RMUTP Research Journal, Vol. 8, No. 1, March 2014

การใช้แผนปฏิบัติการเพื่อปรับคำสั่งเอสคิวแอล
Using Execution Plan for SQL Tuning

ชาญชัย ศุภอรรถกร[1]*

[1]ผู้ช่วยศาสตราจารย์ ภาควิชาคณิตศาสตร์สถิติและคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยอุบลราชธานี
จังหวัดอุบลราชธานี 34190

บทคัดย่อ

ระบบฐานข้อมูลเป็นการเก็บรวบรวมสารสนเทศให้สามารถถูกเข้าถึง จัดการ และแก้ไขได้ง่าย เนื่องจากการขยายตัวอย่างรวดเร็วของอินเทอร์เน็ต ทำให้ปริมาณข้อมูลในฐานข้อมูลเพิ่มขึ้นอย่างรวดเร็ว ปริมาณข้อมูลจำนวนมาก ความซับซ้อนของข้อมูล และความต้องการการตอบสนองที่รวดเร็วกลายเป็นปัญหาที่สำคัญในการเข้าถึงฐานข้อมูล เทคนิคหนึ่งที่จะเพิ่มประสิทธิภาพการสอบถาม คือ การปรับคำสั่งเอสคิวแอล ซึ่งเอสคิวแอลที่ได้รับการปรับแต่งแล้วจะทำให้เข้าถึงฐานข้อมูลได้เร็วขึ้น และใช้ทรัพยากรของเครื่องน้อยกว่า ในบทความจะแนะนำเครื่องมือที่มีชื่อว่า แผนปฏิบัติการเพื่อใช้ปรับคำสั่งเอสคิวแอล นอกจากนั้น ยังได้แสดงตัวอย่างการใช้งานของแผนปฏิบัติการเพื่อตรวจสอบการประมวลผลของคำสั่งเอสคิวแอล

Abstract

Database system is a collection of information that can be easily accessed, managed, and updated. With the rapid expansion of the Internet, database overload is increasing rapidly. Large amount of data, complexity of information and needs for quick responses had caused critical problems of database access. One of the techniques to optimize effectiveness of query is to use SQL tuning. The improved SQL could speed up database access and save the resources of computer. This article introduces a tool called 'execution plan for SQL tuning'. In addition, an example is also provided to check the processing of SQL command.

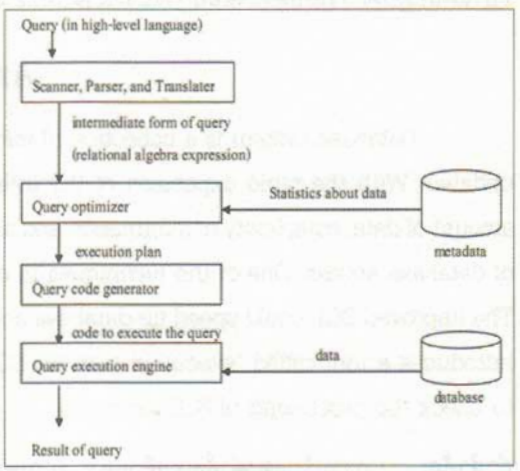
คำสำคัญ : การปรับแต่งคำสั่งเอสคิวแอล การเพิ่มประสิทธิภาพการสอบถามข้อมูล แผนปฏิบัติการ
Keywords : SQL Tuning, Query Optimization, Execution Plan

* *ผู้นิพนธ์ประสานงานไปรษณีย์อิเล็กทรอนิกส์ scchansu@ubu.ac.th โทร. 08 1853 0650*

1. บทนำ

ระบบฐานข้อมูล (Database System) คือ ระบบที่รวบรวมข้อมูลต่าง ๆ ที่เกี่ยวข้องสัมพันธ์กันเข้าด้วยกันในคอมพิวเตอร์ มีการจัดกลุ่มข้อมูลในรูปของตาราง และเชื่อมโยงตารางทั้งหมดเข้าด้วยกัน เพื่อลดความซ้ำซ้อนของข้อมูล ซึ่งผู้ใช้สามารถจัดการกับข้อมูลได้ในลักษณะต่าง ๆ ทั้งการเพิ่ม ลบ แก้ไข และเรียกดูข้อมูล

ปัจจุบันมีการขยายตัวอย่างรวดเร็วของอินเทอร์เน็ต ปริมาณข้อมูลมีการเพิ่มขึ้นในอัตราแบบเอ็กซ์โพเนเชียล (Exponential Rate) ธุรกิจต่าง ๆ จึงต้องสร้างฐานข้อมูลขนาดใหญ่เพื่อรับมือกับอัตราการเพิ่มขึ้นอันมหาศาลของข้อมูล (J Skarie et al., 2007) นอกจากนั้น แนวโน้มการใช้งานฐานข้อมูลมีความซับซ้อนมากยิ่งขึ้น เช่น การนำฐานข้อมูลไปใช้ในงานทางด้านพาณิชย์อิเล็กทรอนิกส์ (Electronic Commerce) เนื่องจากมีจำนวนผู้ใช้งานพร้อม ๆ กันจำนวนมาก เป็นผลให้ประสิทธิภาพของระบบฐานข้อมูลมีความสำคัญต่อความสำเร็จของธุรกิจ (Oracle Corporation, 2011) ดังนั้น ปัญหาที่สำคัญของการใช้งานฐานข้อมูล คือ การเข้าถึงฐานข้อมูลที่มีปริมาณมาก มีความซับซ้อน และต้องการการตอบสนองที่รวดเร็ว (Dandan Li et al., 2010) เช่น ระบบการประมวลผลธุรกรรมออนไลน์ (Online Transaction Processing Systems) การบริหารทรัพยากรองค์กร (Enterprise Resource Planning) การบริหารลูกค้าสัมพันธ์ (Customer Relationship Management) การประมวลผลวิเคราะห์ออนไลน์ (On-line Analytical Processing) และการจัดทำคลังข้อมูลเพื่อการวิเคราะห์ข้อมูล (Data Analysis over Data-warehouses) การสอบถามข้อมูล (Query) เหล่านี้มีปริมาณงานมาก มีความซับซ้อน และฐานข้อมูลมีขนาดใหญ่ (Surajit, 2009) การเพิ่มประสิทธิภาพการสอบถามข้อมูลเป็นกระบวนการที่มีความสำคัญในการดำเนินการของฐานข้อมูลเชิงสัมพันธ์ (Relational Database) โดยเฉพาะอย่างยิ่งการดำเนินการจัดการคำสั่งเอสคิวแอลที่มีความซับซ้อน (Oracle Corporation, 2005) เพื่อให้การสอบถามได้อย่างรวดเร็วในระบบฐานข้อมูลหน้าที่ในการประมวลผลเพื่อแสดงผลลัพธ์จากการสอบถามข้อมูล คือ หน้าที่ของส่วนประมวลผลการสอบถามข้อมูล (Query Processor) โครงสร้างและขั้นตอนการทำงานของส่วนประมวลผลการสอบถามข้อมูลเป็น ดังรูปที่ 1



รูปที่ 1 โครงสร้างและขั้นตอนการทำงานของ Query Processor (นิตยา เกิดประสพ, 2553)

จากรูปที่ 1 โครงสร้างของส่วนประมวลผลการสอบถามข้อมูลประกอบไปด้วย 4 ส่วน คือ

Scanner, Parser and Translator ทำหน้าที่รับคำสั่งเอสคิวแอล แล้วนำไปแปลความหมาย และตรวจสอบความถูกต้อง

Query Optimizer ทำหน้าที่ปรับคำสั่งเอสคิวแอล ให้มีรูปแบบเหมาะสมยิ่งขึ้นกับการ

ประมวลผล ในส่วนนี้จะมีการดึงข้อมูลทางสถิติและสร้างเป็นแผนปฏิบัติการ (Execution Plan) ให้แก่ผู้ใช้สำหรับการปรับปรุงประสิทธิภาพในการสอบถามข้อมูล

Query Code Generator ทำหน้าที่เปลี่ยนคำสั่งเอสคิวแอล ที่ปรับปรุงแล้วให้อยู่ในรูปแบบคำสั่งที่พร้อมจะประมวลผล

Query Execution Engine ทำหน้าที่ประมวลผลและแสดงผลลัพธ์ โดยมีการดึงข้อมูลจากฐานข้อมูลออกมา

จากโครงสร้างดังกล่าวนี้ จะเห็นว่าในส่วนของ Query Optimizer ผู้ใช้งานสามารถทำการปรับคำสั่งเอสคิวแอลให้มีประสิทธิภาพมากที่สุด โดยอาศัยแผนปฏิบัติการ (Execution Plan) และข้อมูลทางสถิติที่ระบบจัดการฐานข้อมูลแสดงออกมาให้ได้

Structured Query Language (SQL) เป็นเครื่องมือที่มีประสิทธิภาพสำหรับใช้ในการสอบถามข้อมูล (Query) ในฐานข้อมูลเชิงสัมพันธ์ (Relational Database) (J. Fan et al., 2011; M. Negri et al., 1991) และได้ถูกนำมาใช้ในเชิงพาณิชย์ต่าง ๆ เอสคิวแอลเป็นมาตรฐานที่ได้รับการยอมรับจากบริษัทผู้ผลิตระบบจัดการฐานข้อมูล (Database Management System – DBMS) และเป็นมาตรฐานอย่างเป็นทางการโดย ANSI และ ISO (M. Negri et al., 1991) คำสั่งเอสคิวแอลถูกใช้ในการดึงข้อมูลจากฐานข้อมูล โดยผลลัพธ์เดียวกันที่ดึงข้อมูลออกมาจากฐานข้อมูลสามารถที่จะเขียนเอสคิวแอลที่แตกต่างกันได้ แต่การใช้คำสั่งเอสคิวแอลที่ดีที่สุดจะทำให้มีประสิทธิภาพในการสอบถามข้อมูล (Beginner SQL Tutorial, 2007) ตัวอย่างเช่น การสอบถามข้อมูล "ให้แสดงรายละเอียดของนักเรียนที่ได้เกรด A ในภาคเรียนล่าสุด" ซึ่งจะสามารถเขียนคำสั่งเอสคิวแอลได้ 3 วิธี โดยได้ผลลัพธ์เหมือนกัน (Burleson Consulting, 2007) ดังนี้

A standard join:

```
SELECT *
FROM STUDENT, REGISTRATION
WHERE
STUDEBT.student_id=REGISTRATION.
student_id
AND
    REGISTRATION.grade = 'A';
```

A nested query:

```
SELECT *
FROM STUDENT
WHERE
    student_id =
    (SELECT student_id
        FROM REGISTRATION
        WHEREgrade = 'A' );
```

A correlated subquery:

```
SELECT *
FROM STUDENT
WHERE
    0 <
    (SELECT count(*)
        FROM REGISTRATION
        WHERE grade = 'A' AND
    student_id = STUDENT.student_id);
```

กระบวนการในการปรับคำสั่งเอสคิวแอลเป็นงานที่ท้าทายของผู้ดูแลระบบฐานข้อมูล (Oracle Corporation, 2003; P. Belknap, 2009) เนื่องจาก 1) ต้องอาศัยความเชี่ยวชาญในการเขียนคำสั่งเอสคิวแอล 2) เป็นกระบวนการที่ใช้เวลานาน เพราะคำสั่งเอสคิวแอลมีมาตรฐานที่ต้องเขียนให้ถูกต้องตามหลักไวยากรณ์ 3) ต้องอาศัยความรู้ทางด้านโครงสร้างฐานข้อมูล และ 4) เป็นงานที่ต้องทำอย่างต่อเนื่อง เช่น เมื่อมีการเปลี่ยนแปลงโครงสร้างของฐานข้อมูลก็ต้องมีการปรับคำสั่งเอสคิวแอลใหม่ (P. Belknap, 2009) สำหรับบทความนี้จะแนะนำการใช้เครื่องมือที่มีชื่อว่า แผนปฏิบัติการ (Execution Plan) เพื่อเป็นข้อมูลประกอบการพิจารณาในการเขียนเอสคิวแอลสำหรับปรับคำสั่งการสอบถามให้มีประสิทธิภาพทั้งในด้านระยะเวลาในการประมวลผล และการใช้ทรัพยากรต่าง ๆ ของเครื่องคอมพิวเตอร์ โดยบทความนี้จะเลือกใช้ระบบจัดการฐานข้อมูล Oracle Database 11g เนื้อหาต่อจากในหัวข้อนี้จะประกอบไปด้วย หัวข้อการใช้งาน Execution Plan หัวข้อตัวอย่างการใช้งาน Execution Plan และหัวข้อสุดท้าย บทสรุป

2. การใช้งาน Execution Plan

แผนปฏิบัติการ (Execution Plan) คือ การแสดงรายละเอียดของขั้นตอนในการประมวลผลคำสั่งเอสคิวแอล ว่าแต่ละขั้นตอนจะมีการแสดงรายละเอียดในการประมวลผลว่าทำอะไรบ้าง และมีจำนวนแถวที่ประมวลผลไปเท่าไร โดยแสดงผลลำดับขั้นตอนการทำงานตามโครงสร้างข้อมูลแบบต้นไม้ (Tree) (Maria Colgan, 2008)

กฎพื้นฐานของ Execution Plan Tree (Ramsundaram, 2009) มีดังนี้

2.1 ใน Execution Plan Tree จะประกอบไปด้วยโหนดราก (Root Node)

2.2 โหนดพ่อแม่ (Parent Node) สามารถมีโหนดลูก (Child Node) ได้ 1 โหนดหรือมากกว่า และโหนดพ่อแม่จะมีค่ารหัส (ID) น้อยกว่าของโหนดลูก

2.3 โหนดลูกสามารถมีโหนดพ่อแม่ได้เพียงโหนดเดียวเท่านั้น โดยจะเขียนคำสั่งของโหนดลูกให้เยื้องไปทางขวา และในกรณีที่มีโหนดลูกหลายโหนด โหนดลูกเหล่านี้ก็จะเขียนให้อยู่ในระดับเดียวกัน

ดูตัวอย่าง Execution Plan จากตัวอย่างต่อไปนี้

```
SQL> explain plan for
  2  select e.empno, e.ename, d.dname
  3  from emp e, dept d
  4  where e.deptno = d.deptno
  5  and e.deptno = 10;

Explained.

SQL> SELECT * FROM table(dbms_xplan.display(null,null,'basic'));

PLAN_TABLE_OUTPUT
--------------------------------------------
Plan hash value: 568005898

--------------------------------------------
| Id  | Operation                    | Name    |
--------------------------------------------
|   0 | SELECT STATEMENT             |         |
|   1 |  NESTED LOOPS                |         |
|   2 |   TABLE ACCESS BY INDEX ROWID| DEPT    |
|   3 |    INDEX UNIQUE SCAN         | PK_DEPT |
|   4 |   TABLE ACCESS FULL          | EMP     |
--------------------------------------------
```

รูปที่ 2 ตัวอย่าง Execution Plan (Ramsundaram, 2009)

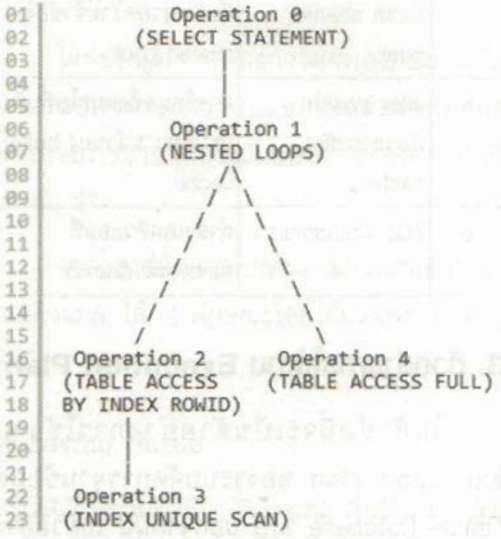
จากตัวอย่างในรูปที่ 2 สามารถอธิบายได้ ดังนี้

Operation 0 เป็นโหนดรากของต้นไม้ มีโหนดลูกเพียงโหนดเดียว คือ Operation 1

Operation 1 มีโหนดลูก 2 โหนด คือ Operation 2 และ 4

Operation 2 มีโหนดลูก 1 โหนด คือ Operation 3

จาก Execution Plan ข้างต้นสามารถแสดงในลักษณะของต้นไม้ได้ ดังนี้



รูปที่ 3 แสดง Execution Plan ในลักษณะของ Tree (Ramsundaram, 2009)

จากรูปที่ 3 ซึ่งเป็นการแสดง Execution Plan ในลักษณะของต้นไม้ สามารถอ่านลำดับขั้นตอนการประมวลผลได้ คือ

Operation 1 จะทำงานได้จะต้องกระทำ Operation 2 และ 4 ก่อน โดยจะกระทำ Operation 2 ได้จะต้องกระทำ Operation 3 ก่อน

Operation 3 เข้าถึงตาราง DEPT โดยใช้วิธี INDEX UNIQUE SCAN และส่งค่า ROWID ไปให้ Operation 2

Operation 2 คืนค่าแถวข้อมูลทั้งหมดจากตาราง DEPT ไปให้ Operation 1

Operation 1 จะกระทำ Operation 4 สำหรับแต่ละแถวที่ Operation 2 คืนค่าให้

Operation 4 จะกระทำการเข้าถึงตาราง EMP แบบ TABLE ACCESS FULL โดยกำหนดเงื่อนไข คือ e.deptno=10 และคืนค่าแถวข้อมูลที่ได้ให้แก่ Operation 1

Operation 1 จะคืนค่าผลลัพธ์สุดท้ายให้แก่ Operation 0

นอกจาก Execution Plan จะแสดงลำดับขั้นตอนการประมวลผลของคำสั่งเอสคิวแอลแล้วยังสามารถแสดงค่าสถิติ (Statistics) การใช้ทรัพยากรต่าง ๆ ของเครื่องคอมพิวเตอร์ด้วย ดังนี้

recursive calls คือ จำนวนครั้งในการเรียกตัวเองซ้ำ

db block gets คือ จำนวนของบล็อกข้อมูลที่ได้รับโดยตรงจากบล็อกบัฟเฟอร์ของ RAM

Consistent gets คือ จำนวนครั้งที่อ่านข้อมูลจากที่มีการร้องขอ และได้รับข้อมูลจากบล็อกข้อมูล (Data Blocks)

Physical reads คือ จำนวนรวมของบล็อกข้อมูล (Data Blocks) ที่ถูกอ่านจากดิสก์ (Disk)

redo size คือ จำนวนรวมของการทำซ้ำ (ไบต์)

bytes sent via SQL*Net to client คือ จำนวนไบต์ที่ส่งไปยังเครื่องลูกข่าย (Client) (ใช้วัด packet sizing)

bytes received via SQL*Net from client คือ จำนวนไบต์ที่ได้รับจากเครื่องลูกข่าย (Client) จากการใช้คำสั่ง SQL (ใช้วัดประสิทธิภาพของเครือข่าย)

SQL*Net roundtrips to/from client คือ จำนวนข้อความที่ส่งไปและได้รับจากเครื่องลูกข่าย (Client)

sorts (memory) คือ จำนวนข้อมูลที่ถูกเรียงลำดับที่เกิดขึ้นภายในหน่วยความจำ

sorts (disk) คือ จำนวนข้อมูลที่ถูกเรียงลำดับและมีการเขียนลงดิสก์

rows processed คือ จำนวนของแถวที่ถูกโปรเซสของคำสั่ง SQL

สำหรับคำสั่งการแสดง Execution Plan ของฐานข้อมูล Oracle 11g สรุปได้ดังตารางที่ 1

ตารางที่ 1 คำสั่งการแสดง Execution Plan ของ Oracle 11g

ขั้นที่	คำสั่ง	ความหมาย
1	set echo on	แสดงคำสั่งที่ต้องการจะดำเนินการบน SQL*Plus
2	set timing on	แสดงสถิติต่าง ๆ ของคำสั่งเอสคิวแอล
3	set autotrace traceonly	แสดง execution path และ explain statistics หลังจากสอบถามข้อมูลเสร็จ
4	alter system flush shared_pool;	ล้างข้อมูลทั้งหมดในพื้นที่หน่วยความจำของ shared pool
5	alter system flush buffer_cache;	ล้างข้อมูลทั้งหมดในพื้นที่หน่วยความจำของ buffer cache
6	SQL Command	คำสั่งเอสคิวแอลที่ต้องการดำเนินการ

3. ตัวอย่างการใช้งาน Execution Plan

ในหัวข้อนี้จะเป็นตัวอย่างการใช้งาน Execution Plan ของระบบจัดการฐานข้อมูล Oracle Database 11g นอกจากนั้น ในตัวอย่างจะใช้ตารางการขาย (mysales) โดยมีคอลัมน์ในตารางนี้ทั้งหมด 7 คอลัมน์และมีแถวของข้อมูลประมาณ 29 ล้านแถวข้อมูล รายละเอียดดังรูปที่ 4 ต่อไปนี้

```
SQL Plus

SQL> describe mysales;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PROD_ID                                   NOT NULL NUMBER
 CUST_ID                                   NOT NULL NUMBER
 TIME_ID                                   NOT NULL DATE
 CHANNEL_ID                                NOT NULL NUMBER
 PROMO_ID                                  NOT NULL NUMBER
 QUANTITY_SOLD                             NOT NULL NUMBER(10,2)
 AMOUNT_SOLD                               NOT NULL NUMBER(10,2)

SQL> select count(*) from mysales;

  COUNT(*)
----------
  29482976

SQL>
```

รูปที่ 4 โครงสร้างตาราง mysales และจำนวนแถวข้อมูล

ตัวอย่างที่ 1

โจทย์ปัญหา: "ต้องการสอบถามข้อมูลนับจำนวนช่องทางการขายแยกในแต่ละรหัสของช่องทางการขาย โดยไม่นับช่องทางการขายรหัสที่เท่ากับ 3"

จากโจทย์ปัญหาข้างต้นสามารถเขียนคำสั่งเอสคิวแอล ได้ 2 ลักษณะโดยได้ผลลัพธ์เดียวกัน ดังนี้

ใช้ having clasue

```
SELECT CHANNEL_ID, count(CHANNEL_ID)
FROM mysales
GROUP BY CHANNEL_ID
Having CHANNEL_ID != 3;
```

ใช้ where clause

```
SELECT CHANNEL_ID, count(CHANNEL_ID)
FROM mysales
WHERE CHANNEL_ID != 3
GROUP BY CHANNEL_ID;
```

ผลลัพธ์ของ Execution Plan ของคำสั่งเอสคิวแอลของทั้ง 2 queries เป็น ดังนี้

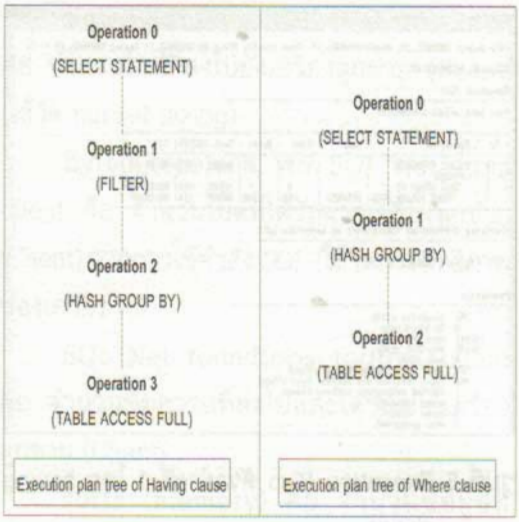
```
SQL Plus

SQL> Select CHANNEL_ID, count(CHANNEL_ID) From mysales Group By CHANNEL_ID Having CHANNEL_ID != 3;

Elapsed: 00:00:18.87

Execution Plan
----------------------------------------------------------
Plan hash value: 2487885872

--------------------------------------------------------------------------------
| Id  | Operation          | Name     | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |          |     3 |     9 | 41842   (5)| 00:08:23 |
|*  1 |  FILTER            |          |       |       |            |          |
|   2 |   HASH GROUP BY    |          |     3 |     9 | 41842   (5)| 00:08:23 |
|   3 |    TABLE ACCESS FULL| MYSALES  |  29M|   84M| 40357   (1)| 00:08:05 |
--------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter("CHANNEL_ID"<>3)

Statistics
----------------------------------------------------------
        389  recursive calls
          0  db block gets
     141662  consistent gets
     141683  physical reads
          0  redo size
        558  bytes sent via SQL*Net to client
        416  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          5  sorts (memory)
          0  sorts (disk)
          3  rows processed
```

รูปที่ 5 Execution Plan ตัวอย่างที่ 1 โดย having clause

```
SQL Plus

SQL> Select CHANNEL_ID, count(CHANNEL_ID) From mysales Where CHANNEL_ID != 3 Group By CHANNEL_ID;

Elapsed: 00:00:09.62

Execution Plan
----------------------------------------------------------
Plan hash value: 4216586231

--------------------------------------------------------------------------------
| Id  | Operation          | Name     | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |          |     4 |    12 | 41519   (4)| 00:08:19 |
|   1 |  HASH GROUP BY     |          |     4 |    12 | 41519   (4)| 00:08:19 |
|*  2 |   TABLE ACCESS FULL| MYSALES  |  22M|   63M| 40414   (1)| 00:08:05 |
--------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - filter("CHANNEL_ID"<>3)

Statistics
----------------------------------------------------------
        389  recursive calls
          0  db block gets
     141661  consistent gets
     141683  physical reads
          0  redo size
        558  bytes sent via SQL*Net to client
        416  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          5  sorts (memory)
          0  sorts (disk)
          3  rows processed
```

รูปที่ 6 Execution Plan ตัวอย่างที่ 1 โดย where clause

จากรูปที่ 5 และ 6 เป็น Execution Plan ของการประมวลผลคำสั่งเอสคิวแอลในโจทย์ตัวอย่างที่ 1 โดยใช้ having clause และ where clause สามารถวาดรูปต้นไม้เพื่อแสดงขั้นตอนการทำงานได้ ดังนี้



รูปที่ 7 ต้นไม้แสดงขั้นตอนการประมวลผลคำสั่งเอสคิวแอลโดยใช้ having และ where

จากรูปที่ 7 สามารถสรุปขั้นตอนการประมวลผลของคำสั่งเอสคิวแอลสำหรับโจทย์ตัวอย่างที่ 1 ได้ ดังนี้

ใช้ having clause

Operation 3 เข้าถึงตาราง MYSALES โดยใช้วิธี TABLE ACCESS FULL จำนวน 29 ล้านแถว ขนาดของข้อมูลเป็น 84 MB. และ Cost (%CPU) เท่ากับ 40,357 (1) หลังจากนั้นจะส่งค่าต่อให้แก่ Operation 2

Operation 2 จะกระทำ GROUP BY คือ จัดกลุ่มข้อมูลตาม CHANNEL_ID หลังจากนั้นจะส่งค่าต่อให้แก่ Operation 1

Operation 1 จะกระทำการกำหนดเงื่อนไข คือ CHANNEL_ID <> 3

Operation 1 จะคืนค่าผลลัพธ์สุดท้ายให้แก่ Operation 0

ใช้ where clause

Operation 2 เข้าถึงตาราง MYSALES โดยใช้วิธี TABLE ACCESS FULL และมีการกำหนดเงื่อนไข คือ CHANNEL_ID <> 3 ได้จำนวนข้อมูล 22 ล้านแถว ขนาดของข้อมูลเป็น 63 MB. และ Cost (%CPU) เท่ากับ 40,424(1) หลังจากนั้นจะส่งค่าต่อให้แก่ Operation 1

Operation 1 จะกระทำ GROUP BY คือ จัดกลุ่มข้อมูลตาม CHANNEL_ID

Operation 1 จะคืนค่าผลลัพธ์สุดท้ายให้แก่ Operation 0

จากการเปรียบเทียบขั้นตอนการประมวลผลของคำสั่งทั้งสองด้วย Execution Plan สรุปได้ว่าในโจทย์ตัวอย่างที่ 1 นี้การเขียนเอสคิวแอลแบบ Where clause จะมีประสิทธิภาพกว่าการใช้ Having clause โดยพิจารณาจากจำนวนขั้นตอนการประมวลผล จำนวนแถวข้อมูล ขนาดของข้อมูล และค่า Cost (%CPU)

ตัวอย่างที่ 2

โจทย์ปัญหา: "ต้องการนับจำนวนการขายที่มียอดจำนวนเงินจากการขายของแต่ละช่วงเวลาน้อยกว่าค่าเฉลี่ยของจำนวนเงินจากการขายของแต่ละช่วงเวลานั้น"

จากโจทย์ปัญหาข้างต้นสามารถเขียนคำสั่งเอสคิวแอลได้ 2 ลักษณะโดยได้ผลลัพธ์เดียวกัน ดังนี้

ใช้ correlated subquery

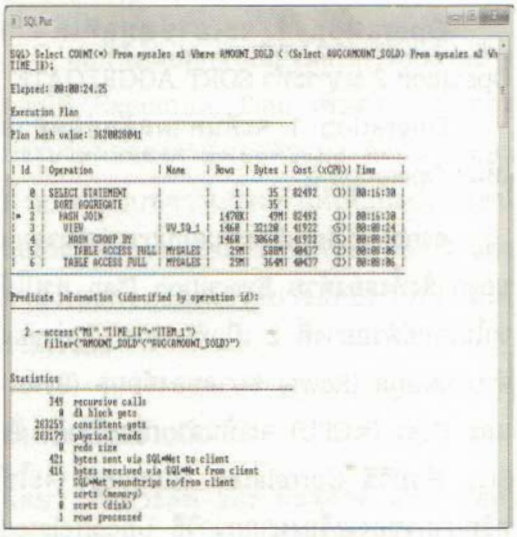
```
SELECT COUNT(*)
FROM mysales m1
WHERE AMOUNT_SOLD <
```

(SELECT AVG(AMOUNT_SOLD)
FROM mysales m2
WHERE m1.TIME_ID = m2.TIME_ID);

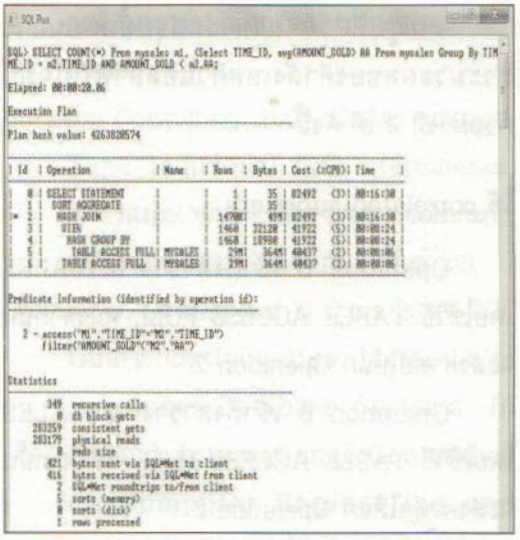
ใช้ uncorrelated subquery

SELECT COUNT(*)
FROM mysales m1,
(Select TIME_ID, avg(AMOUNT_SOLD) AA
FROM mysales
GROUP BY TIME_ID) m2
WHERE m1.TIME_ID = m2.TIME_ID AND
AMOUNT_SOLD < m2.AA;

ผลลัพธ์ของ Execution Plan ของคำสั่งเอสคิวแอลของทั้ง 2 queries เป็น ดังนี้

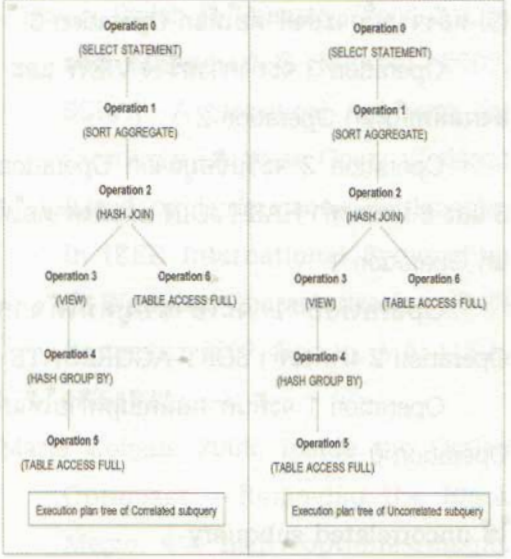


รูปที่ 8 Execution Plan ตัวอย่างที่ 2 โดย Correlated Subquery



รูปที่ 9 Execution plan ตัวอย่างที่ 2 โดย Uncorrelated subquery

จากรูปที่ 8 และ 9 เป็น Execution Plan ของการประมวลผลคำสั่งเอสคิวแอลในโจทย์ตัวอย่างที่ 2 โดยใช้ correlated subquery และ uncorrelated subquery สามารถวาดรูปต้นไม้เพื่อแสดงขั้นตอนการทำงานได้ ดังนี้



รูปที่ 10 Tree แสดงขั้นตอนการประมวลผลคำสั่งเอสคิวแอลโดยใช้ correlated และ uncorrelated

จากรูปที่ 10 สามารถสรุปขั้นตอนการประมวลผลของคำสั่งเอสคิวแอลสำหรับโจทย์ตัวอย่างที่ 2 ได้ ดังนี้

ใช้ correlated subquery

Operation 6 เข้าถึงตาราง MYSALES โดยใช้วิธี TABLE ACCESS FULL หลังจากนั้นจะส่งค่าต่อให้แก่ Operation 2

Operation 6 เข้าถึงตาราง MYSALES โดยใช้วิธี TABLE ACCESS FULL หลังจากนั้นจะส่งค่าต่อให้แก่ Operation 2

Operation 5 เข้าถึงตาราง MYSALES โดยใช้วิธี TABLE ACCESS FULL จำนวน 29 ล้านแถว ขนาดของข้อมูลเป็น 588 MB. และ Cost (%CPU) เท่ากับ 40,437 (2) หลังจากนั้นจะส่งค่าต่อให้แก่ Operation 4

Operation 4 จะกระทำ HASH GROUP BY จำนวน 1,460 แถว ขนาดของข้อมูลเป็น 30,660 Bytes และ Cost (%CPU) เท่ากับ 41,922 (5) หลังจากนั้นจะส่งค่าต่อให้แก่ Operation 3

Operation 3 จะทำการสร้าง VIEW และส่งผลลัพธ์ให้แก่ Operation 2

Operation 2 จะรับข้อมูลจาก Operation 3 และ 6 มากระทำ HASH JOIN และส่งค่าต่อให้แก่ Operation 1

Operation 1 จะเอาข้อมูลที่ได้จาก Operation 2 มากระทำ SORT AGGREGATE

Operation 1 จะคืนค่าผลลัพธ์สุดท้ายให้แก่ Operation 0

ใช้ uncorrelated subquery

Operation 6 เข้าถึงตาราง MYSALES โดยใช้วิธี TABLE ACCESS FULL หลังจากนั้นจะส่งค่าต่อให้แก่ Operation 2

Operation 5 เข้าถึงตาราง MYSALES โดยใช้วิธี TABLE ACCESS FULL จำนวน 29 ล้านแถว ขนาดของข้อมูลเป็น 364 MB. และ Cost (%CPU) เท่ากับ 40,437 (2) หลังจากนั้นจะส่งค่าต่อให้แก่ Operation 4

Operation 4 จะกระทำ HASH GROUP BY จำนวน 1,460 แถว ขนาดของข้อมูลเป็น 18,980 Bytes และ Cost (%CPU) เท่ากับ 41,922 (5) หลังจากนั้นจะส่งค่าต่อให้แก่ Operation 3

Operation 3 จะทำการสร้าง VIEW และส่งผลลัพธ์ให้แก่ Operation 2

Operation 2 จะรับข้อมูลจาก Operation 3 และ 6 มากระทำ HASH JOIN และส่งค่าต่อให้แก่ Operation 1

Operation 1 จะเอาข้อมูลที่ได้จาก Operation 2 มากระทำ SORT AGGREGATE

Operation 1 จะคืนค่าผลลัพธ์สุดท้ายให้แก่ Operation 0

จากการเปรียบเทียบขั้นตอนการประมวลผลของคำสั่งทั้งสองด้วย Execution Plan สรุปได้ว่าในโจทย์ตัวอย่างที่ 2 เมื่อพิจารณาจากค่าของจำนวนข้อมูล (Rows) ขนาดของข้อมูล (Bytes) และ Cost (%CPU) จะเห็นว่า การเขียนคำสั่ง SQL ด้วยวิธี Correlated subquery จะใช้ทรัพยากรของเครื่องมากกว่าวิธี Uncorrelated subquery

4. สรุป

การสอบถามข้อมูล (Query) เป็นการสอบถามไปยังฐานข้อมูลเพื่อดึงข้อมูลที่ต้องการออกมาตามเงื่อนไขที่ต้องการ ระบบจัดการ

ฐานข้อมูลจะใช้ภาษาโครงสร้างการสอบถาม (Structure Query Language – SQL) ในการจัดการและเข้าถึงฐานข้อมูล ปัญหาที่สำคัญประการหนึ่งของการสอบถามข้อมูลออกจากฐานข้อมูล คือ เมื่อมีปริมาณข้อมูลจำนวนมาก และมีความซับซ้อน จะใช้เวลาในการสอบถามข้อมูลเป็นเวลานาน

จากปัญหาข้างต้น เทคนิคหนึ่งที่จะช่วยให้การสอบถามข้อมูลได้เร็วขึ้น คือ การปรับแต่งคำสั่งเอสคิวแอล (SQL Tuning) โดยในบทความได้นำเสนอเครื่องมือ คือ แผนปฏิบัติการ (Execution Plan) ซึ่งสามารถแสดงรายละเอียดของขั้นตอนในการประมวลผลคำสั่ง SQL นอกจากนั้น ยังสามารถแสดงค่าสถิติที่สำคัญของการใช้ทรัพยากรของเครื่องอีกด้วย

สุดท้ายในบทความนี้ยังได้ทำการยกตัวอย่างการใช้ Execution Plan เพื่อตรวจสอบการประมวลผลของคำสั่งเอสคิวแอล ดังนั้น เนื้อหาทั้งหมดของบทความนี้จะทำให้เห็นถึงประโยชน์และวิธีการใช้งาน Execution Plan เพื่อปรับแต่งคำสั่งเอสคิวแอลเพื่อให้การสอบถามข้อมูลได้รวดเร็วขึ้น

5. เอกสารอ้างอิง

นิตยา เกิดประสพ และ กิตติศักดิ์ เกิดประสพ. 2553. **รายงานวิจัยเรื่อง การเพิ่มประสิทธิภาพการประมวลผลข้อคำถามด้วยวิวและโมเดลจากการทำเหมืองข้อมูล.** มหาวิทยาลัยเทคโนโลยีสุรนารี.

Beginner SQL Tutorial. 2007. SQL Tuning or SQL Optimization. สืบค้นวันที่ 29 กุมภาพันธ์ 2556 จาก **http://beginner-sql-tutorial.com/sql-query-tuning.htm.**

Burleson Consulting. 2007. Oracle tuning – Tune individual SQL statements. จาก **http://www.dba-oracle.com/art_sql_tune.htm** [2556, กุมภาพันธ์ 29]

Dandan Li, Lu Han and Yi Ding. 2010. SQL Query Optimization Methods of Relational Database Systems. **In Second International Conference on Computer Engineering and Applications (ICCEA).** 19-21 March 2011. Bali Island, Indonesia. 557-560.

J. Fan, G. Li and L. Zhou. 2011. Interactive SQL Query Suggestion: Making Database User-Friendly. **In 27th International Conference on Data Engineering.** 11-16 April 2011. Hannover, Germany. 351-362.

J Skarie, Biplob K. Debnath, David J. Lilja and Mohamed F. Mokbel. 2007. SCRAP: A Statistical Approach for Creating a Database Query Workload Based on Performance Bottlenecks. **In IEEE International Symposium on Workload Characterization.** 27-29 September 2007. Boston, MA: U.S.A. 183-192.

Maria Colgan. 2008. Inside the Oralce Optimizer – Removing the Black Magic. จาก **http://optimizermagic.blogspot.com/2008/02/displaying-and-reading-execution-plans.html.**

[2556, กุมภาพันธ์ 29]

M. Negri, G. Pelagatti and L. Sbattella. 1991. Formal Semantics of SQL Queries. **ACM Transactions on Database Systems** 17(3): pp. 513-534.

Oracle Corporation. 2003. The Self-Managing Database: Guide Application & SQL Tuning. จาก **http://www.oracle.com/technetwork/database/focus-areas/manageability/twp-manage-automatic-sql-tuning-132307.pdf** [2556, กุมภาพันธ์ 29]

Oracle Corporation Co., Ltd. 2005. Query Optimization in Oracle Database 10g Release 2". **An Oracle White Paper,** June 2005: pp. 4.

P. Belknap, B. Dageville, K. Dias and K. Yagoub. 2009. Self-Tuning for SQL Performance on Oracle Database 11g. **In IEEE International Conference on Data Engineering.** 29 March 29-2 April 2009. Shanghai, China. 1694-1700.

Ramsundaram Perumal. 2009. How to read an Oracle SQL Execution Plan?. จาก **http://perumal.org/how-to-read-an-oracle-sql-execution-plan.** [2556, กุมภาพันธ์ 29]

Surajit Chaudhuri. 2009. Query Optimizers: Time to Rethink the Contract?. In 35th SIGMOD **International Conference on Management of Data Providence.** 29 June 29-2 July 2009. RI, U.S.A. 961-968.