การเพิ่มประสิทธิภาพดัชนีบิตแมปแบบคู่กันโดยใช้วิธีการเข้ารหัส

Enhancing Dual Bitmap Index with Efficient Encoding

ศิเนตร กิ้มเส้ง

Sinate Kimseng

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญา
วิทยาศาสตรมหาบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์
มหาวิทยาลัยสงขลานครินทร์

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Sciences
Prince of Songkla University
2557

ชื่อวิทยานิพนธ์ การเพิ่มประสิทธิภาพดัชนีบิตแมปแบบคู่กันโดยใช้วิธีการเข้ารหัส
ผู้เขียน นางสาวศิเนตร กิ้มเส้ง
สาขาวิชา วิทยาการคอมพิวเตอร์

อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

.....
(ผู้ช่วยศาสตราจารย์ ดร.ศิริรัตน์ วณิชโยบล)

คณะกรรมการสอบ

.....ประธานกรรมการ
(ดร.นพมาศ ปักเข็ม)

……....กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.ศิริรัตน์ วณิชโยบล)

……....กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.วิภาดา เวทย์ประสิทธิ์)

……....กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.ลัดดา ปรีชาวีรกุล)

บัณฑิตวิทยาลัย มหาวิทยาลัยสงขลานครินทร์ อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษา ตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์

……..……

(รองศาสตราจารย์ ดร.ธีระพล ศรีชนะ)
คณบดีบัณฑิตวิทยาลัย

ขอรับรองว่า ผลงานวิจัยนี้มาจากการศึกษาวิจัยของนักศึกษาเอง และได้แสดงความขอบคุณบุคคลที่มีส่วนช่วยเหลือแล้ว

ลงชื่อ……
(ผู้ช่วยศาสตราจารย์ ดร.ศิริรัตน์ วณิชโยบล)
อาจารย์ที่ปรึกษาวิทยานิพนธ์

ลงชื่อ……..
(นางสาวศิเนตร กิ้มเส้ง)
นักศึกษา

ข้าพเจ้าขอรับรองว่า ผลงานวิจัยนี้ไม่เคยเป็นส่วนหนึ่งในการอนุมัติปริญญาในระดับใดมาก่อน และไม่ได้ถูกใช้ในการยื่นขออนุมัติปริญญาในขณะนี้

ลงชื่อ……..
(นางสาวศิเนตร กิ้มเส้ง)
นักศึกษา

ชื่อวิทยานิพนธ์ การเพิ่มประสิทธิภาพดัชนีบิตแมปแบบคู่กันโดยใช้วิธีการเข้ารหัส
ผู้เขียน นางสาวศิเนตร กิ้มเส้ง
สาขาวิชา วิทยาการคอมพิวเตอร์
ปีการศึกษา 2556

บทคัดย่อ

คลังข้อมูลเป็นที่เก็บข้อมูลสำหรับสนับสนุนการตัดสินใจของผู้บริหาร ลักษณะการสอบถามที่เกิดขึ้นบ่อย ๆ ในคลังข้อมูลเป็นการสอบถามแบบทันทีทันใด ดัชนีบิตแมปเป็นเทคนิคที่นิยมใช้ค้นหาข้อมูลที่ต้องการ เนื่องจากการดำเนินการบูลีน (Boolean Operation) ที่มีประสิทธิภาพของการดำเนินการ AND OR และ NOT ในระดับบิต อย่างไรก็ตามดัชนีบิตแมปเหมาะสมสำหรับแอททริบิวต์ที่มีคาร์ดินอลิตี้ต่ำเท่านั้น

วิทยานิพนธ์นี้นำเสนอเทคนิคการเพิ่มประสิทธิภาพดัชนีบิตแมปแบบคู่กัน (Enhancing Dual Bitmap Index: EDBI)โดย EDBIใช้ความถี่ของการสอบถามในการกำหนดค่าให้กับแอททริบิวต์ การลงรหัสดัชนีใช้วิธีการเข้ารหัสทำให้ใช้พื้นที่จัดเก็บดัชนีน้อยกว่าดัชนีบิตแมปแบบคู่กันในกรณีที่แอททริบิวต์ที่มีค่าคาร์ดินอลิตี้สูง และยังคงรักษาประสิทธิภาพในการสอบถาม จากผลการวิเคราะห์และเปรียบเทียบระหว่าง EDBI กับดัชนีบิตแมปแบบคู่กันและดัชนีบิตแมปแบบเข้ารหัส พบว่า EDBI มีประสิทธิภาพเหนือกว่าดัชนีบิตแมปแบบคู่กันและดัชนีบิตแมปแบบเข้ารหัสในแง่ของการแลกเปลี่ยนรหว่างพื้นที่กับเวลา (Space-Time Trade-off)

Thesis Title Enhancing Dual Bitmap Index with Efficient Encoding

Author Miss Sinate Kimseng

Major Program Computer Science

Academic Year 2013

ABSTRACT

A data warehouse is a data repository for supporting decision makers. Most of queries against the data warehouse are ad hoc queries. Bitmap Index is a popular technique used to retrieve target records because of the efficient Boolean operation AND, OR and NOT. However, Bitmap Index is suitable for only low cardinality attributes.

This thesis proposes a technique called Enhancing Dual Bitmap Index (EDBI). EDBI makes use of the frequent query occurring attributes in assigning values. It also uses binary to encode, so it uses much less space than Dual Bitmap Index in high cardinality attributes while maintaining query processing time performance. The comparative study between EDBI, Dual Bitmap Index and Encoded Bitmap Index shows that EDBI is more efficient than the others in term of space-time trade-off perspective.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จได้ด้วยความช่วยเหลือและสนับสนุนจากบุคคลหลายฝ่าย ซึ่งผู้วิจัยรู้สึกซาบซึ้งและขอกราบขอบพระคุณอย่างสูง คือ

ผู้ช่วยศาสตราจารย์ ดร. ศิริรัตน์ วณิชโยบล อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่กรุณาให้คำปรึกษาแนะนำ และช่วยเหลือในการแก้ปัญหาต่างๆ ให้แก่ผู้วิจัยเสมอมา พร้อมทั้งตรวจทานและแก้ไขวิทยานิพนธ์ให้แก่ผู้วิจัย

ดร.นพมาศ ปักเข็ม ประธานกรรมการสอบวิทยานิพนธ์ ที่กรุณาช่วยตรวจทานและแก้ไขวิทยานิพนธ์ให้มีความสมบูรณ์

ผู้ช่วยศาสตราจารย์ ดร. ลัดดา ปรีชาวีรกุล กรรมการในการสอบวิทยานิพนธ์ที่กรุณาให้ความรู้และข้อเสนอแนะในการทำวิจัย รวมทั้งตรวจทานแก้ไขวิทยานิพนธ์

ผู้ช่วยศาสตราจารย์ ดร.วิภาดา เวทย์ประสิทธิ์ กรรมการในการสอบวิทยานิพนธ์ที่กรุณาให้ความรู้และข้อเสนอแนะในการทำวิจัย รวมทั้งตรวจทานแก้ไขวิทยานิพนธ์

อาจารย์ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัย สงขลานครินทร์ทุกท่านที่ให้ความรู้ทางด้านวิชาการ ซึ่งสามารถนำความรู้นี้มาใช้ในการทำวิทยานิพนธ์

เจ้าหน้าที่ภาควิชาวิทยาการคอมพิวเตอร์ และเจ้าหน้าที่บัณฑิตวิทยาลัยทุกท่านที่ให้ความช่วยเหลือ และอำนวยความสะดวกเกี่ยวกับเอกสารต่างๆ

เพื่อนๆ พี่ๆ และน้องๆ ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ ที่ให้คำปรึกษา และช่วยเหลือในการทำวิทยานิพนธ์

คุณพ่อ คุณแม่ และทุกคนในครอบครัว ที่ให้การสนับสนุนคอยเป็นห่วงสุขภาพและให้กำลังใจแก่ผู้วิจัยมาโดยตลอด

ผู้วิจัยขอขอบคุณทุกท่านเป็นอย่างสูง ณ โอกาสนี้

ศิเนตร กิ้มเส้ง

สารบัญ

สารบัญ(ต่อ)

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญตาราง

สารบัญภาพประกอบ

ภาพประกอบ **หน้า**

สารบัญภาพประกอบ(ต่อ)

ภาพประกอบ **หน้า**

บทที่ 1

บทนำ

1.1 ความเป็นมา

คลังข้อมูล (Data Warehouse) เป็นที่ใช้เก็บรวบรวมข้อมูลให้อยู่ในที่เดียวกัน มีการจัดโครงสร้างตามเนื้อหาที่เราสนใจ (Inmon, 1996) โดยข้อมูลที่เก็บรวบรวมนั้นจะเป็นข้อมูลตั้งแต่อดีตจนถึงปัจจุบันเพื่อสนับสนุนการตัดสินใจ (Rainardi, 2008) ในระบบคลังข้อมูลการประมวลผลข้อมูลจะเป็นแบบ OLAP (Online Analytical Processing) (Chaudhuri and Dayal, 1997) และแบบทันทีทันใด (Ad Hoc) เพื่อเพิ่มความเร็วในการสอบถาม และการสอบถามส่วนใหญ่ซับซ้อนทำให้การประมวลผลข้อมูลต้องใช้เวลานาน จากปัญหาที่กล่าวมาจึงมีการทำงานวิจัยที่เกี่ยวข้องกับคลังข้อมูลมากมายส่วนใหญ่แล้วจะเน้นสองเรื่องหลัก คือ เรื่องของพื้นที่ที่ใช้และเรื่องของเวลาในการเข้าถึง เมื่อกล่าวถึงเวลาที่ใช้เข้าถึงข้อมูล วิธีการเพิ่มความเร็วในการเข้าถึงข้อมูลมีหลายวิธีด้วยกันยกตัวอย่างเช่น วิธีการแฮชชิ่ง (Hashing) เหมาะกับการเข้าถึงโดยตรง วิธีการประมวลผลแบบขนาน (Parallel) เป็นการเพิ่มฮาร์ดแวร์ (Hardware) เพื่อให้ทำงานเร็วขึ้น และวิธีการทำดัชนี (Chan and Ioannidis, 1998) เป็นวิธีการเพิ่มความเร็วในการเข้าถึงข้อมูลโดยไม่ต้องเสียค่าใช้จ่ายใดๆแต่เป็นการใช้พื้นที่แลกกับเวลาในการสอบถามที่เร็วขึ้น

การทำดัชนีมีหลายแบบ เช่น B-Tree, B+ Tree และดัชนีบิตแมป ในวิทยานิพนธ์นี้จะกล่าวถึงเฉพาะดัชนีบิตแมปเพราะดัชนีบิตแมปมีความนิยมในการใช้ในการประมวลผลแบบทันที ทันใด (Inmon, 1996; Chaudhuri and Dayal, 1997; Chan and Ioannidis, 1998) เพราะสามารถดำเนินการระดับบิต (AND, OR, XOR, NOT) ก่อนจะเข้าถึงข้อมูลจริง

ในงานวิจัยนี้จะแบ่งดัชนีบิตแมปเป็น 2 กลุ่มคือ กลุ่มที่หนึ่งคือกลุ่มที่ใช้วิธีการบีบอัด (Bitmap Index Compression) (Wu *et al*., 2006) เช่น WAH (Word – Aligned Hybrid) (Deliège and Pedersen, 2010) และ RLH (Run – Length Huffman) (Stabno and R. Wrembel, 2009) เป็นต้น ข้อดีของดัชนีบิตแมปในกลุ่มที่ใช้วิธีการบีบอัด คือ ใช้พื้นที่จัดเก็บดัชนีน้อยมาก แต่ส่วนข้อเสียคือถ้าเราบีบอัดดัชนีเพื่อลดขนาดพื้นที่จัดเก็บแล้ว เมื่อต้องการสอบถามจะต้องขยายดัชนีที่โดนบีบอัดให้กลับมาอยู่ในรูปแบบเดิมก่อนจึงจะทำการสอบถาม (Query) ได้ และกลุ่มที่สองคือกลุ่มที่ใช้วิธีการลงรหัส (Bitmap Index Extension) เช่น ดัชนีบิตแมปแบบพื้นฐาน (Simple Bitmap Index) (O'Neil and Quass, 1997) ดัชนีบิตแมปแบบช่วง (Interval Bitmap Index) (Chan and Ioannidis, 1999) ดัชนีบิตแมปแบบกระจาย (Scatter Bitmap Index) (Vanichayobon *et al*., 2006) ดัชนีบิตแมป

แบบเข้ารหัส (Encoded Bitmap Index) (Wu and Buchmann, 1998; Keawpibal *et al*., 2012) และดัชนีบิตแมปแบบคู่กัน (Dual Bitmap Index) (Wattanakitrungroj and S. Vanichayobon, 2006) ซึ่งข้อดีของดัชนีบิตแมปในกลุ่มที่ใช้วิธีการลงรหัสนี้ คือเมื่อต้องการสอบถาม สามารถทำการสอบถามได้ ทันทีโดยไม่จำเป็นต้องมีการเปลี่ยนแปลงรูปแบบของดัชนีอีก และใช้พื้นที่ในการจัดเก็บดัชนีได้มีประสิทธิภาพ

จากการศึกษางานวิจัยที่ผ่านมาผู้วิจัยสนใจการทำดัชนีบิตแมปกลุ่มลงรหัส และดัชนีบิตแมปที่ใช้ในการทำวิจัยในวิทยานิพนธ์ครั้งนี้ คือ ดัชนีบิตแมปแบบคู่กัน เนื่องจากที่ผ่านมาดัชนีบิตแมปแบบคู่กันมีข้อดีในเรื่องของเวลาในการสอบถาม แต่ยังมีข้อบกพร่องที่สามารถปรับปรุงได้ในเรื่องของพื้นที่ ที่ใช้จัดเก็บดัชนี ผู้วิจัยจึงคิดค้นวิธีการเพิ่มประสิทธิภาพให้กับดัชนีบิตแมปแบบคู่กันขึ้น โดยเล็งเห็นข้อดีของดัชนีบิตแมปแบบคู่กันคือหนึ่งแอททริบิวต์จะแทนด้วย 2 บิตแมปเวกเตอร์ ส่วนข้อเสียของดัชนีบิตแมปแบบคู่กันคือขนาดพื้นที่จัดเก็บดัชนีบิตแมปยังใช้พื้นที่มาก ดังนั้นผู้วิจัยจึงนำเสนอวิธีการ เพิ่มประสิทธิภาพดัชนีบิตแมปแบบคู่กันเพื่อลดพื้นที่ในการจัดเก็บดัชนีให้น้อยลงโดยใช้วิธีการเข้ารหัส และใช้เวลาการสอบถามที่เท่ากันในบางกรณี

1.2 งานวิจัยที่เกี่ยวข้อง

An Overview of Data Warehousing and OLAP Technology

งานวิจัย (Chaudhuri and Dayal, 1997) นี้กล่าวถึงเทคโนโลยีคลังข้อมูล และอธิบายความแตกต่างของคลังข้อมูลกับฐานข้อมูล รวมไปถึงได้อธิบายโครงสร้างสถาปัตยกรรมของคลังข้อมูล เครื่องมือในการออกแบบ เครื่องมือในการสอบถาม และวิธีการออกแบบคลังข้อมูล การเพิ่มประสิทธิภาพการทำงานของคลังข้อมูล และการจัดการคลังข้อมูล

Bitmap Index Design and Evaluation

งานวิจัย (O'Neil and Quass, 1997) นี้กล่าวถึงการทำดัชนีบิตแมปแบบพื้นฐาน เป็นดัชนีบิตแมปที่เหมาะกับแอททริบิวต์ (A) ที่มีคาร์ดินอลิตี้ (C) ต่ำเพราะบิตแมปเวกเตอร์ที่ใช้จะเท่ากับจำนวนของคาร์ดินอลิตี้ $(A=C)$ จึงใช้พื้นที่ในการจัดเก็บดัชนีมากแต่การสอบถามข้อมูลจะทำได้เร็วเพราะสอบถามข้อมูล 1 บิตแมปเวกเตอร์เท่านั้น

Encoded Bitmap Indexing for Data Warehouse

งานวิจัย (Wu and Buchmann, 1998) กล่าวถึงการทำดัชนีบิตแมปแบบเข้ารหัสเป็นดัชนีบิตแมปที่ใช้พื้นที่น้อยที่สุดคือ $\lceil \log_2 C \rceil$ แต่การสอบถามข้อมูลแบบค่าเท่ากันจะดำเนินการกับทุกบิตแมปเวกเตอร์และมีการใช้ตารางเทียบค่าในการลงรหัสของดัชนีบิตแมป

An Efficient Bitmap Encoding Scheme for Selection Queries

งานวิจัยนี้ (Chan and Ioannidis, 1999) กล่าวถึงการทำดัชนีบิตแมปแบบช่วง เป็นดัชนีบิตแมปที่ใช้พื้นที่ได้มีประสิทธิภาพกว่าดัชนีบิตแมปแบบพื้นฐานเพราะใช้บิตแมปเวกเตอร์ครึ่งหนึ่งของดัชนีบิตแมปแบบพื้นฐาน $\lceil C/2 \rceil$ แต่ในทางกลับกันการสอบถามข้อมูลแบบค่าเท่ากันใช้การดำเนินการระหว่าง 2 บิตแมปเวกเตอร์ในกรณีที่สร้างดัชนีบนแอททริบิวต์ที่มีค่าคาร์ดินอลิตี้มากกว่า 3 ทำให้เวลาที่ใช้ในการสอบถามนานกว่าดัชนีบิตแมปแบบพื้นฐาน

Scatter Bitmap: Space-Time Efficient Bitmap Indexing for Equality and Membership Queries

งานวิจัย (Vanichayobon *et al*., 2006) กล่าวถึงการทำดัชนีบิตแมปแบบกระจายใช้พื้นที่ในการจัดเก็บดัชนีน้อยกว่า ดัชนีบิตแมปแบบพื้นฐาน และ ดัชนีบิตแมปแบบช่วงโดยใช้บิตแมปเวกเตอร์เท่ากับ $\lceil 2\sqrt{C} \rceil$ แต่การสอบถามข้อมูลแบบค่าเท่ากันจะดำเนินการกับ 2 บิตแมปเวกเตอร์ เช่นเดียวกับดัชนีบิตแมปแบบช่วง

Dual Bitmap Index: Space-Time Efficient Bitmap Index for Equality and Membership Queries

งานวิจัย (Wattanakitrungroj and S. Vanichayobon, 2006) นี้กล่าวถึงการทำดัชนีบิตแมปแบบคู่กัน ถ้าคาร์ดินอลิตี้เท่ากับ C จะใช้บิตแมปเวกเตอร์เท่ากับ $\lceil \sqrt{2C + 0.25} + 0.5 \rceil$ ซึ่งใช้พื้นที่น้อยกว่าดัชนีบิตแมปแบบช่วง และดัชนีบิตแมปแบบกระจาย และการสอบถามข้อมูลแบบค่าเท่ากับจะดำเนินการ AND ระหว่าง 2 บิตแมปเวกเตอร์ เหมือนกับดัชนีบิตแมปแบบกระจาย

Enhanced Encoded Bitmap Index for Equality Query

งานวิจัย (Keawpibal *et al.*, 2012) กล่าวถึงการเพิ่มประสิทธิภาพการสอบถามข้อมูลแบบค่าเท่ากันของดัชนีบิตแมปแบบเข้ารหัสโดยไม่จำเป็นต้องใช้ตารางเทียบค่า

1.3 วัตถุประสงค์

เพื่อเพิ่มประสิทธิภาพดัชนีบิตแมปแบบคู่กันโดยใช้วิธีการเข้ารหัสและลดพื้นที่ที่ใช้ในการจัดเก็บดัชนีแต่ยังคงรักษาประสิทธิภาพในเรื่องของเวลาในการสอบถาม

1.4 วิธีการดำเนินการวิจัย

1. ศึกษาแนวคิดของคลังข้อมูล การบีบอัดดัชนีบิตแมป และดัชนีบิตแมป 5 แบบคือ ดัชนีบิตแมปแบบพื้นฐาน ดัชนีบิตแมปแบบช่วง ดัชนีบิตแมปกระจาย ดัชนีบิตแมปแบบคู่กัน และดัชนีบิตแมปแบบเข้ารหัส
2. วิเคราะห์และออกแบบเทคนิควิธีการเพิ่มประสิทธิภาพการลงรหัสดัชนีบิตแมปแบบคู่กันโดยใช้วิธีการเข้ารหัสเพื่อลดพื้นที่การจัดเก็บดัชนี และออกแบบวิธีการสอบถามแบบค่าเท่ากันบนดัชนีบิตแมปที่ออกแบบไว้
3. กำหนดรูปแบบและวิธีการประเมินประสิทธิภาพของดัชนีบิตแมปแบบคู่กันที่ออกแบบวิธีการลงรหัสแบบใหม่ กับดัชนีบิตแมปแบบคู่กันที่มีอยู่เดิม โดยทำการประเมินทั้งในด้านพื้นที่ที่ใช้ในการจัดเก็บดัชนี และเวลาที่ใช้ในการสอบถามข้อมูลแบบค่าเท่ากัน ดังขั้นตอนต่อไปนี้
   3.1 ศึกษาข้อมูลที่ใช้ในการทดลองซึ่งเป็นข้อมูลมาตรฐาน TCP-H Benchmark (Transaction Processing Performance Council, 2013)
   3.2 เตรียมข้อมูลที่ใช้ทดสอบโดย เลือกแอททริบิวต์ที่จะนำมาทำดัชนี ทำการเปลี่ยนค่าข้อมูลที่อยู่ในแอททริบิวต์นั้นให้เป็นตัวเลขจำนวนเต็มและเรียงต่อกัน
   3.3 ออกแบบขั้นตอนการสร้างดัชนีบิตแมปแบบคู่กับแบบดั้งเดิม และออกแบบขั้นตอนการสร้างดัชนีบิตแมปแบบคู่กันที่ใช้วิธีการเข้ารหัส
   3.4 พัฒนาโปรแกรมเพื่อสร้างดัชนีบิตแมปทั้งสองตามขั้นตอนที่ได้ออกแบบไว้ในข้อที่ 3.3 โดยใช้ตัวแปลภาษาซี (C Compiler)
4. ออกแบบขั้นตอนวิธีในการสอบถามข้อมูลแบบค่าเท่ากันบนดัชนีบิตแมปแบบคู่กับ EDBI และดัชนีบิตแมปแบบเข้ารหัส

5. พัฒนาโปรแกรมเพื่อสอบถามข้อมูลแบบค่าเท่ากันบนดัชนีบิตแมปทั้งสามตามที่ได้ออกแบบไว้
6. ประเมินประสิทธิภาพการใช้พื้นที่ (Space Requirement) ของดัชนีบิตแมปที่ทำการพัฒนาทั้งสามแบบ
7. ประเมินประสิทธิภาพการสอบถาม (Query Time) ข้อมูลแบบค่าเท่ากันบนดัชนีบิตแมปทั้งสามแบบ
8. ประเมินประสิทธิภาพการแลกเปลี่ยนระหว่างพื้นที่กับเวลา (Space – Time Trade off) บนดัชนีบิตแมปทั้งสามแบบ
9. วิเคราะห์และสรุปผลการประเมินประสิทธิภาพของ EDBI ดัชนีบิตแมปแบบคู่กันและดัชนีบิตแมปแบบเข้ารหัส

1.5 ขอบเขตการวิจัย

1. ศึกษาดัชนีบิตแมปที่มีอยู่ได้แก่ ดัชนีบิตแมปแบบพื้นฐาน ดัชนีบิตแมปแบบช่วง ดัชนีบิตแมปแบบกระจาย ดัชนีบิตแมปแบบเข้ารหัส และดัชนีบิตแมปแบบคู่กัน
2. วิเคราะห์และออกแบบการวิธีการเพิ่มประสิทธิภาพดัชนีบิตแมปแบบคู่กันสำหรับการสอบถามแบบค่าเท่ากันและการสอบถามแบบเป็นสมาชิก

1.6 ขั้นตอนการดำเนินการ

1. ศึกษางานวิจัยและเอกสารที่เกี่ยวข้อง
2. ศึกษาและวิเคราะห์เครื่องมือที่เกี่ยวกับงานวิจัย
3. กำหนดขอบเขตของปัญหางานวิจัย
4. วิเคราะห์และออกแบบวิธีการเพิ่มประสิทธิภาพให้กับดัชนีบิตแมปแบบคู่กัน
5. พัฒนาดัชนีบิตแมปตามที่ออกแบบไว้
6. ประเมินประสิทธิภาพดัชนีบิตแมปที่ทำการพัฒนา
7. สรุปผลและวิเคราะห์ผล
8. จัดทำเอกสารวิทยานิพนธ์

1.7 ระยะเวลาการดำเนินงานและแผนการดำเนินงาน

พฤษภาคม พ.ศ.2556 - มีนาคม พ.ศ.2557

กิจกรรม/ขั้นตอนการดำเนินงาน	เดือน										
	พ.ศ. 2556								พ.ศ. 2557		
	5	6	7	8	9	10	11	12	1	2	3
ศึกษางานวิจัยและเอกสารที่เกี่ยวข้อง	←	—	—	—	—	—	—	—	—	—	→
ศึกษาเทคโนโลยีและเครื่องมือสำหรับงานวิจัย		←	—	—	—	—	—	→			
วิเคราะห์และออกแบบกระบวนการในการทำงาน					←	—	—	—	→		
พัฒนาและทดสอบกระบวนการที่ออกแบบ							←	—	—	—	→
เขียนบทความวิจัย									←	—	→
จัดทำเอกสารวิทยานิพนธ์								←	—	—	→

1.8 เครื่องมือและอุปกรณ์

ด้านฮาร์ดแวร์

1. เครื่องคอมพิวเตอร์ส่วนบุคคล จำนวน 1 เครื่อง
2. CPU : Intel Core2Duo 2.4 GHz
3. Harddisk : 320 GB
4. RAM : 4GB
5. เครื่องพิมพ์ 1 เครื่อง

ด้านซอฟต์แวร์

1. ระบบปฏิบัติการ Microsoft Windows 7 Ultimate
2. ตัวแปลภาษาซี (C Complier)

1.9 ประโยชน์ที่คาดว่าจะได้รับ

ได้ดัชนีบิตแมปแบบใหม่ที่ใช้พื้นที่ (Space) ในการจัดเก็บดัชนีอย่างคุ้มค่าและเวลา (Time) ในการสอบถามที่มีประสิทธิภาพ

บทที่ 2

ทฤษฎีที่เกี่ยวข้อง

ในบทนี้กล่าวถึงคลังข้อมูล ความหมายของคลังข้อมูล คุณลักษณะและสถาปัตยกรรมคลังข้อมูลรวมถึงความแตกต่างระหว่างฐานข้อมูลปฏิบัติงานกับคลังข้อมูล การสร้างดัชนี (Index) และเทคนิคการสร้างดัชนีบิตแมป

2.1 คลังข้อมูล (Data warehouse)

2.1.1 ความหมายของคลังข้อมูล

Data Warehouse (Chaudhuri and Dayal, 1997; Kimball, 2006) เป็นที่เก็บรวบรวมข้อมูลที่มีขนาดใหญ่ซึ่งข้อมูลเหล่านั้นจะมีที่มาจากหลาย ๆ แหล่ง เช่นจากฐานข้อมูลปฏิบัติการ (Operational Database) จากไฟล์ (file) เป็นต้น ซึ่งคลังข้อมูลจะมีคุณสมบัติแบบ Subject-oriented, Integrated, Time-variant และ Non-volatile สำหรับการเข้าถึงข้อมูลในคลังข้อมูลจะมีการประมวลผลเป็นแบบ On-line Analytical Processing หรือที่เรียกว่า OLAP ซึ่งเป็นการประมวลผลในเชิงวิเคราะห์ และการเข้าถึงข้อมูลแบบทันที (Ad Hoc) คลังข้อมูลใช้เพื่อ เพิ่มประสิทธิภาพในระบบการสนับสนุนการตัดสินใจของผู้บริหาร (Decision Support System) และเป็นเครื่องมือที่ใช้ในการวิเคราะห์ข้อมูลในมุมมองหลายมิติ (Multi-Dimensional)

1.1.2 คุณลักษณะของคลังข้อมูล (Inmon, 1996)

- Subject-oriented หมายถึง คลังข้อมูลถูกออกแบบมาเพื่อจัดโครงสร้างตามเนื้อหาที่สนใจ เช่น ลูกค้า สินค้า การขายสินค้า ไม่ได้เน้นไปที่การทำงานหรือกระบวนการอย่างใดอย่างหนึ่งเหมือนฐานข้อมูลปฏิบัติการ ในส่วนของรายละเอียดการจัดเก็บข้อมูลจะแตกต่างกันตามความต้องการใช้งาน เช่น คลังข้อมูลจะเก็บข้อมูลที่มีส่วนเกี่ยวข้องกับการประมวลผลเพื่อสนับสนุนการตัดสินใจ การวิเคราะห์ข้อมูล และการทำเหมืองข้อมูล (Data Mining)
- Integrated หมายถึง ข้อมูลในคลังข้อมูลถูกรวบรวมมาจากหลาย ๆ แหล่ง ซึ่งข้อมูลอาจมาจากระบบที่ต่างกัน จึงมีรูปแบบการเก็บข้อมูลที่ต่างกัน หรืออยู่บนสถาปัตยกรรมที่แตกต่างกันด้วย ดังนั้นจึงต้องแปลงข้อมูลให้เป็นมาตรฐานเดียวกัน และมีความสอดคล้องก่อน จึงจะสามารถโหลดข้อมูลนั้นเข้าสู่คลังข้อมูลได้
- Time-variant หมายถึง ข้อมูลที่จัดเก็บในคลังข้อมูลจะมีการกำหนดช่วงเวลาเอาไว้เพราะในระบบการสนับสนุนการตัดสินใจจำเป็นต้องมีการเปรียบเทียบข้อมูลที่อยู่

คนละช่วงเวลา โดยข้อมูลที่เก็บในคลังข้อมูลจะเก็บข้อมูลนั้นจะเก็บย้อนไปในอดีต 5-10 ปี จนถึงปัจจุบัน

- Non-volatile หมายถึง ข้อมูลที่เก็บอยู่ในคลังข้อมูลจะไม่มีการเปลี่ยนแปลง หรือถ้ามีการเปลี่ยนแปลงก็จะเปลี่ยนแปลงน้อยมาก การดำเนินการกับคลังข้อมูลโดยส่วนใหญ่จะเป็นการโหลดข้อมูลเข้าสู่คลังข้อมูลและการสอบถามข้อมูลจากคลังข้อมูล

2.1.3 ความแตกต่างระหว่างฐานข้อมูลกับคลังข้อมูล

โดยทั่วไปฐานข้อมูลจะมีลักษณะทันต่อเหตุการณ์และถูกสร้างมาเพื่อทำงานอย่างใดอย่างหนึ่ง และข้อมูลนั้นจะมีการอัพเดทอยู่เสมอ เช่น ฐานข้อมูลพนักงานก็จะเก็บเฉพาะพนักงานในปัจจุบัน การลงทะเบียน การฝาก-ถอนเงินในธนาคาร เป็นต้น การประมวลผลของระบบฐานข้อมูลจะทำการประมวลผลทรานแซคชันและประมวลผลแบบออนไลน์เรียกว่า On-line Transaction Processing (OLTP) ส่วนระบบคลังข้อมูลถ้าเป็นคลังข้อมูลของบริษัทก็จะเก็บรวบรวมข้อมูลทุกส่วนของบริษัท ทั้งข้อมูลเก่าและใหม่ไว้ด้วยกันไม่มีการลบข้อมูลทิ้งซึ่งข้อมูลเหล่านี้จะมีประโยชน์ต่อผู้บริหารในการวิเคราะห์ประสิทธิภาพและคุณลักษณะต่าง ๆ รวมถึงแนวโน้มของบริษัทหรือขององค์กรนั้น ใช้การประมวลผลแบบ OLAP ซึ่งได้เปรียบเทียบความแตกต่างของคลังข้อมูลและฐานข้อมูลปฏิบัติการไว้ในตาราง 2-1

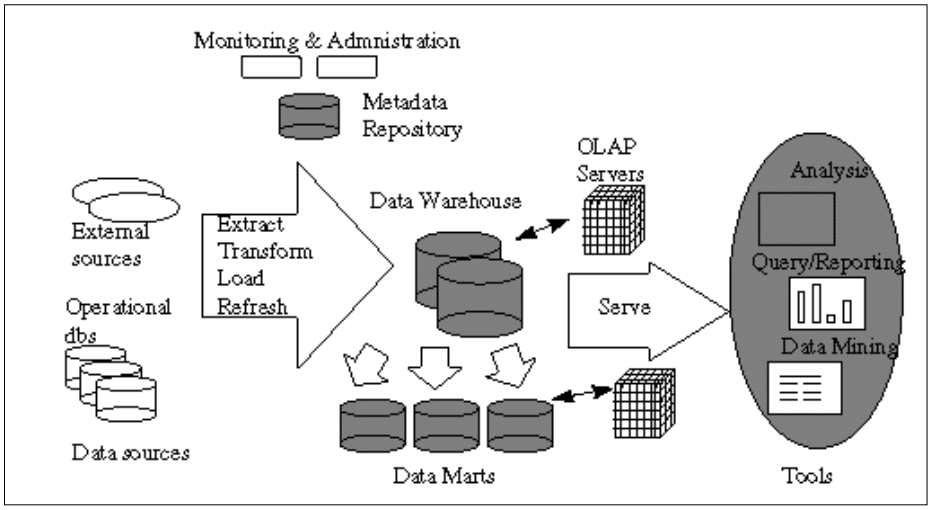
ตาราง 2-1 การเปรียบเทียบฐานข้อมูลกับคลังข้อมูล

ลักษณะ	ฐานข้อมูล	คลังข้อมูล
การประมวลผล	แบบ OLTP	แบบ OLAP
กลุ่มผู้ใช้งาน	พนักงาน ผู้ดูแลระบบ	ผู้บริหาร นักวิเคราะห์ ผู้จัดการ
ลักษณะของข้อมูล	เป็นข้อมูลที่มีความทันสมัย มีการอัพเดทตลอดเวลา	เป็นข้อมูลตั้งแต่อดีตจนถึงปัจจุบัน ข้อมูลที่เก็บไว้แล้วจะไม่มีการเปลี่ยนแปลง
การทำงาน	เป็นการทำงานซ้ำๆใช้เวลาน้อย	เน้นการวิเคราะห์ข้อมูล เวลาที่ใช้ในการประมวลผลนาน

ลักษณะ	ฐานข้อมูล	คลังข้อมูล
การเข้าถึงข้อมูล	เพิ่ม (Insert) แก้ไข (Update) ลบ (Delete)	การเข้าถึงส่วนใหญ่จะเป็นการอ่าน ไม่มีการลบข้อมูลในคลังข้อมูล
ขนาดของฐานข้อมูล	ประมาณ 100 MB – 1 GB	ประมาณ 100 GB – 1 TB
การออกแบบฐานข้อมูล	ใช้แบบจำลองอี-อาร์ (E-R Model)	ใช้แบบจำลองหลายมิติ (Multidimensional Model)
คำสั่งในการสอบถาม	เป็นคำสั่งสอบถามสั้น ๆ มีการเตรียมไว้ล่วงหน้า	เป็นการสอบถามแบบทันที ไม่มีการเตรียมไว้ล่วงหน้าว่าจะสอบถามอะไร
ประสิทธิภาพในการทำงาน	ปริมาณทรานแซคชัน	เวลาในการตอบสนอง
สรุป	ต้องการรายละเอียดและความแม่นยำของข้อมูล	ต้องการข้อมูลสรุปและแสดงผลได้อย่างรวดเร็วเพื่อสนับสนุนการตัดสินใจ

2.1.4 สถาปัตยกรรมคลังข้อมูล

สถาปัตยกรรมคลังข้อมูล (Data Warehouse Architecture: DWA) เป็นโครงสร้างมาตรฐานที่ใช้กันอย่างแพร่หลาย โดยคลังข้อมูลแต่ละระบบจะมีรูปแบบที่แตกต่างกันได้เพื่อความเหมาะสมของการใช้งานในองค์กรนั้น ๆ สำหรับส่วนประกอบของสถาปัตยกรรมคลังข้อมูลที่สำคัญได้แก่ แหล่งข้อมูล (Data source) หน่วยเก็บข้อมูล (Data Storage) OLAP Server และเครื่องมือสำหรับผู้ใช้ (Front-End Tool) แสดงดังภาพประกอบ 2-1



ภาพประกอบ 2- 1 สถาปัตยกรรมคลังข้อมูล (Chaudhuri and Dayal, 1997)

จากภาพที่ 2-1 สถาปัตยกรรมคลังข้อมูล ประกอบด้วยส่วนประกอบของสถาปัตยกรรมดังนี้

- แหล่งข้อมูล (Data Sources) แหล่งที่มาของข้อมูลที่นำมาใช้ในคลังข้อมูลมาจากแหล่งข้อมูลภายนอก (External Sources) และฐานข้อมูล (Operational Database) รวมถึงไฟล์ข้อมูลทั่วไป ซึ่งอาจมีรูปแบบของข้อมูลที่แตกต่างกัน และเนื่องจากข้อมูลมาจากหลายแหล่งที่มาจึงจำเป็นต้องมีการ สกัดข้อมูล และแปลงข้อมูลให้อยู่ในรูปแบบเดียวกันก่อนจะโหลดข้อมูลเข้าสู่หน่วยเก็บข้อมูล

- หน่วยเก็บข้อมูล (Data Storage) ประกอบด้วยคลังข้อมูล (Data Warehouse) คลังข้อมูลย่อย (Data Marts) และหน่วยจัดการความรู้ (Metadata Repository) ข้อมูลในคลังข้อมูลและคลังข้อมูลย่อยจะมีความสัมพันธ์กันเช่น ถ้าเป็นคลังข้อมูลของมหาวิทยาลัย คลังข้อมูลย่อยก็จะเป็นที่เก็บข้อมูลของคณะในมหาวิทยาลัยนั้น หน่วยจัดการความรู้มีหน้าที่จัดเก็บข้อมูลเกี่ยวกับเครื่องมือที่ใช้ในการควบคุมคลังข้อมูล เก็บที่มาของข้อมูล รูปแบบของข้อมูล เวลาที่ปรับปรุงข้อมูลล่าสุด และข้อจำกัดของข้อมูลในคลังข้อมูลเป็นต้น

- OLAP Server ทำหน้าที่ในการจัดเตรียมการเข้าถึงข้อมูลเพื่อเพิ่มประสิทธิภาพการประมวลผลมีความรวดเร็วมากขึ้น ได้แก่ การทำดัชนี การสร้าง Materialize View การเปลี่ยนรูปแบบการสอบถามที่ซับซ้อน และการประมวลผลแบบขนาน (Parallel Processing) เพื่อลดเวลาในการสอบถาม

- เครื่องมือสำหรับผู้ใช้ (Front-End Tool) ประกอบด้วยเครื่องมือในการวิเคราะห์ข้อมูล (Analysis) เครื่องมือสำหรับการสอบถาม เครื่องมือสำหรับการออกรายงาน (Report) และเครื่องมือสำหรับการทำเหมืองข้อมูล

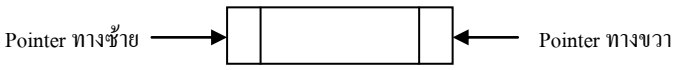
ขั้นตอนการสร้างคลังข้อมูลเริ่มจากดึงข้อมูล (Extraction) จากแหล่งข้อมูลต่างๆ จากนั้นเป็นการทำความสะอาดข้อมูล (Data Cleaning) และเปลี่ยนรูปแบบข้อมูลให้อยู่ในรูปแบบเดียวกัน (Transforming) และให้มีความสอดคล้องกันเนื่องจากคลังข้อมูลมีการรวบรวมข้อมูลจากหลายแหล่งที่มาที่แตกต่างกัน ทำให้ข้อมูลมีโครงสร้างและรูปแบบที่แตกต่างกัน ดังนั้นจึงต้องมีการทำความสะอาดข้อมูลและเปลี่ยนให้อยู่ในรูปแบบเดียวกันเสียก่อนแล้วจึงทำการโหลดเข้าสู่คลังข้อมูล

2.2 การสร้างดัชนี

เป็นการใช้พื้นที่แลกกับเวลาในการสอบถาม เพื่อใช้ในการเพิ่มความเร็วในการเข้าถึงข้อมูลที่ต้องการโดยไม่จำเป็นต้องเสียค่าใช่จ่ายในการเพิ่มฮาร์ดแวร์ เพราะตารางดัชนีมีขนาดเล็กกว่าตารางข้อมูลในคลังข้อมูลมาก ยกตัวอย่างเช่น ดัชนี B-tree และดัชนีบิตแมป เป็นต้น

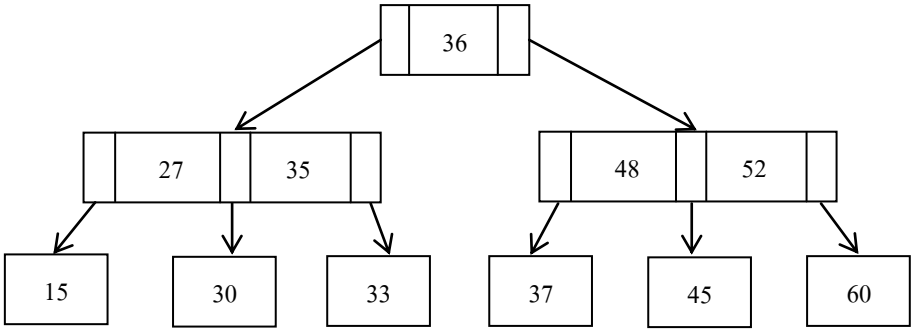
2.2.1 ดัชนี B-tree (Silberschatz, 2002)

เป็นดัชนีที่เก็บข้อมูลแบบโครงสร้างต้นไม้ประกอบด้วย โหนดบนสุดเรียกว่า Root Node และโหนดภายในเรียกว่า Internal Node จะมีคีย์ K คีย์ และจะมีลูกได้ K+1 โหนด และอย่างน้อยที่สุดจะสามารถมีโหนดลูกได้ (K+1)/2 โหนด ยกเว้น Root Node ที่ต้องมีโหนดลูกอย่างน้อย 2 โหนด และ ส่วนโหนดที่อยู่ล่างสุดเรียกว่า Leaf Node โดย Leaf Node ทั้งหมดจะอยู่ในระดับเดียวกัน คีย์ที่อยู่บนโหนดจะเรียงจากน้อยไปหามากและระหว่างคีย์จะมี Pointer เป็นตัวแบ่งช่วงค่าของคีย์ได้ N+1 ช่วง แสดงดังภาพประกอบ 2-2



ภาพประกอบ 2-2 Pointer ของดัชนี B-tree

ความสูงของดัชนีแบบ B-tree จะขึ้นอยู่กับจำนวนของคีย์ที่สามารถให้มีได้ในแต่ละโหนด ถ้าจำนวนคีย์ทั้งหมดคือ n และแต่ละโหนดมีคีย์ได้ K คีย์ ความสูงของดัชนีแบบ B-tree จะเท่ากับ $\log_K n$ เมื่อมีการค้นหาข้อมูลมาถึงโหนดที่มี K คีย์ จะมีทางเลือกในการค้นหาเท่ากับ K+1 ทาง ดังนั้นการค้นหาข้อมูลจึงใช้เวลาเป็น $O(log_K n)$ จากภาพประกอบ 2-3 โหนดลูกที่ถูกเชื่อมจาก Pointer ทางซ้ายของคีย์จะมีค่าน้อยกว่าหรือเท่ากับคีย์นั้น ส่วนโหนดลูกที่ถูกเชื่อมจาก Pointer ทางขวาของคีย์จะมีค่ามากกว่าหรือเท่ากับคีย์นั้น



ภาพประกอบ 2-3 ตัวอย่างดัชนีแบบ B-tree

1.1.2 **ข้อดีของดัชนี B-Tree**

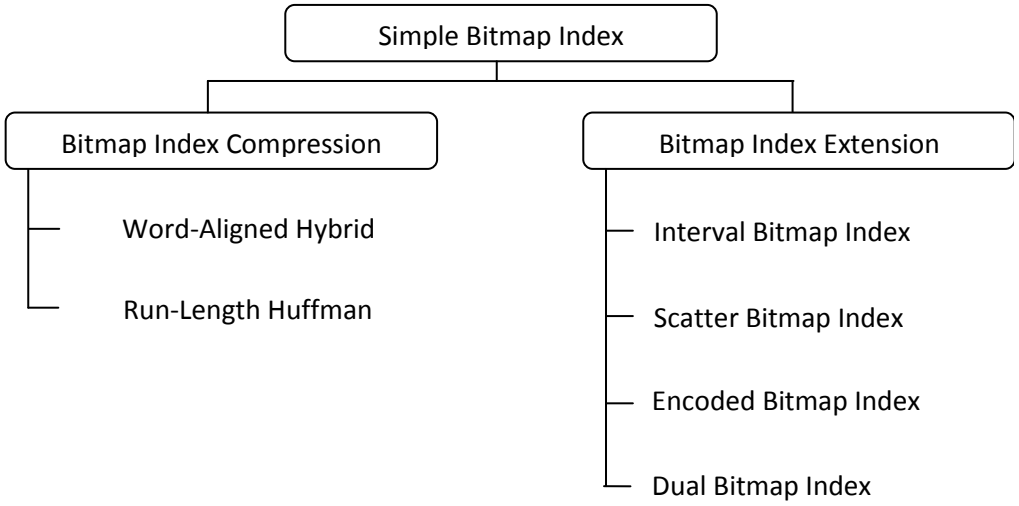
เป็นที่นิยมในโปรแกรมประยุกต์คลังข้อมูลสำหรับแอทริบิวต์ที่ค่ามีคาร์ดินอลิตี้สูง และบางครั้งการสอบถามจะได้คำตอบก่อนจะถึง Leaf Node

1.1.3 **ข้อเสียของดัชนี B-tree**

ไม่เหมาะกับแอทริบิวต์ที่มีค่าคาร์ดินอลิตี้ต่ำ และดัชนี B-Tree มีความเป็นอิสระถ้าหากต้องการสอบถามที่มีค่าข้อมูลมากกว่า 1 แอทริบิวต์จำเป็นต้องสร้างคีย์ผสมขึ้นทำให้ยุ่งยากและใช้เวลาในการสอบถามมากขึ้น

2.3 ดัชนีบิตแมป

ดัชนีบิตแมปเป็นวิธีการที่ดีสำหรับการประมวลผลแบบ Ad Hoc เพราะมีการดำเนินการระดับบิตกับตารางดัชนี (AND, OR, XOR, NOT) ก่อนที่จะเข้าถึงตารางข้อมูลจริง ทำให้มีประสิทธิภาพสูงในเรื่องของเวลาการประมวลผล (O'Neil and Quass, 1997; Stockinger and Wu 2006) จึงเหมาะที่จะนำมาใช้กับคลังข้อมูลเพราะคลังข้อมูลมีปริมาณของข้อมูลที่มากและการทำงานส่วนใหญ่จะเป็นการสอบถามข้อมูล ดังนั้นดัชนีบิตแมปเป็นเทคนิคที่ช่วยให้เราสามารถสอบถามข้อมูลในคลังข้อมูลได้เร็วขึ้นโดยในตารางดัชนีจะแทนด้วยบิต 0 และบิต 1 เท่านั้น ดัชนีบิตแมปแบบแรกจะเรียกว่าดัชนีบิตแมปแบบพื้นฐาน ซึ่งมีการแทนค่าข้อมูล 1 บิตแมปเวกเตอร์ต่อ 1 ค่าของแอททริบิวต์ซึ่ง S^v มีค่าเท่ากับ 1 เมื่อค่าในเรคอร์ด $i = v$ ข้อดีคือ ใช้เวลาสอบถามน้อย ข้อเสียถ้าคาร์ดินอลิตี้ C สูงจำนวนบิตแปมเวกเตอร์ก็จะมาก ทำให้ใช้พื้นที่จัดเก็บดัชนีมากขึ้นด้วย ดังนั้นจึงได้มีการพัฒนาเทคนิคเพิ่มประสิทธิภาพของดัชนีบิตแมปแบบพื้นฐานแบ่งเป็น 2 กลุ่มคือ 1.กลุ่มที่ใช้วิธีการบีบอัด (Bitmap Index Compression) 2.กลุ่มที่ใช้วิธีการลงรหัส (Bitmap Index Extension) ดังภาพประกอบ 2-4



ภาพประกอบ 2-4 การเพิ่มประสิทธิภาพในเชิงพื้นที่ของดัชนีบิตแมป(Keawpibal *et al*., 2012)

กลุ่มที่ใช้วิธีการบีบอัด ประกอบด้วย WAH (Word – Aligned Hybrid) (Deliège and Pedersen, 2010) และ RLH (Run – Length Huffman) (Stabno and R. Wrembel, 2009) ข้อดีก็คือ ลดพื้นที่เก็บดัชนี แต่ข้อเสียก็คือ เมื่อต้องการสอบถามจะต้องแปลงดัชนีแบบบีบอัดให้กลับไปให้อยู่ในรูปแบบของดัชนีบิตแมปแบบพื้นฐานและใช้วิธีการสอบถามแบบเดียวกับดัชนีบิตแมปแบบพื้นฐาน

กลุ่มที่ใช้วิธีการลงรหัส ประกอบด้วย ดัชนีบิตแมปแบบช่วง (Chan and Ioannidis, 1999) เป็นดัชนีบิตแมปที่ใช้พื้นที่ได้มีประสิทธิภาพกว่าดัชนีบิตแมปแบบพื้นฐาน การสอบถามข้อมูลมีโอกาสตรวจสอบแค่ 1 บิตแมปเวกเตอร์ เถ้าคาร์ดินอลิตี้น้อยกว่าหรือเท่ากับ 3 ดัชนีบิตแมปแบบกระจาย (Vanichayobon *et al*., 2006) ใช้พื้นที่ในการจัดเก็บดัชนีน้อยกว่าดัชนีบิตแมปแบบช่วง การสอบถามข้อมูลจะดำเนินการกับ 2 บิตแมปเวกเตอร์ ดัชนีบิตแมปแบบคู่กัน (Wattanakitrungroj and S. Vanichayobon, 2006)ใช้บิตแมปเวกเตอร์น้อยกว่าดัชนีบิตแมปแบบกระจาย การสอบถามข้อมูลจะดำเนินการระหว่าง 2 บิตแมปเวกเตอร์ และดัชนีบิตแมปแบบเข้ารหัส (Wu and Buchmann, 1998; Keawpibal *et al*., 2012) เป็นดัชนีบิตแมปที่ใช้พื้นที่น้อยที่สุด แต่การสอบถามข้อมูลแบบดำเนินการกับทุกบิตแมปเวกเตอร์ ข้อดีของกลุ่มที่ใช้วิธีการลงรหัสคือ เมื่อต้องการสอบถามสามารถทำได้ทันที และมีประสิทธิภาพในเรื่องของการลดพื้นที่การจัดเก็บดัชนีพอสมควร

สำหรับงานวิจัยนี้สนใจการทำดัชนีบิตแมปแบบลงรหัสเนื่องจากข้อดีของการสอบถามที่สามารถทำได้ทันทีไม่จำเป็นต้องแปลงดัชนีบิตแมปให้กลับมาอยู่ในรูปแบบของดัชนีบิตแมปแบบพื้นฐานอีก โดยจะนำเสนอวิธีการทำดัชนีบิตแมปแบบพื้นฐานและดัชนีบิตแมปแบบลงรหัสแต่ละแบบโดยใช้ภาพประกอบ 2-5 เป็นตัวอย่างตารางที่เก็บข้อมูลอยู่ในคลังข้อมูล และจะยกตัวอย่างการทำดัชนีบิตแมปแบบพื้นฐาน แบบช่วง แบบกระจาย แบบคู่กัน และดัชนีบิตแมปแบบเข้ารหัสบนแอททริบิวต์ A ที่มีคาร์ดินอลิตี้ (ให้ C แทน คาร์ดินอลิตี้) เท่ากับ 20 ดังนี้

RID	A	B	C
1	10	a	1253
2	8	d	152
3	3	b	2536
4	15	y	123
5	16	d	24
6	19	j	14523
7	12	a	234
8	1	c	4
9	11	d	4
10	4	v	4545
11	0	x	4534
12	17	a	4534
13	5	d	45

ภาพประกอบ 2-5 ตัวอย่างตารางในคลังข้อมูล

2.3.1 ดัชนีบิตแมปแบบพื้นฐาน (Simple Bitmap Index)

ดัชนีบิตแมปแบบพื้นฐาน (O'Neil and Quass, 1997) จะมีการแทนค่า 1 แอททริบิวต์ต่อ 1 บิตแมปเวกเตอร์บนแอททริบิวต์ที่มีคาร์ดินอลิตี้เท่ากับ C (ค่าที่เป็นไปได้ทั้งหมดของแอททริบิวต์) การทำดัชนีบิตแมปแบบพื้นฐานจะประกอบไปด้วยบิตแมปเวกเตอร์เท่ากับ C บิตแมปเวกเตอร์ $S^0, S^1, S^2, \ldots, S^{C-1}$ โดยที่ S^v แทนบิตแมปเวกเตอร์ที่มีค่าแอททริบิวต์เท่ากับ v

<u>การลงรหัส</u> สามารถทำได้โดย S^v ลงรหัสเป็นบิต 1 เมื่อค่าของแอททริบิวต์ A ในเรคอร์ดเท่ากับ v

<u>สอบถาม</u> การสอบถามข้อมูลแบบค่าเท่ากันของดัชนีบิตแมปแบบพื้นฐานสามารถทำได้โดยให้ "A = v" เป็นรูปแบบการสอบถาม คือต้องการทราบว่าแอททริบิวต์ A มีเรคอร์ดใดบ้างที่มีค่าเท่ากับ v ซึ่งก็คือ A = v อ่านบิตแมปเวกเตอร์ S^v

ตัวอย่าง ถ้าต้องการสร้างดัชนีบนแอททริบิวต์ A ที่มีค่าคาร์ดินอลิตี้เท่ากับ 20 ($C = 20$) คือแอททริบิวต์ A มีค่าที่เป็นไปได้ทั้งหมด ตั้งแต่ 0 ถึง 19 ดังนั้นดัชนีบิตแมปแบบพื้นฐานจะใช้จำนวนบิตแมปเวกเตอร์ $S^0, S^1, S^2,..., S^{19}$ ยกตัวอย่างการลงรหัสแอททริบิวต์ที่มีค่าเท่ากับ 15 บิตแมปเวกเตอร์ที่แทนบิตเป็น “1” ในตำแหน่งเรอคอร์ดที่มีค่าแอททริบิวต์เท่ากับ 15 คือ S^{15} ดังแสดงในภาพประกอบ 2-6

RID	A
1	10
2	8
3	3
4	15
5	6
6	19
7	12
8	1
9	11
10	4
11	0
12	17
13	5

(a) ข้อมูล

s^{19}	s^{17}	s^{15}	s^{12}	s^{11}	s^{10}	…	s^5	s^4	s^3	s^1	s^0
0	0	0	0	0	1	...	0	0	0	0	0
0	0	0	0	0	0	...	0	0	0	0	0
0	0	0	0	0	0	...	0	0	1	0	0
0	0	1	0	0	0	...	0	0	0	0	0
0	0	0	0	0	0	...	0	0	0	0	0
1	0	0	0	0	0	...	0	0	0	0	0
0	0	0	1	0	0	...	0	0	0	0	0
0	0	0	0	0	0	...	0	0	0	1	0
0	0	0	0	1	0	...	0	0	0	0	0
0	0	0	0	0	0	...	0	1	0	0	0
0	0	0	0	0	0	...	0	0	0	0	1
0	1	0	0	0	0	...	0	0	0	0	0
0	0	0	0	0	0	...	1	0	0	0	0

(b) ดัชนีบิตแมปแบบพื้นฐาน

ภาพประกอบ 2-6 การลงรหัสดัชนีบิตแมปแบบพื้นฐาน

จากภาพประกอบ 2-6 ถ้าต้องการสอบถามข้อมูลแบบค่าเท่ากันของดัชนีบิตแมปแบบพื้นฐานสามารถทำได้โดยใช้รูปแบบการสอบถาม A = v คือ ถ้าต้องการทราบว่าแอททริบิวต์ A มีเรคอร์ดใดบ้างที่มีค่าเท่ากับ v ซึ่งก็คือ A = v อ่านบิตแมปเวกเอตร์ S^v เช่น ถ้าต้องการหา A = 15 คือ $v = 15$ ดังนั้นอ่านบิตแมปเวกเตอร์ S^{15} บิตที่ 3 มีค่าเท่ากับ 1 ส่วนบิตอื่น ๆ มีค่าเท่ากับ 0 แสดงว่าเรคอร์ดของแอททริบิวต์ที่มีค่าเท่ากับ 15 คือเรคอร์ดที่ 4 ดังที่แสดงในภาพประกอบ 2-7

RID	A
1	10
2	8
3	3
4	15
5	6
6	19
7	12
8	1
9	11
10	4
11	0
12	17
13	5

(a) ข้อมูล

s^{19}	s^{17}	s^{15}	s^{12}	s^{11}	s^{10}	…	s^{5}	s^{4}	s^{3}	s^{1}	s^{0}
0	0	0	0	0	1	...	0	0	0	0	0
0	0	0	0	0	0	...	0	0	0	0	0
0	0	0	0	0	0	...	0	0	1	0	0
0	0	1	0	0	0	...	0	0	0	0	0
0	0	0	0	0	0	...	0	0	0	0	0
1	0	0	0	0	0	...	0	0	0	0	0
0	0	0	1	0	0	...	0	0	0	0	0
0	0	0	0	0	0	...	0	0	0	1	0
0	0	0	0	1	0	...	0	0	0	0	0
0	0	0	0	0	0	...	0	1	0	0	0
0	0	0	0	0	0	...	0	0	0	0	1
0	1	0	0	0	0	...	0	0	0	0	0
0	0	0	0	0	0	...	1	0	0	0	0

(b) ดัชนีบิตแมปแบบพื้นฐาน

ภาพประกอบ 2-7 ผลการสอบถามแบบค่าเท่ากันของดัชนีบิตแมปแบบพื้นฐาน

การสอบถามแบบเป็นสมาชิกบนดัชนีบิตแมปแบบพื้นฐานทำได้โดย A in $\{v^1, v^2, \ldots v^n\}$ หมายถึง การสอบถามว่าบนแอททริบิวต์ A มีเรคอร์ดใดบ้างที่เท่ากับ $v^1, v^2, \ldots v^{n-1}$ วิธีการสอบถามแบบเป็นสมาชิกบนดัชนีบิตแมปแบบพื้นฐานทำได้โดย อ่านค่าบิตแมปเวกเตอร์แต่ละค่าในเงื่อนไขและนำมาดำเนินตรรกะ OR ตัวอย่างจากภาพประกอบ 2-7 ถ้าต้องการทราบว่า บนแอท-ทริบิวต์ A มีเรคอร์ดใดบ้างที่มีค่าเท่ากับ {3, 12, 1, 0} ดังนั้นอ่านบิตแมปเวกเตอร์ S^3, S^{12}, S^1, S^0 ดำเนินการ OR กันจะได้เรคอร์ดที่มีค่าแอททริบิวต์เท่ากับ {3, 12, 1, 0} คือเรคอร์ดที่ 3, 7, 8 และ11 ดังแสดงในภาพประกอบ 2-8 (c)

ข้อดีของดัชนีบิตแมปแบบพื้นฐาน

เหมาะกับแอททริบิวต์ที่มีค่าคาร์ดินอลิตี้ต่ำ เช่น แอททริบิวต์เพศ ซึ่งมีคาร์ดินอลิตี้เท่ากับสองใช้พื้นที่ในการจัดเก็บบิตแมปเวกเตอร์น้อย ไม่ซับซ้อนและเข้าถึงข้อมูลได้โดยตรง

ข้อเสียของดัชนีบิตแมปแบบพื้นฐาน

ถ้าหากแอททริบิวต์ที่นำมาใช้มีค่าคาร์ดินอลิตี้สูงมากๆ เช่น รายชื่อสินค้าในห้างสรรพสินค้าจะต้องใช้บิตแมปเวกเตอร์จำนวนมากเท่ากับรายการสินค้า ถ้าหากรายการสินค้ามี 1,000 รายการ จำนวนบิตแมปเวกเตอร์ที่ใช้ก็จะเท่ากับ 1,000 บิตแมปเวกเตอร์ เพราะแต่ละค่าของแอททริบิวต์จะถูกแทนด้วย 1 บิตแมปเวกเตอร์จึงทำให้สิ้นเปลืองพื้นที่มาก

RID	A
1	10
2	8
3	3
4	15
5	6
6	19
7	12
8	1
9	11
10	4
11	0
12	17
13	5

(a) ข้อมูล

s^{19}	s^{17}	s^{15}	s^{12}	s^{11}	s^{10}	…	s^5	s^4	s^3	s^1	s^0
0	0	0	0	0	1	...	0	0	0	0	0
0	0	0	0	0	0		0	0	0	0	0
0	0	0	0	0	0	...	0	0	1	0	0
0	0	1	0	0	0	...	0	0	0	0	0
0	0	0	0	0	0	...	0	0	0	0	0
1	0	0	0	0	0	...	0	0	0	0	0
0	0	0	1	0	0	...	0	0	0	0	0
0	0	0	0	0	0	...	0	0	0	1	0
0	0	0	0	1	0	...	0	0	0	0	0
0	0	0	0	0	0	...	0	1	0	0	0
0	0	0	0	0	0	...	0	0	0	0	1
0	1	0	0	0	0	...	0	0	0	0	0
0	0	0	0	0	0	...	1	0	0	0	0

(b) ดัชนีบิตแมปแบบพื้นฐาน

$s^3 \vee s^{12} \vee s^1 \vee s^0$
0
0
1
0
0
0
1
1
0
0
1
0
0

(c) ผลลัพธ์

ภาพประกอบ 2-8 ผลการสอบถามแบบเป็นสมาชิกของดัชนีบิตแมปแบบพื้นฐาน

2.3.2 ดัชนีบิตแมปแบบช่วง (Interval Bitmap Index)

การทำดัชนีบิตแมปแบบช่วง (Chan and Ioannidis, 1999) บนแอททริบิวต์ที่มีคาร์ดินอลิตี้เท่ากับ C จะใช้พื้นที่ในการจัดเก็บดัชนีได้คุ้มค่ากว่าดัชนีบิตแมปแบบพื้นฐาน โดยใช้บิตแมปเวกเตอร์เท่ากับ $\lceil C/2 \rceil$ บิตแมปเวกเตอร์ คือ ตั้งแต่ I^0, I^1, I^2,.. ,$I^{\lceil C/2 \rceil - 1}$ โดย I^j จะแทนบิตแมปเวกเตอร์ที่มีค่าอยู่ในช่วง $I^j = [j, j+m]$ โดยที่ $m = \lceil C/2 \rceil - 1$

<u>การลงรหัส</u> บิตแมปเวกเตอร์ I^j ลงรหัสเป็นบิต 1 เมื่อค่าของแอททริบิวต์ A ใน เรคอร์ดมีค่าอยู่ในช่วง $I^j = [j, j+m]$ โดยค่า m หาได้จากสูตร $m = \lceil C/2 \rceil - 1$ บิตอื่น ๆ ให้มีค่าเท่ากับ 0

<u>การสอบถาม</u> ถ้าต้องการทราบว่าแอททริบิวต์ A มีเรคอร์ดใดบ้างที่มีค่าเท่ากับ v มีอัลกอริทึมดังนี้ ค่า m คำนวณจากสูตร $m = \lceil C/2 \rceil - 1$

$$\text{“A} = v\text{”} = \begin{cases} I^0 \text{ if } v = 0, m = 0 \\ \overline{I^0} \text{ if } v = 1, C = 2 \\ I^1 \text{ if } v = 1, C = 3 \\ I^v \wedge I^{\overline{v+1}} \text{ if } v < m \\ I^v \wedge I^0 \text{ if } v = m, m > 0 \\ I^{v-m} \wedge I^{\overline{v-m-1}} \text{ if } m < v < C-1, m > 0 \\ \overline{I^{\lceil C/2 \rceil - 1} \vee I^0} \text{ if } v = C-1 \end{cases}$$

<u>ตัวอย่าง</u> ถ้าแอททริบิวต์ A มีค่าคาร์ดินอลิตี้เท่ากับ 20 คือตั้งแต่ 0 ถึง 19 จำนวนบิตแมปเวกเตอร์ของดัชนีบิตแมปแบบช่วงคำนวณจาก

$$\lceil C/2 \rceil$$

$$\lceil 20/2 \rceil = 10$$

ดังนั้น ดัชนีบิตแมปแบบช่วงจะใช้บิตแมปเวกเตอร์สิบตัวคือ $I^9\ I^8\ I^7\ I^6\ I^5\ I^4\ I^3\ I^2\ I^1\ I^0$

$$m = \lceil 20/2 \rceil - 1$$

$$m = \lceil 10 \rceil - 1$$

$$m = 9$$

ดังนั้น บิตแมปเวกเตอร์ I^j จะแทนค่าของแอททริบิวต์ A ที่อยู่ในช่วง $I^j = j, j+9$

I^0 = แทนค่าของแอททริบิวต์ A ที่อยู่ในช่วง [0, 9]

I^1 = แทนค่าของแอททริบิวต์ A ที่อยู่ในช่วง [1, 10]

I^2 = แทนค่าของแอททริบิวต์ A ที่อยู่ในช่วง [2, 11]

I^3 = แทนค่าของแอททริบิวต์ A ที่อยู่ในช่วง [3, 12]

I^4 = แทนค่าของแอททริบิวต์ A ที่อยู่ในช่วง [4, 13]

I^5 = แทนค่าของแอททริบิวต์ A ที่อยู่ในช่วง [5, 14]

I^6 = แทนค่าของแอททริบิวต์ A ที่อยู่ในช่วง [6, 15]

I^7 = แทนค่าของแอททริบิวต์ A ที่อยู่ในช่วง [7, 16]

I^8 = แทนค่าของแอททริบิวต์ A ที่อยู่ในช่วง [8, 17]

I^9 = แทนค่าของแอททริบิวต์ A ที่อยู่ในช่วง [9, 18]

จากภาพประกอบที่ 2-9 ถ้าต้องการสอบถามข้อมูลแบบค่าเท่ากันของดัชนีบิตแมปแบบช่วง โดยการหาแอททริบิวต์ A มีเรคอร์ดใดบ้างเท่ากับ 15 สามารถทำได้โดย A = v = 15 และ m = 9 (ที่ได้คำนวณไว้แล้วในข้างต้น)

จาก v = 15, m = 9 เลือกใช้เงื่อนไข

$$I^{v-m} \wedge I^{\overline{v-m-1}} \text{ if } m < I < C-1, m > 0$$

$$I^{15-9} \wedge I^{\overline{15-9-1}} \text{ if } 9 < 10 < 20-1, 9 > 0$$

$$I^6 \wedge I^{\overline{5}} \text{ if } 9 < 10 < 20-1, 9 > 0$$

อ่านบิตแมป $I^6 \wedge I^{\overline{5}}$ จะได้ผลลัพธ์คือบิตในเรคอร์ดที่ 4 มีค่าเท่ากับ 1 คำตอบจึงอยู่ในเรคอร์ดที่ 4 ดังแสดงในภาพประกอบ 2-10(b)

RID	A
1	10
2	8
3	3
4	15
5	6
6	19
7	12
8	1
9	11
10	4
11	0
12	17
13	5

(a) ข้อมูล

I^9	I^8	I^7	I^6	I^5	I^4	I^3	I^2	I^1	I^0
1	1	1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	1	1	1	1
1	1	1	1	0	0	0	0	0	0
0	0	0	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	0	0	0
0	0	0	0	0	0	0	0	1	1
1	1	1	1	1	1	1	1	0	0
0	0	0	0	0	1	1	1	1	1
0	0	0	0	0	0	0	0	0	1
1	1	0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1	1	1

(b) ดัชนีบิตแมปแบบช่วง

ภาพประกอบ 2-9 การลงรหัสดัชนีบิตแมปแบบช่วง

RID	A
1	10
2	8
3	3
4	15
5	6
6	19
7	12
8	1
9	11
10	4
11	0
12	17
13	5

(a) ข้อมูล

I^6	$I^{\overline{5}}$	=	$I^6 \wedge I^{\overline{5}}$
1	0		0
1	0		0
0	1		0
1	1		1
1	0		0
0	1		0
1	0		0
0	1		0
1	0		0
0	1		0
0	1		0
0	1		0
0	0		0

(b) ผลการสอบถาม

ภาพประกอบ 2-10 ผลการสอบถามแบบค่าเท่ากันของดัชนีบิตแมปแบบช่วง

การสอบถามแบบเป็นสมาชิกบนดัชนีบิตแมปแบบช่วง สามารถทำได้โดยนำรูปแบบการลงรหัสแต่ละค่ามาดำเนินการ OR (แทน OR ด้วยเครื่องหมาย +) เช่นจากภาพประกอบที่ 2-9 ถ้าต้องการทราบว่า บนแอททริบิวต์ A มีเรคอร์ดใดบ้างที่มีค่าเท่ากับ {3, 12, 1, 0} ผลลัพธ์

ของการสอบถามคือ $(I^3 \wedge \overline{I^4}) + (I^3 \wedge \overline{I^2}) + (I^1 \wedge \overline{I^2}) + (I^0 \wedge \overline{I^1})$ เรคอร์ดที่เป็นคำตอบของการสอบถามคือ 3, 7, 8, 11 ดังแสดงในภาพประกอบ 2-11(b)

ข้อดีของดัชนีบิตแมปแบบช่วง ใช้พื้นที่ในการจัดเก็บได้คุ้มค่ากว่าดัชนีบิตแมปแบบพื้นฐาน และใช้บิตแมปเวกเตอร์น้อยกว่าดัชนีบิตแมปแบบพื้นฐานถึงครึ่งหนึ่ง

ข้อเสียของดัชนีบิตแมปแบบช่วง การสอบถามข้อมูลแบบค่าเท่ากันมีประสิทธิภาพด้อยกว่าดัชนีบิตแมปแบบพื้นฐานจากขั้นตอนวิธีที่ใช้สอบถามเมื่อแอททริบิวต์มีค่าคาร์ดินอลิตี้มากกว่า 3 จะต้องดำเนินการระหว่าง 2 บิตแมปเวกเตอร์

RID	A
1	10
2	8
3	3
4	15
5	6
6	19
7	12
8	1
9	11
10	4
11	0
12	17
13	5

(a) ข้อมูล

$I^3 \wedge \overline{I^4}$		+	$I^3 \wedge \overline{I^2}$		+	$I^1 \wedge \overline{I^2}$		+	$I^0 \wedge \overline{I^1}$		=	result
1	0		1	0		1	0		0	0		0
1	0		1	0		1	0		1	0		0
1	1		1	0		1	0		1	0		1
0	1		0	1		0	1		0	1		0
1	0		1	0		1	0		1	0		0
0	1		0	1		0	1		0	1		0
1	0		1	1		0	1		0	1		1
0	1		0	1		1	1		1	0		1
1	0		1	0		0	0		0	1		0
1	0		1	0		1	0		1	0		0
0	1		0	1		0	1		1	1		1
0	1		0	1		0	1		0	1		0
1	0		1	0		1	0		1	0		0

(b) ผลลัพธ์การสแบถามแบบเป็นสมาชิกบนดัชนีบิตแมปแบบช่วง

ภาพประกอบ 2-11 ผลการสอบถามแบบเป็นสมาชิกบนดัชนีบิตแมปแบบช่วง

2.3.3 ดัชนีบิตแมปแบบกระจาย (Scatter Bitmap Index)

การทำดัชนีบิตแมปแบบกระจาย (Vanichayobon *et al*., 2006) เหมาะกับข้อมูลที่มีการค้นหาแบบค่าเดียว การทำดัชนีบิตแมปแบบกระจายบนแอททริบิวต์ที่มีคาร์ดินอลิตี้เท่ากับ C มีขั้นตอนดังนี้ การหาจำนวนบิตแมปเวกเตอร์สามารถหาได้จากสูตร $\left\lceil 2\sqrt{C} \right\rceil$ โดยที่บิตแมปเวกเตอร์ของดัชนีบิตแมปแบบกระจายแบ่งเป็น 2 กลุ่ม คือ กลุ่ม Z และ กลุ่ม L หรืออาจจะถูกแทนอยู่ในกลุ่ม Z เพียงกลุ่มเดียว

<u>การลงรหัส</u> เราสามารถหา Z และ L ได้โดย

สูตรการหาบิตแมปเวกเตอร์กลุ่ม Z คือ $Z = \left\lceil \frac{C}{m-1} \right\rceil + 1$

สูตรการหาบิตแมปเวกเตอร์กลุ่ม L คือ $L = m - 2$ โดยที่ค่า m หาได้จากสูตร $m = \left\lceil \sqrt{C} \right\rceil + 1$

การสร้างบิตแมปเวกเตอร์กลุ่ม Z

- บิตแมปเวกเตอร์ Z^0 ลงรหัสเป็นบิต 1 เมื่อ ค่าของแอททริบิวต์ A ในเรคอร์ดมีค่าเท่ากับ 0 ส่วนบิตที่เหลือให้ลงรหัสเป็นบิต 0
- บิตแมปเวกเตอร์ Z^j ลงรหัสเป็นบิต 1 เมื่อ ค่าของแอททริบิวต์ A ในเรคอร์ดมีค่าเท่ากับ v ส่วนบิตที่เหลือให้มีค่าเท่ากับ 0 โดย j คำนวณจาก $j = \left\lfloor \frac{v}{m-1} \right\rfloor + 1$
- ถ้า m-1 หาร v ลงตัว จะให้บิตที่ i ของบิตแมปเวกเตอร์ Z^{j-1} มีค่าเป็น 1 ด้วย

การสร้างบิตแมปเวกเตอร์กลุ่ม L

- บิตแมปเวกเตอร์ L^k ลงรหัสเป็นบิต 1 เมื่อ $k = v \bmod (m-1)$ โดยที่ i มีค่าแอททริบิวต์เท่ากับ v และ $k \neq 0$

<u>การสอบถาม</u> ถ้าต้องการทราบว่าแอททริบิวต์ A มีเรคอร์ดใดบ้างที่มีค่าเท่ากับ v สามารถทำได้โดยใช้ขั้นตอนวิธีดังนี้

$$A = v = \begin{cases} Z^{\left\lfloor \frac{v}{m-1} \right\rfloor} \wedge Z^{\left\lfloor \frac{v}{(m-1)} \right\rfloor + 1} & \text{if } v \bmod (\text{m - 1}) = 0 \\ Z^{\left\lfloor \frac{v}{(m-1)} \right\rfloor + 1} \wedge L^{v \bmod (m-1)} & \text{if Otherwise} \end{cases}$$

<u>ตัวอย่าง</u> ถ้าแอททริบิวต์ A มีค่าคาร์ดินอลิตี้เท่ากับ 20 จำนวนบิตแมปเวกเตอร์ที่ใช้จะเท่ากับ $\left\lceil 2\sqrt{20} \right\rceil = 9$ บิตแมปเวกเตอร์ และค่า m เท่ากับ $m = \left\lceil \sqrt{C} \right\rceil + 1 = 6$

จำนวนบิตแมปเวกเตอร์กลุ่ม Z หาได้โดย

$$Z = \left\lceil \frac{C}{m-1} \right\rceil + 1$$

$$Z = \left\lceil \frac{20}{6-1} \right\rceil + 1$$

$Z = 5$ ดังนั้นกลุ่ม Z มีบิตแมปเวกเตอร์ $Z^4Z^3Z^2Z^1Z^0$

จำนวนบิตแมปเวกเตอร์กลุ่ม L หาได้โดย

$L = m - 2$

$L = 6 - 2$

$L = 4$ ดังนั้นกลุ่ม L มีบิตแมปเวกเตอร์ $L^4L^3L^2L^1$

ถ้าต้องการลงรหัสแอททริบิวต์ที่มีค่าเท่ากับ 15 ให้ $v = 15$, $m = 6$

ลงรหัสบิตแมปเวกเตอร์กลุ่ม Z

$$j = \left\lfloor \frac{v}{m-1} \right\rfloor + 1$$

$$j = \left\lfloor \frac{15}{6-1} \right\rfloor + 1$$

$j = 4$ ดังนั้น ที่แอททริบิวต์ A ที่มีค่าเท่ากับ 15 ถูกลงรหัสในบิตแมปเวกเตอร์ Z^4 เท่ากับ 1

ลงรหัสบิตแมปเวกเตอร์กลุ่ม L

$k = v \bmod (m-1)$

$k = 15 \bmod (6-1)$

$k = 0$ ดังนั้น ที่แอททริบิวต์ A ที่มีค่าเท่ากับ 15 จึงไม่ลงรหัสในบิตแมปเวกเตอร์กลุ่ม L

ตัวอย่างเพื่อให้เห็นภาพชัดเจนขึ้นจะยกตัวอย่างอีกหนึ่งตัวอย่างที่มีความแตกต่างกัน

ถ้าต้องการลงรหัสแอททริบิวต์ที่มีค่าเท่ากับ 1 ให้ $v = 1$, $m = 6$

ลงรหัสบิตแมปเวกเตอร์กลุ่ม Z

$$j = \left\lfloor \frac{v}{m-1} \right\rfloor + 1$$

$$j = \left\lfloor \frac{1}{6-1} \right\rfloor + 1$$

$j = 1$ ดังนั้น ที่แอททริบิวต์ A ที่มีค่าเท่ากับ 1 ถูกลงรหัสในบิตแมปเวกเตอร์ Z^1 เท่ากับ 1

ลงรหัสบิตแมปเวกเตอร์กลุ่ม L

$k = v \bmod (m-1)$

$k = 1 \bmod (6-1)$

$k=1$ ดังนั้น ที่แอททริบิวต์ A ที่มีค่าเท่ากับ 1 ถูกลงรหัสในบิตแมปเวกเตอร์ L^1 เท่ากับ 1

ดังแสดงในภาพประกอบ 2-12

RID	A
1	10
2	8
3	3
4	15
5	16
6	19
7	12
8	1
9	11
10	4
11	0
12	17
13	5

(a) ข้อมูล

L^4	L^3	L^2	L^1	Z^4	Z^3	Z^2	Z^1	Z^0
0	0	0	0	0	1	1	0	0
0	1	0	0	0	0	1	0	0
0	1	0	0	0	0	0	1	0
0	0	0	0	1	1	0	0	0
0	0	0	1	1	0	0	0	0
1	0	0	0	1	0	0	0	0
0	0	1	0	0	1	0	0	0
0	0	0	1	0	0	0	1	0
0	0	0	1	0	1	0	0	0
1	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	1	1
0	0	1	0	1	0	0	0	0
0	0	0	0	0	0	1	1	0

(b) ดัชนีบิตแมปแบบกระจาย

ภาพประกอบ 2-12 การลงรหัสดัชนีบิตแมปแบบกระจาย

จากภาพประกอบ 2-12 การสอบถามแบบค่าเท่ากันของดัชนีบิตแมปแบบกระจาย สามารถทำได้โดย ถ้าต้องการสอบถามว่าแอททริบิวต์ A ที่มีค่าในเรคอร์ดเท่ากับ 15 แทน A=v = 15 แสดงว่า v = 15 และ m = 6 ในสูตร

$$v \bmod (m-1)$$

$$15 \bmod (6-1) = 0$$

เนื่องจากค่าที่คำนวณได้มีค่าเท่ากับ 0 จึงเลือกใช้การสอบถามแบบ

$$Z^{\left\lfloor \frac{v}{m-1} \right\rfloor} \wedge Z^{\left\lfloor \frac{v}{(m-1)} \right\rfloor + 1}$$

$$Z^{\left\lfloor \frac{15}{6-1} \right\rfloor} \wedge Z^{\left\lfloor \frac{15}{(6-1)} + 1 \right\rfloor}$$

$$Z^3 \wedge Z^4$$

ดังนั้นจึงดำเนินการ AND ระหว่าบิตแมปเวกเตอร์ $Z^3 \wedge Z^4$ ดังแสดงในภาพประกอบ 2-13 เรคอร์ดที่มีค่าเท่ากับ 1 ตรงกับเรคอร์ดที่ 4 ซึ่งเป็นคำตอบของการสอบถาม

RID	A
1	10
2	8
3	3
4	15
5	16
6	19
7	12
8	1
9	11
10	4
11	0
12	17
	5

(a) ข้อมูล

Z^4	Z^3		$Z^4 \wedge Z^3$
0	1		0
0	0		0
0	0		0
1	1		1
1	0		0
1	0	=	0
0	1		0
0	0		0
0	1		0
0	0		0
0	0		0
1	0		0
0	0		0

(b) ผลการสอบถาม

ภาพประกอบ 2-13 ผลการสอบถามแบบค่าเท่ากันของดัชนีบิตแมปแบบกระจาย

การสอบถามแบบเป็นสมาชิกบนดัชนีบิตแมปแบบกระจาย สามารถทำได้โดยนำรูปแบบการลงรหัสแต่ละค่ามาดำเนินการ OR เช่น จากภาพประกอบที่ 2-12 ถ้าต้องการทราบว่า บนแอททริบิวต์ A มีเรคอร์ดใดบ้างที่มีค่าเท่ากับ {3, 12, 1, 0} ผลลัพธ์ของการสอบถามคือ $(Z^1 \wedge L^3)+(Z^3 \wedge L^2)+(Z^1 \wedge L^1)+(Z^0 \wedge Z^1)$ เรคอร์ดที่เป็นคำตอบของการสอบถามคือ 3, 7, 8, 11 ดังแสดงในภาพประกอบ 2-14(b)

RID	A
1	10
2	8
3	3
4	15
5	6
6	19
7	12
8	1
9	11
10	4
11	0
12	17
13	5

(a) ข้อมูล

$Z^1 \wedge L^3$			$Z^3 \wedge L^2$			$Z^1 \wedge L^1$			$Z^0 \wedge Z^1$			result
0	0		1	0		0	0		0	0		0
0	1		0	0		0	0		0	0		0
1	1		0	0		1	0		0	1		1
0	0		1	0		0	0		0	0		0
0	0		0	0		0	1		0	0		0
0	0		0	0		0	0		0	0		0
0	0	+	1	1	+	0	0	+	0	0	=	1
1	0		0	0		1	1		0	1		1
0	0		1	0		0	1		0	0		0
1	0		0	0		1	0		0	1		0
1	0		0	0		1	0		1	1		1
0	0		0	1		0	0		0	0		0
1	0		0	0		1	0		0	1		0

(b) ผลลัพธ์การสแบถามแบบเป็นสมาชิกบนดัชนีบิตแมปแบบกระจาย

ภาพประกอบ 2-14 ผลการสอบถามแบบเป็นสมาชิกบนดัชนีบิตแมปแบบกระจาย

ข้อดีของดัชนีบิตแมปแบบกระจาย การดำเนินการในการสอบถามแบบค่าเท่ากันจะดำเนินการแค่สองบิตแมปเวกเตอร์ และใช้พื้นที่ในการจัดเก็บดัชนีน้อยกว่าดัชนีบิตแมปแบบพื้นฐานกับดัชนีบิตแมปแบบช่วง

ข้อเสียของดัชนีบิตแมปแบบกระจาย ใช้พื้นที่ในการจัดเก็บได้ไม่คุ้มค่าเท่าที่ควรจากตัวอย่าง เช่น บิตแมปเวกเตอร์ Z^0 ใช้แทนค่าแอททริบิวต์ได้แค่ค่าเดียว

2.3.4 ดัชนีบิตแมปแบบเข้ารหัส (Encoded Bitmap Index)

ดัชนีบิตแมปแบบเข้ารหัส (Wu and Buchmann, 1998) เป็นดัชนีบิตแมปที่ใช้เนื้อที่น้อยที่สุดโดยถ้าแอททริบิวต์มีค่าคาร์ดินอลิตี้เท่ากับ C ดัชนีบิตแมปแบบเข้ารหัสจะใช้บิตแมปเวกเตอร์ในการจัดเก็บดัชนีแค่ $\left\lceil \log_2 C \right\rceil$ คือมีค่าตั้งแต่ $\mathrm{E}^0\mathrm{E}^1\mathrm{E}^2 \ldots \mathrm{E}^{\left\lceil \log_2 C \right\rceil}$

<u>การลงรหัส</u> จะต้องมีการเปรียบเทียบค่าของแอททริบิวต์กับตารางเทียบค่าที่มีค่าเป็น 0 หรือ 1

<u>การสอบถาม</u> ทำได้โดยการนำค่าข้อมูลที่ต้องการสอบถามไปเทียบค่ากับตารางเทียบค่าและนำไปเทียบในตารางดัชนี

<u>ตัวอย่าง</u> ถ้าแอททริบิวต์ A มีค่าคาร์ดินอลิตี้เท่ากับ 20 จำนวนบิตแมปเวกเตอร์ที่ใช้จะเท่ากับ

$$\left\lceil \log_2 C \right\rceil$$

$$\log_2 20 = 5$$

ดังนั้นบิตแมปเวกเตอร์ที่ใช้มี $\mathrm{E}^4\mathrm{E}^3\mathrm{E}^2\mathrm{E}^1\mathrm{E}^0$ สามารถลงรหัสได้ดังแสดงในภาพประกอบ 2-15

จากภาพประกอบ 2-15 การสอบถามแบบค่าเท่ากันสามารถทำได้โดยการเทียบค่าที่ต้องการสอบถามกับค่าในตารางดัชนี เช่น สอบถามว่าแอททริบิวต์ A ที่มีค่าในเรคอร์ดเท่ากับ 15 จากตารางเทียบค่า $\overline{\mathrm{E}^4}\mathrm{E}^3\mathrm{E}^2\mathrm{E}^1\mathrm{E}^0$ คำตอบคือเรคอร์ดที่ 4 ในภาพประกอบ 2-15(b)

RID	A
1	10
2	8
3	3
4	15
5	16
6	19
7	12
8	1
9	11
10	4
11	0
12	17
13	5

(a) ข้อมูล

E^4	E^3	E^2	E^1	E^0
0	1	0	1	0
0	1	0	0	0
0	0	0	1	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
0	1	1	0	0
0	0	0	0	1
0	1	0	1	1
0	0	1	0	0
0	0	0	0	0
1	0	0	0	1
0	0	1	0	1

(b) ดัชนีบิตแมปแบบเข้ารหัส

0	00000
1	00001
2	00010
3	00011
4	00100
5	00101
6	00110
7	00111
8	01000
9	01001
10	01010
11	01011
12	01100
13	01101
14	01110
15	01111
16	10000
17	10001
18	10010
19	10011
20	10100

(c) ตารางเทียบค่า

ภาพประกอบ 2-15 ดัชนีบิตแมปแบบเข้ารหัส

การสอบถามแบบเป็นสมาชิกบนดัชนีบิตแมปแบบเข้ารหัสที่ได้โดยนำรูปแบบการลงรหัสของแต่ละค่าที่ต้องการสอบถามมาดำเนินการ OR กัน ตัวอย่างเช่น จากภาพประกอบที่ 2-15 ถ้าต้องการทราบว่า บนแอททริบิวต์ A มีเรคอร์ดใดบ้างที่มีค่าเท่ากับ {3, 12, 1, 0} โดยใช้ตารางเทียบค่า จะได้

$$\overline{E^4}\,\overline{E^3}\,\overline{E^2}E^1E^0 + \overline{E^4}E^3E^2\overline{E^1}\,\overline{E^0} + E^4E^3\overline{E^2}E^1E^0 + \overline{E^4}E^3\overline{E^2}\,\overline{E^1}\,\overline{E^0}$$

และต้องตรวจสอบทุกบิตแมปเวกเตอร์ของแต่ละค่า เรคอร์ดที่เป็นคำตอบคือ 3, 7, 8, 11

จากดัชนีบิตแมปแบบเข้ารหัสแบบดั้งเดิมเวลาสอบถามจะต้องใช้ตารางเทียบค่า เมื่อทำการเพิ่มประสิทธิภาพดัชนีบิตแมปแบบใหม่จึงไม่จำเป็นต้องใช้ตารางเทียบค่า (Keawpibal *et al*., 2012) โดยจะเรียกวิธีการเพิ่มประสิทธิภาพดัชนีบิตแมปแบบเข้ารหัสนี้ว่า E-EBI มีวิธีการทำดังนี้

<u>การลงรหัส</u> ถ้าแอททริบิวต์มีค่าคาร์ดินอลิตี้เท่ากับ C E-EBI จะใช้บิตแมปเวกเตอร์ในการจัดเก็บดัชนีแค่ $\lceil \log_2 C \rceil$ คือมีค่าตั้งแต่ $E^0E^1E^2 \ldots E^{\lceil \log_2 C \rceil}$ เหมือนกับดัชนีบิตแมปแบบเข้ารหัส

<u>การสอบถาม</u> การสอบถามแบบค่าเท่ากันทำได้โดยไม่ต้องสร้างตารางเทียบค่า ใช้เงื่อนไขดังนี้ A = v หมายถึง การสอบถามว่าค่าบนแอททริบิวต์ A มีเรคอร์ดใดบ้างที่มีค่าเท่ากับ v โดยเพิ่มการดำเนินการ AND และ NOT มาเพิ่มประสิทธิภาพในการสอบถาม

- สร้างบิตแมปเวกเตอร์ ANS_VECTOR ให้ทุกบิตเท่ากับ “1”

  for $i = 0$ to $\lceil \log_2 C \rceil - 1$

    If $E^i = 1$ then

      ANS_VECTOR = ANS_VECTOR $\wedge E^i$

    Else

      ANS_VECTOR = ANS_VECTOR $\wedge \overline{E^i}$

- ANS_VECTOR ในเรคอร์ดที่เท่ากับ 1 คือคำตอบ

<u>ตัวอย่าง</u> สอบถามว่าแอททริบิวต์ A ที่มีค่าในเรคอร์ดเท่ากับ 15 แทน A= v =15 แสดงว่า15 มีรูปแบบการลงรหัสเป็น 0 1 1 1 1 ($E^4 = 0$, $E^3 = 1$, $E^2 = 1$ $E^1 = 1$, $E^0 = 1$) ดังนั้นสามารถหาได้ E^3, E^2, E^1, E^0 ดำเนินการตรรกะ AND กับบิตแมปเวกเตอร์ ANS_VECTOR และ $\overline{E^4}$ ดำเนินการตรรกะ AND กับ ANS_VECTOR เรคอร์ดที่เท่ากับ 1 คือคำตอบ ดังแสดงในภาพประกอบ 2-16

RID	A
1	10
2	8
3	3
4	15
5	16
6	19
7	12
8	1
9	11
10	4
11	0
12	17
13	5

(a) ข้อมูล

E^4	E^3	E^2	E^1	E^0
0	1	0	1	0
0	1	0	0	0
0	0	0	1	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
0	1	1	0	0
0	0	0	0	1
0	1	0	1	1
0	0	1	0	0
0	0	0	0	0
1	0	0	0	1
0	0	1	0	1

(b) ดัชนีบิตแมปแบบเข้ารหัส E-EBI

$\overline{E^4}$	E^3	E^2	E^1	E^0	ANS_VECTOR
1	1	0	1	0	0
1	1	0	0	0	0
1	0	0	1	1	0
1	1	1	1	1	1
0	0	0	0	0	0
0	0	0	1	1	0
1	1	1	0	0	0
1	0	0	0	1	0
1	1	0	1	1	0
1	0	1	0	0	0
1	0	0	0	0	0
0	0	0	0	1	0
1	0	1	0	1	0

(c) การสอบถามดัชนีบิตแมปแบบเข้ารหัส E-EBI

ภาพประกอบ 2-16 การสอบถามข้อมูลแบบค่าเท่ากันของ E-EBI

ข้อดีของ E-EBI ใช้บิตแมปเวกเตอร์น้อยที่สุดจากดัชนีบิตแมปที่กล่าวมาทั้งหมด จึงเป็นดัชนีบิตแมปที่ใช้พื้นที่ได้คุ้มค่ามากที่สุด ดังนั้นจึงเหมาะสมกับการสร้างดัชนีบนแอททริบิวต์ที่มีคาร์ดินอลิตี้สูงมากๆ

ข้อเสียของ E-EBI ถึงแม้ว่าการทำดัชนีบิตแมปแบบเข้ารหัสจะสามารถลดพื้นที่ในการจัดเก็บดัชนีบิตแมปได้มากแต่การสอบถามแบบค่าเท่ากันจำเป็นต้องเปรียบเทียบกับทุกเรคอร์ดในตารางทำให้เสียเวลา

ต่อมาได้มีงานวิจัยเกี่ยวกับการเพิ่มประสิทธิภาพให้กับดัชนีบิตแมปแบบเข้ารหัสโดยไม่จำเป็นต้องใช้ตารางเทียบค่าแต่ก็ยังดำเนินการกับทุกบิตแมปเวกเตอร์

2.3.6 ดัชนีบิตแมปแบบคู่กัน (Dual Bitmap Index)

จากดัชนีบิตแมปแบบคู่กัน (Wattanakitrungroj and S. Vanichayobon, 2006) สามารถหาจำนวนบิตแมปเวกเตอร์ที่ใช้ในการสร้างดัชนีโดยใช้สูตรต่อไปนี้ ให้ แทน C ในสูตร

$$n = \left\lceil \sqrt{2C + 0.25} + 0.5 \right\rceil$$

<u>การลงรหัส</u> ให้ v แทน ค่าแอททริบิวต์ที่เลือกมาทำดัชนีบิตแมปแบบคู่กันโดยที่ $v = \{0, 1, 2, \ldots, C\text{-}1\}$

ให้ D^j แทน บิตแมปเวกเตอร์ที่ j โดยที่ j = {0, 1, 2, …, *n*-1}

$D^j = 1$ เมื่อ $j = r$ หรือ $j = s$

$D^j = 0$ กรณีอื่นๆ

กำหนดให้

r แทน บิตแมปเวกเตอร์ตัวแรกที่ต้องอ่าน

s แทน บิตแมปเวกเตอร์ตัวที่สองที่ต้องอ่าน

r และ *s* สามารถคำนวณได้จากสมการต่อไปนี้

$$r = \left\lceil \sqrt{2v + 2.25} - 0.5 \right\rceil \qquad \text{โดยที่ } 1 \le r \le n-1$$

และ $$s = v - \frac{r(r-1)}{2} \qquad \text{โดยที่ } 0 \le s < r$$

<u>การสอบถาม</u> ถ้าต้องการสอบถามแบบค่าเท่ากันว่าบนแอททริบิวต์ A มีเรคอร์ดใดบ้างที่มีค่าเท่ากับ *v*

"A=*v*" แทนสูตรต่อไปนี้เพื่อหาค่า *r* และ *s*

$$r = \left\lceil \sqrt{2v + 2.25} - 0.5 \right\rceil$$

$$s = v - \frac{r(r-1)}{2}$$

$D^r \wedge D^s = 1$

<u>ตัวอย่าง</u> ให้มีแอททริบิวต์ A มีคาร์ดินอลิตี้ที่เป็นไปได้เท่ากับ 20 คือมีค่าที่เป็นไปได้ตั้งแต่ 0- 19

การหาจำนวนบิตแมปเวกเตอร์ที่ใช้ในการสร้างดัชนี (ค่าของ *n*)

$$n = \left\lceil \sqrt{2(20) + 0.25} + 0.5 \right\rceil$$

$$n = 7$$

ถ้าคาร์ดินอลิตี้เท่ากับ 20ใช้ 7 บิตแมปเวกเตอร์ คือ $D^6\ D^5\ D^4\ D^3\ D^2\ D^1\ D^0$

เช่น ถ้าต้องการลงรหัสของแอททริบิวต์ที่มีค่าเท่ากับ 15 (*v* =15)

ตำแหน่งบิตแมปเวกเตอร์ *r* คือ $$r = \left\lceil \sqrt{2v + 2.25} - 0.5 \right\rceil$$

$$r = \left\lceil \sqrt{2(15) + 2.25} - 0.5 \right\rceil$$

$$r = 6$$

ตำแหน่งบิตแมปเวกเตอร์ *s* คือ $$s = v - \frac{r(r-1)}{2}$$

$$s = 15 - \frac{6(6-1)}{2}$$

$$s = 0$$

แสดงว่าแอททริบิวต์ที่มีค่าเท่ากับ 15 ลงรหัสเป็น 1 ในตำแหน่งบิตแมปเวกเตอร์ D^6 และ D^0 ได้ดังแสดงในภาพประกอบ 2-17

RID	A
1	10
2	8
3	3
4	15
5	16
6	19
7	12
8	1
9	11
10	4
11	0
12	17
13	5

(a) ข้อมูล

D^6	D^5	D^4	D^3	D^2	D^1	D^0
0	1	0	0	0	0	1
0	0	1	0	1	0	0
0	0	0	1	0	0	1
1	0	0	0	0	0	1
1	0	0	0	0	1	0
1	0	1	0	0	0	0
0	1	0	0	1	0	0
0	0	0	0	1	0	1
0	1	0	0	0	1	0
0	0	0	1	0	1	0
0	0	0	0	0	1	1
1	0	0	0	1	0	0
0	0	0	1	1	0	0

(b) ดัชนีบิตแมปแบบคู่กัน

ภาพประกอบ 2-17 การลงรหัสของดัชนีบิตแมปแบบคู่กัน

จากภาพประกอบ 2-17 ถ้าต้องการทราบว่าแอททริบิวต์ A มีเรคอร์ดใดบ้างที่มีค่าเท่ากับ 15 สามารถทำได้โดย แทน A = v = 15

แทนค่า $v = 15$ จะได้ $r = \left\lceil \sqrt{2(15) + 2.25} - 0.5 \right\rceil = 6$

แทนค่า $v = 15$ และ $r = 6$ จะได้ $s = 15 - \frac{6(6-1)}{2} = 0$

อ่านบิตแมปเวกเตอร์ D^6 และ D^0 ดำเนินการ AND บิตต่อบิตระหว่างบิตแมปเวกเตอร์ D^6 และ D^0 ผลลัพธ์ที่ได้จากข้อ 5 คือเรคอร์ดที่ 4 ค่าของแอททริบิวต์เป็น 15 ดังแสดงในภาพประกอบ 2-18

การสอบถามแบบเป็นสมาชิกบนดัชนีบิตแมปแบบคู่กัน สามารถทำได้โดยนำรูปแบบการลงรหัสแต่ละค่ามาดำเนินการ OR เช่น จากภาพประกอบที่ 2-17 ถ้าต้องการทราบว่า บนแอททริบิวต์ A มีเรคอร์ดใดบ้างที่มีค่าเท่ากับ {3, 12, 1, 0} ผลลัพธ์ของการสอบถามคือ $(D^3 \wedge D^0) + (D^5 \wedge D^2) + (D^2 \wedge D^0) + (D^1 \wedge D^0)$ เรคอร์ดที่เป็นคำตอบของการสอบถามคือ 3, 7, 8, 11 ดังแสดงในภาพประกอบ 2-19(b)

RID	A
1	10
2	8
3	3
4	15
5	16
6	19
7	12
8	1
9	11
10	4
11	0
12	17
13	5

(a) ข้อมูล

D^6	D^0
0	1
0	0
0	1
1	1
1	0
1	0
0	0
0	1
0	0
0	0
0	1
1	0
0	0

=

$D^6 \wedge D^0$
0
0
0
1
0
0
0
0
0
0
0
0
0

(b) ผลการสอบถาม

ภาพประกอบ 2-18 การสอบถามแบบค่าเท่ากันของดัชนีบิตแมปแบบคู่กัน

RID	A
1	10
2	8
3	3
4	15
5	6
6	19
7	12
8	1
9	11
10	4
11	0
12	17
13	5

(a) ข้อมูล

$D^3 \wedge D^0$	
0	1
0	0
1	1
0	1
0	0
0	0
0	0
0	1
0	0
1	0
0	1
0	0
1	0

+

$D^5 \wedge D^2$	
1	0
0	1
0	0
0	0
0	0
0	0
1	1
0	1
1	0
0	0
0	0
0	1
0	1

+

$D^2 \wedge D^0$	
0	1
1	0
0	1
0	1
0	0
0	0
1	0
1	1
0	0
0	0
0	1
1	0
1	0

+

$D^1 \wedge D^0$	
0	1
0	0
0	1
0	1
1	0
0	0
0	0
0	1
1	0
1	0
1	1
0	0
0	0

=

result
0
0
1
0
0
0
1
1
0
0
1
0
0

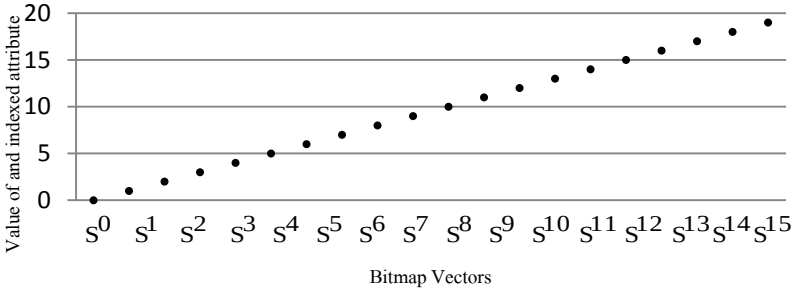
(b) ผลลัพธ์การสอบถามแบบเป็นสมาชิกบนดัชนีบิตแมปแบบคู่กัน

ภาพประกอบ 2-19 ผลการสอบถามแบบเป็นสมาชิกบนดัชนีบิตแมปแบบคู่กัน

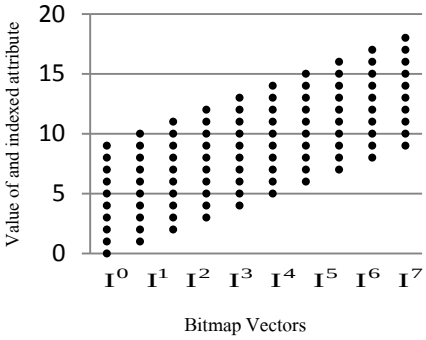
ข้อดีของดัชนีบิตแมปแบบคู่กัน ดัชนีบิตแมปแบบคู่กันจะมีการดำเนินการระหว่าง 2 บิตแมปเวกเตอร์เช่นเดียวกับดัชนีบิตแมปแบบกระจาย แต่จำนวนบิตแมปเวกเตอร์ที่ใช้น้อยกว่าดัชนีบิตแมปแบบพื้นฐาน แบบช่วง และดัชนีบิตแมปแบบกระจายเมื่อแอททริบิวต์มีค่าคาร์ดินอลิตี้

ที่เท่ากัน และเวลาที่ใช้ในการสอบถามแบบค่าเท่ากันมีการดำเนินการเพียงครั้งเดียวซึ่งใช้เวลาในการสอบถามน้อยกว่าดัชนีบิตแมปแบบเข้ารหัส

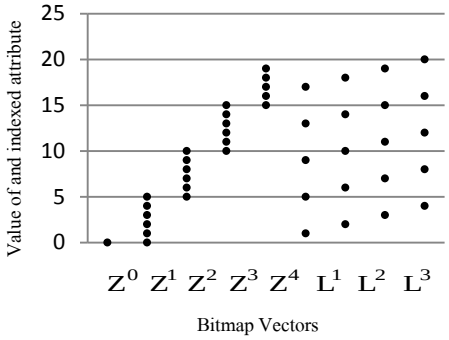
ข้อเสียของดัชนีบิตแมปแบบคู่กัน ใช้พื้นที่ในการจัดเก็บดัชนีมากกว่าดัชนีบิตแมปแบบเข้ารหัสในกรณีที่แอททริบิวต์มีค่าคาร์ดินอลิตี้สูงก็จะมีการใช้บิตแมปเวกเตอร์มากและใช้พื้นที่ในการเก็บดัชนีบิตแมปมากขึ้นด้วย



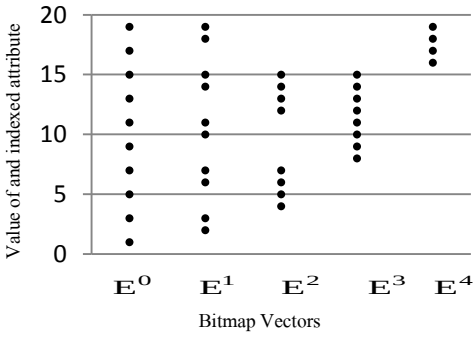
(a) Simple Encoding



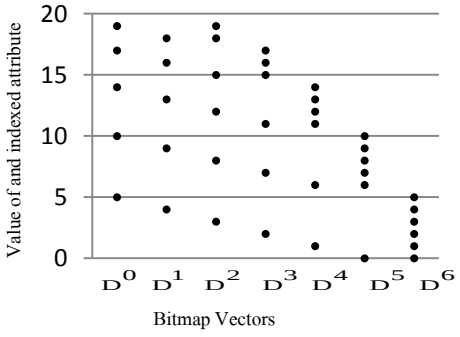
(b) Interval Encoding



(c) Scatter Encoding



(d) E-EBI



(e) Dual Encoding

ภาพประกอบ 2-20 รูปแบบการลงรหัสดัชนีบิตแมปทั้ง 5 แบบ (C=20)

2.4 เปรียบเทียบดัชนีบิตแมปทั้ง 5 แบบ

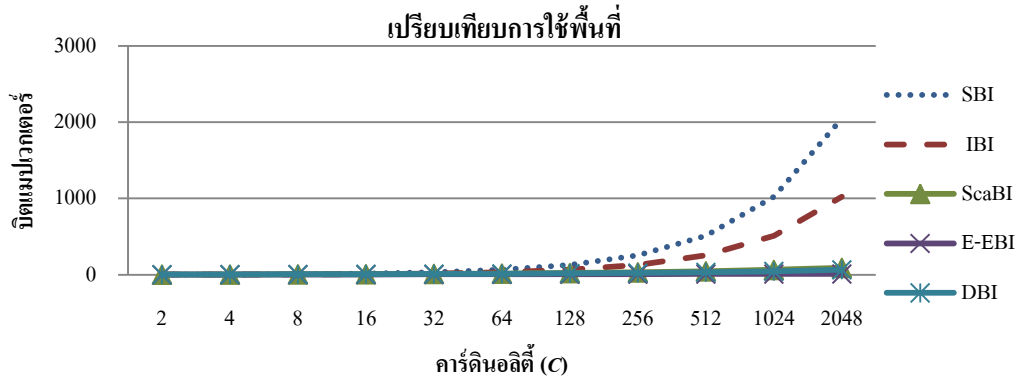
จากภาพประกอบ 2-20 ดัชนีบิตแมปแต่ละแบบจะมีรูปแบบการลงรหัสที่ไม่เหมือนกันแต่ดัชนีบิตแมปแบบเข้ารหัสกับ E-EBI มีการลงรหัสที่เหมือนกันและใช้บิตแมปเวกเตอร์เท่ากับจึงยกตัวอย่างเฉพาะ E-EBI จากรูปแบบการลงรหัสที่แสดงในภาพประกอบ 2-2 จำนวนของบิตแมปเวกเตอร์ที่ใช้ในดัชนีบิตแมปไม่เท่ากัน ทำให้มีวิธีการสอบถามที่แตกต่างกัน ดัชนีบิตแมปแต่ละแบบจึงมีข้อดีและข้อเสียที่แตกต่างกันด้วย จากดัชนีบิตแมปทั้ง 5 รูปแบบที่กล่าวมา สามารถทำการสรุปการใช้พื้นที่จัดเก็บดัชนี และเวลาที่ใช้ในการสอบถามข้อมูลแบบค่าเท่ากันและการสอบถามข้อมูลแบบสมาชิกของดัชนีบิตแมปทั้ง 5 แบบดังตาราง 2-2 แสดงการใช้พื้นที่และการสอบถามของดัชนีบิตแมป โดยจะพิจารณาจากจำนวนบิตแมปเวกเตอร์ที่ใช้ในการสร้างดัชนี และจำนวนครั้งที่ใช้ในการดำเนินตรรกะระหว่างบิตแมปเวกเตอร์ของการสอบถามแบบค่าเท่ากัน และการสอบถามแบบเป็นสมาชิก

ตาราง 2-2 การใช้พื้นที่และการสอบถามของดัชนีบิตแมป

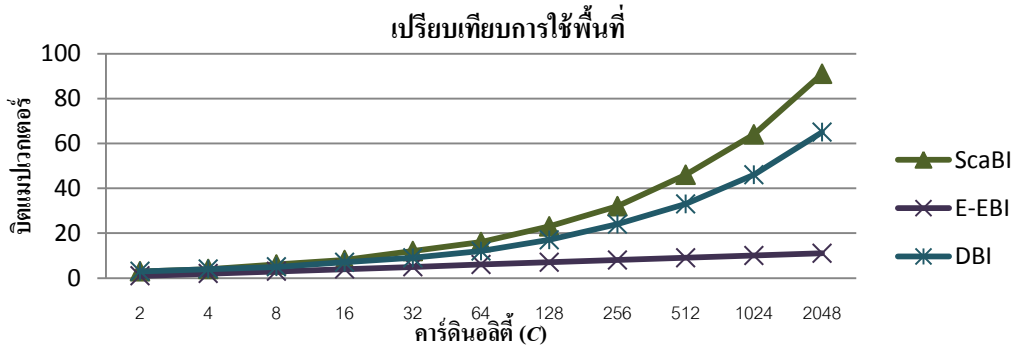
ดัชนีบิตแมป	พื้นที่	จำนวนบิตที่ถูกตรวจสอบระหว่างจำนวนครั้งในการดำเนินการตรรกะ	
	จำนวนบิตแมปเวกเตอร์ที่ใช้ในการสร้างดัชนี	การสอบถามแบบค่าเท่ากัน	การสอบถามแบบเป็นสมาชิก
Simple Bitmap Index (SBI)	C	1: 0	K: K-1 (OR)
Interval Bitmap Index (IBI)	$\lceil C/2 \rceil$	2: 2(1 AND, 1 NOT)	2K: 3K-1 (K AND ,K-1 OR, K NOT)
Scatter Bitmap Index (ScaBI)	$\lceil 2\sqrt{C} \rceil$	2: 1(1 AND)	2K: 2K-1 (K AND ,K-1 OR)
Encoded Bitmap Index (EBI)	$\lceil \log_2 C \rceil$	$\lceil \log_2 C \rceil$: use mapping table	K$\lceil \log_2 C \rceil$: use mapping table
Enhance EBI Bitmap Index (E-EBI)	$\lceil \log_2 C \rceil$	$\lceil \log_2 C \rceil$: 1 AND, $\lceil \log_2 C \rceil$ NOT	K-1 OR
Dual Bitmap Index (DBI)	$\lceil \sqrt{2C + 0.25} + 0.5 \rceil$	2: 1 (1 AND)	2K: 2K-1 (K AND ,K-1 OR)

จากตาราง 2-2 เปรียบเทียบในเรื่องของพื้นที่ที่ใช้ในการจัดเก็บดัชนี และจำนวนครั้งที่ใช้ในการดำเนินตรรกะระหว่างบิตแมปเวกเตอร์ของการสอบถามแบบค่าเท่ากัน และการสอบถามแบบเป็นสมาชิก ของดัชนีบิตแมปทั้ง 6 แบบ

การใช้พื้นที่จัดเก็บดัชนีสามารถเปรียบเทียบได้จากจำนวนบิตแมปเวกเตอร์ที่ใช้ ถ้าจำนวนของคาร์ดินอลิตี้ เท่ากับ C ดัชนีบิตแมปที่ใช้พื้นที่น้อยที่สุดคือ EBI และ E-EBI ซึ่งใช้พื้นเท่ากันคือ $\lceil \log_2 C \rceil$ ดัชนีบิตแมปที่ใช้พื้นที่น้อยรองลงมาคือ DBI ซึ่งใช้พื้นที่เท่ากับ $\lceil \sqrt{2C + 0.25} + 0.5 \rceil$ ดัชนีบิตแมปที่ใช้พื้นที่น้อยรองจากดัชนีบิตแมปแบบคู่กันคือ ScaBI คือ $\lceil 2\sqrt{C} \rceil$ และ IBI ซึ่งใช้พื้นที่เท่ากับ $\lceil C/2 \rceil$ ท้ายสุดคือ SBI ที่ใช้พื้นที่การจัดเก็บดัชนีมากที่สุดเท่ากับ C ในภาพประกอบ 2-21 แสดงกราฟเปรียบเทียบการใช้พื้นที่ของดัชนีบิตแมป 5 แบบ และภาพประกอบ 2-22 แสดงกราฟเปรียบเทียบการใช้พื้นที่ของ ScaBI, E-EBI และDBI



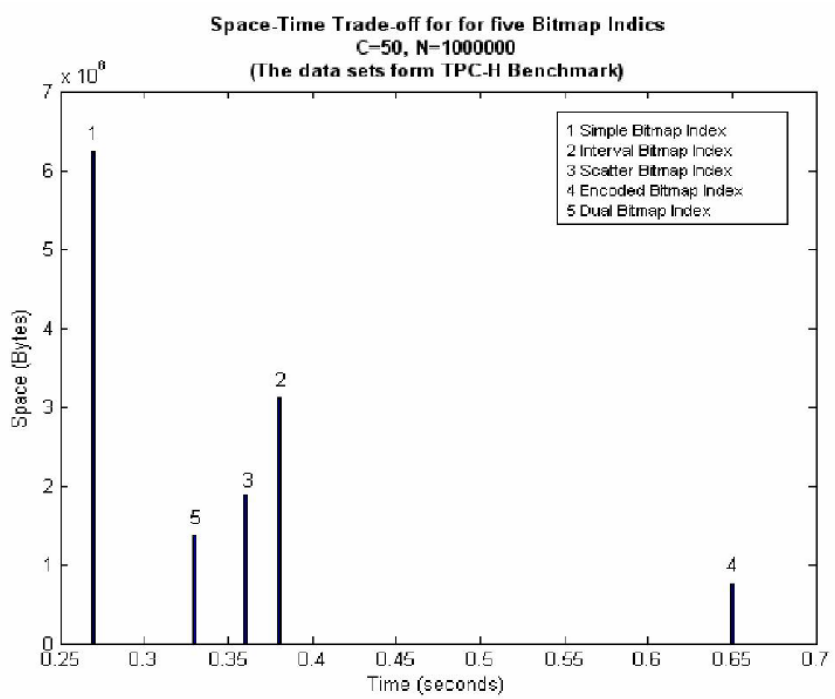
ภาพประกอบ 2-21 กราฟเปรียบเทียบการใช้พื้นที่ของดัชนีบิตแมป 5 แบบ



ภาพประกอบ 2-22 กราฟเปรียบเทียบการใช้พื้นที่ของ ScaBI, E-EBI และ DBI

ในการสอบถามแบบค่าเท่ากัน ดัชนีบิตแมปแบบพื้นฐานใช้เวลาในการสอบถามน้อยสุดเพราะมีการดำเนินการตรรกะเพียงแค่ครั้งเดียว ส่วนดัชนีบิตแมปแบบเข้ารหัสใช้เวลาในการสอบถามมากที่สุด และในการสอบถามแบบเป็นสมาชิก ดัชนีบิตแมปแบบพื้นฐานยังใช้เวลาในการสอบถามน้อยที่สุด และดัชนีบิตแมปแบบช่วง ดัชนีบิตแมปแบบกระจาย ดัชนีบิตแมปแบบคู่กันใช้เวลาในการสอบถามใกล้เคียงกัน ส่วนดัชนีบิตแมปแบบเข้ารหัสใช้เวลาในการสอบถามมากที่สุด

เมื่อพิจารณาทั้งพื้นและเวลาในการสอบถาม (Space-Time Trade-off) จะเห็นว่าดัชนีบิตแมปแบบเข้ารหัสใช้พื้นที่การจัดเก็บดัชนีน้อยสุด แต่ในเรื่องของเวลาการสอบถามจะเห็นว่าดัชนีบิตแมปแบบคู่กันมีเวลาการสอบถามที่ดีกว่าดัชนีบิตแมปแบบเข้ารหัส ดังแสดงในภาพประกอบ 2-23



ภาพประกอบ 2-23 กราฟเปรียบเทียบพื้นที่กับเวลาในการสอบถามของดัชนีบิตแมป 5 แบบ (Wattanakitrungroj and S. Vanichayobon, 2006)

จากที่กล่าวมาจะเห็นว่าดัชนีบิตแมปแบบคู่กันมีประสิทธิภาพในเรื่องของเวลาที่ใช้ในการสอบถามที่ดีกว่าดัชนีบิตแมปแบบเข้ารหัส หากดัชนีบิตแมปแบบคู่กันมีวิธีการลงรหัสที่ดีจะทำให้สามารถลดพื้นที่การจัดเก็บดัชนีให้ใช้พื้นที่จัดเก็บดัชนีให้มีประสิทธิภาพเพิ่มขึ้นด้วย ดังนั้นผู้วิจัยจึงมีแนวคิดใช้วิธีการเข้ารหัสในการลงรหัสให้กับดัชนีบิตแมปแบบคู่กันเพื่อนำไปสู่การเพิ่มประสิทธิภาพการใช้พื้นที่เก็บดัชนีได้

บทที่ 3

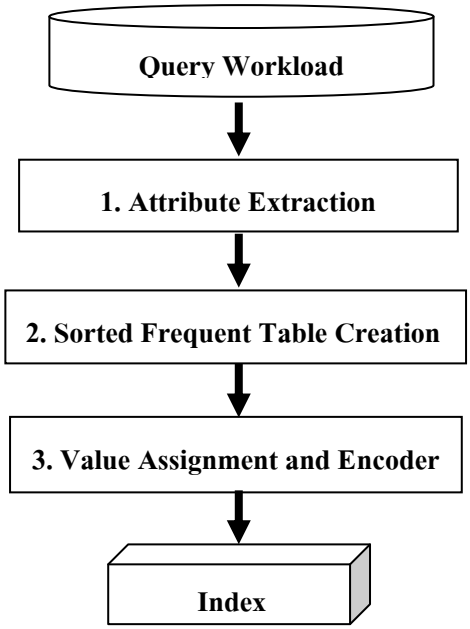
การเพิ่มประสิทธิภาพให้กับดัชนีบิตแมปแบบคู่กัน

บทที่ 2 ได้อธิบายถึงดัชนีบิตแมปแบบลงรหัสในแบบต่าง ๆ ดัชนีบิตแมปส่วนใหญ่จะเน้นสองเรื่องคือ การลดพื้นที่จัดเก็บดัชนี กับการลดเวลาการสอบถาม จากข้อเสียของดัชนีบิตแมปแบบคู่กันเมื่อ “แอททริบิวต์ที่มีคาร์ดินอลิตี้สูงมากๆก็จะมีการใช้พื้นที่ในการเก็บดัชนีบิตแมปที่เป็นบิต “0” จำนวนมาก” ผู้วิจัยจึงได้คิดทำดัชนีบิตแมปแบบใหม่ขึ้น โดยใช้พื้นที่จัดเก็บดัชนีให้มีประสิทธิภาพดีกว่าดัชนีบิตแมปแบบคู่กันแต่ยังคงรักษาประสิทธิภาพในการสอบถามสำหรับบางกรณี ดัชนีบิตแมปที่นำเสนอใช้สูตรการคำนวณหาค่า R เหมือนกับดัชนีบิตแมปแบบคู่กัน แต่ใช้สูตรการคำนวณหาค่า S และใช้วิธีการลงรหัสที่แตกต่างจากดัชนีบิตแมปแบบคู่กัน กล่าวคือ ใช้การลงรหัสค่า R และค่า S ด้วยเลขฐานสองเพื่อใช้พื้นที่ในการจัดเก็บดัชนีน้อยลง

วิทยานิพนธ์นี้นิยามดัชนีบิตแมปที่นำเสนอว่า Enhancing Dual Bitmap Index หรือ EDBI และในบทนี้จะกล่าวถึง การสร้าง EDBI การสอบถามข้อมูลแบบค่าเท่ากันบน EDBI การสอบถามแบบสมาชิกบน EDBI และสรุปข้อเด่นข้อด้อยของ EDBI

3.1 การสร้าง Enhancing Dual Bitmap Index (EDBI)

ขั้นตอนการสร้าง EDBI ประกอบด้วย 3 ขั้นตอนคือ 1) Attribute Extraction เป็นการสกัดค่าแอททริบิวต์ออกมาจาก Query Workload 2) Sorted Frequent Table Creation เป็นขั้นตอนการจัดเรียงความถี่ของแอททริบิวต์ที่สกัดมาได้ และ 3) Value Assignment and Encoder เป็นขั้นตอนการให้ค่าและการลงรหัสให้แอททริบิวต์ แสดงดังภาพประกอบ 3-1



ภาพประกอบ 3-1 ขั้นตอนการสร้าง EDBI

ขั้นตอนที่ 1: Attribute Extraction

เป็นขั้นตอนการสกัดค่าแอททริบิวต์ที่สนใจ (Attribute Extraction) เพื่อที่จะนำมาสร้างดัชนี จาก Query Workload โดยพิจารณาค่าที่สอบถามที่อยู่หลังเงื่อนไข WHERE โดยค่าที่ได้จะถูกสกัดออกมาในขั้นตอนนี้ แต่ละการสอบถามมีการสอบถามค่าของแอททริบิวต์ใดบ้าง ภาพประกอบ 3-2 แสดงตัวอย่างของ Query Workload 4 การสอบถามจากตาราง T ของแอททริบิวต์ type ที่มีค่า คาร์ดินอลิตี้ (*C*) 16 ค่า คือ {A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P} และตารางค่าของแอททริบิวต์ type ที่สกัดได้จาก Query Workload แสดงในภาพประกอบ 3-3

```
Q1: SELECT * FROM T WHERE type IN (A, E, G, D, C, M, N)
Q2: SELECT * FROM T WHERE type = A OR type = E OR type = D
Q3: SELECT * FROM T WHERE type = E OR type = P
Q4: SELECT * FROM T WHERE type = O
```

ภาพประกอบ 3-2 ตัวอย่าง Query Workload ของแอททริบิวต์ type

Query	ค่าของแอททริบิวต์ type
Q1	A, E, G, D, C, M, N
Q2	A, E, D
Q3	E, P
Q4	O

ภาพประกอบ 3-3 ตารางค่าของแอททริบิวต์

ขั้นตอนที่: 2 Sorted Frequent Table Creation

ในขั้นตอนนี้เป็นการเรียงค่าความถี่ (*f*) ของค่าของแอททริบิวต์ที่ถูกสกัดออกมาจากภาพประกอบ 3-3 แอททริบิวต์แต่ละตัวจะถูกคำนวณนับจำนวนความถี่ และความถี่ที่คำนวณได้จะถูกจัดเก็บไว้ในตารางแสดงความถี่ (Frequent Table) แสดงในภาพประกอบ 3-4 จากนั้นทำการเรียงค่าของแอททริบิวต์ type ตามความถี่จากค่ามากไปหาค่าน้อยและจัดเก็บค่าแอททริบิวต์ที่เรียงแล้วในตารางจัดเรียงความถี่ (Sorted Frequent Table) แสดงดังภาพประกอบ 3-5

type	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
Q1	1	0	1	1	1	0	1	0	0	0	0	0	1	1	0	0
Q2	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
Q3	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
Q4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
f	2	0	1	2	3	0	1	0	0	0	0	0	1	1	1	1

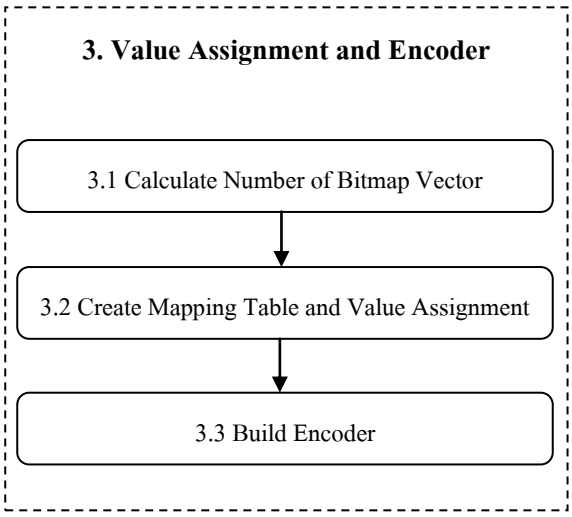
ภาพประกอบ 3-4 Frequent Table ของแอททริบิวต์ type

type	E	A	D	C	G	M	N	O	P	B	F	H	I	J	K	L

ภาพประกอบ 3-5 Sorted Frequent Table ของแอททริบิวต์ type

ขั้นตอนที่: 3 Value Assignment and Encoder

ขั้นตอนนี้เป็นขั้นตอนการกำหนดค่าให้กับแอททริบิวต์ที่ทำการจัดเรียงตามลำดับค่าความถี่แล้วจากขั้นตอนที่ 2 และทำการลงรหัส โดยในขั้นตอนที่ 3 นี้จะมีลำดับการทำงานย่อย 3 Steps.



ภาพประกอบ 3-6 ลำดับขั้นตอนของ Value Assignment and Encoder

ภาพประกอบ 3-6 เป็นการแสดงลำดับขั้นตอนย่อยของ Value Assignment and Encoder ลำดับแรก Calculate Number of Bitmap Vector เป็นการคำนวณหาจำนวนบิตแมปเวกเตอร์ที่ใช้ในการลงรหัสดัชนีบิตแมป ลำดับที่สอง Create Mapping Table and Value Assignment เป็นการสร้างตาราง Mapping Table เพื่อเก็บค่าของแอททริบิวต์และค่าตัวเลขที่ใช้แทนให้แต่ละค่าของแอททริบิวต์ ซึ่งตัวเลขที่นำมาแทนให้กับค่าแอททริบิวต์แต่ละตัวนั้นจะมีการคำนวณให้ได้ตัวเลขที่เหมาะสมเพื่อใช้ในการลงรหัสและใช้ในการสอบถามเพราะค่าที่เก็บในตาราง Mapping Table จะไม่มีการเปลี่ยนแปลง และลำดับที่สาม Build Encoder เป็นการเอาตัวเลขที่แทนให้แต่ละค่าของแอททริบิวต์จากตาราง Mapping Table มาลงรหัสดัชนีบิตแมปด้วยเลขฐานสองและเก็บไว้ในตารางดัชนี

Step 1. คำนวณหาจำนวนบิตที่ใช้แทนค่าแต่ละค่า

$$nB = 2\left\lceil \left(\log_2 \left\lceil \sqrt{2C + 0.25} + 0.5 \right\rceil \right) \right\rceil \qquad (1)$$

Step 2. สร้าง EDBI Mapping Table เพื่อลดเวลาในการคำนวณในกระบวนการสร้างดัชนีและการสอบถาม เก็บค่าที่ใช้แทนให้แต่ละแอททริบิวต์

2.1. คำนวณหาค่าตัวเลขที่ใช้แทนค่าที่ถูกสอบถามบ่อยสุด

$$V_0 = \frac{((2^{nB/2} - 1) + 0.5)^2}{2} - 1.125 \qquad (2)$$

2.2. กำหนดค่าให้กับค่าที่ถูกสอบถามบ่อยตามลำดับโดยที่

$$V_i = V_0 - i \quad \text{สำหรับ} \quad i = 1,2,\ldots,C\text{-}1 \qquad (3)$$

2.3. นำค่า V_i ที่ได้มาคำนวณหาค่า R_i และ S_i สำหรับ $i = 0,1,\ldots,C\text{-}1$ โดยใช้สมการ

$$R_i = \left\lceil \sqrt{2V_i + 2.25} - 0.5 \right\rceil \qquad (4)$$

$$S_i = (R_i - 1) + \frac{R_i(R_i - 1)}{2} - V_i \qquad (5)$$

Step 3. ทำการลงรหัสโดยแทนค่า R_i และ S_i ด้วยเลขฐานสองโดยใช้จำนวนบิตเท่า ๆ กันคือ $\frac{nB}{2}$ บิต รูปแบบของบิตแมปเวกเตอร์ คือ $r^{(nB/2)-1},\ldots,r^2,r^1,r^0,s^{(nB/2)-1},\ldots,s^2,s^1,s^0$

ภาพประกอบ 3-7 ขั้นตอนวิธีการกำหนดค่าและการลงรหัส EDBI

ภาพประกอบ 3-7 แสดงขั้นตอนวิธีการกำหนดค่าและการลงรหัส EDBI อย่างละเอียด สามารถอธิบายขั้นตอนวิธีในการลงรหัสในแต่ละข้อได้ดังต่อไปนี้

Step 1. เป็นขั้นตอนการคำนวณหาจำนวนบิต (*nB*) ที่ใช้แทนค่าแต่ละค่าของแอททริบิวต์ มีที่มาจากดัชนีบิตแมปแบบคู่กันซึ่งใช้บิตแมปเวกเตอร์เท่ากับ $n = \left\lceil \sqrt{2(16) + 0.25} + 0.5 \right\rceil$ และมีการลงรหัสตามตำแหน่งของ r และ s โดย ค่า r มีค่าอยู่ในช่วง $1 \le r \le n - 1$ และค่า s มีค่าอยู่ในช่วง $0 \le s < r$ ดังนั้น ค่า r และ s จึงมีค่าประมาณ n และเมื่อนำค่าของ r และ s มาลงรหัสในรูปแบบของเลขฐานสองจะใช้จำนวนบิตในการลงรหัส r และ s

ประมาณ $log_2 n + log_2 n$ เมื่อ n มาจาก $n = \left\lceil \sqrt{2(16)+0.25}+0.5 \right\rceil$ ดังนั้นสามารถคำนวณจำนวนบิตในการลงรหัสได้จากสูตร $2\left(log_2 \left\lceil \sqrt{2(16)+0.25}+0.5 \right\rceil \right)$

Step 2. เป็นขั้นตอนการสร้าง EDBI Mapping Table เพื่อเก็บ V_i (ค่าตัวเลขที่แทนให้กับแค่ละค่าของแอททริบิวต์), R_i และ S_i

2.1. เป็นการคำนวณหาค่าตัวเลขที่เหมาะสม ที่ใช้แทนให้กับแอททริบิวต์ที่ถูกสอบถามบ่อยที่สุด โดยใช้สมการที่ (2) กำหนดให้ V_0 แทนค่าแอททริบิวต์ที่ถูกสอบถามบ่อยสุด โดยค่า V_0 คำนวณจาก $\frac{nB}{2}$ เพื่อให้ได้ตัวเลขที่เหมาะสมและไปคำนวณหาค่า R_0 และ S_0 ทำให้ค่า R_0 เมื่อแปลงเป็นเลขฐานสองแล้วจะได้เป็นบิต 1 ทั้งหมด และ S_0 ก็จะเป็นบิต 0 ทั้งหมด

2.2. เป็นขั้นตอนการแทนค่าตัวเลขให้กับทุก ๆ ค่าของแอททริบิวต์ โดยใช้สมการที่ (3) กล่าวคือ V_i คำนวณได้ค่า $V_0 - i$ สำหรับ $i = 1,2,...,C\text{-}1$ V_1 แทนค่าแอททริบิวต์ที่ถูกสอบถามบ่อยรองลงมา และให้ V_{C-1} แทนค่าตัวเลขแอททริบิวต์ที่ถูกสอบถามน้อยที่สุด

2.3 เป็นขั้นตอนการคำนวณหาค่า R_i และ S_i จากค่า V_i โดยใช้สมการที่ (4) (Wattanakitrungroj and S. Vanichayobon, 2006) และสมการที่ (5) ตามลำดับ จากนั้นทำการเก็บค่า V_i , R_i และ S_i ไว้ในตาราง EDBI Mapping

Step 3. เป็นขั้นตอนการสร้างดัชนี EDBI โดยการลงรหัสค่า R_i และ S_i ด้วยเลขฐานสองด้วยจำนวนบิตที่เท่า ๆ กันคือ $\frac{nB}{2}$ บิต และบันทึกเลขฐานสองที่ได้ลงในตารางดัชนี

ยกตัวอย่างการสร้าง EBDI บนแอททริบิวต์ type จากภาพประกอบ 3-5 ที่มีค่าคาร์-ดินอลิตี้เท่ากับ 16 สามารถทำได้โดย

Step 1. คำนวณหาจำนวนบิตที่ใช้แทนค่าแต่ละค่าของแอททริบิวต์ แทน $C = 16$ ในสมการ (1)

$$nB = 2\left\lceil \left(\log_2 \left\lceil \sqrt{2(16)+0.25}+0.5 \right\rceil \right) \right\rceil$$

$nB = 6$ (บิตแมปเวกเตอร์ประกอบด้วย $r^2, r^1, r^0, s^2, s^1, s^0$)

Step 2. สร้าง EDBI Mapping Table สำหรับเก็บค่า V_i , R_i และ S_i เพื่อลดเวลาการคำนวณในกระบวนการสร้างดัชนีและการสอบถาม

2.1. คำนวณหาค่าตัวเลขที่ใช้แทนค่าแอททริบิวต์ที่ถูกสอบถามบ่อยสุด แทนค่า $nB = 6$ ที่ได้จาก Step 1. ในสมการที่ (2)

$$V_0 = \frac{((2^{nB/2} - 1) + 0.5)^2}{2} - 1.125$$

$$V_0 = 27$$

จากตารางที่ 3-5 จะเห็นว่า “E” เป็นค่าของแอททริบิวต์ที่ถูกสอบถามบ่อยที่สุดดังนั้น V_0 จึงถูกแทนให้กับค่า “E”

2.2. กำหนดค่าตัวเลขให้กับค่าของแอททริบิวต์ที่ถูกสอบถามบ่อยสุดไปจนถึงค่าของแอททริบิวต์ที่ถูกสอบถามน้อยสุด ตามลำดับจากตารางที่ 3-5 โดยที่ $V_i = V_0 - i$ สำหรับ $i = 1,2,...,C\text{-}1$ จะได้ “A” แทนด้วย V_1=26, “D” แทนด้วย V_2=25, “M” แทนด้วย V_5=22 เป็นต้น

2.3. คำนวณหาค่า R_i และ S_i จากค่า V_i

ตัวอย่างการหาค่า R_0 และ S_0 แทน V_0=27 (type = “E”) ในสมการ (4) และ (5)

$$R_0 = \left\lceil \sqrt{2V_0 + 2.25} - 0.5 \right\rceil \qquad S_0 = (R_0 - 1) + \frac{R_0(R_0 - 1)}{2} - V_0$$

$$R_0 = \left\lceil \sqrt{2(27) + 2.25} - 0.5 \right\rceil \qquad S_0 = (7 - 1) + \frac{7(7 - 1)}{2} - 27$$

$$R_0 = 7 \qquad S_0 = 0$$

แสดงว่าที่ V_0=27 จะได้ $R_0 = 7$ และ $S_0 = 0$

ตัวอย่างการหาค่า R_5 และ S_5 แทน V_5=22 (type = “M”) ในสมการ (4) และ (5)

$$R_5 = \left\lceil \sqrt{2V_5 + 2.25} - 0.5 \right\rceil \qquad S_5 = (R_5 - 1) + \frac{R_5(R_5 - 1)}{2} - V_5$$

$$R_5 = \left\lceil \sqrt{2(22) + 2.25} - 0.5 \right\rceil \qquad S_5 = (7 - 1) + \frac{7(7 - 1)}{2} - 22$$

$$R_5 = 7 \qquad S_5 = 5$$

แสดงว่าที่ V_5=22 จะได้ $R_5 = 7$ และ $S_5 = 5$

ภาพประกอบ 3-8 แสดงการกำหนดค่าตัวเลขให้กับค่าแอททริบิวต์ type ทั้ง 16 ค่า และค่า R_i และ S_i ที่ได้จากสมการ (4) และ(5)

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
type	E	A	D	C	G	M	N	O	P	B	F	H	I	J	K	L
V_i	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12
R_i	7	7	7	7	7	7	7	6	6	6	6	6	6	5	5	5
S_i	0	1	2	3	4	5	6	0	1	2	3	4	5	0	1	2

ภาพประกอบ 3-8 EDBI Mapping Table ของแอททริบิวต์ type

Step 3. ทำการลงรหัสโดยแทนค่า R_i และ S_i ด้วยเลขฐานสองโดยใช้จำนวนบิตเท่า ๆ กันคือ $\frac{nB}{2} = 3$ บิต

ตัวอย่างจาก EDBI Mapping Table ถ้าต้องการลงรหัสค่า E พบว่า $R = 7$ และ $S = 0$ สามารถลงรหัสได้ 111 000 ถ้าต้องการลงรหัสค่า F พบว่า $R = 6$ และ $S = 3$ สามารถลงรหัสได้ 110 011 และ ถ้าต้องการลงรหัสค่า L พบว่า $R = 5$ และ $S = 2$ สามารถลงรหัสได้ 101 010 ภาพประกอบ 3-9 (b) แสดงการลงรหัส EDBI ของแอททริบิวต์ type

จากภาพประกอบ 3-9(b) จะเห็นว่าค่า E ซึ่งเป็นค่าที่ถูกสอบถามบ่อยที่สุด จากขั้นตอนการเรียงค่าความถี่ รูปแบบของการลงรหัสบิตแมปเวกเตอร์ r^i จะลงรหัสด้วยบิต 1 ทั้งหมด และบิตแมปเวกเตอร์ s^i ถูกลงรหัสเป็นบิต 0 ทั้งหมด ทั้งนี้เพื่อลดการอ่านบิตแมปเวกเตอร์สำหรับค่าที่ถูกสอบถามบ่อยที่สุด

RID	type
1	K
2	F
3	G
4	P
5	N
6	B
7	O
8	E
9	A
10	F
11	D
12	C
13	L
14	M

(a) แอททริบิวต์ type ในตาราง T

r^2	r^1	r^0	s^2	s^1	s^0
1	0	1	0	0	1
1	1	0	0	1	1
1	1	1	1	0	0
1	1	0	0	0	1
1	1	1	1	1	0
1	1	0	0	1	0
1	1	0	0	0	0
1	1	1	0	0	0
1	1	1	0	0	1
1	1	0	0	1	1
1	1	1	0	1	0
1	1	1	0	1	1
1	0	1	0	1	0
1	1	1	1	0	1

(b) EDBI

ภาพประกอบ 3-9 ตัวอย่างการลงรหัส EDBI บนแอททริบิวต์ type (C=16)

Step 1. ทำการ lookup ค่า R_i และ S_i ของค่าที่ต้องการสอบถาม type = “V” จาก EDBI Mapping Table

Step 2. กำหนดให้ PATTERN คือ เลขฐานสองของค่า S_i ที่เรียงต่อจากค่า R_i

Step 3. *if* $(S_i = 0)$

ทำการจัดเก็บเรคอร์ดที่ตรงเงื่อนไข $\bigvee_{i=0}^{(nB/2)-1} s^i = 0$

else

ทำการจัดเก็บเรคอร์ดที่ตรงเงื่อนไข $(r^i \wedge s^i = 1)$ โดยที่ $\left(0 \le i, j \le \frac{nB}{2} - 1\right)$ สำหรับค่าตำแหน่ง i, j ที่มีค่า 1 ใน PATTERN

Step 4. นำ PATTERN มาดำเนินการตรรกะ XOR ($\oplus$) กับผลลัพธ์ที่ได้จาก Step 3. เลือกเรคอร์ดที่มีค่าเท่ากับ 0

ภาพประกอบ 3-10 ขั้นตอนวิธีการสอบถามบน EDBI

3.2 การสอบถามข้อมูลแบบค่าเท่ากันโดยใช้ EDBI

ในขั้นตอนวิธีการสอบถามจากภาพประกอบ 3-10 เป็นขั้นตอนวิธีการสอบถามแบบค่าเท่ากันบน EDBI โดยมีวิธีการแต่ละขั้นตอนดังต่อไปนี้

Step 1. เป็นขั้นตอนการค้นหาข้อมูลจากตารางโดยใช้การดูคีย์เวิร์ด เพื่อลดเวลาการคำนวณที่นำไปสอบถาม ในขั้นตอนการสอบถามจะเรียกแทนวิธีการค้นหาแบบที่กล่าวมาว่าเป็นการ lookup เช่น การ lookup ค่า F จากตาราง Mapping Table (ภาพประกอบ 3-8) จะใช้ F เป็นคีย์เวิร์ดในการค้นหา ดังนั้นค่าที่ได้จากการ lookup คือ $R_{10} = 6$ และ $S_{10} = 3$ เป็นต้น

Step 2. เป็นขั้นตอนการแปลงค่า R_i และ S_i ที่ได้จาก Step1. ให้อยู่ในรูปแบบของเลขฐานสอง (ขนาด $\frac{nB}{2}$ บิต) โดยกำหนดให้เป็น PATTERN โดยการให้เลขฐานสองของ S ต่อกับเลขฐานสองของ R เพื่อนำ PATTERN ที่ได้ไปใช้ในการสอบถาม

Step 3. เป็นเงื่อนไขที่ใช้ในการสอบถามเพื่อให้การสอบถามนั้นไม่จำเป็นต้องอ่านบิตแมปเวกเตอร์ทุก ๆ บิตแมปเวกเตอร์ โดยแบ่งออกเป็น 2 เงื่อนไขคือ เมื่อ $S_i = 0$ และ $S_i \neq 0$

เงื่อนไขที่ 1: ถ้า $S_i = 0$ จะพิจารณาบิตแมปเวกเตอร์ที่เป็น s^i ทุกตัว โดยจะจัดเก็บเรคอร์ดที่ตรงกับเงื่อนไข $s^0 \vee s^1 \vee s^2 \ldots \vee s^{(nB/2)-1} = 0$ (เลือกเรคอร์ดที่ทุกบิตเท่ากับ 0)

เงื่อนไขที่ 2: ถ้า $S_i \neq 0$ จะพิจารณาบิตแมปเวกเตอร์ r^i และ s^i ที่ตรงกับเงื่อนไข $r^i = 1$ (บิตแมปเวกเตอร์ r^i ที่เท่ากับ 1) และ $(s^j = 1)$ (บิตแมปเวกเตอร์ s^j ที่เท่ากับ 1)

Step 4. เป็นการนำ PATTERN ไปดำเนินการตรรกะ XOR กับผลลัพธ์ที่ได้จาก Step 3. แล้วเลือกเรคอร์ดที่เท่ากับ 0 คือคำตอบของการสอบถาม

การสอบถามแบบค่าเท่ากันบน EDBI มีรูปแบบของการสอบถามคือ type = "V" หมายถึงการสอบถามว่าบนแอททริบิวต์ type มีค่าใดบ้างที่เท่ากับ V

ตัวอย่างที่ 1 การสอบถามแบบค่าเท่ากันว่าบนแอทริบิวต์ type มีเรคอร์ดใดบ้างที่มีค่าเท่ากับ E ทำได้ด้วยวิธีการดังนี้

1. ค้นหา R และ S ของ E จาก EDBI Mapping Table (ภาพประกอบ 3-8) ได้ $R = 7$ และ $S = 0$ แปลง R และ S ให้อยู่ในรูปแบบเลขฐานสองขนาด $\frac{nB}{2}$ ได้ $R = 111$ และ $S = 000$
2. สร้าง PATTERN โดยเอาเลขฐานสองมาต่อกัน ได้เป็น 111000 ($R = 111$ $S = 000$)
3. เลือกเงื่อนไขการสอบถามในที่นี้ $S = 0$ ดังนั้น เลือกเรคอร์ดที่ตรงเงื่อนไข $(s^0 \vee s^1 \vee s^2 = 0)$ โดยเรคอร์ดที่ตรงตามเงื่อนไขคือ RID 7 และ RID 8 ดังที่แสดงในภาพประกอบ 3-11 (a)

4. นำ PATTERN ไปดำเนินการตรรกะ XOR ผลลัพธ์ที่ได้จากข้อ 3 คำตอบที่ได้คือเรคอร์ดที่ 8 แสดงในดังภาพประกอบ 3-11(b)

ตัวอย่างที่ 2 การสอบถามแบบค่าเท่ากันว่าบนแอทริบิวต์ type มีเรคอร์ดใดบ้างที่มีค่าเท่ากับ F ทำได้ด้วยวิธีการดังนี้

1. ค้นหา R และ S ของ F จาก EDBI Mapping Table (ภาพประกอบ 3-8) ได้ $R = 6$ และ $S = 3$ แปลง R และ S ให้อยู่ในรูปแบบเลขฐานสองขนาด $\frac{nB}{2}$ ได้ $R = 110$ และ $S = 011$
2. สร้าง PATTERN โดยเอาเลขฐานสองมาต่อกัน ได้เป็น 110011 ($R = 110$ $S = 011$ ตำแหน่งที่ 1 และ 2 ของ R และ ตำแหน่งที่ 0 และ 1 ของ S มีค่าเท่ากับ 1)
3. เลือกเงื่อนไขในการสอบถามในที่นี้ $S \neq 0$ ดังนั้น เลือกเรคอร์ดที่ตรงเงื่อนไข $((r^2 \wedge r^1 \wedge s^1 \wedge s^0) = 1)$ โดยเรคอร์ดที่ตรงตามเงื่อนไขคือ RID 2, RID 10 และ RID 12 ดังที่แสดงในภาพประกอบ 3-12 (a)
4. นำ PATTERN ไปดำเนินการตรรกะ XOR ผลลัพธ์ที่ได้จากข้อ 3 คำตอบที่ได้ คือ เรคอร์ด ที่ 2 และ10 แสดงดังภาพประกอบ 3-12(b)

ตัวอย่างที่ 3 การสอบถามแบบค่าเท่ากันว่าบนแอทริบิวต์ type มีเรคอร์ดใดบ้างที่มีค่าเท่ากับ L ทำได้ด้วยวิธีการดังนี้

1. ค้นหา R และ S ของ L จาก EDBI Mapping Table (ภาพประกอบ 3-8) ได้ $R = 5$ และ $S = 2$ แปลง R และ S ให้อยู่ในรูปแบบเลขฐานสองขนาด $\frac{nB}{2}$ ได้ $R = 101$ และ $S = 010$
2. สร้าง PATTERN โดยเอาเลขฐานสองมาต่อกัน ได้เป็น 101010 ($R = 101$ $S = 010$ ตำแหน่งที่ 0 และ 2 ของ R และ ตำแหน่งที่ 1 ของ S มีค่าเท่ากับ 1)
3. เลือกเงื่อนไขในการสอบถามในที่นี้ $S \neq 0$ ดังนั้น เลือกเรคอร์ดที่ตรงเงื่อนไข $((r^2 \wedge r^0 \wedge s^1) = 1)$ โดยเรคอร์ดที่ตรงตามเงื่อนไขคือ RID 5, RID 11, RID 12 และ RID 13 ดังที่แสดงในภาพประกอบ 3-13 (a)
4. นำ PATTERN ไปดำเนินการตรรกะ XOR ผลลัพธ์ที่ได้จากข้อ 3 คำตอบที่ได้คือเรคอร์ดที่ 13 แสดงดังภาพประกอบ 3-13(b)

จากตัวอย่างการสอบถามแบบค่าเท่ากันทั้งสามตัวอย่างจะเห็นว่ามีการดำเนินการตรรกะในการสอบถาม 2 อย่างคือ การอ่านบิตแมปเวกเตอร์ตามเงื่อนไข และการดำเนินการตรรกะ

XOR การอ่านบิตแมปเวกเตอร์เป็นการดำเนินการกับคอลัมน์ตามเงื่อนไขการสอบถามเพื่อสกัดเรคอร์ดที่ตรงตามเงื่อนไข จากนั้นจึงดำเนินการตรรกะ XOR ระหว่าง PATTERN กับ เรคอร์ดที่เหลือเพื่อหาผลลัพธ์ของการสอบถาม

RID	type
1	K
2	F
3	G
4	P
5	N
6	B
7	O
8	E
9	A
10	F
11	D
12	C
13	L
14	M

แอททริบิวต์ type ในตาราง T

r^2	r^1	r^0	s^2	s^1	s^0
1	0	1	0	0	1
1	1	0	0	1	1
1	1	1	1	0	0
1	1	0	0	0	1
1	1	1	1	1	0
1	1	0	0	1	0
1	1	0	0	0	0
1	1	1	0	0	0
1	1	1	0	0	1
1	1	0	0	1	1
1	1	1	0	1	0
1	1	1	0	1	1
1	0	1	0	1	0
1	1	1	1	0	1

EDBI

(a) เรคอร์ดที่ตรงเงื่อนไข ($s^0 \vee s^1 \vee s^2 = 0$)

RID	type
7	O
8	E

แอททริบิวต์ type ในตาราง T

r^2	r^1	r^0	s^2	s^1	s^0		PATTERN		result
1	1	0	0	0	0	⊕	111000	=	001000
1	1	1	0	0	0		111000		000000

EDBI

result

(b) ผลลัพธ์ตรรกะ XOR

ภาพประกอบ 3-11 ผลลัพธ์จากการสอบถามโดยใช้ EDBI

RID	type
1	K
2	F
3	G
4	P
5	N
6	B
7	O
8	E
9	A
10	F
11	D
12	C
13	L
14	M

แอททริบิวต์ type ในตาราง T

r^2	r^1	r^0	s^2	s^1	s^0
1	0	1	0	0	1
1	1	0	0	1	1
1	1	1	1	0	0
1	1	0	0	0	1
1	1	1	1	1	0
1	1	0	0	1	0
1	1	0	0	0	0
1	1	1	0	0	0
1	1	1	0	0	1
1	1	0	0	1	1
1	1	1	0	1	0
1	1	1	0	1	1
1	0	1	0	1	0
1	1	1	1	0	1

EDBI

(a) เรคอร์ดที่ตรงเงื่อนไข $((r^2 \wedge r^1 \wedge s^1 \wedge s^0) = 1)$

RID	type
2	F
10	F
12	C

แอททริบิวต์ type ในตาราง T

r^2	r^1	r^0	s^2	s^1	s^0
1	1	0	0	1	1
1	1	0	0	1	1
1	1	1	0	1	1

EDBI

$\oplus$

PATTERN
110011
110011
110011

=

result
000000
000000
001000

result

(b) ผลลัพธ์ตรรกะ XOR

ภาพประกอบ 3-12 จากการสอบถามโดยใช้ EDBI

RID	type
1	K
2	F
3	G
4	P
5	N
6	B
7	O
8	E
9	A
10	F
11	D
12	C
13	L
14	M

แอททริบิวต์ type ในตาราง T

r^2	r^1	r^0	s^2	s^1	s^0
1	0	1	0	0	1
1	1	0	0	1	1
1	1	1	1	0	0
1	1	0	0	0	1
1	1	1	1	1	0
1	1	0	0	1	0
1	1	0	0	0	0
1	1	1	0	0	0
1	1	1	0	0	1
1	1	0	0	1	1
1	1	1	0	1	0
1	1	1	0	1	1
1	0	1	0	1	0
1	1	1	1	0	1

EDBI

(a) เรคอร์ดที่ตรงเงื่อนไข $((r^2 \wedge r^0 \wedge s^1) = 1)$

RID	type
5	N
11	D
12	C
13	L

แอททริบิวต์ type ในตาราง T

r^2	r^1	r^0	s^2	s^1	s^0
1	1	1	1	1	0
1	1	1	0	1	0
1	1	1	0	1	1
1	0	1	0	1	0

EDBI

PATTERN
101010
101010
101010
101010

$\oplus$ =

result
010100
010000
010001
000000

result

(b) ผลลัพธ์ตรรกะ XOR

ภาพประกอบ 3-13 ผลลัพธ์จากการสอบถามโดยใช้ EDBI

3.3 การสอบถามข้อมูลแบบเป็นสมาชิกโดยใช้ EBDI

การสอบถามข้อมูลแบบเป็นสมาชิกมีรูปแบบเงื่อนไข คือ " type in $\{V^1, V^2, .., V^n\}$" คือการสอบถามว่าบนแอททริบิวต์ type มีเรคอร์ดใดบ้างที่มีค่าเท่ากับ V การสอบถามข้อมูลแบบเป็นสมาชิก EDBI จะใช้การดำเนินการตรรกะ OR กับรูปแบบการลงรหัสแต่ละค่า โดยวิธีการสอบถามข้อมูลแบบค่าเท่ากันทำได้โดยการ อ่าน EDBI Mapping Table (ภาพประกอบ 3-8) เพื่อดูว่าค่าที่ต้องการสอบถามถูกแทนด้วยตัวเลขใด (R_i และ S_i) และจึงแปลงค่าตัวเลขที่ได้ให้อยู่ในเลขฐานสองขนาด $\frac{nB}{2}$ บิต โดยใช้เลขฐานสองของ S ต่อกับเลขฐานสองของ R รูปแบบในการลงรหัสจะกำหนดให้ r^i และ s^i แทนบิต 1 ส่วน $\overline{r^i}$ และ $\overline{s^i}$ แทนบิต 0 เมื่อได้รูปแบบการลงรหัสแล้ว ทำการลดรูปฟังก์ชันการเข้าถึงข้อมูลจากรูแบบการลงรหัส และตรวจสอบบิตแมปเวกเตอร์จากผลลัพธ์ที่ได้จากการลดรูปฟังก์ชัน ขั้นตอนวิธีการสอบถามแบบสมาชิกแสดงในภาพประกอบ 3-14

Step 1. อ่านค่า R_i และ S_i ของค่าที่ต้องการสอบถามจาก EDBI Mapping Table

Step 2. ดำเนินการลดรูปของฟังก์ชันการเข้าถึงข้อมูลได้

Step 3. ตรวจสอบบิตแมปเวกเตอร์ที่ได้จากการลดรูปของฟังก์ชันคำตอบคือเรคอร์ดที่เท่ากับ 1

ภาพประกอบ 3-14 ขั้นตอนวิธีการสอบถามแบบสมาชิกบน EDBI

ตัวอย่างถ้าต้องการสอบถามข้อมูลแบบเป็นสมาชิกตามเงื่อนไขการสอบถาม Q1 ในภาพประกอบ 3-2 โดยใช้รูปแบบการลงรหัสจากภาพประกอบ 3-12 สามารถทำได้ดังนี้

1. อ่านค่า R_i และ S_i ของค่าที่ต้องการสอบถามจาก EDBI Mapping Table

A แทน $R_1 = 7$ และ $S_1 = 1$ ด้วย 111001 รูปแบบการลงรหัส $r^2 \wedge r^1 \wedge r^0 \wedge \overline{s^2} \wedge \overline{s^1} \wedge s^0$

E แทน $R_0 = 7$ และ $S_0 = 0$ ด้วย 111000 รูปแบบการลงรหัส $r^2 \wedge r^1 \wedge r^0 \wedge \overline{s^2} \wedge \overline{s^1} \wedge \overline{s^0}$

G แทน $R_4 = 7$ และ $S_4 = 4$ ด้วย 111100 รูปแบบการลงรหัส $r^2 \wedge r^1 \wedge r^0 \wedge s^2 \wedge \overline{s^1} \wedge \overline{s^0}$

D แทน $R_2 = 7$ และ $S_2 = 2$ ด้วย 111010 รูปแบบการลงรหัส $r^2 \wedge r^1 \wedge r^0 \wedge \overline{s^2} \wedge s^1 \wedge \overline{s^0}$

C แทน $R_3 = 7$ และ $S_3 = 3$ด้วย 111011 รูปแบบการลงรหัส $r^2 \wedge r^1 \wedge r^0 \wedge \overline{s^2} \wedge s^1 \wedge s^0$

M แทน $R_5 = 7$ และ $S_5 = 5$ ด้วย 111101 รูปแบบการลงรหัส $r^2 \wedge r^1 \wedge r^0 \wedge s^2 \wedge \overline{s^1} \wedge s^0$

N แทน $R_6 = 7$ และ $S_6 = 6$ ด้วย 111110 รูปแบบการลงรหัส $r^2 \wedge r^1 \wedge r^0 \wedge s^2 \wedge s^1 \wedge \overline{s^0}$

จากรูปแบบการลงรหัสลักษณะของบิตแมปเวกเตอร์ ในการสอบถามแบบสมาชิก จะทำการดำเนินการตรรกะ OR (ในที่นี้จะใช้สัญลักษณ์ + แทนการดำเนินตรรกะ OR) จะได้ฟังก์ชันการเข้าถึงข้อมูลดังต่อไปนี้

$$r^2 \wedge r^1 \wedge r^0 \wedge \overline{s^2} \wedge \overline{s^1} \wedge s^0 + \quad r^2 \wedge r^1 \wedge r^0 \wedge \overline{s^2} \wedge \overline{s^1} \wedge \overline{s^0} + r^2 \wedge r^1 \wedge r^0 \wedge s^2 \wedge \overline{s^1} \wedge \overline{s^0} +$$
$$r^2 \wedge r^1 \wedge r^0 \wedge \overline{s^2} \wedge s^1 \wedge \overline{s^0} + r^2 \wedge r^1 \wedge r^0 \wedge \overline{s^2} \wedge s^1 \wedge s^0 + r^2 \wedge r^1 \wedge r^0 \wedge s^2 \wedge \overline{s^1} \wedge s^0 +$$
$$r^2 \wedge r^1 \wedge r^0 \wedge s^2 \wedge s^1 \wedge \overline{s^0}$$

2. ดำเนินการลดรูปการเข้าถึงข้อมูลโดยดึงตัวร่วมบิตแมปเวกเตอร์กลุ่ม R ที่มีรูปแบบการลงรหัสเหมือนกันได้ดังนี้

   $r^2 \wedge r^1 \wedge r^0 \wedge \quad (\overline{s^2} \wedge \overline{s^1} \wedge s^0 + \overline{s^2} \wedge \overline{s^1} \wedge \overline{s^0} + s^2 \wedge \overline{s^1} \wedge \overline{s^0} + \overline{s^2} \wedge s^1 \wedge \overline{s^0} +$
   $\overline{s^2} \wedge s^1 \wedge s^0 + s^2 \wedge \overline{s^1} \wedge s^0 + s^2 \wedge s^1 \wedge \overline{s^0})$

   ได้ฟังก์ชันการเข้าถึงข้อมูลดังนี้

   $r^2 \wedge r^1 \wedge r^0 \wedge (\overline{s^2} + \overline{s^1} + \overline{s^0})$

3. จากผลลัพธ์ในที่นี้ได้แสดงว่า ตรวจสอบบิตแมปเวกเตอร์ $r^2 \wedge r^1 \wedge r^0$ ที่มีค่าเท่ากับ 1 และดำเนินตรรกะ AND กับ $\overline{s^2} + \overline{s^1} + \overline{s^0}$ ที่ตรวจสอบบิตแมปเวกเตอร์ s^2 ที่มีบิตเท่ากับ 0 ตรวจสอบบิตแมปเวกเตอร์ s^1ที่มีบิตเท่ากับ 0 และ ตรวจสอบบิตแมปเวกเตอร์ s^0 ที่เท่ากับ 0 เพื่อหาคำตอบของการสอบถาม ผลลัพธ์การสอบถามที่ได้คือ เรคอร์ดที่ 3, 5, 8, 9, 11, 12 และ 14 ดังที่แสดงในภาพประกอบ 3-15

จากตัวอย่างการสอบถามแบบเป็นสมาชิกบน EDBI จะเห็นว่ามีการใช้การลกรูปก่อนกาเข้าถึงข้อมูลและ ดำเนินการตรรกะ AND และการดำเนอนการตรรกะ NOT มาช่วยในการสอบถาม การสอบถามแต่ละครั้งจะอ่านบิตแมปเวกเตอร์ไม่เกิน *nB* บิตแมปเวกเตอร์

r^2	r^1	r^0	s^2	s^1	s^0	$r^2 \wedge r^1 \wedge r^0$	$\overline{s^2}$	$\overline{s^1}$	$\overline{s^0}$	$(r^2 \wedge r^1 \wedge r^0) \wedge (\overline{s^2}+\overline{s^1}+\overline{s^0})$
1	0	1	0	0	1	0	1	1	0	0
1	1	0	0	1	1	0	1	0	0	0
1	1	1	1	0	0	1	0	1	1	1
1	1	0	0	0	1	0	1	1	0	0
1	1	1	1	1	0	1	0	0	1	1
1	1	0	0	1	0	0	1	0	1	0
1	1	0	0	0	0	0	1	1	1	0
1	1	1	0	0	0	1	1	1	1	1
1	1	1	0	0	1	1	1	1	0	1
1	1	0	0	1	1	0	1	0	0	0
1	1	1	0	1	0	1	1	0	1	1
1	1	1	0	1	1	1	1	0	0	1
1	0	1	0	1	0	0	1	0	1	0
1	1	1	1	0	1	1	0	1	0	1

ภาพประกอบ 3-15 ตัวอย่างการสอบถามข้มูลแบบเป็นสมาชิกแบบ EDBI บนแอททริบิวต์ type (C=16)

จากตัวอย่างการสอบถามแบบสมาชิกบน EDBI จะเห็นว่ามีการใช้การดำเนินการตรรกะ AND และ NOT มาช่วยในการสอบถาม การสอบถามแต่ละครั้งจะอ่านบิตแมปเวกเตอร์ไม่เกิน nB บิตแมปเวกเตอร์

3.3 ข้อเด่นของ EDBI

1. EDBI ใช้วิธีการเข้ารหัสในการลงรหัสให้กับดัชนีบิตแมป ทำให้ใช้พื้นที่ได้อย่างประสิทธิภาพดีกว่าดัชนีบิตแมปแบบคู่กัน
2. EDBI ใช้เทคนิคการจัดเรียงความถี่ให้กับค่าของแอททริบิวต์ตามลำดับการสอบถามจาก Query Workload ทำให้แอททริบิวต์ที่มีการสอบถามบ่อยสุดมีการลงรหัสเป็นบิต 1ทั้งหมดในบิตแมปเวกเตอร์กลุ่ม R และลงรหัสเป็นบิต 0 ทั้งหมดในบิตแมปเวกเตอร์กลุ่ม S ซึ่งเหมาะสมกับวิธีการสอบถามที่ออกแบบไว้ในการสอบถามแบบค่าเท่ากันเพราะค่าแอททริ-บิวต์ที่ถูกสอบถามบ่อยจะดำเนินการกับบิตแมปเวกเตอร์ $\frac{nB}{2}$ บิตแมปเวกเตอร์ (ตัวอย่างการสอบถามแบบค่าเท่ากันที่แสดงในภาพประกอบ 3-11) เพื่อลดจำนวนเรคอร์ดที่ต้องดำเนินการ XOR ซึ่งจำนวนเรคอร์ดที่ดำเนินการ XOR ของ EDBI จะน้อยกว่าดัชนีบิตแมป

แบบเข้ารหัสที่ดำเนินการตรรกะ AND และ NOT เท่ากับ N เรคอร์ด และกรณีที่ R ค่าใดๆ และ S = 0 จะดำเนินการกับบิตแมปเวกเตอร์ $\frac{nB}{2}$ บิตแมปเวกเตอร์เช่นกัน และ ในกรณีที่ R และ S มีบิตที่เป็น 1 มากจะสามารถลดจำนวนเรคอร์ดที่ต้องดำเนินการ XOR ได้มากด้วย

3. สำหรับการสอบถามแบบเป็นสมาชิกบน EDBI ถ้าสอบถามแอททริบิวต์ที่มีค่าความถี่ใกล้เคียงกันบิตแมปเวกเตอร์กลุ่ม R จะมีรูปแบบการลงรหัสที่เหมือนกัน และเมื่อทำการสอบถามไปด้วยกันจำนวนบิตแมปเวกเตอร์ที่อ่านคือ กลุ่ม R ที่มีจำนวนเท่ากับ $\frac{nB}{2}$ ส่วนจำนวนบิตแมปเวกเตอร์กลุ่ม S ที่ต้องอ่านขึ้นอยู่กับการลดรูปการเข้าถึง ทำให้จำนวนบิตแมปเวกเตอร์ที่อ่านมากที่สุด เท่ากับ nB ไม่ว่าจำนวนสมาชิกในการสอบถามจะมีกี่ค่า

3.4 ข้อด้อยของ EDBI

1. เนื่องจากต้องมีการใช้ Query Workload มาช่วยในการลงรหัส ถ้าหากมีเงื่อนไขที่ไม่เคยมีการสอบถามมาก่อนจะทำให้ การลงรหัสให้กับค่าแอททริบิวต์นั้นมีการลงรหัสเป็นบิต 0 จำนวนมากอาจทำให้จำนวนเรคอร์ดที่ต้องดำเนินการ XOR มีจำนวนมากและเวลาในการสอบถามก็เพิ่มขึ้น และทำให้ประสิทธิภาพในการสอบถามลดลง
2. การสอบถามแบบค่าเท่ากันบน EDBI จำนวนบิตแมปเวกเตอร์ที่อ่านแต่ละครั้งจะไม่เท่ากันขึ้นอยู่กับจำนวนบิต 1 ที่ใช้ลงรหัส จำนวนบิตแมปเวกเตอร์ที่ทำการสอบถามจึงไม่คงที่
3. EDBI ยังด้อยประสิทธิภาพในการสอบถามแบบค่าเท่ากันเมื่อเทียบกับดัชนีบิตแมปแบบคู่กันที่มีการอ่าน 2 บิตแมปเวกเตอร์ในทุกการสอบถาม
4. การสอบถามแบบเป็นสมาชิกถ้าสอบถามแอททริบิวต์ที่มีค่าความถี่ต่างกัน บิตแมปเวกเตอร์กลุ่ม R จะมีรูปแบบการลงรหัสที่ต่างกันในการสอบถามจะต้องทำการลดรูปฟังก์ชันการเข้าถึงข้อมูลทั้งบิตแมปเวกเตอร์กลุ่ม R และ S

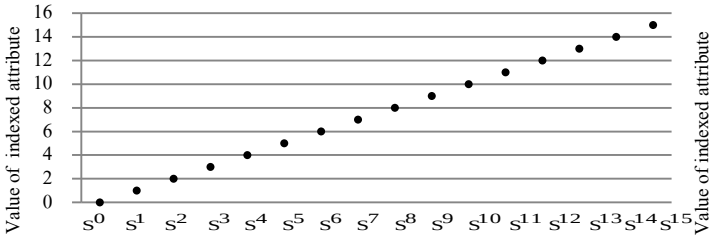
บทที่ 4

การวิเคราะห์และผลการทดลอง

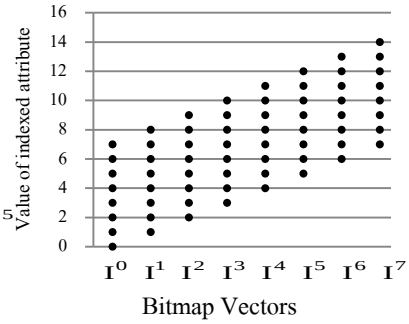
สำหรับในบทนี้จะกล่าวถึงการวิเคราะห์เปรียบเทียบค่าใช้จ่ายเกี่ยวกับพื้นที่ที่ใช้จัดเก็บดัชนี (Space Performance) ระหว่าง EDBI กับดัชนีบิตแมปแบบพื้นฐาน (SBI) ดัชนีบิตแมปแบบช่วง (IBI) ดัชนีบิตแมปแบบกระจาย (ScaBI) ดัชนีบิตแมปแบบคู่กัน (DBI) และดัชนีบิตแมปแบบเข้ารหัส (E-EBI) และวิเคราะห์ค่าใช้จ่ายเกี่ยวกับเวลาในการสอบถามข้อมูลแบบค่าเท่ากัน (Equality Query Time Performance) และการสอบถามแบบเป็นสมาชิก (Membership Query Time Performance) รวมถึงการเปรียบเทียบระหว่างการใช้พื้นที่กับเวลา (Space-Time Trade-Off) ระหว่าง EDBI กับ DBI และ E-EBI

4.1 ค่าใช้จ่ายเกี่ยวกับพื้นที่ที่ใช้ในการจัดเก็บดัชนี

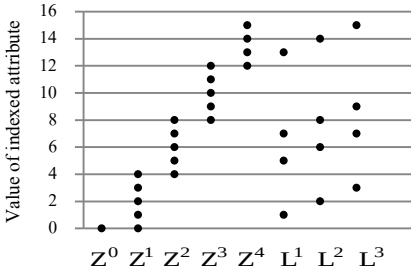
การใช้พื้นที่จัดเก็บดัชนีของดัชนีบิตแมปแต่ละแบบจะขึ้นอยู่กับลักษณะการลงรหัส เพราะเมื่อใช้วิธีการลงรหัสที่ต่างกันจะทำให้ประสิทธิภาพการใช้พื้นที่จัดเก็บดัชนีแตกต่างกันด้วย ตัวอย่างการลงรหัสของดัชนีบิตแมปทั้ง 5 แบบ และการลงรหัสของ EDBI สำหรับแอททริบิวต์ที่มีค่าคาร์ดินอลิตี้ C=16 จากรูปแบบการลงรหัสของ SBI ในภาพประกอบ 4-1(a) ใช้บิตแมปเวกเตอร์ทั้งหมด 16 บิตแมปเวกเตอร์ คือ S^0 - S^{15} และบิตแมปเวกเตอร์แต่ละตัวจะแทนค่าของแอททริบิวต์แค่เพียงค่าเดียว เช่น S^8 ใช้แทนค่าแอททริบิวต์ที่มีค่าเท่ากับ 8 รูปแบบการลงรหัสของ IBI ในภาพประกอบ 4-1(b) ใช้บิตแมปเวกเตอร์ทั้งหมด 8 บิตแมปเวกเตอร์ คือ I^0 – I^7 เช่น บิตแมปเวกเตอร์ I^0 ใช้แทนค่าของแอททริบิวต์ที่อยู่ในช่วง 0 ถึง 7 รูปแบบการลงรหัสของ ScaBI ในภาพประกอบ 4-1(c) ใช้บิตแมปเวกเตอร์ 8 บิตแมปเวกเตอร์ โดยแบ่งบิตแมปเวกเตอร์ออกเป็น 2 กลุ่มในการแทนค่าแอททริบิวต์ คือกลุ่ม Z^0 – Z^4 และกลุ่ม L^1 – L^3 บิตแมปเวกเตอร์ Z^0 จะเก็บค่าดัชนีบิตแมปแค่ตัวเดียวคือ 0 รูปแบบการลงรหัสของ E-EBI ใช้บิตแมปเวกเตอร์ทั้งหมด 4 บิตแมปเวกเตอร์ คือ E^0 – E^3 โดยใช้ 4 บิตแทนแต่ละค่าของแอททริบิวต์ และรูปแบบการลงรหัสของ DBI ใช้บิตแมปเวกเตอร์ทั้งหมด 7 บิตบิตแมปเวกเตอร์ คือ D^0 – D^6 แต่ละแอททริบิวต์แทนด้วย 2 บิตแมปเวกเตอร์ เช่น ค่าของแอททริบิวต์ที่เท่ากับ 0 จะถูกแทนด้วย D^1 และ D^6 รูปแบบการลงรหัสของ EBDI ใช้บิตแมปเวกเตอร์ 2 กลุ่มคือ r^2 ถึง r^0 และ s^2 ถึง s^0 โดยค่าของแอททริบิวต์แต่ละค่าจะถูกแทนด้วยบิตแมปเวกเตอร์ทั้งสองกลุ่มโดยใช้เลขฐานสอง



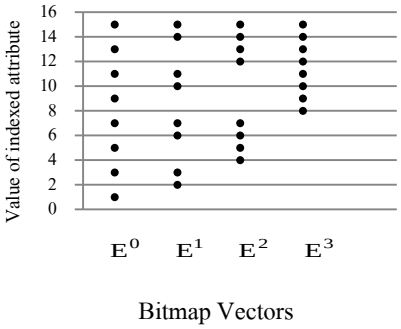
(a) SBI Encoding



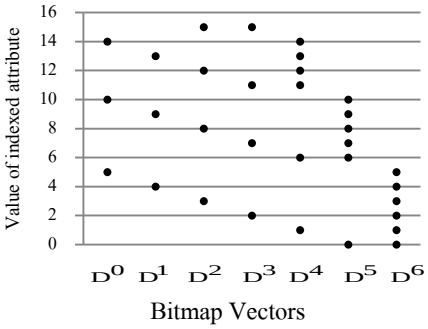
(b) IBI Encoding



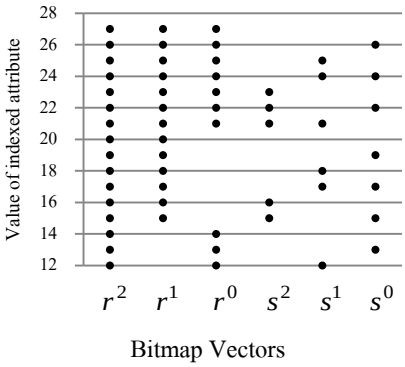
(c) ScaBI Encoding



(d) E-EBI Encoding



(e) DBI Encoding



(f) EDBI Encoding

ภาพประกอบ 4-1 รูปแบบการลงรหัสของดัชนีบิตแมปทั้ง 6 แบบ $C = 16$

จากภาพประกอบ 4-1 แสดงรูปแบบการลงรหัสของดัชนีบิตแมปทั้ง 6 แบบ แกน x แทนบิตแมปเวกเตอร์ และแกน y แทน ค่าของแอททริบิวต์ (Value of indexed attribute) ที่มีค่าตั้งแต่ 0-15 ยกเว้น EDBI ที่ให้แกน y แทนค่าของแอททริบิวต์ที่มีค่าตั้งแต่ 12-27 และสัญลักษณ์ • แทนการลงรหัส 1 นอกเหนือจากนั้นจะเป็นการลงรหัส 0

พิจารณาเปรียบเทียบการใช้พื้นที่จัดเก็บดัชนีบิตแมปทั้ง 6 แบบให้อยู่ในรูปทั่วไป ดังตาราง 4-1

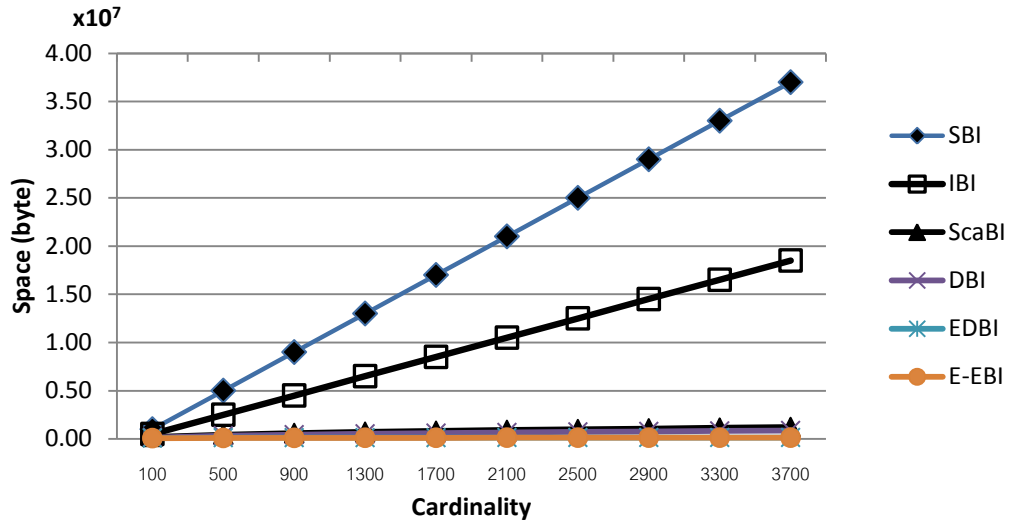
ตาราง 4-1 พื้นที่จัดเก็บของดัชนีบิตแมป 6 แบบ เมื่อ N แทน จำนวนเรคอร์ด และ C แทนค่าคาร์-ดินอลิตี้

ดัชนีบิตแมป	**ตัวย่อ**	**พื้นที่ที่ใช้จัดเก็บดัชนี** (บิต)
ดัชนีบิตแมปแบบพื้นฐาน	SBI	NC
ดัชนีบิตแมปแบบช่วง	IBI	$N\left\lceil C/2 \right\rceil$
ดัชนีบิตแมปแบบกระจาย	ScaBI	$N\left\lceil 2\sqrt{C} \right\rceil$
ดัชนีบิตแมปแบบคู่กัน	DBI	$N\left\lceil \sqrt{2C+0.25}+0.5 \right\rceil$
Enhancing Dual Bitmap Index	EDBI	$N\left(2\left\lceil\left(\log_2\left\lceil\sqrt{2C+0.25}+0.5\right\rceil\right)\right\rceil\right)$
ดัชนีบิตแมปแบบเข้ารหัส	E-EBI	$N\left\lceil \log_2 C \right\rceil$

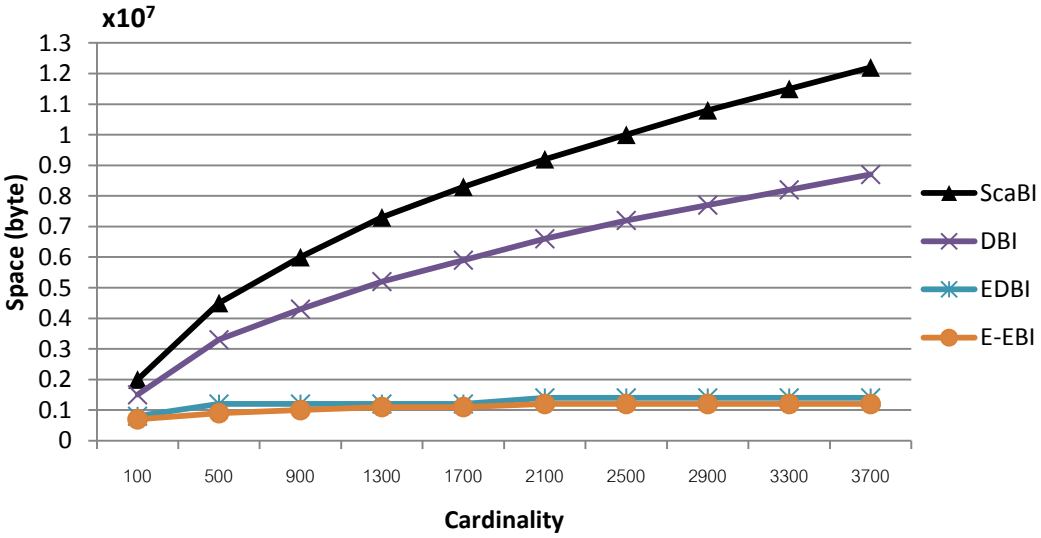
จากตาราง 4-1 จะเห็นว่าการทำดัชนีบิตแมปบนแอททริบิวต์ที่มีเรคอร์ดเท่ากับ N ขนาดของพื้นที่ที่ใช้จัดเก็บดัชนีบิตแมปจะแปรผันตามจำนวนคาร์ดินอลิตี้ ภาพประกอบ 4-2 แสดงกราฟเปรียบเทียบการใช้พื้นที่ของดัชนีบิตแมปทั้ง 6 แบบ โดยค่าคาร์ดินอลิตี้จะมีค่าอยู่ระหว่าง 100 ถึง 3,700 จะเห็นว่าเราสามารถจัดกลุ่มดัชนีบิตแมปได้เป็น 2 กลุ่ม คือ

- กลุ่มที่ใช้พื้นที่มาก ได้แก่ SBI และ IBI โดย SBI จะใช้พื้นที่จัดเก็บดัชนีมากกว่า IBI
- กลุ่มที่ใช้พื้นที่น้อย ได้แก่ ScaBI, DBI, EDBI และ E-EBI พิจารณาจากภาพประกอบที่ 4-3 จะเห็นว่า ScaBI ใช้พื้นที่จัดเก็บดัชนีมากที่สุดและ E-EBI ใช้พื้นที่จัดเก็บดัชนีน้อยที่สุด ส่วน EDBI ใช้พื้นที่จัดเก็บดัชนีอยู่ระหว่าง DBI และ E-EBI

โดยสรุปจะเห็นว่า E-EBI มีประสิทธิภาพในเรื่องของการใช้พื้นที่จัดเก็บดัชนีมากที่สุดและดัชนีบิตแมปที่มีประสิทธิภาพการใช้พื้นที่จัดเก็บดัชนีได้ใกล้เคียงกับ E-EBI มากที่สุดคือ EDBI



ภาพประกอบ 4-2 เปรียบเทียบดัชนีบิตแมป 6 แบบ



ภาพประกอบ 4-3 เปรียบเทียบดัชนีบิตแมป 4 แบบ

4.2 ค่าใช้จ่ายเกี่ยวกับเวลาในการสอบถาม

โดยทำการทดลองบนเครื่องคอมพิวเตอร์รุ่น Intel(R) Core(TM)2 Duo ที่มีหน่วยประมวลผลกลาง 2.40 GHz หน่วยความจำหลักเท่ากับ 4.00 GB ระบบปฏิบัติการ Windows 7 Professional และข้อมูลที่ใช้ในการทดลองเป็นชุดข้อมูลมาตรฐานจาก TCP-H Benchmark (Transaction Processing Performance Council, 2013) สำหรับการสอบถามข้อมูลแบบค่าเท่ากัน เลือก แอททริบิวต์ที่นำมาทำดัชนี มีคาร์ดินอลิตี้เท่ากับ 25, 50 และ 150 มีจำนวนเรคอร์ดเท่ากับ 7,000,000 เรคอร์ด สำหรับการสอบถามข้อมูลแบบเป็นสมาชิก แอททริบิวต์ที่นำมาทำดัชนี มีคาร์ดินอลิตี้เท่ากับ 150 และมีจำนวนเรคอร์ดเท่ากับ 7,000,000 เรคอร์ด

4.2.1 ค่าใช้จ่ายเกี่ยวกับเวลาในการสอบถามข้อมูลแบบค่าเท่ากัน

จากรูปแบบการลงรหัสที่ต่างกัน มีผลต่อการสอบถามข้อมูลแบบค่าเท่ากันของดัชนีบิตแมปแต่ละแบบแตกต่างกัน ทั้งจำนวนบิตแมปเวกเตอร์ที่ต้องอ่าน จำนวนครั้งในการดำเนินตรรกะระหว่างบิตแมปเวกเตอร์ ซึ่งเวลาในการสอบถามข้อมูลแบบค่าเท่ากันจะขึ้นอยู่กับปัจจัยต่างๆ เช่น จำนวนบิตแมปเวกเตอร์ที่อ่าน จำนวนเรคอร์ด ถ้าหากมีการดำเนินการระหว่างบิตหลายครั้งก็จะทำให้ใช้เวลาในการสอบถามนาน ในการเปรียบเทียบค่าใช้จ่ายของการสอบถามข้อมูลแบบค่าเท่ากันจะเป็นการเปรียบเทียบกันระหว่าง EDBI กับ DBI และ E-EBI โดยจำนวนบิตแมปเวกเตอร์ที่ต้องตรวจสอบของดัชนีบิตแมปแสดงในตาราง 4-2 ให้ C แทนค่าคาร์ดินอลิตี้

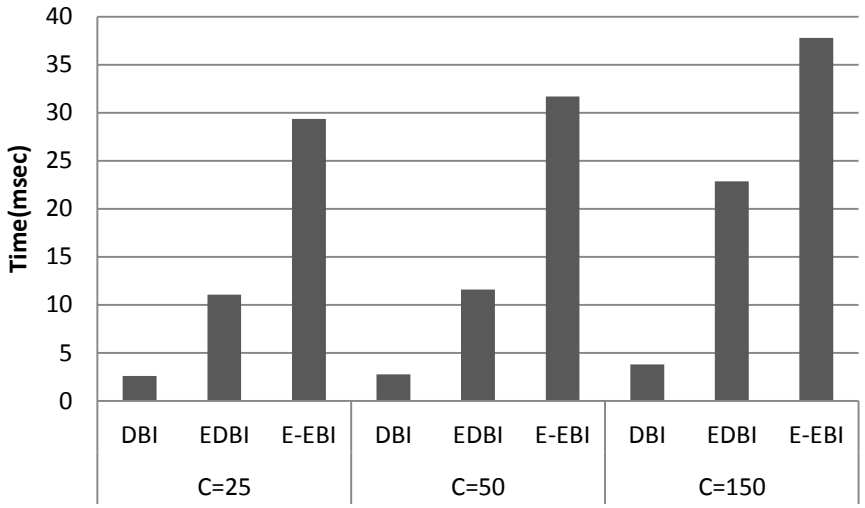
ตาราง 4-2 เปรียบเทียบจำนวนบิตแมปเวกเตอร์ที่ต้องอ่าน ของ DBI, EDBI และ E-EBI

ดัชนีบิตแมป	จำนวนบิตแมปเวกเตอร์ที่อ่าน		จำนวนเรคอร์ด (N)ที่เปรียบเทียบ
	น้อยสุด (บิต)	มากสุด (บิต)	
DBI	2	2	0
EDBI	2	(nB-1)	$< N$
E-EBI	log_2C	log_2C	N

จากตาราง 4-2 จะเห็นว่าการสอบถามข้อมูลแบบค่าเท่ากัน DBI ต้องอ่านบิตแมปเวกเตอร์เพียง 2 บิตแมปเวกเตอร์ทุกครั้งในการสอบถาม และไม่มีการดำเนินการระดับเรคอร์ด (N แทนจำนวนเรคอร์ดทั้งหมด) EDBI อ่านบิตแมปเวกเตอร์น้อยที่สุดคือ 2 บิตแมปเวกเตอร์ และ อ่านบิตแมปเวกเตอร์มากที่สุดไม่เกิน nB-1 (nB คือจำนวนบิตที่ใช้แทนแต่ละค่าของแอททริบิวต์ซึ่งเท่ากับจำนวนบิตแมปเวกเตอร์) การดำเนินการเปรียบเทียบระดับเรคอร์ดใช้วิธีการดำเนินการ

ตรรกะ XOR ซึ่งการดำเนินการนั้นจะน้อยกว่าจำนวนเรคอร์ดเสมอ และ E-EBI อ่านบิตแมปเวกเตอร์เท่ากับ log_2C ทุกครั้งในการสอบถาม มีการดำเนินการตรรกะ AND และ NOT กับทุกเรคอร์ด โดยสรุปคือ DBI ใช้เวลาในการสอบถามข้อมูลแบบค่าเท่ากันน้อยที่สุด และ EDBI มีจำนวนเรคอร์ดที่ดำเนินการตรรกะ น้อยกว่า E-EBI

ในการทดลองเวลาการสอบถามข้อมูลของ EDBI จะมาจากเวลาในการอ่านบิตแมปเวกเตอร์รวมกับเวลาในการดำเนินการตรรกะ XOR และ E-EBI จะมาจากเวลาในการอ่านบิตแมปเวกเตอร์รวมกับเวลาในการดำเนินการตรรกะ AND และ NOT



ภาพประกอบ 4-4 กราฟแนวโน้มของเวลาในการสอบถามแบบค่าเท่ากันบน DBI, EDBI และ E-EBI

จากกราฟ แนวโน้มของเวลาในการสอบถามแบบค่าเท่ากันบน DBI, EDBI และ E-EBI ที่มีคาร์ดินอลิตี้ C = 25 คาร์ดินอลิตี้ C = 50 และคาร์ดินอลิตี้ C = 150 ตามลำดับ จะเห็นว่าเมื่อ C เพิ่มขึ้นจะทำให้เวลาที่ใช้ในการสอบถามเพิ่มขึ้นด้วยทั้งนี้ เพราะจำนวนบิตแมปเวกเตอร์ที่ใช้จัดเก็บดัชนีมีขึ้นอยู่กับ C ถ้า C มาก จำนวนบิตแมปเวกเตอร์มาก เวลาที่ใช้สอบถามจะเพิ่มขึ้นด้วย จากกราฟจะเห็นว่า DBI ใช้เวลาสอบถามโดยเฉลี่ยน้อยที่สุด และรองลงมาคือ EDBI และ E-EBI ตามลำดับ

การสร้างกรณีสอบถามบน EDBI กับ DBI และ E-EBI ที่มีค่าคาร์ดินอลิตี้เท่ากับ 50 โดยยกตัวอย่างการสอบถาม 5 แบบ แทนด้วย Q1, Q2, Q3, Q4 และ Q5 ดัชนีบิตแมปทั้งสามชนิดมีจำนวนเรคอร์ดที่เท่ากันคือ 7,000,000 เรคอร์ด สำหรับการบันทึกผลการทดลองดำเนินการสอบถาม Q1, Q2, Q3, Q4 และ Q5 ทั้ง 5 แบบซึ่งดำเนินการซ้ำทั้งหมด 3 รอบ การสอบถามแต่ละ

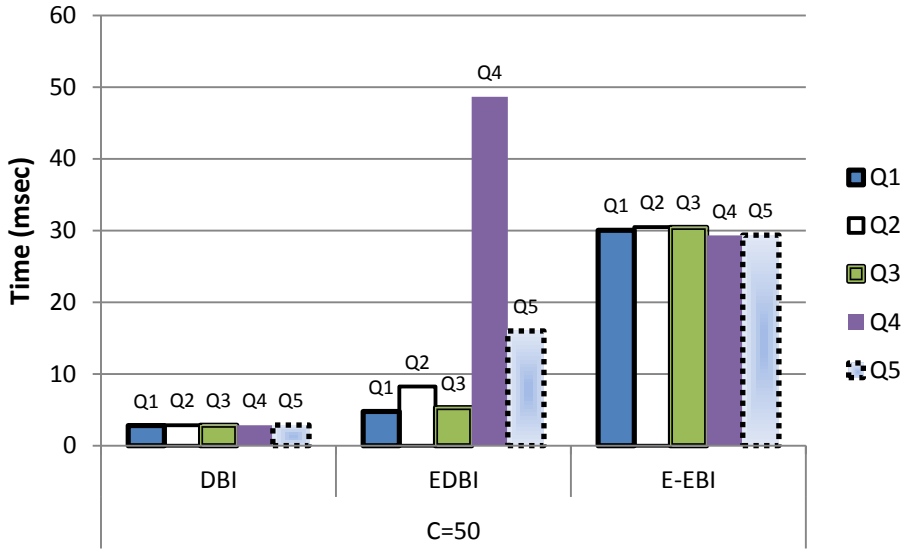
แบบจะมีความแตกต่างกันแสดงรายละเอียดลักษณะการสอบถามดังตาราง 4-3 และหาค่าเฉลี่ยของเวลาที่ใช้ในการสอบถามแบบค่าเท่ากันบนดัชนีบิตแมปทั้ง 3 แบบโดยเวลาการสอบถามแสดงในตาราง 4-4 และนำค่าเฉลี่ยของเวลาที่ได้จากตาราง 4-4 มาเขียนกราฟได้ดังภาพประกอบ 4-5

ตาราง 4-3 ลักษณะการสอบถามทั้ง 5 แบบ

การสอบถาม	**ลักษณะของการสอบถาม**
Q1	Q1: SELECT * FROM T WHERE type = 0
Q2	Q2: SELECT * FROM T WHERE type = 29
Q3	Q3: SELECT * FROM T WHERE type = 14
Q4	Q4: SELECT * FROM T WHERE type = 37
Q5	Q5: SELECT * FROM T WHERE type = 41

ตาราง 4-4 เวลาที่ใช้ในการสอบถามข้อมูลแบบค่าเท่ากันบน EDBI กับ DBI และ EBI (C =50)

คาร์ดินอลลอตี้	**ดัชนีบิตแมป**	**No.**	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**
C=50	DBI	1	2.799239	2.843035	2.811939	2.848279	2.89671
		2	2.801459	2.85953	2.845823	2.852057	2.854795
		3	2.796554	2.867901	2.880634	2.857807	2.852019
		Avg.	**2.7991**	**2.8568**	**2.8461**	**2.8527**	**2.8678**
	EDBI	1	4.773589	7.668474	7.595212	45.30678	14.93086
		2	4.762917	9.163034	4.476181	46.50283	15.27325
		3	4.762917	7.832391	3.82128	54.17642	17.7597
		Avg.	**4.766**	**8.221**	**5.297**	**48.662**	**15.988**
	E-EBI	1	30.58295298	30.85306	31.32044	28.0676	29.67328
		2	30.2942431	29.47895	31.31926	28.02133	29.75373
		3	30.18261099	30.09724	30.68746	30.78394	26.61768
		Avg.	**30.353**	**30.143**	**31.109**	**28.958**	**28.682**



ภาพประกอบ 4-5 กราฟเปรียบเทียบการสอบถามทั้ง 5 แบบบน DBI, EDBI และ E-EBI

ภาพประกอบ 4-5 แสดงกราฟเปรียบเทียบการสอบถามทั้ง 5 แบบบน DBI, EDBI และ E-EBI การสอบถาม 5 แบบแทนด้วย Q1, Q2, Q3, Q4 และ Q5 ในกราฟนี้เป็นการเปรียบเทียบเวลาการสอบถามระหว่าง EDBI กับ DBI และ E-EBI โดยดัชนีบิตแมปทั้งสามชนิดมีจำนวนเรคอร์ดที่เท่ากันคือ 7,000,000 เรคอร์ด ดัชนีบิตแมปแบบแรกที่จะกล่าวถึงคือ DBI วิธีการสอบถามของ DBI จะอ่านบิตแมปเวกเตอร์ 2 บิตแมปเวกเตอร์ทุกครั้ง มีการดำเนินการตรรกะ AND เวลาที่ใช้ในการสอบถามจึงเท่ากันทั้ง 5 แบบDBI การสอบถามบน DBI จึงมีความรวดเร็ว

วิธีการสอบถามของ E-EBI มีการอ่านบิตแมปเวกเตอร์เท่ากับ log_2C และการดำเนินการตรรกะ จะเท่ากับจำนวนเรคอร์ด (N) ดังนั้นเวลาในการสอบถามของจะใกล้เคียงกันทั้ง 5 แบบ ของการสอบถาม

วิธีการสอบถามของ EDBI จะแตกต่างจากดัชนีบิตแมปที่กล่าวมาแล้วทั้ง 2 แบบ เพราะเวลาที่ใช้ในการสอบถามของ EDBI จะขึ้นอยู่กับชนิดของการสอบถามสำหรับการลงรหัสของ EDBI จะเรียงความถี่ค่าของแอทริบิวต์จากมากไปน้อยก่อนลงรหัสโดยความถี่สูงสุดคือค่าแอททริบิวต์ที่เท่ากับ 0 และการดำเนินการตรรกะ XOR น้อยกว่าจำนวนเรคอร์ด จากกราฟจะเห็นว่าแนวโน้มของเวลาที่ใช้ในการสอบถามของ Q1, Q2, Q3 และ Q5 จะใกล้เคียงกัน แต่ Q4 จะใช้เวลาการสอบถามมากที่สุด จึงทำการวิเคราะห์รูปแบบของการสอบถามบน EDBIได้ดังต่อไปนี้

EDBI บนแอททริบิวต์ที่มีจำนวนของคาร์ดินอลิตี้ $C = 50$ จำนวนบิตแมปเวกเตอร์ (nB) ที่ใช้จึงเท่ากับ 8 บิตแมปเวกเตอร์ (สมการที่ (1) ในบทที่ 3) ค่าตัวเลขที่ใช้แทนค่าที่ถูกสอบถาม

บ่อยสุด (V_0) เท่ากับ 119 (สมการที่ (2) ในบทที่ 3) จากตาราง 4-3 ลักษณะของการสอบถามทั้ง 5 แบบเมื่อทำการ lookup ค่า R_i และ S_i จาก EDBI Mapping Table ได้รายละเอียดของ Query ดังแสดงในตาราง 4-5

ตาราง 4-5 รายละเอียดของ Query

Query	R	S	เลขฐานสอง	จำนวนบิตแมปเวกเตอร์ที่อ่าน
Q1: 0	15	0	1111 0000	4
Q2: 29	13	0	1101 0000	4
Q3: 14	15	14	1111 1110	7
Q4: 37	13	8	1101 1000	4
Q5: 41	13	12	1101 1100	5

จะเห็นว่า Q1 และ Q2 ถูกลงรหัสได้ให้บิตแมปเวกเตอร์ S เป็นบิต 0 จึงมีการดำเนินการอ่านบิตแมปเวกเตอร์เท่ากับ $\frac{nB}{2}$ เวลาในการสอบถามจึงน้อย Q3 และ Q5 มีการลงรหัสเป็นบิต 1 จำนวนมากจึงสามารถกรองจำนวนเรคอร์ดที่ต้องดำเนินการตรรกะ XOR ให้มีจำนวนเรคอร์ดน้อยเวลาในการสอบถามจึงน้อย ส่วน Q4 ใช้เวลาการสอบถามนานที่สุดเพราะค่าที่สอบถามมีการลงรหัสเป็นบิต 1 น้อยกว่า Q3 และ Q5 ดังนั้นจำนวนเรคอร์ด ที่ต้องดำเนินการตรรกะ XOR จึงใช้ในเวลาการสอบถามจึงมากขึ้นตามไปด้วย

โดยสรุปเรื่องเวลาในการสอบถามปรากฏว่า DBI ใช้เวลาการสอบถามน้อยที่สุดและโดยภาพราม EDBI ใช้เวลาการสอบถามได้มีประสิทธิภาพมากกว่า E-EBI

4.2.2 ค่าใช้จ่ายเกี่ยวกับเวลาในการสอบถามข้อมูลแบบเป็นสมาชิก

การสอบถามข้อมูลแบบสมาชิก เวลาที่ใช้ในการสอบถามข้อมูลแบบเป็นสมาชิกจะประกอบด้วยเวลาที่อ่านบิตแมปเวกเตอร์กับเวลาในการดำเนินตรรกะระหว่างบิตแมปเวกเตอร์ ถ้าต้องอ่านบิตแมปเวกเตอร์มาก และต้องมีการดำเนินการระหว่าตรรกะหลายครั้งเวลาที่ใช้ในการสอบถามแบบสมาชิกก็จะนานขึ้น ในการเปรียบเทียบค่าใช้จ่ายของการสอบถามข้อมูลแบบค่าเท่ากันจะเป็นการเปรียบเทียบกันระหว่าง EDBI กับ DBI และ E-EBI โดยจำนวนบิตแมปเวกเตอร์ที่ต้องตรวจสอบของดัชนีบิตแมปแสดงในตาราง 4-6

ตาราง 4-6 เปรียบเทียบจำนวนบิตแมปเวกเตอร์ที่ต้องอ่าน ของ DBI, EDBI และ E-EBI

ดัชนีบิตแมป	**จำนวนบิตแมปเวกเตอร์ที่อ่าน**	Boolean Operation
DBI	2 to 2M	1 AND to 2M (M AND, M-1 OR)
EDBI	1, $nB/2$ to nBM	0 to M-1 AND M-1 OR
E-EBI	1 to $\lceil \log_2 C \rceil M$	0 to M-1 OR

จากตาราง 4-6 แสดงให้เห็นว่าการสอบถามแบบเป็นสมาชิกที่มีจำนวนสมาชิก M ค่า บนดัชนีบิตแมปทั้ง 3 แบบ

DBI มีการอ่านบิตแมปเวกเตอร์แค่ 2 บิตแมปเวกเตอร์เท่านั้นและมีการดำเนินการตรรกะ AND หนึ่งครั้งสำหรับกรณีที่ดีที่สุด และในกรณีที่แย่ที่สุด ต้องอ่านบิตแมปเวกเตอร์ 2M บิตแมปเวกเตอร์ และมีการดำเนินการตรรกะ AND 1 ครั้งไปจนถึง 2M (M AND, M-1 OR) ครั้ง

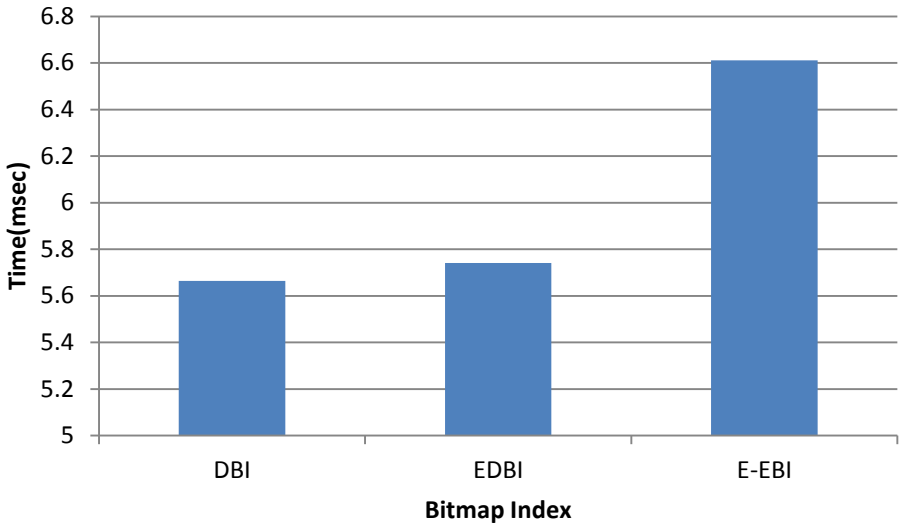
E-EBI มีการอ่านบิตแมปเวกเตอร์ 1 บิตแมปเวกเตอร์ และไม่มีการดำเนินการตรรกะสำหรับกรณีที่ดีที่สุด และใช้การลดรูปก่อนการเข้าถึง ส่วนกรณีที่แย่ที่สุดต้องอ่านบิตแมปเวกเตอร์ $\lceil \log_2 C \rceil M$ มีการดำเนินการตรรกะ OR M-1 ครั้ง

EDBI มีการอ่านบิตแมปเวกเตอร์ 1 บิตแมปเวกเตอร์ขึ้นอยู่กับการลดรูปฟังก์ชันการเข้าถึงเช่นเดียวกับ E-EBI จึงทำให้ไม่มีการดำเนินการตรรกะสำหรับกรณีที่ดีที่สุด และถ้าสอบถามแอททริบิวต์ที่มีค่าความถี่ใกล้เคียงกันบิตแมปเวกเตอร์กลุ่ม R จะมีรูปแบบการลงรหัสที่เหมือนกัน และเมื่อทำการสอบถามไปด้วยกันจำนวนบิตแมปเวกเตอร์ที่อ่านคือ กลุ่ม R ที่มีจำนวนเท่ากับ $\frac{nB}{2}$ (nB แทนจำนวนบิตแมปเวกเตอร์) สามารถลดจำนวนบิตแมปเวกเตอร์ที่ต้องอ่านได้ดีกว่า E-EBI ในกรณีที่แย่ที่สุด ต้องอ่านบิตแมปเวกเตอร์ nBM มีการดำเนินการตรรกะ OR M-1 ครั้ง และ AND M-1 ครั้ง ซึ่งขึ้นอยู่กับ Query Workload ในอดีตที่นำมาสร้างรูปแบบการลงรหัส แต่กรณีที่แย่ที่สุดจะเกิดขึ้นยากมาก

ในการทดลองกำหนดให้การสอบถามข้อมูลแบบเป็นสมาชิกบนดัชนีบิตแมปทั้ง 3 แบบ เวลาในการอ่านบิตแมปเวกเตอร์ และเวลาในการดำเนินการตรรกะระหว่างบิตแมปเวกเตอร์สำหรับการทดลองจะใช้ชุดการสอบถาม 2 ชุดการสอบถาม ซึ่งจะดำเนินการซ้ำ 3 รอบและหาค่าเฉลี่ย

ตาราง 4-7 เวลาที่ใช้ในการสอบถามข้อมูลแบบเป็นสมาชิกบน DBI, EDBI และ E-EBI บนชุดการสอบถามที่ 1

ดัชนีบิตแมป	DBI	EDBI	E-EBI
1	5.655803	5.745334	6.649519
2	5.671263	5.738427	6.622877
3	5.663412	5.740046	6.561575
Avg.	**5.663493**	**5.741269**	**6.611324**

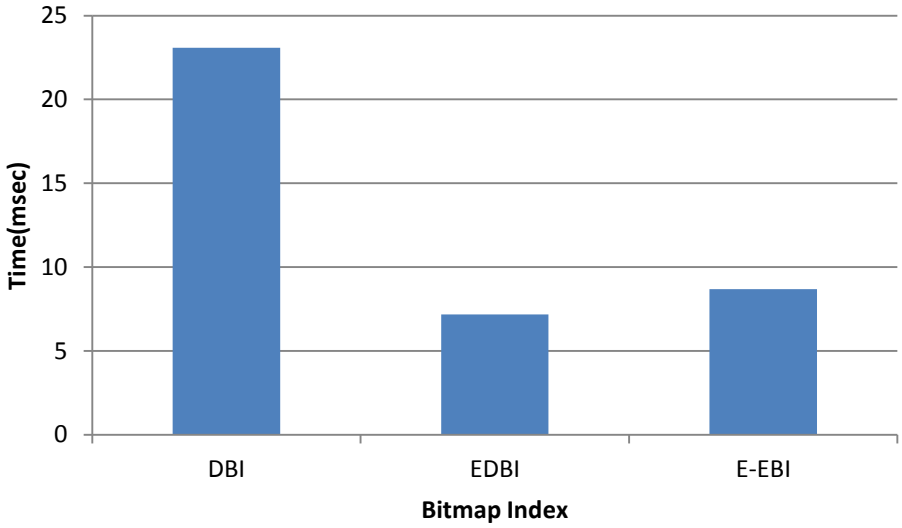


ภาพประกอบ 4-6 กราฟเปรียบเทียบเวลาที่ใช้ในการสอบถามแบบเป็นสมาชิกบน DBI, EDBI และ E-EBI บนชุดการสอบถามที่ 1

ผลการสอบถามของชุดการสอบภามที่ 1 แสดงในตาราง 4-7 และ ในภาพประกอบ 4-6 แสดงกราฟเปรียบเทียบเวลาที่ใช้ในการสอบถามแบบเป็นสมาชิกบน DBI, EDBI และ E-EBI จะเห็นว่า DBI ใช้เวลาในการสอบถามน้อยที่สุด เนื่องจากข้อมูลในการสอบถามมีจำนวนสมาชิกที่ใช้สอบถามน้อย จึงทำให้ DBI ใช้เวลาการสอบถามได้เร็วที่สุด รองลงมาคือ EDBI และ E-EBI ตามลำดับ ซึ่งดัชนีบิตแมปทั้งสองใช้วิธีการลดรูปฟังก์ชันก่อนเข้าถึงข้อมูล แต่ EDBI ลดรูปฟังก์ชันแล้วมีจำนวนบิตแมปเวกเตอร์ที่ต้องอ่านน้อยกว่า E-EBI จึงทำการสอบถามแบบเป็นสมาชิกบนชุดการสอบถามที่ 1 EDBI ใช้เวลาน้อยกว่า E-EBI

ตาราง 4-8 เวลาที่ใช้ในการสอบถามข้อมูลแบบเป็นสมาชิกบน DBI, EDBI และ E-EBI บนชุดการสอบถามที่ 2

ดัชนีบิตแมป	DBI	EDBI	E-EBI
1	23.05781	7.184288	8.681438
2	23.09564	7.177065	8.674488
3	23.07078	7.199299	8.692879
Avg.	**23.07474**	**7.186884**	**8.682935**



ภาพประกอบ 4-7 กราฟเปรียบเทียบเวลาที่ใช้ในการสอบถามแบบเป็นสมาชิกบน DBI, EDBI และ E-EBI บนชุดการสอบถามที่ 2

จากตาราง 4-8 และภาพประกอบ 4-7 แสดงกราฟเปรียบเทียบเวลาที่ใช้ในการสอบถามแบบเป็นสมาชิกบน DBI, EDBI และ E-EBI จะเห็นว่า EDBI ใช้เวลาในการสอบถามน้อยที่สุด เนื่องจากลักษณะข้อมูลในการสอบถามมีจำนวนสมาชิกที่ใช้สอบถามมาก จึงทำให้มีการดำเนินการตรรกะระหว่างบิตแมปเวกเตอร์มาก ส่งผลให้ DBI ใช้เวลาในการสอบถามแบบเป็นสมาชิกมากขึ้นตามจำนวนสมาชิกที่สอบถาม ส่วน EDBI และ E-EBI ที่ใช้วิธีการลดรูปฟังก์ชันการเข้าถึงข้อมูลก่อนการสอบถาม จึงทำให้บิตแมปเวกเตอร์ที่ต้องอ่านมีจำนวนน้อย ทำให้ใช้เวลาในการสอบถามข้อมูลใกล้เคียงกัน โดยสรุปคือ EDBI มีโอกาสใช้เวลาในการสอบถามน้อยกว่า DBI โดยขึ้นอยู่กับ Query Workload ที่ใช้ในการสร้าง EDBI

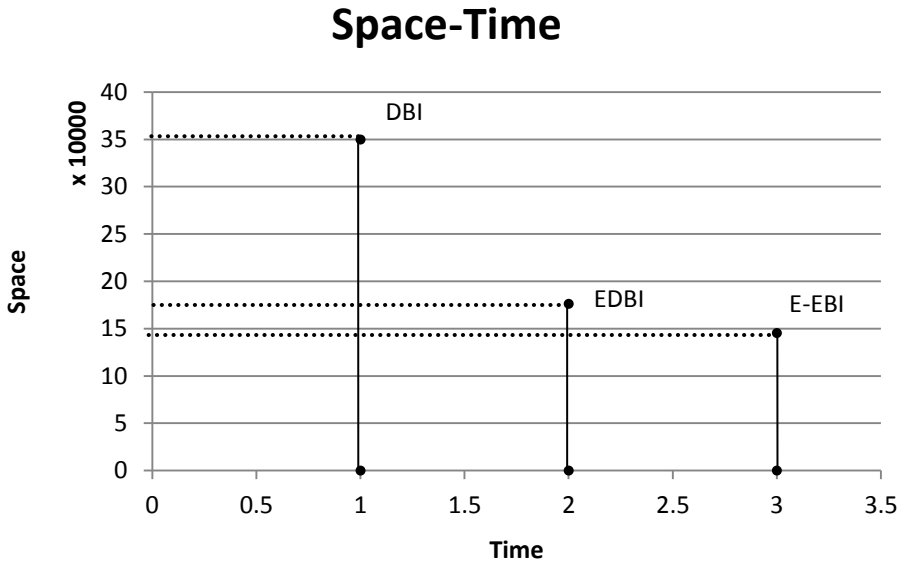
ลักษณะของชุดคำสั่งสอบถามมีผลต่อประสิทธิภาพของการสอบถามเพราะวิธีการลงรหัสของ EDBI มีการจัดลำดับความสำคัญให้ค่าของแอททริบิวต์ ซึ่งใช้การสอบถามที่เคยสอบถามไปแล้วในอดีตมาช่วยในการเรียงลำดับค่าของแอททริบิวต์ รูปแบบการลงรหัสจึงเรียงตามค่าของแอททริบิวต์ด้วย ดังนั้นเมื่อมีการสอบถามค่าแอททริบิวต์ที่มีลำดับความสำคัญหรือในงานวิจัยนี้ใช้ค่าความถี่ที่สูง ประสิทธิภาพในการสอบถามก็จะดี แต่ถ้าชุดคำสั่งที่สอบถามที่มีการสอบถามค่าแอททริบิวต์ที่ลำดับความสำคัญน้อยหรือความถี่ต่ำ ประสิทธิภาพการสอบถามก็จะลดลงทั้งการสอบถามแบบค่าเท่ากันละการสอบถามแบบเป็นสมาชิก

4.3 การแลกเปลี่ยนระหว่างการใช้พื้นที่กับเวลา (Space-Time Trade-Off)

สำหรับการแลกเปลี่ยนพื้นที่กับเวลาในการสอบถามจะเป็นการเปรียบเทียบกันระหว่าง EDBI กับ DBI และ E-EBI

4.3.1 การแลกเปลี่ยนพื้นที่กับเวลาสำหรับการสอบถามข้อมูลแบบค่าเท่ากัน

จากข้อ 4.1 จะเห็นว่าดัชนีบิตแมปที่ใช้พื้นที่จัดเก็บดัชนีได้มีประสิทธิภาพมากที่สุดคือ E-EBI และในหัวข้อ 4.2.1 ดัชนีบิตแมปที่มีประสิทธิภาพในเรื่องของเวลาที่ใช้ในการสอบถามแบบค่าเท่ากันที่ดีที่สุดคือ DBI ดังนั้นจะเห็นว่า EBDI จะอยู่กลางระหว่าดัชนีบิตแมปสองแบบที่กล่าวมาข้างต้น ดังนั้นเมื่อทำการวิเคราะห์ในแง่ของ Space-Time Trade-Off ของการสอบถามข้อมูลแบบแบบค่าเท่ากันกับพื้นที่ที่ใช้จัดเก็บดัชนี แสดงในภาพประกอบ 4-8 คือ กราฟแสดงความสัมพันธ์ของพื้นที่กับเวลาในการสอบถามแบบค่าเท่ากัน โดยใช้พื้นและเวลาบนการสอบถาม Q1 บนแอททริบิวต์ที่มีค่าคาร์ดินอลิตี้เท่ากับ $C = 150$ จะแสดงในภาพประกอบ 4-8



ภาพประกอบ 4-8 กราฟแสดงความสัมพันธ์ของพื้นที่กับเวลาของ DBI, EDBI และ E-EBI

จากกราฟแสดงความสัมพันธ์ของพื้นที่กับเวลาของดัชนีบิตแมปทั้ง 3 แบบ ที่แสดงในภาพประกอบ 4-8 แกน x แสดงเวลาที่ในการสอบถาม แกน y แสดงพื้นที่ที่ใช้ในการจัดเก็บดัชนี จะเห็นว่า DBI ใช้เวลาในการสอบถามน้อยที่สุด แต่ใช้พื้นที่ในการจัดเก็บดัชนีมากที่สุด E-EBI ใช้เวลาในการสอบถามมากที่สุด แต่ใช้พื้นที่ในกาจัดเก็บดัชนีน้อยที่สุดและ EDBI ใช้เวลาในการสอบถามน้อยกว่า E-EBI แต่ใช้พื้นที่จัดเก็บดัชนีน้อยกว่า DBI เมื่อคาร์ดินอลิตี้เท่ากับ $C = 150$ หมายความว่า EDBI มี Space-Time Trade-off ดีที่สุดเมื่อแอททริบิวต์ที่นำมาทำดัชนีบิตแมปมีค่าคาร์ดินอลิตี้สูงและมีการลงรหัสดัชนีได้เหมาะสมกับการสอบถาม

บทที่ 5

บทสรุปและข้อเสนอแนะ

ในวิทยานิพนธ์นี้จะกล่าวถึงการเพิ่มประสิทธิภาพให้กับดัชนีบิตแมปแบบคู่กัน โดยใช้วิธีการเข้ารหัส เพื่อลดพื้นที่ที่ใช้จัดเก็บดัชนี สำหรับการสอบถามข้อมูลแบบค่าเท่ากัน และการสอบถามข้อมูลแบบเป็นสมาชิก มีการใช้เทคนิคการจัดเรียงความถี่มาช่วยในการลงรหัสให้เหมาะสมเพื่อนำมาสร้างเป็นดัชนีบิตแมป สำหรับขั้นตอนการเพิ่มประสิทธิภาพดัชนีบิตแมปแบบคู่กันโดยใช้วิธีการเข้ารหัส (EDBI) จะแบ่งเป็น 3 ขั้นตอนคือ 1) Attribute Extraction เป็นการสกัดค่าแอททริบิวต์ออกมาจาก Query workload 2) Sorted Frequent Table Creation เป็นขั้นตอนการจัดเรียงความถี่ของแอททริบิวต์ที่สกัดมาได้ และ 3) Value Assignment and Encoder เป็นขั้นตอนการให้ค่าและการลงรหัสให้แอททริบิวต์ ผลการดำเนินการที่ได้คือ ประสิทธิภาพในเรื่องของพื้นที่ที่ใช้จัดเก็บดัชนี กับเวลาที่ใช้สอบถามข้อมูลแบบค่าเท่ากันและการสอบถามข้อมูลแบบเป็นสมาชิกสำหรับแอททริบิวต์ที่มีคาร์ดินอลิตี้สูง

5.1 บทสรุป

คลังข้อมูลเป็นที่ใช้เก็บรวบรวมข้อมูลให้อยู่ในที่เดียวกันโดยข้อมูลที่เก็บรวบรวมนั้นจะเป็นข้อมูลตั้งแต่อดีตจนถึงปัจจุบัน เมื่อข้อมูลมีจำนวนมากการสอบถามจากคลังข้อมูลจึงใช้เวลานาน และการสอบถามส่วนใหญ่ซับซ้อนทำให้การประมวลผลข้อมูลต้องใช้เวลามากขึ้น วิธีการทำดัชนี เป็นวิธีการเพิ่มความเร็วในการเข้าถึงข้อมูลโดยไม่ต้องเสียค่าใช้จ่ายใด ๆ แต่เป็นการใช้พื้นที่แลกกับเวลาในการสอบถามที่เร็วขึ้น ซึ่งเป็นที่นิยมในการเพิ่มประสิทธิภาพการสอบถามข้อมูลในคลังข้อมูล และการทำดัชนีที่เลือกคือ การทำดัชนีบิตแมป เพราะดัชนีบิตแมปสามารถดำเนินการระดับบิต (AND, OR, XOR, NOT) ก่อนจะเข้าถึงข้อมูลจริง

การทำดัชนีบิตแมป (Bitmap Index) ในงานวิจัยที่ผ่านมามีการทำดัชนีบิตแมปหลายชนิดโดยแบ่งเป็น 2 กลุ่ม คือ กุล่มที่ใช้วิธีการบีบอัด และกลุ่มที่ใช้วิธีการลงรหัส สำหรับงานในวิทยานิพนธ์นี้ให้ความสนในกลุ่มดัชนีบิตแมปที่ใช้วิธีการลงรหัส ได้แก่ ดัชนีบิตแมปแบบพื้นฐาน (SBI) ดัชนีบิตแมปแบบช่วง (IBI) ดัชนีบิตแมปแบบกระจาย (ScaBI) ดัชนีบิตแมปแบบเข้ารหัส (E-EBI) และ ดัชนีบิตแมปแบบคู่กัน (DBI) โดยดัชนีบิตแมปแต่ละแบบมีวิธีการสร้าง และวิธีการสอบถามที่ไม่เหมือนกัน ประสิทธิภาพในเรื่องพื้นที่ที่ใช้จัดเก็บดัชนี และการสอบถามจึง

แตกต่างกันโดยที่ SBI มีประสิทธิภาพในเรื่องเวลาดีที่สุดแต่เรื่องพื้นที่การจัดเก็บดัชนีแย่ที่สุด ส่วน E-EBI มีประสิทธิภาพเรื่องของพื้นที่ที่ใช้จัดเก็บดัชนีดีที่สุด แต่เวลาที่ใช้ในการสอบถามแย่ที่สุด สำหรับ IBI, ScaBI และ DBI มีประสิทธิภาพในเรื่องเวลาในการสอบถามใกล้เคียงกัน ส่วนพื้นที่ที่ใช้จัดเก็บดัชนีจะเรียงจากมากไปน้อยโดยเริ่มจาก IBI, ScaBI และ DBI ตามลำดับ

ในวิทยานิพนธ์นี้ได้นำเสนอขั้นตอนการเพิ่มประสิทธิภาพของดัชนีบิตแมปแบบคู่กันโดยใช้วิธีการเข้ารหัส โดยจะใช้วิธีการเรียงลำดับค่าของแอททริบิวต์ที่มีการสอบถามบ่อย ๆ จากมากที่สุดไปน้อยที่สุด เพื่อที่จะหารูปแบบการลงรหัสให้เหมาะสมในการลงรหัส เพื่อลดจำนวนบิตแมปเวกเตอร์ในการสอบถาม สำหรับการสอบถามแบบค่าเท่ากันค่าของแอททริบิวต์ที่มีการสอบถามบ่อยจะใช้ประโยชน์จากวิธีการลงรหัสเพื่อเพิ่มความเร็วในการสอบถาม สำหรับการสอบถามแบบเป็นสมาชิก เมื่อมีการสอบถามค่าของแอททริบิวต์ที่ความถี่มีใกล้เคียงกันจะสามารถลดจำนวนบิตแมปเวกเตอร์ที่ใช้ในการสอบถามได้ดี เมื่อนำ EDBI ไปเปรียบเทียบกับดัชนีบิตแมปอีก 2 แบบคือ DBI และ E-EBI ดังแสดงในตาราง 5-1

ตาราง 5-1 แสดงการใช้พื้นที่และเวลาของ DBI, E-EBI และ EDBI

ดัชนีบิตแมป	**พื้นที่**	**จำนวนบิตที่ถูกตรวจสอบระหว่างจำนวนครั้งในการดำเนินการตรรกะ(เวลา)**	
	จำนวนบิตแมปเวกเตอร์ที่ใช้ในการสร้างดัชนี	การสอบถามแบบค่าเท่ากัน	การสอบถามแบบเป็นสมาชิก
DBI	$\left\lceil \sqrt{2C+0.25}+0.5 \right\rceil$	2: 1 (1 AND)	2*M*: 2 *M* -1 (*M* AND , *M* -1 OR)
E-EBI	$\left\lceil \log_2 C \right\rceil$	$\left\lceil \log_2 C \right\rceil$: 1 AND, $\left\lceil \log_2 C \right\rceil$ NOT	1 : $\left\lceil \log_2 C \right\rceil M$ (M-1 OR)
EDBI	$nB = 2\left\lceil \left(\log_2 \left\lceil \sqrt{2C+0.25}+0.5 \right\rceil \right) \right\rceil$	2 to nB-1: (1AND to nB-2 AND), XOR	1: *nB*/2 to nB*M* (0 to *M*-1 OR, *M*-1 AND)

- DBI ใช้พื้นที่มากที่สุดในดัชนีบิตแมปทั้ง 3 แบบ และการสอบถามข้อมูลแบบค่าเท่ากันจะต้องมีการตรวจสอบบิตแมปเวกเตอร์ 2 บิตแมปเวกเตอร์และมีการดำเนินการตรรกะ AND 1 ครั้ง สำหรับการสอบถามแบบเป็นสมาชิกจะตรวจสอบบิตแมปเวกเตอร์เพิ่มขึ้น

ตามจำนวนสมาชิก (*M*)ที่ทำการสอบถาม และดำเนินการตรรกะ AND จำนวน *M* ครั้ง และดำเนินการตรรกะ OR จำนวน *M*-1 ครั้ง

- E-EBI ใช้พื้นที่น้อยที่สุดในดัชนีบิตแมปทั้ง 3 แบบ สำหรับการสอบถามข้อมูลแบบค่าเท่ากันจะต้องมีการตรวจสอบบิตแมปเวกเตอร์ $\lceil \log_2 C \rceil$ บิตแมปเวกเตอร์ และมีการดำเนินการตรรกะ AND $\lceil \log_2 C \rceil$ ครั้ง และมีการดำเนินการตรรกะ NOT ไม่เกิน $\lceil \log_2 C \rceil$ ครั้ง สำหรับการสอบถามแบบเป็นสมาชิก ถ้ากรณีที่ดีที่สุดถ้าใช้วิธีการลดรูปจะตรวจสอบบิตแมปเวกเตอร์ 1 บิตแมปเวกเตอร์ กรณีแย่ที่สุดจะตรวจสอบบิตแมปเวกเตอร์เพิ่มขึ้นตามจำนวน M ค่า คือจะต้องตรวจสอบ $\lceil \log_2 C \rceil M$ บิตแมปเวกเตอร์ มีการดำเนินการตรรกะ OR เท่ากับ M-1 ครั้ง
- EDBI ใช้พื้นที่ใกล้เคียงกับ E-EBI เมื่อแอททริบิวต์มีค่าคาร์ดินอลิตี้สูงขึ้น สำหรับการสอบถามข้อมูลแบบค่าเท่ากันจะต้องมีการตรวจสอบบิตแมปเวกเตอร์น้อยสุดคือ 2 บิตแมปเวกเตอร์ ดำเนินการตรรกะ AND 1 ครั้ง ตรวจสอบบิตแมปเวกเตอร์มาสุดคือ *nB*-1 บิตแมปเวกเตอร์ (*nB* แทนจำนวนบิตแมปเวกเตอร์ทั้งหมด) มีการดำเนินการตรรกะ AND nB-2 ครั้ง และดำเนินการตรรกะ XOR กับเรอคร์ดที่เหลือ สำหรับการสอบถามแบบเป็นสมาชิก ตรวจสอบบิตแมปเวกเตอร์เพิ่มขึ้นตามจำนวน *M* ค่า มีการตรวจสอบบิตแมปเวกเตอร์น้อยที่สุดคือ 1 บิตแมปเวกเตอร์ และ *nB*/2 บิตแมปเวกเตอร์ กรณีที่สมาชิกที่สอบถามมีความถี่ใกล้เคียงกัน และ *nBM* กรณีที่สมาชิกที่สอบถามมีความถี่ต่างกัน และมีการดำเนินการตรรกะ AND *M*-1 ครั้ง และมีการดำเนินการตรรกะ OR *M*-1 ครั้ง

การทำดัชนีบิตแมปแต่ละแบบมีจุดประสงค์ที่แตกต่างกัน เช่น คำนึงถึงประสิทธิภาพในเรื่องของพื้นที่ที่ใช้จัดเก็บดัชนี หรือประสิทธิภาพในเรื่องของเวลา และประสิทธิภาพในเรื่องของ Space-Time Trade-off จึงสรุปได้ดังตาราง 5-2

ตาราง 5-2 ประสิทธิภาพที่ต้องการสำหรับการเลือกใช้ดัชนีบิตแมป

ประสิทธิภาพที่ต้องการ	ดัชนีบิตแมปที่เลือกใช้
ประสิทธิภาพในเรื่องของพื้นที่	E-EBI
ประสิทธิภาพในเรื่องของเวลา	DBI
ประสิทธิภาพในเรื่องของ Space-Time Trade-off	EDBI

จากตาราง 5-2 E-EBI ใช้พื้นที่ในการจัดเก็บดัชนีน้อยจึงเหมาะสำหรับสร้างดัชนีบิตแมปบนแอททริบิวต์ที่มีคาร์ดินอลิตี้สูง ๆ ในกรณีที่มีพื้นที่จำกัด แต่เรื่องเวลาที่ใช้ในการสอบถามจะช้า ส่วน DBI ใช้พื้นที่ในการจัดเก็บดัชนีมากจึงเหมาะสำหรับสร้างดัชนีบิตแมปบนแอททริบิวต์ที่มีคาร์ดินอลิตี้ต่ำ ในกรณีที่มีพื้นที่มาก แต่ต้องการเวลาในการสอบถามที่รวดเร็วและ EDBI ใช้พื้นที่ในการจัดเก็บดัชนีได้มีประสิทธิภาพ สำหรับสร้างดัชนีบิตแมปบนแอททริบิวต์ที่มีคาร์ดินอลิตี้สูง และเหมาะกับพื้นที่ที่มีจำกัด และยังคงมีประสิทธิภาพในเรื่องของการสอบถาม และต้องการประสิทธิภาพในเรื่องของ Space-Time Trade-off

5.2 ข้อจำกัดของ EDBI

EDBI มีการใช้ Query Workload มาช่วยในการลงรหัส ถ้าค่าที่ได้จาก Query Workload ผิดพลาด การลงรหัสให้กับค่าแอททริบิวต์นั้นผิดพลาด ทำให้เวลาที่ใช้ในการสอบถามเพิ่มขึ้น และทำให้ประสิทธิภาพในการสอบถามลดลง ดังนั้นถ้าหากมีเงื่อนไขที่ไม่เคยมีการสอบถามมาก่อนจะทำให้ การลงรหัสให้กับค่าแอททริบิวต์นั้นมีการลงรหัสเป็นบิต 0 จำนวนมาก อาจทำให้จำนวนเรคอร์ดที่ต้องดำเนินการ XOR มีจำนวนมากและเวลาในการสอบถามก็เพิ่มขึ้น

5.3 ข้อเสนอแนะและงานในอนาคต

- ประสิทธิภาพของ EDBI ขึ้นอยู่กับ Query Workload ดังนั้นควรเลือก Query Workload ให้มีความสอดคล้องกับความต้องการและเป้าหมายของงาน
- สำหรับการสอบถามแบบเป็นสมาชิกควรเลือกวิธีการลดรูปฟังก์ชันก่อนการเข้าถึงข้อมูลที่มีประสิทธิภาพจะทำให้ได้จำนวนบิตแมปเวกเตอร์ที่น้อย และทำให้เวลาที่ใช้ในการสอบถามแบบเป็นสมาชิกเร็วขึ้นด้วย
- การเรียงข้อมูลกรณีความถี่เท่ากันในขั้นตอน Sorted Frequent Table Creation ใช้วิธีการเรียงความถี่จากความถี่มากไปหาความถี่น้อย ในงานวิจัยนี้มีข้อด้อยอยู่คือ ถ้าความถี่ของค่าแอททริบิวต์เท่ากัน มีแนวทางในการแก้ปัญหาสำหรับงานในอนาคต 3 ข้อดังนี้

1. ใช้วิธีการเก็บข้อมูลค่าแอททริบิวต์ที่เคยลงรหัสแล้วในอดีตเพื่อช่วยในการจัดเรียงค่าแอททริบิวต์ที่มีความถี่เท่ากันได้เหมาะสม

2. ใช้วิธีการดูรายละเอียดข้อมูลดิบในตารางข้อมูล โดยหาค่าที่เกิดบ่อยครั้งที่สุด (Mode) ซึ่งมีสมมติฐานที่ว่า ค่าของแอททริบิวต์ใดที่มีจำนวนมากในตารางข้อมูล อาจจะมีแนวโน้มที่จะถูกสอบถามมากด้วย
3. ใช้วิธีการ Frequent Item set ในการหาแอททริบิวต์ที่เกิดพร้อมกัน โดยสามารถใช้คู่กับวิธีการในข้อที่ 1 หรือ 2 เพื่อใช้แก้ปัญหาการลงรหัสค่าแอททริบิวต์ที่ถูกสอบถามพร้อมกันให้ลงรหัสใกล้เคียงกันสำหรับการสอบถามแบบเป็นสมาชิก

บรรณานุกรม

Chan, C. Y., and Ioannidis, Y. E. 1998. Bitmap Index Design and Evaluation. In ACM SIGMOD Record ,Vol. 27, No. 2, pp. 355-366

Chan, C. Y., and Ioannidis, Y. E. 1999, June. An Efficient Bitmap Encoding Scheme for Selection Queries. In ACM SIGMOD Record, Vol. 28, No. 2, pp. 215-226

Chaudhuri, S., and Dayal, U. 1997. An Overview of Data Warehousing and OLAP Technology. ACM Sigmod record, Vol.26, pp. 65-74.

Deliège, F., and Pedersen, T. B. 2010. Position List Word Aligned Hybrid: Optimizing Space and Performance for Compressed Bitmaps. In Proceedings of the 13th International Conference on Extending Database Technology. pp. 228-239.

Inmon, W. H. 2005. Building the Data Warehouse., Wiley Computer Publishing.

Keawpibal, A., Wattanakitrungroj, N., and Vanichayobon, S. 2012. Enhanced Encoded Bitmap Index for Equality Query. In Computing Technology and Information Management (ICCM), 2012 8th International Conference, Vol. 1, pp. 293-298.

Kimball, R. 2006. The Data Warehouse Toolkit. Wiley Computer Publishing.

O'Neil, P., and Quass, D. 1997. Improved Query Performance with Variant Indexes. In ACM Sigmod Record, Vol. 26, No. 2, pp. 38-49.

Rainardi, V. 2008. Building a Data Warehouse: with Examples in SQL Server. Springer-Verlag: New York

Silberschatz, A., Korth, H. F., and Sudarshan, S. 2002. Database System Concepts, Vol. 4. New York: McGraw-Hill.

Stabno, M., and Wrembel, R. 2009. RLH: Bitmap Compression Technique Based on Run-Length and Huffman Encoding. Information Systems,Vol.34, pp. 400-414.

Stockinger, K., and Wu, K. 2006. Bitmap Indices for data warehouses. In In Data Warehouses and

OLAP. 2007. IRM.

Transaction Processing Performance Council (TPC), “TPC-H: An Ad Hoc Decision Support Benchmark”, Version2.14.1, [Online]. Availablefrom: http://www.tpc.org/tpch., [2013, December 11].

Vanichayobon, S., Manfuekphan, J., and Gruenwald, L. 2006. Scatter Bitmap: Space-Time Efficient Bitmap Indexing for Equality and Membership Queries. In Cybernetics and Intelligent Systems, 2006 IEEE Conference. pp. 1-6.

Wattanakitrungroj, N., and Vanichayobon, S. 2006. Dual Bitmap Index: Space-Time Efficient Bitmap Index for Equality and Membership Queries. In Communications and Information Technologies, 2006. ISCIT'06. International Symposium. pp. 568-573.

Wu, K., Otoo, E. J., and Shoshani, A. 2006. Optimizing Bitmap Indices with Efficient Compression. ACM Transactions on Database Systems (TODS), Vol.31(1), pp. 1-38.

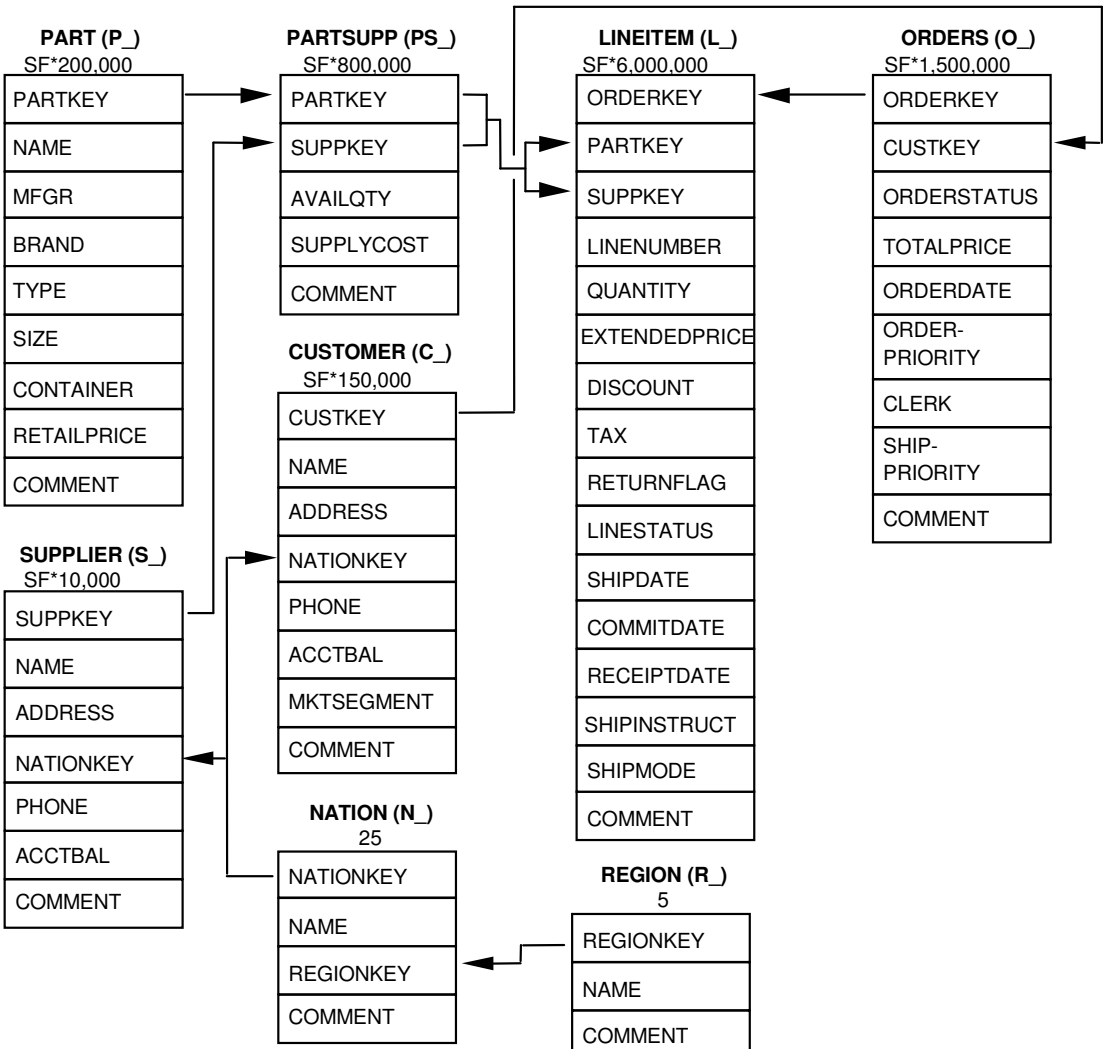
Wu, M. C., and Buchmann, A. P. 1998. Encoded Bitmap Indexing for Data Warehouses. In Data Engineering, 1998. Proceedings., 14th International Conference. pp. 220-230.

ภาคผนวก

ภาคผนวก ก

ก.1 TPC-H Benchmark

ข้อมูลที่ใช้ในการทดลองในวิทยานิพนธ์เล่มนี้ เป็นข้อมูลมาตรฐานจาก TPC-H (Transaction Processing Performance Council, 2013) ซึ่งเป็นการวัดประสิทธิภาพการสอบถามที่ซับซ้อนที่ใช้ในระบบสนับสนุนการตัดสินใจ



ภาพประกอบ ก-1 โครงสร้างข้อมูลของ TCP-H

ก.2 ข้อมูลแอททริบิวต์ที่ใช้ในการทดลอง

ข้อมูลที่ใช้ทำการทดลองมี 3 ชุด มาจากตาราง PART ซึ่งมีรายละเอียดดังนี้

ข้อมูลชุดที่ 1 แอททริบิวต์ BRAND บนตาราง PART มีจำนวนเรคอร์ดเท่ากับ 7,000,000 เรคอร์ด มีคาร์ดินอลิตี้เท่ากับ 25 (C = 25)

ข้อมูลชุดที่ 2 แอททริบิวต์ SIZE บนตาราง PART มีจำนวนเรคอร์ดเท่ากับ 7,000,000 เรคอร์ด มีคาร์ดินอลิตี้เท่ากับ 50 (C = 50)

ข้อมูลชุดที่ 3 แอททริบิวต์ TYPE บนตาราง PART มีจำนวนเรคอร์ดเท่ากับ 7,000,000 เรคอร์ด มีคาร์ดินอลิตี้เท่ากับ 150 (C = 150)

ก.3 โครงสร้างตารางของการวัดเปรียบเทียบสมรรถนะ

โครงสร้างตารางที่ใช้วัดเปรียบเทียบประสิทธิภาพคือตาราง PART ซึ่งเป็นตารางเก็บข้อมูลเกี่ยวกับชิ้นส่วนของสินค้า โดยมีโครงสร้างตารางดังนี้

โครงสร้างตาราง PART

แอททริบิวต์	ชนิดข้อมูล	หมายเหตุ
P_PARTKEY	Identifier	SF*200,000 are populated
P_NAME	Variable text, size 55	
P_MFGR	Fixed text, size 25	
P_BRAND	Fixed text, size 10	
P_TYPE	Variable text, size 25	
P_SIZE	Integer	
P_CONTAINER	Fixed text, size 10	
P_RETAILPRICE	Decimal	
P_COMMENT	Variable text, size 23	

Primary Key: P_PARTKEY

ก. 4 การจัดเตรียมข้อมูลที่ใช้ในการทดลอง

การเตรียมข้อมูลจากแอททริบิวต์ที่เลือกมาทำดัชนีเพื่อใช้ในการสอบถามข้อมูลแบบค่าเท่ากันและสอบถามข้อมูลแบบเป็นสมาชิกบนดัชนีบิตแมปแบบคู่กัน (DBI), ดัชนีบิตแมปแบบเข้ารหัสที่ใช้เทคนิคการจัดกลุ่ม E-EBI และ Enhance Dual Bitmap Index (EDBI) มีขั้นตอนการเตรียมข้อมูลดังต่อไปนี้

1. ติดตั้งโปรแกรม GnuWin32 (gawk-3.1.6-1) เพื่อช่วยในการเลือกแอททริบิวต์ที่นำมาทำดัชนีโดยใช้ MS-DOS

2. การเลือกแอททริบิวต์ที่นำมาทำดัชนีจะใช้คำสั่ง gawk –F "|" "{print $number}" input_filename> output_filename โดยที่ "|" เป็นเครื่องหมายที่คั่นระหว่างแอททริบิวต์ number คือ ลำดับของแอททริบิวต์ที่ต้องการให้อยู่ใน output_filename เช่น gawk –F "|" "{print $5}" part.tbl > part_type.txt เป็นการเลือกแอททริบิวต์ลำดับที่ 5 จากตาราง PART โดยเก็บผลลัพไว้ใน part_type.txt ตัวอย่าง part.tbl แสดงในภาพประกอบ ก-2 และ part_type.txt แสดงในภาพประกอบ ก-3

3. เปลี่ยนค่าของแอททริบิวต์ที่ได้จากขั้นตอนที่สองที่เก็บในไฟล์ part_type7M.txt ให้เป็นจำนวนเต็มที่มีค่าต่อเนื่องกันเริ่มจาก 0, 1, 2 ไปจนถึง *C*-1 โดยการสร้าง file.gawk ตัวอย่างชุดคำสั่งในไฟล์ file.gawk

/ PROMO BURNISHED COPPER/ {print 0}

(เปลี่ยน PROMO BURNISHED COPPER เป็น 0)

/ LARGE BRUSHED BRASS/ {print 1}

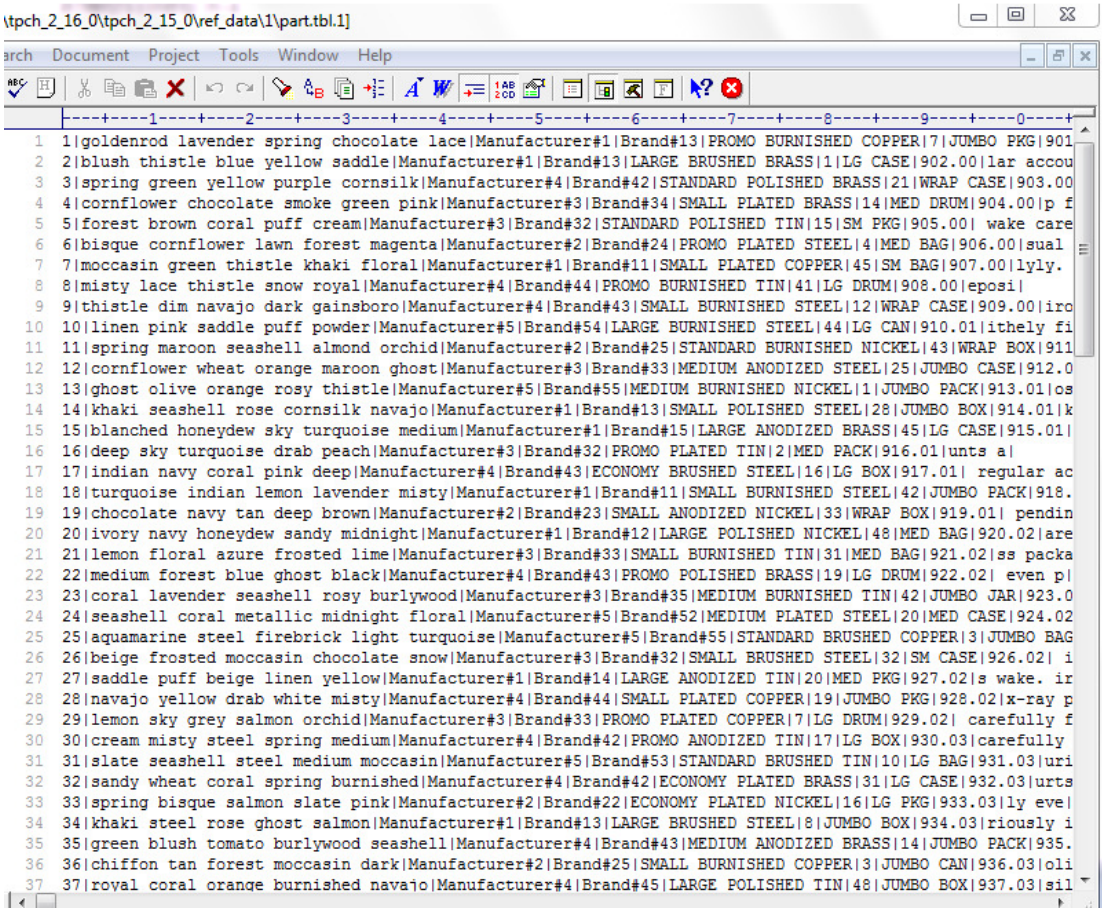
(เปลี่ยน LARGE BRUSHED BRASS เป็น 1)

/STANDARD POLISHED BRASS/ {print 1}

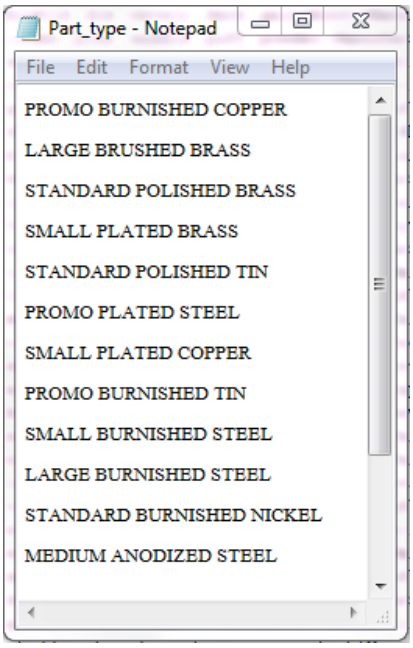
(เปลี่ยน STANDARD POLISHED BRASS เป็น 2)

จากนั้นใช้คำสั่ง gawk –F file.gawk input_filename> output_filename

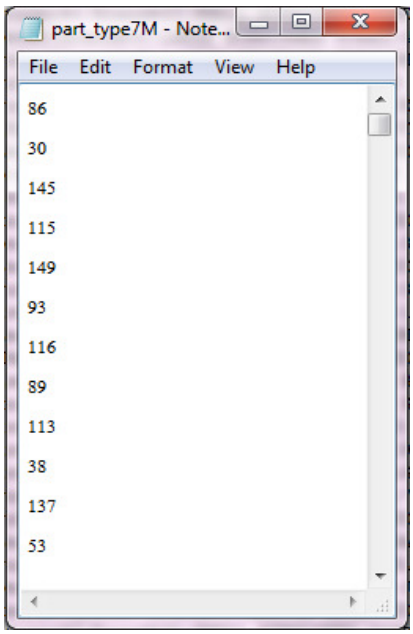
เช่น gawk –F file.gawk part_type.txt > part_type7M.txt ไฟล์ part_type7M.txt ใช้เก็บข้อมูลค่าของแอททริบิวต์ที่เลือก และมีการเปลี่ยนข้อมูลให้อยู่ในรูปแบบของจำนวนเต็มที่ต่อเนื่องกัน ตัวอย่าง part_type7M.txt แสดงในภาพประกอบ ก-4 เป็นตัวอย่างข้อมูลของแอททริบิวต์ TYPE ในตาง PART ที่มีค่าคาร์ดินอลิตี้เท่ากับ 150 แสดงว่าข้อมูลที่อยู่ในรูปแบบของจำนวนเต็มจะมีค่าตั้งแต่ 0 ถึง 149 ตัวอย่าง



ภาพประกอบ ก-2 ตัวอย่างข้อมูลในตาราง PART



ภาพประกอบ ก-3 ไฟล์ค่าของแอททริบิวต์ TYPE จาก ไฟล์ part.tbl



ภาพประกอบ ก-4 ไฟล์ที่ได้จากการเปลี่ยนรูปแบบของแอททริบิวต์ TYPE ให้อยู่ในรูปแบบ

ก.5 การเขียนโปรแกรมเพื่อทำการทดลองการสอบถามข้อมูลแบบค่าเท่ากันและการสอบถามข้อมูลแบบเป็นสมาชิกบนดัชนีบิตแมป

ในการเขียนโปรแกรมเพื่อทำการสอบถามข้อมูลแบบค่าเท่ากันและการสอบถามข้อมูลแบบเป็นสมาชิก มีฟังก์ชันและไฟล์ที่ใช้สอบถามดังต่อไปนี้

- ตัวแปร (variable)

V	ใช้สำหรับเก็บค่าที่ต้องการสอบถาม ($0 \le V \le C-1$)
CARDINALITY	ใช้สำหรับเก็บค่าของคาร์ดินอลิตี้
Bitmapvactor	ใช้เก็บค่าบิตแมปเวกเตอร์ที่ต้องอ่าน

- ฟังก์ชัน (function)

equal_query_dbi(value)	เป็นฟังก์ชันที่ใช้สอบถามแบบค่าเท่ากันบนดัชนีบิตแมปแบบคู่กัน
equal_query_edbi(value)	เป็นฟังก์ชันที่ใช้สอบถามแบบค่าเท่ากันบน Enhance Dual Bitmap Index
equal_query_e-ebi(value)	เป็นฟังก์ชันที่ใช้สอบถามแบบค่าเท่ากันบนดัชนีบิตแมปแบบเข้ารหัสที่ใช้เทคนิคการจัดกลุ่ม
member_query_dbi(value)	เป็นฟังก์ชันที่ใช้สอบถามแบบเป็นสมาชิกบนดัชนี

	บิตแมปแบบคู่กัน
member_query_edbi(value)	เป็นฟังก์ชันที่ใช้สอบถามแบบเป็นสมาชิกบน Enhance Dual Bitmap Index
member_query_e-ebi(value)	เป็นฟังก์ชันที่ใช้สอบถามแบบเป็นสมาชิกบนดัชนีบิตแมปแบบเข้ารหัสที่ใช้เทคนิคการจัดกลุ่ม

- ไฟล์ (file)

dbi_25.txt	เป็นไฟล์ที่เก็บดัชนีบิตแมปแบบคู่กันที่มีคาร์ดินอลิตี้เท่ากับ 25
edbi_25.txt	เป็นไฟล์ที่เก็บ Enhance Dual Bitmap Index ที่มีคาร์ดินอลิตี้เท่ากับ 25
e-ebi_25.txt	เป็นไฟล์ที่เก็บดัชนีบิตแมปแบบแบบเข้ารหัสที่ใช้เทคนิคการจัดกลุ่มที่มีคาร์ดินอลิตี้เท่ากับ 25
dbi_50.txt	เป็นไฟล์ที่เก็บดัชนีบิตแมปแบบคู่กันที่มีคาร์ดินอลิตี้เท่ากับ 50
edbi_50.txt	เป็นไฟล์ที่เก็บ Enhance Dual Bitmap Index ที่มีคาร์ดินอลิตี้เท่ากับ 50
e-ebi_50.txt	เป็นไฟล์ที่เก็บดัชนีบิตแมปแบบแบบเข้ารหัสที่ใช้เทคนิคการจัดกลุ่มที่มีคาร์ดินอลิตี้เท่ากับ 50
dbi_150.txt	เป็นไฟล์ที่เก็บดัชนีบิตแมปแบบคู่กันที่มีคาร์ดินอลิตี้เท่ากับ 150
edbi_150.txt	เป็นไฟล์ที่เก็บ Enhance Dual Bitmap Index ที่มีคาร์ดินอลิตี้เท่ากับ 150
e-ebi_150.txt	เป็นไฟล์ที่เก็บดัชนีบิตแมปแบบแบบเข้ารหัสที่ใช้เทคนิคการจัดกลุ่มที่มีคาร์ดินอลิตี้เท่ากับ 150

ภาคผนวก ข

ผลงานวิจัยที่ได้รับการตีพิมพ์

เรื่อง	การเพิ่มประสิทธิภาพดัชนีบิตแมปแบบคู่กันโดยใช้วิธีการเข้ารหัส
งานประชุมวิชาการ	The 10th National Conference on Computing and Information Technology
สถานที่	จังหวัดภูเก็ต ประเทศไทย
วันที่	ระหว่างวันที่ 8 – 9 พฤษภาคม 2557

การเพิ่มประสิทธิภาพดัชนีบิตแมปแบบคู่กันโดยใช้วิธีการเข้ารหัส
Enhancing Dual Bitmap Index with Efficient Encoding

ศิเนตร กิ้มเส้ง(Sinate Kimseng) [1] และ ศิริรัตน์ วณิชโยบล(Sirirut Vanichayobon)[2]
ห้องปฏิบัติการวิจัยเทคโนโลยีระบบสารสนเทศและการประยุกต์ (iSTAR Lab) ภาควิชาวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์ มหาวิทยาลัยสงขลานครินทร์ วิทยาเขตหาดใหญ่
[1]5510220103@email.psu.ac.th, [2]sirirut.v@psu.ac.th

บทคัดย่อ

คลังข้อมูลเป็นที่เก็บข้อมูลสำหรับสนับสนุนการตัดสินใจของผู้บริหาร ลักษณะการสอบถามที่เกิดขึ้นบ่อย ๆ ในคลังข้อมูลเป็นการสอบถามแบบทันทีทันใด ดัชนีบิตแมปเป็นเทคนิคที่นิยมใช้ค้นหาข้อมูลที่ต้องการ เนื่องจากการดำเนินการ Boolean ที่มีประสิทธิภาพของ AND OR และ NOT อย่างไรก็ตามดัชนีบิตแมปเหมาะสมสำหรับแอททริบิวต์ที่มีคาร์ดินอลิตี้ต่ำเท่านั้น บทความนี้นำเสนอเทคนิคการเพิ่มประสิทธิภาพดัชนีบิตแมปแบบคู่กันโดยใช้วิธีการเข้ารหัสที่เหมาะสมทำให้สามารถนำไปใช้กับแอททริบิวต์ที่มีคาร์ดินอลิตี้สูงขึ้นขณะที่ยังคงรักษาประสิทธิภาพในการสอบถาม

คำสำคัญ: คลังข้อมูล ดัชนีบิตแมป ดัชนีบิตแมปแบบคู่กัน

Abstract

A data warehouse is a repository data for supporting decision makers. Most of queries against the data warehouse are ad hoc query. Bitmap index is a popular technique used to retrieve target records because of the efficient Boolean operation AND, OR and NOT on bits. However, bitmap index is suitable for only low cardinality attributes. This paper proposes a technique to enhance the Dual bitmap index to optimize the use of space in high cardinality attributes while maintaining query processing time performance.

Keyword: Data Warehouse, Bitmap Index, Dual Bitmap Index

1. บทนำ

คลังข้อมูล (Data Warehouse) เป็นที่ใช้เก็บรวบรวมข้อมูลให้อยู่ในที่เดียวกัน มีการจัดโครงสร้างตามเนื้อหาที่เราสนใจ [1] โดยข้อมูลที่เก็บรวบรวมนั้นจะเป็นข้อมูลตั้งแต่อดีตจนถึงปัจจุบันเพื่อสนับสนุนการตัดสินใจ [2] ในระบบคลังข้อมูลการประมวลผลข้อมูลจะเป็นแบบ OLAP (Online Analytical Processing)[3] และแบบทันทีทันใด (Ad Hoc) และการสอบถามส่วนใหญ่ซับซ้อนทำให้การประมวลผลข้อมูลต้องใช้เวลานาน จากปัญหาที่กล่าวมาจึงมีการทำงานวิจัยที่เกี่ยวข้องกับคลังข้อมูลมากมายส่วนใหญ่แล้วจะเน้นสองเรื่องหลัก คือ เรื่องของพื้นที่ที่ใช้และเรื่องของเวลาในการเข้าถึงข้อมูล เมื่อกล่าวถึงการเข้าถึงข้อมูล วิธีการเพิ่มความเร็วในการเข้าถึงข้อมูลมีหลายวิธีด้วยกันยกตัวอย่างเช่น วิธีการ Hashing เหมาะกับการเข้าถึงโดยตรง การประมวลผลแบบ Parallel เป็นการเพิ่ม Hardware เพื่อให้ทำงานเร็วขึ้นและ วิธีการทำดัชนี เป็นวิธีการเพิ่มความเร็วในการเข้าถึงข้อมูลโดยไม่ต้องเสียค่าใช้จ่ายใด ๆ แต่เป็นการใช้พื้นที่แลกกับเวลาในการสอบถามที่เร็วขึ้น

การทำดัชนีมีหลายแบบ เช่น B-Tree, B+ Tree และดัชนีบิตแมป ในบทความนี้จะกล่าวถึงเฉพาะดัชนีบิตแมปเนื่องจากดัชนีบิตแมปมีความนิยมในการใช้ในการประมวลผลแบบทันทีทันใด [1, 3, 4] เพราะสามารถดำเนินการระดับบิต (AND, OR, XOR, NOT) ก่อนจะเข้าถึงข้อมูลจริง

การทำดัชนีบิตแมป (Bitmap Index) ในงานวิจัยที่ผ่านมามีการทำดัชนีบิตแมปหลายชนิดโดยแบ่งเป็น 2 กลุ่ม คือ กุล่มที่ใช้วิธีการบีบอัด และกลุ่มที่ใช้วิธีการลงรหัส ยกตัวอย่างดัชนีบิตแมปที่ใช้วิธีลงรหัส 5 ชนิด คือ ดัชนีบิตแมปแบบพื้นฐาน (Simple Bitmap Index) [4] ดัชนีบิตแมปแบบช่วง (Interval Bitmap Index) [5] ดัชนีบิตแมปแบบกระจาย (Scatter Bitmap Index) [6] ดัชนีบิตแมปแบบเข้ารหัส (Encoded Bitmap Index) [7, 8] และ ดัชนีบิตแมปแบบคู่กัน (Dual Bitmap Index) [9]

จากการศึกษางานวิจัยที่ผ่านมาผู้วิจัยสนใจการดัชนีบิตแมปแบบคู่กัน เพราะยังมีข้อบกพร่องที่สามารถแก้ไขได้ในเรื่องของพื้นที่จัดเก็บดัชนี ผู้วิจัยจึงคิดค้นวิธีการเพิ่มประสิทธิภาพให้กับดัชนีบิตแมปแบบคู่กันขึ้น โดยเล็งเห็นข้อดีและข้อเสียของดัชนีบิตแมปแบบคู่กันคือ หนึ่งแอททริบิวต์จะแทนด้วย 2 บิตแมปเวกเตอร์ ส่วนข้อเสียของดัชนีบิตแมปแบบคู่กันคือขนาดพื้นที่จัดเก็บดัชนีบิตแมปยังใช้พื้นที่มาก ดังนั้นจึงนำเสนอวิธีการ เพิ่มประสิทธิภาพดัชนีบิตแมปแบบคู่กันเพื่อลดพื้นที่ในการจัดเก็บดัชนีให้น้อยลงโดยใช้วิธีการเข้ารหัส และใช้เวลาการสอบถามที่เท่ากันในบางกรณี

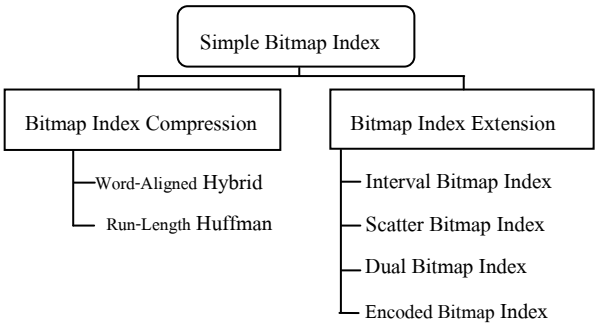
เอกสารนี้ประกอบด้วย 5 ส่วน ส่วนที่ 2 กล่าวถึงดัชนีบิตแมปแบบคู่กัน ส่วนที่ 3 วิธีการที่นำเสนอ ส่วนที่ 4 ผลการวิจัยและ ส่วนที่ 5 ส่วนสรุปและวิจารณ์ผล

2. ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ดัชนีบิตแมป

ดัชนีบิตแมปเป็นเทคนิคที่ช่วยให้เราสามารถค้นหาข้อมูลให้เร็วขึ้นโดยตารางดัชนีจะแทนด้วยบิต 0 และ 1 ดัชนีบิตแมปแบบพื้นฐานมีการแทนค่าข้อมูล 1 บิตแมปเวกเตอร์ต่อ 1 ค่าของแอททริบิวต์ซึ่ง S^v มีค่าเท่ากับ 1 เมื่อเรคอร์ด $i = v$ ข้อดีคือ ใช้เวลาสอบถามน้อย ข้อเสียคือ ถ้าคาร์ดินอลิตี้ C มากจะใช้พื้นที่มากดังนั้นจึงได้มีการพัฒนาเทคนิคเพิ่มประสิทธิภาพของดัชนีบิตแมปแบบพื้นฐานแบ่งเป็น 2 กลุ่มคือ 1. การบีบอัด (Bitmap Index Compression) และ2. การลงรหัส (Bitmap Index Extension) ดังแสดงในภาพที่ 1

กลุ่มที่ใช้วิธีการบีบอัด ประกอบด้วย WAH (Word – Aligned Hybrid) [10] และ RLH (Run – Length Huffman) [11] ข้อดีก็คือ ลดพื้นที่เก็บดัชนี แต่ข้อเสียก็คือ เวลาสอบถามจะต้องแปลงกลับไปให้อยู่ในรูปแบบของดัชนีบิตแมปแบบพื้นฐานและสอบถามแบบเดียวกับดัชนีบิตแมปแบบพื้นฐาน



ภาพที่ 1: การเพิ่มประสิทธิภาพในเชิงพื้นที่ของดัชนีบิตแมป [8]

กลุ่มที่ใช้วิธีการลงรหัส ประกอบด้วย 1) ดัชนีบิตแมปแบบช่วง [5] เป็นดัชนีบิตแมปที่ใช้พื้นที่ได้มีประสิทธิภาพกว่าดัชนีบิตแมปแบบพื้นฐาน การสอบถามข้อมูล 2 บิตแมปและมีโอกาสตรวจสอบ 1 บิตแมปเวกเตอร์เมื่อคาร์ดินอลิตี้น้อยกว่าหรือเท่ากับ 3 2) ดัชนีบิตแมปแบบกระจาย[6] ใช้พื้นที่ในการจัดเก็บดัชนีน้อยกว่าดัชนีบิตแมปแบบช่วง การสอบถามข้อมูลจะดำเนินการกับ 2 บิตแมปเวกเตอร์ 3) ดัชนีบิตแมปแบบคู่กัน [9] ใช้บิตแมปเวกเตอร์น้อยกว่าดัชนีบิตแมปแบบกระจาย การสอบถามข้อมูลจะดำเนินการระหว่าง 2 บิตแมปเวกเตอร์ และ 4) ดัชนีบิตแมปแบบเข้ารหัส [7, 8] เป็นดัชนีบิตแมปที่ใช้พื้นที่น้อยที่สุด แต่การสอบถามข้อมูลแบบดำเนินการกับทุกบิตแมปเวกเตอร์ ข้อดีของดัชนีบิตแมปกลุ่มที่ 2 ใช้วิธีการลงรหัสคือการสอบถาม (Query) สามารถทำได้ทันที และมีประสิทธิภาพในเรื่องของการลดพื้นที่การจัดเก็บดัชนีพอสมควร

2.2 ดัชนีบิตแมปแบบคู่กัน (Dual Bitmap Index)

ดัชนีบิตแมปแบบคู่กันเป็นเทคนิคที่นำเสนอโดย Wattanakitrungroj และ Vanichayobon [9] วิธีการสร้างดัชนีบิตแมปแบบคู่กันมีขั้นตอนดังนี้ (กำหนดให้ C แทนค่าคาร์-ดินอลิตี้)

1. คำนวณหาจำนวนบิตที่ใช้แทนค่าแต่ละค่า

$$n = \left\lceil \sqrt{2C + 0.25} + 0.5 \right\rceil \quad (1)$$

2. กำหนดค่า 0 ถึง C-1 ให้แก่ค่าของแอททริบิวต์
3. คำนวณหาตำแหน่งบิตที่ต้องให้ค่าเท่ากับ 1 โดยใช้สมการ

$$r_i = \left\lceil \sqrt{2v_i + 2.25} - 0.5 \right\rceil \quad (2)$$

$$s_i = v_i - \frac{r(r-1)}{2} \quad (3)$$

ตัวอย่างการสร้างดัชนีบิตแมปแบบคู่กันบนแอททริบิวต์ X ที่คาร์ดินอลิตี้ C=16 (คือค่า A ถึง P) จะได้ n = 7 กำหนดค่าตัวเลขให้กับ A ถึง P เช่น A=0, B=1 เป็นต้น ตารางที่ 1 แสดงค่าของแอททริบิวต์ X

ตารางที่ 1: ตารางแสดงค่าของแอททริบิวต์ X

X	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
v	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

ถ้าต้องการลงรหัสค่า F (v=5) จากสมการที่ (2) และ (3) ได้ r = 3 และ s = 2 ดังนั้นสองบิตที่เท่ากับ 1 คือตำแหน่งที่ 3 (D^3) และ ตำแหน่งที่ 2 (D^2) ดังที่แสดงในตารางที่ 2(ข)

ตารางที่ 2: ตัวอย่างการลงรหัสดัชนีบิตแมปแบบคู่กัน (C=16)

RID	**X**	**D^6**	**D^5**	**D^4**	**D^3**	**D^2**	**D^1**	**D^0**
1	K	0	1	0	0	0	0	1
2	F	0	0	0	1	1	0	0
3	D	0	0	0	1	0	0	1
4	P	1	0	0	0	0	0	1
5	N	0	1	0	1	0	0	0
6	B	0	0	0	0	1	0	1
7	L	0	1	0	0	0	1	0
8	E	0	0	0	1	0	1	0
9	A	0	0	0	0	0	1	1
10	F	0	0	0	1	1	0	0

$D^3 \wedge D^2$
0
1
0
0
0
0
0
0
0
1

(ก) แอททริบิวต์ (ข) ดัชนีบิตแมปแบบคู่กัน (ค)ผลลัพธ์

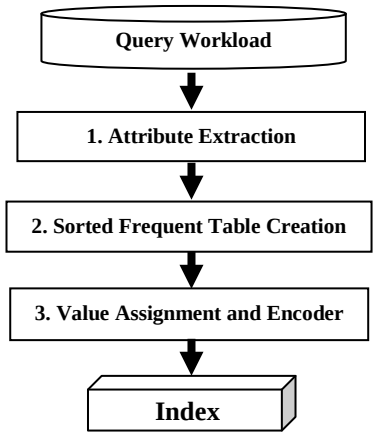
การสอบถามแบบค่าเท่ากันบนดัชนีบิตแมปแบบคู่กันทำได้โดย $D^r \wedge D^s$ เช่น ถ้าต้องการสอบถามค่า F (v = 5) จากสมการ (2) และ (3) ได้ r= 3 และ s= 2 แสดงว่าจะต้องทำการ AND ระหว่างบิต D^3 และ D^2 ได้ผลลัพธ์ดังภาพที่ 2(ค)

ข้อดีของดัชนีบิตแปแบบคู่กันมีการดำเนินการกับบิตแมปเวกเตอร์ 2 บิตเท่านั้นส่วน ข้อเสียคือยังคงใช้พื้นที่ในการจัดเก็บดัชนีมาก

3. Enhancing Dual Bitmap Index

3.1 การสร้าง Enhancing Dual Bitmap Index (EDBI)

การสร้าง EDBI แบ่งเป็น 3 ขั้นตอนคือ 1) Attribute Extraction 2) Sorted Frequent Table Creation และ 3) Value Assignment and Encoder ดังภาพที่ 2



ภาพที่ 2: ขั้นตอนการเพิ่มประสิทธิภาพดัชนีบิตแมปแบบคู่กัน

1). Attribute Extraction

ค่าแอททริบิวต์จาก Query Workload จะถูกสกัดออกมาในขั้นตอนนี้ ภาพที่ 3 แสดงตัวอย่างของ Query Workload 4 รายการและตารางที่ 3 แสดงผลลัพธ์ที่ได้จากการสกัดค่าในแอททริบิวต์ X จาก Query Workload

Q1: SELECT * FROM T WHERE X IN (A, E, G, O, P, C, M, N)

Q2: SELECT * FROM T WHERE X = A OR X = E

Q3: SELECT * FROM T WHERE X = E OR X = D

Q4: SELECT * FROM T WHERE X = D

ภาพที่ 3: Query Workload

ตารางที่ 3: ตารางค่าของแอททริบิวต์

Query	**ค่าของแอททริบิวต์ X**
Q1	A, E, G, O, P, C, M, N
Q2	A, E
Q3	E, D
Q4	D

2). Sorted Frequent Table Creation

ในขั้นตอนนี้ค่าความถี่(*f*) ของค่าของแอททริบิวต์ที่ถูกสกัดแต่ละตัวจะถูกคำนวณและจัดเก็บไว้ในตารางแสดงความถี่ (Frequent Table) จากนั้นเรียงค่าของแอททริบิวต์ X ตามความถี่จากค่ามากไปหาค่าน้อยและจัดเก็บในตารางจัดเรียงความถี่ (Sorted Frequent Table) ตารางที่ 4 และ 5 แสดง Frequent Table และ Sorted Frequent Table ของแอททริบิวต์ X ตามลำดับ

ตารางที่ 4: Frequent Table ของแอททริบิวต์ X

X	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
f	2	0	1	2	3	0	1	0	0	0	0	0	1	1	1	1

ตารางที่ 5: Sorted Frequent Table ของแอททริบิวต์ X

X	E	A	D	C	G	M	N	O	P	B	H	I	J	L	F	K

3). Value Assignment and Encoder

1. คำนวณหาจำนวนบิตที่ใช้แทนค่าแต่ละค่า

$$nB = 2\left\lceil \left(\log_2 \left\lceil \sqrt{2C + 0.25} + 0.5 \right\rceil \right) \right\rceil \quad (4)$$

2. สร้างตาราง EDBI Mapping เพื่อลดเวลาในการคำนวณในกระบวนการสร้างดัชนีและการสอบถาม

2.1. คำนวณหาค่าที่ใช้แทนค่าที่ถูกสอบถามบ่อยสุด

$$V_0 = \frac{((2^{nB/2} - 1) + 0.5)^2}{2} - 1.125 \quad (5)$$

2.2. กำหนดค่าให้กับค่าที่ถูกสอบถามบ่อยตามลำดับโดยที่

$$V_{i+1} = V_i - 1 \quad \text{for } i = 0,1,2,...,C-2 \quad (6)$$

2.3. นำค่า V_i ที่ได้มาคำนวณหาค่า r^i และ s^i

$$\text{for } i = 0,1,\ldots,\frac{nB}{2}-1 \text{ โดยใช้สมการ [9]}$$

$$r^i = \left\lfloor \sqrt{2V_i + 2.25} - 0.5 \right\rfloor \quad (7)$$

$$s^i = V_i - \frac{r^i(r^i - 1)}{2} \quad (8)$$

3. ทำการลงรหัสโดยแทนค่า r^i และ s^i ด้วยเลขฐานสองโดยใช้จำนวนบิตเท่า ๆ กันคือ $\frac{nB}{2}$ บิต

ภาพที่ : 4 ขั้นตอนวิธีการกำหนดค่าและการลงรหัส EDBI

ภาพที่ 4 แสดงขั้นตอนวิธีการกำหนดค่า (Value Assignment) และการลงรหัส (Encoder) โดยจำนวนบิต(nB) ที่ใช้แทนค่าแต่ละค่าของแอททริบิวต์จะถูกคำนวณ จากนั้นตาราง EDBI Mapping ซึ่งเก็บค่ากำหนด(V_i) ซึ่งคำนวณจากสมการที่ (5) และ (6) และค่าที่ใช้ลงรหัสบิตแมป (r^i และ s^i) ซึ่งคำนวณสมการที่ (7) และ (8) จะถูกสร้างขึ้น จากนั้นทำการลงรหัสโดยแทนค่า r^i และ s^i ด้วยเลขฐานสองโดยใช้จำนวนบิตที่เท่ากันคือ $\frac{nB}{2}$ บิต

จากตัวอย่างข้างต้นแอททริบิวต์ X ที่มีคาร์ดินอลิตี้ C=16 จากสมการ (4) และ(5) จะได้ว่า nB = 6 และ V_0=27 ตารางที่ 6 แสดงการกำหนดค่าให้กับแอททริบิวต์ X ทั้ง 16 ค่า และค่า r^i และ s^i ที่ได้จากสมการ (7) และ(8)

ตารางที่ 6: EDBI Mapping Table ของแอททริบิวต์ X

X	E	A	D	G	M	N	O	B	C	H	I	J	L	P	F	K
V_i	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12
r^i	7	7	7	7	7	7	7	6	6	6	6	6	6	5	5	5
s^i	6	5	4	3	2	1	0	5	4	3	2	1	0	4	3	2

จากนั้นให้ใช้ 3 บิตในการลงรหัส r และ s เช่นค่า F จากตาราง EDBI Mapping จะได้ r = 5 และ s = 3 สามารถลงรหัส r =101 และ s = 011 ตารางที่ 7(ข) แสดงการลงรหัส EDBI ของแอททริบิวต์ X จะเห็นว่า EDBI ใช้ 6 บิตแมปเวกเตอร์ซึ่งน้อยกว่า ดัชนีบิตแมปแบบคู่กัน (7 บิตแมปเวกเตอร์)

ตารางที่ 7: แสดงการลงรหัสแบบ EDBI

RID	X	R			S		
		r^2	r^1	r^0	s^2	s^1	s^0
1	K	1	0	1	0	1	0
2	F	1	0	1	0	1	1
3	D	1	1	1	1	0	0
4	P	1	0	1	1	0	0
5	N	1	1	1	0	0	1
6	B	1	1	0	1	0	1
7	L	1	1	0	0	0	0
8	E	1	1	1	1	1	0
9	A	1	1	1	1	0	1
10	F	1	0	1	0	1	1

(ก)แอททริบิวต์ (ข) EDBI

3.2 การสอบถามโดยใช้ EDBI

การสอบถามแบบค่าเท่ากันบน EDBI มีขั้นตอนการสอบถามดังแสดงในภาพที่ 5

1. ทำการlookup ค่า r และ s ของค่าที่ต้องการสอบถาม (V) จากตาราง EDBI Mapping
2. กำหนดให้ PATTERN คือ เลขฐานสองของค่า s ให้เรียงต่อจากค่า r
3. *if* (s = 0)
   ทำการจัดเก็บเรคอร์ดที่ตรงเงื่อนไข
   $(\cup s^0 \cup s^1 \cup s^2 \cup ... \cup s^{(nB/2)-1} = 0)$
   else
   ทำการจัดเก็บเรคอร์ดที่ตรงเงื่อนไข
   $(\cap r^i = 1)$ และ $(\cap s^j = 1)$ โดยที่ $(0 \le i, j \le \frac{nB}{2} - 1)$
   สำหรับค่าตำแหน่ง i, j ที่มีค่า 1ใน PATTERN
4. นำ PATTERN มาเทียบกับผลลัพธ์ที่ได้จากข้อ 3

ภาพที่ 5: ขั้นตอนวิธีการสอบถามบน EDBI

ตัวอย่างการสอบถามแบบค่าเท่ากันว่าบนแอทริบิวต์ X มีเรคอร์ดใดบ้างที่มีค่าเท่ากับ F ทำได้ด้วยวิธีการดังนี้

1. ค้นหา r และ s ของ F จากตาราง EDBI Mapping (ตารางที่ 6) ได้ $r = 5$ และ $s = 3$ (ดูตารางที่ 8(ข))
2. สร้าง PATTERN ได้เป็น 101011 ($r = 101$ $s = 011$)
3. $s = 3$ ดังนั้น ทำการจัดเก็บเรคอร์ดที่ตรงเงื่อนไข $((r^2 \cap r^0) = 1)$ และ $((s^1 \cap s^0) = 1)$
4. นำ PATTERN ไปเทียบกับตารางผลลัพธ์ที่ได้จากข้อ 3 คำตอบที่ได้คือแถวที่ 2 และ10 แสดงดังตารางที่ 8(ก) และ 8(ข)

ตารางที่ 8: แสดงผลลัพธ์จากการสอบถามโดยใช้EDBI

RID	X	R			S		
		r^2	r^1	r^0	s^2	s^1	s^0
1	K	1	0	1	0	1	0
2	F	1	0	1	0	1	1
3	D	1	1	1	1	0	0
4	P	1	0	1	1	0	0
5	N	1	1	1	0	0	1
6	B	1	1	0	1	0	1
7	L	1	1	0	0	0	0
8	E	1	1	1	1	1	0
9	A	1	1	1	1	0	1
10	F	1	0	1	0	1	1

(ก)แอททริบิวต์ (ข) EDBI

4. ผลการดำเนินการวิจัย

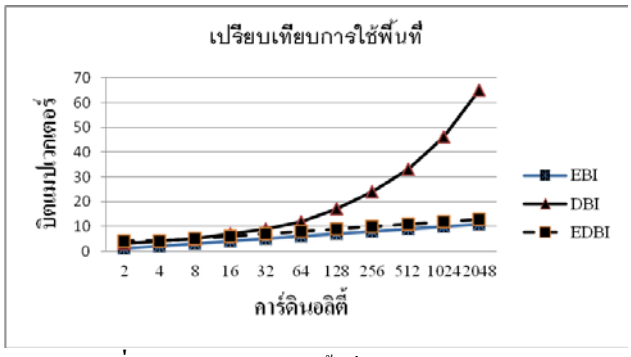
ในส่วนนี้เป็นการวิเคราะห์ประสิทธิภาพในเรื่องของการใช้พื้นที่จัดเก็บดัชนีและประสิทธิภาพในเรื่องของเวลาที่ใช้ในการสอบถาม

4.1 วิเคราะห์ประสิทธิภาพในเรื่องของพื้นที่จัดเก็บดัชนี

ตารางที่ 9 : แสดงการใช้พื้นที่ของดัชนีบิตแมป

ดัชนีบิตแมป	ตัวย่อ	พื้นที่
		จำนวนบิตแมปเวกเตอร์ที่ใช้ในการสร้างดัชนี
Simple Bitmap Index[3]	SBI	C
Interval Bitmap Index[9]	IBI	$\lceil C/2 \rceil$
Scatter Bitmap Index[4]	ScaBI	$\lceil 2\sqrt{C} \rceil$
Dual Bitmap Index[6]	DBI	$\lfloor \sqrt{2C + 0.25} + 0.5 \rfloor$
Enhancing Dual Bitmap Index	EDBI	$2\left\lceil \left(\log_2 \left\lfloor \sqrt{2C + 0.25} + 0.5 \right\rfloor \right) \right\rceil$
Encoded Bitmap Index[5]	EBI	$\lceil \log_2 C \rceil$

ตารางที่ 9 แสดงการใช้พื้นที่ของดัชนีบิตแมป 6 ชนิด พบว่า SBI ใช้พื้นที่จัดเก็บดัชนีมากที่สุด รองลงมาคือ IBI, ScaBI, DBI, EDBI และ EBI ตามลำดับ กราฟในภาพที่ 6 แสดงให้เห็นถึงประสิทธิภาพการใช้พื้นที่ของ EDBI ใกล้เคียง EBI และเหนือกว่า DBI อย่างมากสำหรับแอททริบิวต์ที่มีคาร์ดินอลิตี้สูง

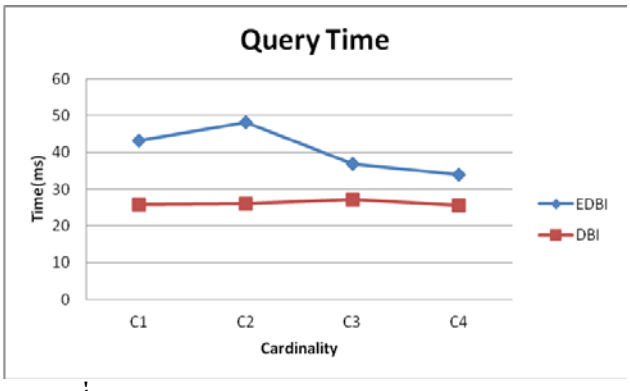


ภาพที่ 6: เปรียบเทียบการใช้พื้นที่ของดัชนีบิตแมป 3 แบบ

4.2 วิเคราะห์ประสิทธิภาพในเรื่องของเวลา

การวิเคราะห์ประสิทธิภาพในเรื่องของเวลาที่ใช้ในการสอบถามใช้ชุดข้อมูลตาราง order เลือกสดมภ์ที่มีคาร์ดินอลิตี้ C ต่าง ๆ กันดังนี้ C1=32, C2=64, C3=128 และ C4=256 จาก TPC-H [12] และทำการทดลองเปรียบเทียบเวลา 2 ดัชนีคือ DBI และ EDBI ภาพที่ 7 แสดงการเปรียบเทียบการใช้เวลาในการสอบถามของดัชนีบิตแมป DBI และ EDBI จะเห็นว่า DBI ใช้เวลาน้อยกว่า EDBI ในทุกการสอบถามทั้งนี้เป็นเพราะว่า ในแต่ละการสอบถาม DBI อ่านแค่ 2 บิตแมปเวกเตอร์ในขณะที่ EDBI อ่านมากกว่า 2 บิตแมปเวกเตอร์ จำนวนบิตแมปเวกเตอร์ที่ EDBI อ่านจะอยู่ระหว่าง 2- nB บิตแมปเวกเตอร์ ส่วน EBI จะอ่าน $\lceil \log_2 C \rceil$ บิตแมปเวกเตอร์

ถ้าพิจารณาประสิทธิภาพในแง่ของพื้นที่และเวลาพร้อมกันจะเห็นว่า EDBI มีประสิทธิภาพเหนือกว่าดัชนีบิตแมปอื่น ๆ



ภาพที่ 7: เปรียบเทียบการใช้เวลาของดัชนีบิตแมป 2 แบบ

5. สรุป

EDBI เป็นการนำดัชนีบิตแมปแบบคู่กันมาลงรหัสแบบใหม่และใช้ ข้อมูลค่าแอททริบิวต์ที่ถูกสอบถามบ่อย ๆ มาช่วยในการลงรหัสทำให้มีประสิทธิภาพกว่าการลงรหัสแบบดั้งเดิม นำไปสู่การเพิ่มขีดความสามารถของดัชนีแบบคู่กันในการใช้กับแอททริบิวต์ที่มีคาร์ดินอลิตี้สูงในแง่ของพื้นที่ที่ดีเทียบเท่าและในแง่ของเวลาที่ใช้ในการสอบถามโดยเฉลี่ยน้อยกว่าดัชนีแบบเข้ารหัส

สำหรับงานที่จะทำในอนาคตได้แก่ การนำ EDBI ไปใช้กับการลงรหัสเป็นกลุ่มสำหรับการสอบถามแบบสมาชิกเพราะ EDBI สามารถลดจำนวนการอ่านบิตแมปเวกเตอร์ได้มากกว่าดัชนีบิตแมปแบบคู่กันซึ่งต้องอ่านทีละ 2 บิตแมปเวกเตอร์สำหรับแต่ละค่าในการสอบถามแบบสมาชิก

เอกสารอ้างอิง

[1] William H. Inmon, *Building the data warehouse*., Wiley Computer Publishing: New York, 1996.

[2] Vincent Rainardi, *Building a Data Warehouse: With Examples in SQL Server*, Springer-Verlag: New York, 2008.

[3] S. Chaudhuri, and U. Dayal. "An overview of data warehousing and OLAP technology.", *ACM Sigmod record* 26.1, pp. 65-74, 1997.

[4] C.Y. Chan and Y. E. Ioannidis. "Bitmap index design and evaluation.", *ACM SIGMOD Record* 27.2, pp.355-366, 1998.

[5] C.Y. Chan and Y. E. Ioannidis. "An efficient bitmap encoding scheme for selection queries.", *ACM SIGMOD Record* 28.2, pp. 215-226, 1999.

[6] S.Vanichayobon, J.Manfuekphan, and L.Gruenwald. "Scatter Bitmap: Space-Time Efficient Bitmap Indexing for Equality and Membership Queries.", *Cybernetics and Intelligent Systems, 2006 IEEE Conference on*. IEEE, 2006.

[7] M.C. Wu and A.P. Buchmann. "Encoded bitmap indexing for data warehouses.", *Data Engineering, 1998. Proceedings., 14th International Conference on*. IEEE, 1998.

[8] A. Keawpibal, N. Wattanakitrungroj and S. Vanichayobon. "Enhanced Encoded Bitmap Index for equality query.", *Computing Technology and Information Management (ICCM), 2012 8th International Conference on*. Vol. 1. IEEE, 2012.

[9] N. Wattanakitrungroj and S. Vanichayobon. "Dual Bitmap Index: Space-Time Efficient Bitmap Index for Equality and Membership Queries.", *Communications and Information Technologies, 2006. ISCIT'06. International Symposium on*. IEEE, 2006.

[10] F. Deliège and T. B. Pedersen. "Position list word aligned hybrid: optimizing space and performance for compressed bitmaps.", *Proceedings of the 13th International Conference on Extending Database Technology*. ACM, 2010.

[11] M. Stabno and R. Wrembel. "RLH: Bitmap compression technique based on run-length and Huffman encoding.", *Information Systems* 34.4, pp.400-414, 2009.

[12] Transaction Processing Performance Council (TPC), “TPC-H: An Ad Hoc Decision Support Benchmark”, Version2.14.1, [Online]. Availablefrom: http://www.tpc.org/tpch., [2013, December 11].

ประวัติผู้เขียน

ชื่อ สกุล นางสาวศิเนตร กิ้มเส้ง

รหัสประจำตัวนักศึกษา 5510220103

วุฒิการศึกษา

วุฒิ	ชื่อสถาบัน	ปีที่สำเร็จการศึกษา
วท.บ. (วิทยาการคอมพิวเตอร์)	มหาวิทยาลัยทักษิณ	2554

การตีพิมพ์เผยแพร่ผลงาน

ศิเนตร กิ้มเส้ง และ ศิริรัตน์ วณิชโยบล. 2557. การเพิ่มประสิทธิภาพดัชนีบิตแมปแบบคู่กันโดยใช้วิธีการเข้ารหัส. การประชุมทางวิชาการระดับชาติด้านคอมพิวเตอร์และเทคโนโลยีสารสนเทศ ครั้งที่ 10 (NCCIT 2014). ภูเก็ต ประเทศไทย, 8-9 พฤษภาคม 2557