

บุญรัตน์ เผลิมรอด 2551: วิธีการใหม่สำหรับสืบค้นแบบหลายความสัมพันธ์บนองค์ความรู้ในฐานข้อมูลเชิงสัมพันธ์ ปริญญาวิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมคอมพิวเตอร์) สาขาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ ประชานุกรมการที่ปรึกษา: รองศาสตราจารย์กฤษณะ ไวยมัย, Ph.D. 64 หน้า

ออนโทโลยีเป็นแนวทางจัดความรู้ แนวคิดในขอบเขตความสนใจให้เป็นมาตรฐาน สามารถนำมาใช้ใหม่ และแลกเปลี่ยนกันได้ ในปัจจุบันออนโทโลยีถูกนำมาประยุกต์ใช้ในระบบสารสนเทศหลายแนวทาง โดยเฉพาะอย่างยิ่งระบบสารสนเทศทางด้านวิทยาศาสตร์สิ่งมีชีวิต เช่น ทางด้านชีววิทยา ทางด้านการแพทย์ ระบบเหล่านี้ประยุกต์ใช้ออนโทโลยีซึ่งจัดกลุ่มความรู้อย่างมีโครงสร้าง เป็นความรู้พื้นฐานสำหรับอธิบายข้อมูล และการสืบค้นข้อมูลเชิงความหมาย

งานวิจัยนี้เสนอแนวทางการสืบค้นข้อมูลในฐานข้อมูลเชิงสัมพันธ์โดยประยุกต์ใช้ออนโทโลยีที่มีหลากหลายความสัมพันธ์เป็นฐานความรู้สำหรับการสืบค้นเชิงความหมาย โดยจัดเก็บออนโทโลยีในฐานข้อมูลเชิงสัมพันธ์รูปแบบความสัมพันธ์เชิงวัตถุที่พิจารณาถึงการอ้างอิงความสัมพันธ์ระหว่างโหนดในออนโทโลยีที่มีหลากหลายความสัมพันธ์ และพัฒนา ONT RELATED โอเปอเรเตอร์ซึ่งจะแปลงคำสืบค้นของผู้ใช้ให้เป็นคำสืบค้นเชิงความหมายที่ใกล้เคียงกับคำสั่งของผู้ใช้มากที่สุด โดยโอเปอเรเตอร์นี้จะพิจารณากลุ่มคำในออนโทโลยีที่สอดคล้องกับคำสั่งสืบค้นตามความสัมพันธ์ที่กำหนด และนำกลุ่มคำที่ได้มาสืบค้นในฐานข้อมูลจากผลการทดลองเปรียบเทียบกับงานวิจัยการสืบค้นข้อมูลในฐานข้อมูลเชิงสัมพันธ์โดยประยุกต์ใช้ออนโทโลยีที่พัฒนาโดยใช้คำสั่ง “START WITH” และ “CONNECT BY” สรุปได้ว่าโอเปอเรเตอร์ที่เสนอในงานวิจัยสามารถสืบค้นข้อมูลได้อย่างมีประสิทธิภาพ และให้ผลการสืบค้นที่ครบถ้วน

ឧបនិព្វាន ដេតិមន្តិកា

ถายมือชอนิติต



ตายมือชื่อประธานกรรมการ

29 / 02 / 2001

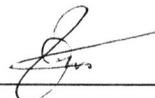
Boonyarat Phadermrod 2008: A Novel Multi-Relation Querying Method on RDBMS-Based Ontology. Master of Engineering (Computer Engineering), Major Field: Computer Engineering, Department of Computer Engineering. Thesis Advisor: Associate Professor Kitsana Waiyamai, Ph.D. 64 pages.

Ontology provides a commonly agreed understanding of a domain. It can be reused and shared across applications. In life science such as biology and medicine, ontology is seen as information system domain knowledge. Within these applications, ontology is merely a structured vocabulary in the form of a tree or directed acyclic graph of concepts, which can be used as background knowledge for describing and supporting semantic-based search of data.

In this research work, we propose a novel multi-relation querying method on RDBMS-based ontology. We develop a technique for storing and representing ontology in RDBMS, and use ontology as background knowledge for performing semantic-based query. The ontology is stored and represented in a form of object-oriented relation with semantic relation derivation between ontology concepts. ONT_RELATED operator is implemented for transforming user-query to an equivalent semantic query. It returns ontology concepts related to the user-query for searching in a database. Experimental results against ontology-based search system with the use of "START WITH" and "CONNECT BY" clauses show that ONT_RELATED operator gives better performances with respect to the processing time and the search result completeness.

Boonyarat Phadermrod

Student's signature



Thesis Advisor's signature

21 / 02 / 2008

วิธีการใหม่สำหรับสืบค้นแบบหลายความสัมพันธ์บนองค์ความรู้ ในฐานข้อมูลเชิงสัมพันธ์

A Novel Multi-Relation Querying Method on RDBMS-Based Ontology

คำนำ

ในปัจจุบันการจัดหมวดหมู่ความรู้ในขอบเขตความสนใจหนึ่ง ๆ หรือออนโทโลยี (Ontology) มีความสำคัญต่อการพัฒนาแอปพลิเคชันในด้านต่าง ๆ ให้มีประสิทธิภาพมากขึ้น โดยเฉพาะการพัฒนาแอปพลิเคชันสำหรับสืบค้นข้อมูลที่ประยุกต์ใช้ออนโทโลยี เพื่อสืบค้นข้อมูลจากแหล่งข้อมูลต่าง ๆ เช่น เอกสาร เว็บเพจ เป็นต้น เนื่องจากออนโทโลยีสามารถนำไปใช้ในการแปลงคำสืบค้นของผู้ใช้ให้อยู่ในรูปแบบคำสั่งสืบค้นเชิงความหมาย จึงได้ผลการสืบค้นที่ครบถ้วน และตรงกับความต้องการของผู้ใช้

ในฐานข้อมูลเชิงสัมพันธ์ปัจจุบัน สืบค้นข้อมูลโดยใช้คำสืบค้นจากผู้ใช้งานเพียงเท่านั้น (Keyword search) ไม่ได้พิจารณาคำสืบค้นเชิงความหมาย เพื่อเพิ่มประสิทธิภาพการสืบค้นในฐานข้อมูลเชิงสัมพันธ์ งานวิจัยนี้เสนอแนวทางการประยุกต์ใช้ออนโทโลยีเพื่อเพิ่มประสิทธิภาพการสืบค้นข้อมูลในฐานข้อมูลเชิงสัมพันธ์ โดยจัดเก็บออนโทโลยีในฐานข้อมูลเชิงสัมพันธ์ในรูปแบบความสัมพันธ์เชิงวัตถุ (Object-Oriented Relation) (Kachai and Waiyamai, 2001) ซึ่งจะทำให้การจัดการออนโทโลยี และการอ้างอิงออนโทโลยีเพื่อการสืบค้นเชิงความหมายในฐานข้อมูลเชิงสัมพันธ์ทำได้ง่ายขึ้นจากการใช้คำสั่ง SQL (Structure Query Language) ในฐานข้อมูล และพัฒนาโอเปอเรเตอร์สำหรับสืบค้นเชิงความหมายที่แปลงคำสั่งสืบค้นจากผู้ใช้งานให้เป็นคำถามสืบค้นเชิงความหมายจากการพิจารณาคำสืบค้นที่สอดคล้องกับออนโทโลยีตามความสัมพันธ์ที่กำหนด โดยโอเปอเรเตอร์นี้สามารถสืบค้นหาโหนดทั้งหมดที่มีความสัมพันธ์กับโหนดในออนโทโลยีที่ต้องการสืบค้นได้โดยไม่ต้องทำการสืบค้นแบบวนซ้ำตามโครงสร้างลำดับชั้นของออนโทโลยี เหมือนกับวิธีการสืบค้นโดยคำสั่ง “START WITH” และ “CONNECT BY” ที่ค้นหาแบบเชิงลึก (Depth-First search) การสืบค้นจึงไม่มีประสิทธิภาพเท่าที่ควรเมื่อระดับชั้นในออนโทโลยีที่สืบค้นสูงขึ้น นอกจากนี้งานวิจัยนี้ยังเสนอแนวทางการพิจารณาความสัมพันธ์ระหว่างโหนดในออนโทโลยีที่มีหลากหลายความสัมพันธ์ และกำหนดความสัมพันธ์ที่เหมาะสมที่สุดเพื่อให้ได้ผลการสืบค้นที่ครบถ้วน เมื่อใช้ออนโทโลยีคอนเซพเหล่านี้แทนคำสั่งสืบค้นจากผู้ใช้งาน

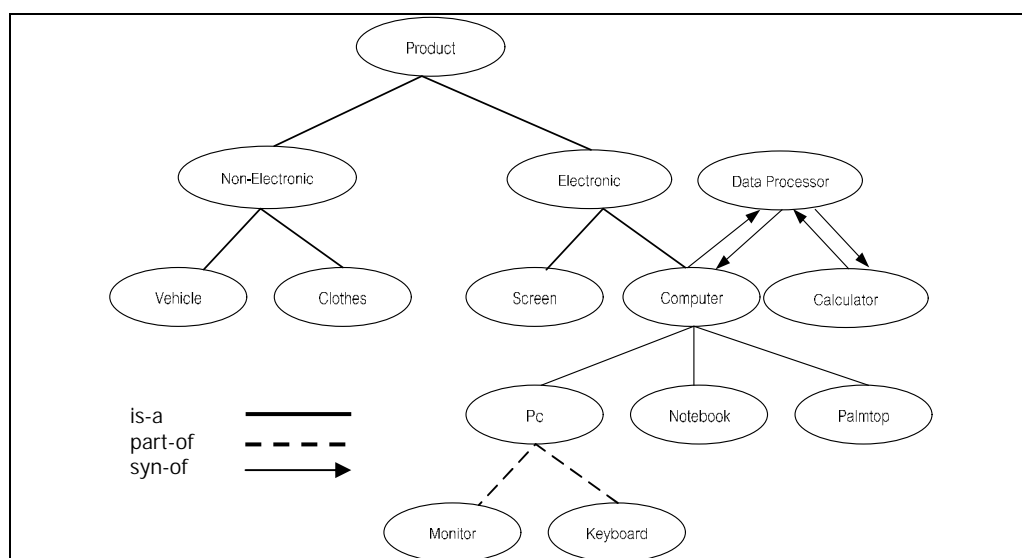
วัตถุประสงค์

1. ออกแบบวิธีการจัดเก็บออนโทโลยีในฐานข้อมูลเชิงสัมพันธ์ เพื่อให้สามารถสืบค้นข้อมูลเชิงความหมายในฐานข้อมูลเชิงสัมพันธ์ได้
2. ออกแบบวิธีการกำหนดความสัมพันธ์ระหว่างโหนดในออนโทโลยีที่มีหลากหลายความสัมพันธ์ เพื่อให้ได้ผลการสืบค้นเชิงความหมายที่ครบถ้วน
3. ออกแบบโอเปอเรเตอร์สืบค้นเชิงความหมายที่ใช้งานร่วมกับ SQL เพื่อหาคำตอบของโอเปอเรเตอร์ในฐานข้อมูลเชิงสัมพันธ์ ให้มีประสิทธิภาพในการทำงานดีกว่าการทำงานของโอเปอเรเตอร์ในฐานข้อมูลเชิงสัมพันธ์ทั่วไป

การตรวจเอกสาร

ออนโทโลยี

ออนโทโลยีเป็นการจัดแบ่งหมวดหมู่ความรู้ในขอบเขตความสนใจหนึ่ง ๆ (Domain) ซึ่งออนโทโลยีนั้นถูกอ้างถึงว่าเป็นการจัดความรู้แนวความคิด หรือคำต่าง ๆ ให้เป็นระเบียบ (Formalization) อยู่ในขอบเขตโดเมนที่สนใจฐานความรู้ที่มีจึงสามารถนำมาแลกเปลี่ยน หรือนำมาใช้ซ้ำ (Reuse) และเผยแพร่ให้แก่ผู้อื่นได้นำไปใช้งานต่อ ๆ ไปอย่างกว้างขวางได้โดยออนโทโลยีนั้นประกอบด้วยออนโทโลยีคอนเซพต์ ที่แสดงกลุ่มคำ และความสัมพันธ์เชิงความหมายระหว่างคำในโดเมน ความสัมพันธ์ของคำในโดเมนที่มีหลากหลายรูปแบบ เช่น **is-a** , **part-of** ซึ่งความสัมพันธ์เหล่านี้จะเชื่อมโยงกลุ่มคำในโดเมนเป็นลำดับชั้น รูปแบบโครงสร้างออนโทโลยีจึงมีลักษณะเป็นลำดับชั้น เช่น โครงสร้างต้นไม้ (Tree), กราฟ (Graph) ตัวอย่างการจัดหมวดหมู่ความรู้ในโดเมนที่สนใจเป็นลำดับชั้น เช่นการจำแนกพืช ชนิดต่าง ๆ ออกเป็น **Genus** และ **Species** เป็นต้น แสดงตัวอย่างออนโทโลยีกุ่มสินค้าได้ดังภาพที่ 1 ซึ่งอธิบายการจัดกลุ่มสินค้าที่ประกอบไปด้วยความสัมพันธ์เชิงความหมาย คือ **is-a** เช่น **PC is-a Computer**, **part-of** เช่น **Keyboard part-of PC** และ **syn-of** เช่น **Data Processor syn-of Computer** เป็นต้น



ภาพที่ 1 ออนโทโลยีกุ่มสินค้า

ในส่วนต่อไปจะอธิบายถึงความรู้พื้นฐานที่เกี่ยวข้องกับออนโทโลยี ได้แก่ องค์ประกอบของออนโทโลยี และภาษาที่ใช้ในการอธิบายออนโทโลยี รวมถึงการนำออนโทโลยีไปประยุกต์ใช้

1. องค์ประกอบของออนโทโลยี (Components of ontology)

ออนโทโลยีประกอบไปด้วยคอนเซพ ซึ่งเป็นพื้นฐานสำคัญในการสร้างฐานความรู้ โดยคอนเซพเหล่านี้จัดเรียงอยู่ในลำดับชั้นการถ่ายทอดความสัมพันธ์ และมีคุณสมบัติเฉพาะในแต่ละคอนเซพ โดยสรุปแล้วองค์ประกอบของออนโทโลยีประกอบด้วยส่วนต่าง ๆ ดังต่อไปนี้

1.1 แนวคิด/คอนเซพ (Concept) ความคิดทั่วไปหรือนามธรรมในโดเมนที่เราสนใจ เช่น **Computer, PC, Notebook, Keyboard** เป็นต้น

1.2 คุณลักษณะ (Property) คุณสมบัติของคอนเซพ เช่น สี น้ำหนัก เป็นต้น

1.3 ความสัมพันธ์ (Relationship) ความสัมพันธ์เชิงความหมายระหว่างคอนเซพออนโทโลยีโดยส่วนใหญ่จะประกอบไปด้วยความสัมพันธ์ ได้แก่

1.3.1 is-a คือ ความสัมพันธ์ที่มีคุณสมบัติการถ่ายทอด คุณสมบัติของคอนเซพแม่ไปยังคอนเซพลูก เช่น **PC is-a Computer** ซึ่งอธิบายได้ว่า **PC** มีคุณสมบัติเป็น **Computer**

1.3.2 part-of คือ ความสัมพันธ์ที่หมายถึงการเป็นส่วนประกอบ เช่น **Keyboard part-of Computer** ซึ่งอธิบายได้ว่า **Computer** จะต้องประกอบไปด้วย **Keyboard**

1.3.3 syn-of คือ ความสัมพันธ์ที่แสดงถึงคอนเซพที่มีความเหมือนเชิงความหมายต่อกัน เช่น **Data Processor syn-of Computer** ซึ่งอธิบายได้ว่า **Computer** มีความหมายเดียวกันกับ **Data Processor** สามารถใช้แทนกันได้

1.3.4 instance-of คือ ความสัมพันธ์ที่แสดงถึงการเป็นตัวแทน หรือสมาชิกของคอนเซพ เช่น **HP instance-of Computer** ซึ่งอธิบายได้ว่า **HP** เป็นคอมพิวเตอร์ประเภทหนึ่ง

นอกจากนี้ออนโทโลยียังประกอบไปด้วยความสัมพันธ์เชิงความหมายอื่น ๆ ที่สอดคล้องกับโดเมนซึ่งกำหนดโดยผู้เชี่ยวชาญ

1.4 ข้อความอธิบายคอนเซพ (Axiom) เงื่อนไขหรือข้อกำหนดเฉพาะในแต่ละคอนเซพเป็นกลไกสำคัญสำหรับการอนุมานความรู้ เพื่อการสร้างความรู้ใหม่จากอนโทโลยี ตัวอย่างเช่น คอมพิวเตอร์จะต้องประกอบด้วยชิพียู เป็นต้น

2 ภาษาที่ใช้อธิบายอนโทโลยี (Ontology languages)

ภาษาที่ใช้พัฒนาอนโทโลยี (Roche, 2003) แบ่งตามรูปแบบภาษาได้ดังต่อไปนี้ ลอจิกเบส (First order logic based), เฟรมเบส (Frame logic based) และเว็บเบส (Web based)

2.1 ลอจิกเบส รูปแบบภาษาที่อธิบายอนโทโลยี โดยการใช้ตรรกะเพื่อการอนุมานความรู้ (อัศนีชัย, 2549) โดยมีส่วนประกอบ คือ ภาคแสดง (Predicate) อาร์กิวเมนต์ (Argument) และตัวบ่งปริมาณ (Quantifier) ตัวอย่าง เช่น $\forall x: PC(X) \rightarrow haspart(keyboard(x))$ หมายถึง PC ทุกเครื่องจะประกอบด้วย keyboard

คุณสมบัติของภาษาอนโทโลยีที่พัฒนาแบบลอจิกเบส คือ วากยสัมพันธ์ (Syntax) มีความชัดเจน และมีรูปแบบเป็นทางการ จึงสามารถสร้างกฎอนุมานได้ (Reference rule) ตัวอย่างภาษา เช่น CYCL, CLASSIC, LOOM เป็นต้น

2.2 เฟรมเบส รูปแบบภาษาที่อธิบายอนโทโลยี โดยใช้หลักการของการคิดเชิงปฏิสัมพันธ์ของมนุษย์ (อัศนีชัย, 2549) เมื่อนึกถึงวัตถุ หรือสิ่งใดจะเชื่อมโยงกับคุณลักษณะเด่นของวัตถุหรือสิ่งนั้นด้วย ดังนั้น เฟรมจึงประกอบไปด้วย เซตของคุณสมบัติ (Attribute) หรือ สล็อต(slot) และแฟกซ์เซต (Facets) ข้อความอธิบายคอนเซพหรือคำอธิบายสล็อต คุณสมบัติของภาษาอนโทโลยีที่พัฒนาแบบเฟรมเบส คือ เข้าใจได้ง่าย ตัวอย่างภาษาที่พัฒนาแบบเฟรมเบส เช่น OKBC, F-logic แสดงตัวอย่างการใช้เฟรมแทนอนโทโลยีคอนเซพ “PC” ได้ดังนี้

(PC	(HP
<Model modelname>	<Model L1760>
<Color color>	<Color Bronze>
< Price price>	< Price 120 USD >
))

จากตัวอย่างเฟรมข้างต้นเป็นเฟรมที่อธิบายองค์ประกอบการแทนคอนเซพ “PC” ซึ่งประกอบด้วย ค่าคุณสมบัติ 3 ค่า ได้แก่ ชื่อรุ่น (Model), สี (Color) และราคา (Price) และการแทนค่าคุณสมบัติของตัวอย่างสมาชิกของคอนเซพ (Instance) คือ “HP”

2.3 เว็บเบส รูปแบบภาษาที่อธิบายออนโทโลยีซึ่งพัฒนาจากภาษาที่ใช้สำหรับอธิบายทรัพยากรบนเว็บ (Roche, 2003) ได้แก่ XML (Extensible Markup Language), RDF (Resource Description Framework) ซึ่งอธิบายออนโทโลยีโดยใช้พื้นฐานของลอจิกเบส และเฟรมเบส ภาษาที่พัฒนาขึ้นจึงมีหลักการและแบบแผน สามารถอ้างอิงได้ และอยู่ในรูปแบบที่มนุษย์สามารถเข้าใจได้ง่าย ตัวอย่างภาษาเช่น DAML + OIL (DARPA Agent Markup Language + Ontology Interface Language), OWL (Web Ontology Language) เป็นต้น

3 การประยุกต์ใช้ออนโทโลยี (Ontology application)

ออนโทโลยีถูกนำไปประยุกต์ใช้ในหลาย ๆ งาน สามารถแบ่งกลุ่มโปรแกรมที่ประยุกต์ใช้ออนโทโลยี (Jasper and Uschold, 1999) ได้ดังนี้

3.1 การนำออนโทโลยีไปใช้เพื่อแปลงข้อมูลให้อยู่ในรูปแบบภาษาต่าง ๆ (Neutral authoring) นำออนโทโลยีไปใช้เพื่อแปลงข้อมูลให้อยู่ในรูปแบบภาษาต่าง ๆ เพื่อให้โปรแกรมอื่น ๆ สามารถใช้งานได้ประโยชน์ที่ได้จากการประยุกต์ใช้ออนโทโลยี คือ การนำความรู้มาใช้ได้อีก (Knowledge reuse)

32 การนำออนโทโลยีมาใช้ในการกำหนดรายละเอียดของซอฟต์แวร์ (Ontology as specification) ประยุกต์ใช้ออนโทโลยีเพื่อออกแบบซอฟต์แวร์ในโดเมน และรวบรวมคำศัพท์สำหรับกำหนดความต้องการในการพัฒนาซอฟต์แวร์ ประโยชน์ที่ได้จากการประยุกต์ใช้ออนโทโลยี คือ การทำคู่มือโปรแกรม การบำรุงรักษาซอฟต์แวร์ และการนำกลับมาใช้ใหม่

33 การประยุกต์ใช้ออนโทโลยีเพื่อการเข้าถึงข้อมูลที่มีโครงสร้าง หรือรูปแบบต่างกัน (Common Access to Information) ออนโทโลยีจัดเตรียมคำที่สามารถเข้าใจได้ตรงกัน หรือจัดกลุ่มคำที่มีความหมายเดียวกัน ประโยชน์ที่ได้ คือ การทำงานร่วมกัน (Inter-operability) และการนำกลับมาใช้ใหม่

34 การประยุกต์ใช้ออนโทโลยีเพื่อการสืบค้นข้อมูลจากแหล่งข้อมูลต่าง ๆ (Ontology-based search) ประยุกต์ใช้ออนโทโลยีเพื่อการสืบค้นข้อมูลจากแหล่งข้อมูลต่าง ๆ เช่น เอกสาร เว็บไซต์ หรือฐานข้อมูล แนวทางนี้ประยุกต์ใช้ออนโทโลยีในการกำหนดคอนเซพที่สอดคล้องกับคำสืบค้นของผู้ใช้ และใช้คอนเซพนั้นในการสืบค้นข้อมูล ทำให้ผลการสืบค้นมีความถูกต้องมากยิ่งขึ้น และเวลาที่ใช้ในการสืบค้นลดลง ซึ่งจะอธิบายรายละเอียดเพิ่มเติมในหัวข้อถัดไป

การสืบค้นข้อมูลโดยประยุกต์ใช้ออนโทโลยี

ในปัจจุบันออนโทโลยีมีความสำคัญต่อการพัฒนาแอปพลิเคชันในด้านต่างๆ ให้มีประสิทธิภาพมากขึ้น โดยเฉพาะการพัฒนาแอปพลิเคชันสำหรับสืบค้นข้อมูลที่ประยุกต์ใช้ออนโทโลยีเพื่อสืบค้นข้อมูลจากแหล่งข้อมูลต่าง ๆ เช่น เอกสาร เว็บไซต์ และการสืบค้นข้อมูลในโดเมนที่เกี่ยวข้องกับวิทยาศาสตร์เพื่อชีวิต เช่น **Gene Ontology Project [GO] (1999), Plant Ontology Consortium[POC] (2004)**

การประยุกต์ใช้ออนโทโลยีในการสืบค้นข้อมูล (Jasper and Uschold ,1999) สามารถอธิบายได้ดังนี้ การประยุกต์ใช้ออนโทโลยีเป็นโครงสร้างพื้นฐานในการจัดการข้อมูล เช่น ประยุกต์ใช้ออนโทโลยีเป็นโครงสร้างของคอนเซพ เพื่อให้ผู้ใช้ทราบถึงขอบเขตของแหล่งข้อมูล ตัวอย่างการใช้เช่น **Yahoo taxonomy** ซึ่งจะแบ่งกลุ่มข้อมูลเป็น 14 กลุ่มใหญ่ที่ประกอบด้วยกลุ่มย่อยที่สอดคล้องกัน, การประยุกต์ใช้ออนโทโลยีเป็นคำศัพท์สำหรับกำหนดเมตาเดต้า (Metadata

language) สำหรับสร้างดัชนี (Index) หรือสร้างแท็ก (Tag) ให้กับข้อมูล อีกแนวทางหนึ่ง คือ การประยุกต์ใช้ออนโทโลยี เพื่อแปลงคำสั่งสืบค้น (Query transformation) เช่น การประยุกต์ใช้ออนโทโลยีในการพัฒนาส่วนการติดต่อกับผู้ใช้ในการสร้าง และแก้ไขคำสั่งสืบค้น, การประยุกต์ใช้ออนโทโลยีเพื่ออ้างอิงถึงคอนเซพที่สอดคล้องกับคำสั่งสืบค้น

ตัวอย่างระบบการสืบค้นข้อมูลโดยใช้ออนโทโลยี ได้แก่ Knowledge-Based Discovery Tool (Eilerts *et al.*, 1999) เมื่อผู้ใช้งานกำหนดคำสั่งสืบค้นที่ต้องการ ระบบจะพิจารณาคำสืบค้นกับออนโทโลยี WordNet เพื่อพิจารณาหาคอนเซพที่สอดคล้องกับคำสั่งสืบค้น และให้ผู้ใช้งานกำหนดคอนเซพที่ถูกต้อง หลังจากนั้นคอนเซพที่มีความหมายไม่ตรงกับคำสั่งสืบค้นก็จะถูกคัดออกไป เพื่อลดความกำกวมของผลการสืบค้น

ในหัวข้อถัดไปจะอธิบายถึงการสืบค้นข้อมูลเชิงสัมพันธ์โดยประยุกต์ใช้ออนโทโลยี ซึ่งเป็นแนวทางที่ประยุกต์ใช้ออนโทโลยี เพื่อเพิ่มประสิทธิภาพในการสืบค้นข้อมูลในฐานข้อมูลเชิงสัมพันธ์ ซึ่งสืบค้นโดยพิจารณาคำสืบค้นโดยตรงให้สามารถสืบค้นโดยพิจารณาคำสืบค้นเชิงความหมายได้ (Semantic search)

การสืบค้นข้อมูลในฐานข้อมูลเชิงสัมพันธ์โดยประยุกต์ใช้ออนโทโลยี

การสืบค้นข้อมูลในฐานข้อมูลเชิงสัมพันธ์โดยประยุกต์ใช้ออนโทโลยี เป็นแนวทางเพิ่มประสิทธิภาพการสืบค้นข้อมูลในฐานข้อมูลเชิงสัมพันธ์ให้สามารถสืบค้นข้อมูลเชิงความหมายและข้อมูลที่มีลักษณะเป็นลำดับชั้นได้ (Concept Hierarchy) เช่น การสืบค้นข้อมูลฐานสมุนไพรระดับตระกูล (Order) และวงศ์ (Family) โดยประยุกต์ใช้ออนโทโลยีเพื่อแปลงคำสั่งสืบค้นของผู้ใช้ให้อยู่ในรูปแบบคำสั่งสืบค้นเชิงความหมาย เช่น การพิจารณาคอนเซพในออนโทโลยีที่เฉพาะเจาะจง (Specialization) คือ การหาโหนดที่เป็นโหนดลูก (Child node) ของโหนดที่เราสนใจ, การพิจารณาคอนเซพในออนโทโลยีในระดับทั่วไป (Generalization) คือ การหาโหนดบรรพบุรุษ (Parent Node) ของโหนดที่เราพิจารณา และการพิจารณาคอนเซพข้างเคียงในออนโทโลยี (Neighborhood) ทำให้ผลการสืบค้นที่ได้มีความครบถ้วนมากยิ่งขึ้น เมื่อเทียบกับการสืบค้นข้อมูลในฐานข้อมูลเชิงสัมพันธ์ทั่ว ๆ ไปที่สืบค้นโดยใช้คำสั่งสืบค้นจากผู้ใช้งานเพียงเท่านั้นอีกทั้งยังลดเวลาที่ใช้ในการสืบค้นอีกด้วย

พิจารณาตัวอย่างการสืบค้นข้อมูลจากตาราง Item ถ้าผู้ใช้ต้องการสืบค้นคำว่า “Computer”

Select * from Item where Name= 'Computer'

ผลการสืบค้นที่ได้จะมีเพียงเรคคอร์ดเดียวเท่านั้น คือ RecID ที่ 123 แต่เมื่อพิจารณาออนโทโลยีกุ่มสินค้าดังภาพที่ 1 จะเห็นได้ว่า Computer มีความหมายเหมือนกันกับ “Calculator” และ “Data Processor” ซึ่งอธิบายด้วย ความสัมพันธ์แบบ syn-of ในออนโทโลยี นอกจากนี้ เมื่อพิจารณาความสัมพันธ์แบบ is-a จะพบว่า “PC” และ “Notebook” สอดคล้องกับคำสืบค้น ดังนั้นผลการสืบค้น คือ RecID ที่ 123, 125, 127, 141

ตารางที่ 1 ตาราง Item

RecID	Name	Model	Price
123	Computer	IBM	3000\$
124	IntelPC	TOSHIBA	5000\$
125	Notebook	DELL	4000\$
127	PC	COMPAQ	2500\$
128	Product	HP	3000\$
129	Monitor	ELSA	1000\$
135	Keyboard	IIT	80\$
136	Desktop	IBM	1000\$
140	MacPC	MAC	2000\$
141	Calculator	SIEMEN	1500\$

การพัฒนาออนโทโลยีเพื่อเพิ่มประสิทธิภาพการสืบค้นข้อมูลในฐานข้อมูลเชิงสัมพันธ์ ต้องพิจารณาถึง แนวทางการจัดเก็บและสืบค้นออนโทโลยี รวมถึง วิธีการสืบค้นข้อมูล ในฐานข้อมูลเชิงสัมพันธ์ที่สอดคล้องกับออนโทโลยี อธิบายตัวอย่างงานที่เกี่ยวข้อง โดย แบ่งตามแนวทางการจัดเก็บและสืบค้นออนโทโลยี ได้ดังต่อไปนี้

1. แนวทางการสืบค้นออนโทโลยีตามการประมวลผลค่าทรานซิทีฟโคลสเชอร์ (Precomputation of transitive closure)

แนวทางนี้จัดเก็บออนโทโลยีโดยประมวลผลหาทรานซิทีฟโคลสเชอร์ของทุกโหนดในออนโทโลยี และจัดเก็บในฐานข้อมูลโดยอาศัยความสัมพันธ์แบบถ่ายทอด เช่น กำหนดโหนด **u**, **v**, **w** เป็นโหนดใด ๆ ในออนโทโลยี เมื่อโหนด **u** เป็นโหนดลูกของโหนด **v** และ โหนด **v** เป็นโหนดลูกของโหนด **w** ดังนั้นโหนด **u** จะเป็นโหนดลูกของโหนด **w** ด้วย ซึ่งการประมวลผลหาทรานซิทีฟโคลสเชอร์จะทำให้ทราบถึงความสัมพันธ์ระหว่างโหนด กับโหนดลูกทั้งหมดของโหนดนั้น การสืบค้นออนโทโลยีจึงทำได้ง่ายโดยไม่ต้องวนซ้ำตามโครงสร้างของออนโทโลยี

ตัวอย่างงานที่เกี่ยวข้องได้แก่ **GO (1999)** จัดเก็บออนโทโลยีในฐานข้อมูลเชิงสัมพันธ์ โดยเก็บข้อมูลรูปแบบกราฟมีทิศทาง และสร้างตารางทรานซิทีฟโคลสเชอร์ (Transitive closure) จึงสามารถสืบค้นออนโทโลยีโดยใช้คำสั่ง **SQL** อย่างง่าย รายละเอียดการจัดเก็บและสืบค้นออนโทโลยีของงานวิจัย อธิบายได้ดังนี้

1.1 การจัดเก็บออนโทโลยี งานวิจัยนี้จัดเก็บออนโทโลยีหลายรูปแบบ เช่น **XML**, **RDF** ไฟล์ รวมถึงจัดเก็บในฐานข้อมูลเชิงสัมพันธ์ **MySQL** โดยจัดเก็บออนโทโลยีในตารางรูปแบบกราฟแบบมีทิศทาง **DAG (Directed Acyclic Graph)** โหนดแทนออนโทโลยีคอนเซ็ปต์เก็บในตาราง **Term** และเส้นทางระหว่างโหนด (**Edge**) เก็บในตาราง **Term2Term** แทนความสัมพันธ์ ในออนโทโลยีแต่ละโหนดสามารถมีโหนดพ่อได้มากกว่า 1 โหนด และมีเส้นทางจากรูทโหนด (**Root node**) ไปถึงทุก ๆ โหนด

1.2 การสืบค้นออนโทโลยี การสืบค้นออนโทโลยีทำได้โดยสืบค้นแบบวนซ้ำในตาราง **Term2Term** ซึ่งจะต้องเรียกใช้คำสั่ง **SQL** หลายครั้งและไม่มีคำสั่ง **SQL** รองรับการสืบค้นประเภทนี้ อีกแนวทาง คือ การสร้างตารางทรานซิทีฟโคลสเชอร์โดยพิจารณาความสัมพันธ์ระหว่างโหนดไปยังทุก ๆ โหนดบรรพบุรุษ รวมทั้งพิจารณาคุณสมบัติสะท้อน (Reflective) นั่นคือ ทุกโหนดจะมีความสัมพันธ์กับตัวเอง เก็บในตารางชื่อ **Graph_path** ซึ่งเก็บชื่อโหนด โหนดบรรพบุรุษ และความสัมพันธ์ รวมทั้งระยะห่างระหว่างโหนด

1.3การพัฒนาส่วนสืบค้นเชิงความหมายในฐานข้อมูลเชิงสัมพันธ์ การสืบค้นทำได้โดยใช้คำสั่ง **SQL** ซึ่งผู้ใช้ต้องทราบถึงโครงสร้างตารางจัดเก็บ ออนโทโลยี หรือ สืบค้นผ่าน **Perl API** ซึ่งจะต้องติดตั้ง **Perl shell** จึงจะสามารถสืบค้น ออนโทโลยีได้ แสดงตัวอย่างคำสั่งสืบค้นดังต่อไปนี้

ตัวอย่างรูปประโยคที่ 1 การหาโหนดลูกทั้งหมดของโหนดที่ต้องการสืบค้น

```
SELECT rchild.*
FROM term as rchild, term as ancestor, grap_path
WHERE grap_path.term2_ID=rchild.ID AND
      grap_path.term1_ID= ancestor.ID AND
      ancestor.name='x'
```

ตัวอย่างรูปประโยคที่ 2 การหาโหนดบรรพบุรุษทั้งหมดของโหนดที่ต้องการสืบค้น

```
SELECT ancestor.*
FROM term as rchild, term as ancestor, grap_path
WHERE grap_path.term2_ID=rchild.ID AND
      grap_path.term1_ID= ancestor.ID AND
      rchild.name='x'
```

2 แนวทางการสืบค้นออนโทโลยีตามการคำนวณค่าการเรียงลำดับ (Precomputation of Pre-Post order ranking)

แนวทางนี้จัดเก็บออนโทโลยีโดยการประยุกต์ใช้แนวทาง **Pre-Post ranking** ที่พิจารณาค่าการเรียงลำดับการขึ้นต่อกันตามความสัมพันธ์แบบพอลูก ซึ่งทำได้โดยการท่องกราฟ (Traversal) แบบเชิงลึกไปตามความสัมพันธ์ และกำหนดค่า **pre-order** และ **post-order** โดยค่า **pre-order** จะถูกกำหนดเมื่อโหนดนั้นถูกเยือนผ่านมาแล้ว ส่วนค่า **post-order** จะถูกกำหนดเมื่อโหนดลูกทั้งหมดของโหนดนั้นถูกเยือนหมดแล้ว

ตัวอย่างงานที่เกี่ยวข้องได้แก่ **Tribl** และ **Leser (2005)** เสนอแนวทางจัดเก็บและสืบค้นออนโทโลยีขนาดใหญ่โดยการกำหนดค่า **Pre-Post order** จากผลการทดลองงานวิจัยนี้ได้ผลว่าแนวทางที่เสนอใช้เวลาในการประมวลผลน้อยกว่าวิธีการสืบค้นออนโทโลยีโดยการวนลูปซ้ำ และ

ใช้เวลามากกว่าการจัดเก็บและสืบค้นออนไลน์โดยวิธีทรานซิติฟโกลสเซอร์แต่ใช้พื้นที่ในการจัดเก็บข้อมูลน้อยกว่า จากการศึกษาสามารถสรุปแนวทางการจัดเก็บ และการสืบค้นออนไลน์ของงานวิจัยนี้ได้ดังนี้

2.1 การจัดเก็บออนไลน์ จัดเก็บออนไลน์ในตารางรูปแบบกราฟแบบมีทิศทาง โหนดแทนออนไลน์คอนเซ็ปต์เก็บในตาราง **Node** แต่ละโหนดสามารถมีโหนดพ่อได้มากกว่า **1** โหนด และมีเส้นทางจากทุกโหนดไปถึงทุก ๆ โหนด ความสัมพันธ์ระหว่างโหนดเก็บในตาราง **Edge** และค่าการเรียงลำดับทั้ง **pre-order** และ **post-order** ของแต่ละโหนดจะจัดเก็บในตาราง **prePostOrder** คู่กับ **NodeID** และจำนวนโหนดลูกของโหนดนั้น ๆ แสดงขั้นตอนวิธีในการกำหนดค่าการเรียงลำดับ **pre-order, post-order** ดังนี้

อัลกอริทึม **1** การกำหนดค่า **Pre-Post order**

FUNCTION prePostOrder(r)

BEGIN

FOR EACH child, $m \in \sigma_{\text{from_node}=r}$ **EDGE DO**

pre=**pr**; **pr**=**pr**+1

prePostOrder(m)

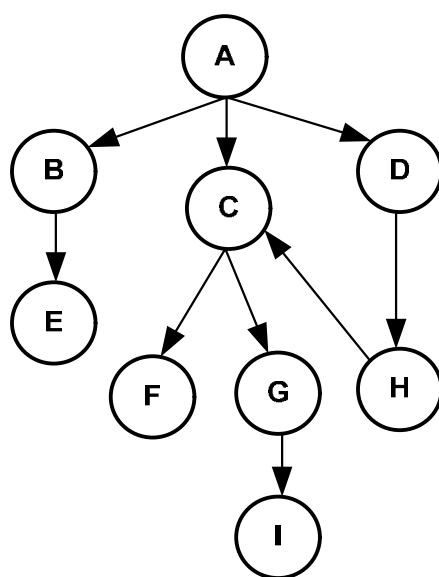
INSERT m, **pre**, **post**, **pr**-**pre** **INTO** **prePostOrder**

Post=**post**+1

END FOR

END

จากตัวอย่างออนไลน์ดังภาพที่ **2** เมื่อประมวลผลกำหนดค่าเรียงลำดับ **pre-order, post-order** ตามอัลกอริทึมที่ **1** สามารถแสดงตัวอย่างตารางจัดเก็บออนไลน์ได้ดังตารางที่ **2** ตาราง **PrePostOrder**



ภาพที่ 2 ตัวอย่างออนโทโลยี

ตารางที่ 2 ตาราง PrePostOrder

Node	pre	post	s
A	0	12	12
B	1	1	1
C	3	5	3
D	7	11	5
E	2	0	0
F	4	2	1
G	5	4	1
H	8	10	4
I	6	3	0
C	9	9	3
F	10	6	0
G	11	8	1
I	12	7	0

2.2 การสืบทอดออนไลน์ จากการจัดเก็บออนไลน์โดยพิจารณาความสัมพันธ์ระหว่าง โหนดและกำหนด ค่า **pre-order** และ **post-order** จะทำให้การสืบทอดออนไลน์สามารถทำได้โดยไม่ต้องทำการวนลูป เนื่องจาก โหนดลูกของโหนด v ใด ๆ จะมีค่า **pre-order** มากกว่า โหนด v และจะมีค่า **post-order** มากกว่า โหนด v เมื่อกำหนดค่า **pre-order** ของโหนด v ให้มีค่า Pre_v และกำหนดให้ s เป็นจำนวนโหนดลูกของโหนด v ให้ โหนด w แทนโหนดลูกทั้งหมดของโหนด v ที่มีค่า **pre-order** เท่ากับ Pre_w โดยค่าของ Pre_w ที่เป็นไปได้จะอยู่ในช่วง $Pre_v < Pre_w \leq Pre_v + s$ การสืบทอดออนไลน์ใช้คำสั่งดังต่อไปนี้

ตัวอย่างรูปประโยคที่ 3 การหาโหนดลูกทั้งหมดของโหนดที่ต้องการสืบค้น

```
SELECT DISTINCT p1.node_name As w
FROM prePostOrder p1
WHERE p1.pre > (
    SELECT p1.pre
    FROM prePostOrder p2
    WHERE p2.node_name = v LIMIT 1)
AND p1.pre ≤ (
    SELECT p2.pre + p2.s
    FROM prePostOrder p2
    WHERE p2.node_name = v LIMIT 1);
```

ตัวอย่างรูปประโยคที่ 4 การหาโหนดบรรพบุรุษทั้งหมดของโหนดที่ต้องการสืบค้น

```
SELECT DISTINCT p2.node_name As u
FROM prePostOrder p1
    prePostOrder p2
WHERE p1.node_name = v
AND p2.pre < p1.pre
    p2.post > p1.post
```


3 แนวทางการสืบค้นออนโทโลยีแบบวนซ้ำ (Recursive database function)

แนวทางนี้จัดเก็บออนโทโลยี เฉพาะโหนดกับโหนดพ่อ หรือโหนดลูกเท่านั้น (ระยะห่างระหว่างโหนดเท่ากับ 1) การจัดเก็บออนโทโลยีในรูปแบบนี้ทำให้ไม่สามารถค้นหาโหนดบรรพบุรุษ หรือโหนดลูกทั้งหมดของโหนดที่กำหนดได้โดยการอ่านข้อมูลจากตารางเพียงครั้งเดียว การค้นหาโหนดดังกล่าว จะต้องวนซ้ำอ่านข้อมูลในตารางเรื่อย ๆ จนกว่าจะได้คำตอบทั้งหมด ซึ่งการสืบค้นแบบวนซ้ำทำได้โดย การพัฒนาฟังก์ชัน หรือการใช้คำสั่งสืบค้นข้อมูลลำดับชั้นที่ระบบฐานข้อมูลเชิงสัมพันธ์พัฒนาขึ้น

ตัวอย่างงานที่เกี่ยวข้องได้แก่ **Das et al (2004a, 2004b)** เสนอโอเปอเรเตอร์สืบค้นเชิงความหมายในฐานข้อมูลเชิงสัมพันธ์โดยการประยุกต์ใช้ออนโทโลยี โดยจัดเก็บออนโทโลยีในฐานข้อมูลเชิงสัมพันธ์ **Oracle** และสืบค้นโดยประยุกต์ใช้คำสั่งสำหรับสืบค้นข้อมูลแบบมีลำดับชั้น “START WITH” และ “CONNECT BY” (**Oracle9i database online documentation, 2003**) รายละเอียดการจัดเก็บและสืบค้นออนโทโลยีในงานวิจัย อธิบายได้ดังนี้

31 การจัดเก็บออนโทโลยี งานวิจัยนี้แปลงออนโทโลยีไฟล์ **OWL** มาจัดเก็บในฐานข้อมูลเชิงสัมพันธ์ **Oracle** ซึ่งจัดเก็บความสัมพันธ์ระหว่างโหนดในออนโทโลยีโดยพิจารณาความสัมพันธ์แต่ละโหนดกับโหนดบรรพบุรุษที่ใกล้ที่สุด จัดเก็บในตาราง **Relationships** ที่แสดงความสัมพันธ์ระหว่างโหนดได้ดังนี้

Relationships (OntologyName, Tem1, Relation, Tem2, ...)
--

32 การสืบค้นออนโทโลยี เนื่องจากงานวิจัยนี้จัดเก็บออนโทโลยีในฐานข้อมูลเชิงสัมพันธ์ **Oracle** จึงสามารถเรียกใช้คำสั่ง “START WITH” และ “CONNECT BY” ที่จะสืบค้นตามโครงสร้างลำดับชั้นของออนโทโลยีตามความสัมพันธ์ที่กำหนด โดยประกอบด้วยรูปประโยคดังนี้

ตัวอย่างรูปประโยคที่ 5 รูปประโยคคำสั่ง “START WITH” และ “CONNECT BY”

```
SELECT    column1, column2, ...
FROM      table
[WHERE    condition]
START WITH condition
CONNECT BY [PRIOR] column1=[PRIOR] column2
[ORDER BY LEVEL]
```

START WITH	กำหนดโหนดเริ่มต้นของการค้นหาข้อมูล หากไม่กำหนดจะใช้ทุกโหนดในตารางเป็นโหนดเริ่มต้น
CONNECT BY	กำหนดความสัมพันธ์ระหว่างข้อมูลแถวที่เป็นพ่อหรือแม่ กับแถวที่เป็นลูกของโครงสร้างลำดับชั้นที่จะค้นหา โดยประกอบด้วยอนุประโยค PRIOR สำหรับกำหนดเงื่อนไขความสัมพันธ์ว่าจะสืบค้นหาโหนดที่เป็นลูกของโหนดที่กำหนดหรือค้นหาโหนดที่เป็นพ่อแม่ ของโหนดที่กำหนด
WHERE	กำหนดเงื่อนไขข้อมูลที่ต้องการค้นหา

การทำงานของประโยค **SQL** ที่กำหนดข้างต้นจะใช้อนุประโยคย่อยทั้ง 3 สร้างโครงสร้างแบบมีลำดับชั้นข้อมูลขึ้นมา การสืบค้นข้อมูลจะใช้วิธีการค้นหาแบบลึกโดยมีวิธีการทำงานดังนี้

ขั้นตอนที่ 1. ทำการค้นหาแถวเริ่มต้นของโครงสร้างแบบมีลำดับชั้น ตามการกำหนดเงื่อนไขในอนุประโยค **START WITH**

ขั้นตอนที่ 2. ทำการค้นหาแถวลูกที่มีคุณสมบัติสอดคล้องกับเงื่อนไขในอนุประโยค **CONNECT BY** กับของแถวเริ่มต้นที่ได้จากข้อ 1

ขั้นตอนที่ 3. ทำการค้นหาแถวที่เป็นลูกของแถวที่ได้จากขั้นตอนที่ 2 จากนั้นวนหาแถวที่เป็นลูกของแถวที่สืบค้นมาเรื่อย ๆ จนกระทั่งได้โครงสร้างแบบมีลำดับชั้นขึ้นมา โดยการเลือกแถวที่เป็นโหนดลูกนั้นต้องมีคุณสมบัติสอดคล้องกับเงื่อนไขในอนุประโยค **CONNECT BY** ทุกครั้ง

3.3การพัฒนาส่วนสืบค้นเชิงความหมายในฐานข้อมูลเชิงสัมพันธ์ การสืบค้นทำได้โดยใช้คำสั่ง **SQL** โดยพัฒนาโอเปอเรเตอร์สืบค้นเชิงความหมายชื่อ **ONT_RELATED** ผู้ใช้จึงไม่จำเป็นต้องทราบถึงโครงสร้างตารางสำหรับจัดเก็บออนโทโลยี โอเปอเรเตอร์นี้พัฒนาโดยใช้คำสั่ง **“START WITH”** และ **“CONNECT BY”** ที่จะสืบค้นตามโครงสร้างลำดับชั้นของออนโทโลยีตามความสัมพันธ์ที่กำหนด โดยใช้หลักการของการค้นหาแบบอ้างอิงตามความลึก วิธีการนี้จึงไม่มีประสิทธิภาพเท่าที่ควรเมื่อโครงสร้างลำดับชั้นของออนโทโลยีมีหลายระดับชั้น แสดงรูปแบบของโอเปอเรเตอร์ ได้ดังนี้

ONT_RELATED (Tem1, RelType,Tem2, OntologyName)
Return Integer

โอเปอเรเตอร์นี้จะพิจารณาว่า **Tem1** และ **Tem2** คำมีความสัมพันธ์ตาม **RelType** ที่ระบุหรือไม่ ถ้ามีความสัมพันธ์กันตามที่ระบุจะคืนค่าเป็น **1** ถ้าไม่มีความสัมพันธ์กันจะคืนค่า **0** โดยการทำงานของโอเปอเรเตอร์ทำได้โดยการแปลงคำสั่งสืบค้นให้อยู่ในรูปแบบคำสั่ง **“START WITH”** และ **“CONNECT BY”** แสดงตัวอย่างดังนี้

ตัวอย่างรูปประโยคที่ 6 การเรียกใช้โอเปอเรเตอร์ **ONT_RELATED**

SELECT * FROM Item

WHERE ONT_RELATED (Itemname,'is-a','Computer',ProductOntology)=1

ตัวอย่างรูปประโยคที่ 7 รูปประโยคคำสั่งสืบค้นด้วยอนุประโยค **“START WITH”** และ **“CONNECT BY”**

SELECT * FROM Item

WHERE Itemname in

(SELECT tem1 from Relationships

START WITH tem2='Computer'

AND Relation='is-a'

CONNECT BY PRIOR tem1=tem2 AND Relation='is-a');

**การเปรียบเทียบงานวิจัยที่เกี่ยวข้องกับการสืบค้นข้อมูลในฐานข้อมูลเชิงสัมพันธ์
โดยประยุกต์ใช้ออนโทโลยี**

จากงานวิจัยที่เกี่ยวข้องกับการสืบค้นข้อมูลในฐานข้อมูลเชิงสัมพันธ์โดยประยุกต์ใช้ออนโทโลยีสามารถแบ่งได้ 3 แนวทางได้แก่ **Æ** แนวทางการสืบค้นออนโทโลยีตามการประมวลผลค่าทรานซิทีฟโคลสเชอร์, • แนวทางการสืบค้นออนโทโลยีตามการคำนวณค่าการเรียงลำดับ, **Ž** แนวทางการสืบค้นออนโทโลยีแบบวนซ้ำ สามารถสรุปตารางเปรียบเทียบได้ดังตารางที่ 3

ตารางที่ 3 ตารางเปรียบเทียบงานวิจัยที่เกี่ยวข้อง

	Æ	•	Ž
1. โครงสร้างออนโทโลยี (Ontology structure)	DAG	DAG	DAG
2. วิธีการเตรียมข้อมูล (Data preparation)	Transitive closure	Pre-Post order ranking	-
3. การสืบค้นออนโทโลยี (Ontology query)	Simple SQL	Simple SQL	StartWith-Connect By clause
4. ขนาดพื้นที่ (Storage space)	3	2	1
5. เวลาที่ใช้ประมวลผล (Processing time)	1	2	3

จากตารางเปรียบเทียบเมื่อ ค่า 1-3 ระบุถึงค่าจาก น้อยไปมาก สามารถสรุปตารางได้ดังนี้

พิจารณาโครงสร้างออนโทโลยีทุกงานวิจัยทุกแนวทางจัดเก็บออนโทโลยีโดยใช้โครงสร้างแบบเดียวกัน คือกราฟแบบมีทิศทาง

เมื่อพิจารณาการเตรียมข้อมูล งานวิจัยในแนวทางที่ 3 ไม่มีการจัดเตรียมข้อมูล สามารถสืบค้นข้อมูลได้โดยใช้คำสั่ง “START WITH” และ “CONNECT BY” งานวิจัยนี้จึงใช้เวลาในการประมวลผลมากที่สุด เนื่องจากต้องใช้เวลาในการทำซ้ำเพื่อสืบค้นข้อมูล ส่วนงานวิจัยในแนวทาง

ที่ 1 จัดเตรียมข้อมูลสำหรับเก็บความสัมพันธ์ระหว่างโหนดทั้งหมดในออนโทโลยี จึงสืบค้นออนโทโลยีได้โดยใช้คำสั่ง **SQL** อย่างง่าย และใช้เวลาในการประมวลผลน้อยที่สุด และงานวิจัยในแนวทางที่ 2 จัดเตรียมข้อมูลสำหรับเก็บค่า **Pre-Post order** จึงสืบค้นออนโทโลยีได้โดยใช้คำสั่ง **SQL** อย่างง่าย และใช้เวลาในการประมวลผลเป็นลำดับที่ 2 เนื่องจากคำสั่งสืบค้นต้องเปรียบเทียบค่า **Pre-Post order**

เมื่อพิจารณาขนาดพื้นที่สำหรับจัดเก็บออนโทโลยี งานวิจัยในแนวทางที่ 3 ใช้พื้นที่น้อยที่สุด เนื่องจากไม่มีการจัดเตรียมข้อมูล งานวิจัยในแนวทางที่ 2 ใช้พื้นที่ในการจัดเก็บออนโทโลยีเป็นลำดับที่ 2 และงานวิจัยในแนวทางที่ 3 ใช้พื้นที่ในการจัดเก็บออนโทโลยีมากที่สุด เนื่องจากจะต้องจัดเก็บความสัมพันธ์ระหว่างโหนดทั้งหมดในออนโทโลยี

อุปกรณ์และวิธีการ

อุปกรณ์

1. ระบบฐานข้อมูลเชิงสัมพันธ์ Oracle เวอร์ชัน 9.2.0.1.0
2. โปรแกรมพัฒนาออนโทโลยี Protégé เวอร์ชัน 3.1.1
3. โปรแกรม Visual Studio.NET version 2003
4. ภาษา Visual Basic
5. ตัวอย่างออนโทโลยี จาก Gene Ontology Project

วิธีการ

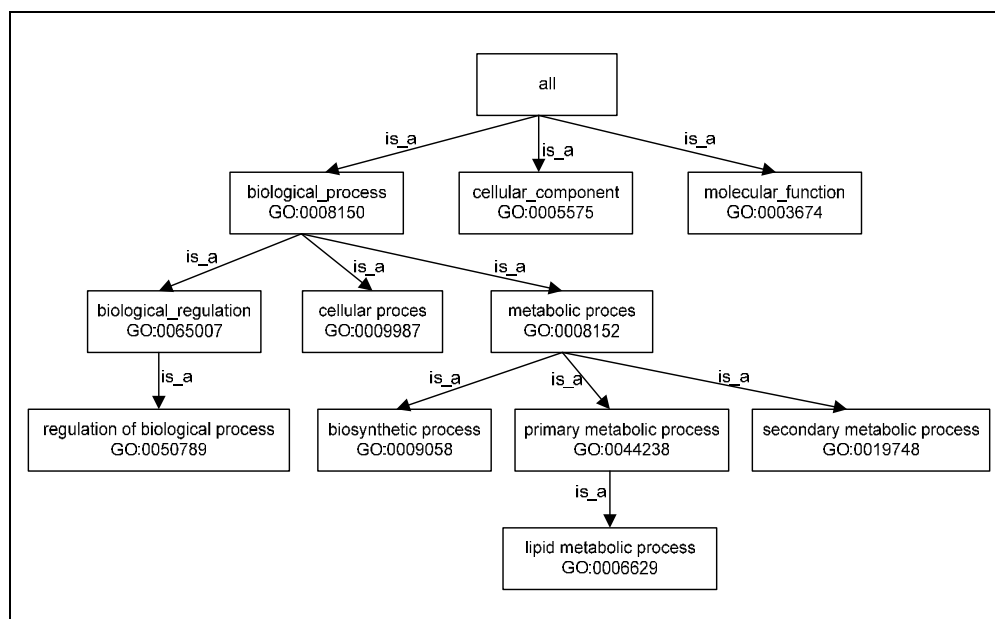
1. การเตรียมตัวอย่างข้อมูล

1.1 ลักษณะของข้อมูลที่นำมาทดลอง ข้อมูลที่นำมาทดลอง ประกอบด้วย ฐานข้อมูลเชิงสัมพันธ์ และออนโทโลยีที่จัดเก็บฐานความรู้ที่สอดคล้องกับฐานข้อมูลเชิงสัมพันธ์นั้น ๆ ซึ่งงานวิจัยนี้ใช้ฐานข้อมูลโปรตีน **Gramene Protein Database (2000)** และ **Gene Ontology**

1.1.1 **Gramene Protein Database** เป็นฐานข้อมูลที่รวบรวมข้อมูลที่เกี่ยวข้องกับโปรตีน จาก พืชตระกูล *Poaceae/Gramineae* จำนวน 78,521 เรคคอร์ด แบ่งเป็น ข้อมูลทั่วไปของโปรตีน (General information) เช่น ชื่อ หมายเลข E.C. , ข้อมูลที่สอดคล้องกับ **Gene Ontology** และ **Plant Ontology** เป็นต้น โดยข้อมูลที่ใช้ในการทดลอง คือ ตาราง **gene_product** แสดงรายละเอียดตารางดังนี้

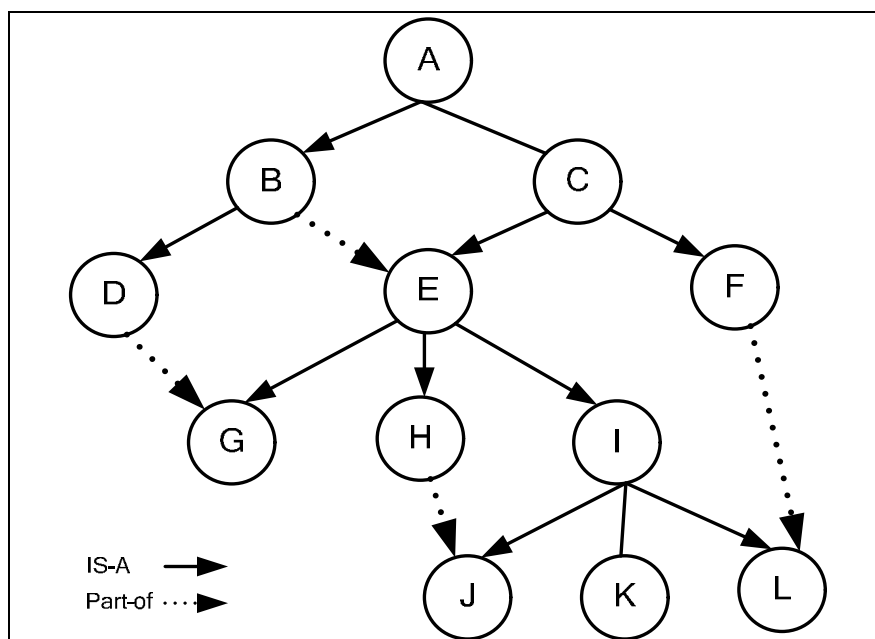
<code>gene_product(gene_id, gene_name, ECnumber, association, ...)</code>

1.1.2 Gene Ontology (GO) รวบรวมข้อมูลที่เกี่ยวข้องกับยีน แบ่งเป็น 3 ส่วนได้แก่ **Molecular function, Biological process** และ **Cellular component** โดยข้อมูลมีความสัมพันธ์กันเป็นแบบลำดับชั้นที่มีความลึก 14 ชั้น ประกอบด้วย 20,080 คอนเซพต์ ที่เชื่อมโยงกันด้วยความสัมพันธ์แบบ **is-a, part-of** แสดงบางส่วนของ **Gene Ontology** ดังภาพที่ 3



ภาพที่ 3 แสดงบางส่วนของ **Gene Ontology**

1.2 โครงสร้างข้อมูล โครงสร้างของออนโทโลยีที่จัดเก็บเป็นกราฟแบบมีทิศทางโหนด แทนออนโทโลยีคอนเซพต์ และเส้นทางระหว่างโหนด แทนความสัมพันธ์ ในออนโทโลยี แต่ละโหนดสามารถมีโหนดพ่อได้มากกว่า 1 โหนด และมีเส้นทางจากโหนดไปถึงทุก ๆ โหนด แสดงลักษณะโครงสร้างข้อมูลดังภาพที่ 4



ภาพที่ 4 ลักษณะโครงสร้างออนโทโลยี

2. การจัดเก็บออนโทโลยีในฐานข้อมูลเชิงสัมพันธ์

ในส่วนนี้อธิบายถึงแนวทางในการจัดเก็บออนโทโลยีในฐานข้อมูลเชิงสัมพันธ์ ได้แก่ ตารางтранซิทีฟโคลสเชอร์(Transitive Closure Table), ตารางтранซิทีฟโคลสเชอร์เมทริกซ์ (Boolean Transitive Closure Matrix) และการอ้างอิงความสัมพันธ์ระหว่างโหนดในออนโทโลยี

21 ตารางтранซิทีฟโคลสเชอร์(Kachai and Waiyamai, 2001) เป็นแนวทางการจัดเก็บความสัมพันธ์ระหว่างโหนดในรูปแบบความสัมพันธ์เชิงวัตถุที่อธิบายว่าแต่ละโหนดในออนโทโลยีมีความสัมพันธ์กันอย่างไร โดยตารางประกอบไปด้วย 3 คอลัมน์ คือ ColID, RowID, Relation โดย RowID จะเก็บหมายเลขประจำโหนดในออนโทโลยี และ ColID เก็บหมายเลขประจำโหนดลูกของโหนดใน RowID โดยพิจารณาความสัมพันธ์ระหว่างโหนดตาม ความสัมพันธ์ที่ระบุในคอลัมน์ Relation จากตัวอย่างออนโทโลยีภาพที่ 4 แสดงตารางтранซิทีฟโคลสเชอร์ได้ ดังตารางที่ 5

ตารางที่ 4 ตารางแสดงหมายเลขโหนด

NodeID	NodeName
1	A
2	B
3	C
4	D
5	E
6	F
7	G
8	H
9	I
10	J
11	K
12	L

ตารางที่ 5 ตารางทรานซิติฟโคลสเชอร์

RowID	ColID	Relation
1	1	Is-a
...
1	3	Is-a
2	5	Part-of
3	5	Is-a
5	5	Is-a
5	7	Is-a
5	8	Is-a
5	9	Is-a

ตารางที่ 5 (ต่อ)

RowID	ColID	Relation
5	10	Is-a
5	11	Is-a
5	12	Is-a
...
12	12	Is-a

2.2 ทรานซิติฟโคลสเซอร์เมทริกซ์ (Kachai and Waiyamai, 2001) การจัดเก็บข้อมูลในรูปแบบนี้ สร้างจากการแปลงความสัมพันธ์ระหว่างโหนดในตารางทรานซิติฟโคลสเซอร์ให้เป็นชุดของไบนารี (Binary) ค่า **0,1** เรียกว่า บิตสตริง (Bit String) โดยขนาดความยาวของบิตสตริงเท่ากับจำนวนโหนดในออนโทโลยี ข้อมูลที่จัดเก็บในรูปแบบบิตสตริงนี้จะใช้ในการสืบค้นร่วมกับลอจิกโอเปอเรเตอร์ **AND, OR** เป็นต้น โดยบิตสตริงในทรานซิติฟโคลสเซอร์เมทริกซ์ แบ่งเป็น 2 รูปแบบดังนี้ แถวบิตสตริง (Row-Bit string) ที่อธิบายว่าโหนดใดบ้างเป็นโหนดลูกของโหนดที่เราพิจารณา ตำแหน่งในบิตสตริงที่มีค่าเท่ากับหมายเลขประจำโหนดโหนดใน **ColID** จากการระบุค่า **RowID** ตำแหน่งบิตนั้นจะมีค่าเป็น **1** ตำแหน่งบิตที่เหลือจะมีค่าเป็น **0** ตัวอย่างเช่น ต้องการหาโหนดลูกของโหนด **E** ทั้งหมด เริ่มต้นให้เลือกค่าของ **ColID** ทั้งหมดที่มีค่า **RowID** เท่ากับหมายเลขโหนดของโหนด **E** คือ **5** และความสัมพันธ์แบบ **is-a** ค่าที่ได้คือ **57891011** และ **12** ให้กำหนดค่าบิตสตริงในตำแหน่งที่ **57891011** และ **12** เป็น **1** ส่วนตำแหน่งที่เหลือ คือ **1 2 3 4** และ **6** กำหนดค่าเป็น **0** แสดงแถวบิตสตริงของโหนด **E** ตามความสัมพันธ์แบบ **is-a** ได้ดังนี้ **000 010111 111** ซึ่งอธิบายได้ว่าโหนดลูกของโหนดนี้ คือ โหนด **E, G, H, I, J, K, L** แสดงขั้นตอนวิธีในการสร้างแถวบิตสตริงได้ดังนี้

อัลกอริทึม 2 การสร้างแถวบิตสตริงของทรานซิติฟโคลสเซอร์เมทริกซ์

Procedure GetRowCode(TermID, RelType, Ontoname_Matrix)

Returns : bit string code

Begin

bitCode = 0;

S = Select ColID From Ontoname_Matrix

Where RowID = TermID and Relation = RelType ;

ForEach $i \in S$

bitCode(i) = 1;

Return(bitCode);

End

รูปแบบที่ 2 คือ คอลัมน์บิตสตริง (**Column-Bit string**) ที่อธิบายว่าโหนดใดบ้างเป็นโหนดบรรพบุรุษของโหนดที่เราพิจารณาดำเนินงานในบิตสตริงที่มีค่าเท่ากับหมายเลขประจำโหนดโหนดใน RowID จากการระบุค่า ColID ตำแหน่งบิตนั้นจะมีค่าเป็น 1 ตำแหน่งบิตที่เหลือจะมีค่าเป็น 0 ตัวอย่างเช่น ต้องการหาโหนดบรรพบุรุษทั้งหมดของโหนด E เริ่มต้นให้เลือกค่าของ RowID ทั้งหมดที่มีค่า ColID เท่ากับ หมายเลขโหนดของโหนด E คือ 5 และความสัมพันธ์แบบ is-a ค่าที่ได้คือ 23 และ 5 ให้กำหนดค่าบิตสตริงในตำแหน่งที่ 23 และ 5 เป็น 1 ส่วนตำแหน่งที่เหลือ คือ 1 4 และ 6-12 กำหนดค่าเป็น 0 แสดงคอลัมน์บิตสตริงของโหนด E ตามความสัมพันธ์แบบ is-a คือ 101 010 000 000 ซึ่งอธิบายได้ว่าโหนดบรรพบุรุษของโหนดนี้ คือ โหนด A, C, E ขั้นตอนวิธีในการสร้างคอลัมน์บิตสตริงแสดงได้ดังนี้

อัลกอริทึม 3 การสร้างคอลัมน์บิตสตริงของทรานซิติฟโคลสเซอร์เมทริกซ์

Procedure GetColCode(TermID, RelType, Ontname_Matrix)

Returns : bit string code

Begin

bitCode = 0;

S = Select RowID From Ontname_Matrix

Where ColID = TermID and Relation = RelType ;

ForEach $i \in S$

bitCode(i) = 1;

Return(bitCode);

End

2.3 การพิจารณาความสัมพันธ์ระหว่างโหนด การจัดเก็บออนโทโลยีในรูปแบบที่เสนอนั้นเก็บทุกโหนดในออนโทโลยีที่มีความสัมพันธ์กัน ในกรณีที่เส้นทางจากโหนดใด ๆ ไปยังโหนดลูกมีหลายเส้นทาง และมีหลายความสัมพันธ์ จึงต้องมีวิธีการพิจารณาความสัมพันธ์ระหว่างโหนดนั้นเพื่อเลือกเส้นทางที่ดีที่สุด และกำหนดความสัมพันธ์ระหว่างโหนดนั้น โดยวิธีการอ้างอิงความสัมพันธ์ระหว่างโหนด (Cruz and Xiao, 2005) ประกอบด้วยขั้นตอนต่อไปนี้

2.3.1 หาเส้นทางทั้งหมดระหว่างโหนด (Path exploration) การพิจารณาหาเส้นทางที่เป็นไปได้ทั้งหมดระหว่างโหนด v และ u ใดๆ แสดงตัวอย่างดังภาพที่ 5 เส้นทางที่เป็นไปได้ทั้งหมดระหว่างโหนด A และ โหนด G

2.3.2 เลือกเส้นทางที่ดีที่สุด (Path selection) ในกรณีที่เส้นทางระหว่างโหนด v และ u ใดๆ มีมากกว่า 1 เส้นทางจะต้องพิจารณาหาเส้นทางที่ดีที่สุดโดยการคำนวณค่าเฉลี่ยของค่าความเหมือนเชิงความหมาย (Semantic similarity) ในแต่ละเส้นทาง และเลือกเส้นทางที่มีค่าเฉลี่ยสูงที่สุด ซึ่งค่าความเหมือนเชิงความหมายจะถูกกำหนดในแต่ละความสัมพันธ์ จากตัวอย่างนี้ความสัมพันธ์แบบ is-a มีค่า 0.8 และ part-of มีค่า 0.5

2.3.3 กำหนดความสัมพันธ์ระหว่างโหนด (Semantic derivation) การกำหนดความสัมพันธ์ระหว่างโหนด v และ u ใดๆ โดยการพิจารณาลำดับความสำคัญ (Priority) ของแต่ละความสัมพันธ์ในเส้นทางระหว่างโหนดที่ได้เลือกไว้ ตามตารางที่กำหนดโดยผู้เชี่ยวชาญ อธิบายขั้นตอนวิธีในการพิจารณาความสัมพันธ์ $Sem(p) = Sem(p_n)$ โดยการทำซ้ำ (Recursive)

เมื่อ $p_n = (r_1 r_2, \dots, r_n)$ และ $r_i (1 \leq i \leq n)$ แทนแต่ละความสัมพันธ์ในเส้นทาง p

$$Sem(p_n) = Sem(p_{n-1}) \wedge Sem(r_n), \quad \text{เมื่อ } n > 1$$

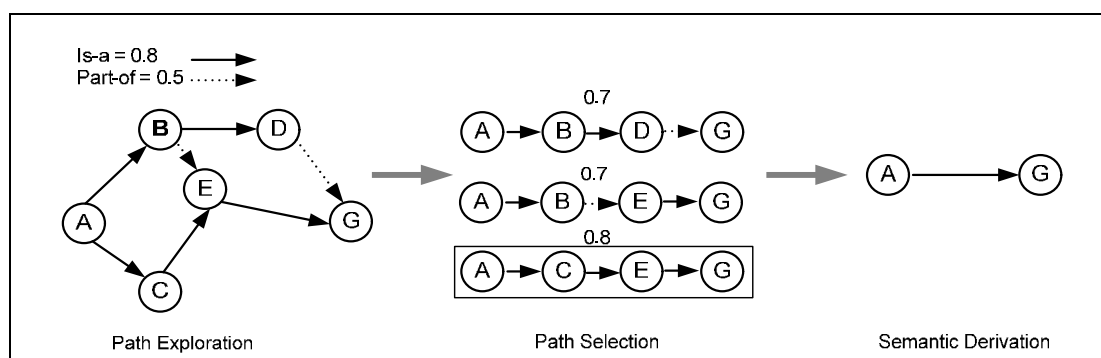
$$Sem(p_n) = \approx, \supset, \neg \quad \text{เมื่อ } n = 1$$

โดยสัญลักษณ์ \approx แทนความสัมพันธ์แบบ **syn-of**, สัญลักษณ์ \supset แทนความสัมพันธ์แบบ **is-a**, สัญลักษณ์ \neg แทนความสัมพันธ์แบบ **part-of** และ สัญลักษณ์ \wedge แทนตัวดำเนินการที่ระบุตามตารางที่กำหนดโดยผู้เชี่ยวชาญ ดังตารางที่ 6

ตารางที่ 6 ตัวดำเนินการ

\dot{U}	\gg	\acute{E}	\emptyset
\gg	\approx	\supset	\neg
\acute{E}	\supset	\supset	\neg
\emptyset	\neg	\neg	\neg

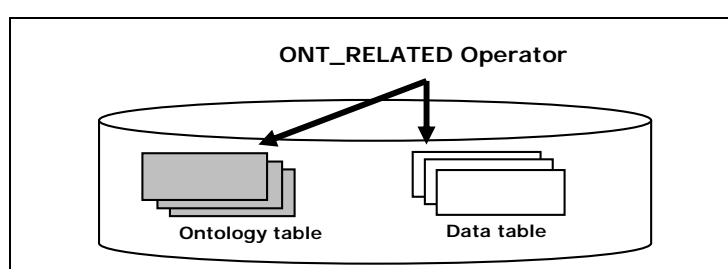
จากขั้นตอนในข้อ 2.31-2.33 แสดงภาพขั้นตอนการพิจารณาขั้นตอนการสัมพันธ์ระหว่างโหนดได้ดังภาพที่ 5



ภาพที่ 5 ขั้นตอนการพิจารณาความสัมพันธ์ระหว่างโหนด

3 การพัฒนาส่วนสืบค้นข้อมูลเชิงความหมายในฐานข้อมูลเชิงสัมพันธ์

การพัฒนาส่วนสืบค้นข้อมูลเชิงความหมายในฐานข้อมูลเชิงสัมพันธ์ประกอบด้วยส่วนสำคัญ 2 ส่วน คือ การจัดเก็บออนโทโลยีในฐานข้อมูลเชิงสัมพันธ์ และการพัฒนาโอเปอเรเตอร์สืบค้นเชิงความหมาย ซึ่งโอเปอเรเตอร์สืบค้นเชิงความหมายจะสืบค้นข้อมูลในตารางข้อมูล(Data table) ที่ต้องการ โดยพิจารณาเงื่อนไขที่สอดคล้องกับผลการสืบค้นข้อมูลในตารางจัดเก็บออนโทโลยี(Ontology table) แสดงภาพรวมการทำงานได้ดังภาพที่ 6



ภาพที่ 6 ภาพรวมการทำงานของระบบ

31 โครงสร้างตารางจัดเก็บออนโทโลยี ตารางจัดเก็บออนโทโลยี ประกอบด้วย 2 ตาราง คือ **Onto_Table** และ **Matrix_Table** อธิบายแต่ละส่วนได้ดังต่อไปนี้

31.1 Onto_Table ตารางเก็บข้อมูลแต่ละโหนด หรือคอนเซพในออนโทโลยี เช่น **TermID** (หมายเลขประจำโหนด), **Term** (ชื่อโหนด), **ParentTermID** (หมายเลขของโหนดพ่อ) เป็นต้น ตารางข้อมูลนี้ใช้สำหรับอ้างอิงรายละเอียดของโหนดในออนโทโลยี

Onto_Table (TermID, Term, ParentTermID , OntologyID, Leaf,...)
--

31.2 Matrix_Table ตารางจัดเก็บโครงสร้างของออนโทโลยีตามรูปแบบตารางทรานซีทิฟโคลสเชอร์ ที่อธิบายในหัวข้อ 21 เพื่อแสดงความสัมพันธ์ระหว่างโหนด โดย **RowID** และ **ColID** เก็บเลขหมายเลขประจำโหนด จึงสามารถอ้างอิงรายละเอียดของแต่ละโหนดใน **Onto_Table** ได้ และ **Relation** ระบุความสัมพันธ์ระหว่างโหนด เช่น **is-a, part-of**

Matrix_Table(RowID, ColID, Relation, ...)

32 การพัฒนาโอเปอเรเตอร์สืบค้นเชิงความหมาย **ONT_RELATED** โอเปอเรเตอร์สืบค้นตารางออนโทโลยีเพื่อหาออนโทโลยี คอนเซพที่สอดคล้องกับคำถามสืบค้นตามความสัมพันธ์ที่กำหนด แล้วนำไปสืบค้นในตารางข้อมูล เพื่อแสดงผลการสืบค้นให้กับผู้ใช้ แสดงรูปแบบคำสั่ง โอเปอเรเตอร์ได้ดังนี้

ONT_RELATED(Col_name, RelType, Tem[AND\OR\NOT Tem], OntologyName)
Return Integer

ONT_RELATED โอเปอเรเตอร์จะพิจารณาข้อมูลในคอลัมน์ที่ระบุโดย **Col_name** ว่าตรงกัน กับออนโทโลยีคอนเซพในออนโทโลยีระบุโดย **OntologyName** ที่สอดคล้องกับคำสืบค้นที่ระบุโดย **Tem** ตามความสัมพันธ์ที่ระบุโดย **RelType** ถ้ามีข้อมูลในตารางข้อมูลตรงกันกับออนโทโลยีคอนเซพจะโอเปอเรเตอร์จะคืนค่ากลับเป็น **1** ถ้าไม่มีข้อมูลในตารางข้อมูลตรงกันกับออนโทโลยีคอนเซพจะโอเปอเรเตอร์จะคืนค่ากลับเป็น **0** เพื่อเพิ่มประสิทธิภาพสืบค้นโอเปอเรเตอร์ยังสามารถสร้างคำสืบค้นที่ซับซ้อนมากขึ้นโดยการสืบค้นร่วมกับลอจิกโอเปอเรเตอร์ **AND, OR, NOT** เป็นต้น

321 การพัฒนาโอเปอเรเตอร์สืบค้นพื้นฐาน โอเปอเรเตอร์สืบค้นข้อมูลในออนโทโลยีตาราง **Matrix_Table** ที่จัดเก็บข้อมูลในรูปแบบตารางทรานซิทีฟโคลสเซอร์โดยพิจารณาหาค่า **ColID** ที่มี **RowID** ตรงกับหมายเลขประจำโหนดที่เป็นคำสืบค้น และความสัมพันธ์ระหว่าง **ColID** กับ **RowID** ตรงกันกับ **RelType** ผลลัพธ์ที่ได้ คือ ออนโทโลยีคอนเซพที่สอดคล้องกับคำสืบค้น สำหรับนำไปสืบค้นในฐานะข้อมูลต่อไป แสดงขั้นตอนวิธีโปรแกรมย่อยได้ดังนี้

อัลกอริทึม 4 การหาโหนดลูกทั้งหมดของโหนดในออนโทโลยีที่สอดคล้องกับคำสืบค้น

Procedure **ONT_EXPAND**(Term, RelType, Ontoname_table, Ontoname)

Returns array of String

Begin

Term_ID=Mapping_ID(Term, Ontoname_table);

S = Select ColID From Ontoname_Matrix

Where RowID = Term_ID and Relation= RelType;

For each $i \in S$

If ChkInstance(i, Ontoname_table) = True

instance(j) = Mapping_Term(i, Ontoname_table);

Return(instance);

End

3.2.2 การพัฒนาโอเปอเรเตอร์สืบค้นร่วมกับลอจิกโอเปอเรเตอร์

3.2.2.1 การพัฒนาโอเปอเรเตอร์ **ONT_RELATED_AND** โอเปอเรเตอร์

สืบค้นข้อมูลในออนโทโลยีตาราง **Matrix_Table** ที่จัดเก็บข้อมูลในรูปแบบตารางทรานซิทีฟโคลสเซอร์โดยพิจารณาหาค่า **ColID** ที่มี **RowID** ตรงกับหมายเลขประจำโหนดที่เป็นคำสืบค้น และความสัมพันธ์ระหว่าง **ColID** กับ **RowID** ตรงกันกับ **RelType** นำผลลัพธ์ที่ได้มาหาสมาชิกร่วมกัน (**Intersect**) หลังจากนั้นจึงแปลงกลับเป็นชื่อโหนดในตารางออนโทโลยี เพื่อนำไปใช้เป็นคำสืบค้นในฐานะข้อมูล แสดงขั้นตอนวิธีโปรแกรมย่อได้ดังนี้

อัลกอริทึม 5 การหาโหนดลูกทั้งหมดของโหนดในออนโทโลยีโดยระบุ AND โอเปอเรเตอร์

Procedure **ONT_RELATED_AND**(Term1, RelType, Term2, Ontoname_table)

Returns array of String

Begin

Term_ID1=Mapping_ID(Term1, Ontoname_table);

Term_ID2=Mapping_ID(Term2, Ontoname_table);

S = Select ColID From Ontoname_Matrix

Where RowID = Term_ID1 and Relation = RelType

Intersect

Select ColID From Ontoname_Matrix

Where RowID = Term_ID2 and Relation = RelType;

For each $i \in S$

If ChkInstance(i, Ontoname_table) = True

instance(j) = Mapping_Term(i, Ontoname_table);

Return(instance);

End

3.2.2.2 การพัฒนาโอเปอเรเตอร์ **ONT_RELATED_OR** โอเปอเรเตอร์สืบค้น

ข้อมูลในออนโทโลยีตาราง **Matrix_Table** ที่จัดเก็บข้อมูลในรูปแบบตารางทราเนซิติฟโคลสเซอร์ โดยพิจารณาหาค่า **ColID** ที่มี **RowID** ตรงกับหมายเลขประจำโหนดที่เป็นคำสืบค้น และความสัมพันธ์ระหว่าง **ColID** กับ **RowID** ตรงกันกับ **RelType** นำผลลัพธ์ที่ได้มาหาสมาชิกรวมกัน (**Union**) หลังจากนั้นจึงแปลงกลับเป็นชื่อโหนดในตารางออนโทโลยี เพื่อนำไปใช้เป็นคำสั่งสืบค้นในฐานข้อมูล แสดงขั้นตอนวิธีโปรแกรมย่อได้ดังนี้

อัลกอริทึม 6 การหาโหนดลูกทั้งหมดของโหนดในออนโทโลยีโดยระบุ **OR**โอเปอเรเตอร์

Procedure ONT_RELATED_OR(Term1, RelType, Term2, Ontoname_table)

Returns array of String

Begin

Term_ID1=Mapping_ID(Term1, Ontoname_table);

Term_ID2=Mapping_ID(Term2, Ontoname_table);

S = Select ColID From Ontoname_Matrix

Where RowID = Term_ID1 and Relation= RelType

Union

Select ColID From Ontoname_Matrix

Where RowID = Term_ID2 and Relation= RelType;

For each $i \in S$

If ChkInstance(i, Ontoname_table) = True

instance(j) = Mapping_Term(i, Ontoname_table);

Return(instance);

End

3.2.2.3 การพัฒนาโอเปอเรเตอร์สืบค้น ONT_RELATED_NOT

โอเปอเรเตอร์นี้ทำการสืบค้นหาสมาชิกทั้งหมดที่ไม่เกี่ยวข้องกับคำสืบค้น คือ โหนดที่ไม่ได้เป็นทั้งโหนดลูก และโหนดบรรพบุรุษของคำสืบค้น การทำงานของโอเปอเรเตอร์นี้ เริ่มต้นจากการหาโหนดลูก และโหนดบรรพบุรุษของคำสืบค้นแสดงในรูปแบบบิตสตริงตามรูปแบบ ทรานซิทีฟโคลสเซอร์เมทริกซ์ ในหัวข้อ 2.2 โดยการประมวลผลขั้นตอนวิธี **GetRowCode**, **GetColCode** นำบิตสตริงที่ได้ มาหาสมาชิกรวมกันโดยใช้ลอจิกโอเปอเรเตอร์ **OR** แล้วพิจารณาเฉพาะบิตสตริงที่มีค่าเป็น 0

อัลกอริทึม 7 การหาโหนดทั้งหมดที่ไม่ได้เป็นสมาชิกของโหนดในออนโทโลยีที่กำหนด
โดยระบบ NOTโอเปอร์เรเตอร์

Procedure ONT_RELATED_NOT(TermID1,TermID2,Ontoname_table)

Return array of string

Begin

While TermID1, TermID2 is not null

Begin

CodeTerm1 = getRowcode(TermID1)

CodeTerm2=getColcode(TermID2)

Result = CodeTerm1 U CodeTerm2

End

For i=0 to code length

If result(i) = 0 then

If chkInstance(element(i)) = True

instance(j) = Mapping_Term(element(i), Ontoname_table);

End if

Return (Instance)

End

4 วิธีทดสอบกับข้อมูลตัวอย่าง

การทดสอบโอเปอร์เรเตอร์สืบค้นเชิงความหมายที่พัฒนาขึ้นนั้นจะแบ่งการทดสอบเป็น 2 ลักษณะ ได้แก่ การทดสอบการทำงานของโอเปอร์เรเตอร์โดยมีลำดับชั้น(Level) ของออนโทโลยีแตกต่างกัน และ ทดสอบความครบถ้วนของผลการสืบค้น โดยเปรียบเทียบโอเปอร์เรเตอร์ที่เสนอกับการสืบค้นข้อมูลโดยใช้คำสั่ง “START WITH” และ “CONNECT BY” มีรายละเอียดดังนี้

41 การทดสอบการทำงานของโอเปอร์เรเตอร์โดยมีลำดับชั้นของออนโทโลยีต่างกัน ทดสอบความเร็วในการสืบค้นโดยการสุ่มตัวอย่างจากระบบฐานข้อมูลมาในแต่ละลำดับชั้นของออนโทโลยี ซึ่งมีค่าอยู่ระหว่าง 1-13 ทำการค้นหาคำตอบของโอเปอร์เรเตอร์ที่พัฒนา จากนั้นทำการ

วัดเวลาในการประมวลผลบันทึกผลและหาค่าเฉลี่ยของแต่ละโอเปอเรเตอร์เปรียบเทียบกับคำสั่ง “START WITH” และ “CONNECT BY” โดยวิธีการใช้คำสั่ง “START WITH” และ “CONNECT BY” หาคำตอบแต่ละโอเปอเรเตอร์มีรายละเอียดดังนี้

41.1 การหาคำตอบของโอเปอเรเตอร์ **ONT_RELATED** ทำได้โดยการหาโหนดลูกทั้งหมดของคำสืบค้น ตามความสัมพันธ์ที่กำหนด แสดงคำสั่ง SQL ได้ดังนี้

```
SELECT TermID FROM Onto_Table
START WITH TermID = input_nodeID
CONNECT BY ParentTermID = PRIOR TermID
AND Relation = input_relation
```

41.2 การหาคำตอบของโอเปอเรเตอร์ **ONT_RELATED_AND** ทำได้โดยการหาโหนดลูกทั้งหมดของคำสืบค้นตามความสัมพันธ์ที่กำหนด แล้วหาสมาชิกร่วมกัน แสดงคำสั่ง SQL ได้ดังนี้

```
SELECT TermID FROM Onto_Table
START WITH TermID = input_nodeID1
CONNECT BY ParentTermID = PRIOR TermID
AND Relation = input_relation
```

INTERSECT

```
SELECT TermID FROM Onto_Table
START WITH TermID = input_nodeID2
CONNECT BY ParentTermID = PRIOR TermID
AND Relation = input_relation
```

41.3 การหาคำตอบของโอเปอเรเตอร์ **ONT_RELATED_OR** ทำได้โดยการหาโหนดลูกทั้งหมดของคำสืบค้นตามความสัมพันธ์ที่กำหนด แล้วหาสมาชิกรวมกัน แสดงคำสั่ง SQL ได้ดังนี้

```

SELECT TermID FROM Onto_Table
START WITH TermID = input_nodeID1
CONNECT BY ParentTermID = PRIOR TermID
AND Relation = input_relation

```

```

UNION

```

```

SELECT TermID FROM Onto_Table
START WITH TermID = input_nodeID2
CONNECT BY ParentTermID = PRIOR TermID
AND Relation = input_relation

```

41.4 การหาคำตอบของโอเปอเรเตอร์ **ONT_RELATED_NOT** ทำการหาโหนดลูก และโหนดบรรพบุรุษของคำสืบค้นตามความสัมพันธ์ที่กำหนด แล้วหาสมาชิกรวมกัน จากนั้นพิจารณาผลลัพธ์ในส่วนที่ไม่อยู่ในกลุ่มสมาชิกนั้นโดยใช้ โอเปอเรเตอร์ **MINUS** แสดงคำสั่ง **SQL** ได้ดังนี้

```

SELECT DISTINCT(TermID) FROM Onto_Table
MINUS
(
    SELECT TermID FROM Onto_Table
    START WITH TermID = input_nodeID1
    CONNECT BY ParentTermID = PRIOR TermID
    AND Relation = input_relation

    UNION

    SELECT TermID FROM Onto_Table
    START WITH TermID = input_nodeID2
    CONNECT BY TermID = PRIOR ParentTermID
    AND Relation = input_relation
)

```

42 การทดสอบความครบถ้วนของผลการสืบค้น โดยการประมาณค่าการสูญหายของผลการสืบค้น (**Information loss**) กำหนดคำสั่งสืบค้นจำนวน 7 คำถาม นำคำสั่งสืบค้นเหล่านี้ไปสืบค้นในฐานข้อมูล **Gramene Protein** แล้วพิจารณาผลการสืบค้นเพื่อประมาณค่าการสูญหาย (**Estimating Information Loss**) ของผลการสืบค้น เปรียบเทียบกับผลการสืบค้นจากคำสั่ง “**START WITH**” และ “**CONNECT BY**”

การประมาณค่าการสูญหายของผลการสืบค้น (Mera *et al*, 2000) เป็นการวัดค่าการสูญหายของผลการสืบค้นที่เกิดจากการแทนคำสั่งสืบค้นของผู้ใช้ด้วยคำในออนโทโลยีที่สอดคล้องกับคำสั่งสืบค้นนั้น ๆ ตามความสัมพันธ์ที่กำหนด ซึ่งเป็นการรวมกันระหว่างการวัดผลโดยใช้ค่าความถูกต้อง (**Precision**) และความครบถ้วน (**Recall**) แสดงสูตรในการวัดค่าการสูญหายของผลการสืบค้นได้ดังนี้

$$\text{Loss} = 1 - \frac{1}{a \left(\frac{1}{\text{precision}} \right) + (1-a) \left(\frac{1}{\text{Recall}} \right)}$$

โดยที่ α ($0 \leq \alpha \leq 1$) คือ พารามิเตอร์สำหรับระบุค่าความสำคัญระหว่างค่าความครบถ้วน และค่าความถูกต้อง เมื่อกำหนด

U-Term แทน คำสืบค้นจากผู้ใช้

Ext(U-Term) แทนจำนวนผลการสืบค้นจากการใช้คำสั่งสืบค้นจากผู้ใช้ (**RelevantSet**)

O-Term แทน คำในออนโทโลยีที่นำมาแทนคำสั่งสืบค้นจากผู้ใช้

Ext(O-Term) แทนจำนวนผลการสืบค้นจากการใช้ คำในออนโทโลยีที่นำมาแทนคำสั่งสืบค้นจากผู้ใช้ (**RetrievedSet**)

ค่าความครบถ้วนของผลการสืบค้นคิดได้จากอัตราส่วนระหว่างจำนวนสมาชิกร่วมของผลการสืบค้นของคำสั่งสืบค้น และคำในออนโทโลยีที่นำมาแทน กับผลการสืบค้นของคำสั่งสืบค้น

$$\text{Recall} = \frac{|\text{RelevantSet} \cap \text{RetrievedSet}|}{|\text{RelevantSet}|} = \frac{|\text{Ext(U-Term)} \cap \text{Ext(O-Term)}|}{|\text{Ext(U-Term)}|}$$

ค่าความถูกต้องของผลการสืบค้นคิดได้จากอัตราส่วนระหว่างจำนวนสมาชิกร่วมของผลการสืบค้นของคำสืบค้น และคำในออนโทโลยีที่นำมาแทน กับผลการสืบค้นของคำในออนโทโลยี

$$\text{Precision} = \frac{|\text{RelevantSet} \cap \text{RetrievedSet}|}{|\text{RetrievedSet}|} = \frac{|\text{Ext}(U - \text{Term}) \cap \text{Ext}(O - \text{Term})|}{|\text{Ext}(O - \text{Term})|}$$

การคำนวณค่าความครบถ้วน และค่าความถูกต้องตามสูตรข้างต้น จะเปลี่ยนแปลงไปตามกรณีต่าง ๆ ในการแทนคำสืบค้นด้วยกลุ่มคำในออนโทโลยี ซึ่งในที่นี้จะพิจารณาเฉพาะกรณีที่คำในออนโทโลยีที่มาแทนเป็นสมาชิก หรือเป็นส่วนหนึ่งของคำสืบค้นจากผู้ใช้ ($\text{Ext}(O - \text{Term}) \subseteq \text{Ext}(U - \text{Term})$) จะได้ว่า

$$\text{Precision} = 1$$

$$\text{Recall}_{\text{low}} = \frac{|\text{Ext}(O - \text{Term})|_{\text{low}}}{|\text{Ext}(O - \text{Term})|_{\text{low}} + |\text{Ext}(U - \text{Term})|}$$

$$\text{Recall}_{\text{high}} = \frac{|\text{Ext}(O - \text{Term})|_{\text{high}}}{\max[|\text{Ext}(O - \text{Term})|_{\text{high}}, |\text{Ext}(U - \text{Term})|]}$$

ค่าความถูกต้องมีค่าเท่ากับ 1 เนื่องจากคำในออนโทโลยีที่มาแทนเป็นสมาชิกของคำสืบค้นจากผู้ใช้งาน ดังนั้นการหาสมาชิกร่วมระหว่างผลการสืบค้นของคำสืบค้น และคำในออนโทโลยีที่นำมาแทนจะมีค่าเท่ากับผลการสืบค้นคำในออนโทโลยีที่นำมาแทน ($\text{Ext}(U - \text{Term}) \cap \text{Ext}(O - \text{Term}) = \text{Ext}(O - \text{Term})$) เนื่องจากการหาสมาชิกร่วมจะได้ผลลัพธ์เป็นสมาชิกของเซตที่เล็กที่สุด

ค่าความครบถ้วนต้องแยกพิจารณาเป็น 2 กรณี คือ กรณีค่าความครบถ้วนน้อยที่สุดที่เป็นไปได้ และค่าความครบถ้วนมากที่สุดที่เป็นไปได้ ตามเงื่อนไขการหาสมาชิกร่วมระหว่างผลการสืบค้นของคำสืบค้นและคำในออนโทโลยีที่นำมาแทน เนื่องจากคำในออนโทโลยีที่มาแทนเป็นสมาชิกของคำสืบค้นจากผู้ใช้งาน ดังนั้นค่าผลการสืบค้นของคำสืบค้นจากผู้ใช้งานจะเท่ากับการหาสมาชิกร่วมระหว่างผลการสืบค้นของคำสืบค้น และคำในออนโทโลยีที่นำมาแทน ($\text{Ext}(U - \text{Term}) = |\text{Ext}(U - \text{Term}) \cup \text{Ext}(O - \text{Term})|$) ซึ่งการพิจารณาการหาสมาชิกรวมแยกเป็น 2 กรณี

กรณีค่าสมาชิกรวมน้อยที่สุด คือ กรณีที่สมาชิกทั้งสองเซตไม่ซ้อนทับกันเลย ดังนั้นค่าสมาชิกรวมจะเท่ากับค่าของสมาชิกในเซตที่ใหญ่ที่สุด

$$|\text{Ext}(\text{U-Tem}) \cup \text{Ext}(\text{O-Tem})|_{\text{low}} = \max[|\text{Ext}(\text{U-Tem})|, |\text{Ext}(\text{O-Tem})|]$$

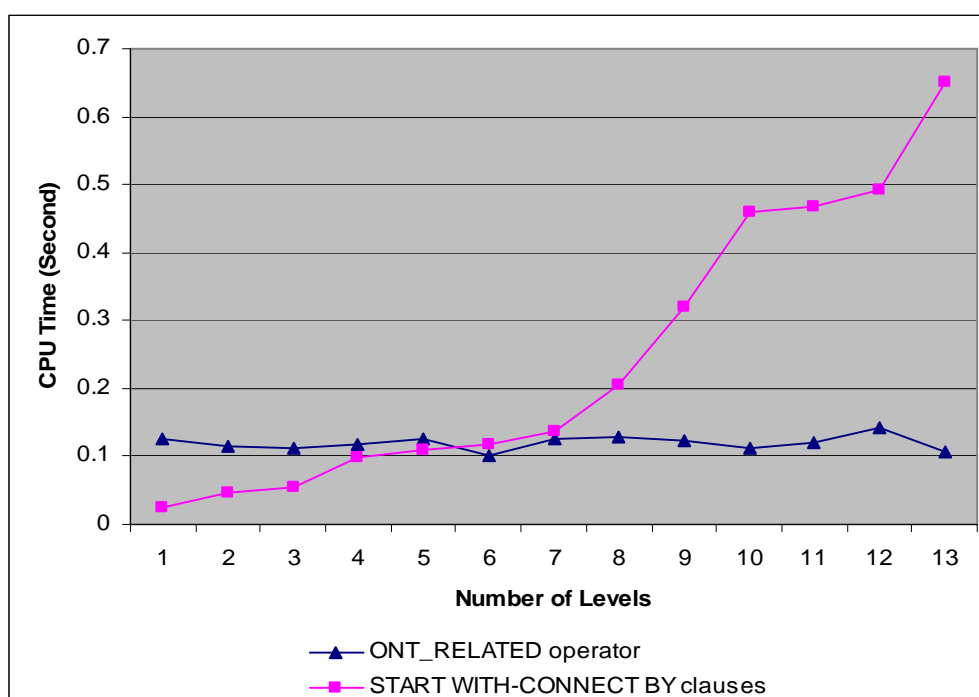
และกรณีสมาชิกรวมมากที่สุด ค่าสมาชิกรวมมีค่าเท่ากับผลบวกของสมาชิกทั้งสองเซต

$$|\text{Ext}(\text{U-Tem}) \cup \text{Ext}(\text{O-Tem})|_{\text{high}} = |\text{Ext}(\text{U-Tem})| + |\text{Ext}(\text{O-Tem})|$$

ผลและวิจารณ์

1. การทดสอบการทำงานของโอเปอเรเตอร์โดยมีลำดับชั้นของออนโทโลยีต่างกัน

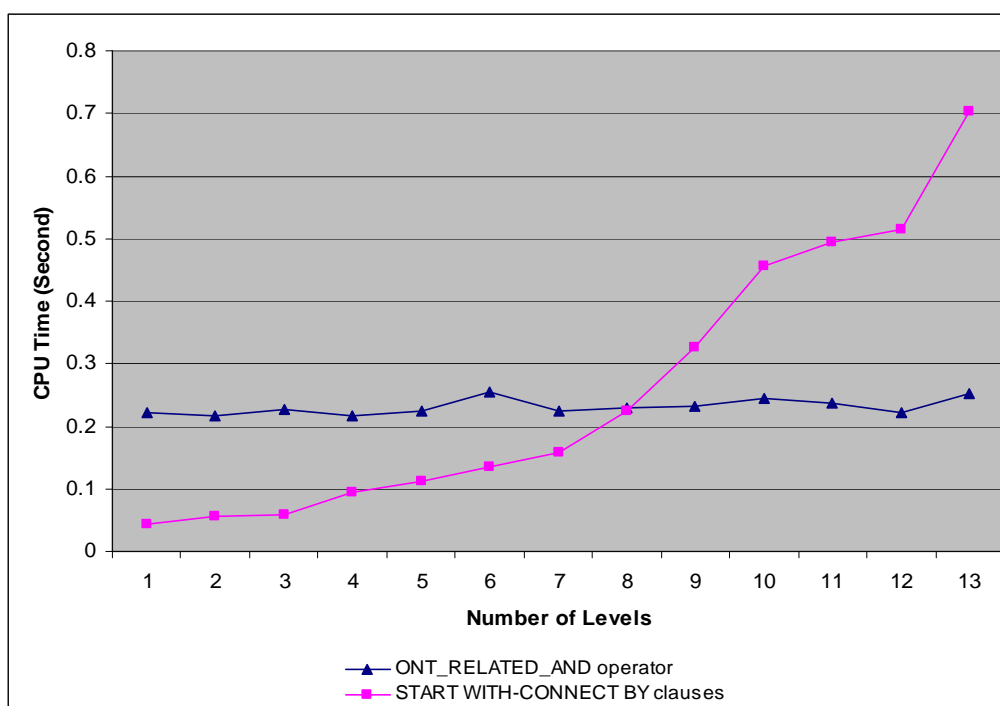
ทดสอบการทำงานของโอเปอเรเตอร์ **ONT_RELATED**, **ONT_RELATED_AND**, **ONT_RELATED_OR**, **ONT_RELATED_NOT** ที่พัฒนาขึ้น โดยนำโอเปอเรเตอร์มาสืบค้นข้อมูลในระบบฐานข้อมูลเชิงสัมพันธ์โดยกำหนดค่าสืบค้นในแต่ละลำดับชั้นของออนโทโลยี เปรียบเทียบเวลาที่ใช้ในการประมวลผลสืบค้นกับคำสั่ง **“START WITH”** และ **“CONNECT BY”** ได้ผลดังนี้



ภาพที่ 7 เวลาเฉลี่ยของ CPU ที่ใช้ในการหาคำตอบของโอเปอเรเตอร์ **ONT_RELATED**

ภาพที่ 7 แสดงผลการทดสอบการทำงานของโอเปอเรเตอร์ **ONT_RELATED** ที่พัฒนาขึ้น เปรียบเทียบกับคำสั่ง **“START WITH”** และ **“CONNECT BY”** แกนตั้งแสดงเวลาที่ซีพียูประมวลผลในหน่วย วินาที(second) โดยรวมเวลาการอ่านข้อมูลจากฐานข้อมูล แกนนอนแสดงจำนวนระดับชั้นของออนโทโลยีที่ต้องการสืบค้นตั้งแต่ระดับที่ 1 ถึง 13 จะเห็นได้ว่าโอเปอเรเตอร์ที่เสนอใช้เวลาในการสืบค้นน้อยกว่า คำสั่ง **“START WITH”** และ **“CONNECT BY”** เมื่อจำนวน

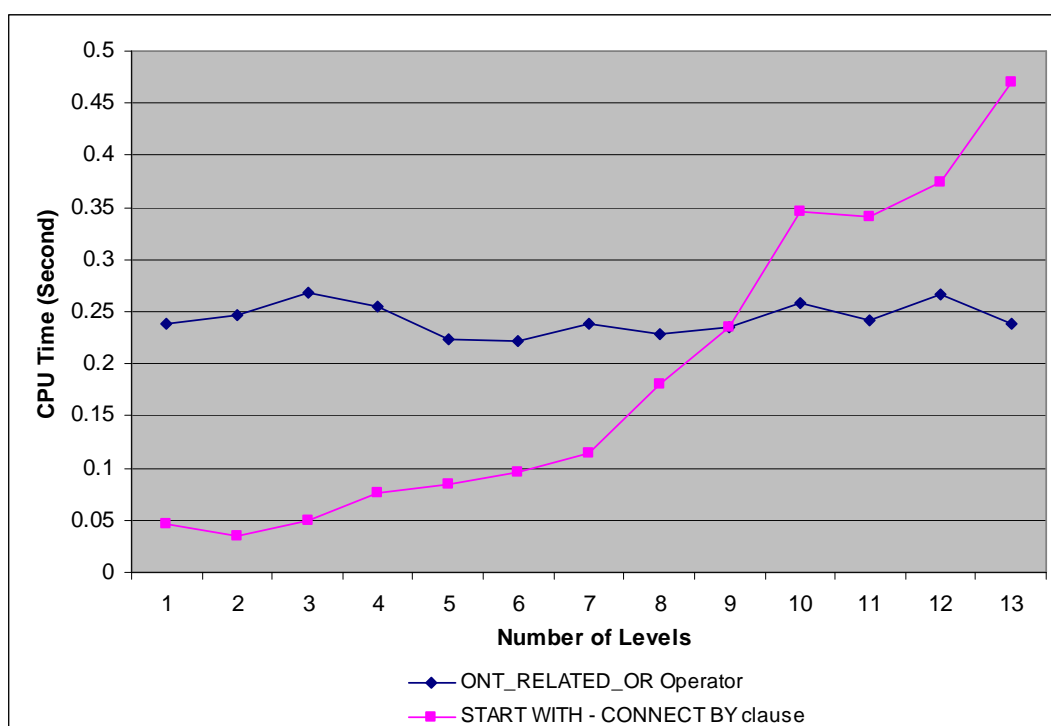
ระดับชั้นสูงขึ้น ซึ่งเวลาที่ใช้ในการประมวลผลของโอเปอเรเตอร์ **ONT_RELATED** มีลักษณะคงที่เป็นเชิงเส้น (**Linear**) แตกต่างจากเวลาที่ใช้ในการประมวลผลของคำสั่ง **“START WITH”** และ **“CONNECT BY”** ที่เพิ่มขึ้นเป็นเอ็กโพเนนเชียล (**Exponential**) เมื่อจำนวนระดับชั้นสูงขึ้น เนื่องจากคำสั่ง **“START WITH”** และ **“CONNECT BY”** ค้นหาข้อมูลแบบเชิงลึก ดังนั้นจึงสรุปได้ว่าโอเปอเรเตอร์ที่เสนอมีประสิทธิภาพดีกว่าคำสั่ง **“START WITH”** และ **“CONNECT BY”**



ภาพที่ 8 เวลาเฉลี่ยของ CPU ที่ใช้ในการหาคำตอบของโอเปอเรเตอร์ **ONT_RELATED_AND**

ภาพที่ 8 แสดงผลการทดสอบการทำงานโอเปอเรเตอร์ **ONT_RELATED_AND** ที่พัฒนาขึ้นเปรียบเทียบกับ คำสั่ง **“START WITH”** และ **“CONNECT BY”** แกนตั้งแสดงเวลาที่ซีพียูประมวลผลในหน่วยวินาทีโดยรวมเวลาการอ่านข้อมูลจากฐานข้อมูล แกนนอนแสดงจำนวนระดับชั้นของออนโทโลยีที่ต้องการสืบค้น ตั้งแต่ระดับที่ 1 ถึง 13 จากกราฟแสดงเวลาเฉลี่ย ที่จำนวนระดับชั้นของออนโทโลยีมีค่า 1-7 คำสั่ง **“START WITH”** และ **“CONNECT BY”** ใช้เวลาในการสืบค้นน้อยกว่า เนื่องจากจำนวนครั้งในการวนลูปสืบค้นมีค่าน้อย ในขณะที่โอเปอเรเตอร์ที่เสนอจะต้องเสียเวลาในการแปลงออนโทโลยีที่จัดเก็บในฐานข้อมูลให้อยู่ในรูปแบบบิตสตริงเพื่อดำเนินการลอจิกโอเปอเรเตอร์**AND** แต่เมื่อจำนวนระดับชั้นสูงขึ้น **ONT_RELATED_AND** โอเปอเรเตอร์ใช้เวลาในการสืบค้นน้อยกว่า และเมื่อพิจารณาโดยภาพรวมแล้ว โอเปอเรเตอร์ที่

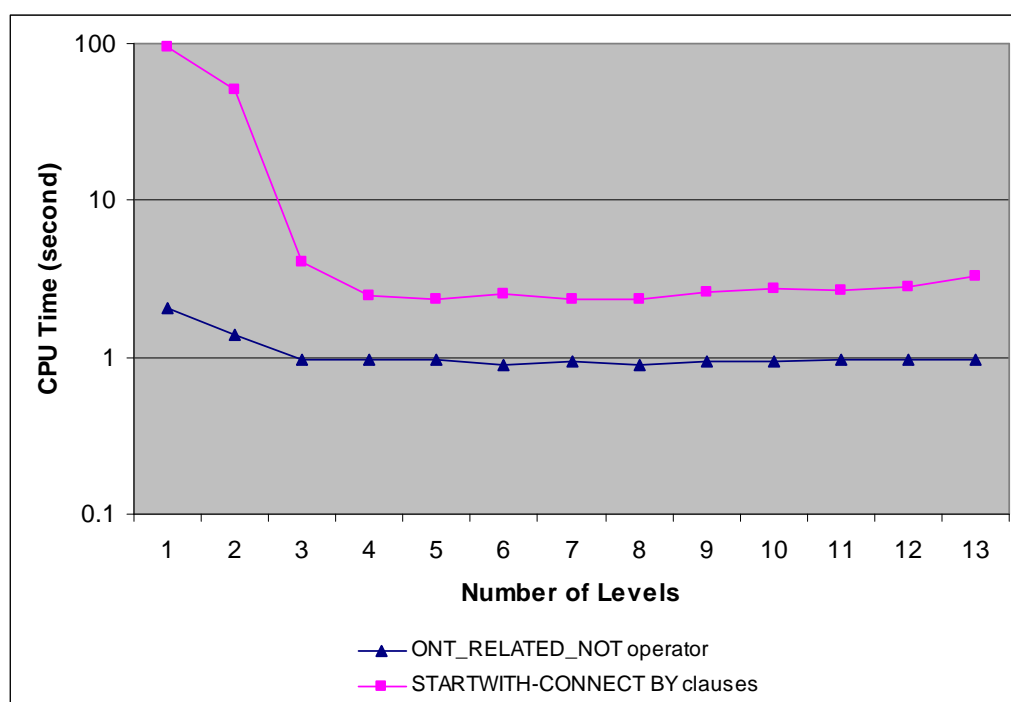
พัฒนาขึ้นมาใหม่ใช้เวลาในการสืบค้นลักษณะเป็นเชิงเส้นแตกต่างจากเวลาที่ใช้ในการประมวลผลของคำสั่ง **“START WITH”** และ **“CONNECT BY”** ที่มีลักษณะเพิ่มขึ้นเป็น เอ็กโพเนนเชียล เมื่อจำนวนระดับชั้นสูงขึ้น เนื่องจากคำสั่ง **“START WITH”** และ **“CONNECT BY”** ค้นหาข้อมูลแบบเชิงลึก ดังนั้นจึงสรุปได้ว่าโอเปอเรเตอร์ที่เสนอใหม่มีประสิทธิภาพดีกว่า คำสั่ง **“START WITH”** และ **“CONNECT BY”**



ภาพที่ 9 เวลาเฉลี่ยของ CPU ที่ใช้ในการหาคำตอบของโอเปอเรเตอร์ **ONT_RELATED_OR**

ภาพที่ 9 แสดงผลการทดสอบการทำงานของโอเปอเรเตอร์ **ONT_RELATED_OR** ที่พัฒนาขึ้นเปรียบเทียบกับ คำสั่ง **“START WITH”** และ **“CONNECT BY”** แกนตั้งแสดงเวลาที่ซีพียูประมวลผลในหน่วยวินาทีโดยรวมเวลาการอ่านข้อมูลจากฐานข้อมูล แกนนอนแสดงจำนวนระดับชั้นของออนโทโลยีที่ต้องการสืบค้น ตั้งแต่ระดับที่ 1 ถึง 13 จะเห็นได้ว่าผลที่ได้จะคล้ายกับเวลาเฉลี่ยที่ใช้ในการหาคำตอบของโอเปอเรเตอร์ **ONT_RELATED_AND** แต่โอเปอเรเตอร์ **ONT_RELATED_OR** จะใช้เวลามากกว่าเนื่องจากจำนวนข้อมูลที่เป็นผลการสืบค้นจะมากกว่า และเมื่อเปรียบเทียบกับคำสั่ง **“START WITH”** และ **“CONNECT BY”** โอเปอเรเตอร์ที่เสนอใช้เวลาในการสืบค้นน้อยกว่า เมื่อจำนวนระดับชั้นสูงขึ้น ซึ่งเวลาที่ใช้ในการประมวลผลของ

โอเปอเรเตอร์ **ONT_RELATED_OR** มีลักษณะเป็นเชิงเส้นแตกต่างจากเวลาที่ใช้ในการประมวลผลของคำสั่ง **“START WITH”** และ **“CONNECT BY”** ที่มีลักษณะเพิ่มขึ้นเป็นเอ็กโพเนนเชียล เมื่อจำนวนระดับชั้นสูงขึ้น เนื่องจากคำสั่ง **“START WITH”** และ **“CONNECT BY”** ค้นหาข้อมูลแบบเชิงลึก ดังนั้นจึงสรุปได้ว่าโอเปอเรเตอร์ที่เสนอใหม่มีประสิทธิภาพดีกว่า คำสั่ง **“START WITH”** และ **“CONNECT BY”**



ภาพที่ 10 เวลาเฉลี่ยของ CPU ที่ใช้ในการหาคำตอบของโอเปอเรเตอร์ **ONT_RELATED_NOT**

ภาพที่ 10 แสดงผลการทดสอบการทำงานของโอเปอเรเตอร์ **ONT_RELATED_NOT** ที่พัฒนาขึ้นเปรียบเทียบกับ คำสั่ง **“START WITH”** และ **“CONNECT BY”** ซึ่งแสดงในรูปแบบล็อกกราฟ (Log) เพื่อให้เห็นความต่างของเวลาที่ใช้ในการหาคำตอบของโอเปอเรเตอร์ได้ชัดเจน โดยแกนตั้งแสดงเวลาที่ซีพียูประมวลผลในหน่วยวินาทีรวมเวลาการอ่านข้อมูลจากฐานข้อมูล แกนนอนแสดงจำนวนระดับชั้นของออนโทโลยีที่ต้องการสืบค้น ตั้งแต่ระดับที่ 1 ถึง 13 จะเห็นได้ว่าโอเปอเรเตอร์ที่พัฒนาขึ้นมาใหม่ใช้เวลาในการสืบค้นน้อยกว่า คำสั่ง **“START WITH”** และ **“CONNECT BY”** ในทุก ๆ ระดับชั้น และเวลาจะแตกต่างกันมากโดยเฉพาะ เมื่อระดับชั้นที่พิจารณามีค่าต่ำสุด เนื่องจากโอเปอเรเตอร์จะต้องทำการหาสมาชิกทั้งหมดที่ไม่ได้เป็นทั้งโหนดบรรพบุรุษ และ โหนดลูกของโหนดที่พิจารณา โดยการหาสมาชิกที่เป็นทั้งโหนดบรรพบุรุษและ

โหนดถูกโดยการค้นหาแบบเชิงลึก แล้วพิจารณาหาสมาชิกที่ไม่ได้อยู่ในเซตของโหนดบรรพบุรุษ และโหนดลูกจึงทำให้ใช้เวลา นาน จึงสรุปได้ว่าโอเปอเรเตอร์ที่เสนอใหม่มีประสิทธิภาพดีกว่า คำสั่ง “START WITH” และ “CONNECT BY”

จากผลการทดลองแสดงในภาพที่ 7 ถึง 10 สรุปได้ว่าโอเปอเรเตอร์ **ONT_RELATED**, **ONT_RELATED_AND**, **ONT_RELATED_OR**, **ONT_RELATED_NOT** ที่พัฒนาขึ้น สามารถสืบค้นข้อมูลได้อย่างมีประสิทธิภาพ เนื่องจากโอเปอเรเตอร์ที่เสนอ ใช้เวลาในการประมวลผลลงที่ในลักษณะเชิงเส้น เมื่อจำนวนระดับชั้นในออนโทโลยีที่สืบค้นสูงขึ้นแตกต่างการประมวลผลโดยใช้คำสั่ง “START WITH” และ “CONNECT BY” ที่ใช้เวลาในการประมวลผลเพิ่มขึ้นเป็นเอ็กโพเนนเชียล

สำหรับระยะเวลาในการเตรียมข้อมูลออนโทโลยีในรูปแบบตารางทรานซิทีฟโคลสเซอร์ที่จัดเก็บในฐานข้อมูลเชิงสัมพันธ์ ใช้เวลาในการแปลงข้อมูลทั้งหมดจำนวน 6 ชั่วโมง ในจำนวนโหนด 20,080 โหนด

2. การทดสอบความครบถ้วนของผลการสืบค้นโดยการประมาณค่าการสูญหายของผลการสืบค้น

การทดสอบความครบถ้วนของผลการสืบค้นโดยการประมาณค่าการสูญหายของผลการสืบค้น ทำได้โดยกำหนดคำสั่งสืบค้นจำนวน 7 คำถาม แบ่งเป็นคำสั่งสืบค้นพื้นฐาน คือ Q1-Q4 และ Q5-Q7 คำสั่งสืบค้นร่วมกับลอจิกโอเปอเรเตอร์ นำคำสั่งสืบค้นเหล่านี้ไปสืบค้นในฐานข้อมูล **Gramene Protein** แล้วพิจารณาผลการสืบค้นเพื่อประมาณค่าการสูญหายของผลการสืบค้นเปรียบเทียบกับผลการสืบค้นจากคำสั่ง “START WITH” และ “CONNECT BY” แสดงตัวอย่างคำสั่งสืบค้นดัง ตารางที่ 7 และ แสดงผลการสืบค้นดังตารางที่ 8

ตารางที่ 7 ตัวอย่างคำถามสืบค้น

No.	Queries	Relation	Logical Operator
Q1	Find all proteins known to play role in “starch metabolism”	is-a	-
Q2	Find all proteins associated to the term “chromosome”	is-a	-

ตารางที่ 7(ต่อ)

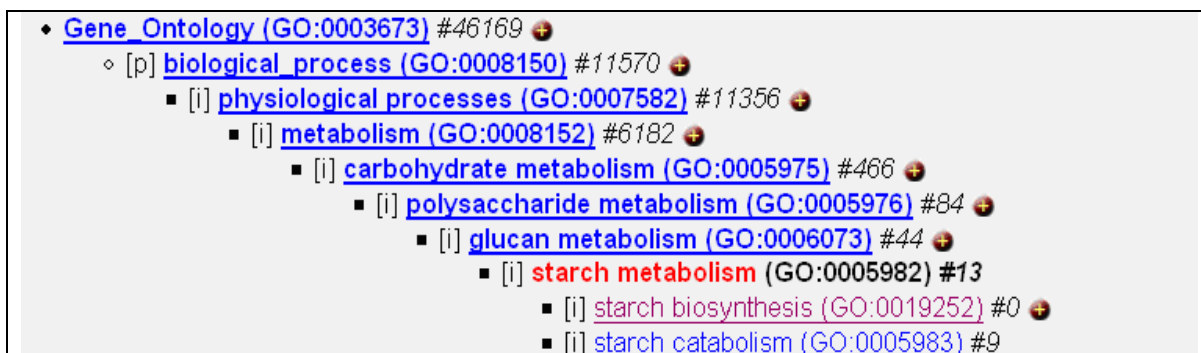
No.	Queries	Relation	Logical Operator
Q3	Find all proteins associated in “transcription” process	part-of	-
Q4	Find all proteins associated in “thylakoid”	part-of	-
Q5	Find all proteins associated in “intracellular organelle” and “intracellular non-membrane-bound organelle”	is-a	AND
Q6	Find all proteins associated in “NADH metabolism” or “NADP metabolism”	is-a	OR
Q7	Find all proteins associated in “photosystem I” or “thylakoid”	part-of	OR

ตารางที่ 8 ผลการสืบค้นข้อมูล

No	Loss of Information (%)	
	Our Operator	Start with - Connect By Clause
Q1	0%<loss< 21.74%	0%<loss< 21.74%
Q2	0%<loss< 1.96%	0%<loss< 1.96%
Q3	0%<loss< 4.22%	9.88%<loss<37.87%
Q4	0%<loss<6.88%	0%<loss<20%
Q5	0%<loss<4.00%	0%<loss<33.74%
Q6	0%<loss<33.33%	0%<loss<33.33%
Q7	0%<loss<54.05%	0%<loss<61.47%

จากตารางที่ 8 แสดงผลการสืบค้นข้อมูล โดยแสดงค่าการสูญหายของข้อมูล อธิบายได้ดังนี้ สำหรับคำถาม Q1 สืบค้นซึ่งระบุความสัมพันธ์แบบ **is-a** ผลการสืบค้นของโอเปอเรเตอร์ที่เสนอ กับคำสั่ง “**START WITH**” และ “**CONNECT BY**” มีค่าเท่ากันเนื่องจากความสัมพันธ์ที่เชื่อมโยงระหว่างคำสืบค้นจนถึงโหนดถูกทั้งหมดเป็นความสัมพันธ์เดียวกันทั้งหมด คำสั่ง “**START WITH**” และ “**CONNECT BY**” จึงสามารถสืบค้นคำสั่งได้อย่างต่อเนื่องตามความสัมพันธ์ที่ระบุ แสดง

บางส่วนของGO ออนโทโลยี ในส่วนคำสืบค้น “starch metabolism” ดังภาพที่ 11 โดยที่ [i] แทนความสัมพันธ์แบบ is-a และ [p] แทนความสัมพันธ์แบบ part-of



ภาพที่ 11 แสดงเส้นทางของโหนด “starch metabolism” ใน GO ออนโทโลยี

แสดงตัวอย่างการประมาณค่าการสูญหายของผลการสืบค้นของคำถาม Q1 พิจารณาโหนดลูกของ “starch metabolism” โดยมีความสัมพันธ์แบบ is-a แสดงโครงสร้างดังภาพที่ 11 โหนดลูกของ “starch metabolism” ได้แก่ starch metabolism, starch catabolism, starch biosynthesis การประมาณค่าการสูญหายทำได้โดยการพิจารณาผลการสืบค้นของโหนดลูกทั้งหมด เพราะโหนดลูกเหล่านี้เป็นกลุ่มคำที่ใช้ในการสืบค้นฐานข้อมูลแทนคำ “starch metabolism” แล้วคำนวณค่าการสูญหายของผลการสืบค้นโดยใช้สูตรคำนวณ ตามหัวข้อ 4.2 ในส่วนอุปกรณ์ และวิธีการ แสดงรายละเอียดดังนี้

$$\begin{aligned}
 |\text{Ext}(\text{starch metabolism})| &= 5, |\text{Ext}(\text{starch catabolism})| = 9, |\text{Ext}(\text{starch biosynthesis})| = 0 \\
 \text{Ext}(O\text{-Term})_{\text{low}} &= \max[|\text{Ext}(\text{starch metabolism})|, |\text{Ext}(\text{starch catabolism})|, |\text{Ext}(\text{starch biosynthesis})|] = 9 \\
 \text{Ext}(O\text{-Term})_{\text{high}} &= \sum[|\text{Ext}(\text{starch metabolism})|, |\text{Ext}(\text{starch catabolism})|, |\text{Ext}(\text{starch biosynthesis})|] = 14
 \end{aligned}$$

$$\text{Precision} = 1$$

$$\text{Recall}_{\text{low}} = \frac{|\text{Ext}(O\text{-Term})_{\text{low}}|}{|\text{Ext}(O\text{-Term})_{\text{low}}| + |\text{Ext}(U\text{-Term})|} = \frac{9}{9 + 5} = 0.642857$$

$$\text{Recall}_{\text{high}} = \frac{|Ext(O - Term)|_{\text{high}}}{\max[|Ext(O - Term)|_{\text{high}}, |Ext(U - Term)|]} = \frac{|14|}{[\max|14|, |5|]} = 1$$

$$\begin{aligned} \text{Loss}_{\text{low}} &= 1 - \frac{1}{a\left(\frac{1}{\text{precision.high}}\right) + (1-a)\left(\frac{1}{\text{Recall.high}}\right)} \\ &= 1 - \frac{1}{0.5\left(\frac{1}{1}\right) + (1-0.5)\left(\frac{1}{1}\right)} = 0 \end{aligned}$$

$$\begin{aligned} \text{Loss}_{\text{high}} &= 1 - \frac{1}{a\left(\frac{1}{\text{precision.low}}\right) + (1-a)\left(\frac{1}{\text{Recall.low}}\right)} \\ &= 1 - \frac{1}{0.5\left(\frac{1}{1}\right) + (1-0.5)\left(\frac{1}{0.642857}\right)} = 0.217391 \end{aligned}$$

เมื่อกำหนดค่า $\alpha = 0.5$ แสดงค่า **loss** คิดเป็นร้อยละ จึงสรุปได้ว่าค่าการสูญหายของคำถามสืบค้น Q1 คือ $0\% < \text{loss} < 21.74\%$

คำถามสืบค้น Q2 ที่ระบุความสัมพันธ์แบบ **part-of** ได้ผลการสืบค้นของโอเพอร์เรเตอร์ที่เสนอกับคำสั่ง “**START WITH**” และ “**CONNECT BY**” เช่นเดียวกันเนื่องจากความสัมพันธ์ที่เชื่อมโยงระหว่างคำสืบค้นจนถึงโหนดลูกทั้งหมดเป็นความสัมพันธ์เดียวกันทั้งหมด คำสั่ง “**START WITH**” และ “**CONNECT BY**” จึงสามารถสืบค้นคำสั่งได้อย่างต่อเนื่องตามความสัมพันธ์ที่ระบุ

คำถามสืบค้น Q3-Q4 ที่ระบุความสัมพันธ์แบบ **part-of** จะเห็นว่าค่าการสูญหายของผลการสืบค้นของคำสั่ง “**START WITH**” และ “**CONNECT BY**” มีค่ามากกว่าค่าการสูญหายของผลการสืบค้นของโอเพอร์เรเตอร์ที่เสนอ เนื่องจากความสัมพันธ์ที่เชื่อมโยงระหว่างคำสืบค้นจนถึงโหนดลูกทั้งหมดเป็นความสัมพันธ์ที่แตกต่างกัน คำสั่ง “**START WITH**” และ “**CONNECT BY**” ไม่สามารถสืบค้นตามความสัมพันธ์ได้อย่างต่อเนื่อง เนื่องจากอนุประโยค “**CONNECT BY**” ไม่สามารถไล่ตามโครงสร้างต้นไม้ของออนโทโลยีที่มีความสัมพันธ์ต่างจากความสัมพันธ์ที่กำหนดได้ทำให้ผลการสืบค้นไม่ครบถ้วน แตกต่างจากโอเพอร์เรเตอร์ที่นำเสนอที่สามารถอ้างอิง

ความสัมพันธ์ที่แตกต่างกันระหว่างโหนดที่พิจารณากับโหนดลูกทั้งหมดได้ แสดง GO ออนโทโลยี ส่วนคำสืบค้น “thylakoid” ดังภาพที่ 12 โดยที่ [i] แทนความสัมพันธ์แบบ is-a และ [p] แทนความสัมพันธ์แบบ part-of

```

+ ⓘ GO:0009579 : thylakoid [11]
  - ⓘ GO:0044436 : thylakoid part [10]
    - ⓘ GO:0009521 : photosystem [5]
      - ⓘ GO:0009522 : photosystem I [2]
        - ⓘ GO:0030093 : chloroplast photosystem I [0]
        - ⓘ GO:0009782 : photosystem I antenna complex [0]
        - ⓘ GO:0009538 : photosystem I reaction center [1]
        - ⓘ GO:0030094 : plasma membrane-derived photosystem I [0]
      + ⓘ GO:0009523 : photosystem II [2]
      - ⓘ GO:0030090 : reaction center (sensu Proteobacteria) [0]
    - ⓘ GO:0009502 : photosynthetic electron transport chain [1]
      - ⓘ GO:0009512 : cytochrome b6f complex [0]
    - ⓘ GO:0042651 : thylakoid membrane [4]
      - ⓘ GO:0031361 : integral to thylakoid membrane [0]
      - ⓘ GO:0031360 : intrinsic to thylakoid membrane [0]
      + ⓘ GO:0031675 : NADH dehydrogenase complex (plastoquinone) [0]
      + ⓘ GO:0031676 : plasma membrane-derived thylakoid membrane [0]
      + ⓘ GO:0055035 : plastid thylakoid membrane [4]
      - ⓘ GO:0042650 : prothylakoid membrane [0]
  
```

ภาพที่ 12 แสดงเส้นทางของโหนด “thylakoid” ใน GO ออนโทโลยี

คำถามสืบค้น Q6 สืบค้นโดย OR โอเปอเรเตอร์โดยระบุความสัมพันธ์แบบ is-a ผลการสืบค้นของโอเปอเรเตอร์ที่เสนอกับคำสั่ง “START WITH” และ “CONNECT BY” มีค่าเท่ากันเนื่องจากความสัมพันธ์ที่เชื่อมโยงระหว่างคำสืบค้นจนถึงโหนดลูกทั้งหมดเป็นความสัมพันธ์เดียวกันทั้งหมด คำสั่ง “START WITH” และ “CONNECT BY” จึงสามารถสืบค้นคำสั่งได้อย่างต่อเนื่องตามความสัมพันธ์ที่ระบุ แสดง GO ออนโทโลยี ดังภาพที่ 13 โดยที่ [i] แทนความสัมพันธ์แบบ is-a

```

    □ ● GO:0019362 : pyridine nucleotide metabolic
    process [399]
    □ ● GO:0006769 : nicotinamide metabolic
    process [340]
    □ ● GO:0006734 : NADH metabolic process [41]
    □ ● GO:0006116 : NADH oxidation [16]
    □ ● GO:0006735 : NADH regeneration [2]
    □ ● GO:0006738 : nicotinamide riboside catabolic
    process [0]
    □ ● GO:0006739 : NADP metabolic process [222]
    □ ● GO:0006741 : NADP biosynthetic process [8]
    □ ● GO:0006742 : NADP catabolic process [2]
    □ ● GO:0006740 : NADPH regeneration [204]
    □ ● GO:0006098 : pentose-phosphate shunt [200]
    □ ● GO:0009780 : photosynthetic NADP+
    reduction [1]
  
```

ภาพที่ 13 แสดงเส้นทางของโหนดสืบค้น “NADH metabolism” และ “NADP metabolism” ใน GO ออนโทโลยี

คำถามสืบค้น Q5 ที่สืบค้นร่วมกับ AND โอเปอเรเตอร์โดยระบุความสัมพันธ์แบบ **is-a** และคำถามสืบค้น Q7 ที่สืบค้นร่วมกับ OR โอเปอเรเตอร์โดยระบุความสัมพันธ์แบบ **part-of** จะเห็นว่าค่าการสูญหายของผลการสืบค้นของคำสั่ง “START WITH” และ “CONNECT BY” มีค่ามากกว่าค่าการสูญหายของผลการสืบค้นของโอเปอเรเตอร์ที่เสนอ เนื่องจากความสัมพันธ์ที่เชื่อมโยงระหว่างคำสืบค้นจนถึงโหนดลูกทั้งหมดเป็นความสัมพันธ์ที่แตกต่างกัน คำสั่ง “START WITH” และ “CONNECT BY” ไม่สามารถสืบค้นตามความสัมพันธ์ได้อย่างต่อเนื่อง เนื่องจากอนุประโยค “CONNECT BY” ไม่สามารถไล่ตามโครงสร้างต้นไม้ของออนโทโลยีที่มีความสัมพันธ์ต่างจากความสัมพันธ์ที่กำหนดได้ทำให้ผลการสืบค้นไม่ครบถ้วนแตกต่างจากโอเปอเรเตอร์ที่นำเสนอที่สามารถอ้างอิงความสัมพันธ์ที่แตกต่างกันระหว่างโหนดที่พิจารณากับโหนดลูกทั้งหมดได้

จากการพิจารณา การประมาณค่าการสูญหายของผลการสืบค้นของโอเปอเรเตอร์ที่เสนอเปรียบเทียบกับผลการสืบค้นจากคำสั่ง “START WITH” และ “CONNECT BY” สรุปได้ว่าโอเปอเรเตอร์ที่เสนอให้ผลการสืบค้นที่ครบถ้วนมากกว่าการสืบค้นโดยใช้คำสั่ง “START WITH” และ “CONNECT BY” เมื่อความสัมพันธ์ระหว่างคำสืบค้นและโหนดลูกทั้งหมดหลากหลาย คือมีมากกว่า 1 ความสัมพันธ์

สรุปและข้อเสนอแนะ

สรุป

การสืบค้นข้อมูลในฐานข้อมูลเชิงสัมพันธ์โดยประยุกต์ใช้ออนโทโลยี เป็นแนวทางเพิ่มประสิทธิภาพการสืบค้นข้อมูลในฐานข้อมูลเชิงสัมพันธ์ โดยประยุกต์ใช้ออนโทโลยีเพื่อแปลงคำสั่งสืบค้นของผู้ใช้ให้อยู่ในรูปแบบคำสั่งสืบค้นเชิงความหมาย สิ่งที่ต้องคำนึงถึงในการประยุกต์ใช้ออนโทโลยี เพื่อการสืบค้นในฐานข้อมูลเชิงสัมพันธ์ คือ แนวทางการจัดเก็บและการจัดการออนโทโลยี รวมถึง การสืบค้นออนโทโลยีเพื่อให้การสืบค้นข้อมูลเป็นไปอย่างมีประสิทธิภาพ

งานวิจัยได้เสนอแนวทางในการจัดเก็บออนโทโลยีที่มีหลายความสัมพันธ์ในฐานข้อมูลเชิงสัมพันธ์ และโอเปอเรเตอร์สืบค้นข้อมูลเชิงความหมายที่ประยุกต์ใช้ออนโทโลยี โดยเก็บออนโทโลยีในรูปแบบความสัมพันธ์เชิงวัตถุ ที่อธิบายว่าแต่ละโหนดในออนโทโลยีมีความสัมพันธ์กันอย่างไร โดยการไล่ตามโครงสร้างต้นไม้ของออนโทโลยี และอ้างอิงความสัมพันธ์ระหว่างโหนดใดๆ กับโหนดลูกทั้งหมด ซึ่งเส้นทางระหว่างโหนดนั้นอาจมีหลายเส้นทาง และประกอบด้วยหลายความสัมพันธ์ จึงต้องเลือกเส้นทางที่มีความเหมือนเชิงความหมายมากที่สุด เพื่อลดการสูญหายของผลการสืบค้น เมื่อนำออนโทโลยีคอนเซพต์ที่จัดเก็บนี้แทนคำสั่งสืบค้นของผู้ใช้ให้อยู่ในรูปแบบคำสั่งสืบค้นเชิงความหมาย ซึ่งแปลงโดย **ONT_RELATED** โอเปอเรเตอร์ ที่จะสืบค้นหาออนโทโลยีที่มีความสัมพันธ์สอดคล้องกับคำสั่งสืบค้นของผู้ใช้ ซึ่งการจัดเก็บออนโทโลยีในรูปแบบที่เสนอทำให้การสืบค้นออนโทโลยีทำได้ง่าย โดยการใช้คำสั่ง **SQL** ที่ระบุหมายเลขประจำโหนดที่ต้องการสืบค้นเพียงเท่านั้นจะสามารถสืบค้นหาโหนดลูกทั้งหมดของโหนดนั้นได้ โดยไม่ต้องทำการสืบค้นแบบวนซ้ำตามโครงสร้างลำดับชั้นของออนโทโลยี นอกจากนี้ **ONT_RELATED** โอเปอเรเตอร์ยังสามารถรองรับคำสั่งสืบค้นแบบซับซ้อนได้ด้วยการสืบค้นร่วมกับลอจิกโอเปอเรเตอร์ **AND, OR, NOT**

การวัดประสิทธิภาพของโอเปอเรเตอร์ที่เสนอในงานวิจัย พิจารณาความเร็วในการประมวลผลสืบค้น และพิจารณาความครบถ้วนของผลการสืบค้น โดยเปรียบเทียบกับคำสั่ง **“START WITH”** และ **“CONNECT BY”** จากผลการทดลองสรุปได้ว่า **ONT_RELATED** โอเปอเรเตอร์ใช้เวลาในการสืบค้นครั้งที่ เมื่อระดับชั้นในออนโทโลยีเพิ่มขึ้น ในขณะที่คำสั่ง

“START WITH” และ “CONNECT BY” ใช้เวลาเพิ่มขึ้นเป็นเอ็กโพเนนเชียล ดังนั้นโอเปอเรเตอร์สืบค้นที่เสนอในงานวิจัยจึงสามารถสืบค้นข้อมูลได้อย่างมีประสิทธิภาพมากกว่าคำสั่ง **“START WITH”** และ **“CONNECT BY”**

และเมื่อพิจารณาถึงความครบถ้วนของผลการสืบค้น **ONT_RELATED** โอเปอเรเตอร์ที่เสนอให้ผลการสืบค้นที่ครบถ้วนกว่า การใช้คำสั่ง **“START WITH”** และ **“CONNECT BY”** เมื่อความสัมพันธ์ระหว่างคำที่ต้องการสืบค้นกับโหนดลูกในออนโทโลยีมีความสัมพันธ์ที่หลากหลาย โดยพิจารณาจากการประมาณค่าการสูญหายของผลการสืบค้น เนื่องจากงานวิจัยนี้จะพิจารณาถึงการอ้างอิงความสัมพันธ์ระหว่างโหนดในออนโทโลยีที่มีหลายความสัมพันธ์ โดยกำหนดความสัมพันธ์ตามค่าความเหมือนเชิงความหมาย และค่าความสำคัญที่กำหนดโดยผู้เชี่ยวชาญ นอกจากนี้ประโยชน์ที่ได้รับจากการจัดเก็บออนโทโลยีในฐานข้อมูลเชิงสัมพันธ์ คือ การจัดการออนโทโลยี เช่น แก้ไข ลบ และการเข้าถึงออนโทโลยีสามารถทำได้ง่ายโดยการใช้คำสั่ง **SQL**

ข้อเสนอแนะ

การจัดเก็บออนโทโลยีที่นำเสนอในงานวิจัยนี้ ยังมีข้อจำกัดในเรื่องเนื้อที่ในการจัดเก็บข้อมูลโดยตารางในฐานข้อมูลเชิงสัมพันธ์ที่จัดเก็บออนโทโลยีตามแนวทางที่เสนอ จะใช้พื้นที่มากกว่าการจัดเก็บออนโทโลยีที่พิจารณาเฉพาะโหนดใกล้ชิดเท่านั้น ซึ่งถ้าข้อมูลมีขนาดมากขึ้นจะทำให้เปลืองพื้นที่ในการจัดเก็บ จึงต้องเปรียบเทียบระหว่างประสิทธิภาพในการสืบค้นที่ได้ กับพื้นที่ในการจัดเก็บข้อมูล นอกจากนี้จะต้องเสียเวลาในการจัดเตรียมข้อมูลเพื่อจัดเก็บข้อมูลตารางในรูปแบบที่กำหนด แต่เวลาที่เสียไปขั้นตอนนี้จะเกิดขึ้นเพียงครั้งเดียวเท่านั้น

ในส่วนการสืบค้นร่วมกับลอจิกโอเปอเรเตอร์ โอเปอเรเตอร์สืบค้นจะต้องแปลงออนโทโลยีที่จัดเก็บในฐานข้อมูล ให้อยู่ในรูปบิตสตริง ซึ่งจะทำให้เวลาในการประมวลผลสืบค้นเพิ่มขึ้น จึงควรจัดเตรียมบิตสตริงและเก็บในฐานข้อมูล เช่นเดียวกันกับการจัดเก็บออนโทโลยี

งานวิจัยนี้ไม่ได้กล่าวถึงกรณีที่มีการเปลี่ยนแปลงออนโทโลยี ซึ่งได้แก่ การเพิ่มโหนด การลบโหนด และการเปลี่ยนแปลงความสัมพันธ์ระหว่างโหนดในออนโทโลยี ตารางจัดเก็บออนโทโลยีที่จัดเตรียมไว้จะต้องมีการเปลี่ยนแปลงหรือไม่อย่างไร ซึ่งจะต้องมีการพัฒนาในส่วนนี้ต่อไป เพื่อความสะดวกในการใช้งานมากยิ่งขึ้น