ระบบประมวลผลคำถาม สำหรับฐานข้อมูลแบบกระจาย ในโครงข่ายคอมพิวเตอร์ TCP/IP

A DISTRIBUTED DATABASE QUERY PROCESSING SYSTEM IN A TCP/IP NETWORK

นาย สมชิด ศิริผลหลาย

MR. SOMCHID SIRIPONLAI

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิศวกรรมไฟฟ้า
บัณฑิตวิทยาลัย
สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหารลาดกระบัง
พ.ศ. 2536

ISBN 974-621-014-9

A DISTRIBUTED DATABASE QUERY PROCESSING SYSTEM IN A TCP/IP NETWORK

SOMCHID SIRIPONLAI

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING

KING MONGKUT 'S INSTITUTE OF TECHNOLOGY LADKRABANG

1993

ISBN 974-621-014-9

หัวข้อวิทยานิพนธ์	ระบบประมวลผลคำถาม สำหรับฐานข้อมูลแบบกระจาย ในโครงข่ายคอมพิวเตอร์ TCP/IP
นักศึกษา	นายสมชิด ศิริผลหลาย
อาจารย์ผู้ควบคุมวิทยานิพนธ์	ผศ. ดร. ศุภมิตร จิตตะยโศธร
ระดับการศึกษา	วิศวกรรมศาสตรมหาบัณฑิต สาขาวิศวกรรมไฟฟ้า
ภาควิชา	วิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหาร ลาดกระบัง
พ.ศ.	2536

บทคัดย่อ

ในปัจจุบันนี้ ระบบจัดการฐานข้อมูลมีบทบาทอย่างมาก ทั้งในระบบงานขนาดเล็ก และขนาดใหญ่ จากปริมาณการใช้งานที่เพิ่มขึ้น ทำให้มีความต้องการเชื่อมต่อ ระบบฐานข้อมูลหลายระบบเข้าด้วยกัน เป็นฐานข้อมูลแบบกระจาย (Distributed Database System) เพื่อให้ผู้ใช้ระบบฐานข้อมูลหนึ่ง สามารถเข้าถึง (access) ข้อมูลในระบบฐานข้อมูลอื่นๆได้ ระบบฐานข้อมูลแบบกระจายนี้ จำเป็นจะต้องมีส่วนเพิ่มเติมพิเศษขึ้นมาจากระบบจัดการฐานข้อมูลธรรมดา

วิทยานิพนธ์นี้เสนอ การออกแบบและสร้างระบบประมวลผลคำถาม สำหรับเชื่อมต่อระบบจัดการการฐานข้อมูล ออราเคิล (Oracle) หลายระบบ ซึ่งแต่ละระบบทำงานโดยอิสระได้อยู่แล้ว ให้สามารถปฏิบัติงานร่วมกันได้ เป็นระบบฐานข้อมูลแบบกระจายในโครงข่ายคอมพิวเตอร์ที่ใช้โปรโตคอล TCP/IP ภายใต้ระบบปฏิบัติการยูนิกซ์ (UNIX) โดยไม่จำเป็นต้องแก้ไขระบบฐานข้อมูลที่มีอยู่แล้วแต่ประการใด สถาปัตยกรรมของระบบนี้ สามารถประยุกต์ให้ใช้งานได้กับระบบปฏิบัติการ (operating system) ชนิดอื่น และระบบโครงข่ายคอมพิวเตอร์ (computer network) ที่ใช้โปรโตคอลลักษณะอื่น รวมทั้งระบบจัดการฐานข้อมูลแบบรีเลชันแนลอื่นๆได้เช่นกัน

Thesis Title	A Distributed Database Query Processing System in a TCP/IP Network
Student	Mr. Somchid Siriponlai
Thesis Advisor	Asst. Prof. Dr. Suphamit Chittayasothorn
Level of Study	Master of Engineering in Electrical Engineering
Department	Computer Engineering Faculty of Engineering King Mongkut 's Institute of Technology Ladkrabang
Year	1993

ABSTRACT

Nowadays, database management systems (DBMS) play important roles in small and large organizations. Because of large amounts and various sources of information, a database system is frequently connected to other systems. When a connection is made, the user in a database system can access data in other systems. In order to establish the connection and form a distributed database system, the local DBMSs must be modified.

This thesis presents the design and construction of a database query processing system which is built on top of Oracle database management system that runs on UNIX machines in a TCP/IP network without any modification to the DBMS. This architecture can be applied to other similar environment as well

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้ สำเร็จลงได้ด้วยดี ก็เพราะได้รับความกรุณา และเอาใจใส่จาก ผู้ช่วยศาสตราจารย์ ดร. ศุภมิตร จิตตะยโศธร ที่ได้ให้คำแนะนำ และเป็นผู้สร้างกำลังใจให้ผู้วิจัยตลอดมา ผู้วิจัยซาบซึ้งและขอกราบขอบพระคุณเป็นอย่างสูง

ขอขอบคุณ บุคคลากรแผนกสารสนเทศ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหารลาดกระบัง ที่ได้ให้ความช่วยเหลือผู้วิจัย ในการทำวิทยานิพนธ์ เป็นอย่างดี

ขอขอบคุณ บริษัทไมโครเอกส์จำกัด ที่ได้ให้ข้อมูล และเอกสารเกี่ยวกับระบบจัดการฐานข้อมูลอินเกส์

ขอขอบคุณ คุณธนา หงษ์สุวรรณ คุณธวัชชัย สุทธิทศธรรม และคุณนภพล ใจดี ที่ได้ให้คำปรึกษา และกำลังใจ แก่ผู้วิจัยมาโดยตลอด

สุดท้ายนี้ ผู้วิจัยขอขอบพระคุณบิดาและมารดา ที่ได้ให้การเลี้ยงดูผู้ทำวิจัยโดยตลอดมา และให้ทุนสนับสนุนการทำวิจัยในครั้งนี้ อีกทั้งเป็นบุคคลที่สร้างกำลังใจแก่ผู้วิจัยเสมอมา

นายสมชิด ศิริผลหลาย

สารบัญ

	หน้า
บทคัดย่อภาษาไทย	I
บทคัดย่อภาษาอังกฤษ	II
กิตติกรรมประกาศ	III
สารบัญ	IV
บทที่ 1 บทนำ	1
1.1 ความรู้ทั่วไป	1
1.2 ส่วนประกอบของวิทยานิพนธ์	2
บทที่ 2 ทฤษฎีพีชคณิตสัมพันธ์	3
2.1 แนะนำหลักการ	3
2.2 ไวยากรณ์พีชคณิตสัมพันธ์	6
2.2.1 ข้อสังเกตเกี่ยวกับไวยากรณ์	6
2.3. ตัวปฏิบัติการเบื้องต้น	7
2.3.1 UNION	7
2.3.2 INTERSECTION	8
2.3.3 DIFFERENCE	8
2.3.4 EXTENDED CARTESIAN PRODUCT	9
2.4 ตัวปฏิบัติการความสัมพันธ์พิเศษ	10
2.4.1 SELECTION	10
2.4.2 PROJECTION	11
2.4.3 JOIN	11
2.4.4 DIVISION	12
2.5 ตัวอย่างการใช้งานพีชคณิตสัมพันธ์	13
บทที่ 3 หลักการของระบบฐานข้อมูลแบบกระจาย	15
3.1 ความรู้เบื้องต้น	15
3.1.1 ความหมายของฐานข้อมูลแบบกระจาย	15
3.1.2 ระบบจัดการฐานข้อมูลแบบกระจาย	18
3.2 ระดับการส่งผ่านการกระจายของฐานข้อมูล	21

3.2.1 แบบแผนโดยรวม 22
3.2.2 แบบแผนส่วนย่อย 22
3.2.3 แบบแผนการอ้างถึงตำแหน่ง 22
3.3 การออกแบบฐานข้อมูลแบบกระจาย 22
3.3.1 การออกแบบฐานข้อมูลจากบนลงล่าง 23
3.3.2 การออกแบบฐานข้อมูลจากล่างขึ้นบน 23
3.3.3 การออกแบบฐานข้อมูลในระบบ DDQ/1 23
3.4 การแปลงคำถาม 24
3.4.1 การเปลี่ยนรูปที่เท่ากันสำหรับพีชคณิตสัมพันธ์ 25
3.4.2 คำถามในรูปตัวปฏิบัติการเชิงต้นไม้ 28
3.5 การเข้าถึงข้อมูลที่ดีที่สุด 30
3.5.1 การกำหนดสถานที่เก็บข้อมูล 30
3.5.2 การปรับปรุงรูปคำถามเชิงต้นไม้ 31
บทที่ 4 สถาปัตยกรรมของระบบประมวลผลคำถามแบบกระจาย DDQ/1 37
4.1 สถาปัตยกรรมของระบบ DDQ/1 โดยรวม 37
4.1.1 ส่วนจัดการกระบวนการเริ่มต้น 37
4.1.2 ส่วนติดต่อผู้ใช้งานน 39
4.1.3 ส่วนประมวลผลคำถามแบบกระจาย 39
4.1.4 ส่วนเชื่อมต่อโครงข่าย 40
4.2 ส่วนจัดการกระบวนการเริ่มต้น 41
4.2.1 ส่วนพจนานุกรมฐานข้อมูล 42
4.2.2 ส่วนป้อนกฎบังคับความถูกต้อง 46
4.3 ส่วนติดต่อผู้ใช้งาน 46
4.4 ส่วนประมวลผลคำถามแบบกระจาย 47
4.5 ส่วนเชื่อมต่อโครงข่าย 49
บทที่ 5 การพัฒนาระบบฐานข้อมูลแบบกระจาย DDQ/1 50
5.1 ฮาร์ดแวร์และซอฟท์แวร์ที่ใช้ 50
5.2 การฝังคำสั่งภาษาสอบถามเชิงโครงสร้างในโปรแกรม 51
5.2.1 ความสัมพันธ์ระหว่างพีชคณิตสัมพันธ์กับภาษาสอบถามเชิงโครงสร้าง 51
5.2.2 การฝังคำสั่งภาษาสอบถามเชิงโครงสร้างแบบสถิตย์ 54

5.2.3 การฝังคำสั่งภาษาสอบถามเชิงโครงสร้างแบบจลน์ 61
5.3 การใช้คำสั่งเรียกใช้ TCP/IP 68
5.3.1 การใส่เปลือกครอบ 70
5.3.2 ที่อยู่แบบอินเทอร์เนท 71
5.3.3 หมายเลขพอร์ท 72
5.3.4 การเรียกใช้ฟังก์ชันสำหรับโปรโตคอล TCP/IP 72
5.4 หน่วยโปรแกรมที่พัฒนาขึ้น 78
5.4.1 ส่วนติดต่อผู้ใช้งาน 79
5.4.2 ส่วนประมวลผลคำถามแบบกระจาย 89
5.4.3 ส่วนจัดการส่งข้อมูลผ่านโครงข่ายสื่อสาร 105
5.4.4 ส่วนสร้างตารางพจนานุกรมฐานข้อมูล 106
5.4.5 ส่วนป้อนข้อมูลพจนานุกรมฐานข้อมูล 106
5.4.6 ส่วนป้อนกฎบังคับความถูกต้อง 111
5.4.7 ส่วนเครื่องมือ 113
บทที่ 6 การเปรียบเทียบคุณสมบัติของระบบ DDQ/1 กับระบบจัดการฐานข้อมูลอื่น 117
6.1 คุณสมบัติที่สำคัญของระบบจัดการฐานข้อมูลแบบกระจาย 117
6.1.1 การประมวลผลคำถามด้วยรูปแบบคาโนนิคัล 117
6.1.2 คำถามไม่ขึ้นกันสถานที่ 117
6.1.3 ไม่มีการแก้ไขระบบจัดการฐานข้อมูล 118
6.1.4 การอนุญาตโดยรวม 118
6.1.5 การประมวลผลคำถามโดยรวมที่ได้ผลดีที่สุด 118
6.1.6 กฎบังคับความถูกต้องโดยรวม 119
6.1.7 พจนานุกรมฐานข้อมูลโดยรวม 119
6.2 ระบบจัดการฐานข้อมูลอินเกรส์ 119
6.2.1 สถาปัตยกรรมกระบวนการประมวลผลของระบบจัดการฐานข้อมูลอินเกรส์ 120
6.2.2 สถาปัตยกรรมการเก็บข้อมูลของระบบจัดการฐานข้อมูลอินเกรส์ 121
6.2.3 สถาปัตยกรรมของระบบจัดการฐานข้อมูลแบบกระจายอินเกรส์ 122
6.3 การเปรียบเทียบระบบจัดการฐานข้อมูลแบบกระจายอินเกรส์กับระบบ DDQ/1 123
6.3.1 การประมวลผลคำถามด้วยรูปแบบคาโนนิคัล 123

6.3.2 คำถามไม่ขึ้นกันสถานที่ 123
6.3.3 ไม่มีการแก้ไขระบบจัดการฐานข้อมูล 123
6.3.4 การอนุญาตโดยรวม 124
6.3.5 การประมวลผลคำถามโดยรวมที่ได้ผลดีที่สุด 124
6.3.6 กฎบังคับความถูกต้องโดยรวม 124
6.3.7 พจนานุกรมฐานข้อมูลโดยรวม 125
6.4 ผลการทดลองใช้ระบบ DDQ/1 125
บทที่ 7 สรุปและแนวทางการวิจัย 128
7.1 สรุป 128
7.2 แนวทางการวิจัย 129
เอกสารอ้างอิง 131
ภาคผนวก ก ตัวอย่างโปรแกรมที่ติดต่อสื่อสารด้วยโปรโตคอล TCP/IC 134
ภาคผนวก ข การเขียนโปรแกรม LEX 139
ภาคผนวก ค การใช้เครื่องมือซอฟท์แวร์ YACC 142
ภาคผนวก ง โปรแกรม DDQ/1 145
1. ส่วนติดต่อผู้ใช้งาน 145
2. ส่วนประมวลผลคำถามแบบกระจาย 158
3. ส่วนเชื่อมต่อโครงข่าย 221
4. ส่วนสร้างและลบตารางพจนานุกรมฐานข้อมูล 224
5. ส่วนป้อนข้อมูลพจนานุกรมฐานข้อมูล 227
6. ส่วนป้อนกฎบังคับความถูกต้อง 241
7. ส่วนเครื่องมือโปรแกรม 250
ภาคผนวก จ โปรโตคอลระดับสูงที่ใช้ในระบบ DDQ/1 264
ประวัติผู้เขียน 267

บทที่ 1
บทนำ

1.1 ความรู้ทั่วไป

นับวันข่าวสารและข้อมูลจะถูกนำมาใช้ให้เป็นประโยชน์มากขึ้น ทั้งในองค์กรของรัฐและเอกชน ดังนั้นจึงมีความจำเป็นที่จะใช้ระบบจัดการฐานข้อมูล เพื่อค้นหาและจัดเก็บข้อมูลที่มีอยู่ให้เป็นไปด้วยความรวดเร็วและถูกต้อง รวมทั้งบำรุงรักษาข้อมูลให้อยู่ในสภาพพร้อมที่จะใช้งาน

อย่างไรก็ตาม ระบบจัดการฐานข้อมูลแบบรวม (Centralized database management system) อาจจะไม่เหมาะสมกับการใช้งานในบางลักษณะ ทั้งนี้อาจเนื่องมาจากสภาพทางภูมิศาสตร์ หรือ ความจำกัดทางกายภาพของระบบจัดการฐานข้อมูลแบบรวม จึงมีการนำระบบจัดการฐานข้อมูลแบบกระจาย (Distributed database management system) มาใช้ ซึ่งระบบจัดการฐานข้อมูลแบบนี้ จะมีการจัดเก็บข้อมูลไว้ตามส่วนต่างๆของระบบ ทั้งนี้ขึ้นอยู่กับการใช้งานของข้อมูล

ปัจจุบันนี้ มีบริษัทต่างๆได้ผลิตโปรแกรม ที่มีความสามารถในการเชื่อมต่อระบบฐานข้อมูลแบบรวมให้เป็น ระบบฐานข้อมูลแบบกระจาย เช่น INGRES/NET [1,2] สามารถเชื่อมต่อระบบฐานข้อมูลที่ใช้ระบบจัดการฐานข้อมูลอินเกรส (INGRES) เข้าด้วยกัน เป็นระบบฐานข้อมูลแบบกระจาย , SQL*NET [3] เป็นโปรแกรมที่เชื่อมต่อระบบฐานข้อมูลที่ใช้ระบบจัดการฐานข้อมูลออราเคิล ให้เป็นระบบฐานข้อมูลแบบกระจาย จะเห็นว่า โปรแกรมต่างๆเหล่านี้ จะถูกผลิตมาเพื่อสนับสนุนระบบจัดการฐานข้อมูลของบริษัทนั้นๆโดยเฉพาะ

ในการเชื่อมต่อระบบจัดการฐานข้อมูลแบบรวม ให้เป็นระบบจัดการฐานข้อมูลแบบกระจายนั้น สามารถที่จะเชื่อมต่อได้ด้วยระบบโครงข่ายคอมพิวเตอร์ ที่ส่งข้อมูลได้ในอัตราที่สูง และเลือกใช้กรรมวิธีในการเชื่อมต่อที่แตกต่างกันออกไป เพื่อให้มีการประมวลผลเร็วที่สุด ซึ่งกรรมวิธีเหล่านี้มีการคิดค้น และรวบรวมไว้มากมาย เช่น การใช้กฎบังคับความถูกต้อง[6] การปรับปรุงคำถาม[7,8] การกำหนดสถานที่ของการประมวลผลคำถาม[9] เป็นต้น

ในบทความนี้ได้นำเสนอ การออกแบบและสร้างระบบประมวลผลคำถาม สำหรับเชื่อมต่อระบบจัดการฐานข้อมูลแบบรวม ที่สามารถทำงานเป็นระบบจัดการฐานข้อมูลได้โดยอิสระ ซึ่งในที่นี้ใช้ระบบจัดการฐานข้อมูลออราเคิล ให้สามารถทำงานเป็นระบบจัดการฐานข้อมูลแบบกระจายได้ในโครงข่ายคอมพิวเตอร์ที่ใช้โปรโตคอลทีซีพี/ไอพี (TCP/IP) ภายใต้ระบบปฏิบัติการยูนิกซ์ (UNIX) โดยมีการวิเคราะห์คำถามและข้อมูล เพื่อให้ใช้เวลาในการประมวลผลน้อยที่สุด [4,5] โดยมีการตรวจสอบคำถามด้วยกฎบังคับความถูกต้อง , การปรับปรุงคำถาม และ การกำหนด

สถานที่ (site) ในการประมวลผลคำถาม โดยจะเรียกระบบที่ได้พัฒนาอย่างย่อๆว่า " DDQ/1 " ซึ่งในรายละเอียดจะได้กล่าวในบทต่อไป

เพื่อให้เข้าใจถึงขั้นตอนการประมวลผลคำถาม จึงขอใช้ภาษาพีชคณิตสัมพันธ์ (Relational Algebra)[10] เป็นภาษาในการแสดงคำถามในวิทยานิพนธ์ฉบับนี้ รวมทั้งใช้เป็นภาษาในการติดต่อระหว่างผู้ใช้งานกับระบบ DDQ/1 ที่ได้พัฒนาขึ้น

1.2 ส่วนประกอบของวิทยานิพนธ์

วิทยานิพนธ์นี้ได้แบ่งออกเป็นหลายบทด้วยกัน ซึ่งจะประกอบไปด้วย ทฤษฎีเบื้องต้นที่จะทำความเข้าใจระบบฐานข้อมูลแบบกระจาย การออกแบบและการพัฒนาระบบ และได้แสดงคำสั่งในการเขียนโปรแกรมไว้ด้วย โดยแบ่งเป็นบทต่างๆดังนี้คือ

บทที่ 2 ทฤษฎีพีชคณิตสัมพันธ์ ในบทนี้จะกล่าวถึงไวยากรณ์และการใช้พีชคณิตสัมพันธ์พร้อมทั้งยกตัวอย่างการใช้งาน ซึ่งจะใช้ในการเข้าถึงข้อมูล

บทที่ 3 หลักการของฐานข้อมูลแบบกระจาย บทนี้จะกล่าวถึงระบบฐานข้อมูลแบบกระจาย และหลักการทั่วไปของระบบฐานข้อมูลแบบกระจาย

บทที่ 4 สถาปัตยกรรมของระบบ DDQ/1 ในบทนี้กล่าวถึงรายละเอียดโครงสร้างของระบบทั้งหมด พร้อมทั้งตัวอย่างการทำงาน

บทที่ 5 การพัฒนาระบบ DDQ/1 ในบทนี้จะกล่าวถึง ฮาร์ดแวร์และซอฟท์แวร์ที่ใช้ รวมถึงวิธีการพัฒนาโปรแกรม

บทที่ 6 การประเมินผลของระบบ DDQ/1 ซึ่งจะเป็นการเปรียบเทียบระบบ DDQ/1 กับระบบที่ใช้กันอยู่ทั่วไป

บทที่ 7 สรุปและแนวทางการวิจัย บทนี้จะสรุปผลการทำงานและอุปสรรคในระหว่างการทำงาน รวมทั้งแนวทางในการทำวิจัยต่อไป

ส่วนท้ายของวิทยานิพนธ์จะเป็นภาคผนวก โดยได้รวบรวมเรื่องต่างๆที่เกี่ยวข้อง หรือมีส่วนที่ช่วยในการทำความเข้าใจในเนื้อหาของวิทยานิพนธ์ และได้รวบรวมโปรแกรมที่ได้พัฒนาทั้งหมดไว้ในส่วนนี้ด้วย

บทที่ 2
ทฤษฎีพีชคณิตสัมพันธ์

2.1 แนะนำหลักการ

ปัจจุบันระบบฐานข้อมูลที่ใช้กันอยู่ มีแนวโน้มที่จะเปลี่ยนแปลงเป็นระบบฐานข้อมูลแบบสัมพันธ์มากขึ้น[10] เนื่องจากการอ้างถึงข้อมูลในระบบฐานข้อมูลแบบสัมพันธ์ จะมีลักษณะเป็นตาราง (Table) โดยอาศัยความสัมพันธ์ของข้อมูล เป็นตัวกำหนดลักษณะของตาราง ซึ่งการอ้างถึงข้อมูลในลักษณะนี้ จะทำให้สามารถควบคุมความถูกต้องของข้อมูล และแก้ปัญหาการเก็บข้อมูลที่ซ้ำซ้อนได้ [10] การเข้าถึง (access) ข้อมูลของผู้ใช้สามารถที่จะกระทำได้ ด้วยคำถาม (query) ในรูปของภาษายุคที่ 4 (4GL) ที่ระบบจัดการฐานข้อมูลแบบสัมพันธ์นั้นๆ สามารถจะเข้าใจได้ เช่น SQL , QBE เป็นต้น

สำหรับคำถามในรูปภาษายุคที่ 4 นั้น ผู้ใช้จะเข้าถึงข้อมูล ด้วยการระบุเพียงข้อมูลที่ต้องการเท่านั้น โดยไม่ต้องมีการกำหนดวิธีการเข้าถึงข้องมูล ระบบจัดการฐานข้อมูลแบบสัมพันธ์จะทำการค้นหาข้อมูลให้ ด้วยวิธีการต่างๆ ตามลักษณะของระบบฐานข้อมูล และความสามารถของระบบจัดการฐานข้อมูลนั้นๆมีอยู่ จะเห็นว่า คำถามในรูปภาษาในยุคที่ 4 นี้ ไม่ต้องมีการกำหนดขั้นตอนการประมวลผลคำถาม

อย่างไรก็ตาม การวิจัยครั้งนี้ได้เลือกใช้ภาษาพีชคณิตสัมพันธ์ (Relational Algebra) ซึ่งเป็นภาษาแสดงคำถามในการเข้าถึงข้อมูล ที่สามารถจะกำหนดขั้นตอนการประมวลผลคำถามได้ อีกทั้งภาษาพีชคณิตสัมพันธ์นี้ ยังเป็นพื้นฐานของคำถามในรูปภาษายุคที่ 4 ด้วย[4] ซึ่งรายละเอียดของภาษาพีชคณิตสัมพันธ์จะได้กล่าวต่อไป

ภาษาพีชคณิตสัมพันธ์ ถูกนำเสนอโดย E. F. Codd เมื่อปี พ. ศ. 2515 [4,11] เป็นภาษาที่สามารถแสดงความสัมพันธ์ของข้อมูลที่สร้างขึ้นใหม่ ในรูปของความสัมพันธ์ของข้อมูลที่มีอยู่เดิมกับตัวปฏิบัติการ (Operation) ซึ่งตัวปฏิบัติการนี้มีทั้งหมด 8 ชนิดด้วยกัน โดยสามารถจะแบ่งตัวปฏิบัติการออกเป็น 2 กลุ่มได้ดังนี้

1. กลุ่มปฏิบัติการเบื้องต้น ได้แก่ UNION , INTERSECTION , DIFFERENCE , CARTESIAN PRODUCT
2. กลุ่มปฏิบัติการพิเศษ ได้แก่ SELECT , PROJECT , JOIN , DIVIDE

การทำงานของตัวปฏิบัติการ ทั้ง 8 ชนิด ได้แสดงไว้ในรูปที่ 2.1 ซึ่งอธิบายโดยสังเขปได้ดังต่อไปนี้

SELECT คือการสร้างความสัมพันธ์ จากการเลือกเอาข้อมูล TUPLES ที่มีคุณสมบัติตรงตามที่ระบุ จากความสัมพันธ์ที่มีอยู่แล้ว

PROJECT คือการสร้างความสัมพันธ์ จากความสัมพันธ์ที่ระบุถึง โดยเลือกเอาข้อมูลเฉพาะ ATTRIBUTE ที่กำหนด

PRODUCT คือการสร้างความสัมพันธ์ จากความสัมพันธ์สองความสัมพันธ์ ซึ่งความสัมพันธ์ที่สร้างขึ้นใหม่นี้ จะประกอบไปด้วยข้อมูลทั้งหมด ที่เกิดจากการนำข้อมูลใน TUPLES ของทั้งสองความสัมพันธ์มาจับคู่กัน

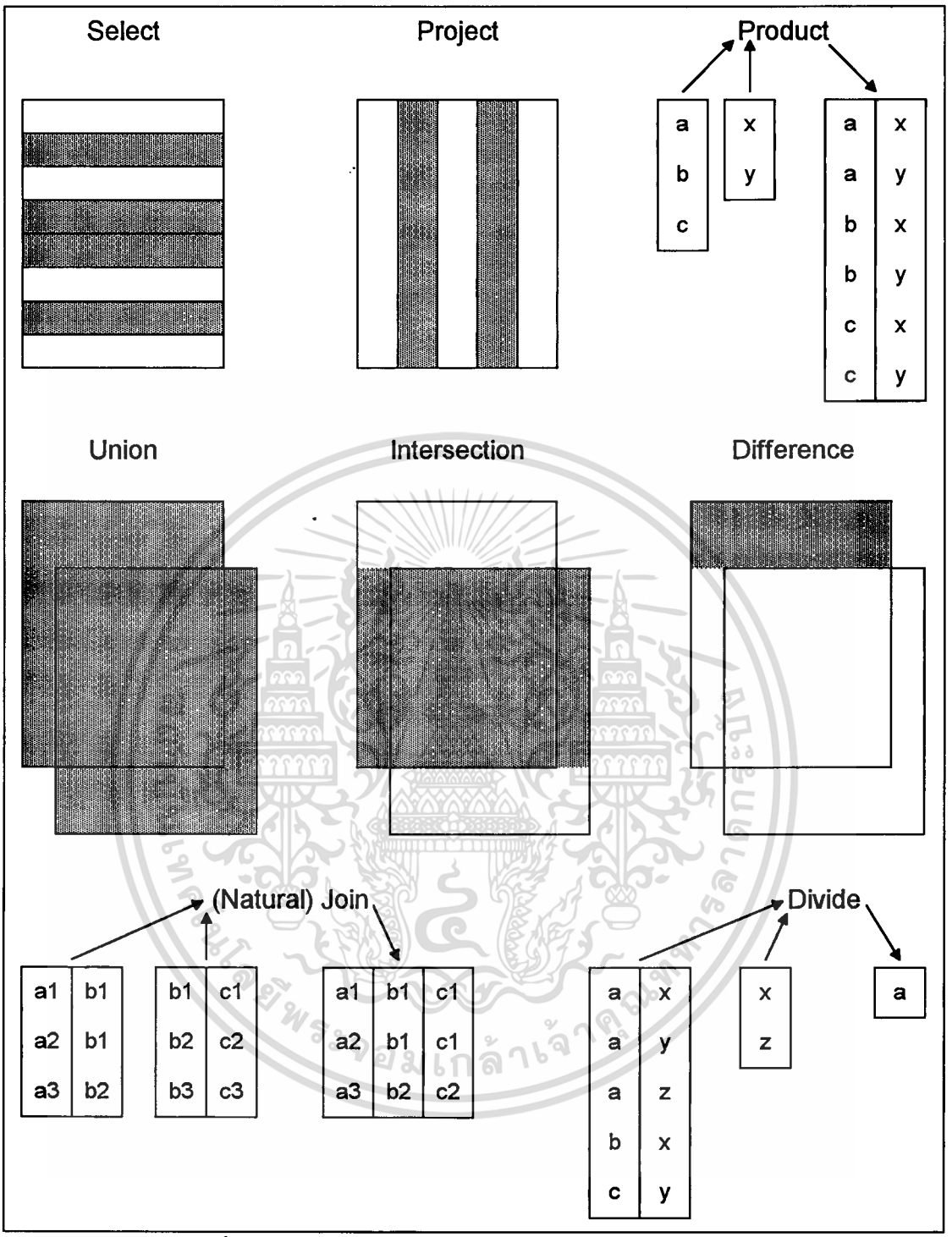
UNION คือการสร้างความสัมพันธ์ขึ้นใหม่ โดยการรวมข้อมูลของสองความสัมพันธ์ที่มีโดเมนเดียวกัน

INTERSECTION คือการสร้างความสัมพันธ์ จากความสัมพันธ์สองความสัมพันธ์ โดยข้อมูลของความสัมพันธ์ที่สร้างขึ้น จะประกอบไปด้วยข้อมูลที่อยู่ในทั้งสองความสัมพันธ์เดิม

DIFFERENCE คือการสร้างความสัมพันธ์ จากความสัมพันธ์สองความสัมพันธ์ โดยข้อมูลของความสัมพันธ์ที่สร้างขึ้นใหม่ ประกอบไปด้วยข้อมูลที่อยู่ในความสัมพันธ์แรกและไม่อยู่ในความสัมพันธ์ที่สอง

JOIN คือการสร้างความสัมพันธ์ จากความสัมพันธ์สองความสัมพันธ์ โดยข้อมูลของความสัมพันธ์ที่สร้างขึ้นใหม่ ประกอบไปด้วยข้อมูลที่เกิดจากการจับคู่ของ TUPLES ของความสัมพันธ์ทั้งสองที่ระบุ ภายใต้เงื่อนไขอย่างใดอย่างหนึ่ง

DIVIDE คือการสร้างความสัมพันธ์ขึ้นใหม่จากความสัมพันธ์ 2 ชนิด คือ ความสัมพันธ์ชนิด BINARY และความสัมพันธ์แบบ UNARY ซึ่งข้อมูลของความสัมพันธ์ที่สร้างขึ้นใหม่จะประกอบด้วย ข้อมูลของความสัมพันธ์แรก คือความสัมพันธ์ชนิด BINARY ใน ATTRIBUTE ที่ไม่ร่วมกัน (non-common) กับ ATTRIBUTE ของความสัมพันธ์ที่สอง คือความสัมพันธ์ชนิด UNARY โดยที่ข้อมูลใน ATTRIBUTE ที่ร่วมกันของความสัมพันธ์ชนิด BINARY จะเหมือนกับข้อมูลทั้งหมดในความสัมพันธ์แบบ UNARY



รูปที่ 2.1 แสดงการทำงานของตัวปฏิบัติการพีชคณิตสัมพันธ์

ซึ่งจะเห็นว่า ผลลัพธ์ของตัวปฏิบัติการพีชคณิตสัมพันธ์ทั้ง 8 ชนิดนี้ ยังคงเป็นความสัมพันธ์อยู่ นั่นคือ ตัวปฏิบัติการพีชคณิตสัมพันธ์ มีคุณสมบัติปิด ด้วยคุณสมบัติปิดนี้เองทำให้สามารถที่จะสร้างความสัมพันธ์ขึ้นใหม่ได้อีก จากความสัมพันธ์ที่มีอยู่และตัวปฏิบัติการทั้ง 8 ชนิด ซึ่งการแสดงความสัมพันธ์จะคล้ายกับพีชคณิตในคณิตศาสตร์ทั่วไป

2.2 ไวยากรณ์พีชคณิตสัมพันธ์

ในหัวข้อนี้ จะได้กล่าวถึงไวยากรณ์ของตัวปฏิบัติการพีชคณิตสัมพันธ์โดยละเอียด เพื่อจะได้เป็นพื้นฐานในการทำความเข้าใจในบทต่อๆไป

ไวยากรณ์ของตัวปฏิบัติการพื้นฐาน ได้ถูกกำหนดไว้ในรูปของ Backus Nauer Form (BNF) ดังนี้แสดงไว้ในรูปที่ 2.2

จากหลักการไวยากรณ์ของตัวปฏิบัติการพีชคณิตสัมพันธ์ ในรูปของ BNF นั้น ส่วนของวงเล็บก้ามปูเป็นส่วนที่ไม่สามารถที่จะละทิ้งได้ เพราะเป็นส่วนหนึ่งของไวยากรณ์

rel-def	::=	DEFINE RELATION rel-name [attr-name-commalist]
alias-defn	::=	DEFINE ALIAS rel-name FOR rel-name
expr	::=	select | projection | infix-expr
selection	::=	primitive WHERE selection-pred
primitive	::=	rel-name | (expr)
projection	::=	primitive | primitive [attr-spec-commalist]
attr-spec	::=	attr-mane | rel-name.attr-name
infix-expr	::=	projection infix-op projection
infix-op	::=	UNION | INTERSECT | MINUS | TIMES | JOIN | DIVIDEBY

รูปที่ 2.2 แสดงไวยากรณ์พีชคณิตสัมพันธ์ในรูปของ BNF

2.2.1 ข้อสังเกตเกี่ยวกับไวยากรณ์

2.2.1.1 ถ้า "xyz" คือกลุ่มของชื่อ attribute ดังนั้น "xyz-commalist" จะแทนกลุ่มของชื่อ attribute ที่ประกอบด้วยชื่อของ attribute ภายในกลุ่ม ตั้งแต่หนึ่งชื่อขึ้นไป เขียนเรียงตาม "xyz" และแต่ละชื่อจะถูกแยกออกจากกันด้วยเครื่องหมาย comma (,)

2.2.1.2 การกำหนดชื่อ rel-name และ attr-name จะต้องกำหนดให้แต่ละชื่อเป็นเอกเทศ (identifiers) ไม่ซ้ำกัน

2.2.1.3 selection-pred ใช้แสดงแทนรูปแบบการเปรียบเทียบ ในลักษณะของฟังก์ชันบูลีน บางครั้งอาจจะเป็นการเปรียบเทียบอย่างง่ายๆ ระหว่างค่าของ attribute กับ attribute หรือ ค่าของ attribute กับค่าคงที่

2.2.1.4 ใน projection จะไม่อนุญาตให้ attr-spec เขียนอยู่ในรูปของ expression อย่างเช่น ATTR1+ATTR2 และในทำนองเดียวกัน ในการเปรียบเทียบของฟังก์ชันบูลีนจะไม่อนุญาตให้เขียนอยู่ในรูปของ expression เช่นเดียวกัน

2.3 ตัวปฏิบัติการเบื้องต้น

ตัวปฏิบัติการกลุ่มเบื้องต้นประกอบด้วย union , intersection , difference และ production ซึ่งตัวปฏิบัติการแต่ละตัว จะใช้ความสัมพันธ์สองความสัมพันธ์ในการปฏิบัติการ ความสัมพันธ์ที่ใช้กับตัวปฏิบัติการดังกล่าว (ยกเว้น cartesian product) จะต้อง union-compatible กัน หรือกล่าวอีกนัยหนึ่ง คือ ความสัมพันธ์ทั้งสองจะต้องมี degree (จำนวนแอททริบิว) เท่ากัน เช่น degree = n และ attribute ที่ i (i = 1,2,3, ... ,n) ของแต่ละความสัมพันธ์ ควรจะมีโดเมนเดียวกัน จากกฎของ union-compatible จะทำให้ผลลัพธ์ที่ได้ยังคงอยู่ในรูปของความสัมพันธ์ หากความสัมพันธ์ที่ใช้กับตัวปฏิบัติการดังกล่าว ไม่เป็นไปตามกฎ union-compatible ผลลัพธ์ที่ได้จะไม่เป็นความสัมพันธ์ แต่จะอยู่ในรูปเซต

2.3.1 UNION

ให้ A และ B เป็นความสัมพันธ์ที่ union-compatible มี degree เป็น n ดังนั้นจะกล่าวได้ว่า A UNION B คือความสัมพันธ์ที่มี degree เท่ากับ n และ tuples ทั้งหมดของผลลัพธ์ที่ได้ จะมาจาก tuples ของความสัมพันธ์ A หรือ B

S

S#	SNAME	STATUS	CITY
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

P

P#	PNAME	COLOR	WEIGHT	CITY
P1	Nut	Red	12	London
P2	Bolt	Green	17	Paris
P3	Screw	Blue	17	Rome
P4	Screw	Red	14	London
P5	Cam	Blue	12	paris
P6	Cog	Red	19	London

J

J#	JNAME	CITY
J1	Sorter	Paris
J2	Punch	Rome
J3	Reader	Athens
J4	Console	Athens
J5	Collator	London
J6	Terminal	Oslo
J7	Tape	London

SPJ

S#	P#	J#	QTY
S1	P1	J1	200
S1	P1	J4	700
S2	P3	J1	400
S2	P3	J2	200
S2	P3	J3	200
S2	P3	J4	500
S2	P3	J5	600
S2	P3	J6	400
S2	P3	J7	800
S2	P5	J2	100
S3	P3	J1	200
S3	P4	J2	500
S4	P6	J3	300
S4	P6	J7	300
S5	P2	J2	200
S5	P2	J4	100
S5	P5	J5	500
S5	P5	J7	100
S5	P6	J2	200
S5	P1	J4	100
S5	P3	J4	200
S5	P4	J4	800
S5	P5	J4	400
S5	P6	J4	500

รูปที่ 2.3 แสดงตารางข้อมูลตัวอย่าง

ตัวอย่างที่ 2.1

จากตารางในรูปที่ 2.3 ให้ A เป็นเซตของ tuples ของ supplier ที่อาศัยอยู่ใน LONDON และ ให้ B เป็นเซตของ tuples ของ supplier ที่ขาย part P1 ดังนั้น A UNION B คือ เซตของ tuples ของ supplier ที่อาศัยอยู่ใน LONDON หรือ ขาย part P1 (หรือทั้งสองอย่าง) และ ผลลัพธ์ที่ได้จากตัวอย่างข้างต้นจะมี attribute qualified name (ชื่อของแอททริบิวที่มีการอ้างถึงชื่อความสัมพันธ์ เช่น R.a) เช่นเดียวกับ ความสัมพันธ์แรก คือ ความสัมพันธ์ A

จากตัวอย่างข้างต้น ถ้าเราสลับที่ระหว่าง A และ B ผลลัพธ์ที่ได้ก็ยังคงเหมือนเดิม ดังนั้นจะได้ว่า UNION มีคุณสมบัติการสลับที่ และจากคุณสมบัติปิด ทำให้ UNION มีคุณสมบัติการจัดหมู่ด้วย กล่าวคือ

(X UNION Y) UNION Z

และ

X UNION (Y UNION Z)

จะให้ผลลัพธ์ที่เหมือนกัน ดังนั้นเพื่อความสะดวก จึงไม่นิยมเขียนใส่วงเล็บ เช่น

X UNION Y UNION Z

2.3.2 INTERSECTION

เราจะกล่าวได้ว่า INTERSECTION ของความสัมพันธ์ A และ B ที่ union-compatible และมี degree เป็น n เขียนแทนด้วย A INTERSECT B คือ ความสัมพันธ์ที่มี degree เป็น n และประกอบด้วย tuples ที่เป็นของความสัมพันธ์ A และเป็นของความสัมพันธ์ B

ตัวอย่างที่ 2.2

จากตัวอย่างที่ 2.1 เราจะกล่าวได้ว่า A INTERSECT B คือเซตของ tuples ของ supplier ที่อาศัยใน LONDON และ ขาย part P1

จากตัวอย่างที่ 2.2 ผลลัพธ์ที่ได้ เหมือนกับผลลัพธ์ที่ได้จาก B INTERSECT A ดังนั้น INTERSECTION มีคุณสมบัติการสลับที่ และจากเหตุผลเดียวกันกับตัวปฏิบัติการ UNION จะได้ว่า INTERSECTION มีคุณสมบัติการจัดหมู่ด้วย

2.3.3 DIFFERENCE

การกระทำการ difference ของความสัมพันธ์ A และ B ที่ union-compatible และมี degree เท่ากับ n เขียนแทนด้วย A MINUS B ผลลัพธ์ที่ได้จะอยู่ในรูปของความสัมพันธ์ ที่มี

degree เท่ากับ n และประกอบด้วย tuples ที่เป็น tuples ของความสัมพันธ์ A แต่ไม่เป็น tuples ของความสัมพันธ์ B

ตัวอย่างที่ 2.3

จากตัวอย่างที่ 2.1 ถ้า A MINUS B จะได้ผลลัพธ์เป็นเซตของ tuples ของ supplier ที่อาศัยอยู่ใน LONDON แต่ไม่ขาย part P1

จะเห็นได้ว่า DIFFERENCE ไม่มีคุณสมบัติการสลับที่ ดังนั้น DIFFERENCE จึงไม่มีคุณสมบัติการของการจัดหมู่ไปด้วย

2.3.4 EXTENDED CARTESIAN PRODUCT

ผลลัพธ์ของตัวปฏิบัติการ EXTENDED CARTESIAN PRODUCT ของความสัมพันธ์ A และ ความสัมพันธ์ B ที่มี degree เป็น m และ n ตามลำดับ จะเขียนแทนด้วย A TIMES B ผลลัพธ์ที่ได้จะมี degree เท่ากับ m+n และประกอบไปด้วยเซตของ tuples t ทั้งหมดที่เกิดจากการนำทุก tuple $a = (a_1,a_2, \ldots ,a_m)$ ของความสัมพันธ์ A มาเขียนต่อด้วยทุก tuple $b = (b_1,b_2, \ldots ,b_n)$ ของความสัมพันธ์ B จะได้ tuple $t = (a_1, \ldots,a_m,b_{(m+1)}, \ldots ,b_{(m+n)})$

ผลลัพธ์ที่ได้จากตัวปฏิบัติการ TIMES จะกำหนด qualified name โดย qualified name ของ A และ B เรียงลำดับจากซ้ายไปขวาดังนี้

$$A.a_1, \ldots ,A.a_m,B.b_{(m+1)}, \ldots ,B.b_{(m+n)}$$

ตัวอย่างที่ 2.4

ให้ A เป็นเซตของ supplier และ B เป็นเซตของ part ดังนั้น A TIMES B จะเป็นเซตของคู่ลำดับ supplier/part

จากการที่ลำดับก่อนหลังของ attribute ในความสัมพันธ์ในระบบฐานข้อมูลแบบสัมพันธ์ไม่มีความสำคัญใดๆ ดังนั้นจะได้ว่า

A TIMES B

และ

B TIMES A

ให้ผลลัพธ์ที่เหมือนกัน ซึ่งสรุปได้ว่า TIMES มีคุณสมบัติการสลับที่ และทำนองเดียวกันกับตัวปฏิบัติการ UNION จะได้ว่า TIMES มีคุณสมบัติการจัดหมู่ด้วยซึ่งเขียนได้ดังนี้

(A TIMES B) TIMES C

เท่ากับ

A TIMES (B TIMES C)

2.4 ตัวปฏิบัติการความสัมพันธ์พิเศษ

สำหรับตัวปฏิบัติการที่จัดอยู่ในกลุ่มนี้ได้แก่ selection , projection , join และ divide ซึ่งมีรายละเอียดดังนี้

2.4.1 SELECTION

ให้ R เป็นความสัมพันธ์ ที่มี degree เท่ากับ n และให้ θ แทนการการเปรียบเทียบของตัวปฏิบัติการ (เช่น = , > , < ,<= เป็นต้น) θ-selection ของความสัมพันธ์ R ที่ใช้การเปรียบเทียบ attribute X และ Y เขียนแทนได้ดังนี้

R WHERE R.X θ R.Y

ผลลัพธ์ที่ได้จะเป็นกลุ่มของ tuples t ของความสัมพันธ์ R ที่การเปรียบเทียบ "R.X θ R.Y" เป็นจริง (attribute X และ Y ควรจะอยู่ในโดเมนเดียวกัน และ การเปรียบเทียบ θ จะต้องสมเหตุสมผลกับข้อมูลในโดเมนนั้นด้วย) R.Y อาจจะแทนด้วยค่าคงที่ก็ได้ดังนี้

R WHERE R.X θ c

เมื่อ c เป็นค่าคงที่

θ-selection จะให้ผลลัพธ์ที่เป็นบางส่วนในแนวนอน (horizontal) ของความสัมพันธ์ที่ปฏิบัติการกับตัวปฏิบัติการ selection และ การเปรียบเทียบสามารถที่จะเขียนในรูปของการเปรียบเทียบ ที่เชื่อมต่อด้วยตัวปฏิบัติการของฟังก์ชันบูลีน (Boolean) อันได้แก่ AND , OR และ NOT เช่น

ตัวอย่างที่ 2.5

R WHERE p_1 AND p_2

ซึ่งจะเหมือนกับการกำหนด

(R WHERE p_1) INTERSECT (R WHERE p_2)

ตัวอย่างที่ 2.6

R WHERE p_1 OR p_2

ซึ่งจะเหมือนกับการกำหนด

(R WHERE p_1) UNION (R WHERE p_2)

ตัวอย่างที่ 2.7

R WHERE NOT p

ซึ่งจะเหมือนกับการกำหนด

R MINUS (R WHERE p)

2.4.2 PROJECTION

ให้ R เป็นความสัมพันธ์ที่มี degree เท่ากับ n และประกอบด้วย attribute a_1, ... ,a_n ดังนั้น การปฏิบัติการ projection กับความสัมพันธ์ R ใน attribute a_i, ... ,a_{ik} เมื่อ $k < n$ เขียนแทนได้ด้วย

R [a_i, ... ,a_{ik}]

และผลลัพธ์ที่ได้คือกลุ่มของ tuples (a_i, ... ,a_{ik}) ที่ tuple t เป็นสมาชิกของความสัมพันธ์ R ซึ่งเป็นการกำหนด บางส่วนในแนวตั้ง (vertical) ของความสัมพันธ์ R นั่นเอง เช่น

ตัวอย่างที่ 2.8

S [CITY]

เป็นความสัมพันธ์ที่เกิดจากการเลือกค่าที่ไม่ซ้ำกันใน attribute CITY ของความสัมพันธ์ R และ attribute ที่ได้จะมี qualified name เหมือนกับความสัมพันธ์ R

2.4.3 JOIN

ให้ R_1 และ R_2 เป็นความสัมพันธ์ที่มี degree เท่ากับ m และ n ตามลำดับ และ θ เป็นตัวเปรียบเทียบทางคณิตศาสตร์ เหมือนกับในหัวข้อ 2.4.1 เราจะกล่าวได้ว่า θ-JOIN ของความสัมพันธ์ R_1 ด้วย attribute a กับความสัมพันธ์ R_2 ด้วย attribute b คือกลุ่มของ tuples t ที่เกิดจากการนำ tuples t_1 ในความสัมพันธ์ R_1 มาต่อท้ายด้วย tuples t_2 ในความสัมพันธ์ R_2 ภายใต้เงื่อนไข " $R_1.a \ \theta \ R_2.b$ " ที่เป็นจริง (a และ b ควรจะอยู่ในโดเมนเดียวกัน)

ตัวอย่างที่ 2.9

greater-than join ของความสัมพันธ์ S ด้วย attribute CITY กับความสัมพันธ์ P ด้วย attribute CITY

อย่างไรก็ตาม θ-JOIN มิใช่ตัวปฏิบัติการโดยแท้จริง นั่นคือเราสามารถที่จะเขียน θ-JOIN ให้อยู่ในรูปของ extended cartesian product ของสองความสัมพันธ์ แล้วทำการเลือกเอา tuples ที่เหมาะสมด้วยตัวปฏิบัติการ selection ดังแสดงในตัวอย่างที่ 2.10

ตัวอย่างที่ 2.10

จากตัวอย่างที่ 2.9 เราสามารถที่จะเขียนในรูปของตัวปฏิบัติการที่แท้จริงได้เป็น

(S JOIN P) WHERE S.CITY > P.CITY

ถ้า θ แสดงความเท่ากัน เราจะเรียกการ JOIN นี้ว่า equi-join และการปฏิบัติการของ equi-join ที่ไม่ได้กำหนด attribute ในการกระทำการ จะเรียกการ join นั้นว่า natural-join ซึ่งในครั้งต่อๆไป เมื่อกล่าวถึงการปฏิบัติการ join แล้ว จะหมายถึง natural-join และจะเขียนแทนได้ด้วย JOIN เช่น

S JOIN SP

ซึ่งโดยปรกติ การเขียนในลักษณะข้างต้นจะถูกนิยามก็ต่อเมื่อ unqualified name ของ attribute ของทั้งสองความสัมพันธ์ ที่กระทำการ จะต้องอยู่ภายใต้โดเมนเดียวกัน

จากที่กล่าวมาจะเห็นว่า join มีคุณสมบัติการสลับที่ นั่นคือ

A JOIN B

จะเท่ากับ

B JOIN A

และ JOIN ยังมีคุณสมบัติในการจัดหมู่ด้วยคือ

(A JOIN B) JOIN C

จะให้ผลลัพธ์ เหมือนกับ

A JOIN (B JOIN C)

ดังนั้น จึงมีผู้นิยมเขียนเป็น

A JOIN B JOIN C

2.4.4 DIVISION

ให้ A เป็นความสัมพันธ์ที่มี degree เท่ากับ m+n และ B เป็นความสัมพันธ์ที่มี degree เท่ากับ n แล้วจะกล่าวได้ว่า ผลลัพธ์ของการ division ความสัมพันธ์ A ด้วยความสัมพันธ์ B เขียนแทนด้วย

A DIVIDEBY B

คือความสัมพันธ์ ที่มี degree เท่ากับ m โดยที่ความสัมพันธ์ A มีข้อมูลของ tuples ใน attributes ที่ m+1 ถึง m+n เหมือนกับข้อมูล ในทุก tuples ของความสัมพันธ์ B ในขณะที่ข้อมูลของ tuples เหล่านั้นใน attributes ที่ 1 ถึง m ของความสัมพันธ์ A มีค่าเพียงค่าเดียว

จากนิยามข้างต้น ข้อมูลใน attribute ที่ m+i ของความสัมพันธ์ A และข้อมูลใน attribute ที่ i ของความสัมพันธ์ B (เมื่อ i = 1, 2, ... ,n) ควรจะอยู่ในโดเมนเดียวกัน และ unqualified name ของผลที่ได้จะเหมือนกับ m attributes แรกของความสัมพันธ์ A

อย่างไรก็ตาม DIVIDEBY ไม่ได้เป็นตัวปฏิบัติการโดยแท้จริง ซึ่งสามารถที่จะเขียนให้อยู่ในรูปของตัวปฏิบัติการที่แท้จริงได้ เช่น

A DIVIDEBY B = A [X] MINUS ((A [X] TIMES B) MINUS A) [X]

เมื่อ X แทน attribute ของความสัมพันธ์ Aที่ไม่ร่วมกันกับ attribute ของความสัมพันธ์ B

ตัวอย่างที่ 2.11

จากตารางข้อมูลตัวอย่างนั้น ถ้าต้องการการหาว่าผู้ขายสินค้าหมายเลขใด ขายสินค้าทุกชนิด จะเขียนเป็นคำถามในภาษาพีชคณิตสัมพันธ์ได้ดังนี้

SP [S#, P#] DIVIDEBY P[P#]

ซึ่งผลลัพธ์ที่ได้คือ S5

2.5 ตัวอย่างการใช้งานพีชคณิตสัมพันธ์

สำหรับตัวอย่างการใช้งานในแต่ละตัวปฏิบัติการนั้น ได้กล่าวไว้ใน หัวข้อ 2.4 แล้ว ในหัวข้อนี้จะเป็นการยกตัวอย่างการใช้งานพีชคณิตสัมพันธ์ ซึ่งในการใช้งานจริงนั้น อาจจะมีหลายตัวปฏิบัติการในหนึ่งคำถาม โดยไม่อาจกำหนดแน่นอนได้ว่าต้องใช้ตัวปฏิบัติการเท่าใด และในหนึ่งคำถามนั้นอาจเขียนเป็นภาษาพีชคณิตสัมพันธ์ ได้มากกว่าหนึ่งแบบ ก็ได้

ดังนั้นในหัวข้อนี้จะได้ยกตัวอย่างเป็นแนวทางในการใช้งาน โดยใช้ข้อมูลจากตารางข้อมูลตัวอย่าง ดังต่อไปนี้

ตัวอย่างที่ 2.12

จงหาชื่อผู้ขายสินค้า (supplier) ที่ขายสินค้า (part) P2

((S JOIN SP) WHERE P# = 'P2') [SNAME]

ตัวอย่างที่ 2.13

จงหาชื่อผู้ขายสินค้า ที่ขายสินค้าสีแดงอย่างน้อยหนึ่งชิ้น

(((P WHERE COLOR = 'Red') [P#] JOIN SP) [S#] JOIN S) [SNAME]

ตัวอย่างที่ 2.14

จงหาชื่อผู้ขายสินค้าที่ขายสินค้าทุกชนิด

((SP [S#, P#] DIVIDEBY P [P#]) JOIN S) [SNAME]

ตัวอย่างที่ 2.15

จงหาหมายเลขของผู้ขายสินค้า ซึ่งสินค้าทั้งหมดที่ขายไม่น้อยกว่าสินค้าที่ขายโดยผู้ขายหมายเลข S2

SP [S#, P#] DIVIDEBY (SP WHERE S# = 'S2') [P#]

ตัวอย่างที่ 2.16

จงหาชื่อของผู้ขายสินค้า ที่ไม่ได้ขายสินค้าหมายเลข P2

((S [S#] MINUS (SP WHERE P# = 'P2') [S#]) JOIN S) [SNAME]

บทที่ 3
หลักการของระบบฐานข้อมูลแบบกระจาย

3.1 ความรู้เบื้องต้น

ในการศึกษาการออกแบบระบบฐานข้อมูลแบบกระจายนั้น จำเป็นต้องมีความรู้พื้นฐานทั่วไปบางประการ ซึ่งในหัวข้อนี้จะได้กล่าวถึงบางส่วนเท่านั้น โดยเนื้อหาส่วนใหญ่ของหัวข้อ 3.1 ถึง 3.4 เป็นการสรุปมาจาก[5] ส่วนหัวข้อ 3.5 เนื้อหาส่วนใหญ่เป็นการสรุปมาจาก [4],[5],[6],[7] และ[8] ซึ่งจะได้กล่าวถึงเป็นหัวข้อดังต่อไปนี้

3.1.1 ความหมายของฐานข้อมูลแบบกระจาย

เป็นที่ทราบกันดีแล้วว่า ในช่วงทศวรรษ 1970 คอมพิวเตอร์ได้ถูกนำมาใช้ประโยชน์ในด้านต่างๆอย่างมากมาย ซึ่งรวมทั้งการนำมาใช้งานเกี่ยวกับระบบฐานข้อมูลด้วย และจากการพัฒนาการของระบบฐานข้อมูล ที่เป็นไปอย่างรวดเร็ว ทำให้ในปัจจุบันมีโปรแกรมใช้งานขนาดต่างๆมากมาย ในขณะเดียวกันโครงข่ายคอมพิวเตอร์ก็ได้รับการพัฒนาอย่างต่อเนื่อง มีการเชื่อมต่อของคอมพิวเตอร์ต่างเครื่องกัน ทำให้สามารถจะแลกเปลี่ยนข้อมูล และทรัพยากรซึ่งกันและกันระหว่างคอมพิวเตอร์ต่างเครื่องกันได้

จากการพัฒนาของวิทยาการทั้งสองสาขา และการผสมผสานกันทำให้เกิดเป็นความรู้อีกสาขาหนึ่งซึ่งเรียกว่า "ระบบฐานข้อมูลแบบกระจาย" (Distributed Database Systems) หากจะกล่าวโดยสรุปแล้ว ระบบฐานข้อมูลแบบกระจายก็คือ "การเก็บรวบรวมข้อมูลที่มีความสัมพันธ์ ในทางตรรก เข้าไว้เป็นระบบเดียวกัน แต่จะกระจายข้อมูลไปเก็บตามสถานที่ (site) ต่างๆ โดยผ่านระบบโครงข่ายคอมพิวเตอร์" [5]

จากคำจำกัดความข้างต้น จะสรุปเป็นหัวข้อได้ 3 ประการคือ

ประการแรก <u>การกระจาย</u> (Distribution) คือ ข้อมูลจะไม่ถูกเก็บไว้ในสถานที่เดียวกัน (ตัวประมวลผลเดียวกัน) ดังนั้น เราสามารถที่จะแยกระบบฐานข้อมูลแบบรวมออกจากระบบฐานข้อมูลแบบกระจายได้

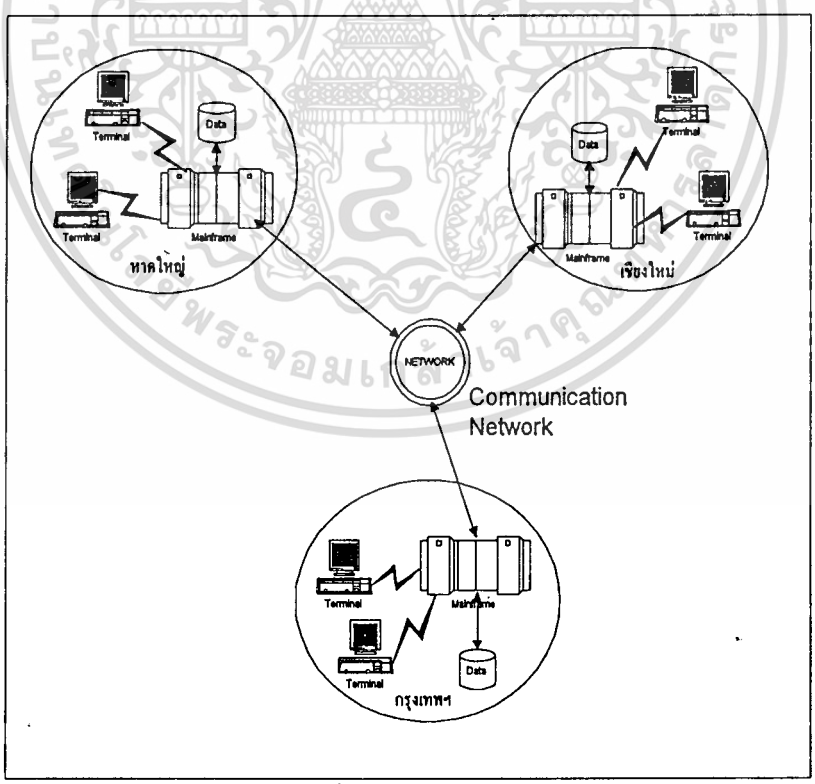
ประการที่สอง <u>ความเกี่ยวพันทางตรรก</u> (Logical Correlation) จากเหตุผลที่ข้อมูลในแต่ละสถานที่ ที่ประกอบกันเป็นระบบฐานข้อมูลแบบกระจาย ควรจะมีความสัมพันธ์กัน ดังนั้นเราสามารถที่จะชี้ให้เห็นความแตกต่าง ของระบบฐานข้อมูลแบบกระจาย กับกลุ่มของระบบฐานข้อมูลแบบรวมที่อยู่ในต่างสถานที่ได้

ประการที่สาม <u>โปรแกรมประยุกต์</u> (Application Programs) คือ ระบบฐานข้อมูลจะต้องมีทั้งโปรแกรมประยุกต์แบบรวม (Global Application Programs) และโปรแกรมประยุกต์แบบท้องถิ่น (Local application Programs)

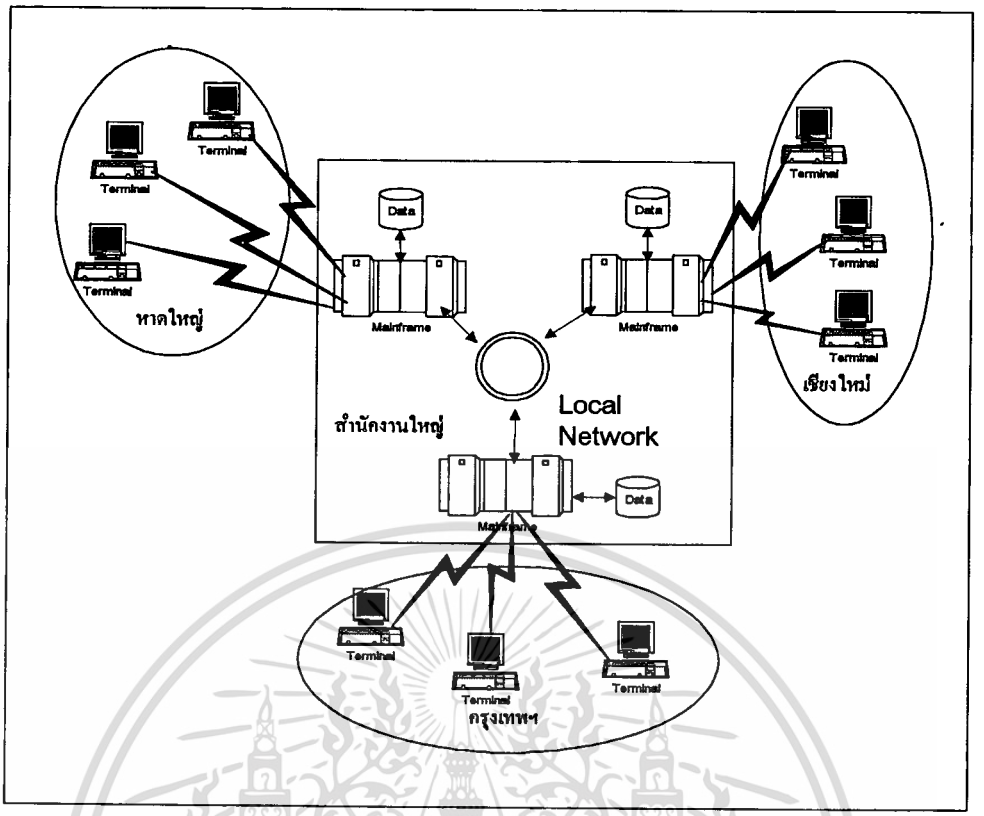
จะขอยกตัวอย่างเพื่อให้เห็นความแตกต่างชัดเจนมากยิ่งขึ้น ระหว่างระบบฐานข้อมูลแบบกระจาย กับระบบฐานข้อมูลแบบรวม

<u>ตัวอย่างที่ 3.1</u>

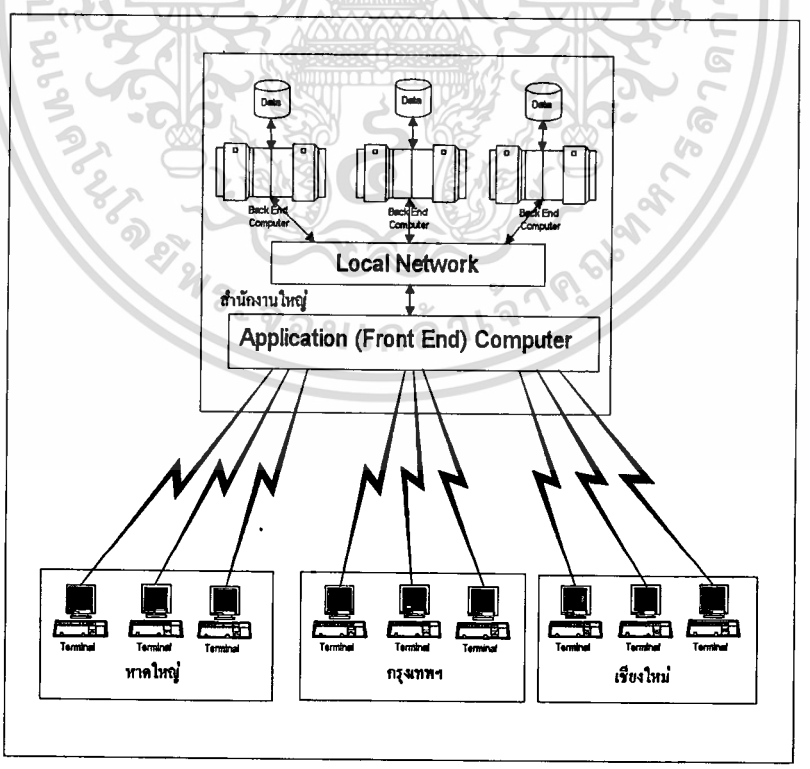
จากรูปที่ 3.1 แสดงการจำลอง ธนาคาร 3 สาขาที่อยู่ต่างสถานที่กัน แต่ละสาขาจะทำการควบคุมเทอร์มินัล และข้อมูลของตัวเอง คอมพิวเตอร์และ ข้อมูลในแต่ละสาขาประกอบกันเรียกว่า "หนึ่งสถานที่" (site) ของฐานข้อมูลแบบกระจาย คอมพิวเตอร์ในแต่ละสถานที่จะต่อถึงกันด้วยโครงข่ายสื่อสาร โดยปรกติแล้ว โปรแกรมประยุกต์จะเรียกใช้ข้อมูลของสาขาที่โปรแกรมทำงานอยู่เท่านั้น ซึ่งเรียกโปรแกรมประยุกต์ชนิดนี้ว่า "โปรแกรมประยุกต์แบบท้องถิ่น" (Local Applications) ส่วนโปรแกรมที่มีการเรียกใช้ข้อมูลที่เก็บต่างสถานที่นั้นจะเรียกว่า "โปรแกรมประยุกต์แบบรวม" (Global Applications) หรือเรียกว่า "โปรแกรมประยุกต์แบบกระจาย" (Distributed Applications)



รูปที่ 3.1 แสดงระบบฐานข้อมูลที่กระจายตามลักษณะภูมิศาสตร์



รูปที่ 3.2 แสดงระบบฐานข้อมูลแบบกระจายที่ใช้โครงข่ายแบบท้องถิ่น



รูปที่ 3.3 แสดงระบบประมวลผลแบบหลายตัวประมวลผล

ตัวอย่างที่ 3.2

จากกรณีของธนาคารในตัวอย่างที่ 3.1 ถ้าโครงสร้างของระบบฐานข้อมูลถูกเปลี่ยนแปลงให้เป็นดังรูปที่ 3.2 คือ ย้ายคอมพิวเตอร์ทั้งหมดมาอยู่รวมกันที่ส่วนกลาง และต่อเทอร์มินัลผ่านสายโทรศัพท์ คอมพิวเตอร์แต่ละเครื่องและฐานข้อมูลของเครื่องนั้นๆ รวมกันเรียกว่าหนึ่งสถานที่ (site) เหมือนเดิม จะเห็นว่า แม้ลักษณะทางกายภาพจะถูกเปลี่ยนแปลงไป แต่สถาปัตยกรรมของระบบ และคุณสมบัติต่างๆทางตรรก ยังคงเหมือนเดิม การทำงานของโปรแกรมประยุกต์แบบท้องถิ่น ก็ยังคงประมวลผลที่เครื่องคอมพิวเตอร์สถานที่เดิม รวมทั้งเข้าถึงข้อมูลของสถานที่เดิมอีกด้วย ซึ่งสรุปว่า การแบ่งความเป็นท้องถิ่น ไม่ได้ขึ้นอยู่กับลักษณะทางกายภาพ แต่จะขึ้นอยู่กับสถานที่ (site) ประมวลผลและข้อมูลที่ใช้

ตัวอย่างที่ 3.3

จากรูปที่ 3.3 เป็นการแสดงโครงสร้างของธนาคาร ทั้ง 3 สาขา ซึ่งโครงสร้างจะเปลี่ยนไปจากตัวอย่างที่ 3.1 และ 3.2 คือ ข้อมูลของธนาคารทั้ง 3 จะเก็บไว้ที่เครื่องคอมพิวเตอร์ส่วนหลัง (backend) 3 เครื่อง ซึ่งจะทำหน้าที่จัดการฐานข้อมูล ส่วนโปรแกรมประยุกต์จะประมวลผลบนคอมพิวเตอร์เครื่องอื่น ซึ่งจะเรียกใช้ข้อมูลผ่านเครื่องคอมพิวเตอร์ส่วนหลัง จะเห็นว่าในตัวอย่างนี้เป็นเพียงระบบฐานข้อมูลแบบรวม ที่ประมวลผลบนตัวประมวลผลหลายตัวเท่านั้น เนื่องจากโปรแกรมประยุกต์ใช้งาน ของธนาคารทั้ง 3 สาขาประมวลผลบนคอมพิวเตอร์เครื่องเดียวกัน แม้ว่าจะแยกเก็บข้อมูลไว้ต่างที่กันก็ตาม ก็ไม่ได้จัดว่าเป็นระบบฐานข้อมูลแบบกระจาย

จากตัวอย่างที่กล่าวมาทั้งหมด คงจะเพียงพอสำหรับการทำความเข้าใจ ความหมาย และแยกความแตกต่างระหว่าง ระบบฐานข้อมูลแบบรวม และระบบฐานข้อมูลแบบกระจายได้

3.1.2 ระบบจัดการฐานข้อมูลแบบกระจาย

ระบบจัดการฐานข้อมูลแบบกระจายนั้น เป็นสิ่งจำเป็นสำหรับการสร้างระบบฐานข้อมูลแบบกระจาย และบำรุงรักษาข้อมูลในระบบฐานข้อมูลแบบกระจาย ซึ่งโดยปรกติแล้วผลิตภัณฑ์ระบบจัดการฐานข้อมูลแบบกระจายที่จำหน่ายในท้องตลาดนั้น จะมีความแตกต่างจากระบบจัดการฐานข้อมูลแบบกระจาย ที่กำลังพัฒนาในห้องทดลองบ้างพอสมควร แต่อย่างไรก็ตามโครงสร้างของระบบจัดการฐานข้อมูลแบบกระจาย จะมีลักษณะคล้ายกับโครงสร้างของระบบจัดการฐานข้อมูลแบบรวมอยู่บ้างดังรูปที่ 3.4 ซึ่งพอจะสรุปส่วนประกอบได้ดังนี้ คือจะประกอบด้วยซอฟท์แวร์ 4 ส่วนด้วยกันได้แก่

1. ส่วนจัดการฐานข้อมูล (DB)

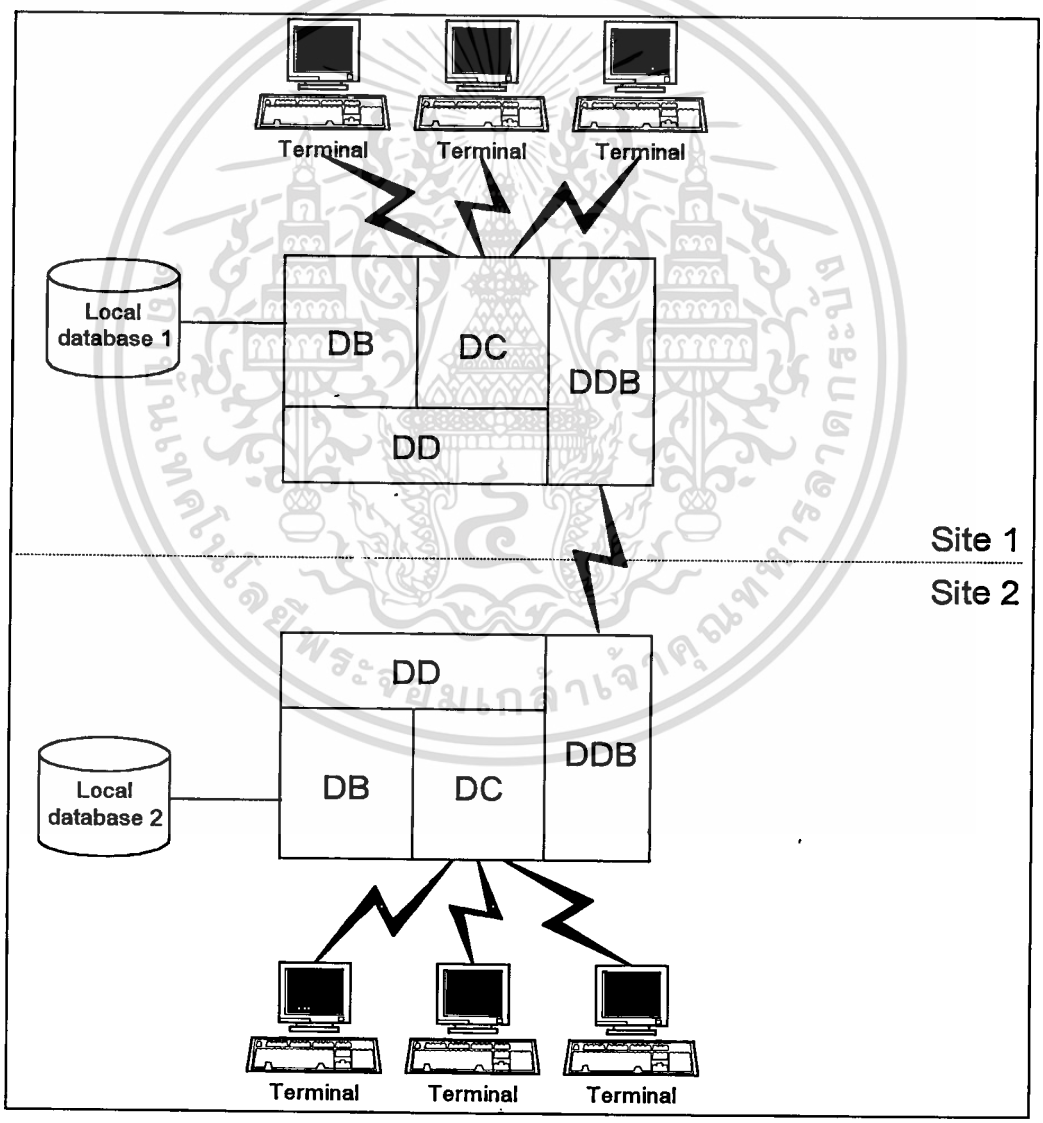
2. ส่วนสื่อสารข้อมูล (DC)

3. ส่วนพจนานุกรมฐานข้อมูล (DD)

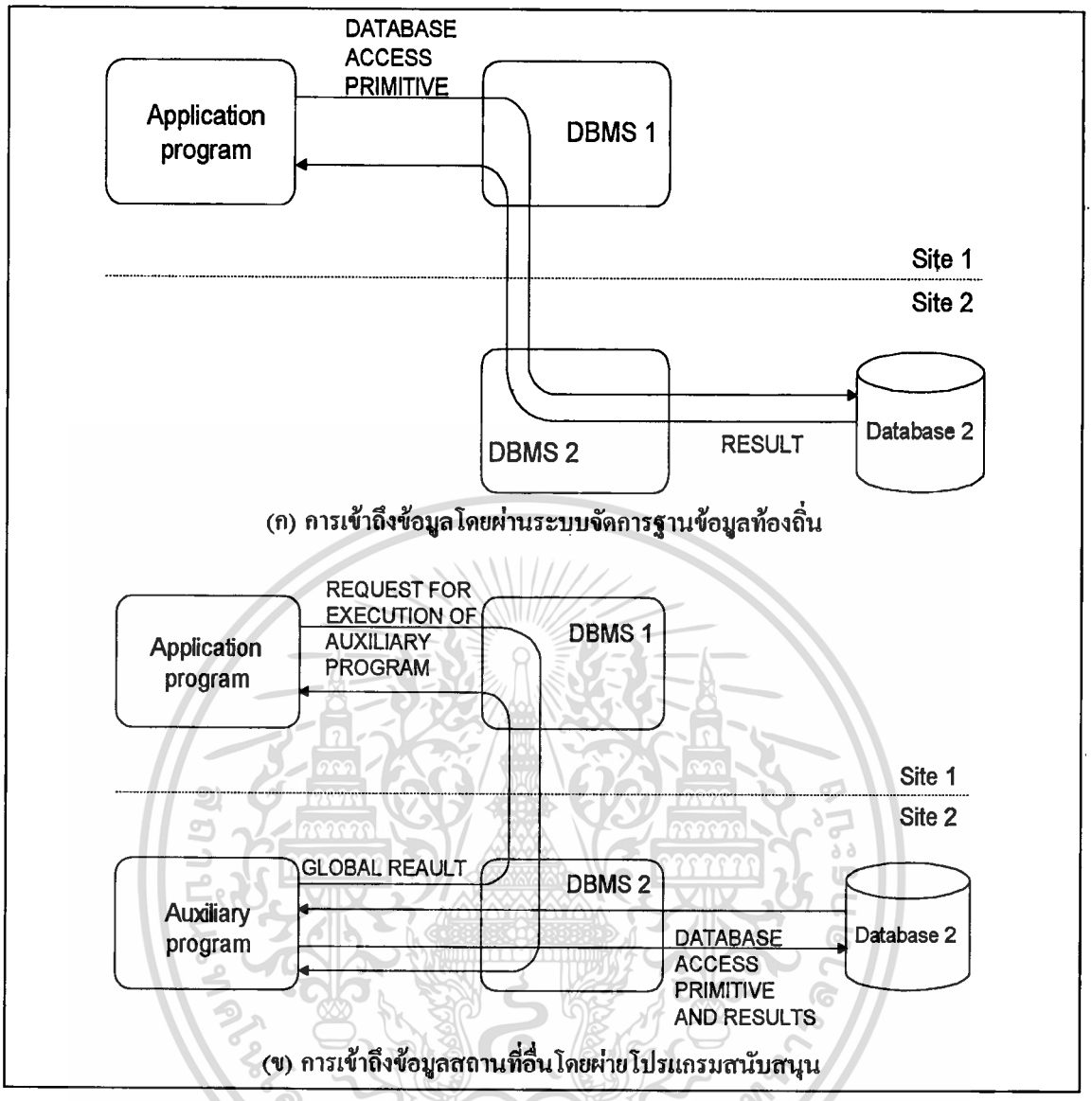
4. ส่วนเชื่อมต่อฐานข้อมูลแบบกระจาย (DDB)

จากภาพที่ 3.4 เราจะเรียกส่วนประกอบทั้ง 4 ส่วนว่า ระบบจัดการฐานข้อมูลแบบกระจาย ซึ่งส่วนประกอบที่จัดว่ามีความสำคัญได้แก่ ส่วนเชื่อมต่อฐานข้อมูลแบบกระจาย ซึ่งจะมีหน้าที่ในการติดต่อกับฐานข้อมูลสถานที่อื่น

ในการเข้าถึงข้อมูลที่อยู่ต่างสถานที่(remote site)นั้น สามารถกระทำได้ 2 วิธีดังแสดงในรูปที่ 3.5 คือ



รูปที่ 3.4 แสดงส่วนประกอบและการต่อเชื่อมของระบบฐานข้อมูลแบบกระจาย



รูปที่ 3.5 แสดงวิธีการเข้าถึงข้อมูลในระบบฐานข้อมูลแบบกระจาย

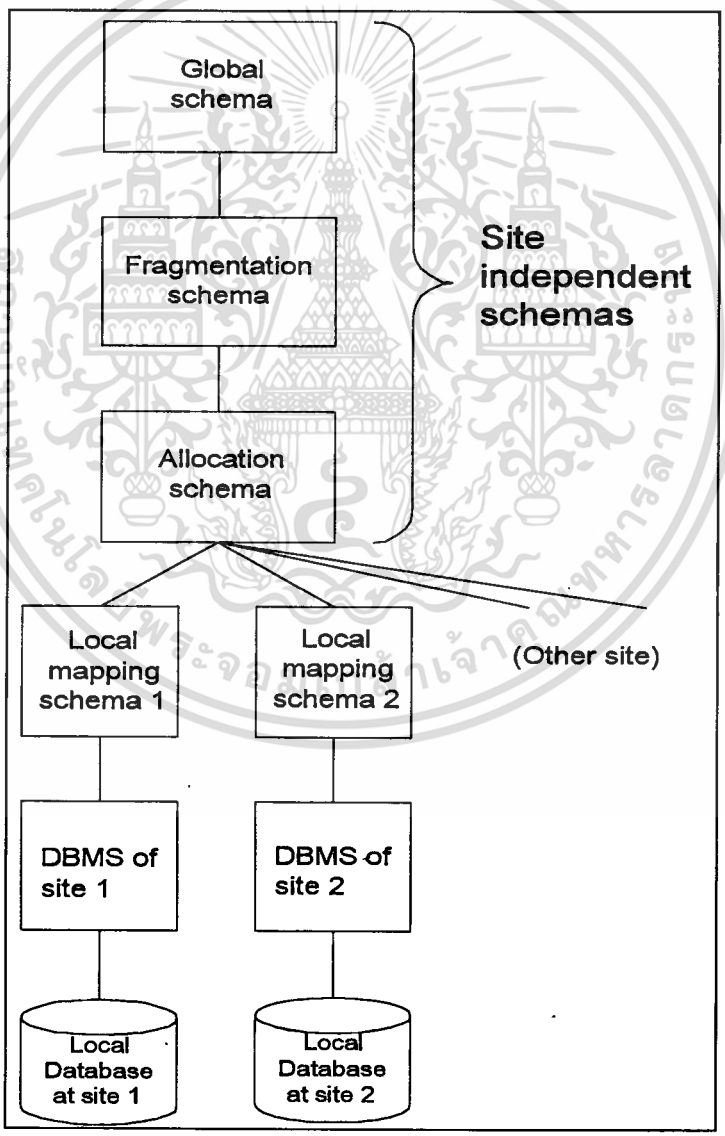
วิธีแรก ดังรูปที่ 3.5 (ก) โปรแกรมประยุกต์จะแสดงความต้องการข้อมูลกับระบบจัดการฐานข้อมูลแบบกระจายท้องถิ่น โดยอ้างถึงสถานที่ของฐานข้อมูล ระบบจัดการฐานข้อมูลท้องถิ่นจะค้นหาเส้นทางการเข้าถึงข้อมูล จากนั้นจะส่งความต้องการให้กับระบบจัดการฐานข้อมูลแบบกระจายของฐานข้อมูลนั้นๆ พร้อมทั้งรอรับข้อมูลที่ต้องการ ซึ่งวิธีนี้การติดต่อจะกระทำระหว่างระบบจัดการฐานข้อมูลกับระบบจัดการฐานข้อมูล

วิธีที่สอง แสดงในรูปที่ 3.5 (ข) เมื่อโปรแกรมประยุกต์แสดงความต้องการข้อมูลต่อระบบจัดการฐานข้อมูลแบบกระจายท้องถิ่นแล้ว ระบบจัดการฐานข้อมูลแบบกระจายท้องถิ่น ก็จะค้นหาสถานที่เก็บข้อมูลที่ต้องการ จากนั้นก็จะส่งความต้องการให้กับโปรแกรมสนับสนุน ที่ทำงานในสถานที่เก็บข้อมูลนั้นๆ โปรแกรมสนับสนุนก็จะทำการประมวลผล และจัดส่งข้อมูลให้กับระบบ

จัดการฐานข้อมูลที่ส่งความต้องการมา ซึ่งจะเห็นว่าวิธีนี้เป็นการติดต่อระหว่างระบบจัดการฐานข้อมูลท้องถิ่นกับโปรแกรมสนันสนุน

3.2 ระดับการส่งผ่านการกระจายของฐานข้อมูล (Levels of Distribution Transparency)

การส่งผ่านการกระจายของฐานข้อมูล กล่าวโดยสรุปหมายถึง ลักษณะการเข้าถึงข้อมูลในระบบฐานข้อมูลแบบกระจายของโปรแกรมใช้งานประยุกต์ ว่าจะมีความเป็นอิสระ ไม่ขึ้นกับการกระจายของข้อมูลมากน้อยเพียงไร หรือกล่าวอีกนัยหนึ่งจะได้ว่า มีลักษณะการเข้าถึงข้อมูลคล้ายกับการเข้าถึงข้อมูลในระบบฐานข้อมูลแบบรวมมากน้อยเท่าใด ซึ่งระดับการส่งผ่านสามารถที่จะจัดแบ่งตามลักษณะแบบแผนการอ้างถึงข้อมูล ได้เป็น 3 ระดับ คือ



รูปที่ 3.6 แสดงสถาปัตยกรรมการอ้างถึงข้อมูลของฐานข้อมูลแบบกระจาย

3.2.1 แบบแผนโดยรวม (Global schema)

จากรูปที่ 3.6 เป็นการแสดงสถาปัตยกรรมการอ้างถึงข้อมูล ของระบบฐานข้อมูลแบบกระจาย ส่วนที่อยู่บนสุดคือแบบแผนการอ้างถึงโดยรวม โปรแกรมประยุกต์ที่มีการอ้างถึงข้อมูลโดยใช้แบบแผนนี้ จะอ้างถึงข้อมูลที่สถานที่ต่างๆ เสมือนกับว่าข้อมูลทั้งหมดไม่มีการกระจายเลย ซึ่งจะเหมือนกับการอ้างถึงข้อมูลในฐานข้อมูลแบบรวม และความสัมพันธ์ที่ถูกอ้างถึงนี้ จะเรียกว่า ความสัมพันธ์โดยรวม (Global relations)

3.2.2 แบบแผนส่วนย่อย (Fragmentation schema)

การอ้างถึงโดยแบบแผนส่วนย่อยนี้ เป็นการเข้าถึงข้อมูลโดย อ้างถึงบางส่วนของความสัมพันธ์โดยรวม ที่ถูกแบ่งออกเป็นส่วนๆไม่ซ้ำกัน โดยจะเรียกแต่ละส่วนว่า ส่วนย่อย (fragment) ซึ่งวิธีการแบ่งความสัมพันธ์โดยรวมออกเป็นส่วนย่อยนี้ สามารถแบ่งได้หลายวิธี เช่น แบ่งตามแนวนอน (Horizontal Fragmentation) คือ แยก ทัปเปิล (tuples) ออกเป็นกลุ่มๆ หรือแบ่งตามแนวตั้ง (Vertical Fragmentation) คือ แยกแอททริบิวออกเป็นกลุ่มๆ ส่วนย่อยที่แบ่งออกมานี้ สามารถที่จะนำกลับมารวมเข้าเป็นความสัมพันธ์โดยรวมได้ดังเดิม ด้วยการทำ UNION และ JOIN ตามลำดับ อย่างไรก็ตามการแบ่งส่วนย่อยนี้ สามารถที่จะกระทำทั้งสองวิธีผสมกัน (Mixed Fragmentation) ได้ ทั้งนี้จะต้องสามารถนำกลับมารวมเป็นความสัมพันธ์โดยรวมได้ดังเดิม

3.2.3 แบบแผนการอ้างถึงตำแหน่ง (Allocation schema)

ส่วนย่อยที่แยกมาจากความสัมพันธ์โดยรวมนั้น จะถูกกำหนดสถานที่ที่จะเก็บข้อมูลส่วนนั้นๆ (Local mapping schema) ซึ่งในแต่ละส่วนย่อยนี้ อาจจะมีสถานที่เก็บข้อมูลมากกว่าหนึ่งแห่งได้ ทั้งนี้ขึ้นอยู่กับการตัดสินใจของผู้ออกแบบระบบฐานข้อมูล ว่าจะยอมให้มีการซ้ำซ้อนของข้อมูลมากน้อยเพียงใด และสอดคล้องกับการใช้งานหรือไม่

3.3 การออกแบบฐานข้อมูลแบบกระจาย (Distributed Database Design)

ในการออกแบบฐานข้อมูลแบบกระจาย โดยทั่วไปนับว่าเป็นขั้นตอนที่มีความสำคัญขั้นตอนหนึ่ง ของการสร้างระบบฐานข้อมูลแบบกระจาย และมีความซับซ้อนพอสมควร เนื่องจากการออกแบบฐานข้อมูลแบบกระจาย จะต้องคำนึงถึงปัจจัยหลายด้าน เช่น ปัญหาและความเป็นไปได้ทางเทคนิค , ข้อจำกัดของการจัดการองค์กร เป็นต้น

ในหัวข้อนี้จะได้กล่าวถึงการออกแบบฐานข้อมูลที่ใช้กับระบบ DDQ/1 เป็นส่วนใหญ่ ดังนั้นขั้นตอนและรายละเอียด ของการออกแบบฐานข้อมูลที่จะได้กล่าวถึงนี้ จะมีความแตกต่างจากการออกแบบฐานข้อมูลแบบกระจายโดยทั่วๆไปอยู่พอสมควร แต่อย่างไรก็ตาม ขั้นตอนในการออกแบบโดยทั่วไป ก็จะกล่าวถึงโดยสังเขป เพื่อเป็นแนวทางในการเปรียบเทียบ และเพิ่มความเข้าใจยิ่งขึ้น การออกแบบฐานข้อมูลแบบกระจายโดยทั่วไป จะมีอยู่ 2 วิธีด้วยกัน คือ วิธีการออกแบบจากบนลงล่าง และ การออกแบบจากล่างขึ้นบน ทั้งสองวิธีมีลำดับในการออกแบบที่แตกต่างกัน รวมทั้งมีความเหมาะสมที่จะใช้งานต่างกันด้วย ซึ่งสรุปได้ดังนี้

3.3.1 การออกแบบฐานข้อมูลจากบนลงล่าง (Top-Down design)

ในการออกแบบจากบนลงล่างนี้ ขั้นตอนเริ่มจากการวิเคราะห์ คือศึกษาข้อมูลและความต้องการของผู้ใช้งาน ผลของการวิเคราะห์จะนำมาใช้ในการออกแบบ Global Conceptual Schema เมื่อเสร็จขั้นตอนนี้จะได้ความสัมพันธ์โดยรวม (Global Relations) ของระบบ เพื่อใช้ในขั้นตอนแยกเป็นส่วนย่อย (Fragmentation) และกำหนดสถานที่เก็บส่วนย่อยต่อไป ต่อจากนั้นจะเป็นขั้นตอนการออกแบบลักษณะการเก็บข้อมูลทางกายภาพของแต่ละสถานที่ หลังจากขั้นตอนนี้แล้วก็จะเป็นขั้นตอนของการพัฒนาระบบและสังเกตปรับแต่งระบบให้ทำงานอย่างมีประสิทธิภาพ

จากขั้นตอนที่กล่าวมาข้างต้นนั้น จะเห็นว่าเป็นการออกแบบระบบจากระบบฐานข้อมูลกระจายโดยรวม ไปสู่ฐานข้อมูลในแต่ละสถานที่ ซึ่งลักษณะการออกแบบชนิดนี้เหมาะสมที่จะใช้ออกแบบระบบฐานข้อมูลแบบกระจายใหม่ทั้งระบบ

3.3.2 การออกแบบฐานข้อมูลจากล่างขึ้นบน (Buttom-Up Design)

การออกแบบจากล่างขึ้นบน เป็นลักษณะการออกแบบฐานข้อมูลแบบกระจาย จากฐานข้อมูลที่มีอยู่แล้วหลายสถานที่ ซึ่งขั้นตอนส่วนใหญ่ จะเป็นการรวม Conceptual Schema ของแต่ละสถานที่ให้เป็น Global Conceptual Schema ของระบบ จากนั้นจึงเป็นการพัฒนาระบบ และปรับแต่งการทำงานของระบบต่อไป ซึ่งการออกแบบลักษณะนี้เหมาะกับการสร้างระบบฐานข้อมูลแบบกระจาย จากฐานข้อมูลแบบรวมที่มีอยู่แล้ว

3.3.3 การออกแบบฐานข้อมูลในระบบ DDQ/1

ในการออกแบบฐานข้อมูลแบบกระจายที่ใช้กับระบบจัดการฐานข้อมูลแบบกระจาย DDQ/1 นี้ สามารถที่จะออกแบบได้ทั้ง 2 วิธี คือวิธีการออกแบบฐานข้อมูลจากบนลงล่าง และการออกแบบฐานข้อมูลจากล่างขึ้นบน ทั้งนี้ขึ้นอยู่กับว่า เป็นการออกแบบฐานข้อมูลจากเริ่มต้นหรือไม่

หรือว่าเป็นการออกแบบจากฐานข้อมูลแบบรวมที่มีอยู่แล้ว เนื่องจากในการพัฒนาระบบจัดการฐานข้อมูลแบบกระจาย DDQ/1 มีจุดประสงค์ที่จะสร้างฐานข้อมูลแบบกระจายจากฐานข้อมูลแบบรวมที่มีอยู่แล้ว ดังนั้น จะขอกล่าวเฉพาะวิธีการออกแบบฐานข้อมูลจากล่างขึ้นบนเท่านั้น อย่างไรก็ตามวิธีการออกแบบฐานข้อมูลจากบนลงล่างก็ใช้ได้ดีเช่นกัน

ในการออกแบบฐานข้อมูลแบบกระจาย ที่ใช้กับระบบประมวลผลคำถามแบบกระจาย DDQ/1 นี้ ผู้ดูแลระบบฐานข้อมูลในแต่ละสถานที่ (Local Database Administator : LDBA) จะเป็นผู้ออกแบบระบบฐานข้อมูลท้องถิ่นในสถานที่นั้น จากนั้นผู้ดูแลระบบฐานข้อมูลแต่ละสถานที่ จะต้องนำ Conceptual Schema ที่ได้จากการออกแบบส่งให้กับผู้ดูแลระบบฐานข้อมูลโดยรวม (Global Database Administator : GDBA)

เมื่อผู้ดูแลระบบฐานข้อมูลโดยรวมได้รับ Conceptual Schema จากผู้ดูแลระบบฐานข้อมูลแต่ละสถานที่แล้ว ก็จะทำการรวม Conceptual Schema ทั้งหมดเป็น Global Conceptual Schema ซึ่งถ้าแต่ละ Conceptual Schema มีการใช้ชื่อความสัมพันธ์ซ้ำกันก็จะมีการกำหนดชื่อใหม่ และ ชื่อแอททริบิวเดียวกันจะต้องมีชนิดและขนาดของข้อมูลเหมือนกัน ทั้งนี้เพื่อจะได้อ้างถึงสิ่งเดียวกัน (ความสัมพันธ์ หรือ แอททริบิว) ด้วยชื่อเดียวกัน

หลังจากได้ Global Conceptual Schema แล้ว ผู้ดูแลระบบฐานข้อมูลแต่ละสถานที่ก็จะนำไปสร้างตารางเก็บข้อมูลต่อไป โดยจะสร้างตารางเฉพาะในส่วนที่ตัวเองได้ออกแบบ Conceptual Schema ไว้เท่านั้น หลังจากนั้นขั้นตอนต่างๆ จะเป็นขั้นตอนในการเริ่มใช้งานระบบประมวลผลคำถามแบบกระจาย DDQ/1 ซึ่งจะได้กล่าวในหัวข้อต่อไป

3.4 การแปลงคำถาม (Translation of Queries)

ในการออกแบบระบบจัดการฐานข้อมูลแบบกระจายโดยทั่วไป ความสัมพันธ์โดยรวม (Global Relation) หนึ่งๆ อาจจะถูกแบ่งเป็นส่วนย่อยหลายส่วนเก็บไว้ในต่างสถานที่กัน เมื่อมีคำถามส่งมาจากผู้ใช้งาน และคำถามมีการอ้างถึงความสัมพันธ์โดยรวมนั้น คำถามเหล่านี้จะถูกแปลงเป็นคำถามย่อยหลายคำถามสอดคล้องกับสถานที่ ที่เก็บส่วนย่อยที่ถูกอ้างถึงโดยคำถาม แต่ในการพัฒนาครั้งนี้ เป็นการสร้างระบบจัดการฐานข้อมูลแบบกระจายจากระบบจัดการฐานข้อมูลแบบรวมที่มีอยู่แล้ว ดังนั้นความสัมพันธ์ (ตารางข้อมูล) ในฐานข้อมูลหนึ่งๆ จึงไม่มีการแบ่งเป็นส่วนย่อย ทำให้คำถามที่ถูกส่งมาจากผู้ใช้งาน ไม่มีการแบ่งเป็นคำถามย่อยเช่นเดียวกัน แต่อย่างไรก็ตาม คำถามก็จะถูกปรับปรุงให้การประมวลผลคำถามเป็นไปด้วยความรวดเร็ว และมีการรับ-ส่งข้อมูลระหว่างสถานที่น้อยที่สุด ดังนั้นใน 2 หัวข้อต่อไปจะได้กล่าวถึง การเปลี่ยนรูปที่เท่ากัน

สำหรับพีชคณิตสัมพันธ์ และตัวปฏิบัติการเชิงต้นไม้ ซึ่งทั้งสองหัวข้อนี้จะเป็นพื้นฐาน สำหรับทำความเข้าใจในหัวข้อการเข้าถึงข้อมูลที่ดีที่สุดต่อไป

3.4.1 การเปลี่ยนรูปที่เท่ากันสำหรับพีชคณิตสัมพันธ์ (Equivalence Transformations for the Relational Algebra)

ก่อนที่จะกล่าวถึงการเปลี่ยนรูป จะขอทำความเข้าใจกับคำว่า "เท่ากัน" ของความสัมพันธ์ให้ตรงกันเสียก่อน ในที่นี้จะใช้ความหมายจาก [14] ซึ่งได้กำหนดความเท่ากันของความสัมพันธ์ว่า "ความสัมพันธ์สองความสัมพันธ์ จะเท่ากันก็ต่อเมื่อข้อมูลในแอททริบิวเดียวกันของความสัมพันธ์ทั้งสองมีค่าเท่ากันทุกแอททริบิวที่มีชื่อเหมือนกัน โดยไม่สนใจว่าลำดับของแอททริบิวในความสัมพันธ์ทั้งสองจะอยู่ในลำดับเดียวกันหรือไม่ก็ตาม" หรือกล่าวอีกนัยหนึ่งได้ว่าลำดับของแอททริบิวไม่มีความสำคัญแต่อย่างใด

เรามาพิจารณานิพจน์ (Expression) 2 นิพจน์คือ นิพจน์ E_1 และ E_2 ของพีชคณิตสัมพันธ์ ความสัมพันธ์ที่ปรากฏในนิพจน์ทั้งสองจะเขียนแทนด้วยตัวแปร เราจะกล่าวว่านิพจน์ทั้งสองมีความเท่ากัน ก็ต่อเมื่อ แทนค่าตัวแปรด้วยความสัมพันธ์แล้ว ผลลัพธ์ที่ได้จากนิพจน์ทั้งสองเป็นผลลัพธ์เดียวกัน ซึ่งเขียนแทนด้วย

$$E_1 \leftrightarrow E_2$$

กำหนดให้ R , S และ T แทนความสัมพันธ์ ให้ U แทนตัวปฏิบัติการพีชคณิตสัมพันธ์ชนิดใช้กับความสัมพันธ์เดียว (Unary Operations) และ ให้ B แทนตัวปฏิบัติการพีชคณิตสัมพันธ์ชนิดใช้กับสองความสัมพันธ์ (Binary Operations) การเปลี่ยนรูปที่เท่ากันสำหรับพีชคณิตสัมพันธ์มีการจัดแบ่งเป็นกลุ่มได้ดังนี้

กลุ่มที่ 1 การสลับที่ของตัวปฏิบัติการชนิดใช้กับความสัมพันธ์เดียว

$$U_1\, U_2\, R \leftrightarrow U_2\, U_1\, R$$

กลุ่มที่ 2 การสลับที่ของความสัมพันธ์ที่ใช้กับตัวปฏิบัติการชนิดใช้กับสองความสัมพันธ์

$$R\, B\, S \leftrightarrow S\, B\, R$$

กลุ่มที่ 3 การจัดหมู่ของตัวปฏิบัติการชนิดใช้กับสองความสัมพันธ์

$$R\, B\, (S\, B\, T) \leftrightarrow (R\, B\, S)\, B\, T$$

กลุ่มที่ 4 การรวมตัวปฏิบัติการชนิดใช้กับความสัมพันธ์เดียว

$$\mathrm{U}\,R \leftrightarrow \mathrm{U}_1\,\mathrm{U}_2\,R$$

กลุ่มที่ 5 การกระจายของตัวปฏิบัติการชนิดใช้ความสัมพันธ์เดียว

$$\mathrm{U}\,(R\,\mathrm{B}\,S) \rightarrow \mathrm{U}\,(R)\,\mathrm{B}\,\mathrm{U}\,(S)$$

กลุ่มที่ 6 การถึงตัวร่วมของตัวปฏิบัติการชนิดใช้ความสัมพันธ์เดียว

$$\mathrm{U}\,(R)\,\mathrm{B}\,\mathrm{U}\,(S) \rightarrow \mathrm{U}\,(R\,\mathrm{B}\,S)$$

	SL_{F2}	PJ_{A2}
$\mathrm{SL}_{F1}(*(R)) \rightarrow *(\mathrm{SL}_{F1}(R))$	Y	Y
$\mathrm{PJ}_{A1}(*(R)) \rightarrow *(\mathrm{PJ}_{A1}(R))$	SNC_1	SNC_2

$SNC_1 : Attr(F2) \subseteq A1$

$SNC_2 : A1 \equiv A2$

ตารางที่ 3.1 แสดงการสลับที่ของตัวปฏิบัติการชนิดใช้ความสัมพันธ์เดียว

	UN	DF	CP	JN_F	SJ_F
$R*S \rightarrow S*R$	Y	N	Y	Y	N
$(R*S)*T \rightarrow R*(S*T)$	Y	N	Y	SNC_1	N

SNC_1 for $(R\ \mathrm{JN}_{F1}\ S)\ \mathrm{JN}_{F2}\ T \rightarrow R\ \mathrm{JN}_{F1}\ (S\ \mathrm{JN}_{F2}\ T) : Attr(F2) \subseteq Attr(S) \cup Attr(T)$

ตารางที่ 3.2 แสดงการสลับที่ของความสัมพันธ์และการจัดหมู่
ของตัวปฏิบัติการชนิดใช้สองความสัมพันธ์

$\mathrm{PJ}_A(R) \rightarrow \mathrm{PJ}_{A1}\mathrm{PJ}_{A2}(R)$	$SNC : A \equiv A1, A \subseteq A2$
$\mathrm{SL}_F(R) \rightarrow \mathrm{SL}_{F1}\mathrm{SL}_{F2}(R)$	$SNC : F = F1 \wedge F2$

ตารางที่ 3.3 แสดงการแยกตัวปฏิบัติการ

	UN	DF	CP	JN_{F3}	SJ_{F3}
$\mathrm{SL}_{F}(R*S) \rightarrow$ $\mathrm{SL}_{FR}(R)*\mathrm{SL}_{FS}(S)$	Y	Y	SNC_1	SNC_1	Y
	$FR = F$, $FS = F$	$FR = F$, $FS = F$	$FR = F1$, $FS = F2$	$FR = F1$, $FS = F2$	$FR = F$, $FS = TRUE$
$\mathrm{PJ}_{A}(R*S) \rightarrow$ $\mathrm{PJ}_{AR}(R)*\mathrm{PJ}_{AS}(S)$	Y	N	Y	SNC_2	SNC_2
	$AR = A$, $AS = A$		$AR = A - Attr(S)$ $AS = A - Attr(R)$	$AR = A - Attr(S)$ $AS = A - Attr(R)$	$AR = A - Attr(S)$ $AS = Attr(S) \cap Attr(F3)$

$SNC_1 : \exists F1, F2 : (F = F1 \wedge F2) \wedge (Attr(F1) \subseteq Attr(F1) \wedge (Attr(F2) \subseteq Attr(S))$ $SNC_2 : Attr(F3) \subseteq A$

ตารางที่ 3.4 แสดงการกระจายของตัวปฏิบัติการชนิดใช้ความสัมพันธ์เดียว
ไปสู่ตัวปฏิบัติการชนิดใช้สองความสัมพันธ์

	UN	DF	CP	JN_{F1}	SJ_{F1}
$SL_{FR}(R)*SL_{FS}(S)$	SNC_1	SNC_2	Y	Y	SNC_4
$\rightarrow SL_R(R*S)$	$F = FR = FS$	$F = FR$	$F = FR \wedge FS$	$F = FR \wedge FS$	$F = FR$
$PJ_{AR}(R)*PJ_{AS}(S)$	SNC_3	N	Y	Y	Y
$\rightarrow PJ_A(R*S)$	$A = AR = AS$		$A = AR \cup AS$	$A = AR \cup AS$	$A = AR$

$SNC_1 : FR = FS$
$SNC_2 : FR \Rightarrow FS$
$SNC_3 : Attr(R) = Attr(S)$
$SNC_4 : FS = \text{true}$

ตารางที่ 3.5 แสดงการดึงตัวร่วมของตัวปฏิบัติการชนิดใช้ความสัมพันธ์เดียว จากตัวปฏิบัติการชนิดใช้สองความสัมพันธ์

คุณสมบัติของตัวปฏิบัติการได้แสดงไว้ในตารางที่ 3.1 ถึง 3.5 โดยจะแสดงการใช้ตัวปฏิบัติการต่างๆร่วมกัน เท่าที่จะปฏิบัติการได้ ข้อความ Att (F) ใช้แทนแอททริบิวที่ปรากฏในสูตร F และ Att (R) แทนแอททริบิวทั้งหมดของความสัมพันธ์ R

ในตารางแต่ละช่องจะมีตัวแสดงค่าความถูกต้อง (Validity Indicator) ของการใช้ตัวปฏิบัติการร่วมกัน ถ้าเป็นเครื่องหมาย Y หมายถึงคุณสมบัตินั้นสามารถใช้ได้ทุกกรณี เครื่องหมาย N หมายความว่าคุณสมบัตินั้นไม่สามารถใช้ได้ เช่น ในแถวแรกและหลักแรกของตารางที่ 3.1 จะเป็นเครื่องหมาย Y หมายความว่าการเปลี่ยนรูปต่อไปนี้ถูกต้อง

$$SL_{F1}\ SL_{F2}\ R \longrightarrow SL_{F2}\ SL_{F1}\ R$$

เมื่อ F1 และ F2 เป็นการระบุการเลือกของตัวปฏิบัติการ Selection

ตัวแสดงค่าความถูกต้องอาจจะเป็นเครื่องหมาย SNC ก็ได้ ซึ่งจะระบุเงื่อนไขที่จะทำให้การเปลี่ยนรูปของนิพจน์เป็นจริง เช่น ตัวแสดงค่าความถูกต้อง SNC_1 ในแถวที่สองและหลักแรกของตาราง 3.1 หมายความว่า การเปลี่ยนรูป

$$PJ_{A1}\ SL_{F2}\ R \longrightarrow SL_{F2}\ PJ_{A1}\ R$$

จะเป็นจริง ถ้า A1 และ F2 เป็นไปตามเงื่อนไข SNC_1 ซึ่งได้แสดงไว้ในส่วนล่างของตาราง 3.1 จากเงื่อนไข SNC_1 ระบุว่าตัวปฏิบัติการ PJ_{A1} (PROJECTION) จะต้องไม่ตัดแอททริบิวที่ใช้ในการคำนวนของสูตร F2 หรือกล่าวอีกนัยหนึ่งจะได้ว่า แอททริบิวที่ใช้ในการคำนวนของสูตร F2 จะต้องถูกระบุใน A1 ด้วย

สังเกตว่า ถ้าเราพิจารณาการเปลี่ยนรูปนิพจน์ที่กลับกันจากที่กล่าวมาข้างต้น คือ

$$SL_{F1}\ PJ_{A2}\ R \longrightarrow PJ_{A2}\ SL_{F1}\ R$$

เราจะพบว่า ตัวแสดงค่าความถูกต้องจะเป็น Y (แถวแรกและหลักที่สอง ของตาราง 3.1) นั่นแสดงว่า ตัวแสดงค่าความถูกต้องอาจจะไม่สมมาตรได้ ดังนั้นกฎการเปลี่ยนรูปนี้จะต้องใช้เป็นกรณีไป

ตัวแสดงค่าความถูกต้องที่ระบุเงื่อนไข ที่มีผลกับสูตรที่อยู่ด้านซ้ายมือของกฎการเปลี่ยนรูป จะระบุส่วนของตัวถูกปฏิบัติการที่อยู่ในเงื่อนไขไว้อย่างชัดเจน แต่ในบางกรณีก็จะมีการกำหนดกฎการก่อกำเนิด (Generation Rule) ซึ่งกฎการก่อกำเนิดนี้จะเป็นตัวบอกเงื่อนไข ที่ระบุกับสูตรที่อยู่ด้านขวามือของกฎการเปลี่ยนรูป โดยการกำหนดจากเงื่อนไขที่ระบุกับสูตรที่อยู่ด้านซ้ายมือของกฎการเปลี่ยนรูป กฎการก่อกำเนิดจะไม่มีผล เมื่อระบุเงื่อนไขกับสูตรที่อยู่ด้านซ้ายมือและด้านขวามือของกฎการเปลี่ยนรูป ด้วยเงื่อนไขเดียวกัน

กฎการก่อกำเนิดที่แสดงไว้ในตารางที่ 3.4 และ 3.5 จะเขียนไว้ใต้ตัวแสดงค่าความถูกต้องที่สัมพันธ์กัน เช่น ตารางที่ 3.4 ในช่องตำแหน่งแถวที่สองและหลักที่สาม จะมีตัวแสดงค่าความถูกต้องเป็น Y และกฎการก่อกำเนิดมีความหมายว่า

$$PJ_A\ (R\ CP\ S) \longrightarrow PJ_{AR}\ (R)\ CP\ PJ_{AS}\ (S)$$

เป็นจริง เมื่อ AR และ AS ถูกกำหนดเป็น

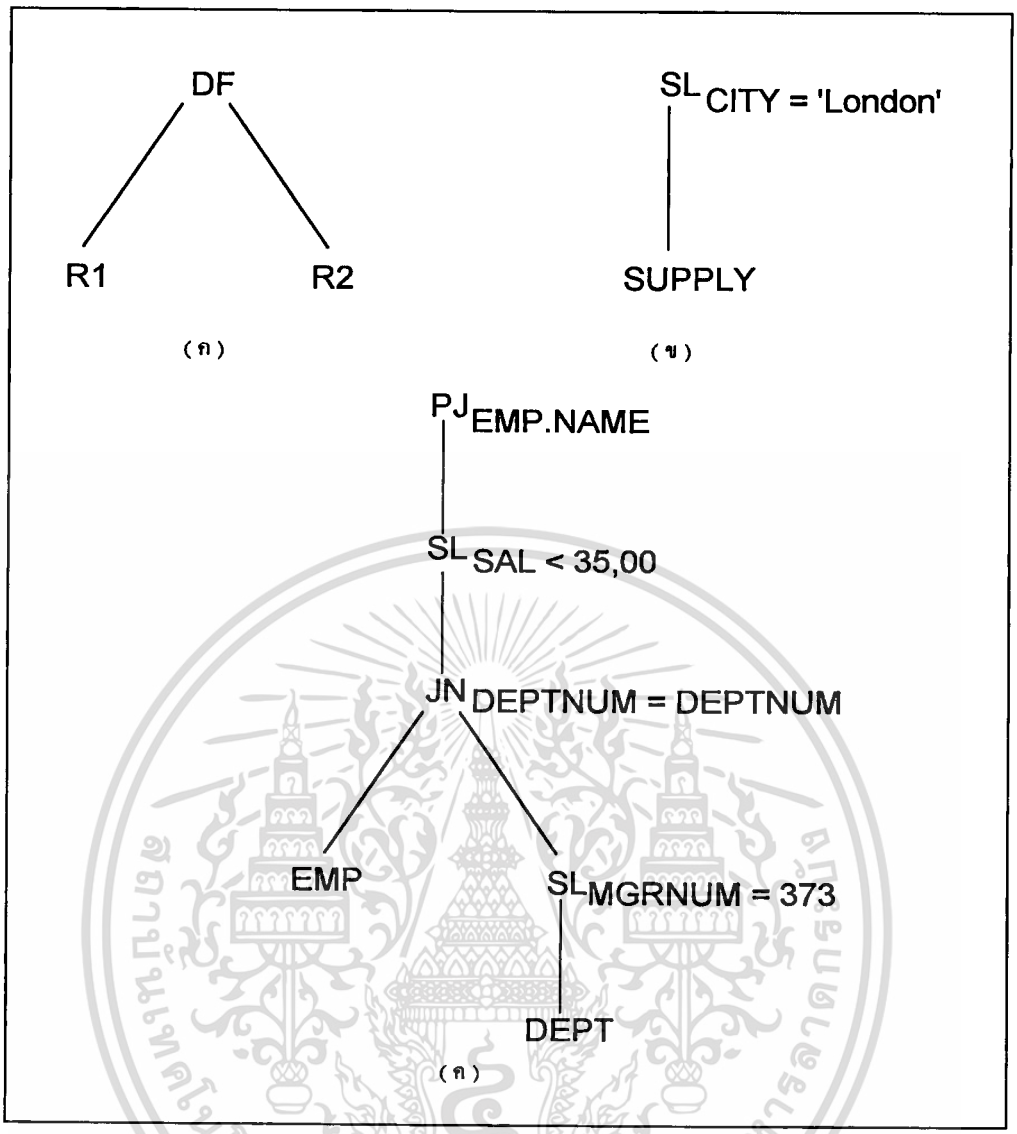
$$AR = A - Attr\ (S)\ ,\ AS = A - Attr\ (R)$$

3.4.2 คำถามในรูปตัวปฏิบัติการเชิงต้นไม้ (Operator Tree of a Query)

คำถามในภาษาพีชคณิตสัมพันธ์นั้น โดยปรกติจะมีลำดับขั้นในการประมวลผลที่แน่นอน แต่ในบางครั้งการลำดับขั้นตอนของการประมวลผลอาจจะมีได้มากกว่า 1 วิธี เช่น

A UNION B UNION C

เราอาจจะลำดับขั้นตอนการประมวลผลเป็นดังนี้ คือ A UNION B ก่อน จากนั้นจึงนำผลที่ได้ไปปฏิบัติการ UNION กับ C หรืออีกวิธีหนึ่ง คือ B UNION C แล้วจึงนำผลลัพธ์ไปปฏิบัติการ UNION กับ A ทั้งสองวิธีจะให้ผลลัพธ์สุดท้ายเหมือนกัน อย่างไรก็ตาม ในการส่งคำถามให้กับโปรแกรมประมวลผล จะต้องมีการกำหนดลำดับขั้นตอน ของการประมวลผลที่แน่นอนไว้เพียงแบบเดียว ดังนั้นจึงใช้การแสดงคำถามด้วยคำถามในรูปตัวปฏิบัติการเชิงต้นไม้ ซึ่งจะแสดงลำดับขั้นตอนการประมวลผลที่แน่นอน



รูปที่ 3.7 แสดงตัวอย่างคำถามเชิงต้นไม้

ในการเขียนคำถามในรูปตัวปฏิบัติการเชิงต้นไม้นั้น โดยปรกติจะประกอบด้วย 3 โนด โนดที่อยู่ด้านซ้ายและขวาจะแทนความสัมพันธ์ที่ถูกปฏิบัติการ ส่วนโนดที่อยู่กลางจะแทนตัวปฏิบัติการ ในกรณีของตัวปฏิบัติการ DIFFERENCE โนดที่อยู่ทางซ้ายจะแทนความสัมพันธ์ที่เป็นตัวตั้ง ส่วนโนดที่อยู่ทางขวามือจะแทนความสัมพันธ์ที่มาลบ ดังรูปที่ 3.7 (ก)

ตัวปฏิบัติการอาจจะประกอบด้วยโนด 2 โนดก็ได้ ในกรณีนี้ตัวปฏิบัติการจะเป็นชนิดใช้ความสัมพันธ์เดียว เช่น SELECTION หรือ PROJECTION เป็นต้น ดังแสดงในรูปที่ 3.7 (ข)

ในกรณีที่คำถามมีความซับซ้อน โนดที่อยู่ด้านซ้ายและด้านขวา อาจจะแทนด้วยส่วนของคำถามเชิงต้นไม้ก็ได้ ดังแสดงตัวอย่างในรูปที่ 3.7 (ค)

จากที่กล่าวมาข้างต้นสรุปเป็นไวยากรณ์ได้ ดังต่อไปนี้

$R \rightarrow$ identifier

$R \rightarrow (R)$

$R \rightarrow$ un_op R

$R \rightarrow R$ bin_op R

un_op $\rightarrow SL_F \mid PJ_A$

bin_op $\rightarrow CP \mid UN \mid DF \mid JN_F \mid IN$

3.5 การเข้าถึงข้อมูลที่ดีที่สุด (Query Optimization)

ในการกล่าวถึงวิธีการเข้าถึงข้อมูลที่ดีที่สุด ในที่นี้จะหมายถึงการเข้าถึงข้อมูลที่มีการใช้จ่ายน้อยที่สุด รวมทั้งใช้เวลาในการประมวลผลน้อยที่สุดด้วย โดยทั่วไปแล้วการใช้จ่ายส่วนใหญ่ในการเข้าถึงข้อมูลในระบบฐานข้อมูลแบบกระจาย จะเป็นค่าใช้จ่ายในการรับ-ส่งข้อมูล ดังนั้นหากเราลดปริมาณการส่งข้อมูลให้เหลือน้อยที่สุด ก็จะประมาณได้ว่ามีค่าใช้จ่ายน้อยที่สุด

โดยทั่วไปในการเข้าถึงข้อมูลที่ดีที่สุดนั้น จะต้องพิจารณาสภาพแวดล้อม (environment) ในการประมวลผลด้วย ทั้งนี้เนื่องจากวิธีการเข้าถึงข้อมูลที่ดีที่สุดในสภาพแวดล้อมหนึ่ง อาจจะเข้าถึงข้อมูลได้ไม่ดีที่สุดในอีกสภาพแวดล้อมหนึ่งก็ได้ ดังนั้นจึงเป็นการยากที่กล่าวว่าวิธีการใดจะเข้าถึงข้อมูลได้ดีที่สุด แต่อย่างไรก็ตามในที่นี้ยังคงใช้คำว่า "วิธีการเข้าถึงข้อมูลที่ดีที่สุด (Optimization)" ซึ่งเป็นที่เข้าใจกันทั่วไป

จากที่ได้กล่าวไว้ในตอนต้นแล้วว่า ระบบ DDQ/1 ที่ได้พัฒนาขึ้นนี้ เป็นการสร้างระบบฐานข้อมูลแบบกระจาย จากระบบฐานข้อมูลรวมที่มีอยู่แล้ว ดังนั้นวิธีการเข้าถึงข้อมูลที่ดีที่สุด คงจะทำได้ในระดับโดยรวมเท่านั้น (Global Query Optimization) คือ การกำหนดสถานที่เก็บข้อมูล , การปรับปรุงรูปร่างของคำถามเชิงต้นไม้ ซึ่งจะได้กล่าวรายละเอียดเป็นหัวข้อๆไป

3.5.1 การกำหนดสถานที่เก็บข้อมูล

ในกรณีที่ความสัมพันธ์มีการแยกเป็นส่วนย่อย และในแต่ละส่วนย่อยถูกเก็บไว้ในหลายสถานที่ จำเป็นจะต้องกำหนดสถานที่ที่ใช้ในการเข้าถึงข้อมูล การกำหนดนี้จะต้องให้เกิดผลในการลดเวลาการรับ-ส่งข้อมูลด้วย

ในระบบ DDQ/1 นี้ความสัมพันธ์ในระบบจะไม่มีการแยกเป็นส่วนย่อย ดังนั้นในการกำหนดสถานที่เก็บข้อมูลที่จะเข้าถึง จึงเป็นไปตามสถานที่อยู่จริงของข้อมูล โดยจะค้นหาได้จากพจนานุกรมฐานข้อมูล

3.5.2 การปรับปรุงรูปคำถามเชิงต้นไม้

จากหัวข้อที่ 3.4.1 เราทราบแล้วว่า ในนิพจน์หนึ่งๆของคำถามพีชคณิตสัมพันธ์ อาจจะเขียนให้อยู่ในรูปนิพจน์อื่นๆได้มากกว่าหนึ่งนิพจน์ โดยที่ผลลัพธ์ยังคงเหมือนเดิม แม้ว่านิพจน์ต่างๆที่สามารถเปลี่ยนรูปไปได้นี้ จะได้ผลลัพธ์เหมือนกันก็ตาม แต่ละนิพจน์ก็จะใช้เวลาในการประมวลผลต่างกัน ดังนั้นเราจะเลือกเปลี่ยนรูปเป็นนิพจน์ที่ใช้เวลาในการประมวลผลน้อยที่สุด หรือจะกล่าวอีกนัยหนึ่งว่าใช้ข้อมูลในการประมวลน้อยที่สุด ซึ่งจะสรุปเป็นกฎในการปรับปรุงคำถามได้ดังนี้

3.5.2.1 กฎที่เกี่ยวกับตัวปฏิบัติการ SELECTION

3.5.2.1.1 รวมตัวปฏิบัติการ SELECTION ที่ปฏิบัติการหลายตัวต่อเนื่องกัน ให้เป็นตัวปฏิบัติการ SELECTION เพียงตัวเดียว โดยรวมเงื่อนไขในการเลือกทั้งหมด เป็นเงื่อนไขเดียว ดังแสดงในรูปที่ 3.8 นั่นคือ

(...((R WHERE p1) WHERE p2) ... WHERE pn)

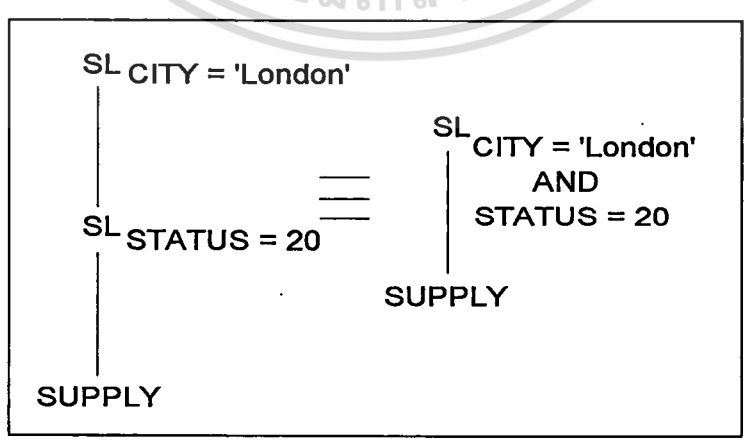
= (R WHERE (p1 AND p2 AND ... AND pn))

3.5.2.1.2 สลับตำแหน่งตัวปฏิบัติการ SELECTION กับตัวปฏิบัติการ Cartesian Product ซึ่งจะเป็นไปได้ 3 กรณีคือ

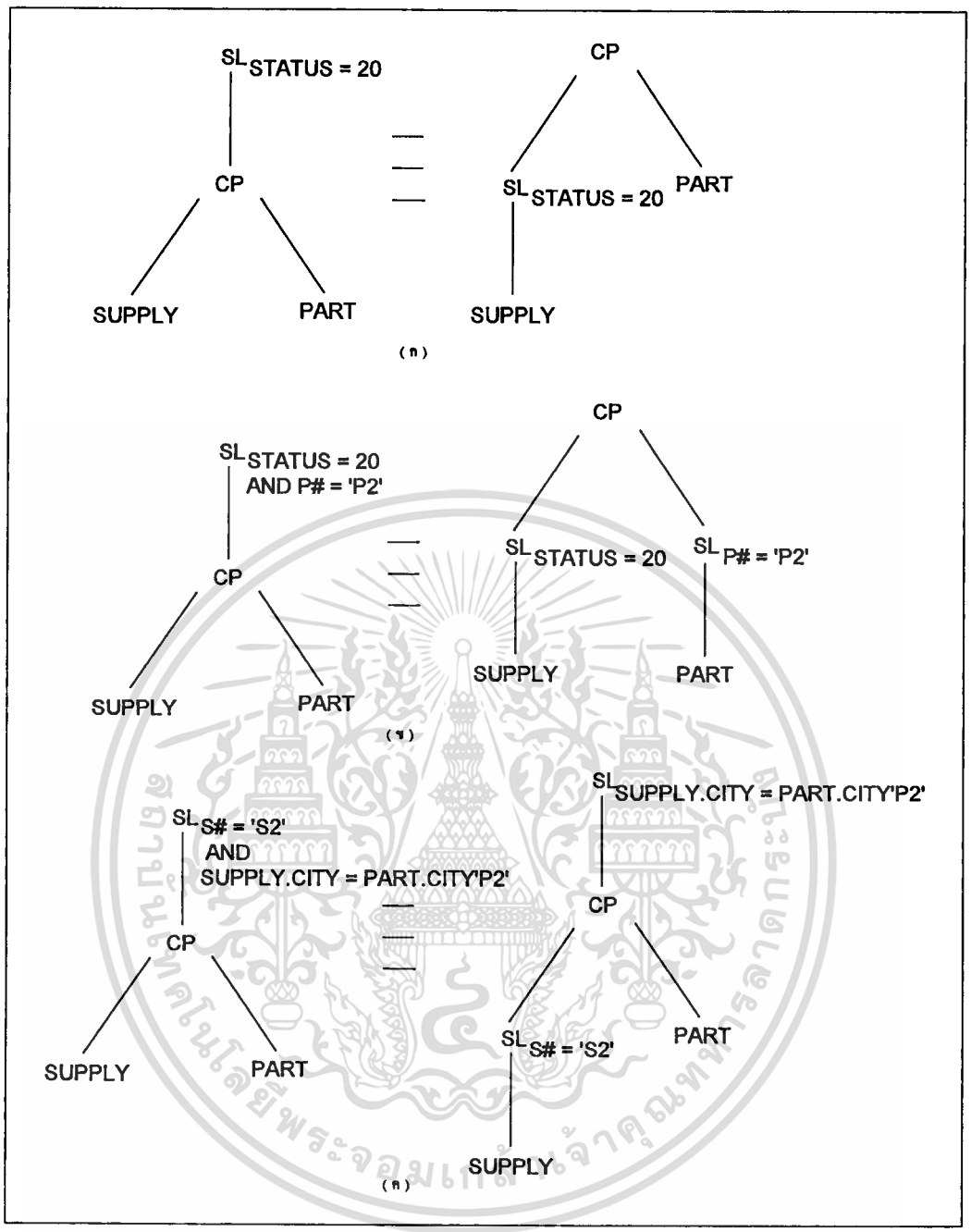
กรณีที่ 1

$$(R_1 \text{ TIMES } R_2) \text{ WHERE } p = (R_1 \text{ WHERE } p) \text{ TIMES } R_2$$

ถ้าแอททริบิวทั้งหมดที่อยู่ในเงื่อนไขการเลือก p เป็นของความสัมพันธ์ R_1 ดังแสดงในรูปที่ 3.9 (ก)



รูปที่ 3.8 แสดงตัวอย่างการรวมตัวปฏิบัติการ SELECTION



รูปที่ 3.9 แสดงการสลับตำแหน่งตัวปฏิบัติการ SELECTION
กับตัวปฏิบัติการ Cartesian Product

กรณีที่ 2

(R_1 TIMES R_2) WHERE (p1 AND p2) = (R_1 WHERE p1) TIMES (R_2 WHERE p2)

ถ้าแอททริบิวทั้งหมดที่อยู่ในเงื่อนไขการเลือก p1 เป็นของความสัมพันธ์ R_1 และแอททริบิวทั้งหมดที่อยู่ในเงื่อนไขการเลือก p2 เป็นของความสัมพันธ์ R_2 ดังแสดงในรูปที่ 3.9 (ข)

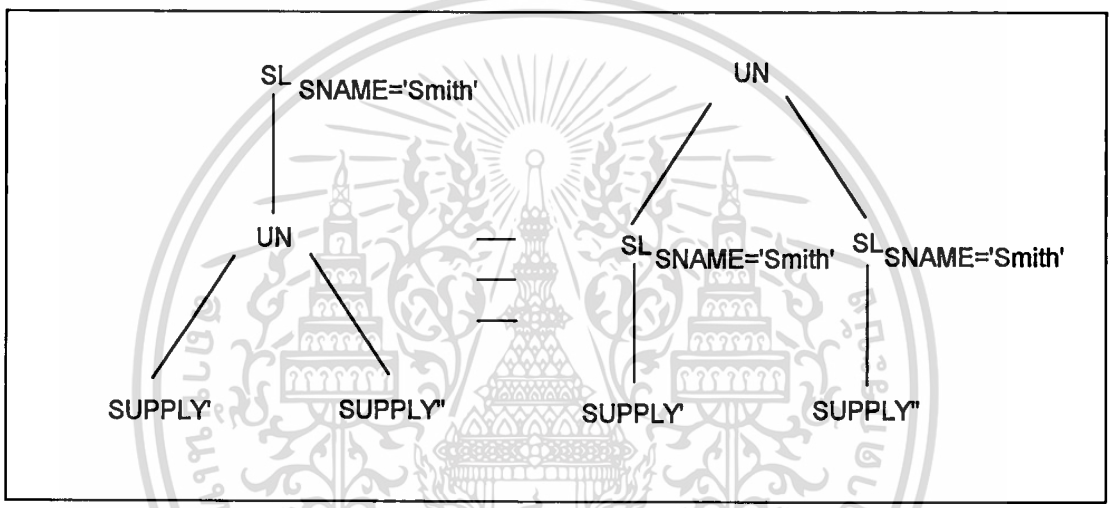
กรณีที่ 3

(R_1 TIMES R_2) WHERE (p1 AND p2) = ((R_1 WHERE p1) TIMES R_2) WHERE p2

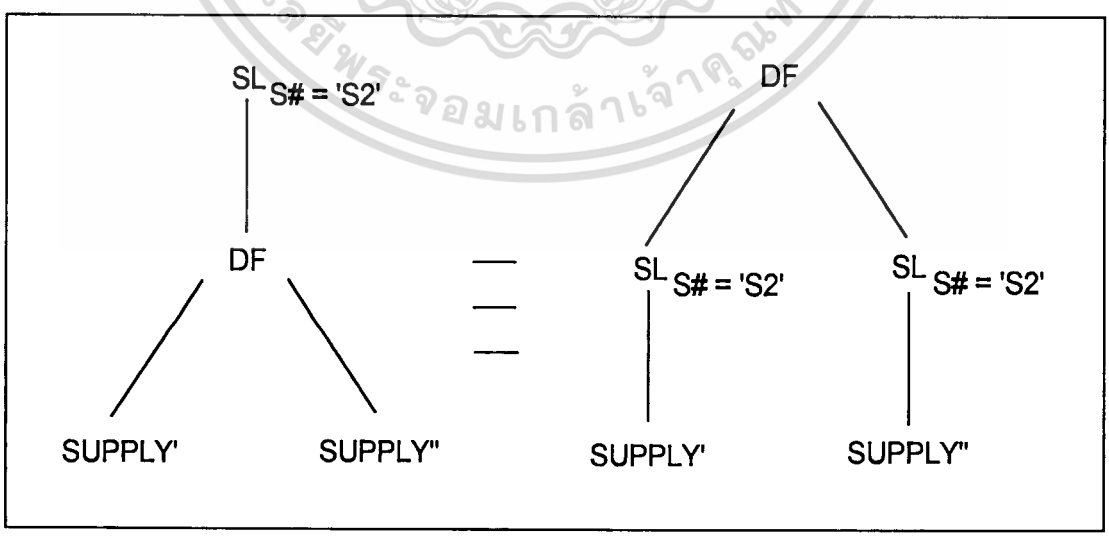
ถ้าแอททริบิวทั้งหมดที่อยู่ในเงื่อนไขการเลือก p1 เป็นของความสัมพันธ์ R_1 และแอททริบิวทั้งหมดที่อยู่ในเงื่อนไขการเลือก p2 เป็นของความสัมพันธ์ R_1 และ R_2 ดังแสดงในรูปที่ 3.9 (ค)

3.5.2.1.3 สลับตำแหน่งตัวปฏิบัติการ SELECTION กับตัวปฏิบัติการ UNION ดังรูปที่ 3.10 คือ

(R_1 UNION R_2) WHERE p = (R_1 WHERE p) UNION (R_2 WHERE p)



รูปที่ 3.10 แสดงการสลับตำแหน่งตัวปฏิบัติ SELECTION กับตัวปฏิบัติการ UNION



รูปที่ 3.11 แสดงการสลับตำแหน่งตัวปฏิบัติการ SELECTION กับตัวปฏิบัติการ DIFFERENCE

3.5.2.1.4 สลับตำแหน่งตัวปฏิบัติการ SELECTION กับตัวปฏิบัติการ DIFFERENCE ดังรูปที่ 3.11 คือ

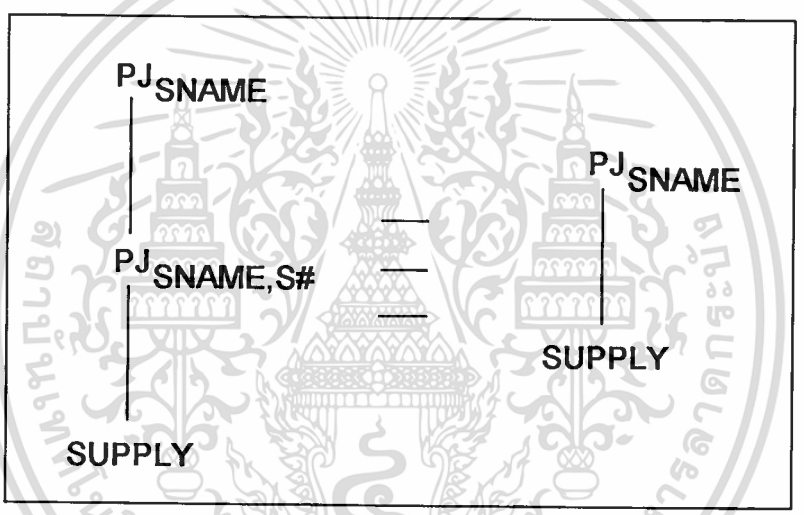
$$(R_1 \text{ MINUS } R_2) \text{ WHERE } p = (R_1 \text{ WHERE } p) \text{ MINUS } (R_2 \text{ WHERE } p)$$

3.5.2.2 กฎที่เกี่ยวกับตัวปฏิบัติการ PROJECTION

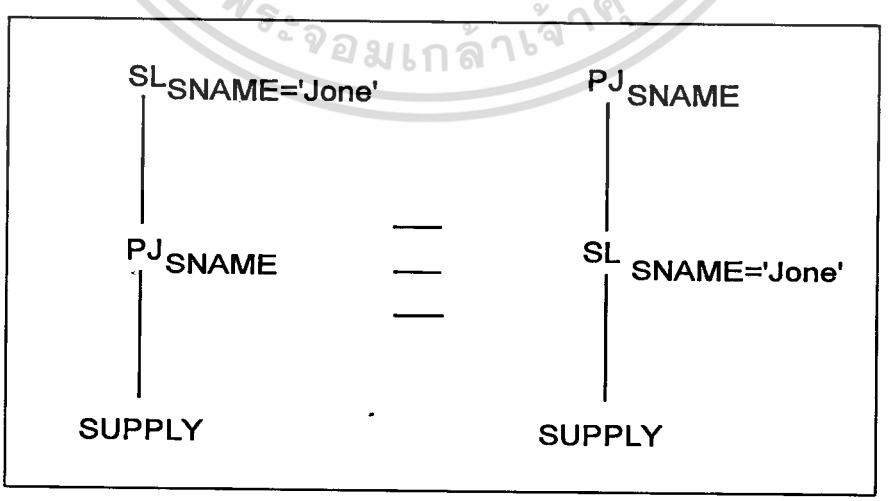
3.5.2.2.1 ตัวปฏิบัติการ PROJECTION หลายตัวที่ปฏิบัติการต่อเนื่องกัน ให้ปฏิบัติการเฉพาะตัวปฏิบัติการตัวสุดท้ายเท่านั้น ดังรูปที่ 3.12 คือ

$$(R [b_1 , \ldots , b_m]) [a_1 , \ldots , a_n] = R [a_1 , \ldots , a_n]$$

เมื่อ แอททริบิว $a_1 , \ldots , a_n$ เป็นซับเซตของแอททริบิว $b_1 , \ldots , b_m$



รูปที่ 3.12 แสดงการลดจำนวนตัวปฏิบัติการ PROJECTION ที่ปฏิบัติการต่อเนื่องกัน



รูปที่ 3.13 แสดงการสลับตำแหน่งตัวปฏิบัติการ SELECTION กับตัวปฏิบัติการ PROJECTION

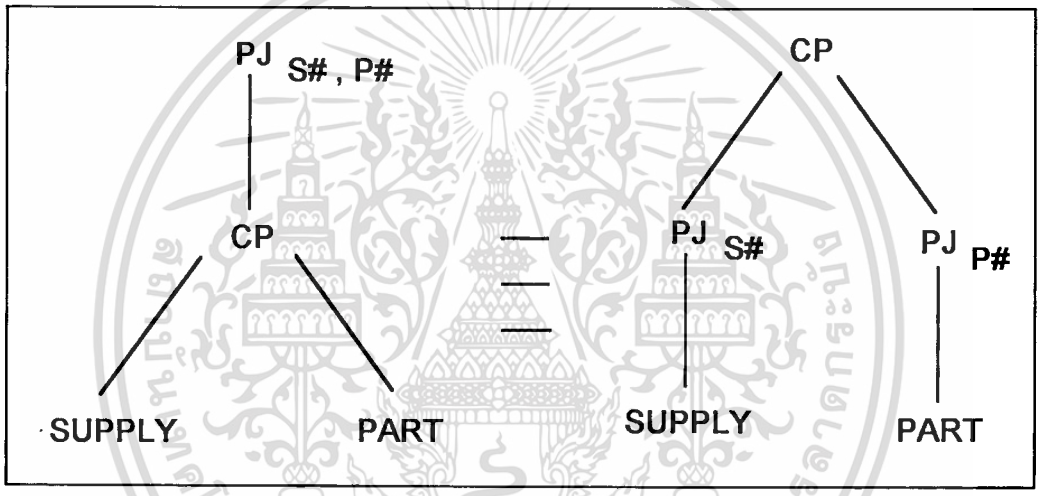
3.5.2.2.2 สลับตำแหน่งตัวปฏิบัติการ SELECTION กับตัวปฏิบัติการ PROJECTION ดังรูปที่ 3.13 นั่นคือ

$$(R [a_1 , \dots , a_n]) \text{ WHERE } p = (R \text{ WHERE } P) [a_1 , \dots , a_n]$$

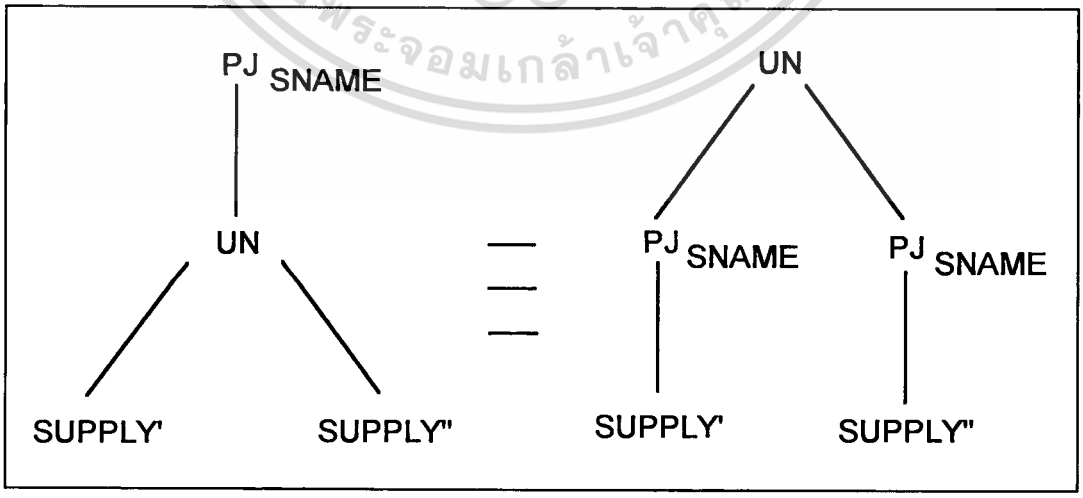
3.5.2.2.3 สลับตำแหน่งตัวปฏิบัติการ PROJECTION กับตัวปฏิบัติการ Cartesian Product ดังรูปที่ 3.14 นั่นคือ

$$(R_1 \text{ TIMES } R_2) [a_1 , \dots , a_n] = (R_1 [b_1 , \dots , b_m]) \text{ TIMES } (R_2 [c_1 , \dots , c_k])$$

เมื่อแอททริบิว $a_1 , \dots , a_n$ เป็นแอททริบิวที่เกิดจากแอททริบิว $b_1 , \dots , b_m$ ของความสัมพันธ์ R_1 และแอททริบิว $c_1 , \dots , c_k$ ของความสัมพันธ์ R_2



รูปที่ 3.14 แสดงการสลับตำแหน่งตัวปฏิบัติการ PROJECTION กับตัวปฏิบัติการ CARTESIAN PRODUCT



รูปที่ 3.15 แสดงการสลับตำแหน่งตัวปฏิบัติการ PROJECTION กับตัวปฏิบัติการ UNION

3.5.2.2.4 สลับตำแหน่งตัวปฏิบัติการ PROJECTION กับตัวปฏิบัติการ UNION ดังรูปที่ 3.15 นั่นคือ

$$(R_1 \text{ UNION } R_2) [a_1 , \dots , a_n] = (R_1 [a_1 , \dots , a_n]) \text{ UNION } (R_2 [a_1 , \dots , a_n])$$

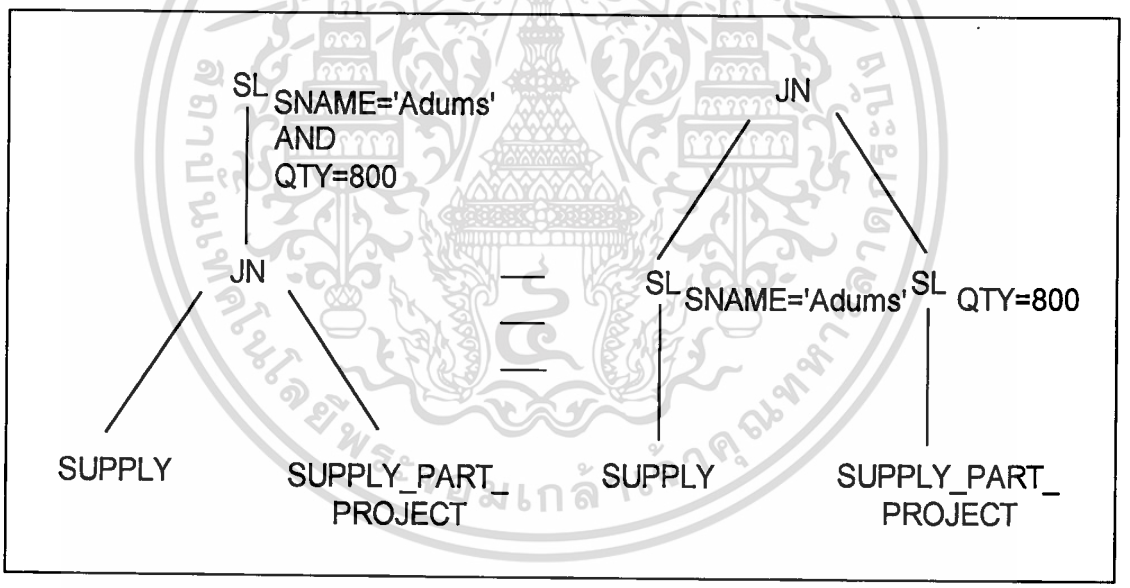
3.5.2.3 กฎที่เกี่ยวกับตัวปฏิบัติการ JOIN

ปฏิบัติการ SELECTION ก่อน แล้วจึงปฏิบัติการ JOIN ดังรูปที่ 3.16 นั่นคือ

$$(R_1 \text{ JOIN } R_2) \text{ WHERE } (p1 \text{ AND } p2) = (R_1 \text{ WHERE } p1) \text{ JOIN } (R_2 \text{ WHERE } p2)$$

เมื่อ เงื่อนไขในการเลือก pi ระบุถึงสำหรับความสัมพันธ์ R_i , i = 1 , 2

จะเห็นว่ากฎต่างๆในการปรับปรุงรูปคำถามเชิงต้นไม้ที่ได้กล่าวมาข้างต้นนั้น สรุปได้ว่าจะพยายามสลับตำแหน่งของตัวปฏิบัติการ เพื่อที่จะให้ตัวปฏิบัติการชนิดใช้ความสัมพันธ์เดียว ได้ปฏิบัติงานก่อน



รูปที่ 3.16 แสดงการสลับตำแหน่งตัวปฏิบัติการ SELECTION กับตัวปฏิบัติการ JOIN

บทที่ 4

สถาปัตยกรรมของระบบประมวลผลคำถามแบบกระจาย DDQ/1

4.1 สถาปัตยกรรมของระบบ DDQ/1 โดยรวม

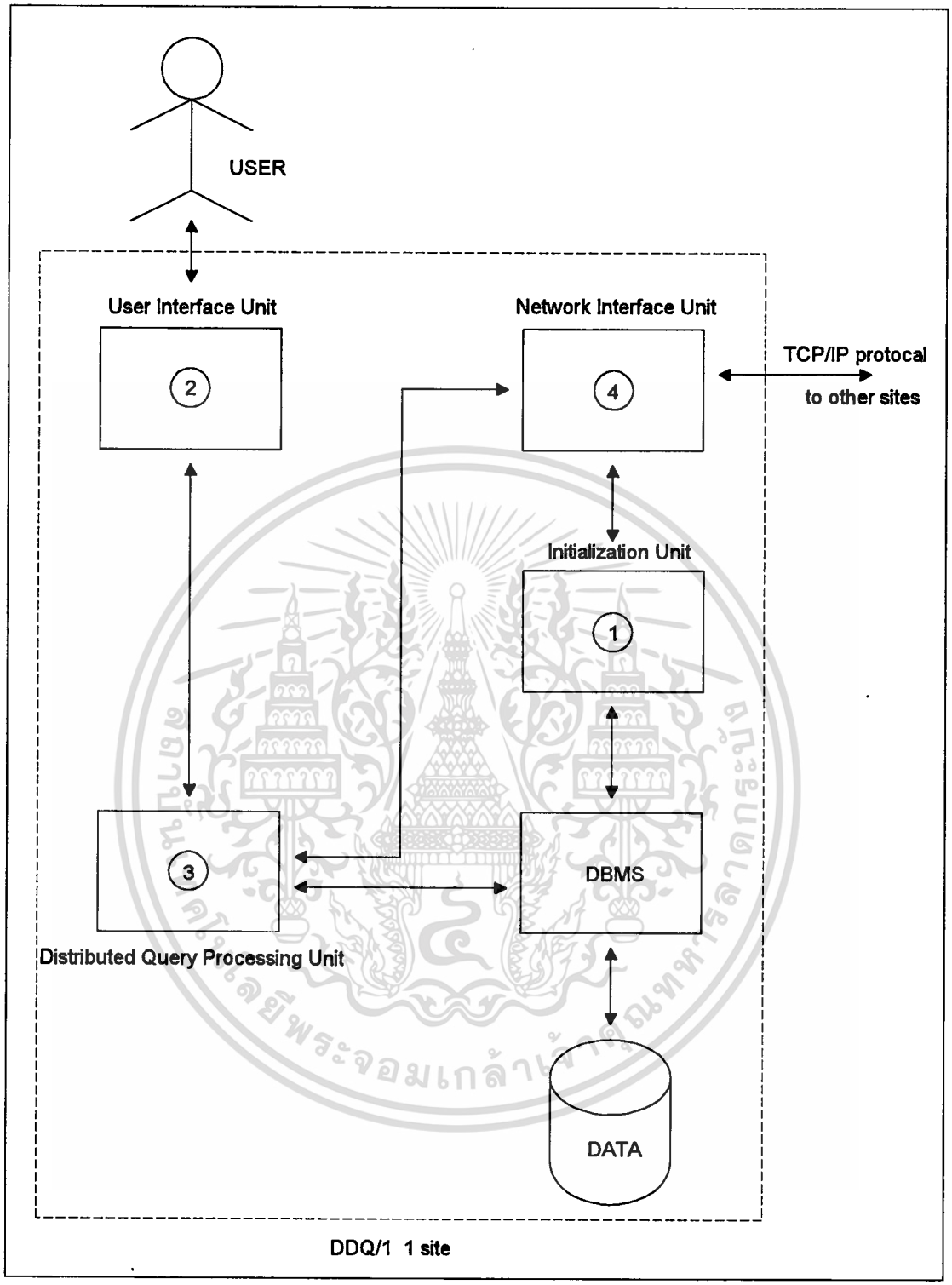
เนื่องจากระบบประมวลผลคำถามแบบกระจาย DDQ/1 ที่ได้พัฒนาขึ้นนี้ ได้ออกแบบสำหรับใช้กับระบบปฏิบัติการทั่วไป ที่เป็นระบบปฏิบัติการแบบหลายงาน (multitasking) คือ สามารถทำงานพร้อมกันได้หลายๆงาน รวมทั้งมีคุณสมบัติที่สามารถสร้างกระบวนการ (process) ขึ้นจากกระบวนการที่ประมวลผลอยู่แล้วได้ โดยอาศัยคุณสมบัติของระบบปฏิบัติการที่กล่าวมา ทำให้สามารถออกแบบระบบฐานข้อมูลแบบกระจาย DDQ/1 ให้มีความสามารถที่จะบริการผู้ใช้งานในสถานที่หนึ่ง ได้หลายผู้ใช้งานในเวลาเดียวกัน

เนื่องจากระบบปฏิบัติการยูนิกซ์ (UNIX) เป็นระบบปฏิบัติการที่มีคุณสมบัติตามที่ได้กล่าวมาข้างต้น [18] และเป็นระบบปฏิบัติการที่นิยมใช้กันอย่างแพร่หลาย จึงได้เลือกใช้ระบบปฏิบัติการยูนิกซ์ เป็นระบบปฏิบัติการในการทดลองสร้างระบบประมวลผลคำถามแบบกระจาย DDQ/1 โดยการออกแบบส่วนจัดการฐานข้อมูลแบบกระจาย ที่อยู่บนเครื่องคอมพิวเตอร์แต่ละเครื่อง จะประกอบไปด้วยส่วนทำงานหลักๆ 4 ส่วนด้วยกัน ที่ทำงานประสานกันดังแสดงไว้ในรูปที่ 4.1 ในหัวข้อนี้จะกล่าวถึงการทำงานของแต่ละส่วนโดยสังเขปเท่านั้น เพื่อจะได้เข้าใจถึงการทำงานร่วมกันของทั้ง 4 ส่วน และการเชื่อมต่อกับฐานข้อมูลที่อยู่บนคอมพิวเตอร์เครื่องอื่น รายละเอียดในแต่ละส่วนจะได้กล่าวถึงในหัวข้อ 4.2 ถึง 4.5 ต่อไป สำหรับการทำงานร่วมกันของทั้ง 4 ส่วนมีดังต่อไปนี้

4.1.1 ส่วนจัดการกระบวนการเริ่มต้น

ส่วนนี้จะเป็นโปรแกรมที่มีหน้าที่จัดการเกี่ยวกับพจนานุกรมฐานข้อมูล (Data Dictionary หรือ System Catalog) ของระบบฐานข้อมูลแบบกระจาย และจัดเก็บกฎบังคับความถูกต้อง (Constraints หรือ Validation Rules) เข้าสู่ฐานข้อมูล โดยผู้ดูแลระบบฐานข้อมูลท้องถิ่น (Local Database Administrator) จะเป็นผู้เรียกใช้โปรแกรมส่วนนี้ในครั้งแรกที่ทำการติดตั้งระบบฐานข้อมูล DDQ/1

ในส่วนจัดเก็บกฎบังคับความถูกต้องนั้น จะรับกฎบังคับความถูกต้องจากผู้ดูแลระบบฐานข้อมูล และจัดเก็บเข้าสู่ฐานข้อมูล ปรกติโปรแกรมส่วนนี้จะถูกเรียกใช้ในตอนติดตั้งระบบเท่านั้น แต่ก็สามารถที่จะเรียกใช้ได้ เมื่อมีการเปลี่ยนแปลงกฎบังคับความถูกต้อง



รูปที่ 4.1 แสดงโครงสร้างระบบฐานข้อมูลแบบกระจาย DDQ/1 บนคอมพิวเตอร์แต่ละเครื่อง

สำหรับส่วนจัดการเกี่ยวกับพจนานุกรมฐานข้อมูล จะทำหน้าที่ในการรับข้อมูลจากผู้ดูแลระบบฐานข้อมูล เก็บเข้าสู่พจนานุกรมฐานข้อมูลท้องถิ่น และยังนำข้อมูลของพจนานุกรมฐานข้อมูลท้องถิ่นจากคอมพิวเตอร์เครื่องอื่นๆ มาสร้างเป็นพจนานุกรมฐานข้อมูลโดยรวมของฐาน

ข้อมูลบนคอมพิวเตอร์เครื่องนั้นๆ ด้วยวิธีการส่งคำถามไปยังคอมพิวเตอร์ทีละเครื่องที่อยู่ในระบบฐานข้อมูลแบบกระจายเดียวกัน ผ่านส่วนเชื่อมต่อโครงข่าย (Network Interface) จากนั้นจะนำข้อมูลที่ได้ส่งให้กับระบบจัดการฐานข้อมูลออราเคิล (ORACLE) จัดเก็บไว้ในตารางพจนานุกรมฐานข้อมูลโดยรวม เพื่อใช้ในการประมวลผลของส่วนอื่นๆต่อไป

หลังจากผู้ดูแลระบบฐานข้อมูลเรียกใช้โปรแกรมส่วนนี้ และโปรแกรมทำงานเสร็จสมบูรณ์แล้ว กระบวนการของโปรแกรมก็จะสิ้นสุดลง และผู้ใช้งานสามารถจะเข้าถึงข้อมูลได้ โดยไม่จำเป็นจะต้องมีกระบวนการของส่วนเริ่มต้นทำงานอยู่

4.1.2 ส่วนติดต่อผู้ใช้งาน

ส่วนติดต่อผู้ใช้งานนี้ จะเป็นโปรแกรมที่ผู้ใช้งานเป็นผู้เรียกใช้ในการเข้าถึงข้อมูล โดยส่วนนี้จะมีหน้าที่ในการรับคำถามจากผู้ใช้งาน เป็นภาษาพีชคณิตสัมพันธ์ และส่งคำถามที่รับมาจากผู้ใช้งานให้กับส่วนประมวลผลคำถามแบบกระจาย ในรูปแบบของต้นไม้เชิงพีชคณิตสัมพันธ์ (Relational Algebra Tree) จากนั้นจะรอรับข้อมูลที่เป็นผลลัพธ์ของคำถามจากส่วนประมวลผลคำถามแบบกระจาย เพื่อนำกลับมาแสดงผลแก่ผู้ใช้งาน และรอรับคำถามใหม่จากผู้ใช้งานต่อไป จนกว่าการทำงานจะสิ้นสุดด้วยการรับคำสั่งสิ้นสุดการทำงานจากผู้ใช้งานโดยตรง

จากการที่ผู้ใช้งานแต่ละคน สามารถเรียกใช้โปรแกรมนี้ได้อย่างอิสระ ทำให้กระบวนการติดต่อผู้ใช้งาน ในแต่ละกระบวนการของผู้ใช้งานแต่ละคน มีการใช้หน่วยความจำเพื่อเก็บข้อมูลเกี่ยวกับกระบวนการ (process information) และคำสั่งทำงาน (instructions) ได้อย่างอิสระ หลังจากที่ผู้ใช้งานเสร็จสิ้นการใช้งานแล้ว กระบวนการติดต่อผู้ใช้งานของผู้ใช้งานนั้นๆ ก็จะสิ้นสุดลงด้วย ทำให้ระบบปฏิบัติการสามารถที่จะนำหน่วยความจำที่ถูกใช้งานโดยกระบวนการที่สิ้นสุดลงไปใช้งานอย่างอื่นได้

4.1.3 ส่วนประมวลผลคำถามแบบกระจาย

ส่วนนี้จะรับคำถามในลักษณะต้นไม้เชิงพีชคณิตสัมพันธ์จากส่วนติดต่อผู้ใช้งาน หรือจากคอมพิวเตอร์เครื่องอื่นที่ส่งผ่านมาทางส่วนเชื่อมต่อโครงข่าย แล้วทำการวิเคราะห์คำถามที่รับมาได้เพื่อให้การประมวลผลคำถามได้ผลลัพธ์ในเวลาน้อยที่สุด และให้มีการส่งข้อมูลน้อยที่สุดในกรณีที่ต้องส่งข้อมูลระหว่างเครื่องคอมพิวเตอร์ จากนั้นจะเปลี่ยนคำถามที่ผ่านการวิเคราะห์แล้วเป็นภาษาสอบถามเชิงโครงสร้าง (SQL) เพื่อส่งให้ระบบจัดการฐานข้อมูลออราเคิล ในกรณีที่คำถามต้องการเข้าถึงข้อมูลในสถานที่อื่น ส่วนประมวลผลคำถามก็จะส่งคำถามในลักษณะต้นไม้เชิงพีชคณิตสัมพันธ์ที่ผ่านการวิเคราะห์แล้ว ไปยังเครื่องคอมพิวเตอร์ที่เก็บข้อมูลอยู่ และรอรับข้อมูลคำตอบส่ง

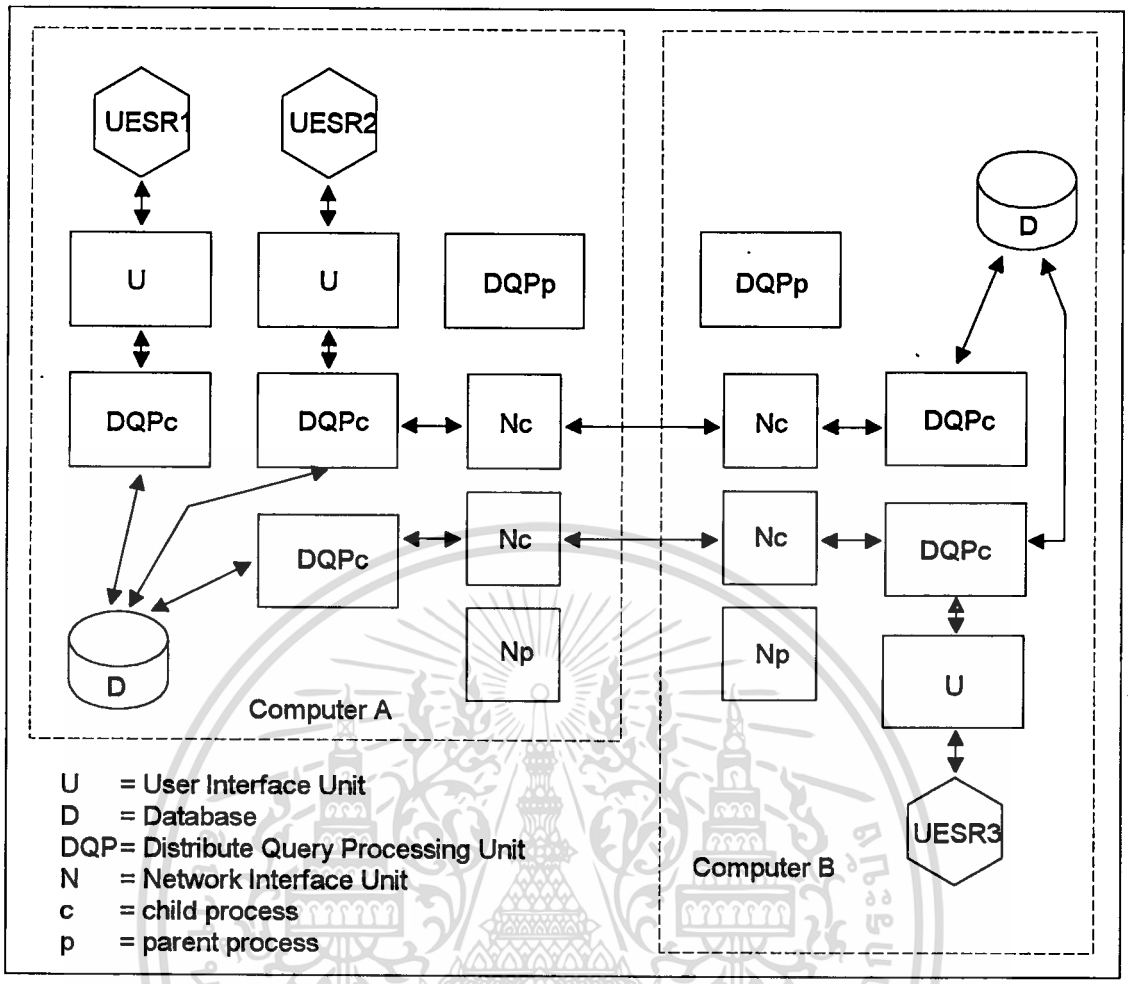
กลับให้กับส่วนที่ส่งคำถามมา การทำงานของกระบวนการประมวลผลคำถามแบบกระจายนี้ จะเป็นลักษณะการทำงานแบบหน่วยบริการ (server) กล่าวคือ โปรแกรมส่วนนี้จะถูกเรียกใช้ครั้งแรกโดยผู้ดูแลเครื่องคอมพิวเตอร์ (System Administrator) แล้วกระบวนการจะรอรับการติดต่อจากกระบวนการส่วนอื่นๆ เมื่อมีการติดต่อมาจากกระบวนการส่วนอื่น กระบวนการประมวลผลคำถามแบบกระจาย ก็จะทำการสร้างกระบวนการขึ้นใหม่ที่มีคุณสมบัติเหมือนเดิม ด้วยการแยก (fork) กระบวนการเป็นสองกระบวนการ ได้แก่ กระบวนการพ่อ (parent process) และกระบวนการลูก (child process) กระบวนการพ่อจะรอรับการติดต่อจากกระบวนการอื่นทันทีที่แยกกระบวนการลูกเสร็จ ส่วนกระบวนการลูกจะทำการประมวลผลคำถามต่อไปจนเสร็จ จากนั้นกระบวนการลูกก็จะสิ้นสุดลง

ด้วยคุณสมบัติการแยกกระบวนการนี้เอง ทำให้เวลาขณะหนึ่งๆ อาจจะมีกระบวนการลูกอยู่หลายกระบวนการก็ได้ ซึ่งกระบวนการลูกทั้งหมดที่แยกตัวออกมาจากกระบวนการพ่อ รวมทั้งกระบวนการพ่อเองด้วย จะมีการใช้หน่วยความจำสำหรับเก็บคำสั่งทำงานร่วมกัน แต่จะแยกเก็บข้อมูลเกี่ยวกับกระบวนการออกจากกัน [18] โดยการทำงานในลักษณะหน่วยบริการนี้ จะทำให้สามารถลดปริมาณการใช้หน่วยความจำลงได้ อย่างไรก็ตามจำนวนกระบวนการจะขึ้นอยู่กับระบบปฏิบัติการ

4.1.4 ส่วนเชื่อมต่อโครงข่าย

ส่วนนี้จะทำหน้าที่คอยรับการติดต่อจากส่วนประมวลผลคำถามแบบกระจาย หรือ ส่วนจัดการกระบวนการเริ่มต้น เพื่อทำหน้าที่เชื่อมต่อกับส่วนเชื่อมต่อโครงข่ายบนคอมพิวเตอร์เครื่องอื่นได้ และทำนองกลับกันก็ทำหน้าที่รับการติดต่อจากส่วนเชื่อมต่อโครงข่ายที่อยู่บนคอมพิวเตอร์เครื่องอื่น ให้สามารถเชื่อมต่อกับส่วนประมวลผลคำถามแบบกระจายได้ เพื่อรับ-ส่งคำถาม หรือข้อมูลระหว่างเครื่องคอมพิวเตอร์ ลักษณะการทำงานของส่วนเชื่อมต่อโครงข่ายนี้ จะมีลักษณะเช่นเดียวกับกระบวนการประมวลผลคำถามแบบกระจาย คือ เป็นแบบหน่วยบริการ กล่าวคือ โปรแกรมจะถูกเรียกใช้โดยผู้ดูแลเครื่องคอมพิวเตอร์ครั้งแรกเพียงครั้งเดียว จากนั้นกระบวนการพ่อจะแยกกระบวนการทุกครั้งที่มีการติดต่อจากกระบวนการอื่น

จากสถาปัตยกรรมที่กล่าวมาข้างต้น จะขอยกตัวอย่างแสดงลักษณะโครงสร้างของกระบวนการทั้งหมดขณะใช้งานจริง ดังรูปที่ 4.2 ซึ่งอธิบายได้ดังนี้คือ ที่เครื่องคอมพิวเตอร์ A ผู้ใช้งาน user1 เรียกใช้โปรแกรมส่วนติดต่อผู้ใช้งาน และมีการป้อนคำถามที่เข้าถึงข้อมูล เฉพาะที่อยู่บนเครื่องคอมพิวเตอร์ A เท่านั้น ส่วนผู้ใช้งาน user2 เรียกใช้โปรแกรมส่วนติดต่อผู้ใช้งานจาก



รูปที่ 4.2 แสดงลักษณะโครงสร้างกระบวนการประมวลผลของฐานข้อมูล DDQ/1 ขณะใช้งาน

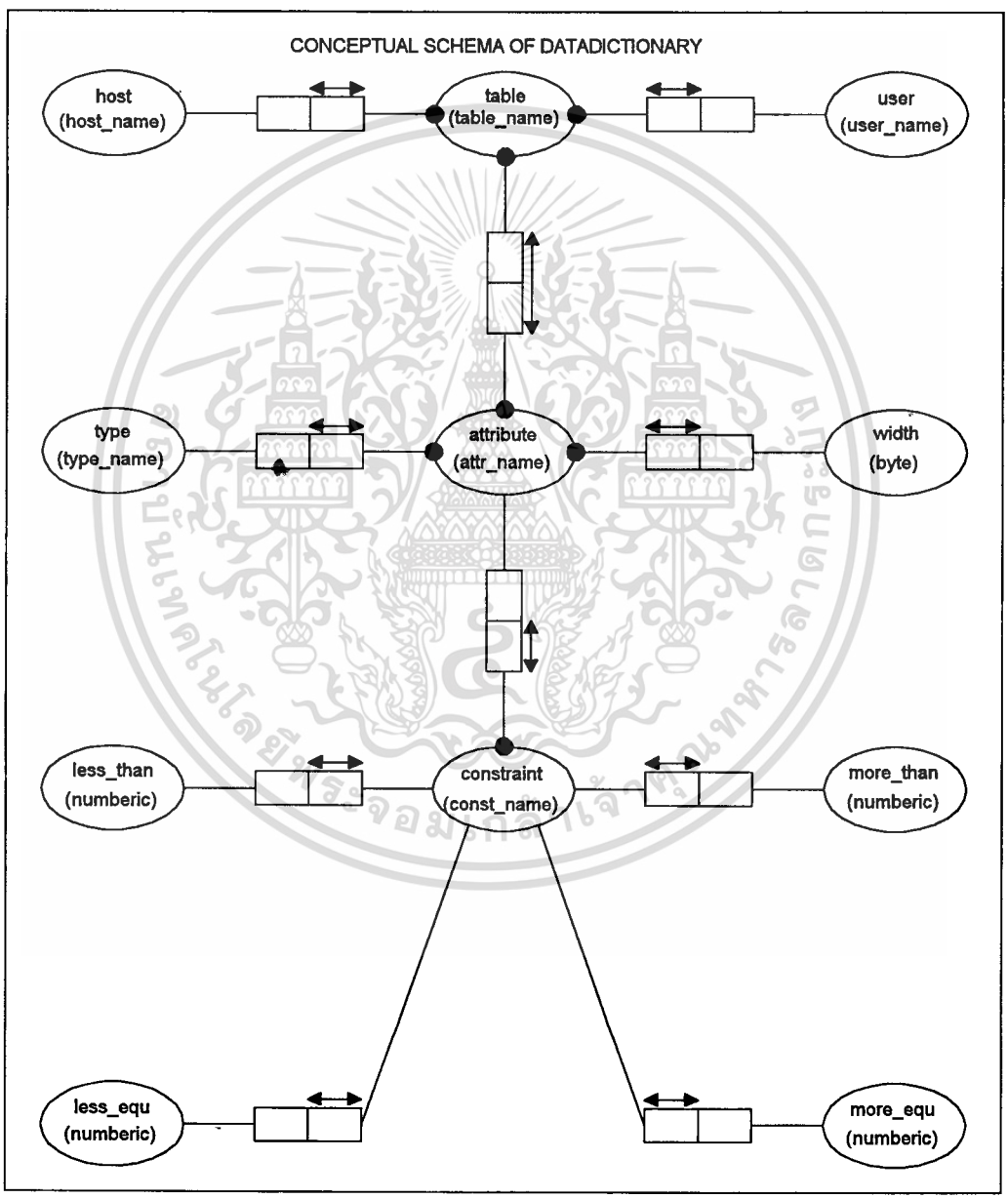
เครื่องคอมพิวเตอร์ A มีการเข้าถึงข้อมูลที่เก็บไว้ในเครื่องคอมพิวเตอร์ A และเครื่องคอมพิวเตอร์ B สำหรับผู้ใช้งาน user3 นั้น เรียกใช้โปรแกรมส่วนติดต่อผู้ใช้งานจากเครื่องคอมพิวเตอร์ B และคำถามที่ป้อนให้กับโปรแกรมมีการเข้าถึงข้อมูลที่อยู่บนเครื่องคอมพิวเตอร์ A และ B

4.2 ส่วนจัดการกระบวนการเริ่มต้น (Initialization Unit)

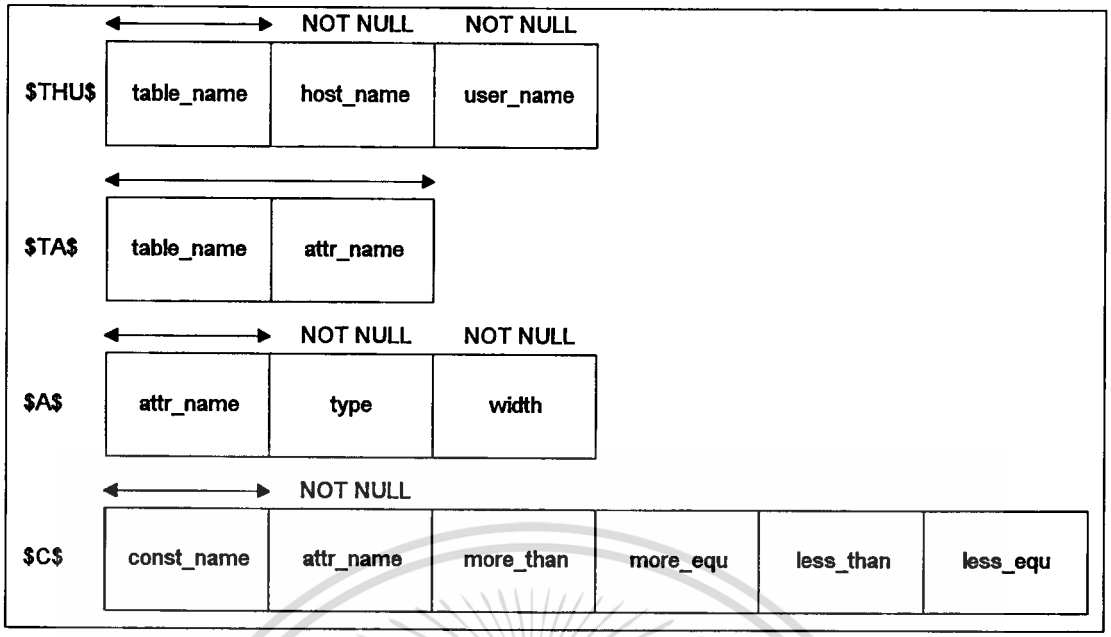
ส่วนจัดการกระบวนการเริ่มต้นนี้ จะทำหน้าที่ในการสร้างและปรับปรุงข้อมูลของพจนานุกรมฐานข้อมูล ของระบบฐานข้อมูลแบบกระจายที่ใช้สำหรับเครื่องคอมพิวเตอร์นั้นๆ และยังทำหน้าที่ในการจัดเก็บกฎบังคับความถูกต้องด้วย ดังนั้นในส่วนรายละเอียดจะแยกกล่าวตามการทำงานเป็นสองส่วน คือ ส่วนพจนานุกรมฐานข้อมูล และส่วนจัดเก็บกฎบังคับความถูกต้อง ดังต่อไปนี้

4.2.1 ส่วนพจนานุกรมฐานข้อมูล

โดยปรกติแล้ว ระบบฐานข้อมูลที่ทำงานได้โดยอิสระ จะมีพจนานุกรมฐานข้อมูลใช้อยู่แล้ว เพื่อเก็บข้อมูลที่เกี่ยวกับฐานข้อมูลนั้นๆ เมื่อระบบฐานข้อมูลถูกนำมาสร้างเป็นระบบฐานข้อมูลแบบกระจายจำเป็นจะต้องมีพจนานุกรมฐานข้อมูลสำหรับระบบฐานข้อมูลแบบกระจายด้วย สำหรับเก็บข้อมูลที่เกี่ยวกับระบบฐานข้อมูลแบบกระจาย เพื่อใช้สำหรับการประมวลผลในขั้นตอนต่างๆ ซึ่งส่วนพจนานุกรมนี้จะประกอบไปด้วยส่วนย่อยๆหลายส่วนด้วยกัน อธิบายได้ดังนี้ คือ



รูปที่ 4.3 แสดง Conceptual Schema ในรูปแบบ NIAM



รูปที่ 4.4 แสดงรูปแบบตารางของพจนานุกรมฐานข้อมูล DDQ/1

4.2.1.1 ส่วนสร้างตารางพจนานุกรมฐานข้อมูล

ตารางพจนานุกรมฐานข้อมูล นับเป็นส่วนสำคัญส่วนหนึ่งในการออกแบบ และสร้างระบบฐานข้อมูล ซึ่งการออกแบบตารางพจนานุกรมฐานข้อมูล จะมีผลถึงการออกแบบส่วนอื่นๆ ด้วย สำหรับตารางพจนานุกรมฐานข้อมูลของระบบ DDQ/1 ที่สร้างขึ้น ได้ออกแบบให้สามารถเก็บกฎบังคับความถูกต้องไว้ด้วย โดยการใช้รูปแบบการแสดง Conceptual Schema ในรูปของ NIAM (Nijssen Information Analysis Methodology) [21] เป็นบรรทัดฐานในการออกแบบ โดยใช้ข้อกำหนดว่า ในระบบฐานข้อมูลที่ทำงานได้โดยอิสระทั้งหมดนั้น จะไม่มีความสัมพันธ์(ตารางข้อมูล)ใดที่อยู่ในระบบฐานข้อมูลที่ทำงานได้โดยอิสระมากกว่าหนึ่งระบบ และชื่อของแอททริบิวที่เหมือนกันจะต้องเก็บข้อมูลที่มีโดเมนเดียวกันด้วย และหากมีกฎบังคับความถูกต้องที่ระบุถึงชื่อแอททริบิวใด กฎบังคับความถูกต้องนั้นก็จะมีผลต่อทุกๆแอททริบิวที่มีชื่อเหมือนกัน โดยกฎบังคับความถูกต้องจะเป็นชนิดบอกโดเมน (domain constraint) ด้วยการใช้ข้อกำหนดดังกล่าวสามารถที่จะสร้างรูปแบบของ NIAM ได้ดังแสดงในรูปที่ 4.3 และเปลี่ยนเป็นตารางสำหรับเก็บข้อมูลของพจนานุกรมฐานข้อมูลได้ ดังแสดงในรูปที่ 4.4 ซึ่งเครื่องคอมพิวเตอร์แต่ละเครื่องที่ประกอบขึ้นเป็นระบบฐานข้อมูลแบบกระจาย DDQ/1 จะต้องมีตารางพจนานุกรมฐานข้อมูลแบบกระจายเป็นของตัวเอง

สำหรับโปรแกรมในส่วนสร้างตารางพจนานุกรมฐานข้อมูล จะทำหน้าที่ในการสร้างตารางพจนานุกรมฐานข้อมูลท้องถิ่น และตารางพจนานุกรมฐานข้อมูลโดยรวม รวมทั้งสร้างตาราง

เก็บกฎบังคับความถูกต้องท้องถิ่นและตารางเก็บกฎบังคับความถูกต้องโดยรวมด้วย โดยโปรแกรมในส่วนนี้ผู้ดูแลระบบฐานข้อมูลท้องถิ่น จะเป็นผู้เรียกใช้ในขั้นตอนการติดตั้งระบบเพียงครั้งเดียวเท่านั้น หลังจากนั้นจะไม่มีการเรียกใช้โปรแกรมอีกเลย เว้นแต่ว่าจะทำการติดตั้งระบบใหม่

4.2.1.2 ส่วนป้อนข้อมูลพจนานุกรมฐานข้อมูลท้องถิ่น

พจนานุกรมฐานข้อมูลท้องถิ่น เป็นตารางข้อมูลที่สร้างขึ้นเพื่อเตรียมข้อมูล สำหรับสร้างพจนานุกรมฐานข้อมูลโดยรวมของระบบฐานข้อมูลแบบกระจาย DDQ/1 ซึ่งผู้ดูแลระบบฐานข้อมูลท้องถิ่นจะเป็นผู้ป้อนข้อมูล เข้าสู่พจนานุกรมฐานข้อมูลท้องถิ่น ด้วยการเรียกใช้โปรแกรมซึ่งโปรแกรมในส่วนนี้จะทำการค้นหาข้อมูล จากพจนานุกรมฐานข้อมูลของระบบฐานข้อมูลที่ทำงานได้โดยอิสระ แล้วทำการเพิ่มข้อมูลเข้าสู่พจนานุกรมฐานข้อมูลท้องถิ่น หลังจานนั้นส่วนป้อนข้อมูลพจนานุกรมฐานข้อมูลโดยรวม จะเป็นส่วนที่นำข้อมูลนี้ไปใช้ในการสร้างพจนานุกรมฐานข้อมูลโดยรวมต่อไป ซึ่งขั้นตอนในการสร้างพจนานุกรมฐานข้อมูลโดยรวม อธิบายไว้ในหัวข้อต่อไป

เนื่องจากพจนานุกรมฐานข้อมูลท้องถิ่นใช้เก็บข้อมูลที่จะนำไปสร้างพจนานุกรมฐานข้อมูลโดยรวม ดังนั้นในการออกแบบ Conceptual Schema ในรูปแบบของ NIAM สำหรับพจนานุกรมฐานข้อมูลท้องถิ่น จึงมีลักษณะเหมือนกับ Conceptual Schema ของพจนานุกรมฐานข้อมูลโดยรวม ดังนั้น Conceptual Schema ในรูปแบบของ NIAM จึงมีลักษณะเช่นเดียวกับในรูปที่ 4.3 และเปลี่ยนเป็นตารางพจนานุกรมฐานข้อมูลท้องถิ่น ได้เช่นเดียวกับตารางในรูปที่ 4.4 แต่เนื่องจากในฐานข้อมูลหนึ่งๆที่สร้างเป็นระบบฐานข้อมูลแบบกระจาย DDQ/1 จะต้องมีการสร้างพจนานุกรมฐานข้อมูล 2 ส่วน คือ พจนานุกรมฐานข้อมูลท้องถิ่น และพจนานุกรมฐานข้อมูลโดยรวม ซึ่งทั้งสองจะมีลักษณะเหมือนกัน ดังนั้นเพื่อให้เกิดความแตกต่าง ในการตั้งชื่อตารางพจนานุกรมฐานข้อมูลท้องถิ่น จึงได้ใช้ตัวอักษรภาษาอังกฤษ " L " นำหน้าชื่อตารางที่ได้จากรูปที่ 4.4 ซึ่งชื่อตารางทั้ง 4 จะเป็นดังนี้

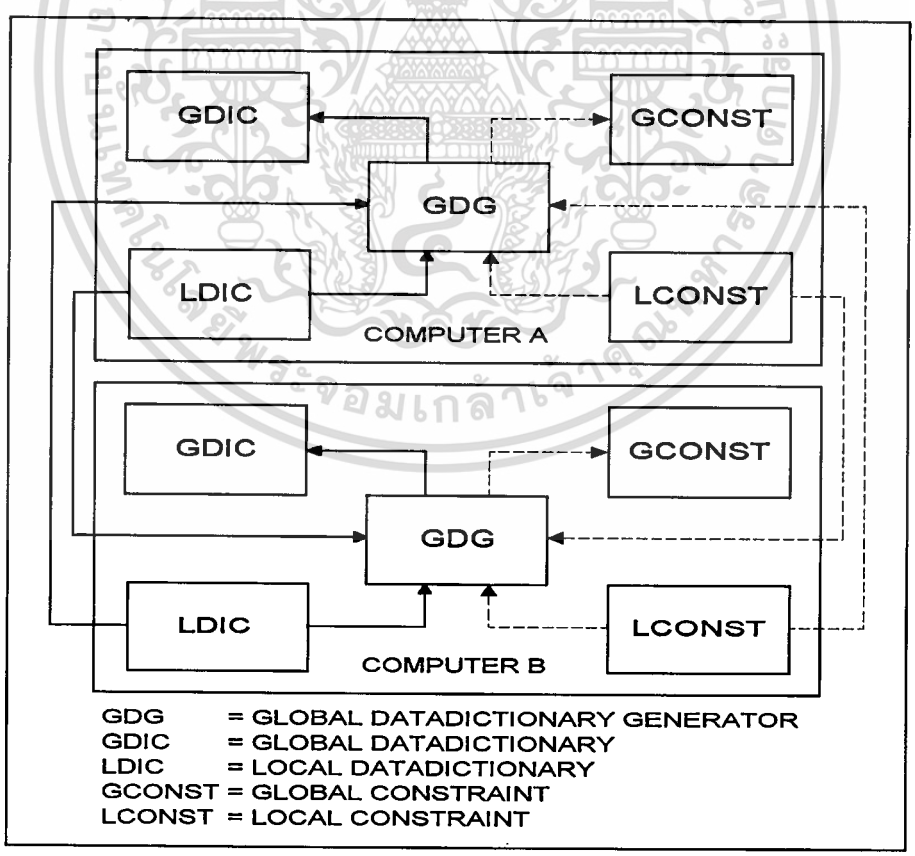
LTHU , LTA , LA , LC

4.2.1.3 ส่วนป้อนข้อมูลพจนานุกรมฐานข้อมูลโดยรวม

ในการสร้างพจนานุกรมฐานข้อมูลโดยรวมในแต่ละเครื่องนั้น จะกระทำได้โดยการนำข้อมูลในพจนานุกรมฐานข้อมูลท้องถิ่น จากคอมพิวเตอร์เครื่องอื่นๆ ที่ต้องการสร้างเป็นระบบฐานข้อมูลแบบกระจาย DDQ/1 มารวมกับพจนานุกรมฐานข้อมูลท้องถิ่นที่อยู่บนคอมพิวเตอร์เครื่อง

นั้นๆ โดยการส่งคำถามในรูปแบบต้นไม้เชิงพีชคณิตสัมพันธ์ ผ่านทางส่วนเชื่อมต่อโครงข่าย ไปยังคอมพิวเตอร์เครื่องที่ต้องการจะติดต่อด้วย และนำข้อมูลที่ได้ป้อนเข้าสู่พจนานุกรมฐานข้อมูลโดยรวมของคอมพิวเตอร์เครื่องนั้นๆ ในขั้นตอนการส่งคำถามรูปแบบต้นไม้เชิงพีชคณิตสัมพันธ์นั้น โปรแกรมในส่วนนี้ จะส่งคำถามที่เข้าถึงข้อมูลกฎบังคับความถูกต้องท้องถิ่นในคอมพิวเตอร์เครื่องที่กำลังติดต่ออยู่ด้วย และนำข้อมูลป้อนเข้าสู่ตารางข้อมูลกฎบังคับความถูกต้องโดยรวม เช่นเดียวกับข้อมูลพจนานุกรมฐานข้อมูล ดังแสดงตัวอย่างในรูปที่ 4.5 ซึ่งเป็นการสร้างพจนานุกรมฐานข้อมูลของระบบฐานข้อมูลแบบกระจาย DDQ/1ที่ประกอบด้วยระบบฐานข้อมูล 2 ระบบ

ในระบบปฏิบัติการยูนิกซ์ โดยปรกติแล้วจะมีไฟล์ชื่อ /etc/hosts อยู่ ซึ่งไฟล์นี้จะเก็บชื่อของเครื่องคอมพิวเตอร์ (host name) และที่อยู่ (address) ของคอมพิวเตอร์ที่เชื่อมต่อกันเป็นระบบโครงข่าย[19] ดังนั้นในการกำหนดเครื่องคอมพิวเตอร์ที่จะติดต่อด้วย จะกระทำได้ด้วยการดูรายชื่อเครื่องคอมพิวเตอร์ในไฟล์ /etc/hosts และทำการติดต่อทีละเครื่องจนกว่าจะครบทุกเครื่อง ในกรณีที่มีคอมพิวเตอร์เครื่องใดเครื่องหนึ่ง ที่ไม่ได้อยู่ในระบบฐานข้อมูล DDQ/1 แต่ถูกเชื่อมต่อไว้ในโครงข่ายเดียวกันนั้น ในการติดต่อเพื่อที่จะส่งคำถามนั้นจะไม่มีการตอบสนองกลับมา ทำให้ทราบได้ว่า คอมพิวเตอร์เครื่องนั้นไม่ได้อยู่ในระบบ DDQ/1



รูปที่ 4.5 แสดงตัวอย่างการสร้างพจนานุกรมฐานข้อมูล

4.2.2 ส่วนป้อนกฎบังคับความถูกต้อง

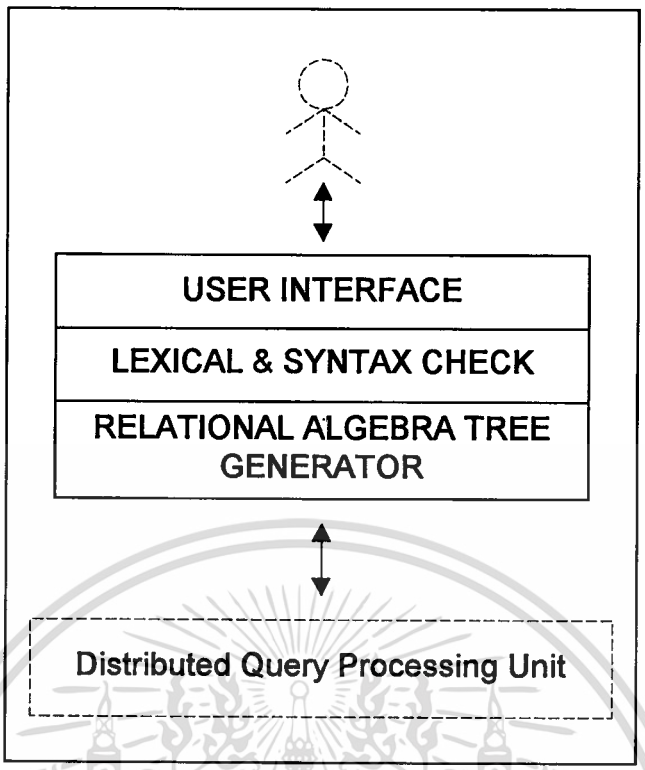
ส่วนป้อนกฎบังคับความถูกต้องนี้จะเป็นโปรแกรมที่ถูกเรียกใช้โดยผู้ดูแลระบบฐานข้อมูลท้องถิ่น เพื่อใช้ในการป้อนกฎบังคับความถูกต้องเข้าสู่ตารางเก็บกฎบังคับความถูกต้องท้องถิ่น สำหรับนำไปสร้างตารางเก็บกฎบังคับความถูกต้องโดยรวม ในการเรียกใช้โปรแกรมนั้น ผู้ดูแลระบบจะป้อนกฎบังคับความถูกต้องในรูปของประโยคคำสั่ง ที่มีไวยากรณ์ดังนี้คือ

```
command     := EXIT ; | statement
statement   := CREATE CONSTRAINT <constraint_name> ON
               ATTRIBUTE <attribute_name> WHERE predicate ;
               | DELETE CONSTRAINT <constraint_name> ;
predicate   := comparison | predicate conjunction comparison
comparison  := <attribute_name> comp <constraint>
conjunction := AND | OR
comp        := = | ^= | < | > | <= | >=
```

จากไวยากรณ์ที่กล่าวมาข้างต้น constraint_name ที่ผู้ดูแลระบบฐานข้อมูลท้องถิ่นใช้นั้น จะถูกกำหนดจากผู้ดูแลระบบฐานข้อมูลโดยรวม เพื่อไม่ให้มีการกำหนด constraint_name ซ้ำกันในแต่ละสถานที่ และแอททริบิวหนึ่งๆนั้นสามารถมีกฎบังคับความถูกต้องได้มากกว่าหนึ่งกฎ อย่างไรก็ตาม แอททริบิวอาจจะไม่มีการกำหนดกฎบังคับความถูกต้องก็ได้

4.3 ส่วนติดต่อผู้ใช้งาน

ส่วนติดต่อผู้ใช้งานนี้ เป็นโปรแกรมที่ถูกเรียกใช้จากผู้ใช้งาน เพื่อเข้าถึงข้อมูลที่ต้องการดังที่ได้อธิบายไว้ในหัวข้อที่ 4.1.2 แล้วนั้น ประกอบไปด้วยส่วนทำงาน 3 ส่วน คือ ส่วนติดต่อผู้ใช้ (user interface) ส่วนแยกคำและตรวจสอบไวยากรณ์ (lexical and syntax check) และส่วนสร้างคำถามรูปแบบต้นไม้เชิงพีชคณิตสัมพันธ์ (relational algebra tree generator) โดยส่วนติดต่อผู้ใช้จะรับคำถามเป็นภาษพีชคณิตสัมพันธ์ในรูปประโยคข้อความ เพื่อส่งให้กับส่วนแยกคำและตรวจสอบไวยากรณ์ ส่วนแยกคำและตรวจสอบไวยากรณ์จะรับคำถามในรูปประโยคข้อความแล้วทำการแยกคำ จากนั้นจะทำการตรวจสอบประโยคคำถามว่าถูกต้องตามหลักไวยากรณ์ภาษาพีชคณิตสัมพันธ์หรือไม่ หากคำถามไม่เป็นไปตามหลักไวยากรณ์ภาษาพีชคณิตสัมพันธ์ ส่วนแยกคำและตรวจสอบไวยากรณ์ก็จะส่งข้อความแสดงข้อผิดพลาดไปยังส่วนติดต่อผู้ใช้เพื่อแสดงผล และส่วนติดต่อผู้ใช้ก็จะกลับมาทำงานเพื่อรับคำถามใหม่ต่อไป



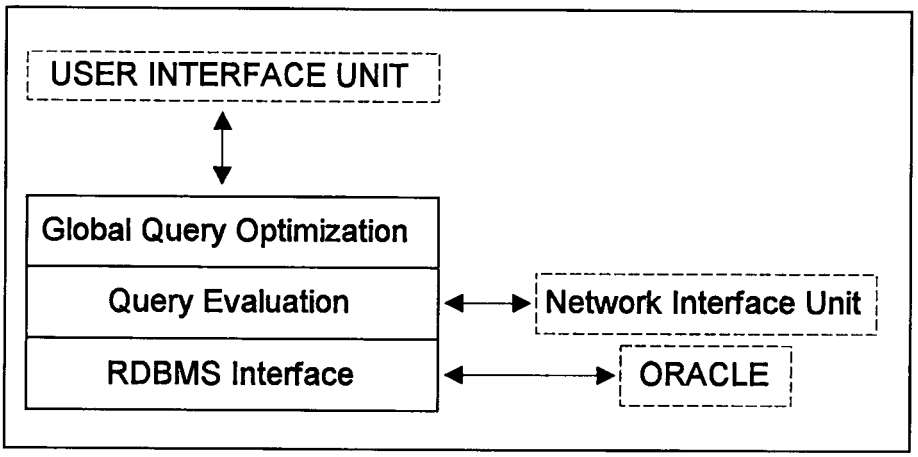
รูปที่ 4.6 แสดงโครงสร้างส่วนติดต่อผู้ใช้งาน

ในกรณีที่ประโยคคำถามถูกต้องตามหลักไวยากรณ์พีชคณิตสัมพันธ์ ส่วนแยกคำและตรวจสอบไวยากรณ์ก็จะส่งกลุ่มคำที่ได้แยกไว้ต่อไปยังส่วนสร้างคำถามรูปแบบต้นไม้เชิงพีชคณิตสัมพันธ์ เพื่อทำการสร้างคำถามรูปแบบต้นไม้เชิงพีชคณิตสัมพันธ์ จากนั้นส่วนสร้างคำถามรูปแบบต้นไม้เชิงพีชคณิตสัมพันธ์ จะส่งคำถามรูปแบบต้นไม้เชิงพีชคณิตสัมพันธ์ไปยังส่วนประมวลผลคำถามแบบกระจายต่อไป

หลังจากที่ส่วนประมวลผลคำถามแบบกระจายทำการประมวลผลเสร็จ ส่วนติดต่อผู้ใช้ก็จะนำข้อมูลไปแสดงผลต่อไป และพร้อมที่จะรับคำถามใหม่ต่อไป ซึ่งโครงสร้างของส่วนติดต่อผู้ใช้งานเป็นไปดังแสดงไว้ในรูปที่ 4.6

4.4 ส่วนประมวลผลคำถามแบบกระจาย

ส่วนประมวลผลคำถามแบบกระจาย ประกอบไปด้วยส่วนทำงาน 3 ส่วนด้วยกัน คือ ส่วนปรับปรุงและตรวจสอบคำถาม (Global Query Optimization) ส่วนประมวลผลคำถาม (Query Evaluation) และส่วนติดต่อระบบจัดการฐานข้อมูล (RDBMS Interface) ดังแสดงโครงสร้างไว้ในรูปที่ 4.7



รูปที่ 4.7 แสดงโครงสร้างส่วนประมวลผลคำถามแบบกระจาย

ในส่วนปรับปรุงและตรวจสอบคำถาม จะทำหน้าที่ในการปรับปรุงและตรวจสอบคำถาม เพื่อให้การประมวลผลคำถามเป็นไปในเวลาน้อยที่สุด และมีการรับ-ส่งข้อมูลน้อยที่สุดด้วย ซึ่งขั้นตอนการทำงานเป็นดังนี้ คือ เมื่อส่วนประมวลผลได้รับคำถามก็จะทำการตรวจสอบคำถามว่าการอ้างถึงความสัมพันธ์และแอททริบิวนั้น เป็นความสัมพันธ์และแอททริบิวที่มีอยู่ในระบบฐานข้อมูลหรือไม่ จากนั้นจะทำการตรวจสอบว่าคำถามที่ได้รับมีความขัดแย้งกับกฎบังคับความถูกต้องหรือไม่ สำหรับขั้นตอนทั้งสองที่กล่าวมานี้ หากคำถามไม่ผ่านขั้นตอนใดขั้นตอนหนึ่งย่อมแสดงว่าถ้ามีการประมวลผลคำถามต่อไปจะหาผลลัพธ์ไม่ได้ หรือผลลัพธ์เป็นศูนย์ทัปเปิล (tuple) นั่นเอง จากนั้นถ้าคำถามผ่านขั้นตอนทั้งสองได้ ก็จะทำการปรับปรุงคำถามด้วยหลักการที่ได้กล่าวไว้ในบทที่ 3 เพื่อให้คำถามมีขั้นตอนการประมวลผลน้อยที่สุด คำถามที่ผ่านการปรับปรุงแล้วจะถูกส่งต่อไปยังส่วนประมวลผลคำถามต่อไป

ส่วนประมวลผลคำถามเมื่อได้รับคำถามในรูปแบบต้นไม้เชิงพีชคณิตสัมพันธ์ จากส่วนปรับปรุงและตรวจสอบคำถาม ก็จะทำการประมวลผลเพื่อให้ได้ข้อมูลที่ต้องการ โดยผ่านส่วนติดต่อระบบจัดการฐานข้อมูล ในการประมวลผลคำถามนี้หากมีตัวปฏิบัติการพีชคณิตสัมพันธ์ใดกระทำการปฏิบัติการกับความสัมพันธ์ที่อยู่ในสถานที่อื่น ส่วนประมวลผลคำถามนี้จะทำการวิเคราะห์ว่า การประมวลผลจะกระทำต่อที่สถานที่ใดจึงจะมีการรับ-ส่งข้อมูลน้อยที่สุด ทั้งนี้ผลการวิเคราะห์จะขึ้นอยู่กับตัวปฏิบัติการด้วย ว่าเป็นตัวปฏิบัติการชนิดใด จากนั้นจึงจะติดต่อกับระบบจัดการฐานข้อมูลที่เก็บความสัมพันธ์ที่ต้องการใช้ในการปฏิบัติการ โดยการติดต่อผ่านส่วนเชื่อมต่อโครงข่าย ถ้าผลการวิเคราะห์สรุปว่าต้องทำการประมวลผลต่อที่สถานที่เดิม ส่วนประมวลผลคำถามก็จะส่งคำถามไปยังสถานที่ที่เก็บความสัมพันธ์นั้น และรับข้อมูลเพื่อทำการประมวลผลต่อไปจนเสร็จ ผลที่ได้จะส่งกลับไปยังส่วนติดต่อผู้ใช้งานเพื่อแสดงผลต่อไป

แต่หากผลการวิเคราะห์สรุปว่า ควรประมวลผลต่อในสถานที่อื่นที่เก็บความสัมพันธ์ไว้ ส่วนประมวลผลคำถามก็จะส่งข้อมูล พร้อมทั้งรายละเอียด (information) ไปทำการประมวลผลต่อยังสถานที่ที่เก็บความสัมพันธ์ที่ต้องการใช้ในการประมวลผล จากนั้นจะรับข้อมูลผลลัพธ์ ไปทำการประมวลผลต่อจนได้ผลลัพธ์สุดท้าย แล้วส่งกลับให้กับส่วนที่ติดต่อมา

สำหรับส่วนติดต่อระบบจัดการฐานข้อมูล จะทำการแปลงคำถามในรูปแบบต้นไม้รูปเชิงพีชคณิตสัมพันธ์ ให้เป็นคำถามในรูปแบบที่ระบบจัดการฐานข้อมูลเข้าใจได้ ซึ่งในที่นี้ใช้ระบบจัดการฐานข้อมูลออราเคิล ดังนั้นส่วนติดต่อระบบจัดการฐานข้อมูลจึงแปลงคำถามในรูปแบบต้นไม้เชิงพีชคณิตสัมพันธ์ ให้เป็นคำถามในรูปแบบภาษาสอบถามเชิงโครงสร้าง (SQL) จากนั้นจะรอจนได้รับข้อมูลคำตอบ และส่งข้อมูลนั้นให้กับส่วนประมวลผลคำถาม

4.5 ส่วนเชื่อมต่อโครงข่าย

ส่วนเชื่อมต่อโครงข่ายจะทำหน้าที่รับ-ส่งข้อมูล ระหว่างส่วนประมวลผลคำถามแบบกระจาย หรือส่วนจัดการกระบวนการเริ่มต้น ไปยังส่วนเชื่อมต่อโครงข่ายในคอมพิวเตอร์เครื่องอื่นๆ โดยกระบวนการเชื่อมต่อโครงข่ายจะติดต่อกับกระบวนการอื่น ด้วยวิธีการเชื่อมต่อที่เรียกว่า "ซอคเกต" (socket interface) ซึ่งการเชื่อมต่อแบบนี้สามารถใช้กับโปรโตคอลได้หลายชนิด[18] เช่น โดเมนยูนิกซ์ (Unix domain) โดเมนอินเทอร์เนท (Internet domain หรือ TCP/IP) และโปรโตคอลอื่นๆอีกหลายชนิด การเชื่อมต่อแบบซอคเกตนี้ สามารถที่จะติดต่อได้จากกระบวนการที่อยู่ภายในคอมพิวเตอร์เครื่องเดียวกัน หรือจากกระบวนการที่อยู่ในคอมพิวเตอร์เครื่องอื่นๆได้ ทั้งนี้จะต้องเลือกใช้โปรโตคอลให้เหมาะสมด้วย

สำหรับการรับ-ส่งข้อมูลระหว่างกระบวนการที่อยู่ภายในคอมพิวเตอร์เครื่องเดียวกัน จะใช้การเชื่อมต่อแบบซอคเกตด้วยโปรโตคอลโดเมนยูนิกซ์ ส่วนการรับ-ส่งข้อมูลระหว่างกระบวนการที่อยู่บนคอมพิวเตอร์ต่างเครื่องกัน จะใช้การเชื่อมต่อแบบซอคเกตด้วยโปรโตคอล TCP/IP

บทที่ 5

การพัฒนาระบบฐานข้อมูลแบบกระจาย DDQ/1

โดยปรกติในการพัฒนาโปรแกรมนั้น จำเป็นจะต้องมีเครื่องมือต่างๆที่ใช้ในการพัฒนาเพื่อให้โปรแกรมทำงานตามที่เราต้องการ และบางครั้งอาจทำให้การพัฒนาโปรแกรมเป็นไปด้วยความรวดเร็วด้วย สำหรับในการพัฒนาระบบ DDQ/1 นี้ก็เช่นเดียวกัน จำเป็นจะต้องใช้เครื่องมือหลายชนิดในการพัฒนาโปรแกรม ทั้งฮาร์ดแวร์และซอฟท์แวร์ ดังนั้นในบทนี้จะขอกล่าวถึงเครื่องมือต่างๆที่จำเป็นต้องใช้ในการพัฒนาระบบ รวมถึงหน่วยโปรแกรมที่พัฒนาขึ้นมาด้วย ซึ่งจะได้กล่าวเป็นต่อไป

5.1 ฮาร์ดแวร์และซอฟท์แวร์ที่ใช้

สำหรับเครื่องมือที่ใช้ในการพัฒนาระบบ DDQ/1 มีหลายอย่างด้วยกัน ได้แก่

เครื่องคอมพิวเตอร์พร้อมระบบปฏิบัติการยูนิกซ์ [2],[18],[19]
ซอฟท์แวร์ NSP [23]
ระบบจัดการฐานข้อมูล ORACLE [3],[24],[25]
ตัวแปลภาษาขั้นต้น Pro*C [24]

ซึ่งเครื่องมือทั้ง 4 ชนิดที่ได้กล่าวมานี้ มีลักษณะการใช้ต่างกันคือ

เครื่องคอมพิวเตอร์เป็นฮาร์ดแวร์ที่ใช้ในการประมวลผลของโปรแกรมที่พัฒนาขึ้น และยังเป็นที่ประมวลผลของเครื่องมือต่างๆด้วย ในการพัฒนาโปรแกรมครั้งนี้ได้พัฒนาบนเครื่องมินิคอมพิวเตอร์ Targon M30 และ Targon M5 ส่วนระบบปฏิบัติการยูนิกซ์ ที่ใช้เป็นของบริษัท AT&T รุ่น System V

ซอฟท์แวร์ NSP (Networking Software Package) เป็นซอฟท์แวร์ที่ใช้ในการเชื่อมต่อคอมพิวเตอร์ ให้สามารถสื่อสารกันได้เป็นโครงข่าย โดยการสื่อสารนี้จะใช้โปรโตคอล TCP/IP

ระบบจัดการฐานข้อมูล ORACLE จะเป็นส่วนที่ทำหน้าที่ในการเก็บข้อมูลต่างๆ ทั้งข้อมูลของระบบฐานข้อมูล และข้อมูลของโปรแกรมด้วย ซึ่งระบบจัดการฐานข้อมูลนี้ จะเป็นระบบจัดการฐานข้อมูลแบบสัมพันธ์

ตัวแปลภาษาขั้นต้น (Precompiler) Pro*C เป็นซอฟท์แวร์ที่ใช้ในการแปลตัวโปรแกรมที่เขียนด้วยภาษา C และมีการฝังคำสั่งภาษาสอบถามเชิงโครงสร้าง (SQL) ทั้งนี้เนื่องจากเขียน

โปรแกรมติดต่อกับระบบจัดการฐานข้อมูล ORACLE จะใช้ภาษาสอบถามเชิงโครงสร้าง แต่ตัวแปลภาษา (compiler) จะแปลภาษา C เป็นภาษาเครื่องเท่านั้น ดังนั้นจึงต้องใช้ตัวแปลภาษาขั้นต้นทำการแปลโปรแกรมที่เขียนด้วยภาษา C ฝังด้วยคำสั่งภาษาสอบถามเชิงโครงสร้าง ให้เป็นโปรแกรมภาษา C ล้วนๆ ด้วยซอฟท์แวร์ Pro*C

5.2 การฝังคำสั่งภาษาสอบถามเชิงโครงสร้างในโปรแกรม (Embedded SQL)

ในการฝังคำสั่งภาษาสอบถามเชิงโครงสร้างในโปรแกรมที่เขียนด้วยภาษา C นั้น มีด้วยกัน 2 วิธี คือ การฝังคำสั่งภาษาสอบถามเชิงโครงสร้างแบบสถิตย์ และการฝังคำสั่งภาษาสอบถามเชิงโครงสร้างแบบจลน์ ซึ่งทั้งสองวิธีมีลักษณะการใช้ต่างกัน ดังจะได้กล่าวในรายละเอียดต่อไป แต่ก่อนอื่น จะขอกล่าวถึงความสัมพันธ์ระหว่างภาษาพีชคณิตสัมพันธ์กับภาษาสอบถามเชิงโครงสร้างก่อน เพื่อจะได้ทราบว่าคำถามภาษาพีชคณิตสัมพันธ์ จะแปลงเป็นภาษาสอบถามเชิงโครงสร้างได้อย่างไร เพื่อที่จะได้นำคำสั่งภาษาสอบถามเชิงโครงสร้าง ไปฝังไว้ในโปรแกรมต่อไป ซึ่งมีรายละเอียดดังนี้ คือ

5.2.1 ความสัมพันธ์ระหว่างพีชคณิตสัมพันธ์กับภาษาสอบถามเชิงโครงสร้าง

ที่ต้องกล่าวถึงความสัมพันธ์ของพีชคณิตสัมพันธ์กับภาษาสอบถามเชิงโครงสร้าง ประการแรกเนื่องมาจาก ภาษาพีชคณิตสัมพันธ์เป็นพื้นฐานของภาษายุคที่ 4 [4] และประการที่สองในการวิจัยครั้งนี้ได้ใช้ภาษาพีชคณิตสัมพันธ์ในการรับคำถาม โดยใช้ระบบจัดการฐานข้อมูลที่รับคำถามด้วยภาษาสอบถามเชิงโครงสร้าง ดังนั้นในหัวข้อนี้จะได้กล่าวถึงความสัมพันธ์ของตัวปฏิบัติการพีชคณิตสัมพันธ์ กับตัวปฏิบัติการของภาษาสอบถามเชิงโครงสร้าง

5.2.1.1 UNION

สำหรับตัวปฏิบัติการ UNION ในภาษาพีชคณิตสัมพันธ์นั้น จะมีใช้ในภาษาสอบถามเชิงโครงสร้างด้วย ซึ่งลักษณะการใช้จะคล้ายคลึงกัน เช่น

A UNION B

จะเขียนเป็นภาษาสอบถามเชิงโครงสร้างได้ดังนี้

```
SELECT * FROM A
UNION
SELECT * FROM B
```

5.2.1.2 DIFFERENCE

ในภาษาสอบถามเชิงโครงสร้างจะมีตัวปฏิบัติการ MINUS ที่ทำหน้าที่เหมือนกับตัวปฏิบัติการ MINUS ในภาษาพีชคณิตสัมพันธ์ ซึ่งแสดงการใช้ได้ดังนี้ เช่น ในภาษาพีชคณิตสัมพันธ์เขียนเป็น

A MINUS B

ภาษาสอบถามเชิงโครงสร้างจะเขียนได้เป็น

```
SELECT * FROM A
MINUS
SELECT * FROM B
```

5.2.1.3 INTERSECTION

ตัวปฏิบัติการนี้ก็เช่นเดียวกันกับตัวปฏิบัติ UNION และ MINUS คือจะมีใช้ในภาษาสอบถามเชิงโครงสร้าง ซึ่งแสดงตัวอย่างการใช้งานในภาษาพีชคณิตสัมพันธ์ และภาษาสอบถามเชิงโครงสร้างได้ดังนี้

A INTERSECT B

เขียนเป็นภาษาสอบถามเชิงโครงสร้างจะได้

```
SELECT * FROM A
INTERSECT
SELECT * FROM B
```

5.2.1.4 EXTENDED CARTESIAN PRODUCT

สำหรับตัวปฏิบัติการนี้ สามารถที่จะกระทำได้ในภาษาสอบถามเชิงโครงสร้างเช่นกัน แต่จะมีวิธีการเขียนที่แตกต่างออกไปดังนี้ เช่นภาษาพีชคณิตสัมพันธ์เขียนเป็น

A TIMES B

จะเขียนเป็นภาษาสอบถามเชิงโครงสร้างได้เป็น

```
SELECT * FROM A , B
```

5.2.1.5 SELECTION

ตัวปฏิบัติการ SELECT ในภาษาพีชคณิตสัมพันธ์ จะมีความคล้ายคลึงกับตัวปฏิบัติการ SELECT ในภาษาสอบถามเชิงโครงสร้าง แต่จะไม่เหมือนกันเลยทีเดียว ดังตัวอย่าง ภาษาพีชคณิตสัมพันธ์เขียนเป็น

R WHERE R . X = c

จะเขียนเป็นภาษาสอบถามเชิงโครงสร้างได้ดังนี้

SELECT * FROM R

WHERE R . X = c

5.2.1.6 PROJECTION

สำหรับตัวปฏิบัติการนี้ สามารถที่จะปฏิบัติการในภาษาสอบถามเชิงโครงสร้างได้เช่นกัน ดังตัวอย่างเช่น ภาษาพีชคณิตสัมพันธ์ เขียนเป็น

R [R . X]

ภาษาสอบถามเชิงโครงสร้างจะเขียนได้เป็น

SELECT R . X FROM R

5.2.1.7 JOIN

จากหัวข้อที่ 2.4.3 เราทราบมาว่า ตัวปฏิบัติการ JOIN เป็นตัวปฏิบัติการที่ไม่แท้ คือ สามารถที่จะเขียนอยู่ในรูปของตัวปฏิบัติการ CARTESIAN PRODUCT และตัวปฏิบัติการ SELECTION ได้ ดังนั้น ในภาษาสอบถามเชิงโครงสร้างก็เช่นกัน อย่างเช่น ภาษาพีชคณิตสัมพันธ์ เขียนเป็น

A JOIN B

จะเขียนเป็นภาษาสอบถามเชิงโครงสร้างได้ดังนี้

SELECT * FROM A , B

WHERE A.a_i = B.b_x

...

AND A.a_k = B.b_y

เมื่อ a_i ... a_k และ b_x ... b_y เป็น attribute ที่มี unqualified name เหมือนกัน

5.2.1.8 DIVISION

สำหรับตัวปฏิบัติการนี้ สามารถที่จะเขียนเป็นภาษาสอบถามเชิงโครงสร้างได้เช่นกัน ตัวอย่างเช่น ภาษาพีชคณิตสัมพันธ์ เขียนเป็น

A DIVIDEBY B

จะเขียนเป็นภาษาสอบถามเชิงโครงสร้างได้เป็น

SELECT A.a_1 ... A.a_m FROM A , B

WHERE A.a_{m+1} = B.b_1

...

AND A.a_{m+n} = B.b_n

GROUP BY A.a_1 ... A.a_m

HAVING COUNT (*) = (SELECT COUNT (*) FROM B)

เมื่อ a_1 ... a_{m+n} เป็น attribute ของความสัมพันธ์ A , b_1 ... b_n เป็น attribute ของความสัมพันธ์ B และทั้งสองความสัมพันธ์มี attribute ที่ร่วมกันคือ a_{m+1} ... a_{m+n} และ b_1 ... b_n

สำหรับตัวอย่างที่กล่าวมาข้างต้นนั้น เป็นเพียงตัวอย่างแบบหนึ่งที่ผู้วิจัยเลือกใช้ ทั้งนี้ได้อธิบายเพื่อเป็นแนวทางในการทำความเข้าใจเท่านั้น

5.2.2 การฝังคำสั่งภาษาสอบถามเชิงโครงสร้างแบบสถิตย์

การฝังคำสั่งภาษาสอบถามเชิงโครงสร้างแบบสถิตย์ เป็นการฝังคำสั่งภาษาสอบถามเชิงโครงสร้างไว้ในโปรแกรมทำงาน เพื่อที่จะติดต่อกับระบบจัดการฐานข้อมูลออราเคิล โดยโปรแกรมจะทำงานตามที่ได้ฝังคำสั่งภาษาสอบถามเชิงโครงสร้างไว้ และจะไม่สามารถเปลี่ยนแปลงคำถามที่จะเข้าถึงข้อมูลในขณะที่เรียกใช้โปรแกรมได้ ดังนั้นผู้เขียนโปรแกรมจะต้องเตรียมคำถามไว้ในขั้นตอนของการฝังคำสั่ง ทั้งนี้เพื่อจะได้กำหนดตัวแปรในภาษา C ได้ถูกต้องตรงกับชนิดของข้อมูล (data type) ที่ต้องการเข้าถึง รวมทั้งผู้เขียนโปรแกรมจะต้องทราบถึงผลของการฝังคำสั่งภาษาสอบถามเชิงโครงสร้างด้วยว่า จะได้ผลลัพธ์มากกว่า 1 ทับเปิลหรือไม่ เพื่อจะได้กำหนดขั้นตอนในการฝังคำสั่งภาษาสอบถามเชิงโครงสร้างให้สอดคล้องกับผลลัพธ์ที่จะเกิดขึ้น ดังตัวอย่างต่อไปนี้ ซึ่งจะได้กล่าวรายละเอียดต่อไป

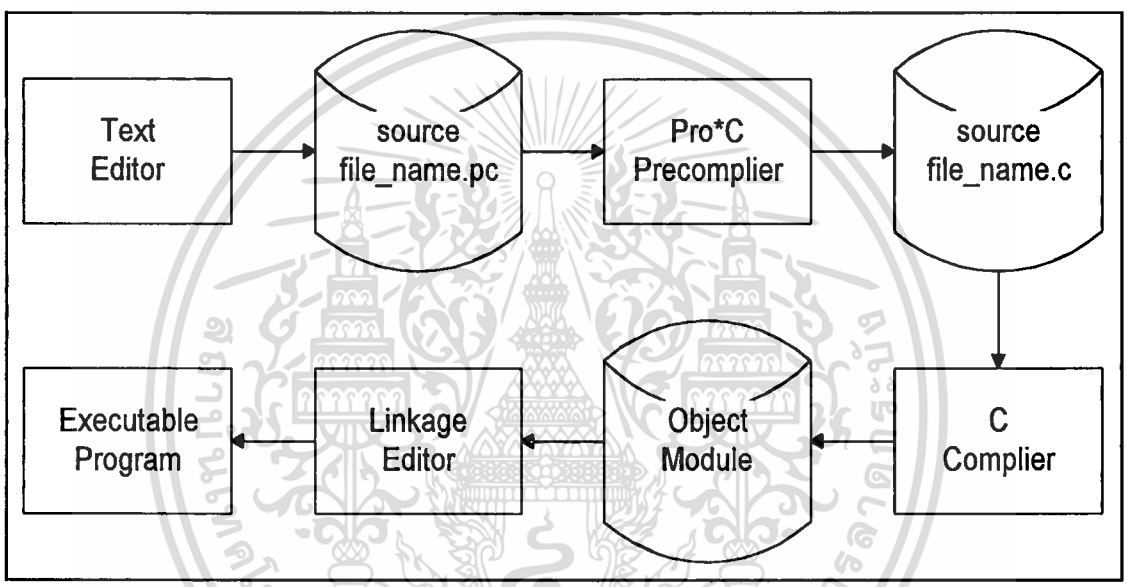
```
EXEC SQL BEGIN DECLARE SECTION ;
        VARCHAR     uid[20] ;
        VARCHAR     pwd[20] ;
        float       sal , comm ;
        char        ename[11] ;
EXEC SQL END DECLARE SECTION ;
EXEC SQL INCLUDE SQLCA ;
```

```
main( )
{
        /* log into ORACLE */
        strcpy(uid.arr,"SCOTT") ;                    /* copy the user name */
        uid.len = strlen(uid.arr) ;
        strcpy(pwd.arr,"TIGER") ;                    /* copy the password */
        pwd.len = strlen(pwd.arr) ;
        EXEC SQL WHENEVER SQLERROR STOP ;
        EXEC SQL CONNECT :uid IDENTIFIED BY :pwd ;
        printf("Connectde to ORACLE user: %s \n",uid.arr) ;
        EXEC SQL DECLARE C1 CURSOR FOR
                SELECT ENAME , SAL , COMM
                FROM EMP WHERE JOB='SALESMAN' ;
        EXEC SQL OPEN C1 ;
        EXEC SQL WHENEVER NOT FOUND STOP ;
        printf("SALESMAN NAME\t\tSALARY\t\tCOMMISSION\n\n") ;
        for( ; ; )
        {
                EXEC SQL FETCH C1 INTO :ename, :sal, :comm ;
                printf("%-10s\t\t%6.2f\t\t6.2f \n",ename,sal,comm) ;
        }
        EXEC SQL CLOSE C1 ;
        EXEC SQL WHENEVER SQLERROR CONTINUE ;    /* don't trap errors */
        EXEC SQL COMMIT WORK RELEASE ;           /* log off database */
        exit (0) ;
}
```

การฝังประโยคคำสั่งภาษาสอบถามเชิงโครงสร้างในโปรแกรมภาษา C นั้น ประโยคคำสั่งภาษาสอบถามเชิงโครงสร้างจะต้องนำหน้าด้วยข้อความ "EXEC SQL" ทั้งนี้เพื่อให้เกิดความแตกต่างระหว่างประโยคคำสั่งภาษาสอบถามเชิงโครงสร้าง และประโยคคำสั่งภาษา C ตัวแปลภาษาขั้นต้น Pro*C จะทำการเลือกแปลเฉพาะประโยคที่ขึ้นต้นด้วยข้อความ "EXEC SQL" ให้เป็น

ประโยคคำสั่งภาษา C จากนั้นโปรแกรมที่ได้จะมีขั้นตอนการพัฒนาเหมือนโปรแกรมที่เขียนด้วยภาษา C ปรกติทั่วไป ดังแสดงไว้ในรูปที่ 5.1 ซึ่งสรุปขั้นตอนในการพัฒนาโปรแกรมที่มีการฝังประโยคคำสั่งภาษาสอบถามเชิงโครงสร้าง ได้ดังนี้

1. เขียนโปรแกรมที่มีการฝังประโยคคำสั่งภาษาสอบถามเชิงโครงสร้าง
2. ทำการแปลโปรแกรมขั้นต้นด้วย ตัวแปลภาษาขั้นต้น Pro*C
3. ทำการแปลโปรแกรมภาษา C ด้วยตัวแปลภาษา C
4. ทำการลิงค์แฟ้มข้อมูลที่เกี่ยวข้องทั้งหมดให้เป็นโปรแกรมประมวลผล
5. เรียกใช้โปรแกรม



รูปที่ 5.1 แสดงขั้นตอนการพัฒนาโปรแกรมที่มีการฝังคำสั่งสอบถามภาษาโครงสร้าง

5.2.2.1 ชนิดของประโยคคำสั่งภาษาสอบถามเชิงโครงสร้าง

ประโยคของคำสั่งภาษาสอบถามเชิงโครงสร้างที่ฝังในโปรแกรมนั้นสามารถที่จะแบ่งออกได้ 2 ประเภท คือ

5.2.2.1.1. ประโยคคำสั่งปฏิบัติการ (Execute SQL statements)

ประโยคคำสั่งปฏิบัติการ จะเป็นการฝังคำสั่งที่เกี่ยวกับการจัดการข้อมูล (Data Mainpulation Language), การกำหนดข้อมูล (Data Definition Language) และการควบคุมข้อมูล (Data Control Language) ซึ่งการปฏิบัติการของประโยคคำสั่งเหล่านี้ จะมีผลต่อส่วนสื่อสารกับระบบจัดการฐานข้อมูล (SQL Communication Area) หรือเรียกอีกอย่างหนึ่งว่าเอสคิวแอลซีเอ

(SQLCA) โดยจะมีการแสดงรหัสสถานะ ของการประมวลผลประโยคคำสั่งการปฏิบัติการที่ส่วนนี้ ว่าปฏิบัติการสำเร็จหรือไม่อย่างไร นอกจากนี้ยังมีผลต่อการเริ่มต้นหรือการสิ้นสุด ของส่วนการเก็บการเรียงลำดับของประโยคคำสั่งภาษาสอบถามเชิงโครงสร้างด้วย (Logical Unit of Work)

5.2.2.1.2. ประโยคคำสั่งการกำหนด (Declarative SQL statements)

ประโยคคำสั่งการกำหนด คือ ประโยคคำสั่งที่จะไม่มีการแสดงรหัสสถานะของการปฏิบัติการ ในส่วนการสื่อสารกับระบบจัดการฐานข้อมูล และส่วนการเก็บการเรียงลำดับของประโยคคำสั่งภาษาสอบถามเชิงโครงสร้าง แต่จะใช้ในการกำหนดหรือบอกค่าตัวแปรต่างๆ ให้กับตัวแปลภาษาขั้นต้น Pro*C เพื่อให้ตัวแปลภาษาขั้นต้นทำงานได้อย่างถูกต้อง

5.2.2.2 ส่วนประกอบของโปรแกรม (Part of the Pro*C Program)

โปรแกรมประกอบด้วย ส่วนประกอบใหญ่ๆ 2 ส่วนคือ

ส่วนเริ่มต้นการประยุกต์ (Application Prologue) คือส่วนกำหนดตัวแปรที่ใช้ในการติดต่อกับฐานข้อมูล, การเตรียมพร้อม และการเริ่มต้นเพื่อที่จะติดต่อกับระบบจัดการฐานข้อมูล

ส่วนตัวโปรแกรมประยุกต์ (Application Body) คือส่วนของโปรแกรมที่มีการฝังประโยคคำสั่งภาษาสอบถามเชิงโครงสร้าง ในการเข้าถึงข้อมูลในฐานข้อมูล

ซึ่งทั้งสองส่วนที่กล่าวมาข้างต้น ยังประกอบไปด้วยส่วนต่างๆอีกหลายส่วนดังนี้ คือ

5.2.2.2.1 ส่วนเริ่มต้นการประยุกต์ (Application Prologue)

ประกอบไปด้วยส่วนต่างๆ 3 ส่วนคือ

5.2.2.2.1.1 ส่วนกำหนดตัวแปร (Declare Section)

คือส่วนที่ใช้กำหนดตัวแปรและชนิดของตัวแปรในภาษาซี ที่จะใช้ในการติดต่อกับระบบจัดการฐานข้อมูลออราเคิล มีรูปแบบในการใช้โดยเริ่มต้นด้วยประโยค

EXEC SQL BEGIN DECLARE SECTION ;

และจบด้วยประโยค

EXEC SQL END DECLARE SECTION ;

ระหว่างประโยคทั้ง 2 จะเป็นการกำหนดตัวแปรและชนิดของตัวแปร ซึ่งตัวแปรที่กำหนดในส่วนนี้สามารถแบ่งได้เป็น ตัวแปรหลัก (Host Variable) และตัวแปรบ่งชี้ (Indicator Variable) ตัวแปรหลักจะเป็นตัวแปรในภาษาซีที่ใช้เก็บค่าข้อมูลต่างๆ ในการติดต่อกับระบบจัดการฐานข้อมูล ส่วน

ตัวแปรบ่งชี้จะใช้ในการเพิ่มข้อมูลในคอลัมน์ที่ไม่ทราบค่า (NULL) และสามารถแสดงให้ทราบว่าข้อมูลที่ตัวแปรหลักได้รับจากฐานข้อมูล มีลักษณะอย่างไรเช่น ไม่ทราบค่า หรือ มีการตัดข้อมูลบางส่วนทิ้งเป็นต้น

สำหรับชนิดของตัวแปรหลักนั้น จะเหมือนกับชนิดของตัวแปรในภาษาซี แต่ที่เพิ่มขึ้นมาคือ ประเภท วาชาร์ (VARCHAR) ซึ่งมีลักษณะเป็นแบบเรคอร์ด (Record)

สำหรับข้อผิดพลาด (Error) ที่เกิดขึ้นในส่วนการกำหนดตัวแปรนี้ในกรณีที่ผู้ใช้มิได้กำหนด ตัวแปรที่ใช้ในโปรแกรมแต่มีการเรียกใช้จะมีการแสดงข้อความผิดพลาดคือ

Undeclared host variable <a> at line <b> in file <c>

5.2.2.2.1.2 ส่วนรวมส่วนสื่อสารกับระบบจัดการฐานข้อมูล (INCLUDE SQLCA)

ส่วนนี้จะมีประโยคคำสั่งเป็นดังนี้

EXEC SQL INCLUDE SQLCA ;

ประโยคคำสั่งนี้จะเป็นการกำหนดให้ตัวแปลภาษา Pro*C รวมส่วนของการสื่อสารกับระบบจัดการฐานข้อมูลเข้าไว้ในโปรแกรม โดยขณะที่ทำการพรีคอมไพล์ ออราเคิลจะทำหน้าที่เปลี่ยนหรือแทนที่ตัวแปรหลักในโปรแกรม ด้วยตัวแปรที่ได้จากแฟ้มข้อมูลที่นำมารวม และหน้าที่สำคัญของส่วนสื่อสารกับระบบจัดการฐานข้อมูลอีกอย่างหนึ่ง นอกเหนือจากการติดต่อกับออราเคิลก็คือ การแสดงข้อผิดพลาด (Error) และข้อระวังต่างๆ (Warning) ที่เกิดขึ้นในการปฏิบัติคำสั่งภาษาสอบถามเชิงโครงสร้างโดยจะแสดงด้วยค่าของตัวแปร

sqlca.sqlcode :
มากกว่า 0 จะแสดงถึงการกระทำคำสั่ง
เท่ากับ 0 แสดงว่าทำคำสั่งได้สมบูรณ์
น้อยกว่า 0 แสดงว่าเกิดการผิดพลาดขึ้น

sqlca.sqlwarn : จะประกอบด้วยอาเรย์ (array) ของแฟลกซ์ (flags) 8 ตัวซึ่งแต่ละตัวก็แสดงถึงลักษณะของข้อระวังที่แตกต่างกันออกไป

5.2.2.2.1.3 ส่วนติดต่อกับระบบจัดการฐานข้อมูลออราเคิล (Connecting to ORACLE)

ก่อนที่ประโยคคำสั่งปฏิบัติการใดๆในภาษาสอบถามเชิงโครงสร้าง จะถูกเรียกใช้นั้น จะต้องมีการติดต่อกับระบบจัดการฐานข้อมูลก่อน ด้วยการบอกชื่อผู้ใช้และรหัสผ่าน ซึ่งมีการใช้ดังนี้

EXEC SQL CONNECT <:oracleuser> IDENTIFIED BY <:oraclepasswd> ;

หรือ

EXEC SQL CONNECT <:oracleid> ;

โดยที่ oracleid อยู่ในรูป <:oracleuser>/<:oraclepasswd> จะเป็นส่วนที่ใช้เพื่อให้โปรแกรมสามารถติดต่อกับระบบจัดการฐานข้อมูลแบบสัมพันธ์ของออราเคิลได้ ซึ่งจะทำให้สามารถเข้าถึงข้อมูลในฐานข้อมูลออราเคิลได้

5.2.2.2.2 ส่วนตัวโปรแกรมประยุกต์

ส่วนนี้เป็นส่วนที่ภาษาซีและภาษาสอบถามเชิงโครงสร้างรวมอยู่ด้วยกัน ซึ่งโดยปรกติแล้วภาษาซีจะเป็นตัวจัดการเกี่ยวกับการแสดงผล (Display) และรูปแบบการใช้งานต่างๆของโปรแกรม เช่น เมนู เป็นต้น ส่วนภาษาสอบถามเชิงโครงสร้าง จะทำงานในด้านการจัดการเกี่ยวกับข้อมูล รวมทั้งการติดต่อกับระบบจัดการฐานข้อมูลออราเคิลด้วย ซึ่งการเรียกใช้นั้นจะต้องมี "EXEC SQL" นำหน้าก่อนเสมอ

5.2.2.3 การสอบถามและการฝังคำสั่ง (Query and Embedded)

สำหรับการสอบถามและการผังคำสั่งภาษาสอบถามเชิงโครงสร้าง มีคำสั่งที่ใช้หลายคำสั่งด้วยกัน ประกอบด้วย

SELECT	INTO
FROM	WHERE
CONNECT	UNION
INTERSECT	MINUS
GROUP BY	HAVING
ORDER BY	

สำหรับตัวแปรที่ใช้ในการสอบถามนั้นมาจาก 2 ที่ คือ คำสั่งภาษาสอบถามเชิงโครงสร้างที่ฝัง และจากตัวแปรในภาษาซี ซึ่งตัวแปรในภาษาซีที่ใช้ในการสอบถามจะต้องมีเครื่องหมาย ":" (colon) นำหน้าชื่อตัวแปรเสมอ

สำหรับลักษณะที่ใช้ในการสอบถามมีอยู่ด้วยกัน 2 แบบ คือ

5.2.2.3.1 การสอบถามที่ให้ผลลัพธ์เพียงหนึ่งแถว (Query which return Single row only)

เป็นการสอบถามที่จะต้องอ้างถึงข้อมูล ที่มีอยู่ในตารางเพียงหนึ่งแถวเท่านั้น (Unique Index) ซึ่งถ้าให้ค่ามากกว่า 1 แถวจะแสดงข้อผิดพลาด

5.2.2.3.2 การสอบถามที่ให้ผลลัพธ์มากว่าหนึ่งแถว (Query which return Multiple rows)

การสอบถามลักษณะนี้ มักจะใช้กับการเข้าถึงข้อมูลที่มีอยู่เป็นกลุ่มในตาราง ซึ่งเมื่อกระทำการสอบถามนี้แล้ว ระบบจัดการฐานข้อมูลออราเคิลจะให้ผลลัพธ์ทั้งหมดออกมาในครั้งเดียว ดังนั้นการใช้การสอบถามแบบนี้จึงจำเป็นต้องเตรียมเนื้อที่หน่วยความจำส่วนหนึ่ง เพื่อที่จะใช้ในการเก็บผลลัพธ์นั้นไว้ แล้วจึงเรียกออกมาใช้ทีละแถว (tuple) ตามที่ต้องการ ซึ่งพื้นที่นั้นเรียกว่า เคอร์เซอร์ (Cursor)

เคอร์เซอร์มีลักษณะการใช้ เรียงตามลำดับดังนี้ คือ

ประกาศเคอร์เซอร์ (DECLARE CURSOR) เพื่อกำหนดพื้นที่หน่วยความจำ, ชื่อ และการสอบถามที่ต้องการ

รูปแบบ : EXEC SQL DECLARE <cursorname> CURSOR FOR [Query] ;

การเปิดเคอร์เซอร์ (OPEN CURSOR) เพื่อให้คำถามได้เข้าถึงข้อมูลได้

รูปแบบ : EXEC SQL OPEN <cursorname> ;

การเฟทช์ (FETCH) เพื่อรับข้อมูลผลลัพธ์ตัวต่อไปเข้าสู่ตัวแปร

รูปแบบ : EXEC SQL FETCH <cursorname> INTO <HostVar> ;

การปิดเคอร์เซอร์ (CLOSE CURSOR) เป็นการยกเลิกเคอร์เซอร์ที่ระบุออกไป

รูปแบบ : EXEC SQL CLOSE <cursorname> ;

ตำแหน่งของเคอร์เซอร์ปัจจุบัน (CURRENT OF CURSOR) ใช้ในการอ้างถึงตำแหน่งแถวปัจจุบันในเคอร์เซอร์ที่มีการเฟทช์ข้อมูล เพื่อนำมาใช้ในการแก้ไขหรือลบข้อมูล ตำแหน่งของเคอร์เซอร์ปัจจุบันสามารถใช้แทนด้วย โรว์ไอดี (ROWID)

5.2.2.4 การตกลงและการยกเลิกการทำงาน (Commit and Rollback)

ในการทำงานของโปรแกรม คำสั่งที่เป็นภาษาสอบถามเชิงโครงสร้างแต่ละคำสั่ง จะถูกออราเคิลมองเป็นส่วนย่อย (Logical unit of work) ซึ่งในแต่ละส่วนนี้จะถูกประมวลผลเป็นลำดับขั้นไปจนจบ หรืออาจมีการถูกยกเลิกกลางคันก็ได้ ซึ่งทั้ง 2 กรณีนี้อาจจะเกิดจาก

ผู้เขียนโปรแกรมเป็นผู้ฝังคำสั่งไว้

หรือ

ระบบ (SYSTEM) ทำงานเอง เช่น ไม่สามารถที่จะทำงานต่อไปได้ เนื่องจากการเดทล็อก (Deadlock) ระบบก็จะยกเลิกการทำงาน หรือ ในกรณีที่มีการสร้างตารางเกิดขึ้นระบบก็จะตกลงการทำงานก่อนหน้าทั้งหมด

การเรียกใช้คำสั่งมีอยู่ 2 คำสั่งคือ

5.2.2.4.1 การตกลง

จะเป็นการตกลงยอมรับการเปลี่ยนแปลง ที่เกิดขึ้นทั้งหมด มีรูปแบบการใช้ คือ

EXEC SQL COMMIT WORK [RELEASE] ;

โดยที่ทางเลือก RELEASE จะเป็นการคืนเนื้อที่ในหน่วยความจำทั้งหมดให้ระบบปฏิบัติการ และหยุดการติดต่อกับระบบจัดการฐานข้อมูล (log off) ซึ่งจะเป็นการจบการทำงานครั้งสุดท้าย

5.2.2.4.2 การยกเลิก

จะเป็นการยกเลิกการทำงานทั้งหมดภายหลังที่มีการตกลงครั้งสุดท้าย จะใช้ในกรณีที่มีความผิดพลาดเกิดขึ้น จากการทำงานของโปรแกรม มีรูปแบบการใช้ลักษณะเดียวกับการตกลง คือ

EXEC SQL ROLLBACK WORK [RELEASE] ;

5.2.2.5 การตรวจสอบข้อผิดพลาด (Error Detection)

ในการตรวจสอบความผิดพลาด จะเรียกใช้คำสั่ง WHENEVER ซึ่งจะทำการตรวจสอบสถานะของส่วนการติดต่อกับระบบจัดการฐานข้อมูลทุกๆครั้ง ที่ประมวลผลคำสั่งภาษาสอบถามเชิงโครงสร้าง ที่ได้ฝังไว้ในโปรแกรม มีรูปแบบการใช้ดังนี้

EXEC SQL WHENEVER [SQLERROR | SQLWARNING | NOT FOUND]
[STOP | CONTINUE | GOTO label]

โดยที่ SQLERROR : จะถูกเซ็ตเมื่อ sqlca.sqlcode เป็นลบ
SQLWARNING : จะถูกเซ็ตเมื่อ sqlca.sqlwarn[0] = "w"
NOT FOUND : จะถูกเซ็ตเมื่อ sqlca.sqlcode = +1403 (no row found)
STOP : หยุดการทำงานของโปรแกรม และ ยกเลิกการทำงานก่อนหน้านี้
CONTINUE : ทำงานต่อไป ไม่ว่า sqlca จะเป็นอย่างไร
GOTO label : ข้ามไปทำงานที่ label

5.2.3 การฝังคำสั่งภาษาสอบถามเชิงโครงสร้างแบบจลน์

การฝังคำสั่งภาษาสอบถามเชิงโครงสร้างแบบจลน์ เป็นการฝังคำสั่งภาษาสอบถามเชิงโครงสร้างลงในโปรแกรม โดยไม่จำเป็นจะต้องกำหนดคำสั่งภาษาสอบถามเชิงโครงสร้างไว้ล่วงหน้า เหมือนกับการฝังคำสั่งภาษาสอบถามเชิงโครงสร้างแบบสถิตย์ และไม่จำเป็นจะต้องทราบว่าผลลัพธ์ของคำสั่งจะให้ข้อมูลชนิดอะไรและจำนวนเท่าไหร่ ทั้งนี้คำสั่งภาษาสอบถามเชิงโครงสร้างจะถูกกำหนดในขั้นตอนการเรียกใช้โปรแกรมโดยผู้ใช้งาน

สำหรับขั้นตอนการพัฒนาโปรแกรม ของการฝังคำสั่งภาษาสอบถามเชิงโครงสร้างแบบจลน์ จะเหมือนกับขั้นตอนการพัฒนาโปรแกรมของการฝังคำสั่งภาษาสอบถามเชิงโครงสร้างแบบสถิตย์ จะแตกต่างกันเฉพาะส่วนของวิธีการฝังคำสั่งภาษาสอบถามเชิงโครงสร้างเท่านั้น ในส่วนที่เหมือนกันระหว่างการฝังคำสั่งภาษาสอบถามเชิงโครงสร้างแบบจลน์และแบบสถิตย์ ได้กล่าวไว้ในหัวข้อของการฝังคำสั่งภาษาสอบถามเชิงโครงสร้างแล้ว ดังนั้นในหัวข้อนี้จะไม่ขอกล่าวซ้ำอีก จะอธิบายเฉพาะส่วนที่แตกต่างไปเท่านั้น ดังต่อไปนี้คือ

5.2.3.1 ส่วนประกอบของโปรแกรม

สำหรับส่วนประกอบของโปรแกรมยังคงแบ่งเป็น 2 ส่วนใหญ่ๆ เช่นเดิม คือ ส่วนเริ่มต้นการประยุกต์ และส่วนตัวโปรแกรมการประยุกต์ ซึ่งทั้งสองส่วนมีรายละเอียดดังนี้

5.2.3.1.1 ส่วนเริ่มต้นการประยุกต์

ในส่วนเริ่มต้นการประยุกต์นี้ แบ่งเป็น 4 ส่วนด้วยกัน คือ ส่วนกำหนดตัวแปร ส่วนรวมส่วนสื่อสารกับระบบจัดการฐานข้อมูล ส่วนรวมส่วนอธิบายคำถามและข้อมูล และส่วนติดต่อกับระบบจัดการฐานข้อมูลออราเคิล ทั้ง 4 ส่วนที่กล่าวมา จะมีเพียงส่วนรวมส่วนอธิบายคำถามและข้อมูลเท่านั้น ที่เพิ่มขึ้นมาจากเดิม (จากการฝังคำสั่งภาษาสอบถามเชิงโครงสร้างแบบสถิตย์) ดังนั้นจะขออธิบายในส่วนรวมส่วนอธิบายคำถามและข้อมูล ที่เพิ่มขึ้นมาเท่านั้น

ในส่วนรวมส่วนอธิบายคำถามและข้อมูล จะมีประโยคคำสั่งเรียกใช้ดังนี้ คือ

```
EXEC SQL INCLUDE SQLDA ;
```

ประโยคคำสั่งนี้จะเป็นการกำหนดแบบโครงสร้างของตัวแปรในภาษาหลัก (define structure of host variable) เรียกตัวแปรชนิดนี้ว่า แบบ SQLDA ตัวแปรที่กำหนดเป็นแบบ SQLDA จะมีจุดประสงค์ 2 ประการ คือ

ประการแรก จะใช้ในการรับรายละเอียดของข้อมูลผลลัพธ์ ซึ่งโปรแกรมที่ใช้วิธีการฝังคำสั่งแบบนี้จะต้องกำหนด และ

ประการที่สอง จะใช้ในการรับรายละเอียดของตัวแปรในคำถาม กรณีที่คำถามมีการกำหนดตัวแปร

ดังนั้น ถ้าคำสั่งภาษาสอบถามเชิงโครงสร้างที่ป้อนให้กับโปรแกรมขณะเรียกใช้งาน มีการกำหนดตัวแปรในภาษาหลักด้วย จะต้องมีการกำหนดตัวแปรแบบ SQLDA 2 ตัวดังตัวอย่าง เช่น

```
EXEC SQL INCLUDE SQLDA ;
SQLDA *bindda, *selectda ;
```

5.2.3.1.2 ส่วนตัวโปรแกรมการประยุกต์

สำหรับส่วนตัวโปรแกรมการประยุกต์ จะเหมือนกับส่วนตัวโปรแกรมการประยุกต์ของการฝังคำสั่งภาษาสอบถามเชิงโครงสร้างแบบสถิตย์ ซึ่งได้กล่าวไว้ในหัวข้อ 5.2.2.2.2 แล้ว ดังนั้นในที่นี้จะไม่ขออธิบายซ้ำอีก

5.2.3.2 การสอบถามและการฝังคำสั่ง

สำหรับวิธีการฝังคำสั่งภาษาสอบถามเชิงโครงสร้างแบบจลน์ จะมีวิธีการฝังคำสั่ง 4 วิธีด้วยกัน ซึ่งแต่ละวิธี จะมีขั้นตอนและข้อจำกัดแตกต่างกันไป ดังจะได้อธิบาย คือ

5.2.3.2.1 วิธีที่หนึ่ง

วิธีนี้จะใช้ประโยคคำสั่ง EXECUTE IMMEDIATE คำสั่งที่ฝังในโปรแกรมด้วยวิธีนี้จะมีข้อจำกัด คือ จะต้องเป็นคำสั่งภาษาสอบถามเชิงโครงสร้าง ที่ให้ค่าผลลัพธ์เป็น สำเร็จ หรือ ไม่สำเร็จเท่านั้น และ คำสั่งภาษาสอบถามเชิงโครงสร้างไม่สามารถที่จะมีตัวแปรในภาษาหลักได้ ตัวอย่างการใช้งาน เช่น

```
EXEC SQL BEGIN DECLARE SECTION ;
   ...
VARCHAR dstring[80] ;
   ...
EXEC SQL END DECLARE SECTION ;
if ( scanf("%s",dstring))
        EXEC SQL EXECUTE IMMEDIATE :dstring ;
```

หรือ

```
EXEC SQL EXECUTE IMMEDIATE "DELETE FROM EMP WHERE EMPNO = 9251" ;
```

5.2.3.2.2 วิธีที่สอง

วิธีนี้จะใช้ประโยคคำสั่ง PREPARE และ EXECUTE ซึ่งจะมีข้อจำกัดเหมือนกับวิธีที่หนึ่ง แต่คำสั่งภาษาสอบถามเชิงโครงสร้างจะสามารถมีตัวแปรในภาษาหลักได้ การทำงานจะแยกเป็นสองขั้นตอน คือ ขั้นตอนการเตรียมคำถาม (PREPARE) และขั้นตอนการประมวลผล (EXECUTE) โดยการแยกเป็นสองขั้นตอนนี้ทำให้เพิ่มความสามารถจากวิธีการที่หนึ่งคือ ในหนึ่ง

คำสั่งจะใช้การเตรียมการเพียงครั้งเดียว แต่จะสามารถประมวลผลได้หลายครั้ง ตัวอย่างเช่น คำสั่งที่ป้อนตอนเรียกใช้โปรแกรมเป็นดังนี้

```
DELETE FROM EMP WHERE EMPNO = :PEMPNO
```

โดยที่โปรแกรมมีการฝังคำสั่งดังนี้

```
scanf ("%s",&DSTRING) ;
EXEC SQL PREPARE S FROM :DSTRING ;
scanf ("%d",&PEMPNO) :
while (PEMPNO != 0)
{
        EXEC SQL EXECUTE S USING :PEMPNO ;
        scanf ("%d",&PEMPNO) ;
}
```

5.2.3.2.3 วิธีที่สาม

วิธีนี้จะใช้ประโยคคำสั่ง PREPARE และ FETCH โดยจะมีข้อจำกัดเหมือนกับวิธีที่สอง แต่คำสั่งภาษาสอบถามเชิงโครงสร้างที่ใช้ฝังในโปรแกรม สามารถที่จะเป็นคำถามที่ให้ผลลัพธ์เป็นข้อมูลในฐานข้อมูลได้ ซึ่งมีลำดับการใช้ประโยคคำสั่งดังนี้

```
PREPARE <statement name> FROM <host character string> ;
DECLARE <cursor name> FOR <statement name> ;
OPEN <cursor name> [USING :bnd-var1[,:bnd-var2, ... ]] ;
FETCH <cursor name> INTO :select-var1[,:select-var2, ... ]] ;
CLOSE <cursor name> ;
```

ตัวอย่างการใช้งาน เช่น

```
strcpy (select,"SELECT ENAME , SAL FROM EMP ") ;
scanf ("%s",where) ;
strcat (select , where) ;
EXEC SQL PREPARE S FROM :select ;
EXEC SQL DECLARE C FOR S ;
EXEC SQL OPEN C ;
```

```
for (i=0; ;i++)
{
        EXEC SQL FETCH C INTO :ename , :sal ;
        ...
}
EXEC SQL CLOSE C ;
```

5.2.3.2.4 วิธีที่สี่

การฝังคำสั่งภาษาสอบถามเชิงโครงสร้างด้วยวิธีนี้ จะสามารถกำหนดคำสั่งภาษาสอบถามเชิงโครงสร้างได้ทุกรูปแบบ ซึ่งมีขั้นตอนการฝังคำสั่งดังนี้

5.2.3.2.4.1 กำหนดตัวแปรในภาษาหลักด้วยประโยคคำสั่ง

```
SQLDA*sqlda_name ;
```

ตัวอย่าง เช่น

```
SQLDA*bdp ;
SQLDA*sdp ;
```

5.2.3.2.4.2 เตรียมประโยคคำสั่งภาษาสอบถามเชิงโครงสร้าง ที่รับจากผู้ใช้งานขณะโปรแกรมถูกเรียกใช้ ซึ่งขั้นตอนนี้จะทำการวิเคราะห์และกำหนดชื่อให้ประโยคคำสั่งที่รับมา ด้วยประโยคคำสั่ง

```
EXEC SQL PREPARE <statement name> FROM <host character string> ;
```

ตัวอย่าง เช่น

```
EXEC SQL PREPARE S FROM :stmt ;
```

5.2.3.2.4.3 กำหนดเคอร์เซอร์ให้กับประโยคที่ได้กำหนดชื่อแล้ว ด้วยประโยคคำสั่ง

```
EXEC SQL DECLARE <cursor name> CURSOR FOR <statement name> ;
```

ตัวอย่าง เช่น

```
EXEC SQL DECLARE C CURSOR FOR S ;
```

5.2.3.2.4.4 กำหนดพื้นที่หน่วยความจำสำหรับตัวแปรแบบ SQLDA ด้วยประโยคคำสั่ง

```
sqlda_name = sqlald( a,b,c ) ;
```

เมื่อ

a แทนจำนวนของตัวแปรที่จะอธิบาย

b แทนขนาดของข้อความที่จะเก็บชื่อตัวแปร

c แทนขนาดของข้อความที่จะเก็บชื่อตัวบ่งชี้ตัวแปร

ตัวอย่าง เช่น

```
bdp = sqlald( bdSize,bvSize,10) ;
sdp = sqlald( sdSize,svSize,0) ;
```

5.2.3.2.4.5 คืนพื้นที่หน่วยความจำในกรณีที่เลิกใช้ หรือต้องการคืนเพื่อกำหนดพื้นที่หน่วยความจำใหม่ ให้มีขนาดตามที่ต้องการใช้ ด้วยประโยคคำสั่ง

```
sqlclu( sqlda_name ) ;
```

ตัวอย่าง เช่น

```
sqlclu( bdp ) ;
sqlclu( sdp ) ;
```

5.2.3.2.4.6 กำหนดรายละเอียดสำหรับตัวแปรแบบ SQLDA ของประโยคคำถาม ด้วยประโยคคำสั่ง

```
EXEC SQL DESCRIBE BIND FOR <statement name> INTO <sqlda_name> ;
```

ตัวอย่าง เช่น

```
EXEC SQL DESCRIBE BIND FOR S INTO bdp ;
```

5.2.3.2.4.7 เรียกใช้เคอร์เซอร์ ด้วยประโยคคำสั่ง

```
EXEC SQL OPEN <cursor name> USING DESCRIPTOR <sqlda_name> ;
```

ตัวอย่าง เช่น

```
EXEC SQL OPEN C USING DESCRIPTOR bdp ;
```

5.2.3.2.4.8 กำหนดรายละเอียดสำหรับตัวแปรแบบ SQLDA ของผลลัพธ์ ด้วยประโยคคำสั่ง

```
EXEC SQL DESCRIBE SELECT LIST FOR <statement name> INTO <outputdescriptor> ;
```

ตัวอย่าง เช่น

EXEC SQL DESCRIBE SELECT LIST FOR S INTO sdp ;

5.2.3.2.4.9 เฟทช์ข้อมูล ด้วยประโยคคำสั่ง

EXEC SQL FETCH <cursor name> USING DESCRIPTOR <outputdescriptor> ;

ตัวอย่าง เช่น

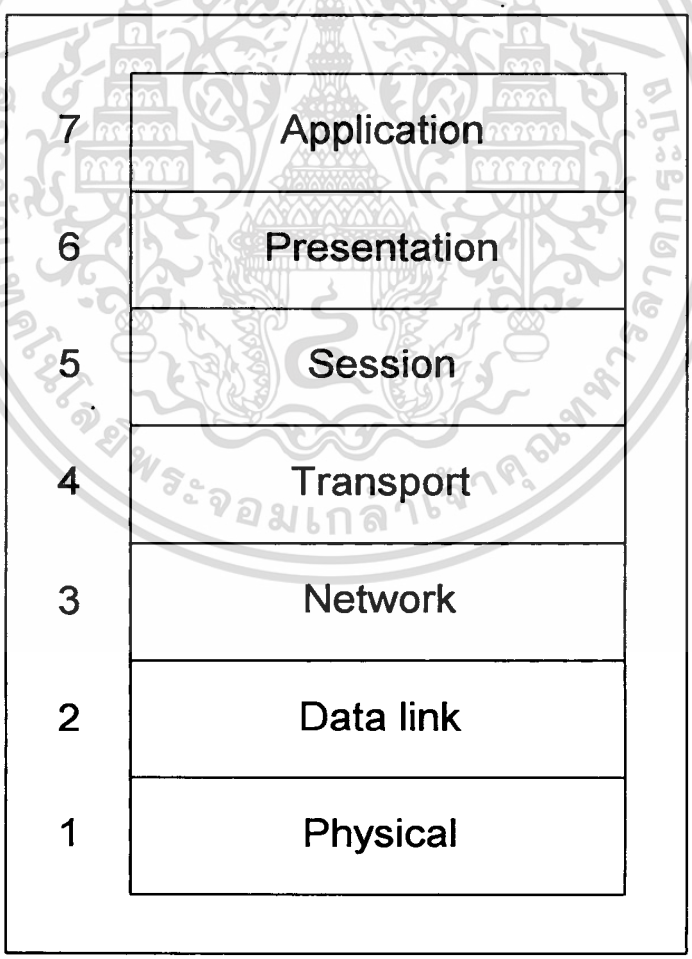
EXEC SQL FETCH C USING DESCRIPTOR sdp ;

5.2.3.2.4.10 ปิดการใช้เคอร์เซอร์ ด้วยประโยคคำสั่ง

EXEC SQL CLOSE <cursor name> ;

ตัวอย่าง เช่น

EXEC SQL CLOSE C ;



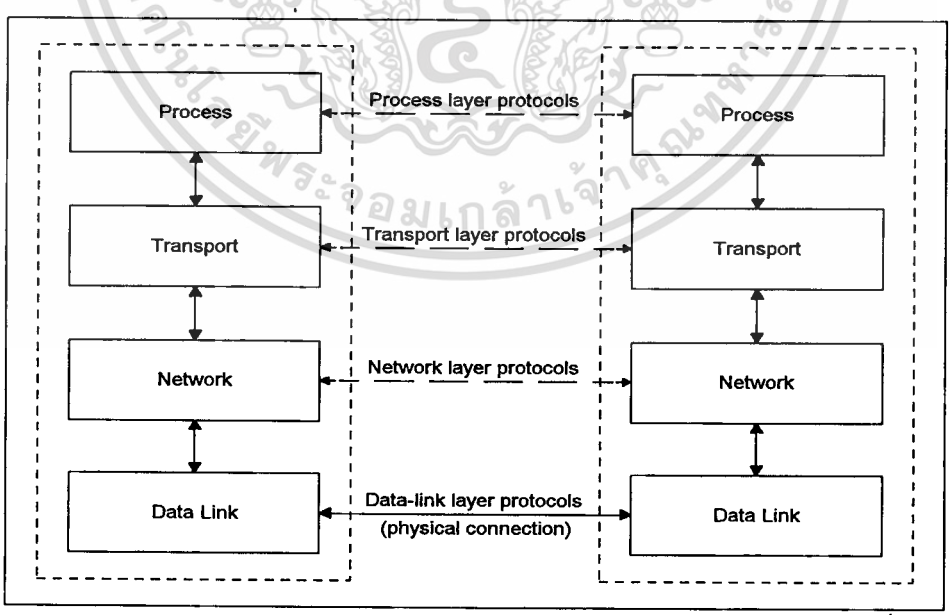
รูปที่ 5.2 แสดงรูปแบบการติดต่อแบบระบบเปิด 7 ชั้นเลเยอร์ (OSI 7-Layer model)

5.3 การใช้คำสั่งเรียกใช้ TPC/IP

ในการติดต่อสื่อสารระหว่างกระบวนการประมวลผลนั้น โดยปรกติในแต่ละกระบวนการประมวลผล จะแบ่งการทำงานออกเป็นส่วนๆ แต่ละส่วนเรียกว่า เลเยอร์ (Layer) แต่ละเลเยอร์จะทำงานแตกต่างกันไปตามหน้าที่ที่ได้กำหนดไว้ รวมทั้งยังสามารถจะสื่อสารกับเลเยอร์ที่อยู่ใกล้เคียงกันได้ ซึ่งองค์การมาตรฐานสากล (International Standards Organization :ISO) ได้กำหนดรูปแบบสำหรับการติดต่อแบบระบบเปิด (Open System Interconnection : OSI) ไว้ 7 ระดับชั้นเลเยอร์ ดังรูปที่ 5.2

ในจำนวนเลเยอร์ 7 ชั้นที่แสดงไว้ในรูปที่ 5.2 นั้น ชั้นเลเยอร์ transport นับว่าเป็นชั้นเลเยอร์ที่มีความสำคัญมากที่สุด เนื่องจากว่าเป็นชั้นเลเยอร์ที่ต่ำที่สุด ที่มีความเชื่อถือได้ในการส่งข้อมูล การเรียกใช้จากชั้นเลเยอร์ที่สูงกว่า จะถือว่าชั้นเลเยอร์ transport นี้ ไม่มีความผิดพลาดในการส่งข้อมูล ซึ่งขั้นตอนต่างๆ เช่น ควบคุมลำดับการส่ง ตรวจสอบความผิดพลาดในการส่งข้อมูล หรือ ส่งข้อมูลซ้ำ เป็นต้น จะถูกจัดการโดยชั้นเลเยอร์นี้ และชั้นเลเยอร์ที่ต่ำกว่า

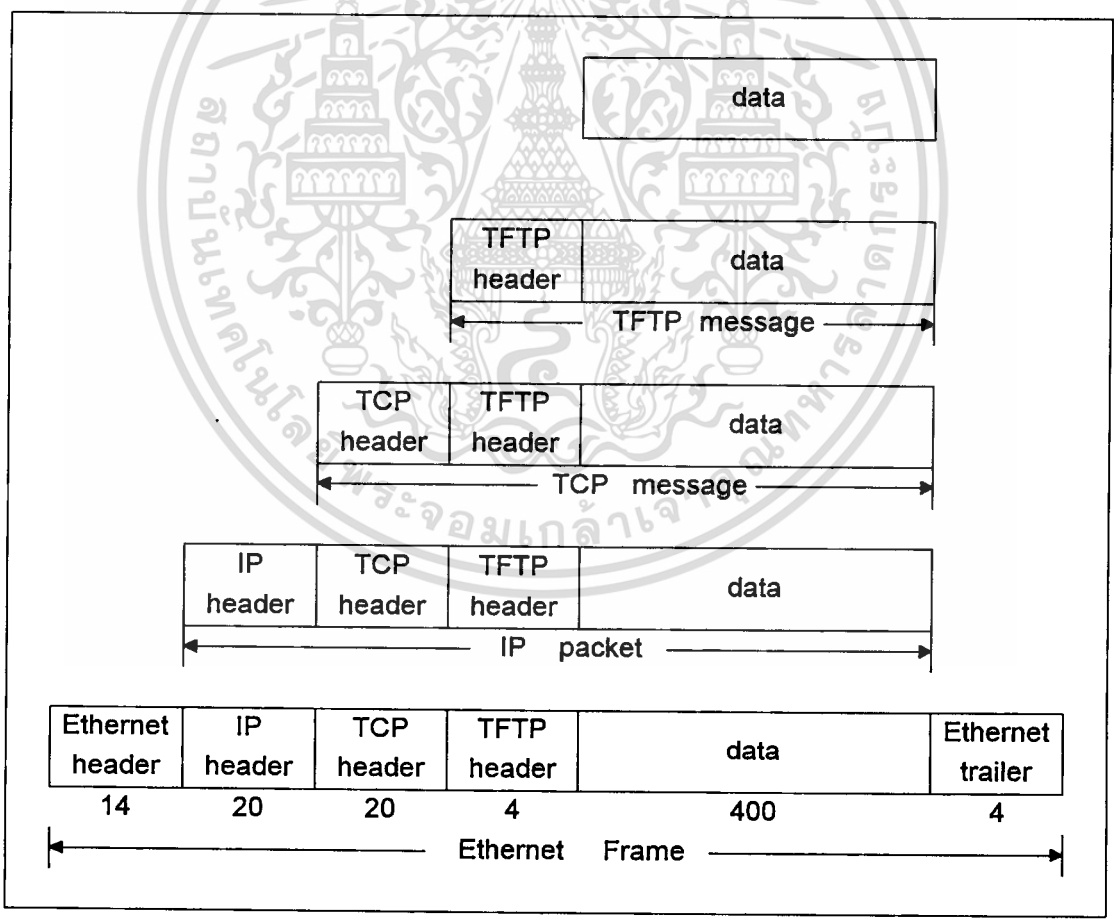
โดยทั่วไปแล้ว โปรแกรมประยุกต์มักจะติดต่อกันด้วยโปรโตคอลในสามเลเยอร์ คือ session , presentation และ application ดังนั้น ทั้งสามชั้นเลเยอร์จึงมักถูกเรียกว่าชั้นเลเยอร์ process รวมทั้งชั้นเลเยอร์ data link และชั้นเลเยอร์ physical มักจะเรียกรวมกันว่าชั้นเลเยอร์ data-link ดังแสดงในรูปที่ 5.3



รูปที่ 5.3 แสดงรูปแบบการติดต่ออย่างง่าย 4 ชั้นเลเยอร์

ในรูปที่ 5.3 เป็นการแสดงการเชื่อมต่อระบบ 2 ระบบด้วยโครงข่าย การเชื่อมต่อระบบทั้งสองจะเกิดขึ้นจริงๆที่ชั้นเลเยอร์ data-link (แสดงด้วยเส้นทึบ) แม้ว่ากระบวนการประมวลผลทั้งสองจะเสมือนติดต่อกันได้จริง (แสดงด้วยเส้นประ) แต่ข้อมูลจะถูกส่งผ่านจากชั้นเลเยอร์ process ลงมายังชั้นเลเยอร์ transport ชั้นเลเยอร์ network และชั้นเลเยอร์ data-link ข้ามโครงข่ายผ่านไปยังชั้นเลเยอร์ data-link ของอีกระบบหนึ่ง และผ่านชั้นเลเยอร์ network ขึ้นไปยังชั้นเลเยอร์ transport แล้วส่งให้กับระดับชั้นเลเยอร์ process จะเห็นว่าข้อมูลถูกส่งผ่านระหว่างชั้นเลเยอร์ ที่อยู่ติดกันทั้งบนและล่าง ดังนั้นการเชื่อมต่อ (interface) ระหว่างชั้นเลเยอร์นับว่ามีความสำคัญสำหรับการเขียนโปรแกรมโครงข่ายเป็นอย่างมาก

ในหัวข้อนี้ จะได้กล่าวถึงรายละเอียดของวิธีการเชื่อมต่อ ระหว่างชั้นเลเยอร์ Transport กับชั้นเลเยอร์ที่สูงกว่า ด้วยวิธีการเชื่อมต่อที่นิยมใช้กัน คือวิธีการเชื่อมต่อโดยการใช้ซอคเกต (socket) ของเบิคเลย์ (Berkeley) ด้วยการใช้โปรโตคอล TCP/IP ที่เรียกใช้ด้วยโปรแกรมที่เขียนด้วยภาษา C บนระบบปฏิบัติการยูนิกซ์เท่านั้น

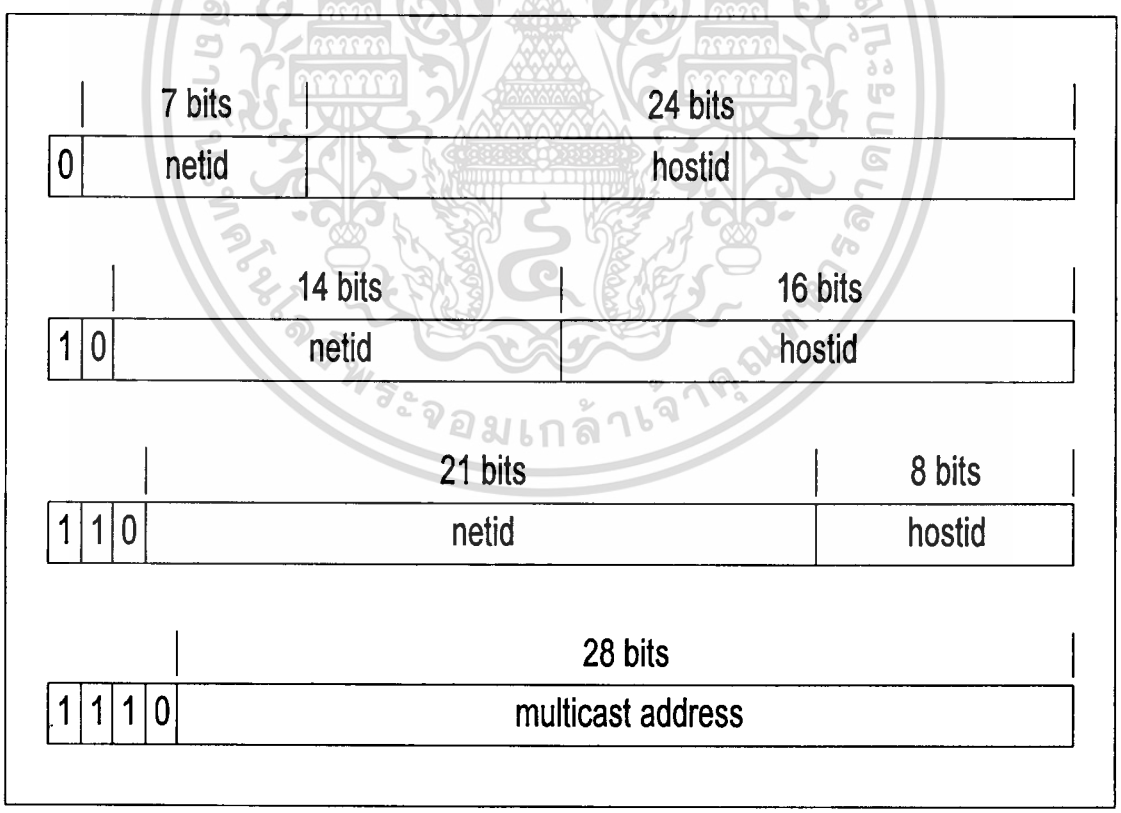


รูปที่ 5.4 แสดงการใส่เปลือกครอบรายละเอียดของโปรโตคอลแต่ละชั้น

5.3.1 การใส่เปลือกครอบ (Encapsulation)

จากที่ได้กล่าวมาข้างต้นนั้น เป็นที่ทราบแล้วว่า ในการส่งข้อมูลของแต่ละเลเยอร์ จะทำการส่งข้อมูลผ่านไปยังชั้นเลเยอร์ที่ต่ำกว่า ซึ่งชั้นเลเยอร์ที่ต่ำกว่าจะถือว่าข้อมูลที่ได้รับเป็นข้อมูลจริงที่ต้องส่งต่อ ก็จะทำการเพิ่มข้อมูลรายละเอียดของโปรโตคอลในชั้นเลเยอร์นั้นครอบข้อมูลจริงแล้วส่งต่อไปยังชั้นเลเยอร์ที่ต่ำกว่า ซึ่งชั้นเลเยอร์ที่ต่ำกว่าก็จะถือว่าข้อมูลที่ได้รับเป็นข้อมูลจริงและเพิ่มข้อมูลครอบเข้าไป จะเป็นเช่นนี้ไปเรื่อยๆจนกว่าจะถึงชั้นที่มีการส่งข้อมูลทางกายภาพจริงๆ จึงจะส่งข้อมูลข้ามโครงข่าย ส่วนฝ่ายรับข้อมูล จะรับข้อมูลที่ชั้นเลเยอร์ที่ต่ำสุด และจะถอดข้อมูลรายละเอียดโปรโตคอลของชั้นนั้นออก จากนั้นจะส่งข้อมูลให้กับชั้นเลเยอร์ที่สูงกว่า เป็นเช่นนี้ไปเรื่อยๆจนกว่าจะถึงชั้นเลเยอร์ระดับเดียวกันกับฝ่ายส่ง

ในรูปที่ 5.4 โปรแกรมประยุกต์ TFTP ใช้โปรโตคอล TCP/IP และเชื่อมต่อ 2 ระบบด้วยอีเธอร์เนท (Ethernet) ถ้าต้องการส่งไฟล์ข้อมูลขนาด 400 ไบต์ จะเห็นว่า TFTP จะเพิ่มข้อมูล 4 ไบต์ และ TCP จะเพิ่มข้อมูล 20 ไบต์ จนสุดท้ายข้อมูลที่ส่งผ่าย อีเธอร์เนททั้งหมดจะเป็น 462 ไบต์

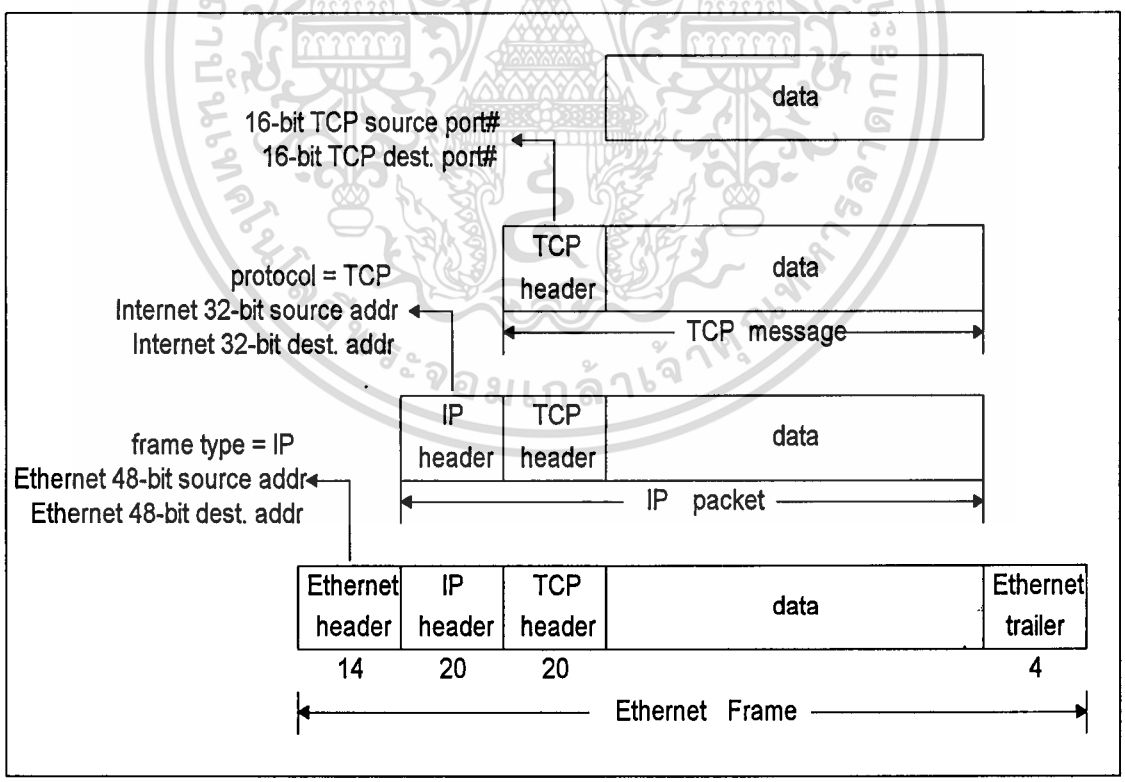


รูปที่ 5.5 แสดงรูปแบบที่อยู่แบบอินเทอร์เนท

5.3.2 ที่อยู่แบบอินเทอร์เนท (Internet Addresses)

โดยทั่วไปแล้ว โปรโตคอลแต่ละชนิดจะมีการกำหนดที่อยู่ (address) สำหรับคอมพิวเตอร์เพื่อที่จะระบุโครงข่าย และเครื่องคอมพิวเตอร์ที่ต้องการอ้างถึง สำหรับที่อยู่แบบอินเทอร์เนทนี้ จะประกอบด้วยข้อมูล 32 บิท ซึ่งข้อมูลนี้จะเก็บหมายเลขโครงข่าย (network ID) และหมายเลขคอมพิวเตอร์ (host ID) ดังนั้นที่อยู่หนึ่งๆจะสามารถที่จะระบุคอมพิวเตอร์ได้เพียงเครื่องเท่านั้น อย่างไรก็ตาม ในกรณีที่เครื่องคอมพิวเตอร์อยู่ในโครงข่ายมากกว่าหนึ่งโครงข่าย ก็จะมีที่อยู่ได้มากกว่าหนึ่งที่อยู่ โดยจะสอดคล้องกับจำนวนโครงข่ายที่คอมพิวเตอร์เครื่องนั้นอยู่ ที่อยู่แบบอินเทอร์เนทนี้ จะถูกกำหนดโดยศูนย์ข้อมูลโครงข่าย (Network Information Center : NIC) ซึ่งจะมีรูปแบบเป็นหนึ่งในสี่รูปแบบที่แสดงไว้ในรูปที่ 5.5

ปรกติแล้ว ที่อยู่แบบอินเทอร์เนท มักจะถูกเขียนในรูปของตัวเลข ที่ถูกแบ่งด้วยจุดทศนิยมออกเป็น 4 ชุด ตัวเลขแต่ละชุดจะแทนข้อมูล 1 ไบต์ของที่อยู่ ตัวอย่าง เช่น ที่อยู่ 0x0102FF04 จะเขียนได้เป็น 1.2.255.4. ซึ่งในระบบปฏิบัติการยูนิกซ์ จะเก็บที่อยู่แบบตัวเลขไว้ในไฟล์ชื่อ /etc/hosts โดยไฟล์ /etc/hosts นี้จะเก็บที่อยู่ของคอมพิวเตอร์ทุกเครื่อง ที่อยู่ในโครงข่ายเดียวกันกับคอมพิวเตอร์ที่มีไฟล์ /etc/hosts อยู่



รูปที่ 5.6 แสดงการใส่เปลือกครอบของข้อมูล TCP บนการเชื่อมต่ออีเธอร์เนท

5.3.3 หมายเลขพอร์ท (Port Numbers)

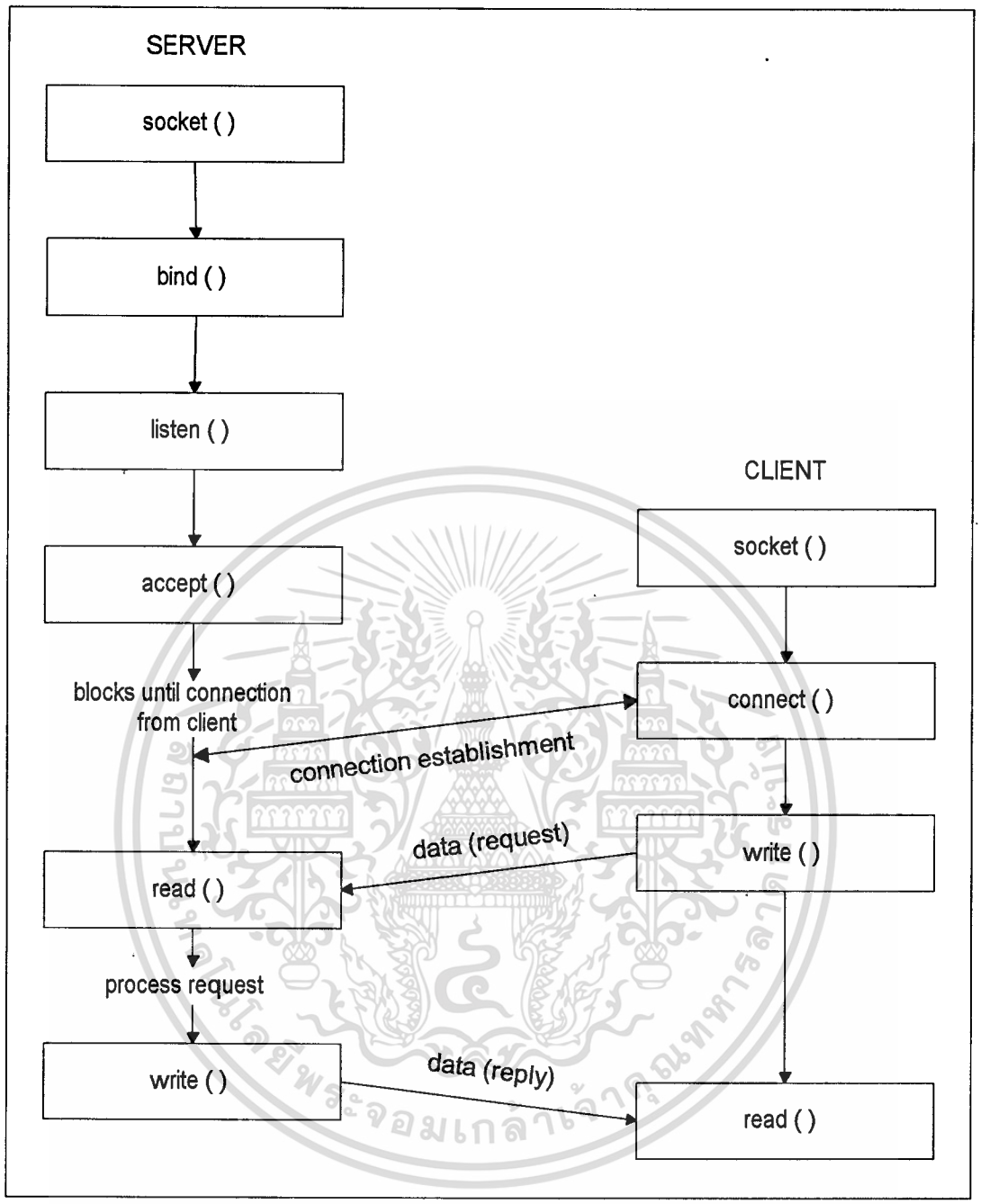
จากการที่ได้มีการกำหนดที่อยู่แบบอินเทอร์เนท ทำให้สามารถที่จะระบุคอมพิวเตอร์ที่ต้องการจะส่งข้อมูลไปถึงได้ อย่างไรก็ตาม ในคอมพิวเตอร์เครื่องหนึ่งๆอาจจะมีกระบวนการมากกว่าหนึ่งกระบวนการที่ใช้โปรโตคอล TCP/IP ได้ ดังนั้นจึงมีการกำหนดหมายเลขพอร์ท (Port Number) ขึ้น เพื่อใช้ระบุถึงกระบวนการที่ต้องการติดต่อ ซึ่งหมายเลขพอร์ทนี้จะเป็นข้อมูลขนาด 16 บิท กระบวนการบริการ (server) จะได้รับการกำหนดหมายเลขพอร์ทไว้ เพื่อรอรับการติดต่อจากกระบวนการลูกข่าย (client) โดยหมายเลขพอร์ทมักจะเป็นที่ทราบกันโดยทั่วไปในการติดต่อ (well-know port) เช่น โปรแกรมประยุกต์ FTP จะใช้หมายเลขพอร์ท 21 สำหรับกระบวนการบริการที่คอยรับการติดต่อจากกระบวนการลูกข่าย ที่ต้องการรับ-ส่งไฟล์

เมื่อกระบวนการลูกข่ายติดต่อไปยังกระบวนการบริการ กระบวนการบริการจะทราบได้ว่ามีการติดต่อมาจากคอมพิวเตอร์หมายเลขใด โดยทราบได้จากข้อมูลที่อยู่แบบอินเทอร์เนทขนาด 32 บิทในรายละเอียดของ IP (IP datagram) และหมายเลขพอร์ทของกระบวนการลูกข่ายจากหมายเลขพอร์ทขนาด 16 บิท ที่อยู่ในรายละเอียด TCP ดังแสดงไว้ในรูปที่ 5.6 ซึ่งหมายเลขพอร์ทนี้จะถูกกำหนดโดยคอมพิวเตอร์ท้องถิ่น เรียกว่า ephemeral port number เพื่อให้กระบวนการลูกข่ายใช้ในการติดต่อกับกระบวนการบริการ หมายเลขพอร์ท ephemeral port number ที่กำหนดโดยคอมพิวเตอร์ท้องถิ่น จะมีค่าไม่ซ้ำกันในการกำหนดให้แต่ละกระบวนการลูกข่าย รวมทั้งหมายเลขพอร์ทที่ผู้เขียนโปรแกรมกำหนดให้กับกระบวนการบริการก็จะต้องไม่ซ้ำกันด้วย ดังนั้นจึงมีหลักเกณฑ์ง่ายๆในการใช้หมายเลขพอร์ทดังนี้

ในการกำหนดหมายเลขพอร์ทของโปรโตคอลกลุ่มอินเทอร์เนท จะใช้หมายเลขพอร์ท 1 ถึง 255 เป็นหมายเลขพอร์ทสำรอง (reserved port) ซึ่งหมายเลขพอร์ทที่ทราบโดยทั่วไปก็จะอยู่ในกลุ่มนี้ ระบบปฏิบัติการบางรุ่น เช่น 4.3BSD จะสำรองพอร์ทหมายเลข 1 ถึง 1023 ไว้สำหรับผู้ดูแลระบบเป็นผู้เรียกใช้ ส่วนงานทั่วไปมักจะถูกกำหนดหมายเลขพอร์ทในช่วง 1024 ถึง 5000 อย่างไรก็ตาม สำหรับผู้ใช้งานใหม่ๆควรเลือกใช้หมายเลขพอร์ทมากกว่า 5000 ขึ้นไป

5.3.4 การเรียกใช้ฟังก์ชันสำหรับโปรโตคอล TPC/IP

ในการเรียกใช้ฟังก์ชัน เพื่อเขียนโปรแกรมรับ-ส่งข้อมูล ที่ใช้โปรโตคอล TCP/IP นั้น สามารถที่จะเรียกใช้ได้ 2 วิธีคือ connection-oriented protocol ซึ่งส่วนรับและส่งข้อมูลจะสร้างการติดต่อทางลอกจิกก่อนที่จะมีการรับ-ส่งข้อมูลกัน และ connectionless protocol ซึ่งจะเป็นลักษณะที่ตรงกันข้ามกับวิธีแรก ตัวอย่างที่แสดงในรูป 5.6 เป็นการเรียกใช้ฟังก์ชัน โดยวิธี connection-oriented protocol สำหรับรายละเอียดจะได้กล่าวเป็นหัวข้อไป



รูปที่ 5.7 แสดงลำดับการเรียกใช้ฟังก์ชัน ในการสื่อสารด้วยซอคเกต

5.3.4.1 โครงสร้างของตัวแปรที่อยู่

จากที่ได้กล่าวมาข้างต้นแล้วว่า การเขียนโปรแกรมรับ-ส่งข้อมูลทางระบบสื่อสารนั้น ตัวโปรแกรมจำเป็นต้องทราบ address ที่จะส่งไป หรือที่จะรับมา แน่นอน ซึ่ง socket address เป็นรูปแบบโครงสร้างของ address ที่ใช้ใน Berkley Socket กำหนดไว้ใน <sys/socket.h> ดังนี้

```
struct sockaddr {
        u_short sa_family ;     /* address family: AF_xxx value */
        char    sa_data[14] ;   /* up to 14 bytes of protocol-specific address */
} ;
```

ลักษณะของโครงสร้าง sockaddr นี้ จะใช้กับทุกโปรโตคอลที่เรียกใช้ฟังก์ชัน socket สำหรับ internet family แล้ว โครงสร้างของ sockaddr จะกำหนดไว้ใน <netinet/in.h> ดังนี้

```
struct in_addr {
        u_long  s_addr ;        /* 32-bit netid/hostid */
                                /* network byte ordered */
} ;

struct sockaddr_in {
        short           sin_family ;    /* AF_INET */
        u_short         sin_port ;      /* 16-bit port number */
                                        /* network byte ordered */
        struct in_addr  sin_addr ;      /* 32-bit netid/hostid */
                                        /* network byte ordered */
        char            sin_zero[8] ;   /* unused */
} ;
```

5.3.4.2 ฟังก์ชันเรียกใช้ (function calls)

รายละเอียดของฟังก์ชันต่างๆที่ควรทราบ สำหรับการใช้โปรโตคอล TCP/IP มีดังต่อไปนี้ คือ

5.3.4.2.1 socket

คำสั่งนี้จะทำหน้าที่จัดการกับ network I/O ซึ่งเป็นคำสั่งแรกของกลุ่มที่จะต้องเรียกใช้ โดยการเรียกใช้จะต้องกำหนดโปรโตคอลที่ต้องการใช้ มีรูปแบบการใช้ดังนี้

```
#include        <sys/types.h>
#include        <sys/socket.h>
int     socket (int family, int type, int protocol) ;
```

	AF_UNIX	AF_INET	AF_NS
SOCK_STREAM	Yes	TCP	SPP
SOCK_DGRAM	Yes	UDP	IDP
SOCK_ARW		IP	Yes
SOCK_SEQPACKET			SPP

รูปที่ 5.8 แสดงโปรโตคอลที่สอดคล้องกันระหว่าง socket family และ type

family คือกลุ่มของโปรโตคอลที่เรียกใช้ ได้แก่

AF_UMIX

AF_INET

AF_NS

AF_IMPLINK

type เป็นการกำหนดลักษณะการส่งข้อมูลได้แก่

SOCK_STREAM

SOCK_DGRAM

SOCK_RAW

SOCK_SEQPACKET

SOCK_RDM

ซึ่งการเลือก type นี้จะต้องสัมพันธ์กับกลุ่มโปรโตคอลที่เลือกใช้ และโปรโตคอลที่ใช้ด้วย ดังแสดงในรูปที่ 5.8

protocol ปรกติแล้วพารามิเตอร์ตัวนี้ จะกำหนดเป็น 0

5.3.4.2.2 bind

เป็นการกำหนดชื่อ (address) ให้แก่ socket ที่ยังไม่มีชื่อ มีการใช้ดังนี้

```
#include        <sys/types.h>
#include        <sys/socket.h>
int     bind ( int sockfd, struct sockaddr *myaddr, int addrlen) ;
```

พารามิเตอร์ที่ 2 จะเป็นพอยน์เตอร์ที่ชี้ไปยัง address ของโปรโตคอลที่ระบุ และพารามิเตอร์ที่ 3 จะเป็นขนาดของ address นั้น

5.3.4.2.3 connect

กระบวนการลูกข่าย จะติดต่อกับกระบวนการบริการ โดยการเรียกใช้ฟังก์ชันนี้ ต่อจากฟังก์ชัน socket ซึ่งมีการใช้ดังนี้

```
#include    <sys/types.h>
#include    <sys/socket.h>
int     connect ( int sockfd, struct sockaddr *servaddr, int addrlen) ;
```

5.3.4.2.4 listen

ฟังก์ชันนี้ใช้สำหรับกระบวนการบริการแบบ connection-oriented เพื่อที่จะแสดงว่ากระบวนการบริการกำลังรอรับการติดต่อจากกระบวนการลูกข่าย ซึ่งมีการใช้ดังนี้

```
int     listen (int sockfd, int backlog) ;
```

โดยปรกติฟังก์ชัน listen จะเรียกใช้ตามหลังฟังก์ชัน socket และ bind แต่จะก่อน ฟังก์ชัน accept พารามิเตอร์ backlog เป็นจำนวนที่ระบุถึงจำนวนการติดต่อจากกระบวนการลูกข่ายที่จะสามารถรอในคิวได้ ขณะที่กระบวนการบริการกำลังประมวลผลฟังก์ชัน accept ซึ่งปรกตินิยมใช้จำนวนมากสุด คือ 5

5.3.4.2.5 accept

ในกระบวนการบริการแบบ connection-oriented จะเรียกใช้ฟังก์ชันนี้ ต่อจากฟังก์ชัน listen ทันที ขั้นตอนการรอการติดต่อจากกระบวนการลูกข่าย จะเกิดขึ้นจริงๆหลังจากที่ได้ประมวลผลฟังก์ชันนี้แล้ว โดยคำสั่ง accept จะติดต่อกับการขอติดต่อที่อยู่ในคิว ด้วยการสร้าง socket ขึ้นอีก socket หนึ่ง ที่มีคุณสมบัติเหมือน sockfd ทุกประการ ซึ่งมีการใช้ดังนี้

```
#include    <sys/types.h>
#include    <sys/socket.h>
int     accept (int sockfd, struct sockaddr *peer, int *addrlen) ;
```

พารามิเตอร์ peer และ addrlen นั้น เป็นตัวแปรที่ส่งให้ฟังก์ชันเพื่อรับค่าที่อยู่ของกระบวนการที่ติดต่อเข้ามา และขนาดของโครงสร้างของที่อยู่ ตามลำดับ

5.3.4.2.6 close

เป็นฟังก์ชันที่ใช้สำหรับบอกการสิ้นสุดของการติดต่อ

```
int     close (int fd) ;
```

5.3.4.2.7 ฟังก์ชันเรียกใช้ในการลำดับไบต์ (byte ordering)

นอกจากฟังก์ชันที่กล่าวมาข้างต้นแล้ว ยังมีฟังก์ชันที่จำเป็นสำหรับการเรียกใช้ฟังก์ชัน ที่เกี่ยวกับโปรโตคอล อีกจำนวนหนึ่ง คือ

```
#include        <sys/types.h>
#include        <netinet/in.h>
u_long  htonl (u_long hostlong) ;
u_short htons (u_short hostshort) ;
u_long  ntohl (u_long netlong) ;
u_short ntohs (u_short netshort) ;
```

ซึ่งฟังก์ชันทั้ง 4 นี้เป็นการจัดลำดับของไบต์สูงและไบต์ต่ำ (byte ordering) ให้อยู่ในรูปแบบที่ต้องการ เช่น ฟังก์ชัน htonl เป็นการแปลงค่าตัวแปรชนิด u_long ในรูปแบบที่มีการเรียงลำดับไบต์ของ host ให้มีรูปแบบที่ใช้ในระบบเครือข่ายทั่วไป

5.3.4.2.8 bcopy

มีรูปแบบการดังนี้คือ

```
bcopy (char *src, char *dest, int nbytes) ;
```

เป็นการสำเนาข้อมูลจำนวน nbytes จากตำแหน่งหน่วยความจำที่ชี้โดยตัวแปร src ไปยังตำแหน่งหน่วยความจำที่ชี้โดยตัวแปร dest ซึ่งใน UNIX System V จะใช้ฟังก์ชัน memcopy แทน

5.3.4.2.9 bzero

มีรูปแบบการใช้ดังนี้คือ

```
bzero (char *dest, int nbytes) ;
```

เป็นการกำหนดค่า 0 ให้กับหน่วยความจำจำนวน nbytes เริ่มจากตำแหน่งหน่วยความจำที่ชี้โดยตัวแปร dest ฟังก์ชันนี้ใน UNIX System V จะใช้ฟังก์ชัน memset แทน

5.3.4.2.10 bcmp

มีรูปแบบการใช้ดังนี้คือ

```
int bcmp (char *ptr1, char *ptr2, int nbytes) ;
```

เป็นการเปรียบเทียบข้อมูลจำนวน nbytes ที่ชี้ตำแหน่งหน่วยความจำโดยตัวแปร ptr1 กับข้อมูลที่ชี้ตำแหน่งหน่วยความจำโดยตัวแปร ptr2 ฟังก์ชันนี้ใน UNIX System V จะใช้ฟังก์ชัน memcmp แทน

5.3.4.2.11 การแปลงรูปแบบที่อยู่ (Address Conversion)

นอกจากฟังก์ชันที่กล่าวมาข้างต้น ยังมีฟังก์ชันอีก 2 ฟังก์ชัน ที่ใช้สำหรับแปลงรูปแบบที่อยู่อินเทอร์เนท ระหว่างรูปแบบที่เขียนในลักษณะทศนิยม 4 ตัว และรูปแบบเลขฐานสองขนาด 32 บิท ซึ่งได้แก่ฟังก์ชันต่อไปนี้

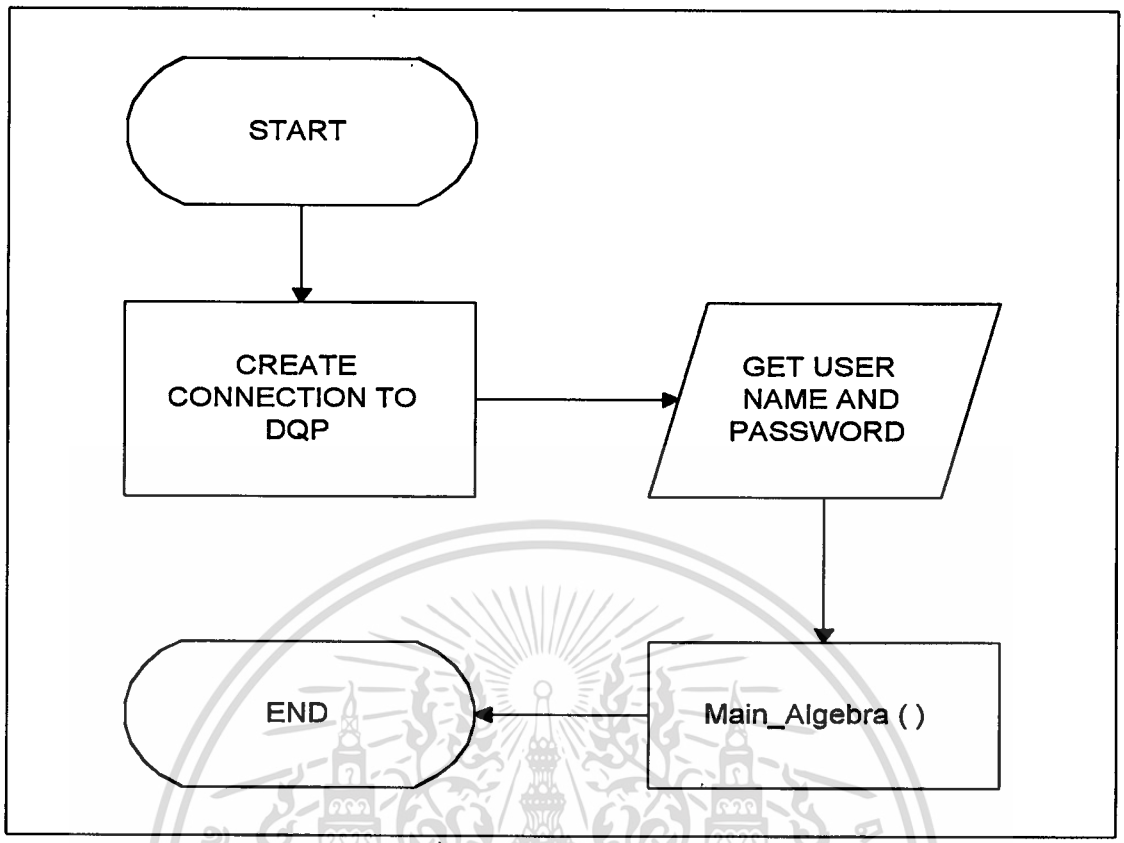
```
#include        <sys/socket.h>
#include        <netinet/in.h>
#include        <arpa/inet.h>
unsigned long   inet_addr (char *ptr) ;
char            * inet_ntoa (atruct in_addr inaddr) ;
```

สำหรับตัวอย่างโปรแกรมที่ใช้โปรโตคอล TCP/IP ได้กล่าวไว้ในภาคผนวก ก ซึ่งเป็นการส่งข้อมูลจากกระบวนการลูกข่าย ไปยังกระบวนการบริการ แล้วส่งกลับมายังกระบวนการลูกข่ายแสดงผลออกทางจอภาพ

5.4 หน่วยโปรแกรมที่พัฒนาขึ้น (Program Modules)

หน่วยโปรแกรมของระบบ DDQ/1 ที่ได้พัฒนาขึ้นนี้ สามารถแบ่งออกเป็นส่วนๆตามลักษณะการทำงานได้ 7 ส่วน ดังนี้ ซึ่งรายละเอียดในแต่ละส่วนจะได้อธิบายในหัวข้อย่อยต่อไป

1 ส่วนติดต่อผู้ใช้งาน
2 ส่วนประมวลผลคำถาม
3 ส่วนจัดการส่งข้อมูลผ่านโครงข่ายสื่อสาร
4 ส่วนสร้างตารางพจนานุกรมฐานข้อมูล
5 ส่วนป้อนข้อมูลพจนานุกรมฐานข้อมูล
6 ส่วนป้อนกฎบังคับความถูกต้อง
7 ส่วนเครื่องมือ



รูปที่ 5.9 ขั้นตอนการทำงานของหน่วยโปรแกรม main( )

5.4.1 ส่วนติดต่อผู้ใช้งาน (User Interface Unit)

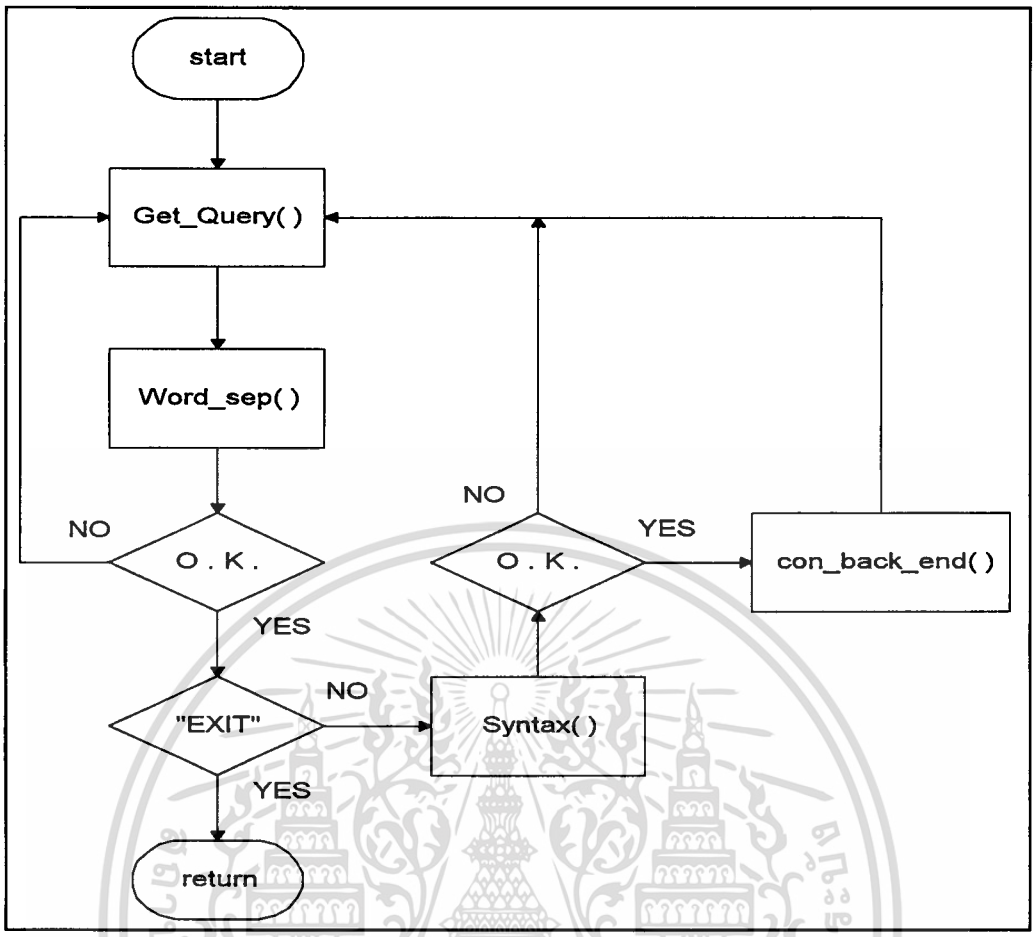
หน่วยโปรแกรมในส่วนติดต่อผู้ใช้งานจะประกอบไปด้วย หน่วยต่างๆ ที่กระจายอยู่ในไฟล์ agb.c และ agblex.c ดังนี้

5.4.1.1 main(argc,argv)

หน่วยโปรแกรมส่วนนี้จะเป็นส่วนโปรแกรมหลักของส่วนติดต่อผู้ใช้งาน โดยจะทำหน้าที่ในการรับชื่อผู้ใช้งานและรหัสผ่าน แล้วสร้างการติดต่อกับส่วนประมวลผลคำถามแบบกระจาย จากนั้นจะเรียกใช้ฟังก์ชัน Main_Algebra( ) ให้ทำงานต่อไป ดังแผนภาพในรูปที่ 5.9

5.4.1.2 Main_Algebra(sockfd)

หน่วยโปรแกรมส่วนนี้จะทำหน้าที่เป็นหน่วยโปรแกรมหลัก ที่เรียกใช้หน่วยโปรแกรมอื่นในการรับคำถาม ส่งคำถามให้ส่วนประมวลผลคำถามแบบกระจาย จนถึงขั้นตอนการแสดงผล และจะวนรอบการทำงานไปเรื่อยๆ จนกว่าผู้ใช้งานจะเลิกใช้ ดังแผนภาพในรูปที่ 5.10



รูปที่ 5.10 แสดงขั้นตอนในการเรียกใช้หน่วยโปรแกรมอื่นของหน่วยโปรแกรม Main_Algebra( )

5.4.1.3 Get_Query(str)

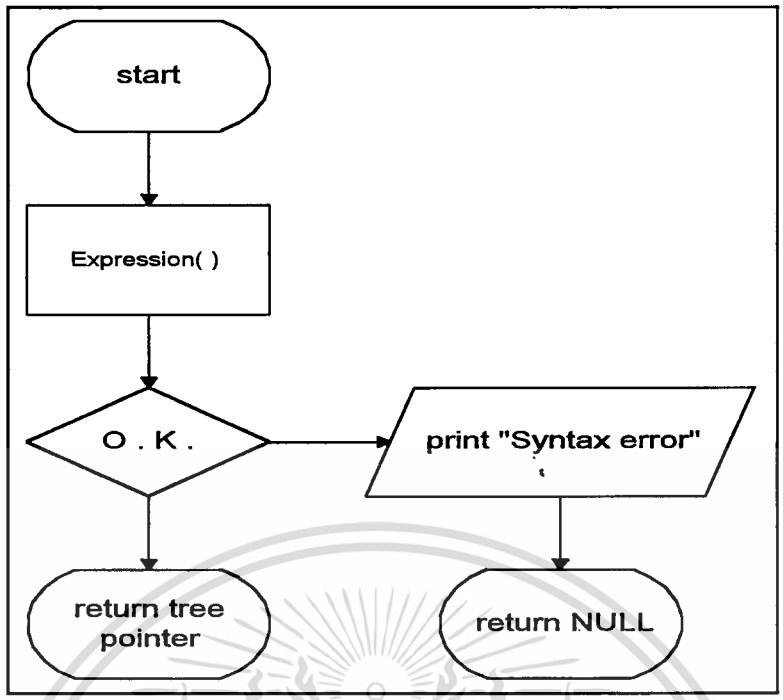
หน่วยโปรแกรมส่วนนี้จะทำหน้าที่ในการรับข้อความคำถาม ของผู้ใช้งานจากแป้นคีย์ แล้วนำคำถามที่ได้เก็บไว้ในตัวแปรชื่อ Query

5.4.1.4 SeparateWord( )

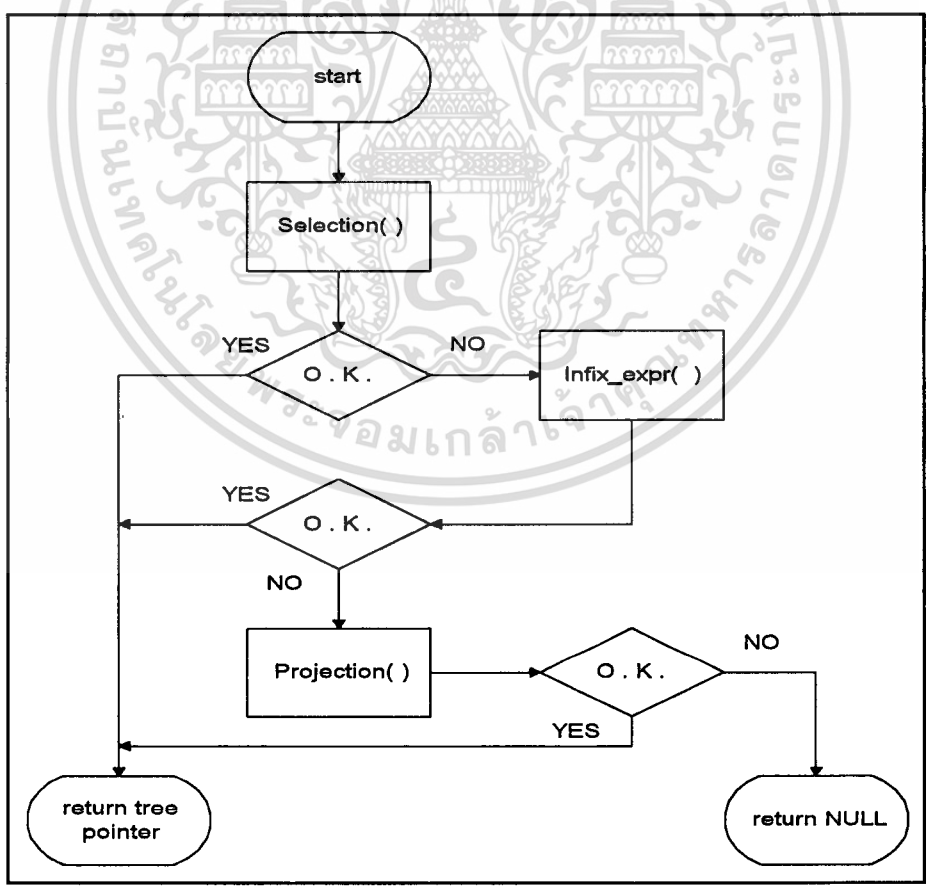
หน่วยโปรแกรมนี้ จะนำข้อความคำถามที่เก็บไว้ในตัวแปร Query มาทำการแยกคำ และนำผลลัพธ์ที่ได้เก็บไว้ในตัวแปรชื่อ Token_No[ ] ซึ่งเป็นตัวแปรแบบอาร์เรย์ (array) ตามตำแหน่งของคำในข้อความ

5.4.1.5 Syntax( )

หน่วยโปรแกรมส่วนนี้จะเป็นหน่วยโปรแกรมหลัก ในการเรียกใช้หน่วยโปรแกรมอื่นเพื่อทำการตรวจสอบไวยากรณ์ของคำถาม ว่าเป็นไปตามหลักไวยากรณ์ของภาษาพีชคณิตสัมพันธ์หรือไม่ ถ้าไม่ถูกต้องตามไวยากรณ์ก็จะแสดงข้อความบอกความผิดพลาด ดังแผนภาพในรูปที่ 5.11



รูปที่ 5.11 แสดงขั้นตอนในการเรียกใช้หน่วยโปรแกรมอื่นของหน่วยโปรแกรม Syntax( )



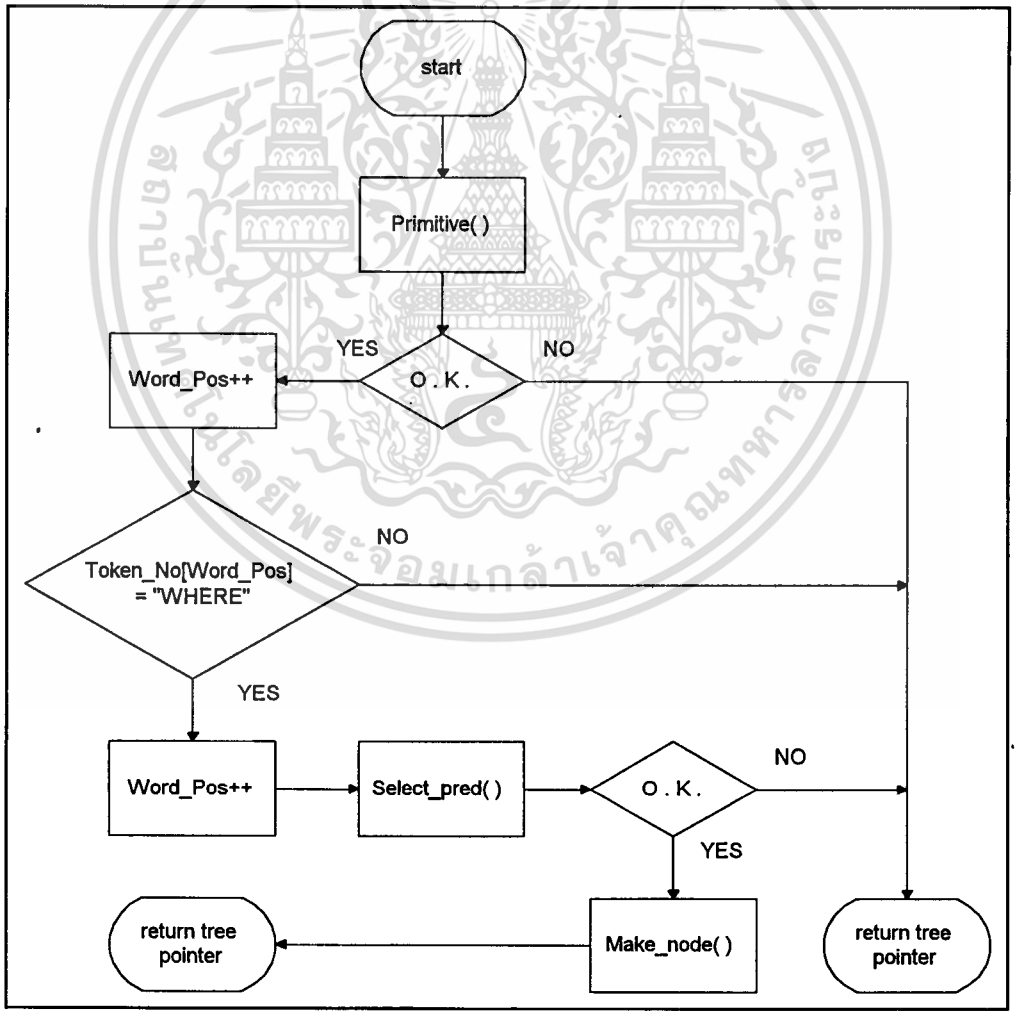
รูปที่ 5.12 แสดงขั้นตอนในการเรียกใช้หน่วยโปรแกรมอื่นของหน่วยโปรแกรม Expression( )

5.4.1.6 Expression( )

หน่วยโปรแกรมนี้ถูกเรียกใช้จากหน่วยโปรแกรม Syntax( ) โดยจะทำหน้าที่ในการตรวจสอบคำถามว่าเป็น Expression ตามหลักไวยากรณ์พีชคณิตสัมพันธ์ที่กล่าวไว้ในบทที่ 2 หรือไม่ โดยที่ Expression อาจเป็นได้หลายอย่าง หน่วยโปรแกรมนี้จึงมีการเรียกใช้หน่วยโปรแกรมอื่นๆ เพื่อตรวจสอบคำถามตามว่าเป็น Expression อย่างใดอย่างหนึ่งหรือไม่ และในแต่ละหน่วยโปรแกรมที่เรียกใช้ จะทำการสร้างคำถามเชิงต้นไม้ด้วย ดังแสดงแผนภาพในรูปที่ 5.12

5.4.1.7 Selection( )

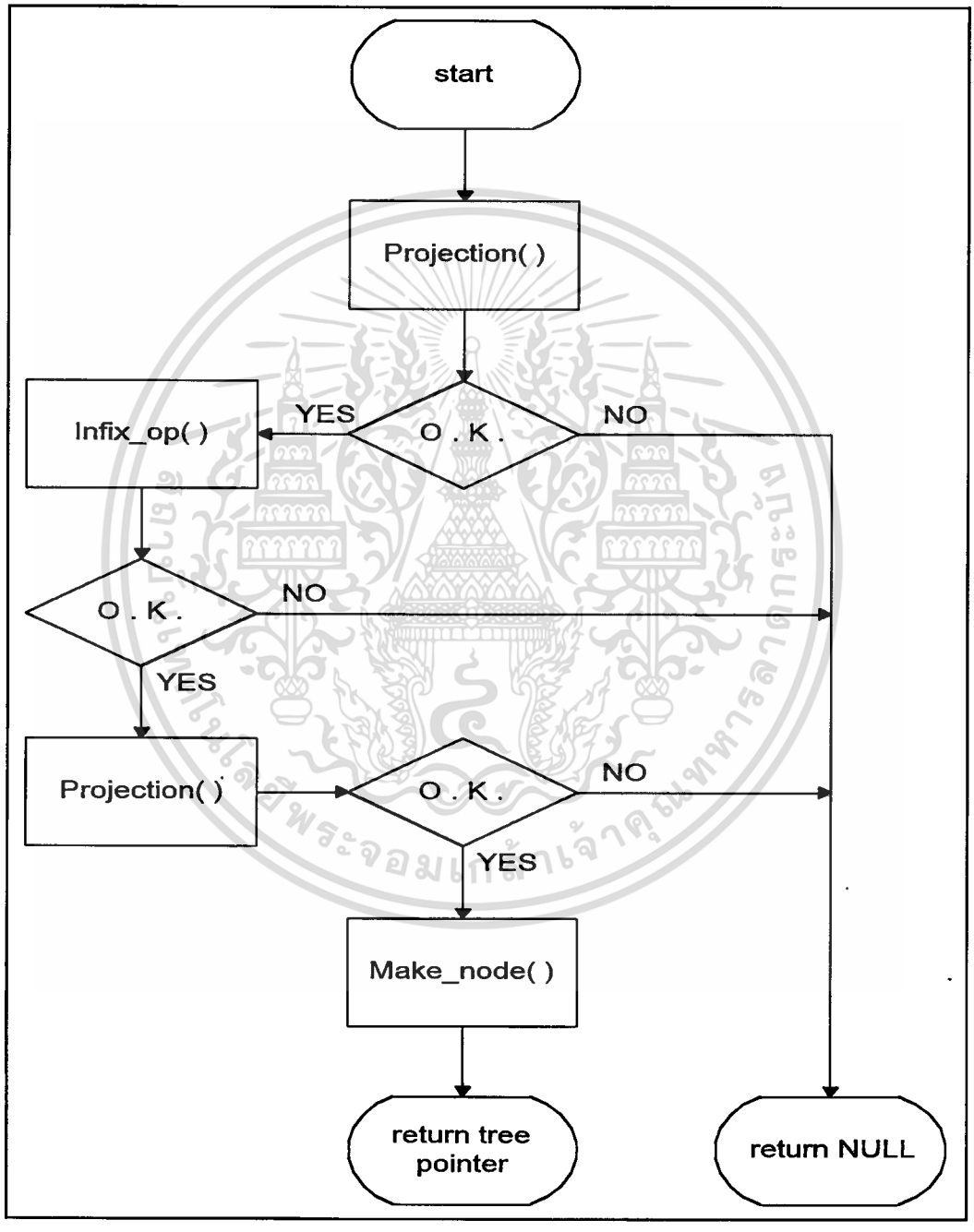
หน่วยโปรแกรมนี้จะทำหน้าที่ตรวจสอบคำถาม ว่าเป็นคำถามแบบ Selection หรือไม่ ถ้าเป็นคำถามแบบ Selection ก็จะสร้างคำถามเชิงต้นไม้เฉพาะในส่วนคำถามที่ทำการตรวจสอบ โดยจะมีการเรียกใช้หน่วยโปรแกรมอื่นประกอบด้วย ดังแสดงด้วยแผนภาพในรูปที่ 5.13



รูปที่ 5.13 แสดงขั้นตอนในการเรียกใช้หน่วยโปรแกรมอื่นของหน่วยโปรแกรม Selection( )

5.4.1.8 Infix_expr( )

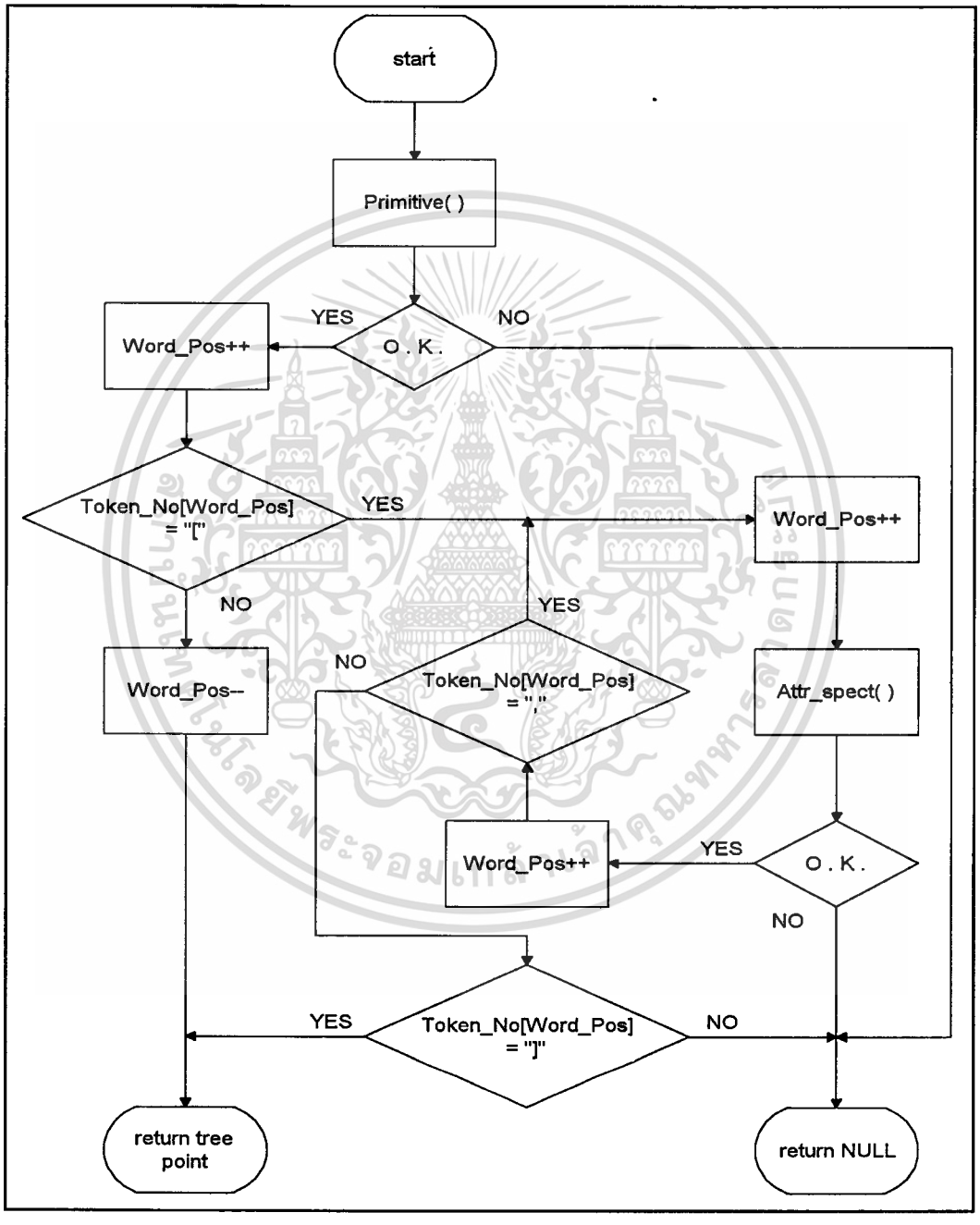
หน่วยโปรแกรมนี้จะทำหน้าที่ตรวจสอบคำถาม ว่าเป็นไปตามกฎ Infix_expr ที่กล่าวไว้ในบทที่ 2 หรือไม่ ถ้าเป็นไปตามกฎก็จะทำการสร้างคำถามเชิงต้นไม้ ซึ่งในการตรวจสอบคำถามจะมีการเรียกใช้หน่วยโปรแกรม Projection( ) ด้วย ดังแสดงไว้ในแผนภาพในรูปที่ 5.14



รูปที่ 5.14 แสดงขั้นตอนในการเรียกใช้หน่วยโปรแกรมอื่นของหน่วยโปรแกรม Infix_expr( )

5.4.1.9 Projection( )

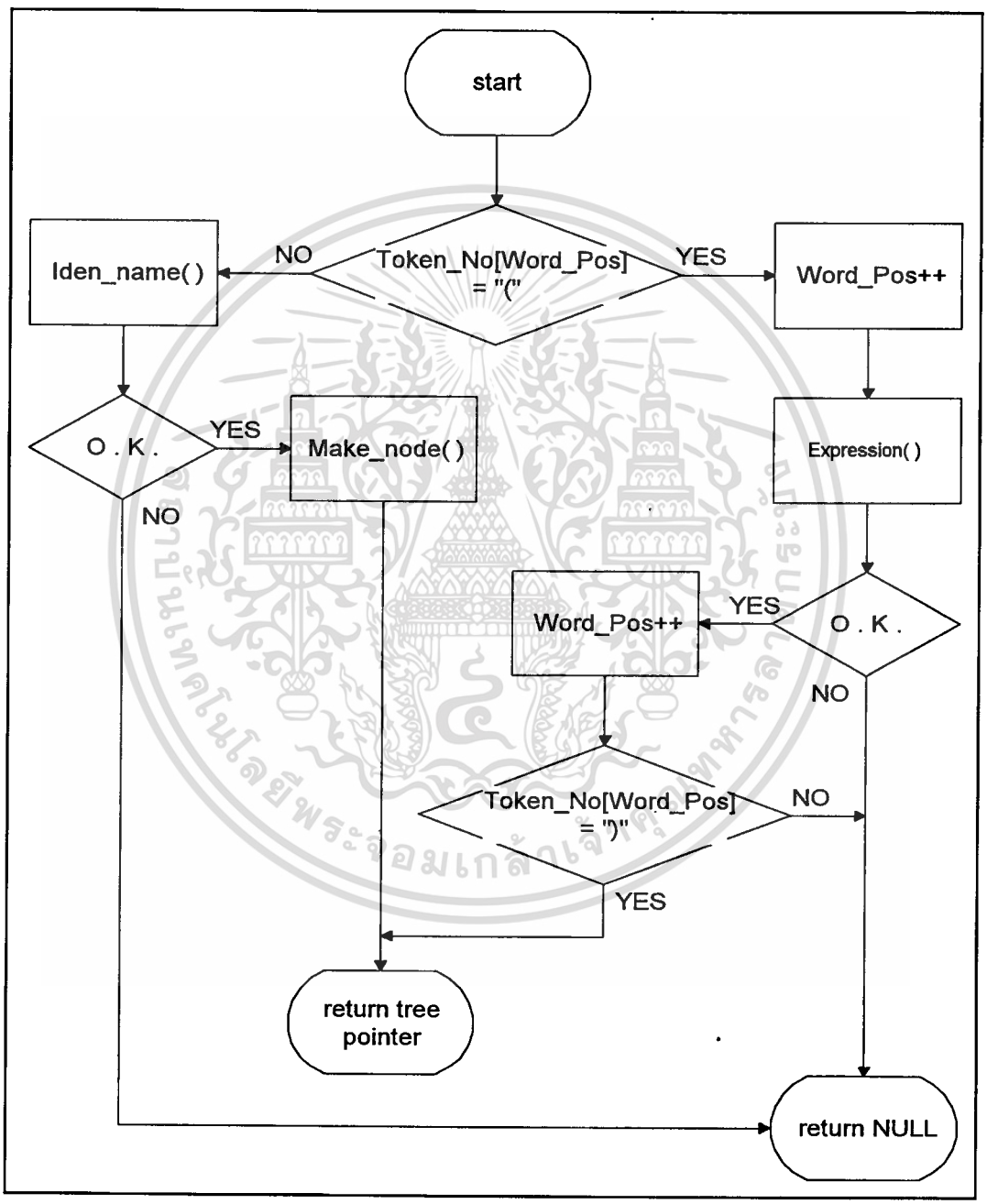
หน่วยโปรแกรมนี้จะทำหน้าที่ตรวจสอบคำถาม ว่าเป็นไปตามกฎ Projection ที่กล่าวไว้ในบทที่ 2 หรือไม่ ถ้าเป็นไปตามกฎที่กล่าวไว้ ก็จะทำการสร้างคำถามเชิงต้นไม้ ในการตรวจสอบคำถาม มีการเรียกใช้หน่วยโปรแกรมอื่นอีก ซึ่งแสดงด้วยแผนภาพในรูปที่ 5.15



รูปที่ 5.15 แสดงขั้นตอนในการเรียกใช้หน่วยโปรแกรมอื่นของหน่วยโปรแกรม Projection( )

5.4.1.10 Primitive( )

หน่วยโปรแกรมนี้ จะทำหน้าที่ในการตรวจสอบส่วนของคำถามว่าเป็น Primitive ตามกฎที่กล่าวไว้ในบทที่ 2 หรือไม่ โดยมีการเรียกใช้หน่วยโปรแกรมอื่นๆ ดังแสดงด้วยแผนภาพในรูปที่ 5.16



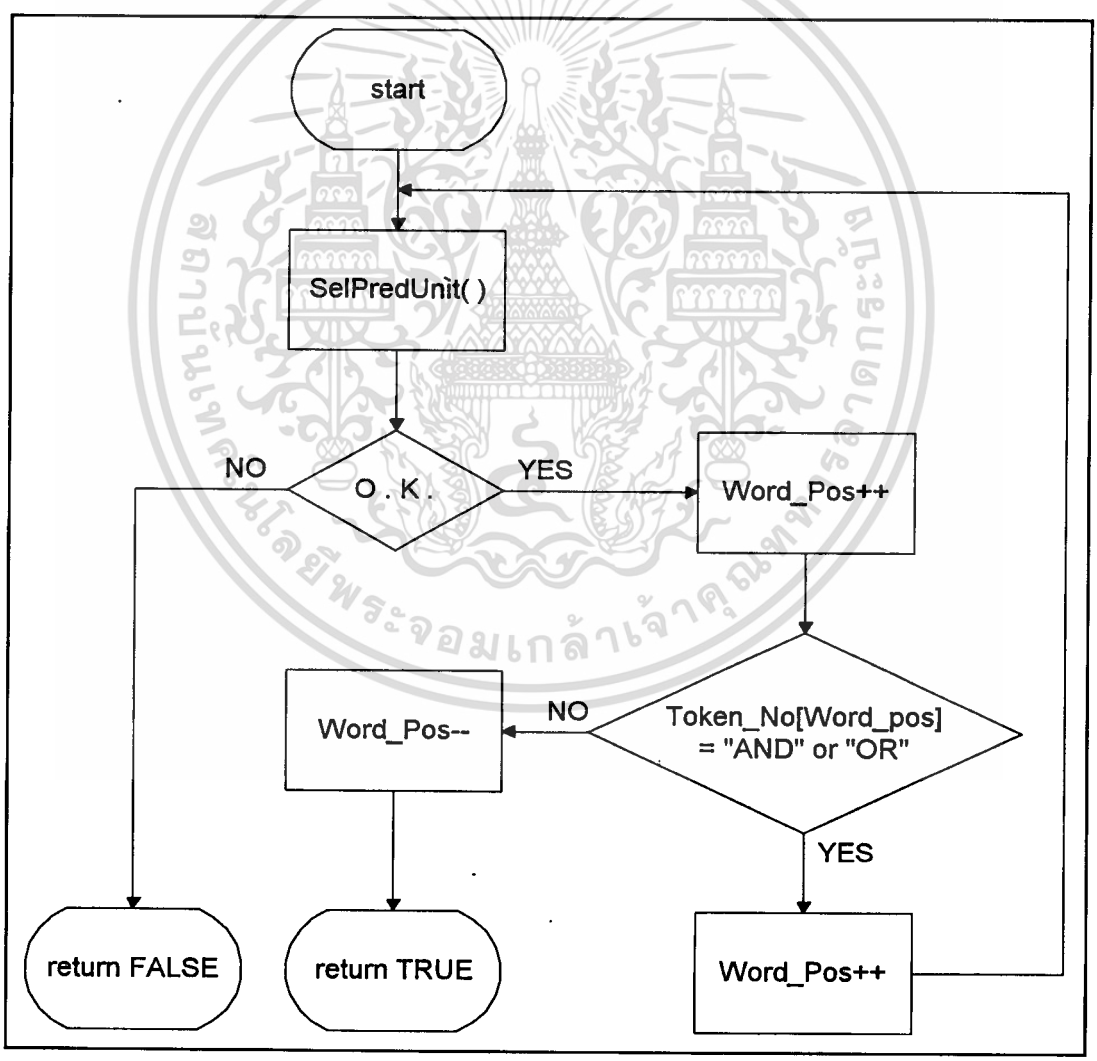
รูปที่ 5.16 แสดงขั้นตอนในการเรียกใช้หน่วยโปรแกรมอื่นของหน่วยโปรแกรม Primitive( )

5.4.1.11 Select_pred(str)

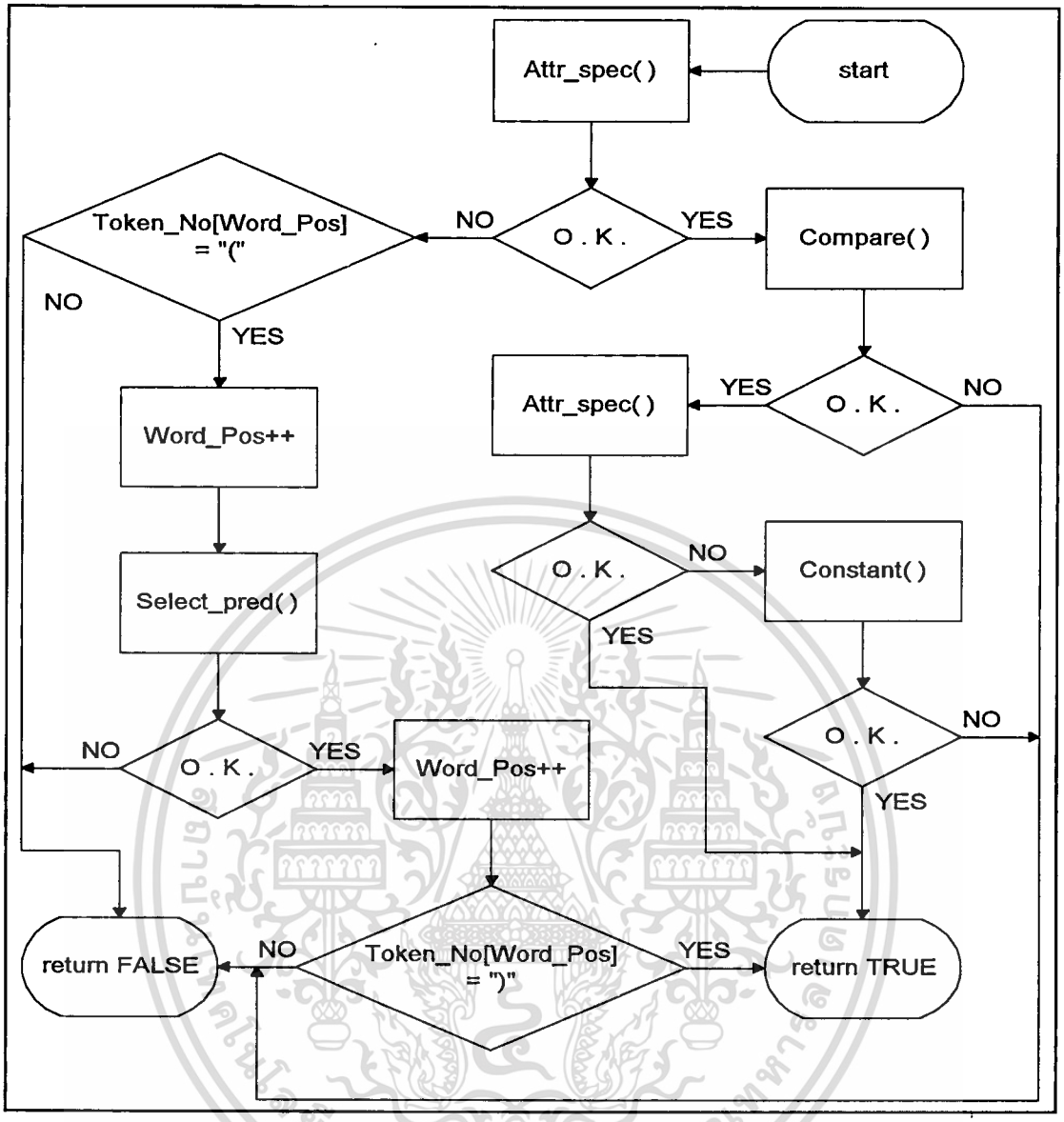
หน่วยโปรแกรมนี้ จะทำหน้าที่ในการตรวจสอบส่วนของคำถามที่เป็นแบบ Selection ว่าเป็นเงื่อนไขในการเลือกข้อมูลหรือไม่ โดยจะมีการเรียกใช้หน่วยโปรแกรม SelPredUnit( ) เพื่อช่วยในการตรวจสอบด้วย ดังแสดงความสัมพันธ์ในรูปที่ 5.17

5.4.1.12 SelPredUnit(str)

หน่วยโปรแกรมนี้จะทำหน้าที่ในการตรวจสอบส่วนของคำถาม ว่าอยู่ในรูปของการเปรียบเทียบหรือไม่ ซึ่งการเปรียบเทียบอาจจะอยู่ในรูปของแอททริบิวเปรียบเทียบกับแอททริบิวหรือค่าคงที่ก็ได้ หรือไม่ก็อาจจะอยู่ในรูปของ Select_perd ที่มีการใส่วงเล็บครอบไว้ดังแสดงขั้นตอนในรูปที่ 5.18



รูปที่ 5.17 แสดงขั้นตอนในการเรียกใช้หน่วยโปรแกรมอื่นของหน่วยโปรแกรม Select_pred( )



รูปที่ 5.18 แสดงขั้นตอนในการเรียกใช้หน่วยโปรแกรมอื่นของหน่วยโปรแกรม SelPredUnit( )

5.4.1.13 Attr_spec(str)

หน่วยโปรแกรมนี้จะทำหน้าที่ในการตรวจสอบส่วนของคำถามว่า เป็นการระบุถึงแอททริบิวหรือไม่ โดยการอ้างถึงแอททริบิวนี้อาจจะเป็นการอ้างถึงชื่อของแอททริบิวโดยตรง หรือ กำหนดชื่อความสัมพันธ์นำหน้า และตามด้วยชื่อของแอททริบิว ซึ่งทั้งสองจะแยกกันด้วยจุด

5.4.1.14 Infix_op( )

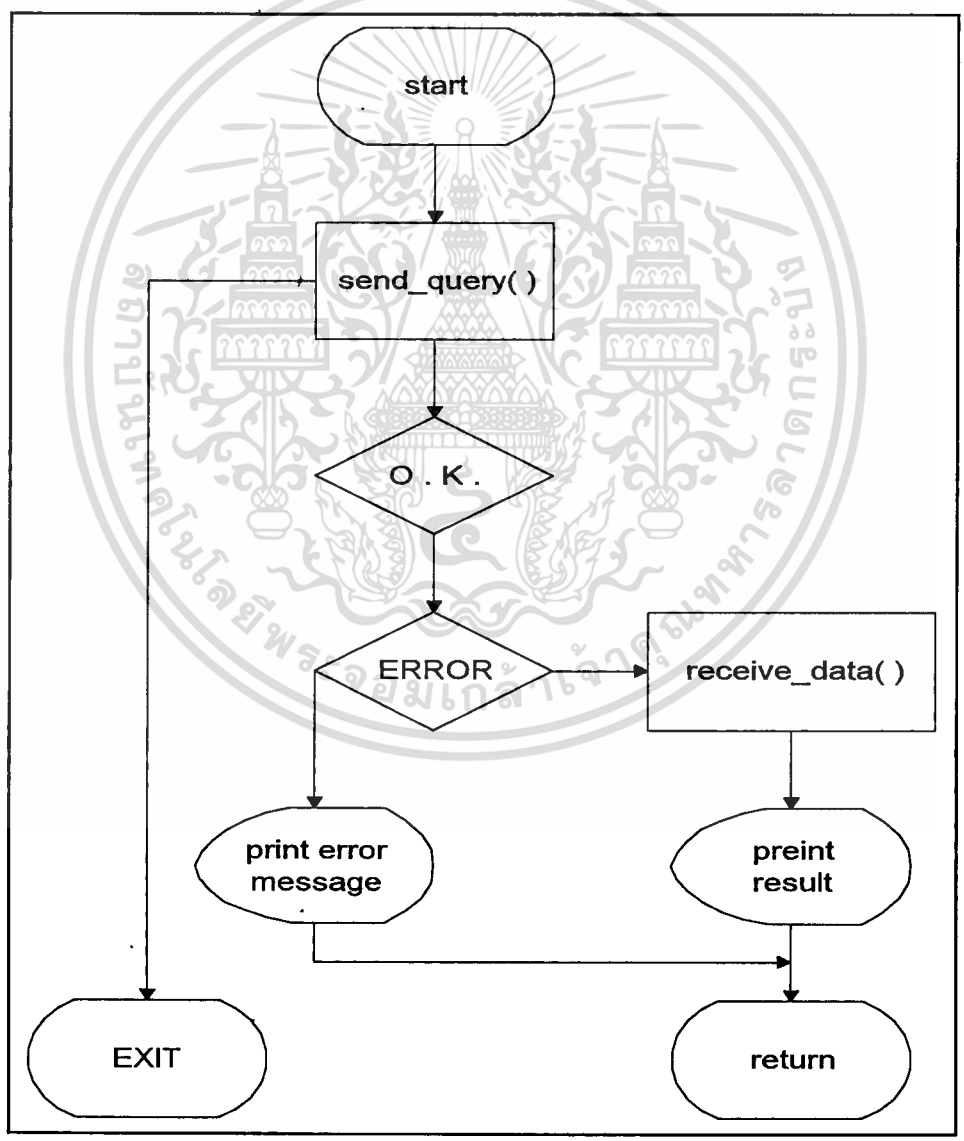
หน่วยโปรแกรมนี้จะทำหน้าที่ในการตรวจสอบส่วนของคำถาม ที่ระบุตำแหน่งของคำด้วยตัวแปร Word_Pos ว่าเป็นตัวปฏิบัติการพีชคณิตสัมพันธ์หรือไม่

5.4.1.15 Iden_name(str)

หน่วยโปรแกรมนี้จะทำหน้าที่ในการตรวจสอบส่วนของคำถาม ที่ระบุตำแหน่งของคำด้วยตัวแปร Word_Pos ว่าเป็นชื่อของความสัมพันธ์หรือไม่ โดยคำที่ทำการพิจารณานั้น จะต้องไม่เป็นตัวปฏิบัติการพีชคณิตสัมพันธ์ และไม่มีตัวเลขนำหน้าคำ

5.4.1.16 Compare(str)

หน่วยโปรแกรมนี้จะทำหน้าที่ในการตรวจสอบ ส่วนของคำถามที่ระบุตำแหน่งของคำด้วยตัวแปร Word_Pos ว่าเป็นเครื่องหมายการเปรียบเทียบหรือไม่ โดยหน่วยโปรแกรมนี้จะถูกเรียกใช้จากหน่วยโปรแกรม SelPredUnit( )



รูปที่ 5.19 แสดงขั้นตอนในการเรียกใช้หน่วยโปรแกรมอื่นของหน่วยโปรแกรม con_back_end( )

5.4.1.17 Constant(str)

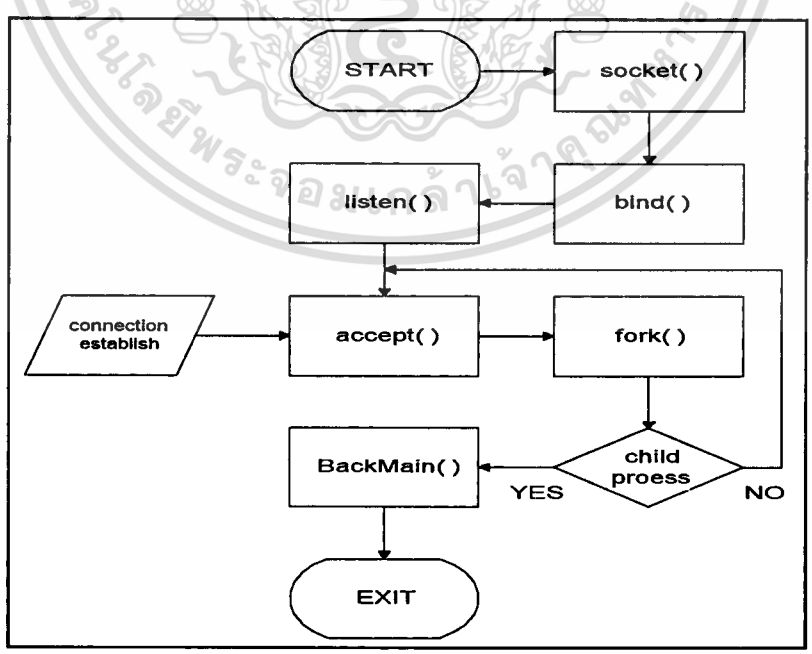
หน่วยโปรแกรมนี้จะทำหน้าที่ในการตรวจสอบ ส่วนของคำถามที่ระบุตำแหน่งของคำ โดยตัวแปร Word_Pos ว่าเป็นค่าคงที่หรือไม่ ซึ่งค่าคงที่นี้จะต้องเป็นตัวเลข หรือข้อความที่อยู่ในเครื่องหมายอัญประกาศ(') เท่านั้น

5.4.1.18 con_back_end(sockfd, top, n_amount)

หน่วยโปรแกรมนี้จะทำหน้าที่ในการส่งคำถามในรูปคำถามเชิงต้นไม้ ที่ได้จากการสร้างและมีการตรวจสอบไวยากรณ์แล้ว ให้กับกระบวนประมวลผลคำถามแบบกระจาย โดยจะมีการเรียกใช้หน่วยโปรแกรม send_query( ) จากส่วนของเครื่องมือโปรแกรม (tools) และนำผลลัพธ์ที่ได้จากกระบวนการประมวลผลคำถามแบบกระจายแสดงผลทางจอภาพ ด้วยการเรียกใช้หน่วยโปรแกรม receive_data( ) จากส่วนของเครื่องมือโปรแกรม ดังแสดงความสัมพันธ์การเรียกใช้หน่วยโปรแกรมอื่นในรูปที่ 5.19

5.4.2 ส่วนประมวลผลคำถามแบบกระจาย (Distributed Query Processing Unit)

หน่วยโปรแกรมในส่วนประมวลผลคำถามแบบกระจายนี้ ประกอบไปด้วยหน่วยโปรแกรมต่างๆมากมาย ซึ่งจะกระจายอยู่ในไฟล์ชื่อ back.c , bk_main.pc , bk_sub1.pc และ bk_sub2.pc ดังจะได้อธิบายหน่วยโปรแกรมต่างๆต่อไปนี้



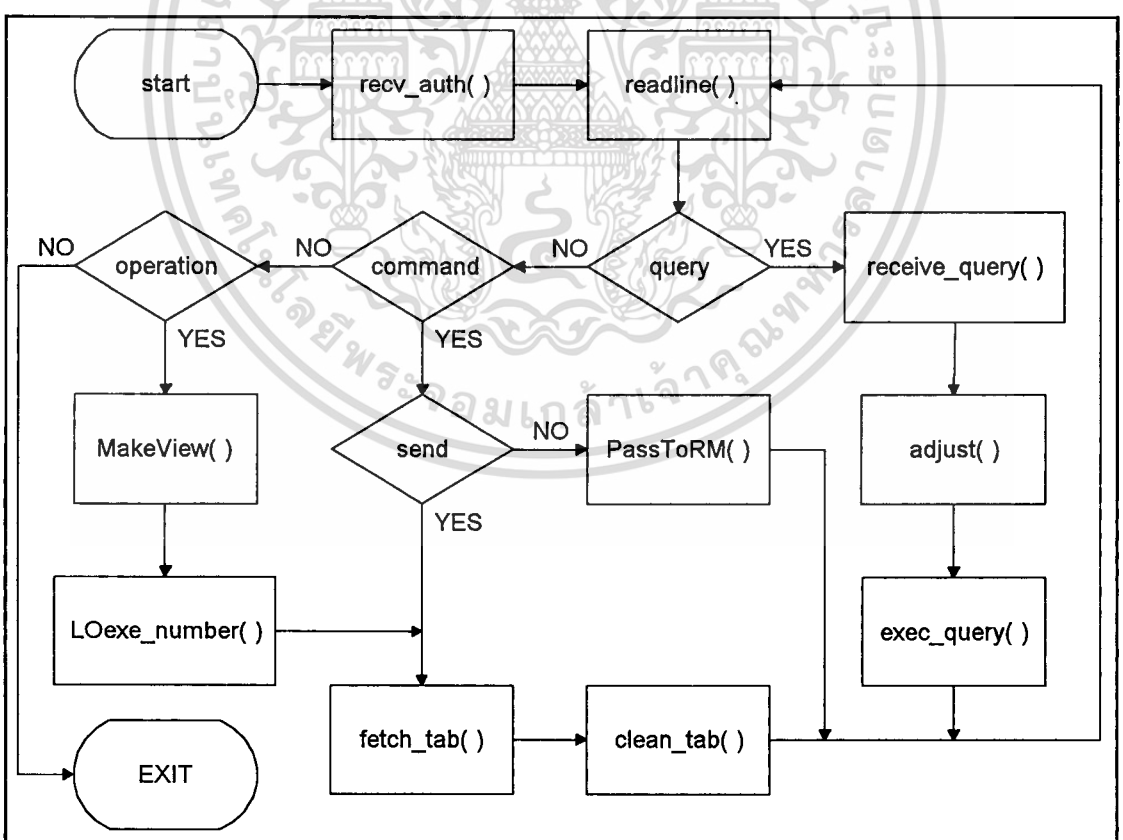
รูปที่ 5.20 แสดงขั้นตอนในการเรียกใช้หน่วยโปรแกรมอื่นของหน่วยโปรแกรม main( )

5.4.2.1 main(argc, argv)

หน่วยโปรแกรมนี้จะเป็นหน่วยโปรแกรมหลักของส่วนประมวลผลคำถามแบบกระจาย ซึ่งจะทำหน้าที่ในการสร้างช่องติดต่อสื่อสาร เพื่อรอรับการติดต่อ (connection) จากกระบวนการประมวลผลอื่น ดังรูปที่ 5.20 เมื่อมีการติดต่อจากกระบวนการประมวลอื่น หน่วยโปรแกรมนี้จะทำการแยก (fork) กระบวนการประมวลผลคำถามแบบกระจายเป็นสองกระบวนการ ในส่วนของกระบวนการที่แยกตัวออกมา(กระบวนการลูก)จะเรียกใช้หน่วยโปรแกรม BackMain( ) ให้ทำการประมวลผลต่อไป ส่วนกระบวนการประมวลผลเดิม(กระบวนการพ่อ) จะกลับไปรอรับการติดต่อจากกระบวนการประมวลผลอื่นต่อไป

5.4.2.2 BackMain(sockfd)

หน่วยโปรแกรมนี้จะทำหน้าที่รับชื่อผู้ใช้งานและรหัสผ่าน จากกระบวนการที่ติดต่อเข้ามาทางซอคเกตที่ระบุค่าด้วยตัวแปร sockfd ถ้าชื่อผู้ใช้งานและรหัสผ่านถูกต้องก็จะวนรอบรับคำถามหรือคำสั่ง จนกว่ากระบวนการที่ติดต่อมาจะหยุดใช้งาน โดยหน่วยโปรแกรมนี้มีการเรียกใช้หน่วยโปรแกรมอื่นๆ ดังแสดงความสัมพันธ์ในรูปที่ 5.21



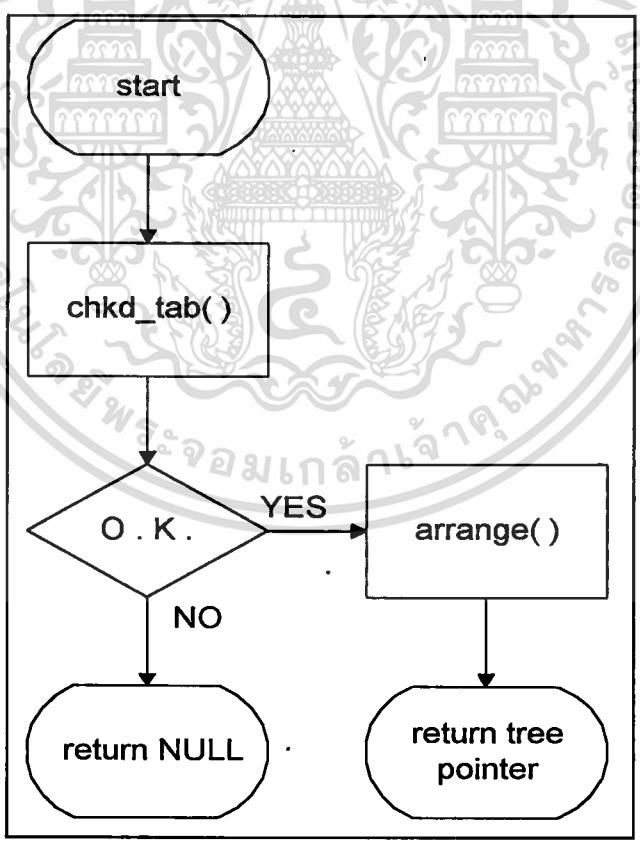
รูปที่ 5.21 แสดงขั้นตอนในการเรียกใช้หน่วยโปรแกรมอื่นของหน่วยโปรแกรม BackMain( )

5.4.2.3 recv_auth(sockfd)

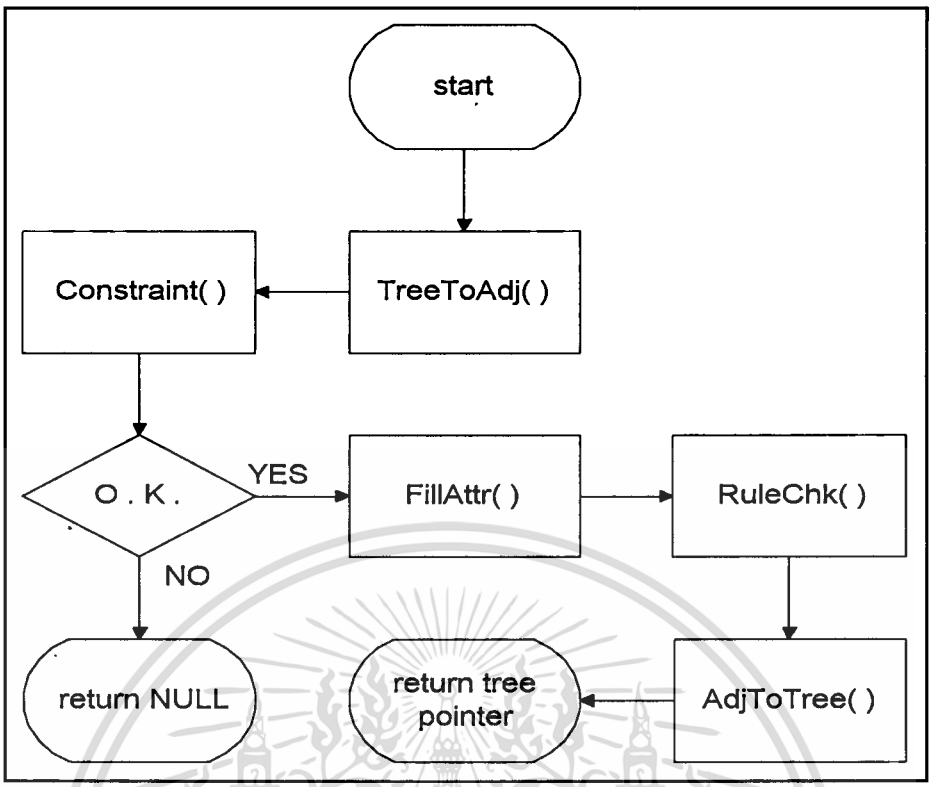
หน่วยโปรแกรมนี้จะเป็นหน่วยโปรแกรมที่ถูกเรียกใช้จากหน่วยโปรแกรม BackMain( ) เพื่อทำหน้าที่ในการรับและตรวจสอบชื่อผู้ใช้งานและรหัสผ่าน ว่าได้รับอนุญาตให้ใช้งานหรือไม่ โดยการรับค่าซอคเกตด้วยตัวแปร sockfd ถ้าถูกต้องก็จะทำการติดต่อกับระบบจัดการฐานข้อมูล เพื่อจะได้เข้าถึงข้อมูลต่อไป

5.4.2.4 adjust(top_ptr)

หน่วยโปรแกรมนี้จะเป็นหน่วยโปรแกรมหลักที่เรียกใช้หน่วยโปรแกรมอื่น เพื่อทำการปรับปรุงรูปร่างคำถามเชิงต้นไม้ ที่ระบุตำแหน่งหน่วยความจำของโนดสูงสุดด้วยตัวแปร top_ptr ให้มีการประมวลผลดีที่สุด (query optimization) และยังตรวจสอบชื่อความสัมพันธ์ที่ถูกอ้างถึงในคำถาม ว่ามีอยู่ในระบบฐานข้อมูลหรือไม่ ก่อนที่จะมีการประมวลผลคำถาม รวมทั้งยังตรวจสอบคำถามว่าเป็นไปตามกฎบังคับความถูกต้องหรือไม่ เพื่อจะได้ลดเวลาในการประมวลผลกรณีที่คำถามหาคำตอบไม่ได้ ซึ่งจะแสดงขั้นตอนในการเรียกใช้หน่วยโปรแกรมอื่นไว้ในรูปที่ 5.22



รูปที่ 5.22 แสดงขั้นตอนในการเรียกใช้หน่วยโปรแกรมอื่นของหน่วยโปรแกรม adjust( )



รูปที่ 5.23 แสดงขั้นตอนในการเรียกใช้หน่วยโปรแกรมอื่นของหน่วยโปรแกรม arrange( )

5.4.2.5 arrange(top_ptr)

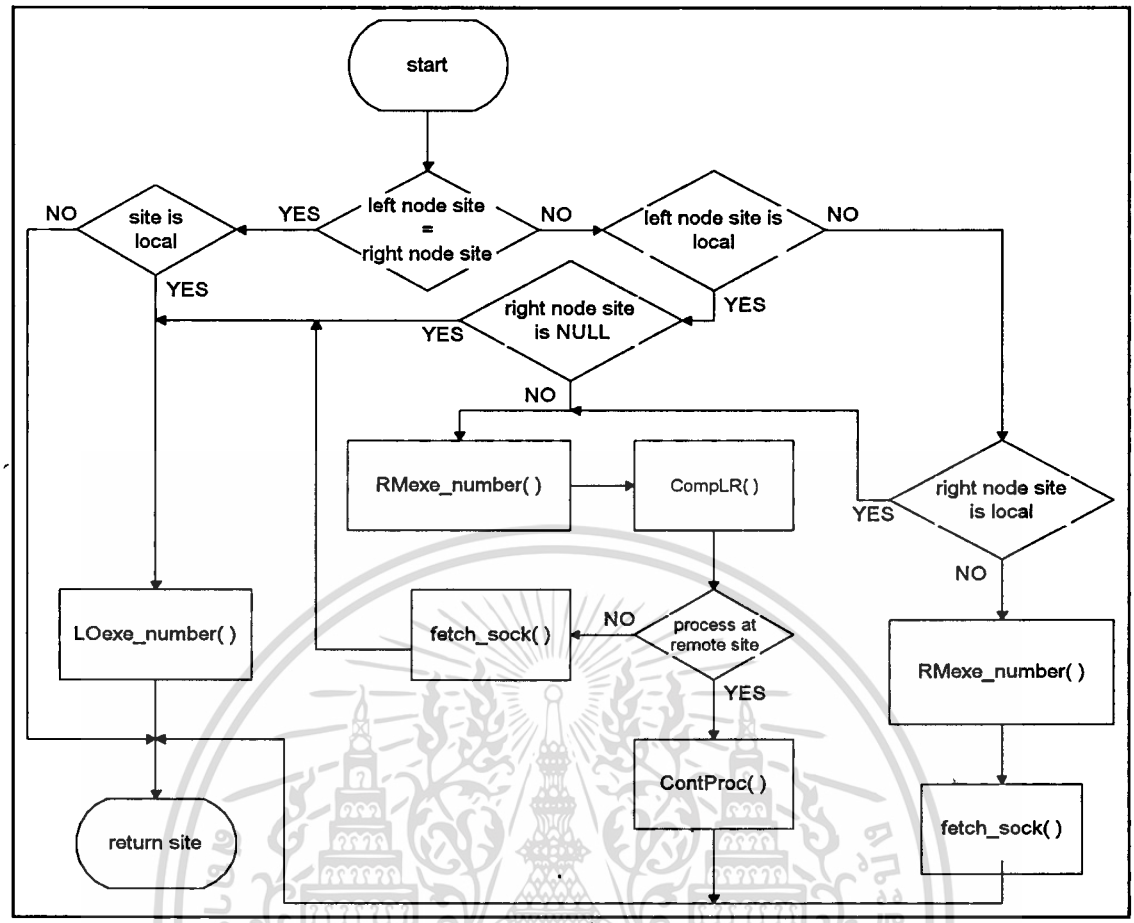
หน่วยโปรแกรมนี้จะทำหน้าที่ในการเรียกใช้หน่วยโปรแกรมอื่นๆ เพื่อทำการตรวจสอบคำถามเชิงต้นไม้ ที่ระบุตำแหน่งหน่วยความจำของโนดสูงสุดด้วยตัวแปร top_ptr ว่าถูกต้องตามกฎบังคับความถูกต้องหรือไม่ หากคำถามไม่ขัดกับกฎบังคับความถูกต้อง ก็จะทำการปรับปรุงรูปของคำถามเชิงต้นไม้ต่อไป ดังแสดงขั้นตอนการเรียกใช้หน่วยโปรแกรมอื่นไว้ในรูปที่ 5.23

5.4.2.6 clean_tab( )

หน่วยโปรแกรมนี้จะทำหน้าที่ในการลบความสัมพันธ์ชั่วคราวทั้งหมดที่สร้างขึ้นเพื่อช่วยในการประมวลผลคำถามเชิงต้นไม้ เพื่อจะได้ประมวลผลคำถามใหม่ต่อไป

5.4.2.7 exec_query(top_ptr, tuples_ptr)

หน่วยโปรแกรมนี้จะทำหน้าที่ในการประมวลผลคำถามเชิงต้นไม้ ที่ระบุตำแหน่งหน่วยความจำของโนดสูงสุดด้วยตัวแปร top_ptr ซึ่งได้ผ่านการปรับปรุงรูปร่างแล้วให้ได้ผลลัพธ์ โดยจะมีการเรียกใช้หน่วยโปรแกรมอื่นในการประมวลผลด้วย ดังแสดงความสัมพันธ์การเรียกใช้ในรูปที่ 5.24 และผลลัพธ์ที่ได้จะเก็บไว้ในหน่วยความจำที่ระบุตำแหน่งโดยตัวแปร tuples



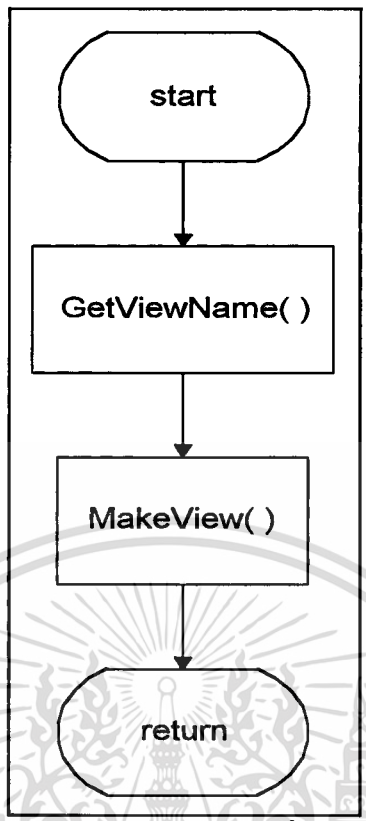
รูปที่ 5.24 แสดงขั้นตอนในการเรียกใช้หน่วยโปรแกรมอื่นของหน่วยโปรแกรม exec_query( )

5.4.2.8 RMexe_number(top, sockfd_ptr,hostname)

หน่วยโปรแกรมนี้ จะทำการติดต่อกับกระบวนการประมวลผลคำถามแบบกระจายในคอมพิวเตอร์เครื่องอื่นที่กำหนดชื่อด้วยตัวแปร hostname เพื่อที่จะส่งคำถามเชิงต้นไม้ที่ระบุตำแหน่งหน่วยความจำของโนดสูงสุดด้วยตัวแปร top ไปทำการประมวลผล โดยใช้ซอคเกตที่ระบุค่าด้วยตัวแปร sockfd ซึ่งผลลัพธ์จากคำถามที่ส่งไปประมวลผล จะเป็นจำนวนทัปเปิลของข้อมูลผลลัพธ์

5.4.2.9 fetch_sock(top_ptr,sockfd)

หน่วยโปรแกรมนี้จะทำหน้าที่ในการรับข้อมูลผลลัพธ์ จากซอคเกตที่อ้างด้วยตัวแปร sockfd ที่ได้จากการส่งคำถามเชิงต้นไม้ของหน่วยโปรแกรม RMexe_number( )ไปประมวลผลในสถานที่อื่น เพื่อเก็บไว้ในตารางข้อมูลชั่วคราวสำหรับทำการประมวลผลต่อ หรือส่งผลลัพธ์ในกับส่วนที่ส่งคำถามมา ซึ่งตารางข้อมูลชั่วคราวนี้จะระบุตำแหน่งหน่วยความจำด้วยตัวแปร top_ptr



รูปที่ 5.25 แสดงขั้นตอนในการเรียกใช้หน่วยโปรแกรมอื่นของหน่วยโปรแกรม fetch_sock( )

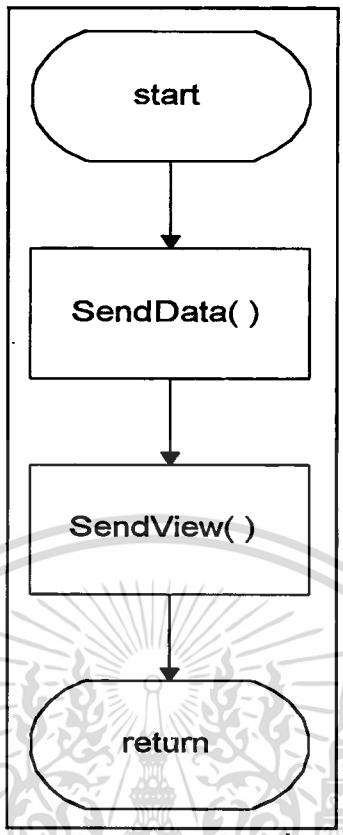
ในกรณีที่คำถามมีการประมวลผลต่อ จะมีการแก้ไขคำถาม เพื่อให้คำถามมีการอ้างถึงตารางข้อมูลผลลัพธ์นี้ ซึ่งการทำงานของหน่วยโปรแกรมนี้ จะมีการเรียกใช้หน่วยโปรแกรมอื่น ดังแสดงไว้ในรูปที่ 5.25

5.4.2.10 fetch_tab(table,sockfd)

หน่วยโปรแกรมนี้จะทำหน้าที่ในการนำผลลัพธ์ที่ได้จากการประมวลผลคำถามเชิงต้นไม้ ซึ่งเก็บไว้ในตารางชั่วคราวที่ระบุชื่อด้วยตัวแปร table ทำการส่งให้กับส่วนที่ส่งคำถามมา โดยผ่านทางซอคเกต sockfd พร้อมทั้งรายละเอียดในการประมวลผล สำหรับใช้ในการประมวลผลในกรณีที่ส่วนที่ส่งคำถามมาต้องการทำการประมวลต่อไป ซึ่งหน่วยโปรแกรมนี้จะมีการเรียกใช้หน่วยโปรแกรม SendData( ) และ SendView( ) ให้ทำงานตามลำดับ ดังแสดงในรูปที่ 5.26

5.4.2.11 LOexe_number(table)

หน่วยโปรแกรมนี้จะทำหน้าที่ในการนับจำนวนทัปเปิลข้อมูล ของความสัมพันธ์ในฐานข้อมูลที่ระบุชื่อด้วยตัวแปร table เพื่อใช้ประกอบการตัดสินใจ ในการกำหนดสถานที่ประมวลผลในกรณีที่มีการปฏิบัติการระหว่างความสัมพันธ์ที่อยู่ต่างสถานที่กัน



รูปที่ 5.26 แสดงขั้นตอนในการเรียกใช้หน่วยโปรแกรมอื่นของหน่วยโปรแกรม fetch_tab( )

5.4.2.12 LOexe_node(top_ptr)

หน่วยโปรแกรมนี้จะทำการประมวลผลส่วนของคำถามเชิงต้นไม้ ที่ระบุตำแหน่งหน่วยความจำของโนดสูงสุดด้วยตัวแปร top_ptr ซึ่งเป็นการปฏิบัติการระหว่างความสัมพันธ์ภายในฐานข้อมูลท้องถิ่นเดียวกัน และจะเก็บผลลัพธ์ที่ได้ไว้ในตารางข้อมูลชั่วคราว

5.4.2.13 CompLR(LNumber,LTable,oper,RNumber,RTable,LocalSite)

ในกรณีที่มีการปฏิบัติการระหว่างความสัมพันธ์ที่อยู่ต่างสถานที่กัน หน่วยโปรแกรมนี้จะทำหน้าที่ตัดสินใจกำหนดสถานที่ในการประมวลผล ว่าจะกระทำการปฏิบัติการที่สถานที่ใด ซึ่งจะพิจารณาจากค่าต่างๆ คือ LNumber แทนจำนวนทัปเปิลของความสัมพันธ์ที่อยู่ด้านซ้ายของตัวปฏิบัติการพีชคณิตสัมพันธ์ , RNumber แทนจำนวนทัปเปิลของความสัมพันธ์ที่อยู่ด้านขวาของตัวปฏิบัติการพีชคณิตสัมพันธ์ , LTable แทนชื่อความสัมพันธ์ของโนดที่อยู่ด้านซ้าย , RTable แทนชื่อความสัมพันธ์ของโนดที่อยู่ด้านขวา , oper จะบอกชนิดของตัวปฏิบัติการพีชคณิตสัมพันธ์ ส่วน LocalSite จะบอกถึงโนดที่เก็บความสัมพันธ์ท้องถิ่น ว่าเป็นโนดซ้ายหรือขวา ซึ่งจะใช้หลักการตัดสินใจแต่ละตัวปฏิบัติการพีชคณิตดังนี้

ตัวปฏิบัติการ MINUS จะพิจารณาว่าความสัมพันธ์ท้องถิ่นอยู่ในโนดด้านซ้ายหรือขวา กรณีที่ความสัมพันธ์ท้องถิ่นอยู่ด้านซ้าย ถ้าจำนวนทัปเปิลของความสัมพันธ์ที่อยู่โนดด้านขวา น้อยกว่าหรือเท่ากับ 1.5 เท่าของความสัมพันธ์ท้องถิ่น ก็จะตัดสินใจประมวลผลที่สถานที่ท้องถิ่น ไม่เช่นนั้นก็จะตัดสินใจประมวลผลที่สถานที่อื่น ถ้าความสัมพันธ์ท้องถิ่นอยู่ที่โนดด้านขวาก็จะทำการตัดสินใจประมวลผลที่สถานที่ท้องถิ่น

ตัวปฏิบัติการ INTERSECT จะพิจารณาว่าความสัมพันธ์ที่อยู่ในสถานที่อื่น มากกว่า 1.5 เท่าของความสัมพันธ์ท้องถิ่นหรือไม่ ถ้ามากว่าก็จะประมวลผลที่สถานที่อื่น ไม่เช่นนั้นก็จะประมวลผลที่สถานที่ท้องถิ่น

ตัวปฏบัติการ DIVIDEBY จะตัดสินใจประมวลผลในสถานที่ ของความสัมพันธ์ที่อยู่ในโนดด้านซ้าย

ส่วนตัวปฏิบัติการ UNION , TIMES , JOIN และตัวปฏิบัติการพีชคณิตอื่นๆที่เหลือ ก็จะทำการประมวลผลที่สถานที่ท้องถิ่น

สำหรับค่า 1.5 เท่านั้น เป็นค่าที่ทดลองกำหนดขึ้นเท่านั้น ซึ่งอาจจะเปลี่ยนแปลงได้ตามสภาพแวดล้อมของการประมวลผลและฮาร์ดแวร์ที่ใช้

5.4.2.14 GetViewName( )

หน่วยโปรแกรมนี้ จะทำหน้าที่ในการกำหนดชื่อตารางข้อมูลชั่วคราว ที่ใช้ในการประมวลผลคำถามเชิงต้นไม้

5.4.2.15 GetTabName( )

หน่วยโปรแกรมนี้ จะทำหน้าที่ในการกำหนดชื่อของตารางข้อมูลชั่วคราว ที่ใช้ในการเก็บข้อมูล ที่รับมาจากฐานข้อมูลอื่น เพื่อใช้ในการประมวลผลต่อไป

5.4.2.16 GetNewAttr(table)

หน่วยโปรแกรมนี้ จะทำหน้าที่ในการกำหนดชื่อใหม่ ให้กับแอททริบิวของความสัมพันธ์ที่ระบุถึงด้วยตัวแปร table เพื่อให้การอ้างถึงแอททริบิวในการประมวลผลคำถามเชิงต้นไม้ เป็นไปอย่างถูกต้อง ในกรณีที่ความสัมพันธ์ทั้งสองที่ปฏิบัติการร่วมกัน มีชื่อแอททริบิวเหมือนกัน ซึ่งการกำหนดชื่อใหม่นี้ กระทำโดยการนำชื่อความสัมพันธ์และชื่อแอททริบิวมารวมกัน

5.4.2.17 GetTabAttr(table)

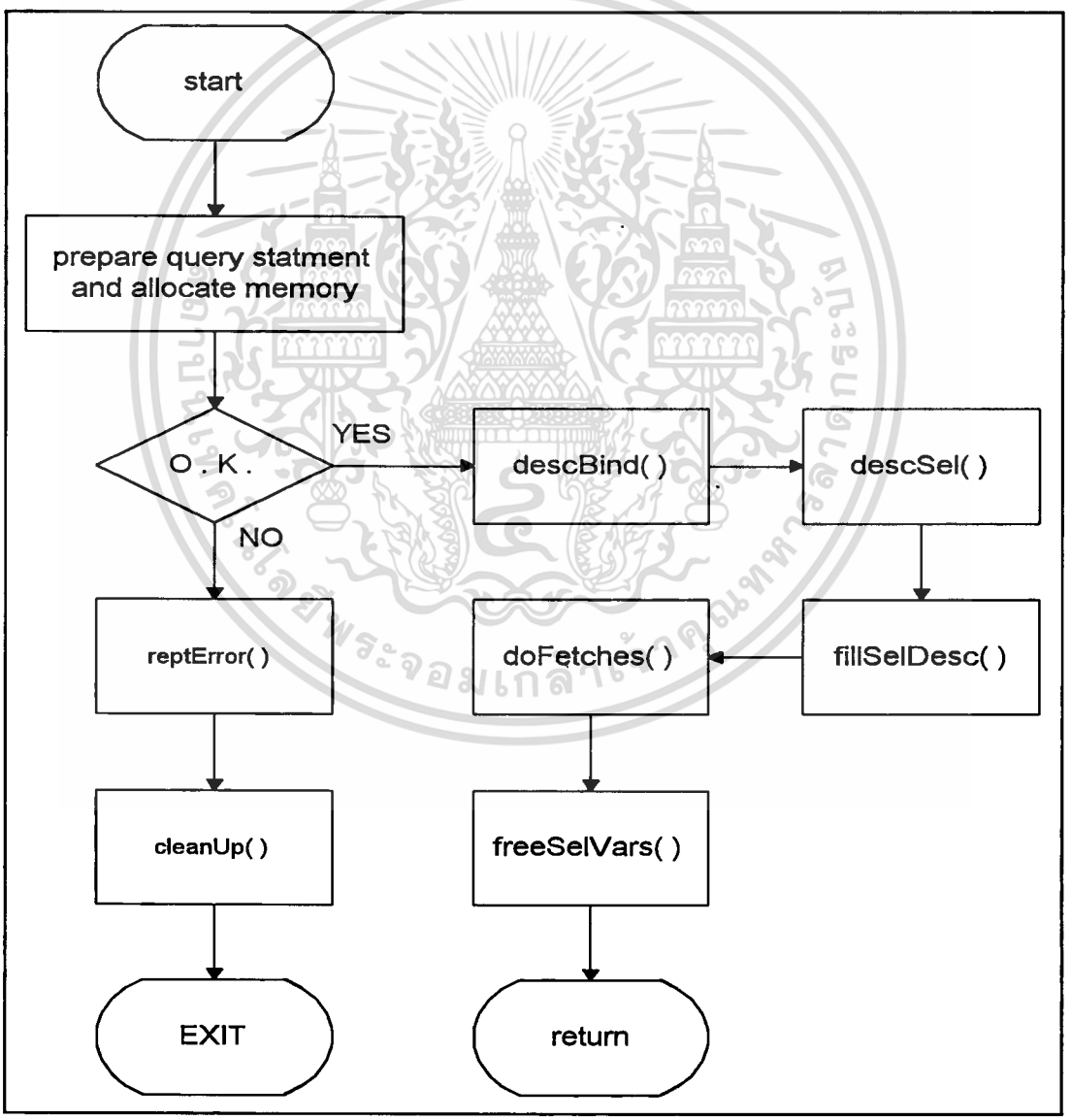
หน่วยโปรแกรมนี้จะทำการค้นหาชื่อแอททริบิวทั้งหมด ของความสัมพันธ์ที่ถูกอ้างถึงด้วยตัวแปร table จากพจนานุกรมฐานข้อมูล

5.4.2.18 GetAttr(viewname)

หน่วยโปรแกรมนี้จะทำหน้าที่ในการค้นหาชื่อแอททริบิว ที่ใช้ในการประมวลผลของความสัมพันธ์ชั่วคราวที่ระบุถึงด้วยตัวแปร viewname โดยจะค้นหาจากส่วนเก็บรายละเอียดการประมวลผล

5.4.2.19 LexTabAttr(tab,attr,str)

หน่วยโปรแกรมนี้จะทำหน้าที่แยกชื่อแอททริบิว ที่เก็บอยู่ในตัวแปร str ซึ่งกำหนดขึ้นจากการรวมกันของชื่อความสัมพันธ์และชื่อแอททริบิวเดิม ให้เป็นชื่อความสัมพันธ์และชื่อแอททริบิวแยกจากกันดังเดิม โดยจะเก็บไว้ในตัวแปร tab และ attr ตามลำดับ



รูปที่ 5.27 แสดงขั้นตอนในการเรียกใช้หน่วยโปรแกรมอื่นของหน่วยโปรแกรม SendData( )

5.4.2.20 SendData(table,sockfd)

หน่วยโปรแกรมนี้ ถูกพัฒนาด้วยวิธีการฝังคำสั่งภาษาสอบถามเชิงโครงสร้างแบบจลน์ ซึ่งจะมีหน้าที่ในการนำข้อมูลในความสัมพันธ์ที่เป็นผลลัพธ์ ซึ่งมีชื่อกำหนดไว้ในตัวแปร table ส่งไปยังส่วนที่ส่งคำถามมา ผ่านทางซอคเกต sockfd โดยจะมีการเรียกใช้หน่วยโปรแกรมต่างๆ ดังแสดงไว้ในรูปที่ 5.27

5.4.2.21 mufwrn( )

หน่วยโปรแกรมนี้จะถูกเรียกใช้หลังจากมีการประมวลผลคำสั่งภาษาสอบถามเชิงโครงสร้าง เพื่อทำการตรวจสอบว่ามีการเตือน (warning) ถึงข้อผิดพลาดที่อาจจะเกิดจากการเรียกใช้คำสั่งภาษาสอบถามเชิงโครงสร้างหรือไม่

5.4.2.22 cleanUp( )

หน่วยโปรแกรมนี้จะถูกเรียกใช้ในหน่วยโปรแกรม SendData( ) เพื่อทำหน้าที่ในการคืนหน่วยความจำให้กับระบบปฏิบัติการ หลังจากที่ไม่มีการใช้งานแล้ว

5.4.2.23 descBind( )

หน่วยโปรแกรมนี้ จะเป็นการเรียกใช้คำสั่งภาษาสอบถามเชิงโครงสร้างแบบจลน์ โดยจะทำหน้าที่ในการวิเคราะห์คำถาม และจองยังหน่วยความจำใหม่ในกรณีที่หน่วยความจำที่จองไว้แล้วไม่พอใช้สำหรับการวิเคราะห์คำถาม

5.4.2.24 descSel( )

หน่วยโปรแกรมนี้ จะทำหน้าที่ในการเตรียมหน่วยความจำสำหรับคำตอบ ที่เกิดจากการประมวลผลคำถามของคำสั่งภาษาสอบถามเชิงโครงสร้างแบบจลน์

5.4.2.25 doFetches(sockfd)

หน่วยโปรแกรมนี้ ทำหน้าที่ในการดึงข้อมูลผลลัพธ์ทีละทัปเปิล ส่งไปยังส่วนที่ส่งคำถามมา ในลักษณะการวนรอบ จนกระทั่งครบทุกทัปเปิล โดยจะส่งผ่านทางซอคเกต sockfd

5.4.2.26 fillSelDesc( )

หน่วยโปรแกรมนี้ จะทำหน้าที่ในการกำหนดชนิดของข้อมูล (data type) ของผลลัพธ์ที่จะเกิดขึ้นจากการประมวลผลคำถามเชิงต้นไม้

5.4.2.27 freeSelVars( )

หน่วยโปรแกรมนี้จะทำหน้าที่ในการคืนหน่วยความจำ ที่ใช้เก็บค่าต่างๆในการรับข้อมูลผลลัพธ์ ให้กับระบบปฏิบัติการ

5.4.2.28 reptError( )

หน่วยโปรแกรมนี้จะถูกเรียกใช้เมื่อพบว่า การเรียกใช้คำสั่งภาษาสอบถามเชิงโครงสร้าง ที่ฝังตัวอยู่ในโปรแกรมเกิดข้อผิดพลาดขึ้น โดยจะแสดงข้อความของข้อผิดพลาดทางจอภาพ

5.4.2.29 min(c1,c2)

หน่วยโปรแกรมนี้จะทำการเปรียบเทียบค่า c1 และ c2 เพื่อเลือกจำนวนที่มีค่าน้อยกว่า

5.4.2.30 max(c1,c2)

หน่วยโปรแกรมนี้จะทำการเปรียบเทียบค่า c1 และ c2 เพื่อเลือกจำนวนที่มีค่ามากกว่า

5.4.2.31 MakeView(viewname,sockfd)

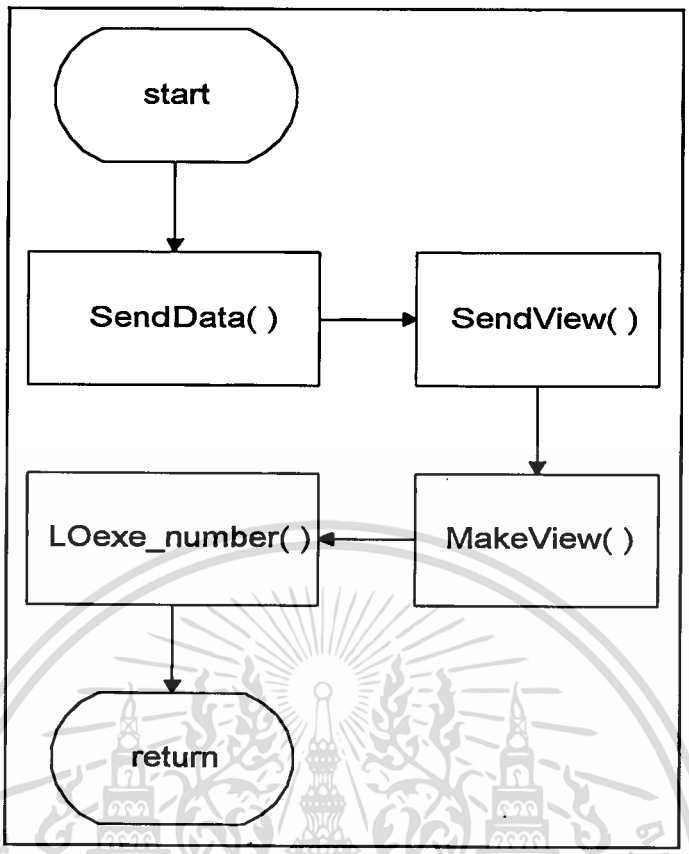
หน่วยโปรแกรมนี้จะทำหน้าที่ในการรับข้อมูลผลลัพธ์ และรายละเอียดในการประมวลผลทางซอคเกต sockfd เพื่อทำการเก็บไว้ในตารางข้อมูลชั่วคราวด้วยชื่อที่เก็บไว้ในตัวแปร viewname สำหรับใช้ในการประมวลผลต่อไป

5.4.2.32 ContProc(sockfd,top_ptr,local)

ในกรณีที่มีการปฏิบัติการพีชคณิตสัมพันธ์ ระหว่างความสัมพันธ์ที่อยู่ต่างสถานที่กันและพิจารณาแล้วว่า จะทำการประมวลผลที่สถานที่อื่น (remote site) หน่วยโปรแกรมนี้ จะทำการส่งตัวปฏิบัติการพีชคณิตสัมพันธ์ ที่ระบุตำแหน่งหน่วยความจำด้วยตัวแปร top_ptr และข้อมูลพร้อมรายละเอียดการประมวลผล ไปทำการประมวลผลต่อในสถานที่อื่น โดยผ่านซอคเกต sockfd ด้วยการเรียกใช้หน่วยโปรแกรมอื่นๆ ดังแสดงไว้ในรูปที่ 5.28 ส่วนตัวแปร local จะบอกว่าความสัมพันธ์ท้องถิ่น อยู่ในโนดด้านซ้ายหรือขวาของตัวปฏิบัติการพีชคณิตสัมพันธ์

5.4.2.33 PassToRM(fd1, fd2)

หน่วยโปรแกรมนี้จะถูกเรียกใช้ให้ทำหน้าที่ในการผ่านข้อมูลผลลัพธ์ จากสถานที่ประมวลผลที่สถานที่อื่นด้วยซอคเกต fd2 ส่งให้กับส่วนที่ส่งคำถามมาทางซอคเกต fd1 ทั้งนี้เนื่องจากคำถามที่รับมานั้นเป็นการเข้าถึงข้อมูลที่สถานที่อื่น จึงส่งคำถามไปประมวลผลยังสถานที่เก็บข้อมูล



รูปที่ 5.28 แสดงขั้นตอนในการเรียกใช้หน่วยโปรแกรมอื่นของหน่วยโปรแกรม ContProc( )

5.4.2.34 chkd_tab(top_ptr)

หน่วยโปรแกรมนี้จะทำหน้าที่ในการตรวจสอบความสัมพันธ์ที่อยู่ในโนด ที่ระบุตำแหน่งหน่วยความจำด้วยตัวแปร top_ptr ว่ามีอยู่ในระบบฐานข้อมูลหรือไม่ จะได้ไม่ต้องเสียเวลาประมวลผล ในกรณีที่ความสัมพันธ์ที่ระบุถึงไม่มีในฐานข้อมูล

5.4.2.35 TreeToAdj(top_ptr)

หน่วยโปรแกรมนี้จะทำหน้าที่ในการเปลี่ยนโครงสร้างข้อมูล (data structure) ของคำถามเชิงต้นไม้ ที่ระบุตำแหน่งหน่วยความจำด้วยตัวแปร top_ptr จากชนิด TREE เป็นชนิด ADJ เพื่อจะใช้ในขั้นตอนของการปรับปรุงรูปคำถามเชิงต้นไม้ และตรวจสอบกับกฎบังคับความถูกต้อง

5.4.2.36 AdjToTree(adj_ptr)

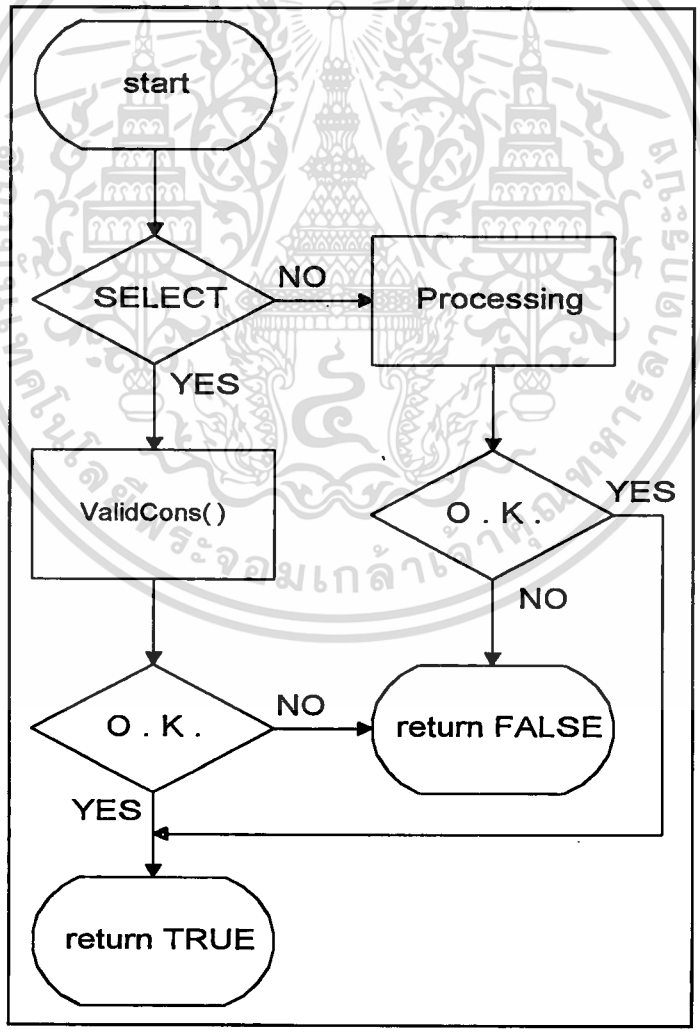
หน่วยโปรแกรมนี้จะทำหน้าที่ในการเปลี่ยนโครงสร้างข้อมูล (data structure) จากชนิด ADJ ที่ระบุตำแหน่งหน่วยความจำด้วยตัวแปร adj_ptr ให้เป็นโครงสร้างข้อมูลชนิด TREE เพื่อทำการประมวลผลหาผลลัพธ์ต่อไป

5.4.2.37 Constraint(adj_ptr)

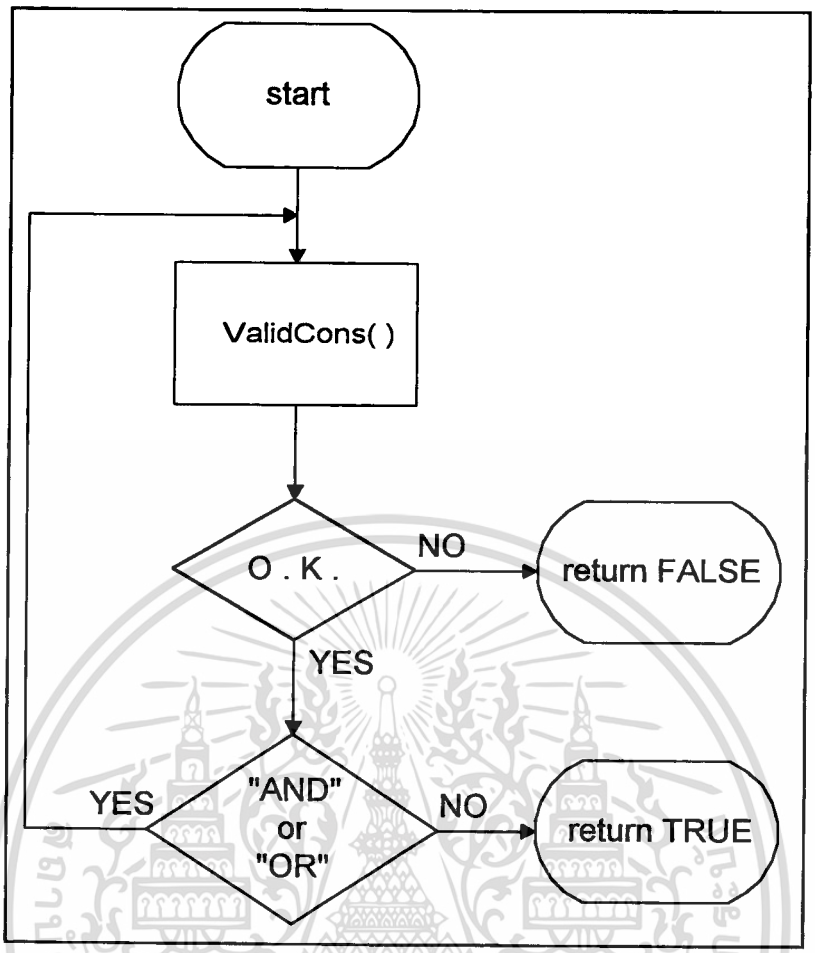
หน่วยโปรแกรมนี้ จะทำการตรวจสอบคำถามเชิงต้นไม้ที่มีโครงสร้างข้อมูลเป็นชนิด ADJ ที่ระบุตำแหน่งหน่วยความจำของโนดสูงสุดด้วยตัวแปร adj_ptr ด้วยกฎบังคับความถูกต้อง (domain constraint) ว่าข้อมูลที่ต้องการขัดกับกฎบังคับความถูกต้องหรือไม่ ถ้าคำถามขัดกับกฎบังคับความถูกต้อง แสดงว่าไม่มีข้อมูลที่ต้องการ การประมวลผลก็จะหยุด ซึ่งหน่วยโปรแกรมนี้จะมีการเรียกใช้หน่วยโปรแกรมอื่น ดังแสดงความสัมพันธ์ในการเรียกใช้ในรูปที่ 5.29

5.4.2.38 ValidCons(item)

หน่วยโปรแกรมนี้จะทำการตรวจสอบเงื่อนไขที่ใช้ในการเลือกข้อมูลทั้งหมด ที่เก็บไว้ในตัวแปร item ว่าถูกต้องตามกฎบังคับความถูกต้องหรือไม่ โดยจะมีการเรียกใช้หน่วยโปรแกรม ValidUnit( ) ในการตรวจสอบแต่ละเงื่อนไข ดังแสดงไว้ในรูปที่ 5.30



รูปที่ 5.29 แสดงขั้นตอนในการเรียกใช้หน่วยโปรแกรมอื่นของหน่วยโปรแกรม Constraint( )



รูปที่ 5.30 แสดงขั้นตอนในการเรียกใช้หน่วยโปรแกรมอื่นของหน่วยโปรแกรม ValidCons( )

5.4.2.39 ValidUnit(item)

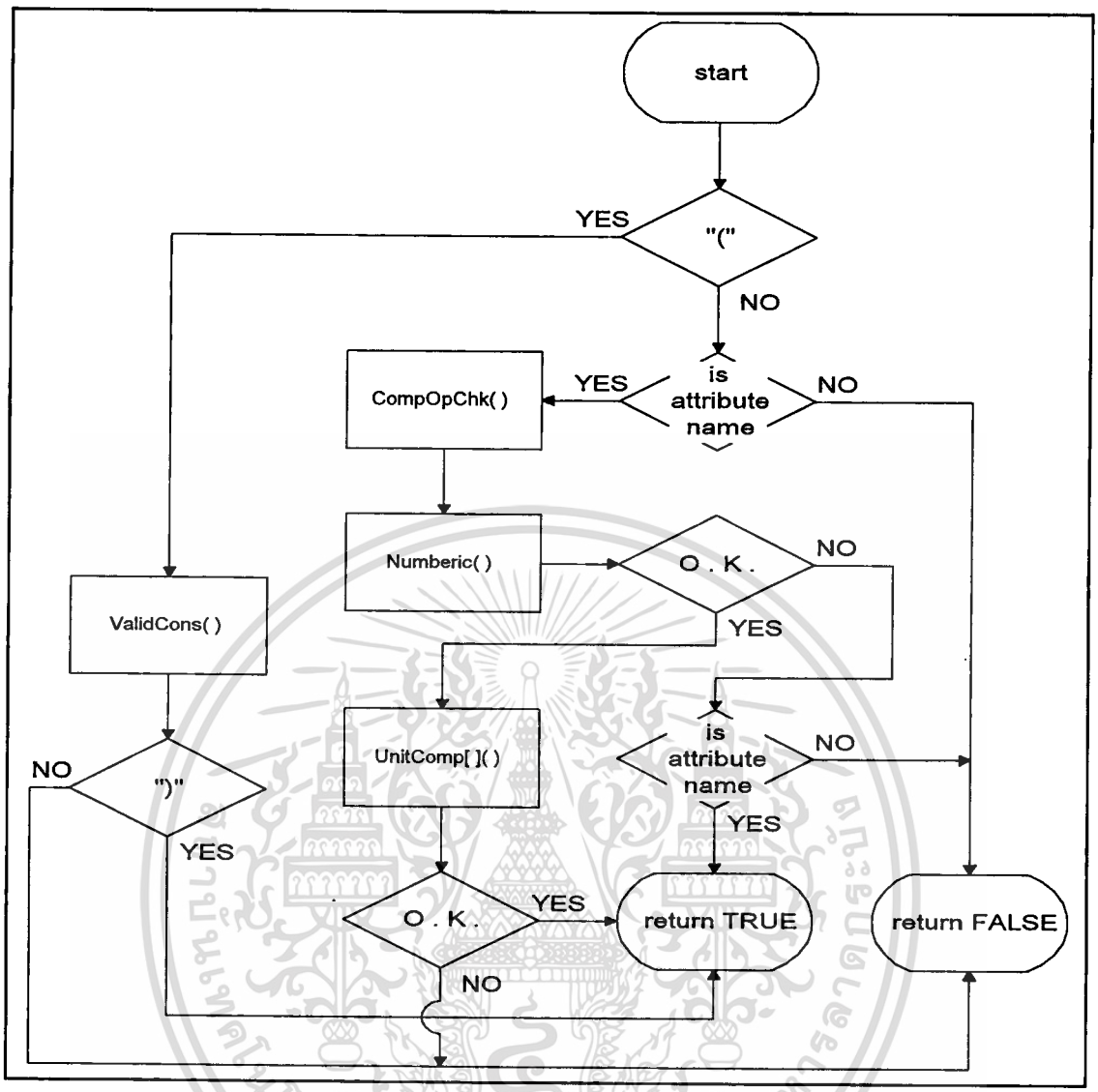
หน่วยโปรแกรมนี้ จะทำการตรวจสอบเงื่อนไขในการเลือกแต่ละเงื่อนไข ที่ถูกเก็บไว้ในตัวแปร item ว่าถูกต้องตามกฎบังคับความถูกต้องหรือไม่ โดยจะมีการเรียกใช้หน่วยโปรแกรมอื่นประกอบการประมวลผลด้วย ดังแสดงในรูป 5.31

5.4.2.40 CompOpChk(str)

หน่วยโปรแกรมนี้ จะทำหน้าที่ในการตรวจสอบส่วนของคำถามที่เก็บตัวในตัวแปร str ว่าเป็นเครื่องหมายการเปรียบเทียบหรือไม่

5.4.2.41 Numberic(str)

หน่วยโปรแกรมนี้ จะทำหน้าที่ในการตรวจสอบส่วนของคำถามที่เก็บอยู่ในตัวแปร str ว่าเป็นตัวเลขหรือไม่



รูปที่ 5.31 แสดงขั้นตอนในการเรียกใช้หน่วยโปรแกรมอื่นของหน่วยโปรแกรม ValidUnit( )

5.4.2.42 EquCons( )

หน่วยโปรแกรมนี้ จะทำหน้าที่ในการตรวจสอบเงื่อนไขการเข้าถึงข้อมูล ของตัวปฏิบัติการ selection ที่มีการเปรียบเทียบค่าด้วยเครื่องหมายเท่ากัน ว่าถูกต้องตามกฎบังคับความถูกต้องหรือไม่

5.4.2.43 NotEquCons( )

หน่วยโปรแกรมนี้ จะทำหน้าที่ในการตรวจสอบเงื่อนไขการเข้าถึงข้อมูล ของตัวปฏิบัติการ selection ที่มีการเปรียบเทียบค่าด้วยเครื่องหมายไม่เท่ากัน ว่าถูกต้องตามกฎบังคับความถูกต้องหรือไม่

5.4.2.44 LessCons( )

หน่วยโปรแกรมนี้ จะทำหน้าที่ในการตรวจสอบเงื่อนไขการเข้าถึงข้อมูล ของตัวปฏิบัติการ selection ที่มีการเปรียบเทียบค่าด้วยเครื่องหมายน้อยกว่า ว่าถูกต้องตามกฎบังคับความถูกต้องหรือไม่

5.4.2.45 MoreCons( )

หน่วยโปรแกรมนี้ จะทำหน้าที่ในการตรวจสอบเงื่อนไขการเข้าถึงข้อมูล ของตัวปฏิบัติการ selection ที่มีการเปรียบเทียบค่าด้วยเครื่องหมายมากกว่า ว่าถูกต้องตามกฎบังคับความถูกต้องหรือไม่

5.4.2.46 LessEquCons( )

หน่วยโปรแกรมนี้ จะทำหน้าที่ในการตรวจสอบเงื่อนไขการเข้าถึงข้อมูล ของตัวปฏิบัติการ selection ที่มีการเปรียบเทียบค่าด้วยเครื่องหมายน้อยกว่าหรือเท่ากับ ว่าถูกต้องตามกฎบังคับความถูกต้องหรือไม่

5.4.2.47 MoreEquCons( )

หน่วยโปรแกรมนี้ จะทำหน้าที่ในการตรวจสอบเงื่อนไขการเข้าถึงข้อมูล ของตัวปฏิบัติการ selection ที่มีการเปรียบเทียบค่าด้วยเครื่องหมายมากกว่าหรือเท่ากับ ว่าถูกต้องตามกฎบังคับความถูกต้องหรือไม่

5.4.2.48 RuleChk(top)

หน่วยโปรแกรมนี้ จะทำหน้าที่ในการปรับปรุงรูปคำถามเชิงต้นไม้ ที่ระบุตำแหน่งหน่วยความจำด้วยตัวแปร top โดยจะทำการตรวจสอบลักษณะของคำถามด้วยกฎการปรับปรุงรูปร่างตามที่กล่าวไว้ในบทที่ 3 ว่าสามารถที่จะปรับปรุงรูปร่างได้หรือไม่ ถ้าสามารถปรับปรุงคำถามได้ก็ทำการปรับปรุง

5.4.2.49 FillAttr(adj_ptr)

หน่วยโปรแกรมนี้เป็นหน่วยโปรแกรมหนึ่ง ที่ถูกเรียกใช้ในขั้นตอนของการเปลี่ยนโครงสร้างข้อมูล จากโครงสร้างข้อมูลชนิด TREE เป็นชนิด ADJ โดยหน่วยโปรแกรมนี้ จะมีหน้าที่ในการกำหนดชื่อแอททริบิวให้กับตัวแปร adj_ptr ซึ่งเป็นตัวแปรโครงสร้างข้อมูลชนิด ADJ

5.4.2.50 Free_Adj(top)

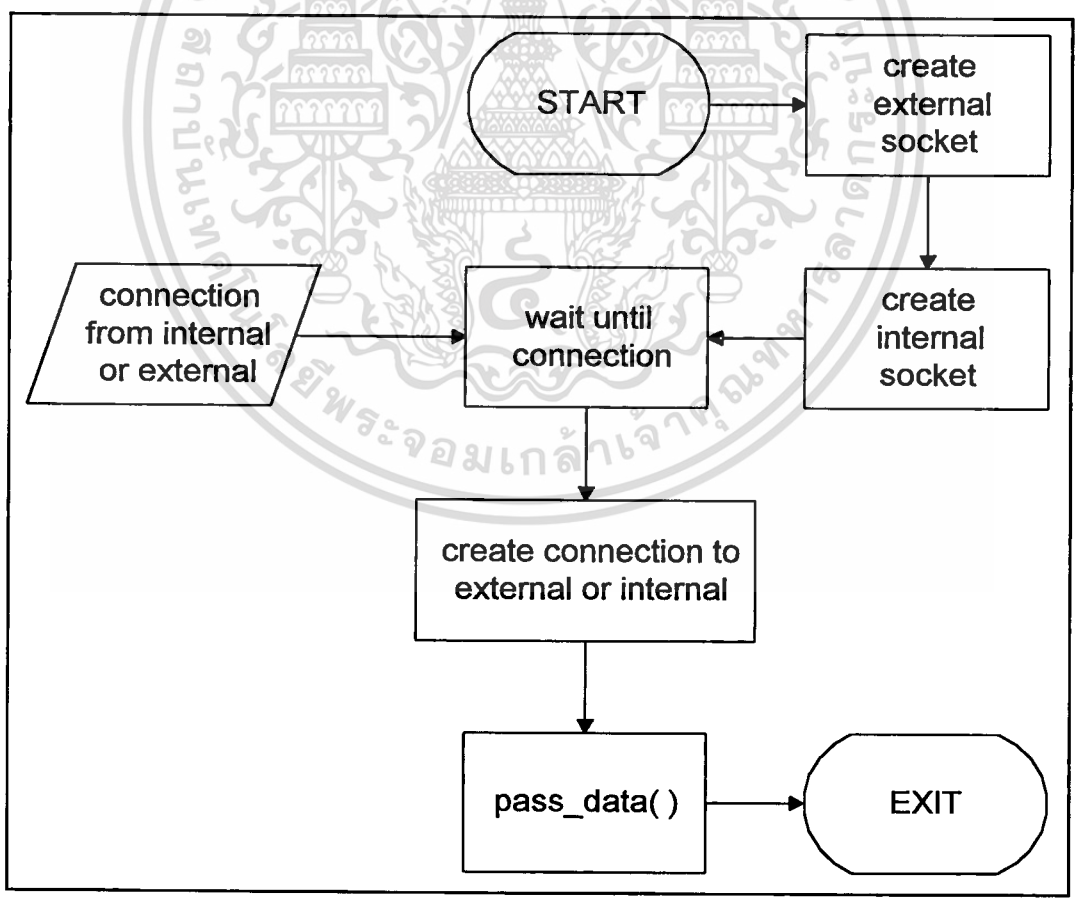
หน่วยโปรแกรมนี้ จะทำหน้าที่ในการคืนหน่วยความจำที่ใช้เก็บข้อมูลชนิดโครงสร้างข้อมูล ADJ ในตำแหน่งที่ระบุด้วยตัวแปร top ให้กับระบบปฏิบัติการ

5.4.2.51 LexAdj(word,words)

หน่วยโปรแกรมนี้ใช้ในการแยกคำในตำแหน่งแรก ออกจากกลุ่มคำที่ถูกเก็บไว้ในตัวแปร words ซึ่งคำเหล่านี้จะแยกกันด้วยช่องว่างหรือเครื่องหมายจุลภาคคั่นระหว่างคำ คำที่แยกออกมาได้จะเก็บไว้ในตัวแปร word

5.4.3 ส่วนจัดการส่งข้อมูลผ่านโครงข่ายสื่อสาร (Network Interface)

หน่วยโปรแกรมส่วนนี้ จะประกอบไปด้วยหน่วยโปรแกรม 2 หน่วย ที่ทำหน้าที่ในการรับส่งข้อมูลต่างๆ ระหว่างโครงข่ายสื่อสาร ซึ่งทั้งสองหน่วยโปรแกรมจะอยู่ภายในไฟล์ชื่อ network.c เพียงไฟล์เดียวเท่านั้น ในแต่ละหน่วยโปรแกรมอธิบายได้ดังนี้



รูปที่ 5.32 แสดงขั้นตอนในการเรียกใช้หน่วยโปรแกรมอื่นของหน่วยโปรแกรม main( )

5.4.3.1 main(argc,argv)

หน่วยโปรแกรมนี้ จะถูกพัฒนาให้ทำงานในลักษณะกระบวนประมวลผลบริการ ซึ่งจะทำหน้าที่ในการรับการติดต่อจากกระบวนการประมวลผลภายนอก หรือจากกระบวนการประมวลผลภายในเครื่องคอมพิวเตอร์เดียวกัน เพื่อทำหน้าที่ในการสร้างการติดต่อกับกระบวนการประมวลผลภายใน หรือกระบวนการประมวลผลภายนอกตามลำดับ หลังจากที่ได้รับการติดต่อจากกระบวนการประมวลผลหนึ่งและสร้างการติดต่อไปยังอีกกระบวนการประมวลผลหนึ่งแล้ว ก็จะทำการเรียกหน่วยโปรแกรม pass_data( ) ให้ทำงาน ดังแสดงความสัมพันธ์ของหน่วยโปรแกรมในรูปที่ 5.32

5.4.3.2 pass_data(fd1,fd2)

หน่วยโปรแกรมนี้จะถูกเรียกใช้จากหน่วยโปรแกรม main( ) หลังจากที่ได้รับการติดต่อและสร้างการติดต่อแล้ว ดังนั้นจึงมีซอคเกต 2 ซอคเกต คือ fd1 และ fd2 หน่วยโปรแกรมนี้จะมีหน้าที่คอยเฝ้าสังเกตว่า มีข้อมูลมาจากซอคเกตใดซอคเกตหนึ่งหรือไม่ หากมีก็จะรับข้อมูลและส่งข้อมูลไปยังอีกซอคเกตหนึ่ง

5.4.4 ส่วนสร้างตารางพจนานุกรมฐานข้อมูล

ส่วนสร้างตารางพจนานุกรมฐานข้อมูลนี้ จะมีหน่วยโปรแกรมเพียงหน่วยโปรแกรมเดียว คือ main( ) ซึ่งจะถูกเขียนไว้ในไฟล์ mak_tab.pc และจะใช้ซอฟท์แวร์ Pro*C ทำการแปลภาษาเบื้องต้น เป็นไฟล์ mak_tab.c จากนั้นจึงทำการแปลภาษาด้วยตัวแปลภาษาซีต่อไป

หน่วยโปรแกรม main( ) นี้ จะทำหน้าที่ในการสร้างตารางพจนานุกรมฐานข้อมูลท้องถิ่นและตารางพจนานุกรมโดยรวม ซึ่งจะมีตารางดังต่อไปนี้ คือ GTHU , GTA , GA , GC , LTHU , LTA , LA และ LC

5.4.5 ส่วนป้อนข้อมูลพจนานุกรมฐานข้อมูล

ในส่วนของการป้อนข้อมูลเข้าสู่พจนานุกรมฐานข้อมูลนั้น จะประกอบด้วย 2 ส่วนด้วยกันคือ ส่วนป้อนข้อมูลเข้าสู่พจนานุกรมฐานข้อมูลท้องถิ่น และส่วนป้อนข้อมูลเข้าสู่พจนานุกรมฐานข้อมูลโดยรวม ซึ่งจะได้แยกอธิบายเป็นหัวข้อดังนี้ คือ

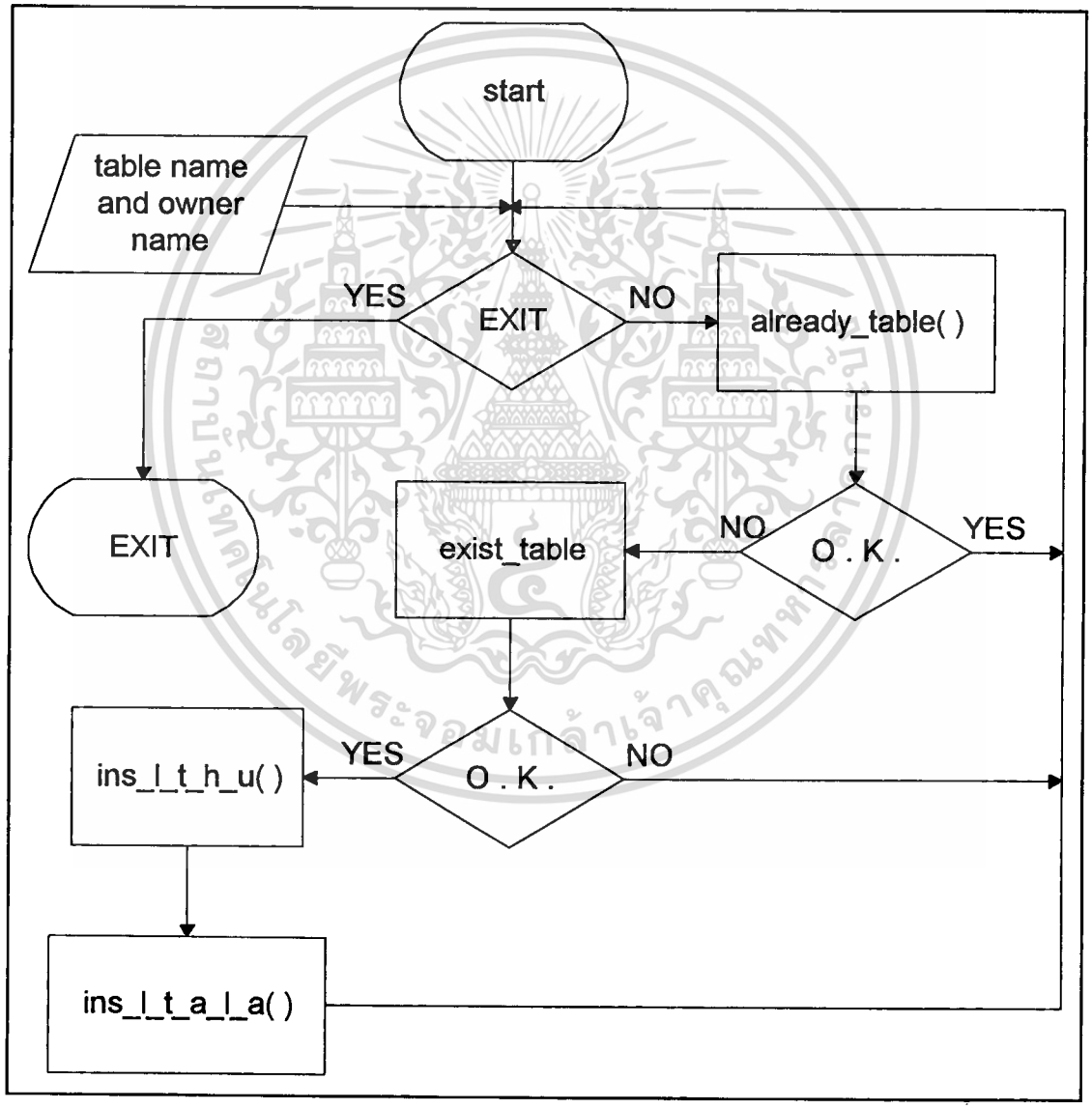
5.4.5.1 ส่วนป้อนข้อมูลพจนานุกรมฐานข้อมูลท้องถิ่น

ส่วนป้อนข้อมูลพจนานุกรมฐานข้อมูลท้องถิ่นนี้ จะทำหน้าที่ในการรับชื่อความสัมพันธ์และชื่อเจ้าของความสัมพันธ์ จากนั้นก็จะนำข้อมูลทั้งสองไปทำการหารายละเอียดต่างๆเพิ่มเติม

จากพจนานุกรมฐานข้อมูลของระบบจัดการฐานข้อมูล แล้วนำข้อมูลที่ได้ป้อนเข้าสู่พจนานุกรมฐานข้อมูลท้องถิ่น ซึ่งส่วนนี้จะมีหน่วยโปรแกรมต่างๆทำงานประสานกัน ดังนี้คือ

5.4.5.1.1 main( )

หน่วยโปรแกรมส่วนนี้จะทำงานในลักษณะวนรอบ เพื่อทำหน้าที่รับชื่อความสัมพันธ์และชื่อเจ้าของความสัมพันธ์ จากนั้นจะเรียกใช้หน่วยโปรแกรมอื่นๆให้ทำงานต่อไป ดังแสดงความสัมพันธ์ในรูปที่ 5.33 การทำงานในลักษณะวนรอบนี้ จะหยุดทำงานเมื่อได้รับชื่อความสัมพันธ์เป็น EXIT



รูปที่ 5.33 แสดงขั้นตอนในการเรียกใช้หน่วยโปรแกรมอื่นของหน่วยโปรแกรม main( )

5.4.5.1.2 already_table( )

หน่วยโปรแกรมนี้ จะทำหน้าที่ในการตรวจสอบชื่อความสัมพันธ์ที่ได้รับ ว่ามีข้อมูลของความสัมพันธ์นั้น อยู่ในพจนานุกรมฐานข้อมูลแบบกระจายท้องถิ่นแล้วหรือยัง เพื่อพจนานุกรมฐานข้อมูลแบบกระจายท้องถิ่นจะไม่ต้องเก็บข้อมูลที่ซ้ำกัน

5.4.5.1.3 exist_table( )

หน่วยโปรแกรมนี้ จะทำหน้าที่ในการตรวจสอบชื่อความสัมพันธ์ที่ได้รับ ว่ามีอยู่ในระบบฐานข้อมูลท้องถิ่นหรือไม่ โดยจะตรวจสอบด้วยข้อมูลในพจนานุกรมฐานข้อมูลของระบบจัดการฐานข้อมูล

5.4.5.1.4 ins_l_t_h_u( )

หน่วยโปรแกรมนี้ จะทำหน้าที่ในการค้นหาข้อมูลในพจนานุกรมฐานข้อมูลของระบบจัดการฐานข้อมูลท้องถิ่น แล้วนำข้อมูลป้อนเข้าสู่ตารางข้อมูล LTHU

5.4.5.1.5 ins_l_t_a_l_a( )

หน่วยโปรแกรมนี้ จะทำหน้าที่ในการค้นหาข้อมูลในพจนานุกรมฐานข้อมูลของระบบจัดการฐานข้อมูลท้องถิ่น แล้วนำข้อมูลป้อนเข้าสู่ตารางข้อมูล LTA และ LA

5.4.5.2 ส่วนป้อนข้อมูลพจนานุกรมฐานข้อมูลโดยรวม

ส่วนป้อนข้อมูลพจนานุกรมฐานข้อมูลโดยรวมนี้ จะทำหน้าที่นำข้อมูลในพจนานุกรมฐานข้อมูลท้องถิ่น และกฎบังคับความถูกต้องท้องถิ่น จากทุกๆสถานที่รวมทั้งสถานที่ที่ทำการประมวลผลด้วย มาป้อนเข้าสู่ตารางพจนานุกรมฐานข้อมูลโดยรวม และตารางเก็บกฎบังคับความถูกต้องตามลำดับ โดยข้อมูลในสถานที่อื่นนั้น จะรับข้อมูลโดยการติดต่อผ่านส่วนติดต่อโครงข่ายสื่อสาร

ตารางพจนานุกรมฐานข้อมูลโดยรวม ยังเก็บรายละเอียดของตารางพจนานุกรมฐานข้อมูลทั้งท้องถิ่นและโดยรวม รวมทั้งรายละเอียดของตารางเก็บกฎบังคับความถูกต้องทั้งท้องถิ่นและโดยรวมของสถานที่นั้นๆด้วย ซึ่งในส่วนนี้จะมีหน่วยโปรแกรมที่ถูกพัฒนาขึ้นหลายหน่วยโปรแกรมดังนี้

5.4.5.2.1 main( )

หน่วยโปรแกรมนี้ จะทำหน้าที่ในการป้อนข้อมูลรายละเอียดของตารางพจนานุกรมฐานข้อมูลทั้งท้องถิ่นและโดยรวม และข้อมูลรายละเอียดของตารางกฎบังคับความถูกต้องทั้งท้องถิ่น

และโดยรวมของสถานที่นั้นๆ จากนั้นจะทำการสร้างการติดต่อไปยังเครื่องคอมพิวเตอร์ต่างๆที่อยู่ในโครงข่ายเดียวกัน เพื่อส่งคำถามเชิงต้นไม้และรับข้อมูลพจนานุกรมฐานข้อมูลท้องถิ่น กฎบังคับความถูกต้องท้องถิ่นของสถานที่นั้นๆ ป้อนเข้าสู่พจนานุกรมฐานข้อมูลโดยรวม และตารางเก็บกฎบังคับความถูกต้อง ตามลำดับ โดยจะมีการเรียกใช้หน่วยโปรแกรมอื่นๆ ดังแสดงไว้ในรูปที่ 5.34

5.4.5.2.2 ins_t_h_u( )

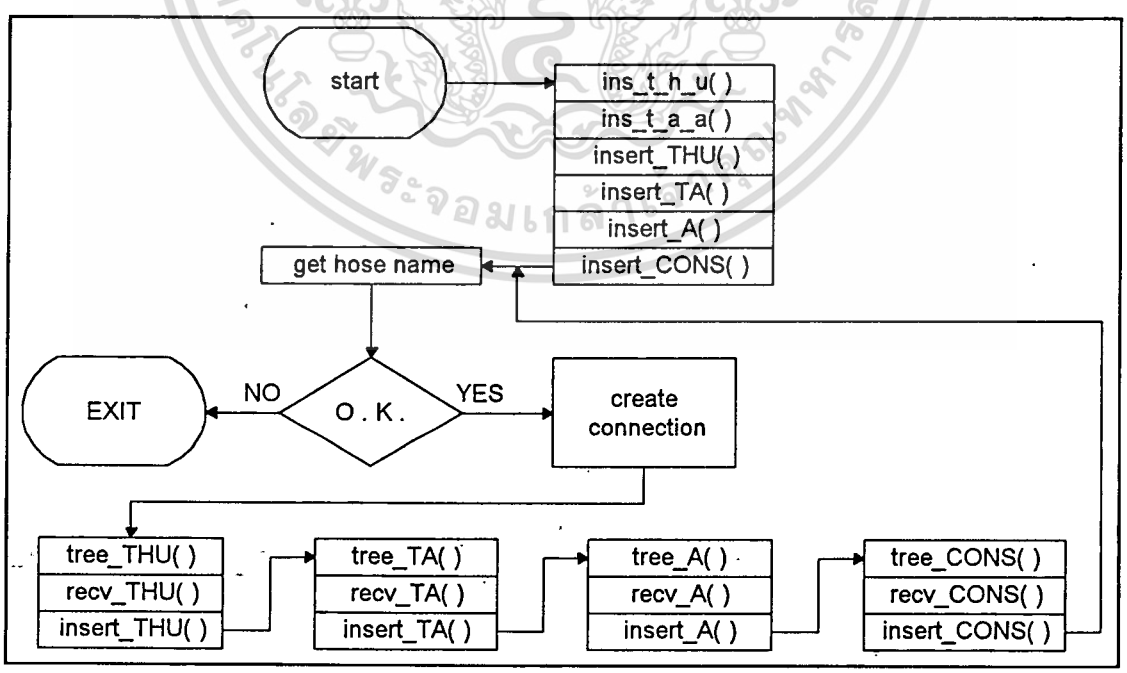
หน่วยโปรแกรมนี้ จะทำหน้าที่ในการนำข้อมูลรายละเอียดของตารางพจนานุกรมฐานข้อมูลท้องถิ่นและโดยรวม และรายละเอียดของตารางเก็บกฎบังคับความถูกต้องท้องถิ่นและโดยรวมของสถานที่นั้นๆ ป้อนเข้าสู่ตารางข้อมูล GTHU

5.4.5.2.3 ins_t_a_a( )

หน่วยโปรแกรมนี้ จะทำหน้าที่ในการนำข้อมูลรายละเอียดของตารางพจนานุกรมฐานข้อมูลท้องถิ่นและโดยรวม และรายละเอียดของตารางเก็บกฎบังคับความถูกต้องท้องถิ่นและโดยรวมของสถานที่นั้นๆ ป้อนเข้าสู่ตารางข้อมูล GTA และตารางข้อมูล GA

5.4.5.2.4 tree_THU( )

หน่วยโปรแกรมนี้ จะทำหน้าที่ในการสร้างคำถามเชิงต้นไม้ ส่งไปยังสถานที่ที่ต้องการข้อมูลในตารางข้อมูล LTHU ของพจนานุกรมฐานข้อมูลท้องถิ่นที่ส่งคำถามไป



รูปที่ 5.34 แสดงขั้นตอนในการเรียกใช้หน่วยโปรแกรมอื่นของหน่วยโปรแกรม main( )

5.4.5.2.5 recv_THU(str)

หน่วยโปรแกรมนี้ จะทำหน้าที่ในการรับข้อมูลพจนานุกรมฐานข้อมูล ที่เก็บไว้ในตัวแปร str ที่เกิดจากการส่งคำถามเชิงต้นไม้ของหน่วยโปรแกรม tree_THU( ) แล้วทำการแยกข้อมูลที่ได้รับออกเป็นข้อมูลแต่ละแอททริบิว เก็บไว้ในตัวแปรสำหรับใช้ต่อไป

5.4.5.2.6 insert_THU( )

หน่วยโปรแกรมนี้จะทำหน้าที่ในการป้อนข้อมูลพจนานุกรมฐานข้อมูล ที่รับมาโดยหน่วยโปรแกรม recv_THU( ) เข้าสู่ตารางข้อมูล GTHU

5.4.5.2.7 tree_TA( )

หน่วยโปรแกรมนี้ จะทำหน้าที่ในการสร้างคำถามเชิงต้นไม้ ส่งไปยังสถานที่ที่ต้องการข้อมูลในตารางข้อมูล LTA ของพจนานุกรมฐานข้อมูลท้องถิ่นที่ส่งคำถามไป

5.4.5.2.8 recv_TA(str)

หน่วยโปรแกรมนี้ จะทำหน้าที่ในการรับข้อมูลพจนานุกรมที่เก็บไว้ในตัวแปร str ที่เกิดจากการส่งคำถามเชิงต้นไม้ของหน่วยโปรแกรม tree_TA( ) แล้วทำการแยกข้อมูลที่ได้รับ ออกเป็นข้อมูลแต่ละแอททริบิว เก็บไว้ในตัวแปรสำหรับใช้ต่อไป

5.4.5.2.9 insert_TA( )

หน่วยโปรแกรมนี้จะทำหน้าที่ในการป้อนข้อมูลพจนานุกรมฐานข้อมูล ที่รับมาโดยหน่วยโปรแกรม recv_TA( ) เข้าสู่ตารางข้อมูล GTA

5.4.5.2.10 tree_A( )

หน่วยโปรแกรมนี้ จะทำหน้าที่ในการสร้างคำถามเชิงต้นไม้ ส่งไปยังสถานที่ที่ต้องการข้อมูลในตารางข้อมูล LA ของพจนานุกรมฐานข้อมูลท้องถิ่นที่ส่งคำถามไป

5.4.5.2.11 recv_A(str)

หน่วยโปรแกรมนี้ จะทำหน้าที่ในการรับข้อมูลพจนานุกรมฐานข้อมูล ที่เก็บไว้ในตัวแปร str ที่เกิดจากการส่งคำถามเชิงต้นไม้ของหน่วยโปรแกรม tree_A( ) แล้วทำการแยกข้อมูลที่ได้รับออกเป็นข้อมูลแต่ละแอททริบิว เก็บไว้ในตัวแปรสำหรับใช้ต่อไป

5.4.5.2.12 insert_A( )

หน่วยโปรแกรมนี้จะทำหน้าที่ในการป้อนข้อมูลพจนานุกรมฐานข้อมูล ที่รับมาโดยหน่วยโปรแกรม recv_A( ) เข้าสู่ตารางข้อมูล GA

5.4.5.2.13 tree_CONS( )

หน่วยโปรแกรมนี้ จะทำหน้าที่ในการสร้างคำถามเชิงต้นไม้ ส่งไปยังสถานที่ที่ต้องการรับข้อมูลกฎบังคับความถูกต้องท้องถิ่น คือตารางข้อมูล LC

5.4.5.2.14 recv_CONS(str)

หน่วยโปรแกรมนี้ จะทำหน้าที่ในการรับข้อมูลกฎบังคับความถูกต้อง ที่เก็บไว้ในตัวแปร str ที่เกิดจากการส่งคำถามเชิงต้นไม้ของหน่วยโปรแกรม tree_CONS( ) แล้วทำการแยกข้อมูลที่ได้รับ ออกเป็นข้อมูลแต่ละแอททริบิว เก็บไว้ในตัวแปรสำหรับใช้ต่อไป

5.4.5.2.15 insert_CONS( )

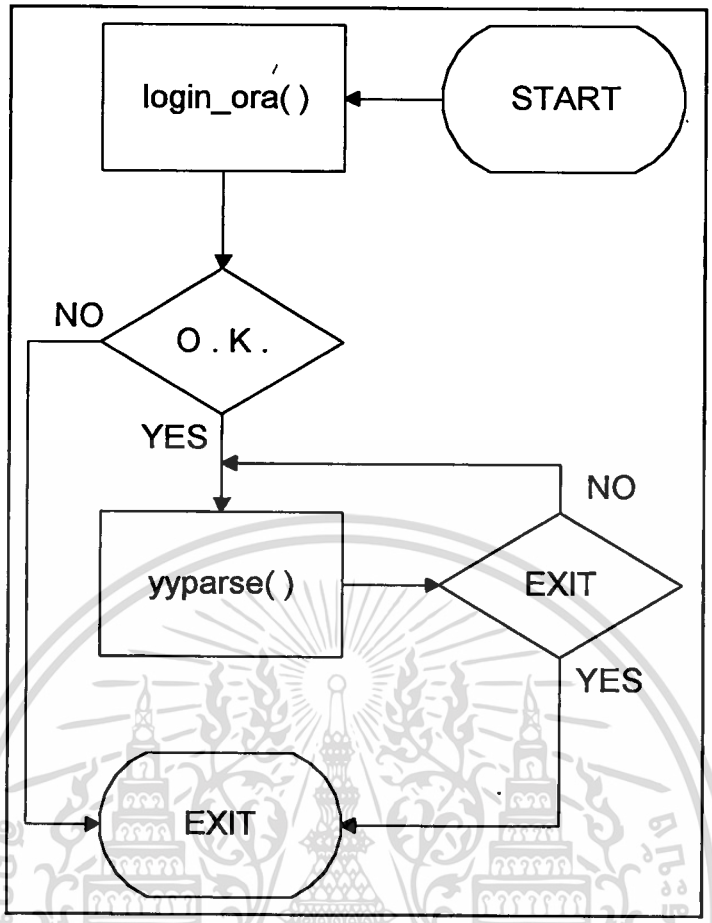
หน่วยโปรแกรมนี้ จะทำหน้าที่ในการป้อนข้อมูลกฎบังคับความถูกต้อง ที่รับมาโดยหน่วยโปรแกรม recv_CONS( ) เข้าสู่ตารางเก็บกฎบังคับความถูกต้อง GC

5.4.6 ส่วนป้อนกฎบังคับความถูกต้อง

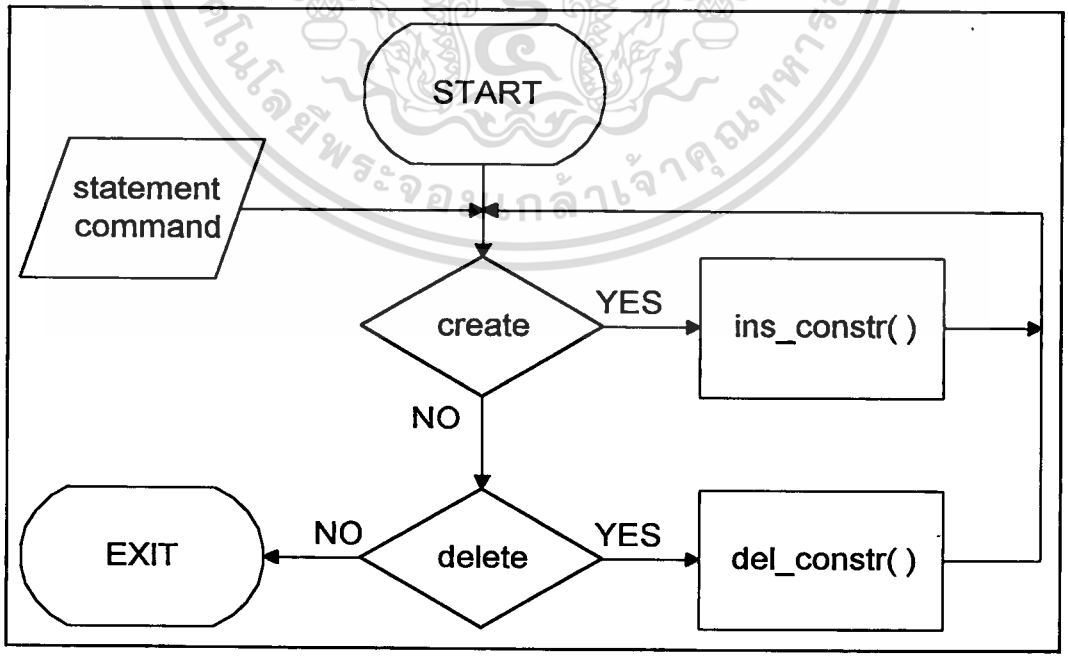
โปรแกรมส่วนนี้ จะทำหน้าที่ในการรับกฎบังคับความถูกต้องท้องถิ่นจากผู้ใช้งาน เก็บไว้ในตารางเก็บกฎบังคับความถูกต้องท้องถิ่น LC เพื่อสร้างเป็นกฎบังคับความถูกต้องโดยรวม โดยส่วนป้อนข้อมูลพจนานุกรมฐานข้อมูลโดยรวมต่อไป และยังทำหน้าที่ในการลบกฎบังคับความถูกต้องที่ไม่ต้องการ ออกจากตารางเก็บกฎบังคับความถูกต้องท้องถิ่นด้วย ซึ่งส่วนนี้จะประกอบไปด้วยหน่วยโปรแกรมต่างๆ กระจายอยู่ในไฟล์ cons_ora.pc , constr.c , constryy.y และ constrlx.l อธิบายได้ดังนี้ คือ

5.4.6.1 main( )

หน่วยโปรแกรมนี้ จะเป็นหน่วยโปรแกรมหลักในการทำงาน ของส่วนป้อนกฎบังคับความถูกต้อง ซึ่งจะมีเรียกใช้หน่วยโปรแกรมอื่น โดยจะวนรอบรับคำสั่งในการเพิ่ม หรือลบกฎบังคับความถูกต้อง จนกว่าจะได้รับคำสั่ง EXIT จึงจะหยุดทำงาน ซึ่งแสดงความสัมพันธ์ในการเรียกใช้หน่วยโปรแกรมอื่น ดังรูปที่ 5.35



รูปที่ 5.35 แสดงขั้นตอนในการเรียกใช้หน่วยโปรแกรมอื่นของหน่วยโปรแกรม main( )



รูปที่ 5.36 แสดงขั้นตอนในการเรียกใช้หน่วยโปรแกรมอื่นของหน่วยโปรแกรม yyparse( )

5.4.6.2 login_ora(user,passwd)

หน่วยโปรแกรมนี้ จะทำหน้าที่ในการติดต่อกับระบบจัดการฐานข้อมูล ORACLE โดยการติดต่อจะอ้างถึงชื่อผู้ใช้งานเป็น user และรหัสผ่านเป็น passwd

5.4.6.3 yyparse( )

หน่วยโปรแกรมนี้ จะถูกพัฒนาโดยโปรแกรมช่วยพัฒนาโปรแกรม LEX และ YACC ซึ่งหน่วยโปรแกรมนี้ จะทำหน้าที่ในการวิเคราะห์คำสั่ง และเรียกใช้หน่วยโปรแกรมอื่นๆ ให้เหมาะสมกับคำสั่งนั้น ดังแสดงความสัมพันธ์ในการเรียกใช้ในรูปที่ 5.36

5.4.6.4 logout_ora( )

หน่วยโปรแกรมนี้ จะถูกเรียกใช้เมื่อต้องการจะยกเลิกการติดต่อกับระบบจัดการฐานข้อมูล ORACLE หลังจากที่เพิ่มหรือลบกฎบังคับความถูกต้องแล้ว

5.4.6.5 ins_constr(constr_name,fst_name,sec_name,thd_name,opt1,val1,opt2,val2)

ในกรณีที่เพิ่มกฎบังคับความถูกต้อง หน่วยโปรแกรมนี้จะทำหน้าที่ในการรับค่าข้อมูลต่างๆมาทำการวิเคราะห์ แล้วเตรียมข้อมูลสำหรับหน่วยโปรแกรม ins_op( ) โดยตัวแปร constr_name จะเก็บชื่อกฎบังคับความถูกต้อง fst_name , sec_name , thd_name จะเก็บชื่อแอททริบิว ที่ถูกอ้างถึงในคำสั่งในตำแหน่งที่ 1, 2 และ 3 ตามลำดับ โดยชื่อแอททริบิวในตัวแปรทั้งสาม จะต้องเป็นชื่อเดียวกัน ตัวแปร opt1 และ opt2 จะเก็บเครื่องหมายการเปรียบเทียบตัวที่ 1 และ 2 ตามลำดับ ส่วนตัวแปร val1 และ val2 จะเก็บค่าที่ใช้เปรียบเทียบตัวที่ 1 และ 2 ตามลำดับ

5.4.6.6 ins_op( )

หน่วยโปรแกรมนี้ จะทำหน้าที่ในการนำข้อมูลที่เตรียมไว้ จากการทำงานของหน่วยโปรแกรม ins_constr( ) ป้อนเข้าสู่ตารางเก็บกฎบังคับความถูกต้อง LC

5.4.6.7 del_constr(name)

หน่วยโปรแกรมนี้ จะทำหน้าที่ในการลบกฎบังคับความถูกต้อง ที่มีชื่อตามที่เก็บไว้ในตัวแปร name ออกจากตารางเก็บกฎบังคับความถูกต้อง LC

5.4.7 ส่วนเครื่องมือ (Tools)

ส่วนนี้จะเป็นหน่วยโปรแกรมที่พัฒนาขึ้นใช้ในหน่วยโปรแกรมต่างๆ โดยจะทำงานทั่วๆไปที่ใช้กันบ่อยๆ หรือใช้กันในหลายส่วน และยังได้รวมถึงหน่วยโปรแกรมที่เรียกใช้ในกรณีที่เกิด

ความผิดพลาดขึ้นด้วย ซึ่งส่วนเครื่องมือนี้จะอยู่ในไฟล์ tool.c และ error.c โดยมีหน่วยโปรแกรมที่สำคัญดังนี้

5.4.7.1 err_quit(va_alist)

หน่วยโปรแกรมนี้ จะถูกเรียกใช้เมื่อตรวจพบว่า โปรแกรมมีข้อผิดพลาดเกิดขึ้น โดยหน่วยโปรแกรมนี้จะทำการแสดงข้อความที่กำหนดไว้ จากนั้นก็จะหยุดการทำงานของโปรแกรม

5.4.7.2 err_sys(va_alist)

หน่วยโปรแกรมนี้ จะถูกเรียกใช้เมื่อตรวจพบว่า โปรแกรมมีข้อผิดพลาดเกิดขึ้นจากการทำงานของระบบปฏิบัติการ โดยหน่วยโปรแกรมนี้จะทำการแสดงข้อความที่กำหนดไว้ พร้อมทั้งหมายเลขของความผิดพลาดที่เกิดขึ้น จากนั้นก็จะหยุดการทำงานของโปรแกรม

5.4.7.3 err_return(va_alist)

หน่วยโปรแกรมนี้ จะถูกเรียกใช้เมื่อตรวจพบว่า หน่วยโปรแกรมที่เรียกใช้มีความผิดพลาดเกิดขึ้น โดยหน่วยโปรแกรมนี้จะทำการแสดงข้อความที่กำหนดไว้ จากนั้นก็จะหยุดการทำงานของหน่วยโปรแกรมที่เรียกใช้ และทำงานในหน่วยโปรแกรมอื่นต่อไป

5.4.7.4 err_dump(va_alist)

หน่วยโปรแกรมนี้ จะถูกเรียกใช้เมื่อตรวจพบว่า โปรแกรมมีข้อผิดพลาดเกิดขึ้น โดยหน่วยโปรแกรมนี้จะทำการแสดงข้อความที่กำหนดไว้ จากนั้นก็จะหยุดการทำงานของโปรแกรม และจะสร้างไฟล์ชื่อ core ด้วย ซึ่งไฟล์นี้สามารถที่จะนำมาหาสาเหตุของความผิดพลาดได้

5.4.7.5 readn(fd,ptr,nbytes)

หน่วยโปรแกรมนี้จะใช้ทำการรับข้อมูลจำนวน nbytes ไบต์ จากซอคเกต fd และจะเก็บข้อมูลไว้ที่หน่วยความจำตำแหน่งที่เริ่มต้นด้วย ptr

5.4.7.6 writen(fd,ptr,nbytes)

หน่วยโปรแกรมนี้จะใช้ทำการส่งข้อมูลจำนวน nbytes ไบต์ ที่อยู่ในหน่วยความจำตำแหน่งที่เริ่มต้นด้วย ptr ไปที่ซอคเกต fd

5.4.7.7 readline(fd,ptr,maxlen)

หน่วยโปรแกรมนี้จะใช้ในการรับข้อมูล 1 บรรทัด จากซอคเกต fd ไปเก็บไว้ที่หน่วยความจำตำแหน่งที่เริ่มต้นด้วย ptr โดยจะรับข้อมูลไปเรื่อยๆจนกว่าจะได้รับข้อมูลแสดงการสิ้นสุดบรรทัด (\n) แต่อย่างไรก็ตามข้อมูลจะต้องไม่มากว่า maxlen ไบต์

5.4.7.8 Free_tree(top_tree)

หน่วยโปรแกรมนี้ จะถูกเรียกใช้เมื่อต้องการคืนหน่วยความจำที่ใช้เก็บคำถามเชิงต้นไม้ทั้งคำถาม ให้กับระบบปฏิบัติการ

5.4.7.9 Free_node(node_ptr)

หน่วยโปรแกรมนี้ จะถูกเรียกใช้เมื่อต้องการคืนหน่วยความจำที่ใช้เก็บโนดของคำถามเชิงต้นไม้ ให้กับระบบปฏิบัติการ

5.4.7.10 Make_node(type,data)

หน่วยโปรแกรมนี้จะถูกเรียกใช้ ในกรณีที่ต้องการสร้างโนดของคำถามเชิงต้นไม้ โดยหน่วยโปรแกรมนี้ โดยจะทำการจองหน่วยความจำให้ด้วย

5.4.7.11 Walk(ptr,init_order)

หน่วยโปรแกรมนี้ จะถูกเรียกใช้ในกรณีที่ต้องการจะกำหนดหมายเลขโนด ให้กับคำถามเชิงต้นไม้ เพื่อจะส่งคำถามเชิงต้นไม้ไปยังสถานที่อื่น เนื่องจากไม่สามารถที่จะส่งตำแหน่งหน่วยความจำข้ามเครื่องคอมพิวเตอร์ได้

5.4.7.12 send_query(sockfd,ptr,n_amount)

หน่วยโปรแกรมนี้ จะถูกเรียกใช้เมื่อต้องการจะส่งคำถามเชิงต้นไม้ ไปยังส่วนประมวลผลในสถานที่อื่น

5.4.7.13 send_tree(sockfd,ptr)

หน่วยโปรแกรมนี้จะถูกเรียกใช้จากหน่วยโปรแกรม send_query( ) ให้ทำการส่งคำถามเชิงต้นไม้ ไปประมวลผลในสถานที่อื่น

5.4.7.14 receive_data(sockfd,line)

หน่วยโปรแกรมนี้จะเรียกใช้ในการรับข้อมูลผลลัพธ์ของการประมวลผลคำถามเชิงต้นไม้

5.4.7.15 pass_view(sockfd)

ในการส่งข้อมูลผลลัพธ์ของส่วนประมวลผลคำถามแบบกระจายนั้น ส่วนประมวลผลคำถามแบบกระจาย จะส่งรายละเอียดการประมวลผลมาด้วยหลังจากที่ได้ส่งข้อมูลมาแล้ว หน่วยโปรแกรมนี้จะทำหน้าที่ในการตอบรับการส่งรายละเอียดการประมวลผลเท่านั้น แต่จะไม่มีการเก็บรายละเอียดการประมวลผลไว้

5.4.7.16 send_auth(uid,pwd,sockfd)

หน่วยโปรแกรมนี้จะใช้ในการส่งชื่อผู้ใช้งานพร้อมรหัสผ่าน ให้กับส่วนประมวลผลคำถามแบบกระจาย เพื่อส่งคำถามไปประมวลผล

5.4.7.17 receive_query(sockfd,number)

หน่วยโปรแกรมนี้ จะถูกเรียกใช้ในการรับคำถามเชิงต้นไม้ ที่ส่งมาจากสถานที่อื่น

5.4.7.18 Make_tree(top,start)

หน่วยโปรแกรมนี้จะถูกเรียกใช้จากหน่วยโปรแกรม receive_query( ) เพื่อทำการสร้างคำถามเชิงต้นไม้ เนื่องจากว่าในการส่งคำถามเชิงต้นไม้ ไม่สามารถที่จะส่งในลักษณะต้นไม้ได้ ดังนั้นจึงต้องส่งเป็นข้อมูลธรรมดา จากนั้นจึงสร้างคำถามเชิงต้นไม้ขึ้นใหม่ ให้มีลักษณะเหมือนเดิม

5.4.7.19 Lex__Attr(str1,str2)

หน่วยโปรแกรมนี้จะทำการแยกเอาชื่อแอททริบิว ออกมาจากชื่อที่มีการรวมกันระหว่างชื่อความสัมพันธ์และชื่อแอททริบิว โดยทั้งสองจะเชื่อกันด้วยเครื่องหมาย " -- "

5.4.7.20 strupr(str)

หน่วยโปรแกรมนี้จะทำการเปลี่ยนข้อความภาษาอังกฤษให้เป็นข้อความตัวพิมพ์ใหญ่

5.4.7.21 LexWord(word,words)

หน่วยโปรแกรมนี้จะทำการแยกคำ ออกจากกลุ่มคำที่แยกกันด้วยช่องว่าง

5.4.7.22 LexSel(str1,str2)

หน่วยโปรแกรมนี้จะใช้ในการแยกคำในส่วนของการกำหนดเงื่อนไข ในคำถามแบบ selection

บทที่ 6

การเปรียบเทียบคุณสมบัติของระบบ DDQ/1 กับระบบจัดการฐานข้อมูลอื่น

ในการพัฒนาระบบประมวลผลคำถาม DDQ/1 นั้น มีจุดประสงค์ที่จะนำระบบฐานข้อมูลแบบรวม ที่สามารถประมวลผลคำถามได้โดยอิสระอยู่แล้ว มาสร้างเป็นระบบฐานข้อมูลแบบกระจาย โดยไม่มีการเปลี่ยนแปลงซอฟท์แวร์ระบบจัดการฐานข้อมูล ดังนั้นในการพัฒนาระบบประมวลผลคำถาม DDQ/1 นี้ ระบบจัดการฐานข้อมูลในแต่ละสถานที่ จะไม่ถูกแก้ไขแต่อย่างใดทั้งสิ้น และปฏิบัติการใน 2 ลักษณะ คือ ปฏิบัติการได้โดยอิสระไม่ขึ้นกับระบบรวม และปฏิบัติการในลักษณะระบบฐานข้อมูลแบบกระจาย

6.1 คุณสมบัติที่สำคัญของระบบจัดการฐานข้อมูลแบบกระจาย

ก่อนที่จะกล่าวถึงการเปรียบเทียบคุณสมบัติของระบบ DDQ/1 กับระบบอื่นๆ จะขอกล่าวถึงคุณสมบัติที่สำคัญของระบบจัดการฐานข้อมูลแบบกระจาย ที่จะได้ในการพิจารณาดังนี้คือ

6.1.1 การประมวลผลคำถามด้วยรูปแบบคาโนนิคัล (Canonical form query processing)

ในการเข้าถึงข้อมูลชุดใดชุดหนึ่งในฐานข้อมูล อาจจะสามารถเข้าถึงได้ด้วยคำถามในหลายรูปแบบ ดังที่ได้กล่าวไว้ในบทที่ 3 เช่น ((S JOIN SPJ) WHERE P# =' P2') [SNAME] = (S JOIN (SPJ WHERE P# = 'P2')) [SNAME] ซึ่งแต่ละรูปแบบจะมีขั้นตอนในการประมวลผลที่ต่างกัน ดังนั้นผู้ใช้งานจะต้องพิจารณาเลือกใช้คำถามที่เข้าถึงข้อมูลดีที่สุด

สำหรับการประมวลผลคำถามในรูปแบบคาโนนิคัลนั้น ไม่ว่าผู้ใช้งานจะเข้าถึงข้อมูลชุดใดชุดหนึ่งด้วยคำถามในรูปแบบใดก็ตาม คำถามจะถูกเปลี่ยนให้อยู่ในรูปแบบเดียวกันก่อนที่จะทำการประมวลผล ซึ่งรูปแบบนี้เรียกว่า Canonical form [27] ซึ่งจะทำให้ผู้ใช้งานไม่ต้องกังวลกับการเลือกใช้รูปแบบของคำถาม

6.1.2 คำถามไม่ขึ้นกับสถานที่ (Site Independence)

ข้อมูลที่กระจายอยู่ในสถานที่ต่างๆในระบบฐานข้อมูลแบบกระจาย จะดูเสมือนว่าเป็นฐานข้อมูลแบบรวมเพียงระบบเดียว ถ้าการใช้คำถามในการเข้าถึงข้อมูลของผู้ใช้งานในระบบฐานข้อมูลแบบกระจายนั้น ผู้ใช้งานสามารถที่จะอ้างถึงตารางข้อมูลในระบบฐานข้อมูลท้องถิ่นต่างๆที่ประกอบกันเป็นระบบฐานข้อมูลแบบกระจาย โดยไม่จำเป็นจะต้องมีการระบุชื่อสถานที่เก็บตาราง

ข้อมูล หรือกล่าวอีกนัยหนึ่งจะได้ว่า ผู้ใช้งานในสถานที่ต่างๆสามารถที่จะเข้าถึงข้อมูลชุดใดชุดหนึ่งได้ด้วยคำถามเดียวกัน ทั้งนี้ไม่ว่าคำถามจะถูกป้อนที่สถานที่ใดก็ตาม

6.1.3 ไม่มีการแก้ไขระบบจัดการฐานข้อมูล (No DBMS Modification)

คุณสมบัติสำคัญอีกประการหนึ่งที่ควรพิจารณาสำหรับระบบจัดการฐานข้อมูลแบบกระจายคือ ระบบจัดการฐานข้อมูลแบบกระจาย ควรจะทำงานได้โดยไม่ทำให้คุณสมบัติของระบบฐานข้อมูลท้องถิ่นที่มีอยู่เดิมต้องเปลี่ยนไป หรือจะกล่าวอีกนัยหนึ่งจะได้ว่าหากระบบฐานข้อมูลท้องถิ่นที่อยู่ในระบบฐานข้อมูลแบบกระจายไม่ถูกแก้ไขใดๆ คุณสมบัติต่างๆที่มีอยู่เดิมย่อมจะไม่เปลี่ยนไป

6.1.4 การอนุญาตโดยรวม (Global Authorization)

เหตุผลหนึ่งที่แสดงให้เห็นว่าข้อมูลที่กระจายอยู่ตามสถานที่ต่างๆ เสมือนกับว่าเป็นฐานข้อมูลโดยรวมเพียงระบบเดียว คือการสร้างการอนุญาตทั้งหมดที่จำเป็น สำหรับผู้เขียนโปรแกรมประยุกต์และผู้ใช้งาน การอนุญาตจะถูกกำหนดในรูปของความสัมพันธ์และตารางเสมือน(views) ซึ่งตารางที่ถูกอ้างถึงเหล่านี้บางครั้งก็จะอยู่ในสถานที่ต่างกัน อีกทั้งการอนุญาตที่จำเป็นอาจจะเปลี่ยนแปลงไปในแต่ละช่วงเวลา ดังนั้นจึงจำเป็นจะต้องมีผู้ดูแลระบบฐานข้อมูลโดยรวมสำหรับจัดการเรื่องเหล่านี้

การกำหนดการอนุญาตทั้งหมดในฐานข้อมูล จะถูกเก็บไว้ในแต่ละฐานข้อมูลโดยรวมที่ถูกใช้โดยผู้ดูแลระบบฐานข้อมูลโดยรวม ซึ่งการกำหนดการอนุญาตที่ถูกรวบรวมเหล่านี้ จะไม่ถูกนำไปใช้สำหรับระบบจัดการฐานข้อมูลใดเป็นการส่วนตัว ดังนั้นจึงมีการรับ-ส่งข้อมูลที่คับคั่งที่ฐานข้อมูลโดยรวม เพื่อที่จะหลีกเลี่ยงปัญหานี้ การกำหนดการอนุญาตจะกระจายไปยังสถานที่ต่างๆ เพื่อเป็นการเพิ่มประสิทธิภาพของระบบให้ดีขึ้น ทั้งนี้จะต้องมีการประสานงานกันระหว่างสิ่งของสองสิ่งคือ การประสานงานกันระหว่างผู้กำหนดการอนุญาต และการประสานงานกันระหว่างระบบจัดการฐานข้อมูลแต่ละสถานที่ ด้วยการประสานงานกันของสิ่งที่กล่าวมา ทำให้ผู้ใช้งานสามารถใช้งานได้ทุกๆสถานที่ โดยการอนุญาตจะไม่ขึ้นกับสถานที่ใช้งาน

6.1.5 การประมวลผลคำถามโดยรวมที่ได้ผลดีที่สุด (Global Query Optimization)

การประมวลผลคำถามโดยรวมที่ได้ผลดีที่สุด จะพิจารณาจากค่าใช้จ่ายที่น้อยที่สุดในการรับ-ส่งข้อมูลระหว่างสถานที่ เพื่อใช้ในการปฏิบัติการของตัวปฏิบัติการพีชคณิตสัมพันธ์ ซึ่งโดยปรกติค่าใช้จ่ายในการส่งข้อมูลจะแปรตามจำนวนข้อมูลที่ส่ง ดังนั้นการประมวลผลคำถามโดยรวม

ที่ได้ผลดีที่สุด จะเป็นการพิจารณาหายุทธวิธีในการประมวลผล โดยการกำหนดสถานที่ในการปฏิบัติการของตัวปฏิบัติการพีชคณิตสัมพันธ์ เพื่อให้ได้ผลลัพธ์ที่มีการรับ-ส่งข้อมูลระหว่างสถานที่น้อยที่สุด[15] ทั้งนี้จะขึ้นอยู่กับชนิดของตัวปฏิบัติการและจำนวนข้อมูลของความสัมพันธ์ที่ร่วมปฏิบัติการ

6.1.6 กฎบังคับความถูกต้องโดยรวม (Global Integrity Constraint)

ในระบบจัดการฐานข้อมูลแบบกระจายที่ดีนั้น กฎบังคับความถูกต้องที่ใช้ในระบบฐานข้อมูลแบบกระจาย ควรจะมีผลกับข้อมูลในฐานข้อมูลท้องถิ่นทุกๆสถานที่ ทั้งนี้ไม่ว่าจะเป็นการอ้างถึง แก้ไข หรือเพิ่มข้อมูลในฐานข้อมูลใดก็ตาม กฎบังคับความถูกต้องจะต้องมีผลบังคับใช้กับทุกฐานข้อมูล ซึ่งเรียกว่ากฎบังคับความถูกต้องโดยรวม กฎบังคับความถูกต้องโดยรวมนั้น จะมีได้จะต้องได้รับความร่วมมือจากผู้ดูแลระบบฐานข้อมูลในแต่ละสถานที่ และการประสานงานกันระหว่างระบบจัดการฐานข้อมูลท้องถิ่นทุกๆสถานที่ด้วยเช่นกัน

6.1.7 พจนานุกรมฐานข้อมูลโดยรวม (Global Data Dictionary)

พจนานุกรมฐานข้อมูลโดยรวม อาจจะสร้างโดยการต่อเติมพจนานุกรมฐานข้อมูลท้องถิ่นหรือแยกสร้างขึ้นใหม่ก็ได้ ขึ้นกับการออกแบบระบบฐานข้อมูลแบบกระจาย โดยปรกติแล้วพจนานุกรมฐานข้อมูลโดยรวม จะถูกใช้ในขั้นตอนการประมวลผลคำถาม เพื่อค้นหาสถานที่ของตารางข้อมูลที่ถูกอ้างถึง ซึ่งถ้าสถานที่ทำการป้อนคำถามมีพจนานุกรมฐานข้อมูลโดยรวมอยู่ จะทำให้การประมวลผลมีความรวดเร็วขึ้น ทั้งนี้เนื่องจากไม่ต้องค้นหาสถานที่เก็บตารางข้อมูลที่ถูกอ้างถึงจากพจนานุกรมฐานข้อมูลในสถานที่อื่น

6.2 ระบบจัดการฐานข้อมูลอินเกรส์

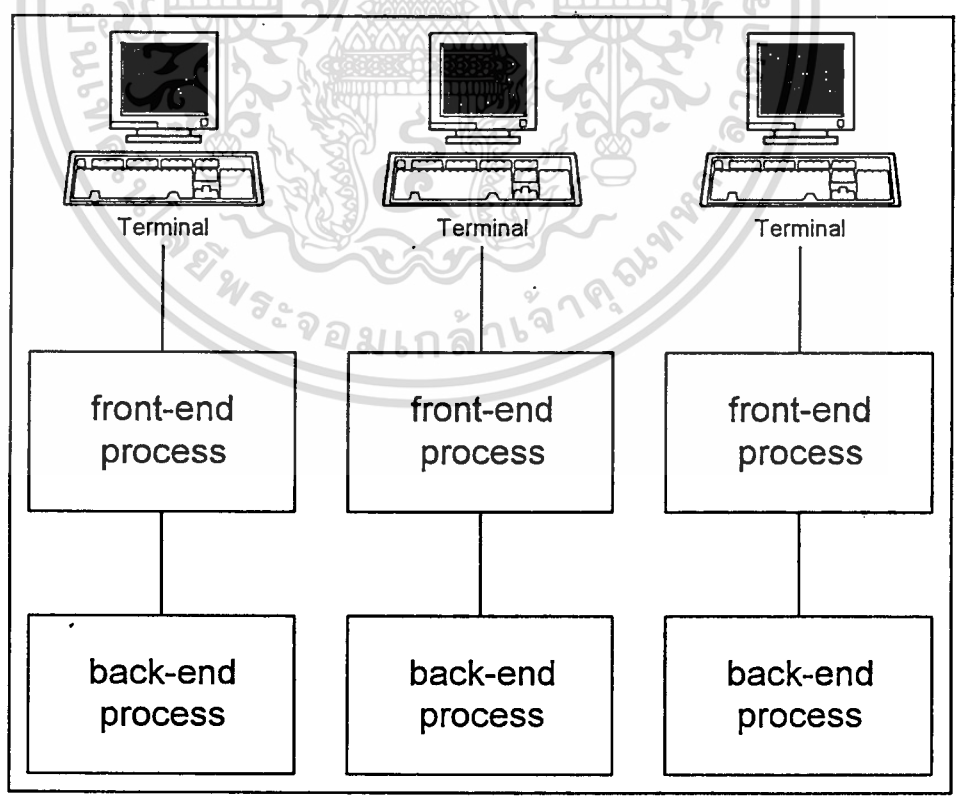
ในการเปรียบเทียบคุณสมบัติครั้งนี้ ได้เลือกระบบจัดการฐานข้อมูลอินเกรส์ (INGRES) ซึ่งเป็นระบบจัดการฐานข้อมูลที่มีใช้กันอย่างแพร่หลาย โดยในหัวข้อนี้จะกล่าวแนะนำความรู้ทั่วไปเกี่ยวกับระบบจัดการฐานข้อมูลอินเกรส์ก่อน ส่วนการเปรียบเทียบจะได้กล่าวไว้ในหัวข้อ 6.3 ต่อไป

ระบบจัดการฐานข้อมูลอินเกรส์ เป็นระบบจัดการฐานข้อมูลแบบสัมพันธ์ เดิมที่เดียวซอฟท์แวร์นี้เป็นผลิตภัณฑ์ของ Berkeley ที่มีใช้สาธารณะทั่วไปในวงการการศึกษาและการวิจัย ต่อมาได้กลายมาเป็นผลิตภัณฑ์ทางการค้าภายใต้ชื่อ INGRES ซึ่งจัดจำหน่ายโดยบริษัท Relational Technology Inc.

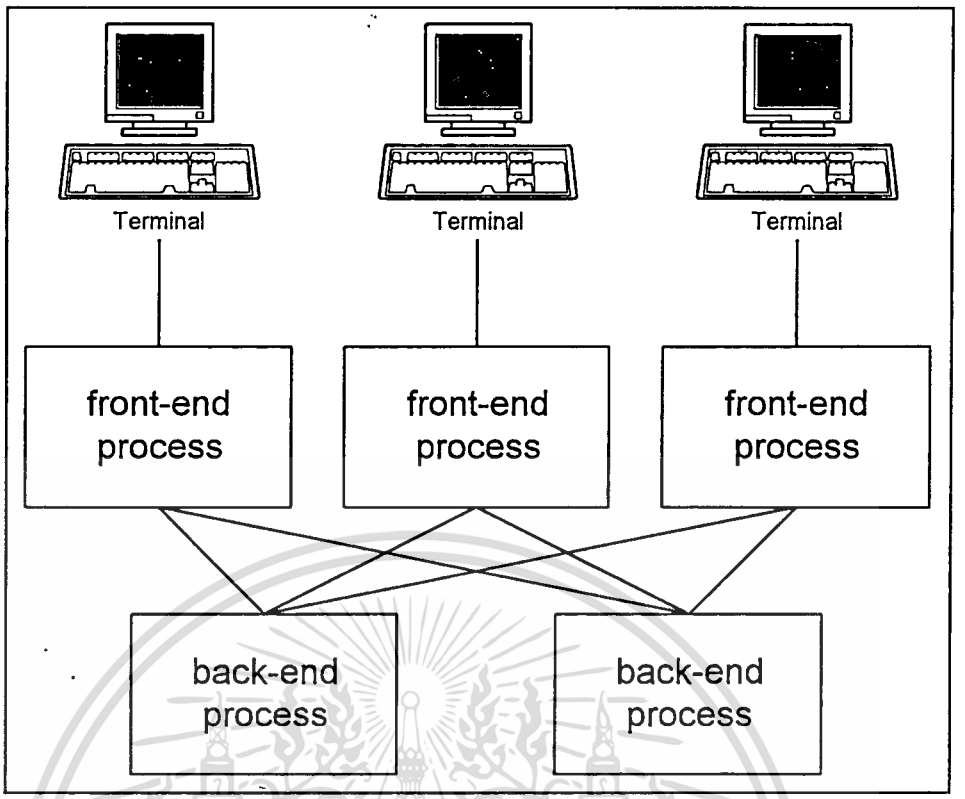
ระบบจัดการฐานข้อมูลอินเกรส์ เป็นระบบจัดการฐานข้อมูลที่ผู้ใช้งานสามารถที่จะติดต่อใช้งาน ด้วยการใช้ภาษาเควล (QUEL) และภาษาสอบถามเชิงโครงสร้าง โดยการใช้ซอฟท์แวร์ INGRES/IQUEL และ INGRES/ISQL ตามลำดับ ซึ่งสถาปัตยกรรมของระบบเป็นดังนี้

6.2.1 สถาปัตยกรรมกระบวนการประมวลผลของระบบจัดการฐานข้อมูลอินเกรส์

สถาปัตยกรรมกระบวนการประมวลผลของระบบจัดการฐานข้อมูลอินเกรส์ประกอบด้วยกระบวนการประมวลผล 2 กระบวนการคือ ส่วนเชื่อมต่อ front-end และส่วนควบคุมการเข้าถึงข้อมูล back-end server ในระบบจัดการฐานข้อมูลอินเกรส์รุ่นที่ 5 สถาปัตยกรรมกระบวนการประมวลผลจะเป็นแบบ Process Model คือ กระบวนการประมวลผล front-end แต่ละกระบวนการจะติดต่อ back-end server หนึ่งกระบวนการดังแสดงในรูปที่ 6.1 สำหรับระบบจัดการฐานข้อมูลอินเกรส์รุ่นที่ 6 สถาปัตยกรรมกระบวนการประมวลผลได้เปลี่ยนไปเป็นแบบ Multiple Server Model คือ กระบวนการประมวลผล front-end หลายกระบวนการจะติดต่อกับกระบวนการประมวลผล back-end จำนวนหนึ่งร่วมกัน ดังแสดงในรูปที่ 6.2 ซึ่งกระบวนการประมวลผล back-end สามารถที่จะกำหนดจำนวนได้ตามต้องการ ทั้งในระบบจัดการฐานนข้อมูลอินเกรส์รุ่นที่ 5 และ 6 กระบวนการประมวลผล front-end และ back-end จะติดต่อกันด้วยวิธีการที่เรียกว่า pipes



รูปที่ 6.1 แสดงสถาปัตยกรรมกระบวนประมวลผลแบบ process model



รูปที่ 6.2 แสดงสถาปัตยกรรมกระบวนประมวลผลแบบ multiple server model

6.2.2 สถาปัตยกรรมการเก็บข้อมูลของระบบจัดการฐานข้อมูลอินเกรส์

ลักษณะการเก็บข้อมูลของระบบจัดการฐานข้อมูลอินเกรส์นั้นมีอยู่ 4 แบบด้วยกันคือ

Heap การเก็บข้อมูลของระบบจัดการฐานข้อมูลอินเกรส์แบบนี้ จะเป็นการเก็บข้อมูลแบบมาตรฐานถ้าไม่มีการกำหนดเป็นแบบอื่น ซึ่งการเก็บข้อมูลแบบนี้จะเป็นแบบลำดับ (sequential) และจะไม่มีการกำหนด key ในการเก็บข้อมูลแบบนี้

Hash การเก็บข้อมูลแบบนี้ จะมีวิธีเลือกตำแหน่งที่จะเก็บ (address) โดยอาศัยค่าของ key เป็นตัวกำหนด วิธีนี้จะลดปริมาณข้อมูลที่เข้า-ออกดิสก์

Isam วิธีการเก็บแบบนี้ จะเรียงข้อมูลโดยใช้ค่าของ key fields ซึ่งดรรชนีจะเป็นแบบสถิตย์ คือ เมื่อมีการเพิ่มข้อมูล จะไม่มีการจัดเรียงลำดับไม่ ดังนี้จะต้องทำการจัดเรียงดรรชนีเองหลังจากมีการแก้ไขข้อมูล

B-Tree วิธีการเก็บข้อมูลแบบนี้ จะค้นหาข้อมูลได้ช้ากว่าวิธีการ Hash แต่ก็เป็นวิธีการที่ใช้งานได้ดีเช่นกัน ดรรชนีจะเป็นแบบจลน์ คือดรรชนีจะถูกปรับปรุงอัตโนมัติในกรณีที่มีการลบหรือเพิ่มข้อมูล วิธีการนี้จะเหมาะสำหรับตารางที่มีการเพิ่มหรือเปลี่ยนขนาดบ่อยๆ และตารางที่มีขนาดใหญ่

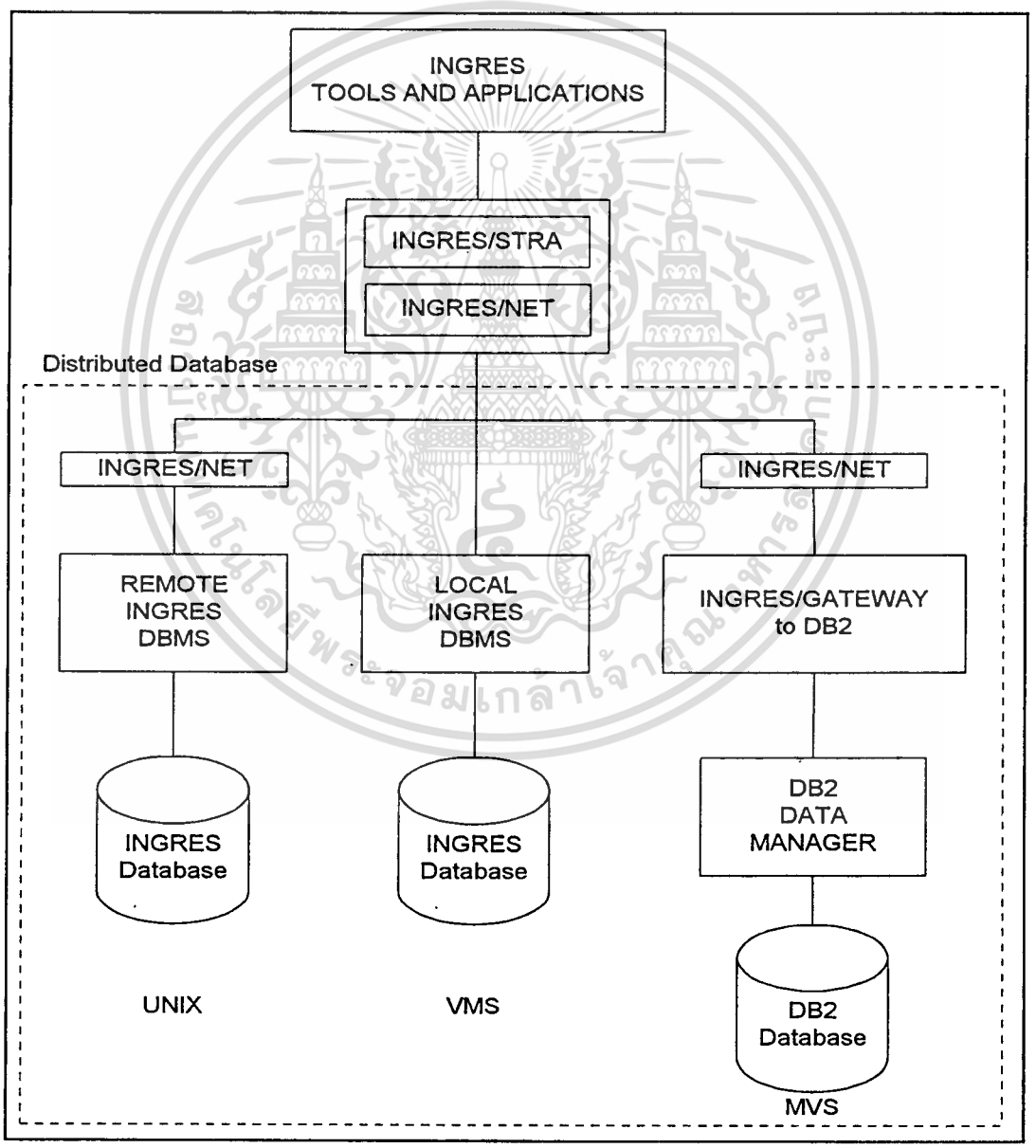
6.2.3 สถาปัตยกรรมของระบบจัดการฐานข้อมูลแบบกระจายอินเกรส์

สำหรับสถาปัตยกรรมระบบจัดการฐานข้อมูลแบบกระจายอินเกรส์ จำเป็นต้องใช้ผลิตภัณฑ์เพิ่มเติมบางอย่าง ดังแสดงในรูปที่ 6.3 อันได้แก่ INGRES/NET, INGRES/GATEWAY, INGRES/STAR ซึ่งผลิตภัณฑ์แต่ละชนิดมีหน้าที่สรุปได้ดังนี้

INGRES/NET มีหน้าที่หลักคือให้บริการในการสื่อสารข้อมูลผ่านโครงข่าย

INGRES/GATEWAY มีหน้าที่หลักในการแปลงโปรโตคอลในการสื่อสาร ในกรณีที่ติดต่อสื่อสารกับระบบจัดการฐานข้อมูลชนิดอื่น

INGRES/STAR มีหน้าที่หลักในการจัดการประมวลผลคำถามแบบกระจาย



รูปที่ 6.3 แสดงตัวอย่างสถาปัตยกรรมของระบบจัดการฐานข้อมูลแบบกระจายอินเกรส์

6.3 การเปรียบเทียบระบบจัดการฐานข้อมูลแบบกระจายอินเกรส์กับระบบ DDQ/1

ระบบจัดการฐานข้อมูลแบบกระจายที่ใช้ในการเปรียบเทียบกับระบบ DDQ/1 ได้เลือกใช้ระบบจัดการฐานข้อมูลแบบกระจายอินเกรส์ ซึ่งใช้ผลิตภัณฑ์ที่ชื่อ INGRES/STAR (ในหัวข้อนี้จะใช้เรียกแทนระบบจัดการฐานข้อมูลแบบกระจายอินเกรส์ด้วย) โดยจะทำการเปรียบเทียบคุณสมบัติตามที่ได้กล่าวไว้ในหัวข้อ 6.1 ทีละคุณสมบัติดังนี้

6.3.1 การประมวลผลคำถามด้วยรูปแบบคาโนนิคัล

DDQ/1

ในขั้นตอนการประมวลผลนั้น หลังจากที่ได้รับคำถามจากผู้ใช้งาน ระบบ DDQ/1 จะมีการเปลี่ยนรูปคำถามให้อยู่ในรูปแบบคาโนนิคัล โดยการใช้กฎที่ได้กล่าวไว้ในหัวข้อ 3.5.2

INGRES/STAR

จากที่ได้ทำการค้นคว้า[28] ในการประมวลผลของ INGRES/STAR มีการเข้าถึงข้อมูลที่ดีที่สุด แต่ไม่มีการยืนยันว่ามีการเปลี่ยนคำถามให้อยู่ในรูปแบบคาโนนิคัล

6.3.2 คำถามไม่ขึ้นกับสถานที่

DDQ/1

ผู้ใช้งานในระบบ DDQ/1 สามารถที่จะใช้คำถามที่อ้างถึงข้อมูลในสถานที่อื่นได้มากกว่าหนึ่งสถานที่ โดยไม่ต้องอ้างถึงสถานที่เก็บข้อมูล เสมือนกับข้อมูลถูกเก็บในสถานที่ท้องถิ่น

INGRES/STAR

ในระบบนี้ผู้ใช้งานจะต้องระบุลักษณะการใช้ในตอนเริ่มต้น ซึ่งมีรูปแบบคำสั่งดังนี้

command [v_node::]dbname[/server_type]

เช่น ingmenu paris::dbname หลังจากที่ได้ระบุลักษณะการใช้แล้ว ผู้ใช้งานสามารถที่จะใช้คำถามที่อ้างถึงข้อมูลในสถานที่ต่างๆได้ เสมือนกับว่าข้อมูลถูกเก็บไว้ในสถานที่เดียวกัน

6.3.3 ไม่มีการแก้ไขระบบจัดการฐานข้อมูล

DDQ/1

ในขั้นตอนการติดตั้งระบบ DDQ/1 จะไม่มีการแก้ไขระบบจัดการฐานข้อมูลเดิมแต่อย่างใด ระบบจัดการฐานข้อมูลเดิมก็ยังคงใช้งานได้ตามปรกติ

INGRES/STAR

ระบบนี้จะไม่มีการแก้ไขระบบจัดการฐานข้อมูลเดิม โดยจะยังคงทำงานได้เช่นเดิม โดยผู้ใช้งานสามารถที่จะกำหนดว่าจะใช้งานด้วยระบบจัดการฐานข้อมูลท้องถิ่น หรือจะใช้งานกับระบบจัดการฐานข้อมูลแบบกระจาย โดยการระบุการใช้ในตอนเริ่มต้นใช้งาน

6.3.4 การอนุญาตโดยรวม

DDQ/1

หลังจากที่ผู้ใช้งานได้รับอนุญาตให้ใช้ระบบแล้ว ไม่ว่าผู้ใช้งานจะเรียกใช้จากสถานที่ใดก็ตาม ผู้ใช้งานจะมีสิทธิใช้ข้อมูลในตารางข้อมูลต่างๆได้ เท่าที่ได้รับอนุญาตไว้ ทั้งนี้จะไม่ขึ้นอยู่กับสถานที่ใช้งาน

INGRES/STAR

ในการอนุญาตให้ผู้ใช้งานใช้ข้อมูลต่างๆนั้น จะมีการเก็บการอนุญาตนี้ไว้ในพจนานุกรมฐานข้อมูล ดังนั้นผู้ใช้งานจะใช้งานที่สถานที่ใดก็ตามจะได้รับการอนุญาตเท่าที่ถูกกำหนดโดยผู้ดูแลระบบจัดการฐานข้อมูล

6.3.5 การประมวลผลคำถามโดยรวมที่ได้ผลดีที่สุด

DDQ/1

ในกรณีที่คำถามมีการประมวลผลข้อมูลระหว่างสถานที่ ระบบนี้จะพิจารณาว่าจะทำการประมวลผลที่สถานที่ใดจึงมีการรับ-ส่งข้อมูลน้อยที่สุด โดยพิจารณาจากจำนวนทัปเปิลของความสัมพันธ์ที่ถูกประมวลผล จากนั้นจึงกำหนดสถานที่ประมวลผล

INGRES/STAR

สำหรับระบบนี้จะมีการพิจารณาจากสถิติการใช้ข้อมูล จำนวนทัปเปิลของความสัมพันธ์ที่ถูกประมวลผล และความสามารถของตัวประมวลผล (CPU) หลังจากที่ได้วิเคราะห์แล้วจึงกำหนดสถานที่ในการประมวลผล

6.3.6 กฎบังคับความถูกต้องโดยรวม

DDQ/1

ระบบ DDQ/1 มีการใช้กฎบังคับความถูกต้องช่วยในการประมวลผลคำถาม ถ้าคำถามที่ทำการประมวลผลขัดแย้งกับกฎบังคับความถูกต้อง ระบบก็จะแจ้งให้ผู้ใช้งานทราบทันทีว่าไม่มีข้อมูลที่อยู่ในเงื่อนไขของคำถาม

INGRES/STAR

ในระบบนี้จะไม่มีการใช้กฎบังคับความถูกต้องช่วยในการประมวลผล

6.3.7 พจนานุกรมฐานข้อมูลโดยรวม

DDQ/1

ในระบบ DDQ/1 นี้ ในแต่ละสถานที่ที่ประกอบกันเป็นระบบฐานข้อมูลแบบกระจาย จะมีพจนานุกรมฐานข้อมูลโดยรวมทุกสถานที่ ดังนั้นการประมวลผลคำถามในแต่ละสถานที่จึงทราบรายละเอียดเกี่ยวกับสถานที่อื่นได้ทันที ทำให้การประมวลผลใช้เวลาน้อยลง

INGRES/STAR

สำหรับระบบนี้จะมีพจนานุกรมฐานข้อมูลโดยรวมเช่นกัน แต่จะมีเฉพาะสถานที่ศูนย์กลางเท่านั้น ดังนั้นการประมวลผลคำถามในสถานที่อื่นที่ไม่ใช่สถานที่ศูนย์กลาง จะต้องทำการค้นพจนานุกรมฐานข้อมูลโดยรวมที่สถานที่ศูนย์กลาง จึงทำให้การประมวลผลใช้เวลามากขึ้น

6.4 ผลการทดลองใช้ระบบ DDQ/1

สำหรับการทดลองใช้ระบบ DDQ/1 นั้น ได้ทำการทดลองด้วยเครื่องมินิคอมพิวเตอร์ Tragon M30 ที่ใช้ตัวประมวลผลของบริษัทโมโตโรล่าหมายเลข 68020 และเครื่องมินิคอมพิวเตอร์ Tragon M5 ที่ใช้ตัวประมวลผลของบริษัทโมโตโรล่าหมายเลข 68030 โดยเครื่องคอมพิวเตอร์ทั้งสองใช้ชื่อว่า kmitl30 และ kmitl5 ตามลำดับ โดยได้ทำการป้อนคำถามที่เครื่องคอมพิวเตอร์ kmitl30 ซึ่งข้อมูลตัวอย่างที่อยู่บนเครื่องคอมพิวเตอร์ kmitl30 จะประกอบด้วยตารางข้อมูล S , M , P , SPJ และข้อมูลตัวอย่างที่อยู่บนเครื่องคอมพิวเตอร์ kmitl5 จะประกอบด้วยตารางข้อมูล S5 , M5 , P5 , SPJ5 ดังแสดงในตารางข้อมูลที่ 6.1-6.5 ข้อมูลที่อยู่บนเครื่องคอมพิวเตอร์ทั้งสองนั้น จะเป็นข้อมูลที่เหมือนกันจะต่างกันเพียงชื่อตารางเท่านั้น ตัวอย่างเช่น ข้อมูลในตารางข้อมูล S จะเหมือนกับข้อมูลในตาราง S5 เป็นต้น ทั้งนี้เพื่อจะได้เปรียบเทียบเวลาที่ใช้ในการประมวลผลของตัวปฏิบัติการพีชคณิตสัมพันธ์หนึ่งๆ ระหว่างการใช้ข้อมูลที่อยู่ในระบบฐานข้อมูลเดียวกัน กับการใช้ข้อมูลที่อยู่ต่างระบบฐานข้อมูลกัน

จากการทดลองป้อนคำถามต่างๆ แล้วทำการจับเวลาที่ใช้ในการประมวลผล ด้วยการเรียกใช้ฟังก์ชัน time() โดยเริ่มจับเวลาตั้งแต่ป้อนคำถามจนกระทั่งเริ่มแสดงผลลัพธ์ ดังแสดงไว้ในตารางที่ 6.6 ซึ่งเป็นเวลาที่ใช้ในการประมวลผลเฉลี่ยของแต่ละคำถาม จะเห็นว่าในคำถามที่เข้าถึงข้อมูลที่เหมือนกัน แต่อยู่ในเครื่องคอมพิวเตอร์ต่างเครื่องกัน คำถามที่เข้าถึงข้อมูลที่เครื่อง

คอมพิวเตอร์นั้นๆจะใช้เวลาสั้นกว่า ยิ่งข้อมูลมีขนาดใหญ่มากขึ้น ความแตกต่างของเวลาก็จะมีมากขึ้นตามไปด้วย

S#	SNAME	STATUS	CITY
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

ตารางที่ 6.1 แสดงข้อมูลในตาราง S และ S5

S#	SNAME	STATUS	CITY
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S6	Shiko	20	Tokyo
S7	Chick	30	Bangkok

ตารางที่ 6.2 แสดงข้อมูลในตาราง M และ M5

P#	PNAME	COLOR	WEIGHT	CITY
P1	Nut	Red	12	London
P2	Bolt	Green	17	Paris
P3	Screw	Blue	17	Rome
P4	Screw	Red	14	London
P5	Cam	Blue	12	paris
P6	Cog	Red	19	London

ตารางที่ 6.3 แสดงข้อมูลในตาราง P และ P5

J#	JNAME	CITY
J1	Sorter	Paris
J2	Punch	Rome
J3	Reader	Athens
J4	Console	Athens
J5	Collator	London
J6	Terminal	Oslo
J7	Tape	London

ตารางที่ 6.4 แสดงข้อมูลในตาราง J และ J5

S#	P#	J#	QTY
S1	P1	J1	200
S1	P1	J4	700
S2	P3	J1	400
S2	P3	J2	200
S2	P3	J3	200
S2	P3	J4	500
S2	P3	J5	600
S2	P3	J6	400
S2	P3	J7	800
S2	P5	J2	100
S3	P3	J1	200
S3	P4	J2	500
S4	P6	J3	300
S4	P6	J7	300
S5	P2	J2	200
S5	P2	J4	100
S5	P5	J5	500
S5	P5	J7	100
S5	P6	J2	200
S5	P1	J4	100
S5	P3	J4	200
S5	P4	J4	800
S5	P5	J4	400
S5	P6	J4	500

ตารางที่ 6.5 แสดงข้อมูลในตาราง SPJ และ SPJ5

คำถามที่ใช้ในการทดลอง (Relational Algebra)	เวลาที่ใช้ในการประมวลผล (วินาที)
S ;	5.33
S5 ;	6.33
M ;	5.00
M5 ;	4.66
P ;	4.66
P5 ;	5.33
J ;	4.00
J5 ;	4.66
SPJ ;	4.00
SPJ5 ;	5.00
S UNION M ;	12.00
S UNION M5 ;	21.00
S MINUS M ;	11.33
S MINUS M5 ;	21.66
S INTERSECT M ;	13.00
S INTERSECT M5 ;	19.66
S TIMES P ;	12.33
S TIMES P5 ;	20.33
P WHERE PNAME = 'Screw' ;	11.00
P5 WHERE PNAME = 'Screw' ;	7.66
S[SNAME] ;	7.00
S5[SNAME] ;	7.00
S JOIN SPJ ;	11.66
S JOIN SPJ5 ;	24.00
SPJ[S#,P#] DIVIDEBY P[P#] ;	21.33
SPJ[S#,P#] DIVIDEBY P5[P#] ;	24.66

ตารางที่ 6.6 แสดงผลการทดลองในการป้อนคำถาม

<u>หมายเหตุ</u>

1. ขณะทำการทดลองเครื่องคอมพิวเตอร์ kmitl30 มีจำนวนกระบวนการประมวลผลเท่ากับ 76 กระบวนการประมวลผล

2. ขณะทำการทดลองเครื่องคอมพิวเตอร์ kmitl5 มีจำนวนกระบวนการประมวลผลเท่ากับ 40 กระบวนการประมวลผล

บทที่ 7
สรุปและแนวทางการวิจัย

7.1 สรุป

จากการนำข้อมูลและข่าวสารมาใช้งานในด้านต่างๆมากมาย ทำให้ระบบจัดการฐานข้อมูลถูกนำมาใช้งานมากขึ้นตามไปด้วย ทั้งนี้เพื่อความสะดวกรวดเร็วในการจัดการและค้นหาข้อมูล รวมทั้งการรักษาความปลอดภัยของข้อมูลด้วย จากสาเหตุที่มีการใช้กันอย่างแพร่หลายนี้ ระบบจัดการฐานข้อมูลจึงได้รับการพัฒนาอย่างต่อเนื่องตลอดมา จากระบบฐานข้อมูลแบบรวมที่สามารถทำงานได้โดยอิสระ จนถึงระบบจัดการฐานข้อมูลแบบกระจาย ที่เชื่อมต่อระบบฐานข้อมูลแบบรวมให้สามารถทำงานประสานกันได้ เสมือนกับเป็นระบบฐานข้อมูลเดียวกัน

ระบบประมวลผลคำถาม DDQ/1 ได้พัฒนาขึ้นเพื่อเชื่อมต่อระบบจัดการฐานข้อมูลแบบรวมหลายระบบ ที่ทำงานได้โดยอิสระในโครงข่ายคอมพิวเตอร์เดียวกัน ให้สามารถทำงานร่วมกันเป็นระบบฐานข้อมูลแบบกระจายได้ โดยไม่มีการแก้ไขเปลี่ยนแปลงระบบฐานข้อมูลเดิม และระบบประมวลผลคำถาม DDQ/1 จะรับคำถามจากผู้ใช้งานด้วยภาษาพีชคณิตสัมพันธ์

ภาษาพีชคณิตสัมพันธ์เป็นภาษาที่สามารถแสดงความสัมพันธ์ของข้อมูลที่สร้างขึ้นใหม่ ในรูปของความสัมพันธ์ของข้อมูลและตัวปฏิบัติการพีชคณิตสัมพันธ์ ซึ่งตัวปฏิบัติการพีชคณิตสัมพันธ์มีทั้งหมด 8 ชนิด คือ

select คือการสร้างความสัมพันธ์ จากการเลือกเอาข้อมูลทัปเปิลที่มีคุณสมบัติตรงตามที่กำหนด จากความสัมพันธ์ที่มีอยู่แล้ว

project คือการสร้างความสัมพันธ์ จากการเลือกเอาข้อมูลในแอททริบิวที่กำหนดจากความสัมพันธ์ที่อ้างถึง

product คือการสร้างความสัมพันธ์ จากความสัมพันธ์สองความสัมพันธ์ ซึ่งความสัมพันธ์ที่สร้างขึ้นใหม่นี้จะประกอบไปด้วยข้อมูลทั้งหมด ที่เกิดจากการนำข้อมูลในทัปเปิลของทั้งสองความสัมพันธ์มาจับคู่กัน

union คือการสร้างความสัมพันธ์ขึ้นใหม่ โดยการรวมข้อมูลของสองความสัมพันธ์ที่มีโดเมนเดียวกัน

intersection เป็นการสร้างความสัมพันธ์จากความสัมพันธ์สองความสัมพันธ์ โดยข้อมูลของความสัมพันธ์ที่สร้างขึ้น จะประกอบไปด้วยข้อมูลที่อยู่ในทั้งสองความสัมพันธ์

difference เป็นการสร้างความสัมพันธ์จากความสัมพันธ์สองความสัมพันธ์ โดยข้อมูลของความสัมพันธ์ที่สร้างขึ้น จะประกอบด้วยข้อมูลที่อยู่ในความสัมพันธ์แรกและไม่อยู่ในความสัมพันธ์ที่สอง

join เป็นการสร้างความสัมพันธ์จากสองความสัมพันธ์ โดยข้อมูลของความสัมพันธ์ที่สร้างขึ้นใหม่จะประกอบด้วยข้อมูลที่เกิดจากการจับคู่ของทัปเปิลของความสัมพันธ์ทั้งสอง ภายใต้เงื่อนไขอย่างใดอย่างหนึ่ง

divide เป็นการสร้างความสัมพันธ์จากความสัมพันธ์สองความสัมพันธ์ คือความสัมพันธ์ชนิด binary และความสัมพันธ์ชนิด unary ซึ่งข้อมูลของความสัมพันธ์ที่สร้างขึ้นใหม่จะประกอบด้วย ข้อมูลของความสัมพันธ์ชนิด binary ในแอททริบิวที่ไม่ร่วมกับแอททริบิวของความสัมพันธ์ชนิด unary โดยที่ข้อมูลในแอททริบิวที่ร่วมกันของชนิด binary จะเหมือนกับข้อมูลทั้งหมดในความสัมพันธ์ชนิด unary

สำหรับสถาปัตยกรรมของระบบประมวลผลคำถามแบบกระจาย DDQ/1 ประกอบด้วยส่วนต่างคือ

ส่วนจัดการกระบวนการเริ่มต้น จะทำหน้าที่ในการสร้างพจนานุกรมฐานข้อมูลท้องถิ่นและพจนานุกรมฐานข้อมูลโดยรวม รวมทั้งยังทำหน้าที่ในการรับกฎบังคับความถูกต้อง เข้าเก็บในตารางเก็บกฎบังคับความถูกต้อง

ส่วนติดต่อผู้ใช้งาน ส่วนนี้จะทำหน้าที่ในการรับคำถามจากผู้ใช้งานด้วยภาษาพีชคณิตสัมพันธ์ โดยจะทำการตรวจไวยากรณ์ว่าถูกต้องหรือไม่ จากนั้นก็จะทำการแปลงคำถามให้อยู่ในรูปคำถามเชิงต้นไม้ส่งให้กับส่วนประมวลผลคำถามแบบกระจายต่อไป

ส่วนประมวลผลคำถามแบบกระจาย ส่วนนี้จะรับคำถามในรูปของคำถามเชิงต้นไม้ และจะทำการปรับปรุงรูปคำถามให้อยู่ในรูปที่ใช้เวลาประมวลผลน้อยที่สุด จากนั้นจะตรวจสอบคำถามว่าสอดคล้องกับกฎบังคับความถูกต้องหรือไม่ แล้วจึงทำการประมวลผลต่อไป โดยมีการใช้การกำหนดสถานที่ในการประมวลผลมาช่วยในการประมวลผลอีกด้วย

ส่วนเชื่อมต่อโครงข่าย ในส่วนนี้จะทำหน้าที่ในการติดต่อรับ-ส่งข้อมูล ระหว่างระบบจัดการฐานข้อมูลต่างสถานที่กัน ทั้งในขั้นตอนการประมวลผลและขั้นตอนการติดตั้งระบบ โดยจะทำงานในลักษณะกระบวนการประมวลผลบริการ

จากการที่ได้ทำการทดลองใช้ระบบประมวลผลคำถาม DDQ/1 ที่ได้พัฒนาขึ้นนี้ สามารถทำงานได้ตามวัตถุประสงค์ที่ได้ตั้งไว้ คือ สามารถเชื่อมต่อระบบจัดการฐานข้อมูลแบบรวมหลายระบบ ที่ทำงานได้โดยอิสระในโครงข่ายคอมพิวเตอร์เดียวกัน ให้สามารถทำงานร่วมกันเป็นระบบฐานข้อมูลแบบกระจายได้ ซึ่งผลการทดลองใช้เป็นที่น่าพอใจ

7.2 แนวทางการวิจัย

สำหรับขอบเขตของการวิจัยในครั้งนี้ ได้ตั้งวัตถุประสงค์ของการทำงานของระบบ DDQ/1 ไว้เพียงระบบประมวลผลคำถามสำหรับฐานข้อมูลแบบกระจายเท่านั้น ดังนั้นจึงไม่มีความสามารถ

ที่จะทำการแก้ไขข้อมูลในฐานข้อมูลได้ ดังนั้นแนวทางในการทำการวิจัยต่อจากการวิจัยในครั้งนี้ ควรจะเพิ่มความสามารถให้กับระบบ DDQ/1 คือสามารถทำการแก้ไขหรือเพิ่มเติมข้อมูลลงในฐานข้อมูลได้ ทั้งนี้จะต้องอยู่บนพื้นฐานของความถูกต้องของข้อมูล (integrity) ด้วยในกรณีที่การทำงานล้มเลว

อีกประการหนึ่ง การประมวลผลของระบบ DDQ/1 ในแต่ละคำถามนั้น ส่วนประมวลผลแบบกระจาย จะทำการประมวลผลตัวปฏิบัติการพีชคณิตสัมพันธ์ทีละตัวปฏิบัติการ ซึ่งเมื่อมีคำถามบางส่วนที่ต้องส่งไปทำการประมวลผลในฐานข้อมูลอื่น ส่วนประมวลผลคำถามแบบกระจายจะรอจนได้รับผลลัพธ์ของส่วนของคำถามที่ส่งไป จากนั้นจึงจะทำการประมวลผลคำถามส่วนที่เหลือต่อไป ทำให้ใช้เวลาส่วนหนึ่งในการรอผลลัพธ์ ดังนั้นในการวิจัยครั้งต่อไป การทำงานในส่วนนี้ควรจะได้รับการพัฒนาให้สามารถประมวลผลคำถามส่วนที่เหลือไว้ล่วงหน้าโดยที่ไม่ต้องรอผลลัพธ์ เมื่อได้รับผลลัพธ์ก็สามารถที่จะประมวลผลต่อได้ทันที

เอกสารอ้างอิง

[1] C. J. Date, "A Guide to Ingers," Addison-Wesley publishing company, 1987 :

[2] Ulka Rodgers, "Unix database management system," Prentice-Hall, 1990 :

[3] ORACLE, Company. "SQL*NET TCP/IP USER 'S GUIDE," Oracle corporation, 1987.

[4] Sitansu S. Mittra, " Principles of Relational Database System," Prectice-Hall, 1991 :69-94,116-129.

[5] Stefano Ceri and Giuseppe ,Pelagatti, "Distribute database principles & systemps," McGraw-Hill, 1984 :1-172.

[6] U. S. Chakravarthy, D. H. Fishman and J. Minker, "Semantic Query Optimization in Expert Systems and Database Systems," Expert Database Systems, The Benjamin/Cummings Publishing Company, Inc. ,1986 :659-674.

[7] Larry Kerschberg ,Peter D. Ting and S. Bing YAo, "Query Optimization in Star Computer," ACM Transactions on Database Systems, Vlo. 7, No. 4, December 1982 :678-711.

[8] S. Bing Yao, "Optimization of Query Evaluation Algorithms," ACM Transactions on Database Systems, Vlo. 4, No. 2, June 1979 :133-155.

[9] Stefano Ceri and Giuseppe Pelagatti, "Allocation of Operations in Distributed Database Access," IEEE Transactions on Computers, Vol. c-31, No. 2, February 1982 :119-129.

[10] C. J. Date, " An Intorduction to Database Systems Volume 1," Fourth Edition, Addison-Wesley, World Student Series, 1986 :12-15

[11] E. F. codd, " Relational Completeness of Data Base Sublanguage," Data Base System, Courant Computer Science Symposia Series, Vol. 6, Prentice-Hall, Englewood Cliffs, NJ, 1972.

[12] Ralph B. Bisland, " Datab Management Developing Application Systems Using Oracle," Prentice-Hall International Editions, Prentice-Hall , Inc., A Division of Simon & Schuster, Englewood cliffs, NJ ,1989 :53-86

[13] Carolyn J. Hursch and Jack L. Hursch,"INGRES SQL DEVELOPER'S GUIDE," Windcrest/McGraw-Hill , 1992 :111-120.

[14] J. D. Ullman, " Principles of Database Systems, " 2nd ed., Computer science Press, 1983.

[15] M. TAMER ÖZSU and PATRICK VALDURIEZ, "PRINCIPLES OF DISTRIBUTED DATABASE SYSTEMS," Prentice-Hall International Editions, Prentice-Hall , Inc., A Division of Simon & Schuster, Englewood cliffs, NJ ,1991 :66-257,425-445.

[16] Esen Ozkarahan, "Database Machines and Database Management," Prentice-Hall International Editions, Prentice-Hall , Inc., A Division of Simon & Schuster, Englewood cliffs, NJ ,1986 :408-453.

[17] HENRY F. KORTH and ABRAHAM SILBERSCHATZ, "DEATABASE SYSTEM CONCEPTS," McGraw-Hill International Editions, McGraw-Hill , Inc., 1986 :301-325, 403-449.

[18] W. Richard Stevens, "Unix Network Programming," Prentice-Hall International Editions, Prentice-Hall , Inc., A Division of Simon & Schuster, Englewood cliffs, NJ ,1991 :1-420.

[19] AT&T, "Unix System V Programmer' s reference manual AT&T," Prentice-Hall , Inc., A Division of Simon & Schuster, Englewood cliffs, NJ ,1987.

[20] M. Negri and G. Pelagatti, "Distributeve Join: A new algorithm for joining relations," ACM Transactions on Database Systems, Vol. 16, No. 4, December 1991, :655-669.

[21] G. M. Nijssen and T. A. Halpin, "Conceptual Schema and Relational Database Design," Prentice-Hall of Australia Pty Ltd, 1989 :1-303.

[22] G. M. Nijssen and E. D. Falkenberg, "Introduction to IBM SQL," Nijssen Data Bases Pty. Ltd., 1984 :1-193.

[23] "Networking Software Package",Nixdorf Computer AG ,W-Germany ,1990 :1-1 - 5-6 .

[24] ORACLE, Company. "PRO*C USER 'S GUIDE," Oracle corporation, 1987 :1-258.

[25] ORACLE, Company. "SQL*Plus REFERENCE GUIDE," Oracle corporation, 1987 :1-1 - 2-111.

[26] ORACLE, Company. "SQL*NET USER 'S GUIDE," Oracle corporation, 1986.

[27] E . F . CODD, "The Relational Model for Database Management Version 2," Addision-Wesley Publishing Company, 1990 :351-430.

[28] Relational Technology Inc.,"Destributed INGRES for the UNIX & VMS Operating Systems," Relational Technology Inc., Alameda, California U.S.A., 1989:1-1 - 8-20.

```
/*
 * Read "n" bytes from a descriptor.
 * Use in place of read() when fd is a stream socket.
 */

int readn(fd, ptr, nbytes)
register int    fd ;
register char   *ptr ;
register int    nbytes ;
{
        int     nleft, nread ;

        nleft = nbytes ;
        while (nleft > 0)
        {
                nread = read (fd, ptr, nleft) ;
                if ( nread < 0)
                        return (nread) ;
                else if (nread == 0)
                        break ;
                nleft -= nread ;
                ptr += nread ;
        }
        return (nbytes - nleft) ;
}
/*
 * Write "n" bytes to a descriptor.
 * Use in place of write() when fd is a stream socket.
 */

int writen( fd, ptr, nbytes)
register int    fd ;
register char   *ptr ;
register int    nbytes ;
{
        int     nleft, nwritten ;

        nleft = nbytes ;
        while(nleft > 0)
        {
                nwritten = write(fd, ptr, nleft) ;
```

```
                if (nwritten <= 0)
                return (nwritten) ;
                nleft -= nwritten ;
                ptr += nwritten ;
        }
        return (nbytes - nleft) ;
}

/*
 * Read a line from a descriptor. Read the line one byte at a time,
 * looking for the newline. We store the newline in the buffer,
 * then follow it with a null ( the same sa fgets(3)).
 * We return the number of characters up to, but not including,
 * the null (the same as strlen(3)).
 */

int readline(fd, ptr, maxlen)
register int    fd ;
register char   *ptr ;
register int    maxlen ;
{
        int     n, rc ;
        char    c ;

        for (n = 1; n < maxlen; n++)
        {
                if ( (rc = read(fd, &c, 1)) == 1)
                {
                        *ptr++ = c ;
                        if (c == '\n')
                        break ;
                }
                else if (rc == 0)
                {
                        if (n == 1)     /* EOF, no data read */
                                return (0) ;
                        else
                                break;  /* EOF, some data was read */
                }
                else
                        return (-1) ;   /* error */
        }
        *ptr = 0 ;
        return (n) ;
}
```

```
/*
 * Read a stream socket one line at a time, and write each line back
 * to the sender.
 *
 * Return when the connection is terminated.
 */

#define MAXLINE     512

str_echo(sockfd)
int     sockfd ;
{
        int     n ;
        char    line[MAXLINE] ;

        for(;;)
        {
                n = readline (sockfd, line, MAXLINE) ;
                if ( n == 0)
                        return ;        /* connection terminated */
                else if (n < 0)
                        err_dump ("str_echo: readline error") ;
                if (writen (sockfd, line, n) != n )
                        err_dump ("str_echo: writen error") ;
        }
}

/*
 * Read the contents of the FILE *fp, write each line to the
 * stream socket (to the server process), then read a line back from
 * the socket and write it to the standard output.
 *
 * Return to caller when an EOF is encountered on the input file.
 */

#include        <stdio.h>
#define MAXLINE     512

str_cli(fp, sockfd)
register FILE   *fp ;
register int    sockfd ;
{
        int     n ;
        char    sendline[MAXLINE], recvline[MAXLINE + 1] ;
```

```
        while ( fgets (sendline, MAXLINE, fp) != NULL)
        {
                n = strlen (sendline) ;
                if (writen (sockfd, sendline, n) != n)
                        err_sys ("str_cli: writen error on socket") ;
                n = readline (sockfd, recvline, MAXLINE) ;
                if (n < 0 )
                        err_dump ("str_cli: readline error") ;
                recvline[n] = 0 ;                       /* null terminate */
                fputs (recvline, stdout) ;
        }
        if (ferror (fp))
                srr_sys ("str_cli: error reading file") ;
}

/* Example of client using TCP protocol. */
#include        "inet.h"
#include        <memory.h>
main(argc,argv)
int     argc ;
char    *argv[ ] ;
{
        int     sockfd ;
        struct sockaddr_in      serv_addr ;

        pname = argv[0] ;
        memset ((char *) &serv_addr,0, sizeof (serv_addr)) ;
        serv_addr.sin_family = AF_INET ;
        serv_addr.sin_addr.s_addr = inet_addr (SERV_HOST_ADDR) ;
        serv_addr.sin_port = htons(SERV_TCP_PORT) ;
        if ((sockfd = socket (AF_INET, SOCK_STREAM, 0)) < 0)
                err_sys ("client: can't open stream socket") ;
        if (connect (sockfd, (struct sockaddr *) &serv_addr, sizeof(serv_addr)) < 0)
                err_sys ("client: can't connect to server") ;
        str_cli (stdin, sockfd) ;
        close (sockfd) ;
        exit (0) ;
}

/* Example of server using TCP protocol. */

#include        "inet.h"
main(argc,argv)
int     argc ;
```

```
char    *argv[ ] ;
{
        int       sockfd, newsockfd, clilen, childpid ;
        struct sockaddr_in        cli_addr, serv_addr ;
        pname = argv[0] ;
        if ((sockfd = socket (AF_INET, SOCK_STREAM, 0)) < 0)
                err_dump ("server: can't open stream socket") ;
        bzero ((char *) &serv_addr, sizeof (serv_addr)) ;
        serv_addr.sin_family = AF_INET ;
        serv_addr.sin_addr.s_addr = htonl (INADDR_ANY) ;
        serv_addr.sin_port = htons (SERV_TCP_PORT) ;
        if (bind (sockfd, (struct sockaddr *) &serv_addr, sizeof (serv_addr)) < 0)
                err_dump ("server: can't bind local address") ;
        listen (sockfd,5) ;
        for ( ;; )
        {
                clilen = sizeof (cli_addr) ;
                newsockfd = accept (sockfd, (struct sockaddr *) &cli_addr, &clelen) ;
                if (newsockfd < 0)
                        err_dump ("server: accept error") ;
                if ((childpid = fork ()) < 0)
                        err_dump ("server: fork error") ;
                else if (childpid == 0)
                {
                        close (sockfd) ;
                        str_echo (newsockfd) ;
                        exit (0) ;
                }
                close (newsockfd) ;
        }
}
```

ภาคผนวก ข

การเขียนโปรแกรม LEX

LEX เป็นเครื่องซอฟท์แวร์ที่ใช้งานด้านต่างๆ เช่น การประมวลผลอักษร การเขียนตัวแปลภาษา เป็นต้น ซึ่งเครื่องมือซอฟท์แวร์ LEX นี้ จะทำหน้าที่ในการตรวจคำ และทำการแยกเป็นคำ พร้อมทั้งสามารถที่จะให้เกิดการทำงานอย่างใดอย่างหนึ่งได้ เมื่อตรวจสอบพบคำที่กำหนดไว้ ขั้นตอนการพัฒนาโปรแกรมด้วยเครื่องมือซอฟท์แวร์ LEX ได้แสดงไว้ในรูปที่ ข.1

ในการเขียนโปรแกรม LEX โดยปรกติแล้ว จะประกอบด้วย 3 ส่วนด้วยกัน คือ

1.ส่วนนิยาม (Definition)

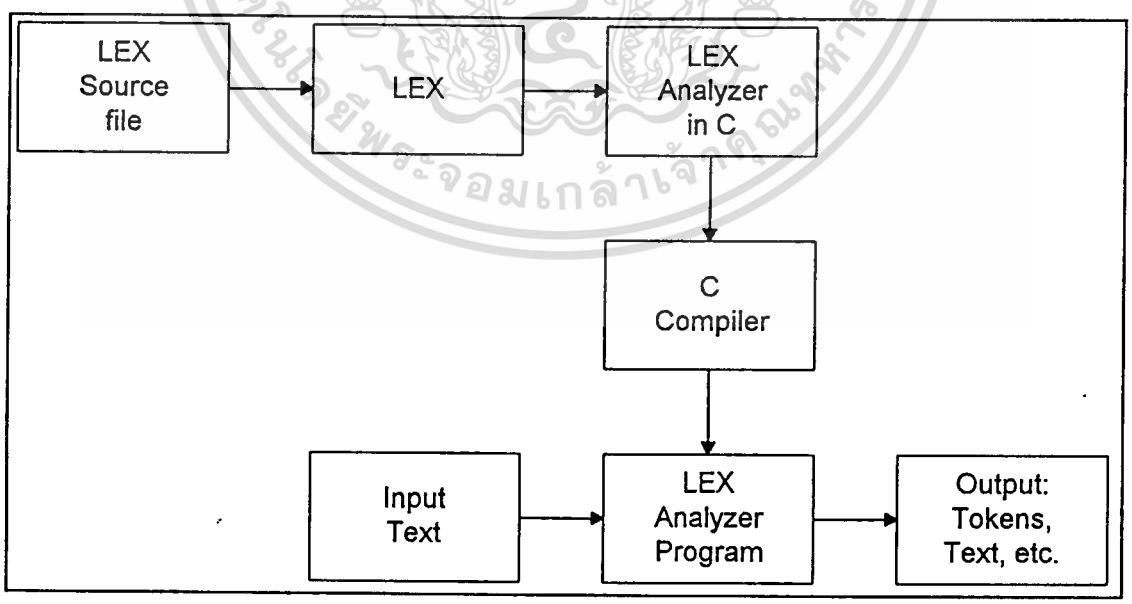
2. ส่วนกฎ (Rules)

3.ส่วนยูสเซอร์ซับรูทีน (User Subroutines)

โดยส่วนของกฎจะเป็นส่วนบังคับที่จำเป็นต้องมี สำหรับส่วนนิยามและส่วนยูสเซอร์ซับรูทีนไม่จำเป็นต้องมีก็ได้ ซึ่งในแต่ละส่วนอธิบายได้ดังนี้

1. ส่วนนิยาม

ส่วนนี้จะถูกประกาศไว้ระหว่างเครื่องหมาย %{ และ %} โดยมีลักษณะเหมือนกับส่วนนิยามในภาษาซี อันประกอบด้วย extern เพื่อบอกว่า ข้อมูลหรือ ฟังก์ชั่นนั้นเป็น เอ็กเทอร์นอลลิงก์เกจ (External Linkage) , ส่วนนิพจน์ #include และส่วนประกาศตัวแปร



รูปที่ ข.1 แสดงขั้นตอนการพัฒนาโปรแกรมด้วยเครื่องมือซอฟท์แวร์ LEX

ตัวอย่าง

```
%{
        #include        "y.tab.h"
        extern  int     tokval ;
        int     lineno ;
%}
```

2.ส่วนกฎ

ส่วนของกฎที่ใช้ในการตรวจสอบคำทั้งหมด จะอยู่ภายในส่วนที่ขึ้นต้นด้วยเครื่องหมาย %% และจบด้วยเครื่องหมาย %% โดยกฎแต่ละกฎจะประกอบด้วย 2 ส่วนคือ ส่วนที่เป็นการบรรยายถึงรูปแบบของโทคเคน(คำที่ต้องการจะทำการตรวจสอบ) เรียกว่า แพทเทอร์น (Pattern) และ ส่วนที่จะให้เกิดการทำงานเมื่อพบแพทเทอร์นดังกล่าว ซึ่งส่วนนี้จะเป็นนิพจน์ของภาษาซี 1 นิพจน์

ตัวอย่าง

```
%%
        -[0-9]          printf ("negative value") ;
        +?[0-9]+        printf ("positive integer") ;
        -0.[0-9]+       printf ("negative fraction,no whole number part") ;
        rail[ ]+road    printf ("railroad is one word") ;
        crook           printf ("Here's acrook") ;
        function        subprogcount++ ;
        G[a-zA-Z]*      {
                                printf ("may have a G word here:",yytext) ;
                                Gstringgcount++ ;
                        }
%%
```

3. ส่วนยูสเซอร์สับรูทีน (User Subroutine)

ส่วนยูสเซอร์สับรูทีนนี้จะเป็นส่วนของฟังก์ชันต่างๆที่ผู้เขียนโปรแกรมเขียนขึ้น เพื่อให้ทำงานอย่างใดอย่างหนึ่งเมื่อตรวจสอบพบคำที่กำหนดไว้ โดยจะเรียกใช้ได้จากส่วนกฎ ในการเขียน

ฟังก์ชันต่างๆเหล่านี้จะเขียนไว้ต่อจากส่วนกฎ ซึ่งในส่วนนี้ผู้เขียนโปรแกรมสามารถที่จะเขียนฟังก์ชันเหมือนกับฟังก์ชันในภาษาซีได้ตามปกติ

ตัวอย่าง

```
%%
.
.
"/*"                    skipcmnts ( ) ;
.
.               /* rest of rules */
%%
skipcmnts ( )
{
        for ( ; ; )
        {
                while (input () != '*') ;
                if (input () != '/')
                        unput (yytext[yylen-1]) ;
                else
                        return ;
        }
}
```

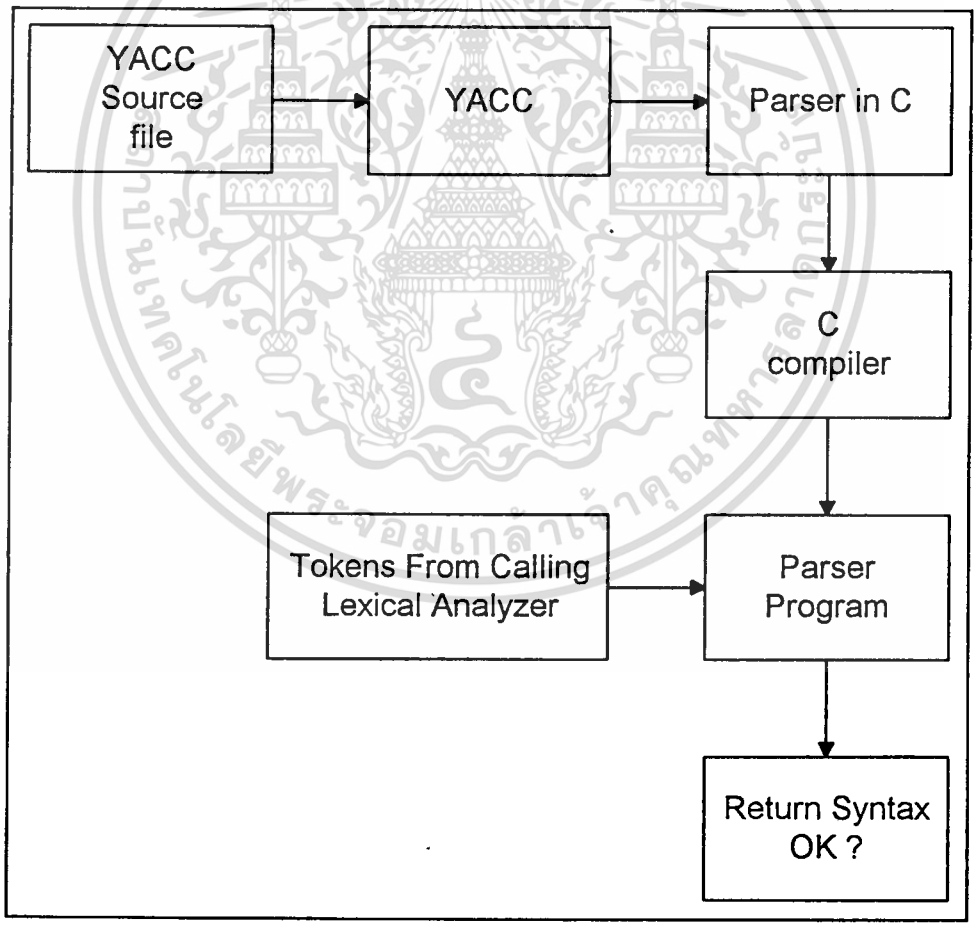
ภาคผนวก ค
การใช้เครื่องมือซอฟท์แวร์ YACC

YACC เป็นเครื่องมือซอฟท์แวร์ที่ใช้ในการวิเคราะห์ไวยากรณ์ของภาษา ซึ่งโดยปรกติแล้วจะใช้งานร่วมกับเครื่องมือซอฟท์แวร์ LEX ซึ่งแสดงขั้นตอนการใช้งานได้ ดังรูปที่ ค.1

ส่วนประกอบของโปรแกรม YACC จะประกอบไปด้วยส่วนต่างๆ 4 ส่วนด้วยกันคือ

1. ส่วนนิยาม (Declarations)

ส่วนนิยามในโปรแกรม YACC นี้จะเหมือนกับส่วนนิยามในโปรแกรม LEX คือ จะใช้ในการประกาศตัวแปรที่ใช้เก็บค่าต่างๆ ทั้งตัวแปรภายใน และตัวแปรภายนอก พร้อมทั้งส่วนนิพจน์ #include ด้วย ซึ่งส่วนนี้จะอยู่ประกาศไว้ระหว่างเครื่องหมาย %{ และ %}



รูปที่ ค.1 แสดงการใช้งานเครื่องมือซอฟท์แวร์ YACC

2. ส่วนประกาศสัญกรณ์ (Declare Token)

ส่วนนี้จะอยู่ระหว่างส่วนนิยามและส่วนไวยากรณ์ ซึ่งจะใช้ในการประกาศว่าในไวยากรณ์มีการใช้สัญกรณ์ที่เป็นแบบเทอร์มินัล(สัญกรณ์ที่ไม่อยู่ทางด้านขวาของกฎใดๆ)อะไรบ้าง และจะเริ่มตรวจสอบด้วยไวยากรณ์ใด โดยในการประกาศแต่ละบรรทัดจะต้องนำหน้าด้วยเครื่องหมาย %

ตัวอย่าง

```
%token CREATE , DELETE
%token WHERE
%start  begin
```

3. ส่วนไวยากรณ์ (Grammar rules)

ในส่วนนี้จะประกอบด้วยส่วนไวยากรณ์ที่อยู่ในรูปสัญกรณ์บีเอ็นเอฟ (BNF) และส่วนที่จะให้เกิดการทำงาน เมื่อสัญกรณ์ที่เข้ามาตรงกับไวยากรณ์นั้นๆ ซึ่งไวยากรณ์ทั้งหมดจะอยู่ระหว่างเครื่องหมาย %% และ %%

ตัวอย่าง

```
begin   :create
          {
              ins_constr(constr_name,fst_attr,sec_attr,thd_attr,opt1,val1,opt2,val2) ;
              return( 1 ) ;
          }
        | delete
          {
              del_constr (constr_name) ;
              return (1) ;
          }
        | EXIT
          {
              return (0) ;
          }
        ;
```

4. ส่วนยูสเซอร์สับรูทีน (User Subroutine)

ส่วนนี้จะเหมือนกันส่วนยูสเซอร์สับรูทีนในโปรแกรมที่ใช้กับเครื่องมือซอฟท์แวร์ LEX โดยจะอยู่ต่อจากส่วนไวยากรณ์

จากที่ได้กล่าวมาทั้ง 4 ส่วนข้างต้น จะสรุปโครงสร้างของโปรแกรมที่ใช้กับเครื่องมือซอฟท์แวร์ ได้ดังนี้

```
%{
        declarations
%}
%token ...
%%
        grammar rules
%%
        subroutines
```

ภาคผนวก ง

โปรแกรม DDQ/1

1. ส่วนติดต่อผู้ใช้งาน

file agb.mak

```
OBJECT= agb.o error.o tool.o agblex.o
TOOL=$(HOME)/prog/tool
agb         : $(OBJECT)
                cc $(CFLAGS) -o agb $(OBJECT)
agb.o       : agb.c $(TOOL)/def.h agbglb.h $(TOOL)/proto.h agb_prot.h
                cc -c $(CFLAGS) agb.c
error.o     :$(TOOL)/error.c $(TOOL)/proto.h
                cc -c $(CFLAGS) $(TOOL)/error.c
tool.o      :$(TOOL)/tool.c $(TOOL)/def.h $(TOOL)/proto.h
                cc -c $(CFLAGS) $(TOOL)/tool.c
agblex.o    : agblex.c $(TOOL)/def.h agbxtern.h $(TOOL)/proto.h agb_prot.h
                cc -c $(CFLAGS) agblex.c
```

file agb_proto.h

```
/* in file agblex.c */
int         Main_Algebra() ;
void        Get_Query() ;
int         SeparateWord() ;
TREE_PTR    Syntax() ;
TREE_PTR    Expression() ;
TREE_PTR    Selection() ;
TREE_PTR    Infix_expr() ;
TREE_PTR    Projection() ;
TREE_PTR    Primitive() ;
int         Select_pred() ;
int         SelPredUnit() ;
int         Attr_spec() ;
char        *Infix_op() ;
int         Iden_name() ;
int         Compare() ;
int         Constant() ;
void        con_back_end() ;
```

file agbglb.h

```
char    Query[AGB_LEN] ;
char    Token_No[MAX_TOKEN][TOKEN_LEN] ;
char    Alpha_Num[LINE_LEN] ;   /* alphanumeric characters */
int     Exit_Flag ;             / if = 1  program terminate */
int     Word_Pos ;
int     Cursor ;
char    uid[15], pwd[15];       /* uid & pwd used to connect to oracle */
```

file agbxtern.h

```
extern  char  Query[AGB_LEN] ;
extern  char  Token_No[MAX_TOKEN][TOKEN_LEN] ;
```

```
extern  char  Alpha_Num[LINE_LEN] ;   /* alphanumeric characters */
extern  int   Exit_Flag ;
extern  int   Word_Pos ;
extern  int   Cursor ;
extern  char  uid[15], pwd[15] ;          /* used to connect to oracle */
```

file agb.c

```
/*
 * Call when user get start.
 */
#include            <string.h>
#include            <sys/types.h>
#include            "../tool/def.h"
#include            "../tool/proto.h"
#include            "agb_prot.h"
#include            "agbglb.h"

void    main(argc, argv)
int     argc;
char    *argv[];
{
        int     sockfd, servlen ;
        struct sockaddr_un  serv_addr;

        /*
        * Fill in the structure "serv_addr" with the address of the
        * server that we want to send to.
        */
        memset((char *)&serv_addr, 0, sizeof(serv_addr));
        serv_addr.sun_family = AF_UNIX;
        strcpy(serv_addr.sun_path, UNIXSTR_PATH0);
        servlen = strlen(serv_addr.sun_path) + sizeof(serv_addr.sun_family);

        if((sockfd = socket(AF_UNIX, SOCK_STREAM, 0)) < 0)
                err_sys("agb: can't open stream socket");
        if(connect(sockfd, (struct sockaddr *)&serv_addr, servlen) < 0)
                err_sys("agb: can't connect to back_end server");

        /* do all */
        if (argc == 1)
        {
                printf("Data base user name: ");
                gets(uid);
                strcpy(pwd, getpass("your password:"));
        }
        else if (argc == 2)
        {
                strcpy(uid,argv[1]) ;
                strcpy(pwd, getpass("your password:"));
        }
        else if (argc == 3)
        {
                strcpy(uid, argv[1]) ;
                strcpy(pwd, argv[2]) ;
        }
        else
        {
                err_quit("Usage: agb [uid [pwd]]\n") ;
        }
        Main_Algebra(sockfd);
        close(sockfd) ;
        exit(0) ;
}
```

file agblex.c

```
#include        "../tool/def.h"
#include        "../tool/proto.h"
#include        "agb_prot.h"
#include        "agbxtern.h"

int order;

int Main_Algebra(sockfd)
int     sockfd ;
{
        int     i, n ;
        char    sendline[10] ;
        TREE_PTR    top ;

        Exit_Flag = FALSE ;
        if (send_auth(uid,pwd,sockfd) == -1)
                err_quit("Main_Algebra: uid and pwd are incorrect") ;
        while( 1 )
        {
                printf ("\nAGB>") ;
                Get_Query (Query) ;
                if (SeparateWord () != TRUE)
                        continue ;
                if (Exit_Flag == TRUE)
                {
                        strcpy(sendline,EXIT) ;
                        strcat(sendline,"\n") ;
                        n = strlen(sendline) ;
                        if (writen(sockfd,sendline,n) != n)
                                err_dump("algebra: writen error") ;
                        break ;
                }
                top = (TREE_PTR)Syntax();
                if (top == NULL)
                        continue ;
                else
                {
                        printf("syntax OK! \n");
                        order = 0 ;
                        order = Walk(top,order) ;
                        con_back_end(sockfd,top,order) ;
                        continue ;
                }
        }
}

/* ..... GET QUERY ..... */

void Get_Query(str)
char *str ;
{
        char    ch, *temp ;

        temp = str ;
        *str = '\0' ;           / clear string variable */
        /* repeat get character until character is ';' follow by Enter */
        do
        {
                ch = (char)getchar();
                switch( ch )
                {
                        case '\b': if ( str > temp)  str-- ;
                        case '\n': *str++ = ' ' ;
                                break ;
                        default  : *str++ = ch ;
```

```
                    break ;
                }
                /* if back space remove character in string too. */
                /* carriage return can use but not be included in the string */
        }
        while (ch != ';') ;
        *str = '\0';            /* add the end of string with null */
        return ;
}

/* ..... SEPARATE WORD ..... */

int SeparateWord(void)
{
        char    *str;
        int     result ;                /* length of match characters */
        int     i, count, str_len, token_no ;
        int     quote_flag ;  /* flag = 0 is out of quote */

        str = Query ;
        quote_flag = 0 ;
        token_no = 0 ;                  /* number of token's element */
        count = 0 ;
        strcpy(Alpha_Num,LETTERS);      /* alphanumeric is letters + digit */
        strncat(Alpha_Num,DIGITS,10);   /* not include dot '.' */
        for (i=0;i<MAX_TOKEN;i++)
                strcpy(Token_No[i],"") ;        /* clear all word first */
        str_len = strlen(str) ;
        while (count < str_len)
        {
                if ((result = strspn(str," ")) != 0)
                {
                        str += result ;
                        count += result ;
                }
                /* don't care for blank */
                /* strspn() return the length of initial segment that consist of */
                else if (strspn(str,SIGN) == 1)
                {
                        result = strspn(str+1,DIGITS) ;
                        if (result != 0)
                        {
                                strncpy(Token_No[token_no],str,result+1) ;
                                strcpy(Token_No[token_no]+result+1,"") ;
                                token_no++ ;
                                str = str+result+1 ;
                                count = count+result+1 ;
                        }
                }
                else if (((result = strspn(str,DIGITS)) != 0) ||
                        ((result = strspn(str,Alpha_Num)) != 0) ||
                        ((result = strspn(str,OPERATOR)) != 0))
                {
                        /* if digits or alphanumeric or operator */
                        /* then save into word */
                        strncpy(Token_No[token_no],str,result);
                        strcpy(Token_No[token_no]+result,"");
                        /* characters out of quotes must be change to upper case */
                        if (quote_flag == 0)
                                strupr(Token_No[token_no]) ;
                        token_no++ ;
                        str += result ;
                        count += result ;
                }
                else if ((result = strspn(str,DELIMITER)) != 0)
                {
                        /* if delimiter, save into word */
                        /* character by character */
```

```
                for (i=0;i<result;i++)
                {
                        strncpy(Token_No[token_no],str++,1) ;
                        strcpy(Token_No[token_no]+1,"") ;
                        if (strcmp(Token_No[token_no],"") == 0)
                        {
                                quote_flag = !quote_flag ;
                                /* dot and blank can use in alpha_num */
                                /* when quotes detect */
                                if (quote_flag == 1)
                                        strcat(Alpha_Num," .") ;
                                else
                                {
                                        strcpy(Alpha_Num,LETTERS) ;
                                        strncat(Alpha_Num,DIGITS,10) ;
                                }
                        }
                        token_no++ ;
                        count++ ;
                }
        }
        /* if character out of list in definition */
        /* character is invalid and query is invalid too */
        else
        {
                printf("token_no is %d\n",token_no);
                printf("token is %s\n",Token_No[token_no]);
                printf("Invalid character occur in this query\n");
                return (FALSE);
        }
    }
    if ((strcmp(Token_No[0],"EXIT") == 0) && (strcmp(Token_No[1],";") == 0))
        Exit_Flag = TRUE;
    return(TRUE);
}

/* ..................... SYNTAX ANALYSIS ..................... */

TREE_PTR Syntax(void)
{
    TREE_PTR top_ptr ;
    Word_Pos = 0 ;
    top_ptr = NULL ;
    top_ptr = Expression() ;

    if ((top_ptr != NULL ) && (strspn(Token_No[++Word_Pos],";") != 0))
    {
        return(top_ptr);
    }
    else
    {
        printf("Syntax error !\n") ;
        Free_tree(top_ptr) ;
        return(NULL);
    }
        /* if query is not in any cases then query error    */
        /* if query is in above cases but next word not ';' */
        /* query is still incomplete. */
}

/* ..................... EXPRESSION ..................... */

TREE_PTR Expression(void)
{
    int     index;
    TREE_PTR    temp ;

    temp = NULL ;
```

```
        index = Word_Pos;       /* mark start checking position */
        temp = Selection() ;
        if (temp == NULL)
        {
                /* if not selection move to start position */
                Word_Pos = index;
                temp = Infix_expr() ;
                if (temp == NULL)
                {
                        /* if not infix_expr move to start position */
                        Word_Pos = index;
                        temp = Projection() ;
                        if (temp == NULL)
                                return(NULL);
                        /* if also not projection expression FALSE */
                }
        }
        return(temp) ;
}

TREE_PTR Selection(void)
{
        TREE_PTR    current, left ;
        char    cond[AGB_LEN] ;

        strcpy(cond,"") ;
        left = Primitive() ;
        if (left == NULL)           /* primitive   */
                return(NULL) ;
        else
        {
                Word_Pos++;
                if (strcmp(Token_No[Word_Pos],"WHERE") != 0)       /* where */
                {
                        Free_tree(left) ;
                        return(NULL) ;
                }
                else
                {
                        Word_Pos++ ;
                        if (!Select_pred(cond))             /* select_pred */
                        {
                                Free_tree(left) ;
                                return(NULL) ;
                        }
                        current = Make_node(SELECT, cond) ;
                        current->left = left ;
                        return(current) ;
                }
        }
}

TREE_PTR Infix_expr(void)
{
        TREE_PTR    left, right, ptr ;
        char    type[4];

        left = Projection() ;
        if (left == NULL)
                return(NULL);
        else
        {
                Word_Pos++;
                strcpy(type, Infix_op());
                if (strcmp(type,NULL) == 0)
                {
                        Free_tree(left);
                        return(NULL);
```

```
            }
            else
            {
                Word_Pos++;
                right = Projection();
                if (right == NULL)
                {
                    Free_tree(left);
                    return(NULL);
                }
                else
                {
                    ptr = Make_node(type,NULL);
                    ptr->left = left;
                    ptr->right = right;
                    return(ptr);
                }
            }
        }
    }

    TREE_PTR Projection(void)
    {
        TREE_PTR   left, right, ptr;
        int     ret;
        char    attr[AGB_LEN];
        char    ttr_name[AGB_LEN];

        strcpy(attr,"") ;
        left = Primitive();
        if (left == NULL)           /* primitive only or */
            return(NULL);/* primitive + '[' + attr_spec commalist + ']' */
        else
        {
            Word_Pos++;
            if (strcmp(Token_No[Word_Pos],"[") != 0)
            {
                /* if next word is not '[' move pointer back to old word */
                Word_Pos--;
                ptr = left;
                return(ptr);
            }
            else
            {
                do
                {
                    Word_Pos++;
                    strcpy(attr_name,"");
                    ret = Attr_spec(attr_name);
                    if (ret == FALSE)
                    {
                        Free_tree(left);
                        return(NULL);
                    }
                    else
                    {
                        strcat(attr,attr_name);
                        Word_Pos++;
                        if (strcmp(Token_No[Word_Pos],",") == 0)
                        /* if more than 1 attribute, continue */
                            strcat(attr,Token_No[Word_Pos]);
                    }
                } while (strcmp(Token_No[Word_Pos],",") == 0);
                if (strcmp(Token_No[Word_Pos],"]") != 0)
                {
                    Free_tree(left);
                    return(NULL);
                }
```

```
                else
                {
                        ptr = Make_node(PROJECT,attr);
                        ptr->left = left;
                        return(ptr);
                }
        }
    }
}

TREE_PTR Primitive(void)
{
    TREE_PTR    ptr ;
    char    rel[TOKEN_LEN] ;
    int     Iden_test ;

    /* if start with '(' primitive is expression then close with ')' */
    /* or primitive can be relation name */
    if (strncmp(Token_No[Word_Pos],"(",1) != 0)
    {
        Iden_test = Iden_name(rel) ;
        if (Iden_test == FALSE)
            return(NULL);
        else
        {
            ptr = Make_node(LEAF,rel) ;
            return(ptr) ;
        }
    }
    else
    {
        Word_Pos++ ;
        ptr = Expression() ;
        if (ptr == NULL)
            return(NULL);
        else
        {
            Word_Pos++;
            if (strncmp(Token_No[Word_Pos],")",1) != 0)
            {
                Free_tree(ptr) ;
                return(NULL) ;
            }
            else
                return(ptr) ;
        }
    }
}

int Select_pred(str)
char *str ;
{
    int     index ;
    char    cond[AGB_LEN] ;

    index = Word_Pos ;
    strcpy(cond,"") ;
    if (!SelPredUnit(cond))
    {
        Word_Pos = index ;
        return (FALSE) ;
    }
    else
    {
        strcpy(str,cond) ;
        Word_Pos++ ;
        if ((strcmp(Token_No[Word_Pos],"AND") == 0) || (strcmp(Token_No[Word_Pos],"OR")  == 0))
        {
```

```
                strcat(str," ") ;
                strcat(str,Token_No[Word_Pos]) ;
                Word_Pos++ ;
                if (Select_pred(cond))
                {
                        strcat(str," ") ;
                        strcat(str,cond) ;
                        return (TRUE) ;
                }
                else
                {
                        Word_Pos = index ;
                        return (FALSE) ;
                }
        }
        else
        {
                Word_Pos-- ;
                return (TRUE) ;
        }
    }
}

int SelPredUnit(str)
char *str ;
{
    int     index, mark ;
    char    cond[AGB_LEN] ;

    index = Word_Pos ;
    if (!Attr_spec(cond))       /* (FRIST) compare (SECOND) */
    {                   /* FIRST must be attribute_spec */
        Word_Pos = index ;
        if (strcmp(Token_No[Word_Pos],"(") != 0)
        {
            Word_Pos = index ;
            return (FALSE) ;
        }
        else
        {
            strcpy(str,"( ") ;
            Word_Pos++ ;
            if (!Select_pred(cond))
            {
                Word_Pos = index ;
                return (FALSE) ;
            }
            else
            {
                strcat(str,cond) ;
                Word_Pos++ ;
                if (strcmp(Token_No[Word_Pos],")") != 0)
                {
                    Word_Pos = index ;
                    return (FALSE) ;
                }
                else
                {
                    strcat(str," )") ;
                    return (TRUE) ;
                }
            }
        }
    }
    else                /* SECOND may be attribute_spec or constant */
    {
        strcpy(str,cond) ;
        Word_Pos++;
```

```
            if (!Compare(cond))
            {
                Word_Pos = index ;
                return(FALSE);
            }
            else
            {
                strcat(str," ") ;
                strcat(str,cond) ;
                mark = ++Word_Pos ;
                if (!Attr_spec(cond))
                {
                    Word_Pos = mark ;
                    if (!Constant(cond))
                    {
                        Word_Pos = index ;
                        return(FALSE) ;
                    }
                    else
                    {
                        strcat(str," ") ;
                        strcat(str,cond) ;
                        return(TRUE) ;
                    }
                }
                else
                {
                    strcat(str," ") ;
                    strcat(str,cond) ;
                    return(TRUE);
                }
            }
        }
}

int Attr_spec(str)
char *str ;
{
        char    att[AGB_LEN] ;

        strcpy(att,"") ;
        /* attribute name only or relation name + dot(.) + attribute name */
        if (Iden_name(att))
        {
            strcpy(str,att) ;
            Word_Pos++;
            if (strcmp(Token_No[Word_Pos],".") == 0)
            {
                strcat(str,Token_No[Word_Pos]) ;
                Word_Pos++ ;
                strcpy(att,"") ;
                if (!Iden_name(att))
                    return(FALSE);
                else
                {
                    strcat(str,att) ;
                    return(TRUE)  ;
                }
            }
            else
            {
                Word_Pos--;
                return(TRUE);
            }
        }
        else
            return(FALSE);
}
```

```
char *Infix_op(void)
{
        int     order;
        static char     *operator[MAXOP] = { "UNION","MINUS","INTERSECT","TIMES","JOIN","DIVIDEBY" };

        for (order=0;order<MAXOP;order++)
                if (strcmp(Token_No[Word_Pos],operator[order])==0)
                {
                        switch (order)
                        {
                                case 0 :return(UNION);
                                case 1 :return(MINUS);
                                case 2 :return(INTERSECT);
                                case 3 :return(TIMES);
                                case 4 :return(JOIN);
                                case 5 :return(DIVIDEBY);
                        }
                }
        return(NULL);
}

int Iden_name(str)
char *str ;
{
        /* reserve word detection for relation name in all query */
        /* relation name can't reserve word */

        if ((strcmp(Token_No[Word_Pos],"TIMES")==0) || (strcmp(Token_No[Word_Pos],"JOIN")==0) ||
        (strcmp(Token_No[Word_Pos],"MINUS")==0) || (strcmp(Token_No[Word_Pos],"INTERSECT")==0) ||
        (strcmp(Token_No[Word_Pos],"UNION")==0) || (strcmp(Token_No[Word_Pos],"DIVIDEBY")==0))
        {
                printf("Invalid table name\n") ;
                return(FALSE) ;
        }
        /* relation name must not begin with numeric */
        if (strspn(Token_No[Word_Pos],DIGITS) != 0)
                return(FALSE);
        else
        {
                if (strspn(Token_No[Word_Pos],Alpha_Num) != 0)
                {
                        strcpy(str,Token_No[Word_Pos]) ;
                        return(TRUE) ;
                }
                else
                        return(FALSE) ;
        }
}

int Compare(str)
char *str ;
{
        int     order ;
        static char     *comparator[MAXCOMP] = {"=","^=","<",">","<=",">="};

        /* if comparator is wrong, query will be incomplete */
        for (order=0;order<MAXCOMP;order++)
        {
                if (strcmp(Token_No[Word_Pos],comparator[order]) == 0)
                {
                        strcpy(str,Token_No[Word_Pos]) ;
                        return(TRUE) ;
                }
        }
        return (FALSE);
}
```

```
int Constant(str)
char *str ;
{
        /* String constant must be in ' ' but */
        /* Numeric constant shoule not. */
        if (strcmp(Token_No[Word_Pos],"") != 0)   /* numeric */
        {
                if ((Token_No[Word_Pos][0] == '+') || (Token_No[Word_Pos][0] == '-'))
                {
                        if (strspn(Token_No[Word_Pos]+1,DIGITS) == 0)
                                return (FALSE) ;
                        else
                        {
                                strcpy(str,Token_No[Word_Pos]) ;
                                return (TRUE) ;
                        }
                }
                else if (strspn(Token_No[Word_Pos],DIGITS) == 0)
                        return(FALSE);
                else
                {
                        strcpy(str,Token_No[Word_Pos]) ;
                        return(TRUE);
                }
        }
        else
        {
                strcpy(str,Token_No[Word_Pos]) ;
                Word_Pos++ ;
                strcat(str,Token_No[Word_Pos]) ;
                Word_Pos++ ;
                if(strcmp(Token_No[Word_Pos],"") == 0)
                {
                        strcat(str,Token_No[Word_Pos]);
                        return(TRUE) ;
                }
                else
                        return(FALSE);
        }
}

void con_back_end(sockfd,top,n_amount)
int sockfd;
TREE_PTR top;
int n_amount;
{
        char    sendline[MAXLINE], recline[MAXLINE+1];
        char    colname[30];
        int     n, m, i, colnum, *collen, *col0 ;

        send_query(sockfd,top,n_amount) ;
        n = readline(sockfd, recline, MAXLINE) ;
        recline[n] = '\0';
        if ( n == 0)
        {
                printf("con_back_end: connection terminated") ;
                exit(-1) ;
        }
        else if (n < 0)
                err_dump("con_back_end: readline error");
        if(strncmp(recline, OK, 3) == 0)
        {
                strcpy(sendline,CM);
                strcat(sendline,SEND) ;
                n = strlen(sendline);
                if(writen(sockfd, sendline, n) != n)
                        err_dump("con_back_end: writen error") ;
        }
```

```
        else
        {
                printf("%s\n",recline+3) ;
                return ;
        }
        receive_data(sockfd,recline) ;
        n = 3 ;
        memcpy(&colnum,recline+n,sizeof(int)) ;
        collen = (int *)malloc(colnum*sizeof(int)) ;
        n += sizeof(int) ;
        col0 = collen ;
        for (i=0;i<colnum;i++)
        {
                memcpy(collen,recline+n,sizeof(int)) ;
                n += sizeof(int) ;
                memcpy(&m,recline+n,sizeof(int)) ;
                n += sizeof(int) ;
                strncpy(sendline,recline+n,m) ;
                sendline[m] = '\0' ;
                n += m ;
                Lex__Attr(colname,sendline) ;
                printf("%-*s ",m,colname) ;
                collen++ ;
        }
        printf("\n") ;
        for (;;)
        {
                collen = col0 ;
                n = receive_data(sockfd,recline) ;
                if (n == 0)             /* DA */
                {
                        n = 3 ;
                        for (i=0;i<colnum;i++)
                        {
                                memcpy(&m,recline+n,sizeof(int)) ;
                                n += sizeof(int) ;
                                strncpy(sendline,recline+n,m) ;
                                sendline[m] = '\0' ;
                                n += m ;
                                printf("%-*s ",*collen,sendline) ;
                                collen++ ;
                        }
                        printf("\n") ;
                }
                else if (n == 1)
                {
                        pass_view(sockfd) ;
                        break ;
                }
                else
                        err_quit("algebra:  receive data error ") ;
        }
        free (collen) ;
}
```

2. ส่วนประมวลผลคำสั่งแบบกระจาย

file back.mak

```
TOOL=$(HOME)/prog/tool
OBJECT=back.o bk_main.o bk_sub1.o bk_sub2.o tool.o error.o
LIBS= $(ORACLE_HOME)/pro/libsql.a $(ORACLE_HOME)/oci/libhli.a $(ORACLE_HOME)/oci/libhlic.a

back_end    : $(OBJECT)
                cc $(CFLAGS) -o back_end $(OBJECT) $(LIBS)
bk_main.c   : bk_main.pc backprot.h backglb.h BK_GLB.H $(TOOL)/def.h
                $(TOOL)/precompi bk_main.pc
bk_sub1.c   : bk_sub1.pc backprot.h backxtrn.h BK_XTRN.H $(TOOL)/def.h
                $(TOOL)/precompi bk_sub1.pc
bk_sub2.c   : bk_sub2.pc backprot.h backxtrn.h BK_XTRN.H $(TOOL)/def.h
                $(TOOL)/precompi bk_sub2.pc
back.o      : back.c backprot.h $(TOOL)/def.h
                cc -c $(CFLAGS) back.c
bk_main.o   : bk_main.c
                cc -c $(CFLAGS) bk_main.c
bk_sub1.o   : bk_sub1.c
                cc -c $(CFLAGS) bk_sub1.c
bk_sub2.o   : bk_sub2.c
                cc -c $(CFLAGS) bk_sub2.c
tool.o      : $(TOOL)/tool.c  $(TOOL)/def.h
                cc -c $(CFLAGS) $(TOOL)/tool.c
error.o     : $(TOOL)/error.c $(TOOL)/def.h
                cc -c $(CFLAGS) $(TOOL)/error.c
```

file backproto.h

```
void        BackMain() ;
int         recv_auth() ;
TREE_PTR    adjust() ;
TREE_PTR    arrange() ;
void        clean_tab() ;
char        *exec_query() ;
int         RMexe_number() ;
void        fetch_sock() ;
void        fetch_tab() ;
int         LOexe_number() ;
int         LOexe_node() ;
int         CompLR() ;
char        *GetViewName() ;
char        *GetTabName() ;         /* temp table name */
void        GetNewAttr() ;
void        GetTabAttr() ;
void        GetAttr() ;
char        *LexTabAttr() ;
void        SendData() ;
void        mufwrn() ;
void        cleanUp() ;
void        descBind() ;
void        descSel() ;
void        doFetches() ;
void        fillSelDesc() ;
void        freeSelVars() ;
void        reptError() ;
int         min() ;
int         max() ;
int         MakeView() ;
int         ContProc() ;
int         PassToRM() ;
```

```
int         chkd_tab() ;
TREE_PTR    AdjToTree() ;
ADJ         *TreeToAdj() ;
int         Constraint() ;
int         ValidCons() ;
int         ValidUnit() ;
int         CompOpChk() ;
int         Numberic() ;
int         EquCons() ;
int         NotEquCons() ;
int         LessCons() ;
int         MoreCons() ;
int         LessEquCons() ;
int         MoreEquCons() ;
ADJ         *RuleChk() ;
void        FillAttr() ;
void        Free_Adj() ;
char        *LexAdj() ;
```

file backglb.h

```
char        local_host_name[30] ;
int         local_host_len ;
char        err_mesg[132] ;
int         err_flg = 0 ;
int         ex_flg = 0 ;
char        temp_table[30]          /* keep "TTEMP$(pid)" */
char        temp_view[30] ;         /* keep "VTEMP$(pid)" */
char        new_attr[20][30] ;
int         new_com[20] ;
int         n_new_attr ;
char        old_attr[20][30] ;
int         old_com[20] ;
int         n_old_attr ;
char        attr[20][30] ;
int         n_attr ;
char        vtemp_name[30][30] ;        /* arr of view name */
int         vt_order = 0 ;
char        ttemp_name[30][30] ;        /* arr of temp tab */
int         tt_order = 0 ;
desc_view   view[100] ;
int         v_order ;               /* point in view */
NAME        *POS ;
int         (*UnitComp[6])() = {EquCons,NotEquCons,LessCons,MoreCons,LessEquCons,MoreEquCons} ;
```

file backxtrn.h

```
extern char     local_host_name[30] ;
extern int      local_host_len ;
extern char     err_mesg[132] ;
extern int      err_flg ;
extern int      ex_flg ;
extern char     temp_table[30] ;        /* keep "TTEMP$(pid)" */
extern char     temp_view[30] ;         /* keep "VTEMP$(pid)" */
extern char     new_attr[20][30] ;
extern int      new_com[20] ;
extern int      n_new_attr ;
extern char     old_attr[20][30] ;
extern int      old_com[20] ;
extern int      n_old_attr ;
extern char     attr[20][30] ;
extern int      n_attr ;
extern char     vtemp_name[30][30] ;        /* arr of view name */
extern int      vt_order ;
extern char     ttemp_name[30][30] ;        /* arr of temp tab */
```

```
extern int          tt_order ;
extern desc_view    view[100] ;
extern int          v_order ;                   /* point in view */
extern NAME         *POS ;
extern int          (*UnitComp[6])() ;
```

file back.c

```
#include          "../tool/def.h"
#include          "../tool/proto.h"
#include          "backprot.h"

main(argc, argv)
int     argc;
char    *argv[];
{
        int     newsockfd, sockfd, servlen, clilen;
        struct sockaddr_un  cli_addr, serv_addr;
        int     childpid;

        if ((sockfd = socket(AF_UNIX, SOCK_STREAM, 0)) < 0)
                err_dump("back_end: can't open stream socket 0");
        memset((char *)&serv_addr, 0, sizeof(serv_addr));
        serv_addr.sun_family = AF_UNIX;
        strcpy(serv_addr.sun_path, UNIXSTR_PATH0);
        servlen = strlen(serv_addr.sun_path) + sizeof(serv_addr.sun_family);
        if (bind(sockfd, (struct sockaddr *)&serv_addr, servlen) < 0)
        {
                err_dump("back_end: can't bind local address 0");
        }
        listen(sockfd, 5);

        for(;;)
        {
                clilen = sizeof(cli_addr);
                newsockfd = accept(sockfd, (struct sockaddr *)&cli_addr, &clilen);
                if (newsockfd < 0)
                        err_dump("back_end: accept error");
                if ((childpid = fork()) < 0)
                        err_dump("back_end: fork error");
                else if (childpid == 0)
                {
                        close(sockfd);
                        /*--- do all ---*/
                        BackMain(newsockfd);
                        /*--------------*/
                        exit(0);
                }
                close(newsockfd);
        }
}
```

file BK_GLB.H

```
VARCHAR     uid[20] ;
VARCHAR     pwd[20] ;
VARCHAR     table_name[30] ;
char        host_name[30] ;
VARCHAR     user_name[30] ;
VARCHAR     attr_name[30] ;
int         colno ;
int         width ;
char        type[10] ;
char        exe_imm[132] ;
VARCHAR     stmt[512] ;
```

```
char        temp_name[30] ;
int         tuples_number ;
char        *stmt1 ;
float       more_than, more_equ, less_than, less_equ ;
short       more_thani,more_equi,less_thani,less_equi ;
```

file BK_XTRN.H

```
extern VARCHAR   uid[20] ;
extern VARCHAR   pwd[20] ;
extern VARCHAR   table_name[30] ;
extern char      host_name[30] ;
extern VARCHAR   user_name[30] ;
extern VARCHAR   attr_name[30] ;
extern int       colno ;
extern int       width ;
extern char      type[10] ;
extern char      exe_imm[132] ;
extern VARCHAR   stmt[512] ;
extern char      temp_name[30] ;
extern int       tuples_number ;
extern char      *stmt1 ;
extern float     more_than, more_equ, less_than, less_equ ;
extern short     more_thani,more_equi,less_thani,less_equi ;
```

file bk_main.pc

```
#include        "../tool/def.h"
#include        "../tool/proto.h"
#include        "backprot.h"
#include        "backglb.h"

EXEC SQL BEGIN DECLARE SECTION ;
EXEC SQL INCLUDE BK_GLB.H ;
EXEC SQL END DECLARE SECTION ;
EXEC SQL INCLUDE SQLCA ;

void BackMain(sockfd)
int sockfd;
{
        char    recline[MAXLINE+1], sendline[MAXLINE] ;
        int     n, n_amount, tuples, newsockfd ;
        TREE_PTR    qu_ptr, top_ptr, leaf_ptr ;
        char    site[30] ;

        if(gethostname(local_host_name,30-1) < 0)          /* get local host name */
                err_dump("gethostname error") ;
        recv_auth(sockfd) ;
        if (err_flg)
        {
                strcpy(sendline,ER) ;
                strcat(sendline,err_mesg) ;
                n = strlen(sendline) ;
                if (writen(sockfd, sendline, n) != n)
                        err_dump("BACK_END: writen error") ;
                exit (-1) ;
        }
        strcpy(sendline,OK) ;
        n = strlen(sendline) ;
        if (writen(sockfd, sendline, n) != n)
                err_dump("BACK_END: writen error") ;
        strcpy(temp_table,"TTEMP$") ;
        strcpy(temp_view,"VTEMP$") ;
        n = getpid() ;
        itoa(n,sendline) ;          /* n and sendline bellow just time */
```

```
strcat(temp_table,sendline) ;
strcat(temp_view, sendline) ;
while(!ex_flg)
{
        err_flg = 0 ;
        n = readline(sockfd, recline, MAXLINE) ;
        if ( n == 0)
        {
                printf("BackEnd: connection terminated\n") ;
                exit (-1) ;
        }
        else if (n < 0)
                err_dump("readline error");
        recline[n] = '\0' ;
        if (strncmp(QU,recline,3) == 0)
        {
                n_amount = atoi(recline+3) ;
                qu_ptr = receive_query(sockfd,n_amount) ;
                if (qu_ptr == NULL)
                        err_quit("BackEnd: receive query error\n") ;
                qu_ptr = adjust(qu_ptr) ;
                if (qu_ptr == NULL)
                {
                        strcpy(sendline,ER) ;
                        strcat(sendline,err_mesg) ;
                        n = strlen(sendline) ;
                        if (writen(sockfd, sendline, n) != n)
                                err_dump("BackEnd: writen error") ;
                        continue ;
                }
                strcpy(site,exec_query(qu_ptr,&tuples)) ;
                if (strcmp(site,local_host_name) != 0)
                {
                        tuples = RMexe_number(qu_ptr,&newsockfd,site) ;
                        if (tuples < 0)             /* err_flg is set too */
                        {
                                close (newsockfd) ;
                        }
                }
                if ((err_flg) || (strcmp(site,"ERROR") == 0))
                {
                        Free_tree(qu_ptr) ;
                        clean_tab() ;
                        strcpy(sendline,ER) ;
                        strcat(sendline,err_mesg) ;
                        n = strlen(sendline) ;
                        if (writen(sockfd, sendline, n) != n)
                        {
                                clean_tab() ;
                                err_dump("BackEnd: writen error") ;
                        }
                        continue ;
                }
                if (tuples < 0)
                {
                        strcpy(sendline,ER) ;
                        strcat(sendline,"BACK_END ERROR") ;
                        n = strlen(sendline) ;
                        if (writen(sockfd, sendline, n) != n)
                        {
                                clean_tab() ;
                                err_dump("BackEnd: writen error") ;
                        }
                        exit (-1) ;
                }
                else
                {
                        strcpy(sendline,OK) ;
```

```
                n = strlen(sendline) ;
                memcpy(sendline+n,&tuples,sizeof(tuples)) ;
                n += sizeof(tuples) ;
                if (writen(sockfd, sendline, n) != n)
                {
                        clean_tab() ;
                        err_dump("BackEnd: writen error") ;
                }
        }
}
else if (strncmp(CM,recline,3) == 0)
{
        if (strncmp(SEND,recline+3,3) == 0)  /* send all result data */
        {
                if (strcmp(site,local_host_name) == 0)
                {
                        fetch_tab(qu_ptr->item,sockfd) ;
                        Free_tree(qu_ptr) ;
                        clean_tab() ;
                        continue ;
                }
                else
                {
                        if (writen(newsockfd, recline, n) != n)
                                err_dump("Back_end: writen error\n") ;
                        PassToRM (sockfd, newsockfd) ;
                        close (newsockfd) ;
                        continue ;
                }
        }
        else            /* operate */
        {
                strcpy(sendline,OK) ;
                n = strlen(sendline) ;
                if (writen(sockfd, sendline, n) != n)
                {
                        clean_tab() ;
                        err_dump("BackEnd: writen error") ;
                }
                top_ptr = Make_node(LEAF,NULL) ;
                leaf_ptr = Make_node(LEAF,GetViewName()) ;
                MakeView(leaf_ptr->item,sockfd) ;
                if (recline[6] == 'L')
                {
                        top_ptr->left = leaf_ptr ;
                        top_ptr->right = qu_ptr ;
                }
                else
                {
                        top_ptr->left = qu_ptr ;
                        top_ptr->right = leaf_ptr ;
                }
                if (strncmp(UNION,recline+3,3) == 0)
                {
                        strcpy(top_ptr->type,UNION) ;
                }
                else if (strncmp(MINUS,recline+3,3) == 0)
                {
                        strcpy(top_ptr->type,MINUS) ;
                }
                else if (strncmp(INTERSECT,recline+3,3) == 0)
                {
                        strcpy(top_ptr->type,INTERSECT) ;
                }
                else if (strncmp(TIMES,recline+3,3) == 0)
                {
                        strcpy(top_ptr->type,TIMES) ;
                }
```

```
                        else if (strncmp(JOIN,recline+3,3) == 0)
                        {
                                strcpy(top_ptr->type,JOIN) ;
                        }
                        else if (strncmp(DIVIDEBY,recline+3,3) == 0)
                        {
                                strcpy(top_ptr->type,DIVIDEBY) ;
                        }
                        if (LOexe_node(top_ptr) < 0)
                        {
                                strcpy(site,"ERROR") ;
                                err_flg = 1 ;
                        }
                        else
                        {
                                tuples = LOexe_number(top_ptr->item) ;
                                strcpy(site,local_host_name) ;
                        }
                        if ((err_flg) || (strcmp(site,"ERROR") == 0))
                        {
                                Free_tree(top_ptr) ;
                                clean_tab() ;
                                strcpy(sendline,ER) ;
                                strcat(sendline,err_mesg) ;
                                n = strlen(sendline) ;
                                if (writen(sockfd, sendline, n) != n)
                                {
                                        clean_tab() ;
                                        err_dump("BackEnd: writen error") ;
                                }
                                continue ;
                        }
                        if (tuples < 0)
                        {
                                strcpy(sendline,ER) ;
                                strcat(sendline,"BACK_END ERROR") ;
                                n = strlen(sendline) ;
                                if (writen(sockfd, sendline, n) != n)
                                {
                                        clean_tab() ;
                                        err_dump("BackEnd: writen error") ;
                                }
                                exit (-1) ;
                        }
                        fetch_tab(top_ptr->item,sockfd) ;
                        Free_tree(top_ptr) ;
                        clean_tab() ;
                        exit (0) ;
                }
        }
        else if (strncmp(EXIT,recline,3) == 0)
        {
                ex_flg = 1 ;
        }
    }
    clean_tab() ;
    exit (0) ;
}

int recv_auth(sockfd)
int sockfd ;
{
    char   recline[MAXLINE+1], sendline[MAXLINE] ;
    int    n ;

    n = readline(sockfd, recline, MAXLINE) ;
    if ( n == 0)
    {
```

```
                printf("BackEnd: connection terminated") ;
                exit(-1) ;
        }
        else if (n < 0)
                err_dump("BackEnd: readline error");
        recline[n] = '\0';
        if (strncmp(recline, US, 3) != 0)
                err_quit("BackEnd: protocal error");
        strcpy(uid.arr,recline+3) ;
        uid.len = strlen(uid.arr) ;
        n = readline(sockfd, recline, MAXLINE) ;
        if ( n == 0)
        {
                printf("BackEnd: connection terminated") ;
                exit(-1) ;
        }
        else if (n < 0)
                err_dump("BackEnd: readline error");
        recline[n] = '\0';
        if(strncmp(recline, PW, 3) != 0)
                err_quit("BackEnd: protocal error");
        strcpy(pwd.arr,recline+3) ;
        pwd.len = strlen(pwd.arr) ;
        EXEC SQL WHENEVER SQLERROR GOTO error ;
        EXEC SQL CONNECT :uid IDENTIFIED BY :pwd ;
        EXEC SQL WHENEVER SQLERROR CONTINUE ;
        return (0) ;
error:
        printf("%.70s\n",sqlca.sqlerrm.sqlerrmc) ;
        strcpy(err_mesg,"CONNECT TO DBMS FAILURE") ;
        err_flg = 1 ;
}

/* Exec_query return number of tuples by reference    */
/* and return data site */

char *exec_query(top_ptr,tuples_ptr)
TREE_PTR    top_ptr ;
int     *tuples_ptr ;
{
        int     Lnumber, Rnumber, sockfd ;
        char    L_site[30], R_site[30] ;

        if (top_ptr->left != NULL)
                strcpy(L_site,exec_query(top_ptr->left,&Lnumber)) ;
        else strcpy(L_site,"") ;
        if (top_ptr->right != NULL)
                strcpy(R_site,exec_query(top_ptr->right,&Rnumber)) ;
        else strcpy(R_site,"") ;
        if ((strcmp(L_site,"ERROR") == 0) || (strcmp(R_site,"ERROR") == 0))
        {
                err_flg = 1 ;
                return ("ERROR") ;
        }
        else if ((top_ptr->left == NULL) && (top_ptr->right == NULL))   /* leaf */
        {
                strcpy(table_name.arr,top_ptr->item) ;
                table_name.len = strlen(table_name.arr) ;
                EXEC SQL SELECT HOST_NAME INTO :host_name FROM G$THU$
                        WHERE TABLE_NAME = :table_name ;
                LexWord(host_name,host_name) ;
                if (strcmp(host_name,local_host_name) == 0)  /* local node */
                {
                        if (LOexe_node(top_ptr) < 0)
                                return ("ERROR") ;
                        *tuples_ptr = LOexe_number(top_ptr->item) ;
                        return (local_host_name) ;
                }
```

```
        else                    /* remote node */
                return (host_name) ;
}
else if ((strcmp(L_site,R_site) == 0) && (top_ptr->left != NULL))        /* L = R */
{
        if (strcmp(R_site,local_host_name) != 0)            /* L,R = remote */
                return(L_site) ;
        else                                        /* L,R = local */
        {
                if (LOexe_node(top_ptr) < 0)
                        return ("ERROR") ;
                *tuples_ptr = LOexe_number(top_ptr->item) ;
                return (local_host_name) ;
        }
}
else if (strcmp(L_site,local_host_name) == 0)                    /* L != R and L = local */
{
        if (strcmp(R_site,"") == 0)                     /* R = NULL */
        {
                if (LOexe_node(top_ptr) < 0)
                        return ("ERROR") ;
                *tuples_ptr = LOexe_number(top_ptr->item) ;
                return (local_host_name) ;
        }
        else if (strcmp(R_site,local_host_name) != 0)                /* R = remote */
        {
                Rnumber = RMexe_number(top_ptr->right,&sockfd,R_site) ;
                if (Rnumber < 0)
                        return ("ERROR") ;
                if (CompLR(Lnumber,top_ptr->left->item,top_ptr->type,Rnumber,top_ptr->right->item,0) == 0)
                {
                        fetch_sock(top_ptr->right,sockfd) ;
                        if (LOexe_node(top_ptr) < 0)
                                return ("ERROR") ;
                        *tuples_ptr = LOexe_number(top_ptr->item) ;
                        return (local_host_name) ;
                }
                else                    /* continue exec. at remote site */
                {
                        *tuples_ptr = ContProc(sockfd,top_ptr,'L') ;
                        return (local_host_name) ;
                }
        }
}
else                            /* L != R and L = remote */
{
        if (strcmp(R_site,local_host_name) == 0)                /* R = local */
        {
                Lnumber = RMexe_number(top_ptr->left,&sockfd,L_site) ;
                if (Lnumber < 0)
                        return ("ERROR") ;
                if (CompLR(Lnumber,top_ptr->left->item,top_ptr->type,Rnumber,top_ptr->right->item,1) == 1)
                {
                        fetch_sock(top_ptr->left,sockfd) ;
                        if (LOexe_node(top_ptr) < 0)
                                return ("ERROR") ;
                        *tuples_ptr = LOexe_number(top_ptr->item) ;
                        return (local_host_name) ;
                }
                else                    /* continue exec. at remote site */
                {
                        *tuples_ptr = ContProc(sockfd,top_ptr,'R') ;
                        return (local_host_name) ;
                }
        }
        else if (strcmp(R_site,"") == 0)                /* R = NULL */
        {
                return (L_site) ;
```

```
                }
                else if (strcmp(R_site,local_host_name) != 0)          /* L != R */
                {                                       /* L and R = remote*/
                        Lnumber = RMexe_number(top_ptr->left,&sockfd,L_site) ;
                        if (Lnumber > 0)
                                return ("ERROR") ;
                        fetch_sock(top_ptr->left,sockfd) ;
                        Rnumber = RMexe_number(top_ptr->right,&sockfd,R_site) ;
                        if (Rnumber < 0)
                                return ("ERROR") ;
                        fetch_sock(top_ptr->right,sockfd) ;
                        if (LOexe_node (top_ptr) < 0)
                                return ("ERROR") ;
                        *tuples_ptr = LOexe_number(top_ptr->item) ;
                        return (local_host_name) ;
                }
        }
}

/* error return < 0 ,put err_mesg, set err_flg */

int RMexe_number(top,sockfd_ptr,hostname)
TREE_PTR    top ;
int     *sockfd_ptr ;
char    *hostname ;
{
        char    sendline[MAXLINE], recline[MAXLINE+1];
        int     order, number ;
        struct sockaddr_un  serv_addr ;
        int     servlen, sockfd, n ;

        order = 0 ;
        order = Walk(top,order) ;
        memset((char *)&serv_addr, 0, sizeof(serv_addr)) ;
        serv_addr.sun_family = AF_UNIX ;
        strcpy(serv_addr.sun_path, UNIXSTR_PATH1) ;
        servlen = strlen(serv_addr.sun_path) + sizeof(serv_addr.sun_family) ;

        if ((sockfd = socket(AF_UNIX, SOCK_STREAM, 0)) < 0)
                err_sys("BACK_END: can't open stream socket") ;
        if (connect(sockfd, (struct sockaddr *)&serv_addr, servlen) < 0)
                err_sys("BACK_END: can't connect to network server");
        *sockfd_ptr = sockfd ;
        strcpy(sendline,hostname) ;
        n = strlen(sendline) ;
        if (writen(sockfd, sendline, n) != n)
                err_dump("BACK_END: writen error") ;
        strcpy(sendline,US);
        strcat(sendline,uid.arr);
        n = strlen(sendline);
        if (writen(sockfd, sendline, n) != n)
                err_dump("BACK_END: writen error") ;
        strcpy(sendline,PW);
        strcat(sendline,pwd.arr);
        n = strlen(sendline);
        if (writen(sockfd, sendline, n) != n)
                err_dump("BACK_END: writen error") ;
        n = readline(sockfd, recline, MAXLINE) ;
        recline[n] = '\0';
        if ( n == 0)
        {
                printf("BACK_END: connection terminated") ;
                exit(-1) ;
        }
        else if (n < 0)
                err_dump("BACK_END: readline error");
        if (strncmp(recline, OK, 3) == 0)
                send_query(sockfd,top,order) ;
```

```
        else
        {
                printf("%s\n",recline+3);
                exit(-1);
        }
        n = readline(sockfd, recline, MAXLINE) ;
        recline[n] = '\0';
        if ( n == 0)
        {
                printf("con_back_end: connection terminated") ;
                exit(-1) ;
        }
        else if (n < 0)
                err_dump("BACK_END: readline error");
        if (strncmp(recline, OK, 3) == 0)
        {
                memcpy(&number,recline+3,sizeof(int)) ;
                return (number) ;
        }
        else
        {
                err_flg = 1 ;
                strcpy(err_mesg,"Remote Back_End : ") ;
                strcat(err_mesg,recline+3) ;
                return (-1) ;
        }
}

void fetch_sock(top_ptr,sockfd)
TREE_PTR    top_ptr ;
int     sockfd ;
{
        char    sendline[MAXLINE], recline[MAXLINE+1] ;
        int     n ;

        strcpy(top_ptr->item,GetViewName()) ;
        strcpy(sendline,CM) ;
        strcat(sendline,SEND) ;
        n = strlen(sendline) ;
        if (writen(sockfd,sendline,n) != n)
                err_dump("BACK_END: writen error") ;
        MakeView(top_ptr->item,sockfd) ;
        strcpy(top_ptr->type,LEAF) ;
        if (top_ptr->left != NULL)
                Free_tree(top_ptr->left) ;
        top_ptr->left = NULL ;
        if (top_ptr->right != NULL)
                Free_tree(top_ptr->right) ;
        top_ptr->right = NULL ;
        close (sockfd) ;
}

void fetch_tab(table,sockfd)
char    *table ;
int     sockfd ;
{
        SendData(table,sockfd) ;
        SendView(table,sockfd) ;
        return ;
error:
        clean_tab() ;
        exit (-1) ;
}

/* if error return < 0 */

int LOexe_node(top_ptr)
TREE_PTR    top_ptr ;
```

```
{
    char   view_name[30] ;
    int    i, j, k, n, join_flg ;
    char   attr0[30], attr1[30], attr2[30] ;
    char   where[256], *sel_ptr;

    EXEC SQL WHENEVER SQLERROR STOP ;
    if (strcmp(top_ptr->type,LEAF) == 0)
    {
        strcpy(view_name,GetViewName()) ;
        GetNewAttr(top_ptr->item) ;
        strcpy(stmt.arr,"CREATE VIEW ") ;
        strcat(stmt.arr,view_name) ;
        strcat(stmt.arr," (") ;
        for (i=0;i<n_new_attr;i++)
        {
            strcat(stmt.arr,new_attr[i]) ;
            if (i<n_new_attr-1)
                strcat(stmt.arr,",") ;
        }
        strcat(stmt.arr,") AS SELECT DISTINCT ") ;
        GetTabAttr(top_ptr->item) ;
        for (i=0;i<n_old_attr;i++)
        {
            strcat(stmt.arr,old_attr[i]) ;
            if (i<n_old_attr-1)
                strcat(stmt.arr,",") ;
        }
        strcat(stmt.arr," FROM ") ;
        strcpy(table_name.arr,top_ptr->item) ;
        table_name.len = strlen(table_name.arr) ;
        EXEC SQL SELECT USER_NAME INTO :user_name
            FROM G$THU$
            WHERE TABLE_NAME = :table_name ;
        n = strlen(stmt.arr) ;
        strncat(stmt.arr,user_name.arr,user_name.len) ;
        n += user_name.len ;
        stmt.arr[n] = '\0' ;
        strcat(stmt.arr,".") ;
        strcat(stmt.arr,top_ptr->item) ;
        stmt.len = strlen(stmt.arr) ;
        EXEC SQL EXECUTE IMMEDIATE :stmt ;
        for (i=0;i<n_old_attr;i++)
        {
            strcpy(view[v_order].view_name,view_name) ;
            strcpy(view[v_order].new_attr,new_attr[i]) ;
            strcpy(view[v_order].old_tab,top_ptr->item) ;
            strcpy(view[v_order].old_attr,old_attr[i]) ;
            view[v_order].order = i+1 ;
            v_order++ ;
        }
        strcpy(top_ptr->item,view_name) ;
        EXEC SQL COMMIT WORK ;
        return (0) ;
    }
    else if (strcmp(top_ptr->type,UNION) == 0)
    {
        GetAttr(top_ptr->left->item) ;
        for (i=0;i<n_attr;i++)
            strcpy(new_attr[i],attr[i]) ;
        n_new_attr = n_attr ;
        GetAttr(top_ptr->right->item) ;
        for (i=0;i<n_attr;i++)
            strcpy(old_attr[i],attr[i]) ;
        n_old_attr = n_attr ;
        if (n_new_attr != n_old_attr)
            return (-1) ;
        strcpy(view_name,GetViewName()) ;
```

```
        strcpy(stmt.arr,"CREATE VIEW ") ;
        strcat(stmt.arr,view_name) ;
        strcat(stmt.arr," AS SELECT ") ;
        for (i=0;i<n_new_attr;i++)
        {
                strcat(stmt.arr,new_attr[i]) ;
                if (i<n_new_attr-1)
                        strcat(stmt.arr,",") ;
        }
        strcat(stmt.arr," FROM ") ;
        strcat(stmt.arr,top_ptr->left->item) ;
        strcat(stmt.arr," UNION SELECT ") ;
        for (i=0;i<n_new_attr;i++)
        {
                for (k=0;k<v_order;k++)
                {
                        if (strcmp(new_attr[i],view[k].new_attr) == 0)
                        {
                                strcpy(attr1,view[k].old_attr) ;
                                break ;
                        }
                }
                for (j=0;j<n_old_attr;j++)
                {
                        for (k=0;k<v_order;k++)
                        {
                                if (strcmp(old_attr[j],view[k].new_attr) == 0)
                                {
                                        strcpy(attr2,view[k].old_attr) ;
                                        break ;
                                }
                        }
                        if (strcmp(attr1,attr2) == 0)
                        {
                                strcat(stmt.arr,old_attr[j]) ;
                                if (j<n_old_attr-1)
                                        strcat(stmt.arr,",") ;
                                break ;
                        }
                }
                if (j == n_old_attr)
                        return (-1) ;
        }
        strcat(stmt.arr," FROM ") ;
        strcat(stmt.arr,top_ptr->right->item) ;
        stmt.len = strlen(stmt.arr) ;
        EXEC SQL EXECUTE IMMEDIATE :stmt ;
        for (i=0;i<v_order;i++)
        {
                if (strcmp(view[i].view_name,top_ptr->left->item) == 0)
                        strcpy(view[i].view_name,view_name) ;
                else if (strcmp(view[i].view_name,top_ptr->right->item) == 0)
                {
                        strcpy(view[i].view_name,view_name) ;
                        for (j=0;j<n_new_attr;j++)
                                if (strcmp(view[i].new_attr,new_attr[j]) == 0)
                        view[i].order = j+1 ;
                }
        }
        strcpy(top_ptr->item,view_name) ;
        strcpy(top_ptr->type,LEAF) ;
        Free_node(top_ptr->left) ;
        top_ptr->left = NULL ;
        Free_node(top_ptr->right) ;
        top_ptr->right = NULL ;
        EXEC SQL COMMIT WORK ;
        return (0) ;
}
```

```
else if (strcmp(top_ptr->type,MINUS) == 0)
{
        GetAttr(top_ptr->left->item) ;
        for (i=0;i<n_attr;i++)
                strcpy(new_attr[i],attr[i]) ;
        n_new_attr = n_attr ;
        GetAttr(top_ptr->right->item) ;
        for (i=0;i<n_attr;i++)
                strcpy(old_attr[i],attr[i]) ;
        n_old_attr = n_attr ;
        if (n_new_attr != n_old_attr)
                return (-1) ;
        strcpy(view_name,GetViewName()) ;
        strcpy(stmt.arr,"CREATE VIEW ") ;
        strcat(stmt.arr,view_name) ;
        strcat(stmt.arr," AS SELECT ") ;
        for (i=0;i<n_new_attr;i++)
        {
                strcat(stmt.arr,new_attr[i]) ;
                if (i<n_new_attr-1)
                        strcat(stmt.arr,",") ;
        }
        strcat(stmt.arr," FROM ") ;
        strcat(stmt.arr,top_ptr->left->item) ;
        strcat(stmt.arr," MINUS SELECT ") ;
        for (i=0;i<n_new_attr;i++)
        {
                for (k=0;k<v_order;k++)
                {
                        if (strcmp(new_attr[i],view[k].new_attr) == 0)
                        {
                                strcpy(attr1,view[k].old_attr) ;
                                break ;
                        }
                }
                for (j=0;j<n_old_attr;j++)
                {
                        for (k=0;k<v_order;k++)
                        {
                                if (strcmp(old_attr[j],view[k].new_attr) == 0)
                                {
                                        strcpy(attr2,view[k].old_attr) ;
                                        break ;
                                }
                        }
                        if (strcmp(attr1,attr2) == 0)
                        {
                                strcat(stmt.arr,old_attr[j]) ;
                                if (j<n_old_attr-1)
                                        strcat(stmt.arr,",") ;
                                break ;
                        }
                }
                if (j == n_old_attr)
                        return (-1) ;
        }
        strcat(stmt.arr," FROM ") ;
        strcat(stmt.arr,top_ptr->right->item) ;
        stmt.len = strlen(stmt.arr) ;
        EXEC SQL EXECUTE IMMEDIATE :stmt ;
        for (i=0;i<v_order;i++)
        {
                if (strcmp(view[i].view_name,top_ptr->left->item) == 0)
                        strcpy(view[i].view_name,view_name) ;
                else if (strcmp(view[i].view_name,top_ptr->right->item) == 0)
                {
                        strcpy(view[i].view_name,view_name) ;
                        for (j=0;j<n_new_attr;j++)
```

```
                        if (strcmp(view[i].new_attr,new_attr[j]) == 0)
                            view[i].order = j+1 ;
                }
        }
        strcpy(top_ptr->item,view_name) ;
        strcpy(top_ptr->type,LEAF) ;
        Free_node(top_ptr->left) ;
        top_ptr->left = NULL ;
        Free_node(top_ptr->right) ;
        top_ptr->right = NULL ;
        EXEC SQL COMMIT WORK ;
        return (0) ;
}
else if (strcmp(top_ptr->type,INTERSECT) == 0)
{
        GetAttr(top_ptr->left->item) ;
        for (i=0;i<n_attr;i++)
                strcpy(new_attr[i],attr[i]) ;
        n_new_attr = n_attr ;
        GetAttr(top_ptr->right->item) ;
        for (i=0;i<n_attr;i++)
                strcpy(old_attr[i],attr[i]) ;
        n_old_attr = n_attr ;
        if (n_new_attr != n_old_attr)
                return (-1) ;
        strcpy(view_name,GetViewName()) ;
        strcpy(stmt.arr,"CREATE VIEW ") ;
        strcat(stmt.arr,view_name) ;
        strcat(stmt.arr," AS SELECT ") ;
        for (i=0;i<n_new_attr;i++)
        {
                strcat(stmt.arr,new_attr[i]) ;
                if (i<n_new_attr-1)
                        strcat(stmt.arr,",") ;
        }
        strcat(stmt.arr," FROM ") ;
        strcat(stmt.arr,top_ptr->left->item) ;
        strcat(stmt.arr," INTERSECT SELECT ") ;
        for (i=0;i<n_new_attr;i++)
        {
                for (k=0;k<v_order;k++)
                {
                        if (strcmp(new_attr[i],view[k].new_attr) == 0)
                        {
                                strcpy(attr1,view[k].old_attr) ;
                                break ;
                        }
                }
                for (j=0;j<n_old_attr;j++)
                {
                        for (k=0;k<v_order;k++)
                        {
                                if (strcmp(old_attr[j],view[k].new_attr) == 0)
                                {
                                        strcpy(attr2,view[k].old_attr) ;
                                        break ;
                                }
                        }
                        if (strcmp(attr1,attr2) == 0)
                        {
                                strcat(stmt.arr,old_attr[j]) ;
                                if (j<n_old_attr-1)
                                        strcat(stmt.arr,",") ;
                                break ;
                        }
                }
                if (j == n_old_attr)
                        return (-1) ;
```

```
        }
        strcat(stmt.arr," FROM ") ;
        strcat(stmt.arr,top_ptr->right->item) ;
        stmt.len = strlen(stmt.arr) ;
        EXEC SQL EXECUTE IMMEDIATE :stmt ;
        for (i=0;i<v_order;i++)
        {
                if (strcmp(view[i].view_name,top_ptr->left->item) == 0)
                        strcpy(view[i].view_name,view_name) ;
                else if (strcmp(view[i].view_name,top_ptr->right->item) == 0)
                {
                        strcpy(view[i].view_name,view_name) ;
                        for (j=0;j<n_new_attr;j++)
                                if (strcmp(view[i].new_attr,new_attr[j]) == 0)
                                        view[i].order = j+1 ;
                }
        }
        strcpy(top_ptr->item,view_name) ;
        strcpy(top_ptr->type,LEAF) ;
        Free_node(top_ptr->left) ;
        top_ptr->left = NULL ;
        Free_node(top_ptr->right) ;
        top_ptr->right = NULL ;
        EXEC SQL COMMIT WORK ;
        return (0) ;
}
else if (strcmp(top_ptr->type,TIMES) == 0)
{
        strcpy(view_name,GetViewName()) ;
        strcpy(stmt.arr,"CREATE VIEW ") ;
        strcat(stmt.arr,view_name) ;
        strcat(stmt.arr," AS SELECT * FROM ") ;
        strcat(stmt.arr,top_ptr->left->item) ;
        strcat(stmt.arr,",") ;
        strcat(stmt.arr,top_ptr->right->item) ;
        stmt.len = strlen(stmt.arr) ;
        EXEC SQL EXECUTE IMMEDIATE :stmt ;
        GetAttr(top_ptr->left->item) ;
        for (i=0;i<v_order;i++)
        {
                if (strcmp(view[i].view_name,top_ptr->left->item) == 0)
                        strcpy(view[i].view_name,view_name) ;
                else if (strcmp(view[i].view_name,top_ptr->right->item) == 0)
                {
                        strcpy(view[i].view_name,view_name) ;
                        view[i].order = n_attr + view[i].order ;
                }
        }
        strcpy(top_ptr->item,view_name) ;
        strcpy(top_ptr->type,LEAF) ;
        Free_node(top_ptr->left) ;
        top_ptr->left = NULL ;
        Free_node(top_ptr->right) ;
        top_ptr->right = NULL ;
        EXEC SQL COMMIT WORK ;
        return (0) ;
}
else if (strcmp(top_ptr->type,JOIN) == 0)
{
        join_flg = 0 ;
        strcpy(where,"") ;
        for (i=0;i<20;i++)
                old_com[i] = new_com[i] = 0 ;
        GetAttr(top_ptr->left->item) ;
        n_new_attr = n_attr ;
        for (i=0;i<n_new_attr;i++)
                strcpy(new_attr[i],attr[i]) ;
        GetAttr(top_ptr->right->item) ;
```

```
n_old_attr = n_attr ;
for (i=0;i<n_old_attr;i++)
        strcpy(old_attr[i],attr[i]) ;
for (i=0;i<n_new_attr;i++)
{
        for (k=0;k<v_order;k++)
        {
                if ((strcmp(view[k].view_name,top_ptr->left->item) == 0)
                && (strcmp(view[k].new_attr,new_attr[i]) == 0))
                {
                        strcpy(attr1,view[k].old_attr) ;
                        break ;
                }
        }
        for (j=0;j<n_old_attr;j++)
        {
                for (k=0;k<v_order;k++)
                {
                        if ((strcmp(view[k].view_name,top_ptr->right->item) == 0)
                        && (strcmp(view[k].new_attr,old_attr[j]) == 0))
                        {
                                strcpy(attr2,view[k].old_attr) ;
                                break ;
                        }
                }
                if (strcmp(attr1,attr2) == 0)
                {
                        join_flg = 1 ;
                        new_com[i] = 1 ;
                        old_com[j] = 1 ;
                        strcat(where," ") ;
                        strcat(where,top_ptr->left->item) ;
                        strcat(where,".") ;
                        strcat(where,new_attr[i]) ;
                        strcat(where," = ") ;
                        strcat(where,top_ptr->right->item) ;
                        strcat(where,".") ;
                        strcat(where,old_attr[j]) ;
                        strcat(where," AND") ;
                }
        }
}
if (join_flg)
{
        k = strlen(where) ;
        where[k-3] = '\0' ;                     /* delete last AND */
}
strcpy(view_name,GetViewName()) ;
strcpy(stmt.arr,"CREATE VIEW ") ;
strcat(stmt.arr,view_name) ;
strcat(stmt.arr," (") ;
for (i=0;i<n_new_attr;i++)
{
        strcat(stmt.arr,new_attr[i]) ;
        if (i<n_new_attr-1)
                strcat(stmt.arr,",") ;
}
for (j=0;j<n_old_attr;j++)
{
        if (old_com[j] == 0)                    /* none commond attr */
        {
                strcat(stmt.arr,",") ;
                strcat(stmt.arr,old_attr[j]) ;
        }
}
strcat(stmt.arr,")") ;
strcat(stmt.arr," AS SELECT ") ;
for (i=0;i<n_new_attr;i++)
```

```
        {
                strcat(stmt.arr,new_attr[i]) ;
                if (i<n_new_attr-1)
                        strcat(stmt.arr,",") ;
        }
        for (j=0;j<n_old_attr;j++)
        {
                if (old_com[j] == 0)
                {
                        strcat(stmt.arr,",") ;
                        strcat(stmt.arr,old_attr[j]) ;
                }
        }
        strcat(stmt.arr," FROM ") ;
        strcat(stmt.arr,top_ptr->left->item) ;
        strcat(stmt.arr,",") ;
        strcat(stmt.arr,top_ptr->right->item) ;
        if (join_flg)
        {
                strcat(stmt.arr," ") ;
                strcat(stmt.arr,"WHERE") ;
                strcat(stmt.arr,where) ;
        }
        stmt.len = strlen(stmt.arr) ;
        EXEC SQL EXECUTE IMMEDIATE :stmt ;
        for (i=0;i<v_order;i++)
        {
                if (strcmp(view[i].view_name,top_ptr->left->item) == 0)
                        strcpy(view[i].view_name,view_name) ;
                else if (strcmp(view[i].view_name,top_ptr->right->item) == 0)
                {
                        k = 0 ;
                        for (j=0;j<n_old_attr;j++)
                        {
                                if (old_com[j] == 0)
                                {
                                        k++ ;
                                        if (strcmp(view[i].new_attr,old_attr[j]) ==0)
                                        {
                                                strcpy(view[i].view_name,view_name) ;
                                                view[i].order = n_new_attr + k ;
                                                break ;
                                        }
                                }
                        }
                }
        }
        Free_node(top_ptr->left) ;
        Free_node(top_ptr->right) ;
        top_ptr->left = NULL ;
        top_ptr->right = NULL ;
        strcpy(top_ptr->type,LEAF) ;
        strcpy(top_ptr->item,view_name) ;
        EXEC SQL COMMIT WORK ;
        return (0) ;
}
else if (strcmp(top_ptr->type,DIVIDEBY) == 0)
{
        GetAttr(top_ptr->left->item) ;
        for (i=0;i<n_attr;i++)
        {
                strcpy(new_attr[i],attr[i]) ;
                new_com[i] = 0 ;
        }
        n_new_attr = n_attr ;
        GetAttr(top_ptr->right->item) ;
        for (i=0;i<n_attr;i++)
        {
```

```
        strcpy(old_attr[i],attr[i]) ;
        old_com[i] = 0 ;
}
n_old_attr = n_attr ;
if (n_new_attr <= n_old_attr)
{
        strcpy(err_mesg,"DIVIDEBY NO COMMOND ATTRIBUTE") ;
        return (-1) ;
}
for (i=0;i<n_old_attr;i++)
{
        for (k=0;k<v_order;k++)
                if ((strcmp(view[k].new_attr,old_attr[i]) == 0)
                && (strcmp(view[k].view_name,top_ptr->right->item) == 0))
                {
                        strcpy(attr2,view[k].old_attr) ;
                        break ;
                }
        for (j=0;j<n_new_attr;j++)
        {
                for (k=0;k<v_order;k++)
                        if ((strcmp(view[k].new_attr,new_attr[j]) == 0)
                        && (strcmp(view[k].view_name,top_ptr->left->item) == 0))
                        {
                                strcpy(attr1,view[k].old_attr) ;
                                break ;
                        }
                if (strcmp(attr1,attr2) == 0)
                {
                        new_com[j] = i+1 ;              /* commond attr */
                        break ;
                }
        }
        if (j == n_new_attr)
        {
                strcpy(err_mesg,"All attr.'s of Divisor are not commond attr.'s") ;
                return (-1) ;
        }
}
strcpy(view_name,GetViewName()) ;
strcpy(stmt.arr,"CREATE VIEW ") ;
strcat(stmt.arr,view_name) ;
strcat(stmt.arr," (") ;
strcpy(exe_imm,"") ;
for (i=0;i<n_new_attr;i++)
{
        if (new_com[i] == 0)                    /* none commond attr */
        {
                strcat(exe_imm,new_attr[i]) ;
                strcat(exe_imm,",") ;
        }
}
i = strlen(exe_imm) ;
exe_imm[i-1] = '\0' ;
strcat(stmt.arr,exe_imm) ;
strcat(stmt.arr,") AS SELECT ") ;
strcat(stmt.arr,exe_imm) ;
strcat(stmt.arr," FROM ") ;
strcat(stmt.arr,top_ptr->left->item) ;
strcat(stmt.arr,",") ;
strcat(stmt.arr,top_ptr->right->item) ;
strcat(stmt.arr," WHERE ") ;
strcpy(where,"") ;
for (i=0;i<n_old_attr;i++)
{
        for (j=0;j<n_new_attr;j++)
                if (new_com[j] == i+1 )
                {
```

```
                        strcat(where,top_ptr->left->item) ;
                        strcat(where,".") ;
                        strcat(where,new_attr[j]) ;
                        strcat(where," = ") ;
                        strcat(where,top_ptr->right->item) ;
                        strcat(where,".") ;
                        strcat(where,old_attr[i]) ;
                        strcat(where," ") ;
                        break ;
                }
            if (i<n_old_attr-1)
                strcat(where,"AND ") ;
        }
        strcat(stmt.arr,where) ;
        strcat(stmt.arr,"GROUP BY ") ;
        strcat(stmt.arr,exe_imm) ;
        strcat(stmt.arr," HAVING COUNT(*) = (SELECT COUNT(*) FROM ") ;
        strcat(stmt.arr,top_ptr->right->item) ;
        strcat(stmt.arr,")") ;
        stmt.len = strlen(stmt.arr) ;
        EXEC SQL EXECUTE IMMEDIATE :stmt ;
        k = 0 ;
        for (i=0;i<n_new_attr;i++)
        {
            if (new_com[i] == 0)
            {
                k++ ;
                for (j=0;j<v_order;j++)
                    if ((strcmp(view[j].view_name,top_ptr->left->item) == 0)
                    && (strcmp(view[j].new_attr,new_attr[i]) == 0))
                    {
                        strcpy(view[j].view_name,view_name) ;
                        view[j].order = k ;
                    }
            }
        }
        Free_node(top_ptr->left) ;
        top_ptr->left = NULL ;
        Free_node(top_ptr->right) ;
        top_ptr->right = NULL ;
        strcpy(top_ptr->item,view_name) ;
        strcpy(top_ptr->type,LEAF) ;
        EXEC SQL COMMIT WORK ;
        return (0) ;
    }
    else if (strcmp(top_ptr->type,SELECT) == 0)
    {
        strcpy(where,"") ;
        sel_ptr = top_ptr->item ;
        do
        {
            sel_ptr = LexSel(attr0,sel_ptr) ;
            switch (attr0[0])
            {
                case '+' :
                case '-' :
                case '0' :
                case '1' :
                case '2' :
                case '3' :
                case '4' :
                case '5' :
                case '6' :
                case '7' :
                case '8' :
                case '9' :
                case '.' :
                case '(' :
```

```
                case ')' :
                case '=' :
                case '^' :
                case '>' :
                case '<' :
                case '\'':
                        strcat(where,attr0) ;
                        strcat(where," ") ;
                        break ;
                default  :
                        if ((strcmp(attr0,"AND") == 0) || (strcmp(attr0,"OR") == 0))
                        {
                                strcat(where,attr0) ;
                                strcat(where," ") ;
                                break ;
                        }
                        else
                        {
                                LexTabAttr(attr1,attr2,attr0) ;
                                if (strcmp(attr1,"") == 0)              /* attr */
                                {
                                        for (i=0;i<v_order;i++)
                                        {
                                                if ((strcmp(view[i].old_attr,attr2) == 0)
                                                && (strcmp(view[i].view_name,top_ptr->left->item) == 0))
                                                {
                                                        strcat(where,view[i].new_attr) ;
                                                        strcat(where," ") ;
                                                        break ;
                                                }
                                        }
                                }
                                else            /* tab.attr */
                                {
                                        for (i=0;i<v_order;i++)
                                        {
                                                if ((strcmp(view[i].old_attr,attr2) == 0)
                                                && (strcmp(view[i].view_name,top_ptr->left->item) == 0)
                                                && (strcmp(view[i].old_tab,attr1) == 0))
                                                {
                                                        strcat(where,view[i].new_attr) ;
                                                        strcat(where," ") ;
                                                        break ;
                                                }
                                        }
                                }
                        }
                        break ;         /* default */
        }
}while(sel_ptr != NULL) ;
strcpy(view_name,GetViewName()) ;
strcpy(stmt.arr,"CREATE VIEW ") ;
strcat(stmt.arr,view_name) ;
strcat(stmt.arr," AS SELECT * FROM ") ;
strcat(stmt.arr,top_ptr->left->item) ;
strcat(stmt.arr," WHERE ") ;
strcat(stmt.arr,where) ;
stmt.len = strlen(stmt.arr) ;
EXEC SQL EXECUTE IMMEDIATE :stmt ;
for (i=0;i<v_order;i++)
{
        if (strcmp(view[i].view_name,top_ptr->left->item) == 0)
                strcpy(view[i].view_name,view_name) ;
}
Free_node(top_ptr->left) ;
top_ptr->left = NULL ;
strcpy(top_ptr->type,LEAF) ;
strcpy(top_ptr->item,view_name) ;
```

```
        EXEC SQL COMMIT WORK ;
        return (0) ;
}
else if (strcmp(top_ptr->type,PROJECT) == 0)
{
        strcpy(where,"") ;
        k = 0 ;
        do
        {
                strcpy(attr1,"") ;
                strcpy(attr2,"") ;
                strcpy(top_ptr->item,LexTabAttr(attr1,attr2,top_ptr->item)) ; /* attr1 = tab, attr2 = attr */
                if (strcmp(attr1,"") == 0)              /* attr */
                {
                        for (i=0;i<v_order;i++)
                        {
                                if ((strcmp(view[i].old_attr,attr2) == 0)
                                && (strcmp(view[i].view_name,top_ptr->left->item) == 0))
                                {
                                        strcat(where,view[i].new_attr) ;
                                        strcpy(attr[k++],view[i].new_attr) ;
                                        strcat(where,",") ;
                                        break ;
                                }
                        }
                }
                else            /* tab.attr */
                {
                        for (i=0;i<v_order;i++)
                        {
                                if ((strcmp(view[i].old_attr,attr2) == 0)
                                && (strcmp(view[i].view_name,top_ptr->left->item) == 0)
                                && (strcmp(view[i].old_tab,attr1) == 0))
                                {
                                        strcat(where,view[i].new_attr) ;
                                        strcpy(attr[k++],view[i].new_attr) ;
                                        strcat(where,",") ;
                                        break ;
                                }
                        }
                }
        }while(strlen(top_ptr->item) != 0) ;
        n_attr = k ;
        i = strlen(where) ;
        where[i-1] = '\0' ; /* cut last , */
        strcpy(view_name,GetViewName()) ;
        strcpy(stmt.arr,"CREATE VIEW ") ;
        strcat(stmt.arr,view_name) ;
        strcat(stmt.arr," (") ;
        strcat(stmt.arr,where) ;
        strcat(stmt.arr,") AS SELECT DISTINCT ") ;
        strcat(stmt.arr,where) ;
        strcat(stmt.arr," FROM ") ;
        strcat(stmt.arr,top_ptr->left->item) ;
        stmt.len = strlen(stmt.arr) ;
        EXEC SQL EXECUTE IMMEDIATE :stmt ;
        for (i=0;i<v_order;i++)
        {
                if (strcmp(view[i].view_name,top_ptr->left->item) == 0)
                {
                        for (j=0;j<n_attr;j++)
                                if (strcmp(view[i].new_attr,attr[j]) == 0)
                                {
                                        strcpy(view[i].view_name,view_name) ;
                                        view[i].order = j+1 ;
                                        break ;
                                }
                }
```

```
            }
            Free_node(top_ptr->left) ;
            top_ptr->left = NULL ;
            strcpy(top_ptr->type,LEAF) ;
            strcpy(top_ptr->item,view_name) ;
            EXEC SQL COMMIT WORK ;
            return (0) ;
        }
        EXEC SQL WHENEVER SQLERROR CONTINUE ;
}
```

file_bk_sub1.pc

```
#include        "../tool/def.h"
#include        "../tool/proto.h"
#include        "backprot.h"
#include        "backxtrn.h"
EXEC SQL BEGIN DECLARE SECTION;
EXEC SQL INCLUDE BK_XTRN.H ;
EXEC SQL END DECLARE SECTION ;
EXEC SQL INCLUDE SQLCA ;
EXEC SQL INCLUDE SQLDA ;
SQLDA       *bdp ;          /* -> Descriptor used for BIND vars */
SQLDA       *sdp ;          /* -> Descuiptor used for SELECT vars */
short  *sdt = 0;            /* -> arr of original DESCRIBE's types */
int     sdtl;           /* nr of entries in sdt[] */
char   *vars = 0;           /* -> area used to hold Bind vars */
int     bdSize = 5;  /* Size of Bind variable descriptor */
int     bvSize = 10; /* Max nr of chars in Bind Var name */
int     sdSize = 5;  /* Size of Select list descriptor */
int     svSize = 80; /* Max nr chars in Select List colname */

extern char    *malloc();      /* Allocate Memory */
extern char    *sqlald();      /* Allocate Descriptor */

void SendData(table,sockfd)
char *table ;
int sockfd ;
{
        stmt1 = malloc(256) ;
        strcpy(stmt1,"SELECT * FROM ") ;
        strcat(stmt1,table) ;
        bdp = (SQLDA *)sqlald(bdSize, bvSize, 10) ;
        if (bdp == NULL)
        {
                puts("BACK_END: bdp = sqlald(bdSize, bvSize, 10)") ;
                exit (-1) ;
        }
        sdp = (SQLDA *)sqlald(sdSize, svSize, 0) ;              /* Allocate a new descriptor */
        if (sdp == NULL)
        {
                puts("BACK_END: sdp = sqlald(sdSize, svSize, 0)") ;
                exit(-1) ;
        }
        sdp->N = 0;         /* init in case exit before DESCRIBE */
        EXEC SQL WHENEVER SQLERROR GOTO CHECKERR ;
        EXEC SQL PREPARE S FROM :stmt1 ;
        if ( sqlca.sqlwarn[0] == 'W')
                mufwrn() ;
        EXEC SQL DECLARE C CURSOR FOR S ;
        bdp->N = bdSize ;
        descBind() ;
        if (bdp->F != 0)
                err_quit("BACK_END: SQL SYNTAX ERROR") ;
        EXEC SQL OPEN C USING DESCRIPTOR bdp ;
        sdp->N = sdSize ;
        descSel() ;
```

```
        if (sdp->F != 0)  fillSelDesc() ;
        if (sdp->N != 0)  doFetches(sockfd) ;
        EXEC SQL CLOSE C ;
        if (sdp->N != 0)  freeSelVars() ;
        if (vars != 0)
        {
                free(vars) ;
                vars = 0 ;
        }
        free (stmt1) ;
        return ;
CHECKERR:
        reptError() ;
        cleanUp() ;
        EXEC SQL WHENEVER SQLERROR CONTINUE ;
        exit (-1) ;
}

int LOexe_number(table)
char *table ;
{
        strcpy(stmt.arr,"SELECT COUNT (*) FROM ") ;
        strcat(stmt.arr,table) ;
        stmt.len = strlen(stmt.arr) ;
        EXEC SQL WHENEVER SQLERROR GOTO CHECKERR ;
        EXEC SQL PREPARE S1 FROM :stmt ;
        EXEC SQL DECLARE C1 CURSOR FOR S1 ;
        EXEC SQL OPEN C1 ;
        EXEC SQL FETCH C1 INTO :colno ;
        EXEC SQL CLOSE C1 ;
        return (colno) ;
CHECKERR:
        reptError() ;
        EXEC SQL WHENEVER SQLERROR CONTINUE ;
        exit (-1) ;
}

EXEC SQL WHENEVER SQLERROR STOP ;
EXEC SQL WHENEVER SQLWARNING CONTINUE ;
EXEC SQL WHENEER NOT FOUND CONTINUE ;

void mufwrn()
{
        if (sqlca.sqlwarn[1] == 'W')
                puts("SQLWARNING; Column was truncates.") ;
        else if ( sqlca.sqlwarn[2] == 'W')
                puts("SQLWARNING: Null values in aggregate(MAX, SUM) function.") ;
        else if ( sqlca.sqlwarn[3] == 'W')
                puts("SQLWARNING: INTO var count not equal column count.") ;
        else if ( sqlca.sqlwarn[4] == 'W')
                puts("SQLWARNING: Update or Delete without Where clause.") ;
        else if ( sqlca.sqlwarn[5] == 'W')
                puts("SQLWARNING: ???") ;
        else if ( sqlca.sqlwarn[6] == 'W')
                puts("SQLWARNING: Rollback required.") ;
        else if ( sqlca.sqlwarn[7] == 'W')
                puts("SQLWARNING: Change after query start on Select For Update.") ;
}

void cleanUp()
{
        if (sdp->N != 0) freeSelVars() ;
        if (vars != 0) { free(vars); vars = 0; }
        sqlclu(bdp) ;
        sqlclu(sdp) ;
}

void descBind()
```

```
{
        EXEC SQL DESCRIBE BIND VARIABLES FOR S INTO bdp ;
        if ( bdp->F < 0)        /* Descriptor not large enough */
        {
                bdSize = -(bdp->F) ;
                sqlclu(bdp) ;
                bdp = (SQLDA *)sqlald(bdSize, 30, 0) ;
                EXEC SQL DESCRIBE BIND VARIABLES FOR S INTO bdp ;
        }
        bdp->N = bdp->F ;
}

void descSel()
{
        EXEC SQL DESCRIBE SELECT LIST FOR S INTO sdp ;
        if (sdp->F < 0)
        {
                sdSize = -(sdp->F) ;
                sqlclu(sdp) ;
                sdp = (SQLDA *)sqlald(sdSize,30,0) ;
                EXEC SQL DESCRIBE SELECT LIST FOR S INTO sdp ;
        }
        sdp->N = sdp->F ;
}

void doFetches(sockfd)
int sockfd ;
{
        int     i, n ;
        short   *ip ;
        char    sendline[MAXLINE], recline[MAXLINE+1] ;
        char    colname[100] ;
        int     colnamel ;

    n = readline(sockfd, recline, MAXLINE) ;
        recline[n] = '\0' ;
        if ( n == 0)
        {
                printf("BackEnd: connection terminated") ;
                exit(-1) ;
        }
        else if (n < 0)
                err_dump("BackEnd: readline error") ;
        if (strncmp(recline, OK, 3) != 0)
                err_quit("BackEnd: protocal error") ;
        EXEC SQL WHENEVER NOT FOUND GOTO NOT_FND ;
        strcpy(sendline,DA) ;
        n = strlen(sendline) ;
        i = sdp->N ;
        memcpy(sendline+n,&i,sizeof(int)) ;
        n += sizeof(int) ;
        for (i=0; i<sdp->N; i++)
        {
                colnamel = sdp->L[i] ;
                memcpy(sendline+n,&colnamel,sizeof(colnamel)) ;
                n = n+sizeof(colnamel) ;
                colnamel = sdp->C[i] ;
                colnamel = min(colnamel,sizeof(colname) -1) ;
                memcpy(colname,sdp->S[i],sdp->C[i]);
                colname[sdp->C[i]] = '\0';
                if (sdp->L[i] > sdp->C[i])
                {
                        strPbnk(colname,colname,sdp->L[i]) ;
                        colnamel = sdp->L[i] ;
                }
                memcpy(sendline+n,&colnamel,sizeof(colnamel)) ;
                n = n+sizeof(colnamel) ;
                strcpy(sendline+n,colname) ;
```

```
            n = n+colnamel ;
        }
        if (writen(sockfd, sendline, n) != n)
            err_dump("BACK_END: writen error") ;
        n = readline(sockfd, recline, MAXLINE) ;
        recline[n] = '\0' ;
        if ( n == 0)
        {
            printf("BackEnd: connection terminated") ;
            exit(-1) ;
        }
        else if (n < 0)
            err_dump("BackEnd: readline error") ;
        if (strncmp(recline, OK, 3) != 0)
            err_quit("BackEnd: protocal error") ;
        for (;;)
        {
            strcpy(sendline,DA) ;
            n = strlen(sendline) ;
            EXEC SQL FETCH C USING DESCRIPTOR sdp ;
            for ( i=0;i < sdp->N; i++)
            {
                ip = sdp->I[i] ;
                if (*ip <  0)           /* return value is null */
                {
                    colnamel = 0 ;
                    memcpy(sendline+n,&colnamel,sizeof(colnamel)) ;
                    n += sizeof(colnamel) ;
                }
                else
                {
                    colnamel = strlen(sdp->V[i]) ;
                    memcpy(sendline+n,&colnamel,sizeof(colnamel)) ;
                    n += sizeof(colnamel) ;
                    memcpy(sendline+n,sdp->V[i],colnamel) ;
                    n += colnamel ;
                }
            }
            if (writen(sockfd, sendline, n) != n)
                err_dump("BACK_END: writen error") ;
            n = readline(sockfd, recline, MAXLINE) ;
            recline[n] = '\0' ;
            if ( n == 0)
            {
                printf("BackEnd: connection terminated") ;
                exit(-1) ;
            }
            else if (n < 0)
                err_dump("BackEnd: readline error") ;
            if (strncmp(recline, OK, 3) != 0)
                err_quit("BackEnd: protocal error") ;
        }
NOT_FND:
        EXEC SQL WHENEVER NOT FOUND CONTINUE ;
        return ;
}

void fillSelDesc()
{
        int     i;
        unsigned char       prec;
        char    scale;

        if (!sdt)
            sdt = (short*)malloc(sizeof(short) * sdp->N);
        else if (sdtl < sdp->N)
            sdt = (short *)realloc(sdt,sizeof(short) * sdp->N);
        sdtl = sdp->N;
```

```
        for ( i=0; i<sdp->N;i++)
        {
                sdp->T[i] = ( sdp->T[i] & ~0x8000);
                sdt[i] = sdp->T[i];
                if (sdp->T[i] == 2)
                {
                        prec = (unsigned char)(sdp->L[i] >>8);
                        scale = (char)sdp->L[i];
                        if (prec == 0)
                                prec = 26;
                        sdp->L[i] = prec;
                        if (scale < 0)
                                sdp->L[i] += -scale;
                        sdp->L[i] += 2 ;
                }
                else if (sdp->T[i] == 12)
                        sdp->L[i] = 9 ;
                sdp->T[i] = 5 ;
                sdp->L[i] = min(sdp->L[i],240);
                sdp->V[i] = malloc(sdp->L[i]);
                sdp->I[i] = (short *)malloc(sizeof(short));
        }
}

void freeSelVars()
{
        int i;
        for (i=0; i<sdp->N;i++)
        {
                free(sdp->V[i]);
                free(sdp->I[i]);
        }
        sdp->N =0;
}

void reptError()
{
        printf("%.70s\n", sqlca.sqlerrm.sqlerrmc) ;
}

int min(c1,c2)
int c1,c2;
{
        if (c1 > c2) return (c2) ;
        else return (c1) ;
}

int max(c1,c2)
int c1,c2 ;
{
        if (c1 > c2) return (c1) ;
        else return (c2) ;
}

void SendView(viewname,sockfd)
char *viewname ;
int sockfd ;
{
        char    sendline[MAXLINE], recline[MAXLINE+1] ;
        char    buff[5] ;
        int     i, n ;

        for (i=0;i<v_order;i++)
        {
                if (strcmp(view[i].view_name,viewname) == 0)
                {
                        strcpy(sendline,VIEW) ;
                        strcat(sendline,view[i].new_attr) ;
```

```
                strcat(sendline," ") ;
                strcat(sendline,view[i].old_tab) ;
                strcat(sendline," ") ;
                strcat(sendline,view[i].old_attr) ;
                strcat(sendline," ") ;
                itoa(view[i].order,buff) ;
                strcat(sendline,buff) ;
                n = strlen(sendline) ;
                if (writen(sockfd, sendline, n) != n)
                        err_dump("BACK_END: writen error") ;
                n = readline(sockfd, recline, MAXLINE) ;
                recline[n] = '\0';
                if ( n == 0)
                {
                        printf("BackEnd: connection terminated") ;
                        exit(-1) ;
                }
                else if (n < 0)
                        err_dump("BackEnd: readline error") ;
                if (strncmp(recline, OK, 3) != 0)
                        err_quit("BackEnd: protocal error") ;
        }
    }
    strcpy(sendline,EN) ;
    n = strlen(sendline) ;
    if (writen(sockfd, sendline, n) != n)
        err_dump("BACK_END: writen error") ;
    n = readline(sockfd, recline, MAXLINE) ;
    recline[n] = '\0';
    if ( n == 0)
    {
        printf("BackEnd: connection terminated") ;
        exit(-1) ;
    }
    else if (n < 0)
        err_dump("BackEnd: readline error") ;
    if (strncmp(recline, OK, 3) != 0)
        err_quit("BackEnd: protocal error") ;
}

int MakeView(viewname,sockfd)
char *viewname ;
int sockfd ;
{
    int     l, m, n, i ;
    char    sendline[MAXLINE], recline[MAXLINE+1] ;
    char    table[30], number[10], *ch_ptr ;

    strcpy(sendline,OK) ;
    n = strlen(sendline) ;
    if (writen(sockfd,sendline,n) != n)
        err_dump("BACK_END: writen error") ;
    n = readline(sockfd, recline, MAXLINE) ;
    recline[n] = '\0';
    if ( n == 0)
    {
        printf("BackEnd: connection terminated") ;
        exit(-1) ;
    }
    else if (n < 0)
        err_dump("BackEnd: readline error") ;
    if (strncmp(recline, DA, 3) != 0)
        err_quit("BackEnd: protocal error") ;
    n = 3 ;
    memcpy(&n_new_attr,recline+n,sizeof(int)) ;
    n += sizeof(int) ;
    for (i=0;i<n_new_attr;i++)
    {
```

```
        n += sizeof(int) ;
        memcpy(&new_com[i],recline+n,sizeof(int)) ;
        n += sizeof(int) ;
        strncpy(new_attr[i],recline+n,new_com[i]) ;
        new_attr[i][new_com[i]] = '\0' ;
        Lex__Attr(old_attr[i],new_attr[i]) ;
        n += new_com[i] ;
}
n_old_attr = n_new_attr ;
strcpy(table,GetTabName()) ;
strcpy(stmt.arr,"CREATE TABLE ") ;
strcat(stmt.arr,table) ;
strcat(stmt.arr," (") ;
for (i=0;i<n_old_attr;i++)
{
        strcpy(attr_name.arr,old_attr[i]) ;
        attr_name.len = strlen(attr_name.arr) ;
        EXEC SQL  SELECT TYPE, WIDTH INTO :type, :width
        FROM G$A$ WHERE ATTR_NAME = :attr_name ;
        LexWord(type,type) ;
        strcat(stmt.arr,old_attr[i]) ;
        strcat(stmt.arr," ") ;
        strcat(stmt.arr,type) ;
        if (strcmp(type,"CHAR") == 0)
        {
                itoa(width,number) ;
                strcat(stmt.arr,"(") ;
                strcat(stmt.arr,number) ;
                strcat(stmt.arr,")") ;
        }
        if (strcmp(type,"NUMBER") == 0)
                old_com[i] = 1 ;
        else
                old_com[i] = 0 ;
        if (i<n_old_attr-1)
                strcat(stmt.arr,",") ;
}
strcat(stmt.arr,")") ;
stmt.len = strlen(stmt.arr) ;
EXEC SQL EXECUTE IMMEDIATE :stmt ;
EXEC SQL COMMIT WORK ;
while(1)
{
        strcpy(sendline,OK) ;
        n = strlen(sendline) ;
        if (writen(sockfd,sendline,n) != n)
                err_dump("BACK_END: writen error") ;
        n = readline(sockfd, recline, MAXLINE) ;
        recline[n] = '\0';
        if ( n == 0)
        {
                printf("BackEnd: connection terminated") ;
                exit(-1) ;
        }
        else if (n < 0)
                err_dump("BackEnd: readline error") ;
        if (strncmp(recline,DA,3) != 0)
                break ;
        n = 3 ;
        strcpy(stmt.arr,"INSERT INTO ") ;
        strcat(stmt.arr,table) ;
        strcat(stmt.arr," VALUES (") ;
        m = strlen(stmt.arr) ;
        for (i=0;i<n_old_attr;i++)
        {
                memcpy(&l,recline+n,sizeof(int)) ;
                n += sizeof(int) ;
                if (old_com[i] == 0)
```

```
                {
                        strcat(stmt.arr,"") ;
                        m++ ;
                }
                strncat(stmt.arr,recline+n,l) ;
                m += l ;
                n += l ;
                stmt.arr[m] = '\0' ;
                if (old_com[i] == 0)
                {
                        strcat(stmt.arr,"") ;
                        m++ ;
                }
                if (i<n_old_attr-1)
                {
                        strcat(stmt.arr,",") ;
                        m++ ;
                }
        }
        strcat(stmt.arr,")") ;
        stmt.len = strlen(stmt.arr) ;
        EXEC SQL EXECUTE IMMEDIATE :stmt ;
}
EXEC SQL COMMIT WORK ;
strcpy(stmt.arr,"CREATE VIEW ") ;
strcat(stmt.arr,viewname) ;
strcat(stmt.arr," (") ;
for (i=0;i<n_new_attr;i++)
{
        strcat(stmt.arr,new_attr[i]) ;
        if (i<n_new_attr-1)
                strcat(stmt.arr,",") ;
}
strcat(stmt.arr,") AS SELECT * FROM ") ;
strcat(stmt.arr,table) ;
stmt.len = strlen(stmt.arr) ;
EXEC SQL EXECUTE IMMEDIATE :stmt ;
EXEC SQL COMMIT WORK ;
/* receive view describe */
strcpy(sendline,OK) ;
m = strlen(sendline) ;
while(1)
{
        if (strncmp(recline,VIEW,3) != 0)
                break ;
        ch_ptr = recline+3 ;
        ch_ptr = LexWord(view[v_order].new_attr,ch_ptr) ;
        ch_ptr = LexWord(view[v_order].old_tab,ch_ptr) ;
        ch_ptr = LexWord(view[v_order].old_attr,ch_ptr) ;
        view[v_order].order = atoi(ch_ptr) ;
        strcpy(view[v_order].view_name,viewname) ;
        v_order++ ;
        if (writen(sockfd,sendline,m) != m)
                err_dump("BACK_END: writen error") ;
        n = readline(sockfd,recline,MAXLINE) ;
        recline[n] = '\0' ;
        if ( n == 0)
        {
                printf("BackEnd: connection terminated") ;
                exit(-1) ;
        }
        else if (n < 0)
                err_dump("BackEnd: readline error") ;
}
if (strncmp(recline,EN,3) != 0)
{
        printf("BACK_END: protocal error") ;
        exit (-1) ;
```

```
        }
        else
        if (writen(sockfd,sendline,m) != m)
                err_dump("BACK_END: writen error") ;
}

/* return new table name , used to create table */
char *GetTabName()
{
        char    number[10] ;

        strcpy(ttemp_name[tt_order],temp_table) ;
        itoa(tt_order,number) ;
        strcat(ttemp_name[tt_order],number) ;
        tt_order++ ;
        return (ttemp_name[tt_order-1]) ;
}

/* drop all temp tables and drop all view. */
void clean_tab()
{
        int     i ;

        strcpy(exe_imm,"DROP VIEW ") ;
        EXEC SQL WHENEVER SQLERROR GOTO error ;
        for (i=0;i<vt_order;i++)
        {
                strcpy(stmt.arr,exe_imm) ;
                strcat(stmt.arr,vtemp_name[i]) ;
                stmt.len = strlen(stmt.arr) ;
                EXEC SQL EXECUTE IMMEDIATE :stmt ;
        }
        EXEC SQL COMMIT WORK RELEASE ;
        EXEC SQL CONNECT :uid IDENTIFIED BY :pwd ;
        strcpy(exe_imm,"DROP TABLE ") ;
        for (i=0;i<tt_order;i++)
        {
                strcpy(stmt.arr,exe_imm) ;
                strcat(stmt.arr,ttemp_name[i]) ;
                stmt.len = strlen(stmt.arr) ;
                EXEC SQL EXECUTE IMMEDIATE :stmt ;
        }
        tt_order = 0 ;
        vt_order = 0 ;
        v_order = 0 ;
        EXEC SQL WHENEVER SQLERROR CONTINUE ;
        return ;
error:
        reptError() ;
        err_quit("BackEnd: clean_tab() error ") ;
}

TREE_PTR adjust(top_ptr)
TREE_PTR top_ptr ;
{
        int     no_exist ;

        no_exist = chkd_tab(top_ptr) ;
        if (no_exist)
        {
                Free_tree(top_ptr) ;
                return(NULL) ;
        }
        else
        {
                top_ptr = arrange(top_ptr) ;
                return (top_ptr) ;
        }
```

```
}

int ContProc(sockfd,top_ptr,local)
int sockfd ;
TREE_PTR top_ptr ;
char local ;
{
        char    sendline[MAXLINE], recline[MAXLINE+1] ;
        int     n ;

        strcpy(sendline,CM) ;
        strcat(sendline,top_ptr->type) ;
        if (local == 'L')
                strcat(sendline,"L") ;
        else
                strcat(sendline,"R") ;
        n = strlen(sendline) ;
        if (writen(sockfd,sendline,n) != n)
                err_dump("BACK_END: writen error") ;
        n = readline(sockfd,recline,MAXLINE) ;
        recline[n] = '\0' ;
        if ( n == 0)
        {
                printf("BackEnd: connection terminated") ;
                exit(-1) ;
        }
        else if (n < 0)
                err_dump("BackEnd: readline error") ;
        if (strncmp(recline, OK, 3) != 0)
                err_quit("BackEnd: protocal error") ;
        if (local == 'L')
        {
                SendData(top_ptr->left->item,sockfd) ;
                SendView(top_ptr->left->item,sockfd) ;
        }
        else
        {
                SendData(top_ptr->right->item,sockfd) ;
                SendView(top_ptr->right->item,sockfd) ;
        }
        strcpy(top_ptr->item,GetViewName()) ;
        MakeView(top_ptr->item,sockfd) ;
        Free_node(top_ptr->left) ;
        top_ptr->left = NULL ;
        Free_node(top_ptr->right) ;
        top_ptr->right = NULL ;
        strcpy(top_ptr->type,LEAF) ;
        return (LOexe_number(top_ptr->item)) ;
}

int PassToRM( fd1, fd2)  /* fd1 = come in from another, fd2 go out to RM */
int fd1, fd2;
{
        int     maxfdp1, nfound, nread ;
        char    buff[MAXLINE] ;
        fd_set readmask ;

        FD_ZERO(&readmask) ;
        for (;;)
        {
                FD_SET(fd1, &readmask) ;
                FD_SET(fd2, &readmask) ;
                maxfdp1 = (fd1 > fd2)? fd1+1 : fd2+1 ;
                nfound = select(maxfdp1, &readmask, (fd_set *) 0, (fd_set *) 0, (struct timeval *) 0) ;
                if (nfound < 0)
                        err_sys("pass_data: select error") ;
                if (FD_ISSET(fd1, &readmask))                   /* data come from fd1 */
                {
```

```
                nread = readline(fd1, buff, MAXLINE) ;
                if (nread < 0)
                        err_sys("pass_data: read error from fd1") ;
                else if (nread == 0)
                        break ;                         /* EOF */
                if (writen(fd2, buff, nread) != nread)
                        err_sys("pass_data: writen error to fd2") ;
        }
        if (FD_ISSET(fd2, &readmask))                   /* data come from fd2 */
        {
                nread = readline(fd2, buff, MAXLINE) ;
                if (nread < 0)
                        err_sys("pass_data: read error from fd2") ;
                else if (nread == 0)
                        break ;                         /* EOF */
                if (writen(fd1, buff, nread) != nread)
                        err_sys("pass_data: writen error to fd1") ;
                if (strncmp(buff,EN,3) == 0)
                        break ;
        }
    }
    nread = readline(fd1, buff, MAXLINE) ;
    if (nread < 0)
            err_sys("pass_data: read error from fd1") ;
    else if (nread == 0)
            return (-1) ;           /* EOF */
    if (writen(fd2, buff, nread) != nread)
            err_sys("pass_data: writen error to fd2") ;
    return (0) ;
}

/* return 0 : continue execute at left site */
/* return 1 : continue execute at right site*/
/* return -1: can't decision */
/* L = left, R = right */
/* LocalSite = 0 when L is local */
/* LocalSite = 1 wher R is local */
int CompLR(LNumber,LTable,oper,RNumber,RTable,LocalSite)
int LNumber, RNumber, LocalSite ;
char *oper, *LTable, *RTable ;
{
    int     number ;

    if (strcmp(oper,UNION) == 0)
    {
            return (LocalSite) ;
    }
    else if (strcmp(oper,MINUS) == 0)
    {
            if (LocalSite == 0)
            {
                    number = 1.5 * LNumber ;
                    if (RNumber > number)
                            return (1) ;
                    else
                            return (0) ;
            }
            else
            {
                    return (1) ;
            }
    }
    else if (strcmp(oper,INTERSECT) == 0)
    {
            if (LocalSite == 0)
            {
                    number = 1.5 * LNumber ;
                    if (RNumber > number)
```

```
                        return (1) ;
                else
                        return (0) ;
        }
        else
        {
                number = 1.5 * RNumber ;
                if (LNumber > number)
                        return (0) ;
                else
                        return (1) ;
        }
    }
    else if (strcmp(oper,TIMES) == 0)
    {
        return (LocalSite) ;
    }
    else if (strcmp(oper,JOIN) == 0)
    {
        return (LocalSite) ;
    }
    else if (strcmp(oper,DIVIDEBY) == 0)
    {
        return (0) ;
    }
    else
        return(LocalSite) ;
}

/* return 0 : all table are exist */
/* return 1 : any table are not exist */
int chkd_tab(top_ptr)
TREE_PTR top_ptr ;
{
    int     no_flg ;

    if ((top_ptr->left == NULL) && (top_ptr->right == NULL))
    {
        strcpy(table_name.arr,top_ptr->item) ;
        table_name.len = strlen(table_name.arr) ;
        EXEC SQL WHENEVER SQLERROR GOTO error ;
        EXEC SQL SELECT HOST_NAME INTO :host_name
        FROM G$THU$
        WHERE TABLE_NAME = :table_name ;
        if (sqlca.sqlcode != 0)
        {
            strcpy(err_mesg,"TABLE '") ;
            strcat(err_mesg,table_name.arr) ;
            strcat(err_mesg,"' DOES NOT EXIST") ;
            err_flg = 1 ;
            return (1) ;
        }
        else return (0) ;
    }
    if (top_ptr->left != NULL)
    {
        no_flg = chkd_tab(top_ptr->left) ;
        if (no_flg)
            return (1) ;
    }
    if (top_ptr->right != NULL)
    {
        no_flg = chkd_tab(top_ptr->right) ;
        if (no_flg)
            return (1) ;
    }
    return (0) ;
error:
```

```
        reptError() ;
        err_quit("BackEnd: clean_tab() error ") ;
}

TREE_PTR arrange(top_ptr)
TREE_PTR top_ptr ;
{
        ADJ   *arr_ptr ;

        arr_ptr = TreeToAdj(top_ptr) ;
        Free_tree (top_ptr) ;
        /* arrange tree */
        if (!Constraint(arr_ptr))
        {
                err_flg = 1 ;
                strcpy(err_mesg,"selected 0 row") ;
                return (NULL) ;
        }
        FillAttr(arr_ptr) ;
        arr_ptr = RuleChk(arr_ptr) ;
        /* ------ */
        top_ptr = AdjToTree(arr_ptr) ;
        Free_Adj(arr_ptr) ;
        return (top_ptr) ;
}

ADJ *TreeToAdj(top_ptr)
TREE_PTR top_ptr ;
{
        int     item_flg ;
        ADJ   *top ;
        NAME *item_ptr ;
        char   *ch_ptr, buff[30] ;

        item_flg = 0 ;        /* 1 when have alloc item */
        top = (ADJ *)malloc(sizeof(ADJ)) ;
        top->parent = NULL ;
        top->item = NULL ;
        ch_ptr = top_ptr->item ;
        do
        {
                ch_ptr = LexAdj(buff,ch_ptr) ;
                if (strcmp(buff,"") != 0)
                {
                        if (!item_flg)  /* first time */
                        {
                                item_ptr = (NAME *)malloc(sizeof(NAME)) ;
                                top->item = item_ptr ;
                                item_ptr->prev = NULL ;
                                item_ptr->next = NULL ;
                                item_flg = 1 ;
                        }
                        else
                        {
                                item_ptr->next = (NAME *)malloc(sizeof(NAME)) ;
                                item_ptr->next->prev = item_ptr ;
                                item_ptr = item_ptr->next ;
                                item_ptr->next = NULL ;
                        }
                        strcpy(item_ptr->string,buff) ;
                }
        }while (ch_ptr != NULL) ;
        strcpy(top->type,top_ptr->type) ;
        top->attr_list = NULL ;
        if (top_ptr->left != NULL)
        {
                top->left = TreeToAdj(top_ptr->left) ;
                top->left->parent = top ;
```

```
        }
        else
                top->left = NULL ;
        if (top_ptr->right != NULL)
        {
                top->right = TreeToAdj(top_ptr->right) ;
                top->right->parent = top ;
        }
        else
                top->right = NULL ;
        return (top) ;
}

TREE_PTR AdjToTree(adj_ptr)
ADJ *adj_ptr ;
{
        TREE_PTR    top_ptr ;
        NAME *item_ptr ;
        char   item[AGB_LEN] ;

        if (adj_ptr->item != NULL)
        {
                if (strcmp(adj_ptr->type,SELECT) == 0)
                {
                        item_ptr = adj_ptr->item ;
                        strcpy(item,item_ptr->string) ;
                        item_ptr = item_ptr->next ;
                        while(item_ptr != NULL)
                        {
                                strcat(item," ") ;
                                strcat(item,item_ptr->string) ;
                                item_ptr = item_ptr->next ;
                        }
                        top_ptr = Make_node(adj_ptr->type, item) ;
                }
                else if (strcmp(adj_ptr->type,PROJECT) == 0)
                {
                        item_ptr = adj_ptr->item ;
                        strcpy(item,item_ptr->string) ;
                        item_ptr = item_ptr->next ;
                        while(item_ptr != NULL)
                        {
                                strcat(item,",") ;
                                strcat(item,item_ptr->string) ;
                                item_ptr = item_ptr->next ;
                        }
                        top_ptr = Make_node(adj_ptr->type, item) ;
                }
                else
                {
                        strcpy(item,adj_ptr->item->string) ;
                        top_ptr = Make_node(adj_ptr->type, item) ;
                }
        }
        else
                top_ptr = Make_node(adj_ptr->type, NULL) ;
        if (adj_ptr->left != NULL)
                top_ptr->left = AdjToTree(adj_ptr->left) ;
        else
                top_ptr->left = NULL ;
        if (adj_ptr->right != NULL)
                top_ptr->right = AdjToTree(adj_ptr->right) ;
        else
                top_ptr->right = NULL ;
        return (top_ptr) ;
}

int Constraint(adj_ptr)
```

```
ADJ *adj_ptr ;
{
        int     Lflg, Rflg ;

        if (adj_ptr->left != NULL)
                Lflg = Constraint(adj_ptr->left) ;
        if (adj_ptr->right != NULL)
                Rflg = Constraint(adj_ptr->right) ;
        if (strcmp(adj_ptr->type,SELECT) == 0)
        {
                if (Lflg)
                {
                        POS = adj_ptr->item ;
                        if (!ValidCons(adj_ptr->item))
                        return (FALSE) ;
                }
                else
                        return (FALSE) ;
        }
        else if (strcmp(adj_ptr->type,UNION) == 0)
        {
                if (Lflg || Rflg)
                        return (TRUE) ;
                else
                        return (FALSE) ;
        }
        else if (strcmp(adj_ptr->type,MINUS) == 0)
        {
                if (Lflg)
                        return (TRUE) ;
                else
                        return (FALSE) ;
        }
        else if (strcmp(adj_ptr->type,INTERSECT) == 0)
        {
                if (Lflg && Rflg)
                        return (TRUE) ;
                else
                        return (FALSE) ;
        }
        else if (strcmp(adj_ptr->type,TIMES) == 0)
        {
                if (Lflg && Rflg)
                        return (TRUE) ;
                else
                        return (FALSE) ;
        }
        else if (strcmp(adj_ptr->type,JOIN) == 0)
        {
                if (Lflg && Rflg)
                        return (TRUE) ;
                else
                        return (FALSE) ;
        }
        else if (strcmp(adj_ptr->type,PROJECT) == 0)
        {
                if (Lflg)
                        return (TRUE) ;
                else
                        return (FALSE) ;
        }
        else if (strcmp(adj_ptr->type,DIVIDEBY) == 0)
        {
                if (Lflg && Rflg)
                        return (TRUE) ;
                else
                        return (FALSE) ;
        }
```

```
        return (TRUE) ;
}

int ValidCons(item)
NAME *item ;
{
        if (!ValidUnit(item))
                return (FALSE) ;
        else
        {
                POS = POS->next ;
                if ((strcmp(POS->string,"AND") == 0) || (strcmp(POS->string,"OR") == 0))
                {
                        POS = POS->next ;
                        if (!ValidUnit(POS))
                                return (FALSE) ;
                        else
                                return (TRUE) ;
                }
                else
                {
                        POS = POS->prev ;
                        return (TRUE) ;
                }
        }
}

int ValidUnit(item)
NAME *item ;
{
        int     oper_flg ;

        if (strcmp(item->string,"(") != 0)
        {
                strcpy(attr_name.arr,item->string) ;
                attr_name.len = strlen(attr_name.arr) ;
                /* chkd valid attribute name */
                EXEC SQL WHENEVER SQLERROR GOTO error ;
                EXEC SQL SELECT TYPE INTO :type FROM G$A$
                WHERE ATTR_NAME = :attr_name ;
                if (sqlca.sqlcode != 0)
                        return (FALSE) ;
                POS = POS->next ;
                oper_flg = CompOpChk(POS->string) ;
                if (oper_flg == MAXCOMP)
                        return (FALSE) ;
                POS = POS->next ;
                if (!Numberic(POS->string))
                {
                        if (strncmp(POS->string,"",1) == 0)
                                return (TRUE) ;
                        else
                        {
                                strcpy(attr_name.arr,POS->string) ;
                                attr_name.len = strlen(attr_name.arr) ;
                                /* chkd valid attribute name */
                                EXEC SQL SELECT TYPE INTO :type FROM G$A$
                                        WHERE ATTR_NAME = :attr_name ;
                                if (sqlca.sqlcode != 0)
                                        return (FALSE) ;
                                else
                                        return (TRUE) ;
                        }
                }
                else
                {
                        if (!(*UnitComp[oper_flg])())
                                return (FALSE) ;
```

```
                else
                        return (TRUE) ;
        }
    }
    else
    {
        POS = POS->next ;
        if (!ValidCons(POS))
            return (FALSE) ;
        else
        {
            POS = POS->next ;
            if (strcmp(POS->string,")") != 0)
                return (FALSE) ;
            else
                return (TRUE) ;
        }
    }
error:
    reptError() ;
    err_quit("BackEnd: clean_tab() error\n") ;
}

int CompOpChk(str)
char *str ;
{
    int     order ;
    static char    *comparator[MAXCOMP] = {"=","^=","<",">","<=",">="};

    for (order=0;order<MAXCOMP;order++)
    {
        if (strcmp(str,comparator[order]) == 0)
            break ;
    }
    return (order);
}

int Numberic(str)
char *str ;
{
    if ((*str >= '0') && (*str <= '9'))
        return (TRUE) ;
    else
        return (FALSE) ;
}
```

file bk_sub2.pc

```
#include        "../tool/def.h"
#include        "../tool/proto.h"
#include        "backprot.h"
#include        "backxtrn.h"
EXEC SQL BEGIN DECLARE SECTION;
EXEC SQL INCLUDE BK_XTRN.H ;
EXEC SQL END DECLARE SECTION ;
EXEC SQL INCLUDE SQLCA ;

/* This comment used in 6 function follow */
/* V is white dot in real-numberic line */
/* A is black dot in real-numberic line */
/* - is interval  in real_numberic line */

int EquCons()
{
    int     false_flg ;
    float   number ;
```

```
false_flg = 0 ;
number = atof(POS->string) ;
strcpy(attr_name.arr,POS->prev->prev->string) ;
attr_name.len = strlen(attr_name.arr) ;
EXEC SQL WHENEVER SQLERROR GOTO error ;
EXEC SQL DECLARE C2 CURSOR FOR
        SELECT MORE_THAN, MORE_EQU, LESS_THAN, LESS_EQU FROM G$C$
                WHERE ATTR_NAME = :attr_name ;
EXEC SQL OPEN C2 ;
while(1)
{
        EXEC SQL FETCH C2 INTO :more_than :more_thani, :more_equ :more_equi,
        :less_than :less_thani, :less_equ :less_equi ;
        if (sqlca.sqlcode != 0)
                break ;
        if (more_thani == 0)
        {
                if (less_thani == 0)
                {
                        if (more_than == less_than)             /* -V- */
                        {
                                if (more_than != number)  /* valid */
                                        continue ;
                                else
                                {
                                        false_flg = 1 ;
                                        break ;
                                }
                        }
                        else if (more_than < less_than)         /* V-V */
                        {
                                if ((more_than < number) && (less_than > number))
                                        continue ;
                                else
                                {
                                        false_flg = 1 ;
                                        break ;
                                }
                        }
                        else   /* -V V- */
                        {
                                if ((more_than < number) || (less_than > number))
                                        continue ;
                                else
                                {
                                        false_flg = 1 ;
                                        break ;
                                }
                        }
                }
                else if (less_equi == 0)
                {
                        if (more_than < less_equ)          /* V-A */
                        {
                                if ((more_than < number) && (less_equ >= number))
                                        continue ;
                                else
                                {
                                        false_flg = 1 ;
                                        break ;
                                }
                        }
                        else   /* -A V- */
                        {
                                if ((more_than < number) || (less_equ >= number))
                                        continue ;
                                else
                                {
```

```
                        false_flg = 1 ;
                        break ;
                    }
                }
            }
            else        /* V- */
            {
                if (more_than < number )
                    continue ;
                else
                {
                    false_flg = 1 ;
                    break ;
                }
            }
        }
        else if (more_equi == 0)
        {
            if (less_thani == 0)
            {
                if (more_equ < less_than)       /* A-V */
                {
                    if ((more_equ <= number) && (less_than > number))
                        continue ;
                    else
                    {
                        false_flg = 1 ;
                        break ;
                    }
                }
                else    /* -V A- */
                {
                    if ((more_equ <= number) || (less_than > number))
                        continue ;
                    else
                    {
                        false_flg = 1 ;
                        break ;
                    }
                }
            }
            else if (less_equi == 0)
            {
                if (more_equ == less_equ)       /* A */
                {
                    if (more_equ == number)
                        continue ;
                    else
                    {
                        false_flg = 1 ;
                        break ;
                    }
                }
                else if (more_equ < less_equ)           /* A-A */
                {
                    if ((more_equ <= number) && (less_equ >= number))
                        continue ;
                    else
                    {
                        false_flg = 1 ;
                        break ;
                    }
                }
                else                    /* -A A- */
                {
                    if ((more_equ <= number) || (less_equ >= number))
                        continue ;
                    else
```

```
                        {
                                false_flg = 1 ;
                                break ;
                        }
                }
        }
        else            /* A- */
        {
                if (more_equ <= number)
                        continue ;
                else
                {
                        false_flg = 1 ;
                        break ;
                }
        }
    }
    else if (less_thani == 0)     /* -V */
    {
        if (less_than > number)
                continue ;
        else
        {
                false_flg = 1 ;
                break ;
        }
    }
    else if (less_equi == 0)      /* -A */
    {
        if (less_equ >= number)
                continue ;
        else
        {
                false_flg = 1 ;
                break ;
        }
    }
  }
  EXEC SQL CLOSE C2 ;
  if (false_flg)
        return (FALSE) ;
  else
        return (TRUE) ;
error:
  reptError() ;
  err_quit("BackEnd: EquCons() error\n") ;
}

int NotEquCons()
{
  int   false_flg ;
  float number ;

  false_flg = 0 ;
  number = atof(POS->string) ;
  strcpy(attr_name.arr,POS->prev->prev->string) ;
  attr_name.len = strlen(attr_name.arr) ;
  EXEC SQL WHENEVER SQLERROR GOTO error ;
  EXEC SQL DECLARE C3 CURSOR FOR
        SELECT MORE_THAN, MORE_EQU, LESS_THAN, LESS_EQU FROM G$C$
            WHERE ATTR_NAME = :attr_name ;
  EXEC SQL OPEN C3 ;
  while(1)
  {
        EXEC SQL FETCH C3 INTO :more_than :more_thani, :more_equ :more_equi,
        :less_than :less_thani, :less_equ :less_equi ;
        if (sqlca.sqlcode != 0)
                break ;
```

```
            if ((more_equi == 0) && (less_equi == 0))          /* A */
            {
                if ((more_equ == less_equ) && (more_equ == number))
                {
                    false_flg = 1 ;
                    break ;
                }
            }
        }
        EXEC SQL CLOSE C3 ;
        if (false_flg)
            return (FALSE) ;
        else
            return (TRUE) ;
error:
        reptError() ;
        err_quit("BackEnd: NotEquCons() error\n") ;
}

int LessCons()
{
        int     false_flg ;
        float   number ;

        false_flg = 0 ;
        number = atof(POS->string) ;
        strcpy(attr_name.arr,POS->prev->prev->string) ;
        attr_name.len = strlen(attr_name.arr) ;
        EXEC SQL WHENEVER SQLERROR GOTO error ;
        EXEC SQL DECLARE C4 CURSOR FOR
            SELECT MORE_THAN, MORE_EQU, LESS_THAN, LESS_EQU FROM G$C$
                WHERE ATTR_NAME = :attr_name ;
        EXEC SQL OPEN C4 ;
        while(1)
        {
            EXEC SQL FETCH C4 INTO :more_than :more_thani, :more_equ :more_equi,
            :less_than :less_thani, :less_equ :less_equi ;
            if (sqlca.sqlcode != 0)
                break ;
            if (more_thani == 0)
            {
                if (less_thani == 0)
                {
                    if (more_than == less_than)          /* -V- */
                        continue ;
                    else if (more_than < less_than)      /* V-V */
                    {
                        if (more_than >= number)
                        {
                            false_flg = 1 ;
                            break ;
                        }
                        else
                            continue ;
                    }
                    else                                 /* -V V- */
                        continue ;
                }
                else if (less_equi == 0)
                {
                    if (more_than < less_equ)            /* V-A */
                    {
                        if (more_than >= number)
                        {
                            false_flg = 1 ;
                            break ;
                        }
                        else
```

```
                        continue ;
                }
                else                    /* -A V- */
                        continue ;
        }
        else                            /* V- */
        {
                if (more_than >= number)
                {
                        false_flg = 1 ;
                        break ;
                }
                else
                        continue ;
        }
}
else if (more_equi == 0)
{
        if (less_thani == 0)
        {
                if (more_equ < less_than)       /* A-V */
                {
                        if (more_equ >= number)
                        {
                                false_flg = 1 ;
                                break ;
                        }
                        else
                                continue ;
                }
                else                    /* -V A- */
                        continue ;
        }
        else if (less_equi == 0)
        {
                if (more_equ == less_equ)       /* A */
                {
                        if (more_equ >= number)
                        {
                                false_flg = 1 ;
                                break ;
                        }
                        else
                                continue ;
                }
                else if (more_equ < less_equ)           /* A-A */
                {
                        if (more_equ >= number)
                        {
                                false_flg = 1 ;
                                break ;
                        }
                        else
                                continue ;
                }
                else                    /* -A A- */
                        continue ;
        }
        else                            /* A- */
        {
                if (more_equ >= number)
                {
                        false_flg = 1 ;
                        break ;
                }
                else
                        continue ;
        }
```

```
            }
            else if (less_thani == 0)                    /* -V */
                continue ;
            else if (less_equi == 0)                     /* -A */
                continue ;
        }
        EXEC SQL CLOSE C4 ;
        if (false_flg)
            return (FALSE) ;
        else
            return (TRUE) ;
error:
        reptError() ;
        err_quit("BackEnd: LessCons() error\n") ;
}

int MoreCons()
{
        int     false_flg ;
        float   number ;

        false_flg = 0 ;
        number = atof(POS->string) ;
        strcpy(attr_name.arr,POS->prev->prev->string) ;
        attr_name.len = strlen(attr_name.arr) ;
        EXEC SQL WHENEVER SQLERROR GOTO error ;
        EXEC SQL DECLARE C5 CURSOR FOR
            SELECT MORE_THAN, MORE_EQU, LESS_THAN, LESS_EQU FROM G$C$
                WHERE ATTR_NAME = :attr_name ;
        EXEC SQL OPEN C5 ;
        while(1)
        {
            EXEC SQL FETCH C5 INTO :more_than :more_thani, :more_equ :more_equi,
            :less_than :less_thani, :less_equ :less_equi ;
            if (sqlca.sqlcode != 0)
                break ;
            if (more_thani == 0)
            {
                if (less_thani == 0)
                {
                    if (more_than == less_than)          /* -V- */
                        continue ;
                    else if (more_than < less_than)      /* V-V */
                    {
                        if (less_than <= number)
                        {
                            false_flg = 1 ;
                            break ;
                        }
                        else
                            continue ;
                    }
                    else                                 /* -V V- */
                        continue ;
                }
                else if (less_equi == 0)
                {
                    if (more_than < less_equ)            /* V-A */
                    {
                        if (less_equ <= number)
                        {
                            false_flg = 1 ;
                            break ;
                        }
                        else
                            continue ;
                    }
                    else                                 /* -A V- */
```

```
                continue ;
        }
        else        /* V- */
                continue ;
}
else if (more_equi == 0)
{
        if (less_thani == 0)
        {
                if (more_equ < less_than)        /* A-V */
                {
                        if (less_than <= number)
                        {
                                false_flg = 1 ;
                                break ;
                        }
                        else
                                continue ;
                }
                else                        /* -V A- */
                        continue ;
        }
        else if (less_equi == 0)
        {
                if (more_equ == less_equ)        /* A */
                {
                        if (more_equ <= number)
                        {
                                false_flg = 1 ;
                                break ;
                        }
                        else
                                continue ;
                }
                else if (more_equ < less_equ)        /* A-A */
                {
                        if (less_equ <= number)
                        {
                                false_flg = 1 ;
                                break ;
                        }
                        else
                                continue ;
                }
                else                        /* -A A- */
                        continue ;
        }
        else                                /* A- */
                continue ;
}
else if (less_thani == 0)                        /* -V */
{
        if (less_than <= number)
        {
                false_flg = 1 ;
                break ;
        }
        else
                continue ;
}
else if (less_equi == 0)                        /* -A */
{
        if (less_equ <= number)
        {
                false_flg = 1 ;
                break ;
        }
        else
```

```
                        continue ;
                }
        }
        EXEC SQL CLOSE C5 ;
        if (false_flg)
                return (FALSE) ;
        else
                return (TRUE) ;
error:
        reptError() ;
        err_quit("BackEnd: MoreCons() error\n") ;
}

int LessEquCons()
{
        int     false_flg ;
        float   number ;

        false_flg = 0 ;
        number = atof(POS->string) ;
        strcpy(attr_name.arr,POS->prev->prev->string) ;
        attr_name.len = strlen(attr_name.arr) ;
        EXEC SQL WHENEVER SQLERROR GOTO error ;
        EXEC SQL DECLARE C6 CURSOR FOR
                SELECT MORE_THAN, MORE_EQU, LESS_THAN, LESS_EQU FROM G$C$
                        WHERE ATTR_NAME = :attr_name ;
        EXEC SQL OPEN C6 ;
        while(1)
        {
                EXEC SQL FETCH C6 INTO :more_than :more_thani, :more_equ :more_equi,
                :less_than :less_thani, :less_equ :less_equi ;
                if (sqlca.sqlcode != 0)
                        break ;
                if (more_thani == 0)
                {
                        if (less_thani == 0)
                        {
                                if (more_than == less_than)             /* -V- */
                                        continue ;
                                else if (more_than < less_than)         /* V-V */
                                {
                                        if (more_than >= number)
                                        {
                                                false_flg = 1 ;
                                                break ;
                                        }
                                        else
                                                continue ;
                                }
                                else                                    /* -V V- */
                                        continue ;
                        }
                        else if (less_equi == 0)
                        {
                                if (more_than < less_equ)               /* V-A */
                                {
                                        if (more_than >= number)
                                        {
                                                false_flg = 1 ;
                                                break ;
                                        }
                                        else
                                                continue ;
                                }
                                else                                    /* -A V- */
                                        continue ;
                        }
                        else                                            /* V- */
```

```
        {
                if (more_than >= number)
                {
                        false_flg = 1 ;
                        break ;
                }
                else
                        continue ;
        }
    }
    else if (more_equi == 0)
    {
        if (less_thani == 0)
        {
                if (more_equ < less_than)          /* A-V */
                {
                        if (more_equ > number)
                        {
                                false_flg = 1 ;
                                break ;
                        }
                        else
                                continue ;
                }
                else                               /* -V A- */
                        continue ;
        }
        else if (less_equi == 0)
        {
                if (more_equ == less_equ)          /* A */
                {
                        if (more_equ > number)
                        {
                                false_flg = 1 ;
                                break ;
                        }
                        else
                                continue ;
                }
                else if (more_equ < less_equ)              /* A-A */
                {
                        if (more_equ > number)
                        {
                                false_flg = 1 ;
                                break ;
                        }
                        else
                                continue ;
                }
                else                               /* -A A- */
                        continue ;
        }
        else                                       /* A- */
        {
                if (more_equ > number)
                {
                        false_flg = 1 ;
                        break ;
                }
                else
                        continue ;
        }
    }
    else if (less_thani == 0)                      /* -V */
        continue ;
    else if (less_equi == 0)                       /* -A */
        continue ;
}
```

```
        EXEC SQL CLOSE C6 ;
        if (false_flg)
                return (FALSE) ;
        else
                return (TRUE) ;
error:
        reptError() ;
        err_quit("BackEnd: LessEquCons() error\n") ;
}

int MoreEquCons()
{
        int     false_flg ;
        float   number ;

        false_flg = 0 ;
        number = atof(POS->string) ;
        strcpy(attr_name.arr,POS->prev->prev->string) ;
        attr_name.len = strlen(attr_name.arr) ;
        EXEC SQL WHENEVER SQLERROR GOTO error ;
        EXEC SQL DECLARE C7 CURSOR FOR
                SELECT MORE_THAN, MORE_EQU, LESS_THAN, LESS_EQU FROM G$C$
                        WHERE ATTR_NAME = :attr_name ;
        EXEC SQL OPEN C7 ;
        while(1)
        {
                EXEC SQL FETCH C7 INTO :more_than :more_thani, :more_equ :more_equi,
                :less_than :less_thani, :less_equ :less_equi ;
                if (sqlca.sqlcode != 0)
                        break ;
                if (more_thani == 0)
                {
                        if (less_thani == 0)
                        {
                                if (more_than == less_than)             /* -V- */
                                        continue ;
                                else if (more_than < less_than)         /* V-V */
                                {
                                        if (less_than <= number)
                                        {
                                                false_flg = 1 ;
                                                break ;
                                        }
                                        else
                                                continue ;
                                }
                                else                            /* -V V- */
                                        continue ;
                        }
                        else if (less_equi == 0)
                        {
                                if (more_than < less_equ)       /* V-A */
                                {
                                        if (less_equ < number)
                                        {
                                                false_flg = 1 ;
                                                break ;
                                        }
                                        else
                                                continue ;
                                }
                                else                            /* -A V- */
                                        continue ;
                        }
                        else            /* V- */
                                continue ;
                }
                else if (more_equi == 0)
```

```
        {
                if (less_thani == 0)
                {
                        if (more_equ < less_than)          /* A-V */
                        {
                                if (less_than <= number)
                                {
                                        false_flg = 1 ;
                                        break ;
                                }
                                else
                                        continue ;
                        }
                        else                               /* -V A- */
                                continue ;
                }
                else if (less_equi == 0)
                {
                        if (more_equ == less_equ)          /* A */
                        {
                                if (less_equ < number)
                                {
                                        false_flg = 1 ;
                                        break ;
                                }
                                else
                                        continue ;
                        }
                        else if (more_equ < less_equ)      /* A-A */
                        {
                                if (less_equ < number)
                                {
                                        false_flg = 1 ;
                                        break ;
                                }
                                else
                                        continue ;
                        }
                        else                               /* -A A- */
                                continue ;
                }
                else                                       /* A- */
                        continue ;
        }
        else if (less_thani == 0)                          /* -V */
        {
                if (less_than <= number)
                {
                        false_flg = 1 ;
                        break ;
                }
                else
                        continue ;
        }
        else if (less_equi == 0)                           /* -A */
        {
                if (less_equ < number)
                {
                        false_flg = 1 ;
                        break ;
                }
                else
                        continue ;
        }
}
EXEC SQL CLOSE C7 ;
if (false_flg)
        return (FALSE) ;
```

```
        else
                return (TRUE) ;
error:
        reptError() ;
        err_quit("BackEnd: MoreEquCons() error\n") ;
}

/* return new view name , used to create view */
char *GetViewName()
{
        char    number[10] ;

        strcpy(vtemp_name[vt_order],temp_view) ;
        itoa(vt_order,number) ;
        strcat(vtemp_name[vt_order],number) ;
        vt_order++ ;
        return (vtemp_name[vt_order-1]) ;
}

void GetNewAttr(table)
char *table ;
{
        int     i, n ;

        strcpy(table_name.arr,table) ;
        table_name.len = strlen(table_name.arr) ;
        EXEC SQL WHENEVER SQLERROR GOTO error ;
        EXEC SQL WHENEVER NOT FOUND GOTO endloop ;
        for (i=0;;i++)
        {
                colno = i+1 ;
                EXEC SQL SELECT ATTR_NAME INTO :attr_name
                        FROM G$TA$
                        WHERE TABLE_NAME = :table_name
                        AND COL_NO = :colno ;
                strcpy(new_attr[i],table) ;
                strcat(new_attr[i],"__") ;
                n = strlen(new_attr[i]) ;
                strncat(new_attr[i],attr_name.arr,attr_name.len) ;
                n += attr_name.len ;
                new_attr[i][n] = '\0' ;
        }
endloop:
        n_new_attr = i ;
        EXEC SQL WHENEVER SQLERROR CONTINUE ;
        return ;
error:
        reptError() ;
        err_quit("BackEnd: GetNewAttr() error\n") ;
}

void GetTabAttr(table)
char *table ;
{
        int     i ;

        strcpy(table_name.arr,table) ;
        table_name.len = strlen(table_name.arr) ;
        EXEC SQL WHENEVER SQLERROR GOTO error ;
        EXEC SQL WHENEVER NOT FOUND GOTO endloop ;
        for (i=0;;i++)
        {
                colno = i+1 ;
                EXEC SQL SELECT ATTR_NAME INTO :attr_name
                FROM G$TA$
                WHERE TABLE_NAME = :table_name
                AND COL_NO = :colno ;
                strncpy(old_attr[i],attr_name.arr,attr_name.len) ;
```

```
                old_attr[i][attr_name.len] = '\0' ;
        }
endloop:
        n_old_attr = i ;
        EXEC SQL WHENEVER SQLERROR CONTINUE ;
        return ;
error:
        reptError() ;
        err_quit("BackEnd: GetTabAttr() error\n") ;
}

void GetAttr(viewname)
char *viewname ;
{
        int     i, j ;

        for (i=0;;i++)
        {
                for (j=0;j<v_order;j++)
                {
                        if ((strcmp(view[j].view_name,viewname) == 0) && (view[j].order == i+1))
                        {
                                strcpy(attr[i],view[j].new_attr) ;
                                break ;
                        }
                }
                if (j == v_order)
                        break ;
        }
        n_attr = i ;
        return ;
}

char *LexTabAttr(tab,attr,str)
char *tab, *attr, str[AGB_LEN] ;
{
        char    temp[30] ;
        int     i, n ;

        n = 0 ;
        while ((str[n] == ' ') || (str[n] == '\t'))
                n++ ;
        i = 0 ;
        while ((str[n] != ' ') && (str[n] != '\0') && (str[n] != ',') && (str[n] != '.'))
                temp[i++] = str[n++] ;
        temp[i] = '\0' ;
        if (str[n] == ',')
        {
                strcpy(attr,temp) ;
                strcpy(tab,"") ;
                n++ ;
                while ((str[n] == ' ') || (str[n] == '\t'))
                        n++ ;
                if (str[n] == '\0')
                        return ("") ;
                else
                        return (&str[n]) ;
        }
        else if (str[n] == '.')
        {
                strcpy(tab,temp) ;
                n++ ;
                i = 0 ;
                while ((str[n] != '\0') && (str[n] != ',') && (str[n] != ' '))
                        temp[i++] = str[n++] ;
                temp[i] = '\0' ;
                strcpy(attr,temp) ;
                while ((str[n] == ' ') || (str[n] == '\t'))
```

```
                n++ ;
            if (str[n] == '\0')
                return ("") ;
            else
                return (&str[n]) ;
        }
        else if (str[n] == '\0')
        {
            strcpy(attr,temp) ;
            strcpy(tab,"") ;
            return ("") ;
        }
        else
        {
            strcpy(attr,temp) ;
            strcpy(tab,"") ;
            while ((str[n] == ' ') || (str[n] == '\t'))
                n++ ;
            return (&str[n]) ;
        }
}

void FillAttr(adj_ptr)
ADJ *adj_ptr ;
{
        NAME *ptr1, *ptr2, *ptr3 ;
        int     flg ;

        if (strcmp(adj_ptr->type,LEAF) == 0)
        {
            GetTabAttr(adj_ptr->item->string) ;
            ptr1 = NULL ;
            while(n_old_attr > 0)
            {
                adj_ptr->attr_list = (NAME *)malloc(sizeof(NAME)) ;
                adj_ptr->attr_list->next = ptr1 ;
                ptr1 = adj_ptr->attr_list ;
                ptr1->prev = NULL ;
                strcpy(ptr1->string,old_attr[n_old_attr-1]) ;
                n_old_attr-- ;
            }
            return ;
        }
        else
        {
            if (adj_ptr->left != NULL )
                FillAttr(adj_ptr->left) ;
            if (adj_ptr->right != NULL )
                FillAttr(adj_ptr->right) ;
        }
        if (strcmp(adj_ptr->type,SELECT) == 0)
        {
            ptr1 = adj_ptr->left->attr_list ;
            flg = 0 ;
            while (ptr1 != NULL)
            {
                if (flg)
                {
                    ptr2->next = (NAME *)malloc(sizeof(NAME)) ;
                    ptr2->next->prev = ptr2 ;
                    ptr2 = ptr2->next ;
                    ptr2->next = NULL ;
                }
                else            /* first time */
                {
                    adj_ptr->attr_list = (NAME *)malloc(sizeof(NAME)) ;
                    ptr2 = adj_ptr->attr_list ;
                    ptr2->prev = NULL ;
```

```
                ptr2->next = NULL ;
                flg = 1 ;
            }
            strcpy(ptr2->string,ptr1->string) ;
            ptr1 = ptr1->next ;
        }
        return ;
}
if (strcmp(adj_ptr->type,PROJECT) == 0)
{
        ptr1 = adj_ptr->item ;
        flg = 0 ;
        while (ptr1 != NULL)
        {
            if (flg)
            {
                ptr2->next = (NAME *)malloc(sizeof(NAME)) ;
                ptr2->next->prev = ptr2 ;
                ptr2 = ptr2->next ;
                ptr2->next = NULL ;
            }
            else        /* first time */
            {
                adj_ptr->attr_list = (NAME *)malloc(sizeof(NAME)) ;
                ptr2 = adj_ptr->attr_list ;
                ptr2->prev = NULL ;
                ptr2->next = NULL ;
                flg = 1 ;
            }
            strcpy(ptr2->string,ptr1->string) ;
            ptr1 = ptr1->next ;
        }
        return ;
}
if ((strcmp(adj_ptr->type,UNION) == 0) || (strcmp(adj_ptr->type,MINUS) == 0)
|| (strcmp(adj_ptr->type,INTERSECT) == 0))
{
        ptr1 = adj_ptr->left->attr_list ;
        flg = 0 ;
        while (ptr1 != NULL)
        {
            if (flg)
            {
                ptr2->next = (NAME *)malloc(sizeof(NAME)) ;
                ptr2->next->prev = ptr2 ;
                ptr2 = ptr2->next ;
                ptr2->next = NULL ;
            }
            else        /* first time */
            {
                adj_ptr->attr_list = (NAME *)malloc(sizeof(NAME)) ;
                ptr2 = adj_ptr->attr_list ;
                ptr2->prev = NULL ;
                ptr2->next = NULL ;
                flg = 1 ;
            }
            strcpy(ptr2->string,ptr1->string) ;
            ptr1 = ptr1->next ;
        }
        return ;
}
if (strcmp(adj_ptr->type,TIMES) == 0)
{
        ptr1 = adj_ptr->left->attr_list ;
        flg = 0 ;
        while (ptr1 != NULL)
        {
            if (flg)
```

```
                {
                        ptr2->next = (NAME *)malloc(sizeof(NAME)) ;
                        ptr2->next->prev = ptr2 ;
                        ptr2 = ptr2->next ;
                        ptr2->next = NULL ;
                }
                else            /* first time */
                {
                        adj_ptr->attr_list = (NAME *)malloc(sizeof(NAME)) ;
                        ptr2 = adj_ptr->attr_list ;
                        ptr2->prev = NULL ;
                        ptr2->next = NULL ;
                        flg = 1 ;
                }
                strcpy(ptr2->string,ptr1->string) ;
                ptr1 = ptr1->next ;
        }
        ptr1 = adj_ptr->right->attr_list ;
        while (ptr1 != NULL)
        {
                ptr2->next = (NAME *)malloc(sizeof(NAME)) ;
                ptr2->next->prev = ptr2 ;
                ptr2 = ptr2->next ;
                ptr2->next = NULL ;
                strcpy(ptr2->string,ptr1->string) ;
                ptr1 = ptr1->next ;
        }
        return ;
}
if (strcmp(adj_ptr->type,JOIN) == 0)
{
        ptr1 = adj_ptr->left->attr_list ;
        flg = 0 ;
        while (ptr1 != NULL)
        {
                if (flg)
                {
                        ptr2->next = (NAME *)malloc(sizeof(NAME)) ;
                        ptr2->next->prev = ptr2 ;
                        ptr2 = ptr2->next ;
                        ptr2->next = NULL ;
                }
                else            /* first time */
                {
                        adj_ptr->attr_list = (NAME *)malloc(sizeof(NAME)) ;
                        ptr2 = adj_ptr->attr_list ;
                        ptr2->prev = NULL ;
                        ptr2->next = NULL ;
                        flg = 1 ;
                }
                strcpy(ptr2->string,ptr1->string) ;
                ptr1 = ptr1->next ;
        }
        ptr1 = adj_ptr->right->attr_list ;
        while (ptr1 != NULL)
        {
                ptr3 = adj_ptr->attr_list ;
                while (ptr3 != NULL)
                {
                        if (strcmp(ptr3->string,ptr1->string) == 0)
                                break ;
                        ptr3 = ptr3->next ;
                }
                if (ptr3 == NULL)
                {
                        ptr2->next = (NAME *)malloc(sizeof(NAME)) ;
                        ptr2->next->prev = ptr2 ;
                        ptr2 = ptr2->next ;
```

```
                        ptr2->next = NULL ;
                        strcpy(ptr2->string,ptr1->string) ;
                    }
                    ptr1 = ptr1->next ;
                }
                return ;
            }
            if (strcmp(adj_ptr->type,DIVIDEBY) == 0)
            {
                ptr1 = adj_ptr->left->attr_list ;
                flg = 0 ;
                while (ptr1 != NULL)
                {
                    if (flg)
                    {
                        ptr2->next = (NAME *)malloc(sizeof(NAME)) ;
                        ptr2->next->prev = ptr2 ;
                        ptr2 = ptr2->next ;
                        ptr2->next = NULL ;
                    }
                    else        /* first time */
                    {
                        adj_ptr->attr_list = (NAME *)malloc(sizeof(NAME)) ;
                        ptr2 = adj_ptr->attr_list ;
                        ptr2->prev = NULL ;
                        ptr2->next = NULL ;
                        flg = 1 ;
                    }
                    strcpy(ptr2->string,ptr1->string) ;
                    ptr1 = ptr1->next ;
                }
                ptr1 = adj_ptr->right->attr_list ;
                while (ptr1 != NULL)
                {
                    ptr3 = adj_ptr->attr_list ;
                    while (ptr3 != NULL)
                    {
                        if (strcmp(ptr3->string,ptr1->string) == 0)
                        {
                            if (ptr3 != adj_ptr->attr_list)
                                ptr3->prev->next = ptr3->next ;
                            else
                                adj_ptr->attr_list = ptr3->next ;
                            if (ptr3->next != NULL)
                                ptr3->next->prev = ptr3->prev ;
                            ptr2 = ptr3 ;
                            ptr3 = ptr3->next ;
                            free (ptr2) ;
                            break ;
                        }
                        else
                            ptr3 = ptr3->next ;
                    }
                    ptr1 = ptr1->next ;
                }
                return ;
            }
        }

        ADJ *RuleChk(top)
        ADJ *top ;
        {
            ADJ   *adj_ptr, *adj1, *adj2, *adj3 ;
            NAME*ptr1, *ptr2, *ptr3 ;
            int     flg, not_first ;

            adj_ptr = top ;
            do
```

```
{
        flg = 0 ;
        if (strcmp(adj_ptr->type,SELECT) == 0)
        {
                if (strcmp(adj_ptr->left->type,SELECT) == 0)
                {
                        flg = 1 ;
                        adj1 = adj_ptr ;
                        adj2 = adj_ptr->left ;
                        ptr1 = (NAME *)malloc(sizeof(NAME)) ;
                        ptr1->next = adj1->item ;
                        adj1->item->prev = ptr1 ;
                        strcpy(ptr1->string,"(") ;
                        adj1->item = ptr1 ;
                        ptr2 = (NAME *)malloc(sizeof(NAME)) ;
                        ptr2->next = NULL ;
                        strcpy(ptr2->string,")") ;
                        while (ptr1->next != NULL)
                                ptr1 = ptr1->next ;
                        ptr2->prev = ptr1 ;
                        ptr1->next = ptr2 ;
                        ptr1 = (NAME *)malloc(sizeof(NAME)) ;
                        ptr1->prev = ptr2 ;
                        ptr2->next = ptr1 ;
                        strcpy(ptr1->string,"AND") ;
                        ptr2 = (NAME *)malloc(sizeof(NAME)) ;
                        ptr2->prev = ptr1 ;
                        ptr1->next = ptr2 ;
                        strcpy(ptr2->string,"(") ;
                        ptr2->next = adj2->item ;
                        while (ptr2->next != NULL)
                                ptr2 = ptr2->next ;
                        ptr1 = (NAME *)malloc(sizeof(NAME)) ;
                        ptr2->next = ptr1 ;
                        ptr1->prev = ptr2 ;
                        ptr1->next = NULL ;
                        strcpy(ptr1->string,")") ;
                        ptr1 = adj2->attr_list ;
                        while (ptr1 != NULL)
                        {
                                ptr2 = ptr1 ;
                                ptr1 = ptr1->next ;
                                free (ptr2) ;
                        }
                        adj1->left = adj2->left ;
                        adj2->left->parent = adj1 ;
                        free (adj2) ;
                        continue ;
                }
                if ((strcmp(adj_ptr->left->type,UNION) == 0) || (strcmp(adj_ptr->left->type,MINUS) == 0))
                {
                        flg = 1 ;
                        adj1 = adj_ptr ;
                        adj2 = adj_ptr->left ;
                        adj3 = (ADJ *)malloc(sizeof(ADJ)) ;
                        adj3->right = NULL ;
                        not_first = 0 ;
                        ptr1 = adj1->item ;
                        while (ptr1 != NULL)
                        {
                                if (not_first)
                                {
                                        ptr2->next = (NAME *)malloc(sizeof(NAME)) ;
                                        ptr2->next->prev = ptr2 ;
                                        ptr2 = ptr2->next ;
                                        ptr2->next = NULL ;
                                }
                                else
```

```
                {
                        adj3->item = (NAME *)malloc(sizeof(NAME)) ;
                        ptr2 = adj3->item ;
                        ptr2->next = NULL ;
                        ptr2->prev = NULL ;
                        not_first = 1 ;
                }
                strcpy(ptr2->string,ptr1->string) ;
                ptr1 = ptr1->next ;
        }
        not_first = 0 ;
        ptr1 = adj1->attr_list ;
        while (ptr1 != NULL)
{
                if (not_first)
                {
                        ptr2->next = (NAME *)malloc(sizeof(NAME)) ;
                        ptr2->next->prev = ptr2 ;
                        ptr2 = ptr2->next ;
                        ptr2->next = NULL ;
                }
                else
                {
                        adj3->attr_list = (NAME *)malloc(sizeof(NAME)) ;
                        ptr2 = adj3->attr_list ;
                        ptr2->next = NULL ;
                        ptr2->prev = NULL ;
                        not_first = 1 ;
                }
                strcpy(ptr2->string,ptr1->string) ;
                ptr1 = ptr1->next ;
        }
        adj3->parent = adj1 ;
        adj1->right = adj3 ;
        adj3->left = adj2->right ;
        adj3->left->parent = adj3 ;
        adj2->right = NULL ;
        if (strcmp(adj2->type,UNION) == 0)
                strcpy(adj1->type,UNION) ;
        else
                strcpy(adj1->type,MINUS) ;
        strcpy(adj2->type,SELECT) ;
        adj2->item = adj1->item ;
        adj1->item = NULL ;
        continue ;
}
if (strcmp(adj_ptr->left->type,PROJECT) == 0)
{
        flg = 1 ;
        adj1 = adj_ptr ;
        adj2 = adj_ptr->left ;
        ptr1 = adj1->item ;
        ptr2 = adj1->attr_list ;
        adj1->item = adj2->item ;
        adj1->attr_list = adj2->attr_list ;
        strcpy(adj1->type,PROJECT) ;
        adj2->item = ptr1 ;
        strcpy(adj2->type,SELECT) ;
        while(ptr2 != NULL)
        {
                ptr1 = ptr2->next ;
                free (ptr2) ;
                ptr2 = ptr1 ;
        }
        ptr1 = adj2->left->attr_list ;
        not_first = 0 ;
        while (ptr1 != NULL)
        {
```

```
                        if (not_first)
                        {
                                ptr2->next = (NAME *)malloc(sizeof(NAME)) ;
                                ptr2->next->prev = ptr2 ;
                                ptr2->next->next = NULL ;
                                ptr2 = ptr2->next ;
                        }
                        else   /* first loop */
                        {
                                adj2->attr_list = (NAME *)malloc(sizeof(NAME)) ;
                                ptr2 = adj2->attr_list ;
                                ptr2->prev = NULL ;
                                ptr2->next = NULL ;
                        }
                        strcpy(ptr2->string,ptr1->string) ;
                }
                continue ;
        }
}
if (strcmp(adj_ptr->type,PROJECT) == 0)
{
        if (strcmp(adj_ptr->left->type,PROJECT) == 0)
        {
                flg = 1 ;
                ptr1 = adj_ptr->attr_list ;
                while (ptr1 != NULL)
                {
                        ptr2 = adj_ptr->left->attr_list ;
                        while (ptr2 != NULL)
                        {
                                if (strcmp(ptr1->string,ptr2->string) == 0)
                                        break ;
                                ptr2 = ptr2->next ;
                        }
                        if (ptr2 == NULL)
                        {
                                flg = 0 ;
                                break ;
                        }
                        ptr1 = ptr1->next ;
                }
                if (flg)
                {
                        adj_ptr->left = adj_ptr->left->left ;
                        Free_Adj(adj_ptr->left->parent) ;
                        adj_ptr->left->parent = adj_ptr ;
                }
                continue ;
        }
        if (strcmp(adj_ptr->left->type,TIMES) == 0)
        {
                flg = 1 ;
                adj1 = adj_ptr ;
                adj2 = adj_ptr->left ;
                adj3 = (ADJ *)malloc(sizeof(ADJ)) ;
                adj3->attr_list = NULL ;
                ptr1 = adj1->attr_list ;
                not_first = 0 ;
                while (ptr1 != NULL)
                {
                        ptr2 = adj1->left->left->attr_list ;
                        while (ptr2 != NULL)
                        {
                                if (strcmp(ptr1->string,ptr2->string) == 0)
                                        break ;
                        }
                        if (ptr2 == NULL)
                        {
```

```
                if (not_first)
                {
                        ptr3->next = ptr1 ;
                        ptr1->prev->next = ptr1->next ;
                        ptr1->next->prev = ptr1->prev ;
                        ptr1 = ptr1->prev ;
                        ptr3->next->next = NULL ;
                        ptr3->next->prev = ptr3 ;
                        ptr3 = ptr3->next ;
                }
                else
                {
                        adj3->attr_list = ptr1 ;
                        ptr1->prev->next = ptr1->next ;
                        ptr1->next->prev = ptr1->prev ;
                        ptr1 = ptr1->prev ;
                        ptr3 = adj3->attr_list ;
                        ptr3->prev = NULL ;
                        ptr3->next = NULL ;
                        not_first = 1 ;
                }
        }
        ptr1 = ptr1->next ;
}
if (adj3->attr_list = NULL)
        flg = 0 ;
else
{
        ptr1 = adj2->attr_list ;
        while (ptr1 != NULL)
        {
                ptr2 = ptr1 ;
                ptr1 = ptr1->next ;
                free (ptr2) ;
        }
        adj1->right = adj3 ;
        adj3->parent = adj1 ;
        adj3->left = adj2->right ;
        adj3->left->parent = adj3 ;
        adj3->right = NULL ;
        adj2->right = NULL ;
        strcpy(adj2->type,PROJECT) ;
        adj2->attr_list = adj1->attr_list ;
        ptr1 = adj2->attr_list ;
        not_first = 0 ;
        while (ptr1 != NULL)
        {
                if (not_first)
                {
                        ptr2->next = (NAME *)malloc(sizeof(NAME)) ;
                        ptr2->next->prev = ptr2 ;
                        ptr2 = ptr2->next ;
                        ptr2->next = NULL ;
                }
                else
                {
                        adj2->item = (NAME *)malloc(sizeof(NAME)) ;
                        ptr2 = adj2->item ;
                        ptr2->prev = NULL ;
                        ptr2->next = NULL ;
                        not_first = 1 ;
                }
                strcpy(ptr2->string,ptr1->string) ;
                ptr1 = ptr1->next ;
        }
        strcpy(adj3->type,PROJECT) ;
        ptr1 = adj3->attr_list ;
        not_first = 0 ;
```

```
                while (ptr1 != NULL)
                {
                        if (not_first)
                        {
                                ptr2->next = (NAME *)malloc(sizeof(NAME)) ;
                                ptr2->next->prev = ptr2 ;
                                ptr2 = ptr2->next ;
                                ptr2->next = NULL ;
                        }
                        else
                        {
                                adj3->item = (NAME *)malloc(sizeof(NAME)) ;
                                ptr2 = adj3->item ;
                                ptr2->prev = NULL ;
                                ptr2->next = NULL ;
                                not_first = 1 ;
                        }
                        strcpy(ptr2->string,ptr1->string) ;
                        ptr1 = ptr1->next ;
                }
                strcpy(adj1->type,TIMES) ;
                adj1->attr_list = adj1->item ;
                adj1->item = NULL ;
        }
        continue ;
}
if (strcmp(adj_ptr->left->type,UNION) == 0)
{
        flg = 1 ;
        adj1 = adj_ptr ;
        adj2 = adj_ptr->left ;
        adj3 = (ADJ *)malloc(sizeof(ADJ)) ;
        strcpy(adj3->type,adj1->type) ;
        ptr1 = adj1->item ;
        not_first = 0 ;
        while (ptr1 != NULL)
        {
                if (not_first)
                {
                        ptr2->next = (NAME *)malloc(sizeof(NAME)) ;
                        ptr2->next->prev = ptr2 ;
                        ptr2 = ptr2->next ;
                        ptr2->next = NULL ;
                }
                else            /* first time */
                {
                        adj3->item = (NAME *)malloc(sizeof(NAME)) ;
                        ptr2 = adj3->item ;
                        ptr2->prev = NULL ;
                        ptr2->next = NULL ;
                        not_first = 1 ;
                }
                strcpy(ptr2->string,ptr1->string) ;
                ptr1 = ptr1->next ;
        }
        ptr1 = adj1->attr_list ;
        not_first = 0 ;
        while (ptr1 != NULL)
        {
                if (not_first)
                {
                        ptr2->next = (NAME *)malloc(sizeof(NAME)) ;
                        ptr2->next->prev = ptr2 ;
                        ptr2 = ptr2->next ;
                        ptr2->next = NULL ;
                }
                else            /* first time */
                {
```

```
                                adj3->attr_list = (NAME *)malloc(sizeof(NAME)) ;
                                ptr2 = adj3->attr_list ;
                                ptr2->prev = NULL ;
                                ptr2->next = NULL ;
                                not_first = 1 ;
                        }
                        strcpy(ptr2->string,ptr1->string) ;
                        ptr1 = ptr1->next ;
                }
                adj3->right = NULL ;
                adj3->left = adj2->right ;
                adj3->left->parent = adj3 ;
                adj3->parent = adj1 ;
                ptr1 = adj2->item ;
                ptr2 = adj2->attr_list ;
                adj2->item = adj1->item ;
                adj2->attr_list = adj1->attr_list ;
                strcpy(adj2->type,PROJECT) ;
                adj2->right = NULL ;
                adj1->right = adj3 ;
                adj1->item = ptr1 ;
                adj1->attr_list = ptr2 ;
                strcpy(adj1->type,UNION) ;
                continue ;
            }
        }
    }while (flg == 1) ;
    if (adj_ptr->left != NULL)
        RuleChk(adj_ptr->left) ;
    if (adj_ptr->right != NULL)
        RuleChk(adj_ptr->right) ;
    return (adj_ptr) ;
}

void Free_Adj(top)
ADJ *top ;
{
    NAME *ptr ;

    if (top->left != NULL)
        Free_Adj(top->left) ;
    if (top->right != NULL)
        Free_Adj(top->right) ;
    if (top->item != NULL)
    {
        ptr = top->item ;
        while (ptr->next != NULL)
        {
            ptr = ptr->next ;
            free (ptr->prev) ;
        }
        free (ptr) ;
    }
    if (top->attr_list != NULL)
    {
        ptr = top->attr_list ;
        while (ptr->next != NULL)
        {
            ptr = ptr->next ;
            free (ptr->prev) ;
        }
        free (ptr) ;
    }
    free (top) ;
}

char *LexAdj(word,words)
char *word, *words ;
```

```
{
        while(*words == ' ')
                words++ ;
        while((*words != ' ') && (*words != '\0') && (*words != ','))
        {
                *word++ = *words++ ;
        }
        *word = '\0' ;
        while((*words == ' ') || (*words == ','))
                words++ ;
        if (*words != '\0')
                return (words) ;
        else
                return (NULL) ;
}
```

3. ส่วนเชื่อมต่อโครงข่าย

file network.mak

```
TOOL=$(HOME)/prog/tool
OBJECT=network.o tool.o error.o
network     : $(OBJECT)
                cc $(CFLAGS) $(OBJECT) -o network
network.o   : network.c  $(TOOL)/def.h
                cc -c $(CFLAGS) network.c
tool.o      : $(TOOL)/tool.c $(TOOL)/def.h
                cc -c $(CFLAGS) $(TOOL)/tool.c
error.o     : $(TOOL)/error.c
                cc -c $(CFLAGS) $(TOOL)/error.c
```

file network.c

```
#include        "../tool/def.h"
#include        "../tool/proto.h"
main(argc,argv)
int     argc;
char    *argv[];
{
        char    serv_host_addr[TOKEN_LEN], buff[MAXLINE] ;
        int     i_sockfd, u_sockfd, newsockfd, maxfdp1, cli_len, childpid,
                u_serv_len, nfound , serv_len, nread ;
        fd_set readmask ;
        struct sockaddr_in  i_cli_addr, i_serv_addr ;
        struct sockaddr_un  u_cli_addr, u_serv_addr ;
        struct hostent      *remote_host ;

        if( (i_sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
                err_dump("network: can't open internet socket") ;
        memset((char *) &i_serv_addr, 0, sizeof(i_serv_addr)) ;
        i_serv_addr.sin_family = AF_INET ;
        i_serv_addr.sin_addr.s_addr = htonl(INADDR_ANY) ;
        i_serv_addr.sin_port = htons(SERV_TCP_PORT) ;

        if( bind(i_sockfd, (struct sockaddr *) &i_serv_addr, sizeof(i_serv_addr)) < 0)
                err_dump("server: can't bind local address");
        listen (i_sockfd,5) ;

        if( (u_sockfd = socket(AF_UNIX, SOCK_STREAM, 0)) < 0)
                err_dump("network: can't open unix socket") ;
        memset((char *) &u_serv_addr, 0, sizeof(u_serv_addr)) ;
        u_serv_addr.sun_family = AF_UNIX ;
        strcpy(u_serv_addr.sun_path, UNIXSTR_PATH1) ;
        u_serv_len = strlen(u_serv_addr.sun_path) + sizeof(u_serv_addr.sun_family) ;

        if( bind(u_sockfd, (struct sockaddr *) &u_serv_addr, u_serv_len) < 0)
                err_dump("server: can't bind local address");
        listen (u_sockfd,5) ;
        for (;;)
        {
                FD_ZERO (&readmask) ;
                FD_SET (i_sockfd, &readmask) ;
                FD_SET (u_sockfd, &readmask) ;
                maxfdp1 = (i_sockfd < u_sockfd)? u_sockfd+1 : i_sockfd+1 ;
                nfound = select(maxfdp1, &readmask, (fd_set *) 0, (fd_set *) 0, (struct timeval *) 0) ;
                if (nfound < 0)
                        err_sys("select error") ;
                if (FD_ISSET(i_sockfd, &readmask))   /* connect from inet */
                {
```

```
            cli_len = sizeof(i_cli_addr) ;
            newsockfd = accept(i_sockfd, (struct sockaddr *) &i_cli_addr, &cli_len) ;
            if(newsockfd < 0)
                  err_dump("server: accept inet error");
            if( (childpid = fork()) < 0)
                  err_dump("server: fork error");
            else if (childpid == 0)  /* child process */
            {
                  close(u_sockfd) ;
                  close(i_sockfd) ;
                  memset((char *) &u_serv_addr, 0, sizeof(u_serv_addr)) ;
                  u_serv_addr.sun_family = AF_UNIX ;
                  strcpy(u_serv_addr.sun_path, UNIXSTR_PATH0) ;
                  u_serv_len = strlen(u_serv_addr.sun_path) + sizeof(u_serv_addr.sun_family) ;
                  if ( (u_sockfd = socket(AF_UNIX, SOCK_STREAM, 0)) < 0)
                        err_sys("client: can't open stream socket") ;
                  if (connect(u_sockfd, (struct sockaddr *) &u_serv_addr, u_serv_len) < 0)
                        err_sys("client: can't connect to back-end server") ;
                  pass_data(newsockfd,u_sockfd) ;
                  close(u_sockfd) ;
                  exit (0) ;
            }
            else
                  close(newsockfd) ;          /* parent process */
      }
      if (FD_ISSET(u_sockfd, &readmask))      /* connect from unix */
      {
            cli_len = sizeof(u_cli_addr) ;
            newsockfd = accept(u_sockfd, (struct sockaddr *) &u_cli_addr, &cli_len) ;
            if (newsockfd < 0)
                  err_dump("server: accept unix error") ;
            if ( (childpid = fork()) < 0)
                  err_dump("server: fork error") ;
            else if (childpid == 0)  /* child process */
            {
                  close(i_sockfd) ;
                  close(u_sockfd) ;
                  nread = readline(newsockfd, buff, MAXLINE) ;
                  if (nread < 0)
                        err_sys("server: read host name error") ;
                  if (nread == 0)
                        exit (0) ;
                  buff[nread] = '\0' ;
                  remote_host = gethostbyname(buff) ;
                  switch (remote_host->h_addrtype)
                  {
                        case AF_INET:
                              strcpy(serv_host_addr,inet_ntoa(*(struct in_addr *) *remote_host->h_addr_list)) ;
                              break ;
                        default:
                              err_quit("unknown address type") ;
                              break ;
                  }
                  memset((char *) &i_serv_addr, 0, sizeof(i_serv_addr)) ;
                  i_serv_addr.sin_family = AF_INET ;
                  i_serv_addr.sin_addr.s_addr = inet_addr(serv_host_addr) ;
                  i_serv_addr.sin_port = htons(SERV_TCP_PORT) ;
                  if ( (i_sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
                        err_sys("client: can't open inet socket") ;
                  if (connect(i_sockfd, (struct sockaddr *) &i_serv_addr, sizeof(i_serv_addr)) < 0)
                        err_sys("client: can't connect inet server") ;
                  pass_data (newsockfd, i_sockfd) ;
                  exit (0) ;
            }
            else
                  close(newsockfd) ;          /* parent process */
      }
}
```

```
}

int pass_data( fd1, fd2)
int fd1, fd2;
{
        int     maxfdp1, nfound, nread ;
        char    buff[MAXLINE] ;
        fd_set readmask ;

        FD_ZERO(&readmask) ;
        for (;;)
        {
                FD_SET(fd1, &readmask) ;
                FD_SET(fd2, &readmask) ;
                maxfdp1 = (fd1 > fd2)? fd1+1 : fd2+1 ;
                nfound = select(maxfdp1, &readmask, (fd_set *) 0, (fd_set *) 0, (struct timeval *) 0) ;
                if (nfound < 0)
                        err_sys("pass_data: select error") ;
                if (FD_ISSET(fd1, &readmask))           /* data come from fd1 */
                {
                        nread = readline(fd1, buff, MAXLINE) ;
                        if (nread < 0)
                                err_sys("pass_data: read error from fd1") ;
                        else if (nread == 0)
                                break ;                 /* EOF */
                        if (writen(fd2, buff, nread) != nread)
                                err_sys("pass_data: writen error to fd2") ;
                }
                if (FD_ISSET(fd2, &readmask))           /* data come from fd2 */
                {
                        nread = readline(fd2, buff, MAXLINE) ;
                        if (nread < 0)
                                err_sys("pass_data: read error from fd2") ;
                        else if (nread == 0)
                                break ;                 /* EOF */
                        if (writen(fd1, buff, nread) != nread)
                                err_sys("pass_data: writen error to fd1") ;
                }
        }
}
```

4. ส่วนสร้างและลบตารางพจนานุกรมฐานข้อมูล

file mak_tab.mak

```
TOOL=$(HOME)/prog/tool
OBJECT= mak_tab.o
LIBS= $(ORACLE_HOME)/pro/libsql.a $(ORACLE_HOME)/oci/libhli.a $(ORACLE_HOME)/oci/libhlic.a

mak_tab     : $(OBJECT)
              cc $(CFLAGS) -o mak_tab $(OBJECT) $(LIBS)
mak_tab.c   : mak_tab.pc
              $(TOOL)/precompi mak_tab.pc
mak_tab.o   : mak_tab.c
              cc -c $(CFLAGS) mak_tab.c
```

file mak_tab.pc

```
#include        <stdio.h>
EXEC SQL BEGIN DECLARE SECTION ;
    VARCHAR uid[30] ;
    VARCHAR pwd[30] ;
EXEC SQL END DECLARE SECTION ;
EXEC SQL INCLUDE SQLCA ;
main ()
{
    printf("\nThis program will create new table all") ;
    printf("\nEnter Oracle's USER NAME: ") ;
    scanf("%s",uid.arr) ;
    uid.len = strlen(uid.arr) ;
    strcpy(pwd.arr, getpass("Password :")) ;
    pwd.len = strlen (pwd.arr) ;

    EXEC SQL WHENEVER SQLERROR GOTO error ;
    EXEC SQL CONNECT :uid IDENTIFIED BY :pwd ;
    printf("Connect to ORACLE user: %s \n",uid.arr) ;

    EXEC SQL CREATE TABLE G$THU$
        (TABLE_NAME CHAR(30) NOT NULL,
        HOST_NAME CHAR(30) NOT NULL,
        USER_NAME CHAR(30) NOT NULL) ;
    EXEC SQL CREATE UNIQUE INDEX GLO_TAB
        ON G$THU$ (TABLE_NAME) ;
    printf("Table G$THU$ created.\n") ;

    EXEC SQL CREATE TABLE G$TA$
        (TABLE_NAME CHAR(30) NOT NULL,
        ATTR_NAME CHAR(30) NOT NULL,
        COL_NO NUMBER NOT NULL) ;
    EXEC SQL CREATE UNIQUE INDEX GLO_TAB_ATT
        ON G$TA$ (TABLE_NAME, ATTR_NAME) ;
    printf("Table G$TA$ created.\n") ;

    EXEC SQL CREATE TABLE G$A$
        (ATTR_NAME CHAR(30) NOT NULL,
        TYPE CHAR(10) NOT NULL,
        WIDTH NUMBER NOT NULL) ;
    EXEC SQL CREATE UNIQUE INDEX GLO_ATT
        ON G$A$ (ATTR_NAME) ;
    printf("Table G$A$ created.\n") ;

    EXEC SQL CREATE TABLE G$C$
        (CONST_NAME CHAR(30) NOT NULL,
        ATTR_NAME CHAR(30) NOT NULL,
```

```
            MORE_THAN NUMBER,
            MORE_EQU  NUMBER,
            LESS_THAN NUMBER,
            LESS_EQU  NUMBER ) ;
        EXEC SQL CREATE UNIQUE INDEX GLO_CONSTRAIN
            ON G$C$ (CONST_NAME) ;
        printf("Table G$C$ created.\n") ;

        EXEC SQL CREATE TABLE L$THU$
            (TABLE_NAME CHAR(30) NOT NULL,
            HOST_NAME CHAR(30) NOT NULL,
            USER_NAME CHAR(30) NOT NULL) ;
        EXEC SQL CREATE UNIQUE INDEX LOC_TAB
            ON L$THU$ (TABLE_NAME) ;
        printf("Table L$THU$ created.\n") ;

        EXEC SQL CREATE TABLE L$TA$
            (TABLE_NAME CHAR(30) NOT NULL,
            ATTR_NAME CHAR(30) NOT NULL,
            COL_NO NUMBER NOT NULL) ;
        EXEC SQL CREATE UNIQUE INDEX LOC_TAB_ATT
            ON L$TA$ (TABLE_NAME, ATTR_NAME) ;
        printf("Table L$TA$ created.\n") ;

        EXEC SQL CREATE TABLE L$A$
            (ATTR_NAME CHAR(30) NOT NULL,
            TYPE CHAR(10) NOT NULL,
            WIDTH NUMBER NOT NULL) ;
        EXEC SQL CREATE UNIQUE INDEX LOC_ATT
            ON L$A$ (ATTR_NAME) ;
        printf("Table L$A$ created.\n") ;

        EXEC SQL CREATE TABLE L$C$
            (CONST_NAME CHAR(30) NOT NULL,
            ATTR_NAME CHAR(30) NOT NULL,
            MORE_THAN NUMBER,
            MORE_EQU  NUMBER,
            LESS_THAN NUMBER,
            LESS_EQU  NUMBER ) ;
        EXEC SQL CREATE UNIQUE INDEX LOC_CONST
            ON L$C$ (CONST_NAME) ;
        printf("Table L$C$ created.\n") ;
        EXEC SQL COMMIT WORK RELEASE ;
        exit(0) ;

error:
        EXEC SQL WHENEVER SQLERROR CONTINUE ;
        printf("Can't create local data dictionary table \n") ;
        printf("%.70s \n",sqlca.sqlerrm.sqlerrmc) ;
        EXEC SQL ROLLBACK WORK RELEASE ;
        exit(-1) ;
}
```

file drop_tab.mak

```
TOOL=$(HOME)/prog/tool
OBJECT=drop_tab.o
LIBS= $(ORACLE_HOME)/pro/libsql.a $(ORACLE_HOME)/oci/libhli.a $(ORACLE_HOME)/oci/libhlic.a

drop_tab    : $(OBJECT)
                cc $(CFLAGS) -o drop_tab $(OBJECT) $(LIBS)
drop_tab.c  : drop_tab.pc
                $(TOOL)/precompi drop_tab.pc
drop_tab.o  : drop_tab.c
                cc -c $(CFLAGS) drop_tab.c
```

file drop_tab.pc

```
#include        <stdio.h>

EXEC SQL BEGIN DECLARE SECTION ;
        VARCHAR uid[20] ;
        VARCHAR pwd[20] ;
EXEC SQL END DECLARE SECTION ;
EXEC SQL INCLUDE SQLCA ;
main ()
{
        printf("\nEnter Oracle's USER NAME: ") ;
        scanf("%s",uid.arr) ;
        uid.len = strlen(uid.arr) ;
        strcpy(pwd.arr, getpass("Password :")) ;
        pwd.len = strlen (pwd.arr) ;

        EXEC SQL WHENEVER SQLERROR GOTO error ;
        EXEC SQL CONNECT :uid IDENTIFIED BY :pwd ;
        printf("Connect to ORACLE user: %s \n",uid.arr) ;

        EXEC SQL DROP TABLE G$THU$ ;
        EXEC SQL DROP TABLE G$TA$ ;
        EXEC SQL DROP TABLE G$A$ ;
        EXEC SQL DROP TABLE G$C$ ;
        EXEC SQL DROP TABLE L$THU$ ;
        EXEC SQL DROP TABLE L$TA$ ;
        EXEC SQL DROP TABLE L$A$ ;
        EXEC SQL DROP TABLE L$C$ ;
        printf("Tables droped.\n") ;
        EXEC SQL COMMIT WORK RELEASE ;
        exit(0) ;

error:
        printf("%.70s \n",sqlca.sqlerrm.sqlerrmc) ;
}
```

5. ส่วนป้อนข้อมูลพจนานุกรมฐานข้อมูล

file dic_glo.mak

```
TOOL=$(HOME)/prog/tool
OBJECT=dic_glo.o tool.o error.o
LIBS= $(ORACLE_HOME)/pro/libsql.a $(ORACLE_HOME)/oci/libhli.a $(ORACLE_HOME)/oci/libhlic.a

dic_glo     : $(OBJECT)
              cc $(CFLAGS) -o dic_glo $(OBJECT) $(LIBS)
dic_glo.c   : dic_glo.pc $(TOOL)/def.h
              $(TOOL)/precompi dic_glo.pc
dic_glo.o   : dic_glo.c
              cc -c $(CFLAGS) dic_glo.c
tool.o      : $(TOOL)/tool.c  $(TOOL)/def.h
              cc -c $(CFLAGS) $(TOOL)/tool.c
error.o     : $(TOOL)/error.c $(TOOL)/def.h
              cc -c $(CFLAGS) $(TOOL)/error.c
```

file dic_glo.h

```
TREE_PTR        tree_THU() ;
void            recv_THU() ;
void            insert_THU() ;
TREE_PTR        tree_TA() ;
void            recv_TA() ;
void            insert_TA() ;
TREE_PTR        tree_A() ;
void            recv_A() ;
TREE_PTR        tree_CONS() ;
void            recv_CONS() ;
void            insert_CONS() ;
int             ins_t_h_u() ;
int             ins_t_a_a() ;
```

file dic_glo.pc

```
/*-------------------------------------------------------------------------*/
/* This program assume table of local and global data dictionary were      */
/* already exist. If table not exist, run "table_dic".                     */
/*-------------------------------------------------------------------------*/

#include        "../tool/def.h"
#include        "../tool/proto.h"
#include        "dic_glo.h"

int     order, fld_cnt ;

EXEC SQL BEGIN DECLARE SECTION ;
        VARCHAR     uid[20] ;
        VARCHAR     pwd[20] ;
        VARCHAR     table_name[30] ;
        VARCHAR     host_name[30] ;
        VARCHAR     user_name[30] ;
        VARCHAR     attr_name[30] ;
        VARCHAR     type[10] ;
        int     width,nn,colno ;
        VARCHAR     const_name[30] ;
        float   more_than, more_equ ;
        short   imore_than, imore_equ ;
        float   less_than, less_equ ;
```

```
        short  iless_than, iless_equ ;
EXEC SQL END DECLARE SECTION ;
EXEC SQL INCLUDE SQLCA ;

main ()
{
        int     host_len, i, sockfd, servlen, n, tuples ;
        char    sendline[MAXLINE], lhost_name[TOKEN_LEN] ;
        char    recline[MAXLINE+1], addr[25], name[25], alias[25] ;
        struct hostent *remote_host ;
        struct sockaddr_un serv_addr ;
        TREE_PTR    top ;
        int     n_amount = 2 ;

        printf("This Program will initialized your global data dictionary\n");
        printf("Do you want continue? [y/n]:");
        i = getchar() ;
        if(i != 'y' && i != 'Y')
                exit(0) ;
        printf("\nEnter Oracle's USER NAME: ") ;
        scanf("%s",uid.arr) ;
        strupr(uid.arr) ;
        uid.len = strlen(uid.arr) ;
        strcpy(pwd.arr, getpass("Password :")) ;
        pwd.len = strlen (pwd.arr) ;

        EXEC SQL WHENEVER SQLERROR GOTO error ;
        EXEC SQL CONNECT :uid IDENTIFIED BY :pwd ;
        printf("Connect to ORACLE user: %s \n",uid.arr) ;
        EXEC SQL DELETE FROM G$THU$ ;
        EXEC SQL DELETE FROM G$TA$ ;
        EXEC SQL DELETE FROM G$A$ ;
        EXEC SQL DELETE FROM G$C$ ;
        strcpy(table_name.arr,"G$THU$") ;
        table_name.len = strlen(table_name.arr) ;
        gethostname(host_name.arr, 30 - 1 ) ;
        host_name.len = strlen(host_name.arr) ;
        ins_t_h_u() ;
        ins_t_a_a() ;
        strcpy(table_name.arr,"L$THU$") ;
        table_name.len = strlen(table_name.arr) ;
        ins_t_h_u() ;
        ins_t_a_a() ;
        strcpy(table_name.arr,"G$TA$") ;
        table_name.len = strlen(table_name.arr) ;
        ins_t_h_u() ;
        ins_t_a_a() ;
        strcpy(table_name.arr,"L$TA$") ;
        table_name.len = strlen(table_name.arr) ;
        ins_t_h_u() ;
        ins_t_a_a() ;
        strcpy(table_name.arr,"G$A$") ;
        table_name.len = strlen(table_name.arr) ;
        ins_t_h_u() ;
        ins_t_a_a() ;
        strcpy(table_name.arr,"L$A$") ;
        table_name.len = strlen(table_name.arr) ;
        ins_t_h_u() ;
        ins_t_a_a() ;
        strcpy(table_name.arr,"G$C$") ;
        table_name.len = strlen(table_name.arr) ;
        ins_t_h_u() ;
        ins_t_a_a() ;
        strcpy(table_name.arr,"L$C$") ;
        table_name.len = strlen(table_name.arr) ;
        ins_t_h_u() ;
        ins_t_a_a() ;
        EXEC SQL DECLARE C1 CURSOR FOR
```

```
        SELECT TABLE_NAME,HOST_NAME,USER_NAME
        FROM L$THU$ ;
EXEC SQL OPEN C1 ;
for (;;)
{
        EXEC SQL FETCH C1 INTO :table_name,:host_name,:user_name ;
        if (sqlca.sqlcode == 1403)
                break ;
        insert_THU() ;
}
EXEC SQL CLOSE C1 ;
EXEC SQL DECLARE C2 CURSOR FOR
        SELECT TABLE_NAME,ATTR_NAME,COL_NO
        FROM L$TA$ ;
EXEC SQL OPEN C2 ;
for (;;)
{
        EXEC SQL FETCH C2 INTO :table_name,:attr_name,:colno ;
        if (sqlca.sqlcode == 1403)
                break ;
        insert_TA() ;
}
EXEC SQL CLOSE C2 ;

EXEC SQL DECLARE C3 CURSOR FOR
        SELECT ATTR_NAME,TYPE,WIDTH
        FROM L$A$ ;
EXEC SQL OPEN C3 ;
for (;;)
{
        EXEC SQL FETCH C3 INTO :attr_name,:type,:width ;
        if (sqlca.sqlcode == 1403)
                break ;
        insert_A() ;
}
EXEC SQL CLOSE C3 ;

EXEC SQL DECLARE C4 CURSOR FOR
        SELECT CONST_NAME, ATTR_NAME, MORE_THAN, MORE_EQU, LESS_THAN, LESS_EQU
        FROM L$C$ ;
EXEC SQL OPEN C4 ;
for (;;)
{
        EXEC SQL FETCH C4 INTO :const_name,:attr_name,:more_than :imore_than,
                :more_equ :imore_equ, :less_than :iless_than,:less_equ :iless_equ) ;
        if (sqlca.sqlcode == 1403)
                break ;
        insert_CONS() ;
}
EXEC SQL CLOSE C4 ;
EXEC SQL COMMIT WORK ;
if(gethostname(lhost_name,(TOKEN_LEN-1)) < 0)           /* get local host name */
        err_dump("gethostname error") ;
while ((remote_host = gethostent()) != NULL)
{
        if (strcmp(remote_host->h_name,"localhost") == 0)
                continue ;
        if (strcmp(remote_host->h_name,lhost_name) == 0)
                continue ;
/*-------------------------------------------------------------------------------------------------*/
/* Fill in the structure "serv_addr" with the address of the server that we want to send to. */
/*-------------------------------------------------------------------------------------------------*/

        memset((char *)&serv_addr, 0, sizeof(serv_addr)) ;
        serv_addr.sun_family = AF_UNIX ;
        strcpy(serv_addr.sun_path, UNIXSTR_PATH1) ;
        servlen = strlen(serv_addr.sun_path) + sizeof(serv_addr.sun_family) ;
```

```
if((sockfd = socket(AF_UNIX, SOCK_STREAM, 0)) < 0)
        err_sys("dic_glo: can't open stream socket") ;
if(connect(sockfd, (struct sockaddr *)&serv_addr, servlen) < 0)
        err_sys("dic_glo: can't connect to network server");
strcpy(sendline,remote_host->h_name) ;
n = strlen(sendline) ;
if (writen(sockfd, sendline, n) != n)
        err_dump("dic_glo: writen error") ;
n = send_auth(uid.arr,pwd.arr,sockfd) ;
if (n == -1)
{
        printf("dic_glo: uid and pwd incorrect ") ;
        exit (0) ;
}
else if (n == -2)               /* remote host don't have server */
        continue ;
top = tree_THU() ;
order = 0 ;
order = Walk(top,order) ;
send_query(sockfd,top,n_amount) ;
n = readline(sockfd,recline,MAXLINE) ;
if (n == 0)
        err_quit("dic_glo: connection terminated") ;
if (n < 0)
        err_dump("dic_glo: readline error") ;
if (strncmp(recline,OK,3) == 0)
        memcpy(&tuples,recline+3,sizeof(int)) ;
switch (tuples)
{
        case 0 :
                break ;
        case -1 :
        {
                printf("%s\n",recline+3) ;
                printf("Error at hostname %s\n",remote_host->h_name) ;
                close(sockfd) ;
                EXEC SQL ROLLBACK WORK ;
                continue ;
        }
        default :
        {
                strcpy(sendline,CM);
                strcat(sendline,SEND) ;
                n = strlen(sendline);
                if (writen(sockfd, sendline, n) != n)
                        err_dump("dic_glo: writen error") ;
                receive_data(sockfd,recline) ;
                memcpy(&fld_cnt,recline+3,sizeof(int)) ;
                for (;;)
                {
                        if (receive_data(sockfd,recline) != 0)
                                break ;
                        recv_THU(recline+3) ;
                        insert_THU() ;
                }
                pass_view(sockfd) ;
        }
}
Free_tree(top) ;
top = tree_TA() ;
order = 0 ;
order = Walk(top,order) ;
send_query(sockfd,top,n_amount) ;
n = readline(sockfd, recline, MAXLINE) ;
if ( n == 0)
{
        printf("connection terminated\n") ;
        printf("Error at hostname %s\n",remote_host->h_name) ;
```

```
        EXEC SQL ROLLBACK WORK ;
}
else if (n < 0)
        err_dump("readline error");
if (strncmp(recline, OK, 3) == 0)
        memcpy(&tuples,recline+3,sizeof(int)) ;
else
{
        printf("%s\n",recline+3) ;
        printf("Error at hostname %s\n",remote_host->h_name) ;
        EXEC SQL ROLLBACK WORK ;
}
switch (tuples)
{
        case 0 :
                break ;
        case -1 :
        {
                printf("%s\n",recline+3) ;
                printf("Error at hostname %s\n",remote_host->h_name) ;
                close(sockfd) ;
                EXEC SQL ROLLBACK WORK ;
                continue ;
        }
        default :
        {
                strcpy(sendline,CM);
                strcat(sendline,SEND) ;
                n = strlen(sendline);
                if (writen(sockfd, sendline, n) != n)
                        err_dump("dic_glo: writen error") ;
                receive_data(sockfd,recline) ;
                memcpy(&fld_cnt,recline+3,sizeof(int)) ;
                for (;;)
                {
                        if (receive_data(sockfd,recline) != 0)
                                break ;
                        recv_TA(recline+3) ;
                        insert_TA() ;
                }
                pass_view(sockfd) ;
        }
}
Free_tree(top) ;
top = tree_A() ;
order = 0 ;
order = Walk(top,order) ;
send_query(sockfd,top,n_amount) ;
n = readline(sockfd, recline, MAXLINE) ;
if ( n == 0)
{
        printf("connection terminated\n") ;
        printf("Error at hostname %s\n",remote_host->h_name) ;
        EXEC SQL ROLLBACK WORK ;
}
else if (n < 0)
        err_dump("readline error");
if (strncmp(recline, OK, 3) == 0)
        memcpy(&tuples,recline+3,sizeof(int)) ;
else
{
        printf("%s\n",recline+3) ;
        printf("Error at hostname %s\n",remote_host->h_name) ;
        EXEC SQL ROLLBACK WORK ;
}
switch (tuples)
{
        case 0 :
```

```
                break ;
        case -1 :
        {
                printf("%s\n",recline+3) ;
                printf("Error at hostname %s\n",remote_host->h_name) ;
                close(sockfd) ;
                EXEC SQL ROLLBACK WORK ;
                continue ;
        }
        default :
        {
                strcpy(sendline,CM);
                strcat(sendline,SEND) ;
                n = strlen(sendline);
                if (writen(sockfd, sendline, n) != n)
                        err_dump("dic_glo: writen error") ;
                receive_data(sockfd,recline) ;
                for (;;)
                {
                        if (receive_data(sockfd,recline) != 0)
                                break ;
                        recv_A(recline+3) ;
                        insert_A() ;
                }
                pass_view(sockfd) ;
        }
}
Free_tree(top) ;
top = tree_CONS() ;
order = 0 ;
order = Walk(top,order) ;
send_query(sockfd,top,n_amount) ;
n = readline(sockfd, recline, MAXLINE) ;
if ( n == 0)
{
        printf("connection terminated\n") ;
        printf("Error at hostname %s\n",remote_host->h_name) ;
        EXEC SQL ROLLBACK WORK ;
}
else if (n < 0)
        err_dump("readline error");
if (strncmp(recline, OK, 3) == 0)
        memcpy(&tuples,recline+3,sizeof(int)) ;
else
{
        printf("%s\n",recline+3) ;
        printf("Error at hostname %s\n",remote_host->h_name) ;
        EXEC SQL ROLLBACK WORK ;
}
switch (tuples)
{
        case 0 :
                break ;
        case -1 :
        {
                printf("%s\n",recline+3) ;
                printf("Error at hostname %s\n",remote_host->h_name) ;
                close(sockfd) ;
                EXEC SQL ROLLBACK WORK ;
                continue ;
        }
        default :
        {
                strcpy(sendline,CM);
                strcat(sendline,SEND) ;
                n = strlen(sendline);
                if (writen(sockfd, sendline, n) != n)
                        err_dump("dic_glo: writen error") ;
```

```
                        receive_data(sockfd,recline) ;
                        memcpy(&fld_cnt,recline+3,sizeof(int)) ;
                        for (;;)
                        {
                                if (receive_data(sockfd,recline) != 0)
                                        break ;
                                recv_CONS(recline+3) ;
                                insert_CONS() ;
                        }
                        pass_view(sockfd) ;
                }
        }
        Free_tree(top) ;
        EXEC SQL COMMIT WORK ;
        strcpy(sendline,EXIT) ;
        if (writen(sockfd, sendline, n) != n)
                err_dump("dic_glo: writen error") ;
        close(sockfd) ;
    }
    EXEC SQL COMMIT WORK RELEASE;
    exit(0) ;
error:
    printf("can't initialization local data dictionary \n") ;
    printf("\n%.70s \n",sqlca.sqlerrm.sqlerrmc) ;
    EXEC SQL ROLLBACK WORK RELEASE ;
    exit(-1) ;

}

int ins_t_h_u()
{
    EXEC SQL WHENEVER SQLERROR GOTO error ;
    EXEC SQL INSERT INTO G$THU$
        (TABLE_NAME, HOST_NAME, USER_NAME)
        VALUES(:table_name, :host_name, :uid) ;
    return (0) ;
error:
    EXEC SQL WHENEVER SQLERROR CONTINUE ;
    printf("%.70s \n",sqlca.sqlerrm.sqlerrmc) ;
    EXEC SQL ROLLBACK WORK RELEASE ;
    exit(-1) ;
}
int ins_t_a_a()
{
    colno = 0 ;
    EXEC SQL WHENEVER SQLERROR GOTO error ;
    do
    {
        colno++ ;
        EXEC SQL SELECT CNAME, COLTYPE, WIDTH
            INTO :attr_name, :type, :width
            FROM SYSCOLUMNS WHERE TNAME = :table_name AND COLNO = :colno
            AND CREATOR = :uid ;
        if (sqlca.sqlcode == 0)                 /* found */
        {
            EXEC SQL INSERT INTO G$TA$
                (TABLE_NAME, ATTR_NAME, COL_NO)
                VALUES(:table_name, :attr_name, :colno) ;
            EXEC SQL SELECT WIDTH INTO :nn
                FROM G$A$ WHERE ATTR_NAME =:attr_name ;
            if(sqlca.sqlcode == 1403) /* not exist */
            {
                EXEC SQL INSERT INTO G$A$
                    (ATTR_NAME, TYPE, WIDTH)
                    VALUES(:attr_name, :type, :width) ;
            }
        }
        else if (sqlca.sqlcode == 1403)         /* not found */
```

```
                break ;
        }
        while( 1 ) ;
        return (0) ;
error:
        printf("%.70s \n",sqlca.sqlerrm.sqlerrmc) ;
        EXEC SQL ROLLBACK WORK RELEASE ;
        exit(-1) ;
}

TREE_PTR tree_THU()
{
        TREE_PTR ptr;
        ptr = Make_node(PROJECT,"TABLE_NAME,HOST_NAME,USER_NAME") ;
        ptr->left = Make_node(LEAF,"L$THU$") ;
        return (ptr) ;
}

void recv_THU(str)
char *str;
{
        int     ch_cnt, i ;

        for ( i=0 ; i<fld_cnt ; i++)
        {
                memcpy(&ch_cnt,str,sizeof(int)) ;
                str += sizeof(int) ;
                switch (i)
                {
                        case 0:
                                table_name.len = ch_cnt ;
                                memcpy(table_name.arr,str,ch_cnt) ;
                                break ;
                        case 1:
                                host_name.len = ch_cnt ;
                                memcpy(host_name.arr,str,ch_cnt) ;
                                break ;
                        case 2:
                                user_name.len = ch_cnt ;
                                memcpy(user_name.arr,str,ch_cnt) ;
                                break ;
                }
                str += ch_cnt ;
        }
}

void insert_THU()
{
        EXEC SQL INSERT INTO G$THU$
                (TABLE_NAME,HOST_NAME,USER_NAME)
                VALUES(:table_name,:host_name,:user_name) ;
        return ;
error:
        printf("error in insert_THU()\n");
        printf("\n%.70s \n",sqlca.sqlerrm.sqlerrmc) ;
        EXEC SQL ROLLBACK WORK ;
        exit (-1) ;
}

TREE_PTR tree_TA()
{
        TREE_PTR    ptr;

        ptr = Make_node(PROJECT,"TABLE_NAME,ATTR_NAME,COL_NO") ;
        ptr->left = Make_node(LEAF,"L$TA$") ;
        return (ptr) ;
}
```

```
void recv_TA(str)
char *str ;
{
        int     ch_cnt, i ;
        char    temp[40] ;

        for ( i=0 ; i<fld_cnt ; i++)
        {
                memcpy(&ch_cnt,str,sizeof(int)) ;
                str += sizeof(int) ;
                switch (i)
                {
                        case 0:
                                table_name.len = ch_cnt ;
                                memcpy (table_name.arr,str,ch_cnt) ;
                                break ;
                        case 1:
                                attr_name.len = ch_cnt ;
                                memcpy (attr_name.arr,str,ch_cnt) ;
                                break ;
                        case 2:
                                memcpy (temp,str,ch_cnt) ;
                                temp[ch_cnt] = '\0' ;
                                colno = atoi(temp) ;
                }
                str += ch_cnt ;
        }
}

void insert_TA()
{
        EXEC SQL INSERT INTO G$TA$
                (TABLE_NAME,ATTR_NAME,COL_NO)
                VALUES(:table_name, :attr_name, :colno) ;
        return ;
error:
        printf("error in insert_TA()\n");
        printf("\n%.70s \n",sqlca.sqlerrm.sqlerrmc) ;
        EXEC SQL ROLLBACK WORK ;
        exit (-1) ;
}

TREE_PTR tree_A()
{
        TREE_PTR ptr;
        ptr = Make_node(PROJECT,"ATTR_NAME,TYPE,WIDTH") ;
        ptr->left = Make_node(LEAF,"L$A$") ;
        return (ptr) ;
}

void recv_A(str)
char *str ;
{
        int     ch_cnt, i ;
        char    temp[40] ;

        for ( i=0 ; i<fld_cnt ; i++)
        {
                memcpy(&ch_cnt,str,sizeof(int)) ;
                str += sizeof(int) ;
                switch (i)
                {
                        case 0:
                                attr_name.len = ch_cnt ;
                                memcpy (attr_name.arr,str,ch_cnt) ;
                                break ;
                        case 1:
                                type.len = ch_cnt ;
```

```
                        memcpy (type.arr,str,ch_cnt) ;
                        break ;
                case 2:
                        memcpy (temp,str,ch_cnt) ;
                        temp[ch_cnt] = '\0' ;
                        width = atoi(temp) ;
                        break ;
                }
                str += ch_cnt ;
        }
}

void insert_A()
{
        EXEC SQL SELECT WIDTH INTO :nn
                FROM G$A$
                WHERE ATTR_NAME = :attr_name ;
        if (sqlca.sqlcode == 1403)
        {
                EXEC SQL INSERT INTO G$A$
                        (ATTR_NAME, TYPE, WIDTH)
                        VALUES(:attr_name, :type, :width) ;
        }
        return ;
error:
        printf("error in insert_A()\n");
        printf("\n%.70s \n",sqlca.sqlerrm.sqlerrmc) ;
        EXEC SQL ROLLBACK WORK ;
        exit (-1) ;
}

TREE_PTR tree_CONS()
{
        TREE_PTR    ptr;
        ptr = Make_node(PROJECT,"CONST_NAME,ATTR_NAME,MORE_THAN,MORE_EQU,LESS_THAN,LESS_EQU") ;
        ptr->left = Make_node(LEAF,"L$C$") ;
        return (ptr) ;
}

void recv_CONS(str)
char *str ;
{
        int     ch_cnt, i ;
        char    temp[40] ;

        for ( i=0 ; i<fld_cnt ; i++)
        {
                memcpy(&ch_cnt,str,sizeof(int)) ;
                str += sizeof(int) ;
                switch (i)
                {
                        case 0:
                                const_name.len = ch_cnt ;
                                memcpy (const_name.arr,str,ch_cnt) ;
                                break ;
                        case 1:
                                attr_name.len = ch_cnt ;
                                memcpy (attr_name.arr,str,ch_cnt) ;
                                break ;
                        case 2:
                                if(ch_cnt == 0)
                                        imore_than = -1 ;           /* null val */
                                else
                                {
                                        memcpy (temp,str,ch_cnt) ;
                                        temp[ch_cnt] = '\0' ;
                                        imore_than = 0 ;
                                        more_than = atof(temp) ;
```

```
                        }
                        break ;
                case 3:
                        if(ch_cnt == 0)
                                imore_equ = -1 ;        /* null val */
                        else
                        {
                                memcpy (temp,str,ch_cnt) ;
                                temp[ch_cnt] = '\0' ;
                                imore_equ = 0 ;
                                more_equ = atof(temp) ;
                        }
                        break ;
                case 4:
                        if(ch_cnt == 0)
                                iless_than = -1 ;       /* null val */
                        else
                        {
                                memcpy (temp,str,ch_cnt) ;
                                temp[ch_cnt] = '\0' ;
                                iless_than = 0 ;
                                less_than = atof(temp) ;
                        }
                        break ;
                case 5:
                        if(ch_cnt == 0)
                                iless_equ = -1 ;        /* null val */
                        else
                        {
                                memcpy (temp,str,ch_cnt) ;
                                temp[ch_cnt] = '\0' ;
                                iless_equ = 0 ;
                                less_equ = atof(temp) ;
                        }
                        break ;
                }
                str += ch_cnt ;
        }
}

void insert_CONS()
{
        EXEC SQL INSERT INTO G$C$
                (CONST_NAME, ATTR_NAME, MORE_THAN, MORE_EQU, LESS_THAN, LESS_EQU)
                VALUES(:const_name, :attr_name, :more_than :imore_than, :more_equ :imore_equ,
                        :less_than :iless_than, :less_equ :iless_equ) ;
        return ;
error:
        printf("error in insert_CONS()\n");
        printf("\n%.70s \n",sqlca.sqlerrm.sqlerrmc) ;
        EXEC SQL ROLLBACK WORK ;
        exit (-1) ;
}
```

file dic_loc.mak

```
TOOL=$(HOME)/prog/tool
OBJECT=dic_loc.o
LIBS=.$(ORACLE_HOME)/pro/libsql.a $(ORACLE_HOME)/oci/libhli.a $(ORACLE_HOME)/oci/libhlic.a

dic_loc     : $(OBJECT)
                cc $(CFLAGS) -o dic_loc $(OBJECT) $(LIBS)
dic_loc.c   : dic_loc.pc
                $(TOOL)/precompi dic_loc.pc
dic_loc.o   : dic_loc.c
                cc -c $(CFLAGS) dic_loc.c
```

file_dic_loc.pc

```
/* This program insert data in local data dictionary */
#include        <stdio.h>

EXEC SQL BEGIN DECLARE SECTION ;
        VARCHAR uid[20] ;
        VARCHAR pwd[20] ;
        VARCHAR table_name[30] ;
        VARCHAR host_name[30] ;
        VARCHAR user_name[30] ;
        VARCHAR attr_name[30] ;
        VARCHAR temp[30] ;
        VARCHAR coltype[10] ;
        int     colno ;
        int     width ;
        int     n ;
EXEC SQL END DECLARE SECTION ;
EXEC SQL INCLUDE SQLCA ;

main ()
{
        printf("\nEnter Oracle's USER NAME: ") ;
        scanf("%s",uid.arr) ;
        uid.len = strlen(uid.arr) ;
        strcpy(pwd.arr, getpass("Password :")) ;
        pwd.len = strlen (pwd.arr) ;

        EXEC SQL WHENEVER SQLERROR GOTO error ;
        EXEC SQL CONNECT :uid IDENTIFIED BY :pwd ;
        printf("Connect to ORACLE user: %s \n",uid.arr) ;
        EXEC SQL WHENEVER SQLERROR CONTINUE ;
        printf("Insert table NAME and OWNER\n") ;
        printf("or EXIT, at 'TABLE NAME' promt, to stop program\n") ;
        do
        {
                printf("TABLE NAME : ") ;
                scanf("%s",table_name.arr) ;
                strupr(table_name.arr) ;
                table_name.len = strlen(table_name.arr) ;
                if (strcmp(table_name.arr,"EXIT") == 0)
                        break ;
                printf("TABLE OWNER : ") ;
                scanf("%s",user_name.arr) ;
                strupr(user_name.arr) ;
                user_name.len = strlen(user_name.arr) ;
                if (already_table())
                {
                        printf("table \"%s\" already in local data dictionary\n",table_name.arr) ;
                        continue ;
                }
                if (!exist_table())
                {
                        printf("table \"%s\" does not exist in system\n",table_name.arr) ;
                        continue ;
                }
                ins_l_t_h_u() ;
                ins_l_t_a_l_a() ;
                printf("data added to local data dictionary\n") ;
                EXEC SQL COMMIT WORK ;
        }
        while(1) ;
        EXEC SQL COMMIT WORK RELEASE ;
        exit(0) ;

error:
        printf("%.70s \n",sqlca.sqlerrm.sqlerrmc) ;
```

```
        EXEC SQL ROLLBACK WORK RELEASE ;
        exit(-1) ;
}

int already_table()
{
        EXEC SQL WHENEVER SQLERROR GOTO error ;
        EXEC SQL SELECT HOST_NAME INTO :host_name FROM L$THU$
                WHERE TABLE_NAME = :table_name ;
        if (sqlca.sqlcode == 1403 )
                return (0) ;
        if (sqlca.sqlcode == 0) /* already exist */
                return (1) ;
error:
        printf("%.70s \n",sqlca.sqlerrm.sqlerrmc) ;
        EXEC SQL ROLLBACK WORK RELEASE ;
        exit(-1) ;
}

int exist_table()
{
        EXEC SQL WHENEVER SQLERROR GOTO error ;
        EXEC SQL SELECT GRANTOR INTO :temp FROM SYSTABAUTH
                WHERE TNAME = :table_name AND CREATOR = :user_name ;
        if (sqlca.sqlcode == 0)
                return (1) ;
        if (sqlca.sqlcode == 1403 )
                return (0) ;
error:
        printf("%.70s \n",sqlca.sqlerrm.sqlerrmc) ;
        EXEC SQL ROLLBACK WORK RELEASE ;
        exit(-1) ;
}

int ins_l_t_h_u()
{
        gethostname(host_name.arr, 30 - 1 ) ;
        host_name.len = strlen(host_name.arr) ;
        EXEC SQL WHENEVER SQLERROR GOTO error ;
        EXEC SQL INSERT INTO L$THU$
                (TABLE_NAME, HOST_NAME, USER_NAME)
                VALUES(:table_name, :host_name, :user_name) ;
        return (0) ;
error:
        printf("%.70s \n",sqlca.sqlerrm.sqlerrmc) ;
        EXEC SQL ROLLBACK WORK RELEASE ;
        exit(-1) ;
}

int ins_l_t_a_l_a()
{
        colno = 0 ;
        EXEC SQL WHENEVER SQLERROR GOTO error ;
        do
        {
                colno++ ;
                EXEC SQL SELECT CNAME, COLTYPE, WIDTH
                        INTO :attr_name, :coltype, :width
                        FROM SYSCOLUMNS WHERE TNAME = :table_name AND COLNO = :colno
                        AND CREATOR = :user_name ;
                if (sqlca.sqlcode == 0)                 /* found */
                {
                        EXEC SQL INSERT INTO L$TA$
                                (TABLE_NAME, ATTR_NAME, COL_NO)
                                VALUES(:table_name, :attr_name, :colno) ;
                        EXEC SQL SELECT WIDTH INTO :n
                                FROM L$A$ WHERE ATTR_NAME =:attr_name ;
                        if(sqlca.sqlcode == 1403) /* not exist */
```

```
                {
                    EXEC SQL INSERT INTO L$A$
                        (ATTR_NAME, TYPE, WIDTH)
                        VALUES(:attr_name, :coltype, :width) ;
                }
            }
            else if (sqlca.sqlcode == 1403)          /* not found */
                break ;
        }
        while(1) ;
        return (0) ;
error:
        printf("%.70s \n",sqlca.sqlerrm.sqlerrmc) ;
        EXEC SQL ROLLBACK WORK RELEASE ;
        exit(-1) ;
}

void strupr(str)
char *str;
{
        while (*str != '\0')
        {
                *str = toupper (*str) ;
                str++ ;
        }
        return  ;
}
```

6. ส่วนป้อนกฎบังคับความถูกต้อง

file constr.mak

```
OBJECT=constr.o constryy.o constrlx.o cons_ora.o
LIBS=$(ORACLE_HOME)/pro/libsql.a $(ORACLE_HOME)/oci/libhli.a $(ORACLE_HOME)/oci/libhlic.a
TOOL=$(HOME)/prog/tool

constr                  : $(OBJECT)
                              cc $(CFLAGS) -o constr $(OBJECT) $(LIBS)
cons_ora.o              :cons_ora.c
                              cc -c $(CFLAGS) cons_ora.c
constr.o                :constr.c constr.h
                              cc -c $(CFLAGS) constr.c
constryy.o              :constryy.c
                              cc -c $(CFLAGS) constryy.c
constrlx.o              :constrlx.c constryy.h
                              cc -c $(CFLAGS) constrlx.c
constryy.c constryy.h   :constryy.y $(TOOL)/def.h constr.h
                              yacc -d constryy.y
                              mv y.tab.c constryy.c
                              mv y.tab.h constryy.h
constrlx.c              :constrlx.l constryy.h
                              lex constrlx.l
                              mv lex.yy.c constrlx.c
cons_ora.c              :cons_ora.pc $(TOOL)/def.h constryy.h
                              $(TOOL)/precompi cons_ora.pc
```

file constr.h

```
int         yylex() ;
int         yyparse() ;
int         yylook() ;
int         yywrap() ;
int         login_ora() ;
int         logout_ora() ;
int         del_constr() ;
int         ins_constr() ;
void        ins_op() ;
double      atof() ;
```

file constr.c

```
#include          <stdio.h>
#include          "constr.h"
int     error_ind=0 ;

main()
{
        int     parse_flg = 1 ;
        char    user[20], passwd[20] ;

        printf("\nThis program used to insert or delete local constrain\n") ;
        printf("\nEnter Oracle's USER NAME: ") ;
        scanf("%s",user) ;
        strcpy(passwd, getpass("Password :")) ;
        if(login_ora(user,passwd) != 0)
                exit(-1) ;
        while (parse_flg != 0)
        {
                printf("\nCONSTRAIN: ") ;
```

```
        parse_flg = yyparse() ;
    }
    (logout_ora() == 0)? exit(0) : exit(-1) ;
}
```

file constryy.h

```
# define    CREATE      257
# define    DELETE      258
# define    CONSTR      259
# define    ON          260
# define    ATTR        261
# define    WHERE       262
# define    AND         263
# define    OR          264
# define    CONSTA      265
# define    MORE_EQU    266
# define    MORE        267
# define    LESS_EQU    268
# define    LESS        269
# define    EQU         270
# define    NOTEQU      271
# define    IDENT       272
# define    EX          273
# define    SEMI        274
```

file constryy.y

```
%{
#include        <stdio.h>
#include        "../tool/def.h"
#include        "constr.h"
extern  char    yytext[] ;
char            constr_name[TOKEN_LEN] ;
char            fst_attr[TOKEN_LEN], sec_attr[TOKEN_LEN], thd_attr[TOKEN_LEN] ;
int             opt1, opt2 ;
float           val1, val2 ;
int             thd_flg ;               /* set when found third attr_name */
int             or_flg ;
extern          error_ind ;
%}
%token          CREATE
%token          DELETE
%token          CONSTR
%token          ON
%token          ATTR
%token          WHERE
%token          AND
%token          OR
%token          CONSTA
%token          MORE_EQU
%token          MORE
%token          LESS_EQU
%token          LESS
%token          EQU
%token          NOTEQU
%token          IDENT
%token          EX
%token          SEMI
%start          program
%%
program     /* empty */
            :create
                {
                    ins_constr (constr_name,fst_attr,sec_attr,thd_attr,opt1,val1,opt2,val2) ;
```

```
                        return (1) ;
                    }
                |delete
                    {
                        del_constr (constr_name) ;
                        return (1) ;
                    }
                |EX SEMI
                    {
                        return(0) ;
                    }
                |error
                    {
                        throw_to_semi() ;
                        yyerrok ;
                        yyclearin ;
                        return (-1) ;
                    }
                ;
create          :sentence predicate SEMI
                ;
sentence            :CREATE
                    {
                        opt2 = 0 ;
                        thd_flg = 0 ;
                        or_flg = 0 ;
                    }
                    CONSTR IDENT
                    {
                        strcpy(constr_name,yytext) ;
                    }
                    ON ATTR IDENT
                    {
                        strcpy(fst_attr,yytext) ;
                    }
                    WHERE
                ;
predicate           :compare
                |compare conjunction
                    {
                        thd_flg = 1 ;
                    }
                    compare
                ;
conjunction         :AND
                    {
                        if (or_flg)
                        {
                            yyerror(yytext) ;
                            throw_to_semi() ;
                            yyerrok ;
                            yyclearin ;
                            return (-1) ;
                        }
                    }
                |OR
                    {
                        if (!or_flg)
                        {
                            yyerror(yytext) ;
                            throw_to_semi() ;
                            yyerrok ;
                            yyclearin ;
                            return (-1) ;
                        }
                    }
                ;
compare             :IDENT
```

```
        {
                if (thd_flg == 1 )
                        strcpy(thd_attr,yytext) ;
                else
                        strcpy(sec_attr,yytext) ;
        }
        operation CONSTA
        {
                if (thd_flg == 1)
                {
                        val2 = atof(yytext) ;
                }
                else
                {
                        val1 = atof(yytext) ;
                }
        }
        ;
operation       :MORE_EQU
                {
                        if (thd_flg == 1)
                                opt2 = MORE_EQU ;
                        else
                        {
                                opt1 = MORE_EQU ;
                                opt2 = 0 ;
                        }
                }
        IMORE
                {
                        if (thd_flg == 1)
                                opt2 = MORE ;
                        else
                        {
                                opt1 = MORE ;
                                opt2 = 0 ;
                        }
                }
        ILESS_EQU
                {
                        if (thd_flg == 1)
                                opt2 = LESS_EQU ;
                        else
                        {
                                or_flg = 1 ;
                                opt1 = LESS_EQU ;
                                opt2 = 0 ;
                        }
                }
        ILESS
                {
                        if (thd_flg == 1)
                                opt2 = LESS ;
                        else
                        {
                                or_flg = 1 ;
                                opt1 = LESS ;
                                opt2 = 0 ;
                        }
                }
        IEQU
                {
                        opt2 = 0 ;
                        opt1 = EQU ;
                }
        INOTEQU
                {
                        opt2 = 0 ;
```

```
                        opt1 = NOTEQU ;
                    }
                ;
delete          :DELETE CONSTR IDENT
                    {
                        strcpy(constr_name,yytext) ;
                    }
                    SEMI
                ;
%%
throw_to_semi()
{
        yylex() ;
}
yyerror(str)
char *str ;
{
    printf("\nThere is a syntax error on \"%s\"",yytext) ;
    error_ind = 1 ;
}
```

file constrlx.l

```
%{
#include        <stdio.h>
#include        <ctype.h>
#include        "constryy.h"
%}
R               [-\+]?[0-9]+\.[0-9]*
I               [-\+]?[0-9]+
NAME            [a-zA-Z][a-zA-Z0-9_#\$]*
%%
[\n]                                    ;
[Cc][Rr][Ee][Aa][Tt][Ee]                return (CREATE) ;
[Dd][Ee][Ll][Ee][Tt][Ee]                return (DELETE) ;
[Cc][Oo][Nn][Ss][Tt][Rr][Aa][Ii][Nn][Tt]    return (CONSTR) ;
[Oo][Nn]                                return (ON) ;
[Aa][Tt][Tt][Rr][Ii][Bb][Uu][Tt][Ee]    return (ATTR) ;
[Ww][Hh][Ee][Rr][Ee]                    return (WHERE) ;
[Aa][Nn][Dd]                            return (AND) ;
[Oo][Rr]                                return (OR) ;
[Ee][Xx][Ii][Tt]                        return (EX) ;
{R}                                     return (CONSTA) ;
{I}                                     return (CONSTA) ;
{NAME}                                  {       /*constr. or attr. name*/
                                            strupr(yytext) ;
                                            return (IDENT) ;
                                        }
\>=                                     return (MORE_EQU) ;
\>                                      return (MORE) ;
\<=                                     return (LESS_EQU) ;
\<                                      return (LESS) ;
=                                       return (EQU) ;
\^=                                     return (NOTEQU) ;
;                                       return (SEMI) ;
%%
yywrap ()
{
    return (1) ;
}
void strupr(str)
char *str ;
{
    while (*str != '\0')
    {
        *str = toupper (*str) ;
        str++ ;
```

```
        }
        return ;
}
```

file cons_ora.pc

```
#include        <stdio.h>
#include        "../tool/def.h"
#include        "constryy.h"

EXEC SQL BEGIN DECLARE SECTION ;
        VARCHAR   uid[20] ;
        VARCHAR   pwd[20] ;
        VARCHAR   const_name[30] ;
        VARCHAR   att_name[30] ;
        float   more_than, more_equ, less_than, less_equ ;
        short   more_thani, more_equi, less_thani, less_equi ;
EXEC SQL END DECLARE SECTION ;
EXEC SQL INCLUDE SQLCA;

int  login_ora(user,passwd)
char   user[20], passwd[20] ;
{
        strcpy(uid.arr,user) ;
        uid.len = strlen(user) ;
        strcpy(pwd.arr,passwd) ;
        pwd.len = strlen(passwd) ;
        EXEC SQL WHENEVER SQLERROR GOTO login_error ;
        EXEC SQL CONNECT :uid IDENTIFIED BY :pwd ;
        printf("Connected to ORACLE user: %s\n",uid.arr) ;
        EXEC SQL WHENEVER SQLERROR CONTINUE ;
        return(0) ;

login_error:
        printf("\n%.70s",sqlca.sqlerrm.sqlerrmc) ;
        printf("\nCANNOT LOGIN TO ORACLE") ;
        printf("\nCHECK YOUR USER NAME AND PASSWORD AGAIN\n") ;
        EXEC SQL ROLLBACK WORK RELEASE ;
        return(-1) ;
}

int  logout_ora()
{
        EXEC SQL WHENEVER SQLERROR GOTO logout_error ;
        EXEC SQL COMMIT WORK RELEASE ;
        EXEC SQL WHENEVER SQLERROR CONTINUE ;
        return(0) ;

logout_error:
        printf("\n %.70s \n",sqlca.sqlerrm.sqlerrmc) ;
        EXEC SQL ROLLBACK WORK RELEASE ;
        return(-1) ;
}

int ins_constr(constr_name,fst_name,sec_name,thd_name,opt1,val1,opt2,val2)
char   constr_name[TOKEN_LEN] ;
char   fst_name[TOKEN_LEN], sec_name[TOKEN_LEN], thd_name[TOKEN_LEN] ;
int    opt1, opt2 ;
float  val1, val2 ;
{
        strcpy(att_name.arr,fst_name) ;
        att_name.len = strlen(fst_name) ;
        strcpy(const_name.arr,constr_name) ;
        const_name.len = strlen(constr_name) ;
        EXEC SQL WHENEVER SQLERROR GOTO ins_error ;
        if (strcmp(fst_name,sec_name) != 0)
        {
```

```
        printf("\nATTRIBUTE NAMEs in query are not identify\n") ;
        return(-1) ;
    }
    if (opt2 != 0)
    {
        if ((val1 > val2 ) || (opt2 == EQU) || (opt2 == NOTEQU) || (strcmp(sec_name,thd_name) != 0))
        {
            printf("\nInvalit predicate\n") ;
            return(-1) ;
        }
        switch (opt1)
        {
            case MORE :
                if (!(opt2 == LESS) && !(opt2 == LESS_EQU))
                {
                    printf("\nInvalit predicate\n") ;
                    return(-1) ;
                }
                more_than = val1 ;
                more_thani = 0 ;
                switch (opt2)
                {
                    case LESS :
                        less_than = val2 ;
                        less_thani = 0 ;
                        more_equi = less_equi = -1 ;
                        ins_op() ;
                        break ;
                    case LESS_EQU :
                        less_equ = val2 ;
                        less_equi = 0 ;
                        more_equi = less_thani = -1 ;
                        ins_op() ;
                        break ;
                }
                break ;
            case MORE_EQU :
                if (!(opt2 == LESS) && !(opt2 == LESS_EQU))
                {
                    printf("\nInvalit predicate\n") ;
                    return(-1) ;
                }
                more_equ = val1 ;
                more_equi = 0 ;
                switch (opt2)
                {
                    case LESS :
                        less_than = val2 ;
                        less_thani = 0 ;
                        more_thani = less_equi = -1 ;
                        ins_op() ;
                        break ;
                    case LESS_EQU :
                        less_equ = val2 ;
                        less_equi = 0 ;
                        more_thani = less_thani = -1 ;
                        ins_op() ;
                        break ;
                }
                break ;
            case LESS :
                if (!(opt2 == MORE) && !(opt2 == MORE_EQU))
                {
                    printf("\nInvalit predicate\n") ;
                    return(-1) ;
                }
                less_than = val1 ;
                less_thani = 0 ;
```

```
                switch (opt2)
                {
                        case MORE :
                                more_than = val2 ;
                                more_thani = 0 ;
                                more_equi = less_equi = -1 ;
                                ins_op() ;
                                break ;
                        case MORE_EQU :
                                more_equ = val2 ;
                                more_equi = 0 ;
                                more_thani = less_equi = -1 ;
                                ins_op() ;
                                break ;
                }
                break ;
        case LESS_EQU :
                if (!(opt2 == MORE) && !(opt2 == MORE_EQU))
                {
                        printf("\nInvalit predicate\n") ;
                        return(-1) ;
                }
                less_equ = val1 ;
                less_equi = 0 ;
                switch (opt2)
                {
                        case MORE :
                                more_than = val2 ;
                                more_thani = 0 ;
                                more_equi = less_thani = -1 ;
                                ins_op() ;
                                break ;
                        case MORE_EQU :
                                more_equ = val2 ;
                                more_equi = 0 ;
                                more_thani = less_thani = -1 ;
                                ins_op() ;
                                break ;
                }
                break ;
    }
}
else
{
    switch (opt1)
    {
        case MORE :
                more_than = val1 ;
                more_thani = 0 ;
                more_equi = less_thani = less_equi = -1 ;
                ins_op() ;
                break ;
        case MORE_EQU :
                more_equ = val1 ;
                more_equi = 0 ;
                more_thani = less_thani = less_equi = -1 ;
                ins_op() ;
                break ;
        case LESS :
                less_than = val1 ;
                less_thani = 0 ;
                more_equi = more_thani = less_equi = -1 ;
                ins_op() ;
                break ;
        case LESS_EQU :
                less_equ = val1 ;
                less_equi = 0 ;
                more_equi = more_thani = less_thani = -1 ;
```

```
                        ins_op() ;
                        break ;
                case EQU :
                        more_equ = less_equ = val1 ;
                        more_equi = less_equi = 0 ;
                        more_thani = less_thani = -1 ;
                        ins_op() ;
                        break ;
                case NOTEQU :
                        more_than = less_than = val1 ;
                        more_thani = less_thani = 0 ;
                        more_equi = less_equi = -1 ;
                        ins_op() ;
                        break ;
            }
        }
        EXEC SQL WHENEVER SQLERROR CONTINUE ;
        return (0) ;
ins_error:
        printf("\n %.70s \n",sqlca.sqlerrm.sqlerrmc) ;
        EXEC SQL ROLLBACK WORK ;
        return(-1) ;
}

void ins_op()
{
        EXEC SQL INSERT INTO L$C$
                (const_name,attr_name,more_than,more_equ,less_than,less_equ)
                VALUES(:const_name, :att_name, :more_than :more_thani,
                :more_equ :more_equi, :less_than :less_thani, :less_equ :less_equi) ;
        EXEC SQL COMMIT WORK ;
        printf("Constrain %s inserted\n",const_name.arr) ;
}

int  del_constr(name)
char    name[TOKEN_LEN];
{
        strcpy(const_name.arr,name) ;
        const_name.len = strlen(name) ;
        EXEC SQL WHENEVER SQLERROR GOTO del_error ;
        EXEC SQL SELECT ATTR_NAME INTO :att_name FROM L$C$
                WHERE CONST_NAME = :const_name ;
        if (sqlca.sqlcode == 1403 )
        {
                printf("\nConstrain \"%s\" not found\n",name) ;
                return (-1) ;
        }
        EXEC SQL DELETE FROM L$C$ WHERE CONST_NAME = :const_name ;
        EXEC SQL COMMIT WORK ;
        EXEC SQL WHENEVER SQLERROR CONTINUE ;
        printf("\nconstrain %s deleted\n",name) ;
        return(0) ;

del_error:
        printf("\n %.70s \n",sqlca.sqlerrm.sqlerrmc) ;
        EXEC SQL ROLLBACK WORK ;
        return(-1) ;
}
```

7. ส่วนเครื่องมือโปรแกรม

file def.h

```
/*
 * Definitions for UNIX domain stream and datagram client/server programs.
 */

#include    <errno.h>
#include    <string.h>
#include    <sys/time.h>
#include    <memory.h>
#include    <math.h>
#include    <fcntl.h>
#include    <stdio.h>
#include    <string.h>
#include    <sys/types.h>
#include    <sys/socket.h>
#include    <sys/un.h>
#ifdef      KMITL30
#define     UNIXSTR_PATH0   /u3/rsch/chck/s.unixstr0    /* what ever with in machine that want
                                                           connect to back-end */
#define     UNIXSTR_PATH1   /u3/rsch/chck/s.unixstr1    /* what ever with in machine that want
                                                           connect to net-interface */
#endif
#ifdef      KMITL5
#define     UNIXSTR_PATH0   /u3/grad/chck/s.unixstr0    /* what ever with in machine that want
                                                           connect to back-end */
#define     UNIXSTR_PATH1   /u3/grad/chck/s.unixstr1    /* what ever with in machine that want
                                                           connect to net-interface */
#endif
#define     UNIXDG_TMP      /tmp/dg.XXXXXX"
#define     SERV_TCP_PORT   6000

#include    <netinet/in.h>
#include    <arpa/inet.h>
#include    <netdb.h>

/* ------------------------------------------------------ */
/*          Constant Definition for All Routines          */
/* ------------------------------------------------------ */
#define     TRUE        1           /* true value */
#define     FALSE       0           /* false value */
#define     AGB_LEN     200         /* max length of algebra query input */
#define     LINE_LEN    80          /* max character for one line */
#define     MAX_TOKEN   100         /* max number of token in a query */
#define     MAX_STR     200         /* length of public string use in convert process */
#define     TOKEN_LEN   30          /* max length of a seperate word from query */
#define     MAXOP       6           /* six operator in infix-expr */
#define     MAXCOMP     6           /* six compare operator in select-pred */
#define     BLANK       " "         /* blank character */
#define     SIGN        "+-"        /* sign of numberic */
#define     OPERATOR    ":<>=^"             /* composits of operators */
#define     DELIMITER   ".(){}[]',;"        /* delimiter for seperating tokens */
#define     DIGITS      "0123456789."       /* numeric character */
#define     LETTERS     "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ_#$"
#define     NUM_VIEW    15          /* number of view */
#define     MAXLINE     512         /* determine number of char. in each send or recv. */

/* ------------------------------------------------------ */
/*              define protocall                          */
/* ------------------------------------------------------ */
#define     EN      "$00"       /* end of send */
#define     QU      "$01"       /* query */
#define     DA      "$02"       /* data */
```

```
#define     OK    "$03"       /* ok receive */
#define     ER    "$04"       /* error */
#define     US    "$05"       /* user id */
#define     PW    "$06"       /* password */
#define     CM    "$07"       /* command */
#define     EXIT  "$08"       /* send when finish */
#define     VIEW  "$09"       /* ind. view desc. data */

typedef struct node {
            struct node  *left ; /* addr. of left node */
            struct node  *right ;      /* addr. of right node */
            char         order ;/* order of node in tree */
            char         type[4] ;     /* present operation */
            char         item[AGB_LEN] ;
            } TREE_NODE, *TREE_PTR ;

typedef struct name {
            struct name *prev ;
            struct name *next ;
            char    string[30] ;
            } NAME ;

typedef struct  adj {
            struct adj  *parent ;
            struct adj  *left ;
            struct adj  *right ;
            NAME        *item ;
            char        type[4] ;
            NAME        *attr_list ;
            } ADJ ;

typedef struct  {
            char        current;
            char        left;
            char        right;
            char        type[4];
            char        item[AGB_LEN];
            } TREE_LINK;

typedef struct  {
            char    view_name[30] ;
            char    new_attr[30] ;
            char    old_tab[30] ;
            char    old_attr[30] ;
            int     order ;
            }desc_view ;

/* ------------------------------------------------------ */
/*                    operation                           */
/* ------------------------------------------------------ */
#define     LEAF        "#00"
#define     UNION       "#01"  /*                         */
#define     MINUS       "#02"  /*                         */
#define     INTERSECT   "#03"  /* also use follow command */
#define     TIMES       "#04"  /*                         */
#define     JOIN        "#05"  /*                         */
#define     DIVIDEBY    "#06"  /*                         */
#define     SELECT      "#07"
#define     PROJECT     "#08"

/* ------------------------------------------------------ */
/*              code follow command                       */
/* ------------------------------------------------------ */
#define     SEND        "@00"
```

file proto.h

```
/* in file tool.c */
int             readn();
int             writen();
int             readline();
void            Free_tree() ;
void            Free_node() ;
TREE_PTR        Tree_alloc() ;
TREE_PTR        Make_node() ;
unsigned char   Walk() ;
void            send_query() ;
void            send_tree() ;
int             receive_data() ;
void            itoa() ;
void            reverse() ;
int             send_auth() ;
TREE_PTR        receive_query() ;
TREE_LINK       *Make_tree() ;
void            Lex_Attr() ; /* convert tab_attr to attr */
void            strPbnk() ;
void            pass_view() ;
void            strupr() ;
char            *LexWord() ;
char            *LexSel() ;

/* in file error.c */

void        err_quit();
void        err_sys();
void        err_ret();
void        err_dump();
void        my_perror();
char        *sys_err_str();
char        *host_err_str();
```

file tool.c

```
#ifndef     TOOL
#define     TOOL
#include    "def.h"
#include    "proto.h"
/*
 * Read "n" bytes from a descriptor.
 * Use in place of read() when fd is a stream socket.
 */
int     readn(fd, ptr, nbytes)
register int    fd;
register char   *ptr;
register int    nbytes;
{
        int     nleft, nread;

        nleft = nbytes;
        while(nleft > 0)
        {
                nread = read(fd, ptr, nleft);
                if( nread < 0)
                        return(nread);
                else if(nread == 0)
                        break;
                nleft   -= nread;
                ptr     += nread;
        }
        return(nbytes - nleft);
}
```

```
/*
 * Write "n" bytes to a descriptor.
 * Use in place of write() when fd is a stream socket.
 */

int writen( fd, ptr, nbytes)
register int            fd;
register char *ptr;
register int    nbytes;
{
        int     nleft, nwritten;
        char    buff[5],blank ;
        int     n, i ;

        blank = ' ' ;
        itoa(nbytes,buff) ;
        n = strlen(buff) ;
        for (i=0;i<4;i++)
        {
                if (i < n)
                        write(fd,&buff[i],1) ;
                else
                        write(fd,&blank,1) ;
        }
        nleft = nbytes ;
        while(nleft > 0)
        {
                nwritten = write(fd, ptr, nleft);
                if (nwritten <= 0)
                        return(nwritten);
                nleft   -= nwritten;
                ptr     += nwritten;
        }
        return(nbytes - nleft);
}

/*
 * Read a line from a descriptor. Read the line one byte at a time,
 * looking for the newline. We store the newline in the buffer,
 * then follow it with a null ( the same sa fgets(3)).
 * We return the number of characters up to, but not including,
 * the null (the same as strlen(3)).
 */


int readline(fd, ptr, maxlen)
register int    fd;
register char *ptr;
register int    maxlen;
{
        int     n, rc, total ;
        char    c ;
        char    buff[5] ;

        n = readn(fd,buff,4) ;
        if (n != 4)
        {
                if (n == 0)
                        return(0) ;
                else if (n < 0)
                        return(-1) ;
        }
        buff[5] = '\0' ;
        total = atoi(buff) ;
        for(n = 1; n < maxlen; n++)
        {
                if( (rc = read(fd, &c, 1)) == 1)
```

```
            {
                *ptr++ = c ;
                if (n == total)
                    break;
            }
            else if (rc == 0)
            {
                if (n == 1)             /* EOF, no data read */
                    return (0);
                else
                    break;              /* EOF, some data was read */
            }
            else
                return (-1);            /* error */
        }
        *ptr = 0;
        return (n);
}

/*
 * use to free ALL node of tree
 * input: pointer of top of the tree. If input = NULL ignor (don't do anything)
 */
void Free_tree(top_tree)
TREE_PTR top_tree ;
{
        if(top_tree != NULL)
        {
                if(top_tree->left != NULL)
                        Free_tree(top_tree->left) ;
                if(top_tree->right != NULL)
                        Free_tree(top_tree->right) ;
                Free_node(top_tree) ;
        }
}

void Free_node(node_ptr)
TREE_PTR node_ptr ;
{
        free(node_ptr) ;
}

TREE_PTR Tree_alloc()
{
        return( (TREE_PTR) malloc(sizeof(TREE_NODE)) ) ;
}

TREE_PTR Make_node(type, data)
char type[4];
char data[AGB_LEN] ;
{
        TREE_PTR p ;
        p = Tree_alloc() ;
        p->left  = NULL ;
        p->right = NULL ;
        strcpy(p->type,type) ;
        if(strcmp(data,NULL) == 0)
                strcpy(p->item," ") ;
        else
                strcpy(p->item,data) ;
        return(p) ;
}

int Walk(ptr,init_order)
TREE_PTR ptr;
int init_order;
{
        ptr->order = (char)++init_order;
```

```
        if (ptr->left != NULL)
                init_order = Walk (ptr->left,init_order);
        if (ptr->right != NULL)
                init_order = Walk (ptr->right,init_order);
        return(init_order);
}

void send_query(sockfd,ptr,n_amount)
int sockfd;
TREE_PTR ptr;
int n_amount;
{
        char line[MAXLINE],s[3];
        int n;

        strcpy(line,QU);
        itoa(n_amount,s);
        strcat(line,s);
        n = strlen(line);
        if(writen(sockfd, line, n) != n)
                err_dump("send_query: writen error") ;
        send_tree(sockfd, ptr);
        strcpy(line,EN);
        n = strlen(line);
        if(writen(sockfd, line, n) != n)
                err_dump("send_query: writen error") ;
}

void send_tree(sockfd,ptr)
int sockfd;
TREE_PTR ptr;
{
        TREE_LINK data;
        int n;
        char *Hlink ;

        Hlink = malloc(sizeof(TREE_LINK)+ 3) ;
        data.current = ptr->order;
        if(ptr->left != NULL)
                data.left = ptr->left->order;
        else
                data.left = (char) 0;
        if(ptr->right != NULL)
                data.right = ptr->right->order;
        else
                data.right = (char) 0;
        strcpy(data.type,ptr->type);
        strcpy(data.item,ptr->item);
        memcpy(Hlink,DA,3) ;
        n = sizeof(data);
        memcpy(Hlink+3,&data,n) ;
        n += 3 ;
        if (writen(sockfd,Hlink,n) != n )
                err_sys("send_query: writen error on socket");
        free (Hlink) ;
        if (ptr->left != NULL)
                send_tree(sockfd,ptr->left) ;
        if (ptr->right != NULL)
                send_tree(sockfd,ptr->right);
        return;
}
int *receive_data(sockfd,line)
int sockfd;
char *line ;
{
        int  n;

        strcpy(line,OK);
```

```
        n = strlen(line);
        if(writen(sockfd, line, n) != n)
                err_dump("receive_data: writen error") ;
        n = readline(sockfd, line, MAXLINE);
        if( n == 0)
                err_quit("receive_data: connection terminated");
        if(n <0)
                err_sys("receive_data: read error on socket");
        line[n] = '\0';
        if(strncmp(line,DA,3) == 0)
                return(0);
        else if(strncmp(line,VIEW,3) == 0)
                return(1);
        else if(strncmp(line,ER,3) == 0)
                return(-1);
        else
                err_quit("receive_data: protocol error");
}

void pass_view(sockfd)
int sockfd ;
{
        char sendline[MAXLINE], recline[MAXLINE+1] ;
        int m, n ;

        strcpy(sendline,OK) ;
        n = strlen(sendline) ;
        do
        {
                if (writen(sockfd,sendline,n) != n)
                        err_dump("pass_view: writen error ") ;
                m = readline(sockfd, recline, MAXLINE);
                if ( m == 0)
                        err_quit("pass_view: connection terminated");
                if (m <0)
                        err_sys("receive_data: read error on socket");
                if (strncmp(recline,VIEW,3) == 0)
                        continue ;
                else if (strncmp(recline,EN,3) == 0)
                {
                        if (writen(sockfd,sendline,n) != n)
                                err_dump("pass_view: writen error ") ;
                        break ;
                }
                else
                        err_quit("pass_view: protocal error") ;
        }while(1) ;
}

/* convert n to characters in s */
void itoa(n, s)
int n;
char s[];
{
        int i, sign;

        if ((sign = n) < 0)  /* record sign */
                n = -n;
        i = 0;
        do{     /* generate digits in reverse order */
                s[i++] = n % 10 + '0';
        }while((n /= 10) > 0);
        if(sign < 0)
                s[i++] = '-';
        s[i] = '\0';
        reverse(s);
}
```

```
/* reverse string s in place */
void reverse(s)
char s[];
{
        int c, i, j;

        for( i = 0, j = strlen(s)-1; i <j; i++, j--)
        {
                c = s[i];
                s[i] = s[j];
                s[j] = c;
        }
}

/*  send authority  */
int send_auth(uid,pwd,sockfd)
char *uid,*pwd ;
int sockfd ;
{
        char    sendline[MAXLINE], recline[MAXLINE+1] ;
        int     n ;

        strcpy(sendline,US) ;
        strcat(sendline,uid) ;
        n = strlen(sendline) ;
        if(writen(sockfd, sendline, n) != n)
                err_dump("writen error") ;
        strcpy(sendline,PW) ;
        strcat(sendline,pwd) ;
        n = strlen(sendline) ;
        if(writen(sockfd, sendline, n) != n)
                err_dump("writen error") ;
        n = readline(sockfd, recline, MAXLINE) ;
        if ( n == 0)
        {
                strcpy("send_auth: connection terminated") ;
                return (-2) ;
        }
        else if (n < 0)
                err_dump("send_auth: readline error");
        if (strncmp(recline, OK, 3) == 0)
                return (1) ;
        else
        {
                printf("%s\n",recline+3) ;
                return (-1) ;
        }
}

/*   receive query then make tree. If success return TREE_PTR     */
/* else return NULL                                               */
/* :sockfd = socket descriptor                                    */
/* :number = number of query nodes                                */

TREE_PTR receive_query(sockfd,number)
int sockfd, number ;
{
        int i, n ;
        char recline[MAXLINE+1] ;
        TREE_LINK *start, *cur_ptr ;
        TREE_PTR top ;

        if((start = (TREE_LINK *)malloc(number*(sizeof(TREE_LINK)))) == NULL)
                err_dump("receive_query: malloc error") ;
        cur_ptr = start ;
        for (i=0;i<number;i++)
        {
                n = readline(sockfd, recline, MAXLINE) ;
```

```
            n = readline(sockfd, recline, MAXLINE) ;
            if ( n == 0)
            {
                printf("receive_query: connection terminated") ;
                exit(-1) ;
            }
            else if (n < 0)
                err_dump("receive: readline error");
            recline[n] = '\0';
            if (strncmp(recline, DA, 3) == 0)
            {
                cur_ptr->current = recline[3] ;
                cur_ptr->left    = recline[4] ;
                cur_ptr->right   = recline[5] ;
                strcpy(cur_ptr->type,recline+6) ;
                strcpy(cur_ptr->item,recline+10) ;
                cur_ptr++ ;
            }
            else
            {
                err_dump("receive_query:protocal error");
                exit(-1);
            }
        }
        n = readline(sockfd, recline, MAXLINE) ;
        if ( n == 0)
        {
            printf("receive_query: connection terminated") ;
            exit(-1) ;
        }
        else if (n < 0)
            err_dump("receive_query: readline error");
        recline[n] = '\0';
        if (strncmp(recline, EN, 3) != 0)
        {
            free(start) ;
            return(NULL) ;
        }
        top = Tree_alloc() ;
        Make_tree(top,start) ;
        free(start) ;
        return(top) ;
}

TREE_LINK *Make_tree(top, start)
TREE_PTR top;
TREE_LINK *start;
{
    strcpy(top->item,start->item);
    strcpy(top->type, start->type);
    top->order = start->current;
    if (start->left != (char)0)
        top->left = Tree_alloc() ;
    else
        top->left = NULL ;
    if (start->right != (char)0)
        top->right = Tree_alloc() ;
    else
        top->right = NULL ;
    if (top->left != NULL)
    {
        start++ ;
        start = Make_tree(top->left,start) ;
    }
    if (top->right != NULL)
    {
        start++ ;
        start = Make_tree(top->right,start) ;
```

```
        }
        return(start) ;
}

void Lex__Attr(str1,str2)
char *str1, *str2 ;
{
        while((*str2 != '_') || (*(str2+1) != '_'))
                str2++ ;
        str2++ ;
        str2++ ;
        while((*str2 != ' ') && (*str2 != '\0'))
                *str1++ = *str2++ ;
        *str1 = '\0' ;
}

void strPbnk(str1,str2,len)
char *str1, *str2 ;
int len ;
{
        int i, n ;
        n = strlen(str2) ;
        for (i=0;i<n;i++)
                *str1++ = *str2++ ;
        for (i=1;i<(len-n);i++)
                *str1++ = ' ' ;
        *str1 = '\0' ;
}
void  strupr(str)
char  *str;
{
        while (*str != '\0')
        {
                *str = toupper (*str) ;
                str++ ;
        }
        return  ;
}

char *LexWord(word,words)
char *word, *words ;
{
        while(*words == ' ')
                words++ ;
        while((*words != ' ') && (*words != '\0'))
        {
                *word++ = *words++ ;
        }
        *word = '\0' ;
        while(*words == ' ')
                words++ ;
        if (*words != '\0')
                return (words) ;
        else
                return (NULL) ;
}

char *LexSel(str1,str2)
char *str1, *str2 ;
{
        while ((*str2 == ' ') || (*str2 == '\t'))
                str2++ ;
        if (*str2 == '\'')
        {
                do
                {
                        *str1++ = *str2++ ;
                }while (*str2 != '\'') ;
```

```
                *str1++ = *str2++ ;
                *str1 = '\0' ;
        }
        else if ((*str2 == '(') || (*str2 == ')'))
        {
                *str1++ = *str2++ ;
                *str1 = '\0' ;
        }
        else
        {
                while ((*str2 != ' ') && (*str2 != '\0') && (*str2 != '(') && (*str2 != ')'))
                        *str1++ = *str2++ ;
                *str1 = '\0' ;
        }
        while ((*str2 == ' ') || (*str2 == '\t'))
                str2++ ;
        if (*str2 == '\0')
                return (NULL) ;
        else
                return (str2) ;
}
```

file_error.c

```
#include        <netdb.h>
#include        <stdio.h>
#include        <varargs.h>
#include        <errno.h>

char    *pname = NULL;
#ifndef         NULL
#define         NULL    ((void *) 0)
#endif

/*
 * Fatal error. Print a message and terminate.
 * Don't dump core and don't print the system's errno value.
 *
 *      err_quit(str, arg1, arg2, ...)
 *
 * The string "str" must specify the conversion specification for any args.
 */

/* VARARGS1 */
err_quit(va_alist)
va_dcl
{
        va_list         args;
        char            *fmt;

        va_start(args);
        if ( pname != NULL)
                fprintf(stderr, "%s: "; pname);
        fmt = va_arg(args, char *);
        vfprintf(stderr, fmt, args);
        fputc('\n', stderr);
        va_end(args);
        exit(-1);
}
/*
 * Fatal error related to a system call. Print a message and terminate.
 * Don't dump core, but do print the system's errno value and its
 * associated message.
 *
 *      err_sys(str, arg1, arg2, ...)
 *
 * The string "str" must specify the conversion specification for any args.
```

```
 */

/* VARARGS1 */
err_sys(va_alist)
va_dcl
{
        va_list         args;
        char            *fmt;

        va_start(args);
        if (pname != NULL)
                fprintf(stderr, "%s: ", pname);
        fmt = va_arg(args, char *);
        vfprintf(stderr, fmt, args);
        va_end(args);
        my_perror();
        exit(-1);
}

/*
 * Recoverable error. Print a message, and return to caller.
 *
 *      err_ret(str, arg1, arg2, ...)
 *
 * The string "str" must specify the conversion specification for any args.
 */

/* VARARGS1 */
err_ret(va_alist)
va_dcl
{
        va_list         args;
        char            *fmt;

        va_start(args);
        if (pname != NULL)
                fprintf(stderr, "%s: ", pname);
        fmt = va_arg(args, char *);
        vfprintf(stderr, fmt, args);
        va_end(args);
        my_perror();
        fflush(stdout);
        fflush(stderr);
        return;
}

/*
 * Fatal error. Print a message, dump core (for debugging) and terminate.
 *
 *      err_dump(str, arg1, arg2, ...)
 *
 * The string "str" must specify the conversion specification for any args.
 */

/* VARARGS1 */
err_dump(va_alist)
va_dcl
{
        va_list         args;
        char            *fmt;

        va_start(args);
        if (pname != NULL)
                fprintf(stderr, "%s: ", pname);
        fmt = va_arg(args, char *);
        vfprintf(stderr, fmt, args);
        va_end(args);
        my_perror();
```

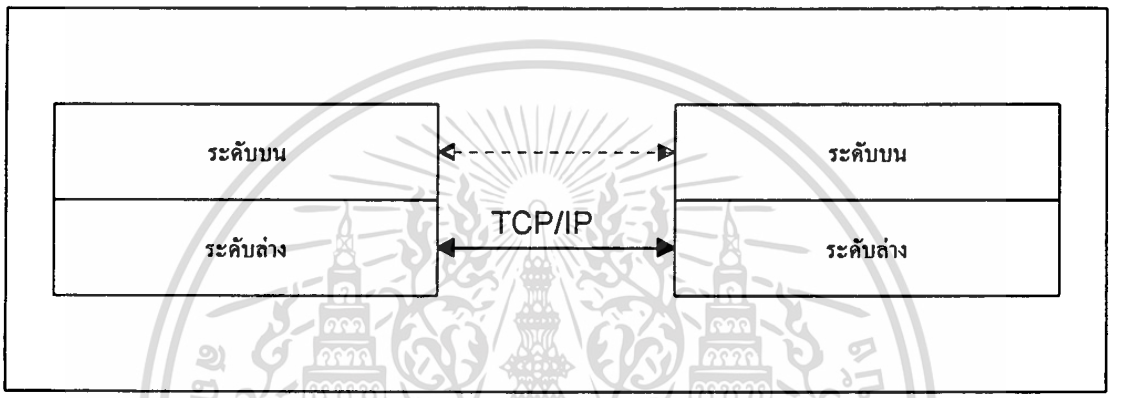
```
        fflush(stdout);             /* abort doesn'tflust stdio buffers */
        fflush(stderr);
        abort();                    /* dump core and terminate */
        exit(-1) ;                  /* shouldn't get here */
}

/*
 * Print the UNIX errno value.
 */

my_perror()
{
        char   *sys_err_str();

        fprintf(stderr, " %s\n", sys_err_str());
}


/* remainder is for both CLIENT and SERVER */

extern int     errno;          /* Unix error number */
extern int     sys_nerr;       /* # of error message strings in sys table */
extern char    *sys_errlist[]; /* the sytem eror message table */

#ifdef SYS5
int     t_errno;        /* in case caller in using TLI, these are "tentative definitions"; else they're "definitions" */
int     t_nerr;
char    *t_errlist[1];
#endif

/*
 * Return a string containing some additional operating-system
 * dependent information.
 * Note that different versions of UNIX assign different meanings
 * to the same value of "errno" (compare errno's starting with 35
 * between System V and BSD, for example). This means that if an error
 * condition is being sent to another UNIX system, we must interpret
 * the errno value on the system that generated the error, and not
 * just send the decimal value of errno to the other system.
 */

char *sys_err_str()
{
        static char     msgstr[200];

        if (errno != 0)
        {
                if ( errno > 0   &&   errno < sys_nerr )
                        sprintf(msgstr, "(%s)", sys_errlist[errno]);
                else
                        sprintf(msgstr, "(errno = %d)", errno);
        }
        else
        {
                msgstr[0] = '\0';
        }
#ifdef SYS5
        if (t_erno != 0)
        {
                char    tmsgstr[100];

                if (t_errno > 0   &&   t_errno < sys_nerr)
                        sprintf(tmsgstr, " (%s)", t_errlist[t_errno]);
                else
                        sprintf(tmsgstr, ', (t_errno = %d)", t_errno);

                strcat(msgstr, tmsgstr);     /* catenate strings */
        }
```

```
#endif
        return(msgstr);
}

/*
 * Return a string xontaining some additional infornation after a
 * host name or address lookup error - gethostbyname() or gethostbyaddr().
 */

int     h_errno;                /* host error number */
char *
host_err_str()
{
        static char     *mesg[]={"HOST_NOT_FOUND", TRY_AGAIN", NO_RECOVERY", NO_ADDRESS"};
        static char unknow[20];
        switch(h_errno)
        {
                case 1:
                        return(mesg[0]);
                break;
                case 2:
                        return(mesg[1]);
                break;
                case 3:
                        return(mesg[2]);
                break;
                case 4:
                        return(mesg[3]);
                break;
                default:
                        sprintf(unknow, "(h_errno = %d)", h_errno);
                        return(unknow);
                break;
        }
}
```

ภาคผนวก จ

โปรโตคอลระดับสูงที่ใช้ในระบบ DDQ/1

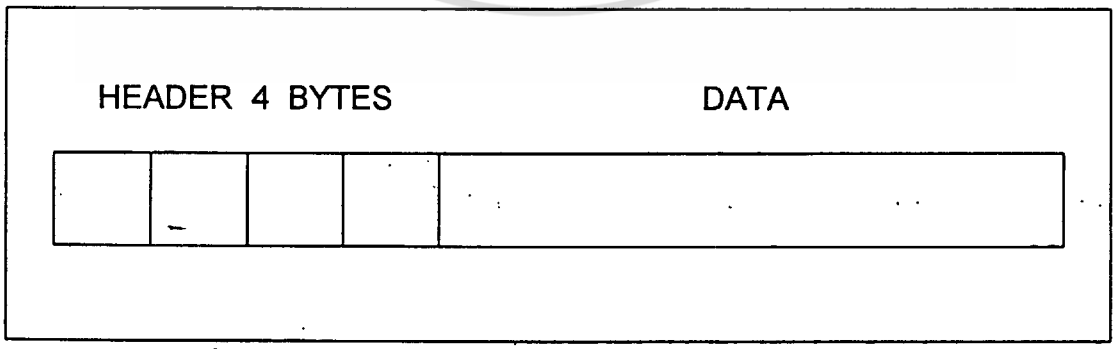
แม้ว่าในการพัฒนาระบบ DDQ/1 จะใช้โปรโตคอล TCP/IP แล้วก็ตาม แต่ก็ยังไม่เพียงพอกับการใช้งาน ทั้งนี้เนื่องจากว่าในการรับส่งข้อมูลของระบบ DDQ/1 มีข้อมูลหลายประเภท และระบบ DDQ/1 จำเป็นจะต้องควบคุมและตรวสอบข้อมูลเหล่านั้นเอง ดังนั้นจึงต้องมีการกำหนดโปรโตคอลเป็นของตัวเอง ซึ่งแบ่งออกได้เป็น 2 ระดับดังแสดงไว้ในรูปที่ จ.1



รูปที่ จ.1 แสดงระดับโปรโตคอลที่ใช้ในระบบ DDQ/1

1. โปรโตคอลระดับล่าง

โปรโตคอลระดับล่างนี้ จะทำหน้าที่ในการควบคุมจำนวนข้อมูลที่ส่ง ด้วยการบอกจำนวนข้อมูลที่ส่ง (DATA) ไว้หน้าข้อมูลนั้นๆ (HEADER) ในลักษณะตัวอักษร (ASCII) ดังแสดงในรูปที่ จ.2 โดยข้อมูลที่จะใช้โปรโตคอลนี้จะต้องผ่านการใช้โปรโตคอลระดับบนก่อน ซึ่งโปรโตคอลส่วนนี้จะอยู่ในส่วนเชื่อมต่อโครงข่าย



รูปที่ จ.2 แสดงโครงสร้างของโปรโตคอลระดับล่างในระบบ DDQ/1

2. โปรโตคอลระดับบน

โปรโตคอลระดับบนนี้จะอยู่ในส่วนประมวลผลคำถามแบบกระจาย โดยจะทำหน้าที่ในการควบคุมการรับ-ส่งข้อมูล เป็นดังนี้คือ 3 ไบต์แรกจะบอกชนิดของข้อมูล ซึ่งมีด้วยกัน 10 ชนิด คือ

2.1 EN

EN จะส่งข้อมูล $00 เพียง 3 ไบต์เพื่อบอกว่าข้อมูลได้ส่งไปหมดแล้ว

2.2 QU

QU นี้เป็นการบอกว่าข้อมูลที่ส่งเป็นคำถาม โดยจะส่งข้อมูล 3 ไบต์แรกเป็น $01 และ 3 ไบต์ต่อมาจะบอกจำนวนโนดของคำถาม จากนั้นจึงเป็นข้อมูลของคำถาม

2.3 DA

DA จะบอกว่าหลังจาก ข้อมูล 3 ไบต์คือ $02 นี้แล้ว จะเป็นข้อมูลที่ต้องการจะส่งจริงๆ

2.4 OK

OK จะส่งเพียงข้อมูล 3 ไบต์เท่านั้นคือ $03 เพื่อเป็นการบอกว่าได้รับข้อมูลถูกต้อง

2.5 ER

ER จะบอกว่าหลังจาก ข้อมูล 3 ไบต์คือ $04 นี้แล้ว จะเป็นข้อความที่แจ้งถึงความผิดพลาดของการทำงาน

2.6 US

US จะบอกชื่อของผู้ใช้งานไว้หลังจาก ข้อมูล 3 ไบต์คือ $05

2.7 PW

PW จะบอกรหัสผ่านของผู้ใช้งานไว้หลังจาก ข้อมูล 3 ไบต์คือ $06

2.8 CM

CM นี้เป็นการบอกว่าข้อมูลที่ส่งเป็นคำสั่ง โดยจะส่งข้อมูล 3 ไบต์แรกเป็น $07 และ 3 ไบต์ต่อมาเป็นคำสั่ง ซึ่งคำสั่งเป็นดังตารางที่ จ.3

คำสั่ง	ข้อมูล 3 ไบต์ที่ส่ง
SEND	@00
UNION	#01
MINUS	#02
INTERSECT	#03
TIMES	#04
JOIN	#05
DIVIDEBY	#06

ตารางที่ จ. 3 แสดงข้อมูล 3 ไบต์ที่ต่อจาก CM

จากนั้นถ้าข้อมูล 3 ไบต์ที่ส่งตามตารางที่ จ. 3 เป็น #01-#06 ก็จะมีข้อมูลอีก 1 ไบต์เป็น L (Left) หรือ R (Right) ที่บอกถึงตำแหน่งของส่วนของคำถามที่จะสร้างเป็นคำถามเชิงต้นไม้ ในกรณีที่มีการประมวลผลต่อที่สถานที่อื่น

2.9 EX

EX จะบอกกับส่วนประมวลผลคำถามแบบกระจายของสถานที่อื่นว่าเลิกการติดต่อ ด้วยการส่งข้อมูลเพียง 3 ไบต์คือ $08

2.10 VW

VW เป็นการบอกว่าข้อมูลที่ต่อจากข้อมูล 3 ไบต์คือ $09 จะเป็นรายละเอียดเกี่ยวกับการประมวลผล ซึ่งจะใช้ในกรณีที่มีการประมวลผลต่อเนื่อง หลังจากที่ได้รับผลลัพธ์จากการส่งคำถามไปประมวลผลที่สถานที่อื่น

ประวัติผู้เขียน

ชื่อผู้เขียน	นายสมชิด ศิริผลหลาย
วันเดือนปีเกิด	วันที่ 7 ตุลาคม พ.ศ.2508
สถานที่เกิด	จังหวัดมหาสารคาม
วุฒิการศึกษาระดับปริญญาตรี	วิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้าอิเลคทรอนิคส์
ปีที่สำเร็จการศึกษา	ปีการศึกษา 2530
ผลงานทางวิชาการที่ได้รับการตีพิมพ์	เรื่องระบบประมวลผลคำถามสำหรับฐานข้อมูลแบบกระจาย ในโครงข่ายคอมพิวเตอร์ TCP/IP
ประสพการณ์การทำงาน	ช่างระดับ 2 กองช่างระบบสื่อสารการบิน บริษัทวิทยุการบินแห่งประเทศไทย เมื่อปี พ.ศ. 2531 ถึง พ.ศ. 2532