ระบบค้นคืนข้อมูลสารสนเทศโดยใช้ภาษาธรรมชาติ

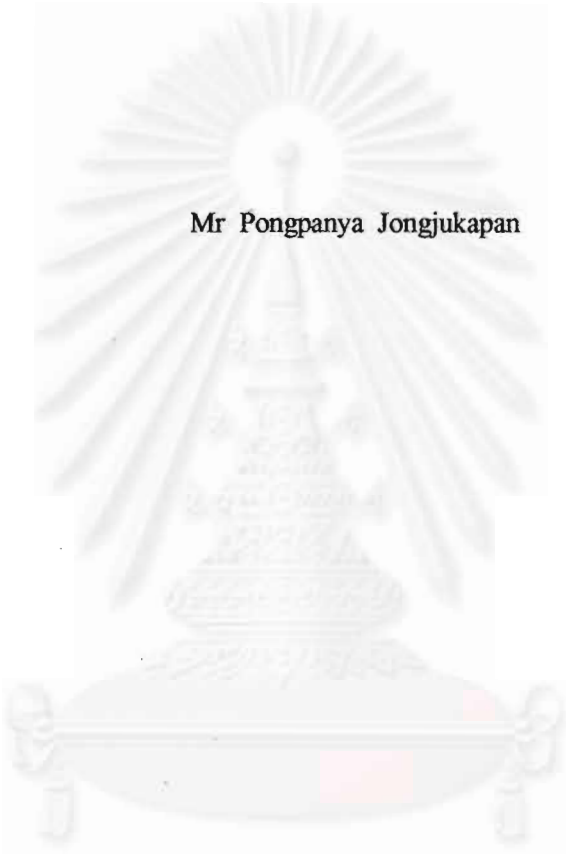
นาย พงษ์ปัญญา จงจักรพันธ์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2542

ISBN 974-333-675-3

AN INFORMATION RETRIEVAL SYSTEM USING NATURAL LANGUAGE

Mr Pongpanya Jongjukapan



A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Science in Computer Science

Department of Computer Engineering

Faculty of Engineering

Chulalongkorn University

Academic Year 1999

ISBN 974-333-675-3

หัวข้อวิทยานิพนธ์	ระบบค้นคืนข้อมูลสารสนเทศโดยใช้ภาษาธรรมชาติ
โดย	นาย พงษ์ปัญญา จงจักรพันธ์
ภาควิชา	วิศวกรรมคอมพิวเตอร์
อาจารย์ที่ปรึกษา	อาจารย์ ดร. บุญเสริม กิจศิริกุล
อาจารย์ที่ปรึกษาร่วม	ผู้ช่วยศาสตราจารย์ ดร. ภัทรสินี ภัทรโกศล

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

..... คณบดีคณะวิศวกรรมศาสตร์

(รองศาสตราจารย์ ดร. ธัชชัย สุมิตร)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ

(ผู้ช่วยศาสตราจารย์ วันพร ปิ้นเก่า)

..... อาจารย์ที่ปรึกษา

(อาจารย์ ดร. บุญเสริม กิจศิริกุล)

..... อาจารย์ที่ปรึกษาร่วม

(ผู้ช่วยศาสตราจารย์ ดร. ภัทรสินี ภัทรโกศล)

..... กรรมการ

(อาจารย์ ดร. ยรรยง เต็งอำนวย)

..... กรรมการ

(ผู้ช่วยศาสตราจารย์ ดร. สาธิต วงศ์ประทีป)

พงษ์ปัญญา จงจักรพันธ์ : ระบบค้นคืนสารสนเทศน์โดยใช้ภาษาธรรมชาติ (AN INFORMATION RETRIEVAL SYSTEM USING NATURAL LANGUAGE) อ.ที่ปรึกษา : อ. ดร. บุญเสริม กิจศิริกุล, อ. ที่ปรึกษาร่วม : ผศ. ดร. ภัทรสินี ภัทรโกศล ; 86 หน้า. ISBN 974-333-675-3.

การวิจัยนี้มีวัตถุประสงค์เพื่อพัฒนาระบบค้นคืนสารสนเทศน์โดยใช้ภาษาธรรมชาติ ซึ่งเป็นการพัฒนาให้ผู้ใช้ระบบสามารถใช้ภาษาธรรมชาติเป็นภาษาสอบถามในการติดต่อกับระบบหลายฐานข้อมูลบนระบบเดี่ยวผ่านทางส่วนที่ติดต่อกับผู้ใช้ โดยระบบจะทำการวิเคราะห์และแปลงคำร้องขอที่ผู้ใช้กำหนดด้วยวิธีการเก็บรวบรวมข้อมูลแบบ non-deterministic top-down โดยใช้รูปแบบไวยากรณ์เป็นตัวช่วยในการหาความหมายร่วมกับพจนานุกรมฐานความรู้ที่ประกอบไปด้วยคำข้อมูลที่มีอยู่ในระบบ ชนิดของไวยากรณ์ การอ้างถึงองค์ประกอบในฐานข้อมูล และความสัมพันธ์ที่ได้จากโมเดลข้อมูลของระบบฐานข้อมูล เมื่อระบบทำการแปลงคำร้องขอจากภาษาธรรมชาติเป็นภาษาสอบถามเชิงโครงสร้างที่ต้องการเรียบร้อย ระบบจะทำการตรวจสอบความสมบูรณ์ตามหลักไวยากรณ์ของภาษาสอบถามที่แปลงได้รวมทั้งความเป็นไปได้ของผลลัพธ์ภายใต้ความสัมพันธ์ของข้อมูลที่ระบุว่ามีปรากฏอยู่จริงในฐานข้อมูลโดยใช้เมตริกซ์ความสัมพันธ์องค์ประกอบของระบบฐานข้อมูล คุณสมบัติของภาษาธรรมชาติที่รองรับมีอิสระในการที่คำร้องขอของผู้ใช้สามารถอ้างอิงถึงฐานข้อมูลที่อยู่ในระบบฐานข้อมูลนั้นได้มากกว่า 1 ฐานข้อมูลในคำร้องขอหนึ่งๆ โดยระบบใช้วิธีในการแยกคำร้องขอไปยังแต่ละฐานข้อมูลแล้วนำผลลัพธ์ที่ได้จากฐานข้อมูลที่แตกต่างกันมาผสานเป็นผลลัพธ์ที่ผู้ใช้ระบบต้องการ ผู้วิจัยได้พัฒนาต้นแบบของระบบค้นคืนสารสนเทศน์โดยใช้ภาษาธรรมชาติบนระบบฐานข้อมูล mSQL ซึ่งมีจุดเด่นในด้านการใช้เนื้อที่น้อย และมีประสิทธิภาพในการเข้าถึงข้อมูลสูง

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต พงษ์ปัญญา จงจักรพันธ์

สาขาวิชา วิทยาศาสตร์คอมพิวเตอร์ ลายมือชื่ออาจารย์ที่ปรึกษา

ปีการศึกษา 2542 ลายมือชื่ออาจารย์ที่ปรึกษาร่วม

3971097221 : MAJOR COMPUTER SCIENCE
KEY WORD: USER INTERFACE / NATURAL LANGUAGE / QUERY PROCESSING / DATABASE MANAGEMENT SYSTEM
PONGPANYA JONGJUKAPAN : AN INFORMATION RETRIEVAL SYSTEM USING NATURAL LANGUAGE. THESIS ADVISOR : BOONSERM KIJSIRIKUL, Ph.D. THESIS CO-ADVISOR : ASSIST. PROF. PATTARASINEE BHATTARAKOSOL, Ph.D. 86 pp. ISBN 974-333-675-3.

This research is focused in the information retrieval system (IRS) using natural language to retrieve data stored in a homogeneous multi-database system through the user interface. The developed of IRS will analyze and interpret a user's query using the non-deterministic top-down approach. This approach uses the set of grammar and the knowledge-based dictionary to determine the meaning of the query. The knowledge-based dictionary consists of all words in the system, types of grammar, references of database components, and entity-relationship model of the database. After the interpretation of the query, a SQL statement will be obtained. Additionally, the completeness of the SQL statement under the syntactic rules of the language, including the potential of successive request will be validated based on the relations of existing data in the domain system. The validation process can be performed using a relational matrix of database components. The characteristics of the natural language under this environment will allow user to retrieve data from the database system(s) at a time. The query will be separated into sub-queries belong to each database system and the returned results will be merged and presents to the user as requested. The researcher had developed a prototype of IRS on mSQL that requires minimal space but provides high performance of data access.

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต
สาขาวิชา วิทยาศาสตร์คอมพิวเตอร์ ลายมือชื่ออาจารย์ที่ปรึกษา
ปีการศึกษา 2542 ลายมือชื่ออาจารย์ที่ปรึกษาร่วม

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ผู้วิจัยต้องขอกราบขอบพระคุณ อาจารย์ ดร.บุญเสริม กิจศิริกุล และผู้ช่วยศาสตราจารย์ ดร.ภัทรสินี ภัทรโกศล เป็นอย่างยิ่งที่ท่านได้ให้คำแนะนำ ข้อคิดเห็นต่าง ๆ ตลอดจนแนวทางในการทำวิจัยด้วยดีตลอดมา และตรวจแก้วิทยานิพนธ์ฉบับนี้อย่างละเอียด ซึ่งเป็นสิ่งผลักดันให้วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงด้วยดี

ขอกราบขอบพระคุณคณะกรรมการทุกท่านที่ได้ให้คำแนะนำ เพื่อแก้ไขรูปแบบและเนื้อหาวิทยานิพนธ์ฉบับนี้จนเสร็จสมบูรณ์

สุดท้ายนี้ผู้วิจัยขอกราบขอบพระคุณบิดา มารดา ที่ให้การสนับสนุนและกำลังใจเสมอมาจนกระทั่งสำเร็จการศึกษา ขอขอบคุณพี่ ๆ และเพื่อน ๆ ที่ได้ให้คำปรึกษาและความช่วยเหลือในด้านต่าง ๆ ซึ่งทำให้การทำงานวิจัยเป็นไปอย่างราบรื่น

สารบัญ

หน้า

สารบัญตาราง

หน้า

สารบัญภาพ

หน้า

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

Hughes Technologies ได้พัฒนาระบบจัดการฐานข้อมูล Mini SQL [2] (ต่อไปนี้จะเรียกว่า mSQL) ชนิดใหม่ในปี 1995 และในปัจจุบันการพัฒนานี้ยังดำเนินอย่างต่อเนื่องเป็นที่รู้จักอย่างแพร่หลายในกลุ่มผู้ใช้ฐานข้อมูล แต่เครื่องมือซอฟต์แวร์ที่พัฒนาขึ้นยังมีข้อจำกัดของภาษาสอบถามเชิงโครงสร้าง คือ ไม่สามารถกำหนดเงื่อนไขแบบซ่าย (nested) ได้ ส่วนในด้านของผู้ใช้ถ้าต้องการค้นคืนสารสนเทศที่เก็บอยู่ในฐานข้อมูล ผู้ใช้ต้องมีความสามารถที่ใช้ภาษาสอบถามเชิงโครงสร้างได้ ทำให้ผู้ใช้อาจต้องเรียนรู้ภาษาดังกล่าวสำหรับการใช้งานเพื่อค้นคืนข้อมูลที่ต้องการจากระบบฐานข้อมูล

ดังนั้นเพื่อเป็นการเพิ่มประสิทธิภาพให้กับระบบดังกล่าว และเพื่อให้ผู้ใช้สามารถทำการค้นคืนข้อมูลจากฐานข้อมูลได้อย่างสะดวกยิ่งขึ้น วิทยานิพนธ์นี้จึงจะทำการศึกษาทดลอง และพัฒนาระบบในส่วนที่ใช้ติดต่อกับผู้ใช้บริการ(user interface) ให้มีประสิทธิภาพสูง ยิ่งขึ้นโดยการยินยอมให้ผู้ใช้ติดต่อกับระบบฐานข้อมูลต่างๆ ของ mSQL ได้โดยใช้ภาษาธรรมชาติ(ภาษาอังกฤษ) เพื่อมิให้เกิดความยุ่งยากและ**ซับซ้อนในการใช้คำสั่ง** หรือสัญรูปที่กำหนดมาให้เฉพาะโปรแกรม ดังนั้นหากมีการเปลี่ยนระบบใหม่ผู้ใช้จะไม่ต้องทำการเรียนรู้คำสั่งในรูปแบบอื่นอีก เนื่องจากชุดคำสั่งของระบบใหม่อาจแตกต่างจากเดิมแม้จะเป็นภาษาโครงสร้างเหมือนกัน แต่ถ้าระบบการเรียกใช้ข้อมูลในส่วนที่ติดต่อกับผู้ใช้บริการยินยอมให้ผู้ใช้ใช้ภาษาธรรมชาติเพื่อทำการอ่านข้อมูลได้แล้ว การเรียกข้อมูลภายใต้ฐานข้อมูลต่างชนิดกันก็สามารถกระทำได้โดยง่าย โดยเพียงทำการปรับปรุงตัวแปลภาษาเพื่อรองรับภาษาธรรมชาติเหล่านั้นเท่านั้น นอกจากนี้แล้ววิทยานิพนธ์นี้จะทำการเพิ่มประสิทธิภาพในการเรียกใช้ข้อมูล ด้วยการให้ผู้ขอใช้บริการเรียกใช้ข้อมูลจากหลายฐานข้อมูลพร้อมๆ กันได้ในขณะหนึ่งๆ บนระบบเดียวกัน ซึ่งจะทำให้ระบบที่พัฒนาขึ้นมีประสิทธิภาพในการจำแนกแหล่งข้อมูลที่ถูกผู้ใช้ร้องขอ อีกทั้งยังเป็นการยินยอมให้ผู้ใช้ในหลายๆ สาขาสามารถใช้ข้อมูลร่วมกันได้โดยสะดวกในการเรียกใช้ข้อมูลเพียงคำถามเดียว ภายใต้สมมติฐานที่ว่าจะมีความสัมพันธ์ระหว่างข้อมูลที่อยู่บนฐานข้อมูลต่างๆ กันเสมอ ในอนาคตการพัฒนาให้เป็นระบบจัดการฐานข้อมูลแบบกระจายจะกระทำได้ง่าย พร้อมกับการขยายความสามารถของการกำหนดเงื่อนไขเพื่อรองรับการกำหนดเงื่อนไขแบบซ่าย ทำให้สามารถค้นคืนข้อมูลได้ตรงกับความต้องการของผู้ใช้มากขึ้น

1.2 วัตถุประสงค์

1) เพื่อพัฒนาระบบการเรียกใช้ข้อมูลบนฐานข้อมูลต่างๆ โดยการใช้ส่วนที่ติดต่อกับผู้ใช้ที่สามารถรองรับความต้องการในการสอบถาม หรือเรียกใช้ข้อมูลเชิงโครงสร้างของ mSQL ในลักษณะของภาษาธรรมชาติได้

2) เพื่อพัฒนาระบบการแจงส่วน วิเคราะห์ และแปลงประโยคของภาษาธรรมชาติให้อยู่ในรูปของภาษาสอบถามเชิงโครงสร้างมาตรฐานที่สามารถเรียกข้อมูลจากฐานข้อมูลต่างๆ ได้ถูกต้อง

3) เพื่อพัฒนาระบบการควบคุมการเรียกใช้ข้อมูล และการแสดงผลข้อมูลเมื่อผู้ใช้เรียกใช้ข้อมูลจากหลายฐานข้อมูลบนระบบเดียวกันได้พร้อมๆ กัน

1.3 ขอบเขตการวิจัย

1) ผู้ขอใช้บริการถูกจำเพาะการทำงานเพื่อการเรียกดูข้อมูลเท่านั้น

2) การเข้าถึงข้อมูลของผู้ใช้สามารถเรียกใช้ได้มากกว่า 1 ฐานข้อมูล ในขณะหนึ่งได้

3) ประโยคภาษาธรรมชาติที่ใช้บนภาษาอังกฤษไม่มีความซับซ้อนและมีเงื่อนไขแบบง่ายไม่เกินกว่า 3 ระดับ

4) สามารถรองรับการสอบถามข้อมูลเชิงโครงสร้าง ที่มีให้ในระบบฐานข้อมูลของ mSQL

5) ภาษาธรรมชาติที่สามารถรองรับได้ ไม่รวมถึงการใช้ในลักษณะ คำย่อ คำแสลง หรือลักษณะภาษาที่ได้กำหนดขึ้นใช้เฉพาะกลุ่ม

6) การกำหนด ระบุ หรือ อ้างถึงเนื้อความของข้อมูลที่ผู้ใช้ต้องการนั้น จะต้องมีความชัดเจนกระชับ และไม่มีคำฟุ่มเฟือย

7) การออกแบบการตั้งชื่อที่ใช้ในฐานข้อมูลต้องไม่ซ้ำซ้อน

8) ระบบทำงานบนระบบปฏิบัติการยูนิกซ์

1.4 ลำดับขั้นตอน

1) ศึกษาการทำงานของระบบฐานข้อมูล mSQL
2) ศึกษาการทำงานของตัวแจงส่วน
3) สร้างและทดสอบส่วนที่ติดต่อกับผู้ใช้ในการรับคำสั่งและแสดงผล
4) สร้างและทดสอบส่วนที่แปลงคำสั่ง
5) ทำการทดสอบระบบรวมทั้งหมด
6) สรุปและวิเคราะห์ผลการทดสอบ

1.5 ประโยชน์ที่จะได้

1) ผู้ใช้สามารถทำงานที่เกี่ยวข้องกับฐานข้อมูลบนระบบปฏิบัติการยูนิกซ์ได้ง่าย และสะดวกขึ้นโดยไม่จำเป็นต้องมีความรู้ทางด้านการพัฒนาโปรแกรมประยุกต์หรือ ภาษาโครงสร้างที่ฐานข้อมูลนั้นใช้ และสามารถสื่อความหมายได้ตรงกับความต้องการของตัวผู้ใช้เองโดยไม่เกิดความสับสน

2) ระบบที่พัฒนาขึ้นสามารถนำมาใช้งานได้เสมือนซอฟต์แวร์สำหรับจัดการฐานข้อมูลทั่วไปที่มีอยู่ในท้องตลาด แต่ต้องการทรัพยากรบนเครื่องที่น้อยกว่าและราคาไม่สูงจนเกินไป

บทที่ 2

ทฤษฎีและแนวความคิดที่ใช้ในงานวิจัย

ในบทนี้จะกล่าวถึงทฤษฎีและแนวคิดต่าง ๆ ที่ใช้ในการวิจัย ได้แก่ ลักษณะของโครงสร้างข้อมูลแบบทู-ทรัย, Directed Hypergraph, mSQL ระบบผู้ใช้บริการ/ผู้ให้บริการ (Client/Server) และ Tcl/Tk เพื่อใช้เป็นความรู้ในการออกแบบและพัฒนาเครื่องมือให้ทำงานได้อย่างถูกต้องและมีประสิทธิภาพ รวมถึงงานวิจัยของ Nabil R. Adam และ Aryya Gangopadhyay คือ A Form-Based Natural Language Front-End to a CIM Database [9] เป็นวิธีที่ได้เสนอแนวความคิดในการพัฒนาส่วนที่ติดต่อกับผู้ใช้ โดยการผสมกันระหว่างการใช้ฟอร์มของภาษาเชิงโครงสร้าง (SQL Form) กับส่วนประมวลผลภาษาธรรมชาติเพื่อติดต่อกับระบบจัดการฐานข้อมูล ORACLE ระบบสามารถทำงานโดยรับข้อมูลผู้ใช้เพื่อระบุฟอร์มที่ครอบคลุมความต้องการ หรือประมวลคำสั่งจากผู้ใช้ที่ใส่เข้ามาเติมลงในฟอร์ม จากนั้นทำการแปลงเป็นภาษาเชิงโครงสร้างแล้วติดต่อไปยังฐานข้อมูลเพื่อให้ได้ข้อมูลตามที่ผู้ใช้ต้องการ ข้อจำกัดของระบบดังกล่าวคือ รูปแบบไวยากรณ์ของคำสั่งที่รองรับได้ขึ้นอยู่กับฟอร์ม และผู้ใช้ต้องใส่ดัชนีหลัก (unique index) เพื่อบ่งชี้สำหรับข้อมูลการทำงานกับฟอร์มนั้นด้วย

2.1 โครงสร้างข้อมูลแบบทู-ทรัย

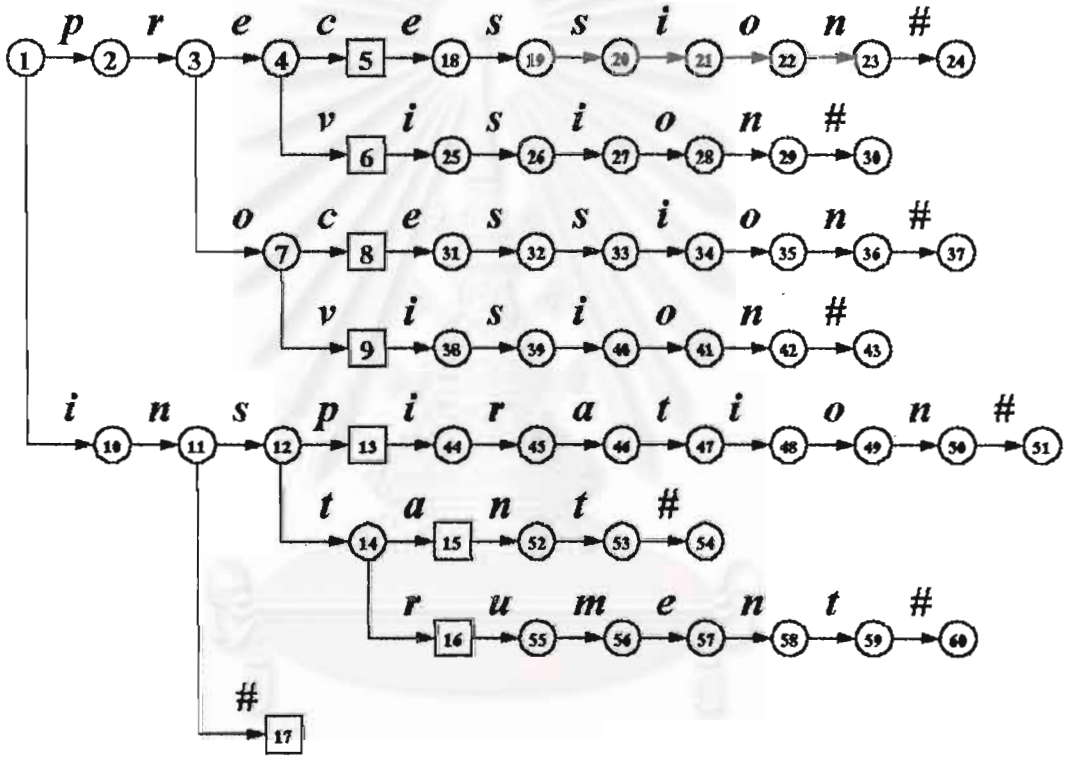
โครงสร้างข้อมูลแบบทู-ทรัย (two-trie structure) ได้พัฒนาขึ้นจากโครงสร้างข้อมูลแบบทรัย (trie structure) ลักษณะของโครงสร้างข้อมูลแบบทรัยเหมาะกับข้อมูลที่ประกอบด้วยตัวอักษรเรียงต่อกันไปจึงทำให้สามารถใช้ได้ทั้งข้อความภาษาไทยและภาษาอังกฤษ และเวลาที่ใช้ในการค้นหาข้อมูลน้อยจึงทำให้การค้นหาข้อมูลมีประสิทธิภาพดี [5] จึงทำให้โครงสร้างข้อมูลแบบทรัยได้นำไปใช้ในงานด้านต่าง ๆ เช่น การสร้างพจนานุกรมสำหรับการประมวลผลภาษาธรรมชาติ และการค้นหาคำสงวน (reserved words) ของตัวแปลภาษา (compiler) เป็นต้น โครงสร้างข้อมูลแบบทรัยยังเหมาะกับการค้นหาแบบส่วนเติมหน้า (prefix searching) อีกด้วยเนื่องจากโครงสร้างข้อมูลแบบทรัยจะเก็บส่วนเติมหน้า (prefix) ของแต่ละดัชนีที่เหมือนกันไว้เพียงที่เดียว

โครงสร้างข้อมูลแบบทรัยประกอบด้วยสตริง (string) 2 ส่วน คือ สตริงส่วนหน้า (front string) และสตริงส่วนหลัง (rear string) โดยที่สเตท (state) หรือโหนด (node) และทรานสิชั่น (transition) ที่มีลักษณะเป็นตัวอักษรของสตริงส่วนหน้าจะใช้ร่วมกันระหว่างดัชนีต่าง ๆ แต่โหนดและทรานสิชั่นของสตริงส่วนหลังจะไม่ใช้ร่วมกันโดยมีเซพพาเรทโหนด (separate node) เป็น

โหนดที่ใช้แยกความแตกต่างของแต่ละดัชนีที่เก็บในโครงสร้างข้อมูลแบบทรัย ซึ่งคั่นระหว่างสตริงส่วนหน้าและสตริงส่วนหลัง และในสตริงส่วนหลังจะใช้เครื่องหมาย # เป็นโหนดสิ้นสุด (terminal node) ของแต่ละดัชนี

ตัวอย่างข้อมูลที่นำมาสร้างโครงสร้างข้อมูลแบบทรัย ได้แก่ inspiration, instant, instrument, in, inspiration, precession, prevision, procession และ provision ซึ่งจะได้โครงสร้างข้อมูลแบบทรัยดังรูปที่ 2.1

รูปที่ 2.1 โครงสร้างข้อมูลแบบทรัย



จากรูปที่ 2.1 สัญลักษณ์ที่ใช้ในโครงสร้างข้อมูลแบบทรัยได้แก่ โหนดคือวงกลมที่มีตัวเลขกำกับ ส่วนทรานสิชั่นคือตัวอักษรที่อยู่บนเส้นทางระหว่างโหนด 2 โหนดใด ๆ และเซพพาเรทโหนดแทนด้วยสี่เหลี่ยมที่มีตัวเลขกำกับ

การค้นหาดัชนีในโครงสร้างข้อมูลแบบทรัยจะนำอักษรแต่ละตัวในดัชนีมาเปรียบเทียบ ตัวอย่างเช่นต้องการค้นหาดัชนี "procession" จะนำอักษรตัวแรกมาค้นหาโดยเริ่มจากโหนด 1 ซึ่งอักษรแรก "p" จะได้เส้นทางจากโหนด 1 ไปโหนด 2 จากนั้นนำอักษรถัดไป "r" มาค้นหาต่อจากโหนดปัจจุบัน (โหนด 2) จะได้เส้นทางจากโหนด 2 ไปยังโหนด 3 ส่วนอักษรถัดไป "o" และ "c" จะได้เส้นทางจากโหนด 3 ไปยังโหนด 8 ที่เป็นเซพพาเรทโหนดเชื่อมไปยังสตริงส่วนหลัง "ession" ซึ่งตรงกับส่วนที่เหลือของดัชนีที่ใช้ในการค้นหา ดังนั้นเวลาที่ใช้ในการค้นหาจึงขึ้นกับ

จำนวนตัวอักษรที่ประกอบขึ้นมาเป็นดัชนี ไม่ขึ้นอยู่กับจำนวนคำที่เก็บทำให้ค้นหาข้อมูลได้อย่างรวดเร็วเมื่อเทียบกับปริมาณข้อมูลที่จัดเก็บ แต่ข้อเสียของโครงสร้างข้อมูลแบบทรัยคือ เมื่อนำไปใช้กับกลุ่มของดัชนี(key set) ขนาดใหญ่จะทำให้โครงสร้างข้อมูลแบบทรัยต้องใช้จำนวนโหนดและจำนวนทรานสิชั่นมากจึงทำให้ใช้เนื้อที่มากในการเก็บ และนอกจากนี้แล้วขั้นตอนวิธีในการเพิ่มและลบดัชนีของโครงสร้างข้อมูลแบบทรัย ทำให้เกิดเนื้อที่ว่างจำนวนมาก [3]

ต่อมา Jun-ichi Aoe, Katsushi Morimoto, Masami Shishibori และ Ki-Hong Park ได้ร่วมกันคิดค้นโครงสร้างข้อมูลแบบทู-ทรัย [4] ซึ่งสามารถช่วยลดขนาดของโครงสร้างข้อมูลแบบทรัยได้ ซึ่งโครงสร้างข้อมูลแบบทู-ทรัยมีลักษณะดังนี้

1) ข้อมูลของแต่ละดัชนีสามารถระบุได้อย่างเป็นเอกลักษณ์(unique)

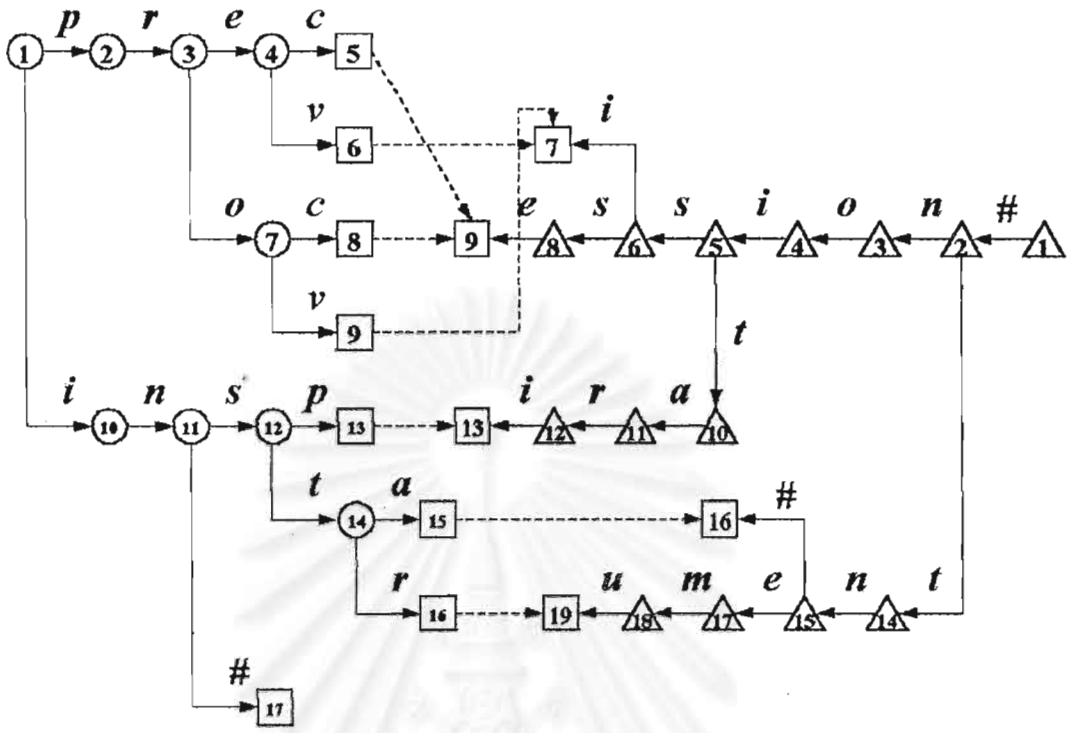
2) ลดจำนวนทรานสิชั่นโดยให้มีการใช้ทรานสิชั่นร่วมกัน ในส่วนของสตริงส่วนหน้าและในส่วนของสตริงส่วนหลัง

3) ใช้ได้กับกลุ่มของดัชนีที่มีลักษณะเป็นแบบพลวัต(dynamic) โดยการใช้ขั้นตอนวิธีในการปรับ(update) ที่มีประสิทธิภาพในการขจัดหน่วยความจำที่ไม่ได้ใช้งานในกรณีที่มีการปรับโครงสร้างข้อมูลแบบทรัย

หลักการของโครงสร้างข้อมูลแบบทู-ทรัยคือ ให้มีการใช้ทรานสิชั่นในสตริงส่วนหลังร่วมกันได้ โดยที่ยังคงความสามารถในการระบุข้อมูลของแต่ละดัชนีได้เพียงหนึ่งเดียว และเวลาที่ใช้ในการค้นหาจะขึ้นกับจำนวนตัวอักษรที่ประกอบขึ้นมาเป็นดัชนี ไม่ขึ้นอยู่กับจำนวนคำที่เก็บ ทำให้ค้นหาข้อมูลได้อย่างรวดเร็วเช่นเดียวกับโครงสร้างข้อมูลแบบทรัย

โครงสร้างข้อมูลแบบทู-ทรัยประกอบด้วยโครงสร้างข้อมูลแบบทรัย 2 ส่วนได้แก่ ทรัยสำหรับสตริงส่วนหน้าเรียกว่า ทรัยส่วนหน้า(FR Trie) และทรัยสำหรับสตริงส่วนหลังเรียกว่า ทรัยส่วนหลัง(RE Trie) ซึ่งทรัยส่วนหลังจะเก็บสตริงในรูปแบบผันกลับ(reverse) โดยที่ระหว่าง ทรัยส่วนหน้าและทรัยส่วนหลังจะมีลิงค์(link) ในการเชื่อมจากเซพพาเรทโหนดของทรัยส่วนหน้าไปยังแอคเซพติงโหนด (accepting node) ของทรัยส่วนหลัง โดยที่เซพพาเรทโหนดหมายถึงโหนดแรกที่สามารถแยกความแตกต่างของแต่ละดัชนี ดังนั้นข้อมูลของแต่ละดัชนีจึงสามารถผูกติด(attach)ไว้ที่เซพพาเรทโหนดของทรัยส่วนหน้าได้ ลักษณะของโครงสร้างข้อมูลแบบทู-ทรัยแสดงดังรูปที่ 2.2

รูปที่ 2.2 โครงสร้างข้อมูลแบบทู-ทรัย



จากรูปที่ 2.2 สัญลักษณ์ที่ใช้ในโครงสร้างข้อมูลแบบทู-ทรัยได้แก่ ลิงค์ใช้แทนด้วยเส้นประ โหนดที่อยู่ในกรอบสี่เหลี่ยมแทนแอคเซพติงโหนด (แอคเซพติงโหนดของทรัยส่วนหน้าเรียกว่า เซพพาเรทโหนด) และโหนดที่อยู่ในรูปสามเหลี่ยมแทนโหนดของทรัยส่วนหลังจะเห็นว่าสตริงที่อยู่ในทรัยส่วนหลังจะเก็บแบบผันกลับเพื่อที่จะลดทรานสิชั่นที่ซ้ำซ้อนกันในกรณีของการเพิ่มและลบดัชนี โดยมีเครื่องหมาย # เป็นโหนดสิ้นสุด

ตัวอย่างข้อมูลที่ใช้ในการสร้างโครงสร้างข้อมูลแบบทู-ทรัยในรูปที่ 2.2 จะใช้ข้อมูลเดียวกับที่ใช้สร้างโครงสร้างข้อมูลแบบทรัยในรูปที่ 2.1 ซึ่งจะเห็นว่าจำนวนโหนดและทรานสิชั่นที่ใช้ในโครงสร้างข้อมูลแบบทู-ทรัยน้อยกว่าจำนวนโหนดและทรานสิชั่นในโครงสร้างข้อมูลแบบทรัย

จากการทดสอบในงานวิจัยของ Jun-ichi Aoe et al. [4] ในการวัดประสิทธิภาพของโครงสร้างข้อมูลทรัยแบบต่าง ๆ โดยใช้กลุ่มของดัชนีที่มีขนาดแตกต่างกัน พบว่าประสิทธิภาพของเวลาที่ใช้ในการค้นหา เพิ่ม และลบดัชนีของโครงสร้างข้อมูลแบบทรัยไม่แตกต่างจากโครงสร้างข้อมูลแบบทู-ทรัยมากนัก แต่ในกรณีที่กลุ่มของดัชนีมีขนาดใหญ่มาก โครงสร้างข้อมูลแบบทู-ทรัยจะใช้เนื้อที่น้อยกว่าโครงสร้างข้อมูลแบบทรัย ดังนั้นโครงสร้างข้อมูลแบบทู-ทรัยจึงเป็นโครงสร้างข้อมูลที่มีประสิทธิภาพ และเหมาะสมกับงานทางด้านค้นหาข้อมูลมากกว่าโครงสร้างข้อมูลแบบทรัย

โครงสร้างข้อมูลทู-ทรัยในเชิงโปรแกรมแบ่งเป็น 2 แบบได้แก่ โครงสร้างเชิงรายการ (list structure) และโครงสร้างแถวลำดับคู่ (double-array structure)

2.2 Directed Hypergraph

directed hypergraph เป็นแนวความคิดเพื่อให้สามารถตรวจสอบความสัมพันธ์ของโหนดต่างๆ ได้อย่างถูกต้องและชัดเจน [6] โดยแต่ละโหนดคือส่วนที่เป็นองค์ประกอบของกฎเกณฑ์เงื่อนไข หรือข้อสรุปในระบบนั้น ซึ่งกฎเกณฑ์เงื่อนไขหรือข้อสรุปอาจเกิดจากการกำหนดร่วมกันมากกว่า 1 ได้ การสร้าง directed hypergraph นั้นไม่สามารถมีโหนดซ้ำกัน ความสัมพันธ์ของแต่ละโหนดถูกแทนด้วยเส้นที่มีทิศทาง (directed arc) ทิศทางของแต่ละเส้นแสดงโหนดที่มีลำดับสูงกว่าไปยังโหนดที่มีลำดับต่ำกว่า หรือโหนดที่เป็นข้อสรุปของโหนดนั้น ดังตัวอย่าง

กฎเกณฑ์ 1 A -> D

กฎเกณฑ์ 2 B + C -> E

กฎเกณฑ์ 3 B + C -> F

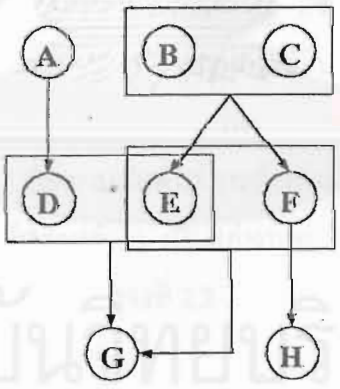
กฎเกณฑ์ 4 D + E -> G

กฎเกณฑ์ 5 E + F -> G

กฎเกณฑ์ 6 F -> H

เราสามารถสร้าง directed hypergraph ดังกฎเกณฑ์ดังกล่าวในรูปที่ 2.3

รูปที่ 2.3 directed hypergraph สำหรับกฎเกณฑ์ที่ 1 ถึง 6



directed hypergraph ถูกอธิบายโดยใช้เมตริกซ์(adajency matrix) ที่มีขนาดของแถวและสดมภ์ เท่ากับจำนวนของโหนดใน directed hypergraph ถ้ามีเส้นความสัมพันธ์จากโหนด U ไปยังโหนด V จะทำให้ค่าในเมตริกซ์ แถวที่ U สดมภ์ที่ V มีค่าเป็น 1 ส่วนที่ไม่มีเส้นความสัมพันธ์มีค่าเว้นว่าง ดังแสดงในตารางที่ 2.1

ตารางที่ 2.1 เมตริกซ์แสดง directed hypergraph ในรูปที่ 2.3

โหนด	*A*	*B*, *C*	*D*	*E*	*F*	*D*, *E*	*E*, *F*	*G*	*H*
A			1						
B, *C*				1	1				
D									
E									
F									1
D, *E*								1	
E, *F*								1	
G									
H									

เมตริกซ์ที่สร้างได้จะแสดงกฎเกณฑ์เงื่อนไข หรือข้อสรุป ทั้งหมดในระดับ 1 หรือระดับการอ้างอิงแรกเท่านั้น เราสามารถสร้าง เมตริกซ์ระดับที่สูงขึ้นโดยการอ้างอิงกฎเกณฑ์ของข้อสรุปที่ได้เช่น *B,C -> E,F* ซึ่งเป็นระดับ 1 เราอ้างอิงกฎเกณฑ์ของข้อสรุป คือ *E,F -> G* จะได้ *B,C - > G* ซึ่งเป็นระดับที่ 2 ได้

กฎเกณฑ์เงื่อนไข หรือข้อสรุปที่แสดงในเมตริกซ์ อาจได้รายละเอียดของความสัมพันธ์ยังไม่สมบูรณ์เนื่องจากอาจมีการถ่ายทอดข้อสรุปจากกฎเกณฑ์อื่นเกิดขึ้นได้ดังจากเมตริกซ์ในตารางที่ 2.1 แสดงความสัมพันธ์ว่า โหนด *H* มีการอ้างถึงกับ โหนด *F* แต่ไม่ได้แสดงว่า โหนด *H* มีการอ้างถึงกับ โหนด *E,F* ที่เป็นโหนดร่วม ซึ่งความสัมพันธ์การอ้างถึงมีความสำคัญต่อการตรวจสอบข้อผิดพลาดของกฎเกณฑ์เงื่อนไข หรือข้อสรุปต่างๆ เป็นอย่างมาก เราจึงต้องทำการแก้ไขเมตริกซ์ในจุดดังกล่าวจนได้เมตริกซ์ที่ปรับปรุงใหม่ในตารางที่ 2.2

ตารางที่ 2.2 เมตริกซ์แสดง directed hypergraph ที่ได้รับการปรับปรุง

โหนด	*A*	*B*, *C*	*D*	*E*	*F*	*D*, *E*	*E*, *F*	*G*	*H*
A			1						
B, *C*				1	1		1		
D									
E									
F									1
D, *E*								1	
E, *F*								1	1
G									
H									

เมตริกซ์ที่ได้สามารถนำไปช่วยตรวจสอบหาข้อผิดพลาดต่างๆ เช่น ข้อผิดพลาดที่เกิดขึ้นกับความสมบูรณ์ของข้อมูลที่ได้รับมา ทำให้ไม่สามารถหาผลลัพธ์ซึ่งเป็นคำตอบของปัญหาที่ต้องการได้

2.3 ภาษา mSQL [2]

2.3.1 ความสามารถของ mSQL

ภาษา mSQL มีความสามารถดังต่อไปนี้

1) mSQL เป็น database engine ขนาดเล็กที่ถูกออกแบบมาเพื่อให้มีความสามารถในการดึงข้อมูลจากฐานข้อมูลตามดิวรีที่ต้องการ และทำการเก็บข้อมูลลงในฐานข้อมูล ในเวลาอันรวดเร็ว พร้อมทั้งใช้เนื้อที่ในหน่วยความจำน้อยกว่า database engine ชนิดอื่นๆ อีกด้วย

2) mSQL ถูกออกแบบมาให้ทำงานอยู่ในระบบระบบผู้ใช้บริการ/ผู้ให้บริการโดยผ่านทางโปรโตคอล ทีซีพีไอพี

3) mSQL ใช้ภาษาสอบถามเชิงโครงสร้างซึ่งเป็นภาษามาตรฐานในการเรียกใช้ข้อมูล

2.3.2 รูปแบบของการใช้คำสั่งบางคำสั่งใน mSQL

รูปแบบของการใช้คำสั่งของภาษา mSQL สามารถแจกแจงได้ดังนี้

1) คำสั่งประโยค CREATE ใช้ในการสร้างตาราง ดัชนี และลำดับเซตข้อมูล ในฐานข้อมูล

รูปแบบ

```
CREATE TABLE table_name (
        col_name        col_type [not null ]
        [ , col_name    col_type [not null] ]**
)
CREATE [UNIQUE] INDEX index_name ON table_name (
        field_name      [ , field_name ] **
)
CREATE SEQUENCE ON table_name [STEP step_val] [VALUE init_val]
```

ตัวอย่างของการสร้างตาราง

```
CREATE TABLE emp_details (
        first_name      char(15) not null,
        last_name       char(15) not null,
        comment         text(50),
        dept            char(20),
        emp_id          int
)
```

จากตัวอย่างจะเป็นการสร้างตารางที่ชื่อว่า emp_details ประกอบด้วย 5 เขตข้อมูล ซึ่งแต่ละเขตข้อมูลสามารถมีชนิดของข้อมูลได้ดังตารางที่ 2.3

ตารางที่ 2.3 ตารางชนิดของเขตข้อมูล

ชนิดเขตข้อมูล	คำอธิบาย
Char(len)	เป็นอักขระ (Character) ระบุความยาวในวงเล็บ
Text (len)	เป็นข้อความ ระบุความยาวเฉลี่ยในวงเล็บ การเข้าถึงข้อมูลได้ช้ากว่า แบบอักขระ ไม่สามารถถูกใช้ในเงื่อนไข Like ได้
Int	เป็นจำนวนเต็ม มีค่าบวกลบ
Real	เป็นจำนวนจริง มีค่าเป็นเลขทศนิยม
Uint	เป็นจำนวนเต็ม ไม่มีค่าเป็นลบ
Date	เป็นวันที่ ซึ่งอยู่ในรูปแบบ DD-Mon-YYYY เช่น 1-Jan-2000
Time	เป็นเวลา ซึ่งเก็บแบบ 24 ชั่วโมง อยู่ในรูปแบบ HH:MM:SS
Money	เป็นค่าตัวเลข มีทศนิยม 2 ตำแหน่ง

ตารางที่สร้างขึ้นนั้น emp_id ซึ่งเป็นค่าเอกลักษณ์ที่ใช้ในการอ้างถึงพนักงานคนหนึ่ง เราจึงจะกำหนดให้ emp_id เป็นกุญแจหลักของตาราง และใช้ first_name กับ last_name เป็นดัชนีร่วมกัน และไม่เป็นเอกลักษณ์เพราะอาจมีระเบียนอื่นมีซ้ำได้

CREATE UNIQUE INDEX idx1 ON emp_details (emp_id)

CREATE INDEX idx2 ON emp_details (first_name, last_name)

ดัชนีที่ถูกสร้างขึ้นจะถูกเรียกโดย Database engine เมื่อไรก็ตามที่มีการค้นหาข้อมูลซึ่งมีการอ้างถึงเขตข้อมูลนั้นๆ ในประโยคเงื่อนไข การสร้างลำดับเขตข้อมูลเพื่อใช้เป็นดัชนีต่างๆ ซึ่งลำดับเขตข้อมูลสามารถถูกปรับปรุงได้โดยเซิร์ฟเวอร์ของ mSQL จากคุณสมบัติดังกล่าวทำให้เซิร์ฟเวอร์ใช้งานในการจัดการเกี่ยวกับ atomic operations และ race condition ได้ ตัวอย่างข้างล่างจะเป็นการสร้างลำดับข้อมูลในตาราง emp_details

CREATE SEQUENCE ON emp_details STEP 1 VALUE 5

SELECT _seq FROM emp_detail

การสร้างลำดับเขตข้อมูลข้างต้นจะกำหนดค่าลำดับเขตข้อมูลเริ่มที่ 5 และเมื่อถูกเรียกใช้จะถูกเพิ่มค่าไป 1 คือ เมื่อมีการเรียกเขตข้อมูล _seq ดังประโยค SELECT ข้างบนจะได้ค่ากลับมาเป็น 5 เมื่อทำการเรียกอีกครั้งจะได้รับค่าเป็น 6 ซึ่งการเพิ่มค่าหลังถูกเรียกใช้สามารถเป็นการลดค่าลำดับได้เช่นเดียวกันโดยใช้ STEP เป็นลบ

2) คำสั่งประโยค DROP ใช้ในการลบตาราง ดัชนี และลำดับเขตข้อมูล ในฐานข้อมูล

<u>รูปแบบ</u>

DROP TABLE table_name

DROP INDEX index_name FROM table_name

DROP SEQUENCE FROM table_name

<u>ตัวอย่างการใช้</u>

DROP TABLE emp_details

DROP INDEX idx1 FROM emp_details

DROP SEQUENCE FROM emp_details

3) คำสั่งประโยค INSERT ใช้ในการเพิ่มระเบียนข้อมูลลงตารางในฐานข้อมูล

<u>รูปแบบ</u>

INSERT INTO table_name [(column [, column] **)]

VALUES (value [, value] **)

ตัวอย่างการใช้

```
INSERT INTO emp_names (first_name, last_name, dept, salary)
        VALUES ('David', 'Hughes', 'Development', 12345.00)
INSERT INTO emp_names
        VALUES ('David', 'Hughes', 'Development', 12345.00)
```

4) คำสั่งประโยค DELETE ใช้ในการลบระเบียนข้อมูลออกจากตาราง

รูปแบบ

```
DELETE FROM table_name
WHERE column OPERATOR value [ AND|OR column OPERATOR value ] **
```

หมายเหตุ OPERATOR ที่สามารถใช้ได้มีดังนี้ ">" "<" "=" "<=" ">=" "<>" และ "LIKE"

ตัวอย่างการใช้

```
DELETE FROM emp_details WHERE emp_id = 12345
DELETE FROM emp_details WHERE salary > 20000 AND salary < 30000
```

5) คำสั่งประโยค UPDATE ใช้ในการเปลี่ยนค่าของเขตข้อมูลใดข้อมูลหนึ่งในตาราง

รูปแบบ

```
UPDATE table_name SET column = value [ , column = value]**
WHERE column OPERATOR value [ AND|OR column OPERATOR value ] **
```

หมายเหตุ OPERATOR ที่สามารถใช้ได้มีดังนี้ ">" "<" "=" "<=" ">=" "<>" และ "LIKE"

ตัวอย่างการใช้

```
UPDATE emp_details SET salary = 30000 WHERE emp_id = 1234
UPDATE emp_details SET salary = 35000, dept = 'Development'
WHERE emp_id = 1234
```

6) คำสั่งประโยค SELECT ใช้ในการเรียกระเบียนข้อมูลที่ต้องการจากตาราง

ข้อจำกัดของคำสั่ง

- ไม่สามารถเรียกระเบียนข้อมูลแบบซ่ายได้ (nested select)
- ไม่มีฟังก์ชันต่างๆ (aggregate function) ให้ใช้เช่น count()avg()

ความสามารถ

- สามารถทำการแสดงผลข้อมูลที่อยู่ในตารางที่แตกต่างกันได้ (join table)
- การกำหนดชื่ออ้างอิงแทนตาราง (Table alias)

- สามารถเลือกแสดงข้อมูลที่ไม่ซ้ำซ้อนกันได้ (DISTINCT row selection)
- สามารถเรียงลำดับข้อมูลได้ (ORDER BY)
- สามารถทำการเปรียบเทียบค่าระหว่างเขตข้อมูลได้ในคำสั่ง WHERE
- สามารถทำการค้นหาข้อมูลที่มีความเหมือนหรือตรงตามคำสั่งต่างๆ ดังนี้

  '_' ใช้แทนตัวอักษรใดๆ หนึ่งตัว

  '%' ใช้แทนตัวอักษรหนึ่งตัวหรือมากกว่า

  '\' ใช้ในการค้นหาตัวอักขระพิเศษเช่น % จะใช้ \%

รูปแบบ

```
SELECT [table.]column [ , [table.]column ]**
FROM table [ = alias] [ , table [ = alias ]]**
[ WHERE [table.]column OPERATOR VALUE
[ AND|OR [table.]column OPERATOR VALUE]**]
[ ORDER BY [table.]column [DESC] [ , [table.]column [DESC]]**]
```

หมายเหตุ OPERATOR ที่สามารถใช้ได้มีดังนี้ ">" "<" "=" "<=" ">=" "<>" และ "LIKE"

ตัวอย่างการใช้

```
SELECT first_name, last_name FROM emp_details WHERE dept = 'Finance'
```

หรือ

```
SELECT first_name, last_name FROM emp_details WHERE dept = 'Finance'
ORDER BY last_name, first_name DESC
```

หรือ

```
SELECT DISTINCT first_name, last_name FROM emp_details
WHERE dept = 'Finance'
ORDER BY last_name, first_name DESC
```

หรือ

```
SELECT first_name, last_name FROM emp_details WHERE dept = '_inance'
ORDER BY last_name, first_name DESC
```

หรือ

```
SELECT first_name, last_name FROM emp_details WHERE dept = 'Fin%'
ORDER BY last_name, first_name DESC
```

ตัวอย่างคิวรีที่ใช้การ join table

```
SELECT emp_details.first_name, emp_details.last_name, project.project_name
FROM emp_details, project
```

```
WHERE  emp_details.emp_id = project.emp_id
ORDER  BY  emp_details.last_name, emp_details.first_name
```

ตัวอย่างคิวรีที่ใช้ alias

```
SELECT  e1.first_name, e1.last_name,  p1.project_name
FROM  emp_details = e1, project = p1
WHERE  e1.emp_id = p1.emp_id
ORDER  BY  e1.last_name, e1.first_name
```

หรือ

```
SELECT  t1.parent, t2.child FROM  parent_data = t1 , parent_data = t2
WHERE  t1.child = t2.parent
```

ในที่นี้ t1 และ t2 อ้างอิงไปยังตาราง parent_data เหมือนกัน

2.3.3 ส่วนต่อประสานกับโปรแกรมประยุกต์ (API: Application Programming Interface)

ส่วนต่อประสานกับโปรแกรมประยุกต์ของ mSQL ซึ่งมีไลบรารี ชื่อว่า libmsql.a เพื่อให้โปรแกรมประยุกต์เรียกใช้ได้โดยการเขียนโปรแกรมภาษาซี ทำให้สามารถเข้าถึงฐานข้อมูลบนระบบบริหารฐานข้อมูลใด ๆ ได้ และต้องมีการเพิ่ม msql.h ไว้ที่ส่วนเริ่มต้นของโปรแกรมที่เขียนขึ้นให้สามารถใช้ฟังก์ชันต่างๆ ที่มีมาให้ได้

2.4 แนวความคิดในการนำระบบเครือข่ายมาใช้

หลังจากการที่การพัฒนาระบบเครือข่ายได้ก้าวหน้าขึ้น มีการพัฒนาระบบซอฟต์แวร์ที่ทำให้มีการเชื่อมโยงคอมพิวเตอร์หลายๆ ระบบเข้าเป็นระบบเดียวกันได้ ทำให้การใช้ทรัพยากรต่างๆ ในระบบเกิดประสิทธิภาพสูงสุด ซึ่งทรัพยากรที่สามารถใช้งานร่วมกันได้ โดยระบบที่นำมาใช้คือ ระบบผู้ใช้บริการ/ผู้ให้บริการติดต่อกันโดยใช้ซอคเก็ตผ่านโปรโตคอลทีซีพี/ไอพี (tcp/ip)

2.5 Tcl and Tk Toolkit [7]

Tcl (Tool Command Language) เป็นภาษาสคริปต์โดยใช้การ interprete ไม่ใช่การ compile มีความสามารถทางด้านการพัฒนาโปรแกรมทั่วๆ ไปเช่น การใช้ตัวแปร(variable) การทำซ้ำ(loop) และการใช้โปรแกรมย่อย(procedure) เป็นต้น

Tk เป็น Toolkit สำหรับระบบ X Window ซึ่งจะช่วยเพิ่มความสามารถให้กับ Tcl ในการสร้างส่วนติดต่อกับผู้ใช้ จากการใช้ Tk นี้ ทำให้เราสามารถสร้าง X Window-like User Interface ได้โดยง่าย

ข้อดีของการใช้ Tcl/Tk Toolkit

1) สามารถพัฒนาโปรแกรมได้โดยใช้เวลาไม่นานนักและเป็นภาษาที่เขียนง่าย เนื่องจากใช้การ interprete จึงง่ายต่อการแก้ไข

2) การพัฒนาโปรแกรมที่ต้องการเชื่อมโยงกับระบบโปรแกรมประยุกต์ที่เขียนด้วยภาษาอื่น เช่น ภาษาซี สามารถทำได้โดยง่าย

3) ภาษา Tcl มีไลบรารีแพกเก็ตที่ช่วยเพิ่มประสิทธิภาพให้กับโปรแกรมและอำนวยความสะดวกมากมายสำหรับการเขียนโปรแกรม นอกจากนี้ ผู้เขียนโปรแกรมสามารถสร้าง ไลบรารีแพกเก็ตขึ้นมาเองได้โดยใช้ภาษาซี ตัวอย่างของไลบรารีแพกเก็ตที่น่าสนใจ เช่น ความสามารถทางด้านการสื่อสารระหว่างระบบโปรแกรมอื่นๆ ลักษณะเดียวกับ OLE ของ Microsoft หรือ ToolTalk ของ Sun Microsystem

บทที่ 3

การวิเคราะห์ และ ออกแบบฟังก์ชันย่อยของระบบ

การพัฒนาระบบที่มีส่วนที่ติดต่อกับผู้ขอใช้บริการให้รองรับคำสั่งของผู้ขอใช้บริการที่เป็นภาษาธรรมชาตินั้น สามารถแบ่งขั้นตอนของการพัฒนาฟังก์ชันได้ออกเป็น 4 ส่วนใหญ่ดังนี้

1) พจนานุกรมฐานความรู้
2) ตารางความสัมพันธ์องค์ประกอบของระบบฐานข้อมูล
3) ไวยกรณ์กับการวิเคราะห์วัตถุประสงค์ของคำร้องขอของผู้ขอใช้บริการโดยใช้หลักสัมพันธ์
4) การผสานผลลัพธ์ของข้อมูลเพื่อนำเสนอ

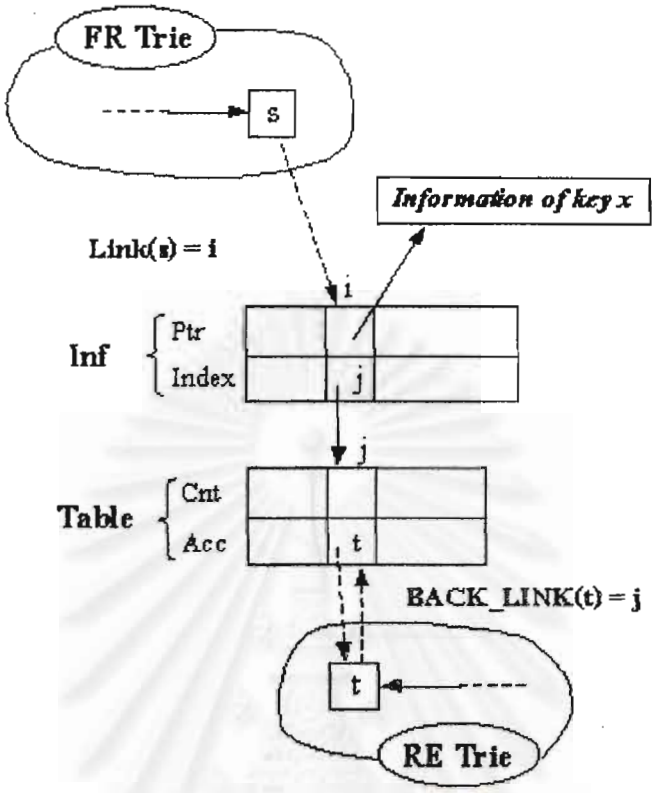
3.1 พจนานุกรมฐานความรู้

โครงสร้างข้อมูลของดัชนีที่นำมาใช้ในพจนานุกรมฐานความรู้นี้ได้แก่โครงสร้างข้อมูลแบบทู-ทรัย เนื่องจากโครงสร้างข้อมูลแบบทู-ทรัย มีลักษณะเด่นต่าง ๆ ที่เหมาะกับงานวิจัยเนื่องจากมีความรวดเร็วในการค้นคืนข้อมูลซึ่งไม่ขึ้นกับขนาดของข้อมูล และใช้เนื้อที่จัดเก็บน้อย

โครงสร้างข้อมูลแบบทู-ทรัยในเชิงโปรแกรมแบ่งเป็น 2 ลักษณะ ได้แก่ โครงสร้างเชิงรายการ และโครงสร้างแถวลำดับคู่ ซึ่งการเพิ่มและลบข้อมูลในโครงสร้างเชิงรายการสามารถทำได้รวดเร็วกว่าโครงสร้างแถวลำดับคู่ แต่โครงสร้างแถวลำดับคู่จะมีประสิทธิภาพในการค้นหาข้อมูลดีกว่าโครงสร้างเชิงรายการ

ในงานวิจัยนี้เลือกใช้โครงสร้างแถวลำดับคู่ในการสร้างโครงสร้างข้อมูลแบบทู-ทรัยในเชิงโปรแกรม เนื่องจากในส่วนการทำงานหลักของงานวิจัยนี้จะเน้นการค้นหาข้อมูลที่ต้องการมากกว่าการทำการเพิ่มและลบข้อมูลในโครงสร้างข้อมูล ลักษณะโครงสร้างข้อมูลแบบทู-ทรัยโดยใช้โครงสร้างแถวลำดับคู่ในเชิงโปรแกรม แสดงดังรูปที่ 3.1

รูปที่ 3.1 โครงสร้างแบบทู-ทรัยโดยใช้โครงสร้างแถวลำดับคู่ในเชิงโปรแกรม



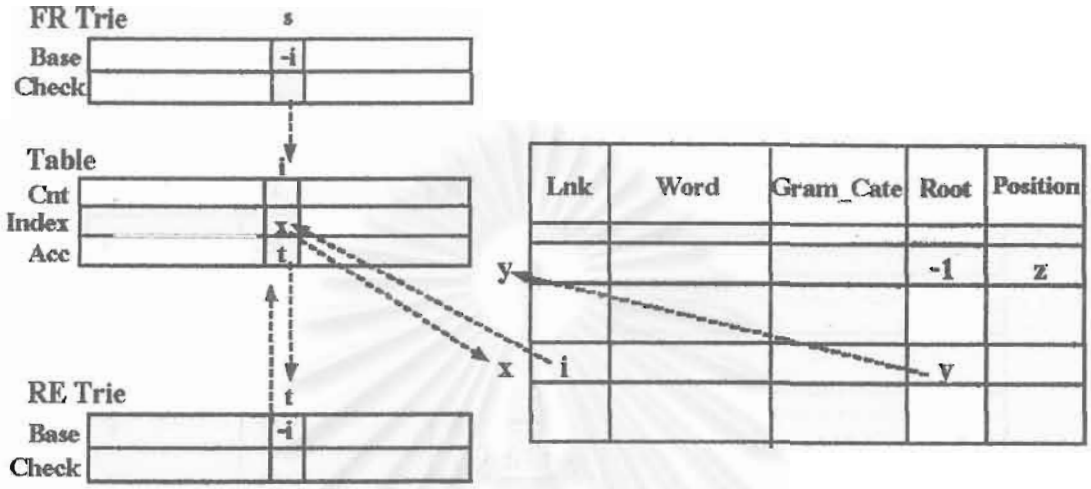
โครงสร้างข้อมูลแบบทู-ทรัยในเชิงโปรแกรมจากรูปที่ 3.1 ประกอบด้วยโครงสร้างแถวลำดับคู่จำนวน 4 แถวได้แก่ แถวลำดับคู่ของทรัยส่วนหน้า แถวลำดับคู่ของทรัยส่วนหลัง แถวลำดับคู่ของข้อมูล(Inf) และ แถวลำดับคู่เทเบิล(Table) โดยที่แถวลำดับคู่ของทรัยส่วนหน้าและแถวลำดับคู่ของทรัยส่วนหลังประกอบด้วยอาร์เรย์(Array) ขนาด 1 มิติ จำนวน 2 อาร์เรย์คืออาร์เรย์เบส (Base) และอาร์เรย์เช็ค(Check) มีหมายเลขประจำโหนดเป็นดัชนีเพื่อแสดงความสัมพันธ์ระหว่างเบส และเช็ค แถวลำดับคู่ของข้อมูลประกอบด้วยอาร์เรย์พอยท์เตอร์(Ptr) และอาร์เรย์อินเดกซ์ (Index) และแถวลำดับคู่เทเบิลประกอบด้วยอาร์เรย์เคาท์เตอร์(Cnt) และอาร์เรย์แอคเซพติง(Acc)

จากงานวิจัยโครงสร้างข้อมูลแบบทู-ทรัยของ Jun-ichi Aoe, Katsushi Morimoto, Masami Shishibori และ Ki-Hong Park ไม่ได้ระบุลักษณะการจัดเก็บข้อมูลของแต่ละดัชนีไว้ ผู้วิจัยจึงได้ออกแบบส่วนของการจัดเก็บข้อมูลของแต่ละดัชนีให้เหมาะสมกับงานวิจัยพจนานุกรมฐานข้อมูลที่พัฒนาขึ้น

เนื่องจากงานวิจัยในส่วนนี้ใช้ดัชนีในการค้นหาข้อมูลทำให้ข้อมูลของแต่ละดัชนีที่จัดเก็บสามารถมีการอ้างอิงถึงความสัมพันธ์กับดัชนีข้อมูลอื่นที่มีลำดับในฐานข้อมูลสูงกว่าได้ ดังนั้นจึงได้ทำการออกแบบส่วนของการจัดเก็บข้อมูลของแต่ละดัชนีเป็นลักษณะโครงสร้างของอาร์เรย์ที่จัดเก็บองค์ประกอบต่างๆ ของข้อมูล อีกทั้งผู้วิจัยได้เพิ่มเติมขั้นตอนการแปลงตัวอักษรภาษาอังกฤษที่เป็นดัชนีให้เป็นอักษรตัวเล็กทั้งหมด ทั้งนี้เพื่อให้เกิดความคล่องตัวในการวิเคราะห์ค้นหาข้อมูล

ดัชนี โดยที่ข้อมูลบนฐานข้อมูลยังคงลักษณะเดิม ลักษณะของโครงสร้างข้อมูลแถวลำดับคู่ที่ออกแบบไว้จะแสดงได้ดังรูปที่ 3.2

รูปที่ 3.2 โครงสร้างแถวลำดับคู่ที่ได้ออกแบบส่วนการจัดเก็บข้อมูลของแต่ละดัชนี



3.1.1 โครงสร้างแถวลำดับพจนานุกรมฐานความรู้

พจนานุกรมฐานความรู้จะรวบรวมคำต่างๆ ที่ถูกใช้ในระบบ คำต่างๆ จะแบ่งได้เป็น 2 ประเภทคือ ประเภทแรกเป็นชื่อความสัมพันธ์ เขตข้อมูล และค่าต่างๆ ที่เป็นได้ของเขตข้อมูล ยกเว้นค่าที่เป็นจำนวน หรือ ตัวเลข ประเภทที่ 2 คำที่ใช้ทั่วไปไม่ได้เกี่ยวข้องกับค่าในฐานข้อมูล แต่เป็นส่วนให้ความหมายของประโยคมีความสมบูรณ์ ซึ่งคำทั้ง 2 ประเภทอาจได้มาจากการศึกษาคำร้องขอของผู้ขอใช้บริการ ถ้าปรากฏคำที่ไม่อยู่ในพจนานุกรมฐานความรู้จะทำการเพิ่มเติมคำดังกล่าวลงในพจนานุกรมฐานความรู้ การเพิ่มเติมคำใหม่ลงในพจนานุกรมฐานความรู้นั้นคำนั้นต้องมีส่วนเพิ่มเติมของข้อมูลที่ประกอบไปด้วย ชนิดไวยากรณ์ของคำ ดัชนีคำอ้างอิง และดัชนีความสัมพันธ์ของคำในตารางแสดงความสัมพันธ์องค์ประกอบของฐานข้อมูล

คำต่างๆ สามารถมีชนิดไวยากรณ์ของคำ ได้แก่ คำนาม คำกริยา เป็นต้น โดยที่คำแต่ละคำอาจมีชนิดของไวยากรณ์มากกว่า 1 ชนิดได้ อาทิเช่น "name" มีชนิดของไวยากรณ์เป็นทั้งคำนามและคำกริยาได้ วิธีกำหนดการจัดเก็บคำที่เป็นค่าต่างๆ ที่สามารถเป็นได้ในเขตข้อมูลของฐานข้อมูลที่มีลักษณะเป็นการใช้คำหลายคำประกอบเข้าด้วยกันนั้นจะต้องเก็บแยกที่ละคำพร้อมทั้งเพิ่มชนิดไวยากรณ์ของคำเป็นประเภท กลุ่มคำ ด้วย ในตารางที่ 3.1 จะแสดงชนิดไวยากรณ์ที่มีเพื่อกำหนดให้คำต่างๆ ดังนั้นค่าชนิดไวยากรณ์ของ "name" คือ 16 (ค่าของคำนาม) + 32 (ค่าของคำกริยา) เท่ากับ 48

ตารางที่ 3.1 ตารางกำหนดค่าชนิดไวยากรณ์

บิตที่	ค่าที่กำหนด	ชนิดไวยากรณ์
0	1	ฐานข้อมูล (database)
1	2	ตาราง (table)
2	4	เขตข้อมูล (field)
3	8	ดัชนีตัวบ่งชี้ (key)
4	16	คำนาม (noun)
5	32	คำกริยา (verb)
6	64	คำคุณศัพท์ (adjective)
7	128	คำบุพบท (preposition)
8	256	คำสันธาน (conjunction)
9	512	คำสรรพนาม (pronoun)
10	1024	คำประพันธสรรพนาม (relative pronoun)
11	2048	ตัวกำหนด (deteminant)
12	4096	ค่าของเขตข้อมูล (value)
13	8192	แนวคำถาม (relative question)
14	16384	กลุ่มคำ (grouped value)

ค่าดัชนีคำอ้างอิง (ROOT) เป็นค่าที่ชี้เฉพาะเจาะจงความสัมพันธ์ของคำศัพท์นั้นไปยังคำศัพท์ในฐานข้อมูลที่มีระดับสูงขึ้นไป คำศัพท์ที่ไม่มีการชี้เฉพาะไปยังคำศัพท์ใดๆ ค่าดัชนีคำอ้างอิงจะมีค่าเป็น 0 ส่วนคำศัพท์ที่มีคำอ้างอิงจะนำค่าดัชนีรายการในแถวลำดับข้อมูลของคำอ้างอิงนั้นมากำหนดแทนดังในตารางที่ 3.2

ตารางที่ 3.2 โครงสร้างแถวลำดับพจนานุกรมฐานความรู้

LNK	WORD	GRAM_CATE	ROOT	POSITION
12	TEST	17	0	17
5	EMP	18	1	14
7	DEP	18	1	15
13	emp_id	24	2	1
14	ename	20	2	2
27	budget	20	3	10
169	like	32	0	0

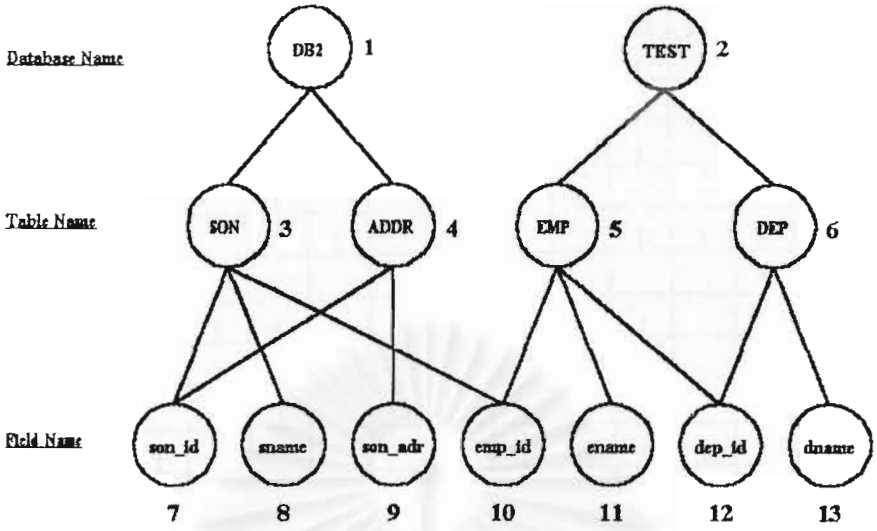
ดัชนีความสัมพันธ์ของคำในตารางแสดงความสัมพันธ์องค์ประกอบของฐานข้อมูล (POSITION) จะถูกกำหนดให้เฉพาะคำศัพท์ที่เป็นเขตข้อมูล ตาราง หรือฐานข้อมูลเท่านั้น นอกนั้นจะถูกใช้กำหนดเพื่อแยกประเภทชนิดของคำประเภทที่ย่อยลงไปอีกของคำประเภทคำกริยา และคำประพันธสรรพนาม

ดังในตารางที่ 3.2 “ename” มีชนิดไวยากรณ์เป็นคำนามกับเขตข้อมูล ค่าดัชนีคำอ้างอิงเท่ากับ 2 อ้างอิงไปยัง “EMP” (ซึ่งเป็นชื่อตารางจะเห็นว่า “emp_id” มีค่าดัชนีคำอ้างอิงเท่ากับ 2 ด้วยเนื่องจากทั้งคู่เป็นเขตข้อมูลภายใต้ตารางเดียวกัน) และมีค่าดัชนีความสัมพันธ์ของคำในตารางแสดงความสัมพันธ์องค์ประกอบของฐานข้อมูลเท่ากับ 2

3.2 ตารางความสัมพันธ์องค์ประกอบของฐานข้อมูล

การสร้างตารางความสัมพันธ์องค์ประกอบของฐานข้อมูลได้นำเอาแนวความคิดของ DH ในเชิงวัตถุเพื่ออธิบายความสัมพันธ์ โดยให้องค์ประกอบที่อยู่ในฐานข้อมูลทั้ง ชื่อเขตข้อมูล ชื่อตาราง และชื่อฐานข้อมูล เป็นโหนดต่างๆ ในกราฟ เส้นเชื่อมระหว่างโหนดแสดงความสัมพันธ์ที่แต่ละโหนดมีต่อกัน โดยที่ข้อความกำกับ(label) ของเส้นเชื่อมจะบอกลักษณะความสัมพันธ์ของโหนดทั้งคู่ ดังในรูป 3.3 ซึ่งเป็นตัวอย่างการอธิบายความสัมพันธ์ขององค์ประกอบทั้งหมดในระบบฐานข้อมูลที่สร้างขึ้น และมีการกำหนดหมายเลขอ้างอิงให้แต่ละโหนดด้วยเพื่อสะดวกในการอ้างอิงถึงโหนดนั้นๆ ในการสร้างตารางความสัมพันธ์

รูปที่ 3.3 กราฟแสดงความสัมพันธ์ในระบบฐานข้อมูล



จากกราฟที่แสดงความสัมพันธ์ในรูปที่ 3.3 สามารถนำมาถ่ายทอดความสัมพันธ์ดังกล่าวสู่ตารางความสัมพันธ์ขนาด N โหนด x N โหนด ดังตารางที่ 3.3 ซึ่งเป็นการถ่ายทอดจากความสัมพันธ์ของโหนดต่างๆ โดยมีกฎเกณฑ์ดังนี้

โหนดระดับฐานข้อมูล จะถ่ายทอดความสัมพันธ์กับโหนดอื่นๆ จนถึงโหนดระดับล่างสุดคือโหนดระดับเขตข้อมูล เช่น โหนด TEST จะมีความสัมพันธ์กับโหนด TEST โหนด EMP โหนด DEP โหนด emp_id โหนด ename โหนด dep_id และ โหนด dname

โหนดระดับตารางจะถ่ายทอดความสัมพันธ์กับโหนดระดับฐานข้อมูลและโหนดระดับเขตข้อมูล เช่น โหนด EMP จะมีความสัมพันธ์กับโหนด TEST โหนด EMP โหนด emp_id โหนด ename และ โหนด dep_id

โหนดระดับเขตข้อมูลจะถ่ายทอดความสัมพันธ์ไปยังโหนดระดับเดียวกัน และขึ้นไปยังโหนดระดับที่สูงกว่าจนถึงโหนดระดับฐานข้อมูล โดยโหนดระดับเขตข้อมูลเท่านั้นที่สามารถถ่ายทอดความสัมพันธ์กับโหนดระดับเดียวกันได้ อีกทั้งทุกโหนดย่อมมีความสัมพันธ์กับโหนดของตนอยู่แล้ว อาทิเช่น โหนด emp_id จะมีความสัมพันธ์กับโหนด emp_id โหนด son_id โหนด sname โหนด ename โหนด dep_id โหนด EMP โหนด SON และ โหนด TEST

จากตัวอย่างจะเห็นได้ว่าการสร้างตารางเพื่อเก็บความสัมพันธ์ระหว่างโหนดต่างๆ ที่มีการกำหนดค่าอ้างอิงโหนดมีความสมมาตร ทำให้การจัดเก็บตารางความสัมพันธ์ที่ใช้ในระบบจะมีขนาดเพียง N(N - 1) / 2 เท่านั้น โดยค่าที่เก็บในจะมีค่าเป็น 0 หรือ 1 เท่านั้น ดังนั้นการเก็บค่าจริงที่เกิดขึ้นจะทำการจัดเก็บด้วยวิธีปฏิบัติการบนบิต(bit vector) ทำให้เนื้อที่ที่ใช้สร้างตารางความสัมพันธ์ลดลงได้มาก ดังในตารางที่ 3.4

ตารางที่ 3.3 ตารางแสดงความสัมพันธ์ระบบฐานข้อมูล

โหนด	1	2	3	4	5	6	7	8	9	10	11	12	13
1	1		1	1			1	1	1	1			
2		1			1	1				1	1	1	1
3	1		1				1	1	1	1			
4	1			1			1		1				
5		1			1					1	1	1	
6		1				1						1	1
7	1		1	1			1	1	1	1			
8	1		1				1	1		1			
9	1			1			1		1				
10	1	1	1		1		1	1		1	1	1	
11		1			1					1	1	1	
12		1			1	1				1	1	1	1
13		1				1						1	1

ตารางที่ 3.4 ตารางความสัมพันธ์ที่ได้รับการปรับปรุง

โหนด	1	2	3	4	5	6	7	8	9	10	11	12	13
1	1		1	1			1	1	1	1			
2		1			1	1				1	1	1	1
3			1				1	1	1	1			
4				1			1		1				
5					1					1	1	1	
6						1						1	1
7							1	1	1	1			
8								1		1			
9									1				
10										1	1	1	
11											1	1	
12												1	1
13													1

การใช้ตารางความสัมพันธ์องค์ประกอบของระบบฐานข้อมูล เพื่อตรวจสอบความสมบูรณ์ของคำร้องขอของผู้ใช้บริการ

กำหนดให้

S คือเซตของสมาชิกทั้งหมดในตารางความสัมพันธ์

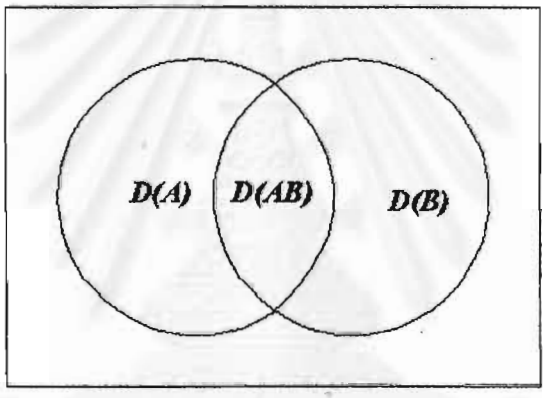
D(A) คือ เซตของสมาชิกใน *S* ที่มีความสัมพันธ์กับ *A* ซึ่งหมายถึง สมาชิกที่มีค่าของความสัมพันธ์เป็น 1 และอยู่ในแถวที่ *A* หรือ สดมภ์ที่ *A* ในตารางความสัมพันธ์

T(A) คือเซตของสมาชิกที่เป็นตารางและมีความสัมพันธ์กับ *A* โดย

$T(A) \subseteq D(A)$

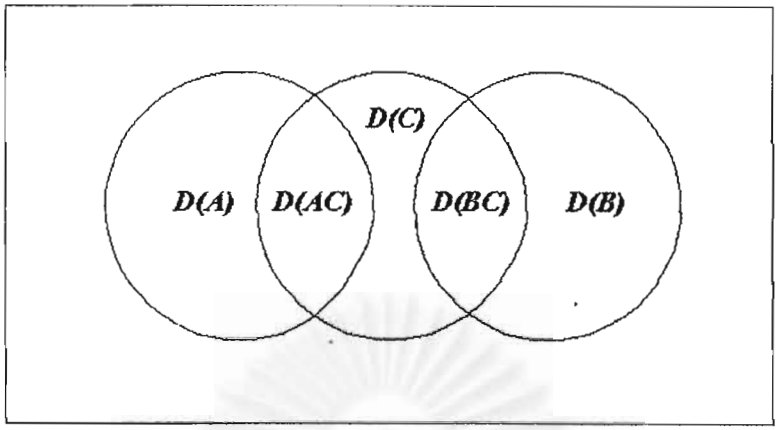
คำร้องขอของผู้ใช้บริการคือความสัมพันธ์ระหว่างสิ่งที่ผู้ใช้บริการต้องการ (*R*) กับสิ่งที่ผู้ใช้บริการทราบ (*K*) และถ้า $D(R) \subseteq D(K)$ (*D(K)* รวมความสัมพันธ์ถ่ายทอดของสมาชิกในเซ็ตด้วย) แสดงว่าสามารถหาข้อมูลได้ตามคำร้องขอ

รูปที่ 3.4 แสดงความสัมพันธ์ระหว่างเซต ***A*** กับ เซต ***B***



ถ้า $c \in (D(A) \cap D(B)) = D(AB)$ ซึ่ง *A* กับ *B* เป็นตารางในระบบฐานข้อมูลแล้ว *c* เป็นเซตข้อมูลที่เป็นตัวเชื่อมความสัมพันธ์ระหว่าง ตาราง *A* กับ ตาราง *B* ดังรูปที่ 3.4 ถ้า *c* เท่ากับ $\varnothing$ ตาราง *A* กับตาราง *B* ไม่มีเซตข้อมูลที่มีเป็นตัวเชื่อมความสัมพันธ์โดยตรง แต่อาจมีความสัมพันธ์กันโดยการถ่ายทอดผ่านเซตข้อมูลที่เป็นตัวกลางอื่นได้ เช่น $D(A) \cap D(B) = \varnothing$, $D(A) \cap D(C) = D(AC) != \varnothing$ และ $D(B) \cap D(C) = D(BC) != \varnothing$ แล้ว $D(AC) \cap D(BC) = \varnothing$ แต่มี $e \in C$ ซึ่ง e มีความสัมพันธ์กับสมาชิกใน *D(AC)* และ *D(BC)* ดังนั้นจะได้ว่า *A* และ *B* มีความสัมพันธ์กัน ดังรูปที่ 3.5

รูปที่ 3.5 แสดงความสัมพันธ์ระหว่างเซต *A* เซต *B* และ เซต *C*



3.3 ไวยากรณ์กับการวิเคราะห์วัตถุประสงค์ของคำถาม โดยใช้หลักความสัมพันธ์

3.3.1 รูปแบบไวยากรณ์

รูปแบบไวยากรณ์ที่ใช้ในงานวิจัยนี้เพื่อค้นคืนข้อมูลของผู้ใช้ระบบมี 2 รูปแบบ คือ ภาษาสอบถามเชิงโครงสร้างกับภาษาธรรมชาติ

รูปแบบแรกคือไวยากรณ์ของภาษาสอบถามเชิงโครงสร้าง ซึ่งใช้รูปแบบไวยากรณ์ของ mSQL รวมถึงการเพิ่มความสามารถในการรองรับภาษาสอบถามเชิงโครงสร้างแบบข่ายซึ่งไม่อยู่ในรูปแบบที่ mSQL ยอมรับดังนั้นการแปลงภาษาสอบถามเชิงโครงสร้างแบบข่ายให้อยู่ในรูปแบบภาษาสอบถามเชิงโครงสร้างแบบ multi-relation ที่ระบบฐานข้อมูล mSQL ยอมรับจะใช้คุณสมบัติการสมมูลกันของภาษาสอบถามเชิงโครงสร้างแบบข่ายกับแบบ multi-relation [8] ภาษาสอบถามเชิงโครงสร้างแบบข่ายที่พิจารณามี 3 ชนิด ดังนี้

พิจารณาตัวอย่างตารางภายในฐานข้อมูล มีรูปแบบความสัมพันธ์ดังนี้

Emp (**#emp**, name, #dep, age, job, sal, commission, town)

Dep (**#dep**, budget, size, city)

Anc (**#emp**, **#dep**, years)

เขตข้อมูลในตารางที่เป็นตัวหนาแสดงถึง กุญแจหลัก

ภาษาสอบถามเชิงโครงสร้างแบบ multi-relation

select *#emp, name* ***from*** *Emp, Dep*

where *Dep.#dep = Emp.#dep* ***and*** *town = city*

ภาษาสอบถามเชิงโครงสร้างแบบข่ายที่ 1 (***in***)

select *#emp, name* ***from*** *Emp* ***where*** *#dep* ***in***

(select #dep from Dep where city = Emp.town)

ภาษาสอบถามเชิงโครงสร้างแบบข่ายที่ 2 (*exist*)

select #emp, name from Emp where exist

*(select * from Dep where #dep = Emp.#dep and city = Emp.town)*

ภาษาสอบถามเชิงโครงสร้างแบบข่ายที่ 3 (*any*)

select #emp, name from Emp where #dep = any

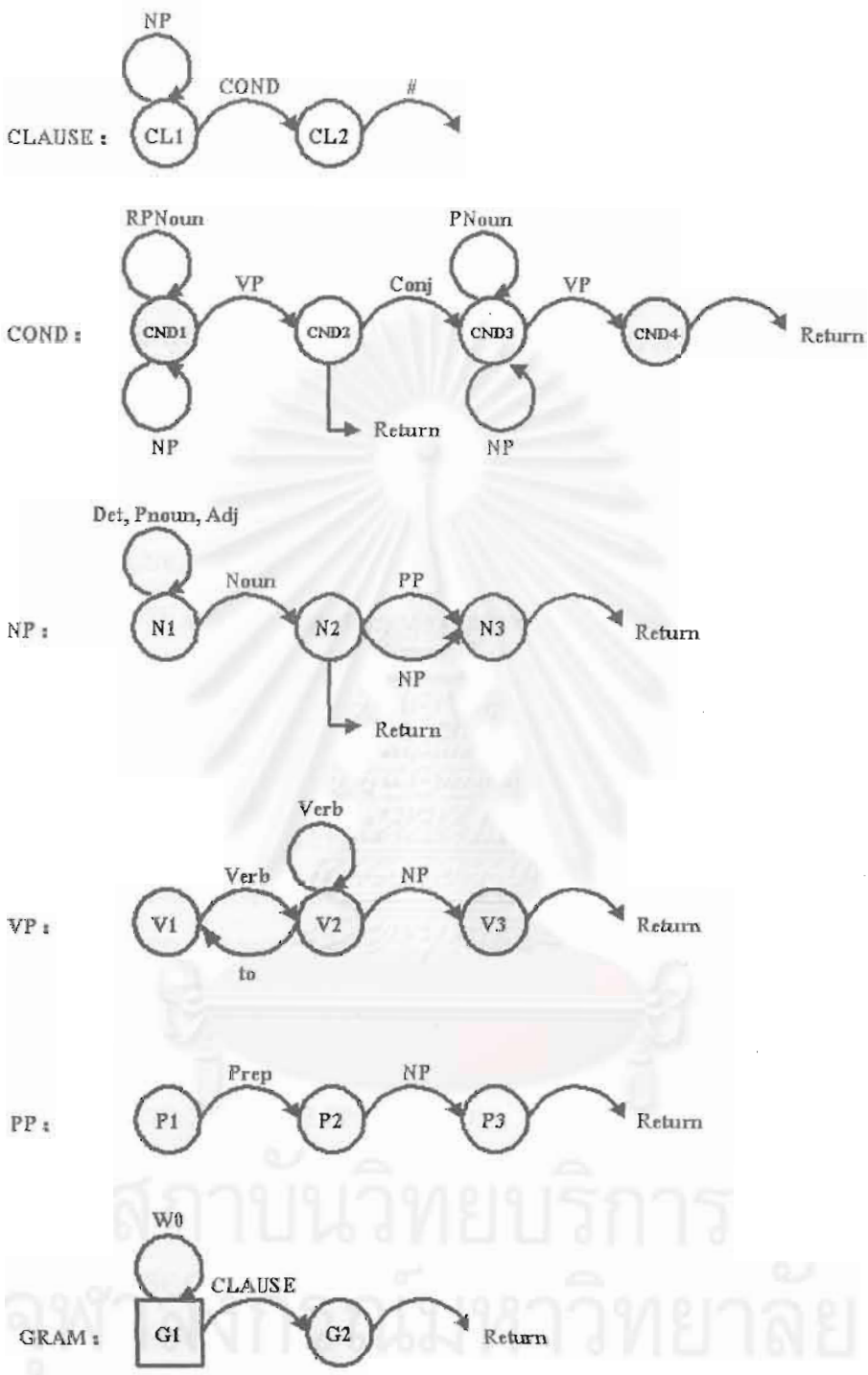
(select #dep from Dep where city = Emp.town)

ในส่วนของภาษาธรรมชาติที่ใช้ในระบบมีขอบเขตของภาษาที่รองรับจำกัดกว่ารูปแบบไวยากรณ์ของภาษาธรรมชาติที่ใช้ปกติทั่วไป อีกทั้งต้องพิจารณาจากรูปแบบไวยากรณ์ต่างๆ ที่ปรากฏในประโยคตัวอย่างทั้งประเภทของคำถามและจุดมุ่งหมายของประโยคแต่ละประเภทซึ่งผู้ใช้ระบบส่วนใหญ่ให้ความสำคัญ จึงทำการรวบรวมตัวอย่างต่างๆ เพื่อมาวิเคราะห์ให้สามารถที่จะแปลงเป็นภาษาสอบถามเชิงโครงสร้างได้มีความหมายใกล้เคียงกับความต้องการของผู้ใช้ระบบมากที่สุดและมีความสะดวกแก่ผู้ใช้ระบบด้วย ดังนั้นการวิจัยจึงได้ใช้โครงสร้างไวยากรณ์ในรูปที่ 3.6 ร่วมกับการใช้โมเดลข้อมูล(data model) ของระบบฐานข้อมูล และการวิเคราะห์ความสัมพันธ์ของตัวอย่างประโยคมาช่วยในการแปลงดังจะกล่าวถึงในส่วนถัดไป

ตารางที่ 3.5 ความหมายของคำย่อที่ใช้ในรูปที่ 3.6

คำย่อ	อธิบายความหมาย	คำย่อ	อธิบายความหมาย
CLAUSE	ฟังก์ชันclause	Pnoun	คำชนิดpronoun
COND	ฟังก์ชันcondition clause	Det	คำชนิดdetermine
NP	ฟังก์ชันnoun phrase	Conj	คำชนิดconjunction
VP	ฟังก์ชันverb phrase	CL*i*	ขั้นตอนที่ *i* ในCLAUSE
PP	ฟังก์ชันpreposition phrase	CND*i*	ขั้นตอนที่ *i* ในCOND
GRAM	ฟังก์ชันcheck grammar	N*i*	ขั้นตอนที่ *i* ในNP
Noun	คำชนิดnoun	V*i*	ขั้นตอนที่ *i* ในVP
Verb	คำชนิดverb	P*i*	ขั้นตอนที่ *i* ในPP
Adj	คำชนิดadjective	G*i*	ขั้นตอนที่ *i* ในGRAM
RPNoun	คำชนิดrelative pronoun	W0	คำชนิดไม่มีความหมาย
Prep	คำชนิดpreposition		

รูปที่ 3.6 รูปแบบไวยากรณ์



จากรูปที่ 3.6 ที่แสดงรูปแบบไวยากรณ์ที่รองรับในส่วนของขั้นตอน G1 ใน GRAM นั้น ระบบจะทำการมองข้ามคำข้อมูล(W0)ที่ไม่มีความสัมพันธ์กับระบบฐานข้อมูล และ กระทบกับการแปลความหมายของคำร้องขอ ซึ่งการวิเคราะห์ไวยากรณ์สามารถอธิบายรูปแบบได้ดังนี้คือ

NP(name of employee) = ส่ง “name of employee” ไปตรวจสอบนามวลีที่ฟังก์ชัน NP

Noun(name) -> N2 = “name” มีชนิดของไวยากรณ์เป็นคำนาม ระบบจะไปทำยังขั้นตอน N2 ต่อไป

ตัวอย่างการตรวจสอบรูปแบบไวยากรณ์

I want to know all department in bangkok

GRAM(I want to know all department in bangkok)

GRAM:G1: I เป็น W0 (ไม่มีความสัมพันธ์และไม่มีผลกระทบ) -> G1

GRAM:G1: want เป็น W0 (ไม่มีความสัมพันธ์,ไม่มีผลกระทบและคำกริยาที่ไม่มีผู้กระทำ) -> G1

GRAM:G1: to เป็น W0 -> G1

GRAM:G1: know เป็น W0 (ไม่มีความสัมพันธ์,ไม่มีผลกระทบและคำกริยาที่ไม่มีผู้กระทำ) -> G1

GRAM:G1: all เป็น W0 (ระบบไม่รองรับการคิวรี all แยกตามตาราง และการคิวรี some ต้องระบุเขตข้อมูลที่ต้องการ) -> G1

GRAM:G1: department (เป็นคำข้อมูลสำคัญชนิดคำนามอ้างถึงตารางแผนก)

GRAM: G1:CLAUSE(department in bangkok)

GRAM: G1:CLAUSE:CL1: department

GRAM: G1:CLAUSE:CL1:NP(department in bangkok)

GRAM: G1:CLAUSE:CL1:NP:N1:Noun(department) -> N2

GRAM: G1:CLAUSE:CL1:NP:N2: in (เป็นคำข้อมูลชนิดคำบุพบท)

GRAM: G1:CLAUSE:CL1:NP:N2:PP(in bangkok)

GRAM: G1:CLAUSE:CL1:NP:N2:PP:P1:Prep(in) -> P2

GRAM: G1:CLAUSE:CL1:NP:N2:PP:P2:bangkok (เป็นคำข้อมูลสำคัญชนิดคำนามที่มีค่าระบุ)

GRAM: G1:CLAUSE:CL1:NP:N2:PP:P2:NP(bangkok)

GRAM: G1:CLAUSE:CL1:NP:N2:PP:P2:NP:N1:Noun(bangkok) -> N2

GRAM: G1:CLAUSE:CL1:NP:N2:PP:P2:NP:N2: (return)

GRAM: G1:CLAUSE:CL1:NP:N2:PP:P2:NP: (OK)

GRAM: G1:CLAUSE:CL1:NP:N2:PP:P3: (return)

GRAM: G1:CLAUSE:CL1:NP:N2:PP: (OK)

GRAM: G1:CLAUSE:CL1:NP:N3: (return)

GRAM: G1:CLAUSE:CL1:NP (OK)

GRAM: G1:CLAUSE:CL1: (return)

GRAM: G1:CLAUSE (OK)

GRAM:G2: (return)

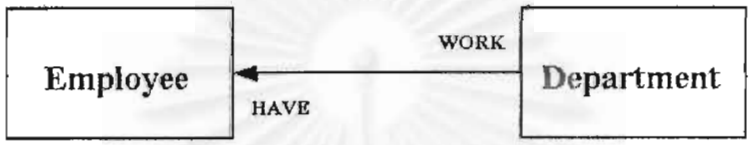
GRAM (OK)

ไวยากรณ์ถูกต้องสิ้นสุดการเก็บข้อมูลและตรวจสอบไวยากรณ์

3.3.2 การวิเคราะห์วัตถุประสงค์ของคำถามโดยใช้หลักความสัมพันธ์

การวิเคราะห์วัตถุประสงค์ของคำถามโดยใช้หลักความสัมพันธ์ ซึ่งพื้นฐานของระบบฐานข้อมูลความสัมพันธ์นั้นเริ่มจากการนำเอาเขตข้อมูล แฟ้มข้อมูล หรือวัตถุที่มีความสัมพันธ์เกี่ยวข้องซึ่งกันและกันภายใต้จุดประสงค์ร่วมกันอย่างใดอย่างหนึ่ง โดยได้มีการออกแบบและอธิบายความสัมพันธ์ระหว่างตารางของระบบฐานข้อมูลแบบ Entity and Relationship Diagram หรือ ER-Diagram ซึ่งสามารถโมเดลข้อมูลได้อย่างมีคุณภาพ ดังในรูปที่ 3.7

รูปที่ 3.7 ตัวอย่างโมเดลข้อมูล



ตัวอย่างโมเดลข้อมูล ในรูปที่ 3.7 สัญลักษณ์สี่เหลี่ยมแสดงถึง เอนทิตี(entity) คือ กลุ่มของสิ่งที่เราสนใจ(ตารางพนักงาน(*Employee*) กับ ตารางแผนก(*Department*)) มีเส้นเชื่อมแสดงความสัมพันธ์ และ ข้อความกำกับเส้นแสดงคำอธิบายความสัมพันธ์ เช่น พนักงานแต่ละคนต้องทำงานในแผนกใดแผนกหนึ่ง (*each employee must work in one department*) และ ทุกแผนกจะมีพนักงานอยู่ในแผนกอย่างน้อยหนึ่งคน (*each department have one or more employee*) เป็นต้น การใช้ข้อความกำกับเส้น หรือ แสดงคำอธิบายความสัมพันธ์นั้นควรใช้คำกริยาที่มีความหมายทั้งสองทางคือ *Employee work in Department* และ *Department worked by one or more Employee* ซึ่งใช้ *work* เป็นตัวอธิบายความสัมพันธ์ซึ่งแน่ใจได้ว่าจะไม่มีการแปลความหมายได้หลายความหมายที่แตกต่างกัน(carrying two separate meanings) การจัดวางตำแหน่งของคำอธิบายความสัมพันธ์นั้นควรวางใกล้เอนทิตีเหนือเส้นหรือใต้เส้น โดยการอธิบายในแนวตามเข็มนาฬิกาจากซ้ายไปขวาใช้คำอธิบายความสัมพันธ์ที่อยู่เหนือเส้น จากขวามาซ้ายใช้ใต้เส้น นอกจากนั้นยังสามารถได้โดยใช้วงกลมหรือวงรีแสดงถึงเขตข้อมูลต่างๆ ในระบบ เขตข้อมูลที่เป็นกุญแจหลักจะมีการขีดเส้นใต้กำกับไว้ ทำให้สามารถเห็นรายละเอียดของความสัมพันธ์ได้ลึกและกว้างมากขึ้น ทำให้การวิเคราะห์เพื่อหาคำตอบสำหรับคำถามได้เฉพาะเจาะจงถึงสิ่งที่ผู้ใช้ต้องการ

ดังนั้นเมื่อเราเพิ่มการเก็บตัวอย่างคำถามประเภทต่างๆ ทั้งแบบประโยคบอกเล่าและประโยคคำถาม แล้วหาลักษณะคำตอบที่ผู้ใช้ระบบส่วนใหญ่ต้องการทราบรวมกับความสัมพันธ์ที่ได้จากโมเดลข้อมูลของระบบฐานข้อมูลเพิ่มลงในพจนานุกรมฐานความรู้ ทำให้พจนานุกรมฐานความรู้ครอบคลุมคำตอบของปัญหาแต่ละประเภทมากขึ้น ในการแปลภาษาธรรมชาติให้เป็นภาษาสอบถามเชิงโครงสร้างยังใช้คำแวดล้อมมาเป็นตัวช่วยในการแปลความสัมพันธ์และเป้าหมายในรูปประโยคเพื่อนำไปหาคำตอบโดยใช้พจนานุกรมฐานความรู้ เช่นจากตัวอย่างโมเดลข้อมูลที่เราได้

ความสัมพันธ์ว่า *Each employee must work in one department* และ *Department worked by one or more Employee* ตัวอย่างรูปแบบคำถามคือ *Who work in accounting department* การวิเคราะห์คำตอบของคำถามจะได้ข้อมูลดังนี้ ประธานคือ *Who* กริยาคือ *work* กรรมคือ *department* คำขยายของกรรมคือ *accounting* อาจต้องการทราบรายละเอียดทั้งหมดของพนักงานหรืออยากทราบเพียงแค่เขตข้อมูลที่เป็นชื่อพนักงานเท่านั้นก็เพียงพอ จึงขึ้นอยู่กับความต้องการของผู้ใช้ระบบส่วนใหญ่ว่าต้องการแบบใด ส่วนในการแปลนามวลีจะขึ้นกับคำบุพบทและตำแหน่งของคำจะช่วยแปลความหมายของคำนามนั้น

ปัญหาในการวิเคราะห์ประโยคและวิธีที่ใช้ในระบบเพื่อแก้ไข

1) คำที่ไม่มีอยู่ในพจนานุกรมฐานความรู้ เนื่องจากคำส่วนมากที่ปรากฏและไม่พบในพจนานุกรมฐานความรู้มักจะเป็นคำประเภท ชื่อของสิ่งต่างๆ เช่น ชื่อพนักงาน ชื่อแผนก หรือ ชื่อวัตถุดิบ เป็นต้น ดังนั้นระบบจะกำหนดให้คำที่ไม่มีอยู่ในพจนานุกรมฐานความรู้เป็นค่าคำนามทั่วไปคือ *name* แล้วระบบจะไประบุเฉพาะเจาะจงลงไปในการแปลโดยอาศัยคำแวดล้อมของคำนามนั้น

2) คำที่เป็นค่าได้ในหลายเขตข้อมูล คำประเภทนี้ถ้ามีความหมายไปในลักษณะใดลักษณะหนึ่งพจนานุกรมฐานความรู้จะเจาะจงไปที่คำทั่วไปลักษณะนั้น เช่น London เป็นคำทั่วไปประเภทสถานที่ก็จะระบุได้ว่าเป็น *address* แต่ถ้าสามารถเป็นชื่อคนได้ด้วย ระบบอาจระบุลักษณะคำทั่วไปตามการใช้ส่วนใหญ่โดยถ้าผู้ใช้ต้องการระบุเป็นชื่อพนักงานจะต้องเจาะจงเข้าไปที่รูปประโยคเพื่อนำไปสู่ความหมายที่ต้องการด้วยตนเอง

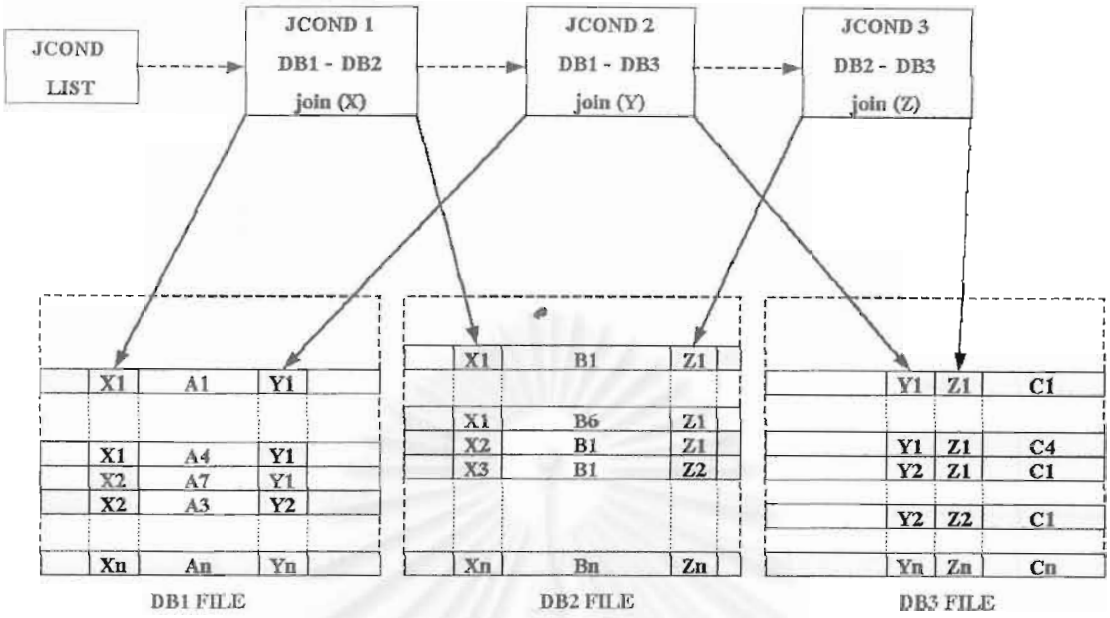
3) คำที่มีความหมายทั่วไปไม่เฉพาะเจาะจง เช่น *name, number, id, address* เป็นต้น คำประเภทนี้ระบบจะต้องอาศัยคำแวดล้อมช่วยในการระบุความชัดเจนของคำ ถ้าคำที่มาก่อนระบุความหมายชัดเจนก็จะใช้คำนั้นเป็นคำช่วยเหลือดังนั้นตำแหน่งของคำมีความหมายในการแปลถ้ารูปประโยคไม่ชัดเจนเช่น ถ้า *MIS* เป็นชื่อแผนก , *Bangkok* เป็น สถานที่, salary เป็นเงินเดือนของพนักงาน ผู้ใช้ให้คำร้องขอมาเป็น *MIS, Bangkok, salary > 25000* จะได้ *Bangkok* เป็นที่อยู่ของแผนก แต่ถ้าคำร้องขอเป็น *MIS, salary > 25000, Bangkok* จะได้ *Bangkok* เป็นที่อยู่ของพนักงาน เป็นต้น

4) คำที่มีชนิดไวยากรณ์มากกว่าหนึ่ง เช่น name, work เป็นต้น คำประเภทนี้ระบบจะเลือกชนิดของไวยากรณ์โดยอาศัยชนิดไวยากรณ์ของคำแวดล้อมเพื่อช่วยในการตัดสินใจร่วมกับกฎเกณฑ์ของไวยากรณ์ เช่น employee's name is ton จะได้ name เป็น ชนิดคำนาม แต่ถ้า employee name ton จะได้ name เป็น ชนิดคำกริยา เป็นต้น

3.4 การผสานผลลัพธ์ของข้อมูลเพื่อนำเสนอ

เมื่อคำร้องขอของผู้ใช้ระบบมีการเชื่อมโยงและอ้างถึงเขตข้อมูลข้ามระหว่างฐานข้อมูลของทั้งชุดเงื่อนไขที่ได้จากความสัมพันธ์โดยตรง ($(D(A) \cap D(B)) = D(AB) != \varnothing$ ซึ่ง Aและ B อยู่ต่างฐานข้อมูลกัน) และ ความสัมพันธ์โดยอ้อม($D(A) \cap D(B) = \varnothing$, $D(A) \cap D(C) = D(AC) != \varnothing$ และ $D(B) \cap D(C) = D(BC) != \varnothing$ แล้ว $D(AC) \cap D(BC) = \varnothing$ แต่มี $e \in C$ ซึ่ง e มีความสัมพันธ์กับสมาชิกใน $D(AC)$ และ $D(BC)$ ดังนั้นจะได้ว่า A และ B มีความสัมพันธ์กัน ซึ่ง Aและ B อยู่ต่างฐานข้อมูลกัน) ระบบจะทำงานโดยการแยกคิวรีคำร้องขอที่อยู่ในลักษณะภาษาสอบถามเชิงโครงสร้างแล้วนั้นตามแต่ละฐานข้อมูลที่อ้างอิง หลังจากการทำคิวรีเสร็จสิ้นจะมาถึงการทำงานในส่วนการผสานผลลัพธ์ที่ได้จากการคิวรีทั้งหมดซึ่งอยู่ในรูปแฟ้มข้อมูล มีการเรียงลำดับเขตข้อมูลที่เป็นตัวเชื่อมโยงด้วย การออกแบบการผสานผลลัพธ์ จะใช้วิธีอ้างอิงถึงระเบียนผลลัพธ์ที่ต้องการโดยตรงจากแฟ้มข้อมูลโดยอ่านขึ้นมาทีละระเบียนข้อมูล และเปรียบเทียบเงื่อนไขชุดเขตข้อมูลที่เป็นตัวเชื่อมโยงระหว่างแฟ้มข้อมูลผลลัพธ์ต่างๆ ถ้าตรงตามเงื่อนไขของทุกชุดเขตข้อมูลที่เป็นตัวเชื่อมจะทำการเขียนระเบียนที่อ่านนั้นจากแฟ้มข้อมูลทั้งหมด ลงในแฟ้มข้อมูลการผสานผลลัพธ์ แล้วจะทำการเลื่อนจุดที่อ้างอิงระเบียนที่อยู่บนแฟ้มข้อมูลไปยังระเบียนถัดไป การเลื่อนจุดที่อ้างอิงระเบียนข้อมูลจะกระทำกับแฟ้มข้อมูลที่มีลำดับสมมติการเรียงจากน้อยไปมากหรือมากไปน้อยก็ได้ ในที่นี้จะใช้จากมากไปน้อย ถ้าไม่ตรงตามเงื่อนไขของทุกชุดเขตข้อมูลที่เป็นตัวเชื่อมจะทำการเลื่อนจุดอ้างอิงระเบียนที่มีเขตข้อมูลที่เป็นตัวเชื่อมที่ผิดเงื่อนไขที่มีค่าเปรียบเทียบมากกว่าคู่เขตข้อมูลของทุกชุดเงื่อนไขไปเริ่มที่จุดเริ่มต้นของแฟ้มข้อมูล แล้วเลื่อนจุดอ้างอิงระเบียนของแฟ้มข้อมูลที่มีสมมติการเรียงที่น้อยกว่าแฟ้มข้อมูลที่มีเขตข้อมูลที่เป็นตัวเชื่อมที่ผิดเงื่อนไขไปเริ่มที่ระเบียนถัดไปของแฟ้มข้อมูลนั้น ทำจนเสร็จสิ้นแฟ้มข้อมูลผลลัพธ์ต่างๆ ก็จะได้แฟ้มการผสานผลลัพธ์ที่ต้องการ

รูปที่ 3.8 แผนภาพตัวอย่างการผสานผลลัพธ์



จากรูปที่ 3.8 แสดงแผนภาพตัวอย่างการผสานผลลัพธ์ซึ่งอธิบายการทำงานของการผสานผลลัพธ์ โดยคำย่อที่ปรากฏในรูป อธิบายได้ดังนี้

JCOND LIST คือ ชุดเงื่อนไขของเขตข้อมูลที่เป็นตัวเชื่อมโยงในการผสานผลลัพธ์

JCOND1 DB1 - DB2 join (*X*) คือ ชุดเงื่อนไขที่ 1 ซึ่งเชื่อมโยงระหว่าง แฟ้มข้อมูลผลลัพธ์ของฐานข้อมูล *DB1* กับ แฟ้มข้อมูลผลลัพธ์ของฐานข้อมูล *DB2* ด้วยเขตข้อมูล *X*

JCOND2 DB1 - DB3 join (*Y*) คือ ชุดเงื่อนไขที่ 2 ซึ่งเชื่อมโยงระหว่าง แฟ้มข้อมูลผลลัพธ์ของฐานข้อมูล *DB1* กับ แฟ้มข้อมูลผลลัพธ์ของฐานข้อมูล *DB3* ด้วยเขตข้อมูล *Y*

JCOND3 DB2 - DB3 join (*Z*) คือ ชุดเงื่อนไขที่ 3 ซึ่งเชื่อมโยงระหว่าง แฟ้มข้อมูลผลลัพธ์ของฐานข้อมูล *DB2* กับ แฟ้มข้อมูลผลลัพธ์ของฐานข้อมูล *DB3* ด้วยเขตข้อมูล *Z*

X1,...,Xn คือ ค่าของเขตข้อมูล *X*

Y1,...,Yn คือ ค่าของเขตข้อมูล *Y*

Z1....,Zn คือ ค่าของเขตข้อมูล *Z*

A1,...,An คือ ค่าของเขตข้อมูล *A*

B1,...,Bn คือ ค่าของเขตข้อมูล *B*

C1.....Cn คือ ค่าของเขตข้อมูล *C*

การทำงานจะเริ่มที่ต้นที่จุดเริ่มของแฟ้มข้อมูลทุกชุดเงื่อนไข จนถึงสิ้นสุดทุกแฟ้มข้อมูลได้ผลลัพธ์ดังตารางที่ 3.6 ข้างล่าง

ตารางที่ 3.6 ตารางแสดงผลที่ได้การผสานผลลัพธ์

ลำดับ	*X*	*A*	*Y*	*X*	*B*	*Z*	*Y*	*Z*	*C*
1	*X1*	*A1*	*Y1*	*X1*	*B1*	*Z1*	*Y1*	*Z1*	*C1*
2	*X1*	*A1*	*Y1*	*X1*	*B1*	*Z1*	*Y1*	*Z1*	*C2*
3	*X1*	*A1*	*Y1*	*X1*	*B1*	*Z1*	*Y1*	*Z1*	*C3*
4	*X1*	*A1*	*Y1*	*X1*	*B1*	*Z1*	*Y1*	*Z1*	*C4*
5	*X1*	*A1*	*Y1*	*X1*	*B2*	*Z1*	*Y1*	*Z1*	*C1*
.	*X1*	*A4*	*Y1*	*X1*	*B6*	*Z1*	*Y1*	*Z1*	*C4*
.	*X2*	*A7*	*Y1*	*X2*	*B1*	*Z1*	*Y1*	*Z1*	*C1*
.	*X2*	*A7*	*Y1*	*X2*	*B1*	*Z1*	*Y1*	*Z1*	*C4*
.	*X2*	*A3*	*Y2*	*X2*	*B1*	*Z1*	*Y2*	*Z1*	*C1*
N	*Xn*	*An*	*Yn*	*Xn*	*Bn*	*Zn*	*Yn*	*Zn*	*Cn*

หลังจากที่ทำการผสานผลลัพธ์เสร็จเรียบร้อยระบบจะจัดการจัดรูปแบบและทำการประเมินผลลัพธ์หาจำนวนระเบียน ค่าผลรวม ค่าเฉลี่ย ค่ามากสุด และค่าน้อยสุด ของเขตข้อมูลที่เป็นผลลัพธ์ที่สามารถทำการประเมินได้

บทที่ 4

การออกแบบระบบ

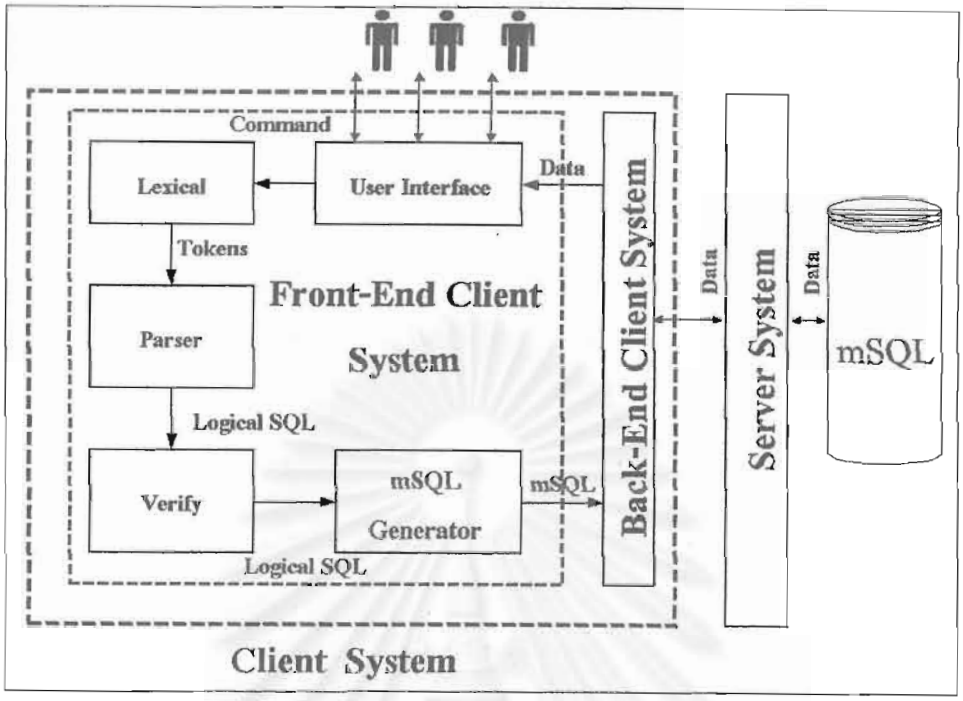
ระบบโปรแกรมที่ทำการพัฒนาจากแนวความคิดเครือข่ายแบบระบบผู้ใช้บริการ/ผู้ให้บริการ การติดต่อระหว่างระบบผู้ให้บริการกับระบบผู้ใช้บริการจะใช้ซอคเก็ตผ่านโปรโตคอลทีซีพี/ไอพี ระบบโปรแกรมจะมีการทำงานในส่วนต่างๆ ซึ่งสามารถจำแนกหน้าที่ได้ดังต่อไปนี้

1) การติดต่อและแสดงผลกับผู้ใช้บริการ
2) การตัดคำ
3) การวิเคราะห์คำร้องขอ
4) การตรวจสอบคำร้องขอ
5) การแปลงคำร้องขอให้เป็นภาษาสอบถามเชิงโครงสร้าง mSQL
6) การแยกภาษาสอบถามเชิงโครงสร้าง mSQL ตามฐานข้อมูล
7) การผสานผลลัพธ์
8) จัดรูปแบบเพื่อนำเสนอ
9) การสรุปข้อมูลที่ได้จากผลลัพธ์
10) การติดต่อประสานงานกับผู้ให้บริการ
11) การติดต่อประสานงานกับผู้ใช้บริการ
12) การติดต่อประสานงานกับส่วนจัดการระบบฐานข้อมูลmSQL

ระบบโปรแกรมแบ่งออกตามลักษณะของการออกแบบระบบได้เป็น 2 ส่วนระบบย่อยดังรูปที่ 4.1 คือ

1) โปรแกรมระบบผู้ใช้บริการ(client system)
2) โปรแกรมระบบผู้ให้บริการ(server system)

รูปที่ 4.1 ภาพรวมของระบบ



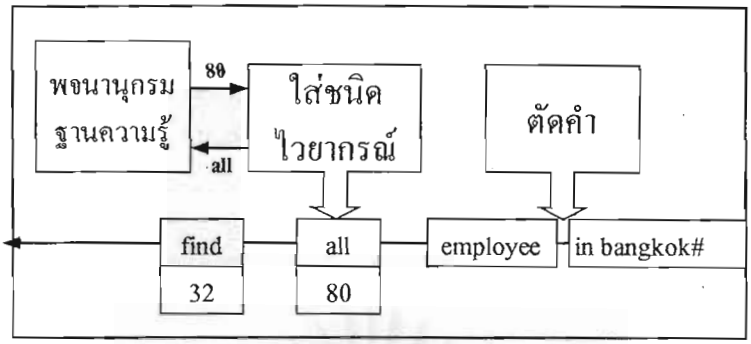
โดยแต่ละระบบย่อยที่ได้พัฒนาขึ้นมีหน้าที่และลักษณะการทำงานดังนี้

4.1 โปรแกรมระบบผู้ใช้บริการ

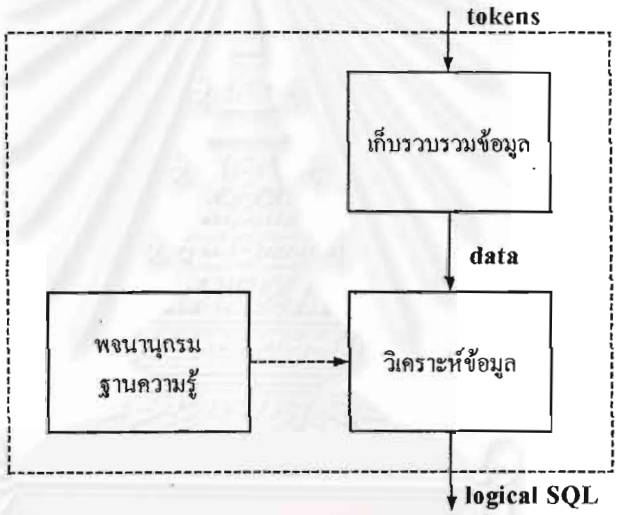
โปรแกรมระบบผู้ใช้บริการจะมีหน้าที่ในการทำงานดังนี้

1) การติดต่อและแสดงผลกับผู้ใช้บริการ
2) การตัดคำ
3) การวิเคราะห์คำร้องขอ
4) การตรวจสอบคำร้องขอ
5) การแปลงคำร้องขอให้เป็นภาษาสอบถามเชิงโครงสร้าง mSQL
6) การแยกภาษาสอบถามเชิงโครงสร้าง mSQL ตามฐานข้อมูล
7) การผสานผลลัพธ์
8) จัดรูปแบบเพื่อนำเสนอ
9) การสรุปข้อมูลที่ได้จากผลลัพธ์
10) การติดต่อประสานงานกับผู้ให้บริการ

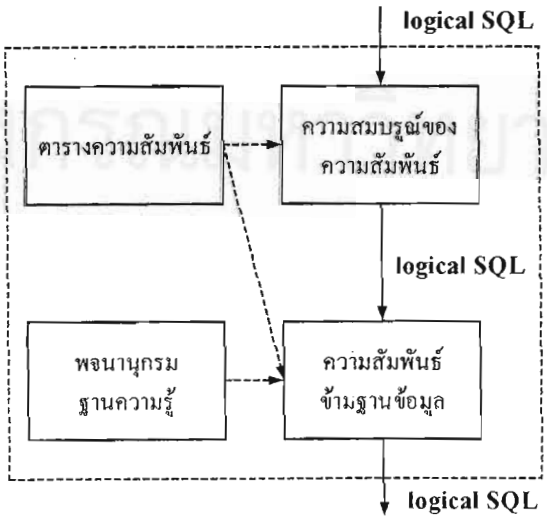
รูปที่ 4.2 แสดงส่วนของการตัดคำ(lexical)



รูปที่ 4.3 แสดงส่วนของการรวบรวมและวิเคราะห์(parser)



รูปที่ 4.4 แสดงส่วนของการตรวจสอบ(verify)



ในส่วนของระบบผู้ใช้บริการสามารถแบ่งตามหน้าที่ได้เป็น 2 ส่วนคือ โปรแกรมส่วนที่ติดต่อกับผู้ใช้บริการ กับ โปรแกรมส่วนที่ติดต่อกับผู้ให้บริการ ดังมีรายละเอียดต่อไปนี้

4.1.1 โปรแกรมส่วนที่ติดต่อกับผู้ใช้บริการ

โปรแกรมส่วนที่ติดต่อกับผู้ใช้บริการเป็นส่วนหน้าของระบบผู้ใช้บริการทำหน้าที่คอยควบคุมลำดับการทำงานต่างๆ ของระบบให้ทำตามคำสั่งของผู้ใช้บริการ ในการให้รายละเอียดโครงสร้างข้อมูลของฐานข้อมูล ตาราง เขตข้อมูล และชนิดของเขตข้อมูล ที่ผู้ใช้ต้องการทราบ เมื่อผู้ใช้มีคำร้องขอเพื่อสอบถามค้นคืนข้อมูลที่ต้องการจากระบบ ระบบจะส่งคำร้องขอไปยังส่วนต่างๆ ของระบบเพื่อทำการตัดคำ วิเคราะห์ แปลง ตรวจสอบ และนำเสนอ ดังนี้

ในส่วนการตัดคำ(lexical) ดังรูปที่4.2 คำร้องขอจะถูกแตกเป็นคำข้อมูลกับชนิดของคำข้อมูลนั้นที่ได้กำหนดให้ไว้ในพจนานุกรมฐานความรู้ ถ้าไม่พบชนิดของคำข้อมูลดังกล่าวระบบจะปฏิบัติตามกฎเกณฑ์ที่ได้กำหนดไว้ (ในระบบนี้กำหนดให้เป็นชนิดคำนามทั่วไปประเภท name) จากนั้นจะส่งผ่านไปยังส่วนรวบรวมข้อมูลและวิเคราะห์วัตถุประสงค์(parser)

ส่วนรวบรวมข้อมูลและวิเคราะห์วัตถุประสงค์(parser) วิเคราะห์คำร้องขอโดยใช้พจนานุกรมฐานความรู้ในส่วนอธิบายความสัมพันธ์เชิงวัตถุของระบบฐานข้อมูล เพื่อทำการแปลงจากภาษาธรรมชาติซึ่งเป็นรูปแบบที่ผู้ใช้บริการใช้ ให้เป็นภาษาเชิงโครงสร้างที่เป็นตรรกะ(logical sql) แสดงในรูปที่4.3

ส่วนของการตรวจสอบ(verify) ดังรูปที่4.4 จะทำการตรวจสอบความสัมพันธ์และความสมบูรณ์ระหว่างคำถามกับคำตอบที่ผู้ใช้ต้องการที่อยู่ในรูปภาษาเชิงโครงสร้างที่เป็นตรรกะ รวมถึงการตรวจสอบคำร้องขอที่อยู่ในลักษณะอ้างอิงฐานข้อมูลมากกว่าหนึ่ง จะทำการแยกคำร้องขอตามฐานข้อมูลกับชุดเงื่อนไขเชื่อมโยงระหว่างฐานข้อมูล การทำงานในส่วนนี้จะใช้ตารางความสัมพันธ์องค์ประกอบของฐานข้อมูลมาช่วยในการตรวจสอบ ถ้าข้อมูลในรูปภาษาเชิงโครงสร้างที่เป็นตรรกะผ่านการตรวจสอบข้อมูลดังกล่าว จะถูกนำไปใช้สร้างภาษาสอบถามเชิงโครงสร้างmSQL (mSQL generator) แล้วทำการส่งผลลัพธ์ที่ได้ไปยังโปรแกรมส่วนที่ติดต่อกับระบบผู้ให้บริการเพื่อทำงานประมวลผลต่อไป

นำผลลัพธ์ที่ได้จากการประมวลผลนำเสนอแก่ผู้ใช้บริการ รวมถึงข้อผิดพลาดต่างๆ ที่เกิดขึ้นในการทำงานตามคำร้องขอดังกล่าว เพื่อยังผลให้ผู้ใช้บริการมีความสะดวกและเข้าใจการทำงานของระบบได้ง่ายในการใช้งานระบบตามความต้องการ

4.1.2 โปรแกรมส่วนที่ติดต่อกับระบบผู้ให้บริการ

มีหน้าที่รับคำสั่งจากส่วนที่ติดต่อกับผู้ใช้บริการ แล้วทำการติดต่อกับระบบผู้ให้บริการ สามารถจำแนกหน้าที่การกำหนดชนิดติดต่อกับระบบผู้ให้บริการของระบบผู้ใช้บริการได้ดังต่อไปนี้

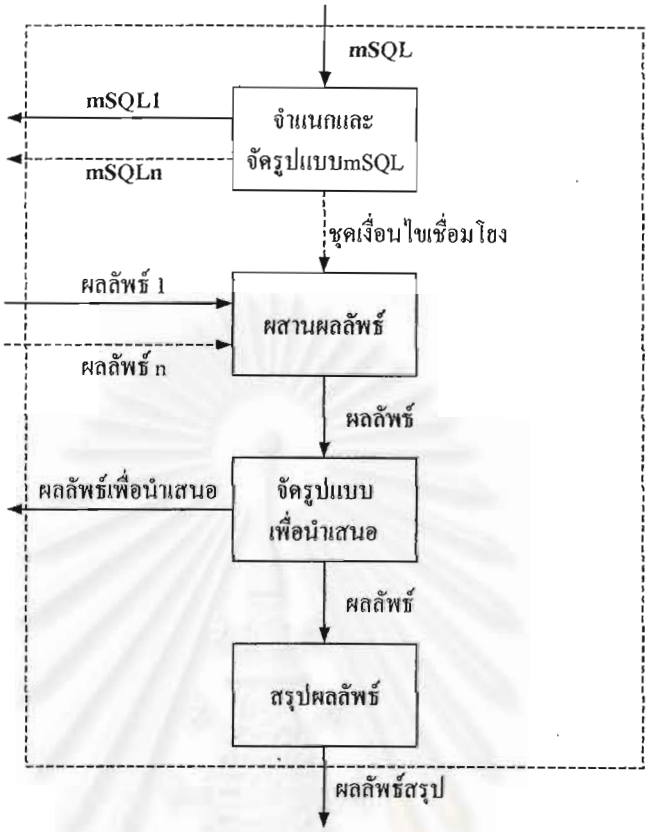
1) ผู้ใช้ต้องการเรียกดูโครงสร้างข้อมูลของระบบฐานข้อมูล
2) คำร้องขอของผู้ใช้อ้างอิงฐานข้อมูลเดียว
3) มีการอ้างอิงหลายฐานข้อมูลในความต้องการของผู้ใช้

รูปที่ 4.5 แสดงการทำงานของระบบเมื่อทำการติดต่อและจัดรูปแบบคำร้องขอตามแต่ชนิดที่กล่าวไปแล้วให้กับระบบผู้ให้บริการ โดยชนิดของคำร้องขอที่เป็นการเรียกดูข้อมูลโครงสร้างกับการสอบถามค้นคืนข้อมูลที่อ้างอิงฐานข้อมูลเดียว เมื่อระบบผู้ให้บริการส่งผลลัพธ์กลับมาให้จะทำการจัดรูปแบบใหม่เพื่อนำเสนอ ส่วนในชนิดรูปแบบที่อ้างอิงหลายฐานข้อมูลนั้นซึ่งต้องทำการจำแนกตามฐานข้อมูลก่อนเพื่อสามารถติดต่อกับฐานข้อมูลที่จำแนกได้นั้นถูกต้องตรงความต้องการแล้วจึงส่งข้อมูลที่จำแนกแล้วไปยังระบบผู้ให้บริการ พร้อมทั้งทำการเก็บข้อมูลเงื่อนไขในการเชื่อมคำร้องขอที่ทำการจำแนกนั้นไว้ด้วย เมื่อระบบผู้ให้บริการส่งผลลัพธ์กลับมาจะทำการผสานผลลัพธ์ที่อ้างอิงหลายฐานข้อมูลตามข้อมูลเงื่อนไขของตัวเชื่อมที่เก็บไว้เพื่อให้มีความสอดคล้องกันของข้อมูลที่ผู้ใช้ต้องการ จากนั้นจะทำการจัดรูปแบบเพื่อนำเสนอไปยังส่วนหน้าของระบบผู้ใช้บริการ อีกทั้งยังทำการรวบรวมข้อมูลของผลลัพธ์ที่ได้ไว้ด้วยเช่น จำนวนระเบียนข้อมูล ค่าเฉลี่ย ค่ามากที่สุด และค่าน้อยที่สุดของแต่ละเขตข้อมูลที่มีชนิดของเขตข้อมูลเป็นตัวเลข

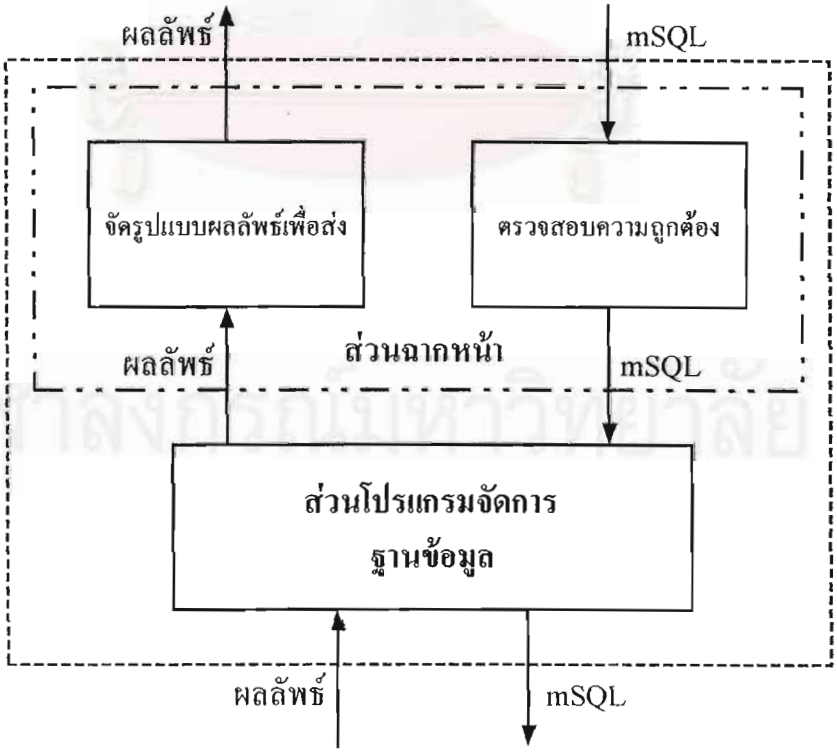
4.2 โปรแกรมระบบผู้ให้บริการ

โปรแกรมระบบผู้ให้บริการจะมีหน้าที่ 2 อย่างคือการติดต่อประสานงานกับผู้ใช้บริการและการติดต่อประสานงานกับส่วนจัดการระบบฐานข้อมูลmSQL ซึ่งจะแบ่งเป็น 2 ส่วนเพื่อรับผิดชอบหน้าที่ดังกล่าวคือ ส่วนฉากหน้าที่มีหน้าที่คอยรับการติดต่อจากระบบผู้ใช้บริการ กับ ส่วนโปรแกรมจัดการฐานข้อมูล ดังในรูปที่ 4.6 และมีรายละเอียดดังต่อไปนี้

รูปที่ 4.5 แสดงโปรแกรมส่วนที่ติดต่อกับผู้ให้บริการ



รูปที่ 4.6 แสดงโปรแกรมระบบผู้ให้บริการ



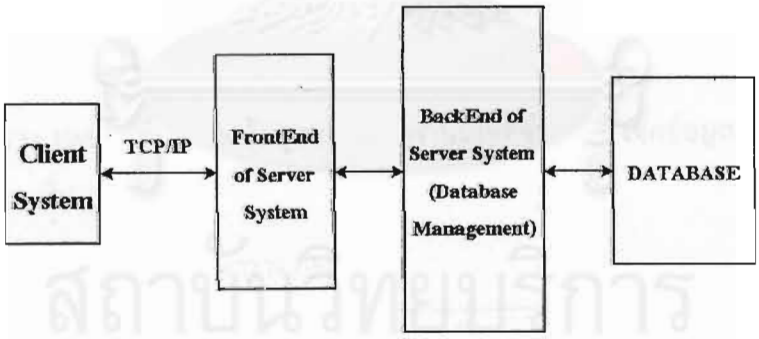
4.2.1 ส่วนฉากหน้าที่มีหน้าที่คอยรับการติดต่อจากระบ บผู้ใช้บริการ

ส่วนฉากหน้าจะทำงานอยู่ตลอดเวลาเพื่อรอการติดต่อจากระบบผู้ใช้บริการ เมื่อระบบผู้ใช้บริการได้ติดต่อเข้ามาจะทำการจำลองตัวเองเพื่อทำงานที่ได้รับมาส่วนหนึ่ง อีกส่วนยังสามารถรองรับการติดต่อของระบบผู้ใช้บริการอื่นๆ เพราะฉะนั้นระบบผู้ให้บริการสามารถให้บริการหลายรายได้พร้อมกัน การทำงานตามชนิดรูปแบบที่ได้รับมานั้นระบบผู้ให้บริการจะทำการตรวจสอบข้อมูลคิวรีที่ส่งมาอีกครั้งว่ามีความสมบูรณ์และถูกต้องหรือไม่ เมื่อพบว่ารูปแบบไม่ถูกต้องหรือมีข้อผิดพลาดเกิดขึ้นจะทำการส่งรายงานความผิดพลาดดังกล่าวกับไปยังระบบผู้ให้บริการแล้วยกเลิกช่องทางที่ทำการติดต่อนั้นลง แต่ถ้าพบว่าข้อมูลคำร้องขอนั้นถูกต้องก็จะทำการจัดเรียงข้อมูลดังกล่าวให้อยู่ในรูปแบบที่ระบบฐานข้อมูลยอมรับ แล้วทำการติดต่อกับโปรแกรมจัดการฐานข้อมูลซึ่งอยู่ในส่วนหลังเพื่อให้ทำการหาผลลัพธ์ที่ได้จากคิวรีนั้นๆ ต่อไป

4.2.2 ส่วนโปรแกรมจัดการฐานข้อมูล

ส่วนโปรแกรมจัดการฐานข้อมูลจะใช้โปรแกรมที่ระบบฐานข้อมูล mSQL ให้มาคือ msql2d ซึ่งมีหน้าที่คือ เมื่อได้รับการติดต่อจากโปรแกรมฉากหน้าแล้วจะรับคิวรีซึ่งถูกส่งมาเพื่อหาผลลัพธ์ของคิวรีจากระบบฐานข้อมูลแล้วทำการส่งผลลัพธ์ที่ได้กลับไปให้โปรแกรมฉากหน้า

รูปที่ 4.7 การทำงานของระบบ



บทที่ 5

การทดสอบระบบ

การทดสอบระบบจะมีขั้นตอนการเตรียมระบบเพื่อใช้ในการทดสอบ 3 ขั้นตอน ได้แก่

1) การสร้างระบบฐานข้อมูล

2) การสร้างตารางความสัมพันธ์องค์ประกอบของระบบฐานข้อมูล

3) การสร้างพจนานุกรมฐานความรู้

และมีการอธิบายขั้นตอนการวิเคราะห์และแปลตัวอย่าง รวมถึงตัวอย่างทั้งหมดที่ใช้ทดสอบการสร้างระบบเพื่อใช้ทดสอบและการวิเคราะห์มีรายละเอียดของแต่ละขั้นตอนดังต่อไปนี้

5.1 การสร้างระบบฐานข้อมูล

ตัวอย่างของระบบฐานข้อมูลที่ใช้ในการอธิบายการทำงานประกอบด้วย 2 ฐานข้อมูล ได้แก่ ฐานข้อมูล TEST และ ฐานข้อมูล DB2 ซึ่งมีรายละเอียดดังนี้

5.1.1 ฐานข้อมูล TEST มี 3 ตารางคือ ตาราง EMP ตาราง DEP และตาราง ANC โดยมีรายละเอียดของแต่ละตารางดังนี้

1) ตาราง EMP เป็นตารางข้อมูลของพนักงานมีเขตข้อมูล 8 เขตข้อมูล มีรายละเอียดของเขตข้อมูลดังตารางที่ 5.1

ตารางที่ 5.1 รายละเอียดเขตข้อมูลตาราง EMP

เขตข้อมูล	รายละเอียด	เขตข้อมูล	รายละเอียด
emp_id	รหัสพนักงาน	job	ตำแหน่งของพนักงาน
ename	ชื่อพนักงาน	sal	เงินเดือนของพนักงาน
dep_id	รหัสแผนกที่พนักงานทำงาน	comm	คอมมิสชั่นของพนักงาน
age	อายุของพนักงาน	town	เมืองที่พนักงานอาศัย

2) ตาราง DEP เป็นตารางข้อมูลของแผนกมีเขตข้อมูล 5 เขตข้อมูล มีรายละเอียดของเขตข้อมูลดังตารางที่ 5.2

ตารางที่ 5.2 รายละเอียดเขตข้อมูลตาราง DEP

เขตข้อมูล	รายละเอียด
dep_id	รหัสแผนก
dname	ชื่อแผนก
budget	งบประมาณของแผนก
size	ระดับขนาดของแผนก
city	เมืองที่แผนกตั้งอยู่

3) ตาราง ANC เป็นตารางข้อมูลประวัติการทำงานของพนักงาน มีเขตข้อมูล 3 เขตข้อมูล มีรายละเอียดของเขตข้อมูลดังตารางที่ 5.3

ตารางที่ 5.3 รายละเอียดเขตข้อมูลตาราง ANC

เขตข้อมูล	รายละเอียด
emp_id	รหัสพนักงาน
dep_id	รหัสแผนก
year	ปีที่เริ่มทำงาน

5.1.2 ฐานข้อมูล DB2 มี 2 ตารางคือ ตาราง SON และ ตาราง ADDR โดยมีรายละเอียดของแต่ละตารางดังนี้

1) ตาราง SON เป็นตารางข้อมูลบุตรของพนักงาน มีเขตข้อมูล 3 เขตข้อมูล มีรายละเอียดของเขตข้อมูลดังตารางที่ 5.4

ตารางที่ 5.4 รายละเอียดเขตข้อมูลตาราง SON

เขตข้อมูล	รายละเอียด
emp_id	รหัสพนักงาน
son_id	รหัสบุตรของพนักงาน
sname	ชื่อบุตรพนักงาน

2) ตาราง ADDR เป็นตารางที่อยู่ของบุตรพนักงาน มีเขตข้อมูล 2 เขตข้อมูล มีรายละเอียดของเขตข้อมูลดังตารางที่ 5.5

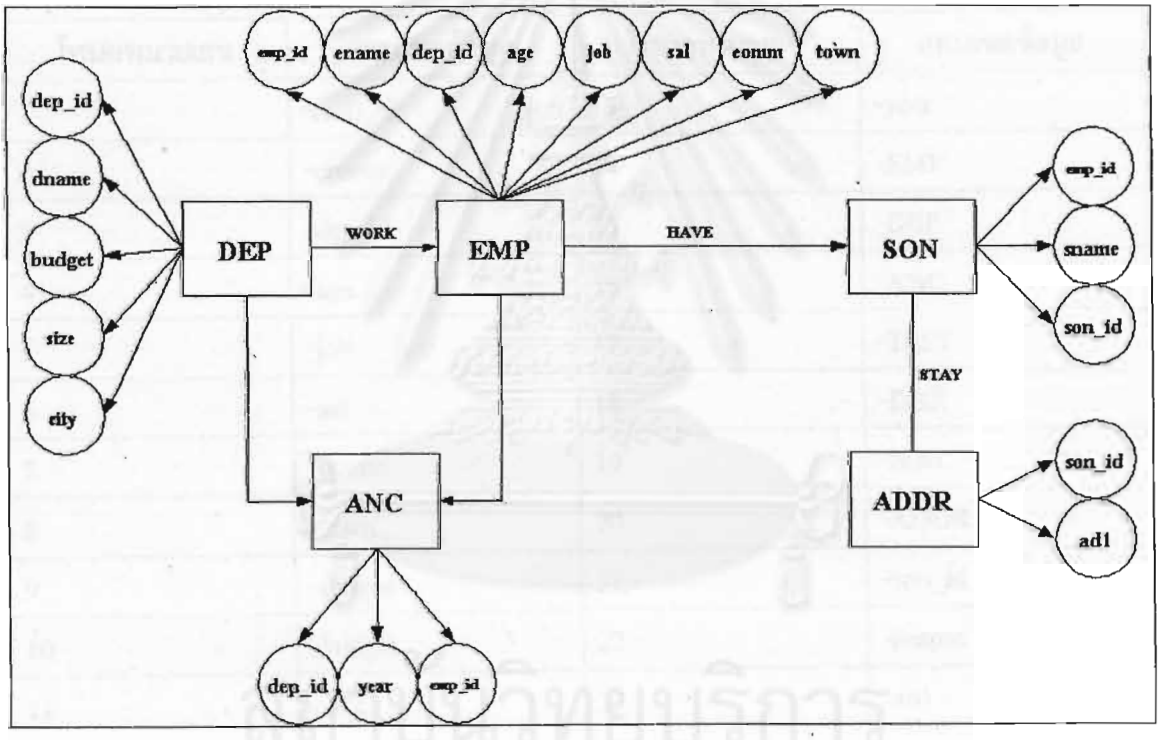
ตารางที่ 5.5 รายละเอียดเขตข้อมูลตาราง ADDR

เขตข้อมูล	รายละเอียด
son_id	รหัสบุตรของพนักงาน
ad1	ที่อยู่ของบุตร

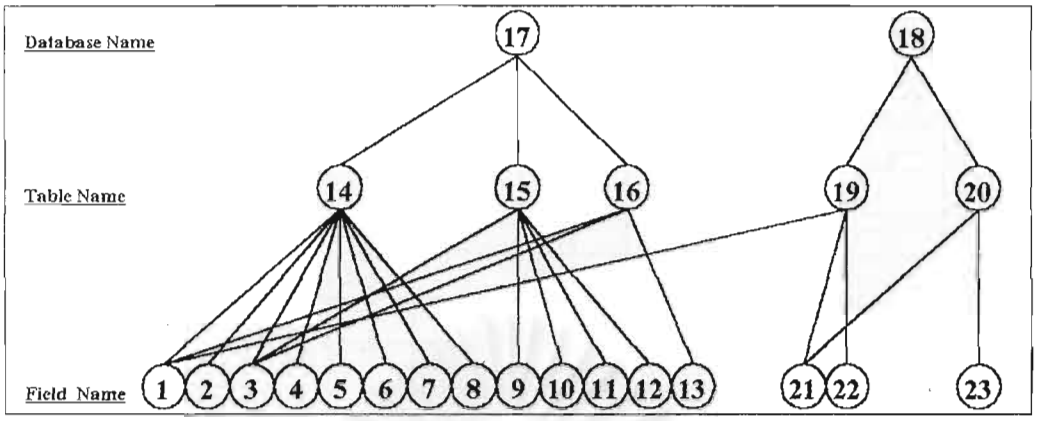
5.2 การสร้างองค์ประกอบความสัมพันธ์ของระบบฐา นข้อมูล

เราจะอธิบายความสัมพันธ์ของตัวอย่างระบบฐานข้อมูลโดยใช้ ER-diagram ดังรูปที่ 5.1 แล้วนำมาแสดงระดับความสัมพันธ์ของโครงสร้างต่างๆ โดยใช้กราฟในรูปที่ 5.2

รูปที่ 5.1 ER-diagram แสดงความสัมพันธ์ของระบบฐานข้อมูล



รูปที่ 5.2 กราฟแสดงความสัมพันธ์ของระบบฐานข้อมูล



ตารางที่ 5.6 คำอธิบายโหนดตัวเลขในรูปที่ 5.2

โหนดหมายเลข	แทนเขตข้อมูล	โหนดหมายเลข	แทนเขตข้อมูล
1	-emp_id	13	-year
2	-ename	14	-EMP
3	-dep_id	15	-DEP
4	-age	16	-ANC
5	-job	17	-TEST
6	-sal	18	-DB2
7	-comm	19	-SON
8	-town	20	-ADDR
9	-dname	21	-son_id
10	-budget	22	-sname
11	-size	23	-ad1
12	-city		

กราฟในรูป 5.2 สามารถถ่ายทอดความสัมพันธ์เป็นเมตริกซ์ได้ดังตารางที่ 5.7

ตารางที่ 5.7 เมตริกซ์แสดงความสัมพันธ์ของระบบฐานข้อมูล

โหนด	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1	1	1	1	1	1	1	1	1					1	1		1	1	1	1		1	1	
2		1	1	1	1	1	1	1						1			1						
3			1	1	1	1	1	1	1	1	1	1	1	1	1	1	1						
4				1	1	1	1	1						1			1						
5					1	1	1	1						1			1						
6						1	1	1						1			1						
7							1	1						1			1						
8								1						1			1						
9									1	1	1	1			1		1						
10										1	1	1			1		1						
11											1	1			1		1						
12												1			1		1						
13													1			1	1						
14														1			1						
15															1		1						
16																1	1						
17																	1						
18																		1	1	1	1	1	1
19																			1		1	1	
20																				1	1		1
21																					1	1	1
22																						1	
23																							1

การนำข้อมูลที่ได้จากตารางความสัมพันธ์องค์ประกอบของระบบฐานข้อมูลไปใส่ลงในแฟ้มข้อมูล *N.ton* ในรูปแบบที่กำหนดของแฟ้มข้อมูลเริ่มด้วยขนาดของเมตริกซ์ทั้งหมดในที่นี้คือ 23 แล้วเว้นช่องว่างจึงเริ่มส่วนที่เป็นข้อมูลจากตาราง ใส่ค่าของบิตที่เป็น 0 หรือ 1 โดยค่าที่ว่างในตารางให้แทนด้วย 0 ในลักษณะเรียงบิตไปตามแถวจากแถวที่ 1 สดมภ์ที่ 2 ไปจนถึง แถวที่ 1 สดมภ์ที่ N แล้วต่อด้วย แถวที่ 2 สดมภ์ที่ 3 จนใส่ข้อมูลลงในแฟ้มข้อมูลครบสมบูรณ์ ดังจะได้ตามตารางที่ 5.8

ตารางที่ 5.8 ข้อมูลที่ถูกเก็บในแฟ้มข้อมูล *N.ton*

23 11111110000110111101101111111100...

5.3 การสร้างพจนานุกรมฐานความรู้

การสร้างพจนานุกรมฐานความรู้จะนำเอาข้อมูลความสัมพันธ์ที่ได้จากโมเดลข้อมูล และตำแหน่งอ้างอิงที่ได้กำหนดในตารางความสัมพันธ์ขององค์ประกอบระบบฐานข้อมูลลงในพจนานุกรมฐานความรู้ การนำข้อมูลเพิ่มลงในพจนานุกรมฐานความรู้นั้นจะมีข้อมูลสำคัญอยู่ 4 ส่วนคือ

1) ชื่อของวัตถุในระบบ (ชื่อฐานข้อมูล ชื่อตาราง ชื่อเขตข้อมูล ความสัมพันธ์ ค่าของเขตข้อมูล และคำที่อยู่ในระบบ)

2) ชนิดไวยากรณ์ของคำข้อมูล

3) คำข้อมูลที่อ้างอิงจะต้องเป็นคำที่มีการกำหนดไว้ก่อนที่จะอ้างถึงคำนั้น เช่น *Pongpanya* เป็นค่าของเขตข้อมูล *ename* ซึ่งเป็นคำข้อมูลที่ *Pongpanya* อ้างถึงนั้นจะต้องถูกกำหนดมาก่อนมิฉะนั้นจะไม่สามารถอ้างถึงคำข้อมูลนั้นได้

4) ตำแหน่งของวัตถุนั้นในตารางความสัมพันธ์

การแบ่งแต่ละส่วนจะใช้ช่องว่างเป็นตัวแบ่ง ตัวอย่างข้างล่างจะเป็นบางส่วนของแฟ้มคำข้อมูลที่สร้างขึ้นในที่นี้ตั้งชื่อแฟ้มข้อมูล *dictdb.ton* ในที่นี้จะแสดงเป็น 3 สดมภ์เพื่อความสะดวกในการมองภาพรวมของแฟ้มข้อมูล

ตารางที่ 5.9 ข้อมูลของพจนานุกรมฐานความรู้

TEST 17 # 17	when 1024 # 3	C001 4120 dep_id 0
EMP 18 TEST 14	what 1024 # 4	P002 4120 emp_id 0
DEP 18 TEST 15	which 1024 # 5	…
ANC 18 TEST 16	whose 1024 # 6	whose,SON 8194 EMP 0
emp_id 24 EMP 1	that 1024 # 1	whose,EMP 8194 SON 0
ename 20 EMP 2	how 1024 # 7	…
dep_id 24 DEP 3	work 50 DEP 0	who,has,SON 8194 EMP 0
…	in 128 # 0	…
emp_id.EMP 24 emp_id 1	department 18 DEP 0	which,EMP 8194 EMP 0
ename.EMP 20 ename 2	locate 32 # 0	who,work,DEP 8194 EMP 0
…	own 96 # 0	where,stay,SON 8196 ad1 0
I 512 # 0	commission 20 comm 0	…
want 32 # 0	salary 20 sal 0	
who 1024 # 1	Pongpanya 4116 ename 0	
where 1024 # 2	David 4116 ename 0	

บันทึกข้อมูลดังกล่าวลงในแฟ้มข้อมูลกำหนดชื่อตามเห็นควรแล้วใช้โปรแกรม *dic* ช่วยสร้างพจนานุกรมฐานความรู้ ในส่วนของโปรแกรม *dic* นั้นมีฟังก์ชันหลักอยู่ 6 ฟังก์ชันคือ

1) ค้นคำข้อมูล

2) เพิ่มคำข้อมูล

3) บันทึกข้อมูลที่สร้างพจนานุกรมฐานความรู้ลงในแฟ้มข้อมูลชื่อ *RESULT.TON*

4) บรรจุข้อมูลจากแฟ้มข้อมูลที่บันทึกไว้ลงในพจนานุกรมฐานความรู้

5) เพิ่มแฟ้มคำข้อมูล ฟังก์ชันนี้จะทำการเพิ่มคำข้อมูลเป็นแฟ้มข้อมูล โดยจะรอรับชื่อแฟ้มคำข้อมูลที่เราได้สร้างไว้

6) ตรวจสอบพจนานุกรมฐานความรู้ ฟังก์ชันนี้จะรอรับชื่อแฟ้มคำข้อมูลที่เราจะตรวจสอบความสมบูรณ์ของการเพิ่มคำข้อมูลดังกล่าวในพจนานุกรมฐานความรู้

ดังตัวอย่างคำข้อมูลจากแฟ้มข้อมูล *dictdb.ton* จะได้เป็นพจนานุกรมฐานความรู้บางส่วนที่แสดงในตารางที่ 5.10

ตารางที่ 5.10 โครงสร้างแถวลำดับพจนานุกรมฐานความรู้ที่สร้างจาก *dictdb.ton*

LNK	WORD	GRAM_CATE	ROOT	POSITION
12	TEST	17	0	17
5	EMP	18	1	14
7	DEP	18	1	15
13	emp_id	24	2	1
14	Ename	20	2	2
27	Budget	20	3	10
...				
...				
169	Like	32	0	0

5.4 ตัวอย่างขั้นตอนการแปลคำร้องขอ

ตัวอย่างคำร้องขอของผู้ใช้ระบบ ขั้นตอนและผลลัพธ์ภาษาสอบถามเชิงโครงสร้างของคำร้องขอนั้น ในการทำงานของโปรแกรมที่พัฒนาขึ้นระบบจะข้ามคำข้อมูลอื่นที่ไม่สำคัญ หรือมีผลต่อการแปลความหมายไปจนถึงคำข้อมูลที่ต้องการจากนั้นจะเริ่มแปลคำร้องขอตามรูปแบบไวยากรณ์ที่กำหนดไว้ ในที่นี้จะละไว้ในฐานที่เข้าใจแต่จะมุ่งประเด็นไปที่การสรุปความหมายของข้อมูลนั้น

- find all department in Bangkok

1) ระบบจะข้ามคำข้อมูล find, all เนื่องจาก find ไม่ได้เป็นคำข้อมูลชนิดกริยาที่ไม่แสดงความสัมพันธ์กับระบบฐานข้อมูลจึงสามารถข้ามไปได้ ส่วน all จากข้อจำกัดของภาษาสอบถามเชิงโครงสร้างmSQL และการวิเคราะห์กลุ่มตัวอย่างจะได้ว่าคำที่มีความหมายตรงข้ามกับ all คือ some ไม่สามารถใช้ในระบบเพราะการเลือกค้นคืนข้อมูลบางส่วนจากตารางใดๆ ต้องระบุสิ่งที่ต้องการ ดังนั้น ทำให้ all จึงเป็นสถานะเริ่มต้นที่ใช้ในการแปลความหมายถ้าไม่มีการระบุถึงสิ่งที่ต้องการระบบจึงสามารถข้ามคำข้อมูลนี้ได้จนถึง department

2) department มีชนิดไวยากรณ์เป็นคำนามอีกทั้งยังมีความสำคัญกับระบบโดยการอ้างถึง DEP ที่เป็นชื่อตารางในระบบ จะทำการใส่ข้อมูลลงในภาษาสอบถามเชิงโครงสร้างแบบตรรกะ

3) Bangkok มีชนิดไวยากรณ์เป็นคำนาม และค่าของเขตข้อมูล โดยอ้างอิงไปยังคำข้อมูล address ซึ่งเป็นคำนามทั่วไปประเภทหนึ่ง เมื่อรวมกับข้อมูลตารางที่ได้ก่อนหน้านี้จะได้ address.DEP ทำให้สรุปได้จากพจนานุกรมฐานความรู้ว่าเป็น เขตข้อมูล city ในตาราง DEP ระบบจะทำการตรวจสอบรูปแบบคำข้อมูลที่ถูกต้องกับพจนานุกรมฐานความรู้ซึ่ง Bangkok ที่ผู้ใช้ระบุมาในระบบฐานข้อมูลอาจเป็น bangkok ฉะนั้นระบบจะทำการเปลี่ยนให้ตรงกับคำข้อมูลที่อยู่ในระบบฐานข้อมูล อีกทั้ง bangkok อาจเป็นบางส่วนของค่าในเขตข้อมูลที่เป็นกลุ่ม จึงทำการเพิ่มเติมเพื่อป้องกันกรณีดังกล่าวจะได้ว่าจาก Bangkok เป็น '%Bangkok%'

4) เมื่อผู้ใช้ไม่ได้ระบุเขตข้อมูลที่ต้องการระบบจะเลือกแสดงทุกเขตข้อมูล และเนื่องจากส่วนใหญ่ผู้ใช้ระบบมักต้องการผลข้อมูลที่ไม่มีความซ้ำซ้อน ระบบจึงทำการเพิ่มเติม distinct เข้าไปในภาษาสอบถามเชิงโครงสร้างเพื่อวัตถุประสงค์ดังกล่าวด้วย

5) ระบบจะตรวจสอบความสมบูรณ์ของคำร้องขอที่ได้รับการแปลงเป็นภาษาสอบถามเชิงโครงสร้างแล้วว่าข้อมูลทุกส่วนในภาษามีความสัมพันธ์กัน ถ้ามีบางส่วนไม่สมบูรณ์ระบบจะทำการค้นหาและเชื่อมโยงชุดเงื่อนไขโดยใช้ตารางความสัมพันธ์เป็นเครื่องสำหรับการกระทำดังกล่าว ซึ่งจะทำให้ได้ใกล้เคียงคำตอบที่ผู้ใช้ต้องการมากที่สุด

6) ผลลัพธ์ที่ได้

```
TEST#select distinct * from DEP where DEP.city like '%Bangkok%'
```

- Find the number and name of the employee who work in a department located in london

1) number และ name เป็นคำนามทั่วไปไม่สามารถระบุได้

2) employee คือตารางEMP ทำให้เราสามารถระบุได้ว่า number.EMP -> emp_id และ name.EMP -> ename

3) department -> DEP เมื่อเราเพิ่มลงในระบบจะได้ว่าตาราง DEP กับ ตารางEMP มีความสัมพันธ์ระหว่างกันซึ่งถูกเชื่อมด้วยเขตข้อมูล dep_id จากการใช้ตารางความสัมพันธ์ขององค์ประกอบระบบฐานข้อมูล dep_id $\in$ (D(DEP) $\cap$ D(EMP))

4) london -> address -> address.DEP -> city

5) EMP.emp_id, EMP.ename $\in$ D(CONDITION)

6) ผลลัพธ์ที่ได้

TEST#select distinct EMP.emp_id, EMP.ename from EMP, DEP

where EMP.dep_id = DEP.dep_id and DEP.city like '%london%'

- Gets information of pongpanya such as age , salary and job

1) pongpanya -> ename -> ename.EMP

2) age -> age.EMP

3) salary -> sal -> sal.EMP

4) job -> job.EMP

5) EMP.age, EMP.sal, EMP.job $\in$ D(CONDITION)

6) ผลลัพธ์ที่ได้

TEST#select distinct EMP.age, EMP.sal, EMP.job from EMP

- Whose son stays at silom street

1) son -> SON

2) silom -> Silom -> address -> address.SON -> ad1 -> ad1.ADDR

3) D(SON) $\cap$ D(ADDR) -> son_id

4) street -> address -> address.SON -> ad1 ->ad1.ADDR

5) whose,son -> EMP

6) D(SON) $\cap$ D(EMP) -> emp_id และ D(ADDR) $\cap$ D(EMP) -> $\varnothing$

7) มีความสัมพันธ์ข้ามฐานข้อมูลระหว่างฐานข้อมูล TEST และ ฐานข้อมูล DB2 ระบบจะทำการแยกข้อมูลที่ได้จากคำร้องขอตามฐานข้อมูล โดยจะเก็บชุดเงื่อนไขตัวเชื่อมความสัมพันธ์ระหว่างฐานข้อมูลทั้งสองไว้ และการเรียงลำดับผลลัพธ์โดยใช้เขตข้อมูลที่อยู่ในชุดเงื่อนไขตัวเชื่อมความสัมพันธ์

8) * $\in$ D(COND1) และ SON.emp_id $\in$ D(COND2)

9) ผลลัพธ์ที่ได้

TEST:DB2#EMP.emp_id = SON.emp_id

TEST#select distinct * from EMP order by EMP.emp_id

DB2#select distinct SON.emp_id from SON, ADDR

where SON.son_id = ADDR.son_id and ADDR.ad1 like '%Silom%'

and ADDR.ad1 like '%street%' order by SON.emp_id

- where is david's son's address#

1) david -> name
2) son -> SON
3) david's son -> SON.name -> name_SON -> name.EMP -> ename -> ename.EMP
4) D(SON) $\cap$ D(EMP) -> emp_id
5) address -> address.SON -> ad1 -> ad1.ADDR
6) cross database
7) ADDR.ad1, SON.emp_id $\in$ D(COND1) และ EMP.emp_id $\in$ D(COND2)
8) ผลลัพธ์ที่ได้

DB2:TEST#SON.emp_id = EMP.emp_id

DB2#select distinct ADDR.ad1, SON.emp_id from SON, ADDR

where SON.son_id = ADDR.son_id order by SON.emp_id

TEST#select distinct EMP.emp_id from EMP

where EMP.ename like '%David%' order by EMP.emp_id

- list name of department of ton's father

1) name
2) department -> DEP
3) name of department -> dname -> dname.DEP
4) ton -> Ton -> name
5) father -> EMP
6) D(DEP) $\cap$ D(EMP) -> dep_id
7) ton -> name_EMP -> name.SON -> sname -> sname.SON
8) D(DEP) $\cap$ D(SON) -> $\varnothing$ และ D(EMP) $\cap$ D(SON) -> emp_id
9) cross database

10) DEP.dname, EMP.emp_id ∈ D(COND1) และ SON.emp_id ∈ D(COND2)

11) ผลลัพธ์ที่ได้

TEST:DB2#EMP.emp_id = SON.emp_id

TEST#select DISTINCT DEP.dname ,EMP.emp_id from DEP ,EMP

where DEP.dep_id = EMP.dep_id order by EMP.emp_id

DB2#select DISTINCT SON.emp_id from SON

where SON.sname like '%Ton%' order by SON.emp_id

- Whose father works in x department and he have the same name as his father

1) father -> EMP
2) whose,EMP -> SON
3) D(EMP) ∩ D(SON) -> emp_id
4) x ไม่มีอยู่ในพจนานุกรมฐานความรู้ จะถูกกำหนดเป็น name
5) department -> DEP
6) D(EMP) ∩ D(DEP) -> dep_id และ D(SON) ∩ D(DEP) -> ∅
7) x -> name -> name.DEP -> dname -> dname.DEP
8) he -> SON
9) same name as his father -> name.SON = name.EMP
10) cross database
11) * ∈ D(COND1) และ EMP.emp_id, EMP.ename ∈ D(COND2)
12) ผลลัพธ์ที่ได้

TEST:DB2#EMP.emp_id = SON.emp_id

DB2:TEST#SON.sname = EMP.ename

DB2#select DISTINCT * from SON order by SON.emp_id , SON.sname

TEST#select DISTINCT EMP.emp_id ,EMP.ename from EMP ,DEP

where EMP.dep_id = DEP.dep_id and DEP.dname like '%x%'

order by EMP.emp_id , EMP.ename

- find employee who has son 's name as same as ton #

1) employee -> EMP
2) son -> SON

3) D(EMP) ∩ D(SON) -> emp_id

4) name -> name.SON -> sname -> sname.SON

5) ton -> Ton

6) cross database

7) * ∈ D(COND1) และ SON.emp_id ∈ D(COND2)

8) ผลลัพธ์ที่ได้

TEST:DB2#EMP.emp_id = SON.emp_id

TEST#select DISTINCT * from EMP order by EMP.emp_id

DB2#select DISTINCT SON.emp_id from SON

where SON.sname like '%Ton%' order by SON.emp_id

- find employee who work at the same area as his son's address #

1) employee -> EMP

2) area -> address

3) EMP,work -> DEP

4) area -> address -> address.DEP -> city -> city.DEP

5) D(EMP) ∩ D(DEP) -> dep_id

6) son -> SON

7) D(EMP) ∩ D(SON) -> emp_id และ D(DEP) ∩ D(SON) -> ∅

8) address -> address.SON -> ad1 -> ad1.ADDR

9) D(EMP) ∩ D(ADDR) -> ∅ และ D(SON) ∩ D(ADDR) -> son_id และ D(DEP) ∩ D(ADDR) -> ∅

10) cross database

11) * ∈ D(COND1) และ SON.emp_id, ADDR.ad1 ∈ D(COND2)

12) ผลลัพธ์ที่ได้

TEST:DB2#EMP.emp_id = SON.emp_id

TEST:DB2#DEP.city like ADDR.ad1

TEST#select DISTINCT * from EMP ,DEP where EMP.dep_id = DEP.dep_id

order by EMP.emp_id , DEP.city

DB2#select DISTINCT SON.emp_id ,ADDR.ad1 from SON ,ADDR

where SON.son_id = ADDR.son_id order by SON.emp_id , ADDR.ad1

5.5 ตัวอย่างประโยคและผลของการประมวลผล

คำร้องขอ

```
select emp_id,ename,emp_id from EMP
where dep_id in ( select dep_id from DEP where city = EMP.town)
```

ผลลัพธ์

```
TEST#select EMP.emp_id ,EMP.ename  from EMP ,DEP
where EMP.dep_id = DEP.dep_id and DEP.city = EMP.town
```

คำร้องขอ

```
select EMP.emp_id,EMP.ename from EMP, DEP
where DEP.city = EMP.town and DEP.dep_id = EMP.dep_id and EMP.comm > 50000
```

ผลลัพธ์

```
TEST#select EMP.emp_id ,DEP.emp_id ,EMP.ename  from EMP ,DEP
where DEP.city = EMP.town and DEP.dep_id = EMP.dep_id and EMP.comm > 50000
```

คำร้องขอ

```
select emp_id,ename from EMP
where exists( select * from DEP where dep_id = EMP.dep_id and
city = EMP.town or city = 'london')
```

ผลลัพธ์

```
TEST#select EMP.emp_id ,EMP.ename  from EMP ,DEP
where DEP.dep_id = EMP.dep_id and DEP.city = EMP.town or DEP.city like 'london'
```

คำร้องขอ

```
select emp_id,ename from EMP
where dep_id = any( select dep_id from DEP where city = EMP.town)
```

ผลลัพธ์

```
TEST#select EMP.emp_id ,EMP.ename  from EMP ,DEP
where EMP.dep_id = DEP.dep_id and DEP.city = EMP.town
```

คำร้องขอ

```
select EMP.ename from EMP where EMP.emp_id = 'P0002'
```

ผลลัพธ์

TEST#select EMP.ename from EMP where EMP.emp_id like 'P0002'

คำร้องขอ

select * from EMP, DEP, ANC

ผลลัพธ์

TEST#select * from EMP ,DEP ,ANC

where DEP.dep_id = EMP.dep_id and ANC.emp_id = EMP.emp_id and ANC.dep_id =

EMP.dep_id and ANC.dep_id = DEP.dep_id

คำร้องขอ

select emp_id,ename from EMP

where dep_id in (select dep_id from DEP where city = EMP.town and

city in (select city from DEP where city like 'Bangkok' and budget > 100000)) and

emp_id = any(select emp_id from SON where sname like 'ton') and COMM > 1000

ผลลัพธ์

TEST:DB2#EMP.emp_id = SON.emp_id

TEST#select EMP.emp_id ,EMP.ename from EMP ,DEP where EMP.dep_id = DEP.dep_id and

DEP.city = EMP.town and and DEP.city like 'Bangkok' and DEP.budget > 100000 and

EMP.comm > 1000 order by EMP.emp_id

DB2#select SON.emp_id from SON where SON.sname like 'ton' order by SON.emp_id

คำร้องขอ

find name of employee who have number of employee is P001

ผลลัพธ์

TEST#select DISTINCT DEP.dname from DEP ,EMP

where DEP.dep_id = EMP.dep_id and EMP.emp_id like '%P001%'

คำร้องขอ

find name of P001

ผลลัพธ์

TEST#select DISTINCT EMP.ename from EMP where EMP.emp_id like '%P001%'

คำร้องขอ

find name of employee who have name of son is “TON”

ผลลัพธ์

TEST:DB2#EMP.emp_id = SON.emp_id

TEST#select DISTINCT EMP.ename ,EMP.emp_id from EMP order by EMP.emp_id

DB2#select DISTINCT SON.emp_id from SON

where SON.sname like 'TON' order by SON.emp_id

คำร้องขอ

which employee gets salary more than 50000 Baht and his son stays in Bangkok

ผลลัพธ์

TEST:DB2#EMP.emp_id = SON.emp_id

DB2#select DISTINCT SON.emp_id from ADDR ,SON

where SON.son_id = ADDR.son_id and ADDR.ad1 like '%Bangkok%' order by SON.emp_id

TEST#select DISTINCT EMP.emp_id from EMP

where EMP.sal > 50000 order by EMP.emp_id

คำร้องขอ

which employee gets salary more than 50000 Baht and his son stays in Bangkok

ผลลัพธ์

TEST:DB2#EMP.emp_id = SON.emp_id

DB2#select DISTINCT SON.emp_id from ADDR ,SON

where SON.son_id = ADDR.son_id and ADDR.ad1 like '%Bangkok%' order by SON.emp_id

TEST#select DISTINCT EMP.emp_id from EMP where EMP.sal > 50000

order by EMP.emp_id

คำร้องขอ

whose father does his work in the year 1998 and gets salary not less than 50000 Baht

ผลลัพธ์

TEST:DB2#EMP.emp_id = SON.emp_id

DB2:TEST#SON.emp_id = ANC.emp_id

DB2#select DISTINCT * from SON order by SON.emp_id

TEST#select DISTINCT EMP.emp_id ,ANC.emp_id from EMP ,DEP ,ANC

where EMP.dep_id = DEP.dep_id and EMP.emp_id = ANC.emp_id and

EMP.dep_id = ANC.dep_id and DEP.dep_id = ANC.dep_id and ANC.year = 1998 and

EMP.sal >= 50000 order by EMP.emp_id , ANC.emp_id

คำร้องขอ

where is ton 's son

ผลลัพธ์

DB2:TEST#SON.emp_id = EMP.emp_id

DB2#select DISTINCT ADDR.ad1 ,SON.emp_id from ADDR ,SON

where SON.son_id = ADDR.son_id order by SON.emp_id

TEST#select DISTINCT EMP.emp_id from EMP where EMP.ename like '%ton%'

order by EMP.emp_id

คำร้องขอ

where does ton 's father work

ผลลัพธ์

TEST:DB2#EMP.emp_id = SON.emp_id

TEST#select DISTINCT DEP.dname ,EMP.emp_id from DEP ,EMP

where EMP.dep_id = DEP.dep_id order by EMP.emp_id

DB2#select DISTINCT SON.emp_id from SON where SON.sname like '%ton%'

order by SON.emp_id

คำร้องขอ

where does father of ton work

ผลลัพธ์

TEST:DB2#EMP.emp_id = SON.emp_id

TEST#select DISTINCT DEP.dname ,EMP.emp_id from DEP ,EMP

where EMP.dep_id = DEP.dep_id order by EMP.emp_id

DB2#select DISTINCT SON.emp_id from SON where SON.sname like '%ton%'

order by SON.emp_id

คำร้องขอ

who work in accounting in the year 1998

ผลลัพธ์

TEST#select DISTINCT * from EMP ,DEP ,ANC

where EMP.dep_id = DEP.dep_id and DEP.dname like '%Accounting%' and

EMP.emp_id = ANC.emp_id and EMP.dep_id = ANC.dep_id and DEP.dep_id = ANC.dep_id

and ANC.year = 1998

คำร้องขอ

find employee who work in accounting in the year 1998

ผลลัพธ์

TEST#select DISTINCT * from EMP ,DEP ,ANC

where EMP.dep_id = DEP.dep_id and DEP.dname like '%Accounting%' and

EMP.emp_id = ANC.emp_id and EMP.dep_id = ANC.dep_id and DEP.dep_id = ANC.dep_id

and ANC.year = 1998

คำร้องขอ

find salary in department that has budget more than 50000

ผลลัพธ์

TEST#select DISTINCT EMP.sal from EMP ,DEP

where EMP.dep_id = DEP.dep_id and DEP.budget > 50000

คำร้องขอ

find person who work as a manager in the accounting department

ผลลัพธ์

TEST#select DISTINCT * from EMP ,DEP where EMP.job like '%manager%' and

EMP.dep_id = DEP.dep_id and DEP.dname like '%Accounting%'

คำร้องขอ

list the department 's name that name as "ac*"

ผลลัพธ์

TEST#select DISTINCT DEP.dname from DEP where DEP.dname like 'ac%'

คำร้องขอ

find employee who work in the same city as his son

ผลลัพธ์

TEST:DB2#EMP.emp_id = SON.emp_id

TEST:DB2#DEP.city like ADDR.ad1

TEST#select DISTINCT * from EMP ,DEP where EMP.dep_id = DEP.dep_id

order by EMP.emp_id , DEP.city

DB2#select DISTINCT SON.emp_id ,ADDR.ad1 from SON ,ADDR

where SON.son_id = ADDR.son_id order by SON.emp_id , ADDR.ad1

คำร้องขอ

select Ename from EMP,DEP where dname in ('MIS, 'accounting' , 'marketing');

ผลลัพธ์

TEST#select EMP.ename from EMP ,DEP where DEP.dname like 'MIS' or

DEP.dname like 'accounting' or DEP.dname like 'marketing' and DEP.dep_id = EMP.dep_id

คำร้องขอ

what is number of Pongpanya#

ผลลัพธ์

TEST#select DISTINCT EMP.emp_id from EMP where EMP.ename like '%Pongpanya%'

คำร้องขอ

what is the name of number P003#

ผลลัพธ์

TEST#select DISTINCT EMP.ename ,EMP.emp_id from EMP

where EMP.emp_id like '%P003%'

คำร้องขอ

what is department of David#

ผลลัพธ์

TEST#select DISTINCT * from DEP ,EMP

where DEP.dep_id = EMP.dep_id and EMP.ename like '%David%'

คำร้องขอ

which is employee in MIS or Accounting or Marketing department#

ผลลัพธ์

TEST#select DISTINCT * from EMP ,DEP where EMP.dep_id = DEP.dep_id and
DEP.dname like '%MIS%' or DEP.dname like '%Accounting%' or DEP.dname like '%Marketing%'

คำร้องขอ

get the all detail of employee who work in bangkok#

ผลลัพธ์

TEST#select DISTINCT * from EMP ,DEP
where EMP.dep_id = DEP.dep_id and DEP.city like '%Bangkok%'

คำร้องขอ

list all of david's son#

ผลลัพธ์

TEST:DB2#EMP.emp_id = SON.emp_id
TEST#select DISTINCT EMP.emp_id from EMP where EMP.ename like '%David%'
order by EMP.emp_id
DB2#select DISTINCT SON.emp_id from SON order by SON.emp_id

คำร้องขอ

how many employee that work in the accounting department from the year 1992 till 1999#

ผลลัพธ์

TEST#select DISTINCT * from EMP ,DEP ,ANC
where EMP.dep_id = DEP.dep_id and DEP.dname like '%Accounting%' and
EMP.emp_id = ANC.emp_id and EMP.dep_id = ANC.dep_id and DEP.dep_id = ANC.dep_id
and ANC.year >= 1992 and ANC.year <= 1999

คำร้องขอ

where is employee who name 'mary'#

ผลลัพธ์

TEST#select DISTINCT * from EMP where EMP.ename like 'mary'

บทที่ 6

สรุปผลการวิจัย

6.1 สรุปผลการวิจัย

การวิจัยระบบค้นคืนข้อมูลสารสนเทศน์โดยใช้ภาษาธรรมชาติที่เป็นภาษาอังกฤษ สามารถค้นคืนสารสนเทศน์ที่มีเงื่อนไขเกี่ยวพันมากกว่า 1 ฐานข้อมูลนั้น ผู้วิจัยพัฒนาต้นแบบโดยใช้แนวความคิดจากการแปลความหมายด้วยวิธี nondeterministic topdown ซึ่งมีหลักไวยากรณ์ของภาษาเป็นตัวกำหนดทางเลือก ร่วมกับหลักความสัมพันธ์ที่ได้จากโมเดลข้อมูลกับข้อมูลที่วิเคราะห์จากคำร้องขอของผู้ใช้ที่อยู่ในพจนานุกรมฐานความรู้ และสร้างตารางความสัมพันธ์องค์ประกอบของระบบฐานข้อมูลเพื่อใช้ในการตรวจสอบข้อผิดพลาดของคำร้องขอ พร้อมทั้งปรับปรุงให้มีความสมบูรณ์ เพื่อให้ได้คำตอบตรงกับความต้องการของผู้ใช้ระบบมากที่สุด

ในส่วนของการพัฒนาวิธีแก้ปัญหาที่ปรากฏในการวิเคราะห์คำร้องขอ คือ กรณีคำข้อมูลที่ไม่มีในระบบ กรณีค่าของเขตข้อมูลที่เป็นกลุ่มคำ กรณีค่าของเขตข้อมูลที่ไม่ระบุเฉพาะเจาะจงไปยังเขตข้อมูลหนึ่งๆ และกรณีของคำข้อมูลที่มีชนิดของไวยากรณ์มากกว่าหนึ่งชนิด ซึ่งวิธีแก้ปัญหาที่ได้พัฒนาขึ้นทำให้สามารถขยายขอบเขตรูปแบบที่รองรับคำร้องขอของผู้ใช้บริการได้มากยิ่งขึ้น

นอกจากนี้ระบบรองรับการสอบถามโดยใช้ภาษาสอบถามเชิงโครงสร้าง mSQL ที่ได้รับการเพิ่มความสามารถแบบข่าย ทำให้ผู้ใช้บริการที่มีความถนัดในการใช้ภาษาสอบถามเชิงโครงสร้างมีความอิสระในการใช้งานกว่าเดิม

6.2 ข้อจำกัดของระบบ

ระบบการค้นคืนที่ทำการพัฒนานี้ยังไม่สามารถตอบสนองรูปแบบคำถามที่มีการเปรียบเทียบกับเขตข้อมูลของตนเองอันจะนำไปสู่รูปแบบภาษาสอบถามเชิงโครงสร้างที่มีการใช้คำเสมือน (alias word) ได้อย่างถูกต้อง นอกจากนี้แล้วข้อจำกัดที่เกิดจากภาษา mSQL เป็นส่วนหนึ่งที่ทำให้ความสามารถของระบบค้นคืนไม่สามารถตอบสนองความต้องการบางประเภทที่สามารถกระทำได้ด้วยระบบฐานข้อมูลอื่น เช่น ความสามารถในด้านของการจัดกลุ่มข้อมูล เป็นต้น

6.3 ข้อเสนอแนะ

1) จากข้อจำกัดของระบบค้นคืนที่มีข้อจำกัดในรูปแบบคำถามที่ผู้ใช้ไม่สามารถสอบถามโดยมีการเปรียบเทียบกับเขตข้อมูลของตนเองได้ ดังนั้นการพัฒนาเพื่อให้ระบบค้นคืนมีประสิทธิภาพเทียบเท่ากับระบบภาษาธรรมชาติได้ จะต้องทำการศึกษากำหนดรูปแบบเฉพาะของการค้นคืนที่สามารถตอบสนองความต้องการของผู้ใช้ในหลากหลายรูปแบบตามความสามารถของภาษาธรรมชาติที่ใช้อยู่ในปัจจุบัน

2) ระบบค้นคืนที่ได้พัฒนานี้สามารถทำการค้นคืนข้อมูลจากหลายฐานข้อมูลที่ปรากฏในระบบเดียวกัน หากแต่ยังไม่สามารถทำการค้นคืนระบบฐานข้อมูลที่อยู่ต่างเครื่องให้บริการได้ ดังนั้นเพื่อเป็นการขยายขอบเขตของการพัฒนาระบบค้นคืนให้กว้างขึ้น การกำหนดรูปแบบของการค้นคืนบนระบบฐานข้อมูลแบบกระจายเต็มรูปแบบ (หลายฐานข้อมูลบนหลายระบบการทำงานที่แตกต่างกัน) จึงเป็นส่วนที่ควรกระทำ

รายการอ้างอิง

[1] วิรัช ศรเลิศล้ำวาณิช, อภิชิต พิทยรัตน์โสภณ และเกรียงชัย จันทร์แสนวิไล. "การจัดการฐานข้อมูลพจนานุกรมไทยด้วยทรัยแถวคู่หลายดรรชนี". การประชุมวิชาการ ครั้งที่ 5, ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ. หน้า 197-206.

[2] Mini SQL 2.0 User Guide, (n.p.):Hughes Technologies Pty, 1997.

[3] Jun-ichi Aoe. "An Efficient Digital Search Algorithm by Using a Double-Array Structure". IEEE Transactions on Knowledge and Data Engineering 15 (1989) : 1066-1077.

[4] Jun-ichi Aoe, Katsushi Morimoto, Masami Shishibori, and Ki-Hong Park. "A Trie Compaction Algorithm for a Large Set of Keys". IEEE Transactions on Knowledge and Data Engineering 8 (June 1996) : 476-491.

[5] Thomas A. Standish. Data Structures, Algorithms & Software Principles In C . (n.p.):Addison Wesley 1994.

[6] Mysore Ramaswamy, Sumit Sarkar, Member, IEEE Computer society, and Ye-Sho Chen. "Using Directed Hypergraphs to Verify Rule-Based Expert Systems". IEEE Transaction Knowledge and Data Engineering 9.2 (March-April 1997) :221-237.

[7] John K. Ousterhout, Tcl and the Tk Toolkit, (n.p.):Addison-Wesley 1994.

[8] Patrick Bosc, and Olivier Pivert. "SQLf: A Relational Database Language for Fuzzy Querying. IEEE Transactions on Fuzzy Systems 3.1 (February 1995) : 1-17.

[9] Nabil R. Adam, and Aryya Gangopadhyay. "A Form-Based Natural Language Front-End to a CIM Database". IEEE Transaction Knowledge and Data Engineering 9.2 (March-April 1997) : 287-300.

ภาคผนวก

ภาคผนวก ก

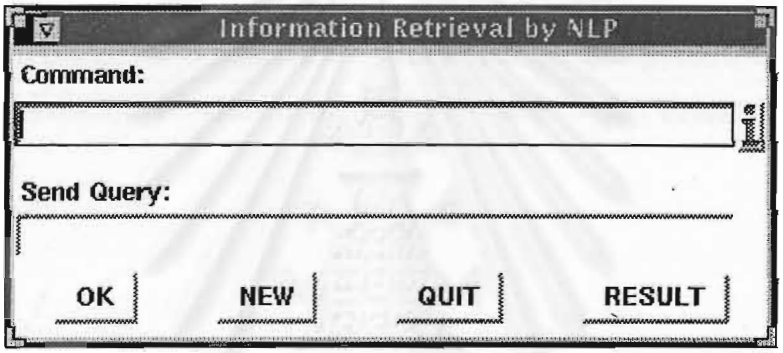
คู่มือการใช้งาน

การเข้าสู่ระบบเพื่อเริ่มการใช้งานกระทำได้โดยพิมพ์คำสั่ง

test

ซึ่งจะปรากฏจอภาพดังต่อไปนี้

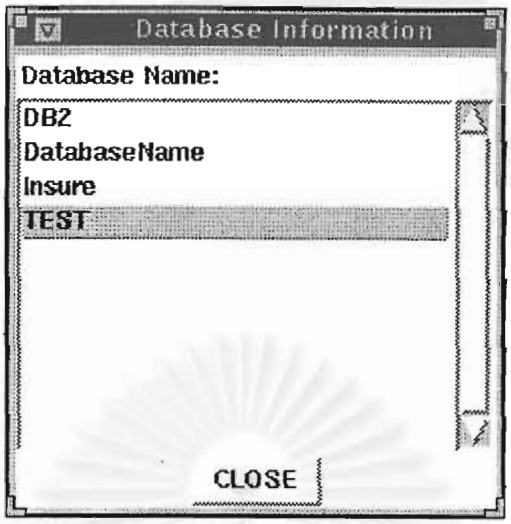
รูปที่ ก.1



การทำงาน:

1. แสดงรูปที่ ก.1 ถ้าผู้ใช้ต้องการเรียกดูโครงสร้างข้อมูลของระบบฐานข้อมูล ไปกระทำยังข้อ 2 ถ้าต้องการค้นคืนข้อมูลไปกระทำตามข้อ 6 เมื่อกดปุ่ม **QUIT** จะออกจากโปรแกรม
2. กดปุ่ม **i** จะปรากฏชื่อฐานข้อมูลทั้งหมดที่มีอยู่ในระบบฐานข้อมูลดังรูปที่ ก.2 และไปยังข้อ 3
3. เมื่อต้องการเรียกดูข้อมูลของโครงสร้างของฐานข้อมูลใด กดสองครั้งติดกันที่ชื่อฐานข้อมูลนั้น จะไปยังข้อ 4 แต่ถ้ากดปุ่ม **CLOSE** จะย้อนกลับไปยังข้อ 1

รูปที่ ก.2

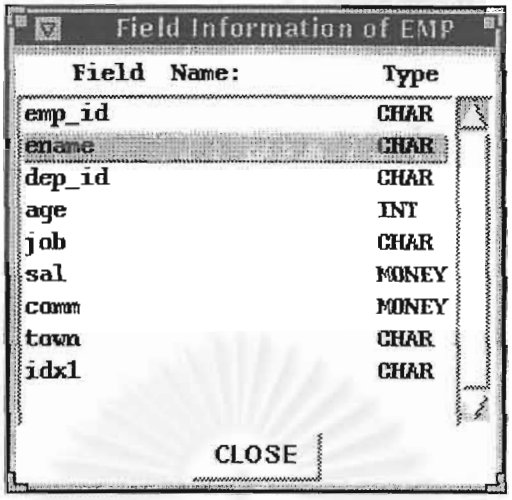


4. จอภาพแสดงชื่อตารางจะปรากฏดังรูปที่ ก.3 เมื่อต้องการเรียกดูข้อมูลของโครงสร้างของตารางใด กดสองครั้งติดกันที่ชื่อตารางนั้น จะไปยังข้อ 5 แต่ถ้ากดปุ่ม **CLOSE** จะย้อนกลับไปยังข้อ 3

รูปที่ ก.3

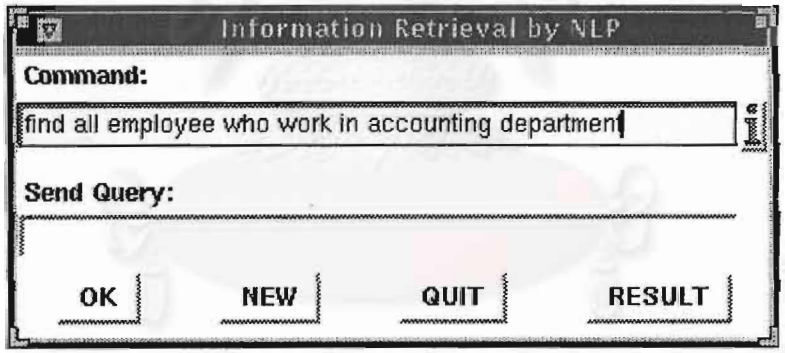
5. จอภาพแสดงชื่อเขตข้อมูลจะปรากฏดังรูปที่ ก.4 แล้วจะเมื่อดูข้อมูลที่ท่านต้องการเสร็จแล้ว ย้อนกลับโดยกดปุ่ม **CLOSE** จะย้อนกลับไปข้อ 4

รูปที่ ก.4



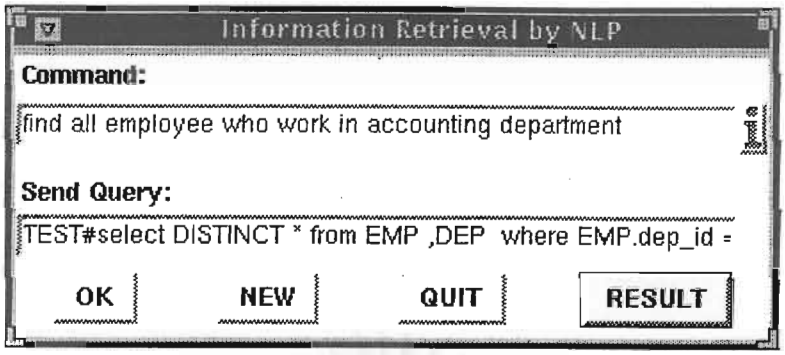
6. ผู้ใช้กรอกคำร้องขอข้อมูลที่ตนต้องการลงในช่องรับข้อความใต้ **COMMAND** ดังรูปที่ ก.5 เสร็จแล้วกดแป้นพิมพ์ **ENTER** หรือ กดปุ่ม **OK** แล้วไปยังข้อ 7

รูปที่ ก.5



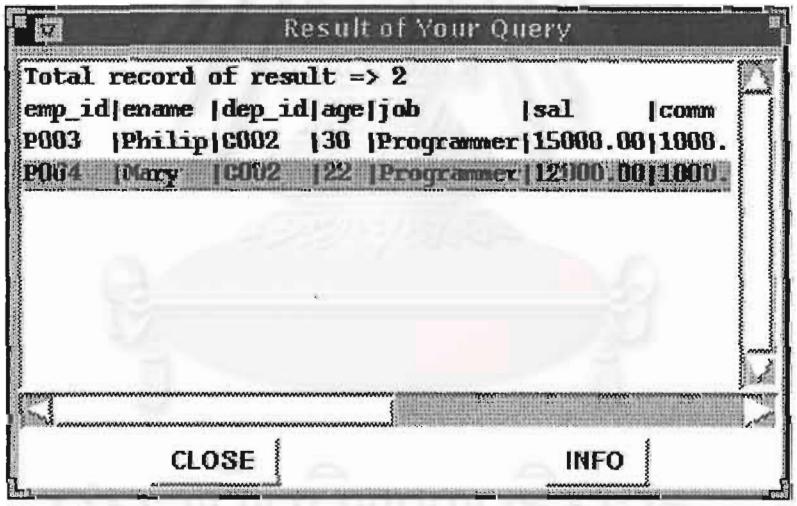
7. ระบบจะทำการแปลงคำร้องขอให้อยู่ในรูปภาษาสอบถามเชิงโครงสร้างแล้วแสดงไว้ในในช่องแสดงข้อความใต้ **SEND QUERY** ดังรูปที่ ก.6 เมื่อผู้ใช้กดปุ่ม **RESULT** แล้วไปยังข้อ 8 กดปุ่ม **NEW** จะกระทำตามข้อ 10 แต่ถ้ามีข้อผิดพลาดเกิดขึ้นจะไปยังข้อ 11 สำหรับคำร้องขอมีปัญหา และไปยังข้อ 12 สำหรับระบบฐานข้อมูลมีปัญหา

รูปที่ ก.6



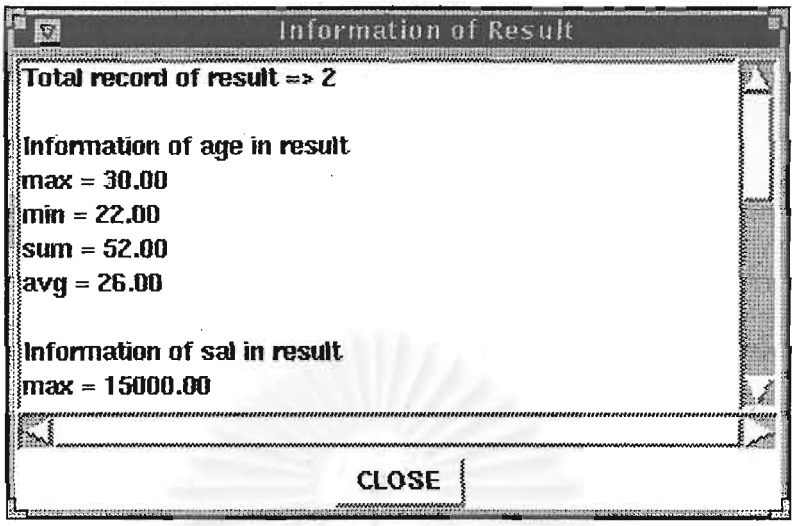
8. ระบบจะส่งคำร้องขอดังกล่าวไปยังฐานข้อมูล แล้วส่งผลลัพธ์ดังกล่าวมาแสดงดังรูปที่ ก.7 ถ้าผู้ใช้ต้องการทราบข้อมูลของผลลัพธ์ที่ได้รับ กดปุ่ม **INFO** จะไปยังข้อ 9 แต่ถ้ากดปุ่ม CLOSE จะกลับไปยังข้อ 7

รูปที่ ก.7



9. รูปที่ ก.8 แสดงผลรวม ค่าเฉลี่ย ค่ามากสุด ค่าน้อยสุด และจำนวนระเบียนของผลลัพธ์ ของเซตข้อมูลที่สามารถคำนวณได้ เมื่อเสร็จสิ้นกดปุ่ม **CLOSE** จะกลับไปยังข้อ 8

รูปที่ ก.8



10. จะกลับไปยังข้อ 1 เพื่อรอรับคำสั่งของผู้ใช้ต่อไป

11. แสดงรูปที่ ก.9 เมื่อกดปุ่ม **OK** จะไปยังข้อ 7

รูปที่ ก.9

12. แสดงรูปที่ ก.10 เมื่อกดปุ่ม **OK** จะไปยังข้อ 7

รูปที่ ก.10

ภาคผนวก ข

ฟังกชันของโปรแกรมต่างๆ ในระบบ

ตาราง ข.1 ฟังก์ชันในโปรแกรมเกี่ยวกับตารางความสัมพันธ์

ชื่อฟังก์ชัน	คำอธิบายฟังก์ชัน
ADD_Q	เพิ่มความสัมพันธ์ที่ปรากฏในเงื่อนไข
ASK_Q	ถามความสัมพันธ์ของสิ่งที่ต้องการทราบกับสิ่งที่ปรากฏในเงื่อนไข
reach	หาความสัมพันธ์เชิงถ่ายทอด
ADD_REL	ตรวจสอบและหาเงื่อนไขความสัมพันธ์ระหว่างตาราง 2 ตาราง
ADD_COND	เพิ่มเงื่อนไขความสัมพันธ์ที่พบจากฟังก์ชัน ADD_REL
RELATE	โปรแกรมหลักควบคุมการหาความสัมพันธ์ของระบบฐานข้อมูล

ตาราง ข.2 ฟังก์ชันในโปรแกรมพจนานุกรมฐานความรู้

ชื่อฟังก์ชัน	คำอธิบายฟังก์ชัน
FORWARD	การเลื่อนไปยังสถานะถัดไปในเส้นทางข้อมูลตัวอักษรที่อ่านเข้ามาทีละหนึ่งตัวอักษร
BACKWARD	การเลื่อนย้อนกลับไปยังสถานะ ในเส้นทางข้อมูลตัวอักษรที่อ่านเข้ามาทีละหนึ่งตัวอักษร
SEPARATE_ CHECK	ตรวจสอบโหนดว่าเป็นโหนดเชื่อมต่อระหว่างทรัยส่วนหน้ากับทรัยส่วนหลัง หรือไม่
LINK_STATE	บอกโหนดเริ่มต้นของทรัยส่วนหลังที่เชื่อมกับโหนดเชื่อมต่อ
READD	ใส่สายอักขระย่อยลงในทรัยส่วนหลัง
FR_INSERT	ใส่ตัวอักษรที่แตกต่างจากเส้นทางข้อมูลอื่น ในทรัยส่วนหน้า และใส่สายอักขระย่อยที่เหลือในทรัยส่วนหลัง
RE_INSERT	ย้ายสายอักขระย่อยในทรัยส่วนหลังที่ซ้ำกับคำข้อมูลที่ใส่มา ไปใน ทรัยส่วนหน้าจนถึงตัวอักษรที่ไม่ซ้ำ แล้วเพิ่มส่วนที่เหลือลงในทรัยส่วนหลัง
Convtoind	แปลงรหัสแอสกีตัวอักษรตัวใหญ่ให้เป็นตัวอักษรตัวเล็ก
DEL_ONE	ลบตัวอักษร ณ. ตำแหน่งนั้น

ตาราง ข.2 ฟังก์ชันในโปรแกรมพจนานุกรมฐานความรู้ (ต่อ)

ชื่อฟังก์ชัน	คำอธิบายฟังก์ชัน
DEL_SOME	ลบเส้นทางสายอักขระที่ไม่ต้องการ
reverse	ผันกลับสายอักขระ
insert	เพิ่มคำข้อมูลลงในพจนานุกรมฐานความรู้
read_chk	ตรวจสอบการค้นหาข้อมูลในพจนานุกรมฐานความรู้
make_INFORM	สร้างโครงสร้างทรัย
make_LINK	สร้างลิสต์ส่วนเชื่อมต่อระหว่างทรัยส่วนหน้า กับทรัยส่วนหลัง
Check_Locate	หาตำแหน่งในลิสต์โครงสร้างทรัย
Chk_Locate	หาตำแหน่งในลิสต์ส่วนเชื่อมต่อทรัยส่วนหน้า กับทรัยส่วนหลัง
ChkU_Locate	หาตำแหน่งในลิสต์รายละเอียดคำข้อมูล
retrieve	ค้นหาคำข้อมูลในพจนานุกรมฐานความรู้
Chk_ses	เพิ่มการหาคำในรูปพหูพจน์
INCR_MODE	เพิ่มค่าตัวนับทางเลือกของโหนด
DECR_MODE	ลดค่าตัวนับทางเลือกของโหนด
OUTDEG	ทางเลือกของโหนด
APP_FIX	หาเนื้อที่สำหรับกลุ่มข้อมูลใหม่ในโครงสร้างทรัย
MOVE_LINK	ย้ายตัวเชื่อมต่อไปยังตำแหน่งใหม่
MODI_LINK	ปรับปรุงตัวเชื่อมต่อ
MODI_STATE	ปรับปรุงโหนด
ADD_INFLNK	เชื่อมโยงระหว่างดัชนีข้อมูลกับรายละเอียดข้อมูล
substr	การตัดสายอักขระในตำแหน่งที่ต้องการ
LINKS	ตรวจสอบจุดเชื่อมต่อของพจนานุกรมฐานความรู้
init_new	ล้างค่าในตัวแปรต่างๆ
do_trie	ควบคุมขั้นตอนการทำงานของพจนานุกรมฐานความรู้
WRITE_HEAD	บันทึกข้อมูลเกี่ยวกับข้อมูลส่วนหัวพจนานุกรมฐานความรู้ลงในแฟ้มข้อมูล
WRITE_INFORM	บันทึกข้อมูลเกี่ยวกับโครงสร้างเส้นทางข้อมูลของพจนานุกรมฐานความรู้ลงในแฟ้มข้อมูล
WRITE_LINK	บันทึกข้อมูลเกี่ยวกับการเชื่อมต่อข้อมูลของพจนานุกรมฐานความรู้ลงในแฟ้มข้อมูล

ตาราง ข.2 ฟังก์ชันในโปรแกรมพจนานุกรมฐานความรู้ (ต่อ)

ชื่อฟังก์ชัน	คำอธิบายฟังก์ชัน
WRITE_INF	บันทึกข้อมูลเกี่ยวกับเนื้อหาข้อมูลของพจนานุกรมฐานความรู้ลงในแฟ้มข้อมูล
save_data	ควบคุมการบันทึกข้อมูลของพจนานุกรมฐานความรู้
READ_HEAD	บรรจุคำข้อมูลที่บันทึกไว้เกี่ยวกับข้อมูลส่วนหัวของพจนานุกรมฐานความรู้
READ_INFORM	บรรจุคำข้อมูลที่บันทึกไว้เกี่ยวกับโครงสร้างเส้นทางข้อมูลคืนสู่พจนานุกรมฐานความรู้
READ_LINK	บรรจุคำข้อมูลที่บันทึกไว้เกี่ยวกับการเชื่อมต่อข้อมูลคืนสู่พจนานุกรมฐานความรู้
READ_INF	บรรจุคำข้อมูลที่บันทึกไว้เกี่ยวกับเนื้อหาข้อมูลคืนสู่พจนานุกรมฐานความรู้
load_data	ควบคุมการบรรจุคำข้อมูลที่บันทึกไว้คืนสู่พจนานุกรมฐานความรู้
startup	ล้างค่าตัวแปรต่างๆ ในระบบ
func_trie	เป็นตัวควบคุมฟังก์ชันต่างๆ เกี่ยวกับการค้นหา และเพิ่มเติมคำข้อมูลลงในพจนานุกรมฐานความรู้
insert_f	เพิ่มแฟ้มคำข้อมูลลงในพจนานุกรมฐานความรู้

ตาราง ข.3 ฟังก์ชันในโปรแกรมจัดการเกี่ยวกับภาษาสอบถามเชิงโครงสร้างของ mSQL

ชื่อฟังก์ชัน	คำอธิบายฟังก์ชัน
Check_mSQL	ตรวจสอบและวิเคราะห์ภาษา mSQL
isOP_SQL	ตรวจสอบและวิเคราะห์ตัวดำเนินการ(operator)
isVALUE	ตรวจสอบและวิเคราะห์คำที่เป็นค่าของเขตข้อมูล
Check_TABLIST	ตรวจสอบและวิเคราะห์ลิสต์ตาราง
Check_FLDLIST	ตรวจสอบและวิเคราะห์ลิสต์เขตข้อมูล
Check_FLD	ตรวจสอบและวิเคราะห์เขตข้อมูล
Check_CONDLIST	ตรวจสอบและวิเคราะห์ลิสต์ชุดเงื่อนไข
Check_COND	ตรวจสอบและวิเคราะห์ชุดเงื่อนไข
Check_ORDBYLIST	ตรวจสอบและวิเคราะห์ลิสต์เขตข้อมูลสำหรับเรียงลำดับผลลัพธ์
make_SQLG	สร้างลิสต์ของข้อมูลพื้นฐานทั่วไป
GET_WORD	ดึงข้อมูลจากแฟ้มข้อมูลทีละคำ

ตาราง ข.3 ฟังก์ชันในโปรแกรมจัดการเกี่ยวกับภาษาสอบถามเชิงโครงสร้างของ mSQL (ต่อ)

ชื่อฟังก์ชัน	คำอธิบายฟังก์ชัน
Make_MSQLCOM	สร้างภาษา mSQL จากโครงสร้างภาษาสอบถามเชิงตรรกะ
Cut_CNDlink	ย้ายชุดเงื่อนไขไปยังโครงสร้างภาษาสอบถามเชิงตรรกะอื่น
Cut_Tblink	ย้ายตารางไปยังโครงสร้างภาษาสอบถามเชิงตรรกะอื่น
Cut_FLDlink	ย้ายเขตข้อมูลไปยังโครงสร้างภาษาสอบถามเชิงตรรกะอื่น
Cut_ORDlink	ย้ายเขตข้อมูลสำหรับการเรียงลำดับผลลัพธ์ไปยังโครงสร้างภาษาสอบถามเชิงตรรกะอื่น
Split_CND	ตรวจสอบชุดเงื่อนที่ข้ามฐานข้อมูล
Split_TB	ตรวจสอบตารางที่ข้ามฐานข้อมูล
Split_FLD	ตรวจสอบเขตข้อมูลที่ข้ามฐานข้อมูล
Split_ORD	ตรวจสอบเขตข้อมูลสำหรับการเรียงลำดับผลลัพธ์ที่ข้ามฐานข้อมูล
Chk_Split	ควบคุมการตรวจสอบการอ้างอิงข้ามฐานข้อมูล

ตาราง ข.4 ฟังก์ชันในโปรแกรมจัดการเกี่ยวกับภาษาสอบถามเชิงโครงสร้างของ SQL

ชื่อฟังก์ชัน	คำอธิบายฟังก์ชัน
make_INFSQL	สร้างโครงสร้างภาษาสอบถามเชิงตรรกะ
make_FLDSQL	สร้างลิสต์ของเขตข้อมูลสำหรับโครงสร้างภาษาสอบถามเชิงตรรกะ
make_CONDSQL	สร้างชุดเงื่อนไขสำหรับโครงสร้างภาษาสอบถามเชิงตรรกะ
make_ORDSQL	สร้างลิสต์ของเขตข้อมูลในการเรียงลำดับผลลัพธ์สำหรับโครงสร้างภาษาสอบถามเชิงตรรกะ
CHK_INF	หาข้อมูลส่วนต่างๆ ของข้อมูลที่ต้องการทราบ
CHK_TYPE	หาประเภทของข้อมูลที่ต้องการทราบ
CHK_WORD	หาคำที่ตรงกับตำแหน่งระบุความสัมพันธ์ของฐานข้อมูล
FILL_TBinFLD	ปรับปรุงเขตข้อมูลที่ไม่สมบูรณ์
FILL_TBinKEY	ปรับปรุงกุญแจหลักที่ไม่สมบูรณ์
Check_SQL	วิเคราะห์และตรวจสอบไวยากรณ์ของภาษาสอบถามเชิงโครงสร้าง
Check_ECONDLIST	วิเคราะห์และตรวจสอบไวยากรณ์ของประโยคชุดเงื่อนไขแบบข่าย
Check_ECOND	วิเคราะห์และตรวจสอบไวยากรณ์ของชุดเงื่อนไขแบบข่าย
Check_Between	วิเคราะห์และตรวจสอบไวยากรณ์ของบุพบทวลี between

ตาราง ข.5 ฟังก์ชันในโปรแกรมจัดการเกี่ยวกับภาษาธรรมชาติ

ชื่อฟังก์ชัน	คำอธิบายฟังก์ชัน
Chk_Clause	วิเคราะห์และตรวจสอบไวยากรณ์ของประโยค
Chk_CondClause	วิเคราะห์และตรวจสอบไวยากรณ์ของประโยคเงื่อนไข
Chk_NounP	วิเคราะห์และตรวจสอบไวยากรณ์ของนามวลี
Chk_VerbP	วิเคราะห์และตรวจสอบไวยากรณ์ของกริยาวลี
Chk_PrepP	วิเคราะห์และตรวจสอบไวยากรณ์ของบุพบทวลี
Chk_AdjP	วิเคราะห์และตรวจสอบไวยากรณ์ของคุณศัพท์วลี
Chk_GRAM	หาจุดเริ่มต้นของการวิเคราะห์ประโยค
OFVALUE	ทำการประเมินและวิเคราะห์เขตข้อมูลที่ได้รับ
GUESS_Blank	ทำการประเมินและวิเคราะห์หาเขตข้อมูลที่ไม่ทราบข้อมูลบางส่วน
Insert_FLD	เพิ่มเขตข้อมูลลงในโครงสร้างเชิงตรรกะของภาษาสอบถามเชิงโครงสร้าง
Insert_TB	เพิ่มตารางลงในโครงสร้างเชิงตรรกะของภาษาสอบถามเชิงโครงสร้าง
Insert_ORD	เพิ่มเขตข้อมูลสำหรับการเรียงลำดับผลลัพธ์ลงในโครงสร้างเชิงตรรกะของภาษาสอบถามเชิงโครงสร้าง
Insert_COND	เพิ่มชุดเงื่อนไขลงในโครงสร้างเชิงตรรกะของภาษาสอบถามเชิงโครงสร้าง
Gram_Op	วิเคราะห์และตรวจสอบเครื่องหมายตัวดำเนินการ
cpystr	คัดลอกสายอักขระ
cmpstr	เปรียบเทียบระหว่างสายอักขระ
Verif_Type	ประเมินและวิเคราะห์ประเภทของคำ ณ. ตำแหน่งปัจจุบัน
Chk_Ungram	วิเคราะห์และตรวจสอบประโยคที่ไม่ตรงรูปแบบไวยากรณ์ที่มีอยู่

ตาราง ข.6 ฟังก์ชันในโปรแกรมที่จัดการเกี่ยวกับการติดต่อระหว่างผู้ให้บริการกับผู้ใช้บริการ

ชื่อฟังก์ชัน	คำอธิบายฟังก์ชัน
readline	อ่านข้อมูลจากซอคเก็ต โดยมีการตรวจสอบความผิดพลาดที่เกิดจากการอ่านข้อมูลด้วย
written	เขียนข้อมูลที่ต้องการส่งเช่น คำร้องขอ ผลลัพธ์ที่ได้จากคำร้องขอ ข้อความที่ใช้ยืนยันการรับส่งข้อมูล เป็นต้น โดยส่งข้อมูลผ่านซอคเก็ตและตรวจสอบความผิดพลาดของการเขียนข้อมูลลงซอคเก็ตด้วย
str_def	ทำหน้าที่ติดต่อไปยัง mSQL เพื่อขอทราบโครงสร้างข้อมูลต่างๆ ในระบบฐานข้อมูล และตรวจสอบความผิดพลาดระหว่างการติดต่อ
str_query	ทำหน้าที่จัดรูปแบบคำร้องขอที่ถูกส่งมาจากผู้ใช้บริการให้อยู่ในรูปภาษาสอบถามเชิงโครงสร้างของ mSQL แล้วทำการติดต่อไปยัง mSQL เพื่อสอบถามข้อมูลต่างๆ ในระบบฐานข้อมูล เมื่อได้ผลลัพธ์จะทำการจัดรูปแบบเพื่อส่งไปยังผู้ใช้บริการและตรวจสอบความผิดพลาดระหว่างการติดต่อ
rev_def	อ่านผลลัพธ์โครงสร้างข้อมูลที่ส่งมาจากผู้ให้บริการ จัดเรียงผลลัพธ์ให้อยู่ในรูปแบบที่ต้องการเพื่อรอการนำเสนอยังผู้ใช้ระบบ และตรวจสอบความผิดพลาดระหว่างการติดต่อ
rev_que	อ่านผลลัพธ์ข้อมูลที่ได้ส่งมาจากผู้ให้บริการ จัดเรียงผลลัพธ์ให้อยู่ในรูปแบบที่ต้องการเพื่อรอการนำเสนอยังผู้ใช้ระบบ และตรวจสอบความผิดพลาดระหว่างการติดต่อ

ตาราง ข.7 ฟังก์ชันในโปรแกรมที่จัดการเกี่ยวกับการผสานผลลัพธ์

ชื่อฟังก์ชัน	คำอธิบายฟังก์ชัน
open_file	ทำการเปิดแฟ้มข้อมูลที่ต้องการและตรวจสอบข้อผิดพลาดที่เกิดขึ้น
make_linkdb	สร้างโครงสร้างสำหรับชุดเงื่อนไขที่เชื่อมระหว่างฐานข้อมูล
make_DBHD	สร้างโครงสร้างสำหรับฐานข้อมูลที่จะผสาน
make_FLDHD	สร้างโครงสร้างสำหรับเขตข้อมูลที่จะผสาน
get_line	อ่านข้อมูลจากแฟ้มข้อมูลทีละบรรทัด
summary	ทำการเก็บรวบรวมข้อมูลเพื่อสรุป
prnsum	เขียนข้อมูลที่สรุปได้ลงในแฟ้มสรุปข้อมูล

ตาราง ข.7 ฟังก์ชันในโปรแกรมที่จัดการเกี่ยวกับการผสานผลลัพธ์ (ต่อ)

ชื่อฟังก์ชัน	คำอธิบายฟังก์ชัน
merge_result	เขียนผลลัพธ์ที่ตรงตามทุกชุดเงื่อนไขลงแฟ้มผลลัพธ์
just_fld	จัดรูปแบบเขตข้อมูลใส่โครงสร้างข้อมูลที่สร้างไว้
just_val	จัดรูปแบบค่าเขตข้อมูลใส่โครงสร้างข้อมูลที่สร้างไว้
chk_like	ตรวจสอบเงื่อนไขที่ใช้ like
link_keydb	ทำการเชื่อมโครงสร้างชุดเงื่อนไขกับโครงสร้างข้อมูลที่อ่านขึ้นมา
chk_cond	ตรวจสอบชุดเงื่อนไขทั้งหมดกับข้อมูลที่อ่านขึ้นมา
get_msql	แยกภาษาสอบถามเชิงโครงสร้างตามฐานข้อมูล
just_result	จัดรูปแบบผลลัพธ์ให้เท่ากับขนาดความยาวของค่าเขตข้อมูลที่มากสุด
add_result	ทำการเก็บรวบรวมข้อมูลจากขั้นตอนต่างๆ

ประวัติผู้วิจัย

นายพงษ์ปัญญา จงจักรพันธ์ เกิดวันที่ 15 มกราคม พ.ศ. 2516 ที่กรุงเทพมหานคร สำเร็จการศึกษาระดับปริญญาตรีวิทยาศาสตรบัณฑิต สาขาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2535 และเข้าศึกษาต่อในหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย เมื่อ พ.ศ. 2539