รูปแบบการเขียนโปรแกรมภาษายุคที่ 3
ในสภาพแวดล้อมฐานข้อมูลเชิงสัมพันธ์

Third Generation Language Style in Relational Database Environment

โดย

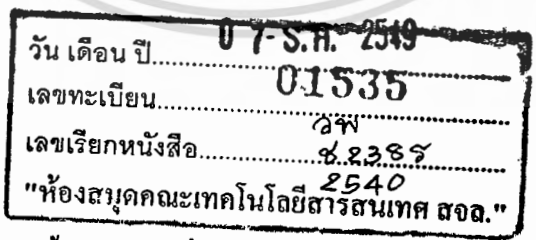
นายชลัท เขตประเสริฐกุล

รหัส 38626054

H001535

อาจารย์ที่ปรึกษา

รศ.ดร.ศุภมิตร จิตตะยโศธร



รายงานนี้เป็นส่วนหนึ่งของวิชาโครงการพัฒนาระบบงาน
หลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
ภาคเรียนที่ 1 ปีการศึกษา 2540
คณะเทคโนโลยีสารสนเทศ
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ชื่อหัวข้อ รูปแบบการเขียนโปรแกรมภาษายุคที่ 3ในสภาพแวดล้อมฐานข้อมูลเชิงสัมพันธ์

นักศึกษา นายชลัท เขตประเสริฐกุล

อาจารย์ที่ปรึกษา รศ.ดร.ศุภมิตร จิตตะยโศธร

ระดับการศึกษา วิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ

แขนวงวิชา วิทยาการสารสนเทศ

พ.ศ. 2540

บทคัดย่อ

ปัจจุบันในการประมวลผลข้อมูลในระบบฐานข้อมูลเชิงสัมพันธ์ จะมีการพัฒนาระบบฐานข้อมูล และภาษาหรือคำสั่งที่ใช้ในการประมวลข้อมูลเพื่อให้สามารถใช้งานได้ง่าย และมีประสิทธิภาพมากขึ้นมาก แต่อย่างไรก็ตามในการพัฒนาระบบงานเพื่อประมวลข้อมูลยังคงต้องใช้ภาษาโปรแกรมในยุคที่ 3 มาใช้งานโดยนำภาษา หรือคำสั่งที่พัฒนาขึ้นมาใหม่ ซึ่งเป็นส่วนที่ใช้ประมวลผลข้อมูลในฐานข้อมูลไปฝังในตัวโปรแกรม (ตย. การใช้คำสั่งเอสคิวแอล (SQL) แต่ในการเขียนโปรแกรมเพื่อประมวลผลข้อมูลในลักษณะดังเดิม คือการอ่านข้อมูลมาครั้งละหนึ่งเรคอร์ด (Record-at-a-time) ก็ยังสามารถนำมาใช้งานกับระบบฐานข้อมูลแบบใหม่ได้เช่นกัน

ฉะนั้นในการศึกษาครั้งนี้จะทำการศึกษารูปแบบการเขียนโปรแกรมภาษายุคที่ 3 เพื่อประมวลผลข้อมูลในฐานข้อมูลเชิงสัมพันธ์ โดยพิจารณาว่าการเขียนโปรแกรมแบบใหม่ที่ใช้คำสั่งเอสคิวแอล (SQL) และแบบดังเดิมที่อ่านข้อมูลมาครั้งละหนึ่งเรคอร์ด (Record-at-a-time) จะมีผลกระทบอย่างไรกับฐานข้อมูลเชิงสัมพันธ์ในด้านของระยะเวลาในการประมวลผลข้อมูล และความถูกต้องของข้อมูลในระบบฐานข้อมูลเมื่อมีการใช้งานในสภาวะพร้อม ๆ กัน

Title Third Generation Language Style in Relational Database Environment

Student Mr.Chalat Khetprasertkul

Advisor Assoc.Prof.Dr.Suphamit Chittayasothorn

Level of Student Master of Science in Information Technology

Major Information Science

Year 1997

ABSTRACT

Although nowadays the database system and the language programming of data processing in relational database which are developed have more efficiency and easier to use, the development of working system is still very important. To embedded the new created language or command in the third generation language which use to operate data in database in the program (Ex. Using SQL command). Also the old programming style for operate data which will read the data one record-at-a-time is still working with the new database system.

In this case, we will study the programming style by using third generation language to data processing in relational database and study the effect of the time that uses for operate the data and the accuracy of the data in relation database when the program that uses SQL command and the program that read one record-at-a-time are run

กิตติกรรมประกาศ

ผลงานการศึกษานี้ได้ประสบผลสำเร็จลุล่วงด้วยดี ด้วยแรงสนับสนุนจากบุคคลดังต่อไปนี้ ดังนี้กระผมจึงใคร่ขอขอบพระคุณ

1. บิดา มารดา เป็นผู้มีพระคุณมาก เป็นผู้ให้กำเนิดและเลี้ยงดู อบรมสั่งสอนให้เป็นคนดี คอยเป็นกำลังใจให้ขยันศึกษา พร้อมทั้งช่วยออกค่าใช้จ่ายต่าง ๆ ในการเล่าเรียนศึกษา
2. รศ.ดร.ศุภมิตร จิตตะยโศธร เป็นอาจารย์ที่ปรึกษาโครงงาน ที่กรุณาให้ปรึกษาแนะนำการศึกษา ร่วมทั้งช่วยแก้ปัญหาต่าง ๆ ที่เกิดขึ้น ในขณะที่ทำการพัฒนาโครงงานการศึกษา
3. คณะเทคโนโลยีสารสนเทศ ได้ให้ความกรุณาเอื้อเพื้อในด้านอุปกรณ์
4. พี่ เพื่อน และน้องทุกคนที่คอยให้ความช่วยเหลือ

นายชลัท เขตประเสริฐกุล

สารบัญ

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญภาพ

สารบัญภาพ(ต่อ)

รูปที่ หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

เนื่องในปัจจุบันองค์กรต่าง ๆ ได้นำระบบคอมพิวเตอร์เข้ามาช่วยในการบริหารงาน ฉะนั้นจึงได้มีการพัฒนาโปรแกรมคอมพิวเตอร์ เพื่อสนับสนุนการทำงานเหล่านั้น และการทำงานส่วนใหญ่ก็จะเป็นการเก็บข้อมูล แล้วนำข้อมูลนั้นมาใช้งานในอนาคต ระบบฐานข้อมูลจึงเข้ามามีส่วนความสำคัญในการจัดเก็บข้อมูล และภาษาที่ใช้ในการเขียนโปรแกรมส่วนมากก็เป็น ภาษาโคบอล ปาสคาล วิชวลเบสิก ซี เป็นต้น จะเห็นได้ภาษาดังกล่าวได้ถูกเรียกว่าภาษายุคที่ 3 ดังนั้นเพื่อให้การประมวลผลข้อมูลสามารถใช้งานได้อย่างมีประสิทธิภาพ และสามารถจัดเก็บข้อมูลได้อย่างถูกต้อง จึงควรที่จะศึกษารูปแบบการเขียนโปรแกรมที่เหมาะสม กับระบบฐานข้อมูลแต่ละประเภทที่ใช้งานอยู่ ไม่เช่นนั้นแล้ว อาจทำให้เกิดปัญหาเมื่อมีการนำโปรแกรมนั้นมาใช้งานจริง

1.2 วัตถุประสงค์ของการศึกษา

1. เพื่อศึกษารูปแบบการเขียนโปรแกรมภาษายุคที่ 3 สำหรับการประมวลผลข้อมูลในระบบฐานข้อมูลที่มีสภาพแวดล้อมการทำงานเชิงแบบสัมพันธ์
2. เพื่อช่วยให้ผู้พัฒนาระบบงานสารสนเทศ ได้เลือกรูปแบบและวิธีการเขียนโปรแกรมให้เหมาะสมกับ ระบบฐานข้อมูลแบบต่าง ๆ
3. เพื่อช่วยเพิ่มประสิทธิภาพการการทำงานของโปรแกรม ในการประมวลข้อมูลในระบบฐานข้อมูลได้มีประสิทธิภาพ และมีความถูกต้องมากขึ้น

1.3 ขอบเขตในการศึกษา

การเขียนโปรแกรมโดยใช้ภาษายุคที่ 3 ในสภาพแวดล้อมฐานข้อมูลเชิงสัมพันธ์ ได้ทำการศึกษาโดยใช้ภาษาปาสคาล หรือเดลไฟ (Delphi) ในการเขียนโปรแกรมเพื่อติดต่อกับระบบฐานข้อมูล โดยมีระบบฐานข้อมูลแบบต่าง ๆ ที่ทำการศึกษา ดังนี้

- ฐานข้อมูลพาราด็อกซ์ (Paradox Database)
- ฐานข้อมูลไมโครซอฟท์แอ็คเซส (Microsoft Access Database)
- ระบบจัดการฐานข้อมูลไมโครซอฟท์เอสคิวแอลเซิร์ฟเวอร์

(MSSQL DBMS : Microsoft SQL Server Database Management System)

- ระบบจัดการฐานข้อมูลออราเคิล

(ORACLE DBMS : Oracle Database Management System)

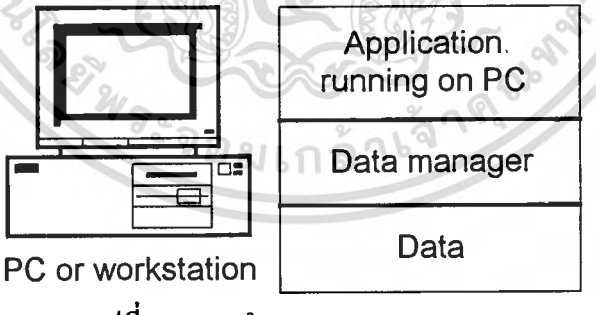
โดยในการศึกษาครั้งนี้ จะทำการศึกษาทั้งในลักษณะที่ระบบฐานข้อมูล อยู่บนเครื่องเดียวกันกับแอพพลิเคชั่นที่ทำงาน และอยู่คนละเครื่องกันเครื่องกัน มีรูปแบบการประมวลผลข้อมูล กับฐานข้อมูลที่เป็นแบบครั้งละหนึ่งเรคอร์ด (Record-at-a-time) และใช้ เอสคิวแอล (SQL) โดยมีการพิจารณาวิธีการจัดการข้อมูลของฐานข้อมูล และลักษณะความแตกต่างกันในด้านของความเร็วในการประมวลผล ในขณะมีการทำงานคนเดียว (Single User)

1.4 รูปแบบการทำงานและระบบคอมพิวเตอร์ที่ใช้ในการศึกษา

ระบบคอมพิวเตอร์ที่ใช้ในการศึกษาเพื่อทำการประมวลผล เมื่อพิจารณาจากการทำงานของแอพพลิเคชั่น และฐานข้อมูลสามารถแบ่งการทำงานออกเป็น 3 รูปแบบดังนี้

1) Standalone

แอพพลิเคชั่นที่ทำการประมวลผลข้อมูล และฐานข้อมูลเชิงสัมพันธ์อยู่บนเครื่องเดียวกัน ซึ่งในรูปแบบนี้จะมีการประมวลผลทั้งหมดอยู่อยู่บนเครื่องคอมพิวเตอร์เครื่องเดียวกัน โดยการศึกษาครั้งนี้การทำงานทั้งหมดจะทำงานอยู่บนคอมพิวเตอร์ลูกค่าย ดังรูปที่ 1.1

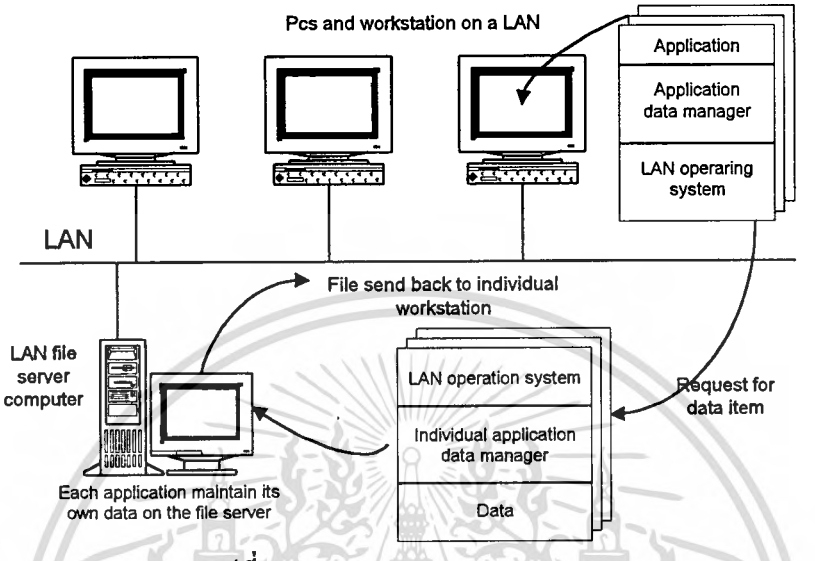


รูปที่ 1.1 การทำงานแบบ Standalone

2) File Server

แอพพลิเคชั่นที่ทำการประมวลผลข้อมูล และฐานข้อมูลเชิงสัมพันธ์จะอยู่ต่างเครื่องกัน สามารถติดต่อสื่อสารกันผ่านเครือข่ายสื่อสารข้อมูล โดยฐานข้อมูลจะถูกเก็บไว้ที่เครื่องคอมพิวเตอร์แม่ค่าย (Server) เมื่อมีการประมวลผลข้อมูล เครื่องคอมพิวเตอร์แม่ค่ายก็จะส่งข้อมูลทั้งหมดไปยังเครื่องคอมพิวเตอร์ลูกค่าย เพื่อให้เครื่องคอมพิวเตอร์ลูกค่ายประมวลผลข้อมูล จะเห็น

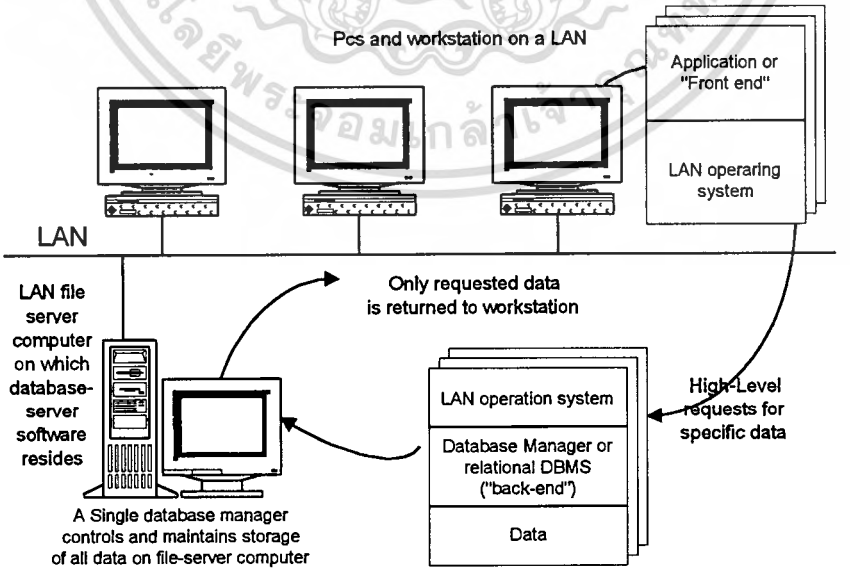
ได้ว่า รูปแบบนี้เครื่องคอมพิวเตอร์แม่ค่ายไม่ได้มีหน้าที่ประมวลผลข้อมูลเลย มีหน้าที่ส่งข้อมูลอย่างเดียว ดังรูปที่ 1.2



รูปที่ 1.2 File-Server architecture

3) Client/Server

แอพพลิเคชั่นที่ทำการประมวลผลข้อมูล และฐานข้อมูลเชิงสัมพันธ์จะอยู่ต่างเครื่องกัน สามารถติดต่อสื่อสารกันผ่านเครือข่ายสื่อสารข้อมูล โดยฐานข้อมูลจะถูกเก็บไว้ที่เครื่องคอมพิวเตอร์แม่ค่าย (Server) เมื่อมีการประมวลผลข้อมูล เครื่องคอมพิวเตอร์ลูกค่ายก็จะส่งคำสั่งมา



รูปที่ 1.3 Client-Server architecture

ยังเครื่องคอมพิวเตอร์แม่ค่าย เครื่องคอมพิวเตอร์แม่ค่ายก็จะทำการประมวลผลข้อมูลตามคำสั่งนั้น แล้วส่งผลลัพธ์ที่ได้กลับไปเครื่องคอมพิวเตอร์ลูกค่าย จะเห็นว่ารูปแบบนี้เครื่องคอมพิวเตอร์แม่ค่ายจะมีหน้าที่ประมวลผลข้อมูลด้วย ดังรูปที่ 1.3

1.5 เครื่องมือที่ใช้ในการศึกษา

1) เครื่องคอมพิวเตอร์ที่เป็นตัวแม่ค่าย (Server)
   - หน่วยประมวลผลกลาง (CPU) ความเร็ว 133 MHz มี 2 CPU
   - หน่วยความจำหลัก 64 Mbytes
   - ระบบปฏิบัติการ Microsoft Windows NT 4.0
   - ซอฟท์แวร์จัดการฐานข้อมูล Oracle Server Version 7.3.3
   - ซอฟท์แวร์จัดการฐานข้อมูล MSSQL Server 6.5
2) เครื่องคอมพิวเตอร์ลูกค่าย (Client)
   - หน่วยประมวลผลกลาง (CPU) มีความเร็ว 133 MHz
   - หน่วยความจำหลัก 16 Mbytes
   - ระบบปฏิบัติการ Microsoft Windows 95
   - ซอฟท์แวร์จัดการฐานข้อมูล Personal Oracle Version 7.3.3
   - โปรแกรมแปลภาษา Delphi 3.0

1.6 รายละเอียดในแต่ละบท

สำหรับรายละเอียดของเนื้อหาในบทถัดไปจะมีรายละเอียดดังนี้

ในบทที่ 2 จะเป็นรายละเอียดเกี่ยวกับทฤษฎีที่เกี่ยวในระบบฐานข้อมูลเชิงสัมพันธ์ นอกจากนี้ยังรวมถึงคุณสมบัติที่ทำให้ฐานข้อมูลที่ ทำหน้าที่ในฐานข้อมูลมีลักษณะเป็นฐานข้อมูลเชิงสัมพันธ์

ในบทที่ 3 จะเป็นรายละเอียดที่เกี่ยวข้องกับการหน้าที่ของระบบจัดการฐานข้อมูล ในการควบคุมภาวะการทำงานพร้อมกัน ปัญหาในการประมวลผลข้อมูลพร้อมกัน ตลอดจนเทคนิคที่ใช้

ในบทที่ 4 จะเป็นการศึกษาเพื่อทดลองเวลาที่ใช้ในการประมวลผลในระบบฐานข้อมูลเชิงสัมพันธ์ ใช้การสอบถามข้อมูลแบบต่าง ๆ

ในบทที่ 5 เป็นการศึกษาเพื่อทดลองการใช้คำสั่งล็อก และการทำงานแบบเป็นรายการเปลี่ยนแปลง (Transaction) ตลอดจนการใช้คำสั่งการบันทึกของมูลลงแถบแม่เหล็ก

ในบทที่ 6 เป็นการสรุปผลการศึกษา

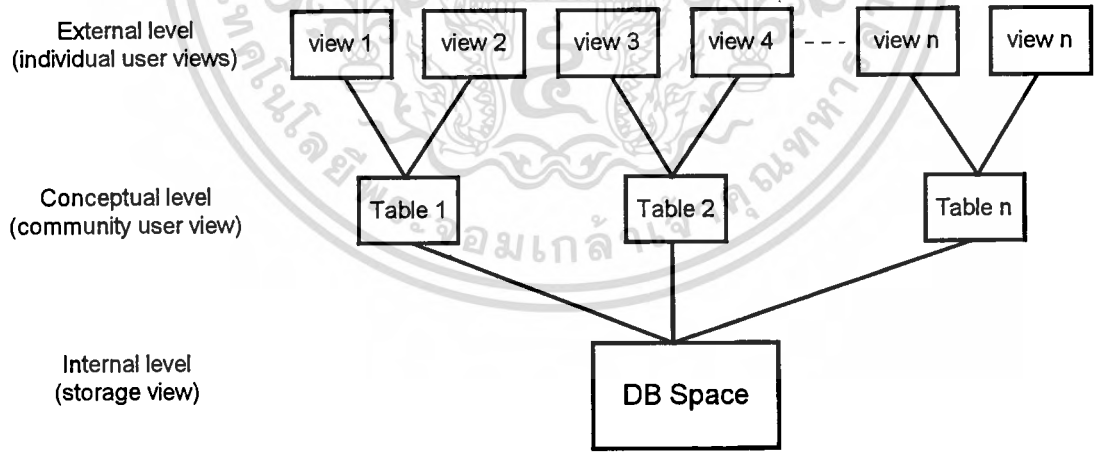
บทที่ 2
ระบบฐานข้อมูลเชิงสัมพันธ์

ฐานข้อมูล คือ โครงสร้างสารสนเทศ (Information) ที่เกิดขึ้นจากการนำข้อมูลต่าง ๆ นำมารวมกัน และข้อมูลเหล่านั้นมีความสัมพันธ์กัน

2.1 สถาปัตยกรรมของระบบฐานข้อมูล

สถาปัตยกรรมของระบบฐานข้อมูล ได้ออกแบบมาเพื่อให้เหมาะสมกับการใช้ฐานข้อมูลแก่ผู้ใช้ โดยที่ผู้ใช้ไม่ต้องกังวลกับรายละเอียดต่าง ๆ ในการเก็บข้อมูล ไม่ต้องทราบรายละเอียดอื่น ๆ ที่ตนไม่ได้ใช้งาน มีการแบ่งระดับของข้อมูลออกเป็น 3 ระดับดังนี้

- นิยามข้อมูลระดับภายใน (Internal Schema)
- นิยามข้อมูลระดับหลักการ (Conceptual Schema)
- นิยามข้อมูลระดับภายนอก (External Schema)



รูปที่ 2.1 แสดงสถาปัตยกรรมของระบบฐานข้อมูล

2.1.1 นิยามข้อมูลระดับภายใน (Internal Schema)

นิยามข้อมูลระดับภายใน เป็นระดับข้อมูลที่อยู่ต่ำสุด โดยมีการกำหนดลักษณะโครงสร้างข้อมูลที่ถูกจัดเก็บจริงในอุปกรณ์เก็บข้อมูล เช่น ถ้าความสัมพันธ์ระหว่างข้อมูลอยู่ในรูปแบบเชิง

สัมพันธ์ซึ่งในระดับหลักการ และระดับภายนอกจะแสดงอยู่ในรูปแบบของตาราง แต่เมื่อข้อมูลของตารางนั้น ๆ ถูกจัดเก็บจริงในฮาร์ดดิสก์ (Harddisk) ข้อมูลอาจจะเก็บด้วย รูปแบบของบี-ทรี (B-Tree) ซี-ไอแซม (C-ISAM) หรือลิงค์ลิสต์ (Link List) ก็ได้ ซึ่งมีการจัดเก็บข้อมูลในระดับนี้ระบบจัดการฐานข้อมูล (DBMS) จะจัดการให้โดยที่ผู้ใช้ได้ไม่ต้องจัดการเอง

2.1.2 นิยามข้อมูลระดับหลักการ (Conceptual Schema)

นิยามข้อมูลระดับหลักการ เป็นระดับข้อมูลที่อยู่ถัด โดยมีการกำหนดลักษณะรูปแบบข้อมูล ขนาดของข้อมูลและความสัมพันธ์ของข้อมูลทั้งหมดในระบบงาน นั่นคือไม่ว่าฐานข้อมูลจะมีความสัมพันธ์ระหว่างข้อมูลอยู่ในรูปแบบใด ก็ตามจะต้องกำหนดการแทนรูปแบบของข้อมูลในนิยามข้อมูลระดับหลักการนี้ เช่น ถ้าความสัมพันธ์ระหว่างข้อมูลอยู่ในรูปแบบเชิงสัมพันธ์ (Relational Model) ในระดับนี้จะแสดงชื่อตาราง ชื่อคอลัมน์ ชนิดข้อมูลแต่ละคอลัมน์ตลอดจนชื่อของคีย์หลัก (Primary Key) และชื่อคีย์นอก (Foreign Key) เป็นต้น

2.1.3 นิยามข้อมูลระดับภายนอก (External Schema)

นิยามข้อมูลระดับภายนอก เป็นระดับข้อมูลที่อยู่บนสุด โดยมีการกำหนดโครงสร้างข้อมูลให้ผู้ใช้ ใช้งานโดยที่ผู้ใช้งานแต่ละคนอาจมีสิทธิ์ที่จะเห็นข้อมูลได้ไม่เหมือนกัน ถึงแม้ว่าจะเป็นข้อมูลนั้นจะเป็นชุดเดียวกัน ในกำหนดสิทธิการใช้งานข้อมูลอาจต้องทำผ่านวิว (View) โดยมีผู้ดูแล และควบคุมฐานข้อมูล หรือผู้ที่มีอำนาจกำหนดสิทธิของตารางนั้น ๆ กำหนดขอบเขตการใช้ข้อมูลเหล่านั้น โดยบางคนอาจมีสิทธิ์ใช้ข้อมูลได้บางแถว (Row) หรือบางคอลัมน์ (Column) ของตารางเท่านั้น หรืออาจเป็นนำข้อมูลจากหลาย ๆ ตารางมารวมกันเป็นต้น

2.2 ระบบจัดการฐานข้อมูล (Database Management System)

ระบบจัดการฐานข้อมูล (DBMS) เป็นซอฟท์แวร์ที่ทำหน้าที่ในการจัดการและดูแลการใช้งานผู้ใช้งานข้อมูลในฐานข้อมูล โดยมีหน้าที่ดังนี้

- การกำหนดโครงสร้างข้อมูล (Data definition) สามารถที่จัดการกับโครงสร้างของข้อมูลทั้ง 3 ระดับ (External Schema, Concept Schema, Internal Schema) และกำหนดโครงสร้างของข้อมูลให้ผู้ใช้แต่ละคนมองเห็น
- การจัดการเกี่ยวกับข้อมูล (Data manipulation) มีหน้าจัดการการเข้าถึงข้อมูล โดยผู้ใช้งานสามารถสอบถาม ค้นหา แก้ไข ลบ ข้อมูลที่มีอยู่ในฐานข้อมูล หรือเพิ่มข้อมูลใหม่เข้าไปยังฐานข้อมูลได้ด้วย

- การทำหน้าพจนานุกรมข้อมูล (Data dictionary) ทำหน้าเก็บข้อมูลการทำงานของระบบโครงสร้างของข้อมูล เพื่อสามารถให้ผู้ใช้งานสอบถามโครงสร้างของข้อมูลที่อยู่ในระบบของตน ได้อย่างถูกต้องและครบถ้วน
- การควบคุมความปลอดภัยของข้อมูล (Data security) มีหน้าที่ป้องกันไม่ให้ผู้ที่ไม่ได้รับอนุญาตเข้ามาเห็นหรือแก้ไขข้อมูลในส่วนที่ต้องการปกป้อง
- การควบคุมภาวะพร้อมกัน (Concurrency control) เป็นหน้าที่ในการคุมควบการใช้ข้อมูล ในสภาพที่มีผู้ใช้งานพร้อม ๆ กันหลายคน และควบคุมลำดับการทำงานให้เป็นไปอย่างถูกต้อง

2.3 รูปแบบเชิงสัมพันธ์ (Relational Model)

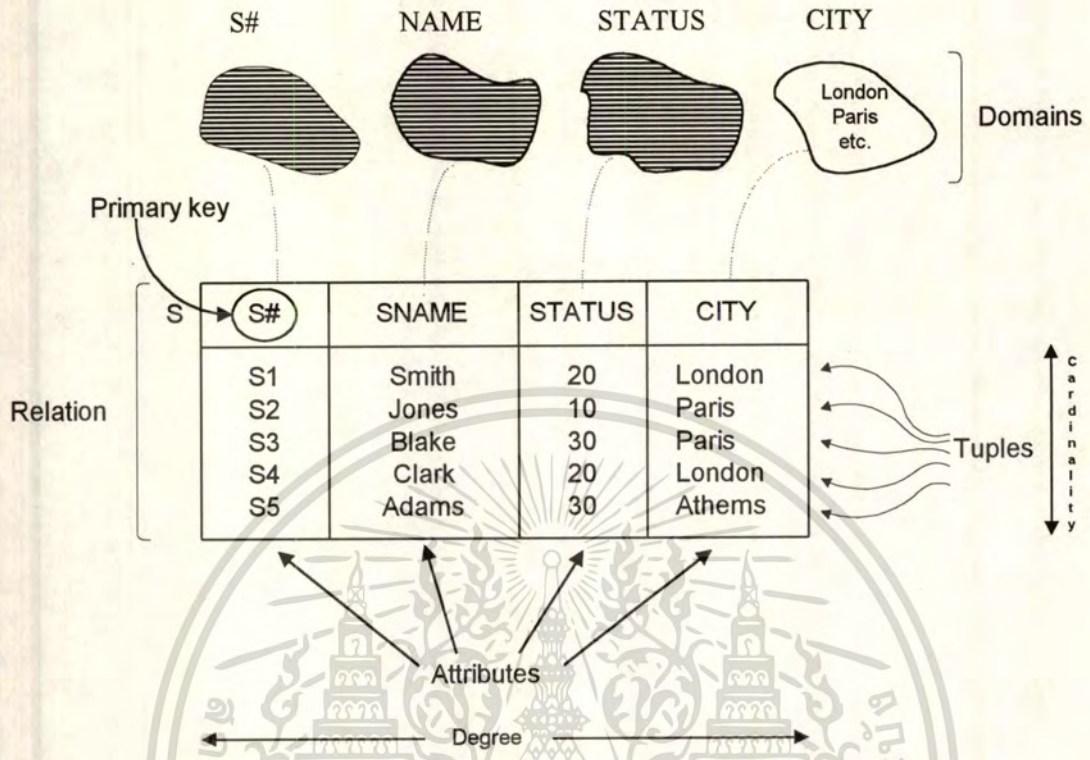
ฐานข้อมูลที่มีความสัมพันธ์ระหว่างข้อมูลอยู่ในรูปแบบเชิงสัมพันธ์ประกอบด้วย 3 ส่วน ได้แก่ โครงสร้างข้อมูล (Data Structure) หรือ ออปเจ็ค (Object) , ความถูกต้องของข้อมูล (Data Integrity) และ การจัดการข้อมูล (Data Manipulation)

2.4 โครงสร้างข้อมูล (Data Structure)

โครงสร้างข้อมูลจะอยู่ในรูปของรีเลชั่น และในฐานข้อมูลเชิงสัมพันธ์จะประกอบไปด้วย รีเลชั่น ซึ่งอาจมีมากกว่าหนึ่งรีเลชั่นก็ได้ โดยที่ในแต่รีเลชั่นนั้นต้องมีความสัมพันธ์กับรีเลชั่นอื่น สามารแสดงให้เห็นในรูปที่ 2.2

เปรียบเทียบศัพท์ในเชิงวิชาการกับคำศัพท์ที่ใช้ทั่ว ๆ ไปโดยมีความหมายเหมือนกันได้ดังนี้

ศัพท์เชิงวิชาการ	ศัพท์ที่ใช้ทั่ว ๆ ไป
รีเลชั่น (Relation)	ตาราง (Table)
ทัพเพิล (Tuple)	แถว (Row) หรือ เรคอร์ด (Record)
คาร์ดินอลิตี้ (Cardinality)	จำนวนของแถว (Row)
แอททริบิวต์ (Attribute)	คอลัมน์ (Column) หรือ ฟิลด์ (Field)
ดีกรี (Degree)	จำนวนคอลัมน์ในแต่ละตาราง
คีย์หลัก (Primary Key)	ค่าที่ไม่มีการซ้ำกัน (Unique Identifier)
โดเมน (Domain)	กลุ่มของค่าที่ถูกต้องและเป็นไปได้



รูปที่ 2.2 แสดงรายละเอียดของรีเลชั่น S

2.4.1 รีเลชั่น (Relation)

รีเลชั่นจะเป็นที่เก็บเซ็ตของค่าโดเมน D1, D2, ... , Dn ซึ่งจะขึ้นอยู่กับจำนวนแอททริบิวต์ (Attribute) ของแต่ละรีเลชั่น โดยที่รีเลชั่นจะประกอบไปด้วย 2 ส่วนคือ ส่วนหัว (Heading) และส่วนตัว (Body)

- ส่วนหัว (Heading) จะประกอบด้วยเซตจำกัดของแอททริบิวต์ เขียนความสัมพันธ์ได้คือ { (A1:D1), (A2:D2), ... , (An:Dn) } ดังนั้นค่าของแอททริบิวต์ Aj ใด ๆ จะมีขอบเขตของข้อมูลสัมพันธ์กับโดเมน Dj ใด ๆ โดยที่ j = 1, 2, ... , n เมื่อ n คือจำนวนแอททริบิวต์ของแต่ละรีเลชั่น หรือดีกรีของรีเลชั่น
- ส่วนตัว (Body) จะประกอบด้วยจำนวนของทัพเพิล (Tuple) ซึ่งจะมีจำนวนทัพเพิลเปลี่ยนไปตามระยะเวลาใด ๆ โดยที่ทัพเพิลจะประกอบด้วยเซ็ตของแอททริบิวต์และเวลลูว์ (Va.lue) สามารถเขียนความสัมพันธ์ดังนี้ {(A1:vi1), (A2:vi2), ... , (An:vin)} โดยที่ i = 1, 2, ... , m

เมื่อ m คือจำนวนของทัพเพิลในแต่ละรีเลชั่น หรือคาร์ดินอลิตี้ (Cardinality) และ n คือจำนวนแอททริบิวต์ของแต่ละรีเลชั่นหรือดีกรีของรีเลชั่น

2.4.2 คุณสมบัติของรีเลชั่น

คุณสมบัติของรีเลชั่นมีอยู่ 4 ข้อดังต่อไปนี้

- แต่ละทัพเพิลของรีเลชั่นเดียวจะบรรจุข้อมูลเพียงค่าเดียว (There are no duplicate tuples)

คุณสมบัติข้อนี้เป็นจริงเนื่องจากส่วนตัว (Body) ของรีเลชั่นแสดงความสัมพันธ์อยู่ในรูปของเซ็ตซึ่งเป็นทฤษฎีทางคณิตศาสตร์ และโดยนิยามของทฤษฎีเซ็ตนั้นจะไม่รวมข้อมูลที่มีการซ้ำซ้อนกัน ดังนั้นคำว่ารีเลชั่น หรือตาราง สำหรับฐานข้อมูลเชิงสัมพันธ์จะมีความหมายรวมถึงค่าของข้อมูลในแถวของตารางจะต้องไม่ซ้ำกัน นั่นแสดงถึงต้องมีคีย์หลัก (Primary Key) เพื่อควบคุมไม่ให้ข้อมูลทั้งหมดในแต่ละทัพเพิลซ้ำกัน

- การเรียงลำดับของทัพเพิลถือว่าไม่มีความสำคัญ (Tuple are unorder)

คุณสมบัติข้อนี้เป็นจริงเนื่องจากส่วนตัว (Body) ของรีเลชั่น แสดงความสัมพันธ์อยู่ในรูปของเซ็ตซึ่งเป็นทฤษฎีทางคณิตศาสตร์ และสมาชิกในเซ็ตนั้นจะไม่มีลำดับ ดังนั้นคำว่ารีเลชั่น หรือตารางสำหรับฐานข้อมูลเชิงสัมพันธ์จะมีความหมายรวมถึง คุณสมบัติการมีลำดับของข้อมูลในแถวของตารางจากบนลงล่างจะไม่มีผลต่อความสัมพันธ์ระหว่างข้อมูลในตาราง

- การเรียงลำดับของแอททริบิวต์ไม่ถือว่ามีความสำคัญ (Attribute are unorder)

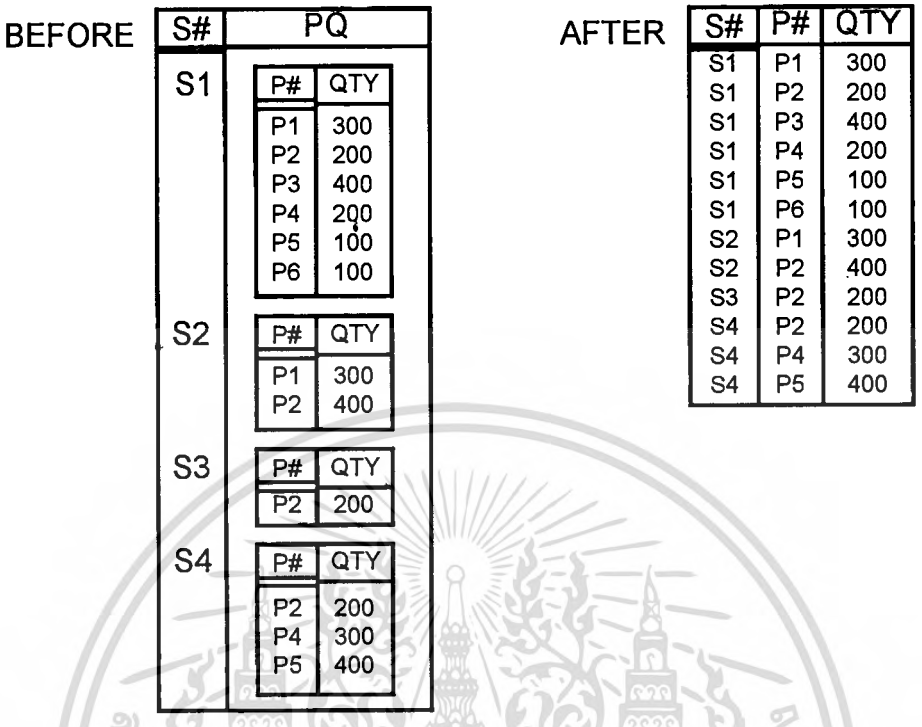
คุณสมบัติข้อนี้เป็นจริงเนื่องจากส่วนหัว (Heading) ของรีเลชั่นแสดงความสัมพันธ์อยู่ในรูปของเซ็ตซึ่งเป็นทฤษฎีทางคณิตศาสตร์และสมาชิกในเซ็ตนั้นจะไม่มีลำดับ ดังนั้นเมื่อระบุชื่อของแอททริบิวต์ใด ๆ ที่มากกว่า 1 ตัวเหมือนกันแต่ลำดับของแอททริบิวต์ต่างกัน ก็จะได้ข้อมูลตัวเดียวกัน ดังนั้นคำว่ารีเลชั่น หรือตารางสำหรับฐานข้อมูลเชิงสัมพันธ์จะมีความหมายรวามถึง คุณสมบัติการมีลำดับของคอลัมน์ในตารางจากซ้ายไปขวา จะไม่มีผลต่อความสัมพันธ์ระหว่างข้อมูลในตาราง

- ข้อมูลแต่ละแอททริบิวต์จะต้องแตกต่างกัน (All attribute values are atomic)

คุณสมบัติข้อนี้ก็คือไม่ให้มีข้อมูลที่มีลักษณะเป็นกลุ่มซ้ำ (Repeating group) เก็บอยู่ในรีเลชั่น ดังนั้นต้องการทำนอร์มอลไลเซชั่น (Normalization) ระดับที่ 1 (1NF) ก่อนเพื่อให้ค่าของแอททริบิวต์ทุกตัวเป็นค่าเดี่ยว (Atomic value) ดังรูปที่ 2.3

2.5 ความคงสภาพของข้อมูล (Data integrity)

รูปแบบเชิงสัมพันธ์ (Relational model) ได้มีการกำหนดความคงสภาพของข้อมูล โดยเป็นการจัดการเกี่ยวกับของเขตค่าของโดเมน และเงื่อนไขการกระทำ ของคีย์แข่งขัน (Candidate key) และคีย์นอก (Foreign key) ที่ควรจะมีในทุก ๆ ฐานข้อมูลดังนี้



รูปที่ 2.3 ตัวอย่างการทำนอร์มอลไลเซชั่น (Normalization)

2.5.1 คีย์แข่งขัน (Candidate key)

เมื่อกำหนดให้ K ซึ่งแทนแอททริบิวต์ 1 ตัวหรือ กลุ่มของแอททริบิวต์ในรีเลชั่น R แล้ว K จะเป็นคีย์แข่งขัน (Candidate key) ของรีเลชั่น R ได้ก็ต่อเมื่อ K มีคุณสมบัติ 2 อย่างคือ

- คุณสมบัติความเป็นหนึ่ง (Uniqueness)
  คือไม่มีทัพเพิลใด ๆ บนรีเลชั่น R ที่มีค่าเหมือนกับ K
- คุณสมบัติการลดน้อยที่สุด (Irreductbility)
  คือ ถ้า K แทนด้วยกลุ่มของแอททริบิวต์ แล้วจะไม่มีแอททริบิวต์ใด ๆ ที่เป็นสมาชิกใน K แล้วทำให้คุณสมบัติความเป็นหนึ่ง (Uniqueness) ของ K สูญเสียไป

เมื่อได้ค่าจำกัดความของคีย์แข่งขันแล้วจะได้ว่า คีย์หลักคือคีย์แข่งขันตัวหนึ่งที่ถูกเลือกขึ้นมาเพื่อความเหมาะสมในการใช้งานมากที่สุด ส่วนคีย์แข่งขันที่เหลือจะเรียกว่าคีย์รอง

2.5.2 คีย์นอก (Foreign key)

คีย์นอก คือแอททริบิวต์ หรือกลุ่มของแอททริบิวต์ที่อยู่ในรีเลชั่น R2 ที่เซ็ตข้อมูลของแอททริบิวต์หรือกลุ่มของแอททริบิวต์นั้นจะเป็นสับเซต (Subset) ของเซตข้อมูลของคีย์แข่งขัน ในรีเล

ชั่น R1 นั่นก็คือค่าข้อมูลของคีย์นอกจะแทนการอ้างอิงหรือเชื่อมโยง ไปยังทัพเพิลที่มีค่าข้อมูลคีย์แข่งขันตรงกับค่าข้อมูลคีย์นอกดังนั้นสามารถแสดงการเชื่อมโยงดังความสัมพันธ์

$$R2 \rightarrow R2$$

ดังนั้นถ้ามีรีเลชั่น n รีเลชั่นแล้วรีเลชั่นที่ n จะเชื่อมโยงกับรีเลชั่นที่ n-1 สามารถแสดงการเชื่อมโยงดังความสัมพันธ์

$$Rn \rightarrow R(n-1) \rightarrow R(n-2) \rightarrow \ldots \rightarrow R2 \rightarrow R1$$

จากความหมายและความสัมพันธ์ต่าง ๆ ในรีเลชั่นเหล่านี้จึงต้องมีกฎควบคุมความคงสภาพของข้อมูลในฐานข้อมูลเชิงสัมพันธ์ 2 ข้อดังนี้

- กฎความคงสภาพของเอนติตี้ (The entity integrity rule)

หมายความว่าจะไม่มีแอททริบิวต์ที่เป็นส่วนประกอบของคีย์หลักตัวใดในรีเลชั่นได้รับอนุญาตให้มีค่าเป็นค่าว่าง โดยที่ค่าว่าง (Null value) นี้จะเป็นค่าที่ไม่ทราบแน่ชัด หรือค่าที่ไม่เหมาะสม หรืออาจกล่าวอีกนัยหนึ่งนั่นคือ ค่าว่างจะเป็นค่าที่ไม่อยู่ในกรอบของโดเมนนั่นเอง จะเห็นว่ากฎข้อนี้จะใช้ควบคุมความถูกต้องของคีย์หลัก

- กฎความคงสภาพในการอ้างอิง (The referential integrity rule)

หมายความว่า ถ้ามีรีเลชั่น R2 ซึ่งมี FK เป็นคีย์นอกที่อ้างถึงคีย์หลัก PK ในรีเลชั่น R1 สำหรับทุก ๆ ค่าของ FK นั้นจะต้อง มีค่าเท่ากับค่า PK ในทัพเพิลใดทัพเพิลหนึ่งในรีเลชั่น R1 หรือมีค่าของแอททริบิวต์ทุกตัวใน R2 เป็นค่าว่าง ความสำคัญของกฎข้อนี้คือเมื่อมีการอ้างอิงจากรีเลชั่นหนึ่งไปยังอีกรีเลชั่นหนึ่งแล้ว เราต้องมีการรับประกันว่าข้อมูลในรีเลชั่นที่สองจะต้องมีตัวตนเสมอ

2.6 การเรียกดูและใช้ข้อมูล (Data Manipulation)

ปัจจุบันได้มีการพัฒนาวิธีการในการสอบถามข้อมูล และใช้งานข้อมูล โดยในที่นี้จะกล่าวถึงการใช้งานรีเลชั่นแนลแอลจิบร้า (Relational Algebra) เป็นภาษาที่ใช้ในการสอบถามข้อมูลแบบโพซิเดอร์ ซึ่งเกิดขึ้นจากกลุ่มของการกระทำที่กระทำกับรีเลชั่นต่าง ๆ โดยต้องบอกลำดับการกระทำเพื่อให้ได้ผลลัทธ์ที่ต้องการออกมา การะทำ (Operation) เหล่านั้นคือ Select (Restrict), Project, Cartesian Product, Rename, Union, Set Difference, Set Intersection, Natural Join, Division และ Assignment ดังสามารถแสดงรูปภาพประกอบความสัมพันธ์ได้ดังรูปที่ 2.4 โดยให้รีเลชั่นต่าง ๆ มีโครงสร้างของข้อมูลดังนี้

Branch = (branch-name, assets, branch-city)

Customer = (customer-name, street, customer-city)

Client = (customer-name, broker-name)

Deposit = (branch-name, account-number, customer-name, balance)

Borrow = (branch-name, load-number, customer-name, amount)

2.6.1 The Select Operation

ตัวปฏิบัติการนี้จะเลือกทัพเพิลในรีเลชั่นออกมาตามเงื่อนไขที่ระบุโดยใช้สัญลักษณ์ Sigma(σ) แทน Select Operation และเงื่อนไขที่ระบุจะเป็นตัวห้อย (Subscript) ของ σ และชื่อของรีเลชั่นจะอยู่ในวงเล็บหลัง σ โดยที่เงื่อนไขการเปรียบเทียบที่สามารถใช้ได้ดังนี้ =, ≠, <, ≤, >, ≥ และสามารถรวมเงื่อนไขต่าง ๆ เข้าด้วยกัน โดยใช้ ตัวเชื่อม "และ(Λ) และตัวเชื่อม "หรือ(V)"

ตัวอย่าง เมื่อต้องเลือกทัพเพิลในรีเลชั่น Borrow ซึ่งกู้เงินมากกว่า 1200 บาท และ มีสาขาอยู่ในเมือง "Perryridge" สามารถแสดงได้ดังนี้

$$\sigma_{(\text{branch-name} = \text{"Perryridge"} \Lambda\ \text{amount} > 1200)}(\text{Borrow})$$

2.6.2 The Project Operation

ตัวปฏิบัติการนี้จะเลือกแอททริบิวต์ที่ต้องการในรีเลชั่นออกมา โดยที่ข้อมูลที่แสดงออกมานั้นจะต้องไม่มีทัพเพิลที่ซ้ำกันอยู่ โดยใช้สัญลักษณ์ Pi(Π) แทน Project Operation

ตัวอย่าง ต้องการแสดงแอททริบิวต์ชื่อลูกค้าและสาขาในรีเลชั่น Borrow สามารถแสดงได้ดังนี้

$$\Pi_{\text{branch-name, customer-name}}(\text{Borrow})$$

2.6.3 The Cartesian Product Operation

ตัวปฏิบัติการนี้จะรวมข้อมูลทั้งหมดที่เป็นไปได้จากรีเลชั่นต่าง ๆ ที่ต้องการโดยใช้ Cross(X) แทน Cartesian Product เนื่องจากสามารถกำหนดรีเลชั่นเป็นซับเซต (Subset) ของการทำ Cartesian Product ของเซตของโดเมนต่าง ๆ ของแต่ละแอททริบิวต์ในรีเลชั่นได้ ดังนั้นถ้ารีเลชั่น r เป็น Cartesian Product ของรีเลชั่น Client และรีเลชั่น Customer สามารถแสดงได้ดังนี้

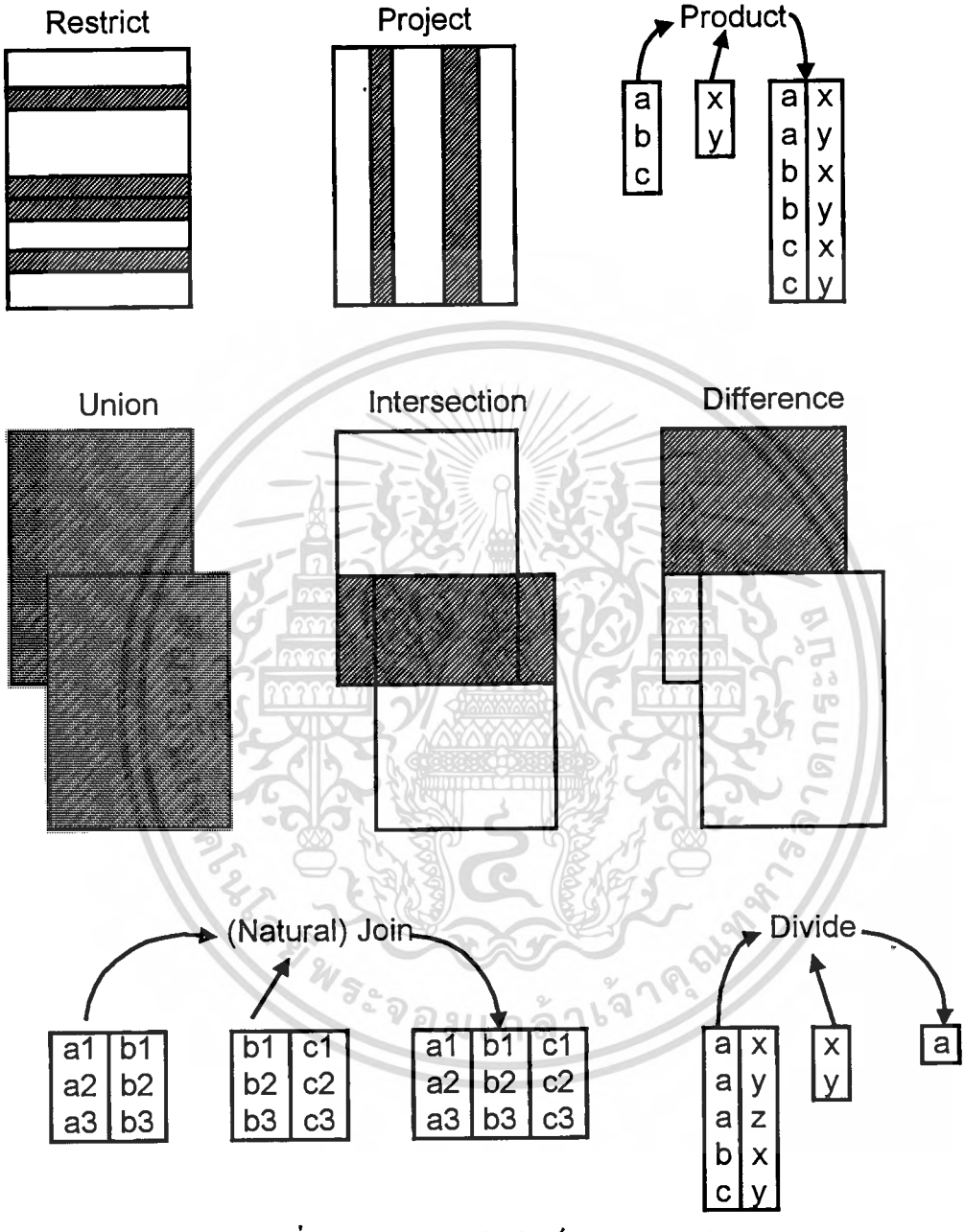
r = Client X Customer

โดยที่รีเลชั่น r จะมีโครงสร้างดังนี้

r = (client.customer-name, client.branker-name, customer.customer-name, customer.street, customer.customer-city)

2.6.4 The Rename Operation

ตัวปฏิบัติการนี้จะใช้เปลี่ยนชื่อของรีเลชั่นเป็นชื่อใหม่เพื่อประโยชน์ในการสอบถามข้อมูล โดยใช้สัญลักษณ์ $\rho_x(r)$ แทนการ Rename Operation รีเลชั่น r เป็นชื่อรีเลชั่น x



รูปที่ 2.4 แสดงความสัมพันธ์ของการกระทำ

2.6.5 The Union Operarion

ตัวปฏิบัติการนี้จะทำการรวบรวมข้อมูลผลลัพธ์ที่ได้จากการ Operation ในแต่ละรีเลชั่นเข้าด้วยกันโดยมีข้อจำกัดว่า แต่ละรีเลชั่นจะต้องมีจำนวนแอททริบิวต์เท่ากัน และต้องมีโดเมนเดียวกันในแอททริบิวต์ที่มีลำดับเดียวกันของแต่ละรีเลชั่น ซึ่งการรวมข้อมูลนี้ถ้ามีการซ้ำกันจะเลือกเอามา

เพียงตัวเดียวเท่านั้นจะใช้สัญลักษณ์ "$\cup$" แทนการยูเนียน ตัวอย่างเช่นต้องการข้อมูลลูกค้าทั้งหมดที่อยู่ในสาขา Perryridge ของรีเลชั่น Deposit และ Borrow จะได้คำตอบโดยแสดงความสัมพันธ์ต่าง ๆ ดังนี้

$$\Pi_{\text{customer-name}}(\sigma_{(\text{branch-name} = \text{"Perryridge"})}(\text{Borrow})) \cup$$

$$\Pi_{\text{customer-name}}(\sigma_{(\text{branch-name} = \text{"Perryridge"})}(\text{Deposit}))$$

2.6.6 The Set Difference Operation

ตัวปฏิบัติการนี้จะหาข้อมูลที่อยู่ในรีเลชั่นหนึ่งแต่ไม่อยู่ในอีกรีเลชั่นหนึ่ง จะใช้สัญลักษณ์ "-" แทน Set Difference Operation นั่นคือ เมื่อมี r-s จะได้ผลลัพธ์ออกมาเป็นทัพเพิลที่อยู่ในรีเลชั่น R แต่ไม่อยู่ในรีเลชั่น S ตัวอย่างเช่น ต้องการทราบจำนวนเงินฝากในบัญชีที่มากที่สุดของลูกค้าในรีเลชั่น Deposit จะได้รับความสัมพันธ์ดังนี้

$$\Pi_{\text{balance}}(\text{Deposit}) - \Pi_{\text{deposit.balance}}(\sigma_{\text{deposit.balance} < \text{d.balance}}(\text{Deposit X } \rho_{\text{d}}(\text{deposit})))$$

2.6.7 The Set Intersection Operation

ตัวปฏิบัติการนี้จะเอาเฉพาะข้อมูลในแต่ละรีเลชั่นที่เหมือนกันออกมาจะใช้สัญลักษณ์ "$\cap$" แทนการ Set Intersection และสามารถแปลงอยู่ในความสัมพันธ์ของ Set Difference ได้ดังนี้

$$r \cap s = r - (r - s)$$

และเมื่อต้องการข้อมูลของลูกค้าที่มีบัญชีอยู่ในสาขา Perryridge ทั้งในรีเลชั่น borrow และ deposit จะสามารถแสดงความสัมพันธ์ได้ดังนี้

$$\Pi_{\text{customer-name}}(\sigma_{(\text{branch-name} = \text{"Perryridge"})}(\text{Borrow})) \cap$$

$$\Pi_{\text{customer-name}}(\sigma_{(\text{branch-name} = \text{"Perryridge"})}(\text{Deposit}))$$

2.6.8 The Natural Join Operation

ตัวปฏิบัติการนี้จะรวมตัวปฏิบัติการ 2 ตัวเข้าด้วยกันคือจะทำ Cartesian Product ระหว่างรีเลชั่นเป็นอันดับแรก หลังจากนั้นจะทำ Select โดยจะเอาเฉพาะข้อมูลที่มีค่าของแอททริบิวต์ในแต่ละรีเลชั่นซึ่งอยู่ในโดเมนเดียวกัน และมีค่าเท่ากันออกมา เมื่อได้ผลลัพธ์แล้วจะเอาคอลัมน์ที่ซ้ำกันออกไป จะใช้สัญลักษณ์ "|X|" แทน Natural Join Operation ตัวอย่างเช่นต้องการข้อมูลของลูกค้าทุกคนที่กู้เงินธนาคารในสาขาที่อยู่ในเมืองที่ลูกค้าอาศัยอยู่ จะได้ความสัมพันธ์เมื่อใช้ Cartesian Product และ Select คือ

$$\Pi_{\text{borrow.customer-name,customer-city}}(\sigma_{\text{borrow.customer-name=customer.customer-name}}(\text{Borrow X Customer}))$$

และเมื่อใช้ Natural Join จะได้ความสัมพันธ์คือ

$$\Pi_{\text{borrow.customer-name,customer-city}}(\text{Borrow } |X| \text{ Customer})$$

เพื่อให้เห็นชัดเจนยิ่งขึ้นจะให้ R และ S เป็นเซตของแอททริบิวต์ในรีเลชั่น r และ s ตามลำดับ ดังนั้นแอททริบิวต์ที่มีอยู่ทั้งหมดใน R และS ก็คือ $R \cap S$ และแอททริบิวต์ทั้งหมดที่อยู่ใน R หรือ S หรือยู่ในทั้ง R และ S ก็คือ $R \cup S$ ดังนั้น Natural Join ของรีเลชั่น r และ s ก็คือ

$$r\ |X|\ s = \Pi_{R \cup S}\ (\sigma_{r.A1=s.A1 \wedge r.A2=s.A2 \wedge \ldots \wedge r.An=s.An}\ r\ X\ s)$$

เมื่อ $R \cap S = \{A1, A2, \ldots, An\}$

2.6.9 The Division Operation

ตัวปฏิบัติการนี้จะจัดการค้นหาข้อมูลที่มีเงื่อนไข "for all" มาเกี่ยวข้องด้วย โดยใช้สัญลักษณ์ "÷" แทน Division Operation ตัวอย่างเช่น ต้องการค้นข้อมูลของลูกค้าทุกคนที่มีบัญชีอยู่ในทุกสาขาในเมือง Brooklyn จะได้ความสัมพันธ์ของOperation ต่าง ๆ ได้ดังนี้

$$\Pi_{\text{customer-name,branch-name}}(\text{Deposit}) \div \Pi_{\text{branch-name}}(\sigma_{\text{branch-city = "Brooklyn"}}\ (\text{Branch}))$$

ถ้าให้ r (R) และ s(S) เป็นรีเลชั่น และให้ $S \subseteq R$ จะได้

$$r \div s = \Pi_{R\text{-}S}(r) - \Pi_{R\text{-}S}((\Pi_{R\text{-}S}(r)\ X\ s) - r)$$

2.6.10 The Assignment Operation

ตัวปฏิบัติการนี้ไม่ได้ช่วยให้ผลลัพธ์ของกลุ่ม Operation ต่าง ๆ ที่มี assignment มาเกี่ยวข้อง แต่จะช่วยให้สามารถแสดงความสัมพันธ์ของ Operation ต่าง ๆ ให้อยู่ในรูปแบบที่ซับซ้อนน้อยลง โดยจะใช้สัญลักษณ์ "←" แทน assignment โดยที่ทางขวาของสัญลักษณ์นี้จะเป็นกลุ่มของ Operation ต่าง ๆ และทางซ้ายของสัญลักษณ์จะเป็นคำตอบของ Operation ต่าง ๆ เหล่านั้นซึ่งเขียนแทนด้วยตัวแปรรีเลชั่นที่กำหนดขึ้นชั่วคราว

ตัวอย่างเช่น จากการทำงานของ Division Operation นั้นสามารถจัดรูปแบบของ r s ได้ใหม่เพื่อความสะดวกต่อการแสดงความสัมพันธ์ได้ดังนี้

$$\text{temp} \leftarrow \Pi_{R\text{-}S}(r)$$

$$\text{temp} \leftarrow \Pi_{R\text{-}S}((\text{temp}\ X\ s) - r)$$

2.7 การปรับปรุงฐานข้อมูล

การปรับปรุงข้อมูลในฐานข้อมูลสามารถทำได้โดยการ เพิ่ม ลบ และแก้ไขข้อมูล ซึ่งเป็นส่วนที่ไม่ได้กล่าวไว้ในการสอบถามและเรียกดูข้อมูล วิธีการปรับปรุงฐานข้อมูลนี้แสดงให้เห็นโดยใช้ รีเลชั่นแอลจิบร้า (Relational Algebra) โดยใช้ Assignment Operation เข้ามาช่วยซึ่งได้กล่าวมาแล้วในหัวข้อ 2.6.10

2.7.1 การลบ (Deletion)

การลบเหมือนกับการสอบถามข้อมูล เพียงแต่แตกกันว่าแทนที่จะแสดงข้อมูลทัพเพิลขึ้นมาก็นำข้อมูลทัพเพิลนั้นไปลบออกจากฐานข้อมูล การลบข้อมูลจะสามารถทำการลบทัพเพิลได้เท่านั้น ไม่สามารถลบค่าในแอททริบิวต์ได้ สามารถแสดงได้ดังนี้

$$r \leftarrow r - E$$

ซึ่ง r เป็นรีเลชั่น และ E เป็นเป็นเซตของข้อมูล ซึ่งสามารถแสดงตัวอย่างการลบข้อมูลในรีเลชั่นแนลแอลจิบร้า (Relational Agrebra) ตัวอย่างต้องการลบบัญชีทั้งหมดของ Smith

$$deposit \leftarrow deposit - \sigma_{customer\text{-}name = \text{“Smith”}}(deposit)$$

2.7.2 การเพิ่ม (Insertion)

การเพิ่มข้อมูลเข้าไปในรีเลชั่น สามารถทำได้โดยกำหนดทัพเพิลเพิ่มเข้าไป หรือทำการสอบถามเพื่อให้ได้ทัพเพิลข้อมูลแล้วทำการเพิ่มเข้า ค่าในแอททริบิวต์ที่เพิ่มเข้าไปต้องเป็นสมาชิกของโดเมนในแอททริบิวต์นั้น นอกจากนั้นทัพเพิลที่ทำการเพิ่มเข้าไปต้องเป็นข้อมูลที่ถูกต้องด้วย สามารถแสดงได้ดังนี้

$$r \leftarrow r \cup E$$

ซึ่ง r เป็นรีเลชั่น และ E เป็นเซตของข้อมูล การเพิ่มข้อมูลที่เป็นทัพเพิลเดียวเข้าไปในรีเลชั่น โดยมีการกำหนด E เป็นค่าคงที่ที่จะไปเก็บไว้ในรีเลชั่น

ตัวอย่างเช่นต้องการเพิ่มข้อมูลของ Smith มีเงิน 1,200 บาทในรหัสบัญชี 9732 ที่สาขา Perryridge สามารถแสดงได้ดังนี้

$$deposit \leftarrow deposit \cup \{(\text{“Perryridge”}, 9732, \text{“Smith”}, 1200)\}$$

2.7.3 การแก้ไข (Updating)

เมื่อต้องการเปลี่ยนแปลงค่าในทัพเพิล ถ้าหากเราต้องการเปลี่ยนค่าทุกแอททิวบิวต์ในทัพเพิล ก็สามารถทำได้โดยการลบข้อมูลทัพเพิลนั้นออกไปแล้วเพิ่มข้อมูลทัพแพิลใหม่เข้ามาได้ แต่การแก้ไขในที่นี้จะทำการแก้ไขข้อมูลในบางแอททริบิวต์เท่านั้น โดยใช้สัญลักษณ์ ซิกม่า (δ) สามารถแสดงได้ดังนี้

$$\delta_A \leftarrow {}_E(r)$$

โดยที่ r เป็นชื่อของรีเลชั่น A เป็นแอททริบิวต์ ซึ่งถูกกำหนดให้มีการเปลี่ยนแปลง E เป็นค่าที่ได้จากการคำนวณหรือเป็นค่าคงที่ ที่ต้องการเก็บไว้ในแอททริบิวต์ ตัวอย่างการใช้งาน โดยต้องการจ่ายดอกเบี้ยเพิ่มให้กับทุกสาขาอีก 5 เปอร์เซ็นต์ สามารถแสดงได้ดังนี้

$$\delta_{balance} \leftarrow {}_{balance * 1.05}(deposit)$$

บทที่ 3

การควบคุมภาวะการทำงานพร้อมกัน

ในระบบจัดการฐานข้อมูลที่ใช้กัน มักจะเป็นระบบที่มีผู้ใช้พร้อมกันหลาย ๆ คน ดังนั้น ปัญหาที่ต้องคำนึงถึงเมื่อมีการเข้าหาฐานข้อมูลเดียวกัน จากผู้ใช้หลาย ๆ คน ในช่วงเวลาเดียวกันก็ได้แก่ความถูกต้องของ การควบคุมภาวะพร้อมกัน (Concurrency Control) โดยเครื่องมือในการควบคุมภาวะพร้อมกันที่ใช้กันได้แก่ การล็อก (Lock) ข้อมูล ยังมีวิธีการควบคุมภาวะพร้อมกันวิธีอื่น ๆ อีก แต่วิธีการล็อกจะเป็นวิธีที่สะดวก และเป็นที่นิยมใช้กันมากที่สุด

3.1 การทำงานที่เป็นลำดับ (Schedules)

เมื่อมีรายการเปลี่ยนแปลงทำงานอยู่พร้อมกันหลายคน ความถูกต้องของข้อมูลสามารถถูกทำลายลงได้ แต่ถ้าการทางสามารถทำงานแบบเรียงลำดับ ก็จะสามารถรับประกันได้ว่าข้อมูลที่ประมวลผลนั้นถูกต้อง

ยกตัวอย่างในระบบการฝากถอนเงิน เมื่อมีรายการเปลี่ยนแปลง (Transaction) ซึ่งอ่าน และแก้ไขข้อมูล อยู่ 2 ราการเปลี่ยนแปลงคือ T_0 และ T_1 ทำการโอนเงินจากบัญชีหนึ่งไปยังอีกบัญชีหนึ่ง โดยที่รายการเปลี่ยนแปลง T_0 จะโอนเงิน 50 บาทจากบัญชี A. ไปยังบัญชี B. และมีการทำงานดังนี

T_0 : read(A);
A := A - 50;
write(A);
read(B);
B := B + 50;
write(B).

รายการเปลี่ยนแปลง T_1 จะโอนเงิน 10 เปอร์เซ็นต์ของยอดคงเหลือในบัญชี A ไปยังบัญชี B และมีการทำงานดังนี้

T_1 : read(A)
temp := A * 0.1;
A := A - temp;

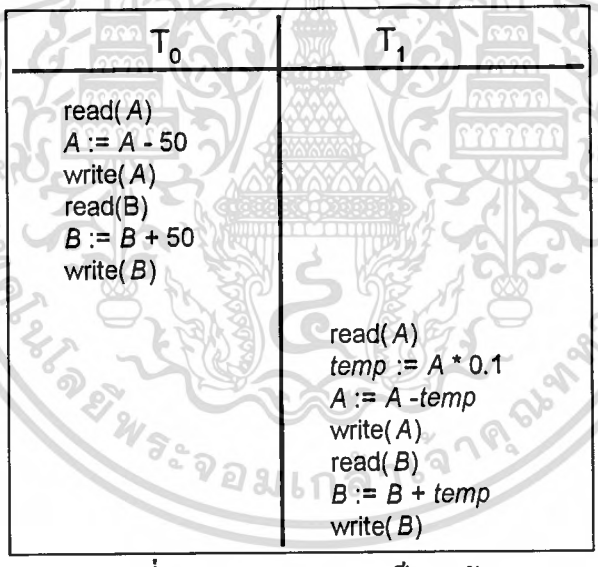
write(A);

read(B);

B := B + temp;

write(B).

ในเริ่มต้นจำนวนเงินคงเหลือบัญชี A และ B มี 1,000 และ 2,000 บาทตามลำดับ เมื่อมีการในขณะที่เรียงโดย T_0 แล้วตามด้วย T_1 การทำงานสามารถแสดงให้เห็นดังรูปที่ 3.1 ในรูปลำดับของคำสั่งจะเป็นการเรียงจากบนลงล่าง ซึ่ง T_0 จะปรากฏในช่องซ้ายของคอลัมน์ และ T_1 จะปรากฏในช่องขวาของคอลัมน์ ค่าที่ได้เมื่อมีการประมวลผลเสร็จแล้วในรูปที่ 3.1 ในบัญชี A และ B จะมีเงิน 855 และ 2145 บาทตามลำดับ เมื่อนำจำนวนเงินของทั้งสองบัญชีมารวมกันทั้งก่อนที่จะประมวลผลและหลังประมวลผลจะเป็นว่าเท่ากัน

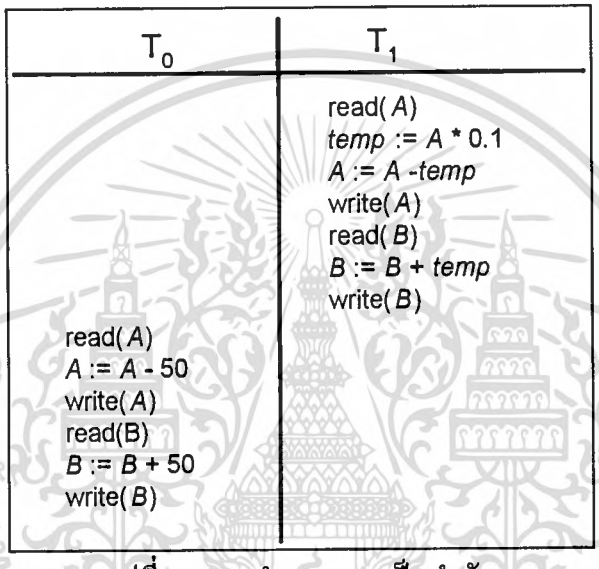


รูปที่ 3.1 การทำงานแบบเป็นลำดับ

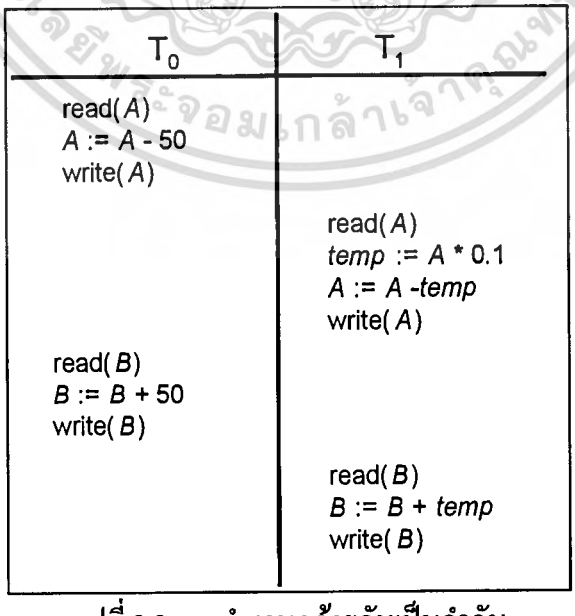
ในขณะเดียวกัน ถ้าการทำงานในขณะหนึ่งมีการเรียง T_1 แล้วตามด้วย T_0 แล้วทำงานดังรูปที่ 3.2 เมื่อนำยอดคงเหลือของทั้งสองบัญชีมารวมทั้งก่อนประมวลผล และหลังประมวลผลก็เท่ากัน แต่ยอดคงเหลือของบัญชี A และบัญชี B เป็น 850 และ 2150 บาทตามลำดับ

ลำดับของคำสั่งที่ใช้ในการประมวลผลเรียกว่า Schedules จะเป็นการเรียงลำดับของคำสั่งที่ใช้ประมวลผลในระบบ และในการทำงานเมื่อมีรายการเปลี่ยนเกิดขึ้นในเวลาเดียวกัน แล้วต้องรอ

ให้รายการเปลี่ยนแปลงก่อนหน้าทำงานเสร็จแล้วค่อยทำรายการเปลี่ยนแปลงถัดไปเรียกว่า serial schedules แต่ในการทำงานที่สามารถจะทำให้รายการเปลี่ยนแปลงทั้งสองคือ T_0แล้วทำ T_1 ซึ่งจะทำงานพร้อมกัน ขนานกันไปเรียกว่า serializable schedule ดังรูปที่ 3.3 เมื่อนำยอดคงเหลือในบัญชีก่อนการประมวลผล และหลังประมวลผลของทั้งสองบัญชีมารวมกันจะเท่ากัน และเมื่อดูยอดคงเหลือในบัญชี A และ B จะมีค่าเป็น 855 และ 2145 บาทตามลำดับ ซึ่งจะเท่ากับการทำงานในรูปที่ 3.1

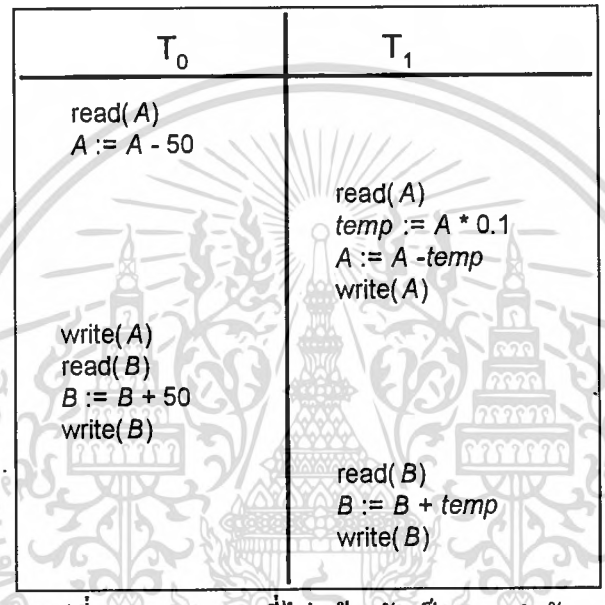


รูปที่ 3.2 การทำงานแบบเป็นลำดับ



รูปที่ 3.3 การทำงานคล้ายกับเป็นลำดับ

เมื่อต้องการทำงานเพื่อให้สามารถทำงานพร้อม ๆ กันแต่มีลำดับของคำสั่งไม่ถูกต้องก็จะทำให้ผลลัพธ์ในหาการประมวลผลข้อมูลไม่ถูกต้อง ดังรูปที่ 3.4 เมื่อทำงานเสร็จแล้วยอดคงเหลือในบัญชี A และ B เป็น 950 และ 2100 บาทตามลำดับ เมื่อนำมารวมกันจะเห็นว่ายอดเงินจะหายไป 50 บาท ทำให้สถานะของข้อมูลไม่ถูกต้อง



รูปที่ 3.4 การทำงานที่ไม่คล้ายกับเป็นแบบลำดับ

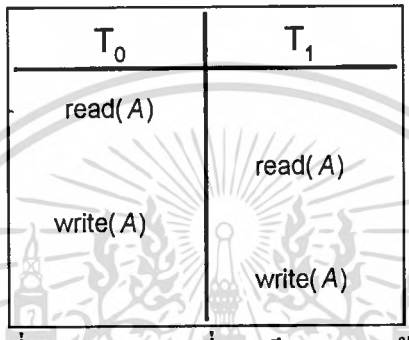
3.2 ปัญหาของภาวะพร้อมกัน 3 แบบ (The Tree Bad Dependencies)

ปัญหาของภาวะพร้อมกันเกิดขึ้นจากการที่มีผู้ใช้ หรือรายการเปลี่ยนแปลงมากกว่า 1 รายการเข้ามาแก้ไขข้อมูลชุดเดียวกันในเวลาพร้อม ๆ กันซึ่งสามารถทำให้เกิด การแทรกสลับ (interleave) การดำเนินงานของรายการเหล่านี้ การแทรกสลับนี้สามารถทำให้เกิดปัญหาต่าง ๆ ได้ 3 ประเภท คือ

- การสูญเสียผลการแก้ไข (Lost Update Problem)
- ปัญหาที่เกิดจากการย้อนกลับ (Uncommited Dependency Problem)
- ปัญหาความขัดแย้ง (Inconsistent Analysis Problem)

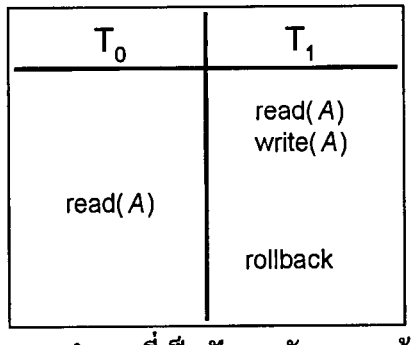
1) การสูญเสียผลการแก้ไข (Lost Update Problem) คือปัญหาที่เกิดจากการอ่านข้อมูลตัวเดียวกันไปใช้งาน โดยที่มีการเปลี่ยนแปลงข้อมูลตัวนั้นในเวลาที่ไม่พร้อมกัน

เมื่อพิจารณา รูปที่ 3.5 จะเห็นว่ามีรายการเปลี่ยนแปลงอยู่ 2 รายการคือ T_0 และ T_1 ซึ่งจะทำการแก้ไขข้อมูลชุดเดียวกันคือ R โดยจะเห็นว่ารายการ T_0 นั่นเริ่มต้นก่อน แต่ยังดำเนินงานไม่เสร็จก็ถูกแทรกด้วยรายการ T_2 (ในการแก้ไขข้อมูลใด ๆ นั้น จำเป็นที่ต้องอ่าน (fetch) ข้อมูลนั้นเข้ามาก่อน จึงจะแก้ไขได้) ซึ่งตามความเป็นจริงแล้วถ้ารายการ T_0 เกิดขึ้นก่อน รายการ T_2 ควรจะนำผลลัพธ์จาก



รูปที่ 3.5 การทำงานที่สูญเสียผลการแก้ไข

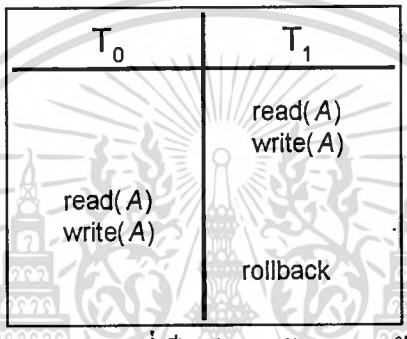
T_0 มาทำการแก้ไข เช่น ถ้าสมมติให้ T_0 และ T_1 คือรายการฝากเงินเข้าบัญชี R โดย T_0 ฝากเงินจำนวน 100 บาท และ T_2 200 บาท สมมติให้เดิมในบัญชีมีเงิน 1,000 บาท ดังนี้ถ้าระบบทำรายการเปลี่ยนแปลงได้อย่างถูกต้องแล้ว ก็ควรจะได้ผลลัพธ์รวมเป็นเงิน 1300 บาท แต่จากรูปที่ 3.x จะเห็นว่ารายการ T_0 จะอ่านค่า R เท่ากัน 1,000 บาท เข้าไปในบัฟเฟอร์ จากนั้น T_1 ก็อ่านค่า R ซึ่งเท่ากับ 1,000 บาทไปเช่นกัน แล้ว T_0 ก็แก้ไขค่า R เป็น 1,000 + 100 (=1,100) จากนั้น T_1 ก็จะแก้ไข R เป็น 1,000 + 200 เท่ากับ 1,200 ซึ่งจะทำให้สรุปว่าค่าของ R เป็น 1,200 แทนที่จะเป็น 1,300 ในสถานการณ์เช่นนี้จะเห็นว่าสูญเสียผลของการแก้ไขจากรายการเปลี่ยนแปลง T_0 ไป



รูปที่ 3.6 การทำงานที่เป็นปัญหากับผลการย้อนกลับ

2) ปัญหาที่เกิดจากการย้อนกลับ (Uncommitted Dependency Problem) คือปัญหาที่เกิดจากการที่ระบบยอมให้รายการเปลี่ยนแปลงหนึ่งอ่านข้อมูล หรือแก้ไขข้อมูลที่เพิ่งได้รับการแก้ไขจากอีกรายหนึ่ง โดยการแก้ไขที่เกิดขึ้นก่อนนี้ยังไม่ได้มีการ Commit

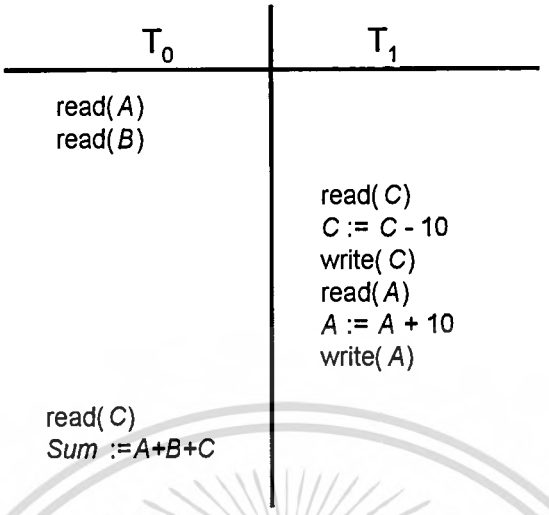
ดังนั้น จึงเกิดปัญหาขึ้นได้ถ้ามีการย้อนกลับเกิดขึ้นดังแสดงในรูปที่ 3.6 และ 3.7 ค่าของ R จะย้อนกลับเป็นค่าเดิมก่อนแก้ไขด้วยรายการ T_1 ซึ่งก็จะเห็นปัญหาว่ารายการเปลี่ยนแปลง T_0 ได้รับค่าข้อมูลที่ไม่ถูกต้องไป



รูปที่ 3.7 การทำงานที่เป็นปัญหากับผลการย้อนกลับ

3) ปัญหาความขัดแย้งของการทำงาน (Inconsistent Analysis Problem)

เมื่อมีรายการเปลี่ยนแปลง 2 รายการคือ T_0 และ T_1 โดย T_0 จะทำการรวมจำนวนเงินในบัญชี A, B และ C ในขณะที่รายการ T_1 จะโอนเงินจากบัญชี C ไปยังบัญชี A เป็นจำนวนเงิน 10 บาท ถ้ารายการทั้ง 2 นี้เกิดขึ้นพร้อม ๆ กัน โดยไม่มีการควบคุมภาวะพร้อมกันที่รัดกุมแล้ว ก็อาจจะให้ผลผิดพลาดได้ดังแสดงในรูปที่ 3.8 ซึ่งจะสมมติให้บัญชี A, B และ C มียอดเงินอยู่ 40, 50 และ 30 บาทตามลำดับ ดังนั้นผลลัพธ์ที่ถูกต้องควรจะให้ยอดรวมเท่ากับ 40+50+30 = 120 บาท และยอดเงินคงเหลือจากการโอนด้วยรายการ T_1ทำให้บัญชี A มีเงิน 50 บาท และบัญชี C มีเงิน 20 บาท แต่เมื่อพิจารณาดูการทำงานในรูปที่ 3.8 แล้วจะเห็นว่าการแทรกสลับของการทำงาน ทำให้รายการ T_0 อ่านค่าจากบัญชี ก ก่อนการโอน และอ่านจากบัญชี C หลังการโอนเงิน จึงให้ค่ารวมที่ขัดแย้งกับความเป็นจริง



รูปที่ 3.8 การทำงานที่เกิดการขัดแย้ง

3.3 การล็อก (Locking)

จากหัวข้อก่อนหน้าจะพบว่าปัญหาของภาวะพร้อมกันเกิดขึ้นมาจากการที่เราเริ่มทำรายการเปลี่ยนแปลง เช่นเริ่มอ่านข้อมูลเข้ามาแล้วยังไม่ทันทีจะทำรายการนี้เสร็จ ก็มีรายการอื่นเข้ามาสอดแทรก โดยเฉพาะอย่างยิ่งการสอดแทรกนั้นเป็นการแก้ไขข้อมูลล่าสุด ดังนั้นวิธีการแก้ไขข้อมูลที่ดูจะสะดวกที่สุด ได้แก่การล็อกข้อมูลเอาไว้ เมื่อเราเริ่มจากรายการเปลี่ยนแปลงกับข้อมูลใด โดยจะไม่ยอมให้มีรายการอื่นเข้ามาใช้ข้อมูลนั้นได้จนกว่าจะทำรายการแรกเสร็จ ในการใช้งานล็อกแก้ปัญหาภาวะพร้อมกันนั้นจะต้องประกอบ 3 ส่วนด้วยกัน

1) Lock Primitives
2) Compatibility matrix
3) Locking Protocol

1) Lock Primitives คือรูปแบบการล็อกข้อมูลใน หัวข้อนี้จะกล่าวถึง 2 รูปแบบด้วยกันคือ

- Shared คือถ้ารายการเปลี่ยนแปลง Ti อยู่ในรูปแบบของ shared-mode lock (ใช้สัญลักษณ์ S) บนข้อมูล Q แล้ว Ti สามารถอ่านข้อมูลได้ แต่ไม่สามารถแก้ไขได้บน ข้อมูล Q
- Exclusive คือถ้ารายการเปลี่ยนแปลง Ti อยู่ในรูปแบบของ exclusive-mode lock (ใช้สัญลักษณ์ X) บนข้อมูล Q แล้ว Ti สามารถไม่สามารถอ่านและแก้ไขข้อมูลได้บนข้อมูล Q

2) Compatibility matrix เป็นเครื่องมือที่ตรวจสอบการล็อกว่าจะยอมให้ล็อก หรือไม่ล็อก โดยในแถวเป็นการแสดงว่ารายการล็อกอยู่ และในคอลัมน์เป็นการขอล็อกบ้าง ส่วนคำตอบถ้าเป็น true หมายความว่ายอมให้ล็อก แต่ถ้าเป็น false หมายความว่าไม่ยอมให้ล็อก ต้องรอ

Compatibility		Mode of Lock	
		Share	Exclusive
Mode of Request	Share	True	False
	Exclusive	False	False

รูปที่ 3.8 Lock compatibility matrix

3) **Locking Protocol** เมื่อมีการใช้ฐานข้อมูลร่วมกันระหว่างผู้ใช้งานหลาย ๆ คนเมื่อรายการใดเริ่มใช้ข้อมูลระบบจัดการฐานข้อมูล (DBMS) ต้องทำการล็อกข้อมูลนั้น เพื่อให้ทำงานถูกต้องตามลำดับ Two Phase Locking Protocol เป็นรูปแบบหนึ่งที่สามารถใช้ในรายการเปลี่ยนแปลงในประเด็นของการล็อก และปลดล็อกสามารถแบ่งได้เป็น 2 ส่วน

- Growing Phase เมื่อรายการเปลี่ยนแปลงทำการล็อกข้อมูล ก็จะทำการล็อกไปเรื่อย ๆ จนกว่าจะเสร็จในรายการเปลี่ยนแปลงนั้น

- Shrink Phase เมื่อจะทำรายการเปลี่ยนแปลงนั้นเสร็จแล้วจะทำการปลดล็อก ก็จะทำการปลดล็อกข้อมูลทั้งหมด ก่อนที่จะมีการล็อกครั้งใหม่

บทที่ 4
เวลาที่ใช้ในการประมวลผล

บทนี้จะกล่าวถึงการวัดประสิทธิภาพของการประมวลผลข้อมูลสำหรับฐานข้อมูลเชิงสัมพันธ์ ในด้านเวลาที่ใช้ในการประมวลผลข้อมูล (Response time) ของการทำงานของแอพพลิเคชั่นที่ร่วมทำงานกับระบบฐานข้อมูลเชิงสัมพันธ์ หรือระบบจัดการฐานข้อมูลเชิงสัมพันธ์ โดยในการศึกษาครั้งนี้ได้นำวิธีการวัดประสิทธิภาพการทำงานของ Winconsin benchmark ซึ่งเป็น benchmark ที่ใช้การสังเคราะห์ข้อมูลในฐานข้อมูลขึ้นมา และใช้คำถามในการสอบถามข้อมูลที่เป็นแบบพื้นฐาน ที่สามารถใช้งานได้ง่ายบนระบบฐานข้อมูลเชิงสัมพันธ์ เข้ามาประยุกต์ให้เกิดความเหมาะสม

4.1 ระบบฐานข้อมูลที่ใช้ในการทดลอง

ระบบฐานข้อมูลก็เป็นสิ่งหนึ่งที่ทำให้ประสิทธิภาพการทำงาน ในด้านของเวลาในการประมวลผลข้อมูลให้ผลที่แตกต่างกันไป โดยในการศึกษาครั้งนี้จะใช้ระบบฐานข้อมูลที่มีการทำงานแตกต่างกัน 2 ชนิดคือ

- ฐานข้อมูลพาราด็อกซ์ (Paradox Database)
- ระบบจัดการฐานข้อมูลออราเคิล
  (ORACLE DBMS : Oracle Database Management System)

ระบบฐานข้อมูลแต่ละชนิด จะมีฐานข้อมูลที่ใช้ในการทดลองจะมีอยู่ 2 ชุดด้วยกัน คือมีอินเด็กซ์ (Index) และไม่มีอินเด็กซ์ (No Index) แต่ละชุดประกอบไปด้วย 4 รีเลชั่น โดยมีแอททริบิวต์และจำนวนคาร์ดินอลิตี้ที่แตกต่างกันดังนี้

รีเลชั่น Course มีข้อมูลจำนวน 1,000 ทัพเพิล

อินเด็กซ์ (Index) คือ Cno, Dept_Code, Department

ชื่อ	ชนิด	รูปแบบ	หมายเหตุ
Cno	String(5)	random	Candidate Key
Cname	Sting(21)	random	Candidate Key
Dept_code	Int	rotating	11,12,13,...,110
Department	String(30)	rotating	ตามชื่อสาขาวิชา
Credit	Int		

รีเลชั่น Student มีข้อมูลจำนวน 10,000 ทัพเพิล

อินเด็กซ์ (Index) คือ Stno, Dept_Code, Department, Faculty

ชื่อ	ชนิด	รูปแบบ	หมายเหตุ
Stno	String(7)	random	Candidate Key
Title	String(4)	rotating	{"นาย", "นส."}
Name	String(30)	random	Candidate Key
Dept_Code	Int	rotating	11,12,13, ... ,110
Department	String(30)	rotating	ตามชื่อสาขา
Faculty	String(20)	rotating	ตามชื่อคณะ
Tel	String(7)	random	0000000-9999999

รีเลชั่น Grade มีข้อมูลจำนวน 40,000 ทัพเพิล

อินเด็กซ์ (Index) คือ Stno+Cno, Cno, EmpNo

ชื่อ	ชนิด	รูปแบบ	หมายเหตุ
Stno	String(7)	random	เป็นคีย์ร่วมกับ Cno
Cno	String(5)	random	เป็นคีย์ร่วมกับ Stno
EmpNo	String(4)	random	จากรีเลชั่น Instructor
Sem	String(5)	มีการกำหนดไว้	"ต้น"
Year	Int	มีการกำหนดไว้	"2540"
Grade	Int	random	0-100

รีเลชั่น Instructor มีข้อมูลจำนวน 1,000 ทัพเพิล

อินเด็กซ์ (Index) คือ EmpNo, Salary, Dept_Code

ชื่อ	ชนิด	รูปแบบ	หมายเหตุ
EmpNo	String(4)	random	Candidate Key
Name	String(30)	random	Candidate Key
RoomNo	Int	random	0-999
Salary	Int	rotating	{10000, 15000, 20000, 25000, 30000}
Rank	String(8)	rotating	{ศ., ศ.ดร., รศ., รศ.ดร., ผศ., ผศ.ดร., ดร., อ. }
Dept_Code	Int	rotating	ตามชื่อสาขาวิชา

4.2 รูปแบบการเขียนโปรแกรม

รูปแบบเขียนโปรแกรมที่จะทำการศึกษานั้นสามารถแบ่งออกไปได้ 2 วิธีหลัก ๆ คือ วิธีแรกคือ การเขียนโปรแกรมแบบอ่านข้อมูลครั้งละหนึ่งเรคอร์ด (record-at-a-time) มาประมวลผล ซึ่งในวิธีนี้ผู้เขียนโปรแกรมต้องควบคุมการทำงานเองทั้งหมด รวมทั้งเทคนิควิธีการเขียนที่จะสามารถให้ได้ผลลัพธ์ออกมาเร็วที่สุด การเรียงลำดับของข้อมูลก็เป็นปัจจัยสำคัญอย่างหนึ่ง ตัวอย่างเช่นเมื่อต้องการรวมข้อมูลจากสองตารางกัน (Join) หากข้อมูลที่มีการเรียงลำดับไว้แล้ว เราสามารถใช้เทคนิคที่เรียกว่าการเขียนโปรแกรมแบบ Merge Join ได้ซึ่งสามารถทำงานได้เร็วกว่าการ Nest Loop Join การเขียนโปรแกรมแบบอ่านข้อมูลครั้งละหนึ่งเรคอร์ด (record-at-a-time) มีการทำงานพื้นฐานดังนี้คือ มีการเปิดแฟ้มข้อมูลจากเครื่องคอมพิวเตอร์ที่ติดต่อกับผู้ใช้ (Open table from Font End), อ่านข้อมูลขึ้นมาขึ้นมา (fetch) แล้วก็ปิดแฟ้มข้อมูล (Close table)

ส่วนอีกวิธีหนึ่งคือใช้คำสั่งเอสคิวแอล (SQL) ในการประมวลผล ในวิธีนี้คำสั่งเอสคิวเอล (SQL) จะถูกส่งไปยังตัวจัดการระบบฐานข้อมูล(DBMS) วิธีการประมวลผลก็ขึ้นอยู่ตัวจัดการระบบฐานข้อมูล(DBMS) เป็นประมวลผลข้อมูลทั้งหมดแล้วส่งผลลัพธ์กลับไป ผู้เขียนโปรแกรมจะไม่ทราบวิธีการทำงานเลย ฉะนั้นในวิธีนี้ระยะเวลาในการประมวลผลข้อมูล จึงขึ้นอยู่กับความเก่งและฉลาดของตัวจัดการระบบฐานข้อมูล(DBMS) ที่เราใช้งานอยู่

4.3 วิธีที่ใช้ทดลองในการสอบถามข้อมูล (Query Test)

จากที่ได้กล่าวไว้แล้วในบทที่ 1 ว่า ในการศึกษาครั้งจะทำการศึกษารูปแบบการเขียนโปรแกรมด้วย ฉะนั้นการเขียนโปรแกรมแบบครั้งละหนึ่งเรคอร์ด (Record-at-a-time) เพื่อให้สามารถทำงานได้เร็วที่สุด ก็จะทำการเขียนโปรแกรมโดยนำการทำงานของ Query Optimization เข้ามาประยุกต์

วิธีที่ใช้สอบถามข้อมูลที่อยู่ในโครงสร้างเชิงสัมพันธ์ และรวมถึงการปรับปรุงข้อมูลในฐานข้อมูลเชิงสัมพันธ์ ที่น่าสนใจสามารถแบ่งออกเป็น 5 ประเภทหลัก ๆ ดังนี้

1) การเลือกทัพเพิลข้อมูล (Selection)
2) การเลือกคอลัมน์ข้อมูล (Projection)
3) การนำข้อมูลมาร่วมกัน ทั้งแบบ 2 ทาง และ 3 ทาง (2-way and 3 way joins)
4) การหาคำตอบจากข้อมูลหลาย ๆ ทัพเพิล (Aggregate)
5) การปรับปรุงข้อมูล-เพิ่ม, แก้ไข, ลบ (Updates: insert, delete, update)

4.4 การเลือกทัพเพิลข้อมูล (Selection)

การเลือกทัพเพิลข้อมูลจากรีเลชั่นในการเขียนโปรแกรมในรูปแบบอ่านข้อมูลครั้งละหนึ่งเรคอร์ด (record-at-a-time) เมื่อไม่ใช้อินเด็ก ในจะต้องเขียนโปรแกรมให้อ่านต้องข้อมูลเริ่มต้นตั้งทัพเพิลแรกไปจนทัพเพิลสุดท้ายของรีเลชั่น และในการอ่านข้อมูลขึ้นมาแต่ละทัพเพิลก็นำมาตรวจสอบเงื่อนไขทันที แล้วค่อยทำการอ่านข้อมูลในทัพเพิลถัดไป ดังแสดงในรูปที่ 4.1

```
for (each tuple  t in R)
    if ( t satisfies the remaining qualification)
            Add to the result tuple   t
    endif
endfor
```

รูปที่ 4.1 การเลือกทัพเพิลเมื่อไม่มีอินเด็ก

การทำงานในรูปแบบอ่านข้อมูลครั้งละหนึ่งเรคอร์ด (record-at-a-time) เมื่อใช้งานอินเด็ก โดยที่อินเด็กใช้อยู่ต้องเป็นอินเด็กตามแอททริบิวต์ที่เราต้องการค้นหาข้อมูลด้วย โดยการเปิดอินเด็กนั้นขึ้นมาใช้งาน แล้วทำการค้นหาข้อมูลที่ต้องการตามอินเด็กดังกล่าว การอ่านข้อมูลข้อมูลก็จะอ่านเฉพาะทัพเพิลที่ต้องการตรงตามเงื่อนไขเท่านั้น ดังแสดงในรูปที่ 4.2

การเลือกทัพเพิลข้อมูลจากระบบฐานข้อมูล จะมีคำถามที่ใช้ในการสอบถาม 3 อย่างดังนี้

Q1) เลือกทัพเพิลจากรีเลชั่น Student โดยที่ Dept_code เท่ากับ 50

Q2) เลือกทัพเพิลจากรีเลชั่น Student โดยที่ Department เป็น "สาขาครุศาสตร์ภาษาไทย"

Q3) เลือกทัพเพิลจากรีเลชั่น Student โดยที่ Faculty เป็น "คณะเป็นคณะครุศาสตร์"

Q4)เลือกทัพเพิลจากรีเลชั่น Student โดยนำอักขระที่3-4 ของ Stno มีค่าเท่ากับ "05"

```
Search index I1 for index records ( v1, ptr), adding to temp1 the pointer ptr
Search index I2 for index records ( v2, ptr), adding to temp2 the pointer ptr
intersect ← (temp1 ∩ temp2)
for (each pointer ptr ∈ intersect)
    Retrieve from F the tuple t pointed to by ptr
    { Note the pointers in intersect can be arranged so that no block of
      F need be read more than once.}
    if (t satisfies the remaining qualification)
       Add to the result the tuple t
    endif
endfor
```

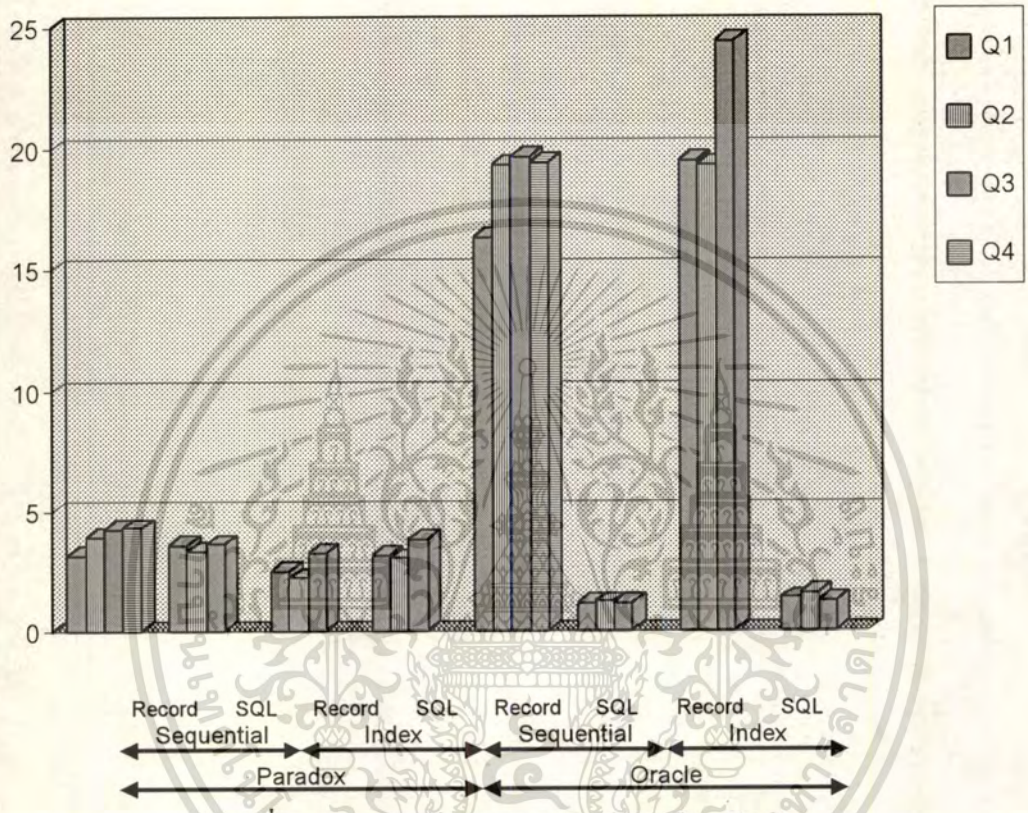
รูปที่ 4.2 การเลือกทัพเพิลเมื่อมีอินเด็ก

ในการเลือกข้อมูลจะนำข้อมูลทุกแอททริบิวต์จากรีเลชั่นขึ้นมา โดยมีการกำหนดเงื่อนไขที่ต้องการ ในการเลือกทัพเพิลข้อมูลนั้นจะทำการสอบถามโดยการเลือกข้อมูลจากรีเลชั่น student ขึ้นมา 1% (Q1 และ Q2) และ 10% (Q3 และ Q4) ของข้อมูลทั้งหมด ในการสอบถามที่ Q1 และ Q2 เป็นการสอบถามในเงื่อนไขเดียวกัน โดยใช้แอททริบิวต์ต่างกัน แต่มีความหมายเหมือนกัน แอททริบิวต์ Dept_code ใช้รหัสแทนข้อมูล แอททริบิวต์ Department แทนด้วยตัวข้อมูล และใน Q4 เป็นการสอบถามข้อมูลโดยมีการเปรียบเทียบข้อมูลบางส่วนของแอททริบิวต์ที่เรียกว่า meaning full digit

จากกราฟในรูปที่ 4.3 จะเห็นว่าการเขียนโปรแกรมแบบอ่านข้อมูลครั้งละหนึ่งเรคอร์ด และมีโครงสร้างไฟล์แบบ Sequential ในการเขียนโปรแกรมต้องมีการต้องมีการเปรียบเทียบข้อมูลเป็นจำนวนมาก การประมวลผลใน Q1 จึงใช้เวลาน้อยกว่าใน Q2 จะทำให้เห็นว่าการประมวลผลในลักษณะดังกล่าว ขนาดของแอททริบิวต์มีผลกับเวลาที่ใช้ในการประมวลผล ส่วนการประมวลในรูปแบบอื่นใน Q1 และ Q2 จะใช้เวลาที่ใกล้เคียงกัน การใช้งานอินเด็กไม่ได้ช่วยให้การประมวลผลแตกต่างกันมากอาจเป็นเพราะมีจำนวนอินเด็กเป็นจำนวนมาก และขนาดของข้อมูลในฐานข้อมูลยังไม่มากเพียงพอ

การเขียนประมวลผลข้อมูลบนฐานข้อมูลพาราด็อกซ์ รูปแบบการเขียนโปรแกรมแบบอ่านข้อมูลครั้งละหนึ่งเรคอร์ด หรือใช้คำสั่งเอสคิวแอล(SQL) จะใช้เวลาไม่แตกต่างกัน แต่ถ้าประมวล

ผลบนฐานข้อมูลออราเคิล การเขียนโปรแกรมแบบอ่านข้อมูลครั้งละหนึ่งเรคอร์ดจะใช้เวลามากกว่าการใช้คำสั่งเอสคิวแอล ส่วนการประมวลผลที่ใช้เวลาน้อยที่สุดคือการเขียนโปรแกรมโดยใช้คำสั่งเอสคิวแอลบนระบบฐานข้อมูลออราเคิล



รูปที่ 4.3 กราฟแสดงผลการทดลองการเลือกทัพเพิล

4.5 การนำข้อมูลจากหลายรีเลชั่นมารวมกัน (Joins)

ในการศึกษาการนำข้อมูลจากหลายรีเลชั่นมารวมกัน (Joins) ปัจจัยที่น่าสนใจอยู่ 2 อย่างคือ

- การใช้อัลกอริทึมในการ Join ที่ทำให้การสอบถามข้อมูลที่มีประสิทธิภาพมากที่สุด (เช่น nested-loops, sort-merge หรือ hashed join) รวมถึงการนำอินเด็กมาใช้งาน
- การเลือกขั้นตอนการทำงาน ที่ให้มีความซับซ้อนในการ Joins น้อยที่สุด

การทำงานในรูปแบบอ่านข้อมูลครั้งละหนึ่งเรคอร์ด (record-at-a-time) เมื่อไม่ใช้อินเด็กจะทำการอ่านข้อมูลขึ้นมาเปรียบเทียบ โดยการเปรียบเทียบข้อมูลที่มีอยู่ทั้งหมด เมื่อข้อมูลในรีเลชั่น R_1 มี m ทัพเพิล และ R_2 มี n ทัพเพิล ในการ join ข้อมูลจึงต้องมีการเปรียบเทียบ m*n ครั้ง ซึ่งเรียกวิธีการทำงานแบบนี้ว่า Nest-Loop Join ดังแสดงในรูป 4.4

การทำงานในรูปแบบอ่านข้อมูลครั้งละหนึ่งเรคอร์ด (record-at-a-time) เมื่อใช้งานอินเด็ก

เมื่อข้อมูลในรีเลชั่นจะถูกเรียงลำดับมาแล้ว ถ้าหากใช้การทำงานแบบ Nest-Loop Join ก็จะทำให้การประมวลผลช้า การนำข้อได้เปรียบของข้อมูลที่เรียงลำดับแล้วในใช้ประมวลผล ซึ่งวิธีหนึ่งที่สามารถทำงานได้เรียกว่า Merge Join จะทำการเปรียบเทียบข้อมูลกันดังนี้คือ เมื่อกำหนดให้การ

```
for (each tuple t in R1)
    for (each tuple u in R2)
        if ( t[A] = u[B])
            Append to result stream    STR the tuple t x u
        endif
    endfor
endfor
```

รูปที่ 4.4 การรวมข้อมูลเมื่อไม่มีอินเด็ก

เรียงลำดับข้อมูลจากน้อยไปหามาก แอททริบิวต์ที่นำมา Join กันมีความสัมพันธ์แบบ 1:1 แล้วในการเปรียบเทียบ โดยจะเริ่มเปรียบเทียบข้อมูลทัพเพิลแรกของรีเลชั่น R_1 กับทัพเพิลแรกของรีเลชั่น R_2 ถ้าข้อมูลในรีเลชั่น R_1 มากกว่าก็จะทำการอ่านข้อมูลในทัพเพิลถัดไปของรีเลชั่น R_2 ขึ้นมาเปรียบเทียบใหม่ ในการเปรียบเทียบครั้งที่สองนี้ ถ้าหากข้อมูลในรีเลชั่น R_2 มากกว่าก็จะทำการอ่านข้อมูลในทัพเพิลถัดไปของรีเลชั่น R_1 ทำการเปรียบเทียบเช่นจนหมดข้อมูล จะเห็นว่าในการเปรียบเทียบจะน้อยลงมาก เมื่อข้อมูลในรีเลชั่น R_1 มี m ทัพเพิล และ R_2 มี n ทัพเพิล ในการ join ข้อมูลจึงมีการเปรียบเทียบมากที่สุด m+n ครั้ง ดังแสดง4.5

การนำข้อมูลจากหลายรีเลชั่นมารวมกัน จะมีคำถามที่ใช้ในการสอบถาม 3 อย่างดังนี้

Q5) นำข้อมูลจากรีเลชั่น Student และ Grade มารวมกัน โดยมีเงื่อนไขแอททริบิวต์ Stno มีค่าเท่ากัน

Q6) นำข้อมูลจากรีเลชั่น Student , Course และ Grade มารวมกัน โดยมีเงื่อนไขแอททริบิวต์ Stno ในรีเลชั่น Student กับ Grade มีค่าเท่ากัน และแอททริบิวต์ Cno ในรีเลชั่น Grade กับ Course มีค่าเท่ากัน

Q7) นำข้อมูลจากรีเลชั่น Student และ Grade มารวมกัน โดยมีเงื่อนไขแอททริบิวต์ Stno มีค่าเท่ากัน และ แอททริบิวต์ Grade ในรีเลชั่น Grade มีค่ามากกว่า 80

ในการ Join แบบ 2 ทาง (Q5) นั้นจะทำการ Join ข้อมูลจาก 2 รีเลชั่นคือ รีเลชั่น Student และ Grade โดยมีข้อมูลอยู่ 10,000 เรคอร์ด และ40,000 เรคอร์ดตามลำดับ เป็นการ Join แบบ 1:m ในการสอบถามที่ 6 (Q6) จะการ Join แบบ 3 ทาง โดยในการ Join เป็นแบบครั้งแรกเป็นแบบ 1:m

```
Init_Traverse( I1, STR1)
Init_Traverse( I2, STR2)
IR1 ← Get_Next( STR1)
IR2 ← Get_Next( STR2)
while (neither IR1 or IR2 is NULL)
    if ( IR1val = IR2val)
        Issue a sequence of calls to Get_Next(STR 1) to retrieve all index
            records from I 1 with the common value IR1val = IR2val = v.
            These recofds are consecutive in the traversal order.
        Retrieve from the data file F1 into the collections of data record
            pointed to by these index records.
        Issue a sequence of calls to Get_Next(STR 2) to retrieve (individually)
            each index record from I1 such that IR2val = v . These record are
            consecutive in the traversal order. As each index record is returned,
            retrieve from the data file F2 each data record t pointed to by this
            index record and append to STR temp×{t}
        {These sequences of calls to Get_Next terminates with IR1 containing
         the first index record of I1 with value greater than v and IR2 containing
         the first index record of I1 with value greater than v. If the traversal of
         either file is complete, IR2 or IR2 will contain the NULL record.}
    elseif ( IR1val < IR2val)
            Get_Next(STR 1)
    else
            { IR1val > IR2val}
            Get_Next(STR 1)
    endif
end while
```

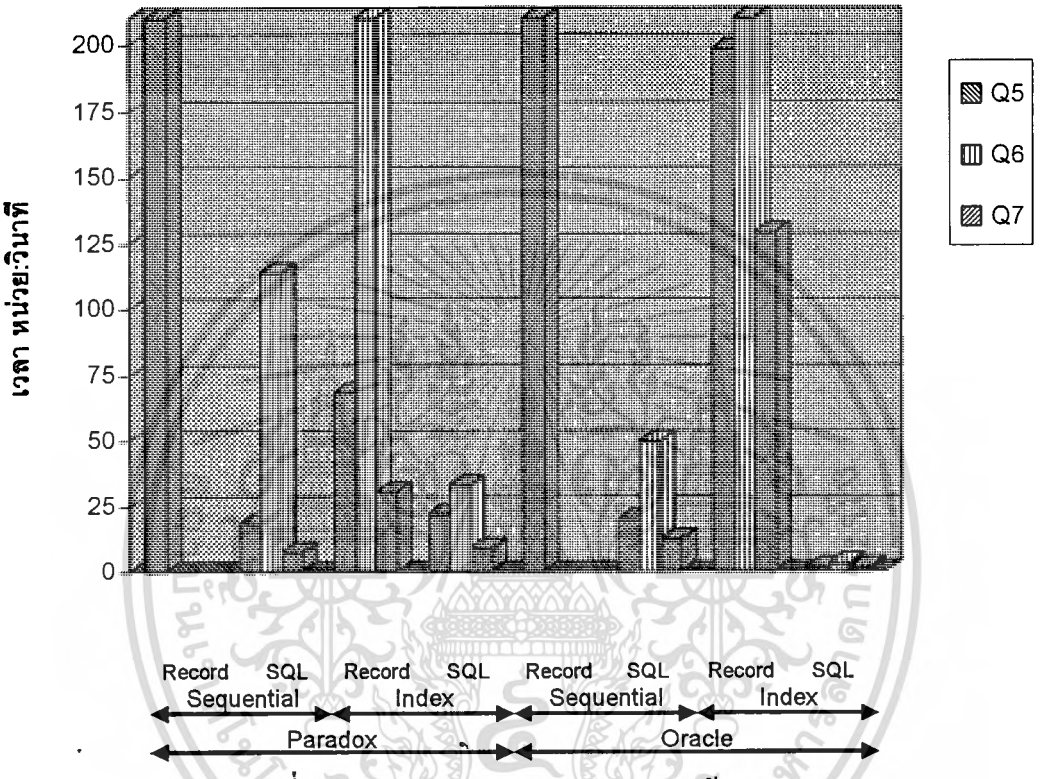
รูปที่ 4.5 การรวมข้อมูลเมื่อมีอินเด็ก

และการ Join ครั้งที่สองเป็นแบบ 1:m ส่วนในการสอบถามที่ 7 (Q7) การ Join แบบ 2 ทาง และมีเงื่อนไขในการเลือกข้อมูลด้วยโดยการทำงาน ที่มีประสิทธิภาพนั้นก็ควร ที่ทำการเลือกข้อมูลในทัพเพิลในเงื่อนไขเป็นอันดับแรกแล้วค่อยทำการ Join ข้อมูล

จากกราฟรูป 4.6 จะเห็นว่าในการเขียนโปรแกรมที่เป็นแบบอ่านข้อมูลมาครั้งหนึ่งเรคอร์ด และใช้กับโครงสร้างแฟ้มข้อมูลที่ไม่มีอินเด็ก จะใช้เวลาในการประมวลผลนานมากคือ 32 ชั่วโมง ในฐานข้อมูลพาราด็อกซ์ และ 88 ชั่วโมงในฐานข้อมูลออราเคิล โดยใช้การสอบถาม Q5 ดังที่ได้แสดงในการทดลองลำดับที่ 25 และ 29 เมื่อเปรียบเทียบกับการทดลองอื่นที่มีการสอบถามแบบโดยใช้คำสั่งเอสคิวแอล (SQL) แต่การเขียนโปรแกรมอ่านข้อมูลครั้งละหนึ่งเรคอร์ดจะเร็วขึ้นมาก ถ้ามีการนำอินเด็กมาใช้งาน คือ 29 วินาทีในระบบฐานข้อมูลพาราด็อกซ์ และ 1 นาที 23 วินาทีในระบบฐานข้อมูลออราเคิล

การเขียนโปรแกรมในการรวมข้อมูลจากหลายรีเลชั่น เมื่อทำงานในสภาพแวดล้อมเดียกันแล้ว การเขียนโปรแกรมครั้งละหนึ่งเรคอร์ด (Record-at-a-time) จะใช้เวลามากกว่าการเขียน

โปรแกรมด้วยคำสั่งเอสคิวแอล (SQL) โดยการทำงานด้วยคำสั่งเอสคิวแอล (SQL) ไม่ได้ใช้งานอินเด็ก จะสามารถประมวลได้เร็วกว่าการเขียนโปรแกรมแบบอ่านข้อมูลขึ้นมาครั้งละหนึ่งโดยมีการใช้งานอินเด็ก



รูปที่ 4.6 กราฟแสดงการทดลองการรวมข้อมูล

4.6 การเลือกแอททริบิวต์จากรีเลชั่น (Project)

การสอบถามข้อมูลแบบการเลือกแอททริบิวต์นั้น เมื่อเลือกข้อมูลไปไว้ในรีเลชั่นใหม่แล้วต้องทำให้ข้อมูลนั้นมีคุณสมบัติเป็นรีเลชั่นที่สมบูรณ์ด้วย คุณสมบัติที่สำคัญอย่างหนึ่งที่ควรคำนึงในที่นี้คือ ข้อมูลแต่ละทัพเพิลของรีเลชั่นเดียวจะบรรจุข้อมูลเพียงค่าเดียว (There are no duplicate tuples) การทำงานในรูปแบบอ่านข้อมูลครั้งละหนึ่งเรคอร์ด (record-at-a-time) เมื่อใช้งานอินเด็กและไม่ใช้งานอินเด็ก จะมีการทำงานที่เหมือนกัน คือต้องมีการอ่านข้อมาแล้วนำแอททริบิวต์ที่ต้องการขึ้นมาจากรีเลชั่นทั้งหมด หลังจากนั้นต้องมีนำข้อมูลที่ซ้ำซ้อนกันออกไปดังรูปที่ 4.7 จะเห็นได้ว่าข้อมูลที่ถูกเรียงลำดับแล้วนั้นไม่ได้ช่วยให้การประมวลข้อมูลเร็วขึ้น แต่ถ้าการสอบข้อมูลที่มีคีย์แข่งขัน (Candidate key) ขึ้นมาก็จะทำให้การนำข้อมูลที่ขึ้นมาครั้งแรกไม่มีทัพเพิลที่ซ้ำซ้อนกัน

การเลือกแอททริบิวต์จากรีเลชั่น จะมีคำถามที่ใช้ในการสอบถาม 2 อย่างดังนี้

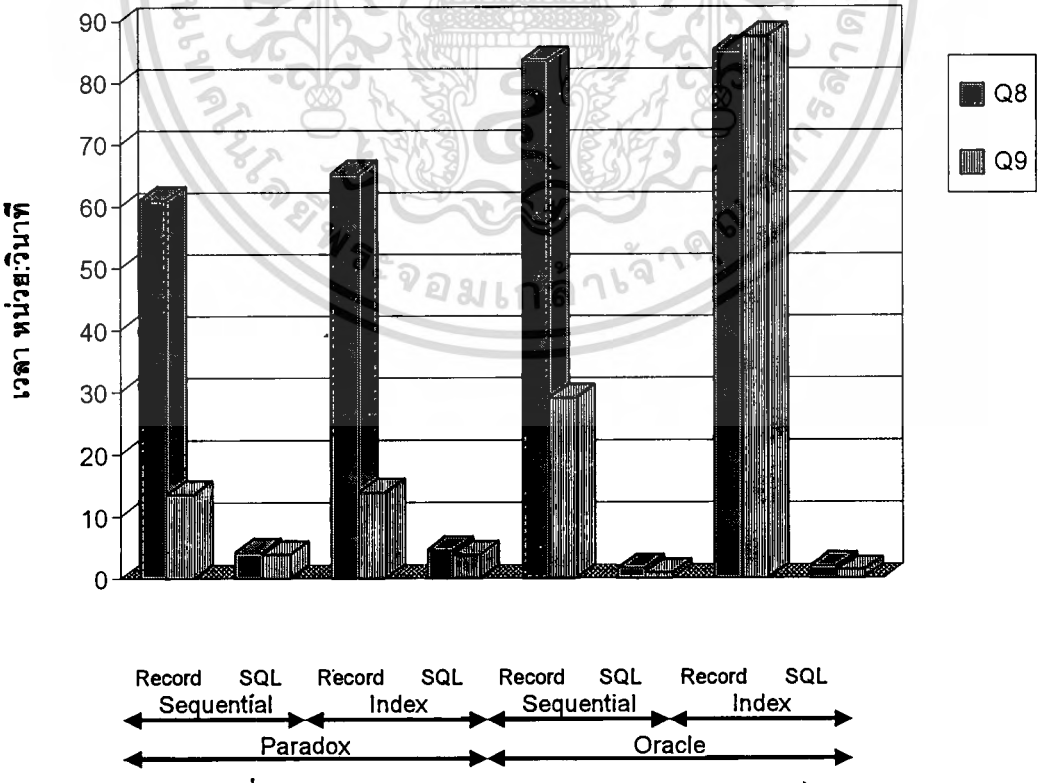
Q8) เลือกแอททริบิวต์ Dept_code, Department และ Faculty จากรีเลชั่น Student

Q9) เลือกแอททริบิวต์ทุกแอททริบิวต์ จากรีเลชั่น Student

```
for (each tuple  t in R)
    Add in  R1 attibute( a1,a2..,ak) tuple  t
endfor
for (each tuple  u in R1)
   delete duplicate tuple
endfor
```

รูปที่ 4.7 การเลือกแอททริบิวต์

การสอบถามข้อมูลที่ 8 (Q8) จะเป็นการเลือกแอททริบิวต์ที่ไม่มีแอททริบิวต์ที่เป็นคีย์แข่งขัน (Candidate Key) ฉะนั้นการประมวลผลข้อมูลจึงแบ่งเป็น 2 ส่วน ส่วนแรกจะเป็นการเลือกข้อมูลในแอททริบิวต์นั้นขึ้นมา และในส่วนที่สองจะเป็นการเลือกข้อมูลที่ซ้ำซ้อนออก แต่ในการสอบถามข้อมูลที่ 9 (Q9) เป็นประมวลผลข้อมูลเลือกแอททริบิวต์ที่มีคีย์แข่งขันมาด้วย ฉะนั้นในประมวลผลก็ทำเพียงส่วนเดียวก็พอ คือเลือกข้อมูลในแอททริบิวต์นั้นขึ้นมา

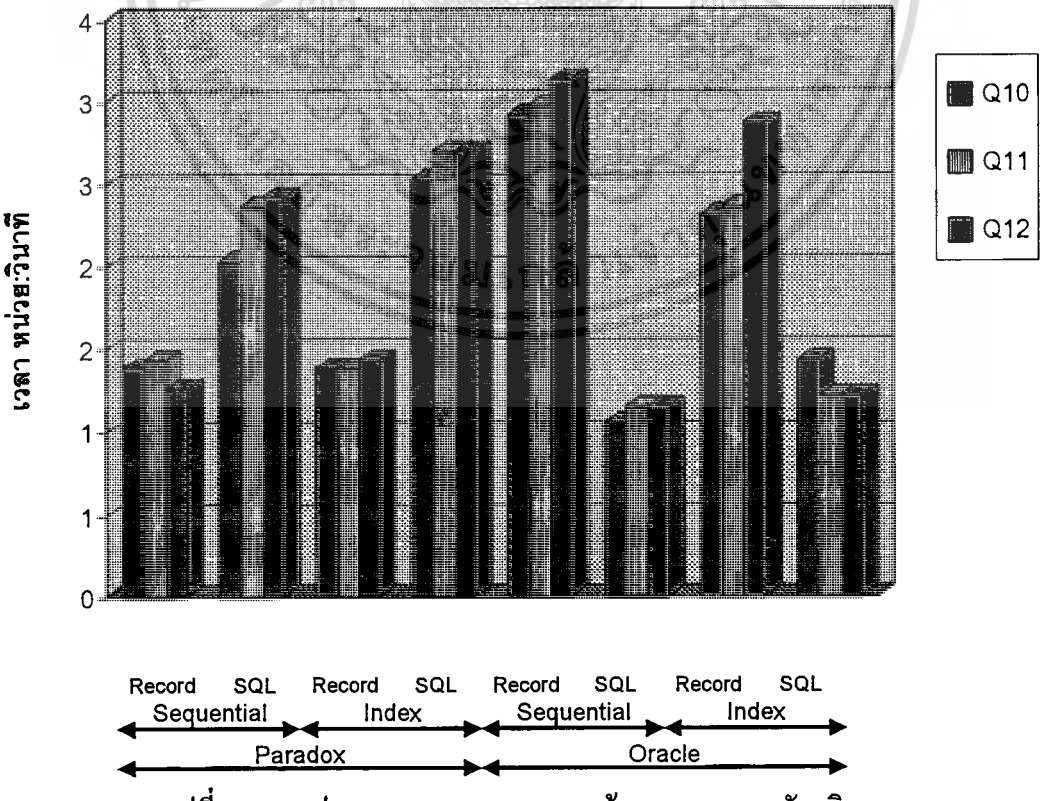


รูปที่ 4.8 กราฟแสดงการทดลองการเลือกแอททริบิวต์

จากกราฟในรูปที่ 4.8 จะเห็นว่าการสอบถามข้อมูล Q8 ในการเขียนโปรแกรมแบบอ่านข้อมูลครั้งละหนึ่งเรคอร์ด จะทำงานได้ช้าว่าการใช้คำสั่งเอสคิวแอล (SQL) การใช้งานอินเด็กไม่มีผลกับการประมวลผล การสอบถามข้อมูล Q9 การเขียนโปรแกรมแบบอ่านข้อมูลขึ้นมาครั้งละหนึ่งเรคอร์ด (Record-at-a-time) และเอสคิวเอล (SQL) สามารถประมวลผลได้รวดเร็วกว่าการสอบถามที่ Q8 ในทุกรูปแบบการประมวลผล และในการประมวลผลที่สามารถทำงานได้เร็วสุดเป็นการเขียนโปรแกรมด้วยคำสั่งเอสคิวแอล (SQL) ประมวลผลกับระบบฐานข้อมูลออราเคิล

4.7 การหาคำตอบจากข้อมูลจากหลาย ๆ ทัพเพิล (Aggregate)

การหาคำตอบจากข้อมูลจากหลาย ๆ ทัพเพิลเป็นการนำข้อมูลนั้นมาหาตัวเลขสรุป (ตย. การหาค่าของแอททริบิวต์ที่น้อยที่สุด) ในการสอบถามแต่ละแบบจะมีรูปแบบการเขียนโปรแกรมที่แตกต่างกันไป ประสิทธิภาพการทำงานก็ขึ้นอยู่กับอัลกอริทึมที่ใช้งาน และมีการใช้งานอินเด็ก ถ้าระบบฐานข้อมูลไม่ใช้งานอินเด็ก รูปแบบการเขียนโปรแกรมแบบครั้งละหนึ่งเรคอร์ด จะต้องอ่านข้อมูลทุกทัพเพิลขึ้นมาทำการประมวลผล แต่ถ้าระบบฐานข้อมูลนั้นมีการใช้งานอินเด็กก็สามารถ



รูปที่ 4.9 กราฟแสดงการทดลองการหาข้อมูลจากหลายทัพเพิล

ทำงานได้เร็วขึ้น และในบางการสอบถามสามารถหาคำตอบได้โดยการอ่านข้อมูลขึ้นมาเพียงครั้งเดียว ตัวอย่างเช่นการหาค่าน้อยที่สุดการนำหาคำตอบจากข้อมูลหลาย ๆ ทัพเพิล มีการสอบถาม 3 แบบดังนี้

Q10) หาค่าที่น้อยที่สุดในแอททริบิวต์ Salary จากรีเลชั่น Instructor

Q11) หาค่าที่น้อยที่สุดในแอททริบิวต์ Salary จากรีเลชั่น Instructor โดยมีเงื่อนไขแอททริบิวต์ Department เท่ากับ 50

Q12) หาผลรวมของแอททริบิวต์ Salary จากรีเลชั่น Instructor โดยมีเงื่อนไขแอททริบิวต์ Department เท่ากับ 50

จากกราฟในรูปที่ 4.9 จะเห็นว่าในระบบฐานข้อมูลพาราด็อกซ์การเขียนโปรแกรมแบบอ่านข้อมูลขึ้นมาครั้งละหนึ่งเรคอร์ด (record-at-a-time) สามารถทำงานได้เร็วกว่าการใช้คำสั่งเอสคิวแอล (SQL) ส่วนในระบบฐานข้อมูลออราเคิล การเขียนโปรแกรมด้วยคำสั่งเอสคิวแอลจะทำงานได้เร็วกว่าการเขียนโปรแกรมแบบอ่านข้อมูลขึ้นมาครั้งหนึ่งเรคอร์ด

4.8 การปรับปรุงฐานข้อมูล

การปรับปรุงฐานข้อมูลจะมีการประมวลผลอยู่ 3 แบบคือ การเพิ่ม แก้ไข และ ลบข้อมูล โดยในการปรับปรุงฐานข้อมูลนี้จะทำการศึกษาอยู่ 2 ประการคือ

1) เวลาที่ต้องใช้ในการปรับปรุงรีเลชั่นและอินเด็กที่มีอยู่
2) เวลาที่สูญเสียไปเพื่อใช้ในการควบคุมการทำงานในสภาวะพร้อมกัน และฟื้นสภาพข้อมูล

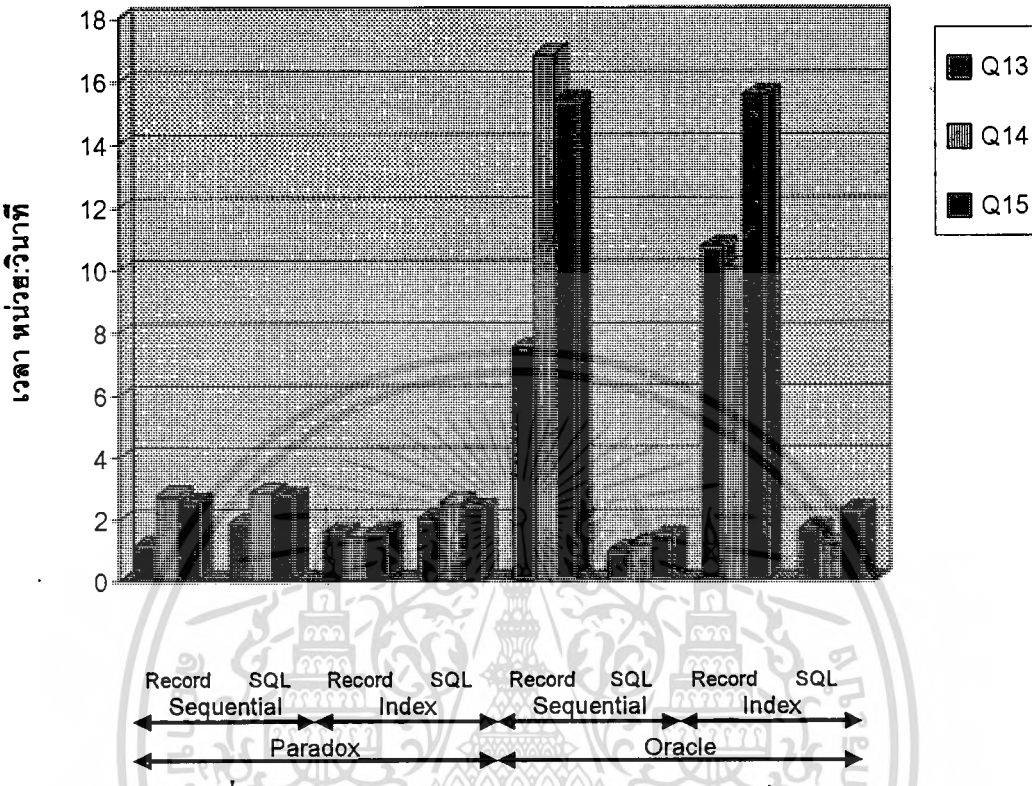
โดยเวลาที่สูญเสียไป เหล่านั้นจะวัดโดยการทดลองการปรับปรุงข้อมูล 3 แบบดังต่อไปนี้

Q13) เพิ่มข้อมูล 1 ทัพเพิลในรีเลชั่น Student

Q14) แก้ไขข้อมูล 1 ทัพเพิลในรีเลชั่น Student

Q15) ลบข้อมูล 1 ทัพเพิลในรีเลชั่น Sudent

จากกราฟในรูปที่ 4.10 ในการปรับปรุงฐานข้อมูลการเขียนโปรแกรมด้วยคำสั่งเอสคิวแอล (SQL) จะสามารถทำงานได้เร็วกว่าการเขียนโปรแกรมแบบอ่านข้อมูลขึ้นมาครั้งละหนึ่งเรคอร์ด เมื่อใช้ฐานข้อมูลที่เป็นออราเคิล และฐานข้อมูลที่เป็นพาราด็อกซ์เมื่อไม่ใช้งานอินเด็กจะใช้เวลาประมวลผลที่ใกล้เคียงกัน แต่ถ้าใช้งานอินเด็กการเขียนโปรแกรมอ่านข้อมูลครั้งละหนึ่งเรคอร์ดจะสามารถทำงานได้ดีกว่า

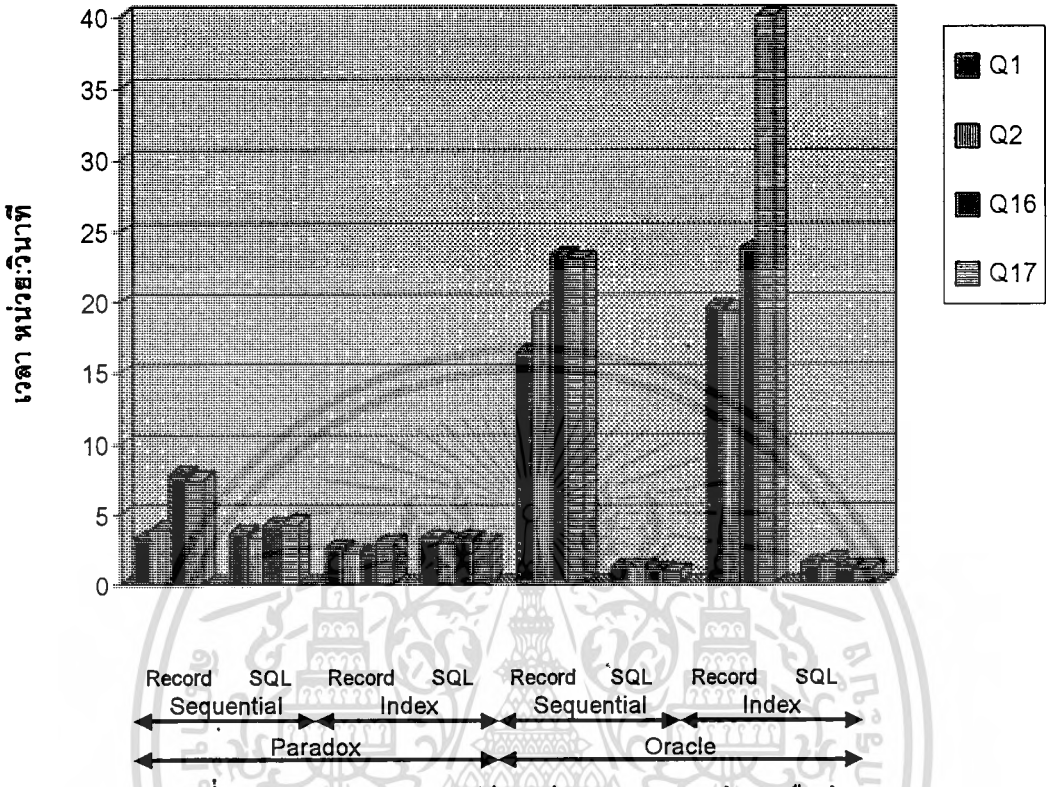


รูปที่ 4.10 กราฟแสดงการทดลองการปรับปรุงฐานข้อมูล

4.9 การประมวลผลผ่านเครือข่าย

ในการทดลองเพื่อเปรียบเทียบการประมวลผลข้อมูลผ่านเครื่อข่าย และประมวลผลภายในเครื่องเดียวกัน จะใช้การสอบถามที่ 1 และ 2 เป็นเลือกทัพเพิลข้อมูลในหัวข้อ 4.4 โดยใน Q1 และ Q2 เป็นการประมวลผลภายในเครื่องเดียวกัน และใน Q16 และ Q17 เป็นการประมวลผลผ่านเครือข่าย โดยใน Q1 และ Q16 เป็นการใช้การสอบถามที่ 1 ส่วนใน Q2 และ Q17 เป็นการใช้การสอบถามที่ 2

จากกราฟในรูปที่ 4.11 ในการประมวลผลข้อมูลโดยใช้ระบบฐานข้อมูลพาราด็อกซ์จะให้ผลการทดลองที่ไม่ค่อยชัดเจนเท่าไร เวลาที่ได้จะมีค่าใกล้เคียงกันมาก แต่ในระบบฐานข้อมูลออราเคิลจะเห็นในการเขียน โปรแกรมแบบอ่านข้อมูลครั้งละหนึ่งเรคอร์ด มีสภาพแวดล้อมแบบการทำงานภายในเครื่องกันจะ ช้าเวลามากกว่าการประมวลข้อมูลที่มีสภาพแวดล้อมแบบผ่านเครือข่าย ในขณะที่การประมวลผลโดยใช้คำสั่ง SQL ใช้เวลาใกล้เคียงกันระหว่างการประมวลผลในสภาพแวดล้อมบนเครื่องโดยใช้เวลามากกว่าการประมวลผลด้วยคำสั่งเอสคิวแอล (SQL)



รูปที่ 4.11 กราฟแสดงการเปรียบเทียบการทำงานผ่านเครือข่าย

4.10 สรุป

จากการทดลองทั้งหมดที่ทำการศึกษามา จะเห็นได้ว่าการเขียนโปรแกรมประมวลผลข้อมูล แบบอ่านข้อมูลขึ้นมาครั้งละหนึ่งเรคอร์ด ในการประมวลผลเพื่อทำการสอบถามข้อมูล เมื่อไม่ได้ใช้งานอินเด็กแล้ว ประสิทธิภาพในค้นหาข้อมูลจะช้าลง โดยเฉพาะการ Join ข้อมูลจากหลายตาราง แต่เมื่อมีการใช้งานอินเด็กประสิทธิภาพจะดีขึ้นมาก แต่ในการบางกรณีการใช้งานอินเด็กอาจ ช้ากว่าได้ในกรณีที่มีการใช้งานอินเด็กเป็นจำนวนมากต้องเสียเวลาเปิดแฟ้มอินเด็ก การที่แฟ้มข้อมูลที่มีการใช้งานอินเด็กทำงานได้เร็วกว่าก็เนื่องมาจากว่าได้นำประโยชน์จากการที่ข้อมูลได้ทำการเรียงลำดับไว้แล้ว และใช้อัลกอริทึกที่ดีสามารถทำงานได้เร็วเข้ามาช่วย แต่ถ้าเปรียบเทียบกับการเขียนโปรแกรมโดยใช้คำสั่ง SQL ประมวลผลข้อมูล เมื่อใช้ระบบฐานข้อมูลชนิดเดียวกัน และทำงานในสภาพแวดล้อมแบบใช้งานบนเครื่องเดียว ก็จะมีประสิทธิภาพในการประมวลผลใกล้เคียงกัน แต่ถ้าทำงานผ่านเครือข่าย การทำงานโดยใช้คำสั่ง SQL บนระบบฐานข้อมูลที่เป็น ระบบจัดการฐานข้อมูล (DBMS) อย่างเช่น ออราเคิล (ORACLE) ก็จะสามารถทำงานได้ประสิทธิภาพดีกว่า การประมวลผลบนฐานข้อมูลธรรมดาที่เป็น File-Base

บทที่ 5

การทดลองการทำงาน

การทำสอบการทำงาน จะทดลองสภาวะการทำงานพร้อมกัน และการทดลองการบันทึกข้อมูล โดยในการทดลองสภาวะการทำงานพร้อมกันในการเขียนโปรแกรมเพื่ออ่านข้อมูลครั้งละหนึ่งเรคอร์ดในต้องใช้เทคนิคการล็อกเข้ามาช่วย ซึ่งในการล็อกนี้ก็ทำได้โดยการล็อกจากโปรแกรมคอมพิวเตอร์ลูกค่าย (Client) ส่วนในการใช้คำสั่ง SQL ก็ไม่สามารถที่ทำการล็อกข้อมูลจากตัวโปรแกรมคอมพิวเตอร์ลูกค่ายได้ ต้องไปทำการล็อกบนโปรแกรมคอมพิวเตอร์แม่ค่าย (Server) โดยที่ระบบจัดการฐานข้อมูล (DBMS) เป็นตัวทำหน้าที่นี้

การทดลองการบันทึกข้อมูล จะทำการศึกษาว่าการบันทึกข้อมูลของระบบฐานข้อมูลต่าง ๆ ทำกันอย่างไรในช่วงไหน โดยในการทดลองการทำงาน จะการเขียนภาษา Delphi ในการเขียนโปรแกรมโดยมีระบบฐานข้อมูลอยู่ 4 แบบ ใช้ในการทดลองดังนี้

- ฐานข้อมูลพาราด็อกซ์ (Paradox Database)
- ฐานข้อมูลไมโครซอฟท์แอ็คเสซ (Microsoft Access Database)
- ระบบจัดการฐานข้อมูลไมโครซอฟท์เอสคิวแอลเซิร์ฟเวอร์
  (MSSQL DBMS : Microseft SQL Server Database Management System)
- ระบบจัดการฐานข้อมูลออราเคิล
  (ORACLE DBMS : Oracle Database Management System)

5.1 การทดลองการใช้งานคำสั่งล็อก

การใช้งานของคำสั่งล็อก จะทำการทดลองโดยการจองล็อกไว้ก่อน หลังจากนั้นก็จะมีอีกโปรแกรมหนึ่งเข้าไปทำการขอล็อกบ้าง เมื่อกำหนดให้ P เป็นโปรแกรมทำการจองล็อกการไว้ก่อน Q เป็นโปรแกรมเข้ามาขอการใช้งานล็อกทีหลัง และ S แทนด้วย Share Lock และ X แทนด้วย Exclusive Lock ในการทดลองจะทำดังนี้

- จองการล็อกเป็นแบบ Share และเข้าไปขอการล็อกแบบ Share
- จองการล็อกเป็นแบบ Share และเข้าไปขอการล็อกแบบ Exclusive
- จองการล็อกเป็นแบบ Exclusive และเข้าไปขอการล็อกแบบ Share
- จองการล็อกเป็นแบบ Exclusive และเข้าไปขอการล็อกแบบ Exclusive

โดยการประมวลผลแต่ละระบบฐานข้อมูลได้ผลการทดลองดังนี้

1) ระบบฐานข้อมูลพาราด็อกซ์ (Paradox)

เมื่อมีการทดลองตามวิธีข้างต้นทั้ง 4 แบบจะได้ผลลัพธ์คือ เมื่อโปรแกรม P มีการจองการล็อกไว้แบบ Share โปรแกรม Q จะสามารถเข้ามาขอการล็อกแบบ Share ได้เช่นกัน แต่ไม่สามารถขอการล็อกแบบ Exclusive ได้ และเมื่อโปรแกรม P มีการจองการล็อกไว้เป็นแบบ Exclusive โปรแกรม Q จะไม่สามารถที่จะเข้ามาขอการล็อกได้เลยไม่ว่าจะเป็นแบบ Share หรือ Exclusive เมื่อไม่สามารถทำการขอล็อกได้จะขึ้นข้อความว่า "File is locked" เพื่อให้แน่ใจว่าการจองการล็อกสามารถทำการล็อกข้อมูลได้จริง จึงให้โปรแกรม P ทำการจองล็อกไว้ แล้วให้อีกโปรแกรมทำการพิ่มข้อมูลในแฟ้มข้อมูล ปรากฏว่าไม่สามารถเพิ่มข้อมูลได้ และก็ขึ้นข้อความ "File is locked" เช่นกัน

2) ระบบฐานข้อมูลแอ็คแซส (MSACCESS)

ไม่ว่าจะให้โปรแกรม P มีการจองการล็อกไว้แบบ Share หรือ Exclusive โปรแกรม Q จะสามารถเข้ามาขอการล็อกได้ และขอการล็อกได้ทั้งแบบ Share และ Exclusive และเพื่อให้แน่ใจว่าการล็อกไม่สามารถล็อกข้อมูลได้ ก็ทำการแก้ไขข้อมูลในระหว่างที่มีจองการล็อกข้อมูลไว้ ปรากฏว่าสามารถแก้ไขข้อมูลได้ทั้งทีมีการจองล็อกแบบ Share และ Exclusive

3) ระบบฐานข้อมูลไมโครซอฟเอสคิวแอลเซริฟเวอร์ (MSSQL Server)

ไม่ว่าจะให้โปรแกรม P มีการจองการล็อกไว้แบบ Share หรือ Exclusive โปรแกรม Q จะสามารถเข้ามาขอการล็อกได้ และขอการล็อกได้ทั้งแบบ Share และ Exclusive และเพื่อให้แน่ใจว่าการล็อกไม่สามารถล็อกข้อมูลได้ ก็ทำการแก้ไขข้อมูลในระหว่างที่มีจองการล็อกข้อมูลไว้ ปรากฏว่าไม่สามารถแก้ไขข้อมูล ต้องรอจนกว่าให้มีจะมีการปลดล็อก จึงจะสามารถแก้ไขข้อมูลได้ ได้ทั้งทีมีการจองล็อกแบบ Share และ Exclusive เมื่อไปดูในโปรแกรมช่วยทำงานของ MSSQL Server ปรากฏว่าเป็นการ Select ข้อมูลขึ้นมาแล้วทำการล็อกข้อมูล

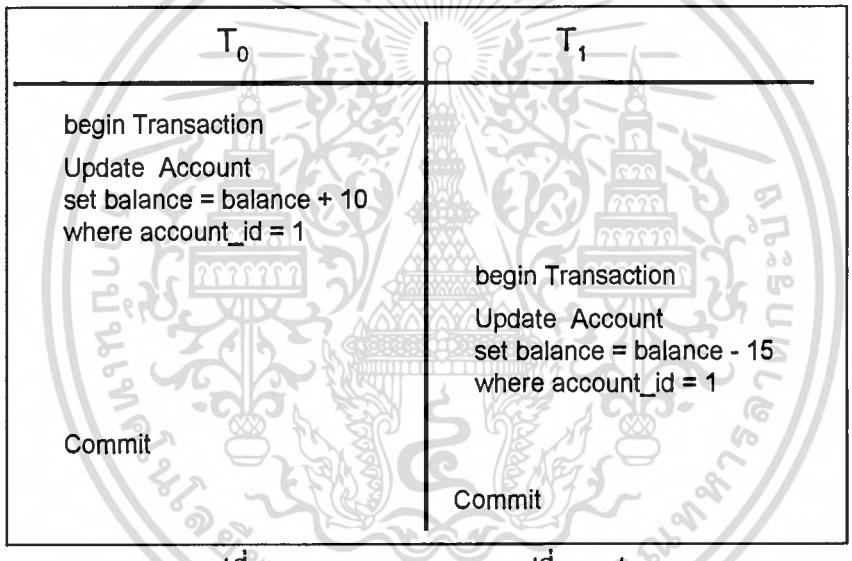
4) ระบบฐานข้อมูลออราเคิล (ORACLE)

เมื่อมีการทดลองระบบฐานข้อมูลออราเคิล โดยให้โปรแกรม P มีการจองการล็อกไว้แบบ Share หรือ Exclusive โปรแกรม Q จะไม่สามารถเข้ามาขอการล็อกได้เลยไม่ว่าเป็นแบบ Share หรือ Exclusive และเมื่อไม่สามารถทำการขอล็อกได้จะขึ้นข้อความว่า "Record locked by another user" และเมื่อทำการแก้ไขข้อมูล ในระหว่างที่มีจองการล็อกอยู่ ไม่สามารถที่แก้ไขได้ ไม่ว่าจะเป็นการจองการล็อกแบบ Share หรือ Exclusive จะข้อความ "Record locked by another user" เหมือนกัน

5.2 การทดลองระดับการขัดแย้งข้อมูล

ในการทดลองภาวะการทำงานพร้อมกัน เป็นการศึกษาว่าระบบฐานข้อมูลแต่ละแบบ มีประสิทธิภาพในการจัดการข้อมูลเมื่อมีผู้ใช้งานหลายคนปรับปรุงข้อมูลพร้อมกัน เมื่อกำหนดระดับการยอมรับความผิดพลาดของข้อมูล (Isolation) ให้สามารถแก้ปัญหาความขัดแย้งของการทำงาน (Inconsistent) โดยให้ระบบจัดการฐานข้อมูล (DBMS) เป็นตัวควบคุมสภาวะการทำงานพร้อมกัน

ในการทดลองนี้ใช้คำสั่ง SQL ในการประมวลผล โดยก่อนที่ใช้คำสั่ง SQL จะมีการใช้คำสั่ง Begin Transaction และหลังคำสั่ง SQL จะมีการ Commit เพื่อให้การทำงานทั้งหมดเป็น 1 รายการเปลี่ยนแปลง (Transaction) โดยมีลำดับการทำงานดังนี้



รูปที่ 5.1 การทดลองรายการเปลี่ยนแปลง

จากรูปที่ 5.1 ในการทำงานตามทฤษฎีที่กล่าวมาแล้วในบทที่ 3 จะเห็นได้ในการทำงานนั้น เมื่อ T_0 ทำงานคำสั่ง Update เสร็จแล้ว แต่ยังไม่ทัน Commit รายการเปลี่ยนแปลง T_1 ก็เข้ามาแก้ไขข้อมูลในทัพเพิลเดียวกับ T_0 นั้นจะไม่สามารถที่จะทำการแก้ไขได้ต้องรอให้ T_0 ทำการ Commit ก่อน แล้ว T_1 จึงจะสามารถทำการแก้ไขแล้วก็ Commit ได้ โดยให้ Balance เริ่มมีค่าเท่า 100 บาท เมื่อประมวลผลเสร็จแล้วในรายการเปลี่ยนแปลง T_0 Balance เท่ากับ 110 และในรายการเปลี่ยนแปลง T_1 Balance เท่ากับ 95 จึงจะเป็นคำตอบที่ถูกต้อง ในการทดลองกันใน 4 ระบบฐานข้อมูลดังที่กล่าวมาแล้วในตอนต้นบท และได้ผลดังนี้

1) ระบบฐานข้อมูลพาราด็อกซ์ (Paradox) และไมโครซอฟแอ็คแซส (MSAccess) จะได้ผลการทดลองเหมือนกันดังนี้

ในการประมวลผลเมื่อรายการเปลี่ยนแปลง T_0 มีการปรับปรุงข้อมูล แล้วยังไม่ Commit โดยให้ T_1 เข้ามาแก้ไขข้อมูลในทัพเพิลเดียวกันได้ สามารถทำการแก้ไขได้ และเมื่อ T_0 ทำการ Commit แล้วอ่านค่า Balance ขึ้นมาได้เท่ากับ 95 และเมื่อ T_0 ทำการ Commit แล้วอ่านค่า Balance ขึ้นมาก็ได้เท่ากับ 95 จะเห็นว่าค่าของ T_0 ที่อ่านได้หลังจาก Commit ไม่ถูกต้อง

2) ระบบฐานข้อมูลไมโครซอฟเอสคิวแอลเซิร์ฟเวอร์ (MSSQL Server) และออราเคิล (ORACLE) จะได้ผลการทดลองดังนี้

ในการประมวลผลเมื่อรายการเปลี่ยนแปลง T_0 มีการปรับปรุงข้อมูลแล้วยังไม่ Commit โดยให้ T_1 เข้ามาแก้ไขข้อมูลในทัพเพิลเดียวกันได้ จะไม่สามารถทำการแก้ไขได้จะถูกรอจนกว่ารายการเปลี่ยนแปลง T_0 ทำการ Commit ก่อน รายการเปลี่ยนแปลง T_1 จึงสามารถทำการแก้ไขปรับปรุงข้อมูลได้ และ Balance ที่อ่านขึ้นมาหลังจากที่แต่ละรายการเปลี่ยนแปลงทำการ Commit ได้ผลถูกต้องคือ T_0 มีค่าเท่ากับ 110 และ T_0 มีค่าเท่ากับ 95

5.3 การทดลองการบันทึกข้อมูล

ในการทดลองการบันทึกข้อมูลนี้ จะทำการทดลองเปรียบเทียบการทำงานโดยการใช้คำสั่ง Flush และไม่ได้ใช่คำสั่ง Flush

5.3.1 การเพิ่มข้อมูลแล้วใช้คำสั่ง Flush

การใช้คำสั่ง Flush เป็นคำสั่งที่ให้นำข้อมูลที่อยู่ใน Buffer ทำการบันทึกลงบนแถบแม่เหล็ก (Disk) คำสั่งนี้เป็นคำสั่งที่นิยมใช้ในการเขียนโปรแกรมแบบ Record-at-a-time เพื่อป้องกันข้อมูลสูญเสียเสียหายในระหว่างที่ยังทำการประมวลผลยังไม่เสร็จ โดยจะทำการทดลองจะทำเพิ่มข้อมูล 1,000 ทัพเพิล แล้วใช้คำสั่ง Flush หลังจากการเพิ่มข้อมูลทุกทัพเพิล เมื่อเสร็จแล้วทำการเปิดเครื่อง แล้วเปิดขึ้นมาดูใหม่ว่าข้อมูลที่ทำการเพิ่มได้ถูกปรับปรุงหรือไม่

1) ระบบฐานข้อมูลพาราด็อกซ์ (Paradox) และไมโครซอฟแอ็คเซส (MSAccess)

สามารถใช้คำสั่ง Flush ได้และเมื่อทำการเพิ่มข้อมูลแล้วทำการปิดเครื่อง เมื่อเปิดเครื่องขึ้นมาใหม่อีกครั้ง ข้อมูลที่ได้ทำการเพิ่มขึ้นมา ได้นำมาปรับปรุงฐานข้อมูล ในระหว่างการประมวลผลข้อมูลจะมีการอ่านเขียนข้อมูลในฮาร์ดดิส (Hard disk) ตลอดเวลา โดยสังเกตุจากไฟที่แสดงสถานะ

การทำงานของฮาร์ดดิส (Hard disk) จะติดอยู่ตลอดเวลา และมีเสียงของฮาร์ดดิสดัง อยู่ตลอดเวลา ในระหว่างการประมวลผล โดยที่ในขณะการประมวลผลไม่มีโปรแกรมอื่นทำงานอยู่เลย

2) ระบบฐานข้อมูลไมโครซอฟเอสคิวแอลเซิร์ฟเวอร์ (MSSQL Server) และออราเคิล (ORACLE)

ไม่สามารถใช้คำสั่ง Flush ได้ เพื่อใช้นำข้อมูลจาก Buffer บันทึกลงแผ่นแม่เหล็ก และขึ้นข้อความบอกการผิดพลาดว่า "Capability not supported"

5.3.2 การเพิ่มข้อมูลโดยไม่ใช้คำสั่ง Flush

จะทำการเพิ่มข้อมูลจำนวน 1,000 ทัพเพิลเมื่อทำการเพิ่มข้อมูลเสร็จแล้วก็ทำการปิดเครื่อง หลังจากนั้นเปิดเครื่องขึ้นมาใหม่เพื่อดูว่าข้อมูลที่เพิ่มได้ปรับปรุงได้ข้อมูลหรือไม่

1) ระบบฐานข้อมูลพาราด็อกซ์ (Paradox)

เมื่อเพิ่มข้อมูลเสร็จแล้วทำการปิดเครื่องแล้วเปิดขึ้นมาใหม่ปรากฏว่าข้อมูลที่เพิ่มขึ้นมาไม่ได้ไม่ได้นำไปปรับปรุงในฐานข้อมูล ข้อมูลยังมีอยู่เท่าเดิมก่อนการประมวลผล

2) ไมโครซอฟแอ็คเซส (MSAccess)

เมื่อเพิ่มข้อมูลเสร็จแล้วทำการปิดเครื่อง แล้วเปิดขึ้นมาใหม่ปรากฏว่าฐานข้อมูลที่เพิ่มขึ้นไม่สามารถที่อ่านข้อมูลได้

3) ระบบฐานข้อมูลไมโครซอฟเอสคิวแอลเซิร์ฟเวอร์ (MSSQL Server) และออราเคิล (ORACLE)

เมื่อทำการเพิ่มข้อมูล แล้วทำการปิดเครื่อง เมื่อเปิดเครื่องขึ้นมาใหม่อีกครั้ง ปรากฏว่าข้อมูลที่ได้ทำการเพิ่มขึ้นมาได้นำมาปรับปรุงฐานข้อมูล แสดงว่าในการปรับปรุงฐานข้อมูลแต่ละครั้งจะมีการบันทึกข้อมูลอยู่ตลอดเวลา

บทที่ 6
สรุปผลการศึกษา

6.1 สรุปผลการศึกษา

การเขียนโปรแกรมโดยใช้ภาษายุคที่3 เพื่อประมวลผลข้อมูลในระบบฐานข้อมูลเชิงสัมพันธ์แล้ว ในการกรณีที่ประมวลผลเป็นแบบผ่านระบบเครือข่ายนั้น รูปแบบที่เหมาะสม ควรใช้การเขียนโปรแกรมโดยใช้คำสั่ง SQL แล้วทำงานกับระบบฐานข้อมูลที่เป็น ระบบจัดการฐานข้อมูล (DBMS) เช่น ออราเคิล เป็นต้น เนื่องสามารถทำประมวลข้อมูลได้รวดเร็ว แม้ว่าในการประมวลผลจะต้องผ่านระบบเครือข่าย ก็ทำให้ประสิทธิภาพในการประมวลผลลดลงไม่มาก สามารถควบคุมการทำงานในสภาวะพร้อมกันได้อย่างมีประสิทธิภาพ โดยที่ระบบฐานข้อมูลธรรมดาไม่สามารถควบคุมภาวะการทำงานพร้อมกันได้ และถ้าใช้การทำงานโดยอ่านข้อมูลครั้งละหนึ่งเรคอร์ด ในการประมวลผลได้ช้ากว่า ในการควบคุมการทำงานพร้อมกันก็ไม่สามารถทำได้อย่างมีประสิทธิภาพ

ส่วนในการทำงานแบบภายในเครื่องเดียว การประมวลผลข้อมูลแบบอ่านข้อมูลครั้งละหนึ่งเรคอร์ด และทำงานกับระบบฐานข้อมูลธรรมดา (File Base) เช่น พาราด็อกซ์ (Paradox) หรือไมโครซอฟแอ็คเซส (MSAccess) เป็นต้น โดยมีการใช้งานอินเด็ก และใช้อัลกอริทึมที่มีประสิทธิภาพ เนื่องจากการทำงานภายในเครื่องเดียว ไม่ต้องควบคุมสภาวะการทำงานพร้อมกัน และระยะเวลาในการประมวลผลสอบถามข้อมูลก็ไม่ได้ช้ากว่าการใช้คำสั่ง SQL จนยอมรับไม่ได้ และในบางรูปแบบอาจดีกว่าด้วยตัวอย่างเช่น การหาคำตอบจากข้อมูลหลายทัพเพิล (Aggregate)

6.2 ข้อเสนอแนะ

ในการศึกษารูปแบบการเขียนโปรแกรมด้วยภาษายุคที่ 3 ในสภาพแวดล้อมฐานข้อมูลเชิงสัมพันธ์ ในการทดลองเวลาที่ใช้ในการประมวลผลในการศึกษาครั้งนี้จะทำงานแบบผู้ใช้งานคนเดียว (Single User) ซึ่งการทำงานในสภาวะจริงนั้น จะมีการทำงานแบบมีผู้ใช้งานหลาคน (Multi User) ด้วย เวลาที่ใช้ก็แตกต่างกันออกไป การพิจารณาใช้อัลกอริทึมที่ดี และมีประสิทธิภาพมากขึ้น ซึ่งมีผลกับความเร็วทั้งนั้น นอกจากนี้ยังมีการประมวลผลข้อมูลโดยมีการใช้งาน Store Procedure นอกจากนี้แล้วในการทดลองต้องมีการควบคุมไม่ให้มีข้อมูลค้างอยู่ในบัฟเฟอร์มิเช่นนั้นจะทำให้ผล การทดลองออกมาผิดพลาดเนื่องจากไม่สามารถควบคุมข้อมูลอยู่ในบัฟเฟอร์มีมากน้อยเพียงไร

บรรณานุกรม

ดวงแก้ว สวามิภักดิ์. ระบบฐานข้อมูล. กรุงเทพฯ:ซีเอ็ดยูเคชั่น.

Annon et al. A Measure of Transaction Processing Power. Datamation,1985.

Date, C.J. An Introduction to Database System Vol. 6th edition. Massachusetts:Addison-Wesley:1995.

Gray, J. The Benchmark Handbook. Ca:Morgan-Kaufman,1991.

Gray, J. and Reuter, Andreas. Transaction Processing:Concepts and Techniques. Ca:Morgan-Kaufman,1993.

Helman, Paul. The Science of Database Management. Richard D. Irwin Inc,1994.

Korth, Henry F. and Silberschatz , Abraham. Database System Concept. Newyork:McGraw-Hill,1986.

Stonebraker, Michael. Readings in Database System: A Restrospective on the Winsconsin Benchmark approach. California:Morgan Kaufman, 1988.

ภาคผนวก ก.

ผลการทดลองเวลาที่ใช้ในการประมวลผล

ทดลองครั้งที่	แบบสอบถาม	ระบบฐานข้อมูล	โครงสร้างข้อมูล	รูปแบบการเขียนโปรแกรม	ที่อยู่ของฐานข้อมูล	เวลา hh:mm:ss:ms
1	Q1	Paradox	Sequential	Record-at-a-time	Front-End	00:00:03:180
2	Q1	Paradox	Sequential	SQL	Front-End	00:00:03:570
3	Q1	Paradox	Index	Record-at-a-time	Front-End	00:00:02:480
4	Q1	Paradox	Index	SQL	Front-End	00:00:03:190
5	Q1	Oracle	Sequential	Record-at-a-time	Front-End	00:00:16:370
6	Q1	Oracle	Sequential	SQL	Front-End	00:00:01:150
7	Q1	Oracle	Index	Record-at-a-time	Front-End	00:00:19:490
8	Q1	Oracle	Index	SQL	Front-End	00:00:01:370
9	Q2	Paradox	Sequential	Record-at-a-time	Front-End	00:00:03:900
10	Q2	Paradox	Sequential	SQL	Front-End	00:00:03:300
11	Q2	Paradox	Index	Record-at-a-time	Front-End	00:00:02:200
12	Q2	Paradox	Index	SQL	Front-End	00:00:03:030
13	Q2	Oracle	Sequential	Record-at-a-time	Front-End	00:00:19:340
14	Q2	Oracle	Sequential	SQL	Front-End	00:00:01:210
15	Q2	Oracle	Index	Record-at-a-time	Front-End	00:00:19:340
16	Q2	Oracle	Index	SQL	Front-End	00:00:01:590
17	Q3	Paradox	Sequential	Record-at-a-time	Front-End	00:00:04:230
18	Q3	Paradox	Sequential	SQL	Front-End	00:00:03:620
19	Q3	Paradox	Index	Record-at-a-time	Front-End	00:00:03:240
20	Q3	Paradox	Index	SQL	Front-End	00:00:03:840
21	Q3	Oracle	Sequential	Record-at-a-time	Front-End	00:00:19:440
22	Q3	Oracle	Sequential	SQL	Front-End	00:00:01:150

ทดลองครั้งที่	แบบสอบถาม	ระบบฐานข้อมูล	โครงสร้างข้อมูล	รูปแบบการเขียนโปรแกรม	ที่อยู่ของฐานข้อมูล	เวลา hh:mm:ss:ms
23	Q3	Oracle	Index	Record-at-a-time	Front-End	00:00:24:440
24	Q3	Oracle	Index	SQL	Front-End	00:00:01:260
25	Q4	Paradox	Sequential	Record-at-a-time	Front-End	00:00:04:290
26	Q4	Oracle	Sequential	Record-at-a-time	Front-End	00:00:19:440
27	Q5	Paradox	Sequential	Record-at-a-time	Front-End	32:43:51:550
28	Q5	Paradox	Sequential	SQL	Front-End	00:00:18:290
29	Q5	Paradox	Index	Record-at-a-time	Front-End	00:01:18:100
30	Q5	Paradox	Index	SQL	Front-End	00:00:21:970
31	Q5	Oracle	Sequential	Record-at-a-time	Front-End	80:57:19:586
32	Q5	Oracle	Sequential	SQL	Front-End	00:00:20:100
33	Q5	Oracle	Index	Record-at-a-time	Front-End	00:03:18:400
34	Q5	Oracle	Index	SQL	Front-End	00:00:01:760
35	Q6	Paradox	Sequential	SQL	Front-End	00:01:53:640
36	Q6	Paradox	Index	Record-at-a-time	Front-End	00:05:28:780
37	Q6	Paradox	Index	SQL	Front-End	00:00:32:900
38	Q6	Oracle	Sequential	SQL	Front-End	00:00:49:650
39	Q6	Oracle	Index	Record-at-a-time	Front-End	00:08:07:520
40	Q6	Oracle	Index	SQL	Front-End	00:00:03:190
41	Q7	Paradox	Sequential	SQL	Front-End	00:00:07:530
42	Q7	Paradox	Index	Record-at-a-time	Front-End	00:00:30:160
43	Q7	Paradox	Index	SQL	Front-End	00:00:09:010
44	Q7	Oracle	Sequential	SQL	Front-End	00:00:12:250
45	Q7	Oracle	Index	Record-at-a-time	Front-End	00:02:09:020
46	Q7	Oracle	Index	SQL	Front-End	00:00:01:810
47	Q8	Paradox	Sequential	Record-at-a-time	Front-End	00:01:01:070
48	Q8	Paradox	Sequential	SQL	Front-End	00:00:04:230
49	Q8	Paradox	Index	Record-at-a-time	Front-End	00:01:05:280

ทดลองครั้งที่	แบบสอบถาม	ระบบฐานข้อมูล	โครงสร้างข้อมูล	รูปแบบการเขียนโปรแกรม	ที่อยู่ของฐานข้อมูล	เวลา hh:mm:ss:ms
50	Q8	Paradox	Index	SQL	Front-End	00:00:04:390
51	Q8	Oracle	Sequential	Record-at-a-time	Front-End	00:01:23:480
52	Q8	Oracle	Sequential	SQL	Front-End	00:00:01:590
53	Q8	Oracle	Index	Record-at-a-time	Front-End	00:01:25:130
54	Q8	Oracle	Index	SQL	Front-End	00:00:01:970
55	Q9	Paradox	Sequential	Record-at-a-time	Front-End	00:00:13:510
56	Q9	Paradox	Sequential	SQL	Front-End	00:00:03:790
57	Q9	Paradox	Index	Record-at-a-time	Front-End	00:00:13:890
58	Q9	Paradox	Index	SQL	Front-End	00:00:03:790
59	Q9	Oracle	Sequential	Record-at-a-time	Front-End	00:00:28:950
60	Q9	Oracle	Sequential	SQL	Front-End	00:00:00:880
61	Q9	Oracle	Index	Record-at-a-time	Front-End	00:01:27:440
62	Q9	Oracle	Index	SQL	Front-End	00:00:01:100
63	Q10	Paradox	Sequential	Record-at-a-time	Front-End	00:00:01:380
64	Q10	Paradox	Sequential	SQL	Front-End	00:00:02:030
65	Q10	Paradox	Index	Record-at-a-time	Front-End	00:00:01:380
66	Q10	Paradox	Index	SQL	Front-End	00:00:02:530
67	Q10	Oracle	Sequential	Record-at-a-time	Front-End	00:00:02:920
68	Q10	Oracle	Sequential	SQL	Front-End	00:00:01:050
69	Q10	Oracle	Index	Record-at-a-time	Front-End	00:00:02:310
70	Q10	Oracle	Index	SQL	Front-End	00:00:01:430
71	Q11	Paradox	Sequential	Record-at-a-time	Front-End	00:00:01:430
72	Q11	Paradox	Sequential	SQL	Front-End	00:00:02:360
73	Q11	Paradox	Index	Record-at-a-time	Front-End	00:00:01:380
74	Q11	Paradox	Index	SQL	Front-End	00:00:02:690
75	Q11	Oracle	Sequential	Record-at-a-time	Front-End	00:00:02:970
76	Q11	Oracle	Sequential	SQL	Front-End	00:00:01:150

	แบบสอบถาม	ระบบฐานข้อมูล	โครงสร้างข้อมูล	รูปแบบการเขียนโปรแกรม	ที่อยู่ของฐานข้อมูล	เวลา hh:mm:ss:ms
77	Q11	Oracle	Index	Record-at-a-time	Front-End	00:00:02:360
78	Q11	Oracle	Index	SQL	Front-End	00:00:01:210
79	Q12	Paradox	Sequential	Record-at-a-time	Front-End	00:00:01:260
80	Q12	Paradox	Sequential	SQL	Front-End	00:00:02:410
81	Q12	Paradox	Index	Record-at-a-time	Front-End	00:00:01:430
82	Q12	Paradox	Index	SQL	Front-End	00:00:02:690
83	Q12	Oracle	Sequential	Record-at-a-time	Front-End	00:00:03:130
84	Q12	Oracle	Sequential	SQL	Front-End	00:00:01:150
85	Q12	Oracle	Index	Record-at-a-time	Front-End	00:00:02:860
86	Q12	Oracle	Index	SQL	Front-End	00:00:01:210
87	Q13	Paradox	Sequential	Record-at-a-time	Front-End	00:00:01:150
88	Q13	Paradox	Sequential	SQL	Front-End	00:00:01:870
89	Q13	Paradox	Index	Record-at-a-time	Front-End	00:00:01:540
90	Q13	Paradox	Index	SQL	Front-End	00:00:01:980
91	Q13	Oracle	Sequential	Record-at-a-time	Front-End	00:00:07:470
92	Q13	Oracle	Sequential	SQL	Front-End	00:00:00:940
93	Q13	Oracle	Index	Record-at-a-time	Front-End	00:00:10:710
94	Q13	Oracle	Index	SQL	Front-End	00:00:00:880
95	Q14	Paradox	Sequential	Record-at-a-time	Front-End	00:00:02:750
96	Q14	Paradox	Sequential	SQL	Front-End	00:00:02:850
97	Q14	Paradox	Index	Record-at-a-time	Front-End	00:00:01:420
98	Q14	Paradox	Index	SQL	Front-End	00:00:02:470
99	Q14	Oracle	Sequential	Record-at-a-time	Front-End	00:00:16:810
100	Q14	Oracle	Sequential	SQL	Front-End	00:00:01:150
101	Q14	Oracle	Index	Record-at-a-time	Front-End	00:00:10:060
102	Q14	Oracle	Index	SQL	Front-End	00:00:02:260
103	Q15	Paradox	Sequential	Record-at-a-time	Front-End	00:00:02:530

	แบบสอบถาม	ระบบฐานข้อมูล	โครงสร้างข้อมูล	รูปแบบการเขียนโปรแกรม	ที่อยู่ของฐานข้อมูล	เวลา hh:mm:ss:ms
104	Q15	Paradox	Sequential	SQL	Front-End	00:00:02:690
105	Q15	Paradox	Index	Record-at-a-time	Front-End	00:00:01:540
106	Q15	Paradox	Index	SQL	Front-End	00:00:02:310
107	Q15	Oracle	Sequential	Record-at-a-time	Front-End	00:00:15:320
108	Q15	Oracle	Sequential	SQL	Front-End	00:00:01:430
109	Q15	Oracle	Index	Record-at-a-time	Front-End	00:00:15:550
110	Q15	Oracle	Index	SQL	Front-End	00:00:02:260
111	Q16	Paradox	Sequential	Record-at-a-time	Server	00:00:07:690
112	Q16	Paradox	Sequential	SQL	Server	00:00:04:120
113	Q16	Paradox	Index	Record-at-a-time	Server	00:00:02:040
114	Q16	Paradox	Index	SQL	Server	00:00:03:130
115	Q16	Oracle	Sequential	Record-at-a-time	Server	00:00:23:240
116	Q16	Oracle	Sequential	SQL	Server	00:00:00:980
117	Q16	Oracle	Index	Record-at-a-time	Server	00:00:23:670
118	Q16	Oracle	Index	SQL	Server	00:00:01:100
119	Q17	Paradox	Sequential	Record-at-a-time	Server	00:00:07:410
120	Q17	Paradox	Sequential	SQL	Server	00:00:04:280
121	Q17	Paradox	Index	Record-at-a-time	Server	00:00:02:910
122	Q17	Paradox	Index	SQL	Server	00:00:02:970
123	Q17	Oracle	Sequential	Record-at-a-time	Server	00:00:23:010
124	Q17	Oracle	Sequential	SQL	Server	00:00:00:770
125	Q17	Oracle	Index	Record-at-a-time	Server	00:00:39:880
126	Q17	Oracle	Index	SQL	Server	00:00:01:030

ประวัติผู้เขียน

ชื่อผู้เขียน	นายชลัท เขตประเสริฐกุล
วันเดือนปีเกิด	8 กรกฎาคม พ.ศ. 2515
สถานที่เกิด	จังหวัดกาญจนบุรี
วุฒิการศึกษาระดับปริญญาตรี	วท.บ.(วิทยาการคอมพิวเตอร์)
สถานที่สำเร็จการศึกษา	คณะวิทยาศาสตร์ มหาวิทยาลัยหอการค้าไทย
ปีการศึกษาที่สำเร็จ	ปีการศึกษา 2536
ประสพการณ์การทำงาน	ตำแหน่งโปรแกรมเมอร์ ธนาคารกสิกรไทย จำกัด(มหาชน) (1 พ.ย. 2536- 30 มิ.ย. 2537) ตำแหน่งโปรแกรเมอร์ บริษัทเงินทุนหลักทรัพย์ ธนชาติ จำกัด(มหาชน) (1 ก.ค. 2537 - 15 ธ.ค. 2539)