมีมา เป็นการนำแนวคิดของ Huffman Coding มาประยุกต์ใช้สำหรับการลงรหัสเพื่อลดพื้นที่ในการจัดเก็บและเวลาในการประมวลผล และใช้ตำแหน่งเริ่มต้นและสิ้นสุดในการเข้าถึงข้อมูลในเอกสาร XML

4. เทคนิคการทำดัชนี XTI (XML Tree Index)

ในส่วนนี้เรานำเสนอเทคนิคการทำดัชนี XTI ซึ่งประกอบ ด้วยกระบวนการสร้างดัชนี XTI (XML Tree Index Creation) และกระบวนการสอบถามข้อมูลสำหรับเอกสาร XML (XML Query Processing)

4.1 XML Tree Index Creation

ในขั้นตอนนี้เราจะทำการสร้าง XML Tree Index ซึ่งในแต่ละโหนดของ Tree มีโครงสร้างข้อมูลแตกต่างกันตามชนิดของโหนด นั่นคือระหว่างโหนดภายใน (Internal Node) หรือโหนดภายนอก (External Node หรือ Leaf Node)

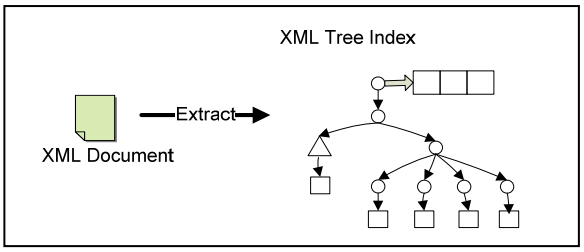
1. ทำการสร้าง XML Tree จากเอกสาร XML โดยกำหนดโครงสร้างข้อมูลของแต่ละ Internal Node ของ XML Tree ประกอบด้วย 3 fields ดังนี้

- *code* หมายถึง รหัสที่กำหนดให้แก่โหนดอิลิเมนต์หรือแอททริบิวต์ โดยใช้แนวคิดของ Huffman Coding ในการสร้างรหัสให้ทุกโหนดที่ลงรหัสมีน้ำหนักเท่ากัน จากเอกสาร XML ในรูปที่ 1 มีจำนวนอิลิเมนต์และแอททริบิวต์เท่ากับ 8 ได้ความยาวของรหัสเป็น $\log_2 8 = 3$ ตารางที่ 1 แสดงรหัสของแต่ละโหนด

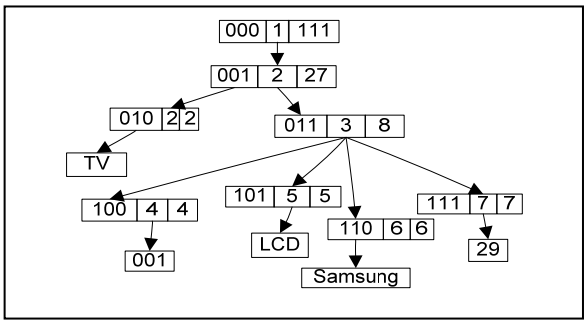
- *start_offset* หมายถึง ตำแหน่งเริ่มต้นของอิลิเมนต์หรือแอททริบิวต์

- *end_offset* หมายถึง ตำแหน่งสิ้นสุดของอิลิเมนต์หรือแอททริบิวต์

สำหรับโครงสร้างข้อมูลที่ Leaf Node ของ XML Tree จะประกอบด้วย Field เดียวเท่านั้นคือ name หมายถึงค่าข้อมูล ในรูปที่ 3 แสดงขั้นตอนการสร้าง XML Tree Index โดยนำเอกสาร XML ผ่านกระบวนการ Extract เพื่อสร้างเป็น XML Tree Index และในรูปที่ 4 แสดง XML Tree Index ที่สร้างจากเอกสาร XML ในรูปที่ 1



รูปที่ 3 ขั้นตอนการสร้าง XML Tree Index

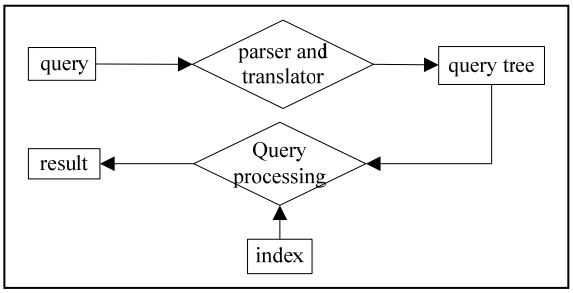


รูปที่ 4 XML Tree Index ของเอกสาร XML ในรูปที่ 1

ตารางที่ 1 Mapping Table

ชื่อโหนด	รหัส (Code)
catalog	000
category	001
name	010
id	100
product	011
pname	101
brand	110
price	111

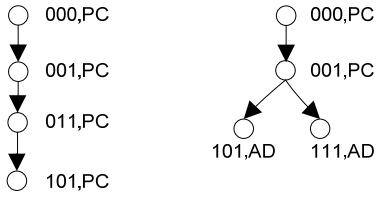
4.2 XML Query Processing



รูปที่ 5 ขั้นตอนการประมวลผล

ในส่วนนี้กล่าวถึงขั้นตอนการประมวลผลการสอบถามแสดงขั้นตอนการทำงานได้ดังรูปที่ 5 ซึ่งมีรายละเอียดดังนี้

1. แปลงการสอบถามให้อยู่ในรูปของต้นไม้การสอบถาม (Query Tree : Qtree) โดยแต่ละโหนดของ Qtree มีโครงสร้างข้อมูลที่ประกอบด้วยสอง Fields คือ code หมายถึงรหัสของโหนดอิลิเมนต์หรือแอททริบิวต์ดังตารางที่ 1 และ type หมายถึงชนิดแกนของแต่ละโหนดที่ใช้สำหรับวิเคราะห์เพื่อทำการค้นหา มี 3 แบบคือ 1) PC เป็น parent-child axis 2) AD เป็น ancestor-descendant axis หรือ 3) V เป็นการแสดงว่าโหนดนี้เป็น Value รูปที่ 6(a) และ 6(b) แสดงQuery Tree ของการสอบถาม catalog/category/product/pname และ catalog/category//pname/[price] ตามลำดับ



(a) catalog/category/product/pname(b) catalog/category//pname/[price]

รูปที่ 6 ตัวอย่าง Query Tree

2. ทำการค้นหา Qtree ใน XTI แบบ DFS (Depth First Search) โดยเปรียบเทียบทีละโหนดของ Qtree กับโหนดของ XTI เริ่มจากโหนดรากของทั้งสองจนถึง Leaf Node ของ Qtree โดยใช้ขั้นตอนวิธีการค้นหาข้อมูลในรูปที่ 7 อธิบายได้ดังนี้

2.1 ในแต่ละโหนดของ Qtree เช็คดูว่าฟิลด์ type มีชนิดแกนเป็นแบบไหน

- ถ้าเป็น PC ให้ทำการดำเนินการตรรกะ XOR ระหว่าง code ของ Qtree กับ code ของโหนดปัจจุบันใน XTI ถ้าผลลัพธ์ที่ได้เป็น 0 แสดงว่าใช่โหนดที่กำลังค้นหาอยู่ (ดูบรรทัดที่ 3-4)

- ถ้าเป็น AD คำนวณจำนวนโหนดลูกของโหนดใน XTI ที่โหนดปัจจุบัน จากนั้นให้ข้ามโหนดนั้นไป 1 ระดับ แล้วทำการดำเนินการตรรกะ XOR ระหว่าง code ของ Qtree กับ code ของโหนดลูกของโหนดปัจจุบันใน XTI จนหมดทุกโหนด (ดูบรรทัดที่ 5-7)

- ถ้าเป็น V ให้ทำการเปรียบเทียบกับค่าข้อมูลใน Qtree กับ ใน XTI (ดูบรรทัดที่ 8-9)

2.2 ถ้าผลลัพธ์ที่ได้มีค่าเท่ากับ code ของโหนดใน Qtree ให้ขยับไปยังโหนดลูกของ Qtree พร้อมกับขยับไปยังโหนดลูกของ XTI แล้วกลับไปทำข้อ 2.1 แต่ถ้าผลลัพธ์ที่ได้มีค่าไม่เท่ากับ code ของ Qtree ให้กลับไปทำข้อ 2 โดยไม่ต้องขยับโหนดใน Qtree แต่ให้ขยับโหนดใน XTI

2.3 ถ้าถึง Leaf Node ของ Qtree ให้ทำการดึงค่า start_offset และ end_offset จากโหนดใน XTI ออกมา

```
QueryPrecessing
// q = Qtree, t = XTI
   1 Build Qtree from query
   2 while q and t is not empty
   3 Case : q.type = PC
   4    FindOffset(q,t)
   5 Case : q.type = AD
   6    t ← getDecendant(t) //advance one level
   7    FindOffset(q, t)
   8 Case : q.type = V
   9    FindOffset(q, t)
```

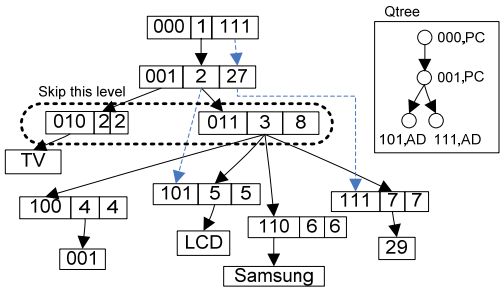
```
FindOffset(q,t)
   1 if (q.code XOR t.code =0)
   2   if (q is leaf node)
   3       OutputOffset(t)
   4   else
   5       t← get node by DFS
   6       q← get node by DFS
   7       FindOffset(q,t)
   8 else
   9       t← get node by DFS
   10      FindOffset(q,t)
```

รูปที่ 7 ขั้นตอนวิธีการค้นหาข้อมูล

ตัวอย่างการสอบถาม catalog/category//pname/[price] สร้างเป็น Qtree ได้ดังรูปที่ 6(b) โหนดรากของ Qtree มี type เป็น PC จากนั้นนำ code ของโหนดใน Qtree ซึ่งมี code เป็น 000 มาดำเนินการตรรกะ XOR กับ code ของโหนดใน XTI ซึ่งมี code เป็น 000 ผลลัพธ์ที่ได้คือ 000 (000 XOR 000) ซึ่งมีค่าเป็น 0 แสดงว่าใช่โหนดที่ต้องการ จากนั้นเปรียบเทียบโหนดลูกของ Qtree กับโหนดลูก

โหนดแรกของ XTI จนผลลัพธ์ที่ได้จากการดำเนินการตรรกะมีค่าเป็น 0 ต่อมาพบว่าโหนดใน Qtree มีโหนดลูก 2 โหนด และมี type เป็น AD ให้คำนวณจำนวนโหนดลูกใน XTI ปัจจุบัน ซึ่งมี 2 โหนด และคำนวณจำนวนโหนดหลานในโหนดลูกแรก ซึ่งมี 1 โหนด

เริ่มทำการเปรียบเทียบโหนดลูกตัวแรกของ Qtree มี code เป็น 101 กับโหนดหลานโหนดแรกในโหนดลูกตัวแรกของ XTI (ข้ามโหนดใน XTI ไป 1 level) ซึ่งไม่มี code แสดงว่าเป็นคนละชนิดข้อมูลกันไม่สามารถเปรียบเทียบได้ จากนั้นขยับมาคำนวณจำนวนโหนดหลานในโหนดลูกที่สอง (โหนด product) ของ XTI พบว่ามี 4 โหนด เริ่มทำการเปรียบเทียบกับโหนดหลานแรกในโหนดลูกตัวที่สองของ XTI มี code เป็น 100 ผลลัพธ์ที่ได้เป็น 001 (101 XOR 100) ซึ่งมีค่าไม่เท่ากับ 0 แสดงว่าไม่ใช่โหนดที่ต้องการ จึงไม่ต้องขยับโหนดใน Qtree แต่ขยับไปยังโหนดหลานที่สองในโหนดลูกที่สองของ XTI แล้วทำการเปรียบเทียบเหมือนเดิมไปเรื่อยๆ จนพบว่า โหนดใน Qtree และถ้าโหนดนั้นเป็น Leaf Node จึงทำการดึงค่า start_offset และ end_offset ซึ่งเป็นตำแหน่งบรรทัดในเอกสาร XML และสุดท้ายพิจารณาว่าหมดโหนดใน Qtree และ XTI แล้วจึงหยุดการทำงาน ในรูปที่ 8 แสดงตัวอย่างการค้นหาข้อมูลใน XTI ของตัวอย่างที่ได้อธิบายข้างต้น

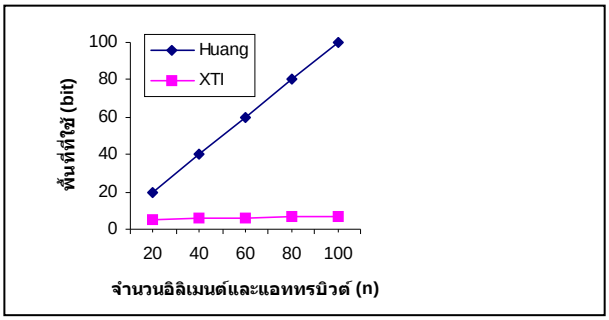


รูปที่ 8 ตัวอย่างการค้นหาข้อมูล

5. การวิเคราะห์ประสิทธิภาพ

ในหัวข้อนี้จะทำการเปรียบเทียบประสิทธิภาพในเรื่องพื้นที่สำหรับเก็บดัชนีและเวลาในการประมวลผลการสอบถามระหว่างดัชนีของ Huang และ คณะ [7] กับดัชนีที่บทความนี้นำเสนอ

พิจารณาพื้นที่สำหรับเก็บดัชนี สำหรับดัชนีที่ในบทความนี้นำเสนอเป็นการสร้าง XML Tree Index เพียงดัชนีเดียว สมมติให้ n เป็นจำนวนของ Internal node ในบทความนี้นำแนวคิดของ Huffman Coding มาประยุกต์ใช้สำหรับการลงรหัส ซึ่งใช้พื้นที่สำหรับเก็บรหัสเพียง $O(\log_2 n)$ ส่วนวิธีของ Huang และ คณะ มี 3 ดัชนี คือ XML Storage Model (XML Tree with Signature) TagIndex และ ValueIndex (ใช้ B^+tree) ในการลงรหัสใช้ n บิตแทนหนึ่งโหนด ดังนั้นใช้พื้นที่ทั้งหมด $O(n)$ พิจารณารูปที่ 9 แสดงความสัมพันธ์ระหว่างความยาวที่ใช้สร้างรหัสกับจำนวนของอิลิเมนต์และแอททริบิวต์และแสดงการเปรียบเทียบพื้นที่สำหรับเก็บดัชนี XTI กับวิธีของ Huang และคณะ



รูปที่ 9 แสดงการเปรียบเทียบพื้นที่สำหรับเก็บดัชนี XTI กับวิธีของ Huang และ คณะ

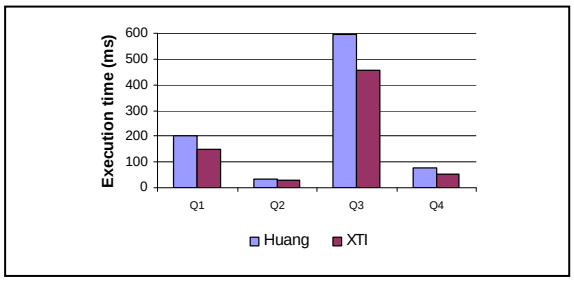
พิจารณาเรื่องเวลาที่ใช้ในการประมวลผลการสอบถามโดยพัฒนาโปรแกรมด้วยภาษาจาวา (JDK 1.6) บนเครื่องคอมพิวเตอร์ Intel® Core™2 Quad ระบบปฏิบัติการ Window XP Professional หน่วยความจำ 2 GB ฮาร์ดดิสก์ 320 GB ใช้ชุดข้อมูลจาก XMark XML Benchmark [13] มีการสอบถามทั้งหมด 4 การสอบถาม ดังตารางที่ 2

ผลการทดลองแสดงดังรูปที่ 10 ซึ่งเห็นได้ว่าเทคนิคการทำดัชนีแบบ XTI ใช้เวลาน้อยในทุกการสอบถาม

โดยเฉพาะ Q3 เนื่องจากดัชนีของ Huang และคณะ ต้องใช้ TagIndex ช่วยเสริมในการประมวลผล สำหรับ Q1 และ Q2 ใช้ XTI ในการค้นหาเหมือนกันแต่พบว่า XTI ใช้เวลาน้อยในการประมวลผล Q1 เพราะว่าเป็นการเข้าถึงข้อมูลในระดับที่ไม่ลึกมาก ซึ่งไม่เหมือนกับ Q2 ที่ต้องท่องต้นไม้ไปถึงระดับ Leaf Node

ตารางที่ 2 การสอบถาม

Q1	/site/regions/africa/item
Q2	/site/people/person/name/[phone]
Q3	/site/closed_auctions/closed_auction//author
Q4	/site/regions/Africa/item/mailbox//to/[date]



รูปที่ 10 เวลาในการประมวลผลการสอบถาม

6. บทสรุป

เพื่อลดพื้นที่สำหรับเก็บดัชนีและเพิ่มความเร็วในการประมวลผลการสอบถาม ในบทความนี้ได้นำเสนอเทคนิคการทำดัชนีแบบใหม่สำหรับการค้นหาข้อมูลในเอกสาร XML ซึ่งสร้างเพียงดัชนีเดียวเท่านั้นเรียกว่า XTI (XML Tree Index) โดยจะทำการลงรหัสให้กับอิลิเมนต์และแอททริบิวต์ ซึ่งนำแนวคิดของ Huffman Coding มาใช้สำหรับสร้างรหัส และใช้ตำแหน่งเริ่มต้นและสิ้นสุดในการเข้าถึงข้อมูลในเอกสาร XML

7. เอกสารอ้างอิง

[1] H. Jiang, H. Lu, W. Wang, and J. X. Yu, “XParent: An Efficient RDBMS-Based XML Database System”, 2002.

[2] J. Shanmugasundaram, K. Tufte, and G. He, “Relational Databases for Querying XML Documents:Limitations and Opportunities”, Proceedings of the 25th VLDB Conference, 1999.

[3] F. Tain, D.J. Dewit, J. Chen and C. Zhang, “The Design and Performance Evaluation of Alternative XML Storage Strategies”, ACM SIGMOD Record, Vol. 31, No. 1, March 2002.

[4] R. Cover, “The SGML/XML Web Page”, http://www.oasis-open.org/cover/xml.html.

[5] T. Bray, J. Paoli, C. M. Sperberg-McQueen, “Extensible Markup Language (XML) 1.0”, Availablefrom: http://www.w3.org/TR/REC-xml.

[6] W3C Recommendation, XML Path Language (Xpath) 1.0, Available from: http://www.w3.org/TR/xpath , 1999.

[7] Y. F. Huang and S. H. Wang, “An Efficient XML Query Processing Based on Combining T-Bitmap and Index Techniques”, ISCC 2008, pp. 858-863, 2008.

[8] T. Milo, D. Suciu, “Index structures for pathexpressions”, in: Proceedings of the InternationalConference on Database Theory, 1999, pp. 277–295.

[9] J. McHugh, S. Abiteboul, R. Goldman, D. Quass, J.Widom, “Lore: A database management system forsemi structured data”, SIGMOD Record 26 (3), 1997, pp. 54–66.

[10] B. F. Cooper, N. Sample, M. J. Franklin, G.R.Hjaltason and M.Shadmon, “A Fast Index for Semistructured Data”, Proceedings of the 27th VLDB Conference, Roma, Italy, 2001.

[11] S. Park and H. J. Kim, “A New Query Processing Technique for XML Based on Signature”, DASFAA ‘01, pp.22, 2001.

[12] Y. D. Chung, J.W. Kim and M. H. Kim, “Efficient preprocessing of XML queries using structured signatures”, pp. 257-264, 2003.

[13] An XML Benchmark Project (XMark), http://monetdb.cwi.nl/xml.

[14] S. Pal, I. Cseri, O. Seeliger, G. Schaller, L. Giakoumakis, V. Zolotov, “Indexing XML Data Stored in a Relational Database”, Proceedings of the 30th VLDB Conference, Toronto, Canada, 2004.

ประวัติผู้เขียน

ชื่อ สกุล นางสาววรารัตน์ จักรหวัด

รหัสประจำตัวนักศึกษา 5110220070

วุฒิการศึกษา

วุฒิ	ชื่อสถาบัน	ปีที่สำเร็จการศึกษา
วท.บ. (วิทยาการคอมพิวเตอร์)	มหาวิทยาลัยสงขลานครินทร์	2548

การตีพิมพ์เผยแพร่ผลงาน

วรารัตน์ จักรหวัด ศิริรัตน์ วณิชโยบล และปรีชา วงศ์หิรัญเดชา. 2552. การเพิ่มประสิทธิภาพการค้นหาข้อมูลในเอกสาร XML ของแคตตาล็อกสินค้าอิเล็กทรอนิกส์. The 6th Joint Conference on Computer Science and Software Engineering (JCSSE2009). ภูเก็ต ประเทศไทย, 13-15 พฤษภาคม 2552. หน้า 365-370.

วรารัตน์ จักรหวัด และศิริรัตน์ วณิชโยบล. 2553. เทคนิคการเพิ่มประสิทธิภาพการทำดัชนีสำหรับเอกสาร XML. The 7th Joint Conference on Computer Science and Software Engineering (JCSSE2010). กรุงเทพมหานคร ประเทศไทย, 12-14 พฤษภาคม 2553. หน้า 250-256.