



การวิเคราะห์และเปรียบเทียบระบบจัดการฐานข้อมูลเชิงสัมพันธ์
ANALYSIS AND COMPARISON OF RELATIONAL DATABASE
MANAGEMENT SYSTEM



ปริญญาานิพนธ์สำหรับปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าลาดกระบัง
ปีการศึกษา 2537

ปริญญาโทบริหารการศึกษา 2537

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การวิเคราะห์และเปรียบเทียบระบบจัดการฐานข้อมูลเชิงสัมพันธ์

Analysis and Comparison of Relational Database Management Systems

ผู้จัดทำ

1. นายพรชัย มกรพันธ์ รหัส 35103236

2. นายเอกอรุพงษ์ หงษ์สาม รหัส 35103262

อาจารย์ที่ปรึกษา

(ดร. เอื้อน ปิ่นเงิน)

การวิเคราะห์และเปรียบเทียบระบบจัดการฐานข้อมูลเชิงสัมพันธ์
Analysis and Comparison of Relational Database Management Systems

โดย นายพรชัย มกรพันธ์
นายเอกอรุพงษ์ หงษ์สาม

อาจารย์ที่ปรึกษา ดร. เอื้อน ปิ่นเงิน

ปีการศึกษา 2537

บทคัดย่อ

ในปัจจุบันนี้ระบบจัดการฐานข้อมูลเชิงสัมพันธ์ ได้รับความนิยมอย่างมากในการนำมาจัดเก็บข้อมูล ซึ่งโครงการนี้ได้นำเสนอการวิเคราะห์และเปรียบเทียบระบบจัดการฐานข้อมูลเชิงสัมพันธ์ โดยทำการเปรียบเทียบระบบจัดการฐานข้อมูลเชิงสัมพันธ์ 4 ตัว คือ ORACLE, INGRES, INFORMIX และ EMPRESS บนโฮสต์เดี่ยว และไคลเอนต์เซิร์ฟเวอร์ โดยนำเอากฎการออกแบบฐานข้อมูล และกฎการวิเคราะห์ระบบจัดการฐานข้อมูลของ E.F. CODD รวมถึงคุณสมบัติและความสามารถด้านต่าง ๆ ของระบบจัดการฐานข้อมูลแต่ละตัว มาเพื่อทำการเปรียบเทียบ

Abstract

At the time being, the Relational Database Management System is quite popular in the way of data collection. This project is intended to propose the analysis and comparison of 4 Relational Database Management Systems , i.e, Oracle, Ingress, Informix and Empress which based on single host and client-server. The law of Database design, law of Database Management Analysis (E.F CODD) and each Database Management System's specification and performance are used as the analysis tools in this project.

สารบัญ

	หน้า
บทที่ 1 : บทนำ	1
บทที่ 2 : หลักการและทฤษฎีที่เกี่ยวข้อง	3
Architecture	3
Data Definition	13
Data Manipulation	14
Semantic Data Control	15
Transaction Management	17
Query Processing	23
Additional Tools	25
Distributed Database Management	26
บทที่ 3 : การวิเคราะห์ระบบจัดการฐานข้อมูลของ ORACLE	29
Architecture	29
Semantic Data Control	30
Transaction Management	32
Query Processing	41
Additional Tools	45
Distributed Database Management	50
บทที่ 4 : การวิเคราะห์ระบบจัดการฐานข้อมูลของ INGRES	52
Architecture	52
Semantic Data Control	52
Transaction Management	55
Query Processing	61
Additional Tools	68
Distributed Database Management	71

สารบัญ (ต่อ)

	หน้า
บทที่ 5 : การวิเคราะห์ระบบจัดการฐานข้อมูลของ INFORMIX	72
Architecture	72
Semantic Data Control	72
Transaction Management	73
Query Processing	77
Additional Tools	78
Distributed Database Management	79
บทที่ 6 : การวิเคราะห์ระบบจัดการฐานข้อมูลของ EMPRESS	80
Architecture	80
Semantic Data Control	81
Transaction Management	82
Query Processing	84
Additional Tools	84
Distributed Database Management	86
บทที่ 7 : การทดสอบระบบจัดการฐานข้อมูลเชิงสัมพันธ์	88
การทดสอบประสิทธิภาพของระบบจัดการฐานข้อมูล	88
ผลการทดสอบประสิทธิภาพของ INGRES V.6.4	94
ผลการทดสอบประสิทธิภาพของ INFORMIX V.5	108
ผลการทดสอบประสิทธิภาพของ EMPRESS V.6	123
บทที่ 8 : กฎของ E.F CODD ในการวิเคราะห์ระบบฐานข้อมูล	137
กฎของ E.F CODD 12 ข้อ ในการวิเคราะห์ระบบฐานข้อมูล	137
กฎของ E.F CODD 16 ข้อ ในการออกแบบระบบฐานข้อมูล	142
บทที่ 9 : บทสรุป	147
หนังสืออ้างอิง	
กิตติกรรมประกาศ	

บทที่ 1

บทนำ

ระบบจัดการฐานข้อมูลเชิงสัมพันธ์ (Relational Database Management Systems, RDBMS) เป็นระบบจัดการฐานข้อมูลที่ได้รับความนิยมแพร่หลายในปัจจุบัน เนื่องจากมีความสะดวกในการควบคุมความถูกต้อง (data Integrity) และช่วยลดความซ้ำซ้อน (redundancy) ของข้อมูล โดยมีวิธีการในการจัดเก็บฐานข้อมูลในลักษณะของตาราง (table) และกำหนดรายละเอียดของตารางว่ามีแอตทริบิวต์ (attribute) หรือ ฟิลด์ (fields) ไตบ้าง ซึ่งข้อมูลเหล่านี้ จะถูกเก็บอยู่ในพจนานุกรมข้อมูล (data dictionary) ซึ่งในตารางนี้จะมีฟิลด์หนึ่งที่ถูกกำหนดขึ้นเป็นคีย์ ที่เรียกว่า primary key โดยจะใช้สำหรับการอ้างอิงข้อมูลในตาราง เป็นการช่วยให้การเข้าถึงข้อมูลในตารางเป็นไปได้อย่างรวดเร็วขึ้น

ในการวิเคราะห์ระบบจัดการฐานข้อมูล ได้นำเอาทฤษฎีของ E.F. Codd มาใช้ในการวิเคราะห์ ซึ่งประกอบไปด้วย กฎ 12 ข้อในการวิเคราะห์ฐานข้อมูล และ กฎ 16 ข้อในการออกแบบการจัดการฐานข้อมูล นอกจากนี้ ยังได้ดูถึงส่วนประกอบต่าง ๆ ของ ระบบจัดการฐานข้อมูล เช่น architecture , concurrency control , additional tool มาช่วยในการวิเคราะห์ระบบจัดการฐานข้อมูล

ในการทดสอบระบบการจัดการฐานข้อมูล เราได้นำทฤษฎีและหลักการต่าง ๆ มาช่วยในการพิจารณาประกอบชุด query ซึ่งในแต่ละ query จะมีเกณฑ์ในการพิจารณาต่าง ๆ กัน เช่น query ที่เปรียบเทียบคุณสมบัติ syntax independence โดยจะเปรียบเทียบผลที่ได้จาก query แบบ join และ subquery รวมถึงเวลาที่ใช้ในการประมวลผล query

วัตถุประสงค์ของโครงการ

1. เพื่อเปรียบเทียบ และวิเคราะห์ระบบจัดการฐานข้อมูลแต่ละตัว
2. เพื่อเปรียบเทียบประสิทธิภาพระบบจัดการฐานข้อมูลแต่ละตัว

ขอบเขตของการศึกษาโครงการ

ในโครงการจะทำการศึกษาดังคุณลักษณะของระบบจัดการฐานข้อมูลเชิงสัมพันธ์ แต่ละตัวจากนั้นจึงทำการทดสอบประสิทธิภาพและความสามารถต่าง ๆ ของระบบจัดการฐานข้อมูลเชิงสัมพันธ์โดยไม่คำนึงถึงเวลาที่ใช้ในการทดสอบเพราะระบบจัดการฐานข้อมูลเชิงสัมพันธ์แต่ละตัวที่นำมาทดสอบนั้นทำงานบน platform ที่ต่างกัน จึงไม่สามารถนำเวลาที่ได้มาเปรียบเทียบกันได้ แต่ก็สามารถได้ผลที่ได้ในการทดสอบมาเป็นบรรทัดฐานในการทดสอบครั้งต่อไป

โดยระบบจัดการฐานข้อมูลเชิงสัมพันธ์ INGRES version 6.4 ทำงานบนเครื่อง VAX 6000/510

O/S	VAX VMS 6/2
RAM	64 MB
Harddisk	7.5 GB
Terminal	Dumb 120 terminals

ระบบจัดการฐานข้อมูลเชิงสัมพันธ์ INFORMIX version 5 ทำงานบนเครื่อง IBM RISC/6000 Serie 300

O/S	AIX/UNIX
RAM	64 MB
Harddisk	2 GB
Terminal	Dumb 2 terminals

ระบบจัดการฐานข้อมูลเชิงสัมพันธ์ EMPRESS version 6 ทำงานบนเครื่อง PC CPU 486DX/2-80 (AMD)

O/S	Interactive UNIX
RAM	16 MB
Harddisk	500 MB SCSI
Terminal	PC 4 terminals

โดยในการทดสอบจะใช้ตารางขนาด 25,000 เรคคอร์ด จำนวน 2 ตาราง และนำเข้าระบบจัดการฐานข้อมูลเชิงสัมพันธ์แต่ละตัว ด้วย function พื้นฐานของระบบจัดการฐานข้อมูลเชิงสัมพันธ์ แล้วเขียน query เพื่อทดสอบเรื่องเวลา และ ประสิทธิภาพ โดยแบ่งเป็นลักษณะของ 1 ผู้ใช้ และ 2 ผู้ใช้

บทที่ 2

หลักการและทฤษฎีที่เกี่ยวข้อง

รายละเอียดของระบบจัดการฐานข้อมูลเชิงสัมพันธ์

ระบบจัดการฐานข้อมูลเชิงสัมพันธ์ประกอบด้วยกลุ่มของข้อมูลที่มีความเกี่ยวข้องกันและกลุ่มของโปรแกรมที่ใช้ในการเข้าถึงข้อมูล กลุ่มของข้อมูลโดยทั่ว ๆ ไป จะหมายถึงฐานข้อมูล ซึ่งจะเป็นข้อมูลเฉพาะของหน่วยงานใดหน่วยงานหนึ่ง จุดประสงค์หลักของระบบจัดการฐานข้อมูลก็คือ เพื่อที่จะอำนวยความสะดวกและเพิ่มประสิทธิภาพในการเก็บข้อมูลเข้าและดึงข้อมูลออกจากระบบจัดการฐานข้อมูล

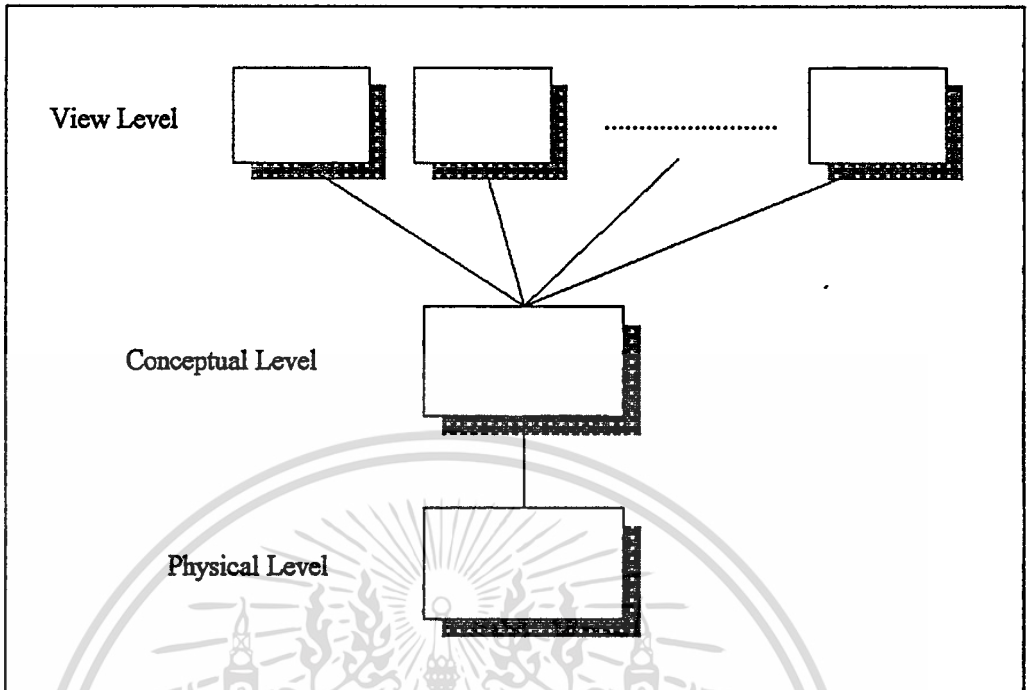
ระบบจัดการฐานข้อมูลถูกออกแบบมาเพื่อใช้ในการจัดการกับข้อมูลขนาดใหญ่จำนวนมาก ๆ ซึ่งรวมถึง การกำหนดโครงสร้างที่ใช้ในการเก็บสะสมข้อมูล และการจัดทากลไกในการจัดการกับข้อมูล นอกจากนี้ระบบจัดการฐานข้อมูลยังต้องคำนึงถึงความปลอดภัยของข้อมูลทั้งในกรณีที่ระบบเกิดการชนกันหรือในกรณีที่ผู้ที่ไม่ได้รับอนุญาตพยายามที่จะเข้าถึงข้อมูล โดยเฉพาะอย่างยิ่งถ้าเป็นข้อมูลที่ใช้ร่วมกันระบบจะต้องสามารถหลีกเลี่ยงข้อผิดพลาดต่าง ๆ ที่อาจเกิดขึ้นได้ด้วย

ส่วนประกอบต่าง ๆ ของระบบจัดการฐานข้อมูลเชิงสัมพันธ์นั้นประกอบด้วย

2.1. ARCHITECTURES

2.1.1) Database System Architecture

สถาปัตยกรรมที่ใช้ในการจัดการฐานข้อมูลนั้น ข้อมูลจะถูกเก็บอยู่ในฐานข้อมูลซึ่งผู้ใช้จะสามารถใช้งานผ่านระบบจัดการฐานข้อมูล ซึ่งรูปแบบของการแบ่งระดับของฐานข้อมูลและการใช้งานจะแบ่งเป็นหลายระดับดังรูปที่ 2.1 ซึ่งประกอบไปด้วย



รูปที่ 2.1 ภาพแสดงมุมมองของการจัดการฐานข้อมูลโดยทั่วไป

1) Physical level

เป็น level ล่างสุดซึ่งใน level นี้จะทำหน้าที่อธิบายว่า “ มีการจัดการ ข้อมูลอย่างไร ” ใน level นี้จะมี data structure ที่ค่อนข้างจะยุ่งยากซับซ้อน

2) Conceptual level

เป็น level ที่อยู่ในระดับที่สูงกว่า physical level ซึ่งทำหน้าที่อธิบายว่า “ มีการจัดการ ข้อมูลใด ” ซึ่งผู้ใช้ใน level นี้ไม่จำเป็นต้องเกี่ยวข้องกับโครงสร้างอันยุ่งยากใน physical level เลยถึงแม้ว่าการ การประยุกต์ใช้งาน ใน level นี้ต้องเกี่ยวข้องกับ physical level ก็ตาม ใน level นี้ ผู้บริหารฐานข้อมูล เป็นผู้ใช้

3) View level

ซึ่งในด้านมุมมองของผู้ใช้นั้นข้อมูลที่ใช้เรียกค้นขึ้นมาใช้จะปรากฏตาม view ที่กำหนด โดยผู้ใช้ไม่ต้องทราบเลยว่าข้อมูลเหล่านั้นเก็บไว้อย่างไรและถูกเรียกใช้ได้อย่างไรจากฐานข้อมูล ผู้ใช้รู้เพียงแต่ว่า ข้อมูลที่เรียกมาใช้เป็นไปตามข้อกำหนด หรือความต้องการเท่านั้น

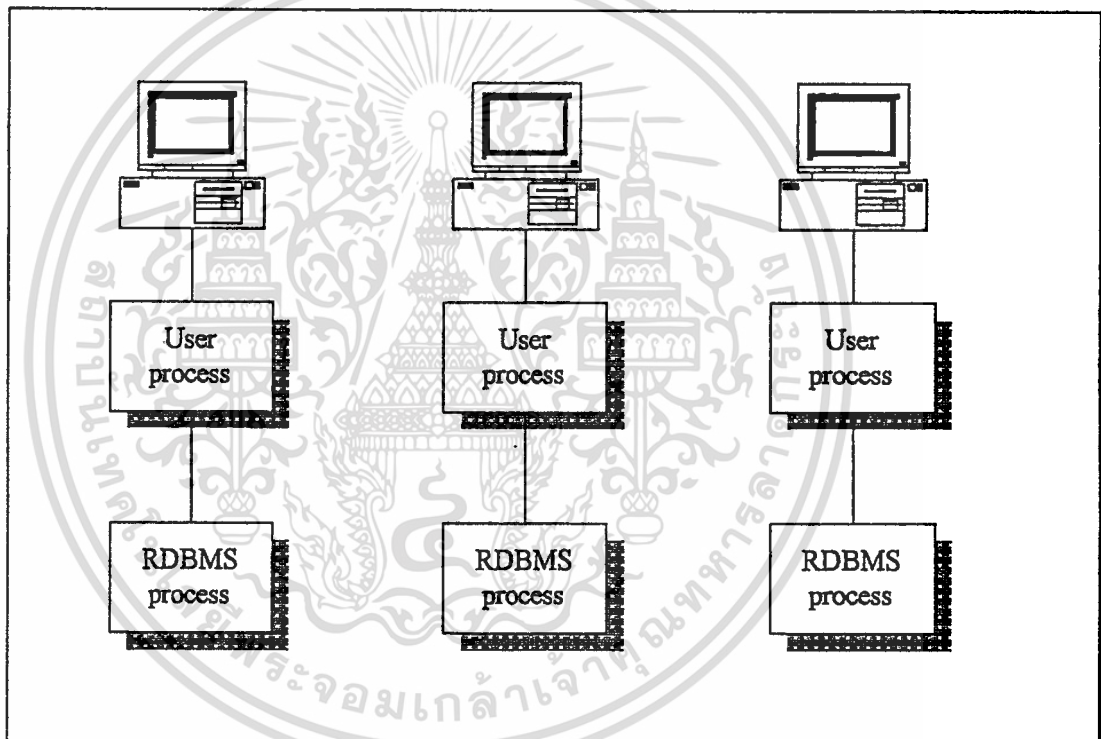
2.1.2 Process Architecture

จะอธิบายการประยุกต์ใช้งาน ของ multiserver DBMS ใน operating system ซึ่งแบ่งออกเป็น 3 model คือ

1) Process Model

เป็นสถาปัตยกรรมที่สนับสนุน “ single threaded ” (“ threaded ” ในความหมายของระบบจัดการฐานข้อมูล คือ execution path) ดังแสดงในรูปที่ 2.2

- มีหลักการทำงาน คือ 1 ผู้ใช้ : 1 process Server
- สามารถทำ multiprogramming ได้



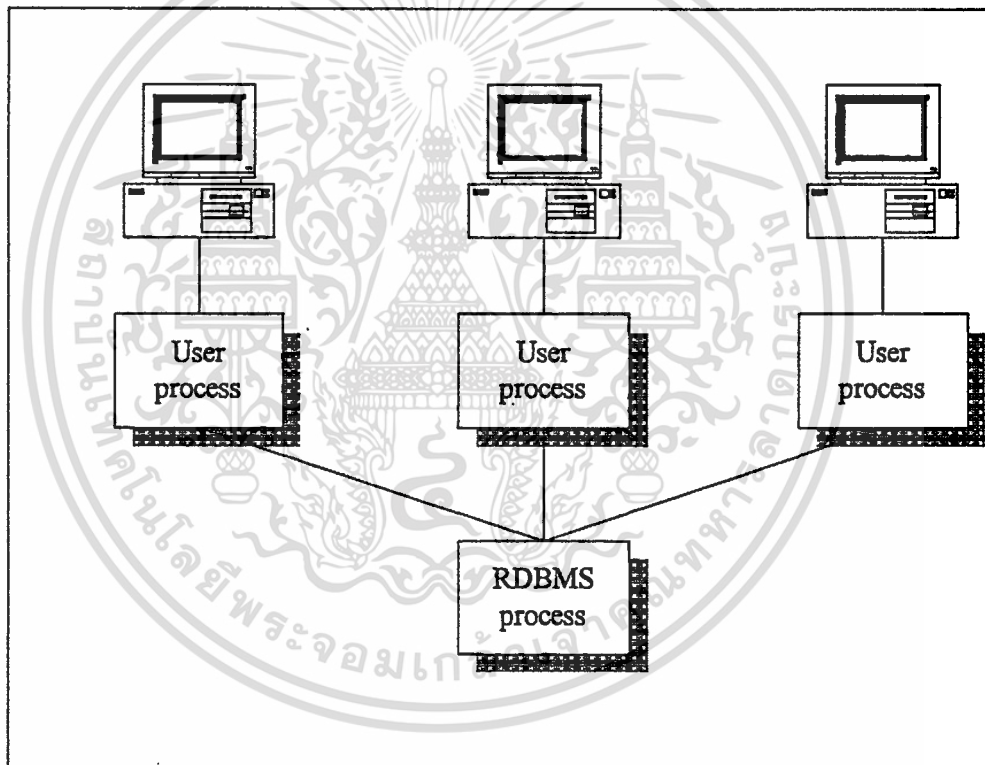
รูปที่ 2.2 ภาพแสดงสถาปัตยกรรมแบบ process model

- ข้อดี
- 1) การประยุกต์ใช้งานง่าย
 - 2) DBMS process มีขนาดเล็ก ดังนั้นสามารถใช้ feature ของ operating system ได้มาก
 - 3) DBMS process ไม่ถูก swap ออกหมด ถ้า swap DBMS process ตัวใดตัวหนึ่งออก ผู้ใช้อื่นๆ ก็สามารถทำงานต่อได้ด้วย DBMS process ของตนเอง

- ข้อเสีย
- 1) จำนวนของ active users น้อยเพราะถูกจำกัดด้วยจำนวนของ process ซึ่งมีเป็นจำนวนมาก
 - 2) เกิด overhead มากเนื่องจาก context switching ของ process มีเป็นจำนวนมาก

2) Single Server Model

จากรูปที่ 2.3 ผู้ใช้แต่ละคนจะส่ง request message เพื่อให้ single server process ทำหน้าที่ server จึงยังถือว่าเป็น “single threaded” เพราะยังถือว่า มี execution path เพียง path เดียว



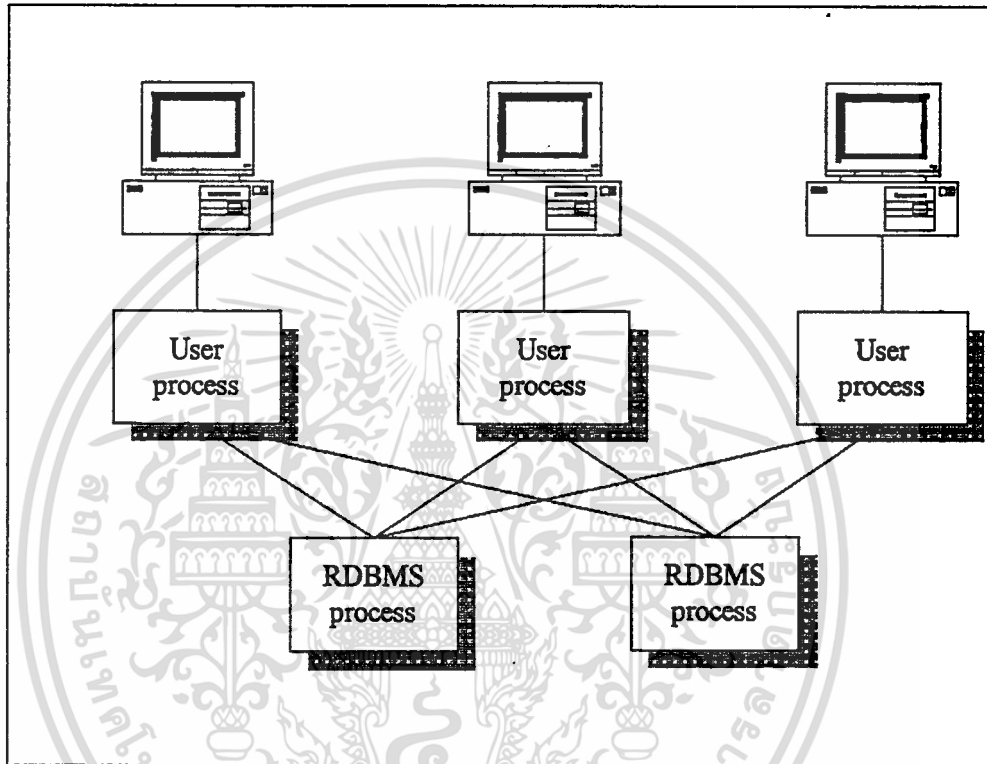
รูปที่ 2.3 ภาพแสดงสถาปัตยกรรมแบบ single server model

- ข้อดี
- 1) เนื่องจากมีเพียง DBMS process เดียวทำให้จำนวนผู้ใช้ที่ active มีจำนวนเพิ่มขึ้น
 - 2) มีจำนวน process ที่ลดลงทำให้ context switching ลดลง ดังนั้นทำให้ overhead ลดลงด้วย
- ข้อเสีย
- 1) ทำ parallel processing ไม่ได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3) Multiple Server Model

มี DBMS process หลายตัวคอย serve users ได้หลายคน ถือว่าเป็น “multithreaded” ดังแสดงในรูป 2.4



รูปที่ 2.4 ภาพแสดงสถาปัตยกรรมแบบ multiple server model

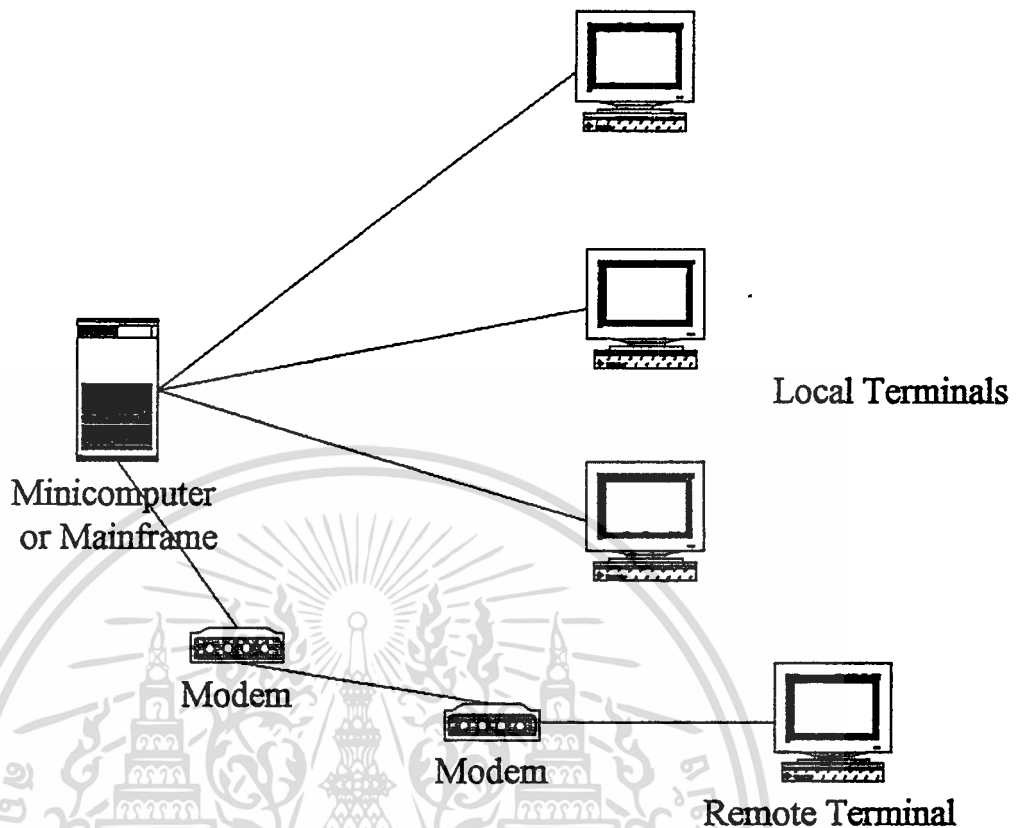
- ข้อดี
- 1) จำนวน process ที่ลดลงทำให้ context switching ลดลง
ดังนั้นทำให้ overhead ลดลงด้วย
 - 2) สามารถทำ parallel processing ได้

2.1.3 Computer System Architectures

1) Centralized Platforms

สำหรับระบบลักษณะนี้ทุก ๆ โปรแกรมจะรันบนคอมพิวเตอร์หลัก (Host Computer) ที่ลิ้นรวมทั้งตัวโปรแกรมประยุกต์ของระบบจัดการฐานข้อมูลที่จะเข้าถึงฐานข้อมูล และ facilities เกี่ยวกับการสื่อสารที่จะทำการส่ง และรับข้อมูลจากเทอร์มินัลของผู้ใช้ด้วย ผู้ใช้สามารถเข้าถึงระบบได้ทั้งโดยการผ่านการเชื่อมต่อแบบโลคอล (local connection) และ การเข้าถึงในระยะไกล (remote access ผ่านโมเด็ม) เทอร์มินัลของระบบแบบนี้จะมีลักษณะเป็นตั้มพ์เทอร์มินัลคือ เทอร์มินัลจะไม่มีความสามารถในการประมวลผลด้วยตัวเอง ส่วนตัวคอมพิวเตอร์หลักมักจะใช้เมนเฟรมหรือมินิคอมพิวเตอร์เป็นหลักโดยการประมวลผลข้อมูลทั้งหมดจะถูกกระทำบนคอมพิวเตอร์หลักนี้

โดยส่วนมากในระบบนี้ เมื่อผู้ใช้เปิดเทอร์มินัล เพื่อที่จะเข้าถึงระบบหน้าจอแรกที่จะปรากฏมักจะเป็นหน้าจอ login เพื่อให้ผู้ใช้ใส่ชื่อ enter login และ password เมื่อมีการ startup โปรแกรมประยุกต์ หน้าจอที่เหมาะสมต่าง ๆ ก็จะถูกส่งผ่านสายมาที่ผู้ใช้เทอร์มินัล และโปรแกรมประยุกต์ก็จะกระทำการต่าง ๆ ตามที่ผู้ใช้กดแป้นพิมพ์หรือใช้เมาส์ ทั้งโปรแกรมประยุกต์ของผู้ใช้ และระบบจัดการฐานข้อมูลต่างก็รันบนคอมพิวเตอร์เดียวกันและจะทำการติดต่อสื่อสารกันโดยผ่าน shared memory ที่ควบคุมและจัดการโดย operating system ของคอมพิวเตอร์หลัก



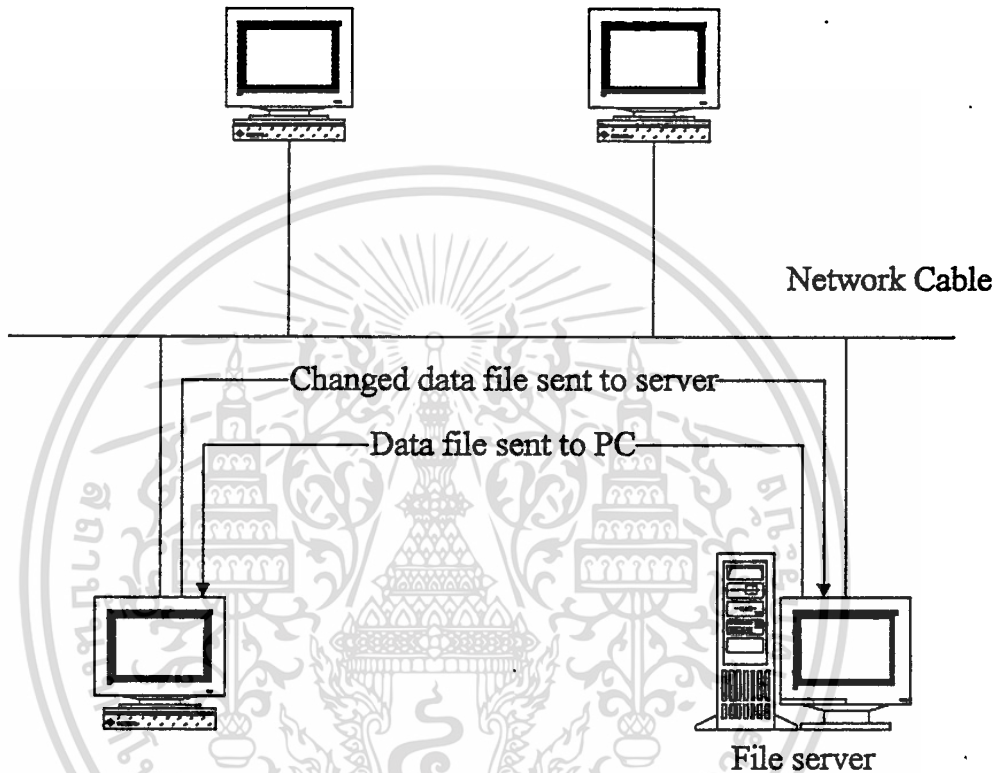
รูปที่ 2.5 ภาพแสดงระบบแบบ centralization

ในรูปที่ 2.5 จะเป็นระบบแบบ centralization ซึ่งเป็นระบบที่สามารถจะรองรับ ผู้ใช้จำนวนมากได้ แต่อาจมีค่าใช้จ่ายสูงทั้งด้านการซื้อและการบำรุงรักษาตัวเมนเฟรม หรือมินิคอมพิวเตอร์ขนาดใหญ่ นั้นบางระบบต้องการ facility พิเศษสนับสนุนมากมาย ตัวอย่างเช่น water cooling systems , climate control systems และ รวมทั้งต้องการทีมงานและผู้เขียนโปรแกรมระบบที่มีความเข้าใจระบบสูงอีกด้วยจึงทำให้มักมีการจัดการฝึกอบรมพนักงานงานที่เกี่ยวข้องเสมอ ๆ

2) PC-based System

การมีระบบจัดการฐานข้อมูลอยู่บน PC , PC จะแสดงตัวเป็นทั้งคอมพิวเตอร์หลักและเทอร์มินัลในขณะเดียวกันคือเป็นระบบ stand-alone หรืออาจจะเรียกว่าระบบ PC-based ก็ได้ ตัวอย่างเช่น ระบบ LANs ที่มีฐานข้อมูลอยู่บน PC และมีการรัน network operating system พิเศษ เช่น Novell's Netware หรือ microsoft LAN manager อยู่บน PC เรียกว่าเป็น file server ซึ่งตัว file server นี้จะเป็นตัวจัดการให้ ผู้ใช้สามารถที่จะ share common data files โดยที่การประมวลผลจริง ๆ ยังคงทำอยู่ที่คอมพิวเตอร์ของผู้ใช้ที่รันโปรแกรมประยุกต์บนฐานข้อมูล file

server จะเพียงแต่ทำการค้นหา ไฟล์ ข้อมูลที่ผู้ใช้ต้องการแล้วส่งผ่านเครือข่ายไปให้คอมพิวเตอร์ของผู้ใช้ และการเปลี่ยนแปลงข้อมูลต่าง ๆ บนคอมพิวเตอร์ของผู้ใช้จะถูกส่งกลับไปที่ file server ในลักษณะการส่งกลับทั้งไฟล์มาเก็บที่ดิสก์บน file server ดังแสดงในรูปที่ 2.6



รูปที่ 2.6 ภาพแสดงระบบแบบ PC-Based

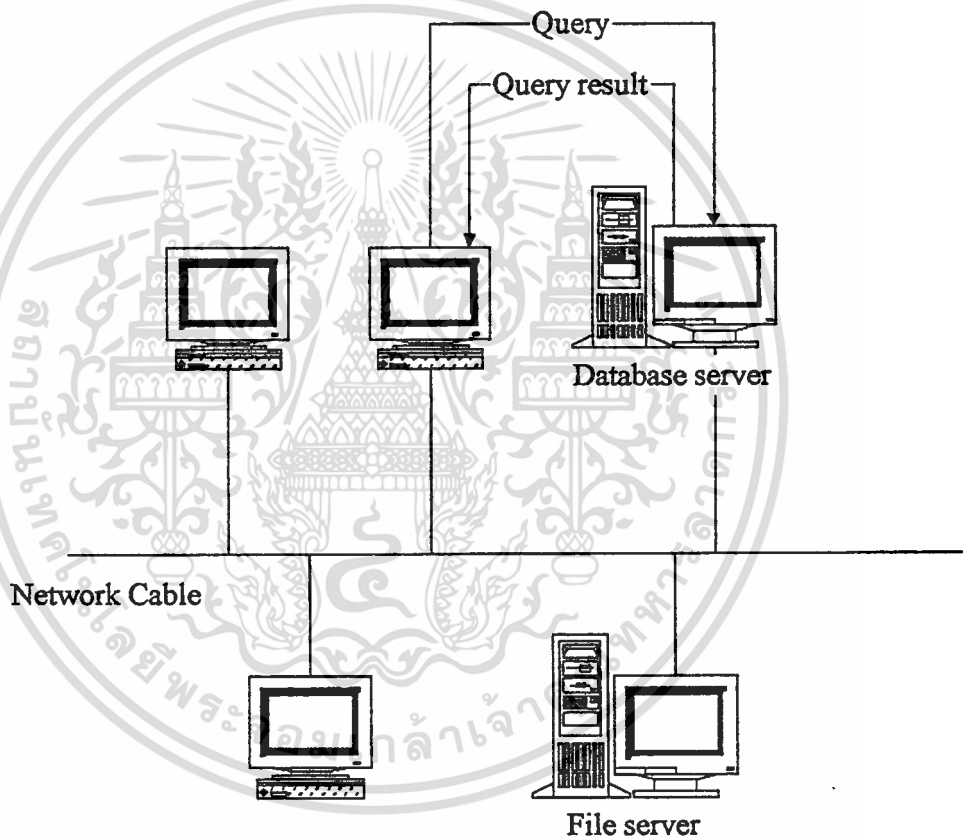
จากการทำงานในลักษณะนี้ความสามารถของตัว file server จะไม่มีผลต่อความเร็วในการประมวลผล เพราะ ความเร็วต่าง ๆ จะถูกจำกัดด้วยความสามารถของคอมพิวเตอร์ของผู้ใช้ (ที่รันระบบจัดการฐานข้อมูล) ระบบนี้ถ้ามีผู้ใช้มาเข้าถึงพร้อม ๆ กันหลาย ๆ คนในไฟล์เดียวกัน ก็อาจจะถูกส่งไปหลาย ๆ ที่ได้ ทำให้เพิ่มการจราจร และทำให้เครือข่ายช้าลงได้

ระบบนี้ ประสิทธิภาพ จะแยกลงเมื่อจำนวนผู้ใช้เพิ่มมากขึ้นจึงไม่สามารถจะรองรับจำนวนผู้ใช้ได้มากเท่ากับ host-based system และระบบ PC-based นี้ไม่สามารถรับประกัน data integrity ได้ เนื่องจากอนุญาตให้มีการทำ direct access กับ data file โดยไม่อยู่ภายใต้การควบคุมของระบบจัดการฐานข้อมูลที่สร้าง data file นั้นขึ้นมา integrity จึงถูกละเมิดได้

3) Client/server database

ระบบนี้ประกอบด้วย

- client เป็นคอมพิวเตอร์ที่รันโปรแกรมประยุกต์ของผู้ใช้
- server เป็นคอมพิวเตอร์ที่รันระบบจัดการฐานข้อมูล (database server) หรือเป็นคอมพิวเตอร์ที่จัดการ ในการจัดสรรทรัพยากร เช่น disk space (file server) หรือ printer (printer server)



รูปที่ 2.7 ภาพแสดงระบบ client/server

จากรูปที่ 2.7 client จะจัดการทุกอย่างเกี่ยวกับหน้าจอและ input/output processing ของผู้ใช้ ส่วน server จะจัดการเกี่ยวกับการประมวลผลข้อมูล การเข้าถึงดิสก์ ฯลฯ เช่น ในกรณีที่ ได้รับ query ที่เกี่ยวข้องกับข้อมูลในฐานข้อมูลจากผู้ใช้โปรแกรมประยุกต์ จะทำการส่ง request นั้นผ่านเครือข่ายไปที่ตัว server และตัว server จะจัดการประมวลผล request นั้นจนได้ผลซึ่งเป็น คำตอบออกมา แล้วส่งข้อมูลที่เป็นคำตอบของ query นั้นกลับไปที่ตัว client ที่เป็นตัว request จะเห็นได้ชัดว่าสามารถลด traffic ได้อย่างมากเมื่อเทียบกับระบบ PC-based โดย client มักจะเป็น PC ในลักษณะที่ server สามารถเป็นได้ตั้งแต่ PC ไปจนถึงเมนเฟรม

เอกสารนี้เป็นเอกสารลิขสิทธิ์ของมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ไม่อนุญาตให้นำไปเผยแพร่โดยไม่ได้รับอนุญาต และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การทำ downsizing คือ การพยายามใช้ เวอร์สเตชันเทคโนโลยี แทนเมนเฟรม หรือ มินิคอมพิวเตอร์ ซึ่งจุดดีก็คือ ผู้ใช้สามารถที่จะเลือกแพลตฟอร์ม (platform) ที่เห็นว่าเหมาะสมกับงานของตนได้

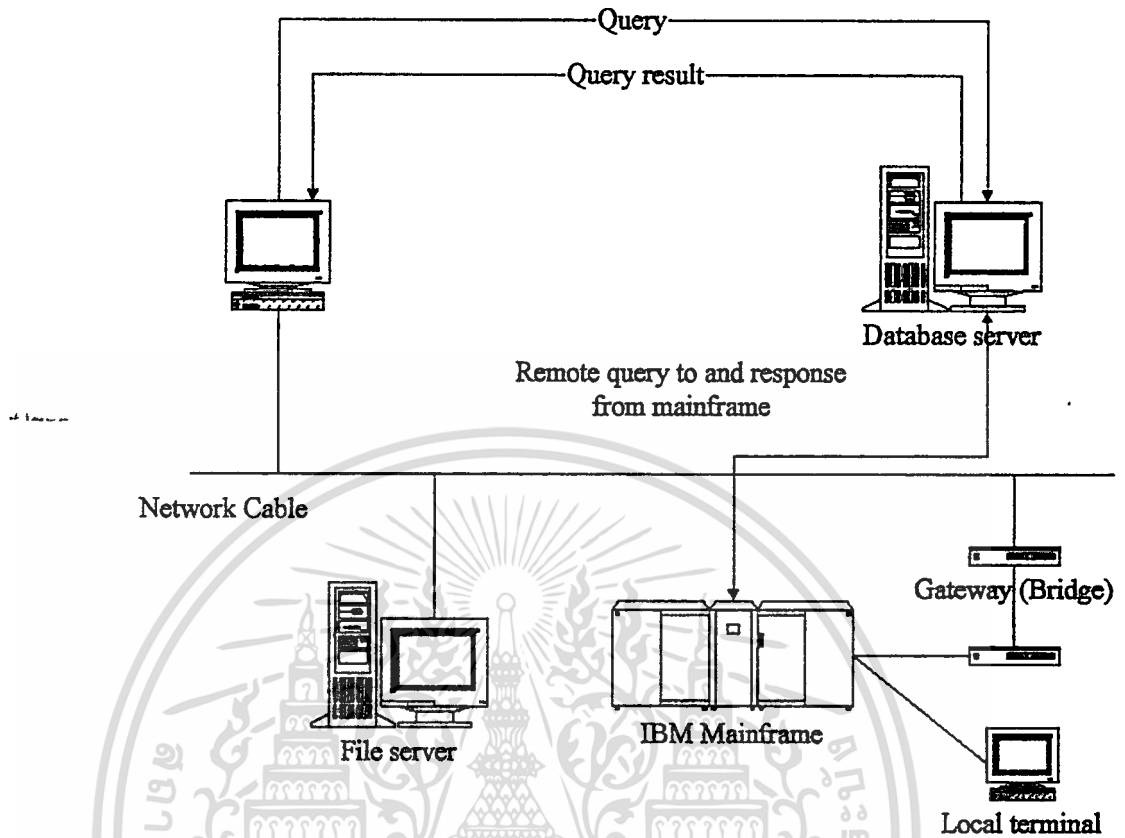
จุดประสงค์ของ client/server คือ

- การลดค่าใช้จ่าย เพราะ เวอร์สเตชัน มีค่าใช้จ่ายซึ่งถ้ามองโดยรวมจะต่ำกว่า เมนเฟรมมาก รวมทั้ง การบำรุงรักษาเมนเฟรมก็มีค่าใช้จ่ายมากกว่า
- สำหรับ CPU intensive job จะให้ throughput ที่สูงขึ้นเพราะการประมวลผล ทำได้ที่ตัว client เลยไม่ไหลลงงานทุกอย่างมาที่ตัวคอมพิวเตอร์หลัก
- แก้ปัญหาเกี่ยวกับการใช้ CPU time เพื่อการใช้ Graphic User Interface (GUI) บนระบบการประมวลผลบนคอมพิวเตอร์หลักได้ เพราะสามารถผลักภาระมาให้ตัว client จัดการเองได้
- เป็นระบบเปิด ที่เปิดโอกาสให้ผู้ใช้เลือก แพลตฟอร์ม ได้เอง

ในปัจจุบันประสิทธิภาพของ PC ในด้านต่างๆ เพิ่มขึ้น ทั้งประสิทธิภาพของ CPU , หน่วยความจำหลัก และ storage ซึ่งเป็นอีกเหตุผลที่ทำให้ PC สามารถรองรับโปรแกรมประยุกต์ที่ซับซ้อนได้ และสำหรับระบบที่ผู้ใช้มีความคุ้นเคยกับ PC อยู่แล้วการใช้ระบบ client/server จะทำให้ลดเวลาในการพัฒนาของโปรแกรมประยุกต์ลงได้และความเป็นไปได้ในการพัฒนาโปรแกรมประยุกต์ด้วยตัวเองของผู้ใช้ก็มีมากขึ้นด้วย รวมทั้งจากการที่ผู้ใช้มีความเข้าใจในระบบ การปรึกษาหรือการระบุความต้องการต่อ system specialist จะทำได้ถูกต้องชัดเจนขึ้น โปรแกรมประยุกต์ ต่าง ๆ ที่ถูกพัฒนาขึ้นมาจะสามารถจะตอบสนองความต้องการได้เป็นอย่างดี

4) Distributed Processing

ข้อมูลสามารถที่จะถูกใช้ร่วมกันระหว่างหลาย ๆ ระบบคอมพิวเตอร์หลักได้โดยผ่านทาง การเชื่อมต่อโดยตรง (network) และบนการเชื่อมต่อระยะไกล (ผ่านโครงข่ายโทรศัพท์) เป็นการ ขยายขอบเขตการขยายและแลกเปลี่ยนข้อมูลกัน เช่นระหว่างแผนกที่อยู่ไกลกันทำให้ผู้ใช้สามารถ ที่จะเข้าถึงข้อมูลข้าม site ได้ ดังแสดงในรูปที่ 2.8



รูปที่ 2.8 ภาพแสดงการประมวลแบบ Distributed

2.2 DATA DEFINITION

การกำหนด schema ของฐานข้อมูลทั้ง 3 ระดับกระทำได้โดยการใช้ภาษาพิเศษ ที่เรียกว่าภาษาสำหรับนิยามข้อมูล (data definition language) หรือเรียกย่อ ๆ ว่า DDL ผลจากการคอมไพล์ประโยคที่เขียนด้วยภาษาสำหรับนิยามข้อมูล จะทำให้เกิดตารางที่จะจัดเก็บข้อมูลในพจนานุกรมข้อมูล ซึ่งจะเก็บข้อมูลที่เกี่ยวข้องกับโครงสร้างที่ได้จากการออกแบบฐานข้อมูลนั้น ๆ และเมื่อไรก็ตามที่มีการเปลี่ยนแปลงหรือเรียกใช้ข้อมูล ระบบจัดการฐานข้อมูล จะต้องอาศัยข้อมูลของโครงสร้างจากไฟล์นี้เสมอ

-2.3 DATA MANIPULATION

การใช้ข้อมูลในระบบจัดการฐานข้อมูล แบ่งออกได้เป็น 3 ประเภท คือ

- การเรียกดูข้อมูล (retrieve data)
- การเพิ่มข้อมูลใหม่เข้าไป (insert data)
- การลบข้อมูล (delete data)

ซึ่งเราก็ทราบแล้วว่าในระดับของผู้ใช้นั้น จะไม่ทราบและไม่ต้องสนใจว่าวิธีการจัดเก็บข้อมูลจริง ๆ นั้นเป็นอย่างไร ดังนั้น การใช้ฐานข้อมูลในระดับของผู้ใช้นั้นก็สามารถกระทำได้โดยผ่านทาง ระบบจัดการฐานข้อมูล โดยใช้ภาษาสำหรับการใช้ข้อมูล (data manipulation language) หรือที่เรียกย่อ ๆ ว่า DML นั่นเอง

โดยทั่วไป แล้วเราสามารถแบ่งชนิดของ DML ออกได้เป็น 2 ชนิดคือ

- procedural ในการใช้ DML แบบนี้ ผู้ใช้เพียงแต่ระบุว่าต้องการข้อมูลอะไร โดยไม่ต้องบอกวิธีการเลย
- nonprocedural สำหรับ DML แบบนี้ ผู้ใช้เพียงแต่ระบุว่าต้องการข้อมูลอะไร โดยไม่ต้องบอกวิธีการเลย

ก่อนที่จะเราได้มีโอกาสศึกษาตัวอย่างของ DML ทั้ง 2 นี้ ผู้เขียนขอกล่าวกว้าง ๆ ไว้ก่อนว่า DML แบบ procedural ก็จะคล้าย ๆ กับการเขียนโปรแกรมด้วยภาษาชั้นสูง (ไม่ว่าจะเป็น โคบอลหรือ ปาสคาล) หรือภาษาเฉพาะของระบบจัดการฐานข้อมูลนั้น ๆ เช่น ภาษา dBASE II/III เป็นต้น ส่วน DML แบบ nonprocedural คือการใช้ภาษาธรรมชาติ ซึ่งก็จะพอเดาได้ว่าการใช้ DML แบบ nonprocedural นั้น จะมีความง่ายกว่ามาก แต่มีประสิทธิภาพ (โดยเฉพาะในแง่ของความเร็ว) สู้แบบ procedural ไม่ได้เลย ระบบจัดการฐานข้อมูล จะต้องอาศัยข้อมูลของโครงสร้างจากไฟล์นี้เสมอ

2.4 SEMANTIC DATA CONTROL

เป็นการควบคุมความถูกต้องในการจัดการข้อมูลให้มีความถูกต้องตามระดับความสามารถที่ผู้ใช้สามารถจะกระทำกับข้อมูลตามที่คุณควบคุมข้อมูลได้กำหนดไว้ให้ โดยสามารถแยกออกเป็นวิธีการต่าง ๆ คือ

- View Management
- Security Control
- Semantic Integrity Control

โดยที่แต่ละวิธีที่กล่าวมานั้นจะช่วยในการดูแลและควบคุมความถูกต้องของฐานข้อมูล โดยเฉพาะในโครงสร้างที่เป็นแบบ relational นั้น semantic data control จะช่วยทำให้บรรลุผลตามนิยามของโครงสร้างแบบ relational

การใช้ view, security constraints และ semantic data constraints สามารถที่จะสร้างกฎ (rules) ไว้เพื่อการควบคุมและบังคับใช้ได้อย่างอัตโนมัติ อย่างเช่นถ้ามี transaction ที่ละเมิดกฎที่ได้ตั้งไว้ก็อาจให้ transaction นั้นหลุด (reject) ออกจากการทำ transaction นั้นได้

การที่จะสร้างกฎไว้เพื่อใช้ในการควบคุมการจัดการข้อมูลนั้น เป็นหน้าที่ของผู้บริหารฐานข้อมูลและผู้ควบคุมฐานข้อมูลนี้จะเป็นผู้ที่จัดระดับความสามารถในการใช้ฐานข้อมูลให้กับองค์กร

การทำ semantic data control ตามวิธีต่าง ๆ ดังที่ได้กล่าวมานั้น จะใช้ภาษาสำหรับนิยามข้อมูลในการทำ เช่นเดียวกับการทำ data definition แต่กฎสำหรับ semantic data control นั้น จะถูกเก็บไว้ในพจนานุกรมข้อมูล

2.4.1 View Management

เป็นเหมือนการสร้างหน้าต่างให้กับผู้ใช้ โดยที่ผู้ใช้นั้น จะมองเห็นเฉพาะบางส่วนของฐานข้อมูล คือเฉพาะส่วนที่ได้เจาะช่องหน้าต่างไว้ และสามารถที่จะทำการ insert, update, delete กับ view นี้ได้เสมือนกับทำกับฐานข้อมูลโดยตรง โดยเมื่อกระทำ query อันใดกับ view แล้ว ที่ฐานข้อมูลนั้นจะเปลี่ยนแปลงตามไปด้วย

วิธีการของ view นั้น ก็เป็นการควบคุมความถูกต้องของฐานข้อมูลโดยการกำหนดส่วนที่จะให้ผู้ใช้เข้าไปจัดการกับข้อมูลในส่วนที่ได้กำหนด view ไว้ให้เท่านั้น โดยผู้ใช้นั้นจะไม่เห็นหรือไม่รับรู้ถึงข้อมูลในส่วนอื่น ๆ ที่ไม่ได้ view ไว้

2.4.2 Security Control

เป็นการป้องกันการเข้าไปใช้ฐานข้อมูลของผู้ใช้ที่ไม่มีสิทธิ์ในการเข้าไปใช้ฐานข้อมูลนั้น โดยการกำหนดสิทธิในการที่จะใช้ฐานข้อมูลนั้น ซึ่งเป็นการกำหนดว่าสามารถทำ operation ใด ๆ ได้บ้าง อย่างเช่น อาจจะ select ได้อย่างเดียว ไม่สามารถ update หรือ delete ได้ นั่นก็คือการ grant privilege ซึ่งก็เป็นคำสั่งในการทำ semantic data control ของภาษา SQL ซึ่งสามารถระบุได้ว่าผู้ใช้แต่ละคนสามารถที่จะทำ operation ใดได้บ้าง

ในการทำ security control นั้น อาจจัดแบ่งให้ผู้ใช้ออกเป็นกลุ่ม ๆ เช่น กลุ่มของผู้ใช้ข้อมูลเพื่อดูผลอย่างเดียว หรือผู้ใช้ที่ต้องสร้างหรือ แก้ไข เปลี่ยนแปลงข้อมูล เป็นต้น เพื่ออำนวยความสะดวกในการควบคุมความถูกต้อง และมีให้เกิดความสูญเสียขึ้นกับข้อมูล

2.4.3 Semantic Integrity Control

เป็นการควบคุมความถูกต้องของข้อมูล ซึ่งสามารถแบ่งออกได้เป็น 2 กลุ่มคือ

- Static integrity : เป็นความถูกต้องของลักษณะของข้อมูล (correctness state of data) เช่น 1 เดือนต้องมีไม่เกิน 31 วัน หรือผู้ชายไม่สามารถลาคลอดได้ เป็นต้น
- Dynamic integrity : เป็นความถูกต้องของการเปลี่ยนสถานะของข้อมูล (correctness state of transaction) เช่น ความถูกต้องในการ update ข้อมูล

2.5 TRANSACTION MANAGEMENT

เป็นส่วนที่อธิบายเกี่ยวกับการแก้ปัญหาในการจัดการลำดับเหตุการณ์ในระบบ multuser เมื่อระบบหรือสื่อเกิดการขัดข้อง ความถูกต้องสมบูรณ์ของข้อมูลอาจถูกแก้ไขโดยการเปลี่ยนแปลงข้อมูลให้ใหม่อยู่เสมอในการดูแลความถูกต้องของข้อมูลการทำการแก้ไขจะถูกรวมเป็นกลุ่มใหญ่ เรียกว่า transaction ซึ่งควบคุมโดยระบบจัดการฐานข้อมูล transaction เป็นส่วนที่จะทำงานในระบบจัดการฐานข้อมูล สร้างความสมบูรณ์แต่อยู่ในฐานข้อมูลที่แตกต่างกัน transaction มีลักษณะเป็น atomic คือจะทำงานได้อย่างสมบูรณ์ หรือยกเลิกการทำงาน concurrency control และ recovery จะเป็นตัวจัดการ transaction

2.5.1 Concurrency Control

หมายถึง การควบคุม transaction ที่ทำงานพร้อม ๆ กัน และมีผลกระทบซึ่งกันและกัน โดยจะป้องกันไม่ให้เกิดความไม่ถูกต้องตรงกันของฐานข้อมูล โดยยึดหลักการที่ว่า serializability โดยสามารถใช้กลไกในการควบคุมดังต่อไปนี้

1) Locking Protocol

เป็นชุดของกฎซึ่งเป็นสภาวะเมื่อ transaction อาจจะมี lock และ unlock แต่ละ data item ในฐานข้อมูลซึ่งสามารถแบ่ง lock ได้เป็นหลายประเภทเช่น exclusive lock , (read lock) , share lock (write lock) โดยจะตรวจสอบว่าสามารถ lock resource ได้หรือไม่โดยดูจาก compatibility matrix

2) Timestamp-ordering Protocol

ทำหน้าที่เลือกลำดับการทำงานระหว่าง 2 transaction ดังนั้นถ้า timestamp ของ T_i จะเล็กกว่า timestamp ของ T_j schedule จะเป็นไปตามลำดับที่จะทำให้เกิด serializability คือ T_i จะปรากฏก่อน T_j และถ้าไม่เป็นไปตามลำดับนี้ ก็จะ rollback transaction กลับไปที่จุดเดิมจะเห็นว่าลำดับที่ทำให้เกิด serializability จะพิจารณาจาก timestamp

3) Validation Technique

เป็นวิธีควบคุม concurrency ในกรณีที่ transaction ส่วนใหญ่เป็น transaction ที่ read-only ดังนั้นอัตราการขัดแย้งระหว่าง transaction จะต่ำ วิธีนี้จะแบ่ง life time ของ transaction เป็น 3 phase คือ

- 1) Read phase เป็นค่าของ data items ต่าง ๆ จะถูกอ่านและเก็บไว้ในตัวแปร
- 2) Validation phase ตรวจ constraint ว่าถ้า write แล้วจะขัดแย้งหรือไม่
- 3) Write phase การ Write ใน phase ก่อนหน้านี้จะ write ลงตัวแปรชั่วคราวนั้น ใน phase นี้จะแก้ไขลงฐานข้อมูลจริง ๆ หรือไม่ก็ rollback หากไม่เป็น serializability

4) Multiversion scheme

แต่ละ operation ของ write (Q) จะสร้าง version ใหม่ ๆ ของ Q และเมื่อมี operation read (Q) ระบบก็จะเลือก version ใด version หนึ่งให้ read โดยจะเลือก version ที่ทำให้เกิดเป็น serializability

5) Multiple Granularity

จะยอมให้ data items มีขนาดต่าง ๆ และกำหนด hierarchy ของ data items ซึ่งจะแสดง hierarchy ของ data items ให้อยู่ในรูปของ tree จะยอมให้มีการ lock โดยจะพิจารณาตามลำดับ root-to-leaf ในขณะที่จะ unlock ตามลำดับ leaf-to-root orders

กลไกนี้จะใช้หลักการ lock-base protocol แต่เพิ่ม Locking Primitive คือมี

- exclusive lock (X-lock)
- shared lock (S-lock)
- intended exclusive lock (IX-lock)
- shard intended exclusive lock (SIX - lock)

และการ lock ของแต่ละ transaction จะทำได้โดยใช้การตรวจสอบตาราง compatibility matrix ว่าสามารถ lock ได้หรือไม่

2.5.2 Backup and Recovery

เนื่องจากระบบคอมพิวเตอร์ก็เหมือนกับระบบทางอิเล็กทรอนิกส์ อื่น ๆ ตรงที่มีโอกาสที่จะเกิด failure ได้ซึ่งคำว่า failure นั้นก็มีหลายอย่างเช่น disk crash , ไฟดับ , software error และอื่น ๆ ซึ่งก็ล้วนแล้วแต่มีผลต่อการหายของ information ในระบบจัดการฐานข้อมูล ดังนั้นการ recovery จึงเป็นส่วนหนึ่งของ ระบบจัดการฐานข้อมูล ที่มีหน้าที่รับผิดชอบต่อการจับว่ามี failure และการกู้ข้อมูลกลับมาให้อยู่ในสถานะที่ consistent เหมือนเดิมก่อนการเกิด failure

Transaction คือ กลุ่มของงานที่ยอมให้มีการละเมิด integrity constraint ได้และเมื่อเสร็จงานแล้ว integrity constraint ต้องถูกต้องหรือพูดง่าย ๆ ก็คือ transaction คือกลุ่มของ operations ที่แสดง ฟังก์ชันเดียว ๆ ในโปรแกรมประยุกต์ของฐานข้อมูล

Transaction State มีดังนี้คือ

Active	คือ initial state
Partially committed	คือ execute statement สุดท้ายเรียบร้อยแล้ว
Failed	คือ ไม่สามารถดำเนินการ execute อย่างปกติต่อไปได้
Aborted	คือ transaction ถูก roll back และ state ของฐานข้อมูลถูกกู้กลับมาที่สภาวะก่อนที่จะเริ่มต้น transaction.
Committed	คือ การ successful completion

ในการ recovery จะมีวิธีดังนี้

1) Log-Base Recovery แบ่งเป็น

- Deferred Database Modification
- Immediate Database Modification

2) Shadow Paging

3) ใช้การ Copy Database เป็น backup เก็บเอาไว้

1) Log-Based Recovery

ในการใช้วิธีนี้จะต้องมีสิ่งที่เกี่ยวข้องคือ Database Log หรือ Log file ซึ่งเป็นที่ๆเอาไว้เก็บสิ่งต่างๆ ดังนี้

- Transaction name เป็นชื่อ unique name ของ data item ที่ทำ write operation
- Data item name เป็นชื่อ unique name ของ data item ที่ถูก write
- Old value คือ ค่าของ data item ก่อนการ write
- New value คือ ค่าของ data item หลังการ write

ชนิดของ log record มีดังนี้

- $\langle T_i, \text{start} \rangle$ เป็น record ที่บอกว่า transaction T_i เริ่มต้น
- $\langle T_i, X_i, V_1, V_2 \rangle$ หมายถึง T_i กำลัง write บน data item X_j และ X_j มีค่า V_1 ก่อน write และ V_2 หลังจาก write แล้ว
- $\langle T_i, \text{commit} \rangle$ หมายถึง T_i commit แล้ว ซึ่งจะมี write ahead protocol ดังนี้ ก่อนที่จะมีการแก้ไขฐานข้อมูล จะต้องมีการสร้าง log record เก็บเอาไว้แล้ว ถึงยอมให้มีการเปลี่ยนแปลงใดๆ ดังนั้นเราจึงสามารถที่จะ undo การเปลี่ยนแปลงใดๆ ที่ได้เกิดขึ้นกับฐานข้อมูลได้โดยใช้ค่า old value ที่เก็บเอาไว้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่ควรเผยแพร่โดยไม่ได้รับอนุญาต
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Log-based recovery แบ่งเป็น 2 วิธีดังนี้

1.1) Deferred Database Modification

หลักการของวิธีนี้คือเฉพาะ AII คือ log record เก็บเฉพาะค่า new value หลังการ write เพียงอย่างเดียวเท่านั้น ไม่มี BIJ (ค่า old value ก่อนการ write)

วิธีนี้ข้อมูลบนดิสก์จะไม่โดนแตะต้องเลยจนกว่าจะ write commit แล้วจึงค่อยถ่ายข้อมูลลงดิสก์

ในการ recovery จะใช้ procedure ดังนี้

- Redo คือ transaction ใดที่ commit ไปแล้วก่อนที่ system จะ crash transaction นั้นก็จะถูก Redo โดยเอาค่า new value ที่เก็บเอาไว้ใน log record ใส่ลงไป ส่วน transaction ใดที่ยังไม่ commit ก็ไม่ต้อง Redo คือไม่ต้องทำอะไรเพราะยังไม่มีการ write ลงบนดิสก์

ข้อดีของ Deferred Database Modification

- 1) ไม่ต้องมี BIJ ทำให้ประหยัดเนื้อที่ใน memory
- 2) เหมาะกับ transaction ขนาดเล็ก ๆ เพราะเร็ว คือรอให้มีการแก้ไขฐานข้อมูลหลายๆ ครั้งแล้วก็ค่อย commit ที่นิ่งแล้วค่อย ๆ ถ่ายข้อมูลลงบนดิสก์ ซึ่งจะให้ข้อดีเพิ่มขึ้นอีกคือ save I/O operation ด้วย

ข้อเสียของ Deferred Database Modification

- 1) ในกรณีที่เป็น transaction ใหญ่ วิธีนี้จะไม่เหมาะเพราะจะเกิดการเปลี่ยนแปลงจำนวนมากซึ่งเราจะต้องรอจนกว่า commit แล้วจึงจะได้ถ่ายข้อมูลลงดิสก์
- 2) การ recovery จะต้องทำทั้ง transaction เนื่องจากมี AII อย่างเดียว
- 3) ใช้เนื้อที่ใน buffer มากดังนั้นไม่เหมาะกับ transaction ขนาดใหญ่

1.2) Immediate Database Modification

หลักการของวิธีนี้คือใน log record จะเก็บทั้ง AUI และ BIJ คือ ทั้ง old value และ new value

Procedure ของการ recovery มีดังนี้

- Undo คือ ขณะ crash ถ้า transaction ใดที่ยังไม่ได้ commit ก็จะนำค่า old value ที่เก็บอยู่ใน log record นั้นไป
- Redo คือ ขณะ crash ถ้า transaction ใดที่ commit ก็จะนำค่า new value ที่เก็บอยู่ใน log record นั้นไป

ข้อดีของ Immediate Database Modification

- 1) ไม่ต้องเก็บข้อมูลใน buffer มาก
- 2) เหมาะกับ transaction ขนาดใหญ่เพราะถ้าหากมีการ write log record ลงไปแล้วก็สามารถย้ายข้อมูลลงดิสก์ไปได้เลย ไม่ต้องรอจน transaction commit ก็ได้ ทำให้ย้ายข้อมูลเสร็จเร็วกว่าใช้ deferred Mode ที่ต้องรอจน commit แล้วจึงย้ายข้อมูลได้ ที่เป็นเช่นนี้เพราะมีการเก็บทั้ง AUI และ BIJ ทำให้สามารถ undo ข้อมูลของ transaction ที่ยังไม่ได้ commit ได้

ข้อเสียของ Immediate Database Modification

- 1) เปลือง I/O operation เพราะมีการ write log record บ่อย ทำให้เสียเวลาจึงทำให้ไม่เหมาะกับ transaction ขนาดเล็ก ๆ
- 2) เปลืองเนื้อที่ในการเก็บค่า AUI

ในการ Recovery แบบ log-based นี้ยังมีจุดสำคัญอีกจุดหนึ่ง คือจุด checkpoint

Checkpoint คือ เวลาที่เกิด action ต่อไปนี้

- log record ทั้งหมดที่ยังอยู่ใน main memory จะย้ายลง stable storage
- ข้อมูลที่ถูกแก้ไขแล้วอยู่ใน buffer block จะถูกย้ายลง stable storage

Checkpoint time นั้นแล้วแต่เราจะเลือกจะให้ถี่แค่ไหนขึ้นกับ volume ของ transaction ถ้าเป็น transaction ขนาดใหญ่ ๆ ก็ไม่ควรให้ถี่นักเพราะต้องเสียเวลาย้ายข้อมูลจาก buffer ลง ดิสก์ซึ่งมีจุดให้พิจารณาดังนี้

- ถ้า Tc ห่างจะเปลืองเนื้อที่ใน main (ข้อมูลใน buffer ที่ยังไม่ย้ายลง ดิสก์ มาก)
- ถ้า Tc ห่างอาจทำให้เกิด bottle neck คือต้องใช้เวลาในการย้ายข้อมูลลง ประโยชน์ของ checkpoint คือ recovery time จะเร็ว

Shadow Paging

หลักการของวิธีนี้คือ จะมีการ page table 2 อัน คือ

- Current page table คือ ตารางที่เปลี่ยนแปลงไปตาม write operation ใน transaction
- Shadow Page Table จะถูกเก็บเป็น shadow page table ในจุดที่ทำการ commit แล้วหลังจากนั้นพอเริ่ม transaction ใหม่ก็จะมี current page table ใหม่ซึ่งมีสถานะเริ่มต้นเหมือน shadow page table ทุกอย่าง

ข้อดีของวิธี Shadow Page Table เท่านั้น

- 1) Recovery time จะเร็วกว่ามาก ๆ เพียงแค่ยกเลิก current page table แล้วเปลี่ยนไปใช้ shadow page table เท่านั้น
- 2) เหมาะกับงานขนาดเล็ก ๆ ที่ต้องการ response time ดี ๆ เพราะในการ commit ก็เป็นการเขียนข้อมูลลงดิสก์เท่านั้น

งานที่เหมาะสมกับ shadow page table ได้แก่ งานประเภท OnLine Transaction Processing (OLTP)

ข้อเสียของวิธี shadow page table

- 1) ดิสก์กระจัดกระจายเพราะ update แต่ละครั้งต้องสร้าง page table ใหม่อาจทำให้ table กระจัดกระจายไม่อยู่ใน cylinder เดียวกัน ทำให้ประสิทธิภาพต่ำลง
- 2) ไม่เหมาะกับงาน transaction ขนาดใหญ่เพราะการ commit หมายถึงการถ่ายข้อมูลลงดิสก์จริง ๆ ดังนั้น commit อาจช้ากว่าการทำ log based
- 3) ไม่มี A/J ดังนั้นการ backup ต้องทำทั้ง volume

2.6 QUERY PROCESSING

วิธีในการทำ query และรูปแบบการเก็บข้อมูลเป็นปัจจัยสำคัญในด้านการจัดการฐานข้อมูล ซึ่งประกอบด้วย

2.6.1 Data Independence

Data independence ก็คือการทำที่เราเปลี่ยน storage structure (ในทาง physical นั้น ข้อมูลถูกเก็บไว้อย่างไร) หรือ การเปลี่ยนวิธีในการเข้าถึงได้โดยไม่มีผลต่อโปรแกรมประยุกต์ เช่น สามารถเปลี่ยนจาก index file ไปเป็น hash-addressed file โดยไม่ต้องมีการทำ major modification ในโปรแกรมประยุกต์เรา

ความจำเป็นที่ต้องเป็น data independent มีอยู่ 2 ประการหลักคือ

- 1) ในโปรแกรมประยุกต์ที่ต่างกันอาจมีมุมมอง (view) ที่ต่างกันบนข้อมูลเดียวกัน เช่น มีโปรแกรมประยุกต์ 2 อันคือ A,B ซึ่งแต่ละอันมี ไฟล์ของตนเองและใช้ฟิลด์ “customer balance” ทั้งคู่ แต่ A จัดเก็บ value นี้เป็น decimal และ B เก็บเป็น binary ซึ่งระบบจัดการฐานข้อมูล ควรจะมีความสามารถในการจัดทำ conversion สำหรับการ represent รูปแบบที่ต่างกันอย่างนี้ เช่นสมมติถ้าตัดสินใจที่จะเก็บเป็นแบบ decimal แล้วในการที่ B จะเข้าถึงก็จะต้องมีการ conversion แปลงจาก binary เป็น decimal เป็น binary
- 2) ผู้บริหารฐานข้อมูล จะต้องมีความอิสระที่จะเปลี่ยน storage structure หรือวิธีในการเข้าถึงโดยไม่ต้องแก้ไขโปรแกรมประยุกต์ที่มีอยู่ เช่น อาจมีข้อมูลชนิดใหม่เพิ่มเข้าไปในฐานข้อมูล หรือ มีการปรับปรุง standard ใหม่ หรือมีการเปลี่ยนแปลง application priority หรือมีการเพิ่ม storage device ชนิดใหม่ ๆ เข้าไป ซึ่งก็เป็นหน้าที่ของ ระบบจัดการฐานข้อมูล ที่จะต้องตรวจจบการเปลี่ยนแปลงนี้แล้วจัดการดูแลให้

Data independence มี 2 ประเภท คือ

1) Physical data independence คือความเป็นอิสระของการเปลี่ยนแปลงดังนี้

- Representation of numeric data เช่น เป็นการเปลี่ยนจากการเก็บแบบ decimal เป็นแบบ binary หรือจาก real เป็น complex
- Representation of numeric data เช่น เป็น EBCDIC , ASCII
- Unit for numeric data เช่น เปลี่ยนจาก inch เป็น centimeter
- Data Coding เช่นเปลี่ยนจากเก็บเป็น green , blue , red แทนด้วย 1,2,3 โดย 1 = green , 2 = blue , 3 = red

2) Logical data independent คือการเปลี่ยนแปลง การเพิ่มคอลัมน์ หรือ สร้างตารางใหม่ไม่มีผลต่อ program ที่มีอยู่เดิม

2.6.2 Optimization Performance

ในเรื่องของการ optimization จะเป็นการเลือกวิธีในการประเมิน relational expression ที่มีประสิทธิภาพซึ่งประเด็นสำคัญของเรื่องนี้จะมุ่งไปที่การทำ query optimization

Query optimization จะเป็นการเลือกวิธีในการเข้าถึงข้อมูลให้เร็วที่สุดโดยจะมีตัว query optimizer เอาไว้เป็นตัวเลือกในการเข้าถึงข้อมูลจริงๆ ในระดับ physical อาจมองเป็น expert system ตัวหนึ่งเพราะจะใช้กฎต่างๆ หากทางเข้าถึงข้อมูลโดย query optimizer จะรับ parse tree มาจาก query parser ซึ่งทำหน้าที่ตรวจสอบไวยากรณ์ของภาษารฐานข้อมูลแล้วสร้างเป็น algebra tree ส่งมาให้ query optimizer

ในการเลือกใช้ ระบบจัดการฐานข้อมูล เราต้องพิจารณาประเด็นที่สำคัญ 2 ประการคือ

1) ระบบจัดการฐานข้อมูล สามารถจัดการ query แบบ compile ได้หรือไม่ คือการมีฟังก์ชันของ query ที่มีการใช้งานบ่อยๆหลาย ๆรูปแบบ มาทำเป็นฟังก์ชันเก็บไว้โดยถ้ามีรูปแบบของ query ที่เก็บไว้ก็จะสามารถทำการเรียกใช้ query รูปแบบนั้นได้โดยส่งเพียงพารามิเตอร์ผ่านเข้าไปเท่านั้น ซึ่งการ Precompiled จะมีผลดีตรงที่เร็วไม่ต้องเสียเวลา interpret

2) ระบบจัดการฐานข้อมูล สามารถจัดการ query แบบ interpret เป็นวิธีการ query แบบธรรมดาที่ทุก ๆ ระบบจัดการฐานข้อมูลมี คือ จะเป็นการรับชุดของ query เข้ามายาว ๆแล้วทำการ interpret ครั้งต่อครั้งไม่มีการเก็บเอาไว้เป็นฟังก์ชัน ซึ่งมีวิธีนี้จะมีข้อดีที่ไม่ต้องเปลืองเนื้อที่ในหน่วยความจำเพื่อรักษาฟังก์ชันนั้นไว้

วิธีในการทำ optimization มีด้วยกัน 2 วิธีคือ

2.1) Cost base : จะเป็นการคำนวณเส้นทางในการเข้าถึงข้อมูลที่สามารถเข้าถึงได้เร็วที่สุด โดยอาศัยการเก็บสถิติ ของการใช้ I/O ในแต่ละเส้นทางที่ใช้ในการเข้าถึงข้อมูล มาเป็นตัวตัดสินใจในทางเลือกเส้นทางที่จะเข้าถึงข้อมูล

2.2) Rule base : จะเป็นการนำเอา query ที่ได้รับ มาทำการจัดรูปแบบ ไวยากรณ์ เสียใหม่ แต่ยังคงมีความหมายที่คงเดิม (semantic) และเป็น เส้นทางที่สามารถเข้าถึงข้อมูลได้เร็วขึ้น ตามกฎที่ได้ตั้งไว้

2.7 ADDITIONAL TOOLS

เครื่องมือต่าง ๆ ที่ติดมาหรือใช้ร่วมกับตัวระบบจัดการฐานข้อมูล เพื่อเป็นการอำนวยความสะดวกให้กับผู้ใช้

- Report generator : เป็นเครื่องมือที่ใช้ในการสร้างรายงานจากข้อมูลที่มีอยู่ ซึ่งในรายงานนอกจากจะมีตัวหนังสือ และ ตารางแล้วนั้น ควรมีความสามารถที่จะสร้างกราฟ หรือ ใส่รูปภาพเข้าไปได้ด้วย
- Graphic interface : เป็นการติดต่อกับผู้ใช้แบบ graphic mode ซึ่งสามารถที่จะใช้งานได้สะดวก และการที่จะแสดง object ต่าง ๆ เช่น กราฟ, รูป, ตัวหนังสือ ฯลฯ สามารถทำให้ดูง่ายและสวยงามกว่า
- Screen manager : ความสามารถที่จะสร้างหน้าจอ เพื่อออกแบบหน้าจอ ไว้สำหรับติดต่อกับผู้ใช้ เพื่อความสะดวกในการใช้งาน
- Application generator : เป็นเครื่องมือที่ใช้สร้างโปรแกรมขึ้นมา เช่น case tools หรืออาจเป็นเครื่องมือที่ใช้ในการพัฒนาโปรแกรมขึ้นมาไว้ใช้กับ ระบบจัดการฐานข้อมูล เช่น โปรแกรมภาษา 4GL

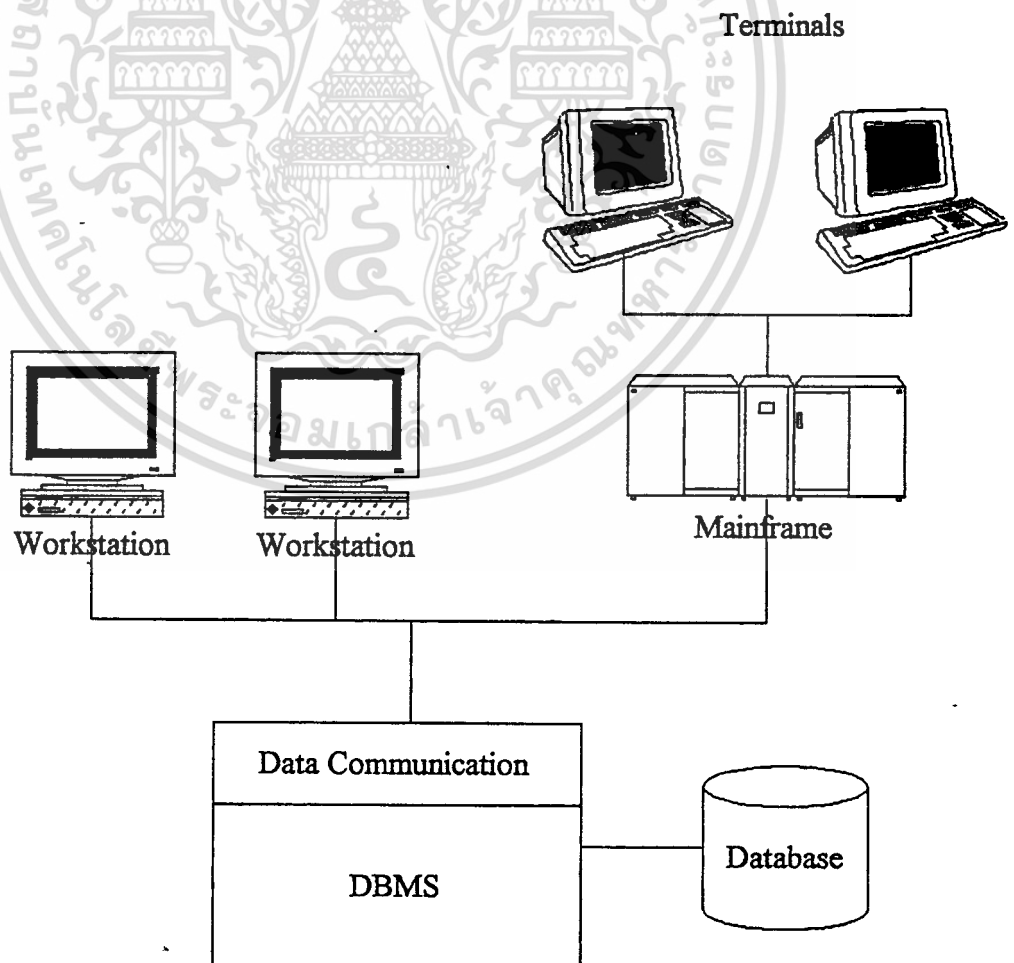
2.8 DISTRIBUTED DATABASE MANAGEMENT

การจัดฐานข้อมูลแบบกระจายนี้จะมีด้วยกัน 2 วิธีคือ

- Remote database
- Distribute database

1) Remote database

จากรูปที่ 2.9 ฐานข้อมูลจะถูกเก็บอยู่ที่คอมพิวเตอร์ ที่ไม่ใช่ของผู้ใช้ฐานข้อมูลนั้นก็คือ remote database ซึ่งผู้ใช้นั้นจะทราบว่า remote database นั้นอยู่ที่ใด เพื่อที่จะสามารถเข้าไปเอาข้อมูลนั้นมาใช้ได้ เพราะฉะนั้น อุปกรณ์ที่ใช้ในการสื่อสาร จึงจำเป็นสำหรับการที่จะใช้ในการเข้าถึงข้อมูลที่ remote database แต่ผู้ใช้นั้นก็สามารถที่จะใช้ฐานข้อมูลของตัวเอง (local database) ได้ด้วย และในอนาคตอันใกล้ เทคโนโลยีของไมโครคอมพิวเตอร์ จะสามารถสนับสนุนการใช้เครื่อง เวิร์คสเตชัน มาเป็นตัวที่ใช้จัดการฐานข้อมูลของตัวเอง โดยมีเครื่องเมนเฟรมเป็นตัวจัดการ remote database และใช้เครื่องไมโครคอมพิวเตอร์สำหรับการเข้าถึงข้อมูล โดยใช้ ระบบจัดการฐานข้อมูล ตัวเดียวกันกับที่ใช้ในเครื่องเมนเฟรม การติดต่อสื่อสารกับเวิร์คสเตชัน นั้นจะถูกกระทำผ่าน local network



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษา รูปที่ 2.9 ภาพแสดง remote database โยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2) Distributed database

ฐานข้อมูลแบบกระจาย เป็นการทํางานร่วมกันระหว่างฐานข้อมูลบนเครื่องคอมพิวเตอร์ที่ต่างเครื่องกัน เรียกว่า sites ซึ่งจะทำการติดต่อสื่อสารกันผ่านทาง network สิ่งที่แตกต่างกันจากแบบ remote database นั่นก็คือ ผู้ใช้ไม่จำเป็นต้องทราบว่าฐานข้อมูลนั้นอยู่ที่ใดบ้าง จะมองเสมือนว่าฐานข้อมูลนั้นเป็นฐานข้อมูลเดียวกัน (local transparency)

ถ้าต้องการให้ฐานข้อมูลมีความมั่นคงเพิ่มขึ้น หรือทำการสำรองข้อมูลไว้เพื่อเปรียบเทียบ นั้น อาจให้เกิดความซ้ำซ้อนของข้อมูลเกินขึ้นได้ (replication transparency) ส่วนประกอบของ distributed database นั้นประกอบไปด้วยอุปกรณ์ของแต่ละ site, อุปกรณ์ที่ใช้ในการติดต่อสื่อสาร, local DBMS และ distributed DBMS (DDBMS) ซึ่ง DDBMS จะมีคุณสมบัติดังนี้ สามารถที่จะจัดการ global data dictionary แบบ local และ replication transparency

- สามารถที่จะจัดการ global data dictionary แบบ local และ replication transparency
- สามารถทำ data definition และ data control แบบกระจายได้
- สามารถทำ query processing , query optimization และ remote access แบบกระจายได้
- สามารถทำ transaction management , concurrency control , recovery และ commit แบบกระจายได้

Distributed database นั้นสามารถแยกออกได้เป็น

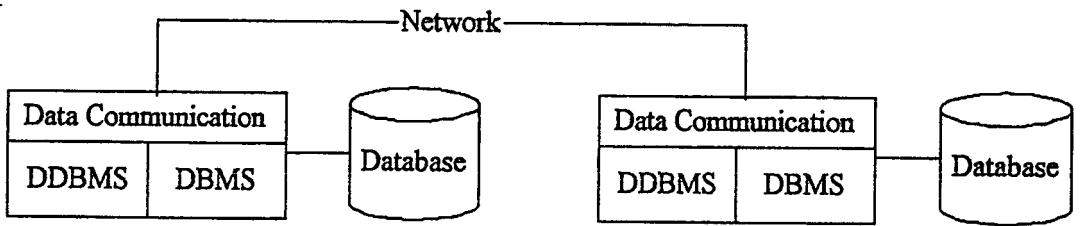
- Homogeneous distributed database
- Heterogeneous distributed database

2.1) Homogeneous distributed database

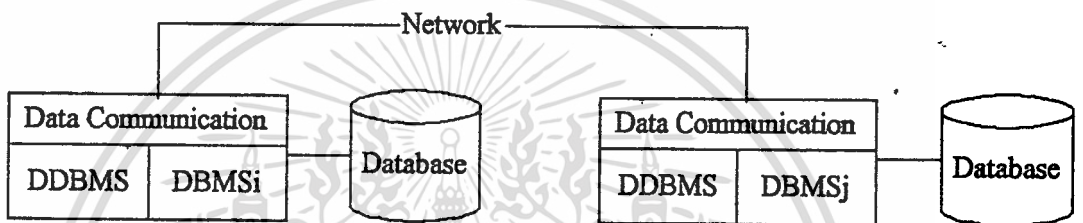
ทุก ๆ local database จะถูกจัดการโดย ระบบจัดการฐานข้อมูล เดียวกัน ซึ่งเป็นวิธีที่ง่าย มีประสิทธิภาพและช่วยในการทำการประมวลผลแบบ parallel processing ดังแสดงในรูปที่ 2.10a

2.2) Heterogeneous distributed database

แต่ละ local database จะถูกจัดการโดย ระบบจัดการฐานข้อมูล คนละตัวกัน เหมาะจะใช้กับระบบที่มีระบบจัดการฐานข้อมูลคนละ model กัน เช่นเป็นแบบ relational model แต่ ระบบจัดการฐานข้อมูล เป็นแบบ hierachical model ดังแสดงในรูปที่ 2.10b



(a)



(b)

รูปที่ 2.10 ภาพ (a) เป็น homogeneous distributed database.

ภาพ (b) เป็น heterogeneous distributed database.

การวิเคราะห์ฐานข้อมูล ORACLE version 7

3.1 ARCHITECTURE

ORACLE มี process ด้วยกัน 2 ชนิดคือ

3.1.1 User/Client processes : เป็นกระบวนการทำงานของผู้ใช้

3.1.2 ORACLE processes ประกอบไปด้วย

1) Server process : ทำหน้าที่จัดการติดต่อสื่อสารระหว่าง server process และ user process เพื่อทำตามที่ user process ต้องการโดย ORACLE มีข้อกำหนดเกี่ยวกับรูปแบบหลายรูปแบบโดยจะแบ่งตามจำนวนของผู้ใช้ต่อ server process แบ่งเป็น

1.1) Dedicated Server Configurations : server process จะรองรับ user process เพียง 1 user process

1.2) Multi-Threaded Server Configurations : user process หลาย process ใช้ server process ร่วมกันเพื่อลดจำนวน server process ให้น้อยที่สุดและมีประสิทธิภาพมากที่สุด โดย user process และ server process สามารถวิ่งอยู่บนอุปกรณ์ที่ต่างกัน ซึ่ง user process และ server process ต้องแยกจากกัน เป็นการช่วยให้

- ลด overhead จาก context switching เนื่องจาก server process ลดลงและยังเป็นการขจัดปัญหาเรื่อง bottleneck ของ CPU, I/O, หน่วยความจำและ operating system
- เพิ่มจำนวนผู้ใช้ต่อ node
- จำนวนของ server process จะเป็นแบบ dynamic กล่าวคือสามารถเปลี่ยนแปลงได้ตามขนาดของงาน โดยไม่มีผลกระทบกระเทือนต่อการใช้งาน

3.2 SEMANTIC DATA CONTROL

3.2.1 View Management

ORACLE จะยอมให้ผู้ใช้สามารถที่จะทำการแทรกแก้ไข หรือลบผ่าน view ได้โดยมีกฎข้อบังคับคือ

- ถ้า view ที่ถูกสร้างขึ้นจาก join operation, distinct operation หรือ group by clause จะไม่สามารถแทรก แก้ไข หรือ ลบ ผ่าน view ได้
- ถ้า view ถูกระบุโดยมี with check option row จะไม่สามารถ แก้ไข หรือ ลบ ผ่าน view ได้
- ถ้า view ถูกระบุให้เป็น not null column (คอลัมน์ที่มีค่าเสมอ เป็น Null ไม่ได้) ที่ไม่มี default clause ใน base table จะไม่สามารถแทรก row ผ่านทาง view ได้
- ถ้า view ถูกสร้างโดย expression จะไม่สามารถแทรก หรือแก้ไขผ่านทาง view ได้โดยรวมแล้ว ORACLE สามารถควบคุมการแก้ไขผ่านทาง view ให้ถูกต้องได้ โดยจะไม่อนุญาตให้มีการแก้ไขตามกรณีค่ากล่าวมาแล้วข้างต้น

3.2.2 Data Integrity

Integrity constraint สามารถแบ่งได้เป็น 2 แบบ คือ

1) Entity integrity ซึ่งแบ่งเป็น

- DEFAULT integrity constraint : สำหรับคอลัมน์ที่ไม่ได้กำหนดระหว่างที่ insert
- NOT NULL integrity constraint : เป็น integrity constraint ซึ่งใช้กำหนดให้คอลัมน์ที่ต้องการในตาราง จะต้องไม่เป็นค่า NULL
- UNIQUE integrity constraint : เป็น integrity constraint ซึ่งใช้กำหนดเมื่อไม่ต้องการให้ rows 2 rows ใด ๆ ในตารางมีค่าเดียวกันในคอลัมน์
- CHECK integrity constraint : เป็น integrity constraint ซึ่งกำหนดคอลัมน์หรือชุดของคอลัมน์ที่ต้องการให้เป็นไปตามเงื่อนไขที่กำหนด โดยตรวจสอบทุก ๆ rows ในตารางแล้วถ้าพบว่าเงื่อนไขนั้นเป็นเท็จ จะระงับ statement นั้นและจะหยุดการเปลี่ยนแปลงข้อมูล

2) Referential Integrity ซึ่งแบ่งเป็น

- PRIMARY KEY integrity constraint

: แต่ละตารางในฐานข้อมูลจะมี constraint นี้ได้เพียง constraint เดียว

: rows ในตารางจะถูกกำหนดระบุและอ้างอิงโดย primary key ซึ่งต้องมี

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้ ความถูกต้องตาม constraint นี้ อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- FOREIGN KEY integrity constraint

: เป็น integrity constraint ที่ทำหน้าที่รักษาความสัมพันธ์ระหว่างค่าของ primary key ใด ๆ จะต้องมีความ match กับค่าของ foreign key ในอีกตารางหนึ่งซึ่งการ match ก็สามารแบ่งได้เป็น

- match full
- match partial
- match null

Actions ซึ่งกำหนดโดย referential integrity constraint แบ่งเป็น

2.1) Update และ delete restrict : ค่าของคีย์ที่เป็นตัวอ้างอิงให้กับคีย์อื่น ๆ จะถูกเปลี่ยนแปลงหรือลบไม่ได้ เช่น primary key จะถูกเปลี่ยนแปลงหรือลบไม่ได้เพราะยังมีค่านั้นปรากฏอยู่เป็น foreign key ในตารางอื่น ๆ

2.2) Delete cascade : ถ้า row ที่มีคีย์ที่เป็นตัวอ้างอิงให้กับคีย์อื่น ๆ ถูกลบแล้วทุก rows ที่มี foreign key ที่อ้างอิงจากคีย์นั้นจะต้องถูกลบไปด้วย

Database Trigger

ORACLE ยอมให้ผู้บริหารฐานข้อมูล และผู้ใช้สามารถควบคุม integrity ได้โดยไม่จำเป็นต้องประกาศแบบที่เรียกว่า “non-declarative” โดยใช้ database trigger

Database trigger หมายถึง database procedure ที่ใช้ควบคุม integrity ซึ่งเก็บไว้และสามารถเรียกใช้ในการ insert, delete หรือ update operations

Database trigger นั้นถูกสร้างโดยผู้บริหารฐานข้อมูลและผู้ใช้ หากผู้ใช้เป็นผู้สร้างก็ควรอยู่ภายใต้ความควบคุมของผู้บริหารฐานข้อมูล

เหตุการณ์ที่ควรใช้ database trigger

- 1) เมื่อใช้ referential integrity โดยไม่ใช้ integrity constraint (database trigger ต่างจาก integrity constraint ตรงที่ database trigger จะใช้ integrity rules โดยไม่ตรวจสอบข้อมูลในตาราง ในขณะที่ integrity constraint จะมีการตรวจสอบข้อมูลในตารางก่อน ดังนั้นควรจะใช้ database trigger เมื่อไม่สามารถใช้ integrity constraint ได้)
- 2) ใช้ referential integrity เมื่อ child และ parent ตารางอยู่ที่ต่าง node กันในระบบ distributed database
- 3) ใช้ business rules ที่ยุ่งยากซับซ้อนและไม่สามารถใช้ integrity constraint ได้

3.3 TRANSACTION MANAGEMENT

3.3.1 Concurrency Control

ORACLE จัดการ concurrency control โดยใช้

1. Lock
2. Multi-version consistency model

1. Lock Mechanism

จะแบ่งออกเป็น 5 ประเภท คือ

- 1) Data Lock (DML lock)
- 2) Dictionary Lock (DDL lock)
- 3) Internal Lock and Latches
- 4) Distributed Lock
- 5) Parallel Cache Management Lock (PCM)

1) Data Lock (DML lock)

เพื่อป้องกันตารางซึ่งมั่นใจได้ว่าข้อมูลในตารางจะถูกต้องตรงกันแม้ว่าจะมีผู้ใช้หลายคนเข้าถึงตารางเดียวกัน แบ่งระดับของ data lock ได้ 2 level คือ

1.1) Row lock

1.2) Table lock

1.1) Row lock

เป็น data lock ที่จะทำการ lock เฉพาะ rows ที่ต้องการ โดยจะจัดเป็น “exclusive data lock” ก็ต่อเมื่อ lock เพื่อต้องการที่จะแก้ไขฐานข้อมูลโดยใช้ statement ต่อไปนี้ : INSERT, UPDATE, DELETE, SELECT พร้อมกับ FOR UPDATE CLAUSE row จะถูก lock เพื่อว่าผู้ใช้อื่นๆ จะได้ไม่สามารถมาแก้ไข rows ที่เข้าถึง อยู่และจะ hold lock จนกว่า transaction ที่ hold lock นั้น commit หรือ roll back

1.2) Table lock

Transaction ที่ต้องการแก้ไขตารางด้วย statement ต่อไปนี้ : INSERT, UPDATE, DELETE, SELECT พร้อมกับ FOR UPDATE CLAUSE และ LOCK TABLE table lock มีไว้เพื่อ

- รักษาระดับของการเข้าถึงตารางแทน transaction
- ป้องกันการขัดแย้งกับ DDL operations

Mode ของ table lock แบ่งออกเป็น 5 mode คือ

1.2.1) Row Share Table Lock (RS)

เมื่อ transaction hold table lock ในขณะที่มีการ lock rows ในตาราง และมีเจตนาที่จะแก้ไข โดยทำให้เกิดระดับของ concurrency มากที่สุดด้วย statement “LOCKTABLE table IN ROW SHARE MODE”

Operations ที่อนุญาตให้เกิด

- จะยอมให้แทรก, เพิ่ม, แก้ไข, ลบ หรือ lock rows ในตารางเดียวกัน
- จะยอมให้ transaction อื่นสามารถทำงานใน mode row share, row exclusive share, share, share row exclusive mode ได้ในตารางเดียวกัน

Operations ที่ห้ามไม่ให้เกิด

- Transaction อื่น ๆ จะอยู่ exclusive mode ไม่ได้

1.2.2) Row Exclusive Table Lock (RX)

เมื่อ transaction hold lock จะต้องเกิดการแก้ไข rows ในตารางแน่นอน ด้วย statement “LOCK TABLE table IN ROW EXCLUSIVE MODE”

Operations ที่อนุญาตให้เกิด

- จะยอมให้ แทรก, เพิ่ม, แก้ไข, ลบ หรือ lock rows ในตารางเดียวกัน
- จะยอมให้ transaction อื่น lock แบบ row exclusive และ row share mode ในตารางเดียวกับที่อีก transaction lock อยู่ได้

Operations ที่ห้ามไม่ให้เกิด

- Transaction อื่นไม่สามารถจะ lock แบบ share mode, share row exclusive, exclusive mode

1.2.3) Share Table Lock (S)

เมื่อ transaction hold lock จะยอมให้ transaction อื่น ๆ ทำได้เพียง query และ lock rows ที่ระบุเท่านั้น ซึ่งจะไม่ยอมให้ transaction อื่น ๆ มาแก้ไขด้วย statement “LOCK TABLE table IN SHARE MODE”

Operations ที่อนุญาตให้เกิด

- จะยอมให้ transaction อื่น ๆ lock แบบ share mode และ row share table mode ได้

Operation ที่ห้ามไม่ให้เกิด

- Transaction อื่นไม่สามารถจะ lock แบบ row exclusive, share row exclusive, exclusive mode

1.2.4) Share Row Exclusive Table Lock (SRX)

จะมีเพียง transaction เดียวเท่านั้น ที่จะ hold lock แบบ share row exclusive table lock ด้วย statement “LOCK TABLE table IN SHARE ROW EXCLUSIVE MODE”

Operations ที่อนุญาตให้เกิด

- จะยอมให้ transaction อื่น ทำ query และ lock rows ที่ระบุโดยไม่มีแก้ไข
- จะยอมให้ transaction อื่น lock แบบ row share mode เท่านั้น

Operations ที่ห้ามไม่ให้เกิด

- Transaction อื่นไม่สามารถจะ lock แบบ row exclusive, share, share row exclusive, exclusive mode

1.2.5) Exclusive Table Lock (X)

จะไม่มีการทำ transaction hold lock ไม่ว่าจะเป็น mode ใดด้วย statement “LOCK TABLE IN EXCLUSIVE MODE”

Operations ที่อนุญาตให้เกิด

- จะยอมให้ transaction อื่น ทำได้เพียง query ที่ไม่มีการ lock เท่านั้น

Operations ที่ห้ามไม่ให้เกิด

- Transaction อื่นไม่สามารถจะ lock แบบใดได้เลย

Data lock Conversion และ Escalation

ORACLE มีความสามารถ convert table lock ซึ่งมีความเข้มงวดน้อยกว่าไปเป็น lock แบบที่มีความเข้มงวดมากกว่าได้ เช่น ถ้ามี transaction ที่ hold แบบ row share table lock หากต้องแก้ไขตารางจริงๆ (อาจด้วย statement “SELECT.. FOR UPDATE”) ORACLE สามารถเปลี่ยนให้เป็น row exclusive table lock ได้เองโดยอัตโนมัติ

2) Dictionary Lock (DDL lock)

จะ protect ส่วนที่เป็นคำนิยามของ statement object ในขณะที่ object นั้นอยู่ภายใต้ DDL operations ใน transaction เช่น เมื่อผู้ใช้สร้าง procedure แล้วผู้ใช้ไม่จำเป็นต้องจัดการเกี่ยวกับการป้องกัน เพราะ ORACLE จะจัดการ lock ให้โดยจะ lock object ที่อ้างอิงทั้งหมดไม่ให้ถูกเปลี่ยนแปลงหรือ drop ทั้งจนกว่า procedure นั้นจะ complied ใหม่เสร็จสมบูรณ์ หากมีการ lock เกิดขึ้นไม่ว่าแบบใด

DDL lock แบ่งเป็น 3 กรณี คือ

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์โดยบริษัท ไม่นานาชาติ จำกัด ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.1) Exclusive DDL lock

สำหรับ resource ซึ่งต้องการป้องกันการแทรกแซงในระหว่าง DDL operation เช่น ในระหว่าง ALTER TABLE จะไม่ทำ DROP TABLE operation หาก object ที่ถูก hold แบบ exclusive DDL lock แล้วมี transaction อื่นจะขอ hold จะต้องรอจนกว่า exclusive DDL lock นั้นถูก released ก่อน

2.2) Share DDL lock

สำหรับ resource ซึ่งต้องการป้องกันการแทรกแซง ในระหว่าง DDL operation ซึ่งอาจทำให้เกิด conflict DDL operations เช่นมี operation CREATE PROCEDURE จะทำให้เกิด share DDL lock และ transaction อื่น ๆ จะสามารถ CREATE PROCEDURE ที่ reference table เดียวกันได้
หมายเหตุ : หากมีการ lock เกิดขึ้นไม่ว่าแบบใด transaction อื่นจะไม่สามารถ lock แบบ exclusive DDL lock ได้ ทำให้มั่นใจได้ว่า definition ของ object ที่ใช้ reference จะเป็น definition เดียวกัน ตลอดช่วงของ transaction นั้น

2.3) Breakable DDL lock

SQL statement (หรือ PL/SQL program unit) ที่ใช้ร่วมกันจะ hold parse lock เฉพาะ object ที่ถูกอ้างอิงโดย SQL statement ซึ่งเป็นผลทำให้จะไม่ มี DDL operations ซึ่งขัดแย้งกันเลย

3) Internal Lock and Latches

จะปกป้องในส่วนภายในของฐานข้อมูลรวมทั้งโครงสร้างของหน่วยความจำในลักษณะเป็นไฟล์

4) Distributed Lock

เพื่อให้มั่นใจว่าข้อมูลและ resource ที่กระจายอยู่ตาม server ต่าง ๆ ในระบบ ORACLE Parallel Server จะยังคงถูกต้องตรงกันอยู่

5) Parallel Cache Management Lock

เป็น distributed lock ซึ่งครอบคลุม data block (ตาราง และ index block) หนึ่ง block หรือมากกว่า ภายใน buffer ของ cache

2. Multi - Version Consistency Model

ORACLE จะสร้างชุดของข้อมูล ในขณะที่กำลังทำ queries (read) พร้อมกับมีการแก้ไข (write) ดังนั้นเมื่อเกิดการแก้ไข จะมีการเก็บค่าเดิมของข้อมูลไว้ในฐานข้อมูล และหาก transaction นั้น ยังไม่ commit และมี user query, ORACLE จะ read จาก version ที่จะทำให้ consistency ที่สุด แต่เมื่อ transaction นั้น ยัง commit แล้วค่าของข้อมูลจะถูกเปลี่ยนแปลงอย่างถาวร ซึ่งจะเป็น 2 level คือ

- 1) Statement-level read consistency : เพื่อรับรองว่า ข้อมูลหลังจากการทำ single query จะถูกต้องตรงกันก่อนทำ query ดังนั้น statement INSERT, DELETE, UPDATE จะต้องทำให้ข้อมูลก่อนและหลัง จากการ execute statement จะถูกต้องตรงกัน
- 2) Transaction-level read consistency : เพื่อ guarantee ว่าในสถานะ multiple-queries ใน transaction ใด transaction หนึ่ง นั้นการ queries ทั้งหมดจะมองเห็น ข้อมูลที่เป็นค่าตรงกันทุกการ queries ที่เวลาใดเวลาหนึ่ง

Deadlock

ORACLE จะจับว่ามี deadlock และเมื่อจับได้แล้วก็แก้ปัญหโดยอัตโนมัติ ด้วยการ roll back transaction ที่เป็นเจ้าของ statement ที่ทำให้เกิด deadlock ทันทีเพื่อ release lock ซึ่งขัดแย้งกันอยู่

ORACLE จะจับ deadlock โดยมีวิธีต่างกันแบ่งตามประเภทของ deadlock

- 1) ใช้วิธี "waits for graph" เพื่อจับ deadlock แบบ local deadlock
- 2) ใช้วิธี "time-out" เพื่อจับ deadlock แบบ global deadlock

6.3.2 Backup และ Recovery

วิธีการ backup และ recovery ใน ORACLE นั้นจะใช้ log-based recovery โดยเมื่อผู้ใช้ส่ง commit statement มาแล้ว commit record ก็จะถูกนำไปเก็บ redo log buffer ทันที แต่จะ 8 คอยการเขียนข้อมูลลงดิสค์ จนกว่าจะมีการเปลี่ยนแปลงใน data buffer

- Group commit ในเวลาที่มี activity มาก ๆ log writer (LGWR) อาจทำการเขียน redo log file โดยใช้ group commit ซึ่งเป็นการประหยัด disk I/O โดยรวมเอา redo entry จากหลาย ๆ ผู้ใช้ แล้วจึงเขียนลงดิสค์ทีเดียว
- Two-phase commit คือในระบบ multiuser ถ้า site หนึ่งทำการ commit หรือ rollback อีก site หนึ่งก็ต้อง commit หรือ rollback ตามเช่นกัน

Process ที่เกี่ยวข้องกับการสร้าง log file ที่สำคัญ ได้แก่

LGWR (Log Writer) : เป็นขบวนการที่ทำการเขียน redo log buffer ลง redo log file บนดิสก์ โดย LGWR process จะเขียนทุก ๆ redo ที่อยู่ในบัฟเฟอร์ตั้งแต่ครั้งสุดท้ายที่ได้ทำการเขียน โดยสิ่งที่ LGWR เขียนมีดังนี้

- Commit record เมื่อผู้ใช้ทำการ process commit transaction
- Redo buffer ทุก ๆ 3 วินาที
- Redo buffer เมื่อ redo log buffer เต็มไป 1 ใน 3
- Redo buffer เมื่อ DBWR ทำการ process write modified buffer ไปยังดิสก์

DBWR (Database Writer) : เป็นขบวนการที่ทำการเขียน buffer data ลงไปยัง data file ซึ่งจะถูกจัดการโดย buffer cache management เมื่อบัฟเฟอร์ใน buffer cache ถูกแก้ไขแล้ว จะทำการตั้งให้เป็น “dirty” และจะถูก “clean” ได้ก็ต่อเมื่อมีการทำ DBWR process เพื่อทำการเขียน dirty data ลงดิสก์ ในขณะที่บัฟเฟอร์ใน cache มีข้อมูลจากฐานข้อมูลมาใส่ และถูกทำให้ dirty โดย user process จะทำให้จำนวนของที่ว่างในบัฟเฟอร์ลดลง ซึ่งถ้าลดลงมากจน process ของผู้ใช้ไม่สามารถหาที่ว่างในบัฟเฟอร์ได้ DBWR จะจัดการ buffer cache ให้กับ process ของผู้ใช้ซึ่ง ORACLE จะจัดการโดยใช้ LRU Algorithm (least recently used) เพื่อที่จะให้ most recently used คือข้อมูลส่วนที่ถูกเรียนใช้บ่อยครั้งยังคงอยู่ในหน่วยความจำ โดย DBWR process จะถูกสั่งให้เขียน dirty buffer ลงดิสก์ ภายใต้งานต่อไปนี้

- เมื่อถึงเวลา checkpoint ซึ่ง LGWR จะทำการส่งสัญญาณให้ DBWR ทำการเขียน
- เมื่อเกิด time out (ทุก ๆ 3 วินาที) ซึ่ง DBWR จะทำการค้นหาที่ว่างในบัฟเฟอร์ ตาม LRU Algorithm ด้วยตัวเอง
- เมื่อ server process พบว่ามีเนื้อที่ว่างในบัฟเฟอร์ ไม่พอกับความต้องการ
- เมื่อ server process ทำการย้าย dirty list ไปยัง buffer และพบว่า dirty list นั้นยังไม่ถึงความยาวของ threshold ซึ่ง server process จะสั่งให้ DBWR เขียน

Redo log buffer เป็น circular buffer คือเมื่อ LGWR ทำการเขียน redo entry จาก redo log buffer ลงไปยัง redo log file แล้วนั้น server process ก็สามารถ copy entry ใหม่ลงไปทับ entry บน redo log buffer ที่ถูกได้ถูกเขียนลงดิสก์ไปแล้วได้

โดย ปกติ LGWR จะทำการเขียนอย่างรวดเร็วเพียงพอ จนสามารถแน่ใจได้ว่าเนื้อที่ว่างใน

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์การเขียนเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปเผยแพร่โดยไม่ได้รับอนุญาต

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บัฟเฟอร์จะมีเพียงพอสำหรับ entry ใหม่ ๆ ไม่ว่าจะมีการเข้าถึงมากเพียงใดก็ตามหรือถ้าเกิดกรณีที่ต้องการเนื้อที่ของบัฟเฟอร์มากกว่าที่มีอยู่ LGWR จะเขียน redo log entry ก่อนที่ transaction จะ commit ซึ่ง entry เหล่านี้จะ permanent ก็ต่อเมื่อ transaction นั้น commit ในเวลาต่อมา

Rollback segment จะถูกใช้สำหรับฟังก์ชันในการปฏิบัติงานของ ORACLE โดย rollback segment ของฐานข้อมูลจะประกอบด้วย rollback entry ซึ่งมีไว้เพื่อเก็บ oldvalue ของข้อมูล โดยข้อมูลที่กำลังถูกเปลี่ยนแปลงโดย transaction ที่กำลังจะทำงานต่อไป โดยเก็บเอาไว้ใน database space เนื่องจาก rollback entry จะทำการเปลี่ยนแปลง block ของข้อมูล ดังนั้นจะมีการเก็บการเปลี่ยนแปลงเอาไว้ใน redo log file ด้วยซึ่งในการเก็บครั้งที่ 2 นี้ จะมีความสำคัญมากสำหรับการทำ transaction ที่ยังไม่ได้ commit ในเวลาที่เกิด system crash Information ใน rollback segment นี้จะถูกใช้ระหว่างการทำ database recovery เพื่อทำ undo ในการเปลี่ยนแปลงที่ uncommitted ดังนั้นถ้าจำเป็นต้องทำ database recovery แล้วหลังจากที่ ใช้ rollback segment เพื่อทำการย้ายข้อมูล ที่ยังไม่ได้ commit ออกจาก data file แล้วฐานข้อมูลก็จะอยู่ในสภาวะที่ consistent

Redo log file จะบันทึกการเปลี่ยนแปลงทั้งหมด ที่ทำต่อฐานข้อมูล ซึ่งจะประกอบด้วย redo log file 2 ประเภท คือ

Online redo log file จะประกอบด้วย

- 1) Redo entry ซึ่งจะทำการบันทึกข้อมูลที่สามารถใช้ในการเปลี่ยนแปลงทั้งหมด ที่กระทำต่อฐานข้อมูลใหม่
- 2) Rollback segment โดย online redo log จะเป็นการป้องกันการ rollback ข้อมูลด้วย redo entry จะเป็น low-level representation ของการเปลี่ยนแปลงในฐานข้อมูล ซึ่งไม่สามารถเขียนหรือทำการเปลี่ยนแปลงใด ๆ ได้

Redo entry จะถูกทำกับ redo log buffer คือเมื่อ transaction commit แล้ว

LGWR จะ เขียน transaction redo entry จาก redo log buffer ไปยัง redo log file และจะมีการให้ค่า sequence change number (SCN) ไปด้วยเพื่อชี้ redo entry สำหรับแต่ละ committed transaction แต่อย่างไรก็ตาม redo entry สามารถที่จะถูกเขียนลง online redo log file ได้ก่อนที่ transaction จะ commit ในกรณีที่ redo log buffer เต็ม

โครงสร้างของ online redo log ของฐานข้อมูลจะประกอบด้วย online redo log file มากกว่าหนึ่งไฟล์ และมีการทำงานแบบหมุนเวียนคือเมื่อ online redo log file หนึ่งไฟล์ ปัจจุบันถูกใช้ในการใส่ข้อมูลแล้ว LGWR จะเริ่มทำการเขียน online redo log file ไฟล์ถัดไปและเมื่อเลื่อนไปถึง redo log file อันสุดท้าย LGWR ก็จะ return กลับมายัง redo log file อันแรก

Online redo log file อันที่สามารถจะใส่ข้อมูลได้ จะขึ้นอยู่กับว่าการตั้ง archiving ถูก enable หรือไม่

1) ถ้า archiving ถูก disable แล้ว online redo log file ที่จะถูกใส่ได้ต้องผ่านการ checkpoint เพื่อให้การเปลี่ยนแปลงที่อยู่ใน log file ครั้งก่อนถูกเขียนลงในไฟล์ข้อมูลก่อน แล้วจึงยอมให้มีการใช้ online redo log file ไฟล์นั้นใหม่ได้

2) ถ้า archiving ถูก enable แล้ว online redo log file ที่จะถูกใส่ได้ต้องผ่านการ checkpoint และ online redo log file จะต้องถูก archive ด้วย

Checkpoint เป็น event ที่ DBWR ทำการเขียน database buffer ที่ถูกเปลี่ยนแปลงทั้งหมดทั้ง committed และ uncommitted data ลงไปยังไฟล์ข้อมูล

เหตุผลที่ต้องมี checkpoint

1) Checkpoint จะทำให้แน่ใจว่า data segment block ในหน่วยความจำที่มีการเปลี่ยนแปลงบ่อย ๆ จะถูกเขียนลงในไฟล์ข้อมูลเป็นระยะ ๆ เนื่องจากการใช้ LRU algorithm ในการทำ DBWR จะทำให้ data segment block ที่มีการเปลี่ยนแปลงบ่อย ๆ จะไม่ถูกเขียนลงดิสก์เลย จึงทำให้จำเป็นต้องมี checkpoint

2) เนื่องจากการเปลี่ยนแปลงของฐานข้อมูล ทั้งหมดจนกระทั่งถึง checkpoint จะถูกบันทึกลงในไฟล์ข้อมูล ทำให้ redo log entry ก่อนหน้า checkpoint ไม่มีความจำเป็นต้องใช้อีก ในกรณีที่เกิดจะต้องทำ recovery ดังนั้น checkpoint จึงมีประโยชน์เพราะจะช่วย recovery เร็วขึ้น

Archived redo log file เป็น option ของ ORACLE ที่อนุญาตให้มีการรวม

กลุ่มของ online redo log file แล้วเก็บลงใน archived (offline) redo log โดยเป็นการเพิ่มประสิทธิภาพในการทำ database recovery เมื่อเกิดเหตุที่ได้อธิบายไว้สำหรับเก็บ

online redo log file เต็ม หรือต้องการที่จะนำเนื้อที่นั้นมาใช้เก็บการเปลี่ยนแปลงของฐานข้อมูลใหม่ ๆ ดังนั้นจึงต้องมีการถ่าย online redo log file เก่ามาเก็บไว้ใน archived redo log file

การทำ archive จะมีประโยชน์ต่อการ backup และ recovery ดังนี้

1) เป็นการทำให้ database backup คือทั้ง online และ archived redo log file จะสามารถรับรองได้ว่า transaction ที่ commit ไปแล้วทั้งหมดจะสามารถ recover

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับใช้ภายในเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2) เป็นการทำการ online backup คือเก็บรายละเอียดการเปิดฐานข้อมูลต่าง ๆ และรายละเอียดในการใช้งานระบบได้อย่างถาวรถ้าไม่มีการลบ archived redo log file นี้ทิ้ง

ขั้นตอนในการทำการ recovery

ORACLE จะมีขั้นตอนการในการ recovery 2 ขั้นตอนคือ

Redo Log and Rolling Forward

การ roll forward เป็นการย้อนกลับการเปลี่ยนแปลงทั้งหมดที่ได้เก็บไว้ใน redo log ไปยังไฟล์ข้อมูล และเนื่องจากข้อมูลในการ rollback จะถูกเก็บเอาไว้ใน redo log ด้วย ดังนั้น roll forward ต้องไปเรียกใช้ rollback segment ด้วย ซึ่งเป็นขั้นตอนของการ rollback

Rolling forward จะทำการจัดการกับ redo log file ที่จำเป็นสำหรับการนำฐานข้อมูลกลับคืนมา โดยจะใช้ทั้ง online redo log file และ archive redo log file หลังจาก roll forward แล้วภายในฐานข้อมูลจะมีทั้งการเปลี่ยนแปลงที่ commit แล้วทั้งหมดและการเปลี่ยนแปลงที่ยังไม่ commit ที่ถูกเก็บอยู่ใน redo log file

Rollback Segment and Rolling Back

Rollback segment จะถูกใช้เพื่อที่จะยกเลิก (undo) uncommitted transaction ที่ถูกทำการเปลี่ยนแปลงมาก่อนหน้านี้ โดย rolling forward phase

การทำ roll forward นั้นการเปลี่ยนแปลงใด ๆ ที่ยังไม่ commit จะต้องถูก undo หลังจากที่ได้ redo log file ได้ทำการย้อนกลับการเปลี่ยนแปลงที่ทำต่อฐานข้อมูลทั้งหมดแล้ว จะต้องมีการใช้ rollback segment ด้วยเพื่อที่จะ undo transaction ที่ยังไม่ได้ commit เนื่องจาก transaction ที่ยังไม่ commit นั้นการเปลี่ยนแปลงทั้งหมดยังอยู่ใน redo log buffer โดยยังไม่ได้เขียนลงใน redo log file โดย process นี้เรียกว่า Rolling back ซึ่งสามารถใช้ integrity constraint นี้กับโปรแกรมประยุกต์ที่เกี่ยวข้องกับฐานข้อมูล โดยไม่ต้องเขียนส่วนนี้ขึ้นเอง

Immediate user feedback : ก่อนที่จะเรียกใช้คำสั่ง SQL จะมีการตรวจสอบความถูกต้องของข้อมูลก่อนโดยใช้ integrity constraint ที่เก็บไว้ใน พจนานุกรมข้อมูล

3.4 QUERY PROCESSING

3.4.1 Storage Structure

ORACLE มีวิธีการจัดการการเก็บข้อมูลในตารางหลายวิธีดังนี้

Cluster

Cluster หมายถึงกลุ่มของตารางซึ่งใช้ data block ร่วมกัน ในลักษณะใช้คอลัมน์ร่วมกันโดยตาราง 2 ตารางซึ่งเป็นที่ใช้คอลัมน์ร่วมกันจะถูกเก็บไว้ใน data block เดียวกัน

Cluster key คือคอลัมน์หรือชุดของคอลัมน์ซึ่งใช้ตารางแบบ cluster ร่วมกันโดยคอลัมน์ดังกล่าวนี้จะต้องระบุตอนสร้างและคอลัมน์เดียวกันนี้จะอยู่ในตารางซึ่งจะเพิ่มเข้า cluster แบบอัตโนมัติ โดยที่

- cluster key มีได้ไม่เกิน 16 คอลัมน์ และค่าของ cluster key จะไม่เกิน 1/3 ของเนื้อที่ของข้อมูลใน block ของข้อมูล
- หากค่าของข้อมูลใน cluster คอลัมน์ มีการแก้ไข จะทำให้ต้องมีการเปลี่ยนแปลงตามตำแหน่งที่เก็บซึ่งอยู่ในระดับกายภาพ

ประโยชน์ของ cluster

- 1) ลด disk I/O และลดเวลาที่ใช้ในการเข้าถึง
- 2) เก็บตารางที่มีความสัมพันธ์กันและ index data ใน cluster ได้ลดลง

ข้อควรพิจารณา

- cluster จะลดความสามารถของ INSERT statement เพื่อเปรียบเทียบกับ การเก็บตารางโดยมีการแยก index
- ตารางซึ่งมี block เป็นจำนวนมากควรเก็บไว้ใน cluster
- ตารางซึ่งเกี่ยวข้องกับ referential integrity และตาราง ที่มีการเรียกใช้ด้วย SELECT-statement และมีการ join ตารางบ่อย ๆ จะเหมาะสมกับวิธี cluster เพราะทำให้ลดจำนวน data block ที่จะเข้าใช้ในการ queries
- ไม่เหมาะกับกรณีที่คอลัมน์ที่เป็น cluster key ที่มีการแก้ไขบ่อย ๆ

Hash

การใช้ hash จะต้องสร้าง hash cluster และเรียกตารางเข้าไปใน cluster โดย row ของตารางใน hash cluster จะเก็บและดึงข้อมูลตามผลที่ได้จาก hash function

- “hash function” ใช้ในการสร้างค่าตัวเลข ที่มีความกระจายซึ่งค่านี้จะเรียกว่า “hash value”
- คีย์ของ hash cluster อาจเป็นคอลัมน์ หรือชุดของคอลัมน์

- ORACLE จะจัดการเก็บ rows ใน hash cluster โดยใช้ cluster key value ที่ได้จาก hash function
- ในการหาหรือจัดเก็บ row จะต้องใช้อย่างน้อย I/O 2 ครั้ง (มักจะมากกว่านี้) โดยใช้ I/O 1 ครั้งหรือมากกว่าในการหาและเก็บคีย์ value ใน index และใช้ I/O มากกว่า 1 ครั้ง สำหรับอ่าน หรือเขียน rows ในตารางหรือ cluster

B+-tree

ทำหน้าที่รักษาสมดุลย์เพื่อให้ใช้เวลาในการเข้าถึง ที่ใกล้เคียงกับ branchblock (block ที่อยู่เหนือกว่า) โดยมี index ซึ่งชี้ไปยัง index block ที่อยู่ระดับต่ำกว่าและ index block ที่อยู่ในระดับต่ำสุด เรียกว่า “leaf block” ซึ่งจะเก็บ index ของค่าของข้อมูลซึ่งอยู่ในรูป ROWID (ใช้บอกตำแหน่งจริงของ row)

- สำหรับ unique index จะใช้ ROWID 1 ค่าต่อค่าของข้อมูล
- สำหรับ non-unique index ROWID จะรวมอยู่ในคีย์ที่ใช้เรียงลำดับ ดังนั้นเราจึงใช้ index key และ ROWID ในการเรียงลำดับ non-unique indexes

ประโยชน์ของการใช้ storage structure แบบ B+-tree

- 1) leaf block ของ tree ทุก ๆ block จะอยู่ในระดับความลึกเดียวกันหมด ดังนั้นการดึง ข้อมูลไม่ว่าอยู่ในตำแหน่งจึงใช้เวลาพอ ๆ กัน
- 2) B+-tree index จะอยู่ในสภาพสมดุลเสมอ
- 3) B+-tree จะทำให้สามารถ queries ได้ในช่วงกว้างรวมทั้งสามารถค้นหาได้อย่างมีประสิทธิภาพ
- 4) สามารถเพิ่มเติม, เปลี่ยนแปลง, ลบได้อย่างรวดเร็ว
- 5) เหมาะกับตารางทั้งขนาดเล็กและใหญ่โดยไม่คำนึงถึงขนาดของตารางที่โตขึ้น

3.3.2 Data Independence ของ ORACLE

Physical Data Independence

ORACLE มีคุณสมบัตินี้คือไม่ว่าจะเปลี่ยนลักษณะการเข้าถึงไฟล์ในระดับล่างเป็นแบบใด ก็จะไม่กระทบในระดับสูงเท่าไหร่นัก โปรแกรมประยุกต์ที่เหนือกว่าไม่ต้องไปทำการเปลี่ยนแปลง

Logical Data Independence

ORACLE มีคุณสมบัติข้อนี้คือ ผู้บริหารฐานข้อมูล สามารถที่จะเปลี่ยนแปลงตารางเพิ่ม คอลัมน์, ตัดคอลัมน์, เพิ่มตาราง สิ่งเหล่านี้จะไม่มีผลต่อการเข้าถึงตารางของผู้ใช้ โดยไม่ต้องการแก้ไขโปรแกรมใหม่

3.3.3 Query Optimization

Optimization ของ ORACLE มีจุดมุ่งหมายเพื่อหาเส้นทางที่มีประสิทธิภาพที่สุด ในการประมวลผล SQL statement โดยวิธีในการทำ optimization สำหรับ SQL statement นั้นมีด้วยกัน 2 วิธี คือ

1) Rule-based approach

2) Cost-Based approach

Rule-based approach : optimizer จะเลือกเส้นทางที่ใช้ในการประมวลผลโดยดูจาก access path และพิจารณา heuristically ranked operation ของ rank ซึ่ง rank ที่มีค่าต่ำ จะแสดงถึงการเข้าถึงข้อมูลที่เร็วกว่า โดย ORACLE ได้กำหนด rank ไว้ดังนี้

Rank	Access Path
- 1 -	Single row by ROWID
- 2 -	Single row by cluster join
- 3 -	Single row by hash cluster key with unique or primary key
- 4 -	Single row by unique or primary key
- 5 -	Cluster join
- 6 -	Hash cluster key
- 7 -	Indexed cluster key
- 8 -	Composite index
- 9 -	Single-column index
- 10 -	Bounded rang search on indexed columns
- 11 -	Unbounded range search on indexed columns
- 12 -	Sort-merge join
- 13 -	MAX of MIN of indexed column
- 14 -	ORDER BY on index column
- 15 -	Full scan table

Cost-based approach : ใช้วิธีการเก็บสถิติไว้เป็นจำนวนมาก ซึ่งจะรวมทั้งรายละเอียดเกี่ยวกับตัวข้อมูล, index, จำนวนแถว, จำนวนหลัก และสถิติที่เกี่ยวกับตาราง โดย optimizer จะเก็บค่าที่เป็นไปได้ของข้อมูล รวมทั้งค่าสูงสุดและค่าต่ำสุดของข้อมูลที่มีอยู่ ซึ่งไม่ได้เก็บข้อมูลในรูป histogram

Stored Procedure and Function

เป็น object ที่เป็นการรวมเอากลุ่มของ SQL และ PL/SQL programming language statement เข้าด้วยกันในทาง logical เพื่อแสดงการทำงานเฉพาะขึ้นมา และเก็บ code ที่ compile แล้วนี้ไว้ใน shar pool ของ SGA และ compiled version นี้จะอยู่ใน shared pool ตาม LRU algorithme

Procedure และ function นี้จะถูกสร้างขึ้นในระดับของผู้ใช้และเก็บไว้ในฐานข้อมูลสำหรับใช้งานต่อไป โดยจะถูกประมวลผลอย่าง interactive โดยใช้ ORACLE tool เช่น SQL*DBA หรืออาจถูกเรียกโดยตรงจากโปรแกรมประยุกต์ของฐานข้อมูล เช่น

SQL forms, procedure อื่น ๆ หรือ trigger

นอกจากนี้ procedures, functions, cursors, variables ที่สัมพันธ์กันยังสามารถนำมารวมเข้าด้วยกันได้รวมเป็นหนึ่งหน่วยเรียกว่า package และถูกนำไปเก็บไว้ในฐานข้อมูลเพื่อการใช้งานต่อไป

ประโยชน์ของ store procedure

- 1) ทางด้าน security เนื่องจาก store procedure สามารถช่วยควบคุมความปลอดภัยของข้อมูล ดังนั้นจึงสามารถจำกัดการทำงานระหว่างผู้ใช้และฐานข้อมูลแต่ละผู้ใช้ได้ เพราะในการเรียกใช้ procedure ORACLE จะมีการตรวจสอบสิทธิในการให้ข้อมูลของผู้ใช้
- 2) Stored procedure สามารถเพิ่มประสิทธิภาพของฐานข้อมูลเนื่องจากสามารถลดข้อมูลที่ต้องการส่งข้ามเครือข่ายเมื่อเทียบกับการส่ง SQL statement เข้าไปและยังไม่ต้องมีการ compile ซ้ำอีกในการ execute code นอกจากนี้ ถ้าเกิดมี procedure อยู่แล้วใน shared pool ทำให้ไม่จำเป็นต้องเรียกขึ้นมาจากดิสก์ สามารถประมวลผลได้ทันที
- 3) ประหยัดหน่วยความจำเนื่องจาก stored procedure ใช้ประโยชน์ของ shared memory ดังนั้นจึงไม่เกิดความซ้ำซ้อนของคำสั่งในหน่วยความจำ เมื่อมีการเรียกใช้โดย multiple users
- 4) Stored procedure จะช่วยเพิ่มประสิทธิภาพ เนื่องจากในการออกแบบโปรแกรมประยุกต์โดยใช้ชุดของ procedure ทำให้สามารถที่จะเลี่ยงความซ้ำซ้อน เช่น procedure จะถูกเขียนขึ้นมาเพื่อแก้ไข, แทรก, ลบ row ออกจากตาราง ซึ่งโปรแกรมประยุกต์จะสามารถเรียกใช้ procedure เหล่านี้โดยไม่ต้องเขียน statment ใหม่ และเมื่อเกิดวิธีของการจัดการข้อมูลเปลี่ยนไป เพียงทำการแก้ไข procedure ใหม่เท่านั้นไม่ต้องทำการแก้ไขโปรแกรมประยุกต์ที่ใช้ procedure นั้น

5) Stored procedure จะช่วยเพิ่ม integrity และ consistency ของโปรแกรมประยุกต์ให้ถูกต้องแน่นอนขึ้นคือแต่ละ procedure หรือ function ต้องการทดสอบเพียงครั้งเดียวเพื่อที่จะรับประกันว่าได้ผลลัพธ์ถูกต้อง หลังจากนั้นโปรแกรมประยุกต์ใด ๆ จะสามารถเรียกใช้ procedure นี้ได้ ถ้าเกิดการเปลี่ยนแปลงของโครงสร้างข้อมูลที procedure นั้นอ้างอิง จะมีเพียง procedure นั้นที่จะต้อง compile ใหม่ โดยไม่ยุ่งเกี่ยวกับโปรแกรมประยุกต์ที่ใช้

3.5 ADDITIONAL TOOLS

Cooperative Development Environment (CDE) เป็นชุดเครื่องมือที่ใช้สำหรับพัฒนาระบบงานคอมพิวเตอร์ ซึ่งจะครอบคลุมทุกขั้นตอนในการพัฒนาโปรแกรมประยุกต์ โดยเริ่มตั้งแต่การ reengineer กระบวนการทำงาน (business process) และการให้คำจำกัดความตามข้อกำหนดของโปรแกรมที่จะใช้งาน (application requirement definition) รวมถึงขั้นตอนการวิเคราะห์ออกแบบพัฒนางานและการแก้ไขเปลี่ยนแปลง โดยใช้ได้ตั้งแต่ single-user จนกระทั่งถึง enterprise-wide system

CDE ประกอบด้วย เครื่องมือ 3 ประเภทได้แก่

- 1) CASE tools
- 2) Application development
- 3) Office automation and end user

3.5.1 CASE Tools

ORACLE มี CASE family ซึ่งจะรวมเอาเครื่องมือที่ใช้สำหรับการพัฒนา คือ ตั้งแต่การกำหนดวิธีไปจนถึงการสร้างออกมาเป็นโปรแกรม โดย ORACLE/CASE ประกอบด้วย

CASE method

เป็นเครื่องมือที่ช่วยในด้านวิธีการที่จะใช้ในการพัฒนาระบบ โดยใช้วิธีของ top-down design ซึ่งใช้การระบุขั้นตอนสำคัญ ๆ เพื่อเป็นช่วยในการออกแบบ

CASE Dictionary (Shared Portable Repository)

เป็นที่เก็บข้อมูลสำหรับทุก ๆ ขั้นตอนและข้อมูลเหล่านั้นจะสามารถถูกควบคุมได้โดยง่ายผ่านทาง forms interface หรือผ่านเครื่องมือทางกราฟฟิค CASE Dictionary มีความสามารถในการสร้างเอกสารประกอบอัตโนมัติ โดยสามารถเลือกได้ถึง 3 ชนิด คือ predefined reports, impact analysis reports และ custom report โดยใช้ SQL reportwriter สำหรับการสร้างเอกสารของระบบทำให้ผู้พัฒนาสามารถสร้างรายงานเองได้อย่างรวดเร็ว และ CASE Dictionary นี้เอกสารนี้เป็นเอกสารที่จะรันได้บนหลาย platform ด้วย

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ตัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

CASE Designer

เป็นเครื่องมือที่ใช้เพื่อกำหนดความต้องการของผู้ใช้และระบบ โดยใช้ graphic model ในขณะที่ผู้ใช้งานตรูปลงใน diagram ตัวใดตัวหนึ่ง CASE designer จะทำการแก้ไข CASE dictionary ในทันทีโดยไม่มีการสูญเสีย concurrency CASE designer จะสนับสนุนระบบ multiuser คือผู้ใช้หลาย ๆ คนสามารถที่จะเข้าถึง CASE designer ได้ในเวลาเดียวกันแม้จะเป็นการทำงานบน models ที่เหลื่อมซ้อนกันก็ตามและผู้ใช้แต่ละคนสามารถที่จะเปิด windows ได้โดยไม่จำกัด เพื่อดูหลาย ๆ พื้นที่ของ model ที่ทำอยู่โดยแต่ละ window จะมีการประมวลผลที่แยกออกจากกัน นอกจากนี้ CASE Designer ยังจะสามารถเคลื่อนย้ายไปรันบนสภาพแวดล้อมทางกราฟฟิคต่าง ๆ ได้เช่น X-window, DEC window และ presentation manager เป็นต้น

CASE Generator Series

เป็นเครื่องมือที่ใช้เพื่อการสร้าง source code โดยใช้ข้อมูลที่เก็บอยู่ใน CASE dictionary มีทั้งสิ้น 4 ตัว

1) CASE Generator for SQL Forms

ใช้ในการสร้างโปรแกรมประยุกต์จาก diagram สำหรับ SQL forms และโปรแกรมประยุกต์ที่ถูกสร้างขึ้นนี้จะสามารถรันได้บนทุก ๆ แพลตฟอร์ม ที่ ORACLE สนับสนุน output ของ generator ตัวนี้เป็น functional interactive application ที่จะรวมเอา SQL statement ต่าง ๆ เช่น enter, edit, retrieve ข้อมูลจากฐานข้อมูลไว้ด้วยและยังรวม logic ต่าง ๆ ที่ใช้ในการบังคับกฎทางธุรกิจที่ถูกกำหนดไว้ใน CASE dictionary ทำให้เราไม่ต้องเขียน code เข้าในส่วนของการควบคุมหน้าจอ, การตรวจสอบข้อมูลและ integrity constraint ซึ่งจะถูกบังคับใช้โดยอัตโนมัติและโดยการใช้ SQL form designer ประกอบ ทำให้เราสามารถที่จะ override default เช่น เปลี่ยนรูปหน้าจอ เพิ่มความสามารถอื่น ๆ เข้าไปในโปรแกรมประยุกต์โดยเขียน routines เพิ่มเข้าไป โปรแกรมประยุกต์ที่ถูกสร้างขึ้นจะ compatible กับ block-mode, character-mode และ bitmap devices

2) CASE Generator for SQL Forms/SQL Menu

เป็น CASE generator ที่สามารถจะสร้างทั้ง form layout และ menu layout และการเข้าถึง ฐานข้อมูลได้โดยอัตโนมัติ และโปรแกรมประยุกต์ที่สร้างสามารถจะสร้างขึ้นใหม่ได้ทุก ๆ เวลาที่ความต้องการของผู้ใช้เปลี่ยนไป ในการสร้างโปรแกรมประยุกต์ CASE generator จะมีการสร้างระบบเมนูเต็มรูปแบบไปด้วย คือมีทั้ง menu item help, menu navigation, menu acces to form, other menus และมีหลายลักษณะของเมนูให้ให้เลือก เช่น full screen, pull down เป็นต้น

3) CASE Generator Database Tables

เป็น CASE generator ที่สามารถจะทำการออกแบบรูปแบบของตารางให้กับผู้ใช้ โดยจะ map มาจาก diagram ที่ผู้ใช้สร้างขึ้นมา

4) CASE Generator for SQL ReportWriter/SQL Plus

เป็น reporting tool ที่ใช้ร่วมกับ CASE dictionary เพื่อสร้าง layout และ report ตามที่ผู้ใช้ต้องการโดย CASE generator จะสร้างโปรแกรมประยุกต์สำหรับออกรายงานจากรายละเอียดต่าง ๆ ที่ถูกเก็บใน CASE dictionary

- CASE exchange เป็น facility ที่ทำให้ข้อมูลสามารถที่จะถูกส่งผ่านระหว่าง non-ORACLE CASE tools และ CASE dictionary ได้

3.5.2 Application Development Tools

PL/SQL

เป็น procedural language ที่เพิ่มเติมจาก SQL มาตรฐาน ทำให้เราสามารถที่จะประมวลผล multistatement หรือ multitransaction PL/SQL procedure โดยใช้คำสั่งเดียว ไปยัง ORACLE จึงทำให้ลดการติดต่อสื่อสารบนเครือข่ายแบบ client/server และ PL/SQL สามารถที่จะประมวลผลการคำนวณของข้อมูล และ form control logic ได้เองโดยไม่ต้องส่งคำสั่งไปที่ฐานข้อมูลซึ่งเป็นการเพิ่มประสิทธิภาพของโปรแกรมประยุกต์เราสามารถใช่ PL/SQL เขียนโปรแกรมประยุกต์ร่วมกับ ORACLE tools อื่น ๆ ได้

ใน PL/SQL procedure สามารถที่จะมี SQL statement มาตรฐานและ conditional control statement ได้ โดยที่ PL/SQL procedure ที่ถูกกำหนดขึ้นมา จะสามารถถูกเรียกใช้ได้จาก form อื่น ๆ ด้วย PL/SQL จะสามารถรันได้บนทุก ๆ แพลตฟอร์ม ที่รัน ORACLE ด้วย

Precompilers

จะอนุญาตให้ฝัง SQL ลงในโปรแกรมหลักที่เขียนด้วย Ada, C, Cobol, Fortran, Pascal PL/1 ได้และสามารถจะควบคุมการติดต่อกับระบบจัดการฐานข้อมูลได้ด้วย

ORACLE precompilers เป็น full ANSI-compatible ซึ่งโปรแกรมประยุกต์ที่พัฒนาโดยใช้ precompilers สามารถที่จะรันได้ทั้งบน ORACLE ANSI - compatible SQL based DBMS และ DB2 จึงสามารถที่จะสร้าง customized application เช่นเราสามารถจะออกแบบโปรแกรมประยุกต์ให้ดึงข้อมูลจาก ORACLE บนเมนเฟรมแล้วให้ประมวลผลข้อมูลบนไมโครคอมพิวเตอร์ และให้แสดงข้อมูลบนเวิร์คสเตชันได้

SQL Plus

ใช้ในการทำ query, define และ control data ที่เก็บใน ORACLE DBMS มีความสามารถเช่นเดียวกับ ORACLE/SQL และ PL/SQL

ด้วย syntax ง่าย ๆ ของ SQL plus ผู้ใช้สามารถที่จะเข้าถึงข้อมูลได้อย่างรวดเร็วและยังสามารถที่จะดึงข้อมูล, ทำการคำนวณและสร้าง adhoc report ได้

SQL plus เป็น dynamic SQL-based queries คือผู้ใช้จะสามารถเปลี่ยน section ของ query ได้ขณะรันเราสามารถจะเพิ่ม PL/SQL block ลงใน SQL statement ได้ (เป็น procedure logic) โดยการใช้ SQL PLUS

3.5.3 Office Automation and End-user Tools

ORACLE Data Query

ถูกออกแบบมาเพื่อให้ผู้ใช้ทำการระบุและรัน adhoc queries ได้โดยไม่จำเป็นต้องรู้ SQL ในการสร้าง queries ผู้ใช้สามารถเลือกตาราง และคอลัมน์จาก lists การใช้งานเป็นแบบ step by step โดยใช้วิธี fill in the blank input ของผู้ใช้จะถูกตรวจสอบทุกขั้นตอนถ้าพบข้อผิดพลาดจะแก้ไขในทันที

ORACLE data query จะอนุญาตให้มีการสร้าง report หลาย ๆ แบบได้จาก 1 query ซึ่งสามารถคำนวณทางสถิติและข้อมูลสามารถจะถูกเรียงลำดับได้หลายวิธี

ขณะที่ query ถูก execute อยู่ผลสามารถจะถูกแสดงขึ้นจอ, เขียนลงไฟล์, หรือส่งตรงไปยัง printer ได้โดยตรง

ORACLE data query สามารถจะรันได้บนหลาย ๆ แพลตฟอร์ม (รวมทั้ง client/server configuration) และสนับสนุนทั้ง character-mode และ block-mode terminal

ORACLE data query เป็นเครื่องมือแบบ read only ผู้ใช้จะไม่สามารถแทรก, แก้ไข, หรือลบข้อมูลได้และ ผู้บริหารฐานข้อมูลสามารถจำกัดการทำ query

SQL QMX

เป็น query manager และ report writer ซึ่งจะรวมเอา browser, query facilities และ report writer ไว้ใน package

ในการใช้ SQL QMX ผู้ใช้สามารถที่จะใช้ function key-driven menus หรือจะใส่คำสั่งที่ยังไม่สมบูรณ์ได้ โดย SQL QMX จะเติมส่วนที่เหลือให้โดยผู้ใช้ไม่จำเป็นต้องรู้ command syntax สำหรับ report writer จะเป็นแบบ fill-in-the-blank formatting panel SQL QMX browser จะช่วยให้ผู้ใช้สามารถหาตารางหรือ view ได้โดยไม่ต้องสร้าง query และเมื่อผู้ใช้สร้าง report browser ก็จะทำให้ผู้ใช้สามารถที่จะ preview report บนหน้าจอก่อนจะ print ได้

ผู้ใช้สามารถที่จะเลือกเขียน SQL statement แทนการใช้ prompt panel ได้ และ SQL QMX ยังสนับสนุน macro procedure และ substitution variables ด้วย

เอกสารนี้เป็นเอกสารสงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

SQL Form และ SQL Menu

โปรแกรมประยุกต์ที่เขียนด้วย SQL form และ SQL menu จะสามารถรันได้บนทุก ๆ แพลตฟอร์ม ที่ ORACLE สนับสนุน โดยสามารถสร้างโปรแกรมประยุกต์โดยไม่ต้องเขียน code แต่ใช้วิธีระบุคุณสมบัติของโปรแกรมประยุกต์แทน โดยใช้ menu-driven user interface specification ดังกล่าวร่วมกับข้อมูลที่เก็บใน ORACLE data dictionary ซึ่งจะถูกใช้ในการสร้างโปรแกรมประยุกต์และสามารถใช้ร่วมกับ PL/SQL routine รวมทั้งสามารถเพิ่ม 3GL routine ลงในโปรแกรมประยุกต์ได้ด้วย

SQL TextRetrieval

เป็นความสามารถที่สามารถที่จะเพิ่มเข้าไปในโปรแกรมประยุกต์ของ ORACLE โดยเป็นการขยายส่วนของ SQL ซึ่งผลที่จะได้จะเป็น common language สำหรับที่เก็บของข้อมูลและตัวอักษร SQL textretrieval จะเก็บข้อมูลทั้งหมดรวมทั้งตัวอักษรต้นแบบและ indexes โดยผู้ใช้จะเข้าถึงข้อมูลทุกตัวได้ภายใต้การควบคุมของ ORACLE ผู้ใช้จะสามารถเข้าถึง SQL textretrieval software tools ได้โดยผ่าน SQL form หรือ 3GL programs

SQL textretrieval ทำให้การค้นหาข้อมูลมีประสิทธิภาพขึ้นโดย query สามารถที่จะรวมเอา words , concepts และโครงสร้างของฟิลด์ข้อมูลไว้ได้ และสามารถที่จะใช้ตัวอักษรจากที่ต่าง ๆ เช่น external file, screen input, export ตัวอักษรระหว่างโปรแกรมประยุกต์ SQL textretrieval สะดวกในการเคลื่อนย้ายบนหลาย ๆ แพลตฟอร์ม

SQL ReprotWriter

เป็นเครื่องมือที่ทำให้ผู้ใช้สามารถที่จะสร้างรายงานได้หลายรูปแบบ โดยใช้ข้อมูลจากหลายฐานข้อมูลทั้ง ORACLE และ non-ORACLE เช่น DB2 โดยไม่ต้องเขียนโปรแกรมเพียงแต่ระบุ report specification ลง forms (ซึ่งเป็นรูปแบบ spread sheet) SQL reprotwriter จะมี default ให้ทั้งตำแหน่งคอลัมน์, column heading และ display format เราสามารถที่จะ override default ได้ถ้าต้องการ SQL reprotwriter สามารถใช้เชื่อมกับ ORACLE tools อื่น ๆ ได้

ORACLE CARD

เป็นเครื่องมือที่ใช้ ในการพัฒนาโปรแกรมประยุกต์ทางกราฟิกโดยวิธีที่ง่าย และโปรแกรมประยุกต์ ยังสามารถรันได้หลายแพลตฟอร์มทั้ง MS-Windows และ Apple Macintosh system card มีความสามารถในการใช้ GUI ร่วมกับฐานข้อมูลแบบรีเลย์เนตได้ ในการสร้างและควบคุมฐานข้อมูล table, develop application และรวมทั้งการสร้าง report นั้นผู้ใช้เพียงแต่ point และ click mouse เชื่อม screen object (fields) เข้ากับ database object (column) แล้ว ORACLE จะจัดการส่วนที่เหลือเองโดยผู้ใช้ไม่ต้องเขียนโปรแกรมหรือ SQL ใดๆเลย และผู้ใช้ยังสามารถที่จะเก็บและดึงภาพในฐานข้อมูล ได้โดยการ click mouse ด้วย

ORACLE card ประกอบด้วยเครื่องมือ เช่น

- Table builder ใช้ในการสร้างและแก้ไขตารางในฐานข้อมูลโดยไม่ต้องเขียน SQL เลย
- Stack builder ใช้พัฒนาโปรแกรมประยุกต์ โดยการ point และ click mouse โดยโปรแกรมประยุกต์สามารถที่จะตรวจสอบ แทรก และแก้ไขทั้งตัวอักษรและภาพข้อมูลได้
- Query builder ใช้ทำ adhoc query ทางกราฟฟิค โดยจะเก็บข้อมูลลงใน clipboard หรือ implied report ก็ได้
- Programming environment ใช้ในการสร้างโปรแกรมประยุกต์ที่ซับซ้อนมากยิ่งขึ้น ในการสร้างโปรแกรมประยุกต์โดยใช้ ORACLE card โดยผู้ใช้จะเลือกที่จะเขียนโปรแกรมหรือไม่ก็ได้ โดยในส่วนของโปรแกรมจะมี ORACLE card 's object oriented programming logic และ 3GL function

3.6 DISTRIBUTED DATABASE MANAGEMENT

การเชื่อมต่อระหว่างโปรแกรมประยุกต์กับฐานข้อมูลยังคงถูกจัดอยู่ภายในข้อจำกัดของการเชื่อมต่อของคอมพิวเตอร์ 2 เครื่อง SQL*Net ของ ORACLE ได้ทำลายเส้นกันของโปรโตคอลสำหรับโปรแกรมประยุกต์ client/server Client สามารถทำงานได้โดย server ไม่ต้องพึ่งพาโปรโตคอลของระบบเครือข่าย SQL*Net เป็นระบบเครือข่ายที่เสมือน ORACLE โดยอนุญาตให้มีการเชื่อมต่อระหว่างโปรแกรมประยุกต์ client และ server ของฐานข้อมูลซึ่งแยกออกจากการเชื่อมต่อระหว่างเครื่องคอมพิวเตอร์ที่กำลังทำงานอยู่

Client/Server Computing

SQL*Net เป็นโปรแกรมควบคุมระยะไกลของ ที่ใช้ระหว่างโปรแกรมประยุกต์ของผู้ใช้และฐานข้อมูล หรือระหว่างฐานข้อมูลหลาย ๆ ตัว และด้วยโปรแกรมประยุกต์ของ SQL*Net ฐานข้อมูลสามารถที่จะกระจายทางด้านกายภาพออกไป โดยแยกเครื่องที่แตกต่างกันและเริ่มติดต่อกันถ้าเป็นแบบ local

Network Transparency

SQL*Net โปรแกรมประยุกต์ client ทั้งหมดจะทำหน้าที่เหมือนกับฐานข้อมูลระยะไกลเป็นของตัวเอง ซึ่งทำให้ระบบเครือข่ายสามารถรวม และดูการเชื่อมต่อกันผ่านทางโปรแกรมประยุกต์ของ client ได้

Multinetwork Data Access

Client และ server สามารถกระจายออกได้ไม่เพียงแต่บนเครื่องที่แตกต่างกันแต่รวมถึงเครือข่ายที่แตกต่างกัน เครื่อง client/server ต้องการใช้เพียงโปรโตคอลพื้นฐานที่มีมาให้นั่นเท่านั้น และยังสามารถติดต่อกับโปรแกรมประยุกต์ด้วยกัน Client และ server ทั้งหมดในระบบเครือข่าย ซึ่งสามารถแบ่งข้อมูลผ่านทางความสัมพันธ์แบบ client/server หรือ server/server ได้

The Interchange for Cross Protocol Support

SQL*Net มี multiprotocol interchange เพื่อที่จะใช้ในการเชื่อมต่อแบบที่ไม่เหมือนระบบเครือข่าย Interchange แบบ single ระหว่างระบบเครือข่ายที่มากกว่า 2 ระบบจะชี้ความต้องการในระบบเครือข่ายรวมจากโปรแกรมประยุกต์แบบ transparent network substrate และส่งผ่านไปยังจุดหมายปลายทาง การรวมกันของ SQL*Net กับ interchange จะยอมให้โปรแกรมประยุกต์ของ client สามารถติดต่อกับ server ของฐานข้อมูลใดก็ได้ โดยไม่ต้องใช้โปรโตคอลของระบบเครือข่ายบนเครื่องแต่ละเครื่อง

Protocol Support

- TCP/IP
- DECnet
- APPC (LU6.2)
- ISO/OSI
- SPX / IPX
- Named Pipes
- NetBIOS
- AppleTalk
- Banyan Vines
- XODIAC
- Wang System Network

นอกจากนี้ยังสนับสนุน SQL*Net Stream และ โปรโตคอลแบบ Packet-based ดังนี้

- VTAM (LU0)
- 3270 (LU2)
- Asynchronous (LU1, RS232)
- X.25

Support for Database Standards

SQL*Net สามารถใช้กับผลิตภัณฑ์ของการเชื่อมต่อของ Oracle SQL เพื่อให้มีการทำงานควบคุมข้อมูลระยะไกลกับฐานข้อมูลที่ไม่ใช่ ORACLE เช่น

- IBM 's DB2
- IBM 's SQL / DS
- DEC 's RMS

- Tandem 's Non-Stop SQL และด้วย SQL*Net ฐานข้อมูลของโปรแกรมประยุกต์ที่ไม่ใช่ ORACLE สามารถเชื่อมติดต่อกับระบบเครือข่ายโปรแกรมประยุกต์ได้

บทที่ 4

การวิเคราะห์ระบบจัดการฐานข้อมูลของ INGRES version 6.4

4.1 ARCHITECTURE ของ INGRES

เป็นสถาปัตยกรรมแบบ multi server architecture คือเป็นแบบ “multithreaded” กล่าวคือ ยอมให้มี DBMS process server หลาย process server ซึ่งเข้าถึงฐานข้อมูล เดียวกัน ซึ่งมีผลดีคือ

- ลด Overhead ต่อผู้ใช้แต่ละคนให้เหลือแค่ 30-50 K ในขณะที่แบบ single-threaded จะต้องใช้ถึง 200-400K
- ใช้ประโยชน์ของ multiprogramming ได้อย่างคุ้มค่าเมื่อเทียบกับแบบ processing model
- สามารถกำหนดคุณสมบัติซึ่งทำให้ระบบจัดการฐานข้อมูล สามารถทำงานได้อย่างมีประสิทธิภาพมากที่สุดเช่น ตั้ง priority ให้กับ on-line transaction processing server มากกว่า

4.2 SEMANTIC DATA CONTROL

4.2.1 View Management

View คือ การสร้างมโนภาพให้กับผู้ใช้จาก ฐานข้อมูล โดยผู้ใช้นั้นจะมองเห็นเฉพาะบางส่วนของข้อมูล และสามารถที่จะทำการแก้ไข base table ผ่าน view ได้โดย INGRES มีข้อยกเว้นสำหรับการแก้ไขผ่าน view คือ

- ไม่สามารถแก้ไขบน view ที่สร้างมาจากตารางมากกว่าหนึ่งตาราง
- ไม่สามารถแก้ไขคอลัมน์ ที่ source ไม่ได้เป็นชื่อคอลัมน์ธรรมดา เช่น คอลัมน์ที่เป็น set ของฟังก์ชัน หรือเป็นการคำนวณ (computation)
- ไม่สามารถทำการแก้ไขคอลัมน์ที่มี with check option เป็นตัวควบคุมอยู่

4.2.2 Data Integrity

INGRES จะยอมให้เข้าไปแก้ไขข้อมูลได้โดยจะต้องมีการตรวจสอบค่าของข้อมูลให้ถูกต้องและเหมาะสมอย่างอัตโนมัติภายใต้ data integrity และกฎต่าง ๆ

- Integrity อาจจะถูกกำหนดสำหรับ คอลัมน์ ที่ต้องการในตารางโดยผู้บริหารฐานข้อมูล หรือผู้ใช้

INGRES จะยอมให้มีการเก็บ referential integrity ,business rules, database rules, database event alerters ไว้ใน server ซึ่งวิธีนี้จะทำให้สามารถดูแล integrity ของข้อมูลจาก centralized location

การสร้าง integrity statement ใช้กำหนด integrities ซึ่งทำการตรวจสอบเพื่อให้แน่ใจว่า rows ที่ปรากฏในตารางนั้นเป็นไปตามสถานะที่กำหนด ซึ่งถ้าไม่เป็นไปตามที่กำหนดก็จะถือว่า reject ไป

Database procedure

คือกลุ่มของ SQL statment ที่มีการเรียกใช้บ่อยครั้ง และถูกนำมาจัดเก็บไว้เป็น object หนึ่งของ INGRES เพื่อความรวดเร็วหรือเมื่อเกิดเหตุการณ์ที่มีความผิดพลาดของข้อมูลเกิดขึ้น

การสร้าง procedure มี 2 ประเภทคือ

- Public

- Private

1) Public : ผู้บริหารฐานข้อมูลเป็นผู้สร้าง database procedure และผู้บริหารฐานข้อมูลจะเป็นผู้อนุญาตให้ผู้อนุญาตให้ผู้ใช้เรียกใช้ database procedure นี้

2) Private: ผู้ใช้เป็นผู้สร้าง database procedure และผู้ใช้ที่สร้างเท่านั้นที่สามารถเรียกใช้ database procedure ได้ ในการเรียกใช้ database procedure ใช้ได้ทั้งแบบ interactive หรือใช้กับ embedded SQL

ข้อดี 1.1) ลดจำนวนครั้งของการติดต่อระหว่างโปรแกรมประยุกต์ กับระบบจัดการฐานข้อมูล

1.2) ผู้บริหารฐานข้อมูล สามารถแก้ไขและควบคุมการเข้าถึงข้อมูลได้ในระดับ extra level

- ผู้บริหารฐานข้อมูล ต้องเป็น owner ของตารางที่อ้างถึงใน database procedure ที่ผู้บริหารฐานข้อมูลสร้างขึ้น และผู้บริหารฐานข้อมูลจะเป็นผู้กำหนดระดับของการเข้าถึงของผู้ใช้

1.3) Database procedure สามารถเรียกใช้ได้จากหลายโปรแกรมประยุกต์ ซึ่งลดเวลาในการทำโปรแกรม

กฎ (rules) หมายถึง กลไกซึ่งกำหนดขึ้นเพื่อเรียกใช้ database procedure เมื่อฐานข้อมูลเปลี่ยนแปลงไปตามที่กำหนดไว้ กล่าวคือ

- จะเรียกใช้ database procedure เมื่อสภาวะที่กำหนดเป็นจริง

- ผู้บริหารฐานข้อมูลหรือผู้ใช้จะเป็นผู้สร้างกฎ แต่ควรจะให้ผู้บริหารฐานข้อมูลเป็นผู้สร้าง และหากผู้ใช้ต้องการจะสร้างกฎสำหรับตารางของตนเอง ผู้ใช้ควรจะอยู่ภายใต้ความควบคุมของ ผู้บริหารฐานข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้คัดลอกเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กฎสามารถทำให้เกิด data Integrity ได้ดังนี้

1) Referential integrity

โดยจะดูแลความสัมพันธ์ระหว่าง 2 ตาราง เช่นค่าใน คอลัมน์ ของตารางหนึ่งจะต้อง match กับค่าใน คอลัมน์ ของอีกตารางหนึ่ง โดยตารางทั้งสองจะมีความสัมพันธ์แบบ parent child ซึ่ง คอลัมน์ ใน parent table จะเรียกว่า “primary key” ในขณะที่ คอลัมน์ ใน child table จะเรียกว่า “foreign key” มีวิธีการจัดการ 3 วิธีคือ

1.1) Reject method จะท

ข้องกับกฎ

1.2) Nullity method จะทำการ set entry ที่ถือว่าเป็น child ของอีก entry ให้มีค่าเป็น NULL

1.3) Cascade method จะทำการแก้ไข stament ทั้งหมดที่เกี่ยวข้องกับกฎให้มีค่าตรงกัน

2) General integrity

เป็น Integrity ใด ๆ ซึ่งไม่ใช่ referential integrity ทำหน้าที่อธิบายความสัมพันธ์ระหว่าง original data และ derive data

3) General purpose rules

เป็นกฎที่สามารถควบคุม external source ได้ด้วย โดยจะแน่ใจได้ว่าข้อมูลที่กำหนดให้ถูก update นั้นไม่ถูกเปลี่ยนแปลงโดยไม่ได้รับอนุญาต

เนื่องจากการสร้าง integrity บนคอลัมน์ซึ่งมีค่าเป็น NULL ตัวสภาวะจะพิจารณาถึงความเป็นไปได้ที่จะเกี่ยวข้องกับค่า NULL ด้วย และเนื่องจาก NULL ไม่มีค่าเป็นของตัวเอง ดังนั้นการเปรียบเทียบในสภาวะเงื่อนไขจึงให้ผลเป็น false จึงต้องมีการกำหนด integrity เพื่อให้จัดการกรณีที่คอลัมน์ ที่พิจารณาเป็น NULL สามารถระบุตั้งแต่สร้างตารางได้ว่า ข้อมูลที่อยู่ในตารางนั้นต้องมีค่าเป็น NULL

Database Events

จะทำให้โปรแกรมประยุกต์หรือระบบจัดการฐานข้อมูลของ INGRES บอกให้โปรแกรมประยุกต์อื่น ๆ ทำงานตามที่ได้กำหนดไว้ใน database procedure โดยทั่วไป event จะมีหลักการทำงานคือ

- ระบบจัดการฐานข้อมูลของ INGRES ให้ monitor โปรแกรมประยุกต์ที่ register ว่าจะรับ event เท่านั้น
- โปรแกรมประยุกต์ที่รับ event จะตอบสนอง event ที่เกิดขึ้นแล้วนำไปเป็นเงื่อนไขในการทำงานตามที่ระบุใน Program

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ไว้เพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ลำดับการใช้ even ใน conjunction ด้วยกฎของ INGRES คือ

- 1) สร้าง event โดยผู้บริหารฐานข้อมูล
- 2) Raising event สามารถเรียกใช้จาก interactive หรือ Embedded SQL หรือ database procedure ซึ่งจะทำหน้าที่ส่ง event ไปให้ server จากนั้น server จะส่ง dbevent ไปให้ผู้ใช้ที่ลงทะเบียนว่าต้องการรับ dbevent
- 3) ทำการระบุโปรแกรมประยุกต์ที่ต้องการรับ dbevent
- 4) รับ dbevent
- 5) ผู้ใช้ที่เป็นผู้สร้าง dbevent นี้เท่านั้นที่สามารถยกเลิกได้ เมื่อยกเลิก event แล้วจะไม่สามารถเรียกใช้ หรือ ทำให้เกิด dbevent ได้อีก

4.3 TRANSACTION MANAGEMENT

4.3.1 Concurrency

INGRES มีความสามารถในการทำ concurrency control โดยจะยอมให้ผู้ใช้หลาย ๆ คน เข้าถึงข้อมูลเดียวกันในเวลาเดียวกันโดยคำนึงถึง

- Concurrency หมายถึง การที่ผู้ใช้ทุกคนสามารถเข้าถึงข้อมูลใด ๆ ในเวลาเดียวกัน
- Consistency หมายถึง การที่สามารถแน่ใจได้ว่าการเปลี่ยนแปลงฐานข้อมูลจะทำให้เกิดค่าที่ถูกต้องเหมือนกันและอยู่ในลำดับซึ่งเป็นไปตามโครงสร้างของข้อมูล

INGRES ใช้ระบบ locking ในการจัดการการเข้าถึง resource ซึ่งประกอบไปด้วย database, table และ page เพื่อรับประกันความสอดคล้องของข้อมูลที่มีการใช้ร่วมกัน ระบบ locking ของ INGRES จะใช้กลไกที่เรียกว่า “MULTIPLE GRANULARITY” โดยมีการ lock ใน 2 ระดับคือ

- 1) Logical lock ซึ่งจะ lock อยู่ตราบเท่าที่ transaction ทำงานอยู่ ซึ่ง logical lock นี้ จะถูกยกเลิกก็ต่อเมื่อ transaction นั้นถูก commit, roll back หรือ abort
- 2) Physical lock เป็น lock ของระบบ lock ของ INGRES มีหน้าที่จัดการ การเข้าถึง resource ซึ่งอาจเกิดขึ้นในเวลาเดียวกันและ physical lock สามารถ lock และ unlock ภายใน transaction ซึ่งมีหลักการทำงานดังต่อไปนี้

Lock Modes

lock ของ INGRES มีการทำงานอยู่ 6 mode

- X : Exclusive lock หรือ write lock ซึ่งจะมี transaction เพียงคนเดียวเท่านั้นที่สามารถ lock resource ได้ผู้ใช้ที่ lock แบบนี้ เรียกว่า “Writer”
- S : Shared lock หรือ read lock ซึ่ง transaction หลาย ๆ transaction สามารถ lock resource ได้โดย transaction ที่ lock แบบนี้จะไม่สามารถ แก้ไขฐานข้อมูลได้ ผู้ใช้ ที่ lock แบบนี้เรียกว่า “Reader”
- IX,IS : Intended exclusive lock, Intend shared lock โดยเมื่อใช้ X,S lock page ในตารางใดแล้ว INGRES จะใช้ IX,IS lock ตาราง เพื่อแสดง granularity โดย IX-lock จะแสดงว่า page ในตารางนี้อาจมีการแก้ไข และ IS-lock จะแสดงถึง page ในตารางที่กำลังถูกอ่าน
- SIX : Shared intended exclusive lock ซึ่ง buffer manager ของ INGRES จะใช้ในการแก้ไข pages ใน cache ของ buffer manager
- N : Null lock คือไม่มีการ lock โดยไม่สนใจความผิดพลาดที่อาจเกิดขึ้น

ระดับของการ lock

ระดับของการ lock จะหมายถึง ขอบเขตของ resource ซึ่งต้องการ lock แบ่งออกเป็น

- Page : ควบคุมการเข้าถึงในระดับ page
- Table : ควบคุมการเข้าถึงในระดับตาราง
- Database : ผู้ใช้ block โดย exclusive block ซึ่งทำให้ไม่สามารถติดต่อกับฐานข้อมูลเองได้ และจะแสดงข้อความแสดงความผิดพลาดเพื่อชี้ว่ามี exclusive lock อยู่
- Control : เป็น physical lock ที่ใช้จัดการตารางในขณะที่กลไกมีการเปลี่ยนแปลง

ผู้บริหารฐานข้อมูลจะจัดการเกี่ยวกับข้อกำหนด (configuration) ของระบบ locking โดยจะตั้งจำนวนครั้งของการ lock ที่เกิดขึ้น และเมื่อมีการ lock เกิดขึ้น ค่าที่ตั้งไว้จะลดลงทีละ 1 เมื่อใดที่ INGRES จะ lock นั้นขึ้นอยู่กับ mode ของ lock ที่ transaction อื่น ๆ ใช้ lock resource อยู่ซึ่งสามารถ check ได้จากตารางแสดงโหมดของการ lock

Deadlock

Deadlock เป็นสภาวะที่เกิดการรอคอยการใช้ resource อย่างไม่มีกำหนด ซึ่งเกิดขึ้นเมื่อ transaction หนึ่งรอ lock ซึ่งอีก transaction หนึ่ง lock อยู่และเป็นเวลาเดียวกับที่ transaction นั้นรอ lock ที่ transaction แรก hold อยู่

เมื่อเกิด Deadlock ระบบ locking ของ INGRES จะยกเลิก transaction ใด transaction หนึ่งเพื่อให้อีก transaction ทำงานต่อไป ส่วน transaction ที่ถูกยกเลิกจะได้รับข้อความที่บอกความผิดพลาด จากนั้นทุก ๆ statement ใน transaction ตั้งแต่การ commit ครั้งสุดท้ายจนถึงจุดที่เกิด deadlock จะถูก recovery มาเนื่องจาก INGRES ใช้การ lock แบบ page-level locking ดังนั้น ถ้ามีอย่างน้อย 2 transaction ที่ต้องการจะเข้าถึงฐานข้อมูล และมีผู้ใช้อย่างน้อย 2 คน พยายามจะเปลี่ยนแปลง row แล้วอาจจะเกิด deadlock ใน query เดียว ๆ ได้หาก

- 1) มีการเข้าถึง pages โดยใช้ path ที่แตกต่างกันในตาราง หมายความว่า transaction หลาย transaction พยายามจะแก้ไขตารางโดยใช้ access path ที่แตกต่างกัน เช่น ในตาราง หนึ่งมี isam structure ซึ่ง index คนละฟิลด์กับที่ hash index
- 2) เกิด lock escalation ซึ่งมีสาเหตุมาจากกรณีใดกรณีหนึ่ง ต่อไปน
 - Convert จาก shared lock เป็น exclusive lock

- Long overflow chains : ตารางที่มี overflow pages เกิน จะทำให้เกิดปัญหา เพราะ INGRES จะต้องหา overflow pages เหล่านั้น และจะ lock แต่ละ page และเก็บรักษาไว้เพื่อลด overflow chain
- มีการ lock เกินค่า maxlock ของระบบ
- Transaction อยู่ใน lock limit
- Index ของ Btree เกิดการ split : กรณีที่มีผู้ใช้หลายคน แทรกเพิ่มลงบน Btree table เล็ก ๆ ก็อาจทำให้เกิดได้

INGRES มี feature ในการจัดการ deadlock ด้วยโปรแกรมประยุกต์ มีโปรแกรมประยุกต์ที่ใช้ในการจัดการ check deadlock หลังจากแต่ละ statement ของ multiple query transaction ถ้าเกิด deadlock เมื่อทำ statement transaction ใดทำอยู่ก็จะ abort transaction นั้นจนกว่าจะไม่เกิด deadlock โปรแกรมประยุกต์นี้เขียนด้วย Embedded SQL/FORTRAN

วิธีที่ INGRES ใช้ในการจัดการ deadlock ในสถานะที่เป็น heavy concurrent แบ่งเป็น 2 วิธีคือ

- 1) Table lock approach : เพื่อให้เกิด deadlock น้อยที่สุดเมื่อมี concurrent activity เกิดขึ้นมาก ๆ ในตารางเล็ก ๆ หรือ ส่วนหนึ่งในตารางใหญ่ ๆ ซึ่งจะทำให้ผู้ใช้ แต่ละคนมี ลำดับการเข้าถึงตาราง ถึงจะเป็นการเสียเวลาแต่สามารถหลีกเลี่ยงสภาวะ deadlock ได้
- 2) Never escalation approach : จะใช้ในกรณีที่ผู้ใช้ทำงานในส่วนต่าง ๆ ของตาราง ถึงแม้ว่าเป็นเพียงการรัน query หรือแก้ไขง่าย ๆ แต่ให้มีการใช้ primary และ secondary index มาก ๆ เพื่อทำให้ผู้ใช้สามารถใช้ตารางเดียวกันได้โดยไม่มีผู้ใช้ใด ใช้วิธี lock เพื่อไม่ให้ผู้ใช้คนอื่นใช้

4.3.2 Backup and Recovery

จะมีวิธีต่าง ๆ ดังนี้

Log - based Recovery

INGRES ใช้วิธีการ fast commit หรือแบบ defered mode โดย INGRES ใช้ commercial term ว่า fast commit ,log file ของ INGRES จะเก็บ log record ซึ่งเป็นการเปลี่ยนแปลงระหว่างการทำ transaction หนึ่ง ๆ ตั้งแต่เริ่มต้น transaction จนถึง commit ซึ่งถ้า transaction commit ก็จะมีสิ่งที่อยู่ใน log file นั้นเก็บลง journal file

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Journal file เป็นไฟล์ที่ INGRES เอาไว้เก็บการเปลี่ยนแปลงทั้งหมดตั้งแต่ checkpoint ครั้งสุดท้ายเป็นต้นมา โดยจะเก็บวันและเวลาขณะเกิดการเปลี่ยนแปลงในฐานข้อมูล

Log file ต่างกับ journal file ตรงที่ journal file จะเก็บข้อมูลน้อยกว่า log file โดยที่ journal file จะเก็บข้อมูลเฉพาะตารางที่สั่งให้ทำ journaling เท่านั้น

Fast commit และ Write behind

ในการทำ transaction ที่มีการแก้ไขฐานข้อมูลนั้น จริง ๆ แล้ว transaction ไม่ได้ทำการแก้ไขที่ฐานข้อมูลโดยตรงแต่จะทำการแก้ไขไปที่ buffer แทนพร้อมทั้งมีการ write การเปลี่ยนแปลง log file ซึ่งก็คือหลักการของ fast commit และหลังจากที่มีการเปลี่ยนแปลงใน buffer แล้ว ก็จะต้องมีการ write การเปลี่ยนแปลงนั้นลงฐานข้อมูลจริง ๆ ซึ่ง INGRES เรียกการ write นี้ว่า “write behind” โดย INGRES จะทำการ write behind ก็ต่อเมื่อ server ว่าง

Checkpoint ของ INGRES นั้นหมายถึงการ backup ฐานข้อมูลปัจจุบันที่มีอยู่ทั้งหมดลงบน binary file ใน stable storage เรียกว่าเป็นการทำ static backup ผู้บริหารฐานข้อมูล จะเป็นผู้ทำคำสั่ง checkpoint เท่านั้น ซึ่งในการทำ checkpoint นั้นถ้ายังมีผู้ใช้งานฐานข้อมูลอยู่ INGRES จะรอจนผู้ใช้นั้นใช้งานเสร็จ และข้อมูลที่อยู่ใน buffer ได้ write behind ลงบนฐานข้อมูลแล้วจึงจะทำการ backup ส่วนผู้ใช้ที่ไม่ได้ใช้งานฐานข้อมูล และต้องการที่จะเข้ามาใช้ในขณะ checkpoint นั้นจะไม่มีสิทธิเพราะตารางจะโดน lock ไว้ต้องรอให้ checkpoint เสร็จจึงจะเข้ามาเรียกใช้ตารางได้

Checkpoint จะเป็นการ backup ฐานข้อมูลที่มีอยู่เท่านั้น ถ้าหากฐานข้อมูลนั้นถูกทำลายไป (destroy) จะเป็นการ delete checkpoint ไปด้วย ดังนั้นจะไม่สามารถสร้างฐานข้อมูลนั้นขึ้นมาใหม่จาก checkpoint ได้

Journaling คือการเก็บการเปลี่ยนแปลงของฐานข้อมูลตั้งแต่ checkpoint ครั้งสุดท้ายลงบน journal file เรียกว่าเป็นการทำ dynamic backup ซึ่งในการ recover จะสามารถกู้ข้อมูลได้จาก static backup ที่ checkpoint ครั้งสุดท้ายมารวมกับ dynamic backup ที่ได้จากการทำ Journaling หลัง checkpoint ซึ่งจะทำให้สามารถกู้ข้อมูลกลับมากได้อย่างสมบูรณ์

การทำ journaling นั้นจะต้องบอกเอาไว้ตั้งแต่ตอน สร้างตารางแล้วว่าจะให้ทำที่ตารางใดบ้าง ซึ่งส่วนมากจะทำเฉพาะตารางที่สำคัญ และมีการเปลี่ยนแปลงบ่อยๆ เช่นใน ตารางการฝากถอน เงิน ส่วนตารางที่ไม่ได้เปลี่ยนแปลงบ่อยๆ เช่น เก็บข้อมูลพนักงานก็ไม่จำเป็นต้อง set journaling ซึ่งตารางที่ไม่ได้มีการทำ journaling ไว้ั้นการกู้ข้อมูลก็จะทำได้ถึงแค่เวลา checkpoint ครั้งสุดท้ายเท่านั้น

File Storage

การเก็บไฟล์ของ INGRES จะเป็นแบบ multiple file storage คือเก็บตารางหนึ่งเป็นหลาย ๆ ไฟล์ ซึ่งมีข้อดีในแง่ของการ recovery คือถ้าเกิดไฟล์เสียไปไฟล์หนึ่งก็จะเป็นความเสียหายเล็กๆ เพราะไม่ได้เสียหายทั้งตารางและไฟล์นั้นเพียงไฟล์เดียวที่ต้องทำการกู้กลับมาจาก backup ซึ่งจะช่วยให้ recovery time เร็วมาก

Feature ในการ commit เพิ่มเติม

1) Group commit หรืออีกชื่อคือ piggybacked commit คือเป็นการ group commit information จากหลายๆ ผู้ใช้รวมกันเพื่อประโยชน์ในการลด transaction log file bottlenecks และ เป็นการลดปริมาณการใช้ I/O ด้วย

2) Two-phase commit เป็น feature สำหรับ multuser คือเมื่อผู้ใช้ในฝั่งหนึ่งทำการ commit แล้วอีกฝั่งหนึ่งก็จะ commit ด้วยหรือถ้าฝั่งหนึ่ง rollback อีกฝั่งก็จะ rollback ตาม

ข้อดีของการ backup แบบ checkpoint คือ เก็บเป็นไฟล์เดียวไม่เปลืองที่

ข้อเสียของการ backup แบบ checkpoint คือ เนื่องจากเก็บไฟล์เป็น binary จึงเป็น machine dependent ถ้าข้ามเครื่องจะใช้ checkpoint ไม่ได้

2) Backup โดยการ copy ฐานข้อมูล

จะเป็นการ backup ฐานข้อมูลที่เป็นของผู้ใช้ไม่ใช่ฐานข้อมูลทั้งหมดโดยใช้คำสั่ง copydb ของ INGRES ซึ่งจะเก็บเป็นแบบ text file

วิธีนี้จะมีข้อดีตรงที่เป็นการ backup สำหรับผู้ใช้ที่ไม่ได้เป็นผู้บริหารฐานข้อมูล ซึ่งทำให้ผู้ใช้เหล่านั้นสามารถ backup ตาราง , view และ procedure ของตัวเองได้

3) Backup โดยการ Unloaded และ Reloaded

จะคล้ายกับการ backup แบบ copy และเก็บเป็น text file เช่นเดียวกับการ backup แบบนี้จะเป็นการ copy เก็บทั้ง system table และ user table ทั้งหมดซึ่งใน system table จะประกอบด้วย

ส่วนของ server ซึ่งเป็นส่วนสำคัญได้แก่

- เก็บตารางทุกตาราง
- เก็บรายละเอียดของผู้ใช้
- เก็บ view
- เก็บ database procedure

ส่วนของ front end เช่น

- เก็บ QBF name (form + ตาราง)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับเก็บ join dept ระหว่างตารางเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.4 QUERY PROCESSING

4.4.1 Structure storage

Structure storage คือ การจัดไฟล์เพื่อใช้ในการเข้าถึงข้อมูลในตารางการที่มีคีย์จะทำให้สามารถเข้าถึงข้อมูลได้รวดเร็วกว่าไม่มีคีย์

INGRES มี storage structure อยู่ 4 แบบ คือ

- Heap เป็นการเข้าถึงข้อมูลแบบ sequential
- Hash มีหลักการในการเลือก address โดยใช้ค่าของคีย์ข้อมูลเป็นตัวกำหนด
- Isam จะจัดเรียงข้อมูลโดยใช้ค่าของคีย์คอลัมน์ สำหรับการเข้าถึงแบบเร็ว บนค่าที่แน่นอนและการดึงข้อมูลเป็นช่วง, index เป็นแบบ static และต้องการการแก้ไขทุก ๆ ครั้งที่ตารางมีขนาดใหญ่ขึ้น
- Btree จะจัดเรียงข้อมูลโดยใช้ค่าของคีย์คอลัมน์ สำหรับการเข้าถึงแบบเร็ว บนค่าที่แน่นอนและการดึงข้อมูลเป็นช่วง, index เป็นแบบ dynamic และจะโตตามขนาดตารางที่ใหญ่ขึ้น

Heap

จะไม่มีคีย์ จะเป็นลักษณะการเก็บข้อมูลแบบต่อ ๆ กันไป ดังนั้นเมื่อเพิ่มเติม row เข้า ไปจะถูกเก็บไว้ท้ายสุด

Heap เป็น storage structure ที่ดีหากใช้ในกรณีที่

- 1) Load ข้อมูลจากตาราง เนื่องจากเป็น storage structure ที่เพิ่มเติมข้อมูลได้เร็วที่สุด เนื่องจากจะเพิ่มเติมข้อมูลเข้าไปที่ท้ายของ Heap ดังนั้นจึงไม่มี overhead ที่เกิดจากคำนวณที่ใส่ข้อมูล
- 2) เป็นตารางที่มีขนาดเล็ก (มี Page ไม่มากต่อตาราง)
- 3) ดึงข้อมูลทุก ๆ row โดยไม่ต้อง sorting
- 4) ใช้ secondary index กับตารางที่มีขนาดใหญ่และต้องการรักษาพื้นที่ไว้

ข้อเสียของ Heap storage structure และ วิธีแก้ปัญหา

1) การเข้าถึงตารางใหม่ที่สร้างโดยคำสั่ง “ create Table As Select ” นั้นจะช้า แก้ปัญหาโดยใช้ clause “ with structure = type ” เมื่อ execute คำสั่ง create

2) Space ที่ deleted rows จะไม่นำมาใช้ใหม่ แก้ปัญหาโดยใช้ statement modify เพื่อนำ space นั้นมาใช้ใหม่

3) Select และแก้ไขช้ามากเพราะเข้าถึงแบบ sequential แก้ปัญหาโดยถ้าเป็นตารางขนาดเล็กก็ไม่ใช่ปัญหา แต่ถ้าเป็นตารางขนาดใหญ่ก็สร้าง secondary index

Hash

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เป็นโครงสร้างที่ใช้ match key value โดย storage structure แบบนี้จะต้องมีการกำหนดคีย์ ซึ่งถ้าไม่มีการกำหนดคีย์ จะใช้ฟิลด์แรกเป็นคีย์

Hash เป็น storage structure ที่ดีหากใช้ในกรณีที่

- 1) ดึงข้อมูลโดยใช้ key value ที่กำหนด
- 2) ข้อมูลที่ต้องมีการคำนวณ เนื่องจาก storage structure แบบนี้ สามารถคำนวณได้อย่างรวดเร็ว

ข้อเสียของ Hash storage structure และวิธีแก้ปัญหา

- 1) ไม่เหมาะกับรูปแบบที่เป็นการ match เพราะ ประสิทธิภาพ จะลดลง
แก้ปัญหาโดยใช้ Isam, Btree แทน
- 2) ไม่เหมาะกับการดึง range of values เพราะประสิทธิภาพจะลดลง
แก้ปัญหาโดยใช้ Isam, Btree แทน
- 3) ไม่ควรใช้กับส่วนของ multi-column key เพราะ ประสิทธิภาพ จะลดลง
แก้ปัญหาโดยใช้ Isam, Btree แทน
- 4) มี overflow page ใน ตารางที่แก้ไขใหม่
แก้ปัญหาโดยถ้าคีย์เป็น unique, algorithm ของ Hash อาจจะสามารถกระจายข้อมูลได้ไม่ดีถ้าคอลัมน์เป็น character column ก็ให้ใช้ Isam แทน
- 5) มี overflow page หลังจากมีการเพิ่มเติม row
แก้ปัญหาโดยแก้ไขใหม่หรือใช้ dynamic structure แบบ Btree

Isam

เป็น storage structure ซึ่งใช้ match key value แต่จะสมบรูณ์กว่า hash เนื่องจากสามารถดึงช่วงของ key values ได้อีกทั้งยังทำการ match แบบมีรูปแบบได้และใช้ partial key และจะต้องมีการกำหนดคีย์ ซึ่งถ้าไม่มีกำหนดคีย์ จะใช้คอลัมน์แรกเป็นคีย์

Isam จะใช้ static index ซึ่งไปยัง static numbers ของ page หลักซึ่งเป็น static เช่นเดียวกัน หากต้องการเปลี่ยนแปลงตารางจะถูกจัดลำดับโดยคีย์ที่กำหนด ดังนั้นจะสร้าง sparse index ขึ้น ซึ่ง index จะประกอบไปด้วยช่วงของคีย์ และ pointer ไปยัง index pages หรือ data page

ถ้ามีการเพิ่มเติม row ให้แก้ไขใหม่ เพื่อเป็นการหลีกเลี่ยง overflow page ดังนั้นสำหรับ static table, storage structure แบบ Isam อาจจะเหมาะสมกว่า Btree Isam เป็น storage structure ที่ดีหากใช้ในกรณีที่

- 1) เมื่อตารางเป็นแบบ static

- 2) ดึงข้อมูลแบบ pattern matching โดยช่วงของ key values และเป็นส่วนท้ายสุดของ multi-column เท่านั้น
- 3) สามารถรองรับการทำงานที่มีผู้ใช้เข้าถึงตารางพร้อม ๆ กันหลาย ๆ คน

ข้อเสียของ Isam storage structure

- 1) ไม่เหมาะกับตาราง ที่มีการขยายตัวอย่างรวดเร็วเพราะจะทำให้เกิด overflow page
- 2) ไม่เหมาะกับ ตาราง ที่มีขนาดใหญ่เกินกว่าจะ modify เพราะจะทำให้เกิด overflow page
- 3) คีย์ที่เป็นแบบ sequential
- 4) หากใช้ pattern matching แล้วไม่ระบุ character ท้ายสุด จะทำให้ต้อง scan ทั้งตาราง เพราะหากไม่ระบุค่าจะไม่สามารถเปลี่ยนแปลงสภาวะการค้นหาได้

Binary tree (Btree)

เป็น storage structure ที่ดีที่สุด สามารถใช้การเข้าถึงโดยคีย์ สามารถรองรับการค้นหาแบบเป็นช่วง และ การ match แบบมี pattern

- index ของ Btree เป็นแบบ dynamics จะเพิ่มตามการเพิ่มขนาดของตาราง
- ใช้ sparse index ซึ่งไปยัง Page ใน leaf level
- ใน leaf level จะเป็น dense index ซึ่งมี คีย์ ซึ่งไปยัง row บน data page ในตาราง

Index Growth

ข้อแตกต่างที่ระหว่าง Isam กับ Btree คือ

- Btree นั้นใช้ dynamic index กล่าวคือ dynamic index จะเพิ่มตามการเพิ่มขนาดของตารางในขณะที่ Isam นั้นใช้ static index
- Isam จะไม่มีการจำกัด page สุดท้ายของตารางดังนั้นถ้ามีการเพิ่มเติมข้อมูลที่เป็นคีย์ลงไป ใน page นั้นเรื่อย ๆ อาจทำให้เกิด overflow page ที่ page สุดท้ายได้ในขณะที่
- Btree จะมีการจำกัด leaf page ที่เป็น page สุดท้ายไว้หากมีการเพิ่มเติมข้อมูลที่เป็นคีย์จน page นั้นเต็มก็จะเกิดการแบ่งแยกออกเป็น 2 leaf pages ทำให้ไม่เกิด overflow pages และใน index page ก็ใช้หลักการเดียวกัน เพราะฉะนั้นจะเห็นได้ว่า index จะเพิ่มตามขนาดของตาราง

Locking และ Btree table

โดยปกติแล้ว leaf และ data page จะถูก lock จนกระทั่งจบ transaction หากมีการ lock เกิดขึ้น ที่ leaf level ในขณะที่ค้นหา Btree index จะเกิดการ lock ใน index page ตามเส้นทางที่จะไป leaf page นั้นและทำการ lock แบบชั่วคราวตามลำดับ level ที่เป็น Parent level ของ leaf page ที่เราต้องการจะเข้าถึง

Delete Rows: Data Pages

ถ้า row ที่ถูกลบอยู่บน data page ที่มีลักษณะเป็นแบบ associated เมื่อที่ว่างนั้นจะถูกนำมาใช้จะถูกนำมาใช้อีกครั้งแต่ถ้ามีหลาย ๆ row ที่อยู่บน data page ที่มีลักษณะเป็นแบบ nonassociated และถูกลบก็จะนำข้อมูลแต่ละ page ไปวางไว้ที่ free list

- ข้อดี
- 1) สามารถใช้กับ ตารางที่มีอัตราการโตอย่างรวดเร็ว
 - 2) สามารถใช้กับ pattern matching
 - 3) สามารถใช้กับการ ดึง key value แบบเป็นช่วง
 - 4) สามารถ ดึง โดยใช้เพียงแค่ leftmost ของ multi-column key เท่านั้น
- ข้อเสีย
- 1) ไม่เหมาะที่จะใช้กับตารางที่มีลักษณะเป็นแบบ relatively static
 - 2) ไม่เหมาะที่จะใช้กับตารางที่มีขนาดเล็ก และมีคนใช้พร้อม ๆ กันหลายคน
 - 3) หากจะใช้การ match แบบมีรูปแบบแต่ไม่กำหนด character ตัวซ้ายสุด จะทำให้ต้อง scan ทั้งตาราง ซึ่งเป็นผลให้เสียเวลา
 - 4) หากจะใช้บางส่วนของ multi-column key แต่ไม่ระบุ คอลัมน์ ข้ายสุด จะทำให้ต้องสร้าง secondary index ที่เป็นคอลัมน์ ที่ใช้ค้นหาอยู่
 - 5) จะทำให้ลบบ่อยพอกับเพิ่มเติมข้อมูล

สรุปเปรียบเทียบกรณีที่ใช้ Isam แทนที่จะใช้ Btree storage structure

- 1) กรณีที่ใช้ static table(ตารางที่มีขนาดคงที่)
- 2) ใช้ disk operation ในการเข้า data page น้อยกว่า Btree เพราะ Btree ต้องทำถึง leaf level
- 3) ไม่ต้อง lock จาก index page ดังนั้น concurrency จะดีกว่า
- 4) Sequential traversal ของตารางแบบ Isam ที่ไม่เรียงลำดับจะทำได้เร็วกว่า Btree เพราะ Isam มี pointer ที่ชี้ไปยัง data page โดยตรง

สรุปเปรียบเทียบกรณีที่ใช้ Btree แทนที่จะใช้ Isam storage structure

- 1) ตาราง มีอัตราการเพิ่มขนาดอย่างรวดเร็ว (อาจทำให้เกิด overflow ใน Isam storage structure)
- 2) ใช้เรียงลำดับได้ง่ายเพราะทำการเข้าถึงแบบ sequential

3.4.2 Query Optimization

INGRES's intelligent query optimizer เป็นชื่อทาง INGRES ตั้งเอาไว้เพื่อแสดงความสามารถในการเป็น expert system ของตัว optimizer ซึ่งจะเป็นตัวที่พิจารณาหาวิธีที่ดีที่สุดสำหรับการทำ query แต่ละครั้ง โดยดูจากสถิติในการหาเส้นทางในครั้งก่อนๆ และใช้ heuristic ซึ่งข้อดีของ query optimizer นี้คือการเป็น syntax-independent คือ INGRES จะทำการ normalize query ให้อยู่ในรูปแบบที่ common ก่อนการทำ optimization ดังนั้นประสิทธิภาพของการ query จะไม่ขึ้นอยู่กับลำดับข้อความของ query

เช่น การ select * from A,B จะเท่ากับ select * from B,A

การ optimizedb

Optimizedb จะมีผลกับ speed ของการทำ query processing แต่ไม่ได้บอกว่า query นั้นสามารถ execute ได้หรือไม่ ซึ่งถ้ามีการเก็บค่า statistics ที่ถูกต้อง และสมบูรณ์แล้วจะทำให้ประสิทธิภาพของการ query ดีขึ้นและมีผลทำให้ ประสิทธิภาพ ของ ระบบเร็วขึ้น

ถ้าไม่มีการ run optimizdb แล้ว optimizer ก็จะสรุปว่า

1) ในการจับคู่ที่เป็น exact match ทั้งหมดในตารางจะมี row return ออกมา 1% ของตาราง ยกเว้นกรณีที่ดีๆ หรือ index ถูกระบุให้ unique ซึ่งในกรณีนี้จะมีเพียง row เดียวที่ถูก return กลับมาสำหรับ attribute ที่ถูก index ไว้เช่น

WHERE emp.empno = 275

2) นอกเหนือจากกรณีการจับคู่ที่เป็น exact match แล้ว กรณีอื่นจะมี row return 10% ของตาราง ดังนั้นถ้ามี qualification ที่ไม่เป็น exact อยู่ 3 อันก็จะมีจำนวนตารางที่ถูก select ดังนี้คือ

$$1/10 * 1/10 * 1/10 = 1/1000$$

3) join ทุกอันจะถูกสรุปให้เป็น 1:1 โดยมี base อยู่บน data set ของตารางอันที่เล็กกว่า เช่น เมื่อตาราง 1 (มี 100 rows) ถูก join กับ ตาราง 2 (มี 1000 rows) ผลลัพธ์โดยประมาณก็จะได้ 100 rows

4) กรณีที่มีเงื่อนไขการ join ตารางแล้ว จำนวน row ผลลัพธ์จะมากกว่า หรือเท่ากับ ของ 10% lower bound ของ row ที่มาจากตารางที่เล็กกว่า

สิ่งที่เก็บอยู่ใน optimizedb เป็นสิ่งที่เก็บไว้สำหรับทำ query execution plan ได้แก่

- 1) จำนวนของ row ในตาราง ซึ่งโดยปกติจะมีอยู่ใน system catalog อยู่แล้ว
- 2) จำนวนของ unique value ในคอลัมน์หนึ่ง ๆ
- 3) เก็บค่า NULL count ซึ่งแสดงถึงเปอร์เซ็นต์ของ column value ที่เป็น NULL

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4) จำนวนของ duplicate value ใน คอลัมน์หนึ่ง ๆ โดยเฉลี่ยซึ่งค่านี้เรียกว่า repetition factor

5) histogram ที่แสดง data distribution ซึ่งจาก histogram เราจะได้ค่า minimum และ maximum สำหรับคอลัมน์หนึ่ง ๆ

Information ต่าง ๆ เหล่านี้จะถูกใช้โดย INGES query optimizer เพื่อคำนวณ cost ของ QEP แต่ละอัน

ชนิดของ statistics

1) Non-sampled statistic คือทุก ๆ row บนตารางจะถูกดึงออกมา

2) Sampled statistics คือ subset ของ row จากตารางถูกดึงออกมา

ใน default mode ของ optimizedb จะเป็นแบบ non-sampled คือทุก ๆ row ของตาราง จะถูกใช้ในการ generate statistics ซึ่ง mode นี้จะได้รับ statistic ที่ถูกต้องมากที่สุดเท่าที่จะเป็นไปได้เพราะมีการพิจารณาข้อมูลทุก ๆ ค่า

ใน mode ของ sampled statistics นั้นจะเหมาะกับ base table ที่ใหญ่มาก ๆ ซึ่งใน optimizedb นี้จะสามารถระบุเปอร์เซ็นต์ของ row ที่จะถูก sample ได้และถ้าใช้การ sampling ที่เพียงพอก็จะทำให้ได้ statistics ที่คล้ายคลึงกับการทำ statistics บน full table

นอกจากนี้ทั้ง Non-sampled และ sampled ยังมี option ในการทำ statistics ที่เกี่ยวกับการกระจายของข้อมูลที่ statistic จะ hold ได้คือ

1) Full statistics จะ copy information ทั้งหมดเกี่ยวกับ data distribution ซึ่ง cost จะสูงที่สุด สำหรับแต่ละคอลัมน์ของตารางจะถูก scan เพียงครั้งเดียวและค่าของคอลัมน์จะถูกดึงออกมาทำ sorted order ดังนั้นต้องมีการสร้าง temporary table ที่มีการ sort data process ของการทำ full statistics และสามารถทำให้สั้นลงได้โดยใช้ option ของการ sampling (คือใช้ Sampled statistic) ซึ่งจะทำให้สามารถเลือกได้ว่าจะพิจารณา row ออกมากี่ % ของตารางแล้วทำ full statistics

2) Minmax statistics เป็นวิธีที่ถูกกว่า full statistics โดยปกติจะทำการ scan เพียงครั้งเดียวสำหรับทั้งตาราง และ statistics นี้จะสร้างจาก information ของค่า minimum และ maximum สำหรับแต่ละคอลัมน์

Minimum ก็มีการใช้ option ของทั้ง sampled และ non-sampled ได้ซึ่งถ้าใช้ non-sampled ก็จะได้ข้อมูลที่ถูกต้องกว่าแต่ใน sampled ก็จะทำให้ process สั้นลงได้ซึ่งสามารถเลือกได้ว่าจะให้พิจารณา row กี่ % ของตาราง ทั้งหมดแล้วนำเอา row ส่วนที่ดึงออกมานั้นทำ minmax statistics

นอกจากนี้ทั้ง full statistics และ minmax statistics ยังมี option ให้เลือกอีกคือ key column statistics ซึ่งเป็นการทำ full หรือ minmax บนคีย์ หรือคอลัมน์ที่ถูก index ไว้เท่านั้น

Input Text File Statistics

Statistics สามารถอ่านเข้ามาจาก text file ได้โดย input text file ต้องอยู่ใน format ที่ตายตัวซึ่งไฟล์นี้สามารถที่จะถูกแก้ไข เพื่อบอกการเปลี่ยนแปลงในการกระจายของข้อมูลตามความต้องการก่อนที่จะทำการ optimizedb ซึ่งจะมีประโยชน์เมื่อ

- 1) Information กำลังถูกย้ายจากฐานข้อมูล หนึ่งไปยังอีกอันหนึ่ง (ใช้คำสั่ง copy) และต้องการที่จะย้าย statistics สำหรับตารางนั้นไปด้วย
- 2) รู้การกระจายของข้อมูลในตาราง และต้องการที่จะอ่านหรือใส่ค่าลงไปโดยตรงแทนการให้ optimizedb ทำการสร้างให้

Query Execution Plan (QEP)

เมื่อ INGRES optimizer ทำการ execute query นั้นจะมีการ generate QEP ที่แสดงถึงว่า query จะถูก execute อย่างไร เมื่อ QEP ถูก generater ขึ้นมาครั้งหนึ่ง INGRES สามารถใช้มันได้มากกว่า 1 ครั้งเพื่อ execute database query รูปแบบนี้

QEP นี้สามารถ print ออกมาได้ว่า query ทำงานอย่างไรที่ถูกควบคุมจาก INGRES optimizer

Compiled Database Procedures

ใน INGRES จะมี compiled database procedures ซึ่งเป็นฟังก์ชัน ที่ถูกเขียนขึ้นมาโดย INGRES/4GL ซึ่งถูก compile เอาไว้เรียบร้อยแล้วและจัดเก็บ transaction เหล่านี้เอาไว้ใน shard memory

ข้อดีของการมี compiled database procedure คือ

- 1) ลดการใช้ CPU เนื่องจากเป็นการ compile เก็บเอาไว้
- 2) Transaction ที่ถูก compile แล้วจะเก็บเอาไว้ใน share memory ทำให้ลดปริมาณ memory ที่ใช้ลงแทนการเก็บสำเนาเอาไว้ให้แต่ละผู้ใช้
- 3) ลด network overhead โดยแทนที่จะต้องเขียน SQL ยาว ๆ เพื่อทำ query ก็สามารถส่งเพียง procedure name และ parameter ต่าง ๆ ไปให้ compiled database procedure เท่านั้นทำให้ลดการ communication ระหว่าง client/server และ network traffic ต่าง ๆ ลงได้
- 4) จะเกิดการ enforce integrity คือผู้ใช้ และ programmer จะไม่ได้เข้าไปในข้อมูลโดยตรง แต่ใช้การแก้ไขและดึงข้อมูลผ่านทาง procedure

ส่วนในการทำ interpret INGRES จะมี feature หนึ่งที่ใช้ในการจัดการ knowledge ภายใน database server คือ feature ทางด้าน resource control

Resource control feature ของ INGRES query optimizer จะเป็นตัวจำกัดการใช้ query ของผู้ใช้คือ จะมีการทำนายประสิทธิภาพของระบบ โดยการป้องกันไม่ให้เกิด query ที่ return row ออกมาไม่รู้จบจะมีการจำกัด row ที่ยอมให้ return ออกมา และยังมีความสามารถในการ control การใช้ disk I/O ด้วย

4.4.3) Data Independence

Physical Data Independence

จะพิจารณาตามหัวข้อต่อไปนี้

- Representation of numeric data ใน INGRES ไม่มีคุณสมบัตินี้ เพราะข้อมูลจะมีการเก็บแบบ binary เท่านั้น ผู้บริหารฐานข้อมูลไม่มีสิทธิ์เลือก
- Unit for numeric data ใน INGRES ไม่มีคุณสมบัตินี้ เพราะไม่มีการเก็บหน่วยของข้อมูล จะเก็บเฉพาะ value
- Data coding ใน INGRES ไม่สามารถเปลี่ยนโดยให้ program รับรู้เองได้
- Structure of stored file ใน INGRES มีคุณสมบัตินี้

Logical Data Independence

INGRES มีคุณสมบัติข้อนี้คือ ผู้บริหารฐานข้อมูลสามารถที่จะเปลี่ยนแปลงตาราง เพิ่ม คอลัมน์, ตัดคอลัมน์, เพิ่มตาราง สิ่งเหล่านี้จะไม่มีผลต่อการเข้าถึงตารางของผู้ใช้โดยไม่ต้องมีการแก้ไขโปรแกรมใหม่

4.5 ADDITIONAL TOOLS

Tools set ของ INGRES สามารถแบ่งออกเป็น 3 พวกใหญ่ๆ คือ

- 1) End-user Tools
- 2) Application Development Tools
- 3) CASE Tools

4.5.1 End-user Tools

INGRES มีเครื่องมือที่ง่ายต่อการใช้งานสำหรับผู้ทั่วไป โดยผู้ใช้ไม่จำเป็นต้องเขียนโปรแกรมหรือรู้คำสั่ง SQL และเครื่องมืออื่นๆ ตัวจะมีการใช้งานเหมือนกันหมด (consistent user interface) ดังนั้นการเรียนรู้จึงทำเพียงครั้งเดียวก็จะสามารถใช้งาน เครื่องมือ ที่เหลืออื่นๆ ได้ เครื่องมือต่างๆ ประกอบด้วย

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ไว้เพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- 1) INGRES/Menu เป็น shell ที่ครอบคลุมเครื่องมือทั้งหมดของ INGRES เพื่อให้ผู้ใช้เรียกใช้งานได้สะดวกและรวดเร็ว
- 2) INGRES/Query-By-Forms (QBF) เป็นเครื่องมือช่วยในการทำ query ของผู้ใช้โดยที่ผู้ใช้ไม่จำเป็นต้องรู้ภาษา SQL เงื่อนไขสามารถถูกกำหนดได้อย่างอิสระและเก็บ ไว้ใช้ในอนาคตได้อีกสำหรับหน้าจอของ INGRES/QBF จะขึ้นมาให้เป็น default แต่ผู้ใช้สามารถตกแต่งใหม่ได้ตามต้องการ (โดยการใช้ INGRES / Vifred)
- 3) INGRES/Reprot-By-Forms (RBF) เป็น เครื่องมือ สำหรับสร้างรายงานที่ง่าย เพราะมีคำสั่งในลักษณะเป็นตัวเลือก (menus) สามารถทำและจัดรูปแบบรายงานโดยไม่ต้องเขียนโปรแกรมรายงานที่สร้างขึ้นสามารถเก็บและเรียกใช้ได้ในภายหลัง
- 4) INGRES/Vision-Forms-Editor (Vifred) คือเครื่องมือที่ใช้ในการตกแต่งหน้าจอซึ่งผู้ใช้สามารถสร้างหน้าจอได้ตามที่ต้องการ หน้าจอที่สร้างด้วย Vifred สามารถใช้งานได้กับ INGRES/QBF, INGRES /4GL และ 3GL เช่น C, COBAL, FORTRAN เป็นต้น
- 5) INGRES/Graph คือเครื่องมือในทางกราฟฟิค ซึ่งผู้ใช้สามารถใช้ในการสร้างกราฟเพื่อแสดงผลได้

4.5.2 Application Development Tools

INGRES มีเครื่องมือดังต่อไปนี้คือ

- 1) INGRES/Vision เป็น code generator สำหรับสร้าง 4 GL application สามารถทำงานได้ง่ายโดยการกำหนดหน้าจอ ความสัมพันธ์ระหว่างตาราง และระดับการทำงานระหว่างหน้าจอโดยใช้ visual programming และเราจะสามารถสร้างโครงสร้างของโปรแกรมประยุกต์ได้ง่าย เหมือนกับการสร้าง organization chart, INGRES/Vision จะช่วยสร้างโปรแกรมที่สมบูรณ์ที่ประกอบด้วยหน้าจอ (forms) , menus, queries, 4GL หรือแม้แต่การจัดการข้อผิดพลาดหากทำงานไม่สำเร็จโดยอัตโนมัติ การใช้ INGRES/Vision จึงทำให้

- ลดระยะเวลาที่ต้องการใช้ในการพัฒนา โปรแกรมประยุกต์ใหม่ลงได้อย่างมาก
- สามารถสร้างโปรแกรมประยุกต์ที่ flexible และ powerful ได้ในเวลาอันสั้น
- การบำรุงรักษาและการพัฒนาโปรแกรมประยุกต์ทำได้ง่าย

2) INGRES/Window4GL เป็น object oriented 4GL สำหรับการพัฒนาโปรแกรมประยุกต์ทางกราฟฟิค ซึ่งมี

- powerful object oriented 4GL และ debugger
- ความสามารถที่จะใช้ได้กับหลายแพลตฟอร์ม โดยไม่ต้องมีการแก้ไข code คือ มีสภาพแวดล้อมที่สนับสนุนหลาย window manager และหลายแพลตฟอร์ม
- facilities สำหรับการจัดการโปรแกรมประยุกต์ที่ซับซ้อนได้

INGRES/Window4GL จึงเหมาะสำหรับการสร้างโปรแกรมประยุกต์ของฐานข้อมูลแบบ client/server ที่ต้องการทั้ง window toolkit SQL database interface และ programming language เช่น C, C++ การใช้ INGRES/Window4GL นี้จะสามารถประหยัดเวลาในการพัฒนาโปรแกรมประยุกต์ได้ถึง 3-4 เท่า เมื่อเทียบกับ 3GL-based graphical tools

โปรแกรมประยุกต์ที่พัฒนาบน INGRES/Window4GL สามารถที่จะใช้ได้กับหลาย ๆ OS , hardware platforms และ window managers เช่น microsoft windows , presentation manager, OSF/Motif และ openlook window managers โดยไม่ต้องเปลี่ยนแปลงโปรแกรมแต่อย่างใด

INGRES/Window4GL เป็น event-based programming คือโปรแกรมประยุกต์ที่ถูกสร้างขึ้นจาก event block เมื่อ event ใด ๆ เกิดขึ้น block of code ของ event นั้นจะถูก execute ตัวอย่างของ event เช่นการ click mouse, การเปลี่ยนขนาดของ window, การรับ message จาก window อื่น เป็นต้น

นอกจากนั้น INGRES/Window4GL application ยังสามารถจะส่งข้อมูลระหว่าง window ได้ทำให้สามารถมี multiple active window ได้ เช่น ถ้า 2 window มีข้อมูลที่เกี่ยวข้องกันการเปลี่ยนแปลงข้อมูลใน window หนึ่ง จะทำให้อีก window หนึ่งถูกเปลี่ยนแปลงโดยอัตโนมัติ โดยการใช้ database event คือเมื่อ batch program ทำการแก้ไขฐานข้อมูลเสร็จเรียบร้อยแล้วจะสามารถบอกโปรแกรมประยุกต์ว่าการแก้ไขสมบูรณ์แล้วโดยการส่ง database event และหน้าจอจะถูก resfresh ด้วยข้อมูลใหม่ที่แก้ไขแล้วโดยอัตโนมัติ

4.6 DISTRIBUTED DATABASE MANAGEMENT

INGRES มี distributed modules ให้เลือกใช้งานในหลายรูปแบบ สำหรับความต้องการที่แตกต่างกัน distributed modules ของ INGRES จะ transparent ผู้พัฒนาและผู้ใช้ไม่รู้ถึงความแตกต่างในการใช้งาน distributed modules และระบบงานที่พัฒนาสำหรับฐานข้อมูลเดียวสามารถนำ distributed modules มาประยุกต์ใช้ให้เป็นฐานข้อมูลกระจายได้ทันที โดยไม่ต้องแก้ไข source code

INGRES/Replicater คือ ตัวกระจายข้อมูลจากเครื่องต้นทาง (source) ไปยังเครื่องอื่นๆ (target) สามารถทำได้หลายแบบไม่ว่าจะเป็นทั้งฐานข้อมูล, ตาราง, คอลัมน์ รวมทั้งการจัด config เช่น peer to peer, master-slave หรือ cascade และเวลาในการ replicate สามารถกำหนดเป็นแบบทันที ทันใด ตามตารางเวลา หรือทำเมื่อสั่งก็ได้

INGRES/Star เป็น distributed data manager จะช่วยจัดการให้เห็นฐานข้อมูลหลายฐานข้อมูลเสมือนเป็นฐานข้อมูลเดียวกัน โดยจะ transparent กับผู้ใช้และผู้พัฒนาทั้งบน network, โปรโตคอล และฐานข้อมูล พร้อมประกันความถูกต้องของข้อมูลด้วย โปรโตคอล two phase commit

INGRES/Gateway เป็นส่วนที่ช่วยให้สามารถติดต่อใช้งานร่วมกันได้ระหว่าง INGRES กับระบบจัดการฐานข้อมูลอื่นๆ เช่น DB2, R26, IMS, ORACLE ฯลฯ ซึ่งไม่จำกัดว่าระบบจัดการฐานข้อมูลอื่นๆ จะเป็น relational หรือไม่ ซึ่งการเชื่อมต่อสามารถทำได้ทั้งเรียกดู เพิ่มเติม ลบ และเปลี่ยนแปลงข้อมูล

การวิเคราะห์ระบบการจัดการฐานข้อมูล INFORMIX version 5

5.1 ARCHITECTURE

System architecture ของ INFORMIX เป็นแบบ shared memory คือ มีการใช้หน่วยความจำร่วมกันของ process ซึ่ง process ที่อยู่ในระบบของ INFORMIX สามารถแบ่งออกได้เป็น 2 ชนิดคือ

1) User process : เป็นกระบวนการทำงานของผู้ใช้ ซึ่ง user process ของผู้ใช้ทุก ๆ คน จะเข้าไปใช้ฐานข้อมูล โดยผ่านทางหน่วยความจำใน segment เดียวกันหรือกลุ่มของ segment เดียวกัน ด้วยวิธีการสร้าง visual address space เพื่อทำการอ้างถึงตำแหน่งจริงในหน่วยความจำ

2) Database server process : เป็นกระบวนการทำงานของระบบจัดการฐานข้อมูลที่จะทำหน้าที่บริหารการเข้าไปใช้ฐานข้อมูลที่เกิดขึ้นทั้งหมดจากผู้ใช้หรือโปรแกรมประยุกต์ โดยมีหน้าที่หลักอยู่ 2 ประการ คือ

- ประมวลผล query ของโปรแกรมประยุกต์ที่ถูกส่งมาให้กับ database server
- ส่งข้อมูลผลลัพธ์ที่ประมวลผลได้ กลับไปยังโปรแกรมประยุกต์

INFORMIX ใช้วิธีการที่เรียกว่า interprocess communication ในการจัดสรรหน่วยความจำสำหรับการทำงานที่มีหลาย user process เข้ามาใช้ฐานข้อมูล โดยในเวลาหนึ่ง ๆ นั้นจะมีเพียง user process เดียวที่อยู่ใน shared memory และ database server process ก็จะคอยบริการงานให้กับ user process ที่อยู่ใน shared memory เท่านั้น ทำให้

- ลดดีด I/O เนื่องจากมีเพียง process เดียวที่เข้าไปใช้บัฟเฟอร์
- ลดเวลาในการประมวลผล เนื่องจากมีเพียง process เดียวที่อยู่ใน shared memory ทำให้เหลือเนื้อที่สำหรับข้อมูลมากและสามารถค้นหาข้อมูลได้เร็วขึ้น

5.2 SEMANTIC DATA CONTROL

5.2.1 View Management

ในการแก้ไข view (insert, update, delete) ของ INFORMIX นั้นไม่สามารถทำการแก้ไขผ่าน view ได้เมื่อ

- ถ้า view ถูกสร้างมาจาก base table มากกว่า 1 ตารางนอกจากนี้ยังไม่สามารถ update, insert, delete ถ้า view ถูกสร้างขึ้นโดยมี with check option
- ถ้า view ถูกสร้างโดยมีเงื่อนไข group by clause distinct และ aggregate function ไม่สามารถ update, insert หรือ delete ได้
- ไม่สามารถใช้คำสั่ง alter table, create index และ update statistic ผ่าน view

- สำหรับคอลัมน์ที่ถูกระบุให้เป็น not null เมื่อ insert ผ่าน view ไม่สามารถเติมค่า null ใส่คอลัมน์นั้นได้
- ในการ drop view ไม่สามารถ drop view เป็นคอลัมน์ได้ (ต้อง drop ทั้ง view)

5.2.2 Data Integrity

การควบคุมความถูกต้องของข้อมูลที่เก็บไว้ในฐานข้อมูล โดยจะควบคุมความถูกต้องโดยใช้คำสั่ง “Tbcheck - option” เพื่อตรวจสอบความถูกต้องของข้อมูลดังนี้

- ตรวจสอบ reserved page ซึ่งจะเก็บ primary online control ไว้
- ตรวจสอบ system catalog ของฐานข้อมูล เช่น tables, column, indexes, view
- ตรวจสอบขอบเขตของฐานข้อมูลว่ามีการทับซ้อนกันหรือไม่
- ตรวจสอบ index ในตาราง ซึ่งในขณะที่ตรวจสอบจะ lock ตารางไว้ และจะไม่ยอมรับ transaction ใด ๆ ในขณะที่ตรวจสอบอยู่
- ตรวจสอบความถูกต้องของ page ของตารางที่อยู่ในฐานข้อมูล ซึ่งเมื่อตรวจสอบพบข้อผิดพลาดแล้วต้อง restore ข้อมูลจาก archive ยกเว้นข้อสุดท้ายที่ต้อง drop table ที่ทิ้งแล้ว create table ใหม่พร้อมทั้ง load ข้อมูลใหม่อีกครั้งหนึ่ง

5.3 TRANSACTION MANAGEMENT

5.3.1 Concurrency Control

INFORMIX จะยอมให้ผู้ใช้หลาย ๆ คนเข้าถึงข้อมูลในเวลาเดียวกัน และสามารถแน่ใจได้ว่าการเปลี่ยนแปลงฐานข้อมูลจะทำให้เกิดความถูกต้อง และอยู่ในลำดับซึ่งเป็นไปตามโครงสร้างของข้อมูล โดยมีกลไกในการจัดการอยู่ 3 ส่วนคือ

- Buffer locks
- Latches
- Timestamps

1) Buffer locks ช่วยในการแยกแยะ process เพื่อควบคุมความถูกต้องของการใช้ และการแก้ไขข้อมูลของการทำ user process ซึ่ง INFORMIX ได้แบ่งระดับของการ lock ออกเป็น 2 ระดับ คือ

- Tablelock : ควบคุมการใช้และเปลี่ยนแปลงข้อมูลในระดับตาราง
- Record lock : ควบคุมการใช้และเปลี่ยนแปลงข้อมูลในระดับ record หรือ row

และสามารถแบ่งออกตามสิทธิในการ lock ข้อมูลออกได้เป็น 3 mode คือ

- Shared lock : user process ทั้งหมดสามารถที่จะเข้าไปอ่านข้อมูลเดียวกันได้ โดยที่จะไม่มีการเปลี่ยนแปลงแก้ไขข้อมูลใด ๆ

- Update lock : จะมีเพียง user process เดียวที่สามารถทำการแก้ไขข้อมูลได้ ในขณะที่ user process สามารถที่จะอ่านข้อมูลแต่ไม่สามารถจะแก้ไขข้อมูลได้
- Exclusive lock : จะมีเพียง user process เดียวที่สามารถใช้ข้อมูลในขณะนั้นได้ ไม่ว่าจะเป็นการดู หรือเปลี่ยนแปลงแก้ไขข้อมูล โดยที่ user process อื่นที่ต้องการจะทำการ lock ข้อมูลต้องคอยจนกว่าจะทำการ unlock แบบ exclusive ออกไปก่อนจึงจะใช้ข้อมูลได้

2) Latches ใช้เพื่อเป็นตัวประสานงานการแก้ไขเปลี่ยนแปลงของ user process เกิดขึ้นใน shared memory โดยก่อนที่จะทำการแก้ไขเปลี่ยนแปลงทุก ๆ อย่างที่ใน shared memory นั้นจะต้องนำ user process เข้ามาใน latch ก่อนจึงจะสามารถทำการแก้ไขได้ และเมื่อได้ทำการแก้ไขเรียบร้อยแล้ว user process นั้นจะถูกนำออกจาก latch ซึ่งถ้ามี user process มากกว่าหนึ่ง user process ที่จะทำการเปลี่ยนแปลงแก้ไขแล้วนั้นจะมีเพียง user process เดียวที่จะเข้าไปใน latch ได้ ส่วนอีก user process ต้องคอยจนกว่า user process แรกจะทำการแก้ไขเสร็จ โดยใช้วิธีการที่เรียกว่า Test And Set (TAS) ซึ่งจะใช้การตรวจสอบ latch ว่าว่างหรือไม่เพื่อที่จะได้เข้าไปทำการแก้ไข และถ้าไม่ว่างจะทำให้เกิดเงื่อนไข 2 อย่างคือ

- คอยต่อไปและทำการจอง latch เพื่อคอยใช้เป็นอันดับต่อไป
- ไม่คอย โดยจะยกเลิกการทำ user process นั้นออกไป

3) Timestamps ใช้เพื่อจะชี้เวลา เมื่อมีเหตุการณ์หนึ่งเกิดขึ้นซึ่งเกี่ยวข้องกับอีกหลายเหตุการณ์ที่มีลักษณะเดียวกัน timestamp ไม่ใช่ลักษณะเวลาซึ่งเป็นแบบที่ระบุไว้ เช่น ชั่วโมง, นาที หรือวินาที และเมื่อมี timestamp 2 ตัวในระบบ ตัวที่มีค่าต่ำกว่าจะเป็นตัวที่เก่ากว่า

ในแต่ละ page จะมี timestamp หนึ่งตัวที่ page header และอีกหนึ่งตัวจะอยู่ที่ 4 ไบต์สุดท้ายของ page ส่วน timestamp ของ page header และ timestamp ของส่วนท้ายสุดจะถูก synchronize กันหลังจากการเขียนแต่ละครั้ง ซึ่ง timestamp นี้จะถูกใช้เมื่อ page ถูกอ่าน และการอ่านแต่ละครั้งจะเปรียบเทียบ timestamp เพื่อทดสอบหาความถูกต้องของข้อมูล โดยถ้าการทดสอบล้มเหลว ข้อผิดพลาดจะถูกส่งไปยังผู้ใช้ ซึ่งข้อมูลนั้นแสดงว่าข้อมูลใน page นั้นไม่ได้ถูกเขียนอย่างสมบูรณ์หรือ page อาจถูกเขียนทับเพียงบางส่วนบนดิสก์หรือบน share memory

5.3.2 Backup and Recovery ของ INFORMIX

INFORMIX ใช้การทำ mirror ทั้งฐานข้อมูลและ log ไว้ เพื่อเป็นการป้องกันในกรณีที่เกิดความผิดพลาดขึ้นกับสื่อที่ใช้เก็บข้อมูล

Mirroring

การทำ mirror ฐานข้อมูลก็คือการทำสำรองข้อมูลเก็บไว้ ดังนั้นทุกครั้งที่มีการเขียนลงบนข้อมูลตัวหลัก จะเกิดการเขียนที่ข้อมูลของ mirror โดยอัตโนมัติ และถ้าหากมีการผิดพลาดหรือการล้มเหลวของข้อมูลหลักแล้ว ตัว mirror จะยอมให้มีการอ่านหรือเขียนจากข้อมูลตัว mirror ได้จนกว่าตัวข้อมูลหลักจะถูกซ่อมแซมให้นำกลับมาใช้ใหม่ได้ โดยไม่มีการไปรบกวนการทำงานของผู้ใช้

ผู้บริหารฐานข้อมูลสามารถที่จะจัดการทำ mirror โดยผ่าน DB-Monitor ซึ่งมีหน้าที่ในการจัดการเนื้อที่ของดิสก์ หรือใช้ในการแก้ไขส่วนควบคุมไฟล์โดยใช้ Unix Editor และจัดการเรื่อง share memory ใหม่ โดยสามารถที่จะเริ่มหรือจบการทำ mirror ได้ทุกเมื่อ การทำ mirror จะต้องมีการใช้ blob space หรือ db space ซึ่งไม่สามารถทำ mirror กับข้อมูลที่เลือกไว้ภายใน blob space หรือ db space

Checkpoint

check point ของ INFORMIX มีความยืดหยุ่นมาก ซึ่งสามารถบังคับ check point ให้เกิดขึ้นตามเวลา (เช่น ทุก 15 นาที) หรือโดยคิดเป็นเปอร์เซ็นต์ของขนาดสูงสุดของ log หรืออาจจะอาศัย โปรแกรม DB-monitor ก็ได้

ในขณะที่ทำการ check point ไม่สามารถบังคับให้การทำ transaction ขนาดใหญ่หยุดการเขียนลงบน log ได้ ดังนั้นการมีความสามารถในการบังคับให้มี check point จึงเป็นทางเลือกของความปลอดภัย ซึ่งโดยปกติแล้วเมื่อ log เต็ม การทำงานทั้งหมดจะหยุดจนกว่าผู้บริหารฐานข้อมูลจะแก้ไขปัญหาเสร็จ

Logical Log

Logical log file เป็นสิ่งที่ใช้ในกลไกการทำ recovery ซึ่งภายใน log file ของ INFORMIX นั้นประกอบด้วยข้อมูล 3 ส่วน คือ

- SQL data definition statements ที่ใช้สำหรับทุก ๆ ฐานข้อมูล
- การเปลี่ยนแปลง configuration ภายในระบบเช่น การเปลี่ยนแปลง db spaces, blob spaces
- ผลลัพธ์และ transaction ที่ได้ทำไปก่อนที่จะ checkpoint จะเกิดขึ้น

Logical log file มีหน้าที่หลัก 3 อย่าง ที่ช่วยในการทำ recovery ข้อมูล และการตรวจสอบความสอดคล้องของข้อมูล

- เมื่อต้องมีการทำ rollback transaction ขึ้นจะใช้ข้อมูลที่เก็บอยู่ใน logical log file มาทำการเปลี่ยนความเปลี่ยนแปลงของ transaction ที่ได้ทำไปแล้วให้กลับมาเป็นเหมือนก่อนการทำ transaction
- เมื่อต้องการทำ rollforward transaction โดยจะใช้ข้อมูลใน logical log file เพื่อ commit การเปลี่ยนแปลงที่ได้ทำไปแล้วทั้งหมด
- เมื่อระบบเกิด shutdown ขึ้นในขณะที่ใช้งาน โดยจะใช้ข้อมูลใน logical log file เพื่อทำการซ่อมแซมข้อมูลที่สูญหายไปให้กลับคืนมา

Logical Log Backup

INFORMIX ใช้การ backup log file ลงใน magnetic tape เพื่อเก็บไว้ในการทำ recovery ข้อมูลเมื่อเกิดความเสียหายกับข้อมูลขึ้น โดยวัตถุประสงค์ของการ backup ก็เพื่อ

- ทำการเก็บ logical log record ไว้สำหรับการทำ rollforward
- ทำการลบ logical log file เดิมทิ้งไป เพื่อรอ logical log file ใหม่ เนื่องจาก logical log file เดิมได้ถูก backup เก็บไว้แล้ว

Fast Recovery

เป็นวิธีการที่จะแก้ไขข้อมูลให้กลับสู่ปกติก่อนที่ระบบหรือข้อมูลจะเกิดความเสียหายขึ้นโดยเร็ว ซึ่งมีวิธีการ 4 ขั้นตอนดังนี้

- 1) นำเอา pages ทั้งหมดที่ได้ใช้ไปในเวลาใกล้จุด checkpoint จาก physical log files เข้ามา
- 2) ค้นหา record ที่อยู่ใกล้จุด checkpoint ที่สุดใน logical log files
- 3) ทำการ Roll forward ทุก ๆ logical log record หลังจากจุด checkpoint
- 4) Roll back transaction ทั้งหมดที่ยังเหลืออยู่

ผลที่ได้ก็คือทุก ๆ transaction ที่ได้ทำการ commit ไปแล้วก็จะถูกนำกลับคืนมา และทุก ๆ transaction ที่ยังไม่ได้ commit ก็จะทำการ roll back กลับไปก่อนทำ transaction นั้น

5.4 QUERY PROCESSING ของ INFORMIX

5.4.1 Storage Structure

Hash

Hash ช่วยให้การค้นหาข้อมูลในตารางเป็นไปอย่างรวดเร็ว ตารางการแบ่งหน่วยความจำ 3 ตารางของ INFORMIX ที่มีการเชื่อมกับตาราง hash มี ตารางการ lock, ตาราง active tblspace และ ตาราง buffer

แต่ละรายการซึ่งจะต้องถูกเก็บไว้ในตารางนั้นจะมี unique key ซึ่งถ้ามีการใช้ตาราง hash แล้ว unique key จะถูก hashed ซึ่งหมายถึง algorithm ที่ถูกระบุไว้ที่จะใช้เพื่อ map กับตัวคีย์ที่เป็นตัวเลขจำนวนจริง ซึ่งเป็นค่า hash และ algorithm จะถูกใช้เมื่อทำการ hash แล้วคีย์จะถูกแบ่งออกเป็นส่วน ๆ ในลักษณะหนึ่ง ซึ่งผลลัพธ์ไม่ได้เป็นการ map ที่มีลักษณะเป็น unique ซึ่ง คีย์ที่ต่างกัน 2 คีย์นั้นสามารถนำไป map แล้วได้ค่าเดียวกัน ซึ่งรายการจะถูกเก็บโดยค่า hash ที่ไม่ใช่ค่าคีย์เดียว ๆ

ตัวอย่างเช่น ใช้ algorithm ง่าย ๆ คือ “หารค่าคีย์โดย 100 และใช้ค่าคงเหลือหรือค่าเศษ” และด้วย algorithm นี้ คุณสามารถหาค่า hash ได้ 100 ค่าที่ต่างกัน ซึ่งก็เท่ากับค่าเศษคือตั้งแต่ 0 - 99 ค่าแต่ละตัวของ hash จะตรงกับรายการในตาราง hash

และเพื่อที่จะหารายการหนึ่งในตาราง ตัวคีย์ของรายการจะถูกส่งผ่าน hashing algorithm และค่าของการ hash จะถูกคำนวณ โดยใช้ค่านี้ รายการซึ่งอยู่ใน hash table ก็จะถูกทำการค้นหา

แต่ละรายการใน hash table จะประกอบไปด้วยตัวชี้ซึ่งชี้ไปยังรายการที่เกี่ยวข้องในตารางที่เกี่ยวข้อง (ตารางการ lock, ตาราง active tblspace หรือตารางผู้ใช้) โดยใช้ค่า hash ซึ่งค่ารายการหลายรายการซึ่งมีค่า hash เดียวกัน จะถูกเชื่อมต่อกันในรายการการเชื่อมต่อแบบลูกโซ่

INFORMIX จะเปรียบเทียบคีย์รายการกับค่าคีย์ที่ใช้ค้นหา ถ้าทั้งคู่ตรงกันจะแสดงว่ารายการนั้นถูกค้นพบ แต่ถ้าไม่แต่ละรายการในรายการการเชื่อมต่อแบบลูกโซ่และจะถูกค้นหาจนค้นพบรายการนั้นหรือการจบการค้นหาเสร็จสิ้นลง

5.4.2 Data Independence

Physical Data Independence

การเปลี่ยนแปลงการจัดเก็บฐานข้อมูลของ INFORMIX (physical) จะไม่มีผลกระทบต่อการทำ โปรแกรมประยุกต์ ที่ทำงานอยู่และ ผู้ใช้ ไม่จำเป็นต้องทราบโครงสร้างการเก็บข้อมูลเปลี่ยน

Logical data Independence

การเปลี่ยนแปลงตารางโดยการเพิ่ม คอลัมน์, ตัด คอลัมน์ เพิ่มตารางไม่มีผลกระทบต่อ การเข้าถึงตารางของผู้ใช้

5.4.3 Optimized ของ INFORMIX

INFORMIX ใช้การ optimize แบบตามราคา (cost base) แต่ระบบจะใช้สถิติต่างๆเพียงเล็กน้อยในการประเมิน สำหรับการทำให้ optimize โดยจะเก็บจำนวนแถว, index ที่มีอยู่, ค่าสูงสุดและค่าต่ำสุด ไม่มีการสร้างฮิสโตแกรมของข้อมูล (ไม่เหมือน INGRES ที่มีการสร้างฮิสโตแกรมของข้อมูล) แต่จะถือว่าข้อมูลการกระจายอย่างสม่ำเสมอระหว่างค่าสูงสุดและต่ำสุด แล้วต่อ optimizer จะนำเส้นทางการเข้าถึงทั้งหมดมาตัดราคา และเลือกเส้นทางที่ถูกที่สุดมาใช้

INFORMIX ยังมีส่วนเฝ้าดูความสามารถในการทำงาน (DB Monitor) ซึ่งใช้งานง่าย เนื่องจากใช้ระบบเมนู เราสามารถใช้ DB Monitor นี้ ในการปรับแต่งค่าต่างๆ เพื่อให้เกิดความสามารถในการทำงานทันทีที่เหมาะสมที่สุดได้ นอกจากนี้ DB Monitor ยังเก็บค่าทางสถิติไว้มากมายเพื่อให้อำนวยความสะดวกในการปรับแต่งระบบ เช่น อัตราความถูกต้องของแคช (cache hit) , การอ่านและเขียนดิสก์, ค่า checkpoint , lock, และสถานะของผู้ใช้ แต่ค่าต่างๆ เหล่านี้ ไม่มีส่วนในการนำไปช่วย optimize ระบบเลย ซึ่งแตกต่างจาก ORACLE 7 ที่นำค่าเหล่านี้ไปมีส่วนในการ optimize

5.5 ADDITIONAL TOOLS

INFORMIX มีเครื่องมือไว้ใช้สำหรับพัฒนา โปรแกรมประยุกต์ มากมายดังนี้

INFORMIX 4GL

เป็นภาษา 4GL มี 2 แบบคือ INFORMIX 4GL-RDS Rapid Development System และ INFORMIX 4GL Interactive ซึ่งช่วยลดระยะเวลาในการพัฒนา โปรแกรมประยุกต์ สามารถสร้างโปรแกรมประยุกต์ที่ยืดหยุ่น และมีประสิทธิภาพในเวลาอันสั้น การบำรุงรักษาและพัฒนาโปรแกรมประยุกต์ทำได้ง่าย

INFORMIX 4GL Forms

เป็น code generator ช่วยในการกระทำเกี่ยวกับข้อมูลได้ง่าย โดยการกำหนดหน้าจอความสัมพันธ์ระหว่างตาราง เราสามารถสร้างโครงสร้าง โปรแกรมประยุกต์ ได้ง่าย ๆ

INFORMIX - Menus

เป็นเครื่องช่วยสร้าง Menus และ Submenus ทำให้สามารถ run program , ทำ report หรือเรียกใช้คำสั่งเกี่ยวกับ OS ได้ง่าย

INFORMIX - ESQL for C และ Cobol

ใช้สำหรับการ run bed ภาษา SQL กับ ภาษา C หรือ Cobol

INFORMIX - Ada/SAME

เป็น SQL module Language compiler ไว้สำหรับรองรับภาษา Ada

INFORMIX New Era View Point

เป็นโปรแกรมกราฟฟิคสำหรับระบบ ฐานข้อมูลที่ใช้ภาษา SQL ในการเข้าถึงข้อมูล ซึ่งถูกออกแบบมาเพื่อให้ end user สามารถเข้าถึงข้อมูลได้ง่ายกว่าภาษา C หรือ Cobol

5.6 DISTRIBUTED DATABASE MANAGEMENT

INFORMIX สนับสนุนการทำ distributed database โดยมีเครื่องมือที่ใช้ดังนี้

INFORMIX-Star เป็น distributed data manager จะช่วยจัดการให้เห็นฐานข้อมูลหลายฐานข้อมูลเสมือนเป็นหนึ่งเดียวกัน โดยจะ transparent กับผู้ใช้และผู้พัฒนาทั้งบน Network, protocol และฐานข้อมูลพร้อมประกันความถูกต้องของข้อมูลด้วย protocol two phase commit

INFORMIX-Net เป็นส่วนเชื่อมต่อระหว่าง client/server เพื่อสนับสนุนการทำงานของ INFORMIX-Star และ INFORMIX-Gateway

INFORMIX-Gateway with DRDA เป็นส่วนช่วยให้สามารถติดต่อใช้งานร่วมกันระหว่าง INFORMIX กับระบบจัดการฐานข้อมูลแบบสัมพันธ์ของ IBM (DB2, DB2/400, DB2/VM)

INFORMIX-TP/XA เป็นส่วนช่วยในการเชื่อมต่อ INFORMIX database server เข้ากับเครื่อง เวิร์คสเตชัน (X/open)

การวิเคราะห์ระบบจัดการฐานข้อมูล EMPRESS version 6

การพิจารณาคูณลักษณะของระบบจัดการฐานข้อมูลของ EMPRESS นั้น คุณสมบัติของภาษาที่ใช้ในการจัดการฐานข้อมูล อันได้แก่ คำสั่งที่ใช้สำหรับนิยามข้อมูล , คำสั่งสำหรับการใช้ข้อมูล และ คำสั่งสำหรับการควบคุมความถูกต้องของข้อมูลนั้น เป็นไปตามมาตรฐานของ ANSI ซึ่งสามารถที่จะใช้ภาษาในการจัดการฐานข้อมูลได้ 2 ทางคือ ผ่านทาง EMPRESS SQL และ EMPRESS 4GL และ ใน EMPRESS นั้น นอกจากจะสามารถจะใช้คำสั่งในการจัดการฐานข้อมูลแบบ command line แล้วนั้น ยังสามารถใช้แบบ interactive interface ได้อีกวิธีหนึ่ง ซึ่งแบบ interactive interface นี้จะมีลักษณะเป็น form ขึ้นมาเพื่อให้เติมหรือเปลี่ยนแปลงแก้ไขข้อมูล เช่น ชื่อ attribute , ชนิดของ attribute หรือ ความยาวของ attribute เป็นต้น โดยจะใช้ได้กับคำสั่งที่สร้างหรือเปลี่ยนแปลงข้อมูลคือ create , alter , insert , update

6.1 ARCHITECTURE

สถาปัตยกรรมฐานข้อมูลของ EMPRESS นั้นเป็นแบบ fully distributed, client-server และ multi-server โดยมีการเชื่อมต่อกับ node อื่นผ่านทางโครงข่าย ที่เรียกว่า heterogeneous distributed database

ในแต่ละ node สามารถที่จะกำหนดให้เป็น server, client หรือให้เป็นทั้ง 2 อย่างได้ โดยที่แต่ละ server สามารถที่จะทำงานแบบ multiple CPU คือสามารถที่จะรองรับการทำงานของหลาย client ในเวลาเดียวกัน รวมถึงความสามารถของ server ที่จะทำงานแบบ multi-process server ด้วย โดยที่แต่ละ client สามารถที่จะเข้าถึงหลาย ๆ server ในเวลาเดียวกัน

EMPRESS ใช้การ shared memory เพื่อเพิ่มความเร็วในการ lock, การทำ on-line transaction processing และการเรียกใช้ข้อมูลที่มีการเรียกซ้ำ โดย shared memory จะอนุญาตให้ทำการจองและยกเลิกการจอง resources ต่าง ๆ ที่มีการใช้ร่วมกันของหลายผู้ใช้ด้วยความรวดเร็ว ซึ่งจะอนุญาตให้ผู้ใช้สามารถปรับแต่งปรับแต่งโปรแกรมประยุกต์โดยการ map ส่วนของ index, attribute, ไฟล์ และตาราง เข้ากับ shared memory โดยไม่ต้องแก้ไขที่โปรแกรมประยุกต์

6.2 SEMANTIC DATA CONTROL

6.2.1 View Management

ในการ update ผ่าน view นั้นจะต้องสร้างมาจากตารางเพียงหนึ่งตารางเท่านั้น โดยถ้ามากกว่า 1 ตารางไม่สามารถ update ได้) และในกรณีที่มีคอลัมน์ที่ถูกกำหนดเป็น not null หรือ ถูก create unique index จะไม่สามารถใส่ค่า null ลงไป ในเงื่อนไข where clause จะยอมให้ update เมื่อ

- เป็นการเปรียบเทียบ Attribute กับค่าคงที่
- เป็นการเปรียบเทียบระหว่าง 2 Attribute ใน table เดียวกัน
- ในกรณีที่ค่า attribute ที่อยู่ต่างตารางกันนั้นเหมือนกัน ยกเว้น operation not, or และ view ที่ถูกสร้างมาจาก base table มากกว่า 1 ตาราง ไม่สามารถถูก update ได้ ยกเว้น where clause operation “ = ” เท่านั้น ที่สามารถ update ได้

6.2.2 Data Integrity

คำสั่งสำหรับการควบคุมความถูกต้องของข้อมูลนั้น EMPRESS มีการควบคุมความถูกต้องของลักษณะข้อมูล ด้วยการใช้คำสั่ง create range check เพื่อกำหนดขอบเขตความเป็นไปได้ของข้อมูล และการควบคุมความถูกต้องของการเปลี่ยนแปลงสถานะของข้อมูล และป้องกันปัญหาความไม่ถูกต้องตรงกันของข้อมูล โดยการ referential integrity constraints โดยได้รวมเอาไว้เป็นส่วนหนึ่งของพจนานุกรมข้อมูล

นอกจากนั้น EMPRESS ยังใช้ script และ event ในการควบคุมความถูกต้องโดยผ่านทาง EMPRESS 4GL

Script

เป็นส่วนประกอบหนึ่งของ EMPRESS 4GL ซึ่ง script นี้เป็น procedure module ที่คอยบอก EMPRESS 4GL ว่าจะต้องทำอะไรบ้างในช่วงเวลาต่าง ๆ โดย script นี้จะทำงานสัมพันธ์ไปกับ event และ subscript ซึ่งเป็น procedure ที่สามารถจะเรียกใช้ได้จาก script ใด ๆ การใช้ script นี้จะใช้เหมือนกับการเขียน procedural language ทั่วไป แต่ใช้ได้ง่ายกว่า

Event

เป็นส่วนที่แตกต่างไปจาก procedural language ใน EMPRESS 4GL ซึ่งเรียกว่า “event driven”

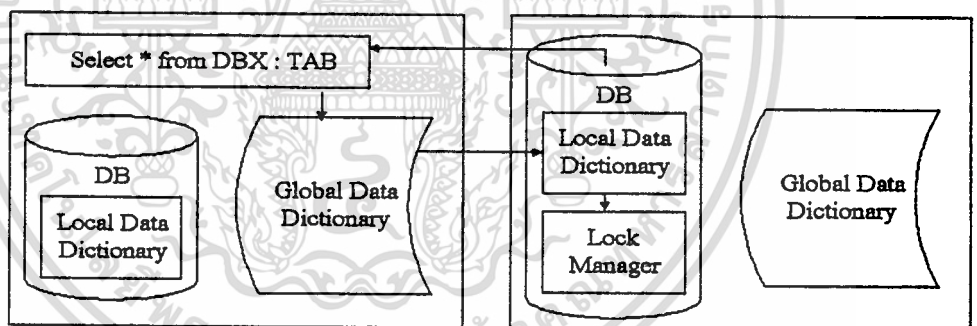
โดยปกติแล้ว procedural language นั้น การทำงานจะเป็นตามลำดับการทำงานที่เขียนในโปรแกรม แต่ event driven นั้น ลำดับการทำงานไม่ได้ขึ้นอยู่กับโปรแกรม แต่จะสัมพันธ์กับ event โดยที่ไม่ต้องเขียนโปรแกรมเพื่อค้นหา event ที่เกิดขึ้น เพียงแต่กำหนดว่าจะให้ตอบสนองต่อ event ใดที่จะเกิดขึ้นเท่านั้น

การนำ script และ event ไปใช้นั้น ทำได้โดยการกำหนดการตอบสนองต่อ event ของ event driven ซึ่งเมื่อมี event ที่ตรงกับที่ได้กำหนดไว้ใน event driven แล้วก็ให้กระโดดไปทำตาม script ซึ่งจะถูกนำไปใช้ในการควบคุมความถูกต้องคือ เมื่อมี event ของเหตุการณ์ที่ไม่ถูกต้องเกิดขึ้น ก็อาจให้ทำการ rollback transaction กลับไปใหม่ได้ เป็นต้น

6.3 TRANSACTION MANAGEMENT

6.3.1 Concurrency Control

สำหรับความสามารถด้านการทำรายการนั้น EMPRESS มีความสามารถในการทำ โดยอนุญาตให้มีการทำ lock ในระดับของ table , group , record และ null คือ ไม่อนุญาตให้มีการ lock ข้อมูลใด ๆ โดยไม่สนใจกับความผิดพลาดหรือ ความเสียหายที่เกิดขึ้นกับข้อมูล หรือระบบได้ ในการ lock แบบเป็น group นั้น จะทำกับเฉพาะฟิลด์ ที่ถูกกำหนดให้มี index เท่านั้น ซึ่งถ้าในตารางนั้นไม่มี index อยู่เลย การ lock แบบ group นั้นก็จะให้ผลแบบเดียวกับการ lock แบบตาราง EMPRESS จะใช้ lock manager (รูปที่ 5.1) ในการควบคุมการ lock ข้อมูลต่าง ๆ ที่เกิดขึ้นในฐานข้อมูล ดังนั้น เมื่อทำการเข้าถึงข้อมูลใด ๆ ในระบบเครือข่าย หรือแม้จะเป็น local database ก็ยังคงรักษาความถูกต้องไว้ได้



รูปที่ 5.1 ภาพแสดงการทำงานของ lock manager

ในสภาวะแวดล้อมที่เป็นแบบ multi-user EMPRESS มีกลไกในการ lock อยู่ 2 วิธีคือ

- 1) Automatic locking เป็นการ lock ที่จะทำกับทุกตารางในขณะที่ทำการใช้ตารางนั้น โดย query language statement
- 2) Explicit locking เป็นการ lock ที่ผ่านทาง EMPRESS programming interface โดยอาจทำการ lock ฐานข้อมูล, ตาราง หรือ record สำหรับการอ่านและเขียน โดย โปรแกรมภายนอก และยังคง lock จนกว่าจะทำการ unlock จากโปรแกรมภายนอก

6.3.2 Backup & Recocery

Backup

EMPRESS จะทำการคัดลอกฐานข้อมูลไปเก็บไว้ใน magnetic tape และลบ recovery log file ที่ซึ่งจะทำการ back up ข้อมูลทั้งหมดในฐานข้อมูล รวมถึง พจนานุกรมข้อมูล และ remote table ข้อควรจำสำหรับการทำการ backup ก็คือ ไม่ควรจะให้มีไฟล์ ที่ไม่ใช่ database file ใน database directory หรือใน directory ที่ใช้สำหรับ remote tables เพราะไฟล์เหล่านั้นจะไม่ถูก backup ไปด้วย และเมื่อทำการ restore กลับมาไฟล์ นั้นก็จะหายไป

Recovery

เมื่อเกิดความเสียหายขึ้นกับฐานข้อมูล เช่น operating system หรือดิสค์ ได้รับความเสียหาย EMPRESS จะทำการซ่อมแซมฐานข้อมูล จาก backup copy ครั้งล่าสุดและ recovery log file ให้ฐานข้อมูลมีสภาพกลับมาเป็นดังเดิมเหมือนก่อนที่จะได้รับความเสียหาย

Audit Trail

ใน EMPRESS มีการทำ audit trail file ซึ่งเป็นไฟล์ที่ทำการเก็บคำสั่งที่ผู้ใช้ทุกคนได้กระทำกับฐานข้อมูลได้ โดยจะเก็บข้อมูลจำเพาะของผู้ใช้ คำสั่งที่ทำ และ เวลาที่ทำ ตารางที่ทำ และ record ที่ทำ ฯลฯ

โดยสรุปแล้ว audit trail นั้นจะเป็นการควบคุมดูแล การทำงานของผู้ใช้ ให้สามารถทำงานได้สำเร็จตามวัตถุประสงค์ โดยจะทำการกับข้อมูลในระดับตาราง

Recovering to a Save Point

วิธีนี้จะมีการกำหนด save point เพื่อที่จะทำ recover บางส่วนของ transaction ที่ได้ทำไปแล้ว จนถึงจุดที่ได้กำหนด save point เมื่อ transaction นั้น ถูกขัดจังหวะ ซึ่งใน EMPRESS นั้นมีการกำหนด protection transaction คือถ้า transaction นั้นถูก protection ไว้ก็จะไม่มีการ recovery transaction นั้น แต่ถ้าไม่ได้ protection ไว้ EMPRESS ก็จะทำการ recovery transaction นั้นจนถึงจุดที่ได้กำหนด save point ไว้

นอกจากนั้น EMPRESS ยังใช้การ commit ข้อมูลแบบ two - phase commit คือ สถานะของข้อมูลหลังจากถูกขัดจังหวะ จะกลับไปเป็นสถานะเดิมก่อนที่จะเริ่มประมวลผลรายการนั้น เมื่อเกิดความผิดพลาดอันเนื่องจากการถูกขัดจังหวะในขณะที่กำลัง ประมวลผลรายการจะกลับไปเป็นสถานะเดิมก่อนที่จะเริ่มประมวลผลรายการนั้น เมื่อเกิดความผิดพลาดอันเนื่องจากการถูกขัดจังหวะในขณะที่กำลังประมวลผลรายการ

6.4 QUERY PROCESSING

EMPRESS จะใช้หลักการของ cost base ในการเข้าถึงข้อมูลที่สามารถเข้าถึงได้เร็วที่สุด โดยอาศัยการเก็บสถิติ ของการใช้ I/O ในแต่ละเส้นทางที่ใช้ในการเข้าถึงข้อมูล มาเป็นตัวตัดสินใจในทางเลือกเส้นทางที่จะเข้าถึงข้อมูล

6.5 ADDITIONAL TOOLS

EMPRESS report writer

เป็นเครื่องมือสำหรับสร้างรายงานตามรูปแบบที่ผู้ใช้สามารถกำหนดได้ โดยสามารถนำข้อมูลมาจากฐานข้อมูลของ EMPRESS หลายๆตัว หรือตัวเดียวก็ได้ รวมถึงข้อมูลที่ได้จาก EMPRESS 4GL สามารถใช้คำสั่งประมวลผลทางคณิตศาสตร์ เช่น SUM, MAX, MIN, AVERAGE

ผู้ใช้สามารถที่จัดรูปแบบของรายงานได้ เช่น จัดตำแหน่งของตัวหนังสือ, รูปภาพ หรือ ย้ายตำแหน่งของคอลัมน์, จัดกึ่งกลาง, กำหนดหัวกระดาษ เป็นต้น

ลักษณะของการใช้งานนั้นอยู่ในรูปของการเขียนโปรแกรม โดยใช้คำสั่งง่ายๆเพื่อกำหนดจุดต่างๆ บนหน้าจอ

EMPRESS 4GL เป็นตัวพัฒนาโปรแกรมประยุกต์ ที่มีภาษาและเครื่องมือที่ช่วยให้ผู้ใช้สามารถที่จะสร้างโปรแกรมประยุกต์ได้อย่างรวดเร็ว และเป็นลักษณะของ event driven programming ซึ่งสามารถสร้าง form ซึ่งประกอบไปด้วยตัวอักษร, fields, lines และ video, windows ซึ่ง form เป็นพื้นที่ในการแสดงผล, fields เป็นพื้นที่บน form ที่ใช้สำหรับแสดงผลข้อมูล, คีย์ สำหรับให้ผู้ใช้ไว้เลือก, scripts ซึ่งเป็นการรวมเหตุการณ์อย่างง่าย ๆ ไปจนถึงโปรแกรมประยุกต์ที่มีความซับซ้อน

Language

Scripts จะถูกเขียนด้วยภาษา script โดยผ่าน application editor ซึ่งภาษา script ประกอบไปด้วยส่วน variable declarations, control statements, database commands และ system commands

เครื่องมือในการพัฒนา

ส่วนในตัวของเครื่องมือในการพัฒนาประกอบไปด้วย default application denerator, application generator, application manager, form manager, source debugger, executable builder, compiler, linker เป็นต้น

Default Application Generator ใช้สร้างโปรแกรมประยุกต์ แบบ interactive window ซึ่งสามารถให้ผู้เลือกใช้ทำงานโดยใช้ฟังก์ชันคีย์ ซึ่งในส่วนนี้ไม่เกี่ยวกับการเขียน programming ซึ่งถ้ามีชื่อของโปรแกรมประยุกต์ และตารางแล้ว เครื่องจะสร้างโปรแกรมประยุกต์ ที่ขึ้นอยู่กับ table definition ในพจนานุกรมข้อมูลโดยอัตโนมัติ ซึ่งใช้สำหรับสร้างตัว prototype ซึ่งคุณสามารถใช้ application editor มาแก้ไขโปรแกรมประยุกต์ได้

Application Generator ใช้สร้างโปรแกรมประยุกต์ที่เป็น interactive menu-driven บนหนึ่งตารางหรือมากกว่านั้น ซึ่งทำให้ผู้ใช้สามารถใช้งานข้อมูลในฐานข้อมูลได้ ซึ่งในส่วนนี้ไม่เกี่ยวกับการเขียน programming และสามารถสร้างโปรแกรมประยุกต์ โดยมีเมนูให้เลือกดังนี้ รูปแบบหน้าจอที่ต้องการ, ตารางและ attribute, function keys, records และข้อมูลช่วยเหลือซึ่งใช้สำหรับสร้างตัว prototype

Application Manager ใช้สร้างโปรแกรมประยุกต์ใหม่ และแก้ไขโปรแกรมประยุกต์ที่มีอยู่แล้วได้ ซึ่งจะใช้บ่อยมากในการพัฒนา โดย application manager จะยอมให้ออกจาก form manager ได้โดยไม่ต้องออกจาก application manager

Application Editor ใช้ในการแก้ไข script ซึ่งเป็นตัวสร้างโปรแกรมประยุกต์ โดยที่สามารถจะเขียน application script, window definitions, field scripts, key scripts, subscripts และ domain definitions ซึ่งตัว script จะถูกเก็บไว้ในตารางของระบบ และจะมีเมนูให้เลือกตารางที่ต้องการจะแก้ไข

Form Editor เป็น full screen editor ซึ่งให้คุณสร้าง, ออกแบบ และแก้ไข forms โดยมีลักษณะเป็นหน้ากระดาษที่ให้ผู้ใช้งานสามารถลงสี ซึ่งการแก้ไขจะประกอบไปด้วย การพิมพ์อักษรที่ background การลงสี วาดเส้น และการสร้าง field

Form Manager ใช้ในการสร้าง form ผ่านทาง form editor

Source Debugger เป็นเครื่องมือที่ใช้วิเคราะห์หาที่เกิด runtime errors โดยมีลักษณะเป็น menu-driven และใช้ window ในการแสดงผล ซึ่งสามารถที่จะหยุดการทำงานของโปรแกรมได้ตลอดเวลาเพื่อใช้ในการหาจุดบกพร่องได้

Executable Builder ใช้สร้างตัว executable โดยมีการเชื่อมโยง read-only runtime executable

Compiler and Linker ใช้สำหรับ compile สร้าง library ในโปรแกรมประยุกต์ linking ใช้สร้างตัว executable application ที่ต้องเชื่อม library เข้าด้วยกัน โดยจะทำการหา syntax error, valid window size และ undefined variables

6.6 DISTRIBUTED DATABASE MANAGEMENT

EMPRESS เป็น fully distributed database management system ที่มีความสามารถที่จะทำงานทั้งในแบบ homogeneous และ heterogeneous โดยที่วัตถุประสงค์ของ heterogeneous distributed database นั้นจะช่วยให้ผู้ใช้เลือกใช้ hardware ที่เหมาะสมสำหรับการทำงาน, สถานที่ของผู้ใช้ โดยยังคงทำงานเหมือนกับใช้โครงสร้างของ hardware เดียวกัน

ประโยชน์ที่จะได้รับจาก heterogeneous distributed database นั้นก็คือผู้ใช้สามารถที่จะเข้าถึงข้อมูลและใช้ข้อมูลจากที่ใดก็ได้ในระบบเครือข่าย พร้อมกับสามารถเก็บข้อมูลลงไปในที่ใดก็ได้ในระบบเครือข่าย

EMPRESS Database Server

Database server จะมีเครื่องมือหลายตัวในการทำงานเกี่ยวกับข้อมูลภายในตัวเอง ข้อมูลจะถูกแจกจ่ายและแยกเก็บไว้ต่างที่กัน ซึ่งสามารถทำงานที่ใดก็ได้ในระบบเครือข่าย ซึ่ง “การทำงาน” หมายถึงความสามารถในการใช้ฟังก์ชันมาตรฐานของ SQL

Database server สามารถที่จะทำงานบนระบบเครือข่ายของ heterogeneous nodes หรือแบบอื่น เช่น ฐานข้อมูลถูกเก็บใน Sun SPARC และทำงานจากลูกข่ายของ Sun อีกตัวหนึ่ง เป็นต้น ทั้งนี้ข้อมูลในฐานข้อมูลจากแพลตฟอร์มที่ต่างกันจะแสดงออกไม่เหมือนกัน EMPRESS สามารถบอกได้ในส่วนนี้โดยใช้ Data Format Code (DFC) ซึ่งใช้บอกถึงการสร้างฐานข้อมูล

ถ้า DFC ของฐานข้อมูลกับ DFC ของเครื่องลูกข่ายไม่เหมือนกัน database server จะเปลี่ยนให้เป็นรูปแบบที่ทั้งสองเข้าใจได้

Database Administrator สามารถที่จะสร้างไฟล์ Global Data Dictionary (GDD) บนเครื่อง client เพื่อรักษาความสัมพันธ์ระหว่างชื่อของฐานข้อมูลกับที่อยู่ของข้อมูล, ชื่อของ server ของฐานข้อมูล และ Server Configuration File (SCF) ที่จำเป็นสำหรับ client และ server node ในเครือข่าย ซึ่งประกอบด้วยชื่อของ host ของเครื่องที่เป็น server, เบอร์ของ subprocess server, ชื่อของเจ้าของ server, และลักษณะของการติดต่อของ server ซึ่ง EMPRESS จะมี utilities สำหรับสร้างไฟล์เหล่านี้ซึ่งง่ายและรวดเร็ว

เมื่อ GDD และ SCF ถูกสร้างอย่างถูกต้อง ผู้ใช้ทุกคนสามารถใช้ฐานข้อมูลข้ามเครือข่ายโดยไม่ต้องคำนึงถึงสถานที่หรือ node หรือสถาปัตยกรรมของเครื่อง ซึ่งฟังก์ชัน ของ EMPRESS ทั้งหมดให้สนับสนุนการข้ามเครือข่ายได้

ข้อมูลจำเพาะของ EMPRESS

ความต้องการพื้นฐานทางด้าน software และ hardware

EMPRESS สามารถทำงานได้บนคอมพิวเตอร์ซึ่งใช้ระบบปฏิบัติการ UNIX (SYSTEM V , Berkeley , etc) หรือ VMS อย่างใดอย่างหนึ่ง และต้องมีหน่วยความจำหลักอย่างน้อย 1 เมกะไบต์ นอกจากนั้น EMPRESS ยังสามารถทำงานภายใต้ระบบปฏิบัติการ MS-DOS โดยใช้หน่วยความจำขยาย (extended memory) ได้อีกด้วย สำหรับโปรแกรมระบบจัดการฐานข้อมูล EMPRESS นี้ต้องการพื้นที่ในหน่วยความจำสำรองประมาณ 9 เมกะไบต์

จำนวนฐานข้อมูล

สามารถสร้างฐานข้อมูลได้มากที่สุดเท่ากับจำนวน directory บนระบบปฏิบัติการ โดยปกติแล้วในระบบไฟล์ขนาด 300 เมกะไบต์สามารถสร้างฐานข้อมูลได้มากถึง 10,000 ฐานข้อมูล

ขนาดฐานข้อมูล

สามารถสร้างได้ใหญ่เท่ากับขนาดหน่วยความจำสำรองที่มีเหลืออยู่

จำนวน table ต่อหนึ่งฐานข้อมูล

สามารถสร้างได้มากที่สุดถึง 9,999 tables

จำนวน record ต่อหนึ่ง table

สามารถมีได้มากที่สุดประมาณ 2,147,483,647 records

ขนาด record

สามารถได้มากที่สุดประมาณ 2,147,483,647 records

ความยาวของข้อมูลแบบ char

สามารถได้ยาวที่สุดประมาณ 2,147,483,647 records

จำนวนฐานข้อมูลที่เปิดใช้ในเวลาเดียวกัน

ขึ้นอยู่กับเครื่องคอมพิวเตอร์ที่ใช้แต่โดยทั่วไปแล้วจะมีค่าอยู่ระหว่าง

50-100 ฐานข้อมูล

จำนวนผู้ใช้ที่ table ในเวลาเดียว

ไม่จำกัดจำนวน (ในกรณีที่ไม่มี lock table)

จำนวน คีย์ ที่ใช้ในการ sort

สามารถเลือก sort ตามคีย์ ได้มากที่สุดถึง 9,999 เท่าของจำนวน attribute ที่มีได้มากที่สุดในหนึ่ง table

จำนวน indice ต่อหนึ่ง table

สามารถมีได้มากเท่ากับจำนวน attribute ในหนึ่ง table

จำนวน level ของ nesting ในคำสั่ง select

สามารถมีได้มากที่สุดถึง 100 level

บทที่ 7

การทดสอบระบบจัดการฐานข้อมูลเชิงสัมพันธ์

7.1 การทดสอบประสิทธิภาพของระบบจัดการฐานข้อมูล

การทดสอบจะแบ่งการวัดประสิทธิภาพออกเป็น 2 ส่วน ดังนี้

- 1) การ Lookup จะเป็นการ scan ข้อมูลในตารางในรูปแบบต่าง ๆ ทั้งแบบมีเงื่อนไขและไม่มีเงื่อนไข
- 2) การ update จะเป็นการทดสอบการแก้ไขข้อมูล และการลบข้อมูลออก โดยจะมุ่งประเด็นไปที่การ lock ไม่ได้มุ่งที่เวลา

ซึ่งฐานข้อมูลนี้ได้มาจาก มหาวิทยาลัยรามคำแหง จำนวน 2 ตาราง ตารางละ 25,000 records และได้ตัดเอาเฉพาะบางคอลัมน์มาใช้ดังนี้

ตารางที่ 1

FIELD NAME	POSITION	LENGTH	FORMAT	COMMENT
ID-NO	1 - 9	9	character	รหัสประจำตัวนักศึกษา
PRENAME	10 - 11	2	character	คำนำหน้านาม
SPECIAL CODE	12	1	character	code พิเศษสำหรับยศทหาร
THAI NAME	13 - 52	40	character	ชื่อ-สกุล ภาษาไทย
ENGLISH NAME	53 - 86	34	character	ชื่อ-สกุล ภาษาอังกฤษ
FACULTY CODE	87 - 88	2	character	คณะที่สมัคร
MAJOR CODE	89 - 90	2	character	สาขาที่สมัคร
FACULTY COUNTER	91	1	character	จำนวนคณะที่เรียนตั้งแต่เข้ามาศึกษา
REGISTER STATUS	92	1	character	สภาพการลงทะเบียนและการเป็น นศ.
OLD REGISTER STATUS	93	1	character	ค่า register status ในภาคก่อน
CERTIFICATE STATUS	94	1	character	สถานภาพของวุฒิบัตร
BIRTH OF DATE	95 - 102	8	integer	วัน/เดือน/ปี เกิดของนักศึกษา

ตารางที่ 2

FIELD NAME	POSITION	LENGTH	FORMAT	COMMENT
STATUS CODE	1	1	character	สถานภาพนักศึกษา
BATCH	2 - 4	3	character	
SERIAL	5 - 9	5	character	
SUMPACK	10 - 11	2	character	จำนวนวิชาที่ลงทะเบียน
NEAR-GRADUATE	12	1	character	การขอจบการศึกษา
ID-NO	13 - 21	9	character	รหัสประจำตัวนักศึกษา
NO-REPETTIVE-HIS	22	1	integer	ตรวจสอบการลงทะเบียนซ้ำหรือไม่
FILE-CODE-FLAG	23	1	integer	ทำการลงทะเบียนจากเครื่องเบอร์ใด
YY-MM-DD	24 - 29	6	integer	วัน/เดือน/ปี เกิดของนักศึกษา
PERIOD	30	1	integer	คาบ เข้า/บ่าย
COUNTER	31 - 33	3	integer	
BILL-NO	34 - 39	6	integer	เลขที่ของใบเสร็จ
TIME	40 - 47	8	integer	เวลาที่ลงทะเบียน
SUBJECT1	48 - 52	5	character	รหัสวิชาที่ลงอันดับที่ 1
CREDIT1	53	1	integer	จำนวนหน่วยกิตที่ลงอันดับที่ 1
.....				
SUBJECT8	90 - 94	5	character	รหัสวิชาที่ลงอันดับที่ 8
CREDIT8	95	1	integer	จำนวนหน่วยกิตที่ลงอันดับที่ 8

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ขั้นตอนการทดสอบ

- 1) สร้างฐานข้อมูลให้ชื่อฐานข้อมูลว่า “TESTDB”
- 2) สร้างตารางให้ชื่อตารางที่ 1 ชื่อ “STUDENT” และตารางที่ 2 ชื่อ “SUBJECT”
- 3) นำข้อมูลเข้าสู่ตารางนำข้อมูลเข้าโดยใช้ความสามารถของระบบจัดการฐานข้อมูล
- 4) สร้างตัวชี้ข้อมูลทำกับ 4 คอลัมน์ในตารางที่ 1 คือ ID_NO , T_NAME , E_NAME, FAC_CODE และในตารางที่ 2 คือ ID_NO
- 5) เขียน query เพื่อทดสอบดูเวลาและประสิทธิภาพการทดสอบจะเป็นในลักษณะของ 1 ผู้ใช้ และ 2 ผู้ใช้

Query และวัตถุประสงค์ในการทดสอบ

1) Join และ Subquery

วัตถุประสงค์ เพื่อทดสอบ response time ของ query ที่ให้ผลลัพธ์เดียวกัน โดยใช้ query แบบ join และ subquery

q -11

```
select t_name
from student, subject
where student.id_no = subject.id_no
and id_no = 35 %
```

Query นี้จะเป็นการแสดงค่าใน คอลัมน์ t_name จากตาราง student โดยมีเงื่อนไขคือ ให้ค่าของคอลัมน์ id_no ของตาราง student เท่ากับค่าของคอลัมน์ id_no ของตาราง subject โดยเฉพาะ row ที่มีค่าในคอลัมน์ id_no มีค่าขึ้นต้นด้วย ‘35’ เช่น 35..... เป็นต้น ซึ่งเป็นการใช้วิธีการ join ระหว่าง 2 ตาราง

q -12

```
select t_name
from student
where id_no = (select id_no from subject
where id_no = ‘35 %’)
```

เป็นการแสดงค่าจาก คอลัมน์ t_name จากตาราง student โดยมีเงื่อนไขคือ ให้ค่าของคอลัมน์ id_no ของตาราง student มีค่าเท่ากับค่าคอลัมน์ id_no ของตาราง subject โดยเฉพาะ row ที่ค่าในคอลัมน์ id_no ของตาราง subject มีค่าขึ้นต้นด้วย ‘35’ เช่น 35..... เป็นการใช่วิธี subquery

2) Built-in Function

วัตถุประสงค์ เพื่อทดสอบประสิทธิภาพในการประมวลผลทางคณิตศาสตร์ ซึ่งจะเป็นการรวมเอาขอบเขตการเปรียบเทียบค่า integer และการ search แบบ sequential ด้วย

q - 21 select max(time)
 from subject

Query นี้ จะหาเวลาที่มากที่สุดที่นักศึกษามาลงทะเบียน

วัตถุประสงค์ เพื่อทดสอบประสิทธิภาพในการประมวลผลทางคณิตศาสตร์ อีกแบบว่า จะเป็นการรวมเอาขอบเขตการคำนวณค่า integer และการ scan แบบ sequential ด้วย

q - 22 select avg(time)
 from subject

Query นี้ จะหาค่าเฉลี่ยทั้งคอลัมน์โดยใช้คอลัมน์ time เป็นคอลัมน์ทดสอบ

3) Syntax Independent

วัตถุประสงค์ เพื่อทดสอบคุณสมบัติในการทำ query ของ ระบบจัดการฐานข้อมูล ว่าลำดับของเงื่อนไข where - clause มีผลต่อการทำ query หรือไม่ เพื่อทดสอบว่า ระบบจัดการฐานข้อมูลมี query optimization ทำหน้าที่ในการเลือกเส้นทางที่ดีที่สุดในการ access ข้อมูล โดยไม่สนใจลำดับของการทำ query หรือไม่

q - 31 select *
 from student
 where id_no = '35 %'
 and fac_code = '03'

q - 32 select *
 from student
 where fac_code = '03'
 and id_no = '35 %'

ทั้ง 2 query จะหานักศึกษาที่ส 35 ที่อยู่คณะ 03 โดย q - 31 จะหาที่ส นักศึกษา ก่อน แล้วค่อยหาค่า fac_code และ q - 32 จะหา fac_code ก่อน แล้วค่อยหา id_no

4) Create View และการดูข้อมูลจาก View

วัตถุประสงค์ เพื่อทดสอบคุณสมบัติในการทำ query ว่า เวลาที่ได้จากการ scan จาก base table กับเวลาที่ scan จาก view จะต่างกันอย่างไร ซึ่งในการทดสอบจะเป็นการ scan แบบไม่มีเงื่อนไข เพราะไม่ต้องการให้เกิดผลกระทบจากเงื่อนไขใน query

q - 41 create view data_1 as
 select *
 from student

q - 42 select *
 from student

q - 43 select *
 from data_1

โดย q - 42 จะเป็นการ scan จาก base table

โดย q - 43 จะเป็นการ scan จาก view

5) Subquery 3 ชั้น

วัตถุประสงค์ เพื่อทดสอบประสิทธิภาพของระบบจัดการฐานข้อมูลในการทำ lookup table กับ records จำนวนมาก ๆ การทำ subquery จะเป็นการเพิ่มจำนวน records ที่เกี่ยวข้องให้มากขึ้น และระบบจัดการฐานข้อมูลจะต้องหาข้อมูลในตารางมากกว่า 1 รอบ เพื่อหาผลของ query ในแต่ละชั้น

q - 7 select pre_name, id_no, t_name, e_name
 from student
 where req_status = 'c'
 and t_name not in (select t_name
 from student
 where fac_count = 'a'
 and id_no in (select id_no
 from student
 where cert_status = '2'))
 and t_name in (select t_name
 from student
 where fac_code = '01'))

Query ที่เป็นการหาว่านักศึกษาที่ย้ายคณะมากกว่า 1 ครั้ง และเป็นนักศึกษาเทียบโอนมาจากนักศึกษา 8 ปี และในขณะนี้เรียนคณะนิติศาสตร์ ว่ามีใครบ้าง

6) Update

วัตถุประสงค์ เพื่อทดสอบคุณสมบัติ concurrency control เมื่อมีการแก้ไขข้อมูล

q - 51

update student

set fac_code = '99'

where id_no = '35%'

Query นี้จะทำการเปลี่ยนค่าข้อมูลในคอลัมน์ fac_code ให้มีค่าเท่ากับ '99' เฉพาะ record ที่ค่าในคอลัมน์ id_no มีค่าขึ้นต้นด้วย 35

q - 52

update student

set fac_code = '88'

where id_no = '36'

Query นี้จะทำการเปลี่ยนค่าข้อมูลในคอลัมน์ fac_code ให้มีค่าเท่ากับ '88' เฉพาะ record ที่ค่าใน คอลัมน์ id_no มีค่าขึ้นต้นด้วย 36

7) Delete

วัตถุประสงค์ เป็นการทดสอบที่มุ่งประเด็นการทดสอบไปที่การ lock ของ ระบบจัดการฐานข้อมูล ว่ามีการ lock ระดับใด

q - 61

delete from student

where fac_code = '03'

q - 62

delete from student

where fac_code = '01'

โดยจะทำการทดสอบดังนี้

- ทดสอบที่ละ active user โดยใช้ query ข้างต้น โดยทดสอบทีละ query แล้วดูเวลาที่ได้ว่าเป็นอย่างไร โดยแต่ละครั้งที่ทำการทดสอบ จะทำกับ ตาราง ที่สมบูรณ์เท่านั้น จะนำตารางที่เคยถูก delete แล้ว มาทำซ้ำไม่ได้ เพราะจะได้เวลาคลาดเคลื่อนไปมาก
- ทดสอบ 2 active user โดยใช้ตารางเดียวกัน โดย user 1 จะใช้ query ที่ 1 และ user 2 จะใช้ query 2 แล้วจะพิจารณาจากเวลาที่ ได้ว่ามีการ lock ระดับใด

7.2 ผลการทดสอบประสิทธิภาพของ INGRES version 6.4

Run บนเครื่อง VAX 6000/510

O/S	VAX VMS 6/2
RAM	64 MB
Harddisk	7.5 GB
Terminal	Dumb 120 terminals

ลำดับขั้นตอนการทดสอบ

1. สร้างฐานข้อมูล โดยใช้คำสั่ง CREATEDB TESTDB
2. สร้างตาราง เข้าสู่โปรแกรมภาษา SQL โดยใช้คำสั่ง ISQL TESTDB

สร้างตารางที่ 1

CREATE TABLE STUDENT

(ID_NO	CHAR(9) NOT NULL,
PRENAME	CHAR(2),
SPEC_CODE	CHAR(1),
T_NAME	CHAR(40) NOT NULL,
E_NAME	CHAR(34),
FAC_CODE	CHAR(2),
MAJ_CODE	CHAR(2),
FAC_COUNT	CHAR(1),
REG_STATUS	CHAR(1),
OLD_REG_STATUS	CHAR(1),
CERT_STATUS	CHAR(1),
BIRTH_DATE	I4)

สร้างตารางที่ 2

CREATE TABLE SUBJECT

(STATUS_CODE	CHAR(1),
SUMPACK	CHAR(1),
BATCH	CHAR(3),
SERIAL	CHAR(5),
SUMPACK	I2,
NEAR_GRAD	CHAR(1),
ID_NO	CHAR(9) NOT NULL,
NO_REP_HIS	CHAR(1),
CODE_FLAG	CHAR(1),
YY_MM_DD	I4,
PERIOD	CHAR(1),
COUNTER	I2,
BILL_NO	I4,
TIME	I4,
SUBJECT1	CHAR(5),
CREDIT1	CHAR(1),
SUBJECT2	CHAR(5),
CREDIT2	CHAR(1),
SUBJECT3	CHAR(5),
CREDIT3	CHAR(1),
SUBJECT4	CHAR(5),
CREDIT4	CHAR(1),
SUBJECT5	CHAR(5),
CREDIT5	CHAR(1),
SUBJECT6	CHAR(5),
CREDIT6	CHAR(1),
SUBJECT7	CHAR(5),
CREDIT7	CHAR(1),
SUBJECT8	CHAR(5),
CREDIT8	CHAR(1))

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3. นำข้อมูลเข้าตาราง

นำข้อมูลเข้าตารางที่ 1

COPY TABLE STUDENT

```
(ID_NO           =CHAR(9),
PRENAME         =CHAR(2),
SPEC_CODE       =CHAR(1),
T_NAME          =CHAR(40),
E_NAME          =CHAR(34),
FAC_CODE        =CHAR(2),
MAJ_CODE        =CHAR(2),
FAC_COUNT       =CHAR(1),
REG_STATUS      =CHAR(1),
OLD_REG_STATUS  =CHAR(1),
CERT_STATUS     =CHAR(1),
BIRTH_DATE      =CHAR(8)NL)
FROM 'RU2555.DAT'
```

นำข้อมูลเข้าตารางที่ 2

COPY TABLE STUDENT

(STATUS_CODE	=CHAR(1),
BATCH	=CHAR(3),
SERIAL	=CHAR(5),
SUMPACK	=CHAR(2),
NEAR_GRAD	=CHAR(1),
ID_NO	=CHAR(9),
NO_PER_HIS	=CHAR(1),
CODE_FLAG	=CHAR(1),
YY_MM_DD	=CHAR(6),
PERIOD	=CHAR(1),
COUNTER	=CHAR(3),
BILL_NO	=CHAR(6),
TIME	=CHAR(8),
SUBJECT1	=CHAR(5),
CREDIT1	=CHAR(1),
SUBJECT2	=CHAR(5),
CREDIT2	=CHAR(1),
SUBJECT3	=CHAR(5),
CREDIT3	=CHAR(1),
SUBJECT4	=CHAR(5),
CREDIT4	=CHAR(1),
SUBJECT5	=CHAR(5),
CREDIT5	=CHAR(1),
SUBJECT6	=CHAR(5),
CREDIT6	=CHAR(1),
SUBJECT7	=CHAR(5),
CREDIT7	=CHAR(1),
SUBJECT8	=CHAR(5),
CREDIT8	=CHAR(1)NL)

FROM 'RU2444.DAT'

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับใช้เพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4. สร้างตัวชี้ข้อมูล

สร้างตัวชี้ข้อมูลในตารางที่ 1

```
CREATE INDEX ID0 ON TABLE STUDENT(ID_NO)
```

```
CREATE INDEX TNAME ON TABLE STUDENT(T_NAME)
```

```
CREATE INDEX ENAME ON TABLE STUDENT(E_NAME)
```

```
CREATE INDEX FACCODE ON TABLE STUDENT(FAC_CODE)
```

สร้างตัวชี้ข้อมูลในตารางที่ 2

```
CREATE INDEX ID1 ON TABLE SUBJECT(ID_NO)
```

5. ทดสอบ query

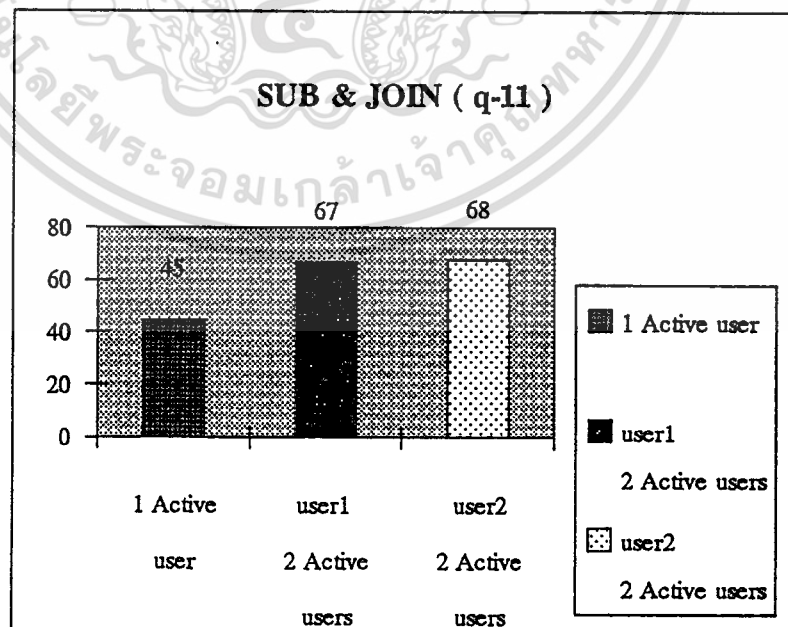
คำนวณเวลาที่ใช้ในการทำ query ด้วยการนำค่าที่อยู่ใน dbmsinfo มาลบกัน
โดยใช้คำสั่ง :

```
SELECT DBMSINFO ('_BINTIM') ไว้ที่ก่อนและหลังการทำ query
```

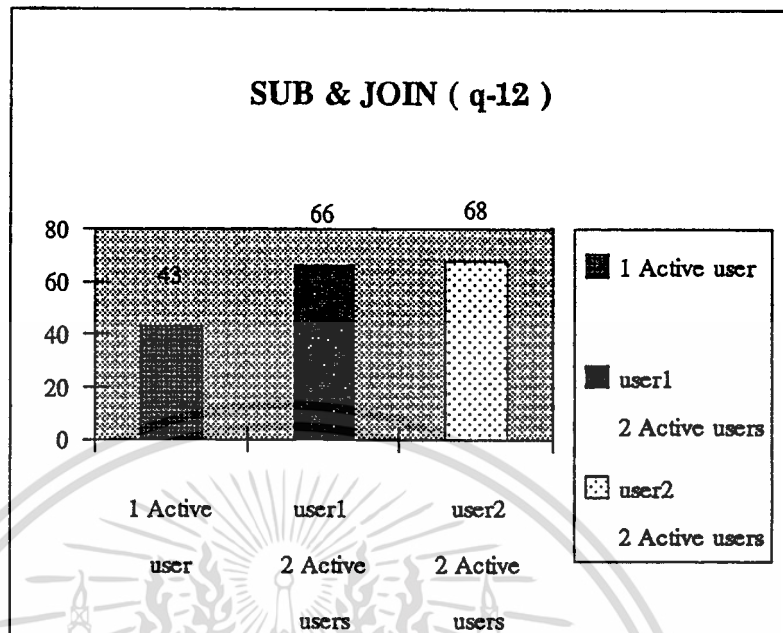
1) Sub & Join

ผลการทดสอบ

	1 Active User	2 Active User	
q - 11	45	67	68
q - 12	43	66	68



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



วิเคราะห์ผลการทดสอบ

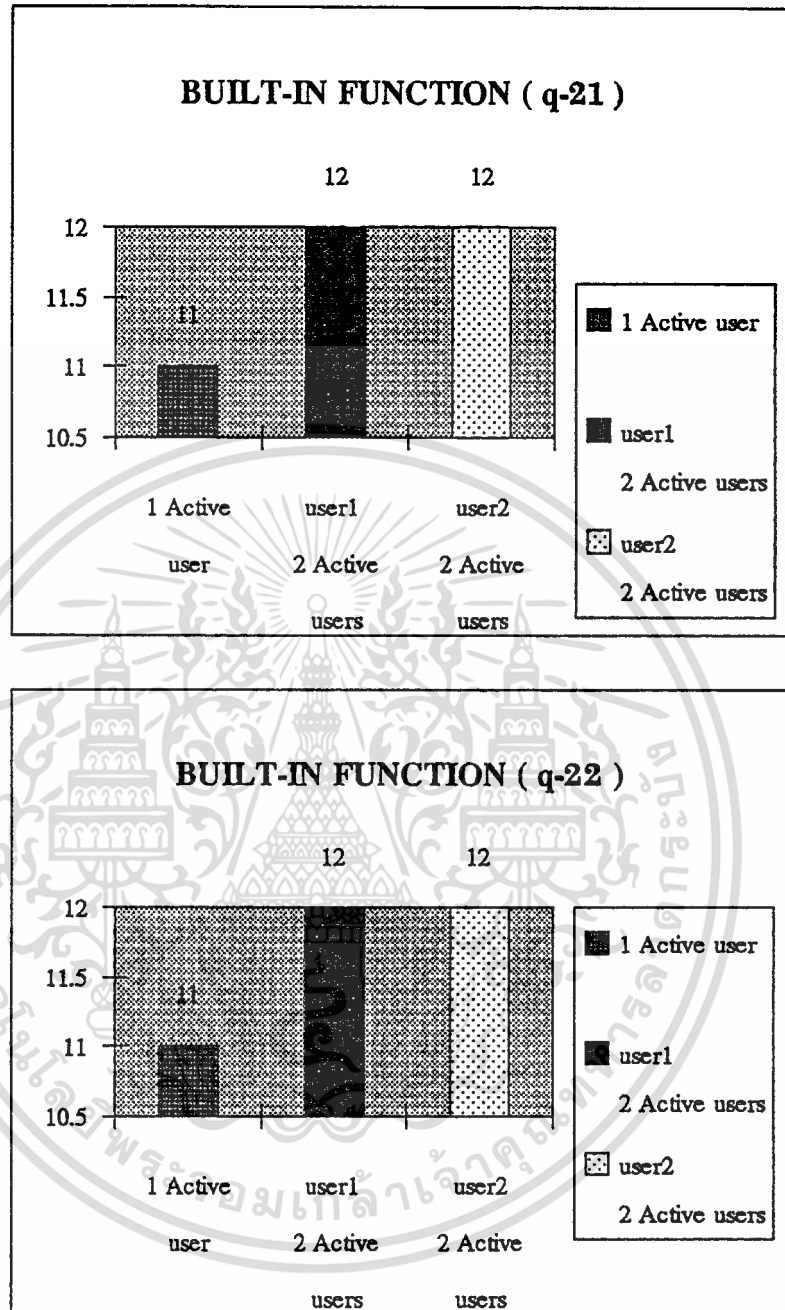
จากผลการทดสอบจะเห็นว่า เวลาที่ใช้ในการประมวลผลจะใกล้เคียงกันมาก โดยเมื่อใช้ query แบบ join จะใช้เวลาประมวลผลมากกว่าแบบ sub และผลต่างจะเพิ่มมากขึ้นตามจำนวนผู้ใช้

และจากหลักการออกแบบ ระบบจัดการฐานข้อมูล ของ Codd นั้น ในกรณีที่ได้ผลลัพธ์เหมือนกัน แต่ใช้วิธีการ query และวิธีการเลือกเส้นทางที่ต่างกัน ควรจะได้ response time เดียวกัน ซึ่งจากผลการทดสอบ INGRES ไม่มีคุณสมบัติข้อนี้

2) Built-in Function

ผลการทดสอบ

	1 Active User	2 Active User	
q - 21	11	12	12
q - 22	11	12	12



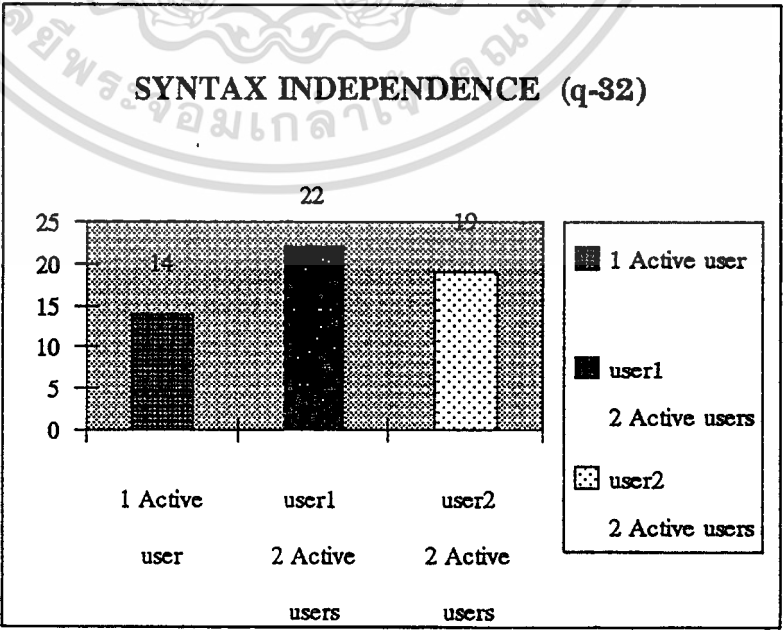
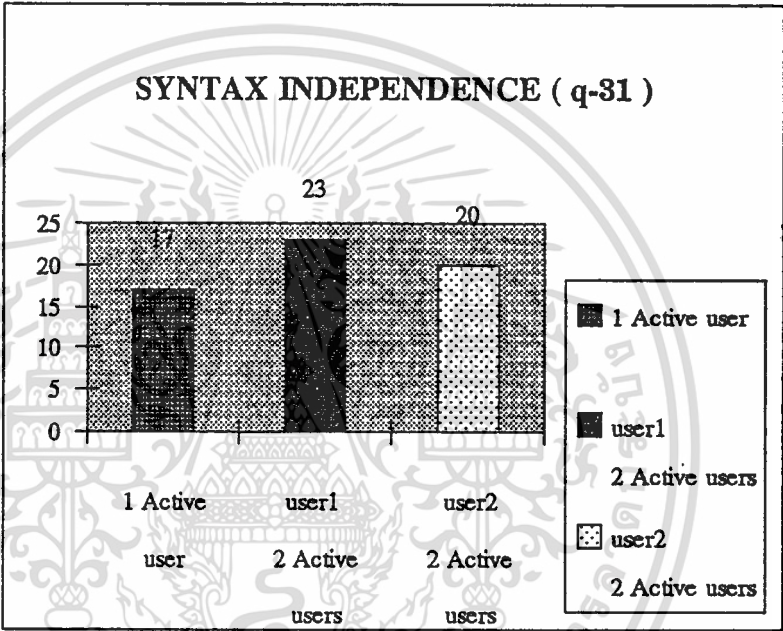
วิเคราะห์ผลการทดสอบ

การทดสอบประสิทธิภาพในเรื่องการประมวลผล function ทางคณิตศาสตร์ทั้งแบบเปรียบเทียบและหาค่าเฉลี่ยนี้ ใช้เวลาค่อนข้างจะใกล้เคียงกัน และใช้เวลาไม่น้อยเมื่อเทียบกับการ scan ใน query อื่น ๆ ในการทดสอบ ซึ่งทำให้คาดว่า function ที่นำมาทดสอบทั้งสองแบบอาจมีการใช้ค่า statistics ที่เก็บใน catalog มาช่วยในการประมวลผล

3) Syntax Independence

ผลการทดสอบ

	1 Active User	2 Active User	
q - 31	17	23	20
q - 32	14	22	19



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

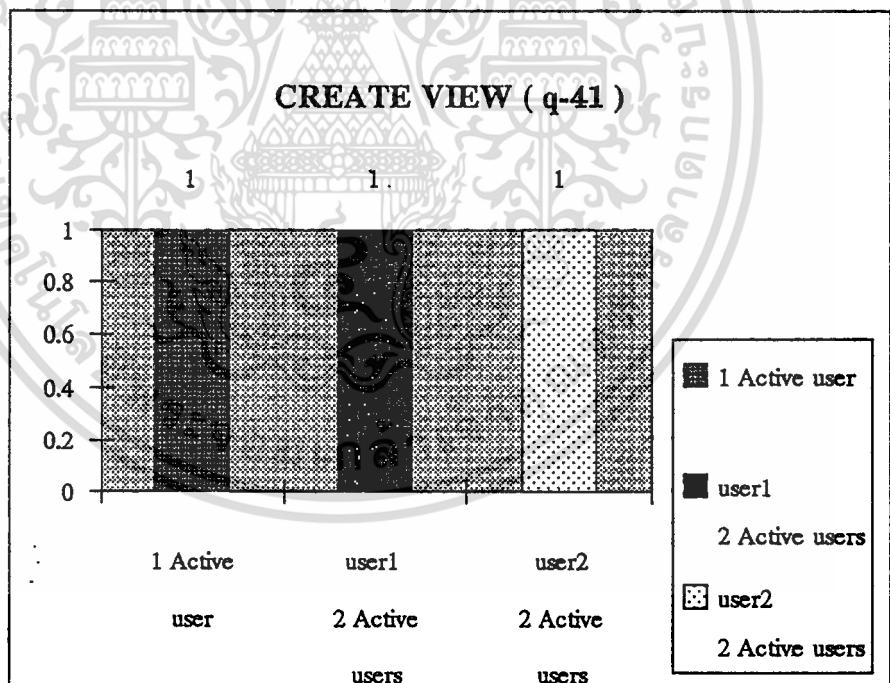
วิเคราะห์ผลการทดสอบ

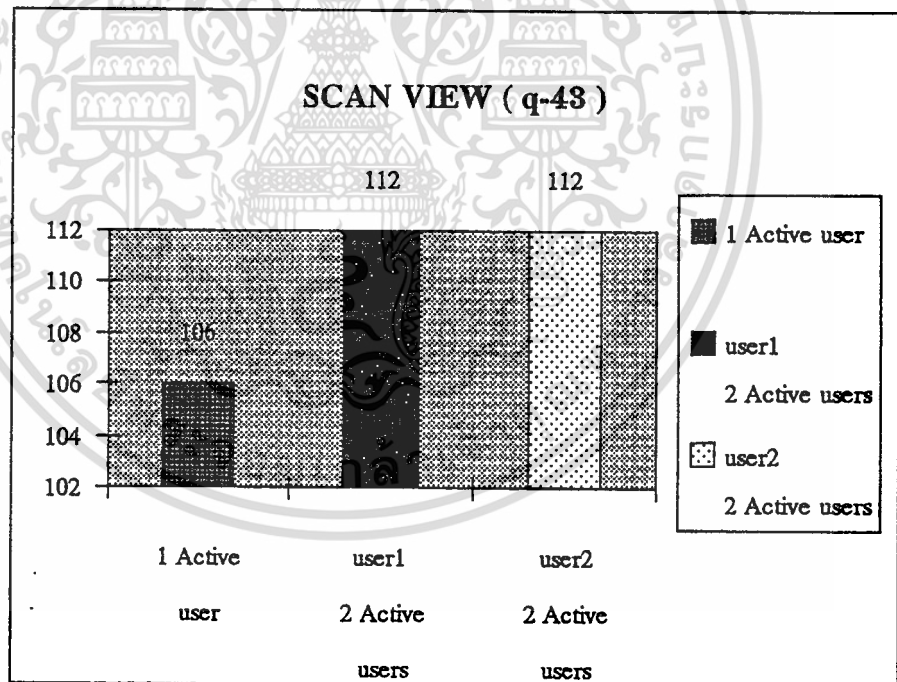
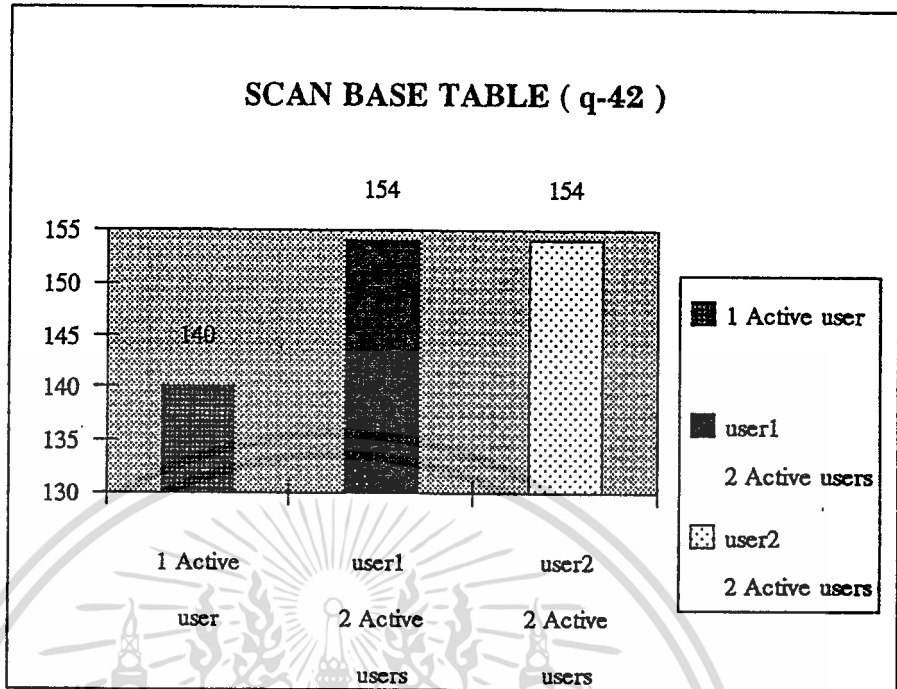
จากผลต่างของเวลา ทำให้สามารถวิเคราะห์ได้ว่า INGRES ขาดคุณสมบัติในด้าน Syntax-Independence หมายถึงลำดับของเงื่อนไขใน query มีผลต่อการทำ query แสดงว่า query optimizer ทำงานด้านนี้ไม่สมบูรณ์

4) Create View และ Scan ข้อมูลจาก View

ผลการทดสอบ

	1 Active User	2 Active User	
q - 41	1	1	1
q - 42	140	154	154
q - 43	106	112	112





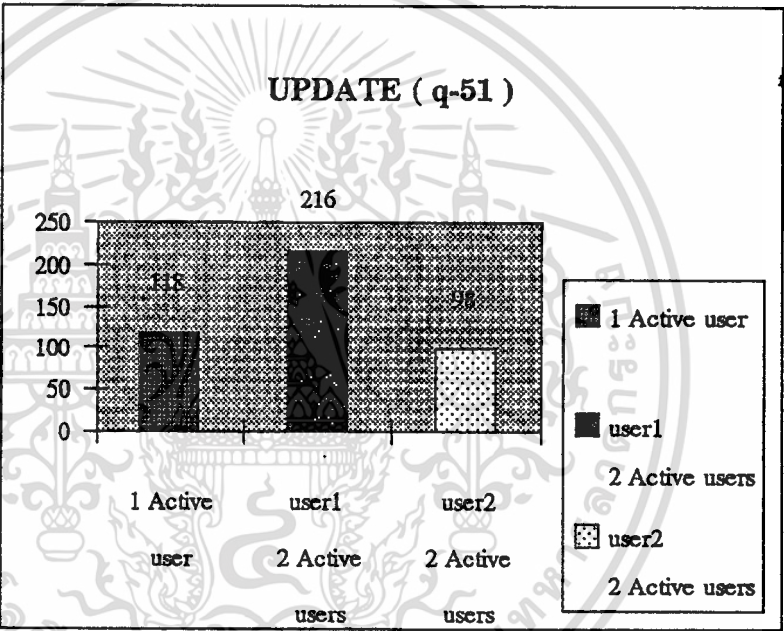
วิเคราะห์ผลการทดสอบ

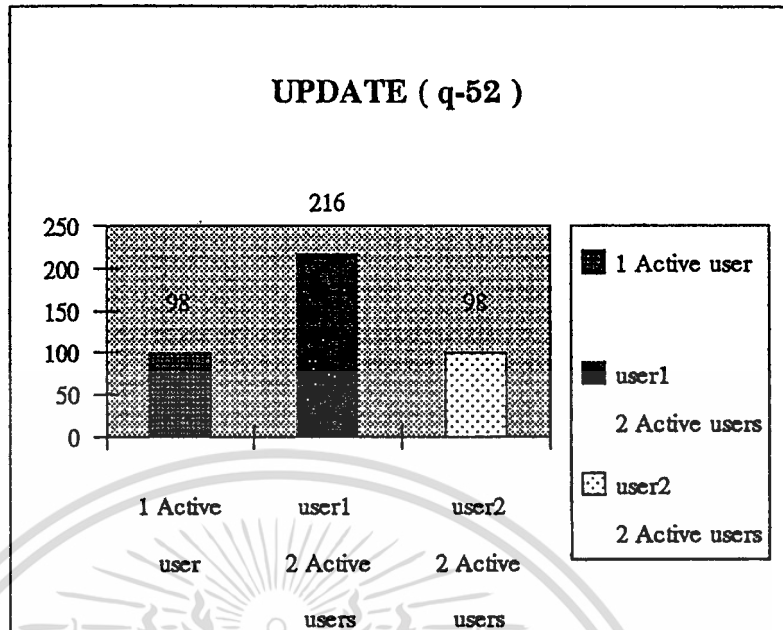
เนื่องจาก view ไม่ใช้การสร้างตารางขึ้นมาจริง ๆ แต่เป็นมโนภาพจาก base table ดังนั้น เวลาที่ใช้ในการสร้าง view จึงน้อยมาก และเวลาที่ได้จากการ scan view จะมากกว่า การ scan จาก base table อย่างเห็นได้ชัด

5) Update

ผลการทดสอบ

	1 Active User	2 Active User	
		q - 51	q - 52
q - 51	118	216	98
q - 52	98		





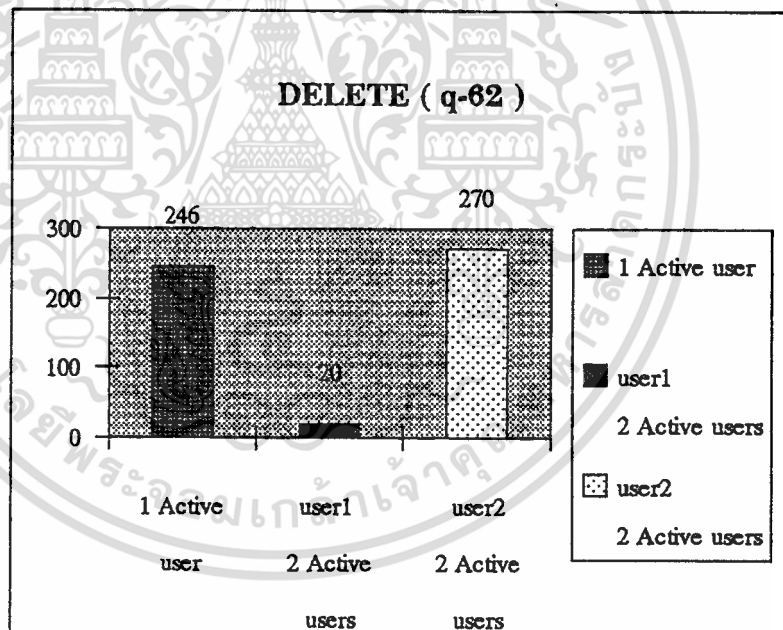
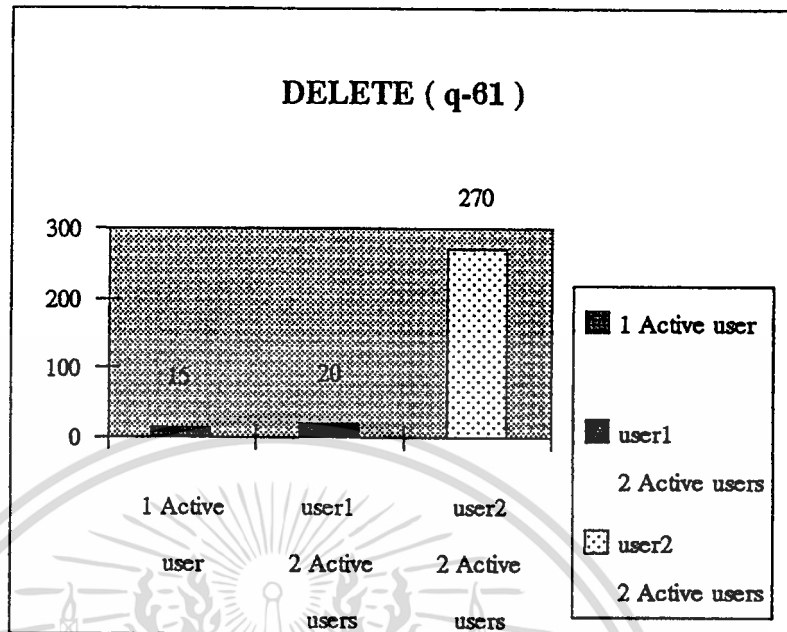
วิเคราะห์ผลการทดสอบ

เมื่อพิจารณาเวลาที่ใช้ในการทดสอบในการประมวลผลของ query พบว่า ในสภาวะแบบ multi-user คือมีผู้ใช้มากกว่า 1 คนและผู้ใช้นั้นแก้ไขข้อมูลในตารางเดียวกันในช่วงเวลาเดียวกัน จึงจำเป็นต้องมี concurrency control ซึ่ง INGRES ใช้กระบวนการ lock โดยประกันได้ว่า ขณะที่ transaction หนึ่งทำการแก้ไขข้อมูลอยู่ จะไม่มี transaction ใดสามารถใช้ข้อมูลเหล่านั้นได้ จนกว่า transaction ที่แก้ไขข้อมูลจะ commit แล้ว

6) Delete

ผลการทดสอบ

	1 Active User	2 Active User	
		q - 61	q - 62
q - 61	15	20	270
q - 62	246		



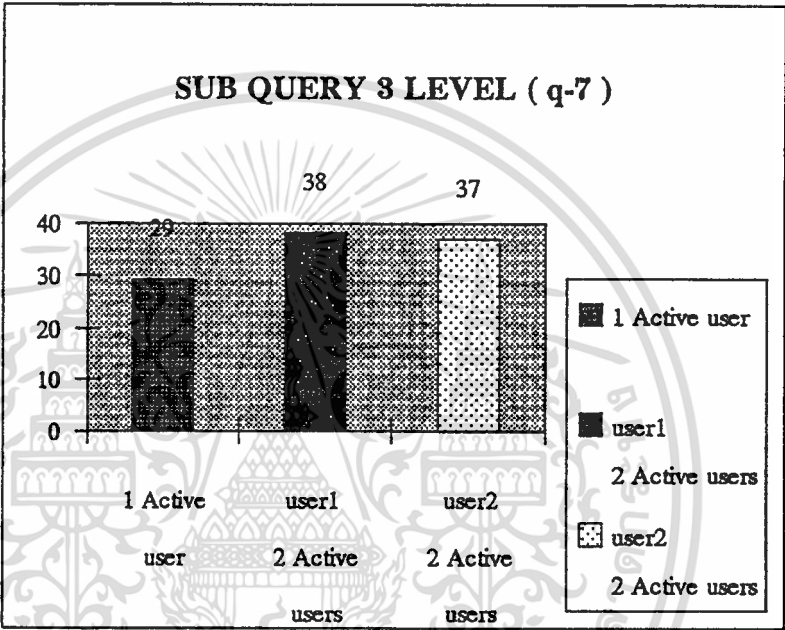
วิเคราะห์ผลการทดสอบ

จากผลการทดสอบจะพบว่าในการ access ตารางเดียวกันพร้อม ๆ กัน หลาย ๆ คน แล้ว ถ้าเป็นการ delete INGRES จะทำการ lock ระดับตารางเอาไว้ โดยจะ lock ตารางให้ผู้ใช้ที่เข้าไปก่อนได้ทำงานเสร็จ แล้วจึงจะปล่อย lock ให้ ผู้ใช้ อื่น ๆ ได้ทำต่อไป

7) Subquery 3 ชั้น

ผลการทดสอบ

	1 Active User	2 Active User	
q - 7	29	38	37



วิเคราะห์ผลการทดสอบ

ในการทดสอบการทำงานแบบ concurrency แต่ละ terminal จะใช้เวลาประมวลผลใกล้เคียงกัน เนื่องจาก query นี้เป็นการค้นหาข้อมูลเพียงอย่างเดียวไม่มีการแก้ไขเปลี่ยนแปลงข้อมูลในตาราง การ lock จึงอยู่ใน mode ของ share lock (S-lock) ทำให้สามารถทำงานพร้อมกันได้ และ response time ที่เพิ่มขึ้นนั้น เนื่องมาจาก CPU ต้อง share เวลาให้กับ transaction จึงทำให้เวลาเพิ่มขึ้น

7.3 ผลการทดสอบประสิทธิภาพของ INFORMIX version 5

Run บนเครื่อง IBM RISC/6000 Serie 300

O/S	AIX/UNIX
RAM	64 MB
Harddisk	2 GB
Terminal	Dumb 2 terminals

ลำดับขั้นตอนการทดสอบ

1. สร้างฐานข้อมูล

เข้าสู่โปรแกรมภาษา SQL โดยคำสั่ง ISQL และเลือกที่ menu database และ create ใส่ชื่อว่า TESTDB

2. สร้าง ตาราง

เข้าสู่โปรแกรมภาษา SQL โดยคำสั่ง ISQL TESTDB
สร้างตารางที่ 1

CREATE TABLE STUDENT

(ID_NO	CHAR(9) NOT NULL,
PRENAME	CHAR(2),
SPEC_CODE	CHAR(1),
T_NAME	CHAR(40) NOT NULL,
E_NAME	CHAR(34),
FAC_CODE	CHAR(2),
MAJ_CODE	CHAR(2),
FAC_COUNT	CHAR(1),
REG_STATUS	CHAR(1),
OLD_REG_STATUS	CHAR(1),
CERT_STATUS	CHAR(1),
BIRTH_DATE	INTEGER)

สร้างตารางที่ 2

CREATE TABLE SUBJECT

(STATUS_CODE	CHAR(1),
SUMPACK	CHAR(1),
BATCH	CHAR(3),
SERIAL	CHAR(5),
SUMPACK	INTEGER,
NEAR_GRAD	CHAR(1),
ID_NO	CHAR(9) NOT NULL,
NO_REP_HIS	CHAR(1),
CODE_FLAG	CHAR(1),
YY_MM_DD	INTEGER,
PERIOD	CHAR(1),
COUNTER	INTEGER,
BILL_NO	INTEGER,
TIME	INTEGER,
SUBJECT1	CHAR(5),
CREDIT1	CHAR(1),
SUBJECT2	CHAR(5),
CREDIT2	CHAR(1),
SUBJECT3	CHAR(5),
CREDIT3	CHAR(1),
SUBJECT4	CHAR(5),
CREDIT4	CHAR(1),
SUBJECT5	CHAR(5),
CREDIT5	CHAR(1),
SUBJECT6	CHAR(5),
CREDIT6	CHAR(1),
SUBJECT7	CHAR(5),
CREDIT7	CHAR(1),
SUBJECT8	CHAR(5),
CREDIT8	CHAR(1))

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3. นำข้อมูลเข้าตาราง

สร้างไฟล์ที่มีข้อมูลดังต่อไปนี้

ข้อมูลในไฟล์สำหรับตารางที่ 1 ชื่อ SCHEMA.1

FILE RU2555.DAT

(ID_NO	1-9,
PRENAME	10-11,
SPEC_CODE	12-12,
T_NAME	13-52,
E_NAME	53-86,
FAC_CODE	87-88,
MAJ_CODE	89-90,
FAC_COUNT	91-91,
REG_STATUS	92-92,
OLD_REG_STATUS	93-93,
CERT_STATUS	94-94,
BIRTH_DATE	95-102)

INSERT INTO STUDENT

VALUES (ID_NAME, PRENAME, SPEC_CODE, T_NAME,
E_NAME, FAC_CODE, MAJ_CODE, FAC_COUNT,
REG_STATUS, OLD_REG_STATUS, CERT_STATUS,
BIRTH_DATE)

ข้อมูลในไฟล์สำหรับตารางที่ 2 ชื่อ SCHEMA.2

FILE RU2444.DAT

(STAT_CODE	1-1,
BATCH	2-4,
SERIAL	5-9,
SUMPACK	10-11,
NEAR_GRAD	12-12,
ID_NO	13-21,
NO_REP_HIS	22-22,
CODE_FLAG	23-23,
YY_MM_DD	24-29,
PERIOD	30-30,
COUNTER	31-33,
BILL_NO	34-39,
TIME	40-47,
SUBJECT1	48-52,
CREDIT1	53-53,
SUBJECT2	54-58,
CREDIT2	59-59,
SUBJECT3	60-64,
CREDIT3	65-65,
SUBJECT4	66-70,
CREDIT4	71-71,
SUBJECT5	72-76,
CREDIT5	77-77,
SUBJECT6	78-82,
CREDIT6	83-83,
SUBJECT7	84-88,
CREDIT7	89-89,
SUBJECT8	90-94,
CREDIT8	95-95)

INSERT INTO SUBJECT

VALUES (STAT_CODE, BATCH, SERIAL, SUMPACK,
 NEAR_GRAD, ID_NO, NO_REP_HIS, CODE_FLAG,
 YY_MM_DD, PERIOD, COUNTER, BILL_NO, TIME,
 SUBJECT1, CREDIT1, SUBJECT2, CREDIT2,
 SUBJECT3, CREDIT3, SUBJECT4, CREDIT4,
 SUBJECT5, CREDIT5, SUBJECT6, CREDIT6,
 SUBJECT7, CREDIT7, SUBJECT8, CREDIT8)

กลับสู่ COMMAND PROMPT และใช้คำสั่ง

DBLOAD -D TESTDB -C SCHEMA.1

DBLOAD -D TESTDB -C SCHEMA.2

4. สร้างตัวชี้ข้อมูล

สร้างตัวชี้ข้อมูลในตารางที่ 1

CREATE INDEX ID0 ON TABLE STUDENT(ID_NO)

CREATE INDEX TNAME ON TABLE STUDENT(T_NAME)

CREATE INDEX ENAME ON TABLE STUDENT(E_NAME)

CREATE INDEX FACCODE ON TABLE STUDENT(FAC_CODE)

สร้างตัวชี้ข้อมูลในตารางที่ 2

CREATE INDEX ID1 ON TABLE SUBJECT(ID_NO)

5. ทดสอบ query

คำนวณเวลาที่ใช้ในการทำ query ด้วยการนำค่าที่อยู่ในตารางที่สร้างขึ้นเพื่อเก็บ

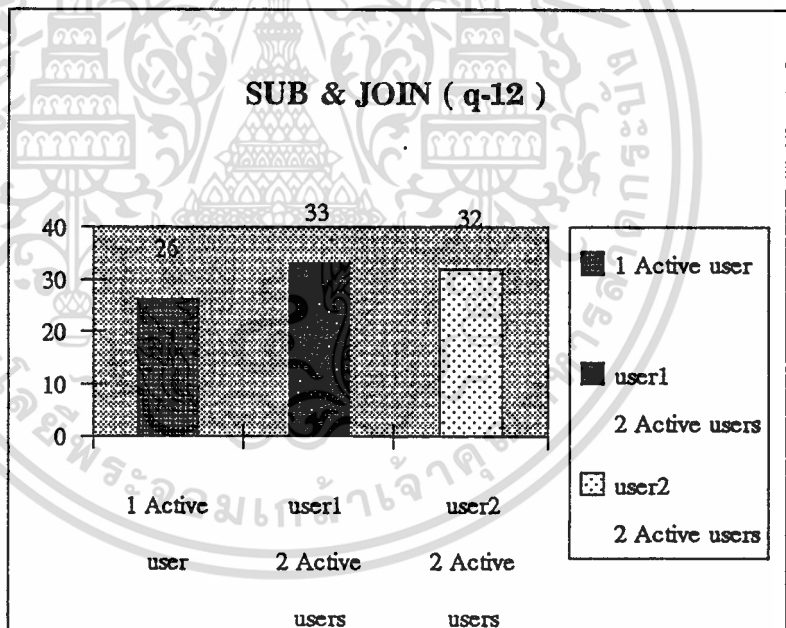
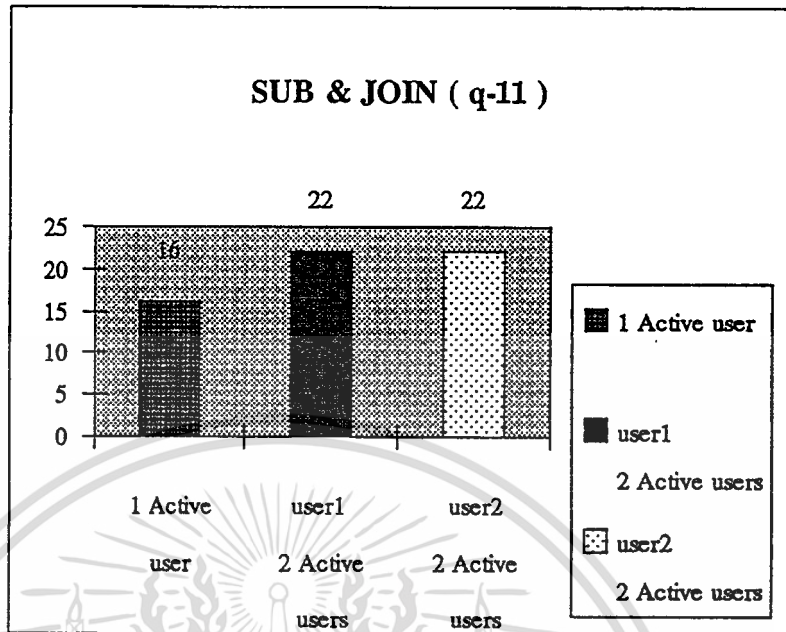
ค่าของเวลาในระบบมาลบกัน โดยใช้คำสั่ง :

INSERT INTO table VALUES (CURRENT) ไว้ทั้งก่อนและหลังการทำ query

1) Sub & Join

ผลการทดสอบ

	1 Active User	2 Active User	
q - 11	16	22	22
q - 12	26	33	32



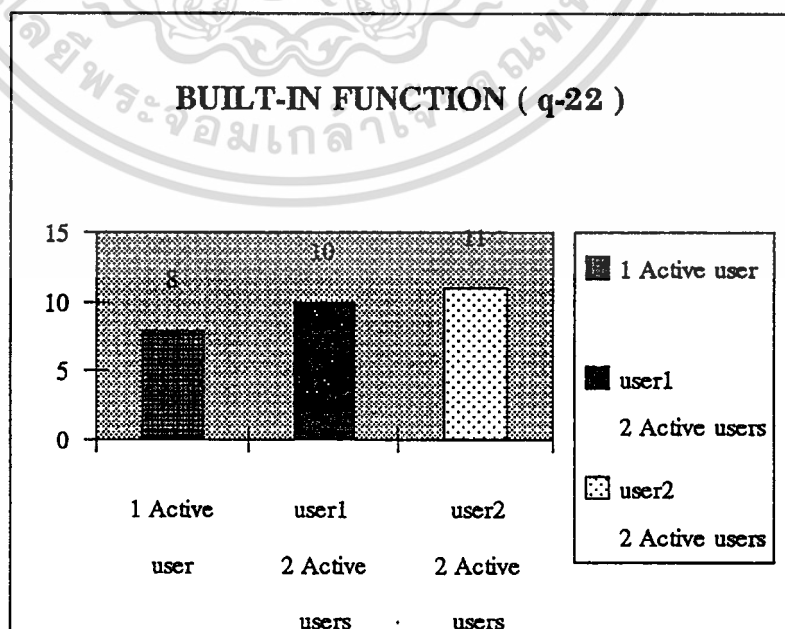
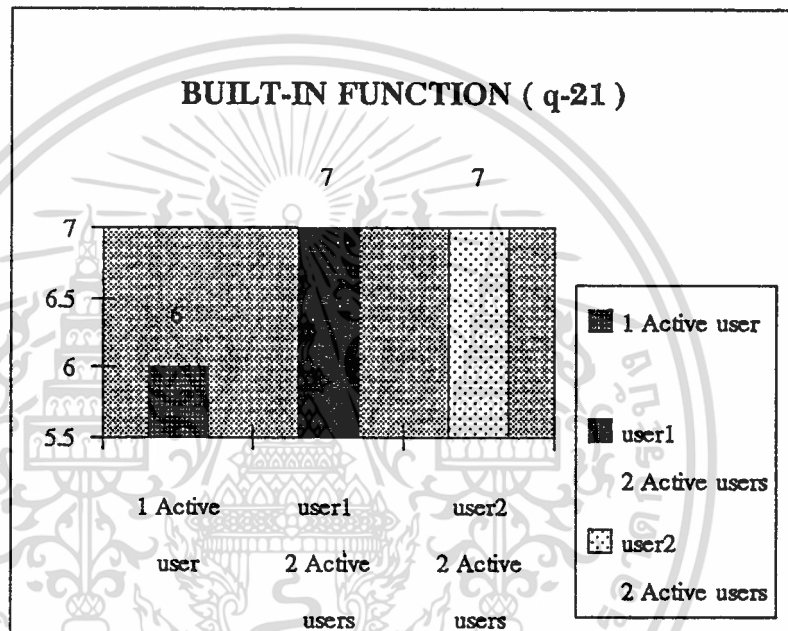
วิเคราะห์ผลการทดสอบ

จากผลการทดสอบจะเห็นว่า ผลต่างของเวลามีมาก และจะเพิ่มขึ้นเมื่อจำนวน ผู้ใช้ เพิ่มขึ้น ซึ่งจากหลักการออกแบบระบบจัดการฐานข้อมูลของ Codd ในกรณีที่ผลลัพธ์ของ query เหมือนกันและเลือกวิธี และเส้นทางการเข้าถึงข้อมูลต่างกัน ควรจะได้ response time เดียวกัน แสดงว่า INFORMIX ขาดคุณสมบัติข้อนี้

2) Built-in Function

ผลการทดสอบ

	1 Active User	2 Active User	
q - 21	6	7	7
q - 22	8	10	11



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

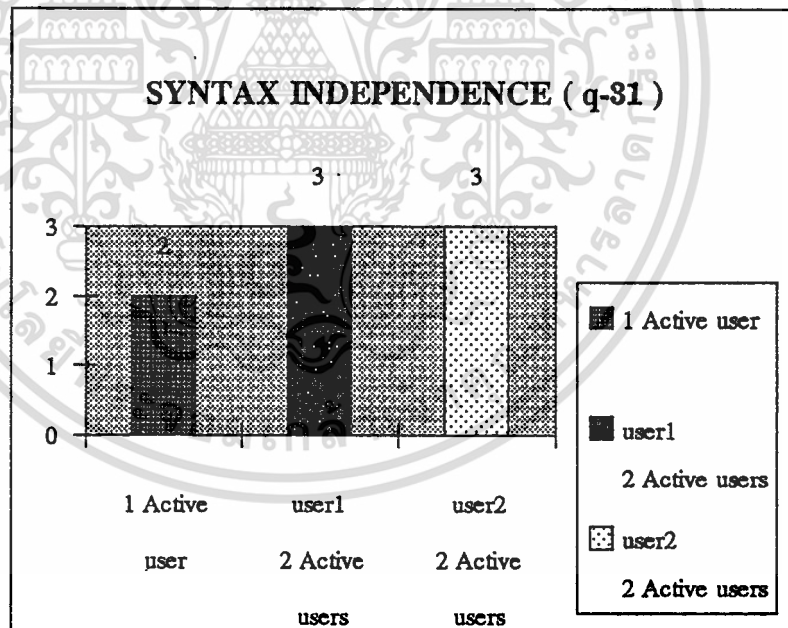
วิเคราะห์ผลการทดสอบ

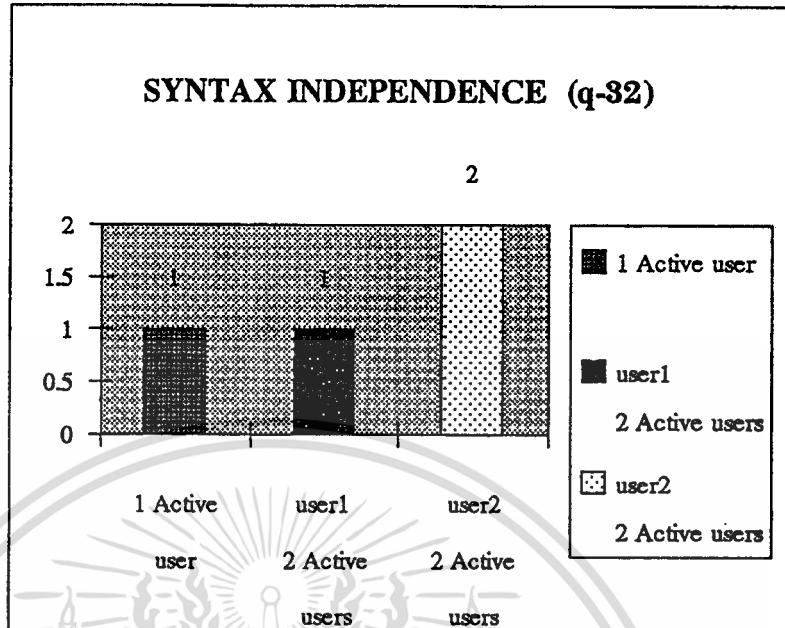
การทดสอบประสิทธิภาพในด้านการประมวลผล function ทางคณิตศาสตร์ ทั้งแบบเปรียบเทียบและหาค่าผลรวมนี้ใช้เวลาอย่างมากเมื่อเทียบกับ query อื่นๆทำให้คาดว่าใน function ทั้งแบบนี้ อาจนำค่า Statistic ใน Catalog มาช่วยในการประมวลผลด้วย

3) Syntax Independence

ผลการทดสอบ

	1 Active User	2 Active User	
q - 31	2	3	3
q - 32	1	1	2





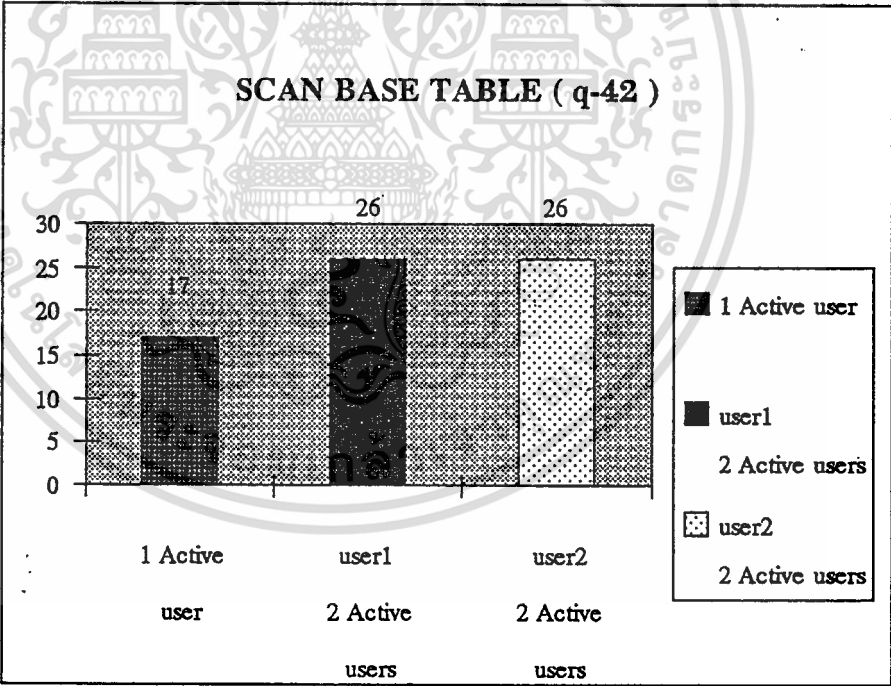
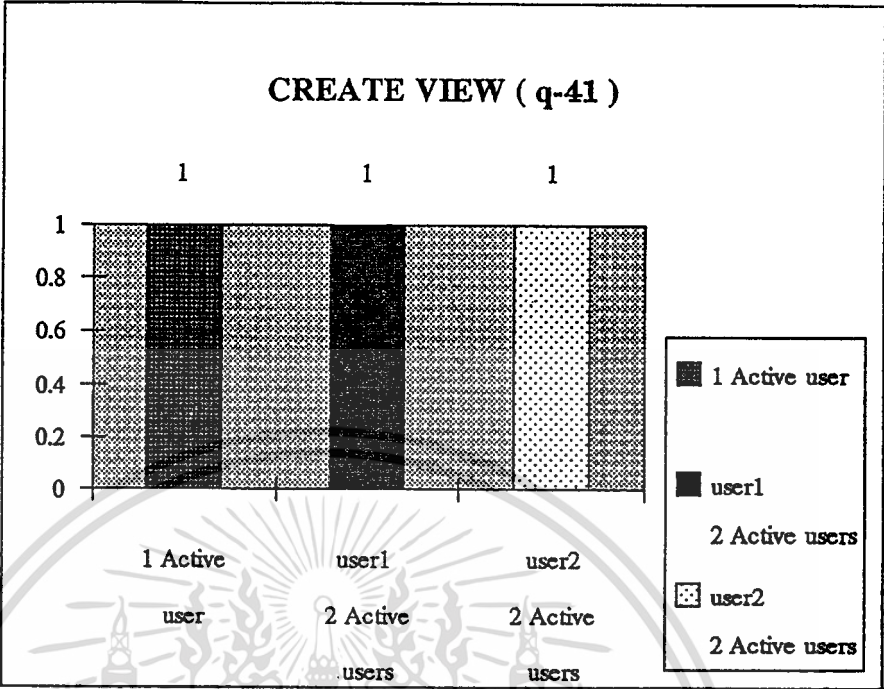
วิเคราะห์ผลการทดสอบ

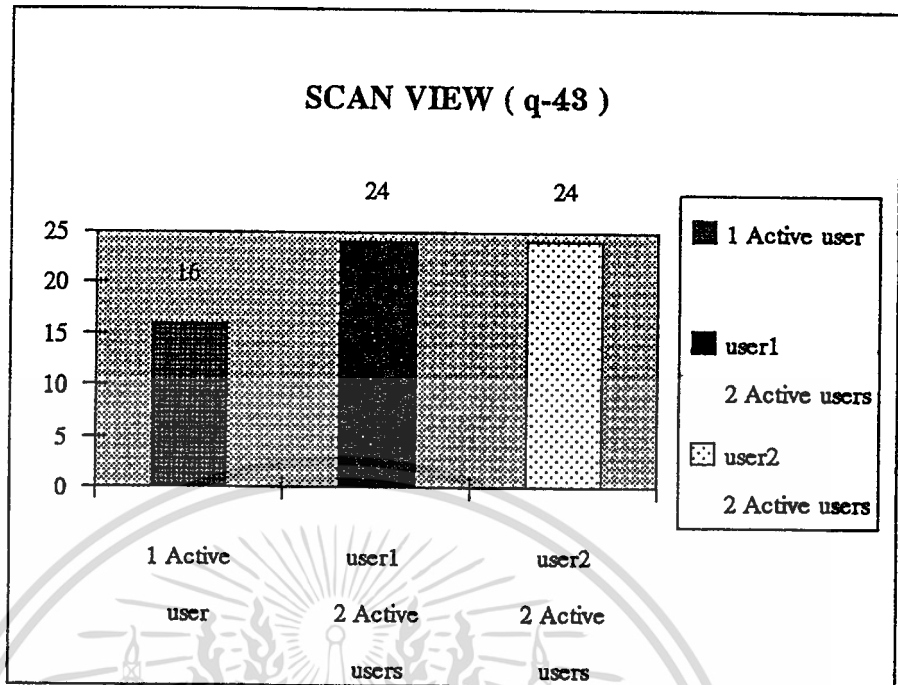
จากผลต่างของเวลา จะเห็นว่า ลำดับการทำ query มีผลต่อการประมวลผล แสดงว่า INFORMIX ขาดคุณสมบัติด้าน syntax independence และ query optimizer ของ INFORMIX ทำงานด้านนี้ไม่สมบูรณ์

4) Create View และ Scan ข้อมูลจาก View

ผลการทดสอบ

	1 Active User	2 Active User	
q - 41	1	1	1
q - 42	17	26	26
q - 43	16	24	24





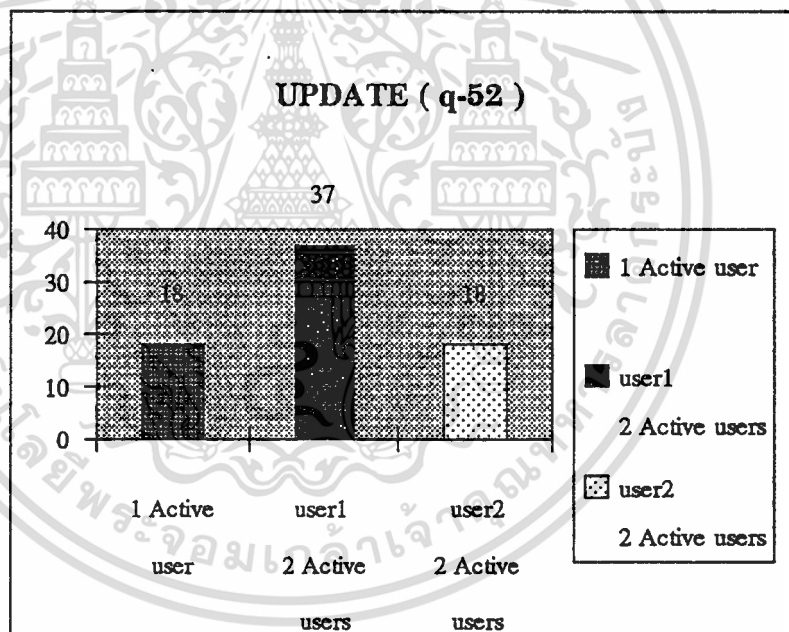
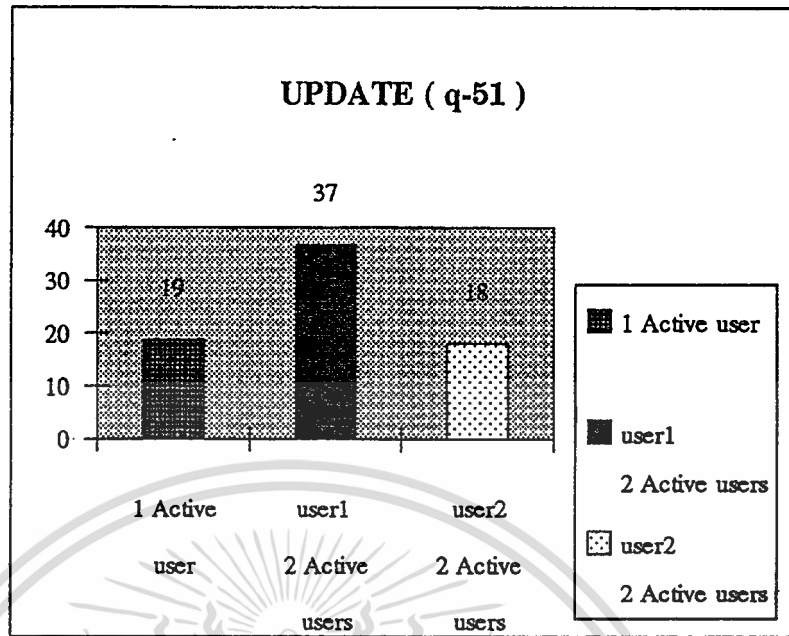
วิเคราะห์ผลการทดสอบ

การ create view จะใช้เวลาน้อยมากเพราะไม่ใช่ base table แต่เป็นการสร้างมโนภาพจาก base table ดังนั้นเวลาที่ใช้ในการสร้าง view จึงน้อยมาก นอกจากนี้ จะเห็นะเวลาที่ได้จากการ scan view จะมากกว่า การ scan จาก base table จนเห็นได้ชัด

5) Update

ผลการทดสอบ

	1 Active User	2 Active User	
		q - 51	q - 52
q - 51	19	37	18
q - 52	18		



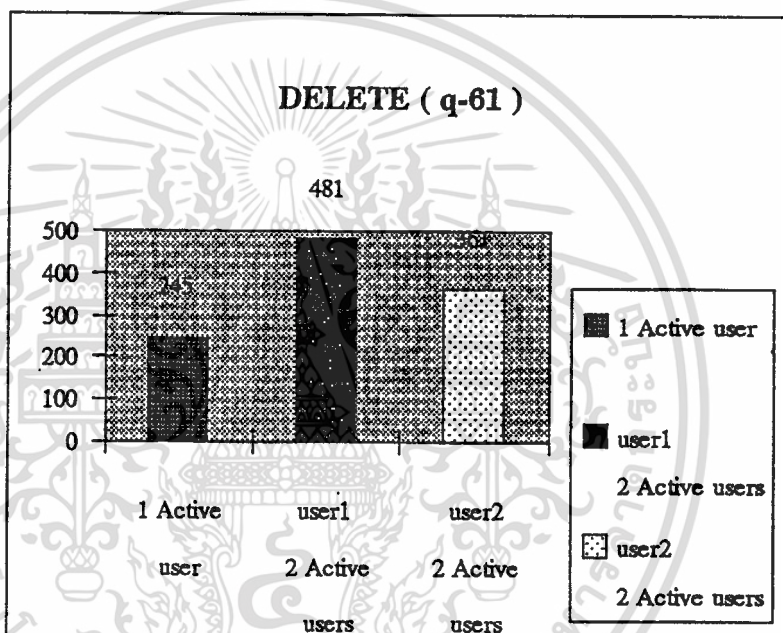
วิเคราะห์ผลการทดสอบ

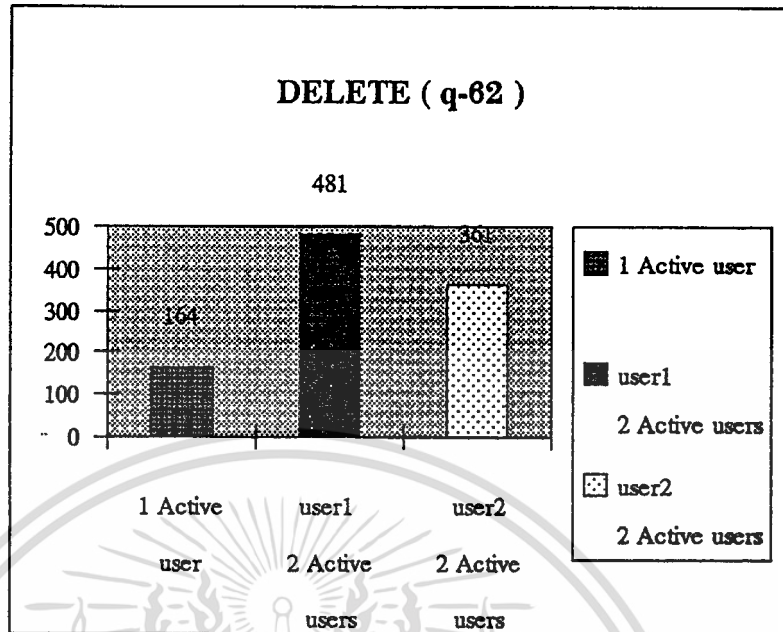
จากผลการทดสอบในการประมวลผลของ query จะเห็นว่าในสถานะที่มีผู้ใช้มากกว่า 1 คนและผู้ใช้นั้นแก้ไขข้อมูลในตารางเดียวกัน ในช่วงเวลาเดียวกัน ตัว INFORMIX ใช้กลไกในการ lock (lock mechanism) ทำให้ประกันได้ว่า ขณะที่ transaction ใด ๆ ตัวหนึ่งกำลังทำการแก้ไขข้อมูลอยู่ จะไม่มี transaction ใด สามารถใช้ข้อมูลนั้นได้จนกว่า transaction ที่แก้ไขอยู่จะ commit

6) Delete

ผลการทดสอบ

	1 Active User	2 Active User	
		q - 61	q - 62
q - 61	245	481	361
q - 62	164		





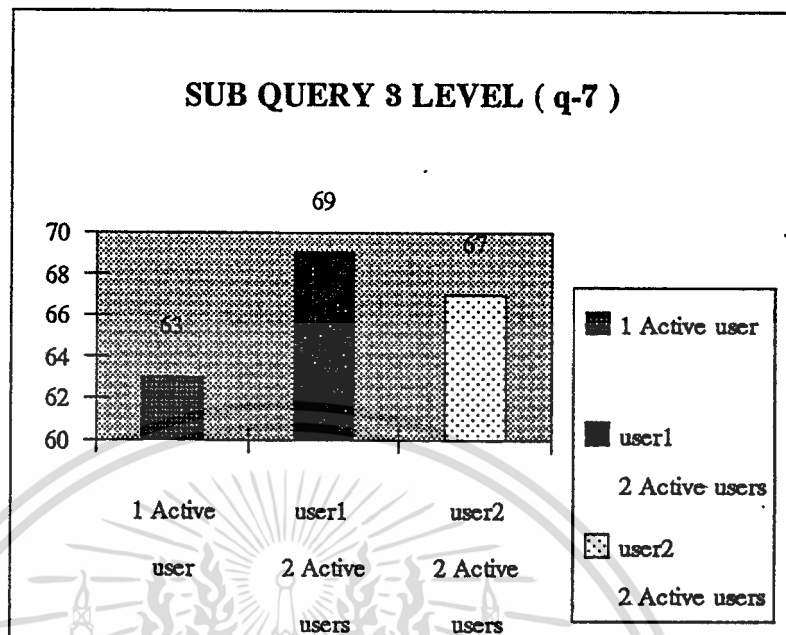
วิเคราะห์ผลการทดสอบ

จากผลการทดสอบทำให้สามารถสรุปได้ว่า ในการ access ตารางเดียวกันพร้อม ๆ กันหลาย ๆ คน นั้นแล้ว ถ้าเป็นการ delete INFORMIX จะทำการ lock ระดับ ตาราง เอาไว้ไม่ให้ access หลาย ๆ คนโดยจะ lock ตารางให้ ผู้ใช้ ที่เข้าไปก่อนได้ใช้งานจนเสร็จสิ้นแล้วจึงค่อยปล่อย lock ให้ ผู้ใช้ อื่นได้ทำงานต่อไปน จะเข้าไปใช้ตารางนั้นไม่ได้ โดยจะแสดง 'table lock busy' แล้วจะ reject transaction ออกมา

7) Subquery 3 ชั้น

ผลการทดสอบ

	1 Active User	2 Active User	
q - 7	63	69	67



วิเคราะห์ผลการทดสอบ

ในการทำ Subquery 3 ชั้น ของ INFORMIX ในขณะทำงาน 1 active users. จะให้ค่าผลต่างใกล้เคียงกับการทดสอบ 2 active user แสดงว่าในขณะที่ ผู้ใช้ คนหนึ่งทำ transaction อยู่ ผู้ใช้ อีกคน สามารถทำ transaction ไปพร้อม ๆ กันได้ด้วย เพราะเป็น การ look up table อย่างเดียวไม่มีการแก้ไขตาราง mode ในการ lock จึงเป็น s-lock จากเวลาที่ต่างกันเพียงเล็กน้อย เป็นเพราะ CPU ต้อง share time ให้แต่ละ process

7.4 ผลการทดสอบประสิทธิภาพของ EMPRESS version 6

Run บนเครื่อง PC CPU 486DX/2-80 (AMD)

O/S	Interactive UNIX
RAM	16 MB
Harddisk	500 MB SCSI
Terminal	PC 4 terminals

ลำดับขั้นตอนการทดสอบ

1. สร้างฐานข้อมูล โดยใช้คำสั่ง EMPMKDB TESTDB
2. สร้าง ตาราง

เข้าสู่โปรแกรมภาษา SQL โดยใช้คำสั่ง EMPSQL TESTDB

สร้างตารางที่ 1

CREATE TABLE STUDENT

(ID_NO	CHAR(9,1) NOT NULL,
PRENAME	CHAR(2,3),
SPEC_CODE	CHAR(1,3),
T_NAME	CHAR(40,3) NOT NULL,
E_NAME	CHAR(34,1),
FAC_CODE	CHAR(2,3),
MAJ_CODE	CHAR(2,3),
FAC_COUNT	CHAR(1,3),
REG_STATUS	CHAR(1,3),
OLD_REG_STATUS	CHAR(1,3),
CERT_STATUS	CHAR(1,3),
BIRTH_DATE	LONGINTEGER)

สร้างตารางที่ 2

CREATE TABLE SUBJECT

(STATUS_CODE	CHAR(1,3),
SUMPACK	CHAR(1,3),
BATCH	CHAR(3,3),
SERIAL	CHAR(5,3),
SUMPACK	INTEGER,
NEAR_GRAD	CHAR(1,3),
ID_NO	CHAR(9,1) NOT NULL,
NO_REP_HIS	CHAR(1,3),
CODE_FLAG	CHAR(1,3),
YY_MM_DD	LONGINTEGER,
PERIOD	CHAR(1,3),
COUNTER	INTEGER,
BILL_NO	LONGINTEGER,
TIME	LONGINTEGER,
SUBJECT1	CHAR(5,3),
CREDIT1	CHAR(1,3),
SUBJECT2	CHAR(5,3),
CREDIT2	CHAR(1,3),
SUBJECT3	CHAR(5,3),
CREDIT3	CHAR(1,3),
SUBJECT4	CHAR(5,3),
CREDIT4	CHAR(1,3),
SUBJECT5	CHAR(5,3),
CREDIT5	CHAR(1,3),
SUBJECT6	CHAR(5,3),
CREDIT6	CHAR(1,3),
SUBJECT7	CHAR(5,3),
CREDIT7	CHAR(1,3),
SUBJECT8	CHAR(5,3),
CREDIT8	CHAR(1,3))

3. นำข้อมูลเข้าตาราง

สร้างไฟล์ที่มีข้อมูลดังต่อไปนี้

ข้อมูลในไฟล์สำหรับตารางที่ 1 ชื่อ SPEC.1

LENGTH 103

INTO STUDENT

(ID_NO	POSITION(1:9),
PRENAME	POSITION(10:11),
SPEC_CODE	POSITION(12:12),
T_NAME	POSITION(13:52),
E_NAME	POSITION(53:86),
FAC_CODE	POSITION(87:88),
MAJ_CODE	POSITION(89:90),
FAC_COUNT	POSITION(91:91),
REG_STATUS	POSITION(92:92),
OLD_REG_STATUS	POSITION(93:93),
CERT_STATUS	POSITION(94:94),
BIRTH_DATE	POSITION(95:102))

ข้อมูลในไฟล์สำหรับตารางที่ 2 ชื่อ SPEC.2

LENGTH 96

INTO SUBJECT

(STAT_CODE	POSITION(1:1),
BATCH	POSITION(2:4),
SERIAL	POSITION(5:9),
SUMPACK	POSITION(10:11),
NEAR_GRAD	POSITION(12:12),
ID_NO	POSITION(13:21),
NO_REP_HIS	POSITION(22:22),
CODE_FLAG	POSITION(23:23),
YY_MM_DD	POSITION(24:29),
PERIOD	POSITION(30:30),
COUNTER	POSITION(31:33),
BILL_NO	POSITION(34:39),

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

TIME	POSITION(40:47),
SUBJECT1	POSITION(48:52),
CREDIT1	POSITION(53:53),
SUBJECT2	POSITION(54:58),
CREDIT2	POSITION(59:59),
SUBJECT3	POSITION(60:64),
CREDIT3	POSITION(65:65),
SUBJECT4	POSITION(66:70),
CREDIT4	POSITION(71:71),
SUBJECT5	POSITION(72:76),
CREDIT5	POSITION(77:77),
SUBJECT6	POSITION(78:82),
CREDIT6	POSITION(83:83),
SUBJECT7	POSITION(84:88),
CREDIT7	POSITION(89:89),
SUBJECT8	POSITION(90:94),
CREDIT8	POSITION(95:95))

กลับสู่ COMMAND PROMPT และใช้คำสั่ง

EMPDBLD TESTDB SPEC.1 RU2555.DAT

EMPDBLD TESTDB SPEC.2 RU2444.DAT

4. สร้างตัวชี้ข้อมูล

สร้างตัวชี้ข้อมูลในตารางที่ 1

CREATE INDEX ID0 ON TABLE STUDENT.ID_NO

CREATE INDEX TNAME ON TABLE STUDENT.T_NAME

CREATE INDEX ENAME ON TABLE STUDENT.E_NAME

CREATE INDEX FACCODE ON TABLE STUDENT.FAC_CODE

สร้างตัวชี้ข้อมูลในตารางที่ 2

CREATE INDEX ID1 ON TABLE SUBJECT.ID_NO

5. ทดสอบ query

คำนวณเวลาที่ใช้ในการทำ query ด้วยการนำค่าที่อยู่ในตารางที่สร้างขึ้นเพื่อเก็บค่าของเวลาในระบบมาลบกัน โดยใช้คำสั่ง :

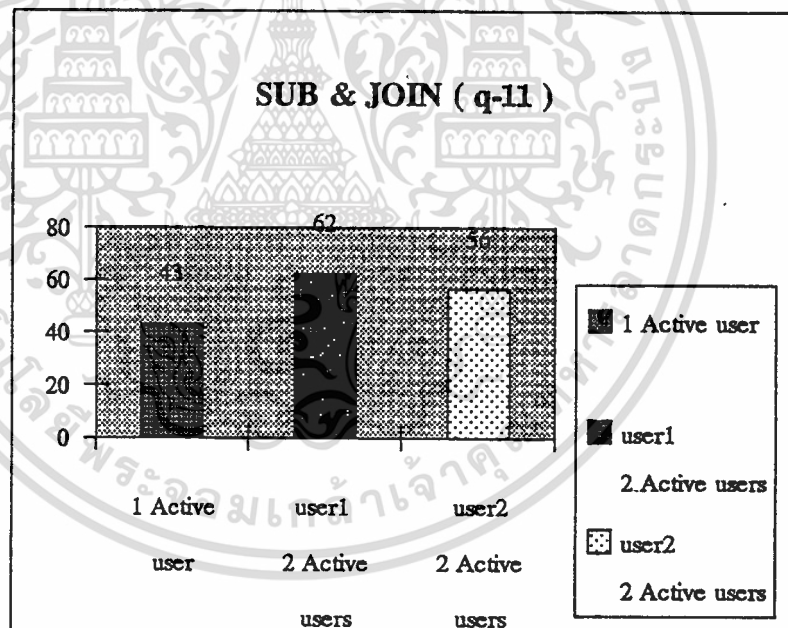
INSERT INTO table SET column VALUES (NOW)

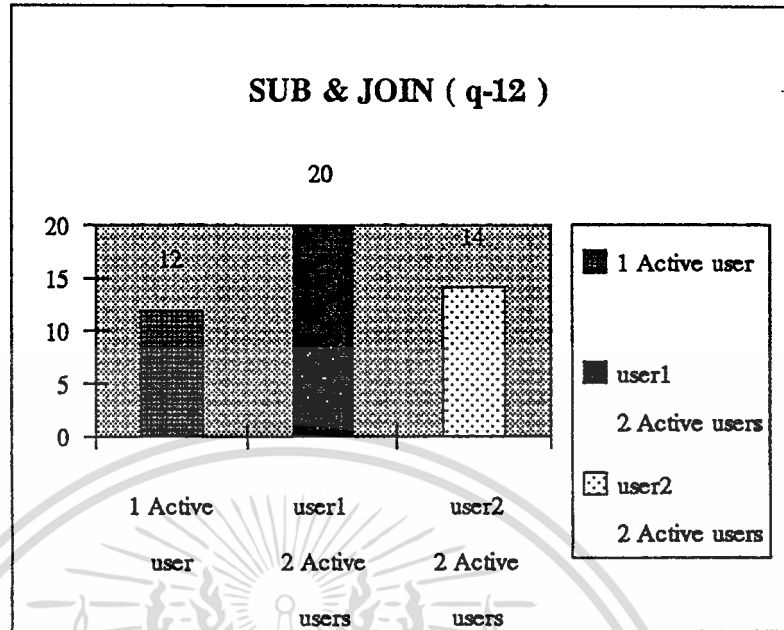
ไว้ทั้งก่อนและหลังการทำ query

1) Sub & Join

ผลการทดสอบ

	1 Active User	2 Active-User	
q - 11	43	62	56
q - 12	12	20	14





วิเคราะห์ผลการทดสอบ

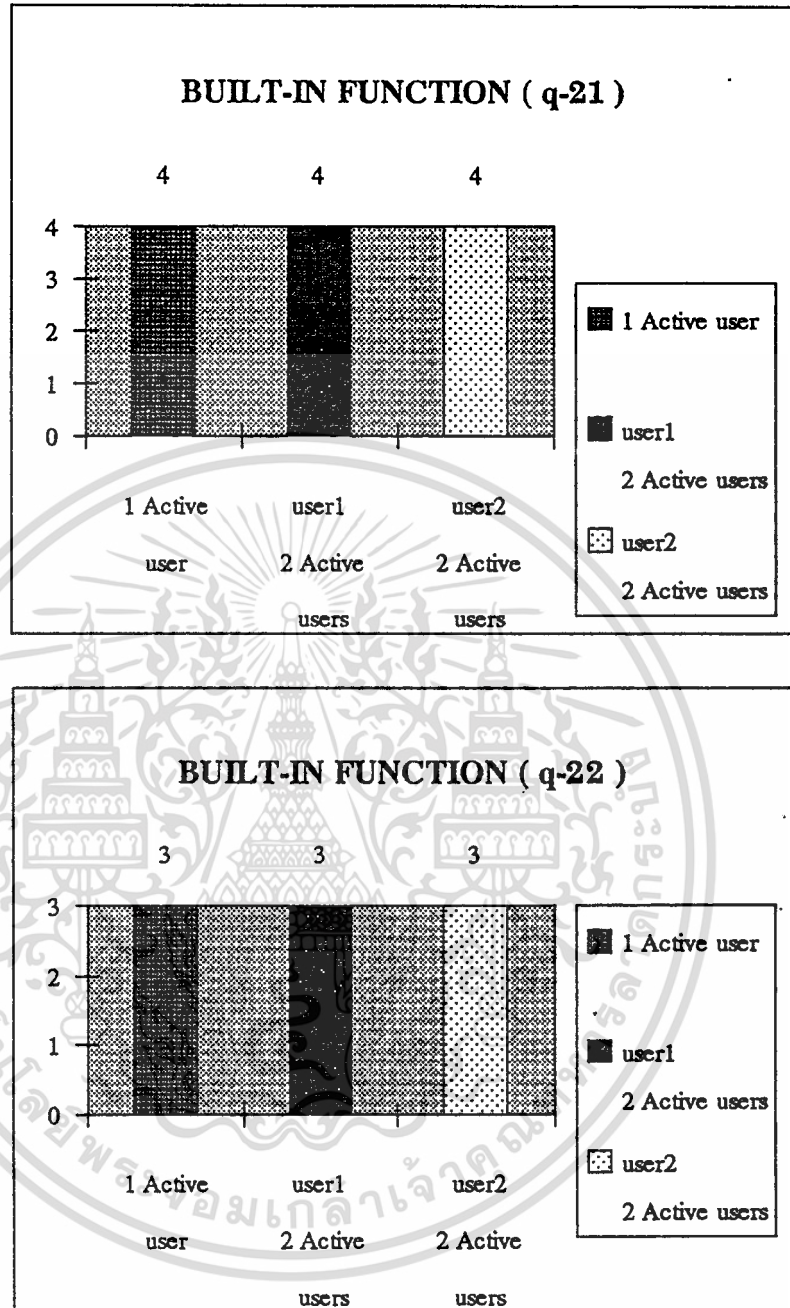
จากผลการทดสอบจะเห็นว่า เวลาที่ใช้ในการประมวลผลต่างกันมากและจะยิ่งต่างกันมากขึ้นเมื่อจำนวน ผู้ใช้ เพิ่มขึ้น

และจากหลักการออกแบบระบบจัดการฐานข้อมูลของ Codd นั้น ในกรณีที่ได้ผลลัพธ์เหมือนกัน แต่ใช้วิธีการ query และวิธีการเลือกเส้นทางที่ต่างกัน ควรจะได้ response time เท่ากัน ซึ่งจากผลการทดสอบ EMPRESS ไม่มีคุณสมบัติข้อนี้

2) Built-in Function

ผลการทดสอบ

	1 Active User	2 Active User	
q - 21	4	4	4
q - 22	3	3	3



วิเคราะห์ผลการทดสอบ

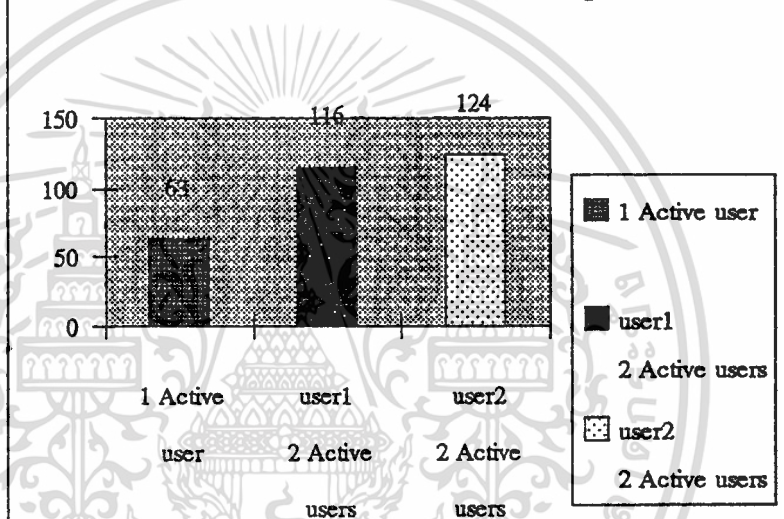
การทดสอบประสิทธิภาพในด้านการประมวลผล function ทางคณิตศาสตร์ ทั้งแบบเปรียบเทียบและหาค่าผลรวมนี้ใช้เวลาไม่น้อยเมื่อเทียบกับ query อื่น ๆ

3) Syntax Independence

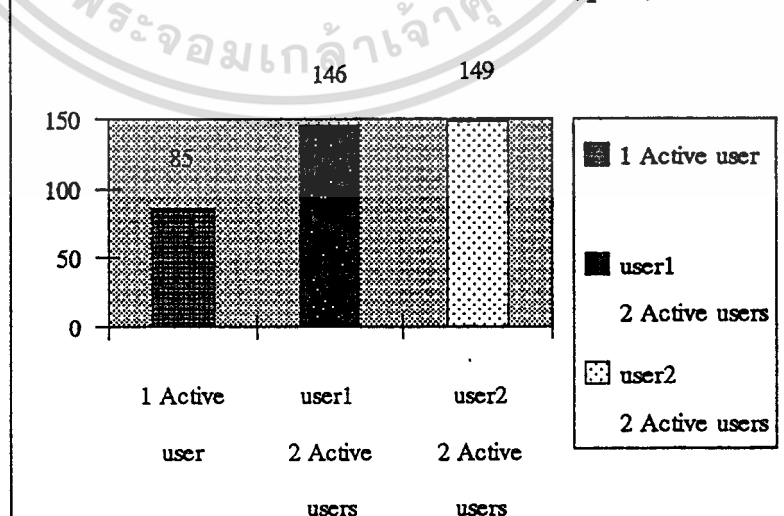
ผลการทดสอบ

	1 Active User	2 Active User	
q - 31	63	116	124
q - 32	85	146	149

SYNTAX INDEPENDENCE (q-31)



SYNTAX INDEPENDENCE (q-32)



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

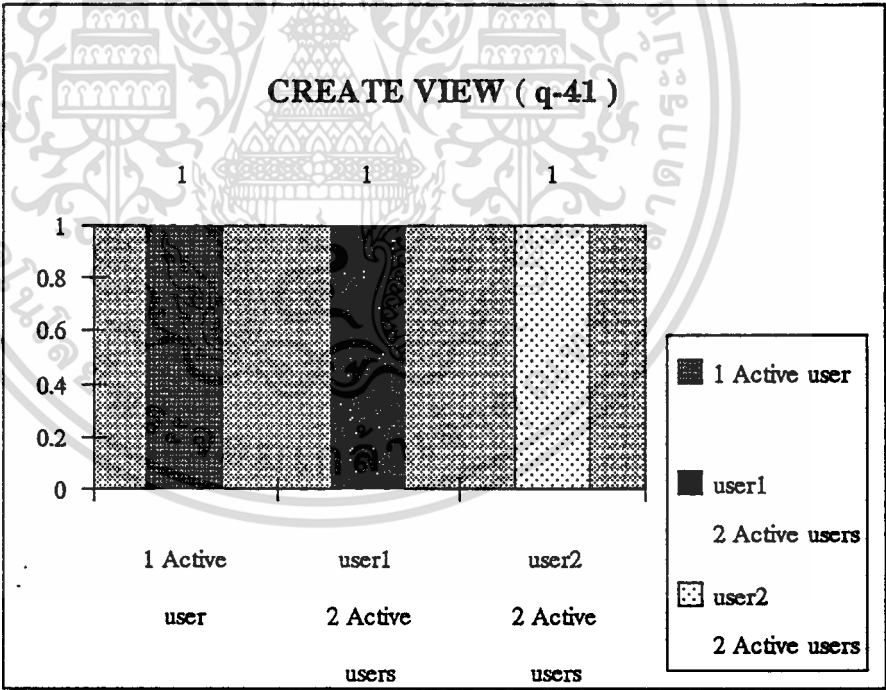
วิเคราะห์ผลการทดสอบ

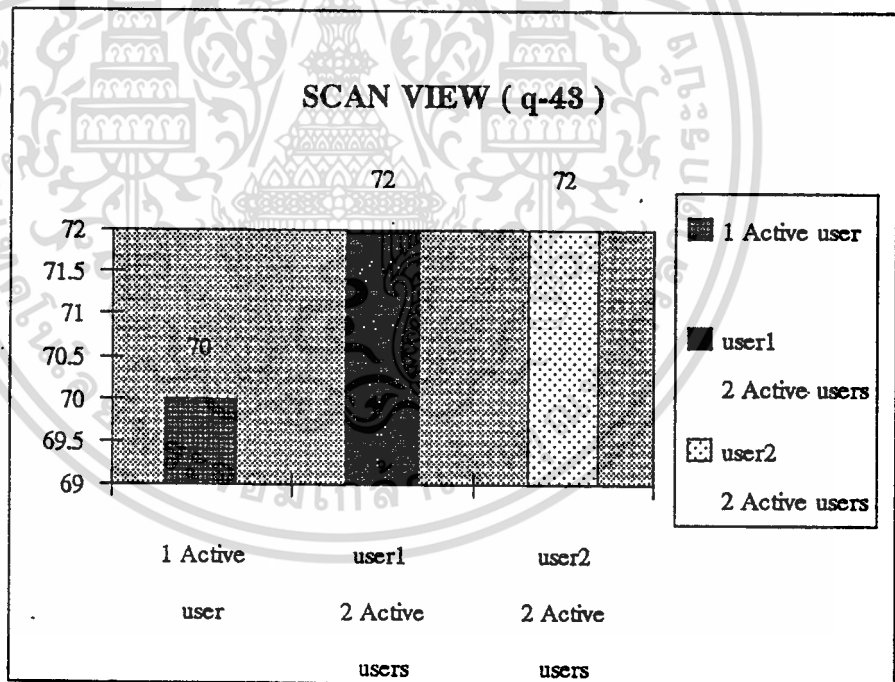
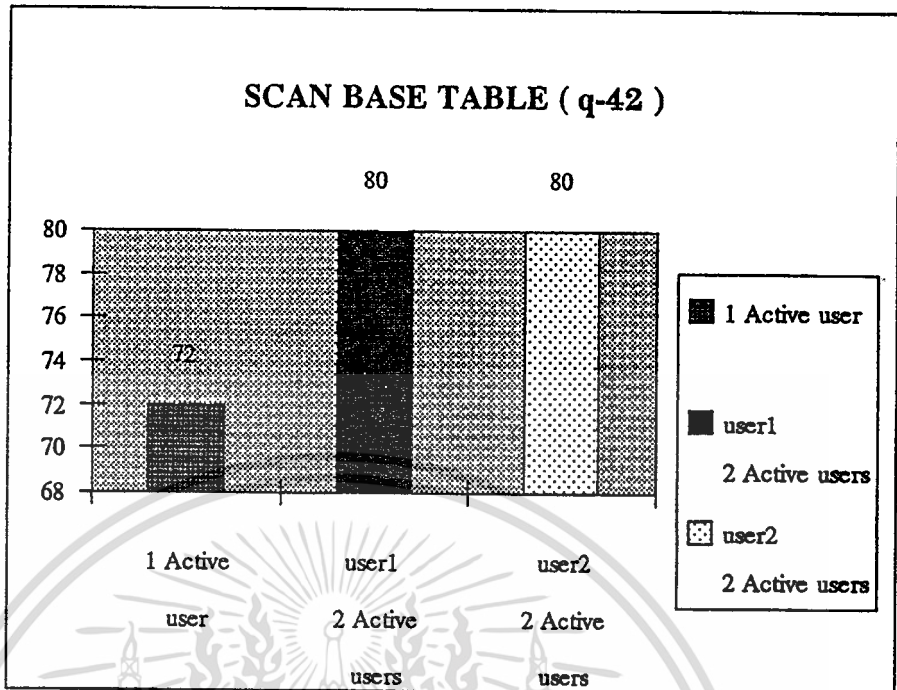
จากผลต่างของเวลา จะเห็นว่า ลำดับการทำ query มีผลต่อการประมวลผล แสดงว่า EMPRESS ขาดคุณสมบัติด้าน syntax independence และ query optimizer ของ EMPRESS ทำงานด้านนี้ไม่สมบูรณ์

4) Create View และ Scan ข้อมูลจาก View

ผลการทดสอบ

	1 Active User	2 Active User	
q - 41	1	1	1
q - 42	72	80	80
q - 43	70	72	72





วิเคราะห์ผลการทดสอบ

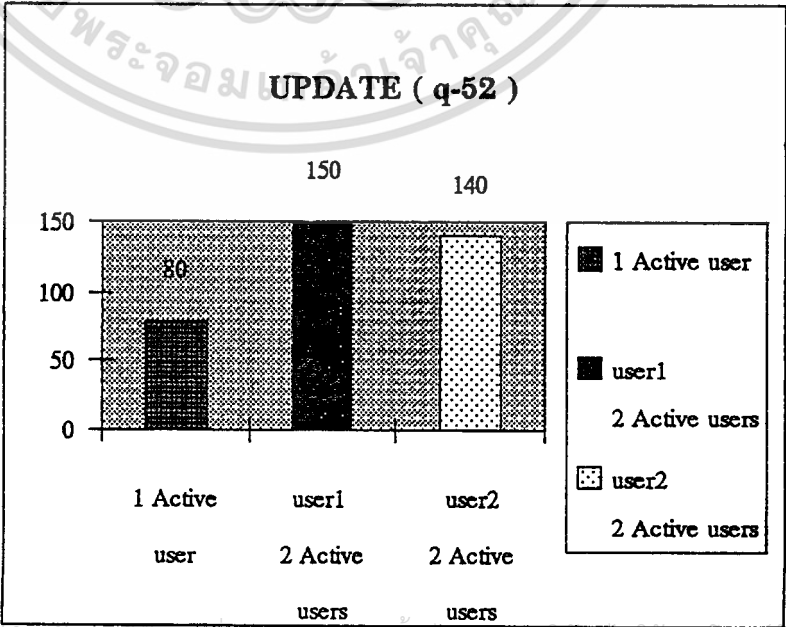
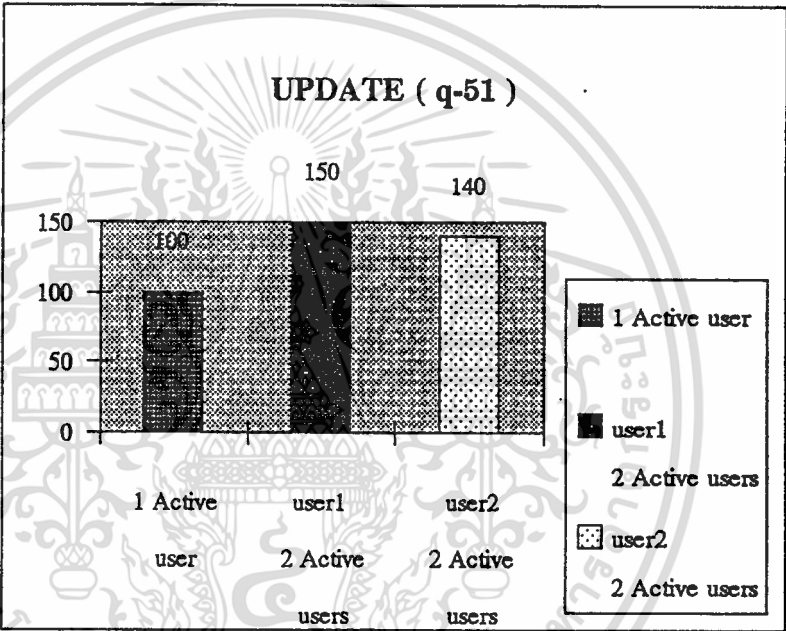
การ create view จะใช้เวลาน้อยมากเพราะไม่ใช่ base table แต่เป็นการสร้างมโนภาพจาก base table ดังนั้นเวลาที่ใช้ในการสร้าง view จึงน้อยมาก นอกจากนี้ จะเห็นะเวลาที่ได้จากการ scan view จะมากกว่า การ scan จาก base table

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5) Update

ผลการทดสอบ

	1 Active User	2 Active User	
		q - 51	q - 52
q - 51	100	150	140
q - 52	80		



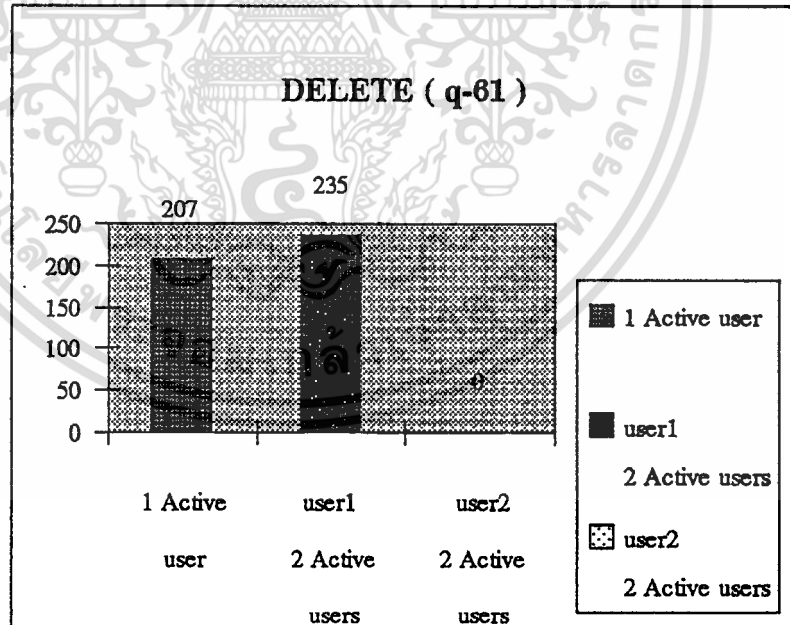
วิเคราะห์ผลการทดสอบ

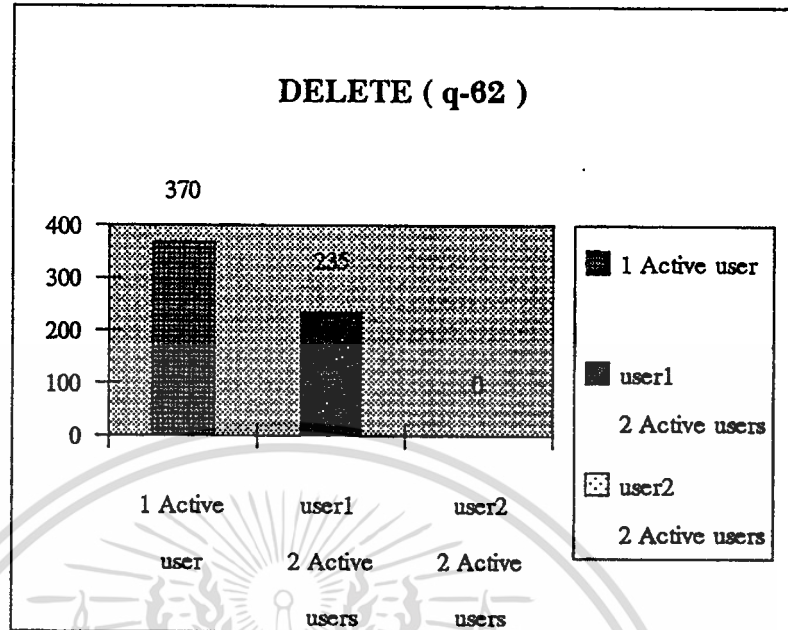
จากผลการทดสอบ ผลต่างของเวลาต่างกันไม่มาก แสดงว่า ในการ update data ของ EMPRESS นั้น จะใช้การ lock ในระดับ record หรือ row โดยผู้ใช้ที่เข้าไปใช้ก่อนจะเสร็จก่อนผู้ใช้ที่เข้าไปแก้ไขทีหลัง

6) Delete

ผลการทดสอบ

	1 Active User	2 Active User	
q - 31	207	235	reject
q - 32	370		





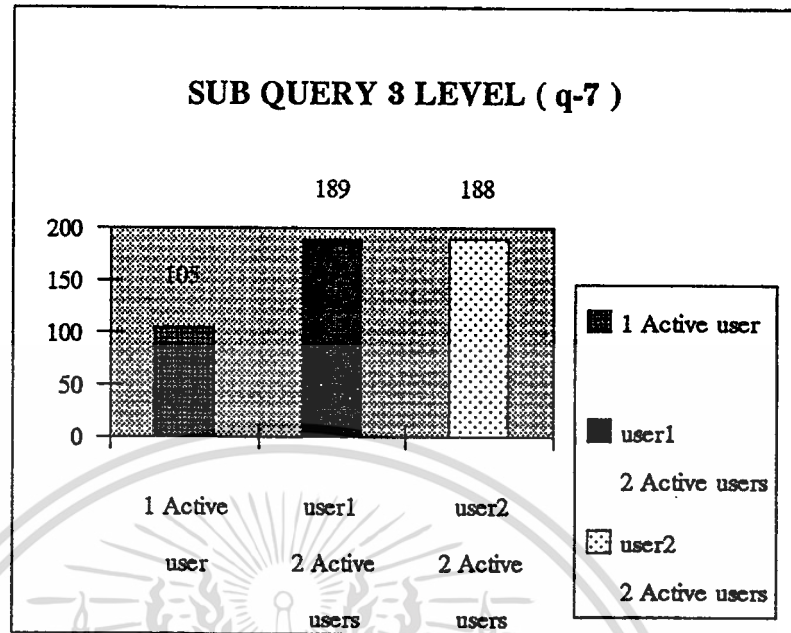
วิเคราะห์ผลการทดสอบ

จากผลการทดสอบจะพบว่าในการ access ตารางเดียวกันพร้อม ๆ กัน หลาย ๆ ผู้ใช้ แล้ว ถ้าเป็นการ delete EMPRESS จะทำการ lock ระดับ ตาราง เอาไว้โดยจะ lock ตารางให้ ผู้ใช้ ที่เข้าไปก่อน โดย ผู้ใช้ อื่น จะเข้าไปใช้ ตาราง นั้นไม่ได้ โดยจะแสดง 'table lock busy' แล้ว จะ reject transaction ออกมา

7) Subquery 3 ชั้น

ผลการทดสอบ

	1 Active User	2 Active User	
q - 7	105	189	188



วิเคราะห์ผลการทดสอบ

จากเวลาที่ได้มาจะเห็นว่า ได้ใกล้เคียงกันมาก เพราะเป็นการค้นหาข้อมูลเพียงอย่างเดียว จึงเป็นการ lock แบบ s-lock ทำให้สามารถทำงานพร้อมกันได้ และ response time ที่เพิ่มมากขึ้นเนื่องจาก CPU ต้อง share time ให้แต่ละ query

บทที่ 8

กฎของ E.F. Codd ในการวิเคราะห์ระบบจัดการฐานข้อมูล

8.1 กฎ 12 ข้อของ Codd ในการวิเคราะห์ระบบจัดการฐานข้อมูล

E.F. Codd ได้เสนอกฎเกณฑ์ไว้ทั้งหมด 12 ข้อเพื่อใช้เป็นมาตรฐานเบื้องต้นในการเปรียบเทียบและวิเคราะห์ software กฎทั้ง 12 ข้อนี้ได้สรุปถึงปัจจัยต่างๆ ที่ ระบบจัดการฐานข้อมูล จำเป็นต้องมีดังนี้

นิยามของกฎของ Codd 12 ข้อ

กฎข้อที่ 1 The Information Rule

ข้อมูลต่าง ๆ ในระบบจัดการฐานข้อมูลจะต้องถูกแสดงในระดับ logical ไว้ด้วยวิธีการเดียวเท่านั้นคือ การใช้ตารางโดยอย่างน้อยจะต้องประกอบด้วยสิ่งต่อไปนี้

- ชื่อตาราง
- ชื่อคอลัมน์
- คอลัมน์ใดเป็น index หรือคีย์
- ชนิดของข้อมูลในแต่ละคอลัมน์
- ขอบเขตของค่าข้อมูลในแต่ละคอลัมน์ (domain)

วิเคราะห์การทดสอบ

ORACLE, INGRES, EMPRESS, INFORMIX มีคุณสมบัติตามกฎ เพราะระบบจัดการฐานข้อมูล ทั้ง 4 ตัวเก็บข้อมูลในระดับ logical เป็นตาราง

กฎข้อที่ 2 Guaranteed Access Rule

ผู้ใช้งานสามารถเข้าถึงข้อมูลทุกตัวในตารางได้ด้วยการระบุชื่อตาราง ค่าของ primary key และชื่อคอลัมน์ที่ต้องการ และถ้าไม่มีการเปลี่ยนแปลงข้อมูลในตารางแล้ว คำสั่งเดิมที่ใช้ในการเข้าถึงข้อมูลจะต้องได้ผลออกมาเพียงค่าเดียว และเหมือนเดิมทุกครั้ง

วิเคราะห์การทดสอบ

ORACLE, INGRES, INFORMIX, EMPRESS มีคุณสมบัติตามกฎ

เพราะสามารถเข้าถึงข้อมูลโดยการระบุชื่อตาราง, primary key และชื่อคอลัมน์ที่ต้องการได้ และการเข้าถึงตารางด้วยคำสั่งเดิมก็ครั้งก็ตาม ก็จะได้ผลออกมาเช่นเดิม (ถ้าตารางไม่ได้ถูกแก้ไข)

กฎข้อที่ 3 Systematic Treatment of Missing Information

มีการใช้ค่า null แทนในส่วนที่ไม่มีข้อมูล วิเคราะห์การทดสอบ

วิเคราะห์การทดสอบ

ORACLE, INGRES, INFORMIX, EMPRESS มีคุณสมบัติตามกฎ

เพราะระบบจัดการฐานข้อมูลทั้ง 4 ตัวนั้นจะแทนค่า null สำหรับข้อมูลที่ยังไม่มีในตาราง ระบบจัดการฐานข้อมูลทั้ง 4 ตัวจะมี default สำหรับ cell ที่ยังไม่มีข้อมูลใด ๆ ให้มีการใส่ค่า null ลงไป ยกเว้นเฉพาะคอลัมน์ที่มีการระบุเป็น not null ซึ่งคอลัมน์ดังกล่าวจะต้องมีค่าเสมอ เป็น null ไม่ได้ จึงถือว่าระบบจัดการฐานข้อมูล ทั้ง 4 ตัวมีคุณสมบัติตามกฎ

กฎข้อที่ 4 Dynamic On-Line Catalog must be based on the Relational Model

กฎข้อนี้เน้นคุณสมบัติที่สามารถให้ผู้ใช้ เรียกดูและแก้ไขโครงสร้างข้อมูลใน catalog ของระบบได้ด้วยภาษา และวิธีเดียวกับการเรียกดูข้อมูลของระบบ

วิเคราะห์การทดสอบ

ORACLE, INGRES, INFORMIX, EMPRESS มีคุณสมบัติตามกฎ

เพราะระบบจัดการฐานข้อมูลทั้ง 4 ตัวนั้นจะเก็บรายละเอียดเกี่ยวกับผู้ใช้ และตารางต่าง ๆ ในลักษณะของ relational และการเข้าถึง system catalog จะสามารถทำได้เหมือนกับการเข้าถึงข้อมูลทั่วไป โดยใช้ query language

กฎข้อที่ 5 Comprehensive Data Sublanguage Rule

ระบบจัดการฐานข้อมูล ควรมีภาษาย่อยอย่างน้อย 1 ภาษาที่สามารถทำการต่อไปนี้ได้

1. นิยามโครงสร้างระบบข้อมูล
2. นิยาม view
3. เปลี่ยนแปลงแก้ไขข้อมูล ทั้งด้วยวิธีการใช้คำสั่งแบบ interactive และด้วยการเขียนโปรแกรม
4. การควบคุม integrity ทั้ง entity integrity และ referential integrity
5. การให้สิทธิในการใช้งาน
6. ขอบเขตของ transaction คือการกำหนดว่าการทำงานกับข้อมูลในหน่วยใดหน่วยหนึ่งนั้นจะเริ่มต้น และสิ้นสุดตรงไหนทั้งนี้ก็เพราะหากระบบเกิด fail ระหว่างกลาง transaction ระบบจะได้สามารถ redo ได้

วิเคราะห์การทดสอบ

ORACLE, INGRES, INFORMIX, EMPRESS มีคุณสมบัติตามกฎ

ระบบจัดการฐานข้อมูลทั้ง 4 ตัวใช้ภาษา SQL ในการจัดการฐานข้อมูล แต่ INGRES มีภาษา QUEL อีกภาษาที่มีความสามารถเหมือน SQL

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กฎข้อที่ 6 View Updating Rule

ระบบจัดการฐานข้อมูล จะต้องตัดสินใจว่าผู้ใช้สามารถเพิ่มเติม หรือลบ tuple หรือแก้ไขข้อมูลในคอลัมน์ใดๆ โดยการทำการผ่าน view ได้หรือไม่ คือการกระทำดังกล่าวจะทำให้มีการสูญเสียความถูกต้องเกิดขึ้นกับตารางจริง หรือไม่ ซึ่งเป็นส่วนที่ ระบบจัดการฐานข้อมูลจะต้องควบคุมได้

วิเคราะห์การทดสอบ

ORACLE, INGRES, INFORMIX, EMPRESS มีคุณสมบัติตามกฎ

ORACLE มีคุณสมบัติตามกฎจะยอมให้ผู้ใช้สามารถที่จะทำการแทรกแก้ไข หรือลบผ่าน view ได้โดยมีกฎข้อบังคับคือ

1. ถ้า view ที่สร้างขึ้นถูกระบุโดย มี join operator set หรือ distinct operation group by clause ใน view นั้น จะไม่สามารถแทรก แก้ไข หรือ ลบได้
 2. ถ้า view ถูกระบุโดยมี with check option row ใน view นั้น จะไม่อนุญาตแก้ไข หรือลบได้
 3. ถ้า view ถูกระบุโดย not null คอลัมน์ (คอลัมน์ที่มีค่าเสมอ เป็น null ไม่ได้) ที่ไม่มี default clause ของ base table จะไม่สามารถแทรก row ได้โดยผ่านทาง view
 4. ถ้า view ถูกสร้างโดย expression จะไม่สามารถถูกแทรกหรือ แก้ไขโดยใช้ view ได้
- โดยรวมแล้วถือว่า ORACLE สามารถควบคุมการแก้ไขผ่านทาง view ให้ถูกต้องได้โดยจะไม่อนุญาตให้มีการแก้ไขตามกรณีค่ากล่าวมาแล้วข้างต้น จึงกล่าวได้ว่า ORACLE มีคุณสมบัติตามกฎนี้แม้จะไม่สมบูรณ์

INGRES มีคุณสมบัติตามกฎ โดยมีข้อยกเว้นดังนี้

1. ถ้า view นั้นถูกกำหนดขึ้นโดยมี with check option ข้อมูลใน view จะไม่อนุญาตให้มีการ update ได้
2. ถ้า view นั้นถูกระบุขึ้นโดยการคำนวณ หรือการใช้ aggregate function จะไม่สามารถ update ได้

3. ถ้า view นั้นถูกสร้างขึ้นจาก base table มากกว่า 1 ตาราง จะไม่สามารถ update ได้

INFORMIX มีคุณสมบัติตามกฎโดยจะมีข้อยกเว้นสำหรับการแก้ไขผ่าน view คือ

1. ไม่สามารถแก้ไขบน view ที่สร้างมาจาก base table มากกว่า 1 ตาราง
2. ไม่สามารถแก้ไขคอลัมน์ที่ source ไม่ได้เป็นชื่อคอลัมน์ธรรมดา เช่น คอลัมน์ที่เป็น set ของ function หรือเป็นการคำนวณ (computation)
3. ไม่สามารถทำการแก้ไขบนคอลัมน์ที่มี with check option เป็นตัวควบคุมอยู่

EMPRESS มีคุณสมบัติตามกฎ โดยการ update ผ่าน view นั้นจะต้องสร้างมาจากตาราง เพียงหนึ่งตารางเท่านั้น (มากกว่า 1 ตารางไม่สามารถ update ได้) และ ในกรณีที่คอลัมน์

ที่ถูก defined ถ้า not null หรือถูก create unique index นั้นห้ามใส่ค่า null ลงไป ในเงื่อนไข where clause จะยอมให้ update เมื่อ

1. เป็นการเปรียบเทียบ attribute กับค่าคงที่
2. เป็นการเปรียบเทียบระหว่าง 2 attribute ใน ตารางเดียวกัน
3. ในกรณีที่เท่ากันของค่า 2 attribute ที่อยู่ต่างตาราง ยกเว้น operation not, or และ view ถูกสร้างมาจาก base table มากกว่า 1 ตาราง ไม่สามารถถูก update ได้ ยกเว้น where clause operation “ = ” เท่านั้น ที่สามารถ update ได้

กฎข้อที่ 7 High-Level Insert, Update and Delete

ระบบจัดการฐานข้อมูล ควรจะมีภาษาที่สามารถให้ผู้ใช้เพิ่ม ลด หรือแก้ไขข้อมูลในหลาย ๆ แถวหรือหลาย ๆ คอลัมน์ ได้ด้วยการออกคำสั่งเพียงคำสั่งเดียว

วิเคราะห์การทดสอบ

ORACLE, INGRES, INFORMIX, EMPRESS มีคุณสมบัติตามกฎ
ระบบจัดการฐานข้อมูลทั้ง 4 ตัวใช้ภาษา SQL และ 4GL ในการเพิ่ม, ลด หรือแก้ไขข้อมูล แต่ INGRES มีภาษา QUEL อีกภาษาที่มีความสามารถตามกฎ

กฎข้อที่ 8 Physical Data Independence

ผู้ใช้ไม่จำเป็นต้องรับรู้เกี่ยวกับการจัดเก็บข้อมูลจริง รวมทั้งการเปลี่ยนแปลงวิธีการจัดเก็บเหล่านี้จะไม่กระทบกระเทือนกับการใช้งานที่ทำไว้แต่ดั้งเดิม

วิเคราะห์การทดสอบ

ORACLE, INGRES, INFORMIX, EMPRESS มีคุณสมบัติตามกฎ
เพราะว่าการเก็บทาง physical ของระบบจัดการฐานข้อมูล ทั้ง 4 ตัวจะถูกแยกเก็บออกจากโปรแกรมของผู้ใช้โดยเด็ดขาด ดังนั้นวิธีการจัดเก็บข้อมูลจะไม่มีผลต่อโปรแกรมประยุกต์และการทำงานของผู้ใช้เลย

กฎข้อที่ 9 Logical Data Independence

การเปลี่ยนแปลงข้อมูลในระดับ logical (การเปลี่ยนแปลงโครงสร้างข้อมูล เช่น การเพิ่มเติมคอลัมน์เข้าในตาราง หรือการสร้างตารางใหม่) ไม่มีผลต่อคำสั่งและโปรแกรมที่เขียนขึ้นไว้ก่อน

วิเคราะห์การทดสอบ

ORACLE, INGRES, INFORMIX, EMPRESS มีคุณสมบัติตามกฎ

เพราะว่าการเพิ่ม คอลัมน์ ในตาราง หรือการเพิ่มตารางใหม่ จะไม่มีผลต่อผู้ใช้ ยกเว้น การตัดคอลัมน์, การ drop ตาราง หรือการแยกตารางของผู้ใช้เท่านั้น จึงจะมีผลกระทบต่อ โปรแกรมประยุกต์ และผู้ใช้

กฎข้อที่ 10 Integrity Independence

ระบบจัดการฐานข้อมูล ควรจะสามารถจัดเก็บข้อมูลเกี่ยวกับการควบคุม integrity ไว้ใน catalog ของระบบโดยให้เป็นอิสระจากโปรแกรม

วิเคราะห์การทดสอบ

ORACLE, INGRES, INFORMIX, EMPRESS มีคุณสมบัติตามกฎ

เพราะสามารถเขียนกฎต่าง ๆ เพื่อใช้ในการควบคุมความถูกต้องของข้อมูล โดยการกำหนด referential integrity และการเขียน database procedure ที่นำไปเก็บไว้ใน system catalog ช่วยให้ผู้ใช้เขียนโปรแกรม โดยไม่ต้องเขียนส่วนตรวจสอบความถูกต้อง ซึ่งวิธีนี้ EMPRESS ไม่รองรับความสามารถด้านนี้

กฎข้อที่ 11 Distribution Independence

ผู้ใช้ระบบไม่ต้องสนใจว่า ข้อมูลจะถูกจัดเก็บอย่างไร แม้จะถูกโยกย้ายไปยัง คอมพิวเตอร์ อื่น ๆ ที่เชื่อมโยงไว้ก็ตาม(หลักการของ distributed data base system)

วิเคราะห์การทดสอบ

ORACLE, INGRES, INFORMIX, EMPRESS มีคุณสมบัติตามกฎ

เพราะว่าระบบจัดการฐานข้อมูลทั้ง 4 ตัวนั้นอยู่บนพื้นฐานที่ว่า ผู้ใช้ไม่ต้องสนใจว่าข้อมูล ถูกเก็บอยู่ที่ไหน บนเครื่องใดในเครือข่าย แม้จะถูกโยกย้ายไปยังเครื่องอื่น ๆ ในระบบก็ไม่มีผล ต่อการทำงานของผู้ใช้

กฎข้อที่ 12 Nonsubversion Rule

Integrity ต่าง ๆ ที่ถูกระบุไว้ใน catalog จะต้องสามารถใช้ควบคุมความถูกต้องของข้อมูล ในระบบได้ตลอดเวลา ไม่ว่าผู้จะใช้จะเข้าถึงข้อมูลด้วยเครื่องมือใด

วิเคราะห์การทดสอบ

ORACLE, INGRES, INFORMIX, EMPRESS มีคุณสมบัติตามกฎ

เพราะ integrity ที่ระบุไว้สามารถจะควบคุมความถูกต้องของข้อมูลในระบบได้ตลอดเวลา ไม่ว่าผู้จะใช้จะใช้เครื่องมือใดในการเข้าถึงข้อมูล

8.2 กฎ 16 ข้อของการออกแบบระบบจัดการฐานข้อมูลของ Codd

E.F. Codd กล่าวถึงกฎ 16 ข้อในการออกแบบระบบจัดการฐานข้อมูลเชิงสัมพันธ์ซึ่งโดยทั่วไปแนวความคิดทางทฤษฎีของ relational model ค่อนข้างชัดเจนอยู่แล้ว แต่ผู้ค้าของระบบจัดการฐานข้อมูลส่วนใหญ่ยังมีข้อผิดพลาดอีกมากมายที่เกิดขึ้นในส่วนการ การประยุกต์ใช้งาน (ออกแบบและสร้างให้เป็นไปตามที่ต้องการ) ดังนั้นกฎ 16 ข้อของการออกแบบระบบจัดการฐานข้อมูลเชิงสัมพันธ์ที่จะกล่าวถึงนี้จะเน้นในด้านการ การประยุกต์ใช้งาน ดังต่อไปนี้

กฎข้อที่ 1 Non-Violation for any Fundamental Law of Mathematics

(ไม่ควรเกิดกรณีที่ขัดแย้งกับกฎทางคณิตศาสตร์)

ผลจากการทำงานของระบบจัดการฐานข้อมูลเชิงสัมพันธ์ และภาษาที่ใช้ซึ่งจะต้องเป็น ภาษา relational (ภาษา SQL) ไม่ควรขัดแย้งกับกฎทางคณิตศาสตร์ โดยพิจารณาจากกฎการ สลับที่ได้ของลอจิก เช่นในกรณีที่เงื่อนไขเป็น where Y and X แล้วหากมีการเรียกค้นหาข้อมูลที่มี เงื่อนไข where X and Y แล้วผลลัพธ์ที่ได้จะต้องได้ผลลัพธ์ที่เหมือนกัน ซึ่งเป็นไปตามกฎการ สลับที่ได้ของคณิตศาสตร์

ผลการทดสอบ

ผลที่ได้จากการทดสอบ query โดยใช้เงื่อนไข where clause ที่ต่างกันของ ระบบจัดการ ฐานข้อมูล ทั้ง 4 ตัวนั้นต่างก็ให้ผลลัพธ์ที่เหมือนกัน แต่เวลาที่ใช้แตกต่างกัน จึงจัดได้ว่าระบบ จัดการฐานข้อมูล ทั้ง 4 ตัวมีคุณสมบัติข้อนี้

กฎข้อที่ 2 Under-the -Covers Representation and Access

(ระบบจัดการฐานข้อมูลจะมีหน้าที่จัดการเกี่ยวกับการจัดเก็บและการเข้าถึงข้อมูลระดับล่าง)

สิ่งที่เกี่ยวข้องกับการเข้าถึงข้อมูลระดับล่าง ไม่ว่าจะเป็นวิธีการเก็บรักษาข้อมูลและวิธีการเข้า ถึงข้อมูลระดับล่างเป็นหน้าที่ของระบบจัดการฐานข้อมูลโดยที่ผู้ใช้ไม่จำเป็นต้องรับรู้ เช่นในกรณี ที่ query ที่ได้ผลลัพธ์เหมือนกัน แต่ใช้ query และวิธีการเลือกเส้นทางที่ต่างกันควรจะได้ response time เดียวกัน

ผลการทดสอบ

เมื่อทดสอบระบบจัดการฐานข้อมูลทั้ง 4 ตัว โดยใช้วิธีการ join และ subquery นั้นจะใช้ เวลาในการประมวลผลต่างกัน ซึ่งผลต่างนั้นจะเพิ่มมากขึ้นตามจำนวนของผู้ใช้ แสดงว่าระบบ จัดการฐานข้อมูล ทั้ง 4 ตัวไม่มีคุณสมบัติในข้อนี้

กฎข้อที่ 3 Sharp Boundary

(ระบบจัดการฐานข้อมูลจะต้องแยกระหว่างส่วนที่จัดการเกี่ยวกับประสิทธิภาพ และส่วนที่จัดการเกี่ยวกับ semantic)

ระบบจัดการฐานข้อมูลจะต้องแยกระหว่างส่วนที่จัดการเกี่ยวกับประสิทธิภาพ เช่น index และส่วนที่จัดการเกี่ยวกับ semantic เช่น primary key , foreign key ฯลฯ เช่นการใช้ primary integrity เป็นตัวกำหนด primary key แทน

ผลการทดสอบ

ORACLE, INGRES, INFORMIX, EMPRESS มีคุณสมบัติตามกฎ

เพราะระบบจัดการฐานข้อมูลทั้ง 4 ตัวมีการแยกส่วนที่จัดการเกี่ยวกับประสิทธิภาพ และความถูกต้องออกจากกัน

กฎข้อที่ 4 Concurrency Independence

(ระบบจัดการฐานข้อมูลจะควบคุมการใช้ข้อมูลร่วมกันของหลายผู้ใช้ในเวลาเดียวกัน)

ระบบจัดการฐานข้อมูลจะจัดการเรียกใช้และเข้าถึงข้อมูลในฐานข้อมูลในสถานะแบบ multi-user กล่าวคือ มีผู้ใช้มากกว่า 1 คน และผู้ใช้เหล่านั้นสามารถแก้ไขข้อมูลในตารางเดียวกันในเวลาเดียวกันได้

ผลการทดสอบ

ระบบจัดการฐานข้อมูล ทั้ง 4 ตัวมีคุณสมบัติข้อนี้ โดยใช้กลไกการ lock ในการควบคุมการใช้ฐานข้อมูลร่วมกัน โดยประกันได้ว่าในขณะที่ transaction ใด transaction หนึ่ง แก้ไขข้อมูล โดยอยู่จะไม่มี transaction อื่นใดสามารถเข้าถึง ข้อมูลเหล่านั้นได้ไม่ว่าชนิดของการเข้าถึงนั้นจะเป็น read หรือ write ก็ตามจนกว่า transaction ที่แก้ไข ข้อมูลนั้นจะ commit แล้วเพื่อรักษาสถานะความถูกต้องโดยไม่ยอมให้ข้อมูลและการประมวลผลกลุ่มของงานต้องขัดแย้งกันและทำให้ข้อมูลในฐานข้อมูลไม่ถูกต้องโดยที่ผู้ใช้ไม่จำเป็นต้องรับรู้

กฎข้อที่ 5 Protection Against Unauthorized Long-Term Locking

(ระบบจัดการฐานข้อมูลต้องมีการจัดการป้องกันการ lock ในระยะเวลาที่นานเกินไป)

ระบบจัดการฐานข้อมูลต้องมีการกำหนดระยะเวลา ที่ผู้ใช้แต่ละคนจะสามารถ lock ทรัพยากรข้อมูล

ผลการทดสอบ

INGRES, EMPRESS, INFORMIX มีคุณสมบัติข้อนี้โดยการใช้ parameter “time out” ซึ่งจะกำหนดระยะเวลาที่ผู้ใช้สามารถเรียกใช้และเข้าถึงข้อมูลได้

ORACLE มีคุณสมบัติข้อนี้โดยการใช้ TPO (Transaction processing option)

กฎข้อที่ 6 Orthogonality in DBMS Design

(แนวทางการออกแบบระบบจัดการฐานข้อมูลต้องอยู่ในแนวทางเดียวกัน)

ผลการทดสอบ

ORACLE, INGRES, INFORMIX, EMPRESS มีคุณสมบัติตามกฎ

กฎข้อที่ 7 Domain-Based Index

(ความสามารถในการสร้าง index ที่ domain)

ผลการทดสอบ

ระบบจัดการฐานข้อมูล ทั้ง 4 ตัวไม่มีคุณสมบัติข้อนี้ เนื่องจากระบบจัดการฐานข้อมูลในปัจจุบันยังใช้การสร้าง index บน attribute

กฎข้อที่ 8 Database Statistics

(ระบบจัดการฐานข้อมูลต้องมีข้อมูลทางสถิติของฐานข้อมูล)

ควรมีการเก็บข้อมูลทางสถิติของข้อมูลที่เกิดขึ้นในฐานข้อมูลและเก็บข้อมูลทางสถิติเหล่านี้ใน catalog ของระบบเพื่อช่วยในการทำ query optimization (การเลือกวิธีในการ retrieve และจัดการเก็บข้อมูลในฐานข้อมูลให้มีประสิทธิภาพมากที่สุด)

ORACLE มีคุณสมบัติข้อนี้ โดยจะทำ query optimization ด้วยวิธีการ rule based approach ในการพิจารณาเส้นทางในการเข้าถึง และ cost-based approach พิจารณา cost ของแต่ละ execution plan ของ SQL statement โดยจะพิจารณาจากข้อมูลทางสถิติ

INGRES มีคุณสมบัติข้อนี้ โดยมีการเก็บจำนวน row, unique value ในคอลัมน์ null, จำนวน duplicate value ในคอลัมน์ รวมถึงการใช้ histogram ในการคำนวณหาเส้นทาง

INFORMIX มีคุณสมบัติข้อนี้ โดยใช้ข้อมูลเกี่ยวกับ index ของตาราง และ System catalog

EMPRESS มีคุณสมบัติข้อนี้ โดยใช้ข้อมูลเกี่ยวกับ Index ของตาราง และ Catalog statistics

กฎข้อที่ 9 Interrogation of Statistics

(ต้องสามารถเรียกข้อมูลทางสถิติได้)

ผู้บริหารฐานข้อมูล จะต้องสามารถเรียกดูข้อมูลทางสถิติที่เก็บไว้ใน catalog ของระบบได้

ผลการทดสอบ

ระบบจัดการฐานข้อมูล ทั้ง 4 ตัวมีคุณสมบัติข้อนี้

กฎข้อที่ 10 Changing Storage Representation and Access Options

(ระบบจัดการฐานข้อมูลสามารถเปลี่ยนแปลงการเก็บข้อมูลและวิธีการเข้าถึงข้อมูลได้)

ระบบจัดการฐานข้อมูลสามารถเปลี่ยนแปลงการเก็บข้อมูลและวิธีการเข้าถึงข้อมูลได้โดยที่โปรแกรมประยุกต์ ที่วิ่งอยู่ไม่จำเป็นต้องรับรู้หรือเปลี่ยนแปลงโปรแกรมประยุกต์ตาม

ผลการทดสอบ

ระบบจัดการฐานข้อมูล ทั้ง 4 ตัวมีคุณสมบัติข้อนี้

ORACLE มี 3 แบบ คือ Cluster, Hash และ B-tree

INGRES มี 4 แบบ คือ Heap, Isam, Hash และ B-tree

INFORMIX มี 3 แบบ คือ Cluster, Hash และ B-tree

EMPRESS มี 1 แบบ คือ B+-tree

กฎข้อที่ 11 Automatic Protection in Case of Malfunctions

(ถ้า transaction ใด transaction หนึ่งผิดพลาดแล้ว ฐานข้อมูลต้องไม่เสียหาย)

ฐานข้อมูลควรอยู่ใน consistence state ไม่ว่า transaction จะผิดพลาดหรือถูกต้องสมบูรณ์

ผลการทดสอบ

ระบบจัดการฐานข้อมูล ทั้ง 4 ตัวมีคุณสมบัติข้อนี้

กฎข้อที่ 12 Automatic Recovery in Case of Malfunctions

(ถ้าเกิดความเสียหายกับ transaction แล้ว ระบบจัดการฐานข้อมูลต้องมีความสามารถที่จะ recovery transaction คืนได้)

ผลการทดสอบ

ระบบจัดการฐานข้อมูล ทั้ง 4 ตัวมีคุณสมบัติข้อนี้

กฎข้อที่ 13 Atomic Execution of Relational Command

(คำสั่ง relational ต้องเป็น atomic ในแง่ของผู้ใช้ คือ ถ้าผิดพลาดต้องผิดพลาดทั้ง command)

ผลการทดสอบ

ระบบจัดการฐานข้อมูล ทั้ง 4 ตัวมีคุณสมบัติข้อนี้

กฎข้อที่ 14 Atomic Archiving**(มีที่สำหรับสำรองการเก็บข้อมูลหรือส่วนควบคุม)****ผลการทดสอบ**

ระบบจัดการฐานข้อมูล ทั้ง 4 ตัวมีคุณสมบัติข้อนี้

ORACLE ใช้กระบวนการที่เรียกว่า ARCH จะเป็นการเก็บกลุ่มของ redo log file เก่า ๆ ลงใน archived redo log file เพื่อเพิ่มที่ว่างให้กับ online redo log file

INGRES มีการทำ static back up และ dynamic back up ที่เรียกว่า archiver process

INFORMIX มีการทำ mirror ในการ back up

EMPRESS มีการ back up ฐานข้อมูลทั้งแบบ online back up และ offline back up โดยใช้ฟังก์ชัน emparch ในการทำ back up

กฎข้อที่ 15 Avoiding Cartesian product

ไม่ควรมีการกำเนิด cartesian product ลงในฐานข้อมูล และไม่ควรให้มีผลลัพธ์สุดท้ายเป็น cartesian product เด็ดขาด

ผลการทดสอบ

ระบบจัดการฐานข้อมูล ทั้ง 4 ตัวขาดคุณสมบัติข้อนี้

กฎข้อที่ 16 Responsibility for decryption and encryption**(decryption และ encryption ควรเป็นเรื่องของไต่ตารางที่ผู้ใช้ไม่ต้องทำ)****ผลการทดสอบ**

ระบบจัดการฐานข้อมูล ทั้ง 4 ตัวมีคุณสมบัติข้อนี้

บทที่ 9

สรุปและวิจารณ์

จากการทดสอบและวิเคราะห์ระบบการจัดการฐานข้อมูลเชิงสัมพันธ์ ที่มีอยู่ในปัจจุบันคือ ORACLE, INGRES, INFORMIX และ EMPRESS ถึงลักษณะต่างๆ ทั้งทางกายภาพ การใช้ งาน และการจัดเก็บข้อมูล ซึ่งข้อมูลที่รวบรวมมานั้น ได้มาจากการศึกษาคู่มือของผลิตภัณฑ์ พูดยุคกับทางบริษัทที่เป็นตัวแทน จำหน่ายผลิตภัณฑ์ ทั้ง 4 โดยการศึกษา ตามหัวข้อต่างๆ ดังได้กล่าวมาแล้ว

และเนื่องจากว่า ระบบจัดการฐานข้อมูลที่ได้นำมาทดสอบและวิเคราะห์นั้น ได้ทำงานอยู่บน platform ที่ต่างกัน ซึ่งมีประสิทธิภาพต่างกันค่อนข้างมาก ผลที่ได้จึงไม่ได้นำมาแสดงบนหัวข้อที่ทดสอบหัวข้อเดียวกัน และกราฟเดียวกัน เพื่อเปรียบเทียบให้เห็นอย่างชัดเจน เพราะเห็นว่าเป็นการไม่ยุติธรรมต่อบาง products ที่นำมาทำการทดสอบ จึงทำให้ไม่สามารถใช้เวลาที่ได้จากการทดสอบมาเป็นเกณฑ์ในการตัดสินประสิทธิภาพในเชิงเปรียบเทียบกันระหว่างระบบจัดการฐานข้อมูลทั้ง 4 ได้

และด้วยเหตุผลบางประการ จึงทำให้ไม่สามารถทำการทดสอบ ระบบจัดการฐานข้อมูลเชิงสัมพันธ์ ORACLE ได้ ทำให้ไม่มีผลของการทดสอบของ ORACLE

ผลที่ได้จากการทดสอบ และ การวิเคราะห์ ระบบจัดการฐานข้อมูล สามารถสรุปแบ่งได้เป็น 5 ข้อดังนี้

Enterprise wide application : ระบบจะต้องมีความสามารถรองรับฐานข้อมูล ที่มีขนาดใหญ่ได้ สนับสนุนการจัดการฐานข้อมูลแบบกระจาย สนับสนุนการ optimize และสามารถรองรับโปรโตคอลได้หลายแบบ ระบบจัดการฐานข้อมูลทั้ง 4 ตัวนั้น เป็นระบบจัดการฐานข้อมูลแบบกระจาย ชนิด heterogeneous ทั้งหมด รองรับการทำ two phase commit และการ update ฐานข้อมูลแบบกระจาย ความสามารถในการรับข้อมูลอยู่ในระดับสูงกว่าล้าน records ขึ้นไปถึงล้าน รวมถึงการ optimize ทั้งหมดที่เป็นแบบ cost base โดยที่ ORACLE นั้นจะสามารถทำ optimize แบบ rule base ได้และ INGRES มีการเก็บสถิติของข้อมูล ในรูปแบบของ histogram ด้านความสามารถรองรับโปรโตคอลนั้น ระบบจัดการฐานข้อมูลทุกตัวสามารถรองรับโปรโตคอลแบบ TCP/IP ได้ ดังแสดงในตารางที่ 9.1 และ ตารางที่ 9.2

Departmental application : ความสามารถในการเข้าถึงได้ง่ายของผลิตภัณฑ์ติดตั้ง ใช้งานง่าย ไม่ต้องปรับแต่งมาก มีเอกสารประกอบที่ดี การทำงานของระบบย่อย ๆ ของผลิตภัณฑ์

ORACLE , INGRES , INFORMIX และ EMPRESS มีรายละเอียดของเอกสารมากไปหาน้อยตามลำดับ ซึ่งรวมไปถึงความสามารถที่จะปรับแต่งระบบจัดการฐานข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

คู่มือการใช้ของ ORACLE นั้นเน้นที่ความเข้าใจและเขียนอย่างดี โดยให้รายละเอียดเกี่ยวกับแนวความคิดและมีตัวอย่างจำนวนมากด้วย นอกจากนี้คู่มือยังสามารถแสดงให้เห็นเลขที่สามารถนำไปใช้ในสถานการณ์ที่แตกต่างกันได้อย่างไร

คู่มือการใช้ของ INGRES เหมาะที่จะเรียกว่า เอกสารอ้างอิงจะดีกว่า เพราะรายละเอียดทางด้านเทคนิคนั้นมีน้อยมาก ไม่ได้อธิบายถึงแนวคิดในการทำงาน มีตัวอย่างเพียง 2 ถึง 3 ตัวอย่าง ไม่มีคู่มือการใช้งาน รวมทั้งภาพรวมของสถาปัตยกรรม เพื่อที่จะเข้าใจการทำงาน จึงทำให้จำเป็นต้องอ่านคู่มือหลาย ๆ เล่ม หรือขอคำปรึกษาจากผู้รู้ INGRES ที่ชำนาญ

คู่มือการใช้ของ INFORMIX นั้นมีการนำเสนอรายละเอียดเกี่ยวกับแนวความคิดและโครงสร้างของระบบจัดการฐานข้อมูลพอสมควร โดยมุ่งเน้นไปในทางการใช้งานเป็นส่วนมาก

คู่มือการใช้ของ EMPRESS มีรายละเอียดทางด้านเทคนิคน้อยมาก โดยเฉพาะทางด้านสถาปัตยกรรม ไม่ได้อธิบายถึงแนวคิดในการทำงาน แต่จะมุ่งเน้นด้านการใช้งาน ซึ่งมีตัวอย่างที่พอเพียงกับการใช้งาน

Decision-Support application: สนับสนุนการตัดสินใจ ความสามารถในการ load ข้อมูลได้อย่างรวดเร็ว การตั้งคำถามเฉพาะกิจ (ad hoc queries) ความสามารถทางด้าน front end (แสดงในตารางที่ 9.3)

ระบบจัดการฐานข้อมูลทั้งหมด มีความสามารถด้านการตั้งคำถามเฉพาะกิจเท่าเทียมกัน โดยการใช้ภาษา SQL ผ่านทาง SQL editor ของระบบ หรือผ่านทาง tools ต่าง ๆ เช่น 4GL , form ได้เหมือนกัน

ORACLE และ INGRES มีความสามารถทำ stored procedure สำหรับคำถามที่มีการเรียกใช้เป็นประจำ ORACLE, INGRES และ EMPRESS สามารถกำหนด user defined function คือสามารถสร้างคำสั่งขึ้นใหม่ แล้วนำมารวมกับ SQL มาตรฐานได้ ส่วน front - end นั้น ORACLE, INGRES และ INFORMIX ใช้ menu เข้ามาช่วยสำหรับการใช้คำสั่ง SQL เช่น สร้างตาราง, เรียกดูตารางได้ ในขณะที่ EMPRESS ใช้ interactive mode และคำสั่งพิเศษ

Mission-Critical application : ความมีเสถียรภาพ และไม่มีข้อผิดพลาด (bug free) ความสามารถในการแบ็คอัพขณะที่ทำงานอยู่ (on-line backup) การจัดการด้านความปลอดภัย (แสดงในตารางที่ 9.4)

ระบบจัดการฐานข้อมูลทั้ง 4 ตัว มีการทำ online backup , การรักษาความปลอดภัยของข้อมูลในระดับ databases , relations , views , fields และการใช้วิธีการของ audit trail

ORACLE , INGRES และ EMPERSS สามารถรองรับการทำ trigger เพื่อตรวจสอบเหตุการณ์และแก้ไขความผิดพลาดต่าง ๆ ที่อาจเกิดขึ้นได้ ส่วน INFORMIX มีความสามารถในการทำ mirror ฐานข้อมูลได้

Transaction Processing : ความสามารถด้านการทำรายการ ความสามารถด้านการทำ concurrency กับความถูกต้องของฐานข้อมูล (consistency) ความถูกต้องในการอ้างอิง (referential integrity)

ORACLE , INFORMIX และ EMPRESS สามารถรองรับการ lock ในระดับต่ำที่สุดได้ ในระดับ row(record) ,INGRES ในระดับ page ระบบการจัดการฐานข้อมูลทั้ง 4 ตัวมีความสามารถในการควบคุมความสอดคล้องของข้อมูล ด้วยการทำ referential integrity และ ทำการ recovery ข้อมูล แบบอัตโนมัติ, การ roll forward และ roll back เมื่อฐานข้อมูล หรือ transaction ได้รับความเสียหาย หรือเกิดความผิดพลาดขึ้น แสดงในตารางที่ 9.5

1



ตารางที่ 9.1 Enterprise wide application

ENTERPRISE WIDE APPLICATION				
COMPARISON CRITERIA:	ORACLE V7	INGRES V6.4	INFORMIX V6	EMPRESS V6
ARCHITECTURE:				
Client Server:	MultiThreaded, MultiServer	MultiThreaded, MultiServer, Shared Memory	MultiServer, Shared Memory	MultiServer, Shared Memory
Server to Server:	Yes	Yes	Yes	Yes
SPECIFICATIONS:				
Max. Field Size :	256 Characters	2008 Bytes	32K Characters	2 Million Characters
Max. Record Size :	64,770 Characters	2008 Bytes	32K Characters	2 Million Characters
Max. Field/Record (Row Size):	254	300	Unlimited	9,999 per Table
Max. Records/Relation (Table):	Hardware Dependant	Unlimited	4.276 Million (Table is < 36.8K Bytes)	2 Billion per Table
Max. Index/Relation (Table):	Hardware Dependant	300	100	9,999
Max. Relations/Database:	Hardware Dependant	Unlimited	Unlimited	9,999
Max Number of Open Tables:	Hardware Dependant	Hardware Dependant	Hardware Dependant	Hardware Dependant
Max. Number of Users:	Hardware Dependant	No Logical Limit	1000	Hardware Dependant
TWO PHASE COMMIT:	Yes	Yes	Yes	Yes
DISTRIBUTED UPDATES:	Yes	Yes	Yes	Yes
OPTIMIZER:	Cost Base and Rule Base	Cost Base	Cost Base	Cost Base
OPENNESS (to other Products):	Offers SQL, Connect to DB2, SQL / DS, RMS & Turbo Image	DB2, RDB, Ulrix / SQL, and others	DB2, SQL/DS, IMS, IDMS ADABAS, TOTAL, VSAM & SAM	Offers Oracle gateway

ตารางที่ 9.2 Departmental application

ARCHITECTURE					
COMPARISON CRITERIA:	ORACLE V7	INGRES V6.4	INFORMIX V5	EMPRESS V6	
SUPPORTED DATA TYPES:					
Null or Missing	Yes	Yes	Yes	Yes	
Integer	Yes	Yes	Yes	Yes	
Decimal	Yes	Yes	Yes	Yes	
Floating Point	Yes	Yes	Yes	Yes	
Character	Yes	Yes	Yes	Yes	
Date	Yes	Yes	Yes	Yes	
Time	Yes	Yes	Yes	Yes	
BLOB	Yes	Yes (varies)	Yes (2 GB)	Yes (2 GB)	
User Defined	Yes	Yes	No	Yes	
DATA DICTIONARY:	Yes	Yes	Yes	Yes	
INDICES:					
Index Storage Options	B+-Tree, Clustered	B-Tree, ISAM, Hash	B-Tree, Clustered	B+-Tree	
Updated Transparently	Yes	Yes	Yes	Yes	
Index on Expressions	No	Yes	No	No	
DATA STORAGE OPTIONS:	Hash, B+-tree, Clustered	B-Tree, ISAM, Hash, Heap	Culstered, Hash	B+-Tree	

ตารางที่ 9.3 Decision-Support application

DECISION SUPPORT APPLICATION				
COMPARISON CRITERIA:	ORACLE V7	INGRES V6.4	INFORMIX V5	EMPRESS V6
SQL:				
ANSI Standard:	Yes	Yes	Yes	Yes
Embedded SQL:	Yes	Yes	Yes	Yes
Dynamic SQL:	Yes	Yes	Yes	No
Precompiled SQL:	Yes	Yes	No	Yes
USER DEFINED FUNCTION:	Yes (Forms Only)	Yes	No	Yes
STORED PROCEDURES:	Yes - forms Only	Yes	No	No
TOOLS:				
Interactive SQL	Yes	Yes	Yes	Yes
Report Writer	Yes	Yes	Yes	Yes
Rorums Utility	Yes	Yes	Yes	Yes
4GL	Yes	Yes	Yes	Yes
LANGUAGE SUPPORT:				
Ada	Yes	Yes	Yes	Yes
C	Yes	Yes	Yes	Yes
FORTRAN	Yes	Yes	Yes	Yes

ตารางที่ 9.4 Mission-Critical application

MISSION CRITICAL APPLICATION					
COMPARISON CRITERIA:	ORACLE V7	INGRES V6.4	INFORMIX V5	EMPRESS V8	
SHADOWING / MIRRORING:	No	Yes	Yes	Simulated views or OS	
JOURNALING:	Yes	Yes	Yes	Yes	
SECURITY:					
Placed on Databases	Yes	Yes	Yes	Yes	
Placed on Relations	Yes	Yes	Yes	Yes	
Placed on Views	Yes	Yes	Yes	Yes	
Placed on Fields	Yes	Yes	Yes	Yes	
Audit Trail Support	Yes	Yes	Yes	Yes	
BACKUPS:					
On-line	Yes	Yes	Yes	Yes	
Roll Back	Yes	Yes	Yes	Yes	
TRIGGERS:	Yes, forms only	Yes (Via RULES)	No	Yes, via EMPRESS 4GL	

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 9.5 Transaction Processing

TRANSACTION PROCESSING				
COMPARISON CRITERIA:	ORACLE V7	INGRES V6.4	INFORMIX V5	EMPRESS V6
LOCKING LEVEL:	Table, Row	DB, Table, Page, Control	Table, Row	Table, Group, Null, and record
REFERENTIAL INTEGRITY:	Yes	Yes	Yes	Yes
RECOVERY:				
Automatic	Yes	Yes	Yes	Yes
Roll Forward	Yes	Yes	Yes	Yes
RELATIONAL FEATURES:				
Views	Yes	Yes	Yes	Yes
Tree Structured Query	Yes	Yes	No	Yes
Primary Keys	Yes	Yes	Yes	Yes
Outer Join	Yes	Yes	Yes	Yes
Nested Query	Yes	Yes	Yes	Yes
Null Value	Yes	Yes	Yes	Yes

กิตติกรรมประกาศ

ขอขอบคุณ บริษัท ไมโครเอกซ์ , บริษัท Oracle System Thailand ที่ให้คำปรึกษา
และให้ข้อมูลผลิตภัณฑ์

ขอขอบคุณ สถาบันคอมพิวเตอร์ มหาวิทยาลัยรามคำแหงที่เอื้อเฟื้อเครื่องมือ
ในการทดสอบ INGRES DBMS และข้อมูลในการทดสอบฐานข้อมูล

ขอขอบคุณ บริษัท WAVE DEVELOPMENT LIMITED ที่เอื้อเฟื้อเครื่องมือ
ในการทดสอบ INFORMIX DBMS

ขอขอบคุณ บริษัท IBC SYMPHONY COMPANY LIMITED ที่เอื้อเฟื้อเครื่องมือ
ในการทดสอบ ORACLE

ขอขอบคุณ บริษัท Uniphice และ บริษัท Premier Soft ที่ให้คำปรึกษาและข้อมูล
ผลิตภัณฑ์ รวมทั้งเอื้อเฟื้อ เครื่องมือในการทดสอบ EMPRESS DBMS

ขอขอบคุณ ดร. เอื้อน ปิ่นเงิน ที่ให้คำปรึกษา และแนะนำเกี่ยวกับปริญญานิพนธ์นี้

ขอขอบคุณ เพื่อนๆ ทุกคนที่ช่วยให้คำปรึกษา , เสนอแนะ และให้ความร่วมมือ
ในการทำปริญญานิพนธ์ ฉบับนี้ ให้สำเร็จลุล่วงไปด้วยดี

หนังสืออ้างอิง

1. Valduriez , Patrice , “ Analysis and comparison fo relational database management system ”, Addison Wesley,1991
2. Henry F.Korth , Abraham silbersechatz , “ Database System concept ”, Mcgraw Hill,1991
3. C.J. Date , “ A guide to INGRES ”, Addison Wesley, 1987
4. C.J. Date , “ An Introduction to database System ”, Addison Wesley, 1990
5. C.J. Date , “ A Guide to the SQL Standard ” , Addison Wesley, 1989
6. Carl Malanud “ INGRES : tools for building An Information Architecture ” , Van Nostrand Reinhold, 1989
7. Michael Bronzite “ Intorduction to Oracle ”, Mcgraw Hill, 1989
8. Jack L Hursch , Carolyn J Hursch “ Ingres SQL developer 's guide ”, Windcrest Books, 1992
9. เอก โชติชาลวงค์ , “ SQL SERVER ” , วารสารขอบึงคอมพิวเตอร์ มีนาคม 2537 , หน้า 225-270