ระบบฐานข้อมูลแบบไคล์เอ็นต์เซอร์ฟเวอร์
CLIENT - SERVER DATABASE SYSTEM

โดย

นาย วีระ งามลักษมีแข 36013172

นาย ศิริพงษ์ เจียมวิจิตรกุล 36013175

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญา
วิศวกรรมศาสตร์
สาขาวิชาวิศวกรรมคอมพิวเตอร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2538

ปริญญานิพนธ์ปีการศึกษา 2538
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
เรื่อง ระบบฐานข้อมูลแบบไคล์เอ็นต์เซอร์ฟเวอร์

ผู้จัดทำ

1. นาย วีระ งามลักษมีแข 36013172
2. นาย ศิริพงษ์ เจียมวิจิตรกุล 36013175

..... อาจารย์ที่ปรึกษา
(ดร. บุญธีร์ เครือตราชู)

ระบบฐานข้อมูลแบบไคล์เอ็นต์เซอร์ฟเวอร์

วีระ งามลักษมีแข
ศิริพงษ์ เจียมวิจิตรกุล
ดร. บุญธีร์ เครือตราชู อาจารย์ที่ปรึกษา
ปีการศึกษา 2538

บทคัดย่อ

ในการทำงานเกี่ยวกับระบบฐานข้อมูลแบบไคล์เอ็นต์เซอร์ฟเวอร์ การที่จะตัดสินใจใช้โครงสร้างของตารางในฐานข้อมูลเชิงสัมพันธ์ในแต่ละแบบจะมีผลกระทบต่อเวลาที่ใช้ในการทำงานของแต่ละคำสั่งที่กระทำกับข้อมูล และการที่เซอร์ฟเวอร์แต่ละตัวสามารถที่จะติดต่อกับไคล์เอ็นต์ได้พร้อมกันหลายๆตัว ซึ่งจำนวนของไคล์เอ็นต์ที่ต่ออยู่กับเซอร์ฟเวอร์ จะมีผลกระทบต่อเวลาที่เซอร์ฟเวอร์จะตอบสนองกลับไปที่ไคล์เอ็นต์แต่ละตัว ปริญญานิพนธ์นี้ มุ่งเน้นที่จะศึกษาพฤติกรรมของระบบไคล์เอ็นต์เซอร์ฟเวอร์ที่มีไคล์เอ็นต์จำนวนมากต่ออยู่กับเซอร์ฟเวอร์ และวัดประสิทธิภาพในการตอบสนองของเซอร์ฟเวอร์ต่อไคล์เอ็นต์ในการทำงานแต่ละคำสั่งบนโครงสร้างของตารางในฐานข้อมูลเชิงสัมพันธ์ที่แตกต่างกัน

CLIENT / SERVER DATABASE SYSTEM

Vira Ngramluxsameekhae
Siripong Jiamwijitkul
Dr. Boontee Kruatrachue Advisor
1995

Abstract

In Client / Server Database, discussion use structure of table on relational database is impact to time used in data operation and amount of client with connect to server is impact to server response time. This thesis aims at investigating performance of Client / Server database system of server response time to each client working data's operation of database base on difference structure of table in relational database system.

สารบัญ

บทที่ 1
บทนำ

ระบบไคล์เอ็นต์เซอร์ฟเวอร์ ที่มีการเชื่อมต่อไคล์เอ็นต์หลายๆตัวเข้ากับเซอร์ฟเวอร์จะทำให้ผลตอบสนองของเซอร์ฟเวอร์ต่อไคล์เอ็นต์แต่ละตัวจะใช้เวลาที่แตกต่างกันไปตามจำนวนของไคล์เอ็นต์ ที่ต่ออยู่กับเซอร์ฟเวอร์แต่ละตัว ส่วนโครงสร้างของตารางในฐานข้อมูลเชิงสัมพันธ์ก็จะมีผลต่อเวลาที่ใช้ในการทำงานของแต่ละคำสั่งที่กระทำกับข้อมูลในตารางในปริญญานิพนธ์นี้มุ่งเน้นที่จะศึกษาผลกระทบของจำนวนไคล์เอ็นต์ที่มีต่อระบบไคล์เอ็นต์เซอร์ฟเวอร์และยังศึกษาถึงผลกระทบในการทำงานของแต่ละคำสั่งที่กระทำกับข้อมูลที่อยู่ในโครงสร้างในแต่ละแบบของตารางในฐานข้อมูลเชิงสัมพันธ์และทำการศึกษาโปรแกรมเดลไฟ(Delphi) และ เพาเวอร์บิวเดอร์(PowerBuilder) ซึ่งเป็นโปรแกรมที่ใช้ในการพัฒนาแอพพลิเคชันที่มีการใช้งานเกี่ยวกับระบบฐานข้อมูล โดยจะใช้โปรแกรมทั้งสองในการสร้างแอพพลิเคชั่นเพื่อใช้ในการทดสอบผลกระทบของการใช้งานระบบไคล์เอ็นต์เซอร์ฟเวอร์ที่ได้กล่าวในตอนต้น

บทที่ 2
ทฤษฎีที่ใช้ในการพัฒนาระบบฐานข้อมูล

2.1 โมเดลเชิงสัมพันธ์ (Relational model)

โมเดลเชิงสัมพันธ์เป็นโมเดลที่ใช้ในการอธิบายความสัมพันธ์ของข้อมูลที่ถูกเก็บด้วยระบบจัดการฐานข้อมูลเชิงสัมพันธ์ (Relational DataBase Management System : RDBMS) ซึ่งเป็นผลงานของ ดร. คอดด์ (Codd) ที่ได้เสนอผลงานวิจัยให้ชาวโลกรู้จักในปี พ.ศ. 2513 โดยมีการนำไปใช้งานกับเครื่องระดับตั้งแต่เมนเฟรมลงไปจนถึงเครื่องระดับไมโครคอมพิวเตอร์ด้วย และก็เป็นที่ยอมรับกันแล้วว่า บรรดาผู้ใช้ระบบฐานข้อมูล (โดยเฉพาะผู้ที่ทำงานด้วยเครื่องระดับมินิคอมพิวเตอร์และระดับไมโครคอมพิวเตอร์) จะมีความคุ้นเคยกับโมเดลเชิงสัมพันธ์นี้มากกว่าอีก 2 โมเดล คือ โมเดลเชิงแตกสาขา (Hierarchical model) และโมเดลเชิงโครงข่าย (Network model) ที่มีมาก่อนหน้านี้

นอกเหนือจากความแพร่หลายของโมเดลเชิงสัมพันธ์นี้แล้ว ข้อดีของโมเดลเชิงสัมพันธ์ที่มีมากกว่าอีก 2 โมเดล ดังนี้

1. โมเดลเชิงสัมพันธ์เป็นโมเดลที่สามารถสร้างความเข้าใจได้ง่ายกว่า เพราะภาพลักษณ์ของข้อมูลที่เก็บโดยโมเดลเชิงสัมพันธ์จะมาจากมุมมองของผู้ใช้ ซึ่งจะมีความซับซ้อนน้อยกว่าภาพลักษณ์ของข้อมูลที่เก็บโดยอีก 2 โมเดล
2. ระบบส่วนใหญ่ที่ใช้โมเดลเชิงสัมพันธ์นี้มักจะมีเครื่องมือที่ช่วยให้ผู้ใช้สามารถจัดการกับข้อมูลที่เก็บอยู่ได้ง่ายกว่าข้อมูลที่จัดเก็บด้วยโมเดลแบบอื่น
3. โมเดลเชิงสัมพันธ์นี้มีเครื่องมือที่ช่วยให้ผู้ใช้สามารถค้นพบปัญหาที่เกิดขึ้นในการออกแบบระบบฐานข้อมูลได้โดยง่าย และยังง่ายในการแก้ไขการออกแบบที่ผิดพลาดนั้นด้วย
4. โมเดลเชิงสัมพันธ์เป็นโมเดลที่มีความสอดคล้องกับหลักการของฐานข้อมูล ผู้ใช้ไม่ต้องพะวงกับรายละเอียดของการจัดเก็บข้อมูลเหมือนกับการจัดข้อมูลของโมเดลอื่น
5. ภาษาที่ใช้ในการจัดการกับข้อมูลที่จัดเก็บด้วยระบบจัดการฐานข้อมูลเชิงสัมพันธ์ ภาษา เอสคิวแอล (SQL:Structure Query Language) เป็นภาษาแบบ เชิงกลุ่ม (set oriented) ซึ่งจะต่างกับภาษาที่ใช้ในการจัดการกับข้อมูลที่จัดเก็บด้วยระบบจัดการฐานข้อมูลของโมเดลอื่นที่เป็นภาษาแบบ เชิงเรคคอร์ด (record-at-a-time)

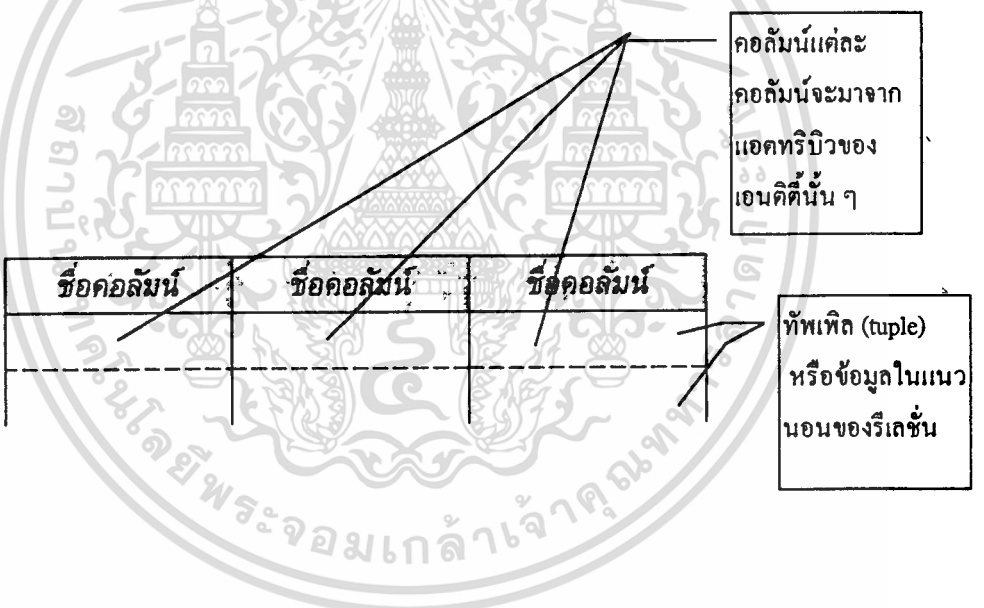
แม้ว่าโมเดลเชิงสัมพันธ์จะมีข้อดีหลายประการดังได้กล่าวไปแล้ว แต่ในปัจจุบันก็ยังมีจุดอ่อนที่มีการอ้างอิงถึงเสมอ คือ ระบบจัดการฐานข้อมูลแบบโมเดลเชิงสัมพันธ์นี้มักจะมีประสิทธิภาพในการใช้งานสู้อีก 2 โมเดลไม่ได้ โดยเฉพาะในการประยุกต์ใช้งานขององค์กรขนาดใหญ่ จุดอ่อนนี้ก็ได้มีการแย้งกลับมาในแง่ที่ว่า โมเดลเชิงสัมพันธ์เป็นโมเดลที่มีอายุการพัฒนาน้อยกว่าอีก 2 โมเดล จึงเป็นไปได้ว่า การพัฒนาที่ผ่านมาของโมเดลเชิงสัมพันธ์ก็ยังมีจำนวนระดับขั้นที่ได้พัฒนาไปแล้วน้อยกว่าอีก 2 โมเดล ดังนั้นหากต้องการเปรียบเทียบการทำงานระหว่างโมเดลเชิงสัมพันธ์กับโมเดลอื่นก็ควรที่จะทำการเปรียบเทียบที่ระดับจำนวนขั้นการพัฒนาที่เท่ากันจึงจะสมเหตุสมผล

2.1.1 ศัพท์เฉพาะของโมเดลเชิงสัมพันธ์

ในหัวข้อนี้จะกล่าวถึงโมเดลเชิงสัมพันธ์ โดยกำหนดนิยามและกล่าวถึงคำศัพท์ต่าง ๆ ที่เกี่ยวข้องกับโมเดลนี้

จากการที่ข้อมูลที่เก็บด้วยโมเดลเชิงสัมพันธ์จะถูกเก็บไว้ในตารางที่จะถูกเรียกว่า *"รีเลชั่น"* โดยที่รีเลชั่นทุกรีเลชั่นจะอยู่ในรูปของตาราง แต่ตารางบางตารางอาจไม่เป็นรีเลชั่นก็ได้ ดังนี้ตารางที่มีลักษณะเป็นรีเลชั่นจะต้องมีคุณลักษณะดังนี้

1. แต่ละช่องของตารางจะบรรจุข้อมูลได้เพียงค่าเดียว
2. ชื่อหัวข้อในแต่ละคอลัมน์มีความแตกต่างกัน อันเป็นชื่อของแอตทริบิวของเอนติตี้
3. ค่าข้อมูลที่อยู่ในแต่ละคอลัมน์ คือ ค่าของแอตทริบิวตามที่ระบุหัวข้อไว้ที่หัวของคอลัมน์นั้นๆ
4. การเรียงลำดับคอลัมน์ไม่ถือว่ามีความสำคัญ
5. ข้อมูลแต่ละแถวจะต้องแตกต่างกัน
6. การเรียงลำดับแถวไม่ถือว่ามีความสำคัญ



ตารางที่มีคุณลักษณะดังกล่าวจะเรียกว่า *รีเลชั่น*

ดังนั้น เราจะได้นิยามของ*ฐานข้อมูลเชิงสัมพันธ์* คือ ฐานข้อมูลที่เกิดจากการรวบรวมรีเลชั่นต่าง ๆ ที่มีความสัมพันธ์ (relationship) ระหว่างกัน

เราจะเรียกข้อมูลแต่ละแถวในแนวนอนของรีเลชั่นว่า *ทัพเพิล (tuple)* และเรียกข้อมูลแต่ละแถวในแนวตั้งหรือแนวคอลัมน์ว่า *แอตทริบิว (attribute)* โดยที่คำว่า *คีย์ (key)* จะหมายถึงข้อมูลที่เกิดจากแอตทริบิว 1 ตัวหรือหลายตัวก็ได้

แต่ละรีเลชั่นจะต้องมีสิ่งที่เรียกว่า *คีย์หลัก (primary key)* คือ ข้อมูลของแอตทริบิว 1 ตัวหรือมากกว่า 1 ตัวก็ได้ที่สามารถใช้เป็นตัวเจาะจงบอกเราได้ว่ากำลังอ้างอิงถึงข้อมูลทัพเพิลใด ส่วนคีย์ที่เป็นแอตทริบิวของรีเลชั่นอื่นที่ซ้ำกับแอตทริบิวที่เป็นคีย์หลักของรีเลชั่นหนึ่งจะเรียกว่า *คีย์นอก*

(foreign key) (เช่น รีเลชั่น A มีแอตทริบิวรหัสประจำตัวเป็นคีย์หลัก แล้วในรีเลชั่น B มีแอตทริบิวรหัสประจำตัวเช่นเดียวกับ A เราจะเรียก แอตทริบิวดังกล่าวของรีเลชั่น B ว่า เป็นคีย์นอกของคีย์หลักของรีเลชั่น A) ในกรณีที่รีเลชั่นมีแอตทริบิวหรือกลุ่มแอตทริบิวที่มีคุณสมบัติเป็นคีย์หลักได้อยู่หลายแอตทริบิว เราจะเรียกแอตทริบิวหรือกลุ่มแอตทริบิวที่มิได้ถูกเลือกให้เป็นคีย์หลักว่า *คีย์คู่แข่ง (candedate key)* หรือ *คีย์สำรอง (alternate key)* และแอตทริบิวอื่น ๆ ที่เหลือที่มิได้เป็นคีย์หลักและไม่ได้เป็นส่วนใดส่วนหนึ่งของคีย์หลักก็จะถูกเรียกว่า *นันคีย์ (nonkey attribute)*

คำว่า *โดเมน (domain)* จะหมายถึง กรอบของค่าต่าง ๆ ที่เป็นไปได้ เช่น โดเมนของแอตทริบิววันที่ก็จะหมายถึงค่าของวันที่ที่เป็นไปได้ คือ มีค่าเท่ากับ 1 ถึง 31 ในเดือนที่ลงท้ายด้วยคำว่า "คม", มีค่าเท่ากับ 1 ถึง 30 ในเดือนที่ลงท้ายด้วยคำว่า "ยน" และในเดือนกุมภาพันธ์อาจมีค่าเท่ากับ 1 ถึง 28 หรือ 29 ก็ได้ แต่ในการเก็บค่าข้อมูลลงในรีเลชั่นนั้น บางกรณีที่เรามีการกำหนดโดเมนให้กับแอตทริบิวแล้ว แต่ข้อมูลที่จะถูกเก็บเข้าไปอาจถูกบรรจุเข้าไปในภายหลัง ลักษณะนี้จะคำให้เกิด *ค่าว่าง (Null value)* ขึ้นชั่วขณะก่อนที่จะมีการบรรจุค่าข้อมูลที่อยู่ในโดเมนที่กำหนดไว้เข้าไป ดังนั้น คำว่า "ค่าว่าง" จึงหมายถึงค่าที่ยังมิทราบชัดว่าแอตทริบิวนั้นจะมีค่าเป็นค่าใด หรือ ค่าของข้อมูลที่ไม่อยู่ในโดเมนที่กำหนด โดยมีข้อบังคับไว้ว่า แอตทริบิวที่ทำหน้าที่เป็นคีย์หลักของรีเลชั่นจะมีค่าข้อมูลเป็นค่าว่างไม่ได้เสมอ เพราะจะทำให้การเข้าถึงข้อมูลในทัพเพิล (tuple) นั้นกระทำไม่ได้

เมื่อมีการจัดเก็บข้อมูลในฐานข้อมูลใด ๆ แล้ว ข้อมูลจะถูกแยกออกเป็นกลุ่มของข้อมูลเป็นชุดที่ประกอบด้วยแอตทริบิวต่าง ๆ ที่มีความสัมพันธ์กัน เช่น การเก็บข้อมูลของบุคลากรในโรงเรียนก็อาจแยกเก็บเป็นกลุ่มข้อมูลของนักเรียน, กลุ่มข้อมูลของครูอาจารย์, และกลุ่มข้อมูลของนักการภารโรง เป็นต้น กลุ่มข้อมูลแต่ละกลุ่มนี้จะเรียกว่า *เอนติตี้ (entity)* ซึ่งแต่ละเอนติตี้จะประกอบไปด้วยแอตทริบิวต่าง ๆ ที่มีความสัมพันธ์กัน เช่น เอนติตี้ของนักเรียนก็จะประกอบไปด้วยชื่อ, นามสกุล, ที่อยู่, ชั้นเรียน เป็นต้น

จากการแยกจัดเก็บข้อมูลออกเป็นเอนติตี้นี้ แต่ละเอนติตี้ก็จะมีความสัมพันธ์กัน ความสัมพันธ์ระหว่างเอนติตี้สามารถแบ่งออกเป็น 3 ชนิด คือ

- ความสัมพันธ์แบบหนึ่งต่อหนึ่ง (one to one)
- ความสัมพันธ์แบบหนึ่งต่อกลุ่ม (one to many)
- ความสัมพันธ์แบบกลุ่มต่อกลุ่ม (many to many)

ความสัมพันธ์แบบหนึ่งต่อหนึ่ง

ความสัมพันธ์แบบหนึ่งต่อหนึ่งระหว่างเอนติตี้ก็หมายความว่า เมื่อเอนติตี้หนึ่งมีข้อมูลของคีย์หลักค่าหนึ่ง ค่าข้อมูลดังกล่าวก็จะมีความสัมพันธ์กับค่าข้อมูลของคีย์หลักของอีกเอนติตี้หนึ่งเพียงค่าเดียวเท่านั้น เช่น หากเรากำหนดให้ความสัมพันธ์ระหว่างเอนติตี้นักเรียนกับเอนติตี้ผู้ปกครองเป็นแบบหนึ่งต่อหนึ่งแล้วก็จะหมายความว่า การที่เราอ้างถึงนักเรียนคนใดคนหนึ่งก็จะสามารถอ้างถึงผู้ปกครองได้เพียงคนเดียวเท่านั้น และในการตรงกันข้ามก็ต้องเป็นจริงด้วย คือ เมื่อเราอ้างถึงผู้ปกครองคนใดคนหนึ่งแล้วก็จะสามารถอ้างอิงถึงนักเรียนได้เพียงคนเดียวเท่านั้น

ชื่อนักเรียน	ชื่อผู้ปกครอง
A	a
B	b
C	c

ความสัมพันธ์แบบหนึ่งต่อกลุ่ม

ความสัมพันธ์แบบหนึ่งต่อกลุ่มระหว่างเอนติตี้ก็หมายความว่า เมื่อเอนติตี้หนึ่งมีข้อมูลของคีย์หลักค่าหนึ่ง ค่าข้อมูลดังกล่าวก็จะมีความสัมพันธ์กับค่าข้อมูลของคีย์หลักของอีกเอนติตี้หนึ่งได้หลายค่า เช่น หากเรากำหนดให้ความสัมพันธ์ระหว่างเอนติตี้นักเรียนกับเอนติตี้ผู้ปกครองเป็นแบบหนึ่งต่อกลุ่มแล้วก็จะหมายความว่า การที่เราอ้างถึงนักเรียนคนใดคนหนึ่งก็จะสามารถอ้างถึงผู้ปกครองได้หลายคน และในการตรงกันข้ามก็จะมีความหมายว่า เมื่อเราอ้างถึงผู้ปกครองคนใดคนหนึ่งแล้วก็จะสามารถอ้างอิงถึงนักเรียนได้เพียงคนเดียวเท่านั้น แต่ผู้ปกครองที่เราอ้างถึงเป็นคนละคนกันก็อาจจะอ้างอิงถึงนักเรียนคนเดียวกันก็เป็นได้

ชื่อนักเรียน	ชื่อผู้ปกครอง
A	a
B	a
C	c

ความสัมพันธ์แบบกลุ่มต่อกลุ่ม

ความสัมพันธ์แบบกลุ่มต่อกลุ่มระหว่างเอนติตี้ก็หมายความว่า ค่าข้อมูลของคีย์หลักของเอนติตี้หนึ่งที่ต่างกันอาจอ้างอิงถึงค่าข้อมูลของคีย์หลักของอีกเอนติตี้หนึ่งได้ค่าเดียวหรือ หลายค่าก็ได้ เช่น หากเรากำหนดให้ความสัมพันธ์ระหว่างเอนติตี้นักเรียนกับเอนติตี้ผู้ปกครองเป็นแบบกลุ่มต่อกลุ่มแล้วก็จะหมายความว่า การที่เราอ้างถึงนักเรียนคนหนึ่งหรือหลายคนก็จะสามารถอ้างอิงถึง ผู้ปกครองคนเดียวกันก็ได้ และในทางกลับกัน การที่เราอ้างถึงผู้ปกครอง คนหนึ่งหรือหลายคนก็จะสามารถอ้างอิงถึงนักเรียนคนเดียวกันก็ได้

ชื่อนักเรียน	ชื่อผู้ปกครอง
A	a
B	a
C	c
C	d

เมื่อเราได้แอตทริบิวต่าง ๆ มาประกอบกันเป็นเอนติตี้ และสามารถกำหนดความสัมพันธ์ระหว่างเอนติตี้ได้แล้ว ขั้นตอนต่อไปก็คือการเขียนภาพแสดงความสัมพันธ์ระหว่างเอนติตี้ต่าง ๆ ด้วยโมเดล ER (ER-model)

นอกจากนี้ เอนติตี้ยังมีเอนติตี้บางประเภทที่การอ้างถึงตัวมันได้อย่างสมบูรณ์นั้นจะต้องอ้างถึงเอนติตี้อื่นเสมอ เช่น หากระบบฐานข้อมูลของบุคลากรในโรงเรียนมีเอนติตี้ของลูกนักการภารโรงด้วยแล้ว การอ้างอิงถึงเอนติตี้นี้ได้จะต้องอ้างถึงเอนติตี้นักการภารโรงด้วยเสมอ เอนติตี้ประเภทนี้เราเรียกว่า *เอนติตี้ชนิดอ่อน (Weak entity)*

2.2 ทฤษฎีของระบบฐานข้อมูลเชิงสัมพันธ์

2.2.1 กฎของความคงสภาพ (Integrity rule)

กฎของความคงสภาพของโมเดลเชิงสัมพันธ์เป็นทฤษฎีที่ช่วยยืนยันความถูกต้องของความสัมพันธ์ระหว่างข้อมูลว่า รีเลชั่นใดที่เป็นไปตามกฎของความคงสภาพนี้แล้วย่อมจะมีความสัมพันธ์ระหว่างข้อมูลอย่างถูกต้องอยู่ตลอดเวลา ไม่ว่ารีเลชั่นนั้นจะมีการเปลี่ยนแปลงแก้ไขข้อมูลไปในรูปแบบใดก็ตาม

กฎของความคงสภาพมีความหมายอยู่ 2 ลักษณะ คือ *กฎความคงสภาพของเอนติตี้ (entity integrity rule)* และ *กฎความคงสภาพของการอ้างอิง (referential integrity rule)* ดังอธิบายได้ดังนี้

1. *กฎความคงสภาพของเอนติตี้* กล่าวว่า

"แอตทริบิวทุกตัวที่เป็นส่วนของคีย์หลักจะไม่อนุญาตให้มีค่าว่าง"

หมายความว่า คีย์หลักของทุกรีเลชั่นจะไม่สามารถเก็บค่าข้อมูลที่เป็นค่าว่างได้ เหตุผลของข้อกำหนดนี้ก็คือ เพื่อให้การเข้าถึงข้อมูลในทัพเพิลใด ๆ ของรีเลชั่นมีความเป็นไปได้เสมอ เพราะถ้าคีย์หลักของทัพเพิลใดมีค่าข้อมูลเป็นค่าว่างแล้ว ก็จะส่งผลให้การเข้าถึงข้อมูลในทัพเพิลนั้นไม่สามารถกระทำได้อย่างแน่นอน

2. *กฎความคงสภาพของการอ้างอิง* กล่าวว่า

"ถ้าเรามีรีเลชั่น R2 ซึ่งมี FK เป็นคีย์นอกที่อ้างอิงถึงคีย์หลัก PK ใน รีเลชั่น R1 สำหรับทุกค่าของ FK ใน R2 จะต้อง

ก. มีค่าเท่ากับค่า PK ในทัพเพิลใดทัพเพิลหนึ่งในรีเลชั่น R1

หรือ

ข. มีค่าของแอตทริบิวทุกตัวใน FK เป็นค่าว่าง"

หมายความว่า แอตทริบิวใด ๆ ที่เป็นคีย์หลักของรีเลชั่นหนึ่ง เมื่อมีการนำแอตทริบิวนั้นไปเป็นคีย์นอกของอีกรีเลชั่นหนึ่ง การเป็นคีย์นอกของแอตทริบิวนั้นจะต้องมีโดเมนเป็นโดเมนเดียวกันกับแอตทริบิวที่เป็นคีย์หลัก ทั้งนี้ ก็เพื่อให้การนำรีเลชั่นมาใช้งานร่วมกัน (การนำรีเลชั่นมา join กัน) กระทำได้อย่างถูกต้อง คือ ทุกแอตทริบิวที่เป็นคีย์นอกจะต้องมีข้อมูลซ้ำกับข้อมูลของแอตทริบิวที่เป็นคีย์หลักอย่างแน่นอน แต่อาจมีบางค่าข้อมูลของแอตทริบิวที่เป็นคีย์หลักเป็นข้อมูลไม่อยู่ในโดเมน

ของแอตทริบิวที่เป็นคีย์นอกก็ได้ นั่นคือ โดเมนของคีย์นอกจะต้องเล็กกว่าหรือเท่ากับโดเมนของคีย์หลักเสมอ

2.2.2 ฟังก์ชันการขึ้นต่อกัน (Functional dependency)

ฟังก์ชันการขึ้นต่อกันเป็นข้อกำหนดที่ช่วยให้เราเห็นถึงความสัมพันธ์ของแอตทริบิวต่าง ๆ ที่อยู่ในรีเลชั่น ทั้งเพราะแอตทริบิวต่าง ๆ ที่อยู่ในเอนติตี้เดียวกันก็เป็นไปได้ที่แอตทริบิวเหล่านั้นจะมีความสัมพันธ์กันเองโดยที่ความสัมพันธ์นี้อาจเกี่ยวข้องหรือไม่เกี่ยวกับความสัมพันธ์ที่มันมีต่อคีย์หลักของเอนติตี้นั้นก็เป็นได้ ซึ่งการที่แอตทริบิวเหล่านั้นมีความสัมพันธ์กันเองจะเป็นสิ่งที่เราต้องพิจารณาแยกออกเป็นรีเลชั่นย่อย ๆ เพราะแอตทริบิวของแต่ละรีเลชั่นก็ควรจะมีความสัมพันธ์กับคีย์หลักของรีเลชั่นของตนเองเท่านั้น

กำหนดรีเลชัน R ถ้ามีแอตทริบิว Y ของ R เป็นฟังก์ชันที่ขึ้นต่อแอตทริบิว X ของรีเลชั่น เราสามารถเขียนแทนได้ด้วยสัญลักษณ์

R.X - - -> R.Y

อ่านว่า R.X มีฟังก์ชันการขึ้นอยู่กับ R.Y

หรือ R.X มีฟังก์ชันในการเลือก R.Y

หรือ R.Y ขึ้นอยู่กับ R.X

นิยาม R.X มีฟังก์ชันการขึ้นอยู่กับ R.Y ก็ต่อเมื่อ ทุกค่าข้อมูลของแอตทริบิว X ใน R จะมีค่าข้อมูลของแอตทริบิว Y ใน R ได้เพียงค่าเดียวเสมอ โดยที่แอตทริบิว X และ Y อาจจะเป็นคีย์แบบรวม (Composite key) ก็ได้

รีเลชั่น R

X	Y
a	1
b	2
a	1
b	1

นิยาม R.X มีฟังก์ชันการขึ้นอยู่กับ R.Y อย่างเต็มที่ (R.Y fully functionally dependent on R.X) ก็ต่อเมื่อ R.Y มีฟังก์ชันการขึ้นอยู่กับ R.X และไม่ขึ้นอยู่กับข้อมูลเพียงบางส่วนของ R.X โดยที่แอตทริบิว X และ Y อาจจะเป็นคีย์แบบรวมก็ได้

รีเลชั่น R

X		Y
A	a	1
B	b	2
A	a	1
B	c	3

2.3 ระบบไคล์เอ็นต์เซอร์ฟเวอร์

ในปัจจุบันใช้งานไมโครคอมพิวเตอร์มีการพัฒนาก้าวหน้าอย่างรวดเร็ว ทำให้มีการใช้งานไมโครคอมพิวเตอร์อย่างกว้างขวาง และเพื่อให้การทำงานมีความคล่องตัวและแพร่หลายจึงมีการพัฒนาระบบคอมพิวเตอร์ทางด้านเครือข่ายหรือการเชื่อมโยงเครือข่าย เพื่อให้ผู้ใช้ในระดับส่วนบุคคลสามารถทำการติดต่อสื่อสารเพื่อเลือกเปลี่ยนข้อมูล ซึ่งจะเป็นการเพิ่มข้อมูลของการใช้งานระบบได้อีกมากมาย ดังนั้นแนวโน้มของเทคโนโลยีคอมพิวเตอร์ในปัจจุบันจึงเป็นการมุ่งสู่การสร้างระบบเครือข่ายและรูปแบบการใช้งานจะเป็นระบบไคล์เอ็นต์เซอร์ฟเวอร์ (Client/Server System)

ในอดีตการพัฒนาระบบงานคอมพิวเตอร์จะทำบนเครื่องเมนเฟรม ซึ่งเป็นเครื่องขนาดใหญ่ทำให้ประสบปัญหาเกี่ยวกับการขยายระบบงาน เพื่อให้ได้ผลตอบสนองการทำงานรวดเร็วขึ้น ซึ่งการขยายระบบงานของเครื่องเมนเฟรมจะมีค่าใช้จ่ายที่สูงมาก ดังนั้นจึงเป็นจุดเริ่มต้นของแนวความคิดที่จะใช้ระบบคอมพิวเตอร์ที่มีเครื่องคอมพิวเตอร์ขนาดเล็กๆหลายเครื่องทำงานรวมกันในลักษณะที่เรียกว่า ระบบไคล์เอ็นต์เซอร์ฟเวอร์

โครงสร้างระบบคอมพิวเตอร์ในอดีตมักจะวางแบบลำดับขั้น กล่าวคือ มีคอมพิวเตอร์เมนเฟรมขนาดใหญ่อยู่ยอดสุดและแตกกระจายมายังคอมพิวเตอร์ขนาดย่อมหรือมินิคอมพิวเตอร์และกระจายลงเทอร์มินัล

การเปลี่ยนแปลงโครงสร้างจากที่เป็นแบบเทอร์มินัลต่ออยู่กับเครื่องเมนเฟรมเข้าสู่ระบบที่มีประสิทธิภาพการใช้งานที่ดีกว่า ในราคาที่ถูกกว่า เช่น ต้องการให้เครื่องของผู้ใช้งานเป็นเครื่องคอมพิวเตอร์ส่วนบุคคล(Personal Computer) เพื่อทำงานแบบกราฟิก มีรูปแบบการใช้งานที่สวยงามและง่ายต่อการใช้งาน ดังนั้นเครื่องของผู้ใช้งานจึงเป็นเครื่องที่ทำงาน ตามระบบมาตรฐานบางอย่าง เช่น ระบบติดต่อกับผู้ใช้แบบกราฟฟิก (GUI:Graphic User Interface) ส่วนเครื่องหลักอาจเป็นเอสคิวแอล ดาต้าเบส ซึ่งจะเก็บฐานข้อมูลไว้ เพื่อให้เครื่องลูกเรียกใช้ได้

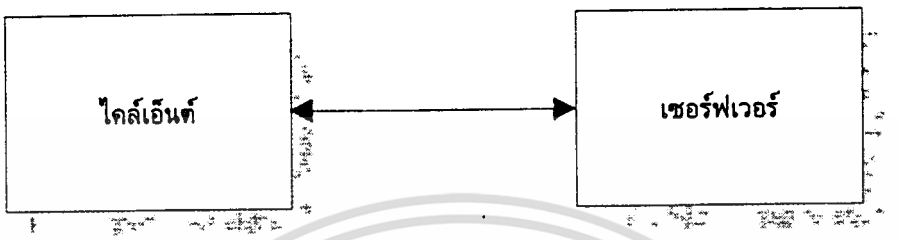
2.3.1 โครงสร้างของระบบไคล์เอ็นต์เซอร์ฟเวอร์

ในระบบไคล์เอ็นต์เซอร์ฟเวอร์จะประกอบด้วยเครื่องคอมพิวเตอร์หลายๆเครื่องต่อทำงานร่วมกัน ซึ่งจะสามารถแบ่งส่วนประกอบของระบบได้เป็น 2 ส่วน ดังนี้

2.3.1.1 ส่วนเซอร์ฟเวอร์ หมายถึง เครื่องคอมพิวเตอร์เครื่องหนึ่งในเครือข่ายที่ทำ หน้าที่เป็นศูนย์กลางการให้บริการทรัพยากรต่างๆของระบบ เช่น ถ้าให้บริการข้อมูลหรือโปรแกรมก็เรียกว่า

ไฟล์เซอร์ฟเวอร์ และถ้าให้บริการงานด้านการพิมพ์ก็เรียกว่า พรินเตอร์เซอร์ฟเวอร์ และถ้าทำหน้าที่เป็นตัวให้บริการการสื่อสารก็เรียกว่า คอมมูนิเคชั่นเซอร์ฟเวอร์

2.3.1.2. ส่วนไคล์เอ็นต์ หมายถึง เครื่องคอมพิวเตอร์ที่อยู่ในเครือข่าย ไม่ได้ทำหน้าที่ให้บริการ แต่จะเป็นตัวขอใช้บริการจากเซอร์ฟเวอร์ ซึ่งจะแสดงดังรูปที่ 1

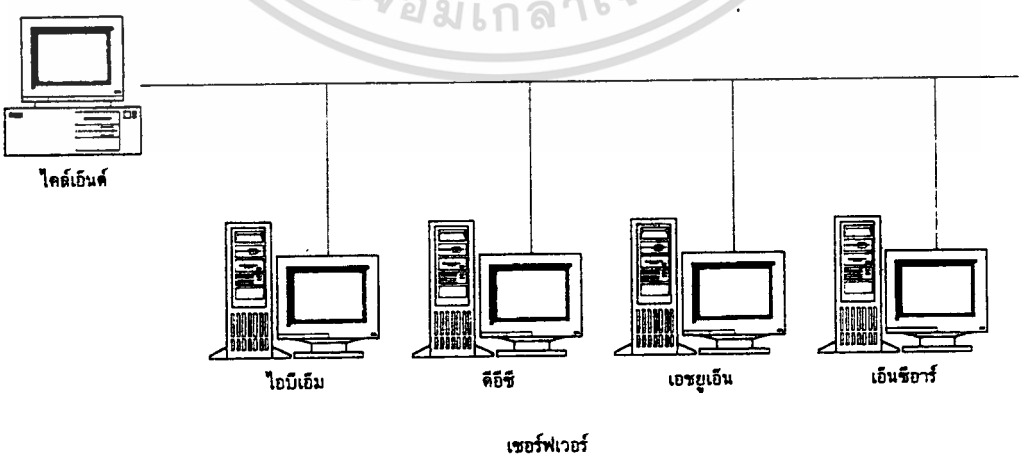


รูปที่ 1 โมเดลไคล์เอ็นต์เซอร์ฟเวอร์ จะมีคอมพิวเตอร์เครื่องหนึ่งขอใช้บริการและอีกเครื่องให้บริการ

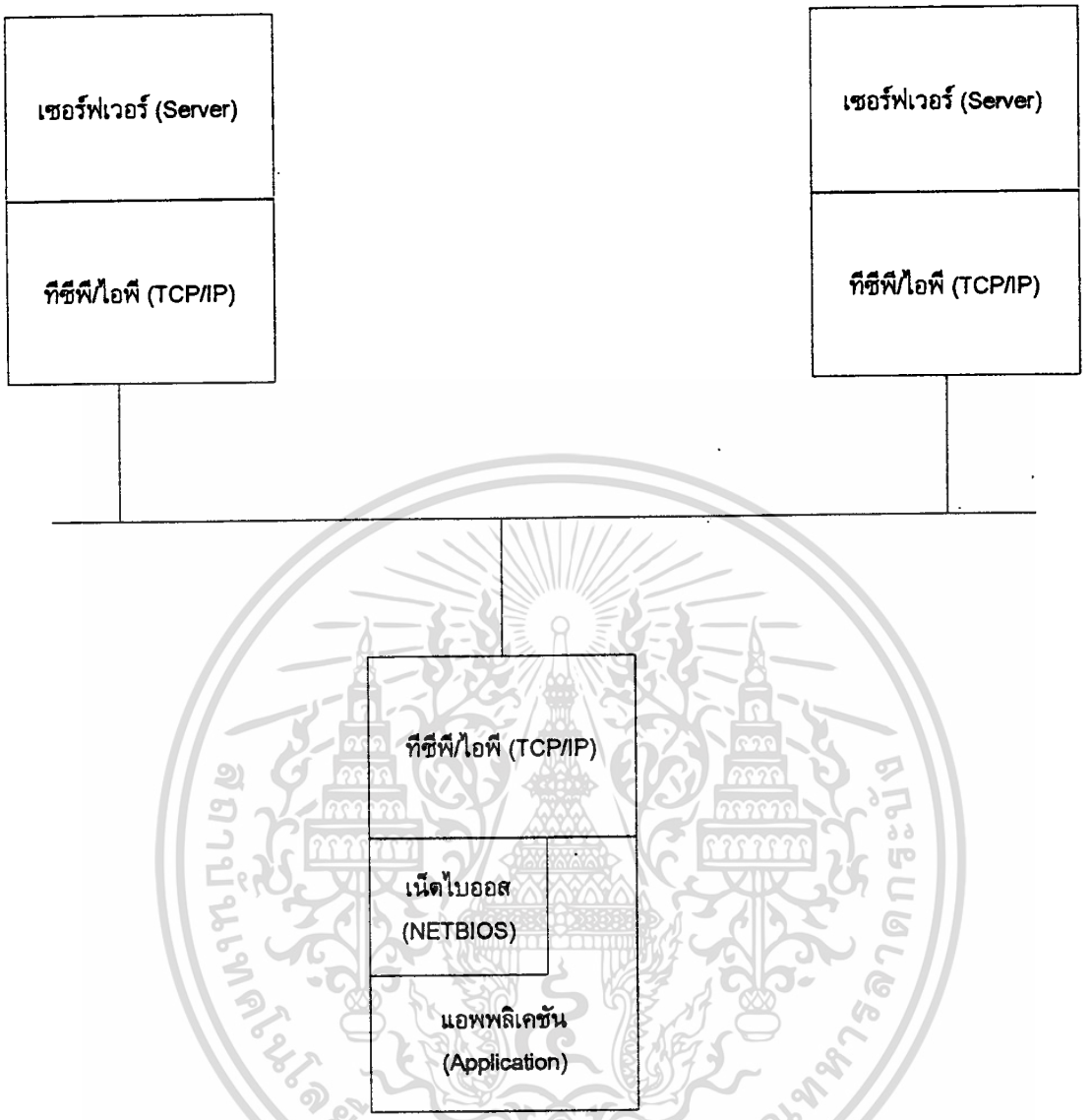
2.3.2 แนวความคิดในการนำเครือข่ายมาใช้กับระบบคอมพิวเตอร์

จากการที่ระบบเครือข่ายมีการพัฒนาก้าวหน้าขึ้น จึงทำให้มีการพัฒนาซอฟต์แวร์ที่สามารถเชื่อมโยงคอมพิวเตอร์หลาย ๆยี่ห้อเข้าไประบบเดียวกันได้ ซึ่งจะทำให้การใช้ทรัพยากรต่าง ๆในระบบเกิดประสิทธิภาพสูงสุด โดยเครื่องไคล์เอ็นต์สามารถเลือกใช้ข้อมูล เลือกใช้โปรแกรมหรือเลือกใช้ทรัพยากรอื่นจากเซอร์ฟเวอร์หลาย ๆยี่ห้อได้ ดังแสดงในรูปที่ 2

การที่ระบบเครือข่ายจะมีการติดต่อสื่อสารอย่างมีประสิทธิภาพได้จำเป็นจะต้องมีมาตรฐานการเชื่อมโยง เช่น TCP/IP IPX/SPX DecNet และจำเป็นจะต้องมีโปรแกรมจัดระบบงานในข่ายเครือข่ายที่เรียกว่า ระบบปฏิบัติการเครือข่าย (NOS : Network Operating System) พิจารณาการทำงานของโมเดลดังรูปที่ 3



รูปที่ 2 เครื่องคอมพิวเตอร์ CLIENT สามารถไปใช้ SERVER ตัวใดก็ได้



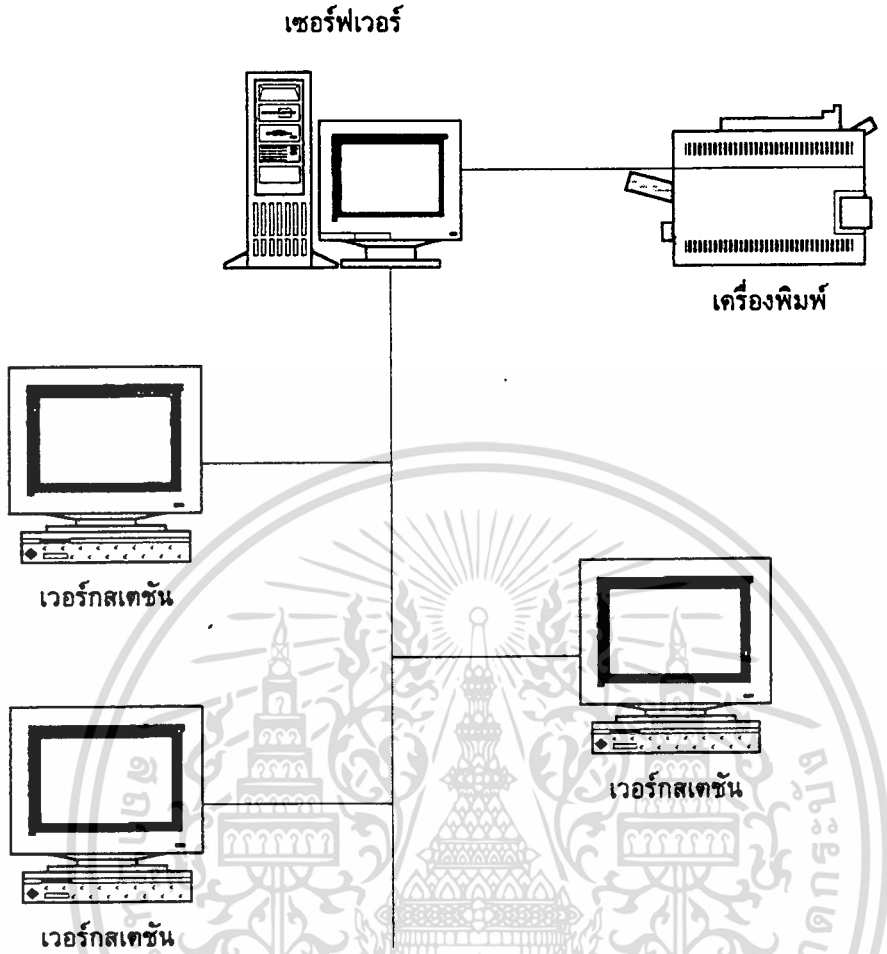
รูปที่ 3 การใช้ TCP/IP ใน ระบบไคล์เอ็นต์เซอร์ฟเวอร์

2.3.3 ลักษณะของระบบไคล์เอ็นต์เซอร์ฟเวอร์

ลักษณะการใช้งานตามรูปแบบของระบบไคล์เอ็นต์เซอร์ฟเวอร์ จะเป็นการแบ่งสรรทรัพยากรต่าง ๆให้ใช้ร่วมกัน โดยที่ผู้ใช้คิดว่าทรัพยากรนั้นอยู่บนเครื่องของตน โดยการทำงานของระบบไคล์เอ็นต์เซอร์ฟเวอร์ในระบบเครือข่ายคอมพิวเตอร์ จะสามารถแบ่งออกได้เป็นสองชนิดคือ

1. ระบบการใช้อุปกรณ์ต่าง ๆภายในระบบร่วมกัน (Share Device Processing)
2. ระบบไคล์เอ็นต์เซอร์ฟเวอร์ โปรเซส (Client/Server processing)

ระบบการใช้อุปกรณ์ต่าง ๆภายในระบบร่วมกัน ถูกพัฒนาขึ้นเพื่อตอบสนองการขยายตัวอย่างรวดเร็วของระบบเครือข่าย ซึ่งจะมีการทำงานดังรูปที่ 4



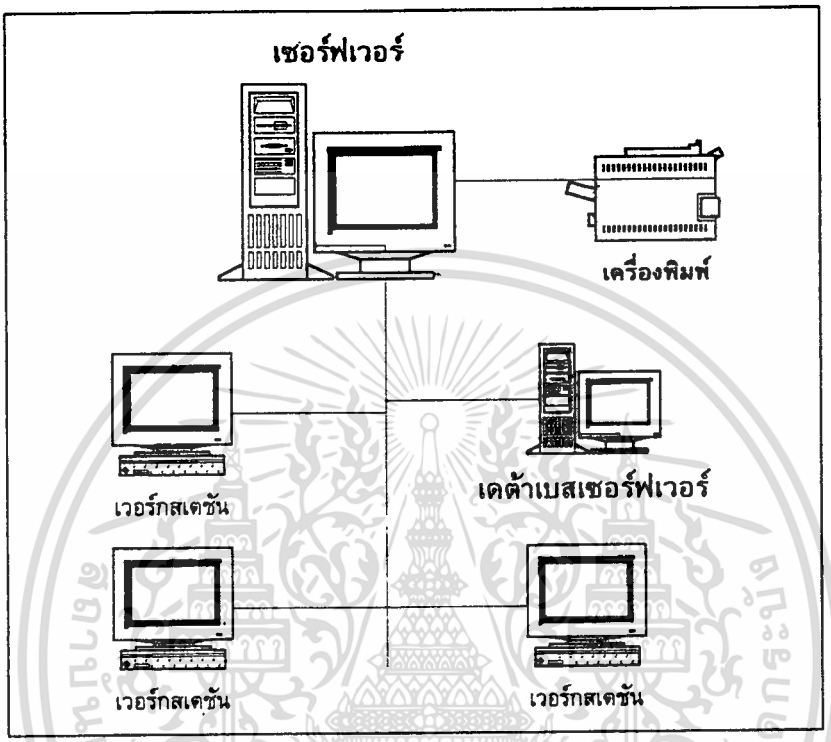
รูปที่ 4 แสดงการทำงานระบบการใช้อุปกรณ์ต่างๆภายในระบบร่วมกัน

การทำงานของระบบในลักษณะนี้ แอพพลิเคชันจะมีการทำงานที่บนโฮสต์ (Host) ซึ่งในระบบดังกล่าวจะเรียก โฮสต์ ว่า เซอร์ฟเวอร์ การทำงานส่วนใหญ่จะยอมให้ผู้ใช้มาร่วมใช้ทรัพยากร (Resource)ต่างๆที่อยู่ภายในระบบ แต่รีซอร์ต่างๆจะถูกจำกัดให้เป็นเพียงการใช้ไฟล์และเครื่องพิมพ์เท่านั้น

ข้อเสียของระบบคือ การทำงานส่วนใหญ่จะอยู่ในเครื่องเวอร์กสเตชันของผู้ใช้เท่านั้น ยกเว้นฟังชั่นทางการพิมพ์และการทำงานเกี่ยวกับไฟล์ที่อยู่บนเครื่องเซอร์ฟเวอร์ เมื่อผู้ใช้ต้องการใช้แอพพลิเคชันหรือไฟล์ ก็จะมีการร้องขอมาที่เซอร์ฟเวอร์ เซอร์ฟเวอร์จะส่งไฟล์ทั้งหมดหรือแอพพลิเคชันที่ต้องการ ผ่านระบบเครือข่ายไปให้กับเครื่องเวอร์กสเตชันของผู้ใช้จากนั้นเวอร์กสเตชันจะโหลดเอาแอพพลิเคชันเข้ามาเพื่อทำงานต่อไป เซอร์เวอร์ที่ทำหน้าที่ในการแบ่งไฟล์ต่างๆนี้เรียกว่า ไฟล์เซอร์ฟเวอร์

ฟังก์ชันการพิมพ์มีการทำงาน คือ เมื่อผู้ใช้ต้องการพิมพ์ไฟล์ข้อมูล ก็จะมีการส่งไฟล์ข้อมูลที่ต้องการไปยังเซอร์ฟเวอร์เพื่อส่งออกมาพิมพ์ เซอร์ฟเวอร์ที่ทำหน้าที่นี้จะเรียกว่า พริ้นเซอร์ฟเวอร์

ระบบไคล์เอ็นต์เซอร์ฟเวอร์ โปรเซส จะมีลักษณะคล้ายกับระบบ การใช้อุกรณ์ต่างๆ ร่วมกัน แต่แทนที่การคำนวนทางด้านแอพพลิเคชันจะอยู่ในเครื่องเวอร์กสเตชันของผู้ใช้ทั้งหมด ก็จะมีการแบ่งการคำนวนมาอยู่มาอยู่ในส่วนของเซอร์ฟเวอร์ด้วยดังรูปที่ 5



รูปที่ 5 แสดงการทำงาน ระบบไคล์เอ็นต์เซอร์ฟเวอร์ โปรเซส

จะเห็นว่าแอพพลิเคชั่นทำงานทั้งในส่วนของ เซอร์ฟเวอร์ และ ส่วนของเวอร์กสเตชันด้วย เมื่อผู้ใช้ต้องการข้อมูลบางส่วนที่อยู่ในดาต้าเบสไฟล์ขนาดใหญ่ เครื่องเวอร์กสเตชันจะส่งคำร้องขอข้อมูลนั้นไปที่ตัวเซอร์ฟเวอร์ เซอร์ฟเวอร์จะทำการคำนวนและดึงข้อมูลเหล่านั้นออกมา และจะส่งเฉพาะข้อมูลนั้นๆกลับมายังเครื่องเวอร์กสเตชันที่มีการร้องขอมา เมื่อเวอร์กสเตชันได้รับข้อมูลก็จะนำมาคำนวณต่อไป

จะเห็นได้ว่าข้อมูลที่ส่งอยู่ภายในระบบจะมีเพียงข้อมูลที่จำเป็นเท่านั้นในขณะที่ถ้าส่งในระบบการใช้งานอุปกรณ์ต่างๆร่วมกัน จะมีการส่งไฟล์ดาต้าเบสใหญ่ๆทั้งไฟล์มาที่เครื่องเวอร์ก-สเตชัน ทั้งที่ต้องการข้อมูลเพียงไม่กี่ไบต์ก็ตาม

เมื่อองค์กรใดมีการใช้งานแอพพลิเคชันแบบการใช้งานอุปกรณ์ต่างๆร่วมกัน บนระบบเครือข่ายมากเกินไปจะทำให้ระบบเครือข่ายนั้นไม่สามารถที่จะรองรับปริมาณของข้อมูลที่มากๆได้ ทำให้เวลาในการตอบสนองของระบบต่อผู้ใช้มากตามไปด้วยแต่ถ้าหน่วยงานใดหันมาใช้แอพพลิเคชันแบบไคล์เอ็นต์เซอร์ฟเวอร์มากขึ้นก็จะทำให้การทำงานของระบบงานดีขึ้นจึงทำให้ระบบการทำงานแบบ ไคล์เอ็นต์เซอร์ฟเวอร์ เป็นที่นิยมมากในปัจจุบัน

ระบบไคล์เอ็นต์เซอร์ฟเวอร์ยังสามารถแบ่งรายละเอียดได้เป็น 5 ชนิดคือ

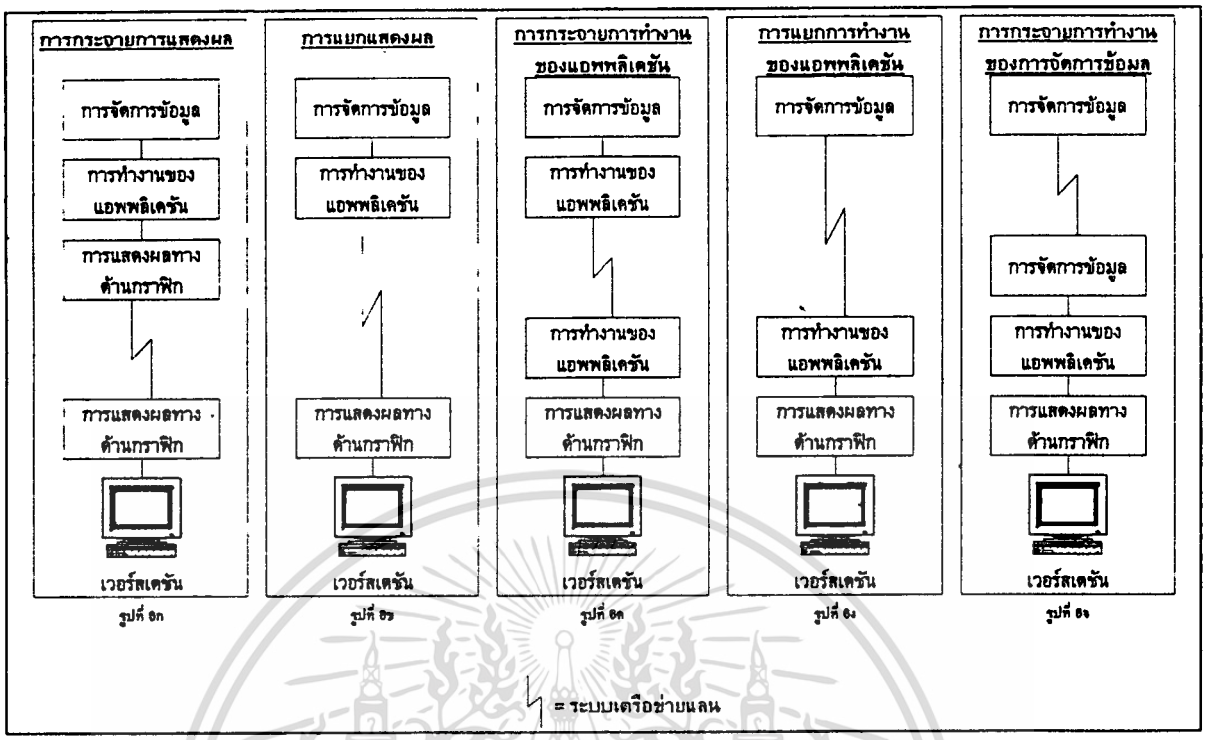
1. ชนิดการกระจายการแสดงผล (Distributed Presentation) เป็นการทำงานภายใต้ การติดต่อกับผู้ใช้ในรูปแบบของกราฟิก เช่น วินโดวส์ (Windows) เซอร์ฟเวอร์จะทำการคำนวณและส่งภาพกราฟิกที่ใช้ติดต่อกับผู้ใช้มาที่เครื่องเวอร์กสเตชันเมื่อมีการร้องขอมา ในขณะเดียวกันกับผู้ใช้ก็จะไม่สามารถบอกกว่าแตกต่างได้เลยว่าภาพที่ปรากฏอยู่นั้นเป็นภาพที่มาจากเซอร์ฟเวอร์หรือจากในตัวเครื่องเอง การทำงานของระบบไคล์เอ็นต์เซอร์ฟเวอร์ชนิดนี้จะเป็นแบบที่กระจายการทำงานน้อยที่สุดของทั้ง 5 แบบ ดังแสดงในรูปที่ 6ก

2. ชนิดแยกการแสดงผล (Remote Presentatuion) จากรูปที่ 6ข จะเห็นว่าการแสดงผลต่างๆ ต่อผู้ใช้จะถูกกำหนดจากเวอร์กสเตชันหรือไคล์เอ็นต์เท่านั้น การทำงานของแอพพลิเคชันลอจิกจะอยู่ในเซอร์ฟเวอร์ทั้งหมด รวมทั้งการจัดการข้อมูลด้วย ข้อมูลที่ส่งข้ามระบบเครือข่ายก็จะมีน้อยกว่าชนิดที่หนึ่งเพราะว่าไม่ต้องส่งภาพกราฟิกที่มีขนาดใหญ่ข้ามระบบไปให้ผู้ใช้เลย

3. ชนิดการกระจายการทำงานของแอพพลิเคชัน (Distributed Logic) ระบบชนิดนี้จะมีการกระจายการทำงานของตัวแอพพลิเคชันเองให้อยู่ในทั้งตัวเซอร์ฟเวอร์และเวอร์กสเตชันดังเช่นในรูปที่ 6ค จะเห็นได้ว่าเวอร์กสเตชันจะช่วยเซอร์ฟเวอร์ทำงานทางด้านแอพพลิเคชันเป็นส่วนหน้า (Front-End) ซึ่งมีการเชื่อมโยงการแอพพลิเคชันส่วนหลัง (Back-End) ที่อยู่ในตัวเซอร์ฟเวอร์ด้วย การคำนวณทางด้านแอพพลิเคชันก็จะถูกแบ่งออกจากเซอร์เวอร์มากขึ้นตัวอย่างเช่น ในเวอร์กสเตชันมีการคำนวนหลายชุด ก็จะมีการส่งการคำนวณบางส่วนหรือทั้งหมดไปให้เซอร์ฟเวอร์คำนวณและจะส่งผลลัพธ์กลับมายังส่วนของตัวเวอร์กสเตชัน ดังรูป

4. ชนิดแยกการทำงานของแพพลิเคชัน (Remote Date Management) การทำงานของแอพพลิเคชันทั้งหมดจะอยู่ในเวอร์กสเตชันทั้งหมด และเซอร์ฟเวอร์จะมีส่วนการจัดการข้อมูลเท่านั้น เมื่อ เวอร์กสเตชันต้องการข้อมูลส่วนใดก็สามารถร้องขอไปที่เซอร์เวอร์ได้ เซอร์เวอร์จะทำการค้นหาข้อมูลให้และส่งกลับมาที่เครื่องเวอร์กสเตชันดังแสดงในรูปที่ 6 ง

5. ชนิดการกระจายการจัดการข้อมูล (Distributed DataBase) เป็นชนิดที่นิยมใช้กันอย่างแพร่หลายในแอพพลิเคชัน ไซเบส(Sybase) หรือ เอสคิวแอว เซอร์ฟเวอร์ (SQL Server) เป็นต้น การคำนวณทางด้านแอพพลิเคชันทั้งหมดและบางส่วนของการจักการข้อมูลจะอยู่ในเครื่องเวอร์ก-สเตชัน และส่วนหลังของการจัดการข้อมูลจะอยู่ในเซอร์ฟเวอร์ดังรูปที่ 6จ วิธีนี้สามารถเพิ่มประสิทธิภาพตัวเซอร์ฟเวอร์ขึ้นมาก เพราะการคำนวณส่วนใหญ่จะอยู่ในส่วนของเวอร์กสเตชัน แต่การค้นหาข้อมูลที่มีความซับซ้อนจะอยู่ในตัวส่วนของเซอร์ฟเวอร์ แต่เวอร์กสเตชันอาจจะต้องมีความสามารถในการทำงานสูงพอสมควร การทำงานของทั้งระบบก็จะมีความรวดเร็วมากขึ้น



รูปที่ 6 ชนิดต่างๆของระบบแบบไคล์เอ็นต์เซอร์เวอร์

2.3.4 การแบ่งแยกโครงสร้างเครือข่ายตามลักษณะผู้ใช้งาน

ระบบเครือข่ายคอมพิวเตอร์ในแนวทางอุดมคติต้องการรวบรวมทรัพยากรทั้งหมดมาใช้ร่วมกัน เพื่อให้เกิดการทำงานมีประสิทธิภาพรวมสูงสุด และทำงานโดยมีฐานของเครือข่ายสนับสนุนจึงเรียกชื่อลักษณะการทำงานได้หลายอย่างเช่น

- **การคำนวนแบบเป็นกลุ่ม** (Workgroup Computing)

เป็นลักษณะการทำงานในระบบเครือข่ายที่ทุกคนรวมกลุ่มกันทำงาน และใช้ทรัพยากรร่วมกัน อาจจะทำงานงานเดียวแต่ช่วยกันทำหลายคน เช่น การพัฒนาซอฟต์แวร์โดยแบ่งแยกกันทำคนละโมเดลแล้วเอามารวมกัน มีการใช้ฮาร์ดดิสก์และพรินเตอร์ร่วมกัน สามารถแบ่งข้อมูลหรือโปรแกรมทำงานร่วมกัน ทำให้เกิดการประหยัดและเพิ่มประสิทธิภาพ

- **การคำนวนแบบกระจาย** (Distributed Computing)

เป็นการนำเอาเครือข่ายคอมพิวเตอร์มาช่วยสนับสนุนการทำงาน เช่น การกระจายฐานข้อมูลโดยแต่ละส่วนของระบบจะดูแลรักษาข้อมูลของตน เมื่อทั้งระบบสามารถรวมกันได้ก็ทำให้การดำเนินงานมองเป็นระบบเดียวกัน

- **การคำนวนแบบเครือข่าย** (Network Computing)

เป็นแนวทางการนำเครือข่ายมารวมกับระบบเป็นมุมมองของการจัดรวมศูนย์โดยใช้ฮาร์ดแวร์แยกกันแต่ระบบการทำงานเดียวกัน และจะส่งผลให้เกิดการทำงานในลักษณะรวมศูนย์เหมือนระบบเมนเฟรมคอมพิวเตอร์

- **การคำนวนแบบไคล์เอ็นต์เซอร์ฟเวอร์** (Client Server Computing)

เป็นระบบที่ใช้คอมพิวเตอร์เครื่องหนึ่งเป็นผู้ใช้ทรัพยากรของเครื่องอื่นผ่านทางเครือข่ายมีลักษณะการแบ่งทรัพยากรให้เกิดประโยชน์มากที่สุด

2.3.5 ไคล์เอ็นต์เซอร์ฟเวอร์กับการประมวลผลรายการย่อยแบบเวลาจริง (OLTP : Online Transaction Processing)

Online Transaction Processing เป็นระบบที่ทำการประมวลผลรายการย่อยแบบเวลาจริง เช่น ระบบงานของธนาคารในการฝากถอนผ่านตู้เอทีเอ็ม ระบบในลักษณะนี้มีการทำงาน คือ ส่วนของเทอร์มินัลจะส่งรายการย่อยไปประมวลผลที่หน่วยประมวลผลกลางของระบบ การวัดประสิทธิภาพของระบบจึงต้องพิจารณากันที่จำนวนรายการย่อยที่รับส่งเข้าประมวลผลได้สูงสุดต่อวินาที

สำหรับโมเดลของไคล์เอ็นต์เซอร์ฟเวอร์ที่กล่าวมาแล้วสามารถแทนสถาปัตยกรรมของการประมวลผลรายการย่อยแบบเวลาจริงได้ โดยซีพียูบนเครือข่ายสามารถส่งรายการย่อยไปยังซีพียูตัวอื่นเพื่อให้ประมวลผลได้ การทำงานนี้อาจส่งข้ามระบบปฏิบัติงานกันก็ได้

อย่างไรก็ดี ระบบเซอร์ฟเวอร์ที่ใช้สำหรับงานการประมวลผลรายการย่อยแบบเวลาจริง ก็ยังคงเป็น เอสคิวแอล ดาต้าเบส ทั้งนี้เพราะความเร็วในการตอบสนองทำได้สูง การสร้างโปรแกรมประยุกต์จะแยกออกจากตัวข้อมูลจึงทำให้การปรับปรุงเปลี่ยนแปลงทั้งทางด้านฮาร์ดแวร์และซอฟต์แวร์ทำได้ง่ายกว่า และมีราคาของระบบโดยรวมถูกกว่า

2.3.6 แนวความคิดของไคล์เอ็นต์เซอร์ฟเวอร์

- **แนวความคิดของผู้ใช้และผู้ให้บริการ**

ระบบการสื่อสารข้อมูลทำให้คอมพิวเตอร์ติดต่อสื่อสารถึงกันได้ ระบบจัดการเครือข่ายท้องถิ่นอาศัยการสื่อสารข้อมูล ในการให้บริการเก็บรักษาแฟ้ม โดยการรวบรวมเอาแฟ้มข้อมูลต่างๆ ไว้ที่เดียวกัน ทำให้ผู้ใช้หลายคนสามารถเข้าถึงแฟ้มข้อมูลที่ต้องการใช้ร่วมกันได้ โปรแกรมต่างๆ สามารถเก็บไว้ที่เดียวบนศูนย์บริการแฟ้มข้อมูล (NFS : Network File Server) ทำให้สะดวกในการใช้แหล่งทรัพยากรร่วมกัน ถ้ามองดูการบริการแฟ้มข้อมูลของระบบจัดการเครือข่ายท้องถิ่นจะพบว่าศูนย์บริการแฟ้มข้อมูลเหล่านี้ทำหน้าที่เพื่อเชื่อมต่อผู้ใช้ที่อยู่ปลายทางเข้ากับหน่วยเก็บข้อมูลขนาดใหญ่ (Mass Storage) ที่อยู่ที่ไฟล์เซอร์ฟเวอร์เท่านั้น โดยที่ผู้ที่ใช้ปลายทางมองดูเหมือนกับว่ามีแฟ้มข้อมูลเหล่านั้นอยู่บนเครื่องของผู้ใช้เอง โดยระบบการประมวลผลทุกอย่างยังคงกระทำที่เครื่องคอมพิวเตอร์ปลายทางที่ใช้อยู่ ซึ่งจะต่างจากการใช้ ดัม เทอร์มินัล (Dump Terminal) ของเครื่องเมนเฟรมหรือเครื่อง Multi User โดยทั่วๆไป

- รูปแบบของผู้ใช้และผู้ให้บริการ

ระบบจัดการฐานข้อมูลจะคอยทำหน้าที่บริการโปรแกรมที่ทำงานเรียกระบบจัดการฐานข้อมูลจากผู้ใช้หลายๆคนพร้อมกัน โดยที่ระบบจัดการฐานข้อมูลจะทำงานเป็น โปรเซสแบบเบื้องหลัง (Background Process) ทำหน้าที่คอยบริการให้กับ โปรเซสส่วนหน้า (Front End Process) ซึ่งก็คือกระบวนการทำงานโปรแกรมของผู้ใช้แต่ละคน ถ้าจะมองว่าเป็นคอนเซ็ปต์ของผู้ใช้และผู้ให้บริการ ก็คงจะเป็นผู้ใช้และผู้ให้บริการระหว่างโปรเซสภายในเครื่องเดียวกัน คือ ระหว่าง โปรเซสส่วนหลัง (Back End Process) และ โปรเซสส่วนหน้า ซึ่งมีหลายโปรเซสตามผู้ใช้ แนวความคิดของไคล์เอ็นต์เซอร์ฟเวอร์ดังกล่าวได้ถูกพัฒนามาอยู่บนเครือข่ายท้องถิ่น โดยแอพพลิคเคชั่นตัวแรกที่ถูกนำมาใช้คือ ระบบจัดการฐานข้อมูล โดยจะมีศูนย์บริการฐานข้อมูลแยกอยู่บนเครื่องคอมพิวเตอร์เครื่องหนึ่งโดยเฉพาะ ส่วนไคล์เอ็นต์โปรเซสแยกทำงานอยู่บนพีซีที่เป็นสมาชิกของระบบเครือข่ายท้องถิ่น

2.3.7 Btrieve ผู้ใช้และผู้ให้บริการตัวแรกของเน็ตแวร์

เน็ตแวร์เป็นที่รู้จักในระบบเครือข่ายในลักษณะของผู้ให้บริการแฟ้มข้อมูล ในขณะเดียวกันเน็ตแวร์เองก็ได้เสริมโปรเซสอันหนึ่งไว้ในศูนย์บริการแฟ้มที่เป็นลักษณะของผู้ใช้และผู้ให้บริการนั่นคือ บีทีฟ (Btrieve) ซึ่งเน็ตแวร์เรียกว่า เรคอร์ดแมนเนเจอร์ ลักษณะการทำงานของบีทีฟ เป็นลักษณะของผู้ใช้และผู้ให้บริการจริงๆ โดยผู้ให้บริการทำงานร่วมกับผู้บริการแฟ้มข้อมูลคอยรับคำสั่งจากผู้ใช้บริการในรูปแบบของการเข้าถึงเรคอร์ดในแฟ้มข้อมูลแล้วตอบสนองต่อคำสั่งนั้นแอพพลิเคชั่นที่ใช้บีทีฟจะทำงานได้รวดเร็วในสภาพการทำงานในลักษณะเครือข่าย อย่างไรก็ตามบีทีฟยังเป็นแค่เพียงผู้จัดการเรื่องเรคอร์ดในแฟ้มข้อมูลเท่านั้น ยังไม่ใช่ระบบจัดการฐานข้อมูลที่แท้จริง

2.3.8 ดาต้าเบสเซอร์ฟเวอร์ (ศูนย์บริการฐานข้อมูล)

ผู้ให้บริการฐานข้อมูลต้องมีระบบจัดการฐานข้อมูลที่ดี เพื่อบริการผู้ใช้บริการอย่างมีประสิทธิภาพ ฐานข้อมูลที่สามารถตอบสนองความต้องการของผู้ใช้ได้ค่อนข้างสูงคือ ระบบฐานข้อมูลเชิงสัมพันธ์ เพราะเป็นฐานข้อมูลที่สามารถสร้างความสัมพันธ์ระหว่างตาราง โดยที่ความสัมพันธ์ระหว่างตาราง คือ ความเป็นอันหนึ่งอันเดียวของข้อมูล และสามารถแสดงความสัมพันธ์ระหว่างตารางได้ง่าย และมีการย้อนกลับ (Rollback) เมื่อมีความผิดพลาดเกิดขึ้น

สิ่งที่ควบคู่มากับระบบฐานข้อมูลเชิงสัมพันธ์ ก็คือ ภาษาเอสคิวแอล ซึ่งเป็นภาษาอย่างง่ายๆ ในการเข้าถึงระบบฐานข้อมูลเชิงสัมพันธ์ ผู้ใช้บริการ (ไคล์เอ็นต์) จะใช้คำสั่งเอสคิวแอล ส่งมายังเครือข่ายไปยังผู้ให้บริการฐานข้อมูล ซึ่งที่นั่นคำสั่งจะถูกแปลออกมาเป็นความหมายในการประมวลผลเพื่อส่งข้อมูล (ที่เป็นผลลัพธ์ของคำสั่ง) พร้อมทั้งสถานภาพกลับไปยังผู้ใช้บริการผู้ให้บริการฐานข้อมูลจะส่งเฉพาะข้อมูลที่ขอมาจากผู้ใช้บริการเท่านั้น การรักษาความถูกต้องและการจัดการในเรื่องของฐานข้อมูล ดำเนินการ ณ จุดผู้ให้บริการฐานข้อมูล โดยผู้ใช้บริการไม่จำเป็นต้องดำเนินการใดๆเลย เช่น ไม่ต้องตรวจสอบว่าความสัมพันธ์ระหว่างตาราง ถูกต้องหรือไม่ และไม่ต้องคอยบอกว่าให้ใช้อินเด็กซ์ตัวใดตัวหนึ่ง

2.3.9 การกระจายฐานข้อมูล (Distributed Database)

ระบบผู้ให้และผู้ใช้บริการทางด้านฐานข้อมูลทำให้ระบบการกระจายฐานข้อมูลเป็นไปได้โดยง่าย ผู้ให้บริการ (เซอร์ฟเวอร์) อาจจะอยู่ในส่วนไหนของระบบเครือข่ายก็ได้ เมื่อต้องการที่จะเพิ่มฐานข้อมูล เพียงแค่เพิ่มผู้ให้บริการพร้อมกับตั้งชื่อฐานข้อมูลที่ให้บริการไม่ให้ซ้ำกับที่มีอยู่แล้วในเครือข่าย ก็สามารถเพิ่มการให้บริการในเครือข่ายได้ ประการสำคัญ คือ ทำให้การสื่อสารตรงกลางไม่ได้เป็นอุปสรรคในการเรียกค้นหาข้อมูลที่ต้องการ เพราะว่าในระบบเครือข่ายท้องถิ่นเมื่อต่อเชื่อมเข้าหากันระหว่างท้องถิ่นจะเป็นระบบเครือข่ายพื้นที่กว้าง การสื่อสารระหว่างท้องถิ่นเป็นสิ่งสำคัญที่จำเป็นของการทำงานทั้งหมด หากพูดกันน้อยแต่ได้งานมากตามระบบของผู้ให้และผู้รับบริการก็จะทำให้การกระจายข้อมูลไปตามส่วนต่าง ๆเป็นไปได้โดยง่าย

2.3.10 ขนาดที่เล็กลงและขนาดที่เหมาะสม (Down Sizing และ Right Sizing)

เมื่อการกระจายฐานข้อมูลได้ก็สามารถเพิ่มเติมฐานข้อมูลได้โดยง่ายผู้ใช้มองไม่เห็นความแตกต่างของการมีฐานข้อมูลที่แยกออกจากกัน เราก็สามารถที่จะปรับขนาดของฐานข้อมูลและผู้ให้บริการฐานข้อมูลให้มีขนาดเล็กลงได้ ไม่จำเป็นต้องมีเครื่องคอมพิวเตอร์ขนาดใหญ่ไว้คอยให้บริการดัม เทอร์มินัล (Dump terminal) อีกต่อไปเป็นการลดขนาดลงให้เหมาะสมกับความต้องการ (Down Sizing) และเพิ่มเติมฐานข้อมูลในภายหลังได้โดยง่าย ในปัจจุบันจะใช้เฉพาะที่เหมาะสมก่อน (Right Sizing)

2.3.11 การสื่อสารกันระหว่างผู้ให้บริการและผู้ใช้บริการ

ระบบผู้ใช้และผู้ให้บริการแบ่งการทำงานออกเป็น 3 ชั้นหลัก ๆ คือ

1. ชั้นของโปรแกรมแอพพลิเคชัน
    1.1 บนเครื่องของผู้ใช้บริการ คือ ตัวโปรแกรมที่ติดต่อกับผู้ใช้ ทั้งการดึงข้อมูลมาใช้ ปรับปรุงแก้ไขเพิ่มเติมตามคำสั่งของผู้ใช้
    1.2 บนเครื่องของผู้ให้บริการ คือโปรแกรมจัดการฐานข้อมูลทำหน้าที่ปฏิบัติตามคำสั่งจากผู้ใช้บริการในการจัดการเกี่ยวกับฐานข้อมูล
2. ชั้นของเชื่อมต่อกับระบบสื่อสาร ได้แก่ โปรแกรมที่ทำหน้าที่นำเอาคำสั่ง เอสคิวแอล จากโปรแกรมผู้ใช้บริการมาจัดเป็นกลุ่มบรรจุหีบห่อ (Package) ที่มีชื่อผู้ส่งและผู้รับพร้อมที่จะจัดส่งลงในระบบการสื่อสารที่เป็นชั้นล่างสุดขณะเดียวกันก็ทำหน้าที่รับเอาหีบห่อจากชั้น 1 การสื่อสารในส่วนล่างมาแยกออกเป็นคำสั่งหรือข้อมูลที่จะจัดส่งให้กับโปรแกรมแอพพลิเคชันในส่วนบน โปรแกรมส่วนนี้มักจะฝังตัวเองในหน่วยความจำของดอส เรียกว่า โปรแกรมเร้าเตอร์ (Router)
3. ชั้นของระบบจัดการสื่อสาร เป็นชั้นของระบบจัดการสื่อสารโดยตรง ซึ่งจะทำหน้าที่ นำหีบห่อที่ได้รับจากชั้นที่ 2 มาจัดลำดับและทำหน้าที่ส่งไปตามสายการสื่อสาร โปรแกรมส่วนนี้ได้แก่ IPX หรือ SPX ของเน็ตแวร์

2.3.12 การนำระบบผู้ใช้และผู้ให้บริการมาใช้ในวงการธุรกิจ

หากจะแยกแอพพลิเคชันที่นำเอาระบบผู้ใช้และผู้ให้บริการมาใช้อาจจะแยกได้ตามรูปแบบของการแบ่งส่วนประกอบทางตรรกและการกระจาย ชิ้นส่วนทางตรรกนั้นระหว่างผู้ใช้บริการและผู้ให้บริการชิ้นส่วนทางตรรกของแอพพลิเคชันอาจจะแบ่งได้เป็น 3 ส่วน คือ

1. ส่วนแสดงผล ติดต่อกับแป้นพิมพ์
2. ส่วนการตรวจสอบความถูกต้องของข้อมูล
3. ส่วนประยุกต์ข้อมูลมาใช้กับธุรกิจ

โดยแอพพลิเคชันส่วนมากจะมีชิ้นส่วนทางตรรกที่เป็นส่วนแสดงผลเพียงส่วนเดียว แต่อาจจะมีส่วนประยุกต์กับธุรกิจและส่วนประกอบของข้อมูลหลายส่วน การกระจายส่วนประกอบทั้ง 3 นี้ ระหว่างผู้ใช้บริการและผู้ให้บริการทำให้อาจจะแบ่งเป็นสถาปัตยกรรมผู้ใช้และผู้ให้บริการได้ 3 รูปแบบ คือ

1. รูปแบบการเข้าถึงข้อมูลระยะไกล (Remote Data Acess Model)

ส่วนแสดงผลและส่วนประยุกต์ข้อมูลกระทำบนฐานฮาร์ดแวร์ของผู้ใช้บริการ ส่วนการเข้าถึงข้อมูลนั้นผู้ใช้บริการจะใช้คำสั่งเอสคิวแอล ให้ผู้ให้บริการดำเนินการให้ ข้อมูลยังคงมีการย้ายโอนไปมาระหว่างผู้ใช้และผู้ให้บริการอยู่เป็นจำนวนไม่น้อย

2. รูปแบบการบริการฐานข้อมูล (Database Server Model)

ส่วนที่เป็นผู้ใช้บริการทำหน้าที่แสดงผลและแป้นพิมพ์หรืออุปกรณ์อื่นๆ (เกือบจะทำหน้าที่คล้ายกับดัมเทอร์มินัลในระบบหลายผู้ใช้ของเครื่องเมนเฟรม) ส่วนที่เหลือผู้ให้บริการดำเนินการให้หมดในลักษณะของการเก็บคำสั่ง หรือกระบวนการไว้บริการผู้ใช้บริการ

3. รูปแบบบริการแอพพลิเคชัน (Application Server)

ในรูปแบบนี้ผู้ใช้บริการจะจัดการเกี่ยวกับการแสดงผลและแป้นพิมพ์อย่างเดย แต่ตรรกในทางการประยุกต์ถูกกำหนดไว้ที่ผู้ให้บริการ และพร้อมที่จะให้โปรแกรมส่วนของผู้ใช้บริการเรียกใช้ได้ในลักษณะโปรแกรมสู่โปรแกรม

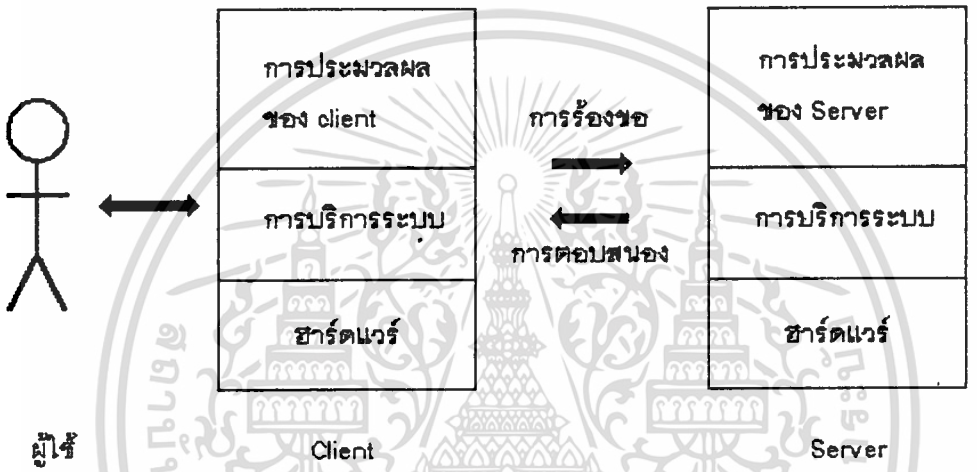
2.3.13 โปรแกรมประยุกต์ฟรอนต์เอ็นของระบบไคล์เอ็นต์เซอร์ฟเวอร์

ระบบไคล์เอ็นต์เซิร์ฟเวอร์ ประกอบด้วยหน่วยที่เรียกว่าไคล์เอ็นต์ (ผู้รับบริการ) จะทำหน้าที่เป็นฟรอนต์เอ็น (Front-end)เพื่อติดต่อรับคำสั่งจากผู้ใช้และร้องขอรับบริการจากหน่วยที่เรียกเซิร์ฟเวอร์ (ผู้ให้บริการ) ที่เป็นแบ็คเอ็น (Back-end) ในการให้บริการต่างๆ ทั้งหน่วยไคล์เอ็นต์และหน่วยเซิร์ฟเวอร์ต่างเป็นเครื่องคอมพิวเตอร์ที่แยกจากกัน แต่เชื่อมโยงติดต่อถึงกันด้วยเครือข่ายคอมพิวเตอร์ที่อาจเป็นระบบแลน (LAN) หรือแวน (WAN) ก็ได้

ระบบไคล์เอ็นต์เซิร์ฟเวอร์นี้ได้มีการนำไปประยุกต์ในงานเรียกค้นข้อมูลจากฐานข้อมูลในเครือข่ายมากที่สุดโดยหน่วยไคล์เอ็นต์จะรับคำสั่งเรียกค้นจากผู้ใช้แล้วสร้างเป็นรูปแบบคำสั่งภาษาเอสคิวแอล แล้วส่งไปยังหน่วยเซิร์ฟเวอร์เพื่อขอรับบริการข้อมูล มักจะหมายความถึง ดาต้าเบสเซิร์ฟเวอร์ หน่วยเซิร์ฟเวอร์จะทำงานตามคำร้องขอและให้ผลลัพธ์ข้อมูลกลับคืนไปยังฟรอนต์เอ็นของหน่วยไคล์เอ็นต์เพื่อการจัดทำรูปแบบนำเสนอต่อผู้ใช้อีกครั้งหนึ่ง

ในหน่วยไคล์เอ็นต์และเซิร์ฟเวอร์อาจแบ่งเป็นชั้น ๆ ได้สามระดับ ดังนี้

1. ส่วนของฮาร์ดแวร์ (Hardware)
2. ส่วนการบริการระบบ (System Services) ประกอบด้วย โปรแกรมปฏิบัติการ โปรแกรมเครือข่าย และโปรแกรมวินโดวส์ โดยจะรวมถึงซอฟต์แวร์ต่าง ๆ
3. ส่วนของโปรแกรมประยุกต์ แยกเป็นส่วนของไคล์เอ็นต์ เรียกว่า ฟรอนต์เอ็นแอพพลิเคชั่น (Front-end Applications) และส่วนของเซิร์ฟเวอร์ เรียกว่า แบ็คเอ็น ดาต้าเบส เอ็นจิน (Back-end Database Engine)



รูปที่ 7 ระบบไคล์เอ็นเซอร์ฟเวอร์

ฟรอนต์เอ็น สามารถแบ่งได้เป็น 4 ประเภทตามลักษณะการใช้งานหลักของโปรแกรมนั้น ๆ ได้แก่

1. โปรแกรมผนวกรวมในโปรแกรมประยุกต์ที่มีอยู่เดิม (Add-on existing product) เช่น ดีเบส(dBase) หรือโลตัส123 (Lotus123) เพื่อให้สามารถเรียกค้นข้อมูลจากดาต้าเบสเซิร์ฟเวอร์ได้โดยตรง
2. ซอฟต์แวร์ช่วยพัฒนาโปรแกรมประยุกต์ (Application Development Products) สำหรับนักพัฒนาหรือนักเขียนโปรแกรมใช้สร้างโปรแกรมประยุกต์ฟรอนต์เอ็นอื่น ๆ ได้ง่ายและสะดวกรวดเร็วยิ่งขึ้น
3. ซอฟต์แวร์ช่วยเรียกค้นและสร้างรายงาน (Query Tool and Report Writer) สำหรับผู้ใช้ปลายทางหรือผู้ที่ไม่ชอบเขียนโปรแกรมได้ใช้ในการเรียกค้นข้อมูลและจัดทำรายงานจากข้อมูลที่รับมาจากแบ็คเอ็นของดาต้าเบสเซิร์ฟเวอร์
4. ซอฟต์แวร์ช่วยรวมและวิเคราะห์ข้อมูล (Data Integration and Analysis Tools) สำหรับผู้ใช้ปลายทางที่อาจเป็นผู้จัดการหรือผู้บริหารระดับสูง ใช้ตรวจดูข้อมูลที่เรียกมาจากแหล่งต่าง ๆ แล้วทำการวิเคราะห์และใช้ช่วยในการตัดสินใจ

2.3.14 การทำงานของโปรแกรมฟรอนต์เอ็น

โปรแกรมประยุกต์ฟรอนต์เอ็นอาจเป็นโปรแกรมสำเร็จรูปที่มีผลิตมาขายทั่วไปหรืออาจเป็นโปรแกรมที่เขียนขึ้นมาเฉพาะเพื่อใช้สำหรับงานภายในองค์กรหนึ่ง ๆ เท่านั้นซึ่งทั้งสองลักษณะนี้จะมีการใช้งานเหมือนกันโดยมีการเรียกติดต่อกับฐานข้อมูลที่อยู่ห่างไกลออกไปผ่านทางซอฟต์แวร์ โดยการเรียกคันที่กล่าวถึงนี้จะหมายถึงการเรียกติดต่อกับฐานข้อมูล การแทรกเพิ่มข้อมูลและการลบข้อมูล

2.3.14.1 โปรแกรมประยุกต์ฟรอนต์เอ็นเป็นลักษณะที่ผนวกรวมในโปรแกรมประยุกต์เดิม

การเรียกคันข้อมูลนิยมให้ผู้ใช้ใช้ภาษาตามโปรแกรมเดิม กรณีนี้ส่วนของไคล์เอ็นต์จะทำงานมากขึ้นเพราะจะต้องแปลภาษาเรียกคันของโปรแกรมเดิมให้เป็นภาษาเรียกคันแบบ เอสคิวแอล ก่อนแล้วส่งไปยังดาต้าเบสเซิร์ฟเวอร์ ขณะเดียวกันผลลัพธ์ที่ได้รับจากดาต้าเบสเซิร์ฟเวอร์ก็จะต้อง ทำการแปลงรูปให้อยู่ในรูปแบบของโปรแกรมเดิมเพื่อนำเสนอต่อผู้ใช้ ส่วนเพิ่มเติมนี้อาจเป็นผลเสียต่อระบบ ทำให้หน่วยไคล์เอ็นต์ต้องเปลืองพื้นที่หน่วยความจำและเสียเวลาของซีพียูมากขึ้น

2.3.14.2 ซอฟต์แวร์ช่วยพัฒนาโปรแกรมประยุกต์

การสร้างโปรแกรมประยุกต์ในอดีตที่ผ่านมา จะเป็นการเขียนโปรแกรมด้วยภาษาระดับสูง เช่น ภาษาซี ภาษาโคบอล หรือภาษาปาสคาล ซึ่งการพัฒนาโปรแกรมในแนวทางนี้ค่อนข้างยากและใช้เวลาในการเรียกคันข้อมูลจากฐานข้อมูล ซอฟต์แวร์ช่วยพัฒนาโปรแกรมประยุกต์นี้จะมีโปรแกรมหรือคำสั่งเพื่อให้ผนวกรวมภายในโปรแกรมประยุกต์ เพื่อไปเรียกคันข้อมูลหรือสร้างฟรอนต์เอ็นแอพพลิเคชั่น โปรแกรมช่วยพัฒนาต่างๆ เหล่านี้จะสนับสนุนภาษาเอสคิวแอล และเรียกติดต่อไคล์เอ็นต์เซิร์ฟเวอร์ ลักษณะเด่นของโปรแกรมช่วยพัฒนาจะเน้นในด้านการใช้งานง่าย ส่วนใหญ่นิยมทำงานอยู่ภายใต้วินโดวส์มีภาษาและฟังก์ชันใช้งานเฉพาะของตนเอง ช่วยในการพัฒนาโปรแกรมประยุกต์แบบลักษณะของ event-message driven หรือเป็นภาษายุคที่สี่ บางโปรแกรมจะสนับสนุนให้ใช้งานเชิงวัตถุวิสัย (Object-oriented programming) เน้นการนำมาใช้ใหม่เป็นคลาสต่าง ๆ มีลักษณะบักกิ้ง (Debugging) เน้นในส่วนที่ให้ผู้พัฒนาตรวจสอบแก้ไขและปรับปรุงโปรแกรมประยุกต์ นอกจากนี้ผู้พัฒนายังสามารถใช้งานเรียกติดต่อข้อมูลโดยไม่ต้องรู้คำสั่ง(Command)หรือไวยกรณ์(Syntex)ของ ภาษาเอสคิวแอล และการเรียกข้อมูลนี้สามารถเรียกนำมาจากหลายฐานข้อมูลหลายแหล่ง

เนื่องจากโปรแกรมช่วยพัฒนาส่วนใหญ่ใช้งานภายใต้วินโดวส์ ดังนั้นจึงสามารถใช้ลักษณะ Dynamic Data Exchange (DDE) ซึ่งเป็นโปรโตคอลของการสื่อสารในวินโดวส์ให้มีการโอนย้ายแลกเปลี่ยนข้อมูลระหว่างโปรแกรมประยุกต์อื่นๆ อย่างอัตโนมัติได้เหมือนโปรแกรมประยุกต์บนวินโดวส์อื่น ๆ ทั่วไป การทำรายงานของโปรแกรมประยุกต์ช่วยพัฒนาจึงทำได้สวยงามและนำผลออกแสดงที่เอาต์พุตของเครื่องพิมพ์อย่างสมบูรณ์

2.3.14.3 ซอฟต์แวร์ช่วยเรียกค้นและสร้างรายงาน

ซอฟต์แวร์ประเภทนี้ถือเป็นซอฟต์แวร์ช่วยของผู้ใช้ปลายทางที่เน้นให้ใช้งานได้ง่าย สำหรับผู้ใช้ทั่วไปที่ไม่จำเป็นต้องเป็นนักโปรแกรมในการเรียกติดต่อข้อมูล โปรแกรมประยุกต์ฟรอนต์เอ็นประเภทนี้ช่วยทำงานเรียกค้นข้อมูลจากดาต้าเบสเซิร์ฟเวอร์มาแสดงบนจอ หลังจากทำการออกแบบให้ได้รายงานแล้วนำผลส่งออกที่เครื่องพิมพ์ ลักษณะชิ้นงานต่าง ๆ อาจไม่เป็นภาษาการโปรแกรม แต่อาจเป็นสคริปต์หรือโปรแกรมมาโครที่เก็บรายการการใช้ปุ่มกด ให้สามารถนำกลับมาใช้งาน โดยปกติแล้วจะมีการเก็บการเรียกค้นไว้ในไฟล์ให้ใช้ใหม่หรือดัดแปลงใหม่ได้ภายหลัง

ซอฟต์แวร์กลุ่มนี้บางโปรแกรมอาจใช้งานได้ดีทางด้านเรียกค้นข้อมูลแต่ทำรายงานไม่ดี ในทางกลับกันบางโอกาสอาจมีการเรียกค้นข้อมูลไม่ดีแต่ใช้ทำรายงานได้ดี การพิจารณาเลือกซอฟต์แวร์กลุ่มนี้มีข้อควรพิจารณาหลายประเด็น เช่น

- สนับสนุนดาต้าเบสเซิร์ฟเวอร์ตระกูลใดบ้าง ตรงกับที่ใช้งานหรือไม่
- สนับสนุนภาษาเอสคิวแอล และ ดีบีทู (DB2) หรือไม่
- ใช้ระบบติดต่อผู้ใช้เป็นมิตรมากน้อยเพียงไร
- ใช้ทำงานอย่างอัตโนมัติได้หรือไม่
- มีระบบให้ความช่วยเหลือตอบสนองตรงจุดหรือไม่
- สามารถจัดแต่งสี กราฟิก ภาพ ไอคอน รูปกราฟ และภาพ ได้ดีเพียงไร
- มีระบบให้ความปลอดภัยในการเข้าติดต่อข้อมูลได้จริงจังเพียงไร
- สามารถรวมเข้ากับซอฟต์แวร์ประยุกต์อื่น ๆ หรือไม่
- สนับสนุนการแลกเปลี่ยนข้อมูลกับโปรแกรมวินโดวส์อื่นหรือไม่เพียงใด
- มีวิธีการทำรายงานอย่างไร

2.3.14.4 ซอฟต์แวร์ช่วยรวมและวิเคราะห์ข้อมูล

ซอฟต์แวร์กลุ่มนี้จัดเป็นซอฟต์แวร์สำหรับผู้ใช้ปลายทางอีกเหมือนกัน โดยที่ผู้ใช้ไม่จำเป็นต้องเรียนรู้ภาษามาช่วยในการสร้างหรือพัฒนา ลักษณะของซอฟต์แวร์นี้จะรวบรวมข้อมูลจากหลายๆ แหล่งเพื่อมาทำการวิเคราะห์ซึ่งลักษณะนี้บางครั้งเราจะเรียกว่าซอฟต์แวร์สารสนเทศเพื่อการบริหาร (EIS:Executive Information System) เหมาะในการรวบรวมข้อมูลในองค์กร แล้วทำการรวบรวมข้อมูลในองค์กร แล้วทำการวิเคราะห์ทำเป็นรายงานและรูปกราฟนำเสนอผู้บริหารเพื่อใช้ประกอบการตัดสินใจ ทำให้เหมาะกับงานทางด้านธุรกิจมาก

ซอฟต์แวร์นี้ส่วนใหญ่ทำงานภายใต้วินโดวส์ จึงมีระบบติดต่อกับผู้ใช้ดีเชื่อมโยงข้อมูลติดต่อกับโปรแกรมประยุกต์ได้ง่าย ซอฟต์แวร์กลุ่มนี้เสมือนรวมโปรแกรมประเภทที่สามของการเรียกค้นและทำรายงานให้รวมเข้ากับโปรแกรมด้านวิเคราะห์หรือสเปรดชีตและนำเสนอด้วยกราฟิกมารวมกันในตัวเดียว ทำให้เป็นซอฟต์แวร์ที่น่าสนใจเหมาะสำหรับงานด้าน MIS มาก

2.4 ฐานข้อมูลแบบกระจาย

การใช้งานฐานข้อมูลแบบกระจายจะขึ้นอยู่กับการสื่อสารบนเครือข่ายซึ่งในปัจจุบันมีโปรโตคอล (protocol) ที่ใช้หลักๆ อยู่ 2 ตัว ได้แก่ IPX และ TCP/IP เมื่อเทคโนโลยีทางด้านเครือข่ายและระบบสื่อสารถูกพัฒนาให้เชื่อมโยงระบบต่าง ๆ บนเครือข่ายได้ดีแล้ว ระบบจัดการฐานข้อมูลก็มีการพัฒนาจนมีระบบมาตรฐานการเรียกข้อมูลระหว่างกัน เช่น ระบบ เอสคิวแอล หรือระบบคำสั่งการเรียกใช้ฐานข้อมูล ได้รับการพัฒนาให้ก้าวหน้าและใช้งานได้ดีขึ้น ระบบคำสั่งเอสคิวแอล มีลักษณะที่สำคัญ เพื่อลดปริมาณข้อมูลในเครือข่ายเพื่อประโยชน์ในงานด้านออนไลน์ งานประมวลผลข้อมูลรายการย่อย (Transaction Processing) ซึ่งระบบคำสั่ง เอสคิวแอล

ในระบบอินเตอร์เน็ตเป็นตัวอย่างของการจัดการฐานข้อมูลแบบกระจายที่เห็นได้ชัดเจนที่สุด เพราะในปัจจุบันเครือข่ายอินเตอร์เน็ตมีขนาดใหญ่ที่สุดในโลก มีเครือข่ายย่อย ๆ หลายแสนเครือข่ายเชื่อมโยงต่ออยู่ ระบบบริการข้อมูลบนเครือข่ายอินเตอร์เน็ตมีหลายอย่างที่เป็นลักษณะการสร้างฐานข้อมูลแบบกระจาย เช่น Gopher, Archie, WAIS และ WWW สิ่งเหล่านี้มีการพัฒนาให้ก้าวหน้าเป็นลำดับ เช่น การค้นหาข้อมูลด้วย Archie เป็นระบบที่จะเข้าถึงข้อมูลที่อยู่ที่ต่าง ๆ ทั่วโลก

หลัก 10 ประการในการออกแบบระบบฐานข้อมูลกระจายที่ดี

1. ฐานข้อมูลแต่ละระบบมีความอิสระ

ระบบการจัดการฐานข้อมูล แต่ละระบบจะต้องมีความอิสระ ระบบเหล่านี้อาจจะเป็นระบบที่ต่างเผ่าพันธุ์กันหรือต่างยี่ห้อกันก็ได้ ความเป็นอิสระนี้จะต้องให้แต่ละฐานข้อมูลมีระบบจัดการของตนเองไม่เกี่ยวข้องกัน โดยระบบการจัดการฐานข้อมูลแต่ละสถานที่จะต้องดูแลข้อมูล จัดระบบรักษาความปลอดภัยของข้อมูล จัดการเรื่องระบบล็อกกิ้ง การดูแลอัพเดต ปรับปรุงข้อมูล ดูแลความสัมพันธ์ สร้างรายงาน การกู้ข้อมูล การดำเนินในส่วนข้อมูลหลักที่เป็นระบบ
โลคอลจะใช้ระบบการจัดการฐานข้อมูล ตัวโลคอลเป็นหลักแต่สามารถเชื่อมโยงกับฐานข้อมูลโดยมีโปรโตคอลที่เชื่อมโยงกันเพื่อประมวลผลรายงานย่อยร่วมกันได้

ความเป็นอิสระของแต่ละฐานข้อมูลทำให้แต่ละหน่วยงานดูแลและจัดการข้อมูลของตนเอง ทำให้การใช้งานและการเกี่ยวข้องกับข้อมูลไม่มีปัญหาเพราะระบบข้อมูลมีความสัมพันธ์กับเจ้าหน้าที่ปฏิบัติงานอยู่แล้ว

2. ในการดำเนินงานปกติใช้ฐานข้อมูลย่อย

ระบบเครือข่ายคอมพิวเตอร์มีความสัมพันธ์ต่อการสร้างระบบ ดังนั้นระบบฐานข้อมูลแบบกระจายต้องมีโครงสร้างที่ลดปัญหาช่องสื่อสาร การดำเนินงานในภาวะปกติต้องใช้กับฐานข้อมูลย่อยนั้น ๆ ได้โดยไม่ต้องเกี่ยวข้องกับฐานข้อมูลกลาง ฐานข้อมูลย่อยแต่ละแห่งจึงต้องประมวลผลและสนับสนุนการดำเนินงานเฉพาะแต่ละฐานได้ แต่หากจะต้องมีการเชื่อมโยงข้อมูลระบบจะต้องมีกลไกในการอัพเดตข้อมูลร่วมกันเพื่อให้ฐานข้อมูลได้รับการปรับปรุงอย่างถูกต้อง

นอกจากนี้ระบบฐานข้อมูลกระจายยังสามารถสร้างระบบสำเนาข้อมูลร่วมกันเพื่อสร้างความเชื่อถือของระบบให้สูงขึ้น มีระบบ online recovery เมื่อฐานข้อมูลตัวใดตัวหนึ่งมีปัญหา ข้อมูลที่สำรองหรือสำเนาที่อยู่ที่ฐานข้อมูลอื่นจะนำมาใช้ในการกู้ฐานข้อมูลได้

3. ผู้ใช้งานมองไม่เห็นตำแหน่งฐานข้อมูล

ความหมายในลักษณะนี้คือ location transperency กล่าวคือผู้ใช้จะมองฐานข้อมูลกระจายที่เชื่อมโยงกันทั้งหมดเป็นระบบเดียวกัน การดำเนินการต่าง ๆ จะเป็นไปอย่างอัตโนมัติ ตำแหน่งทางฟิสิคัลของฐานข้อมูลกระจายเป็นตำแหน่งที่ไม่ปรากฏให้ผู้ใช้หรือโปรแกรมเห็น ผู้ใช้หรือโปรแกรมจะเรียกหรืออ้างถึงข้อมูลได้โดยตรง การจัดการข้อมูลทั้งระบบจะกระทำได้ทั้งการปรับปรุง การเรียกค้นหรือการเคลื่อนย้ายข้อมูลระหว่างกัน ฐานข้อมูลแบบกระจายในลักษณะนี้จึงเป็นลักษณะแบบอุดมคติที่จะสร้างระบบให้เหมือนการใช้เมนเฟรมคอมพิวเตอร์หรือฐานข้อมูลแบบรวมศูนย์

4. การเชื่อมโยงตารางข้อมลระหว่างฐานข้อมูลได้

เพื่อเข้าใจง่ายขึ้นลงพิจารณาระบบฐานข้อมูลแบบรีเลชั่นแนลที่ประกอบด้วยข้อมูลที่เป็นตารางหลาย ๆ ตาราง แต่ละตารางมีการเชื่อมโยงถึงกัน การเกี่ยวข้องของข้อมูลเหล่านี้สร้างขึ้นโดยมีการกำหนดเป็นพจนานุกรมข้อมูลเพื่อกำหนดการเกี่ยวพันต่าง ๆ

สำหรับฐานข้อมูลกระจายก็เช่นกัน มีการสร้างตารางข้อมูลกระจายอยู่ตามที่ต่าง ๆ เช่น ข้อมูลบุคลากร ข้อมูลการฝึกอบรม ข้อมูลเรื่องเงินเดือน ตารางข้อมูลในฐานข้อมูลเหล่านี้จำเป็นต้องมีการเชื่อมโยงถึงกัน เพื่อสร้างระบบการใช้งานร่วมกันได้

5. การประมวลการเรียกค้นแบบกระจาย

การเรียกค้นข้อมูลจะต้องได้รับการประมวลผลเพื่อเข้าหาฐานข้อมูลที่ถูกต้องได้ ระบบฐานข้อมูลแบบกระจายต้องสนับสนุนการเรียกค้นข้อมูลแบบกระจาย โดยมีกรรมวิธีเพื่อให้การเรียกเข้าสู่เครือข่ายไปยังฐานข้อมูลที่ต้องการอย่างรวดเร็ว สามารถหาเส้นทางและลดปริมาณการวิ่งบนเครือข่ายลง การประมวลผลบนเครือข่ายจะทำให้ประสิทธิภาพของระบบโดยรวมสูงขึ้น

6. สามารถประมวลผลรายการย่อยแบบกระจาย

ในการทำงานฐานข้อมูลแบบออนไลน์บนเครือข่ายแบบกระจายมีความจำเป็นที่จะต้องใช้วิธีการประมวลผลรายการย่อยเพื่อให้ตอบสนองการทำงานของระบบโดยรวมได้ ระบบที่ทำงานออนไลน์จึงเกี่ยวข้องกับการประมวลผลรายการย่อยที่เรียกว่า การประมวลผลรายการย่อยแบบเวลาจริง การประมวลผลในลักษณะนี้ใช้เวลาจริงเพื่อให้เป็นระบบที่ทดแทนระบบขนาดใหญ่ได้

7. ไม่ขึ้นกับฮาร์ดแวร์

ระบบฐานข้อมูลกระจายในระบบที่เชื่อมโยงกันมักเป็นระบบแบบมัลติเวนเดอร์ กล่าวคือใช้ฮาร์ดแวร์จากหลากหลายยี่ห้อ ต่อเชื่อมผ่านเครือข่ายคอมพิวเตอร์ มีระบบสื่อสารข้อมูลเชื่อมโยงกัน

เช่น มี TCP/IP ระบบเหล่านี้ไม่ขึ้นกับฮาร์ดแวร์ ซอฟต์แวร์จัดการฐานข้อมูลจะวางตัวอยู่กับ TCP/IP ทำให้ไม่ขึ้นกับฮาร์ดแวร์ที่ใช้

8. ไม่ขึ้นกับโปรแกรมจัดระบบงาน

การใช้งานในปัจจุบันบนเครือข่ายขององค์กรมีการใช้เครื่องหลากหลายยี่ห้อ ระบบจัดการฐานข้อมูล จึงเป็นระบบที่ไม่ควรให้ขึ้นกับโปรแกรมจัดระบบงาน เครื่องที่ใช้อาจมีการสร้างฐานข้อมูลบนระบบที่มีโปรแกรมโอเอสต่าง ๆ กันได้

9. ไม่ขึ้นกับเครือข่าย

เครือข่ายที่ใช้มีหลากหลายรูปแบบ เช่น เป็นอีเธอร์เน็ต โทเก้นริง มีหลายโปรโตคอล ตลอดจนมีการเชื่อมโยงระหว่างเครือข่ายต่าง ๆ เข้าด้วยกัน ระบบจัดการฐานข้อมูล จึงไม่ควรขึ้นกับรูปแบบของเครือข่ายที่ใช้

10. ทำงานร่วมกันในลักษณะ interpretability

เนื่องจากข้อกำหนดที่กำหนดให้ระบบจัดการฐานข้อมูล แต่ละตัวมีความเป็นอิสระ ระบบจัดการฐานข้อมูล แต่ละระบบต้องทำงานร่วมกันในลักษณะ interpretability ได้ ระบบนี้มีการทำงานเป็นเนื้อเดียวร่วมกันเพื่อให้เสมือนเป็นระบบเดียวกัน

2.5 หลักการของโอดีบีซี (ODBC : Open Database Connectivity)

2.5.1 ความหมายของ โอดีบีซี

โอดีบีซี คือวิธีการติดต่อและเข้าถึงจากแอพพลิเคชั่นสู่ระบบจัดการฐานข้อมูล โดยใช้ภาษาเอสคิวแอล เป็นมาตราฐานการเข้าถึงข้อมูล ความสามารถในการเชื่อมต่อแบบนี้ทำให้แอพพลิเคชั่นสามารถเข้าถึงฐานข้อมูลได้หลายรูปแบบ ซึ่งทำให้ผู้พัฒนาโปรแกรมสามารถพัฒนาโปรแกรมไปได้โดยไม่ต้องทำการระบุชนิดของระบบจัดการฐานข้อมูล

แต่เดิมนั้นการพัฒนาโปรแกรมประยุกต์ที่ใช้งานเกี่ยวกับฐานข้อมูล การเข้าใช้ฐานข้อมูลโปรแกรมเหล่านี้จะทำผ่านการเรียกใช้เอ็มเบทเด็ด เอสคิวแอล (Embedded SQL) ซึ่งในขณะนั้นวิธีการแบบนี้ก็ดูจะไปได้ทีเดียว เพราะว่าตัวโปรแกรมสามารถทำการเปลี่ยนรูปแบบของระบบไม่ว่าจะเป็นทางด้านฮาร์ดแวร์ หรือ ซอฟท์แวร์ได้หลายรูปแบบ รวมทั้งระบบปฏิบัติการด้วย (โดยการคอมไพล์ใหม่ทุกครั้งที่มีการย้ายระบบ)

อย่างไรก็ตามในการพัฒนาโปรแกรมในระบบที่มีความแตกต่างกัน เช่น การเรียกใช้ข้อมูลของออราเคิล(Oracle)จาก ไมโครซอฟท์ เอ็กซ์เซล(Microsoft Excel) วิธีการเข้าถึงข้อมูลแบบเดิมนั้นจะต้องทำการ พรีคอมไพล์โค้ดของเอ็กซ์เซล และออราเคิลโดยใช้ ไอบีเอ็ม พรีคอมไพเลอร์ (IBM Precompiler) และ ออราเคิล พรีคอมไพเลอร์ (Oracle Precompiler) ตามลำดับ ซึ่งจะเห็นว่าเป็นการยุ่งยากมากทีเดียว

วิธีการต่อเชื่อมแบบโอดีบีซี จะให้ความสะดวกในการติดต่อข้อมูลมากกว่าวิธีการดั้งเดิมโดยการกำหนดมาตรฐานการต่อเชื่อมของข้อมูล และ แนวทางนี้ได้ทำให้เกิดความคิดที่จะสร้างไดร์เวอร์การติดต่อกับของงานข้อมูลขึ้นมา

2.5.2 ข้อดีของการติดต่อโดยใช้โอดีบีซี

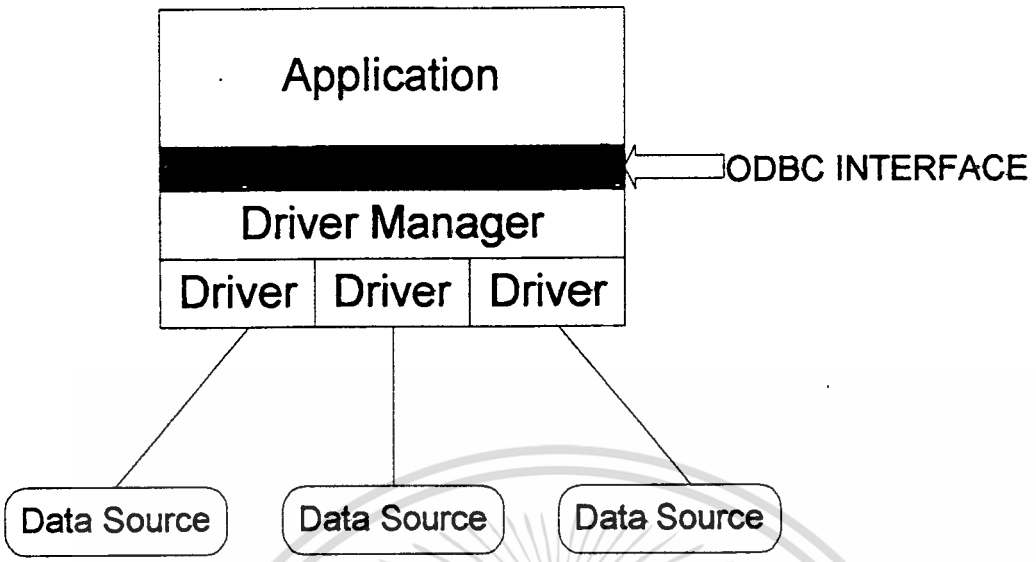
- ฟังก์ชั่นของโอดีบีซี อนุญาติให้ แอพพลิเคชั่นติดต่อกับดีบีเอ็มเอส ได้โดยสะดวก (การทำคำสั่งเอสคิวแอลและการรับผลลัพท์)
- ใช้ภาษาเอสคิวแอลตามมาตราฐาน SQL CAE, X/Open และ SQL Access Grop (SAG)
- มีการกำหนด การส่งกลับรหัสความผิดพลาด (Error Code) เป็นมาตรฐานเดียวกัน
- เป็นวิธีการมาตรฐานในการติดต่อกับดีบีเอ็มเอส
- มีการกำหนดชนิดของข้อมูลเป็นมาตรฐาน
- ชุดคำสั่งเอสคิวแอล สามารถกำหนดได้แม้ในขณะรัน
- สามารถเขียนโปรแกรมชุดเดียวแต่สามารถเข้าใช้ดีบีเอ็มเอสได้หลายตัว
- ตัวโปรแกรมไม่ต้องรับผิดชอบในการดูแลการติดต่อข้อมูลกับดีบีเอ็มเอส
- ค่าข้อมูลสามารถถูกส่งหรือรับได้ในรูปแบบที่สะดวกขึ้น

2.5.3 องค์ประกอบของโอดีบีซี

สถาปัตยกรรมของโอดีบีซีประกอบด้วย 4ส่วนสำคัญ

1. แอพพลิเคชั่น ทำหน้าที่ประมวลผลและเรียกใช้ฟังก์ชั่นของโอดีบีซี ตามคำสั่งภาษาเอสคิวแอลพร้อมทั้งทำการรับผลลัพธ์ด้วย
2. ตัวจัดการไดร์เวอร์(Driver Manager) ทำหน้าที่ จัดการไดร์เวอร์ให้เชื่อมต่อกับแหล่งข้อมูล
3. ไดร์เวอร์(Driver)ทำหน้าที่ประมวลผลการเรียกใช้ฟังก์ชั่นของโอดีบีซี ส่งคำสั่งเอสคิวแอลไปสู่แหล่งข้อมูลที่ถูกต้องและทำการส่งผลลัพธ์กลับให้แอพพลิเคชั่นและในบางครั้ง ไดร์เวอร์ จะทำหน้าที่แปลงคำสั่งที่ส่งมาให้อยู่ในรูปที่สนับสนุนโดย DBMS แต่ละชนิดอีกด้วย

4.แหล่งข้อมูล(Data Source)เป็นแหล่งข้อมูลที่ผู้ใช้ต้องการเข้าถึง



รูปที่ 8 องค์ประกอบของโอบีดีซี

ตัวโปรแกรมจะเรียกใช้การเชื่อมโอบีดีซี ในการทำงานต่อไปนี้

1. ร้องขอการต่อเชื่อมกับแหล่งข้อมูล
2. ส่งคำสั่งเอสคิวแอลสู่แหล่งข้อมูล
3. กำหนดพื้นที่การจัดเก็บและรูปแบบของข้อมูล ที่เป็นผลลัพธ์จากเอสคิวแอล รีเควสซ์ (SQL request)
4. ร้องขอผลลัพธ์
5. ประมวลผลและจัดการกับข้อผิดพลาด
6. รายงานผลให้กับผู้ใช้ (ถ้าจำเป็น)
7. ร้องขอการคอมมิท (Commit) หรือ โรลแบ็ค (Roll back) สำหรับควบคุมการประมวลผลทรานแอคชั่น (Tranaction)
8. ยกเลิกการติดต่อกับแหล่งข้อมูล

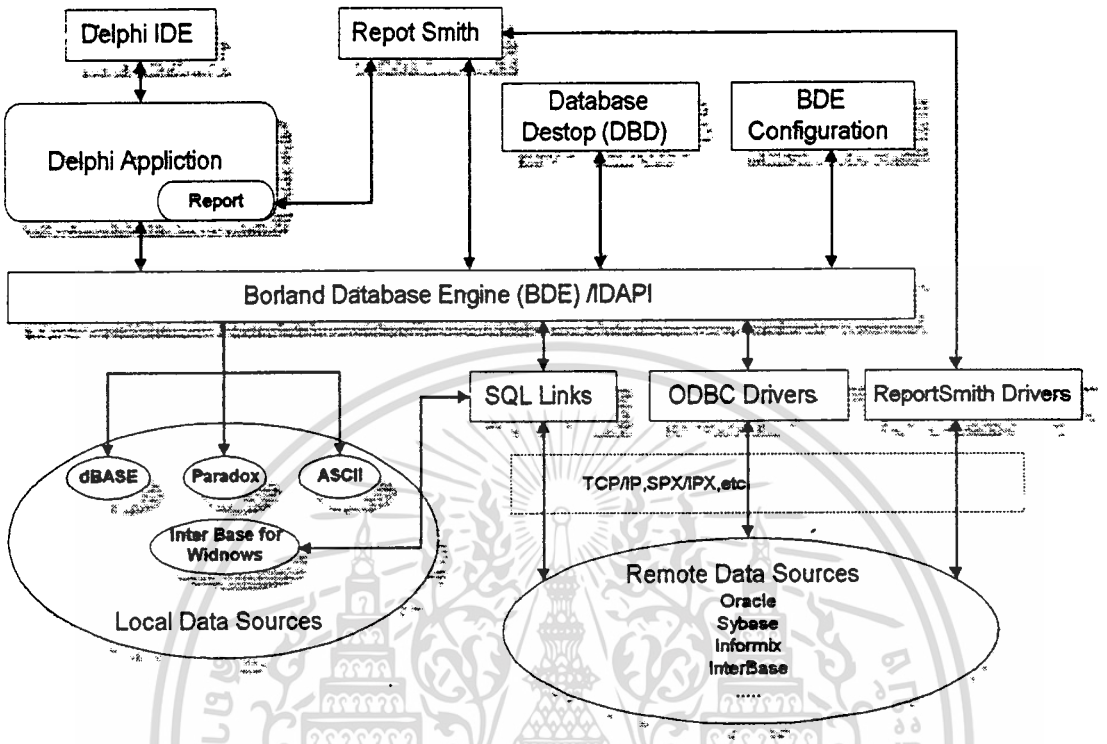
บทที่ 3
แนะนำซอฟต์แวร์

3.1 โปรแกรมเดลไฟ (Borland Delphi for Windows)

เดลไฟ(Delphi) เป็นเครื่องมือสำหรับเขียโปรแกรมเพื่อสร้างแอพพลิเคชันบนวินโดวส์ โดยภาษาที่ใช้ในการเขียนโปรแกรมจะเป็นภาษาปาสคาล ที่มีการสนับสนุนการเขียนโปรแกรมเชิงวัตถุ (Object Oriented Programming) ดังนั้นในการเขียนโปรแกรมขนาดใหญ่สามารถที่จะใช้คุณสมบัติ Encapsulation มาช่วยในการควบคุมการเข้าถึงตัวแปรต่างๆได้ดี และได้มีการรวมคอมไพเลอร์ ลิงค์เคอร์ เอดิเตอร์ และส่วนที่ใช้จัดการฐานข้อมูลเข้าด้วยกันและการเขียนโปรแกรมจะเป็นลักษณะของวิชวลโปรแกรมมิ่ง โดยจะมีลักษณะที่สำคัญดังนี้

1) ขั้นตอนของการเตรียมโปรแกรมมีขั้นตอนที่ง่ายโดยเฉพาะด้านฐานข้อมูล
2) เดลไฟจะคอมไพล์โปรแกรมเป็นไฟล์ .EXE อย่างแท้จริง ไม่มี p-code ซึ่งจะต้องแปลในแบบการตีความ(interprete) อีกในขณะรัน จึงรันได้กว่า 10-20 เท่า
3) เป็นการเขียนโปรแกรมในแบบโอโอพี (OOP) อย่างแท้จริง คือสามารถนำ โปรแกรมเก่ามาใช้ได้อีก
4) มีคอมโปเนนต์ (Component) ที่จะใช้ในการโปรแกรมให้เลือกมากกว่า 75 ตัว ตั้งแต่คอนโทรลธรรมดา เช่น ปุ่มควบคุมและปุ่มวิทยุ(Radio Button) ระดับกลาง เช่น นาฬิกา และกรอบรายการไฟล์ และที่ซับซ้อน ได้แก่ คอมโปเนนท์ทางด้านฐานข้อมูล
5) ใช้ภาษาเดลไฟปาสคาล ในการสร้างคอมโปเนนต์และใช้เขียนโปรแกรมซึ่งสามารถใช้ติดต่อกับ OLE 2.0(โอแอลอี 2.0) , DDE(ดีดีอี) , วีบีเอกซ์(VBX) และ โอดีบีซี ได้
6) สามารถเขียนโปรแกรมติดต่อกับฐานข้อมูลแยกได้เป็น 3 ทางคือ
    6.1) ติดต่อโดยตรงกับฐานข้อมูลภายในคือไฟล์ของ ดีเบส , พาราดอก และอินเตอร์เบสของบอร์แลนด์
    6.2) ผ่านทางการเชื่อมต่อโดยเอสคิวแอล ลิงค์ (SQL Link) เพื่อการติดต่อกับระบบฐานข้อมูลภายนอกคือ ออราเคิล , ไซเบส , อินฟอร์มิก , ไมโครซอฟท์ เอสคิวแอลเซริฟเวอร์ และอินเตอร์เบส ที่เรียกว่า ไคล์เอ็นต์เซอร์ฟเวอร์
    6.3) ผ่านทางโอดีบีซีไดร์เวอร์ เพื่อการติดต่อกับระบบฐานข้อมูลภายนอกคือ บีทีฟ , ไมโครซอฟท์แอกเซส และ ดีบีทู ซึ่งก็จัดอยู่ในจำพวกไคล์เอ็นต์เซอร์ฟเวอร์เช่นกัน

การติดต่อระหว่างเดลไฟและฐานข้อมูลจะแสดงได้ดังรูป



ปัจจุบันเดลไฟมีให้ใช้งานได้ 2 ชุด คือ

1. ชุดเดสค์ทอป (Delphi Destop) สามารถที่จะจัดการกับระบบฐานข้อมูลได้ด้วยไฟล์ของ ดีเบส พาราดอก และ อินเตอร์เบส ซึ่งเป็นฐานข้อมูลแบบฐานข้อมูลภายใน (Local Database) และสามารถติดต่อกับระบบฐานข้อมูลในแบบไคล์เอ็นต์เซอร์ฟเวอร์ได้โดยผ่านทางโอดีบีซี ไดร์เวอร์ โดยเดลไฟจะทำการเพิ่ม โอดีบีซี เมนเนเจอร์ ใน คอนโทรล เพนเนล (Control Panel) ของวินโดวส์ให้โดยอัตโนมัติ เพื่อใช้ในการตั้งและแก้ไขค่าคอนฟิกซ์ (Config) ของโอดีบีซี คาต้าซอร์ส และตัวของเดลไฟ จะมี โอดีบีซี ไดร์เวอร์ ให้เพื่อใช้กับฐานข้อมูล ดังนี้

   - Access data (*.mdb)
   - Access files (*.mdb)
   - Borland InterBase
   - Btrieve Data (file.ddf)
   - Btrieve files (files.ddf)
   - dBase files (*.dbf)
   - Excel files (*.xls)
   - Foxpro files (*.dbf)
   - Q + E Paradox file
   - Textfiles (*.txt,*.csv)

2. ชุดไคล์เอ็นต์เซอร์ฟเวอร์ จะสามารถใช้ติดต่อกับระบบฐานข้อมูลแบบไคล์เอ็นต์เซอร์ฟเวอร์ โดยผ่านเนทีฟ ไดร์เวอร์ ของระบบจัดการฐานข้อมูลแต่ละตัวโดย เนทีฟ ไดร์เวอร์ ที่ให้มากับ เดลไฟไคล์เอ็นต์เซอร์ฟเวอร์ คือ ออราเคิล , อินฟอร์มิก , ไซเบส และ อินเตอร์เบส

3.1.1 ลักษณะต่าง ๆของเดลไฟ

- โอโอพีที่เกี่ยวข้องกับเดลไฟ

เมื่อจะใช้งานออปเจ็กต์ในเดลไฟจะต้องสร้างสกุลก่อน กล่าวคือ

1. การสร้างสกุล แบ่งออกเป็น 2 ภาค คือ
   - 1.1 Declaration part คือ การกำหนดสกุล(Class) เช่นเดียวกับการกำหนดแบบข้อมูลเรคคอร์ด แต่จะประกอบด้วย
     - ฟิลด์ข้อมูล (attribute หรือ properties)
     - ฟิลด์กิจกรรม (method) ชื่อกิจกรรม คือ ชื่อโปรแกรมย่อย
   - 1.2 Implementation part คือ การเขียนโปรแกรมของกิจกรรม
2. เมื่อสร้างสกุลแล้วนำไปใช้ได้โดย
   - 2.1 กำหนดอินสแตนซ์ (instance) ซึ่งก็คือ ตัวแปร แต่แตกต่างตรงที่ยังให้ค่าไม่ได้
   - 2.2 ต้องให้กำเนิดอินสแตนซ์เป็นออปเจ็กต์ เช่น ด้วย Create (ชื่ออินสแตนซ์กับชื่อออปเจ็กต์คือชื่อเดียวกัน)
   - 2.3 เมื่อให้กำเนิดออปเจ็กต์แล้วจึงจะใช้งานได้ ด้วยการให้ค่าออปเจ็กต์ หรือส่งสื่อความ (message) ไปให้ สื่อความคือ ชื่อกิจกรรมของออปเจ็กต์นั้น ซึ่งก็คือเป็นการเรียกใช้โปรแกรมย่อยของออปเจ็กต์นั้น แต่ถือว่าออปเจ็กต์จะเป็นจริงในขณะรันเท่านั้น ตามที่กล่าวมานี้เป็นการอนุโลมเรียก

เมื่อให้กำเนิดออปเจ็กต์แล้ว จึงจะให้ค่าและส่งเมสเสจให้ได้ในโปรแกรม ซึ่งก็คือในขณะรัน จะมีวิธีเขียนโปรแกรม ดังนี้

```
Button1.Caption := 'No K';
Image1.LoadFromFile('ship.bmp');
```

- คำสั่งแรกเป็นการให้ค่า Caption ของ Button1 เป็น No K เท่ากับเป็นการเปลี่ยนชื่อปุ่ม
- คำสั่งที่สองเป็นการส่งสื่อความ LoadFromFile ให้แก่ Image1 เท่ากับเป็นการเรียกใช้กิจกรรมนี้ของ Timage1 ให้โหลดข้อมูลจากไฟล์ ship.bmp ออกแสดง

- ชื่อของสกุล คอมโปเนนต์ และ ออปเจ็กต์

ในเดลไฟตั้งชื่อสกุลโดยให้ขึ้นต้นด้วยอักษร T เช่น Timage เมื่อนำไปทำเป็นคอมโปเนนต์ เดลไฟจะกำหนดอินสแตนซ์และให้กำเนิดเป็นออปเจ็กต์แทนเรา ให้ชื่อตามชื่อคอมโปเนนต์ต่อท้ายด้วยหมายเลข เช่น เป็น Image1 และ Image2 เป็นต้น เพราะคอมโปเนนต์หนึ่งๆสามารถกำหนดเป็นออปเจ็กต์ได้หลายตัว และชื่อนี้จะเป็นค่าของ Name ซึ่งเป็นค่าพรอพเพอร์ตี้ส์ (ในหน้า Properties)

- ฟอร์มกับหน้าต่าง

ฟอร์ม คือ แบบฟอร์มให้เรากำหนดคอมโปเนนต์ แต่เมื่อรันจะเปลี่ยนฐานะเป็นหน้าต่างตามความหมายของวินโดวส์

- คอนโทรล

คอมโปเนนต์แบ่งออกเป็น 2 พวก คือ

1. ที่เป็นคอนโทรล จะแสดงตัวในขณะรัน จึงต้องการทั้งตำแหน่งและขนาดในฟอร์มด้วยการตีกรอบ (หากเพียงแต่คลิ๊กเม้าส์จะได้ขนาดมาตราฐาน)

2.ที่ไม่เป็นคอนโทรล (เช่นพวก Data Access) ไม่แสดงตัวในขณะรันจึงไม่ต้องการทั้งตำแหน่งและขนาด แต่ต้องกำหนดในฟอร์ม ซึ่งจะเห็นภาพคอมโปเนนต์นั้นในขณะเตรียมโปรแกรม (สามารถทับกันกับคอมโปเนนต์อื่นได้) โดยที่คอนโทรลปรากฏต่อสายตาของผู้ใช้ ผู้ใช้จึงรู้จักในฐานะของคอนโทรล

- ชื่อคอนโทรล กับค่า Caption และ Text

คอมโปเนนต์ที่เป็นคอนโทรลจำนวนหนึ่งมีค่าพรอพเพอร์ตี้ส์เป็นค่า Caption หรือ Text ค่าที่ให้แก่ Caption หรือ Text คือค่าที่จะไปปรากฏเป็นชื่อของคอนโทรล หากไม่มีพรอพเพอร์ตี้ส์ทั้งสองนี้ก็จะเป็นคอนโทรลที่ไม่มีชื่อ เช่น กรอบรายการ

การที่แบ่งเป็นค่า Caption และ Text แบ่งตามสายตาของผู้ใช้ โดยที่ผู้ใช้จะเห็นค่าที่เราให้แก่ Caption หรือ Text เป็นดังนี้

- ค่า Caption คือ ชื่อของสิ่งต่างๆ เช่น ชื่อหน้าต่าง ชื่อกรอบ และ ชื่อปุ่ม เป็นต้น ซึ่งผู้ใช้ป้อนแก้ไม่ได้
- ค่า Text เป็นค่าที่ให้ผู้ใช้ป้อนแก้ เช่น ในกรอบอีดิต

- ชื่อออปเจ็กต์และชื่อคอนโทรล

เมื่อเรากำหนดคอมโปเนนต์ เช่น ให้เป็นปุ่มควบคุม 3 ปุ่ม เดลไฟจะให้กำเนิดออปเจ็กต์ตามที่กล่าวมา ให้ชื่อออปเจ็กต์เป็น Button1,Button2 และ Button3 และใช้ชื่อนี้เป็นชื่อของคอนโทรลด้วย ซึ่งปรากฏชื่อนี้ในที่ต่างๆ ดังนี้

1. ชื่อออปเจ็กต์ คือ
   - ที่กรอบรายการออปเจ็กต์ (เป็นกรอบคอมโบอยู่ใต้ไตเติลบาร์ของหน้าต่าง Object Inspector) พร้อมด้วยชื่อสกุล
   - เป็นค่า Name ซึ่งถ้าเราต้องการเปลี่ยนชื่อออปเจ็กต์ ให้ทำได้โดยให้ชื่อใหม่แก่ Name
   - เป็นชื่ออินสแตนซ์
   - เป็นส่วนหนึ่งของชื่อกิจกรรม
2. ชื่อของคอนโทรล คือ
   - เป็นค่า Caption หรือ Text ซึ่งถ้าต้องการเปลี่ยนชื่อของคอนโทรล ให้ให้ชื่อใหม่แก่ Caption หรือ Text
   - ปรากฏเป็นชื่อของคอนโทรล (คือเป็นภาพชื่อ)

ในการโปรแกรมจะต้องเปลี่ยนชื่อของคอนโทรลให้สื่อความหมายกับผู้ใช้ เช่น อาจจะตั้งชื่อปุ่มเป็น Load,Save หรือ Clear เป็นต้น และเช่นกันควรตั้งชื่อออปเจ็กต์ให้จำง่ายในการเขียนโปรแกรม

เมื่อเปลี่ยนชื่อออปเจ็กต์คือเมื่อให้ค่าแก่ Name ค่าทั้งหมดตามที่กล่าวมานี้ รวมทั้งชื่อของคอนโทรลจะเปลี่ยนตาม แต่ถ้าให้ชื่อใหม่แก่คอนโทรลแล้วจะไม่เปลี่ยนตาม

- เขียนโปรแกรมกับเหตุ

การเขียนโปรแกรมกับวินโดวส์เป็นการเขียนโปรแกรมกับเหตุ (Event Driven Porgramming) โดยเหตุส่วนมากเกิดจากผู้ใช้ เช่น

- เมื่อผู้ใช้คลิ๊กเม้าส์ (จะให้ทำอะไร)
- เมื่อผู้ใช้เลือกรายการในเมนู (จะให้ทำอะไร)

โปรแกรมจึงไม่เป็นผืนเดียวติดต่อกัน และเมื่อจบจากการทำงานกับโปรแกรมกับเหตุหนึ่งๆ จะไปทำงานที่วินโดวส์ ซึ่งวินโดวส์อาจทำงานของตนเอง เช่น จัดระเบียบหน่วยความจำ เก็บข้อมูลลงไฟล์ หรือ ไปทำงานในแอพพลิเคชันอื่นตามความต้องการของผู้ใช้

- กรอบคำแนะ

เมื่อเลื่อนเคอร์เซอร์ไปทับคอมโปเนนต์ใดในกล่องอุปกรณ์ หรือที่สปีดบาร์ จะแสดงกรอบคำแนะ(Hint) เป็นกรอบสีเหลืองเล็กๆ แสดงชื่อของสิ่งนั้นๆ หรือบางกรณีเป็นการอธิบายย่อ ซึ่งจะใช้วิธีนี้ในการหาคอมโปเนนต์หรือชื่อปุ่มที่สปีดบาร์

- ยูนิต

ในการเขียนโปรแกรมโดยทั่วไปสกุลไม่จำเป็นต้องอยู่ในยูนิต แต่ของเดลไฟกำหนดให้อยู่ในยูนิต โดยที่ในยูนิตหนึ่งๆอาจมีหลายสกุลได้

ตามลิสติงที่ 1 แสดงถึงการสร้างสกุล TForm1 ไว้ในยูนิต Myfirst1 และเพราะอยู่ในยูนิตที่คู่กับฟอร์ม เดลไฟจึงเป็นผู้สร้างสกุลเกือบทั้งหมดแทนเรา กล่าวคือ

1. กำหนดให้ TForm1 สืบสกุลจาก TForm เพื่อใช้โปรแกรมที่มีอยู่ใน Tform คือที่บ่งใช้ใน uses (ไม่ทั้งหมด) ทำให้เราเขียนโปรแกรมได้สั้น
2. เมื่อเรากำหนดคอมโปเนนต์ในฟอร์ม เดลไฟจะกำหนดอินสแตนต์ของคอมโปเนนต์นั้นในสกุลเป็นฟิลด์ข้อมูลฟิลด์หนึ่งๆ คือ Button1 และ Lable1
3. เมื่อเราเลือกอีเวนต์ OnClick ของ Button1 เดลไฟจะกำหนดกิจกรรม Button1Click ให้ และให้เราป้อนโปรแกรมของกิจกรรมนี้ในส่วน Implementation ข้างล่าง

ซึ่งถ้าเราสร้างสกุลของเรา คือ เมื่อเปิดยูนิตโดยไม่มีฟอร์ม เราจะต้องป้อนโปรแกรมเช่นนี้ด้วยตนเอง

ลิสติง 1

```
unit Myfirst;
interface
uses
    SysUtils , WinTypes , WinProcs , Messages , Classes
    Graphics , Controls , Forms , Dialogs , StdCtrls;
type
    TFrom1  = class(TForm)
        Button1 : TButton;
        Lable1  :  TLable;
    procedure ButtonClick(Sender : TObject);
    end;
var
    Form1 : TForm1;
implementation
{$R  *.DFM}
```

```
procedure TForm1.ButtonClick(Sender : TObject);
        begin
            Lable1.Caption := 'Welcome to Delphi' + #13#10 + ' for windows.';
        end;
end.
```

3.1.2 การเตรียมโปรแกรมเดลไฟ

การเตรียมโปรแกรมแยกออกเป็น 3 ขั้นตอนย่อย คือ

1. กำหนดคอมโปเนนต์ โดยเลือกคอมโปเนนต์จากกล่องอุปกรณ์ แล้วนำไปกำหนดตำแหน่งและขนาดในฟอร์มด้วยการคลิ๊กเม้าส์หรือตีกรอบ จะได้เป็นออปเจ็กต์ในฟอร์ม (หมายถึงภาพออปเจ็กต์ในฟอร์ม)

2. ให้ค่าพรอพเพอร์ตี้ส์ โดยเลือกออปเจ็กต์แล้วเลือกหน้า Properties หรือกลับกัน แล้วให้ค่าพรอพเพอร์ตี้ส์ตามต้องการ ด้วยการให้ค่าเฉพาะ (ตามที่กล่าวมา)

3. ป้อนโปรแกรม โดยเลือกออปเจ็กต์แล้วเลือกหน้า Event หรือกลับกัน แล้วดับเบิลคลิ๊ก ณ. อีเวนต์ที่ต้องการ จะแสดงโครงโปรแกรมของกิจกรรมให้ป้อนโปรแกรม

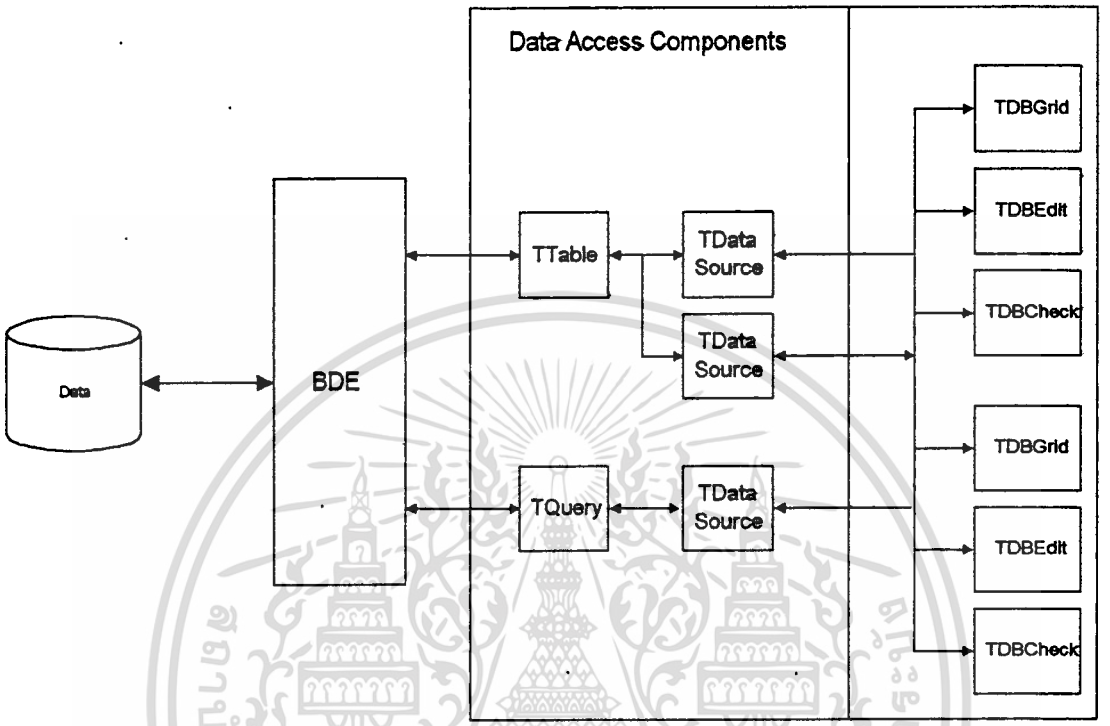
ในเดลไฟจะเรียกการทำงานในขั้นตอนทั้งสามนี้ว่า During design time

3.1.3 การติดต่อเดลไฟกับฐานข้อมูล

การใช้เดลไฟติดต่อกับฐานข้อมูลจะกระทำได้โดยกำหนดคอมโปเนนต์ชุดหนึ่งประกอบด้วยคอมโปเนนต์ 3 ชนิด คือ

1. ดาต้าเซต (Data Set) เป็นตัวแทนของเทเบิลเมื่อมองในแบบโอโอพี ได้แก่
   - TTable เพื่อการโปรแกรมด้วยเดลไฟปาสคาล
   - TQuery เพื่อการโปรแกรมด้วยเดลไฟปาสคาล
   - TStoredProc เพื่อใช้ในการติดต่อกับ Stored Procedure
2. ดาต้าซอร์ส (Data Source) เป็นตัวกลางเชื่อมระหว่างดาต้าเซตกับดาต้าอะแวร์ มีเพียงชนิดเดียว คือ ดาต้าซอร์ส
3. ดาต้าอะแวร์ (Data Aware) เป็นภาคแสดงผล มีหลายชนิด ดังนี้
   - DBGrid เพื่อการแสดงผลในรูปช่องตาราง
   - DBEdit เพื่อการแสดงผลเป็นฟิลด์
   - DBMemo เพื่อการแสดงผลเป็นฟิลด์ที่เป็น memo (เป็นข้อความหลายๆ บรรทัด)
   - DBImage เพื่อการแสดงผลเป็นฟิลด์ที่เป็นรูปภาพ
   - DBLookupList เพื่อการแสดงผลเป็นตาราง เป็นต้น

คอมโปเนนต์ชุดนี้คือที่ประกอบด้วย ดาต้าเซต ดาต้าซอร์ส และดาต้าอะแวร์ โดยในเดลไฟไม่ได้กำหนดชื่อเรียกไว้ มีแต่ที่รวมเรียก ดาต้าเซตกับดาต้าซอร์ส เป็น Data Access ดังรูป



3.1.4 ตัวอย่างการใช้งานกับฐานข้อมูล

1.การแสดงผลในแบบฟอร์มตาราง

1.1 กำหนดคอมโปเนนท์เป็น Data Access:table และ Data Access:DataSource และ DataControls:DBGrid ได้เป็นออปเจกต์ 3 ตัวคือ Table1 , DataSource1 และ DBGrid1 โดยที่ DBGrid1 คือตัวแบบฟอร์มตารางดังที่อยู่ในส่วนล่างของรูป ส่วน DataSource1 และ Table1 จะไม่แสดงในขณะรันจึงไม่ต้องกำหนดขนาด

1.2 ให้ค่าพรอพเพอร์ตี้ของออปเจกต์ต่างๆตามลำดับดังนี้คือ

-ค่า DataSource ของ DBGrid1 เป็น DataSource1

-ค่า DataSet ของ DataSource1 เป็น Table1

-ค่า DatabaseName ของ Table1 เป็น BIOLIFE.DB

-ค่า Active ของ Table1 เป็น True

จากนั้นก็จะนำข้อมูลออกแสดงในแบบฟอร์มตารางทันที และเมื่อรันจะได้ผลรัยเช่นเดียวกัน

2. การแสดงผลด้วยเอสคิวแอล

2.1 กำหนดคอมโปเนนท์เป็น Data Access:Query และ DataAccess:DataSource กับ DataControls:DBGrid แต่ยังไม่แสดงข้อมูล

2.2 ให้ค่าพรอพเพอร์ตี้ของออปเจกต์ต่างๆเช่นเดียวกับตัวอย่างที่แล้ว โดยเปลี่ยนจาก Table1 เป็นQuery1 และสำหรับ Query1 ให้ค่าเฉพาะ DatabaseName เท่านั้นก่อน แล้วให้ค่าพรอพเพอร์ตี้ของ Query1 ต่อดังนี้

- ให้ค่า SQL เป็นประโยค SQL
- ให้ค่า Active เป็น True

จากตัวอย่างทั้ง 2 นี้สามารถให้กับคอมโปเนนท์อื่นนอกจากตารางได้ และสารถเขียนโปรแกรมแทนการให้ค่าได้

3.1.5 ระบบไฟล์ของเดลไฟ

แอพพลิเคชันที่พัฒนาด้วยเดลไฟจะพัฒนาตามหลักการของโปรเจ็ค โดยเดลไฟโปรเจ็ค (Delphi Project) คือ กลุ่มของไฟล์ที่ประกอบกันเป็นเดลไฟแอพพลิเคชัน โดยบางไฟล์อาจจะถูกสร้างขณะ ดีไซน์ไทมร์ (design time) และบางไฟล์จะถูกสร้างขึ้นมาขณะคอมไพล์โปรเจ็คซอร์สโค้ดของเดลไฟ

โปรเจ็คไฟล์จะประกอบด้วยไฟล์ต่างๆ ดังนี้

- ไฟล์ที่ถูกสร้างขณะออกแบบโปรแกรม จะประกอบด้วย
  - .DPR Project file
  - .PAS Unit source (Object code)
  - .DFM Graphical form file
  - .OPT Project options file
  - .RES Compiler resource file
  - .DSK Desktop setting
- ไฟล์ที่ถูกสร้างขึ้นจากคอมไพล์เลอร์
  - .EXE Compiler executable file
  - .DCU Unit object code
  - .DLL Compiled dynamic-link library

3.1.6 การนำแอพพลิเคชันของเดลไฟไปใช้งาน

การนำแอพพลิเคชันของเดลไฟไปใช้งาน โดยเครื่องที่นำไปใช้นั้นไม่มีการติดตั้งเดลไฟไว้ ซึ่งการใช้งานจะต้องการสิ่งต่าง ๆ ดังต่อไปนี้

1. แอพพลิเคชัน .EXE ไฟล์ และ .DLL ต่าง ๆที่แอพพลิเคชันมีการเรียกใช้
2. บอร์แลนด์ ดาต้าเบส เอ็นต์จิน (Borland Database Engine) ในกรณีที่เป็นแอพพลิเคชันที่เกี่ยวข้องกับฐานข้อมูล
3. เอสคิวแอล ลิงค์ ซัตพอร์ต (SQL Link Support) ในกรณีที่เป็นแอพพลิเคชันที่มีการติดต่อกับระบบฐานข้อมูลแบบไคล์เอ็นต์เซอร์ฟเวอร์
4. รีพอร์ตสมิทช์ รันไทม์ ซัพพอร์ต(ReportSmith Runtime Support) ในกรณีที่เป็นแอพพลิเคชันที่มีการใช้งาน รีพอร์ต (Reports)
5. อินเตอร์เบส เอสคิวแอล ลิงค์ (InterBase SQL Link) ในกรณีที่แอพพลิเคชันมีการติดต่อกับ โลคอล อินเตอร์เบส เซอร์ฟเวอร์ (Local InterBase Server) หรือ อินเตอร์เบส เวอร์กกรุ๊ป เซอร์ฟเวอร์ (InterBase Workgroup Servers)
6. โลคอล อินเตอร์เบส เซอร์ฟเวอร์ ในกรณีที่แอพพลิเคชันมีการติดต่อกับ โลคอล อินเตอร์เบส เซอร์ฟเวอร์

โดยที่เดลไฟจะให้ไฟล์ต่าง ๆเหล่านี้มาในรูปของ ไฟล์ที่มีการบีบอัด (Compressed files) ซึ่งสามารถนำไปติดตั้งยังเครื่องที่ต้องการจะใช้งานแอพพลิเคชันของเดลไฟและเมื่อมีการติดตั้งไฟล์ต่าง ๆเหล่านี้ จะมีการแก้ไขไฟล์ของระบบบางไฟล์ เช่น ไฟล์ WIN.INI ให้โดยอัตโนมัติ

- แอพพลิเคชัน .EXE ไฟล์

แอพพลิเคชันที่สร้างขึ้นด้วยเดลไฟจะไม่ต้องอาศัย ตัวแปลประโยคขณะทำงาน (run-time interpreter) ดังนั้นการใช้งานแอพพลิเคชัน ที่สร้างขึ้นด้วยเดลไฟจะต้องการเพียงไฟล์ .EXE และ ไฟล์ .DLL ต่าง ๆที่แอพพลิเคชันมีการเรียกใช้ เช่น BIVBX.DLL สำหรับ แอพพลิเคชันที่มีการเรียกใช้ VBX คอนโทรล

- กลุ่มไฟล์ BDE (สำหรับทุก ๆแอพพลิเคชันที่มีการใช้งานฐานข้อมูล)

เป็นไฟล์ที่มีการบีบอัดที่อยู่ในไดเรกทรอรี BDEINST เมื่อมีการติดตั้งแล้วจะได้ไฟล์ต่าง ๆและโปรแกรม ดาต้าเบส เอ็นต์จิน คอนฟิกซ์กูเรชั่น (Database Engine Configuration) เพื่อใช้ในการกำหนดค่าและแก้ไขค่าเริ่มต้น ของ IDAPI.CFG

โดยไฟล์ต่าง ๆ ที่ได้จากการติดตั้งกลุ่มไฟล์ BDE คือ

IDAPI01.DLL	สำหรับเดลไฟแอพพลิเคชั่นที่ใช้งานฐานข้อมูล
IDR10009.DLL	สำหรับเดลไฟแอพพลิเคชั่นที่ใช้งานฐานข้อมูล
ILD01.DLL	สำหรับเดลไฟแอพพลิเคชั่นที่ใช้งานฐานข้อมูล

IDAPI.CFG	สำหรับเดลไฟแอพพลิเคชั่นที่ใช้งานฐานข้อมูล
IDQRY01.DLL	สำหรับ เอสคิวแอล คิวรี (SQL queries) (จะต้องใช้งานร่วมกับไฟล์ IDPDX01.DLL)
IDBAT01.DLL	สำหรับ เอสคิวแอล คิวรี และการทำ การเคลื่อนย้ายแบบเป็นกลุ่ม (Batch move operation)
IDPDX01.DLL	สำหรับการใช้งานกับ พาราดอก
IDDBAS01.DLL	สำหรับการใช้งานกับ ดีเบส
IDASCI01.DLL	สำหรับการใช้งานกับ เอสกี เทเบิล
IDODBC01.DLL	สำหรับการใช้งานกับ โอดีบีซี
ODBC.NEW	Microsoft ODBC Driver Manager DLL version 2.0
ODBCINST.NEW	Microsoft ODBC Driver installation DLL version 2.0
BDECFG.EXE	BDE Config Utility สำหรับการกำหนดและแก้ไขเอเลียส (aliases)
BDECFG.HLP	BDE Config Utility Help สำหรับการกำหนดและแก้ไขเอเลียส

- กลุ่มไฟล์ เอสคิวแอล ลิงค์ (ใช้สำหรับ ไคล์เอ็นต์เซอร์ฟเวอร์ แอพพลิเคชัน)

จะเป็นชุดติดตั้งที่มาพร้อมกับเวอร์ชัน ไคล์เอ็นต์ / เซอร์ฟเวอร์ ของเดลไฟโดยจะอยู่ในไดเรกทรอรี SQLINST เมื่อติดตั้งแล้วจะได้ไฟล์ต่าง ๆ ดังนี้

ไฟล์ที่ใช้ในการติดต่อกับ อินฟอร์มิกซ์

SQLD_INF.DLL
SQLD_INF.HLP
SQL_INF.CNF
LDLLSQLW.DLL
ISAM.IEM
OS.IEM
RDS.IEM
SECURITY.IEM
SQL.IEM
ZRDSTERM.IEM

ไฟล์ที่ใช้ในการติดต่อกับ ออราเคิล

SQLD_ORA.DLL
SQLD_ORA.HLP
SQL_ORA.CNF

ORA6WIN.DLL

ORA7WIN.DLL

COREWIN.DLL

ORAWE850.LD

NWIPXSPX.DLL

NWNETAPI.DLL

ไฟล์ที่ใช้ในการติดต่อกับ ไซเบส และ ไมโครซอฟต์ เอสคิวแอล เซอร์ฟเวอร์

SQLD_SS.DLL

SQLD_SS.HLP

SQL_SS.CNF

W3DBLIB.DLL

DBNMP3.DLL

SYDC437.LD

SYDC850.LD

NWIPXSPX.DLL

NWNETAPI.DLL

- กลุ่มไฟล์ รีพอร์ตสมิทช์ รันไทม์ (ReportSmith Runtime)

ใช้สำหรับแอพพลิเคชันที่มีการใช้งาน Treport คอมโปเนนต์ โดยจะประกอบด้วยไฟล์ต่าง ๆ ดังนี้

RED110.DLL	DRVACCSS.HLP	BTRV110.DLL
DRVBTRV.HLP	RCS0.RST-RCS8.RST	RSCTSFMT.RST
DRVDBASE.HLP	XBS110.DLL	RS_DBLIB.DLL
XLSISAM.DLL	DRVEXCEL.HLP	DRVFOX.HLP
RS_GUP.DLL	RS_INGR.DLL	B4RPT.MAC
DONDFMT.MAC	DATE.MAC	DISABLE.MAC
ENABLE.MAC	GREETING.MAC	ID2NAME.MAC
LOADREP.MAC	LOADREPS.MAC	RECNO.MAC
THETIME.MAC	CTL3D.DLL	MSJETDSP.DLL
ODBC.DLL	ODBCADM.EXE	ODBCINST.DLL
SIMADMIN.DLL	SIMBA.DLL	ODBCINST.HLP

COPOBJ.DLL	STDOLE.TLD	DRVTEXT.HLP
RS_ORA6.DLL	RS_ORA7.DLL	BIBAS04.DLL
BIFLT04.DLL	BIMDS04.DLL	BIUTL04.DL
ODBCCURS.DLL	PXENGCFG.EXE	PXENGWIN.DLL
BIPDX04.DLL	BIPDX04.HLP	QEBI.LIC
README.TXT	BC30RTL.DLL	DATALIB2.DLL
DLOLE2.DLL	IMAGEMAN.DLL	IMGPCX.DIL
IMGTIFF.DIL	RSSQLIF.TXT	RS_FMT.DLL
RS_IDABP.DLL	RS_ODBC.DLL	RS_OGLWL.DLL
RS_OLE2C.DLL	RS_OLE2U.DLL	RS_OWL31.DLL
RS_SIFUT.DLL	RS_SQLIF.DLL	RS_SYS.DLL
RS_SYS.DLL	RS_TBP1.DLL	RS_TBP2.DLL
RS_TCL31.DLL	RS_TKRIB.DLL	RS_DFWEN.DLL
RS_TKSTB.DLL	RS_TKTB.DLL	RS_UTIL.DLL
SSMRTHEAP.DLL	REPVAR.FRM	REPVAR.MAK
RSR_CTAB.DLL	RSR_DBAC.DLL	RSR_DBUI.DLL
RSR_MAIN.DLL	RSR_RPT.DLL	RSR_RUTL.DLL
RSR_XPRT.DLL	RSTEST.FRM	RSTEST.FRM
RS_RUN.EXE	RS_RUN.HLP	RUN_TEST.EXE
RS_SQLIF.INI	CTL3DV2.DLL	TXTISAM.DLL

- กลุ่มไฟล์ที่สนับสนุน โอเอลอี (OLE Support)

ใช้สำหรับแอพพลิเคชันที่มีการใช้งาน OLE Container Conponents โดยจะต้องมีการติดตั้งไฟล์ BOLE16D.DLL ซึ่งเป็น Borland's OLE2 Support DLL ลงในไดเรกทรอรี WINDOWS\SYSTEM และจะต้องมีการติดตั้ง Microsoft OLE DLLs ในไดเรกทรอรี WINDOWS\SYSTEM ซึ่งจะประกอบด้วยไฟล์ต่างๆ ดังนี้

LE2.DLL

LE2.REG

LE2CONV.DLL

LE2DISP.DLL

LE2NLS.DLL

LE2PROX.DLL

LECLI.DLL

LESVR.DLL

TORAGE.DLL

YPELIB.DLL

OMPOBJ.DLL

สำหรับ วินโดวส์ 3.1 ที่มีการติดตั้ง Microsoft OLE DLLs version 2.0 จะต้องทำการติดตั้งใหม่ เนื่องจากเดลไฟจะสนับสนุน โอเอลอี ในเวอร์ชัน 2.02

- อินเตอร์เบส เอสคิวแอล ลิงค์ และ ไคล์เอ็นต์ ไฟล์ (ใช้สำหรับ อินเตอร์เบส แอพพลิเคชัน)

สำหรับแอพพลิเคชันที่มีการติดต่อกับ โลคอล อินเตอร์เบส เซอร์ฟเวอร์ หรือ อินเตอร์เบส เวอร์กกรุ๊ป เซอร์ฟเวอร์ ซึ่งจะประกอบด้วยไฟล์ต่างๆ ดังนี้

SQLD_IB.DLL

SQLD_IB.HLP

SQL_IB.CNF

CONNECT.EXE

CONNECT.HLP

GDS.DLL

REMOTE.DLL

INTERBAS.MSG

IUTLS.DLL

DSQL.DLL

NWIPXSPX.DLL

NWCALLS.DLL

NWNETAPI.DLL

ในกรณีที่แอพพลิเคชันใช้การติดต่อกับเซอร์เวอร์สผ่านโปรโตคอล TCP/IP และเครื่องทางด้านไคล์เอ็นท์ใช้ winsock 1.1 จะต้องมีการติดตั้งไฟล์ต่างๆ ดังนี้

MVWASYNC.EXE

VSL.INI

WINSOCK.DLL

MSOCKLIB.DLL

ในกรณีไคล์เอ็นท์ที่ไม่มี winsock 1.1 จะสามารถใช้งานผ่าน TCP/IP อินเตอร์เฟส โดยใช้ไฟล์ต่างๆ ดังนี้

M3OPEN.EXE	3Com 3+Open TCP
	Microsoft LAN Manager
	Digital Pathworks for DOS
M3OPEN.DLL	3Com 3+Open TCP Version 2.0
MBW.EXE	Beame & Whiteside TCP/IP
MFTP.EXE	FTP PC/TCP
MHPARPA.DLL	HP ARPA Service for DOS
MNETONE.EXE	Ungermann-Bass Net/One
MNOVLWP.DLL	Novell LAN WorkPlace for DOS
MPATHWAY.DLL	Wollongong Pathway Access for DOS
MPCNFS.EXE	Sun PC NFS
MPCNFS2.EXE	Sun PC NFS v3.5
MPCNFS4.DLL	Sun PC NFS v4.0
MWINTCP.EXE	Wollongong WIN TCP\IP for DOS

Security, License Files

INTERBASE.MSG

ISC4.GDB

ISC_LIC.DAT

InterBase Utilities

WISQL.EXE

WISQL.HLP

SQLREF.HLP

IBMGR.EXE

SVRMGR.HLP

COMDIAG.EXE

COMDIAG.INI

COMDIAG.HLP

BLINT04.HLP

3.2 โปรแกรมเพาเวอร์บิวเดอร์ (PowerBuilder)

เพาเวอร์บิวเดอร์ เป็นเครื่องมือที่ใช้ในการสร้างแอพพลิเคชันบนวินโดวส์ ซึ่งภายในตัวโปรแกรมจะประกอบด้วย สิ่งที่เรียกว่า เพนเทอร์ (Painter) ซึ่งจะเป็นส่วนที่รวมแอพพลิเคชันคอมโปเนนต์ต่างๆไว้ เช่น วินโดวส์ , เมนู , ดิวรี และ รีพอร์พ รวมทั้งส่วนของ การออกแบบดาต้าเบส และ ส่วนของการทดสอบโปรแกรม โดยภาษาที่ใช้ในการเขียนโปรแกรมจะเรียกว่า เพาเวอร์สคริปต์ (PowerScript)

เพาเวอร์บิวเดอร์ จะสามารถสร้างแอพพลิเคชัน แล้วทำการคอมไพล์เป็นไฟล์ .EXE เพียงไฟล์เดียวและสามารถที่จะคอมไพล์ให้เป็น Dynamic Library .PBD หลายๆไฟล์ ซึ่งสามารถที่จะนำไปทำงานได้ โดยไฟล์ .EXE ที่ได้นั้นจะอยู่ในรูปแบบของ พี-โค้ด (P-Code : tokenized Pseudo-Code) ซึ่งจะต้องทำการแปล (Interpreted) ขณะทำงาน

แอพพลิเคชันที่สร้างจากเพาเวอร์บิวเดอร์ จะประกอบด้วยออปเจกต์ต่างๆ ประกอบกันโดยออปเจ็กต์แต่ละตัวจะมี พร็อพเพอร์ตี้ (Properties) และอีเวนต์ (Event) แตกต่างกันออกไป โดยที่พร็อพเพอร์ตี้จะเป็นคุณสมบัติ ประจำของออปเจ็กต์แต่ละตัว และอีเวนต์ จะเป็นเหตุการณ์ต่างๆที่ออปเจ็กต์นั้นๆ จะสามารถที่จะตอบสนองได้ ดังนั้นลักษณะของโปรแกรมที่สร้างขึ้นด้วยเพาเวอร์บิวเดอร์ จะเป็นการเขียนโปรแกรมเพื่อตอบสนองต่อเหตุการณ์ ต่างๆที่เกิดขึ้นกับออปเจ็กต์แต่ละตัว ซึ่งเรียกว่า Event Drivent Programming

ตั้งแต่เพาเวอร์บิวเดอร์ เวอร์ชัน 3 เป็นต้นมา เพาเวอร์บิวเดอร์ ได้มีการใช้ โค้ด (CODE : Client-server Open Development Environment) ซึ่งเป็น โอเพนต์ เอพีไอ (open API) ของเพาเวอร์บิวเดอร์ ไลบรารี เมนเนเจอร์ (PowerBuilder Library Manager) ซึ่งจะทำให้ผลิตภัณฑ์ของบริษัทอื่นๆสามารถที่จะใช้งานร่วมกับเพาเวอร์บิวเดอร์ ได้สะดวกยิ่งขึ้น เช่นการเขียน อีอาร์ โมเดล (ER-Model) ในโปรแกรม ERWIN/ERX สามารถที่จะใช้ ดาต้า ดิชชันนารี (Data Dictionary) ร่วมกับโปรแกรม เพาเวอร์บิวเดอร์ ได้

3.2.1 เพาเวอร์สคริปต์

เพาเวอร์สคริปต์เป็นภาษาที่ใช้ในการเขียนโปรแกรมของเพาเวอร์บิวเดอร์ โดยลักษณะของภาษาจะเป็นภาษาระดับสูง และจะเป็นการเขียนโปรแกรมเพื่อรองรับเหตุการณ์ที่เกิดขึ้น เพาเวอร์สคริปต์สามารถที่จะเรียกใช้ฟังก์ชันต่างๆที่เพาเวอร์บิวเดอร์ได้เตรียมไว้ให้ประมาณ 500 ฟังก์ชัน และเพาเวอร์สคริปต์จะสามารถเขียนโปรแกรมซึ่งมีการฝัง (embedded) ภาษาเอสคิวแอล เพื่อใช้ในการติดต่อกับฐานข้อมูลได้โดยตรง

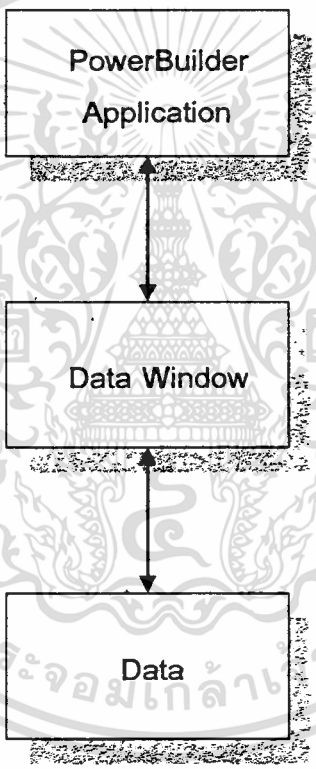
เพาเวอร์สคริปต์จะมีการอ้างถึงแอตทริบิวต์ของออปเจ็กต์ใดๆ ได้โดยการใช้สัญญลักษณ์ จุด (Dot Notation) เช่น

```
w_main.cb_close.enabled = FALSE
```

ในเพาเวอร์สคริปต์สามารถใช้ โฟร์ คอนโทรล สเตตเมนท์ (Flow Control Statement) และสามารถใช้คุณสมบัติขอบเขตการใช้งานของตัวแปร (Scope of Variable) ได้

3.2.2 ดาต้าวินโดวส์

ดาต้าวินโดวส์ เป็นคุณลักษณะที่สำคัญอย่างหนึ่งของเพาเวอร์บิวเดอร์ โดยที่ดาต้าวินโดวส์ออปเจ็กต์ จะเป็นสิ่งที่ใช้ในการติดต่อระหว่างฐานข้อมูลกับผู้ใช้ โดยจะสามารถใช้งานได้ในรูปแบบกราฟฟิค ทำให้สามารถใช้งานฐานข้อมูลทำได้ง่ายมากยิ่งขึ้น ดังนั้นดาต้าวินโดวส์ คือ สิ่งที่อยู่ระหว่างแอพพลิเคชันที่สร้างขึ้นกับระบบฐานข้อมูล โดยแอพพลิเคชันจะสามารถใช้งานระบบฐานข้อมูลโดยผ่านดาต้าวินโดวส์ ดังรูป



3.2.3 คุณสมบัติทางด้านการโปรแกรมเชิงวัตถุ (Object-Oriented Features)

เพาเวอร์บิวเดอร์จะสามารถเขียนโปรแกรมโดยใช้คุณสมบัติของการโปรแกรมเชิงวัตถุในหลายๆคุณลักษณะ ดังนี้

1. Inheritance เป็นคุณสมบัติที่ยอมให้สร้างออปเจ็กต์ใหม่ได้จากคลาสพื้นฐานเดิมที่มีอยู่แล้ว โดยคลาสใหม่ที่สร้างขึ้นสามารถที่จะใช้คุณสมบัติของคลาสเดิมทั้งหมด หรือ เลือกใช้เพียงบางส่วนก็ได้ รวมทั้งสามารถเพิ่มเติมคุณสมบัติใหม่เข้าไปได้ด้วย

2. Polymorphism เป็นคุณสมบัติที่ยอมให้มีการเรียกใช้ฟังก์ชันชื่อเดียวกัน แต่มีการทำงานที่แตกต่างกันไปตามแต่ละออปเจ็กต์ที่ฟังก์ชันนั้นทำงาน เช่น มีคลาสขอไฟล์และคลาสของไดเรกทรอรี โดยที่

ทั้งสองคลาสมีฟังก์ชัน delete ดังนั้นถ้าหากมีการเรียกใช้ฟังก์ชัน delete เพื่อใช้กับคลาสของไฟล์จะเป็นการลบไฟล์ แต่หากเรียกใช้ฟังก์ชันนี้เพื่อใช้กับคลาสของไดเรกทรอรี จะเป็นการลบไดเรกทรอรี

3.2.4 การใช้งานเพาเวอร์บิวเดอร์ติดต่อกับระบบฐานข้อมูล จะสามารถทำได้ 2 รูปแบบ คือ

1. เอมเบดเดด เอสคิวแอล ซึ่งจะมีการชังานได้ 3 รูปแบบ คือ
   - 1.1 with cursor ใช้กับข้อมูลที่มีการส่งค่ากลับมาเป็นกลุ่ม และจะใช้ข้อมูลเหล่านั้นได้โดยผ่านเดอร์เซอร์ (Cursor)
   - 1.2 without cursor ใช้กับข้อมูลที่มีการใช้งานร่วมกับตัวแปรโฮสต์ (Host Variable)
   - 1.3 stored procedure ใช้ในการติดต่อใช้งาน stored procedure ที่อยู่ที่ตัวระบบจัดการฐานข้อมูล
2. ดาต้าวินโดวส์ คือ ส่วนที่ใช้ในการติดต่อระหว่างระบบฐานข้อมูลกับผู้ใช้ โดยสามารถที่จะจัดรูปแบบการแสดงของข้อมูลที่ได้ในหลายๆรูปแบบ

3.2.5 เพาเวอร์บิวเดอร์เพนเทอร์

เพาเวอร์บิวเดอร์เพนเทอร์เป็นชุดของเครื่องมือที่เพาเวอร์บิวเดอร์ใช้ในการสร้างแอพพลิเคชัน โดยในตอนแรกที่เพาเวอร์บิวเดอร์เริ่มต้นทำงานจะเริ่มต้นที่วินโดวส์เพนเทอร์เสมอ โดยรายละเอียดของแต่ละเพนเทอร์จะมีรายละเอียด ดังนี้

- Application painter ใช้ในการกำหนดแอตทริบิวต่างๆของแอพพลิเคชัน และใช้ในการสร้างออปเจ็กต์ของแอพพลิเคชัน
- Window painter ใช้ในการสร้างวินโดวส์ ซึ่งจะเป็น ส่วนประกอบหนึ่งของแอพพลิเคชัน
- Menu painter ใช้ในการสร้างเมนู ซึ่งสามารถรวมเข้ากับบางวินโดวส์ได้
- Datawindow painter ใช้ในการสร้างและ กำหนดสภาวะแวดล้อมของดาต้าวินโดวส์
- Structure painter ใช้ในการกำหนดชุดของตัวแปรต่างๆ
- Preferences painter ใช้ในการกำหนดสภาวะแวดล้อมต่างๆ ของเพาเวอร์บิวเดอร์
- Help painter ใช้ในการใช้ระบบความช่วยเหลือของเพาเวอร์บิวเดอร์
- Database painter ใช้ในการติดต่อกับระบบฐานข้อมูล
- Query painter ใช้ในการสร้างประโยคเอสคิวแอล โดยจะอยู่ในรูปแบบกราฟฟิค
- Function painter ใช้ในการสร้างฟังก์ชันต่างๆ
- Project painter ใช้ในการสร้างโปรเจ็กต์ออปเจ็กต์
- Library painter ใช้ในการสร้างเพาเวอร์บิวเดอร์ไลบรารี

- User Object painter ใช้ในการกำหนดออปเจ็กต์ใหม่ขึ้นมา
- Debug ใช้ในการทดสอบความถูกต้องและ แก้ไขโปรแกรม ที่สร้างขึ้น

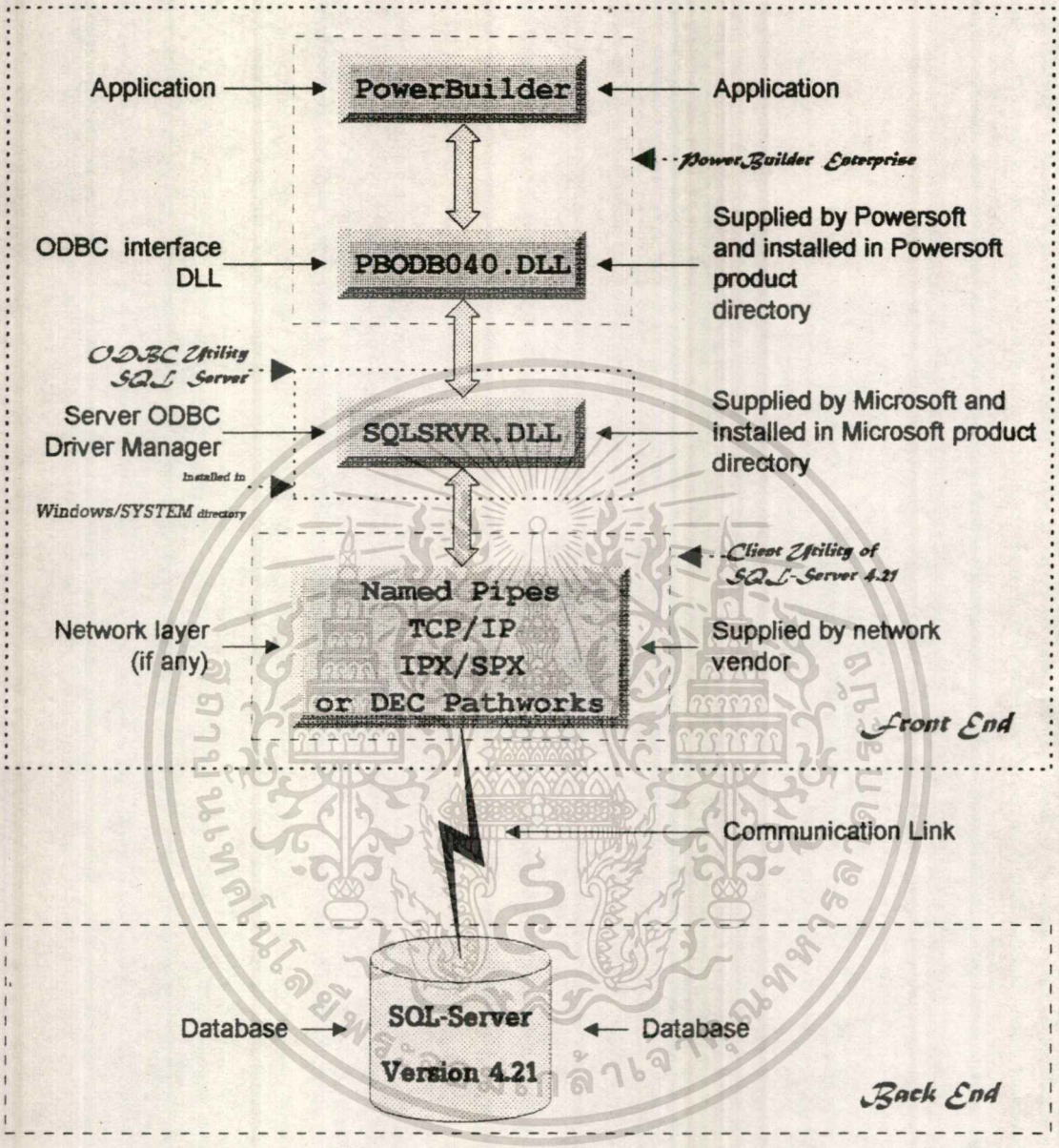
โค้ดบางส่วนของโปรแกรมเพาเวอร์บิวเดอร์ในส่วนการติดต่อกับดาต้าเบส

```
r_date          = 32
r_sale          = rand(10)+1
r_custom        = rand(100)+1
r_quantity      = (rand(10000)+1
r_price         = (rand(600) * rand(100))+1
r_part          = (rand(100) * rand(100) * rand(100))+1
UPDATE corder SET corder# = corder# + 1 ;
 SELECT corder.corder# INTO :r_porder# FROM corder  ;
commit;
INSERT INTO head_odbc_noindex ( porder#,dates,sale#,custom#,station)
          VALUES ( :r_porder#, :r_date, :r_sale, :r_custom, :char_ ) ;
INSERT INTO detail_odbc_noindex ( porder#, part#, quantity, price )
          VALUES ( :r_porder#, :r_part, :r_quantity, :r_price ) ;
commit;
```

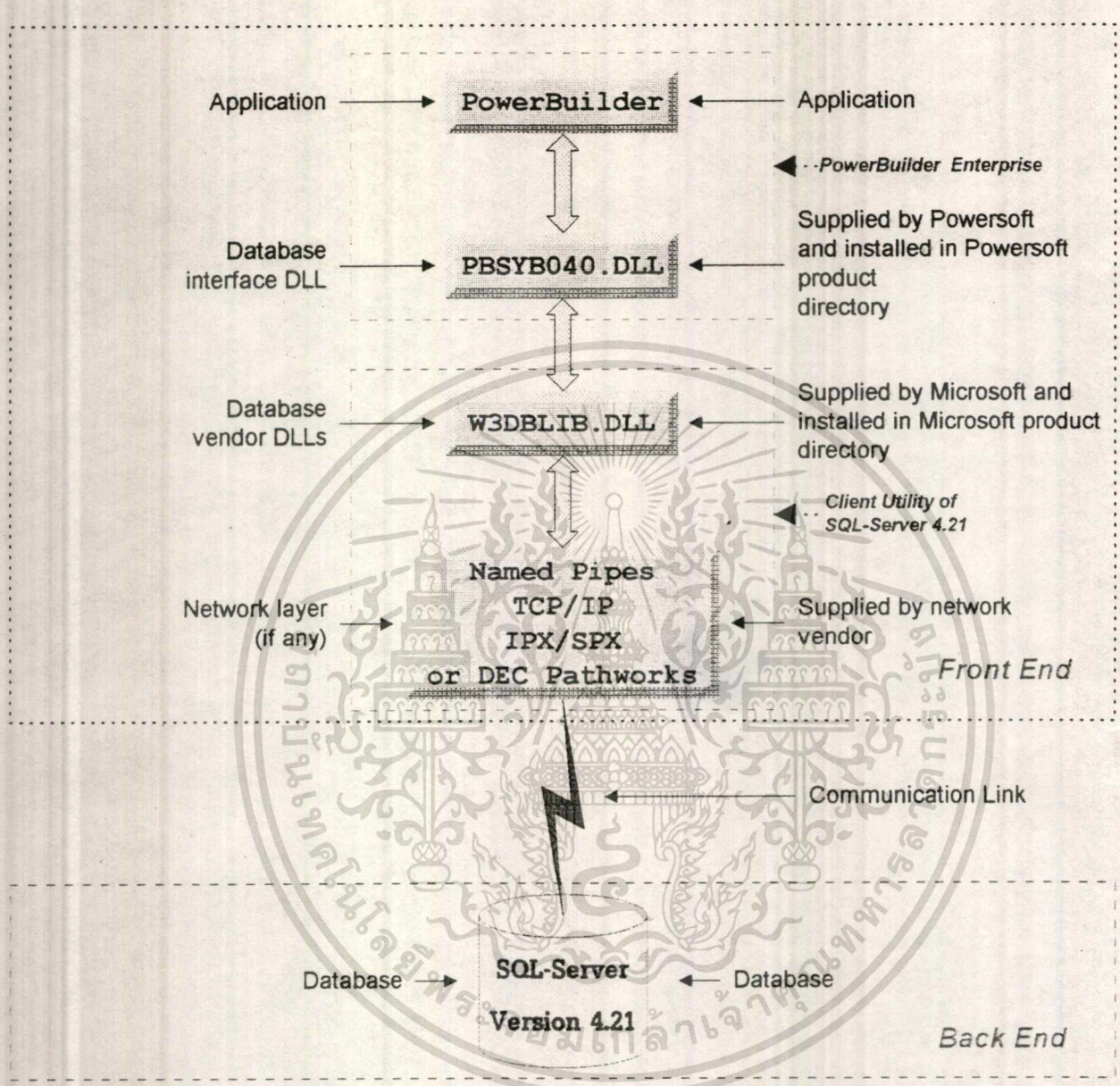
โค้ดบางส่วนของโปรแกรมเดลไฟในส่วนการติดต่อกับดาต้าเบส

```
h_date := 32;
h_sale := random(10)+1;
h_custom := random(100)+1;
d_part1 := random(40001)+1;
d_part2 := random(60001)+1;
d_part := d_part1 + d_part2;
d_quantity := random(10000)+1;
d_price := random(60000)+1;
f_Ttime := gettickcount;
close;
sql.clear;
sql.add('  update corder
     set corder# = corder# + 1');
execsql;
close;
sql.clear;
sql.add('select * from corder');
open;
query1.first;
porder := query1.fields[0].asinteger;
close;
sql.clear;
sql.add('commit');
execsql; {retrive number
   from corder# complete  }
     close;
     sql.clear;
sql.add(format(statement_h,[table1]));
     params[0].asinteger := porder;
     params[1].asinteger := h_date;
     params[2].asinteger := h_sale;
     params[3].asinteger := h_custom;
     params[4].asstring :=  station;
     execsql;
     close;
     sql.clear;
     sql.add('commit');
     execsql;
     close;
     sql.clear;
sql.add(format(statement_d,[table2]));
     params[0].asinteger := porder;
     params[1].asinteger := d_part;
     params[2].asinteger := d_quantity;
     params[3].asinteger := d_price;
     execsql;
     close;
     sql.clear;
     sql.add('commit');
     execsql;
```

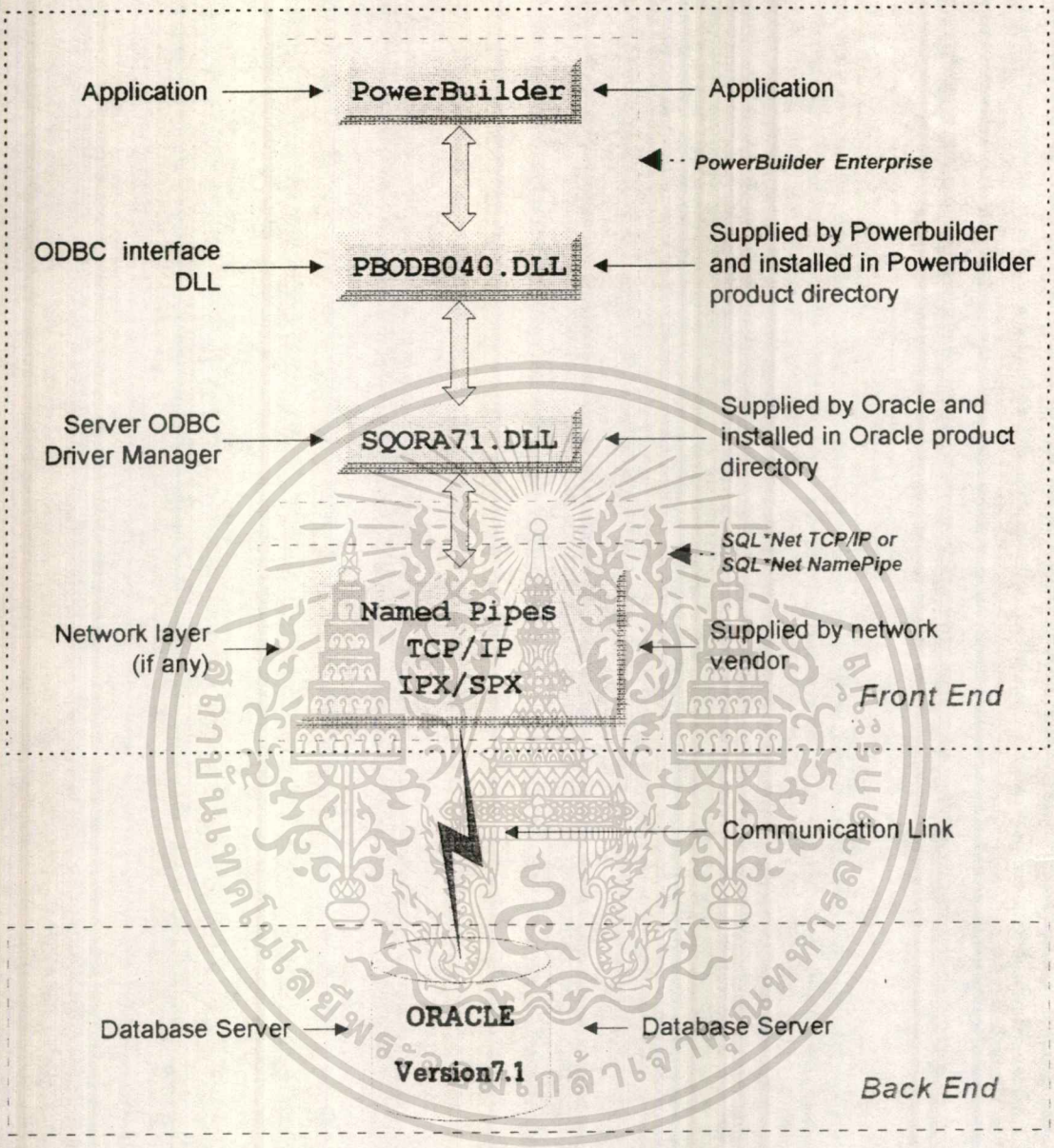
การเชื่อมต่อระหว่าง Front End (PB4) และ Back End (SQL-Server 4.21) โดยใช้ ODBC



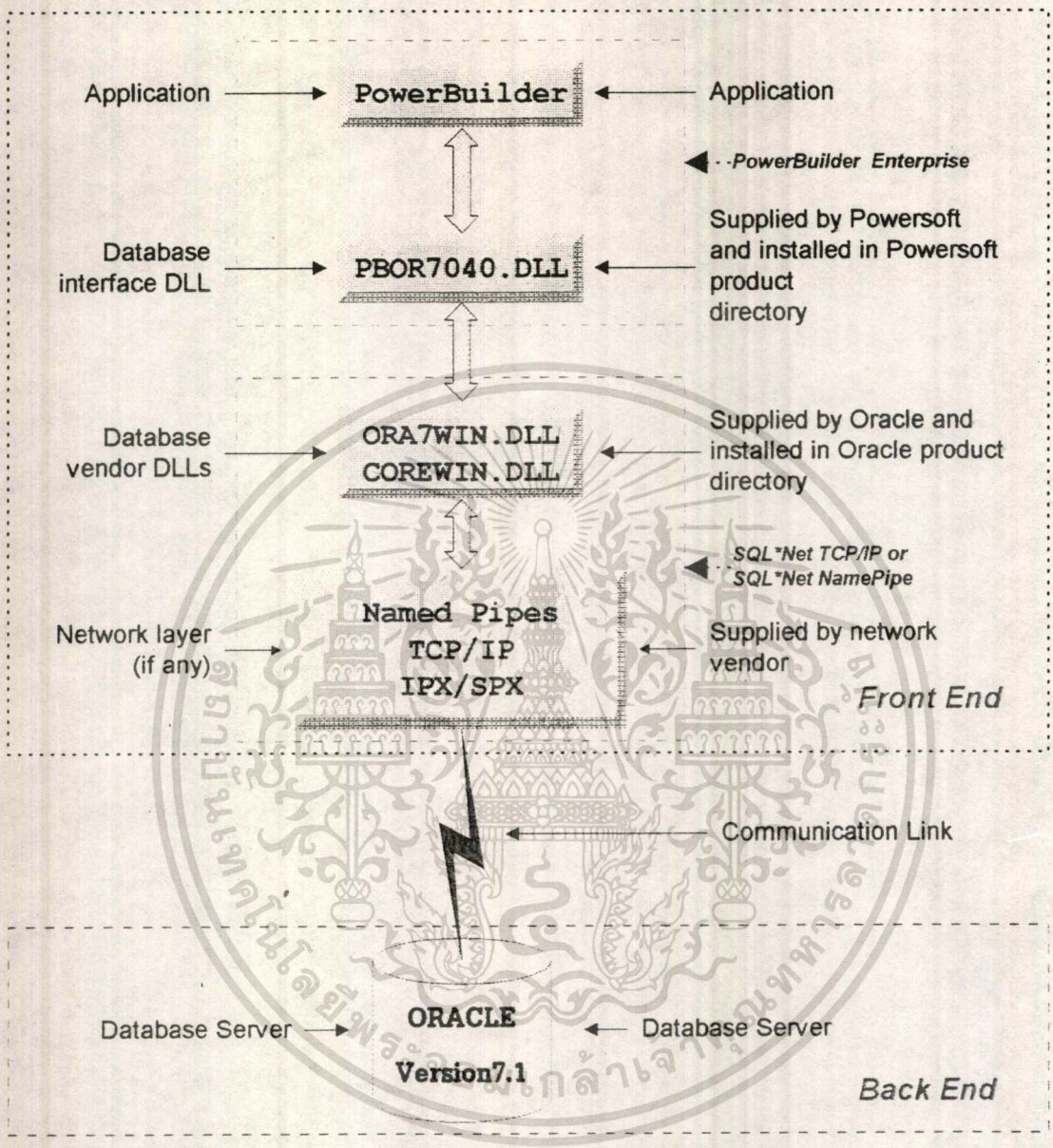
การเชื่อมต่อระหว่าง Front End (PB4) และ Back End (SQL-Server 4.21) โดยใช้ Native Driver



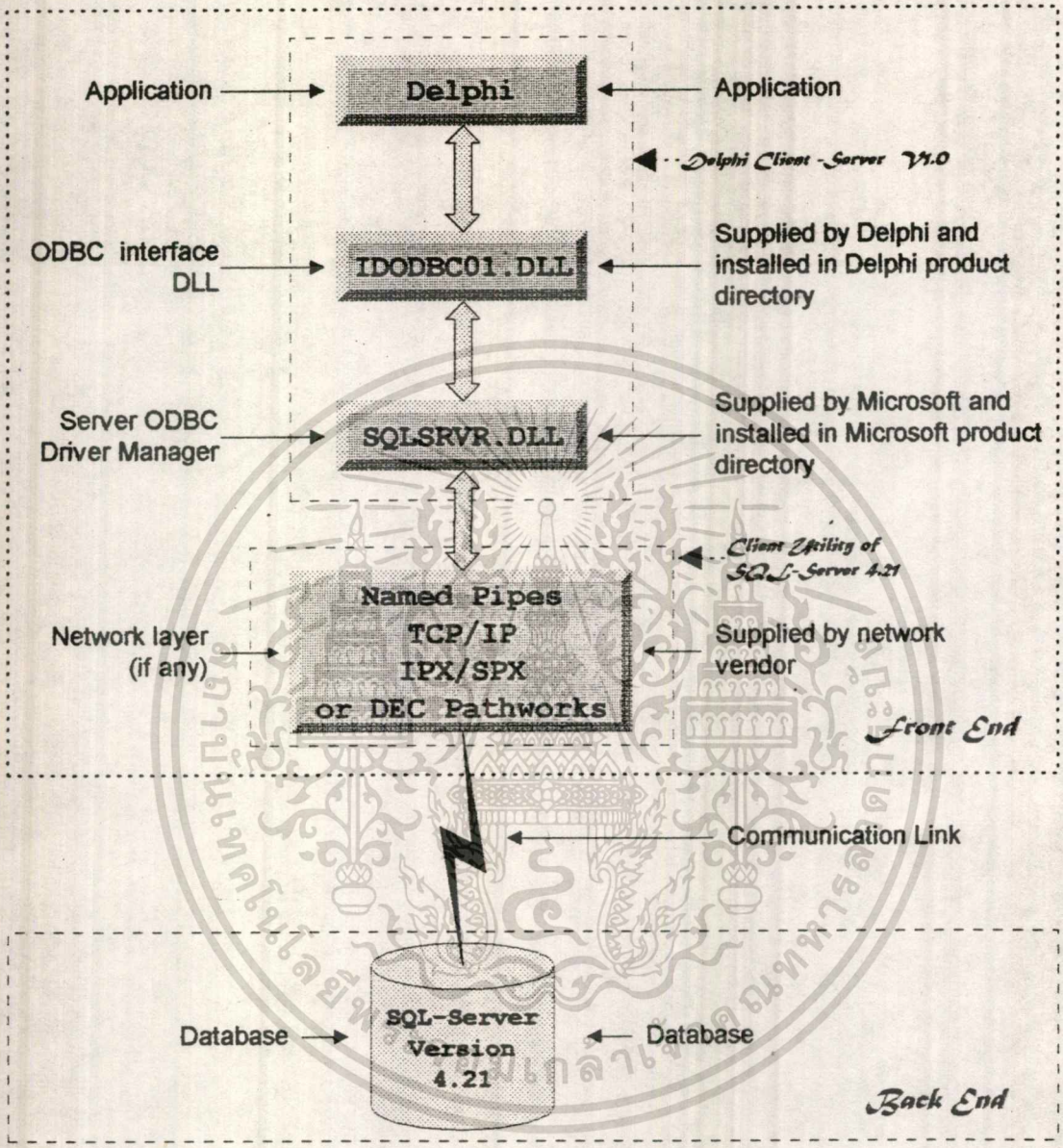
การเชื่อมต่อระหว่าง Front End (PB4) และ Back End (Oracle7.1) โดยใช้ ODBC



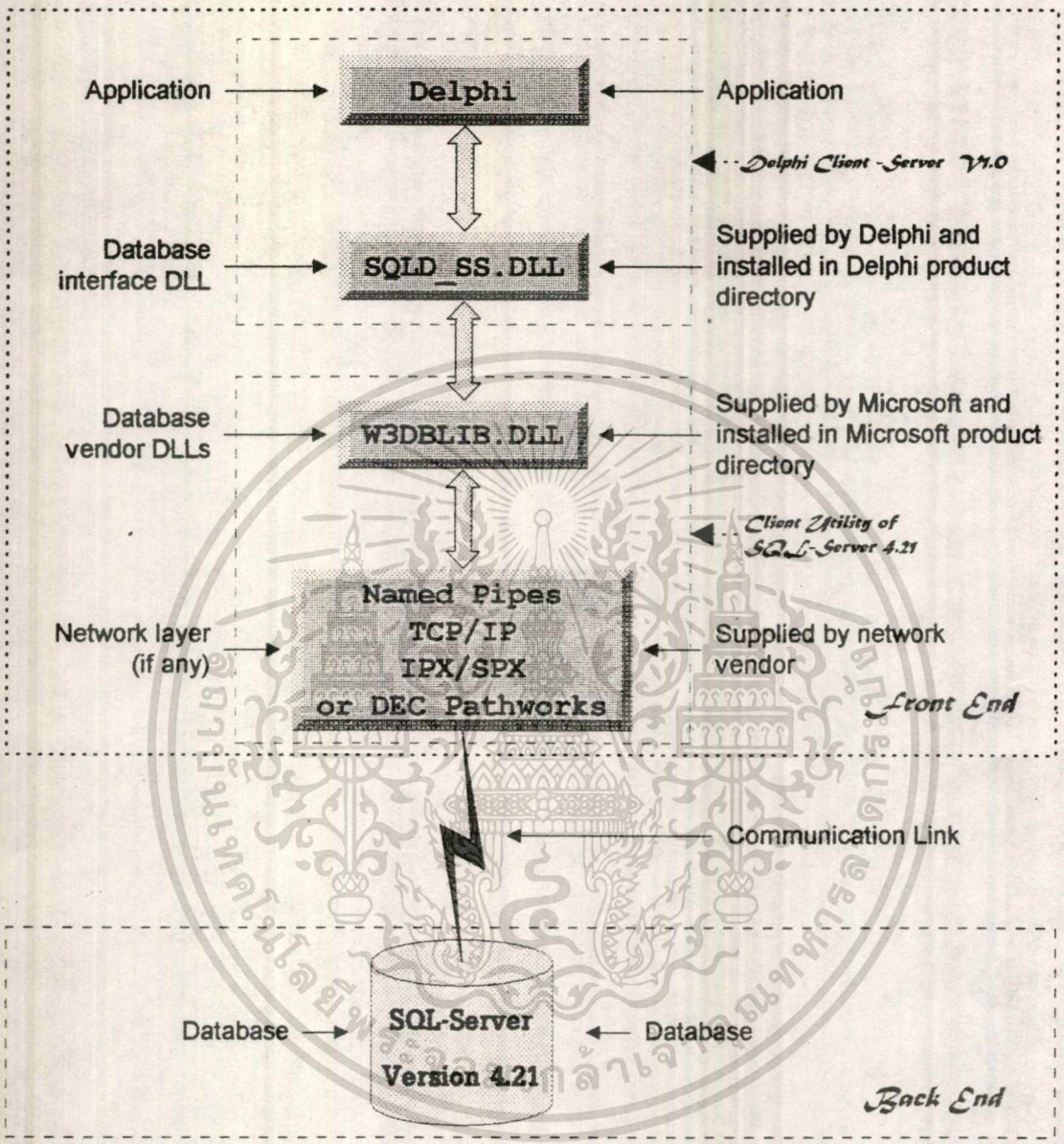
การเชื่อมต่อระหว่าง Front End (PB4) และ Back End (Oracle7.1) โดยใช้ Native Driver



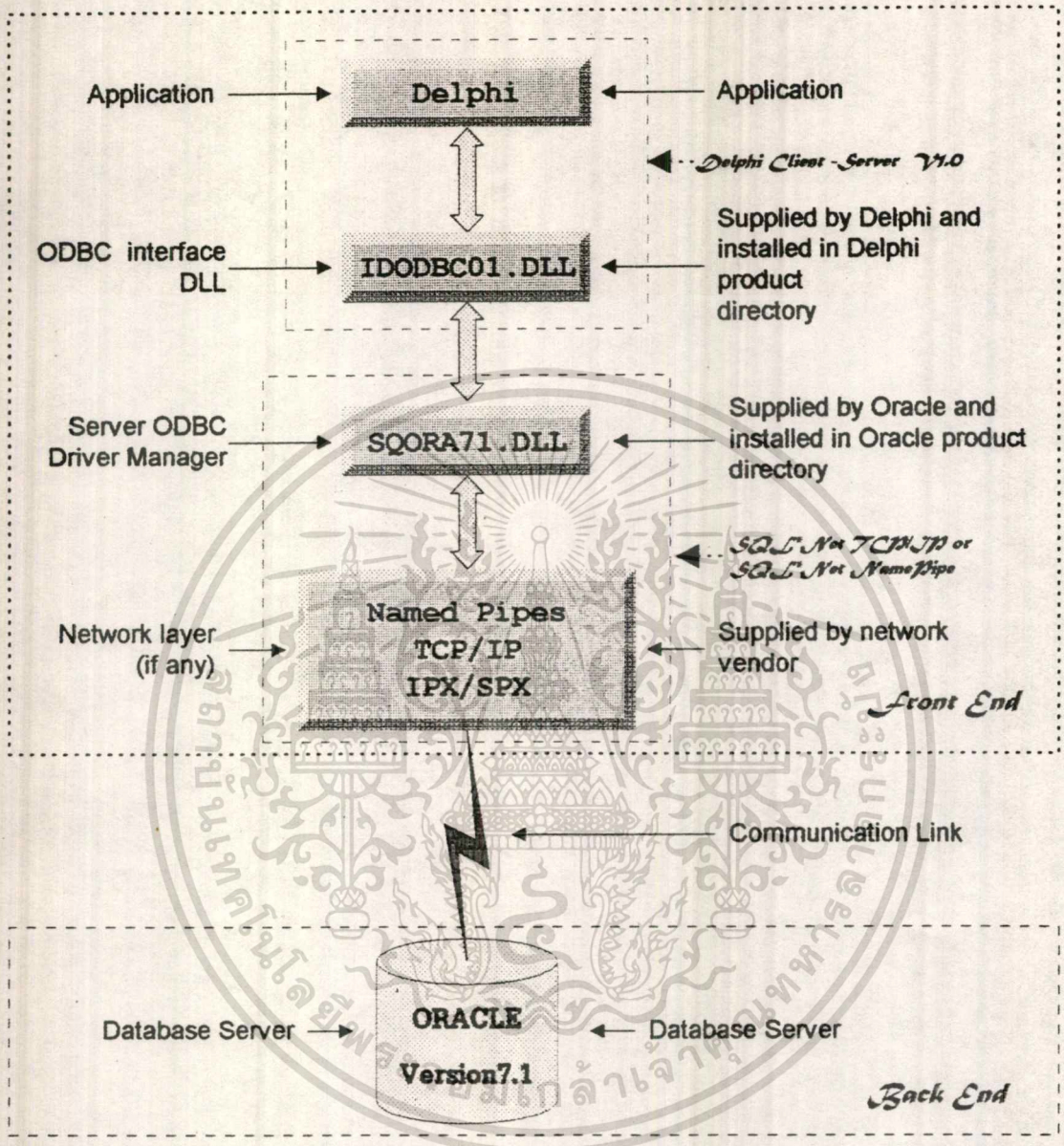
การเชื่อมต่อระหว่าง Font End (Delphi) และ Back End (SQL-Server 4.21) โดยใช้ ODBC



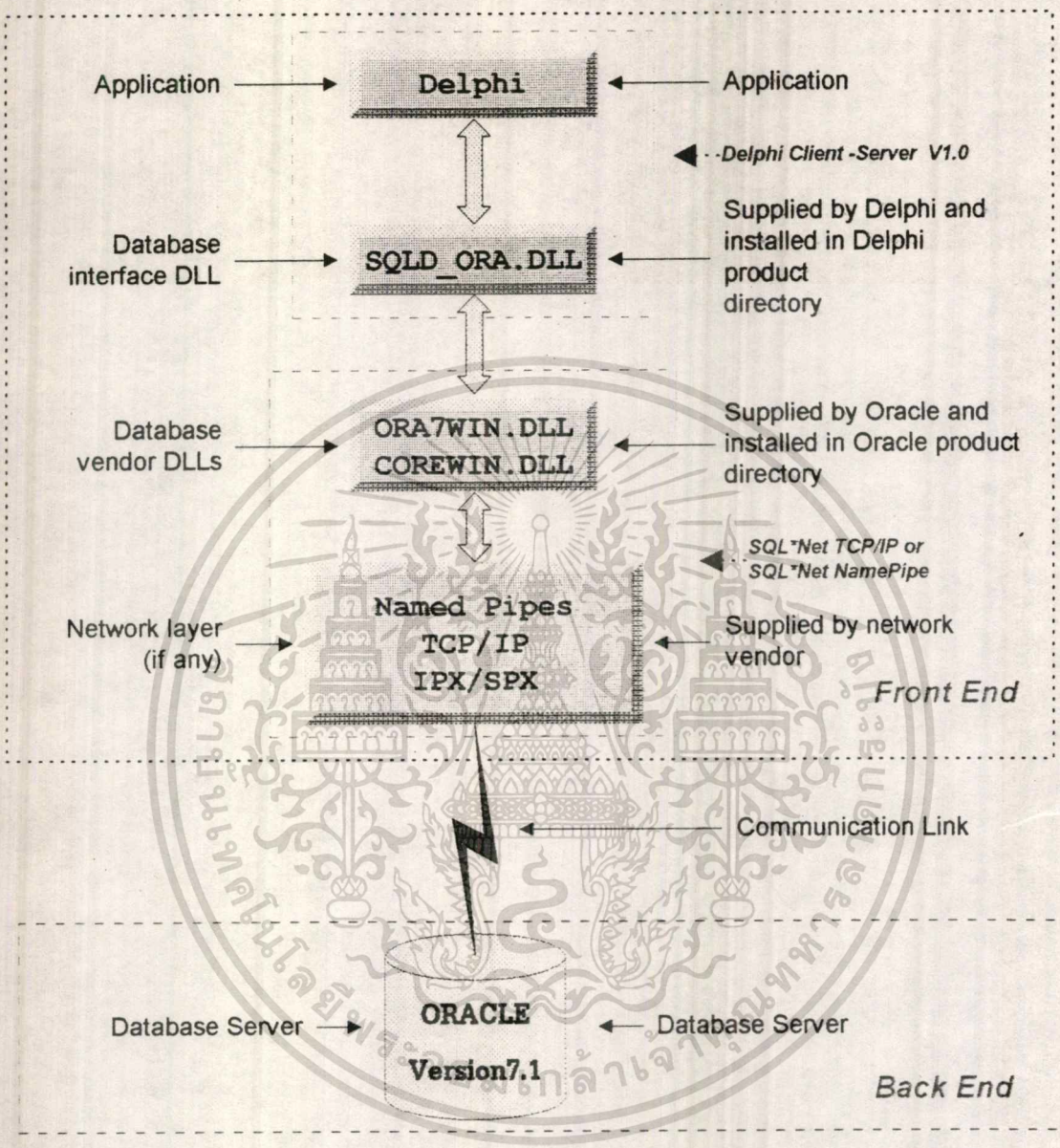
การเชื่อมต่อระหว่าง Front End (Delphi) และ Back End (SQL-Server 4.21) โดยใช้ Native Driver



การเชื่อมต่อระหว่าง Front End (Delphi) และ Back End (Oracle7.1) โดยใช้ ODBC



การเชื่อมต่อระหว่าง Front End (Delphi) และ Back End (Oracle7.1) โดยใช้ Native Driver



บทที่ 4
การเตรียมตารางและข้อมูลสำหรับการทดลอง

การเตรียมตารางเพื่อใช้ในการทดลองจะใช้ชุดไดล์เอ็นต์ของ เอสคิวแอล เซอร์ฟเวอร์ ดาต้า เบส เพื่อให้สะดวกแก่การสร้างตารางเนื่องจากมีการแสดงผลแบบกราฟฟิก โดยใน ออราเคิล จะใช้โปรแกรม ออปเจ็กต์ เมนเนเจอร์ (Object Manager) ส่วน ไมโครซอฟต์ เอสคิวแอล เซอร์ฟเวอร์ (Microsoft SQL Server) จะใช้โปรแกรม เอสคิวแอล ออปเจ็กต์เมนเนเจอร์ (SQLObject Manager)

แต่ในกรณีของออราเคิล จะไม่สามารถที่จะใช้ โปรแกรมออปเจ็กต์ เมนเนเจอร์ ในการสร้างตารางที่มี คลัสเตอร์ อินเด็กซ์ (Cluset Index) ดังนั้นจึงต้องใช้โปรแกรม เอสคิวแอล พลัส (SQL*PLUS) ในการสร้างตารางที่มีการทำคลัสเตอร์ อินเด็กซ์ โดยจะมีขั้นตอนการสร้างตาราง detail_100th_cluster และ ตาราง head_100th_cluster ดังนี้

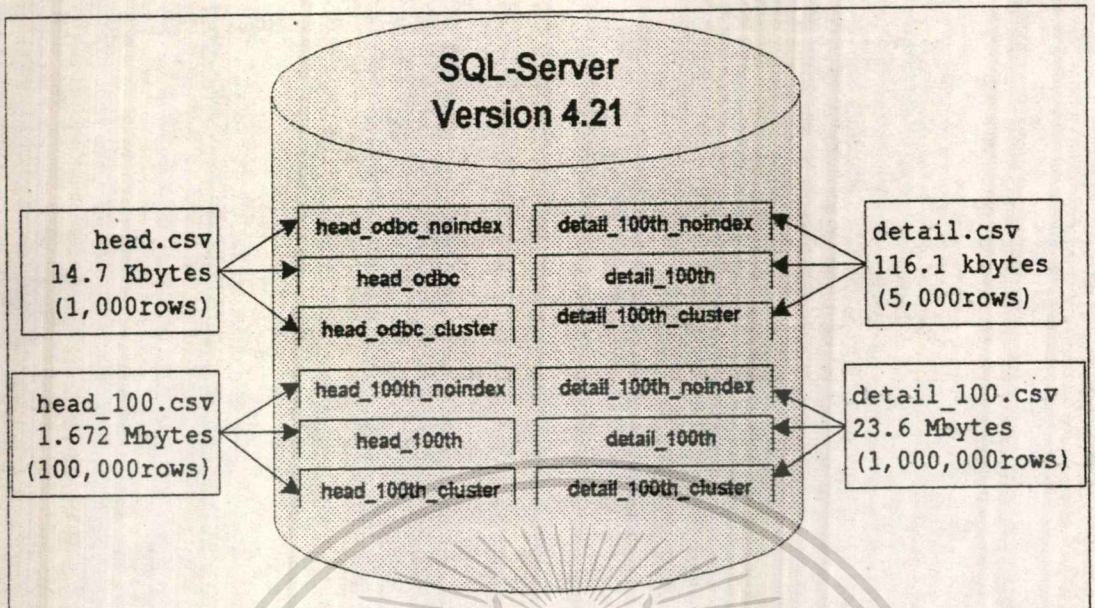
```
create cluster cluster_porder
(porder_number number(10))
size 512
storage(initial 100k next 50k pctincrease 10);

create table head_100th_cluster
(porder# number(10),
dates number(10)  not null,
sale# number(10) not null,
custom# number(10) not null,
station char(1) not null)
cluster cluster_porder(porder#);

create table detail_100th_cluster
(porder# number(10),
part# number(10),
quantity number(10) not null,
price number(10) not null)
cluster cluster_porder(porder#);

create index idx_porder_100th_cluster on cluster cluster_porder
```

ส่วนข้อมูลที่จะใช้ในการทดลองจะทำการเลือกข้อมูลจากข้อมูลที่ได้มีการทำการแทรกข้อมูลไว้ก่อนหน้าแล้ว โดยที่ข้อมูลที่จะมาทำการแทรกจะได้มาจากการสุ่มขึ้นมา



SQL Server for NT 4.12			
ขนาดจำนวน 100,000 rows		ขนาดจำนวน 1,000,000 rows	
Head_100th_noindex	2088 Kbytes	Detail_100th_noindex	20004 Kbytes
Head_100th	3708 Kbytes	Detail_100th	32812 Kbytes
Head_100th_cluster	2114 Kbytes	Detail_100th_cluster	36286 Kbytes
ขนาดจำนวน 1,000 rows		ขนาดจำนวน 5,000 rows	
Head_odbc_noindex	22 Kbytes	Detail_odbc_noindex	100 Kbytes
Head_odbc	40 Kbytes	Detail_odbc	164 Kbytes
Head_odbc_cluster	24 Kbytes	Detail_odbc_cluster	154 Kbytes

ขนาดข้อมูลที่ทำการ Import (text file)	
ขนาดจำนวน 1,000 rows	14,736 byte
ขนาดจำนวน 5,000 rows	116,091 byte
ขนาดจำนวน 100,000 rows	1,672,699 byte
ขนาดจำนวน 1,000,000 rows	23,594,892 byte

บทที่ 5
การทดลองและผลการทดลอง

การศึกษาผลกระทบของจำนวนไคล์เอ็นต์ที่มีต่อระบบไคล์เอ็นต์เซอร์ฟเวอร์และการศึกษาถึงผลกระทบในการทำงานของแต่ละคำสั่งที่กระทำกับข้อมูลที่อยู่ในโครงสร้างในแต่ละแบบของตารางในฐานข้อมูลเชิงสัมพันธ์ โดยใช้แอพพลิเคชั่นที่เขียนโดยโปรแกรม เดลไฟ และ เพาเวอร์บิวเดอร์

โดยจะทำการทดลองกับเซอร์ฟเวอร์ที่มีความแตกต่างกันทั้งด้านซอฟท์แวร์และฮาร์ดแวร์ ดังนี้

1. Oracle 7 ทำงานบนระบบปฏิบัติการ UNIX บนเครื่อง SUN SPARC

<u>คุณสมบัติของเครื่องที่ใช้ทดสอบ</u>

ตัวเครื่อง : WorkStation Sun SPARC Server20 Model 61

- หน่วยความจำหลัก 64 Mbyte
- Harddisk 6 Gbyte SCSI-2
- Ethernet IEEE 802.3 10Base-T 2 ชุด
- 1.44 Mbyte 3.5 นิ้ว Floppy Disk Drive 1 ชุด
- 1 Parallel port และ 1 RS-232 Serial Port
- Console เป็น Text Mode จอสีเดียวขนาด 14 นิ้ว พร้อม Keyboard
- Unix Operating System

2. Oracle 7 ทำงานบนระบบปฏิบัติการ Windows NT 3.50

<u>คุณสมบัติของเครื่องที่ใช้ทดสอบ</u>

ตัวเครื่อง : PC Server CPU486DX4-100 Cache 256 Kbyte

- หน่วยความจำหลัก 32 Mbyte
- Harddisk 1.2 Gbyte IDE
- Ethernet IEEE 802.3 10Base-T 1 ชุด (32 bit)
- 1.44 Mbyte 3.5 นิ้ว Floppy Disk Drive 1 ชุด
- 1 Rarallel port และ 1 RS-232 Serial Port
- Console เป็น Graphic Mode จอสี ขนาด 14 นิ้ว พร้อม Keyboard
- Microsoft Windows NT Server Operating System Version 3.50

3. SQL Server 4.21 ทำงานบนระบบปฏิบัติการ Windows NT 3.50

<u>คุณสมบัติของเครื่องที่ใช้ทดสอบ</u>

ตัวเครื่อง : PC Server CPU486DX4-100 Cache 256 Kbyte

- หน่วยความจำหลัก 32 Mbyte
- Harddisk 1.2 Gbyte IDE
- Ethernet IEEE 802.3 10Base-T 1 ชุด (32 bit)
- 1.44 Mbyte 3.5 นิ้ว Floppy Disk Drive 1 ชุด
- 1 Parallel port และ 1 RS-232 Serial Port
- Console เป็น Graphic Mode จอสี ขนาด 14 นิ้ว พร้อม Keyboard
- Microsoft Windows NT Server Operating System Version 3.50

4. Client ทุกๆตัวทั้งหมด 10 ตัว

<u>คุณสมบัติของเครื่องที่ใช้ทดสอบ</u>

ตัวเครื่อง : PC Client CPU486DX2-50 Cache 256 Kbyte

- หน่วยความจำหลัก 8 Mbyte
- Harddisk 256 Mbyte IDE
- Ethernet IEEE 802.3 10Base-T 1 ชุด (16 bit)
- 1.44 Mbyte 3.5 นิ้ว Floppy Disk Drive 1 ชุด
- 1 Parallel port และ 1 RS-232 Serial Port
- Console เป็น Graphic Mode จอสี ขนาด 14 นิ้ว พร้อม Keyboard
- MS-DOS version 6.22 Thai Edition Operating System
- Microsoft Windows 3.11 Thai Edition

ส่วนทางด้านไคล์เอ็นต์จะใช้แอพพลิเคชันที่พัฒนาขึ้นมาโดยจะทำการติดต่อกับเซอร์ฟเวอร์และวัดเวลาที่เซอร์ฟเวอร์ตอบสนองกลับมายังไคล์เอ็นต์แต่ละตัว โดยจะใช้ไคล์เอ็นต์ติดต่อไปที่เซอร์ฟเวอร์เป็นจำนวนสูงสุด 10 ตัว และ ไคล์เอ็นต์แต่ละตัวจะทำคำสั่งในการ แทรกข้อมูล โลดอลอินเตอร์เบส เซอร์ฟเวอร์ (Insert) ข้อมูลเข้าไปในฐานข้อมูล และทำคำสั่งในการ เลือกข้อมูล(select) ข้อมูลที่อยู่ในฐานข้อมูลกลับมาที่ไคล์เอ็นต์ โดยที่ฐานข้อมูลแต่ละตัวจะมีการสร้างตารางเตรียมไว้ ซึ่งจะเป็นตารางที่มีโครงสร้างแตกต่างกันไปโดยจะทดสอบกับตารางที่มีโครงสร้างดังนี้

1. ตารางที่ไม่มีการสร้างอินเด็กซ์
2. ตารางที่มีการสร้างอินเด็กซ์
3. ตารางที่มีการสร้างคัสเตอร์อินเด็กซ์

หลังจากนั้นทำการตรวจสอบเวลาที่เซอร์ฟเวอร์แต่ละตัวตอบสนองกับไคล์เอ็นต์แต่ละจำนวน และในแต่ละคำสั่งที่กระทำต่อแต่ละโครงสร้างของตาราง

การทำคำสั่ง แทรกข้อมูล จะทำการแทรกข้อมูลที่ได้จากการสุ่มเข้าไปไว้ในตาราง Head_100th และตาราง Detail_100th โดยก่อนที่จะทำการแทรกข้อมูล ไคล์เอ็นต์แต่ละตัวจะต้องไปทำการอ่านค่าของข้อมูลจากตาราง Corder ที่เก็บค่า Corder# ก่อนเสมอ ซึ่งการเข้าไปอ่านข้อมูลในตาราง Corder ของแต่ละไคล์เอ็นต์ในแต่ละครั้งจะต้องทำการเปลี่ยนแปลง (Update) ข้อมูลที่อยู่ในตารางนั้นโดยจะทำการเพิ่มค่าที่อยู่ในตารางขึ้นไปอีกหนึ่ง เสมอ โดยหลังจากที่ได้ค่าของ Corder# มาแล้วไคล์เอ็นต์จะใช้ค่าที่ได้ในการแทรกข้อมูลนี้เป็นข้อมูล Porder# เข้าไปที่ตาราง Head_100th และตาราง Detail_100th ด้วยโดยมีข้อแม้ว่าค่าของ Porder# จะห้ามไม่ให้ซ้ำกัน ดังนั้นเมื่อไคล์เอ็นต์แต่ละตัวเข้าไปอ่านค่าของตาราง Corder จะต้องทำการล็อก (Lock) ตาราง Corder เพื่อไม่ให้ไคล์เอ็นต์ตัวอย่างนำค่าปัจจุบันไปใช้ซ้ำโดยจะปล่อยล็อกเมื่อได้มีการเปลี่ยนแปลงค่า Corder# ให้เป็นค่าใหม่แล้ว ซึ่งจากเงื่อนไขนี้จะทำให้ไคล์เอ็นต์แต่ละตัวจะเกิดการล็อกกันที่ตาราง Corder เสมอ ดังนั้นเมื่อมีการเพิ่มจำนวนของไคล์เอ็นต์ที่ติดต่อไปยังเซอร์ฟเวอร์พร้อมกันหลายๆไคล์เอ็นต์ จะทำให้การไคล์เอ็นต์แต่ละตัวรับการตอบสนองจากเซอร์ฟเวอร์ด้วยระยะเวลาที่นานขึ้น โดยคำสั่งแทรกข้อมูลที่จะใช้ คือ

```
insert into     head_100th
values          (:porder# , :dates , :sale# , :custom# , :station)
insert into     detail_100th
values          (:porder#,:part#,:quantity,:price)
```

โดย :porder# , :dates , :sale# , :custom# , :station เป็นค่าที่ได้จากการสุ่มจาก ไคล์เอ็นต์ :part# , :quantity , :station เป็นค่าที่ได้จากการสุ่มจากไคล์เอ็นต์

การทำคำสั่ง เลือกข้อมูล จะต้องทำป้อนค่าข้อมูลที่ต้องการเลือกเพื่อให้การเลือกข้อมูลจากเซอร์ฟเวอร์ เป็นไปด้วยค่าเดียวกันในโครงสร้างทั้งสามของตาราง โดยคำสั่งเลือกข้อมูลที่จะใช้ คือ

```
select      p1.sale# , p2.quantity
from        head_100th p1 , detail_100th p2
where       p1.dates = :dates  and  p2.part# > :lower
            and  p1.part# < :upper  and  p1.porder# = p2.porder#
```

โดย :dates , :lower , :upper เป็นค่าที่ป้อนเข้าไปที่ไคล์เอ็นต์ แม้ว่าการทำประโยคคำสั่งที่ผ่านมากับเครื่องแต่ละเซอร์ฟเวอร์จะเหมือนกันในส่วนใหญ่แต่ในรายละเอียดปลีกย่อยจะมีความแตกต่างกันในแต่ละเซอร์ฟเวอร์ คือ

ในออราเคิลจะทำการเปลี่ยนข้อมูลในตาราง Corder ก่อนที่จะทำการ เลือกข้อมูล เพื่อให้เกิดการล็อกตาราง Corder เอาไว้จนกว่าไคล์เอ็นต์ที่ทำการ เลือกข้อมูลนั้นจะนำข้อมูลไปใช้แล้วจึงจะทำการปล่อยล็อกโดยการทำคำสั่ง commit ดังนั้นในเซอร์ฟเวอร์ออราเคิล จะมีลำดับการทำคำสั่งต่างๆ ดังนี้

```
update corder
select  corder
commit
insert into head_100th
commit
insert into detail_100th
commit
```

ใน เอสคิวแอล เซอร์ฟเวอร์ จะทำคำสั่ง คอมมิท ให้โดยอัตโนมัติเมื่อทำแต่ละคำสั่งเสร็จสิ้นลง ซึ่งจะสามารถควบคุมการทำ รายการ (Tranaction) โดยใช้ คำสั่ง Begin transaction และจบการทำรายการโดยใช้คำสั่ง Commit Transaction ดังนั้นในเซอร์ฟเวอร์ เอสคิวแอล เซอร์ฟเวอร์ จะมีลำดับการทำคำสั่งต่างๆ ดังนี้

```
begin transaction
update corder
select corder
commit transaction
insert into head_100th
insert into detail_100th
```

ส่วนหนึ่งของโปรแกรมที่เขียนด้วยเดลไฟที่ใช้ในการแทรกข้อมูลเข้าในเซอร์ฟเวอร์ ออราเคิล

```
with query1 do
begin
 h_date := random(30)+1;
 h_sale := random(10)+1;
 h_custom := random(100)+1;
 d_part1 := random(40001)+1;
 d_part2 := random(60001)+1;
 d_part := d_part1 + d_part2;
 d_quantity := random(10000)+1;
 d_price := random(60000)+1;
 f_Ttime := gettickcount;
 close;
 sql.clear;
 sql.add('update corder set corder# = corder# + 1');
 execsql;
 close;
```

```
sql.clear;
sql.add('select * from corder');
open;
query1.first;
porder := query1.fields[0].asinteger;
close;
sql.clear;
sql.add('commit');
execsql;   { retrive number from corder# complete }
close;
sql.clear;
sql.add(format(statement_h,[table1]));
params[0].asinteger := porder;
params[1].asinteger := h_date;
params[2].asinteger := h_sale;
params[3].asinteger := h_custom;
params[4].asstring :=  station;
execsql;
close;
sql.clear;
sql.add('commit');
execsql;
close;
sql.clear;
sql.add(format(statement_d,[table2]));
params[0].asinteger := porder;
params[1].asinteger := d_part;
params[2].asinteger := d_quantity;
params[3].asinteger := d_price;
execsql;
close;
sql.clear;
sql.add('commit');
execsql;
l_Ttime := gettickcount;
```

```
d_price := random(60000)+1;
close;
sql.clear;
sql.add(format(statement_h,[table1]));
params[0].asinteger := porder;
params[1].asinteger := h_date;
params[2].asinteger := h_sale;
params[3].asinteger := h_custom;
params[4].asstring :=  station;
execsql;
close;
sql.clear;
sql.add(format(statement_d,[table2]));
params[0].asinteger := porder;
params[1].asinteger := d_part;
params[2].asinteger := d_quantity;
params[3].asinteger := d_price;
execsql;
l_Ttime := gettickcount;
if ((i mod 5) + 1) = 1 then
  writeln(f_handle,timetostr(time));
writeln(f_handle,l_Ttime-f_Ttime);
end;
```

ส่วนหนึ่งของโปรแกรมที่เขียนด้วยเพาเวอร์บิวเดอร์ที่ใช้ในการแทรกข้อมูลเข้าในเซอร์ฟเวอร์

```
w_main.st_34.text = string(counter)
r_date      = rand(30)+1
r_sale              = rand(10)+1
r_custom    = rand(100)+1
r_quantity  = (rand(100) * rand(100))+1
r_price     = (rand(600) * rand(100))+1
r_part              = (rand(100) * rand(100) * rand(100))+1
time_start  = cpu()

UPDATE corder SET corder# = corder# + 1 ;
```

```
SELECT corder.corder# INTO :r_porder# FROM corder  ;
commit;
INSERT INTO head_odbc_noindex ( porder#, dates, sale#, custom#, station) VALUES (
                :r_porder#, :r_date,:r_sale, :r_custom, :char_ ) ;
INSERT INTO detail_odbc_noindex ( porder#, part#, quantity, price ) VALUES (
                 :r_porder#, :r_part,:r_quantity, :r_price ) ;
commit;
time_end = cpu() - time_start
end_time = string(counter)+',' + string(time_end)+','+string(Now(),"mm:ss")
filewrite(nfile1,end_time)
counter += 1
```

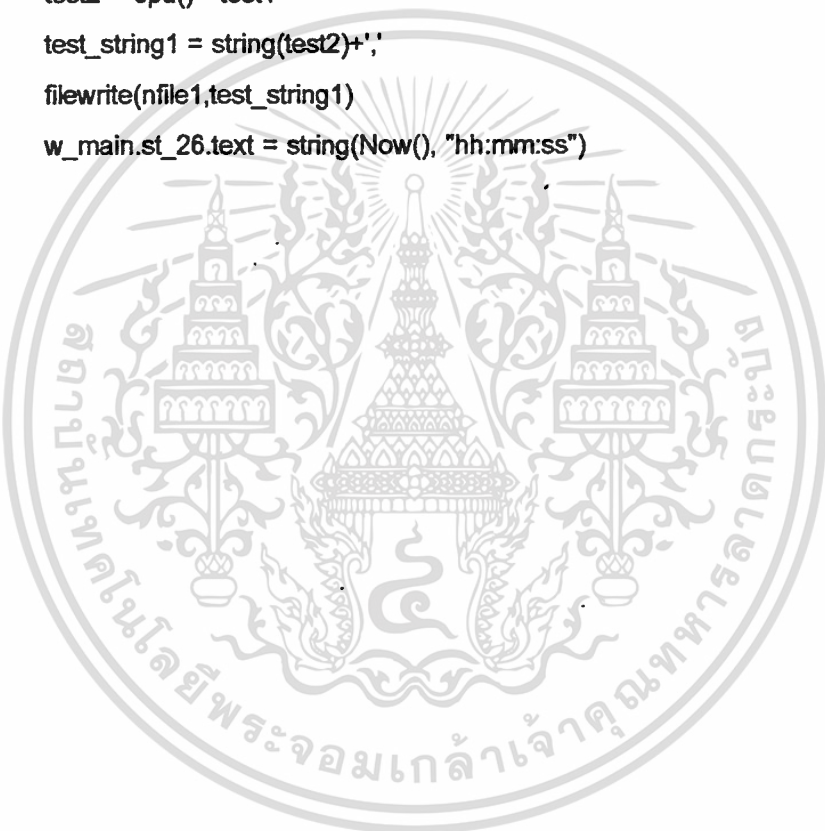
ส่วนหนึ่งของโปรแกรมที่เขียนด้วยเดลไฟที่ใช้ในการเลือกข้อมูลจากเซอร์ฟเวอร์

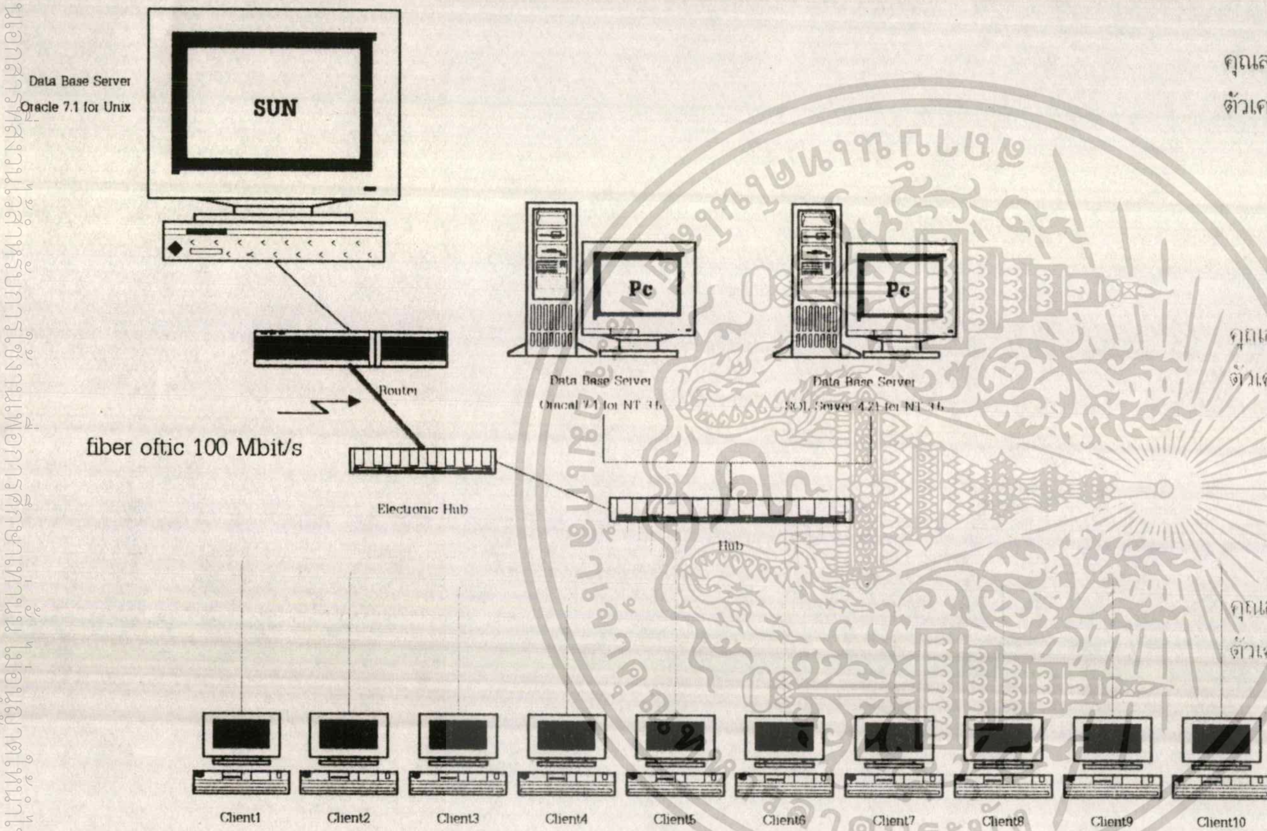
```
with query1 do
begin
  ran_date := arr_date[i];
  ran_Lamount := arr_part[i];
  ran_Hamount := ran_Lamount + 50000;
  DBGrid1.DataSource := DataSource1;
  f_Ttime := gettickcount;
  close;
  sql.clear;
  sql.add('select p1.sale#,p2.quantity');
  sql.add(format(from_statement,[edit8.text,edit9.text]));
  sql.add('where p1.dates = :date and p2.part# > :lower and p2.part# < :upper');
  sql.add('and  p1.porder# = p2.porder#');
  params[0].asinteger := ran_date;
  params[1].asinteger := ran_Lamount;
  params[2].asinteger := ran_Hamount;
  open;
  l_Ttime := gettickcount;
  writeln(f_handle,l_Ttime-f_Ttime);
end; {end with query1 do}
```

ส่วนหนึ่งของโปรแกรมที่เขียนด้วยเพาเวอร์บิวเดอร์ที่ใช้ในการเลือกข้อมูลจากเซอร์ฟเวอร์

```
time_it = Now()
LOOP Until (time_now = time_it)or(time_now < time_it)
        w_main.sle_1.text = type_show2 + string("NO.CSV")
soutfile1          = string(w_main.sle_1.text)
nfile1    = fileopen(soutfile1,linemode!,write!,lockwrite!,Append!)
test_string1 = 'Time Select = '
filewrite(nfile1,test_string1)
FOR cout  =1 TO 1
        couter_num = couter_num + 1
        CHOOSE CASE couter_num
        CASE 1
                choose_date = string(w_main.sle_14.text)
                choose_part#  = string(w_main.sle_7.text)
        CASE 2
                choose_date = string(w_main.sle_13.text)
                choose_part#  = string(w_main.sle_6.text)
        CASE 3
                choose_date = string(w_main.sle_10.text)
                choose_part#  = string(w_main.sle_5.text)
        CASE 4
                choose_date = string(w_main.sle_9.text)
                choose_part#  = string(w_main.sle_4.text)
        CASE 5
                choose_date = string(w_main.sle_8.text)
                choose_part#  = string(w_main.sle_3.text)
        END CHOOSE
        b = Long(w_main.sle_11.text)
        c = long(choose_part#)
        a = b + c
        choose_part = string(a)
        time_start = cpu()
        w_main.st_22.text = string(Now(), "hh:mm:ss")
        test1= cpu()
        NewSyn  = 'SELECT p1.sale#,p2.quantity' &
```

```
                +' FROM head_odbc_noindex p1,detail_odbc_noindex p2'
                &  +' WHERE ((p1.porder# = p2.porder# ) and (p1.dates = '
                + choose_date +')' &  +' and (p2.part# <' + choose_part +')
                and ( p2.part# >'+ choose_part#  +'))'
dw_main.Settransobject (sqlca)
dw_main.SetSQLSelect(NewSyn)
dw_main.Retrieve()
dw_main.setfocus()
test2 = cpu() - test1
test_string1 = string(test2)+','
filewrite(nfile1,test_string1)
w_main.st_26.text = string(Now(), "hh:mm:ss")
```





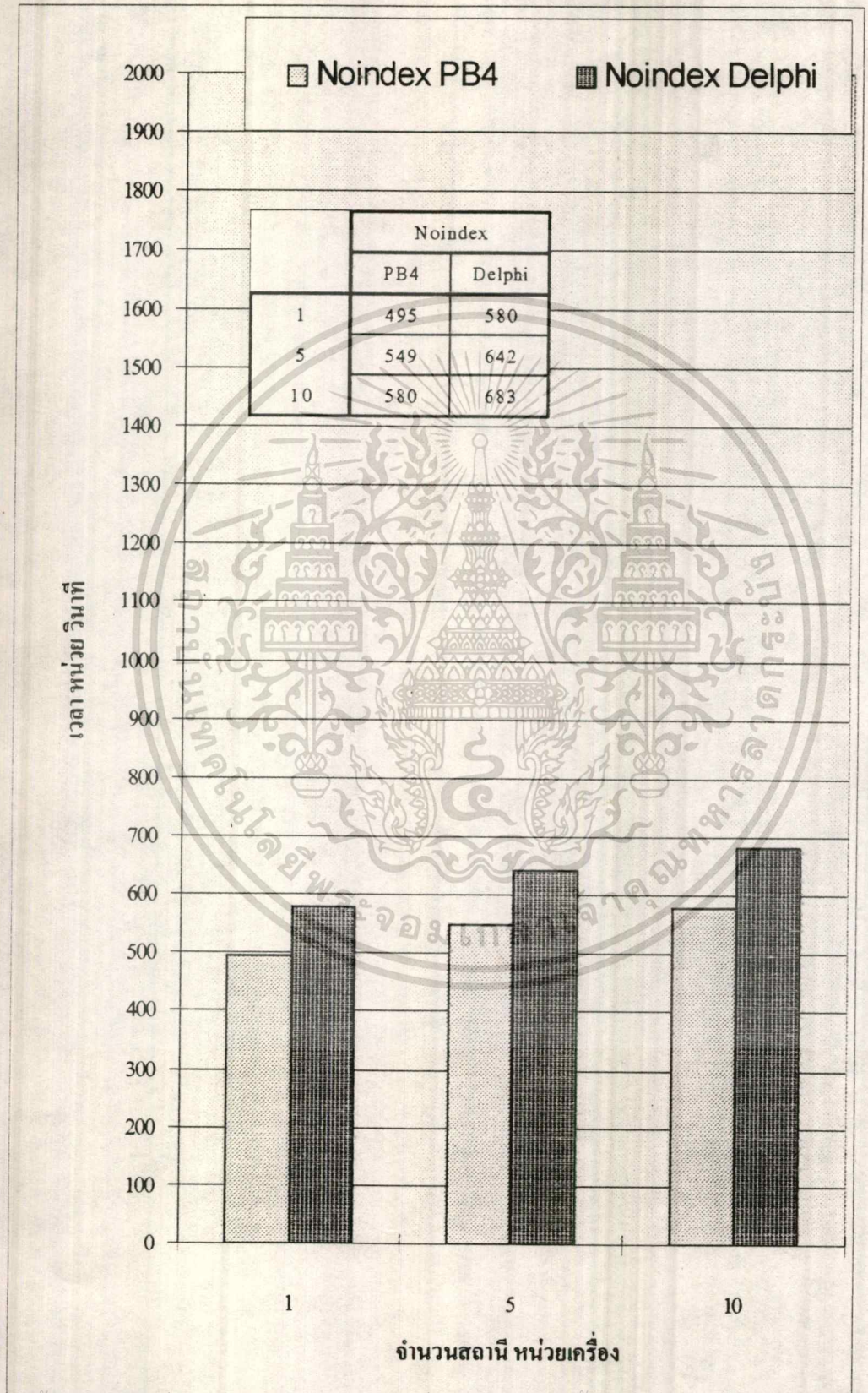
คุณสมบัติของเครื่องที่ใช้ทดสอบ
ตัวเครื่อง: WorkStaiton SUN SPARC Server20 Model61
- หน่วยความจำหลัก 64Mbyte
- Harddisk 6GByte SCSI-2
- Ethernet IEEE802.3 10Base-T 2ชุด
- Unix Operating System

คุณสมบัติของเครื่องที่ใช้ทดสอบ
ตัวเครื่อง: PC Server CPU486DX4-100 Cache 256Kbyte
- หน่วยความจำหลัก 32Mbyte
- Harddisk 1.2GByte IDE
- Ethernet IEEE802.3 10Base-T 1ชุด (32Bit)
- NT Server Operating System Version 3.5

คุณสมบัติของเครื่องที่ใช้ทดสอบ
ตัวเครื่อง: PC Client CPU486DX2-50 Cache 256Kbyte
- หน่วยความจำหลัก 8Mbyte
- Harddisk 256MByte IDE
- Ethernet IEEE802.3 10Base-T 1ชุด (16Bit)
- Windows95 ThaiEdition Operating System

ผลการทดลอง Select ข้อมูลขนาด 5,000 rows โดยใช้ Front End คือ Pb4,Delphi

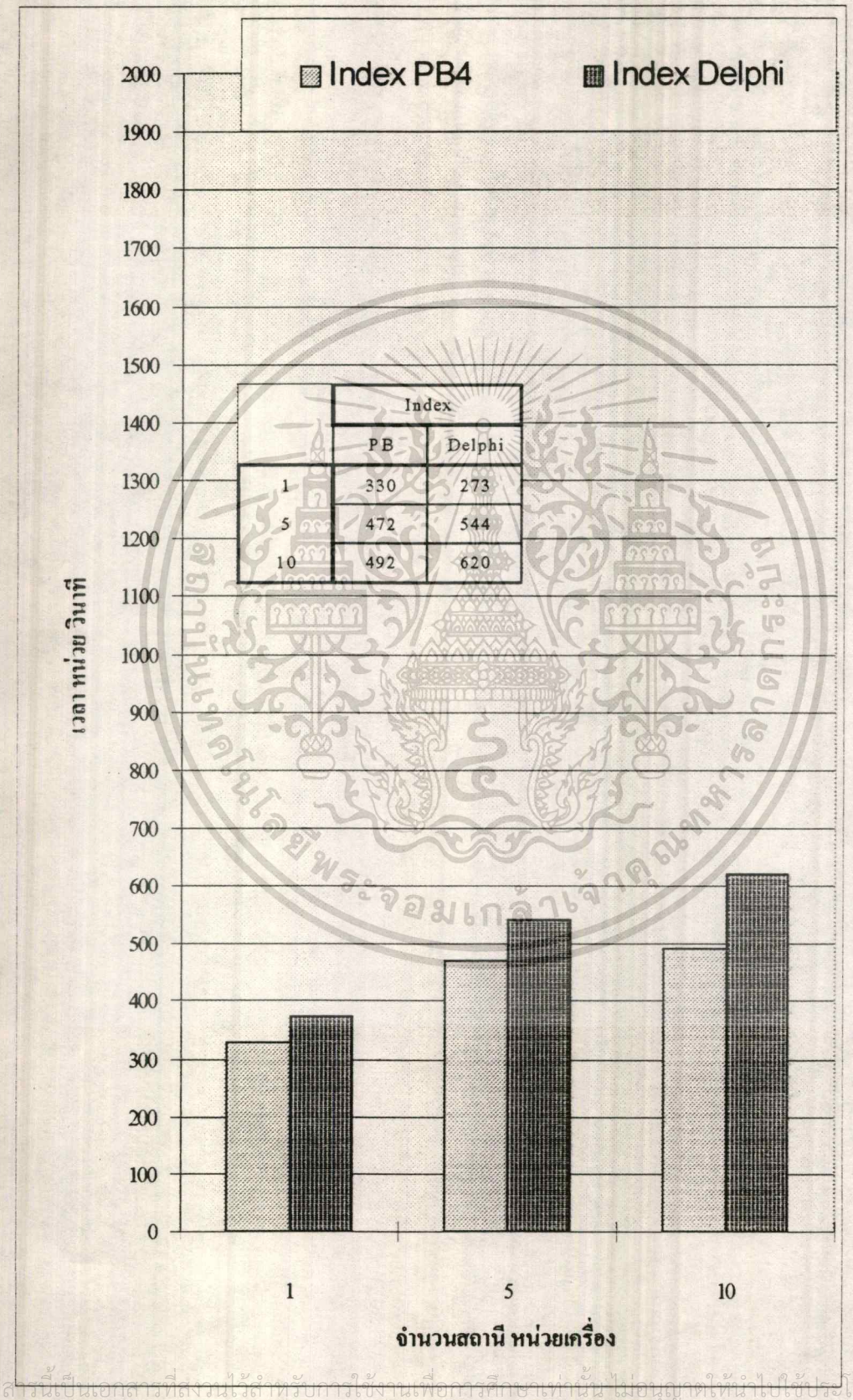
Back End คือ SQL-Server 4.21 ทำการติดต่อแบบ ODBC



	Noindex	
	PB4	Delphi
1	495	580
5	549	642
10	580	683

ผลการทดลอง Select ข้อมูลขนาด 5,000 rows โดยใช้ Front End คือ Pb4,Delphi

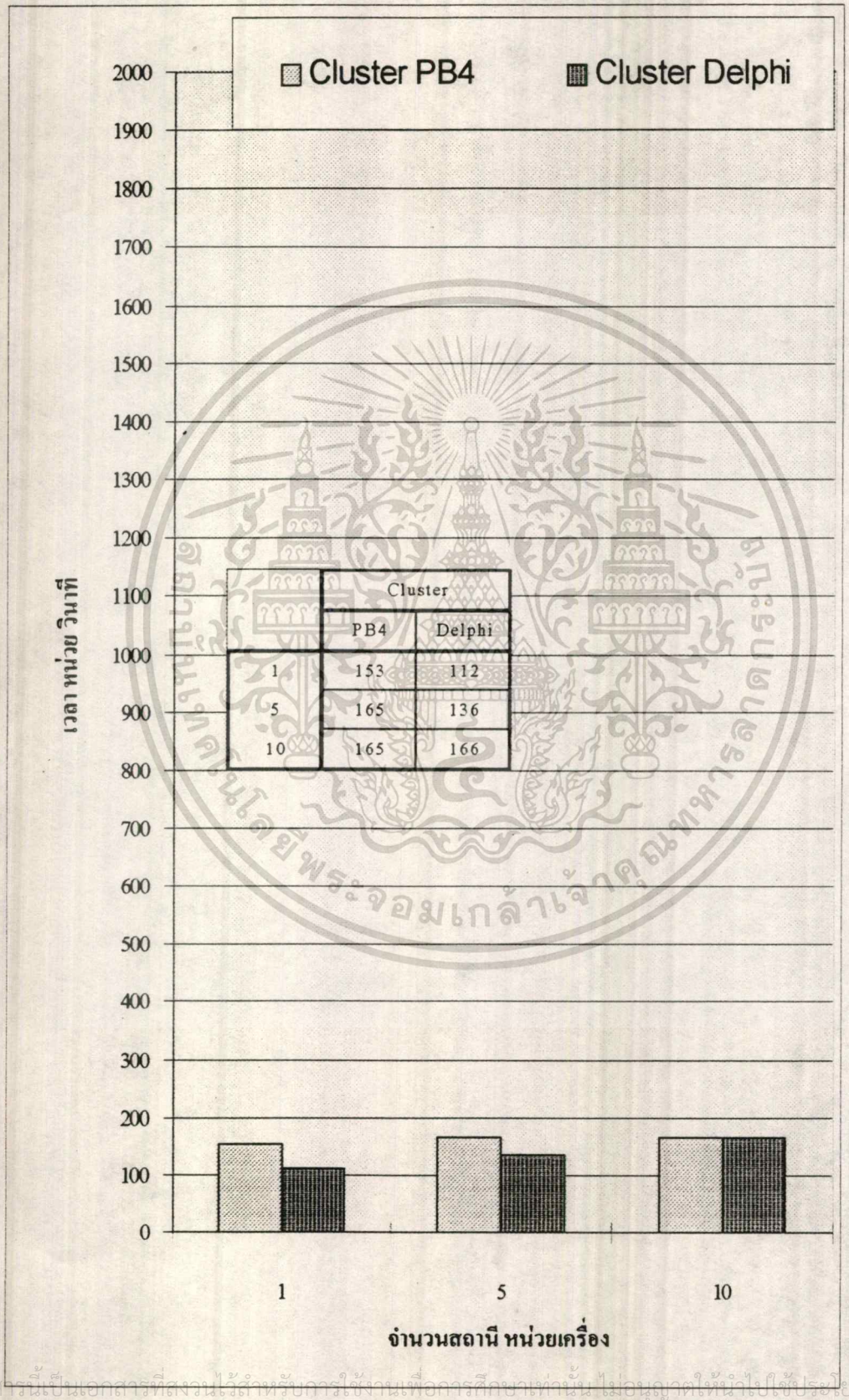
Back End คือ SQL-Server 4.21 ทำการติดต่อแบบ ODBC



	Index	
	PB	Delphi
1	330	273
5	472	544
10	492	620

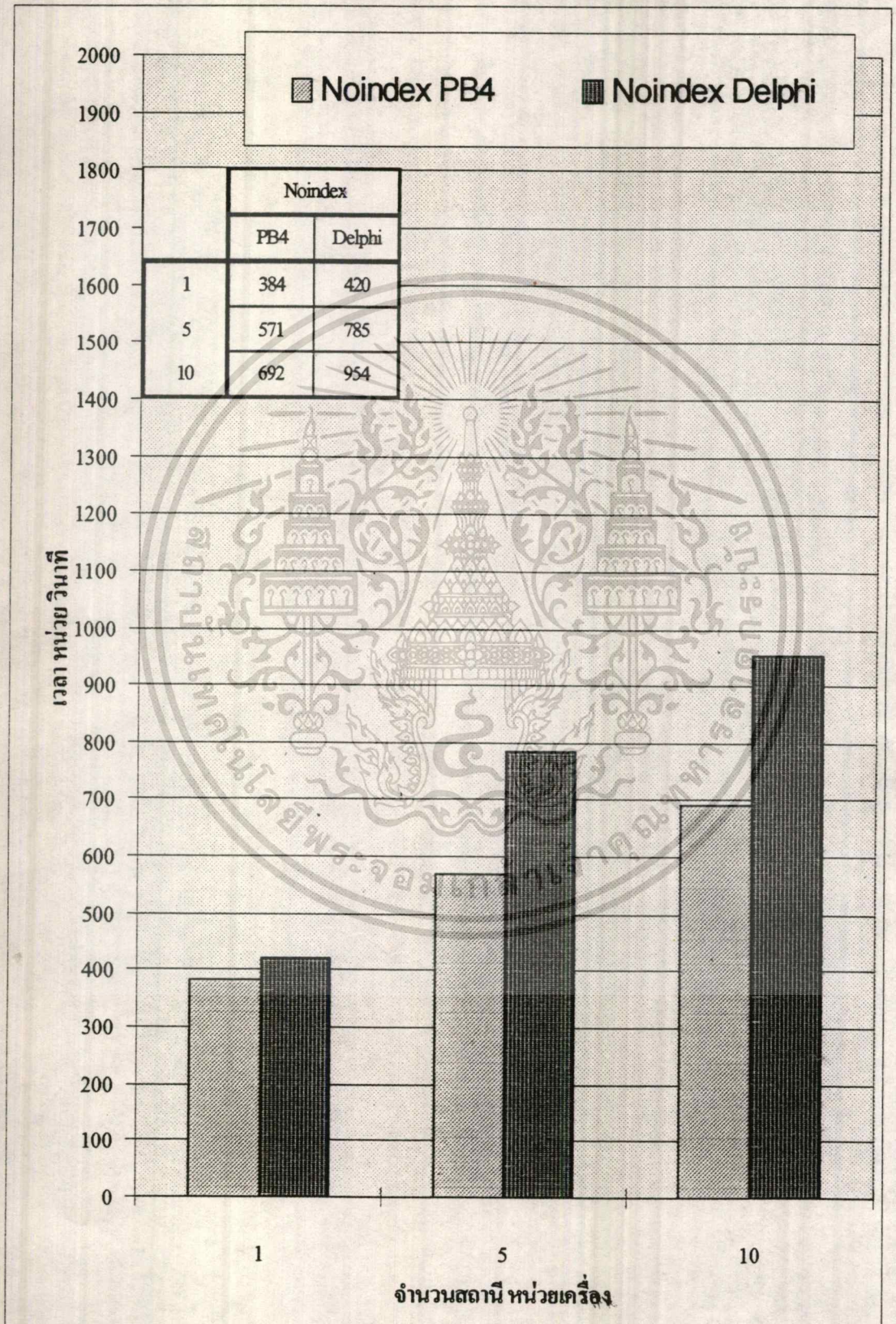
ผลการทดลอง Select ข้อมูลขนาด 5,000 rows โดยใช้ Front End คือ Pb4,Delphi

Back End คือ SQL-Server 4.21 ทำการติดต่อแบบ ODBC



ผลการทดลอง Select ข้อมูลขนาด 5,000 rows โดยใช้ Front End คือ Pb4,Delphi

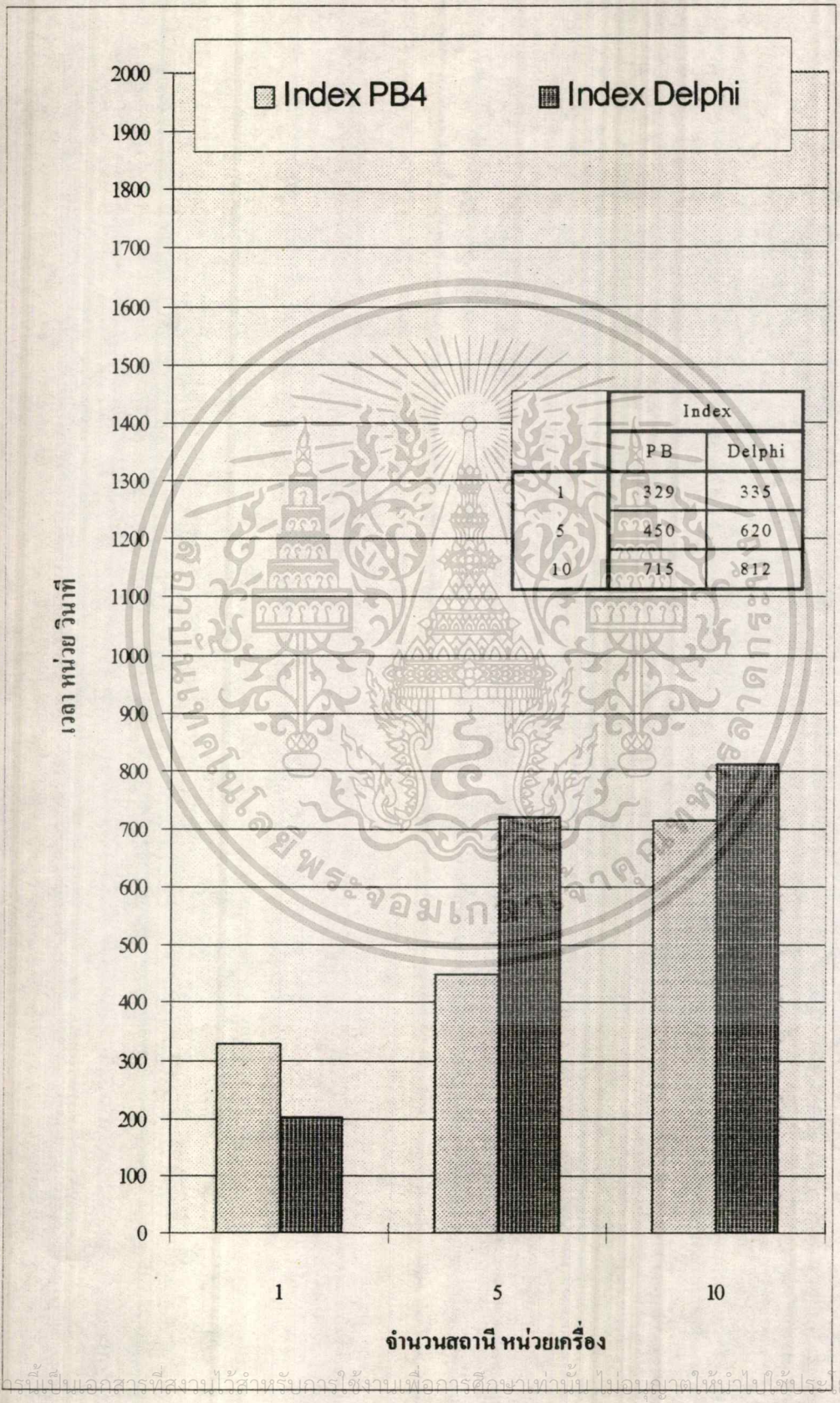
Back End คือ SQL-Server 4.21 for NT ทำการติดต่อแบบ Native Driver



	Noindex	
	PB4	Delphi
1	384	420
5	571	785
10	692	954

ผลการทดลอง Select ข้อมูลขนาด 5,000 rows โดยใช้ Front End คือ Pb4,Delphi

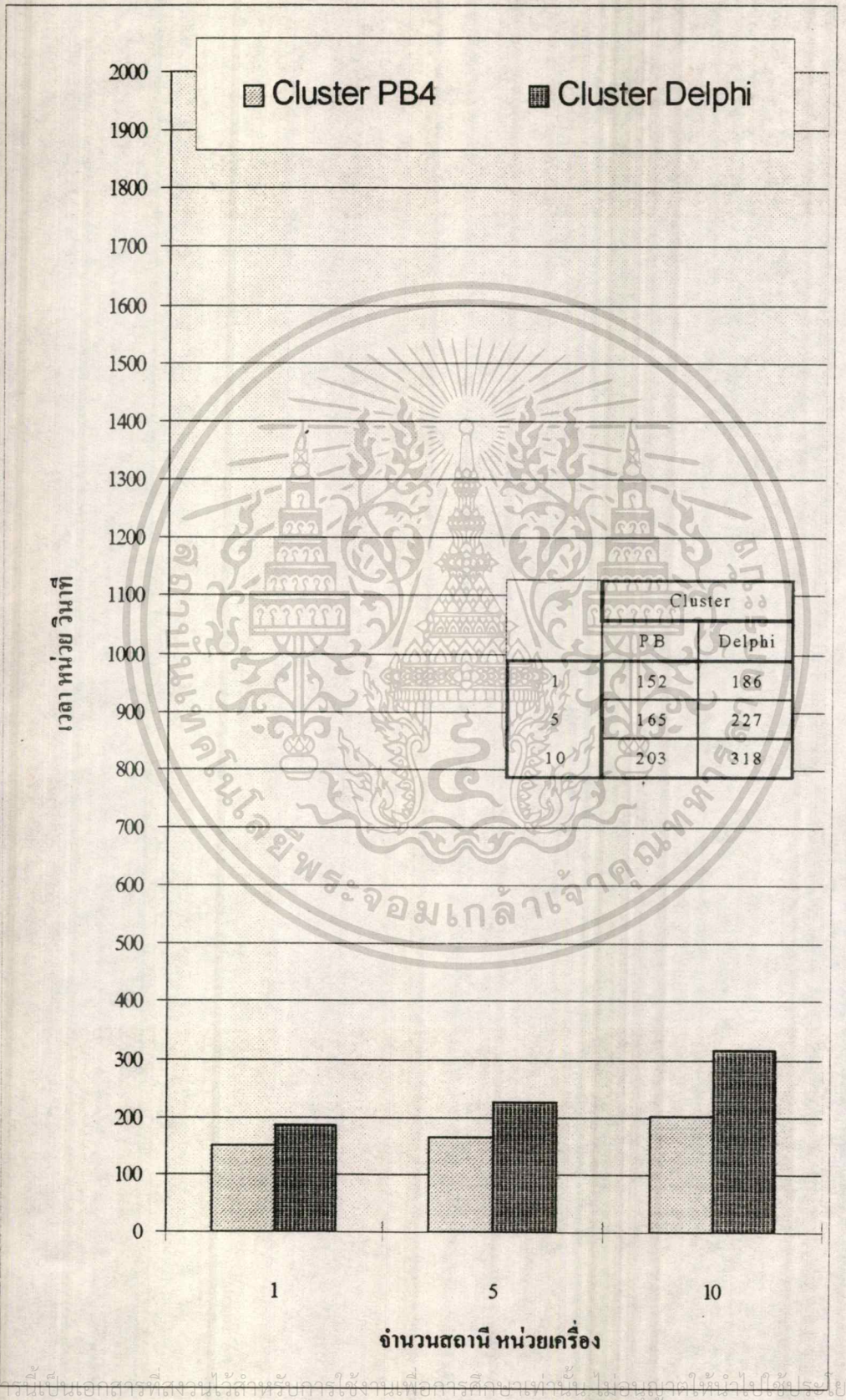
Back End คือ SQL-Server 4.21 ทำการติดต่อแบบ Native Driver



	Index	
	PB	Delphi
1	329	335
5	450	620
10	715	812

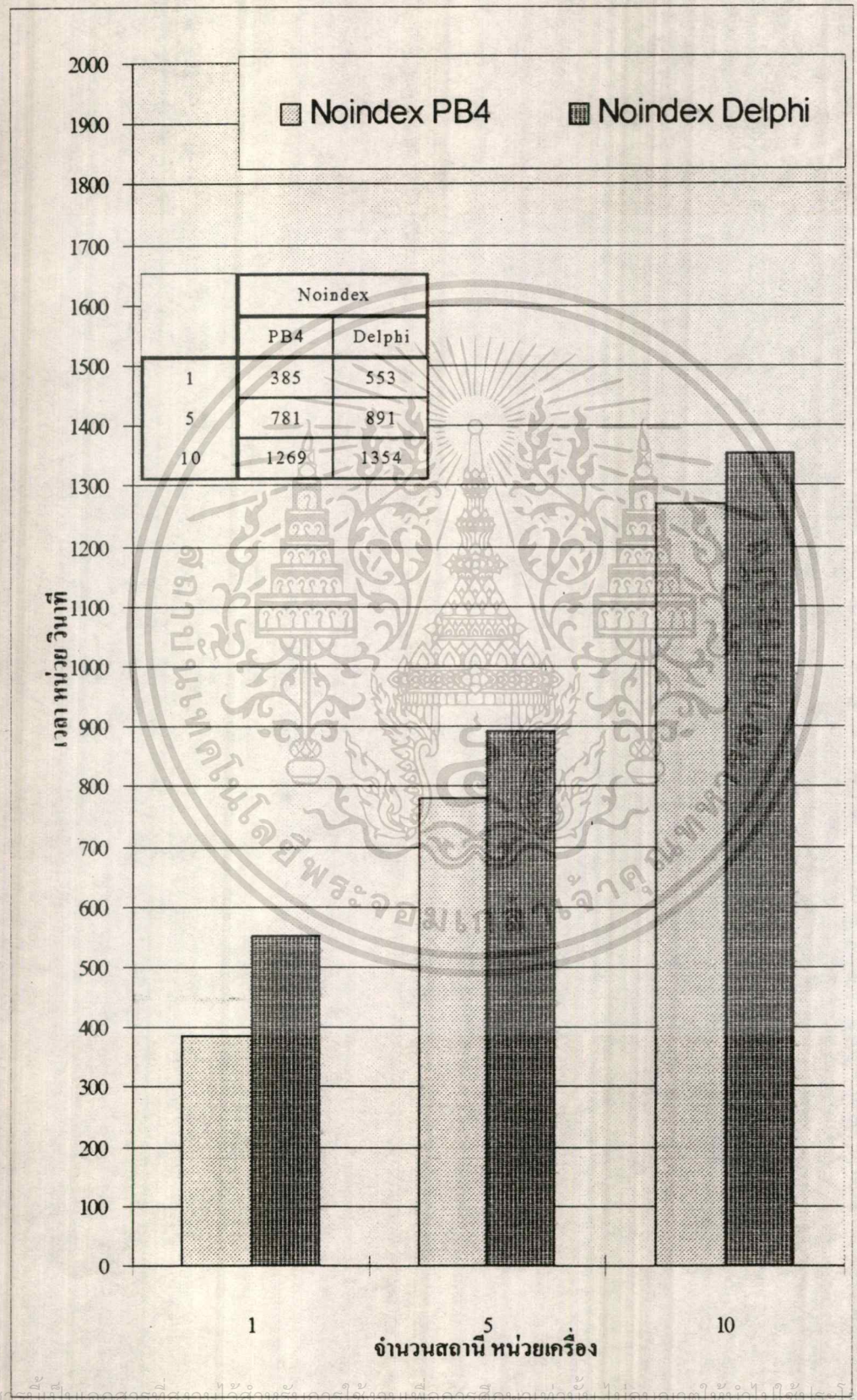
ผลการทดลอง Select ข้อมูลขนาด 5,000 rows โดยใช้ Front End คือ Pb4,Delphi

Back End คือ SQL-Server 4.21 ทำการติดต่อแบบ Native Driver



ผลการทดลอง Insert ข้อมูลขนาด 5,000 rows โดยใช้ Front End คือ Pb4,Delphi

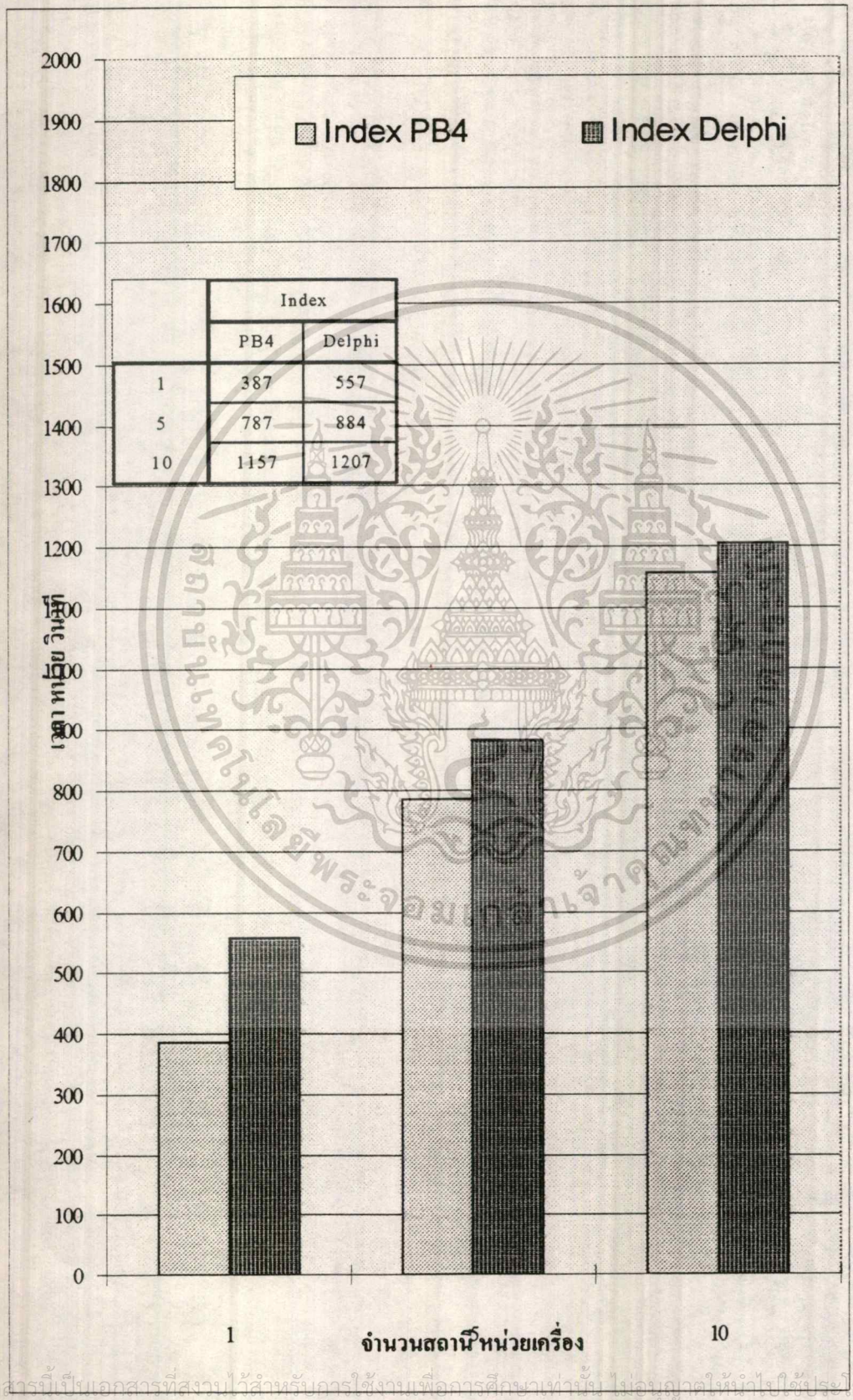
Back End คือ Oracle 7.1 for NT ทำการติดต่อแบบ ODBC



	Noindex	
	PB4	Delphi
1	385	553
5	781	891
10	1269	1354

ผลการทดลอง Insert ข้อมูลขนาด 5,000 rows โดยใช้ Front End คือ Pb4,Delphi

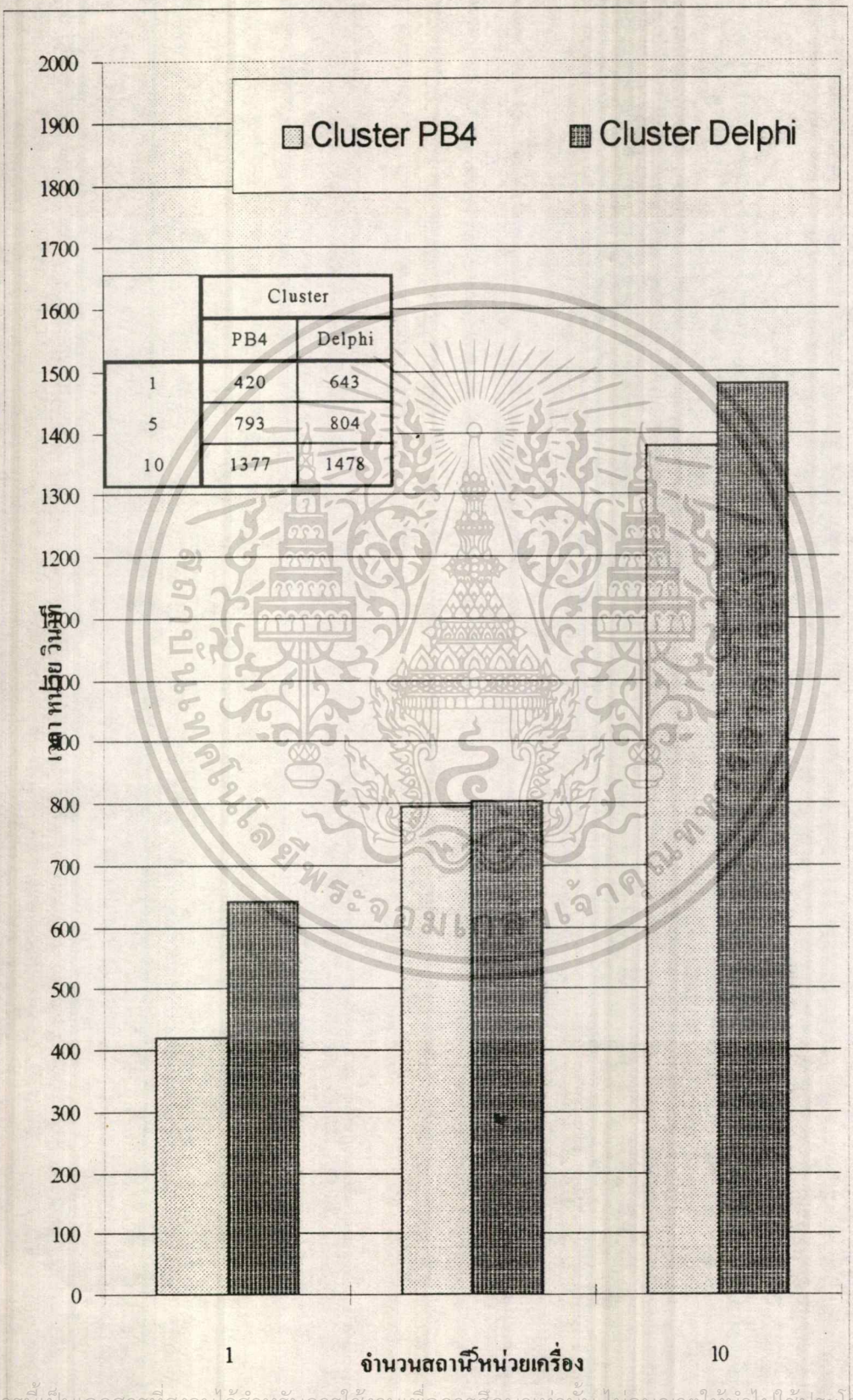
Back End คือ Oracle 7.1 for NT ทำการติดต่อแบบ ODBC



	Index	
	PB4	Delphi
1	387	557
5	787	884
10	1157	1207

ผลการทดลอง Insert ข้อมูลขนาด 5,000 rows โดยใช้ Front End คือ Pb4,Delphi

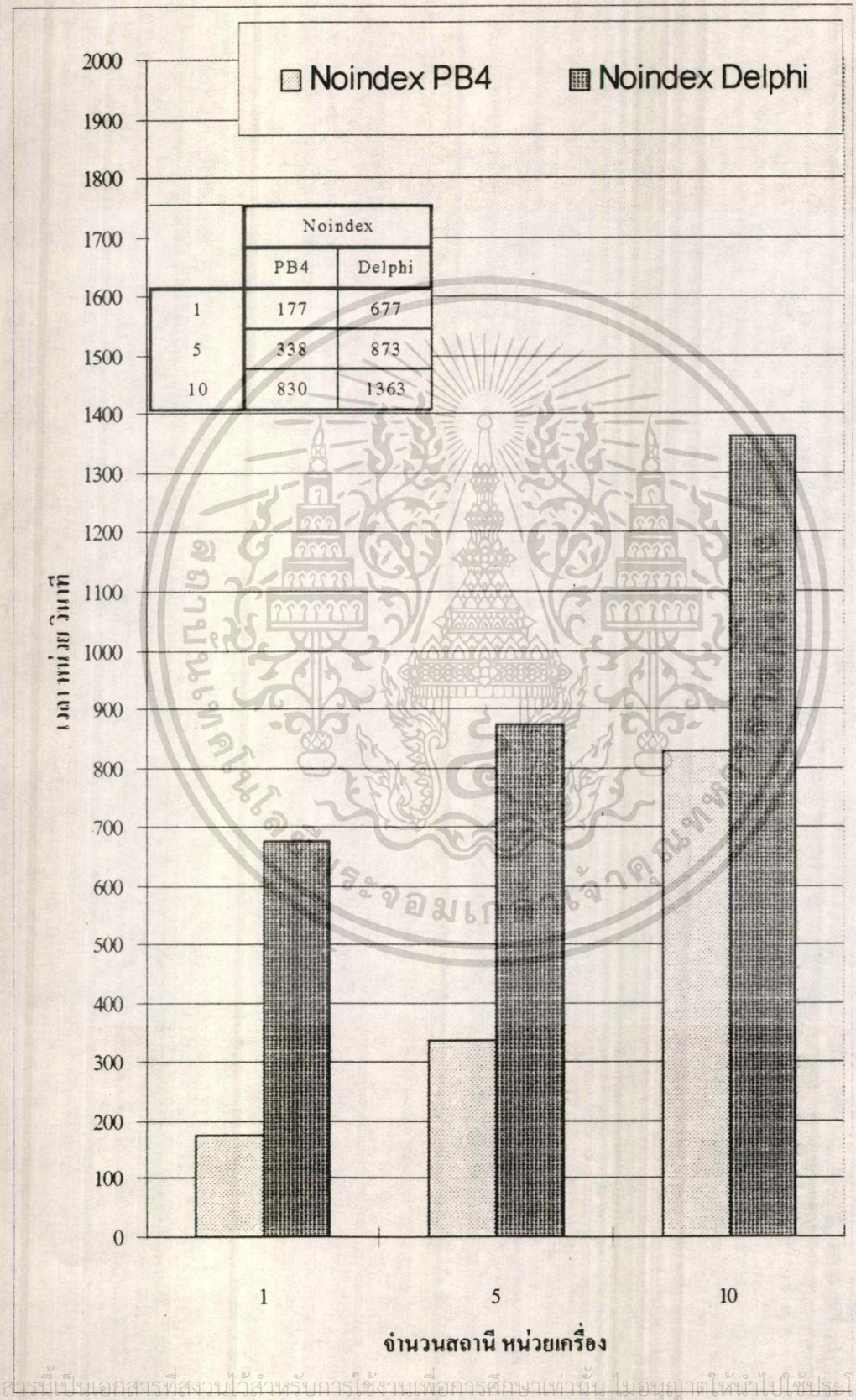
Back End คือ Oracle 7.1 for NT ทำการติดต่อแบบ ODBC



	Cluster	
	PB4	Delphi
1	420	643
5	793	804
10	1377	1478

ผลการทดลอง Insert ข้อมูลขนาด 5,000 rows โดยใช้ Front End คือ Pb4,Delphi

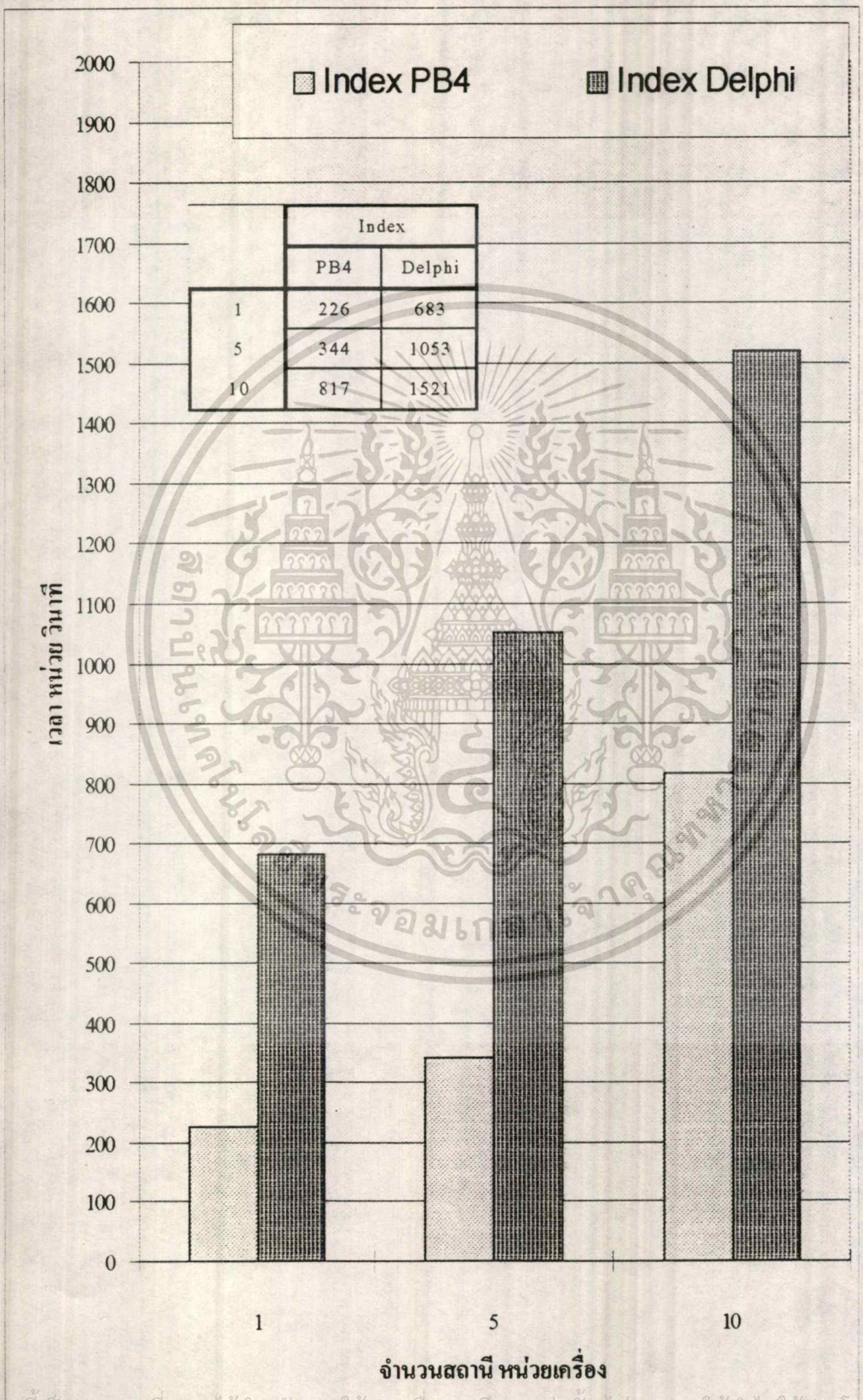
Back End คือ Oracle 7.1 for NT ทำการติดต่อแบบ Native Driver



	Noindex	
	PB4	Delphi
1	177	677
5	338	873
10	830	1363

ผลการทดลอง Insert ข้อมูลขนาด 5,000 rows โดยใช้ Front End คือ Pb4,Delphi

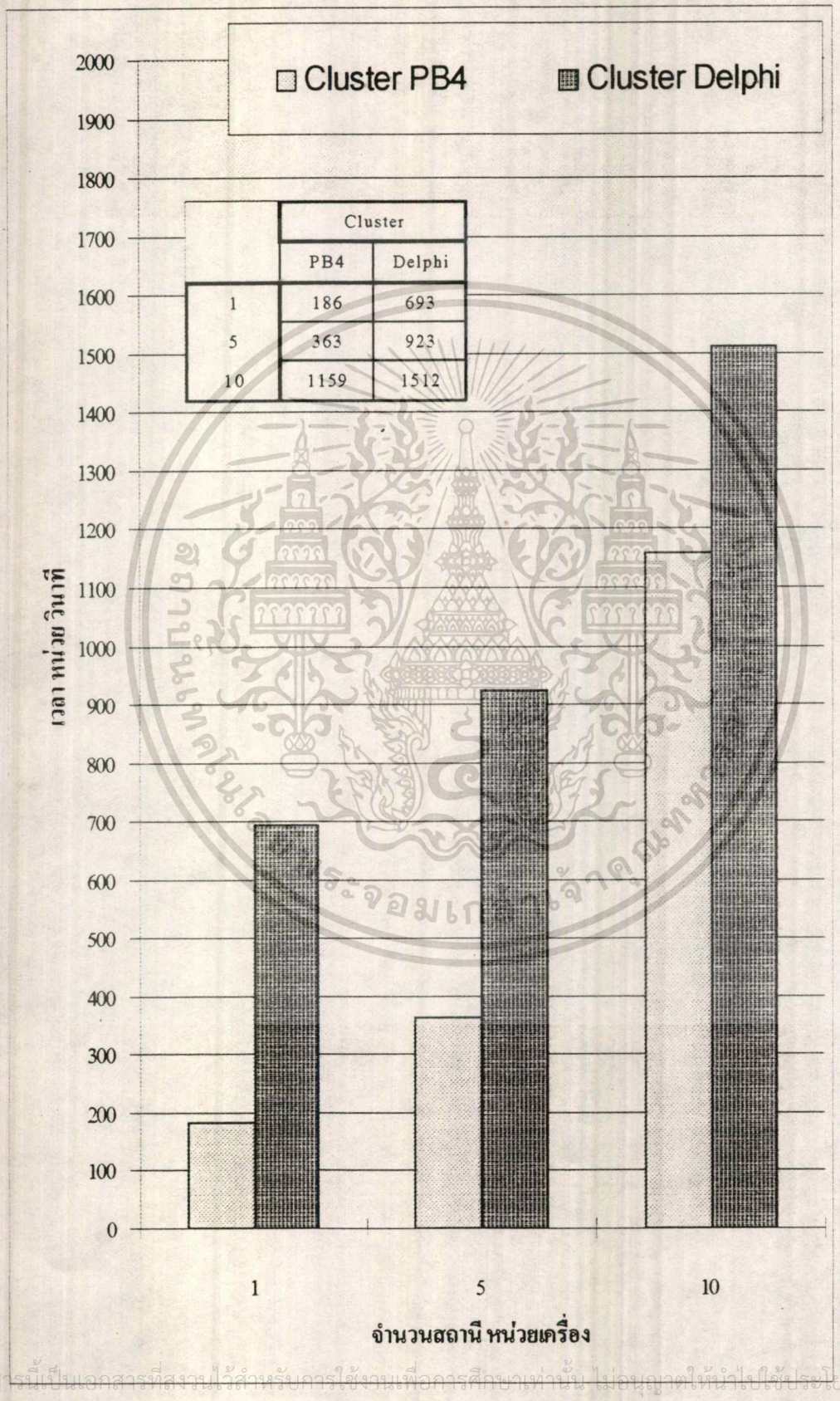
Back End คือ Oracle 7.1 for NT ทำการติดต่อแบบ Native Driver



	Index	
	PB4	Delphi
1	226	683
5	344	1053
10	817	1521

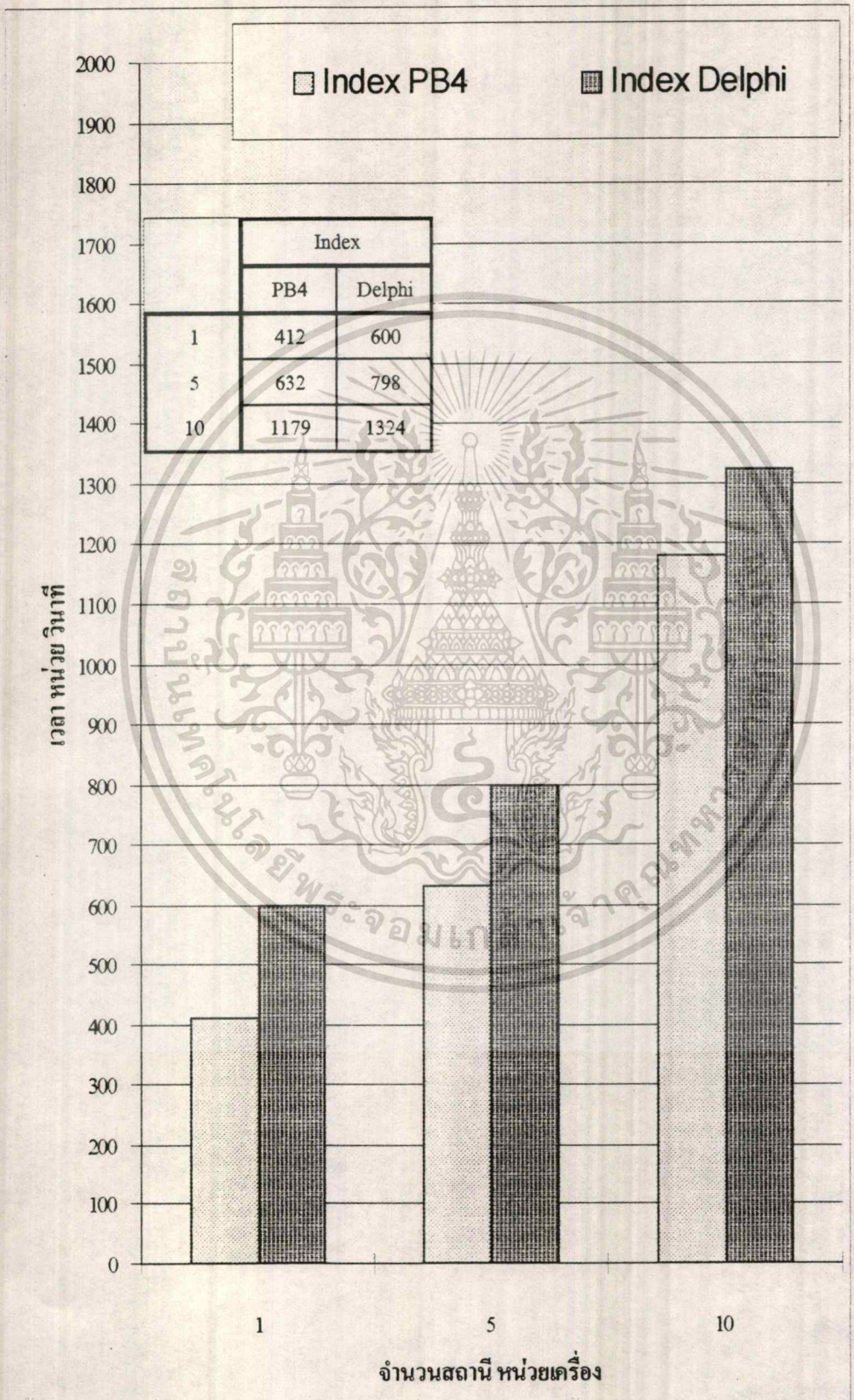
ผลการทดลอง Insert ข้อมูลขนาด 5,000 rows โดยใช้ Front End คือ Pb4,Delphi

Back End คือ Oracle 7.1 for NT ทำการติดต่อแบบ Native Driver



	Cluster	
	PB4	Delphi
1	186	693
5	363	923
10	1159	1512

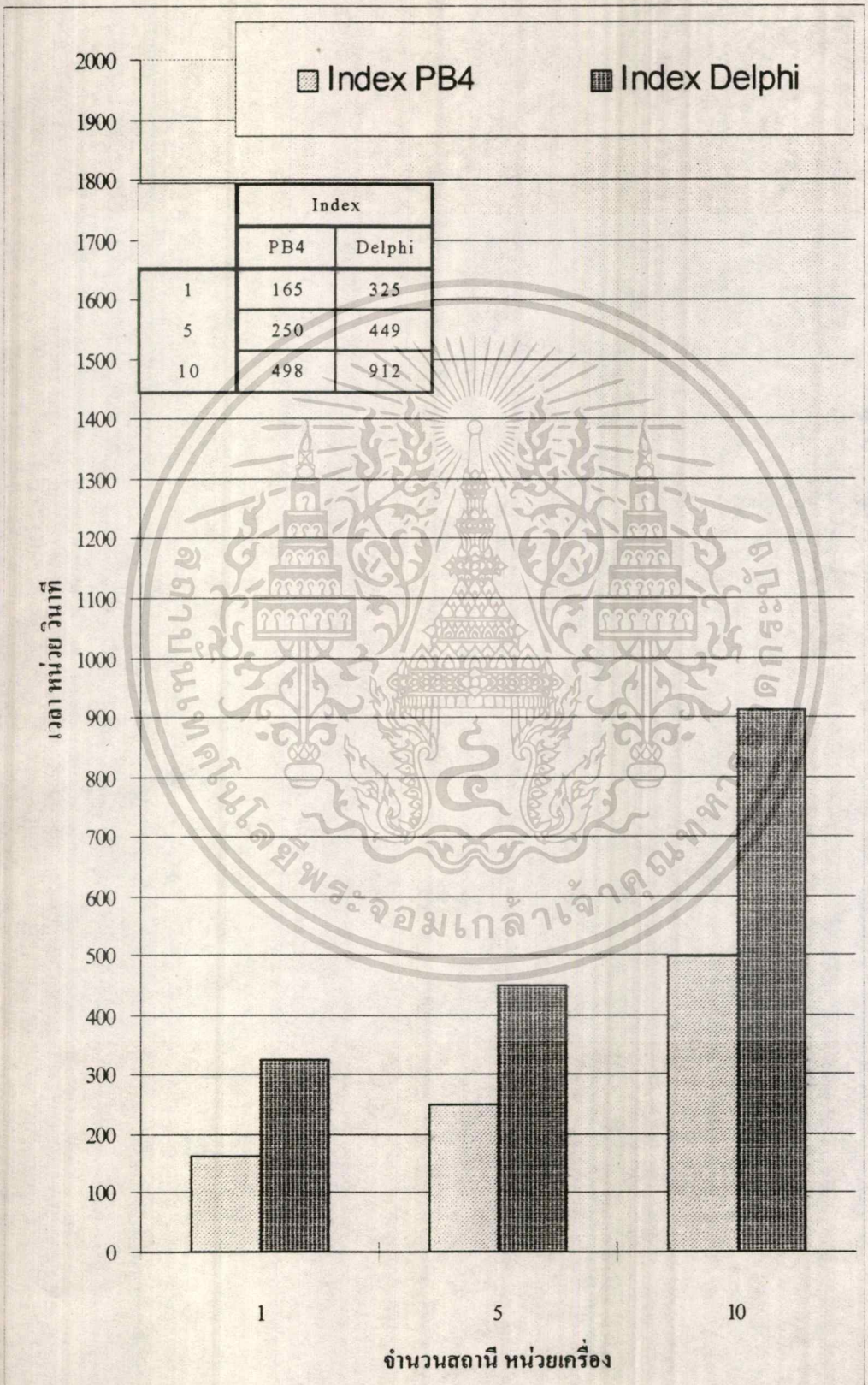
ผลการทดลอง Insert ข้อมูลขนาด 1,000,000 rows โดยใช้ Front End คือ Pb4,Delphi Back End คือ Oracle 7.1 for Sun ทำการติดต่อแบบ ODBC



	Index	
	PB4	Delphi
1	412	600
5	632	798
10	1179	1324

ผลการทดลอง Insert ข้อมูลขนาด 1,000,000 rows โดยใช้ Front End คือ Pb4,Delphi

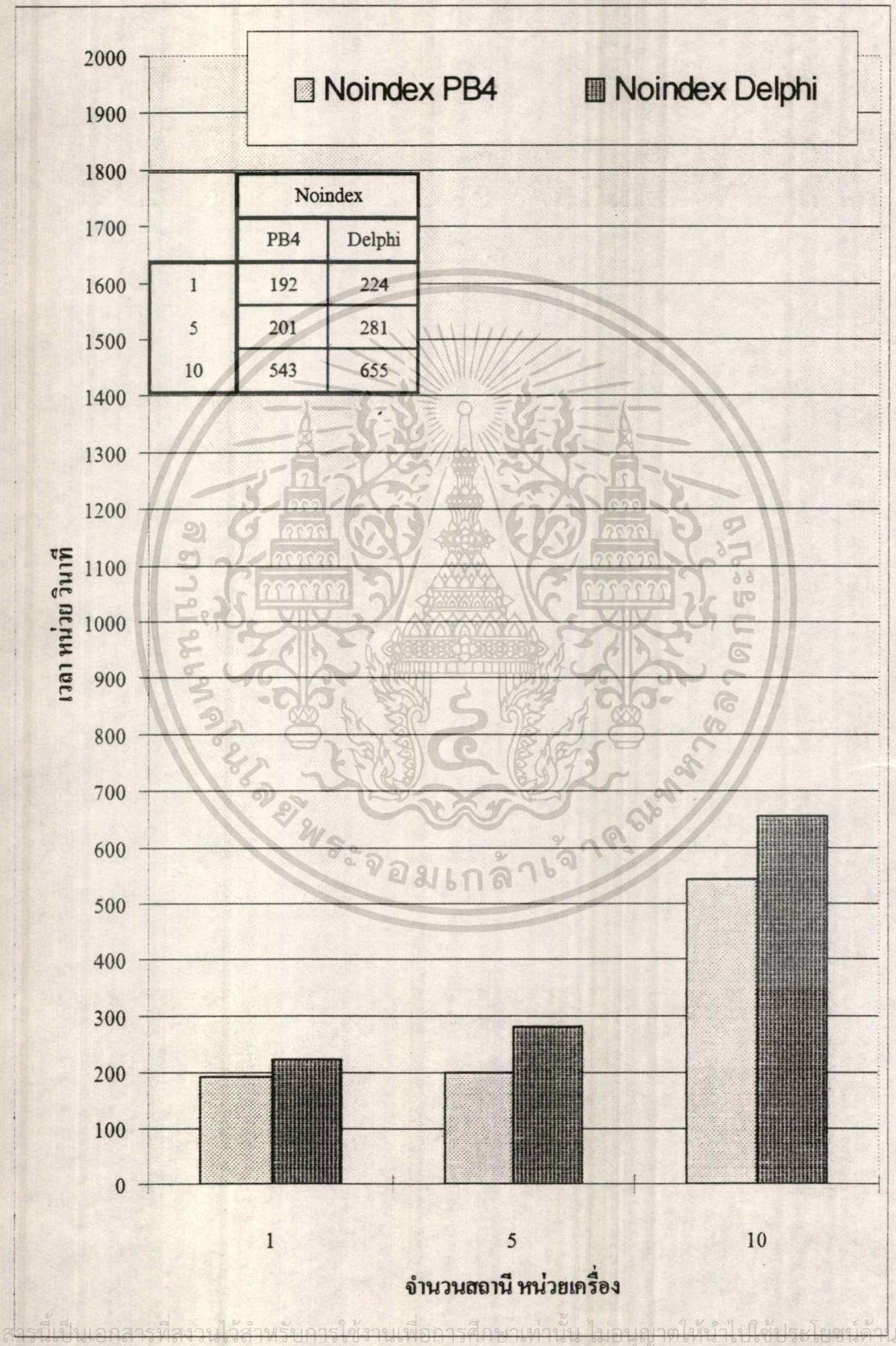
Back End คือ Oracle 7.1 for Sun ทำการติดต่อแบบ Native Driver



	Index	
	PB4	Delphi
1	165	325
5	250	449
10	498	912

ผลการทดลอง Insert ข้อมูลขนาด 1,000,000 rows โดยใช้ Front End คือ Pb4,Delphi

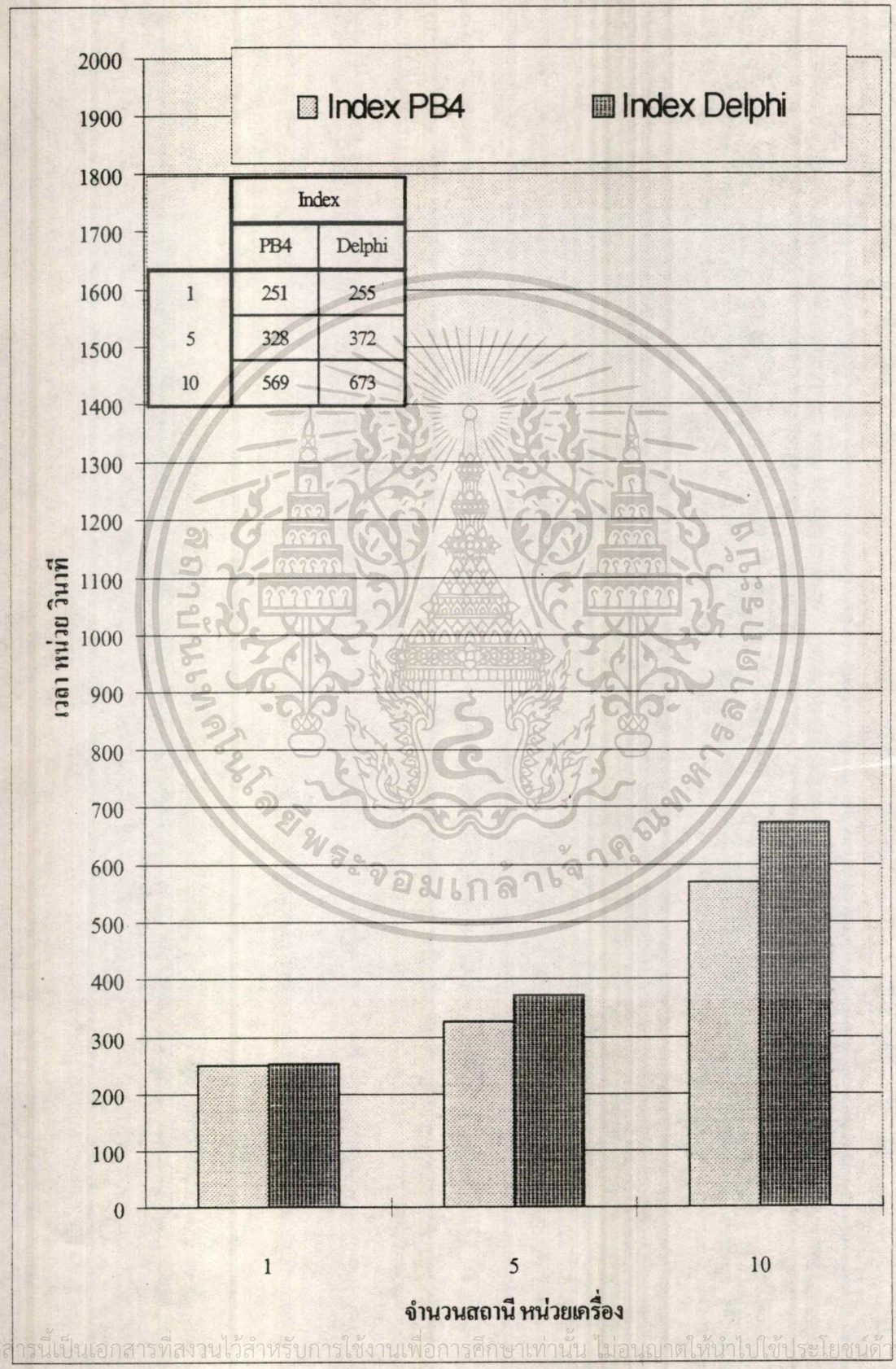
Back End คือ SQL-Server 4.21 for NT ทำการติดต่อแบบ ODBC



	Noindex	
	PB4	Delphi
1	192	224
5	201	281
10	543	655

ผลการทดลอง Insert ข้อมูลขนาด 1,000,000 rows โดยใช้ Front End คือ Pb4,Delphi

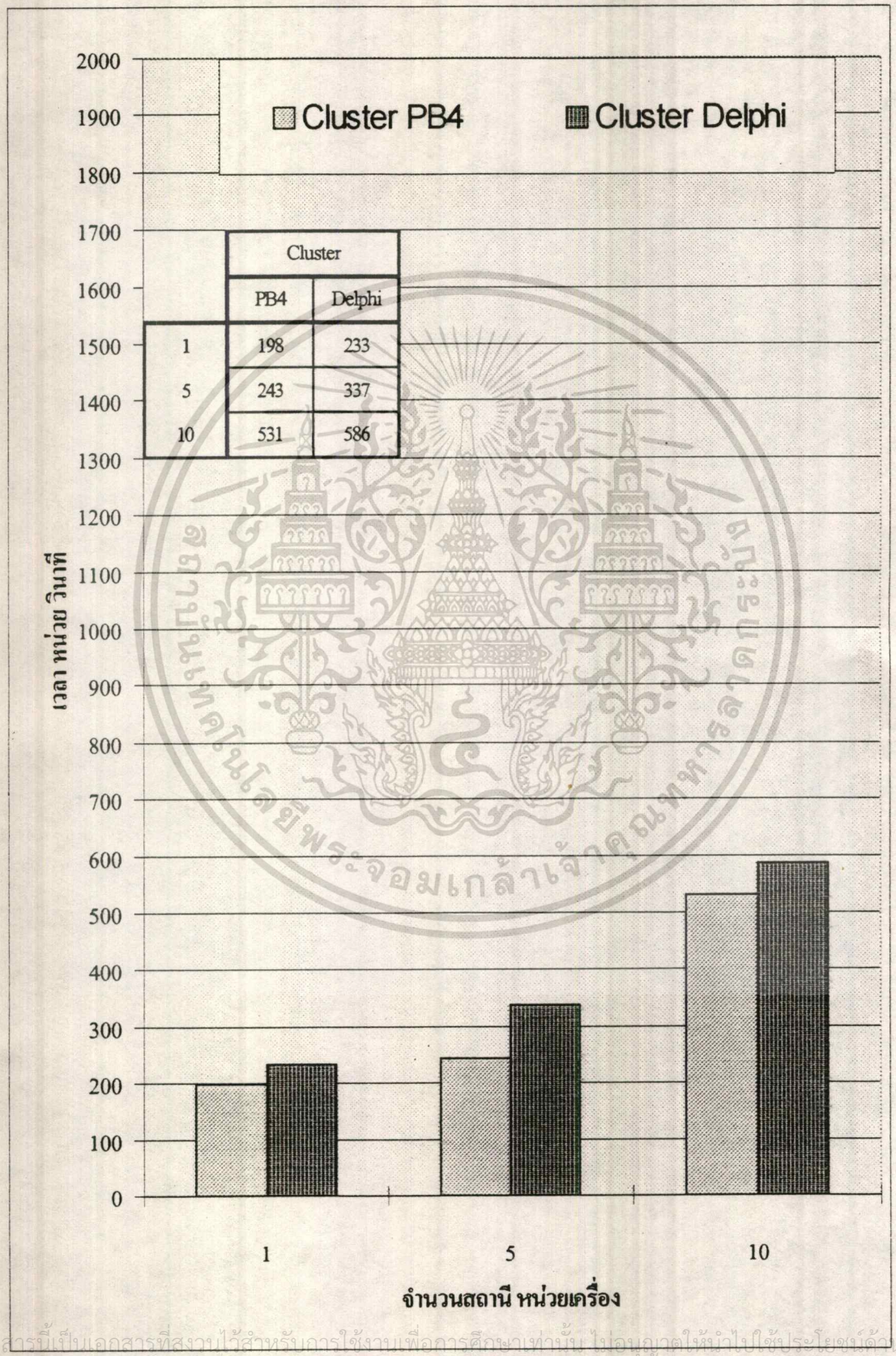
Back End คือ SQL-Server 4.21 for NT ทำการติดต่อแบบ ODBC



	Index	
	PB4	Delphi
1	251	255
5	328	372
10	569	673

ผลการทดลอง Insert ข้อมูลขนาด 1,000,000 rows โดยใช้ Front End คือ Pb4,Delphi

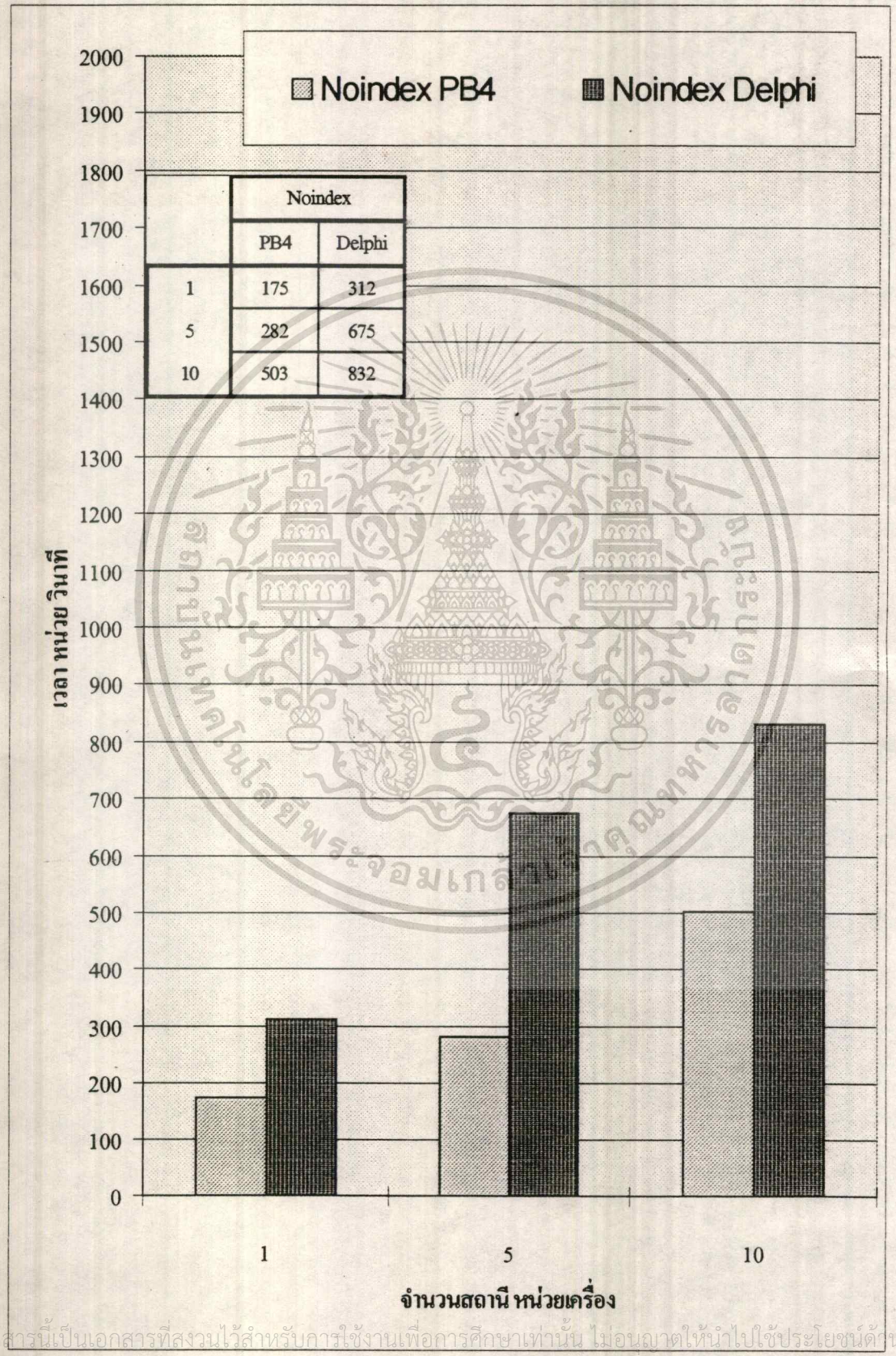
Back End คือ SQL-Server 4.21 for NT ทำการติดต่อแบบ ODBC



	Cluster	
	PB4	Delphi
1	198	233
5	243	337
10	531	586

ผลการทดลอง Insert ข้อมูลขนาด 1,000,000 rows โดยใช้ Front End คือ Pb4,Delphi

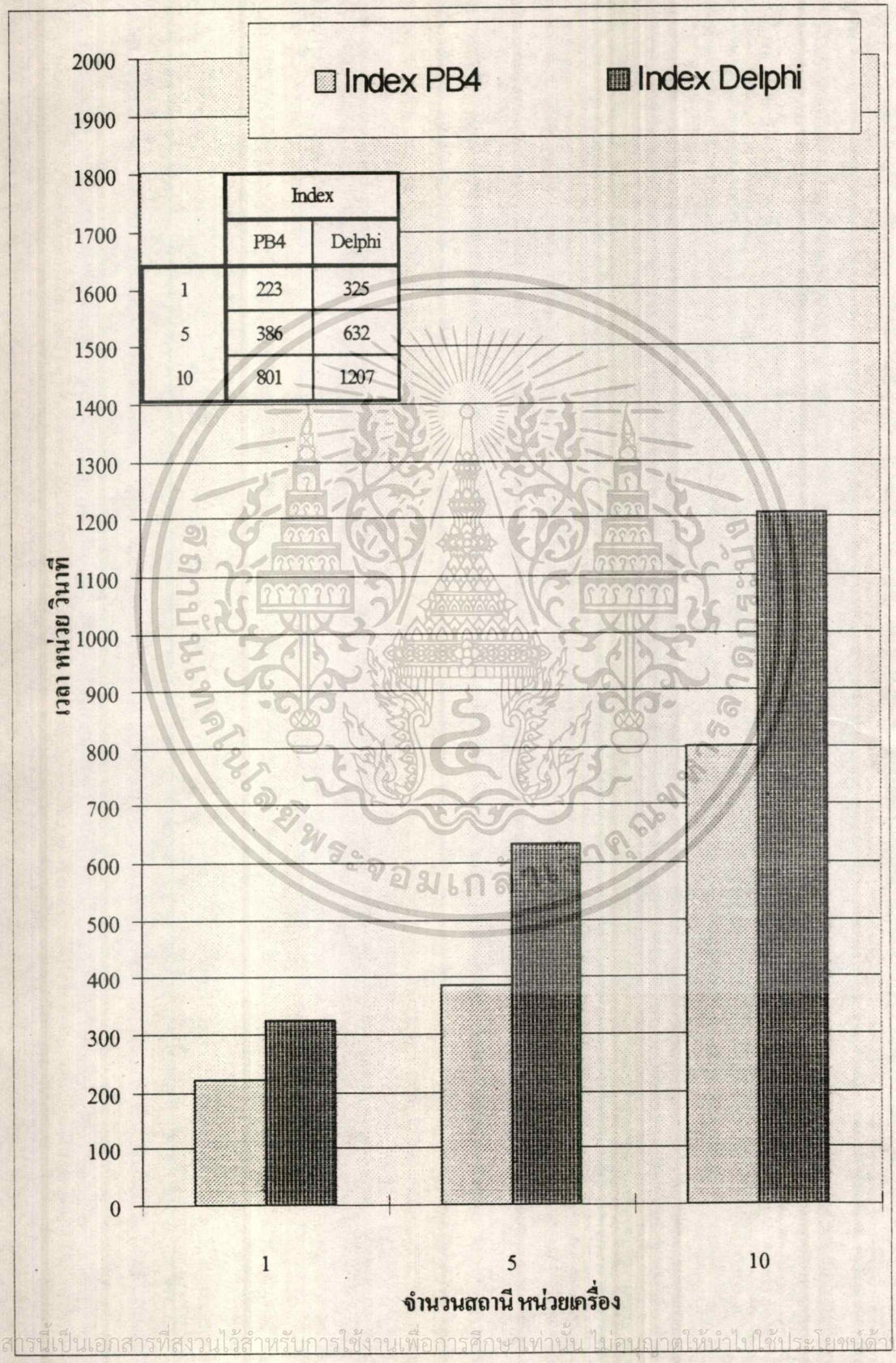
Back End คือ SQL-Server 4.21 for NT ทำการติดต่อแบบ Native Driver



	Noindex	
	PB4	Delphi
1	175	312
5	282	675
10	503	832

ผลการทดลอง Insert ข้อมูลขนาด 1,000,000 rows โดยใช้ Front End คือ Pb4,Delphi

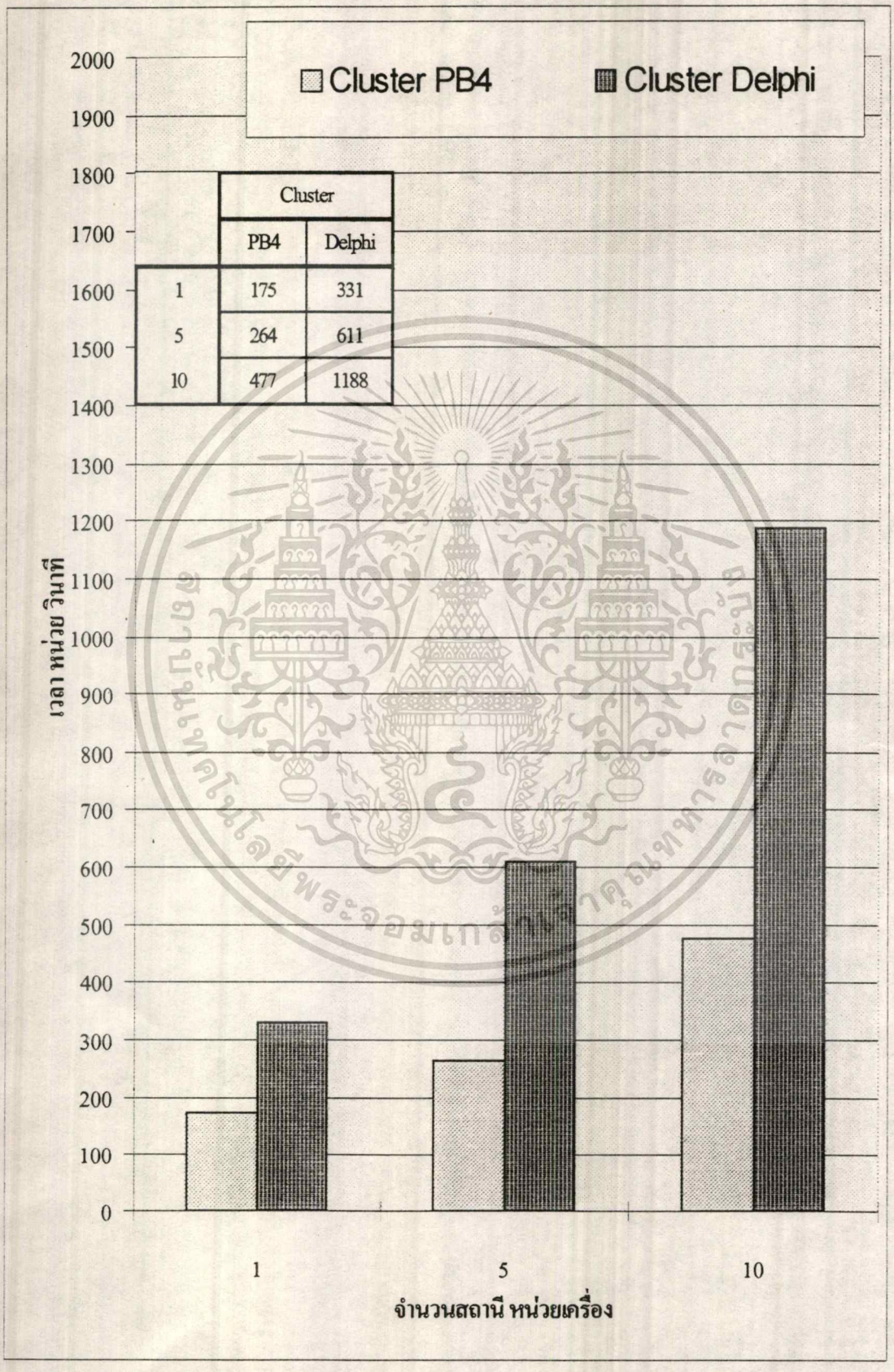
Back End คือ SQL-Server 4.21 for NT ทำการติดต่อแบบ Native Driver



	Index	
	PB4	Delphi
1	223	325
5	386	632
10	801	1207

ผลการทดลอง Insert ข้อมูลขนาด 1,000,000 rows โดยใช้ Front End คือ Pb4,Delphi

Back End คือ SQL-Server 4.21 for NT ทำการติดต่อแบบ Native Driver



	Cluster	
	PB4	Delphi
1	175	331
5	264	611
10	477	1188

บทที่ 6
บทวิจารณ์และสรุป

จากการทดลองการทำคำสั่งที่กระทำกับข้อมูลในตารางที่มีโครงสร้างของข้อมูลที่แตกต่างกัน จะทำเห็นว่า โครงสร้างของข้อมูลที่เหมาะกับการทำคำสั่ง แทรกข้อมูล (ใช้เวลาในการทำงานน้อย) จะไม่เหมาะกับการทำคำสั่งเลือกข้อมูล ดังนั้นในการเลือกใช้โครงสร้างของตารางในระบบฐานข้อมูลเชิงสัมพันธ์ในรูปแบบใดจะพิจารณาให้เหมาะสมกับลักษณะของงานที่จะต้องกระทำบ่อย ๆ

ในส่วนของไคล์เอ็นต์ที่ใช้ในการสร้างแอพพลิเคชันสำหรับใช้ทดสอบ เปรียบเทียบระหว่างโปรแกรมเดลไฟ และเพาเวอร์บิวเดอร์ จากการทดสอบจะมีสิ่งที่น่าสนใจ ดังนี้

โปรแกรมเดลไฟและเพาเวอร์บิวเดอร์ ต่างก็เป็นโปรแกรมที่ใช้สร้างแอพพลิเคชันที่มีการติดต่อกับระบบฐานข้อมูล โดยเดลไฟจะเป็นการเขียนโปรแกรมด้วยภาษาปาสคาล ส่วนเพาเวอร์บิวเดอร์จะเขียนโดยใช้ภาษาเพาเวอร์สคิป ซึ่งจะมีลักษณะของการโปรแกรมโดยทั่วไปที่ง่ายและเรียนรู้ได้เร็วกว่าเดลไฟ ส่วนในด้านการใช้งานกับฐานข้อมูลเดลไฟจะมีการใช้งานได้ง่ายและให้ความคุ้นเคยกับผู้ใช้(user friendly)ได้ดีกว่าเนื่องจากเป็นการใช้คอมโปเนนต์ในการติดต่อและสามารถที่จะแสดงข้อมูลได้แม้ในขณะออกแบบโปรแกรม (design time)

การใช้งานขณะออกแบบโปรแกรมนั้นเดลไฟจะให้ความคุ้นเคยกับผู้ใช้มากกว่าเนื่องจากจะมีการใช้ Object Inspector เพื่อแสดงคุณลักษณะ และ เหตุการณ์ ของคอมโปเนนต์แต่ละตัวได้สะดวกกว่าเพาเวอร์บิวเดอร์

ในด้านคุณลักษณะ(feature)ที่สำคัญในการใช้งานกับฐานข้อมูลนั้น เพาเวอร์บิวเดอร์จะทำได้ดีกว่ามากและสนับสนุนการทำงานในหลาย ๆรูปแบบ เช่น การใช้ Embedded SQL ซึ่งจะสนับสนุนทั้ง without cursor SQL statement , with cursor SQL statement และ Procedure ส่วนเดลไฟจะสนับสนุนการใช้งานฐานข้อมูลแบบ with cursor และ Procedure เท่านั้น รวมทั้งการทำงานกับ SQL Statement ที่มีการส่งค่าตัวแปรได้ (Dynamic SQL) นั้นเพาเวอร์บิวเดอร์ สามารถที่จะทำได้กระทัดรัดกว่าในเดลไฟมาก

การเขียนแอพพลิเคชันที่สนับสนุน OLE,DEE โปรแกรมเพาเวอร์บิวเดอร์จะสามารถเขียนได้ง่ายและใช้งานได้กว้างกว่าการใช้เดลไฟ

การเขียนแอพพลิเคชั่นที่เป็น DLL ทั้งสองโปรแกรมสามารถที่จะทำได้แต่เพาเวอร์บิวเดอร์จะต้องทำงานร่วมกับ WATCOM C++ ในขณะที่เดลไฟสามารถจะทำได้สะดวกกว่า

และจากการผลการทดสอบจะเห็นว่าการใช้งานกับระบบฐานข้อมูลในเพาเวอร์บิวเดอร์จะทำได้ดีกว่าเดลไฟเนื่องมีเวลาตอบสนองที่เร็วกว่า และในกรณีของเพาเวอร์บิวเดอร์เมื่อมีการใช้งานระบบฐานข้อมูลโดยผ่าน เนทีฟ ไดร์เวอร์จะดีว่าการใช้งานโดยผ่าน โอดีบีซี ไดร์เวอร์ แต่ในกรณีของโปรแกรมเดลไฟจะใช้งานระบบฐานข้อมูลโดยผ่านโอดีบีซี ไดร์เวอร์ จะดีว่าการใช้งานโดยผ่าน เนทีฟ ไดร์เวอร์

จากการทดสอบนี้ อาจจะมีข้อผิดพลาดของผลการทดสอบ ซึ่งเกิดได้เนื่องจากหลายสาเหตุ เช่น

- ข้อผิดพลาดที่เกิดขึ้นจากระบบเครือข่าย เนื่องจากการทดสอบจะมีการรับ-ส่งข้อมูลโดยผ่านทางระบบเครือข่าย
- ข้อผิดพลาดที่เกิดขึ้นจากกรณีที่ เซอร์ฟเวอร์มีการทำงานหลายโปรเซส โดยที่เซอร์ฟเวอร์แต่ละตัวมีการทำงานโปรเซสไม่เท่ากัน ทำให้เซอร์ฟเวอร์ที่ทำหลาย ๆโปรเซสทำงานได้ช้ากว่า
- ข้อผิดพลาดที่เกิดขึ้นเนื่องจากการสร้างแอพพลิเคชันโดยใช้เคลไฟและเพาเวอร์บิวเดอร์ โดยโปรแกรมที่เขียนขึ้นอาจจะใช้วิธีการที่ไม่เหมือนกัน
- ข้อผิดพลาดที่เกิดขึ้นจากการสร้างตารางในระบบจัดการฐานข้อมูลใช้วิธีการที่ไม่เหมือนกัน ในกรณีที่ใช้ระบบจัดการฐานข้อมูลที่แตกต่างกัน หรือในกรณีที่ใช้ระบบจัดการฐานข้อมูลเดียวกันแต่ ถ้าหากสร้างตารางขึ้นด้วยค่า พารามิเตอร์ (Parameter) ที่แตกต่างกันก็จะให้ผลที่แตกต่างกัน

กิตติกรรมประกาศ

การทำปริญญานิพนธ์ในครั้งนี้ ผู้จัดทำขอขอบคุณ

1. ดร. บุญธีร์ เครือตราชู ที่ดูแลและให้คำแนะนำปรึกษาตลอดมา
2. ผศ.ดร. ศุภมิตร จิตตะยโสธร ผู้สอนวิชา Database และให้ความรู้เกี่ยวกับระบบฐานข้อมูล
3. คุณ บัณฑิต จาก บริษัท Sytech ที่ให้คำแนะนำในการใช้งาน PowerBuilder และ Oracle
4. ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์ ที่เอื้อเฟื้อและสนับสนุน ทางด้านเครื่องไมโครคอมพิวเตอร์ ที่ใช้ในการทดสอบ
5. สำนักวิจัยและบริการคอมพิวเตอร์ ที่เอื้อเฟื้อและสนับสนุน ทางด้าน Software และ เครื่อง SUN SPARC 20 Server
6. ขอบคุณอย่างสูงสำหรับพี่ กฤษดา ที่ช่วยอำนวยความสะดวกและให้คำแนะนำในการใช้งาน Oracle 7 บนเครื่อง SUN SPARC

เอกสารอ้างอิง

1. CHARIES WOOD , BLAINE BICKAR , "Using PowerBuilder" , Que Corporation.
2. C.J. DATE , "An Introduction to Database Systems Volume I" , Addison-Wesley Publishing Company
3. DAVID McCLANAHAN , "PowerBuilder 4 A Developer's Guide" , M&T Books.
4. HENRY F. KORTH , ABRAHAM SILBERSCHATZ , "Database System Concepts", McGraw-Hill , Inc.
5. Kent Marsh , Bruce Braunstein , "PowerBuilder 4 developer's Guide" , SAMs Publishing.
6. "Database Application Developer's Guide" , Borland International , Inc.
7. "Delphi User's Guide" , Borland International , Inc.
8. "Oracle RDBMS Database Administrator's Guide version 6" , Oracle Corporation.
9. "Oracle 7 Server Concepts Manual" , Oracle Corporation.
10. "Oracle 7 Server SQL Language Reference Manual" , Oracle Corporation.
11. "System Administrator's Guide" , Microsoft Corporation.
12. "Tranact-SQL Reference" , Microsoft Corporation.