ว.วิทย. มข. 40(3) 844-855 (2555) KKU Sci. J. 40(3) 844-855 (2012)

การเพิ่มประสิทธิภาพการสอบถามข้อมูลด้วยตารางอินเด็กซ์

Query Optimization with Indexing Table

ชาญชัย ศุภอรรถกร[1]

บทคัดย่อ

การเพิ่มประสิทธิภาพการสอบถามข้อมูลเป็นองค์ประกอบของระบบจัดการฐานข้อมูลที่พยามยามหาวิธีที่มีประสิทธิภาพมากที่สุดในการกระทำการค้นหา เทคนิคหนึ่งในการเพิ่มประสิทธิภาพการสอบถามข้อมูล คือ ตารางอินเด็กซ์ซึ่งเป็นวิธีการเพิ่มความเร็วในการเข้าถึงฐานข้อมูล หลักการทำงานของตารางอินเด็กซ์ คือ การเลือกคอลัมน์จากตารางฐานข้อมูลเพื่อสร้างเป็นอินเด็กซ์ ตารางอินเด็กซ์จะถูกสร้างขึ้น โดยบรรจุคอลัมน์ที่ได้เลือกไว้และคอลัมน์ ROWID สำหรับเก็บที่อยู่ของข้อมูล แทนที่จะค้นหาทั่วทั้งตาราง เราสามารถใช้อินเด็กซ์เพื่อหาตำแหน่งที่ตรงกับข้อมูลที่ต้องการได้อย่างรวดเร็ว และประหยัดเวลาได้มากในการค้นหา นอกจากนั้นเรายังได้นำเสนอถึงหลักการเลือกอินเด็กซ์ เช่น การเลือกคอลัมน์ candidate การเลือกคอลัมน์ที่มีค่า cardinality ที่สูงเป็นต้น สุดท้าย เรายังได้ทดสอบประสิทธิภาพของตารางอินเด็กซ์ โดยมีการทดลอง 2 ตัวอย่างเพื่อเปรียบเทียบระหว่างการใช้อินเด็กซ์และไม่ใช้อินเด็กซ์ ผลลัพธ์ของตัวอย่างที่ 1 เมื่อมีการใช้อินเด็กซ์ค่าของเวลาการเข้าถึงดิสก์ลดลง 95.07% และค่าเปอร์เซ็นต์การใช้ซีพียูลดลง 97.96% ผลลัพธ์ของตัวอย่างที่ 2 เมื่อมีการใช้อินเด็กซ์ค่าของเวลาการเข้าถึงดิสก์ลดลง 81.25% และค่าเปอร์เซ็นต์การใช้ซีพียูลดลง 66.99%

[1]ภาควิชาคณิตศาสตร์ สถิติและคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยอุบลราชธานี อ.วารินชำราบ จ.อุบลราชธานี 34190

E-mail: scchansu@ubu.ac.th

ABSTRACT

Query optimization is the component of a database management system that attempts to determine the most efficient way to execute a query. One of the techniques to optimize the query is to use an indexing table that is the primary mean of speeding up access to the database. The principle of indexing table is to select columns from a database table to create the index. Index table is created. This table contains a column that was selected as the index and ROWID column for storing the address of data. Instead of searching full table, we can use the index to quickly position to the first matching value and save a lot of time for searching. In addition, we present the principle of choosing indexes, for example candidate column, high cardinality, etc. Lastly, we tested the efficiency of the indexing table. There are 2 test examples to compare between index and non-index. The first test example result for using index, disk access time reduced by 95.07% and % CPU cost reduced by 97.96%. The second test example result for using index, disk access time reduced by 81.25% and % CPU cost reduced by 66.99%.

คำสำคัญ: การเพิ่มประสิทธิภาพการสอบถามข้อมูล ตารางอินเด็กซ์

Keywords: Query optimization, Indexing table

บทนำ

ข้อมูลนับว่าเป็นสิ่งที่มีความสำคัญต่อหน่วยงาน หรือองค์กรทุกแห่ง เนื่องจากข้อมูลจะเปรียบเสมือนเครื่องมือที่ช่วยประกอบการตัดสินใจ ถ้าหน่วยงาน หรือองค์กรใดขาดซึ่งข้อมูลก็จะไม่สามารถขับเคลื่อนธุรกิจให้บรรลุวัตถุประสงค์ และเป้าหมายได้เลย ดังนั้นจึงเป็นที่มาของระบบฐานข้อมูล (database system) โดยฐานข้อมูลจะหมายถึง กลุ่มของข้อมูลที่ถูกเก็บรวบรวมไว้ โดยมีความสัมพันธ์ซึ่งกันและกัน มีการกำจัดความซ้ำซ้อนของข้อมูลออก และเก็บข้อมูลเหล่านี้ไว้ที่ศูนย์กลาง เพื่อที่จะนำข้อมูลเหล่านี้มาใช้ร่วมกัน โดยทั่วไปองค์กรต่าง ๆ จะสร้างฐานข้อมูลไว้เพื่อเก็บข้อมูลต่าง ๆ ขององค์กร โดยเฉพาะอย่างยิ่งข้อมูลในเชิงธุรกิจ เช่น ข้อมูลของลูกค้า ข้อมูลของสินค้า ข้อมูลของลูกจ้าง และการจ้างงาน เป็นต้น (ชาญชัย, 2553)

ปัญหาที่สำคัญหนึ่งของการใช้งานฐานข้อมูล คือ การเข้าถึงฐานข้อมูลที่มีปริมาณมาก มีความซับซ้อน และต้องการการตอบสนองที่รวดเร็ว (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) การสอบถามข้อมูลเหล่านี้มีปริมาณงานมาก มีความซับซ้อน และฐานข้อมูลมีขนาดใหญ่ (Chaudhuri, 2009) ในทาง

ปฏิบัติมีหลายเทคนิคที่ช่วยเพิ่มความเร็วในการเข้าถึงฐานข้อมูล หรือเพิ่มประสิทธิภาพในการสอบถามข้อมูล (query optimization) เช่น การสร้างตารางอินเด็กซ์ (indexing table) การปรับแต่งคำสั่งเอสคิวแอล (SQL tuning) การปรับเครื่องแม่ข่ายฐานข้อมูล (database server tuning) เป็นต้น สำหรับในบทความนี้จะกล่าวถึงเฉพาะเทคนิคการเพิ่มประสิทธิภาพการสอบถามข้อมูลด้วยตารางอินเด็กซ์

ระบบจัดการฐานข้อมูลหลายตัวนิยมใช้เทคนิคการสร้างตารางอินเด็กซ์เพื่อเพิ่มประสิทธิภาพการสอบถามฐานข้อมูล ในระบบจัดการฐานข้อมูลเชิงสัมพันธ์ (relational database management systems) เทคนิคตารางอินเด็กซ์ได้รับความนิยมและมีประสิทธิภาพในการเพิ่มความเร็วในการประมวลผลฐานข้อมูล (Honishi et al., 1992) นอกจากนั้น ในฐานข้อมูลเชิงวัตถุ (object-oriented database; OODB) ที่วัตถุมีความซับซ้อนและมีปริมาณมาก ก็นิยมใช้เทคนิคตารางอินเด็กซ์ในการเพิ่มความเร็วการสอบถามบนฐานข้อมูลเช่นกัน (Huang et al., 2000)

บทความนี้จะประกอบไปด้วย หัวข้อหลักการทำงานของตารางอินเด็กซ์ หัวข้อข้อดีและข้อเสียของตารางอินเด็กซ์ หัวข้อหลักการเลือกอินเด็กซ์ หัวข้อการทดสอบประสิทธิภาพของตารางอินเด็กซ์ และบทสรุป

หลักการทำงานของตารางอินเด็กซ์

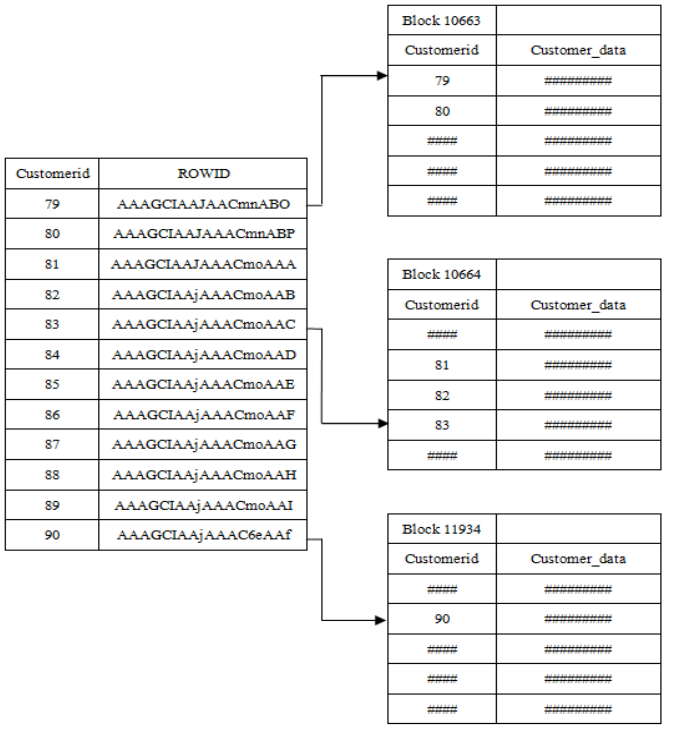
ตารางอินเด็กซ์ (indexing table) คือ โครงสร้างในการเข้าถึงข้อมูลในตารางฐานข้อมูลที่ใช้เพิ่มความเร็วในการดึงแถวข้อมูล หรือเรคคอร์ด (record) ออกมาจากตารางฐานข้อมูล โดยตารางอินเด็กซ์ที่ถูกสร้างขึ้นจะถูกเก็บไว้แยกจากตารางปกติในฐานข้อมูล โดยปกติถ้าไม่มีการประกาศตารางอินเด็กซ์ไว้การค้นหาข้อมูลในตารางนั้นจะต้องทำแบบเรียงลำดับจากเรคคอร์ดแรกไปจนถึงเรคคอร์ดสุดท้าย การสร้างตารางอินเด็กซ์ใด ๆ จะทำได้โดยการเลือกคอลัมน์ใดคอลัมน์หนึ่ง หรือมากกว่าหนึ่งคอลัมน์จากตารางมาเป็นอินเด็กซ์และตารางหนึ่ง ๆ สามารถมีได้หลายอินเด็กซ์

สำหรับหลักการทำงานของตารางอินเด็กซ์จะยกตัวอย่างตารางข้อมูลลูกค้า (customer table) ตามรูปต่อไปนี้

Customer_id	Customer_data
82	######
80	######
85	######
83	######
79	######
81	######
88	######
86	######
84	######
90	######
89	######
83	######

รูปที่ 1 Customer Table

จากรูปที่ 1 ตารางข้อมูลลูกค้า (customer table) ประกอบไปด้วยคอลัมน์รหัสลูกค้า (Customer_id) และคอลัมน์ข้อมูลรายการลูกค้า (Customer_data) สังเกตว่าข้อมูลในคอลัมน์รหัสลูกค้าไม่ได้ถูกเรียงลำดับ สมมุติต้องการสอบถามข้อมูลของลูกค้ารหัส 83 จะต้องทำการค้นหาทั้งตาราง (full scan) ถึงจะเจอข้อมูลลูกค้ารหัส 83 นี้ ดังนั้น จึงได้ทำการสร้างอินเด็กซ์เพื่อเพิ่มประสิทธิภาพในการสอบถามข้อมูล ดังรูปต่อไปนี้



รูปที่ 2 Access Path from Indexed to Table (บริษัท ออราเคิล คอร์ปอเรชั่น (ประเทศไทย) จำกัด, 2554)

จากรูปที่ 2 ตารางข้อมูลลูกค้านี้ได้มีการสร้างอินเด็กซ์ให้กับคอลัมน์ Customer_id ดังนั้นจะมีการสร้างตารางอินเด็กซ์ขึ้นมา โดยตารางอินเด็กซ์ที่สร้างขึ้นจะประกอบไปด้วยคอลัมน์ Customer_id ที่เลือกให้เป็นอินเด็กซ์ และ ROWID ทำหน้าที่เก็บตำแหน่งที่อยู่ของรหัสลูกค้าในแต่ละรหัส ข้อมูลในคอลัมน์ Customer_id ของตารางอินเด็กซ์จะมีการเรียงลำดับข้อมูลเอาไว้ สมมุติต้องการสอบถามข้อมูลลูกค้าที่มีรหัส 83 การทำงานจะเริ่มจากการเข้าไปค้นรหัสลูกค้า 83 ในตารางอินเด็กซ์ โดยจะเริ่มเปรียบเทียบรหัสลูกค้าตั้งแต่รหัสลูกค้า 79 ว่าคือรหัสลูกค้า 83 หรือไม่ ถ้าไม่ใช่ก็เลื่อนไปยังรหัสลูกค้า 80 ทำเช่นนี้ไปจนเจอรหัสลูกค้าที่ 83 เมื่อเจอแล้วก็จะอาศัย ROWID เพื่อชี้ตำแหน่งที่เก็บข้อมูลลูกค้าคนนี้ในตาราง customer ซึ่งก็คือ ตารางข้อมูล block ที่ 10664 จะเห็นได้ว่าการค้นหานี้จะไม่ได้ทำการค้นหาข้อมูลหมดทั้งตาราง ทำให้การสอบถามข้อมูลได้รวดเร็วขึ้น นอกจากนั้น เมื่อค้นหาผ่านรหัสลูกค้า 83 ไปจนถึงรหัสลูกค้า 84 แล้วก็จะออกจากการทำงานทันที เนื่องจากค่ารหัสลูกค้า 84 มีค่าเกินรหัสลูกค้า 83 ไปแล้ว จึงไม่ต้องเสียเวลาไปค้นหาข้อมูลที่เหลือ และเหตุที่ไม่เรียงลำดับข้อมูลรหัสลูกค้าในตารางลูกค้าก่อน ก็เนื่องจากในกรณีที่ต้องการ

สร้างอินเด็กซ์มากกว่าหนึ่งอินเด็กซ์ เช่น ต้องการสร้างอินเด็กซ์ให้กับรหัสลูกค้า และชื่อลูกค้า จะไม่สามารถทำได้ เนื่องจาก เราไม่สามารถเรียงลำดับแถวของข้อมูลด้วยคอลัมน์ 2 คอลัมน์ที่แตกต่างภายในครั้งเดียวกันได้ แต่สำหรับตารางอินเด็กซ์มีความสามารถที่จะสร้างหลาย ๆ อินเด็กซ์พร้อมกันได้

ข้อดีและข้อเสียของตารางอินเด็กซ์ (DuBois, 2009)

จากหัวข้อที่ผ่านมาในเรื่องหลักการทำงานของตารางอินเด็กซ์ ในหัวข้อนี้จะสรุปถึงข้อดี และข้อเสียในการใช้งานตารางอินเด็กซ์ ดังนี้

ข้อดี

1) การใช้งานตารางอินเด็กซ์ ทำให้เพิ่มความเร็วในการสอบถามข้อมูลที่ต้องการ หรือข้อมูลแถวในตารางที่ตรงตามเงื่อนไขของประโยคคำสั่ง Where

2) การสอบถามข้อมูลที่ใช้ฟังก์ชัน MIN() ในการหาค่าน้อยที่สุด และ MAX() ในการหาค่ามากที่สุด ตารางอินเด็กซ์จะช่วยค้นหาข้อมูลได้อย่างรวดเร็ว โดยการค้นหาจะไม่ต้องตรวจสอบข้อมูลทุกแถวในตาราง

3) การสอบถามข้อมูลที่ใช้คำสั่ง ORDER BY ในการเรียงลำดับข้อมูล และ GROUP BY ในการจับกลุ่มข้อมูล ตารางอินเด็กซ์จะกระทำได้อย่างรวดเร็ว

ข้อเสีย

1) การสอบถามข้อมูลที่เป็นการดึงข้อมูลออกมาจากตาราง การใช้งานตารางอินเด็กซ์จะทำให้การทำงานเร็วขึ้น แต่การสอบถามข้อมูลที่เป็นการเพิ่ม (insert) ลบ (delete) และแก้ไข (update) ข้อมูลที่เป็นอินเด็กซ์จะทำงานได้ช้า เนื่องจากการทำงานเหล่านี้จะเป็นการเขียนข้อมูล ซึ่งไม่เพียงเขียนข้อมูลในแต่ละแถวข้อมูลในตาราง แต่ยังมีการเปลี่ยนแปลงที่ตารางอินเด็กซ์ด้วย

2) การสร้างตารางอินเด็กซ์จะทำให้เปลืองเนื้อที่ของเครื่องคอมพิวเตอร์มากกว่าการไม่ใช้ตารางอินเด็กซ์

หลักการเลือกอินเด็กซ์ (DuBois, 2009)

ในบางครั้งเมื่อสร้างตารางอินเด็กซ์แล้ว ลองทำการสอบถามข้อมูลปรากฏว่าการดึงข้อมูลยังทำงานได้ช้า ประเด็นที่สำคัญในหัวข้อนี้ คือ การเลือกสร้างอินเด็กซ์ให้กับคอลัมน์ไหนดีถึงจะทำให้ทำการสอบถามข้อมูลได้อย่างรวดเร็ว หลักการเลือกสร้างอินเด็กซ์ให้กับคอลัมน์ มีดังนี้

1) ให้เลือกอินเด็กซ์คอลัมน์ที่เป็นคอลัมน์ที่จะใช้เป็นเงื่อนไขในการค้นหา เรียงลำดับ หรือจัดกลุ่มข้อมูล ไม่ควรเลือกคอลัมน์ที่เป็นส่วนของการ select ผลลัพธ์อย่างเดียวเท่านั้น ในที่นี้จะเรียกคอลัมน์ที่เหมาะกับการเลือกสร้างเป็นอินเด็กซ์ว่า candidate

```
SELECT   col_a ← not a candidate
FROM   tbl1 LEFT JOIN tbl2
ON tbl1.col_b = tbl2.col_c ← candidates
WHERE col_d = expr; ← a candidate
```

ในที่นี้คอลัมน์ที่เป็น candidate ที่ดีควรเลือกเป็นอินเด็กซ์ คือ col_b, col_c และ col_d

2) ให้เลือกคอลัมน์ที่มีค่า cardinality สูง ค่า cardinality ในที่นี้ คือ ตัวเลขแสดงจำนวนของค่าที่ไม่ซ้ำกันของคอลัมน์ในตาราง เช่น ในคอลัมน์หนึ่งบรรจุข้อมูล 1, 3, 7, 4, 7 และ 3 คอลัมน์นี้จะมีค่า cardinality เท่ากับ 4 ดังนั้น สมมุติในตารางฐานข้อมูลมีคอลัมน์อายุ (เก็บข้อมูลอายุที่แตกต่างกันจำนวนมาก) และ เพศ (เก็บข้อมูล คือ ‘M’ แทนเพศชาย และ ‘F’

แทนเพศหญิง เท่านั้น) คอลัมน์ที่ควรเลือกในการสร้างอินเด็กซ์ คือ คอลัมน์อายุ เนื่องจากมีค่า cardinality มากกว่าคอลัมน์เพศ นอกจากนั้น หลักการเลือกคอลัมน์ที่มีค่า cardinality สูง สามารถหาได้จากสูตรต่อไปนี้ DISTINCT_KEYS/ NUM_ROWS โดยคอลัมน์ที่เราควรเลือกควรจะเป็นค่าที่ไม่ซ้ำ และไม่เป็นค่าว่าง (NULL)

ตัวอย่างในการเลือกอินเด็กซ์ที่ดี (ORACLE DBA & ALL IT, 2554)

ถ้าตารางมีข้อมูล 100,000 เรคคอร์ด และในคอลัมน์ที่ต้องการจะนำมาทำอินเด็กซ์มีจำนวน 88,000 เรคคอร์ด ที่มีค่าไม่ซ้ำกัน หลังจากนั้นให้เอามาแทนค่าในสูตร ได้เป็น 88,000/100,000 = 0.88 โดยถ้าผลลัพธ์เข้าใกล้ 1 จะถือว่าเป็นอินเด็กซ์ที่ดี

ตัวอย่างในการเลือกอินเด็กซ์ที่ไม่ดี (ORACLE DBA & ALL IT, 2554)

ถ้าตารางมีข้อมูล 100,000 เรคคอร์ด และในคอลัมน์ที่ต้องการจะนำมาทำอินเด็กซ์มีจำนวน 500 เรคคอร์ด ที่มีค่าไม่ซ้ำกัน หลังจากนั้นให้เอามาแทนค่าในสูตร ได้เป็น 500/100,000 = 0.005 โดยถ้าผลลัพธ์ออกมาแบบนี้ แสดงว่าคอลัมน์นี้ไม่เหมาะกับการสร้างเป็นอินเด็กซ์ การสอบถามข้อมูลแบบ full scan จะมีประสิทธิภาพมากกว่าการใช้อินเด็กซ์

3) ให้เลือกคอลัมน์ที่มีประเภทของข้อมูล (Data Type) ขนาดเล็ก เช่น คอลัมน์เก็บข้อมูลจำนวนเต็ม ในการประกาศประเภทของข้อมูลควรเลือกประกาศประเภทของข้อมูลเป็น MEDIUMINT จะดีกว่า BIGINT ถ้าประเภทข้อมูล MEDIUMINT ก็เพียงพอในการเก็บข้อมูล หรือในการเก็บข้อมูลอักขระไม่ควรเลือกคอลัมน์ที่ประกาศประเภทข้อมูลเป็น char(100) ถ้าข้อมูลอักขระไม่ยาวเกิน 25 ตัว เหตุผลที่ควรเลือกคอลัมน์ที่มีประเภทของข้อมูลขนาดเล็ก เนื่องจากข้อมูลที่มีขนาดเล็กจะทำการเปรียบเทียบได้เร็วกว่าและข้อมูลที่มีขนาดเล็กจะต้องการการถ่ายโอนข้อมูลเข้าและออกระหว่างดิสก์ (Disk I/O) ที่น้อยกว่า

4) ไม่ควรสร้างอินเด็กซ์ให้กับทุกคอลัมน์ บางกรณีเมื่อสร้างอินเด็กซ์ให้กับคอลัมน์แล้วจะเป็นการเพิ่มประสิทธิภาพในการสอบถามข้อมูล จึงสร้างอินเด็กซ์ให้กับทุกคอลัมน์ แต่การทำเช่นนั้นเป็นการเพิ่มพื้นที่ดิสก์ของเครื่องคอมพิวเตอร์ และเป็นการเพิ่มงานให้กับการเขียนข้อมูล ผลลัพธ์คือ เมื่อมีการเขียนข้อมูลของตาราง เช่น การเพิ่ม ลบ และแก้ไขข้อมูลจะทำงานได้ช้าเพราะต้องทำกับทุกคอลัมน์ที่ถูกสร้างเป็นอินเด็กซ์

5) ในระดับตารางมีคำแนะนำว่า ไม่ควรสร้างอินเด็กซ์กับตารางที่มีแถวข้อมูลไม่มากนัก เพราะถ้ามีแถวข้อมูลไม่มากการเลือกใช้วิธีสอบถามข้อมูลแบบ full scan คือค้นหาตั้งแต่แถวแรกไปจนถึงแถวสุดท้าย ระยะเวลาการสอบถามข้อมูลก็จะไม่ต่างกับการค้นหาจากอินเด็กซ์มากนัก (CODE FUNCTION.IN.TH, 2553)

6) ประการสุดท้าย ไม่ควรสร้างอินเด็กซ์กับตารางที่มีการเพิ่ม ลบ แก้ไขข้อมูลบ่อย การสอบถามข้อมูลที่เป็นการเพิ่ม ลบ แก้ไขข้อมูลจะทำงานได้ช้า เพราะเมื่อมีการเพิ่ม ลบ แก้ไขข้อมูลทุกครั้ง ก็จะต้องมีการไปปรับปรุงอินเด็กซ์ทุกครั้ง (CODE FUNCTION.IN.TH, 2553)

การทดสอบประสิทธิภาพของตารางอินเด็กซ์

ในหัวข้อนี้จะเป็นการวัดประสิทธิภาพการสอบถามข้อมูลระหว่างการใช้ และไม่ใช้ตารางอินเด็กซ์ โดยในการทดสอบประสิทธิภาพนี้จะกระทำบนเครื่อง PC AMD Athlon 64x2 Dual Core Processor 5200+2.70 GHZ หน่วยความจำ (RAM) เท่ากับ 1 GB บนระบบปฏิบัติการ Microsoft Windows XP

Service Pack3 และใช้ระบบจัดการฐานข้อมูล Oracle Database 11g

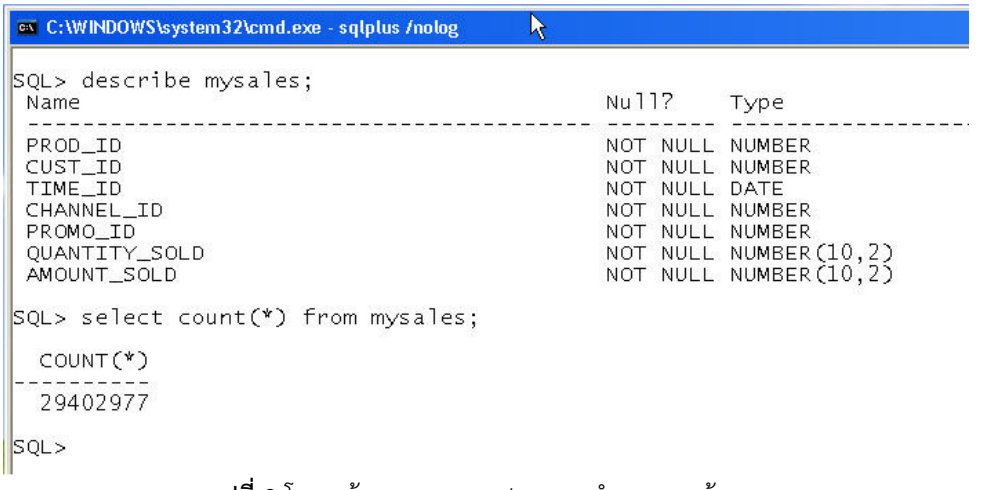
ในการวัดประสิทธิภาพจะใช้ค่า disk access time และค่า % CPU cost ในการเปรียบเทียบประสิทธิภาพ โดยนิยามความหมายของทั้ง 2 ค่า คือ

1) Disk access time คือ ระยะเวลาในการเข้าถึง ตั้งแต่ได้รับคำสั่งไปประมวลผลข้อมูลของโปรเซสเซอร์ และส่งค่ากลับคืนมา โดยปกติค่า disk access time นี้จะวัดจากเวลาในการค้นหา (seek time) ซึ่งเป็นช่วงเวลาที่หัวอ่านเลื่อนจากแทร็คที่อยู่ในขณะนั้นไปอยู่เหนือแทร็คที่ต้องการ (Huang Y.F. et al., 2000)

2) % CPU cost คือ ค่าเปอร์เซ็นต์ของรอบการทำงานของเครื่อง (machine cycle) ในการกระทำคำสั่งต่าง ๆ โดยเป็นค่าที่เกี่ยวข้องกับโหลดการทำงานของ CPU และการใช้ทรัพยากรของเครื่อง (BURLESON CONSULTING, 2011)

การทดลองที่ 1 (ทดสอบประสิทธิภาพการใช้และไม่ใช้อินเด็กซ์ในการสอบถามข้อมูล)

เป็นการทดสอบประสิทธิภาพการสอบถามข้อมูลของตารางการขาย (mysales) โดยมีคอลัมน์ในตารางนี้ทั้งหมด 7 คอลัมน์และมีแถวของข้อมูลประมาณ 29 ล้านแถวข้อมูล รายละเอียดดังรูปที่ 3 ต่อไปนี้



รูปที่ 3 โครงสร้างตาราง mysales และจำนวนแถวข้อมูล

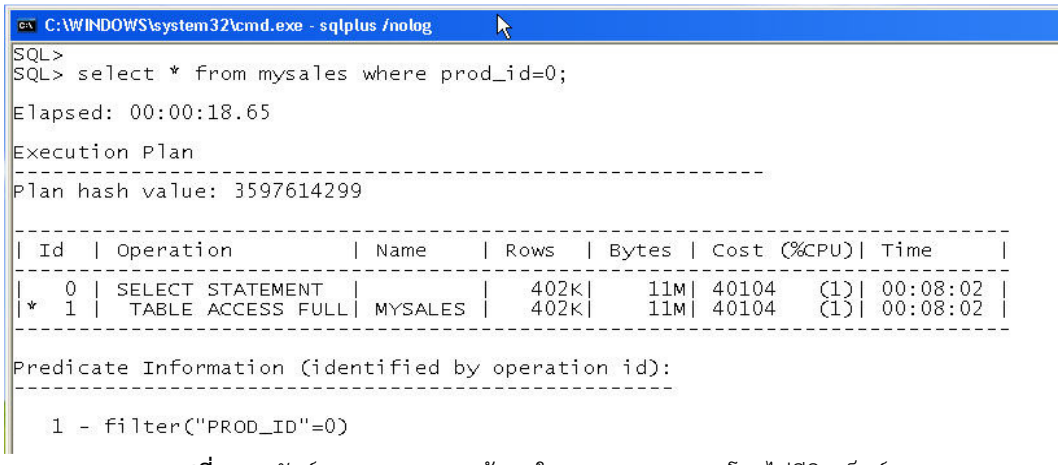
หมายเหตุ

1. แถวข้อมูลสุดท้ายได้เพิ่มข้อมูลโดยใช้คำสั่ง SQL ต่อไปนี้ insert into mysales values (0,0,sysdate,0,0,0,0); เพื่อใช้เป็นเงื่อนไขในการสอบถามข้อมูล

2. ตาราง mysales นี้ไม่ได้สร้างอินเด็กซ์ให้กับคอลัมน์ใด ๆ หลังจากนั้นได้ทำการสอบถามข้อมูลด้วยคำสั่ง SQL ต่อไปนี้

select * from mysales where prod_id = 0;

เพื่อให้การค้นหาเจอในแถวข้อมูลสุดท้ายของตาราง ผลลัพธ์ของการสอบถามข้อมูลนี้ ปรากฏดังรูปที่ 4 ต่อไปนี้

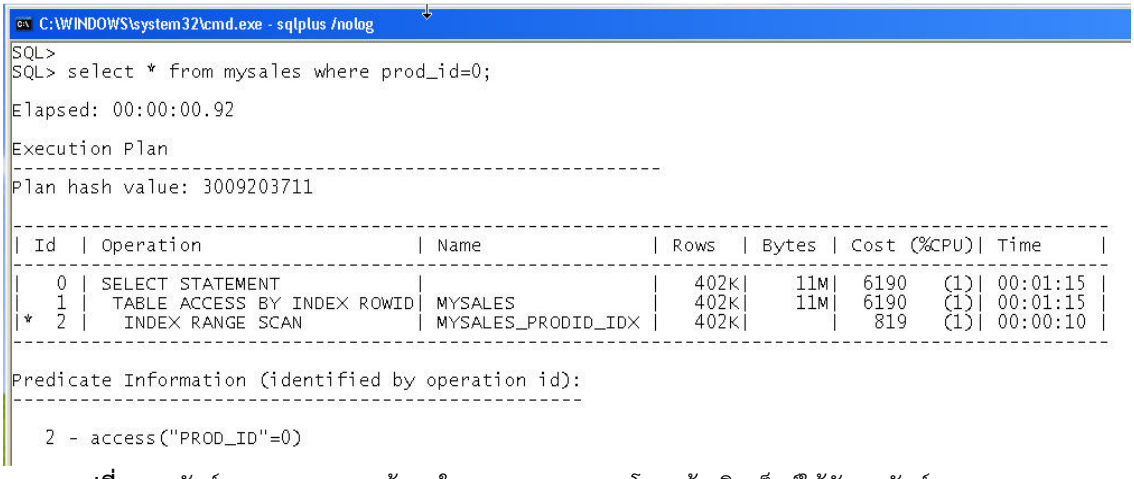


```
C:\WINDOWS\system32\cmd.exe - sqlplus /nolog
SQL>
SQL> select * from mysales where prod_id=0;

Elapsed: 00:00:18.65

Execution Plan
----------------------------------------------------------
Plan hash value: 3597614299

--------------------------------------------------------------------------------
| Id  | Operation          | Name    | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |         |   402K|    11M| 40104   (1)| 00:08:02 |
|*  1 |  TABLE ACCESS FULL| MYSALES |   402K|    11M| 40104   (1)| 00:08:02 |
--------------------------------------------------------------------------------

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

   1 - filter("PROD_ID"=0)
```

รูปที่ 4 ผลลัพธ์จากการสอบถามข้อมูลในตาราง mysales โดยไม่มีอินเด็กซ์

จากรูปที่ 4 ปรากฏว่าในการสอบถามข้อมูลจะใช้เวลา Disk Access Time เท่ากับ 18.65 มิลลิวินาที และมีค่า % CPU Cost เท่ากับ 40,104 โดยการสอบถามข้อมูลนี้เป็นแบบ TABLE ACCESS FULL ซึ่งก็คือ การค้นหาทั้งตาราง (Full Scan)

ต่อมาทำการสร้างอินเด็กซ์ให้กับคอลัมน์ prod_id และทำการสอบถามข้อมูลใหม่ได้ผลลัพธ์ ดังรูปที่ 5 ต่อไปนี้



```
C:\WINDOWS\system32\cmd.exe - sqlplus /nolog
SQL>
SQL> select * from mysales where prod_id=0;

Elapsed: 00:00:00.92

Execution Plan
----------------------------------------------------------
Plan hash value: 3009203711

--------------------------------------------------------------------------------------------------
| Id  | Operation                    | Name                | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                     |   402K|    11M|  6190   (1)| 00:01:15 |
|   1 |  TABLE ACCESS BY INDEX ROWID| MYSALES             |   402K|    11M|  6190   (1)| 00:01:15 |
|*  2 |   INDEX RANGE SCAN           | MYSALES_PRODID_IDX |   402K|       |   819   (1)| 00:00:10 |
--------------------------------------------------------------------------------------------------

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

   2 - access("PROD_ID"=0)
```

รูปที่ 5 ผลลัพธ์จากการสอบถามข้อมูลในตาราง mysales โดยสร้างอินเด็กซ์ให้กับคอลัมน์ prod_id

จากรูปที่ 5 ปรากฏว่าในการสอบถามข้อมูลจะใช้เวลา Disk Access Time เท่ากับ 00.92 มิลลิวินาที และมีค่า % CPU Cost เท่ากับ 819 โดยการสอบถามข้อมูลนี้เป็นแบบ INDEX RANGE SCAN ซึ่งก็คือ การค้นหาโดยใช้ตารางอินเด็กซ์

การทดลองที่ 2 (ทดสอบประสิทธิภาพการใช้และไม่ใช้อินเด็กซ์ในการสอบถามข้อมูล และการเลือก candidate column)

เป็นการทดสอบประสิทธิภาพการสอบถามข้อมูลของตารางลูกค้า (customers) โดยมีคอลัมน์ใน

ตารางนี้ทั้งหมด 23 คอลัมน์ และมีแถวของข้อมูล 55,500 แถวข้อมูล รายละเอียดดังรูปที่ 6 ต่อไปนี้

```
C:\WINDOWS\system32\cmd.exe - sqlplus /nolog
SQL> describe customers;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 CUST_ID                                   NOT NULL NUMBER
 CUST_FIRST_NAME                           NOT NULL VARCHAR2(20)
 CUST_LAST_NAME                            NOT NULL VARCHAR2(40)
 CUST_GENDER                               NOT NULL CHAR(1)
 CUST_YEAR_OF_BIRTH                        NOT NULL NUMBER(4)
 CUST_MARITAL_STATUS                                VARCHAR2(20)
 CUST_STREET_ADDRESS                       NOT NULL VARCHAR2(40)
 CUST_POSTAL_CODE                          NOT NULL VARCHAR2(10)
 CUST_CITY                                 NOT NULL VARCHAR2(30)
 CUST_CITY_ID                              NOT NULL NUMBER
 CUST_STATE_PROVINCE                       NOT NULL VARCHAR2(40)
 CUST_STATE_PROVINCE_ID                    NOT NULL NUMBER
 COUNTRY_ID                                NOT NULL NUMBER
 CUST_MAIN_PHONE_NUMBER                    NOT NULL VARCHAR2(25)
 CUST_INCOME_LEVEL                                  VARCHAR2(30)
 CUST_CREDIT_LIMIT                                  NUMBER
 CUST_EMAIL                                         VARCHAR2(30)
 CUST_TOTAL                                NOT NULL VARCHAR2(14)
 CUST_TOTAL_ID                             NOT NULL NUMBER
 CUST_SRC_ID                                        NUMBER
 CUST_EFF_FROM                                      DATE
 CUST_EFF_TO                                        DATE
 CUST_VALID                                         VARCHAR2(1)

SQL>
```

รูปที่ 6 โครงสร้างตาราง customers

หลังจากนั้นได้ทำการสอบถามข้อมูลด้วยคำสั่ง SQL ต่อไปนี้

```
select /* + FULL(c) */ c.*
from customers c
where cust_gender = 'M'
AND cust_postal_code = 40804
AND cust_credit_limit  = 10000
```

ซึ่งเป็นการสอบถามข้อมูลโดยบังคับให้ใช้วิธี TABLE ACCESS FULL ผลลัพธ์ของการสอบถามข้อมูลนี้ ปรากฏดังรูปที่ 7 ต่อไปนี้

```
C:\WINDOWS\system32\cmd.exe - sqlplus /nolog
SQL>
SQL> SELECT /*+ FULL(c) */ c.*
  2  FROM   customers c
  3  WHERE  cust_gender    = 'M'
  4  AND    cust_postal_code = 40804
  5  AND    cust_credit_limit = 10000
  6  /

6 rows selected.

Elapsed: 00:00:00.48

Execution Plan
----------------------------------------------------------
Plan hash value: 2008213504

--------------------------------------------------------------------------------
| Id  | Operation         | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |           |     5 |   900 |   406   (1)| 00:00:05 |
|*  1 |  TABLE ACCESS FULL| CUSTOMERS |     5 |   900 |   406   (1)| 00:00:05 |
--------------------------------------------------------------------------------
```

รูปที่ 7 ผลลัพธ์จากการสอบถามข้อมูลในตาราง customers โดยไม่มีอินเด็กซ์

จากรูปที่ 7 ปรากฏว่าในการสอบถามข้อมูลจะใช้เวลา disk access time เท่ากับ 00.48 มิลลิวินาที และมีค่า % CPU Cost เท่ากับ 406 โดยการสอบถามข้อมูลนี้เป็นแบบ TABLE ACCESS FULL

ต่อมาได้ทำการสร้างอินเด็กซ์ให้กับตาราง customers นี้ทั้งหมด 3 คอลัมน์ คือ cust_gender, cust_postal_code และ cust_credit_limit และทำการสอบถามข้อมูลด้วยคำสั่ง SQL ต่อไปนี้

```
select /* + INDEX(c) */ c.*
from customers c
where cust_gender = 'M'
AND cust_postal_code = 40804
AND cust_credit_limit  = 10000
```

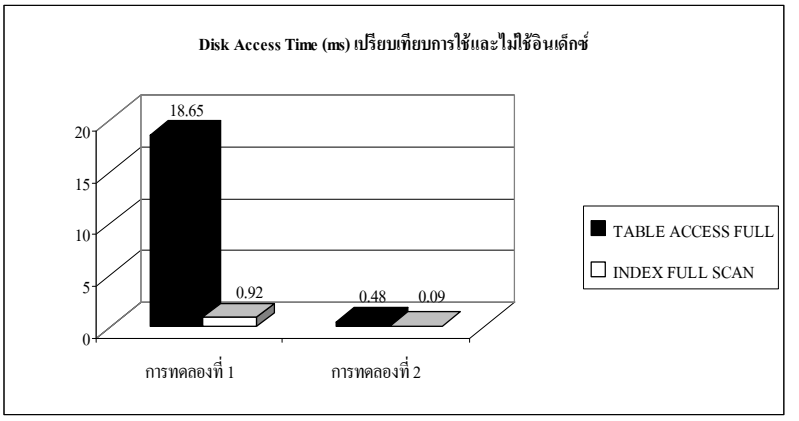
ซึ่งเป็นการสอบถามข้อมูลโดยให้ใช้อินเด็กซ์ในการค้นหา และให้เลือกคอลัมน์ที่จะเป็นอินเด็กซ์ในการค้นหาเองจาก 3 คอลัมน์ข้างต้น ผลลัพธ์ของการสอบถามข้อมูลนี้ ปรากฏดังรูปที่ 8 ต่อไปนี้

```
C:\WINDOWS\system32\cmd.exe - sqlplus /nolog
SQL> SELECT /*+ INDEX(c) */ c.*
  2  FROM   customers c
  3  WHERE  cust_gender   = 'M'
  4  AND    cust_postal_code = 40804
  5  AND   cust_credit_limit = 10000
  6  /

6 rows selected.

Elapsed: 00:00:00.09

Execution Plan
----------------------------------------------------------
Plan hash value: 1928091631

-------------------------------------------------------------------------------------------------------------
| Id  | Operation                    | Name                       | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                            |     5 |   900 |   199   (1)| 00:00:03 |
|*  1 |  TABLE ACCESS BY INDEX ROWID| CUSTOMERS                  |     5 |   900 |   199   (1)| 00:00:03 |
|*  2 |   INDEX FULL SCAN            | CUST_CUST_POSTAL_CODE_IDX |    69 |       |   134   (1)| 00:00:02 |
-------------------------------------------------------------------------------------------------------------
```

รูปที่ 8 ผลลัพธ์จากการสอบถามข้อมูลในตาราง customers โดยกำหนดให้เลือกคอลัมน์อินเด็กซ์เอง

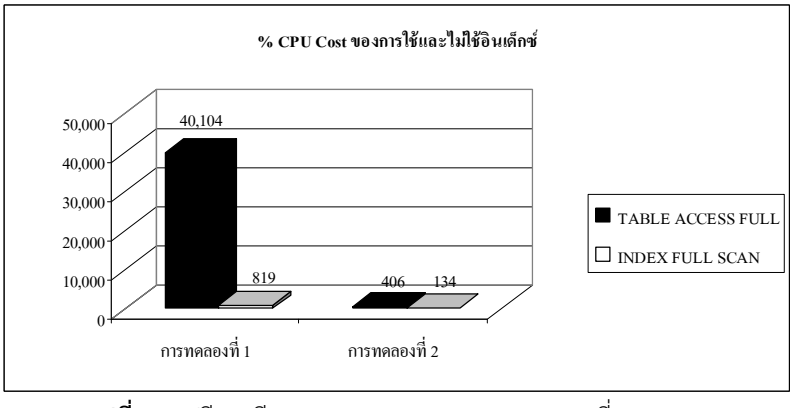
จากรูปที่ 8 ปรากฏว่าในการสอบถามข้อมูลจะใช้เวลา disk access time เท่ากับ 0.09 มิลลิวินาที และมีค่า % CPU cost เท่ากับ 134 โดยการสอบถามข้อมูลนี้เป็นแบบ index full scan ด้วยคอลัมน์ cust_postal_code เนื่องจากคอลัมน์นี้เก็บค่ารหัสไปรษณีย์ของลูกค้า ซึ่งมีค่าไม่ซ้ำกันมากทำให้มีค่า cardinality สูงที่สุด ส่วนคอลัมน์ cust_credit_limit เก็บค่าวงเงินเครดิตของลูกค้า ซึ่งมีค่าไม่ซ้ำกันน้อยกว่า cust_postal_code และคอลัมน์ cust_gender เก็บข้อมูลเพศของลูกค้ามีค่าไม่ซ้ำกันน้อยที่สุด

สรุปจากการทดลองทั้ง 2 ครั้ง ได้ข้อสรุป disk access time ตามรูปที่ 9 ดังนี้



รูปที่ 9 เปรียบเทียบ Disk Access Time (ms) ของการทดลองที่ 1 และ 2

เปรียบเทียบการสอบถามข้อมูลด้วยการใช้และไม่ใช้อินเด็กซ์จากการทดลองที่ 1 และ 2 ได้ว่า การทดลองที่ 1 การใช้อินเด็กซ์ disk access time ลดลง 95.07% และ การทดลองที่ 2 การใช้อินเด็กซ์ disk access time ลดลง 81.25% และข้อสรุปของ % CPU cost ตามรูปที่ 10 ดังนี้



รูปที่ 10 เปรียบเทียบ % CPU Cost ของการทดลองที่ 1 และ 2

เปรียบเทียบการสอบถามข้อมูลด้วยการใช้และไม่ใช้อินเด็กซ์จากการทดลองที่ 1 และ 2 ได้ว่า การทดลองที่ 1 การใช้อินเด็กซ์ % CPU cost ลดลง 97.96% และ การทดลองที่ 2 การใช้อินเด็กซ์ % CPU cost ลดลง 66.99%

บทสรุป

การสอบถามข้อมูล (query) เป็นการสอบถามไปยังฐานข้อมูลเพื่อดึงข้อมูลที่ต้องการออกมาตามเงื่อนไขที่ต้องการ ระบบจัดการฐานข้อมูลจะใช้ภาษาโครงสร้างการสอบถาม (structure query language; SQL) ในการจัดการและเข้าถึงฐานข้อมูล ปัญหาที่สำคัญประการหนึ่งของการสอบถามข้อมูลออกจากฐานข้อมูล คือ เมื่อมีปริมาณข้อมูลจำนวนมาก และมีความซับซ้อน จะใช้เวลาในการสอบถามข้อมูลเป็นเวลานาน

จากปัญหาข้างต้น เทคนิคหนึ่งที่จะช่วยให้การสอบถามข้อมูลได้เร็วขึ้นคือ การสร้างตารางอินเด็กซ์ (indexing table) โดยจะต้องเลือกคอลัมน์ใน

ตารางสำหรับการสร้างอินเด็กซ์ ในตารางอินเด็กซ์ที่สร้างขึ้นจะมีคอลัมน์ที่เลือกมาสร้างอินเด็กซ์ซึ่งมีการเรียงลำดับข้อมูล และคอลัมน์ ROWID เก็บที่อยู่สำหรับการชี้ไปยังตารางฐานข้อมูล ด้วยวิธีการนี้จะทำให้การสอบถามข้อมูลได้รวดเร็วขึ้น อย่างไรก็ตามการสร้างตารางอินเด็กซ์จะต้องใช้พื้นที่จำนวนมากบนเครื่องคอมพิวเตอร์ และการสร้างตารางอินเด็กซ์ไม่เหมาะกับการสอบถามข้อมูลที่เป็นการเขียนข้อมูล คือ เพิ่ม ลบ และแก้ไข

ก่อนการสร้างตารางอินเด็กซ์ทุกครั้ง สิ่งที่ควรคำนึงถึงก็คือ การพิจารณาเลือกคอลัมน์สำหรับการสร้างอินเด็กซ์ เช่น การเลือกคอลัมน์ที่เป็น candidate column การเลือกคอลัมน์ที่มีค่า cardinality สูง การเลือกคอลัมน์ที่มีประเภทของข้อมูลขนาดเล็ก เป็นต้น

สุดท้ายในบทความนี้ยังได้ทำการทดสอบประสิทธิภาพการสอบถามของตารางอินเด็กซ์ โดยทดสอบค่า disk access time และค่า % CPU cost ของการการใช้และไม่ใช้อินเด็กซ์ในการสอบถามข้อมูล จากผลการทดสอบปรากฏว่าทั้งค่า disk access time และ % CPU cost จากการสร้างตารางอินเด็กซ์จะมีค่าที่ลดลง ดังนั้นเนื้อหาทั้งหมดของบทความนี้ทำให้การสร้างตารางอินเด็กซ์เป็นเทคนิคหนึ่งที่จะช่วยให้การสอบถามข้อมูลได้รวดเร็วยิ่งขึ้น

เอกสารอ้างอิง

ชาญชัย ศุภอรรถกร. (2553). แบบเรียนระบบฐานข้อมูล. (พิมพ์ครั้งที่ 1). กรุงเทพฯ: ซัคเซส มีเดีย: 1-7.

บริษัท ออราเคิล คอร์ปอเรชั่น (ประเทศไทย) จำกัด. (2554). Oracle Tuning Overview & Inefficient SQL. ในเอกสารการอบรมโครงการ Oracle Academy Program. บทที่ 2: 21.

BURLESON CONSULTING. (2011). Track CPU and I/O cost with Oracle9i. สืบค้นเมื่อ 14 เมษายน 2554, Available Source: http://www.dba-oracle.com/art_builder_cpu_io.htm

Chaudhuri, S. (2009). Query Optimizers: Time to Rethink the Contract?. In 35th SIGMOD International Conference on Management of Data. 961-968.

CODE FUNCTION.IN.TH. (2553). SQL SERVEL บทความสอนเกี่ยวกับคำสั่งของ Index. สืบค้นเมื่อ 14 เมษายน 2554, เข้าถึงได้จาก: http://code.function.in.th/sqlserver/indexes

DuBois, P. (2009). MySQL Developer's Library. (4th Ed.). United States of America: Addison Wesley. 303-311.

Honishi, T., Satoh, T. and Inoue, U. (1992). An Index Structure for Parallel Database Processing. In Second International Workshop on Data Engineering. 224-225.

Huang, Y.F. and Chen, J.M. (2000). The study of indexing techniques on object oriented databases. Journal Information Science 130: 109-131.

Li, D., Han, L. and Ding, Y. (2010). SQL Query Optimization Methods of Relational Database System. In Second International Conference on Computer Engineering and Applications (ICCEA). (V.1.): 557-560.

ORACLE DBA & ALL IT. (2554). เทคนิคการเลือก Index อย่างไรให้ work. สืบค้นเมื่อ 13 เมษายน 2554, เข้าถึงได้จาก: http://oracle.jookku.com/2011/03/tip-choose-index/

❑❑❑❑❑