ห้องสมุดคณะเทคโนโลยีสารสนเทศ สจล.

การศึกษาระบบฐานข้อมูลแบบขนาน

A Parallel DBMS Study

โดย

นาย ณรงค์ ประติพัทธ์พงษ์

รหัส 38626040

อาจารย์ที่ปรึกษา

รศ.ดร.ศุภมิตร จิตตะยโศธร

วัน เดือน ปี 07 ธ.ค. 2549
เลขทะเบียน 01527
เลขเรียกหนังสือ สพ ณ 2121 2540
"ห้องสมุดคณะเทคโนโลยีสารสนเทศ สจล."

รายงานฉบับนี้เป็นส่วนหนึ่งของวิชาโครงงานพัฒนาระบบงาน

หลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ

ภาคเรียนที่ 1 ปีการศึกษา 2540

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ชื่อหัวข้อ	การศึกษาระบบฐานข้อมูลแบบขนาน
นักศึกษา	นาย ณรงค์ ประติพัทธ์พงษ์
อาจารย์ที่ปรึกษา	รศ.ดร. ศุภมิตร จิตตะยโศธร
ระดับการศึกษา	วิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
แขนงวิชา	วิทยาการสารสนเทศ
พ.ศ.	2540

บทคัดย่อ

ระบบฐานข้อมูลแบบขนาน เป็นระบบฐานข้อมูลที่ทำงานอยู่บนเครื่องคอมพิวเตอร์แบบขนาน ที่มี CPUมากกว่า 1 ตัวขึ้นไปช่วยกันทำงานกับคำสั่งที่เข้ามาในระบบ สถาปัตยกรรมในระบบมีอยู่ 3 ประเภท คือ shared memory system, shared disk system และ shared nothing system ระบบฐานข้อมูลแบบขนานแบ่งได้เป็น 2 ประเภท คือ ประเภทที่ผู้ผลิต hardware และ software เป็นบริษัทเดียวกัน กับ ประเภทที่ผู้ผลิต hardware และ software เป็นคนละบริษัท ในปัจจุบันได้มีการนำระบบฐานข้อมูลแบบขนานไปใช้ในงาน 2 ประเภท คือ On-line transaction processing (OLTP) system และ Decision support system (DSS) ระบบฐานข้อมูลที่สามารถทำงานในแบบขนานมีอยู่หลายๆผลิตภัณฑ์ เช่น Oracle, Sybase, Informix เป็นต้น

โครงงานฉบับนี้ได้วัดคุณสมบัติการทำงานระบบฐานข้อมูลแบบขนานหลายๆผลิตภัณฑ์ เพื่อที่จะได้ทำการตรวจสอบว่า ระบบฐานข้อมูลผลิตภัณฑ์ไหนที่สามารถทำงานในแบบขนานได้ดีที่สุด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Title	A Parallel DBMS Study
student	Mr. Narong Pratipatpong
Advisor	Assoc. Prof. Dr. Suparmit Jittayasothorn
level of study	Master of Science in Information Technology
Major	Information Science
Year	1997

ABSTRACT

A Parallel DBMS is a database system which works on parallel computers. A parallel computer is a computer which has more than one CPU. The computer's architecture has 3 categories, being Shared Memory System. Shared Disk System and Non Sharing System. A parallel DBMS also has 2 categories , being same vendor Hardware and Software and different vendor Hardware and Software. At present, parallel DBMS is used in On-line transaction processing (OLTP) system and Decision support system (DSS).Many vendors provide parallel DBMS such as Oracle , Sybase and Informix.

This project measures the property of parallel DBMS , to determine the most efficient parallel DBMS.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

ขอขอบพระคุณ รศ.ดร. ศุภมิตร จิตตะยโศธร ที่ได้กรุณารับเป็นอาจารย์ที่ปรึกษา ให้คำแนะนำที่เป็นประโยชน์ ให้กำลังใจ ตลอดจนช่วยเหลือในการตรวจสอบและแก้ไขเนื้อหาเป็นอย่างดี

ขอขอบคุณ ท่านอาจารย์ทุกท่านที่ประสิทธิ์ประสาทความรู้ ตลอดระยะเวลาที่ผ่านมา รวมทั้งเพื่อนร่วมรุ่นที่คอยเป็นห่วงและให้กำลังใจด้วยดีตลอดระยะเวลาที่ศึกษา

ขอบคุณ คุณพิพัฒน์ ที่ได้ให้คำแนะนำที่เป็นประโยชน์ในการทำงาน

สุดท้ายขอกราบขอบพระคุณ บิดา มารดา ที่ได้ให้กำลังใจ และสนับสนุนด้านทุนทรัพย์ด้วยดีตลอดมา

ณรงค์ ประติพัทธ์พงษ์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ(ต่อ)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญตาราง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญภาพ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญภาพ (ต่อ)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 1
บทนำ

1.1 ความเป็นมา

ในปัจจุบันระบบคอมพิวเตอร์แบบขนานที่มี CPUมากกว่า 1 ตัวขึ้นไป เป็นที่นิยมใช้กันอย่างแพร่หลาย เนื่องจากจะทำให้ประสิทธิภาพการทำงานในระบบสูงขึ้น เพราะระบบสามารถที่จะใช้ CPU หลายๆตัวช่วยกันทำงานอย่างใดอย่างหนึ่งพร้อมๆกัน ระบบฐานข้อมูลแบบขนานสามารถที่จะใช้ประโยชน์จากระบบคอมพิวเตอร์แบบขนาน โดยที่ระบบสามารถที่จะใช้ CPU หลายๆตัวที่มีอยู่ทำงานกับคำสั่งใดคำสั่งหนึ่งพร้อมๆกันได้ ทำให้ระบบฐานข้อมูลแบบขนานสามารถค้นหาข้อมูลได้รวดเร็วยิ่งขึ้นและสามารถที่จะรองรับข้อมูลได้มากขึ้นด้วย นอกจากนี้ยังช่วยป้องกันความผิดพลาดอันเกิดมาจาก อุปกรณ์ใดอุปกรณ์หนึ่งในระบบเกิดความเสียหาย

1.2 วัตถุประสงค์

1 ศึกษาว่าระบบฐานข้อมูลแบบขนานคืออะไร แบ่งเป็นกี่ประเภทอะไรบ้าง สถาปัตยกรรมที่ใช้ในการทำงานรวมทั้งข้อดี-ข้อเสียในแต่ละประเภท

2 ศึกษาว่าในปัจจุบันเทคโนโลยีที่ใช้ในระบบฐานข้อมูลแบบขนานมีอะไรบ้าง

3 ศึกษาถึงการทำงานของระบบฐานข้อมูลแบบขนานในแต่ละประเภท

4 ศึกษารายละเอียดการทำงานของระบบฐานข้อมูลแบบขนานที่มีจำหน่ายอยู่ในปัจจุบัน โดยในที่นี้จะใช้ Oracle เป็นกรณีตัวอย่าง

5 เปรียบเทียบคุณสมบัติการทำงาน ของระบบฐานข้อมูลแบบขนานในแต่ละผลิตภัณฑ์

1.3 ประโยชน์ที่คาดว่าจะได้รับ

1 เข้าใจการทำงานของระบบฐานข้อมูลแบบขนานในแต่ละสถาปัตยกรรม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2 ได้ทราบว่าเทคโนโลยีที่ใช้ในระบบฐานข้อมูลแบบขนานมีอะไรบ้าง และระบบฐานข้อมูลแบบขนานที่มีจำหน่ายอยู่ในปัจจุบัน มีความสามารถที่จะทำงานในแบบขนานได้อย่างไร

3 ได้มีการวัดคุณสมบัติการทำงานของระบบฐานข้อมูลแบบขนานในหลายๆ ผลิตภัณฑ์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 2
ระบบฐานข้อมูลแบบขนาน

2.1 นิยาม

ระบบฐานข้อมูลแบบขนาน คือ ระบบฐานข้อมูลที่ทำงานอยู่บนคอมพิวเตอร์ที่มี CPU มากกว่า 1 ตัวขึ้นไป โดยที่ระบบสามารถทำงานกับคำสั่งหลายๆคำสั่งพร้อมๆกัน โดยใช้ CPU หลายๆตัวช่วยกันทำงานกับคำสั่งเหล่านั้น ทำให้ประสิทธิภาพการทำงานของระบบสูงขึ้น สามารถรองรับข้อมูลได้มากขึ้นและสามารถทำงานได้รวดเร็วยิ่งขึ้น ระบบฐานข้อมูลจะหมายถึงระบบฐานข้อมูลแบบ relational เนื่องจากว่าเป็นระบบฐานข้อมูลที่มีความเหมาะสมที่จะทำงานในแบบขนาน เพราะสาเหตุ 2 ประการ

1 คำสั่งใน relational database สามารถแยกออกเป็นส่วนย่อยๆ หลายๆส่วน ที่มีอิสระต่อกันในการทำงาน แต่ละส่วนสามารถกระจายไปให้ CPU แต่ละตัวทำงานแต่ละส่วนพร้อมๆกัน

2 คำสั่งใน relational database สามารถที่จะให้ CPU หลายๆตัว ช่วยกันทำงานกับคำสั่งนั้นพร้อมๆกัน

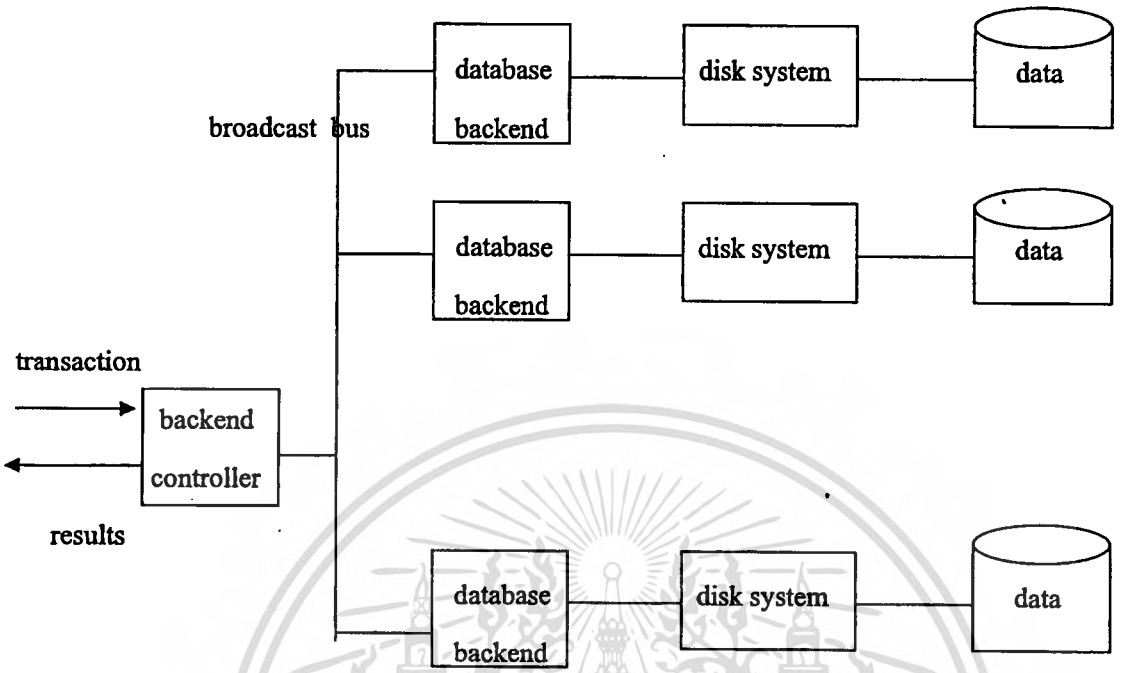
ระบบฐานข้อมูลแบบขนานจะทำงานอยู่บนระบบคอมพิวเตอร์แบบขนาน ซึ่งเป็นระบบคอมพิวเตอร์ที่มี CPU, memory และ disks อยู่หลายๆตัวภายในระบบ โดยที่ระบบคอมพิวเตอร์แบบขนานจะมีอยู่ 2 ประเภทใหญ่ๆ คือ [10]

1 ระบบคอมพิวเตอร์ที่มี CPU อยู่หลายๆตัวในระบบ แต่ว่าจะใช้ memory และ disk ร่วมกัน ระบบคอมพิวเตอร์ประเภทนี้จะเรียกว่า Symmetric Multiprocessing (SMP) system

2 ระบบคอมพิวเตอร์ที่มี CPU อยู่หลายๆตัว ในแต่ละ CPU จะมี memory และ disk เป็นของตนเอง ไม่มีการใช้งานร่วมกับ CPU ตัวอื่น การติดต่อระหว่าง CPU กับ CPU เป็นการติดต่อผ่านทางระบบ network ระบบคอมพิวเตอร์ประเภทนี้จะเรียกว่า Massively Parallel processing (MPP) system

แต่ว่าในระบบฐานข้อมูลแบบขนานยังสามารถทำงานบนระบบคอมพิวเตอร์แบบขนานที่มี database backend หลายๆ backend ดังภาพที่ 1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 1 แสดง database backend system

จากภาพระบบจะมี backend controller คอยรับ transaction จาก user เข้ามาในระบบ หลังจากนั้น backend controller จะกระจาย transaction ออกไปยัง database backend ทุกๆตัวที่มีอยู่ในระบบพร้อมๆกัน เมื่อแต่ละ database backend รับ transaction จาก controller แล้ว จะนำ transaction นั้นมาทำงานกับข้อมูลที่อยู่ใน disk ทันที โดยที่ในแต่ละ database backend จะมี CPU และ memory เป็นของตนเอง การทำงานในแบบขนานก็คือ การที่ transaction แต่ละ transaction ถูกทำงานในแต่ละ database backend พร้อมๆกัน เมื่อแต่ละ database backend ทำงานกับ transaction นั้นแล้วจะส่งผลลัพธ์จากการทำงานมาที่ backend controller เพื่อรวบรวมผลที่ได้ในแต่ละ database backend เข้าด้วยกันกลายเป็นผลลัพธ์ของ transaction นั้น ก่อนที่จะนำผลลัพธ์ส่งไปให้ user ต่อไป

ในการทำงาน backend controller อาจจะไม่ต้องส่ง transaction ไปที่ database backend ทุกๆตัวที่มีอยู่อาจจะส่ง transaction ไปที่ database backend ใด backend หนึ่งซึ่งจะเป็นการทำงานในแบบ 1-1 และ backend controller อาจจะติดต่อกับ computer อื่นๆเพื่อรับ transaction จากคอมพิวเตอร์เครื่องอื่นเข้ามาทำงานในแบบขนานในแต่ละ database backend แทนที่จะรับ transaction จาก user เพียงอย่างเดียว ภาพนี้จะเป็นการทำงานสถาปัตยกรรมระบบฐานข้อมูลแบบขนานในรูปแบบของ shared nothing system

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.2 ข้อดีของระบบฐานข้อมูลแบบขนาน [1][2][5]

1 ทำให้ระบบมีประสิทธิภาพมากขึ้น ประสิทธิภาพในที่นี้หมายถึง ระบบสามารถรองรับข้อมูลที่มีปริมาณเพิ่มขึ้น และสามารถค้นหาข้อมูลได้อย่างรวดเร็วยิ่งขึ้น

2 เนื่องจากระบบมี CPU, memory และ disk อยู่ในระบบอยู่หลายๆตัวมันจะสามารถป้องกันความผิดพลาดอันเกิดจากอุปกรณ์ภายในระบบเสีย เช่น การทำ data replication โดยระบบจะเก็บชุดเดียวกันออกเป็นหลายๆ disk โดยใช้ protocol ROWA (read one write all) เช่น ระบบสามารถเก็บข้อมูลชุดเดียวกันไว้ 2 disk ระบบจะเขียนข้อมูลลงไปใน disk ทั้ง 2 ตัว พร้อมๆกัน แต่เวลาอ่านข้อมูลจะอ่านเพียง disk เดียว ซึ่งจะช่วยป้องกันความผิดพลาดที่เกิดจาก disk ใด disk หนึ่งเสียระบบก็ยังสามารถอ่านข้อมูลอีก disk หนึ่งได้หรือในกรณีที่ CPU ตัวใดตัวหนึ่งในระบบเสียระบบก็ยังสามารถใช้ CPU ที่เหลือทำงานต่อไปได้

3 ระบบสามารถขยายความสามารถของระบบโดยการเพิ่มจำนวน CPU, memory และ disk เข้ามาในระบบได้อีกขึ้นอยู่กับสถาปัตยกรรมของระบบที่จะสนับสนุนให้เพิ่มได้

4 ระบบสามารถรองรับผู้ใช้ในปริมาณที่มากๆได้ โดยในบางระบบสามารถจะรองรับผู้ใช้ได้มากกว่า 1000 คนขึ้นไปที่จะเข้ามาใช้ข้อมูลพร้อมๆกัน

2.3 ประเภทของระบบฐานข้อมูลแบบขนาน

ระบบฐานข้อมูลแบบขนานสามารถแบ่งได้เป็น 2 ประเภทใหญ่ๆ ตามลักษณะของ hardware และ software ที่ใช้งาน

1 ผู้ผลิต hardware และ software เป็นบริษัทเดียวกัน ประเภทนี้ได้แก่ software ที่ชื่อ teradata database system ของ บริษัท NCR system software ประเภทนี้จะทำงานอยู่บน 2 platform คือ [6]

1.1 Teradata database system for unix ประเภทนี้จะทำงานอยู่บนระบบ unix

1.2 Teradata database system for NCR system 3600 and DBS/1012 ประเภทนี้ จะทำงานบนเครื่อง NCR system รุ่น 3600/3600-XP servers

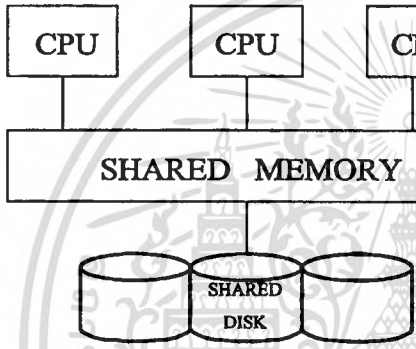
2 ผู้ผลิต hardware และ software เป็นคนละบริษัท ประเภทนี้ผู้ผลิต software จะผลิต software ให้สามารถทำงานบนเครื่องคอมพิวเตอร์แบบขนานทุกบริษัท และทุกๆรุ่นที่มีการผลิตออกมา ประเภทนี้มีอยู่หลายๆยี่ห้อ เช่น Oracle ,Sybase ,Informix เป็นต้น software ประเภทนี้มักจะทำงานอยู่บนเครื่องคอมพิวเตอร์ ที่ใช้ OS ที่เป็น UNIX

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.4 สถาปัตยกรรมระบบฐานข้อมูลแบบขนาน

สถาปัตยกรรมระบบฐานข้อมูลแบบขนาน สามารถแบ่งได้เป็น 3 ประเภทคือ [1][2][5][7][20]

1) Shared memory Systems รูปแบบนี้จะคล้ายกับระบบคอมพิวเตอร์แบบขนานที่เป็น Symmetric multiprocessing systems (SMP) คือภายในจะมี CPU อยู่หลายๆตัว โดยที่ในระบบจะมีการใช้ memory ,I/O และ disks ร่วมกัน การติดต่อระหว่าง CPU จะติดต่อผ่านทาง memory ที่เชื่อมต่อกันอยู่ ประสิทธิภาพของระบบจะขึ้นอยู่กับ ชนิดของ bus ที่ใช้งาน



ภาพที่ 2 แสดง Shared Memory System

ตัวอย่างของ shared memory system ที่ถูกออกแบบมาสำหรับใช้ใน parallel database system เช่น XPRS,DBS3 และ Volcano ระบบ shared memory จะใช้การค้นหาข้อมูลเป็นแบบ inter query parallelism [ ]

ข้อดี

1 การติดต่อระหว่าง CPU โดยผ่านทาง memory จะเป็นการติดต่อที่เร็วกว่าการติดต่อแบบอื่นๆ เนื่องจากเป็นการทำงานแบบ internal synchronization

2 การจัดการระบบจะทำได้ง่ายกว่าแบบอื่น

3 การเพิ่มจำนวน CPU สามารถที่จะเพิ่มเข้ามาได้โดยไม่ต้องเปลี่ยนแปลงแก้ไข config ของระบบ

ข้อเสีย

1 การเพิ่มจำนวน CPU เข้ามาในระบบจะเพิ่มได้อย่างจำกัด เนื่องจากจะต้องใช้ memory และ disks ร่วมกัน

2 ประสิทธิภาพของระบบจะขึ้นอยู่กับชนิดของ bus ที่ใช้งานถ้าใช้ bus ที่มีความเร็วต่ำๆจะทำให้การทำงานของระบบช้าไปทั้งหมด

3 เนื่องจากว่าการทำงาน CPU จะติดต่อผ่านทาง memory ดังนั้นถ้าหากว่า memory ในระบบเกิดเสีย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

CPU ในระบบจะไม่สามารถอ่านข้อมูลจาก disk ได้

2) Shared disk Systems รูปแบบนี้จะมีคุณลักษณะดังนี้.

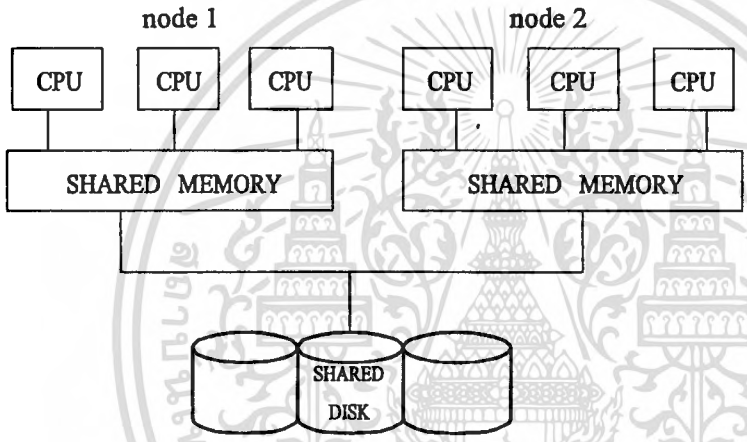
1 ระบบจะแบ่งออกเป็นแต่ละ nodeๆ จะมี CPU ได้ 1 ตัวหรือมากกว่าก็ได้

2 แต่ละ node จะมี memory เป็นของตนเองจะไม่มีการใช้ memory ร่วมกันระหว่าง node

3 แต่ละ node จะติดต่อไปยัง disks ซึ่งจะมีการใช้ disks ร่วมกัน

4 การติดต่อระหว่าง node จะติดต่อผ่านทางbus ที่มีความเร็วสูง

5 ชนิดของ bus จะเป็นตัวจำกัดจำนวน nodes ที่มีอยู่ในระบบ



ภาพที่ 3 แสดง Shared Disk System

ตัวอย่างของ shared disk system ที่ถูกออกแบบมาสำหรับ parallel database system เช่น IBM's IMS/VS data sharing product และ DEC's VAX DBMS และ RDB products ระบบ shared disk จะมีการค้นหาข้อมูลเป็นแบบ inter query parallelism

ข้อดี

1 ระบบจะมีเสถียรภาพมากขึ้น ถึงแม้ว่า node ใด node หนึ่ง จะเสียไป node อื่นก็ยังสามารถติดต่อข้อมูลที่อยู่ใน disks ได้

2 ระบบสามารถขยายความสามารถของระบบโดยการเพิ่มจำนวน node เข้ามาในระบบได้อีก

3 แต่ละ CPU จะมี memory เป็นของตนเอง ดังนั้น CPU จึงไม่ต้องติดต่อไปยัง disk เพื่อที่จะหาข้อมูลอยู่ตลอดเวลา ทำให้การติดต่อระหว่าง CPU กับ disk ลดน้อยลงเป็นการประหยัด costในการติดต่อ

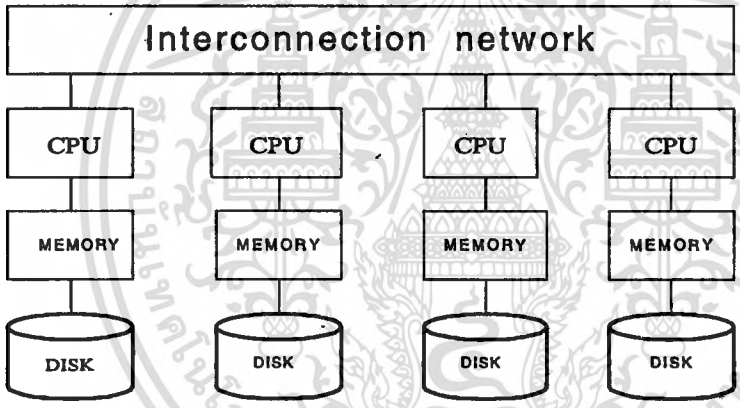
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ข้อเสีย

1 ถ้างานที่ใช้อยู่ไม่มีลักษณะที่เป็น partition จะต้องมีการติดต่อระหว่าง node มาก จะทำให้ bus ถูกใช้งานมากจนอาจทำให้เกิดการ overhead เกิดขึ้น

2 การติดต่อระหว่าง node กับ node จะต้องใช้ bus ที่มีความเร็วสูงมิฉะนั้น ระบบจะเกิดการ bottleneck เกิดขึ้น

3) Shared nothing Systems รูปแบบนี้จะคล้ายกับระบบคอมพิวเตอร์แบบขนานที่เป็นแบบ Massively Parallel Processing Systems (MPP) คือระบบจะแบ่งออกเป็น node ในแต่ละnode จะมี CPU,memory และ disk เป็นของตนเอง การติดต่อระหว่าง node กับ node จะผ่านทางระบบ network ที่เชื่อมต่อกันอยู่ โดยที่ในแต่ละ node จะมี OS และ DBMS เป็นของตนเองไม่มีการใช้งานร่วมกับ node อื่น ข้อมูลที่อยู่ใน disk จะถูกจัดการโดย node ที่ควบคุมอยู่เท่านั้น



ภาพที่ 4 แสดง Shared nothing system

ตัวอย่างของ shared nothing system ที่ถูกออกแบบมาสำหรับ parallel database system เช่น Teradata's DBC,Tandem's Nonstop SQL รวมทั้งยังมีระบบต้นแบบอีกหลายระบบ เช่น Bubba,Eds,Grace และ Arbre เป็นต้น ระบบ shared nothing system จะใช้การค้นหาข้อมูลทั้งแบบ inter query parallelism และ intra query parallelism

ข้อดี

1 รูปแบบนี้สามารถขยายขนาดของระบบได้ง่าย เนื่องจากว่าแต่ละ node จะมี memory และ disk เป็นของตนเองดังนั้นระบบสามารถขยายขนาดของระบบโดยไม่มีข้อจำกัดทางด้าน hardware

2 ระบบจะมีการป้องกันความผิดพลาดที่ดี โดยระบบสามารถทำ data replication โดยการจะก็อปปี้ข้อมูลไปยัง disk ในระบบทำให้สามารถที่จะป้องกันความเสียหายที่เกิดจาก disk พังได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ห้องสมุดคณะเทคโนโลยีสารสนเทศ สจล.

3 ถ้าหากว่า CPUตัวใดตัวหนึ่งเสียไป มันจะเสียแค่จุดนั้นเพียงจุดเดียว ส่วนอื่นก็ยังสามารถทำงานต่อไปได้

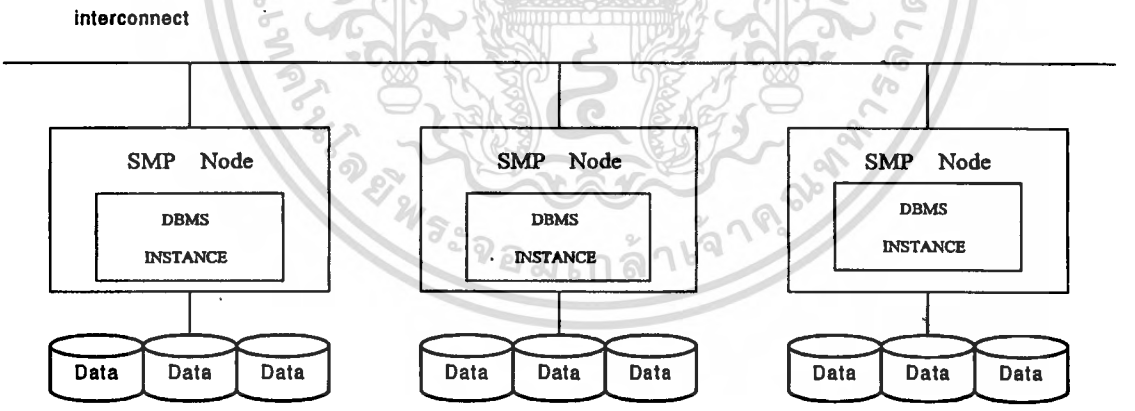
ข้อเสีย

1 ถ้า CPU ตัวใดตัวหนึ่งเสียไป CPUตัวอื่นๆ ไม่สามารถที่จะอ่านข้อมูลจาก disk ที่ทำงานกับ CPUตัวที่เสียไปได้

2 การทำงานร่วมกันระหว่าง CPU เป็นสิ่งที่มีความสำคัญมากเนื่องจากข้อมูลจะกระจายอยู่ในแต่ละ disks CPU จะต้องมีการทำงานอย่างใกล้ชิดและต้องมีการกระจายข้อมูลไปทั่วทุก disk ด้วยขนาดเท่าๆกันทุก disk

3 การขยายระบบโดยการเพิ่ม CPU จะต้องมีการแก้ไข config ของระบบใหม่ทั้งหมด

นอกจากรูปแบบทั้ง 3 แบบที่กล่าวมาแล้ว ยังมีระบบอีกแบบหนึ่งที่เรียกว่า virtual machines system รูปแบบนี้เป็นการนำเอารูปแบบของ shared memory system กับ shared nothing system มารวมกันโดยที่รูปแบบนี้จะมี SMP หลายๆ node มาเชื่อมโยงกันผ่านทางระบบ network ดังภาพ [2]



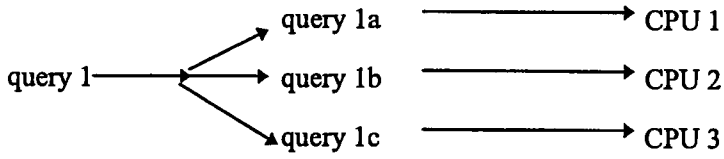
ภาพที่ 5 แสดง virtual machines

รูปแบบนี้แต่ละ SMP จะเรียกว่า 1 instance ซึ่งคำว่า instance จะหมายถึง การที่ระบบมี CPU อยู่ 1 ตัวหรือมากกว่าก็ได้มี memory เป็นของตนเอง แต่ว่าจะมี disk เป็นของตนเองหรือใช้รวมกับ instance อื่นก็ได้ ฉะนั้นแต่ละ node จะมี OS และ DBMS เป็นของตนเอง รูปแบบนี้ลักษณะที่คล้ายกับรูปแบบ distributed database system การทำ query ในระบบ parallel database สามารถแบ่งได้เป็น2 ประเภทใหญ่ๆตามลักษณะของการทำงาน

1 intraquery parallelism หรือเรียกอีกชื่อหนึ่งว่า parallel query processing วิธีนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ระบบจะรับคำสั่ง query เข้ามา 1 query จากนั้นระบบจะกระจายคำสั่งออกไปยัง CPU ที่มีอยู่ในระบบ เพื่อให้ CPU หลายๆตัวทำงานกับคำสั่ง query 1 query พร้อมๆกันดังภาพ



ภาพที่ 6 แสดง intraquery parallelism

2 interquery parallelism หรือเรียกอีกชื่อหนึ่งว่า parallel transaction processing [20] วิธีนี้ระบบจะรับคำสั่ง query เข้ามาหลายๆคำสั่ง แต่ละคำสั่งจะถูกทำงานโดย CPU แต่ละตัว จากนั้นระบบฐานข้อมูลจะเช็คว่าคำสั่งย่อยๆในคำสั่ง query คำสั่งไหนที่สามารถทำงานในแบบขนานได้ ถ้าระบบฐานข้อมูลพบว่าคำสั่งไหนมีคำสั่งย่อยคำสั่งไหนที่สามารถทำงานในแบบขนานได้ ระบบจะให้ CPU หลายๆตัวที่มีอยู่ในระบบเข้ามาทำงานกับคำสั่งนั้นพร้อมๆกัน ดังภาพ

ภาพที่ 7 แสดง interquery parallelism

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 3

หลักการทำงานในระบบฐานข้อมูลแบบขนาน

3.1 Data Partition

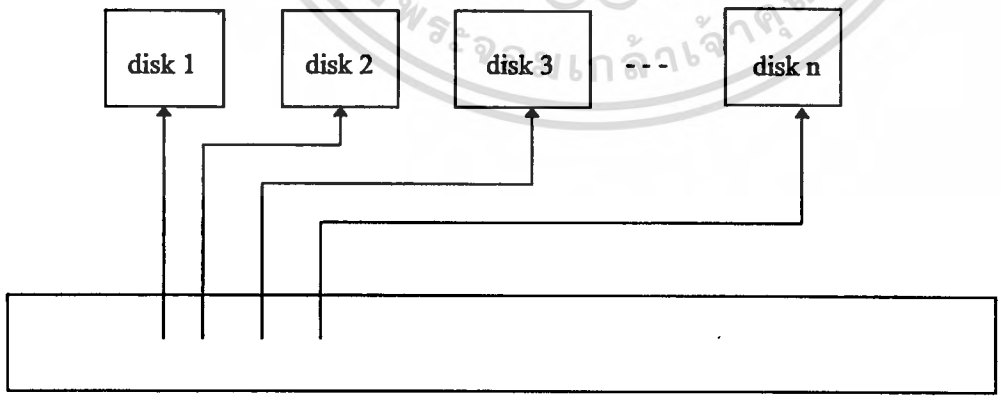
การทำ data partition เป็นการนำ tuple ที่อยู่ในแต่ละ table กระจายออกไปยัง disk ทุกๆตัวที่อยู่ในระบบ สาเหตุที่ต้องมีการทำ data partition มีอยู่ 2 ประการ

1 ขนาดของ table อาจจะมีขนาดมากกว่า disk ทำให้ต้องกระจายข้อมูลใน table ออกไปหลายๆ disk

2 เพื่อที่จะได้สามารถใช้ประสิทธิภาพของ disk ที่มีอยู่ในระบบได้อย่างเต็มประสิทธิภาพ โดยการให้ข้อมูลที่อยู่ใน table กระจายออกไปทุกๆ disk ในการค้นหาข้อมูลที่อยู่ใน table จะได้สามารถใช้ CPU ค้นหาข้อมูลที่เก็บอยู่ใน disk ทำให้สามารถค้นหาข้อมูลได้อย่างรวดเร็วมากกว่าการที่ข้อมูลจะอยู่ที่ disk ใด disk หนึ่งเพียงที่เดียว

การทำ data partition มีอยู่ 4 วิธีใหญ่คือ [4],[20]

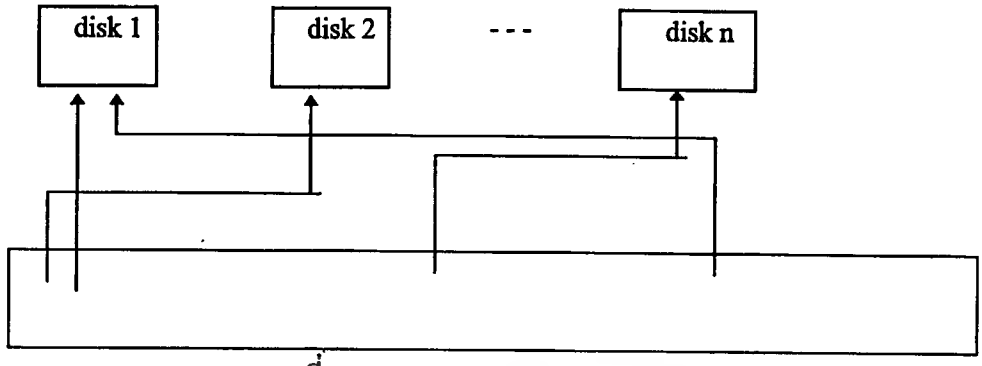
1 round-robin partition วิธีนี้จะใช้การคำนวณหาตำแหน่งของ tuple โดยจะ map tuple ที่อยู่ใน table ไปเก็บไว้ใน disk แต่ละตัวโดยใช้สูตร I mod n โดยที่ I = เลขที่ของ tuple ที่อยู่ใน table n = จำนวน disk ที่มีอยู่ในระบบดังภาพ



ภาพที่ 8 แสดง round-robin partition

2 hashing partition วิธีนี้จะนำข้อมูลในแต่ละ tuple จะต้องผ่าน hashing function ก่อนเพื่อที่จะได้รู้ว่าจะนำเอาข้อมูลในแต่ละ row ไปไว้ที่ disk ไหนดังภาพ

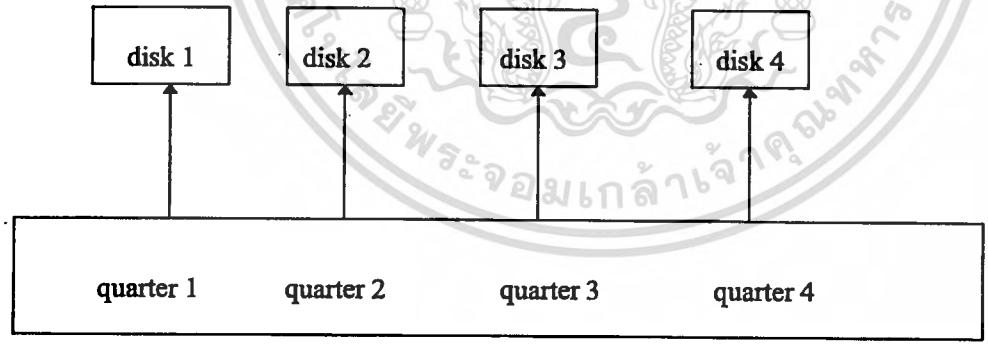
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 9 แสดง hashing partition

วิธีนี้เป็นการป้องกันไม่ให้ข้อมูลไปกระจุกตัวอยู่ใน disk ใด disk หนึ่งมากเกินไป จนอาจทำให้เกิดปัญหา data skew ขึ้นได้ เนื่องจากวิธีนี้สามารถกระจายข้อมูลไปยัง disk ที่ยังไม่มีข้อมูลได้ วิธีนี้นิยมใช้ในระบบ shared nothing system

3 range partition วิธีนี้จะเก็บข้อมูลในแต่ละ disk โดยการแบ่งตามขนาดที่ผู้ใช้เป็นผู้กำหนด โดยที่ผู้ใช้อาจจะแบ่งได้หลายอย่าง เช่น ในตารางการขาย ผู้ใช้อาจจะแบ่งข้อมูลตามช่วงระยะเวลาในการขาย โดยข้อมูลการขายในquarter 1 จะอยู่ใน disk ที่ 1 ข้อมูลการขายใน quarter ที่ 2 จะอยู่ใน disk ที่ 2 ข้อมูลการขายใน quarter ที่ 3 จะอยู่ใน disk ที่ 3 ข้อมูลการขายใน quarter ที่ 4 จะอยู่ใน disk ที่ 4 เป็นต้นดังภาพ

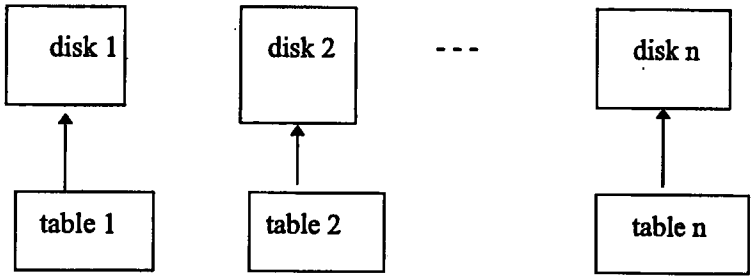


ภาพที่ 10 แสดง range partition

วิธีนี้อาจจะมีปัญหาที่ข้อมูลอาจจะไปกระจุกตัวอยู่ใน disk ใด disk หนึ่งมากเกินไป แต่ขณะที่ในอีก disk หนึ่ง ข้อมูลที่เก็บไว้มีอยู่น้อยซึ่งเราเรียกปัญหานี้ว่าการเกิด data skew

4 schema partition วิธีนี้จะเก็บข้อมูลโดยจะเก็บข้อมูลทั้งตารางเข้าไปไว้ใน disk เพียง disk เดียว เช่น ในตารางที่ 1 จะเก็บไว้ใน disk ตัวที่1 ,ในตารางที่ 2 จะเก็บไว้ใน disk ตัวที่ 2 ,ในตารางที่ 3 จะเก็บไว้ใน disk ตัวที่ 3,ในตารางที่ 4 จะเก็บไว้ใน disk ตัวที่ 4 เป็นต้น ดังภาพ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 11 แสดง schema partition

ใน Oracle 7 การทำ data partition ยังใช้ OS utility เป็นส่วนที่ใช้ในการจัดการกระจายข้อมูลไปยัง disk ต่างๆ ไม่ได้ใช้ DBMS เป็นผู้จัดการ โดยที่ Oracle จะทำงานโดยใช้วิธี range partition ในการทำงาน [7]

ข้อดีในการทำ data partition

1 ช่วยป้องกันความผิดพลาดในกรณีที่ disk ที่เก็บข้อมูลเสีย ข้อมูลในตารางก็จะเสียเพียงแค่ส่วนเดียวเท่านั้น ไม่มีผลกระทบกับส่วนอื่น

2 การดูแลรักษาจะทำได้สะดวกกว่า เช่น การทำ backup,การทำ recovery จะทำได้รวดเร็วมากกว่าการที่จะต้องทำทั้งตาราง

3 ประสิทธิภาพของระบบจะสูงขึ้น เนื่องจากข้อมูลกระจายออกไปยังหลายๆ disk ดังนั้นการนำข้อมูลจาก disk มายัง memory จะทำได้รวดเร็วมากกว่าการที่ข้อมูลอยู่ใน disk เพียง disk เดียว

3.2 Parallel Sorting

การจัดเรียงข้อมูล (sorting) เป็นขบวนการที่สำคัญอันหนึ่งในระบบฐานข้อมูล การจัดเรียงข้อมูลสามารถทำได้ทั้งจากน้อยไปมาก (ascending) หรือจากมากไปน้อย (descending) คำสั่งที่ใช้ในการ sort ข้อมูลมักจะเป็นคำสั่งที่ถูกใช้บ่อยๆในระบบฐานข้อมูล การ sort ข้อมูลจึงมีผลต่อประสิทธิภาพในการทำงานของระบบฐานข้อมูล การ sort ข้อมูลสามารถแบ่งได้เป็น 2 ประเภทใหญ่ๆคือ [14]

1 internal sort ข้อมูลที่จะ sort จะมีขนาดน้อยกว่า memory ดังนั้น ข้อมูลจะถูก sort ภายใน memory

2 external sort ข้อมูลที่จะ sort จะมีขนาดที่ใหญ่กว่า memory ดังนั้นข้อมูลจึงไม่สามารถใน memory ได้ข้อมูลจะต้อง sort ใน disk

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในระบบฐานข้อมูลแบบขนาน ข้อมูลที่เก็บไว้ในจะมีอยู่เป็นจำนวนมาก ดังนั้นข้อมูลที่จะ sort มักจะมีขนานที่ใหญ่กว่า memory ดังนั้นการ sort จึงต้องใช้วิธี external sort Algorithm ที่ใช้ใน Parallel External sorting มีอยู่ 2 ประเภท คือ

3.2.1 merge-based parallel sorting วิธีนี้จะแบ่งข้อมูลที่จะ sort ออกเป็น page แต่ละ page จะมีขนาดน้อยกว่า memory ที่มีอยู่ในระบบ ระบบจะนำข้อมูลในแต่ละpage มา sort บน memory จากนั้นก็จะนำข้อมูลที่ sort กลับไปไว้บน disk ดังเดิม การทำงานในแบบขนานก็คือ การที่ CPU ในแต่ละตัวจะนำข้อมูล 2 page มารวมกัน จากนั้นก็จะ sort ข้อมูลใน page ที่รวมกันนั้น โดยในการทำงานจะทำบน disk โดยที่ระบบจะรวมข้อมูลไปเรื่อยๆจนกว่าข้อมูลใน page จะรวมเป็น page เดียว โดยในการรวมข้อมูลให้เป็น page เดียวจะใช้ CPU เพียงตัวเดียวในการทำงาน วิธีนี้นิยมใช้ใน shared memory และ shared disk system

3.2.2 Partition-based parallel sorting วิธีนี้จะนำข้อมูลที่จะ sort มาแบ่งข้อมูลออกเป็นส่วนๆโดยใช้วิธี range-partition จากนั้นก็จะกระจายข้อมูลไปตาม CPU ที่มีอยู่ในระบบ การทำงานในแบบขนานคือ การที่ CPU จะ sort ข้อมูลที่กระจายออกไปอย่างมีอิสระและทำงานพร้อมๆกัน โดยในการ sort ข้อมูล ถ้าข้อมูลที่กระจายออกไปมีขนานน้อยกว่า memory ระบบก็จะนำข้อมูลมา sort บน memory แต่ถ้าข้อมูลมีขนาดมากกว่า memory ระบบก็จะ sort ข้อมูลบน disk เมื่อแต่ละ CPU ได้ sortข้อมูลแล้วก็จะนำข้อมูลที่ sort ในแต่ละส่วนมารวมกัน ณ. ที่เดิม เพื่อให้ CPU เพียง 1 ตัวมารวมรวมข้อมูลที่ถูก sort ในแต่ละส่วนเข้าด้วยกัน จากนั้นก็จะใช้ CPU เพียง 1 ตัว ในการ sort ข้อมูลทั้งหมด วิธีนี้นิยมใช้ใน shared nothing system

3.3 Parallel Processing of Join

การเชื่อมโยงตาราง 2 ตาราง เพื่อทีจะหาข้อมูลที่ต้องการหรือที่เรียกว่า การ join เป็นสิ่งที่มีความสำคัญมากในระบบฐานข้อมูล วิธีการที่ใช้สำหรับการ join มีอยู่ 3 วิธี

1 nested-loops join

2 sort-merge join

3 hash-based join

วิธีการ join ทั้ง 3 ประเภทนี้ สามารถใช้ในระบบฐานข้อมูลที่มี CPU เพียง 1 ตัวในการทำงานและระบบฐานข้อมูลที่มี CPU หลายๆตัวช่วยกันทำงานเราสามารถอธิบายการทำงานแต่ประเภทโดยแยกตามจำนวน CPU ที่มีอยู่ในระบบได้ดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.3.1 ระบบฐานข้อมูลที่ทำงานบน CPU เพียง 1 ตัว (Uniprocessor database systems)

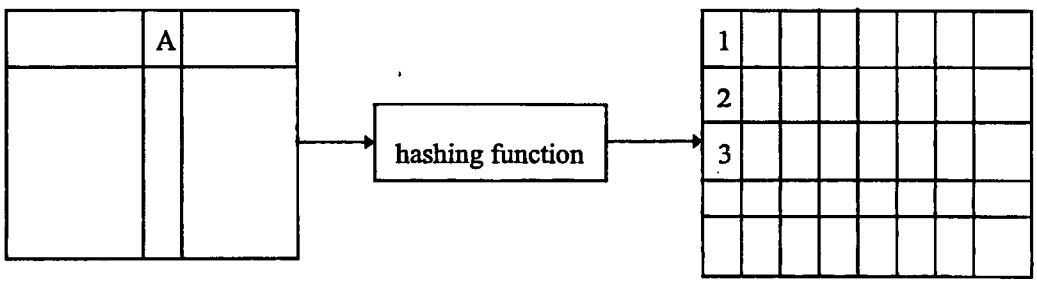
1 nested-loops join วิธีนี้เป็นการทำงานแบบพื้นฐาน โดยในการ join จะมีตารางอยู่ 2 ตาราง ตารางหนึ่งเรียกว่า outer table อีกตารางหนึ่งเรียกว่า inner table การ join ก็คือการนำเอาตาราง 2 ตาราง มาเชื่อมโยงกันผ่าน attribute ที่เหมือนกัน วิธีนี้จะนำข้อมูลในแต่ละ row ของ outer table มา join กับข้อมูลทุกๆ row ที่อยู่ใน inner table โดยที่ในการ join จะทำงานตั้งแต่ ข้อมูลใน row แรกไปจนถึงข้อมูลใน row สุดท้ายของ outer table

2 sort-merge join วิธีนี้จะมีการเรียงลำดับ (sorting) ข้อมูลที่อยู่ใน field ที่ต้องการจะ join ก่อนที่จะมีการ join โดยปกติการเรียงลำดับข้อมูลจะมีการเรียงลำดับจากน้อยไปมาก (ascending) หลังจากที่มีการเรียงลำดับแล้ว ถึงจะนำข้อมูลที่ได้จากการเรียงลำดับมา join โดยในการ join จะคล้ายกับวิธี nested-loops join แต่เนื่องจากข้อมูลมีการเรียงลำดับกันอยู่แล้ว วิธีนี้จะ join เร็วกว่าวิธี nested-loops join เพราะว่าวิธีนี้ไม่ต้องมีการ join ทั้งตารางถ้า field ที่ใช้ในการ join เป็น primary key ซึ่งเป็นค่าที่ไม่ซ้ำกัน ถ้าพบข้อมูลที่ตรงกันจะ join กันเพียง 1 ครั้ง แต่ถ้าหากว่า field ที่ใช้สำหรับการ join ไม่ได้เป็น primary key เราอาจจะต้องมีการ join มากกว่า 1 ครั้ง

3 hash-based join วิธีนี้เป็นการแก้ปัญหาของวิธี nested - loops join เนื่องจากว่าวิธี nested-loops join ต้องมีการค้นหาข้อมูลทุกๆ row ของ column ที่จะนำมา join กัน ถ้าหากว่าข้อมูลที่อยู่ใน column ที่ต้องการจะ join มีมาก แต่ว่าผลลัพธ์ที่ได้จากการ join มีน้อยทำให้เป็นการเสียเวลาในการ join มาก จึงได้มีการคิดวิธีการ hash-based join เพื่อที่จะหลีกเลี่ยงปัญหาที่เกิดขึ้น

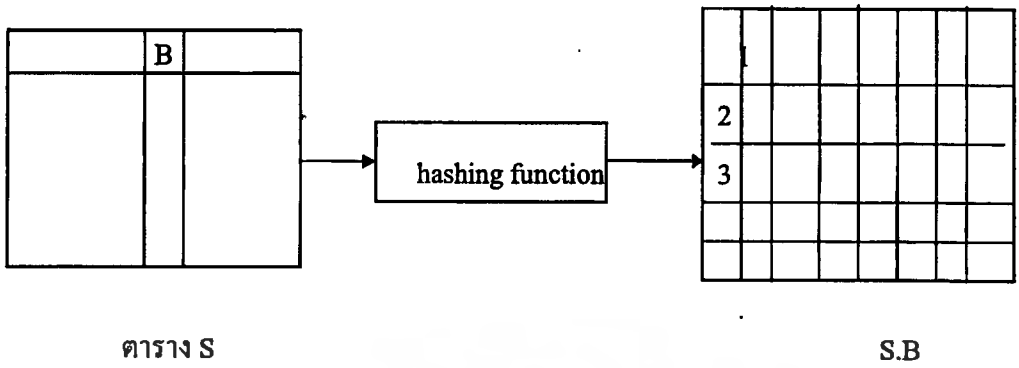
สมมติว่า ถ้าต้องการที่จะ join ตาราง R กับตาราง S เข้าด้วยกันโดยผ่านทาง attribute a กับ b วิธีการของ hash - based join จะมีวิธีการทำงานดังนี้

1 นำข้อมูลที่ต้องการจะ join ทั้ง 2 ตารางมาทำการ hashing นำข้อมูลที่ได้จากการ hashing แยกเป็นแต่ละ bucket



ตาราง R R.A

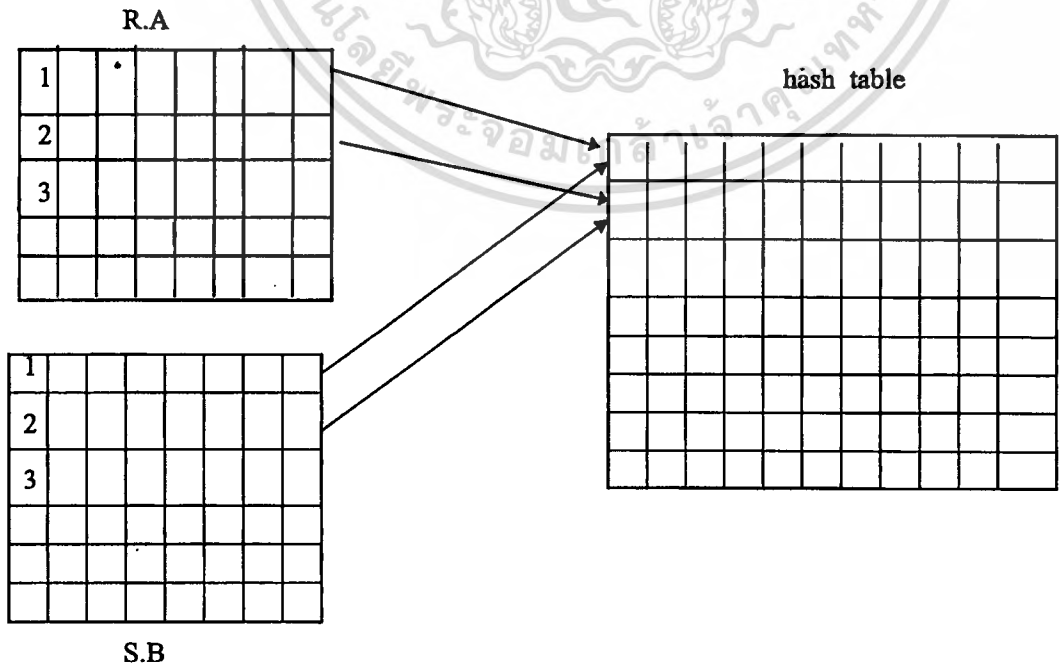
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 12 แสดง hashing function

hashing function ต้องใช้ function เดียวกัน จำนวน bucket ในแต่ละ bucket แต่ละ table จะต้องเท่ากัน เนื่องจากใช้ hashing function เหมือนกัน แต่ว่าจำนวน record ในแต่ละ bucket อาจจะมีค่าไม่เท่ากัน เนื่องจากข้อมูลที่เก็บในแต่ละ table อาจจะมีไม่เท่ากัน

2 ระบบจะสร้าง hash table ขึ้นมา จากนั้นจะนำเอาข้อมูลที่อยู่ใน bucket ที่ 1 ของ attribute R.A ไปไว้ใน hash table ใน bucket ที่ 1 จากนั้นจะนำเอาข้อมูลใน bucket ที่ 1 ของ attribute S.B ไปไว้ใน hash table ที่ bucket เดียวกัน จากนั้นจะนำเอาข้อมูลที่อยู่ใน bucket มา join เพื่อที่จะหาผลลัพธ์จากการ join โดยจะเริ่มตั้งแต่ bucket แรก ไปจนถึง bucket สุดท้าย โดยในการ join จะทำเฉพาะ ข้อมูลที่อยู่ในbucket เดียวกัน ไม่มีการ join ข้าม bucket



ภาพที่ 13 แสดง hash-based join

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การทำงานใน hash-based join มี 2 ขั้นตอนใหญ่ๆ คือ

1 partition phase ส่วนนี้จะเป็นการนำข้อมูลแต่ละ ตารางที่จะ join มาทำการ hashing โดยใช้ hashing function ตัวเดียวกันแยกเป็นแต่ละ bucket

2 join phase ส่วนนี้จะทำหน้าที่สร้าง hash table ขึ้นมา จากนั้นจะนำเอาข้อมูลที่อยู่ใน bucket มาเก็บไว้ใน hash table เพื่อที่จะนำข้อมูลที่เก็บไว้มา join

ข้อที่ควรระวังในการใช้วิธี hash-based join ก็คือ การเลือก hashing function ให้สามารถคลอบคลุมข้อมูลที่อยู่ในตารางได้ทั้งหมด และการสร้างhash table ที่ขนาดใหญ่เพียงพอที่จะนำเอาข้อมูลที่อยู่ใน bucket เข้าไปเก็บไว้ใน hash table เพื่อที่จะหาผลลัพธ์จากการ join

3.3.2 ระบบฐานข้อมูลที่ทำงานบน CPU หลายๆ ตัว (Parallelprocessor database systems)

ในการทำงานแบบ Parallel ระบบจะทำงานบนเครื่องคอมพิวเตอร์ที่มี CPU มากกว่า 1 ตัวขึ้นไปช่วยกันทำงานจะเป็นการทำงานที่เรียกว่า parallel processing ซึ่งสามารถแบ่งการทำงานออกเป็นงานย่อยๆ ได้ 3 ขั้นตอน [4]

1 แบ่งงานที่ต้องทำในการ join ออกเป็นงานย่อยๆหลายๆงาน

2 ในแต่ละงานจะถูกทำงานโดย CPU แต่ละตัว

3 นำผลลัพธ์ที่ได้จาก CPU แต่ละตัวมารวมกันให้เป็นผลลัพธ์ของการทำงานทั้งหมด

ระบบสามารถแสดงการทำงานในแต่ละประเภทได้ดังนี้

1 Parallel nested-loops join สามารถแบ่งการทำงานได้เป็น 2 ประเภทตามลักษณะของสถาปัตยกรรมคอมพิวเตอร์ คือ

1.1 shared memory และ shared disk system process ที่มีในระบบจะแบ่งข้อมูลในตาราง outer table ออกเป็นส่วนๆ แต่ละส่วนจะ join กับข้อมูลทั้งหมดที่อยู่ใน inner table ซึ่งในแต่ละ process จะทำงานพร้อมๆกันเป็นการทำงานที่เรียกว่า parallel processing

1.2 shared nothing system เนื่องจากข้อมูลจะมีการกระจายข้อมูล (data partition) ไปยัง disk ที่มีอยู่ในระบบและระบบจะมี CPU, memory และ disk เป็นของตนเอง ดังนั้นหลังจากที่มีการscanข้อมูลจาก disk แล้วจะนำข้อมูลใน inner table มารวมกัน จากนั้นจะนำข้อมูลในinner table ไปยัง disk ที่มีข้อมูลใน outer table ที่กระจายบน disk เพื่อให้ CPU ใน disk ได้ join ข้อมูลในแต่ละ CPU ก่อนที่จะนำเอาผลลัพธ์ที่ได้ในแต่ละ CPU มารวมกันให้เป็นผลลัพธ์ในการ join ทั้งหมด

2 Parallel sort-merge join จะแบ่งการทำงานได้เป็น 2 ประเภทคือ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.1 shared memory และ shared disk system ในแต่ละ user process จะนำข้อมูลที่จะ join มา sort เรียงลำดับข้อมูลทั้ง 2 ตารางจากนั้นจะนำข้อมูลที่ได้จากการ sort มา merge เพื่อที่จะหาผลลัพธ์จากการ join

2.2 shared nothing system หลังจากที่แต่ละ node scan ข้อมูลจาก disk แล้ว แต่ละ node จะ sort ข้อมูลในแต่ละ disk ทั้ง 2 ตารางที่จะ join นำข้อมูลในตารางหนึ่งเข้าไปใน disk ที่เก็บข้อมูลอีกตารางหนึ่งโดยจะต้องแบ่งข้อมูลเท่าๆกันในแต่ละ disk จากนั้นจะ merge ข้อมูลเพื่อที่จะหาผลลัพธ์จากการ join ในแต่ละ disk ก่อนที่จะนำผลที่ได้จากการ join ในแต่ละ instance มารวมกันเพื่อที่จะเป็นผลลัพธ์จากการ join ทั้งหมด

3 Parallel hash-based join วิธี hash-based join เป็นวิธีที่เหมาะสมในการทำงานแบบ parallel ทั้ง partition phase และ join phase สามารถทำงานในแบบขนานได้ Parallel hash-based join สามารถแบ่งการทำงานได้เป็น 2 ประเภทตามลักษณะของสถาปัตยกรรมคอมพิวเตอร์

3.1 shared memory และ shared disk system ในส่วนของ partition phase server process จะนำข้อมูลในตารางที่จะ join ทั้ง 2 ตารางเข้ามาทำ hashing function แยกเป็นแต่ละ bucket ในขณะเดียวกันระบบจะทำงานในส่วน join phase โดย process จะสร้าง hash table และนำข้อมูลจาก bucket ที่ได้จาก hashing function ในแต่ละ bucket มา join โดยที่ในส่วน partition phase และ join phase จะทำพร้อมๆกันทั้ง 2 phase ซึ่งจะเป็นการทำงานในแบบ parallel processing

3.2 shared nothing system ในรูปแบบนี้แต่ละ CPU จะมี disk และ memory เป็นของตนเองข้อมูลจะมีการกระจายไปยัง disk ต่างๆที่มีในระบบ ดังนั้นหลังจากที่มีการ scan ข้อมูลในแต่ละ CPU ข้อมูลในตารางใดตารางหนึ่งจะรวมกันกลายเป็นตารางเดียวกระจายไปยัง disk ที่มีข้อมูลในอีกตารางหนึ่งอยู่ การทำ partition phase CPU แต่ละตัวจะทำ hashing function ทั้ง 2 ตารางแยกเป็นแต่ละ bucket จากนั้นจะนำข้อมูลในแต่ละ bucket ทั้ง 2 ตารางมาทำในส่วน join phase โดยระบบจะสร้าง hash table นำข้อมูลในแต่ละ bucket มาเก็บไว้ในแต่ละ hash table ก่อนที่จะนำข้อมูลใน hash table มา join เพื่อที่จะหาผลลัพธ์ในแต่ละ CPU ก่อนที่จะนำเอาผลลัพธ์ที่ได้ในแต่ละ CPU มารวมกันกลายเป็นผลลัพธ์ในการ join ทั้งหมด

ข้อที่ต้องพิจารณาในการใช้วิธี parallel hash-based join

1 การเลือก hashing function ให้สามารถที่จะคลอบคลุมจำนวนข้อมูลทั้งหมดที่เก็บไว้ในtable

2 ขนาดของ hash table ต้องมีขนาดใหญ่เพียงพอที่จะนำเอาข้อมูลใน bucket ทั้งหมด เข้ามาใน hash table ได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.3.3 สรุป

Algorithm ทั้ง 3 ประเภทที่ได้กล่าวมาแล้วล้วนแล้วแต่มีประสิทธิภาพในการทำงาน การที่จะเลือกใช้วิธีไหนขึ้นอยู่กับขนาดข้อมูลและการกระจายข้อมูลในแต่ละ disk โดยปกติ วิธี hash-based join เป็นวิธีที่มีประสิทธิภาพในการทำงานมากที่สุดทั้ง 3 วิธีที่กล่าวมา แต่ว่าวิธี nested - loops join จะเหมาะสมในการ join ที่ขนาดข้อมูลทั้ง 2 ตาราง ไม่แตกต่างกันมากสำหรับวิธี sort-merge join จะเหมาะสมในกรณีที่ทั้ง 2 table มีขนาดข้อมูลเท่าๆกัน ข้อเสียของวิธี Parallel hash-based join ก็คือถ้าเกิดมีปัญหาในเรื่อง data skew จะทำให้ประสิทธิภาพตกลงไปมากในทางตรงกันข้ามวิธี Parallel nested-loops join จะไม่มีผลกระทบที่เกิดมาจาก data skew เช่นเดียวกับวิธี Parallel sort-merge join ซึ่งในวิธี sort-merge join จะมีข้อดีตรงที่ผลลัพธ์ที่ได้จากการ join จะมีการเรียงลำดับข้อมูลด้วย

3.4 Data Skew and Load Balancing

Data skew [4][13] เป็นปัญหาที่เกิดจากการแบ่งข้อมูล (data partition) ในตารางกระจายออกไปยัง disk หลายๆตัวที่มีอยู่ในระบบ แต่ว่าข้อมูลกลับไปรวมอยู่ใน disk ใด disk หนึ่งโดยเฉพาะ ไม่ได้มีการกระจายข้อมูลไปยัง disk ทุกๆตัวอย่างเท่าเทียมกัน การเกิด data skew จะทำให้ประสิทธิภาพในการ join ลดลงเนื่องจาก CPU ในระบบจะมีการใช้งานที่ไม่เท่ากัน บาง CPU จะมีการใช้งานมากแต่ในบาง CPUจะมีการใช้น้อย ปัญหา data skew จะเป็นปัญหาที่สำคัญใน shared nothing system เนื่องจาก CPU ในระบบจะมี memory และ disk เป็นของตนเอง ดังนั้นการที่ข้อมูลกระจายอย่างไม่เท่ากันภายใน disk ที่มีอยู่ในระบบจะทำให้ประสิทธิภาพในการ join ของระบบลดลงไป แต่ถ้าเป็นใน shared memory หรือ shared disk system ระบบจะไม่สนใจการเกิด data skew เนื่องจาก CPU ในระบบสามารถติดต่อ disk ได้ทุกๆ disk ที่มีในระบบ ดังนั้นใน shared memory system จึงไม่สนใจปัญหา data skew ถึงแม้ว่าข้อมูลจะกระจายออกไปยัง disk ในระบบอย่างไม่เท่ากันก็ตาม การทำ load balancing จึงเป็นการแก้ปัญหาการเกิด data skew โดยการทำให้การ join ข้าม CPU ในระบบเกิดความสมดุลกันหรือทำให้ CPUในระบบทำงานได้อย่างเท่าเทียมกัน การทำ load balancing สำหรับ parallel join algorithim ใน shared nothing system สามารถแบ่งได้เป็น 3 ขั้นตอน ดังนี้

1 task generation ขั้นตอนแรกเป็นการสร้าง task ขึ้นเพื่อที่จะ join ข้อมูลในระบบจะมีการกระจายข้อมูลไปยัง disk ต่างๆ ที่มีในระบบ การสร้าง task จะเป็นการนำข้อมูลที่กระจายอยู่ในแต่ละ disk มา join เช่น ใน parallel nest-loops join ข้อมูลจะเป็นการ join ระหว่าง outer table กับ inner table โดย outer table จะกระจายออกเป็นส่วนๆในแต่ละ CPU และในinner table จะกระจาย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ไปใน disk ที่มี outer table อยู่ จำนวนtask จะเท่ากับ จำนวน ข้อมูลในแต่ละส่วนของ outer table กับข้อมูลทั้งหมดที่อยู่ใน inner table จำนวน task จะกระจายไปในแต่ละ CPU ที่มีอยู่ในระบบ

2 task allocation/load balancing phase หลังจากที่มีการสร้าง task เกิดขึ้น ระบบจะมีการนำtask ไปให้แต่ละ CPU เพื่อให้ CPUได้ทำงาน ในขั้นตอนนี้สามารถแบ่งการทำงานได้เป็น 2 วิธี คือ

2.1 กระจาย task ไปในแต่ละ CPUโดยที่ไม่สนใจว่าแต่ละ CPU จะมีการทำงานเป็นอย่างไร ในวิธีนี้จะไม่มีการทำ load balancing เมื่อกำหนด task แล้วจะนำ task ไปไว้ในแต่ละ CPU เลย วิธีนี้จะทำให้ประสิทธิภาพของระบบลดลงถ้าข้อมูลที่กระจายในแต่ละ CPU ไม่เท่ากัน จะทำให้ CPU แต่ละตัวทำงานไม่เท่ากันซึ่งจะทำให้ประสิทธิภาพของระบบลดลง

2.2 กระจาย task ไปในแต่ละ CPU โดยต้องมีการทำ load balancing ระบบจะมีการทำ load balancing ถ้าหากพบว่าใน CPUใดมี task อยู่มาก ระบบจะมีการนำ task ไปให้กับอีก CPU หนึ่งในระบบ เป็นการกระจายการทำงานในแต่ละ CPU วิธีนี้เป็นการทำงานในแบบ static load balancing

3 task execution ขั้นตอนนี้แต่ละ CPU จะนำข้อมูลในแต่ละ task มา join ในการนำข้อมูลมา join จะมีขั้นตอนการทำงานอยู่ 2 ขั้นตอน

3.1 Workload ขั้นตอนนี้สามารถแบ่งการทำงานได้เป็น 2 วิธี

1 ไม่มีการกระจาย task ในระหว่างการทำงาน ในขณะที่มีการทำงานจะไม่มีการกระจาย task ไปยัง CPU อื่นถึงแม้ว่าในระบบจะเกิดการไม่ balance ขึ้นก็ตาม

2 จะมีการกระจาย task ในระหว่างการทำงาน ในขณะที่มีการ join ตารางถ้าระบบตรวจสอบว่าเกิดการไม่ balanced ที่ CPUไหน ระบบสามารถกระจาย task จาก CPU ที่ overload ไปให้ CPU ที่ไม่เกิดการ overload วิธีนี้จะเป็นการทำงานในแบบ dynamics load balancing

3.2 Local join methods แต่ละ CPU จะนำข้อมูลในแต่ละ disk ที่มีอยู่ มา join โดยที่การ join ในระบบจะมีอยู่ 3 วิธี คือ nested-loops join, sort-merge join และ hash-based join

3.4.1 Static and dynamics load balancing

การทำload balancing [4] เป็นการทำให้การ join ตาราง 2 ตารางที่ข้อมูลในตารางกระจายอยู่ใน disk หลายๆตัวเกิดความสมดุล การที่ระบบไม่สมดุลเนื่องมาจากข้อมูลในตารางกระจายอยู่ใน disk แต่ละตัวไม่เท่ากันทำให้disk ใด disk หนึ่งทำงานมากเกินไป ในขณะที่ disk ที่มีข้อมูลอยู่น้อย การ join ตาราง ใน disk นั้นก็จะน้อยตามไปด้วย การทำ load balancing มีวิธีการอยู่ 2 วิธี

1 Static Load balancing วิธีนี้จะเช็คจำนวนข้อมูลในแต่ละ node ที่จะนำมา join ว่า ข้อมูลที่กระจายใน nodeใด node หนึ่งมีจำนวนมากเกินไปทำให้ node ต้องทำงานกับข้อมูลมากเกินไปหรือ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ไม่ ถ้าพบว่ามีข้อมูลอยู่มากเกินไป ระบบจะต้องมีการกระจายข้อมูลที่กระจุกตัวใน node ใด node หนึ่งออกไปยัง node อื่นๆเพื่อที่จะทำให้จำนวนข้อมูลในแต่ละ node เกิดความสมดุลก่อนที่จะนำเอาข้อมูลในแต่ละ node มา join

ขั้นตอนในการทำงานแบบ static load balancing มีดังนี้

1.1 ในแต่ละ task ที่อยู่ในแต่ละ node จะมีการนำข้อมูลที่อยู่ใน disk มาเก็บไว้ใน memory เพื่อที่จะ join

1.2 ในแต่ละ node จะคำนวณว่ามี task อยู่จำนวนเท่าไรในระบบ และประมาณเวลาที่แต่ละ task จะใช้ในการทำงาน ก่อนที่ในแต่ละ node จะส่งผลเวลาที่ใช้ในการทำงานในแต่ละ task มาที่ส่วนของ controller เพื่อที่จะคำนวณหาค่าเฉลี่ยในการทำงานในระบบทั้งหมดก่อนที่จะส่งผลการคำนวณไปให้ทุกๆ node ที่มีอยู่ในระบบ

1.3 ขั้นตอนนี้จะเป็นการตรวจสอบว่าข้อมูลที่กระจายอยู่ในแต่ละ node ใด node หนึ่ง มีข้อมูลอยู่มากเกินไปจนเกิดการ overload หรือไม่ การตรวจสอบจะเช็คจากสมการ

$$\sum_{j=1}^{k} \text{Cost } j <= \text{Cost avg} \text{ and } \sum_{j=1}^{k+1} \text{Cost } j > \text{Cost avg}$$

k เป็นจำนวน task ที่น้อยที่สุดที่จะทำให้สมการนี้เป็นจริง

Cost j เป็นเวลาที่จะ process task j

Cost avg เป็นเวลาเฉลี่ยที่ใช้ในการทำงาน

การเช็คว่า node ไหนจะเกิดการ overload หรือไม่ สามารถตรวจสอบได้จากค่า k

ถ้า k น้อยกว่าค่าที่กำหนดไว้ แสดงว่า node นั้นเกิดการ overload เกิดขึ้น

ถ้า k มากกว่าค่าที่กำหนดไว้ แสดงว่า node นั้น underload

<u>ตัวอย่าง</u> สมมติว่ามีจำนวน task = 16 task แต่เมื่อเราใช้สมการมาตรวจสอบ ปรากฏว่าเพียงแค่ 10 task ก็จะใช้เวลาเท่ากับเวลาเฉลี่ยใน การทำงาน แสดงว่าเหลืออีก 6 task ที่ยังไม่ได้ทำงาน แสดงว่า node นั้นจะเกิดการ overload เกิดขึ้น

1.4 ถ้า node ไหนเกิดการ overload ระบบจะมีการทำ load balancing ก่อน ถึงจะให้มีการ join ข้อมูลในแต่ละ disk ได้

การแก้ปัญหาโดยใช้วิธี static load balancing หลังจากที่พบว่า processor ใด เกิดการ overload เราจะนำข้อมุลที่อยู่ใน node กระจายไปยัง node อื่นที่ยัง underload อยู่เพื่อเป็นการลดเวลาการทำงานที่เกิดจาก disk ที่ทำให้เกิดการ overload ขึ้น task ที่กระจายออกไปจะถูก process

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โดย node ตัวอื่น ทำให้ node ที่มี task อยู่มากจะใช้เวลาทำงานน้อยลงทำให้แต่ละ node ทำงานได้เท่าๆกัน

2 Dynamics Load balancing การแก้ปัญหา การเกิด overload โดยใช้วิธี Dynamics load balancing จะแตกต่างจากวิธี static load balancing ตรงที่การกระจาย task ใน node ที่เกิดการ overload ถ้าเป็นวิธี static จะมีการกระจาย task ก่อนถึงจะทำการ join ตาราง แต่ถ้าเป็นวิธี Dynamics load balancing จะกระจาย task ในระหว่างการ join

ขั้นตอนในการทำงานของวิธี Dynamics Load Balancing มีดังนี้

2.1 node ไหนที่มีข้อมูลที่ต้องการ join อยู่มาก จะเรียกว่า donor node node ไหนที่มีข้อมูลที่จะ join อยู่น้อยจะเรียกว่า idle node node ไหนที่มีข้อมูลที่จะjoin น้อยจึงสามารถที่จะทำงานเสร็จเร็วกว่า donor node เมื่อ idle node ทำงานเสร็จแล้วก็จะส่ง message มาที่ coordinator เพื่อที่แจ้งว่าในขณะนี้ node ได้ทำงานเสร็จแล้ว

2.2 coordinator จะรับ messages จาก idle node มาเข้า queue เพื่อที่จะตรวจสอบดูว่ามี node ไหนบ้างที่ยังทำงานไม่เสร็จ

2.3 แต่ละ node จะคำนวณหา task ที่ตนเองได้ทำอยู่เพื่อที่จะตรวจสอบว่ามี task ไหนบ้างที่ยังไม่ได้ join จะส่งผลการตรวจสอบไปที่ coordinator

2.4 หลังจากส่งผลมาแล้ว coordinator จะกำหนดว่า node ไหนจะเป็น donor node จากนั้นจะส่ง messages ที่รับมาจาก idle node ไปที่ donor node เพื่อที่จะให้ donor node ส่ง task ไปให้ idle node

2.5 donor node จะกำหนดว่าtask ไหนจะส่งไปให้ idle node จากนั้นก็จะส่ง task ไปให้ idle node เพื่อที่จะ process task นั้น

ในวิธีการของ dynamics load balancing ต้องมีประเด็นที่ต้องระวังอยู่หลายประเด็น

1 จำนวนของ task ที่จะ load จาก donor node ไปยัง idle node

2 การเปลี่ยน task จาก donor nodeไปยัง idle node จะทำให้ประสิทธิภาพของระบบต้องลดลงเนื่องจากเป็นการส่งข้อมูลข้ามระบบ network

3.4.2 สรุป

ทั้ง static และ dynamics load balancing เป็นวิธีการในการแก้ปัญหาการเกิด data skew ในระบบ parallel database โดยในวิธี static จะเป็นการกระจายข้อมูลไปยัง node ต่างๆ ก่อนที่จะมีการ join วิธีการนี้จะเหมาะสมสำหรับข้อมูลที่เป็น high data skew คือมีความแตกต่างระหว่างข้อมูลในแต่ละ diskมากๆ แต่ถ้าข้อมูลเกิดเป็น low data skew วิธีstatic load balancing จะให้ประสิทธิภาพที่

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ไม่ดี แต่ถ้าเป็นวิธี dynamics load balancing จะมีการกระจายข้อมูลในระหว่างที่เกิดการ join ซึ่งจะให้ประสิทธิภาพที่ดีกว่าแบบ static แต่เนื่องจากว่าวิธีนี้จะมีการกระจายข้อมูลในระหว่างที่เกิดการ join ดังนั้นการทำงานก็จะยุ่งยากมากกว่าแบบ static

3.5 Parallel Query Optimization

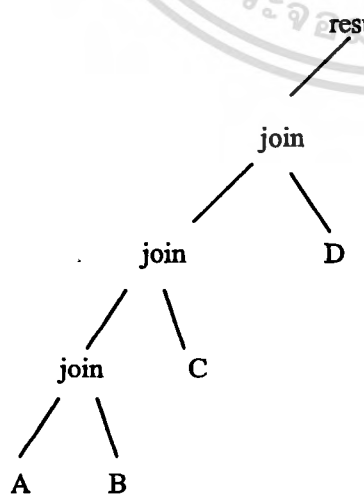
การ join จะหมายถึง การนำตาราง 2 ตาราง มาเชื่อมโยงกันโดยผ่าน attribute เดียวกัน แต่ถ้าเป็น query จะหมายถึง การนำตารางหลายๆตารางมาเชื่อมโยงกันเพื่อที่จะหาข้อมูลที่ต้องการ query จึงเป็นสิ่งที่มีความสำคัญมากในระบบฐานข้อมูล การทำ query optimization จึงเป็นการค้นหาวิธีการที่ดีที่สุดในการนำตารางหลายๆตารางมาเชื่อมโยงกัน โดยใช้เวลาน้อยที่สุดและได้ผลลัพธ์ออกมามากที่สุด การทำ query optimization มีหลักการที่จะต้องคำนึงอยู่ 2 ประการ [4]

1 การเลือกวิธีการ join ที่เหมาะสมในการ join แต่ละ ตาราง

2 ลำดับของตารางที่จะนำมา join

วิธีการ join ได้กล่าวถึงไปแล้วในเรื่องของการ join ว่ามีอยู่ 3 วิธี คือ nested-loops ,sort-merge และ hash-based join ลำดับในการ join หมายถึง การจัดลำดับของตารางที่จะนำมา join ก่อน-หลัง เพื่อที่จะทำให้ระบบมีประสิทธิภาพมากที่สุด ลำดับในการ join มีอยู่ 3 ประเภท

1 left-deep tree วิธีนี้จะ join ตารางที่อยู่ทางซ้ายมือสุดก่อนแล้วถึงค่อยนำผลลัพธ์ที่ได้มา join กับตารางที่อยู่ทางขวามือ เช่น ถ้าต้องการ join ตาราง A,B,C,D ในแบบ left-deep tree จะสามารถแสดงผลการทำงานได้ดังนี้

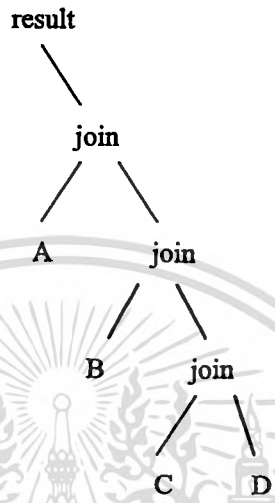


ภาพที่ 14 แสดง left-deep tree

ระบบแสดงลำดับการทำงานได้เป็น (((A*B)*C)*D)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

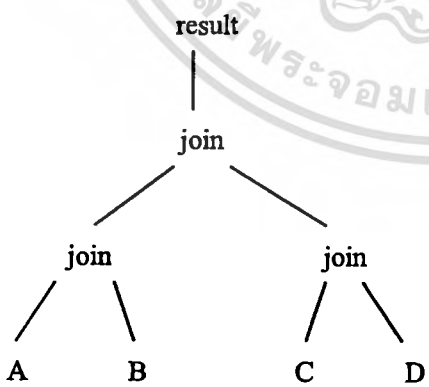
2 right-deep tree วิธีนี้จะ join ตารางที่อยู่ทางขวามือก่อนแล้วถึงจะนำเอาผลลัพธ์ที่ได้มา join กับตารางที่อยู่ทางด้านซ้ายมือ เช่น การ join ตาราง A,B,C,D แบบ right-deep tree จะสามารถแสดงผลการทำงานได้ดังนี้



ภาพที่ 15 แสดง right-deep tree

ระบบแสดงลำดับการทำงานได้เป็น (A*(B*(C*D)))

3 bushy tree วิธีนี้จะ join ตารางทั้งหมดพร้อมๆกัน แต่ว่าจะ join กันคราวละ 2 ตาราง จากนั้นจะนำผลที่ได้จากการ join มาหาผลลัพธ์อีกครั้ง เช่น การ join A,B,C,D แบบ bushy tree จะสามารถแสดงผลการทำงานได้ดังนี้



ภาพที่ 16 แสดง bushy tree

ระบบแสดง ลำดับการทำงานได้เป็น ((A*B)*(C*D))

ในการทำ query optimization สามารถแบ่งได้เป็น 2 ประเภทใหญ่ๆตามลักษณะของ CPU

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1 Uniprocessor optimization techniques เป็นการทำ query optimization โดยจะทำงานบนคอมพิวเตอร์ที่มี CPU อยู่เพียง 1 ตัว algorithm ที่ใช้ในการทำงานจะคำนวณหาวิธีการjoin ที่เหมาะสมและลำดับในการ join ที่จะทำให้ได้ประสิทธิภาพในการ query ได้สูงสุด algorithm ที่ใช้ในการทำงานก็จะมี exhaustive, dynamics programming with branch and bound, polynomial heuristics และ combinatorial algorithm เป็นต้น

2 Parallelprocessor optimization techniques ในการทำ Parallel optimization นอกจากจะต้องคำนึงถึงหลักการ 2 ประการแล้วสิ่งที่ยังต้องคำนึงถึงคือเรื่องของ resource allocation เนื่องจากว่าใน Parallel database ที่มีจำนวน CPU, memory และ disks อยู่หลายๆตัวในระบบ ดังนั้นการที่จัดสรรทรัพยากรที่มีอยู่ในระบบให้สามารถทำงานกับคำสั่ง query หลายๆคำสั่ง โดยใช้ประสิทธิภาพที่มีอยู่ในระบบได้อย่างเต็มที่จึงเป็นเรื่องที่มีความสำคัญมาก การทำ Parallel optimization แบ่งออกได้เป็น 2 ประเภท

2.1 one-phase optimization

2.2 two-phase optimization

ภายใน two-phase optimization จะแบ่งการทำงานออกเป็น 2 phase คือ plan generation และ resource scheduling ภายใน plan generation จะมีการกำหนดแผนการในการ query โดยการกำหนดไว้ว่าใน query ที่มีหลายๆตาราง จะใช้วิธีการ join อย่างไรและจะใช้ลำดับในการ joinอย่างไร แต่จะไม่มีการพิจารณาถึงการใช้ resource ภายในระบบ ส่วน resource scheduling จะเป็นการกำหนดการใช้ resource ภายในระบบให้เหมาะสมกับแผนที่ได้กำหนดไว้ใน plan generation

โดยปกติในระบบ Parallel database จะใช้ two-phase optimization ในการทำงานแต่ถ้าเป็น one-phase optimization จะมีแต่ส่วนที่เป็น plan generation เพียงส่วนเดียว จะไม่มีส่วนที่เป็น resource scheduling

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4

การทำงานในระบบฐานข้อมูลแบบขนาน

การทำงานในระบบฐานข้อมูลแบบขนาน สามารถแยกการทำงานได้เป็น 3 ประเภทใหญ่ๆ ตามลักษณะของ hardware [6] :

1 ระบบฐานข้อมูลที่ทำงานบน 1 instance และ 1 DBMS (shared memory system)

2 ระบบฐานข้อมูลที่ทำงานบนหลายๆ instance แต่ว่าอยู่บน 1 DBMS (shared disk system)

3 ระบบฐานข้อมูลที่ทำงานบนหลายๆ instance และหลายๆ DBMS (shared nothing system)

ในการทำงานแบบ shared memory system และ shared disk system จะมีการ query เป็นแบบ interquery parallelism โดยใน shared memory system ระบบสามารถรองรับผู้ใช้ให้เข้ามาทำงานได้หลายๆคนพร้อมๆกัน โดยระบบจะทำการค้นหาข้อมูลโดยใช้ CPU แต่ละตัวทำงานกับคำสั่ง query 1 คำสั่ง โดยภายในระบบจะนำคำสั่ง query ที่ผู้ใช้เข้ามาค้นหาข้อมูลนำมาเก็บไว้ใน memory โดยจะมีการเข้า queue ในการทำงาน จากนั้น CPU ที่มีอยู่ในระบบจะนำคำสั่ง query เข้ามาทำงานภายใน CPU โดยที่แต่ละ query จะถูกทำงานโดย CPU แต่ละตัว ซึ่งในแต่ละ query จะเป็นการทำงานที่เรียกว่า parallel transaction processing แต่ถ้าเป็นแบบ shared disk system จะเป็นการขยายการทำงานในแบบ shared memory system โดยในระบบจะมีหลายๆ instance แต่ละ instance จะมี CPU และ memory เป็นของตนเอง ดังนั้นระบบสามารถรองรับผู้ใช้ได้มากขึ้นและสามารถทำงานได้อย่างรวดเร็วยิ่งขึ้น แต่ว่าในระบบนี้จะต้องระมัดระวังปัญหาการใช้ข้อมูลร่วมกัน (concurrency control) ที่เกิดจากการที่แต่ละ instance เข้าไปแก้ไขข้อมูลเดียวกันในตารางเดียวกันพร้อมๆกัน จะทำให้ข้อมูลมีปัญหาในเรื่องของความถูกต้องได้ ใน shared nothing system จะมีการทำงานทั้งแบบ interquery และ intraquery parallelism โดยที่ระบบสามารถที่จะกระจายคำสั่ง 1 คำสั่งไปให้ node ทุกๆ node ช่วยกันทำงานกับคำสั่ง 1 คำสั่งพร้อมๆกัน จากนั้นเมื่อคำสั่งกระจายไปยังแต่ละnode ถ้าหากว่าในแต่ละ node เป็นการนำ shared memory system หลายๆ node มาเชื่อมโยงกัน ในแต่ละ node จะมี CPU มากกว่า 1 ตัวขึ้นไป ดังนั้นระบบก็ยังสามารถที่จะใช้ CPU หลายๆตัวช่วยกันทำงานกับคำสั่ง 1 คำสั่งได้ เราสามารถแบ่งประเภทการทำงานในระบบฐานข้อมูลแบบขนานออกเป็น 2 ประเภทคือ

1 Shared memory system

2 Shared nothing system

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.1 Shared Memory Systems

หลังจากที่ CPU แต่ละตัวรับคำสั่ง query จาก user เข้าไปแล้ว ภายใน shared memory system จะมีการทำงานเป็นแบบ parallel processing คือ ระบบจะมีการแบ่งงาน 1 งานออกเป็นงานย่อยๆหลายๆงาน และงานย่อยๆเหล่านั้นจะถูกทำงานโดย CPU หลายๆตัวช่วยกันทำงานพร้อมๆกัน ทำให้ระบบสามารถทำงานได้อย่างรวดเร็วยิ่งขึ้น เป็นการทำงานที่คล้ายกับการทำ multitasking ระบบสามารถแบ่งการทำงานในแบบ parallel processing ออกได้เป็น 2 ประเภทใหญ่ๆ คือ [16]

4.1.1 throughput parallelism

วิธีนี้หลังจากที่รู้ว่าคำสั่งย่อยๆคำสั่งไหนใน query ที่สามารถทำงานในแบบขนานได้ ระบบจะมี process หลายๆ process มาจัดการกับคำสั่งนั้น โดยที่ CPU ในระบบที่มีอยู่หลายๆตัวนั้น จะจัดการกับ process เหล่านั้น พร้อมๆกัน โดย OS จะเป็นส่วนที่จะกระจาย process ไปให้ CPU ช่วยกันทำงานกับ process เหล่านั้น รูปแบบนี้จะเป็นการทำงานที่เรียกว่า multiprocessing ใน Oracle จะใช้วิธีนี้ในการทำงาน

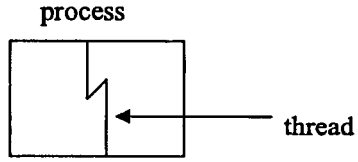
4.1.2 latency parallelism

วิธีนี้เป็นการทำงานที่เรียกว่า multithreading โดยที่ thread จะหมายถึง subprocess หรือคำสั่งย่อยๆที่สามารถทำงานได้อย่างเป็นอิสระต่อกันไม่ขึ้นต่อกันในการทำงาน

<u>ตัวอย่าง</u> คำสั่ง SQL 1 คำสั่ง

```
select *
from A,B
where A.X=B.X
```

จากคำสั่ง SQL นี้จะเป็นคำสั่ง query 1 query หรือ 1 process โดยที่คำสั่ง 1 query จะมีคำสั่งย่อยๆอยู่เพียง 1 thread ซึ่งสามารถแสดงการทำงานได้ดังภาพ



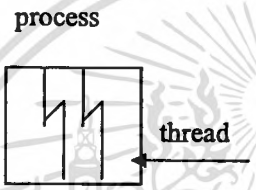
ภาพที่ 17 แสดง single- threaded

จากภาพเป็นการแสดงการทำงานของ process ที่มี thread เพียง 1 thread ในปกติคำสั่ง query ที่เข้ามาทำงานภายในระบบใน 1 process หรือ 1 query จะมี thread อยู่หลายๆ thread ทำงานอยู่ใน process เช่น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
select   hobby, pres_name
   from   pres_hobby
   where pres_name  in
   ( select   pres_name
     from    president
     where  yrs_serv > = 8)
```

จากคำสั่ง query นี้ สามารถที่จะแบ่งออกเป็นคำสั่งย่อยๆที่เป็นอิสระต่อกันในการทำงานได้ 2 thread ดังภาพ



ภาพที่ 18 แสดง multithreaded

การทำงานแบบขนานในแบบ multithreading ก็คือ การนำ thread แต่ละ thread กระจายไปให้ CPU แต่ละตัวในระบบทำงานกับ thread แต่ละ thread พร้อมๆกัน ใน Sybase และ Informix จะใช้วิธีนี้ในการทำงาน

4.2 Shared Nothing System

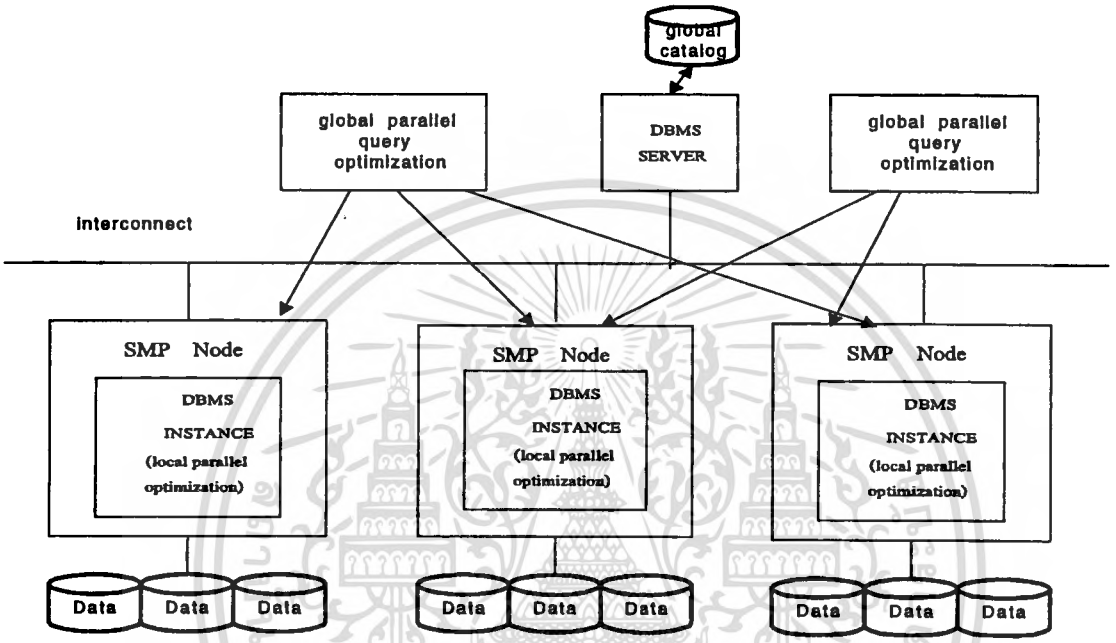
ในการทำงานของ shared nothing system หลังจากระบบรับคำสั่ง transaction จาก user ผ่านทาง backend controller แล้วระบบจะกระจายคำสั่ง (broadcast) ไปยัง database backend เพื่อให้แต่ละ backend ช่วยกันทำงานกับคำสั่งที่กระจายออกไปเหล่านั้นพร้อมๆกัน ในรูปแบบของ virtual machines ที่มี SMP หลายๆ node มารวมกันผ่านทาง network เมื่อคำสั่งย่อยๆไปถึง node ระบบจะเช็คว่าคำสั่งนั้นสามารถทำงานในแบบขนานได้หรือไม่ ถ้าสามารถทำได้ระบบก็จะให้ CPUหลายๆตัวที่มีอยู่ในแต่ละ node ช่วยกันทำงานกับคำสั่งย่อยๆเหล่านั้นพร้อมๆกัน คำสั่งย่อยๆจะถูกทำงานในแบบ parallel processing โดยระบบสามารถทำงานแบบ parallel processing ได้ทั้ง 2 ประเภท คือ throughput parallelism และ latency parallelism

ในระบบ shared nothing system ระบบจะทำ parallel query ข้าม instance และ parallel query ภายใน instance ใน shared nothing system จะมี controller ที่คอยรับ query จาก user เข้ามาในระบบ ใน controller จะกระจายคำสั่ง query ออกไปในแต่ละ instance เพื่อให้ instance แต่ละ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

instance ทำงานกับคำสั่ง query พร้อมๆกัน ส่วนนี้จะเรียกว่า global parallel optimization

ใน query ที่กระจายไปในแต่ละ instance จะถูกทำงานเป็น parallel query ภายใน instance แต่ละ instance อีก ซึ่งจะเรียกการทำงานในส่วนนี้ว่า local parallel optimization การทำงานใน shared nothing system จะเรียกว่า two-level parallelism



ภาพที่ 19 แสดง multilevel parallelism

พิจารณาคำสั่ง SQL 1 คำสั่งดังนี้

```
select *
from  T1,T1
where T1.x=T2.x
```

การทำงานของคำสั่งนี้จะเริ่มจาก

1 หาข้อมูล attribute x จากตาราง T1 ใน disk ที่มีอยู่ในระบบ

2 หาข้อมูล attribute x จากตาราง T2 ใน diskที่มีอยู่ในระบบ

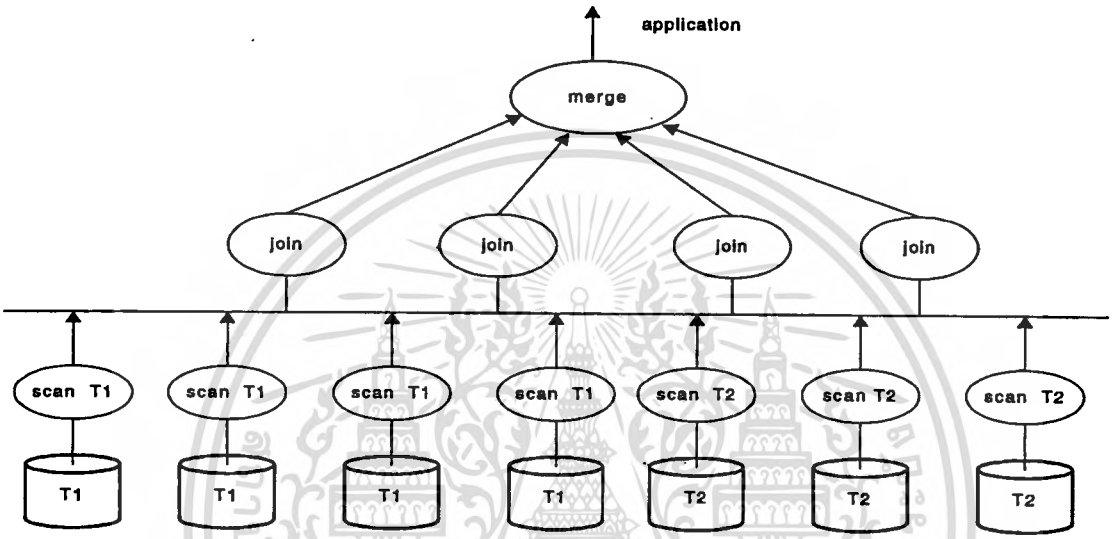
3 นำข้อมูลทั้ง 2 ตาราง มา join

4 นำผลลัพธ์ที่ได้มาแสดงผล

การทำงานในแบบ shared nothing system จะทำงานในลักษณะ split and merge คือจะนำเอาคำสั่งที่ใช้ในการค้นหาตาราง 1 ตารางกระจายออกไปหลายๆ instance เนื่องจากตารางแต่ละตารางจะถูกกระจายข้อมูล(data partition) ออกไปยัง disk ที่มีอยู่ในระบบ แต่ละ instance จะค้นหาข้อมูลและจะนำข้อมูลเข้ามาเก็บไว้ใน memory ชั่วคราว ในแต่ละ instance จากนั้นจะนำข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในตาราง ที่จะต้อง join จะเป็นตาราง T1 หรือ ตาราง T2 ตารางใดตารางหนึ่งมารวมกัน จากนั้นจะนำข้อมูลที่รวมกันนั้นกระจายออกไปในแต่ละ instance ที่มีข้อมูลอีกตารางหนึ่งอยู่ นำข้อมูลทั้ง 2 ตาราง มา join โดยในการ join จะเป็นแบบ local join จากนั้นจะนำข้อมูลที่ได้จากการ join ใน local join มา merge กันกลายเป็นผลลัพธ์ของคำสั่งทั้งหมด ดังภาพ



ภาพที่ 20 แสดง parallel query execution

ใน shared nothing system แต่ละ node จะติดต่อกันผ่านทางระบบ network เช่น ระบบ LAN ในระบบนี้จึงมีการเรียกว่าเป็นระบบ **network in a box** ดังนั้นประสิทธิภาพการของระบบจึงขึ้นอยู่กับระบบ network เป็นสำคัญ ในระบบนี้จะมีหลักการที่ทำงานที่สำคัญอยู่ 3 ประการ [16]

1 scalability of the network การเพิ่มจำนวน node เข้ามาในระบบขึ้นอยู่กับประสิทธิภาพของระบบ network ถ้าสามารถขยายประสิทธิภาพของระบบ network ได้ดีขึ้นระบบก็จะสามารถรองรับ node ได้มากขึ้นและ node แต่ละ node ก็จะสามารถส่งข้อมูลข้าม nodeได้อย่างรวดเร็วยิ่งขึ้น การขยายความสามารถของระบบ network ขึ้นอยู่กับปัจจัย 3 ประการดังนี้

1.1 bandwidth ประสิทธิภาพของระบบ network จะขึ้นอยู่กับความกว้างของช่องสัญญาน (bandwidth) ในการส่งข้อมูลยิ่งมี bandwidth กว้างมากเท่าไร ระบบก็จะยิ่งสามารถส่งข้อมูลได้อย่างรวดเร็วมากขึ้นเท่านั้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1.2 communication latency เป็นเวลาที่ใช้ในการส่ง message ข้าม network จาก node หนึ่งไปยังอีก node หนึ่ง ถ้าใช้เวลาในการส่งข้อมูลน้อย (low latency) ระบบก็จะยิ่งสามารถทำงานได้อย่างรวดเร็วมากขึ้น

1.3 reliable ระบบ network ที่ดีต้องมีความน่าเชื่อถือในการส่งข้อมูลระหว่าง node และต้องสามารถกำหนดระยะเวลาในการส่งข้อมูลข้าม node กันได้ โดยไม่ขึ้นอยู่กับระยะทางระหว่าง node

2 availability and fault tolerance of the network ใน shared nothing system ระบบ network จะต้องสามารถมีการป้องกันความผิดพลาดที่ดี (fault tolerance) เนื่องจากระบบมี node อยู่หลายๆ node ในการทำงานอาจจะมี node ใด nodeหนึ่ง เกิดเสียในระหว่างการทำงาน ระบบจะต้องสามารถทำงานต่อไปได้ โดย node ที่เสียไป จะต้องไม่มีผลกระทบต่อการทำงานของระบบ ในปัจจุบันคุณสมบัติทางด้าน reliability, availability เป็นคุณสมบัติหลักในการทำงานของระบบ

3 Usability of the network ใน shared nothing system จะถูกออกแบบมาสำหรับการทำงานที่เกี่ยวข้องกับข้อมูลจำนวนมาก ดังนั้นระบบที่ใช้ต้องเป็นระบบที่ให้ประสิทธิภาพการทำงานที่สูง สามารถรองรับข้อมูลที่เข้ามาทำงานได้คราวละมากๆ และสามารถขยายความสามารถของระบบได้ ดังนั้นการใช้ระบบ network เพื่อเชื่อมต่อระหว่าง node กับ node จำเป็นต้องใช้ระบบ network ที่สามารถให้ประสิทธิภาพการทำงานได้อย่างเต็มประสิทธิภาพ

4.3 ประเภทการทำงานในแบบขนาน

การนำระบบฐานข้อมูลแบบขนานมาใช้เป็นการเพิ่มประสิทธิภาพ การทำงานของระบบให้สูงขึ้น การทำงานแบบขนานเป็นการทำงานแบบ parallel processing ซึ่งสามารถแบ่งการทำงานออกได้เป็น 3 ประเภทใหญ่ๆ คือ

4.3.1 Parallel I/O

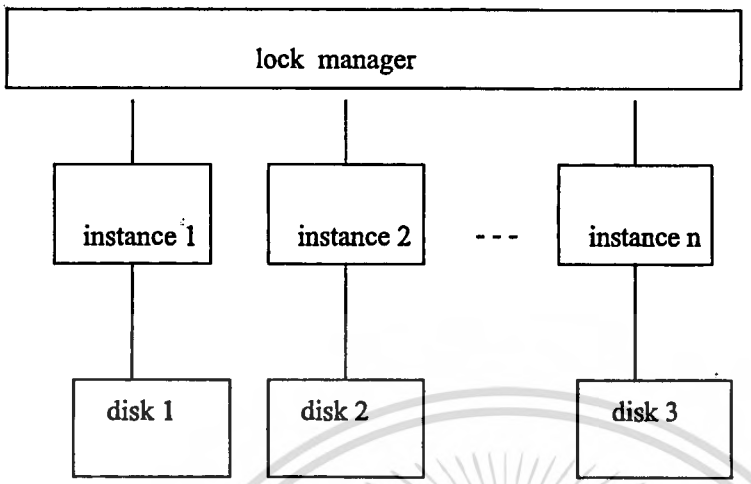
ในระบบฐานข้อมูลจะเป็นการทำงานที่เกี่ยวข้องกับข้อมูลเป็นจำนวนมากๆ ในการทำงานจะต้องมีการนำข้อมูลจากdisk เข้ามายัง memory เพื่อที่จะให้ CPU 7จัดการกับข้อมูล เช่น การ join insert, update, delete เป็นต้น หลังจากทำงานเสร็จจะต้องมีการนำข้อมูลจาก memory กลับไปยัง disk เพื่อที่จะนำข้อมูลชุดใหม่เข้ามาทำงาน เนื่องจากข้อมูลมีปริมาณมาก ระบบไม่สามารถนำข้อมูลจาก disk เข้ามายัง memory ได้ทั้งหมด จำเป็นต้องมีการนำข้อมูลเข้าและออกจาก memory อยู่ตลอดเวลา ระบบฐานข้อมูลจึงเป็นระบบที่ต้องมีการเกี่ยวข้องกับระบบ I/O เป็นอย่างมาก โดยปกติการทำงานเกี่ยวกับ disk ในการดึงข้อมูลออกจากdisk และการนำข้อมูลเข้าไปใน disk จะเป็นส่วนที่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ทำงานได้ช้าที่สุดในระบบคอมพิวเตอร์ การที่สามารถเพิ่มประสิทธิภาพการทำงานของระบบโดยการกระจายข้อมูลในตาราง 1 ตารางออกไปหลายๆ disk (data partition) ทำให้ระบบสามารถใช้ CPU หลายๆตัวดึงข้อมูลจาก disk ที่มีอยู่หลายๆ disk ในระบบพร้อมๆกัน จะเป็นการเพิ่มประสิทธิภาพการทำงานของระบบให้สูงขึ้น เพราะสามารถนำข้อมูลจาก disk มายัง memory ได้เร็วขึ้น ในกรณีของ shared disk system และ shared nothing system ที่มี instance หลายๆ instance อยู่ภายในระบบแต่ละ instance สามารถทำงานได้อย่างขนานกันและเป็นอิสระต่อกันในการทำงานข้อมูลใน disk ต้องมีการกระจายข้าม instance ที่มีอยู่ในระบบ ทำให้ต้องมีการติดต่อระหว่าง instance ซึ่งจะเป็นสาเหตุให้เกิดปัญหาในเรื่องการขอใช้ข้อมูลโดยในรูปแบบของ shared disk system อาจจะมี instance หลายๆ instance เข้ามาแก้ไขข้อมูลในตารางเดียวกันพร้อมๆกันจะทำให้เกิดปัญหา concurrency control ทำให้ข้อมูลเกิดการผิดพลาดหรือในกรณี shared nothing system ที่มี instance หลายๆ instance แต่ว่า instance แต่ละตัวมีข้อมูลเป็นของตนเองมีการจัดการข้อมูลภายใน instance อย่างเป็นอิสระ ถ้ามีการทำงานเป็นแบบ inter query parallelism คือมีการส่งคำสั่งการขอแก้ไขข้อมูลมายัง instance หนึ่งแต่ว่าข้อมูลที่จะแก้ไขอยู่อีก instance หนึ่งทำให้ instance ที่รับคำสั่งการแก้ไขข้อมูลไม่สามารถที่จะแก้ไขข้อมูลได้ ในระบบฐานข้อมูลแบบขนานจะมีวิธีการแก้ไขปัญหาได้ 2 วิธี

• 1 I/O shipping วิธีนี้ในระบบจะมีส่วนที่เรียกว่า Lock manager คอยควบคุมการใช้ข้อมูลในแต่ละ instance ถ้าหากว่า instance ไหนต้องการที่จะแก้ไขข้อมูลในตารางใด จะต้องทำการขอ lock การใช้ตารางจาก lock manager เพื่อให้ได้มีการตรวจสอบว่าตารางที่ต้องการจะแก้ไขในขณะนั้นมี instance อื่นกำลังใช้งานอยู่หรือไม่ ถ้าไม่มีการใช้ lock manager ก็จะอนุญาตให้มีการแก้ไขข้อมูลในตารางนั้น แต่ถ้าหากว่าในตารางที่ขอแก้ไขข้อมูลมี instance อื่นกำลังใช้งานอยู่ lock manager จะมีการแก้ไข queue เพื่อรอจนกว่า instance นั้นจะแก้ไขข้อมูลเสร็จ instance อื่นถึงจะมีสิทธิ์เข้ามาขอแก้ไขข้อมูลได้ วิธีนี้จะเป็นการเลื่อน I/O หรือว่าเลื่อนคำสั่งที่ขอแก้ไขข้อมูลออกไปเพื่อที่จะรอให้ตารางที่จะแก้ไขไม่มีการใช้งานจาก instance อื่นๆ ในระบบอีก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 21 แสดง I/O shipping

ใน Oracle จะใช้รูปแบบนี้ในการทำงานทั้ง shared disk system และ shared nothing system

2 function shipping วิธีการนี้ในแต่ละinstance จะคอยควบคุมดูแลข้อมูลของตัวเอง เมื่อ instance ใดต้องการแก้ไขข้อมูลในตารางใด instance นั้นจะต้องรู้ว่าข้อมูลที่จะแก้ไขอยู่ใน instance ใด จากนั้น instance นั้นจะส่งคำสั่ง SQL ไปที่ instance ที่มีข้อมูลอยู่ เพื่อให้instance นั้นได้จัดการกับคำสั่งนั้น ก่อนที่จะส่งผลลัพธ์กลับมาให้ instance ที่ส่งคำสั่งไป ในรูปแบบนี้ instance จะต้องรู้ว่าจะต้องส่งคำสั่ง SQL ไปให้ instanceใด ซึ่งจะมีการตรวจสอบได้ 2 วิธี

1 broadcast วิธีนี้ instance จะต้องกระจายคำสั่ง SQL ไปให้ instance ที่มีในระบบทั้งหมดได้ตรวจสอบว่าinstance ไหนมีข้อมูลอยู่ ถ้า instance ใดมีข้อมูลที่ตองการจะแก้ไขก็จะรับคำสั่งนั้นไปจัดการก่อนที่จะส่งผลลัพธ์กลับคืนมายัง instance เดิม วิธีนี้จะเป็นการทำให้ระบบ network ทำงานได้ช้าลง

2 วิธีนี้ instance จะต้องรู้ว่าข้อมูลที่จะแก้ไขอยู่ใน instance ใด จากนั้นจะส่งคำสั่งไปทำงานที่ instance นั้นวิธีนี้จะเป็นการประหยัด bandwidth ในระบบ แต่ว่าระบบจะต้องมีการตรวจสอบว่าข้อมูลเก็บอยู่ใน instance ใดก่อนที่จะส่งคำสั่งไป

วิธีนี้เป็นการเลื่อนคำสั่งจาก instance ที่ไม่มีข้อมูลไปยัง instance ที่มีข้อมูลทำงานกับคำสั่งนั้นแทน ก่อนที่จะส่งผลลัพธ์กลับมาให้กับ instance ที่ส่งคำสั่งไป เพื่อที่จะได้นำผลลัพธ์ส่งกลับไปที่ user ต่อไป วิธีนี้มักจะใช้ในสถาปัตยกรรมที่เป็นแบบ shared nothing system โดยที่ใน IBM, Sybase และ Informix ใช้วิธีนี้ในการทำงาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.3.2 Parallel SQL

เป็นการทำให้คำสั่ง SQL 1 คำสั่งสามารถที่จะถูกทำงานโดย CPU หลายๆตัวช่วยกันทำงานพร้อมๆกัน ในสมัยก่อนคำสั่ง SQL จะถูกทำงานโดย CPU เพียงตัวเดียวซึ่งมีสาเหตุอยู่ 2 ประการ

1 ขนาดของข้อมูลที่เก็บอยู่ใน disk ยังมีปริมาณน้อย การใช้ CPU เพียง 1 ตัวยังสามารถทำงานได้อย่างมีประสิทธิภาพ

2 คำสั่ง query ยังเป็นคำสั่งที่ไม่ซับซ้อนจึงยังไม่จำเป็นต้องใช้ CPU หลายๆตัวช่วยในการทำงาน

ปัจจุบันข้อมูลที่เก็บไว้ในระบบฐานข้อมูลมีขนาดมากขึ้นเป็นหลายๆ terabytes และคำสั่ง query ก็มีการทำงานที่ซับซ้อนมากขึ้นดังนั้นการที่คำสั่ง SQL สามารถให้ CPU หลายๆตัวทำงานร่วมกันจะเป็นการเพิ่มประสิทธิภาพการทำงานของระบบให้สูงขึ้น คำสั่ง SQLที่สามารถทำงานในแบบขนานได้จะขึ้นอยู่กับความสามารถของระบบฐานข้อมูล ที่จะอนุญาตให้ CPU หลายๆตัวช่วยกันทำงานในคำสั่งนั้นได้หรือไม่ โดยปกติคำสั่ง SQL ที่สามารถทำงานในแบบขนานก็จะมีคำสั่ง join, sort เป็นต้น

4.3.3 Parallel Administration

การทำงานที่เป็นส่วนที่ใช้ในการดูแลระบบฐานข้อมูล เช่น การสร้าง index, การทำ backup ในระบบฐานข้อมูลขนาดใหญ่ที่มีข้อมูลอยู่เป็นจำนวนมาก การดูแลจัดการระบบจะเป็นส่วนที่มีความสำคัญมากเพราะว่าจะเกี่ยวกับประสิทธิภาพการทำงานในระบบ การจัดการระบบฐานข้อมูลที่สามารถทำงานในแบบขนานสามารถแบ่งได้เป็น 3 ส่วนคือ

1 parallel index building การสร้าง index จะช่วยในการค้นหาข้อมูลที่อยู่ในตารางซึ่งจะช่วยทำให้การค้นหาข้อมูลเป็นไปอย่างรวดเร็วยิ่งขึ้น ในระบบฐานข้อมูลขนาดใหญ่การสร้าง index โดยใช้ CPU หลายๆตัวช่วยกันทำงานจะเป็นการประหยัดเวลาในการทำงานมากกว่าการสร้าง index ที่ใช้ CPU ตัวเดียวในการทำงาน

2 parallel data loading การทำ loading เป็นการนำข้อมูลที่อยู่ในรูปของ text file หรือ binary file เข้าไปไว้ในตารางที่สร้างเอาไว้ ในการนำข้อมูลเข้าไปไว้อาจจะมีการ reformat ข้อมูลเพื่อให้เหมาะสมสำหรับการเก็บข้อมูลลงในตาราง การทำ loading สามารถทำงานในแบบขนานได้เนื่องจากมีการทำ data partition กระจายข้อมูลในตารางออกไปหลายๆ disk ดังนั้นการนำข้อมูลเข้าไปเก็บไว้ในตารางสามารถ ที่จะแบ่งข้อมูลออกเป็นส่วนย่อยๆหลายๆส่วน แต่ละส่วนจะถูกทำงานโดย CPU แต่ละตัวเพื่อสามารถที่จะนำข้อมูลเข้าไปเก็บไว้ใน disk ที่มีอยู่ในระบบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3 parallel backup/restore/recovery ในระบบฐานข้อมูลความผิดพลาดในการทำงานทำให้ข้อมูลในระบบได้รับความเสียหายจึงต้องมีการกู้ข้อมูล (recovery) โดยปกติการกู้ข้อมูลจะต้องมีการ backup ข้อมูลไว้ก่อนถึงจะกู้ข้อมูลขึ้นมาได้ความผิดพลาดในระบบมีสาเหตุอยู่ 2 ประการคือ

3.1 hardware ในระบบ เช่น disk ที่เก็บข้อมูลไม่สามารถอ่านข้อมูลได้ ระบบจะต้องมีการ backup ข้อมูลไปที่ disk อื่นๆที่มีอยู่ในระบบ เมื่อเกิดปัญหาจะมีการกู้ข้อมูล (recovery) ได้นำเอา disk ที่ backup ข้อมูลกลับมาใช้งานได้ทันที

3.2 software ในระบบ เช่น ระบบฐานข้อมูลมีปัญหาไม่สามารถทำงานได้ วิธีนี้จะต้องใช้การ restore ข้อมูล โดยการนำข้อมูลที่ backup ในช่วงเวลาที่ระบบฐานข้อมูลเสีย นำข้อมูลในช่วงนั้นกลับคืนมาอีกครั้งหนึ่งเพื่อที่จะประมวลผลข้อมูลชุดนั้นอีกครั้ง ในวิธีนี้การทำ backup จะต้องมีการทำงานอยู่บ่อยๆ อาจจะทุกๆชั่วโมงหรือทุกๆครึ่งชั่วโมง เนื่องจากระบบไม่สามารถรู้ว่าระบบฐานข้อมูลจะมีปัญหาเมื่อใด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5
การนำระบบฐานข้อมูลแบบขนานไปใช้งานในปัจจุบัน

ในปัจจุบันข้อมูลที่เก็บไว้ในระบบฐานข้อมูล นับวันก็จะยิ่งมีปริมาณเพิ่มขึ้นทุกวันการค้นหา,แก้ไขหรือเปลี่ยนแปลงข้อมูล จำเป็นต้องใช้ระบบฐานข้อมูลที่สามารถค้นหาข้อมูลได้คราวละมากๆ และสามารถค้นหาข้อมูลได้อย่างรวดเร็ว การใช้ระบบฐานข้อมูลแบบขนานที่มี CPU หลายๆตัวช่วยกันทำงานจะทำให้ประสิทธิภาพของระบบสูงขึ้น จึงได้มีการนำเอาระบบฐานข้อมูลแบบขนานไปใช้กับงานหลายๆอย่างเช่น [6,7]

5.1 On-line transaction processing (OLTP) system

เป็นระบบที่มีการเก็บข้อมูลไว้ที่ศูนย์กลางและมีการเรียกใช้ข้อมูลจากผู้ใช้หลายๆคน เข้ามาอ่านข้อมูล,แก้ไขข้อมูล หรือเปลี่ยนแปลงข้อมูลพร้อมๆกัน ในปัจจุบันได้มีการใช้ระบบ OLTP ในงานหลายๆอย่าง เช่น ในระบบธนาคาร แต่ละสาขาก็จะมีการเก็บรายชื่อบัญชีลูกค้า,ยอดเงินกู้ของลูกค้าแต่ละรายที่ปล่อยกู้ออกไปหรือการโอนเงินต่างสาขา ฯลฯ ข้อมูลเหล่านี้จะมีการส่งข้อมูลไปมาระหว่างสำนักงานใหญ่กับสาขา หรือระหว่างสาขากับสาขาด้วยกัน ในระบบร้านขายของยุคใหม่ที่เป็นร้าน convenient store แต่ละสาขา จะมีการเชื่อมโยงข้อมูลกับสำนักงานใหญ่ที่ใช้เก็บคลังสินค้า เมื่อใดที่ร้านขายสินค้าชนิดใด แต่ละร้านก็จะส่งข้อมูลมาที่คลังสินค้าเพื่อบอกถึงยอดขายรวมทั้งตัด stock สินค้าในคลังสินค้าด้วย

ลักษณะโดยทั่วๆไปของระบบ OLTP

1 ข้อมูลจะต้องมีการส่งผลลัพธ์ออกไปทันทีที่มีการขอเรียกใช้ข้อมูล

2 ข้อมูลที่เก็บไว้ในระบบจะต้องสามารถอ่าน,เขียนและแก้ไขข้อมูลได้

3 ข้อมูลที่เก็บไว้จะมีปริมาณเพิ่มขึ้นตลอดเวลา เนื่องจากมีการเพิ่มข้อมูลเข้าไปเก็บไว้อยู่ตลอดเวลา

4 ผู้ใช้ในระบบจะมีอยู่หลายๆคน และมีการเรียกใช้ข้อมูลพร้อมๆกันในเวลาเดียวกัน

5 ระบบจะต้องมีการทำงานตลอด 24 ชม. ดังนั้นระบบ OLTP จะต้องมีระบบป้องกันความผิดพลาดที่ดี (fault tolerance system)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การนำระบบฐานข้อมูลแบบขนานมาใช้กับระบบ OLTP

ปัจจุบันสามารถนำเอาระบบฐานข้อมูลแบบขนานมาใช้กับระบบ OLTP โดยในระบบ OLTP อาจจะใช้สถาปัตยกรรมที่เป็นทั้งแบบ shared memory system หรือ shared nothing system ก็ได้

ประโยชน์การนำเอาระบบฐานข้อมูลแบบขนานมาใช้กับระบบ OLTP

1 สามารถเพิ่มประสิทธิภาพการทำงานของระบบ OLTP

2 จะมีระบบป้องกันความผิดพลาด ถ้าหากว่ามี CPU ตัวใดตัวหนึ่งเสียไป ระบบก็ยังมี CPU ตัวอื่นๆสามารถทำงานแทนกันได้

3 ระบบฐานข้อมูลแบบขนานสามารถสนับสนุนผู้ใช้ให้เข้ามาทำงานได้คราวละหลายๆคนพร้อมๆกัน

5.2 Decision support system (DSS)

เป็นระบบที่ช่วยในการตัดสินใจ โดยระบบนี้จะทำการรวบรวมข้อมูลทั้งหมดที่เกิดขึ้น เพื่อที่จะทำนายผลที่จะเกิดในอนาคตเป็นการให้ผู้ตัดสินใจได้รู้ถึงเหตุการณ์ที่จะเกิดในอนาคต โดยในระบบนี้จะมีแบบจำลองที่ช่วยในการตัดสินใจ ให้กับผู้ที่จะตัดสินใจสามารถรู้ได้ว่า ถ้าหากว่าเลือกที่จะตัดสินใจทางใดทางหนึ่งแล้ว ผลที่เกิดขึ้นจะเป็นอย่างไร ระบบนี้จึงเป็นระบบที่ต้องจัดการกับข้อมูลในปริมาณที่มากๆ โดยจะนำเอาข้อมูลทั้งหมดมาประมวลผลเพื่อหาทางเลือกที่ดีที่สุดที่มีอยู่

ลักษณะโดยทั่วๆไปของระบบ DSS

1 ระบบต้องมีความสามารถที่จะค้นหาข้อมูลได้อย่างรวดเร็ว

2 การค้นหาข้อมูลจะเกี่ยวข้องกับรูปแบบการค้นหาข้อมูลหลายๆรูปแบบ ข้อมูลที่ต้องการค้นหาอาจจะเป็น กราฟ หรือ รูปภาพ ก็ได้

3 ข้อมูลที่เก็บไว้จะไม่มีการเขียนข้อมูลลงไป ส่วนใหญ่จะเป็นการอ่านข้อมูลเพียงอย่างเดียว

การนำระบบฐานข้อมูลแบบขนานมาใช้กับระบบ DSS

ระบบ DSS เป็นระบบที่มีการอ่านข้อมูลเพียงอย่างเดียวไม่มีการเขียนข้อมูลลงไป ดังนั้น ระบบ DSS จึงเหมาะสมสำหรับสถาปัตยกรรมที่เป็นแบบ shared nothing system มากกว่าแบบ shared memory system เนื่องจากในแบบ shared nothing system เป็นระบบที่เหมาะสมสำหรับที่จะอ่านข้อมูลอย่างเดียว ไม่เหมาะที่จะเขียนข้อมูลลงไปในระบบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.3 การวิเคราะห์สถาปัตยกรรมระบบฐานข้อมูลแบบขนาน

สถาปัตยกรรมระบบฐานข้อมูลแบบขนาน มีอยู่ 3 ประเภท คือ shared memory system, shared disk system และ shared nothing system ซึ่งจะทำงานอยู่บนระบบคอมพิวเตอร์แบบขนานซึ่งสามารถแบ่งได้เป็น 2 ประเภท คือ

1 ระบบคอมพิวเตอร์แบบขนานแบบ SMP (symmetric multiprocessing system)

2 ระบบคอมพิวเตอร์แบบขนานแบบ MPP (massively parallel processing system)

โดยใน shared memory และ shared disk system จะเป็นการทำงานประเภท SMP ส่วนใน shared nothing system จะเป็นการทำงานประเภท MPP โดยระบบคอมพิวเตอร์แบบขนานทั้ง 2 รูปแบบ สามารถแสดงความแตกต่างของทั้ง 2 ระบบได้ดังนี้ [18]

ลักษณะต่างๆ	SMP	MPP
ชนิดของข้อมูลที่เข้ามาในระบบ	การตรวจสอบข้อมูลที่ต้องการคำตอบทันทีทันใด	การวิเคราะห์ข้อมูลที่มีความสลับซับซ้อน
ประเภทของงาน	ระบบ OLTP หรือระบบ DSS ที่มีปริมาณข้อมูลไม่มาก	ระบบ DSS ที่มีปริมาณข้อมูลเป็นจำนวนมาก
การเพิ่มปริมาณข้อมูล	มีการเพิ่มข้อมูลอย่างช้าๆ จะเพิ่มไม่เกิน 20 % ต่อปี	มีการเพิ่มปริมาณข้อมูลอย่างรวดเร็ว โดยปกติจะมีการเพิ่มมากกว่า 50 % ต่อปี
ขนาดข้อมูล	10 GB - 100 GB	มากกว่า 300 GB
จำนวน user ที่เข้ามาใช้งานในระบบ	200 - 500 คน	มากกว่า 2000 คนขึ้นไป
จำนวน transaction ที่เข้ามาทำงานใน 1 วินาที	500 - 1000 transaction	มากกว่า 2000 transaction ขึ้นไป

ตารางที่ 1 แสดงการเปรียบเทียบระบบคอมพิวเตอร์แบบ SMP กับ MPP

จากตารางที่แสดงความแตกต่างระหว่างระบบ SMP และ ระบบ MPP จะเห็นว่าระบบ SMP มีความเหมาะสมที่จะใช้งานในระบบ OLTP ในขณะที่ระบบ MPP มีความเหมาะสมที่จะใช้ในระบบ DSS

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สาเหตุที่ระบบ SMP เหมาะสมที่จะใช้ในระบบ OLTP

1 ระบบ SMP มีเสถียรภาพมากกว่าระบบ MPP ในการทำงานของระบบ OLTPจะเป็นการทำงานทุกๆวันตลอดทั้ง 24 ชม. ดังนั้นจึงควรใช้ระบบที่มีเสถียรภาพ มีความน่าเชื่อถือ ไม่มีการ down บ่อยๆ

2 ระบบ OLTP เป็นระบบที่ต้องการติดต่อกับผู้ใช้จำนวนมาก และต้องใช้เวลาในการทำงานอย่างรวดเร็ว ระบบ SMP จึงเป็นระบบที่มีความเหมาะสม เนื่องจากการติดต่อระหว่าง CPU ในระบบ SMP เป็นการติดต่อผ่านทาง memory ซึ่งจะเป็นการติดต่อที่เร็วกว่าระบบ MPP ที่ CPU ในระบบมีการติดต่อผ่านทางระบบ network

สาเหตุที่ระบบ MPP เหมาะสมที่จะใช้ในระบบ DSS

1 ระบบ DSS เป็นระบบที่เกี่ยวข้องกับการวิเคราะห์ ข้อมูลที่มีปริมาณมากๆจึงจำเป็นต้องใช้กับระบบที่สามารถรองรับข้อมูลได้มากๆและสามารถขยายขนาดของระบบได้ ระบบ MPP เป็นระบบที่สามารถรองรับข้อมูลที่มีปริมาณมากๆได้ เพราะว่าระบบสามารถขยายขนาดของระบบโดยที่ไม่มีข้อจำกัดทางด้าน hardware มาเกี่ยวข้อง

2 ระบบ DSS เป็นระบบที่มีการวิเคราะห์ข้อมูลโดยจะมีการอ่านข้อมูลจาก disk เข้ามาทำการวิเคราะห์โดยที่จะไม่มีการเปลี่ยนแปลงข้อมูลในระหว่างการวิเคราะห์ทำให้ระบบ MPP เหมาะสมที่จะใช้ในระบบ DSS เพราะว่าระบบ MPP เป็นระบบที่เหมาะสมสำหรับการอ่านข้อมูล ระบบ MPP สามารถแบ่งให้แต่ละ instance ทำงานพร้อมๆกันได้ทำให้สามารถค้นหาข้อมูลได้อย่างรวดเร็ว

ถึงแม้ว่าระบบ SMP จะเหมาะสมสำหรับระบบ OLTP และระบบ MPP จะเหมาะสมสำหรับการทำงานแบบ DSS แต่การที่จะเลือกใช้สถาปัตยกรรมแบบไหนกับงานในแต่ละประเภทก็ยังจะมีปัจจัยที่ต้องพิจารณาอีกหลายประการ เช่น

1 การเพิ่มปริมาณข้อมูลที่มีอยู่ใน disk ถ้าหากในระบบ OLTP มีการเพิ่มปริมาณข้อมูลมากกว่า 50% ต่อปี ระบบ SMP คงจะไม่เหมาะสมในการทำงาน เนื่องจากระบบ SMP มีการขยายขีดความสามารถได้อย่างจำกัดไม่สามารถขยายขนาดเพื่อรองรับ ข้อมูลที่มีปริมาณมากๆได้

2 ปริมาณข้อมูลที่มีอยู่ใน disk ระบบ SMP จะสามารถรองรับข้อมูลที่มีอยู่ได้อย่างจำกัดโดยปกติจะไม่เกิน 300 GB ถ้าหากนำใช้ระบบ OLTP ที่มีขนาดข้อมูลมากกว่า 300 GB กับระบบ SMP ก็อาจจะทำให้ระบบ SMP เกิดการ overload เกิดขึ้น

3 ระบบ SMP สามารถรองรับจำนวน transaction ได้ถึง 1000 transaction ใน 1 วินาที แต่ถ้าหากระบบเกิดมีจำนวน transaction มากกว่านั้นระบบ SMP จะไม่สามารถรองรับ transaction ที่มีปริมาณเพิ่มขึ้นได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในระบบ SMP สามารถแบ่งได้เป็น 2 ประเภท คือ shared memory system และ shared disk system โดยที่ใน shared disk system จะมีข้อดีมากกว่า shared memory system ดังนี้

1 เป็นการขยายขนาดของระบบให้มีขนาดใหญ่ขึ้น ทำให้สามารถรองรับ user ให้สามารถทำงานได้มากขึ้นและสามารถรองรับ transaction ได้เพิ่มขึ้น

2 เป็นการเพิ่ม reliability ให้กับระบบ เนื่องจากใน shared memory system จะมี จำนวน CPU อยู่หลายๆตัวติดต่อผ่านทาง memory เพียงที่เดียว ถ้าหากว่า memory ในระบบเสีย จะทำให้ระบบไม่สามารถทำงานได้แต่ในรูปแบบของ shared disk system จะมี memory อยู่หลายๆตัว จึงเป็นการช่วยเพิ่ม reliability ให้กับระบบเพราะถึงแม้ memory ที่ใดที่หนึ่งจะเสียๆไป ระบบก็ยังสามารถใช้ memory ที่เหลือทำงานทดแทนกันได้

ใน Oracle จะสนับสนุนระบบฐานข้อมูลแบบขนานที่ทำงานบนสถาปัตยกรรมแบบ shared disk system ในขณะที่ Sybase, Informix และ IBM สนับสนุนสถาปัตยกรรมแบบ shared nothing system

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 6

ORACLE

ปัจจุบันระบบจัดการฐานข้อมูลแบบขนานได้หลายๆระบบ เช่น Oracle, Sybase, Informix เป็นต้น ในที่นี้จะอธิบายการทำงานของ oracle ที่ทำงานเป็นแบบขนาน Oracle ที่ทำงานอยู่บนระบบฐานข้อมูลแบบขนาน สามารถแบ่งออกได้เป็น 3 ส่วนใหญ่ๆ คือ

1 Standard Oracle

2 Parallel query option

3 Oracle Parallel server

โดยใน Standard Oracle จะเป็นการทำงานอยู่บนระบบคอมพิวเตอร์ที่มี CPU, memory และ disk เพียง 1ตัว ถ้าหากว่าระบบเป็น shared memory system จะต้องใช้ Standard Oracle + Parallel query option แต่ถ้าหากว่าระบบเป็น shared disk system หรือ shared nothing system ระบบจะต้องใช้ Standard Oracle + Parallel query option + Oracle Parallel server ใน shared memory system จะมีจำนวน node หรือจำนวน instance เพียง 1 instance จึงไม่จำเป็นต้องใช้ Oracle Parallel server แต่ว่าในระบบ shared disk system หรือ shared nothing system จะมีจำนวน instance > 1 instance จึงจำเป็นต้องใช้ Oracle Parallel server มิฉะนั้นถ้าแต่ละ instance เข้ามาใช้ข้อมูล ในตาราง เดียวกันพร้อมๆกัน จะทำให้เกิดปัญหา concurrency control ได้ จำนวน instance หมายถึง ระบบคอมพิวเตอร์ที่มี CPUจำนวน 1 ตัวหรือมากกว่าก็ได้แต่ว่าต้องมี memory เป็นของตนเอง แต่ว่า disk อาจจะมีเป็นของตนเองหรืออาจจะใช้งานร่วมกับ instance อื่นก็ได้

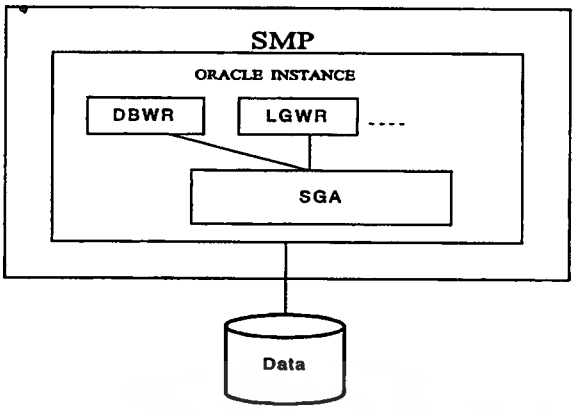
6.1 ประเภทของ hardware

Oracle มีการแบ่ง hardware ออกได้เป็น 4 ประเภทใหญ่ๆ คือ

6.1.1 Single instance with exclusive access

รูปแบบนี้ระบบจะมี 1 instance ทำงานอยู่ โดยระบบจะมีลักษณะเหมือนกับ shared memory system โดยที่ในระบบจะมี CPUอยู่หลายๆตัว แต่ว่าใช้ memory และ disk ร่วมกัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 22 แสดง Single Instance

System global area (SGA) จะมีอยู่ทุกๆ instance SGA จะเป็นส่วนที่อยู่ใน memory ซึ่งจะประกอบด้วยข้อมูลและส่วนที่ใช้ในการควบคุม (background process) การทำงานของ oracle instance ข้อมูลที่เก็บอยู่ใน SGA จะแบ่งออกเป็นหลายๆส่วน ดังนี้

1 Database buffer cache ส่วนนี้จะเป็นการเก็บข้อมูลที่ถูกใช้บ่อยๆเอาไว้ใน memory ข้อมูลส่วนไหนที่ไม่ได้ถูกใช้จะถูกนำไปไว้ในdisk

2 Redo log buffer ส่วนนี้จะใช้ในการเก็บ redo entries ซึ่งจะเป็นส่วนที่ใช้ในการเก็บการเปลี่ยนข้อมูลใน database เอาไว้ ส่วนนี้จะถูกใช้ในกรณีที่ข้อมูลใน dusk เกิดเสียหายก็จะใช้ส่วนนี้ในการกู้ข้อมูลกลับคืนมา

3 Shared pool ส่วนนี้จะใช้ในการเก็บโครงสร้างของ memory เอาไว้

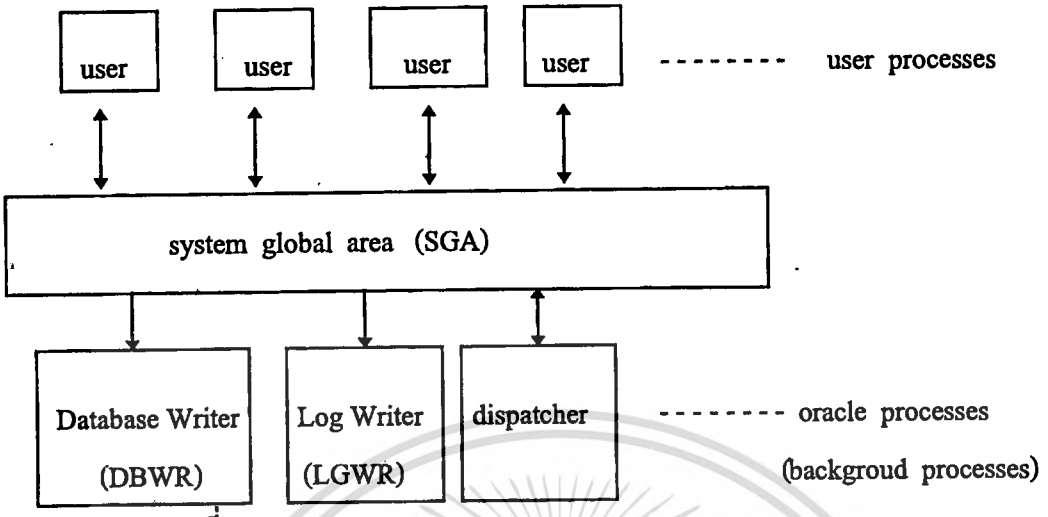
4 Cursors เป็นส่วนที่ใช้ในการติดต่อคำสั่งพิเศษที่มีอยู่ในระบบ

Process คือ การที่ OS จะนำคำสั่งเข้ามาทำงานภายในระบบ ภายใน Oracle โครงสร้าง ของ process สามารถแบ่งได้เป็น 2 ประเภทใหญ่ๆคือ

1 single-process oracle instance ระบบนี้จะยอมให้ user ส่ง process เข้ามาทำงานได้คราวละ 1 process ในรูปแบบนี้ oracle จะถูกทำงานบน OS ที่เป็น MS-DOS

2 Multiple-Process Oracle Instance ระบบจะยอมให้ user สามารถเข้ามาใช้ข้อมูลได้พร้อมๆกันหลายๆคน ดังภาพ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 23 แสดง multiple- process

ใน multiple- process system Process สามารถแบ่งได้เป็น 2 ประเภทใหญ่ๆคือ

1 user processes เป็น processes ที่ถูกสร้างขึ้นโดย user ซึ่ง user จะเข้ามาใช้ข้อมูลหรือเปลี่ยนแปลงข้อมูลในระบบ เช่น insert, delete, update เป็นต้น

2 oracle processes เป็น processes ที่ถูกเรียกโดย processes อื่นๆ process นี้จะช่วยการทำงานของ process อื่นๆ ที่เข้ามาทำงานในระบบ oracle processes จะมีอยู่ 2 ประเภทใหญ่ๆ คือ

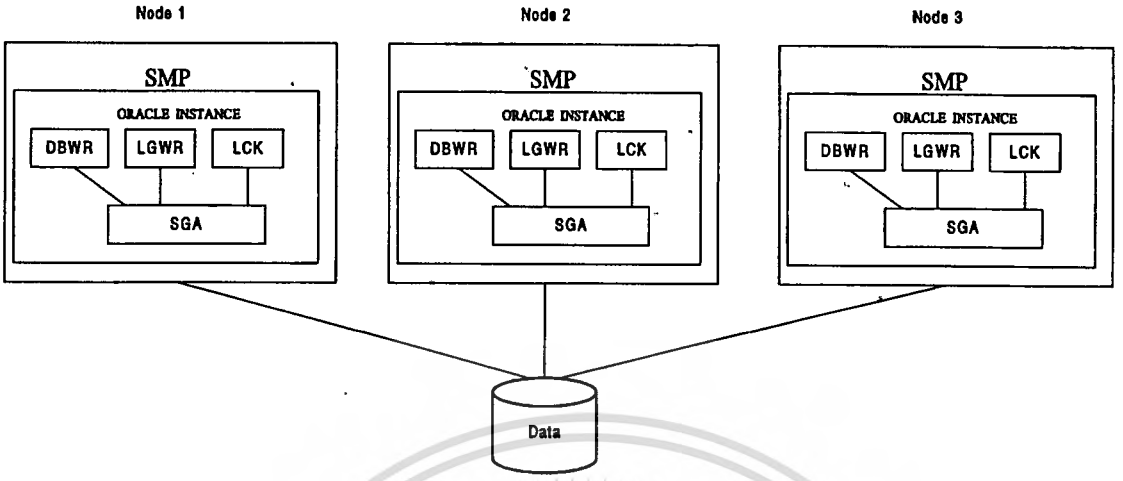
2.1 server processes เป็น processesในส่วนของ server ที่จัดการการทำงานในคำสั่ง user processes

2.2 background processes เป็น processes ที่ช่วยในการทำงานของระบบ โดยที่ process จะทำงานในแบบ asynchronous background processes จะมีอยู่หลายๆ processes เช่น Dispatcher, Lock เป็นต้น

6.1.2 Multiple-instances Database system

รูปแบบนี้จะเป็นลักษณะของ shared disk system โดยในรูปแบบนี้ระบบจะมีจำนวน instance อยู่ หลายๆ instance แต่ว่าจะมี DBMS เพียง 1 ตัว โดยที่แต่ละ instance จะเข้ามาใช้ข้อมูลในส่วนกลางเพียงส่วนเดียว ในรูปแบบนี้จะต้องใช้ LCK process เข้ามาป้องกันการใช้ข้อมูลพร้อมกันในแต่ละ instance

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

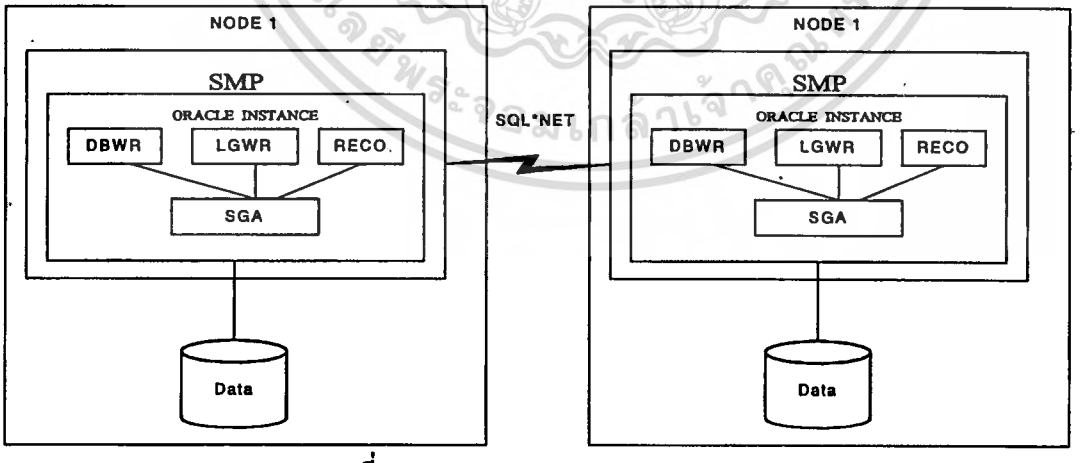


ภาพที่ 24 แสดง multiple-instance

โดยระบบนี้จะต้องทำงานโดยใช้ Oracle Parallel Server (OPS) ช่วยในการทำงานป้องกันไม่ให้แต่ละ instance แก้ไขข้อมูลในตารางเดียวกันพร้อมๆกัน

6.1.3 distributed database system

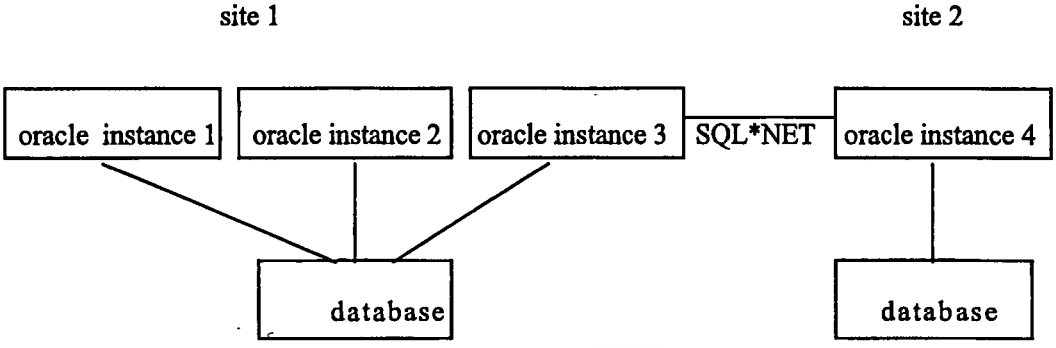
รูปแบบนี้จะเป็นการทำงานในแบบ distributed แต่ละnode จะเชื่อมโยงกันผ่านทาง SQL*NET การทำงานในส่วนนี้จะไม่มี LCK processes ในการทำงานเพราะว่าจะไม่มีการใช้ OPS ในการทำงาน [5]



ภาพที่ 25 แสดง distributed database system

ในแบบนี้จะเพิ่ม background process ส่วนที่เป็น RECO เข้ามาในการทำงานแบบ distributed แต่การทำงานในแบบนี้ในแต่ละ site อาจจะมีหลายๆ node ทำงานอยู่ก็ได้ ถ้าหากว่าในแต่ละ site มี node อยู่หลายๆ node การทำงานใน site นั้นจะเป็นแบบ parallel

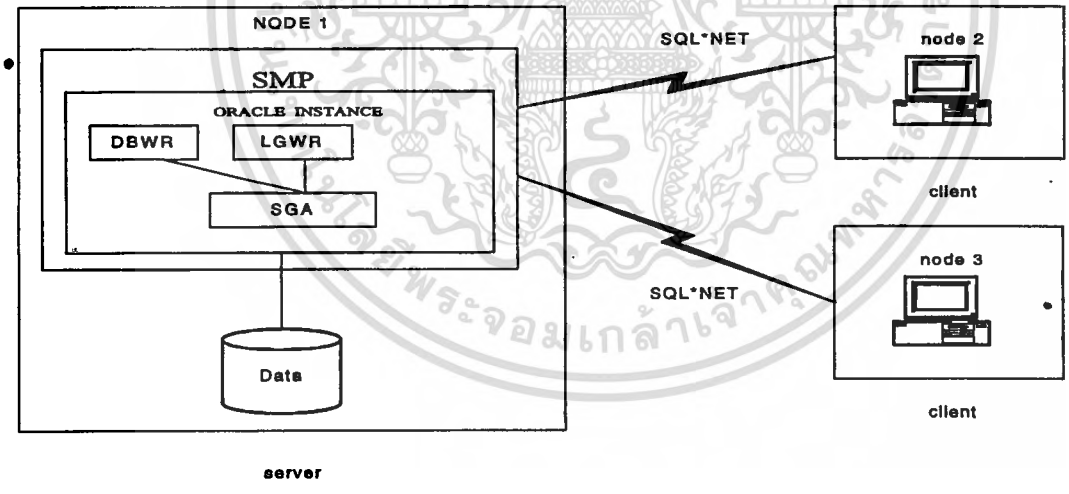
เอกสารนี้มีลิขสิทธิ์ ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ถ้าเป็นรูปแบบนี้ ใน site 1 จะต้องใช้ OPS เพื่อที่จะได้สามารถให้หลายๆ instance ใช้ข้อมูลร่วมกันได้

6.1.4 client-server systems

รูปแบบนี้จะทำงานเป็นแบบ client-server ใน client จะเป็นผู้ขอบริการ server จะเป็นผู้ให้บริการ ใน server อาจจะมีจำนวน instance เพียง 1 instance หรือมากกว่าก็ได้ในการงาน [5]



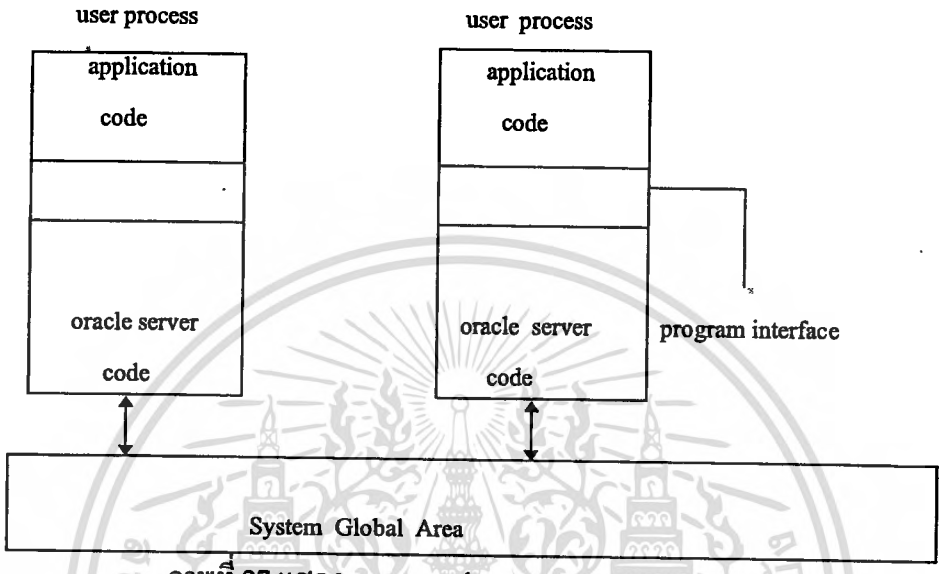
ภาพที่ 26 แสดง client-server systems

6.2 ประเภทของ server processes

ใน Oracle สามารถที่จะแบ่งการทำงาน server process ที่จะจัดการกับ user process ได้เป็น 3 ประเภท

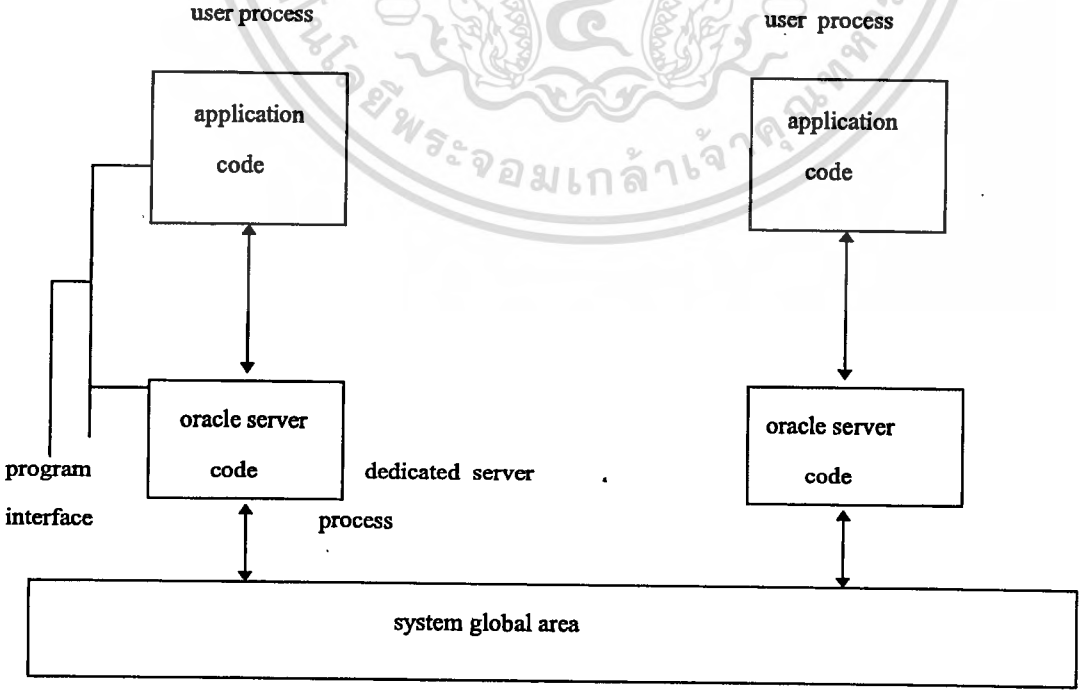
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6.2.1 เมื่อระบบรับ user process เข้ามาแต่ละ user process จะรวมกับ server process กลายเป็น process เดียวรูปแบบนี้จะใช้ใน OS ที่เป็น VAX,VMS แต่ถ้าเป็นระบบ UNIX ระบบจะไม่สามารถทำงานในรูปแบบนี้ได้ เนื่องจากจะเกิดปัญหา data integrity ขึ้น



ภาพที่ 27 แสดง compound server process

6.2.2 เมื่อ user process เข้ามาในระบบ server จะสร้าง server process เพื่อมาจัดการกับคำสั่ง user process รูปแบบนี้เรียกว่า dedicates server architecture

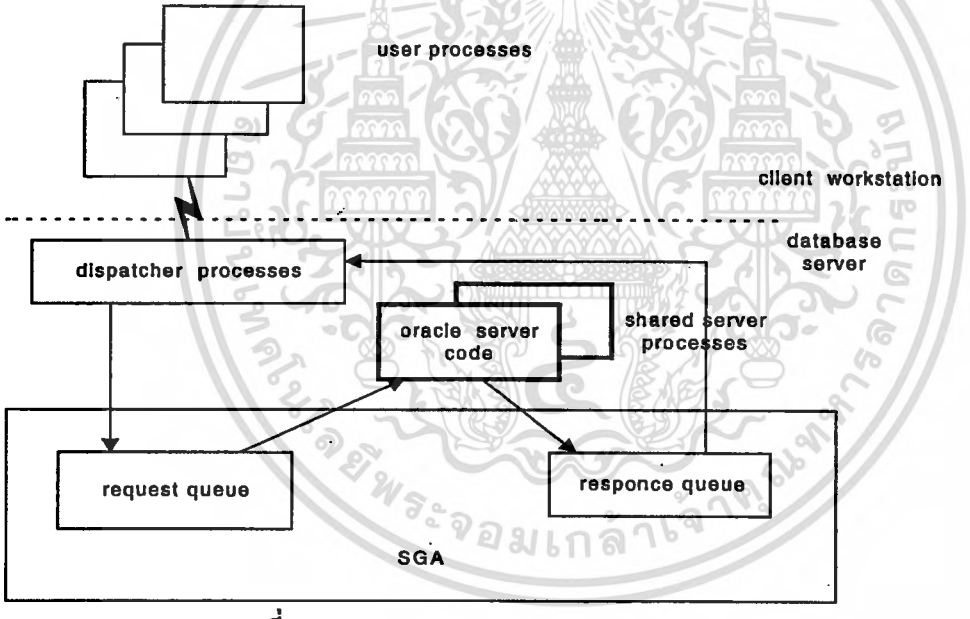


ภาพที่ 28 แสดง dedicated server process

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากภาพเมื่อ user process เข้ามา ระบบจะสร้าง server process ขึ้นมารองรับการทำงานของ user process นั้นๆโดยเฉพาะ เมื่อทำงานเสร็จ server process ก็จะสลายไปเป็นการทำงานในแบบ 1-1 ระหว่าง user process กับ server process รูปแบบนี้ user process กับ server process จะทำงานกันคนละเครื่อง มักจะเป็นรูปแบบของ client-server architecture

6.2.3 รูปแบบนี้ภายในระบบจะสร้าง shared server processes หลายๆ process เตรียมไว้ในระบบอยู่แล้ว ระบบจะมี background process ที่ชื่อ dispatcher จะรับ user process ที่อยู่ใน request queue เข้าไปทำงานใน shared server process ที่มีอยู่ในระบบ โดยที่แต่ละ user process จะถูกทำงานโดย server process แต่ละ process ที่อยู่ใน shared server process ในแบบนี้จะเรียกว่า multi-threaded server architecture

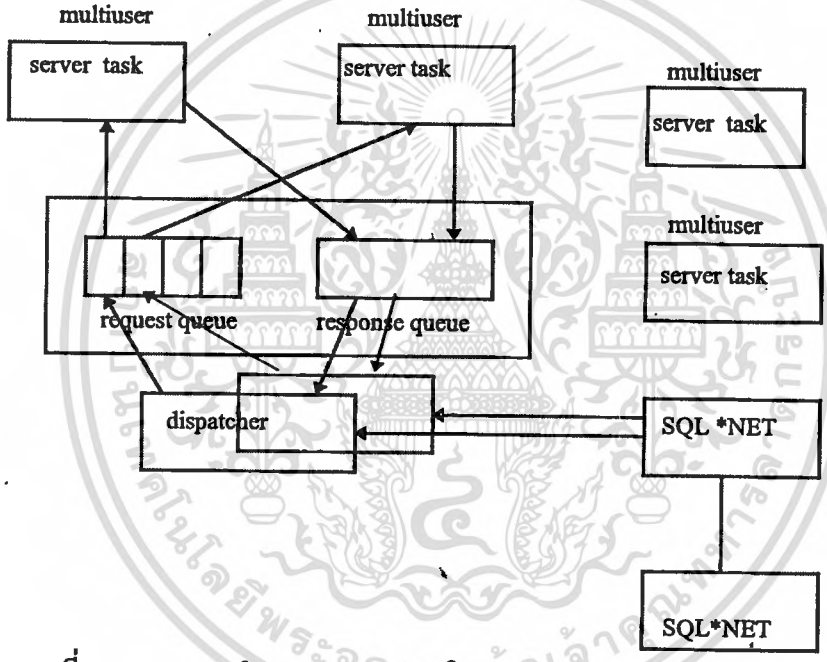


ภาพที่ 29 แสดง multi-thread server process

จากภาพ user processes จะถูกส่งเข้ามาทำงานใน database server โดยมี background processes ที่ชื่อว่า dispatcher เป็น process ที่นำคำสั่งจาก user processes ไปเข้า queue ใน request queue จากนั้น server processes ที่ว่างจะเข้ามาเช็ค queue ที่อยู่ใน request queue เพื่อที่จะนำ processes ใน request queue เข้าไปทำงาน หลังจากนั้นจะนำผลลัพธ์มาที่response queue เพื่อที่จะให้ dispatcher processes นำผลลัพธ์จาก response queue ไปให้กับ user processes ที่มีการส่งคำถามเข้ามาใน SGA ใน multi-thread server process จะสามารถทำให้ระบบรองรับผู้ใช้ได้มากกว่ารูปแบบอื่นๆ

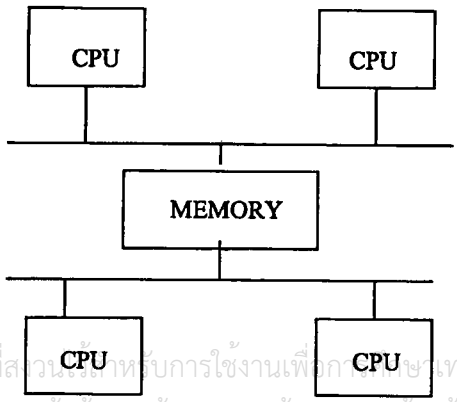
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ใน shared memory system และ shared disk system จะมีการทำงานเป็นแบบ interquery parallelism คือระบบจะมี CPU อยู่หลายๆตัวดังนั้นเมื่อระบบรับ user process เข้ามาในระบบ dispatcher process จะนำ user process แต่ละprocess ไปทำงานกับ CPU ในแต่ละตัว เช่น สมมติว่า user ส่งคำสั่ง query เข้ามา 6 query และในระบบมี CPU อยู่ 3 ตัว query ที่ 1 จะไปทำงานใน CPU ตัวที่ 1 , query ที่ 2 จะไปทำงานใน CPU ตัวที่ 2 ,query ที่ 3 จะไปทำงานใน CPU ตัวที่ 3,query ที่ 4 จะไปทำงานใน CPU ตัวที่ 1 สลับกันเช่นนี้ไปเรื่อยๆจนกว่าจะครบจำนวน query ที่เข้ามาใช้ข้อมูลทั้งหมด ดังภาพ [7]



ภาพที่ 30 แสดงการทำงานของ Oracle ในแบบ shared memory system

จากภาพการทำงานของระบบ สามารถเปรียบเทียบกับภาพของ shared memory system ที่มี CPU ในระบบอยู่หลายๆตัว ดังนี้



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในการที่แต่ละ CPU รับ user process เข้าไปทำงาน ภายใน CPU จะมีการทำงานเป็นแบบ parallel query ซึ่งจะได้กล่าวอย่างละเอียดในส่วนของ parallel query option ในส่วนของ shared disk system และ shared nothing system เนื่องจากระบบมีอยู่หลายๆ instance ดังนั้นระบบจึงต้องใช้ Oracle Parallel server ช่วยในการทำงานเพื่อเป็นการป้องกันปัญหา concurrency control อันเนื่องมาจากการที่ instance หลายๆตัวเข้ามาแก้ไขข้อมูลตารางเดียวกันพร้อมๆกัน ทำให้ข้อมูลเกิดความไม่ถูกต้องได้ ดังนั้นในระบบ shared disk และ shared nothing system จะต้องใช้ standard oracle + parallel query option + oracle parallel server ในการทำงาน

6.3 Oracle Parallel Server

Oracle Server เป็นระบบฐานข้อมูลที่จัดการข้อมูลบน Server Oracle Server แบ่งออกเป็น 2 ส่วน คือ Oracle database และ Oracle instance Oracle Server ทำงานอยู่ 2 mode คือ exclusive mode และ shared mode

1 exclusive mode เป็นการทำงานในระบบที่มี Oracle instance เพียง 1 instance ทำงานอยู่บนระบบ ใน mode นี้จะมี OPS อยู่หรือไม่มีก็ได้

2 shared mode เป็นการทำงานที่สามารถมีoracle instance อยู่ 1 instance หรือมากกว่าในการทำงาน ใน mode นี้จะต้องมี OPS ติดตั้งอยู่บนระบบด้วย ใน shared mode จะแบ่งการทำงานได้เป็น 2 อย่าง

1 single shared mode เป็นการทำงานที่มี Oracle instance เพียง 1 instance ทำงานอยู่บนระบบ

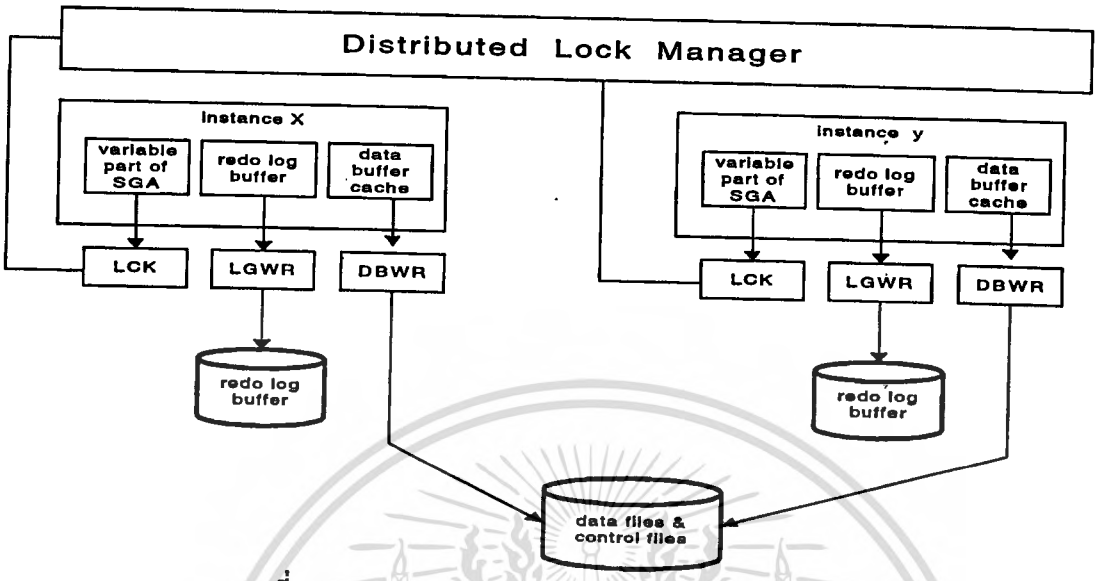
2 multiple shared mode เป็นการทำงานที่มี Oracle instance หลายๆ instance ทำงานอยู่บนระบบ

oracle+option	exclusive mode	shared mode	
		single mode	multiple mode
OPS not installed	yes	no	no
OPS installed	yes	single shared	multiple shared

ตารางที่ 2 แสดงการทำงานของ OPS

โครงสร้างของ Oracle Parallel Server จะมีรูปแบบดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 31 แสดงการทำงานของOracle Parallel Server

จากภาพจะเห็นว่าแต่ละ instance จะมี system global area (SGA) และ background processes และจะมี redo log files ในแต่ละ instance ในระบบจะมีการใช้ datafile และ control file ร่วมกัน แต่ละ instance สามารถรองรับ users หลายๆคนในการเข้ามาใช้ข้อมูลการที่จะให้ instance หลายๆ instance สามารถทำงานพร้อมๆกันได้ จะต้องมีการใช้การทำงาน 3 ส่วน คือ

1 lock process (LCKn) เป็น background processes ที่มีอยู่ใน system global area (SGA) ส่วนนี้จะเป็นส่วนที่คอยทำหน้าที่รับ-ส่งสัญญาน การติดต่อข้อมูลใน disk LCKn จะมีทั้งหมด 10 processes คือตั้งแต่ (LCK0-LCK9)

2 Parallel cache management (PCM) ทำหน้าที่ในการ lock ข้อมูลเพื่อที่จะทำให้ instance สามารถเข้ามาแก้ไขข้อมูล (update)จะเป็นการทำงานในแบบ instance lock

3 Distributed lock manager (DLM) เป็นส่วนประกอบของ software จะทำหน้าที่ควบคุมการแก้ไขข้อมูลในทุกๆ instance ที่มีในระบบ ส่วนนี้จะเป็นการป้องกันไม่ให้มีการแก้ไขข้อมูลตารางเดียวกัน พร้อมๆกัน

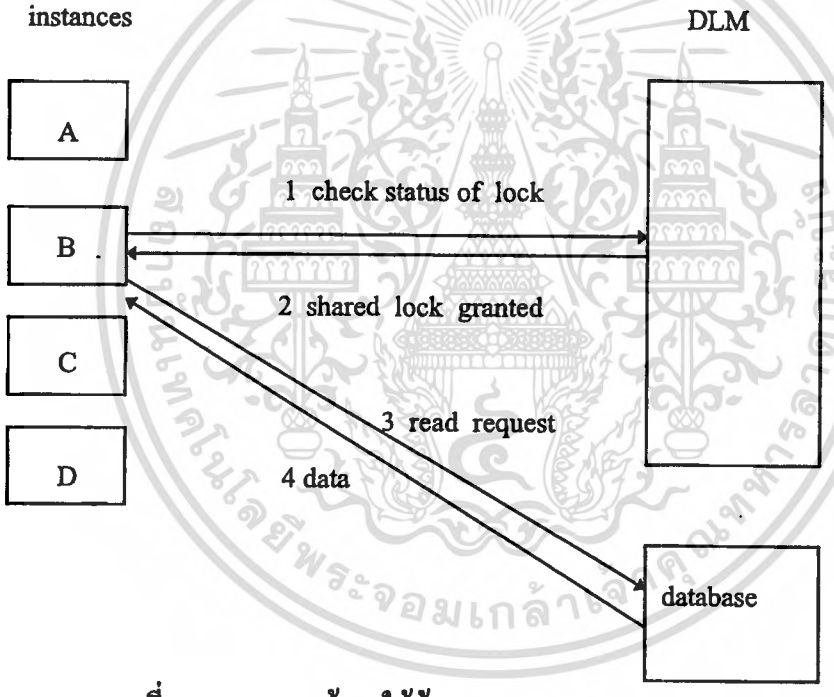
การที่ระบบมีหลายๆ instance ทำงานพร้อมๆกัน จะเกิดปัญหา concurrency control ดังนั้นแต่ละ instance เมื่อจะมีการแก้ไขเปลี่ยนแปลงข้อมูลใน disk แต่ละ instance จะต้องทำ PCM lock โดยการใช้ PCM ส่งสัญญาน (LCKn) ขอแก้ไขข้อมูลไปที่ DLM เพื่อที่จะให้ DLM ได้ตรวจสอบว่าข้อมูลในตารางที่ต้องการจะแก้ไขมี instance อื่นๆ กำลังแก้ไขตารางนั้นอยู่หรือไม่ โดยใน DLM จะมีอยู่ 2 queue ที่ใช้ในการจัดการกับคำสั่ง lock

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1 grant queue คำสั่ง lock จะสามารถ lock ตารางเพื่อที่จะให้instance สามารถแก้ไขเปลี่ยนแปลงข้อมูลได้

2 convert queue ถ้าคำสั่ง lock ที่เข้ามาปรากฏว่าตารางนั้นถูกทำงานโดย instance อื่น ที่อยู่ใน grant queue คำสั่ง lock จะเข้ามาที่ convert queue เพื่อที่จะรอว่าถ้าเมื่อไรที่ instance ที่ lock ตารางทำงานเสร็จแล้วปลด lock คำสั่ง lock จาก convert queue จะถูกส่งเข้ามาทำงานที่grant queue

ถ้าหากว่าคำสั่ง lock นำมาไว้ที่ grant queue แล้ว DLM จะส่งสัญญานมาที่ instance เพื่อให้ instance สามารถอ่านข้อมูลจาก disk ขึ้นมาที่ SGA เพื่อที่จะทำการเปลี่ยนแปลงข้อมูลในตาราง หลังจากแก้ไขข้อมูลแล้ว ระบบจะทำการปลด lock เพื่อที่จะให้ instance อื่นสามารถเข้ามาใช้ข้อมูลในตารางนั้นได้ ดังภาพ



ภาพที่ 32 แสดงการเข้ามาใช้ข้อมูลของ instance

การใช้ PCM lock ในการนำข้อมูลจาก disk มาที่ SGA เพื่อที่จะแก้ไข ข้อมูล หลังจากแก้ไขแล้วจะต้องมีการเขียนข้อมูลลงไปใน disk ก่อนถึงจะทำการปลด lock ได้

<u>ตัวอย่าง</u> การทำงานของ PCM locks ในระบบ

1 instance x เป็นเจ้าของ PCM lock นำข้อมูลจากตาราง X เข้ามาใน SGA เพื่อที่จะ update แถวที่ 1

2 instance y ขอเข้ามาแก้ไขตาราง X ในแถวที่ 3

3 instance X แก้ไขข้อมูลเสร็จเขียนข้อมูลลง disk แล้วปล่อย PCM lock

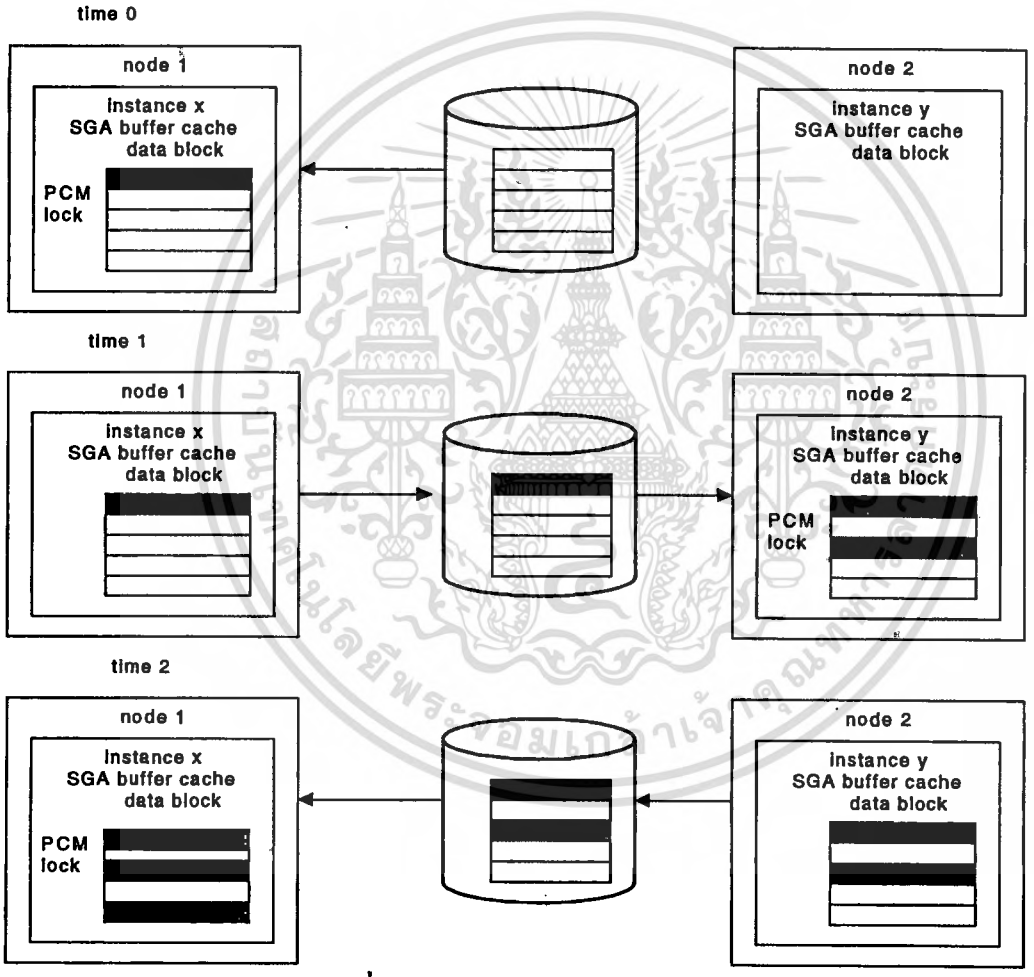
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4 instance y เป็นเจ้าของ PCM lock นำข้อมูลจากตาราง X เข้ามาใน SGA เข้ามาใน SGA เพื่อที่จะแก้ไขแถวที่ 3 5 instance x ขอแก้ไขตาราง X ในแถวที่ 5

6 instance y เขียนข้อมูลลง disk แล้วปล่อย PCM lock

7 instance y เป็นเจ้าของ PCM lock นำข้อมูลตาราง X เข้ามาใน SGA เพื่อที่จะ update แถวที่ 5

8 instance x เขียนข้อมูลลง disk และจะเป็นเจ้าของ PCM lock ต่อไปจนกว่าจะมีinstance อื่นมาขอแก้ไขข้อมูลในตาราง X



ภาพที่ 33 แสดงตัวอย่างการทำงานของ PCM

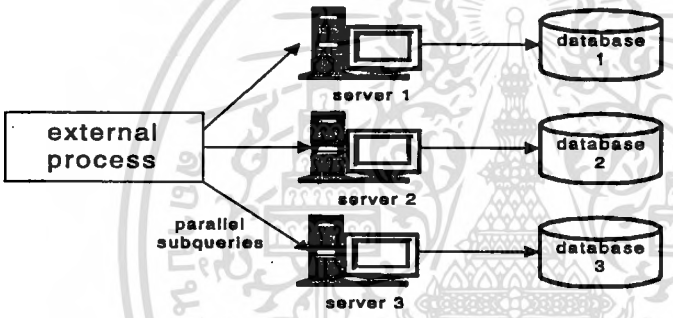
ใน Oracle 7 จะสามารถทำงานในแบบ shared memory และ shared disk system ได้ดีกว่าแบบ shared nothing system โดยใน Oracle v 7 จะสามารถทำงานใน shared nothing system หากว่า hardware สามารถมอง disk แต่ละตัวในระบบให้เหมือนกับมี disk อยู่เพียงตัวเดียวเหมือนกับรูปแบบของ shared disk system.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6.4 Parallel query option

การทำ Parallel query เป็นการทำให้ระบบสามารถทำงานได้อย่างรวดเร็วยิ่งขึ้น Parallel query สามารถทำงานได้ในสถาปัตยกรรมทั้ง 3 รูปแบบของ parallel database โดยที่ Parallel query สามารถที่จะทำงานได้ด้วยตนเองหรือสามารถทำงานร่วมกับ Oracle Parallel Server ก็ได้ ระบบสามารถแบ่งประเภทของ Parallel query ออกได้เป็น 2 ประเภท คือ

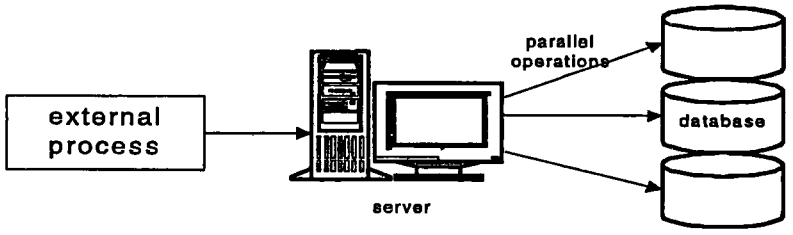
1 External Parallelism รูปแบบนี้จะเป็นรูปแบบของ shared nothing system คือในแต่ละ CPU จะมี disk และ memory เป็นของตนเองข้อมูลแต่ละตารางจะกระจายอยู่ในแต่ละ disk ระบบจะรับ external process เข้ามาและแตกออกเป็น subquery ย่อยๆ แต่ละ subquery จะกระจายไปในแต่ละ CPU เพื่อให้สามารถทำงานได้พร้อมๆกัน ดังภาพ



ภาพที่ 34 แสดง external parallelism

ใน subquery จะถูกทำงานพร้อมๆกันในแต่ละserver ก่อนที่จะส่งผลที่ได้ในแต่ละserver มารวมกันที่ external process

2 Internal Parallelism รูปแบบนี้จะเป็นการทำงานของ shared memory และ shared disk system โดยที่ server จะแบ่งคำสั่ง query ที่เข้ามาทำงานออกเป็นคำสั่งย่อยๆ และตรวจสอบว่าคำสั่งไหนสามารถทำงานในแบบขนานได้ก่อน ถ้าพบว่ามีคำสั่งไหนที่สามารถทำงานในแบบขนานได้จะมี process หลายๆ process มาทำงานกับคำสั่งย่อยๆเหล่านั้นพร้อมๆกัน ซึ่งจะเป็นการทำงานที่เรียกว่า parallel processing



ภาพที่ 35 แสดง internal parallelism

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

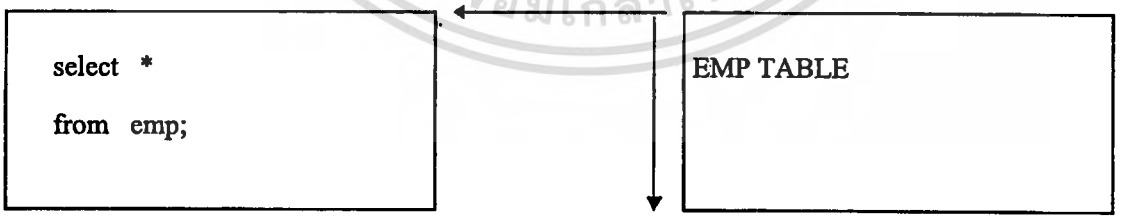
รูปแบบนี้จะเป็นการทำให้ระบบเกิดความสมดุล (load balancing) ในการทำงาน เนื่องจาก CPU ในระบบสามารถที่จะติดต่อกับ disk ได้ทุกๆตัวในระบบซึ่งจะต่างจาก external parallelism ที่จะเกิดปัญหานี้เพราะว่า CPU มี disk เป็นของตนเอง ดังนั้นถ้าข้อมูลในตารางกระจายในแต่ละ disk ไม่เท่ากันจะทำให้ระบบเกิดความไม่สมดุลเกิดขึ้น

6.4.1 Parallel query processing

การทำ Parallel query option [9] เป็นการทำให้ process หลายๆ process ทำงานกับคำสั่ง SQL 1 คำสั่งพร้อมๆกันซึ่งจะเป็นการทำงานที่เรียกว่า parallel query processing โดยปกติเมื่อ user ส่งคำสั่ง query ที่เป็น user process เข้ามา ระบบจะมี server process เพียง 1 process จัดการกับ user process นั้นแต่ถ้าเป็นการทำงานในส่วนของ parallel query option เมื่อมี user process เข้ามา ระบบจะมี process หลายๆ process จัดการกับคำสั่งใน user process โดยที่ server process จะเช็คว่าคำสั่งใน SQL คำสั่งไหนสามารถทำงานแบบขนานได้ ถ้าพบว่าคำสั่งไหนสามารถทำงานในแบบขนานได้ ระบบจะมี process หลายๆ process ช่วยทำงานกับคำสั่งนั้น ทำให้คำสั่ง SQL นั้นสามารถทำงานได้อย่างรวดเร็วมากยิ่งขึ้นทำให้ประสิทธิภาพของระบบสูงขึ้น

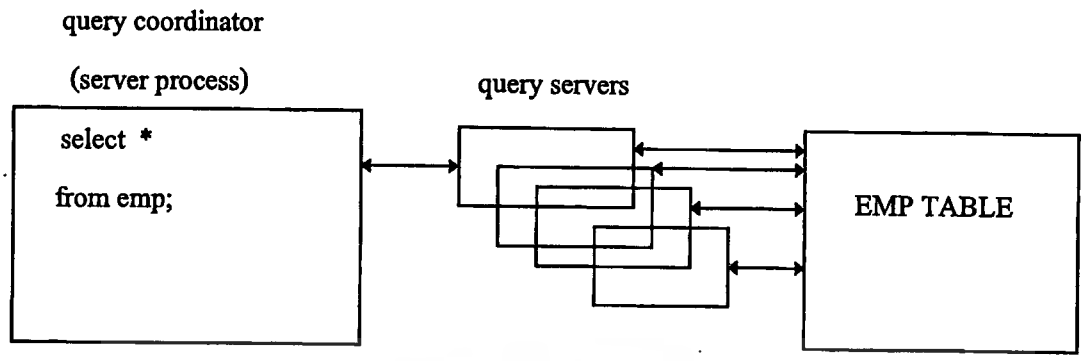
สถาปัตยกรรมของ Parallel query Processing

ในรูปแบบปกติที่ไม่มีการใช้ parallel query option ระบบจะมี process เพียง 1 process ที่ทำงานกับคำสั่ง query นั้นๆ เช่น



แต่ถ้าหากเป็นการทำงานแบบ parallel query option ใน server process ที่เรียกว่า query coordinator จะเป็นส่วนที่ใช้ตรวจสอบว่าคำสั่งไหนใน query ที่สามารถทำงานในแบบขนานได้ ถ้าพบว่าคำสั่งไหนสามารถทำงานแบบขนานได้จะส่งคำสั่งนั้นไปที่ process ที่ชื่อว่า query servers เพื่อให้ query servers ที่มีอยู่หลายๆ process ช่วยกันทำงานคำสั่งนั้นพร้อมๆกันจากนั้น แต่ละ query server จะส่งผลลัพธ์ในการทำงานที่ได้ในแต่ละprocess กลับมารวบรวมที่ query coordinator ต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



จากภาพการทำงานจะเห็นว่าการค้นหาตาราง EMP ภายใน query optimization จะแบ่งการทำงานออกเป็นหลายๆส่วน แต่ละส่วนจะถูกจัดการโดย query server เมื่อ query server ทำงานเสร็จก็จะนำเอาผลลัพธ์ที่ได้มารวมกันที่ query coordination เพื่อที่จะแสดงผลลัพธ์ต่อไป

query coordinator จะเป็น server process ที่ถูกสร้างขึ้นมาเพื่อที่จะจัดการกับ user process ที่เข้ามาในระบบ แต่query coordinator สามารถนำคำสั่งจาก user process ไปให้ query serversที่มีหลายๆ process ทำงานพร้อมๆกันได้ และจะนำผลลัพธ์ที่ได้ในแต่ละ query server กลับมารวบรวมอยู่ในquery coordinator เพื่อที่จะได้เป็นผลลัพธ์ของคำสั่ง user process จำนวน query server ที่สามารถทำงานกับคำสั่งที่สามารถทำงานในแบบขนานได้จะเรียกว่า degree of parallelism

6.4.2 การทำงานในแบบขนาน

การทำงานเป็นแบบขนานจะเริ่มจากระบบจะรับคำสั่ง query จาก user ที่เป็น user process เข้ามาสู่ระบบ จากนั้นจะนำคำสั่งไปยังส่วน optimization เพื่อที่จะเลือกวิธีที่มีประสิทธิภาพมากที่สุดในการทำงาน เช่น เลือกวิธีการ join ที่มีประสิทธิภาพมากที่สุด เป็นต้น จากนั้นระบบจะมี server process ที่ชื่อ query coordinator มาตรวจสอบว่า คำสั่งไหนใน query ที่สามารถทำงานในแบบขนานได้ ก่อนที่จะนำคำสั่งนั้นส่งไปให้กับ query servers ที่มีหลายๆ process มาจัดการคำสั่งนั้นๆ และรวบรวมผลลัพธ์ที่ทำได้ในแต่ละ process มารวมไว้ที่ query coordinator เพื่อที่จะนำผลลัพธ์ที่ได้ส่งกลับไปหา user ต่อไป

คำสั่ง query ที่สามารถทำงานแบบขนานได้มีอยู่ 5 คำสั่ง คือ

1 sort

2 join

3 scans

4 table population

5 index creation

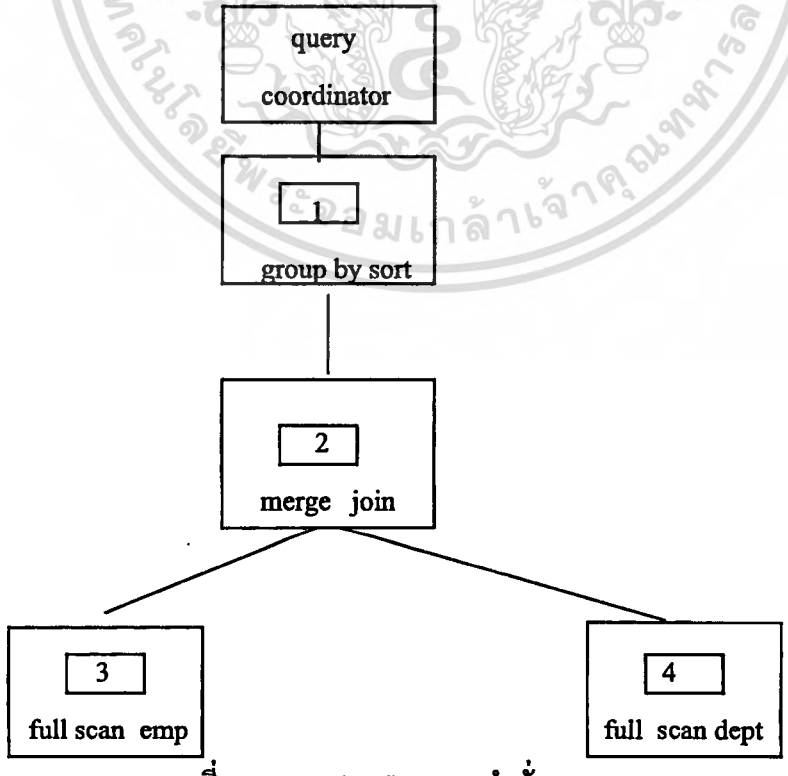
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ใน parallel query option นอกจากจะทำงานในสถาปัตยกรรมแบบขนานแล้ว ในระบบฐานข้อมูลที่มี CPUเพียง 1 ตัว ก็ยังสามารถใช้ Parallel Query ช่วยเพิ่มประสิทธิภาพการทำงานได้ เพราะ CPU ในระบบสามารถที่จะทำงานได้หลายๆ process พร้อมๆกัน เช่น ถ้าระบบมี CPUอยู่เพียง 1 ตัวแต่ว่ามีจำนวน disk อยู่ในระบบถึง 3 ตัว ภายในระบบอาจจะกำหนดจำนวน query servers อยู่ 3 process ในกรณีที่ระบบต้องการค้นหาข้อมูลในตารางใดตารางหนึ่ง ระบบก็จะมี process อยู่ 3 process ค้นหาข้อมูลจาก disk ทั้ง 3 disk พร้อมๆกัน ทำให้สามารถทำงานได้เร็วกว่าการที่มีเพียง process เดียวค้นหาตารางจาก disk ทั้ง 3 ตัว

<u>ตัวอย่าง</u> การทำงานของคำสั่ง SQL

```
select   dname, max(sal), avg(sal)
from     emp,dept
where    emp.deptno = dept.deptno
group by dname;
```

หลังจากคำสั่ง SQL ผ่านส่วนที่เป็น optimization แล้วระบบจะสามารถตรวจสอบการทำงานของคำสั่ง SQL ว่าควรที่จะใช้วิธีการ join วิธีไหนถึงจะได้ประสิทธิภาพการทำงานมากที่สุด จากนั้นในส่วนของ query coordinator จะกำหนด dataflow ของคำสั่ง SQL ดังภาพ



ภาพที่ 36 แสดง dataflow ของคำสั่ง SQL

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

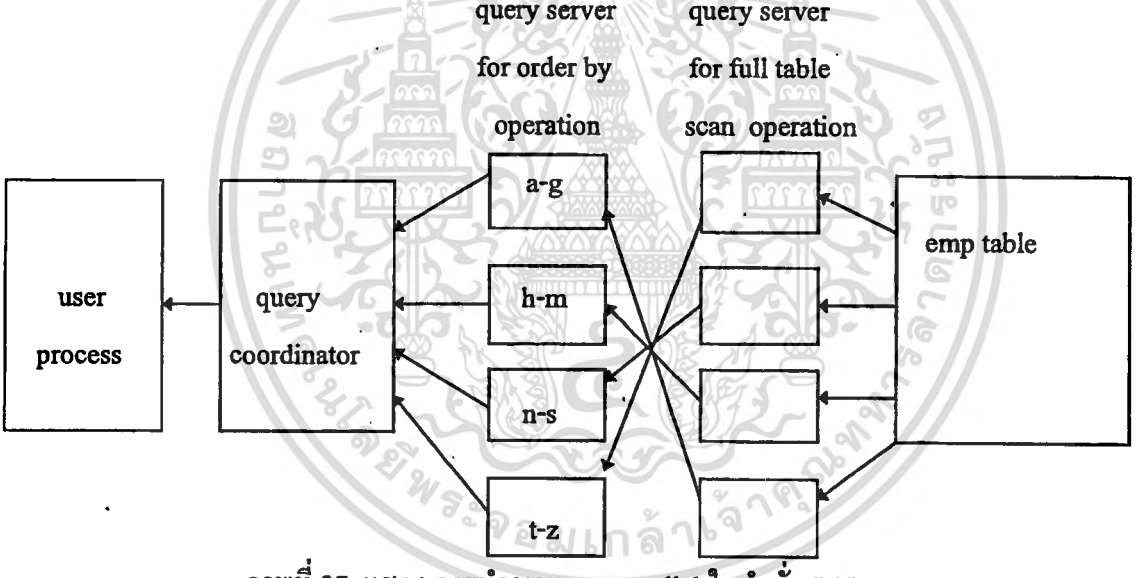
คำสั่งแบบขนานที่ใช้ในคำสั่ง SQL แบ่งออกได้เป็น 2 ประเภทตามลักษณะการทำงาน

1 intra-operation parallelism คำสั่งนี้สามารถ ทำงานได้อย่างเป็นอิสระไม่ขึ้นแก่กัน เช่น คำสั่ง scan หาตาราง , คำสั่ง join เป็นต้น

2 inter-operation parallelism คำสั่งนี้สามารถต่อเนื่องกันไปได้ได้เลย เช่น

```
select *
from emp
order by ename;
```

จากคำสั่ง SQL นี้สามารถแสดงการทำงานโดยที่การหาตาราง emp และการจัดลำดับข้อมูลใน attribute ename สามารถทำได้พร้อมๆกัน ดังภาพ



ภาพที่ 37 แสดง การทำงานแบบ parallel ในคำสั่ง SQL

จากภาพจะเห็นว่า ในระบบจะมีจำนวน query server อยู่ 4 process แต่จากภาพการทำงานของระบบจะเห็นว่าระบบจะมี 8 query server ทำงานในคำสั่งนี้เนื่องจากหลังจากที่ค้นหาตาราง emp แล้วก็จะนำข้อมูลในตารางนั้นมาทำการเรียงลำดับข้อมูลตาม ename ทันที เป็นการทำงานในแบบ inter query parallelism

6.4.3 การกำหนดจำนวน degree of parallelism

ในการกำหนดค่าของ degree of parallelism จะมีกำหนดได้ 3 รูปแบบ

1 กำหนดในขณะที่กำลัง query

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2 กำหนดในตอนสร้างตาราง

3 กำหนดเป็น parameter ใน instance

ในการกำหนดจำนวน degree of parallelism ถ้าหากว่ามีการกำหนดค่าในแต่ละตารางไม่เท่ากัน ระบบจะถือเอาค่าที่มีจำนวนมากที่สุดเป็นจำนวน degree of parallelism เช่น ถ้าหากมีการกำหนดค่าในตาราง EMP ให้เป็น 5 ,กำหนดค่าในตาราง DEPT ให้เป็น 6 ถ้าต้องการจะนำตารางทั้ง 2 มา join ระบบจะใช้จำนวน degree of parallelism ให้เป็น 6

การกำหนดจำนวน query server เมื่อ instance เริ่มทำงานเป็นการกำหนดค่าเริ่มต้น และค่าที่มากที่สุดที่ระบบสามารถมีจำนวน query servers ในระบบได้ จำนวน query server ที่oracle server จะสร้างขึ้นมาเพื่อที่จะทำงานในระบบจะถูกกำหนดที่ parameter ที่ชื่อ parallel_min_servers จำนวน query server ใน instance จะมีได้มากที่สุดจะถูกกำหนดที่ parallel_max_server เพื่อเป็นการป้องกันไม่ให้มีจำนวน process มากเกินไปทำให้กระทบกระเทือนการทำงานในส่วนอื่นๆของระบบ Parallel query option เป็นการทำให้ระบบมีประสิทธิภาพสูงขึ้น ดังนั้นก่อนที่จะมีการใช้parallel query option จะต้องมีปัจจัยที่จะต้องพิจารณา ดังนี้

1 ระบบอะไรจะเป็นระบบที่เหมาะสมในการทำงานเป็น parallel query option

2 จำนวน I/O ที่จะมาเกี่ยวข้องกับ parallel query option

3 การกำหนดจำนวน degree of parallelism จะทำที่ไหน อย่างไร

4 จำนวน query server จะมีอยู่มากเท่าไรในระบบ

Parallel query option มีความเหมาะสมที่จะใช้งานในระบบต่างๆดังนี้

1 ระบบคอมพิวเตอร์แบบขนานทั้งแบบ Symmetric multiprocessing system (SMP) และ Massively parallel processing system (MPP)

2 ระบบคอมพิวเตอร์ที่มี I/O bandwidth สูงๆ

3 ระบบคอมพิวเตอร์ที่ใช้ประสิทธิภาพของ CPUได้อย่างไม่เต็มประสิทธิภาพ

การทำ parallel query จะมีความเหมาะสมมาก ถ้าข้อมูลที่อยู่ในตารางจะกระจายออกไปยัง disk หลายๆตัวที่มีอยู่ในระบบ เพราะว่าจะได้สามารถใช้ประสิทธิภาพของ I/Oได้อย่างเต็มประสิทธิภาพ เนื่องจากในระบบฐานข้อมูลต้องการที่จะติดต่อกับ disk บ่อยๆ ดังนั้นการที่สามารถเพิ่มประสิทธิภาพในการนำข้อมูลจาก disk มาบน memory จะทำให้ประสิทธิภาพของระบบจะสูงขึ้น การกำหนดจำนวน degree of parallelism จะมีปัจจัยที่จะต้องพิจารณาดังนี้

1 จำนวนและความสามารถของ CPUในระบบ

2 ข้อจำกัดของจำนวน process ที่สามารถมีได้ในระบบ

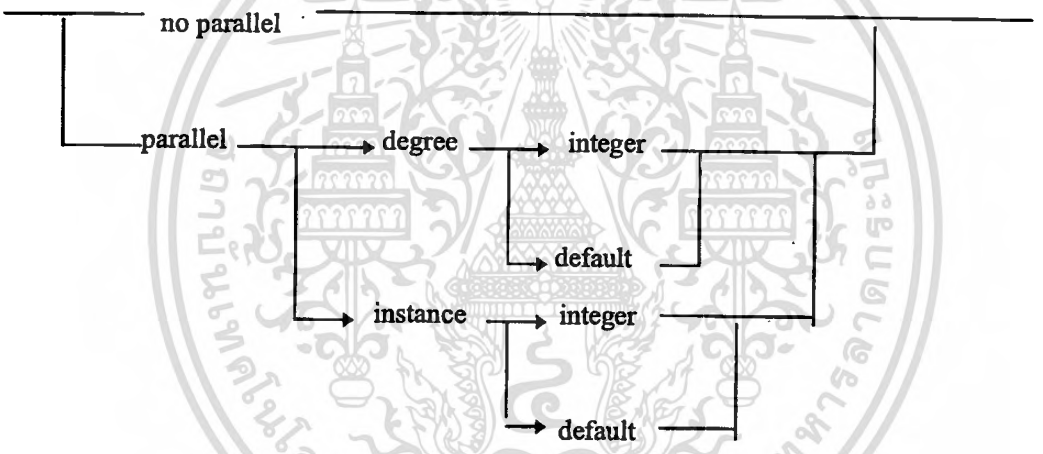
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3 ถ้าข้อมูลในตารางกระจายออกไปยังหลายๆ disk จำนวน disk ที่ข้อมูลกระจายออกไปจะเป็นปัจจัยที่จะต้องพิจารณา

การกำหนด degree of parallelism ในแต่ละquery ไม่มีหลักเกณฑ์ที่แน่นอนตายตัว ดังนั้นหลังจากที่มีการทำงานแล้วควรจะต้องมีการทดสอบประสิทธิภาพที่ได้รับและอาจจะต้องมีการปรับจำนวน degree of parallelism เพื่อที่จะทำให้ระบบมีประสิทธิภาพในการทำงานสูงที่สุด

การ set parameter กำหนดจำนวน degree of parallelism

1) กำหนดในตอนเริ่มสร้างตาราง จะเป็นการกำหนดในส่วนที่เป็น parallel clause ซึ่งสามารถเขียนโครงสร้างการทำงานได้ดังนี้



จากแผนภาพเป็นการกำหนดค่าการทำงานให้เป็น parallel ถ้ามีการกำหนดค่าไม่ให้เป็น parallel การทำงานจะเป็นแบบ serial แต่ถ้ากำหนดค่าให้เป็น parallel ระบบก็จะทำงานแบบ parallel ในการกำหนดค่าให้เป็นแบบ parallel จะมีการกำหนดค่าอยู่ 2 ค่า

1 Degree เป็นส่วนที่ใช้กำหนดว่าจะมีจำนวน degree of parallelism เท่าไรในแต่ละตาราง

1.1 integer จะกำหนดจำนวน query server ที่จะใช้งาน

1.2 default ถ้าหากว่ากำหนดเป็น default ระบบจะเช็คว่าในระบบจะมีจำนวน CPU และจำนวน disk เท่าไรนำเอาค่าที่น้อยที่สุดมาเป็นจำนวน degree of parallelism เช่น ถ้าระบบมีจำนวน CPUอยู่ 15 ตัว, มี disk อยู่ 20 disk จำนวน degree of parallelism ควรจะมีค่าเท่ากับ 15

2 instance เป็นส่วนที่ใช้กำหนดจำนวน instance เท่าไรในการทำงานกับตาราง

2.1 integer จะกำหนดจำนวน instance ที่จะใช้งาน

2.2 default จะเป็นส่วนที่บอกว่าจะใช้จำนวน instance ทั้งหมดในการทำงาน

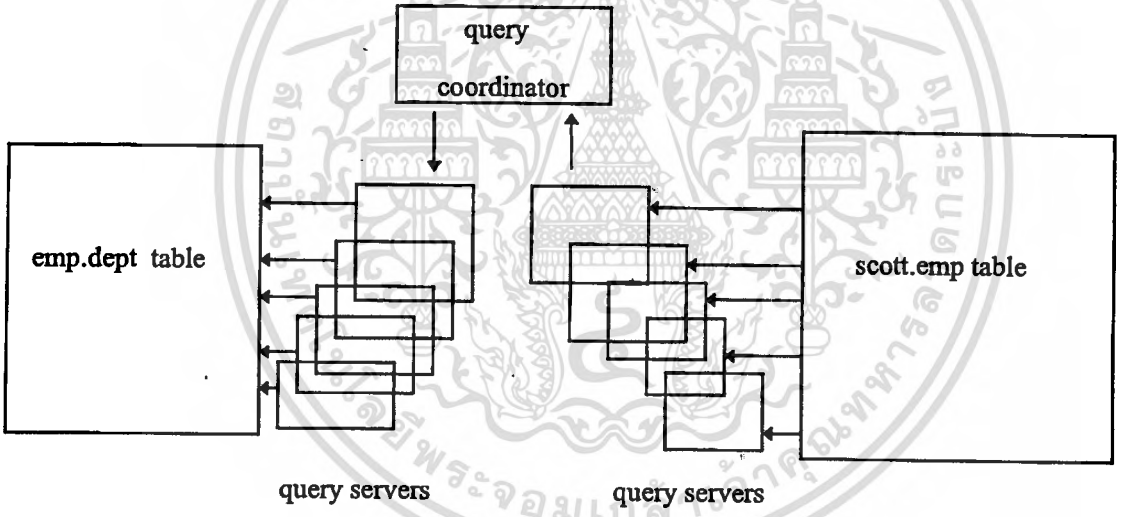
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตัวอย่าง การทำงานในแบบ parallel clause

1 ถ้าต้องการที่จะสร้างตาราง โดยการใช้ 10 query servers ,5 query server จะหาข้อมูลจากตาราง scott.emp และอีก 5 query servers จะนำข้อมูลจากตาราง scott.emp ไปไว้ที่ตาราง emp_dept การทำงานในแบบนี้เป็นการทำงานที่เรียกว่า table population

```
create table emp_dept
 parallel (degree 5)
  as  select *
   from  scott_emp
   where dept_no = 10;
```

จากคำสั่ง create table สามารถเขียนแผนภาพแสดงการทำงานได้ดังนี้



2 การสร้าง index โดยใช้ 10 query server 5 query server จะหาข้อมูลจากตาราง scott.emp ใน attribute (ename) อีก 5 query server จะนำข้อมูลจากตาราง scott_emp ไปสร้าง index ที่ชื่อ emp_idx

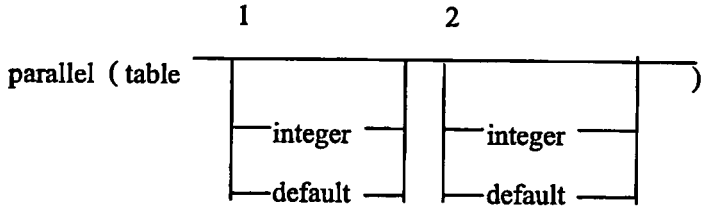
```
create index  emp_idx
  on scott_emp(ename)
    parallel 5
```

3 การเปลี่ยนแปลง query server ที่จะใช้การกับตาราง emp

```
alter table emp
  parallel(degree 9)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2) กำหนดในตอนที่ query ตาราง รูปแบบการทำงานจะเป็น



การกำหนดค่าที่เป็น parallel จะมีการกำหนดค่า 2 ค่า

ค่า 1 จะเป็นการกำหนด degree of parallelism ในแต่ละตาราง ที่จะทำ query

1.1 integer ผู้ใช้เป็นผู้กำหนดจำนวน query server

1.2 default ระบบเป็นผู้กำหนดจำนวน query server เอง

ค่า 2 กำหนดจำนวนinstance ที่จะใช้ในการทำงาน

2.1 integer ผู้ใช้เป็นผู้ กำหนดจำนวน instance ที่จะใช้งาน

2.2 default ระบบจะเป็นผู้กำหนดให้ใช้จำนวน instance ที่มีทั้งหมดในการทำงาน

<u>ตัวอย่าง</u>

```
1  select ( *+ full(scott_emp) parallel (scott_emp,5) */
             ename
           from  scott.emp,scott_emp;
```

ตัวอย่างนี้จะเป็นการกำหนดจำนวน query sever = 5 ในการทำงานกับตาราง scott_emp โดยที่ไม่ได้สนใจว่าตอนที่สร้างตาราง scott_emp จะกำหนดค่าไว้เท่าไร

```
2  select /* + full (scott.emp) parallel(scott_emp,default,default)
            ename
           from scott.emp,scott_emp;
```

ตัวอย่างนี้จะเป็นการกำหนดจำนวน query server และจำนวน instance ให้เป็น default โดยค่านี้จะทับจำนวน query server ที่กำหนดในตอนสร้างตาราง scott_emp

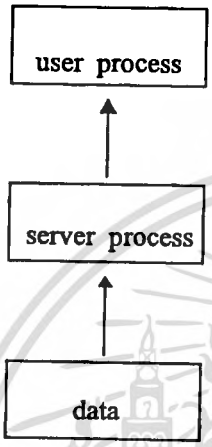
```
3  select  /* + noparallel (scott_emp) */
           ename
           from  scott.emp,scott_emp;
```

ตัวอย่างนี้จะเป็นการค้นหาตาราง scott_emp ให้เป็นแบบ serial ถึงแม้ว่าการสร้างตาราง scott_emp จะเป็นการกำหนดให้เป็น parallel ก็ตาม

6.5 สรุป

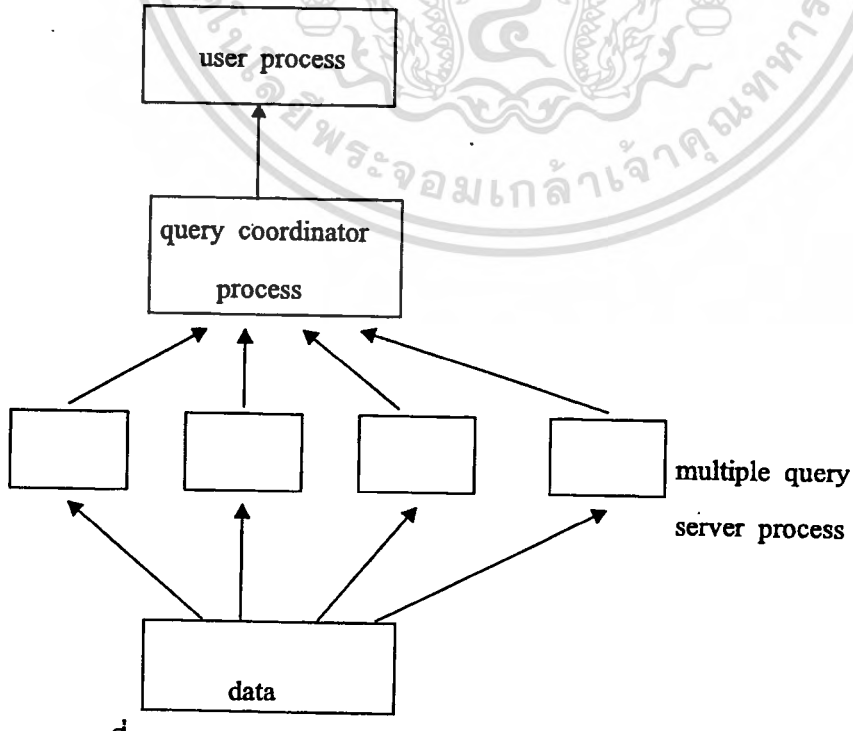
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Parallel Query Option เป็นคำสั่งที่สามารถทำให้คำสั่ง query 1 คำสั่ง กระจายออกเป็น process ย่อยๆหลายๆ process แต่ละ process สามารถที่จะทำให้ CPU หลายๆตัวที่มีอยู่ในระบบช่วยกันทำงานกับ process ย่อยๆเหล่านั้นพร้อมๆกัน ถ้าหากว่าไม่ใช้ส่วนที่เป็น Parallel Query การทำงานจะมี process 1 process มาทำงานกับคำสั่งนั้น ดังภาพ [21]



ภาพที่ 38 แสดงการทำงานแบบ serial processing

แต่ถ้าหากว่าเป็นการทำงานในส่วนของ parallel จะมี process หลายๆ process มาทำงานกับคำสั่งนั้น ดังภาพ



ภาพที่ 39 แสดงการทำงานแบบ parallel processing

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Parallel Query Option สามารถแบ่งการทำงานได้เป็น 4 ส่วนใหญ่ๆคือ [10]

1 parallel query เป็นการทำให้คำสั่ง SQL 1 คำสั่งสามารถที่จะมี process หลายๆ process ช่วยกันทำงานกับคำสั่งนั้น เช่น คำสั่ง scan, sort, join เป็นต้น

2 parallel recovery ในการกู้ข้อมูล (recovery) จาก disk เข้ามาในระบบสามารถที่จะมี process หลายๆ process ทำงานในแบบขนานได้

3 parallel load ในการนำข้อมูลจากแหล่งต่างๆเข้ามายังตาราง สามารถที่จะให้ CPUหลายๆ ตัวช่วยกันทำงานได้

4 parallel index creation ในการสร้าง index เพื่อที่จะช่วยค้นหาข้อมูลในตารางสามารถที่จะทำงานในแบบขนานได้

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

บทที่ 7
การวัดคุณสมบัติในระบบฐานข้อมูลแบบขนาน

การวัดคุณสมบัติระบบฐานข้อมูลแบบขนาน เป็นการกำหนดชุดคำสั่งที่เป็นมาตราฐาน ที่จะใช้ในการเปรียบเทียบระบบฐานข้อมูลแบบขนานหลายๆผลิตภัณฑ์ ซึ่งชุดคำสั่งดังกล่าวจะได้มาจากการวัดประสิทธิภาพในระบบฐานข้อมูล (Database benchmark)

การวัดประสิทธิภาพของระบบฐานข้อมูล เป็นการกำหนดชุดคำสั่งเพื่อการทดสอบและเปรียบเทียบ ประสิทธิภาพในระบบฐานข้อมูล 2 ระบบ หรือมากกว่าซึ่งในการเปรียบเทียบ จะต้องมีการควบคุมการทำงานของชุดคำสั่งดังกล่าว การวัดประสิทธิภาพจะใช้ hardware, software และ application มาทดสอบการทำงาน โดยการทดสอบจะใช้คำสั่งทดสอบที่เป็นมาตรฐาน เพื่อที่จะสามารถวัดประสิทธิภาพการทำงานของ hardware และ software หลายๆยี่ห้อได้ ดังนั้นคำสั่งที่ใช้ในการทดสอบอาจจะทำงานได้ดีบนระบบหนึ่ง แต่อาจจะให้ผลที่ไม่ดีในอีกระบบหนึ่งก็ได้ การวัดประสิทธิภาพสามารถแบ่งได้เป็น 2 ประเภท [10]

1 industry standard benchmark วิธีนี้เป็นการร่วมมือกันระหว่างผู้ผลิต hardware และ software ได้รวมตัวเพื่อกำหนดชุดคำสั่งที่ใช้ในการวัดประสิทธิภาพของระบบ จากผู้ผลิตหลายๆยี่ห้อ เพื่อที่จะสามารถทดสอบ การทำงานของ hardware และ software ได้อย่างมีมาตรฐานและเป็นที่ยอมรับ ในการวัดประสิทธิภาพระบบฐานข้อมูลได้มีการออกมาตรฐานที่เรียกว่า TPC (transaction processing performance council) โดยในปัจจุบันได้ออกมาตรฐานที่เป็น TPC-A,TPC-B,TPC-C และ TPC-D

2 custom benchmark เป็นการกำหนดมาตรฐานการทดสอบจากผู้วิจัย จะใช้ในการเปรียบเทียบฟังก์ชั่นการทำงานในแต่ละระบบ ในปัจจุบันผู้ผลิต hardware และ software ใช้วิธีนี้ในการทดสอบหาจุดบกพร่องในระบบของตนเอง การทดสอบในวิธีนี้เป็นการทดสอบการทำงานของคำสั่งในระบบฐานข้อมูล เช่น คำสั่ง selection คำสั่ง join เป็นต้น ดังนั้นในการทดสอบจะต้องมีการกำหนดตัวแปรที่มีผลต่อประสิทธิภาพการทำงานภายในระบบ เช่น ขนาดข้อมูล,ข้อจำกัดทางด้าน hardware ของระบบ เป็นต้น ในปัจจุบันมีวิธีการทดสอบอยู่หลายวิธี เช่น Wisconsin,AS3AP และ 007 เป็นต้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การวัดประสิทธิภาพจะต้องมีการทดสอบหลายๆอย่าง การวัดประสิทธิภาพจะต้องวัดผลจากการทำงานของคำสั่งอย่างแท้จริง ดังนั้นจะต้องมีการควบคุมไม่ให้มีสิ่งอื่นๆมากระทบทำให้ผลที่ได้เกิดความคลาดเคลื่อน เช่น

1 ข้อมูลที่ค้างอยู่ใน memory ที่เกิดจากการผลการทดสอบครั้งก่อน

2 ความเร็วของ disk และ network

3 ข้อจำกัดในด้านของจำนวนผู้ใช้ที่สามารถเข้ามาใช้ข้อมูลได้พร้อมๆกัน ซึ่งเกิดมาจากข้อจำกัดทางด้านของ hardware

4 การจัดการทางด้าน memory ภายในระบบ

5 การที่ process ไม่ยอมปล่อย resource ของระบบหลังจาก process นั้นทำงานเสร็จแล้ว

6 ขนาดของ buffer ที่ใช้ในการทำงานกับข้อมูลที่มีปริมาณมากๆ

7 การจัดการทางด้านของ disk ที่ใช้ในการเก็บข้อมูล

จุดประสงค์ในการวัดประสิทธิภาพของระบบฐานข้อมูล

1 เพื่อเปรียบเทียบประสิทธิภาพของระบบฐานข้อมูลหลายๆชนิด

2 เป็นการนำผลที่ได้ไปใช้ในการทดสอบกับปัญหาต่างๆที่ยังไม่ได้มีการทดสอบ

ในที่นี้จะศึกษาถึงการวัดประสิทธิภาพการทำงานของระบบฐานข้อมูลอยู่ 2 วิธี คือ

7.1 Wisconsin benchmark

วิธีนี้เป็นการวัดประสิทธิภาพของระบบฐานข้อมูลวิธีแรก [24] ใช้วัดประสิทธิภาพในการติดต่อข้อมูลและการ query ในแบบ relational database วิธีนี้จะวัดจากการทำงานที่มีผู้ใช้อยู่เพียงคนเดียวในระบบ คำสั่งที่ใช้ในการวัดประสิทธิภาพในวิธี Wisconsin benchmark มีอยู่ 5 คำสั่ง

7.1.1 คำสั่ง selections ในคำสั่งนี้จะมีตัวแปรที่เกี่ยวข้องกับการทำงานอยู่ 4 อย่าง คือ

1 ความเร็วของ hardware และคุณภาพของ software

2 index และ การใช้ memory ภายในระบบ

3 ผลกระทบจากการเลือกใช้ factor ที่อยู่ในระบบ

4 คำสั่ง query ที่ใช้ในการทำงาน

คำสั่ง index ที่ใช้ในการทดสอบจะมีอยู่ 3 อย่าง คือ noindex, primary index และ secondary index

7.1.2 คำสั่ง join ในคำสั่ง join จะมีตัวแปรอยู่ 2 ประการ คือ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1 ในการ join ต้องสามารถ join ได้ทั้ง 3 วิธี คือ nested-loops, sort-merge และ hashed-based join

2 การทำ optimization ในการ join

ในการทดสอบจะต้องมีการทดสอบแต่ละวิธีที่ใช้ในการ join ซึ่งจะมีอยู่ 3 วิธี และการใช้ index ซึ่งจะมีอยู่ 3 อย่างคือ noindex, primary index และ secondary index

7.1.3 คำสั่ง projection การทำ projection เป็นการกำจัดข้อมูลที่ซ้ำๆออกไป โดยในคำสั่ง SQL จะใช้คำสั่ง distinct ในการทำงาน การทำ projection จะมีขั้นตอนในการทำงานอยู่ 2 ขั้นตอน

1 เลือก attribute ที่มีข้อมูลซ้ำๆกัน

2 กำจัดข้อมูลที่ซ้ำๆกันออกไปให้เหลือเฉพาะข้อมูลที่ไม่ซ้ำ

7.1.4 คำสั่ง aggregates คำสั่งนี้เป็นคำสั่งที่ใช้ในการหาผลรวม,ค่าที่มากที่สุด,ค่าที่น้อยที่สุดของ field ที่อยู่ในระบบ ในการทดสอบการทำงานจะต้องมีการทดสอบอยู่ 3 อย่างคือ

1 ค่าที่น้อยที่สุดของข้อมูล (min)

2 ค่าที่มากที่สุดของข้อมูล (max)

3 ค่าผลรวมของข้อมูล (sum)

โดยที่การทดสอบจะมีการทดสอบ 2 รอบ ในรอบแรกจะเป็นการทดสอบกับข้อมูลที่ไม่มี index อีกระบบหนึ่งจะเป็นการทดสอบข้อมูลที่เป็น secondary index

7.1.5 คำสั่ง updates การ updates ข้อมูลภายในระบบจะต้องมีการทดสอบอยู่ 4 อย่างคือ

1 การเพิ่มข้อมูล 1 แถวเข้ามาในระบบ (insert)

2 การเปลี่ยนแปลงข้อมูล 1 แถวโดยเป็นข้อมูลของ primary key (update)

3 การเปลี่ยนแปลงข้อมูล 1 แถวโดยเป็นข้อมูลของ non key

4 การลบข้อมูล 1 แถว (delete)

7.2 AS3AP (Ansi SQL standard and portable) benchmark

วิธีนี้เป็นการขยายความสามารถของวิธี Wisconsin โดยวิธีนี้จะวัดประสิทธิภาพการทำงาน [23] ในรูปแบบที่มีผู้ใช้งานอยู่คนเดียว (single user) และรูปแบบที่มีผู้ใช้งานอยู่หลายๆคน (multiuser) การทดสอบการทำงานจะมีการทดสอบดังนี้

1 ในรูปแบบที่มีผู้ใช้อยู่คนเดียว (single user) คำสั่งที่ใช้ในการทดสอบการทำงานนอกจากจะเหมือนกับในการทดสอบของวิธี Wisconsin ยังมีการเพิ่มคำสั่งอีก คือ การ loading, backup,

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

restore, recovery และ build index โดยที่การทดสอบในวิธีนี้จะวัดประสิทธิภาพการทำงาน ได้จากเวลาที่ใช้ในการทำงานและจำนวนผลลัพธ์ที่ได้

2 ในแบบที่มีผู้ใช้อยู่หลายๆคน (multiuser) ในรูปแบบนี้จะมีการทดสอบการทำงานอยู่ 3 อย่าง

2.1 ระบบสามารถรองรับผู้ใช้ให้สามารถเข้ามาแก้ไขข้อมูลในตารางเดียวกัน พร้อมๆกันได้กี่คน

2.2 ระบบสามารถรองรับผู้ใช้ให้เข้ามาใช้ข้อมูลพร้อมๆกันได้กี่คน

2.3 ระบบสามารถรองรับผู้ใช้ให้เข้ามาใช้และแก้ไขข้อมูลพร้อมๆกันได้กี่คน

การทดสอบในวิธีนี้ เป็นการทดสอบประสิทธิภาพการทำงานของระบบ ในการรองรับผู้ใช้ที่จะเข้ามาทำงานภายในระบบพร้อมๆกัน (concurrency users)

การวัดคุณสมบัติในระบบฐานข้อมูลแบบขนาน จะแบ่งคำสั่งที่ใช้ในการวัดคุณสมบัติออกเป็น 2 ประเภทคือ

1 คำสั่งทั่วๆไปที่ใช้ในการวัดประสิทธิภาพ เช่น คำสั่ง scan, join, aggregate เป็นต้น คำสั่งพวกนี้จะปรากฏอยู่ในวิธีการวัดประสิทธิภาพของระบบฐานข้อมูล

2 คำสั่งพิเศษซึ่งจะเป็นคำสั่งที่จะใช้ในระบบฐานข้อมูลแบบขนานโดยเฉพาะ คือ คำสั่ง data partition และคำสั่ง load balancing

ในที่นี้จะเปรียบเทียบประสิทธิภาพระบบฐานข้อมูลแบบขนานอยู่ 3 ผลิตภัณฑ์คือ

1 Oracle 7.3

2 Sybase Adaptive Server Enterprise Release 11.5

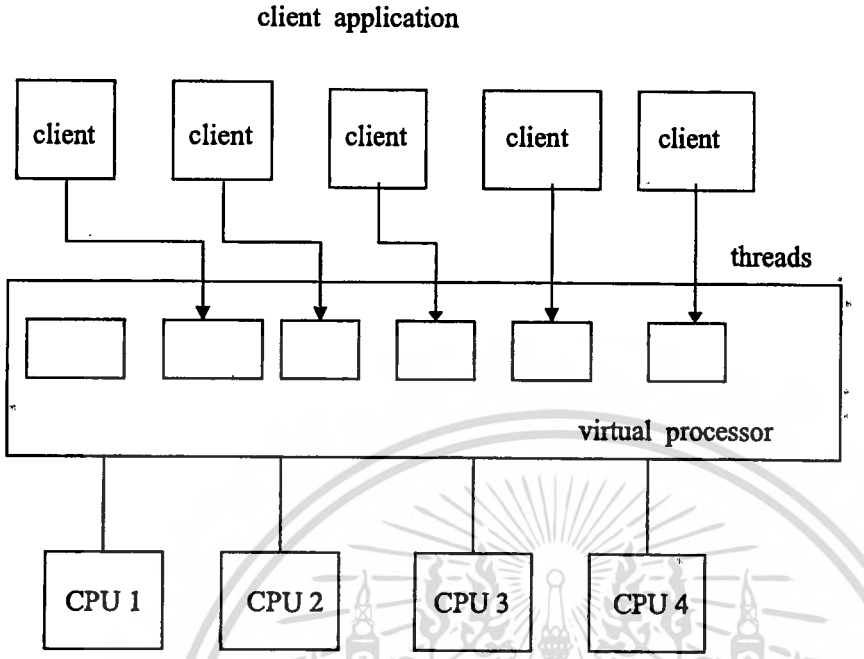
3 Informix Online Extended Parallel Server 8.1

โดยในที่นี้จะกล่าวถึงสถาปัตยกรรมของ Sybase และ Informix ที่มีการทำงานในแบบขนาน

7.3 Informix

Informix สามารถทำงานในแบบขนานโดยใช้เทคโนโลยีที่เรียกว่า Dynamics Scalable Architecture (DSA) โดยในการทำงานจะมี virtual processor (VP) ซึ่งจะเป็น server processes ที่คอยรับคำสั่ง query จากผู้ใช้เข้ามาทำงานในระบบ ในการทำงานสามารถแสดงภาพการทำงานได้ดังนี้ [19]

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 40 แสดงสถาปัตยกรรมของ Dynamics Scalable Architecture (DSA)

thread คือ sub processes หรือ lightweight processes เป็นprocess ย่อยๆที่อยู่ในuser process การทำงานของ virtual processor เป็นการทำงานที่เรียกว่า multithreaded processes คือระบบสามารถที่จะทำงานกับ thread หลายๆ thread พร้อมๆกันซึ่งจะเป็นการทำงานที่เรียกว่า multiple concurrent thread (MCT) โดยที่แต่ละ virtual processor สามารถทำงานกับ thread เพียง 1 thread ในเวลาใดเวลาหนึ่ง การที่ virtual processor สามารถทำงานกับ thread ได้หลายๆ thread พร้อมๆกันจะใช้การ switching ระหว่าง thread โดยที่ virtual processor แต่ละตัวจะรับ thread เข้ามาทำงานภายในระบบ เมื่อทำงานกับ thread ใด thread หนึ่งเสร็จ virtual processor สามารถที่จะ switch ไปทำงานกับอีก thread หนึ่งได้ทันที หรือถ้าหากว่า thread นั้นกำลังติดต่อข้อมูลที่เก็บอยู่บน disk virtual processor สามารถที่จะนำอีก thread หนึ่งเข้ามาทำงานได้ทันทีและรอให้ระบบสามารถนำข้อมูลจากdisk เข้ามายัง memory ได้ก่อน virtual processor ถึงจะ switch มาทำงานกับ thread นั้นอีกครั้ง การใช้ virtual processor ในระบบจะมีข้อดีดังนี้

1 เป็นการประหยัด memory และ resource การทำ multithreading เป็นการใช้ resource อย่างมีประสิทธิภาพมากเนื่องจากสามารถใช้ virtual processor ร่วมกันได้นอกจากนี้การ switching ระหว่าง thread ยังทำได้รวดเร็วมากกว่าการ switch ระหว่าง process ซึ่งจะทำให้การทำงานทำได้อย่างรวดเร็วมากขึ้น และมีประสิทธิภาพยิ่งขึ้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

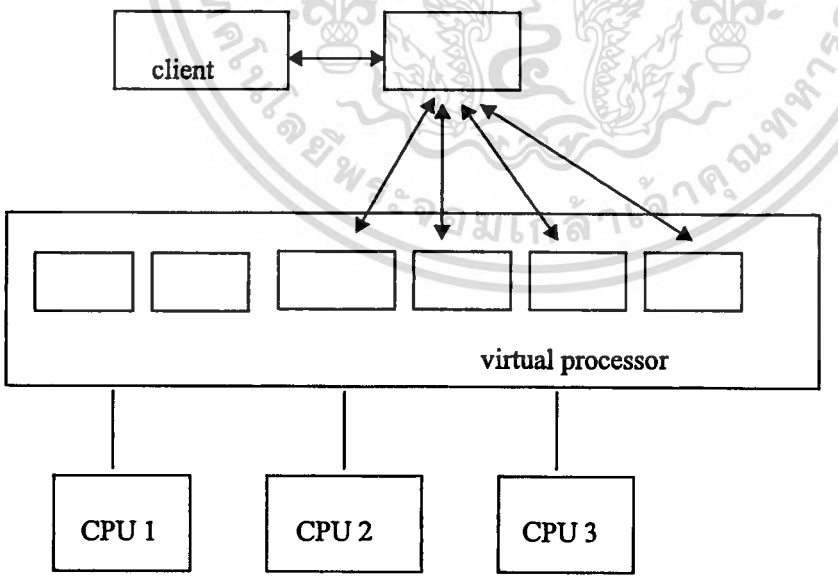
2 สามารถทำงานที่เป็น parallel processing ได้ โดยการใช้ virtual processor หลายๆ process ทำงานกับ thread ใด thread หนึ่งซึ่งจะทำให้ thread นั้นสามารถทำงานได้อย่างรวดเร็วมากยิ่งขึ้น

3 สามารถที่จะเพิ่มหรือลดจำนวน virtual processor ได้ ถึงแม้ว่าระบบกำลังทำงานอยู่ ทำให้สามารถกำหนดจำนวน virtual processor ให้เหมาะสมกับงานที่กำลังทำอยู่และสามารถปรับจำนวน virtual processor ให้เหมาะสมกับทรัพยากรของระบบที่มีอยู่ได้

4 virtual processor จะไม่ขึ้นอยู่กับ thread ใด thread หนึ่งโดยเฉพาะ ถ้าหากว่า CPU ใดเกิดว่างCPUนั้นสามารถที่จะรับ thread ที่เข้ามาในระบบเข้าไป execute ได้ทันที

โดยปกติการกำหนดจำนวน virtual processor จะต้องกำหนดให้มากกว่าจำนวน CPU ที่มีอยู่ในระบบ เพื่อที่จะได้สามารถรองรับงานที่มีจำนวนมากได้ นอกจากนี้ยังเป็นการกระจายงานไปยัง CPUที่มีอยู่หลายๆตัวในระบบได้ และลดส่วนคอขวดของระบบได้ ถ้ามีงานจาก VP เข้ามายัง CPU ตัวใดที่ยังว่างอยู่ CPU ตัวนั้นสามารถที่จะนำเอางานนั้นไปทำได้ทันทีโดยที่ CPU จะไม่ผูกติดอยู่กับงานใดงานหนึ่งโดยเฉพาะ

ในการทำงานแบบ parallel จะมี virtual processor หลายๆ processes เข้ามาทำงานกับ thread ที่สามารถทำงานในแบบขนานได้ ดังภาพ

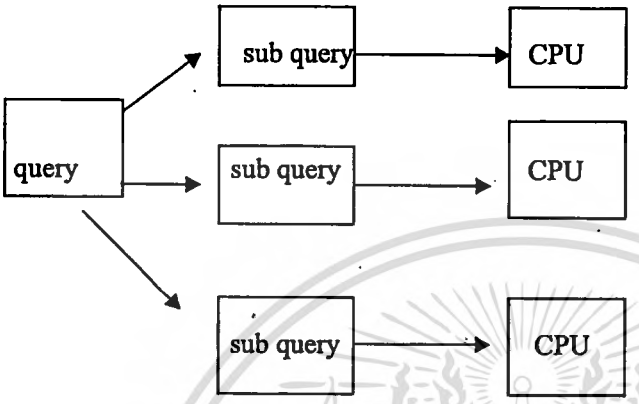


ภาพที่ 41 แสดงการทำงานแบบขนานของ Informix

PDQ (Parallel Data Query) เป็นเทคนิคที่ช่วยกระจายคำสั่ง SQL 1 คำสั่ง ออกไปหลายๆ CPU เพื่อให้ CPU หลายๆตัวช่วยกันทำงานกับคำสั่งนั้นพร้อมๆกันเป็นการทำงานในแบบขนาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การทำงานของ PDQ จะเรียกว่า PDQ query เมื่อคำสั่ง query 1 คำสั่งเข้ามาในระบบ PDQ จะแบ่งคำสั่งออกเป็นหลายๆ sub query ย่อยๆ แต่ละsub query จะถูกทำงานโดย CPU แต่ละตัวพร้อมๆ กันดังภาพ



ภาพที่ 42 แสดงการทำงานของ Parallel Data Query

ใน PDQ จะกำหนดจำนวน degree of parallelism หรือจำนวน virtual processors หลายๆ process ที่จะทำงานกับ sub query แต่ละ sub query พร้อมๆกันเป็นการทำงานในแบบขนาน คำสั่งที่สามารถทำงานในรูปแบบขนานได้จะมีอยู่ 7 คำสั่ง

1 Parallel scan เป็นการที่ใช้ virtual processor หลายๆprocess ค้นหาข้อมลจาก disk ที่มีอยู่ในระบบโดยที่ข้อมูลอาจจะกระจายอยู่ใน disk ใด disk หนึ่งหรืออาจจะกระจายออกไปยังหลายๆ disk ก็ได้

2 Parallel sort ในการเรียงลำดับข้อมูลจากน้อยไปมาก (ascending) หรือจากมากไปน้อย (descending) สามารถที่จะทำงานในแบบขนานได้ โดยการแบ่งข้อมูลที่จะเรียงลำดับออกเป็นส่วนๆ แต่ละส่วนจะใช้ virtual processor แต่ละprocess จัดการเรียงลำดับข้อมูลก่อนที่จะนำข้อมูลมารวมกัน

3 Parallel join ในการ join ตาราง 2 ตารางหรือมากกว่า ระบบสามารถแบ่งแบ่งข้อมูลที่จะ join ออกเป็นส่วนๆ แต่ละส่วนจะถูกทำงานโดย virtual processor แต่ละ process ทำให้สามารถทำงานได้อย่างรวดเร็วมากยิ่งขึ้น

4 Parallel Aggregation ในการทำงานกับคำสั่งบางคำสั่ง เช่น การหาค่า average ของตาราง, การหาค่า minimum ของตารางหรือการหาค่าผลรวม สามารถใช้ virtual processor หลายๆ process ช่วยกันทำงานเป็นแบบขนานได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5 Parallel insert ในการเพิ่มข้อมูลใหม่เข้าไปในตารางสามารถทำงานในแบบขนานได้ เนื่องจากข้อมูลในตารางอาจจะกระจายไปในหลายๆ disk ทำให้สามารถใช้ virtual processor หลายๆ process ทำงานกับข้อมูลในแต่ละ disk ได้

6 Parallel delete ในการลบข้อมูลที่อยู่ในตารางสามารถทำงานในแบบขนานได้ เช่นเดียวกับการเพิ่มข้อมูลเข้าไปในตาราง

7 Parallel index building ในการสร้าง index ที่ช่วยในการค้นหาข้อมูลที่เก็บอยู่ในตาราง สามารถทำงานโดยใช้ virtual process หลายๆ process ทำงานในแบบขนานสร้าง index นั้นได้

การทำ data partition ระบบสามารถที่จะแบ่งข้อมูลที่อยู่ในตารางออกไปหลายๆ disk ที่มีอยู่ในระบบได้ ซึ่งในการแบ่งข้อมูลออกไปยัง disk ต่างๆ จะมีวิธีการแบ่งได้หลายๆวิธีคือ

1 Round-Robin partition

2 Range partition

3 Expression partition วิธีนี้เป็นการขยายการทำงานของวิธี range partition

4 Hash partition

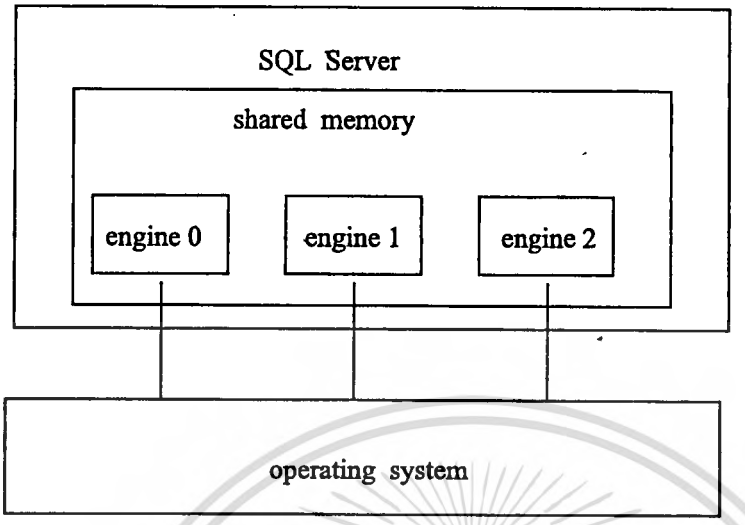
5 Hybrid partition วิธีนี้สามารถใช้วิธีการแบ่งข้อมูลหลายๆวิธีมารวมกัน เช่น อาจจะใช้วิธี range partition กับ hash partition มารวมกันเพื่อกระจายข้อมูลในตารางใดตารางหนึ่ง

ระบบยังสามารถทำ backup, restore และ recovery โดยใช้ virtual processor หลายๆ process ทำงานร่วมกันในแบบขนานได้ นอกจากนี้ระบบยังสามารถ loading data ให้สามารถทำงานในแบบขนานได้อีกด้วย

7.4 Sybase

ใน Sybase สามารถทำงานในแบบขนานได้โดยใช้สถาปัตยกรรมที่เรียกว่า Virtual Server Architecture (VSA) รูปแบบนี้จะมีการทำงานที่เรียกว่า multithreaded โดยระบบจะมี engine คอยจัดการคำสั่งที่เข้ามาทำงานภายในระบบโดยที่ engine จะหมายถึง server processes ที่จะ execute คำสั่งที่เข้ามาทำงานภายในระบบใน Sybase จะเรียก thread ว่า task โดยสถาปัตยกรรมของ Sybase Adaptive Server Enterprise จะสามารถแสดงได้ดังภาพ [4]

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 43 แสดงสถาปัตยกรรมของ Virtual Server Processor (VSA)

สถานะของ task ที่จะทำงานอยู่บนระบบจะมีอยู่ 3 สถานะ

1 running เป็นสถานะที่ task ถูก execute อยู่บนระบบ

2 runnable queue เป็นสถานะที่ task ไม่ได้ทำงาน แต่กำลังรอการ synchronize ระหว่าง task

3 sleeping เป็นสถานะที่ task หยุดทำงาน เนื่องจากกำลังรอ resource ในระบบอยู่

เมื่อ user ส่ง process เข้ามาทำงานภายในระบบ การทำงานภายในระบบจะมีดังนี้

1 client ส่ง process เข้ามา ซึ่ง process นั้นจะถูกส่งเข้ามาที่ adaptive server engine

2 adaptive server จะตรวจสอบดูว่า process ที่ส่งเข้ามาสามารถแบ่งออกได้เป็นกี่ task

3 adaptive server จะทำการ optimize จำนวน taskที่มีอยู่

4 ถ้ามีการกำหนดให้มีการทำงานเป็นแบบขนาน adaptive server จะกำหนดจำนวน worker processes หลายๆ process มาทำงานกับ task นั้น

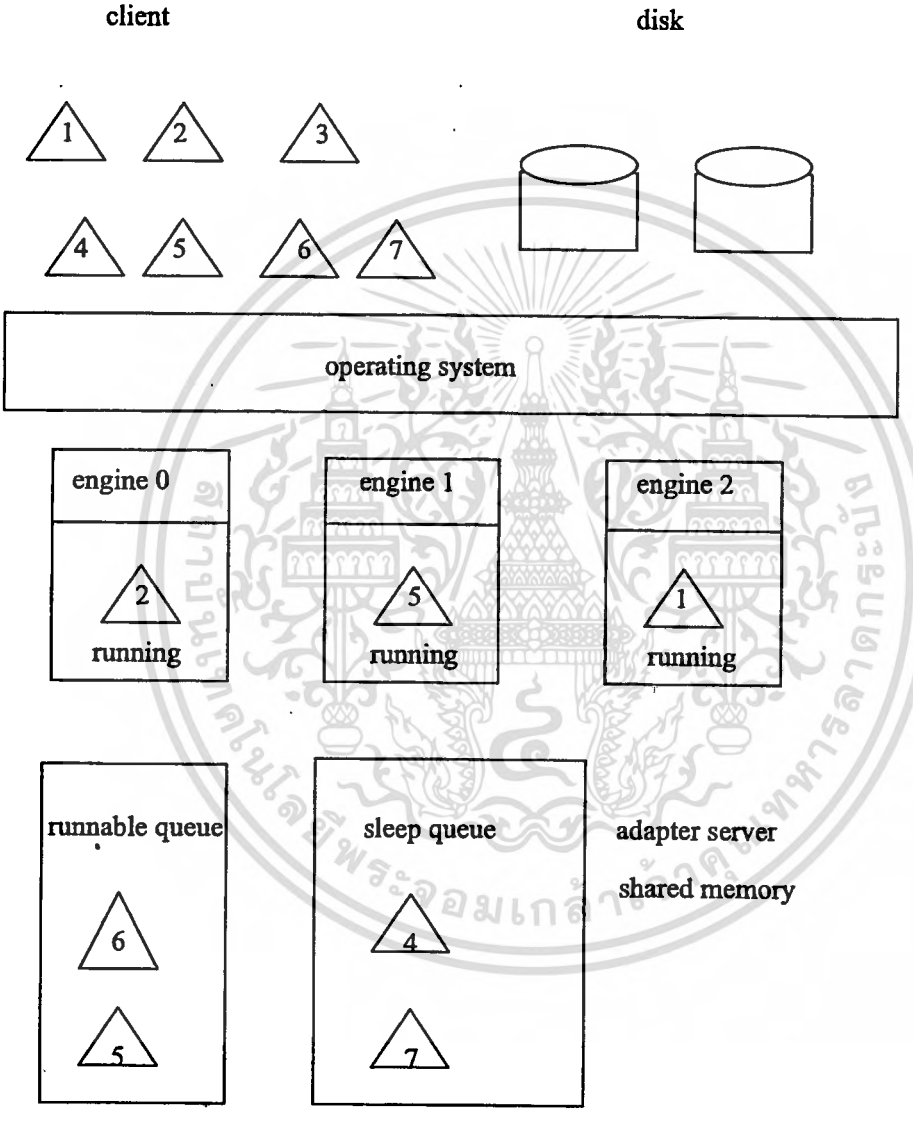
5 adaptive server จะ execute task จากกนั้นจะส่งผลลัพธ์ที่ได้กลับไปที่ user ต่อไป

ในการทำงานที่เป็น multithreaded ระบบจะใช้การทำงานที่เป็น context switching โดยการนำ task เข้ามาทำงานภายใน engine ถ้าหากว่า task นั้นต้องการทำงานบางอย่าง ยังไม่สามารถที่จะexecute ได้ engine จะสลับนำเอาอีก task หนึ่งเข้ามาทำงานแทน ซึ่งก็จะเป็นการสลับกันเข้ามาทำงานภายใน engine นั้น

ในรูปแบบที่เป็น Symmetric multiprocessing (SMP) system จะมีจำนวน CPU อยู่หลายๆตัวในระบบ ดังนั้น จำนวน engine ในระบบสามารถมีได้หลายๆ engine แต่ว่าจำนวน engine จะต้อง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

มีจำนวนที่ไม่มากกว่าจำนวน CPU ที่มีอยู่ในระบบ เมื่อ user ส่ง process เข้ามาทำงานภายในระบบ adaptive server จะแบ่ง process ให้ออกเป็น subprocess หรือว่า task จากนั้น engine ที่มีอยู่หลายๆ ตัวในระบบจะนำเอา task เหล่านั้นเข้ามาทำงานภายใน engine



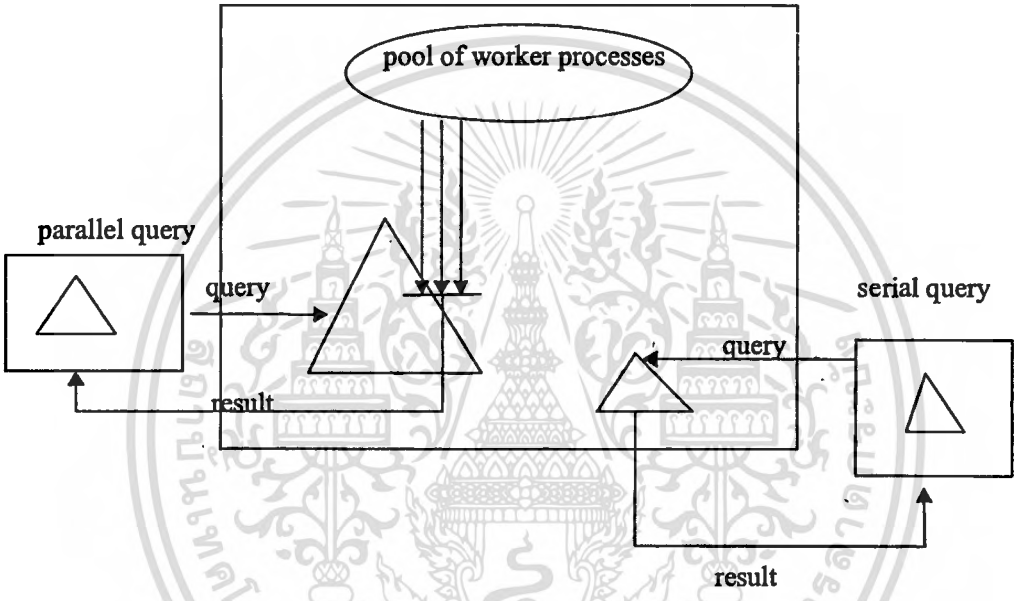
ภาพที่ 44 แสดงการทำงานของ Virtual Server Architecture (VSA)

ใน user process จะมีจำนวน task อยู่หลายๆ task ขึ้นอยู่ กับความซับซ้อนของคำสั่งที่เข้ามาทำงานภายในระบบ แต่ละ task จะถูก engine นำ task เข้าไป execute ถ้าหากว่า task ไหนไม่มีการ synchronize ก็จะถูกรออยู่ที่ runnable queue และถ้า task ไหนยังไม่สามารถที่จะทำงานได้เนื่องจากต้องรอการนำข้อมูลจาก disk เข้ามาใน memory จะมีการนำ task นั้นเข้าไปไว้ใน sleep

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

queue engine จะไม่ขึ้นอยู่กับ CPUใด CPU หนึ่งโดยเฉพาะ ถ้า CPU ใดในระบบว่างก็สามารถที่จะนำ task ที่อยู่ใน adaptive server เข้ามาทำงานภายใน engine ได้

ในการทำงานแบบขนานระบบจะเช็คดูว่า task ไหนที่สามารถทำงานในแบบขนานได้ ถ้าพบว่ามี task นั้นอยู่ ระบบจะมีจำนวน worker processes ที่เป็น process หลายๆ process มา execute คำสั่งนั้นพร้อมๆกัน ใน engine แต่ถ้าหากว่าเป็นการทำงานแบบ serial task ก็จะทำงานบน process เพียง process เดียว ดังภาพ



ภาพที่ 45 แสดงการทำงานแบบขนานของ Sybase

degree of parallelism เป็นการกำหนดจำนวน worker processes ที่จะทำงานกับคำสั่ง query ที่จะเข้ามาทำงาน ใน Sybase SQL Server จะมีคำสั่งที่สามารถทำงานในแบบขนานได้ 2 ประเภท คือ

1 Parallel querying จะมีคำสั่ง scan, join และคำสั่ง aggregates

2 Parallel utilites จะมีคำสั่ง sort, index creation, backup, recovery, insert, update, delete และคำสั่ง loading

ในการทำdata partition Sybase จะมีการทำงานอยู่ 3 วิธีคือ

1 hash partition

2 range partition

3 schema partition

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

7.5 การเปรียบเทียบคุณสมบัติและการให้คะแนน

สถาปัตยกรรมของ Oracle จะทำงานในรูปแบบ multiprocessing parallel processing ในขณะที่ Sybase และ Informix จะทำงานในรูปแบบของ multithreaded parallel processing เราสามารถเปรียบเทียบประสิทธิภาพการทำงาน ของระบบฐานข้อมูลแบบขนานแยกเป็นผลิตภัณฑ์ตามตารางได้ดังนี้

คำสั่ง	product 1	product 2	product 3
1 selection	สามารถที่จะ scan หาข้อมูลจาก disk โดยใช้ process หลายๆ process ทำงานร่วมกันเป็นแบบขนานได้	สามารถที่จะทำงานโดยใช้ worker processes หลายๆ processes นำงานรวมกันเป็นแบบขนานได้	สามารถที่จะใช้ virtual processor (VP) หลายๆprocess ช่วยกันทำงานได้
2 join	สามารถที่จะแบ่งข้อมูลออกเป็นส่วนๆ แต่ละส่วนจะทำงานโดยใช้ process แต่ละprocess พร้อมๆกัน	สามารถที่จะใช้ worker processes หลายๆ process แบ่งข้อมูลที่จะ joinออกเป็นส่วนๆ แต่ละส่วนจะถูก process ช่วยกันทำงาน พร้อมๆกัน	สามารถที่จะใช้ virtual processor หลายๆ process ช่วยกันjoinข้อมูลที่มีอยู่ในmemory พร้อมๆกันได้
3 projection	ไม่สามารถทำงานกับคำสั่งนี้ในแบบขนานได้	สามารถที่จะทำงานกับคำสั่งนี้ในแบบขนานได้	ไม่สามารถที่จะทำงานในแบบขนานได้
4 aggregate (sum, min, max ,avg)	ไม่สามารถที่จะทำงานกับคำสั่งนี้ในแบบขนานได้	สามารถที่จะทำงานกับคำสั่งนี้ในแบบขนานได้	สามารถทำงานกับคำสั่งนี้ในแบบขนานได้
5 update ข้อมูล (insert, delete, update)	ไม่สามารถที่จะทำงานกับคำสั่งนี้ในแบบขนานได้	สามารถที่จะทำงานกับคำสั่งinsert, delete และ update ในแบบขนานได้	สามารถที่จะทำงานแบบขนานกับคำสั่ง insert และ delete

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6 loading	สามารถทำงานกับคำสั่งนี้ในแบบขนานได้	สามารถที่จะทำงานคำสั่งนี้ในแบบขนานได้	สามารถทำงานกับคำสั่งนี้ในแบบขนานได้
7 utility (backup /restore/ recovery)	สามารถทำงานกับคำสั่ง recovery ในแบบขนานได้	สามารถที่จะทำงานกับคำสั่ง backup กับ restore ในแบบขนานได้	สามารถที่จะทำงานกับคำสั่ง backup กับ recovery ในแบบขนานได้
8 build index	สามารถที่จะสร้าง index โดยใช้การทำงานเป็นแบบขนานได้	สามารถที่จะทำงานกับคำสั่งนี้ในแบบขนานได้	สามารถที่จะทำงานกับคำสั่งนี้ในแบบขนานได้
9 data partition	ไม่สามารถที่จะกระจายข้อมูลในตารางออกไปยัง disk ต่างๆ ได้ด้วยตนเอง ต้องอาศัย OS utility ช่วยในการทำงานโดยจะกระจายข้อมูลแบบ range partition	สามารถที่จะกระจายข้อมูลออกไปยัง disk ต่างๆ โดยในการกระจายข้อมูลจะใช้วิธี hash, range และ schema partition	สามารถที่จะกระจายข้อมูลออกไปยัง disk ต่างๆโดยใช้วิธี round-robin, range, hash, expression และ hybrid partition
10 sorting	สามารถที่จะ sort ข้อมูลในแบบขนานได้	สามารถที่จะทำงานในแบบขนานได้	สามารถที่จะทำงานในแบบขนานได้
11 load balancing	ไม่มีคำสั่งการทำ load balancing ในระบบ	สามารถทำงานกับคำสั่ง load balancing โดยใช้วิธี static load balancing	สามารถทำงานกับคำสั่ง load balancing โดยใช้วิธี dynamics load balancing

ตารางที่3 แสดงการเปรียบเทียบความสามารถในการทำงานแต่ละผลิตภัณฑ์

จากตารางการเปรียบเทียบการทำงานของระบบฐานข้อมูลแบบขนานทั้ง 3 ประเภท เราสามารถนำตารางมาวิเคราะห์และเปรียบเทียบผลิตภัณฑ์แต่ละตัว เพื่อที่จะให้คะแนนความสามารถในการทำงาน โดยที่คำสั่งแต่ละคำสั่งจะมีคะแนนไม่เท่ากันขึ้นอยู่กับ ความสำคัญของคำสั่งและความถี่ที่คำสั่งนั้นจะถูกใช้งาน โดยที่คำสั่งไหนที่มีความสำคัญ และถูกใช้งานบ่อยๆก็จะให้คะแนนคำสั่งนั้น 10 คะแนน แต่ถ้าหากว่าคำสั่งไหนที่ไม่ถูกใช้งานบ่อยมากนัก ก็จะให้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

คะแนนคำสั่งนั้นเป็น 5 คะแนน เราสามารถให้คะแนนผลิตภัณฑ์แต่ละประเภทแยกเป็นคำสั่งได้ดังนี้

คำสั่ง	product 1	product 2	product 3
1 selection (10 คะแนน)	10	10	10
2 join (10 คะแนน)	10	10	10
3 projection (5 คะแนน)	0	5	0
4 aggregate (10 คะแนน)	0	10	10
5 update (10 คะแนน)	0	10	7
6 loading (5 คะแนน)	5	5	5
7 utility (10 คะแนน)	3	7	7
8 build index (10 คะแนน)	10	10	10
9 data partition (10 คะแนน)	5	10	10
10 sorting (10 คะแนน)	10	10	10
11 load balancing (5 คะแนน)	0	4	5
รวม (95 คะแนน)	53	91	84

ตารางที่ 4 แสดงการให้คะแนนความสามารถในการทำงานแต่ละผลิตภัณฑ์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โดยคำสั่งส่วนใหญ่จะมีคะแนนเป็น 10 คะแนน เนื่องจากเป็นคำสั่งที่มีความสำคัญและมีการใช้งานอย่างมาก ในระบบฐานข้อมูล มีคำสั่ง projection, loading และคำสั่ง load balancing ที่ได้ 5 คะแนน เนื่องจากโดยปกติจะไม่ค่อยได้ใช้คำสั่งนี้ จากตารางการให้คะแนนผลิตภัณฑ์ เราสามารถอธิบายการให้คะแนนในแต่ละคำสั่งได้ดังนี้

1 คำสั่ง selection ทั้ง 3 ผลิตภัณฑ์ ต่างก็สามารถที่จะหาข้อมูลในแบบขนานได้ โดยข้อมูลที่ต้องการหาอาจจะเก็บไว้ต่าง disk โดยที่ทั้ง 3 ผลิตภัณฑ์ สามารถที่จะหาข้อมูลในตารางเดียวกันแต่เก็บไว้ต่าง disk ได้พร้อมๆกัน ดังนั้นจึงให้คะแนนเต็มทั้ง 3 ผลิตภัณฑ์

2 คำสั่ง join ทั้ง 3 ผลิตภัณฑ์ สามารถที่จะแบ่งข้อมูลที่จะ join ออกเป็นส่วนๆ แต่ละส่วนจะถูกทำงานโดย CPUในแต่ละส่วนพร้อมๆกันเป็นแบบขนาน ดังนั้นจึงได้คะแนนเต็มทั้ง 3 ผลิตภัณฑ์

3 คำสั่ง projection ใน product 2 สามารถทำงานกับคำสั่ง distinct เพื่อที่จะแยกข้อมูลที่ซ้ำๆออกมาในแบบขนานได้ แต่ว่าใน product 1 และ product 3 ไม่สามารถที่จะทำงานกับคำสั่งนี้ในแบบขนานได้ ดังนั้นใน product 1 และ product 3 จึงไม่มีคะแนนส่วน product 2 ได้คะแนนเต็มสำหรับคำสั่งนี้

4 คำสั่ง aggregate ใน product 1 ไม่สามารถทำงานกับคำสั่งนี้ในแบบขนานได้ แต่ว่า product 2 และ product 3 สามารถทำงานกับคำสั่งนี้ในแบบขนานได้ ดังนั้นในคำสั่งนี้ product 1 จึงไม่ได้คะแนน แต่ว่า product 2 กับ product 3 ได้คะแนนเต็มสำหรับคำสั่งนี้

5 คำสั่ง update (insert, delete, update) ใน product 1 ไม่สามารถทำงานกับคำสั่งนี้ในแบบขนานได้ product 2 สามารถทำงานทั้ง 3 คำสั่งในแบบขนานได้ ส่วน product 3 สามารถทำงานกับคำสั่ง insert และdelete ในแบบขนานได้ ส่วนคำสั่ง update ไม่สามารถทำงานในแบบขนาน ดังนั้นในคำสั่งนี้ product 1 จึงไม่ได้คะแนน product 2 ได้คะแนนเต็ม ส่วน product 3 ได้ 7 คะแนน

6 คำสั่ง loading ทั้ง 3 ผลิตภัณฑ์สามารถที่จะ loadข้อมูลเข้ามายังตารางโดยที่สามารถที่จะทำงานในแบบขนานได้ ดังนั้นจึงได้คะแนนเต็มทั้ง 3 ผลิตภัณฑ์

7 คำสั่ง utility (backup, restore, recovery) product 1 สามารถทำงานกับคำสั่ง recovery ในแบบขนานได้เพียงคำสั่งเดียว ดังนั้นจึงได้คะแนนไป 3 คะแนน product 2 สามารถทำงานกับคำสั่ง backup และ restore ในแบบขนานได้ จึงได้คะแนนไป 7 คะแนน product 3 สามารถทำงานกับคำสั่ง backup และrecovery ในแบบขนานได้ ดังนั้นจึงได้คะแนนไป 7 คะแนน

8 คำสั่ง build index ทั้ง 3 ผลิตภัณฑ์สามารถที่จะทำงานกับคำสั่งนี้ในแบบขนานได้ ดังนั้นจึงได้คะแนนเต็มทั้ง 3 ผลิตภัณฑ์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

9 คำสั่ง data partition product 1 ไม่สามารถกระจายข้อมูลในตารางออกไปหลายๆ disk ได้ ต้องใช้ OS utility เป็นส่วนที่ใช้ในกระจายข้อมูล และต้องใช้การทำ data partition แบบ range partition ด้วย ดังนั้นจึงได้คะแนนไป 5 คะแนน ส่วน product 2 สามารถที่จะกระจายข้อมูลได้ด้วยตนเองโดยสามารถที่จะกระจายข้อมูลได้ 3 วิธีคือ range partition, hash partition, schema partition จึงได้คะแนนเต็มสำหรับคำสั่งนี้ ส่วน product 3 สามารถที่จะกระจายข้อมูลได้ด้วยตนเอง โดยสามารถที่จะกระจายข้อมูลได้ 5 แบบ คือ range, hash, round-robin, expression และ hybrid partition จึงได้คะแนนเต็มเช่นเดียวกัน

10 คำสั่ง sorting ทั้ง 3 ผลิตภัณฑ์สามารถที่จะทำงานกับคำสั่งนี้ในแบบขนานได้ ดังนั้นจึงได้คะแนนเต็มทั้ง 3 ผลิตภัณฑ์

11 คำสั่ง load balancing ใน product 1 ไม่มีคำสั่งการทำ load balancing ได้ดังนั้นจึงไม่มีคะแนนสำหรับคำสั่งนี้ product 2 สามารถทำงานกับคำสั่งนี้ในรูปแบบ static load balancing จึงได้คะแนนไป 4 คะแนน ส่วน product 3 สามารถที่จะทำงานกับคำสั่งนี้ในรูปแบบของ dynamics load balancing ดังนั้น product 3 จึงได้คะแนนเต็มสำหรับคำสั่ง load balancing เนื่องจากการทำงานในแบบ dynamics load balancing จะให้ประสิทธิภาพการทำงานที่ดีกว่าแบบ static load balancing

7.6 สรุป

จากตารางการให้คะแนนคุณสมบัติระบบฐานข้อมูลแบบขนานทั้ง 3 ผลิตภัณฑ์ จะได้ว่าในผลิตภัณฑ์ที่ 2 จะมีคะแนนสูงที่สุดคือได้ไป 91 คะแนน ในผลิตภัณฑ์ที่ 3 ได้คะแนนเป็นอันดับสองได้คะแนน 84 คะแนน ส่วนผลิตภัณฑ์ที่ 1 ได้คะแนนเป็นอันดับสามได้คะแนน 53 คะแนน ดังนั้นในผลิตภัณฑ์ที่สองจะมีคุณสมบัติที่สามารถทำงานในแบบขนานได้ดีที่สุดตามมาด้วยผลิตภัณฑ์ที่สาม สุดท้ายคือผลิตภัณฑ์ที่หนึ่ง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เอกสารอ้างอิง

Aronoff,E.,Loney,K and Sonawalla,N. Advanced Oracle Tuning and administration. Berkeley ;McGraw-Hill,1997

Bitton ,D and Turbyfill ,C , "A Retrospevtive on the Wisconsin Benchmark," appears in Reading in Database System ,Ed . by M.Stonbebraker , Morgan Kaufmann ,INC., 1988, pp.280-299

Bobrowski,S.,Chin-Lee,C.,Closkey,C.,Frazzini,J.,Sokokolsky,D.(1996). Oracle 7 Server Concepts

Bontempo,Charles J and Saracco,Cynthia M. Database Management principle and products. Newyork :Printicle hall PTR ,1995

Dewitt, D.,and J. Gray. "Parallel Database Systems: The Future of High Performance Database Systems." Communication of the ACM,35(6),June 1992.

Ferguson,M.(1994,November). "Breaking up is hard to do." Database programming&design,pp.33-41

Feruson,M.(1994,october). "Parallel database the shape of thing to come." Database programming&design, pp.33-43

Frazzini,J.,Moran,R.(1996). Oracle 7 Server Tuning , Release 7.3. redwood city :1996

HALL,J.,HSIAO,D.,KAMEL,M.(1991). " Performance Evaluations of a Parallel, Scalable and Expandable Database Computer system." system science, proceeding of the twenty-fourth annual hawii international conference ,85-97

Linden,B. and others. Oracle 7 server SQL reference release 7.3 . redwood city : 1996

LU,H.,OOI,B.,Tan,K.(1994). Query Processing in Parallel Relational Database system. SINGAPORE:IEEE COMPUTER Society Press.

Moran ,R.(1996). Oracle 7 Parallel Server Concept&administration Release 7.3. redwood city :1996

NCR System Corporation "NCR Products :Teradata Database System." See also HTTP://www.ncr.com/product/integrated/pages

Norton,S. and Dipasquale,M. Threadtime the Multithreaded Programming Guide . New Jersey : Prentices Hall Press,1997

Release 7.3. redwood city :1996

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Rudin,K.(December,1995). "The practical side of going parallel." Database programming&design, pp.27-33

Ryan,N. ,Smith,D. Database System Engineering . Sheffield; Thansom Publishing Inc, 1995

Schnedies,D.A.and Dewitt,D,J.(1990). " Tradeoffs in processing complex join queries via hashing in multiprocessor database machines." Proc .16 th VLDB conf , pp 469-480

Turbyfill, c., C. Orji , and Bitton D, "AS3AP A Comparative Relational Database Benchmark " Proceeding of the IEEE COMPCON, 1989 ,PP.560-564.

Valduriez,P.(1993). "Parallel database system : the case for share something." Data Engineering ,1992 Proceeding . Ninth International Conference ,pp.460-465.

Walton,C.B.,Dale,A.G.(1991). " Data skew and the scalability of parallel joins ." Parallel and distributed processing proceeding of the third IEEE symposium,44-51

Whalen,E.(1996). Oracle Performance Tuning and Optimization.United stated : Sams Publishing

www.informix.com

www.sybase.com

Zamick,j.,Warren,R. and O'Sullivan,J. Enterprise client/server technology massively parallel processing for business. London :International Thomson Computer Press , 1995

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ประวัติผู้เขียน

ชื่อผู้เขียน	นาย ณรงค์ ประติพัทธิ์พงษ์
วัน เดือน ปีเกิด	วันที่ 14 ตุลาคม พ.ศ. 2513
สถานที่เกิด	จังหวัด นครปฐม
วุฒิการศึกษาระดับปริญญาตรี	วิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์
สถานที่สำเร็จการศึกษา	มหาวิทยาลัยเกษตรศาสตร์
ปีที่สำเร็จการศึกษา	ปี พ.ศ. 2536
อาชีพปัจจุบัน	รับราชการที่กรมตำรวจ กระทรวงมหาดไทย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้