วิธีการทางเอเจนต์ในการเก็บสถิติและวางแผนคำถามในระบบฐานข้อมูลแบบกระจาย

AN AGENT APPROACH FOR STATISTICS COLLECTION AND QUERY OPTIMIZATION IN DISTRIBUTED DATABASE SYSTEMS

ณัฐลดา รัตนวิจัย
NUTLADA RATTANAVIJAI

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์
บัณฑิตวิทยาลัย
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ. 2559

วิธีการทางเอเจนต์ในการเก็บสถิติและวางแผนคำถามในระบบฐานข้อมูลแบบกระจาย

AN AGENT APPROACH FOR STATISTICS COLLECTION AND QUERY OPTIMIZATION IN DISTRIBUTED DATABASE SYSTEMS

ณัจลดา รัตนวิจัย

NUTLADA RATTANAVIJAI

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์
บัณฑิตวิทยาลัย
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ. 2550

AN AGENT APPROACH FOR STATISTICS COLLECTION AND QUERY OPTIMIZATION IN DISTRIBUTED DATABASE SYSTEMS

NUTLADA RATTANAVIJAI

A THESIS SUBMITTED IN PARTAIL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

MASTER OF ENGINEERING IN COMPUTER ENGINEERING

SCHOOL OF GRADUATE STUDIES

KING MONGKUT'S INSTITUTE OF THECHNOLOGY LADKRABANG

2007

หัวข้อวิทยานิพนธ์	วิธีการทางเอเจนต์ในการเก็บสถิติและวางแผนคำถามในระบบฐานข้อมูลแบบกระจาย
นักศึกษา	นางสาวณัจลดา รัตนวิจัย
รหัสประจำตัว	45061033
ปริญญา	วิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
พ.ศ.	2550
อาจารย์ที่ปรึกษาวิทยานิพนธ์	รศ.ดร. ศุภมิตร จิตตะยโศธร

บทคัดย่อ

วิทยานิพนธ์นี้นำเสนอวิธีใช้โมบายเอเจนต์ (Mobile Agent) เพื่อเพิ่มประสิทธิภาพการทำงานบนระบบฐานข้อมูลแบบกระจาย (Distributed Database System) โดยใช้โมบายเอเจนต์ในการวิเคราะห์สถิติ (Statistics) รวมถึงเลือกวิธีในการเก็บสถิติ โดยมีเป้าหมายให้แต่ละฐานข้อมูลในระบบมีสถิติที่มีความถูกต้อง และมีข้อมูลเพียงพอสำหรับใช้ในการวางแผนคำถาม (Query Optimization)

ในส่วนของการวางแผนคำถามนอกจากจะใช้สถิติของข้อมูลในแต่ละตารางแล้ว ประสิทธิภาพของเครื่องที่ตอบคำถาม รวมถึงปริมาณข้อมูลที่ต้องรับส่งกันระหว่างฐานข้อมูลแต่ละแห่ง ล้วนเป็นปัจจัยในการคำนวณหาแผนที่มีประสิทธิภาพมากที่สุด โดยจุดเด่นในการวางแผนคำถามคือการมีข้อมูลสถิติที่มากและดีพอสำหรับการวางแผนคำถาม และการส่งโมบายเอเจนต์ไปประมวลผลลัพธ์ย่อยที่แต่ละฐานข้อมูล แทนการให้แต่ละฐานข้อมูลส่งข้อมูลมาให้ส่วนกลางเป็นตัวประมวลผล ทำให้การประมวลผลคำถามย่อยสามารถทำงานไปพร้อมๆ กัน และลดปริมาณการส่งข้อมูลเนื่องจากได้ใช้โมบายเอเจนต์ไปประมวลผลแต่คำตอบที่ต้องการเท่านั้น

และในตอนท้ายมีการทดลองเปรียบเทียบผลการทำงานระหว่างระบบฐานข้อมูลที่มีเอเจนต์กับระบบฐานข้อมูลออราเคิล

Thesis Title	An Agent Approach for Statistic Collection and Query Optimization in Distributed Database Systems
Student	Miss Nutlada Rattanavijai
Student ID.	45061033
Degree	Master of Engineering
Program	Computer Engineering
Year	2007
Thesis Advisor	Assoc. Prof. Dr. Suphamit Chittayasothorn

ABSTRACT

To improve performance on distributed database system, this thesis bases on Mobile Agent by analyzing statistics and choosing the best way to collect them. That helps each database be precise and enough information for Query Optimization.

According to query optimization, not only table's statistics and the efficiency of optimizer, the number of data to transmission is also the essential part to generate execution plan. When the statistics are numerous and good enough for query optimization, sending mobile agent to execute Sub-query, instead of data sent by each database, makes execute Sub-query work parallel and also reduce the number of database sent by mobile agent which takes only their result.

Finally, the experiment comparing between the results of database system based on mobile agent and Oracle was also shown in the end of this thesis.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จด้วยดีได้ ด้วยความช่วยเหลือของ รศ.ดร.ศุภมิตร จิตตะยโศธร ซึ่งเป็นอาจารย์ผู้ควบคุมวิทยานิพนธ์ ที่ได้ให้คำปรึกษา และคำแนะนำที่ดีตลอดการทำวิทยานิพนธ์ ข้าพเจ้ารู้สึกซาบซึ้งในความเมตตากรุณาของท่านในด้านต่างๆ และรู้สึกขอบพระคุณเป็นอย่างยิ่ง

ขอขอบคุณคณาจารย์ทุกๆ ท่านในภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ ที่ได้ประสิทธิประสาทวิชาความรู้ให้แก่ข้าพเจ้า

ขอขอบคุณเพื่อนๆ พี่ๆ น้องๆ ทุกท่านที่เป็นกำลังใจ และเป็นที่ปรึกษา รวมถึงให้ความช่วยเหลือในด้านต่างๆ

สุดท้ายนี้ขอบกราบขอบพระคุณบิดา มารดา และน้องสาวข้าพเจ้าที่ได้ให้ความช่วยเหลือทุกๆ ด้าน ทำให้สามารถทำวิทยานิพนธ์ฉบับนี้สำเร็จได้

ณัจลดา รัตนวิจัย

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบันมีการใช้ฐานข้อมูลเพื่อเก็บข้อมูลกันของแต่ละองค์กร แต่ต่อมาเมื่อองค์กรมีขนาดใหญ่ขึ้น การเก็บข้อมูลของทั้งองค์กรไว้ยังส่วนกลางที่เดียว เป็นการไม่สะดวกต่อการใช้งาน จึงนิยมที่จะให้แต่ละส่วนขององค์กรเก็บเฉพาะข้อมูลส่วนของตนไว้ที่ฐานข้อมูลของตนเอง แล้วเชื่อมโยงข้อมูลทั้งหมดผ่านระบบเครือข่าย เพื่อให้สามารถเข้าถึงข้อมูลของทุกส่วนได้ จึงเกิดเป็นระบบฐานข้อมูลแบบกระจาย (Distributed Database System)

เมื่อกระจายข้อมูลไปยังฐานข้อมูลต่างๆ แล้ว โดยปกติงานในการเพิ่ม (insert) ลบ (Delete) หรือเปลี่ยนแปลง (update) มักทำจากเครื่องที่อยู่ในเครือข่ายบริเวณเฉพาะที่ (LAN) ซึ่งมีความเร็วในการรับส่งข้อมูลผ่านระบบเครือข่ายสูง แต่หากเป็นการสอบถาม (select) ข้อมูลอาจต้องการทั้งข้อมูลจากฐานข้อมูลของเครื่องที่อยู่เครือข่ายบริเวณเฉพาะที่ หรือต้องการจากฐานข้อมูลที่อื่น ซึ่งการส่งข้อมูลจากฐานข้อมูลอื่นจะต้องส่งข้อมูลผ่านเครือข่ายบริเวณกว้าง (WAN) ที่มีความเร็วในการรับส่งข้อมูลต่ำกว่าเครือข่ายเฉพาะที่หลายเท่า ซึ่งจุดนี้ทำให้เกิดปัญหาคอขวด (bottle neck) ทำให้ต้องเสียเวลารอข้อมูลเป็นเวลานาน

ในการสอบถามข้อมูล ระบบจัดการฐานข้อมูล (Database Management System: DBMS) ใช้ออพติไมเซอร์ (Optimizer) ทำหน้าที่ในการวางแผนคำถาม (Optimization) เพื่อเลือกวิธีในการเข้าถึงข้อมูลที่มีประสิทธิภาพมากที่สุด ในการวางแผนคำถามของออพติไมเซอร์จะใช้ข้อมูลสถิติ (Statistics) ในการประมาณค่าใช้จ่าย (cost) และออพติไมเซอร์จะเลือกวิธีที่มีค่าใช้จ่ายโดยประมาณต่ำที่สุดในการประมวลผล แต่ข้อมูลสถิติจะได้จากการเก็บสถิติ (Statistics Collection) เท่านั้น ซึ่ง DBMS ที่ใช้อยู่ในปัจจุบันจะมีคำสั่งที่เรียกให้ DBMS เก็บสถิติ แต่งานในการเก็บสถิติเป็นงานที่ใช้ทรัพยากรในการทำงานสูง เพื่อไม่ให้มีผลต่อการทำงานอื่น การเก็บสถิติจึงควรเลือกเวลาในการเก็บสถิติให้เหมาะสม ใช้เวลาในการเก็บสถิติให้น้อยที่สุด และเลือกเก็บสถิติเฉพาะตารางที่จำเป็นเท่านั้น ดังนั้นประเด็นสำคัญที่ต้องคำนึงถึงคือเวลาที่เหมาะในการเก็บสถิติคือเวลาใด และตารางใดบ้างที่ต้องการเก็บสถิติ

การถามคำถามที่ต้องการข้อมูลจากฐานข้อมูลอื่น สามารถทำได้โดยการส่งคำถามไปให้กับระบบจัดการฐานข้อมูลที่ใกล้ที่สุด ซึ่งระบบจัดการฐานข้อมูลที่มีอยู่ในปัจจุบันจะทำหน้าที่ในการถามคำถามต่อไปยังฐานข้อมูลอื่นๆ เพื่อให้ได้ข้อมูลตามที่ต้องการได้ แต่ความสามารถในการวางแผนเข้าถึง ข้อมูลยังไม่มีประสิทธิภาพดีพอ เนื่องจากประการแรกในการประมาณค่าใช้จ่ายออพติไมเซอร์จะรู้เพียงข้อมูลสถิติที่เครื่องตัวเอง สถิติของฐานข้อมูลอื่นเป็นการประมาณเท่านั้น อีกทั้ง

การประมวลผลก็ต้องให้ฐานข้อมูลแต่ละแห่งส่งข้อมูลของแต่ละฐานข้อมูลมาประมวลผลที่เครื่องที่รับคำถามเท่านั้น ซึ่งปริมาณข้อมูลที่ส่งมามักมีจำนวนมากเกินความจำเป็น ทำให้ต้องใช้เวลาในการประมวลผลนานกว่าที่ควรจะเป็น

เทคโนโลยีโมบายเอเจนต์ ถือกำเนิดขึ้นจากการนำเอาความสามารถของเอเจนต์มารวมกันกับความสามารถในการเคลื่อนที่ เพื่อรองรับการทำงานบนระบบแบบกระจาย ด้วยความสามารถของโมบายเอเจนต์ในการเคลื่อนย้ายตัวเองไปทำงานบนเครื่องใดๆ ในระบบเครือข่าย โดยไม่ต้องคำนึงถึงฮาร์ดแวร์ (Hardware) หรือระบบปฏิบัติการ (Operating System) ที่ใช้ จึงมีการนำเอาโมบายเอเจนต์ไปประยุกต์ใช้ในงานด้านต่างๆ เพื่อเพิ่มความสามารถในการทำงานนั้นๆ

1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา

- ศึกษาถึงลักษณะ วิธีการทำงาน ข้อดี และข้อจำกัดของโมบายเอเจนต์ เพื่อหาแนวทางในการนำโมบายเอเจนต์มาพัฒนากับระบบฐานข้อมูลแบบกระจาย
- นำโมบายเอเจนต์มาใช้ในการเก็บสถิติบนระบบฐานข้อมูลแบบกระจาย โดยให้เอเจนต์เลือกวิธีและเวลาที่เหมาะสมในการเก็บสถิติ
- นำโมบายเอเจนต์มาใช้ในการวางแผนคำถาม โดยให้โมบายเอเจนต์ทำหน้าที่คิวรี่คำถามย่อยจากฐานข้อมูลต่างๆ ที่เครื่องฐานข้อมูลแต่ละแห่ง
- วัดประสิทธิภาพการทำงานของฐานข้อมูล เพื่อศึกษาว่าการใช้โมบายเอเจนต์ในการทำงานมีส่วนช่วยเพิ่มประสิทธิภาพการทำงานได้มากน้อยเพียงใด

1.3 สมมุติฐานของการศึกษา

ฐานข้อมูลที่ใช้อยู่ในปัจจุบัน มีวิธีในการวางแผนคำถามโดยใช้การประมาณค่าใช้จ่ายของแต่ละวิธีในการเข้าถึงข้อมูล เพื่อเลือกวิธีที่มีค่าใช้จ่ายต่ำที่สุดในการประมวลผล โดยค่าใช้จ่ายที่นำมาคำนวณได้มาจากข้อมูลสถิติ เพื่อให้การคำนวณค่าใช้จ่ายมีความผิดพลาดน้อยที่สุด จึงจำเป็นต้องมีการเก็บสถิติของฐานข้อมูลอยู่เสมอๆ แต่การเก็บสถิติเป็นงานที่ต้องการทรัพยากรในการทำงานมาก หากทำในขณะที่มีผู้ใช้อื่นใช้งานฐานข้อมูลอยู่ จะทำให้ไปแย่งทรัพยากรจากผู้ใช้มาก ถึงแม้ว่าข้อมูลสถิติเป็นข้อมูลที่มีประโยชน์ แต่ในการเก็บสถิติเป็นงานที่ส่งผลกระทบต่อผู้ใช้ ดังนั้นจุดที่เหมาะสมที่สุดในการเก็บสถิติคือ การเลือกเก็บสถิติเฉพาะตารางที่มีความจำเป็นต้องเก็บสถิติเท่านั้น เพื่อลดเวลาในการเก็บสถิติให้น้อยที่สุด และควรเลือกเก็บสถิติกระทำในช่วงเวลาที่ไม่มีผู้ใช้อื่นใช้งานอยู่ แต่การจะวิเคราะห์ว่าตารางใดมีความจำเป็นต้องเก็บสถิติต้องอาศัยเวลาในการเฝ้าดูการเปลี่ยนแปลงของข้อมูล ตรงจุดนี้ในงานวิจัยจึงนำโมบายเอเจนต์มาใช้ในการวิเคราะห์สถิติและเก็บสถิติของแต่ละฐานข้อมูล

ในการวางแผนคำถามเพื่อเลือกวิธีที่ดีที่สุดในการเข้าถึงข้อมูล สิ่งสำคัญที่ต้องคำนึงถึงเมื่อฐานข้อมูลกระจายอยู่ยังที่ต่างๆ คือปริมาณข้อมูลที่จะต้องส่งไปมาระหว่างฐานข้อมูลแต่ละแห่ง เพื่อให้การส่งข้อมูลมีปริมาณน้อยที่สุดเท่าที่จะทำได้ จึงจำเป็นต้องมีการวางแผนในการเข้าถึงข้อมูลโดยคำนึงถึงปริมาณข้อมูลที่ต้องส่งไปมาระหว่างฐานข้อมูลด้วย หรือเปลี่ยนแนวความคิดในการส่งข้อมูลไปเพื่อให้เครื่องใดเครื่องหนึ่งประมวลผล เป็นการส่งโมบายเอเจนต์ไปยังฐานข้อมูลต่างๆ เพื่อให้ทุกๆ เครื่องสามารถเป็นเครื่องประมวลผลได้ ขึ้นอยู่กับปริมาณข้อมูลที่จะเป็นตัวกำหนดว่าเครื่องใดควรเป็นเครื่องประมวลผล ซึ่งการทำในลักษณะนี้น่าจะเป็นการลดปริมาณข้อมูลที่ต้องถูกส่งออกไประหว่างฐานข้อมูลได้

1.4 ทฤษฎีหรือแนวความคิดที่ใช้ในการวิจัย

สถิติเป็นข้อมูลสำคัญที่มีส่วนช่วยให้ออพติไมเซอร์สามารถเลือกวิธีในการเข้าถึงข้อมูลได้อย่างมีประสิทธิภาพ แต่ในการเก็บสถิติแต่ละครั้งไม่จำเป็นต้องเก็บสถิติทุกตารางในฐานข้อมูล แต่ควรเลือกที่จะเก็บสถิติกับตารางที่ข้อมูลมีการเปลี่ยนแปลงมากในระดับหนึ่ง หรือเป็นตารางที่มีจำนวนการเปลี่ยนแปลงของข้อมูลไม่มาก แต่เมื่อนำข้อมูลสถิติที่มีอยู่ไปใช้ในการวางแผนคำถามแล้วทำให้เกิดการวางแผนคำถามผิดพลาด โมบายเอเจนต์จะเป็นส่วนที่ช่วยในการตรวจสอบและปรับปรุงข้อมูลสถิติอยู่ตลอดเวลา เพื่อให้ข้อมูลสถิติมีความสอดคล้องกับข้อมูลจริงๆ ในฐานข้อมูล ส่วนเวลาที่ควรเก็บสถิติเอเจนต์จะเลือกเป็นช่วงเวลาที่ไม่มีผู้ใช้อื่นใช้งานฐานข้อมูลนั้น เพื่อไม่ให้ผู้ใช้อื่นได้รับผลกระทบในระหว่างที่ทำการเก็บสถิติ

ในการวางแผนวางแผนคำถาม เป็นการเลือกวิธีที่ใช้ในการเข้าถึงข้อมูลอย่างมีประสิทธิภาพมากที่สุด นอกจากจะใช้ข้อมูลสถิติในฐานข้อมูลนั้นมาใช้เป็นส่วนตัดสินใจเลือกวิธีแล้ว สิ่งสำคัญที่ต้องคำนึงถึงคือปริมาณข้อมูลที่ต้องส่งผ่านเครือข่าย เนื่องจากความเร็วของระบบเครือข่ายมีค่าน้อยกว่าความเร็วของดิสก์หลายเท่า ดังนั้นก่อนที่จะส่งข้อมูลจากฐานข้อมูลใดๆ ควรที่จะมีการคัดกรอง (filter) ให้เหลือเฉพาะข้อมูลที่ต้องการจริงๆ เท่านั้น จะได้ไม่ต้องใช้เวลาไปกับการส่งข้อมูลที่ไม่จำเป็นผ่านเครือข่าย ที่ต้องใช้เวลาในการส่งมาก

1.5 ขอบเขตของการศึกษา

เพื่อศึกษาและทดลองใช้โมบายเอเจนต์ในการเก็บสถิติและวางแผนคำถามกับระบบฐานข้อมูลแบบกระจาย โดยที่

1. เครื่องคอมพิวเตอร์ที่ใช้ในการทดลองประกอบด้วย
    1.1 เครื่องที่เป็นฐานข้อมูลจำนวน 2 เครื่อง
    1.2 เครื่องที่ทำหน้าที่เป็น Statistics Server จำนวน 1 เครื่อง

1.3 เครื่องไคลเอนต์สำหรับรับคำสั่ง SQL

2. ฐานข้อมูลที่ใช้ในการทดลองเป็นฐานข้อมูล Oracle เวอร์ชัน 10g
3. เครื่องมือที่ใช้ในการพัฒนาโมบายเอเจนต์คือ Voyager
4. ฐานข้อมูลทั้ง 2 เชื่อมต่อผ่านระบบเครือข่ายเฉพาะที่ (LAN)
5. การวัดประสิทธิภาพจะวัดจากจำนวนเวลาที่ใช้ในการสืบค้นข้อมูลจะเริ่มนับตั้งแต่เริ่มส่งคำสั่ง SQL ไปให้กับระบบจัดการฐานข้อมูลจนถึงเวลาที่ได้รับข้อมูลทั้งหมด

1.6 ขั้นตอนของการศึกษา

- ศึกษาลักษณะและการทำงานของโมบายเอเจนต์
- ศึกษาเครื่องมือที่ใช้ในการพัฒนาโมบายเอเจนต์ แล้วเปรียบเทียบข้อดีข้อเสียของเครื่องมือแต่ละชนิด เพื่อเลือกเครื่องมือที่จะนำมาใช้ในการพัฒนา
- ศึกษาถึงวิธีการทำงานของระบบจัดการบริหารข้อมูล โดยเน้นในจุดที่สามารถพัฒนาแอพพลิเคชั่นที่ช่วยประสิทธิภาพการทำงานให้กับระบบจัดการบริหารข้อมูล
- สร้างโมบายเอเจนต์ที่ใช้ในการเก็บสถิติ แล้วเปรียบเทียบผลที่ได้จากระบบที่มีโมบายเอเจนต์ในการเก็บสถิติกับระบบที่ให้ฐานข้อมูลบริหารจัดการสถิติเอง
- ออกแบบวิธีการวางแผนคำถามที่ใช้ข้อมูลสถิติในการคำนวณค่าใช้จ่าย โดยให้โมบายเอเจนต์ทำหน้าที่ในการสืบค้นข้อมูลจากฐานข้อมูลต่างๆ แล้วเปรียบเทียบเวลาที่ใช้ในการสืบค้นข้อมูลโดยใช้โมบายเอเจนต์กับระบบบริหารจัดการฐานข้อมูล
- สรุปผลการทดลองพร้อมจัดทำเอกสารวิทยานิพนธ์

1.7 รายละเอียดในแต่ละบท

วิทยานิพนธ์ฉบับนี้ได้แบ่งเนื้อหาออกเป็น 6 บทด้วยกันคือ

บทที่ 1 กล่าวถึงความเป็นมาและความสำคัญของปัญหา วัตถุประสงค์ ขอบเขตของงานวิจัยและขั้นตอนการศึกษา

บทที่ 2 กล่าวถึงทฤษฎีต่างๆ ที่ใช้งานวิจัย

บทที่ 3 กล่าวถึงวิธีในการนำโมบายเอเจนต์มาใช้ในการเก็บสถิติ

บทที่ 4 กล่าวถึงวิธีในการวางแผนคำถามที่ใช้โมบายเอเจนต์ในการสืบค้นข้อมูล

บทที่ 5 กล่าวถึงการดำเนินการทดลอง และผลการทดลอง

บทที่ 6 กล่าวถึงการสรุป ข้อเสนอแนะ และแนวทางการทำวิจัยต่อ

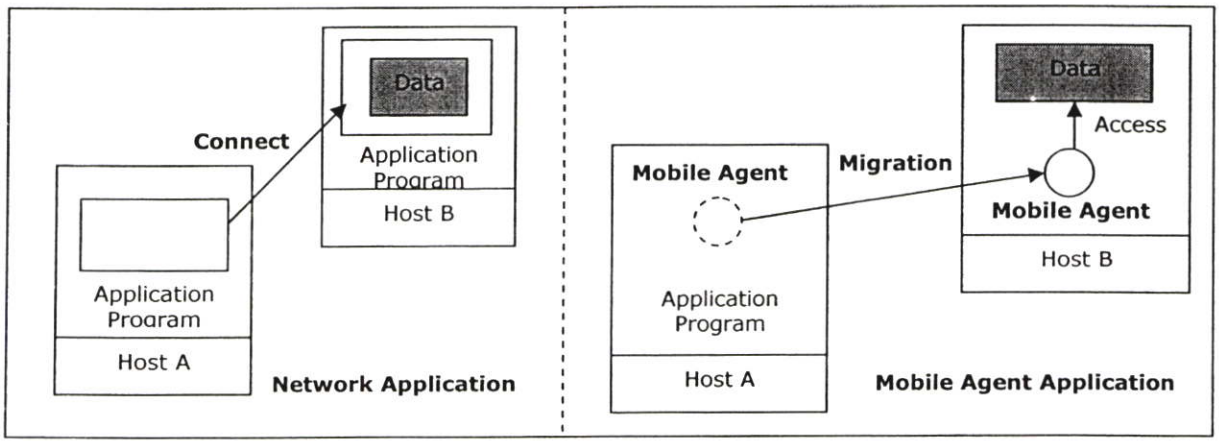
บทที่ 2
ทฤษฎีที่เกี่ยวข้องในงานวิจัย

2.1 โมบายเอเจนต์

2.1.1 ลักษณะของโมบายเอเจนต์

โมบายเอเจนต์เป็นการนำเอาเทคโนโลยีของเอเจนต์ ซึ่งเป็นซอฟแวร์ที่ใช้ในการทำงานแทนคน โดยมีการกำหนดวิธีการทำงานและเป้าหมายให้กับเอเจนต์มารวมเข้ากับความสามารถในการเคลื่อนที่(โมบายเทคโนโลยี) ทำให้โมบายเอเจนต์เป็นซอฟแวร์เอเจนต์ที่มีความสามารถเคลื่อนที่ไปทำงานยังเครื่องที่ต้องการได้โดยไม่จำเป็นต้องมีการติดตั้งโปรแกรมก่อน ในการเคลื่อนที่ของเอเจนต์ เอเจนต์จะนำโค้ดโปรแกรม ข้อมูลและสถานะในขณะนั้นของเอเจนต์ติดตัวไปพร้อมกับการเคลื่อนที่ผ่านเครือข่ายไปยังเครื่องเป้าหมาย เมื่อไปถึงเป้าหมายจึงสามารถทำงานได้โดยอาศัยข้อมูลที่ติดตัวมาโดยไม่จำเป็นต้องมีการติดตั้งซอฟแวร์อื่นเพิ่มเติม และสามารถทำงานต่อจากสถานะที่ทำค้างอยู่ต่อได้ โดยไม่จะเป็นต้องเริ่มทำงานจากจุดเริ่มต้นใหม่ทุกครั้ง

โดยปกติการทำงานกับฐานข้อมูลมักเป็นการร้องขอข้อมูลที่ต้องการเพื่อให้ฐานข้อมูลส่งมาแล้วจึงนำข้อมูลที่ได้ทั้งหมดมาประมวลผล แต่โมบายเอเจนต์ใช้แนวคิดในการส่งเอเจนต์ไปยังที่ต่างๆ เพื่อเก็บข้อมูลหรือเดินทางไปประมวลผลที่เครื่องคอมพิวเตอร์อื่นๆ แล้วนำเฉพาะผลลัพธ์กลับมา ซึ่งวิธีการนี้เป็นการนำเอาข้อดีของการใช้เอเจนต์ไปทำงานต่างๆ บนเครื่องคอมพิวเตอร์ไปรวมกับความสามารถในการเคลื่อนที่ ทำให้ลดภาระงานของระบบเครือข่าย เนื่องจากไม่ต้องส่งข้อมูลมาให้เครื่องอื่นประมวลผล แต่ให้โมบายเอเจนต์เดินทางไปประมวลผลที่เครื่องนั้นๆ เอง



รูปที่ 2.1 การเปรียบเทียบการทำงานระหว่างโมบายเอเจนต์กับโปรแกรมทั่วๆ ไป

การส่งเอเจนต์ไปทำงานยังเครื่องคอมพิวเตอร์อื่นสามารถทำได้โดยไม่จำเป็นต้องใช้โมบายเอเจนต์ก็ได้ แต่ข้อแตกต่างของโมบายเอเจนต์กับการส่งเอเจนต์อื่นทำงานคือ ในการส่งเอเจนต์ไปทำงาน เอเจนต์จะถูกส่งไปยังเครื่องอื่นๆ ได้เมื่อถึงคำสั่งที่ใช้ในการเคลื่อนที่เท่านั้น แต่ในโมบายเอเจนต์ในขณะที่เอเจนต์กำลังทำงานอื่นใดอยู่ก็ตาม สามารถส่งโมบายเอเจนต์ไปยังเครื่องต้องการได้ตลอดเวลา และเอเจนต์ยังสามารถบันทึกสถานะของเอเจนต์ก่อนเคลื่อนย้ายเก็บไว้ได้ ทำให้ทำงานต่อในภายหลังได้

เอเจนต์แต่ละตัวมีความเป็นอิสระ เป็นตัวของตัวเอง สามารถทำงานได้ด้วยตัวเองไม่จำเป็นต้องพึ่งพาเอเจนต์อื่นใด และมีความสามารถที่จะตัดสินใจกระทำสิ่งใดก็ตามที่จะทำให้ไปบรรลุ เป้าหมายที่กำหนดไว้

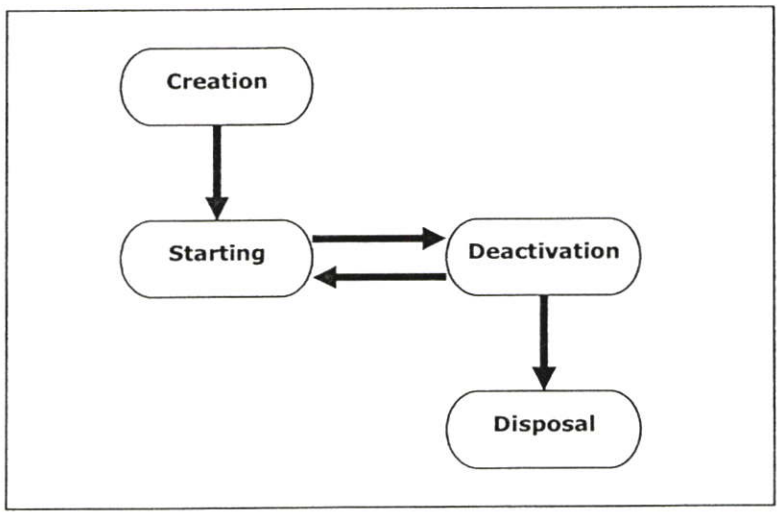
2.1.2 สถานะและวงจรชีวิตของโมบายเอเจนต์

เมื่อเอเจนต์ตัวหนึ่งถูกสร้างขึ้นมาแล้ว จะถูกกำหนดให้อยู่ในสถานะใดสถานะหนึ่งเสมอ โดยสถานะของเอเจนต์สามารถเปลี่ยนได้ตามลักษณะการทำงาน ซึ่งเอเจนต์หนึ่งๆ จะมีลักษณะวงจรชีวิตเป็นดังรูปที่ 2.2 ซึ่งประกอบไปด้วยสถานะต่างๆ ดังนี้

1. **สถานะการเกิด (Creation)** เป็นสถานะที่เกิดขึ้นกับเอเจนต์เพียงครั้งเดียว เกิดขึ้นเมื่อมีการสร้างเอเจนต์ขึ้นมาใหม่ และในขั้นตอนนี้จะมีการกำหนดหมายเลขไอดีให้กับโมบายเอเจนต์ โดยที่หมายเลขไอดีนี้จะต้องไม่ซ้ำกับโมบายเอเจนต์ตัวอื่น
2. **สถานะเริ่มต้นทำงาน (Starting)** เป็นสถานะที่เกิดขึ้นเมื่อเอเจนต์ได้เคลื่อนย้ายไปยังโฮสต์ใหม่ การเริ่มต้นการทำงานนี้อาจเป็นการเริ่มต้นทำงานใหม่เลย หรือเป็นการทำงานต่อจากเดิมที่ทำงานค้างอยู่ที่โฮสต์ก่อนหน้านี้ก็ได้ ขึ้นอยู่กับวิธีการเคลื่อนย้ายของโมบายเอเจนต์
3. **สถานะพักการทำงาน (Deactivationl)** เป็นสถานะที่เอเจนต์หยุดการทำงานลงชั่วคราว โดยเอเจนต์จะเก็บสถานะการทำงานล่าสุดที่ทำงานอยู่ในขณะนั้นลงในดิสก์ ก่อนที่เอเจนต์จะหยุดการทำงาน
4. **สถานะจบการทำงาน (Disposal)** เป็นสถานะที่เอเจนต์หยุดการทำงานอย่างถาวรอาจเป็นเพราะได้ทำงานบรรลุตามเป้าหมายที่กำหนดไว้แล้ว หรือเหตุผลอื่นใดก็ตาม ทั้งนี้เอเจนต์จะคืนทรัพยากรทั้งหมดที่เอเจนต์ใช้งานให้กับระบบ

จากรูปที่ 2.2 วงจรชีวิตของโมบายเอเจนต์นั้นเริ่มต้นด้วยการสร้างโมบายเอเจนต์ โดยอาจเป็นการสร้างใหม่หรือจำลองตัว (Clone) มาจากเอเจนต์ต้นแบบอื่นก็ได้ ในการจำลองตัวนี้จะทำให้ได้โมบายเอเจนต์ที่มีลักษณะเหมือนกับเอเจนต์ต้นแบบทุกอย่าง ยกเว้นเพียงหมายเลขไอดีของเอเจนต์เท่านั้นที่มีค่าที่แตกต่างกัน เอเจนต์ที่ถูกสร้างแล้วก็จะอยู่ในสถานะของการสร้างจนกระทั่งเอเจนต์เคลื่อนที่ไปยังโฮสต์ใหม่ เมื่อโมบายเอเจนต์เคลื่อนที่ไปยังโฮสต์ใหม่แล้วเอเจนต์ก็จะเปลี่ยน

สถานะเป็นเริ่มต้นการทำงาน และเอเจนต์จะอยู่ในสถานะนี้ไปจนกว่าจะมีการสั่งให้เคลื่อนย้ายไปยังโฮสต์ใหม่อีกครั้ง โดยก่อนที่เอเจนต์จะเคลื่อนที่ไปยังโฮสต์ใหม่นั้นเอเจนต์จะอยู่ในสถานะพักการทำงานเพื่อบันทึกข้อมูลของเอเจนต์ในขณะนั้นก่อนที่จะเคลื่อนย้ายไปยังโฮสต์ใหม่ และส่งข้อมูลไปยังโฮสต์ใหม่ ซึ่งเอเจนต์จะสลับสถานะระหว่างเริ่มต้นการทำงานกับพักการทำงานทุกครั้งที่มีการสั่งให้เอเจนต์เคลื่อนที่ไปยังโฮสต์ใหม่ และก่อนที่เอเจนต์จะจบการทำงาน เอเจนต์จะกลับมาอยู่ในสถานะพักการทำงานอีกครั้งก่อนที่จะเปลี่ยนสถานะเป็นจบการทำงานพร้อมกับคืนทรัพยากรทั้งหมดที่เอเจนต์ใช้ให้กับระบบ



รูปที่ 2.2 วงจรชีวิตของโมบายเอเจนต์

2.1.3 การเคลื่อนที่ของโมบายเอเจนต์

โมบายเอเจนต์เป็นเอเจนต์ที่สามารถเคลื่อนย้ายตัวเองไปทำงานยังโฮสต์อื่นใดๆ ก็ตามในระบบเครือข่าย โดยรูปแบบที่ใช้ในการเคลื่อนย้ายนั้นแบ่งได้ 2 ประเภทคือ

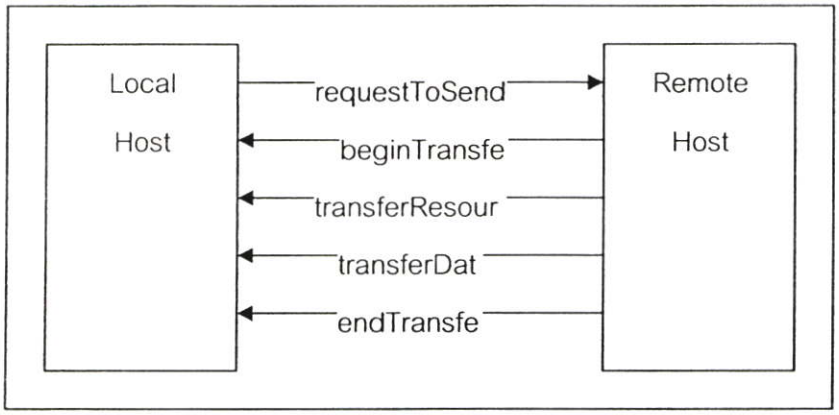
1. การเคลื่อนที่แบบ weak (weak migration) เป็นการเคลื่อนย้ายที่เอเจนต์จะนำเฉพาะโค้ด (code) และข้อมูล (data) ติดตัวไปขณะเคลื่อนย้าย เมื่อเอเจนต์ไปถึงโฮสต์ใหม่ เอเจนต์จะต้องเริ่มต้นทำงานจากจุดเริ่มต้นใหม่ทุกๆ ครั้ง

2. การเคลื่อนที่แบบ strong (strong migration) เป็นการเคลื่อนย้ายที่เอเจนต์จะนำเอาโค้ด, ข้อมูล และสถานะ (state) ติดตัวไปขณะเคลื่อนย้ายไปยังโฮสต์ใหม่ เมื่อเอเจนต์ไปถึงโฮสต์ใหม่ ทำให้เอเจนต์สามารถทำงานต่อจากสถานะเดิมที่โฮสต์เก่าตามที่ได้บันทึกไว้ได้

การเคลื่อนย้ายเอเจนต์จากโฮสต์หนึ่งไปยังอีกโฮสต์หนึ่งนั้น ไม่ว่าจะเป็นการเคลื่อนย้ายแบบ weak หรือ strong ก็ตาม จะเคลื่อนย้ายโดยบันทึกข้อมูลของเอเจนต์นั้นๆ ลงดิสก์ จากนั้นจึงส่งข้อมูลที่บันทึกไปแล้วผ่านระบบเครือข่ายไปยังโฮสต์ปลายทางที่ต้องการ เมื่อข้อมูลทั้งหมดถูก

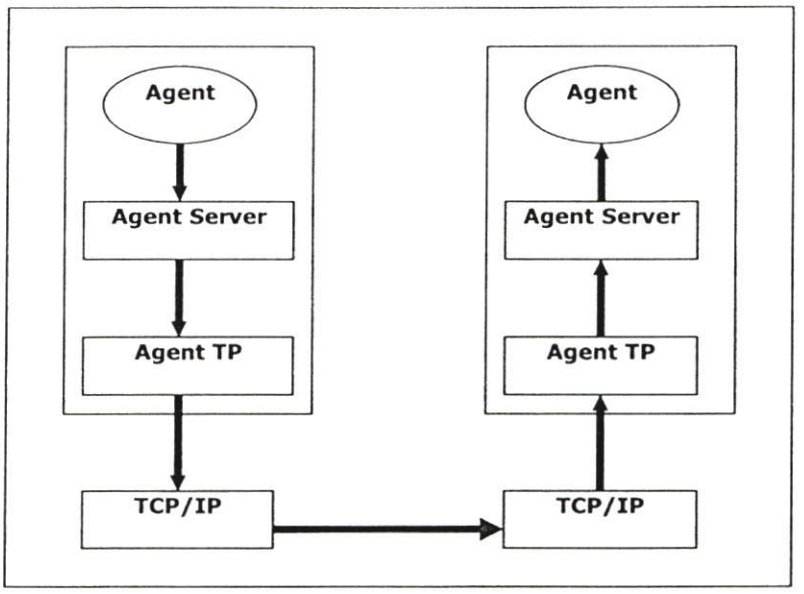
ส่งผ่านไปเรียบร้อยแล้ว ที่เครื่องโฮสต์ปลายทางจะสร้างเอเจนต์ขึ้นมาจากข้อมูลที่ส่งมากลับไปเป็นเอเจนต์เหมือนเดิม ซึ่งวิธีในการส่งผ่านข้อมูลระหว่างโฮสต์หนึ่งไปยังโฮสต์หนึ่งมีวิธีอยู่ 2 แบบ คือ ผ่านทาง RMI หรือผ่าน Socket

การติดต่อแมสเสจระยะไกล (Remote Method Invocation : RMI) เป็นเทคโนโลยีเฉพาะของจาวาที่ใช้ในการติดต่อสื่อสารระหว่างโปรแกรมที่เขียนด้วยจาวาผ่านระบบเครือข่าย เป็นการทำให้เครื่องต้นทางสามารถติดต่อแอพพลิเคชั่นที่ทำงานอยู่บนเครื่องปลายทางได้ด้วยการส่งอ็อบเจคจากเครื่องต้นทางไปทำงานบนเครื่องปลายทางได้เลย แทนการให้เครื่องปลายทางสร้างอ็อบเจคใหม่ขึ้นมาทำงานบนเครื่องปลายทางเอง สำหรับโมบายเอเจนต์ที่ใช้วิธี RMI ในการเคลื่อนย้ายจะเริ่มจากการที่เครื่องปลายทางที่เอเจนต์ต้องการเคลื่อนย้ายไป จะต้องส่งสัญญาณร้องขอมายังเครื่องต้นทางก่อน เพื่อระบุอ็อบเจค (object) ที่ต้องการจากเครื่องต้นทาง จากนั้นเครื่องต้นทางจะเริ่มส่งอ็อบเจคที่ต้องการไปให้เครื่องปลายทาง ดังรูปที่ 2.3



รูปที่ 2.3 ขั้นตอนการเคลื่อนย้ายของโมบายเอเจนต์ผ่าน RMI

อีกวิธีหนึ่งที่โมบายเอเจนต์สามารถใช้ในการเคลื่อนย้ายคือ การเคลื่อนย้ายโดยตรงผ่าน Socket ในการเคลื่อนย้ายผ่าน Socket เป็นวิธีที่มีความยืดหยุ่นในการใช้งานน้อย เนื่องจากต้องระบุเครื่องปลายทางและพอร์ต (port) ที่ต้องการส่งข้อมูลไป หากพอร์ตที่ระบุไว้ถูกใช้งานโดยโปรแกรมอื่นอยู่ก็จะไม่สามารถส่งข้อมูลได้ ไม่เหมือนกับการส่งข้อมูลผ่าน RMI ที่มีตัวกลางในการหาช่องทางในการเชื่อมต่อให้ รูปที่ 2.4 เป็นขั้นตอนในการเคลื่อนย้ายโมบายเอเจนต์ผ่าน Socket



รูปที่ 2.4 ขั้นตอนการเคลื่อนย้ายโมบายเอเจนต์ผ่าน Socket

2.1.4 การติดต่อสื่อสารระหว่างโมบายเอเจนต์

โมบายเอเจนต์สามารถติดต่อสื่อสารกับโมบายเอเจนต์อื่น โดยรูปแบบที่เอเจนต์ใช้สามารถในการติดต่อสื่อสารได้แก่

1. Procedure Call เป็นการสื่อสารแบบซิงโครไนส์ (synchronous) เริ่มต้นจาการที่เอเจนต์เอจะส่งคำสั่งร้องขอไปหาเอเจนต์บี ซึ่งในระหว่างที่รอให้เอเจนต์บีตอบกลับมานั้นเอเจนต์เอจะไม่สามารถทำงานอื่นได้เลย และเมื่อเอเจนต์บีตอบกลับมา เอเจนต์เอจึงจะสามารถไปทำงานอื่นต่อได้

2. The callback เป็นการสื่อสารแบบอะซิงโครไนส์ (asynchrounous) เนื่องจากเมื่อเอเจนต์เอส่งคำสั่งร้องขอไปหาเอเจนต์บีแล้ว เอเจนต์เอสามารถไปทำงานอื่นต่อได้เลย โดยไม่ต้องรอให้เอเจนต์บีตอบกลับมาก่อน และเมื่อเอเจนต์บีทำงานเสร็จก็จะตอบกลับไปหาเอเจนต์เอ

3. The mailbox เป็นวิธีที่ผสมผสานระหว่างซิงโครไนส์และอะซิงโครไนส์ โดยเมื่อเอเจนต์เอต้องการส่งคำสั่งร้องขอไปให้กับเอเจนต์บีนั้น เอเจนต์เอจะต้องส่งไปที่เมลล์บ็อกซ์ (mailbox) ของเอเจนต์บีแล้วเอเจนต์เอก็จะกลับไปทำงานอย่างอื่นต่อ และเมื่อเอเจนต์บีมาเช็คที่เมลล์บ็อกซ์ของตัวเองก็จะพบคำสั่งร้องขอจากเอเจนต์เอ จากนั้นเอเจนต์บีก็จะทำงานตามที่เอเจนต์เอได้ร้องขอไว้ เมื่อได้คำตอบก็จะตอบกลับไปที่เมลล์บ็อกซ์ของเอเจนต์เอ เมื่อเอเจนต์เอมาเช็คที่เมลล์บ็อกซ์ ของตัวเองก็จะพบคำตอบที่เอเจนต์บีส่งมาให้

ในกรณีที่เอเจนต์ต้องการสื่อสารกับเอเจนต์มากกว่า 1 ตัวจะใช้วิธีที่เรียกว่าบรอดแคส (broadcast) หรือมัลติแคส (multicast) ซึ่งเป็นการส่งข้อความเดียวกันให้กับเอเจนต์หลายๆ ตัว ในเวลาเดียวกัน

2.1.5 เครื่องมือที่ใช้ในการพัฒนาโมบายเอเจนต์

ในปัจจุบันมีเครื่องมือมากมายที่ใช้ในการพัฒนาโมบายเอเจนต์ โดยเครื่องมือแต่ละตัวจะมีฟังก์ชันในการสร้างและใช้งานโมบายเอเจนต์ เพื่อให้ผู้พัฒนาง่ายต่อการพัฒนาแอพพลิเคชั่นอื่น ซึ่งส่วนใหญ่มักเป็นโมบายเอเจนต์ที่พัฒนาด้วยภาษาจาวา

2.1.5.1 Aglets

Aglets เป็นผลงานของ IBM's Tokyo Research Laboratory ถือกำเนิดขึ้นเมื่อในช่วงปี ค.ศ. 1995 มีเป้าหมายในการนำความสามารถในการเคลื่อนที่ไปเพิ่มให้กับ applets โดยคำว่า Aglet มากจาก **agent** รวมกับ **applet** สำหรับชุดซอฟแวร์ที่ใช้ในการพัฒนาโปรแกรม (Aglets Software Development Kit : ASDK) ประกอบไปด้วย ชุดโปรแกรมสำหรับใช้ในการสร้างและใช้งานโมบายเอเจนต์ เอกสารอ้างอิง ตัวอย่างโปรแกรม Aglet สามารถดาวน์โหลดได้จาก http://www.trl.ibm.com/aglets/ สำหรับ Aglet ที่ทำหน้าที่เป็นเซิร์ฟเวอร์มีชื่อว่า Tahiti และ Aglet ที่ทำงานบนเวปเบราเซอร์มีชื่อว่า Fiji

Tahiti เป็นแอพพลิเคชั่นโปรแกรมที่ทำงานเป็นเอเจนต์เซิร์ฟเวอร์ โดยมีหน้าจออินเทอร์เฟสที่ง่ายต่อการใช้งาน มีฟังก์ชันในการเฝ้าดู (monitor), สร้าง (create), ทำลาย (dispose), สร้างแบบจำลอง (clone), ส่งเอเจนต์ไปยังเครื่องอื่น (dispatch), เรียกเอเจนต์กลับมาที่เซิร์ฟเวอร์ (retract), พักการทำงาน (deactivate) และกระตุ้นให้ทำงาน (activate) ซึ่งเครื่องคอมพิวเตอร์เครื่องหนึ่งๆ สามารถใช้เอเจนต์หลายๆ ตัวทำงานบนพอร์ต (port) ที่แตกต่างกันได้

Fiji เป็นจาวาแอพเพล็ตที่สามารถสร้าง Aglets หรือดึง Aglets ที่มีอยู่ไปทำงานบนเวปเบราเซอร์ โดย Fiji applet สามารถถูกอ้างถึงได้จากการส่ง URL ของเอเจนต์เป็นพารามิเตอร์ ทำให้สามารถนำไปแปะไว้ในโค้ด HTML ได้เหมือนๆ กับ Java applet

ในการเคลื่อนที่ของ Aglet ใช้วิธีการเคลื่อนที่ผ่าน Sockets ก่อนที่จะเคลื่อนที่ Aglet จะเปลี่ยนข้อมูลและโค้ดโปรแกรมให้อยู่ในรูปของไบท์อาร์เรย์ แล้วจึงส่งข้อมูลทั้งหมดไปยังเครื่องเป้าหมาย เมื่อไปถึงเครื่องเป้าหมายก็จะประกอบข้อมูลกลับเป็น Aglet เพื่อทำงานต่อไป สำหรับคลาสที่เป็นคลาสของระบบจะไม่มีการส่งผ่านเครือข่ายไป เนื่องจากต้องการลดปริมาณการส่งข้อมูลที่ไม่จำเป็นออก โดยถือว่าคลาสของระบบจะต้องมีอยู่ที่เครื่องปลายทาง และ Aglet สามารถทำงานได้เฉพาะเครื่องที่มี Aglet เซิร์ฟเวอร์ทำงานอยู่เท่านั้น

ในการติดต่อสื่อสารAglet ที่เป็นไคลเอนต์ด้วยกัน หรือกับเซิร์ฟเวอร์ สามารถใช้ได้ทั้งการสื่อสารแบบซิงโครไนส์และแบบอะซิงโครไนส์ ข้อมูลที่ใช้ในการสื่อสารจะอยู่ในรูปของ Message Aglets ซึ่งมีลักษณะเหมือนกัน Aglet ธรรมดาแต่ไม่มีโค้ดอยู่ในข้อมูลที่ส่งไป

Aglet เป็นที่นิยมใช้ในการพัฒนาโมบายเอเจนต์อย่างมาก เนื่องจากชุดพัฒนาโปรแกรมง่ายต่อการติดตั้ง และยังมีส่วนอินเทอร์เฟสที่ง่ายต่อการใช้งาน แต่ข้อเสียของ Aglet คือ

ไม่มีเมดธอด (method) ที่ใช้สำหรับบันทึกสถานะของ Aglet ทำให้ไม่สร้างเอเจนต์ที่เป็น persistence ได้

2.1.5.2 Concordia

Concordia เป็นผลงานของ Mitsubishi Electric ITCA (MEITCA) เป็นระบบโมบายเอเจนต์ที่ใหม่ที่สุด มีความน่าสนใจสำหรับงานที่เป็นแอพพลิเคชันขนาดใหญ่บนเวป Concordia เป็นเฟรมเวิร์ค (framework) ที่ใช้ในการพัฒนาและบริหารจัดการประสิทธิภาพของเครือข่ายที่ใช้โมบายเอเจนต์ในการเข้าถึงข้อมูลต่างๆ ที่อยู่บนเครื่องใดๆ ผ่านทางบริการต่างๆ เช่น Agent Manager ที่ใช้ในการติดต่อสื่อสารกับเซิร์ฟเวอร์ และใช้ในการส่งโมบายเอเจนต์ ในการทำงาน Concordia ยอมให้เอเจนต์หลายๆ ตัวทำงานร่วมกันได้

Concordia มีบริการต่างๆ เพื่อใช้ในการทำงาน ทั้งในส่วนที่ดูแลการสร้างและทำลายเอเจนต์ ส่วนที่ให้บริการด้านการติดต่อสื่อสารกับเซิร์ฟเวอร์ ดูแลด้านความปลอดภัย รวมถึงบริการในการติดต่อสื่อสาร โดยสามารถใช้ทั้งแบบซิงโครไนส์ และอะซิงโครไนส์

2.1.5.3 Odyssey

เป็นผลิตภัณฑ์ของ General Maginc ที่ใช้พัฒนาโมบายเอเจนต์ เป็นเครื่องมือตัวแรกที่ใช้งานได้จริง แต่ไม่เป็นที่นิยม มีเอเจนต์เซิร์ฟเวอร์ชื่อว่า Odyssey ที่ใช้สำหรับสร้าง ดูแล ส่ง และทำลายโมบายเอเจนต์

ในการเคลื่อนที่ของโมบายเอเจนต์เป็นแบบ weak ผ่าน RMI เหมือนๆ กับโมบายเอเจนต์อื่นๆ Odyssey มีข้อเสียคือ ไม่มีส่วนของความปลอดภัย มีเพียงเท่าที่เป็นความปลอดภัยของจาวา และไม่สามารถบันทึกเอเจนต์เพื่อนำเอเจนต์เพื่อนำกลับมาใช้อีก

2.1.5.4 Voyager

Voyager เป็นแนวคิดเรื่องโมบายเอเจนต์ของ ObjectSpace เกิดขึ้นในช่วงกลางปี ค.ศ. 1996 รองรับการทำงานแบบซีเรียลไรซ์ออบเจค (Serializable object) ทั้งในส่วนของโค้ดจาวา และคลาสไฟล์ ในชุดพัฒนาโปรแกรมประกอบด้วยไลบรารี่ (Library) ที่ใช้ในการสร้างโมบายเอเจนต์ เอกสารอ้างอิง และตัวอย่างโปรแกรม สามารถดาวน์โหลดชุดพัฒนาโปรแกรมได้ฟรีจากเวป http://www.recursionsw.com/Products/voyager.html โปรแกรมที่ทำหน้าที่เป็นเซิร์ฟเวอร์มีชื่อว่า Voyager

Voyager ใช้วิธีการเคลื่อนที่ผ่านการติดต่อแมสเสจระยะไกล (RMI) โดยจะมีการเก็บข้อมูลว่าเอเจนต์ได้เคลื่อนที่ไปยังที่ใด เพื่อให้สามารถติดต่อเอเจนต์โดยวิธีการฟอร์เวิร์ดข้อมูลที่ส่งมายังเครื่องเก่าทั้งหมดไปที่เครื่องใหม่

ในการติดต่อสื่อสารของโมบายเอเจนต์ สามารถติดต่อกันระหว่างเอเจนต์กับเอเจนต์ด้วยกันเองได้ โดยวิธีในการติดต่อสื่อสารได้แก่ ซิงโครไนส์, อะซิงโครไนส์, เมล์บ็อกซ์ (mailbox) และบรอดแคส (broadcast)

ตารางที่ 2.1 ตารางเปรียบเทียบเครื่องมือต่างๆ ที่ใช้ในการพัฒนาโมบายเอเจนต์

Agent System	**Aglets**	**Concordia**	**Odyssey**	**Voyager**
GUI	มี	มี	ไม่มี	มี
Modular design	ไม่มี	มี	ไม่มี	มี
Mobility mechanism	Sockets	RMI	RMI	RMI
Persistence	ไม่มี	มีแต่ไม่ชัดเจน	ไม่มี	มี
Security	Security Manager	Security Manager	Java based	Restricted operation
Direct agent-agent communication	มี	ไม่มี	ไม่มี	มี
Communication type provided	Synchronous Asynchronous Broadcast	Group events Filter events Collaboration	ไม่มี	Synchronous Asynchronous Mailbox Broadcast

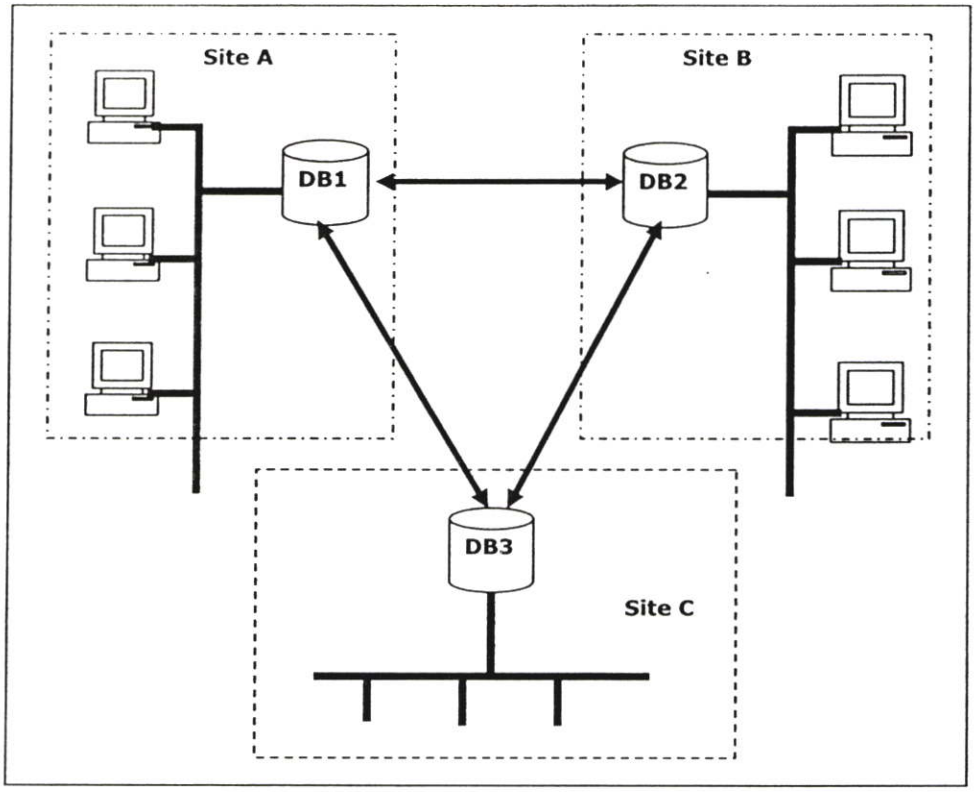
2.2 ระบบฐานข้อมูลแบบกระจาย

2.2.1 ลักษณะของฐานข้อมูลแบบกระจาย

ปัจจุบันมีการใช้งานฐานข้อมูลกันอย่างกว้างขวาง แต่ในบางองค์กรที่มีขนาดใหญ่มาก การนำเอาข้อมูลทั้งหมดของทั้งองค์กรมารวมกันไว้ที่ส่วนกลางแห่งเดียว (Centralize) อาจไม่เหมาะกับการใช้งานจริง เนื่องจากต้องมีค่าใช้จ่ายสำหรับอุปกรณ์ที่ใช้ในการสื่อสารกับส่วนกลาง และขาดความคล่องตัวในการทำงาน จึงเป็นที่มาของระบบฐานข้อมูลแบบกระจาย ซึ่งเป็นการกระจายฐานข้อมูลไปอยู่ยังที่ต่างๆ

ระบบฐานข้อมูลแบบกระจาย (Distributed Database System) เป็นการกระจายการจัดเก็บฐานข้อมูลไว้ในหลายๆ สถานที่ ในที่นี้จะเรียกว่าไซต์ (site) ในแต่ละไซต์จะมีเครื่องคอมพิวเตอร์และระบบฐานข้อมูล (Database Management System: DBMS) เป็นของตัวเอง เพื่อรองรับการใช้

งานจากผู้ใช้ในไซต์นั้นๆ รวมถึงมีการเชื่อมต่อกับฐานข้อมูลที่อยู่ไซต์อื่นผ่านระบบเครือข่าย เพื่อให้ผู้ใช้สามารถใช้งานฐานข้อมูลที่อยู่ที่ไซต์อื่นได้อีกด้วย



รูปที่ 2.5 การเชื่อมต่อฐานข้อมูลเป็นระบบฐานข้อมูลแบบกระจาย

2.2.1.1 ประเภทของระบบฐานข้อมูลแบบกระจาย

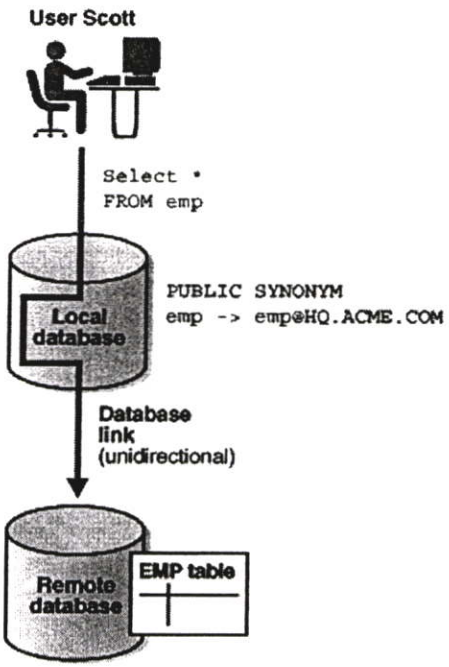
ลักษณะของระบบฐานข้อมูลแบบกระจายแบ่งออกเป็น 2 ลักษณะคือ

1. ระบบเหมือนกัน (Homogenous system) เป็นระบบฐานข้อมูลแบบกระจาย ที่ฐานข้อมูลในแต่ละไซต์เป็นระบบเดียวกัน ดังนั้นรูปแบบของข้อมูลที่ใช้ในแต่ละฐานข้อมูล จึงมีรูปแบบเดียวกัน ส่งผลให้การนำข้อมูลจากแต่ละฐานข้อมูลมาใช้งานไม่ต้องมีการแปลงรูปแบบของข้อมูล จึงกระทำได้ง่าย และมักถูกนำไปใช้ในระบบฐานข้อมูลที่พัฒนาขึ้นใหม่

2. ระบบหลากหลาย (Heterogeneous System) เป็นระบบฐานข้อมูลแบบกระจายที่ฐานข้อมูลของแต่ละไซต์มีการใช้โปรแกรม DBMS ที่มีรูปแบบที่ต่างกัน ซึ่งมักเกิดกับบริษัทที่แต่ละสาขาต่างมีระบบคอมพิวเตอร์เป็นของตนเองอยู่ก่อนแล้ว จึงส่งผลให้แต่ละสาขามีระบบฐานข้อมูลที่มีรูปแบบที่แตกต่างกัน ดังนั้นในการนำข้อมูลจากแต่ละฐานข้อมูลมาใช้งาน จึงต้องอาศัยโปรแกรมที่ทำหน้าที่เป็นตัวกลางเพื่อแปลงรูปแบบข้อมูล (Data Conversion) จากที่ใช้ในฐานข้อมูลหนึ่ง มาอยู่ในรูปแบบที่ใช้ในอีกฐานข้อมูลหนึ่ง

สำหรับหน้าที่ในการเชื่อมต่อฐานข้อมูลในแต่ละไซต์เข้าด้วยกันนั้นเป็นหน้าที่ของผู้ดูแลระบบฐานข้อมูล (Database Administrator: DBA) เมื่อฐานข้อมูลทั้งหมดได้ถูกเชื่อมต่อกันผ่านระบบเครือข่ายแล้ว ผู้ใช้ก็สามารถอ้างถึงตารางในฐานข้อมูลไซต์ใดๆ ก็ได้ แต่ในการอ้างถึงข้อมูลที่อยู่ที่ ไซต์อื่น ผู้ใช้จะต้องระบุด้วยว่าตารางที่ต้องการนั้นอยู่ที่ฐานข้อมูลใด ดังนั้นเพื่อให้ผู้ใช้สามารถใช้งานได้สะดวก และไม่ต้องรับรู้ถึงโครงสร้างของระบบฐานข้อมูลแบบกระจาย ว่าแต่ละตารางนั้นอยู่บนฐานข้อมูลที่ไซต์ใด และเพื่อไม่ให้ผู้ใช้หรือโปรแกรมได้รับผลกระทบหากมีการเปลี่ยนแปลงที่อยู่ทางกายภาพของตาราง หน้าที่อีกอย่างหนึ่งของ DBA ในการดูแลระบบฐานข้อมูลแบบกระจายคือการสร้างชื่อย่อสำหรับเรียกใช้ในการอ้างถึงตารางที่อยู่ในไซต์อื่น เพื่อให้ผู้ใช้เรียกใช้งานได้สะดวกขึ้น

จากรูปที่ 2.6 แสดงการเรียกใช้ข้อมูลจากฐานข้อมูลอื่น โดยผู้ใช้จะบอกเพียงชื่อตารางที่ต้องการ ระบบฐานข้อมูลจะนำชื่อที่ได้ไปค้นหาชื่อจริงของตาราง เพื่อติดต่อไปยังฐานข้อมูลที่มีตารางนั้นๆ อยู่จริง ผ่านทาง Database Link ของ Oracle



รูปที่ 2.6 เป็นการดึงข้อมูลจากฐานข้อมูลที่อยู่ที่เครื่องอื่นผ่าน Database Link

2.2.1.2 คุณสมบัติของระบบฐานข้อมูลแบบกระจาย

ฐานข้อมูลแบบกระจายต้องมีคุณสมบัติต่างๆ ดังนี้

1. Location Transparency เป็นคุณสมบัติที่โปรแกรมเป็นอิสระจากการอ้างถึงฐานข้อมูลในไซต์ต่างๆ กล่าวคือ ผู้พัฒนาโปรแกรมสามารถที่จะใช้เพียงคำสั่งในการเรียกใช้ข้อมูล

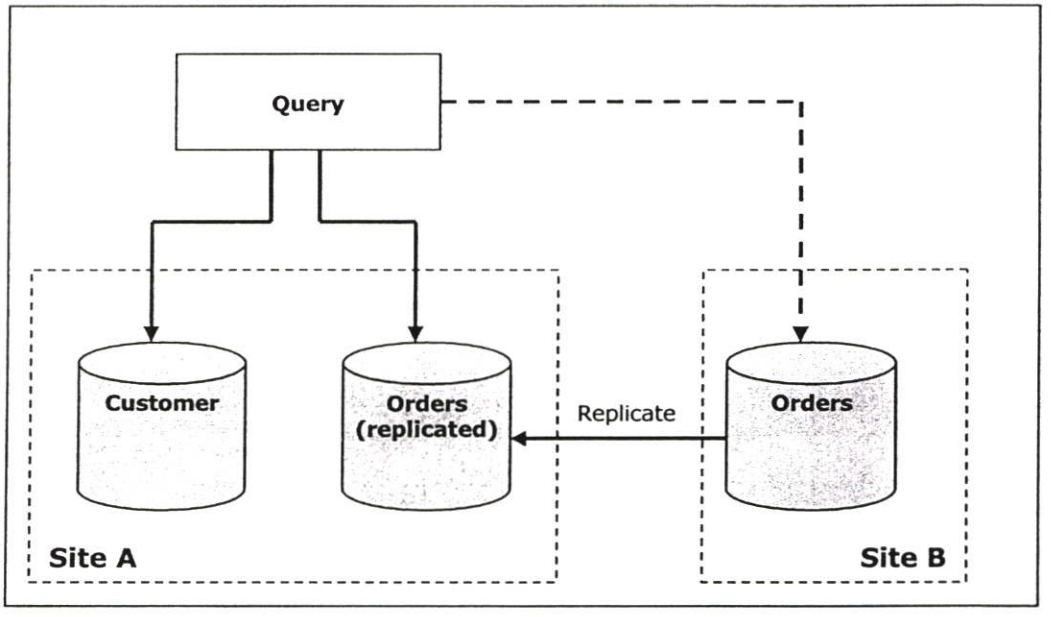
จากฐานข้อมูลในไซต์ต่างๆ ได้โดยไม่จำเป็นที่จะต้องทราบถึงโครงสร้างทางกายภาพของการถ่ายโอนข้อมูลระหว่างฐานข้อมูล จะถูกกำหนดให้เป็นหน้าที่ของระบบฐานข้อมูล

2. Concurrency Control ระบบฐานข้อมูลของฐานข้อมูลแบบกระจาย จะต้องสามารถควบคุมให้เกิดการเปลี่ยนแปลงค่าของข้อมูลชุดเดียวกัน ที่จัดเก็บในฐานข้อมูลของไซต์ต่างๆ ได้อย่างถูกต้อง และครบถ้วน

2.2.1.3 วิธีการจัดเก็บข้อมูลของฐานข้อมูลแบบกระจาย

ในการจัดเก็บข้อมูลของฐานข้อมูลแบบกระจาย มีอยู่ 2 วิธีดังนี้

1. การทำสำเนาข้อมูล (Replication) เป็นการจัดเก็บข้อมูลต่างๆ ที่อยู่ในฐานข้อมูลอื่นไว้ที่ฐานข้อมูลของตน ในการทำสำเนาข้อมูลอาจเป็นการทำสำเนาของทั้งฐานข้อมูล หรือทำสำเนาเพียงบางตารางก็ได้



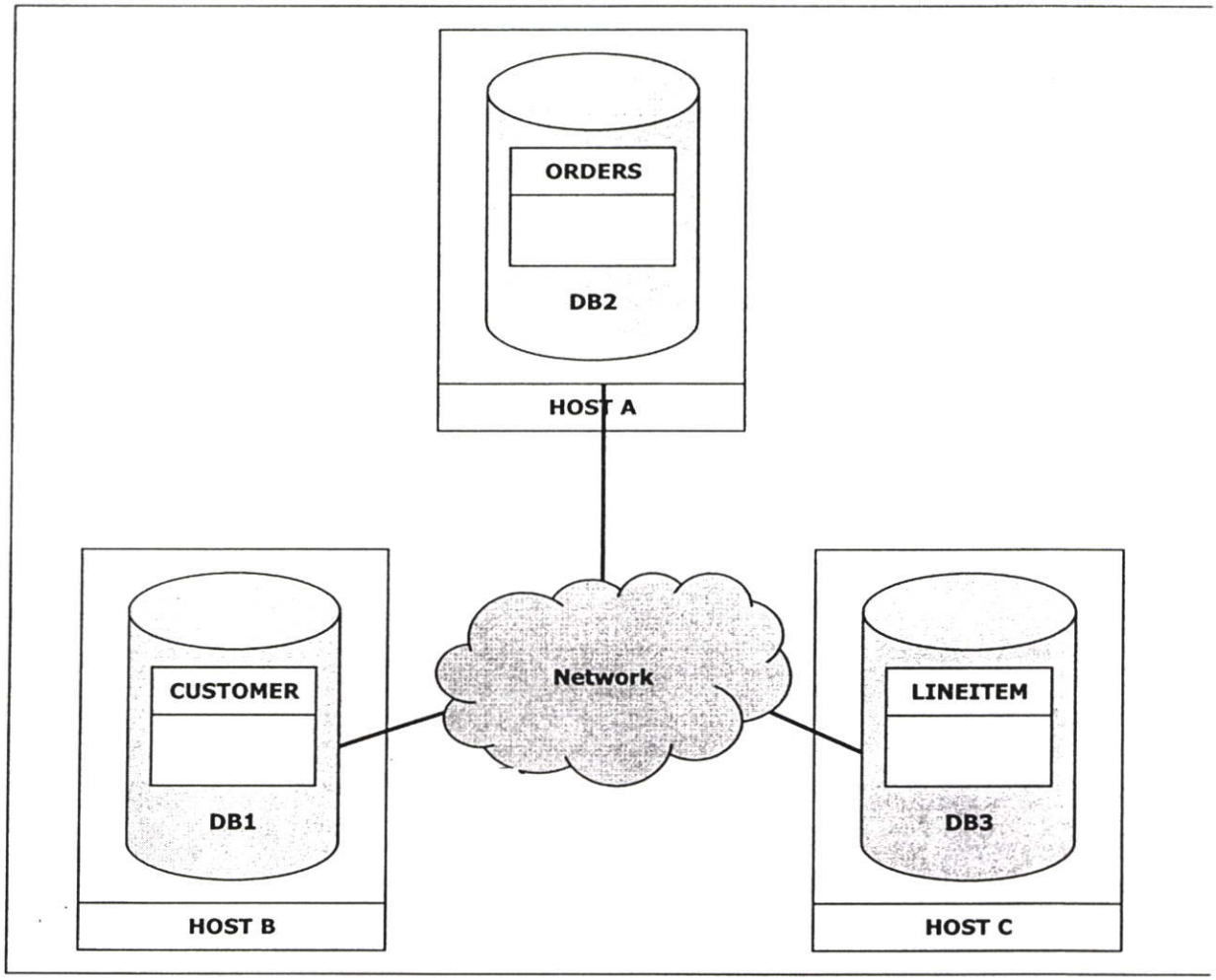
รูปที่ 2.7 แสดงลักษณะการจัดเก็บข้อมูลแบบสำเนาข้อมูล

จากรูปที่ 2.7 เป็นการสำเนาตาราง Orders ที่อยู่ที่ไซต์ B มาไว้ที่ไซต์ A ซึ่งฐานข้อมูลทั้งสองสามารถใช้ข้อมูลในตาราง Orders จากฐานข้อมูลของตนเองได้

วิธีการจัดเก็บในลักษณะนี้ มักใช้กับฐานข้อมูลแบบกระจายแบบหลากหลาย เนื่องจากข้อมูลที่สำเนามาใช้ส่วนใหญ่ มักจะเป็นข้อมูลที่มีการเปลี่ยนแปลงค่าน้อย ดังนั้นจึงสามารถทำการแปลงรูปแบบของข้อมูลได้ในขณะที่ทำสำเนา ส่งผลให้รูปแบบของข้อมูลในแต่ละฐานข้อมูลไม่จำเป็นที่จะต้องมีรูปแบบเดียวกัน

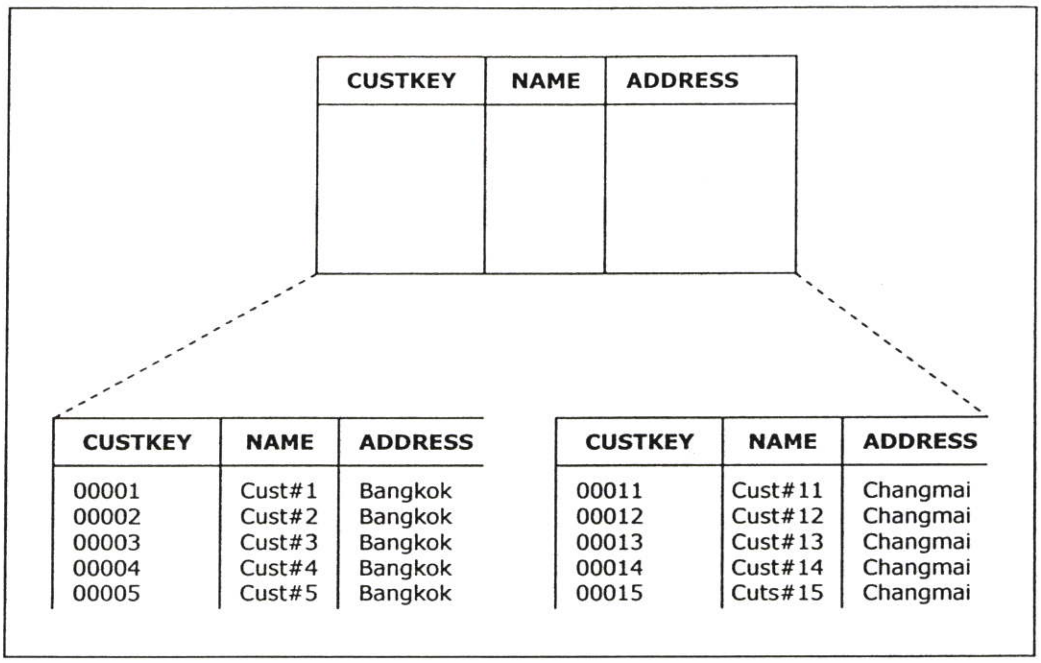
2. การเก็บข้อมูลแบบแยกส่วน (Fragmentation) เป็นวิธีการจัดเก็บข้อมูลด้วยการแบ่งข้อมูลออกเป็นส่วนๆ โดยข้อมูลแบบแยกส่วนเหล่านี้ จะถูกแยกจัดเก็บในฐานข้อมูลของไซต์ต่างๆ สำหรับแบ่งข้อมูลออกเป็นส่วนๆ มีอยู่ 3 วิธีดังนี้

1. กำหนดให้ตารางต่างๆ ในฐานข้อมูลเป็นข้อมูลแบบแยกส่วนเช่น ฐานข้อมูลที่ประกอบด้วยตาราง Customer, Orders และ Lineitem โดยทั้ง 3 ตารางจะถูกกำหนดเป็นข้อมูลแยกส่วนและแยกจัดเก็บอยู่ในฐานข้อมูลของแต่ละไซต์ดังรูปที่ 2.8



รูปที่ 2.8 แสดงการแยกข้อมูลแบบแยกส่วนในแต่ละไซต์

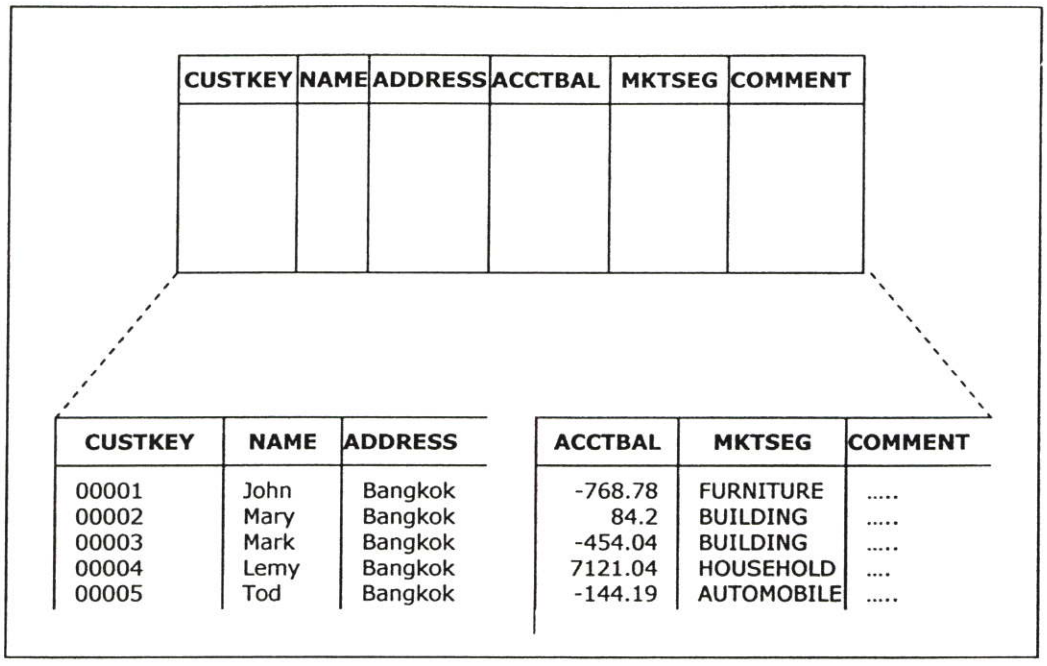
2. แบ่งข้อมูลในตารางออกมาเป็นข้อมูลแบบแยกส่วน โดยแบ่งข้อมูลตามแนวนอน (Horizontal Fragment) โดยแต่ละไซต์จะมีโครงสร้างของตารางหน้าตาเหมือนกัน แต่เงื่อนไขในการเก็บข้อมูลของแต่ละไซต์ต่างกัน ดังรูปที่ 2.9 เป็นการแบ่งข้อมูลจากตาราง Customer แบบแนวนอน โดยกำหนดให้ไซต์แรกเก็บข้อมูลของ Customer ที่มี Address = Bangkok ส่วนไซต์ที่สองจะเก็บข้อมูลของ Customer ที่มี Address = Changmai



รูปที่ 2.9 แสดงการแบ่งข้อมูลในตารางออกมาเป็นข้อมูลแบบแยกส่วนตามแนวนอน

3. แบ่งข้อมูลในตารางออกเป็นข้อมูลแบบแยกส่วนต่างๆ ตามความต้องการในแต่ละไซต์ เช่นเดียวกับวิธีที่ 2 แต่ในการแบ่งตาม จะแบ่งตามคอลัมน์แทน โดยโครงสร้างของตารางเดียวกันที่ฐานข้อมูลแต่ละที่จะมีโครงสร้างไม่เหมือนกัน ดังรูปที่ 2.10 ตาราง Customer ถูกแบ่งตามแนวตั้ง โดยกำหนดให้คอลัมน์ Custkey, Name และ Address ของตาราง Customer อยู่ที่ไซต์ที่ 1 ส่วนคอลัมน์ Acctbak, Mktseg และ Comment อยู่ที่ไซต์ที่ 2 สำหรับการทำแบ่งข้อมูลในลักษณะนี้ถูกเรียกว่าการแบ่งข้อมูลตามแนวนอน (Vertical Fragmentation)

จากทั้ง 3 วิธีจะสังเกตเห็นว่า วิธีที่ง่ายที่สุดในการกำหนดการแบ่งข้อมูลได้แก่ การกำหนดให้แต่ละข้อมูลแยกส่วนเป็นแต่ละฐานข้อมูล ซึ่งด้วยวิธีนี้ อาจก่อให้เกิดต้นทุนในการสื่อสารข้อมูลระหว่างฐานข้อมูลต่างๆ ที่เพิ่มขึ้น แต่ก็สามารถแก้ไขได้โดยการนำเอาข้อมูลแยกส่วนที่มีการใช้งานบ่อย จัดเก็บไว้ในฐานข้อมูลส่วนกลาง หรือใช้การสำเนาข้อมูลแยกส่วนที่มีการใช้งานบ่อย ไว้ในแต่ละฐานข้อมูล



รูปที่ 2.10 แสดงการแบ่งข้อมูลในตารางออกมาเป็นข้อมูลแบบแยกส่วนตามแนวตั้ง

ในการแบ่งข้อมูลในตารางนั้นไม่จำเป็นว่าจะต้องเลือกทำวิธีใดวิธีหนึ่ง สามารถนำเอาหลายๆ วิธีมารวมกันในการแบ่งข้อมูลได้

2.2.2 ปัญหาของระบบฐานข้อมูลแบบกระจาย

ในการใช้งานระบบฐานข้อมูลแบบกระจาย ซึ่งส่วนใหญ่เป็นการดึงข้อมูลจากฐานข้อมูลที่อยู่ที่เครื่องอื่น โดยข้อมูลที่ต้องการจะถูกส่งผ่านระบบเครือข่ายมา ปัญหาสำคัญที่ผู้ใช้พบคือใช้เวลาในการส่งข้อมูลผ่านระบบเครือข่ายนาน ทั้งนี้เนื่องจากขีดจำกัดของความเร็วในระบบเครือข่าย หรือในขณะนั้นมีผู้ใช้ระบบเครือข่ายเป็นจำนวนมากทำให้ต้องแย่งกันใช้ช่องสัญญาณ แต่โดยปกติแล้วความเร็วของดิสก์มักเร็วกว่าความเร็วของเครือข่ายหลายเท่า จึงทำให้ในช่วงที่มีการส่งข้อมูลผ่านเครือข่ายเกิดเป็นปัญหาคอขวด ดังนั้นในการทำงานกับฐานข้อมูลแบบกระจายจึงจำเป็นต้องทำให้มีการใช้เครือข่ายให้น้อยที่สุด เพื่อเป็นการลดเวลาที่ต้องเสียไปในระหว่างที่รอคอยการส่งข้อมูล

2.2.3 การวางแผนคำถาม

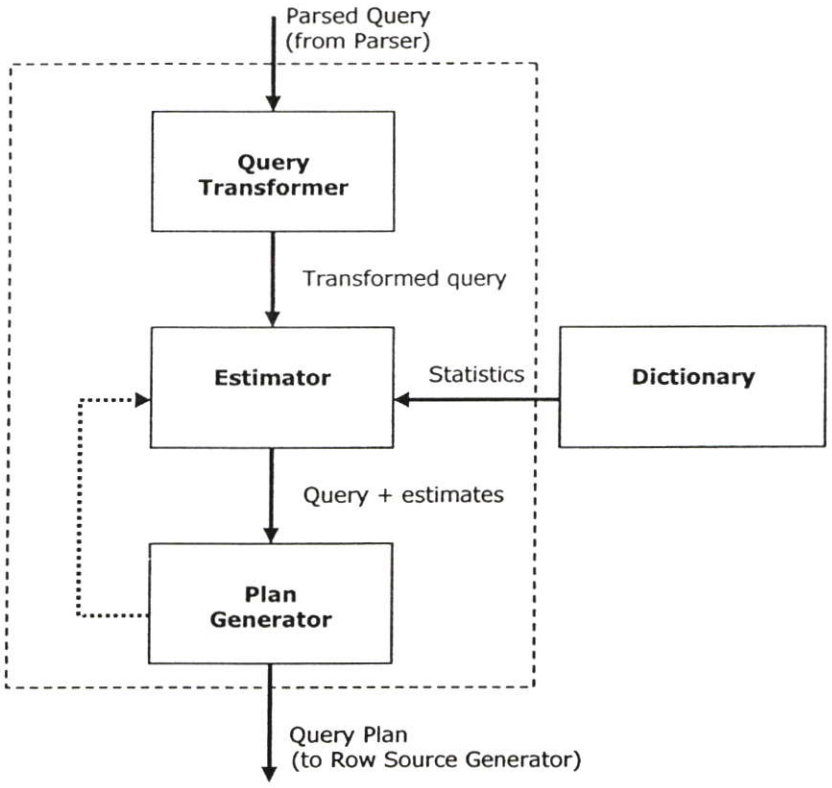
เป็นกระบวนการที่ทำหน้าที่ในการเลือกวิธีในการเข้าถึงข้อมูลที่มีประสิทธิภาพที่สุด วิธีที่ใช้ในการวางแผนคำถามมี 2 แบบ คือ

1. Cost-based Optimization เป็นการพิจารณาว่าแผนการประมวลผลวิธีใดมีประสิทธิภาพมากที่สุด โดยเลือกจากวิธีการเข้าถึงข้อมูล (access paths) ที่มีค่าใช้จ่าย (cost) ในแต่ละขั้นตอนซึ่งได้มาจากข้อมูลสถิติของตารางนั้นๆ

ขั้นตอนการทำงานของ Cost-based optimization เริ่มต้นจากการที่ออพติไมเซอร์สร้างวิธีที่เป็นไปได้ทั้งหมดในการหาผลลัพธ์ จากนั้นออพติไมเซอร์จะทำการประมาณค่าใช้จ่ายของแต่ละวิธี โดยใช้ข้อมูลสถิติ ซึ่งค่าใช้จ่ายประกอบด้วยจำนวนทรัพยากรอันได้แก่ I/O CPU และ memory ที่คาดว่าจะใช้ในการประมวลจริง วิธีที่ออพติไมเซอร์เลือกเพื่อนำไปใช้ในการประมวลผลจริงนั้นคือวิธีที่มีค่าใช้จ่ายรวมต่ำที่สุด

2. Heuristic-based Optimization เป็นวิธีที่พิจารณาว่าแผนการประมวลผลใดมีประสิทธิภาพมากที่สุดโดยการใช้วิธี Heurisitic ในการพิจารณา

สำหรับฐานข้อมูลที่มีใช้ในปัจจุบันจะใช้วิธีการ Cost-based optimization ในการวางแผนคำถาม ซึ่งการวางแผนคำถามประกอบด้วยส่วนต่างๆ ดังรูปที่ 2.11



รูปที่ 2.11 ส่วนประกอบของการวางแผนคำถาม

จะเห็นว่าในการวางแผนคำถามนั้นสิ่งที่สำคัญที่มีผลต่อประสิทธิภาพนั้นคือข้อมูลสถิติซึ่งถูกเก็บอยู่ใน Data Dictionary เนื่องจากออพติไมเซอร์จะใช้ข้อมูลดังกล่าวในการเปรียบเทียบวิธีต่างๆ ก่อนที่จะเลือกว่าจะใช้วิธีใด ดังนั้นหากข้อมูลสถิตินั้นมีความถูกต้อง สอดคล้องกับสภาพข้อมูลจริงก็จะทำให้ออพติไมเซอร์เลือกวิธีการที่ใช้ค่าใช้จ่ายน้อยที่สุดจริงๆ ซึ่งหากข้อมูลดังกล่าวไม่ได้มีการปรับปรุงมาเป็นเวลานาน เมื่อออพติไมเซอร์นำข้อมูลมาใช้ก็จะนำข้อมูลผิดๆ มาใช้ จึงอาจทำให้วิธีที่ออพติไมเซอร์เลือกมานั้นไม่ใช่วิธีที่มีประสิทธิภาพมากที่สุด

สำหรับการวางแผนคำถามกับระบบฐานข้อมูลแบบกระจายจะมีความซับซ้อนมากกว่าการวางแผนคำถามกับฐานข้อมูลเครื่องเดียว เนื่องจากมีปัจจัยอื่นที่มีผล เช่น ความเร็วในการเชื่อมต่อระหว่างฐานข้อมูล ความเร็วในการประมวลผลของแต่ละเครื่อง ฯลฯ ซึ่งในปัจจุบันออราเคิลยังไม่สามารถนำปัจจัยเหล่านี้มาใช้ในการวางแผนของระบบฐานข้อมูลแบบกระจายได้ เมื่อออราเคิลรับคำสั่งมา ออราเคิลจะแบ่งคำสั่งออกเป็นคำสั่งย่อย แล้วส่งคำสั่งย่อยไปยังฐานข้อมูลอื่นเพื่อให้ส่งผลลัพธ์กลับมา เมื่อได้ผลลัพธ์ทั้งหมดแล้ว จึงจะนำผลลัพธ์มารวมกันที่เครื่องที่รับคำสั่งมาเท่านั้น โดยไม่มีการคำนึงถึงปริมาณข้อมูลที่ส่งผ่านเครือข่าย

2.2.4 สถิติและการวิเคราะห์สถิติ

2.2.4.1 ความหมายของสถิติ

สถิติ (Statistics) เป็นข้อมูลที่ถูกเก็บอยู่ใน Data Dictionary ใช้บอกถึงรายละเอียดและลักษณะของข้อมูลต่างๆ ในฐานข้อมูล ถูกใช้โดยออพติไมเซอร์ โดยใช้ข้อมูลสถิติเพื่อตัดสินใจเลือกแผนที่ดีที่สุดสำหรับการประมวลผลคำถาม ข้อมูลสถิติประกอบด้วย

- สถิติเกี่ยวกับตาราง
  - จำนวนข้อมูลทั้งหมดในตาราง
  - จำนวนบล็อกทั้งหมดในตาราง
  - ความยาวของแถวโดยเฉลี่ย
- สถิติเกี่ยวกับคอลัมน์
  - จำนวนค่าที่แตกต่างกันทั้งหมดในคอลัมน์
  - จำนวนที่เป็นนัลในคอลัมน์
  - การกระจายตัวของข้อมูล
- สถิติเกี่ยวกับอินเด็กซ์
  - จำนวนบล็อกที่เป็น leaf node
  - จำนวนชั้น (Level)
  - Clustering factor
- สถิติของระบบ
  - ประสิทธิภาพและการใช้งานส่วนอินพุตเอ้าพุต (I/O)
  - ประสิทธิภาพและการใช้งานหน่วยประมวลผล (CPU)

ในการเก็บสถิติในฐานข้อมูลหลายๆ ตัวเช่นออราเคิล จะใช้วิธีเรียกฟังก์ชันของระบบฐานข้อมูลที่ใช้สำหรับเก็บข้อมูลสถิติและปรับปรุงสถิติ แต่ในการทำงานเพื่อเก็บข้อมูลและปรับปรุงสถิติของระบบฐานข้อมูลเหล่านั้นมีข้อเสียหลายอย่าง โดยข้อเสียหลักๆ ได้แก่

● ในการเก็บข้อมูลสถิติจำเป็นต้องใช้ทรัพยากรในการทำงานเป็นจำนวนมาก ทำให้ไปแย่งใช้ทรัพยากรที่ควรจะนำไปใช้อย่างอื่นจนทำให้ระบบทำงานได้ช้า

● ในกรณีที่ไม่ต้องการให้ระบบต้องทำงานมากเกินไปก็ไม่ควรจะเรียกใช้ฟังก์ชันเพื่อเก็บข้อมูลสถิติบ่อยเกินไป ซึ่งก็อาจเป็นเหตุให้ข้อมูลสถิตินั้นเป็นข้อมูลที่ไม่สอดคล้องกับข้อมูลจริง และหากออพติไมเซอร์นำข้อมูลไปใช้ก็จะทำให้ขบวนการในการประมวลผลคำถามไม่มีประสิทธิภาพ

● ในบางฐานข้อมูลที่มีผู้ใช้มีการกำหนดประเภทของค่าใช้จ่ายขึ้นมาเพื่อใช้งานเองนั้น ฟังก์ชันที่ใช้เรียกเก็บข้อมูลสถิติไม่สามารถเก็บข้อมูลสถิติในส่วนนี้ได้ จึงทำให้ข้อมูลสถิติที่ได้ไม่ครบถ้วนตามที่ต้องการ

● สำหรับฐานข้อมูลที่มีขนาดยิ่งใหญ่ก็จะยิ่งมีค่าใช้จ่ายในการวิเคราะห์สถิติมาก และหากฐานข้อมูลนั้นเป็นฐานข้อมูลที่ไม่มีการเปลี่ยนแปลงของข้อมูล การวิเคราะห์สถิติถือเป็นเรื่องที่ทำอย่างเปล่าประโยชน์

● การที่ให้ผู้ใช้เป็นคนเรียกฟังก์ชันเพื่อเก็บข้อมูลสถิติเมื่อมีการเปลี่ยนแปลงของฐานข้อมูลที่มีความสำคัญเท่านั้น ทำให้ข้อมูลสถิติที่ออพติไมเซอร์ ใช้ในการวางแผนคำถามนั้นเป็นข้อมูลที่ไม่เป็นปัจจุบัน

2.2.4.2 การวิเคราะห์สถิติ

ในการเก็บสถิติ เนื่องจากไม่มีความจำเป็นต้องทำการวิเคราะห์สถิติทุกครั้งที่มีการเปลี่ยนแปลงข้อมูลในฐานข้อมูล เพราะอาจไม่มีความเปลี่ยนแปลงที่มีผลกับฐานข้อมูล แต่หากไม่มีการวิเคราะห์สถิติก็อาจทำให้ได้สถิติที่ไม่สอดคล้องกับฐานข้อมูล ณ เวลานั้น จึงมีคำถามที่น่าสนใจว่าเวลาที่เหมาะสมที่จะเก็บสถิติควรเป็นเวลาใด และควรใช้วิธีการเก็บสถิติแบบใด เพื่อให้มีผลกระทบกับการทำงานปกติให้น้อยที่สุดแต่ได้สถิติที่มีความสอดคล้องกับข้อมูลในฐานข้อมูลจริงๆ สำหรับวิธีที่ง่ายที่สุดในการเลือกช่วงเวลาในการเก็บสถิติ อาจใช้วิธีการกำหนดว่า ทุกๆ ช่วง 100 คำสั่งให้ทำการเก็บสถิติ แต่วิธีนี้ก็ยังไม่มีประสิทธิภาพเพียงพอ เนื่องจากข้อมูลสถิติไม่ได้มีความเปลี่ยนแปลงเท่ากับจำนวนคิวรี่ เพราะถึงแม้ว่าจะมีคำสั่งเข้ามามากแต่ไม่ได้หมายความว่าข้อมูลในตารางมีการเปลี่ยนแปลงมากตามไปด้วย และในทางกลับกันบางครั้งการมีคำสั่งมาเพียงไม่กี่คำสั่งก็อาจส่งผลให้ข้อมูลมีการเปลี่ยนแปลงอย่างมาก อีกทางเลือกหนึ่งสำหรับคือให้ผู้ดูแลระบบฐานข้อมูลเป็นคนสั่งให้ทำเก็บสถิติในเวลาที่ต้องการเอง แต่ข้อเสียของวิธีนี้เป็นการเพิ่มภาระให้กับผู้ดูแลระบบฐานข้อมูลเพราะจะต้องเป็นคนดูเองว่าเวลาที่จะเก็บสถิตินี้มีผู้ใช้ใช้งานฐานข้อมูลอยู่หรือไม่

2.2.5 การเก็บสถิติ

ในออราเคิลมีคำสั่งที่ใช้ในการเก็บสถิติแบ่งออกเป็นระดับต่างๆ ในการเก็บสถิติคือ

2.2.5.1 การเก็บสถิติของทั้งฐานข้อมูล

เป็นการสั่งให้เก็บสถิติของตารางทุกตารางในฐานข้อมูลที่มีการเปลี่ยนแปลง โดยรูปแบบของคำสั่งมีดังนี้

```
DBMS_STATS.GATHER_DATABASE_STATS (
  estimate_percent      NUMBER DEFAULT to_estimate_percent_type
                        (get_param('ESTIMATE_PERCENT')),
  block_sample          BOOLEAN DEFAULT FALSE,
  method_opt            VARCHAR2 DEFAULT get_param
                        ('METHOD_OPT'),
  degree                NUMBER DEFAULT to_degree_type(get_param
                        ('DEGREE')),
  granularity           VARCHAR2 DEFAULT 'AUTO',
  cascade               BOOLEAN DEFAULT to_cascade_type(get_param
                        ('CASCADE')),
  stattab               VARCHAR2 DEFAULT NULL,
  statid                VARCHAR2 DEFAULT NULL,
  options               VARCHAR2 DEFAULT 'GATHER',
  objlist               OUT ObjectTab,
  statown               VARCHAR2 DEFAULT NULL,
  gather_sys            BOOLEAN DEFAULT FALSE,
  no_invalidate         BOOLEAN DEFAULT to_no_invalidate_type (
                        get_param('NO_INVALIDATE')));
```

2.2.5.2 การเก็บสถิติเฉพาะอินเด็กซ์ที่กำหนด

เป็นการเก็บเฉพาะสถิติที่เกี่ยวกับอินเด็กซ์ทั้งหมดในฐานข้อมูล

รูปแบบคำสั่งที่ใช้ในการเก็บสถิติมีดังนี้

```
DBMS_STATS.GATHER_INDEX_STATS (
  ownname               VARCHAR2,
  indname               VARCHAR2,
  partname              VARCHAR2 DEFAULT NULL,
  estimate_percent      NUMBER DEFAULT to_estimate_percent_type
                        (get_param('ESTIMATE_PERCENT')),
```

```
stattab          VARCHAR2 DEFAULT NULL,
statid           VARCHAR2 DEFAULT NULL,
statown          VARCHAR2 DEFAULT NULL,
degree           NUMBER DEFAULT to_degree_type(get_param
                 ('DEGREE')),
granularity      VARCHAR2 DEFAULT 'AUTO',
no_invalidate    BOOLEAN DEFAULT to_no_invalidate_type (
                 get_param('NO_INVALIDATE')));
```

2.2.5.3 การเก็บสถิติเฉพาะ Schema ที่กำหนด

เป็นการสั่งให้เก็บสถิติเฉพาะตารางที่อยู่ใน schema ที่กำหนด และตารางนั้นมีการเปลี่ยนแปลงของข้อมูล โดยมีรูปแบบของคำสั่งดังนี้

```
DBMS_STATS.GATHER_SCHEMA_STATS (
ownname          VARCHAR2,
estimate_percent NUMBER DEFAULT to_estimate_percent_type
                 (get_param('ESTIMATE_PERCENT')),
block_sample     BOOLEAN DEFAULT FALSE,
method_opt       VARCHAR2 DEFAULT get_param
                 ('METHOD_OPT'),
degree           NUMBER DEFAULT to_degree_type(get_param
                 ('DEGREE')),
granularity      VARCHAR2 DEFAULT 'AUTO',
cascade          BOOLEAN DEFAULT to_cascade_type(get_param
                 ('CASCADE')),
stattab          VARCHAR2 DEFAULT NULL,
statid           VARCHAR2 DEFAULT NULL,
options          VARCHAR2 DEFAULT 'GATHER',
statown          VARCHAR2 DEFAULT NULL,
no_invalidate    BOOLEAN DEFAULT to_no_invalidate_type (
                 get_param('NO_INVALIDATE')));
```

2.2.5.4 การเก็บสถิติของระบบ

เป็นการ โดยมีรูปแบบคำสั่งที่ใช้ดังนี้

```
DBMS_STATS.GATHER_SYSTEM_STATS (
  gathering_mode        VARCHAR2 DEFAULT 'NOWORKLOAD',
  interval              INTEGER DEFAULT NULL,
  stattab               VARCHAR2 DEFAULT NULL,
  statid                VARCHAR2 DEFAULT NULL,
  statown               VARCHAR2 DEFAULT NULL);
```

2.2.5.6 การเก็บสถิติเฉพาะตารางที่กำหนด

เป็นการเลือกเก็บสถิติโดยระบุชื่อตารางที่ต้องการเก็บสถิติ (ถ้าตารางนั้นมีการเปลี่ยนแปลง) โดยมีรูปแบบคำสั่งที่ใช้ดังนี้

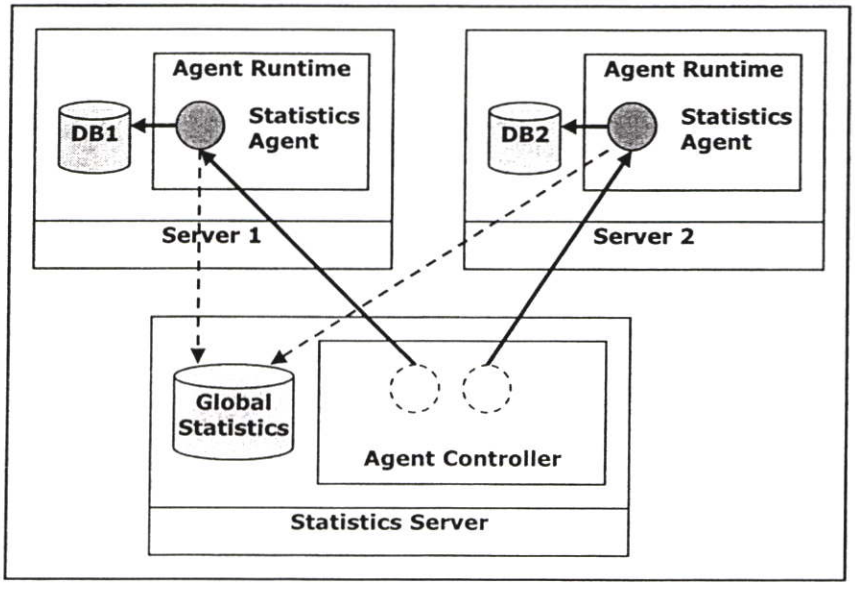
```
DBMS_STATS.GATHER_TABLE_STATS (
  ownname               VARCHAR2,
  tabname               VARCHAR2,
  partname              VARCHAR2 DEFAULT NULL,
  estimate_percent      NUMBER DEFAULT to_estimate_percent_type
                        (get_param('ESTIMATE_PERCENT')),
  block_sample          BOOLEAN DEFAULT FALSE,
  method_opt            VARCHAR2 DEFAULT get_param
                        ('METHOD_OPT'),
  degree                NUMBER DEFAULT to_degree_type(get_param
                        ('DEGREE')),
  granularity           VARCHAR2 DEFAULT 'AUTO',
  cascade               BOOLEAN DEFAULT to_cascade_type(get_param
                        ('CASCADE')),
  stattab               VARCHAR2 DEFAULT NULL,
  statid                VARCHAR2 DEFAULT NULL,
  statown               VARCHAR2 DEFAULT NULL,
  no_invalidate         BOOLEAN DEFAULT to_no_invalidate_type (
                        get_param('NO_INVALIDATE')));
```

บทที่ 3
โมบายเอเจนต์ในการเก็บสถิติบนระบบฐานข้อมูลแบบกระจาย

3.1 การใช้โมบายเอเจนต์ในการปรับปรุงสถิติของแต่ละฐานข้อมูล

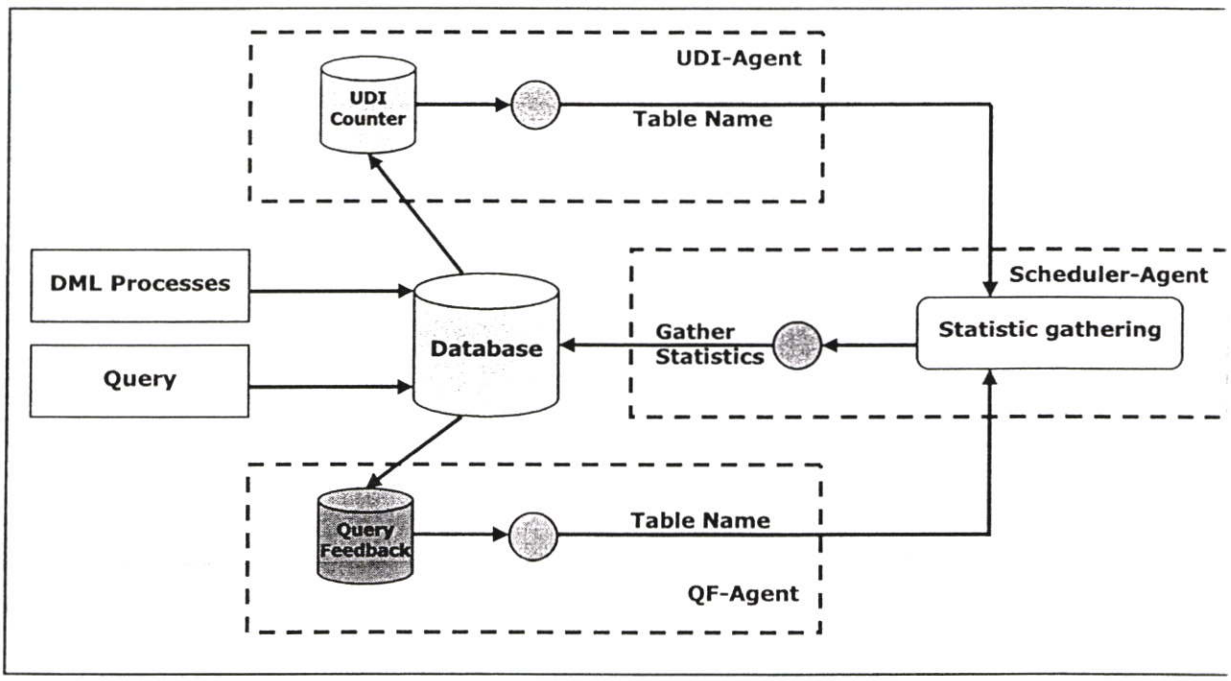
การเก็บสถิติเป็นงานที่จำเป็นต้องทำบนเครื่องที่มีฐานข้อมูลอยู่เท่านั้น ดังนั้นในระบบฐานข้อมูลแบบกระจายจึงจำเป็นต้องให้แต่ละฐานข้อมูลดูแลงานในการเก็บสถิติเอง ซึ่งหากมีฐานข้อมูลบางแห่งที่บริหารจัดการเก็บสถิติได้ไม่ดีพอ เมื่อออพติไมเซอร์นำสถิติที่ไม่ถูกต้องนั้นมาใช้ในการวางแผนคำถามทำให้การประมาณค่าใช้จ่ายมีผิดพลาด และทำให้วิธีที่ออพติไมเซอร์เลือกเป็นวิธีที่ไม่มีประสิทธิภาพ มีผลให้ภาพรวมของประสิทธิภาพการทำงานของทั้งระบบลดลง ในงานวิจัยนี้จึงได้นำโมบายเอเจนต์มาช่วยในการวิเคราะห์และเก็บสถิติของแต่ละฐานข้อมูล เพื่อให้ทุกๆ ฐานข้อมูลมีสถิติที่สอดคล้องกับข้อมูลจริงอยู่เสมอ และยังเป็นการช่วยลดภาระงานของผู้ดูแลระบบในการจัดการเก็บสถิติเองอีกด้วย

โมบายเอเจนต์ที่ใช้ในการเก็บสถิติประกอบด้วยเอเจนต์คอนโทรลเลอร์ (Agent Controller) และเอเจนต์รันไทม์ (Agent Runtime) โดยเอเจนต์คอนโทรลเลอร์เป็นโปรแกรมที่ใช้สำหรับส่งโมบายเอเจนต์ไปทำงานยังเครื่องต่างๆ สามารถกำหนดให้เครื่องใดก็ได้เป็นเอเจนต์คอนโทรลเลอร์ ซึ่งเครื่องที่เป็นเอเจนต์คอนโทรลเลอร์จะต้องมีการติดตั้งโปรแกรมเอเจนต์คอนโทรลเลอร์ก่อน ส่วนเครื่องที่จะส่งโมบายเอเจนต์ไปทำงานจะต้องมีโปรแกรมเอเจนต์รันไทม์ทำงานอยู่เสมอ



รูปที่ 3.1 ภาพรวมการทำงานของโมบายเอเจนต์ในการเก็บสถิติ

เมื่อโมบายเอเจนต์เคลื่อนที่ไปถึงไซต์เป้าหมายแล้ว โมบายเอเจนต์จะแบ่งการทำงานออกเป็นเอเจนต์ย่อย 3 ตัว คือ UDI-Agent, QF-Agent และ Scheduler-Agent ซึ่งเอเจนต์แต่ละตัวจะมีหน้าที่ที่แตกต่างกัน สามารถทำงานได้พร้อมๆ กันหมด ดังรูปที่ 3.2



รูปที่ 3.2 ส่วนประกอบของ Statistics Agent ในการเก็บสถิติ

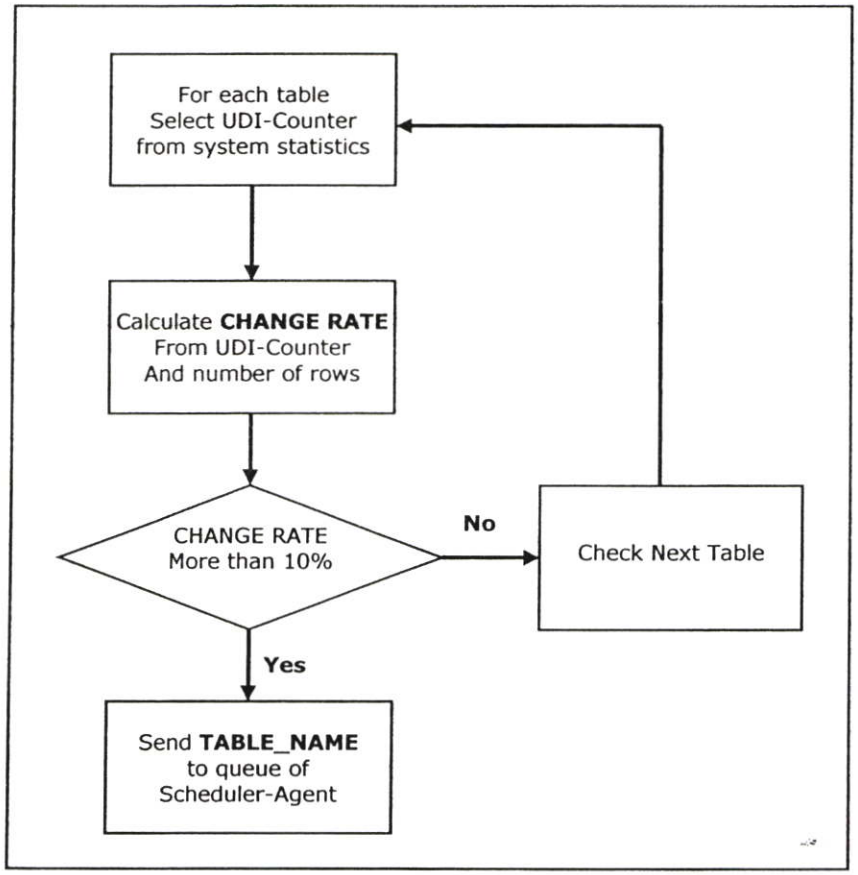
UDI-Agent และ QF-Agent เป็นเอเจนต์ที่ทำหน้าที่ในวิเคราะห์สถิติ โดย UDI-Agent จะวิเคราะห์สถิติจากปริมาณการเปลี่ยนแปลงของข้อมูลในแต่ละตาราง ส่วน QF-Agent จะวิเคราะห์สถิติจากประสิทธิภาพในการวางแผนคำถามของ Optimizer โดยตารางทั้งหมดที่ถูก UDI-Agent และ QF-Agent วิเคราะห์แล้วว่าควรเก็บสถิติจะเป็นหน้าที่ของ Scheduler-Agent ในการจัดลำดับความสำคัญในการเก็บสถิติ และเลือกเวลาที่เหมาะสมในการเก็บสถิติ

3.2 การวิเคราะสถิติด้วยโมบายเอเจนต์

เนื่องจากงานในการเก็บสถิติเป็นงานที่ใช้ทรัพยากรในการทำงานสูงมาก เพื่อไม่ให้การเก็บสถิติต้องมีผลกระทบต่อผู้ใช้งานหรือโปรแกรมอื่นที่ใช้งานฐานข้อมูลนั้นอยู่ ในการเก็บสถิติโมบายเอเจนต์จะเลือกเก็บสถิติเฉพาะตารางที่จำเป็นต้องเก็บสถิติจริงๆ เท่านั้น ซึ่งวิธีในการวิเคราะห์แบ่งได้ 2 แบบคือ

3.2.1 การวิเคราะห์สถิติจากจำนวนการเปลี่ยนแปลงของข้อมูล

ในการใช้งานฐานข้อมูลมักมีการเพิ่มข้อมูล ลบข้อมูล หรือมีการเปลี่ยนแปลงข้อมูลอยู่ตลอดเวลา จึงควรปรับปรุงให้สถิติมีความสอดคล้องกับการเปลี่ยนแปลงของข้อมูลอยู่เสมอ ในการวิเคราะห์สถิติจากจำนวนการเปลี่ยนแปลงของข้อมูลนี้จะใช้ UDI-Agent ในการทำงาน UDI-Agent จะดูข้อมูลที่บันทึกจำนวนการเปลี่ยนแปลงของแต่ละตาราง แล้วนำค่าการเปลี่ยนแปลงมาเทียบกับจำนวนข้อมูลทั้งหมดในตาราง เพื่อหาออกมาเป็นเปอร์เซ็นต์การเปลี่ยนแปลง โดยจะทำงานเป็นช่วงเวลา โดยมีขั้นตอนการทำงานดังรูปที่ 3.3



รูปที่ 3.3 ขั้นตอนการวิเคราะห์สถิติของ UDI-Agent

ในขั้นตอนแรกเอเจนต์จะดึงจำนวนการเปลี่ยนแปลงของแต่ละตารางในฐานข้อมูล ซึ่งในออราเคิลจะมีตาราง ALL_TAB_MODIFICATIONS ที่คอยเก็บจำนวนการเปลี่ยนแปลงของข้อมูล โดยแยกประเภทของการเปลี่ยนแปลงออกเป็นการเพิ่มข้อมูล (insert) เปลี่ยนแปลงข้อมูล (update) และลบข้อมูล (delete) ค่าที่แสดงของแต่ละประเภทการเปลี่ยนแปลงจะเริ่มต้นนับจากครั้งล่าสุดที่ได้ทำการเก็บสถิติของตารางนั้นไป โดยมี UDI-Agent เป็นเอเจนต์ที่ทำหน้าที่เฝ้าดูจำนวนการเปลี่ยนแปลงของข้อมูล แล้วนำค่าที่ได้ไปวิเคราะห์เพื่อหาว่าตารางใดจำเป็นต้องเก็บสถิติใหม่

สำหรับคำสั่งที่เอเจนต์ใช้เพื่อดึงจำนวนการเปลี่ยนแปลงของข้อมูลคือ

```
SELECT    T1.TABLE_NAME, T1.INSERTS, T1.UPDATES, T1.DELETES,
          T2.NUM_ROWS, T2.LAST_ANALYZED, T2.OWNER
FROM      USER_TAB_MODIFICATIONS T1, ALL_TABLES T2
WHERE     T1.TABLE_NAME=T2.TABLE_NAME AND DROPPED='NO'
ORDER BY  TABLE_NAME DESC;
```

ตารางที่ 3.1 ตัวอย่างข้อมูลการเปลี่ยนแปลงของข้อมูลในแต่ละตาราง

TABLE_ NAME	INSERTS	UPDATES	DELETES	NUM_ROWS	LAST_ ANALYZED
PART	5	2	0	20	14-MAR-07
SUPPLIER	10	0	0	10	

เมื่อเอเจนต์ได้จำนวนการเปลี่ยนแปลงของข้อมูลแล้ว ก็จะนำข้อมูลมาหาเปอร์เซ็นต์การเปลี่ยนแปลง โดยวิธีการคำนวณหาเปอร์เซ็นต์การเปลี่ยนแปลงได้จากสูตร

$$\text{เปอร์เซ็นต์การเปลี่ยนแปลง} = \frac{(\text{INSERTS} + \text{UPDATES} + \text{DELETES}) \times 100}{\text{NUM\_ROWS}} \qquad (3.1)$$

สำหรับตารางใดๆ ก็ตามที่มีค่าเปอร์เซ็นต์การเปลี่ยนแปลงมากกว่าหรือเท่ากับค่าที่กำหนด ซึ่งในที่นี้จะกำหนดไว้ที่ 10% UDI-Agent จะถือว่าตารางนั้นมีการเปลี่ยนแปลงของข้อมูลมากพอที่ต้องเก็บสถิติของตารางนั้นใหม่ และส่งข้อมูลของตารางนั้นเข้าไปที่คิวของ Scheduler-Agent เพื่อรอเวลาในการเก็บสถิติ

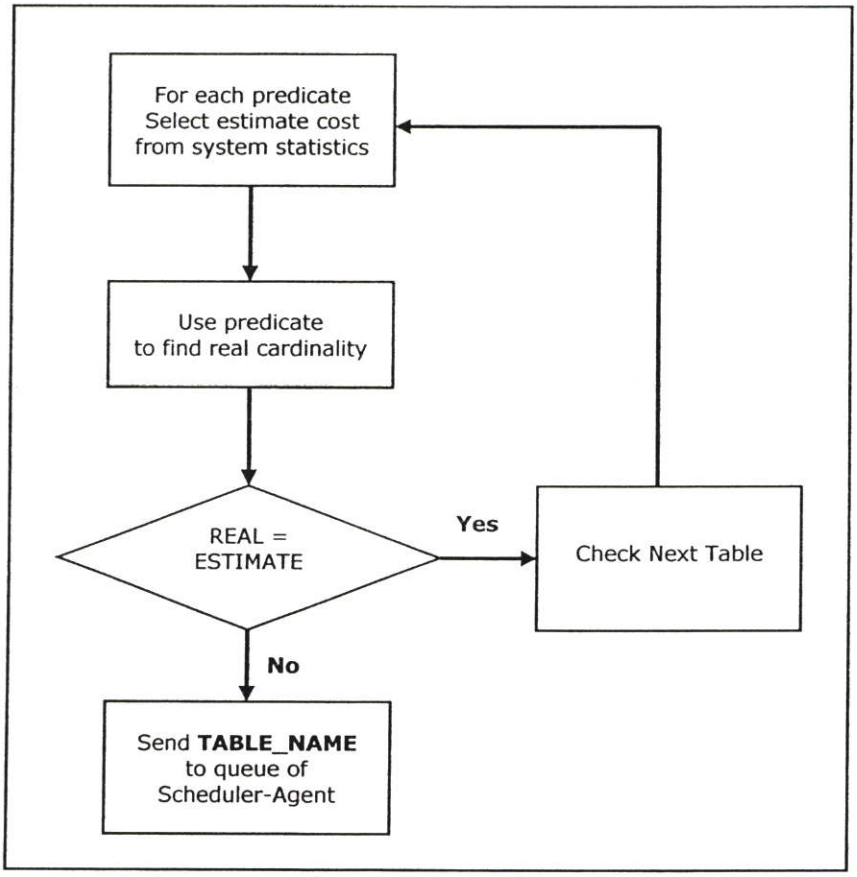
ตารางที่ 3.2 ตัวอย่างการคำนวณค่าเปอร์เซ็นต์การเปลี่ยนแปลงของข้อมูล

ชื่อตาราง	เปอร์เซ็นต์การเปลี่ยนแปลง
PART	((5+2+0)*100)/20 = 35%
SUPPLIER	((10+0+0*100))/10 = 100%

3.2.2 การวิเคราะห์สถิติจากประสิทธิภาพในการวางแผนคำถาม

เป็นการวิเคราะห์สถิติจากข้อมูลที่ออพติไมเซอร์ใช้ในการวางแผนคำถาม เพื่อตรวจสอบดูว่าค่าที่ออพติไมเซอร์ประมาณมานั้นตรงกับความเป็นจริงหรือไม่ เนื่องจากค่าที่ออพติไมเซอร์

ประมาณนั้นได้มาจากค่าสถิติ ดังนั้นเมื่อค่าที่ประมาณไม่ตรงกับความเป็นจริงก็แสดงว่าข้อมูลสถิติที่ใช้อยู่น่าจะมีปัญหา ซึ่งการวิเคราะห์ส่วนนี้เป็นหน้าที่ของ QF-Agent โดยดึงค่าที่ออพติไมเซอร์ประมาณในการวางแผนคำถามแต่ละคำถามออกมา แล้วดึงจำนวนข้อมูลที่มีอยู่จริงมาเปรียบเทียบกัน หากค่าที่ออพติไมเซอร์ประมาณออกมาไม่เท่ากับจำนวนข้อมูลจริง QF-Agent จะถือว่าตารางนั้นจำเป็นต้องเก็บสถิติ แล้วส่งข้อมูลของตารางไปยังคิวของ Scheduler-Agent โดยมีขั้นตอนการทำงานดังรูปที่ 3.4



รูปที่ 3.4 ขั้นตอนการวิเคราะห์สถิติของ QF-Agent

ขั้นแรกเอเจนต์จะดึงข้อมูลที่ออพติไมเซอร์เคยประมาณค่าของแต่ละเงื่อนไขออกมาก่อน โดยใช้คำสั่งในการดึงข้อมูลดังนี้

```
SELECT      OBJECT_OWNER, OBJECT_NAME,
            CARDINALITY, FILTER_PREDICATES
FROM        PLAN_TABLE
WHERE       OBJECT_NAME IS NOT NULL AND
            FILTER_PREDICATES IS NOT NULL
```

ตารางที่ 3.3 ตัวอย่างข้อมูลที่ได้จากออพติไมเซอร์ประมาณของแต่ละเงื่อนไข

OBJECT_OWNER	OBJECT_NAME	CARDINALITY	FILTER_PREDICATES
ORCL	REGION	1	R_NAME"='ASIA'
ORCL	PART	52	"P_SIZE"=28 AND "P_TYPE" LIKE '%LARGE%'

เมื่อได้ข้อมูลที่ได้จากการประมาณมา เอเจนต์จะนำข้อมูลดังกล่าวมามาเป็นข้อมูลสำหรับใช้ในการหาค่าจำนวนข้อมูลจริง โดยใช้เงื่อนไขที่กำหนดมาเป็นเงื่อนไขที่ใช้ในการนับจำนวนข้อมูลจริงในตาราง โดยใช้คำสั่ง

SELECT COUNT(*)

FROM <OBJECT_OWNER>.<OBJECT_NAME>

WHERE <FILTER_PREDICATES>

ตารางที่ 3.4 ตัวอย่างข้อมูลที่ได้จากการเปรียบเทียบ

OBJECT_OWNER	OBJECT_NAME	CARDINALITY	FILTER_PREDICATES	REAL
ORCL	REGION	1	R_NAME"='ASIA'	1
ORCL	PART	52	"P_SIZE"=28 AND "P_TYPE" LIKE '%LARGE%'	7

เมื่อได้ข้อมูลทั้งที่ออพติไมเซอร์เป็นตัวประมาณค่าและข้อมูลจริงแล้ว เอเจนต์จะนำค่าทั้งสองมาเปรียบเทียบกัน หากตารางใดมีอย่างน้อย 1 เงื่อนไขในตารางที่มีค่าที่ออพติไมเซอร์ประมาณมาไม่เท่ากับค่าจริงของข้อมูล แสดงว่าข้อมูลสถิติที่มีไม่สามารถใช้ในการประมาณค่าของออพติไมเซอร์ได้อย่างถูกต้อง จึงต้องมีการเก็บสถิติของตารางนั้นใหม่ โดยส่งข้อมูลของตารางดังกล่าวให้กับ Scheduler-Agent เพื่อเก็บไว้ในคิวรอการเก็บสถิติต่อไป

3.3 การจัดลำดับความจำเป็นในการเก็บสถิติ

แต่ละตารางที่ต้องเก็บสถิติมีระดับความสำคัญในการเก็บสถิติไม่เท่ากัน จึงเป็นหน้าที่ของ Scheduler-Agent ที่ต้องจัดลำดับความสำคัญของตารางที่ต้องการเก็บสถิติ เพื่อให้ตารางที่

จำเป็นต้องเก็บสถิติมากที่สุดได้เก็บสถิติก่อน ในการจัดระดับความจำเป็นในการเก็บสถิติ Scheduler-Agent ได้แบ่งระดับความสำคัญออกเป็น 5 ระดับดังนี้

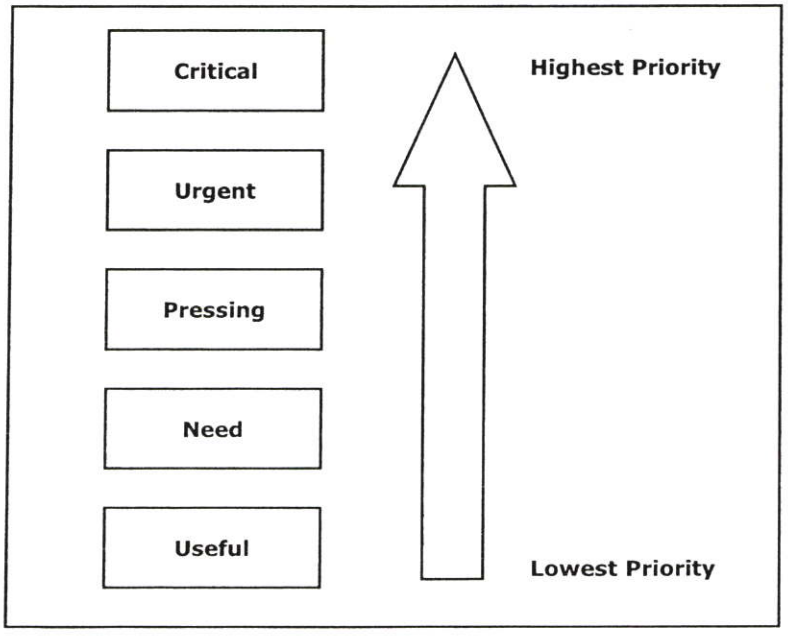
1. **Useful** เป็นตารางที่มีความจำเป็นในการเก็บสถิติน้อยที่สุด โดยตารางที่อยู่ในระดับความจำเป็นนี้เป็นตารางที่ UDI-Agent พบว่ามีเปอร์เซ็นต์การเปลี่ยนแปลงของข้อมูลมากกว่าค่าที่กำหนด แต่ไม่เกิน 50% ของข้อมูลทั้งหมดในตาราง และเป็นตารางที่ QF-Agent ไม่พบความแตกต่างระหว่างค่าที่ออพติไมเซอร์ประมาณกับค่าจริง

2. **Need** เป็นตารางที่มีความจำเป็นในการเก็บสถิติมากเป็นลำดับถัดมา เป็นตารางที่ QF-Agent พบว่ามีอย่างน้อยหนึ่งคอลัมน์ที่ออพติไมเซอร์ประมาณค่าไม่ตรงกับค่าจริง แต่ UDI-Agent ไม่พบการเปลี่ยนแปลงของข้อมูลหรือมีเปลี่ยนแปลงแต่ไม่เกิน 10%

3. **Pressing** เป็นตารางที่มีความจำเป็นในการเก็บสถิติมากเป็นลำดับถัดมา เป็นตารางที่ UDI-Agent พบว่ามีเปอร์เซ็นต์การเปลี่ยนแปลงของข้อมูลมากกว่าหรือเท่ากับ 50% ขึ้นไป แต่ QF-Agent ไม่พบความแตกต่างของค่าที่ออพติไมเซอร์ประมาณกับค่าจริง

4. **Urgent** เป็นตารางที่มีความจำเป็นต้องเก็บสถิติมากเป็นลำดับถัดมา โดยตารางที่อยู่ในระดับนี้เป็นตารางที่พบทั้งจากใน UDI-Agent และ QF-Agent เป็นตารางที่มีเปอร์เซ็นต์การเปลี่ยนแปลงของข้อมูลอย่างน้อย 10% ขึ้นไป และพบความแตกต่างระหว่างค่าที่ออพติไมเซอร์ประมาณกับค่าจริง

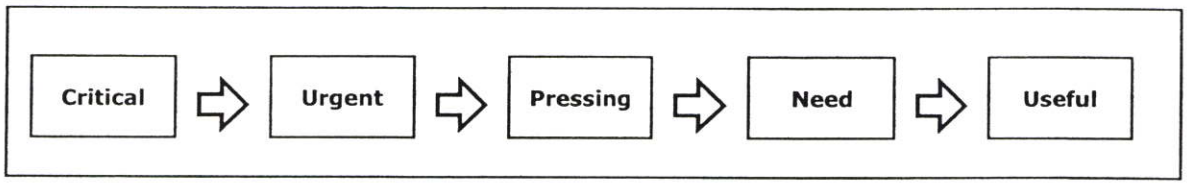
5. **Critical** เป็นตารางที่ได้จาก UDI-Agent หรือ QF-Agent หรือจากทั้ง 2 อย่างก็ได้ แต่เป็นตารางที่ไม่เคยมีการเก็บสถิติมาก่อน



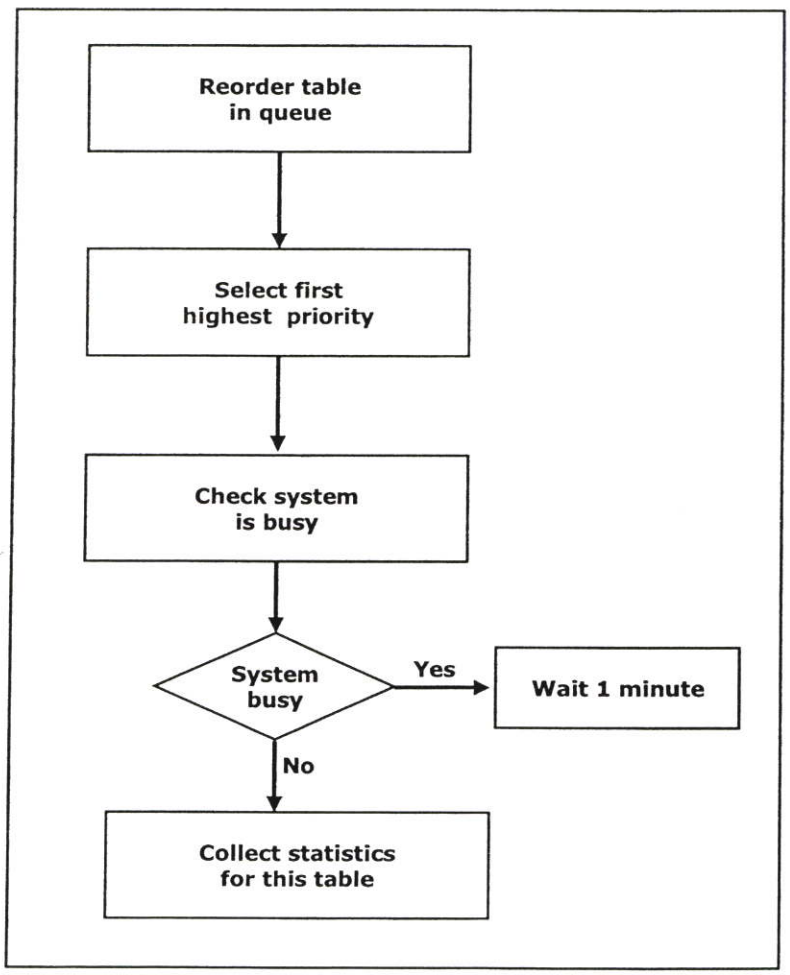
รูปที่ 3.5 ลำดับความจำเป็นในการเก็บสถิติ

3.4 การใช้โมบายเอเจนต์ในการเก็บสถิติ

หลังจากที่ Scheduler-Agent ได้จัดลำดับในการเก็บสถิติแล้ว เอเจนต์จะเริ่มเก็บสถิติโดยเริ่มจากตารางที่อยู่ในคิวระดับ Critical ก่อน แล้วไล่ไปจนคิวระดับ Useful



รูปที่ 3.6 ลำดับการเก็บสถิติ



รูปที่ 3.7 ขั้นตอนการทำงานของ Scheduler-Agent

ก่อนที่เอเจนต์จะเก็บสถิติของตารางตามลำดับในคิวนั้น Scheduler-Agent จะต้องตรวจสอบก่อนว่าในขณะนั้นระบบว่างหรือไม่ เนื่องจากงานในการเก็บสถิติเป็นงานที่ใช้ทรัพยากร

มาก หากเก็บสถิติในขณะที่มีผู้ใช้คนอื่นใช้งานฐานข้อมูลอยู่ด้วย ก็จะทำให้การเก็บสถิติไปแย่งใช้ทรัพยากรกับผู้ใช้คนอื่นทำให้ผู้ใช้คนอื่นต้องใช้เวลาในการทำงานมากขึ้นกว่าปกติ

วิธีในการตรวจสอบว่าระบบว่างหรือไม่จะใช้การนับจำนวนผู้ใช้งานอยู่ในขณะนั้น โดยใช้คำสั่ง

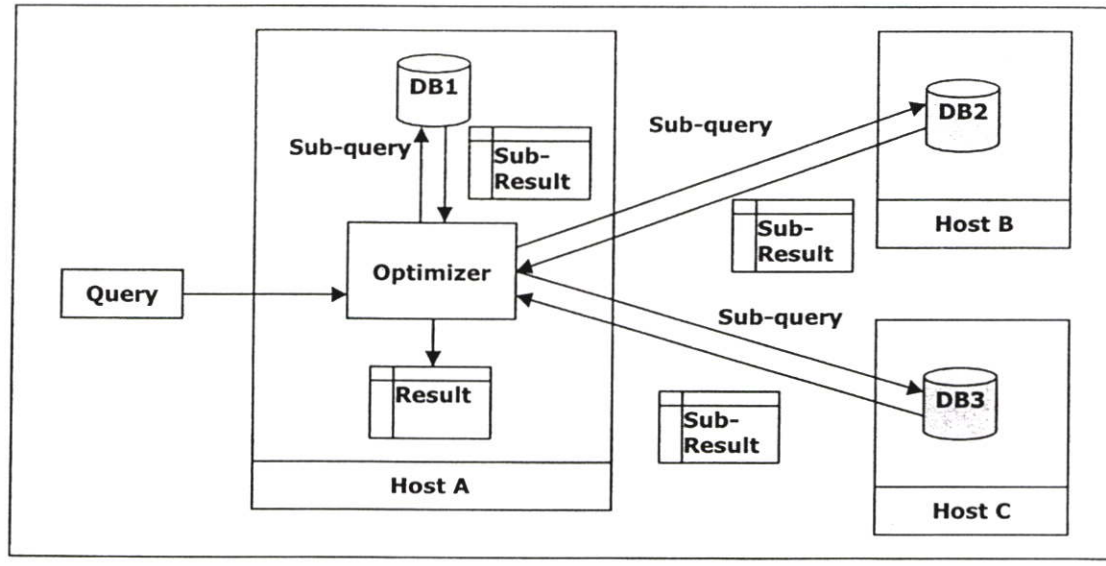
```
SELECT    COUNT(*)
FROM      V$SESSION
WHERE     USER# NOT IN (0,22,54) AND STATUS='ACTIVE
```

ซึ่งหากค่าที่ได้มีจำนวนมากกกว่าหนึ่ง แสดงว่าในขณะนั้นมีผู้ใช้คนอื่นใช้งานฐานข้อมูลอยู่ด้วย เอเจนต์จะหยุดการเก็บสถิติไว้ระยะเวลาหนึ่ง แล้วจึงเริ่มกลับมาเช็คจำนวนผู้ใช้งาน ก่อนที่จะเก็บสถิติทุกครั้ง และเมื่อทำการเก็บสถิติของแต่ละตารางเสร็จสิ้นลงก็จะปรับปรุงสถิติที่เครื่องเซิร์ฟเวอร์เก็บสถิติให้ข้อมูลสถิติตรงกับสถิติใหม่

บทที่ 4
โมบายเอเจนต์และวิธีการวางแผนคำถาม

4.1 การใช้โมบายเอเจนต์วางแผนคำถาม

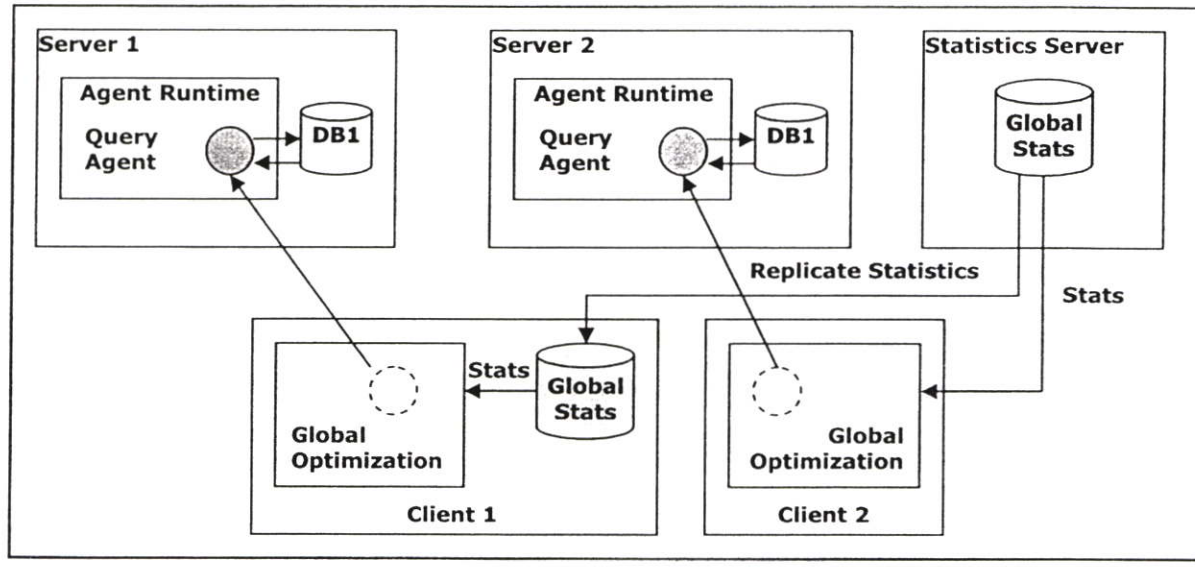
ถึงแม้ว่าสถิติของฐานข้อมูลแต่ละแห่งในระบบฐานข้อมูลแบบกระจายจะมีการปรับปรุงให้มีความสอดคล้องกับข้อมูลจริงๆ อยู่ตลอดเวลา จากรูปที่ 4.1 เป็นการวางแผนคำถามของออราเคิล โดยใช้วิธีแบ่งคำถามของผู้ใช้ออกเป็นคำถามย่อยๆ แล้วส่งคำถามย่อยไปยังฐานข้อมูลที่มีข้อมูลที่ต้องการ ฐานข้อมูลแต่ละแห่งที่ได้รับคำถามย่อยก็จะผลลัพธ์ของแต่ละคำถามย่อยกลับมาให้ เพื่อให้ฐานข้อมูลที่ได้รับคำถามทำการประมวลผลขั้นสุดท้ายก่อนส่งผลลัพธ์กลับไปให้ผู้ใช้ วิธีที่ออพติไมเซอร์วางแผนนี้ทำงานโดยมีเครื่องที่รับคำถามเป็นไคลเอนต์และเครื่องอื่นๆ เป็นเซิร์ฟเวอร์ โดยให้เซิร์ฟเวอร์ส่งข้อมูลที่ไคลเอนต์ต้องการมาให้ โดยไม่ได้มีการคำนึงถึงปริมาณของข้อมูลที่ต้องส่งผ่านระบบเครือข่าย เพื่อให้การวางแผนคำถามเป็นวิธีการที่สามารถทำได้จริง และมีประสิทธิภาพมากกว่าการวางแผนคำถามของออราเคิล



รูปที่ 4.1 วิธีการประมวลคำถามของออราเคิล

ในวิทยานิพนธ์นี้ขอเสนอการวางแผนคำถามแบบภาพรวม (Global Query Optimization) โดยใช้โมบายเอเจนต์ในการประมวลผลคำถามย่อย ซึ่งการประมวลผลของโมบายเอเจนต์จะทำให้เครื่องทุกเครื่องสามารถเป็นเซิร์ฟเวอร์เพื่อส่งข้อมูลหรือเป็นไคลเอนต์เพื่อรับข้อมูลจากฐานข้อมูลอื่นได้ทั้งสองแบบ ขึ้นอยู่กับวิธีที่จะใช้ในการประมวลผลคำถาม รูปที่ 4.2 เป็นภาพรวมการวางแผน

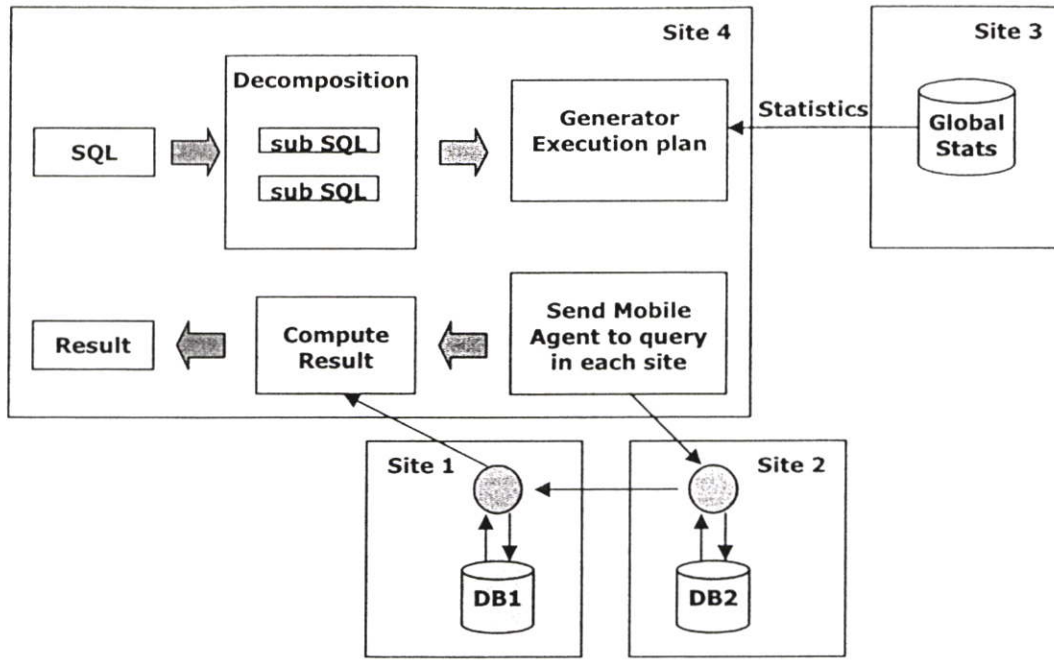
คำถามโดยใช้โมบายเอเจนต์ในการคิวรี่ย่อย ประกอบด้วยเครื่องไคลเอนต์ที่ติดตั้งโปรแกรม Global Optimization สำหรับการคิวรี่ ส่วนเครื่องที่มีฐานข้อมูลอยู่ต้องติดตั้งโปรแกรม Agent Runtime เพื่อให้โมบายเอเจนต์สามารถเคลื่อนที่ไปคิวรี่ข้อมูลที่เครื่องที่มีฐานข้อมูลได้



รูปที่ 4.2 ภาพรวมการวางแผนคำถามโดยใช้โมบายเอเจนต์ในการคิวรี่ย่อย

ในกรณีที่ใช้โมบายเอเจนต์ในการประมวลผลลัพธ์จากคำถามของผู้ใช้ ในส่วนของการวางแผนจะใช้วิธีการแบ่งคำถามออกเป็นคำถามย่อยก่อน แล้วนำข้อมูลสถิติมาใช้ในการคำนวณค่าใช้จ่าย เพื่อหาวิธีที่ใช้ค่าใช้จ่ายในการประมวลผลน้อยที่สุด จากนั้นจะส่งโมบายเอเจนต์ไปยังฐานข้อมูลต่างๆ เพื่อประมวลผลคำถามย่อยตามแผนคำถามที่ถูกเลือก ซึ่งแต่ละฐานข้อมูลจะมีสิทธิเป็นผู้รับหรือผู้ส่งข้อมูลก็ได้ ขึ้นอยู่กับแผนคำถามว่าให้ฐานข้อมูลใดเป็นฐานข้อมูลที่ประมวลผล ดังนั้นในการประมวลผลคำถามสามารถไปประมวลผลคำถามที่ฐานข้อมูลอื่นทั้งหมด จนได้ผลลัพธ์แล้วจึงค่อยส่งผลลัพธ์กลับมาที่ฐานข้อมูลที่ได้รับคำถามก็ได้

ในการวางแผนคำถามโดยใช้โมบายเอเจนต์ในการคิวรี่ย่อย จะแบ่งขั้นตอนการทำงานออกเป็นขั้นตอนในการแบ่งคำถามออกเป็นคำถามย่อย การวางแผนคำถาม และการดำเนินการตามแผนคำถามที่ได้วางไว้ รูปที่ 4.3 แสดงขั้นตอนการวางแผนคำถามโดยใช้โมบายเอเจนต์ในการคิวรี่ย่อย



รูปที่ 4.3 ขั้นตอนการวางแผนคำถามและการประมวลผลคำถาม

4.2 การแบ่งคำถามออกเป็นคำถามย่อย

ในการวางแผนคำถามจะใช้โมบายเอเจนต์ในการคิวรี่ เนื่องจากเป็นระบบฐานข้อมูลแบบกระจาย เพื่อเป็นการลดจำนวนข้อมูลที่ต้องส่งผ่านระบบเครือข่ายในการประมวลผล ดังนั้นเราจึงแบ่งคำถามออกเป็นคำถามย่อย โดยแต่ละคำถามย่อยเป็นคำถามที่จะถูกส่งไปทำงานบนแต่ละฐานข้อมูล ในการแบ่งคำถามออกเป็นคำถามย่อย นอกจากจะช่วยลดจำนวนข้อมูลที่ต้องส่งผ่านเครือข่าย ยังทำให้การทำงานแบบการทำงานแบบขนาน (Parallel) ซึ่งจะช่วยให้สามารถหาคำตอบได้รวดเร็วมากขึ้น ในการแบ่งคำถามออกเป็นคำถามย่อยมีขั้นตอนในการแบ่งดังนี้

1. แยกส่วนคำถามออกเป็นส่วนๆ โดยใช้ข้อมูลสถิติในการแบ่งชื่อคอลัมน์ และเงื่อนไขของแต่ละฐานข้อมูล

ตัวอย่างคำถามจากผู้ใช้

```
SELECT    C_NAME, O_ORDERSTATUS, O_TOTALPRICEC
FROM      CUSTOMER, ORDERS
WHERE     C_CUSTKEY = O_CUSTKEY
          AND O_ORDERPRIORIY = '1-URGENT'
```

เมื่อนำข้อมูลจากสถิติมาใช้ จะทำให้สามารถแยกได้ว่าคอลัมน์ที่ต้องการมาจากตารางใดจากฐานข้อมูลใด ส่วนเงื่อนไขจะแบ่งเงื่อนไขออกเป็นเงื่อนไขของตาราง กับเงื่อนไขที่ใช้ในการ

join ตาราง ซึ่งเงื่อนไขที่เป็นการ join ตารางจะเกิดจากเงื่อนไขที่เกิดจากคอลัมน์จาก 2 ตาราง ในขั้นตอนนี้จะทำให้แบ่งคำถามออกเป็นส่วนๆ ดังนี้

ฐานข้อมูล 1

ชื่อคอลัมน์ C_NAME, C_CUSTKEY

ชื่อตาราง CUSTOMER

เงื่อนไข ไม่มี

ฐานข้อมูล 2

ชื่อคอลัมน์ O_ORDERSTATUS, O_TOTALPRICE, O_CUSTKEY

ชื่อตาราง ORDERS

เงื่อนไข O_ORDERPRIORITY = '1-URGENT'

และเงื่อนไขที่เป็นการ join ตาราง

C_CUSTKEY = O_CUSTKEY,

2. นำข้อมูลที่แยกออกเป็นส่วนย่อยทั้งหมดมาประกอบเป็นคำถามย่อย ได้เป็น

คำถามย่อยที่ฐานข้อมูล 1 ได้เป็น

SELECT C_NAME, C_CUSTKEY

FROM CUSTOMER

คำถามย่อยที่ฐานข้อมูล 2 ได้เป็น

SELECT O_ORDERSTATUS, O_TOTALPRICE, O_CUSTKEY

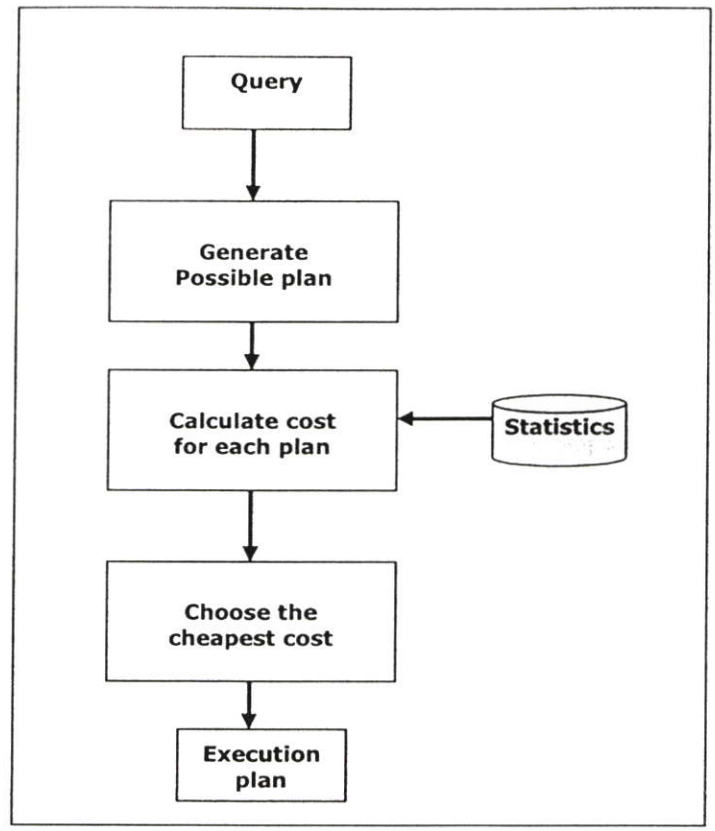
FROM ORDERS

WHERE O_ORDERSTATUS = '1-URGENT'

3. เมื่อได้คำถามย่อยทั้งหมดแล้วออพติไมเซอร์จะนำคำถามย่อยที่ได้ มาวางแผนคำถามเพื่อกำหนดลำดับในการประมวลผลคำถามย่อยต่อไป

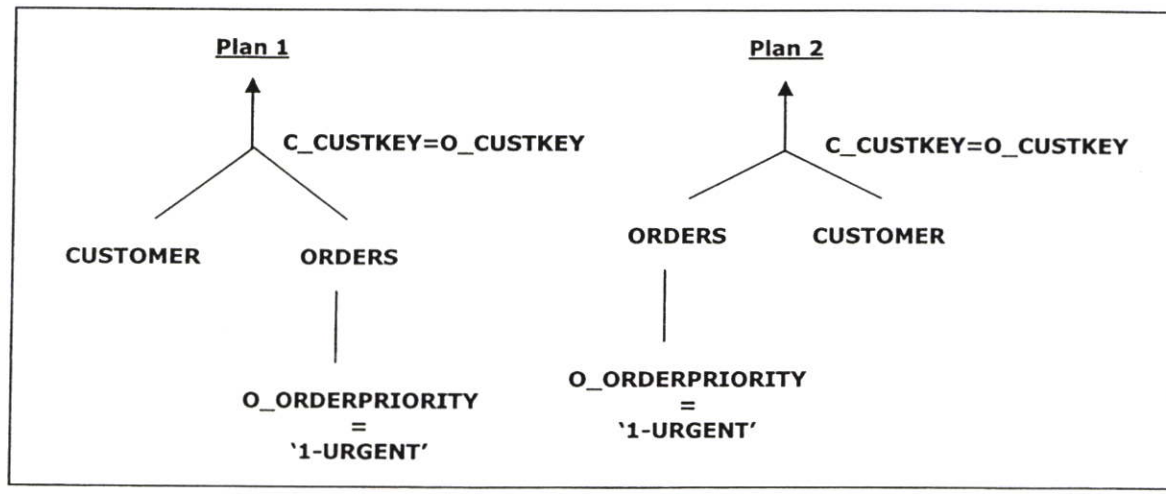
4.3 การวางแผนคำถาม

เป็นขั้นตอนที่คิดวิธีที่เป็นไปได้ทั้งหมดในการหาผลลัพธ์ เพื่อเลือกวิธีที่มีค่าใช้จ่ายในการประมวลผลน้อยที่สุดในการตอบคำถาม ในการหาวิธีที่เป็นไปได้ทั้งหมดจะเป็นการหาวิธีในการรวมข้อมูลที่มีความสัมพันธ์กันรูปแบบต่างๆ



รูปที่ 4.3 ขั้นตอนในการวางแผนคำถาม

ในการวางแผนคำถามจะค้นหาวิธีที่เป็นไปได้ทั้งหมดในการหาผลลัพธ์ โดยวิธีที่เป็นไปได้ทั้งหมดจะเกิดจากการเรียงชื่อข้อมูลทั้งหมดแล้วสร้างเป็นไบนารี่ทรี (Binary Tree) โดยที่โหนดสุดท้ายเป็นชื่อตารางที่มีการคัดกรองข้อมูลออกแล้ว แล้วเชื่อมทรีขึ้นทีละชั้น



รูปที่ 4.4 วิธีที่เป็นไปได้ในการหาผลลัพธ์

เมื่อหาวิธีที่เป็นไปได้ทั้งหมดแล้วจะคิดเป็นค่าใช้จ่ายในการส่งข้อมูลโดยค่าใช้จ่ายรวมในการส่งข้อมูลมีค่าเท่ากับผลรวมของการส่งข้อมูลในแต่ละครั้ง และในการส่งข้อมูลแต่ละครั้งมีค่าใช้จ่ายเท่ากับผลคูณของจำนวนข้อมูลกับขนาดของข้อมูล เมื่อคิดค่าใช้จ่ายของทุกวิธีที่เป็นไปได้ในการสืบค้นข้อมูลแล้วจึงเลือกวิธีที่ใช้จ่ายใช้จ่ายในการส่งข้อมูลต่ำที่สุดเพื่อใช้เป็นวิธีในการเข้าถึงข้อมูลจริงๆ

ในการคิดค่าใช้จ่ายจะพิจารณาจากค่าดังต่อไปนี้

Nr = เป็นจำนวนแถวทั้งหมดของตาราง r

Sr = เป็นขนาดของแต่ละแถวในตาราง r

V(A,r) = เป็นจำนวน distinct value ของคอลัมน์ A ในตาราง r

SC(A,r) = เป็นจำนวน cardinality ของค่าในคอลัมน์ A ในตาราง r

ตารางที่ 4.1 แสดงการประมาณค่าใช้จ่ายของแต่ละตาราง

Attribute	Nr	Sr	V(A,r)	SC(A,r)
r=Customer, A=C_CUSTKEY	1,500	22	1,500	1,500
r=Orders, A=O_ORDERPRIORITY	1,900	15	5	400

ตารางที่ 4.2 แสดงค่าใช้จ่ายของแต่ละตาราง

Relation	Cardinality	Size/tuple	Size
Customer	1,500	22	33,000
Orders	400	22+15 = 35	14,000

เมื่อได้ค่าใช้จ่ายรวมทั้งหมดสำหรับแต่ละแผนแล้วออพติไมเซอร์จะดูว่าแผนใดมีค่าใช้จ่ายต่ำที่สุด แล้วจะเลือก แผนนั้นเป็นวิธีในการประมวลผลคำถามต่อไป

4.4 การประมวลผลคำถามตามแผนคำถาม

เมื่อได้แผนในการดำเนินการแล้ว จะสร้างโมบายเอเจนต์ขึ้นตามจำนวนคำถามย่อย โดยเอเจนต์แต่ละตัวจะได้รับแผนในการประมวลผลคำถาม และคิวรี่ย่อยเพื่อไปคิวรี่ข้อมูลที่ฐานข้อมูลโดยตรงไปด้วย Global Query Optimization เมื่อโมบายเอเจนต์ไปถึงเครื่องเป้าหมายก็จะคิวรี่ข้อมูลจากฐานข้อมูล หากโมบายเอเจนต์ตัวใดมีเครื่องเป้าหมายที่ต้องไปคิวรี่คำถามย่อยอีก ก็จะเคลื่อนที่ไปยังเครื่องเป้าหมายต่อไป และไปดึงข้อมูลจากฐานข้อมูลนั้นๆ จนกว่าจะหมดขั้นตอนประมวลผลที่กำหนดให้กับโมบายเอเจนต์ ในแต่ละครั้งที่โมบายเอเจนต์เคลื่อนที่ไปคิวรี่คำถามย่อย เอเจนต์จะ

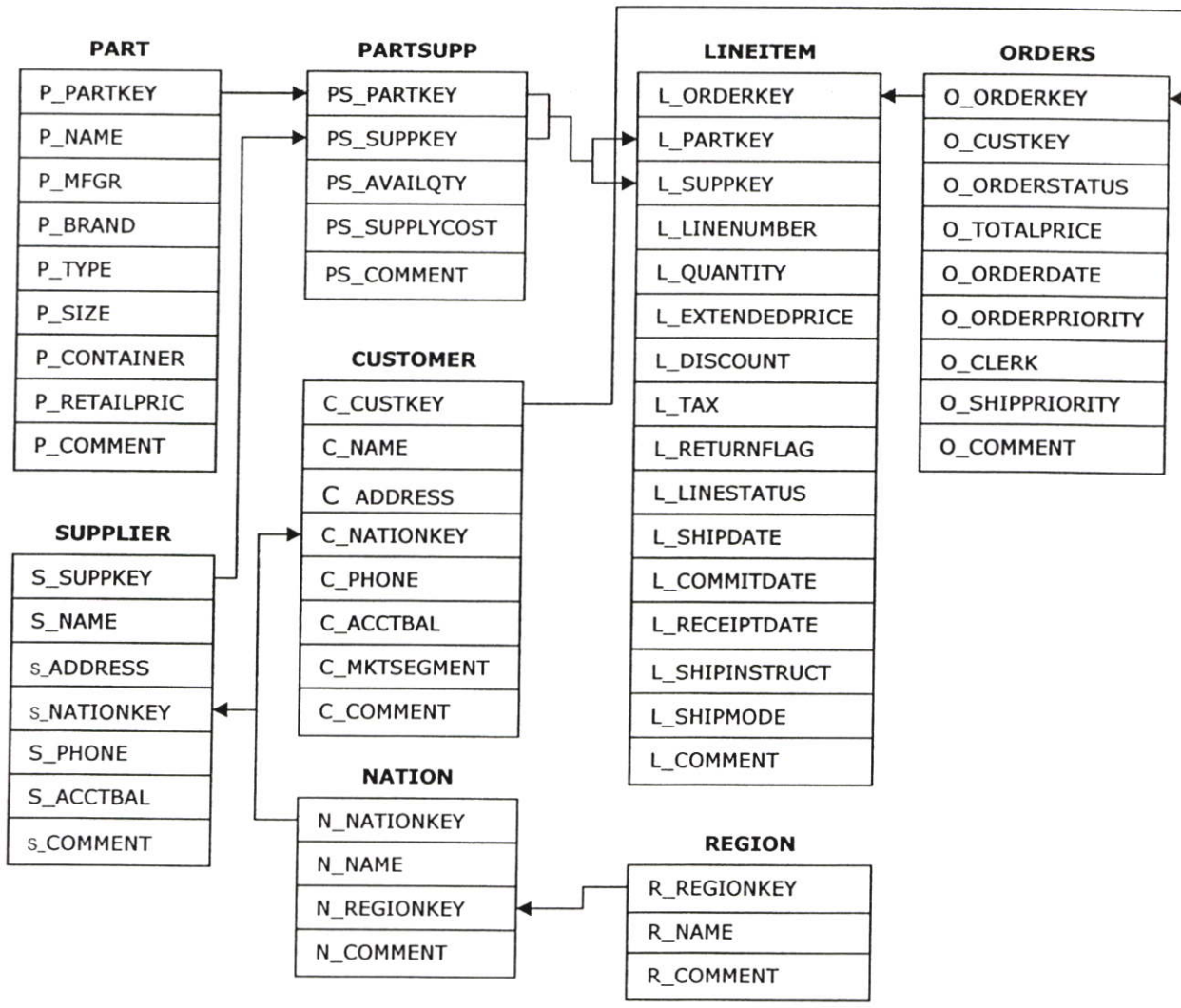
ตรวจสอบก่อนว่าที่ฐานข้อมูลนั้นมีโมบายเอเจนต์อื่นอยู่หรือไม่ หากพอว่ามีโมบายเอเจนต์ใดอยู่ก็จะนำข้อมูลจากโมบายเอเจนต์นั้นมาประมวลผลรวมกับข้อมูลที่ตัวเองมีอยู่

เมื่อโมบายเอเจนต์ใดๆ ทำหน้าที่ในการคิวรี่ข้อมูลเสร็จสิ้นแล้วก็จะจบการทำงานที่เครื่องใดๆ ที่โมบายเอเจนต์อยู่ จนเหลือเพียงโมบายเอเจนต์ตัวสุดท้ายที่มีข้อมูลผลลัพธ์จากการคิวรี่ที่จะเคลื่อนที่กลับมายังเครื่องตั้งต้น พร้อมกับส่งคำตอบให้กับผู้ใช้

บทที่ 5

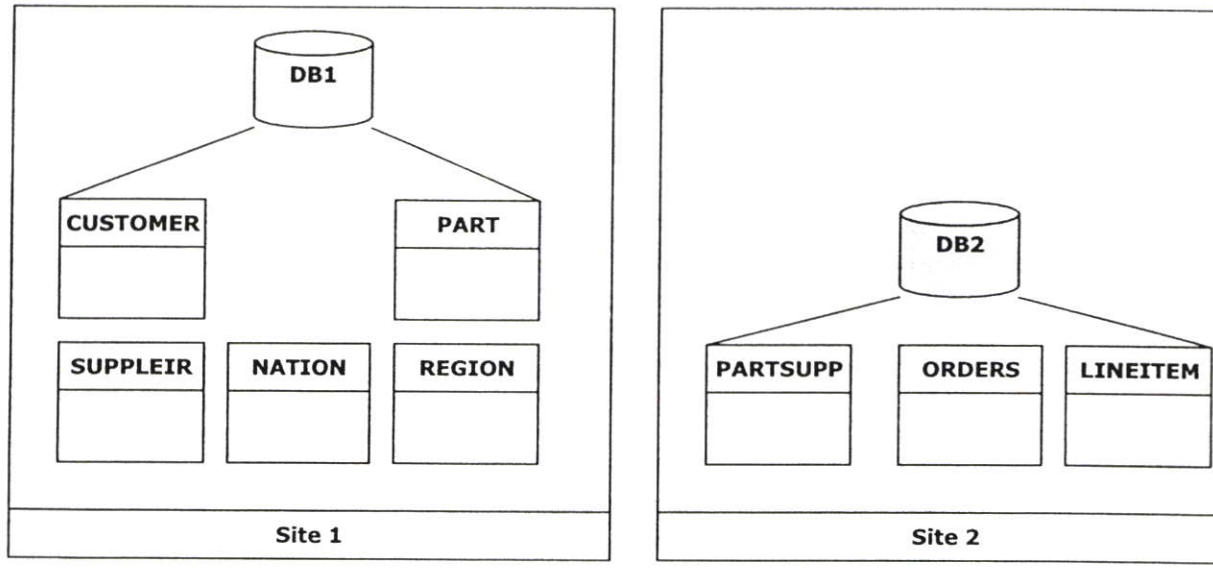
การดำเนินการทดลอง และผลการทดลอง

ในการวัดผลการดำเนินการทดลองจะแบ่งออกเป็น 2 ส่วน ส่วนแรกเป็นการวัดประสิทธิภาพของการสืบค้นข้อมูลในระบบฐานข้อมูลแบบกระจาย โดยมีโมบายเอเจนต์ทำหน้าที่เก็บสถิติในแต่ละฐานข้อมูล และส่วนที่สองเป็นการวัดประสิทธิภาพการวางแผนคำถามในระบบฐานข้อมูลแบบกระจายโดยใช้โมบายเอเจนต์ในการประมวลผลคำถาม สำหรับสภาพแวดล้อมของระบบประกอบด้วย เครื่องคอมพิวเตอร์ที่มีฐานข้อมูล oracle10g 2 ฐานข้อมูล มีการเชื่อมต่อระหว่างฐานข้อมูลทั้ง 2 ให้สามารถเรียกดูข้อมูลจากฐานข้อมูลทั้งสองได้ สำหรับข้อมูลในฐานข้อมูลประกอบด้วยความสัมพันธ์ดังรูปที่ 5.1



รูปที่ 5.1 ความสัมพันธ์ระหว่างตาราง

ในการทดลองได้จำลองระบบฐานข้อมูลแบบกระจายโดยการสร้างฐานข้อมูล 2 แห่ง โดยฐานข้อมูลแรกชื่อว่า DB1 ประกอบด้วยตาราง SUPPLIER, PART, CUSTOMER, NATION และ REGION ส่วนฐานข้อมูลที่สองมีชื่อว่า DB2 ประกอบด้วยตาราง PARTSUPP, ORDERS และ LINEITEM ดังรูปที่ 5.2 แต่ละฐานข้อมูลจะมีการกำหนดข้อมูลเริ่มต้นมากน้อยต่างกันไป



รูปที่ 5.2 การเก็บข้อมูลของแต่ละฐานข้อมูล

5.1 การเตรียมการทดลอง

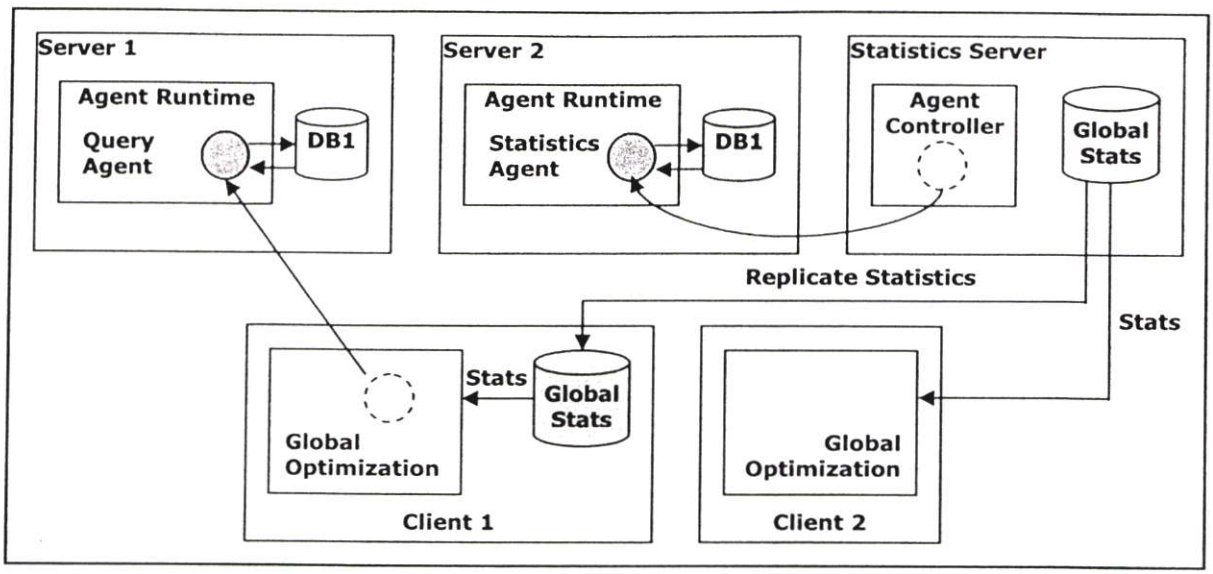
ในการทดลองประกอบด้วย

1 คอมพิวเตอร์สำหรับติดตั้งฐานข้อมูล 2 เครื่อง
   - ติดตั้งชุดไลบรารี่ของ Voyager
   - ติดตั้งโปรแกรม Agent Runtime

2 เครื่องคอมพิวเตอร์ที่ทำหน้าที่เป็นเซิร์ฟเวอร์เก็บสถิติ 1 เครื่อง
   - ติดตั้งชุดไลบรารี่ของ Voyager
   - ติดตั้งโปรแกรม Agent Controller

3 เครื่องไคลเอนต์สำหรับใช้ในการคิวรี่ข้อมูล 2 เครื่อง
   - ติดตั้งชุดไลบรารี่ของ Voyager
   - ติดตั้งโปรแกรม Global Optimization

4 กำหนดค่าของสภาวะแวดล้อมต่างๆ เพื่อให้สามารถใช้โมบายเอเจนต์ทำงานได้ โดยวิธีการกำหนดค่าสามารถดูได้ในภาคผนวก ก.



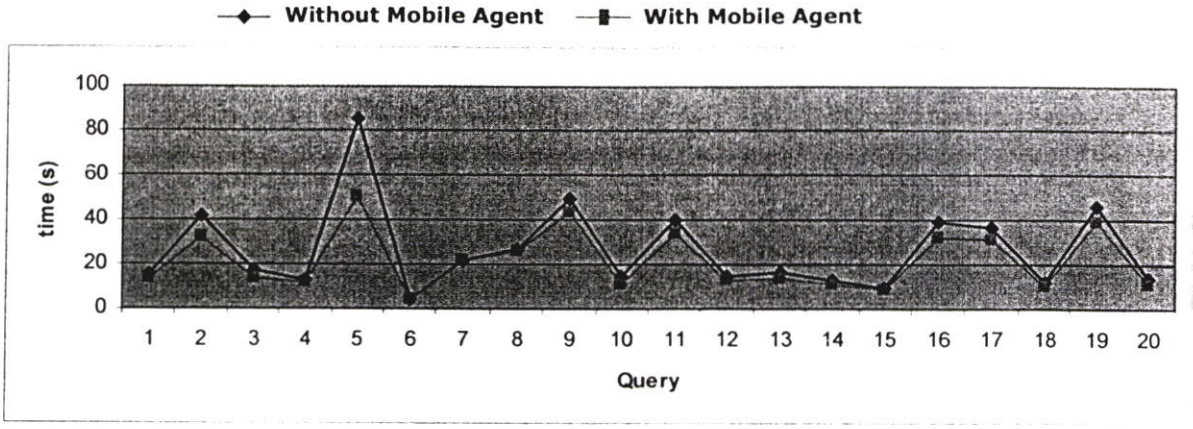
รูปที่ 5.3 ภาพรวมของการทดลอง

5.2 การเก็บสถิติโดยใช้โมบายเอเจนต์

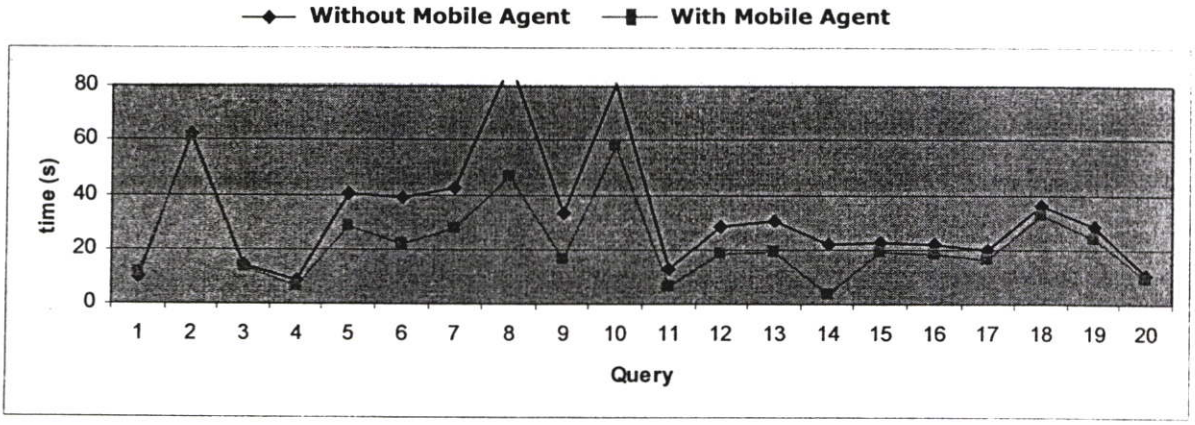
ในการใช้โมบายเอเจนต์ในการเก็บสถิต กำหนดให้เครื่องที่เป็นเอเจนต์คอนโทรลเลอร์คือเครื่องที่เป็นเซิร์ฟเวอร์เก็บสถิติ ซึ่งผู้ใช้จะต้องสร้างโมบายเอเจนต์จากเครื่องเอเจนต์คอนโทรลเลอร์แล้วส่งโมบายเอเจนต์ไปยังเครื่องต่างๆ ที่มีฐานข้อมูล เพื่อให้โมบายเอเจนต์เก็บสถิติของฐานข้อมูลนั้นๆ และเมื่อสถิติมีการเปลี่ยนแปลงโมบายเอเจนต์จะต้องนำข้อมูลสถิติมาปรับปรุงที่เครื่องเซิร์ฟเวอร์เก็บสถิติด้วย

วัตถุประสงค์ของการทดลองนี้ เพื่อเปรียบเทียบประสิทธิภาพการวางแผนคำถามของออพติไมเซอร์ในฐานข้อมูลที่ใช้โมบายเอเจนต์เก็บสถิติ กับฐานข้อมูลที่ DBMS เก็บสถิติเอง ในเบื้องต้นจะกำหนดให้มีจำนวนข้อมูลเริ่มต้นเหมือนกัน ต่อมาทำการเพิ่ม ลบ และเปลี่ยนแปลงข้อมูลกับทุกๆ ตารางมากน้อยแตกต่างกันไป หลังจากที่ได้เปลี่ยนแปลงข้อมูลในฐานข้อมูลแล้ว ก็จะทำการคิวรี่ข้อมูลด้วยคำถามต่างๆ รายละเอียดของแต่ละคำถามอยู่ในภาคผนวก ข. โดยรูปที่ 5.4 และ 5.5 จะแสดงจำนวนเวลาที่ใช้ในการประมวลผลคำถาม ตั้งแต่เริ่มต้นส่งคำถามไปจนถึงเวลาที่ได้รับผลลัพธ์กลับมา

สำหรับฐานข้อมูลที่ DBMS ทำหน้าที่ในการเก็บสถิติเองนั้น กำหนดให้งานในการเก็บสถิติทำเองอัตโนมัติเมื่อถึงช่วงเวลาที่กำหนดให้เป็นช่วงเวลาในการเก็บสถิติ ซึ่งกำหนดไว้เป็นช่วงเวลา 02.00-04.00 ส่วนฐานข้อมูลที่ใช้โมบายเอเจนต์เก็บสถิตินั้น จะส่งโมบายเอเจนต์ไปยังทุกๆ ไซต์ที่มีฐานข้อมูลอยู่ เพื่อให้โมบายเอเจนต์คอยเฝ้าดูข้อมูลในฐานข้อมูล และเก็บสถิติในช่วงเวลาที่เอเจนต์เห็นว่าเหมาะสม



รูปที่ 5.4 ผลการทดลองเปรียบเทียบจำนวนเวลาที่ใช้ในประมวลผลคำถามบนฐานข้อมูลออราเคิล ที่ใช้และไม่ใช้โมบายเอเจนต์ในการเก็บสถิติเปรียบเทียบกันในช่วงเวลาทำงาน

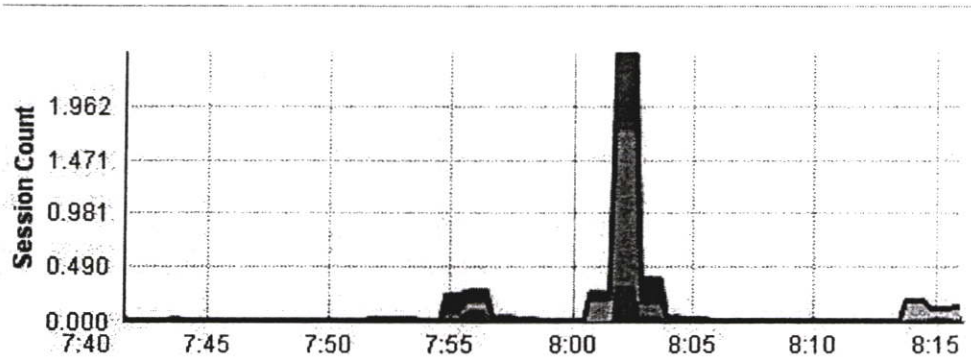


รูปที่ 5.5 ผลการทดลองเปรียบเทียบจำนวนเวลาที่ใช้ในประมวลผลคำถามบนฐานข้อมูลออราเคิล ที่ใช้และไม่ใช้โมบายเอเจนต์ในการเก็บสถิติเปรียบเทียบกันในช่วงนอกเวลาทำงาน

ในรูปที่ 5.4 เป็นการวัดผลโดยวัดในช่วงเวลา 08.00-10.00 ซึ่งเป็นช่วงที่มีการเพิ่ม ลบ หรือแก้ไขข้อมูลในฐานข้อมูลอยู่ตลอดเวลา จากกราฟจะแสดงให้เห็นว่าเวลาที่ใช้ในการคิวรี่บนฐานข้อมูลที่ไม่ได้ใช้โมบายเอเจนต์ จะใช้เวลามากกว่าการคิวรี่บนฐานข้อมูลที่ใช้โมบายเอเจนต์ในการเก็บสถิติ เนื่องจากฐานข้อมูลออราเคิลจะเก็บสถิติเองอัตโนมัติได้เฉพาะในช่วงเวลาที่ DBA กำหนดให้เก็บสถิติได้เท่านั้น ซึ่งโดยปกติแล้ว DBA ก็จะกำหนดให้ช่วงเวลาดังกล่าวเป็นเวลาที่นอกเวลาทำงาน เนื่องจากไม่ต้องการให้การเก็บสถิติส่งผลกระทบต่อผู้ใช้อื่นที่ใช้งานระบบอยู่เป็นจำนวนมากในช่วงเวลาทำงาน จึงทำให้ข้อมูลสถิติที่ใช้ในระหว่างวันนั้นเป็นข้อมูลสถิติที่เก่าเกินไป จึงทำให้ออพติไมเซอร์คำนวณวิธีในการเข้าถึงข้อมูลผิดพลาด ทำให้ไม่ได้วิธีที่ดีที่สุดในการ

ประมวลผล แต่สำหรับฐานข้อมูลที่ใช้โมบายเอเจนต์ในการเก็บสถิตินั้นจะมีโมบายเอเจนต์คอยปรับปรุงสถิติอยู่ตลอดเวลา โดยจะเลือกเวลาที่ไม่กระทบต่อผู้ใช้คนอื่นๆ ด้วย ดังนั้นเมื่อมีคำถามเข้ามาทำให้ออพติไมเซอร์มีข้อมูลสถิติที่สอดคล้องกับข้อมูลจริงมาใช้ในการเลือก จึงทำให้ได้วิธีที่ดีที่สุดในการประมวลผลจริงๆ

ในกรณีที่เป็นช่วงนอกเวลาทำงาน ที่ DBA กำหนดให้เป็นช่วงเวลาที่ DBMS เก็บสถิติ หากในช่วงเวลาที่กำลังเก็บสถิติอยู่นั้นมีผู้ใช้ส่งคำถามเข้ามา DBMS ก็จะยังคงเก็บสถิติตามกำหนดการเดิม โดยไม่ได้สนใจกับสิ่งอื่นรอบๆ ตัว จากรูปที่ 5.5 จะเห็นว่าเวลาที่ใช้ในการประมวลผลคำถามต้องใช้เวลาในการประมวลผลนานมากขึ้นเพราะ ในการเก็บสถิติเป็นการทำงานที่ใช้ทรัพยากรเป็นจำนวนมาก ทำให้ต้องแย่งกันใช้ทรัพยากรที่มีอยู่อย่างจำกัด แต่หากเป็นฐานข้อมูลที่ใช้โมบายเอเจนต์เก็บสถิตินั้น เนื่องจากในช่วงเวลาที่ได้มีการเปลี่ยนแปลงข้อมูลนั้น โมบายเอเจนต์ได้ทำการเก็บสถิติไปแล้ว ดังนั้นในช่วงนี้โมบายเอจนต์จะทำหน้าที่เพียงแค่เฝ้าดูข้อมูลเท่านั้น ไม่ได้มีการเก็บสถิติ แต่ถึงหากมีการเก็บสถิติเนื่องจากโมบายเอเจนต์เป็นตัวที่ทำหน้าที่ในการสั่งการเก็บสถิติ ทุกๆ ครั้งก่อนที่จะทำการเก็บสถิตินั้นเอเจนต์จะตรวจสอบดูก่อนว่าในขณะนั้นระบบว่างหรือไม่ หากเอเจนต์พบว่ามีคำถามถูกส่งมากำลังประมวลผลอยู่นั้น เอเจนต์จะยังไม่เก็บสถิติในทันที แต่จะรอจนกว่าระบบจะว่างจึงเก็บสถิติ จึงทำให้คำถามที่ส่งมาไม่ต้องได้รับผลกระทบจากการเก็บสถิติ



รูปที่ 5.6 กราฟแสดงปริมาณทรัพยากรที่ถูกใช้ในช่วงเวลาต่างๆ

ช่วงเวลาที่โมบายเอเจนต์วิเคราะห์สถิตินั้นจะใช้ทรัพยากรของเครื่องน้อยมาก จากรูปที่ 5.6 เป็นกราฟแสดงปริมาณทรัพยากรที่ถูกใช้ในช่วงเวลาต่างๆ โดยในช่วงเวลา 7.55-7.77 เป็นช่วงเวลาที่มีคำถามส่งเข้ามาและทำการประมวลผล ส่วนช่วงเวลา 8.00-8.05 เป็นช่วงเวลาที่โมบายเอเจนต์ทำการเก็บสถิติ โดยที่ช่วงเวลาอื่นๆ นั้นโมบายเอเจนต์ที่ทำการวิเคราะห์สถิติอยู่นั้น แทบไม่ได้มีการใช้ทรัพยากรของเครื่องเลย

5.3 การวางแผนคำถามโดยใช้โมบายเอเจนต์ในการคิวรี่ข้อมูล

การทดลองนี้เป็นการทดลองที่วัดประสิทธิภาพของการวางแผนคำถาม โดยแบ่งออกเป็น 3 การทดลองย่อยดังนี้

5.3.1 การวางแผนคำถามที่ข้อมูลที่ต้องการอยู่ฐานข้อมูลอื่นทั้งหมด

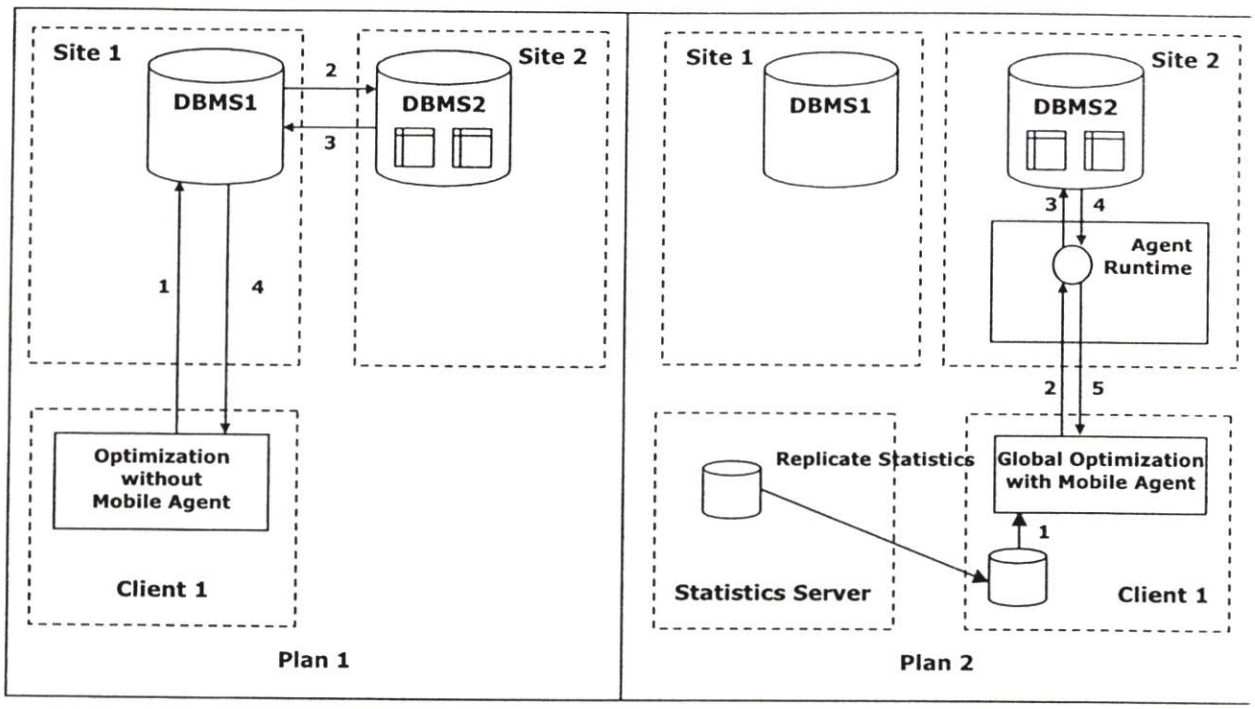
ในการทดลองนี้เป็นประมวลผลคำถามที่ข้อมูลทั้งหมดอยู่ที่ฐานข้อมูลอื่น โดยคำถามที่ใช้ในการทดสอบคือ

```
SELECT
        L_SHIPMODE, O_ORDERPRIORITY
FROM
        ORDERS@DB2, LINEITEM@DB2
WHERE
        O_ORDERKEY = L_ORDERKEY
        AND L_COMMITDATE < L_RECEIPTDATE
        AND L_SHIPDATE < L_COMMITDATE
        AND RECEIPTDATE >= DATE '1993-01-01'
        AND L_RECEIPTDATE < DATE '1993-01-01' + INTERVAL '1' YEAR
```

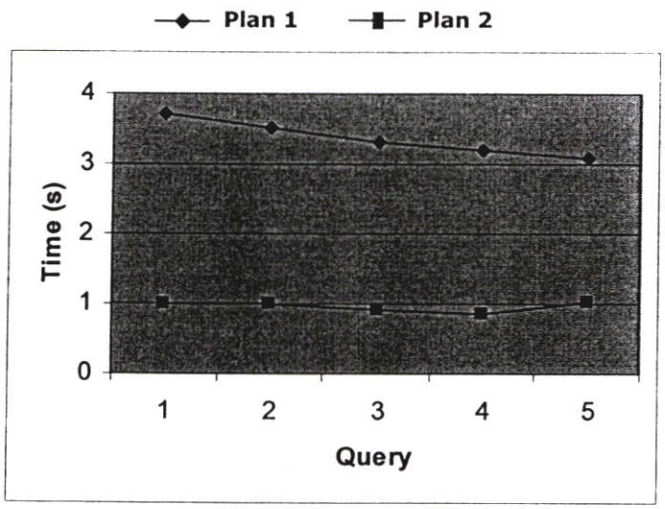
จากรูปที่ 5.7 เป็นการแสดงขั้นตอนในการคิวรี่ ในแบบแรกเมื่อ DBMS1 ได้รับคิวรี่จากเครื่องไคลเอนต์แล้ว DBMS1 จะทำหน้าที่ในการวางแผนคำถามเอง เมื่อได้แผนแล้วก็จะเริ่มดำเนินการโดยแยกคิวรี่ออกเป็นคิวรี่ย่อยสำหรับส่งไปที่ฐานข้อมูลที่ไซต์ 2 โดยจะส่งคิวรี่ย่อยไปให้กับ DBMS2 เพื่อให้ DBMS2 ส่งข้อมูลที่ต้องการกลับมาให้แล้วนำมาประมวลผลในขั้นตอนสุดท้ายก่อนเป็นผลลัพธ์ แต่ในการทดลองนี้เนื่องจากข้อมูลทั้งหมดอยู่ที่ไซต์ 2 ดังนั้นเมื่อได้ข้อมูลกลับมาจากไซต์ 2 แล้ว DBMS1 ก็นำผลลัพธ์ตอบกลับไปที่เครื่องไคลเอนต์ 1

ส่วนแบบที่สองเป็นการประมวลผลโดยใช้วิธี Global Query Optimization โดยการทำ Global Optimization จะทำที่เครื่องของไคลเอนต์ ดังนั้นเมื่อได้แผนในการประมวลผลแล้ว Global Optimization จะสร้างโมบายเอเจนต์ขึ้นมาแล้วส่งไปยังไซต์ 2 เพื่อนำคิวรี่ไปประมวลผลที่ไซต์ 2 และเมื่อได้ผลลัพธ์ก็จะนำผลลัพธ์ตอบกลับมาที่เครื่องไคลเอนต์ 1 ได้เลย

เมื่อเปรียบเทียบขั้นตอนการทำงานของทั้ง 2 วิธี จะเห็นว่าการใช้โมบายเอเจนต์ช่วยตัดสินใจให้เลือกติดต่อกับฐานข้อมูลที่มีข้อมูลอยู่จริงๆ เท่านั้น ไม่เหมือนกับการทำงานของ DBMS ซึ่งจะขึ้นอยู่กับว่าเครื่องไคลเอนต์เชื่อมต่ออยู่กับฐานข้อมูลใดก็ต้องติดต่อกับฐานข้อมูลนั้นก่อน แล้วให้ฐานข้อมูลนั้นทำหน้าที่ในการติดต่อไปยังฐานข้อมูลที่มีข้อมูลอยู่จริง



รูปที่ 5.7 ขั้นตอนในการคิวรี่ระหว่าง DBMS กับ Global Optimization with Mobile Agent



รูปที่ 5.8 แสดงจำนวนเวลาที่ใช้ในการประมวลผลระหว่าง DBMS กับ Global Optimization with Mobile Agent

รูปที่ 5.8 เป็นกราฟเปรียบเทียบเวลาที่ใช้ในการคิวรี่ซึ่งแบบแรกเป็นการคิวรี่โดย DBMS ทำหน้าที่วางแผนคำถามและประมวลผลเอง ส่วนแบบที่สองเป็นการวางแผนคำถามโดย Global Query Optimization แล้วใช้โมบายเอเจนต์ในการประมวลผล ซึ่งผลที่ได้คือ Global Query Optimization ที่ใช้โมบายเอเจนต์ในการประมวลผลใช้เวลาในการประมวลผลจริงน้อยกว่ามาก

5.3.2 การวางแผนคำถามที่ใช้ข้อมูล 2 ตารางที่อยู่ต่างฐานข้อมูลในการ join

คำถามที่ใช้ในการทดลองคือ

```
SELECT
        C_CUSTKEY, C_NAME, L_EXTENDEDPRICE, L_DISCOUNT
        C_ACCTBAL, N_NAME, C_ADDRESS, C_PHONE, C_COMMENT
FROM
        CUSTOMER@DB1, ORDERS@DB2, LINEITEM@DB2
WHERE
        C_CUSTKEY = O_CUSTKEY
        AND L_ORDERKEY = O_ORDERKEY
        AND O_ORDER >= DATE '1994-09-01'
        AND O_ORDERDATE < '1994-09-01' + INTERVAL '3' MONTH
        AND L_RETURNFLAG = 'R'
        AND C_NATIONKEY = N_NATIONKEY
```

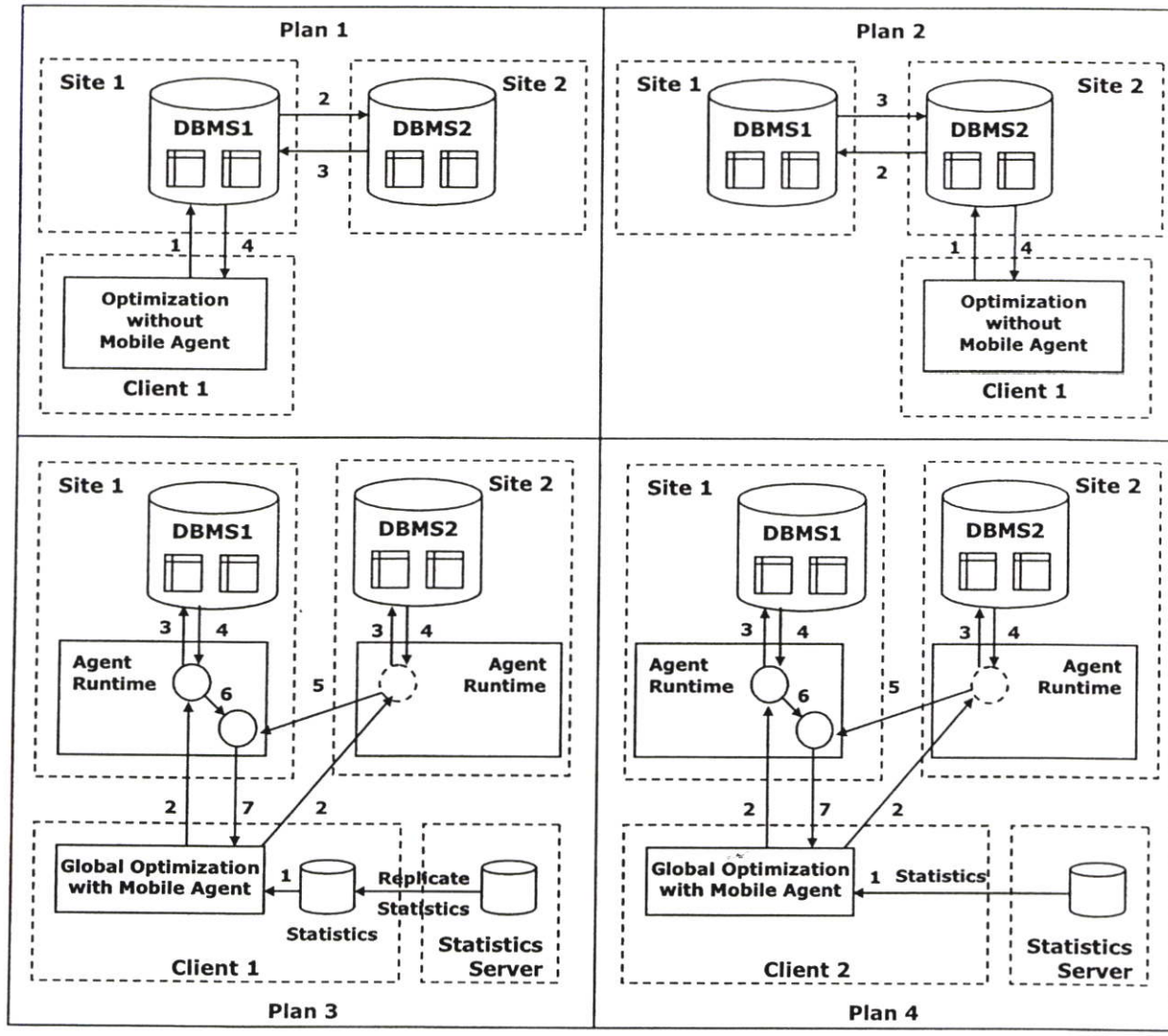
แบบที่ 1 เป็นการส่งคำถามจากเครื่องไคลเอนต์ 1 ซึ่งเชื่อมต่ออยู่กับฐานข้อมูลที่ไซต์ 1 เมื่อ DBMS1 ได้รับคำถามก็จะแบ่งออกเป็นคำถามย่อย โดยแบ่งออกเป็นคำถามที่ดึงข้อมูลจากตาราง Customer, Nation และคำถามที่ดึงข้อมูลจากตาราง Orders, Lineitem คำถามย่อยแรกจะประมวลผลที่ DBMS1 ส่วนคำถามย่อยที่ 2 จะส่งไปยัง DBMS2 เพื่อให้ DBMS2 ประมวลผล เมื่อได้ผลลัพธ์แล้ว DBMS2 จะส่งผลลัพธ์กลับมาให้กับ DBMS1 เพื่อให้ DBMS1 นำผลลัพธ์มาประมวลผลรวมกับผลลัพธ์ที่ DBMS1 ได้ แล้วส่งกลับไปให้กับไคลเอนต์ 1

แบบที่ 2 เป็นการส่งคำถามจากเครื่องไคลเอนต์ 2 ซึ่งเชื่อมต่ออยู่กับฐานข้อมูลที่ไซต์ 2 โดยขั้นตอนการทำงานก็จะคล้ายกับแบบที่ 1 คือ DBMS2 จะแยกออกเป็นคำถามย่อย แล้วส่งคำถามย่อยที่ใช้ข้อมูลจากตาราง Customer, Nation ไปให้ DBMS1 ประมวลผล เมื่อได้ผลลัพธ์กลับมาก็นำผลลัพธ์ที่ได้ มาประมวลผลรวมกับผลลัพธ์จากตาราง Orders, Lineitem แล้วจึงส่งคำตอบกลับไปยังเครื่องไคลเอนต์ 2

แบบที่ 3 เป็นการทำ Global Query Optimization แบบที่มีการทำสำเนาข้อมูลสถิติไว้ที่เครื่องไคลเอนต์ เมื่อได้รับคำถามมา Global Optimization จะใช้ข้อมูลสถิติที่มีคำนวณวิธีในการประมวลผลเพื่อเลือกวิธีที่มีประสิทธิภาพมากที่สุด จากนั้นก็จะสร้างโมบายเอเจนต์แล้วส่งไปยังไซต์ 1 เพื่อคิวรี่ข้อมูลจากตาราง Customer, Nation และส่งโมบายเอเจนต์ไปที่ไซต์ 2 เพื่อคิวรี่ข้อมูล

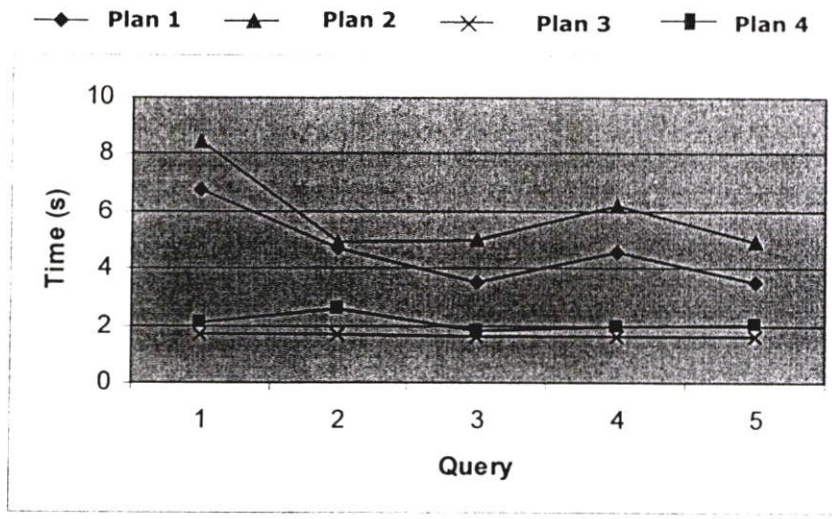
จากตาราง Orders, Lineitem จากนั้นให้โมบายเอเจนต์ที่ไซต์ 1 เคลื่อนที่ไปที่ไซต์ 2 เพื่อนำผลลัพธ์ที่ได้มาประมวลผลรวม ก่อนจะได้เป็นคำตอบแล้วส่งกลับมาไคลเอนต์ 1

แบบที่ 4 เป็นการทำ Global Query Optimization เช่นเดียวกับแบบที่ 3 แต่ที่เครื่องไคลเอนต์ 2 ไม่มีการทำสำเนาข้อมูลสถิติไว้ ในขั้นตอนแรกจึงต้องขอข้อมูลสถิติจากเครื่องเซิร์ฟเวอร์ที่เก็บสถิติ เพื่อนำมาหาวิธีในการประมวลผล ซึ่งเมื่อได้ข้อมูลสถิติมาแล้วลำดับในการประมวลผลก็จะเหมือนกับวิธีการในแบบที่ 3



รูปที่ 5.9 ขั้นตอนการคิวรี่ด้วย DBMS กับ Global Optimization with Mobile Agent

รูปที่ 5.10 เป็นกราฟเปรียบเทียบเวลาที่ใช้ในการคิวรี่ซึ่งแบบแรกเป็นการคิวรี่โดย DBMS ทำหน้าที่วางแผนคำถามและประมวลผลเอง ส่วนแบบที่สองเป็นการวางแผนคำถามโดย Global Query Optimization แล้วใช้โมบายเอเจนต์ในการประมวลผล ซึ่งผลที่ได้คือ Global Query Optimization ที่ใช้โมบายเอเจนต์ในการประมวลผลใช้เวลาในการประมวลผลจริงน้อยกว่า

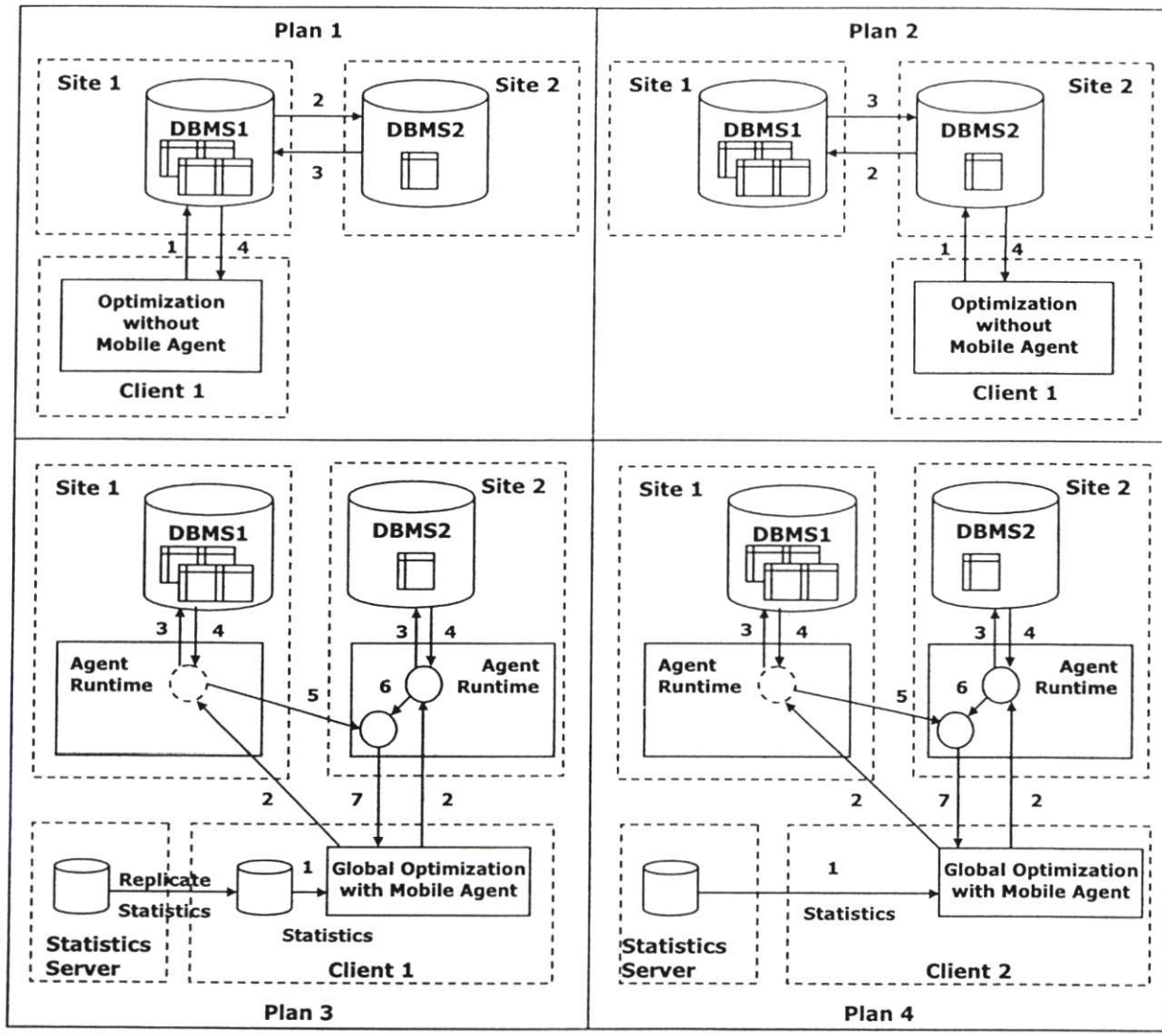


รูปที่ 5.10 แสดงจำนวนเวลาที่ใช้ในการประมวลผลระหว่าง DBMS กับ Global Optimization with Mobile Agent

5.3.3 การวางแผนคำถามที่ใช้ข้อมูล 3 ตารางที่อยู่ต่างฐานข้อมูลในการ join

คำถามที่ใช้ในการทดลองคือ

```
SELECT
        S_ACCTBAK, S_NAME, N_NAME, P_PARTKEY,
        P_MFGR, S_ADDRESS, S_COMMENT
FROM
        PART@DB1, SUPPLIER@DB1, PARTSUPP@DB2,
        NATION@DB1, REGION@DB1
WHERE
        P_PARTKEY = PS_PARTKEY
        AND S_SUPPKEY = PS_SUPPKEY
        AND P_TYPE LIKE '%STEEL'
        AND S_NATIONKEY = N_NATIONKEY
        AND N_REGIONKEY = R_REGIONKEY
```



รูปที่ 5.11 ขั้นตอนการคิวรี่ด้วย DBMS กับ Global Optimization with Mobile Agent

สำหรับแบบที่ 1 และ 2 เป็นการวางแผนคำถามด้วย DBMS โดยแบบที่ 1 เป็นการส่งคำถามจากเครื่องไคลเอนต์ 1 ที่เชื่อมต่อกับฐานข้อมูลที่ไซต์ 1 ส่วนแบบที่ 2 เป็นการส่งคำถามจากเครื่องไคลเอนต์ 2 ที่เชื่อมต่อกับฐานข้อมูลที่ไซต์ 2

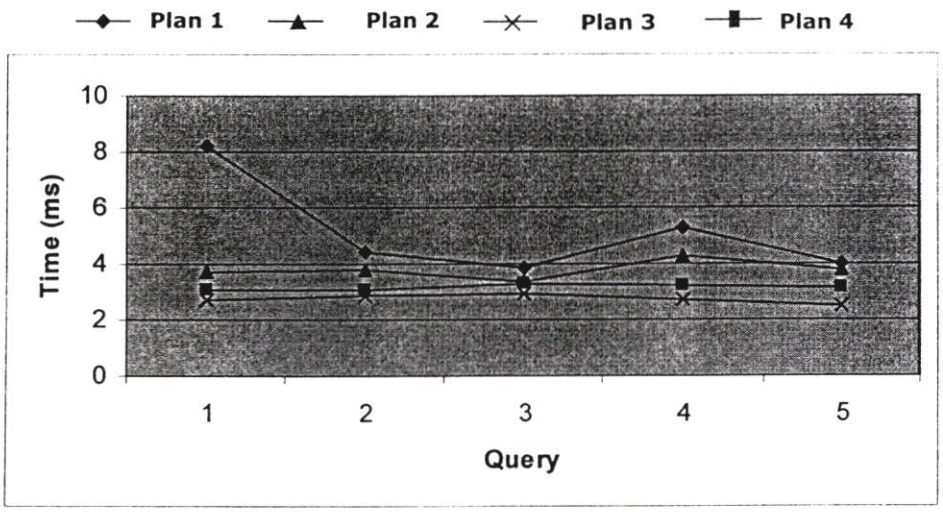
แบบที่ 3 และ 4 เป็นการวางแผนคำถามด้วย Global Query Optimization โดยแบบที่ 3 มีการทำสำเนาข้อมูลสถิติจากเครื่องเซิร์ฟเวอร์ข้อมูลสถิติมาไว้ที่เครื่องไคลเอนต์ ส่วนแบบที่ 4 จะขอข้อมูลสถิติจากเครื่องเซิร์ฟเวอร์ข้อมูลสถิติ

แบบที่ 1 เริ่มต้นจากการส่งคำถามไปที่ DBMS1 จากนั้น DBMS1 จะแบ่งคำถามออกเป็นคำถามย่อย แล้วส่งคำถามย่อยต่อไปให้กับ DBMS2 เมื่อ DBMS2 ส่งคำตอบกลับไปให้กับ DBMS1 แล้ว DBMS1 จะนำคำตอบที่ได้มาประมวลผลรวมกับคำตอบของตัวเอง ก่อนจะออกมาเป็นผลลัพธ์ส่งให้กลับไคลเอนต์ 1

แบบที่ 2 จะส่งคำถามไปที่ DBMS2 เมื่อ DBMS2 ได้รับคำถามมาก็จะแบ่งคำถามออกเป็นคำถามย่อยแล้วส่งต่อไปให้กับ DBMS1 เมื่อ DBMS1 ประมวลผลจนได้คำตอบแล้วก็จะส่งคำตอบกลับไปให้กับ DBMS2 เพื่อทำการประมวลผลออกมาเป็นคำตอบ

แบบที่ 3 Global Optimization จะใช้ข้อมูลสถิติที่ทำสำเนาไว้มาคิดแผนในการประมวลผล จากนั้นสร้างโมบายเอเจนต์แล้วส่งไปที่ไซต์ 1 และ 2 เพื่อคิวรี่ข้อมูล จากนั้นให้โมบายเอเจนต์ที่ไซต์ 1 เคลื่อนที่ไปที่ไซต์ 2 เพื่อนำผลลัพธ์มาประมวลผลรวมกับผลลัพธ์จากไซต์ 2 เมื่อได้คำตอบก็ส่งคำตอบกลับไปที่เครื่องไคลเอนต์

แบบที่ 4 จะต่างกับแบบที่ 3 เฉพาะในขั้นตอนของการใช้ข้อมูลสถิติ โดยแบบที่ 2 จะต้องรอข้อมูลสถิติจากเครื่องเซิร์ฟเวอร์เก็บสถิติเพื่อนำมาใช้ในการสร้างแผนในการประมวลผล



รูปที่ 5.12 แสดงจำนวนเวลาที่ใช้ในการประมวลผลระหว่าง DBMS กับ Global Optimization with Mobile Agent

รูปที่ 5.12 แสดงเวลาที่ใช้ในการประมวลผลคำถามในแบบต่างๆ โดยแบบที่ 1 และ 2 เป็นการประมวลผลโดย DBMS แบบที่ 3 และ 4 เป็นการประมวลผลคำถามด้วยวิธี Global Query Optimization โดยใช้โมบายเอเจนต์ในการประมวลผล ซึ่งผลลัพธ์ที่ได้คือวิธี Global Query Optimization ใช้เวลาในการประมวลผลน้อยกว่า

จากผลการทดลองทั้ง 3 แบบ จะเห็นว่าการใช้โมบายเอเจนต์ในการทำงานมีส่วนช่วยให้เวลาที่ใช้ในการทำงานลดลง ซึ่งตรงกับวัตถุประสงค์ของวิทยานิพนธ์ฉบับนี้

บทที่ 6
สรุปผลการวิจัย และข้อเสนอแนะ

จากความสามารถที่น่าสนใจของโมบายเอเจนต์ เราสามารถนำโมบายเอเจนต์มาประยุกต์ใช้ในงานต่างๆ ได้มากมาย ทั้งเพื่อเพิ่มประสิทธิภาพการทำงาน หรือแก้ปัญหาในการทำงานต่างๆ โดยงานวิจัยนี้ได้เลือกนำโมบายเอเจนต์มาประยุกต์ใช้กับเก็บสถิติ และการวางแผนคำถาม เนื่องจากได้เล็งเห็นความสำคัญของสถิติ และปัญหาที่พบในการเก็บสถิติ รวมถึงขีดจำกัดของการทำงานของ DBMS ในการวางแผนคำถามในระบบฐานข้อมูลแบบกระจาย จึงทดลองนำโมบายเอเจนต์มาใช้ในการทำงานเพื่อให้สามารถวางแผนคำถามให้มีความซับซ้อนมากขึ้นแต่มีประสิทธิภาพที่มากกว่า

วิทยานิพนธ์ฉบับนี้ได้นำเสนอวิธีที่ช่วยเพิ่มประสิทธิภาพการทำงานให้กับ DBMS โดยการนำโมบายเอเจนต์มาใช้เพื่อนำมาแก้ปัญหาที่พบในการเก็บสถิติ และนำมาเพิ่มประสิทธิภาพในการประมวลผลคำถาม

ในการเก็บสถิติโดยการให้ DBMS เก็บสถิติเองโดยอัตโนมัติ ยังมีข้อด้อยอยู่มากมาย ทั้งเรื่องที่สถิติไม่ได้รับการปรับปรุงทันทีในเวลาที่ควรปรับปรุง เพราะต้องรอให้ถึงเวลาที่กำหนดไว้ก่อนนั้น การใช้โมบายเอเจนต์ในการเก็บสถิติจะช่วยให้ปัญหาข้อนี้ถูกแก้ไป เนื่องจากเอเจนต์จะปรับปรุงข้อมูลสถิติให้สัมพันธ์กับการเปลี่ยนแปลงของข้อมูลในฐานข้อมูล โดยจะเริ่มปรับปรุงสถิติทันทีที่ระบบว่าง ทำให้ออพติไมเซอร์มีข้อมูลที่ถูกต้องใช้ในการวางแผนคำถาม ซึ่งทำให้ได้ plan ที่ดีที่สุดสำหรับแต่ละคำถาม ซึ่งการใช้โมบายเอเจนต์ในการเก็บสถิตินั้นจะดีกว่าการให้ DBMS ทำการเก็บสถิติเองเพราะเอเจนต์สามารถเก็บสถิติได้ตลอดเวลาตามความจำเป็นในการเก็บสถิติ แต่ DBMS ที่เก็บสถิติเองนั้นสามารถเก็บสถิติได้เฉพาะในช่วงเวลาที่ DBA กำหนดไว้เท่านั้น เมื่อถึงช่วงเวลาที่เก็บสถิติ DBMS ก็จะทำการเก็บสถิติของทุกตารางที่จำเป็นต้องเก็บสถิติต่อเนื่องกัน โดยไม่สนใจว่าจะมีผู้ใช้อื่นในระบบหรือไม่ แต่หากเป็นเอเจนต์จะเก็บสถิติทีละตาราง โดยก่อนที่จะเก็บสถิติในแต่ละตารางจะมีการตรวจสอบก่อนว่ามีผู้ใช้ในระบบหรือไม่ หากพบว่ามีผู้ใช้ก็จะพักการเก็บสถิติเป็นระยะเวลาหนึ่ง แล้วจึงกลับมาตรวจสอบใหม่ว่ามีผู้ใช้หรือไม่ เมื่อพบว่าไม่มีผู้ใช้จึงจะทำการเก็บสถิติตารางนั้นๆ จะเห็นว่าการใช้โมบายเอเจนต์นั้นช่วยให้งานในการเก็บสถิตินั้นเลือกเก็บสถิติเฉพาะตารางที่จำเป็นต้องเก็บสถิติ และเลือกเวลาในการเก็บสถิติที่เหมาะสมมากกว่าการเก็บสถิติโดย DBMS

ในส่วนของการวางแผนคำถาม สำหรับ DBMS Oracle นั้นยังไม่สามารถทำการวางแผนคำถามแบบ Global ได้ ทำได้แค่เพียงแบ่งคำถามออกเป็นคำถามย่อยแล้วส่งไปยัง DBMS อื่นเพื่อให้ส่งคำตอบที่ต้องการกลับมาประมวลผลที่เครื่องเท่านั้น ซึ่งการทำงานแบบนี้นอกจากขั้นตอนในการทำงานไม่ใช่วิธีที่ดีที่สุดแล้ว การรอข้อมูลจากฐานข้อมูลส่งมาเพื่อประมวลผลก็ใช้เวลานานด้วยหาก

ข้อมูลที่ได้มีปริมาณมาก แต่ในวิทยานิพนธ์ฉบับนี้แสดงวิธีการวางแผนคำถามแบบ Global โดยอาศัยข้อมูลจาก Statistics Server เพื่อนำมาคำนวณค่าใช้จ่ายก่อนที่จะเลือก plan ที่ดีที่สุดในการประมวลผล และเมื่อได้ plan แล้วก็จะส่งโมบายเอเจนต์ไปประมวลผลคำถามย่อยที่ฐานข้อมูลต่างๆ โดยในการประมวลผลที่ต้องรวมข้อมูลจากฐานข้อมูลหลายๆ แห่งจะให้ฐานข้อมูลที่มีปริมาณข้อมูลน้อยกว่าเป็นตัวส่งข้อมูลมา เพื่อลดเวลาที่ต้องรอในระหว่างส่งข้อมูล เมื่อประมวลผลเสร็จจึงค่อยนำผลลัพธ์กลับไปที่เครื่องที่รับคำถามมา ทำให้ช่วยลดปริมาณข้อมูลที่ต้องถูกส่งผ่านเครือข่าย ซึ่งช่วงเวลาที่ส่งข้อมูลนี้ใช้เวลาในการส่งข้อมูลเป็นเวลานาน

ผลลัพธ์จากการประมวลผลด้วยวิธี Global Optimization ในวิทยานิพนธ์ฉบับนี้ก็ได้แสดงให้เห็นว่ามีประสิทธิภาพมากกว่าการประมวลผลด้วย Optimizer ของ DBMS ที่เป็น Oracle ซึ่งถือได้ว่าเป็น DBMS อันดับต้นๆ อีกด้วย เนื่องจากวิธีที่ใช้ในการประมวลผลมีประสิทธิภาพมากกว่าประกอบกับการมีข้อมูลสถิติที่มีการปรับปรุงอยู่ตลอดเวลา ซึ่งช่วยให้ได้วิธีการประมวลผลที่มีประสิทธิภาพสูงสุด

ส่วนข้อเสนอแนะสำหรับการปรับปรุงวิธีในการประมวลผลนั้น เป็นส่วนของขั้นตอนในการ join ข้อมูล เนื่องจากในวิทยานิพนธ์ฉบับนี้ ในขั้นตอนการ join ข้อมูลนั้นจะใช้วิธีส่งข้อมูลทั้งแถวไปใช้ในการประมวลผลรวม ซึ่งหากปรับเปลี่ยนวิธีการประมวลผลรวมโดยให้ส่งเฉพาะ key ที่ใช้ในการ join ไป join ก่อนแล้วจึงค่อยส่งข้อมูลที่เหลือตามไป ก็อาจช่วยให้การทำงานมีประสิทธิภาพมากขึ้นได้

บรรณานุกรม

[1] A. Aboulnaga, P. J. Haas, M. Kandil, S. Lightstone, G. M. Lohman, V. Markl, I. Popivanov, and V. Raman, "Automated Statistics Collection in DB2 UDB," In Proceedings of the 30th International Conference on Very Large Data Bases, August 2004.

[2] A. I. Wand, C. F. Sørensen, "A Comparison of Two Different Java Technologies to Implement a Mobile Agent System," In proceedings of the IASTED International Conference on Applied Informatics 2003 (AI'2003), February 2003.

[3] A. Silberschatz , H. F. Korth , S. Sudarshan Database System Concepts,Third Edition, Mc Graw Hill, 1997.

[4] B. Brewington, R. Gray, K. Moizumi, D. Kotz, G. Cybenko, D. Rus, "Mobile agents in distributed information retrieval," In: M. Klusch (ed.) Inteligent Information Agents

[5] B. Groselj, Q. M. Maluhi, "Combinatorial Optimization of Distributed Queries," IEEE Trans. On Knowledge and Data Engineering, Vol. 17, December 1995.

[6] C. Gould, Z. Su, P. Devanbu, "Static Checking of Dynamically Generated Queries in Database Applications," ICSE 2004, Edingburgh, UK.

[7] C. Hsu, C. A. Knoblock, "Semantic query optimization for query plans of heterogeneous multidatabase systems," Knowledge and Data Engineering, Vol. 12, pp.959-978, 2000.

[8] D. Horvat, D. Cvetkovic, V. Milutinovic, P. Kocovic, V. Kovacevic, "Mobile Agents and Java Mobile Agents Toolkits," In Proceeding of the 33rd Hawaii International Conference on System Sciences, 2000.

[9] D. Kossmann, "The state of art in Distributed Query Processing," ACM Computing. Surveys, Vol. 32, pp.411-469, December 2000.

[10] D. Kotz, R. Gray, "Mobile Agents and Future of internet," in proceedings of ACM Operating Systems Review, pp.7-13, August 1999.

[11] H. Andrade, T. M. Kirc, A. Sussman, J. H. Saltz, "Optimizing the Execution of Multiple Data Analysis Queries on Parallel and Distributed Environments," IEEE Trans. Parallel Distributed System, pp.520-532, 2004.

[12] H. Kache, W. S. Han, V. Markl, S. Ewen, "POP/FED: Progressive Query Optimization for Federated Queries in DB2," ACM VLDB 06, September 2006.

[13] I. F. Ilyas, J. Rao, G. Lohman, D. Gao, and E. Lin, "Estimating Compilation Time of a Query Optimizer," In Proceedings of the ACM SIGMOD Conference on Management of Data , pp.373-384, June 2003.

[14] J. Cardiff, "Semantic query optimization in heterogeneous DBMSs," System Sciences, Vol. 2, pp. 273-282, January 1994.

[15] J. C. Wang, J. T. Horng, Y. M. Hsu, "A genetic algorithm for set query optimization in distributed database systems," IEEE Int. System, pp.14-17, 1996.

[16] L. F. Mackert, G. M. Lohman, "R* Optimizer Validation and Performance Evaluation for Distributed Queries," Proc 1986 VLDB Conference, pp.149-159, 1986.

[17] P. Bodorik, J.S Riordon, "Distributed Query Processing Optimization Objectives," proceeding of the fourth International Conference on Data Engineering, pp.320-329, 1988.

[18] P. Braun, J. Eismann, C. Erfurth, W. Rossak, "Concepts and Architecture of the Mobile Agent System Tracy,"

[19] P. M. G. Apers, A. R. Hevner, S. Bing Yao, "Optimization Algorithms for Distributed Queries," Software Engineering, IEEE Transactions, Vol. 9, pp.57-68, January 1983.

[20] Q. Zhu, B. Dunkel, N. Soparkar, S. Chen and B. Schiefer, T. Lai, "A Piggyback Method to Collect Statistics for Query Optimization in Database Management Systems," Proc. of 1998 Conference of the Centre for Advanced Studies on Collaborative Research (CASCON'98), pp 67-82, 1998.

[21] Q. Zhu, "An Integrated Method for Estimating Selectivities in a Multidatabase System," Proceeding of 1993 CAS Conference, Vol. 2, pp.832-847, 1993.

[22] S. Chaudhuri, "An Overview of Query Optimization in Relational Systems," In Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, pp34-43, June 1998.

[23] S. Chaudhuri, V. R. Narasayya, "Automating Statistics Management for Query Optimizers," IEEE Trans. Knowl. Data Eng, pp.7-20, 2001.

[24] S. Seshadri, V. Kumar, B. F. Cooper, "Optimizing Multiple Queries in Distributed Data Stream Systems," In Proceedings of the 22nd International Conference on Data Engineering Workshops (ICDEW'06), pp.25, 2006.

[25] T. Win, K. M. Lar Tun, "Mobile Agent Cooperation Methods in Hybrid Query Optimization," In Proceedings. 6th Asia-Pacific Symposium on Volume, pp.71-76, November 2005.

[26] V. Josifovski, T. Katchaounov, T. Risch, "Optimizing Queries in Distributed and Composable Mediators," CoopIS, pp.291-302, 1999

[27] V. S. Markl, V. Raman, D. Simmen, G. Lohman, H. Pirahesh, M. Cilimdzic, "Robust query processing through progressive optimization," ACM SIGMMOD international conference on Management of Paris, pp. 659-670, 2004.

ภาคผนวก

ภาคผนวก ก.

การกำหนดค่าของ Voyager

การกำหนดค่าของ Voyager

ในการใช้งาน Voyager หลังจากติดตั้งโปรแกรม Voyager เป็นที่เรียบร้อยแล้ว จำเป็นต้องกำหนดค่า Environment Variable ดังนี้

1. **JAVA_HOME** กำหนดให้ชี้ไปที่ไดเรกเทอรี่ที่ได้ติดตั้ง j2sdk ไว้ เช่น C:\jdk1.4\bin
2. **PATH** กำหนดให้ชี้ไปที่ไดเรกเทอรี่ที่ได้ติดตั้งโปรแกรม Voyager ไว้เช่น C:\voyager\bin
3. **CLASS_PATH** กำหนดให้ชี้ไปที่ไดเรกเทอรี่ที่มี class ไฟล์ของโปรแกรมที่ต้องการจะรัน และชี้ไปที่ไลบราลี่ทั้งหมดของ Voyager เช่น C:\voyager\lib\voyager.jar

ภาคผนวก ข.

ตัวอย่างคิวรี่ที่ใช้ในการทดลอง

ตัวอย่างคิวรี่ที่ใช้ในการทดลอง

Query 1

```
select   l_returnflag,l_linestatus,
         sum(l_quantity) as sum_qty,
         sum(l_extendedprice) as sum_base_price,
         sum(l_extendedprice * (1 - l_discount)) as sum_disc_price,
         sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) as sum_charge,
         avg(l_quantity) as avg_qty,avg(l_extendedprice) as avg_price,
         avg(l_discount) as avg_disc,count(*) as count_order
from     lineitem
where    l_shipdate <= date '1998-12-01' - interval '96' day (3)
group by
         l_returnflag,l_linestatus
order by
         l_returnflag,l_linestatus
```

Query 2

```
select   s_acctbal,s_name,n_name,p_partkey,
         p_mfgr,s_address,s_phone,s_comment
from     part,supplier,partsupp,nation,region
where    p_partkey = ps_partkey
         and s_suppkey = ps_suppkey
         and p_size = 28
         and p_type like '%STEEL'
         and s_nationkey = n_nationkey
         and n_regionkey = r_regionkey
         and r_name = 'MIDDLE EAST'
         and ps_supplycost = (
                  select   min(ps_supplycost)
                  from     partsupp,supplier,nation,region
```

```
                where
                        p_partkey = ps_partkey
                        and s_suppkey = ps_suppkey
                        and s_nationkey = n_nationkey
                        and n_regionkey = r_regionkey
                        and r_name = 'MIDDLE EAST')
        and rownum <= 100
order by
        s_acctbal desc,  n_name,s_name,p_partkey
```

Query 3

```
select  l_orderkey,sum(l_extendedprice * (1 - l_discount)) as revenue,
        o_orderdate,o_shippriority
from    customer,orders,lineitem
where   c_mktsegment = ' BUILDING'
        and c_custkey = o_custkey
        and l_orderkey = o_orderkey
        and o_orderdate < date ' 1995-03-31'
        and l_shipdate > date ' 1995-03-31'
        and rownum <= 10
group by
        l_orderkey,o_orderdate,o_shippriority
order by
        revenue desc,o_orderdate
```

Query 4

```
select  o_orderpriority,count(*) as order_count
from    orders
where   o_orderdate >= date ' 1997-10-01'
        and o_orderdate < date ' 1997-10-01' + interval '3' month
        and exists (
                select * from lineitem
```

```
                where l_orderkey = o_orderkey
                and l_commitdate < l_receiptdate)
group by
        o_orderpriority
order by
        o_orderpriority
```

Query 5

```
Select  n_name,
        sum(l_extendedprice * (1 -l_discount)) as revenue
from    customer,orders,lineitem,
        supplier,nation,region
where   c_custkey = o_custkey
        and l_orderkey = o_orderkey
        and l_suppkey = s_suppkey
        and c_nationkey = s_nationkey
        and s_nationkey = n_nationkey
        and n_regionkey = r_regionkey
        and r_name = ' MIDDLE EAST'
        and o_orderdate >= date ' 1994-01-01'
        and o_orderdate < date ' 1994-01-01' + interval '1' year
group by
        n_name
order by
        revenue desc
```

Query 6

```
select  sum(l_extendedprice * l_discount) as revenue
from    lineitem
where   l_shipdate >= date ' 1994-01-01'
        and l_shipdate < date ' 1994-01-01' + interval '1' year
        and l_discount between 0.05 - 0.01 and 0.05 + 0.01 and l_quantity < 25
```

Query 7

```
select   supp_nation,cust_nation,l_year,
         sum(volume) as revenue
from     (select  n1.n_name as supp_nation,
                  n2.n_name as cust_nation,
                  extract(year from l_shipdate) as l_year,
                  l_extendedprice * (1 - l_discount) as volume
         from     supplier,lineitem,orders,customer,
                  nation n1,nation n2
         where    s_suppkey = l_suppkey
                  and o_orderkey = l_orderkey
                  and c_custkey = o_custkey
                  and s_nationkey = n1.n_nationkey
                  and c_nationkey = n2.n_nationkey
                  and ((n1.n_name = ' IRAN'
                  and n2.n_name = ' UNITED STATES')
                  or (n1.n_name = ' UNITED STATES'
                  and n2.n_name = ' IRAN'))
                  and l_shipdate between
                  date '1995-01-01' and
                  date '1996-12-31')
group by
         supp_nation,cust_nation,l_year
order by
         supp_nation,cust_nation,l_year
```

Query 8

```
select   o_year, sum(
         case when nation = ' UNITED STATES'
         then volume else 0
         end) / sum(volume) as mkt_share
from     (select  extract(year from o_orderdate) as o_year,
```

```
                l_extendedprice * (1 - l_discount) as volume,
                n2.n_name as nation
        from    part,supplier,lineitem,orders,
                customer,nation n1,nation n2,region
        where   p_partkey = l_partkey
                and s_suppkey = l_suppkey
                and l_orderkey = o_orderkey
                and o_custkey = c_custkey
                and c_nationkey = n1.n_nationkey
                and n1.n_regionkey = r_regionkey
                and r_name = ' AMERICA'
                and s_nationkey = n2.n_nationkey
                and o_orderdate between
                date '1995-01-01' and
                date '1996-12-31'
                and p_type = ' MEDIUM BRUSHED STEEL')
group by
        o_year
order by
        o_year
```

Query 9

```
select  nation,o_year,
        sum(amount) as sum_profit
from    (select  n_name as nation,
                 extract(year from o_orderdate) as o_year,
                 l_extendedprice * (1 - l_discount) – ps_supplycost * l_quantity  as amount
        from     part,supplier,lineitem,
                 partsupp,orders,nation
        where    s_suppkey = l_suppkey
                 and ps_suppkey = l_suppkey
                 and ps_partkey = l_partkey
```

```
                and p_partkey = l_partkey
                and o_orderkey = l_orderkey
                and s_nationkey = n_nationkey
                and p_name like '% lime%')
group by
        nation,o_year
order by
        nation,o_year desc
```

Query 10

```
select  c_custkey,c_name,
        sum(l_extendedprice * (1 - l_discount)) as revenue,
        c_acctbal,n_name,c_address,
        c_phone,c_comment
from    customer,orders,lineitem,nation
where   c_custkey = o_custkey
        and l_orderkey = o_orderkey
        and o_orderdate >= date ' 1994-09-01'
        and o_orderdate < date ' 1994-09-01' + interval '3' month
        and l_returnflag = 'R'
        and c_nationkey = n_nationkey
        and rownum <= 20
group by
        c_custkey,c_name,c_acctbal,c_phone,
        n_name,c_address,c_comment
order by
        revenue desc
```

Query 11

```
select  ps_partkey,
        sum(ps_supplycost * ps_availqty) as value
from    partsupp,supplier,nation
where   ps_suppkey = s_suppkey
```

```
        and s_nationkey = n_nationkey
        and n_name = ' KENYA'
group by
        ps_partkey
having
        sum(ps_supplycost * ps_availqty) > (
                select  sum(ps_supplycost *
                ps_availqty) * 0.0001000000
        from    partsupp,supplier,nation
        where   ps_suppkey = s_suppkey
        and s_nationkey = n_nationkey
        and n_name = ' KENYA')
order by
        value desc
```

Query 12

```
select  l_shipmode,
        sum(case
                when o_orderpriority = '1-URGENT'
                or o_orderpriority = '2-HIGH'
                then 1 else 0
        end) as high_line_count,
        sum(case
                when o_orderpriority <> '1-URGENT'
                and o_orderpriority <> '2-HIGH'
                then 1 else 0
        end) as low_line_count
from    orders,lineitem
where   o_orderkey = l_orderkey and l_shipmode in (' SHIP', 'AIR')
        and l_commitdate < l_receiptdate and l_shipdate < l_commitdate
        and l_receiptdate >= date ' 1993-01-01'
        and l_receiptdate < date ' 1993-01-01' + interval '1' year
```

```
group by
        l_shipmode
order by
        l_shipmode
```

Query 13

```
with    c_orders as(select c_custkey,
        count(o_orderkey) as c_count
        from    customer left outer join orders on c_custkey = o_custkey
                and o_comment not like '%special%packages%'
        group by
                c_custkey)
select  c_count,count(*) as custdist
from    c_orders
group by
        c_count
order by
        custdist desc,c_count desc
```

Query 14

```
select  case  when p_type like 'PROMO%'
        then 100.00 * sum(l_extendedprice * (1 - l_discount))
        / sum(l_extendedprice * (1 - l_discount))
        else  0
        end  as promo_revenue
from    lineitem,part
where   l_partkey = p_partkey
        and l_shipdate >= date ' 1993-01-01'
        and l_shipdate < date ' 1993-01-01' + interval '1' month
group by
        p_type
```

Query 15

```
select   p_brand, p_type, p_size,
         count(distinct ps_suppkey) as supplier_cnt
from     partsupp, part
where    p_partkey = ps_partkey
         and p_brand <> ' Brand#45'
         and p_type not like ' ECONOMY BURNISHED%'
         and p_size in (13, 11, 15, 47, 34, 3, 27, 5)
         and ps_suppkey not in (
                  select   s_suppkey
                  from     supplier
                  where    s_comment like '%Customer%Complaints%')
group by
         p_brand, p_type, p_size
order by
         supplier_cnt desc, p_brand, p_type, p_size
```

Query 16

```
select   sum(l_extendedprice) / 7.0 as avg_yearly
from     lineitem, part
where    p_partkey = l_partkey
         and p_brand = ' Brand#54'
         and p_container = ' WRAP DRUM'
         and l_quantity < (
                  select   0.2 * avg(l_quantity)
                  from     lineitem
                  where    l_partkey = p_partkey)
```

Query 17

```
select   c_name, c_custkey, o_orderkey, o_orderdate,
         o_totalprice, sum(l_quantity)
from     customer, orders, lineitem
where    o_orderkey in (
```

```
                select  l_orderkey
                from    lineitem
                group by
                        l_orderkey having
                        sum(l_quantity) > 314)
        and c_custkey = o_custkey
        and o_orderkey = l_orderkey
        and rownum<=100
group by
        c_name, c_custkey, o_orderkey, o_orderdate, o_totalprice
order by
        o_totalprice desc, o_orderdate
```

Query 18

```
select  sum(l_extendedprice* (1 - l_discount)) as revenue
from    lineitem, part
where   (p_partkey = l_partkey
        and p_brand = ' Brand#53'
        and p_container in ('SM CASE', 'SM BOX', 'SM PACK', 'SM PKG')
        and l_quantity >= 6 and l_quantity <= 6 + 10
        and p_size between 1 and 5
        and l_shipmode in ('AIR', 'AIR REG')
        and l_shipinstruct = 'DELIVER IN PERSON')
        or
        (p_partkey = l_partkey
        and p_brand = ' Brand#55'
        and p_container in ('MED BAG', 'MED BOX', 'MED PKG', 'MED PACK')
        and l_quantity >= 20 and l_quantity <= 20 + 10
        and p_size between 1 and 10
        and l_shipmode in ('AIR', 'AIR REG')
        and l_shipinstruct = 'DELIVER IN PERSON')
        or
```

```
        (p_partkey = l_partkey
        and p_brand = ' Brand#55'
        and p_container in ('LG CASE', 'LG BOX', 'LG PACK', 'LG PKG')
        and l_quantity >= 22 and l_quantity <= 22 + 10
        and p_size between 1 and 15
        and l_shipmode in ('AIR', 'AIR REG')
        and l_shipinstruct = 'DELIVER IN PERSON')
```

Query 19

```
select  s_name, s_address
from    supplier, nation
where   s_suppkey in (
                select  ps_suppkey
                from    partsupp
                where   ps_partkey in (
                                select  p_partkey
                                from    part
                                where   p_name like 'ivory%')
                        and ps_availqty > (
                                select  0.5 * sum(l_quantity)
                                from    lineitem
                                where   l_partkey = ps_partkey
                                        and l_suppkey = ps_suppkey
                                        and l_shipdate >= date ' 1997-01-01'
                                        and l_shipdate < date ' 1997-01-01' + interval '1' year))
        and s_nationkey = n_nationkey
        and n_name = ' UNITED KINGDOM'
order by
        s_name
```

Query 20

```
select  s_name, count(*) as numwait
```

```
from    supplier, lineitem l1, orders, nation
where   s_suppkey = l1.l_suppkey
        and o_orderkey = l1.l_orderkey
        and o_orderstatus = 'F'
        and l1.l_receiptdate > l1.l_commitdate
        and exists (
                select  *
                from    lineitem l2
                where   l2.l_orderkey = l1.l_orderkey
                        and l2.l_suppkey <> l1.l_suppkey)
        and not exists (
                select  *
                from    lineitem l3
                where   l3.l_orderkey = l1.l_orderkey
                        and l3.l_suppkey <> l1.l_suppkey
                        and l3.l_receiptdate > l3.l_commitdate)
        and s_nationkey = n_nationkey
        and n_name = ' UNITED KINGDOM'
        and rownum <= 100
group by
        s_name
order by
        numwait desc, s_name
```