ระบบผู้เชี่ยวชาญสำหรับออกแบบฐานข้อมูลรีเลชั่นแนล

AN EXPERT SYSTEM FOR RELATIONAL DATABASE DESIGN

นายพิทักษ์ ธรรมวาริน

MR.PITAK THUMWARIN

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต

สาขาวิชาวิทยาการคอมพิวเตอร์และเทคโนโลยีสารสนเทศ

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2539

ISBN 974-621-526-4

AN EXPERT SYSTEM FOR RELATIONAL DATABASE DESIGN

PITAK THUMWARIN

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE
MASTER OF SCIENCE PROGRAM IN COMPUTER SCIENCE
AND INFORMATION TECHNOLOGY
SCHOOL OF GRADUATE STUDIES
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
1996

ISBN 974-621-526-4

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองวิทยานิพนธ์

หัวข้อวิทยานิพนธ์ ระบบผู้เชี่ยวชาญสำหรับออกแบบฐานข้อมูลรีเลชั่นแนล

AN EXPERT SYSTEM FOR RELATIONAL DATABASE DESIGN

ชื่อนักศึกษา นายพิทักษ์ ธรรมวาริน **รหัสประจำตัว** 34628021

หลักสูตร วิทยาศาสตรมหาบัณฑิต

สาขาวิชา วิทยาการคอมพิวเตอร์และเทคโนโลยีสารสนเทศ

ภาควิชา คณิตศาสตร์และวิทยาการคอมพิวเตอร์

อาจารย์ผู้ควบคุมวิทยานิพนธ์ ผศ.ดร.ศุภมิตร จิตตะยโศธร

คณะกรรมการสอบวิทยานิพนธ์		ลายมือชื่อ
ผศ.ดร.ศุภมิตร	จิตตะยโศธร	
ดร.เอื้อน	ปิ่นเงิน	
ดร.พีระพนธ์	โสพัศสถิตย์	
อาจารย์บรรจง	ปียธำรง	

ค่าระดับคะแนนที่เป็นเอกฉันท์จากคณะกรรมการสอบ **GOOD**

วัน/เดือน/ปี ที่สอบ 5 เมษายน 2539 เวลา 10.00 น. เป็นต้นไป

สถานที่สอบ สอบห้อง 234 ณ ห้องสอบสัมมนาชั้น 2 อาคารสำนักวิจัยและบริการคอมพิวเตอร์

บัณฑิตวิทยาลัยรับรองแล้ว

(รศ.ดร.มนัส สังวรศิลป์)

คณบดีบัณฑิตวิทยาลัย

วันที่.....เดือน.....พ.ศ.....

หมายเหตุ การวัดผลวิทยานิพนธ์ให้ใช้ค่าระดับคะแนนดังนี้

ค่าระดับคะแนน	ผลการศึกษา
O	Outstanding (ดีเยี่ยม)
G	Good (ดี)
P	Pass (ผ่าน)
F	Fail (ไม่ผ่าน)

หัวข้อวิทยานิพนธ์	ระบบผู้เชี่ยวชาญสำหรับออกแบบฐานข้อมูลรีเลชั่นแนล
นักศึกษา	นายพิทักษ์ ธรรมวาริน
อาจารย์ผู้ควบคุมวิทยานิพนธ์	ผศ.ดร. สุภมิตร จิตตะยโสธร
ระดับการศึกษา	วิทยาศาสตรมหาบัณฑิต สาขาวิทยาการคอมพิวเตอร์และเทคโนโลยีสารสนเทศ
ภาควิชา	คณิตศาสตร์ และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ.	2539

บทคัดย่อ

ระบบผู้เชี่ยวชาญ (Expert system) เป็นแขนงหนึ่งของวิชาปัญญาประดิษฐ์ (Artificial Intelligence : AI) โดยตัวระบบผู้เชี่ยวชาญคือ โปรแกรมคอมพิวเตอร์ที่รวบรวมความรู้ และวิธีการวินิจฉัยปัญหาที่ยุ่งยากซับซ้อน ต้องใช้ประสบการณ์ความชำนาญของมนุษย์จึงจะแก้ไขได้ เปรียบเสมือนกับผู้เชี่ยวชาญในสาขานั้นจริง ๆ ที่ให้คำปรึกษาเกี่ยวกับเรื่องต่าง ๆ ในสาขานั้นได้

การออกแบบฐานข้อมูล เป็นการออกแบบการจัดเก็บข้อมูลลงในฐานข้อมูล เพื่อความปลอดภัย ความถูกต้อง และประหยัดเนื้อที่ในการเก็บข้อมูล นอกเหนือจากการออกแบบฐานข้อมูลแล้ว การปรับแต่งฐานข้อมูลก็ถือว่าเป็นสิ่งสำคัญสิ่งหนึ่งเช่นกัน โดยเฉพาะการกำหนดรูปแบบ และกลไกในการเข้าถึงข้อมูล ซึ่งจะช่วยเพิ่มประสิทธิภาพในการเข้าถึงข้อมูลในฐานข้อมูล ทั้งในด้านความเร็ว และการใช้ทรัพยากรของระบบ ซึ่งโดยปกติแล้วไม่ว่าจะเป็นการออกแบบ และการปรับแต่งฐานข้อมูล จะต้องใช้ผู้เชี่ยวชาญทางด้านนี้มากระทำ ซึ่งปัจจุบันปัญหาขาดแคลนผู้เชี่ยวชาญทางด้านนี้มีอยู่มาก

งานวิจัยนี้นำเสนอการสร้างระบบผู้เชี่ยวชาญ ในการออกแบบฐานข้อมูล และการปรับแต่งฐานข้อมูล ในรูปของการกำหนดรูปแบบ และกลไกในการเข้าถึงข้อมูล โดยยึดหลักการของ Normalization method เป็นหลักในการออกแบบฐานข้อมูล และสำหรับความรู้ในการกำหนดรูปแบบ และกลไกในการเข้าถึงข้อมูลได้มาจาก ตำราที่เกี่ยวกับการปรับแต่งฐานข้อมูล จากผู้เชี่ยวชาญทางด้านฐานข้อมูล จากคู่มือการใช้งานฐานข้อมูล และจากการนำความรู้ทั้งหมดที่ได้มาทำการทดลองกับฐานข้อมูลจริง ซึ่งในที่นี้ใช้ระบบจัดการฐานข้อมูลแบบสัมพันธ์ INGRES เพื่อหาข้อสรุปในกำหนดรูปแบบ และกลไกในการเข้าถึงข้อมูลที่แน่นอน ระบบที่สร้างขึ้นมานี้จะช่วยแก้ปัญหาการขาดแคลนบุคลากรดังกล่าวได้ เพราะสามารถทำงานแทนผู้เชี่ยวชาญดังกล่าวได้ ทั้งยังช่วยให้การออกแบบ และปรับแต่งฐานข้อมูล ทำได้สดวก รวดเร็ว และถูกต้องยิ่งขึ้น

Thesis Title	AN EXPERT SYSTEM FOR RELATIONAL DATABASE DESIGN
Student	Pitak Thumwarin
Thesis Advisor	Assoc. Prof Dr. Suphamit Chittayasothon
Level of Study	Master of Science Program in Computer science and Information technology
Department	Mathematical and Computerscience Faculty of Science King Mongkut's Institute of Technology Ladkrabang
Year	1996

ABSTRACT

Expert system is an active field in Artificial Intelligence (AI). An expert system is a collection of knowledge and methods for solving any complex problems that require human abilities , experiences and skills.

The basics of database design are accuracy , security and reduction of redundant data. Besides , database performance tuning is an important issue. Structure definition and data access mechanisms increase efficiency of data access and efficiency of resource usage. These activities normally are carried out by special experts which are very few in number.

This thesis presents an expert system for relational database design and performance tun , Structure definition and data access mechanism. Selection are provided Normalization method is used for the design of the logical schema. The structure defining and data access mechanisms come from texts related to database performance tuning , special experts , database manual usage and results from experiments database management systems INGRES. The system can be used as a database design and performance tuning assistance in the case that the human experts are not available.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลงด้วยดีเนื่องมาจากความร่วมมือ และช่วยเหลือจากบุคคลหลายฝ่ายด้วยกัน โดยเฉพาะอาจารย์ ศุภมิตร จิตตยะโสธร อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่คอยให้คำแนะนำ และให้คำปรึกษา ทั้งทางด้านวิชาการ และด้านอื่น ๆ ตลอดการทำวิทยานิพนธ์นี้ ซึ่งต้องขอขอบพระคุณท่านอาจารย์เป็นอย่างมาก และต้องขอขอบพระคุณบุคคลสำคัญอีกสองท่านที่คอยถามถึงให้กำลังใจ และช่วยเหลือข้าพเจ้ามาตลอดคือ บิดา มารดา อันเป็นที่รักเคารพยิ่งของข้าพเจ้า ซึ่งในชีวิตช่วงวัยศึกษาของข้าพเจ้า ท่านทั้งสองได้ทุ่มเททั้งกำลังกาย กำลังใจ อย่างมากจนหาที่เปรียบมิได้ ทำให้ข้าพเจ้าผ่านพ้นอุปสรรคต่าง ๆ จนมาถึง ณ จุดนี้ได้ พระคุณอันยิ่งใหญ่นี้ข้าพเจ้าจะขอระลึกถึงและจดจำไปชั่วชีวิต

ขอขอบคุณ พี่ เพื่อน และน้อง ชาวลาดกระบังทุกท่าน ที่คอยแนะนำ ให้กำลังใจ และช่วยเหลือข้าพเจ้ามาตลอดในการทำวิทยานิพนธ์นี้ ความเป็น พี่ เพื่อน น้อง และความช่วยเหลือเหล่านี้จะคงอยู่ในใจของข้าพเจ้าตลอดไป

ขอขอบคุณสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง โดยเฉพาะภาควิชาสถิติประยุกต์ คณะวิทยาศาสตร์ ที่ช่วยให้ความรู้พื้นฐานกับข้าพเจ้าในช่วงปริญญาตรี ภาควิชาคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สำนักวิจัย และบริการคอมพิวเตอร์ แผนกสารสนเทศ คณะวิศวกรรมศาสตร์ ที่ให้ความช่วยเหลือทั้งทางด้านความรู้ ตำรา และเครื่องมือต่าง ๆ ที่ใช้ในการทำวิทยานิพนธ์นี้

ขอขอบคุณบริษัทไมโครเอกซ์ผู้ซึ่งเป็นตัวแทนจำหน่ายระบบจัดการฐานข้อมูล INGRES ที่ได้ให้คำแนะนำต่าง ๆ แก่ข้าพเจ้า

สุดท้ายขอขอบคุณ มูลนิธิ C&C และผู้ที่เกี่ยวข้องทุกท่าน รวมทั้งบัณฑิตวิทยาลัย ที่ได้มอบทุนการศึกษาให้กับข้าพเจ้า ซึ่งเป็นส่วนหนึ่งที่ทำให้วิทยานิพนธ์ฉบับนี้สำเร็จลงได้

พิทักษ์ ธรรมวาริน

สารบัญ

สารบัญ(ต่อ)

สารบัญ(ต่อ)

สารบัญตาราง

ตารางที่ หน้า

สารบัญภาพ

หน้า

สารบัญภาพ (ต่อ)

หน้า

บทที่ 1

บทนำ

ความสำคัญและที่มา

ฐานข้อมูลนับว่าเป็นสิ่งที่สำคัญสิ่งหนึ่งในหน่วยงานต่างๆ ถ้าฐานข้อมูลได้รับการออกแบบที่ดีและเหมาะสมแล้ว จะทำให้การบริหารการใช้งานข้อมูลในฐานข้อมูลได้อย่างมีประสิทธิภาพ ไม่ว่าจะเป็นทรัพยากรที่ใช้ในการเก็บข้อมูล เวลาที่ใช้ในการค้นหาข้อมูล เวลาที่ใช้ในการแก้ไขข้อมูล ตลอดจนความปลอดภัย และความถูกต้องของข้อมูลในฐานข้อมูล

เทคโนโลยีทางด้านฐานข้อมูลได้พัฒนาก้าวหน้าไปมาก โดยนักวิทยาศาสตร์ในสาขานี้เป็นจำนวนมากได้ทำวิจัย และพัฒนาวิธีการออกแบบ และเก็บข้อมูลลงในฐานข้อมูล โดยมีแบบจำลองข้อมูลในรูปแบบต่างๆ และมีวิธีการจัดเก็บที่เป็นระบบสามารถลดความซ้ำซ้อน (Redundancy) รักษาความถูกต้อง (Integrity) ลดการสูญหายของข่าวสาร (Information lost) และลดข้อผิดพลาดจากการแก้ไขข้อมูล (Update Anomalies)

นอกจากพัฒนาการออกทางด้านการออกแบบฐานข้อมูลแล้ว เทคโนโลยีทางด้านระบบจัดการฐานข้อมูล (Database Management Systems : DBMS) ก็ได้รับการพัฒนาไปอย่างมากเช่นเดียวกัน ระบบจัดการฐานข้อมูลได้รับการพัฒนาให้มีคุณภาพสูง และมีสถาปัตยกรรมที่ใกล้เคียงสถาปัตยกรรมมาตราฐานของ ANSI และ ISO นอกจากนี้ระบบจัดการฐานข้อมูลในปัจจุบันยังได้พัฒนาในส่วนของความปลอดภัย มีระบบเรียกข้อมูลคืนป้องกันการสูญหาย (Recovery) มีระบบใช้ข้อมูลร่วมกันระหว่างผู้ใช้หลายคน (Multi-user) และมีภาษาจัดการข้อมูลที่มีประสิทธิภาพ อย่างน้อยเทียบเท่ากับพีชคณิตรีเลชัน (Relational Algebra) และใช้เทคโนโลยีทางด้าน Query Optimization เพื่อให้การประมวลผล Query ได้ประสิทธิภาพสูงสุด

เทคโนโลยีทางด้านปัญญาประดิษฐ์ก็ได้พัฒนาไปมากเช่นกัน โดยเฉพาะเทคโนโลยีทางด้านการพัฒนาระบบผู้เชี่ยวชาญ ได้มีนักวิจัยในสาขาปัญญาประดิษฐ์พยายามสร้างแบบแทนความรู้ (Knowledge Representation) ชนิดต่างๆมาใช้แทนความรู้ (Domain Knowledge) เพื่อที่จะเก็บความ

รู้ลงบนฐานข้อมูลความรู้ (Knowledge Base) สำหรับใช้แก้ปัญหาต่างๆ ซึ่งระบบผู้เชี่ยวชาญนี้จะสามารถทำงานและตัดสินใจแทนผู้เชี่ยวชาญได้ จึงช่วยแก้ปัญหาการขาดผู้เชี่ยวชาญทางด้านต่างๆ ได้เป็นอย่างดี

ถึงแม้ว่าเทคโนโลยีทางด้านฐานข้อมูลได้พัฒนาไปมาก แต่ก็ยังคงต้องใช้บุคลากรที่มีรู้และเชี่ยวชาญทางด้านฐานข้อมูลมาเป็นผู้ออกแบบและปรับแต่งฐานข้อมูล ซึ่งในปัจจุบันปัญหาขาดแคลนบุคลากรทางด้านนี้มีมาก จึงได้พัฒนาระบบผู้เชี่ยวชาญสำหรับช่วยในการออกแบบ และปรับแต่งฐานข้อมูลขึ้นมา โดยในโครงงานวิจัยนี้ได้รวบรวมความรู้ในการออกแบบฐานข้อมูล และปรับแต่งฐานข้อมูลในรูปของการเลือกรูปแบบ และกลไกในการเข้าถึงข้อมูล จากผู้เชี่ยวชาญทางด้านฐานข้อมูล จากหนังสือ,งานวิจัยที่เกี่ยวข้องกับฐานข้อมูล และจากการทดลองกับระบบจัดการฐานข้อมูลของ INGRES จากนั้นนำความรู้ที่ได้มาสร้างเป็นระบบผู้เชี่ยวชาญสำหรับช่วยในการออกแบบ และปรับแต่งฐานข้อมูล โดยระบบนี้สามารถทำงานแทนผู้เชี่ยวชาญได้ ซึ่งนอกจากจะช่วยแก้ปัญหาขาดแคลนผู้เชี่ยวชาญแล้ว ยังช่วยให้การออกแบบ และการปรับแต่งฐานข้อมูลทำได้สดวกรวดเร็ว และถูกต้องยิ่งขึ้น

<u>ขอบเขตของงานวิจัย</u>

ขอบเขตของงานวิจัยนี้แบ่งได้เป็น 2 ส่วน

1. ขอบเขตในการออกแบบฐานข้อมูล

ในส่วนการออกแบบฐานข้อมูลกระทำบน Relational Model โดยใช้วิธี Normalization ซึ่งผลลัพท์ที่ได้จากการออกแบบของระบบจะได้ตารางข้อมูลที่อยู่ในรูปของ Fifth Normal From พร้อมแสดง Primary Key และ Candidate Key ของแต่ละตาราง

สำหรับข้อมูลที่จะนำมา Input สู่ระบบการออกแบบนั้นจะอยู่ในรูปของ Function dependency และชื่อกลุ่มของ Function dependency ซึ่งจะนำไปเป็นชื่อตารางของผลลัพธ์ที่ได้จากการออกแบบ

2. ขอบเขตในการปรับแต่งฐานข้อมูล

การปรับแต่งฐานข้อมูลจะกระทำในรูปของการกำหนดรูปแบบและกลไกในการเข้าถึงข้อมูลของ DBMS ซึ่งในงานวิจัยนี้ใช้ INGRES ความรู้ที่ใช้ในการปรับแต่งฐานข้อมูลนำมาจากทฤษฎีในการปรับแต่งฐานข้อมูลทั่วไปร่วมกับจากการศึกษาการทำงานของ Query Optimizer ใน INGRES โดยการทดลองประมวลผล Query ต่าง ๆ ในสภาวะของการเลือกรูปแบบและกลไกในการเข้าถึงข้อมูลที่แตกต่างกันแล้วพิจารณาการใช้ CPU time และ I/O time ของแต่ละทางเลือก จากนั้นนำความรู้ที่ได้มาเก็บในฐานข้อมูลความรู้ (Knowledge base) ของระบบ ผลลัพท์ที่ได้จากระบบนี้คือข้อเสนอแนะการเลือกรูปแบบ และวิธีการเข้าถึงข้อมูล โดยจะพิจารณาจากตารางข้อมูลที่ได้มาจากการออกแบบของส่วนออกแบบฐานข้อมูลในเบื้องต้น ร่วมกับ Query ที่ใช้งานอยู่ประจำ การใช้งานข้อมูลที่นอกเหนือจาก Query และตารางสถิติเก็บรายละเอียดการ กระจายของข้อมูลที่มีอยู่

<u>ประโยชน์ที่คาดว่าจะได้รับ</u>

แยกพิจารณาได้เป็น 2 ส่วน

1. ระบบออกแบบฐานข้อมูล

ระบบนี้จะช่วยให้การออกแบบฐานข้อมูลทำได้ง่าย และสะดวกขึ้น ลดข้อผิดพลาดในการออกแบบฐานข้อมูล การใช้งานในส่วนนี้เหมาะสำหรับผู้ที่มีความรู้เบื้องต้นในการออกแบบฐานข้อมูล แต่ไม่ได้เป็นผู้เชี่ยวชาญทางด้านนี้ หรือเป็นผู้ที่ยังไม่มีประสบการณ์ทางด้านนี้ ซึ่งระบบนี้จะช่วยให้สามารถออกแบบฐานข้อมูลได้อย่างรวดเร็ว และถูกต้องเหมือนกระทำโดยผู้เชี่ยวชาญ

2. ระบบปรับแต่งฐานข้อมูล

โดยทั่วไปใน RDBMS จะมี Query optimizer มาช่วยวิเคราะห์ ในการเลือกเส้นทางการทำงานของ Query เพื่อให้ได้ทางเลือกที่ดีที่สุด ระบบนี้จะช่วยผู้ที่ออกแบบฐานข้อมูลในการที่จะเลือกรูปแบบ และกลไกในการเข้าถึงข้อมูลให้สอดคล้องกับการทำงานของ Query optimizer ซึ่งตาม

ปกติแล้วการที่จะเลือกรูปแบบ และวิธีการในการเข้าถึงข้อมูลนั้น ผู้ออกแบบจะต้องศึกษาการทำงานของ Query optimizer ของ RDBMS ที่ใช้อยู่อย่างลึกซึ้ง และต้องมีความรู้ความเชี่ยวชาญเกี่ยวกับกลไกในการเข้าถึงข้อมูลเป็นอย่างมาก ประกอบกับต้องพิจารณาถึงการกระจายของข้อมูลจริงที่อยู่ในฐานข้อมูล อีกทั้งยังต้องพิจารณาถึงลักษณะการใช้งานฐานข้อมูลประกอบด้วย โดยทั้งหมดนี้จะต้องมีความสัมพันธ์กัน ซึ่งระบบนี้ได้บรรจุความรู้เหล่านี้ไว้ในฐานความรู้ และใช้ความรู้เหล่านี้มาช่วยในการเลือกรูปแบบ และกลไกในการเข้าถึงข้อมูลที่ดีที่สุด

<u>วิธีการดำเนินการและส่วนประกอบของวิทยานิพนธ์</u>

งานวิจัยนี้จะเริ่มต้นด้วยการศึกษาทฤษฎีพื้นฐานต่าง ๆ ที่เกี่ยวข้องกับงานวิจัยซึ่งมีเรื่องหลักอยู่ 3 เรื่องด้วยกันคือ

1. ทฤษฎีในการออกแบบฐานข้อมูล Relational รายละเอียดกล่าวในบทที่ 2
2. ทฤษฎีในการปรับแต่งฐานข้อมูล ในรูปของการเลือกรูปแบบและกลไกในการเข้าถึงข้อมูลที่เหมาะสม รายละเอียดกล่าวในบทที่ 3
3. ระบบผู้เชี่ยวชาญ การออกแบบและสร้างฐานความรู้ รายละเอียดกล่าวในบทที่ 5 และ 7

จากนั้นจะศึกษาเครื่องมือที่จะนำมาใช้ในการพัฒนางานวิจัยนี้ ซึ่งมีอยู่ 2 ชนิดด้วยกันคือ โปรแกรมที่ใช้พัฒนาระบบผู้เชี่ยวชาญ ใช้ PC/PLUS ซึ่งเป็น Expert system shell ร่วมกับภาษา LISP และ ระบบจัดการฐานข้อมูล INGRES ซึ่งรายละเอียดของทั้งสองนี้จะมีกล่าวไว้ใน ภาคผนวก ก และ ข สำหรับบทที่ 4 จะนำความรู้ที่ได้จากบทที่ 3 มาประยุกต์ใช้งานกับ INGRES เพื่อหาความรู้ที่จะใช้ปรับแต่งฐานข้อมูลบน INGRES ในบทที่ 6 จะเป็นการกล่าวถึงสถาปัตยกรรมของระบบซึ่งจะอธิบายถึงการทำงานและภาพรวมของระบบ บทที่ 7 จะเป็นการออกแบบฐานข้อมูลความรู้ที่ใช้ในระบบ (Knowledge base design) และกล่าวถึงทฤษฎีที่ใช้ในการลดความซ้ำซ้อนความรู้ที่มีอยู่ในฐานความรู้ จากนั้นก็จะเอาความรู้ทั้งหมดที่ได้ศึกษามาพัฒนาระบบซึ่งกล่าวในบทที่ 8 โดยจะกล่าวถึงเครื่องมือ ขั้นตอน และวิธีการในการพัฒนาระบบ

สำหรับบทที่ 9 จะแสดงวิธีการใช้งาน และการทดสอบระบบที่ได้พัฒนาขึ้น บทที่ 10 ซึ่งเป็นบทสุดท้ายจะเป็นสรุปการทำงาน ปัญหา และผลที่ได้รับจากงานวิจัยชิ้นนี้ ตลอดจนเสนอแนะแนวทางที่จะพัฒนางานวิจัยนี้เพิ่มเติมต่อไป สำหรับภาคผนวก ก จะเป็นการอธิบายถึงขั้นตอนการทำงาน

และการใช้งาน เปลือกระบบผู้เชี่ยวชาญ PC/PLUS โดยสรุป ภาคผนวก ข เป็นการกล่าวถึงการทำงานโดยทั่วไปของระบบจัดการฐานข้อมูล INGRES โดยสรุป

บทที่ 2

การออกแบบฐานข้อมูลแบบรีเลชั่นแนล

ฐานข้อมูลแบบรีเลชั่นแนล

ระบบฐานข้อมูลที่ใช้กันอยู่ในปัจจุบันสามารถแบ่งประเภทตามโมเดลข้อมูลได้ 3 ประเภท คือ เน็ตเวิร์ก ไฮราคี และ รีเลชั่นแนล ซึ่งรีเลชั่นแนลเป็นโมเดลในการเก็บข้อมูลที่นิยม และแพร่หลายมากที่สุดซึ่งอาจกล่าวได้ว่าระบบฐานข้อมูลส่วนใหญ่ในโลกนี้เป็นฐานข้อมูลแบบรีเลชั่นแนล ซึ่งดูได้จากซอฟแวร์ที่เกี่ยวกับระบบการจัดการฐานข้อมูลที่มีขายอยู่ทั่วไป ล้วนใช้โมเดลแบบรีเลชั่นแนลทั้งสิ้น

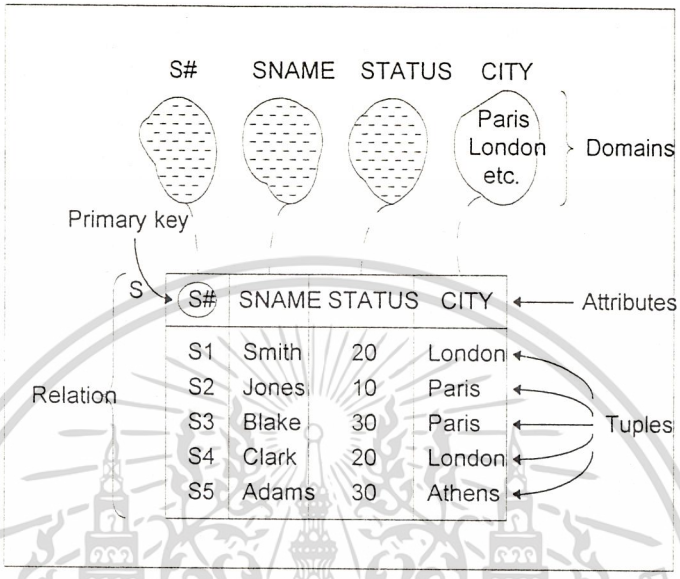
โครงสร้างฐานข้อมูลแบบรีเลชั่นแนล

ฐานข้อมูลแบบรีเลชั่นแนลจะแสดงการเก็บข้อมูลในรูปของตาราง ซึ่งง่ายต่อความเข้าใจ และสามารถตอบสนองต่อคำถามที่ซับซ้อนได้ดีกว่าโมเดลชนิดอื่น องค์ประกอบที่สำคัญของระบบจัดการฐานข้อมูลแบบรีเลชั่นแนลแบ่งเป็น 3 ส่วนคือ

- โครงสร้างข้อมูล
- ความถูกต้องของข้อมูล
- การจัดการข้อมูล

1. โครงสร้างข้อมูล (Data structure)

ภาพที่ 1



แสดงตารางความสัมพันธ์ข้อมูลลูกค้า

จากภาพที่ 1 มีคำเฉพาะที่จะใช้ในการเรียกขานส่วนต่างๆ ของตารางข้อมูลดังนี้

แอททริบิวต์ (Attributes) หมายถึง ชื่อคอลัมน์ของตารางความสัมพันธ์ ซึ่งจะเป็นตัวที่บอกถึงคุณลักษณะต่างๆที่มีความสัมพันธ์กันแล้วประกอบขึ้นเป็นตารางฐานข้อมูลเช่น แอททริบิวต์ SNAME หมายถึง ชื่อพนักงาน แอททริบิวต์ CITY หมายถึง ชื่อเมือง มีความสัมพันธ์กันคือ พนักงานชื่อ (SNAME) ..... อาศัยอยู่ที่เมือง (CITY) ..... เป็นต้น (ลักษณะของความสัมพันธ์ระหว่างแอททริบิวต์จะไม่สามารถมองเห็นได้จากตารางฐานข้อมูลแต่จะมองเห็นได้จากแบบจำลองฐานข้อมูลเช่น Fd Diagram,แผนภาพ NIAM หรือ แบบจำลองอีอาร์ เป็นต้น)

โดเมน (Domains) หมายถึง ขอบเขตของข้อมูลในแต่ละแอททริบิวต์เช่น ข้อมูลที่เก็บลงบนแอททริบิวต์ CITY จะต้องเป็นชื่อของเมืองต่างๆในทวีปยุโรปเท่านั้น หรือ ข้อมูลที่เก็บลงบนแอททริบิวต์ SNAME จะต้องเป็นชื่อคนเท่านั้นจะเป็นอย่างอื่นไม่ได้เป็นต้น

ทับเปิล (Tuples) หมายถึง แถวของตารางฐานข้อมูล

ไพรมารี่คีย์ (Primary key) หมายถึง แอททริบิวต์หรือกลุ่มของแอททริบิวต์ที่ทำหน้าที่เป็นตัวแยกความแตกต่างของข้อมูลในแต่ละทับเปิลในตารางฐานข้อมูล หรือกล่าวอีกนัยหนึ่งได้ว่าไพรมารี่คีย์คือ แอททริบิวต์หรือกลุ่มของแอททริบิวต์ที่จะเป็นตัวกำหนดเพื่อให้ได้ข้อมูล 1 ทับเปิลที่ไม่ซ้ำกัน (unique) จากตารางฐานข้อมูล

รีเลชั่น (Relation) หมายถึง ผลคูณคาร์ทีเซียนของโดเมนที่สนใจในแอพลิเคชั่น ประกอบขึ้นด้วยองค์ประกอบ 2 ส่วนคือ เฮดดิ้ง (heading) กับ บอดี้ (body) กำหนดให้ D1,D2,...,Dn แทนโดเมนทั้งหลายในระบบงาน

เฮดดิ้ง (Heading) ประกอบขึ้นมาจากเซ็ตที่มีขนาดคงที่ของแอททริบิวต์ A1,A2,...,An โดย Ai คือ แอททริบิวต์ที่แทนข้อมูลในโดเมน Di (I = 1,2,...,n)

บอดี้ (Body) ประกอบขึ้นจากเซ็ตของทัปเปิลที่มีขนาดแปรผันตามเวลาของรีเลชั่น โดยแต่ละทัปเปิลประกอบด้วยค่าข้อมูลของแอททริบิวต์ต่างๆ ซึ่งแสดงได้ดังนี้ (Ai:vi) (I= 1,2,3,..,n) คือแต่ละค่าแอททริบิวต์ Ai ในเฮดดิ้งจะกำหนดค่า (Ai:vi) โดยที่ vi เป็นต่างๆ ที่ไม่ซ้ำกันจากโดเมน Di ที่มีความสัมพันธ์กับแอททริบิวต์ Ai

ในหนึ่งทัปเปิลจะประกอบขึ้นด้วยค่าข้อมูลของแอททริบิวต์ที่อ้างอิงโดเมนต่างๆในตารางจากภาพที่ 1 แสดงได้ดังนี้

(S# : 'S1')

(SANME : 'SMIT')

(STATUS : 20)

(CITY : 'LONDON)

ตารางความสัมพันธ์ที่มีจำนวนแอททริบิวต์ n แอททริบิวต์ จะเรียกว่ามีดีกรีขนาด n (n degree) เรียกรีเลชั่นที่มีแอททริบิวต์เดียวว่า อันนารี่ (unary) รีเลชั่นที่มีสองแอททริบิวต์ว่า ไบนารี่ (binary) และรีเลชั่นที่มีสามแอททริบิวต์ว่า เทอร์นารี่ (ternary) และหากมีดีกรีขนาด n ก็เรียก เอนนารี่ (n-ary)

<u>คุณสมบัติของฐานข้อมูลรีเลชั่นแนล พิจารณาจากข้อกำหนด 4 ประการ คือ</u>

- ต้องไม่มีทัปเปิลที่ซ้ำกันภายในรีเลชั่น
- ลำดับของทัปเปิลไม่มีความสำคัญในการเก็บ
- ลำดับของแอททริบิวต์ไม่มีความสำคัญในการเก็บ
- ค่าข้อมูลของแต่ละแอททริบิวต์ต้องเป็นค่าเดียว (atomic value)

1.1 ต้องไม่มีทัปเปิลที่ซ้ำกันภายในรีเลชั่น

เนื่องจากบอดี้ของรีเลชั่นซึ่งประกอบขึ้นจากทัปเปิลหลายทัปเปิล มีลักษณะเป็นเซ็ตทางคณิตศาสตร์ในรูปผลคูณคาร์ทีเซียน การซ้ำกันเองของทัปเปิลจึงเหมือนกับการซ้ำกันของสมาชิกภายในเซ็ต ซึ่งผิดนิยามทางคณิตศาสตร์ ดังนั้นจึงกำหนดคุณสมบัติข้อนี้ เพื่อให้โครงสร้างข้อมูลแบบสัมพันธ์เป็นไปตามกฎทางคณิตศาสตร์ และเนื่องจากความเป็นหนึ่ง (ไม่ซ้ำกัน) ของแต่ละทัปเปิลในรีเลชั่นเอง ทำให้เราสามารถกล่าวได้ว่า "ทุก ๆ รีเลชั่นจะต้องมีไพรมารี่คีย์เกิดขึ้นเสมอ" เพราะอย่างน้อยที่สุดกลุ่มของแอททริบิวต์ที่ค่าข้อมูลมีความเป็นหนึ่ง ก็คือเซตของแอททริบิวต์ทั้งหมดในรีเลชั่นนั่นเอง

1.2. ลำดับของทัปเปิลไม่มีความสำคัญในการเก็บ

จากนิยามทางคณิตศาสตร์ที่ถือว่าสมาชิกภายในเซต จะอยู่กระจายไม่มีลำดับ บอดี้ของรีเลชั่นซึ่งเป็นเซตที่มีสมาชิกคือ ทัปเปิล จึงไม่ให้ความสำคัญกับลำดับของทัปเปิล

1.3. ลำดับของแอททริบิวต์ไม่มีความสำคัญในการเก็บ

ในนิยามเดียวกันกับคุณสมบัติข้อสอง ลำดับของแอททริบิวต์จึงไม่มีความสำคัญต่อการเก็บข้อมูลเช่นเดียวกัน

1.4. ข้อมูลของแต่ละแอททริบิวต์ต้องเป็นข้อมูลเดียวที่ไม่สามารถแบ่งแยกได้ หรือเรียกว่า อะตอมมิกแวลลู (atomic value)

หมายความว่าเมื่อมีการกำหนดชื่อรีเลชั่น ชื่อแอททริบิวต์ และทัปเปิลที่ต้องการแล้ว จะต้องได้ค่าข้อมูลแอททริบิวต์ออกมาเพียงค่าหนึ่งเท่านั้น หรือสามารถกล่าวได้อีกอย่างหนึ่งได้ว่า "รีเลชั่นใดๆ จะต้องไม่มีกลุ่มซ้ำ (repeating group) อยู่ภายใน"

2. ความถูกต้องของข้อมูล (Data integrity)

ความถูกต้องของข้อมูลในรีเลชั่น ขึ้นอยู่กับพฤติกรรมของข้อมูลที่มีต่อกฎความถูกต้อง (Integrity rules) ซึ่งมีอยู่ 2 กฎได้แก่

2.1 กฎความถูกต้องของเอนติตี้ (entity integrity)

เอนติตี้ข้อมูลซึ่งหมายถึงค่าของข้อมูลในแอททริบิวต์ ซึ่งเป็นแอททริบิวต์ไพรมารี่คีย์ของรีเลชั่นใดๆ ต้องไม่เป็นนัล (null) (นัล หมายถึง ไม่บันทึกค่า)

2.2 กฎความถูกต้องในการอ้างอิง (referential integrity)

ถ้ามีความสัมพันธ์ R2 ซึ่งมีฟอร์เรนคีย์ FK ที่เกี่ยวเนื่องกับไพรมารี่คีย์ PK ของความสัมพันธ์ R1 แล้ว ทุกๆค่าที่อยู่ใน FK ของความสัมพันธ์ R2 จะต้องปรากฏค่าอยู่ใน PK ของความสัมพันธ์ R1 เสมอยกเว้นในกรณีที่ค่าของ FK เป็นนัล

3. การจัดการข้อมูล (Data manipulation)

ข้อมูลภายในฐานข้อมูลจะถูกจัดการด้วยภาษาจัดการฐานข้อมูลภายในฐานข้อมูล ภาษาจัดการฐานข้อมูลนี้ต้องเป็นไปตามนิยามของ รีเลชั่นแนลที่สมบูรณ์คอมพลีท (relational complete) กล่าวคือต้องสนับสนุนกลุ่มปฏิบัติการพื้นฐานที่นิยามในภาษา พีชคณิตสัมพันธ์ (Relation Algebra) หรือ แคลคูลัสสัมพันธ์ (Relation Calculus) ซึ่งประกอบด้วยปฏิบัติการ 8 อย่างโดยแบ่งออกเป็น 2 กลุ่มได้ดังนี้

3.1. กลุ่มปฏิบัติการพื้นฐาน (Traditional Set Operations) ได้แก่ ผลคูณคาร์ทีเซียน,ยูเนียน ,อินเตอร์เซคชั่น และดิฟเฟอร์เรนซ์

3.2. กลุ่มปฏิบัติการพิเศษ (Special Relation Operations) ได้แก่ ซีเล็ค,โปรเจ็ค,จอยน์ และดิไวท์

<u>ลักษณะการทำงานของตัวปฏิบัติการต่างๆ</u>

SELECT คือ การสร้างความสัมพันธ์จากการเลือกเอาข้อมูลในทัปเปิล ที่มีคุณสมบัติตามที่ระบุออกมาจากความสัมพันธ์ที่มีอยู่

PROJECT คือ การสร้างความสัมพันธ์โดยการเลือกเอาข้อมูลจากแอททริบิวต์ บางตัวตามที่ได้กำหนด ออกมาจากความสัมพันธ์ที่มีอยู่

PRODUCT คือ การสร้างความสัมพันธ์ขึ้นมาใหม่จากความสัมพันธ์ 2 ความสัมพันธ์ ซึ่งความสัมพันธ์ที่สร้างขึ้น จะประกอบด้วยข้อมูลที่เป็นไปได้ทั้งหมด ที่เกิดจากการนำข้อมูลในทัปเปิลของความสัมพันธ์มาจับคู่กัน

สารบัญ(ต่อ)

UNION คือการสร้างความสัมพันธ์ขึ้นใหม่จากความสัมพันธ์ 2 ความสัมพันธ์ โดยความสัมพันธ์ที่สร้างขึ้นนี้ จะมีข้อมูลที่ประกอบไปด้วยข้อมูลของทั้งสองความสัมพันธ์

INTERSECT คือ การสร้างความสัมพันธ์จากความสัมพันธ์ 2 ความสัมพันธ์ที่กำหนดให้ โดยความสัมพันธ์ที่ได้ จะประกอบด้วยข้อมูลที่อยู่ในทัปเปิล ที่เหมือนกันของทั้งสองความสัมพันธ์ที่กำหนด

DIFFERENCE คือ การสร้างความสัมพันธ์จากความสัมพันธ์ 2 ความสัมพันธ์ที่กำหนดให้ ซึ่งความสัมพันธ์ที่เกิดขึ้นใหม่จะประกอบด้วย ข้อมูลในทัปเปิลทั้งหมดที่มีอยู่ในความสัมพันธ์แรกที่กำหนดให้ แต่ไม่อยู่ในความสัมพันธ์ที่สอง

JOIN การสร้างความสัมพันธ์จากความสัมพันธ์ 2 ความสัมพันธ์ โดยความสัมพันธ์ที่สร้างขึ้นใหม่ จะประกอบด้วยทัปเปิลที่เกิดจากการจับคู่ระหว่างทัปเปิลของทั้งสองความสัมพันธ์ซึ่งการจับคู่ของทัปเปิลนี้ จะต้องอยู่ภายใต้เงื่อนไขอย่างหนึ่ง

DIVIDE คือ การสร้างความสัมพันธ์จากความสัมพันธ์ 2 แบบ คือ ความสัมพันธ์แบบไบนารี่และความสัมพันธ์แบบยูนารี่ ซึ่งความสัมพันธ์ที่สร้างขึ้นจะประกอบด้วยข้อมูลทั้งหมดของแอททริบิวต์แรกของความสัมพันธ์แบบไบนารี่ ที่มีข้อมูลในแอททริบิวต์ที่สองเหมือนกับข้อมูลทั้งหมดในความสัมพันธ์แบบยูนารี่

<u>การออกแบบฐานข้อมูลในรูปนอร์มอลฟอร์ม</u>

หลักสำคัญในการจัดเก็บข้อมูลลงบนฐานข้อมูลคือ สามารถนำข้อมูลที่จัดเก็บมาใช้งานได้อย่างถูกต้อง และมีประสิทธิภาพสูงสุดตลอดจนใช้เนื้อที่ในการเก็บข้อมูลให้ได้ประโยชน์สูงสุด ซึ่งในกรณีที่ข้อมูลมีขนาดเล็กอาจใช้ความชำนาญ และประสบการณ์เข้ามาช่วยในการจัดเก็บได้ แต่ถ้าฐานข้อมูลที่มีขนาดใหญ่จำเป็นต้องมีวิธี และกฎเกณฑ์เข้ามาช่วยในการจัดเก็บข้อมูล ซึ่งสิ่งเหล่านี้สามารถช่วยผู้ที่ไม่มีประสบการณ์ และมีประสบการณ์ในด้านนี้ให้สามารถออกแบบการจัดเก็บข้อมูลได้อย่างมีประสิทธิภาพ อีกทั้งยังช่วยตรวจสอบความถูกต้องในการออกแบบด้วย นอร์มอลไลเซชั่นเป็นวิธีการในการออกแบบฐานข้อมูลแบบรีเลชันแนลให้อยู่ในรูปของนอร์มอลฟอร์ม โดยมีจุดประสงค์ในการออกแบบก็เพื่อที่จะให้ได้ตารางฐานข้อมูลที่ไม่มีความซ้ำซ้อนเกิดขึ้น ซึ่งฐานข้อมูลใดก็ตามที่ยังมีความซ้ำซ้อนอยู่อาจเป็นสาเหตุทำให้เกิดการประมวลผลข้อมูลที่ไม่ตรงกับความเป็นจริง (database inconsistency and modification anomalies)

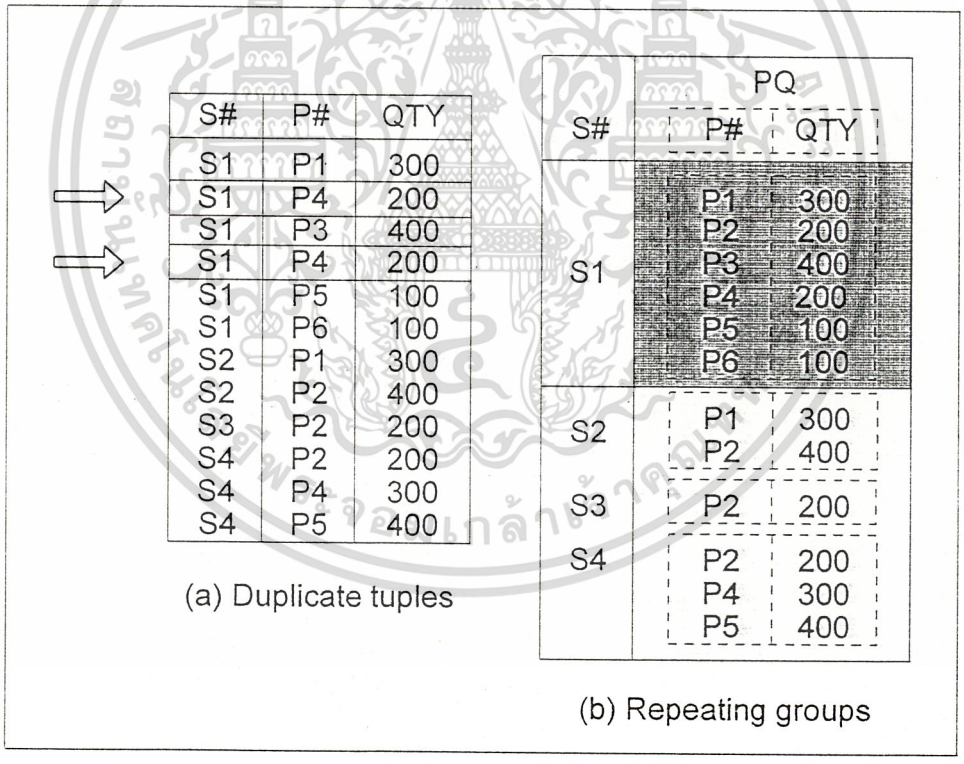
ฐานข้อมูลที่อยู่ในรูปนอร์มอลฟอร์มขั้นแรก (FIRST NORMAL FROM : 1NF)

ตารางฐานข้อมูลที่สามารถใช้ได้กับระบบจัดการฐานข้อมูลแบบรีเลชั่นแนลนั้น จะต้องมีคุณสมบัติอย่างน้อยที่สุดเป็นตารางฐานข้อมูลที่อยู่ในคุณสมบัติของรีเลชั่นดังที่กล่าวไว้ในข้างต้น และตารางที่มีคุณสมบัติ 1NF ก็ต่อเมื่อ ค่าทุกค่าของทุกแอททริบิวท์ในรีเลชั่นเป็นอะตอมมิค-แวลลู

12

ตารางที่ 1

แสดงตารางฐานข้อมูลที่ไม่อยู่ในรูปนอร์มอลฟอร์ม หรือเรียกอีกอย่างหนึ่งว่าไม่เป็นรีเลชั่น



S#	P#	QTY
S1	P1	300
S1	P4	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

(a) Duplicate tuples

S#	PQ	
	P#	QTY
S1	P1	300
	P2	200
	P3	400
	P4	200
	P5	100
	P6	100
S2	P1	300
	P2	400
S3	P2	200
S4	P2	200
	P4	300
	P5	400

(b) Repeating groups

ตารางที่ 2

แสดงตารางฐานข้อมูลที่อยู่ในรูปนอร์มอลฟอร์มขั้นแรก

S#	P#	QTY
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

<u>ฐานข้อมูลที่อยู่ในรูปนอร์มอลฟอร์มขั้นที่สอง</u>

ฐานข้อมูลที่อยู่ในรูปนอร์มอลฟอร์มขั้นที่สองต้องมีคุณสมบัติดังต่อไปนี้[12]

- ต้องเป็นฐานข้อมูลที่อยู่ในรูปนอร์มอลฟอร์มขั้นแรก
- ทุกแอททริบิวต์ที่ไม่ใช่ไพรมารี่คีย์ของตาราง ต้องขึ้นอยู่กับไพรมารี่คีย์เท่านั้นจะขึ้นอยู่กับสับเซ็ตของไพรมารี่คีย์ไม่ได้ หรือกล่าวอีกนัยหนึ่งว่าทุกแอททริบิวต์ที่ไม่ใช่ไพรมารี่คีย์ของตารางต้องขึ้นอยู่ทั้งหมด (Fully depen on) กับไพรมารี่คีย์

ในการที่จะตรวจสอบว่าแอททริบิวต์ใดขึ้นอยู่กับ (depend on) แอททริบิวต์ใดนั้น เราสามารถเขียนความสัมพันธ์ของแอททริบิวต์ให้อยู่ในรูปของ functional dependency (FD) ได้ซึ่งมีรูปแบบดังต่อไปนี้

[Determinant Attribute] --> [Dependent Attribute]

โดยความสัมพันธ์นี้จะอยู่ในรูปของ one to one หรือ many to one หรืออาจกล่าวได้ว่า Determinant Attribute คือ แอททริบิวต์ หรือกลุ่มของแอททริบิวต์ ที่สามารถกำหนดค่าของ Dependent Attribute หนึ่งค่าไม่ซ้ำกันได้ หรือมีคุณสมบัติเป็นหนึ่ง (unique) นั่นเอง

พิจารณาจากตัวอย่างต่อไปนี้

ตารางที่ 3

แสดงตารางฐานข้อมูลที่ไม่อยู่ในรูปนอร์มอลฟอร์มขั้นที่สอง

PROJECT-DATA

PERSON_ID	PROJ#	PROJ_BUDGET	TOTAL_TIME_SPENT_BY_PERSON_ON_PROJECT
P1	PROJ1	20	20
P3	PROJ1	20	16
P2	PROJ2	17	35
P2	PROJ3	84	42
P3	PROJ2	17	17
P2	PROJ1	20	83
P4	PROJ3	84	41
-	PROJ4	90	-

จากตารางฐานข้อมูลในตารางที่ 3 เป็นตารางฐานข้อมูลที่อยู่ในรูปนอร์มอลฟอร์มขั้นแรกที่มีไพรมารี่คีย์ของตารางคือแอททริบิวต์ PERSON_ID รวมกับ แอททริบิวต์ PROJ# ตารางฐานข้อมูลนี้ยังไม่เป็นตารางฐานข้อมูลในรูปนอร์มอลฟอร์มขั้นที่สองเนื่องจากแอททริบิวต์ PROJECT_BUDGET สามารถขึ้นอยู่กับแอททริบิวต์ที่เป็นสับเซ็ตของไพรมารี่คีย์คือ PROJ# เท่านั้น ซึ่งเขียนเป็น FD ได้ดังนี้

PERSON_ID , PROJ# --> TOTAL_TIME_SPENT_BY_PERSON_ON_PROJECT

PROJ# --> PROJ_BUDGET

จะเห็นว่า PROJ_BUDGET ไม่มีคุณสมบัติขึ้นอยู่กับทั้งหมดกับไพรมารี่คีย์ (fully depen on primary) ซึ่งทำให้เกิดความซ้ำซ้อนของตารางฐานข้อมูลขึ้นได้แก่ เมื่อไหร่ก็ตามที่เรากำหนด PROJ# จะได้ผลลัพธ์ของ PROJECT_BUDGET ซึ่งเป็นค่าเดียวกันจำนวนหลายแถว ข้อเสียคือถ้าต้องการแก้ไขข้อมูลของ PROJ# จะต้องแก้ไขข้อมูลของ PROJECT_BUDGET ด้วยค่าที่สอดคล้องกันจะแก้ไขเพียงแอททริบิวต์ใดแอททริบิวต์หนึ่งไม่ได้ จากเหตุผลดังกล่าวทำให้ข้อมูลที่ถูกเก็บลงในตารางฐานข้อมูลนี้อาจจะไม่ตรงกับความจริงได้ถ้าเมื่อไรก็ตามที่มีการแก้ไข PROJ# หรือ PROJECT_BUDGET เพียงแอททริบิวต์เดียว วิธีการกำจัดความซ้ำซ้อนของตารางฐานข้อมูลนี้เราต้องแยกตารางฐานข้อมูลนี้ออกเป็นสองตารางดังแสดงในตารางที่ 4 ซึ่งตารางฐานข้อมูลทั้งสองที่เกิดขึ้นใหม่นี้จัดอยู่ในตารางฐานข้อมูลในรูปนอร์มอลฟอร์มขั้นที่สอง

ตารางที่ 4

แสดงตารางฐานข้อมูลที่อยู่ในรูปนอร์มอลฟอร์มขั้นที่สอง

PROJECTS

PROJ#	PROJ_BUDGET
PROJ1	20
PROJ2	17
PROJ3	84
PROJ4	90

WORK

PERSON_ID	PROJ#	TOTAL_TIME_SPENT_BY_PERSON_ON_PROJECT
P1	PROJ1	20
P3	PROJ1	16
P2	PROJ2	35
P2	PROJ3	42
P3	PROJ2	17
P3	PROJ1	83
P4	PROJ3	41

ฐานข้อมูลที่อยู่ในรูปนอร์มอลฟอร์มขั้นที่สาม

ฐานข้อมูลที่อยู่ในรูปนอร์มอลฟอร์มขั้นที่สามต้องมีคุณสมบัติดังต่อไปนี้ [12]

- ต้องเป็นฐานข้อมูลที่อยู่ในรูปนอร์มอลฟอร์มขั้นที่สอง
- ทุกแอททริบิวต์ที่ไม่ใช่ไพรมารี่คีย์ของตาราง ต้องไม่ขึ้นอยู่กับแอททริบิวต์ที่ไม่ใช่ไพรมารี่คีย์ของตารางด้วยกันเอง

ตารางที่ 5

แสดงตารางฐานข้อมูลที่ไม่อยู่ในรูปนอร์มอลฟอร์มขั้นที่สาม

PROJECTS

PROJ#	MANAGER	DATE_OF_BIRTH
P1	Joe	Jan 63
P3	Vicki	March 57
P2	Joe	Jan 63
P4	Marilyn	July 53

จากตารางฐานข้อมูลในตารางที่ 5 เป็นตารางฐานข้อมูลที่อยู่ในรูปนอร์มอลฟอร์มขั้นที่สอง เนื่องจากไพรมารี่คีย์ของตารางประกอบด้วยแอททริบิวต์เพียงแอททริบิวต์เดียวคือ PROJ# ตารางฐานข้อมูลนี้ยังไม่เป็นตารางฐานข้อมูลในรูปนอร์มอลฟอร์มขั้นที่สาม เนื่องจากแอททริบิวต์ที่ไม่ได้เป็นไพรมารี่คีย์ของตารางคือ DATE_OF_BIRTH สามารถขึ้นอยู่กับแอททริบิวต์ MANAGER ซึ่งเป็นแอททริบิวต์ที่ไม่เป็นไพรมารี่คีย์ของตารางเช่นเดียวกันซึ่งแสดงได้ด้วย FD ดังนี้

PROJ# --> MANAGER

PROJ# --> DATE_OF_BIRTH

MANAGER --> DATE_OF_BIRTH

จะเห็นว่าแอททริบิวต์ DATE_OF_BIRTH ขึ้นอยู่กับแอททริบิวต์ที่ไม่เป็นคีย์ ซึ่งทำให้เกิดความซ้ำซ้อนของตารางฐานข้อมูลขึ้นได้แก่ เมื่อใดก็ตามที่เรากำหนด MANAGER จะได้ผลลัพธ์ของ DATE_OF_BIRTH ซึ่งเป็นค่าเดียวกันจำนวนหลายแถว ข้อเสียคือถ้าต้องการแก้ไขข้อมูลของ MANAGER จะต้องแก้ไขข้อมูลของ DATE_OF_BIRTH ด้วยค่าที่สอดคล้องกันจะแก้ไขเพียงแอททริบิวต์ใดแอททริบิวต์หนึ่งไม่ได้ จากเหตุผลดังกล่าวทำให้ข้อมูลที่ถูกเก็บลงในตารางฐานข้อมูลนี้อาจจะไม่ตรงกับความจริงได้ถ้าเมื่อไรก็ตามที่มีการแก้ไข MANAGER หรือ DATE_OF_BIRTH เพียงแอททริบิวต์เดียว วิธีการกำจัดความซ้ำซ้อนของตารางฐานข้อมูลนี้เราต้องแยกตารางฐานข้อมูลนี้ออกเป็นสองตารางดังแสดงในตารางที่ 6 ซึ่งตารางฐานข้อมูลทั้งสองที่เกิดขึ้นใหม่นี้จัดอยู่ในตารางฐานข้อมูลในรูปนอร์มอลฟอร์มขั้นที่สาม

ตารางที่ 6

แสดงตารางฐานข้อมูลที่อยู่ในรูปนอร์มอลฟอร์มขั้นที่สาม

PROJECTS

PROJ#	MANAGER
P1	Joe
P2	Joe
P3	Vicki
P4	Marilyn

MANAGERS

MANAGER	DATE_OF_BIRTH
Joe	Jan 63
Vicki	March 57
Marilyn	July 53

ฐานข้อมูลที่อยู่ในรูปนอร์มอลฟอร์มขั้น Boyce/Codd

ฐานข้อมูลที่อยู่ในรูปนอร์มอลฟอร์มขั้น Boyce/Codd ต้องมีคุณสมบัติดังต่อไปนี้ [12]

-ต้องเป็นฐานข้อมูลที่อยู่ในรูปนอร์มอลฟอร์มขั้นที่สาม

-ทุกแอททริบิวต์ที่เป็นไพรมารี่คีย์ของตารางจะต้องไม่ขึ้นอยู่กับสับเซ็ตของไพรมารี่คีย์อื่น หรือกล่าวอีกนัยหนึ่งว่าทุก determinant ต้องเป็น candidate key

ตารางที่ 7

แสดงตารางฐานข้อมูลที่ไม่อยู่ในรูปนอร์มอลฟอร์มขั้น Boyce/Codd

ATTENDANCE

TEACHER	SEMESTER	SUBJECT	SECTION	ATTENDANCES
Joe	1/88	COBOL	SECTION1	35
Jenny	1/88	MATHS	SECTION1	40
Gregory	2/88	UNIX	SECTION2	33
Jenny	1/88	MATHS	SECTION2	42
Gregory	2/88	UNIX	SECTION1	47
Joe	1/88	COBOL	SECTION2	50
Joe	1/88	COBOL	SECTION3	12

จากตารางฐานข้อมูลในตารางที่ 7 เป็นตารางฐานข้อมูลที่อยู่ในรูปนอร์มอลฟอร์มขั้นที่สาม เนื่องจากแอททริบิวต์ที่ไม่เป็นไพรมารี่คีย์ของตารางมีเพียงแอททริบิวต์เดียวคือ ATTENDANCES(ไม่มีแอททริบิวต์ที่ไม่ใช่ไพรมารี่คีย์ของตารางตัวอื่นมาขึ้นกับแอททริบิวต์ ATTENDANCES) โดยมีไพรมารี่คีย์ของตารางจำนวนสองตัวได้แก่ กลุ่มแอททริบิวต์ {TEACHER,SEMESTER,SECTION} กับ {SUBJECT,SEMESTER,SECTION} ตารางฐานข้อมูลนี้ยังไม่เป็นตารางฐานข้อมูลในรูปนอร์มอลฟอร์มขั้น Boyce/Codd เนื่องจากแอททริบิวต์ที่เป็นส่วนหนึ่งในไพรมารี่คีย์ของตารางตัวที่หนึ่งคือ TEACHER ขึ้นอยู่กับสับเซ็ตของไพรมารี่คีย์ตัวที่สองคือแอททริบิวต์ SEMESTER รวมกับ SUBJECT แสดงได้ดัง FD ต่อไปนี้

TEACHER,SEMESTER,SECTION --> ATTENDANCES

SEMESTER,SUBJECT --> TEACHER

จาก FD จะเห็นว่า SEMESTER SUBJECT เป็น determinant ที่ไม่ได้เป็นคีย์ ซึ่งทำให้เกิดความซ้ำซ้อนของตารางฐานข้อมูลขึ้นได้แก่ เมื่อไหร่ก็ตามที่เรากำหนด SEMESTER กับ SUBJECT จะได้ผลลัพธ์ของ TEACHER ซึ่งเป็นค่าเดียวกันจำนวนหลายแถว ข้อเสียคือถ้าต้องการแก้ไขข้อมูลของ SEMESTER กับ SUBJECT จะต้องแก้ไขข้อมูลของ TEACHER ด้วยค่าที่สอดคล้องกันจะแก้ไขเพียงแอททริบิวต์ใดแอททริบิวต์หนึ่งไม่ได้ จากเหตุผลดังกล่าวทำให้ข้อมูลที่ถูกเก็บลงในตารางฐานข้อมูลนี้อาจจะไม่ตรงกับความจริงได้ถ้าเมื่อไรก็ตามที่มีการแก้ไข SEMESTER กับ SUBJECT หรือ TEACHER เพียงแอททริบิวต์เดียว วิธีการกำจัดความซ้ำซ้อนของตารางฐานข้อมูลนี้เราต้องแยกตารางฐานข้อมูลนี้ออกเป็นสองตารางดังแสดงในตารางที่ 8 ซึ่งตารางฐานข้อมูลทั้งสองที่เกิดขึ้นใหม่นี้จัดอยู่ในตารางฐานข้อมูลในรูปนอร์มอลฟอร์มขั้น Boyce Codd

ตารางที่ 8

แสดงตารางฐานข้อมูลที่อยู่ในรูปนอร์มอลฟอร์มขั้น Boyce/Codd

TEACHERS

TEACHER	SEMESTER	SUBJECT
Joe	1/88	COBOL
Jenny	1/88	MATHS
Gregory	2/88	UNIX

ATTENDANCE

SEMESTER	SUBJECT	SECTION	ATTENDANCES
1/88	COBOL	SECTION1	35
1/88	MATHS	SECTION1	40
2/88	UNIX	SECTION2	33
1/88	MATHS	SECTION2	42
2/88	UNIX	SECTION1	47
1/88	COBOL	SECTION2	50
1/88	COBOL	SECTION3	12

ฐานข้อมูลที่อยู่ในรูปนอร์มอลฟอร์มขั้นที่สี่

ฐานข้อมูลที่อยู่ในรูปนอร์มอลฟอร์มขั้นที่สี่ต้องมีคุณสมบัติดังต่อไปนี้ [12]

- ต้องเป็นฐานข้อมูลที่อยู่ในรูปนอร์มอลฟอร์มขั้น Boyce/Codd
- ต้องไม่มี multivalued dependency (MVD) มากเกินหนึ่งตัว

ในบางครั้ง FD ไม่สามารถอธิบายความสัมพันธ์บางอย่างได้เช่น คน ๆ หนึ่ง (Person_id) มีความชำนาญ (skill) ได้หลายด้าน และความชำนาญแต่ละด้านก็สามารถมีได้หลายคน ซึ่งเราสามารถแสดงความสัมพันธ์เหล่านี้ได้ด้วย Multivalued dependency ดังนี้

Person_id --> --> skill

อ่านว่า แอททริบิวท์ skill multidependent แอททริบิวท์ Person_id

ตารางที่ 9

แสดงตารางฐานข้อมูลที่ไม่อยู่ในรูปนอร์มอลฟอร์มขั้นสี่

CTX

COURSE	TEACHER	TEXT
Physics	Prof. Green	Basic Mechanics
Physics	Prof. Green	Principles of Optics
Physics	Prof. Brown	Basic Mechanics
Physics	Prof. Brown	Principles of Optics
Math	Prof. Green	Basic Mechanics
Math	Prof. Green	Vector Analysis
Math	Prof. Green	Trigonometry

จากตารางฐานข้อมูลในตารางที่ 9 เป็นตารางฐานข้อมูลที่อยู่ในรูปนอร์มอลฟอร์มขั้น Boyce/Codd เนื่องจาก มีไพรมารี่คีย์เพียงตัวเดียว ตารางฐานข้อมูลนี้ยังไม่เป็นตารางฐานข้อมูลในรูปนอร์มอลฟอร์มขั้นที่สี่ เนื่องจากมี MVD มากกว่าหนึ่งตัว ได้แก่ เมื่อเรากำหนด COURSE จะได้เซ็ตของ TEACHER และเซ็ตของ TEXT ข้อเสียของความซ้ำซ้อนที่เกิดขึ้นได้แก่ เมื่อต้องการเพิ่มข้อมูลเพียงว่า วิชา MATH สามารถสอนด้วย Prof. White เราจำเป็นอย่างยิ่งที่จะต้อง INSERT ข้อมูลนี้ถึง 3 Tuples เนื่องจาก TEXT ที่ใช้ในการสอนวิชา MATH มีด้วยกันจำนวนสามเล่ม ดังนั้นจะเห็นว่าเราไม่สามารถที่จะ INSERT ข้อมูลทีละ 1 Tuple ได้ถ้ามี MVD เกิดขึ้นมากกว่าหนึ่งตัวในตาราง

เดียวกัน วิธีการกำจัดความซ้ำซ้อนของตารางฐานข้อมูลนี้เราต้องแยกตารางฐานข้อมูลนี้ออกเป็นสองตารางดังแสดงในตารางที่ 10 ซึ่งตารางฐานข้อมูลทั้งสองที่เกิดขึ้นใหม่นี้จัดอยู่ในตารางฐานข้อมูลในรูปนอร์มอลฟอร์มขั้นที่สี่

ตารางที่ 10

แสดงตารางฐานข้อมูลที่อยู่ในรูปนอร์มอลฟอร์มขั้นสี่

CT

COURSE	TEACHER
Physics	Prof. Green
Physics	Prof. Brown
Math	Prof. Green

CX

COURSE	TEXT
Physics	Basic Mechanics
Physics	Principles of Optics
Math	Basic Mechanics
Math	Vector Analysis
Math	Trigonometry

ฐานข้อมูลที่อยู่ในรูปนอร์มอลฟอร์มขั้นที่ห้า

ฐานข้อมูลที่อยู่ในรูปนอร์มอลฟอร์มขั้นที่ห้าต้องมีคุณสมบัติดังต่อไปนี้ [12]

- ต้องเป็นฐานข้อมูลที่อยู่ในรูปนอร์มอลฟอร์มขั้นที่สี่
- ผลลัพธ์ของการ Join ของทุก ๆ Join dependency ต้องเหมือนเดิมไม่เกิน หรือสูญหาย (Lossless Join)

ตารางที่ 11

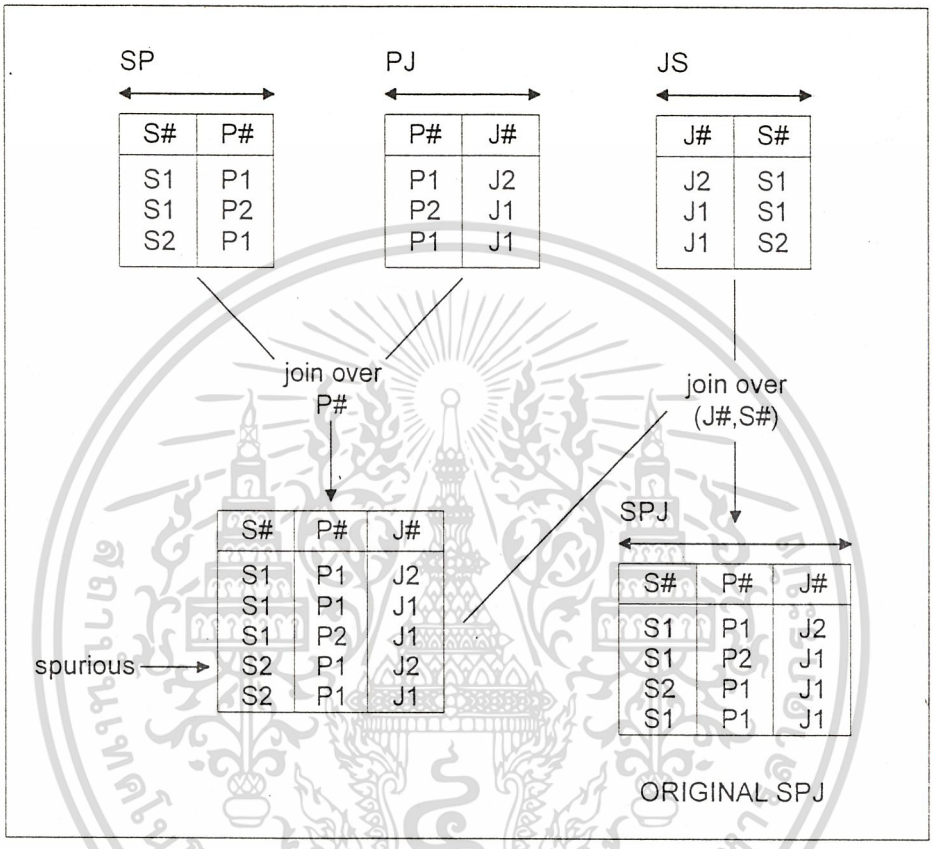
แสดงตารางฐานข้อมูลที่ไม่อยู่ในรูปนอร์มอลฟอร์มขั้นห้า

SPJ

S#	P#	J#
S1	P1	J2
S1	P2	J1
S2	P1	J1
S1	P1	J1

ตารางที่ 12

แสดงตารางฐานข้อมูลที่อยู่ในรูปนอร์มอลฟอร์มขั้นห้า



จากตารางฐานข้อมูลในตารางที่ 11 เป็นตารางฐานข้อมูลที่อยู่ในรูปนอร์มอลฟอร์มขั้นที่สี่ เนื่องจากทุกแอททริบิวต์รวมเป็นไพรมารี่คีย์ของตาราง และจากตารางฐานข้อมูลในตารางที่ 12 เราแยกตาราง SPJ ออกเป็น 3 ตารางคือ (SP,PJ,JS) จะเห็นว่าเมื่อทำการ join ตารางทั้งสามจะทำให้ผลลัพธ์ของข้อมูลในตารางมีค่าเท่ากับตารางเดิมทุกประการ แต่ถ้าทำการ join ตารางแค่เพียงสองตารางจะทำให้ข้อมูลของตารางที่เกิดขึ้นใหม่มีจำนวนเพิ่มมากขึ้นซึ่งไม่ถูกต้อง แสดงให้เห็นว่าเราไม่สามารถที่จะแยกตารางออกเป็นสองตารางได้แต่แยกเป็นสามตารางได้ ซึ่งการ join ของตารางทั้งสามตารางสามารถอธิบายได้ดังนี้

ถ้ามีข้อมูล (s1,p1) ปรากฏในตาราง SP

และมีข้อมูล (p1,j1) ปรากฏในตาราง PJ

และมีข้อมูล (j1,s1) ปรากฏในตาราง JS

แล้วจะต้องมีข้อมูล (s1,p1,j1) ปรากฏในตาราง SPJ

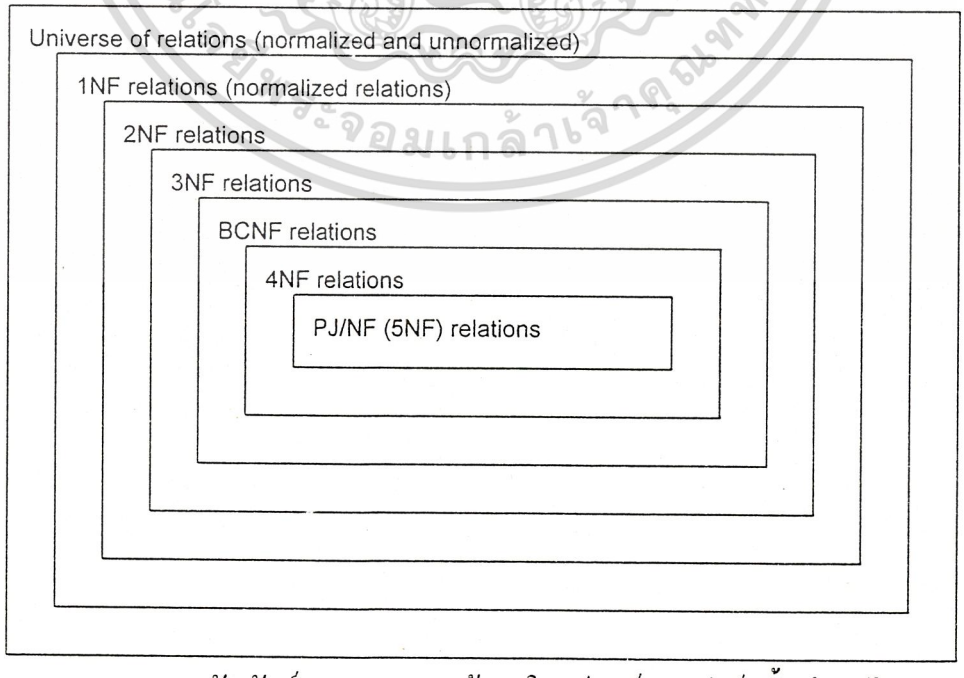
หรือกล่าวอีกนัยหนึ่งว่า

ถ้ามีข้อมูล (s1,p1,j2),(s2,p1,j1),(s1,p2,j1) ปรากฏในตาราง SPJ

แล้วจะต้องมีข้อมูล (s1,p1,j1) ปรากฏในตาราง SPJ ด้วยเช่นกัน ดังนั้นลักษณะความซ้ำซ้อนของตาราง SPJ คือ ถ้าเราต้องการแทรกข้อมูล (s2,p1,j1) เราจะต้องแทรกข้อมูล (s1,p1,j1) ด้วยเช่นเดียวกัน แต่ในทางกลับกันไม่จำเป็น และถ้าเราต้องการลบข้อมูล (s2,p1,j1) จะไม่มีผลกระทบใดๆ กับตาราง SPJ แต่ถ้าเราต้องการลบข้อมูล (s1,p1,j1) ข้อมูลอื่นจะต้องถูกลบทิ้งด้วยอีกหนึ่งแถว จะเห็นว่าในการแทรกและลบข้อมูลของตาราง SPJ มีความยุ่งยากมากและเกิดความผิดพลาดได้ง่าย ดังนั้นถ้าเราแยกตาราง SPJ ออกเป็นสามตารางแล้วปัญหาต่างๆที่เกิดขึ้นจะหมดไป ซึ่งตารางฐานข้อมูลทั้งสามเกิดขึ้นใหม่นี้จัดอยู่ในตารางฐานข้อมูลในรูปนอร์มอลฟอร์มขั้นที่ห้า เพราะการ join กันของตารางทั้งสามเป็น Lossless join

ที่ได้กล่าวมาทั้งหมดนี้ทำให้สามารถแสดงความสัมพันธ์ระหว่างฐานข้อมูลในรูปนอร์มอลฟอร์มขั้นต่างๆในรูปของเซ็ตได้ดังรูปที่ 2 คือ ตารางฐานข้อมูลที่อยู่ในรูปนอร์มอลฟอร์มขั้นที่ห้าซึ่งเป็นขั้นสูงสุดนั้น เป็นตารางฐานข้อมูลในรูปนอร์มอลฟอร์มขั้นที่สี่ด้วยแต่ตารางฐานข้อมูลที่อยู่ในรูปนอร์มอลฟอร์มขั้นที่สี่ไม่จำเป็นที่จะต้องเป็นตารางฐานข้อมูลที่อยู่ในรูปนอร์มอลฟอร์มขั้นที่ห้า หรือกล่าวโดยสรุปว่าตารางฐานข้อมูลที่อยู่ในรูปนอร์มอลฟอร์มขั้นที่ห้าจะต้องผ่านคุณสมบัติของการเป็นตารางฐานข้อมูลรูปนอร์มอลฟอร์มขั้นที่หนึ่งจนถึงขั้นที่สี่มาก่อนแล้วทั้งสิ้น และยังต้องมีคุณสมบัติของการเป็นตารางฐานข้อมูลรูปนอร์มอลฟอร์มขั้นที่ห้าอีกด้วย

ภาพที่ 2



แสดงความสัมพันธ์ของตารางฐานข้อมูลในรูปนอร์มอลฟอร์มขั้นต่างๆ [12]

บทที่ 3

การปรับแต่งฐานข้อมูลรีเลชั่นแนล

<u>บทนำ</u>

ในบทนี้จะกล่าวถึงการปรับแต่งฐานข้อมูลในส่วนของการกำหนดรูปแบบ และกลไกในการเข้าถึงข้อมูล ซึ่งมีเนื้อหาสรุปได้ดังนี้

ในส่วนแรก จะกล่าวถึงว่ากลไกในการเข้าถึงข้อมูลคืออะไร มีประโยชน์อย่างไรบ้าง และแสดงรูปแบบของกลไกในการเข้าถึงข้อมูลที่ใช้กันแพร่หลายบน RDBMS ที่มีอยู่ในปัจจุบัน

ในส่วนที่สอง จะกล่าวถึงภาษาที่ใช้งานฐานข้อมูลทั่วไป และภาษาที่ใช้ในการกำหนดรูปแบบ และกลไกในการเข้าถึงข้อมูลบน RDBMS

ในส่วนที่สาม จะกล่าวถึง Query optimizer บน RDBMS และความสัมพันธ์ระหว่างการทำงานของ Query optimizer กับ การกำหนดรูปแบบ และกลไกในการเข้าถึงข้อมูล

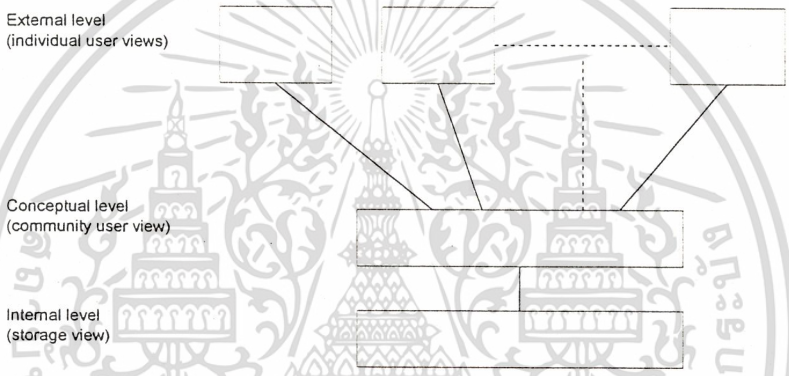
ในส่วนที่สี่ จะกล่าวถึงปัจจัยที่ใช้พิจารณาในการกำหนดรูปแบบ และกลไกในการเข้าถึงข้อมูล

ในส่วนสุดท้าย จะกล่าวถึงกฎเกณฑ์ และหลักการในการกำหนดรูปแบบ และกลไกในการเข้าถึงข้อมูล

<u>กลไกในการเข้าถึงข้อมูล</u>

สถาปัตยกรรมฐานข้อมูลมาตรฐานของ ANSI และ ISO ซึ่งเป็นสถาปัตยกรรมที่ได้รับการยอมรับ และนำไปใช้อย่างกว้างขวางแสดงได้ดังนี้

ภาพที่ 3



แสดงสถาปัตยกรรมฐานข้อมูลมาตรฐานของ ANSI และ ISO

จากภาพที่ 3 จะแบ่งระดับของรูปแบบข้อมูล และการเข้าถึงข้อมูลออกเป็น 3 ระดับคือ ระดับภายนอก (External) ซึ่งจะเก็บข้อมูลการมองเห็น และเข้าถึงข้อมูลของผู้ใช้แต่ละคนไว้ ระดับแนวคิด (conceptual) จะเก็บรูปแบบข้อมูลที่แทนด้วยแบบจำลองข้อมูล และระดับภายใน (Internal) จะเก็บรูปแบบการเก็บข้อมูลภายในฐานข้อมูลนั้น ๆ

จากสถาปัตยกรรม 3 ระดับของ RDBMS ในส่วนของระดับที่ 3 คือ Internal level เป็นส่วนที่เกี่ยวกับการเก็บข้อมูลในระดับ physical ซึ่งนอกจากจะมีการระบุชนิดของข้อมูลที่จัดเก็บแล้ว ยังมีการระบุถึงวิธีการในการเข้าถึงข้อมูล ซึ่งจะรวมไปถึงวิธีการในการจัดเก็บข้อมูลด้วย เช่นการสร้าง index ซึ่งสิ่งเหล่านี้เรียกว่ากลไกในการจัดเก็บ และเข้าถึงข้อมูล

สิ่งที่จะเป็นตัวกำหนดรูปแบบ และกลไกในการเข้าถึงข้อมูลในระบบจัดการฐานข้อมูลทั่วไปคือการสร้าง index ซึ่งในระบบจัดการฐานข้อมูลจะมีคำสั่งในการสร้าง index โดยผู้ใช้จะต้องกำหนด ชื่อของตารางข้อมูล ชื่อของคอลัมน์ ที่จะสร้าง index และระบุชนิดของ index

index คืออะไร และมีประโยชน์อย่างไร ?

index คือเนื้อที่ส่วนหนึ่งของฐานข้อมูลที่มีไว้เพื่อการค้นหาข้อมูลโดยเฉพาะ โดยข้อมูลที่เก็บใน index จะสอดคล้องกับข้อมูลของคอลัมน์หนึ่งหรือหลายคอลัมน์ของตารางที่กระทำ index นั้น และข้อมูลที่อยู่ใน index จะต้องสอดคล้องกับ logical address ของบรรทัดข้อมูลที่ตรงกันในตารางนั้นด้วยแสดงดังตารางที่ 13 ซึ่งคอลัมน์ที่สร้าง index สามารถแก้ไขเปลี่ยนแปลงได้ตามความเหมาะสม วิธีในการเข้าถึงข้อมูลของ index มีหลายวิธีด้วยกันเช่น hash,isam และ btree เป็นต้น ซึ่งรายละเอียดจะกล่าวในหัวข้อถัดไป

ตารางที่ 13

แสดงตัวอย่าง index

index บนคอลัมน์ location ของตาราง VENDOR

Location	Row
ชลบุรี	6
ระยอง	5
เชียงใหม่	3
เชียงใหม่	4
ภูเก็ต	1
ภูเก็ต	2

ตาราง VENDOR

Name	Location
สุชาติ	ภูเก็ต
บัณฑิต	ภูเก็ต
ภุชงค์	เชียงใหม่
ธวัชชัย	เชียงใหม่
สมชิด	ระยอง
สุกิจ	ชลบุรี

โดยทั่วไปการสร้าง index จะเป็นขั้นตอนที่กระทำหลังจากสร้างตารางข้อมูลที่ได้จากการออกแบบแล้วบน RDBMS คอลัมน์ที่จะนำมาสร้าง index นั้นส่วนมากจะนำมาจากคอลัมน์ที่เป็น primary key ของตารางนั้น ซึ่งอาจประกอบด้วยคอลัมน์เดียวหรือหลายคอลัมน์ก็ได้ หลังจากนั้นเมื่อมีการใช้งานตารางข้อมูลขึ้น ก็จะมาพิจารณาปรับเปลี่ยน index ให้เหมาะสมกับการใช้งาน

จุดประสงค์ของการสร้าง index เพื่อ *หลีกเลี่ยงการค้นหาข้อมูลทั้งตาราง , จำกัดจำนวนการค้นหาข้อมูลในตาราง และหลีกเลี่ยงการเรียงลำดับข้อมูล เป็นต้น* ตัวอย่างเช่น พิจารณา query ต่อไปนี้

```
SELECT NAME,LOCATION
FROM VENDOR
WHERE LOCATION = 'เชียงใหม่'
```

จากภาพที่ 13 RDBMS สามารถรู้ได้จากตาราง index ว่าบรรทัดข้อมูลที่ตรงกับเงื่อนไขใน query ข้างต้นคือ บรรทัดที่ 3 และ 4 ดังนั้น RDBMS จึงสามารถดึงข้อมูลที่ต้องการได้เลยไม่จำเป็นต้องไปค้นหาข้อมูลทั้งตาราง

ถึงแม้ว่าจะต้องมีการค้นหาข้อมูลในตาราง index ก่อน เพื่อให้ได้ที่อยู่ของบรรทัดข้อมูลที่ตรงกันกับในตารางข้อมูล แล้วจึงไปค้นหาข้อมูลจากตารางข้อมูลตามที่อยู่ที่ได้จากตาราง index แต่ผลรวมของการใช้งาน input/output จากการค้นหาข้อมูลใน index กับการค้นหาข้อมูลในตารางข้อมูลยังน้อยกว่าการค้นหาข้อมูลทั้งตาราง

- index สามารถจำกัดจำนวนการค้นหาข้อมูลได้ดังตัวอย่างต่อไปนี้

```
SELECT LOCATION
FROM VENDOR
```

จาก query ข้างต้นจะพบว่า RDBMS สามารถดึงข้อมูลจาก index ได้เลยโดยไม่ต้องไปยุ่งเกี่ยวกับตารางข้อมูล ซึ่งใน index จะมีขนาดเล็กกว่าตารางข้อมูลทำให้การใช้งาน input/output น้อยกว่าการกระทำบนตารางข้อมูลจึงเป็นการจำกัดจำนวนการค้นหาข้อมูล

- index ทำให้สามารถหลีกเลี่ยงการเลียงลำดับข้อมูลได้ดังตัวอย่างต่อไปนี้

```
SELECT NAME,LOCATION
FROM VENDOR
ORDER BY LOCATION
```

จาก query ข้างต้นจะเห็นว่าต้องการข้อมูลจากตาราง VENDOR โดยเรียงลำดับข้อมูลตามคอลัมน์ LOCATION จากตารางที่ 13 จะเห็นว่าเป็น order index โดยจะเลียงลำดับตาม LOCATION แต่ในตารางข้อมูลจริงคอลัมน์ LOCATION ไม่ได้มีการเลียงลำดับ RDBMS สามารถประมวลผลเพื่อให้ได้คำตอบได้ 2 วิธีคือ วิธีแรกคือ ดึงข้อมูลจากตาราง VENDOR และทำการเลียงลำดับข้อมูล ประการที่สองคือดึงข้อมูลจากตาราง VENDOR โดยใช้ index จาก LOCATION index ซึ่งจะเลียงลำดับดังนี้คือ บรรทัดที่ 6 , บรรทัดที่ 5 , บรรทัดที่ 3 , บรรทัดที่ 4 , บรรทัดที่ 1 และบรรทัดที่ 2 จะเห็นว่าข้อมูลที่ได้มีการเลียงลำดับของข้อมูลแล้วจึงไม่จำเป็นต้องนำมาเลียงลำดับอีก

ชนิดของกลไกในการเข้าข้อมูล

กลไกในการเข้าถึงข้อมูลที่ใช้กันแพร่หลายอยู่ทั่วไปมี 4 ชนิดคือ

1. Full table scan

Full table scan เป็นวิธีการในการเข้าถึงข้อมูลวิธีหนึ่ง โดยจะเข้าถึงข้อมูลเรียงลำดับไปเรื่อยๆ จนกว่าจะพบข้อมูลตามเงื่อนไขที่ผู้ใช้ระบบต้องการหรือหมดข้อมูล ไม่ต้องมีการกำหนดคอลัมน์ที่เป็นคีย์เพราะวิธีนี้ไม่มีคีย์ ถ้ามีการเพิ่มบรรทัดข้อมูลเข้ามาก็จะนำไปเก็บต่อไว้ท้ายสุดของแฟ้มข้อมูล

ตัวอย่างลักษณะการเก็บข้อมูลแบบ Full table scan ของระบบจัดการฐานข้อมูล INGRES ซึ่งใน INGRES เรียกว่า HEAP

ภาพที่ 4

	empno	name	age	salary
page 0	17	Shigio	29	28000.00
	9	Blumberg	33	32000.00
	26	Stover	38	35000.00
	1	Mandic	46	43000.00
page 1	18	Giller	47	46000.00
	10	Ming	23	22000.00
	27	Curry	34	32000.00
	2	Ross	50	55000.00
Page 2	19	McTigue	44	41000.00
	11	Robinson	64	80000.00
	28	Kay	41	38000.00
	3	Stein	44	40000.00

แสดงตัวอย่างการเก็บข้อมูลแบบ Heap

2.HASH

แฮชเป็นกลไกการในเข้าถึงข้อมูลที่สามารถเข้าถึงที่อยู่ของบรรทัดข้อมูลได้โดยตรง โดยการกำหนดค่าของคอลัมน์บางคอลัมน์ในบรรทัดข้อมูลนั้น ซึ่งโดยทั่วไปใน RDBMS จะมีตัวกำหนดที่อยู่ของแต่ละบรรทัดข้อมูล เช่น ใน ORACLE มี ROWID หรือใน INGRES มี TUPLE -

IDENTIFIERS สำหรับกำหนดที่อยู่ของบรรทัดข้อมูล ในวิธีแฮชจะมีแฮชฟังก์ชัน ซึ่งเมื่อนำค่าของคอลัมน์ที่ทำแฮชมาผ่านแฮชฟังก์ชั่นแล้วก็จะได้ค่าที่อยู่ของบรรทัดข้อมูล (hash address) ที่ตรงกับค่าของคอลัมน์ที่นำมาผ่านแฮชฟังก์ชั่น

ตัวอย่างลักษณะการเก็บข้อมูลแบบ HASH ของระบบจัดการฐานข้อมูล INGRES

ภาพที่ 5

	age	name	salary				
Page 0	50	Rose	55000.00				
	20	Smit	10000.00				
	30	Curan	30000.00				
	20	Sabel	21000.00				
Page 1							
Page 2	22	McShane	22000.00				
	32	Gregort	31000.00				
	42	Brodie	40000.00				
					Overflow Chain for Page 3		
Page 3	33	Blumberg	32000.00	----->	23	Ramos	30000.00
	43	Clark	40000.00		53	McTigue	41000.00
	23	Ming	22000.00				
	43	Kay	38000.00				

แสดงตัวอย่างการเก็บข้อมูลแบบ Hash

จากภาพที่ 5 สร้าง Hash บนคอลัมน์ age กำหนด Hash function ดังนี้ Page = Key MOD Main_pages โดย Main_pages เป็นจำนวน pages ทั้งหมดที่มีอยู่มีค่าเท่ากับ 10 เช่น

บรรทัดข้อมูลที่มีค่า name = Ross , age = 50 จะได้ว่า 50 MOD 10 = 0 จะเก็บข้อมูลไว้ที่ page 0

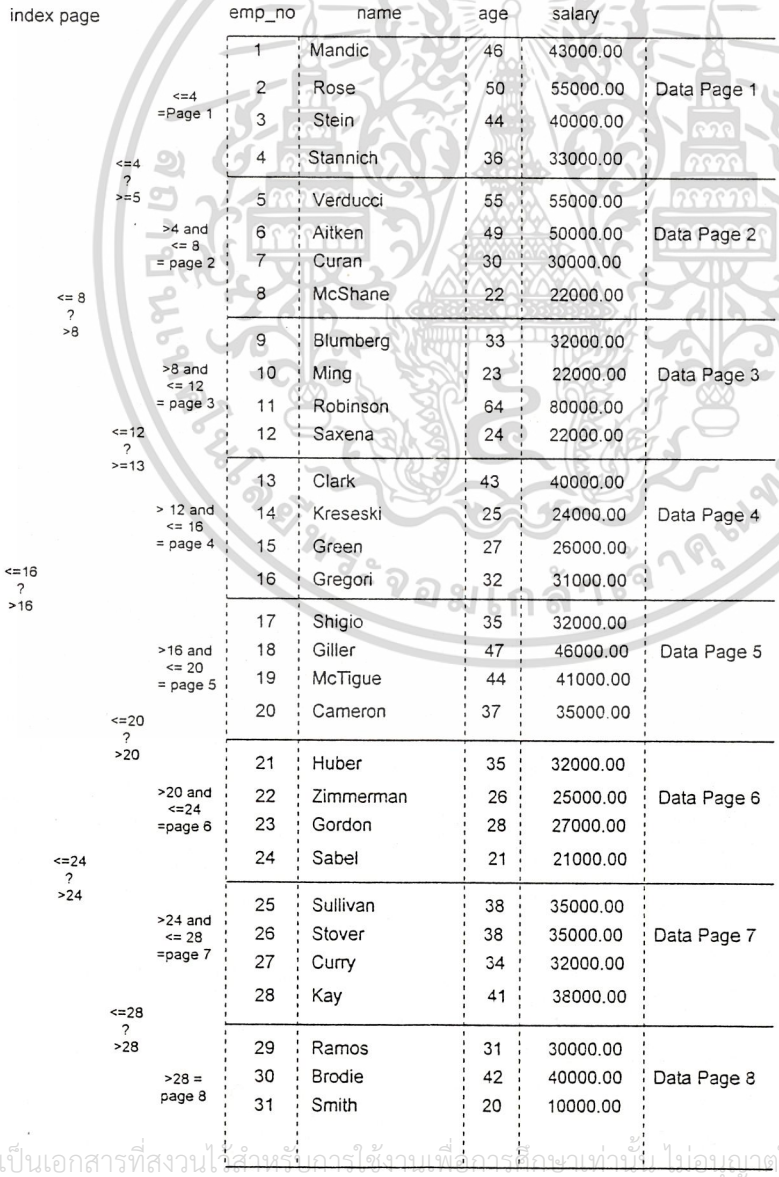
บรรทัดข้อมูลที่มีค่า name = Mcshane , age = 22 จะได้ว่า 22 MOD 10 = 2 จะเก็บข้อมูลไว้ที่ page 2

ในกรณีที่บาง page เต็มก็จะมี over flow page เช่น page ที่ 3

3. ISAM

เป็นกลไกในการเข้าถึงข้อมูลที่ใช้ตาราง index ในการเข้าถึงข้อมูล โดยที่ในตาราง index จะเก็บเฉพาะคอลัมน์ที่เป็นคีย์กับที่อยู่ของข้อมูลในตารางจริง ซึ่งการเก็บค่าของข้อมูลที่เป็นคีย์ที่เก็บในตาราง index จะแบ่งได้เป็น 2 ลักษณะคือ ลักษณะแรกจะเก็บค่าทั้งหมดทุกค่าที่มีอยู่เรียกว่า Dense index ลักษณะที่สองเก็บเฉพาะบางค่าเรียกว่า Sparse index ในบางครั้งสามารถสร้างตาราง index ขึ้นมาหลายชั้นได้ กล่าวคือตาราง index ในชั้นแรกจะเก็บที่อยู่ของข้อมูลในตาราง index ชั้นที่สอง เป็นเช่นนี้ต่อ ๆ ไปจนถึงชั้นสุดท้ายจะเก็บที่อยู่ของข้อมูลจริง ส่วนใหญ่นิยมสร้างตาราง index ไม่เกินสามชั้น

ภาพที่ 6



แสดงตัวอย่างการเก็บข้อมูลแบบ ISAM

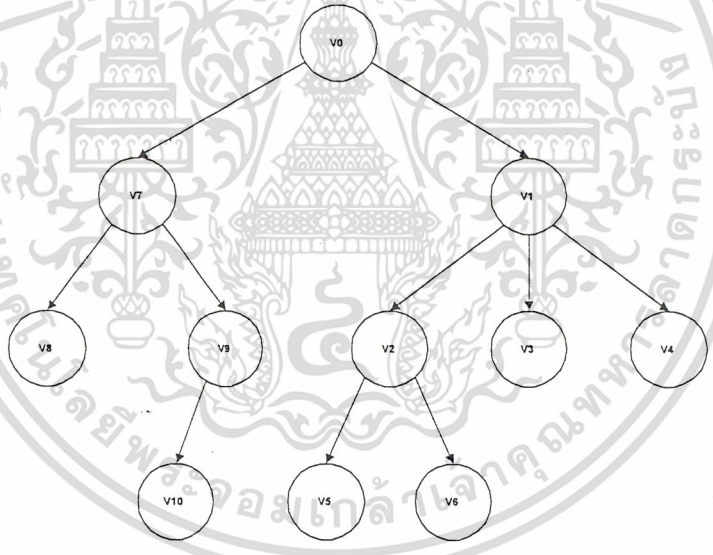
จากภาพที่ 6 จะใช้ static index ชี้ไปยัง static number ของ page หลักซึ่งเป็น static เช่นเดียวกัน และสร้าง sparse index ขึ้น ซึ่ง index จะประกอบด้วยช่วงของคีย์ และ pointer ชี้ไปยัง index page หรือ data page

4. BTREE

โครงสร้างข้อมูลแบบทรี มีลักษณะคล้ายต้นไม้ โดยจุดที่แตกกิ่งก้านจะเรียกว่า โหนด (node) ซึ่งจะเป็นที่เก็บข้อมูลไว้ส่วนการสร้าง tree นั้นจะทำจากบนลงล่างซึ่งจะตรงข้ามจากต้นไม้จริง

คุณสมบัติและโครงสร้างข้อมูลแบบทรี

ภาพที่ 7

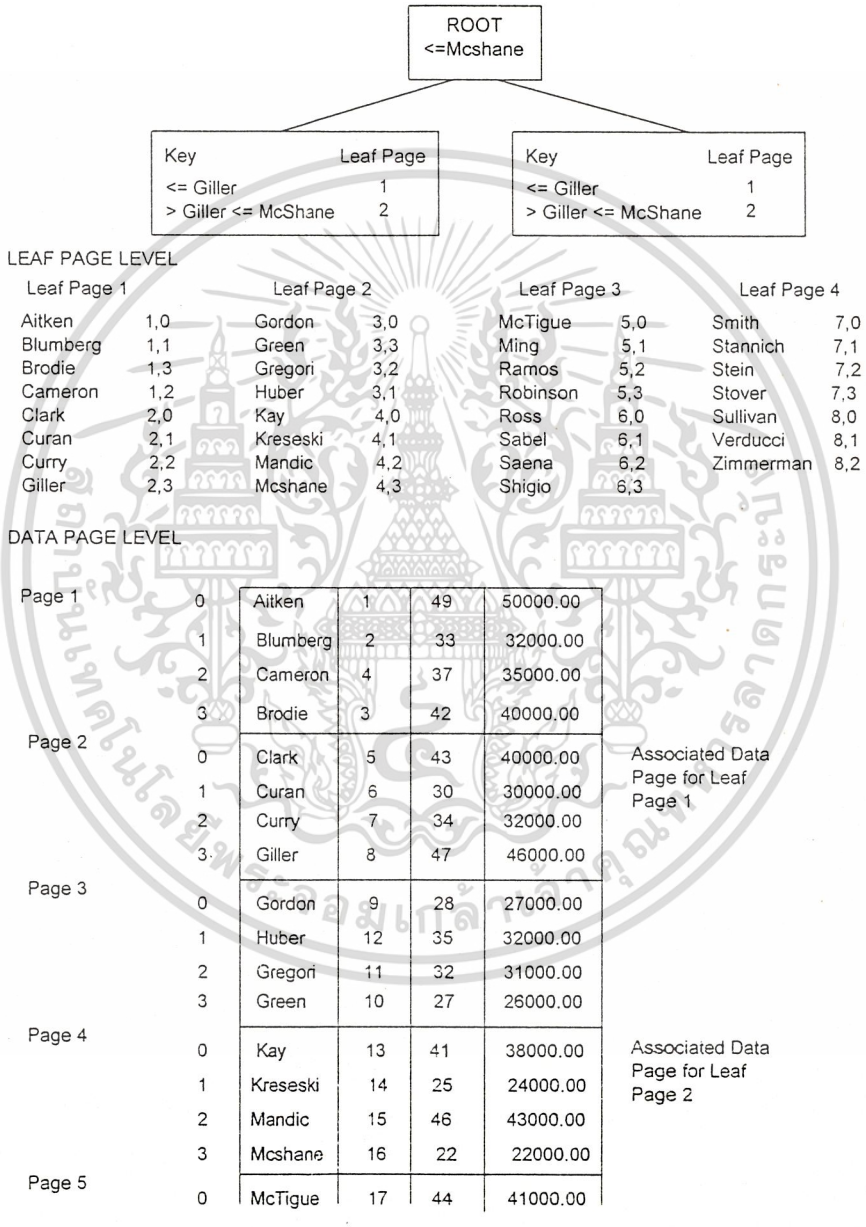


แสดงโครงสร้างข้อมูลแบบทรี

- มีโหนดอยู่ที่จุดยอด 1 โหนด เราจะเรียกว่า รูดโหนด (root node) เช่นในภาพที่ 7 ทุกรูป V0 จะเป็นรูดโหนดของทรี
- มีส่วนที่เรียกว่า ต้นไม้ย่อย (subtree) เช่นในรูปที่ 7 ทุกรูป V0 จะเป็นรูดโหนดของต้นไม้ย่อยของ V1 และ V7 โดยที่ V1 เป็นรูดโหนดของต้นไม้ย่อยของ V2 , V3 และ V4 ส่วน V9 เป็นรูดโหนดของ V10 เป็นต้น
- มีส่วนที่เป็นใบ (left node หรือ terminal node) คือจะมีจำนวนต้นไม้ย่อยเป็น 0 หรือบางครั้งเรียกว่า outdegree หรือ degree 0 ฉะนั้นอาจกล่าวได้ว่าดีกรีก็คือจำนวนต้นไม้ย่อยนั่นเองเช่นจากภาพที่ 7 จะมีโหนดใบได้แก่ V3 , V4 , V5 , V6 และ V10 โหนดที่มีดรีกรีเป็น 2 ได้แก่ V2 , V7 เป็นต้น

ตัวอย่างลักษณะการเก็บข้อมูลแบบ BTREE ของระบบจัดการฐานข้อมูล INGRES

ภาพที่ 8



แสดงตัวอย่างการเก็บข้อมูลแบบ Btree

จากรูป จะมีการสร้าง sparse index ที่ชี้ไปยัง leaf page จากนั้นในส่วนของ leaf page จะมีการสร้าง dense index ที่ประกอบด้วยคีย์ และที่อยู่ของบรรทัดข้อมูล

ภาษาที่ใช้งานบนฐานข้อมูลทั่วไป และภาษาที่ใช้ในการกำหนดรูปแบบ และกลไกในการเข้าถึงข้อมูลบน RDBMS

โดยทั่วไปผู้ใช้งานฐานข้อมูลรีเลชั่นแนลจะกระทำผ่านทางภาษาที่ใช้ในการจัดการฐานข้อมูล ซึ่งภาษาที่ใช้งานกันแพร่หลายอยู่ในปัจจุบันคือ ภาษา SQL (Structured query language) สามารถแบ่งประเภทตามการทำงานหลักๆ ได้เป็น 2 ประเภทคือ Data manipulation language และ Data definition language

1. Data manipulation language (DML)

เป็นภาษาที่ตั้งอยู่บนทฤษฎี relational algebra และ relational calculus ไวยากรณ์ของ DML จะช่วยในการกำหนดข้อมูลที่จะทำการเข้าถึง โดยจะกำหนดในรูปของตาราง , คอลัมน์ และบรรทัดข้อมูล ซึ่งมีคำสั่งหลัก 4 คำสั่งคือ SELECT , INSERT , UPDATE และ DELETE ดังตัวอย่างต่อไปนี้

คำสั่ง SELECT เป็นคำสั่งสำหรับระบุข้อมูลที่จะดึงขึ้นมาจากตารางในฐานข้อมูล

```
SELECT DEPT , SUM(SALARY)
FROM  EMPLOYEE
WHERE  LOCATION = 'New York'
GROUP BY  DEPT
HAVING SUM(SALARY) > 1000000.00
ORDER BY DEPT
```

คำสั่ง INSERT เป็นคำสั่งสำหรับเพิ่มบรรทัดข้อมูลลงในตารางฐานข้อมูล

```
INSERT INTO VENDOR
        (NAME , STATUS)
        VALUES ('Weigh-Up' , 'active')
```

คำสั่ง UPDATE เป็นคำสั่งสำหรับแก้ไขข้อมูลในคอลัมน์เดียว หรือหลายคอลัมน์ และในบรรทัดข้อมูลเดียว หรือหลายบรรทัด

```
UPDATE EAST-COAST-VENDOR
        SET STATUS = 'active'
        WHERE LOCATION = 'New York'
```

คำสั่ง DELETE เป็นคำสั่งสำหรับลบข้อมูลบรรทัดข้อมูลเดียว หรือหลายบรรทัด ออกจากตารางข้อมูลในฐานข้อมูล

```
DELETE FROM EAST-COAST-VENDOR
        WHERE STATUS IS NULL
```

2. Data definition language (DDL)

เป็นภาษาที่ใช้ในการสร้างตารางข้อมูล และลบตารางข้อมูลในฐานข้อมูล หรือสร้าง และลบความสัมพันธ์ระหว่างข้อมูล นอกจากนั้นยังใช้ในการสร้าง index และแก้ไขความสัมพันธ์ในตารางข้อมูลด้วย
ตัวอย่างภาษาที่ใช้ในการสร้างตารางข้อมูล

```
CREATE TABLE VENDOR (NAME CHAR(20) NOT NULL, LOCATION CHAR(15))
```

ตัวอย่างภาษาที่ใช้ในการสร้าง index

```
CREATE UNIQUE INDEX XVENDOR-NAME ON VENDOR(NAME)
```

Query optimization

Query optimization เป็นการเลือกวิธีเข้าถึงข้อมูลให้เร็วที่สุดโดยจะมี Query optimizer เอาไว้เป็นตัวพิจารณาในการเข้าถึงข้อมูลจริงๆ ในระดับ physical ซึ่งเราอาจมองได้ว่า Query optimizer

เป็น Expert system ตัวหนึ่งสำหรับช่วยในการวิเคราะห์เส้นทางในการเข้าถึงข้อมูล หลักการทำงานทั่วไปของ Query optimizer คือ จะรับ parse tree มาจาก Query parse ซึ่งทำหน้าที่ในการตรวจสอบไวยากรณ์ของภาษาฐานข้อมูล แล้วสร้างเป็น algebra tree ส่งมาให้ Query optimizer ทำการวิเคราะห์โดยใช้กฎที่มีอยู่ร่วมกับข้อมูลจากตารางสถิติที่มีอยู่ในฐานข้อมูล ซึ่งส่วนมากจะนำมาจาก system catalog เช่น จำนวนของบรรทัดข้อมูล , จำนวน และชนิดของ index ที่มีอยู่เป็นต้น และบางส่วนได้มาจากคำนวณจากข้อมูลจริงที่มีอยู่แล้วนำมาเก็บไว้ในตารางสถิติเช่น จำนวนของ unique value และค่าการกระจายของข้อมูลของค่าในคอลัมน์ในตารางใดๆเป็นต้น หลังจากนั้นจะคำนวณเวลาที่ใช้ CPU (CPU time) และประมาณการใช้งาน Input/Output (I/O operation) ของแต่ละเส้นทาง ซึ่งจะนำมาช่วย Query optimizer ในการตัดสินใจเลือกเส้นทางการเข้าถึงข้อมูล

จะเห็นว่าผู้ใช้ไม่จำเป็นต้องพิจารณาถึงเส้นทางการเข้าถึงข้อมูล (access path) เพราะใน RDBMS ส่วนมากจะมี Query Optimizer ช่วยในการวิเคราะห์เส้นทางการเข้าถึงข้อมูลจากเส้นทางต่างๆที่ตรงกับความต้องการใช้งานฐานข้อมูล และจะเลือกเส้นทางที่ดีที่สุด แต่ถ้าจะให้ได้ประสิทธิภาพสูงสุดจากการตัดสินใจของ Query Optimizer ยังขึ้นอยู่กับปัจจัยที่สำคัญอีกอย่างหนึ่ง คือ รูปแบบ และกลไกในการเข้าถึงข้อมูล ซึ่งเป็นหน้าที่ของผู้ที่ออกแบบ และดูแลฐานข้อมูลจะต้องพิจารณาเลือกรูปแบบ และกลไกในการเข้าถึงข้อมูลให้สอดคล้องกับการทำงานของ Query Optimizer ซึ่งถ้าเลือกรูปแบบ และกลไกในการเข้าถึงข้อมูลไม่สอดคล้องกับการทำงานของ Query Optimizer แล้ว สิ่งที่เลือก และสร้างขึ้นมานอกจากจะไม่ถูกนำมาใช้งานแล้วยังอาจทำให้เพิ่ม overhead ทำให้การค้นหา และเข้าถึงข้อมูลช้าลงด้วย ตัวอย่างเช่น เราจะมั่นใจได้อย่างไรว่า index ที่เราสร้างขึ้นมาบนตารางถูกนำไปใช้งานจริงๆ เพราะถ้าเราสร้าง index แล้วไม่ได้นำไปใช้งานนอกจากจะเปลืองเนื้อที่ในการเก็บข้อมูลแล้ว ถ้าตารางนั้นมีการค้นหาข้อมูลแบบทั้งตาราง (full table scan) จะทำให้การค้นหาข้อมูลช้ากว่ากระทำบนตารางที่ไม่มีการสร้าง index

จากจุดนี้เองจึงต้องอาศัยความชำนาญของผู้ที่ออกแบบ และดูแลฐานข้อมูล ร่วมกับความเข้าใจในการทำงานของ Query Optimizer และความเข้าใจในโครงสร้างการเก็บข้อมูลระดับ Physical ของ RDBMS ที่ใช้อยู่ มาช่วยพิจารณาในการกำหนดรูปแบบ และกลไกในการเข้าถึงข้อมูล

<u>ปัจจัยที่ใช้พิจารณาในการกำหนดรูปแบบ และกลไกในการเข้าถึงข้อมูล</u>

ปัจจัยที่จะนำมาพิจารณาในการศึกษาการกำหนดรูปแบบ และกลไกในการเข้าถึงข้อมูลแบ่งได้เป็น 4 ข้อดังนี้

1. พิจารณาจากลักษณะความต้องการใช้งานฐานข้อมูล

ความต้องการใช้งานฐานข้อมูลที่แตกต่างกัน ย่อมมีความเหมาะสมกับรูปแบบ และกลไกในการเข้าถึงข้อมูลที่แตกต่างกัน พิจารณาความต้องการใช้งานฐานข้อมูลได้จาก SQL query ที่ใช้อยู่ประจำ โดยความเหมาะสมจะพิจารณาจากภายใต้รูปแบบ และกลไกในการเข้าถึงข้อมูลที่แตกต่างกันบนสภาวะแวดล้อมเดียวกัน จะพิจารณาถึงเวลาที่ใช้ CPU (CPU time) น้อยที่สุด และการใช้งาน Input\Output น้อยที่สุด ตัวอย่างเช่น พิจารณาจาก query ต่อไปนี้

```
SELECT R.A, R.D
FROM R, S
WHERE R.B = S.C ;
```

จาก query ข้างต้นถ้าสร้าง index บนคอลัมน์ B บนตาราง R และสร้าง index บนคอลัมน์ C บนตาราง S จะสามารถช่วยเพิ่มประสิทธิภาพในการเข้าถึงข้อมูลได้ แต่ถ้า query เป็นดังต่อไปนี้

```
SELECT R.A, R.D
FROM R, S
WHERE R.B = S.C
AND S.D = 5
AND R.E = 6;
```

จะต้องเปลี่ยนคอลัมน์ที่จะสร้าง index ใหม่ เพราะว่ามีเงื่อนไขในการดึงข้อมูลเข้ามาเกี่ยวข้อง ซึ่งโดยทั่วไปแล้วเงื่อนไข S.D หรือ R.E จะถูกกระทำก่อนที่จะเชื่อมตาราง R กับ S สำหรับขั้นตอนและวิธีทำนั้นขึ้นอยู่กับการทำงานของ Query Optimizer ของ RDBMS แต่ละตัว

2. พิจารณาจากการทำงานของ Query Optimizer ใน RDBMS ที่ใช้อยู่

ผู้ที่ออกแบบ และดูแลฐานข้อมูลต้องศึกษาให้เข้าใจถึงการทำงานของ Query Optimizer โดยพิจารณาได้จากคู่มือการใช้งานของ RDBMS ที่ใช้อยู่ ร่วมกับการทดลองประมวลผล Query และพิจารณาการทำงานของ Query นั้นๆ ส่วนใหญ่ใน RDBMS ที่ใช้กันแพร่หลายอยู่ทั่วไป จะมีฟังก์ชันให้เราสามารถดูการทำงานของ Query Optimizer ได้ซึ่งทำให้เราสามารถพิจารณาเส้นทาง

การทำงานของ Query ได้ เช่นเราต้องการจะตรวจสอบว่า index ที่เราสร้างขึ้นถูกนำไปใช้งานจริงหรือไม่ หรือเงื่อนไขในการดึงข้อมูลเงื่อนไขใดถูกเลือกมากระทำก่อน

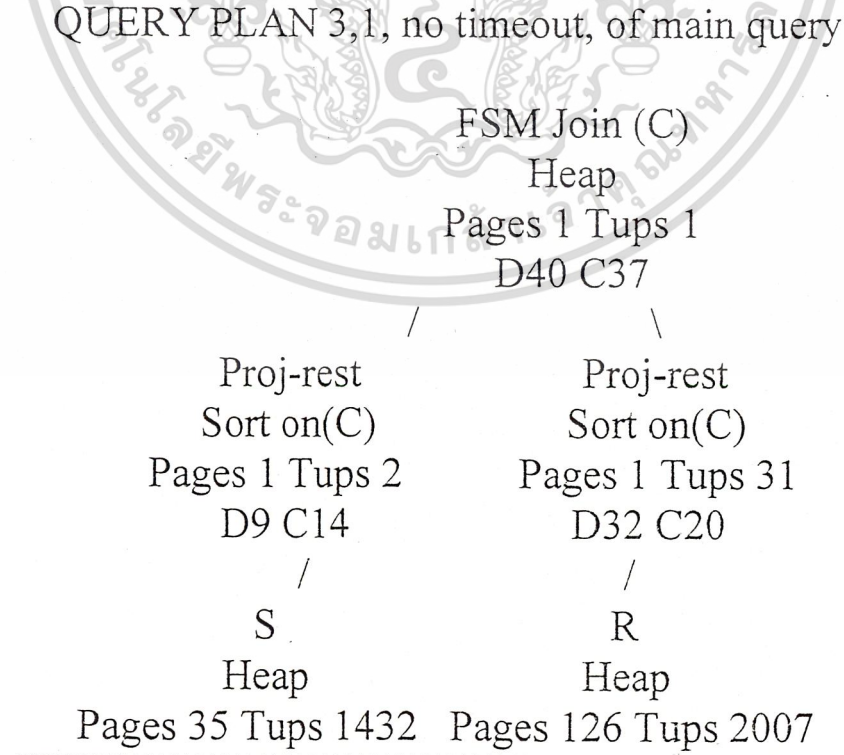
ตัวอย่าง พิจารณาจาก Query ต่อไปนี้

```
SELECT R.A, R.D
FROM R, S
WHERE R.C = S.C
AND S.D = 5
AND R.E = 6;
```

จาก Query เราต้องการจะตรวจสอบว่าระหว่างเงื่อนไข S.D และ R.E เงื่อนไขใดที่จะถูกเลือกมาประมวลผล หรือประมวลผลก่อน โดยจะพิจารณาจากเส้นทางที่ Query Optimizer เลือกประมวลผล ซึ่งแสดงได้ดังตัวอย่างต่อไปนี้

ตัวอย่างแสดง แผนในการประมวลผล Query (Query executetion plan) ของ INGRES

ภาพที่ 9



****

แสดง Query executetion plan

จากตัวอย่างแสดงการทำงานของ Query ในรูป tree จากการพิจารณาจะพบว่าเงื่อนไข S.D จะถูกเลือกนำมาค้นหาข้อมูลก่อน เมื่อได้บรรทัดมูลที่ตรงกับเงื่อนไขแล้วทำให้สมารถรู้ค่าของ S.C ด้วย จากนั้นจะเลียงลำดับข้อมูล S.C และ R.C และเชื่อมตารางโดยใช้คอลัมน์ทั้งสองที่เลียงลำดับแล้ว เมื่อได้บรรทัดข้อมูลจากการเชื่อมบนตาราง R แล้ว ทำให้สามารถรู้ค่าของ R.E ด้วย จึงสามารถกรองบรรทัดข้อมูลตามเงื่อนไขของ R.E ได้เลย จากทั้งหมดข้างต้นนี้สามารถวิเคราะห์ได้ว่าการค้นหาตำแหน่งบรรทัดข้อมูลในตารางจะกระทำเพียงสองช่วง คือช่วงแรกค้นหาตามเงื่อนไข S.D และช่วงที่สองค้นหา R.C เพื่อเชื่อมตารางกับ S.C ดังนั้นเราควรสร้าง index เพื่อช่วยในการค้นหาข้อมูลบนสองคอลัมน์นี้คือ S.D และ R.C สำหรับรายละเอียดการทำงานของแผนการประมวลผล Query ของ INGRES จะมีกล่าวไว้ในบทที่ 4

3. พิจารณาจากปริมาณของข้อมูลที่เข้ามาเกี่ยวข้องกับขั้นตอนต่างๆในการประมวลผล query

โดยจะพิจารณาได้จากตารางสถิติของข้อมูลที่มีอยู่ในฐานข้อมูล และ System Catalog ประกอบกับลักษณะความต้องการในการใช้งานฐานข้อมูลซึ่งพิจารณาได้จาก SQL query ที่ใช้อยู่ประจำ ตัวอย่างเช่น จากรายละเอียดการทำงานของแผนการประมวลผล Query ในข้อที่ผ่านมา มีข้อที่หน้าสังเกตว่าเหตุใด Query Optimizer จึงเลือกทำเงื่อนไข S.D ก่อน จากการตรวจสอบพบว่าจำนวนบรรทัดข้อมูลที่ตรงกับเงื่อนไข S.D มีค่าน้อยกว่าจำนวนบรรทัดข้อมูลที่ตรงกับเงื่อนไข R.E แสดงว่า Query Optimizer จะเลือกทำเงื่อนไขที่มีค่าบรรทัดข้อมูลที่ตรงกับเงื่อนไขน้อยกว่าก่อนเพื่อลดจำนวนการค้นหาข้อมูล ดังนั้นจะเห็นได้ว่า ปริมาณของข้อมูลมีผลต่อการเลือกเส้นทางการทำงานของ Query optimizer

4. พิจารณาจากโครงสร้างการเก็บข้อมูลระดับ physical ของ RDBMS ที่ใช้อยู่

ศึกษาได้จากคู่มือการใช้งานของ RDBMS ที่ใช้อยู่ ส่วนมากวิธีที่ใช้แพร่หลายกันอยู่มี 4 วิธีด้วยกันคือ full table scan , hash , isam และ btree ซึ่งแต่ละวิธีจะมีความเหมาะสมกับความต้องการใช้งานฐานข้อมูลที่แตกต่างกัน รายละเอียดของแต่ละวิธีจะมีกล่าวในตอนท้ายของบทนี้

<u>กฎเกณฑ์และหลักการในการกำหนดรูปแบบ และกลไกในการเข้าถึงข้อมูล</u>

มีคำถามที่เราจะต้องนำมาพิจารณาเกี่ยวกับการสร้าง index 3 ข้อคือ

1. ควรจะสร้าง index บนตารางข้อมูลใด ?
2. ควรจะสร้าง index บนคอลัมน์ใด ?
3. index ที่สร้างควรใช้วิธีใด ?

1. เราควรจะสร้าง index บนตารางข้อมูลใดบ้าง ?

โดยทั่วไปจะสร้าง index กับตารางที่มีขนาดตั้งแต่กลางไปจนถึงมีขนาดใหญ่ เพราะใน RDBMS ทั่วไป ถ้าการค้นหาข้อมูลกระทำกับตารางที่มีขนาดเล็กแล้ว (น้อยกว่าหรือเท่ากับ 6 physical block) การค้นหาข้อมูลแบบทั้งตารางจะทำได้ดีโดยไม่จำเป็นต้องใช้ index [13] และตารางที่จะสร้าง index ควรมีคุณสมบัติดังต่อไปนี้

1.1 ใน Query มีการอ้างถึงตารางและมีเงื่อนไขในการเรียกค้นข้อมูลจากตาราง โดยที่อัตราส่วนของ ข้อมูลที่ค้นหาได้ตามเงื่อนไข/ข้อมูลทั้งหมด มีค่าน้อย(น้อยกว่า 20%) [13]
ตัวอย่างเช่น

```
SELECT STD_ID,NAME
FROM STUDENT
WHERE CLASS = '2';
```

จาก Query ข้างต้น ถ้าในตาราง STUDENT มีจำนวนบรรทัดข้อมูลที่คอลัมน์ CLASS = '2' น้อยกว่า 20 % ของข้อมูลทั้งหมดแล้วควรสร้าง index บนตาราง STUDENT บนคอลัมน์ CLASS

1.2 หลีกเลี่ยงการเข้าถึงข้อมูลทั้งตาราง ในกรณีข้อมูลที่ต้องการเป็นเพียง subset ของคอลัมน์ของข้อมูลทั้งหมด [18] ตัวอย่างเช่น
ตาราง STUDENT มีส่วนประกอบดังนี้

STD_NO | NAME | ADDRESS | GRADE | CLASS | STATUS | SEX

จากตารางถ้ามี query ดังนี้

```
SELECT NAME
FROM STUDENT
```

จาก Query ข้างต้นจะเห็นว่ามีการอ้างถึงข้อมูลในตาราง STUDENT เพียงคอลัมน์เดียวคือ NAME ดังนั้นถ้าเราแยกคอลัมน์ NAME ออกมาจากตาราง STUDENT ได้โดยการสร้าง index บนคอลัมน์ NAME แล้ว การทำงานของ Query นี้ก็จะกระทำแค่ตาราง index เท่านั้น ซึ่งมีขนาดเล็กลงจากตารางเดิมทำให้การเข้าถึงข้อมูลทำได้เร็วขึ้น

1.3 ในกรณีที่ผู้ใช้ระบบมีความต้องการข้อมูลจากตารางในลักษณะที่เรียงลำดับ และการเลียงลำดับกระทำบนคอลัมน์ที่อัตราส่วนของ ข้อมูลที่ค้นหาได้/ข้อมูลทั้งหมด มีค่าน้อย(น้อยกว่า 20%) [18] ตัวอย่างเช่น

```
SELECT STD_ID , NAME , GRADE
FROM STUDENT
ORDER BY GRADE
WHERE GRADE > 2.5
```

จาก Query ข้างต้นจะเห็นว่าต้องการเรียงลำดับข้อมูลในคอลัมน์ GRADE จึงควรสร้าง index แบบเรียงลำดับ(order index) บนตาราง STUDENT บนคอลัมน์ GRADE

1.4 ถึงแม้ว่าการสร้าง index จะช่วยให้การค้นหาข้อมูลทำได้เร็วขึ้น แต่ก็จะทำให้การประมวลเหล่านี้ช้าลงคือ การแก้ไข เพิ่มเติม และลบข้อมูลที่เกี่ยวข้องกับคอลัมน์ที่ทำ index , การ recover,reorganization และ backup ตารางที่สร้าง index ดังนั้นตารางที่จะสร้าง index จึงควรหลีกเลี่ยงการประมวลผลเหล่านี้หรือไม่ควรมีการประมวลผลเหล่านี้บ่อย [18]

2. เราควรจะสร้าง index บนคอลัมน์ใด ?

สิ่งสำคัญที่จะมาเป็นตัวกำหนดคอลัมน์ที่จะสร้าง index คือรูปแบบ และลักษณะความต้องการใช้งานฐานข้อมูลของผู้ใช้ ซึ่งจะพิจารณาได้จาก Query ที่ใช้งานอยู่กับฐานข้อมูลนั้น เนื่องจาก Query ที่ใช้งานมีจำนวนไม่จำกัด และมีรูปแบบที่แตกต่างกัน จึงแบ่ง Query ออกเป็นกลุ่ม ซึ่งภายในแต่ละกลุ่มจะมีคุณสมบัติบางประการเหมือนกัน โดยคุณสมบัติเหล่านี้จะนำไปเป็นตัวกำหนดการเลือกคอลัมน์ที่จะสร้าง index และชนิดของ index ซึ่งแสดงได้ดังนี้

2.1 มีเงื่อนไขเท่ากับ

ตัวอย่างเช่น

```
SELECT NAME
FROM STUDENT
WHERE CLASS = '4'
```

จาก query จะเห็นว่ามีเงื่อนไขในการค้นหาข้อมูลเป็นแบบเท่ากับ ดังนั้นควรสร้าง index บนคอลัมน์ CLASS ซึ่งจะช่วยลดการค้นหาข้อมูล แต่ทั้งนี้จะต้องพิจารณาด้วยว่า จำนวนบรรทัดข้อมูลที่ตรงกับเงื่อนไขควรจะมีค่า อัตราส่วนของ ข้อมูลที่ค้นหาได้ตามเงื่อนไข/ข้อมูลทั้งหมด มีค่าน้อย(น้อยกว่า 20%) [13] แต่ถ้ามีเงื่อนไขมากกว่าหนึ่งเงื่อนไข จะต้องศึกษา และเปรียบเทียบการใช้งาน index ของ Query Optimizer ระหว่างการสร้าง composite index กับ การสร้าง single-column index หลายตัว

ตัวอย่างเช่น

```
SELECT NAME,ADDRESS,SALARY
FROM EMPLOYEE
WHERE DEPT = 'RESEARCH'
AND SALARY = 5000
```

จาก Query ข้างต้นสามารถสร้าง index ได้หลายแบบดังนี้

- สร้าง index บนคอลัมน์ DEPT
- สร้าง index บนคอลัมน์ SALARY
- สร้าง index บนคอลัมน์ DEPT และสร้าง index บนคอลัมน์ SALARY
- สร้าง index บนคอลัมน์ DEPT,SALARY
- สร้าง index บนคอลัมน์ SALARY,DEPT
- สร้าง index บนคอลัมน์ DEPT,SALARY,NAME

จากทางเลือกข้างต้นทุกทางล้วนช่วยให้การเข้าถึงข้อมูลเร็วขึ้น แต่เราจะต้องพิจารณาว่าข้อใดคือทางเลือกที่ดีที่สุด ซึ่งเราแบ่งการพิจารณาได้เป็น 3 ข้อ

- จากทางเลือกที่ 3 จะต้องพิจารณาว่า RDBMS สามารถใช้ index ทั้งคู่และนำผลลัพท์มา join กันได้หรือไม่ หรือ RDBMS พยายามจะใช้ index เพียงตัวเดียว และตัวไหนเป็นทางเลือกที่ดีที่สุด

- จากทางเลือกที่ 4 , 5 จะต้องพิจารณาถึงจำนวนวนบรรทัดข้อมูลที่ตรงกับเงื่อนไขของแต่ละคอลัมน์

- จากทางเลือกที่ 6 จะต้องพิจารณาว่าเป็นการหลีกเลี่ยงการเข้าถึงข้อมูลทั้งตารางหรือไม่ คือจะพิจารณาว่าคอลัมน์ที่ใช้งานใน Query เป็น subset ของคอลัมน์ในตารางข้อมูลหรือไม่ และลำดับก่อนหลังของคอลัมน์ที่สร้าง index ควรเป็นอย่างไร

เมื่อทำการพิจารณาเปรียบเทียบภายในของแต่ละวิธีเพื่อได้ทางเลือกที่ดีที่สุดของแต่ละวิธีแล้ว จะต้องเปรียบเทียบระหว่างวิธีด้วยว่าวิธีใดเป็นวิธีที่ดีที่สุด

ในกรณีสร้าง index หนึ่งตัวบนหลายๆ คอลัมน์ (composite index) ต้องคำนึงถึงความสัมพันธ์ของลำดับก่อนหลังของคอลัมน์ที่จะนำมาสร้าง index กับการใช้งาน index ของ query ที่เกี่ยวข้องกับ index นั้น ซึ่งในบาง query อาจมีการอ้างถึงแค่บางคอลัมน์ที่สร้าง index ดังแสดงในอย่างต่อไปนี้

สมมุติว่าผู้ใช้สร้าง index ด้วยภาษา SQL ดังนี้

```
CREATE INDEX I2
ON S(SNAME,CITY,STATUS) ASC;
```

ใช้เมื่อมีคำถามเกี่ยวกับ 3 คอลัมน์นี้บ่อยๆ แต่ถ้าต้องการติดต่อเฉพาะบางคอลัมน์ใน 3 คอลัมน์นี้ เปรียบเทียบความเร็วได้ดังนี้ [13]

sanme,city,status	เร็วที่สุด
sname,city	เร็วมาก
sanme	เร็วมาก
city	เร็ว
status	พอใช้

2.2 มีเงื่อนไขไม่เท่ากับ

ตัวอย่างเช่น

```
SELECT NAME
FROM STUDENT
WHERE CLASS != '4'
```

จาก query จะเห็นว่ามีเงื่อนไขในการค้นหาข้อมูลเป็นแบบไม่เท่ากับ การทำงานของ query นี้จะต้องมีการค้นหาข้อมูลทั้งตาราง ดังนั้นจึงไม่ควรสร้าง index บนตารางนี้เพราะจะทำให้การค้นหาข้อมูลแบบทั้งตารางทำได้ช้าลง [18]

2.3 มีเงื่อนไขเป็นช่วง

ตัวอย่างเช่น

```
SELECT NAME
FROM STUDENT
WHERE CLASS > '4'
```

การพิจารณากำหนดคอลัมน์ที่จะสร้าง index ใช้หลักการเดียวกันกับการพิจารณาเงื่อนไขเท่ากับในข้อที่หนึ่ง

2.4 มีเงื่อนไขที่มีการอ้างถึงเพียงบางส่วนของค่าในคอลัมน์ในทางซ้ายมือ

ตัวอย่างเช่น

```
SELECT NAME
FROM STUDENT
WHERE NAME LIKE 'S%'
```

การพิจารณากำหนดคอลัมน์ที่จะสร้าง index ใช้หลักการเดียวกันกับการพิจารณาเงื่อนไขเท่ากับในข้อที่หนึ่ง

2.5 มีการทำงานที่เกี่ยวกับการเรียงลำดับข้อมูล

ตัวอย่างเช่น

```
SELECT STD_ID, NAME
FROM STUDENT
ORDER BY STD_ID
```

จาก query ข้างต้นจะเห็นว่ามีความต้องการข้อมูลในลักษณะเรียงลำดับ ดังนั้นควรสร้าง index แบบเรียงลำดับบนคอลัมน์ STD_ID ซึ่งคำสั่ง SQL ที่จะต้องมีการเรียงลำดับข้อมูลอีกคือ GROUP BY ,DISTINCT [13]

2.6 มีการใช้งานตารางหลายตารางร่วมกัน

ในกรณีที่มีการ join กันของตารางข้อมูล ถ้าเป็นการ join กันแบบไม่มีเงื่อนไขแล้วควรสร้าง index บนคีย์ที่ใช้ join กัน ตัวอย่างเช่น

```
SELECT  STD_ID,NAME,SUBJ
FROM  STUDENT,SUBJECT
WHERE STD_ID.STUDENT = STD_ID.SUBJECT
```

จาก query ตัวอย่างควรสร้าง index บน คอลัมน์ STD_ID บนตาราง STUDENT และ ตาราง SUBJECT [13] แต่ถ้ามีเงื่อนไขในการ join ตารางด้วย ดังตัวอย่างต่อไปนี้

```
SELECT  STD_ID,NAME,SUBJ
FROM  STUDENT,SUBJECT
WHERE STD_ID.STUDENT = STD_ID.SUBJECT
AND STUDENT.CLASS = '2'
AND SUBJECT.CODE = '1'
```

จะต้องพิจารณาถึงการทำงานของ Query optimizer ประกอบด้วย ซึ่งใน RDBMS ส่วนมาก Query optimizer จะประมวลผลในส่วนของเงื่อนไขก่อนในที่นี้คือ คอลัมน์ CLASS = '2' หรือ คอลัมน์ CODE = '1' ก่อน ซึ่งการที่จะเลือกกระทำเงื่อนไขใดนั้น Query optimizer จะพิจารณาจาก จำนวนวนของบรรทัดข้อมูลที่ตรงกับเงื่อนไขนั้น โดยจะเลือกทำเงื่อนไขที่จำนวนของบรรทัดข้อมูลที่ได้น้อยที่สุดก่อน เพื่อที่จะลดจำนวนบรรทัดข้อมูลที่จะนำมาใช้ในการ join ตาราง

3. index ที่สร้างควรใช้วิธีใด ?

ดังที่ได้กล่าวในตอนต้นถึงกลไกในการเข้าถึงข้อมูลที่ใช้กันแพร่หลายมีอยู่ 4 วิธีคือ Scan , Hash , Btree , Isam พิจารณากฎเกณฑ์ในการเลือกชนิดของ index ได้ดังนี้

สแกน (SCAN)

การ จะใช้วิธีสแกนเมื่อตารางควรมีลักษณะดังต่อไปนี้

-ตารางมีขนาดเล็ก (ประมาณตั้งแต่ 6 Physical block ลงมา) [13]

-ตารางมีขนาดกลางหรือขนาดใหญ่(มากกว่า 6 physical block) และมีอัตราส่วนของ ข้อมูลที่ค้นหาได้/ข้อมูลทั้งหมด มีค่าสูง(ตั้งแต่ 20% ขึ้นไป) [13]

-ข้อมูลในตารางมีความสำคัญน้อย และไม่ต้องการเปลืองเนื้อที่ในการเก็บข้อมูล

แฮช (HASH)

การที่จะพิจารณาว่าควรจะสร้าง hash บนคอลัมน์ใดจะพิจารณาจากสองทางคือ

- ทางแรกพิจารณาจาก ลักษณะความต้องการใช้งานฐานข้อมูลซึ่งดูได้จาก query ซึ่งควรมีคุณสมบัติต่อไปนี้ [18]

- มีเงื่อนไขเท่ากับ
- ไม่มีเงื่อนไขเป็นช่วง
- ไม่มีเงื่อนไขที่มีการอ้างถึงเพียงบางส่วนของค่าในคอลัมน์
- คอลัมน์ที่เป็นค่าของแฮชคีย์จะต้องไม่ถูกแก้ไขบ่อย
- การติดต่อกับบรรทัดข้อมูลที่ต้องการเป็นลักษณะสุ่ม

- ทางที่สองพิจารณาจาก ลักษณะของข้อมูลของคอลัมน์ และตารางที่จะสร้าง hash ซึ่งควรมีคุณสมบัติดังนี้ [18]

- โดยทั่วไปจะทำแฮชกับตารางที่มีขนาดตั้งแต่กลางไปจนถึงมีขนาดใหญ่ (มากกว่า 6 physical block)
- ค่าของคอลัมน์ที่จะใช้ในการสร้างแฮช ต้องเป็นค่าไม่ต่อเนื่อง
- ควรเลือกแฮชคีย์ และขั้นตอนวิธีแฮชที่มีการกระจายการติดต่อข้อมูล ซึ่งจะทำให้ลดปัญหาการที่มีคีย์คนละตัวแต่แฮชแล้วได้ตำแหน่งข้อมูลเดียวกัน (collision)

บีทรี (BTREE)

การที่จะพิจารณาว่าควรจะสร้างบีทรีบนคอลัมน์ใดจะพิจารณาจากสองทางคือ

- ทางแรกพิจารณาจากลักษณะความต้องการใช้งานฐานข้อมูลซึ่งดูได้จาก query ซึ่งควรมีคุณสมบัติต่อไปนี้ [13]

- เงื่อนไขเท่ากับ หรือมีเงื่อนไขเป็นช่วง
- มีเงื่อนไขที่มีการอ้างถึงเพียงบางส่วนของค่าในคอลัมน์ที่สร้าง index
- ในกรณีที่สร้าง index หนึ่งตัวบนหลายคอลัมน์ แล้วมีการอ้างถึง index เพียงบางคอลัมน์ทางซ้ายของ index

- ทางที่สองพิจารณาจากลักษณะของข้อมูลของคอลัมน์. และตารางที่จะสร้างบีทรี ซึ่งควรมีคุณสมบัติดังนี้ [13]

- โดยทั่วไปจะทำบีทรีกับตารางที่มีขนาดตั้งแต่กลางไปจนถึงมีขนาดใหญ่ (มากกว่า 6 physical block) ไม่เหมาะกับตารางที่มีขนาดเล็กและข้อมูลค่อนข้างคงที่
- ไม่เหมาะกับตารางที่มีการใช้งานพร้อมๆ กันหลายคน

ไอแซม (ISAM)

การที่จะพิจารณาว่าควรจะสร้างไอแซมบนคอลัมน์ใดจะพิจารณาจากสองทางเช่นกันคือ

- ทางแรกพิจารณาจากลักษณะความต้องการใช้งานฐานข้อมูลซึ่งดูได้จาก query ซึ่งควรมีคุณสมบัติต่อไปนี้ [13]

- เงื่อนไขเท่ากับ หรือมีเงื่อนไขเป็นช่วง
- มีเงื่อนไขที่มีการอ้างถึงเพียงบางส่วนของค่าในคอลัมน์ที่สร้าง index
- ในกรณีที่สร้าง index หนึ่งตัวบนหลายคอลัมน์ แล้วมีการอ้างถึง index เพียงบางคอลัมน์ทางซ้ายของ index

- ทางที่สองพิจารณาจากลักษณะของข้อมูลของคอลัมน์ และตารางที่จะสร้างบีทรี ซึ่งควรมีคุณสมบัติดังนี้ [18]

- โดยทั่วไปจะทำบีทรีกับตารางที่มีขนาดตั้งแต่กลางไปจนถึงมีขนาดใหญ่ (มากกว่า 6 physical block) ไม่เหมาะกับตารางที่มีขนาดเล็กและข้อมูลที่มีการเพิ่มปริมาณอยู่เรื่อยๆ ไม่คงที่
- เหมาะกับตารางที่มีการใช้งานพร้อมๆ กันหลายคน

บทที่ 4

การปรับแต่งฐานข้อมูลในรูปของกลไกการเข้าถึงข้อมูลใน INGRES

<u>บทนำ</u>

จากบทที่ 3 กล่าวถึงทฤษฎีที่เกี่ยวกับการกำหนดรูปแบบ และกลไกในการเข้าถึงข้อมูลสำหรับ RDBMS ทั่วไป ซึ่งการที่จะสรุปออกมาเป็นกฎเกณฑ์ที่แน่นอนได้นั้น จะต้องนำทฤษฎีเหล่านี้มาทดสอบร่วมกับการทำงานของ RDBMS ที่ใช้อยู่ พร้อมทั้งต้องพิจารณาการทำงานของ RDBMS ที่ใช้อยู่ด้วย จึงจะสามารถสรุปออกมาเป็นกฎที่แน่นอนได้ ในบทนี้จะกล่าวถึงการนำทฤษฎีจากบทที่ 3 มาทำการทดสอบโดยใช้ระบบจัดการฐานข้อมูล INGRES แล้วทำการสรุปออกมาเป็นกฎ เพื่อที่จะนำไปเก็บในฐานข้อมูลความรู้ของระบบผู้เชี่ยวชาญต่อไป

ข้อมูลที่นำมาใช้ในการทดสอบนี้เป็นข้อมูลเกี่ยวกับรายการหนังสือจากห้องสมุดคณะวิศวกรรมศาสตร์ ซึ่งมีตารางที่ใช้ในการทดสอบทั้งสิ้น 3 ตาราง มีรายละเอียดดังนี้

ตาราง BORROW

std_id	char(9)
p_access_no	char(15)
b_date	char(9)
r_date	char(9)

ตาราง PCALLN

p_call_no	char(21)
title_reg1	char(80)
title_reg2	char(60)
p_length	char(4)

p_publish char(70)

author_name char(20)

author_surname char(30)

ตาราง PRLIB

p_access_no char(15)

p_publish_plc char(70)

p_be_yr char(5)

p_ad_yr char(5)

p_call_no char(21)

การพิจารณาจะแบ่งได้เป็น 3 ลักษณะ

1. พิจารณาจากแผนการประมวลผล Query (Query Execution Plan : QEP)

2. พิจารณาจากเวลา,ปริมาณการใช้งานของ input/output และปริมาณการใช้งานของ CPU ที่ใช้ในการประมวลผล Query

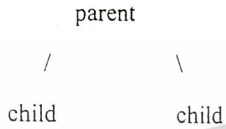
3. พิจารณาจากคู่มือการใช้งาน RDBMS ของ INGRES

<u>การทดสอบโดยพิจารณาจากแผนการประมวลผล Query</u>

ก่อนอื่นเรามาพิจารณาวิธีการประมวลผล Query ของ INGRES ก่อน INGRES ใช้ Query Optimizer ในการกำหนดเส้นทางการประมวลผลของ Query เพื่อให้ได้เส้นทางที่ดีที่สุด โดยจะวิเคราะห์จากข้อมูลพื้นฐานเบื้องต้นเช่น ขนาดของบรรทัดข้อมูล,จำนวนของบรรทัดข้อมูล, คอลัมน์ที่เป็น primary key และการกำหนด index เป็นต้น โดยจะมี Optimizedb เป็น Utility สำหรับช่วยในการเก็บสถิติดังกล่าวจากข้อมูลที่มีอยู่จริงโดยสามารถระบุได้ว่าจะเก็บทั้งฐานข้อมูล หรือเฉพาะตาราง โดยสามารถระบุคอลัมน์ที่จะเก็บสถิติข้อมูลได้ และรายละเอียดของข้อมูลที่จะนำมาเก็บเป็นสถิติสามารถกำหนดได้ว่าจะเก็บจากข้อมูลจริงทั้งหมดหรือสุ่มเฉพาะบางส่วนมา เราสามารถดูเส้นทางการทำงานของ Query ก่อนที่จะนำไปประมวลผลจริงได้โดยใช้ QEP ซึ่งเป็น Utility ที่แสดงเส้นทางการประมวลผล Query ของ INGRES ซึ่งจะแสดงในรูปของ Algebra tree ให้เราตรวจสอบการทำงานของ Query ได้

โครงสร้างของ QEP อยู่ในรูปของ Binary tree การเข้าถึงข้อมูลในแต่ละ node ใช้วิธี Postorder เฉพาะ node ที่มีการ join หรือ Cartesian product เท่านั้นที่มี node children 2 node นอกนั้นจะมีเพียง left child เท่านั้นดังรูป

ภาพที่ 10



แสดงโครงสร้าง QEP

ลักษณะของ node ทั่วไปของ QEP แบ่งได้เป็น 4 รูปแบบ

1. Orig (or leaf) node

เป็น node ที่ไม่มี children จะแสดงรายละเอียดเกี่ยวกับตาราง ซึ่งจะบอกลักษณะการเข้าถึงข้อมูลในตาราง ซึ่งแสดงถึงโครงสร้างการเก็บข้อมูลของตาราง ซึ่งถูกใช้โดย Query รูปแบบของ node เป็นดังนี้

table name

storage structure (colname)

Pages *n* Tups *m*

table name

เป็นชื่อของตารางที่ Query ใช้ประมวลผล

storage structure

เป็นโครงสร้างการเก็บข้อมูลมีดังนี้

Btree(key|NU)

Hashed(key|NU)

Heap

Isam(key|NU)

colname

เป็นชื่อของคอลัมน์ ที่นำมาประมวลผล

Pages *n* Tups *m*

n เป็นจำนวน pages ของ INGRES ที่ใช้ใน node นั้น (1 page = 2Kbyte)

m เป็นจำนวนของบรรทัดข้อมูลในตาราง

2. Proj-rest node

เป็น node ที่กำหนดคุณสมบัติหรือข้อกำหนดต่างๆ ของข้อมูลที่ต้องการ จะบอกถึงปริมาณของข้อมูลที่น้อยที่สุดที่ผ่านการกรองจากข้อกำหนดใน where เพื่อที่จะนำมา process ใน Query ต่อไป node ที่อยู่บนสุดของ tree จะเป็นตัวบอกการใช้ทรัพยากรทั้งหมดในการประมวลผล Query

รูปแบบของ node เป็นดังนี้

Proj-rest

heap

Pages *n* Tups *m*

D *x* C *y*

หรือ

Proj-rest

Sort on(*(colname)*

Pages *n* Tups *m*

D *x* C *y*

colname เป็นชื่อของคอลัมน์ ที่ทำการประมวลผล

Pages *n* Tups *m*

n เป็นจำนวน pages ของ INGRES ที่ใช้ใน node นั้น (1 page = 2Kbyte)

m เป็นจำนวนของบรรทัดข้อมูล

D *x* C *y*

x เป็น Disk I/O cost.

y เป็น CPU usage.

3. Sort node

ใช้ในกรณีที่มีการเลียงลำดับข้อมูล ทุกๆ node ยกเว้น orig-node สามารถกำหนดการเลียงลำดับได้ ใน Query ที่ต้องการผลลัพธ์ของข้อมูลแบบเรียงลำดับ sort node จะอยู่บนสุดของ tree

รูปแบบของ node เป็นดังนี้

Sort (*(colname)*

Pages *n* Tups *m*

D *x* C *y*

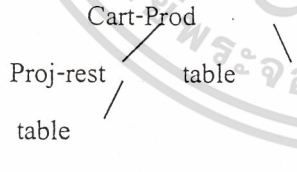
4.Join nodes

เป็น node ที่กำหนดเกี่ยวกับการ join โดยจะมี inner และ outer tree ทุกๆ join node ซึ่ง outer tree จะอยู่ทางซ้ายและ inner tree จะอยู่ทางขวา การ join มีหลายรูปแบบ แต่วิธีการ join เหมือนกัน คือแต่ละบรรทัดข้อมูลจาก outer tree จะ join กับทุกบรรทัดข้อมูล ที่มีคุณสมบัติตามที่ต้องการจาก inner tree หลังจากนั้น บรรทัดข้อมูลต่อไปจาก outer tree ก็จะเริ่มประมวลผลต่อ เป็นเช่นนี้ไปเรื่อยๆจนกว่าจะหมดข้อมูล

รูปแบบการ join มีหลายวิธีดังนี้

- cartesian product

row จาก outer node จะ compared กับทุก row จาก inner node



แต่ไม่ได้หมายความว่าทุกบรรทัดข้อมูลจะถูกอ่านขึ้นมา เฉพาะบรรทัดข้อมูลที่มีคุณสมบัติตามเงื่อนไขที่กำหนดใน Query เท่านั้นที่จะถูกนำมาเปรียบเทียบ

รูปแบบของ node เป็นดังนี้

Cart-Prod (*colname*)

heap

Pages *n* Tups *m*

D *x* C *y*

colname เป็นชื่อของคอลัมน์ที่ทำการประมวลผล

Pages *n* Tups *m*

n เป็นจำนวน pages ของ INGRES ที่ใช้ใน node นั้น (1 page = 2Kbyte)

m เป็นจำนวนของบรรทัดข้อมูล

D *x* C *y*

x เป็น Disk I/O cost.

y เป็น CPU usage.

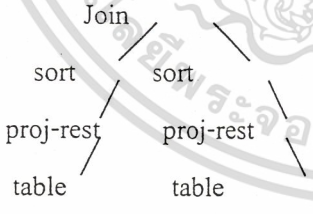
จะใช้ในกรณีที่มีการ join แบบไม่มีความสัมพันธ์กันระหว่างตารางใน Query ,ตารางที่มีขนาดเล็ก,เป็นการ join แบบไม่เท่ากัน หรือมี or operation

ตัวอย่าง. where r1.col1 != r2.col2

วิธีนี้จะใช้ disk I/O มากถ้าข้อมูลมีมากจะมีผลต่อ performance มาก

- full sort merge

เป็นวิธีที่พยายามจะลดการเปรียบเทียบข้อมูลระหว่าง inner tree กับ outer tree ให้น้อยลงกว่า Car-prod โดยการเรียงลำดับ trees ทั้ง 2 node ก่อนการเปรียบเทียบ



รูปแบบของ node เป็นดังนี้

FSM join (*colname)*

Heap

Pages *n* Tups *m*

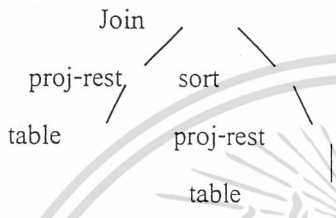
D *x* C *y*

ส่วนใหญ่จะใช้กับการ join ที่ไม่มีเงื่อนไข หรือข้อกำหนดใดๆ เช่น

select * from r1,r2 where r1.joincol = r2.joincol;

- Partial Sort Merge(PSM)

เป็นการผสมระหว่าง full sort merge กับ car-prod โดย inner tree จะถูกเรียงลำดับแต่ outer tree อาจถูกเรียงลำดับหรือไม่ก็ได้ ส่วนใหญ่ outer tree ใน PSM join จะอ้างอิงค่ามาจากตาราง Index



รูปแบบของ node เป็นดังนี้

PSM join (*colname*)

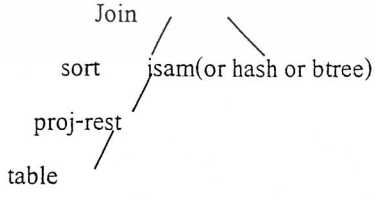
Heap

Pages *n* Tups *m*

D *x* C *y*

- Key and Tid Lookup Join

ในแต่ละบรรทัดข้อมูลของ outer row จะนำไปค้นหาค่าที่ join จาก key ใน inner join ในการค้นหาโดย key มีวิธีค้นหา 2 วิธี คือ key access กับ tid (tupple identifier)



รูปแบบของ node เป็นดังนี้

K|T join (*colname*)

Heap

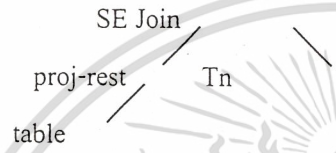
Pages *n* Tups *m*

D *x* C *y*

ส่วนใหญ่จะใช้ในกรณีที่ proj-rest returns ค่าน้อย ซึ่งควรใช้ key lookup บน isam(hash or btree) จะเร็วกว่าใช้ sort merge

- Subquery Joins

เพราะว่า SQL สามารถมี subselect ใน Query ได้ ซึ่ง node นี้จะทำหน้าที่ในการ join data ในกรณีที่มี subselect เกิดขึ้น



เมื่อ Tn คือ ตัวกำหนด subselect ที่แสดงใน QEP ก่อนหน้านี้

รูปแบบของ node เป็นดังนี้

SE join (*colname*)

Heap

Pages *n* Tups *m*

D *x* C *y*

จากที่ผ่านมาเป็นการอธิบายถึงรายละเอียดการประมวลผล Query ของ INGRES และยังกล่าวถึงวิธีที่จะพิจารณาการประมวลผลของ Query ซึ่ง INGRES มีเครื่องมือที่ช่วยในด้านนี้ คือ QEP ต่อจากนี้จะพิจารณาเส้นทางประมวลผลของ Query โดยจะทดลองประมวลผล Query แล้วใช้ QEP ดูเส้นทางการประมวลผล เพื่อหาข้อสรุปของกฎเกณฑ์ในการกำหนดคอลัมน์ที่จะนำมาใช้ในการตัดสินใจสร้าง index

Query ที่นำมาทดสอบนี้จะมีการ join ตารางข้อมูลหลายตาราง และมีเงื่อนไขในการดึงข้อมูลหลายเงื่อนไขจากหลายตารางดังนี้

SQL command

```
select p_ad_yr,title_reg2
from pcalln a,prlib b,borrow c
where a.p_call_no = b.p_call_no and b.p_access_no = c.p_access_no
and a.p_length = '23' and b.p_ad_yr = '1987'
and c.std_id = '38013043'
```

จาก Query นี้จะพิจารณาว่าควรสร้าง index บนคอลัมน์ใดให้ได้ประสิทธิภาพสูงสุดนั้น ต้องพิจารณาจากเส้นทางการทำงานของ Query Optimizer ภายใต้สภาวะของการสร้าง index ที่แตกต่างกัน โดยสภาวะของการสร้าง index ที่แตกต่างกัน 4 สภาวะดังนี้

สภาวะที่ 1 ตาราง borrow ใช้โครงสร้างข้อมูลแบบ heap

ตาราง pcalln ใช้โครงสร้างข้อมูลแบบ heap

ตาราง prlib ใช้โครงสร้างข้อมูลแบบ heap

จากการประมวลผลโดยใช้ QEP ได้เส้นทางการทำงานของ Query ดังนี้

QUERY PLAN 3,1, no timeout, of main query

FSM Join(p_call_no)
Heap
Pages 1 Tups 1
D93 C59

/ \

FSM Join (p_acess_no) Proj-rest
Sort on (p_call_no) Sort on (p_call_no)
Pages 1 Tups 1 Pages 5 Tups 94
D40 C37 D53 C13

/ \ /

Proj-rest Proj-rest pcalln
Sort on(p_access_no) Sort on(p_access_no) (a)
Pages 1 Tups 2 Pages 1 Tups 31 Heap
D9 C14 D32 C20 Pages 211 Tups 1263

/ /

borrow prlib
(c) (b)
Heap Heap
Pages 35 Tups 1432 Pages 126 Tups 2007

สภาวะที่ 2 ตาราง borrow ใช้โครงสร้างข้อมูลแบบ hash บนคอลัมน์ std_id

ตาราง pcalln ใช้โครงสร้างข้อมูลแบบ hash บนคอลัมน์ p_length

ตาราง prlib ใช้โครงสร้างข้อมูลแบบ hash บนคอลัมน์ p_call_no

จากการประมวลผลโดยใช้ QEP ได้เส้นทางการทำงานของ Query ดังนี้

สภาวะที่ 3 ตาราง borrow ใช้โครงสร้างข้อมูลแบบ hash บนคอลัมน์ std_id
ตาราง pcalln ใช้โครงสร้างข้อมูลแบบ hash บนคอลัมน์ p_call_no
ตาราง prlib ใช้โครงสร้างข้อมูลแบบ hash บนคอลัมน์ p_ad_yr

จากการประมวลผลโดยใช้ QEP ได้เส้นทางการทำงานของ Query ดังนี้

QUERY PLAN 4,1, no timeout, of main query

K Join(p_call_no)

Heap

Pages 1 Tups 1

D8 C2

/ \

FSM Join(p_acess_no) pcalln

Sorted(p_acess_no) (a)

Pages 1 Tups 1 Hashed(p_call_no)

D7 C2 Pages 517 Tups 1263

/ \

Proj-rest Proj-rest

Sort on(p_acess_no) Sort on(p_acess_no)

Pages 1 Tups 2 Pages 1 Tups 31

D1 C0 D6 C0

/ /

borrow prlib

(c) (b)

Hashed(std_id) Hashed(p_ad_yr)

Pages 128 Tups 1432 Pages 358 Tups 2007

****

สภาวะที่ 4 ตาราง borrow ใช้โครงสร้างข้อมูลแบบ hash บนคอลัมน์ std_id

ตาราง pcalln ใช้โครงสร้างข้อมูลแบบ hash บนคอลัมน์ p_call_no

ตาราง prlib ใช้โครงสร้างข้อมูลแบบ hash บนคอลัมน์ p_acess_no

จากการประมวลผลโดยใช้ QEP ได้เส้นทางการทำงานของ Query ดังนี้

QUERY PLAN 4,1, no timeout, of main query

K Join(p_call_no)

Heap

Pages 1 Tups 1

D4 C0

/ \

K Join(p_acess_no) pcalln

Heap (a)

Pages 1 Tups 1 Hashed(p_call_no)

D3 C0 Pages 517 Tups 1263

/ \

Proj-rest prlib

Heap (b)

Pages 1 Tups 2 Hashed(p_acess_no)

D1 C0 Pages 256 Tups 2007

/

borrow

(c)

Hashed(std_id)

Pages 66 Tups 1432

****

จากการทดสอบข้างต้นจะเห็นว่าในกรณีที่มีการใช้งานตารางข้อมูลตั้งแต่ 2 ตาราง ขึ้นไปร่วมกันและในคำสั่ง where มีเงื่อนไข = มีหลายเงื่อนไขจากหลายตาราง พิจารณาจาก QUERY PLAN พบว่า Query optimizer ของ INGRES จะเลือกทำเงื่อนไขที่มี row return น้อยที่สุดก่อน(ควรสร้าง index บนคอลัมน์นี้) เมื่อได้บรรทัดข้อมูลที่ต้องการก็จะทำให้รู้ค่าของคอลัมน์ที่จะนำมาใช้เชื่อมตารางด้วย (ไม่จำเป็นต้องสร้าง index บนคอลัมน์นี้) หลังจากนั้นก็จะทำการเชื่อมตารางโดยใช้ key lookup ไปยังคอลัมน์ที่ใช้เชื่อมตารางของอีกตารางหนึ่ง (ควรสร้าง index บนคอลัมน์นี้) แล้วทำการกรองตารางตามเงื่อนไขของคอลัมน์ (ไม่จำเป็นต้องสร้าง index บนคอลัมน์นี้) เพื่อให้ได้บรรทัดข้อมูลตามเงื่อนไขของตารางนั้น

สรุปได้ว่าในกรณีที่มีการใช้งานตารางข้อมูลร่วมกันตั้งแต่สองตารางขึ้นไป และมีเงื่อนไขในการกรองข้อมูลหลายเงื่อนไขจากหลายตาราง ให้สร้าง index บนคอลัมน์ที่เป็นเงื่อนไขในการกรองข้อมูลที่มีจำนวนบรรทัดข้อมูลที่ตรงกับเงื่อนไขน้อยที่สุด จากนั้นสร้าง index บนตาราง

ที่เชื่อมกับตารางที่สร้าง index ไว้แล้วในตอนต้นบนคอลัมน์ที่ใช้เชื่อมตาราง และสร้าง index บนคอลัมน์ที่ใช้เชื่อมตารางของตารางต่อ ๆ ไป

พิจารณาจากเวลา,ปริมาณการใช้งานของ input/output และปริมาณการใช้งานของ CPU ที่ใช้ในการประมวลผล Query

โดยใช้ Dbmsinfo( ) Function ของ INGRES ซึ่งเป็น function ที่ return ค่าเวลา,ปริมาณการใช้งานของ input/output และปริมาณการใช้งานของ CPU ณ เวลาหนึ่ง ซึ่งสามารถคำนวณค่าที่ใช้ในการประมวลผล Query ได้จากการนำค่าที่ได้จาก function ก่อนและหลัง Query มาลบกันจะได้ค่าที่ใช้ไปในการประมวลผล Query

รูปแบบของ SQL command ที่นำมาใช้ในการทดสอบ

1. มีคำสั่ง where โดยมีเงื่อนไขแบบ like (pattern matching)
2. มีคำสั่ง where โดยมีเงื่อนไขแบบ = (Exact-match keyed retrive)
3. มีคำสั่ง where โดยมีเงื่อนไขแบบ < , > , <= , >= (range searches)
4. มีคำสั่ง where โดยมีเงื่อนไขแบบ = หลายเงื่อนไข และเชื่อมเงื่อนไขด้วย and
5. มีคำสั่ง where โดยมีเงื่อนไขแบบ = หลายเงื่อนไข และเชื่อมเงื่อนไขด้วย or
6. มีคำสั่ง where โดยมีเงื่อนไขแบบ !=
7. มีคำสั่ง Order by

1. มีคำสั่ง where โดยมีเงื่อนไขแบบ like (pattern matching)

SQL command

```
select *
from prlib
where p_ad_yr like '198%'
```

ชนิดการทดสอบ	โครงสร้างข้อมูล			
	Heap	Hash	Isam	Btree
เวลาที่ใช้ในการประมวลผล	110	151	70	63
ปริมาณการใช้งานของ input/output	46	95	18	15
ปริมาณการใช้งาน CPU	150	200	90	60

SQL command

```
select *
from prlib
where p_ad_yr like '%%76'
```

ชนิดการทดสอบ	โครงสร้างข้อมูล			
	Heap	Hash	Isam	Btree
เวลาที่ใช้ในการประมวลผล	25	36	34	35
ปริมาณการใช้งานของ input/output	31	92	90	56
ปริมาณการใช้งาน CPU	110	230	170	200

จากการทดสอบข้างต้นสรุปได้ว่า Isam หรือ Btree เป็นวิธีที่ดีที่สุดในการเก็บข้อมูลในกรณีที่ Query มีคำสั่ง where ใช้เงื่อนไข like แต่ถ้าในเงื่อนไข like มีการอ้างถึงเพียงบางส่วนของข้อมูลทางด้านขวา Heap จะเป็นวิธีที่ดีที่สุดสำหรับเก็บข้อมูล

2. มีคำสั่ง where โดยมีเงื่อนไขแบบ = (Exact-match keyed retrievals)

SQL command

```
select *
from prlib
where p_ad_yr = '1976'
```

ชนิดการทดสอบ	โครงสร้างข้อมูล			
	Heap	Hash	Isam	Btree
เวลาที่ใช้ในการประมวลผล	55	24	35	45
ปริมาณการใช้งานของ input/output	36	7	9	12
ปริมาณการใช้งาน CPU	140	30	60	50

จากการทดสอบข้างต้นสรุปได้ว่า Hash เป็นวิธีที่ดีที่สุดสำหรับเก็บข้อมูลในกรณีที่มีเงื่อนไขใน where เป็น =

3. มีคำสั่ง where โดยมีเงื่อนไขแบบ < , > , <= , >= (range searches)

SQL command

```
select *
from prlib
where p_ad_yr > '1988'
```

ชนิดการทดสอบ	โครงสร้างข้อมูล			
	Heap	Hash	Isam	Btree
เวลาที่ใช้ในการประมวลผล	71	91	39	40
ปริมาณการใช้งานของ input/output	36	95	12	9
ปริมาณการใช้งาน CPU	150	180	40	50

จากการทดสอบข้างต้นสรุปได้ว่า Isam หรือ Btree เป็นวิธีที่ดีที่สุดในการเก็บข้อมูลในกรณีที่ Query มีคำสั่ง where ใช้เงื่อนไข < , > , >= , <=

4. มีคำสั่ง where โดยมีเงื่อนไขแบบ = หลายเงื่อนไขและเชื่อมเงื่อนไขด้วย and

SQL command

```
select *
from  prlib
where p_publish_plc = 'LONDON' and  p_ad_yr = '1971'
```

ชนิดการทดสอบ	โครงสร้างข้อมูล		
	Heap	Hash on(p_publish_plc,p_ad_yr)	Hash on (p_ad_yr,p_publish_plc)
เวลาที่ใช้ในการประมวลผล	72	38	24
ปริมาณการใช้งานของ input/output	34	6	1
ปริมาณการใช้งาน CPU	140	60	20

```
select count(*)
from prlib
where p_publish_plc = 'LONDON'
```

ได้ผลลัพท์ = 233

```
select count (*)
from prlib
where p_ad_yr = '1971'
```

ได้ผลลัพท์ = 85

จากการทดสอบข้างต้นสรุปได้ว่าในกรณีที่ where มีเงื่อนไข = และมีหลายเงื่อนไข ทำ combine key ด้วยวิธี Hash เป็นวิธีที่ดีที่สุดโดยเรียงลำดับจากเงื่อนไขจากเงื่อนไขที่มีจำนวนบรรทัดข้อมูลที่ตรงกับเงื่อนไขน้อยที่สุดไปมาก

5. มีคำสั่ง where โดยมีเงื่อนไขแบบ = หลายเงื่อนไขและเชื่อมเงื่อนไขด้วย OR

SQL command

```
select *
from  prlib
where p_publish_plc = 'LONDON' or  p_ad_yr = '1971'
```

ชนิดการทดสอบ	โครงสร้างข้อมูล	
	Heap	Hash on (p_ad_yr,p_publish_plc)
เวลาที่ใช้ในการประมวลผล	63	92
ปริมาณการใช้งานของ input/output	31	87
ปริมาณการใช้งาน CPU	130	190

จากการทดสอบข้างต้นสรุปได้ว่าในกรณีที่ where มีเงื่อนไข = และมีหลายเงื่อนไข โดยมีตัวเชื่อม OR เข้ามาเกี่ยวข้อง Heap เป็นวิธีที่ดีที่สุดในการเก็บข้อมูล

6. มีคำสั่ง where โดยมีเงื่อนไขแบบ !=

SQL command

```
select *
from prlib
where p_be_yr != ' '
```

ชนิดการทดสอบ	โครงสร้างข้อมูล	
	Heap	Hash
เวลาที่ใช้ในการประมวลผล	38	54
ปริมาณการใช้งานของ input/output	34	101
ปริมาณการใช้งาน CPU	140	190

จากการทดสอบข้างต้นสรุปได้ว่าในกรณีที่ where มีเงื่อนไข != Heap เป็นวิธีที่ดีที่สุดในการเก็บข้อมูล เพราะต้องมีการค้นหาข้อมูลทั้งตาราง

7. มีคำสั่ง ORDER BY

SQL command

```
select *
from prlib
where p_ad_yr = '1971'
order by p_call_no
```

ชนิดการทดสอบ	โครงสร้างข้อมูล			
	Heap	Hash	Isam	Btree
เวลาที่ใช้ในการประมวลผล	49	43	31	28
ปริมาณการใช้งานของ input/output	36	72	9	8
ปริมาณการใช้งาน CPU	140	160	80	70

จากการทดสอบข้างต้นสรุปได้ว่าในกรณีที่มีคำสั่ง ORDER BY ซึ่งมีการทำงานเกี่ยวกับการเรียงลำดับข้อมูล Btree เป็นวิธีที่ดีที่สุดในการเก็บข้อมูล

พิจารณาจากคู่มือการใช้งาน RDBMS ของ INGRES

INGRES สรุปการเลือกชนิดของกลไกในการเข้าถึงข้อมูลให้เหมาะสมกับการใช้งานตารางข้อมูลไว้ดังนี้

INGRES มีชนิดของกลไกในการเข้าถึงข้อมูล อยู่ 4 แบบ คือ

- Heap เป็นการเข้าถึงข้อมูลแบบเรียงลำดับ
- Hash มีหลักการในการเลือกที่อยู่ของข้อมูล โดยใช้ค่าของคีย์ข้อมูลเป็นตัวกำหนด
- Isam จะจัดเรียงข้อมูลโดยใช้ค่าของคีย์คอลัมน์ และทำการเข้าถึงข้อมูลบนค่าที่แน่นอน
- Btree จะจัดเรียงข้อมูลโดยใช้ค่าของคีย์คอลัมน์ และทำการเข้าถึงข้อมูลบนค่าที่แน่นอน

Heap

จะไม่มีคีย์ จะเป็นลักษณะการเก็บข้อมูลแบบต่อ ๆ กันไป ดังนั้นเมื่อเพิ่มเติม row เข้าไปจะถูกเก็บไว้ท้ายสุด

Heap เป็นกลไกในการเข้าถึงข้อมูล ที่ดีหากใช้ในกรณีที่ [18]

1.Load ข้อมูลจากตาราง เนื่องจากเป็นกลไกในการเข้าถึงข้อมูลที่เพิ่มเติมข้อมูลได้เร็วที่สุด เพราะจะเพิ่มเติมข้อมูลเข้าไปที่ท้ายของ Heap ดังนั้นจึงไม่มี overhead ที่เกิดจากคำนวณที่ ที่จะใส่ข้อมูล

2. เป็นตารางข้อมูลที่มีขนาดเล็ก (มีจำนวน Page ไม่มากต่อตาราง)

3. มีการเข้าถึงข้อมูลทุก ๆ row โดยไม่ต้อง sorting

ข้อเสียของ Heap และวิธีแก้ปัญหา [18]

1. Space ที่ deleted rows จะไม่นำมาใช้ใหม่ แก้ปัญหาโดยใช้ statement modify เพื่อนำมาใช้ใหม่

2. Select และแก้ไข้ช้ามากเพราะเข้าถึงแบบ sequential แก้ปัญหาโดยถ้าเป็นตารางขนาดเล็กก็ไม่เป็นปัญหา แต่ถ้าเป็นตารางขนาดใหญ่ก็สร้าง secondary index

Hash

เป็นโครงสร้างที่ใช้ match key value โดยจะต้องมีการกำหนดคีย์ ซึ่งถ้าไม่มีการกำหนดคีย์ จะใช้ฟิลด์แรกเป็นคีย์

Hash เป็นกลไกในการเข้าถึงข้อมูลที่ดีหากใช้ในกรณีที่ [18]

1. จะดึงข้อมูลได้เร็วที่สุดในกรณีที่มีเงื่อนไขโดยใช้ key value ที่กำหนด

2. เงื่อนไขในการดึงข้อมูลเป็นแบบ '='

3. สามารถรองรับการทำงานที่มีผู้ใช้แก้ไขข้อมูลในตารางพร้อม ๆ กันหลาย ๆ คน ได้ดีกว่า BTREE

ข้อเสียของ Hash และวิธีแก้ปัญหา [18]

1. ไม่เหมาะกับรูปแบบที่เป็นการใช้ pattern matching คือการนำเฉพาะบางส่วนของคีย์มาทำการเปรียบเทียบ และค้นหาข้อมูล เพราะจะทำให้ประสิทธิภาพจะลดลง แก้ปัญหาโดยใช้ Isam, Btree แทน

2. ไม่เหมาะกับการดึง range of values คือการดึงข้อมูลในลักษณะที่เป็นช่วงเช่น > หรือ < เป็นต้น จะทำให้ประสิทธิภาพจะลดลง แก้ปัญหาโดยใช้ Isam, Btree แทน

3. ไม่ควรใช้กับส่วนใดของ multi-colurmn key คือในกรณีที่คีย์ประกอบด้วยหลายคอลัมน์และมีการใช้งานที่อ้างเพียงบางคอลัมน์ของคีย์นั้น จะทำให้ประสิทธิภาพจะลดลง แก้ปัญหาโดยใช้ Isam, Btree แทน

4. มี overflow page ในตารางที่แก้ไขใหม่ ถ้าคีย์ซ้ำกันก็ไม่ควรใช้วิธีนี้ แต่ถ้าคีย์เป็น unique แสดงว่า algorithm ของ Hash อาจจะกระจายข้อมูลได้ไม่ดี

5.มี overflow page หลังจากมีการเพิ่มเติม row ใช้ dynamic structure แบบ Btree

Isam

เป็นกลไกในการเข้าถึงข้อมูล ที่ใช้คีย์เป็นตัวกำหนดในการข้าถึงข้อมูลเช่นกัน โดยจะต้องมีการกำหนดคีย์ ซึ่งถ้าไม่มีกำหนดคีย์ จะได้คอลัมน์แรกเป็นคีย์เช่นกัน

Isam เป็นกลไกในการเข้าถึงข้อมูล ที่ดีหากใช้ในกรณีที่ [18]

1. ขนาดของข้อมูลในตารางคงที่ไม่ค่อยมีการเปลี่ยนแปลง

2. มีการดึงข้อมูลแบบ pattern matching คือการนำเฉพาะบางส่วนของคีย์มาทำการเปรียบเทียบและค้นหาข้อมูล แต่ต้องเริ่มเปรียบเทียบจากส่วนซ้ายสุดของคีย์เท่านั้น ซึ่งใช้ได้ทั้งในกรณีที่เป็นคีย์แบบ multicolumn ด้วย

3. สามารถใช้กับการเงื่อนไขในการดึงข้อมูลที่เป็นช่วง เช่น > ,<

4. สามารถรองรับการทำงานที่มีผู้ใช้แก้ไขข้อมูลในตารางพร้อม ๆ กันหลาย ๆ คน ได้ดีกว่า BTREE

Isam ไม่เหมาะกับการใช้งานต่อไปนี้ [18]

1. ไม่เหมาะกับตารางที่มีการขยายตัวอย่างรวดเร็วเพราะทำให้เกิด overflow page ควรใช้ BTREE แทน

2. หากใช้ pattern matching แล้วไม่ระบุ character ซ้ายสุด จะทำให้ต้องค้นหาข้อมูลทั้งตาราง

Binary tree (Btree)

เป็นกลไกในการเข้าถึงข้อมูล ที่ใช้คีย์เป็นตัวกำหนดในการข้าถึงข้อมูลเช่นกัน โดยจะต้องมีการกำหนดคีย์ ซึ่งถ้าไม่มีกำหนดคีย์ จะได้คอลัมน์แรกเป็นคีย์เช่นกัน

Index Growth

ข้อแตกต่างระหว่าง Isam กับ Btree คือ [18]

- Btree นั้นใช้ dynamic index กล่าวคือ dynamic index จะเพิ่มตามการเพิ่มขนาดของตารางในขณะที่ Isam นั้นใช้ static index
- Isam จะไม่มีการจำกัด page สุดท้ายของตารางดังนั้นถ้ามีการเพิ่มข้อมูลที่เป็นคีย์ลงไปใน page นั้นเรื่อย ๆ อาจทำให้เกิด overflow page ที่ page สุดท้ายได้ในขณะที่Btree จะมีการจำกัด leaf page ที่เป็น page สุดท้ายไว้หากมีการเพิ่มเติมข้อมูลที่เป็นคีย์จน page นั้นเต็มก็จะเกิดการแบ่งแยกออกเป็น 2 leaf paged ทำให้ไม่เกิด overflow pages

และใน index page ก็ใช้หลักการเดียวกัน เพราะฉะนั้นจะเห็นได้ว่า index จะเพิ่มตามขนาดของตาราง

Locking และ Btree table [18]

โดยปกติแล้ว leaf-page และ data-page จะถูก lock จนกระทั่งจบ transaction หากมีการ lock เกิดขึ้นที่ leaf level ในขณะที่ค้นหา Btree index จะเกิดการ lock ใน index page ตามเส้นทางที่จะไป leaf page นั้นทำการ lock แบบชั่วคราวตามลำดับ level ที่เป็น Pattern level ของ leaf page ที่เราต้องการจะเข้าถึง ดังนั้นจะเห็นได้ว่าจำนวนของการ lock ใน Btree มีมากกว่าใน Isam และ Hash ถึงสองเท่า

Delete Rows: Data Pages [18]

ถ้า row ที่ถูกลบอยู่บน data pages ที่มีความสัมพันธ์กัน เนื้อที่ว่างนั้นจะถูกนำมาใช้อีกครั้ง แต่ถ้า rows ที่อยู่บน data page ที่ไม่มีความสัมพันธ์กันถูกลบทั้งหมด เนื้อที่ว่างส่วนนั้นจะไม่ถูกนำมาใช้ ต้องทำ midify to merge กับ page นั้นลงไปที่ free list เพื่นำที่ว่างนั้นไปใช้ต่อไป

ข้อดีของ Btree [18]

1. สามารถใช้กับ ตารางที่มีอัตราการโตอย่างรวดเร็ว
2. สามารถใช้กับ pattern matching
3. สามารถใช้กับการ ดึง key value แบบเป็นช่วง
4. ในกรณีที่เป็น multi column key สามารถดึงข้อมูลโดยอ้างแค่เพียงบางคอลัมน์ทางซ้ายมือของ (leftmost) ของคีย์

ข้อเสียของ Btree [18]

1. ไม่เหมาะที่จะใช้กับตารางที่มีขนาดเล็ก และมีคนใช้พร้อม ๆ กันหลายคน
2. หากจะใช้ pattern matching แต่ไม่กำหนด character ตัวซ้ายสุดจะทำให้ต้องค้นหาข้อมูลทั้งตาราง ซึ่งเป็นผลให้เสียเวลา
3. หากจะใช้ในบางส่วนของ multi-column key แต่ไม่ระบุคอลัมน์ ซ้ายสุดจะทำให้ต้องสร้าง secondary index เป็นคอลัมน์ ที่ใช้ค้นหาอยู่

สรุปเปรียบเทียบกรณีที่เลือกใช้ Isam แทนที่จะใช้ Btree [18]

1. กรณีที่ใช้ตารางที่มีขนาดคงที่ (static table)

2. ใช้การกระทำเกี่ยวกับดิสก์ (disk operation) ในการเข้าถึง data page น้อยกว่า Btree เพราะ Btree ต้องกระทำถึง leaf level

3. ไม่ต้อง lock จาก index page ดังนั้น concurrency จะดีกว่า

4. การเข้าถึงข้อมูลของตารางแบบ Isam กรณีที่ไม่มีความต้องการข้อมูลแบบเรียงลำดับจะทำได้เร็วกว่า Btree เพราะ Isam มี pointer ที่ใช้ไปยัง data page โดยตรง

สรุปเปรียบเทียบกรณีที่เลือกใช้ Btree แทนที่จะใช้ Isam [18]

1. ตารางมีอัตราการเพิ่มขนาดอย่างรวดเร็ว (อาจทำให้เกิด overflow .ใน Isam storage structure)

2. ในกรณีที่มีความาต้องการข้อมูลแบบเรียงลำดับบนคีย์ Btree จะดีกว่า Isam เพราะว่าใน Btree จะมีการเข้าถึงข้อมูลแบบเรียงลำดับอยู่แล้ว

จากรายละเอียดทั้งหมด INGRES ได้นำมาสร้างตารางเพื่อนำมาช่วยในการตัดสินใจในการพิจารณาเลือกชนิดโครงสร้างของ index โดยพิจารณาจากรูปแบบของเงื่อนไขต่างๆ ใน SQL query ได้ดังนี้

ตารางที่ 14

แสดงการเปรียบเทียบลักษณะการใช้งานตารางข้อมูลของ INGRES

รูปแบบของเงื่อนไข	HASH	ISAM	BTREE
Need pattern matching	4	1	1
Need range searches	4	1	1
Exact-match keyed retrievals	1	2	2
Sort data	4	2	1
Concurrent updates	1	1	2
Static data	1	1	2
Dynamic data	3	3	1
Delete frequent	1	1	3
Update frequent	1	1	2

ค่าเปรียบเทียบที่ใช้ในตาราง 4 Bad ,3 OK ,2 Good ,1 Excellent [18]

SECONDARY INDEX

INGRES มี Secondary index เข้ามาช่วยเพิ่มประสิทธิภาพในกรณีที่มีความต้องการสร้าง index ที่แตกต่างกันบนตารางข้อมูลเดียวกัน ซึ่งทำให้สามารถสร้าง index บนตารางใด ๆ ได้โดยไม่จำกัดจำนวนของ index กฎเกณฑ์ที่จะนำ Secondary index มาใช้เพื่อเพิ่มประสิทธิภาพของการทำงานมีดังนี้ [18]

1. การเก็บ Secondary Index ของ INGRES จะสร้างตาราง index แยกมาจากตารางหลักของข้อมูล ซึ่งตารางที่แยกออกมานี้จะเก็บข้อมูลเฉพาะคอลัมน์ที่นำมาสร้างเป็น Secondary index เท่านั้น จึงทำให้มีขนาดเล็กกว่าตารางหลัก ดังนั้นถ้ามี Query ที่อ้างถึงเฉพาะคอลัมน์ที่มีอยู่ใน Secondary index เท่านั้น ก็จะทำให้ลดปริมาณของการใช้ disk I/O ลง เพราะไม่ต้องไปดึงข้อมูลจากตารางหลักซึ่งมีขนาดใหญ่กว่า

2. ถ้ามีความต้องการสร้าง index สองวิธีบนตารางเดียวกัน เช่น Query1 สรุปว่าต้องใช้วิธี HASH และ Query2 สรุปว่าต้องใช้ ISAM หรือ BTREE ควรสร้าง ISAM หรือ BTREE บนตารางหลักเพราะการสร้าง ISAM หรือ BTREE จะทำให้ข้อมูลอยู่ใกล้ชิดกันบน page ข้อมูลเดียวกัน ทำให้ลดปริมาณ disk I/O ได้ ส่วน วิธี HASH การเก็บข้อมูลจะเป็นแบบสุ่ม ทำให้การเก็บข้อมูลกระจายไม่อยู่ใกล้กัน

บทที่ 5

ระบบผู้เชี่ยวชาญ

<u>บทนำ</u>

ปัญญาประดิษฐ์ (Artificial Intelligence) เป็นแขนงวิชาที่มุ่งเน้นในด้านการทำให้คอมพิวเตอร์สารมารถแสดงความฉลาดออกมาได้ เช่น การคิด การหาเหตุผล การรับรู้ หรือกระทำเป็นต้น ระบบผู้เชี่ยวชาญเป็นแขนงหนึ่งของวิชาปัญญาประดิษฐ์ (Artificial Intelligence:AI) โดยตัวระบบคือโปรแกรมคอมพิวเตอร์ที่ถูกสร้างขึ้นมาเพื่อทำหน้าที่เป็นผู้เชี่ยวชาญ และให้คำปรึกษาช่วยแก้ปัญหาที่ยากซับซ้อน โดยปัญหาที่จะนำมาแก้ไขหรือให้คำปรึกษาจะเป็นปัญหาเฉพาะเรื่อง มีขอบเขตจำกัด และในการแก้ปัญหาไม่สามารถกำหนดขั้นตอนได้ล่วงหน้า โดยระบบจะอาศัยความรู้ (knowledge) ที่มีอยู่ในตัวของมันเองมาทำการวินิจฉัย (inference) ด้วยกลไกการวินิจฉัย (inference engine) ร่วมกับความจริง (fact) ที่ได้มาใหม่จากผู้ใช้แล้วให้คำแนะนำหรือวินิจฉัยออกมาได้ โดยความรู้ที่ใช้ในการวินิจฉัยนั้นได้มาจากความรู้ที่เป็นความจริงที่อาจจะถูกบันทึกไว้ในรูปของตำราหรือเอกสารทางวิชาการ และความรู้ที่ได้จากประสบการณ์ที่อาจจะไม่อยู่ในรูปของตำราหรือเอกสารทางวิชาการ แต่จะต้องดึงออกมาจากผู้เชี่ยวชาญหรือผู้ชำนาญที่มีประสบการณ์

ศาสตราจารย์ Edward Feigenbaum แห่งมหาวิทยาลัยสแตนฟอร์ด ซึ่งเป็นนักคันคว้าขั้นแนวหน้าในสาขาปัญญาประดิษฐ์ (Artificial Intelligence) ได้ให้คำจำกัดความของระบบผู้เชี่ยวชาญ (Expert System) ไว้ว่า ระบบผู้เชี่ยวชาญคือโปรแกรมคอมพิวเตอร์ที่มีความฉลาดด้วยการใช้ความรู้และขบวนการอนุมาน (inference procedure) ในการแก้ปัญหาที่ยุ่งยากขนาดที่ต้องใช้ประสบการณ์ความชำนาญการของมนุษย์ จึงจะแก้ไขได้

ความเป็นมาของระบบผู้เชี่ยวชาญ และวิวัฒนาการของระบบผู้เชี่ยวชาญ

เนื้อหาวิชาของระบบผู้เชี่ยวชาญ เป็นการพิจารณาถึงวิธีการและเทคนิคในการสร้างระบบผู้เชี่ยวชาญเพื่อใช้ในการแก้ปัญหาเฉพาะด้าน ความเชี่ยวชาญในที่นี้ประกอบด้วยความรู้ในเรื่องใดเรื่องหนึ่ง ความเข้าในถึงปัญหาในเรื่องนั้น ๆ และทักษะในการแก้ปัญหาบางอย่างในเรื่องนั้น

วิวัฒนาการของระบบผู้เชี่ยวชาญใน 2 ทศวรรษที่ผ่านมาโครงงานส่วนใหญ่จะใช้เวลาในการดำเนินการหลายปี ในที่นี้จะนำโครงงานที่สำคัญ และน่าสนใจที่ได้มีการพัฒนา และประสบผลสำเร็จมากล่าวไว้พอเป็นสังเขปดังนี้

1. Dendral โครงงานนี้ได้รับการพัฒนาขึ้นโดยมาหวิทยาลัยสแตนฟอร์ด โดยใช้ระยะเวลาในการพัฒนาจนถึงปัจจุบันนี้รวมทั้งสิ้น 16 ปี ผลงานของโครงงานนี้คือ Dendral และ Meta-Dendral ระบบ Deneral ใช้สำหรับวิเคราะห์ mass spectrographic, nuclear magnetic resonance และข้อมูลการทดลองทางเคมีอื่น ๆ โดยสามารถวินิจฉัยหาโครงสร้างทางเคมีที่เป็นไปได้ของสารประกอบ Meta-Dendral เป็นระบบที่พัฒนาต่อจาก Dendral โดยเพิ่มความรู้ในการเสนอ และเลือกกฎต่าง ๆ ที่แยกกันอยู่ สำหรับโครงสร้างทางอินทรีย์เคมี ระบบนี้สามารถที่จะให้กำเนิดกฎ และทดสอบกฎเหล่านั้นได้ โดยตรวจสอบกับข้อมูลที่ได้จาการทดลอง

2. Macsyna เป็นระบบผู้เชี่ยวชาญทางด้านคณิตศาสตร์ พัฒนาขึ้นโดย MIT ระบบนี้มีความสามารถในการทำดิฟเฟอเรนชิเอท และอินทิเกรท โดยใช้สัญญาลักษณ์และความสามารถในการลดรูปนิพจน์ทางคณิตศาสตร์ได้อย่างดีเยี่ยม Macsyna ประกอบด้วยกฎที่ได้รับจากผู้เชี่ยวชาญทางคณิตศาสตร์ประยุกต์หลายร้อยกฎ แต่ละกฎจะแสดงถึงการเปลี่ยนรูปจากนิพจน์หนึ่งไปเป็นนิพจน์หนึ่งที่สมมูลกัน เพื่อใช้ในการแก้ปัญหาระบบนี้พัฒนาด้วยภาษา LISP

3. Hearsat-ll เป็นระบบเข้าใจคำพูด (Speech Understanding System) ซึ่งการสำรวจระบบที่สามารถเข้าใจคำพูดนั้น นับได้ว่าเป็นงานที่ยากที่สุดเมื่อเทียบกับสาขาอื่น ๆ ของปัญญาประดิษฐ์ Hearsat-ll ถูกพัฒนาขึ้นโดยมหาวิทยาลัยคาร์เนกิเมลลอน เป็นระบบปัญญาประดิษฐ์ระบบแรกที่สามารถเข้าใจการสนทนาติดต่อกันรวมแล้วเป็นศัพท์กว่า 1,000 คำ

นอกจากนี้ยังมีระบบอื่น ๆ ที่น่าสนใจ ดังแสดงไว้ในตาราง 15

ตารางที่ 15
แสดงระบบผู้เชี่ยวชาญในยุคต้น ๆ

SYSTEM	DATE	AUTHOR	SUBJECT
Denaral	1965	Stanford	Infers information about chemical structures
Macsymn	1965	MIT	Performs complex mathematical analysis
Mycin	1972	Stanford	diagnosis of blld disecese
Teiresias	1972	Stanford	Knowledge transformation tool
Protpector	1972	Stanford Res, Lnst	Mineral sxploration and identification tool
Kearsay	1973	Carnegie-Mellon	Natural-language interpertation for subset language
Age	1973	Stanford	Expert-system-generation tool
OPS5	1974	Carnegie-Mellon	Expert-system-building tool

<u>วิศวกรรมทางด้านความรู้</u>

ในช่วงปี 1950 ถึง 1960 ความรู้ที่ใช้ในโปรแกรมปัญญาประดิษฐ์จะเป็นไปในลักษณะเขียนโปรแกรมจะเปลี่ยนความรู้ไปเป็นรหัส โดยไม่มีการแยกความรู้ออกจากส่วนของกลไกการหาเหตุผล ทำให้ผู้เขียนโปรแกรมจะต้องเรียนรู้ความชำนาญจากผู้เชี่ยวชาญก่อน หรือผู้เขียนโปรแกรมจะต้องเป็นผู้เชี่ยวชาญด้วยจึงจะสามารถเขียนโปรแกรมได้

แต่ในระยะหลัง วิศวกรรมด้านความรู้กลายมาเป็นวิธีที่ใช้เพื่อให้ได้มาซึ่งความรู้โดยการให้ผู้เชี่ยวชาญได้มีการติดต่อโต้ตอบกับวิศวกรความรู้ หรือโปรแกรมที่ใช้สร้างระบบผู้เชี่ยวชาญ สาเหตุที่ทำให้ต้องมีการศึกษาเกี่ยวกับวิศวกรรมทางด้านความรู้คือ เนื่องจากความเชี่ยวชาญของผู้เชี่ยวชาญมักจะทำให้อยู่ในรูปของอัลกอริทึมไม่ได้ จึงต้องหาวิธีที่จะสามารถนำไปใช้ให้อยู่ในรูปที่คอมพิวเตอร์สามารถนำไประมวลผลได้ หัวข้อหลักทั่วไปที่มีการค้นคว้าและวิจัยกันทางด้านวิศวกรรมความรู้คือ การแสดงความรู้ (Knowledge representation) กลไกการอนุมาน (Inference mechanism) การรับและจัดการความรู้ (Knowledge acquisition) และการติดต่อกับผู้ใช้ (User interface)

ศาสตราจารย์ Edward A. Feigenbaum ได้อธิบายความหมายของวิศวกรรมความรู้ (knowledge engineering) ไว้ดังนี้

"วิศวกรทางด้านความรู้ (knowledge engineering) จะเป็นผู้ที่เอาหลักการ และเครื่องมือของงานวิจัยปัญญาประดิษฐ์มาใช้กับปัญหาของงานที่ยาก ซึ่งต้องอาศัยความรู้ของผู้เชี่ยวชาญเพื่อใช้แก้ปัญหานั้น หัวข้อทางเทคนิคของการได้มาซึ่งความรู้ การแทนความรู้ และการใช้สิ่งเหล่านั้นให้เหมาะสมกับการสร้าง และอธิบายการอ้างอิงเหตุผล ซึ่งส่วนนี้เป็นปัญหาสำคัญในการออกแบบฐานความรู้ของระบบผู้เชี่ยวชาญ ศิลปะในการสร้างตัวแทนของความฉลาดนี้เป็นทั้งส่วนหนึ่งของโปรแกรม และเป็นส่วนที่เพิ่มเติมเข้าไป ซึ่งเป็นศิลปะของการสร้างโปรแกรมที่มีความซับซ้อน เพื่อใช้แทนหรืออ้างอิงถึงความรู้ที่มีอยู่"

<u>โครงสร้างระบบผู้เชี่ยวชาญ</u>

ระบบผู้เชี่ยวชาญจะต้องประกอบด้วยส่วนสำคัญ 6 ส่วนดังนี้

1. ฐานความรู้ (Knowledge base) เป็นส่วนที่ใช้เก็บสะสมความรู้เบื้องต้นใน domain knowledge หรือขอบเขตของความรู้ในระบบผู้เชี่ยวชาญที่ต้องการสร้างขึ้นในส่วนนี้เปรียบเสมือนกับข้อมูลในโปรแกรมธรรมดาหรือฐานข้อมูล ต่างกันตรงที่ฐานความรู้ประกอบด้วย

1.1 ข้อเท็จจริง และกฎต่าง ๆ (facts and rules)

1.2 สมมติฐาน และการเชื่อ (assumptions and beliefs)

ความรู้ที่เก็บสะสมอยู่นี้ คือสิ่งที่เกื้อหนุนให้ระบบผู้เชี่ยวชาญสามารถทำงานได้ใกล้เคียงผู้เชี่ยวชาญ โดยปกติแล้วความรู้เหล่านี้จะเก็บอยู่ในรูปของความจริง และกฎ อย่างไรก็ตามกรรมวิธีการเก็บความรู้เหล่านี้ก็อาจจะแตกต่างกันไปตามวิธีการแสดงความรู้ (Knowledge representation) ในระบบผู้เชี่ยวชาญแต่ละประเภท

การรวบรวมความรู้ทั้งหลายจากผู้เชี่ยวชาญเฉพาะด้าน แล้วนำมาจัดเรียงให้อยู่ในรูปแบบที่เหมาะสมนับเป็นปัญหาที่สำคัญที่สุดในการออกแบบระบบผู้เชี่ยวชาญ และหน้าที่ดังกล่าวนี้จะเป็นของวิศวกรความรู้ซึ่งจะต้องใช้ความสังเกต พูดคุย และทำงานร่วมกับกลุ่มผู้เชี่ยวชาญ เพื่อที่จะกำหนดกรรมวิธีในการหาเหตุผลเพื่อแก้ปัญหาของผู้เชี่ยวชาญให้อยู่ในรูปแบบที่คอมพิวเตอร์สามารถเข้าใจได้ แล้วจึงทำการถ่ายทอดลงในโปรแกรม แต่ในขณะนี้ยังไม่มีทฤษฎีโดยทั่วไป ในการสร้างฐานความรู้ และออกแบบระบบผู้เชี่ยวชาญ ดังนั้นในการสร้างระบบผู้เชี่ยวชาญ วิศวกรความรู้จะต้องเป็นผู้พิจารณาว่า จะควรเลือกใช้เทคนิคอย่างไร หลักการที่มักใช้กันทั่วไปคือ การเรียบเรียงข้อมูลที่คาดว่าจะเป็นไปได้มากที่สุดเอาไว้ใส่วนแรก ๆ ของโปรแกรม และจัดข้อมูลที่

คาดว่าจะเป็นไปได้น้อยที่สุด หรือมีการเรียกใช้น้อยครั้งเอาไว้ในส่วนท้ายสุดของโปรแกรม อย่างไรก็ตามในบางครั้งแม้แต่ผู้เชี่ยวชาญเองก็ไม่สามารถจะเรียงลำดับความสำคัญของข้อมูลตามลักษณะดังที่กล่าวมานี้ได้ ทั้งนี้เนื่องจากระบบผู้เชี่ยวชาญมักจะประกอบด้วยข้อมูลจำนวนมากอยู่ในฐานความรู้ ฉะนั้นหลักการดังกล่าวอาจจะทำไม่ได้เลยในทางปฏิบัติ วิธีการที่จะช่วยให้การพัฒนาระบบเป็นไปอย่างมีประสิทธิภาพอีกวิธีหนึ่งคือ การทดลองใช้งานระบบผู้เชี่ยวชาญที่สร้างขึ้นเป็นจำนวนหลาย ๆ ครั้ง ๆในหลาย ๆ เป้าหมาย แล้วสังเกตดู หรือบันทึกข้อมูลแต่ละส่วนที่ถูกเรียกใช้จากนั้นจึงจัดลำดับข้อมูลในฐานะความรู้ใหม่ จนกว่าจะได้โปรแกรมที่ทำงานได้อย่างมีประสิทธิภาพ

2. กลไกวินิจฉัย (inference engine) หรือเรียกได้อีกอย่างหนึ่งว่า เครื่องอนุมาน หมายถึงส่วนประกอบของระบบผู้เชี่ยวชาญที่ทำหน้าที่วินิจฉัยปัญหาที่ต้องการแก้ไข จะทำหน้าที่ควบคุมการใช้ความรู้ในฐานความรู้เพื่อแก้ไขปัญหาอย่างมีประสิทธิภาพ โดยใช้ข้อมูลในฐานความรู้จนกว่าจะพบคำตอบ หรือจนกว่าจะหาคำตอบไม่พบอันเนื่องจากฐานความรู้มีไม่เพียงพอ

วิธีการสร้างกลไกวินิจฉัย ในการพัฒนาระบบผู้เชี่ยวชาญมีวิธีการพื้นฐานที่นิยมใช้ในการสร้างกลไกวินิจฉัยทั้ง 2 ประเภทดังกล่าวข้างต้น 2 วิธีด้วยกันคือ

2.1. Backward-Chaining Method กลไกวินิจฉัยวิธีนี้จะเริ่มต้นที่กำหนดเป้าหมายหลักของระบบก่อน จากนั้นจึงพยายามที่จะค้นหาข้อมูลที่จะสนับสนุนให้เป้าหมายนี้เป็นจริง

2.2. Forward-Chaining Method การวินิจฉัยวิธีนี้มีลักษณะตรงข้ามกับวิธี Backward-Chaining กล่าวคือ แทนที่จะสมมุติเป้าหมายแล้วพยายามค้นหาคุณลักษณะเพื่อสนับสนุนเป้าหมายนั้น วิธี Forward-Chaining กลับใช้ความรู้หรือความจริงที่มีอยู่ หรือถามคำถามจากผู้ใช้เพื่อให้ทราบคุณลักษณะ และความจริงทั้งหมดเสียก่อน แล้วใช้ประโยชน์จากคุณลักษณะเหล่านั้นในการหาทางเดินเข้าสู่เป้าหมายที่สอดคล้องกับคุณลักษณะที่ได้มา โดยการเดินทางสู่เป้าหมายจะกระทำไปตามความสมบูรณ์ของกฎเกณฑ์ต่างๆ ที่มีอยู่ในฐานความรู้

3 ส่วนดึงความรู้ (Knowledge Acquisition) เป็นส่วนที่ทำหน้าที่ดึงความรู้จากผู้เชี่ยวชาญ เอกสาร หนังสือ หรือฐานข้อมูล

หลักการทำงานของส่วนดึงความรู้นั้นจะมีหน้าที่ใหญ่ ๆ อยู่ 2 ประการคือ

ก. เป็นหน่วยรับความรู้เช่น กฎเกณฑ์ต่าง ๆ จาก ผู้เชี่ยวชาญ หรือจากวิศวกรความรู้ แล้วนำความรู้ที่ได้เหล่านี้ส่งต่อให้กลไกวินิจฉัยเพื่อนำไปใช้ในการวินิจฉัยต่อไป

ข. ทำหน้าที่ติดต่อกับผู้ใช้เพื่อรับข้อมูลที่ผู้ใช้ต้องการที่จะปรึกษา มาทำการประมวลผลร่วมกับความรู้ที่มีอยู่ในฐานความรู้

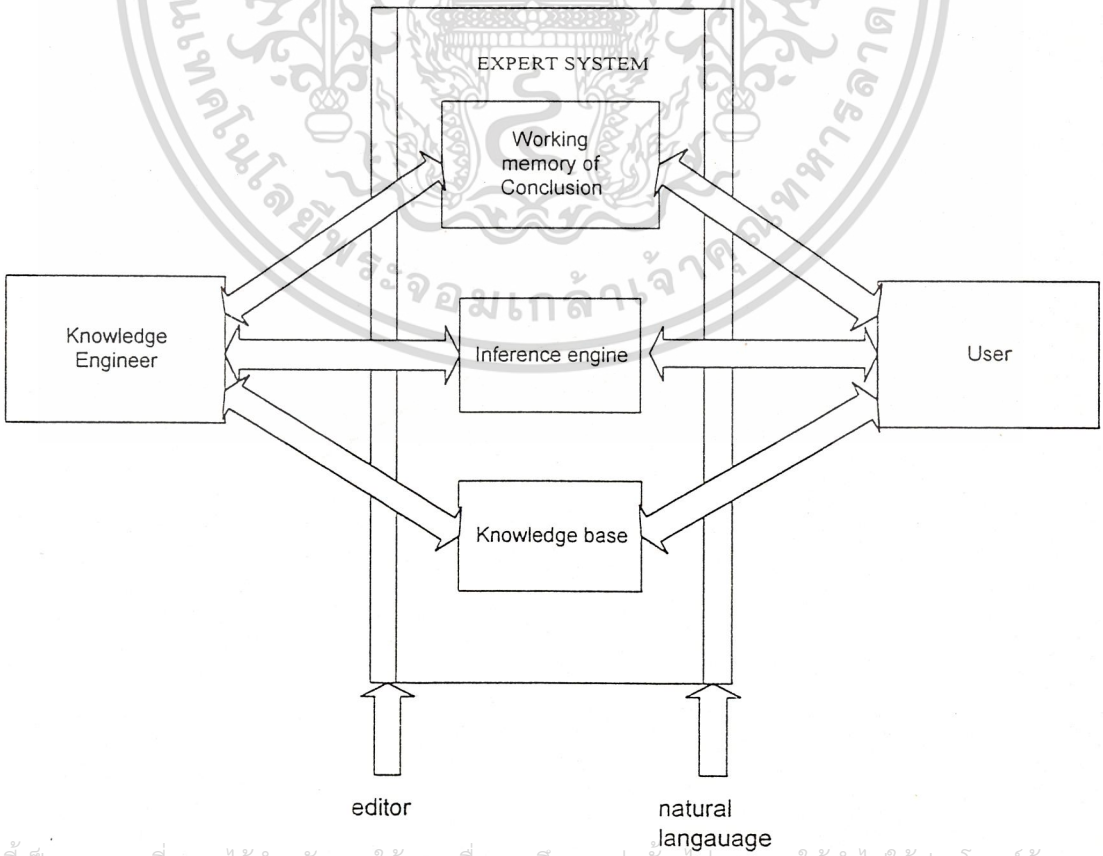
4. ส่วนอธิบาย (Explanation) เป็นส่วนที่ทำหน้าที่อธิบาย และให้เหตุผลแก่ผู้ใช้งาน ในขณะที่กำลังใช้งานนั้นอยู่ เช่นให้เหตุผลแก่ผู้ใช้งานว่าทำไมระบบผู้เชี่ยวชาญจึงได้ตั้งคำถามนั้นขึ้นมา และคำถามนั้นมีความเกี่ยวข้องกับความรู้ในฐานข้อมูลอย่างไรบ้าง เป็นต้น การให้คำอธิบายนี้จะเป็นลักษณะเดียวกันกับที่ผู้เชี่ยวชาญ จะให้คำอธิบายเมื่อมีผู้มาขอคำปรึกษา

5. ส่วนติดต่อกับผู้ใช้ (User Interface) จะทำหน้าที่เป็นตัวกลางระหว่างผู้ใช้โปรแกรมกับระบบผู้เชี่ยวชาญ เพื่อให้การสื่อสารระหว่างทั้งสองฝ่ายเป็นไปอย่างราบรื่น ส่วนติดต่อกับผู้ใช้ที่ดีจะต้องอำนวยความสะดวกต่อผู้ใช้ให้ได้มากที่สุด

6. หน่วยความจำของระบบ (Working Memory) เป็นส่วนที่ใช้เก็บข้อมูลในระหว่างการทำงานของระบบ ซึ่งข้อมูลนี้จะประกอบด้วยฐานความรู้ซึ่งเป็นข้อมูลหลักของระบบซึ่งประกอบด้วยข้อเท็จจริง และกฎต่าง ๆ และส่วนของข้อมูลที่ได้จากส่วนดึงความรู้และส่วนติดต่อกับผู้ใช้ ซึ่งเป็นข้อมูลที่จะต้องมีการรับส่งกันระหว่างระบบกับผู้ใช้

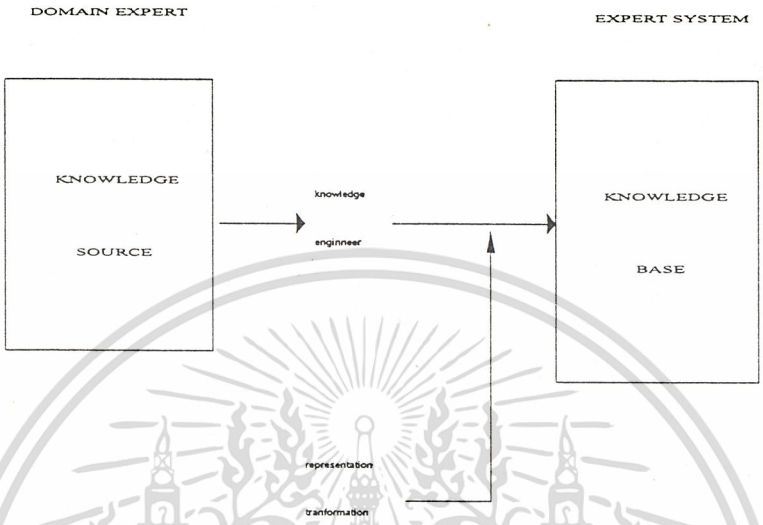
โครงการสร้างของระบบผู้เชี่ยวชาญตามที่ได้กล่าวมาทั้งหมดสามารถแสดงได้ดังภาพที่ 11

ภาพที่ 11



แสดงโครงสร้างของระบบผู้เชี่ยวชาญ

ภาพที่ 12



แสดงขบวนการดึงความรู้ (Knowledge acquisition facility)

ในทางปฏิบัติแล้วระบบผู้เชี่ยวชาญไม่จำเป็นต้องมีส่วนประกอบครบทุกส่วนก็ได้ หรือผู้สร้างระบบบางรายอาจรวมส่วนประกอบบางส่วนที่ทำหน้าที่คล้ายกันเข้าด้วยกันก็ได้ เช่น ส่วนดึงความรู้ และส่วนติดต่อกับผู้ใช้ แต่สิ่งที่สำคัญของระบบผู้เชี่ยวชาญคือ ทุกระบบนั้นจะต้องประกอบด้วย ฐานความรู้ และกลไกวินิจฉัย เป็นอย่างน้อย

<u>การแทนความรู้ของระบบผู้เชี่ยวชาญ</u>

การแทนความรู้ (Knowledge Representation) คือ ลักษณะการจัดรูปแบบของฐานความรู้เพื่อจัดเก็บบนเครื่องคอมพิวเตอร์ การเลือกหรือออกแบบลักษณะของการแทนความรู้นี้มีผลกระทบอย่างมากต่อการออกแบบชนิดของกลไกวินิจฉัย และประสิทธิภาพโดยส่วนรวมของระบบผู้เชี่ยวชาญ

แบ่งประเภทของความรู้ได้เป็น 4 ประเภทคือ

1. ความรู้ที่บอกความจริง, ลักษณะหรือคุณสมบัติ เช่น ทางสายนี้ยาว 10 กม.
2. ความรู้ที่บอกความสัมพันธ์ เช่น ปลาวาฬเป็นสัตว์เลี้ยงลูกด้วยนม
3. ความรู้ที่บอกขั้นตอนหรือวิธีการ เช่น ถ้ามาตรวัดอุณหภูมิมีเข็มชี้ไปที่ High ให้ปิดเครื่องและตรวจสอบระดับน้ำในหม้อน้ำ

4. ความรู้ที่เกี่ยวกับความรู้ (meta knowledge) เป็นความรู้ที่เกี่ยวกับคุณลักษณะของความรู้อื่น หรือเกี่ยวกับวิธีการใช้ความรู้อื่น อาจมองได้ว่าเป็นความรู้พื้นฐาน หรือ sense ที่มนุษย์เรามีอยู่

<u>คุณสมบัติของวิธีการแสดงความรู้ที่ดี</u>

1. มีความสามารถในการแสดงความรู้ชนิดต่าง ๆ ได้ เช่น ต้องสามารถบันทึกความรู้ทั้งที่มีโครงสร้าง และความรู้ที่มีความไม่แน่นอน โดยการใช้โครงสร้างชนิดเดียวกัน และถ้าหากเป็นไปได้โครงสร้างที่ใช้ในการแสดงความรู้จะต้องเป็นโครงสร้างที่ง่าย แต่มีความสามารถในการแสดงความรู้สูง

2. มี modularity กล่าวคือ ความสามารถในการแยกออกเป็นข้อย่อย (module) ทั้งนี้เพื่อให้สมารถเพิ่มหรือแก้ไขฐานความรู้ คุณสมบัติอันนี้จำเป็นเพราะทำให้เกิดความยืดหยุ่นในการใช้ความรู้

3. ง่ายต่อการจัดการ กล่าวคือ เป็นคุณสมบัติที่ช่วยในการตรวจสอบฐานความรู้ อย่างเช่นช่วยในการตรวจดูความขัดแย้งในความรู้ การซ้ำกัน หรือความผิดพลาดในความรู้

4. ง่ายต่อการเข้าใจของมนุษย์ การแสดงความรู้ที่ดีนอกจากเข้ากับคอมพิวเตอร์ได้แล้วยังจะต้องให้เข้ากับมนุษย์ได้ดีด้วย คุณสมบัติอันนี้ช่วยทำให้การสร้างส่วนอธิบายในระบบผู้เชี่ยวชาญง่ายขึ้น นอกจากนั้นยังช่วยในการตรวจความผิดพลาดในการเพิ่มความรู้เข้าไปในฐานความรู้ด้วย

5. เข้ากันได้ดีกับการอนุมาน ทั้งนี้เนื่องจากการอนุมานต้องใช้ความรู้ในฐานความรู้เป็นข้อมูล ดังนั้นเพื่อที่จะให้การอนุมานมีประสิทธิภาพดี การแสดงความรู้จะต้องเข้ากันได้ดีกับการอนุมาน

ตัวอย่างของการแสดงความรู้ที่ใช้กันอยู่ในระบบผู้เชี่ยวชาญในปัจจุบัน

<u>การแสดงความรู้ในรูปของกฎ (Rule Base System)</u>

การแสดงความรู้ในรูปของกฎ หรือเรียกอีกอย่างว่า การแสดงความรู้ในรูป Production system โดยที่ความรู้ ขั้นตอนการปฏิบัติ และการประมวลผลจะถูกบันทึกอยู่ในให้อยู่ในรูปเซ็ทของกฎ การประมวลผลกฎจะเลือกกระทำตามความสมบูรณ์ และถูกต้องของเงื่อนไขในแต่ละกฎโดยจะมีการอนุมานเป็นคอยตัวควบคุมการเลือกใช้กฎอีกทีหนึ่ง

กฎ ใน Rule Base จะอยู่ในรูป

IF..Permise THEN Action

ส่วนของ IF เรียกว่า ส่วนเงื่อนไข ซึ่งจะมี premise เป็นเงื่อนไขของกฎนั้น และส่วนของ THEN เรียกว่า ส่วนข้อสรุปหรือส่วนการปฏิบัติ ซึ่งจะมี action เป็นตัวปฏิบัติการของกฎนั้น
ตัวอย่างบางส่วนของกฎเกี่ยวกับการออกแบบฐานข้อมูล

IF FD มีความซ้ำซ้อน
THEN ประมวลผลขบวนการตัด FD ที่ซ้ำซ้อน

IF determinant ไม่ได้เป็น Key
THEN ประมวลผลขบวนการแยกตาราง

ส่วนประกอบในการทำงานของ Rule base จะประกอบด้วยส่วนสำคัญ 3 ส่วนคือ

1. ส่วนฐานความรู้ หรือเรียกได้อีกอย่างว่าฐานกฎ ซึ่งจะประกอบด้วยกฎต่าง ๆ ที่เกี่ยวกับความรู้ที่จะใช้แก้ปัญหาในระบบนั้น
2. ส่วน Inference เป็นส่วนกลไกในการอนุมาน
3. ส่วน working memory (WM)

อธิบายการทำงานของระบบได้ดังนี้ ในระหว่างการใช้งานระบบ WM จะเป็นตัวเก็บเก็บข้อมูล และสถานะต่าง ๆ ของระบบในขณะนั้น โดยที่จะนำข้อมูลที่อยู่ในส่วนของ WM ไปเป็นตัวตรวจสอบเงื่อนไขในส่วนของ IF clause เมื่อได้กฎที่เหมาะสมก็จะกระทำในส่วนของ action คือส่วนของ Then ซึ่งจะมีผลทำให้สถานะของ WM เปลี่ยนไป จะทำเช่นนี้ไปจนกว่าจะได้คำตอบที่ต้องการ ซึ่งจะมีส่วน Inference คอยควบคุมการเลือกกฎจากฐานความรู้ที่มีเงื่อนไขครบ ตรงตามสถานะของ WM ขึ้นมาปฏิบัติการ ซึ่งขั้นตอนในการปฏิบัติมีดังนี้

1. Matching : ตรวจสอบดูความรู้ใน WM และ ฐานความรู้ เพื่อหากฎทั้งหมดที่มีเงื่อนไขพร้อมจะทำการปฏิบัติการ

2. Conflict resolution : จากกฎที่ได้จากการ matching จะเลือกกฎที่เหมาะสมขึ้นมา 1 กฎ โดยมีวิธีการดังนี้

หลังจากการทำ matching แล้ว อาจจะมีกฎที่มีความถูกต้องของเงื่อนไขครบสมบูรณ์มากกว่าหนึ่งกฎ จึงจำเป็นจะต้องเลือกกฎใดกฎหนึ่งขึ้นมา วิธีการเลือกกฎที่ใช้กันทั่วไปทำได้หลายวิธีดังนี้

2.1. เลือกตามลำดับความสำคัญของกฎ วิธีการนี้จะนำกฎมาทำการ matching ตามลำดับความสำคัญของกฎ กฎแรกที่ทำการ matching ประสบความสำเร็จจะได้รับการปฏิบัติการทันที

2.2. เลือกตามลำดับความละเอียดของส่วน IF ของกฎ : กฎที่ส่วน IF มีการบันทึกเงื่อนไขไว้ละเอียด หรือเฉพาะที่สุดจะได้รับเลือกก่อน เช่นถ้ามี 3 กฎ

กฎแรก IF X เป็นผู้หญิง THEN มีสิทธิที่ใช้ขบวนการหย่า

กฎที่สอง IF X เป็นผู้หญิง และ จดทะเบียนสมรสแล้ว THEN มีสิทธิที่ใช้ขบวนการหย่า

กฎที่สาม IF X เป็นผู้หญิง และ จดทะเบียนสมรสแล้ว และ สามียินยอมให้หย่า THEN มีสิทธิที่ใช้ขบวนการหย่า

จากกฎทั้ง 3 กฎนี้ กฎที่สาม จะถูกเลือกก่อนเพราะส่วน IF ของกฎที่สาม มีข้อมูลละเอียดเฉพาะมากกว่าของกฎที่หนึ่ง และที่สอง

2.3. เลือกตามความใหม่ของกฎ : กฎที่ได้รับการปฏิบัติการล่าสุดจะได้รับการเลือกก่อน

2.4. เลือกตามความใหม่ของข้อมูลใน WM : เลือกกฎที่ match ข้อมูลล่าสุดใน WM ไว้ก่อน

2.5. แบบขนาน : ให้กฎทุกกฎที่ match เสร็จ (ส่วนเงื่อนไขสอดคล้อง) ปฏิบัติการพร้อมกัน

3. Action : ปฏิบัติการตามส่วน THEN ของกฎที่ได้จากการคัดเลือกในข้อ 2 ซึ่งบางการปฏิบัติการอาจจะเป็นการเปลี่ยนเนื้อหาของ WM

<u>ทิศทางการอนุมาน</u>

ใน Rule Base ส่วนใหญ่ใช้วิธีการอนุมานสองวิธีคือ 1. การอนุมานแบบเดินหน้า (forward chaining) 2. การอนุมานแบบย้อนหลัง (backward chaining) ซึ่งรายละเอียดมีกล่าวไว้ในตอนต้น

<u>ข้อดีของ RULE BASE SYSTEM</u> [23]

Modularity of Knowledge เนื่องจากกฎแต่ละกฎมีความสมบูรณ์ในตัวเอง และเป็นอิสระจากกันในฐานความรู้ การสืบหาค่าความจริงเพื่อที่จะสรุปปัญหาหรือการกระทำต่าง ๆ ที่อยู่ในส่วนของ THEN สามารถพิจารณาได้จากในส่วนของ IF ซึ่งแยกจากส่วนของ THEN ความเป็นอิสระจากกันของฐานความรู้นี้ทำให้ง่ายต่อการตรวจสอบความถูกต้อง

Natural Expression ปัญหาส่วนมากสามารถแสดงได้ในรูปของ IF..THEN... ซึ่งการแสดงความรู้ในรูปนี้เป็นรูปแบบที่มนุษย์เราค่อนข้างจะคุ้นเคย จึงเป็นการง่ายต่อการเข้าใจ และง่ายต่อการสร้างฐานความรู้

ข้อเสียของ RULE BASED SYSTEM [23]

กระบวนการควบคุมความรู้มักจะไปเกี่ยวข้องกับฐานความรู้หลักเช่น ลำดับของกฎในฐานความรู้อาจมีความสำคัญ Rule base ไม่มีโครงสร้างที่แท้จริง ซึ่งขึ้นอยู่กับความสัมพันธ์ระหว่าง rule ทำให้ยากในการแก้ไขถ้าฐานความรู้มีขนาดใหญ่ และไม่สามารถกำหนดความสัมพันธ์กันของความรู้ได้อย่างชัดเจน

ในกรณีที่ฐานข้อมูลความรู้มีขนาดใหญ่มี RULE มาก การทำงานของระบบอาจทำได้ช้า เพราะกลไกการวินิจฉัยจะต้องใช้เวลาในการค้นหา RULE ที่จะกระทำจาก RULE ที่มีอยู่ทั้งหมด

การแสดงความรู้โดยใช้ตรรกศาสตร์ (LOGIC)

ในส่วนนี้จะกล่าวถึงการใช้ first-order predicate logic (FOL) ซึ่งทฤษฎีหนึ่งทางด้านตรรกศาสตร์มาเป็นตัวอธิบายการแสดงความรู้ และการอนุมาน โดยจะใช้ well-formed formula (WFF) และ clause มาเป็นตัวแทนความรู้

WFF คือ

1. atomic formula หรือ
2. WFF ที่เชื่อมด้วย logic symbol หรือ
3. WFF ที่มี quantifier

atomic formula คือ

ถ้าให้ P เป็น predicate symbol และ t1,...,tn เป็น P(t1,...,tn) จะเป็น atomic formula

โดยที่ predicate ในตรรกวิทยาหมายจะถึง สิ่งที่ใช้กำหนดข้อเท็จจริง และความสัมพันธ์ของสิ่งต่าง ๆ ในโลก

ตัวอย่างของ predicate เรามีข้อเท็จจริงอยู่ว่า สมปองเป็นน้องสมชาย เราสามารนำมาเป็น predicate ในรูปของ FOL ได้ดังนี้

น้องชาย(สมปอง,สมชาย)

จาก predicate นี้เราสามารถเขียนให้อยู่ในรูปของตัวแปรได้ดังนี้

P (x , y)

โดย P เป็น predicate symbol น้องชาย และ x,y เป็น variable symbol

logic symbol มีดังนี้

~	:	not
$\wedge$	:	and
$\vee$	:	or
$\equiv$	:	equivalent
$\rightarrow$	:	emply

quantifier มีดังนี้

$\forall$	:	universal quantifier หมายถึง ทั้งหมด(ทุกสิ่ง)
$\exists$	:	existentional quantifier หมายถึง บางอย่าง

ตัวอย่างของ WFF

$(\forall x)$ [คน (x) -> สัตว์เลี้ยงลูกด้วยนม (x)]

หมายความว่า สำหรับ x ทุกค่าแล้ว ถ้า x เป็นคนแล้ว x จะเป็นสัตว์เลี้ยงลูกด้วยนม

$(\exists x)$ [เสือ (x) สี (x, ขาว)]

หมายความว่าเสือบางตัวมีสีขาว

clause

clause คือ literal ที่เชื่อมด้วย $\vee$

literal คือ P(x1,...,xn) หรือ ~P(x1,...,xn) โดยที่ P(x1,...,xn) เป็น atomic formula

ตัวอย่างของ clause คือ

กำหนดให้ P1, P2, Q1,...Qm เป็น atomic formula

จะได้ clause $P1 \vee P2 \vee \sim Q1 \vee \ldots \vee \sim Qn$

และจากที่มีสูตรว่า $(P \vee \sim Q) \equiv (Q \rightarrow P)$

ดังนั้นจาก clause ข้างต้นสามารถเขียนได้เป็น $Q1 \wedge \ldots \wedge Qm \rightarrow P1 \vee P2$

สามารถพิสูจน์ว่าทุก WFF สามารถแปลงเป็น clause หรือ clause ที่เชื่อมด้วย $\wedge$ ได้ ดังนั้น การแสดงความรู้ด้วยตรรกวิทยาก็คือ การแสดงความรู้ในรูปของ clause ต่าง ๆ นั่นเอง

มี clause ที่รู้จักกันแพร่หลายอยู่ชนิดหนึ่งคือ Horn clause ในที่นี้จะกล่าวถึงพอสังเขป เพราะจะมีการนำ Horn clause ไปใช้ในส่วนของการลดความซ้ำซ้อนของความรู้ในฐานความรู้ในบทที่ 7 ในเรื่องการออแบบฐานความรู้

Horn clause คือ clause ที่มี literal ที่เป็นบวก (คือ atomic formula ที่ไม่มี ~) อย่างมากที่สุดเพียงตัวเดียว เราเรียกว่า Horn clause

$P1 \vee \sim Q1 \vee ... \vee \sim Qm$

คือ Horn clause ซึ่งมีค่าเท่ากับ

$Q1 \wedge ... \wedge Qm \rightarrow P1$

$\sim Q1 \vee ... \vee \sim Qm$

ก็เป็น Horn clause ซึ่งมีค่าเท่ากับ

$Q1 \wedge ... \wedge Qm$

<u>การอนุมาน</u>

จะใช้หลักการ resolution ซึ่งเสนอโดย J.A.Robinson (1965) หลักการสำคัญของขบวนการ resolution คือ จะแปลงความรู้ทั้งหมดให้อยู่ในรูป clause จากนั้นทำการตรวจสอบค่าความจริงของ clause นั้น

<u>ข้อดี และข้อเสียของการแสดงความรู้ด้วยตรรกศาสตร์</u>

ข้อดีคือ การแสดงความรู้โดยใช้ตรรกศาสตร์มีรากฐานทางทฤษฎีของคณิตศาสตร์รองรับ จึงทำให้มีกฎข้อบังคับ และวิธีการที่แน่นอน สามารถพิสูจน์ออกมาในรูปของทฤษฎีทางคณิตศาสตร์ได้ ซึ่งเป็นคุณสมบัติที่เหมาะสมในการแทนความรู้

ข้อเสีย มีความรู้บางอย่างที่ยากที่จะแทนได้ด้วย predicate ในตรรกศาสตร์ เช่น วันนี้อากาศร้อนมาก หรือคนที่มีผมสีบลอนด์มักจะมีตาสีฟ้าเป็นต้น

การแสดงความรู้โดยใช้เฟรม (Frame)

การแสดงการเขียนความรู้แบบเฟรมถูกพัฒนาขึ้นมาในปี พ.ศ. 2517 โดย มินสกี้ (MINSKY) โดยเฟรมจะเก็บความรู้ในรูปของกลุ่มความรู้ที่มีความสัมพันธ์กัน โครงสร้างของเฟรมง่ายต่อการเก็บความรู้พื้นฐานทั่วไป และเหมาะกับการแสดงความเป็นเหตุเป็นผลกัน ซึ่งความรู้ภายในเฟรมอาจเป็นกฎเกณฑ์ต่างๆ , คำอธิบายหรือข้อสังเกต , การแก้ไขเปลี่ยนแปลง หรือคำถามที่จะถามผู้ใช้ในขณะกำลังใช้ระบบก็ได้ โครงสร้างของเฟรมแสดงได้ดังภาพที่ 13

โครงสร้างข้อมูลพื้นฐานของเฟรม

ภาพที่ 13

<Frame>	<Frame name>		
	<Slot name>	<Slot value>	
	<Slot name>	<slot value>	
		<Facet name>	<Facet value>
		<Facet name>	<Facet value>
	.....	.....	.....
	.....	.....	.....

แสดงโครงสร้างของเฟรม

จากภาพที่ 13 Frame name เป็นชื่อของเฟรม โดยปกติแล้วจะตั้งชื่อให้สื่อความหมายกับข้อมูลความรู้ภายในเฟรม Slot name , Slot value จะเป็นตัวแปร และค่าของตัวแปรสำหรับเก็บความรู้ภายในเฟรม ส่วน Facet name , Facet value เป็นส่วนที่ขยายของ Slot เพื่อความเข้าใจยิ่งขึ้น พิจารณาได้จากตัวอย่างต่อไปนี้

สมมติว่ามีประโยคต่อไปนี้ “ช้างเป็นสัตว์เลี้ยงลูกด้วยนมชนิดหนึ่งมีสีเทา มีสี่ขา กินพืชเป็นอาหาร” ประโยคนี้แสดงโดยใช้เฟรมได้ดังนี้

frame : ช้าง

is a : สัตว์เลี้ยงลูกด้วยนม

สี : เทา

ขา : 4

อาหาร : พืช

slot "is-a" เป็นตัวบอกความสัมพันธ์ระหว่างเฟรมนี้กับเฟรมอื่น ๆ จากตัวอย่างข้างต้น ชื่อเฟรม "ช้าง" ซึ่งเป็นตัวอย่างหนึ่งของเฟรมชื่อ "สัตว์เลี้ยงลูกด้วยนม" ซึ่งเป็นเฟรมที่อยู่ในระดับสูงกว่า

<u>การขยาย slot</u>

พื้นฐานการอนุมานในระบบเฟรม อยู่ที่การกำหนดค่าสล็อตที่ยังไม่รู้ภายในเฟรมที่กำลังพิจารณาอยู่ ขอกยกตัวอย่างแสดงการอธิบายดังนี้

ในเฟรมยังมีส่วนของการขยาย slot เพื่อช่วยให้การกำหนดค่าของ slot เป็นไปอย่างมีประสิทธิภาพยิ่งขึ้น โดยจะใช้ slot ย่อยเรียกกันว่า facet มาเป็นส่วนขยายการกำหนดค่าให้ slot ดังแสดงในรูปที่

facetนั้นมีหลายชนิดด้วยกัน แต่ที่เป็นหลัก ได้แก่

(1) value facet : แสดงค่าของ slot

(2) data-type facet : แสดงชนิดข้อมูลของ slot

(3) rage facet : แสดงขอบข่ายที่ค่า slot จะเป็นไปได้

(4) default facet : แสดงค่าดีเฟาล์ตของ slot

ตัวอย่างต่อไปนี้แสดงวิธีใช้ facet

frame : รถยนต์ส่วนบุคคล

self : value : (a kind of รถยนต์)

manufacturer : value:

data-type : แถวตัวอักษรไทย

range : [เบ็นซ์ โวลโว เปอร์โยต์ โตโยต้า นิสสัน มิซูบิชิ ฮอนด้า]

default : โตโยต้า

color : value :

data-type: แถวตัวอักษรไทย

range : [ขาว แดง ฟ้า เขียว เหลือง เขียว ดำ]

default : ขาว

data type facet และ range facet สามารถใช้เป็นข้อมูลในการตรวจดูว่าค่าของ slot ที่พิมพ์เข้ามานั้นผิด หรือไม่มีในเซ็ท จากตัวอย่างข้างต้นแสดงการบันทึก range facet แบบ explicit ที่จริงแล้วโดยทั่วไปจะมีการบันทึก range facet ในแบบ implicit เช่นบอกเป็น ฟังก์ชั่นเงื่อนไขที่ใช้ควบคุมค่าของ slot

<u>ขบวนการประกอบ (procedure attachment)</u>

เราสามารถใช้ facet ในอีกรูปแบบหนึ่งได้โดยการบันทึกขั้นตอน หรือขบวนการในการอ้างอิงหรือเปลี่ยนแปลงค่าของ slot และเมื่อจำเป็นก็เรียก facet มาใช้ facetประเภทนี้มีชื่อเรียกว่า facet ขบวนการประกอบ มีอยู่หลายชนิดด้วยกัน แต่เป็นหลัก ๆ ได้แก่

1. if-needed facet ในกรณีที่มีการเรียกร้องให้กำหนดค่าของ slot ขบวนการที่ถูกบันทึกในfacet นี้จะถูกเรียกโดยอัตโนมัติเพื่อปฏิบัติการหาค่าของ slot ผลที่ได้จะถูกเขียนเข้าไปใน value facet ของ slot นั้น

ตัวอย่างพิจารณาเฟรมต่อไปนี้

frame : สมชาย

self : value : (a member or คนไข้)

sex : value : ชาย

temp : value :

if-needed : (measure temperature by thermometer)

เพื่อที่จะหาค่าอุณหภูมิร่างกายของสมชาย จะมีการบันทึกขั้นตอนการหาว่า "ให้ใช้เครื่องวัดอุณหภูมิร่างกาย" ค่าที่ได้จะใส่ใน value facet ของ slot temp

2. if-added facet ขบวนการที่ถูกบันทึกอยู่ใน facet นี้จะปฏิบัติการโดยอัตโนมัติทันทีเมื่อมีค่าเข้ามาใน value facet ของ slot เดียวกันกับ if-added facet

ตัวอย่างพิจารณาเฟรมต่อไปนี้

frame : สมชาย

self : value : (a member or มนุษย์)

birth : value : 20/5/2515

if-added : (compute age)

age : value :

เฟรมข้างต้นแสดงค่า value-facet ของ slot birth เพิ่งถูกใส่เข้ามาในตอนนั้น ขั้นตอน หรือขบวนการที่ถูกบันทึกอยู่ในค่าของ if-added facet จะปฏิบัติโดยการอัตโนมัติทันที ผลที่ได้คืออายุจะถูกบันทึกเข้าไปใน value facet ของ slot age

3. if-removed facet จะทำในสิ่งตรงกันข้ามกับ if-added facet กล่าวคือเมื่อค่าของ value facet ถูกลบออกไปขบวนการที่ถูกบันทึกอยู่ใน facet นี้จะปฏิบัติการโดยอัตโนมัติ

<u>การถ่ายทอดคุณสมบัติ</u>

เป็นคุณสมบัติที่สำคัญอย่างหนึ่งของเฟรม ในการใช้งานจริงส่วนใหญ่ในระบบจะประกอบด้วยเฟรมหลายเฟรม ซึ่งเรียกว่าระบบเฟรม (frame base system) ซึ่งแต่ละเฟรมจะมีความสัมพันธ์กัน ซึ่งจะเป็นความสัมพันธ์ในลักษณะของลำดับชั้น (Hierarchy) โดยใช้ slot “is-a” เป็นตัวกำหนดความสัมพันธ์ต่างระดับระหว่างเฟรม และจากความสัมพันธ์นี้ทำให้มี “การถ่ายทอดคุณสมบัติ” เกิดขึ้น กล่าวคือเฟรมที่อยู่ระดับต่ำกว่าจะมีคุณสมบัติของเฟรมที่อยู่ระดับสูงกว่า ซึ่งแสดงได้ดังตัวอย่างต่อไปนี้

สมมติว่ามีประโยคต่อไปนี้ “นักศึกษาโดยทั่วไปเป็นคนชอบอ่านหนังสือและมีมารยาทสุภาพ นักศึกษาวิชาวิศวกรรมศาสตร์ส่วนใหญ่มีลักษณะเคร่งขรึมและมีความสนในใจเรื่องเครื่องจักรกล สมชายเป็นนักศึกษาวิชาวิศวกรรมศาสตร์คนหนึ่งที่มีนิสัยร่าเริงแต่ไม่ชอบการอ่านหนังสือส่วนความสนใจ และมารยาทของสมชายนั้นไม่ทราบ” ประโยคเหล่านี้แสดงโดยใช้เฟรมดังนี้

frame	:	นักศึกษา	
self	:	(a kind of อาชีพ)	
attitude	:	ชอบอ่านหนังสือ	(1)
manner	:	สุภาพ	
frame	:	นักศึกษาวิชาวิศวกรรมศาสตร์	
self	:	(a kind of นักศึกษา)	
character:		เคร่งขรึม	(2)
interest	:	เครื่องกล	

frame	:	สมชาย	
self	:	(a member of นักศึกษาวิชาวิศวกรรมศาสตร์)	
character:		ไม่ชอบอ่านหนังสือ	(3)
interest	:		
manner	:		

โปรดสังเกตที่เฟรมของสมชาย ใน (3) ค่าของslot interest และ manner ยังว่างอยู่แต่เราสามารถใช้ข้อมูลของเฟรมที่อยู่ในระดับที่สูงกว่ามาเติมได้ เนื่องจากมีลักษณะการถ่ายทอดคุณสมบัติ กล่าวคือในกรณีของ slot interest จาก (2) ซึ่งอยู่สูงกว่า (3) เราได้ "เครื่องกล" และในกรณีของ slot manner จาก (1) ซึ่งอยู่สูงกว่า (2) เราได้ "สุภาพ" ดังนั้นข้อมูลทั้งหมดเกี่ยวกับสมชายจะแสดงในเฟรมต่อไปนี้

farm	:	สมชาย
self	:	(a mamber of นักศึกษาวิชาวิศวกรรมศาสตร์)
character:		ร่าเริง
interest	:	เครื่องกล
attitude	:	ไม่ชอบอ่านหนังสือ
manner	:	สุภาพ

<u>กลไกการอนุมาน</u>

กลุ่มของเฟรมจะประกอบขึ้นเป็นฐานความรู้ ฐานความรู้นี้บวกกับกลไกการอนุมาน จะเป็นระบบฐานกรอบ (frame base system) หรือเรียกง่าย ๆ ว่าระบบเฟรม (frame system) พื้นฐานการอนุมานในระบบเฟรมคือ การกำหนดค่า slot ของเฟรมที่ถูกพิจารณา วิธีการเช่นนี้เรียกว่าเติม slot (filling in slot) ซึ่งจากจุดนี้จะนำไปสู่การเชื่อมต่อกันของระบบเฟรม เนื่องจากยังไม่มีการกำหนดวิธีการอนุมานที่แน่นอนในระบบเฟรม ส่วนใหญ่จะใช้วิธีพิจารณาลักษณะปัญหาเป็นกรณี ๆ ไป แล้วเลือกวิธีการอนุมานที่เหมาะสม นอกจากนั้นยังขึ้นอยู่กับเครื่องมือที่ใช้ในการพัฒนาอีกด้วย ตัวอย่างของการอนุมานในระบบเฟรมเช่น ในโปรแกรมเปลือกระบบผู้เชี่ยวชาญ PC/PLUS ใช้กลไกการอนุมานที่ค่อนไปทางกฎ เป็นการพยายามนำเอากลไกควบคุมในระบบ Rule base มาใช้งานกับระบบเฟรม โดยเพิ่มในส่วนของการถ่ายทอดคุณสมบัติตามลำดับชั้นของเฟรมเข้าไปด้วย เป็นต้น จากนั้นก็จะมีขบวนการ attach procedure เข้ามาใช้เพื่อช่วยเพิ่มความอิสระในการเรียกใช้งานเฟรม

<u>ข้อดีข้อเสียของเฟรม</u>

ข้อดี เป็นการแสดงความรู้ที่สามารถแสดงความรู้ได้อย่างชัดเจน การเชื่อมกันแบบลำดับชั้นทำให้สามารถแสดงความรู้หลักที่เป็นพื้นฐานได้ นอกจากนั้นยังสามารถแสดงความสัมพันธ์ของความรู้ได้ เฟรมสามารถแสดงความรู้ได้หลายประเภทนับตั้งแต่ความรู้ที่เป็นความจริง จนถึงความรู้ที่เป็นแบบขั้นตอน หรือขบวนการ ทำให้มีอิสระมากขึ้นในการแทนความรู้ ทั้งยังเป็นการแสดงความรู้ที่ประหยัด เพราะการแสดงความรู้แบบแบ่งเป็นระดับของเฟรมนั้นทำให้สามารถใช้ลักษณะการถ่ายทอดคุณสมบัติเพื่อประหยัดเนื้อที่ในการเก็บความรู้

ข้อเสีย ปัญหาในการตรวจสอบความรู้ เนื่องจากเฟรมเป็นการแสดงความรู้ที่มีการแสดงความรู้หลายได้หลายประเภท ทำให้การตรวจสอบความถูกต้องของความรู้เป็นไปได้ยาก นอกจากนั้นยังมีปัญหาในการกำหนดความสัมพันธ์ของความรู้ซึ่งเป็นข้อดีของเฟรม แต่การที่จะทำให้สมบูรณ์กระทำได้ยาก ปัญหาการสร้างแบบจำลอง การที่ไม่มีการกำหนดวิธีการอนุมานแบบตายตัว

ทำให้ระบบเฟรมมีความยืดหยุ่นในการใช้กับปัญหาประเภทต่าง ๆ แต่ในอีกแง่หนึ่งก็เป็นการเพิ่มภาระแก่ผู้ใช้ในการที่จะต้องตัดสินใจเลือกวิธีการอนุมาน

บทที่ 6

สถาปัตยกรรมของระบบ

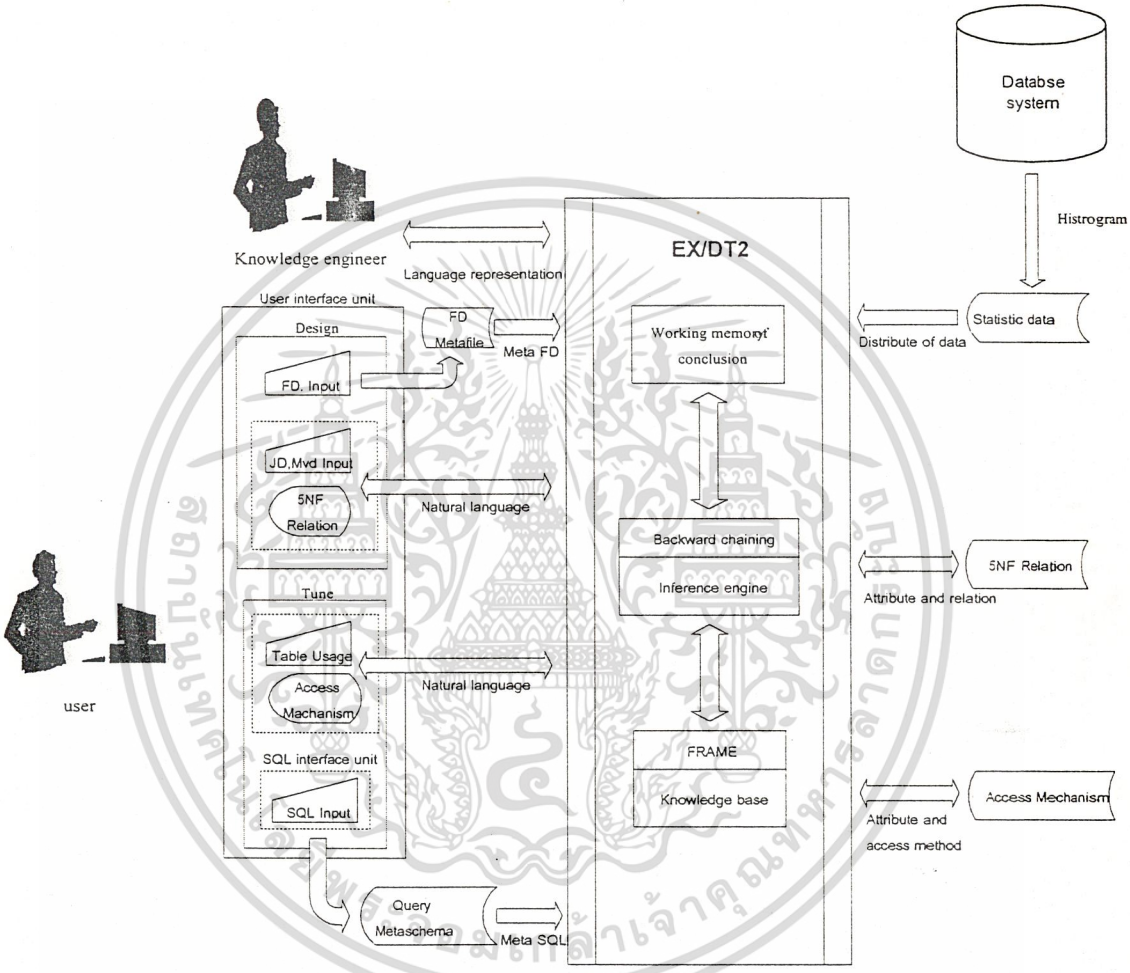
<u>บทนำ</u>

ในบทที่ผ่านมาเราได้ศึกษาถึงทฤษฎีต่าง ๆ ได้ทำการทดสอบเพื่อหาความรู้ และข้อสรุป ที่จำเป็นต่อการเรียนรู้ และสร้างงานวิจัยชิ้นนี้ขึ้นมา เราได้ทราบถึงประโยชน์ของการออกแบบฐานข้อมูลที่ดี และการออกแบบฐานข้อมูลโดยวิธี Normalization เราได้ทราบถึงหลักการในการเลือกรูปแบบ และกลไกในการเข้าถึงข้อมูลให้เหมาะสมกับความต้องการใช้งานฐานข้อมูล ปริมาณข้อมูลที่มีอยู่ และการทำงานของ Query Optimizer ของ RDBMS ที่ใช้อยู่ นอกจากนั้นเรายังได้ทราบรายละเอียดเกี่ยวกับระบบผู้เชี่ยวชาญในเรื่อง ส่วนประกอบของระบบ โครงสร้างของระบบ การทำงานของระบบ และทฤษฎีที่ใช้ในระบบ ในบทนี้ และบทต่อ ๆ ไปจะเป็นส่วนในการออกแบบ สร้าง และตรวจสอบงานวิจัยชิ้นนี้ ซึ่งระบบที่สร้างขึ้นมานี้มีชื่อว่า EX/DT2 โดยในบทนี้จะกล่าวถึงสถาปัตยกรรมโดยรวมของระบบ

การออกแบบสถาปัตยกรรมของระบบ นับว่าเป็นสิ่งสำคัญสิ่งหนึ่งเพราะจะทำให้เรามองเห็นภาพรวมของระบบ รู้ถึงขอบเขตของระบบ และยังสามารถแบ่งการทำงานของระบบออกเป็นสัดส่วนตามลำดับก่อนหลังได้อีกด้วย ถ้าระบบมีการออกแบบสถาปัตยกรรมที่ดีแล้วทำให้การพัฒนาระบบเป็นไปอย่างถูกต้อง และสำเร็จโดยเร็ว

สถาปัตยกรรมของระบบ EX/DT2

ภาพที่ 14



แสดงสถาปัตยกรรมของระบบผู้เชี่ยวชาญในการออกแบบ และปรับแต่งฐานข้อมูลรีเลชั่นแนล EX/DT2

จากภาพที่ 14 แสดงสถาปัตยกรรมของระบบผู้เชี่ยวชาญในการออกแบบ และปรับแต่งฐานข้อมูลรีเลชั่นแนล EX/DT2 ซึ่งแบ่งเป็นส่วนหลักได้ 3 ส่วนคือ

1. ส่วนติดต่อกับผู้ใช้
2. ส่วนติดต่อกับแฟ้มข้อมูล
3. ส่วนระบบหลัก

<u>ส่วนติดต่อกับผู้ใช้</u>

แบ่งชนิดของผู้ใช้เป็น 2 ระดับคือ

1. วิศวกรความรู้

จะมีหน้าที่เกี่ยวกับการจัดการความรู้ในฐานข้อมูลความรู้ โดยจะกระทำผ่านทางภาษาที่ใช้ในการจัดการฐานข้อมูลความรู้ ซึ่งในระบบนี้จะกระทำผ่านรูปแบบการแสดงความรู้ของเปลือกระบบผู้เชี่ยวชาญ ซึ่งในที่นี้ใช้โปรแกรมเปลือกระบบผู้เชี่ยวชาญ PC\PLUS โดยรายละเอียดการใช้งานมีกล่าวโดยสรุปไว้ใน ภาคผนวก ก

2. ผู้ใช้ทั่วไป

2.1 หน่วยรับข้อมูล FD Meta File

FD Meta File เป็นส่วนที่เก็บ Function dependency และความสัมพันธ์อื่น ๆ ของข้อมูลทั้งหมด ซึ่งเป็นข้อมูลเบื้องต้นที่จะนำมาออกแบบฐานข้อมูล สิ่งสำคัญในส่วนนี้คือผู้ใช้ต้องจะต้องกำหนดแอททริบิวท์ของข้อมูลทั้งหมดให้ได้ก่อน จากนั้นก็หาความสัมพันธ์ของแต่ละแอททริบิวท์และกำหนดให้อยู่ในรูปของ FD พร้อมชื่อของกลุ่ม FD แต่ละกลุ่ม เสร็จแล้วป้อนข้อมูลที่ได้ทั้งหมดเก็บลงบนแฟ้มข้อมูล META_FD โดยใช้ Text editor ทั่วไป หรือป้อนผ่าน editor ของระบบใช้ space bar เป็นตัวคั่นข้อมูลแต่ละตัว ชื่อแอททริบิวท์ที่เป็น Determinant และ Dependent จะอยู่ภายใต้วงเล็บ มีรูปแบบการป้อนข้อมูลดังนี้

"ชื่อกลุ่ม FD" "(ชื่อแอททริบิวท์ที่เป็น Determinant)" "(ชื่อแอททริบิวท์ที่เป็น Dependent)"

2.2 หน่วยรับข้อมูล Query Meta schema

Query Meta schema เป็นส่วนที่เก็บรายละเอียดเกี่ยวกับ SQL Query ที่มีใช้อยู่กับฐานข้อมูลที่ต้องการจะปรับแต่ง โดยจะกระทำผ่านทางระบบประมวลผลภาษา SQL ซึ่งจะทำการวิเคราะห์และจัดเก็บภาษา SQL ลงในตาราง META_SQL ซึ่งในระบบนี้จะมีหน้าจอสำหรับให้ผู้ใช้ใส่ Query และกำหนดความสำคัญมากน้อยของแต่ละ Query ด้วย โดยกำหนดให้อยู่ในรูปของเลขจำนวนเต็มตั้งแต่ 1 ถึง 10 ให้ค่า 10 มีความสำคัญสูงที่สุด จากนั้นเก็บรายละเอียดทั้งหมดลงบนแฟ้มข้อมูล META_SQL

2.3 หน่วยรับข้อมูล ความสัมพันธ์ JD และ MVD

ทำหน้าที่รับข้อมูลความสัมพันธ์ JD หรือ MVD จากผู้ใช้ระบบ เพื่อนำมาใช้ในการออกแบบฐานข้อมูล โดยในส่วนนี้จะทำงานร่วมกับ ส่วนแสดงผล JD หรือ MVD ทางจอภาพ ซึ่งส่วนแสดงผลนี้จะแสดงความสัมพันธ์ FD ที่อาจมี JD หรือ MVD แอบแฝงอยู่ จากนั้นจะติดต่อกับผู้ใช้เพื่อรับข้อมูลเพิ่มในส่วนของความสัมพันธ์ JD หรือ MVD

2.4 หน่วยรับข้อมูล ลักษณะการใช้งานตารางข้อมูล

จะเป็นหน่วยที่รับลักษณะการใช้งานตารางข้อมูลที่มีอยู่ในฐานข้อมูลที่นอกเหนือจากการใช้งานโดย Query เช่น ลักษณะการเปลี่ยนแปลงแก้ไขของข้อมูลในตารางข้อมูล การเจริญเติบโตของข้อมูลในตารางข้อมูล หรือลักษณะการใช้งานตารางข้อมูลร่วมกันเป็นต้น ซึ่งข้อมูลเหล่านี้จะนำมาเป็น fact เพื่อช่วยในการตัดสินใจเลือกชนิดของกลไกการเข้าถึงข้อมูล

2.5 หน่วยแสดงผล ข้อมูล JD และ MVD

เป็นส่วนที่แสดงสัมพันธ์ FD ที่อาจมี JD หรือ MVD แอบแฝง เพื่อให้ผู้ใช้เลือก และ ป้อนข้อมูล JD หรือ MVD โดยใช้หน่วยรับข้อมูล ความสัมพันธ์ JD และ MVD ดังที่กล่าวไว้ในตอนต้น

2.6 หน่วยแสดงผลความสัมพันธ์ที่อยู่ในรูป 5NF

เป็นส่วนที่แสดงผลลัพธ์ของความสัมพันธ์ทั้งหมดที่ได้จากการออกแบบฐานข้อมูล ซึ่งความสัมพันธ์เหล่านี้จะอยู่ในรูป FIFTH NORMAL FROM (5NF) โดยจะแสดง ชื่อของความสัมพันธ์ , แอททริบิวท์ของภายในแต่ละความสัมพันธ์ ตลอดจน primary key และ candidate key (ถ้ามี)

2.7 หน่วยแสดงผลรูปแบบ และกลไกในการเข้าถึงข้อมูล

เป็นส่วนที่แสดงผลลัพธ์ที่ได้จากการเลือกรูปแบบ และกลไกในการเข้าถึงข้อมูล โดยจะแสดง ชื่อของความสัมพันธ์ , แอททริบิวท์ที่เป็นตัวกำหนดกลไกในการเข้าถึงข้อมูล และชนิดของกลไกในการเข้าถึงข้อมูล พร้อมทั้งระบุถึงการเป็น primary index และ secondary index ของแอททริบิวท์เหล่านี้ด้วย

<u>ส่วนติดต่อแฟ้มข้อมูล</u>

1. ส่วนติดต่อแฟ้มข้อมูล FD Meta file

ดังที่กล่าวมาแล้วว่า FD Meta file เป็นส่วนที่เก็บ Function dependency และ dependency อื่น ๆ ของข้อมูลทั้งหมดที่จะนำมาออกแบบฐานข้อมูล ระบบหลักจะอ่านข้อมูลจากแฟ้มข้อมูลนี้

มาสร้างเป็น List เป็นความจริงเก็บไว้ใน working memory เพื่อนำไปสู่กระบวนการ Normalization ต่อไป

2. ส่วนติดต่อแฟ้มข้อมูล 5NF Relation

แฟ้มข้อมูล 5NF Relation เป็นแฟ้มข้อมูลที่เก็บผลลัพธ์ของความสัมพันธ์ทั้งหมดที่ได้จากการออกแบบฐานข้อมูล ความสัมพันธ์เหล่านี้อยู่ในรูปของ FIFTH NORMAL FROM กล่าวคือ นอกจากระบบหลักจะแสดงผลที่ได้จากการออกแบบฐานข้อมูลทางหน้าจอแล้ว ยังเก็บข้อมูลเหล่านั้นลงบนแฟ้มข้อมูลอีกด้วย จากนั้นระบบหลักก็จะอ่านข้อมูลนี้จากแฟ้มข้อมูล เพื่อจะนำไปใช้ในการกำหนดแอททริบิวท์ที่จะสร้างกลไกในการเข้าถึงข้อมูลต่อไป

3. ส่วนติดต่อแฟ้มข้อมูล Query Meta schema

ดังที่กล่าวมาแล้วในตอนต้นว่า Query Meta schema เป็นส่วนที่เก็บรายละเอียดเกี่ยวกับ SQL Query ที่มีใช้อยู่กับฐานข้อมูลที่ต้องการจะปรับแต่ง ระบบหลักจะใช้ข้อมูลจากส่วนนี้มาวิเคราะห์ถึงลักษณะการใช้งานฐานข้อมูล ร่วมกับการลักษณะการใช้งานตารางข้อมูลที่ให้ผู้ใช้ระบบป้อนเข้ามา เพื่อหาแนวทางในการกำหนดรูปแบบ และกลไกในการเข้าถึงข้อมูล โดยในขั้นแรกจะกระทำภายใน Query เดียวกันก่อน และทำการเก็บผลลัพท์ที่ได้ลง TEMP ไว้ จากนั้นก็จะทำการวิเคราะห์รวมทั้งหมด ซึ่งในจุดนี้ต้องคำนึงถึงผลกระทบระหว่าง Query ด้วย โดยจะใช้ความสำคัญของแต่ละ Query ที่มีเก็บอยู่ใน Query Meta schema มาช่วยในการพิจารณา

4. ส่วนติดต่อแฟ้มข้อมูล Statistic data

Statistic data เป็นแฟ้มข้อมูลสถิติที่ถูกสร้างขึ้นมาในลักษณะของ Histogram สำหรับเก็บรายละเอียดเกี่ยวกับการกระจายของข้อมูล โดยรวบรวมจากข้อมูลจริงที่มีอยู่ในระบบจัดการฐานข้อมูล สิ่งสำคัญที่ระบบหลักได้จากแฟ้มข้อมูลนี้คือ ระบบหลักต้องการรู้ว่าจำนวนของบรรทัดข้อมูลที่ตรงกับเงื่อนไขใน Query ที่ใช้ดึงข้อมูลว่ามีมากน้อยเท่าใด และปริมาณของข้อมูลในตารางของมูลทั้งหมด ซึ่งสิ่งเหล่านี้จะนำไปเป็น fact สำหรับช่วยในการพิจารณากำหนดรูปแบบ และกลไกในการเข้าถึงข้อมูล

<u>ระบบหลัก</u>

ระบบหลักประกอบด้วยส่วนสำคัญ 3 ส่วนคือ

- Working memory of conclusion

- Inference engine

- Knowledge base

1. Knowledge base จะเป็นตัวเก็บความรู้ทั้งหมดที่ใช้ในระบบ ซึ่งในระบบนี้ใช้ FRAME เป็นแบบแทนความรู้

2. Inference engine เป็นส่วนที่ค่อยควบคุมการใช้งานความรู้ที่มีอยู่ในฐานความรู้เพื่อให้ได้ประสิทธิภาพสูงสุดในการประมวลผล ซึ่งในระบบนี้ใช้วิธี Backward chaining เป็นหลัก

3. Working memory จะเป็นตัวเก็บข้อมูล และสถานะต่างๆ ที่ตัวระบบหลักดึงเข้ามาสู่ระบบในระหว่างการให้คำปรึกษาของระบบ โดยระบบจะใช้ข้อมูลที่มีอยู่ใน Working memory มาประมวลผลร่วมกับความรู้ที่มีอยู่ใน Knowledge base โดยมี inference engine เป็นตัวควบคุมการทำงานทั้งหมด

<u>การทำงานรวมของระบบ</u>

จากที่กล่าวมาในตอนต้นเป็นการอธิบายถึงรายละเอียดการทำงานของแต่ละส่วน เพื่อให้มองภาพรวมของระบบได้ชัดเจนยิ่งขึ้น ต่อจากนี้จะเป็นการอธิบายความสัมพันธ์ และลำดับขั้นตอนการทำงานของระบบรวมทั้งหมด โดยในส่วนนี้จะกล่าวถึงการทำงานในส่วนของการใช้งานระบบกับผู้ใช้ระบบทั่วไป ส่วนในกรณีการใช้งานระบบในลักษณะของวิศวกรความรู้จะพิจารณารายละเอียดได้จากการออกแบบ และพัฒนาระบบซึ่งมีกล่าวไว้ในบทต่อๆ ไป

การทำงานเริ่มจากผู้ใช้เตรียมข้อมูล FD ที่จะนำมาออกแบบฐานข้อมูลเก็บลงบน FD Meta file และเตรียมข้อมูลสถิติโดยการประมวลผลโปรแกรมสร้างสถิติบนระบบจัดการฐานข้อมูล INGRES เมื่อเตรียมข้อมูลทั้ง 2 พร้อมแล้ว ก็จะทำการประมวลผลตัวระบบผู้เชี่ยวชาญ ระบบจะดึง FD จากแฟ้มข้อมูลมาเก็บใน working memory แล้วทำการวิเคราะห์โดยใช้ความรู้ในการ Normalization ที่มีอยู่ในฐานข้อมูลความรู้ โดยมี inference engine เป็นตัวควบคุมการทำงาน จากนั้นจะมีการตรวจสอบความสัมพันธ์ JD และ MVD ถ้ามีความเป็นไปได้ที่จะมีความสัมพันธ์ดังกล่าวแอบแฝงอยู่ จะติดต่อให้ผู้ใช้ระบบเลือก และใส่ความสัมพันธ์ดังกล่าว เมื่อมีความสัมพันธ์ JD หรือ MVD เข้ามาสู่ working memory แล้ว ก็จะประมวลผลต่อจนได้ผลลัพท์ออกมาเป็นความสัมพันธ์ที่อยู่ในรูป 5NF และแสดงผลทางหน้าจอพร้อมกับบันทึกผลที่ได้ลงบนแฟ้มข้อมูล 5NF relation จากนั้นระบบจะถามผู้ใช้ว่า จะทำการประมวลผลในส่วนของการกำหนดรูปแบบ และกลไกในการเข้า

ถึงข้อมูลต่อหรือไม่ ถ้าทำต่อระบบก็จะให้ผู้ใช้ป้อน SQL Query ที่ใช้งานประจำอยู่กับฐานข้อมูลนี้แล้วเก็บข้อมูลที่ได้ลงบนแฟ้มข้อมูล Query meta schema จากนั้นระบบจะดึงข้อมูลจาก 5NF_relation , Query meta schema และจาก statistic มาทำการ inference ร่วมกับความรู้ในการกำหนดรูปแบบ และกลไกในการเข้าถึงข้อมูลที่มีอยู่ในฐานข้อมูลความรู้ ซึ่งในระหว่างการ inference จะมีการถามให้ผู้ใช้ระบบตอบเกี่ยวกับลักษณะการใช้งานตารางข้อมูลที่นอกเหนือจาก Query ที่ป้อนเข้ามา จากนั้นจะประมวลผลต่อจนเสร็จได้ผลลัพท์ออกมาเป็นการกำหนดรูปแบบและกลไกในการเข้าถึงข้อมูลที่มีอยู่ในฐานข้อมูล โดยจะแสดงผลทางหน้าจอพร้อมกับบันทึกผลที่ได้ลงบนแฟ้มข้อมูล

บทที่ 7

การออกแบบฐานข้อมูลความรู้

<u>บทนำ</u>

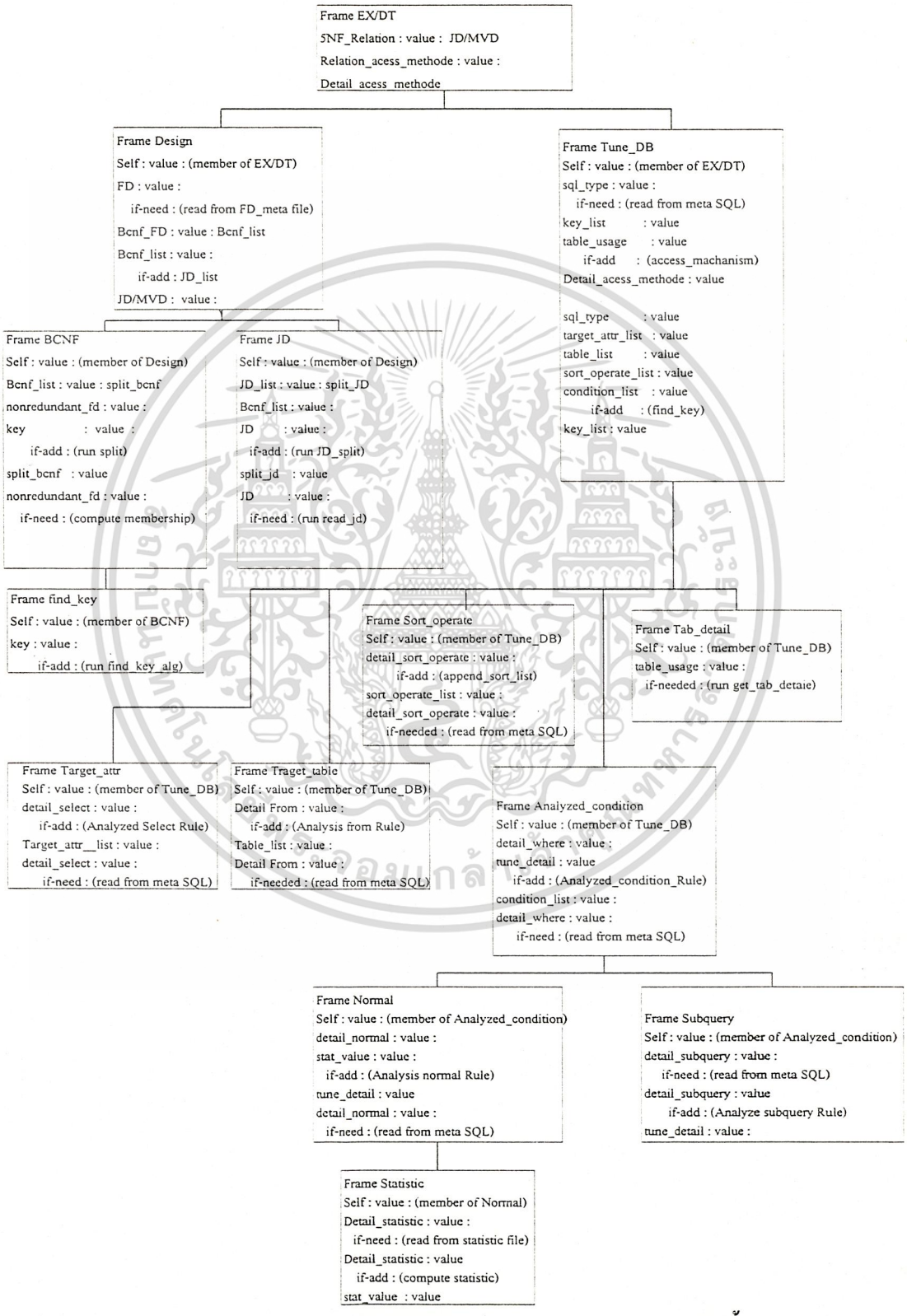
ในบทนี้จะกล่าวถึงการออกแบบฐานข้อมูลความรู้ โดยจะนำความรู้เรื่องการออกแบบฐานข้อมูลที่กล่าวในบทที่ 2 และความรู้เรื่องการเลือกรูปแบบ และกลไกในการเข้าถึงข้อมูลที่กล่าวในบทที่ 3 และ 4 มาออกแบบการจัดเก็บลงบนฐานข้อมูลความรู้ ซึ่งในงานวิจัยนี้ได้เลือกใช้เฟรม (FRAME) เป็นแบบแทนความรู้ และจากนั้นจะกล่าวถึงทฤษฎีที่จะนำมาช่วยลดความซ้ำซ้อนของความจริง (fact) และ กฎ (rule) ที่จะนำมาใส่ในฐานข้อมูลความรู้ตามเฟรมที่ได้ออกแบบไว้ โดยใช้วิธีการของ JOHN K. DEBENHAM ที่กล่าวไว้ในเรื่อง NORMAL FORM FOR KNOWLEDGE

<u>**การออกแบบระบบเฟรม (Frame base system)**</u>

ระบบเฟรมประกอบด้วย กลุ่มของเฟรม และกลไกการวินิจฉัย เฟรมความรู้ที่ใช้ในระบบนี้เป็นเฟรมของเปลือกระบบผู้เชี่ยวชาญ PC/PLUS ซึ่งมีลักษณะผสมระหว่าง Rule base กับ Frame โดยแต่ละเฟรมจะเชื่อมกันในลักษณะของ tree

จากความรู้ที่ได้จากบทที่ 2 บทที่ 3 และ บทที่ 4 สามารถนำมาสร้างระบบเฟรมได้ดังนี้

ภาพที่ 15



แสดงรายละเอียด และความสัมพันธ์ของเฟรมในลักษณะลำดับชั้น

จากภาพที่ 15 สามารถแบ่งการทำงานของเฟรมได้เป็น 2 กลุ่ม คือ

<u>กลุ่มแรก</u>

เป็นเฟรมที่ใช้ในกลไกการพิจารณาออกแบบฐานข้อมูล ประกอบด้วยเฟรมสี่เฟรมคือ Frame Design , Frame BCNF , Frame JD และ Frame find_key รายละเอียดของการทำงานภายใน เฟรมนี้แบ่งได้เป็น 3 ส่วนคือ

1. ส่วนประมวลผล BCNF Rule

ในขั้นแรกระบบจะดึงข้อมูลจาก FD Meta file เข้ามาสร้างเป็น List ซึ่งจะเป็น Fact ในระบบ หลังจากนั้นจะใช้ขบวนการ List processing ร่วมกับขบวนการ Inference ความรู้ที่มีอยู่ใน Frame BCNF ซึ่งรายละเอียดของการทำงาน จะเป็นการกำจัดความซ้ำซ้อนของความสัมพันธ์ระหว่างแอททริบิวต์ที่เกิดขึ้นทั้งหมด ผลที่ได้คือได้ FD ที่ไม่มีความซ้ำซ้อนของความสัมพันธ์ระหว่างข้อมูล หลังจากนั้นจะหาคีย์ของแต่ละความสัมพันธ์โดยใช้ความรู้ใน frame find_key เมื่อได้คีย์ของแต่ละความสัมพันธ์แล้วก็จะนำมาตรวจสอบความเป็น BCNF โดยใช้ความรู้ในส่วนของframe BCNF

2. ส่วนตรวจสอบความสัมพันธ์เจดีและเอ็มวีดี (JD and MVD Interface Unit)

ทำหน้าที่ตรวจสอบ FD ที่ได้จากส่วนประมวลผล BCNF Rule ว่าตารางใดมีความเป็นไปได้ที่มีความสัมพันธ์เจดีหรือเอ็มวีดี แอบแฝงอยู่ จากนั้นก็จะติดต่อกับผู้ใช้เพื่อรับข้อมูลเพิ่มในส่วนของความสัมพันธ์เจดีหรือเอ็มวีดี และจะเพิ่มเป็น fact ใหม่เข้าไปในระบบเพื่อส่งต่อขบวนการต่อไป

3. ส่วนประมวลผล 4NF และ 5NF Rule

ในขั้นแรกระบบระบบจะรับ fact ที่ส่งมาจากหน่วยติดต่อผู้ใช้ด้านความสัมพันธ์เจดี และเอ็มวีดี จากนั้นทำการ Inference ร่วมกับความรู้ ที่มีอยู่ในFrame JD รายละเอียดของการทำงานคือในส่วนแรกจะเป็นการตรวจสอบความถูกต้องของข้อมูล เจดี หรือ เอ็มวีดีที่เข้ามา หลังจากนั้นจะทำ

การแตกตารางตามกฎของ 4NF และ 5NF สุดท้ายก็จะได้ความสัมพันธ์ของข้อมูลที่อยู่ในรูป 5NF และแสดงความสัมพันธ์ที่ได้ทางหน้าจอ พร้อมกับบันทึกข้อมูลความสัมพันธ์นี้ลงบนแฟ้มข้อมูล

<u>กลุ่มที่สอง</u>

เป็นเฟรมที่ใช้ในการวิเคราะห์การเลือกรูปแบบและกลไกในการเข้าถึงข้อมูล ประกอบด้วย 9 เฟรมด้วยกัน คือ 9 เฟรมที่เหลือ ยกเว้นเฟรม EX/DT ซึ่งมีรายละเอียดของการทำงานของกลุ่มเฟรมดังต่อไปนี้

1. ส่วนประมวลผล Analyze Query Rule

ในขั้นแรกระบบจะดึงข้อมูลมาจากแฟ้มข้อมูลที่ได้จากขั้นตอนการออกแบบฐานข้อมูล และข้อมูล Query Meta schema ซึ่งเก็บรายละเอียดเกี่ยวกับ Query ที่ใช้งานอยู่ ประจำมาเป็น fact ในระบบเพื่อใช้ Inference ร่วมกับความรู้ที่มีอยู่ในระบบ รายละเอียดของความรู้จะเกี่ยวกับการวิเคราะห์การลักษณะทำงานของ Query Sql ที่เข้ามาในระบบ โดยในขั้นแรกจะพิจารณาจากชนิดของคำสั่ง SQL ที่เข้ามา จากชนิดของคำสั่งจะนำไปสู่การพิจารณาดังต่อไปนี้

- พิจารณาแอททริบิวท์เป้าหมาย
- พิจารณาตารางข้อมูลที่ใช้
- พิจารณาการใช้งานข้อมูลแบบเลียงลำดับ
- พิจารณาลักษณะของเงื่อนไขในการดึงข้อมูล
- พิจารณาลักษณะการใช้งานตารางร่วมกันของข้อมูล

ถ้าลักษณะการใช้งานตารางข้อมูลมีเงื่อนไขในการเรียกค้นข้อมูล ก็จะดึงข้อมูลจากตารางสถิติมาประกอบ จากนั้นหน่วยรับลักษณะการใช้งานตารางข้อมูลซึ่งอยู่ใน Frame Tab_detail จะรับลักษณะการใช้งานตารางข้อมูลจากผู้ใช้ระบบ เช่น ลักษณะการ เปลี่ยนแปลงแก้ไขของข้อมูลในตารางข้อมูล การเจริญเติบโตของข้อมูลในตารางข้อมูล เป็นต้น และนำข้อมูลเหล่านี้จะนำมาเป็น fact ร่วมกับ fact ที่ได้จากขบวนการก่อนหน้านี้ส่งต่อไปยังขบวนการต่อไป

2. ส่วนประมวลผลการกำหนดกลไกการเข้าถึงข้อมูล (Access Mechanism Rule)

จะเป็นส่วนการทำงานของกลไกภายในระบบผู้เชี่ยวชาญ จะนำ fact ที่ได้จาก 2 ขบวนการ คือ fact ที่เกี่ยวกับการวิเคราะห์ลักษณะการใช้งาน Query และ fact ที่เกี่ยวกับการใช้งานตารางข้อมูลที่นอกเหนือจาก Query มา Inference ร่วมกับความรู้ในการกำหนดกลไกการเข้าถึงข้อมูล โดยในขั้นแรกจะกำหนดให้ได้ก่อนว่าควรสร้างกลไกการเข้าถึงข้อมูลบนคอลัมน์ใด หลังจากนั้นก็จะพิจารณาต่อว่าควรใช้วิธีใดคือ Hash Isam หรือ Btree ซึ่งความรู้เหล่านี้ได้จากการ สรุปกฎเกณฑ์ในการกำหนดรูปแบบ และกลไกการเข้าถึงข้อมูลในบทที่ 4

สำหรับเฟรม EX/DT เป็นรูทเฟรมที่เชื่อม fact ระหว่างสองกลุ่ม และแสดงผลลัพธ์ที่ได้จากระบบทั้งหมด

<u>กลไกการอนุมานที่ใช้ในการทำงานของเฟรมในระบบ</u>

จะเห็นว่าระบบเฟรมที่สร้างขึ้นมีลักษณะเชื่อมต่อกันเป็น tree เฟรมที่อยู่บนสุดเรียกรูทเฟรม (root frame) เฟรมที่ต่ำชั้นลงมาเรียกซับเฟรม (subframe) และในแต่ละคู่ของเฟรมที่เชื่อมต่อกัน เฟรมที่อยู่ลำดับชั้นสูงกว่าเรียกเฟรมพ่อ (parent frame) เฟรมที่อยู่ลำดับชั้นต่ำกว่า เรียกเฟรมลูก (child frame)

กลไกในการวินิจฉัย ใช้วิธี Backward chaining โดยเราต้องกำหนดเป้าหมาย (goal) ที่เราต้องการก่อน ในเฟรมทุกเฟรมจะมีเป้าหมายการทำงานเป็นของตัวเองซึ่งในบางเฟรมอาจมีหลายเป้าหมาย เป้าหมายหลักจะถูกกำหนดไว้ที่รูทเฟรม กลไกการวินิจฉัยจะเริ่มทำงานจากเป้าหมายหลักที่กำหนดไว้ในรูทเฟรมก่อน การแก้ปัญหาของเป้าหมายหลักนั้นจะมีองค์ประกอบด้วยการแก้ปัญหาของเป้าหมายย่อยในซับเฟรม จึงต้องเชื่อมซับเฟรมเข้ามาเพื่อหาคำตอบของเป้าหมายย่อยของแต่ละซับเฟรมก่อน เมื่อได้คำตอบของเป้าหมายย่อยเสร็จครบทุกเป้าหมายแล้ว ก็จะสามารแก้ปัญหาเป้าหมายหลักได้ก็เป็นอันจบการทำงานของระบบ

<u>การเชื่อมต่อระหว่างเฟรมพ่อ กับเฟรมลูก (Instantiation) แบ่งได้เป็นดังนี้</u>

1. Consider frame

ในกรณีนี้เราสามารถสั่งให้ทำการเชื่อมระหว่างเฟรมพ่อกับเฟรมลูกได้ตามความต้องการ

2. Backward chaining แบ่งได้เป็น 2 กรณี

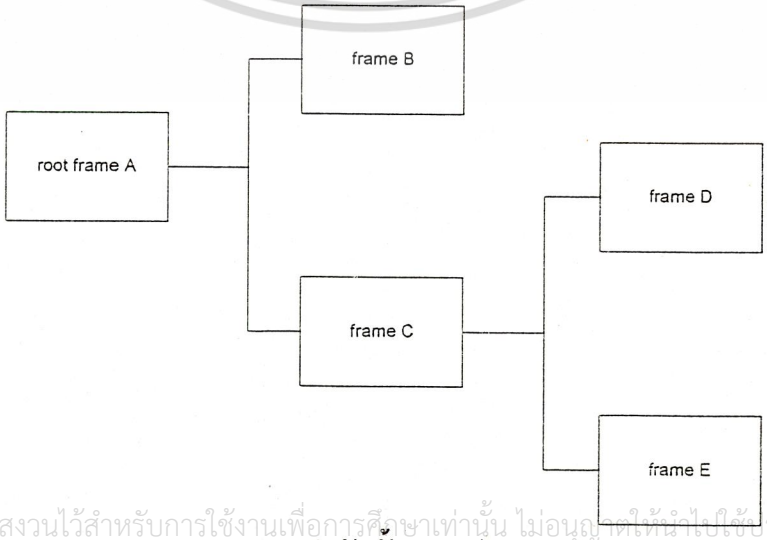
ใช้ตัวแปรเป้าหมาย (goal parameter) กลไกในการวินิจฉัยจะพยายามหาค่าเป้าหมายที่เราต้องการ ถ้ามีการกำหนดค่าให้กับตัวแปรเป้าหมายในส่วนของเฟรมลูกก็จะถูกเชื่อมต่อเข้ามา

เช็คตัวแปร (parameter) เพื่อให้ได้ค่าตัวแปรเป้าหมายที่ต้องการในบางครั้งจะต้องมีการกำหนดค่าให้กับตัวแปรอื่นก่อน ซึ่งถ้ามีการกำหนดค่าให้กับตัวแปรตัวนั้นในเฟรมลูกก็จะทำการเชื่อมเข้ามา

<u>การสืบทอดคุณสมบัติ (Inheritance)</u>

เป็นหลักการที่สำคัญของเฟรม คือจะมองเฟรมในลักษณะของเฟรมลูก และเฟรมพ่อ โดยที่เฟรมลูกสามารถอ้างถึงตัวแปรที่อยู่ในเฟรมพ่อได้ และเฟรมพ่อก็สามารถอ้างถึงกฎที่อยู่ในเฟรมลูกได้ดังภาพที่ 16 และตารางที่ 16

ภาพที่ 16



แสดงลำดับชั้นของเฟรม

จากภาพที่ 16 นำมาแสดงความสามารถในการอ้างถึงตัวแปรและกฎได้ดังตารางที่ 16

ตารางที่ 16

แสดงคุณสมบัติการสืบทอดของเฟรม

เฟรม	สามารถอ้างถึงตัวแปรจากเฟรม	สามารถอ้างถึงกฎจากเฟรม
A	A	A B C D E
B	A B	B
C	A C	C D E
D	A C D	D
E	A C E	E

<u>การลดความซ้ำซ้อนของ Fact กับ Rule ในฐานข้อมูลความรู้</u>

ภายในฐานข้อมูลความรู้จะประกอบด้วยเฟรมต่าง ๆ และภายในเฟรมจะประกอบด้วย Fact กับ Rule ซึ่งมีจำนวนมาก ดังนั้นการที่จะเขียน Fact กับ Rule ลงบนฐานความรู้ควรคำนึงถึงความซ้ำซ้อนของสิ่งเหล่านี้ด้วย ฐานข้อมูลความรู้ที่ไม่มีความซ้ำซ้อนของ Fact กับ Rule จะมีขนาดเล็กกว่า ทำให้ประหยัดเนื้อที่ในการจัดเก็บ การ Inference ทำได้เร็วขึ้น และง่ายต่อการเปลี่ยนแปลงแก้ไขด้วย

<u>NORMAL FORMS สำหรับฐานความรู้</u>[4]

ในที่นี้จะมองฐานความรู้ในรูปของการแสดงความสัมพันธ์ระหว่างความจริงหรือความรู้ต่าง ๆ ในฐานความรู้ ซึ่งจะใช้ Horn clause เป็นตัวแสดง Normal form สำหรับฐานความรู้ โดยจะกำหนดความรู้ให้อยู่ในรูปเซ็ทของ clause หรือ groups

Function ที่จะแสดงความรู้ในรูป Horn clause แบ่งได้เป็น 2 ลักษณะ (รายละเอียดของ Horn clause มีกล่าวไว้ในบทที่ 5 ในหัวข้อ การแสดงความรู้โดยใช้วิธีตรรกศาสตร์)

1. function ที่แสดงความสัมพันธ์ระหว่าง head ของ clause (group) กับ body ของ clause(group)

2. function ที่กำหนดโดย resolution คือมีการกำหนดค่า clauses(group) มาจาก function อื่น

ข้อแตกต่างระหว่าง information function กับ knowledge function คือ information function จะแสดงความสัมพันธ์ในรูปของ tupples และ relation ซึ่งประกอบไปด้วยความสัมพันธ์ระหว่าง key กับ nonkey ส่วน knowledge function จะแสดงความสัมพันธ์ในรูปของ clauses และ groups ซึ่งประกอบไปด้วยความสัมพันธ์ระหว่าง head กับ body ซึ่ง body เทียบได้กับ key ของ relation ถึงแม้ว่า relation จะประกอบไปด้วย tupples และ group ประกอบด้วย clauses แต่ clauses แตกต่างจาก tupples เพราะว่าโครงสร้างแตกต่างกันคือ tupples ประกอบด้วย ค่าคงที่และแสดงค่าความจริงเพียง 1 ค่า ส่วน clause ประกอบด้วยตัวแปรที่แสดงส่วนใดส่วนหนึ่งหรือทั้งหมดของ rule ซึ่งจากจุดนี้อาจกล่าวได้ว่า tupple เป็นตัวอย่างค่าคงที่ของ clause ที่ทุกตัวแปรถูกแทนด้วยค่าคงที่จาก tupple

Normal form สำหรับ clauses

เป็น normal form สำหรับ clause ต่าง ๆ ใน group และ normal form สำหรับตัวของ group เอง โดยในส่วนแรกจะพิจารณาจาก ข้อมูลต่าง ๆ ภายใน clause และส่วนหลังจะพิจารณาจาก body of clause to head of clause ซึ่งมีกฎเกณฑ์ดังต่อไปนี้

K.1 ไม่ควรมี label ที่เป็นค่าคงที่ปรากฏอยู่ใน clause ควรแทน label นั้นด้วยตัวแปรอื่น

ตัวอย่าง

item/sell-price(x,y) <-- item/by-price(x,z), y = z * 1.2

จาก clause นี้จะขัดแย้งกับ กฎ K.1 ซึ่งจะทำให้เกิดปัญหาได้คือถ้าค่า 1.2 เปลี่ยนแปลงจะต้องตามไปแก้ทุก clause ที่เกี่ยวข้องกับค่า 1.2 ดังนั้นเราจึงควรแตก fact นี้ออกมาเป็น 2 fact โดยกำหนดให้ mark-up factor = 1.2 และให้ sell-price = by-price * mark-up actor ดังนี้ จะเห็นว่า fact แรกเป็นประชากรเรียกว่า mark-up-factor ที่มีค่า label เพียง 1 ค่าคือ 1.2 และ fact ที่ 2 แสดงได้ด้วย clause ดังนี้

item/sell-price(x,y) <-- item/by-price(x,z),

is-the [mark-up-factor](w), y = z * w

แต่มีบางกรณีที่มีค่าคงที่ใน clause แต่ไม่ขัดแย้งกับกฎ K.1 เช่น กำหนดให้ mark-up rate = 20 % และ selling price = bying price + mark-up rate percent ดังนี้

item/sell-price(x,y) <-- item/buy-price(x,z),

is-the [mark-up-rate:%](w),y = z * (1+ w/100)

จะเห็นว่า clause นี้มีค่า label 100 ปรากฏอยู่ แต่ใน กรณีนี้ไม่ขัดกับ K.1 เพราะว่าเมื่อ mark-up-rate เปลี่ยนไป ค่า label 100 จะไม่เปลี่ยนตาม

K.2 คล้ายกับ second normal form คือไม่ควรมี predicate ที่ซ้ำซ้อนใน body ของ clause ความหมายของ predicate ที่ซ้ำซ้อนใน body ของ clause นั้น คือถ้ามี predicate ใดถูกลบไปแล้ว ค่าความถูกต้องของ clause นั้นไม่เปลี่ยนแปลง แสดงว่า predicate นั้นซ้ำซ้อน ซึ่ง predicate ที่ซ้ำซ้อนนั้นพิจารณาได้ดังนี้คือ

- พิจารณาจาก Duplicate predicate คือมีการแสดง predicate ซ้ำกันภายใน body ของ clause

ตัวอย่าง

A(x,y) <-- B(x,z),B(x,z),C(z,x)

predicate B เป็น duplicate คือมีการแสดงความสัมพันธ์ซ้ำกันใน clause

- พิจารณาจาก Irrelevant predicate คือเป็น predicate ที่ไม่เกี่ยวข้องกับ predicate ใด ๆ

ตัวอย่าง

A(x,y) <-- B(y,z),C(z,x),D(u,v)

predicate D เป็น irrelevant predicate จะเห็นว่า u,v ไม่มีความสันพันธ์กับใคร ดังนั้น D(u,v) จะเป็นจริงเสมอ

- พิจารณาจากค่าที่สัมพันธ์กันของ Information subset แต่ละ clause ที่เกี่ยวข้องกัน

A(x,y) <-- B(x,y)----------[1]

C(x,y) <-- A(x,y),B(x,y),D(x,y)----------[2]

จาก clause ที่ 1 เรากล่าวได้ว่า predicate B เป็น information ของ predicate A และสามารถลบ predicate A(x,y) ออกจาก clause ที่ 2 ได้

K.3. คล้าย third normal form คือความรู้ใด ๆ ควรอยู่ใน clause ใด clause หนึ่งเพียง clause เดียว ไม่ควรปรากฏอยู่ใน clause อื่น

ตัวอย่าง

A(x,y) <-- B(y,z),C(z,u),D(u,x)

E(z,x) <-- C(z,u),D(u,x)

จาก clause แรกสามารถเปลี่ยนเป็น

A(x,y) <-- B(y,z),E(z,x)

K.4. คล้าย fourth normal form คือ ซึ่งจะเกี่ยวกับส่วนประกอบของ body ของ clause clause ที่ไม่ขัดต่อกฎข้อ K.4 ก็ต่อเมื่อ ไม่สามารถแบ่ง clause นั้นได้เป็น 2 clause และใน 2 clause ย่อยประกอบด้วยจำนวนของ body predicate น้อยกว่า clause เดิม

ตัวอย่าง

A(x,y) <-- B(y,z),C(z,w),D(w,y)

จาก clause เบื้องต้นสามารถแบ่งเป็น 2 clause ย่อยดังนี้

A(x,y) <-- E(y,w),D(w,y)

E(y,w) <-- B(y,z),C(z,w)

แต่ก็มีบางกรณีที่ไม่สามารถแบ่ง clause ได้ตามกฎข้างต้นตัวอย่างเช่น

A(u,v) <-- B(u,v),C(u,w),D(v,w),E(u,x),F(v,x),G(w,x)

จาก clause ข้างต้นจะเห็นว่าความสัมพันธ์ของตัวแปรในแต่ละ predicate ของ body อยู่ในรูปของ complex cross-linking ซึ่งจะเป็นตัวที่ทำให้ไม่สามารถแบ่งเป็น clause ย่อยได้

จากกฎเบื้องต้นทำให้ลดความซ้ำซ้อนของ predicate และถ้ามีการอ้างถึง predicate ย่อยหลายที่ถ้า predicate ย่อยมีการแก้ไขก็จะแก้ไขเพียงที่เดียว

Normal form สำหรับ groups

K.5. ไม่ควรมี clause ที่ซ้ำซ้อนใน group สำหรับ clause ที่ซ้ำซ้อนคือ เมื่อลบ clause ออกจาก group แล้วไม่มีความแตกต่างเกิดขึ้นในการ derive กลุ่มของ clauseใน group โดย resolution

ตัวอย่าง

A(x,y) <-- B(y,x)

A(x,y) <-- B(y,x),C(x,y)

จากตัวอย่างข้างต้นจะเห็นว่า clause ที่ 2 มีความซ้ำซ้อน ซึ่งการลบ clause ที่มีความซ้ำซ้อนออกจาก group มีความสำคัญมาก เพราะถ้า group ประกอบด้วย clause ที่มีความซ้ำซ้อน และมีการแก้ไขซึ่งกระทำบน clause ที่มีความซ้ำซ้อนเพียงอย่างเดียว จะทำให้เกิดความผิดพลาดได้

K.6. ทุกๆ clause ที่ อ้างอิง (derive) จาก clause ใน group โดยใช้ resolution ควรเป็น unique derivation

ตัวอย่าง

item/cost('pencil',0.4:$)<--

item/cost('book',1.2:$)<--

item/cost('pen',0.7:$)<--

item/sell(x,y:$)<-- item/cost:$(x,z:$),Y:$ =z:$ * 1.2

item/sell('book',1.44:$)<--

จากตัวอย่างจะเห็นว่า clause ที่ 2,4 และ 5 ผิดกฎข้อ K.6 เนื่องจากสามารถอ้างอิงค่าได้มากกว่า 1 clause

K.7. ไม่ควรแทน clause ตั้งแต่ 2 clause ขึ้นไปใน group กับ clause ที่น้อยกว่า

ตัวอย่าง Quick sort program

Sort(x1.x2,y)<--Part(x1,x2,u1,u2),Sort(u1,v1),Sort(u2,v2),Cat(v1,x1.v2,y)

Sort(nil,nil)<--

Part(x1,z.x2,z.u1,u2)<-- <=(z,x1),Part(x1,x2,u1,u2)

Part(x1,z.x2,u1,z.u2)<-- >(z,x1),Part(x1,x2,u1,u2)

Part(x,nil,nil,nil)<--

Cat(u.x,y,u.z)<-- Cat(x,y,z)

Cat(nil,y,y)

จาก Sort group ข้างต้นสามารถแทนได้ด้วย 3 clause ต่อไปนี้

Sort(nil,nil)<--

Sort(x.nil,x.nil)<--

Sort(x1.x2.x3,y)<--Part(x1,x2.x3,u1),Sort(u1,v1),Sort(u2,v2),Cat(v1,x1.v2,y)

จะเห็นว่าใน Sort group ของตัวอย่างที่ 2 ผิดกฎเพราะสามารถแทน clause 2,3 ในตัวอย่างที่ 2 ได้ด้วย clause แรกในตัวอย่างแรก

K.8. ในกรณีที่สามารถแบ่ง clause ใน clause group ออกเป็นตั้งแต่ 2 subset clause ขึ้นไป และเมื่อใดก็ตามที่มีการอ้างถึงเพียง subset ของ clause แล้วสามารถ กำหนดเป้าหมายที่ต้องการได้ แสดงว่า clause นั้นขัดกับกฎข้อ 4 ในกรณีนี้ต้องสร้าง head predicate ขึ้นมาใหม่เพื่อรองรับ subset clause ดังกล่าว

ตัวอย่าง

A(x,y) <-- >=(x,1), B(x,z), C(z,y)

A(1,3) <--

A(x,y) <-- <=(x,0), D(x,z), C(z,y)

A(0,4) <--

จากตัวอย่างจะเห็นว่าขัดต่อกฎข้อ K.8 ดังนั้นจึงสร้าง head predicate ขึ้นมาใหม่เพื่อรองรับ subset clause เหล่านั้นดังนี้

E(x,y) <-- B(x,z),C(z,y)

E(1,3) <--

F(x,y) <-- D(x,z),C(z,y)

F(0,4) <--

จาก clause ข้างต้นสามารถแทนได้ดังนี้

A(x,y) <-- >=(x,1),E(x,y)

A(x,y) <-- <=(x,0),F(x,y)

บทที่ 8

การพัฒนาระบบฐานข้อมูล

<u>บทนำ</u>

ในบทนี้จะกล่าวถึงการพัฒนาระบบโดยรวมทั้งหมด ซึ่งประกอบด้วยระบบใหญ่สองระบบคือ ระบบออกแบบฐานข้อมูล และระบบวิเคราะห์รูปแบบ และกลไกในการเข้าถึงข้อมูล ซึ่งภายในระบบทั้งสองประกอบด้วย ส่วนติดต่อกับผู้ใช้ ส่วนฐานข้อมูลความรู้

<u>อุปกรณ์ที่ใช้ในการพัฒนาระบบ</u>

- เครื่องคอมพิวเตอร์

เครื่องคอมพิวเตอร์ที่ใช้พัฒนาระบบนี้เป็นไมโครคอมพิวเตอร์ IBM PC คอมแพทิเบิลโดยมีหน่วยประมวลผลกลาง (CPU : Central Processing Unit) ขนาด 16 บิต 80386 DX-40 หน่วยความจำ 4 เมกะไบต์ ฮาร์ดดิสก์ขนาด 120 เมกะไบต์ ฟลอปปี้ดิสก์ขนาด 1.44 และ 1.2 เมกะไบต์ อย่างละ 1 ตัว จอภาพแบบ Super VGA (1024 X 768) 256 สี แป้นพิมพ์ 101คีย์

Server computer Hp Serie 9000 Model 720 หน่วยความจำ 128 เมกะไบต์ ฮาร์ดดิสก์ขนาด 5 กิกะไบต์

- ซอฟต์แวร์ที่ใช้พัฒนาระบบ

โปรแกรมระบบปฏิบัติการ MS-DOS 6.2

โปรแกรมเปลือกระบบผู้เชี่ยวชาญ PCPLUS ใช้สำหรับพัฒนาตัวระบบผู้เชี่ยวชาญ

โปรแกรมระบบจัดการฐานข้อมูล INGRES ใช้สำหรับการวิเคราะห์หาข้อสรุป และกฎเกณฑ์ในการกำหนดรูปแบบและกลไกในการเข้าถึงข้อมูล

คอมไพล์เลอร์ภาษา C ใช้สำหรับสร้างหน่วยติดต่อผู้ใช้ในด้านการรับ Meta SQL

โปรแกรม TEXT EDITOR ทั่วไป สำหรับป้อน Meta Function Dependency(Meta FD)

INGRES/4GL ใช้สำหรับสร้างตารางเก็บรายละเอียดการกระจายของข้อมูลในฐานข้อมูล

<u>การพัฒนาระบบในส่วนการออกแบบฐานข้อมูล</u>

1. การพัฒนาในส่วนรับข้อมูล Meta FD

Meta FD เป็นข้อมูลเป็นข้อมูลเบื้องต้นที่เก็บรายละเอียดเกี่ยวกับความสัมพันธ์ของแอททริบิวท์ที่จะนำมาออกแบบฐานข้อมูล โดยผู้ใช้ระบบป้อนจาก TEXT EDITOR ซึ่งมีความสัมพันธ์ของข้อมูลดังนี้

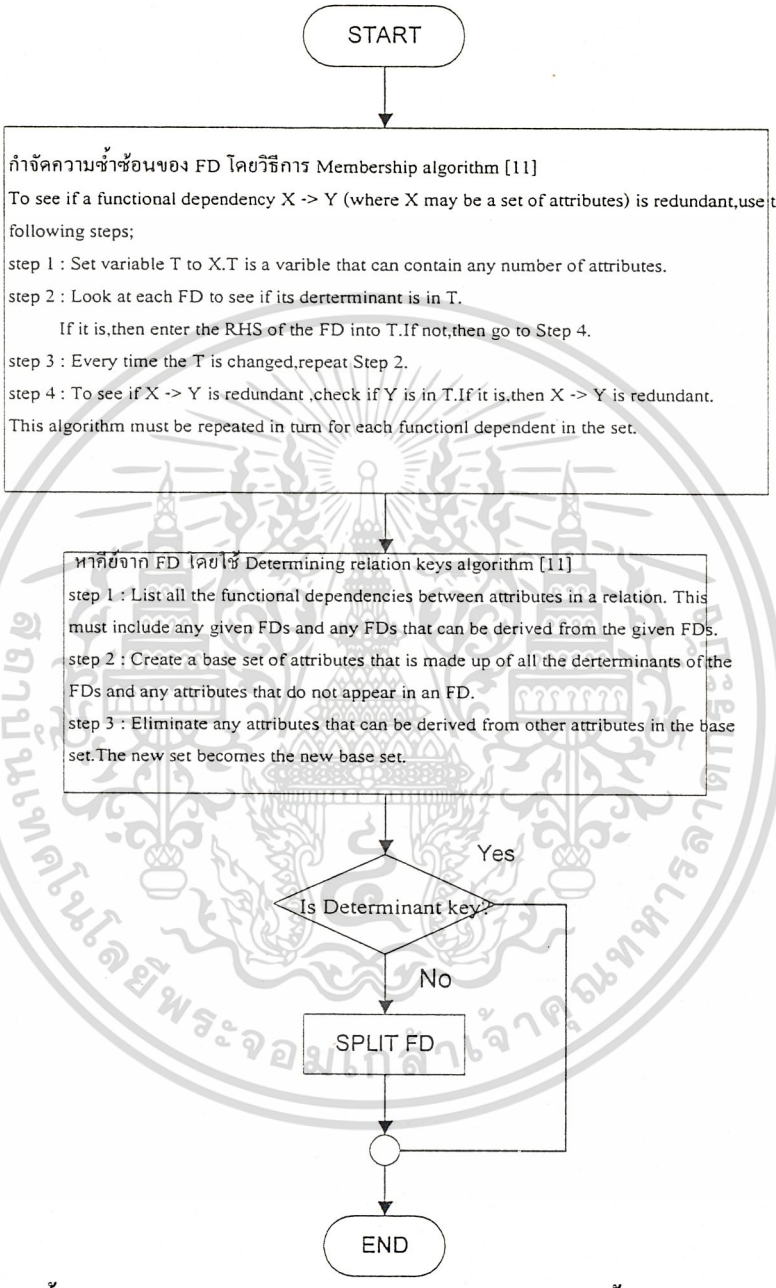
| TABLE_NAME | DETERMINANT ATTRIBUTE | DEPENDENT ATTRIBUTE |

ระบบจะรับข้อมูล Meta FD ซึ่งเป็น TEXT FILE มาสร้างเป็น list ซึ่งมีโครงสร้างดังนี้

((table_name1,determinant1,dependent1), (table_name1,determinant1,dependent2),...)ในส่วนนี้พัฒนาโดยภาษา LISP

2. การพัฒนาฐานข้อมูลความรู้ในส่วนวิเคราะห์ BCNF RULE มีขั้นตอนดังต่อไปนี้

ภาพที่ 17



แสดงขั้นตอนการทำงานในส่วนสร้างตารางฐานข้อมูลในขั้น Boyce/Codd

จากภาพที่ 17 ในขั้นแรกจะกำจัด FD ที่ซ้ำซ้อนโดยใช้วิธีการของ MEMBERSHIP ALGORITHM [11] ในขั้นที่สองจะหาคีย์ของแต่ละความสัมพันธ์ [11] โดยจะพิจารณาจาก FD ที่มีอยู่ หลังจากนั้นจะนำ Determinant ของทุก FD มาตรวจสอบดูว่าเป็นคีย์หรือไม่ ถ้าไม่เป็นก็จะทำการแยกความสัมพันธ์นั้นออกมาเป็นอีกความสัมพันธ์หนึ่ง หลักการที่สำคัญในการแยกความสัมพันธ์คือใช้หลักการ Nonloss Decomposed คือเมื่อแยกแล้วต้องไม่มีข้อมูลหรือความสัมพันธ์สูญหาย และ

เมื่อทำการเชื่อมความสัมพันธ์กลับต้องได้เหมือนเดิม ดังนั้นคอลัมน์ที่จะใช้เชื่อมต้องเป็นคีย์ของความสัมพันธ์กล่าวคือในบางครั้งการแยกความสัมพันธ์จะต้องมีการตัดสินใจเลือกแอททริบิวท์ที่จะคงอยู่ไว้ และแอททริบิวท์ที่จะแยกออกไป เช่น มีความสัมพันธ์ดังนี้

AB -> C , AB -> D , D -> A

คีย์ของความสัมพันธ์นี้คือ DB และ AB จาก D -> A จะเห็นว่า D ไม่ได้เป็นคีย์ ดังนั้นจึงต้องแยก FD นี้ออกมาจากความสัมพันธ์ ทำให้ได้ความสัมพันธ์ใหม่ 2 ความสัมพันธ์คือ

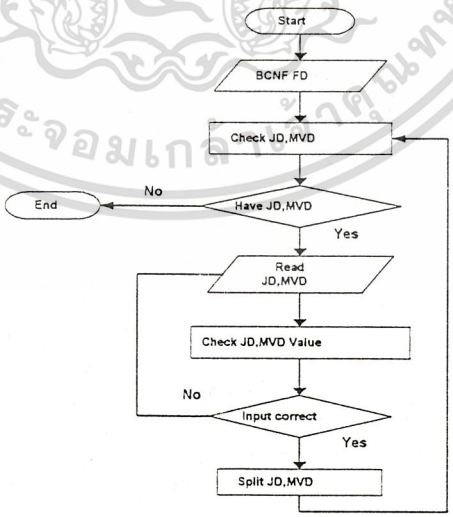
AB -> C และ D -> A

ในการแยก D -> A ออกมา จะเห็นว่าแอททริบิวท์ A เป็นแอททริบิวท์ที่ใช้เชื่อม ความสัมพันธ์ทั้งสองที่แยกมา (comunon attribute) แต่ A ไม่ได้เป็น determinant อาจทำให้ความสัมพันธ์สูญหายได้ เพื่อที่จะหลีกเลี่ยงการแยกแล้วทำให้ข้อมูลสูญหาย จึงต้องเลือกการแยกที่มีแอททริบิวท์ที่ใช้ในการเชื่อมความสัมพันธ์เป็น Determinant ของ FD ดังนั้นความสัมพันธ์ใหม่ 2 ความสัมพันธ์ที่ได้คือ

DB -> C และ D -> A

3. การพัฒนาในส่วนรับข้อมูล JD หรือ MVD การพัฒนาในส่วนนี้มีขั้นตอนดังนี้

ภาพที่ 18



แสดงการพัฒนาในส่วนรับข้อมูล JD หรือ MVD และการประมวลผลในส่วนกฎของ JD,MVD

จากรูปหลังจากผ่านขบวนการ BCNF RULE แล้วจะได้ FD มาจำนวนหนึ่ง ซึ่งเป็น FD ที่ไม่มีความซ้ำซ้อนแล้ว จากนั้นจะนำ FD ที่ได้มาทำการตรวจสอบว่ามีความเป็นไปได้ที่จะมี MVD

หรือ JD แอบแฝงอยู่หรือไม่ ถ้าไม่มี ความสัมพันธ์ที่ได้ก็อยู่ในรูปของ 5NF เลย แต่ถ้ามีก็จะให้ผู้ใช้ใส่ความสัมพันธ์ JD หรือ MVD ที่แอบแฝงอยู่ หลังจากที่ใส่ความสัมพันธ์แล้วก็จะทำการตรวจสอบดูว่าความสัมพันธ์นั้นถูกต้องหรือไม่ ถ้าไม่ถูกต้องก็จะให้ผู้ใช้ใส่ความสัมพันธ์ใหม่ ถ้าถูกต้องแล้วก็จะส่งความสัมพันธ์ที่ได้สู่ระบบ ตรวจสอบ JD และ MVD ต่อไป

4. การพัฒนาฐานข้อมูลความรู้ในส่วนวิเคราะห์ JD and MVD RULE มีขั้นตอนดังต่อไปนี้

ทำการตรวจสอบความถูกต้องของข้อมูล JD หรือ MVD ที่รับเข้ามาดังนี้

กฎข้อที่ 1 การกำหนดแอทริบิวท์ในความสัมพันธ์ JD ในแต่ละครั้งจะต้องประกอบด้วยแอททริบิวต์จำนวนไม่ต่ำกว่า 2 แอททริบิวต์

กฎข้อที่ 2 ความสัมพันธ์ JD จะต้องครอบคลุมทุกแอททริบิวต์

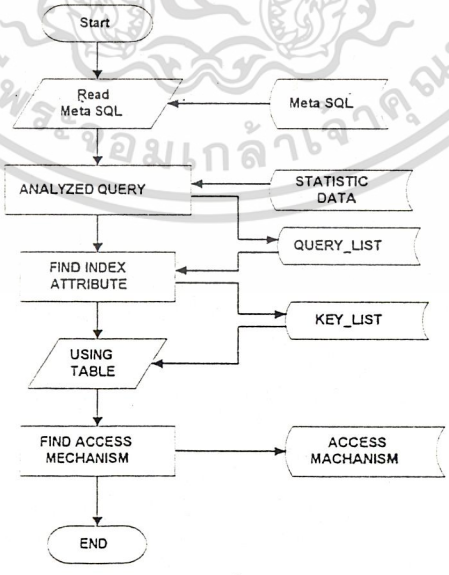
กฎข้อที่ 3 ความสัมพันธ์ JD จะต้องเป็น Consequence ของ Candidate Key

เมื่อตรวจสอบความถูกต้องตามกฎแล้วจะแยกความสัมพันธ์ตาม JD ที่กำหนด

ผลลัพท์ที่ได้จากขบวนการนี้จะเป็นความสัมพันธ์ที่อยู่ในรูป 5 NF

<u>การพัฒนาระบบในส่วนการเลือกรูปแบบและกลไกในการเข้าถึงข้อมูล</u>

ภาพที่ 19

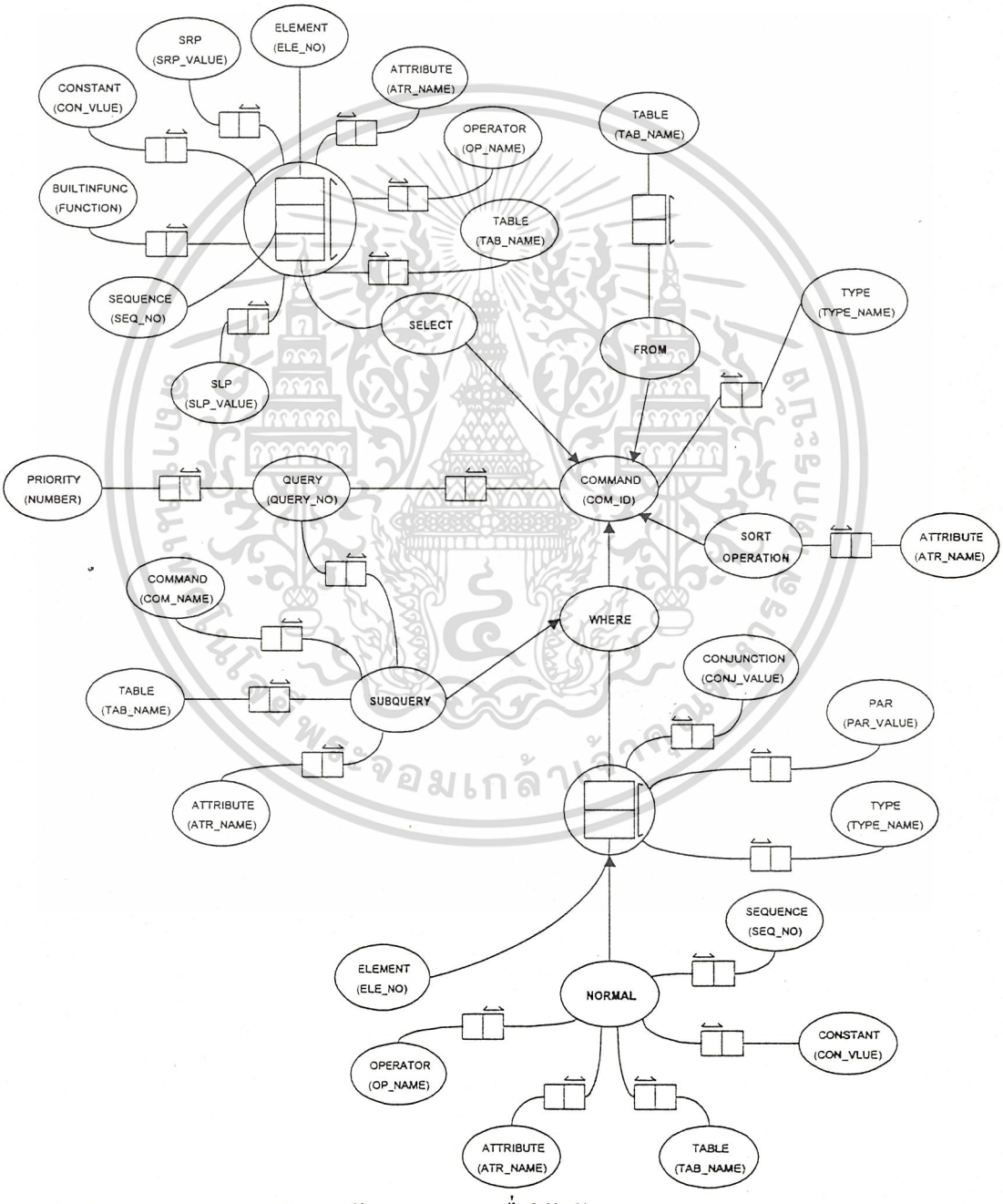


แสดงขั้นตอนการทำงานในส่วนของการเลือกรูปแบบและกลไกในการเข้าถึงข้อมูล

1. การพัฒนาส่วนรับข้อมูล Meta SQL ส่วนนี้พัฒนาด้วยภาษา C โดยจะสร้างหน้าจอสำหรับให้ผู้ใช้ป้อน Query ที่ใช้งานอยู่ประจำ ซึ่งจะอยู่ในรูปของของภาษา SQL พร้อมทั้งให้ผู้ใช้กำหนด

ความสำคัญของแต่ละ Query โดยจะกำหนดให้เป็นเลขจำนวนเต็มอยู่ในระหว่าง 1 - 10 โดยที่ 1 จะมีความสำคัญต่ำสุดและ 10 จะมีความสำคัญสูงสุด หลังจากนั้นจะเก็บรายละเอียดทั้งหมดลงบนแฟ้มข้อมูล Meta SQL ซึ่งมีรายละเอียดดังต่อไปนี้

ภาพที่ 20



แสดง แบบจำลองข้อมูล NIAM เพื่อใช้สร้าง METASQL TABLE

จากแบบจำลอง NIAM ข้างต้น สามารถนำมาสร้างตารางข้อมูลได้ดังนี้

[COMMAND_ID | QUERY_NO | TYPE]

[QUERY_NO | PRIORITY]

[COMMAND_ID | TABLE]

[COMMAND_ID | ELEMENT_NO | SEQUENCE_NO | TABLE | ATTRIBUTE | SRP | SLP | OPERATOR | CONSTANT | BUILTIN_FUNC]

[COMMAND_ID | ELEMENT_NO | CONJUNCTION | TYPE | WPAR]

[COMMAND_ID | ATTRIBUTE]

[COMMAND_ID | QUERY_NO | TABLE | ATTRIBUTE | COMMAND]

[COMMAND_ID | ELEMENT_NO | SEQUENCE_NO | TABLE | ATTRIBUTE | OPERATOR | CONSTANT]

2. การพัฒนาส่วนสร้างตารางสถิติเก็บข้อมูลจากฐานข้อมูลที่มีอยู่ ในส่วนนี้พัฒนาด้วยภาษา C ที่เป็น Embedded SQL ของ INGRES ตัวโปรแกรมนี้จะ run บนเครื่องที่มี INGRES และมีฐานข้อมูลที่ใช้งานจริงอยู่ โดยโปรแกรมจะสร้าง Histogram แบบ Relative frequency histogram ในรูปของอันตรภาคชั้น [24] สำหรับเก็บรายละเอียดเกี่ยวกับการกระจายของข้อมูล ซึ่งจะกำหนดค่าเบื้องต้นในการแบ่งชั้นไว้เป็น 15 ชั้น ซึ่งมีรายละเอียดดังนี้

| DATABASE NAME | TABLE | ATTRIBUTE | ROWRETURN | DATA VALUE |

แอททริบิวต์ที่ 1 เป็นชื่อฐานข้อมูลที่ใช้

แอททริบิวต์ที่ 2 เป็นชื่อตารางข้อมูลที่ใช้

แอททริบิวต์ที่ 3 เป็น Relative Frequency (f/n) ซึ่งคำนวณได้จาก (จำนวนของข้อมูลจริงที่ตกอยู่ในช่วงของอันตรภาคชั้นใด ๆ / จำนวนของข้อมูลทั้งหมดที่มีอยู่) [24]

แอททริบิวต์ที่ 4 เป็นค่าข้อมูลในแอทริบิวท์

ตัวอย่างของข้อมูลจากตารางสถิติ

test , borrow , std_id, 0.036313 , "010074" ,

test , borrow , std_id, 0.034916 , "010114" ,

test , borrow , std_id , 0.034916 , "010157" ,

test , borrow , std_id , 0.034916 , "020016" ,

3. ส่วนการวิเคราะห์การใช้งาน Query SQL กับฐานข้อมูลที่มีอยู่ ในส่วนนี้จะดึง Fact จากแฟ้มข้อมูล Meta SQL มาวิเคราะห์ เพื่อหาลักษณะการใช้งานตารางข้อมูลจาก Query พร้อมทั้งนำข้อมูลจากตารางสถิติที่สอดคล้องกับลักษณะการใช้งานตารางข้อมูลขึ้นมา แล้วเก็บรายละเอียดลงบน list ซึ่งมีรายละเอียดดังต่อไปนี้

JOIN_LIST

จะเป็นการเก็บรายละเอียดเกี่ยวกับการใช้งานตารางข้อมูลร่วมกันตั้งแต่สองตารางขึ้นไป ซึ่งการตรวจสอบจะแบ่งเป็นสองลักษณะคือ

การเชื่อมตารางแบบธรรมดา คือมีการระบุลักษณะการเชื่อมตารางภายใน where condition ภายในหนึ่ง Query เดียวกันเช่น

```
select p_call_no,p_name
from pcalln a,prlib b
where a.p_call_no = b.p_call_no
and  p_ad_yr > '1980';
```

จาก query ข้างต้น จะสามารถเก็บรายละเอียดลงบน Join_list ได้ดังนี้

((pcalln,p_call_no) (prlib,p_call_no))

การเชื่อมตารางแบบ Subquery เช่น

```
select p_call_no
from  pcalln
where p_call_no in (
select p_call_no
from p_call_no
where p_ad_yr > '1980');
```

จาก query นี้ จะสามารถเก็บรายละเอียดลงบน Join_list ได้ดังนี้

((pcalln,p_call_no) (prlib,p_call_no))

RANGE_LIST

จะเก็บรายละเอียดเกี่ยวกับเงื่อนไขที่เป็นช่วงโดยจะเก็บ ชื่อตาราง ชื่อคอลัมน์ และค่า Rowreturn ของเงื่อนไขนั้นเช่น

```
select p_call_no,p_name
```

```
from pcalln
where p_length > 20;
```

จาก query ข้างต้น จะสามารถเก็บรายละเอียดลงบน Range_list ได้ดังนี้

((pcalln ,(p_length),0.45))

0.45 เป็นค่า Rowreturn ซึ่งแสดงค่าการกระจายของข้อมูลที่ตกอยู่ในช่วงของเงื่อนไขนั้น โดยได้มาจากการคำณวนจากตารางสถิติ

LIKE_LIST

จะเก็บรายละเอียดเกี่ยวกับเงื่อนไขที่เป็นลักษณะของ pattern matching โดยจะเก็บ ชื่อตาราง ชื่อคอลัมน์ และค่า Rowreturn ของเงื่อนไขนั้นเช่น

```
select p_call_no,p_name
from pcalln
where p_publih_plc like 'LONDON%';
```

จาก query ข้างต้น จะสามารถเก็บรายละเอียดลงบน Like_list ได้ดังนี้

((pcalln ,(p_publish_plc),0.25))

EQUAL_LIST

จะเก็บรายละเอียดเกี่ยวกับเงื่อนไขที่อยู่ในรูปของเงื่อนไขเท่ากับ โดยจะเก็บ ชื่อตาราง ชื่อคอลัมน์ และค่า Rowreturn ของเงื่อนไขนั้นเช่น

```
select p_call_no,p_name
from pcalln
where p_publih_plc = 'NEWYORK';
```

จาก query ข้างต้น จะสามารถเก็บรายละเอียดลงบน Equal_list ได้ดังนี้

((pcalln ,(p_publish_plc),0.36))

NOT_EQUAL_LIST

จะเก็บรายละเอียดเกี่ยวกับเงื่อนไขที่อยู่ในรูปของเงื่อนไขเท่ากับ โดยจะเก็บชื่อตารางและชื่อคอลัมน์ของเงื่อนไขนั้นเช่น

```
select p_call_no,p_name
from pcalln
```

where p_publih_plc != 'NEWYORK';

จาก query ข้างต้น จะสามารถเก็บรายละเอียดลงบน Not_equal_list ได้ดังนี้

((pcalln ,(p_publish_plc)))

4. การพัฒนาส่วนรับข้อมูล ในส่วนของลักษณะการใช้งานตารางข้อมูล ในส่วนนี้จะสร้างหน้าจอสำหรับให้ผู้ใช้ตอบคำถามเกี่ยวกับลักษณะการใช้งานตารางข้อมูลต่าง ๆ ซึ่งมีรายละเอียดดังนี้

ข้อมูลในตารางมีลักษณะเป็น static หรือไม่ ?

ข้อมูลในตารางมีลักษณะเป็น dynamic หรือไม่ ?

ผู้ใช้มีการเปลี่ยนแปลงแก้ไขข้อมูลในตารางในลักษณะทำพร้อม ๆ กันหลายคนหรือไม่ ?

ข้อมูลในตารางมีการเปลี่ยนแปลงแก้ไข บ่อยใช่หรือไม่ ?

มีการลบข้อมูลในตารางบ่อยใช่หรือไม่ ?

จะเห็นว่าคำถามเหล่านี้เป็นคำถามในลักษณะ yes/no โดยจะให้ผู้ใช้กำหนดว่าการใช้งานตารางข้อมูลตามรูปแบบต่าง ๆ ที่ได้กำหนดเป็นจริงหรือเท็จ ซึ่งในทางปฏิบัติแล้วการที่กำหนดค่าจริงหรือเท็จตามเหตุการณ์ต่าง ๆ เหล่านี้นั้น อาจมีความไม่แน่นอนเข้ามาเกี่ยวข้องด้วย ดังนั้นเพื่อความสมบูรณ์ และถูกต้อง จึงให้ผู้ใช้สามารถกำหนดค่าความไม่แน่นอนของค่าความจริงที่ตอบคำถาม ซึ่งจะกำหนดในรูปของ certainty factor range : CF โดยในแต่ละคำถามเหล่านี้จะมีตัวแปรมารองรับเพื่อเก็บค่า CF ของความจริงในแต่ละคำถามดังแสดงหน้าจอต่อไปนี้

ข้อมูลในตารางมีลักษณะเป็น static หรือไม่ ?

[. . . . .] (yes/no)

จะเห็นว่ามีจุดอยู่ 10 จุด ถ้าเราเลื่อนแถบแสงเลือกครบทั้ง 10 จุดแสดงว่ามีความจริงในการเลือกคำตอบร้อยเปอร์เซนต์

5. การพัฒนาส่วนวิเคราะห์การเลือกรูปแบบ และกลไกในการเข้าถึงข้อมูล ในส่วนนี้จะเป็นการนำ fact ที่ได้มาจากการวิเคราะห์ Query SQL ที่ใช้อยู่ประจำซึ่งจะอยู่ในรูปของ list ต่าง ๆที่ได้กล่าวมาแล้วข้างต้นร่วมกับ Fact ที่ได้มาจากลักษณะการใช้งานตารางข้อมูล และข้อมูลจากตารางสถิติ มา Inference ร่วมกับ Rule ที่ได้มาจากการสรุปการทดสอบการประมวลผล Query ของ Ingres ตามทฤษฎีของการกำหนดรูปแบบ และกลไกในการเข้าถึงข้อมูลในบทที่ 4 การทำงานในส่วนนี้แบ่งได้เป็น 2 ขั้นตอนคือ

ขั้นแรก จะหาแอททริบิวท์ที่จะสร้าง index เพื่อเข้าถึงข้อมูลก่อน ในส่วนนี้จะใช้ fact ที่ได้มาจากการวิเคราะห์ Query Sql ร่วมกับข้อมูลจากตารางสถิติมาทำการ inference ร่วมกับกฎเกณฑ์ดังตัวอย่างต่อไปนี้

กำหนดให้ list ที่ได้มาจากการวิเคราะห์ Query SQL มีดังนี้

Join_list = (((pcalln ,(p_call_no)),(prlib,(p_no))))

Equal_list = ((pcalln,(p_publish_plc,p_ad_yr),0.468),(prlib,(p_name),0.765))

จาก list ข้างต้นจะเห็นว่ามีการใช้งานตารางข้อมูลสองตารางร่วมกัน และมีเงื่อนไขในการเข้าถึงข้อมูลทั้งสองตารางในกรณีนี้จะเข้ากฎในการพิจารณาที่ว่า เมื่อมีการใช้งานตารางสองตารางร่วมกันและมีเงื่อนไขในการดึงข้อมูลทั้งสองตาราง จะสร้าง index บนคอลัมน์ที่เป็นเงื่อนไขในการดึงข้อมูลที่มีค่า Row Return น้อยที่สุด และสร้าง index บนคอลัมน์ที่ใช้ในการเชื่อมตาราง ของตารางที่มีค่า Row Return ของเงื่อนไขในการดึงข้อมูลมากกว่า ซึ่งจาก list ข้างต้นจะได้ list ใหม่ที่ใช้ในการเก็บตาราง และคอลัมน์ที่ใช้สร้าง index ได้ดังนี้

Key_list = ((pcalln , (p_publish_plc,p_ad_yr)) , (prlib,(p_no)))

ขั้นที่สอง จะเป็นการเลือกชนิดของกลไกในการเข้าถึงข้อมูล โดยในงานวิจัยชิ้นทำกับ INGRES ซึ่งมีวิธีในการเข้าถึงข้อมูลได้ 4 วิธีคือ HEAP , HASH , ISAM และ BTREE ในบางกรณีมีการใช้งานที่แตกต่างกันบนตารางข้อมูลเดียวกัน การใช้งานที่แตกต่างกันนั้นบางครั้งทำให้เกิดการขัดแย้งกัน ดังนั้นจึงกำหนดให้มีค่า weight_parameter ของแต่ละวิธี โดยกำหนดให้อยู่ในรูปของความน่าจะเป็นที่จะเลือกวิธีในการเข้าถึงข้อมูลแต่ละวิธีดังกล่าวข้างต้น

โดยทั่วไปแนวความคิดในการพิจารณาความน่าจะเป็นแบ่งได้เป็น 2 แนวคิดคือ [21]

แนวคิดแรก กำหนดว่าความน่าจะเป็นของเหตุการณ์ใด ๆ ควรจะเป็นความถี่ของเหตุการณ์นั้นที่เกิดขึ้น โดยเทียบจากเหตุการณ์ที่เกิดขึ้นมาก ๆ ปัญหาของแนวความคิดนี้คือ จำนวนครั้งของเหตุการณ์ที่เกิดขึ้นควรเป็นปริมาณเท่าใดจึงจะถือว่ามากพอ และในทางปฏิบัติเราสามารถทำให้เหตุการณ์นั้นเกิดขึ้นซ้ำ ๆ กันได้หรือไม่

แนวคิดที่สอง กำหนดว่าความน่าจะเป็น P ของ สมมติฐาน H ซึ่งแสดงโดย P[H] จำนวนตั้งแต่ 0 - 1 ซึ่งใช้แสดงความเชื่อในสมมติฐาน H ส่วนความน่าจะเป็นพร้อมเงื่อนไขของสมมติฐาน หรือ P[H|E] คือความน่าจะเป็นของสมมติฐาน H ภายหลังจากที่รู้หลักฐาน E ซึ่งแนวความคิดนี้มีชื่อว่าทฤษฎีของเบย์ (Baysian theory)

เนื่องจากงานวิจัยชิ้นนี้เกี่ยวข้องกับประสบการณ์ และความเชื่อส่วนตัวของผู้เชี่ยวชาญ และผู้ใช้ระบบจึงเหมาะสมกับแนวความคิดที่สองมากกว่า

ทฤษฎีของเบย์

ในทฤษฎีของเบย์กำหนดให้ความน่าจะเป็นของสมมติฐานก่อน และหลังได้รับหลักฐานอาจมีค่าแตกต่างกัน โดยกำหนดให้ P[Hk] เป็นความน่าจะเป็นของสมมติฐาน Hk ก่อนได้รับหลักฐาน และกำหนดให้ P[Hk|E] เป็นความน่าจะเป็นของสมมติฐาน Hk หลังได้รับหลักฐาน E ซึ่งทฤษฎีของเบย์มีวิธีคำนวณดังนี้

กำหนด [H1,H2,H3,...,Hn] เป็นสมมติฐานที่ไม่ซ้ำกัน และให้ P[Hi] != 0 สำหรับ i = 1,2,3,...,n นอกจากนั้นให้ E เป็นหลักฐาน โดยที่ P[E] != 0 จะได้ว่า [23]

$$P[Hk|E] = \frac{P[Hk]\,P[E|Hk]}{P[E]} = \frac{P[Hk]\,P[E|Hk]}{\sum_{i=0}^{n} P[Hi]\,P[E|Hi]}$$

สำหรับ k = 1,2,3,...,n

สำหรับในงานวิจัยนี้ กำหนดให้สมมติฐาน Hi คือ กลไกในการเข้าถึงข้อมูลตามวิธีต่าง ๆ ที่ได้กล่าวมาในตอนต้น และหลักฐานนำมาใช้ประกอบกับสมมติฐานคือ ลักษณะการใช้งานตารางข้อมูลในฐานข้อมูล ซึ่งจะประกอบไปด้วยลักษณะที่ได้จาก Query และได้จากตารางในบทที่ 4 กำหนดให้ความน่าจะเป็นในการที่จะเกิดสมมติฐานก่อนได้รับหลักฐาน P[Hi] มีค่าเท่ากัน สำหรับความน่าจะเป็นของการเกิดสมมติฐานต่าง ๆ ณ หลักฐานนั้น P[E|Hi] มีค่าเท่ากับค่าจากการแปลงค่าจากตาราง ในบทที่ 4 โดยจะแปลงค่าสถานะในตารางให้เป็นค่าความน่าจะเป็นดังตัวอย่างต่อไปนี้

จากตารางแบ่งสถานะในการกำหนดไว้เป็น 4 ระดับ คือ Bad,OK,Good,Excellent โดยให้ Excellent มีค่าความน่าจะเป็นสูงสุดในการที่จะถูกเลือกมีค่าเท่ากับ 1.00 ส่วน Good มีค่าความน่าจะเป็นในการเลือกรองลงมามีค่าเท่ากับ 0.75 สำหรับ OK และ Bad มีค่าเท่ากับ 0.50 และ 0.25 ลดลงมาตามลำดับ

จากที่ได้กล่าวไว้แล้วว่าการรับค่าลักษณะการใช้งานตารางข้อมูล ในส่วนที่นอกเหนือจากการใช้งานจาก Query จากผู้ใช้ระบบ มีค่าความไม่แน่นอนเข้ามาเกี่ยวข้องด้วย ดังนั้นจึงมีความไม่แน่นอนของหลักฐานเกิดขึ้น เบย์ได้กำหนดทฤษฎีความหน้าจะเป็นในกรณีที่หลักฐานมีความไม่แน่นอนเกิดขึ้นไว้ดังนี้

กำหนด E เป็นหลักฐานในอุดมคติที่ถูกต้องครบสมบูรณ์ และ E' เป็นหลักฐานที่อาจมีความไม่แน่นอนเกิดขึ้น โดย P[E|E'] จะแสดงความน่าจะเป็นที่ E' จะเป็น E และ P [Ec|E'] จะแสดงความน่าจะเป็นที่ E' จะไม่เป็น E โดยที่ Ec เป็นเซ็ทของคู่ประกอบ (comlement set) ของ E

กล่าวคือ E U Ec = U โดย U เป็น universal set ดังนั้นถ้าคำนึงถึงความไม่แน่นอนของหลักฐานที่มีต่อสมมติฐานจะได้ว่า [23]

$$P[Hk|E'] = P[Hk|E]\ P[E|E'] + P[Hk|Ec]\ P[Ec|E']$$

ซึ่งค่า P[E|E'] ได้จากค่า certainty factor ที่ผู้ใช้ใส่ในกรณีที่ถามถึงการใช้งานตารางที่นอกเหนือจาก Query ส่วนความน่าจะเป็นของหลักฐานการใช้งานตารางที่เกี่ยวข้องกับ Query จะกำหนดให้มีค่าความน่าจะเป็นเท่ากับ 1.0 เพราะหลักฐานนั้นได้ระบุแน่ชัดไว้ใน Query อยู่แล้วคือมีความแน่นอนร้อยเปอร์เซ็น สำหรับค่า P[Ec|E'] = 1 - P[E|E'] และ ค่า P[Hk|Ec] ก็ได้จากทฤษฎีของเบย์ในสูตรแรกเช่นกัน เมื่อได้ค่าความน่าจะเป็นของแต่ละสมมติฐาน และของแต่ละหลักฐานแล้ว ก็จะนำมาหาค่าเฉลี่ยของค่าความน่าจะเป็นของแต่ละสมมติฐานภายใต้หลักฐานทั้งหมด จากนั้นจะเลือกสมมติฐานที่มีความน่าจะเป็นสูงสุดมาเป็นกลไกในการเข้าถึงข้อมูล

การพิจารณาค่าให้กับ P[E|E'] แบ่งได้เป็นสองส่วน

- ส่วนแรก พิจารณาจาก Query_list ในส่วนนี้จะนำ Key_list ซึ่งเป็น Fact ที่ได้มาจากขบวนการแรกมาพิจารณาร่วมกับ List อื่น ๆ ที่กล่าวมาแล้วข้างต้น เพื่อให้ทราบถึงลักษณะการใช้งานตารางโดยดูจากลักษณะของเงื่อนไขในการดึงข้อมูล เช่นถ้าเงื่อนไขในการดึงข้อมูลมาจาก LIKE_LIST ทำให้ทราบว่าการใช้งานตารางข้อมูลเป็นลักษณะ pattern matching เป็นต้น สำหรับการกำหนดค่า certainty factor นั้น จะเห็นว่าการใช้งานตารางข้อมูลในส่วนนี้ เป็นไปในลักษณะที่ชัดเจน และแน่นอนจึงให้ค่า certainty factor มีค่าเท่ากับร้อยเปอร์เซนต์ หรือมีค่าความน่าจะเป็นเท่ากับ 1.0

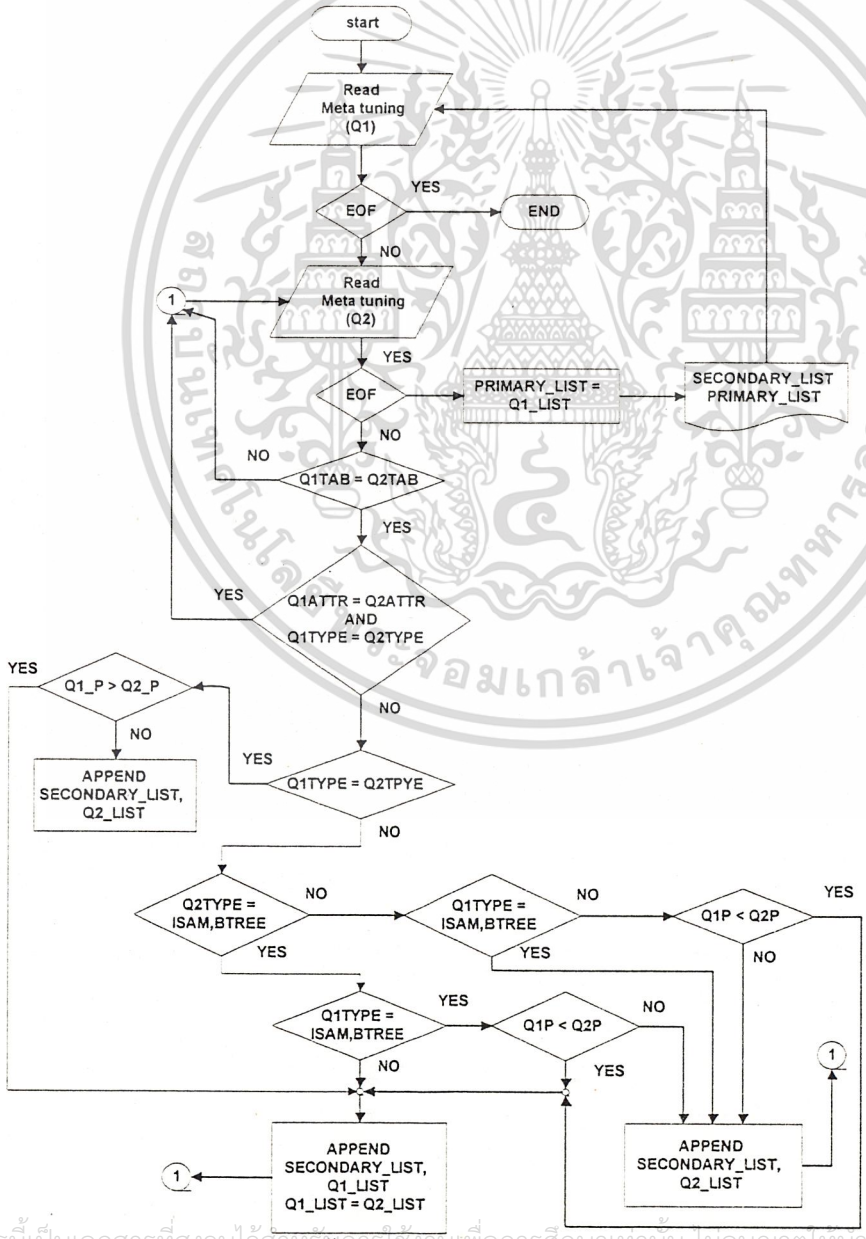
-ส่วนที่สอง เป็นการใช้งานตารางที่นอกเหนือจากการใช้งานโดย Query ซึ่งในส่วนนี้จะแสดงออกมาให้ผู้ใช้เป็นผู้กำหนดค่า certainty factor ของแต่ละลักษณะการใช้งานตารางข้อมูล ดังตัวอย่างที่ได้กล่าวมาแล้ว

จากนั้นก็จะเก็บรายละเอียดทั้งหมด ที่เกี่ยวกับการกำหนดรูปแบบและกลไกในการเข้าถึงข้อมูลลงตาราง Meta tune ซึ่งมีรายละเอียดดังต่อไปนี้

| DATABASE_NAME | USER_NAME | QUERY_NO | TABLE_NAME | COLUMN_NAME | TYPE_INDEX |

6. การพัฒนาในส่วน Multi Query tunning จากที่ผ่านมาเราพิจารณากำหนดรูปแบบ และกลไกในการเข้าถึงข้อมูลที่เหมาะสมของแต่ละ Query แต่ในทางเป็นจริงแล้วการใช้งานจะมี Query มากกว่าหนึ่ง ซึ่งมีความเป็นไปได้ที่มีความต้องการใช้งานของแต่ละ Query ต่าง ๆ ที่แตกต่างกัน บนตารางข้อมูล และคอลัมน์เดียวกัน ซึ่งอาจทำให้ได้ผลลัพธ์ที่แตกต่างกัน ในส่วนนี้จะใช้ Secondary index เข้ามาช่วยแก้ปัญหาตามกฎที่กล่าวไว้ในตอนท้ายของบทที่ 4 โดยจะอ่านข้อมูลที่เกี่ยวกับการกำหนดรูปแบบและกลไกในการเข้าถึงข้อมูลของแต่ละ Query จากตาราง Meta_tune นำมา inference ร่วมกับ Rule ในส่วนของ Multiquery tuning ซึ่งมีขั้นตอนดังต่อไปนี้

ภาพที่ 21



แสดงขั้นตอนการทำงานในส่วน Multiquery tuning

บทที่ 9

การทดสอบ และใช้งาน EX/DT2

<u>บทนำ</u>

ในบทนี้จะเป็นการทดสอบความถูกต้องของระบบ EX/DT2 พร้อมทั้งอธิบายการใช้งานควบคู่ไปด้วย โดยในส่วนแรกจะเป็นการแสดงข้อมูลที่จะนำมาทำการทดสอบ และการเตรียมข้อมูลเพื่อที่จะนำเข้าสู่ระบบในส่วนออกแบบฐานข้อมูล ในส่วนที่สองจะกล่าวถึงการใช้งาน และการทดสอบการทำงานของ EX/DT2 ในส่วนการออกแบบฐานข้อมูล พร้อมแสดงผลที่ได้จากการออกแบบ ในส่วนที่สามเป็นการแสดงข้อมูลที่จะนำมาทำการทดสอบ และการป้อนข้อมูลสู่ระบบในส่วนการเลือกกลไกการเข้าถึงข้อมูล ในส่วนสุดท้ายจะกล่าวถึงการใช้งาน และการทดสอบการทำงานของ EX/DT2 ในส่วนการเลือกกลไกการเข้าถึงข้อมูล พร้อมทั้งแสดงผลที่ได้จากระบบนี้

<u>การนำข้อมูลเข้าสู่ระบบ EX/DT2 ในส่วนการออกแบบฐานข้อมูล</u>

1. ข้อมูลที่จะนำมาทดสอบ

ข้อมูลที่จะนำมาทดสอบแบ่งได้เป็นทั้งหมด 6 ชุด ด้วยกัน มีรายละเอียดดังนี้

ข้อมูลชุดที่หนึ่ง เป็นของ **SUPPLIERS** ซึ่งจะเก็บรายละเอียดเกี่ยวกับ ชื่อ เมือง และระยะทางของ SUPPLIER ซึ่งประกอบด้วยแอททริบิวท์ดังต่อไปนี้

SUPPLIER , CITY , DISTANCE

จากแอททริบิวท์เหล่านี้สามารถนำมาเขียนความสัมพันธ์ ซึ่งแสดงได้ด้วย FD : function dependencyดังต่อไปนี้

SUPPLIER ---> CITY

SUPPLIER ---> DISTANCE

CITY ---> DISTANCE

ข้อมูลชุดที่สอง เป็นของ **RELATION_GRADES** ใช้เก็บผลการศึกษาของนักศึกษาทุกๆ วิชา และหมายเลขโทรศัพท์ของนักศึกษา ซึ่งประกอบด้วยแอททริบิวท์ดังต่อไปนี้

STUDENT_ID , PHONE_NO , COURSE , GRADE

จากแอททริบิวท์เหล่านี้สามารถนำมาเขียนความสัมพันธ์ ซึ่งแสดงได้ด้วย FD ดังต่อไปนี้

STUDENT_ID , COURSE ---> GRADE

STUDENT_ID ---> PHONE_NO

ข้อมูลชุดที่สาม เป็นของ **SALES-AREAS** จะเก็บรายละเอียดเกี่ยวกับพื้นที่การขาย ซึ่งประกอบด้วยแอททริบิวท์ดังต่อไปนี้

REPRESENTATIVE , CUSTOMER , PRODUCT-CLASS

จากแอททริบิวท์เหล่านี้สามารถนำมาเขียนความสัมพันธ์ ซึ่งแสดงได้ด้วย FD ดังต่อไปนี้

REPRESENTATIVE , CUSTOMER , PRODUCT-CLASS ---> REPRESENTATIVE , CUSTOMER , PRODUCT-CLASS

ข้อมูลชุดที่สี่ เป็นของ **ATTENDANCE** จะเก็บรายละเอียดเกี่ยวกับตารางการสอนของครูแต่ละคน ซึ่งประกอบด้วยแอททริบิวท์ดังต่อไปนี้

TEACHER , SEMESTER , SUBJECT , SECTION , ATTENDANCES

จากแอททริบิวท์เหล่านี้สามารถนำมาเขียนความสัมพันธ์ ซึ่งแสดงได้ด้วย FD ดังต่อไปนี้

TEACHER , SEMESTER , SECTION ---> ATTENDANCES

SUBJECT , SEMESTER , SECTION ---> ATTENDANCES

SEMESTER , SUBJECT --> TEACHER

TEACHER , SEMESTER --> SUBJECT

ข้อมูลชุดที่ห้า เป็นของ **PRLIB** จะเก็บรายละเอียดเกี่ยวกับหนังสือในห้องสมุด ซึ่งประกอบด้วย แอททริบิวท์ดังต่อไปนี้

P_ACCESS_NO , P_PUBLISH_PLC , P_AD_YR , P_BE_YR ,P_CALL_NO

จากแอททริบิวท์เหล่านี้สามารถนำมาเขียนความสัมพันธ์ ซึ่งแสดงได้ด้วย FD ดังต่อไปนี้

P_ACCESS_NO ---> P_PUBLISH_PLC

P_ACCESS_NO ---> P_AD_YR

P_ACCESS_NO ---> P_BE_YR

P_ACCESS_NO ---> P_CALL_NO

ข้อมูลชุดที่หก เป็นของ **PCALLN** จะเก็บรายละเอียดเกี่ยวกับหนังสือในห้องสมุด ซึ่งประกอบด้วย แอททริบิวท์ดังต่อไปนี้

P_CALL_NO , TITLE_REG1 , TITLE_REG2 , P_LENGTH , P_PUBLISH , AUTHOR_NAME , AUTHOR_SURNAME ,

จากแอททริบิวท์เหล่านี้สามารถนำมาเขียนความสัมพันธ์ ซึ่งแสดงได้ด้วย FD ดังต่อไปนี้

P_CALL_NO ---> TITLE_REG1

P_CALL_NO ---> TITLE_REG2

P_CALL_NO ---> P_LENGTH

P_CALL_NO ---> P_PUBLISH

P_CALL_NO ---> AUTHOR_NAME

P_CALL_NO ---> AUTHOR_SURNAME

2. นำความสัมพันธ์ทั้งหมดเก็บลงแฟ้มข้อมูล

การป้อนข้อมูลจะกระทำโดยผ่านทาง Text editor โดยมีรูปแบบการป้อนข้อมูลดังนี้ ในแต่ละบรรทัดข้อมูลที่ป้อนจะประกอบด้วย 3 คอลัมน์ คอลัมน์แรกจะเป็นชื่อของความสัมพันธ์ คอลัมน์ที่สองเป็นชื่อของแอททริบิวท์ที่เป็น determinant และคอลัมน์ที่สามจะเป็นชื่อของแอททริบิวท์ที่เป็น dependent โดยแต่ละคอลัมน์จะคั่นด้วยที่ว่าง และสำหรับคอลัมน์ที่สองและสามจะต้องอยู่ภายใต้วงเล็บเปิด และวงเล็บปิด ตัวอย่างความสัมพันธ์ SUPPLIERS มี FD ดังนี้

SUPPLIER ---> CITY

สามารถป้อนได้ดังนี้

SUPPLIERS (SUPPLIER) (CITY)

จากข้อมูล FD เบื้องต้นทั้งหมด นำมาป้อนข้อมูลได้ดังนี้

SUPPLIERS (SUPPLIER) (CITY)

SUPPLIERS (SUPPLIER) (DISTANCE)

SUPPLIERS (CITY) (DISTANCE)

RELATION_GRADES (STUDENT_ID COURSE) (GRADE)

RELATION_GRADES (STUDENT_ID) (PHONE_NO)

SALES-AREAS (REPRESENTATIVE CUSTOMER PRODUCT-CLASS)

(REPRESENTATIVE CUSTOMER PRODUCT-CLASS)

ATTENDANCE (TEACHER SEMESTER SECTION) (ATTENDANCES)

ATTENDANCE (SUBJECT SEMESTER SECTION) (ATTENDANCES)

ATTENDANCE (TEACHER SEMESTER) (SUBJECT)

ATTENDANCE (SUBJECT SEMESTER) (TEACHER)

PRLIB (P_ACCESS_NO) (P_PUBLISH_PLC)

PRLIB (P_ACCESS_NO) (P_AD_YR)

PRLIB (P_ACCESS_NO) (P_BE_YR)

PRLIB (P_ACCESS_NO) (P_CALL_NO)

PCALLN (P_CALL_NO) (TITLE_REG1)

PCALLN (P_CALL_NO) (TITLE_REG2)

PCALLN (P_CALL_NO) (P_LENGTH)

PCALLN (P_CALL_NO) (P_PUBLISH)

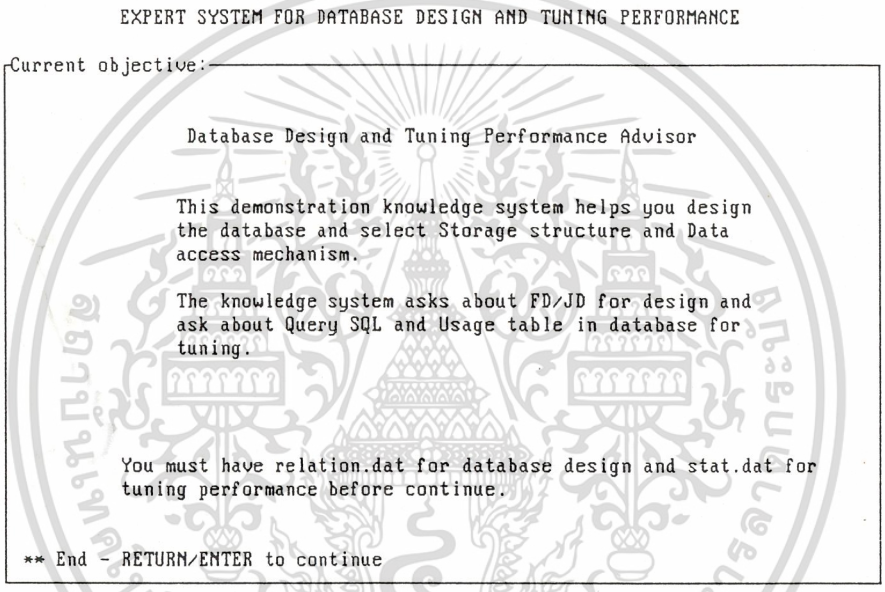
PCALLN (P_CALL_NO) (AUTHOR_NAME)

PCALLN (P_CALL_NO) (AUTHOR_SURNAME)

<u>การใช้งาน และการทดสอบการทำงานของ EX/DT2 ในส่วนการออกแบบฐานข้อมูล</u>

เมื่อเตรียมแฟ้มข้อมูลที่จะนำมาออกแบบฐานข้อมูลเสร็จแล้ว copy แฟ้มข้อมูลดังกล่าวลงในสับไดเรกทอรี่ "\pcplus\data\relation.dat "จากนั้นทำการประมวลผลโปรแกรมระบบผู้เชี่ยวชาญ EX/DT2 โดยมีขั้นตอนการให้คำปรึกษาดังต่อไปนี้

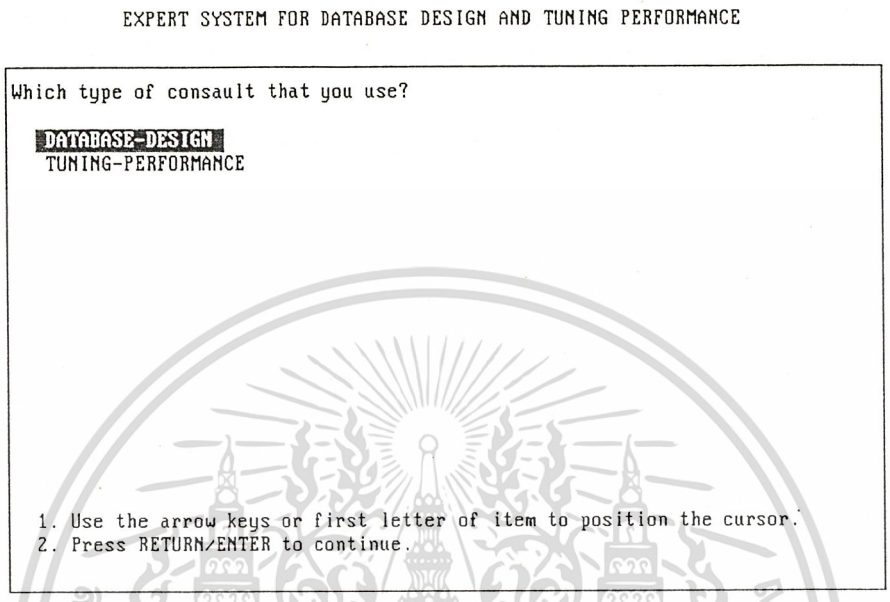
ภาพที่ 22



แสดงหน้าจอบอกรายละเอียดการทำงานของ EX/DT2

เป็นหน้าจอที่อธิบายรายละเอียดการให้คำปรึกษาของระบบ

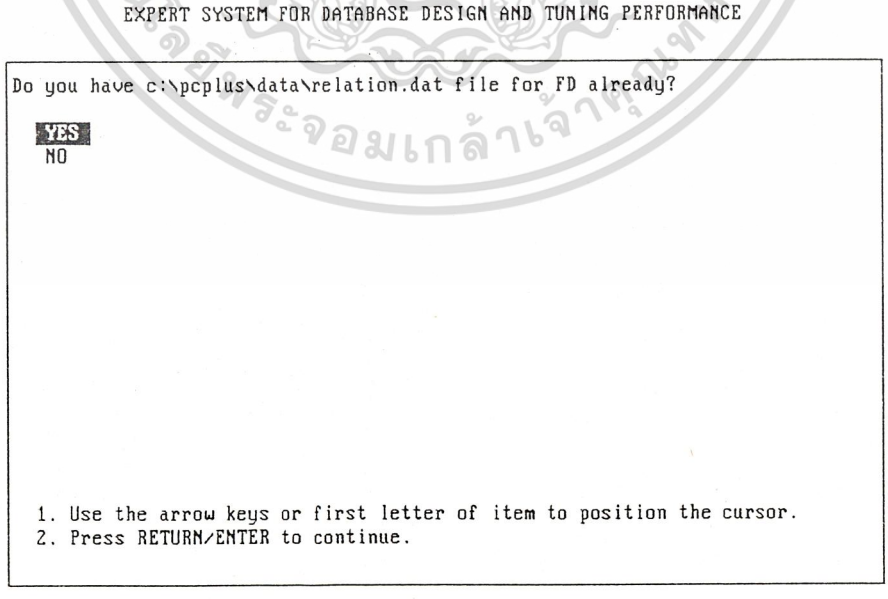
ภาพที่ 23



แสดงหน้าจอสำหรับเลือกชนิดของการให้คำปรึกษา

เป็นหน้าจอที่ให้ผู้ใช้เลือกชนิดของการให้คำปรึกษา ซึ่งมีอยู่สองชนิดคือการออกแบบฐานข้อมูล และการปรับแต่งฐานข้อมูล

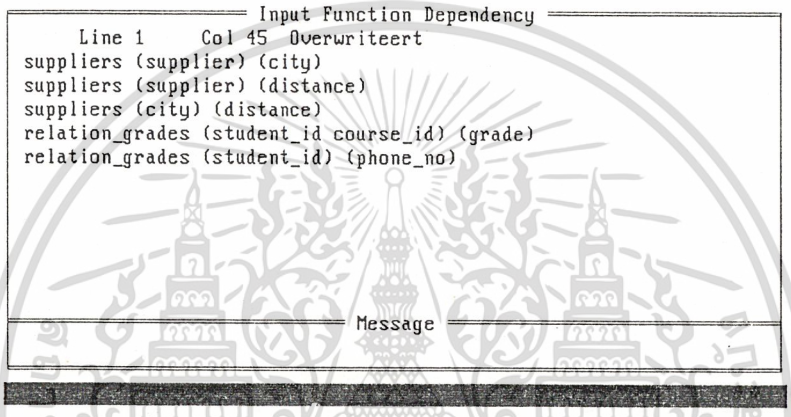
ภาพที่ 24



แสดงหน้าจอให้ผู้ใช้ยืนยันแฟ้มข้อมูล FD

เป็นหน้าจอที่ให้ผู้ใช้ยืนยันความพร้อมของข้อมูล FD : Function dependency ที่จะนำมาใช้ในออกแบบฐานข้อมูล ในกรณีที่ข้อมูลยังไม่พร้อม ระบบจะให้ผู้ใช้ป้อนข้อมูล FD ผ่านทาง editor ของระบบดังนี้

ภาพที่ 25



แสดงหน้าจอ editor ของระบบสำหรับป้อนข้อมูล FD

Function key ที่ใช้ใน editor ของระบบมีดังนี้

F1 ใช้สำหรับ load แฟ้มข้อมูลเพื่อนำมาแก้ไขหรือเพิ่มเติม

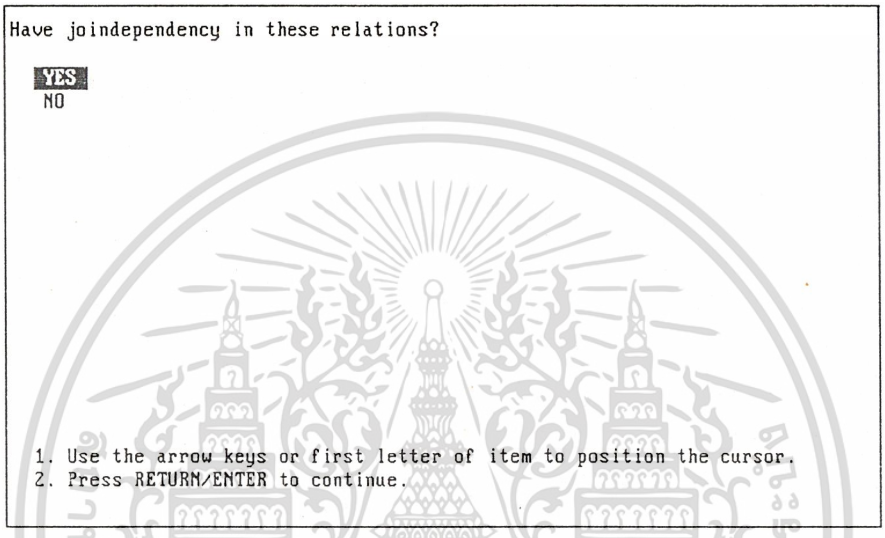
F2 ใช้สำหรับ save ข้อมูล

F10 ใช้สำหรับออกจาก editor

สำหรับคีย์ ที่ใช้ในการเปลี่ยนแปลงแก้ไขข้อมูลเหมือนกับคีย์ที่ใช้งานใน editor มาตราฐานทั่วไป หลังจากเตรียมข้อมูล FD เสร็จแล้ว จะเข้าสู่การประมวลผลในส่วนของการออกแบบฐานข้อมูล

ภาพที่ 26

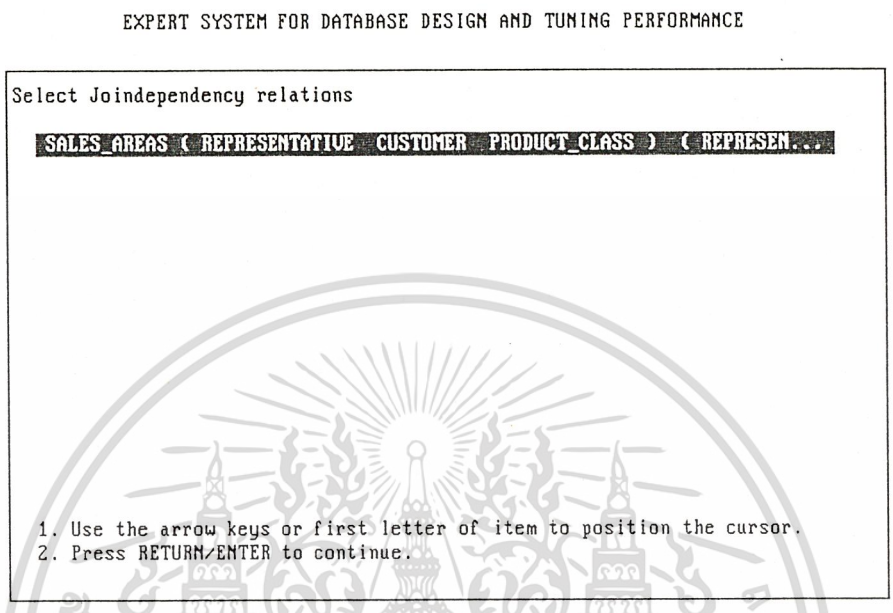
EXPERT SYSTEM FOR DATABASE DESIGN AND TUNING PERFORMANCE



แสดงหน้าจอสำหรับให้ผู้ใช้ยืนยันข้อมูล JD/MVD

เป็นหน้าจอสำหรับแสดงในกรณีที่ ระบบตรวจสอบข้อมูลที่ใช้ในการออกแบบแล้วพบว่ามีความเป็นไปได้ที่จะความสัมพันธ์ JD/MVD แอบแฝงอยู่ ระบบจะแสดงหน้าจอให้ผู้ใช้ยืนยันว่ามีความสัมพันธ์ดังกล่าวอยู่จริงหรือไม่ ถ้าผู้ใช้ยืนยันว่ามีอยู่จริงระบบจะแสดงความสัมพันธ์ทั้งหมดที่สามารถมี JD/MVD แอบแฝงอยู่ได้ให้ผู้ใช้เลือกดังนี้

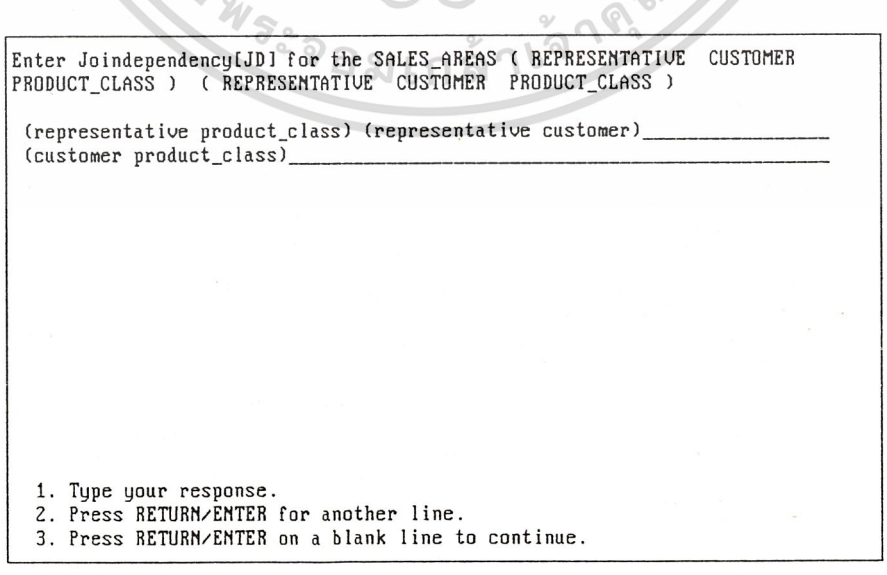
ภาพที่ 27



แสดงหน้าจอสำหรับให้ผู้ใช้เลือกความสัมพันธ์ที่มี JD/MVD อยู่

จากข้อมูล FD เบื้องต้น กำหนดให้มีความสัมพันธ์ JD แอบแฝงอยู่ในความสัมพันธ์ SALES-AREAS ดังนั้นระบบจึงให้ผู้ใช้ป้อนความสัมพันธ์ JD ดังต่อไปนี้

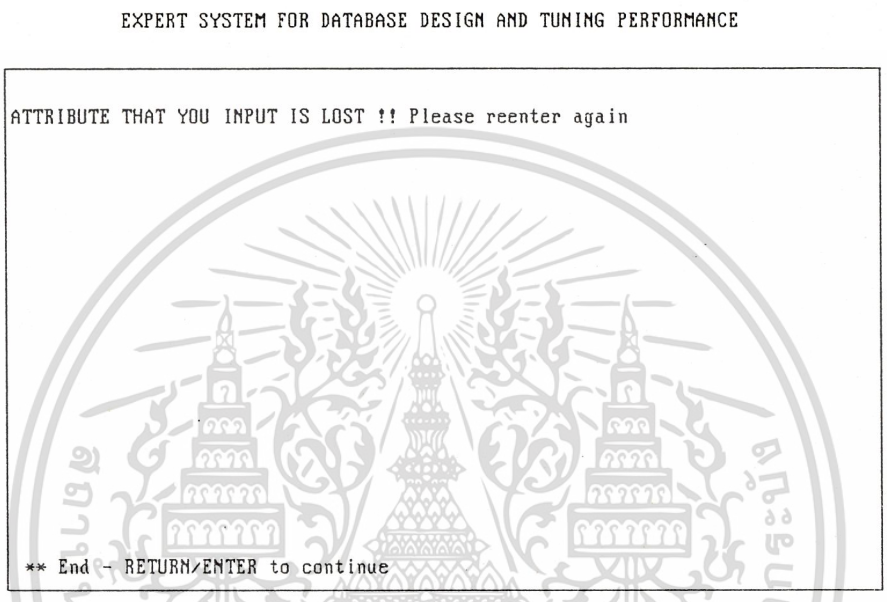
ภาพที่ 28



แสดงหน้าจอสำหรับให้ผู้ใช้ป้อนความสัมพันธ์ JD/MVD

ในการป้อนข้อมูลระบบจะมีการตรวจสอบความถูกต้องของข้อมูล JD/MVD ที่ป้อน ถ้าข้อมูลที่ป้อนมีความผิดพลาดระบบจะแสดงหน้าจอเตือนให้ผู้ใช้ป้อนข้อมูลใหม่แสดงได้ดังนี้

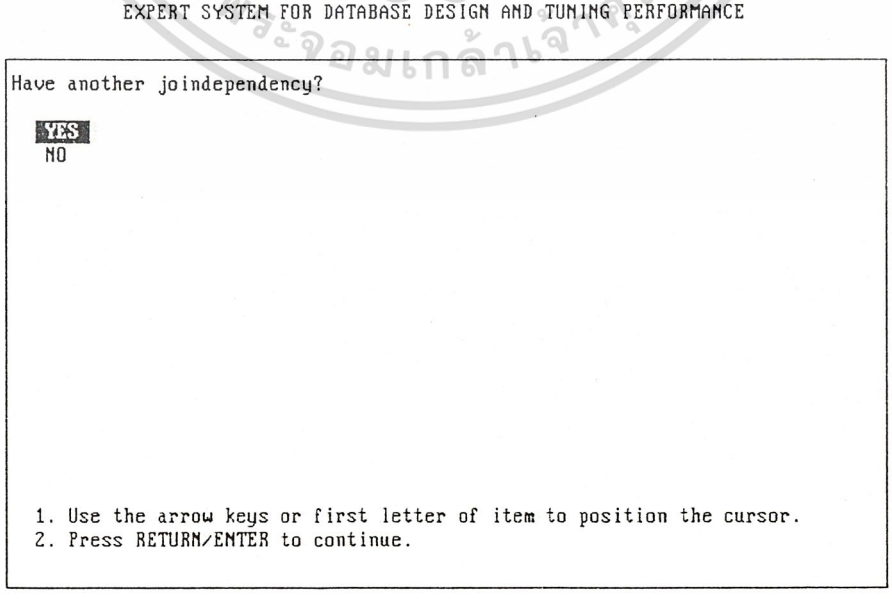
ภาพที่ 29



แสดงหน้าจอเตือนความผิดพลาดของข้อมูล JD/MVD

หลังจากการป้อนข้อมูล JD/MVD แล้ว ถ้าระบบตรวจสอบพบว่าสามารถมีข้อมูล JD/MVD แอบแฝงอยู่อีกระบบจะแสดงหน้าจอดังนี้

ภาพที่ 30



แสดงหน้าจอยืนยันการป้อนข้อมูล JD/MVD

หลังจากจบการป้อนข้อมูล JD/MVD แล้ว ระบบจะแสดงผลลัพท์ที่ได้จาการออกแบบฐานข้อมูลจากความสัมพันธ์ทั้งหมด ซึ่งเป็นข้อมูลที่อยู่ใน 5NF ดังนี้

ภาพที่ 31

```
EXPERT SYSTEM FOR DATABASE DESIGN AND TUNING PERFORMANCE

TABLE =>  SPLITRELATION_GRADES1
Primary Key => STUDENT_ID
Candidate Key =>  Not have Candidate Key
Attribute =>  STUDENT_ID PHONE_NO

TABLE =>  SPLITATTENDANCE1
Primary Key =>  SEMESTER SUBJECT
Candidate Key =>  (SEMESTER SUBJECT) (SEMESTER TEACHER)
Attribute =>  SEMESTER SUBJECT TEACHER

TABLE =>  SUPPLIERS
Primary Key =>  SUPPLIER
Candidate Key =>  Not have Candidate Key
Attribute =>  SUPPLIER CITY

TABLE =>  SPLITSUPPLIERS1
Primary Key =>  CITY
Candidate Key =>  Not have Candidate Key
Attribute =>  CITY DISTANCE

 ** More - RETURN/ENTER to continue

        EXPERT SYSTEM FOR DATABASE DESIGN AND TUNING PERFORMANCE

TABLE =>  PRLIB
Primary Key =>  P_ACCESS_NO
Candidate Key =>  Not have Candidate Key
Attribute =>  P_ACCESS_NO P_PUBLISH_PLC P_AD_YR P_BE_YR P_CALL_NO

TABLE =>  JD_SALES_AREAS1
Primary Key =>  REPRESENTATIVE PRODUCT_CLASS
Candidate Key =>  Not have Candidate Key
Attribute =>  REPRESENTATIVE PRODUCT_CLASS

TABLE =>  JD_SALES_AREAS2
Primary Key =>  REPRESENTATIVE CUSTOMER
Candidate Key =>  Not have Candidate Key
Attribute =>  REPRESENTATIVE CUSTOMER

TABLE =>  JD_SALES_AREAS3
Primary Key =>  CUSTOMER PRODUCT_CLASS
Candidate Key =>  Not have Candidate Key
Attribute =>  CUSTOMER PRODUCT_CLASS

 ** More - RETURN/ENTER to continue
```

EXPERT SYSTEM FOR DATABASE DESIGN AND TUNING PERFORMANCE

```
TABLE =>  RELATION_GRADES
Primary Key =>  STUDENT_ID COUSE
Candidate Key =>  Not have Candidate Key
Attribute =>  STUDENT_ID COUSE GRADE

TABLE =>  ATTENDANCE
Primary Key =>  SUBJECT SEMESTER SECTION
Candidate Key =>  Not have Candidate Key
Attribute =>  SUBJECT SEMESTER SECTION ATTENDANCES

TABLE =>  BORROW
Primary Key =>  STD_ID P_ACCESS_NO B_DATE
Candidate Key =>  Not have Candidate Key
Attribute =>  STD_ID P_ACCESS_NO B_DATE R_DATE

TABLE =>  PCALLN
Primary Key =>  P_CALL_NO
Candidate Key =>  Not have Candidate Key
Attribute =>  P_CALL_NO TITLE_REG1 TITLE_REG2 P_LENGTH P_PUBLISH AUTHOR_NAME
              AUTHOR_SURNAME
 ** More - RETURN/ENTER to continue
```

หน้าจอแสดงผลลัพท์ที่ได้จากการออกแบบของระบบ

จากนั้นระบบจะเก็บความสัมพันธ์ที่ได้จากการออกแบบทั้งหมดลงบนแฟ้มข้อมูล "\pcplus\data\5NF.dat" และระบบจะถามผู้ใช้ว่าต้องการปรึกษาระบบในส่วนของการปรับแต่งฐานข้อมูลหรือไม่ ซึ่งจะเป็นการนำไปสู่การทำงานในส่วนการกำหนดรูปแบบ และกลไกในการเข้าถึงข้อมูลต่อไป

<u>การใช้งานและทดสอบในส่วนของการปรับแต่งฐานข้อมูล</u>

1. **การป้อนข้อมูลสู่ระบบในส่วนการปรับแต่งฐานข้อมูล**

ข้อมูลที่ใช้งานในส่วนนี้มีทั้งหมด 3 แฟ้มข้อมูลด้วยกัน

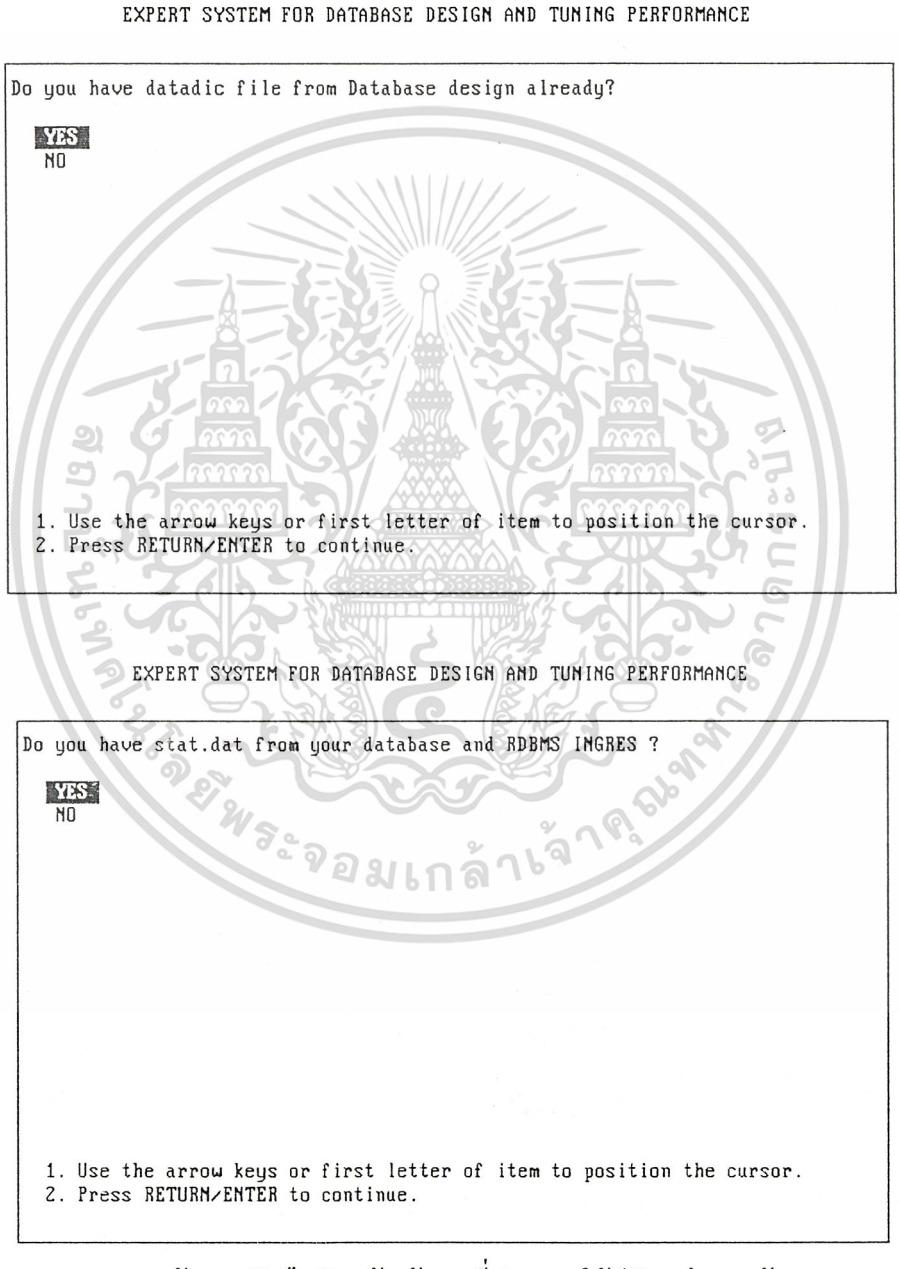
แฟ้มข้อมูล "5NF.dat" เป็นแฟ้มข้อมูลที่ได้จากขั้นตอนการออกแบบฐานข้อมูล ซึ่งมีอยู่ในซับไดเรกทอรี่ "\pcplus\data\5NF.dat "

แฟ้มข้อมูล "stat.dat" เป็นแฟ้มข้อมูลที่ได้จากการประมวลผลโปรแกรม "makestat" ซึ่งประมวลผลบนระบบปฏิบัติการ UNIX ภายใต้ระบบจัดการฐานข้อมูล INGRES ที่มีฐานข้อมูลที่ต้องการจะ

ปรับแต่งใช้งานอยู่ จากนั้นถ่ายเทแฟ้มข้อมูลสถิติที่ได้ลงมาในระบบโดยเก็บไว้ในซับไดเรกทอรี่ "\pcplus\data\stat.dat"

หลังจากการให้คำปรึกษาของระบบในส่วนการออกแบบฐานข้อมูลแล้วระบบจะขอคำยืนยันสำหรับแฟ้มข้อมูลที่ใช้ในการให้คำปรึกษาส่วนของการปรับแต่งฐานข้อมูลต่อแสดงได้ดังนี้

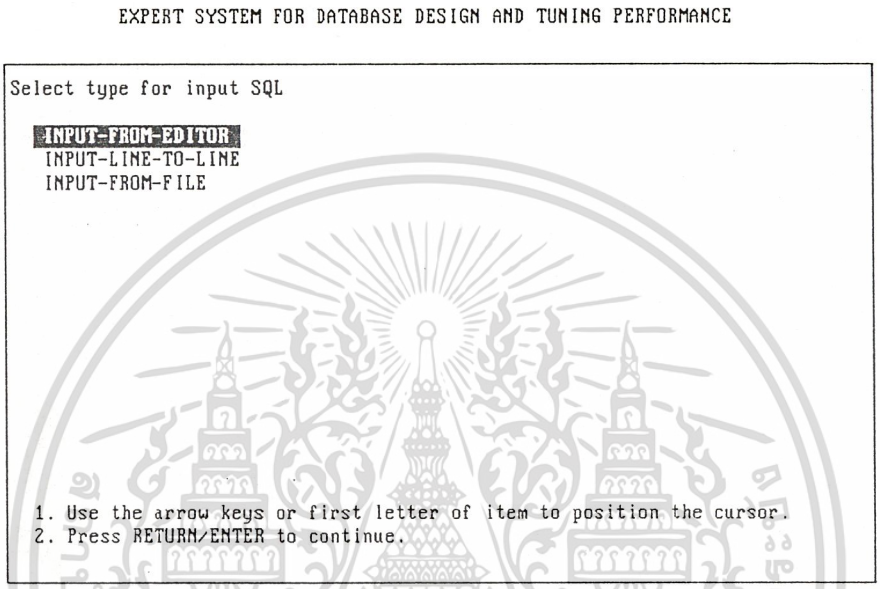
ภาพที่ 32



แสดงหน้าสำหรับยืนยันแฟ้มข้อมูลที่นำจะมาใช้ปรับแต่งฐานข้อมูล

หลังจากที่ผู้ใช้ให้คำยืนยันในส่วนของแฟ้มข้อมูลต่างๆ แล้ว ระบบจะให้ผู้ใช้เลือกรูปแบบการนำข้อมูล meta SQL เข้าสู่ระบบได้ดังนี้

ภาพที่ 33



แสดงหน้าจอสำหรับให้ผู้ใช้เลือกรูปแบบการป้อนข้อมูล SQL

แฟ้มข้อมูล “meta_sql” เป็นแฟ้มข้อมูลที่เก็บรายละเอียดของ query ที่ใช้ประจำอยู่ในระบบฐานข้อมูล ซึ่งจะกระทำได้ 3 วิธี คือ วิธีแรก ป้อนข้อมูลโดยใช้ Text editor ทั่วไป ซึ่งจะกระทำก่อนการใช้งานระบบ วิธีที่สอง ป้อนข้อมูลโดยใช้ Editor ของระบบ ซึ่งจะกระทำในระหว่างการให้คำปรึกษาของระบบ วิธีที่สาม ป้อนข้อมูลโดยใช้ Editor แบบ line to line ของระบบ ซึ่งจะกระทำในระหว่างการให้คำปรึกษาของระบบเช่นกัน

Query ที่นำมาใช้ในการทดสอบแสดงได้ดังนี้

```
select std_id,title_reg2
from pcalln a,prlib b,borrow c
where a.p_call_no = b.p_call_no and
b.p_access_no = c.p_access_no and
b.p_publish_plc = 'NEW YORK' and
c.b_date = '30-JUN-95' and r_date = '07-JUL-95' and
p_publish = 'MCGRAW-HILL'
```

priority 8

select p_call_no , title_reg2

from pcalln

order by p_call_no

priority 5

select std_id,b_date

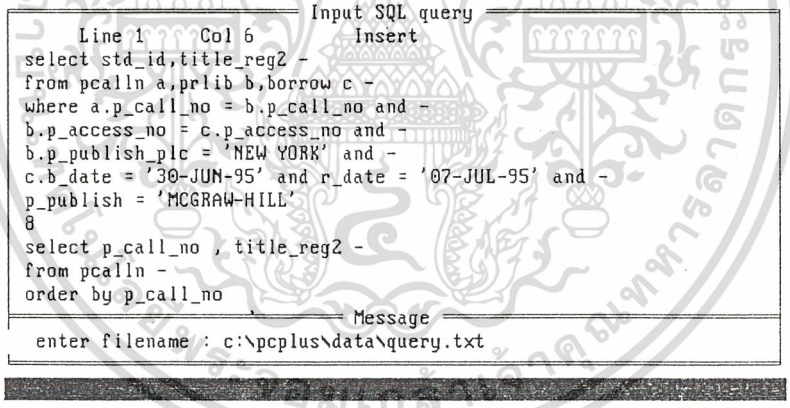
from borrow

order by std_id

priority 5

ตัวอย่างของรูปแบบการป้อน query จาก editor ของระบบแสดงได้ดังนี้

ภาพที่ 34



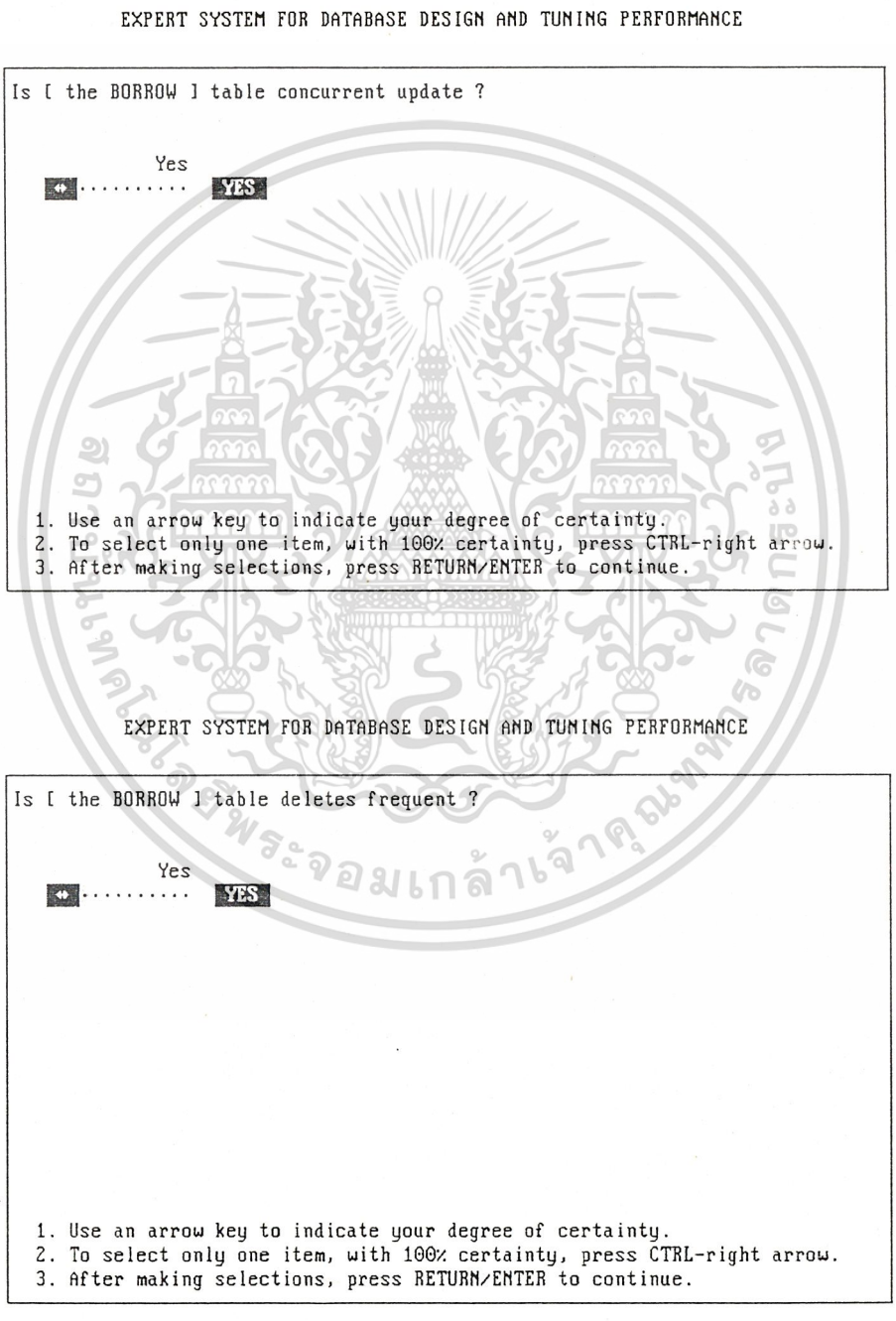
แสดงหน้าจอ การป้อน query ผ่านทาง editor ของระบบ

ในการป้อนข้อมูล query ใช้ ";" หรือ "-" ปิดท้ายในกรณีที่ยังไม่จบ query ถ้าจบ query เคาะ enter โดยไม่ต้องใส่ ";" หรือ "-" ในส่วนท้ายของแต่ละ query จะต้องใส่ค่า priority ของแต่ละ query โดยกำหนดให้เป็นตัวเลขตั้งแต่ 1 - 10 ในกรณีของ line-to-line editor ถ้าเคาะ enter 2 ครั้งโดยไม่ใส่ ";" หรือ "-" ถือว่าเป็นการจบการป้อนข้อมูล query

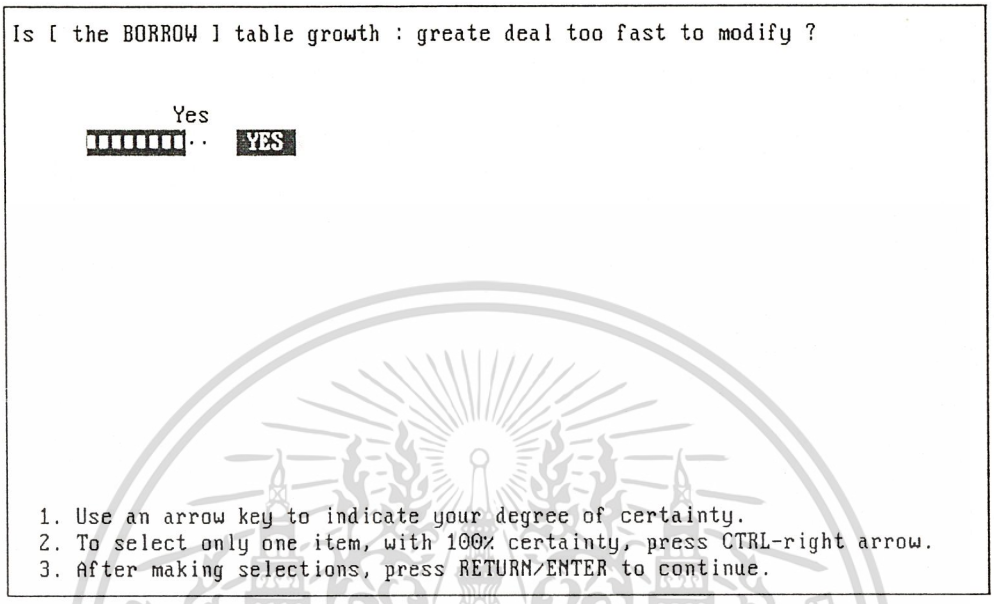
หลังจากป้อน query เสร็จแล้ว ระบบจะถามคำถามที่เกี่ยวกับการใช้งานตารางข้อมูลที่เกี่ยวข้องกับ query โดยคำถามจะเป็นในลักษณะของ yes/no และผู้ใช้สามารถระบุค่าความแน่นอนของ

ความจริงที่ตอบด้วย ซึ่งจะกำหนดในรูปของ certainty factor ตัวอย่างคำถามการใช้งานตาราง BORROW แสดงได้ดังนี้

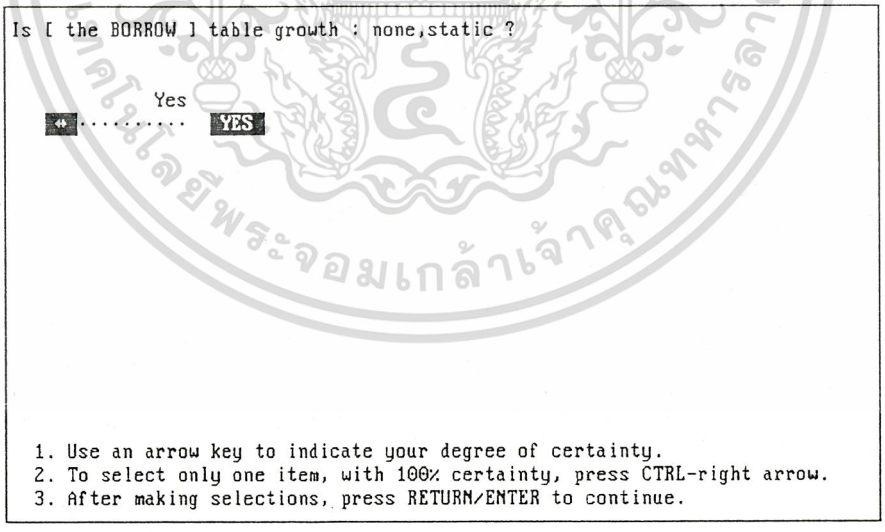
ภาพที่ 35



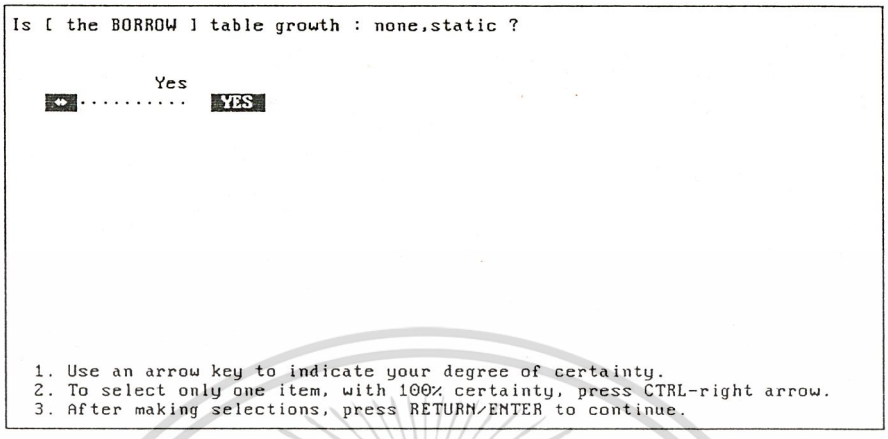
EXPERT SYSTEM FOR DATABASE DESIGN AND TUNING PERFORMANCE



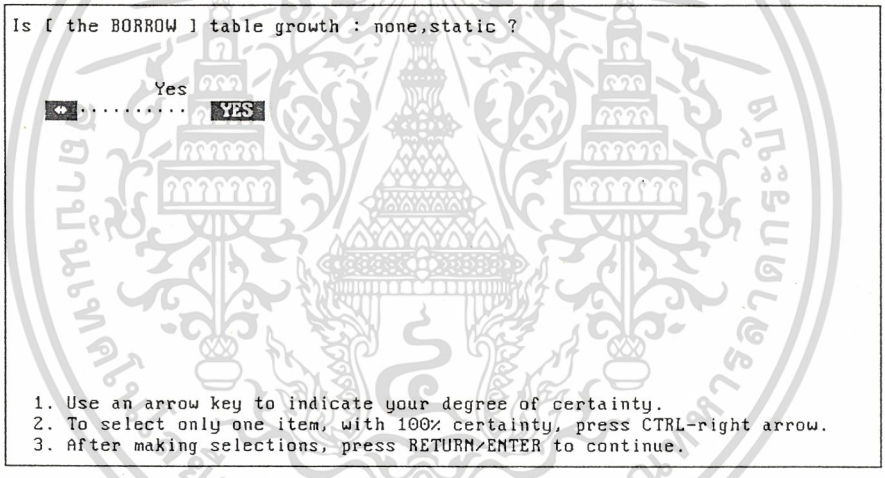
EXPERT SYSTEM FOR DATABASE DESIGN AND TUNING PERFORMANCE



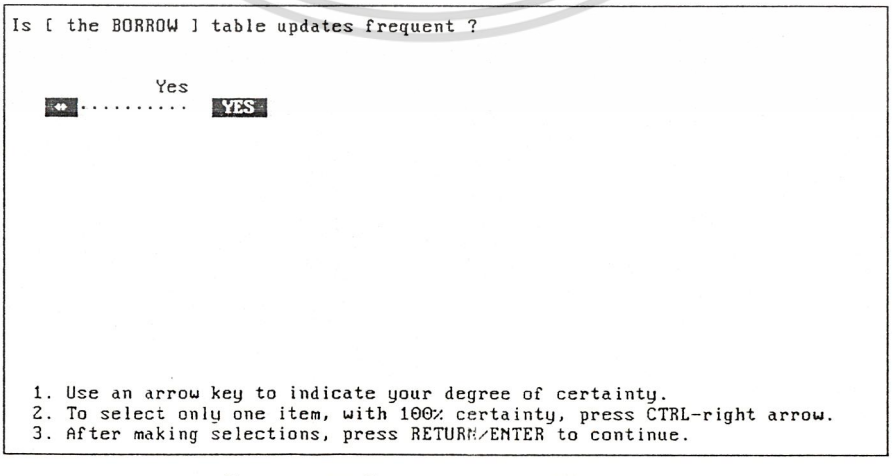
EXPERT SYSTEM FOR DATABASE DESIGN AND TUNING PERFORMANCE



EXPERT SYSTEM FOR DATABASE DESIGN AND TUNING PERFORMANCE



EXPERT SYSTEM FOR DATABASE DESIGN AND TUNING PERFORMANCE



แสดงหน้าจอสำหรับป้อนลักษณะการใช้งานตารางข้อมูล

จากนั้นระบบจะทำการสรุปและเสนอแนะการกำหนดรูปแบบ และกลไกในการเข้าถึงข้อมูล พร้อมกับบันทึกผลที่ได้ลงบนแฟ้มข้อมูล "\pcplus\data\primary.dat" และ "\pcplus\data\secondary.dat" ซึ่งแสดงได้ดังนี้

ภาพที่ 36

EXPERT SYSTEM FOR DATABASE DESIGN AND TUNING PERFORMANCE

```
PRIMARY INDEX
PCALLN  :  P_CALL_NO   :  BTREE
PRLIB   :  P_ACCESS_NO  :  HASH
BORROW  :  STD_ID   :  BTREE

SECONDARY INDEX
PCALLN  :  P_CALL_NO  :  HASH
BORROW    R_DATE B_DATE   HASH




** End - RETURN/ENTER to continue
```

หน้าจอแสดงผลลัพท์ที่ได้จากการปรับแต่งของระบบ

จากทั้งหมดที่กล่าวมาเป็นการทดสอบความถูกต้องของระบบ EX/DT2 พร้อมทั้งอธิบายการใช้งานระบบในลักษณะของผู้ใช้ทั่วไป สำหรับการใช้งานในส่วนของผู้ใช้ที่เป็นวิศวกรความรู้ ซึ่งจะใช้งานในลักษณะของการเพิ่มเติม หรือลบความรู้ในฐานความรู้ จะศึกษาได้จากในส่วนภาคผนวก ก ซึ่งจะกล่าวถึงสรุปการใช้งานโปรแกรมเปลือกระบบผู้เชี่ยวชาญ PC/PLUS เบื้องต้น แต่ถ้าต้องการรายละเอียดเพิ่มเติมสามารถศึกษาได้จากคู่มือการใช้งาน PC/PLUS

บทที่ 10

สรุปผลและแนวทางการวิจัย

สรุปผลงานวิจัย

จากที่ได้กล่าวมาในตอนต้นของวิทยานิพนธ์ ถึงเรื่องทฤษฎีของการออกแบบฐานข้อมูล และการปรับแต่งฐานข้อมูลในรูป การเลือกรูปแบบ และกลไกในการเข้าถึงข้อมูล จะเห็นว่าเป็นทฤษฎีที่ยุ่งยากและซับซ้อน จะต้องใช้ผู้เชี่ยวชาญเฉพาะด้านเท่านั้น จึงจะทำให้ได้ผลลัพท์ออกมาถูกต้องและเหมาะสมกับความต้องการใช้งาน โดยเฉพาะการเลือกรูปแบบ และกลไกในการเข้าถึงข้อมูล ซึ่งนอกจากจะต้องเข้าใจถึงทฤษฎีพื้นฐานทางด้านนี้แล้ว ยังต้องมีความเข้าใจการทำงานของระบบจัดการฐานข้อมูลที่ใช้อยู่ด้วย เพราะจะต้องนำทฤษฎีพื้นฐานมาวิเคราะห์ร่วมกับทำงานของระบบจัดการฐานข้อมูลที่ใช้อยู่ จึงจะทำให้การเลือกรูปแบบ และกลไกในการเข้าถึงข้อมูลเป็นไปอย่างถูกต้อง และเหมาะสมกับความต้องการใช้งานฐานข้อมูลนั้น จะเห็นว่า EX/DT2 เข้ามาช่วยแก้ปัญหานี้ได้ ทำให้ผู้ที่มีความรู้แค่เพียงพื้นฐานทางด้านนี้ ไม่ค่อยมีประสบการณ์ทางด้านนี้ สามารถใช้ระบบนี้(EX/DT2) เข้ามาช่วยแก้ปัญหา ซึ่งผลลัพท์ที่ได้จากระบบนี้มีค่าเท่ากับการกระทำโดยผู้เชี่ยวชาญ ทำให้แก้ปัญหาขาดแคลนผู้เชี่ยวชาญทางด้านนี้ได้

ปัญหาที่เกิดขึ้น

ปัญหาแรกที่เกิดขึ้นในการพัฒนาระบบนี้คือ ปัญหาเกี่ยวกับการกำหนดทฤษฎีในการเลือกรูปแบบ และกลไกในการเข้าถึงข้อมูล เนื่องจากระบบจัดการฐานข้อมูลรีเลชั่นแนลที่มีใช้กันอยู่ในปัจจุบันมีมากมายหลายชนิด ดังนั้นการที่จะกำหนดทฤษฎีให้ครอบคลุมทั่วถึงทุกชนิดเป็นไปได้ยาก เพราะแต่ละชนิดก็จะมีเทคนิคในการพัฒนา และการใช้งานที่แตกต่างกัน จะมีก็แต่ทฤษฎีพื้นฐาน

บางส่วนเท่านั้นที่เหมือนกัน ซึ่งถ้าจะนำเฉพาะทฤษฎีพื้นฐานบางส่วนที่เหมือนกันมาสร้างเป็นฐานข้อมูลความรู้ จะทำให้ผลลัพท์ที่ได้จากการเลือกรูปแบบ และกลไกในการเข้าถึงข้อมูลได้ไม่ดีเท่าที่ควร ดังนั้นจึงต้องนำทฤษฎีพื้นฐานมาวิเคราะห์ร่วมกับการทำงานของระบบจัดการฐานข้อมูลที่ใช้อยู่ ซึ่งในงานวิจัยนี้ใช้ระบบจัดการฐานข้อมูล INGRES จากนั้นก็สรุปเป็นกฎเกณฑ์ออกมาเพื่อนำไปใส่ในฐานข้อมูลความรู้ต่อไป

ปัญหาที่สองคือการพัฒนาระบบ ระบบนี้ใช้เปลือกระบบผู้เชี่ยวชาญ PC/PLUS ในการพัฒนา เนื่องจากการพัฒนาในบางส่วนมีโครงสร้าง และการประมวลผลที่ซับซ้อนจึงจำเป็นต้องใช้ภาษา LISP เข้ามาช่วยในการพัฒนาในส่วนที่ใกล้เคียงกับ PC/PLUS และสำหรับหน่วยติดต่อกับผู้ใช้ในส่วนรับข้อมูลภาษา SQL ลงมาเก็บบนแฟ้มข้อมูล METASQL ซึ่งเป็นส่วนที่ห่างไกลจาก PC/PLUS ใช้ภาษา C ในการพัฒนา

ปัญหาสุดท้ายคือความเร็วในการประมวลผล ทั้งนี้เนื่องจากระบบนี้มีการติดต่อ และเรียกค้นข้อมูลจากแฟ้มข้อมูลจำนวนวนมาก ซึ่ง PC/PLUS ไม่เอื้ออำนวยทางด้านนี้ จึงต้องใช้เวลาในการประมวลผลนานพอสมควร

<u>แนวทางในการประยุกต์ใช้</u>

EX/DT2 นี้ถูกสร้างขึ้นมาในลักษณะของระบบผู้เชี่ยวชาญ จึงทำให้มีความคล่องตัวในการที่จะนำไปประยุกต์ใช้งานกับระบบจัดการฐานข้อมูลอื่น ๆ ที่ไม่ใช่ INGRES เพราะการทำงานของระบบผู้เชี่ยวชาญ จะแยกส่วนของกลไกการวินิจฉัย กับส่วนของฐานข้อมูลความรู้ออกจากกัน ดังนั้นเพียงแต่เรานำความรู้เกี่ยวกับการเลือกรูปแบบ และกลไกในการเข้าถึงข้อมูลบนระบบจัดการฐานข้อมูลอื่นมาใส่ในฐานข้อมูลความรู้ ก็สามารถนำ EX/DT2 ไปใช้กับระบบจัดการฐานข้อมูลนั้นได้ แต่การที่จะนำความรู้ลงไปเก็บในฐานข้อมูลความรู้ได้นั้น จะต้องมีความรู้และความเข้าใจในการใช้เปลือกระบบผู้เชี่ยวชาญ PC/PLUS พอสมควร ซึ่งรายละเอียดเกี่ยวกับการใช้งาน PC/PLUS มีกล่าวไว้ในภาคผนวก ก แต่สำหรับในส่วนของการออกแบบฐานข้อมูลนั้น ความรู้ที่ใช้ในการออกแบบเป็นทฤษฎีที่มาตราฐาน สามารถนำไปใช้ได้กับระบบจัดการฐานข้อมูลรีเลชั่นแนลทุกระบบ

แนวทางในการพัฒนาต่อ

แนวทางในการพัฒนาต่อแบ่งได้เป็นสองส่วน

ส่วนแรก แนวทางในการพัฒนาต่อในส่วนของการออกแบบฐานข้อมูล สิ่งที่น่าสนใจที่สามารถนำไปพัฒนาต่อในส่วนนี้คือการป้อนข้อมูล Function dependency ซึ่งในส่วนนี้ผู้ใช้ระบบต้องมีความรู้เรื่องฐานข้อมูลพอสมควร แต่ถ้าเราสร้างระบบที่สามารถกำหนด Function dependency ได้จากการป้อนตัวอย่างความสัมพันธ์ของข้อมูลได้ ก็จะทำให้ใช้งานได้สดวกยิ่งขึ้น

ส่วนที่สอง แนวทางในการพัฒนาต่อในส่วนการปรับแต่งฐานข้อมูล งานวิจัยชิ้นนี้จะกระทำในรูปของการเลือกรูปแบบ และกลไกในการเข้าถึงข้อมูล ซึ่งจะเน้นถึงการใช้ I/O และ CPU time น้อยที่สุด ซึ่งการปรับแต่งฐานข้อมูลยังสามารถกระทำในด้านอื่นได้อีก ซึ่งถ้าสามารถนำมาเพิ่มเติมความรู้ให้กับระบบนี้ก็ จะทำให้การปรับแต่งฐานข้อมูลเป็นไปอย่างมีประสิทธิภาพยิ่งขึ้น

บรรณานุกรม

[1] Michael A. Carrico , John E. Girard , Jennifer P. Jones : "Buitding Knowledge Systems " McGra-Hill Book Comany , 1989

[2] Elaine Rich , Kevin Knight : "Artificial Intelligence" Second Edition , McGraw-Hill ,1991

[3] D.S.W.Tansley , C.C.Hayball : "Knowledge-Based System Analysis and Design" , Pretice Hall , 1993

[4] John K. Debenham : "Knowledge System Design" , Pretice Hall , 1989

[5] Adrian A. Hopgood , : "Knowledge-Based Systems" , CRC Press , 1993

[6] Tore Amble , : "Logic Programming and Knowledge Engineering" , Addison-Wesley , 1987

[7] Randall Davis , : "Knowledge-Based Systems in Artificial Intelligence" , McGraw-Hill , 1982

[8] Michael L. Brodie , John Mylopoulos : "On Knowledge Based Management Systems" , Springer-Verlag Ne York ,1986

[9] Sholom M. Weiss , Casimir A. Kulikowski : "A Practical Guide to Designing Expert Systems" , Roman & Allanheld Publishers , 1984

[10] M.J. Coombs : "Developments in expert systems" , Academic press , 1980

[11] I.T.Hawryszkiewycz : "Relational Database Design" , Prentice Hall , 1990

[12] C.J.Date : "An Introduction to Database Systems" , Addison-Wesley , 1986

[13] Candace C.Fleming , Barbara von Halle : "Handbook of Relational Database Design" , Addison-Wesley , 1989

[14] Wolfang Kreutzer & Bbrce McKenzie : "Programming for Artificial Intelligence" , Addison-Wesley ,1990

[15] Robert Keller : "Expert System Technology" , Prentice-Hall , 1987

[16] Ralph B. Bisland : "Database Management" , Prentice-Hall , 1989

[17] C.J.Date : "A Guide to The SQL Standard" , Addison-Wesley ,1987

[18] "Ingres Database Administrator's Guide (for the UNIX Operating System)" ,1991

บรรณานุกรม (ต่อ)

[19] "Personal Consultant Plus Reference Guide" , Texas Instruments , 1987

[20] Henry F.Korth Abraham Silberschatz : "Database System Concepts" , McGraw-Hill , 1991

[21] วิลาศ วูวงศ์ , บุญเจริญ ศิรินวกุล : "Expert System" , NECTEC , 2535

[22] Peter Corrigan , Mark Gurry : "Oracle Performance Tuning" , O'Reilly & Associates , 1993

[23] John Durkin : "Expert Systems DESIGN AND DEVELOPMENT", Macmillan , 1994

[24] Harry Frank , Steven C. Althoen ; "STATISTICS Concepts and Applications" , Cambridge , 1994

ภาคผนวก

ภาคผนวก ก

โปรแกรมเปลือกระบบผู้เชี่ยวชาญ

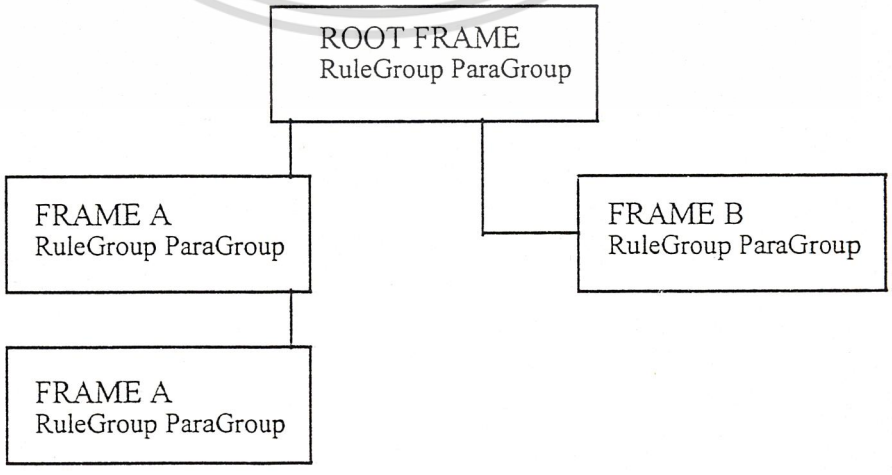
Personal Consultant Plus (PC/PLUS)

PC/PLUS เป็นเปลือกระบบผู้เชี่ยวชาญที่ผลิตโดยบริษัท Texas Instrument ระบบที่ผลิตขึ้นนี้สำหรับใช้งานกับเครื่องคอมพิวเตอร์ TI และ IBM PC XT และ AT โดยตัวระบบเปลือกระบบผู้เชี่ยวชาญนี้จะทำงานภายใต้ PC Scheme (เป็นส่วนหนึ่งของ Common LISP) เนื่องจากเปลือกระบบผู้เชี่ยวชาญนี้สร้างขึ้นมาด้วยภาษา LISP ดังนั้น สำหรับผู้พัฒนาระบบที่สามารถเขียนภาษา LISP ได้สามารถใช้ภาษา LISP สำหรับสร้างฟังก์ชันที่จะช่วยเพิ่มคุณประสิทธิภาพให้กับระบบได้

รูปแบบการแสดงความรู้ของ PC/PLUS

PC/PLUS ใช้เฟรมเป็นตัวแสดงความรู้ โดยภายในเฟรมจะรวบรวมข้อมูลที่ใช้ในการแก้ปัญหา และข้อมูลที่ใช้ในการกำหนดโครงสร้าง และการดำเนินงาน จะต้องมีการกำหนดเป้าหมายของการแก้ปัญหา (goal) ไว้ในแต่ละเฟรม ซึ่งในหนึ่งเฟรมสามารถกำหนดได้หลายเป้าหมาย องค์ประกอบหลักสำคัญของเฟรมประกอบด้วย กลุ่มของพารามิเตอร์ (parameter group) และกลุ่มของกฎที่ใช้ในการแก้ปัญหา (rule group) การที่จะนำความรู้เก็บในฐานความรู้ได้นั้น จะต้องสร้างเป็นเฟรมของความรู้ขึ้นมาก่อน โดยมีการกำหนดเป้าหมาย และกฎสำหรับแก้ปัญหาไว้ภายในเฟรม ในกรณีที่ปัญหามีขนาดใหญ่ และมีความซับซ้อนมาก ก็จำเป็นต้องสร้างเฟรมขึ้นมาหลายเฟรม แต่ละเฟรมจะเชื่อมต่อกันในลักษณะของ tree ซึ่งจะประกอบด้วยรูทเฟรม(root frame) และซับเฟรม(subframe) โดยที่เป้าหมายของการหาเหตุผลขั้นสุดท้ายนั้นจะต้องถูกกำหนดไว้ที่รูทเฟรมดังแสดงในรูป

ภาพที่ 37

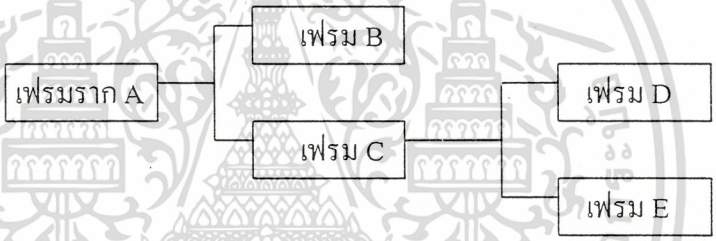


แสดงโครงสร้างของเฟรมใน PC/PLUS

จากที่มีการเชื่อมต่อกันของเฟรมทำให้เกิดคุณสมบัติสำคัญ การสืบทอด (Inheritance) และขอบเขตของการสืบทอด

คือเราจะมองสองเฟรมใด ๆ ที่เชื่อมกันในลักษณะของเฟรมพ่อ (parent frame) และเฟรมลูก (child frame) โดยที่เฟรมลูกจะเกิดจากการสร้างซับเฟรมต่อจากเฟรมพ่อ ซึ่งจะทำให้เกิดคุณสมบัติต่อไปนี้ เฟรมลูกสามารถใช้พารามิเตอร์ของเฟรมพ่อได้ แต่เฟรมพ่อไม่สามารถแอกเซสไปยังพารามิเตอร์ของเฟรมลูกได้ แต่ในทางตรงกันข้าม เฟรมพ่อสามารถแอกเซสกฎที่อยู่ในเฟรมลูกที่สืบลงมาได้ (descendant frame) แต่ไม่สามารถแอกเซสกฎที่อยู่ในเฟรมที่อยู่เหนือกว่าได้ (ancestor frames) เพื่อความเข้าใจ พิจารณาได้จากรูปต่อไปนี้

ภาพที่ 38



แสดงภาพการจัดลำดับเฟรม

จากรูปพบว่า A เป็นเฟรมราก .จึงไม่มีพ่อ A มีลูกสองเฟรม คือ B และ C ส่วนเฟรม D และ E เป็นเฟรมลูกของ C เฟรม B ไม่มีเฟรมลูก จากขอบเขตดังที่ได้อธิบายไปแล้วข้างต้นจะพบว่า

1. เฟรม A สามารถแอกเซสพารามิเตอร์ในเฟรม A ได้ และสามารถอ้างถึงกฎในเฟรม A , B , C , D และ E ได้

2. เฟรม B สามารถแอกเซสพารามิเตอร์ในเฟรม A และเฟรม B ได้ และสามารถอ้างถึงกฎในเฟรม B ได้

3. เฟรม C สามารถแอกเซสพารามิเตอร์ในเฟรม A และC ได้ และสามารถอ้างถึงกฎในเฟรม C , D และ E ได้

4. เฟรม D สามารถแอกเซสพารามิเตอร์ในเฟรม D , C และ A ได้ และสามารถอ้างถึงกฎในเฟรม D ได้

5. เฟรม E สามารถแอกเซสพารามิเตอร์ในเฟรม E , C และ A ได้ และสามารถอ้างถึงกฎในเฟรม E ได้

การอ้างถึงเฟรมหรือการเชื่อมต่อเฟรม (Instantiation)

PC/PLUS ไม่ได้สร้างกระบวนการอ้างถึงเฟรมย่อยเอาไว้อย่างอัตโนมัติ ดังนั้นในระหว่างที่ระบบให้คำปรึกษากับผู้ใช้งาน เราต้องการให้มีการเชื่อมต่อระหว่างเฟรมพ่อกับเฟรมลูก จำเป็นต้องทำการสร้างกระบวนการที่ใช้ในการอ้างถึง และควบคุมการอ้างถึงในฐานความรู้เองซึ่งสามารถทำได้ดังนี้

- Consider frame

ในกรณีนี้เราสามารถสั่งให้ทำการเชื่อมระหว่างเฟรมพ่อ กับเฟรมลูกได้ตามความต้องการ

- Backward chaning แบ่งได้เป็น 2 กรณี

ใช้ตัวแปรเป้าประสงค์ (goal parameter) กลไกในการวินิจฉัยจะพยายามหาค่าเป้าประสงค์ที่เราต้องการ ถ้ามีการกำหนดค่าให้กับตัวแปรเป้าประสงค์ในส่วนของกรอบความรู้ลูกก็จะถูกเชื่อมต่อเข้ามา

เช็คตัวแปร (parameter) เพื่อให้ได้ค่าตัวแปรเป้าประสงค์ที่ต้องการในบางครั้งจะต้องมีการกำหนดค่าให้กับตัวแปรอื่นก่อน ซึ่งถ้ามีการกำหนดค่าให้กับตัวแปรตัวนั้นในกรอบความรู้ลูกก็จะทำการเชื่อมเข้ามา

<u>การอนุมาน</u>

ระบบ PC/PLUS เป็นเปลือกระบบผู้เชี่ยวชาญที่สามารถกำหนดค่าของความมั่นใจ (certainty factor) ได้ด้วยการใส่ค่าลงในส่วนของกฎ และความจริง ค่าของความมั่นใจในระบบนี้จะต้องกำหนดเป็นตัวเลขระหว่าง 100 (มั่นใจมาก) ถึง - 100 (ไม่เห็นด้วย)

สำหรับอนุมานของระบบ สามารถทำได้ทั้งแบบเดินหน้า หรือย้อนหลัง นอกจากนี้แล้วระบบยังมีวิธีการที่เรียกว่า "Access Methods" เพื่อให้ผู้พัฒนาระบบสามารถเรียกการทำงานในส่วน ACTION ของกฎ โดยอาศัยการเข้าสู่พารามิเตอร์ หรือกฎก็ได้ ซึ่งวิธีการเช่นนี้จะเป็นเครื่องมือที่มีประสิทธิภาพสำหรับผู้พัฒนาระบบในการกำหนดการอนุมานแบบเดินหน้า

การติดต่อกับผู้พัฒนาระบบ

ในการสร้างฐานความรู้ให้กับระบบผู้เชี่ยวชาญนี้ทำได้ไม่ยากนัก เนื่องจากว่าระบบมีระบบการจัดการกฎที่ค่อนข้างดี ผู้พัฒนาสามารถทำตามขั้นตอนที่ได้มาจากเมนู ส่วนของระบบจัดการกฎจะมีเมนูที่กำหนดลำดับขั้นตอนของการสร้างฐานความรู้ เพื่อให้ผู้พัฒนาได้ทำตามทีละขั้น สำหรับรายละเอียดของการทำงาน และวิธีการใช้งานของระบบการจัดการกฎ สามารถศึกษาเพิ่มเติมจากคู่มือการใช้งาน PC/PLUS

ในส่วนของการสร้างฐานความรู้นั้น PC/PLUS ยังมีความสามารถเกี่ยวกับการติดต่อกับภายนอกได้ โดยทั่วไปแล้ว ระบบนี้จะติดต่อกับโปรแกรมภายนอกด้วยการใช้ LISP เป็นตัวกลาง แต่ในกรณีที่ผู้พัฒนาไม่คุ้นเคยกับภาษานี้ ก็สามารถทำได้ด้วยการให้ระบบติดต่อกับ DOS ได้โดยตรง และใช้โปรแกรมอื่นกำหนดการทำงาน ซึ่งการทำเช่นนี้ทำให้ระบบสามารถติดต่อกับโปรแกรมภายนอกได้อย่างไม่มีขอบเขตจำกัด

การติดต่อกับผู้ใช้

ในระหว่างการให้คำปรึกษาของระบบกับผู้ใช้ระบบ จะมีการถามตอบกับผู้ใช้ระบบ ในการตอบคำถามของผู้ใช้สามารถทำได้โดยการพิมพ์คำตอบเข้าทางคีย์บอร์ด หรือเลือกจากรายการของคำตอบที่ระบบให้มา และในการให้คำตอบนี้ผู้ใช้สามารถให้คำตอบที่มากกว่าหนึ่งคำตอบได้ และเช่นกันการให้คำตอบของผู้ใช้ก็สามารถกำหนดค่าของความมั่นใจได้ด้วย สำหรับการให้ค่าของความมั่นใจนี้จะมีขอบเขตอยู่เฉพาะที่ 0 (ไม่ทราบ) ถึง 100 (มั่นใจมาก)

ระหว่างทางของการขอคำปรึกษาของผู้ใช้ ระบบยังอนุญาตให้ผู้ใช้ถามคำถามระหว่างทางได้ด้วย คำถามเหล่านี้คือ “How” และ “How” หมายถึง การถามระบบว่าข้อสรุปนี้ได้มาอย่างไรและ “Why” หมายถึง ทำไมถึงถามคำถามนี้ เมื่อระบบได้รับคำถามดังกล่าว ก็จะให้คำตอบเกี่ยวกับเรื่องนั้น ๆ ออกมา

กฎ (RULE)

กฎของ PC/PLUS จะแสดงในรูปของ if-then ซึ่งจะเป็นตัวแสดงความสัมพันธ์ระหว่างพารามิเตอร์ รายละเอียดของพารามิเตอร์มีกล่าวในหัวข้อถัดไป

ตัวอย่างของการแสดงกฎในรูปของ if-then ซึ่งจะแสดงค่าที่เป็นไปได้ของพารามิเตอร์ capital มีเงื่อนไขดังนี้

If country is Thailand, then the capital must be Bangkok.

จากข้อความข้างต้นสามารถแปลงให้อยู่ในรูปกฎของ PC/PLUS ได้ดังนี้

IF statement : COUNTRY = THAILAND

Then statement : CAPITAL = BANGKOK

และเช่นกันถ้ามีเงื่อนไขอื่นที่แสดงค่าที่เป็นไปได้ของ capital

If country is England, then the capital must be London.

จากข้อความข้างต้นสามารถแปลงให้อยู่ในรูปกฎของ PC/PLUS ได้ดังนี้

IF statement : COUNTRY = ENGLAND

Then statement : CAPITAL = LONDON

นอกจากนั้นภายในกฎยังสามารถเขียนเป็นภาษา Lisp ได้แต่ต้องมีรูปแบบเงื่อนไขตามที่กำหนดคือ อยู่ภายในวงเล็บ และมีตัว E นำหน้า เช่น (E (set! list1 (append list1 list2)))

หลักการทำงานคือ PC/PLUS จะพยายามประมวลผล rule โดยจะพิจารณาจากเงื่อนไขในส่วนของ IF เมื่อเงื่อนไขใน IF ถูกค้นพบ และมีค่าเป็นจริง rule นั้นจะมีสถานะเป็น pass ก็จะประมวลผลในส่วนของ THEN แต่ถ้าเงื่อนไขใน IF มีค่าเป็นเท็จ rule นั้นจะมีสถานะเป็น fail

พารามิเตอร์ (Parameters)

พารามิเตอร์ เป็นองค์ประกอบอย่างหนึ่งของ PC/PLUS ซึ่งคุณสมบัติต่างๆจะถูกนำมาใช้ร่วมกับการตัดสินใจของการทำงานที่มีอยู่ตลอดเวลา โดยพารามิเตอร์จะเป็นโครงสร้างที่ประกอบไปด้วยคุณสมบัติต่างๆ ซึ่ง PC/PLUS จะใช้คุณสมบัติเหล่านี้ไปตลอดจนการให้คำปรึกษาของระบบ เราเปรียบเทียบได้ว่าพารามิเตอร์เปรียบเสมือนตัวแปรชนิดหนึ่งที่ใช้ในการเขียนโปรแกรม แต่พารามิเตอร์เป็นตัวแปรที่ฉลาดกว่า เพราะการอนุมานกฎจะใช้พารามิเตอร์เข้าไปเกี่ยวข้องด้วย

ชื่อของพารามิเตอร์ ถ้ามีมากกว่า 1 คำ ก็จะใช้เครื่องหมายยัติภังค์ "-" หรือ "_" เชื่อมคำเหล่านั้นเข้าด้วยกันเป็นคำเดียว เช่น INTERATE_RATE หรือ ENGINE_SIZE โดยชื่อต่างๆ ถ้าเป็นภาษาอังกฤษก็จะเก็บเป็นตัวพิมพ์ใหญ่ทั้งหมด แม้ว่าขณะพิมพ์ชื่อจะเป็นตัวพิมพ์เล็กก็ตาม แต่เมื่อนำเข้าสู่ระบบแล้วระบบจะทำการเปลี่ยนให้เอง

พารามิเตอร์ประกอบไปด้วยคุณสมบัติต่างๆ ดังต่อไปนี้ คือ

1. TYPE

TYPE เป็นการบอกชนิดของพารามิเตอร์ โดยจะมีผลต่อการดำเนินงานของพารามิเตอร์และหน้าจอ และเมื่อมีส่วน TYPE ก็จะต้องมีส่วน PROMPT อยู่ด้วยเสมอ โดย TYPE แบ่งออกเป็น 4 ชนิดดังนี้ คือ

- ASK-ALL

พารามิเตอร์ที่กำหนดชนิดเป็น ASK-ALL จะสามารถมีค่าที่ปรากฏเป็นค่าความแน่นอนได้หลายค่า โดยจะต้องมีองค์ประกอบที่เป็น PROMPT และ EXPECT ด้วยทุกครั้ง ซึ่ง EXPECT จะทำหน้าที่ในการเก็บค่าต่างๆที่ต้องการมีไว้เพื่อให้ผู้ใช้ได้เลือกในการใช้งาน

การทำงานจะสามารถมี CERTAINTY-FACTOR-RANGE ซึ่งจะเก็บค่าได้ต่างๆกัน อาจเป็น FULL หรือ POSITIVE ซึ่งรายละเอียดของ CERTAINTY-FACTOR-RANGE จะมีกล่าวในตอนหลังของบทนี้

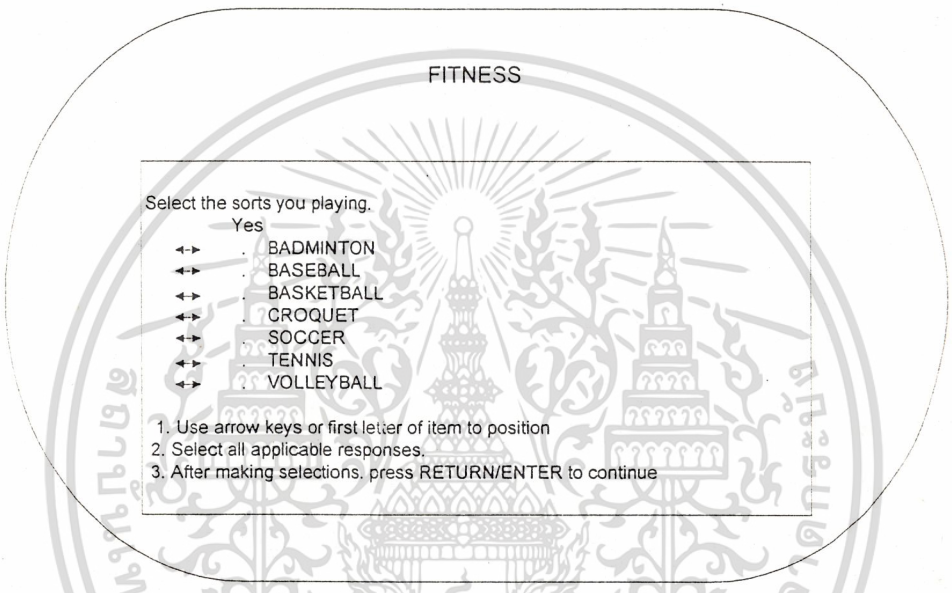
ตัวอย่าง

ในระบบผู้เชี่ยวชาญสำหรับช่วยในสถานบริหารร่างกาย ASK-ALL เป็นพารามิเตอร์สำหรับกำหนดกีฬาที่ผู้ใช้ระบบสนใจ ซึ่งมีค่า EXPECT ดังนี้

BADMINTON BASEBALL BASKETBALL CROQUET SOCCER TENNIS VOLLEYBALL

จากค่า EXPECT นี้สามารถนำไปแสดงคำถามกับผู้ใช้ระบบได้ดังนี้

ภาพที่ 39



แสดงตัวอย่างพารามิเตอร์แบบ ASK-ALL

- MULTIVALUED

พารามิเตอร์ที่กำหนด TYPE เป็น MULTIVALUED จะคล้ายกับ ASK-ALL คือสามารถกำหนดค่าให้พารามิเตอร์ได้มากกว่าหนึ่งค่า แต่จะมีข้อแตกต่างดังต่อไปนี้ คือ

- MULTIVALUED จะไม่มีคุณสมบัติที่เป็น EXPECT
- MULTIVALUED ไม่จำเป็นต้องใช้คุณสมบัติที่เป็น PROMPT
- ถ้า MULTIVALUED มีคุณสมบัติที่เป็น PROMPT แล้ว PC/PLUS จะแสดงให้ผู้ใช้กำหนดค่าให้กับพารามิเตอร์หนึ่งค่า ในหนึ่งครั้งที่แสดงผล

ตัวอย่าง

กำหนด RUG-COLOR เป็นพารามิเตอร์มี TYPE เป็น MULTIVALUED และมีคุณสมบัติของ PROMT ดังนี้

Would you like a VALU rug in this room ?

VALU เป็นคำสงวนที่ใช้แทนค่าพารามิเตอร์ โดย PC/PLUS จะแทนค่า VALU ด้วยค่าของพารามิเตอร์ขณะที่แสดงค่า PROMPT

และในฐานความรู้มีกฎอยู่ดังนี้

RULE 8

IF statement : WALL-COLOR = WHITE AND RUG-COLOR = BLUE

THAN statement : DRAPERY-COLOR = GROUPA

RULE 9

IF statement : WALL-COLOR = WHITE AND RUG-COLOR = GRAY

THAN statement : DRAPERY-COLOR = GROUPB

จากกฎข้างต้นในการที่จะหาค่าของ DRAPERY-COLOR นั้น PC/PLUS จะต้องหาค่าของ RUG-COLOR ก่อน ซึ่งถ้า PC/PLUS ประมวลผลกฎข้อ 8 จะได้ prompt ปรากฏดังนี้

Would you like a BLUE rug in this room?

และถ้า PC/PLUS ประมวลผลกฎข้อ 9 จะได้ prompt ปรากฏดังนี้

Would you like a GRAY rug in this room?

ซึ่ง PC/PLUS สามารที่จะแทนค่า RUG-COLOR กี่ครั้งก็ได้ในขณะที่ใช้งานระบบ

- SINGLEVALUED

พารามิเตอร์ที่กำหนดชนิดเป็น SINGLEVALUED จะสามารถมีค่าที่ปรากฏเป็นค่าความแน่นอนได้เพียงค่าเดียว แต่อย่างไรก็ตามค่าพารามิเตอร์อาจมีหลายค่าถ้าค่าความแน่นอนของแต่ละค่าน้อยกว่า 100 เปอร์เซ็นต์ ในการกำหนดค่าพารามิเตอร์ให้เป็น SINGLEVALUED นั้น PC/PLUS จะให้ผู้ใช้กำหนด EXPECT ของตัวแปรนั้นด้วย ซึ่งสามารถกำหนดค่า EXPECT ได้ตามรูปแบบต่อไปนี้

- SINGLE-LINE-INPUT
- MULTI-LINE-INPUT
- INTEGER
- NUMBER
- POSITIVE-NUMBER

- User-defined เป็นการกำหนดค่าในกรณีอื่นๆ ที่นอกเหนือจากที่กล่าวมาแล้ว

- YES/NO

จะแสดงค่าเป็นได้ทั้งตอบรับ และปฏิเสธ โดยมีเงื่อนไขเป็นข้อกำหนดในการตัดสินใจ และในภาษาอังกฤษคำต่อไปนี้ ได้แก่ are , did , do , does , had , has , have , is , shall , should , was , were , will , would จะสามารถใช้ได้กรณีพิเศษในการกำหนดค่าใน TRANSLATION เช่น กำหนดค่าใน TRANSLATION เป็น it is a car. PC/PLUSจะทำการเปลี่ยนค่าให้ตอนแสดงผลเป็นประโยคคำถาม คือ Is it a car?

แต่ถ้ากำหนดคุณสมบัติให้เป็น PROMPT แต่ไม่ใช้คำภาษาอังกฤษที่ได้กำหนดเอาไว้ PC/PLUSจะสร้างวลี Is it ture that นำหน้าประโยคภาษาอังกฤษที่เขียนไว้ใน TRANSLATION และเครื่องหมาย ! เมื่อนำไปไว้หน้าคำใด จะมีความหมายปฏิเสธคำๆนั้น

2. ACTIVE-VALUE

เป็นพารามิเตอร์ที่ใช้ในการบอก PC/PLUS ถึงการกระทำที่พิเศษ โดยกำหนดให้มีการกระทำพิเศษเกิดขึ้นเมื่อพารามิเตอร์มีการรับค่าเข้ามา ซึ่งการกระทำนั้นๆ จะสามารถเป็นส่วนของการกระทำที่อยู่ในสถานะ THEN ของกฎได้ ลักษณะของการกระทำมีหลายอย่างเช่น

- ใช้เปลี่ยนค่าของพารามิเตอร์อื่น
- ใช้เกี่ยวกับการสร้างเสียง
- ใช้เกี่ยวกับการแสดงผลออกเป็นรูปภาพ
- ใช้เพื่อเป็นตัวกำหนดให้การกระทำบางอย่างเกิดขึ้น โดยการกระทำนั้นอยู่บนพื้นฐานของการเรียกใช้ User function

ตัวอย่าง

มีพารามิเตอร์ PRESSURE มีค่า ACTIVE-VALUE ดังนี้

PICTURE “draw-dail”

เมื่อมีการกำหนดค่าให้กับ PRESSURE แล้ว PC/PLUS จะแสดงรูปภาพในแฟ้มข้อมูล DRAW-DAIL

3. ASK-FIRST

เป็นพารามิเตอร์ที่มีคุณสมบัติในการบอกให้ PC/PLUS ถามผู้ใช้ ถึงค่าของพารามิเตอร์ก่อนหน้าที่จะใช้กฎต่างๆ ในการกำหนดค่า ASK-FIRST จะต้องมีคุณสมบัติที่เป็น PROMPT อยู่ด้วยเสมอ พารามิเตอร์ที่มีคุณสมบัติ ASK-FIRST ส่วนมากจะเป็นพารามิเตอร์ที่ผู้ใช้รู้ค่า หรือ PC/PLUS ไม่สามารถอ้างอิงค่าจากฐานความรู้ได้ เช่นชื่อของผู้ใช้เป็นต้น

4. CERTAINTY-FACTOR-RANGE

เป็นการกำหนดระดับความแน่นอน หรือความน่าจะเป็น ของความจริงโดยให้ผู้ใช้เป็นผู้กำหนด ซึ่งมี 3 รูปแบบดังนี้ คือ

- FULL : เป็นการกำหนดช่วงความแน่นอนจาก YES ไปยัง NO
- POSITIVE : เป็นการกำหนดช่วงความแน่นอนจากตำแหน่งใดๆ ไปยัง YES
- UNKNOWN : เป็นการรวมการเลือกที่เป็นไปได้อื่นเข้าไว้ด้วยกัน

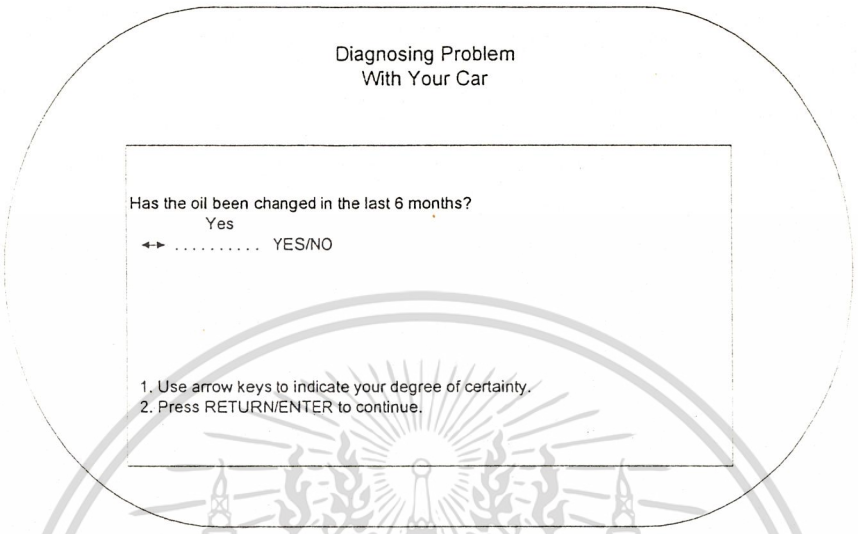
ตัวอย่างสถานการณ์ที่ควรใช้ CERTAINTY-FACTOR-RANGE เพื่อกำหนดค่าความไม่แน่นอน มีดังต่อไปนี้

- เป็นการคาดคะเนถึงเหตุการณ์ในอนาคต
- เมื่อผู้ใช้ไม่แน่ใจในคำตอบที่ตอบ
- เมื่อผู้ใช้ต้องการกำหนดคำตอบในรูปของอัตรา หรือปริมาณของความแน่นอนในคำตอบ

ถ้าเลือก CERTAINTY-FACTOR-RANGE เป็น FULL หรือ POSITIVE จะมีการแสดงเป็นสเกลซึ่งประกอบด้วยกรอบสี่เหลี่ยมเล็กๆรวมอยู่ด้วย ขึ้นอยู่กับการเลือกค่าความแน่นอนของผู้ใช้ โดยที่พารามิเตอร์ YES/NO จะมี FULL , POSITIVE หรือ UNKNOWN ในขณะที่ ASK-ALL จะเป็น FULL หรือ POSITIVE ส่วน MULTIVALUED จะไม่มีค่าความแน่นอน เว้นแต่จะเปลี่ยนเป็นพารามิเตอร์ YES/NO

ตัวอย่างของ พารามิเตอร์ที่มีคุณสมบัติเป็น YES/NO และมี CERTAINTY-FACTOR-RANGE มีคุณสมบัติเป็น POSITIVE

ภาพที่ 40



แสดงตัวอย่างของการใช้งาน CERTAINTY-FACTOR-RANGE

5. DEFAULT

PC/PLUS จะนำค่า default มาใช้ถ้ากับพารามิเตอร์ ถ้าไม่สามารอ้างอิงค่าจากทางอื่นๆ ได้ ทางอื่นๆ ที่ว่านี้อาจเป็นจาก คุณสมบัติ ACTIVE-VALUE หรือถามจากผู้ใช้เป็นต้น

6. DICTIONARY

เป็นคุณสมบัติอย่างหนึ่งที่ทำงานร่วมกับคำสั่ง HOW และ VIEW ในระหว่างที่ระบบให้คำปรึกษา ซึ่งคำสั่ง HOW จะแสดงพารามิเตอร์ ที่ค่าของมันอ้างอิงจากทางอื่นที่ไม่ใช่การถามจากผู้ใช้ระบบ ส่วนคำสั่ง REVIEW จะแสดงพารามิเตอร์ต่างๆ ที่ในระหว่างการให้คำปรึกษาของระบบมีการแทนค่าพารามิเตอร์นั้นโดยการถามจากผู้ใช้ระบบ

7. EXPECT

จะเป็นคุณสมบัติที่มีไว้สำหรับกำหนดค่าที่เป็นไปได้ของ SINGLE VALUE หรือ ASK-ALL พารามิเตอร์ เช่นถ้าเราสร้างพารามิเตอร์ประเภท SINGLEVALUE ในส่วนของคุณสมบัติ PROMPT PC/PLUS จะให้ผู้ใช้กำหนดคุณสมบัติ EXPECT ซึ่งผู้ใช้สามารถเลือกคุณสมบัติได้ดังต่อไปนี้ SINGLE-LINE-INPUT , MULTI-LINE-INPUT , INTEGER , NUMBER , POSITIVE-NUMBER และ User-defined เป็นการกำหนดค่าในกรณีอื่นๆ ที่นอกเหนือจากที่กล่าวมาแล้ว

8. PROMPT

PROMPT เป็นคุณสมบัติที่บอก PC/PLUS ถึงลักษณะการแสดงคำถามที่เกี่ยวกับค่าของพารามิเตอร์ กับผู้ใช้ระบบ ถ้าไม่มีการถามผู้ใช้ระบบเกี่ยวกับค่าของพารามิเตอร์ พารามิเตอร์ตัวนั้นก็ไม่จำเป็นต้องมีคุณสมบัติของ PROMPT โดยจะมีเงื่อนไขในการกำหนด prompt อยู่ 3 ประการ กำหนด prompt เอง หรือ PC/PLUS สร้าง prompt ให้เอง และถ้าไม่ต้องการพารามิเตอร์ที่มี prompt ก็ให้กดคีย์ ENTER ผ่านไปได้ในขณะที่ระบบให้ผู้ใช้ใส่คุณสมบัติ PROMPT

9. TRANSLATION

เป็นคุณสมบัติที่เก็บข้อความที่บรรยายถึงพารามิเตอร์นั้น โดย PC/PLUS มีวัตถุประสงค์ในการใช้คุณสมบัติ TRANSLATION ดังต่อไปนี้ คือ

- เพื่ออ้างถึงพารามิเตอร์ขณะเคลื่อนย้ายกฎ
- เพื่อสร้าง PROMPT สำหรับพารามิเตอร์ ASK-ALL , YES/NO และ SINGLE-VALUED ถ้า PROMPT มีค่าเป็น YES
- เพื่ออ้างถึงพารามิเตอร์ซึ่งตอบรับการทำงานไปยังคำสั่ง HOW , WHY หรือ REVIEW

จากที่กล่าวมาทั้งหมดเป็นการกล่าวพอสรุป เพื่อให้มองเห็นภาพรวมการใช้งาน และทำงานของ PC/PLUS ส่วนในรายละเอียดการของใช้งานจริง ๆ ยังมีข้อปลีกย่อยอีกมาก ซึ่งสามารถศึกษาเพิ่มเติมได้จากคู่มือการใช้งานของ PC/PLUS

ภาคผนวก ข

ระบบจัดการฐานข้อมูลแบบสัมพันธ์ INGRES

(INGRES RELATIONAL DATA BASE MANAGEMENT SYSTEM)

INGRES เป็นระบบจัดการฐานข้อมูลแบบสัมพันธ์ชนิดหนึ่งที่ใช้กันแพร่หลายทั่วไป ระบบจัดการฐานข้อมูลแบบสัมพันธ์ (RDBMS : Relational Database Management System) ที่ใช้อยู่ทั่วไปจะทำงานในลักษณะเป็นตัวเชื่อมต่อระหว่างหน่วยเก็บข้อมูลแบบกายภาพ (PHYSICAL) กับข้อมูลทางตรรกะ (LOGICAL) ซึ่งในการเชื่อมต่อนี้จะมีเครื่องมือ (Tools) เข้ามาช่วยในการปฏิบัติงานต่าง ๆ ซึ่งมีหลายรูปแบบพอสรุปได้ดังนี้

- ให้นิยาม และสร้างฐานข้อมูล (Define a database)
- สอบถามข้อมูลจากฐานข้อมูล (Query the database)
- เพิ่มเติม แก้ไข้ และลบข้อมูล (Add,edit and delete data)
- เปลี่ยนแปลงโครงสร้างของฐานข้อมูล (Modify the structure of the database)
- รักษาความปลอดภัยแก่ข้อมูลจากการใช้งาน (Secure data from public access)
- ติดต่อสื่อสารระหว่างระบบเครือข่าย (Communicate within networks)
- ส่งข้อมูลออก และรับข้อมูลเข้า (Export and import data)

การที่มีผู้ใช้งานหลายคนบนฐานข้อมูลเดียวกันนับว่าเป็นสิ่งจำเป็นมากสำหรับระบบจัดการฐานข้อมูลแบบสัมพันธ์ โดยต้องคำนึงถึงความปลอดภัยของข้อมูลที่มีอยู่ด้วย ดังนั้นโปรแกรมประยุกต์เกี่ยวกับฐานข้อมูลจึงได้มีการพัฒนามาเป็นระบบที่มีผู้ใช้หลายคน (Multi-user) ด้วย

<u>การจัดการข้อมูลของ INGRES</u>

ในทางตรรกะ ข้อมูลทั้งหมดของ INGRES จะถูกเก็บ และแสดงในรูปของตาราง (Table) ในแต่ละตารางจะประกอบไปด้วยคอลัมน์ (Columns) หรือบางทีเรียกว่า ฟิลด์ (fields) และแถว (Rows) แต่ละแถวข้อมูลจะเรียกว่า 1 เรคอร์ด จากตารางผู้ใช้อาจจะเลือกสับเซตของแถวหรือคอลัมน์ แต่ผลลัพธ์จะแสดงออกมาบนหน้าจอ หรือถูกพิมพ์ออกมาเป็นรูปแบบของตารางด้วย

วิว (View) ได้มาจากตารางซึ่งผู้ใช้สามารถสร้างเพื่อจุดประสงค์สำหรับการแสดงผล ถึงแม้ว่า วิวจะมองดูเหมือนกับว่าเป็นตารางจริง ๆ แต่ในฐานข้อมูล วิวจะเก็บในลักษณะนิยามเท่านั้น ด้วยเหตุผลนี้ วิวก็จะถูกอ้างว่าเป็นตารางเสมือน (Virtual Tables) โดยตารางที่ถูกวิวถ่ายทอดเรียกว่า

ตารางพื้นฐาน (Base Tables) วิวอาจจะเกิดจากการรวมกันของ 2 ตารางพื้นฐาน หรือเป็นซับเซตของตารางพื้นฐาน ผู้ใช้สามารถใช้วิวเพื่อที่จะเข้าถึงตารางได้ และเพื่อทำให้งานที่มีความยุ่งยากให้มีความง่ายขึ้น

เนื่องจาก INGRES เป็นระบบจัดการฐานข้อมูลแบบสัมพันธ์ ผู้ใช้จึงสามารถที่จะเชื่อมโยงข้อมูลที่ถูกเก็บไว้ในหลาย ๆ ตารางเพื่อเพิ่มประสิทธิภาพในการทำงาน และเพื่อการหลีกเลี่ยงการทำซ้ำ การเลือก (Selection) เป็นกระบวนการเพื่อสร้างตารางใหม่ซึ่งประกอบไปด้วยกลุ่มของแถวจากตารางใดๆ ซึ่งตรงกับเกณฑ์ที่ต้องการ โปรเจ็คชั่น (Projection) เป็นกระบวนการเพื่อสร้างตารางจากกลุ่มของคอลัมน์จากตารางใดๆ ซึ่งตรงกับเกณฑ์ที่ต้องการ การจอยน์ (Join) จะสร้างตารางใหม่ซึ่งเป็นการเชื่อมกันของแถวทั้งหมดใน 2 ตาราง โดยไม่รวมแถวที่มีข้อมูลเหมือนกัน

ในทางกายภาพ INGRES มีโครงสร้างในการเก็บข้อมูล 4 แบบคือ HEAP , ISAM ,HASH และ BTREE ซึ่งรายละเอียดมีกล่าวไว้ในบทที่ 3

<u>การเข้าถึงข้อมูลของ INGRES</u>

INGRES มีภาษาที่ใช้ในการเข้าถึงข้อมูลคือ เอสคิวแอล (SQL:Structured Query Language) เป็นภาษาซึ่งผู้ใช้สามารถติดต่อกับ INGRES เอสคิวแอลประกอบไปด้วยกลุ่มคำพื้นฐานของภาษาอังกฤษ เช่น Select และ Create เป็นต้น ซึ่งผู้ใช้สามารถใช้คำสั่งโครงสร้างเพื่อที่จะเข้าถึง และจัดการข้อมูลที่ถูกเก็บไว้ในฐานของมูลแบบสัมพันธ์ ซึ่งมีกล่าวราละเอียดไว้ในบทที่ 2 นอกจากใช้ภาษาเอสคิวแอลในการเข้าถึงข้อมูลแล้ว INGRES ยังมีเครื่องมืออื่นช่วยในการเข้าถึงข้อมูลได้อีกเช่น มี FROM ในการดึงข้อมูลจากตารางในฐานข้อมูล

INGRES มี Query optimizer สำหรับช่วยเลือกเส้นทางการเข้าถึงข้อมูลเพื่อให้ได้เส้นทางที่ดีที่สุด โดยจะมีคำสั่งในการสร้างตารางสถิติเก็บรายละเอียดการกระจายของข้อมูลที่มีอยู่ในฐานข้อมูล เพื่อนำมาให้ Query optimizer ใช้ประกอบการตัดสินใจ นอกจากนั้นยังมี Query execution plan ไว้สำหรับดูการทำงานของ Query โดยจะแสดงถึงเส้นทางที่ Query optimizer เลือกในการประมวลผล Query พร้อมทั้งแสดงปริมาณการใช้งาน INPUT/OUTPUT และการใช้งาน CPU ของ Query นั้นอีกด้วย

ประวัติผู้เขียน

ชื่อผู้เขียน	นาย พิทักษ์ ธรรมวาริน
วันเดือนปีเกิด	วันที่ 22 สิงหาคม พ.ศ. 2511
สถานที่เกิด	จังหวัดชลบุรี
วุฒิการศึกษาระดับประถม	โรงเรียนชุมชนวัดโบสถ์
วุฒิการศึกษาระดับมัธยม	โรงเรียนเบญจมราชรังสฤษฎิ์
วุฒิการศึกษาระดับปริญญาตรี	วิทยาศาสตร์บัณฑิต สาขาวิชา สถิติประยุกต์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ผลงานการศึกษาระดับปริญญาตรี	พัฒนาโปรแกรมช่วยวิเคราะห์เส้นทางการจราจรในเขตกรุงเทพมหานคร ซึ่งได้รับการเผยแพร่ผลงานทางไทยทีวีสีช่อง 5
ผลงานทางวิชาการที่ได้รับการตีพิมพ์	เรื่องระบบผู้เชี่ยวชาญสำหรับช่วยปรับแต่งฐานข้อมูลแบบรีเลชั่นแนล ในการประชุมวิชาการทางวิศวกรรมไฟฟ้า ครั้งที่ 16
ประสบการณ์การทำงาน	- programmer analysis บริษัท MIS จำกัด - หัวหน้าแผนกสารสนเทศ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง