# รายงานการวิจัย เรื่อง
# กรณีศึกษาการใช้ Result Cache ปรับปรุงประสิทธิภาพการสืบค้นฐานข้อมูลของระบบบัญชี 3 มิติ มจพ.
# A case study of using result cache for improving KMUTNB GL3D database query performance


โดย

นางณิชาพร ชื่อสุธรรม

ฝ่ายพัฒนาระบบสารสนเทศ

สำนักคอมพิวเตอร์และเทคโนโลยีสารสนเทศ



แหล่งทุน

สำนักคอมพิวเตอร์และเทคโนโลยีสารสนเทศ

มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

พ.ศ. 2561 (ปีงบประมาณที่ได้รับทุน)

# กรณีศึกษาการใช้ Result Cache ปรับปรุงประสิทธิภาพการสืบค้นฐานข้อมูลของระบบบัญชี 3 มิติ มจพ.

นางณิชาพร ชื่อสุธรรม

แหล่งทุน
สำนักคอมพิวเตอร์และเทคโนโลยีสารสนเทศ
มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ
พ.ศ. 2561

ชื่อ : นางณิชาพร ชื่อสุธรรม
ชื่องานวิจัย : กรณีศึกษาการใช้ result cache ปรับปรุงประสิทธิภาพการสืบค้นฐานข้อมูลของระบบบัญชี 3 มิติ มจพ.
ส่วนงาน : สำนักคอมพิวเตอร์และเทคโนโลยีสารสนเทศ
มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ
ปีงบประมาณ : 2561

**บทคัดย่อ**

ระบบงบประมาณ พัสดุ การเงิน และบัญชีกองทุนโดยเกณฑ์พึงรับพึงจ่าย ลักษณะ 3 มิติ หรือเรียกอย่างย่อว่า ระบบบัญชี 3 มิติ มีการใช้งานกันในทุกส่วนงานของมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ (มจพ.) เมื่อมีปริมาณข้อมูลและจำนวนผู้ใช้งานเพิ่มขึ้น ในขณะที่ระบบฐานข้อมูลมีทรัพยากรเท่าเดิม การสืบค้นข้อมูลจึงมีประสิทธิภาพลดลง เนื่องจากมีภาระงานมากขึ้น

ผู้วิจัยมีแนวคิดที่จะปรับปรุงประสิทธิภาพในการสืบค้นข้อมูลด้วยการใช้ Result Cache เนื่องจากเป็นวิธีการที่ทำได้ง่าย ไม่ต้องจัดหาทรัพยากรเพิ่มเติม โดยหลักการของ Result Cache คือ เก็บผลลัพธ์ของการสืบค้นข้อมูลเอาไว้ใช้ซ้ำ เมื่อมีการสืบค้นแบบเดิมอีกครั้งหนึ่ง ระบบฐานข้อมูลจะส่งกลับผลลัพธ์ที่เก็บไว้ได้ทันที โดยไม่ต้องประมวลผลใหม่อีกครั้ง การใช้ Result Cache เหมาะสมกับข้อมูลที่มีการใช้งานบ่อย และมีการเปลี่ยนแปลงน้อย โดยเฉพาะอย่างยิ่งเหมาะกับการสืบค้นข้อมูลจำนวนมากที่ได้ผลลัพธ์การสืบค้นจำนวนน้อย

ผู้วิจัยได้ทดลองทำการสืบค้นข้อมูลแบบใช้และไม่ใช้ Result Cache บนเครื่องแม่ข่าย เพื่อเปรียบเทียบเวลาที่ใช้ในการสืบค้นทั้งสองแบบ ผลการทดลองแสดงให้เห็นว่า การสืบค้นข้อมูลแบบที่ใช้ Result Cache นั้นช่วยลดเวลาประมวลผลลงอย่างมีนัยสำคัญ ในกรณีที่มีผลลัพธ์ที่ต้องการพร้อมใช้งานอยู่ในหน่วยความจำ แต่จะใช้เวลาประมวลผลเพิ่มขึ้นเล็กน้อย ในกรณีที่ผลลัพธ์ที่ต้องการไม่มีอยู่ในหน่วยความจำหรือไม่พร้อมใช้งาน

(รายงานวิจัยมีจำนวนทั้งสิ้น 34 หน้า)

คำสำคัญ : การเก็บผลลัพธ์การสืบค้น, การปรับปรุงประสิทธิภาพการสืบค้น

Name : Mrs. Nichaporn Chuesutham

Research Title : A case study of using result cache for improving KMUTNB GL3D database query performance

Work for : Institute of Computer and Information Technology
King Mongkut's University of Technology North Bangkok

Academic Year : 2018

## Abstract

The budget, procurement, finance and general ledger in 3-dimension system, in abbreviated GL3D, is used by every department of King Mongkut's University of Technology North Bangkok (KMUTNB). When the volume of data and the number of users is increasing while the database system has the same resources, data retrieval might be less performance due to more workload.

The researcher has a notion to improve query performance by using result cache because it is easy to apply and no needed additional resource. Concept of result cache is to keep result of query for reuse. When the same query is called again, then the database system is immediately returned the stored result without re-execution. Result Cache is appropriated with frequently used and not rarely changes data, especially it is suitable for large amount data query with small amount result.

The researcher tested data queries using and not using server result cache to compare elapsed time of both queries. The testing results shows that querying using server result cache can reduce execution time significantly in the event that the desired result is available in the cache. But it will take a little more execution time in the event that no desired result is in the cache or it is not available.

(Total 34 pages)

Keyword : Query Result Cache, Query Performance Tuning

## กิตติกรรมประกาศ

งานวิจัยฉบับนี้สำเร็จลงได้ด้วยดี เนื่องจากได้รับความกรุณาให้คำปรึกษาและได้รับความช่วยเหลือจากผู้บริหาร และบุคลากรของสำนักคอมพิวเตอร์และเทคโนโลยีสารสนเทศทุกท่านเป็นอย่างดี จึงขอกราบขอบพระคุณเป็นอย่างสูงไว้ ณ ที่นี้

อนึ่ง ผู้วิจัยหวังว่างานวิจัยฉบับนี้จะมีประโยชน์ไม่น้อย จึงขอมอบส่วนดี ๆ ทั้งหมดเหล่านี้ให้แก่เหล่าคณาจารย์ที่ได้ประสิทธิประสาทวิชาจนทำให้ผลงานวิจัยเป็นประโยชน์ต่อผู้ที่เกี่ยวข้อง และขอมอบความกตัญญูกตเวทีตาคุณแด่บิดา มารดา และผู้มีพระคุณทุกท่าน สำหรับข้อบกพร่องต่าง ๆ ที่อาจจะเกิดขึ้นนั้น ผู้วิจัยขอน้อมรับผิดเพียงผู้เดียว และยินดีรับฟังคำแนะนำจากคณาจารย์รวมถึงบุคคลทั่วไปทุกท่านที่ได้เข้ามาศึกษา เพื่อเป็นประโยชน์ในการพัฒนางานวิจัยต่อไป

ณิชาพร ชื่อสุธรรม

# สารบัญ

**สารบัญ (ต่อ)**

**สารบัญตาราง**

**สารบัญภาพ**

# บทที่ 1
# บทนำ

## 1.1 ความเป็นมาและความสำคัญของปัญหา

ระบบงบประมาณ พัสดุ การเงิน และบัญชีกองทุนโดยเกณฑ์พึงรับพึงจ่าย ลักษณะ 3 มิติ หรือที่เรียกอย่างย่อว่า ระบบบัญชี 3 มิติ เป็นระบบสารสนเทศที่รองรับการดำเนินงานด้านการจัดซื้อจัดจ้างและการเงินการคลังของมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ (มจพ.) ซึ่งมีลักษณะเป็นระบบประมวลผลรายการ (Transaction Processing System : TPS) ระบบนี้มีการใช้งานกันในทุกส่วนงานของมหาวิทยาลัย การสืบค้นข้อมูลจากระบบมีอยู่ตลอดเวลา เมื่อมีปริมาณข้อมูลและจำนวนผู้ใช้งานเพิ่มมากขึ้น การสืบค้นข้อมูลจึงมีประสิทธิภาพลดลง เนื่องจากระบบฐานข้อมูลมีทรัพยากรเท่าเดิม แต่มีภาระงาน (Load) มากยิ่งขึ้น

การปรับปรุงประสิทธิภาพในการสืบค้นข้อมูลนั้นมีหลายวิธี วิธีการหนึ่งที่น่าสนใจคือการใช้ Result Cache เนื่องจากเป็นวิธีการที่ทำได้ง่าย ใช้เวลาดำเนินการไม่นาน ไม่ต้องจัดหาทรัพยากรเพิ่มเติม และคาดว่าน่าจะช่วยลดภาระงานของระบบฐานข้อมูลได้มาก จากผลการทดลองในบทความจำนวนหนึ่งซึ่งแสดงให้เห็นผลการทดลองว่า Result Cache ช่วยลดเวลาในการสืบค้นข้อมูลลงอย่างมาก โดยหลักการของ Result Cache คือการเก็บผลลัพธ์ของการสืบค้นข้อมูล (Result Cache) เอาไว้ใช้ในครั้งต่อไป ไม่ต้องประมวลผลคำสั่งสอบถามข้อมูล (Query) คำสั่งเดิมทุกครั้ง เพื่อลดภาระงานของระบบฐานข้อมูล การใช้ Result Cache จึงเหมาะสมกับข้อมูลที่มีการใช้งานบ่อย มีปริมาณน้อยและไม่ค่อยมีการเปลี่ยนแปลง เพื่อให้เกิดการใช้ซ้ำสูงสุด และไม่ทำให้เกิดการสับเปลี่ยนข้อมูลในหน่วยความจำ (Page Swapping) บ่อยๆ

งานวิจัยนี้ไม่ได้มุ่งเน้นการวัดประสิทธิภาพการทำงานของระบบฐานข้อมูลในสภาวะที่มีผู้ใช้จำนวนมากกำลังใช้งานระบบพร้อมกัน หากแต่ต้องการศึกษาผลการใช้ Result Cache ในฝั่งเซิร์ฟเวอร์ของระบบฐานข้อมูล Oracle ของระบบบัญชี 3 มิติ ว่ามีผลต่อความเร็วในการสืบค้นข้อมูลมากเพียงใด และข้อมูลปริมาณเท่าไรที่จะเหมาะสมกับการใช้ Result Cache โดยไม่ส่งผลให้เกิดการสับเปลี่ยนข้อมูลในหน่วยความจำบ่อยๆ

## 1.2 วัตถุประสงค์ของงานวิจัย

1.2.1 เพื่อศึกษาผลการใช้ Result Cache บนฝั่งเซิร์ฟเวอร์ที่มีต่อความเร็วในการสืบค้นข้อมูล

1.2.2 เพื่อศึกษาหาสัดส่วนของขนาดข้อมูลที่เหมาะสมเมื่อเทียบกับขนาดของ Result Cache

## 1.3 สมมติฐานของการวิจัย

การใช้ Result Cache ในฝั่งเซิร์ฟเวอร์ของระบบฐานข้อมูล Oracle ของระบบบัญชี 3 มิติ เมื่อใช้กับข้อมูลที่มีขนาดเหมาะสมจะมีผลให้การสืบค้นข้อมูลเร็วขึ้นอย่างมีนัยสำคัญ

## 1.4 ขอบเขตของการวิจัย

1.4.1 ขอบเขตด้านเนื้อหาที่ใช้ศึกษา

1.4.1.1 ศึกษาการใช้งาน Query Result Cache บนระบบฐานข้อมูล Oracle

1.4.1.2 ศึกษาวิธีการวัดประสิทธิภาพการสืบค้น (Query Performance)

1.4.1.3 ศึกษาระบบฐานข้อมูล Oracle

1.4.2 ขอบเขตด้านการทดลอง

งานวิจัยครั้งนี้เป็นการทดลองกับระบบฐานข้อมูล Oracle ซึ่งเก็บข้อมูลของระบบบัญชี 3 มิติ ที่มีชนิดข้อมูลเป็นตัวอักษร ตัวเลข และวันที่เท่านั้น

1.4.3 ระยะเวลาที่ใช้วิจัย

การวิจัยครั้งนี้ดำเนินการภายในระยะเวลา 7 เดือน

## 1.5 นิยามศัพท์เฉพาะ

1.5.1 Result Cache หมายถึง พื้นที่ส่วนหนึ่งในหน่วยความจำบนเซิร์ฟเวอร์ (Server) ที่ใช้เก็บผลลัพธ์ของการสืบค้นข้อมูล

1.5.2 ประสิทธิภาพการสืบค้น หมายถึง ความสามารถในการสืบค้นข้อมูลโดยพิจารณาจากระยะเวลา (Time) และทรัพยากร CPU (CPU Cost) ที่เซิร์ฟเวอร์ใช้ในการดำเนินการสืบค้นข้อมูล

1.5.3 ระบบฐานข้อมูลของระบบบัญชี 3 มิติ มจพ. หมายถึง ระบบฐานข้อมูลเชิงสัมพันธ์ (Relational Database) ที่มีการจัดเก็บข้อมูลชนิดตัวอักษร ตัวเลข และวันที่ ซึ่งใช้โปรแกรมจัดการฐานข้อมูล Oracle เป็นเครื่องมือ

## 1.6 ประโยชน์ของงานวิจัย

งานวิจัยนี้จะเป็นแนวทางที่เหมาะสมในการใช้ Result Cache เพื่อปรับปรุงประสิทธิภาพการสืบค้นข้อมูลของระบบบัญชี 3 มิติได้

# บทที่ 2
# ทฤษฎีและผลงานวิจัยที่เกี่ยวข้อง

ทฤษฎีที่เกี่ยวข้องในงานวิจัยฉบับนี้จะเป็นการศึกษาเกี่ยวกับหลักการพื้นฐานของระบบฐานข้อมูล และกระบวนการสืบค้นข้อมูลในระบบฐานข้อมูล Oracle และการวัดประสิทธิภาพการสืบค้น (Query Performance Measurement) โดยครอบคลุมหัวข้อต่าง ๆ ดังนี้

## 2.1 ระบบฐานข้อมูลของระบบบัญชี 3 มิติ

งานวิจัยนี้เป็นกรณีศึกษาบนระบบฐานข้อมูลของระบบบัญชี 3 มิติ ซึ่งใช้ Oracle Database เวอร์ชัน 11g Release 2 เป็นระบบจัดการฐานข้อมูล (Database Management System - DBMS)

ฐานข้อมูลของระบบบัญชี 3 มิติ เป็นระบบฐานข้อมูลเชิงสัมพันธ์ (Relational Database) ซึ่งจัดเก็บข้อมูลในรูปแบบของตาราง ประกอบไปด้วยชนิดข้อมูลที่เป็นตัวอักษร ตัวเลข และวันที่ ปัจจุบันมีจำนวนประมาณ 155 ตาราง ตารางที่มีขนาดเล็กที่สุดมีขนาด 64 KB 2 แถว 3 คอลัมน์ ตารางที่มีขนาดใหญ่สุดมีขนาด 702 MB 7.8 ล้านแถว 29 คอลัมน์

## 2.2 หลักการพื้นฐานเกี่ยวกับระบบฐานข้อมูล Oracle

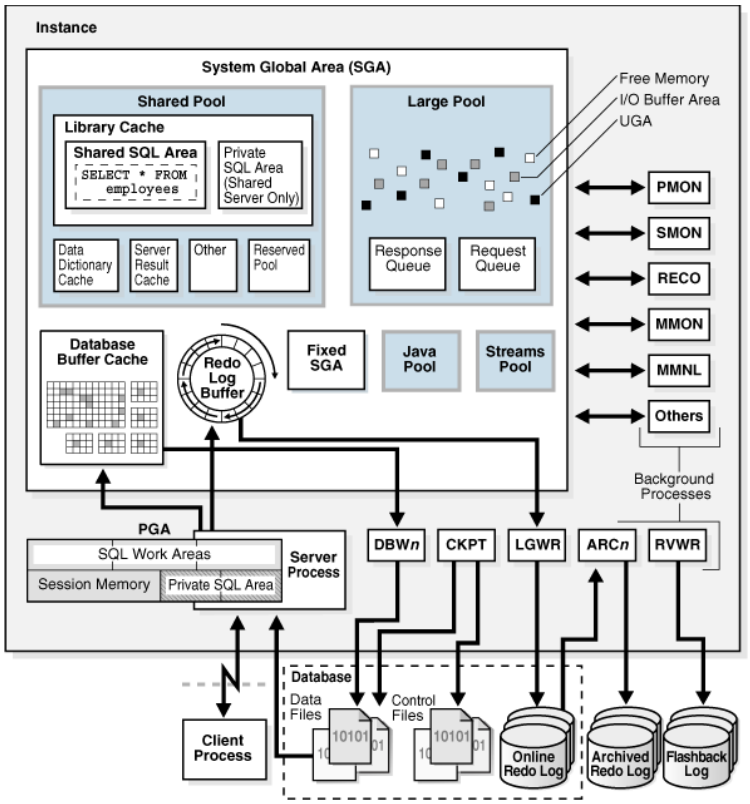
Database Server คือ เครื่องแม่ข่ายฐานข้อมูลซึ่งทำหน้าที่บริหารจัดการข้อมูลจำนวนมากในสิ่งแวดล้อมที่มีผู้ใช้หลายคนสามารถเข้าถึงข้อมูลเดียวกันในช่วงเวลาเดียวกันได้อย่างน่าเชื่อถือ รวมถึงป้องกันการเข้าถึงข้อมูลโดยผู้ที่ไม่มีสิทธิ์ และมีวิธีการกู้คืนระบบฐานข้อมูลที่มีประสิทธิภาพเมื่อเกิดความล้มเหลวอีกด้วย

Oracle Database Server นั้นประกอบด้วย Database และ Database Instance (นิยมเรียกอย่างย่อว่า Instance) อย่างน้อย 1 ตัว แต่เนื่องจาก Database และ Instance มีความเชื่อมโยงกันอย่างใกล้ชิด บางครั้งการเอ่ยถึง Oracle Database จึงหมายรวมถึงทั้ง Database และ Instance โดยไม่ได้เจาะจงอย่างใดอย่างหนึ่ง ส่วนความหมายโดยเฉพาะของทั้งสองคำนั้นเป็นดังต่อไปนี้

Oracle Database คือ ชุดของไฟล์ที่อยู่บนดิสก์ซึ่งเป็นที่เก็บข้อมูล ไฟล์เหล่านั้นมีอยู่อย่างเป็นอิสระจาก Database Instance ใดๆ โดยโครงสร้างทางกายภาพในการจัดเก็บข้อมูลของ Database จะประกอบไปด้วย Data Files และ Control Files รวมทั้ง Online Redo Log

- Data Files เป็นที่จัดเก็บข้อมูลทั้งหมดของฐานข้อมูล ข้อมูลในตาราง และ Index ต่างๆ ซึ่งเป็นโครงสร้างข้อมูลเชิงตรรกะของฐานข้อมูลจะถูกจัดเก็บอยู่ใน Data File เหล่านี้ ฐานข้อมูลจะต้องมี Data Files อย่างน้อย 1 ไฟล์ขึ้นไป
- Control File เป็นที่จัดเก็บข้อมูล Metadata หรือข้อมูลที่อธิบายรายละเอียดของฐานข้อมูล เช่น ชื่อของฐานข้อมูล ชื่อและตำแหน่งของ Data Files เป็นต้น ฐานข้อมูลจะมี Control File ได้เพียง 1 ไฟล์เท่านั้น
- Online Redo Log เป็นที่จัดเก็บบันทึกการเปลี่ยนแปลงทั้งหมดที่เกิดขึ้นกับข้อมูล ฐานข้อมูลจะมี Online Redo Log เพียง 1 ชุดเท่านั้น แต่ใน Online Redo Log จะประกอบไปด้วย Redo Log Files ตั้งแต่ 2 ไฟล์ขึ้นไป

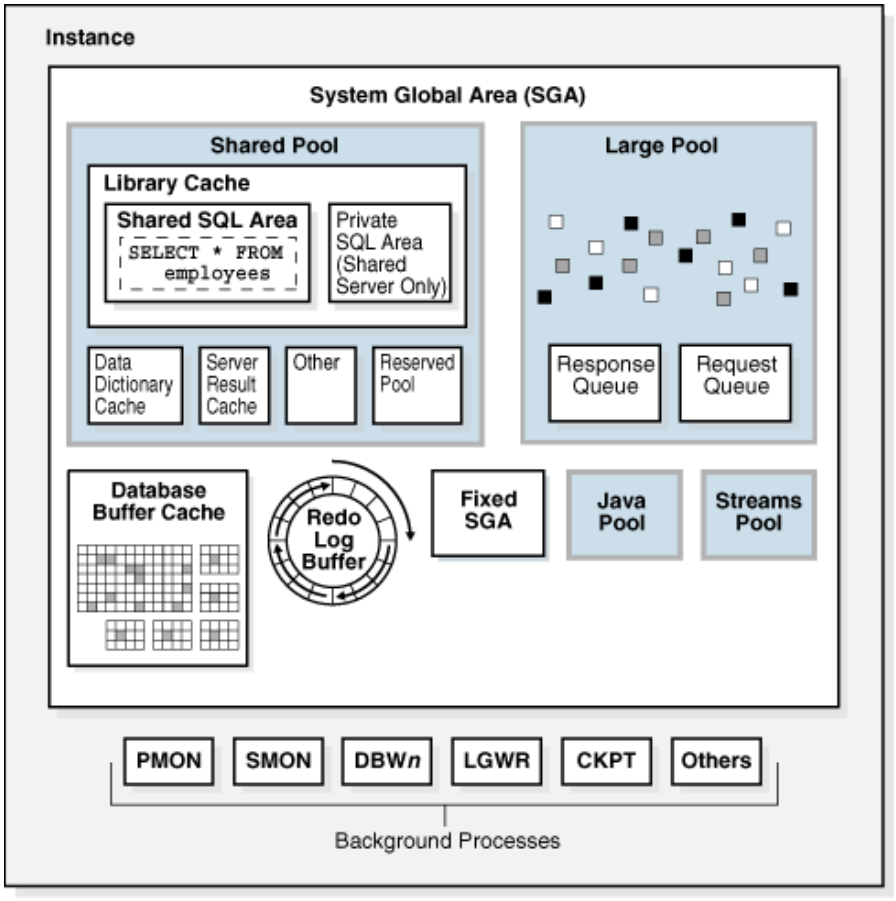
Oracle Database Instance คือ ชุดของโครงสร้างหน่วยความจำที่ใช้ในการจัดการไฟล์ต่างๆ ของฐานข้อมูล โดย Instance มีอยู่อย่างเป็นอิสระจากไฟล์ฐานข้อมูลเหล่านั้น โดย Instance จะประกอบด้วยพื้นที่หน่วยความจำที่ใช้งานร่วมกัน (Shared Memory area) ที่เรียกว่า System Global Area (SGA) และชุดของ Background Process จำนวนหนึ่ง




**ภาพที่ 2-1** Oracle Instance and Database [1]

ภาพที่ 2-1 แสดงให้เห็นถึง Database และ Instance เมื่อผู้ใช้เรียกใช้งานแอปพลิเคชันบนเครื่องคอมพิวเตอร์ของผู้ใช้ (นิยมเรียกว่า Client) แอปพลิเคชันนั้นจะถูกดำเนินการโดย Client

Process การเชื่อมต่อกับ Database Server จะเป็นการเชื่อมต่อกับ Instance โดยแต่ละ Client Process เมื่อเชื่อมต่อกับ Instance แล้ว จะมี Server Process ของตัวเองเกิดขึ้นเพื่อดำเนินงานบนฝั่งเซิร์ฟเวอร์ ให้ โดยแต่ละ Server Process จะถูกจัดสรรพื้นที่หน่วยความจำส่วนตัวที่เรียกว่า Program Global Area (PGA) ให้เป็นของตนเอง




**ภาพที่ 2-2** Database Instance เฉพาะส่วนของ SGA [2]

เมื่อ Instance ถูกสั่งให้เริ่มต้นทำงาน ระบบฐานข้อมูลจะทำการจองพื้นที่หน่วยความจำที่เรียกว่า System Global Area (SGA) ขึ้นเพื่อให้ Oracle Process ใช้งานร่วมกัน รวมทั้งสร้าง Background Process ต่างๆ ขึ้น ภาพที่ 2-2 แสดงให้เห็นส่วนประกอบภายใน SGA โดยส่วนประกอบที่สำคัญ มีดังนี้

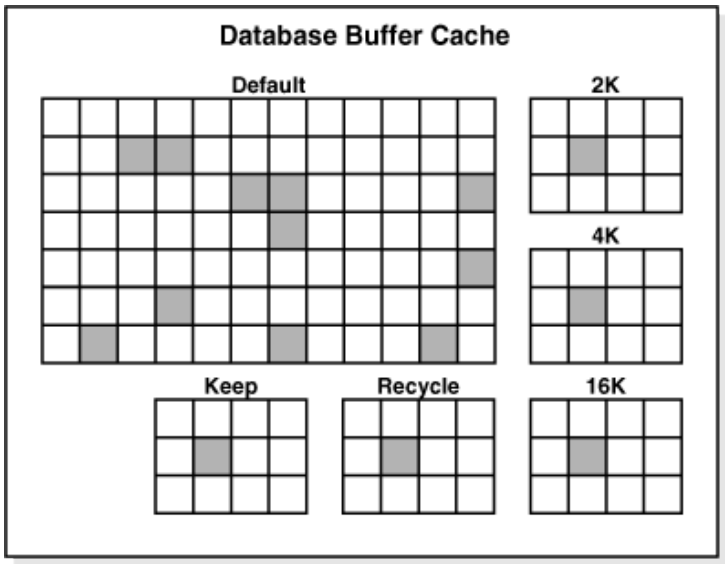
1. Database Buffer Cache

พื้นที่สำหรับเก็บสำเนาของข้อมูลที่ถูกอ่านจาก Data Files บนดิสก์ เรียกอย่างย่อว่า Buffer Cache ข้อมูลที่ถูกอ่านขึ้นมาจะอยู่ในหน่วยของข้อมูลที่เรียกว่า Data Block โดยปกติ 1 Data Block จะมีขนาด 8 KB (ขนาดมาตรฐานขึ้นอยู่กับขนาด Block ของระบบปฏิบัติการ) การจัดการพื้นที่ใน Buffer Cache จะใช้อัลกอรทึม Least Recently Used (LRU) นั่นคือข้อมูลที่ถูกอ่าน

ขึ้นมาเพื่อใช้งานล่าสุดจะถูกเก็บไว้ใน Cache นี้ ส่วนข้อมูลที่ถูกใช้งานมื่อนานมาแล้วจะถูกกำจัดออกเพื่อให้มีพื้นที่ว่างสำหรับข้อมูลที่ถูกอ่านจากดิสก์ชุดต่อไปมาแทน ข้อมูลใน Buffer ส่วนที่ถูกเรียกใช้งานบ่อยๆ เรียกว่า Hot Buffer ส่วนที่ไม่ค่อยถูกเรียกใช้งานเรียกว่า Cold Buffer

เมื่อข้อมูลมีการเปลี่ยนแปลง ระบบฐานข้อมูลจะทำการปรับปรุงข้อมูลใน Buffer Cache นี้ รวมถึงเก็บข้อมูลการเปลี่ยนแปลงลงใน Redo Log Buffer หลังจากมีการ Commit เพื่อยืนยันการเปลี่ยนแปลงข้อมูลแล้ว ระบบฐานข้อมูลจะเขียนข้อมูลใน Redo Log Buffer ลงบนดิสก์ แต่จะไม่เขียนข้อมูลใน Data Block ลงบนดิสก์ในทันที โดย Database Writer (DBW) ซึ่งเป็นผู้ทำหน้าที่เขียนข้อมูลจาก Buffer Cache ลงใน Data Files จะทำการเขียนข้อมูลในลักษณะ Background Process ในภายหลัง (Lazy Write) เมื่อมีการเรียกใช้ข้อมูลจากไคลเอนท์ ขณะที่ข้อมูลถูกเปลี่ยนแปลง แต่ยังไม่มีการ Commit ระบบฐานข้อมูลจะอ่านข้อมูลจาก Buffer Cache ได้สอง รูปแบบ คือ Current Mode Get หรือ DB Block Get ซึ่งเป็นการอ่านข้อมูลในเวอร์ชันที่มีการเปลี่ยนแปลงแล้ว และ Consistent Read Get หรือ Consistent Get ซึ่งเป็นการอ่านข้อมูลในเวอร์ชันที่ยังไม่มีการปลี่ยนแปลง โดยปกติในการสืบค้นข้อมูลหรือ Select Statement จะใช้การอ่านแบบ Consistent Get ส่วน DB Block Get จะใช้ในระหว่างกระบวนการเปลี่ยนแปลงข้อมูล

สถานะของ Buffer มีได้ 3 สถานะ ได้แก่ สถานะ Unused หมายถึง Buffer ไม่ได้ถูกใช้งานในปัจจุบัน สถานะ Clean หมายถึง Buffer มีข้อมูลอยู่และไม่มีการเปลี่ยนแปลงหรือเป็นเวอร์ชัน Read-Consistent และสถานะ Dirty หมายถึง Buffer มีข้อมูลที่มีการเปลี่ยนแปลงอยู่และยังไม่ถูกเขียนลงดิสก์




**ภาพที่ 2-3** Database Buffer Cache [3]

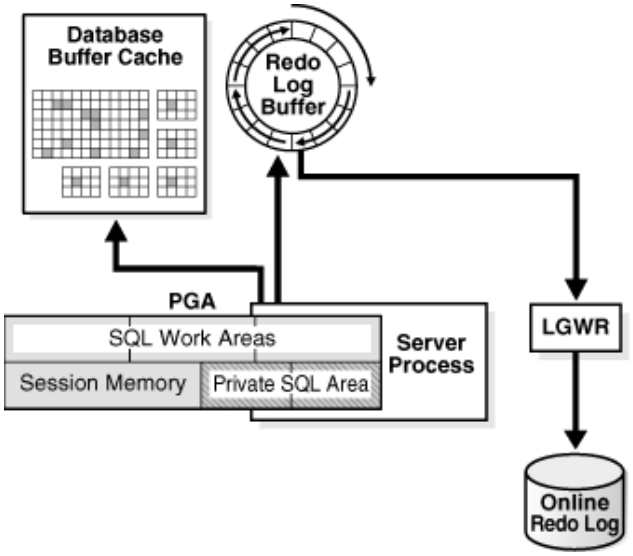
การอ่านและเขียนข้อมูลบน Buffer Cache เรียกว่า Logical I/O หรือ Buffer I/O เมื่อมีการร้องขอข้อมูลที่ไม่มีอยู่บน Buffer Cache ระบบฐานข้อมูลจะทำการอ่านข้อมูลจากดิสก์ ซึ่งจะ

เรียกว่า Physical I/O แล้วนำข้อมูลมาสำเนาไว้บน Buffer Cache หลังจากนั้นจะเกิด Logical I/O เพื่ออ่านข้อมูลบน Buffer ต่อไป

Buffer Pool คือกลุ่มของ Buffer ซึ่งใน Buffer Cache จะมีเพียงหนึ่ง Buffer Pool หรือหลาย Buffer Pool ก็ได้ โดย Buffer Pool จะแบ่งเป็นสามประเภท ได้แก่ Default Pool ซึ่งจะเก็บ Data Block ที่มีขนาดมาตรฐานที่มีการใช้งานตามปกติ ส่วน Keep Pool จะเป็นที่เก็บ Data Block ที่ถูกใช้งานบ่อย แต่ถูกกำจัดออกจาก Default Pool เนื่องจากพื้นที่ไม่เพียงพอ และประเภทสุดท้าย Recycle Pool จะเป็นที่เก็บ Data Block ที่ถูกใช้งานไม่ค่อยบ่อย แต่ยังไม่สมควรถูกกำจัดออกจาก Cache เผื่อมีการเรียกใช้งานอีก ระบบฐานข้อมูลจะจัดการกับ Data Block ที่มีขนาดต่างกัน โดยแยกไว้ต่าง Buffer Pool กัน แต่มีการจัดการด้วยอัลกอริทึมแบบเดียวกัน ภาพที่ 2-3 แสดงให้เห็นถึงส่วนประกอบภายใน Database Buffer Cache โดยขนาดมาตรฐานของ Data Block คือ 8 KB ดังนั้นใน Default Pool จึงเก็บ Data Block ที่มีขนาด 8 KB ส่วน Data Block ที่มีขนาดอื่น จะถูกเก็บอยู่ใน Pool ของตนเองแยกกันตามขนาด Data Block

2. Redo Log Buffer

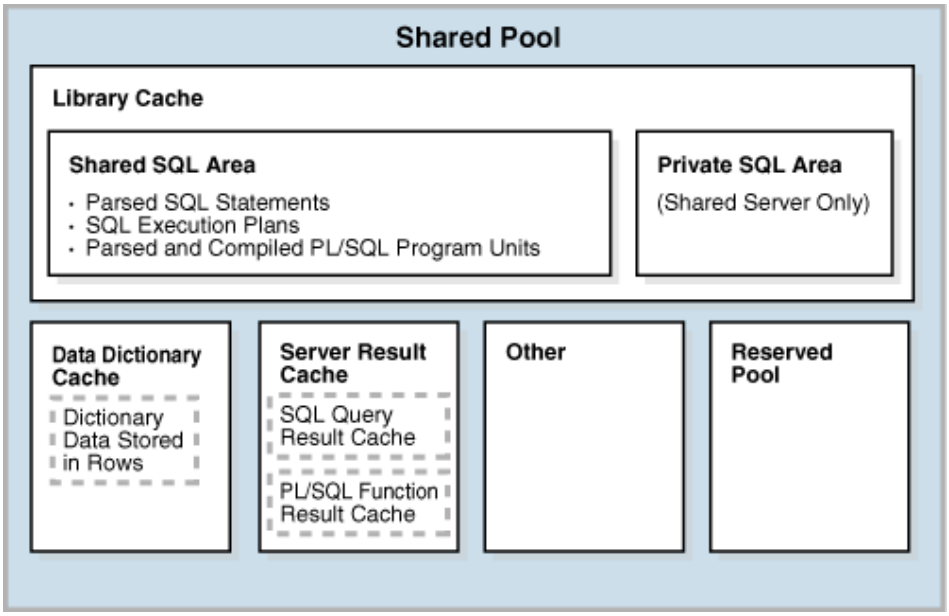
พื้นที่สำหรับเก็บรายการการเปลี่ยนแปลงที่เกิดขึ้นกับฐานข้อมูล ไว้เพื่อประโยชน์ในการกู้คืน (Recovery) ในภายหลัง เมื่อฐานข้อมูลเกิดความเสียหายและข้อมูลที่มีอยู่ไม่เป็นปัจจุบันหรือไม่มีการเปลี่ยนแปลงตรงกับ Redo Log เมื่อเกิดการเปลี่ยนแปลงกับฐานข้อมูล ระบบฐานข้อมูลจะบันทึกรายการเปลี่ยนแปลงนั้นลงใน Redo Log Buffer จากนั้น Log Writer (LGWR) จะทำหน้าที่เขียนข้อมูลใน Redo Log Buffer ลงไปที่ Online Redo Log Group บนดิสก์ โดยเป็นการเขียนแบบต่อเนื่องตามลำดับ (Sequentially) ต่างจาก DBW ที่เขียนแบบกระจัดกระจาย ตามจุดที่ข้อมูลเก็บอยู่ ดังนั้น LGWR จึงทำงานได้เร็วกว่า DBW ภาพที่ 2-4 แสดงกลไกของ Redo Log Buffer




**ภาพที่ 2-4** Redo Log Buffer [3]

3. Shared Pool

พื้นที่สำหรับเก็บข้อมูลหลากหลายประเภท ซึ่งเกี่ยวข้องกับแทบทุกการกระทำที่เกิดขึ้นกับฐานข้อมูล ส่วนประกอบภายใน Shared Pool แสดงตามภาพที่ 2-5




**ภาพที่ 2-5** Shared Pool [3]

ส่วนประกอบที่สำคัญของ Shared Pool มีดังนี้

Library Cache

เป็นพื้นที่ส่วนกลางที่ผู้ใช้ทั้งหมดสามารถใช้งานร่วมกันได้ มีไว้สำหรับเก็บคำสั่ง SQL และ PL/SQL ที่สามารถใช้ประมวลผลได้ ซึ่งจะแบ่งเป็นพื้นที่สำหรับ SQL และ PL/SQL รวมทั้งโครงสร้างการควบคุมต่างๆ เช่น Lock ต่างๆ และตัวจัดการ (handles) ของ Library Cache เมื่อ SQL Statement ถูกเรียกใช้งาน ระบบฐานข้อมูลจะพยายามนำคำสั่งที่เคยประมวลผลไปแล้วมาใช้ซ้ำ ถ้าพบว่ามี SQL Statement เดียวกันอยู่ใน Library Cache แล้ว ก็จะนำคำสั่งนั้นไปประมวลผลเลย ลักษณะอย่างนี้เรียกว่า Soft Parse หรือ Library Cache Hit ถ้าไม่พบ ระบบฐานข้อมูลจะจัดสรรพื้นที่ใน Shared SQL Area และเก็บคำสั่งที่ประมวลผลได้ที่สร้างขึ้นมาใหม่ ซึ่งเรียกว่า Hard Parse หรือ Library Cache Miss การจัดเก็บคำสั่งที่ประมวลผลได้ของ SQL Statement หนึ่งๆ จะเก็บได้เพียงที่เดียวเท่านั้น ไม่มีการเก็บซ้ำซ้อนกัน ระบบฐานข้อมูลโดยทั่วไปจะใช้พื้นที่ Library Cache ในส่วนของ Shared SQL Area ยกเว้นระบบฐานข้อมูลที่มีสถาปัตยกรรมแบบ Shared Server จะใช้ Private SQL Area เพื่อใช้เก็บคำสั่งที่ประมวลผลได้แยกสำหรับแต่ละ Session สำหรับส่วนของ PL/SQL Program และ Java Class ระบบฐานข้อมูลจะจัดการกับโปรแกรมที่ใช้

ประมวลผลได้ในลักษณะเดียวกันกับ SQL Statement แต่จะจัดสรรพื้นที่ใน Private SQL Area เพื่อเก็บค่าเฉพาะของแต่ละ Session เช่น ค่าของตัวแปรต่างๆ ด้วย

Data Dictionary Cache

เป็นข้อมูลอ้างอิงที่เกี่ยวกับตารางและวิวต่างๆ ในฐานข้อมูล เช่น รายละเอียดโครงสร้าง ผู้ใช้งานที่เป็นเจ้าของ เป็นต้น ระบบฐานข้อมูลจะเข้าถึง Data Dictionary บ่อยครั้ง เพื่อใช้ในการแปลง (Parse) SQL Statement ให้เป็นคำสั่งที่ประมวลผลได้ Data Dictionary จะเก็บข้อมูลในลักษณะแถว (Row) แทนที่จะเป็น Buffer จึงถูกเรียกอีกอย่างว่า Row Cache

Server Result Cache

เป็นพื้นที่จัดเก็บผลลัพธ์ที่ได้จากการประมวลผล SQL Select Statement ลักษณะการเก็บข้อมูลจะเป็นชุดของผลลัพธ์ (Result Set) ไม่ได้เก็บเป็น Data Block เหมือนกับ Buffer Cache การจัดการกับพื้นที่ภายใน Server Result Cache จะใช้อัลกอริทึม LRU เช่นเดียวกันกับ Buffer ส่วนประกอบภายในแบ่งเป็น SQL Query Result Cache และ PL/SQL Function Result Cache

SQL Query Result Cache เป็นพื้นที่สำหรับเก็บผลลัพธ์ของการประมวลผล Select Statement เมื่อมีการเรียกใช้งาน Select Statement ที่ใช้ Result Cache ระบบฐานข้อมูลจะตรวจสอบว่าภายใน Query Result Cache มีผลลัพธ์ของ Select Statement ดังกล่าวอยู่หรือไม่ ถ้าพบจะส่งผลลัพธ์นั้นกลับไปให้ผู้ใช้ทันที ถ้าไม่พบจะทำการประมวลผลคำสั่ง Select Statement และเก็บผลลัพธ์ไว้ใช้ซ้ำในครั้งถัดไป ด้วยวิธีการนี้ระบบฐานข้อมูลจึงไม่ต้องอ่านข้อมูลจาก Data Block และคำนวณผลลัพธ์ใหม่ทุกครั้ง เมื่อมีการเปลี่ยนแปลงข้อมูลของตารางหรือวิวที่ Select Statement นั้นอ้างอิงอยู่ ระบบฐานข้อมูลจะกำหนดให้ผลลัพธ์ที่เก็บใน Result Cache มีสถานะเป็น Invalid โดยอัตโนมัติ เพื่อให้มีการอ่านข้อมูลที่เปลี่ยนแปลง คำนวณผลลัพธ์ และจัดเก็บผลลัพธ์ใหม่ เมื่อมีการร้องขอให้ประมวลผล Select Statement นั้นอีก

PL/SQL Function Result Cache เป็นพื้นที่สำหรับเก็บผลลัพธ์การประมวลผลของ Function ทำนองเดียวกันกับ SQL Query Result Cache เมื่อมีการสั่งให้ฟังก์ชันเดิมทำงานอีกครั้งหนึ่งด้วยค่าพารามิเตอร์เดียวกัน ระบบฐานข้อมูลจะส่งผลลัพธ์ที่อยู่ใน Function Result Cache กลับไปยังผู้ใช้ได้ทันที แต่ถ้าไม่พบผลลัพธ์นั้นใน Function Result Cache ก็จะทำการประมวลผลฟังก์ชันใหม่และเก็บผลลัพธ์นั้นลงใน Function Result Cache

Reserved Pool

เป็นพื้นที่ภายใน Shared Pool ที่ใช้สำหรับจัดสรรพื้นที่ขนาดใหญ่ที่ต่อเนื่องกันสำหรับวัตถุที่มีขนาดใหญ่กว่า 5 KB ให้สามารถบรรจุอยู่ใน Cache โดยไม่เกิดปัญหา Fragmentation

4. Large Pool

พื้นที่สำรองไว้เพื่อการจัดสรรพื้นที่ขนาดใหญ่เกินกว่าที่ควรจะจัดสรรใน Shared Pool การจัดการพื้นที่ใน Large Pool ไม่ได้ใช้อัลกอริทึม LRU โดยเมื่อมีการจัดสรรพื้นที่ส่วนใดใน Large Pool แล้วกระบวนการทำงานที่ใช้พื้นที่นั้นจะต้องทำงานเสร็จสิ้นทั้งหมดก่อน พื้นที่ส่วนนั้นจึงจะว่างสำหรับใช้งานได้

5. Java Pool

พื้นที่สำหรับเก็บข้อมูลเฉพาะแต่ละ Session ของ Java Code รวมถึงข้อมูลภายใน Java Virtual Machine ด้วย

6. Streams Pool

พื้นที่สำหรับเก็บ Buffered Queue Messages และ Process ต่างๆ ของ Oracle Stream เป็นพื้นที่ที่มีไว้สำหรับให้ Oracle Stream ใช้งานเท่านั้น

7. Fixed SGA

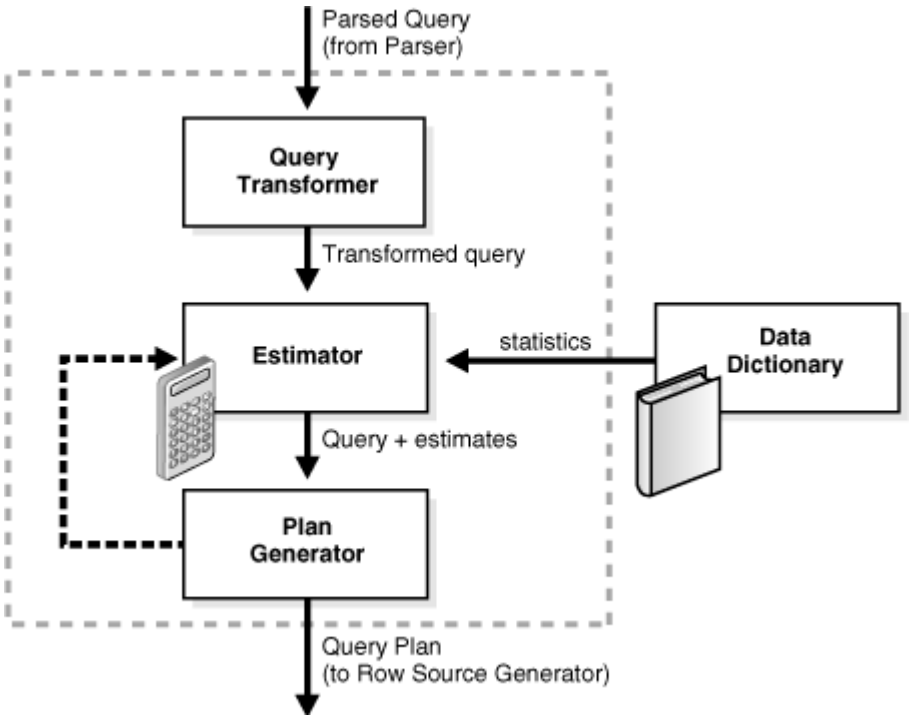
พื้นที่สำหรับระบบฐานข้อมูลใช้ในการจัดการภายในเท่านั้น เช่น สถานะของ Database และ Instance ที่ Background Process ต่างๆ ต้องใช้ หรือข้อมูลที่ใช้สื่อสารกันระหว่าง Process

## 2.3 กระบวนการสืบค้นข้อมูลในระบบฐานข้อมูล Oracle

ส่วนประกอบของระบบฐานข้อมูล Oracle ที่มีความสำคัญอย่างยิ่งต่อกระบวนการสืบค้นข้อมูลนั่นก็คือ Optimizer บางทีถูกเรียกว่า Query Optimizer หรือ Cost-based Optimizer ที่ทุกๆ SQL Statement ใช้เพื่อพิจารณาหาความหมายที่ดีที่สุดในการเข้าถึงข้อมูลที่ Statement นั้นระบุไว้

วิธีการประมวลผล SQL Statement อาจจะมีได้หลายวิธีที่แตกต่างกัน เช่น ลำดับการเข้าถึงตารางหรือดัชนีที่แตกต่างกัน เป็นต้น Optimizer จะทำการสร้าง Execution Plan ขึ้นมาจำนวนหนึ่ง ซึ่งแต่ละแผนจะอธิบายวิธีการที่เป็นไปได้ในการประมวลผล โดย Optimizer จะประเมินหา Execution Plan ที่มีประสิทธิภาพสูงสุด โดยพิจารณาจากข้อมูลหลายแหล่ง รวมถึงเงื่อนไขของการสืบค้น เส้นทางการเข้าถึงข้อมูล สถิติที่ระบบฐานข้อมูลเก็บรวมรวมไว้ และคำแนะนำ (Hint) เป็นต้น

Optimizer สร้างวิธีการที่เป็นไปได้ทั้งหมดในการประมวลผล และคำนวณต้นทุน (Cost) ที่ใช้ในแต่ละขั้นตอนของ Execution Plan ที่สร้างขึ้นนั้น โดยแผนที่มีค่าต้นทุนต่ำที่สุดจะถูกเลือกเป็น Query Plan เพื่อทำการประมวลผล ส่วนประกอบของ Optimizer แสดงในภาพที่ 2-6




**ภาพที่ 2-6** ส่วนประกอบของ Optimizer [4]

ข้อมูลขาเข้าของ Optimizer คือ SQL Statement ที่ผ่านการ Parsing แล้ว ซึ่งจะเรียกว่า Parsed Query โดย Optimizer จะทำงานตามขั้นตอน ดังนี้

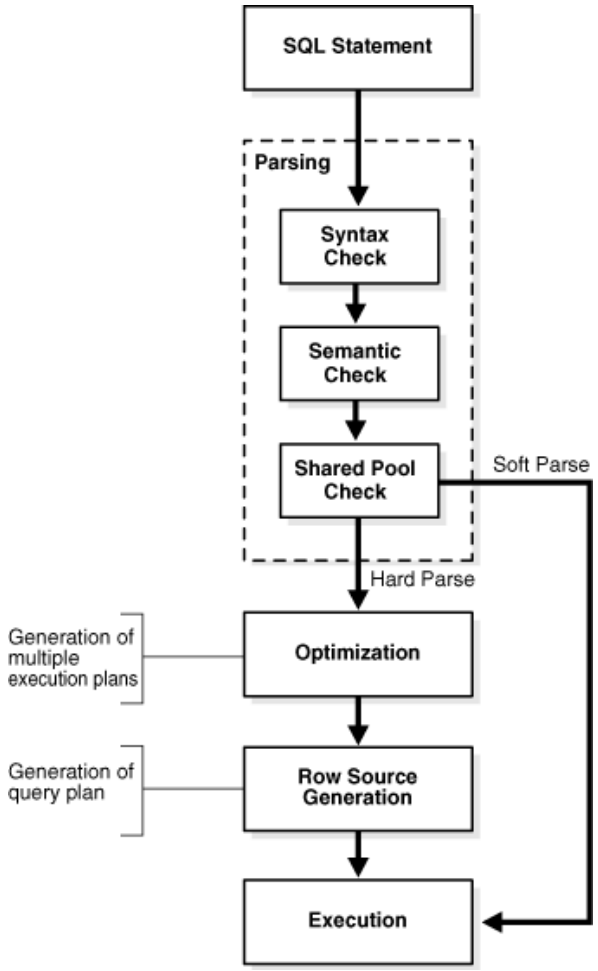
1. รับ Parsed Query แล้วทำการสร้างแผนการประมวลผลที่มีศักยภาพขึ้นมาจำนวนหนึ่ง โดยพิจารณาจากเส้นทางการเข้าถึงข้อมูลและคำแนะนำที่ระบุใน Statement
2. ประเมินต้นทุนของแต่ละแผนจากค่าสถิติใน Data Dictionary โดยต้นทุนถูกคำนวณจากสัดส่วนการใช้ทรัพยากรที่จำเป็นต่อการประมวลผลตามวิธีการที่ระบุในแผน
3. เปรียบเทียบต้นทุนของแต่ละแผน และเลือกแผนที่มีต้นทุนต่ำที่สุดเป็น Query Plan แล้วส่งต่อให้ Row Source Generator

Query Transformer จะพยายามแปลง Parsed Query เป็นรูปแบบที่แตกต่างจากเดิมเพื่อพิจารณาว่าจะสามารถสร้าง Execution Plan ที่ดีกว่าได้หรือไม่

Estimator จะประเมินต้นทุนรวมของแต่ละแผนโดยสร้างค่าที่ใช้วัดขึ้นมา 3 ชนิด ได้แก่ Selectivity จำนวนแถวที่ถูกเลือกในชุดข้อมูล Cardinality จำนวนแถวในชุดข้อมูล และ Cost หน่วยของการทำงานหรือทรัพยากรที่ถูกใช้งาน และถ้ามีข้อมูลสถิติอื่นที่ใช้ได้ Estimator จะใช้ประกอบการพิจาณาเพื่อเพิ่มความแม่นยำในการประเมินด้วย

Plan Generator จะพยายามทดลองสร้างแผนที่แตกต่างกันสำหรับ Statement โดยสร้างจากเส้นทางการเข้าถึงข้อมูล วิธีการ Join และลำดับการ Join ที่ต่างกัน

การประมวลผล SQL Statement ในระบบฐานข้อมูล Oracle แบ่งเป็น 4 ระยะ ได้แก่ Parsing Optimization Row Source Generation และ Execution ดังในภาพ 2-7




**ภาพที่ 2-7** กระบวนการประมวลผล SQL Statement [4]

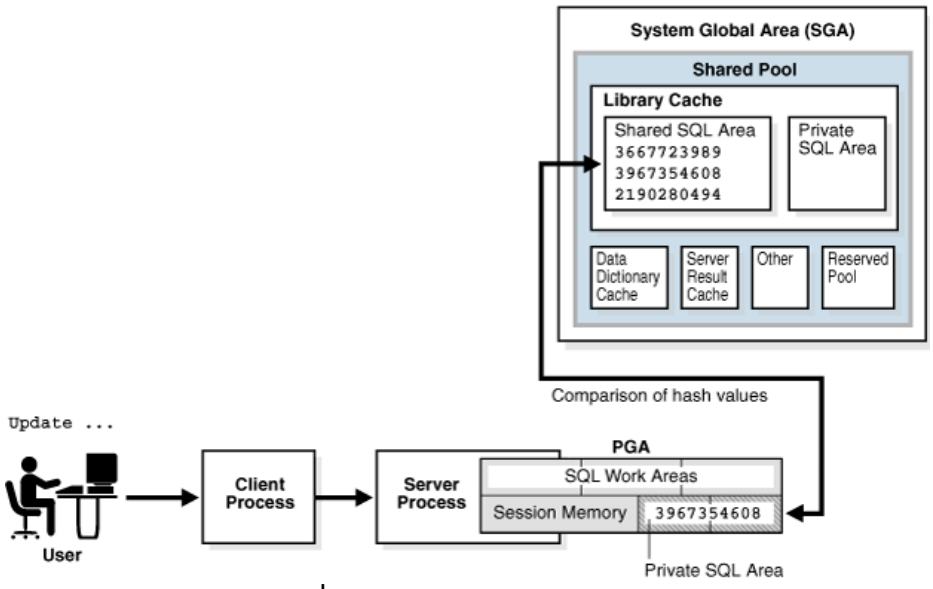
Parsing เป็นระยะแรกของกระบวนการ ซึ่งจะทำการแบ่งส่วนประกอบของ SQL Statement ให้เป็นโครงสร้างข้อมูลที่กระบวนการในระยะถัดไปจะนำไปใช้งานได้ เมื่อแอปพลิเคชันเรียกใช้ SQL Statement ใด แอปพลิเคชันนั้นจะทำ Parse Call ไปยังระบบฐานข้อมูลเพื่อให้เตรียม Statement นั้นให้พร้อมสำหรับการประมวลผล โดย Parse Call จะทำการสร้าง Cursor ตัวหนึ่งขึ้นมา เพื่อทำหน้าที่จัดการพื้นที่ในส่วนของ Private SQL Area ของแต่ละ Session ในการเก็บ SQL Statement ที่ผ่านการ Parsing แล้ว รวมทั้งข้อมูลสำหรับกระบวนการอื่นด้วย ทั้ง Cursor และ Private SQL Area อยู่ในส่วนของ PGA

ในช่วงของ Parse Call ระบบฐานข้อมูลจะทำการตรวจสอบ 3 ขั้นตอนตามลำดับ ซึ่งจะช่วยให้ตรวจพบข้อผิดพลาดที่มีได้ก่อนที่จะประมวลผล SQL Statement แต่ข้อผิดพลาดบางอย่างก็ไม่อาจตรวจพบได้ในการ Parsing เช่น ข้อผิดพลาดจากการแปลงชนิดข้อมูล เป็นต้น

Syntax Check จะทำการตรวจสอบความถูกต้องของไวยากรณ์ของ SQL Statement

Semantic Check จะทำการตรวจสอบความหมายของ SQL Statement เช่น ตรวจสอบว่าตารางหรือวิวใน Statement นั้นมีอยู่จริงหรือไม่

Shared Pool Check จะทำการตรวจสอบพื้นที่ Shared SQL Area ภายใน Shared Pool ว่ามี Statement ดังกล่าวที่ผ่านการ Parsing แล้วอยู่หรือไม่ โดยการค้นหาตาม SQL ID ซึ่งเป็นค่า Hash ของ Statement ที่คำนวณโดยใช้อัลกอริทึม Hash ซึ่งค่า Hash ของแต่ละ Statement จะแตกต่างกันไปตามที่อยู่บนหน่วยความจำและค่า Hash ของ Execution Plan ของ Statement นั้น โดย SQL Statement หนึ่งสามารถมีได้หลาย Execution Plan ซึ่งแต่ละ Plan จะมีค่า Hash ต่างกัน ภาพที่ 2-8 อธิบายเกี่ยวกับการทำงานในส่วนของ Shared Pool Check




**ภาพที่ 2-8** Shared Pool Check [4]

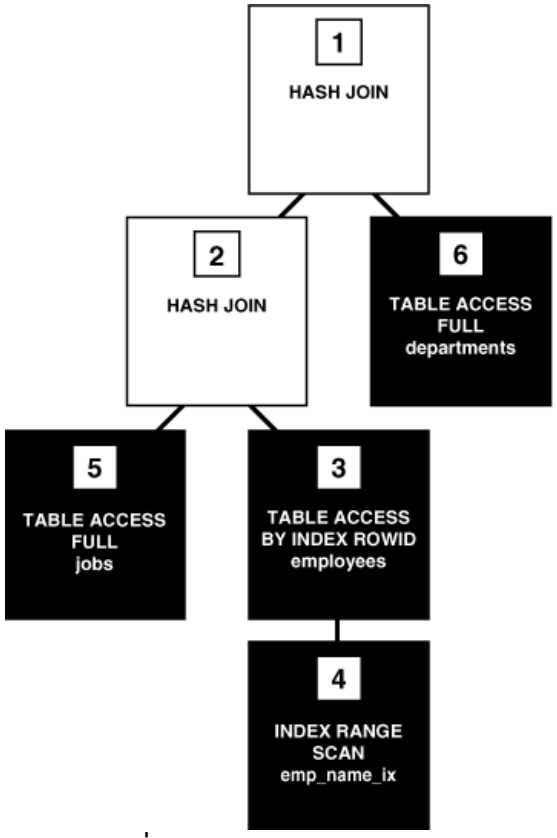
Hard Parse จะดำเนินการเมื่อค้นหา SQL ID ที่ต้องการไม่พบใน Shared SQL Area โดยจะทำการสร้างคำสั่งที่สามารถประมวลผลได้ของ Statement นั้น โดยระหว่างการสร้างจะมีการเข้าถึง Library Cache และ Data Dictionary Cache หลายครั้งเพื่อตรวจสอบ Data Dictionary โดยในการเข้าถึง Cache ทั้งสองส่วนนี้จะมีการวางอุปกรณ์ที่เรียกว่า Latch ไว้บนวัตถุที่ต้องการใช้งาน (เช่น ตารางหรือวิวที่ Statement อ้างอิง) เพื่อไม่ให้นิยามของวัตถุนั้นมีการเปลี่ยนแปลงจนกว่าจะใช้งานวัตถุนั้นเสร็จ

Soft Parse เป็นการ Parsing ที่ไม่ใช่ Hard Parse ซึ่งจะต่างกันไปตามการทำงาน เช่น การกำหนดค่าต่างๆ ให้กับ Session ใน Shared SQL Area โดยทั่วไป Soft Parse เป็นที่ต้องการมากกว่า Hard Parse เนื่องจากข้ามขั้นตอน Optimization แล้วประมวลผลได้ทันที

SQL Optimization เป็นกระบวนการในการคัดเลือกแผนที่มีประสิทธิภาพสูงสุดสำหรับการประมวลผล SQL Statement ดังที่กล่าวมาแล้วข้างต้น

SQL Row Source Generation ทำหน้าที่รับ Query Plan มาสร้างเป็น Iterative Plan ซึ่งเป็นโปรแกรมไบนารีที่ SQL Virtual Machine จะใช้ในการประมวลผล และสร้างชุดของผลลัพธ์ที่เรียกว่า Result Set

Query Plan นั้นจะประกอบไปด้วยขั้นตอนการทำงานหลายขั้นตอน แต่ละขั้นตอนจะให้ผลลัพธ์เป็นชุดของแถว (Row Set) ที่จะถูกใช้ในขั้นตอนถัดไป Row Source คือ ชุดของแถวที่เป็นผลลัพธ์ของขั้นตอนใดๆ และมีโครงสร้างการควบคุมสำหรับประมวลผลแถวของข้อมูลเหล่านั้นซ้ำได้ Row Source อาจจะเป็นตาราง วิว หรือผลลัพธ์ของการ Join หรือการจัดกลุ่มก็ได้ โดยที่ Row Source Generator จะสร้าง Row Source Tree ซึ่งเป็นกลุ่มของ Row Source ที่แสดงถึงข้อมูลต่างๆ ได้แก่ ลำดับของตารางที่ถูกอ้างถึงใน Statement วิธีการเข้าถึงตารางเหล่านั้น วิธีการเชื่อมโยงข้อมูลระหว่างตารางเหล่านั้น และการกระทำต่อข้อมูล เช่น การกรอง การเรียงลำดับ หรือการสรุปข้อมูล ตัวอย่างของ Row Source Tree แสดงในภาพที่ 2-9




**ภาพที่ 2-9** Row Source Tree [4]

SQL Execution จะทำการประมวลผลทีละ Row Source ใน Row Source Tree การประมวลผลนี้ทำโดย SQL Engine ในระหว่างการประมวลผล ระบบฐานข้อมูลจะทำการอ่านข้อมูลจากดิสก์ถ้าไม่พบในหน่วยความจำ โดยอาจจะต้องมีการใช้กลไก Lock และ Latch เพื่อให้แน่ใจใน

ความถูกต้องตรงกันของข้อมูล (Data Integrity) และขั้นตอนสุดท้ายเมื่อการทำงานทุกอย่างเสร็จสิ้น คือทำการปิดการใช้งาน Cursor

## 2.4 การวัดประสิทธิภาพการสืบค้น

ระบบฐานข้อมูล Oracle จะใช้ค่าสถิติต่างๆ ที่ระบบฐานข้อมูลเก็บรวมรวมไว้เพื่อใช้ในการประเมินประสิทธิภาพการทำงานของระบบฐานข้อมูล ซึ่งมีอยู่เป็นจำนวนมาก

การวัดประสิทธิภาพการสืบค้นจะใช้ข้อมูลและค่าสถิติบางส่วนที่เกี่ยวข้องกับการประมวลผลของแต่ละ Statement โดยตรง ซึ่งได้มาจากการประมวลผล Query Plan ซึ่งค่าที่สำคัญ ได้แก่

Cost เป็นต้นทุนของการใช้ทรัพยากรในการประมวลผล SQL Statement ซึ่งคำนวณมาจากต้นทุนของการใช้งาน CPU (CPU Cost) และต้นทุนในการติดต่อกับดิสก์หรือส่วนแสดงผล (I/O Cost)

%CPU เป็นอัตราส่วนการใช้ CPU ที่ Optimizer ใช้ประมวลผลคำสั่ง

Time เป็นเวลาที่ใช้ในการประมวลผลคำสั่ง มีหน่วยเป็น Millisecond (ms)

การแสดงข้อมูลของ Query Plan ทำได้โดยเรียกใช้คำสั่ง Autotrace ซึ่งสามารถใช้งานได้ทั้งในโปรแกรม SQL Plus ที่ถูกติดตั้งมาพร้อมกับชุดติดตั้ง Oracle Client หรือโปรแกรม SQL Developer ซึ่งเป็นโปรแกรมที่ Oracle ให้ดาวน์โหลดมาติดตั้งใช้งานโดยไม่มีค่าใช้จ่าย

การเรียกใช้คำสั่ง Autotrace แบบ Script หรือเรียกใช้ในโปรแกรม SQL Plus จะได้ผลลัพธ์ในส่วนของ Query Plan ตามภาพที่ 2-10

```
Plan hash value: 3769312669

-----------------------------------------------------------------------------------------------
| Id  | Operation                    | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |               |    37 |  1480 |     4  (25)| 00:00:01 |
|   1 |  SORT ORDER BY               |               |    37 |  1480 |     4  (25)| 00:00:01 |
|*  2 |   FILTER                     |               |       |       |            |          |
|   3 |    TABLE ACCESS FULL         | DEPARTMENT    |   201 |  8040 |     3   (0)| 00:00:01 |
|*  4 |    TABLE ACCESS BY INDEX ROWID| DEPARTMENT    |     1 |     9 |     1   (0)| 00:00:01 |
|*  5 |     INDEX UNIQUE SCAN        | SYS_C0010363  |     1 |       |     0   (0)| 00:00:01 |
```

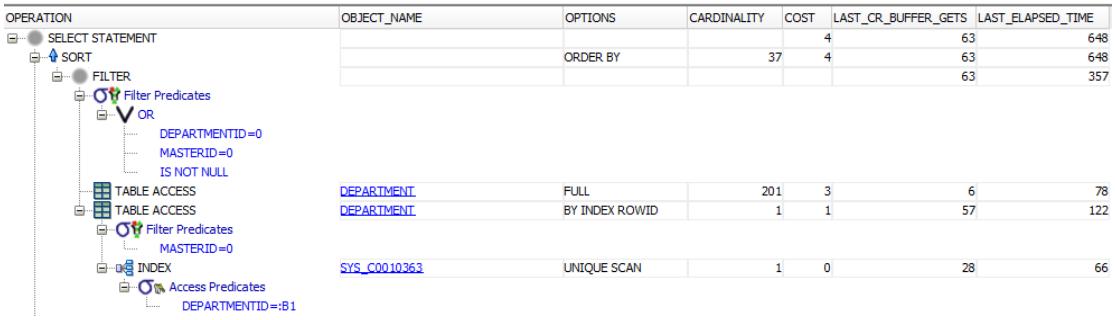
**ภาพที่ 2-10** ตัวอย่าง Query Plan หรือ Explain Plan ใน SQL Plus

ส่วนการแสดงข้อมูลของ Query Plan ด้วยปุ่มคำสั่ง Autotrace ใน SQL Developer ดังภาพที่ 2-11 จะแสดงค่าสถิติที่แตกต่างไปจากการเรียกใช้คำสั่งแบบ Script เล็กน้อย โดยไม่แสดงค่า Bytes %CPU และ Time แต่จะแสดงค่าต่อไปนี้ดังนี้

CARDINALITY คือ จำนวนแถวที่ได้จากการทำ Operation

LAST_CR_BUFFER_GETS คือ จำนวน Buffer ที่ถูกอ่านแบบ Consistent ในระหว่างการประมวลผลครั้งล่าสุด

LAST_ELAPSED_TIME คือ เวลาที่ใช้ไปในการทำ Operation ในระหว่างการประมวลผลครั้งล่าสุด หน่วยเป็น Microsecond



| OPERATION | OBJECT_NAME | OPTIONS | CARDINALITY | COST | LAST_CR_BUFFER_GETS | LAST_ELAPSED_TIME |
|---|---|---|---|---|---|---|
| SELECT STATEMENT | | | | 4 | 63 | 648 |
| SORT | | ORDER BY | 37 | 4 | 63 | 648 |
| FILTER | | | | | 63 | 357 |
| Filter Predicates | | | | | | |
| OR | | | | | | |
| DEPARTMENTID=0 | | | | | | |
| MASTERID=0 | | | | | | |
| IS NOT NULL | | | | | | |
| TABLE ACCESS | DEPARTMENT | FULL | 201 | 3 | 6 | 78 |
| TABLE ACCESS | DEPARTMENT | BY INDEX ROWID | 1 | 1 | 57 | 122 |
| Filter Predicates | | | | | | |
| MASTERID=0 | | | | | | |
| INDEX | SYS_C0010363 | UNIQUE SCAN | 1 | 0 | 28 | 66 |
| Access Predicates | | | | | | |
| DEPARTMENTID=:B1 | | | | | | |

**ภาพที่ 2-11** ตัวอย่าง Query Plan หรือ Explain Plan ใน SQL Developer

## 2.5 ผลงานวิจัยที่เกี่ยวข้อง

จากการค้นคว้าบทความพบว่ามีผู้สนใจศึกษาการใช้งาน Result Cache ในลักษณะต่างๆ ดังนี้

Steven Feuerstein [5] ได้ทำการทดลองใช้ Function Result Cache โดยทดลองเปรียบเทียบกับการประมวลผลฟังก์ชันซ้ำๆ โดยไม่ใช้ Result Cache และเทียบกับการใช้ข้อมูลจาก Cache โดยเก็บข้อมูลในตารางทดสอบไว้ใน Package Collection ซึ่งใช้พื้นที่ใน PGA การทดลองได้ข้อสรุปว่า การใช้ Function Result Cache ใช้เวลาในการประมวลผลน้อยกว่าแบบไม่ใช้ ส่วนการใช้ Cache ที่เป็น Package Collection นั้น ใช้เวลาประมวลผลน้อยที่สุด แต่จะใช้พื้นที่หน่วยความจำในส่วน PGA มากขึ้น เมื่อมี Session มากขึ้น ซึ่งจะเกิดปัญหาตามมาในภายหลัง

Alex Fatkulin [6] เห็นว่า Result Cache ไม่ได้ช่วยปรับปรุงประสิทธิภาพของการสืบค้นข้อมูลในโลกแห่งความจริง เนื่องจากไม่เหมาะสมกับการใช้งานที่มีการเข้าถึงข้อมูลพร้อมกันในระดับสูงและมีอัตราการเข้าถึงที่สูง อีกทั้งไม่สามารถทำการขยายความสามารถของระบบฐานข้อมูล (Scalability) ด้วยการทำ Parallel Processing ได้ โดยผลการทดลองแสดงให้เห็นว่า ยิ่งมีจำนวน Process ที่ทำงานคู่ขนานไปพร้อมกันมากเท่าไหร่ Query ที่ใช้ Server Result Cache ยิ่งทำงานช้าลงมากเท่านั้น

Dean Richards [7] นำเสนอการทดลองใช้ Result Cache เพื่อเก็บผลลัพธ์ของการสืบค้นข้อมูลจากตารางซึ่งไม่ค่อยมีการเปลี่ยนแปลง โดยทดลองใช้ทั้ง Server Result Cache และ Function Result Cache การสืบค้นข้อมูลในการทดลองเป็นการหาผลรวมจำแนกตามกลุ่มของข้อมูลที่ได้ผลลัพธ์ 47 เรคคอร์ดจากข้อมูลทั้งหมด 2.9 ล้านเรคคอร์ด โดยแสดงผลการทดสอบว่า การสืบค้นแบบไม่ใช้ Result Cache ใช้เวลาประมวลผล 00:00:12.04 ในขณะที่การสืบค้นแบบใช้ Result Cache ใช้เวลา 00:00:00.06 ซึ่งเห็นได้ว่า การใช้ Result Cache ช่วยลดเวลาในการสืบค้นข้อมูลลงอย่างมาก

Franck Pachot [8] ได้ทำการทดลองใช้ Function Result Cache โดยทำการทดลอง 3 กรณี กรณีละ 1,000 ครั้ง ได้แก่ กรณีมีผลลัพธ์ใน Result Cache (Cache Hit) กรณีไม่มีผลลัพธ์ใน Result Cache (Cache Miss) และกรณีมีผลลัพธ์ใน Result Cache แต่ผลลัพธ์มีค่าเปลี่ยนแปลง (Cache Invalid) การทดลองแสดงให้เห็นว่า การใช้ Result Cache จะมีประโยชน์เฉพาะกรณี Cache Hit เท่านั้น แต่กรณีเกิด Cache Miss หรือ Cache Invalid ขึ้น จะทำให้ระบบฐานข้อมูลทำงานช้าลงมาก และสรุปว่าการใช้ Result Cache อย่างไม่ถูกต้องจะนำไปสู่ภาวะการขัดแย้งสูง (High contention) เมื่อระบบมีภาระงานมากจากการเข้าใช้งานพร้อมกัน เนื่องจาก Result Cache มีการควบคุมด้วย Latch เพียงตัวเดียว เมื่อเข้าใช้งานพร้อมกันแบบ Exclusive Mode เป็นจำนวนมาก จะเกิดการรอใช้งานได้นานเป็นเวลาหลายนาทีจนถึงหลายชั่วโมงเลยทีเดียว

# บทที่ 3
# วิธีดำเนินงานวิจัย

ในการดำเนินงานวิจัยครั้งนี้จะเป็นการศึกษาเกี่ยวกับผลการใช้งาน Server Result Cache บนระบบฐานข้อมูล Oracle ของระบบบัญชี 3 มิติ เป็นการศึกษาวิจัยเชิงทดลอง โดยผู้วิจัยได้วางแนวทางในการดำเนินการวิจัยเพื่อให้บรรลุตามวัตถุประสงค์ไว้เป็นลำดับดังนี้

3.1 กลุ่มตัวอย่าง

3.2 การเก็บข้อมูลทดลอง

3.3 เครื่องมือที่ใช้ในการวิจัย

3.4 กาวิเคราะห์ข้อมูล

## 3.1 กลุ่มตัวอย่าง

กลุ่มตัวอย่างที่ใช้ในการวิจัยนี้จะคัดเลือกจาก SQL Statement ที่ใช้ในระบบบัญชี 3 มิติ ที่มีการใช้งานสูงสุด 100 อันดับแรก โดยคัดเลือกเฉพาะ Select Statement ที่สืบค้นข้อมูลจากตารางที่มีการเข้าถึงบ่อย 20 อันดับแรก และเป็นตารางข้อมูลอ้างอิงซึ่งมีการเปลี่ยนแปลงข้อมูลน้อย

| | COUNT(*) | OBJECT_NAME |
|---|---|---|
| 1 | 5775 | PREPAYMENTITEM |
| 2 | 5407 | BUDGET |
| 3 | 4907 | DEPARTMENT |
| 4 | 4647 | PRITEM |
| 5 | 4172 | PR |
| 6 | 3813 | RECEIVEITEM |
| 7 | 3796 | PREPAYMENT |
| 8 | 3320 | GL |
| 9 | 3250 | GLHEAD |
| 10 | 3188 | SYSUSER |
| 11 | 2449 | DNITEM |
| 12 | 2150 | RECEIVE |
| 13 | 2031 | ACCOUNT |
| 14 | 1988 | SYSBYTEDES |
| 15 | 1339 | REFER |
| 16 | 1333 | PAYMENTITEM |
| 17 | 1295 | ACCOUNTTYPE |
| 18 | 1238 | BUDGETGROUP |
| 19 | 1188 | ACTIVITY |
| 20 | 1124 | DN |

**ภาพที่ 3-1** ตารางที่มีการเข้าถึงมากที่สุด 20 อันดับแรก

จากภาพที่ 3-1 ตารางข้อมูลอ้างอิง ได้แก่ DEPARTMENT, ACCOUNT, SYSBYTEDES, REFER, ACCOUNTTYPE, BUDGETGROUP, ACTIVITY แต่จะคัดเลือกเฉพาะตาราง DEPARTMENT, ACCOUNT, BUDGETGROUP, SYSBYTEDES เนื่องจากเป็นตารางที่มีการเปลี่ยนแปลงข้อมูลน้อย

ดังนั้น การคัดเลือก Select Statement ที่ใช้ในการทดลอง จะคัดเลือกจากรายการ SQL Statement ที่มีการใช้งานสูงสุด 100 อันดับแรก เฉพาะที่เป็น Select Statement ที่สืบค้นข้อมูลจากตาราง DEPARTMENT, ACCOUNT, BUDGETGROUP และเป็น Select Statement แบบที่ไม่มีเงื่อนไขการคัดกรองข้อมูลแบบ Bind Variable เท่านั้น เนื่องจาก Statement ที่ใช้ Bind Variable ไม่สามารถใช้งาน Result Cache ได้

```
SELECT COUNT(*) FROM ACCOUNT
WHERE ACCOUNTID = :B1
AND ACCOUNTLEVEL=2
```

**ภาพที่ 3-2** ตัวอย่าง Select Statement ที่ใช้ Bind Variable

| | EXECUTIONS | PARSE_CALLS | SQL_TEXT |
|---|---|---|---|
| 1 | 109411 | 17170 | SELECT PROGRAM FROM SYS.V_$SESSION WHERE AUDSID = SYS_CONTEXT('USERENV', 'SESSIONID') |
| 2 | 97001 | 21664 | SELECT SYSUSERGROUPID FROM SYSUSER WHERE DBLOGIN = USER |
| 3 | 51096 | 8659 | SELECT PRSTATUS FROM PR WHERE PRID = :B1 |
| 4 | 32283 | 1812 | SELECT COUNT(*) FROM ACCOUNT WHERE ACCOUNTID = :B1 AND ACCOUNTLEVEL=2 |
| 5 | 32268 | 7183 | SELECT COUNT(*) FROM SYSTRANSACTION WHERE SYSUSERGROUPID = :B2 AND TRANSACTIONCODE = 'BUDGET_USE' AND :B1 >= DEPARTMENTIDFROM AND :B1 <= DEPARTMENTID |
| 6 | 27672 | 5349 | SELECT NVL(SUM(NVL(ARITEMPAY.AMOUNT,0)),0) FROM ARITEM, ARITEMPAY WHERE ARITEM.ARID = ARITEMPAY.ARID AND ARITEM.ARITEM = ARITEMPAY.ARITEM AND ARITEMP |
| 7 | 27363 | 4351 | UPDATE PRITEM SET PREPAYMENTBALANCE = :B3 WHERE PRID = :B2 AND PRITEM = :B1 |
| 8 | 23831 | 20256 | SELECT BUDGETGROUPID, BUDGETGROUPID \|\| ' : ' \|\| BUDGETGROUPNAME AS BUDGETGROUPNAME FROM MASTER3D.BUDGETGROUP WHERE BUDGETGROUPSTAIUS = 'Y' ORDER BY B |
| 9 | 23180 | 3848 | SELECT NVL(SUM(NVL(ARITEMPAY.AMOUNT,0)),0) FROM ARITEM, ARITEMPAY WHERE ARITEM.ARID = ARITEMPAY.ARID AND ARITEM.ARITEM = ARITEMPAY.ARITEM AND ARITEMP |
| 10 | 22932 | 6507 | SELECT NVL(SUM(DECODE(PREPAYMENTSTATUS,'S',DR,'C',DR,0)),0) FROM PREPAYMENTITEM WHERE BUDGETID = :B1 AND PREPAYREF IS NULL |
| 11 | 20274 | 20274 | select '00.00.0000 '\|\|banner, user, 99/100 from v$version where banner like 'Oracle%' |
| 12 | 20005 | 4190 | SELECT PI.ARID,PI.ARITEM, NVL(SUM(DECODE(PREPAYMENTSTATUS,'W',DR,'C',DR,'S',DR,'V',0)),0) AS PBALANCE,NVL(SUM(DECODE(PREPAYMENTSTATUS,'W',DR,'C',DR,' |
| 13 | 17857 | 6902 | SELECT SYSUSERGROUPID FROM MASTER3D.SYSUSER WHERE UPPER(DBLOGIN) = USER |
| 14 | 16632 | 7794 | SELECT SYSUSERGROUPID FROM SYSUSER WHERE UPPER(DBLOGIN) = USER |
| 15 | 15748 | 5102 | SELECT ACTIVITYID, ACTIVITYID \|\| ' : ' \|\| ACTIVITYNAME AS ACTIVITYNAME,PROJECTID,DEPARTMENTID FROM MASTER3D.ACTIVITY WHERE (ACTIVITYSTATUS = 'Y') AND |
| 16 | 14090 | 2796 | SELECT ACCOUNTID, ACCOUNTNAME, MASTERID FROM MASTER3D.BANKACCOUNT ORDER BY ACCOUNTID ASC |
| 17 | 14086 | 2794 | SELECT DISTINCT MASTERID, MASTERNAME FROM MASTER3D.BANKACCOUNT ORDER BY MASTERID ASC |
| 18 | 13813 | 2748 | SELECT /*+ RESULT_CAHCE */ DNBOOKCODE, TO_CHAR(DNBOOKCODE) \|\| ': ' \|\| DNBOOKNAME AS DISPLAY FROM MASTER3D.DNBOOK ORDER BY DNBOOKCODE ASC |
| 19 | 13813 | 2748 | SELECT /*+ RESULT_CAHCE */ BUDGETGROUPID, TO_CHAR(BUDGETGROUPID) \|\| ': ' \|\| BUDGETGROUPNAME AS DISPLAY FROM MASTER3D.BUDGETGROUP ORDER BY BUDGETGROUP |
| 20 | 13813 | 2748 | SELECT /*+ RESULT_CAHCE */ PREPAYMENTBOOKCODE, TO_CHAR(PREPAYMENTBOOKCODE) \|\| ': ' \|\| PREPAYMENTBOOKNAME AS DISPLAY FROM MASTER3D.PREPAYMENTBOOK ORDE |
| 21 | 13813 | 2749 | SELECT /*+ RESULT_CAHCE */ BUDGETGROUPID, TO_CHAR(BUDGETGROUPID) \|\| ': ' \|\| BUDGETGROUPNAME AS DISPLAY, SUBSTR(BUDGETGROUPID,1,2) AS BGSOURCE FROM MA |
| 22 | 11647 | 2464 | SELECT DNID,DNITEM,NVL(SUM(DECODE(PREPAYMENTSTATUS,'S',DR,'C',DR,'W',DR,'V',0)),0) AS BALANCE FROM MASTER3D.PREPAYMENTITEM WHERE DNID IS NOT NULL AND |
| 23 | 11040 | 2947 | SELECT NVL(SUM(DECODE(PREPAYMENTSTATUS,'W',DR,'S',DR,'C',DR,0)),0) FROM PREPAYMENTITEM WHERE BUDGETID = :B2 AND PREPAYMENTID <> :B1 AND PREPAYREF IS |
| 24 | 8537 | 6814 | SELECT BUDGETPERIODID, BUDGETGROUPID FROM MASTER3D.BUDGETPERIOD WHERE (BUDGETPERIODSTATUS = 'Y') ORDER BY BUDGETPERIODID DESC |
| 25 | 8381 | 3288 | SELECT COUNT(*) FROM MASTER3D.SYSTRANSACTION WHERE SYSUSERGROUPID = :B2 AND TRANSACTIONCODE = 'BUDGET_USE' AND :B1 >= DEPARTMENTIDFROM AND :B1 <= DEP |

**ภาพที่ 3-3** ตัวอย่าง SQL Statement ที่มีการใช้งานสูงสุด 25 อันดับแรก

จากภาพที่ 3-3 จะเห็นได้ว่า SQL Statement ที่มีการใช้งานสูงสุด ส่วนใหญ่จะเป็น Select Statement ที่ใช้ Bind Variable และเป็น Statement ประเภท Update และ Insert การคัดเลือก Select Statement จากรายการ SQL Statement ที่มีการใช้งานสูงสุด 100 อันดับแรก จึงคัดเลือกได้มา 6 Statement กำหนดชื่อให้เป็น S1 ถึง S6 สำหรับใช้ในการทดลองแบบใช้และไม่ใช้ Server Result Cache จึงกำหนดชื่อ NC ต่อท้ายชื่อ Statement แบบที่ไม่ใช้ Result Cache และกำหนดชื่อ C ต่อท้ายชื่อ Statement แบบที่ใช้ Result Cache โดย Statement ที่ใช้ในการทดลองมีดังต่อไปนี้

S1-NC: สืบค้นข้อมูลจากตาราง BUDGETGROUP จำนวน 2 คอลัมน์ โดยไม่ใช้ Result Cache

```
SELECT BUDGETGROUPID, BUDGETGROUPID || ' : ' || BUDGETGROUPNAME AS BUDGETGROUPNAME
FROM MASTER3D.BUDGETGROUP
WHERE BUDGETGROUPSTATUS = 'Y'
ORDER BY BUDGETGROUPID;
```

S1-C: สืบค้นข้อมูลจากตาราง BUDGETGROUP จำนวน 2 คอลัมน์ โดยใช้ Result Cache

```
SELECT /*+ RESULT_CAHCE */ BUDGETGROUPID, BUDGETGROUPID || ' : ' || BUDGETGROUPNAME AS BUDGETGROUPNAME
FROM MASTER3D.BUDGETGROUP
WHERE BUDGETGROUPSTATUS = 'Y'
ORDER BY BUDGETGROUPID;
```

S2-NC: สืบค้นข้อมูลจากตาราง BUDGETGROUP จำนวน 3 คอลัมน์ โดยไม่ใช้ Result Cache

```
SELECT BUDGETGROUPID, TO_CHAR(BUDGETGROUPID) || ': ' || BUDGETGROUPNAME AS DISPLAY,
    SUBSTR(BUDGETGROUPID,1,2) AS BGSOURCE
FROM MASTER3D.BUDGETGROUP
ORDER BY BUDGETGROUPID ASC
```

S2-C: สืบค้นข้อมูลจากตาราง BUDGETGROUP จำนวน 3 คอลัมน์ โดยใช้ Result Cache

```
SELECT /*+ RESULT_CACHE */ BUDGETGROUPID, TO_CHAR(BUDGETGROUPID) || ': ' || BUDGETGROUPNAME AS DISPLAY,
    SUBSTR(BUDGETGROUPID,1,2) AS BGSOURCE
FROM MASTER3D.BUDGETGROUP
ORDER BY BUDGETGROUPID ASC
```

S3-NC: สืบค้นข้อมูลจากตาราง DEPARTMENT โดยไม่ใช้ Result Cache

```
SELECT DEPARTMENTID, TO_CHAR(DEPARTMENTID) || ': ' || DEPARTMENTNAME AS DISPLAY
FROM MASTER3D.DEPARTMENT
WHERE (DEPARTMENTID = 0 OR MASTERID = 0
OR (MASTERID IN (SELECT DEPARTMENTID
                 FROM MASTER3D.DEPARTMENT
                 WHERE MASTERID = 0)))
                 ORDER BY DEPARTMENTID ASC
```

S3-C: สืบค้นข้อมูลจากตาราง DEPARTMENT โดยใช้ Result Cache

```
SELECT /*+ RESULT_CACHE */ DEPARTMENTID, TO_CHAR(DEPARTMENTID) || ': ' || DEPARTMENTNAME AS DISPLAY
FROM MASTER3D.DEPARTMENT
WHERE (DEPARTMENTID = 0 OR MASTERID = 0
OR (MASTERID IN (SELECT DEPARTMENTID
                 FROM MASTER3D.DEPARTMENT
                 WHERE MASTERID = 0)))
                 ORDER BY DEPARTMENTID ASC
```

S4-NC: สืบค้นข้อมูลจากตาราง ACCOUNT โดยไม่ใช้ Result Cache

```
SELECT ACCOUNTID||' : '||ACCOUNTNAME AS ACCOUNTNAME, ACCOUNTID
FROM MASTER3D.ACCOUNT
WHERE ACCOUNTSTATUS = 'Y' AND ACCOUNTLEVEL = '2' ORDER BY ACCOUNTID
```

S4-C: สืบค้นข้อมูลจากตาราง ACCOUNT โดยใช้ Result Cache

```
SELECT /*+ RESULT_CACHE */ ACCOUNTID||' : '||ACCOUNTNAME AS ACCOUNTNAME, ACCOUNTID
FROM MASTER3D.ACCOUNT
WHERE ACCOUNTSTATUS = 'Y' AND ACCOUNTLEVEL = '2' ORDER BY ACCOUNTID
```

S5-NC: สืบค้นข้อมูลจากวิว BANKACCOUNT ที่อ้างอิงตาราง ACCOUNT โดยไม่ใช้ Result Cache

```
SELECT ACCOUNTID, ACCOUNTNAME, MASTERID
FROM MASTER3D.BANKACCOUNT
ORDER BY ACCOUNTID ASC
```

S5-C: สืบค้นข้อมูลจากวิว BANKACCOUNT ที่อ้างอิงตาราง ACCOUNT โดยใช้ Result Cache

```
SELECT /*+ RESULT_CACHE */ ACCOUNTID, ACCOUNTNAME, MASTERID
FROM MASTER3D.BANKACCOUNT
ORDER BY ACCOUNTID ASC
```

S6-NC: สืบค้นข้อมูลจากตาราง SYSBYTEDES โดยไม่ใช้ Result Cache

```
SELECT BYTECODE,BYTEDESENG AS BYTEDES
FROM MASTER3D.SYSBYTEDES
WHERE TABLENAME='ALL' AND COLUMNNAME='STATUS';
```

S6-C: สืบค้นข้อมูลจากตาราง SYSBYTEDES โดยใช้ Result Cache

```
SELECT /*+ RESULT_CACHE */ BYTECODE,BYTEDESENG AS BYTEDES
FROM MASTER3D.SYSBYTEDES
WHERE TABLENAME='ALL' AND COLUMNNAME='STATUS';
```

### 3.2 การเก็บข้อมูลทดลอง

สำหรับการเก็บข้อมูลทดลองในงานวิจัยนี้จะเป็นการเก็บข้อมูลสถิติการประมวลผล Select Statement บนระบบฐานข้อมูลระบบบัญชี 3 มิติที่ใช้งานอยู่จริง โดยจะทำการประมวลผลในช่วงที่ไม่มีผู้ใช้งานระบบ เพื่อป้องกันปัจจัยอื่นที่เกี่ยวข้องกับการใช้งานหน่วยความจำซึ่งกระทบต่อการทดลอง และเพื่อไม่ให้ส่งผลกระทบต่อการใช้งานของผู้ใช้งานระบบบัญชี 3 มิติ อีกทั้งงานวิจัยนี้ไม่ได้มุ่งเน้นการวัดประสิทธิภาพการทำงานของระบบฐานข้อมูลในสภาวะที่มีผู้ใช้จำนวนมากกำลังใช้งานระบบพร้อมกัน (Overload) จึงไม่ได้ทำการทดลองในสถานการณ์เช่นนั้น

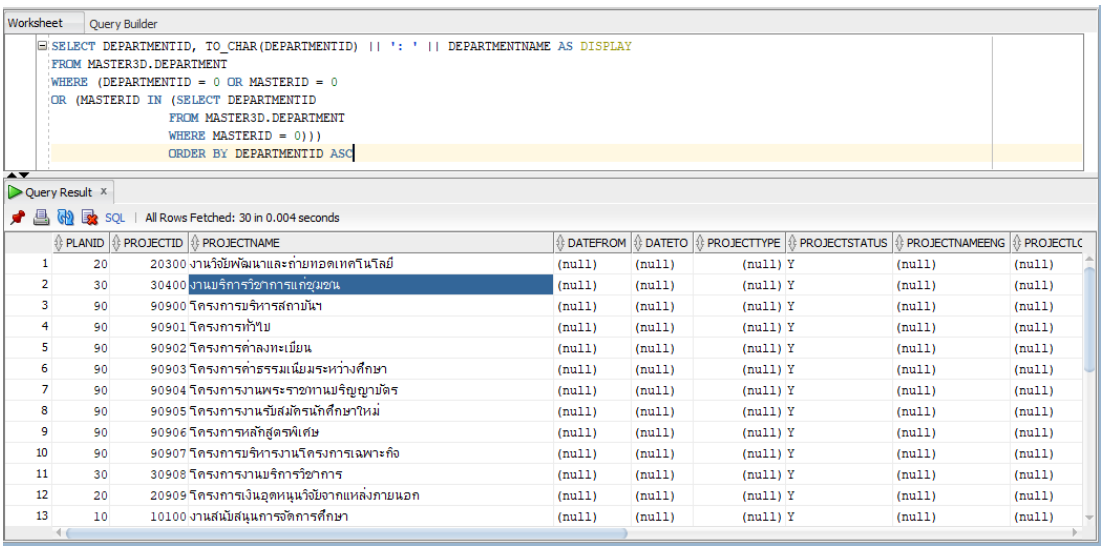
การทดลองจะดำเนินการโดยประมวลผล Select Statement แบบที่ใช้และไม่ใช้ Server Result Cache อย่างละ 10 ครั้ง ต่อเนื่องกัน โดยจดบันทึกค่าสถิติจากการประมวลผลในแต่ละครั้ง โดยจะเริ่มจากการประมวลผล S1-NC ซึ่งไม่ใช้ Result Cache ก่อน เมื่อประมวลผลและเก็บค่าสถิติครบ 10 ครั้งแล้ว จึงประมวลผล S1-C ซึ่งใช้ Result Cache เป็นลำดับต่อมา ไล่เรียงในลักษณะนี้ตามลำดับจนถึง S6-C

ผู้วิจัยคาดว่าผลลัพธ์ของการทดลองจะเป็น ดังนี้

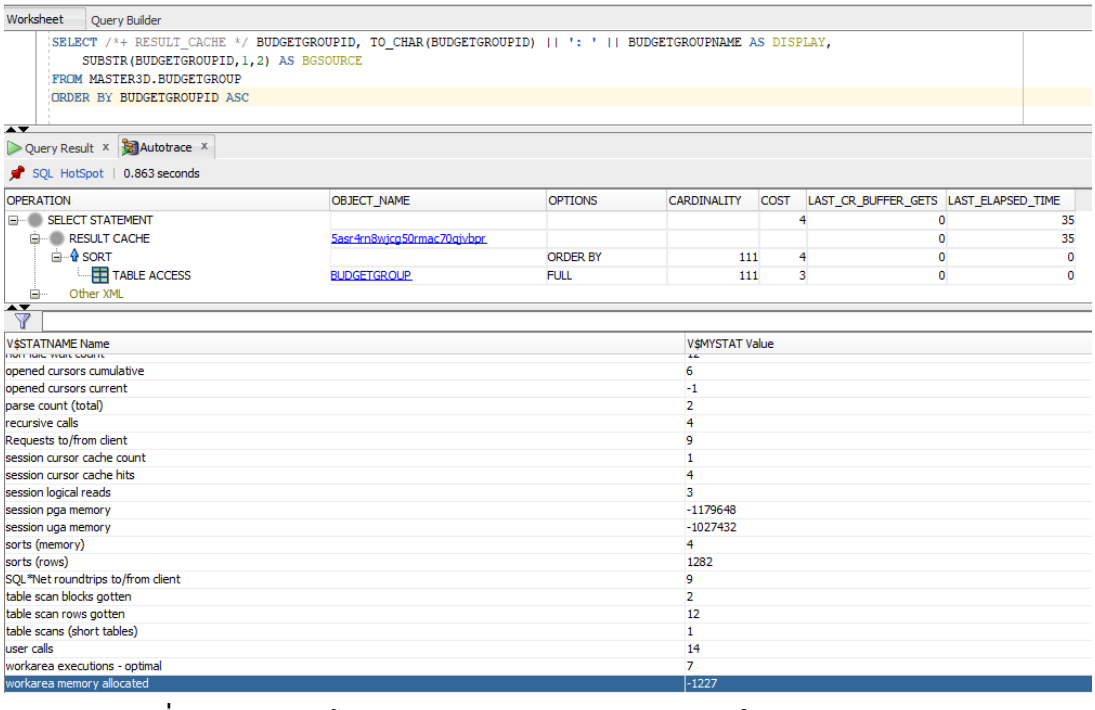
1. เวลาในการประมวลผล Select Statement แบบไม่ใช้ Result Cache แต่ละครั้งจะไม่ต่างกันมาก เนื่องจากขั้นตอนการประมวลผลไม่ต่างกัน และไม่ต้องทำการ Parsing เพราะมีอยู่ใน Library Cache อยู่แล้ว สามารถประมวลผลได้ทันที
2. เวลาในการประมวลผล Select Statement แบบใช้ Result Cache ในครั้งแรกจะมากกว่าในครั้งถัดไป เนื่องจากการประมวลครั้งแรกต้องทำขั้นตอน Parsing ก่อน
3. เวลาในการประมวลผลครั้งถัดไปจากครั้งแรกของ Select Statement แบบที่ใช้ Result Cache จะใช้เวลาน้อยกว่าการประมวลผลแบบที่ไม่ใช้ Result Cache เนื่องจากใช้ผลลัพธ์ที่มีอยู่ใน Result Cache จึงไม่ต้องประมวลผลใหม่

## 3.3 เครื่องมือที่ใช้ในการวิจัย

เครื่องมือที่ใช้ในการวิจัยนี้จะใช้โปรแกรม Oracle SQL Developer ซึ่งเป็นเครื่องมือสำหรับการบริหารจัดการฐานข้อมูล Oracle ใช้สำหรับเรียกใช้คำสั่ง Select Statement และแสดงค่าสถิติของการประมวลผล โดยการสั่งให้ประมวลผล Select Statement ทำในส่วนของ Worksheet ดังตัวอย่างหน้าจอโปรแกรมตามภาพที่ 3-3 และการสั่งให้แสดงค่าสถิติทำโดยใช้คำสั่ง Autotrace ตามตัวอย่างหน้าจอในภาพที่ 3-4 ตามลำดับ




**ภาพที่ 3-4** ตัวอย่างโปรแกรม Oracle SQL Developer ในส่วนของ Worksheet




**ภาพที่ 3-5** ตัวอย่างโปรแกรม Oracle SQL Developer ในส่วนของ Autotrace

### 3.4 การวิเคราะห์ข้อมูล

สำหรับการทดลองในหัวข้อวิจัยนี้จะเน้นในเรื่องประสิทธิภาพการสืบค้นข้อมูล โดยวัดจากค่าสถิติที่ได้จาก Query Plan ซึ่งมีรายการดังต่อไปนี้

| | |
|---|---|
| Cost | ต้นทุนของการใช้ทรัพยากรสำหรับ Operation ในการประมวลผล เป็นค่าที่คำนวณจากสัดส่วนของเวลาที่ CPU และ I/O ใช้ในการประมวลผล จึงเป็นค่าการวัดที่ไม่มีหน่วย |
| LAST_ELAPSED_TIME | เวลาที่ใช้ในการประมวลผล มีหน่วยเป็น Microsecond |
| LAST_CR_BUFFER_GETS | เป็นจำนวน Buffer ที่ถูกอ่านแบบ Consistent มีหน่วยเป็นจำนวน Data Block ซึ่งเป็นหน่วยการเก็บข้อมูลของ Oracle |

เมื่อเก็บรวมรวมค่าสถิติดังกล่าวแล้วจะนำมาเปรียบเทียบค่าที่ได้ในการประมวลผล Select Statement ทั้งแบบที่ใช้และไม่ใช้ Server Result Cache โดยจะใช้ค่าสถิติ LAST_ELAPSED_TIME ในการเปรียบเทียบประสิทธิภาพ

# บทที่ 4
# ผลการทดลอง

การทดลองในงานวิจัยนี้ จะใช้กลุ่มตัวอย่างของ Select Statement ที่มีการใช้งานจริงในระบบบัญชี 3 มิติ มาทดลองทำการประมวลผลทั้งแบบเดิมที่ไม่ใช้ Result Cache และแบบปรับปรุงให้ใช้ Result Cache เพื่อเปรียบเทียบว่า การใช้งาน Statement แบบที่ใช้ Result Cache จะส่งผลต่อประสิทธิภาพการสืบค้นมากเพียงใด

## 4.1 ผลการเก็บข้อมูลจากการทดลอง

การเก็บข้อมูลการทดลองนั้น จะเก็บข้อมูลที่แสดงในหน้าจอซอฟต์แวร์ SQL Developer โดยเก็บข้อมูลสถิติการประมวลผลในส่วนของ Explain Plan ที่เป็นผลลัพธ์จากคำสั่ง Autotrace ค่าสถิติที่ใช้ในการวัดประสิทธิภาพการสืบค้นข้อมูลในงานวิจัยนี้มีด้วยกัน 3 ค่า ค่าแรกคือ Cost ซึ่งเป็นต้นทุนของการใช้ทรัพยากรในการประมวลผล ค่าที่สองคือ LAST_CR_BUFFER_GETS เป็นจำนวน Buffer ที่ถูกอ่านแบบ Consistent และค่าสุดท้ายคือ LAST_ELAPSED_TIME ซึ่งเป็นเวลาที่ใช้ไปในระหว่างการประมวลผล ผลการทดลองด้านเวลาที่ใช้ในการประมวลผล Statement ดังตารางที่ 4-1

**ตารางที่ 4-1** เวลาที่ใช้ในการประมวลผล Statement S1-S6 จำนวน 10 ครั้ง

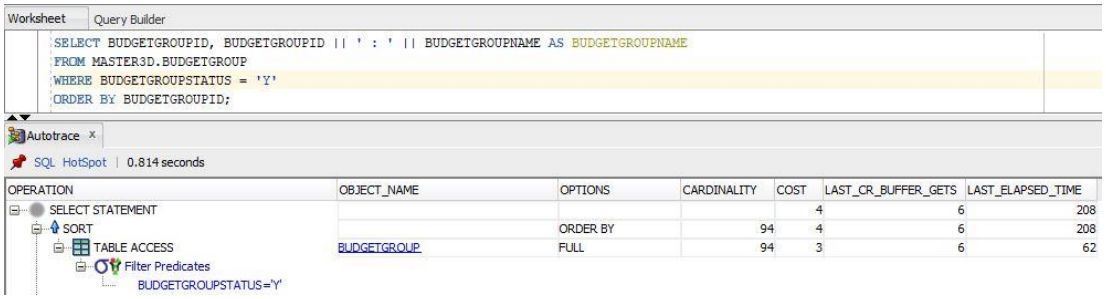
| ครั้งที่ | LAST_ELAPSED_TIME (Microsecond) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | S1-NC | S1-C | S2-NC | S2-C | S3-NC | S3-C | S4-NC | S4-C | S5-NC | S5-C | S6-NC | S6-C |
| 1 | 205 | 594 | 384 | 718 | 630 | 1406 | 2574 | 2813 | 1873 | 2000 | 31 | 114 |
| 2 | 220 | 38 | 322 | 26 | 658 | 37 | 2601 | 2807 | 1828 | 2009 | 120 | 15 |
| 3 | 212 | 35 | 333 | 35 | 646 | 63 | 2626 | 2811 | 1770 | 2005 | 64 | 20 |
| 4 | 211 | 41 | 287 | 36 | 651 | 50 | 2606 | 2804 | 1942 | 2003 | 59 | 20 |
| 5 | 209 | 36 | 255 | 33 | 666 | 47 | 2604 | 2805 | 1767 | 2019 | 60 | 21 |
| 6 | 227 | 62 | 285 | 31 | 651 | 41 | 2590 | 2802 | 1778 | 1974 | 73 | 19 |
| 7 | 205 | 35 | 306 | 33 | 648 | 48 | 2592 | 2809 | 1777 | 2102 | 70 | 17 |
| 8 | 213 | 39 | 151 | 40 | 660 | 42 | 2600 | 2804 | 1801 | 1994 | 72 | 21 |
| 9 | 210 | 37 | 301 | 33 | 641 | 44 | 2620 | 2809 | 1799 | 1986 | 73 | 21 |
| 10 | 209 | 38 | 280 | 35 | 627 | 54 | 2594 | 2800 | 1786 | 2005 | 69 | 16 |
| เฉลี่ย | 212.1 | 95.5 | 290.4 | 102 | 647.8 | 183.2 | 2600.7 | 2806.4 | 1812.1 | 2009.7 | 69.1 | 28.4 |

จากตารางที่ 4-1 จะเห็นว่าเวลาในการประมวลผล Statement ในแต่ละครั้งไม่แตกต่างกันมาก ยกเว้นการประมวลผล Statement แบบใช้ Server Result Cache ครั้งแรกจะใช้เวลามากกว่าครั้งถัดไปอย่างมาก เนื่องจากระบบต้องทำการประมวลผล Statement ซึ่งต้องไปอ่านข้อมูลจาก Database Buffer Cache แล้วมาทำการประมวลผลข้อมูลจนได้ผลลัพธ์ จากนั้นทำการจองใช้พื้นที่หน่วยความจำในส่วนของ Server Result Cache และทำการเขียนผลลัพธ์บน Server Result Cache ตามที่ได้รับจัดสรร ส่วนการประมวลผลครั้งที่สองถึงครั้งที่สิบนั้นใช้เวลาน้อยกว่า เนื่องจากไม่ต้องทำการอ่านและประมวลผลข้อมูล รวมทั้งไม่ต้องเขียนผลลัพธ์บน Server Result Cache แต่จะเข้าไปตรวจสอบว่ามีผลลัพธ์บน Server Result Cache หรือไม่ เมื่อตรวจพบจึงอ่านข้อมูลผลลัพธ์มาใช้ได้เลย ซึ่งจะเห็นได้ว่าการประมวลผลในครั้งแรกนั้นยังไม่ได้ใช้ประโยชน์จาก Server Result Cache ทั้งนี้ เพื่อให้เห็นความแตกต่างอย่างชัดเจนระหว่างการใช้และไม่ใช้ Server Result Cache จึงนำค่าสถิติของการประมวลผลเฉพาะครั้งถัดไปคือครั้งที่สองถึงครั้งที่สิบมาคำนวณค่าเฉลี่ย แล้วนำมาเปรียบเทียบกับค่าสถิติในการประมวลผลครั้งแรก ดังแสดงในตารางที่ 4-2

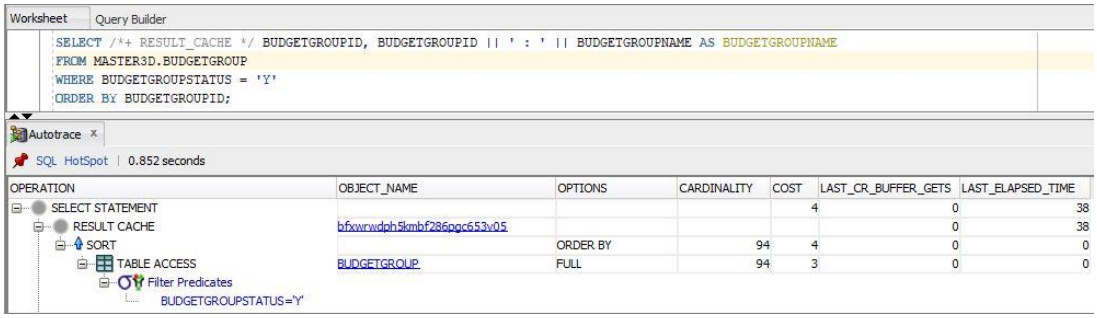
**ตารางที่ 4-2** สถิติจากการประมวลผล Statement ครั้งที่ 1 และค่าสถิติเฉลี่ยของครั้งถัดไป

| Statement | Cost | | Last Cr buffer Gets (Blocks) | | Last Elapsed Time (Microsecond) | | | |
|---|---|---|---|---|---|---|---|---|
| ครั้งที่ | #1 | #Next | #1 | #Next | #1 | #Next | Diff | %Diff |
| S1-NC | 4 | 4 | 6 | 6 | 205 | 212.9 | -7.9 | -3.85% |
| S1-C | 4 | 4 | 6 | 0 | 594 | 40.1 | 553.9 | 93.25% |
| S2-NC | 4 | 4 | 6 | 6 | 384 | 280.0 | 104.0 | 27.08% |
| S2-C | 4 | 4 | 6 | 0 | 718 | 33.6 | 684.4 | 95.32% |
| S3-NC | 4 | 4 | 63 | 63 | 630 | 649.8 | -19.8 | -3.14% |
| S3-C | 4 | 4 | 62 | 0 | 1406 | 47.3 | 1358.7 | 96.64% |
| s4-NC | 16 | 16 | 60 | 60 | 2574 | 2603.7 | -29.7 | -1.15% |
| s4-C | 16 | 16 | 60 | 60 | 2813 | 2805.7 | 7.3 | 0.26% |
| s5-NC | 16 | 16 | 69 | 69 | 1873 | 1805.3 | 67.7 | 3.61% |
| s5-C | 16 | 16 | 69 | 69 | 2000 | 2010.8 | -10.8 | -0.54% |
| S6-NC | 3 | 3 | 3 | 6 | 31 | 73.3 | -42.3 | -136.45% |
| S6-C | 3 | 3 | 6 | 0 | 114 | 18.9 | 95.1 | 83.42% |

จากตารางที่ 4-2 จะเห็นว่า ค่า Cost ของการประมวลผลแต่ละ Statement ทั้งแบบใช้และไม่ใช้ Result Cache ไม่แตกต่างกัน เนื่องจากระบบฐานข้อมูลใช้แผนการสืบค้น (Query Plan) เดียวกันในการประมวลผลทั้งสองแบบ ดังจะเห็นได้จากภาพที่ 4-1 แผนการสืบค้นของ Statement S1 แบบไม่ใช้ Result Cache และภาพที่ 4-2 แผนการสืบค้นของ Statement S1 แบบใช้ Result Cache ที่แสดงให้เห็นวิธีการเข้าถึงข้อมูลและจัดเรียงข้อมูลที่เหมือนกัน ยกเว้น Statement S4 และ S5 ที่ค่า Cost ทั้งสองแบบไม่แตกต่างกัน เนื่องจากไม่สามารถเก็บผลลัพธ์ใน Result Cache ได้

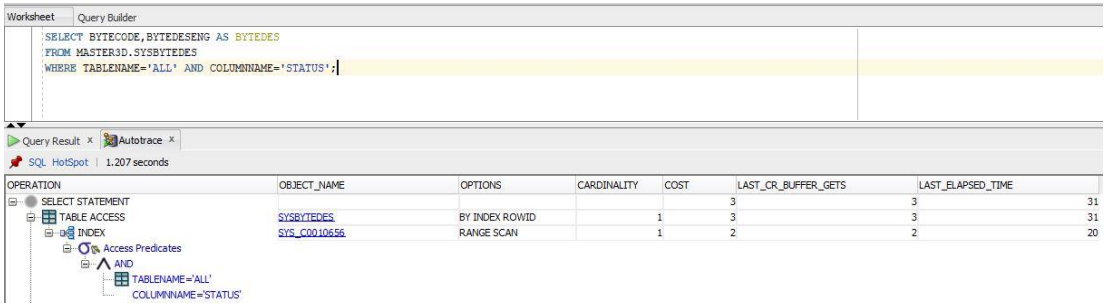



**ภาพที่ 4-1** แผนการสืบค้นของ Statement S1 แบบไม่ใช้ Result Cache

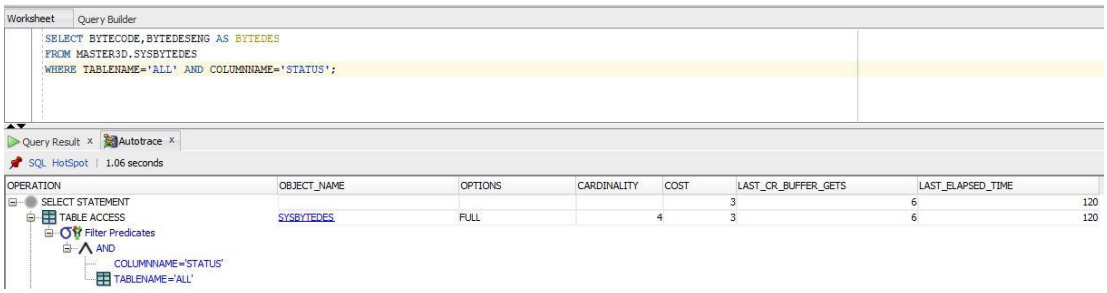



**ภาพที่ 4-2** แผนการสืบค้นของ Statement S1 แบบใช้ Result Cache

ค่า LAST_CR_BUFFER_GETS ของ Statement แบบไม่ใช้ Result Cache ครั้งแรกและครั้งถัดไปมีค่าไม่ต่างกัน เพราะอ่านข้อมูลจาก Buffer Cache จำนวนเท่ากัน ส่วน LAST_ELAPSED_TIME มีค่าต่างกันโดยเฉลี่ยประมาณร้อยละ 7.8 ยกเว้น Statement S6 แบบไม่ใช้ Result Cache ที่แผนการสืบค้นในครั้งถัดไปต่างจากแผนการสืบค้นในครั้งแรก ทำให้ค่า LAST_CR_BUFFER_GETS และ LAST_ELAPSED_TIME ในครั้งแรกต่างจากในครั้งถัดไปอย่างมาก ดังภาพที่ 4-3 และภาพที่ 4-4




**ภาพที่ 4-3** แผนการสืบค้นของ Statement S6 แบบไม่ใช้ Result Cache ครั้งที่ 1




**ภาพที่ 4-4** แผนการสืบค้นของ Statement S6 แบบไม่ใช้ Result Cache ครั้งที่ 2

ค่า LAST_CR_BUFFER_GETS และค่า LAST_ELAPSED_TIME ของ Statement แบบที่ใช้ Result Cache ครั้งแรกและครั้งถัดไปมีความแตกต่างกันมาก ค่า LAST_CR_BUFFER_GETS ของครั้งถัดไปเป็น 0 แสดงว่าไม่มีการอ่านข้อมูลจาก Buffer Cache สอดคล้องกับค่า LAST_ELAPSED_TIME ที่ระยะเวลาในการประมวลผลครั้งถัดไปน้อยกว่าครั้งแรกโดยเฉลี่ยประมาณร้อยละ 92 โดยยกเว้น Statement S4 และ S5 ที่อ่านค่าจาก Result Cache ไม่ได้ นอกจากนี้ในการประมวลผลครั้งแรก ค่า LAST_ELAPSED_TIME ของ Statement แบบที่ใช้ Result Cache มีค่ามากกว่าแบบที่ไม่ใช้ Result Cache ประมาณ 1-2 เท่า เนื่องจากมีกระบวนการเขียนผลลัพธ์ใส่ใน Result Cache เพิ่มขึ้น

สำหรับ S4 และ S5 ค่า LAST_CR_BUFFER_GETS และค่า LAST_ELAPSED_TIME ครั้งแรกและครั้งถัดไปไม่แตกต่างกัน เพราะการประมวลผลไม่ได้มีการอ่านค่าจาก Result Cache เมื่อไปตรวจสอบข้อมูลที่ถูกจัดเก็บอยู่ใน Result Cache พบว่า สถานะของผลลัพธ์ของ Statement S4 และ S5 มีค่าเป็น Bypass ส่วน Statement อื่นมีสถานะเป็น Published โดยสถานะ Bypass หมายถึงพื้นที่ Result Cache ของ Statement นั้นยังไม่พร้อมใช้งาน และสถานะ Published หมายถึงพื้นที่ Result Cache ของ Statement นั้นพร้อมใช้งานอยู่ จึงแปลความได้ว่า การเขียนผลลัพธ์ของ Statement S4 และ S5 ในครั้งแรกบนพื้นที่ Result Cache ยังไม่เสร็จสิ้น ยังไม่สามารถใช้งานได้ ดังนั้น การประมวลผล Statement เหล่านี้ในครั้งถัดไปทันทีจึงไม่ได้ใช้ประโยชน์จาก Result Cache

เมื่อนำข้อมูลในตารางที่ 4-1 เฉพาะส่วนของ Statement S1-S3 และ S6 ซึ่งได้ใช้ประโยชน์จาก Result Cache หรือเรียกว่ากรณี Cache Hit มาคำนวณหาร้อยละของระยะเวลาที่ลดลงในการประมวลผล Statement แบบใช้ Result Cacheเทียบกับแบบไม่ใช้ รายละเอียดดังตารางที่ 4-3

**ตารางที่ 4-3** ร้อยละของเวลาที่ลดลงในการประมวลผลแบบใช้ Result Cache กรณี Cache Hit

| ครั้งที่ | LAST_ELAPSED_TIME (microsecond) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | S1-NC | S1-C | ลดลง (%) | S2-NC | S2-C | ลดลง (%) | S3-NC | S3-C | ลดลง (%) | S6-NC | S6-C | ลดลง (%) |
| 1 | 205 | 594 | - | 384 | 718 | - | 630 | 1406 | - | 31 | 114 | - |
| 2 | 220 | 38 | 82.73 | 322 | 26 | 91.93 | 658 | 37 | 94.38 | 120 | 15 | 87.50 |
| 3 | 212 | 35 | 83.49 | 333 | 35 | 89.49 | 646 | 63 | 90.25 | 64 | 20 | 68.75 |
| 4 | 211 | 41 | 80.57 | 287 | 36 | 87.46 | 651 | 50 | 92.32 | 59 | 20 | 66.10 |
| 5 | 209 | 36 | 82.78 | 255 | 33 | 87.06 | 666 | 47 | 92.94 | 60 | 21 | 65.00 |
| 6 | 227 | 62 | 72.69 | 285 | 31 | 89.12 | 651 | 41 | 93.70 | 73 | 19 | 73.97 |
| 7 | 205 | 35 | 82.93 | 306 | 33 | 89.22 | 648 | 48 | 92.59 | 70 | 17 | 75.71 |
| 8 | 213 | 39 | 81.69 | 151 | 40 | 73.51 | 660 | 42 | 93.64 | 72 | 21 | 70.83 |
| 9 | 210 | 37 | 82.38 | 301 | 33 | 89.04 | 641 | 44 | 93.14 | 73 | 21 | 71.23 |
| 10 | 209 | 38 | 81.82 | 280 | 35 | 87.50 | 627 | 54 | 91.39 | 69 | 16 | 76.81 |
| เฉลี่ย | 212.10 | 95.50 | 81.23 | 290.40 | 102.00 | 87.63 | 647.80 | 183.20 | 92.87 | 69.10 | 28.40 | 69.10 |

จากตารางที่ 4-3 จะเห็นได้ว่าค่าเฉลี่ยของร้อยละของเวลาที่ลดลงในการประมวลผลแบบใช้ Result Cache ของ Statement ทั้งสี่เมื่อเทียบกับแบบไม่ใช้จะมีค่า 81.23 87.63 92.87 และ 69.10 ตามลำดับ ซึ่งคำนวณค่าเฉลี่ยจากค่าทั้งสี่จะได้ค่าเป็น 82.71 อาจกล่าวโดยสรุปได้ว่าการประมวลผลแบบใช้ Result Cache ในกรณี Cache Hit ใช้เวลาลดลงโดยเฉลี่ยร้อยละ 82.71 เมื่อเทียบกับแบบไม่ใช้ Result Cache

เมื่อนำข้อมูลในตารางที่ 4-1 เฉพาะส่วนของ Statement S4 และ S5 ซึ่งไม่ได้ใช้ประโยชน์จาก Result Cache หรือที่เรียกว่ากรณี Cache Miss นั้นมาคำนวณหาร้อยละของระยะเวลาที่ลดลงหรือเพิ่มขึ้นในการประมวลผลแบบใช้ Result Cache เทียบกับแบบไม่ใช้ รายละเอียดดังตารางที่ 4-4

**ตารางที่ 4-4** ร้อยละของเวลาที่ลดลงในการประมวลผลแบบใช้ Result Cache กรณี Cache Miss

| ครั้งที่ | LAST_ELAPSED_TIME (microsecond) | | | | | |
|---|---|---|---|---|---|---|
| | S4-NC | S4-C | เพิ่มขึ้น (%) | S5-NC | S5-C | เพิ่มขึ้น (%) |
| 1 | 2574 | 2813 | 9.29% | 1873 | 2000 | 6.78% |
| 2 | 2601 | 2807 | 7.92% | 1828 | 2009 | 9.90% |
| 3 | 2626 | 2811 | 7.04% | 1770 | 2005 | 13.28% |
| 4 | 2606 | 2804 | 7.60% | 1942 | 2003 | 3.14% |
| 5 | 2604 | 2805 | 7.72% | 1767 | 2019 | 14.26% |
| 6 | 2590 | 2802 | 8.19% | 1778 | 1974 | 11.02% |
| 7 | 2592 | 2809 | 8.37% | 1777 | 2102 | 18.29% |
| 8 | 2600 | 2804 | 7.85% | 1801 | 1994 | 10.72% |
| 9 | 2620 | 2809 | 7.21% | 1799 | 1986 | 10.39% |
| 10 | 2594 | 2800 | 7.94% | 1786 | 2005 | 12.26% |
| เฉลี่ย | 2600.7 | 2806.4 | 7.91% | 1812.1 | 2009.7 | 11.00% |

จากตารางที่ 4-4 จะเห็นได้ว่าระยะเวลาในการประมวลผล Statement แบบใช้ Result Cache กลับเพิ่มขึ้นเมื่อเทียบกับการประมวลผล Statement แบบไม่ใช้ Result Cache ซึ่งระยะเวลาเพิ่มขึ้นโดยเฉลี่ยร้อยละ 7.91 และร้อยละ 11 สำหรับ Statement S4 และ S5 ตามลำดับ เมื่อคำนวนค่าเฉลี่ยรวมจะได้ค่าเป็น 9.46 อาจกล่าวโดยสรุปได้ว่าการประมวลผลแบบใช้ Result Cache ในกรณี Cache Miss ใช้เวลาเพิ่มขึ้นโดยเฉลี่ยร้อยละ 9.46 เมื่อเทียบกับแบบไม่ใช้ Result Cache

ข้อมูลบน Result Cache ถูกจัดเก็บเป็นบล็อก (Block) หนึ่งบล็อกมีขนาดหนึ่งกิโลไบต์ (Kbyte) ขนาดสูงสุดของ Result Cache ในระบบฐานข้อมูลนี้เท่ากับ 26240 กิโลไบต์หรือ 26240 บล็อก โดยผลลัพธ์ของการสืบค้นหนึ่งชุดมีขนาดได้สูงสุด 1312 กิโลไบต์หรือ 1312 บล็อก ดังนั้น จะมีจำนวนชุดข้อมูลที่มีขนาดสูงสุดได้ 20 ชุด รายละเอียดตามภาพที่ 4-5

```
R e s u l t   C a c h e   M e m o r y   R e p o r t
[Parameters]
Block Size          = 1K bytes
Maximum Cache Size  = 26240K bytes (26240 blocks)
Maximum Result Size = 1312K bytes (1312 blocks)
[Memory]
Total Memory = 5352 bytes [0.000% of the Shared Pool]

... Fixed Memory = 5352 bytes [0.000% of the Shared Pool]
... Dynamic Memory = 0 bytes [0.000% of the Shared Pool]


PL/SQL procedure successfully completed.
```

**ภาพที่ 4-5** ขนาดพื้นที่ของ Sever Result Cache

ผู้วิจัยได้ทำการตรวจสอบข้อมูลที่เก็บบน Result Cache เพื่อหาสาเหตุที่ Statement S4-C และ S5-C ไม่อยู่ในสถานะพร้อมใช้งานทันทีหลังการประมวลผลเช่นเดียวกับ Statement อื่น โดยได้จดบันทึกข้อมูลที่ได้จากการตรวจสอบ Result Cache ของแต่ละ Statement ดังรายละเอียดในตารางที่ 4-5

**ตารางที่ 4-5** ข้อมูลของ Statement บน Result Cache

| Statement | จำนวนแถว (1) | จำนวนบล็อก | แถวใหญ่สุด (Bytes) | แถวเล็กสุด (Bytes) | ขนาดแถวเฉลี่ย (Bytes) (2) | ขนาดข้อมูล (Bytes) (3) = (1) x (2) |
|---|---|---|---|---|---|---|
| S1-C | 94 | 7 | 112 | 27 | 64 | 6016 |
| S2-C | 111 | 8 | 115 | 30 | 65 | 7215 |
| S3-C | 202 | 10 | 78 | 23 | 45 | 9090 |
| S4-C | 1712 | 118 | 163 | 22 | 68 | 116416 |
| S5-C | 548 | 47 | 157 | 36 | 85 | 46580 |
| S6-C | 4 | 1 | 13 | 11 | 11 | 44 |

จากการสังเกตพบว่า ผลลัพธ์ของ Statement S4 และ S5 มีขนาดใหญ่กว่า Statement อื่น จนไม่สามารถเขียนลงบน Result Cache ได้ครบถ้วนในทันที จึงมีสถานะเป็น Bypass อยู่จนกว่าจะเขียนสำเร็จ ซึ่งบางครั้งใช้เวลานานหลายชั่วโมง ผู้วิจัยจึงทำการทดลองเพิ่มเติม โดยการลดขนาดผลลัพธ์ของ Statement S4 และ S5 ให้เล็กลงทีละน้อย เพื่อหาว่าข้อมูลขนาดใหญ่ที่สุดขนาดเท่าใดที่สามารถถูกจัดเก็บบน Result Cache ได้ครบถ้วนในทันทีหลังการประมวลผล ในการทดลองได้ทำการประมวลผล Statement S4 และ S5 ที่มีขนาดผลลัพธ์แตกต่างกันอย่างละ 6 ขนาด แล้วจดบันทึกข้อมูลผลลัพธ์แต่ละขนาดที่เก็บบน Result Cache ดังแสดงในตารางที่ 4-6 ผลการทดลองประมวลผล Statement ทั้งสองมีความแตกต่างกัน ผลลัพธ์ของ S4 ที่มีขนาดไม่เกิน 33000 ไบต์หรือ 33 กิโลไบต์จะถูกเขียนบน Result Cache ได้ภายใน 10 วินาที ส่วน S5 ผลลัพธ์ที่มีขนาดไม่เกิน

44000 ไบต์หรือ 43 กิโลไบต์จะถูกเขียนบน Result Cache ได้ภายใน 10 วินาที แสดงว่านอกจากปัจจัยด้านขนาดของข้อมูลแล้วยังคงมีปัจจัยอื่นที่ส่งผลต่อการเขียนข้อมูลบน Result Cache อีก

**ตารางที่ 4-6** ข้อมูลของ Statement S4 และ S5 ที่มีขนาดต่างกันบน Result Cache

| Statement | จำนวนแถว (1) | จำนวนบล็อก | แถวใหญ่สุด (Bytes) | แถวเล็กสุด (Bytes) | ขนาดแถวเฉลี่ย (Bytes) (2) | ขนาดข้อมูล (Bytes) (3) = (1) x (2) | เวลาที่จัดเก็บบน Cache (วินาที) |
|---|---|---|---|---|---|---|---|
| S4-C-1 | 210 | 14 | 111 | 29 | 63 | 13230 | 10 |
| S4-C-2 | 337 | 20 | 114 | 28 | 57 | 19209 | 10 |
| S4-C-3 | 407 | 27 | 164 | 28 | 65 | 26455 | 10 |
| S4-C-4 | 561 | 33 | 114 | 28 | 59 | 33099 | 10 |
| S4-C-5 | 562 | 36 | 164 | 28 | 64 | 35968 | 840 |
| S4-C-6 | 745 | 61 | 168 | 27 | 82 | 61090 | 1380 |
| S5-C-1 | 251 | 22 | 164 | 63 | 87 | 21837 | 10 |
| S5-C-2 | 443 | 38 | 164 | 63 | 86 | 38098 | 10 |
| S5-C-3 | 491 | 42 | 164 | 61 | 86 | 42226 | 10 |
| S5-C-4 | 499 | 42 | 164 | 61 | 86 | 42914 | 10 |
| S5-C-5 | 501 | 43 | 164 | 61 | 87 | 43587 | 10 |
| S5-C-6 | 503 | 44 | 164 | 43 | 88 | 44264 | 1800 |

หากพิจารณาเฉพาะปัจจัยด้านขนาดของข้อมูลแล้ว จากตารางที่ 4-6 สามารถสรุปได้ว่า Statement ที่มีข้อมูลผลลัพธ์ขนาดไม่เกิน 33 กิโลไบต์จะสามารถถูกจัดเก็บบน Result Cache ได้ภายใน 10 วินาทีหลังจากการประมวลผล ซึ่งจะทำให้มีโอกาสเกิดกรณี Cache Hit ได้สูง เมื่อมีการสืบค้นด้วย Statement เดิมอีกครั้งในระยะเวลาที่ใกล้เคียงกับการประมวลผล Statement นั้นครั้งแรก หรือกล่าวอีกนัยหนึ่งคือช่วยลดโอกาสเกิดกรณี Cache Miss ลง หากเรานำขนาดของผลลัพธ์นี้ไปคำนวณเทียบกับขนาดสูงสุดของ Result Cache ซึ่งมีขนาด 26240 กิโลไบต์ จะได้จำนวนชุดข้อมูลที่สามารถจัดเก็บใน Result Cache ได้ประมาณ 795 ชุด ซึ่งเป็นจำนวนที่น่าจะเพียงพอสำหรับการใช้งาน

# บทที่ 5
# สรุปผล อภิปรายผล และข้อเสนอแนะ

การศึกษาวิจัยเรื่อง “กรณีศึกษาการใช้ Result Cache ปรับปรุงประสิทธิภาพการสืบค้นฐานข้อมูลของระบบบัญชี 3 มิติ มจพ.” ต้องการศึกษาผลการใช้ Result Cache บนฝั่งเซิร์ฟเวอร์ที่มีต่อความเร็วในการสืบค้นข้อมูล จากการทำการทดลองสามารถสรุปได้ดังนี้

## 5.1 สรุปผล

การวิจัยครั้งนี้ต้องการเปรียบเทียบผลการประมวลผลการสืบค้นแบบที่ใช้และไม่ใช้ Server Result Cache เพื่อศึกษาว่า Server Result Cache จะช่วยเพิ่มประสิทธิภาพในการสืบค้นข้อมูลมากเพียงใด เพื่อเป็นแนวทางในการประยุกต์ใช้ Server Result Cache ในระบบบัญชี 3 มิติต่อไป

ผู้วิจัยใช้วิธีทดลองทำการสืบค้นข้อมูลแบบไม่ใช้ Server Result Cache เพื่อเปรียบเทียบการสืบค้นข้อมูลโดยใช้ Server Result Cache โดยทดลองกับกลุ่มตัวอย่าง Select Statement จำนวน 6 Statement ซึ่งคัดเลือกมาจากกลุ่มของ Select Statement ที่มีการใช้บ่อยที่สุดในระบบฐานข้อมูล 100 อันดับแรก และเป็นการสืบค้นข้อมูลจากตารางที่ถูกเข้าถึงบ่อยมากที่สุด 20 อันดับแรก โดยการประมวลผล Select Statement จะทำแบบละ 10 ครั้ง แล้วเก็บรวบรวมข้อมูลสถิติจากการประมวลผลการสืบค้นทั้งหมดมาใช้ในการเปรียบเทียบเพื่อประเมินว่า การสืบค้นข้อมูลโดยใช้ Server Result Cache นั้นใช้เวลาน้อยลงเพียงใดเมื่อเทียบกับการสืบค้นข้อมูลแบบไม่ใช้ Server Result Cache

จากการทดลองกับ Select Statement ที่คัดเลือกมาได้จำนวน 6 Statement พบว่าการประมวลผล Statement แบบใช้ Result Cache ใช้เวลาลดลงโดยเฉลี่ยร้อยละ 82.71 ขึ้นไป ในกรณี Cache Hit แต่จะใช้เวลาเพิ่มขึ้นโดยเฉลี่ยร้อยละ 9.46 ในกรณี Cache Miss เมื่อเทียบกับ Statement แบบไม่ใช้ Result Cache

อย่างไรก็ดี เมื่อพิจารณาการทดลองที่เกิดกรณี Cache Miss นั้นพบว่า เกิดจากการจัดเก็บข้อมูลบน Result Cache ไม่เสร็จสิ้นทันทีหลังการประมวลผล ผลลัพธ์ของ Statement นั้นจึงมีสถานะไม่พร้อมใช้งาน ผู้วิจัยจึงทำการทดลองเพิ่มเติมเพื่อศึกษาเกี่ยวกับปัจจัยด้านขนาดของข้อมูลที่มีผลต่อการจัดเก็บบน Result Cache โดยจากการทดลองได้ข้อสรุปว่า Statement ที่มีข้อมูลผลลัพธ์ขนาดไม่เกิน 33 กิโลไบต์ จะถูกจัดเก็บบน Result Cache ได้เสร็จสิ้นทันทีหลังการประมวลผล

## 5.2 อภิปรายผล

ผลการวิจัยเรื่อง “กรณีศึกษาการใช้ Result Cache ปรับปรุงประสิทธิภาพการสืบค้นฐานข้อมูลของระบบบัญชี 3 มิติ มจพ.” จากค่าความแตกต่างของเวลาที่ใช้ในการสืบค้นข้อมูลแบบใช้ Server Result Cache ระหว่างครั้งแรกและครั้งถัดไป ในกรณีที่มีผลลัพธ์ของการสืบค้นที่พร้อมใช้งานอยู่บน Server Result Cache ซึ่งเรียกว่า Cache Hit นั้น มีระยะเวลาที่ลดลงโดยเฉลี่ยร้อยละ 82.71 ซึ่งเป็นระยะเวลาที่ลดลงอย่างมีนัยสำคัญ แต่ในกรณีที่ไม่มีผลลัพธ์ของการสืบค้นอยู่บน Server Result Cache ซึ่งเรียกว่า Cache Miss นั้น จะใช้เวลาเพิ่มขึ้นโดยเฉลี่ยร้อยละ 9.46 เมื่อเทียบกับแบบไม่ใช้ Result Cache เนื่องจากมีกระบวนการจัดเก็บผลลัพธ์บน Server Result Cache เพิ่มขึ้น ซึ่งเป็นตัวเลขที่ไม่สูงมาก แต่หากมี Statement จำนวนมากที่ถูกประมวลผลพร้อมกันและเกิดกรณี Cache Miss อาจจะส่งผลกระทบต่อประสิทธิภาพโดยรวมของระบบฐานข้อมูลได้

ขนาดของข้อมูลที่เหมาะสมกับการใช้ Server Result Cache ควรมีขนาดไม่เกิน 33 กิโลไบต์ เพื่อลดการเกิดกรณี Cache Miss ซึ่งเมื่อเทียบกับขนาดสูงสุดของ Result Cache ซึ่งมีขนาด 26240 กิโลไบต์ จะมีจำนวนชุดข้อมูลที่สามารถจัดเก็บใน Server Result Cache ได้ประมาณ 795 ชุด ซึ่งเป็นจำนวนที่คาดว่าเพียงพอสำหรับการใช้งาน

จากผลการวิจัยดังกล่าว ผู้วิจัยมีความเห็นว่า การใช้ Server Result Cache จะช่วยเพิ่มประสิทธิภาพของการสืบค้นข้อมูลในระบบบัญชี 3 มิติได้เป็นอย่างดี เมื่อมีการใช้กับขนาดของข้อมูลที่เหมาะสม โดยจะช่วยลดระยะเวลาและทรัพยากรที่ใช้ในการสืบค้นลง

## 5.3 ข้อเสนอแนะจากผลการศึกษา

จากการศึกษาวิจัยกรณีศึกษาการใช้ result cache ปรับปรุงประสิทธิภาพการสืบค้นฐานข้อมูลของระบบบัญชี 3 มิติ มจพ. โดยมีข้อเสนอแนะดังนี้

5.3.1 เพิ่มการทดสอบการสืบค้นที่มีการเข้าถึงข้อมูลจำนวนมากแต่มีผลลัพธ์จำนวนน้อย

5.3.2 เพิ่มการทดสอบการสืบค้นโดยใช้ Server Result Cache ในสภาวะที่มีผู้ใช้งานเข้าใช้พร้อมกันจำนวนมาก

## 5.4 ข้อเสนอแนะเพื่อการศึกษาวิจัยครั้งต่อไป

ในการศึกษาวิจัยครั้งต่อไปควรศึกษาปัจจัยอื่นที่มีผลกระทบต่อการสืบค้นข้อมูล

# เอกสารอ้างอิง


1. Lance Ashdown and Tom Kyte. “Oracle Database Architecture.” Oracle Database Concept 11g Release 2 (11.2). (May 2015). *[online]*. Available from : URL : https://docs.oracle.com/cd/E11882_01/server.112/e40540/intro.htm.
2. Lance Ashdown and Tom Kyte. “Oracle Database Instance.” Oracle Database Concept 11g Release 2 (11.2). (May 2015). *[online]*. Available from : URL : https://docs.oracle.com/cd/E11882_01/server.112/e40540/startup.htm.
3. Lance Ashdown and Tom Kyte. “Memory Architecture.” Oracle Database Concept 11g Release 2 (11.2). (May 2015). *[online]*. Available from : URL : https://docs.oracle.com/cd/E11882_01/server.112/e40540/memory.htm.
4. Lance Ashdown and Tom Kyte. “SQL.” Oracle Database Concept 11g Release 2 (11.2). (May 2015). *[online]*. Available from : URL : https://docs.oracle.com/cd/E11882_01/server.112/e40540/sqllangu.htm.
5. Steven Feuerstein. “On the PL/SQL Function Result Cache.” Oracle Magazine. (October 2007). *[online]*. Available from : URL : https://blogs.oracle.com/oraclemagazine/on-the-plsql-function-result-cache.
6. Alex Fatkulin. “Oracle 11G Result Cache In The Real World.” Pythain: Technical Track. (May 2008). *[online]*. Available from : URL : https://blog.pythian.com/oracle-11g-result-cache-in-the-real-world.
7. Dean Richards. “1000% Performance Improvement Using Oracle 11g Result Cache.” LogicalRead. (December 2012). *[online]*. Available from : URL : https://logicalread.com/2012/12/20/huge-performance-gains-using-oracle-11g-result-cache-dr01.
8. Franck Pachot. “Result Cache: when *not* to use it.” Dbi services: Blog. (January 29, 2018). *[online]*. Available from : URL : https://blog.dbi-services.com/result-cache-when-not-to-use-it/.

# ประวัติผู้วิจัย

ชื่อ : นางณิชาพร ชื่อสุธรรม

ชื่องานวิจัย : กรณีศึกษาการใช้ result cache ปรับปรุงประสิทธิภาพการสืบค้นฐานข้อมูลของระบบบัญชี 3 มิติ มจพ.

สังกัด : สำนักคอมพิวเตอร์และเทคโนโลยีสารสนเทศ

ประวัติ

ปัจจุบันอายุ 45 ปี โดยเกิดและเติบโตที่กรุงเทพมหานคร ปัจจุบันภูมิลำเนาอยู่ที่จังหวัดนนทบุรี

เคยศึกษาระดับมัธยมที่โรงเรียนสตรีนนทบุรี เมื่อสำเร็จการศึกษาในปีกาศึกษา 2535 และได้เข้าศึกษาต่อระดับปริญญาตรีที่มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือจนสำเร็จการศึกษาในปีการศึกษา 2540 ต่อมาได้เข้าศึกษาระดับปริญญาโทที่มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือในปีการศึกษา 2542 จนสำเร็จการศึกษาในปีการศึกษา 2546

หลังจากเรียบจบระดับปริญญาตรีได้เข้าทำงานในตำแหน่งนักวิชาการคอมพิวเตอร์ที่สำนักคอมพิวเตอร์และเทคโนโลยีสารสนเทศจนถึงปัจจุบัน