การเพิ่มประสิทธิภาพการสอบถามของดัชนีบิตแมปแบบเข้ารหัส โดยใช้เทคนิคการจัดกลุ่มข้อมูล

Improved Query Efficiency to Encoded Bitmap Index using Data Clustering

อมรเทพ แก้วภิบาล

Amorntep Keawpibal

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
วิทยาศาสตรมหาบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์
มหาวิทยาลัยสงขลานครินทร์

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Science Prince of Songkla University 2555

ชื่อวิทยานิพนธ์ การเพิ่มประสิทธิภาพการสอบถามของดัชนีบิตแมปแบบเข้ารหัสโดยใช้เทคนิคการจัดกลุ่มข้อมูล
ผู้เขียน นายอมรเทพ แก้วภิบาล
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

.....
(ศาสตราจารย์ ดร.อมรรัตน์ พงศ์ดารา)
คณบดีบัณฑิตวิทยาลัย

ชื่อวิทยานิพนธ์ การเพิ่มประสิทธิภาพการสอบถามของดัชนีบิตแมปแบบเข้ารหัสโดยใช้เทคนิคการจัดกลุ่มข้อมูล
ผู้เขียน นายอมรเทพ แก้วภิบาล
สาขาวิชา วิทยาการคอมพิวเตอร์
ปีการศึกษา 2554

บทคัดย่อ

ดัชนีบิตแมปเป็นวิธีการที่รู้จักกันดีในการเพิ่มความเร็วในการสอบถามข้อมูลแบบซับซ้อนและแบบทันทีทันใดในระบบคลังข้อมูล เนื่องจากสามารถดำเนินการตรรกะระดับบิตซึ่งเป็นการสนับสนุนการทำงานของฮาร์ดแวร์ก่อนเข้าถึงข้อมูลจริงทำให้การประมวลผลมีความรวดเร็วมากยิ่งขึ้น

ดัชนีบิตแมปแบบเข้ารหัสเป็นเทคนิคการทำดัชนีบิตแมปที่มีประสิทธิภาพมากที่สุดในด้านการใช้พื้นที่ในการจัดเก็บดัชนี เมื่อเปรียบเทียบกับดัชนีบิตแมปที่เคยมีมาได้แก่ ดัชนีบิตแมปแบบพื้นฐาน ดัชนีบิตแมปแบบแถว ดัชนีบิตแมปแบบช่วง ดัชนีบิตแมปแบบกระจาย และดัชนีบิตแมปแบบคู่กัน วิทยานิพนธ์นี้เป็นการนำเสนอขั้นตอนวิธีการเพิ่มประสิทธิภาพการสอบถามข้อมูลบนดัชนีบิตแมปแบบเข้ารหัสโดยใช้เทคนิคการจัดกลุ่มข้อมูล (Cluster-EBI) โดยใช้หลักการจัดกลุ่มค่าของแอตทริบิวต์ที่มีการสอบถามไปด้วยกันและสอบถามร่วมกัน เมื่อค่าของแอตทริบิวต์เหล่านั้นถูกสอบถามด้วยกันอีกจะสามารถลดจำนวนบิตแมปเวกเตอร์ที่ต้องตรวจสอบให้เหลือจำนวนน้อยที่สุดได้ นอกจากนี้วิทยานิพนธ์นี้นำเสนอขั้นตอนวิธีการสอบถามข้อมูลแบบค่าเท่ากันที่มีประสิทธิภาพโดยใช้การดำเนินการตรรกะต้นทุนต่ำ จากผลการวิเคราะห์และทดลองเปรียบเทียบระหว่างดัชนีบิตแมปแบบเข้ารหัสที่ใช้เทคนิคการจัดกลุ่มข้อมูลกับดัชนีบิตแมปที่เคยมีมาพบว่า ดัชนีบิตแมปแบบเข้ารหัสที่ใช้เทคนิคการจัดกลุ่มข้อมูลมีประสิทธิภาพในแง่ของการแลกเปลี่ยนระหว่างประสิทธิภาพของพื้นที่กับเวลา (Space-Time Trade-off) สำหรับการสอบถามข้อมูลแบบสมาชิกและแบบค่าเท่ากัน เมื่อแอตทริบิวต์ที่นำมาทำดัชนีมีคาร์ดินอลิตี้สูง

Thesis Title	Improved Query Efficiency to Encoded Bitmap Index using Data Clustering
Author	Mr. Amorntep Keawpibal
Major Program	Computer Science
Academic Year	2011

ABSTRACT

Bitmap Indexes are well-known method to improve query processing time for complex and interactive queries in a data warehouse. They significant improve query processing time by utilizing low-cost Boolean operations and performing predicate conditions on the index level before accessing to the primary source data.

Encoded Bitmap Index outperforms the best in term of space requirement, comparing with existing Bitmap Indexes (i.e., Simple Bitmap Index, Range Bitmap Index, Interval Bitmap Index, Scatter Bitmap Index and Dual Bitmap Index). In this thesis, we proposed optimization query method to Encoded Bitmap Index using Data Clustering (Cluster-EBI) to group attribute values that are frequently queried together before encoding, leading to reduce bitmap vectors to be scanned for membership query. Moreover, we proposed an efficient algorithm to improve equality query on Cluster-EBI by making use of low-cost Boolean operations. The comparative study shows that the performance of Cluster-EBI is better than other existing Bitmap Indexes for membership and equality queries from the point of space-time trade-off when indexed attribute values have high cardinality.

สารบัญ

หน้า

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญ (ต่อ)

หน้า

รายการตาราง

รายการภาพประกอบ

รายการภาพประกอบ (ต่อ)

ภาพประกอบ **หน้า**

รายการภาพประกอบ (ต่อ)

บทที่ 1

บทนำ

1.1 ความเป็นมา

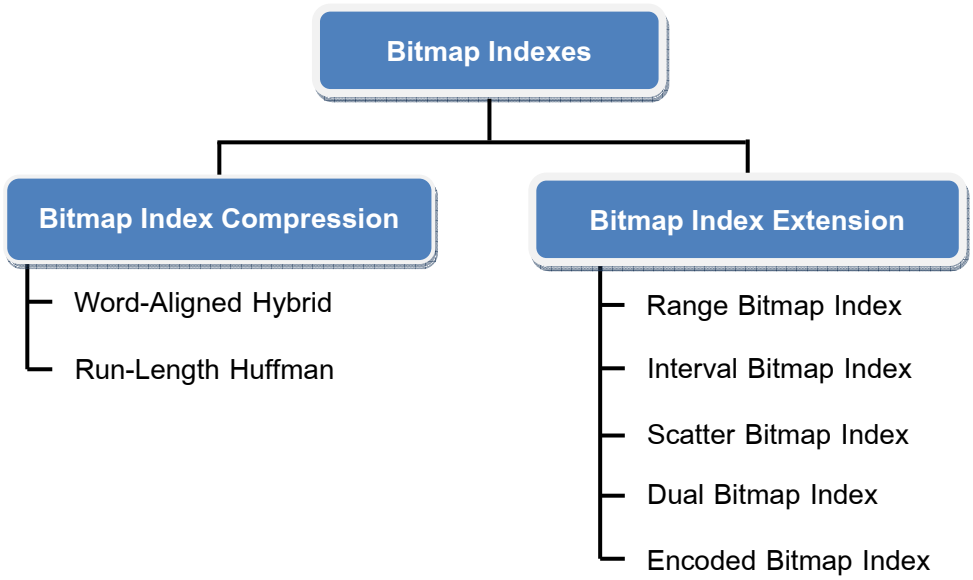
ในปัจจุบันข้อมูลในคลังข้อมูลมีความสำคัญในการใช้วิเคราะห์และสนับสนุนการตัดสินใจของผู้บริหารในองค์กรเป็นอย่างมาก ซึ่งต้องมีการเก็บรวบรวมข้อมูลตั้งแต่อดีตจนถึงปัจจุบัน การสอบถามข้อมูลในคลังข้อมูลจึงเป็นการสอบถามที่มีความซับซ้อน (Complex Query) กล่าวคือ ไม่ทราบล่วงหน้าว่าผู้ใช้จะสอบถามข้อมูลอะไรบ้างจึงต้องมีการรวบรวมข้อมูลจากหลายๆ แหล่งและเก็บข้อมูลตั้งแต่อดีตจนถึงปัจจุบัน ส่งผลให้ข้อมูลในคลังข้อมูลมีปริมาณมาก การสอบถามข้อมูลจึงต้องทำการค้นหาข้อมูลเป็นจำนวนมากทำให้ใช้เวลาในการค้นหาข้อมูลเป็นเวลานาน (Chaudhuri and Dayal, 1997; Han and Kamber, 2000; Silberschatz *et al.*, 2011)

เทคนิคหนึ่งที่สามารถลดเวลาในการค้นหาข้อมูลและเพิ่มประสิทธิภาพการประมวลผลข้อมูลโดยไม่ต้องเสียค่าใช้จ่ายใดๆ เพิ่มขึ้นคือ การทำดัชนี (O'Neil and Quass, 1997; Chan and Ioannidis, 1998; Silberschatz *et al.*, 2011) ซึ่งการทำดัชนีมีหลายวิธี เช่น B-tree, hash และดัชนีบิตแมป เป็นต้น โดยดัชนีบิตแมปเป็นเทคนิคที่มีการประมวลผลได้รวดเร็วและมีประสิทธิภาพ เพราะสามารถดำเนินการระดับบิต (AND, OR, NOT, XOR เป็นต้น) ระหว่างบิตแมปก่อนเข้าถึงข้อมูลจริงซึ่งดัชนีบิตแมปแบบพื้นฐาน (O'Neil and Quass, 1997) เหมาะกับข้อมูลที่ไม่มีการเปลี่ยนแปลง ซึ่งพบในคลังข้อมูลและเหมาะกับการทำบนแอตทริบิวต์ที่มีค่าคาร์ดินอลิตี้ต่ำ เช่น แอตทริบิวต์เพศมีค่าคาร์ดินอลิตี้เท่ากับ 2 คือ เพศชายและเพศหญิงซึ่งใช้เพียง 2 บิตแมปเวกเตอร์ แต่ถ้าทำดัชนีบิตแมปบนแอตทริบิวต์มีคาร์ดินอลิตี้สูง เช่น แอตทริบิวต์ชื่อคน เป็นต้น จำนวนบิตแมปเวกเตอร์จะเพิ่มมากขึ้น ทำให้สิ้นเปลืองเนื้อที่ในการจัดเก็บดัชนี

งานวิจัยส่วนใหญ่ (Wu and Yu, 1998; Chan and Ioannidis, 1999; Wu *et al.*, 2006; Stabno and Wrembel, 2009) มุ่งเน้นในการลดขนาดพื้นที่ที่ใช้ในการจัดเก็บดัชนีบิตแมป และลดเวลาในการสอบถามข้อมูล จากภาพประกอบ 1-1 แสดงวิธีการสำหรับการลดขนาดดัชนี ซึ่งสามารถแบ่งเป็น 2 วิธีการ ดังนี้

1) การบีบอัดสำหรับดัชนีบิตแมป (Bitmap Index Compression) เหมาะกับการบีบอัดดัชนีบิตแมปบนแอตทริบิวต์ที่มีค่าคาร์ดินอลิตี้สูง ได้แก่ การบีบอัดแบบ Word-Aligned Hybrid (WAH) (Wu *et al.*, 2006) และการบีบอัดแบบ Run-Length Huffman (RLH)

เป็นต้น ไม่สามารถสอบถามข้อมูลหรือดำเนินการตรรกะบนดัชนีได้โดยตรงจำเป็นต้องคลายดัชนีบิตแมปที่ถูกบีบอัดก่อนสอบถามข้อมูลทุกครั้ง



ภาพประกอบ 1-1 วิธีการการลดขนาดดัชนีบิตแมป

2) การขยายแนวคิดของดัชนีบิตแมป (Bitmap Index Extension) เป็นการขยายโครงสร้างของดัชนีบิตแมปเพื่อให้ใช้เนื้อที่ในการจัดเก็บน้อยลง แต่ยังคงมีประสิทธิภาพในการสอบถามข้อมูล โดยทุกๆ ค่าข้อมูลในแอตทริบิวต์ที่นำมาสร้างดัชนีจะถูกลงรหัสด้วยจำนวนบิตที่เท่ากันและการเข้าถึงข้อมูลสามารถทำได้โดยการคำนวณจากฟังก์ชัน หรือการใช้ตารางเทียบค่า (Mapping Table) (Stabno and Wrembel, 2009) ได้แก่ การทำดัชนีบิตแมปแบบแถว (Range Bitmap Index) (Wu and Yu, 1998) การทำดัชนีบิตแมปแบบแบ่งช่วง (Interval Bitmap Index) (Chan and Ioannidis, 1999) การทำดัชนีบิตแมปแบบกระจาย (Scatter Bitmap Index) (Vanichayobon *et al.*, 2006) การทำดัชนีบิตแมปแบบคู่กัน (Dual Bitmap Index) (Wattanakitrungroj and Vanichayobon, 2006) และการทำดัชนีบิตแมปแบบเข้ารหัส (Encoded Bitmap Index) (Wu and Buchmann, 1998) เป็นต้น

ดัชนีบิตแมปแบบเข้ารหัสเป็นดัชนีบิตแมปที่ใช้พื้นที่ในการจัดเก็บดัชนีน้อยที่สุด นั่นคือ ใช้ $\lceil log_2 C \rceil$ บิตแมปเวกเตอร์ เมื่อ C คือ ค่าคาร์ดินอลิตี้ การสอบถามข้อมูลแบบค่าเท่ากันของดัชนีบิตแมปต้องตรวจสอบบิตทุกบิตแมปเวกเตอร์ทำให้ใช้เวลาในการสอบถามนาน หากมีวิธีการที่สามารถตรวจสอบบิตทุกบิตแมปเวกเตอร์ได้รวดเร็วขึ้นจะทำให้เวลาในการสอบถามเร็วขึ้นด้วย และการสอบถามข้อมูลแบบสมาชิกสามารถทำได้โดยการหาคำตอบของแต่ละค่าจากนั้นนำมาดำเนินการตรรกะ OR ทำให้ใช้เวลาในการสอบถามข้อมูลเพิ่มขึ้น งานวิจัย

(Weahama *et al.*, 2009) ได้นำเสนอการเพิ่มประสิทธิภาพการสอบถามข้อมูลแบบสมาชิกของดัชนีบิตแมปแบบกระจายด้วยการใช้เทคนิคการจัดกลุ่มข้อมูลและแสดงให้เห็นว่าการใช้เทคนิคการจัดกลุ่มข้อมูลสามารถเพิ่มประสิทธิภาพในการสอบถามข้อมูลแบบสมาชิกของดัชนีบิตแมปแบบกระจาย และงานวิจัย (Sainui *et al.*, 2008; Alam *et al.*, 2008) ได้นำเสนอการเพิ่มประสิทธิภาพการสอบถามข้อมูลแบบสมาชิกของดัชนีบิตแมปแบบเข้ารหัสด้วยการใช้เทคนิคการหาข้อมูลที่ปรากฏด้วยกันบ่อย ซึ่งแสดงให้เห็นว่าการเข้ารหัสค่าข้อมูลที่เหมาะสมสามารถลดจำนวนบิตแมปเวกเตอร์ที่ต้องตรวจสอบได้ทำให้มีประสิทธิภาพในการสอบถามแบบสมาชิกบนดัชนีบิตแมปแบบเข้ารหัส

งานวิจัยนี้จึงนำเสนอเทคนิคในการเพิ่มประสิทธิภาพในการสอบถามข้อมูลแบบสมาชิกและการสอบถามข้อมูลแบบค่าเท่ากันบนดัชนีบิตแมปแบบเข้ารหัสโดยการนำเทคนิคการทำเหมืองข้อมูล เช่น การจัดกลุ่มข้อมูล (Data Clustering) มาช่วยในการจัดกลุ่มค่าของแอตทริบิวต์ที่สอบถามไปด้วยกันและสอบถามร่วมกันเพื่อช่วยในการลงรหัสค่าแต่ละค่าของแอตทริบิวต์ซึ่งการสร้างรูปแบบการลงรหัส (Encoded Scheme) มีความสำคัญอย่างมากในการสอบถามข้อมูล เพราะรูปแบบการเข้ารหัสที่เหมาะสมจะทำให้สามารถลดจำนวนบิตแมปเวกเตอร์ที่สามารถตรวจสอบเมื่อมีการสอบถามแบบสมาชิกส่งผลให้มีประสิทธิภาพในการสอบถามแบบสมาชิกบนดัชนีบิตแมปแบบเข้ารหัส

1.2 งานวิจัยที่เกี่ยวข้อง

An Overview of Data Warehousing and OLAP Technology

งานวิจัยนี้ (Chaudhuri and Dayal, 1997) กล่าวถึง ความสำคัญของคลังข้อมูล สถาปัตยกรรมของคลังข้อมูล การออกแบบโครงสร้างของคลังข้อมูล ความแตกต่างระหว่างฐานข้อมูลปฏิบัติการกับคลังข้อมูล ขั้นตอนการสร้างคลังข้อมูล เครื่องมือที่ใช้ในการสอบถามและวิเคราะห์ข้อมูล การเพิ่มประสิทธิภาพในการสอบถามข้อมูล และการจัดการคลังข้อมูล

Bitmap Index Design and Evaluation

งานวิจัยนี้ (Chan and Ioannidis, 1998) กล่าวถึง ความสำคัญของการทำดัชนีบิตแมป การทำดัชนีบิตแมปแบบพื้นฐาน (Simple Bitmap Index) ซึ่งใช้จำนวนบิตแมปเวกเตอร์เท่ากับค่าคาร์ดินอลิตี้ของแอตทริบิวต์ที่นำมาทำดัชนีบิตแมป ดังนั้นการทำดัชนีบิตแมปแบบพื้นฐานจึงเหมาะกับการทำบนแอตทริบิวต์ที่มีคาร์ดินอลิตี้ต่ำ แต่เมื่อแอตทริบิวต์มีคาร์ดินอลิตี้เพิ่มขึ้นจะทำให้พื้นที่ที่ใช้ในการจัดเก็บดัชนีมากขึ้น ผู้วิจัยได้นำเสนอขั้นตอนการสร้างดัชนีบิตแมปสำหรับการสอบถามแบบค่าเท่ากันและการสอบถามแบบช่วงที่ใช้พื้นที่ในการจัดเก็บดัชนีน้อยกว่าดัชนีบิตแมปแบบพื้นฐาน และการแลกเปลี่ยนระหว่างพื้นที่ที่ใช้จัดเก็บดัชนีและเวลาในการสอบถามข้อมูล

Using Data Clustering to Optimize Scatter Bitmap Index for Membership Queries

งานวิจัยนี้ (Weahama *et al.*, 2009) กล่าวถึงการทำดัชนีบิตแมปแบบกระจาย (Scatter Bitmap Index) ซึ่งใช้จำนวนบิตแมปเวกเตอร์ $\lceil 2\sqrt{C} \rceil$ บิตแมปเวกเตอร์และการเพิ่มประสิทธิภาพการสอบถามแบบสมาชิกบนดัชนีบิตแมปแบบกระจาย โดยใช้เทคนิคการแบ่งกลุ่มข้อมูล นั่นคือเทคนิค k-modes ซึ่งแสดงให้เห็นว่าการทำดัชนีบิตแมปแบบกระจายโดยใช้เทคนิคการแบ่งกลุ่มข้อมูลใช้เวลาในการสอบถามแบบสมาชิกน้อยกว่าการทำดัชนีบิตแมปแบบกระจายทั่วไป เพราะการจัดกลุ่มทำให้ข้อมูลที่ถูกสอบถามด้วยกันถูกจัดให้อยู่ในกลุ่มเดียวกันเพื่อลดจำนวนบิตแมปเวกเตอร์ในการตรวจสอบ

Encoded Bitmap Indexing for Data Warehouses

งานวิจัยนี้ (Wu and Buchmann, 1998) กล่าวถึงปัญหาในการจัดเก็บดัชนีบิตแมปแบบต่างๆ ในคลังข้อมูล และนำเสนอการทำดัชนีบิตแมปแบบเข้ารหัส (Encoded Bitmap Index) ที่ใช้พื้นที่ในการจัดเก็บดัชนีที่น้อยกว่าดัชนีบิตแมปแบบอื่นๆ นั่นคือใช้จำนวนบิตแมปเวกเตอร์เพียง $\lceil log_2 C \rceil$ บิตแมปเวกเตอร์ เมื่อ C คือค่าคาร์ดินอลิตี้ของแอตทริบิวต์ที่นำมาทำดัชนี จึงสามารถสร้างดัชนีบิตแมปบนแอตทริบิวต์ที่มีคาร์ดินอลิตี้สูงได้ แต่การสอบถามข้อมูลแบบค่าเท่ากันบนดัชนีบิตแมปแบบเข้ารหัสใช้เวลาในการสอบถามมากกว่าดัชนีบิตแมปแบบอื่นๆ เนื่องจากต้องตรวจสอบทุกบิตในแต่ละบิตแมปเวกเตอร์ และต้องมีการเทียบรูปแบบการเข้ารหัสในตารางเทียบค่า

Optimizing Encoded Bitmap Index using Frequent Itemsets Mining

งานวิจัยนี้ (Sainui *et al.*, 2008) กล่าวถึงการทำดัชนีบิตแมปแบบเข้ารหัสเทคนิคกฎความสัมพันธ์ (Association rule) เทคนิคการหากลุ่มข้อมูลที่ปรากฏด้วยกันบ่อย (Frequent Itemsets Mining) และการเพิ่มประสิทธิภาพการสอบถามแบบสมาชิกบนดัชนีบิตแมปแบบเข้ารหัสโดยใช้เทคนิคการกลุ่มข้อมูลที่ปรากฏด้วยกันบ่อย และแสดงให้เห็นว่าการเข้ารหัสที่เหมาะสมทำให้ลดจำนวนบิตแมปเวกเตอร์ที่ต้องตรวจสอบได้ จึงเป็นการเพิ่มประสิทธิภาพในการสอบถามข้อมูลแบบสมาชิกบนดัชนีบิตแมปแบบเข้ารหัส

1.3 วัตถุประสงค์

1. เพื่อคิดค้นเทคนิคการเพิ่มประสิทธิภาพดัชนีบิตแมปแบบเข้ารหัสสำหรับการสอบถามข้อมูลแบบสมาชิกและแบบค่าเท่ากันด้วยเทคนิคการจัดกลุ่มข้อมูล

2. เพื่อเปรียบเทียบประสิทธิภาพของการสอบถามข้อมูลแบบค่าเท่ากันและแบบสมาชิกบนดัชนีบิตแมปแบบเข้ารหัสที่ได้ออกแบบไว้กับดัชนีบิตแมปแบบอื่นๆ ที่เคยมีมา

1.4 วิธีการดำเนินการวิจัย

1. ศึกษาแนวคิดของคลังข้อมูล วิธีการบีบอัดดัชนีบิตแมป และดัชนีบิตแมปทั้ง 6 ชนิด ได้แก่ ดัชนีบิตแมปแบบพื้นฐาน ดัชนีบิตแมปแบบแถว ดัชนีบิตแมปแบบช่วง ดัชนีบิตแมปแบบกระจาย ดัชนีบิตแมปแบบคู่กัน และดัชนีบิตแมปแบบเข้ารหัส

2. วิเคราะห์และออกแบบเทคนิคการเพิ่มประสิทธิภาพการสอบถามข้อมูลแบบค่าเท่ากันและแบบสมาชิกบนดัชนีบิตแมปแบบเข้ารหัสโดยใช้เทคนิคการจัดกลุ่มข้อมูล

3. กำหนดรูปแบบการประเมินประสิทธิภาพของดัชนีบิตแมปแบบเข้ารหัสที่ใช้เทคนิคการจัดกลุ่มข้อมูล เปรียบเทียบกับดัชนีบิตแมปทั้ง 6 ชนิด ได้แก่ ดัชนีบิตแมปแบบพื้นฐาน ดัชนีบิตแมปแบบแถว ดัชนีบิตแมปแบบช่วง ดัชนีบิตแมปแบบกระจาย ดัชนีบิตแมปแบบคู่กัน และดัชนีบิตแมปแบบเข้ารหัส โดยทำการประเมินทั้งในด้านพื้นที่ที่ใช้ในการจัดเก็บดัชนี และเวลาที่ใช้ในการสอบถามข้อมูลแบบค่าเท่ากันและแบบสมาชิก ดังขั้นตอนต่อไปนี้

3.1 ศึกษาข้อมูลที่ใช้ในการทดลองซึ่งเป็นข้อมูลมาตรฐาน TPC-H Benchmark (Transaction Processing Performance Council, 2011)

3.2 จัดเตรียมข้อมูลทดสอบ โดยเลือกเฉพาะแอตทริบิวต์ที่นำมาทำดัชนีและเปลี่ยนค่าข้อมูลเป็นจำนวนเต็มที่เรียงต่อเนื่องกัน

3.3 ออกแบบขั้นตอนวิธีในการสร้างดัชนีบิตแมปทั้ง 7 ชนิด ได้แก่ ดัชนีบิตแมปแบบพื้นฐาน ดัชนีบิตแมปแบบแถว ดัชนีบิตแมปแบบช่วง ดัชนีบิตแมปแบบกระจาย ดัชนีบิตแมปแบบคู่กัน ดัชนีบิตแมปแบบเข้ารหัส และดัชนีบิตแมปแบบเข้ารหัสที่ใช้เทคนิคการจัดกลุ่มข้อมูล

3.4 พัฒนาโปรแกรมเพื่อสร้างดัชนีบิตแมปทั้ง 7 ชนิดตามขั้นตอนที่ได้ออกแบบไว้ในข้อ 3.3 โดยใช้ตัวแปลภาษาซี (C Compiler)

4. ออกแบบขั้นตอนวิธีในการสอบถามข้อมูลแบบสมาชิกและแบบค่าเท่ากันบนดัชนีบิตแมปทั้ง 7 ชนิด ได้แก่ ดัชนีบิตแมปแบบพื้นฐาน ดัชนีบิตแมปแบบแถว ดัชนีบิตแมปแบบช่วง ดัชนีบิตแมปแบบกระจาย ดัชนีบิตแมปแบบคู่กัน ดัชนีบิตแมปแบบเข้ารหัส และดัชนีบิตแมปแบบเข้ารหัสที่ใช้เทคนิคการจัดกลุ่มข้อมูล

5. พัฒนาโปรแกรมเพื่อสอบถามข้อมูลแบบสมาชิกและแบบค่าเท่ากันบนดัชนีบิตแมปทั้ง 7 ชนิดตามขั้นตอนที่ได้ออกแบบไว้ในข้อ 4 โดยใช้ตัวแปลภาษาซี

6. ประเมินประสิทธิภาพการใช้พื้นที่ในการจัดเก็บดัชนี (Space Requirement) ของดัชนีบิตแมปทั้ง 7 ชนิด

7. ประเมินประสิทธิภาพการใช้เวลาในการสอบถามข้อมูลแบบสมาชิกและแบบค่าเท่ากัน (Time) บนดัชนีบิตแมปทั้ง 7 ชนิด ด้วยการรันโปรแกรมที่ได้พัฒนาขึ้นในข้อ 5 และบันทึกเวลาที่ใช้ในการสอบถามข้อมูลของดัชนีบิตแมปทั้ง 7 ชนิด

8. ประเมินประสิทธิภาพการแลกเปลี่ยนระหว่างพื้นที่ที่ใช้จัดเก็บดัชนีกับเวลา (Space-Time Trade off) สำหรับการสอบถามแบบสมาชิกและแบบค่าเท่ากันบนดัชนีบิตแมปทั้ง 7 ชนิด

9. วิเคราะห์และสรุปผลการประเมินประสิทธิภาพของดัชนีแบบเข้ารหัสที่ใช้เทคนิคการจัดกลุ่มข้อมูลเปรียบเทียบกับดัชนีบิตแมปทั้ง 6 ชนิด

1.5 ขอบเขตการวิจัย

1. ศึกษาวิธีการบีบอัดดัชนีบิตแมปแบบพื้นฐาน ได้แก่ เทคนิค Word-Aligned Hybrid (WAH) และเทคนิค Run-Length Huffman (RLH)

2. ศึกษาดัชนีบิตแมปที่มีอยู่ ได้แก่ ดัชนีบิตแมปแบบพื้นฐาน ดัชนีบิตแมปแบบแถว ดัชนีบิตแมปแบบแบ่งช่วง ดัชนีบิตแมปแบบกระจาย ดัชนีบิตแมปแบบคู่กัน และดัชนีบิตแมปแบบเข้ารหัส

3. วิเคราะห์และออกแบบเทคนิคการเพิ่มประสิทธิภาพดัชนีบิตแมปแบบเข้ารหัสสำหรับการสอบถามแบบสมาชิกและการสอบถามแบบค่าเท่ากัน

1.6 ขั้นตอนการดำเนินงาน

1. ศึกษาเอกสารและงานวิจัยที่เกี่ยวข้อง
2. ศึกษาและวิเคราะห์เครื่องมือสำหรับงานวิจัย
3. กำหนดขอบเขตของปัญหาในการทำวิจัย
4. วิเคราะห์และออกแบบเทคนิคการเพิ่มประสิทธิภาพดัชนีบิตแมปแบบเข้ารหัส
5. พัฒนาดัชนีบิตแมปแบบเข้ารหัสตามที่ได้ออกแบบไว้
6. ประเมินประสิทธิภาพดัชนีบิตแมปแบบเข้ารหัส
7. สรุปและวิเคราะห์ผลการประเมินประสิทธิภาพกับดัชนีเปรียบเทียบ
8. จัดทำเอกสารประกอบการวิจัย

1.7 ระยะเวลาดำเนินงานและแผนการดำเนินงาน

มกราคม 2554 – มีนาคม 2555

ตาราง 1-1 แผนการดำเนินงาน

ขั้นตอนการดำเนินงาน	เดือน														
	2554												2555		
	1	2	3	4	5	6	7	8	9	10	11	12	1	2	3
1. ศึกษาเอกสารและงานวิจัยที่เกี่ยวข้อง															
2. ศึกษาและวิเคราะห์เครื่องมือสำหรับงานวิจัย															
3. กำหนดขอบเขตของปัญหา															
4. วิเคราะห์และออกแบบเทคนิค															
5. พัฒนาดัชนีบิตแมปแบบเข้ารหัสตามที่ได้ออกแบบไว้															
6. ประเมินประสิทธิภาพดัชนีบิตแมปแบบเข้ารหัส															
7. สรุปและวิเคราะห์ผลการประเมินประสิทธิภาพ															
8. จัดทำเอกสารประกอบการวิจัย															

1.8 เครื่องมือและอุปกรณ์

ด้านฮาร์ดแวร์

1. เครื่องคอมพิวเตอร์จำนวน 1 เครื่อง รุ่น Intel(R) Core(TM)2 Duo
2. หน่วยประมวลกลาง 2.00 GHz
3. หน่วยความจำหลัก 3 GB
4. ฮาร์ดดิสก์ 250 GB

ด้านซอฟต์แวร์

1. ระบบปฏิบัติการ Windows 7
2. ตัวแปลภาษาซี (C Compiler) สำหรับใช้ในการพัฒนาโปรแกรมเพื่อวัดประสิทธิภาพของเวลาที่ใช้ในการสอบถามข้อมูล

1.9 ประโยชน์ที่คาดว่าจะได้รับ

ได้เทคนิคการเพิ่มประสิทธิภาพดัชนีบิตแมปแบบเข้ารหัสที่มีประสิทธิภาพสำหรับการสอบถามข้อมูลแบบสมาชิกและแบบค่าเท่ากัน

บทที่ 2

ทฤษฎีที่เกี่ยวข้อง

สำหรับบทนี้ กล่าวถึง คลังข้อมูล (Data Warehouse) การสร้างดัชนี (Indexing) เทคนิคการทำดัชนีบิตแมป (Bitmap Indexing) และการจัดกลุ่มข้อมูล (Data Clustering)

2.1 คลังข้อมูล (Data Warehouse)

2.1.1 ความหมายของคลังข้อมูล

คลังข้อมูล (Chaudhuri and Dayal, 1997; Han and Kamber, 2000; Silberschatz *et al.*, 2011) หมายถึงฐานข้อมูลขนาดใหญ่ขององค์กรหรือหน่วยงานหนึ่งๆ ซึ่งเก็บรวบรวมข้อมูลมาจากฐานข้อมูลปฏิบัติการภายในองค์กร (Operational Database) และแหล่งข้อมูลภายนอกองค์กร (External Sources) รวมทั้งจากไฟล์ข้อมูลทั่วไป โดยข้อมูลที่ถูกจัดเก็บในคลังข้อมูลนั้นมีวัตถุประสงค์ในการนำมาใช้งานและมีลักษณะของการจัดเก็บแตกต่างไปจากข้อมูลในฐานข้อมูลระบบอื่น และสามารถเก็บข้อมูลย้อนหลังได้หลายๆ ปี เพื่อใช้เป็นข้อมูลในการตัดสินใจหรือใช้ในการวิเคราะห์ข้อมูลที่ถูกต้องและมีประสิทธิภาพ ข้อมูลในคลังข้อมูลจะถูกนำมาใช้เพื่อสนับสนุนการตัดสินใจบริหารงานของผู้บริหาร โดยเฉพาะการเป็นข้อมูลพื้นฐานให้กับระบบงานเพื่อการบริหารงานอื่น เช่น ระบบ DSS (Decision Support System) และระบบ CRM (Customer Relationship Management) เป็นต้น

2.1.2 คุณลักษณะของคลังข้อมูล

จากความหมายของคลังข้อมูลที่บอกถึงความแตกต่างระหว่างคลังข้อมูลกับฐานข้อมูลปฏิบัติการ ซึ่งสามารถสรุปคุณลักษณะของคลังข้อมูลได้ดังนี้ (Inmon, 2005; Kimball and Ross, 2002; Han and Kamber, 2000; Silberschatz *et al.*, 2011)

- **Subject Oriented** หมายถึง คลังข้อมูลถูกออกแบบเพื่อมุ่งเน้นไปในเนื้อหาที่สนใจ เช่น ลูกค้า สินค้า ยอดการขาย เป็นต้น ไม่ได้เน้นไปที่การทำงานหรือกระบวนการแต่ละอย่างโดยเฉพาะเหมือนอย่างฐานข้อมูลปฏิบัติการ สำหรับรายละเอียดของข้อมูลที่จัดเก็บในระบบทั้งสองแบบจะแตกต่างกันไปตามความต้องการการใช้งาน คลังข้อมูลจะไม่เก็บข้อมูลที่ไม่มีส่วนเกี่ยวข้องกับการประมวลผลเพื่อสนับสนุนการตัดสินใจ การวิเคราะห์ การทำรายงาน และการทำเหมืองข้อมูล (Data Mining)

- **Integrated** หมายถึง การรวบรวมข้อมูลจากแหล่งข้อมูลหลายๆ แหล่งเข้าด้วยกันซึ่งข้อมูลอาจมาจากระบบที่แตกต่างกัน รวมทั้งอาจมีการจัดเก็บข้อมูลในรูปแบบที่แตกต่างกันหรืออาจมีความแตกต่างทางด้านแพล็ตฟอร์ม (Platform) ดังนั้นจึงต้องทำให้ข้อมูลมีความสอดคล้องและมีมาตรฐานเดียวกัน เช่น กำหนดให้ข้อมูลเดียวกันมีรูปแบบเดียวกันทั้งหมดและมีค่าอยู่ในช่วงเดียวกัน เป็นต้น

- **Time Variant** หมายถึง ข้อมูลในคลังข้อมูลต้องจัดเก็บโดยกำหนดช่วงเวลาเอาไว้เพราะการตัดสินใจด้านการบริหารจำเป็นต้องมีข้อมูลเปรียบเทียบในแต่ละช่วงเวลา โดยข้อมูลที่เก็บในคลังข้อมูลจะเป็นข้อมูลตั้งแต่อดีตจนถึงปัจจุบัน ย้อนหลัง 5-10 ปี

- **Nonvolatile** หมายถึง ข้อมูลในคลังข้อมูลจะไม่มีการเปลี่ยนแปลง โดยจะมีการเข้าถึงข้อมูล 2 แบบคือ การเพิ่มข้อมูลและการสอบถามข้อมูลเท่านั้น

2.1.3 ความแตกต่างระหว่างฐานข้อมูลปฏิบัติการและคลังข้อมูล

ระบบฐานข้อมูลปฏิบัติการจะทำงานประมวลผลทรานแซคชัน และประมวลผลแบบออนไลน์เรียกระบบนี้ว่า On-Line Transaction Processing (OLTP) เป็นระบบที่ครอบคลุมการดำเนินการประจำวันขององค์กร เช่น การลงทะเบียน การจัดซื้อ การขาย การฝากเงิน การถอนเงิน เป็นต้น ส่วนระบบคลังข้อมูลจะทำการวิเคราะห์ข้อมูลและสนับสนุนการตัดสินใจของผู้บริหาร เรียกระบบนี้ว่า On-Line Analytical Processing (OLAP) การเปรียบเทียบความแตกต่างระหว่างระบบฐานข้อมูลปฏิบัติการและคลังข้อมูลแบ่งตามลักษณะต่างๆ ได้ดังตารางที่ 2-1

ตาราง 2-1 การเปรียบเทียบฐานข้อมูลปฏิบัติการและคลังข้อมูล

ลักษณะ	ฐานข้อมูลปฏิบัติการ	คลังข้อมูล
ผู้ใช้	- เสมียน นักธุรกิจ ผู้เชี่ยวชาญด้านฐานข้อมูล	- คนที่ทำงานเกี่ยวกับความรู้ เช่น ผู้บริหาร นักวิเคราะห์
การประมวลผล	- แบบ OLTP	- แบบ OLAP
การกำหนดทิศทาง	- มุ่งเน้นลูกค้าเป็นสำคัญ	- มุ่งเน้นการตลาด - หัวเรื่องในการวิเคราะห์เป็นสำคัญ
ข้อมูล	- เป็นข้อมูลปัจจุบัน - ข้อมูลมีการเปลี่ยนแปลงได้ตลอดเวลา - มุ่งประเด็นนำข้อมูลดิบเข้า	- เป็นข้อมูลในอดีตถึงปัจจุบันที่มีความถูกต้องแม่นยำในช่วงเวลานานๆ - ข้อมูลไม่มีการเปลี่ยนแปลง และมุ่งประเด็นนำความรู้ออก

ลักษณะ	ฐานข้อมูลปฏิบัติการ	คลังข้อมูล
การสรุปความ	- ง่ายๆ ไม่ซับซ้อน - ต้องการรายละเอียดมาก - มีความถูกต้องแม่นยำในขณะมีการเข้าถึง	- มีการสรุปข้อมูลในอดีตไว้ให้ใช้ - มีความมั่นคง - สามารถแสดงผลข้อมูลในอดีตได้อย่างรวดเร็ว
มุมมอง	- แสดงรายละเอียดโดยที่โครงสร้างข้อมูลไม่มีการเปลี่ยนแปลง แต่ข้อมูลที่อยู่ข้างในสามารถเปลี่ยนแปลงได้	- มีการสรุปไว้ในลักษณะหลายมิติ ซึ่งโครงสร้างมีความยืดหยุ่นได้
การสอบถาม	- เป็นการสอบถามแบบสั้นๆ ง่ายๆ	- เป็นการสอบถามที่ซับซ้อน
ขนาดของฐานข้อมูล	- 100 MB ถึง GB	- 100 GB ถึง 1 TB
การเข้าถึง	- อ่าน / เขียน - มีจำนวนแถวที่ถูกเข้าถึงเป็นหลักสิบ - มีจำนวนผู้เข้าถึงข้อมูลเป็นหลักพัน	- มีการอ่านเป็นส่วนใหญ่ - มีจำนวนแถวที่ถูกเข้าถึงเป็นหลักล้าน - มีจำนวนผู้เข้าถึงข้อมูลเป็นหลักร้อย
การวัดประสิทธิภาพ	- ปริมาณทรานแซคชัน	- ปริมาณการสอบถาม - เน้นเวลาในการตอบสนอง

จากตารางที่ 2-1 จะเห็นได้ว่า ฐานข้อมูลปฏิบัติการและคลังข้อมูลมีความแตกต่างกันหลายประการ สาเหตุหลักที่ต้องแยกสองระบบนี้ออกจากกันคือ เพื่อเพิ่มความสามารถในการทำงานของทั้งสองระบบ เนื่องจากความแตกต่างทางด้านโครงสร้าง ซึ่งหากเรามีการดำเนินการวิเคราะห์ข้อมูลแบบออนไลน์โดยตรงบนฐานข้อมูล พร้อมกับการดำเนินการประมวลผลทรานแซคชันบนฐานข้อมูลปฏิบัติการแล้ว จะก่อให้เกิดความเสียหาย และเป็นการลดประสิทธิภาพของทั้งระบบ OLTP และ OLAP ด้วย

2.1.4 สถาปัตยกรรมของคลังข้อมูล

สถาปัตยกรรมของคลังข้อมูล (Data Warehouse Architecture: DWA) เป็นโครงสร้างมาตรฐานที่ใช้กันแพร่หลาย โดยทั่วไปแล้วคลังข้อมูลแต่ละระบบมีรูปแบบที่ไม่เหมือนกันได้ เพื่อให้เหมาะสมกับองค์กรนั้นๆ ทั้งนี้ส่วนประกอบต่างๆ ภายในสถาปัตยกรรมคลังข้อมูลที่สำคัญ แสดงดังภาพประกอบ 2-1 ได้แก่ แหล่งข้อมูล หน่วยเก็บข้อมูล OLAP Server และเครื่องมือสำหรับผู้ใช้ ดังนี้

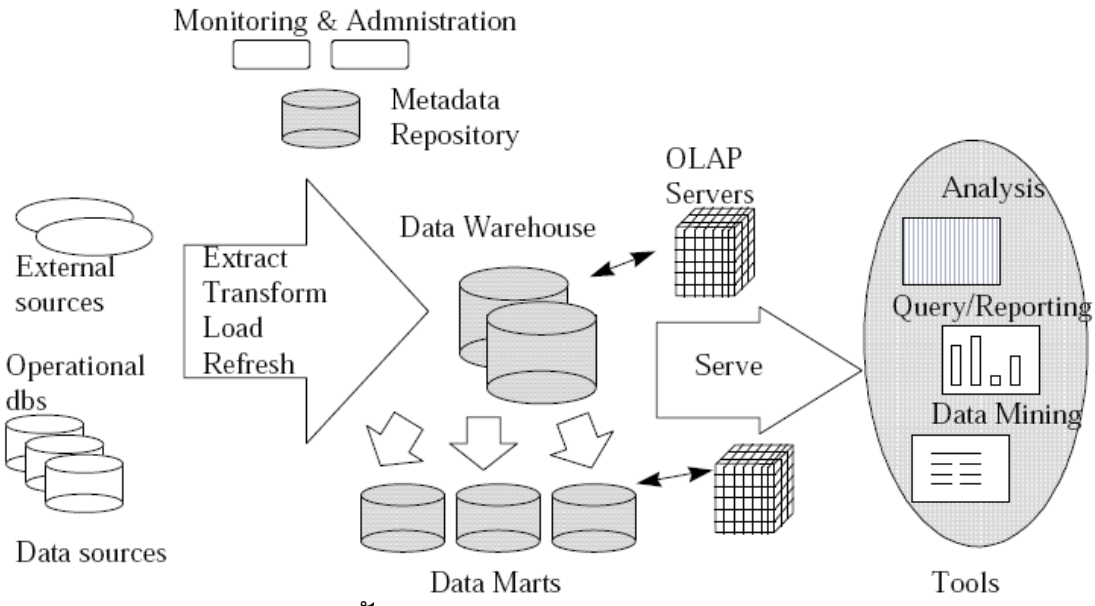
แหล่งข้อมูล (Data Source) โดยทั่วไปแหล่งข้อมูลที่นำมาใช้ในการสร้างคลังข้อมูลมาจากฐานข้อมูลปฏิบัติการ (Operational Database) และแหล่งข้อมูลภายนอก (External Sources) รวมถึงไฟล์ข้อมูลทั่วไป ซึ่งอาจมีรูปแบบที่แตกต่างกัน

หน่วยเก็บข้อมูล (Data Storage) ประกอบด้วย คลังข้อมูล (Data Warehouse) คลังข้อมูลย่อย (Data Mart) และหน่วยจัดการความรู้ (Metadata Repository) ซึ่งหน่วยจัดการความรู้มีหน้าที่จัดเก็บข้อมูลเกี่ยวกับเครื่องมือที่ใช้ในการควบคุมคลังข้อมูล เช่น ที่มาของข้อมูล รูปแบบของข้อมูล เวลาที่ปรับปรุงข้อมูลล่าสุด ข้อจำกัดของข้อมูล เป็นต้น

OLAP Server ทำหน้าที่ในการจัดเตรียมการเข้าถึงข้อมูลเพื่อให้การประมวลผลมีความรวดเร็วมากขึ้น ได้แก่ การทำดัชนี การสร้าง Materialize View การเปลี่ยนรูปแบบการสอบถามที่ซับซ้อน และการประมวลผลแบบขนานเพื่อลดเวลาในการสอบถาม

เครื่องมือสำหรับผู้ใช้งาน (Front-End Tools) ประกอบด้วยเครื่องมือในการวิเคราะห์ข้อมูล (Analysis) เครื่องมือสำหรับการสอบถาม (Query) เครื่องมือสำหรับออกรายงาน (Report) และเครื่องมือสำหรับการทำเหมืองข้อมูล (Data Mining)

ขั้นตอนในการสร้างคลังข้อมูล เริ่มจากการดึงข้อมูล (Extraction) จากแหล่งข้อมูลต่างๆ จากนั้นเป็นการทำความสะอาดข้อมูล (Data Cleaning) และเปลี่ยนรูปแบบข้อมูล (Transforming) ให้มีความสอดคล้องกัน เนื่องจากคลังข้อมูลมีการรวบรวมข้อมูลจากหลายๆ แหล่งที่แตกต่างกัน ทำให้ข้อมูลมีโครงสร้างและรูปแบบที่แตกต่างกัน ดังนั้นจึงต้องมีการทำความสะอาดข้อมูลและเปลี่ยนข้อมูลให้อยู่ในรูปแบบที่ตรงกัน จากนั้นเป็นการโหลดข้อมูล (Load) เข้าสู่คลังข้อมูล



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

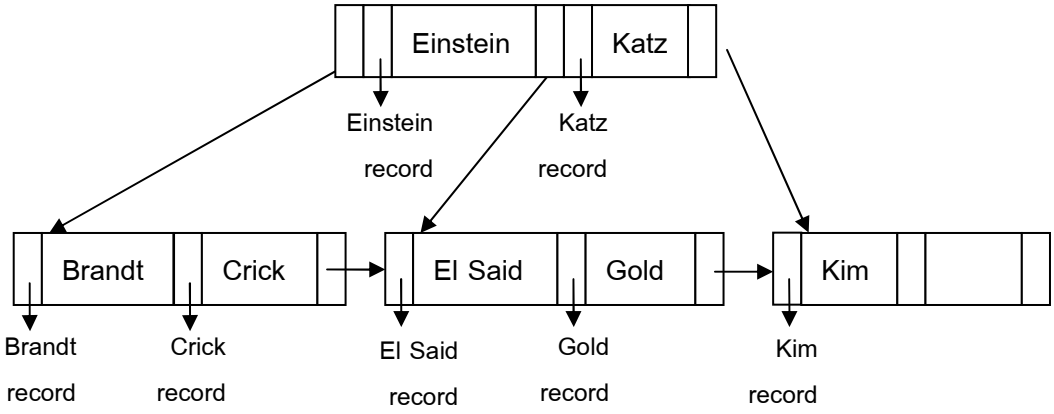
2.2 การสร้างดัชนี

การสร้างดัชนีเป็นเทคนิคหนึ่งในการเพิ่มความเร็วในการค้นหาข้อมูล โดยที่ไม่ต้องเสียค่าใช้จ่ายใดๆ ในการเพิ่มฮาร์ดแวร์ โดยดัชนีจะถูกเก็บไว้ในไฟล์ประกอบด้วยคีย์ (Key) และที่อยู่ (Address) ที่ทำให้สามารถเข้าถึงข้อมูลที่ต้องการได้โดยตรง ขนาดของดัชนีจะเล็กกว่าข้อมูลจริงที่เก็บไว้มาก จึงเป็นการลดการเข้าถึง I/O ได้มาก เพราะถ้าไม่มีดัชนี การเข้าถึงข้อมูลจะต้องอ่านข้อมูลในตารางทั้งหมด เพื่อให้ได้ข้อมูลที่ต้องการ ซึ่งเป็นวิธีที่ช้าและไม่มีประสิทธิภาพ การทำดัชนีมีหลายวิธี เช่น ดัชนีแบบ B-tree และดัชนีแบบบิตแมป เป็นต้น (Silberschatz *et al.*, 2011; Bontempo and Saracco, 2011)

2.2.1 ดัชนีแบบ B-tree

ดัชนีแบบ B-tree เป็นดัชนีที่จัดเก็บข้อมูลแบบโครงสร้างต้นไม้ ซึ่งอยู่ในรูปแบบของ Balance tree ประกอบด้วย โหนดบนสุด เรียกว่า Root Node โหนดภายใน เรียกว่า Internal Node และโหนดสุดท้าย เรียกว่า Leaf Node

โดยทั่วไป ดัชนีแบบ B-tree มีโครงสร้างคือ Leaf node ทั้งหมดจะอยู่ในระดับเดียวกัน โดย Root node ประกอบด้วยโหนดลูกอย่างน้อย 2 โหนด สำหรับ Internal node ที่มี m คีย์ สามารถมีโหนดลูกได้ทั้งหมด m+1 โหนด และมีโหนดลูกได้อย่างน้อย (m+1)/2 โหนด ทุกๆ คีย์ที่อยู่บนโหนดจะเรียงจากน้อยไปหามากและมีจะพอยต์เตอร์ระหว่างคีย์ โดย Root node และ Internal node จะมีพอยต์เตอร์ 1 ตัวชี้ไปยังตำแหน่งของข้อมูล และพอยต์เตอร์อีก 1 ตัวชี้ไปยังโหนดลูก ซึ่งโหนดลูกที่เชื่อมมาจากพอยต์เตอร์ทางซ้ายของคีย์จะต้องมีค่าที่น้อยกว่าหรือเท่ากับคีย์นั้น ส่วนโหนดที่เชื่อมมาจากพอยต์เตอร์ทางขวาของคีย์จะต้องมีค่ามากกว่าคีย์นั้น และ Leaf node จะมีพอยต์เตอร์เพียง 1 ตัวที่ชี้ไปยังตำแหน่งของข้อมูล ความสูงของดัชนีแบบ B-tree ขึ้นอยู่กับจำนวนคีย์ที่สามารถให้มีได้ในแต่ละโหนด ถ้าจำนวนคีย์ทั้งหมดคือ n และแต่ละโหนดมีคีย์ได้ m คีย์ ความสูงของดัชนีแบบ B-tree เท่ากับ $log_m n$ เมื่อมีการค้นหาข้อมูลมาถึงโหนดที่มี m คีย์ ก็จะมีทางเลือกในการค้นหา m+1 เส้นทาง ดังนั้นการค้นหาข้อมูลบนดัชนีแบบ B-tree จึงใช้เวลาเป็น O($log_m n$) ตัวอย่างดัชนีแบบ B-tree แสดงดังภาพประกอบ 2-2



ภาพประกอบ 2-2 ตัวอย่างดัชนีแบบ B-tree (Silberschatz *et al.*, 2011)

2.2.2 ข้อดีของดัชนีแบบ B-tree

ดัชนีแบบ B-tree เป็นโครงสร้างข้อมูลที่ออกแบบมาสำหรับการเข้าถึงข้อมูลโดยตรงบนหน่วยความจำสำรอง เช่น ไฟล์ข้อมูลและฐานข้อมูลที่เก็บในดิสก์ มีการจัดการเกี่ยวกับ I/O ได้ดี ซึ่งระบบฐานข้อมูลส่วนใหญ่มีการนำดัชนีแบบ B-tree รวมถึงโครงสร้างอื่นๆ ที่อยู่บนพื้นฐานของดัชนีแบบ B-tree เช่น B+ tree และ B* tree ซึ่งมีการออกแบบให้มีประสิทธิภาพในการสอบถามข้อมูลแบบช่วงและ แอตทริบิวต์ที่มีคาร์ดินอลิตี้สูงได้ดีขึ้น

2.2.3 ข้อจำกัดของดัชนีแบบ B-tree

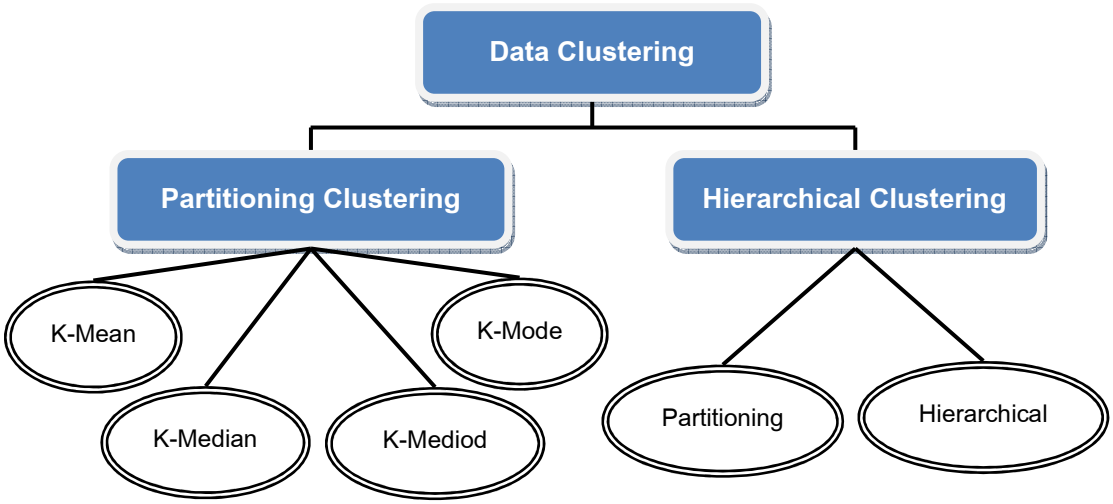
ดัชนีแบบ B-tree ไม่เหมาะสำหรับการสอบถามที่มีความซับซ้อน เนื่องจากจะต้องมีการสร้างคีย์ผสมขึ้นหากเป็นการสอบถามที่ต้องการข้อมูลมากกว่า 1 แอตทริบิวต์ ซึ่งมีความยุ่งยากและใช้เวลาในการสอบถามมากขึ้น นอกจากนี้การทำดัชนีแบบ B-tree ไม่เหมาะกับแอตทริบิวต์ที่มีคาร์ดินอลิตี้ต่ำ

2.3 การจัดกลุ่มข้อมูล

การจัดกลุ่มข้อมูล (Han and Kamber, 2000; Goebel and Gruenwald, 1999; Zaman *et al.*, 2004) เป็นเทคนิคหนึ่งของการทำ Data Mining ซึ่งเป็นการจัดชุดข้อมูลออกเป็นกลุ่ม โดยชุดข้อมูลที่มีลักษณะเหมือนกันหรือคล้ายกันจัดไว้ในกลุ่มเดียวกัน ขั้นตอนวิธีที่ใช้ในการจัดกลุ่มจะขึ้นอยู่กับความเหมือนของข้อมูล (Similarity) หรือความใกล้ชิด (Proximity) โดยอาศัยการวัดระยะระหว่างเวกเตอร์ของข้อมูล การวัดระยะระหว่างของเวกเตอร์มีหลายแบบ เช่น การวัดระยะแบบยูคลิเดียน การวัดระยะแบบแมนฮัตตัน และการวัดระยะแบบเชบิเชฟ เป็นต้น

ในการแบ่งกลุ่มข้อมูลนั้นจะไม่มีการจัดประเภท (Classes) ก่อน แต่จะเป็นการจัดเก็บอยู่ในกลุ่มเดียวกัน ซึ่งดูจากความคล้ายคลึงกันของข้อมูล และความคล้ายนั้นจะถูกกำหนดตามผู้ใช้งานที่จะเลือกให้ส่งผลในเรื่องใด เพื่อยึดถือว่าเป็นผลของการแบ่งกลุ่ม

ขั้นตอนวิธีการพิจารณาการจัดกลุ่มข้อมูล สามารถแยกออกได้หลายประเภท เช่น กรณีที่ทราบจำนวนกลุ่มที่ต้องการจัดกลุ่มข้อมูล และกรณีที่ไม่ทราบจำนวนกลุ่มที่ต้องการจัดกลุ่มข้อมูล เป็นต้น ตัวอย่างวิธีการจัดกลุ่มข้อมูลเช่น Partitioning Clustering และ Hierarchical Clustering เป็นต้น (Han and Kamber, 2000) แสดงดังภาพประกอบ 2-3



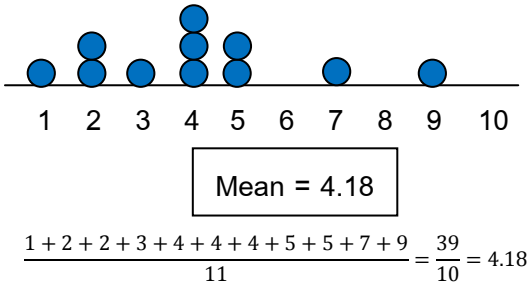
ภาพประกอบ 2-3 วิธีการการจัดกลุ่มข้อมูล

2.3.1 วิธีการจัดกลุ่มแบบ Partitioning Clustering

วิธีการจัดกลุ่มแบบ Partitioning Clustering (Sheng and Liu, 2004; Zaman *et al.*, 2004; He *et al.*, 2011; Huang, 1998) เป็นการจัดกลุ่มข้อมูลให้มีจำนวนเท่ากับจำนวนกลุ่มที่ต้องการ โดยทุกกลุ่มข้อมูลไม่มีการซ้อนทับกัน วิธีการจัดกลุ่มแบบ Partitioning Clustering มีขั้นตอนการทำงานพื้นฐาน ดังนี้

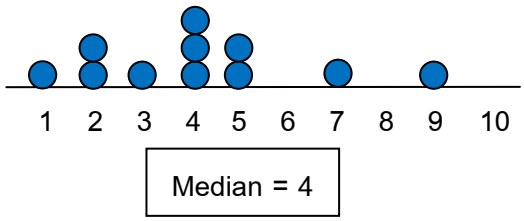
1. ทำการหาข้อมูล K ตัวซึ่งใช้เป็นค่าศูนย์กลาง หรือตัวแทนของกลุ่มในแต่ละกลุ่ม จำนวน K กลุ่ม
2. ทำการอ่านข้อมูล จากนั้นจัดให้ข้อมูลแต่ละตัวอยู่ในกลุ่มที่มีค่าระยะห่างน้อยที่สุด (เหมือนกันมากที่สุด)
3. หาค่าศูนย์กลางกลุ่มของแต่ละกลุ่มใหม่ จำนวน K กลุ่ม
4. ทำการหากลุ่มให้ข้อมูลทุกตัวจากศูนย์กลางใหม่ที่ได้
5. ทำซ้ำในขั้นตอนที่ 3 และ 4 จนกว่าข้อมูลทุกตัวจะไม่มีการเปลี่ยนกลุ่ม

วิธีการจัดกลุ่มแบบ Partitioning Clustering มีอัลกอริทึมหลายอัลกอริทึม ซึ่งแต่ละอัลกอริทึมจะมีขั้นตอนการทำงานพื้นฐานเดียวกัน แต่จะมีวิธีการเลือกค่าศูนย์กลาง หรือตัวแทนของกลุ่มที่แตกต่างกัน

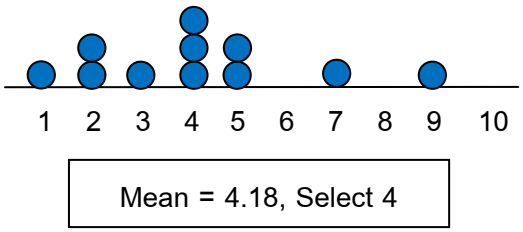


$$\frac{1+2+2+3+4+4+4+5+5+7+9}{11} = \frac{39}{10} = 4.18$$

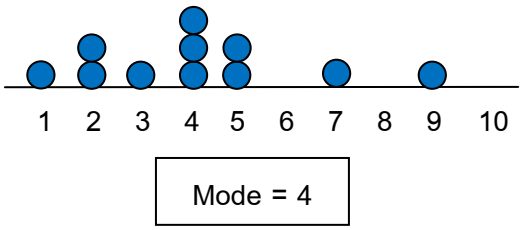
(ก) K-Mean



(ข) K-Median



(ค) K-Mediod



(ง) K-Mode

ภาพประกอบ 2-4 วิธีการเลือกค่าศูนย์กลางหรือตัวแทนของกลุ่ม

จากภาพประกอบ 2-4 จะมีวิธีการเลือกค่าศูนย์กลาง หรือตัวแทนของกลุ่มรูปแบบต่างๆ ดังนี้

- **K-Mean Clustering**

เป็นการนำค่าเฉลี่ยของข้อมูลมาเป็นค่าศูนย์กลาง หรือตัวแทนของกลุ่ม ตัวอย่างเช่น จากภาพประกอบ 2-4(ก) นำข้อมูลทั้ง 11 ค่ามาหาค่าเฉลี่ย จะได้ว่า ข้อมูลดังกล่าวมีค่าเฉลี่ยเท่ากับ 4.18 ดังนั้นค่า 4.18 จึงเป็นค่าศูนย์กลาง หรือตัวแทนของกลุ่ม

- **K-Median Clustering**

เป็นการนำค่ากึ่งกลางของข้อมูลที่ถูกจัดเรียงลำดับแล้วมาเป็นค่าศูนย์กลาง หรือตัวแทนของกลุ่ม ตัวอย่างเช่น จากภาพประกอบ 2-4(ข) นำข้อมูลทั้ง 11 ค่ามาจัดเรียงข้อมูลตามลำดับ จากนั้นแยกข้อมูลออกให้เท่าๆ กัน และนำค่าตรงกลางมาเป็นค่าศูนย์กลาง หรือตัวแทนของกลุ่ม นั่นคือ ค่า 4 เป็นค่าศูนย์กลาง หรือตัวแทนของกลุ่ม

- **K-Mediod Clustering**

เป็นการนำค่าจริงของข้อมูลที่มีค่าใกล้เคียงกับค่าเฉลี่ยของข้อมูลมาเป็นค่าศูนย์กลาง หรือตัวแทนของกลุ่ม ตัวอย่างเช่น จากภาพประกอบ 2-4(ค) นำข้อมูลทั้ง 11 ค่ามา

หาค่าเฉลี่ย จะได้ว่าข้อมูลดังกล่าวมีค่าเฉลี่ยเท่ากับ 4.18 แต่ค่า 4.18 ไม่มีอยู่ในค่าของข้อมูลจริง ดังนั้นจึงนำค่าข้อมูลจริงที่ใกล้เคียงมากที่สุด นั่นคือ ค่า 4 เป็นค่าศูนย์กลางหรือตัวแทนของกลุ่ม

- **K-Mode Clustering**

วิธีการหาค่าศูนย์กลาง หรือตัวแทนของกลุ่มแบบ K-Mode จะมีลักษณะข้อมูลเป็นข้อมูลที่ไม่สามารถนำมาคำนวณได้ หรือวัดค่าได้ยาก เช่น สี เพศ เป็นต้น ดังนั้นวิธีการหาศูนย์กลาง หรือตัวแทนของกลุ่มแบบ K-Mode จะนำค่าที่มีความถี่มากที่สุดมาเป็นค่าศูนย์กลางหรือตัวแทนกลุ่ม ตัวอย่างเช่น จากภาพประกอบ 2-4(ง) ค่าที่มีความถี่มากที่สุดคือ ค่า 4 ดังนั้นค่า 4 จึงเป็นค่าศูนย์กลาง หรือตัวแทนของกลุ่ม

ข้อดีของการจัดกลุ่มแบบ Partitioning Clustering

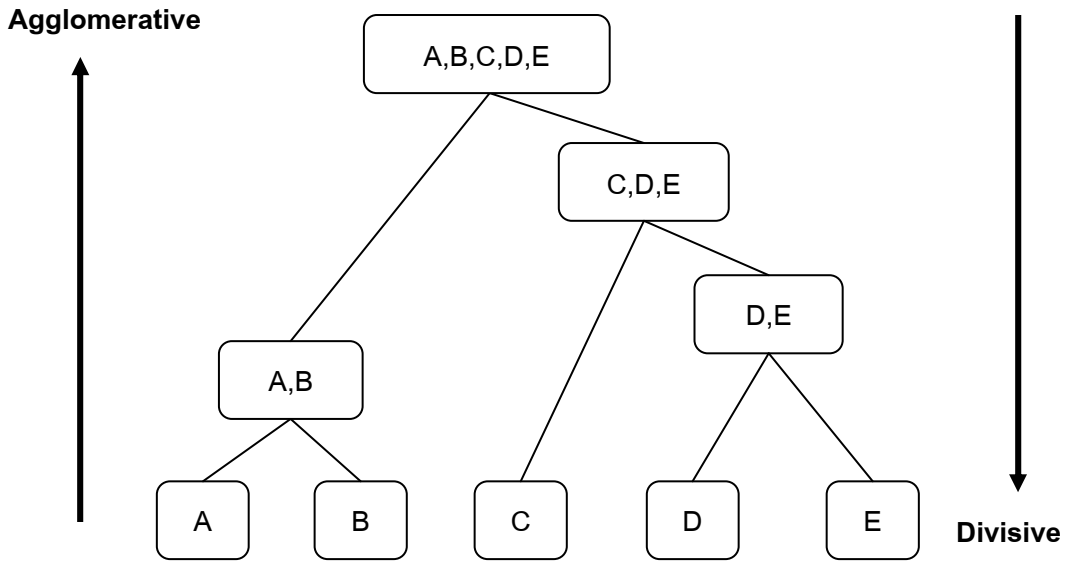
วิธีการจัดกลุ่มแบบ Partitioning Clustering เหมาะกับการจัดกลุ่มข้อมูลที่มีจำนวนมาก เพราะเมื่อมีจำนวนข้อมูลมาก วิธีการจัดกลุ่มแบบ Partitioning Clustering จะใช้เวลาในการคำนวณน้อยกว่าวิธีการจัดกลุ่มแบบ Hierarchical Clustering

ข้อจำกัดของการจัดกลุ่มแบบ Partitioning Clustering

วิธีการจัดกลุ่มแบบ Partitioning Clustering ผู้ใช้จำเป็นต้องกำหนดจำนวนกลุ่มข้อมูลที่แน่นอนไว้ก่อนล่วงหน้า ซึ่งในกรณีที่ผู้ใช้ไม่แน่ใจว่าควรกำหนดกลุ่มข้อมูลกี่กลุ่มจึงจะเหมาะสม ผู้ใช้อาจใช้วิธีใดวิธีหนึ่งในการหาจำนวนกลุ่มข้อมูล เช่น การวิเคราะห์การจัดกลุ่มแต่ละครั้ง โดยการกำหนดจำนวนกลุ่มที่แตกต่างกันออกไป แล้วพิจารณาจำนวนกลุ่มที่เหมาะสม เป็นต้น

2.3.2 วิธีการจัดกลุ่มแบบ Hierarchical Clustering

วิธีการการจัดกลุ่มแบบ Hierarchical Clustering (Han and Kamber, 2000; Silberschatz *et al.*, 2011; Yoon *et al.*, 2001) เป็นวิธีการจัดกลุ่มที่นิยมใช้ในการจัดกลุ่มข้อมูลที่ไม่ทราบจำนวนกลุ่มข้อมูลที่ชัดเจนมาก่อน วิธีการจัดกลุ่มแบบ Hierarchical Clustering แบ่งเป็น 2 เทคนิคย่อย คือ Agglomerative Hierarchical Clustering และ Divisive Hierarchical Clustering แสดงดังภาพประกอบ 2-5



ภาพประกอบ 2-5 ตัวอย่างการจัดกลุ่มข้อมูลแบบ Hierarchical Clustering

- **Divisive Hierarchical Clustering**

เทคนิคการจัดกลุ่มแบบ Divisive Hierarchical Clustering หรือ Top-Down Approach มีขั้นตอนการจัดกลุ่มพื้นฐาน ดังนี้

1. เริ่มต้นจะกำหนดให้มีกลุ่มข้อมูลเพียง 1 กลุ่มข้อมูลประกอบด้วยค่าข้อมูลทั้งหมด n ค่า
2. หาระยะทางของแต่ละกลุ่มข้อมูลที่ห่างกันมากที่สุด หรือมีความคล้ายกันน้อยที่สุด
3. ทำการแบ่งกลุ่มข้อมูลออกเป็น 2 กลุ่มข้อมูลย่อยที่มีระยะทางห่างกันมากที่สุด หรือมีความคล้ายกันน้อยที่สุด
4. ทำซ้ำในขั้นตอนที่ 1-3 จนกว่าจะมีจำนวนกลุ่มข้อมูลทั้งหมด n กลุ่มซึ่งแต่ละกลุ่มประกอบด้วยข้อมูลเพียง 1 ค่า

- **Agglomerative Hierarchical Clustering**

เทคนิคการจัดกลุ่มแบบ Agglomerative Hierarchical Clustering หรือ Bottom-Up Approach มีขั้นตอนการจัดกลุ่มพื้นฐาน ดังนี้

1. กำหนดให้แต่ละค่าข้อมูลเป็นกลุ่มข้อมูลของตัวเอง เช่น มีค่าข้อมูลทั้งหมด n ค่า ดังนั้น จะมีกลุ่มข้อมูลย่อยทั้งหมด n กลุ่มข้อมูล
2. หาระยะทางของแต่ละกลุ่มข้อมูลที่สั้นที่สุด หรือมีความคล้ายกันมากที่สุด

3. รวมกลุ่มข้อมูลที่มีระยะสั้นที่สุด หรือมีความคล้ายกันมากที่สุด ดังนั้นจะมีกลุ่มข้อมูลเหลืออยู่ n-1 กลุ่มย่อย

4. ทำซ้ำในขั้นตอนที่ 1-3 จนกว่าจะมีเพียง 1 กลุ่มข้อมูลซึ่งประกอบด้วยข้อมูล n ค่า

ข้อดีของการจัดกลุ่มแบบ Hierarchical Clustering

วิธีการจัดกลุ่มแบบ Hierarchical Clustering ผู้ใช้ไม่จำเป็นต้องทราบจำนวนกลุ่มข้อมูลที่แน่ชัด เนื่องจากวิธีการจัดกลุ่มแบบ Hierarchical Clustering จะเป็นการหาระยะห่าง หรือความคล้ายกันของแต่ละกลุ่มข้อมูล และรวมกลุ่มข้อมูลโดยอัตโนมัติ

ข้อจำกัดของการจัดกลุ่มแบบ Hierarchical Clustering

วิธีการจัดกลุ่มแบบ Hierarchical Clustering จะสิ้นเปลืองเวลาในการจัดกลุ่มข้อมูลมาก ถ้าจำนวนข้อมูลที่นำมาจัดกลุ่มมีจำนวนมาก เนื่องจากจะทำการลดจำนวนกลุ่มลงทีละ 1 กลุ่ม หรือเพิ่มจำนวนกลุ่มทีละ 1 กลุ่ม จะทำให้ต้องใช้เวลาในการประมวลผลนานขึ้น

วิทยานิพนธ์นี้ได้เลือกใช้วิธีการจัดกลุ่มแบบ Hierarchical Clustering มาช่วยในการจัดกลุ่มค่าของแอตทริบิวต์ที่มีการสอบถามไปด้วยกันและสอบถามร่วมกันเพื่อเพิ่มประสิทธิภาพในการสอบถามข้อมูลแบบสมาชิกและการสอบถามข้อมูลแบบค่าเท่ากัน เนื่องจากวิธีการจัดกลุ่มข้อมูลดังกล่าวเหมาะกับค่าข้อมูลในแอตทริบิวต์ที่นำมาดัชนีซึ่งเป็นข้อมูลที่ไม่ทราบจำนวนกลุ่มข้อมูลที่ชัดเจน

2.4 การลดรูปสมการบูลีน (Boolean Minimization)

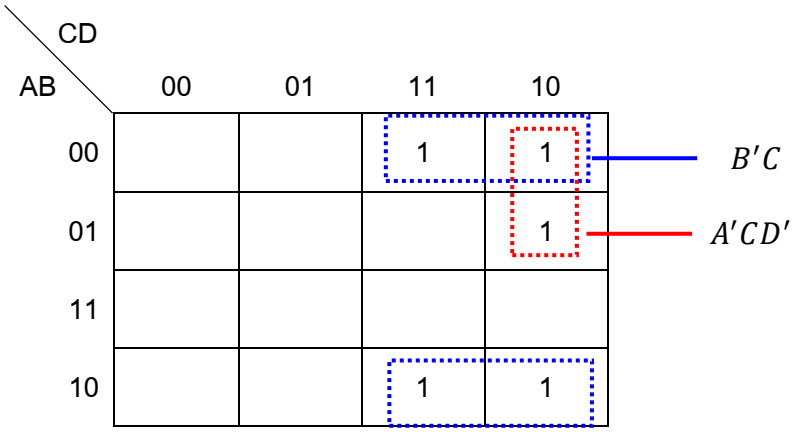
การลดรูปสมการบูลีนคือ การลดจำนวนตัวแปรในสมการให้มีจำนวนลดน้อยลง การลดรูปสมการบูลีนสามารถทำได้หลายวิธี เช่น ใช้ทฤษฎีบทของบูลีน แผนผังคาร์โนห์ (karnaugh map) และวิธีการของควิน-แมคคลอสกี้ (Quine-McCluskey) ซึ่งแต่ละวิธีการที่แตกต่างกัน ประโยชน์ของการลดรูปสมการบูลีนคือ เมื่อนำไปเขียนวงจรลอจิก จะทำให้จำนวนตัวดำเนินการในวงจรลดลง ต้นทุนในการทำงานก็จะลดลงตามไปด้วย วิธีการที่เหมาะสมในการลดรูปสมการบูลีนบนระบบคอมพิวเตอร์ ดังจะได้อธิบายต่อไป

2.4.1 การลดรูปสมการบูลีนโดยใช้แผนผังคาร์โนห์ (Karnaugh map)

การลดรูปสมการบูลีนโดยใช้แผนผังคาร์โนห์ (k-map) (Null and Lobur, 2003) เป็นวิธีการที่ง่ายและเหมาะกับตัวแปรตั้งแต่ 2 ตัวแปรขึ้นไปแต่ไม่ควรเกิน 4 ตัวแปร สำหรับการสร้างแผนผังคาร์โนห์สามารถสร้างได้จากตารางค่าความจริงและสามารถลดรูปของสมการบูลีนได้โดยการเขียนให้อยู่ให้รูปของ Sum of Product ในการลดรูปของสมการแต่ละตำแหน่งใน

แผนผังเรียกว่า เซลล์ (cell) ในแต่ละเซลล์จะถูกแทนค่าด้วย 0 หรือ 1 ตามสมการเพื่อจะนำมาลดรูปโดยมีหลักว่า รวมเซลล์ที่มีค่าเท่ากับ 1 ที่อยู่ติดกันเข้าด้วยกันซึ่งเซลล์ที่ถูกรวมเข้าด้วยกันจะต้องมีขนาด 2^i เมื่อ $i \geq 0$ และต้องรวมเซลล์ที่มีขนาดใหญ่ที่สุดเท่าที่สามารถจัดกลุ่มได้ก่อนจากนั้นจึงจัดกลุ่มให้มีขนาดเล็กลงมาตามลำดับ

ตัวอย่างการสร้างแผนผังคาร์โนห์โดยมีสมการบูลีนคือ $Y = A'B'CD' + A'B'CD + A'BCD' + AB'CD + AB'CD'$ จะได้แผนผังคาร์โนห์ตามภาพประกอบ 2-6 ซึ่งประกอบด้วย 4 ตัวแปรคือ A, B, C และ D ซึ่ง $B'C$ จะถูกแทนโดยค่า 1 ที่อยู่ใน 4 เซลล์ติดกันนั่นคือเกิดจากค่า B' และ C ร่วมกัน และ $A'CD'$ จะถูกแทนโดยค่า 1 ลงในเซลล์ซึ่งเกิดจาก A', C และ D' ร่วมกัน แสดงดังภาพประกอบ 2-6



ภาพประกอบ 2-6 แผนผังคาร์โนห์สำหรับ 4 ตัวแปร

อย่างไรก็ตามแผนผังคาร์โนห์เป็นวิธีการลดรูปสมการบูลีนที่ง่ายและเหมาะกับตัวแปรไม่เกิน 4 ตัวแปร เมื่อตัวแปรเพิ่มมากขึ้นจะทำให้การทำงานมีความซับซ้อนเพิ่มขึ้นส่งผลให้เวลาที่ใช้ในการทำงานเพิ่มมากขึ้นอย่างรวดเร็ว

2.4.2 วิธีการของควิน-แมคคลอสกี้ (Quine-McCluskey)

วิธีการของควิน-แมคคลอสกี้ (Quine, 1952; Tomaszewski *et al.*, 2003) เหมาะสำหรับการลดรูปสมการบูลีนที่มีตัวแปรจำนวนมาก โดยพยายามหาส่วนที่เป็น Essential Prime Implicant (ESS PI) และเป็นวิธีการที่มีความยืดหยุ่นมากกว่า k-map ในด้านการนำไปประยุกต์เพื่อสร้างโปรแกรมคอมพิวเตอร์อย่างมีประสิทธิภาพและด้านการใช้งานกับสมการที่มีจำนวนแปรมากๆ วิธีการของควิน-แมคคลอสกี้ประกอบด้วย 2 ขั้นตอนหลักได้แก่ ขั้นตอนแรกคือการสร้างตาราง Implication Table เพื่อรวมเทอมเข้าด้วยกัน และขั้นตอนที่สองคือการสร้างตาราง Prime Implicant Chart เพื่อหา Essential Prime Implicant

ตัวอย่างการลดรูปสมการบูลีนด้วยวิธีการของควิน-แมคคลอสกี้โดยมีสมการคือ $Y = A'B'CD' + A'B'CD + A'BCD' + AB'CD + AB'CD'$ สามารถทำได้ดังนี้

<u>ขั้นตอนที่ 1</u> สร้างตารางค่าความจริงจากสมการบูลีนดังกล่าว แสดงดังภาพประกอบ 2-7

A	B	C	D		Y
0	0	0	0		0
0	0	0	1		0
0	0	1	0		1
0	0	1	1		1
0	1	0	0		0
0	1	0	1		0
0	1	1	0		1
0	1	1	1		0
1	0	0	0		0
1	0	0	1		0
1	0	1	0		1
1	0	1	1		1
1	1	0	0		0
1	1	0	1		0
1	1	1	0		0
1	1	1	1		0

ภาพประกอบ 2-7 ตารางค่าความจริง

<u>ขั้นตอนที่ 2</u> รวมเทอมเข้าด้วยกันเพื่อลดจำนวนเทอม โดยพิจารณาจากเทอมที่มีค่าเลขฐานสองต่างกันเพียง 1 บิต ตัวอย่างเช่น 0010 กับ 0011 (ต่างกันที่บิตที่สี่) เป็นต้น ดังนั้นเพื่อให้ง่ายต่อการพิจารณาจะทำการแบ่งกลุ่มที่มีค่าเลขฐานสองต่างกันเพียง 1 บิต แสดงดังภาพประกอบ 2-8

จำนวนบิต 1	เทอม
1	0010
2	0011 0110 1010
3	1011

ภาพประกอบ 2-8 กลุ่มของเลขฐานสองที่มีบิตต่างกัน 1 บิต

จากภาพประกอบ 2-8 จะเห็นได้ว่าเราสามารถแบ่งกลุ่มของเทอมที่มีบิตต่างกัน 1 บิตได้ 3 กลุ่ม

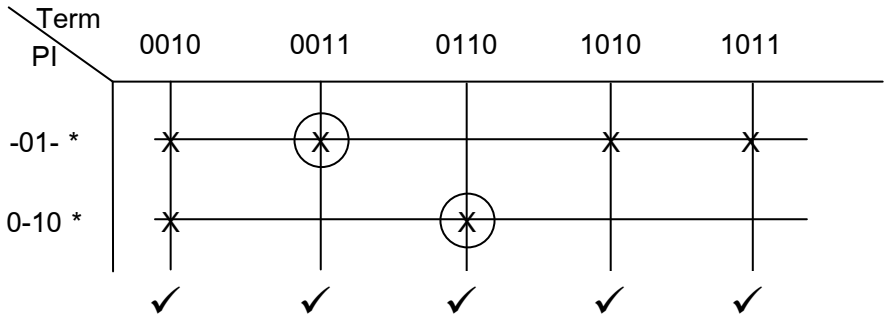
ขั้นตอนที่ 3 สร้างตาราง Implication Table เพื่อหา Prime Implicant โดยเปรียบเทียบค่าของเทอมแต่ละเทอมที่มีบิตต่างกันเพียง 1 บิต (ค่าของกลุ่มติดกันที่แบ่งไว้ในภาพประกอบ 2-8) เพื่อรวมเทอมเป็นตัวเดียวกัน โดยแทนตัวที่มีบิตต่างกันด้วยเครื่องหมาย “-” ตัวอย่างเช่นจากภาพประกอบ 2-8 เปรียบเทียบ 0010 กับ 0011 มีจำนวนบิตต่างกัน 1 บิตเมื่อรวมกันจะได้ 001- หรือเปรียบเทียบ 0010 กับ 0110 มีจำนวนบิตต่างกัน 1 บิตเมื่อรวมกันจะได้ 0-10 เป็นต้น เมื่อยุบรวมเทอมเรียบร้อยแล้วให้เขียนเครื่องหมาย “✓” ไว้ด้านหลังเทอมที่ถูกยุบรวมแล้ว หากยุบรวมเทอมไม่ได้ให้เขียนเครื่องหมาย “*” ไว้ด้านหลังเทอมที่ไม่สามารถยุบรวมได้ จากนั้นทำการเปรียบเทียบเทอมแต่ละเทอมไปเรื่อยๆ จนกระทั่งไม่สามารถยุบรวมได้อีก แสดงดังภาพประกอบ 2-9

รอบที่ 1	รอบที่ 2	รอบที่ 3
0010 ✓	001- ✓	-01- *
	0-10 *	-01- *
0011 ✓	-010 ✓	
0110 ✓		
1010 ✓	-011 ✓	
	101- ✓	
1011 ✓		

ภาพประกอบ 2-9 ตาราง Implication Table

จากภาพประกอบ 2-9 จะเห็นได้ว่ามี Prime Implicant -01- ซ้ำกันจึงเลือกเพียงตัวเดียวมาพิจารณาในขั้นตอนถัดไป

<u>ขั้นตอนที่ 4</u> จากภาพประกอบ 2-9 สร้างตาราง Prime Implicant Chart เพื่อหา Essential Prime Implicant แสดงดังภาพประกอบ 2-10



ภาพประกอบ 2-10 Prime Implicant Chart

<u>ขั้นตอนที่ 5</u> หา Essential Prime Implicant โดยมีขั้นตอนดังนี้

1. ในกรณีที่มี X เพียงตัวเดียวในคอลัมน์ของเทอมจะได้ว่า Prime Implicant ตัวนั้นเป็น Essential Prime Implicant ตัวอย่างเช่น จากภาพประกอบ 2-10 คอลัมน์ 0011 มี X เพียงตัวเดียวจะได้ -01- เป็น Essential Prime Implicant

2. เมื่อได้ -01- เป็น Essential Prime Implicant แล้วให้ดูว่า แถวของ -01- ครอบคลุมเทอมไหนบ้าง แล้วทำการตัดเทอมนั้นออกจาก Prime Implicant Chart ตัวอย่างเช่น จากภาพประกอบ 2-10 จะได้ว่า -01- ครอบคลุมเทอม 0010, 1010 และ 1011 ดังนั้นจึงตัดเทอมดังกล่าวออกจาก Prime Implicant Chart แล้วกลับไปทำตามข้อ 1 อีกครั้งจนกระทั่งได้ครบทุกเทอม

3. เมื่อได้ครบทุกเทอม จากภาพประกอบ 2-10 จะได้ Essential Prime Implicant คือ -01- และ 0-10 และสามารถเขียนเป็นคำตอบของการลดรูปสมการบูลีนได้ดังนี้

$$Y = A'B'CD' + A'B'CD + A'BCD' + AB'CD + AB'CD'$$
$$= B'C + A'CD'$$

วิธีการของควิน-แมคคลอสกี้เป็นวิธีการที่สามารถนำไปประยุกต์ใช้ในการเขียนโปรแกรมคอมพิวเตอร์ได้ง่ายและสามารถใช้งานกับสมการที่มีจำนวนตัวแปรจำนวนมากได้ ดังนั้นงานวิจัยนี้จึงเลือกใช้วิธีการของควิน-แมคคลอสกี้ในการลดรูปฟังก์ชันการเข้าถึงข้อมูลของดัชนีบิตแมปแบบเข้ารหัส

2.5 การทำดัชนีบิตแมป (Bitmap Indexing)

ดัชนีบิตแมป (Bontempo and Saracco, 2011; O'Neil and Quass, 1997; Chan and Ioannidis, 1998; Wu *et al.*, 2006; Stabno and Wrembel, 2009) เป็นเทคนิคหนึ่งที่ช่วยให้เราสามารถค้นหาข้อมูลได้รวดเร็ว เนื่องจากมีการดำเนินการระดับบิต เช่น AND, OR, XOR, NOT ระหว่างบิตแมปเวกเตอร์ก่อนเข้าถึงข้อมูลจริงจึงทำให้มีประสิทธิภาพในการค้นหาข้อมูลและประมวลผลข้อมูลจึงเหมาะที่จะนำมาใช้ในคลังข้อมูลซึ่งมีข้อมูลปริมาณมาก ยิ่งกว่านั้นการทำดัชนีบิตแมปยังใช้พื้นที่ในการจัดเก็บดัชนีที่มีประสิทธิภาพ โดยเฉพาะกรณีที่แอตทริบิวต์มีคาร์ดินอลิตี้ต่ำ งานวิจัยที่ผ่านมาได้มีการนำเสนอเทคนิคการทำดัชนีบิตแมปแบบต่างๆ ได้แก่

- ดัชนีบิตแมปแบบพื้นฐาน (Simple Bitmap Index)
- การบีบอัดดัชนีบิตแมป (Bitmap Index Compression)
  - การบีบอัดแบบ Word-Align Hybrid
  - การบีบอัดแบบ Run-Length Huffman
- การขยายแนวคิดดัชนีบิตแมป (Bitmap Index Extension)
  - ดัชนีบิตแมปแบบแถว (Range Bitmap Index)
  - ดัชนีบิตแมปแบบช่วง (Interval Bitmap Index)
  - ดัชนีบิตแมปแบบกระจาย (Scatter Bitmap Index)
  - ดัชนีบิตแมปแบบคู่กัน (Dual Bitmap Index)
  - ดัชนีบิตแมปแบบเข้ารหัส (Encoded Bitmap Index)

2.5.1 ดัชนีบิตแมปแบบพื้นฐาน (Simple Bitmap Index)

การทำดัชนีบิตแมปแบบพื้นฐาน (Bontempo and Saracco, 2011; O'Neil and Quass, 1997; Chan and Ioannidis, 1998;) บนแอตทริบิวต์ที่มีคาร์ดินอลิตี้เท่ากับ C (ค่าที่แตกต่างกัน) ประกอบด้วย C บิตแมปเวกเตอร์ นั่นคือ ใช้ 1 บิตแมปเวกเตอร์แทน 1 ค่าข้อมูล $(S_0, S_1, S_2, \dots, S_{C-1})$ โดยที่ S_v แทน บิตแมปเวกเตอร์ที่มีค่าของแอตทริบิวต์เท่ากับ v ซึ่งในการแทนค่ากำหนดให้บิตที่ i ของบิตแมปเวกเตอร์ S_v มีค่าเป็น 1 ก็ต่อเมื่อ แถวที่ i มีค่าของแอตทริบิวต์เท่ากับ v

RID	X
1	2
2	1
3	7
4	4
5	10
6	3
7	2
8	14
9	0
10	15
11	5
12	11
13	7
14	4

(ก) แอตทริบิวต์ X

S_{15}	S_{14}	...	S_5	S_4	S_3	S_2	S_1	S_0
0	0	…	0	0	0	1	0	0
0	0	…	0	0	0	0	1	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	…	0	0	0	1	0	0
0	1	…	0	0	0	0	0	0
0	0	…	0	0	0	0	0	1
1	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	0	0	0	0	0
0	0	…	0	1	0	0	0	0

(ข) ดัชนีบิตแมปแบบพื้นฐานบนแอตทริบิวต์ X

S_2
1
0
0
0
0
0
1
0
0
0
0
0
0
0

(ค) การสอบถาม “X=2”

ภาพประกอบ 2-11 ดัชนีบิตแมปแบบพื้นฐานบนแอตทริบิวต์ X เมื่อ C = 16

จากภาพประกอบ 2-11(ก) แอตทริบิวต์ X มีค่าที่เป็นไปได้ 16 ค่า คือ 0,1,2,…,15 ดังนั้นดัชนีบิตแมปแบบพื้นฐานประกอบด้วย 16 บิตแมปเวกเตอร์คือ $S_0, S_1, S_2, \dots, S_{15}$ โดยที่แต่ละบิตแมปเวกเตอร์แทนแต่ละค่าของแอตทริบิวต์ X เช่น บิตแมปเวกเตอร์ S_2 แทนค่าของแอตทริบิวต์ X ที่มีค่าเท่ากับ 2

จากภาพประกอบ 2-11(ข) แถวที่ 1 ของแอตทริบิวต์ X มีค่าเท่ากับ 2 ดังนั้น บิตที่ 1 ของบิตแมปเวกเตอร์ S_2 จึงมีค่าเท่ากับ 1 ส่วนบิตอื่นที่เหลือของบิตแมปเวกเตอร์ S_2 แถวที่ 1 มีค่าเท่ากับ 0

สำหรับการสอบถามแบบค่าเท่ากันบนดัชนีบิตแมปแบบพื้นฐานมีรูปแบบคือ "$X = v$" หมายถึงการสอบถามว่าบนแอตทริบิวต์ X แถวใดบ้างที่มีค่าเท่า v การสอบถามแบบค่าเท่ากันบนดัชนีบิตแมปแบบพื้นฐานสามารถทำได้โดยอ่านค่าบิตแมปเวกเตอร์ที่สัมพันธ์กับค่า v นั่นคือ บิตแมปเวกเตอร์ S_v ดังนั้นการสอบถามแบบค่าเท่ากันบนดัชนีบิตแมปแบบพื้นฐานจึงมีรูปแบบทั่วไปคือ

$$"X = v" = S_v$$

เมื่ออ่านบิตแมปเวกเตอร์ S_v แล้วพบว่าบิตใดมีค่าเป็น 1 แสดงว่าค่าแถวนั้นคือคำตอบ ตัวอย่างการสอบถามแบบเท่ากันบนดัชนีบิตแมปแบบพื้นฐาน จากภาพประกอบ 2-11

ต้องการทราบว่าบนแอตทริบิวต์ X แถวใดบ้างที่มีค่าเท่ากับ 2 จะได้ว่า "$X = v$" $= S_v$ ดังนั้นอ่านบิตแมปเวกเตอร์ S_2 พบว่าบิตที่ 1 และบิตที่ 7 มีค่าเท่ากับ 1 ส่วนบิตอื่นมีค่าเท่ากับ 0 จะได้ว่า แถวที่มีค่าของแอตทริบิวต์เท่ากับ 2 คือแถวที่ 1 และแถวที่ 7

สำหรับการสอบถามแบบสมาชิกบนดัชนีบิตแมปแบบพื้นฐานมีรูปแบบคือ $X\ in\ \{v_1, v_2, \dots, v_n\}$ หมายถึง การสอบถามว่าบนแอตทริบิวต์ X แถวใดบ้างที่มีค่าเท่ากับ $v_1, v_2, \dots, v_n$ การสอบถามแบบสมาชิกบนดัชนีบิตแมปแบบพื้นฐานสามารถทำได้โดย อ่านค่าบิตแมปเวกเตอร์ที่สัมพันธ์กับแต่ละค่าในเงื่อนไขและนำมาดำเนินการตรรกะ OR ตัวอย่างการสอบถามแบบสมาชิกบนดัชนีบิตแมปแบบพื้นฐาน จากภาพประกอบ 2-11 ต้องการทราบว่าบนแอตทริบิวต์ X แถวใดบ้างที่มีเท่ากับ {0, 3, 10, 11} ดังนั้นอ่านบิตแมปเวกเตอร์ S_0, S_3, S_{10} และ S_{11} จากนั้นดำเนินตรรกะ OR กันจะได้แถวที่มีค่าของแอตทริบิวต์เท่ากับ {0, 3, 10, 11} คือแถวที่ 5, 6, 9 และ 12

ข้อดีของดัชนีบิตแมปแบบพื้นฐาน

ดัชนีบิตแมปแบบพื้นฐานเหมาะกับแอตทริบิวต์ที่มีคาร์ดินอลิตี้ต่ำ เช่น แอตทริบิวต์เพศที่มีคาร์ดินอลิตี้เท่ากับ 2 คือ เพศชายและเพศหญิง เพราะมีการใช้ 2 บิตแมปเวกเตอร์ในการสร้างดัชนีทำให้ลดพื้นที่ในการจัดเก็บดัชนีได้ดี

ข้อจำกัดของดัชนีบิตแมปแบบพื้นฐาน

ดัชนีบิตแมปแบบพื้นฐานไม่เหมาะกับแอตทริบิวต์ที่มีคาร์ดินอลิตี้สูงๆ เมื่อคาร์ดินอลิตี้ของแอตทริบิวต์ที่นำมาสร้างดัชนีสูงขึ้น จำนวนบิตแมปเวกเตอร์ที่ใช้ในการสร้างดัชนีบิตแมปแบบพื้นฐานจะสูงขึ้นด้วยทำให้พื้นที่ที่ใช้จัดเก็บดัชนีบิตแมปแบบพื้นฐานเพิ่มมากขึ้นและใช้ประโยชน์จากบิตแมปเวกเตอร์ได้ไม่เต็มที่ ซึ่งปัญหาที่แอตทริบิวต์มีคาร์ดินอลิตี้สูงขึ้นนำไปสู่การคิดค้นเทคนิคการทำดัชนีบิตแมปแบบอื่นๆ ตามมา

2.5.2 การบีบอัดดัชนีบิตแมป

การบีบอัดดัชนีบิตแมปหมายถึง การบีบอัดแต่ละบิตแมปเวกเตอร์ของดัชนีบิตแมปแบบพื้นฐาน เทคนิคการบีบอัดดัชนีที่ได้ศึกษาได้แก่ การบีบอัดแบบ Word-Align Hybrid (WAH) และ Run-Length Huffman (RLH) มีรายละเอียดดังนี้

2.5.2.1 การบีบอัดดัชนีบิตแมปแบบ Word-Align Hybrid (WAH)

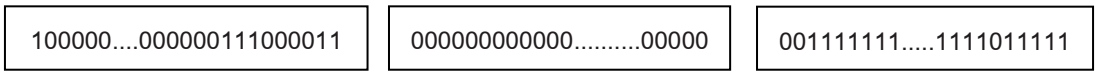
เทคนิคการบีบอัดดัชนีบิตแมปแบบ WAH (Wu *et al.*, 2006; Stabno and Wrembel, 2009) มีแนวคิดในการบีบอัดคือ จะทำการบีบอัดบิตที่มีค่าเหมือนกันและอยู่ติดกันในแต่ละบิตแมปเวกเตอร์ หลักการบีบอัดแบบ WAH มี 3 ขั้นตอน ดังนี้

- แบ่งบิตแมปออกเป็นกลุ่มๆ ละ 31 บิต ดังตัวอย่างบิตแมปเวกเตอร์ในภาพประกอบ 2-12 สามารถแบ่งได้เป็น 176 กลุ่ม แสดงดังภาพประกอบ 2-12(ข)

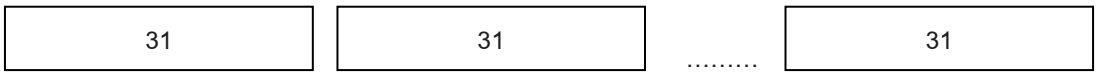
- รวมกลุ่มของกลุ่มที่มีค่าบิตเหมือนกันและอยู่ติดกันให้เหลือเพียงกลุ่มเดียว ดังแสดงดังภาพประกอบ 2-12(ค) โดยกลุ่มที่ 1 ข้อมูลมีบิตแตกต่างกัน ภายในกลุ่มที่ 2 ถึง 175 มีบิตเหมือนกันคือบิต 0 จึงรวมให้เป็นกลุ่มเดียวที่มีจำนวนบิตเท่ากับ 174*31 บิต และกลุ่มที่ 3 มีบิตแตกต่างกัน

- บีบอัดให้แต่ละกลุ่มมี 32 บิต มีหลักการคือ สำหรับกลุ่มที่มีค่าบิตแตกต่างกันให้เติมบิต 0 ไว้ทางบิตซ้ายสุด สำหรับกลุ่มที่เกิดจากการรวมกันหลายๆ กลุ่มให้เติมบิต 1 ไว้ทางบิตซ้ายสุดและบิตถัดมาเป็นค่าของบิตที่ถูกบีบอัดคือ บิต 0 ส่วนอีก 30 บิตที่เหลือแทนจำนวนกลุ่มที่ถูกบีบอัดเข้าด้วยกัน ดังภาพประกอบ 2-12(ง) เป็นผลลัพธ์ที่เกิดจากการบีบอัดดัชนีบิตแมปแบบ WAH

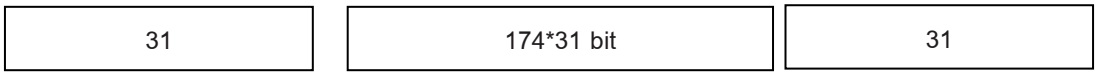
สำหรับการสอบถามข้อมูลมีการดำเนินการเช่นเดียวกับดัชนีบิตแมปแบบพื้นฐาน แต่จะต้องทำให้บิตแมปเวกเตอร์ที่ถูกบีบอัดนั้นกลับคืนสู่สภาพเดิมเหมือนกับตอนที่ยังไม่ได้บีบอัดจึงสามารถดำเนินการได้



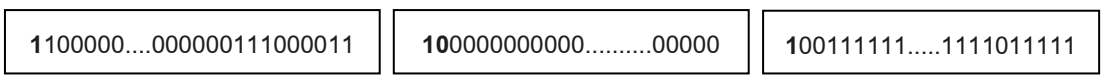
(ก) ตัวอย่างบิตแมปเวกเตอร์ที่จะทำการบีบอัดจำนวน 5456 บิต



(ข) แบ่งบิตแมปเวกเตอร์ออกเป็นกลุ่มๆ 31 บิต



(ค) รวมกลุ่มที่มีค่าเหมือนกันและอยู่ติดกัน



(ง) เข้ารหัสแต่ละกลุ่มให้มี 32 บิต

ภาพประกอบ 2-12 ขั้นตอนการบีบอัดดัชนีบิตแมปแบบ WAH (Hamadou and Yang, 2008)

ข้อดีของการบีบอัดดัชนีบิตแมปแบบ WAH

การบีบอัดดัชนีบิตแมปแบบ WAH ใช้พื้นที่ในการจัดเก็บดัชนีและเวลาในการสอบถามข้อมูลน้อยกว่าดัชนีบิตแมปแบบพื้นฐาน เมื่อแต่ละบิตแมปเวกเตอร์มีบิตที่เหมือนกันและอยู่ติดกันจำนวนมาก เนื่องจากขนาดดัชนีลดลง และมีการใช้ I/O น้อยลง

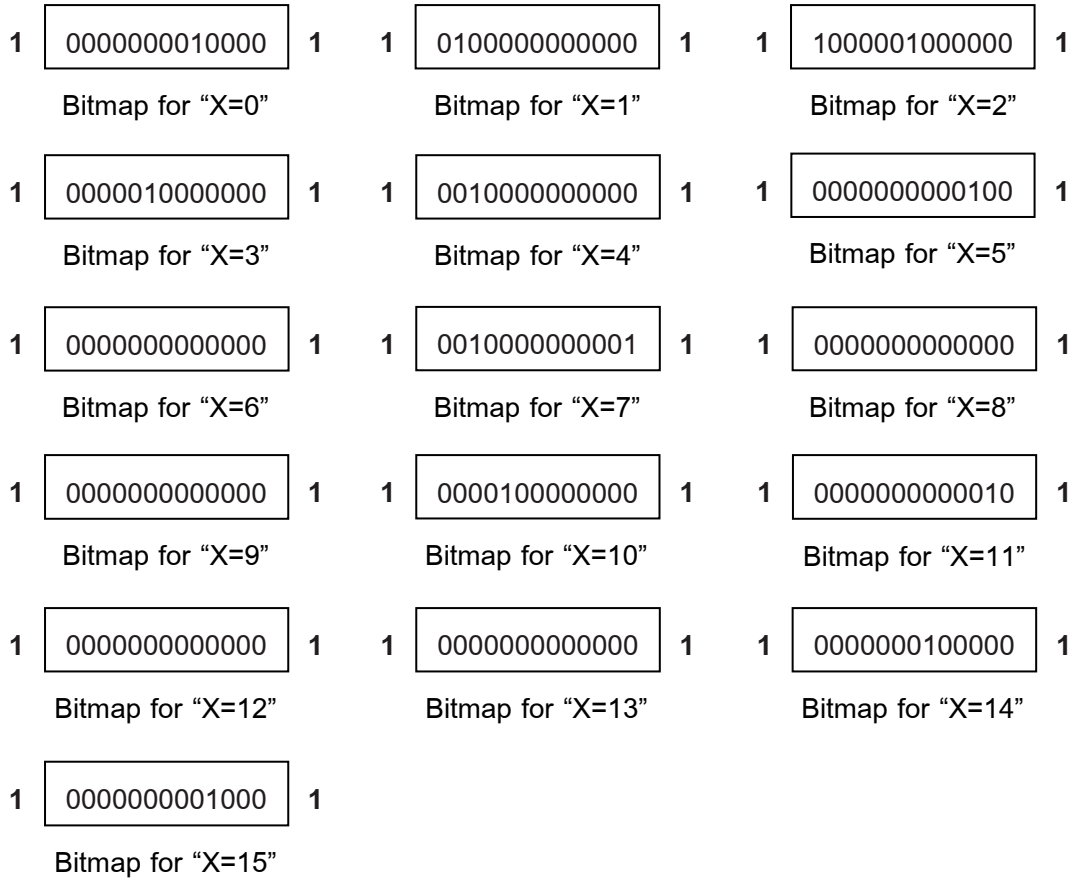
ข้อจำกัดของการบีบอัดดัชนีบิตแมปแบบ WAH

การบีบอัดดัชนีบิตแมปแบบ WAH ไม่สามารถดำเนินการตรรกะบนดัชนีได้โดยตรง นอกจากนี้หากแต่ละบิตแมปเวกเตอร์มีการกระจายของบิต 0 และบิต 1 อย่างสม่ำเสมอ นั่นคือแต่ละกลุ่มมีค่าบิตที่แตกต่างกัน มีกลุ่มจำนวนน้อยที่มีบิตเหมือนกันและอยู่ติดกันอย่างต่อเนื่อง จะทำให้ประสิทธิภาพการบีบอัดน้อยลง

2.5.2.2 การบีบอัดดัชนีบิตแมปแบบ Run-Length Huffman (RLH)

การบีบอัดดัชนีบิตแมปแบบ RLH (Stabno and Wrembel, 2009) มีขั้นตอนคือ มีขั้นตอนคือ จากดัชนีบิตแมปแบบพื้นฐาน female และ male ดังภาพประกอบ 2-13

- แปลงบิตแมปเวกเตอร์ โดยการหาจำนวนบิต 0 ที่อยู่ระหว่างบิต 1 สองบิต เมื่อกำหนดให้บิตซ้ายสุดและขวาสุดมีค่าเท่ากับ 1 ตัวอย่างเช่น บิตแมปเวกเตอร์ “X = 2” ดังภาพประกอบ 2-13 จะเห็นได้ว่า บิตซ้ายสุดกับบิต 1 ตัวแรกของบิตแมปเวกเตอร์ “X = 2” มีจำนวนบิต 0 ระหว่างบิต 1 ทั้งสองอยู่ 0 บิต สำหรับบิต 1 ตัวแรกกับบิต 1 ตัวที่สองมีจำนวนบิต 0 อยู่ระหว่างบิต 1 ทั้งสองอยู่ 5 บิต และบิต 1 ตัวที่สองกับบิต 1 ขวาสุดมีจำนวนบิต 0 อยู่ระหว่างบิต 1 ทั้งสองอยู่ 7 บิตเป็นต้น ดังนั้นจากบิตแมปเวกเตอร์ ดังภาพประกอบ 2-13 จะได้ผลลัพธ์การแปลงบิตแมปเวกเตอร์ของทั้งสองบิตแมปเวกเตอร์ดังภาพประกอบ 2-14
- นับความถี่ของแต่ละค่าทั้งหมด จากภาพประกอบ 2-14 ได้ผลลัพธ์แสดงดังภาพประกอบ 2-15
- นำความถี่ที่ได้ มาสร้าง Huffman Tree แสดงดังภาพประกอบ 2-16 เพื่อหา Huffman Code ของแต่ละค่า แสดงดังภาพประกอบ 2-17
- นำ Huffman Code ที่ได้ไปแทนที่แต่ละค่าของผลลัพธ์ที่ได้ในขั้นตอนแรก จะได้ผลลัพธ์สุดท้ายของการบีบอัดดัชนีบิตแมปแบบ RLH แสดงดังภาพประกอบ 2-18



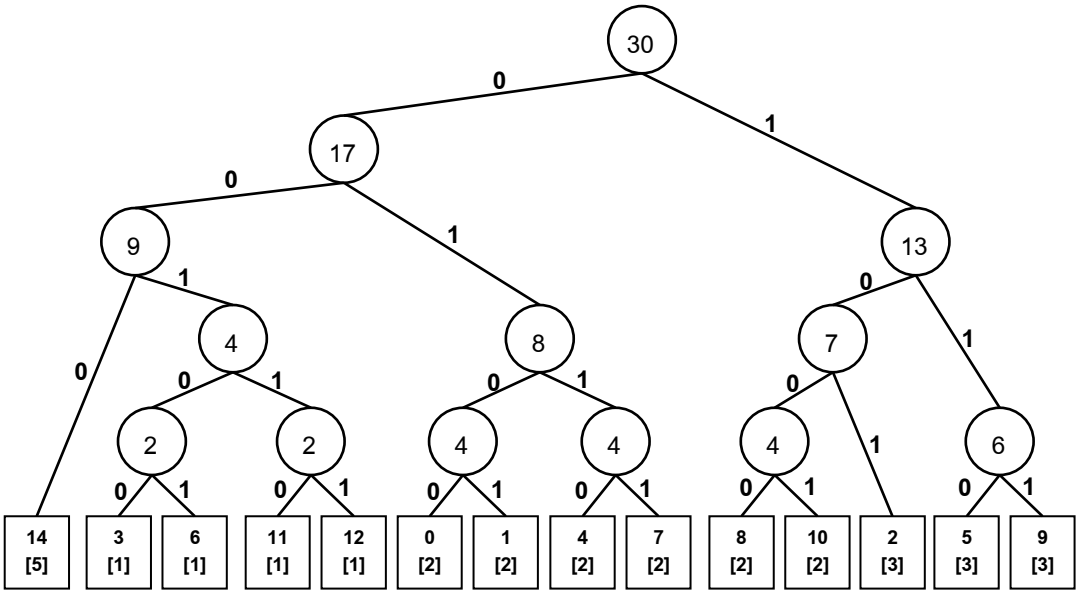
ภาพประกอบ 2-13 การบีบอัดดัชนีบิตแมปแบบ RLH

Value of Attribute X															
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
8	1	0	5	2	10	14	2	14	14	4	11	14	14	7	9
5	12	5	8	10	3		9			9	2			6	4
		7		0			1								

ภาพประกอบ 2-14 ระยะห่างระหว่างบิต 1 สองบิต

distance	14	2	5	9	0	1	4	7	8	10	3	6	11	12
frequency	5	3	3	3	2	2	2	2	2	2	1	1	1	1

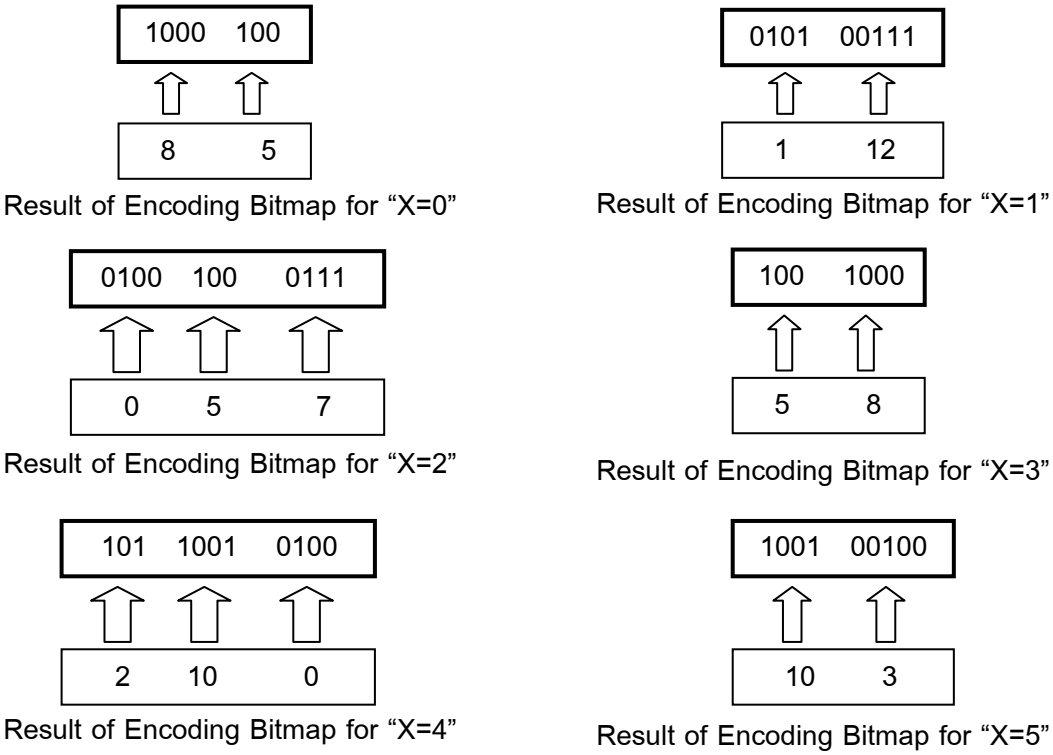
ภาพประกอบ 2-15 ความถี่ของระยะห่าง



ภาพประกอบ 2-16 Huffman Tree

Symbol	Symbol Code
0	0100
1	0101
2	101
3	00100
4	0110
5	110
6	00101
7	0111
8	1000
9	111
10	1001
11	00110
12	00111
14	000

ภาพประกอบ 2-17 Huffman Code



ภาพประกอบ 2-18 ตัวอย่างผลลัพธ์ที่ได้จากการบีบอัดดัชนีบิตแมปแบบ RLH

สำหรับการสอบถามข้อมูลมีการดำเนินการเช่นเดียวกับดัชนีบิตแมปแบบพื้นฐาน แต่จะต้องทำให้บิตแมปเวกเตอร์ที่ถูกบีบอัดนั้นกลับคืนสู่สภาพเดิมเหมือนกับตอนที่ยังไม่ได้บีบอัดจึงสามารถดำเนินการได้

ข้อดีของการบีบอัดดัชนีบิตแมปแบบ RLH

การบีบอัดดัชนีบิตแมปแบบ RLH เหมาะกับแอตทริบิวต์ที่มีคาร์ดินอลิตี้สูง สามารถลดพื้นที่ในการจัดเก็บดัชนีได้มากกว่าดัชนีบิตแมปแบบพื้นฐาน และการบีบอัดดัชนีบิตแมปแบบ WAH และการบีบอัดดัชนีบิตแมปแบบ RLH ใช้เวลาในการสอบถามข้อมูลน้อยกว่าดัชนีบิตแมปแบบพื้นฐานและการบีบอัดดัชนีบิตแมปแบบ WAH เนื่องจากมีการใช้ I/O น้อยกว่า และหาก Huffman Tree มีขนาดเล็กสามารถโหลดเข้าสู่หน่วยความจำหลักได้ในครั้งเดียว จึงทำให้การบีบอัดแบบ RLH มีประสิทธิภาพมากขึ้น

ข้อจำกัดของการบีบอัดดัชนีบิตแมปแบบ RLH

การบีบอัดดัชนีบิตแมปแบบ RLH ไม่เหมาะกับแอตทริบิวต์ที่มีคาร์ดินอลิตี้ต่ำ เนื่องจากบิตแมปเวกเตอร์มีบิต 1 จำนวนมาก ทำให้จำนวนบิต 0 ที่อยู่ระหว่างบิต 1 สองบิตมีจำนวนที่แตกต่างกันหลายค่าทำให้ Huffman Tree มีขนาดใหญ่ส่งผลต่อประสิทธิภาพการบีบอัดแบบ RLH ลดลง

2.5.3 การขยายแนวคิดดัชนีบิตแมป

การขยายแนวความคิดดัชนีบิตแมปเป็นการขยายโครงสร้างของดัชนีบิตแมป เพื่อให้ใช้เนื้อที่ในการจัดเก็บน้อยลง แต่ยังคงมีประสิทธิภาพในการสอบถามข้อมูล การสร้างดัชนีบิตแมปโดยการขยายแนวความคิดดัชนีบิตแมปที่ได้ศึกษาได้แก่ ดัชนีบิตแมปแบบแถว (Range Bitmap Index) ดัชนีบิตแมปแบบช่วง (Interval Bitmap Index) ดัชนีบิตแมปแบบกระจาย (Scatter Bitmap Index) ดัชนีบิตแมปแบบคู่กัน (Dual Bitmap Index) และดัชนีบิตแมปแบบเข้ารหัส (Encoded Bitmap Index) เป็นต้น

2.5.3.1 ดัชนีบิตแมปแบบแถว (Range Bitmap Index)

การทำดัชนีบิตแมปแบบแถว (Wu and Yu, 1998; Chan and Ioannidis, 1999) บนแอตทริบิวต์ที่มีคาร์ดินอลิตี้เท่ากับ C ประกอบด้วย $C-1$ บิตแมปเวกเตอร์ คือ $R_0, R_1, \ldots, R_{C-2}$ โดยที่ R^j อยู่ในช่วงของ $[0, j]$ ในการแทนค่าข้อมูลจะกำหนดให้ บิตที่ i ของบิตแมปเวกเตอร์ R^j มีค่าเท่ากับ 1 เมื่อแถวที่ i มีค่าของแอตทริบิวต์อยู่ในช่วง $[0, j]$ ส่วนบิตอื่นมีค่าเท่ากับ

RID	X
1	2
2	1
3	7
4	4
5	10
6	3
7	2
8	14
9	0
10	15
11	5
12	11
13	7
14	4

(ก) แอตทริบิวต์ X

R_{14}	R_{13}	R_{12}	R_{11}	...	R_6	R_5	R_4	R_3	R_2	R_1	R_0
1	1	1	1	…	1	1	1	1	1	0	0
1	1	1	1	…	1	1	1	1	1	1	0
1	1	1	1	…	0	0	0	0	0	0	0
1	1	1	1	…	1	1	1	0	0	0	0
1	1	1	1	…	0	0	0	0	0	0	0
1	1	1	1	…	1	1	1	1	0	0	0
1	1	1	1	…	1	1	1	1	1	0	0
1	0	0	0	…	0	0	0	0	0	0	0
1	1	1	1	…	1	1	1	1	1	1	1
0	0	0	0	…	0	0	0	0	0	0	0
1	1	1	1	…	1	1	0	0	0	0	0
1	1	1	1	…	0	0	0	0	0	0	0
1	1	1	1	…	0	0	0	0	0	0	0
1	1	1	1	…	1	1	1	0	0	0	0

(ข) ดัชนีบิตแมปแบบแถวบนแอตทริบิวต์ X

ภาพประกอบ 2-19 ดัชนีบิตแมปแบบแถวบนแอตทริบิวต์ X เมื่อ C = 16

จากภาพประกอบ 2-19(ก) แอตทริบิวต์ X มีค่าที่เป็นไปได้ 16 ค่าคือ 0, 1, 2,..., 15 ดังนั้นดัชนีบิตแมปแบบแถวประกอบด้วย 16 - 1 = 15 บิตแมปเวกเตอร์คือ $R_0, R_1, R_2, \dots, R_{14}$

จากภาพประกอบ 2-19(ข) แถวที่ 1 ของแอตทริบิวต์ X มีค่าเท่ากับ 2 ซึ่งอยู่ในช่วงของบิตแมปเวกเตอร์ $R_2, R_3, R_4, \dots, R_{14}$ ดังนั้นบิตที่ 1 ของบิตแมปเวกเตอร์ $R_2, R_3, R_4, \dots, R_{14}$ จึงมีค่าเท่ากับ 1 ส่วนบิตที่ 1 ของบิตแมปเวกเตอร์ R_0 และ R_1 มีค่าเท่ากับ 0

สำหรับการสอบถามแบบค่าเท่ากันบนดัชนีบิตแมปแบบแถวมีรูปแบบทั่วไป คือ

$$"v_1 \le X \le v_2" = \begin{cases} R_0 & \textit{If } v_1 = v_2 = 0 \\ R_{v_1} \oplus R_{v_1-1} & \textit{If } 0 < v_1 = v_2 < C-1 \\ \overline{R_{C-2}} & \textit{If } v_1 = v_2 = C-1 \\ \overline{R_{v_1-1}} & \textit{If } 0 < v_1 < C-1, v_2 = C-1 \\ R_{v_2} & \textit{If } v_1 = 0, 0 \le v_2 < C-1 \\ R_{v_2} \oplus R_{v_1-1} & \textit{Otherwise} \end{cases}$$

ตัวอย่างการสอบถามแบบค่าเท่ากันบนดัชนีบิตแมปแบบแถว จากภาพประกอบ 2-19 ต้องการทราบว่า บนแอตทริบิวต์ X แถวใดบ้างมีค่าเท่ากับ 2 ซึ่งตรงกับเงื่อนไขที่ 2 ($v_1 = v_2 = 2$ และ $0 < 2 < 15$) จะได้ว่าสามารถหาคำตอบได้จาก "$X = 2$" เท่ากับ $R_2 \oplus R_1$ ซึ่งแบ่งเป็นขั้นตอนการหาคำตอบได้ดังนี้

1) อ่านบิตแมปเวกเตอร์ R_2

2) อ่านบิตแมปเวกเตอร์ R_1

3) ดำเนินการตรรกะ XOR บิตต่อบิตระหว่างบิตแมปเวกเตอร์ผลลัพธ์ที่ได้จากข้อ 1) และ 2)

4) คำตอบที่ได้ พิจารณาจากบิตแมปเวกเตอร์ผลลัพธ์ที่ได้จากข้อ 3) พบว่าบิตแถวที่ 1 และแถวที่ 6 มีค่าเท่ากับ 1 ส่วนบิตอื่นๆ มีค่าเท่ากับ 0 จึงได้ว่าแถวที่มีค่าของแอตทริบิวต์เท่ากับ 2 คือแถวที่ 1 และแถวที่ 7

สำหรับการสอบถามแบบสมาชิกบนดัชนีบิตแมปแบบแถวสามารถหาได้โดยการหาบิตแมปเวกเตอร์ผลลัพธ์ของแต่ละค่าหรือการแปลงคำถามสอบถามให้ตรงกับเงื่อนไขแล้วนำมาดำเนินตรรกะ OR เช่น จากภาพประกอบ 2-19 ต้องการทราบว่าบนแอตทริบิวต์ X มีแถวใดบ้างที่มีค่าเท่ากับ {0, 3, 10, 11} ผลลัพธ์ของการสอบถามคือ $R_0 + (R_3 \oplus R_2) + (R_{11} \oplus R_9)$ นั่นคือแถวที่ 5, 6, 9 และ 12 เป็นคำตอบของการสอบถาม

ข้อดีของดัชนีบิตแมปแบบแถว

การทำดัชนีบิตแมปแบบแถวใช้พื้นที่ในการจัดเก็บดัชนีน้อยกว่าการทำดัชนีบิตแมปแบบพื้นฐานกล่าวคือ สามารถลดพื้นที่ในการจัดเก็บดัชนีได้ $C-1$ และในการสอบถามแบบค่าเท่ากันมีโอกาสตรวจสอบเพียง 1 บิตแมปเวกเตอร์

ข้อจำกัดของดัชนีบิตแมปแบบแถว

ในการสอบถามข้อมูลแบบค่าเท่ากันและแบบสมาชิกของดัชนีบิตแมปแบบแถวยังด้อยประสิทธิภาพกว่าดัชนีบิตแมปแบบพื้นฐาน เนื่องจากการสอบถามแต่ละค่าบนดัชนีบิตแมปแบบแถวใช้การดำเนินการตรรกะระหว่าง 2 บิตแมปเวกเตอร์ แต่การสอบถามแต่ละค่าบนดัชนีบิตแมปแบบพื้นฐานตรวจสอบเพียง 1 บิตแมปเวกเตอร์เท่านั้น

2.5.3.2 ดัชนีบิตแมปแบบช่วง (Interval Bitmap Index)

การทำดัชนีบิตแมปแบบช่วง (Chan and Ioannidis, 1999) บนแอตทริบิวต์ที่มีคาร์ดินอลิตี้เท่ากับ C ประกอบด้วย $\lceil C/2 \rceil$ บิตแมปเวกเตอร์ คือ $I_0, I_1, \ldots, I_{\lceil C/2 \rceil - 1}$ โดยที่ I^j อยู่ในช่วงของ $[j, j+m]$ เมื่อ $m = \lfloor C/2 \rfloor - 1$ ในการแทนค่าจะกำหนดให้ บิตที่ i ของบิตแมปเวกเตอร์ I^j มีค่าเท่ากับ 1 เมื่อแถวที่ i มีค่าของแอตทริบิวต์ที่ทำดัชนีอยู่ในช่วง $[j, j+m]$ ส่วนบิตอื่นมีค่าเท่ากับ 0

RID	X
1	2
2	1
3	7
4	4
5	10
6	3
7	2
8	14
9	0
10	15
11	5
12	11
13	7
14	4

(ก) แอตทริบิวต์ X

I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0
0	0	0	0	0	1	1	1
0	0	0	0	0	0	1	1
1	1	1	1	1	1	1	1
0	0	0	1	1	1	1	1
1	1	1	1	1	0	0	0
0	0	0	0	1	1	1	1
0	0	0	0	0	1	1	1
1	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	1	1	1	1	1	1
1	1	1	1	0	0	0	0
1	1	1	1	1	1	1	1
0	0	0	1	1	1	1	1

(ข) ดัชนีบิตแมปแบบช่วงบนแอตทริบิวต์ X

I_3	I_2	$I_2 \wedge \overline{I_3}$
0	1	1
0	0	0
1	1	0
1	1	0
1	0	0
1	1	0
0	1	1
0	0	0
0	0	0
0	0	0
1	1	0
0	0	0
1	1	0
1	1	0

(ค) การสอบถาม “X = 2”

ภาพประกอบ 2-20 ดัชนีบิตแมปแบบช่วงบนแอตทริบิวต์ X เมื่อ C = 16

จากภาพประกอบ 2-20(ก) แอตทริบิวต์ X มีค่าที่เป็นไปได้ 16 ค่า คือ 0, 1, 2,..., 15 ดังนั้นดัชนีบิตแมปแบบช่วงประกอบด้วย $\lceil 16/2 \rceil$ = 8 บิตแมปเวกเตอร์คือ $I_0, I_1, \ldots, I_7$ และ $m = \lceil 16/2 \rceil - 1$ = 7 ดังนั้นบิตแมปเวกเตอร์ I^j แทนค่าแอตทริบิวต์ X ที่อยู่ช่วง $[j, j+7]$ จึงได้ว่า

$$I_0 = [0,7], \qquad I_1 = [1,8],$$
$$I_2 = [2,9], \qquad I_3 = [3,10],$$
$$I_4 = [4,11], \qquad I_5 = [5,12],$$
$$I_6 = [6,13], \qquad I_7 = [7,14]$$

จากภาพประกอบ 2-20(ข) แถวที่ 1 ของแอตทริบิวต์ X มีค่าเท่ากับ 2 ซึ่งอยู่ในช่วงของบิตแมปเวกเตอร์ I_0, I_1 และ I_2 ดังนั้นบิตที่ 1 ของบิตแมปเวกเตอร์ I_0, I_1 และ I_2 จึงมีค่าเท่ากับ 1 ส่วนบิตอื่นที่เหลือในแถวที่ 1 มีค่าเท่ากับ 0

การสอบถามแบบค่าเท่ากันของดัชนีบิตแมปแบบช่วงมีรูปแบบทั่วไปคือ

$$\text{"}X = v\text{"} = \begin{cases} I_0 & \textit{if } v = 0,\ m = 0 \\ \overline{I_0} & \textit{if } v = 1,\ C = 2 \\ I_1 & \textit{if } v = 1,\ C = 3 \\ I_v \wedge \overline{I_{v+1}} & \textit{if } v < m \\ I_v \wedge I_0 & \textit{if } v = m,\ m > 0 \\ I_{v-m} \wedge \overline{I_{v-m-1}} & \textit{if } m < v < C\text{-}1,\ m > 0 \\ \overline{I_{\lfloor C/2 \rfloor - 1} \vee I_0} & \textit{if } v = C\text{-}1 \end{cases}$$

การสอบถามแบบค่าเท่ากันบนดัชนีบิตแมปแบบช่วง จากภาพประกอบ 2-20 ต้องการทราบว่าบนแอตทริบิวต์ X มีแถวใดบ้างที่มีค่าเท่ากับ 2 ซึ่งตรงกับเงื่อนไขที่ 4 ($v < m$ นั่นคือ 2 < 7) จะได้ว่าคำตอบสามารถหาได้จาก “X = 2” เท่ากับ $I_2 \wedge \overline{I_3}$ ซึ่งแบ่งเป็นขั้นตอนเพื่อหาคำตอบได้ดังนี้

1) อ่านบิตแมปเวกเตอร์ I_2
2) อ่านบิตแมปเวกเตอร์ I_3
3) ดำเนินการตรรกะ NOT ทุกบิตบนบิตแมปเวกเตอร์
4) ดำเนินการตรรกะ AND บิตต่อบิตระหว่างบิตแมปเวกเตอร์ผลลัพธ์ที่ได้จากของ 1) และ 3)

5) คำตอบที่ได้ พิจารณาจากบิตแมปเวกเตอร์ผลลัพธ์ที่ได้จากข้อ 4) พบว่าบิตแถวที่ 1 และ 7 มีค่าเท่ากับ 1 ส่วนบิตอื่นๆ มีค่าเท่ากับ 0 จึงได้ว่าแถวที่มีค่าของแอตทริบิวต์เท่ากับ 2 คือแถวที่ 1 และแถวที่ 7

สำหรับการสอบถามแบบสมาชิกบนดัชนีบิตแมปแบบช่วง สามารถหาได้โดยการหาบิตแมปเวกเตอร์ผลลัพธ์ของแต่ละค่า แล้วนำมาดำเนินตรรกะ OR เช่น จากภาพประกอบ 2-20 ต้องการทราบว่าบนแอตทริบิวต์ X มีแถวใดบ้างที่มีค่าเท่ากับ {0, 3, 10, 11} ผลลัพธ์ของการสอบถามคือ $(I_0 \wedge \overline{I_1}) + (I_3 \wedge \overline{I_4}) + (I_3 \wedge \overline{I_2}) + (I_4 \wedge \overline{I_3})$ นั่นคือแถวที่ 5, 6, 9 และ 12 เป็นคำตอบของการสอบถาม

ข้อดีของดัชนีบิตแมปแบบช่วง

การทำดัชนีบิตแมปแบบช่วงใช้พื้นที่ในการจัดเก็บดัชนีน้อยกว่าการทำดัชนีบิตแมปแบบพื้นฐานมาก กล่าวคือสามารถลดพื้นที่ในการจัดเก็บดัชนีได้ครึ่งหนึ่ง นั่นคือ $\lceil C/2 \rceil$ และในการสอบถามแบบค่าเท่ากันใช้การดำเนินการตรรกะระหว่างบิตแมปเวกเตอร์ 2 บิตแมปเวกเตอร์ และมีโอกาสตรวจสอบเพียง 1 บิตแมปเวกเตอร์ เมื่อแอตทริบิวต์มีคาร์ดินอลิตี้น้อยกว่าหรือเท่ากับ 3

ข้อจำกัดของดัชนีบิตแมปแบบช่วง

ในการสอบถามข้อมูลแบบค่าเท่ากันและแบบสมาชิกของดัชนีบิตแมปแบบช่วงยังด้อยประสิทธิภาพกว่าดัชนีบิตแมปแบบพื้นฐาน เมื่อแอตทริบิวต์มีคาร์ดินอลิตี้มากกว่า 3 เนื่องจากการสอบถามแต่ละค่าบนดัชนีบิตแมปแบบช่วงใช้การดำเนินการตรรกะระหว่าง 2 บิตแมปเวกเตอร์ แต่การสอบถามแต่ละค่าบนดัชนีบิตแมปแบบพื้นฐานตรวจสอบเพียง 1 บิตแมปเวกเตอร์เท่านั้น

2.5.3.3 ดัชนีบิตแมปแบบกระจาย (Scatter Bitmap Index)

การทำดัชนีบิตแมปแบบกระจาย (Vanichayobon *et al.*, 2006; Weahama *et al.*, 2009) บนแอตทริบิวต์ที่มีคาร์ดินอลิตี้เท่ากับ C ประกอบด้วย $\lceil 2\sqrt{C} \rceil$ บิตแมปเวกเตอร์ แบ่งเป็น 2 กลุ่มคือ กลุ่มบิตแมปเวกเตอร์ Z (มี $\left\lceil \frac{C}{m-1} \right\rceil + 1$ บิตแมปเวกเตอร์) และกลุ่มบิตแมปเวกเตอร์ L (มี $\lceil m-2 \rceil$ บิตแมปเวกเตอร์) เมื่อ $m = \lceil \sqrt{C} \rceil + 1$ โดยแต่ละค่าของแอตทริบิวต์จะถูกแทนด้วยบิตแมปเวกเตอร์ที่อยู่ในกลุ่ม Z และกลุ่ม L หรืออาจจะถูกแทนด้วยบิตแมปเวกเตอร์ในอยู่กลุ่ม Z เพียงกลุ่มเดียว

RID	X
1	2
2	1
3	7
4	4
5	10
6	3
7	2
8	14
9	0
10	15
11	5
12	11
13	7
14	4

(ก) แอตทริบิวต์ X

Z_0	Z_1	Z_2	Z_3	Z_4	L_1	L_2	L_3
0	1	0	0	0	0	1	0
0	1	0	0	0	1	0	0
0	0	1	0	0	0	0	1
0	1	1	0	0	0	0	0
0	0	0	1	0	0	1	0
0	1	0	0	0	0	0	1
0	1	0	0	0	0	1	0
0	0	0	0	1	0	1	0
1	1	0	0	0	0	0	0
0	0	0	0	1	0	0	1
0	0	1	0	0	1	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	0	1
0	1	1	0	0	0	0	0

(ข) ดัชนีบิตแมปแบบกระจายบนแอตทริบิวต์ X

Z_1	L_2
1	1
1	0
0	0
1	0
0	1
1	0
1	1
0	1
1	0
0	0
0	0
0	0
0	0
1	0

(ค) การสอบถาม “X = 2”

$Z_1 \wedge L_2$
1
0
0
0
0
0
1
0
0
0
0
0
0
0

ภาพประกอบ 2-21 ดัชนีบิตแมปแบบกระจายบนแอตทริบิวต์ X เมื่อ C = 16

การสร้างบิตแมปเวกเตอร์กลุ่ม Z และกลุ่ม L สามารถทำได้ดังนี้ เมื่อกำหนด C คือคาร์ดินอลิตี้ของแอตทริบิวต์ที่นำมาสร้างดัชนี และ m คือ จำนวนสมาชิกภายในแต่ละกลุ่ม Z

การสร้างบิตแมปเวกเตอร์กลุ่ม Z

1) สร้างบิตแมปเวกเตอร์ Z_0 โดยกำหนดให้ บิตที่ i ของบิตแมปเวกเตอร์ Z_0 มีค่าเท่ากับ 1 ก็ต่อเมื่อ แถวที่ i มีค่าของแอตทริบิวต์เท่ากับ 0 ส่วนบิตอื่นๆ ให้มีค่าเท่ากับ 0

2) สร้างบิตแมปเวกเตอร์ Z_j โดยกำหนดให้ บิตที่ i ของบิตแมปเวกเตอร์ Z_j มีค่าเท่ากับ 1 ก็ต่อเมื่อ $j = \left\lfloor \frac{v}{m-1} \right\rfloor + 1$ โดยที่แถวที่ i มีค่าของแอตทริบิวต์เท่ากับ v ส่วนบิตอื่นๆ ให้มีค่าเท่ากับ 0 และถ้า $m-1$ หาร v ลงตัวแล้วจะกำหนดให้บิตที่ i ของบิตแมปเวกเตอร์ Z_{j-1} มีค่าเท่ากับ 1 ด้วย ส่วนบิตอื่นๆ ให้มีค่าเท่ากับ 0

การสร้างบิตแมปเวกเตอร์กลุ่ม L

1) กำหนดให้ บิตที่ i ของบิตแมปเวกเตอร์ L_k มีค่าเท่ากับ 1 ก็ต่อเมื่อ แถวที่ i มีค่าของแอตทริบิวต์ที่นำมาทำดัชนีเป็น v โดยที่ $k = v \, mod \, (m-1), v \in \{0,1,\dots,C-1\}$ และ $k \neq 0$

ภาพประกอบ 2-21(ก) แอตทริบิวต์ X มีค่าที่เป็นไปได้ 16 ค่า คือ 0, 1, 2, …, 15 ดังนั้นดัชนีบิตแมปแบบช่วงประกอบด้วย $\lceil 2\sqrt{16} \rceil$ = 8 บิตแมปเวกเตอร์ประกอบด้วย

บิตแมปเวกเตอร์กลุ่ม Z มี 5 $(\left\lceil\frac{16}{5-1}\right\rceil+1, m=\lceil\sqrt{16}\rceil+1=5)$ บิตแมปเวกเตอร์ คือ Z_0, Z_1, Z_2, Z_3, Z_4 และบิตแมปเวกเตอร์กลุ่ม L มี 3 (=5-2) บิตแมปเวกเตอร์ คือ L_1, L_2, L_3

สำหรับการสอบถามข้อมูลแบบค่าเท่ากันบนดัชนีบิตแมปแบบกระจายพิจารณาได้ดังนี้

$$"X=v" = \begin{cases} Z_{\frac{v}{(m-1)}} \wedge Z_{\frac{v}{(m-1)}+1} & \textit{If v mod (m-1) =0} \\ Z_{\frac{v}{(m-1)}+1} \wedge L_{v \bmod (m-1)} & \textit{Otherwise} \end{cases}$$

จากภาพประกอบ 2-21(ค) ต้องการทราบว่า บนแอตทริบิวต์ X มีแถวใดบ้างที่มีค่าเท่ากับ 2 ซึ่งตรงกับเงื่อนไขที่ 2 (2 mod (5-1) ≠ 0) ดังนั้นคำตอบสามารถหาได้จาก

$$\begin{aligned} "X=2" &= Z_{\frac{v}{(m-1)}+1} \wedge L_{v \bmod (m-1)} \\ &= Z_{\frac{2}{(5-1)}+1} \wedge L_{2 \bmod (5-1)} \\ &= Z_1 \wedge L_2 \end{aligned}$$

ซึ่งแบ่งเป็นขั้นตอนหาคำตอบดังนี้

1) อ่านบิตแมปเวกเตอร์ Z_1

2) อ่านบิตแมปเวกเตอร์ L_2

3) ดำเนินการตรรกะ AND บิตต่อบิตระหว่างบิตแมปเวกเตอร์ผลลัพธ์ที่ได้จากข้อ 1) และ 2)

4) คำตอบที่ได้ พิจารณาจากบิตแมปเวกเตอร์ผลลัพธ์ที่ได้จากข้อ 3) พบว่าบิตที่ 1 และบิตที่ 7 มีค่าเท่ากับ 1 ส่วนบิตอื่นๆ มีค่าเท่ากับ 0 จึงได้ว่า แถวที่มีค่าของแอตทริบิวต์เป็น 2 คือแถวที่ 1 และแถวที่ 7

สำหรับการสอบถามแบบสมาชิกบนดัชนีบิตแมปแบบกระจายสามารถทำได้โดยการหาบิตแมปเวกเตอร์ผลลัพธ์ของแต่ละค่ามาดำเนินการตรรกะ OR เช่น จากภาพประกอบ 2-21 ต้องการทราบว่าบนแอตทริบิวต์ X มีแถวใดบ้างที่มีค่าเท่ากับ {0, 3, 10, 11} ผลลัพธ์ของการสอบถามคือ $(Z_0 \wedge Z_1)+(Z_1 \wedge L_3)+(Z_3 \wedge L_2)+(Z_3 \wedge L_3)$ นั่นคือแถวที่ 5, 6, 9 และ 12 เป็นคำตอบของการสอบถาม

ข้อดีของดัชนีบิตแมปแบบกระจาย

การทำดัชนีบิตแมปแบบกระจายใช้พื้นที่ในการจัดเก็บดัชนีน้อยกว่าการทำดัชนีบิตแมปแบบแบ่งช่วง การสอบถามข้อมูลแบบค่าเท่ากันของดัชนีบิตแมปแบบกระจายใช้การดำเนินการตรรกะระหว่าง 2 บิตแมปเวกเตอร์เช่นเดียวกับดัชนีบิตแมปแบบแบ่งช่วง และการสอบถามแบบสมาชิกของดัชนีบิตแมปแบบกระจายมีโอกาสตรวจสอบเพียง 1 บิตแมปเวกเตอร์เมื่อสมาชิกที่ต้องการสอบถามอยู่ในกลุ่ม Z หรือ L เดียวกัน

ข้อจำกัดของดัชนีบิตแมปแบบกระจาย

การสอบถามข้อมูลแบบค่าเท่ากันและแบบสมาชิกของดัชนีบิตแมปแบบกระจายด้อยประสิทธิภาพกว่าดัชนีบิตแมปแบบพื้นฐาน เนื่องจากการสอบถามแต่ละค่าบนดัชนีบิตแมปแบบกระจาย ใช้การดำเนินการตรรกะระหว่าง 2 บิตแมปเวกเตอร์ แต่การสอบถามแต่ละค่าบนดัชนีบิตแมปแบบพื้นฐาน ตรวจสอบเพียง 1 บิตแมปเวกเตอร์เท่านั้น

2.5.3.4 ดัชนีบิตแมปแบบคู่กัน (Dual Bitmap Index)

การทำดัชนีบิตแมปแบบคู่กัน (Wattanakitrungroj and Vanichayobon, 2006) บนแอตทริบิวต์ที่มีคาร์ดินอลิตี้เท่ากับ C ประกอบด้วย $\lceil\sqrt{2C + 0.25} + 0.5\rceil$ บิตแมปเวกเตอร์ คือ $D_0, D_1, D_2, \ldots, D_{\lceil\sqrt{2C+0.25}+0.5\rceil-1}$ ในการแทนค่าข้อมูลบนแอตทริบิวต์จะกำหนดให้บิตที่ i ของบิตแมปเวกเตอร์ D_j มีค่าเท่ากับ 1 เมื่อ $j = r$ หรือ $j = s$ หรือ โดยที่ $r = \left\lceil\sqrt{2(hiC - v) + 0.24} - 0.5\right\rceil$ และ $s = \left\lceil r - 1 - \left[\left(v - \frac{(n-r)(n-r-1)}{2}\right) mod\ r\right]\right\rceil$ เมื่อ v คือค่าข้อมูลในแถวที่ i บนแอตทริบิวต์ที่ทำดัชนี และ $hiC = \binom{n}{2}$ เมื่อ n คือจำนวนบิตแมปเวกเตอร์

RID	X
1	2
2	1
3	7
4	4
5	10
6	3
7	2
8	14
9	0
10	15
11	5
12	11
13	7
14	4

(ก) แอตทริบิวต์ X

D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	0	0	0
1	0	1	0	0	0	0
0	1	0	1	0	0	0
1	0	0	0	0	1	0
0	1	0	0	0	0	1
1	0	0	0	1	0	0
1	0	0	1	0	0	0
0	0	1	0	0	0	1
1	1	0	0	0	0	0
0	0	0	1	1	0	0
1	0	0	0	0	0	1
0	0	1	1	0	0	0
0	1	0	1	0	0	0
1	0	0	0	0	1	0

(ข) ดัชนีบิตแมปแบบคู่กันบนแอตทริบิวต์ X

D_3	D_6	$D_3 \wedge D_6$
1	1	1
0	1	0
1	0	0
0	1	0
0	0	0
0	1	0
1	1	1
0	0	0
0	1	0
1	0	0
0	1	0
1	0	0
1	0	0
0	1	0

(ค) การสอบถาม “X = 2”

ภาพประกอบ 2-22 ดัชนีบิตแมปแบบคู่กันบนแอตทริบิวต์ X เมื่อ C = 16

จากภาพประกอบ 2-22(ก) แอตทริบิวต์ X มีค่าที่เป็นไปได้ 16 ค่า คือ 0, 1, 2, …, 15 ดังนั้นดัชนีบิตแมปแบบช่วงประกอบด้วย 7 (=$\lceil\sqrt{2(16)+0.25}+0.5\rceil$) บิตแมปเวกเตอร์ คือ $D_0, D_1, \ldots, D_6$ ซึ่งมีรูปแบบการลงรหัส แสดงดังภาพประกอบ 2-22(ข)

สำหรับการสอบถามแบบค่าเท่ากันบนดัชนีบิตแมปแบบคู่กันพิจารณาได้ดังนี้

"$X = v$" $= D_r \wedge D_s$

เมื่อ $r = \left\lceil\sqrt{2(hiC - v) + 0.24} - 0.5\right\rceil, hiC = \binom{n}{2}$

และ $s = \left\lceil r - 1 - \left[\left(v - \frac{(n-r)(n-r-1)}{2}\right) mod\ r\right]\right\rceil$

จากภาพประกอบ 2-22(ค) ต้องการทราบว่า บนแอตทริบิวต์ A มีแถวใดบ้างที่มีค่าข้อมูลเท่ากับ 2 จะได้ว่าคำตอบสามารถหาได้จาก "$X = 2$" เท่ากับ $D_6 \wedge D_3$ ซึ่งแบ่งเป็นขั้นตอนเพื่อหาคำตอบได้ดังนี้

1) อ่านบิตแมปเวกเตอร์ D_6

2) อ่านบิตแมกเวกเตอร์ D_3

3) ดำเนินการตรรกะ AND บิตต่อบิตระหว่างบิตแมปเวกเตอร์ผลลัพธ์ที่ได้จากข้อ 1) และ 2)

4) คำตอบที่ได้ พิจารณาจากบิตแมปเวกเตอร์ผลลัพธ์ที่ได้จากข้อ 3) พบว่าบิตที่ 1 และบิตที่ 7 มีค่าเท่ากับ 1 ส่วนบิตอื่นๆ มีค่าเท่ากับ 0 จึงได้ว่า แถวที่มีค่าของแอตทริบิวต์เป็น 2 คือแถวที่ 1 และแถวที่ 7

สำหรับการสอบถามข้อมูลแบบสมาชิกบนดัชนีบิตแมปแบบคู่กันสามารถทำได้โดยหาบิตแมปเวกเตอร์ผลลัพธ์ของแต่ละค่ามาดำเนินการตรรกะ OR เช่น จากภาพประกอบ 2-22 ต้องการทราบว่าบนแอตทริบิวต์ X มีแถวใดบ้างที่มีค่าเท่ากับ {0, 3, 10, 11} ผลลัพธ์ของการสอบถามคือ $(D_6 \wedge D_5) + (D_6 \wedge D_2) + (D_5 \wedge D_0) + (D_4 \wedge D_3)$ นั่นคือแถวที่ 5, 6, 9 และ 12 เป็นคำตอบของการสอบถาม

ข้อดีของดัชนีบิตแมปแบบคู่กัน

การทำดัชนีบิตแมปแบบคู่กันใช้พื้นที่ในการจัดเก็บดัชนีน้อยกว่าการทำดัชนีบิตแมปแบบช่วงและแบบกระจาย และการสอบถามข้อมูลแบบค่าเท่ากันของดัชนีบิตแมปแบบคู่กันยังคงใช้การดำเนินการตรรกะระหว่าง 2 บิตแมปเวกเตอร์เหมือนกับดัชนีบิตแมปแบบแบ่งช่วงและแบบกระจาย

ข้อจำกัดของดัชนีบิตแมปแบบคู่กัน

การสอบถามข้อมูลแบบค่าเท่ากันและแบบสมาชิกของดัชนีบิตแมปแบบคู่กันด้อยประสิทธิภาพกว่าดัชนีบิตแมปแบบพื้นฐาน เนื่องจากการสอบถามแต่ละค่าบนดัชนีบิตแมป

แบบคู่กัน ใช้การดำเนินการตรรกะระหว่าง 2 บิตแมปเวกเตอร์ แต่การสอบถามแต่ละค่าบนดัชนีบิตแมปแบบพื้นฐาน ตรวจสอบเพียง 1 บิตแมปเวกเตอร์เท่านั้น

2.5.3.5 ดัชนีบิตแมปแบบเข้ารหัส (Encoded Bitmap Index)

ดัชนีบิตแมปแบบเข้ารหัส (Wu and Buchmann, 1998; Sainui *et al.*, 2008) เป็นเทคนิคการทำดัชนีบิตแมปที่ใช้เนื้อที่น้อยที่สุด ซึ่งการทำดัชนีบิตแมปแบบเข้ารหัสบนแอตทริบิวต์ที่มีคาร์ดินอลิตี้เท่ากับ C ประกอบด้วย $\lceil \log_2 C \rceil$ บิตแมปเวกเตอร์ คือ $E_{\lceil \log_2 C \rceil - 1}, \ldots, E_2, E_1, E_0$ โดยการลงรหัสข้อมูลใช้วิธีการเทียบค่าจากตารางการเทียบค่า (Mapping Table) เพื่อกำหนดให้กับแต่ละบิตแมปเวกเตอร์ โดยในการแทนค่า บิตที่ i ของบิตแมปเวกเตอร์จะมีค่าเท่ากับ 0 หรือ 1 ขึ้นอยู่กับค่าข้อมูลของแอตทริบิวต์แถวที่ i ว่าเทียบค่าของแต่ละบิตแมปเวกเตอร์ตรงกับค่าใด

RID	X
1	2
2	1
3	7
4	4
5	10
6	3
7	2
8	14
9	0
10	15
11	5
12	11
13	7
14	4

(ก) แอตทริบิวต์ X

E_3	E_2	E_1	E_0
0	0	1	0
0	0	0	1
0	1	1	1
0	1	0	0
1	0	1	0
0	0	1	1
0	0	1	0
1	1	1	0
0	0	0	0
1	1	1	1
0	1	0	1
1	0	1	1
0	1	1	1
0	1	0	0

(ข) ดัชนีบิตแมปแบบเข้ารหัสบนแอตทริบิวต์ X

Mapping Table

0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

(ค) ตารางเทียบค่า

ภาพประกอบ 2-23 ดัชนีบิตแมปแบบเข้ารหัสบนแอตทริบิวต์ X เมื่อ C = 16

จากภาพประกอบ 2-23(ก) แอตทริบิวต์ X มีค่าที่เป็นไปได้ 16 ค่า คือ 0,1,2,…,15 ดังนั้นดัชนีบิตแมปแบบเข้ารหัสประกอบด้วย $\lceil log_2 16 \rceil$ = 4 บิตแมปเวกเตอร์ คือ

E_3, E_2, E_1, E_0 แต่ละค่าของแอตทริบิวต์ถูกแทนด้วย 4 บิตแมปเวกเตอร์ ซึ่งมีรูปแบบการลงรหัสแสดงดังภาพประกอบ 2-23(ข)

สำหรับการสอบถามแบบค่าเท่ากันบนดัชนีบิตแมปแบบเข้ารหัสสามารถทำได้โดยการนำค่าข้อมูลที่ต้องการสอบถามไปเทียบในตารางเทียบการค่าว่าแต่ละบิตแมปเวกเตอร์ถูกเข้ารหัสแบบใด จากนั้นนำไปตรวจสอบกับทุกๆ แถวบนดัชนี ถ้าแถวใดมีค่าที่ตรงกันกับค่าที่ได้จากตารางเทียบการค่าทุกบิตแมปเวกเตอร์แล้วแถวนั้นคือคำตอบ ตัวอย่างเช่น จากภาพประกอบ 2-23 ต้องการทราบว่าบน แอตทริบิวต์ X แถวใดบ้างมีค่าเท่ากับ 2 สามารถทำได้โดยการอ่านค่าการเข้ารหัสจากตารางเทียบค่า ซึ่งมีการเข้ารหัสเป็น 0010 ($E_3E_2E_1E_0$) จากนั้นทำกันอ่านบิตแมปเวกเตอร์เข้ามาเพื่อทำการตรวจสอบทุกแถวบนดัชนีที่มีบิตแมปเวกเตอร์ E_1 มีค่าเท่ากับ 1 และบิตแมปเวกเตอร์ E_3, E_2 และ E_0 มีค่าเท่ากับ 0 จะได้แถวที่ 1 และแถวที่ 7 เป็นคำตอบ

สำหรับการสอบถามแบบสมาชิกบนดัชนีบิตแมปแบบเข้ารหัสสามารถทำได้โดยการนำรูปแบบการลงรหัสของแต่ละค่ามาดำเนินการตรรกะ OR จากนั้นจะทำการลดรูปฟังก์ชันที่หาได้เพื่อลดจำนวนบิตแมปเวกเตอร์ที่ต้องตรวจสอบ ตัวอย่างเช่น จากภาพประกอบ 2-23 ต้องการทราบว่าบนแอตทริบิวต์ X มีแถวใดบ้างที่มีค่าเท่ากับ {0, 3, 10, 11} โดยใช้ตารางเทียบค่า จะได้ฟังก์ชันเป็น $E_3'E_2'E_1'E_0' + E_3'E_2'E_1E_0 + E_3E_2'E_1E_0' + E_3E_2'E_1E_0$ จะเห็นได้ว่าไม่สามารถลดรูปฟังก์ชันได้ จึงต้องตรวจสอบทุกบิตแมปเวกเตอร์ของแต่ละค่า นั่นคือแถวที่ 5, 6, 9 และ 12 เป็นคำตอบ

อย่างไรก็ตามถ้าเรามีตารางเทียบค่าดังภาพประกอบ 2-24 หากเปรียบเทียบกับการสอบถามที่เหมือนกัน จะได้ฟังก์ชันการเข้าถึงข้อมูลเป็น $E_3'E_2'E_1'E_0' + E_3'E_2'E_1'E_0 + E_3'E_2'E_1E_0' + E_3'E_2'E_1E_0$ ซึ่งสามารถลดรูปฟังก์ชันการเข้าถึงข้อมูลได้เป็น $E_3'E_2'$ เพราะฉะนั้นตรวจสอบเพียงบิตแมปเวกเตอร์ E_3' และ E_2' นั่นคือตรวจสอบบิตแมปเวกเตอร์ E_3' ที่มีค่าเท่ากับ 0 และตรวจสอบบิตแมปเวกเตอร์ E_2' ที่มีค่าเท่ากับ 0 ก็สามารถได้คำตาอบของการสอบถาม

RID	X
1	2
2	1
3	7
4	4
5	10
6	3
7	2
8	14
9	0
10	15
11	5
12	11
13	7
14	4

(ก) แอตทริบิวต์ X

E_3	E_2	E_1	E_0
1	0	1	1
1	0	1	0
1	1	1	0
0	1	1	0
0	0	1	0
0	0	0	1
1	0	1	1
0	1	1	1
0	0	0	0
1	1	1	1
0	1	0	1
0	0	1	1
1	1	1	0
0	1	1	0

(ข) ดัชนีบิตแมปแบบเข้ารหัสบนแอตทริบิวต์ X

Mapping Table	
0	0000
1	1010
2	1011
3	0001
4	0110
5	0101
6	0100
7	1110
8	1000
9	1001
10	0010
11	0011
12	1100
13	1101
14	0111
15	1111

(ค) ตารางเทียบค่า

ภาพประกอบ 2-24 ดัชนีบิตแมปแบบเข้ารหัสบนแอตทริบิวต์ X โดยใช้เทคนิคการจัดกลุ่มข้อมูล

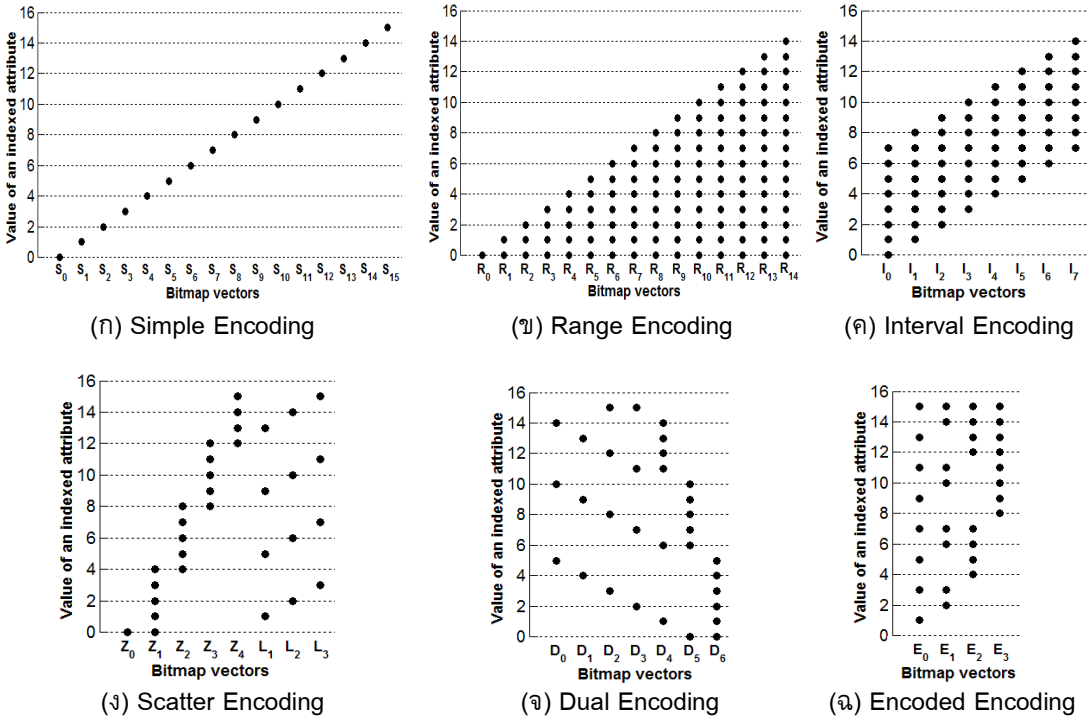
ข้อดีของดัชนีบิตแมปแบบเข้ารหัส

การทำดัชนีบิตแมปแบบเข้ารหัสมุ่งเน้นไปที่การใช้พื้นที่ให้มีประสิทธิภาพสูงสุด กล่าวคือ การทำดัชนีบิตแมปแบบเข้ารหัสใช้พื้นที่น้อยที่สุด เนื่องจากดัชนีบิตแมปแบบเข้ารหัสใช้จำนวนบิตแมปเวกเตอร์ในการสร้างดัชนีน้อยกว่าดัชนีบิตแมปแบบพื้นฐาน ดัชนีบิตแมปแบบแถว ดัชนีบิตแมปแบบช่วง ดัชนีบิตแมปแบบกระจาย และดัชนีบิตแมปแบบคู่กัน นอกจากนี้หากดัชนีบิตแมปแบบเข้ารหัสมีการเข้ารหัสที่ดีจะทำให้มีประสิทธิภาพในการสอบถามข้อมูลแบบสมาชิกด้วย

ข้อจำกัดของดัชนีบิตแมปแบบเข้ารหัส

ถึงแม้ว่าการทำดัชนีบิตแมปแบบเข้ารหัสจะสามารถลดพื้นที่ในการจัดเก็บดัชนีได้มาก แต่ดัชนีบิตแมปแบบเข้ารหัสไม่เหมาะกับการสอบถามข้อมูลแบบค่ากัน เนื่องจากต้องมีการเปรียบเทียบกับทุกบิตของบิตแมปเวกเตอร์ และหากดัชนีบิตแมปแบบเข้ารหัสมีรูปแบบการเข้ารหัสที่ไม่ดีจะส่งผลให้ประสิทธิภาพการสอบถามข้อมูลแบบสมาชิกลดลง ดังนั้นหากดัชนีบิตแมปแบบเข้ารหัสมีรูปแบบการเข้ารหัสที่ดีจะสามารถลดจำนวนตัวดำเนินการตรรกะและ

บิตแมปเวกเตอร์ที่ใช้สอบถามข้อมูล โดยไม่ต้องตรวจสอบทุกบิตแมปเวกเตอร์ จะทำให้เวลาที่ใช้ในการสอบถามน้อยลง



(ก) Simple Encoding (ข) Range Encoding (ค) Interval Encoding

(ง) Scatter Encoding (จ) Dual Encoding (ฉ) Encoded Encoding

ภาพประกอบ 2-25 รูปแบบการลงรหัสของดัชนีบิตแมปทั้ง 6 แบบ เมื่อ C = 16

2.6 เปรียบเทียบการขยายแนวคิดการทำดัชนีบิตแมปทั้ง 6 แบบ

จากที่กล่าวมาข้างต้น ดัชนีบิตแมปแต่ละแบบมีรูปแบบการลงรหัสและการสอบถามข้อมูลที่แตกต่างกันจึงมีข้อเด่นและข้อจำกัดแตกต่างกันออกไปด้วย พิจารณารูปแบบการลงรหัสที่แตกต่างกันของดัชนีบิตแมปแต่ละแบบ ดังภาพประกอบ 2-25 จากรูปแบบการลงรหัสและการสอบถามข้อมูลแบบสมาชิกและการสอบถามข้อมูลแบบค่าเท่ากันของดัชนีบิตแมปแต่ละแบบ สามารถสรุปความต้องการพื้นที่ที่ใช้ในการจัดเก็บดัชนีและเวลาที่ใช้ในการสอบถามข้อมูลทั้งแบบสมาชิกและแบบค่าเท่ากันของดัชนีบิตแมปแต่ละแบบแสดงดังตาราง 2-2 โดยพื้นที่ที่ใช้ในการจัดเก็บดัชนี พิจารณาจากจำนวนบิตแมปเวกเตอร์ที่ใช้ในการสร้างดัชนีและเวลาที่ใช้ในการสอบถามข้อมูลทั้งแบบสมาชิกและแบบค่าเท่ากัน พิจารณาจากจำนวนบิตแมปเวกเตอร์ที่ถูกตรวจสอบและจำนวนครั้งในการดำเนินการตรรกะ

ตาราง 2-2 เปรียบเทียบประสิทธิภาพของดัชนีบิตแมปทั้ง 6 แบบ (เมื่อ C คือ คาร์ดินอลิตี้ และ k คือ จำนวนสมาชิกที่ถูกสอบถาม)

ดัชนีบิตแมป	**พื้นที่**	**เวลา**		
	จำนวนบิตแมปเวกเตอร์ที่ใช้ในการสร้างดัชนี	จำนวนบิตแมปเวกเตอร์ที่ถูกตรวจสอบ : จำนวนครั้งในการดำเนินการตรรกะ		
		การสอบถามแบบสมาชิก		การสอบถามแบบค่าเท่ากัน
		best case	worse case	
แบบพื้นฐาน	C	k:k-1 (OR)		1:0
แบบแถว	$C-1$	1:0	2k:2k-1 (k XOR, k-1 OR)	2:1 (1 XOR)
แบบช่วง	$\lceil C/2 \rceil$	1:0	2k:3k-1 (k AND, k-1 OR, k NOT)	2:2 (1 AND, 1 NOT)
แบบกระจาย	$\lceil 2\sqrt{C} \rceil$	1:0	2k:2k-1 (k AND, k-1 OR)	2:1 (1 AND)
แบบคู่กัน	$\lceil \sqrt{2C+0.25}+0.5 \rceil$	1:0	2k:2k-1 (k AND, k-1 OR)	2:1 (1 AND)
แบบเข้ารหัส	$\lceil log_2 C \rceil$	1:0	k $\lceil log_2 C \rceil$: use mapping table (k-1 OR)	$\lceil log_2 C \rceil$: use mapping table

จากตาราง 2-2 เมื่อพิจารณาพื้นที่ที่ใช้ในการจัดเก็บดัชนีจะเห็นได้ว่าดัชนีบิตแมปแบบเข้ารหัสใช้พื้นที่ในการจัดเก็บดัชนีน้อยที่สุด คือ ใช้ $\lceil log_2 C \rceil$ บิตแมปเวกเตอร์ ส่วนดัชนีบิตแมปแบบพื้นฐานใช้พื้นที่ในการจัดเก็บดัชนีมากที่สุด ซึ่งใช้ C บิตแมปเวกเตอร์ และเมื่อพิจารณาเวลาที่ใช้ในการสอบถามข้อมูลแบบสมาชิก ในกรณีที่ดีที่สุดดัชนีบิตแมปแบบแถว ดัชนีบิตแมปแบบช่วง ดัชนีบิตแมปแบบกระจาย ดัชนีบิตแมปแบบคู่กัน และดัชนีบิตแมปแบบเข้ารหัสมีโอกาสใช้เวลาในการสอบถามข้อมูลน้อยกว่าดัชนีบิตแมปแบบพื้นฐาน ในกรณีที่แย่ที่สุดดัชนีบิตแมปแบบพื้นฐานใช้เวลาในการสอบถามข้อมูลน้อยที่สุด ส่วนดัชนีบิตแมปแบบเข้ารหัสใช้เวลาในการสอบถามข้อมูลมากที่สุด ดัชนีบิตแมปแบบแถว ดัชนีบิตแมปแบบช่วง ดัชนีบิตแมปแบบกระจาย และดัชนีบิตแมปแบบคู่กันใช้เวลาในการสอบถามใกล้เคียงกัน แต่ยังคงใช้เวลาในการสอบถามมากกว่าดัชนีบิตแมปแบบพื้นฐาน สำหรับการสอบถามข้อมูลแบบ

ค่าเท่ากันจะเห็นได้ว่า ดัชนีบิตแมปแบบพื้นฐานใช้เวลาน้อยที่สุด ส่วนดัชนีบิตแมปแบบเข้ารหัสใช้เวลามากที่สุด

จากการขยายแนวคิดของดัชนีบิตแมปตามที่ได้กล่าวมาจะเห็นได้ว่า ดัชนีบิตแมปแบบเข้ารหัสมีประสิทธิภาพในการใช้พื้นที่ในการจัดเก็บดัชนีมากที่สุด หากมีรูปแบบการลงรหัสที่ดีจะทำให้ประสิทธิภาพในการสอบถามข้อมูลแบบสมาชิกเพิ่มขึ้นด้วย ดังนั้นหากเราสามารถลงรหัสค่าของแอตทริบิวต์ที่มีการสอบถามไปด้วยกันและสอบถามร่วมกันให้อยู่ในรูปแบบที่สามารถลดจำนวนบิตแมปเวกเตอร์ที่ใช้ในการสอบถามข้อมูลได้ก็จะเป็นการเพิ่มประสิทธิภาพในการสอบถามข้อมูลแบบสมาชิกกล่าวคือ สามารถลดเวลาที่ใช้ในการสอบถามข้อมูลได้และหากมีการใช้ตัวดำเนินการตรรกะต้นทุนต่ำก็จะช่วยเพิ่มประสิทธิภาพการสอบถามข้อมูลแบบค่าเท่ากันด้วยเช่นกัน ซึ่งในการหาลำดับค่าของแอตทริบิวต์ที่มีการสอบถามไปด้วยกันและสอบถามร่วมกันเราสามารถนำเอาเทคนิคการจัดกลุ่มข้อมูล (Data Clustering) มาใช้ในการจัดกลุ่มค่าของแอตทริบิวต์ก่อนการหารูปแบบการลงรหัสซึ่งนำไปสู่การลดจำนวนบิตแมปเวกเตอร์เมื่อมีการสอบถามข้อมูลแบบสมาชิก

บทที่ 3

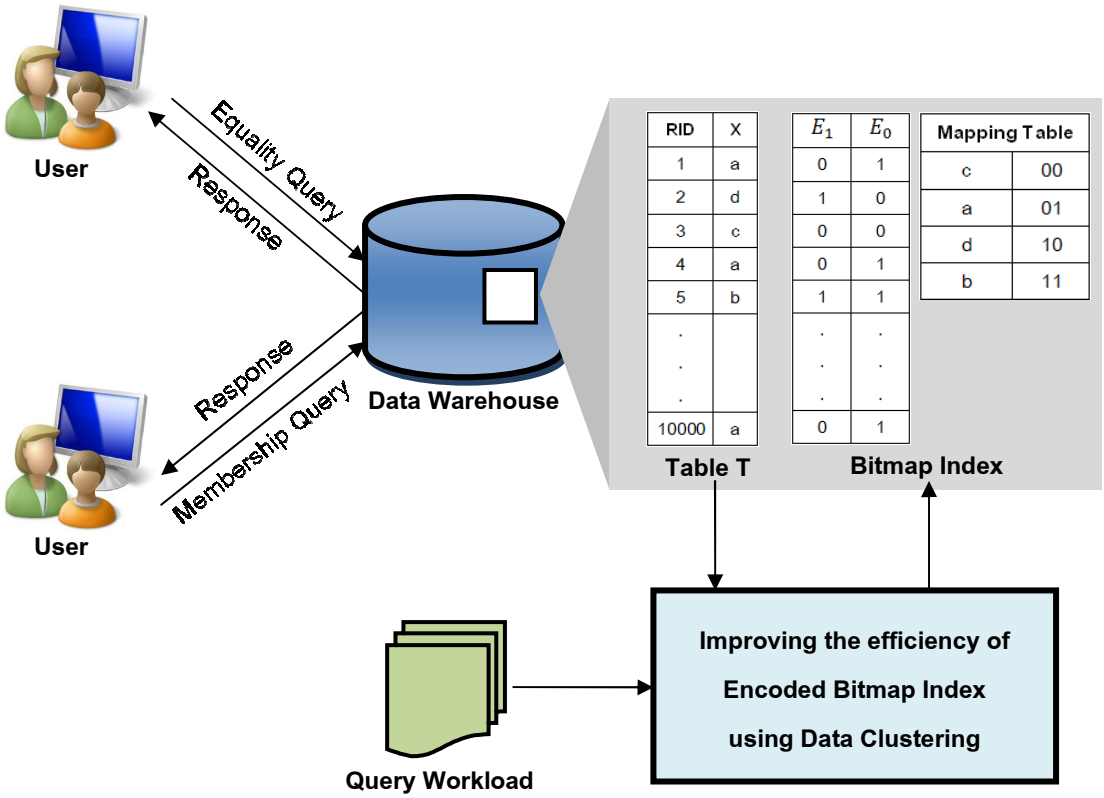
การเพิ่มประสิทธิภาพการสอบถามข้อมูลบนดัชนีบิตแมปแบบเข้ารหัส

บทที่ 2 อธิบายเกี่ยวกับเทคนิคการทำดัชนีบิตแมปแบบต่างๆ ในคลังข้อมูล ซึ่งงานวิจัยส่วนใหญ่มุ่งเน้นในเรื่องของการเพิ่มประสิทธิภาพด้านพื้นที่การจัดเก็บดัชนีและเวลาที่ใช้ในการสอบถามข้อมูล ซึ่งแยกออกเป็นการสอบถามแบบค่าเท่ากันและการสอบถามแบบสมาชิก กล่าวคือ สำหรับการสอบถามข้อมูลแบบสมาชิก ในกรณีที่แย่ที่สุด ดัชนีบิตแมปแบบช่วง ดัชนีบิตแมปแบบกระจาย และดัชนีบิตแมปแบบคู่กัน จะใช้เวลาในการสอบถามข้อมูลแบบสมาชิกใกล้เคียงกัน แต่ยังคงใช้เวลามากกว่าดัชนีบิตแมปแบบพื้นฐานและดัชนีบิตแมปแบบแบบแถว ส่วนดัชนีบิตแมปแบบเข้ารหัสใช้เวลาในการสอบถามข้อมูลแบบสมาชิกมากที่สุด เนื่องจากต้องตรวจสอบทุกบิตแมปเวกเตอร์และใช้ตารางเทียบค่าเพื่อหารูปแบบการลงรหัสของแต่ละค่า ดังนั้นรูปแบบการลงรหัสจึงมีความสำคัญอย่างมากในการเพิ่มประสิทธิภาพของการสอบถามแบบสมาชิกบนดัชนีบิตแมปแบบเข้ารหัส เพราะรูปแบบการลงรหัสที่เหมาะสมจะช่วยลดจำนวนบิตแมปเวกเตอร์ที่จะต้องตรวจสอบเมื่อมีการสอบถามแบบสมาชิกเข้ามา นั่นคือสามารถลดเวลาที่ใช้ในการสอบถามข้อมูลได้ (Wu and Buchmann, 1998; Sainui *et al.*, 2008) สำหรับการสอบถามข้อมูลแบบค่าเท่ากัน ดัชนีบิตแมปแบบพื้นฐาน สามารถสอบถามข้อมูลแบบค่าเท่ากันได้รวดเร็ว ส่วนดัชนีบิตแมปแบบแถว ดัชนีบิตแมปแบบช่วง ดัชนีบิตแมปแบบกระจาย และดัชนีบิตแมปแบบคู่กันใช้เวลาในการสอบถามข้อมูลแบบค่าเท่ากันมากกว่าดัชนีบิตแมปแบบพื้นฐาน แต่ใช้พื้นที่ในการจัดเก็บดัชนีน้อยกว่าดัชนีบิตแมปแบบพื้นฐาน (Wattanakitrungroj and Vanichayobon, 2006) ดัชนีบิตแมปแบบเข้ารหัสใช้เวลาในการสอบถามข้อมูลแบบค่าเท่ากันมากที่สุด ถึงแม้ว่าดัชนีบิตแมปแบบพื้นฐานใช้เวลาในการสอบถามข้อมูลแบบค่าเท่ากันดีที่สุด แต่ไม่เหมาะกับแอตทริบิวต์ที่มีคาร์ดินอลิตี้สูงๆ เพราะใช้พื้นที่ในการจัดเก็บดัชนีมาก (Chan and Ioannidis, 1999; Wu *et al.*, 2006; Stabno and Wrembel, 2009) ในทางตรงกันข้าม ดัชนีบิตแมปแบบเข้ารหัสใช้พื้นที่ในการจัดเก็บดัชนีมีประสิทธิภาพมากที่สุด

วิทยานิพนธ์นี้ได้เสนอการเพิ่มประสิทธิภาพการสอบถามข้อมูลแบบสมาชิกและการสอบถามข้อมูลแบบค่าเท่ากันของดัชนีบิตแมปแบบเข้ารหัสโดยการนำเทคนิคการจัดกลุ่ม (Data Clustering) มาช่วยในการหากลุ่มข้อมูลที่ถูกสอบถามร่วมกันบ่อยจากประวัติการสอบถามข้อมูลในอดีต (Query Workload) เพื่อหารูปแบบการลงรหัสของแต่ละค่าให้สามารถลดจำนวนบิตแมปเวกเตอร์ที่ต้องตรวจสอบ เมื่อมีการสอบถามแบบสมาชิก

ในบทนี้นำเสนอขั้นตอนการเพิ่มประสิทธิภาพของดัชนีบิตแมปแบบเข้ารหัสด้วยเทคนิคการจัดกลุ่มข้อมูล การสอบถามข้อมูลแบบสมาชิกบนดัชนีบิตแมปแบบเข้ารหัสที่ใช้

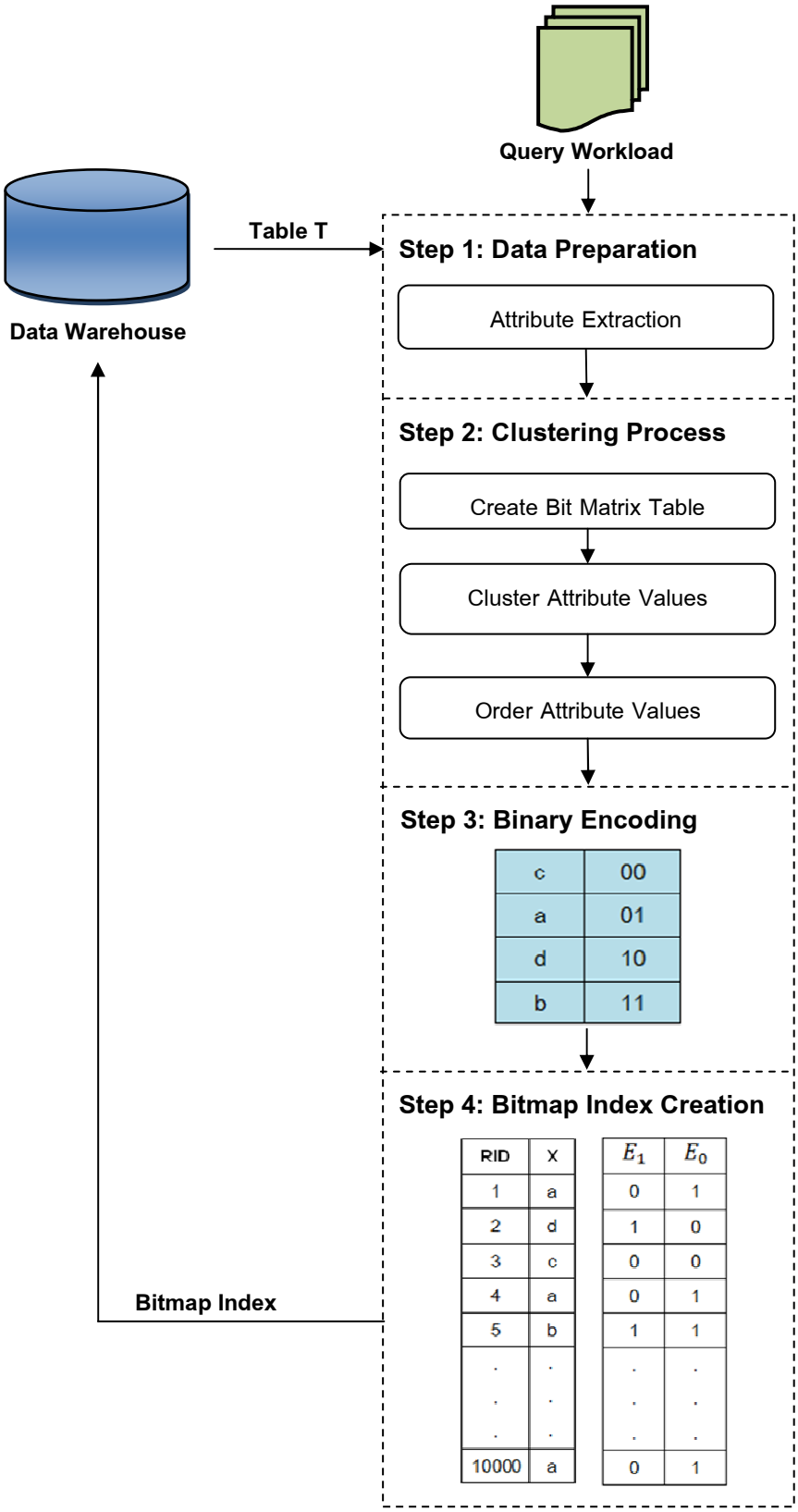
เทคนิคการจัดกลุ่มข้อมูล การสอบถามข้อมูลแบบค่าเท่ากันบนดัชนีบิตแมปแบบเข้ารหัสที่ใช้เทคนิคการจัดกลุ่มข้อมูล ข้อเด่นของดัชนีบิตแมปแบบเข้ารหัสที่ใช้เทคนิคการจัดกลุ่มข้อมูล และข้อจำกัดของดัชนีบิตแมปแบบเข้ารหัสที่ใช้เทคนิคการจัดกลุ่มข้อมูล สถาปัตยกรรมโดยรวมของระบบ แสดงดังภาพประกอบ 3-1



ภาพประกอบ 3-1 สถาปัตยกรรมโดยรวมของระบบ

3.1 การเพิ่มประสิทธิภาพของดัชนีบิตแมปแบบเข้ารหัสด้วยเทคนิคการจัดกลุ่มข้อมูล (Improving the Efficiency of Encoded Bitmap Index using Data Clustering)

ส่วนนี้เป็นการนำเสนอขั้นตอนการเพิ่มประสิทธิภาพของดัชนีบิตแมปแบบเข้ารหัสด้วยเทคนิคการจัดกลุ่มข้อมูลเพื่อช่วยจัดกลุ่มและหาลำดับค่าของแอตทริบิวต์เพื่อเพิ่มประสิทธิภาพในการสอบถามข้อมูลในคลังข้อมูลประกอบด้วย 4 ขั้นตอนหลักคือ 1) การเตรียมข้อมูล (Data Preparation) 2) กระบวนการจัดกลุ่มข้อมูล (Clustering Process) 3) การลงรหัสค่าของแอตทริบิวต์ (Binary Encoding) และ 4) การสร้างดัชนีบิตแมปแบบเข้ารหัส (Bitmap Index Creation) แสดงดังภาพประกอบ 3-2 โดยแต่ละขั้นตอนมีรายละเอียดดังต่อไปนี้



ภาพประกอบ 3-2 ขั้นตอนการเพิ่มประสิทธิภาพของดัชนีบิตแมปแบบเข้ารหัสโดยใช้เทคนิคการจัดกลุ่มข้อมูล

ขั้นตอนที่ 1: การเตรียมข้อมูล

ขั้นตอนการเตรียมข้อมูลเป็นขั้นตอนการสกัดค่าของแอตทริบิวต์ที่สนใจ (Attribute Extraction) ที่จะนำมาสร้างดัชนีบิตแมปแบบเข้ารหัสจาก Query Workload โดยการพิจารณาค่าที่สอบถามของแอตทริบิวต์ที่อยู่หลังเงื่อนไข Where เพื่อนำมาสร้างตารางค่าของแอตทริบิวต์ ตารางค่าของแอตทริบิวต์แสดงให้เห็นว่า แต่ละการสอบถามมีการสอบถามค่าของแอตทริบิวต์ค่าใดบ้าง ภาพประกอบ 3-3 แสดงตัวอย่าง Query Workload ของแอตทริบิวต์ X เมื่อกำหนดให้แอตทริบิวต์ X มีค่าที่เป็นไปได้ 16 ค่า คือ {A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P} และตารางค่าของแอตทริบิวต์ X ที่สกัดจาก Query Workload แสดงดังภาพประกอบ 3-4

```
Q1 :  SELECT * FROM T WHERE X IN {C, D, J, K, L, M, N P}
Q2 :  SELECT * FROM T WHERE X IN {C, E, J, K, P}
Q3 :  SELECT * FROM T WHERE X IN {A, B, C, D, F, G, H, I, J, K, L, M, N, O}
Q4 :  SELECT * FROM T WHERE X IN {A, B, D, F, G, J, K, L, M, N, P}
Q5 :  SELECT * FROM T WHERE X IN {A, B, C, J, K, P}
Q6 :  SELECT * FROM T WHERE X IN {A, B, C, F, G, H, I, P}
```

ภาพประกอบ 3-3 ตัวอย่าง Query Workload ของแอตทริบิวต์ X

Query	Value of Attribute X
Q1	C, D, J, K, L, M, N, P
Q2	C, E, J, K, P
Q3	A, B, C, D, F, G, H, I, J, K, L, M, N, O
Q4	A, B, D, F, G, J, K, L, M, N, P
Q5	A, B, C, J, K, P
Q6	A, B, C, F, G, H, I, P

ภาพประกอบ 3-4 ตารางค่าของแอตทริบิวต์ X

ขั้นตอนที่ 2: กระบวนการจัดกลุ่มข้อมูล

ในขั้นตอนนี้นำเสนอขั้นตอนวิธีสำหรับจัดกลุ่มค่าของแอตทริบิวต์ที่มีการสอบถามไปด้วยกัน (Together) และสอบถามร่วมกัน (Joint) ซึ่งนำไปสู่รูปแบบการลงรหัสที่เหมาะสมเพื่อลดจำนวนบิตแมปเวกเตอร์ที่ต้องตรวจสอบ ประกอบด้วย 3 ขั้นตอนย่อยคือ 1) การสร้างตาราง Bit Matrix จากตารางค่าของแอตทริบิวต์ 2) การจัดกลุ่มข้อมูล และ 3) การจัดลำดับค่าของแอตทริบิวต์

ขั้นตอนที่ 2.1 การสร้างตาราง Bit Matrix จากตารางค่าของแอตทริบิวต์

ขั้นตอนนี้เป็นการสร้างตาราง Bit Matrix จากตารางค่าของแอตทริบิวต์ ตัวอย่างเช่น จากตารางค่าของแอตทริบิวต์ ดังภาพประกอบ 3-4 สามารถสร้างตาราง Bit Matrix มีขนาดเท่ากับ W x C เมื่อ W คือจำนวนรายการสอบถามทั้งหมดใน Query Workload และ C คือค่าคาร์ดินอลิตี้ของแอตทริบิวต์ที่นำมาทำดัชนี ซึ่งแต่ละแถวแสดงให้เห็นว่า แต่ละการสอบถามมีการสอบถามข้อมูลค่าใดบ้าง และในแต่ละคอลัมน์แสดงให้เห็นว่าแต่ละค่าถูกสอบถามในการสอบถามใดบ้าง กำหนดให้บิตเมตริกซ์มีค่าเป็น BitMatrix[i,j] เมื่อ i คือลำดับใน Query Workload โดยเริ่มจาก 0 ถึง W-1 และ j คือตำแหน่งลำดับค่าของแอตทริบิวต์ โดยเริ่มต้นจาก 0 ถึง C-1 ถ้าค่าข้อมูล j ถูกสอบถามในการสอบถามที่ i จะได้ว่า BitMatrix[i,j] เท่ากับ 1 ตัวอย่างเช่น จากภาพประกอบ 3-5 ค่า C ถูกสอบถามในการสอบถามที่ 1 (Q1) จะได้ว่า BitMatrix[0,2] มีค่าเท่ากับ 1 และค่า E ไม่ได้ถูกสอบถามในการสอบถามที่ 1 (Q1) จะได้ว่า BitMatrix[0,4] มีค่าเท่ากับ 0 การสร้างตาราง Bit Matrix เป็นการอ่านตารางค่าของแอตทริบิวต์เพียงครั้งเดียวและสามารถดำเนินการตรรกะบนตาราง Bit Matrix ได้โดยตรงจึงเพิ่มประสิทธิภาพในการประมวลผล แต่สิ้นเปลืองพื้นที่ในการจัดเก็บตาราง Bit Matrix เมื่อมีจำนวนรายการสอบถามและคาร์ดินอลิตี้ของแอตทริบิวต์เพิ่มสูงขึ้น

Query	**Value of Attribute *X***															
	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**O**	**P**
Q1	0	0	1	1	0	0	0	0	0	1	1	1	1	1	0	1
Q2	0	0	1	0	1	0	0	0	0	1	1	0	0	0	0	1
Q3	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	0
Q4	1	1	0	1	0	1	1	0	0	1	1	1	1	1	0	1
Q5	1	1	1	0	0	0	0	0	0	1	1	0	0	0	0	1
Q6	1	1	1	0	0	1	1	1	1	0	0	0	0	0	0	1
Frequency	4	4	5	3	1	3	3	2	2	5	5	3	3	3	1	5

ภาพประกอบ 3-5 ตัวอย่างตาราง Bit Matrix

จากตาราง Bit Matrix สามารถหาความถี่ (Frequency) ของแต่ละค่า โดยการนับจำนวนบิต 1 ของแต่ละค่า ตัวอย่างเช่น ค่า A มีบิต 1 ทั้งหมด 4 บิต ดังนั้นความถี่ของค่า A มีค่าเท่ากับ 4 และสามารถหาความถี่ของค่าอื่นๆ แสดงดังภาพประกอบ 3-5

ขั้นตอนที่ 2.2 การจัดกลุ่มข้อมูล

ขั้นตอนนี้เป็นการจัดกลุ่มข้อมูลที่มีโอกาสถูกสอบถามไปด้วยกันมากและมีจำนวนครั้งที่ถูกสอบถามร่วมกันบ่อย จากตาราง Bit Matrix ในขั้นตอนที่ 2.1 โดยใช้อัลกอริทึม Cluster-EBI แสดงดังภาพประกอบ 3-6 และชื่อตัวแปรที่ใช้ในอัลกอริทึมนี้ได้แสดงในตาราง 3-1

Algorithm: Cluster-EBI

Input: An array of BitMatrix[0..W-1, 0..C-1], Minimum Frequency, Threshold

Output: Clusters of Attribute values

1. Remove Attribute values that have frequency lower than Minimum Frequency
2. Initial each Attribute value to its own cluster
3. TJ_Table = ***buildTJ_Table(BitMatrix[0..W-1, 0..C-1])***
4. minTogether = min (TJ_Table.Together)
5. maxJoint = max (TJ_Table.Joint)
6. if (minTogether = 0)
7. cluster = ***clusterByZero(TJ_Table)***
8. cluster = ***clusterByTogether(TJ_Table, 1)***
9. end if
10. c = Count (cluster)
11. while (minTogether <= Threshold and maxJoint > 0)
12. TJ_Table = ***buildTJ_Table(BitMatrix[0..W-1, 0..c-1])***
13. minTogether = min (TJ_Table.Together)
14. maxJoint = max (TJ_Table.Joint)
15. cluster = ***clusterByMinAndMax(TJ_Table, minTogether, maxJoint)***
16. cluster = ***clusterByTogether(TJ_Table, minTogether)***
17. c = Count (cluster)
18. end while
19. Return cluster[1..c]

ภาพประกอบ 3-6 อัลกอริทึม Cluster-EBI

ตาราง 3-1 ชื่อตัวแปรที่ใช้ในอัลกอริทึม Cluster-EBI

ชื่อตัวแปร	ความหมาย
c	จำนวนกลุ่มข้อมูล ณ ปัจจุบัน หลังผ่านการจัดกลุ่มใหม่
TJ_Table	ตารางเก็บค่าของกลุ่มข้อมูลที่ถูกสอบถามไปด้วยกัน (Together) และค่าของกลุ่มข้อมูลที่ถูกสอบถามร่วมกัน (Joint)
TJ_Table.Together	ค่าของกลุ่มข้อมูลที่ถูกสอบถามไปด้วยกัน ในตาราง TJ_Table
TJ_Table.Joint	ค่าของกลุ่มข้อมูลที่ถูกสอบถามร่วมกัน ในตาราง TJ_Table
minTogether	ค่าที่น้อยที่สุดของกลุ่มข้อมูลที่ถูกสอบถามไปด้วยกัน
maxJoint	ค่าที่มากที่สุดของกลุ่มข้อมูลที่ถูกสอบถามร่วมกัน
Threshold	ค่าที่ตั้งไว้เป็นจุดตรวจสอบกับค่า minTogether
cluster	กลุ่มข้อมูลที่ถูกจัดกลุ่มแล้ว ณ ปัจจุบัน

อัลกอริทึม Cluster-EBI เป็นขั้นตอนวิธีที่ได้นำเสนอเพื่อจัดกลุ่มค่าของแอตทริบิวต์ที่สอบถามไปด้วยกันและสอบถามร่วมกัน ภาพประกอบ 3-6 แสดงอัลกอริทึม Cluster-EBI ประกอบด้วยขั้นตอนการทำงาน 16 ขั้นตอน มีรายละเอียดของแต่ละบรรทัดดังนี้

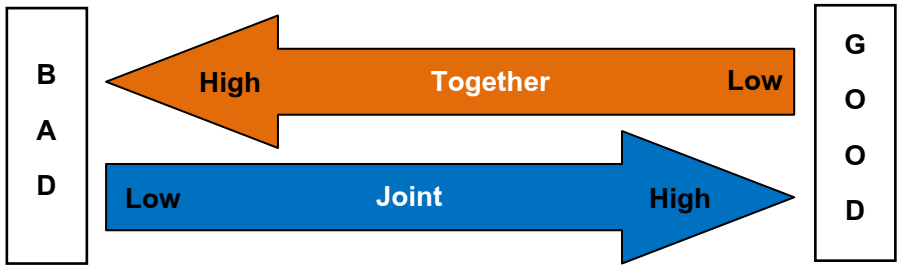
บรรทัดที่ 1 เลือกค่าข้อมูลของแอตทริบิวต์ที่มีความถี่น้อยกว่าความถี่ขั้นต่ำ (Minimum Frequency) ออกจากตาราง Bit Matrix ตัวอย่างเช่น กำหนดให้ความถี่ขั้นต่ำมีค่าเท่ากับ 2 จากภาพประกอบ 3-5 ค่าข้อมูล E และ O มีความถี่เท่ากับ 1 ซึ่งน้อยกว่าความถี่ขั้นต่ำที่กำหนดไว้จึงนำค่าข้อมูล E และ O ออกจากตาราง Bit Matrix แสดงดังภาพประกอบ 3-7

Query	Value of Attribute *X*													
	A	B	C	D	F	G	H	I	J	K	L	M	N	P
Q1	0	0	1	1	0	0	0	0	1	1	1	1	1	1
Q2	0	0	1	0	0	0	0	0	1	1	0	0	0	1
Q3	1	1	1	1	1	1	1	1	1	1	1	1	1	0
Q4	1	1	0	1	1	1	0	0	1	1	1	1	1	1
Q5	1	1	1	0	0	0	0	0	1	1	0	0	0	1
Q6	1	1	1	0	1	1	1	1	0	0	0	0	0	1

ภาพประกอบ 3-7 ตาราง Bit Matrix เมื่อตัดค่าข้อมูลที่มีความถี่น้อยกว่าความถี่ขั้นต่ำออก

บรรทัดที่ 2 กำหนดให้ทุกๆ ค่าของแอตทริบิวต์เป็นกลุ่มของตัวเอง จากภาพประกอบ 3-7 มีกลุ่มค่าของแอตทริบิวต์ทั้งหมด 14 กลุ่ม

บรรทัดที่ 3-9 เป็นขั้นตอนการจัดกลุ่มค่าของแอตทริบิวต์ที่มีการสอบถามไปด้วยกันและสอบถามร่วมกัน โดยพิจารณาค่าของแอตทริบิวต์ที่มีค่า Together เท่ากับ 0 และ 1 เพื่อจัดกลุ่มค่าแอตทริบิวต์ที่มีการสอบถามพร้อมกันเสมอ หากมีค่า Together น้อยหมายความว่า กลุ่มของค่าแอตทริบิวต์มีการสอบถามไปด้วยมาก นั่นคือสามารถจัดกลุ่มได้ดี และหากมีค่า Joint มากหมายความว่า กลุ่มของค่าแอตทริบิวต์มีการสอบถามร่วมกันมาก นั่นคือสามารถจัดกลุ่มได้ดี แสดงดังภาพประกอบ 3-8



ภาพประกอบ 3-8 คุณสมบัติของค่า Together และค่า Joint

บรรทัดที่ 3 เป็นการสร้างตาราง TJ_Table เพื่อจัดเก็บค่า Together และ Joint โดยเรียกใช้ฟังก์ชัน buildTJ_Table() แสดงดังภาพประกอบ 3-9

Function: buildTJ_Table(BitMatrix[0..W-1, 0..c-1])

Input: An array of BitMatrix[0..W-1, 0..c-1]

Output: A structure of array of TJ_Table

```
1.  for i = 0 to N-1
2.      for j = i+1 to N-1
3.          for k = 0 to W-1
4.              TJ_Table.Together_ij = count (BitMatrix[k,i] ⊕ BitMatrix[k,j])
5.              TJ_Table.Joint_ij = count (BitMatrix[k,i] ∧ BitMatrix[k,j])
6.          end for
7.      end for
8.  end for
9.  Return TJ_Table
```

ภาพประกอบ 3-9 ฟังก์ชัน buildTJ_Table

การทำงานของฟังก์ชัน buildTJ_Table() เป็นการหาค่า Together และค่า Joint ซึ่งมีเงื่อนไขในการพิจารณาในการจัดกลุ่มคือ จำนวนสมาชิกภายในกลุ่มข้อมูลทั้งสองกลุ่มที่พิจารณาต้องมีขนาดรวมกันเท่ากับ 2^i เมื่อ i = 1, 2, 3, …, $\lceil log_2 C \rceil$ – 1 สำหรับการหาค่า Together สามารถหาได้โดยการนับจำนวนบิต 1 ที่เกิดจากการดำเนินการตรรกะ XOR ระหว่างบิตเวกเตอร์ของกลุ่มข้อมูล สามารถคำนวณได้จากสมการที่ 1 การหาค่า Joint สามารถหาได้โดยการนับจำนวนบิต 1 ที่เกิดจากการดำเนินการตรรกะ AND ระหว่างบิตเวกเตอร์กลุ่มข้อมูล สามารถคำนวณได้จากสมการที่ 2 ผลลัพธ์ในขั้นตอนนี้คือตาราง TJ_Table ซึ่งเก็บค่า Together และค่า Joint ฟังก์ชันนี้จะสิ้นสุดการทำงานเมื่อพิจารณาครบกลุ่มข้อมูลแล้ว

$$TJ\_Table.Together_{ij} = count_{k=0}^{W-1} \left(BitMatrix[k,i] \oplus BitMatrix[k,j]\right) \quad (1)$$

$$TJ\_Table.Joint_{ij} = count_{k=0}^{W-1} \left(BitMatrix[k,i] \wedge BitMatrix[k,j]\right) \quad (2)$$

บรรทัดที่ 4 เป็นการหาค่า Together ที่น้อยที่สุด (minTogether) ในตาราง TJ_Table นั่นคือ เป็นการหากลุ่มค่าแอตทริบิวต์ที่มีสอบถามไปด้วยกันมากที่สุด

บรรทัดที่ 5 เป็นการหาค่า Joint ที่มากที่สุด (maxJoint) ในตาราง TJ_Table นั่นคือ เป็นการหากลุ่มค่าแอตทริบิวต์ที่มีจำนวนครั้งที่ถูกสอบถามร่วมกันมากที่สุด

บรรทัดที่ 6 เป็นการตรวจสอบค่า minTogether ว่ามีค่าเท่ากับ 0 หรือไม่ นั่นคือเป็นการตรวจสอบว่ามีกลุ่มค่าของแอตทริบิวต์ที่สอบถามไปด้วยกัน พร้อมกัน และจำนวนครั้งที่สอบถามเท่ากันหรือไม่ ซึ่งประกอบด้วย 2 ขั้นตอนย่อย คือ ขั้นตอนย่อยที่ 1 จัดกลุ่มค่าของแอตทริบิวต์ที่มีค่า Together เท่ากับ 0 และขั้นตอนย่อยที่ 2 จัดกลุ่มค่าของแอตทริบิวต์ที่มีค่า Together เท่ากับ 1 และมีค่า Joint มากที่สุด

บรรทัดที่ 7 เป็นการจัดกลุ่มค่าของแอตทริบิวต์ที่มีค่า Together เท่ากับ 0 นั่นคือเป็นการจัดกลุ่มค่าของแอตทริบิวต์ที่มีการสอบถามพร้อมกันและจำนวนครั้งในการสอบถามเท่ากัน โดยการเรียกใช้ฟังก์ชัน clusterByZero() แสดงดังภาพประกอบ 3-10

Function: clusterByZero(TJ_Table)

Input: A structure of array of TJ_Table

Output: An array of cluster

```
1.  for i = 1 to c
2.      for j = i+1 to c
3.          if (TJ_Table.Together_ij = 0)
4.              Add member of cluster_j to cluster_i
5.              Update BitMatrix
6.              Remove cluster_j
7.              c = c-1
8.          end if
9.      end for
10. end for
11. Return cluster[1..c]
```

ภาพประกอบ 3-10 ฟังก์ชัน clusterByZero

การทำงานของฟังก์ชัน clusterByZero() เป็นการจัดกลุ่มค่าของแอตทริบิวต์ที่มีค่า Together เท่ากับ 0 โดยนำสมาชิกของกลุ่มข้อมูลทั้งสองกลุ่มมารวมกัน จากนั้นจึงทำการอัพเดตค่าการสอบถามในตาราง Bit Matrix โดยดำเนินการตรรกะ AND ระหว่างบิตเวกเตอร์ของกลุ่มข้อมูลทั้งสอง และลดจำนวนกลุ่มข้อมูลลง ผลลัพธ์ในขั้นตอนนี้คือกลุ่มข้อมูลที่มีการสอบถามพร้อมกัน สอบถามไปด้วยกัน และมีจำนวนครั้งที่สอบถามเท่ากันเสมอ ฟังก์ชันนี้จะสิ้นสุดการทำงานเมื่อพิจารณาครบกลุ่มข้อมูลแล้ว

บรรทัดที่ 8 เป็นการจัดกลุ่มค่าของแอตทริบิวต์ที่มีค่า Together เท่ากับ 1 และกลุ่มค่าของแอตทริบิวต์ที่มีค่า Joint มากที่สุด เนื่องจากกลุ่มค่าของแอตทริบิวต์ที่มีค่า Together เท่ากับ 1 จะมีโอกาสที่ถูกสอบถามไปด้วยกันมาก โดยการเรียกใช้ฟังก์ชัน clusterByTogether() แสดงดังภาพประกอบ 3-11

Function: clusterByTogether(TJ_Table, together_value)

Input: A structure of array of TJ_Table and together value

Output: An array of cluster

```
1.  for i = 1 to c
2.      for j = i+1 to c
3.          if (TJ_Table.Together_ij = together_value)
4.              if (TJ_Table.Joint_ij is maximum)
5.                  Add member of cluster_j to cluster_i
6.                  Update BitMatrix
7.                  Remove cluster_j
8.                  c = c-1
9.              end if
10.         end if
11.     end for
12. end for
13. Return cluster[1..c]
```

ภาพประกอบ 3-11 ฟังก์ชัน clusterByTogether

การทำงานของฟังก์ชัน clusterByTogether() เป็นการจัดกลุ่มค่าของแอตทริบิวต์ที่มีค่า Together เท่ากับค่า Together_value ที่กำหนดให้ และกลุ่มค่าของแอตทริบิวต์ที่มีค่า Joint มากที่สุด โดยการนำสมาชิกของทั้งสองกลุ่มมารวมกัน จากนั้นทำการอัพเดตค่าการสอบถามในตาราง Bit Matrix โดยดำเนินการตรรกะ AND ระหว่างบิตเวกเตอร์ของกลุ่มข้อมูลทั้งสอง และลดจำนวนกลุ่มข้อมูลลง ผลลัพธ์ในขั้นตอนนี้คือ กลุ่มค่าของแอตทริบิวต์ที่มีค่า Together เท่ากับ Together_value ที่กำหนดมาและมีค่า Joint มากที่สุด ฟังก์ชันนี้จะสิ้นสุดการทำงานเมื่อพิจารณาครบกลุ่มข้อมูลแล้ว

บรรทัดที่ 9 สิ้นสุดการตรวจสอบเงื่อนไขของค่า minTogether เท่ากับ 0

บรรทัดที่ 10 เป็นการนับจำนวนกลุ่มค่าแอตทริบิวต์ที่เหลืออยู่ทั้งหมด ณ ปัจจุบัน เพื่อนำไปสร้างตาราง TJ_Table ขึ้นมาใหม่

บรรทัดที่ 11-18 เป็นขั้นตอนการจัดกลุ่มค่าของแอตทริบิวต์ที่มีค่า Together น้อยที่สุดและค่า Joint มากที่สุด

บรรทัดที่ 11 เป็นการตรวจสอบค่า minTogether ว่ามีค่าน้อยกว่าหรือเท่ากับค่า Threshold ที่กำหนดไว้หรือไม่ และตรวจสอบค่า maxJoint ว่ามีค่ามากกว่า 0 หรือไม่ นั่นคือเป็นการตรวจสอบว่ายังสามารถจัดกลุ่มค่าของแอตทริบิวต์ได้อีกหรือไม่

บรรทัดที่ 12 เป็นการสร้างตาราง TJ_Table ใหม่จากกลุ่มข้อมูลใหม่เพื่อจัดเก็บค่า Together และ Joint โดยเรียกใช้ฟังก์ชัน buildTJ_Table() แสดงดังภาพประกอบ 3-9

บรรทัดที่ 13 เป็นการหาค่า Together ที่น้อยที่สุด (minTogether) ในตาราง TJ_Table นั่นคือเป็นการหากลุ่มค่าแอตทริบิวต์ที่มีสอบถามไปด้วยกันมากที่สุด

บรรทัดที่ 14 เป็นการหาค่า Joint ที่มากที่สุด (maxJoint) ในตาราง TJ_Table นั่นคือเป็นการหากลุ่มค่าแอตทริบิวต์ที่มีจำนวนครั้งที่ถูกสอบถามร่วมกันมากที่สุด

บรรทัดที่ 15 เป็นการจัดกลุ่มค่าของแอตทริบิวต์ที่มีค่า Together เท่ากับ minTogether และค่า Joint เท่ากับ maxJoint โดยการเรียกใช้ฟังก์ชัน clusterByMinAndMax() แสดงดังภาพประกอบ 3-12

Function: clusterByMinAndMax(TJ_Table, minTother, maxJoint)

Input: A structure of array of TJ_Table, value of min Together and max Joint

Output: An array of cluster

```
1.  for i = 1 to c
2.      for j = i+1 to c
3.          if (TJ_Table.Together_ij=minTogether and TJ_Table.Joint_ij=maxJoint)
4.              Add member of cluster_j to cluster_i
5.              Update BitMatrix
6.              Remove cluster_j
7.              c = c-1
8.          end if
9.      end for
10. end for
11. Return cluster[1..c]
```

ภาพประกอบ 3-12 ฟังก์ชัน clusterByMinAndMax

การทำงานของฟังก์ชัน clusterByMinAndMax() เป็นการจัดกลุ่มค่าของแอตทริบิวต์ที่มีค่า Together เท่ากับ minTogether และค่า Joint เท่ากับค่า maxJoint โดยการ

นำสมาชิกของทั้งสองกลุ่มมารวมกัน จากนั้นทำการอัพเดตค่าการสอบถามในตาราง Bit Matrix โดยการดำเนินการตรรกะ AND ระหว่างบิตเวกเตอร์ของทั้งสองกลุ่ม และลดจำนวนกลุ่มข้อมูลลง ผลลัพธ์ในขั้นตอนนี้คือกลุ่มค่าของแอตทริบิวต์ที่มีค่า Together เท่ากับ minTogether และค่า Joint เท่ากับ maxJoint ฟังก์ชันนี้จะสิ้นสุดการทำงานเมื่อพิจารณาครบกลุ่มข้อมูลแล้ว

บรรทัดที่ 16 ในขั้นตอนนี้สามารถดำเนินการได้เช่นเดียวกันกับขั้นตอนในบรรทัดที่ 9 โดยการเรียกใช้ฟังก์ชัน clusterByTogether() แสดงดังภาพประกอบ 3-10 ซึ่งเป็นการจัดกลุ่มข้อมูลที่มีค่า Together เท่ากับ minTogether และค่า Joint มากที่สุด

บรรทัดที่ 17 เป็นการนับจำนวนกลุ่มค่าแอตทริบิวต์ที่เหลืออยู่ทั้งหมด ณ ปัจจุบันเพื่อนำไปสร้างตาราง TJ_Table ขึ้นมาใหม่

การทำงานของอัลกอริทึม Cluster-EBI จะหยุดการทำงานก็ต่อเมื่อมีค่า minTogether มากกว่าค่า Threshold ที่กำหนดไว้และค่า Joint น้อยกว่าหรือเท่ากับ 0 นั่นคือกลุ่มค่าของแอตทริบิวต์แต่ละกลุ่มมีโอกาสที่ถูกสอบถามไปด้วยกันน้อยมาก และกลุ่มค่าของแอตทริบิวต์แต่ละกลุ่มไม่ถูกสอบถามร่วมกัน ผลลัพธ์ที่ได้คือ กลุ่มค่าของแอตทริบิวต์ที่มีจำนวนสมาชิกที่แตกต่างกันและมีสมาชิกไม่ซ้ำกัน

ตัวอย่างการจัดกลุ่มข้อมูลตามขั้นตอนวิธีการข้างต้นโดยใช้ตาราง Bit Matrix จากภาพประกอบ 3-7 สร้างกลุ่มข้อมูลให้กับแต่ละค่าของแอตทริบิวต์ และสร้างตาราง TJ_Table เช่น บิตเวกเตอร์ของกลุ่มค่าแอตทริบิวต์ (A) คือ 001111 มีจำนวนสมาชิกเท่ากับ 1 และบิตเวกเตอร์ของกลุ่มค่าแอตทริบิวต์ (B) คือ 001111 มีจำนวนสมาชิกเท่ากับ 1 ทั้งสองกลุ่มมีจำนวนสมาชิกรวมกันเท่ากับ 2 (2^1) จึงนำทั้งสองกลุ่มมาคำนวณหาค่า Together และค่า Joint โดยคำนวณหาค่า Together ได้จากการดำเนินการตรรกะ XOR ระหว่างบิตเวกเตอร์ของกลุ่มค่าแอตทริบิวต์ (A) และบิตเวกเตอร์ของกลุ่มค่าแอตทริบิวต์ (B) ผลลัพธ์ที่ได้คือ 000000 จากนั้นนับจำนวนบิตที่มีค่าเท่ากับ 1 จะได้จำนวนบิตที่มีค่าเท่ากับ 1 ทั้งหมด 0 ตัว ดังนั้นค่า Together ของกลุ่มค่าของแอตทริบิวต์ (A) และ (B) จึงเท่ากับ 0 และคำนวณหาค่า Joint ได้จากการดำเนินการตรรกะ AND ระหว่างบิตเวกเตอร์ของกลุ่มค่าของแอตทริบิวต์ (A) และบิตเวกเตอร์ของกลุ่มค่าของแอตทริบิวต์ (B) ผลลัพธ์ที่ได้คือ 001111 จากนั้นนับจำนวนบิตที่มีค่าเท่ากับ 1 จะได้จำนวนบิตที่มีค่าเท่ากับ 1 ทั้งหมด 4 ตัว ดังนั้นค่า Joint ของกลุ่มค่าแอตทริบิวต์ (A) และ (B) เท่ากับ 4 และสามารถคำนวณหาค่า Together และค่า Joint ของกลุ่มค่าของแอตทริบิวต์อื่นๆ แสดงดังภาพประกอบ 3-13 เมื่อตัวเลขที่อยู่หน้าเครื่องหมาย “:” หมายถึงค่า Together และตัวเลขที่อยู่หลังเครื่องหมาย “:” หมายถึงค่า Joint

	A	B	C	D	F	G	H	I	J	K	L	M	N	P
A		0:4	3:3	3:2	1:3	1:3	2:2	2:2	3:3	3:3	3:2	3:2	3:2	3:3
B			3:3	3:2	1:3	1:3	2:2	2:2	3:3	3:3	3:2	3:2	3:2	3:3
C				4:2	4:2	4:2	3:2	3:2	2:4	2:4	4:2	4:2	4:2	2:4
D					2:2	2:2	3:1	3:1	2:3	2:3	0:3	0:3	0:3	4:2
F						0:3	1:2	1:2	4:2	4:2	2:2	2:2	2:2	4:2
G							1:2	1:2	4:2	4:2	2:2	2:2	2:2	4:2
H								0:2	5:1	5:1	3:1	3:1	3:1	5:1
I									5:1	5:1	3:1	3:1	3:1	5:1
J										0:5	2:3	2:3	2:3	2:4
K											2:3	2:3	2:3	2:4
L												0:3	0:3	4:2
M													0:3	4:2
N														4:2
P														

ภาพประกอบ 3-13 ตาราง TJ_Table ในรอบที่ 1

กำหนดให้ค่า Threshold ในการจัดกลุ่มค่าของแอตทริบิวต์เท่ากับ 3 จากภาพประกอบ 3-13 ตาราง TJ_Table มีค่า minTogether เท่ากับ 0 และค่า maxJoint = 5 ในรอบที่ 1 เนื่องจากค่า minTogether เท่ากับ 0 จึงจัดกลุ่มข้อมูลที่มีค่า Together เท่ากับ 0 ก่อน (ช่องที่มีการเน้นสี ในภาพประกอบ 3-13) ตัวอย่างเช่น พิจารณาแถวของกลุ่มค่าแอตทริบิวต์ (A) จะเห็นได้ว่า คอลัมน์ของกลุ่มค่าแอตทริบิวต์ (B) มีค่า Together เท่ากับ 0 จึงรวมสมาชิกของกลุ่มค่าแอตทริบิวต์ (A) และกลุ่มค่าแอตทริบิวต์ (B) จะได้กลุ่มค่าแอตทริบิวต์ (A, B) จากนั้นนำบิตเวกเตอร์ของกลุ่มแอตทริบิวต์ (A) และบิตเวกเตอร์กลุ่มค่าแอตทริบิวต์ (B) มาดำเนินการตรรกะ AND และลบกลุ่มค่าแอตทริบิวต์ (B) ออกจากตาราง TJ_Table พิจารณาแถวถัดไป คือ แถวของกลุ่มแอตทริบิวต์ (C) จะเห็นได้ว่าไม่มีคอลัมน์ใดมีค่า Together เท่ากับ 0 ดังนั้นกลุ่มค่าแอตทริบิวต์ (C) ไม่สามารถรวมกลุ่มกับกลุ่มค่าแอตทริบิวต์อื่นได้ พิจารณาแถวของกลุ่มค่าแอตทริบิวต์ (D) จะเห็นได้ว่า คอลัมน์ของกลุ่มค่าแอตทริบิวต์ (L), (M) และ (N) มีค่า Together เท่ากับ 0 จึงรวมสมาชิกของกลุ่มค่าแอตทริบิวต์ (D), (L), (M) และ (N) จากนั้นนำบิตเวกเตอร์ของกลุ่มค่าแอตทริบิวต์ (D), (L), (M) และ (N) มาดำเนินการตรรกะ AND และลบกลุ่มค่าแอตทริบิวต์ (L), (M) และ (N) ออกจากตาราง TJ_Table ผลลัพธ์ในขั้นตอนนี้คือ กลุ่มค่าแอตทริบิวต์ที่มีค่า Together เท่ากับ 0 แสดงดังภาพประกอบ 3-14

#Cluster	Member	Number of Member
1	A, B	2
2	C	1
3	D, L, M, N	4
4	F, G	2
5	H, I	2
6	J, K	2
7	P	1

ภาพประกอบ 3-14 กลุ่มข้อมูลที่มีค่า Together เท่ากับ 0 ในรอบที่ 1

จากภาพประกอบ 3-14 นำกลุ่มที่ได้มาจัดกลุ่มค่าแอตทริบิวต์ที่มีค่า Together เท่ากับ 1 และมีค่า Joint มากที่สุด (ช่องที่มีการเน้นสี ▭ ในภาพประกอบ 3-13) เช่น จากตาราง TJ_Table ดังภาพประกอบ 3-13 พิจารณาแถว A ซึ่งเป็นกลุ่มค่าแอตทริบิวต์ (A, B) ที่ได้จากการจัดกลุ่มก่อนหน้านี้จะเห็นได้ว่า คอลัมน์ F ซึ่งเป็นกลุ่มค่าแอตทริบิวต์ (F, G) ที่ได้จากการจัดกลุ่มก่อนหน้านี้มีค่า Together เท่ากับ 1 และค่า Joint เท่ากับ 3 และเมื่อพิจาณากลุ่มค่าแอตทริบิวต์ (F, G) กับกลุ่มค่าแอตทริบิวต์ (H, I) ซึ่งมีค่า Together เท่ากับ 1 และค่า Joint เท่ากับ 2 กลุ่มค่าแอตทริบิวต์ (A, B) และกลุ่มค่าแอตทริบิวต์ (F, G) มีค่า Together น้อยที่สุด และมีค่า Joint มากที่สุด จึงรวมสมาชิกของกลุ่มข้อมูล (A, B) และกลุ่มข้อมูล (F, G) เข้าด้วยกันจะได้กลุ่มค่าแอตทริบิวต์ใหม่ (A, B, F, G) จากนั้นนำบิตเวกเตอร์ของกลุ่มค่าแอตทริบิวต์ (A, B) และกลุ่มค่าแอตทริบิวต์ (F, G) มาดำเนินการตรรกะ AND และลบกลุ่มค่าแอตทริบิวต์ (F, G) ออกจากตาราง TJ_Table และสามารถพิจารณาการจัดกลุ่มค่าแอตทริบิวต์อื่นๆ แสดงดังภาพประกอบ 3-15 และตาราง Bit Matrix ที่ได้จากการจัดกลุ่มในรอบที่ 1 แสดงดังภาพประกอบ 3-16

#Cluster	Member	Number of Member
1	A, B, F, G	4
2	C	1
3	D, L, M, N	4
4	H, I	2
5	J, K	2
6	P	1

ภาพประกอบ 3-15 กลุ่มข้อมูลที่ได้ในรอบที่ 1

Query	Value of Attribute *X*					
	A,B,F,G	**C**	**D,L,M,N**	**H,I**	**J,K**	**P**
Q1	0	1	1	0	1	1
Q2	0	1	0	0	1	1
Q3	1	1	1	1	1	0
Q4	1	0	1	0	1	1
Q5	0	1	0	0	1	1
Q6	1	1	0	1	0	1

ภาพประกอบ 3-16 ตาราง Bit Matrix ที่ได้ในรอบที่ 1

เนื่องจากค่า minTogether น้อยกว่าค่า Threshold ที่กำหนดไว้และค่า maxJoint มากกว่า 0 ในรอบที่ 2 จึงสร้างตาราง TJ_Table โดยใช้ตาราง Bit Matrix จากภาพประกอบ 3-16 ซึ่งพิจารณาในทำนองเดียวกันแสดงดังภาพประกอบ 3-17 เมื่อตัวเลขที่อยู่หน้าเครื่องหมาย ":" หมายถึงค่า Together ตัวเลขที่อยู่หลังเครื่องหมาย ":" หมายถึงค่า Joint และเครื่องหมาย "*" หมายถึงกลุ่มข้อมูลที่ไม่สามารถนำมาพิจารณาได้

	A,B,F,G	C	D,L,M,N	H,I	J,K	P
A,B,F,G		*	2:2	*	*	*
C			*	*	*	2:4
D,L,M,N				*	*	*
H,I					5:1	*
J,K						*
P						

ภาพประกอบ 3-17 ตาราง TJ_Table ในรอบที่ 2

จากภาพประกอบ 3-17 ตาราง TJ_Table มีค่า minTogether เท่ากับ 2 และมีค่า maxJoint = 4 เนื่องจากค่า minTogether มีค่าไม่เท่ากับ 0 ดังนั้นการจัดกลุ่มค่าแอตทริบิวต์ในรอบที่ 2 จึงจัดกลุ่มค่าแอตทริบิวต์ที่มีค่า Together เท่ากับ 2 และมีค่า Joint เท่ากับ 4 ก่อน (ช่องที่มีการเน้นสี ▭ ในภาพประกอบ 3-17) ตัวอย่างเช่น แถวของกลุ่มค่าแอตทริบิวต์ (C) และคอลัมน์ของกลุ่มค่าแอตทริบิวต์ (P) มีค่า Together เท่ากับ 2 และค่า Joint เท่ากับ 4 ดังนั้นจึงรวมสมาชิกของกลุ่มค่าแอตทริบิวต์ (C) และกลุ่มค่าแอตทริบิวต์ (P) จะได้กลุ่มค่าแอตทริบิวต์ (C, P) จากนั้นนำบิตเวกเตอร์ของกลุ่มแอตทริบิวต์ (C) และบิตเวกเตอร์ของกลุ่มค่าแอตทริบิวต์

(P) มาดำเนินการตรรกะ AND และลบกลุ่มค่าแอตทริบิวต์ (P) ออกจากตาราง TJ_Table และพิจารณากลุ่มค่าแอตทริบิวต์อื่นๆ จนครบทุกกลุ่ม

จากนั้นทำการจัดกลุ่มค่าแอตทริบิวต์ที่มีค่า Together เท่ากับ 2 และมีค่า Joint มากที่สุด (ช่องที่มีการเน้นสี  ในภาพประกอบ 3-17) ตัวอย่างเช่น แถวของกลุ่มค่าของแอตทริบิวต์ (A, B, F, G) และคอลัมน์ของกลุ่มค่าแอตทริบิวต์ (D, L, M, N) มีค่า Together เท่ากับ 2 และมีค่า Joint เท่ากับ 2 ซึ่งเป็นค่า Joint ที่มากที่สุด ดังนั้นจึงรวมสมาชิกของกลุ่มค่าแอตทริบิวต์ (A, B, F, G) และกลุ่มค่าแอตทริบิวต์ (D, L, M, N) จะได้กลุ่มค่าแอตทริบิวต์ (A, B, F, G, D, L, M, N) จากนั้นนำบิตเวกเตอร์ของกลุ่มค่าแอตทริบิวต์ (A, B, F, G) และกลุ่มค่าแอตทริบิวต์ (D, L, M, N) มาดำเนินการตรรกะ AND และลบกลุ่มค่าแอตทริบิวต์ (D, L, M, N) ออกจากตาราง TJ_Table ผลลัพธ์จากการจัดกลุ่มในรอบที่ 2 แสดงดังภาพประกอบที่ 3-18 และตาราง Bit Matrix ที่ได้จากการจัดกลุ่มในรอบที่ 2 แสดงดังภาพประกอบ 3-19

#Cluster	Member	Number of Member
1	A, B, F, G, D, L, M, N	8
2	C, P	2
3	H, I	2
4	J, K	2

ภาพประกอบ 3-18 กลุ่มข้อมูลที่ได้ในรอบที่ 2

Query	Value of Attribute *X*			
	A,B,F,G,D,L,M,N	C,P	H,I	J,K
Q1	0	1	0	1
Q2	0	1	0	1
Q3	1	0	1	1
Q4	1	0	0	1
Q5	0	1	0	1
Q6	0	1	1	0

ภาพประกอบ 3-19 ตาราง Bit Matrix ที่ได้ในรอบที่ 2

สร้างตาราง TJ_Table โดยใช้ตาราง Bit Matrix จากภาพประกอบ 3-19 ซึ่งพิจารณาในทำนองเดียวกัน แสดงดังภาพประกอบ 3-20 เมื่อตัวเลขที่อยู่หน้าเครื่องหมาย “:” หมายถึงค่า Together ตัวเลขที่อยู่หลังเครื่องหมาย “:” หมายถึงค่า Joint และเครื่องหมาย “*” หมายถึงกลุ่มข้อมูลที่ไม่สามารถนำมาพิจารณาได้

	A,B,F,G,D,L,M,N	C,P	H,I	J,K
A,B,F,G,D,L,M,N		*	*	*
C,P			4:1	3:3
H,I				5:1
J,K				

ภาพประกอบ 3-20 ตาราง TJ_Table ในรอบที่ 3

จากภาพประกอบ 3-20 ตาราง TJ_Table มีค่า minTogether เท่ากับ 3 และมีค่า maxJoint = 3 ในรอบที่ 3 จึงจัดกลุ่มค่าแอตทริบิวต์ที่มีค่า Together เท่ากับ 3 และมีค่า Joint เท่ากับ 3 (ช่องที่มีการเน้นสี ▭ ในภาพประกอบ 3-20) ตัวอย่างเช่นแถวของกลุ่มค่าแอตทริบิวต์ (C, P) และคอลัมน์ของกลุ่มค่าแอตทริบิวต์ (J, K) มีค่า Together เท่ากับ 3 และค่า Joint เท่ากับ 3 จึงรวมสมาชิกของกลุ่มค่าแอตทริบิวต์ (C, P) และกลุ่มค่าแอตทริบิวต์ (J, K) จะได้กลุ่มค่าแอตทริบิวต์คือ (C, P, J, K) จากนั้นนำบิตเวกเตอร์ของกลุ่มค่าแอตทริบิวต์ (C, P) และบิตเวกเตอร์ของกลุ่มค่าแอตทริบิวต์ (J, K) มาดำเนินการตรรกะ AND และลบกลุ่มค่าแอตทริบิวต์ (J, K) ออกจากตาราง TJ_Table ผลลัพธ์จากการจัดกลุ่มในรอบที่ 2 แสดงดังภาพประกอบที่ 3-21 และตาราง Bit Matrix ที่ได้จากการจัดกลุ่มในรอบที่ 2 แสดงดังภาพประกอบ 3-22

#Cluster	**Member**	**Number of Member**
1	A, B, F, G, D, L, M, N	8
2	C, P, J, K	4
3	H, I	2

ภาพประกอบ 3-21 กลุ่มข้อมูลที่ได้ในรอบที่ 3

Query	Value of Attribute *X*		
	A,B,F,G,D,L,M,N	**C,P,J,K**	**H,I**
Q1	0	1	0
Q2	0	1	0
Q3	1	0	1
Q4	1	0	0
Q5	0	1	0
Q6	0	1	1

ภาพประกอบ 3-22 ตาราง Bit Matrix ที่ได้ในรอบที่ 3

สร้างตาราง TJ_Table โดยใช้ตาราง Bit Matrix จากภาพประกอบ 3-22 ซึ่งพิจารณาในทำนองเดียวกัน แสดงดังภาพประกอบ 3-23 เมื่อตัวเลขที่อยู่หน้าเครื่องหมาย ":" หมายถึงค่า Together ตัวเลขที่อยู่หลังเครื่องหมาย ":" หมายถึงค่า Joint และเครื่องหมาย "*" หมายถึงกลุ่มข้อมูลที่ไม่สามารถนำมาพิจารณาได้

	A,B,F,G,D,L,M,N	C,P,J,K	H,I
A,B,F,G,D,L,M,N		*	*
C,P,J,K			*
H,I			

ภาพประกอบ 3-23 ตาราง TJ_Table เมื่อผ่านการจัดกลุ่มในรอบที่ 4

จากภาพประกอบ 3-23 จะเห็นได้ว่าไม่สามารถหาค่า Together และค่า Joint ของแต่ละกลุ่มได้ ดังนั้นจากตัวอย่างที่กล่าวมาข้างต้นจะได้กลุ่มข้อมูลที่ต้องการตามอัลกอริทึม Cluster-EBI แสดงดังภาพประอบ 3-24

#Cluster	Member	Number of Member
1	A, B, F, G, D, L, M, N	8
2	C, P, J, K	4
3	H, I	2

ภาพประกอบ 3-24 กลุ่มข้อมูลที่ได้จากอัลกอริทึม Cluster-EBI

ขั้นตอนที่ 2.3 การจัดลำดับค่าของแอตทริบิวต์

ขั้นตอนนี้เป็นการจัดลำดับค่าของแอตทริบิวต์แต่ละค่าให้อยู่ในตำแหน่งที่เหมาะสมก่อนเข้ารหัสในขั้นตอนที่ 3 โดยการนำสมาชิกในกลุ่มข้อมูลที่หาได้แต่ละกลุ่มมาจัดเรียงให้เป็นลำดับที่ต่อเนื่องกัน โดยการจัดเรียงตามจำนวนสมาชิกที่มีขนาดเท่ากับ 2^i เมื่อ i = $\lceil log_2 C \rceil$-1,..., 3, 2 จากมากไปน้อย จัดเรียงกลุ่มข้อมูลที่เหลือตามจำนวนสมาชิกที่มีขนาดเป็นจำนวนคู่จากมากไปน้อย และจัดเรียงกลุ่มข้อมูลที่เหลือตามจำนวนสมาชิกที่มีขนาดเป็นจำนวนคี่จากมากไปน้อยตามลำดับ จากนั้นนำค่าแอตทริบิวต์ที่เหลือ (ค่าของแอตทริบิวต์ที่ไม่ได้เป็นสมาชิกของกลุ่มใด) มาจัดเรียงต่อจากข้อมูลที่จัดเรียงอยู่แล้ว ตัวอย่างเช่น กลุ่มค่าแอตทริบิวต์ จากภาพประกอบ 3-24 จะได้ว่ากลุ่มข้อมูลที่มีจำนวนสมาชิกที่มีขนาดเท่ากับ 2^i มากที่สุดคือ กลุ่มข้อมูล {A, B, F, G, D, L, M, N} รองลงมาคือ กลุ่มข้อมูล {C, P, J, K} กลุ่มข้อมูลที่เหลือที่มีจำนวนสมาชิกเป็นจำนวนคู่คือ กลุ่มข้อมูล {H, I} และไม่มีกลุ่มข้อมูลที่มีขนาดเป็นจำนวนคี่ ดังนั้นผลลัพธ์ของการจัดลำดับค่าของแอตทริบิวต์ คือ A, B, F, G, D, L, M, N, C, P, J, K, H, I ตามลำดับ และเมื่อนำค่าของแอตทริบิวต์ที่เหลือ นั่นคือ E และ O มาจัดเรียงต่อจะได้ผลลัพธ์สุดท้ายของการจัดเรียงลำดับค่าของแอตทริบิวต์เป็น A, B, F, G, D, L, M, N, C, P, J, K, H, I, E, O ตามลำดับ

ขั้นตอนที่ 3: การลงรหัสค่าของแอตทริบิวต์

ขั้นตอนการลงรหัสค่าของแอตทริบิวต์เป็นขั้นตอนการลงรหัสค่าของแอตทริบิวต์ที่ถูกจัดเรียงจากขั้นตอนที่ 2 ตั้งแต่ค่าข้อมูลแรกจนถึงค่าข้อมูลตัวสุดท้าย โดยกำหนดให้แต่ละค่ามีรูปแบบการลงรหัสเป็นรูปแบบของเลขฐานสอง (Binary Encoded) เริ่มตั้งแต่ 0 ถึง C-1 ซึ่งนำไปสู่การสร้างดัชนีบิตแมปแบบเข้ารหัสที่มีประสิทธิภาพสำหรับการสอบถามข้อมูลแบบสมาชิกและการสอบถามข้อมูลแบบค่าเท่ากัน ดังนั้นจากตัวอย่างลำดับค่าของแอตทริบิวต์ที่ถูกจัดเรียงไว้ในขั้นตอน 2.3 ซึ่งแอตทริบิวต์ *X* มีคาร์ดินอลิตี้เท่ากับ 16 จึงใช้ 4 บิตในการลงรหัสค่าของแอตทริบิวต์ตามลำดับตั้งแต่ค่าแรก (A) จนถึงค่าสุดท้าย (O) จะได้ผลลัพธ์ดังแสดงในภาพประกอบ 3-25(ข)

RID	*X*
1	N
2	B
3	P
4	F
5	H
6	D
7	K
8	A
9	L
10	C
.	.
.	.
.	.
10000	M

E_3	E_2	E_1	E_0
0	1	1	1
0	0	0	1
1	0	0	1
0	0	1	0
1	1	0	0
0	1	0	0
1	0	1	1
0	0	0	0
0	1	0	1
1	0	0	0
.	.	.	.
.	.	.	.
.	.	.	.
0	1	1	0

(ก) ดัชนีบิตแมปแบบเข้ารหัสที่ใช้เทคนิคการจัดกลุ่มข้อมูล

A	0000
B	0001
F	0010
G	0011
D	0100
L	0101
M	0110
N	0111
C	1000
P	1001
J	1010
K	1011
H	1100
I	1101
E	1110
O	1111

(ข) รูปแบบการลงรหัส

ภาพประกอบ 3-25 ดัชนีบิตแมปแบบเข้ารหัสที่ใช้เทคนิคการจัดกลุ่มข้อมูล

ขั้นตอนที่ 4: การสร้างดัชนีบิตแมปแบบเข้ารหัส

ขั้นตอนการสร้างดัชนีบิตแมปแบบเข้ารหัสเป็นขั้นตอนการสร้างดัชนีบิตแมปแบบเข้ารหัสโดยการใช้รูปแบบการลงรหัสจากขั้นตอนที่ 3 แทนค่าของแอตทริบิวต์ที่นำมาทำดัชนีในแต่ละบิตแมปเวกเตอร์ ตัวอย่างเช่น แอตทริบิวต์ *X* มีคาร์ดินอลิตี้เท่ากับ 16 แสดงว่าต้องใช้ 4 บิตแมปเวกเตอร์ในการสร้างดัชนี ได้แก่ E_3, E_2, E_1 และ E_0 โดยที่แต่ละค่าถูกลงรหัสในรูปแบบ $E_3E_2E_1E_0$ จากภาพประกอบ 3-25 แถวที่ 1 มีค่าของแอตทริบิวต์เท่ากับ N ซึ่งค่า N มีรูปแบบการลงรหัสเป็น 0111 ดังนั้นแถวที่ 1 ของดัชนีบิตแมปแบบเข้ารหัสจึงมีบิตแมปเวกเตอร์ E_3 เท่ากับ 0 บิตแมปเวกเตอร์ E_2 เท่ากับ 1 บิตแมปเวกเตอร์ E_1 เท่ากับ 1 และบิตแมปเวกเตอร์ E_0 เท่ากับ 1 และสามารถแทนค่าของแอตทริบิวต์ที่นำมาดัชนีในแต่ละแถวแสดงดังภาพประกอบ 3-25

3.2 การสอบถามข้อมูลแบบสมาชิกบนดัชนีบิตแมปแบบเข้ารหัส

การสอบถามข้อมูลแบบสมาชิกมีรูปแบบเงื่อนไขคือ “X in $\{v_1,\ v_2,\ \ldots,\ v_n\}$” หมายถึง การสอบถามว่าบนแอตทริบิวต์ X มีแถวใดบ้างที่มีค่าเท่ากับ $v_1,\ v_2,\ldots,\ v_n$ สำหรับการสอบถามข้อมูลแบบสมาชิกบนดัชนีบิตแมปแบบเข้ารหัสที่ใช้เทคนิคการจัดกลุ่มข้อมูลประกอบด้วย การอ่านลำดับของแต่ละค่าที่สอบถาม การค้นหารูปแบบการเข้ารหัสของแต่ละค่าที่สอบถาม การดำเนินการตรรกะ OR ระหว่างลำดับของแต่ละค่า การลดรูปฟังก์ชันการเข้าถึงข้อมูล การอ่านบิตแมปเวกเตอร์ และดำเนินการตรรกะตามฟังก์ชันที่ได้ลดรูปแล้ว

ตัวอย่างเช่น เมื่อต้องการสอบถามข้อมูลแบบสมาชิกตามเงื่อนไขของการสอบถาม Q1 ในภาพประกอบ 3-3 โดยใช้รูปแบบการลงรหัสจากภาพประกอบ 3-25(ข) สามารถทำได้ดังนี้

1) อ่านรูปแบบการลงรหัส

C มีรูปแบบการลงรหัสเป็น $E_3E_2{}'E_1{}'E_0{}'$ (1000)

D มีรูปแบบการลงรหัสเป็น $E_3{}'E_2E_1{}'E_0{}'$ (0100)

J มีรูปแบบการลงรหัสเป็น $E_3E_2{}'E_1\mathrm{E}_0{}'$ (1010)

K มีรูปแบบการลงรหัสเป็น $E_3E_2{}'E_1E_0$ (1011)

L มีรูปแบบการลงรหัสเป็น $E_3{}'E_2E_1{}'E_0$ (0101)

M มีรูปแบบการลงรหัสเป็น $E_3{}'E_2E_1E_0{}'$ (0110)

N มีรูปแบบการลงรหัสเป็น $E_3{}'E_2E_1E_0$ (0111)

P มีรูปแบบการลงรหัสเป็น $E_3E_2{}'E_1{}'E_0$ (1001)

2) ดำเนินการตรรกะ OR (ในที่นี้ใช้สัญลักษณ์ + แทนการดำเนินการตรรกะ OR) ระหว่างรูปแบบการลงรหัสของแต่ละค่า จะได้ฟังก์ชันการเข้าถึงข้อมูลดังนี้

$$E_3E_2{}'E_1{}'E_0{}' + E_3{}'E_2E_1{}'E_0{}' + E_3E_2{}'E_1\mathrm{E}_0{}' + E_3E_2{}'E_1E_0 + E_3{}'E_2E_1{}'E_0 + E_3{}'E_2E_1E_0{}' + E_3{}'E_2E_1E_0 + E_3E_2{}'E_1{}'E_0$$

3) ดำเนินการลดรูปฟังก์ชันการเข้าถึงข้อมูล จะได้ผลลัพธ์ดังนี้

$$E_3{}'E_2 + E_3E_2{}'$$

จากผลลัพธ์ที่ได้แสดงว่า เราต้องตรวจสอบบิตแมปเวกเตอร์ E_3 ที่มีค่าเท่ากับ 0 และบิตแมปเวกเตอร์ E_2 ที่มีค่าเท่ากับ 1 และตรวจสอบบิตแมปเวกเตอร์ E_3 ที่มีค่าเท่ากับ 1 และบิตแมปเวกเตอร์ E_2 ที่มีค่าเท่ากับ 0 และดำเนินการตรรกะ OR เพื่อหาคำตอบของการสอบถามนี้ ผลลัพธ์จากการสอบถามจะได้แถวที่ 1, 3, 6, 7, 9, 10 และ 10,000 เป็นคำตอบของการสอบถาม แสดงดังภาพประกอบ 3-26

E_3	E_2	E_1	E_0
0	1	1	1
0	0	0	1
1	0	0	1
0	0	1	0
1	1	0	0
0	1	0	0
1	0	1	1
0	0	0	0
0	1	0	1
1	0	0	0
.	.	.	.
.	.	.	.
.	.	.	.
0	1	1	0

$E_3{}'$	E_2	E_3	$E_2{}'$
1	1	0	0
1	0	0	1
0	0	1	1
1	0	0	1
0	1	1	0
1	1	0	0
0	0	1	1
1	0	0	1
1	1	0	0
0	0	1	1
.	.	.	.
.	.	.	.
.	.	.	.
1	1	0	0

$(E_3{}' \wedge E_2) + (E_3 \wedge E_2{}')$
1
0
1
0
0
1
1
0
1
1
.
.
.
1

ภาพประกอบ 3-26 ตัวอย่างการสอบถามข้อมูลแบบสมาชิก

3.3 การสอบถามข้อมูลแบบค่าเท่ากันบนดัชนีบิตแมปแบบเข้ารหัส

การสอบถามข้อมูลแบบค่าเท่ากันมีรูปแบบเงื่อนไขคือ “X = v” หมายถึง การสอบถามว่าบนแอตทริบิวต์ X มีแถวใดบ้างที่มีค่าเท่ากับ v การสอบถามแบบค่าเท่ากันบนดัชนีบิตแมปแบบเข้ารหัสที่ใช้เทคนิคการแบ่งกลุ่มได้นำข้อเด่นของตัวดำเนินการตรรกะ (AND และ NOT) มาเพิ่มประสิทธิภาพการสอบแบบค่าเท่ากันบนดัชนีบิตแมปแบบเข้ารหัสให้รวดเร็วมากยิ่งขึ้น เนื่องจากตัวดำเนินการตรรกะเป็นตัวดำเนินการพื้นฐานบนเครื่องคอมพิวเตอร์ และสามารถทำงานได้รวดเร็วและสนับสนุนการทำงานจากหน่วยประมวลผลกลางโดยตรง การเพิ่มประสิทธิภาพการสอบถามแบบค่าเท่ากันบนดัชนีบิตแมปแบบเข้ารหัสโดยใช้อัลกอริทึม EE-EBI (Enhanced Equality Encoded Bitmap Index) ตามภาพประกอบ 3-27 และนิยามสัญลักษณ์ที่ใช้ในอัลกอริทึมนี้ได้แสดงในตาราง 3-2

Algorithm: EE-EBI

Input: Specify value of condition query (QUERY_VALUE)

Output: Bitmap vector of answering query (ANS_VECTOR)

1. Initialize all bits in *ANS_VECTOR* be '1'.
2. Find *PATTERN* from binary encoding.
3. for i = 0 to $\lceil log_2 C \rceil$-1
4. if (E_i = 1) then
5. *ANS_VECTOR* = *ANS_VECTOR* $\wedge E_i$
6. else
7. *ANS_VECTOR* = *ANS_VECTOR* $\wedge \overline{E_i}$
8. Output *ANS_VECTOR*

ภาพประกอบ 3-27 อัลกอริทึม EE-EBI

ตาราง 3-2 ชื่อตัวแปรที่ใช้ในอัลกอริทึม EE-EBI

ชื่อตัวแปร	ความหมาย
C	คาร์ดินอลิตี้ของแอตทริบิวต์ที่นำมาทำดัชนี
N	จำนวนแถวทั้งหมดในตาราง
QUERY_VALUE	ค่าที่ถูกสอบถามในเงื่อนไขสอบถาม
ANS_VECTOR	บิตแมปเวกเตอร์ผลลัพธ์ มีขนาด N บิต
PATTERN	รูปแบบการเข้ารหัสของ QUERY_VALUE มีขนาด $\lceil log_2 C \rceil$ บิต

อัลกอริทึม EE-EBI เป็นขั้นตอนในการเพิ่มประสิทธิภาพการสอบถามข้อมูลแบบค่าเท่ากันบนดัชนีบิตแมปแบบเข้ารหัส จากภาพประกอบ 3-27 แสดงอัลกอริทึม EE-EBI ประกอบด้วยขั้นตอนการทำงาน 8 ขั้นตอน มีรายละเอียดของแต่ละบรรทัดดังนี้

บรรทัดที่ 1 กำหนดให้ทุกๆ บิตของบิตแมปเวกเตอร์ผลลัพธ์ (ANS_VECTOR) มีค่าเท่ากับ 1 นั่นคือ ANS_VECTOR = (1, 1, 1, 1, …, 1)

บรรทัดที่ 2 ค้นหาลำดับ และรูปแบบการเข้ารหัสของค่าที่สอบถามในเงื่อนไขการสอบถาม ตัวอย่างเช่น จากภาพประกอบ 3-25(ข) ค่า B มีลำดับ และรูปแบบการลงรหัสเป็น 0001 นั่นคือ PATTERN มีค่าเป็น 0001

บรรทัดที่ 3-7 เป็นการหาคำตอบของการสอบถาม ถ้า PATTERN มีค่าเท่ากับ 1 จะนำบิตแมปเวกเตอร์มาดำเนินการตรรกะ AND กับบิตแมปเวกเตอร์ ANS_VECTOR แต่ถ้า

PATTERN มีค่าเท่ากับ 0 จะต้องทำการกลับบิตของบิตแมปเวกเตอร์นั้นก่อนจึงจะนำมาดำเนินการตรรกะ AND กับบิตแมปเวกเตอร์ ANS_VECTOR

ตัวอย่างเช่น การสอบถามแบบค่าเท่ากันมีเงื่อนไข “X = B” จากรูปแบบการลงรหัส ดังภาพประกอบที่ 3-25(ข) B มีรูปแบบการเข้ารหัสเป็น 0001 ($E_3 = 0$, $E_2 = 0$, $E_1 = 0$ และ $E_0 = 1$) ดังนั้นการหาคำตอบของการสอบถามนี้สามารถนำบิตแมปเวกเตอร์ E_0 ดำเนินการตรรกะ AND กับบิตแมปเวกเตอร์ ANS_VECTOR ได้โดยตรง และนำบิตแมปเวกเตอร์ E_3, E_2 และ E_1 มาทำการกลับบิตก่อน แล้วจึงนำมาดำเนินการตรรกะ AND กับบิตแมปเวกเตอร์ ANS_VECTOR จะได้แถวที่ 2 เป็นคำตอบ แสดงดังภาพประกอบ 3-27

E_3	E_2	E_1	E_0
0	1	1	1
0	0	0	1
1	0	0	1
0	0	1	0
1	1	0	0
0	1	0	0
1	0	1	1
0	0	0	0
0	1	0	1
1	0	0	0
.	.	.	.
.	.	.	.
.	.	.	.
0	1	1	0

$E_3{}'$	$E_2{}'$	$E_1{}'$	E_0
1	0	0	1
1	1	1	1
0	1	1	1
1	1	0	0
0	0	1	0
1	0	1	0
0	1	0	1
1	1	1	0
1	0	1	1
0	1	1	0
.	.	.	.
.	.	.	.
.	.	.	.
1	0	0	0

ANS_VECTOR
0
1
0
0
0
0
0
0
0
0
.
.
.
0

ภาพประกอบ 3-28 ตัวอย่างการสอบถามข้อมูลแบบค่าเท่ากันเงื่อนไข “X = B”

3.4 ข้อเด่นของดัชนีบิตแมปแบบเข้ารหัสที่ใช้เทคนิคการจัดกลุ่มข้อมูล

ดัชนีบิตแมปแบบเข้ารหัสที่ใช้เทคนิคการจัดกลุ่มข้อมูลเป็นเทคนิคการทำดัชนีบิตแมปที่ใช้พื้นที่ในการจัดเก็บดัชนีน้อยที่สุดเช่นเดียวกับการทำดัชนีบิตแมปแบบเข้ารหัสทั่วไป เนื่องจากดัชนีบิตแมปแบบเข้ารหัสที่ใช้เทคนิคการจัดกลุ่มข้อมูลใช้จำนวนบิตแมปเวกเตอร์ในการสร้างดัชนีน้อยกว่าดัชนีบิตแมปแบบพื้นฐาน ดัชนีบิตแมปแบบแถว ดัชนีบิตแมปแบบช่วง ดัชนีบิตแมปแบบกระจาย และดัชนีบิตแมปแบบคู่กัน นอกจากนี้ยังมีประสิทธิภาพในการ

สอบถามข้อมูลแบบสมาชิกและการสอบถามข้อมูลแบบค่าเท่ากัน สำหรับการสอบถามข้อมูลแบบสมาชิกได้มีการลงรหัสค่าของแอตทริบิวต์ที่มีการสอบถามไปด้วยกันและสอบถามร่วมกันให้อยู่ในรูปแบบที่สามารถลดจำนวนบิตแมปเวกเตอร์ที่ต้องตรวจสอบให้เหลือน้อยที่สุด สำหรับการสอบถามแบบค่าเท่ากันได้ใช้ตัวดำเนินการตรรกะมาช่วยเพิ่มประสิทธิภาพการสอบถามแบบสมาชิกและการสอบถามข้อมูลแบบค่าเท่ากัน โดยมีการเข้ารหัสค่าของแอตทริบิวต์ที่มีการสอบถามพร้อมกันและสอบถามร่วมกันให้อยู่ในรูปแบบที่สามารถลดจำนวนบิตแมปเวกเตอร์ที่ต้องตรวจสอบให้เหลือน้อยที่สุด

3.5 ข้อจำกัดของดัชนีบิตแมปแบบเข้ารหัสที่ใช้เทคนิคการจัดกลุ่มข้อมูล

เนื่องจากการสร้างดัชนีบิตแมปแบบเข้ารหัสที่ใช้เทคนิคการจัดกลุ่มข้อมูลได้ใช้ Query Workload เข้ามาช่วยในการจัดกลุ่มและลงรหัสค่าของแอตทริบิวต์ ดังนั้นหากมีเงื่อนไขการสอบถามที่ไม่เคยสอบถามมาก่อนอาจจะไม่สามารถลดจำนวนบิตแมปเวกเตอร์ที่ต้องตรวจสอบให้เหลือน้อยที่สุดได้ นอกจากนี้ในการจัดกลุ่มข้อมูลได้ใช้ตาราง Bit Matrix เพื่อหากลุ่มข้อมูลที่สอบถามไปด้วยกันและสอบถามร่วมกัน ซึ่งหากคาร์ดินอลิตี้ของแอตทริบิวต์ที่นำมาสร้างดัชนีมีจำนวนมากขึ้นจะทำให้ขนาดของตาราง Bit Matrix มีขนาดใหญ่ขึ้นด้วยจึงใช้พื้นที่ในการสร้างตาราง Bit Matrix เพิ่มมากขึ้น

บทที่ 4

การวิเคราะห์และผลการทดลอง

สำหรับบทนี้กล่าวถึงการวิเคราะห์เปรียบเทียบค่าใช้จ่ายเกี่ยวกับพื้นที่ที่ใช้ในการจัดเก็บดัชนี ค่าใช้จ่ายเกี่ยวกับเวลาที่ใช้ในการสอบถามแบบสมาชิก ค่าใช้จ่ายเกี่ยวกับเวลาที่ใช้ในการสอบถามแบบค่าเท่ากัน และการแลกเปลี่ยนระหว่างประสิทธิภาพของพื้นที่กับเวลา (Space-Time Trade-off) ของดัชนีบิตแมปแบบเข้ารหัสที่ได้คิดค้นขึ้น (Cluster-EBI) และดัชนีบิตแมปที่เคยมีมาทั้ง 6 แบบได้แก่ ดัชนีบิตแมปแบบพื้นฐาน (Simple Bitmap Index) ดัชนีบิตแมปแบบแถว (Range Bitmap Index) ดัชนีบิตแมปแบบช่วง (Interval Bitmap Index) ดัชนีบิตแมปแบบกระจาย (Scatter Bitmap Index) ดัชนีบิตแมปแบบคู่กัน (Dual Bitmap Index) และดัชนีบิตแมปแบบเข้ารหัส (Encoded Bitmap Index)

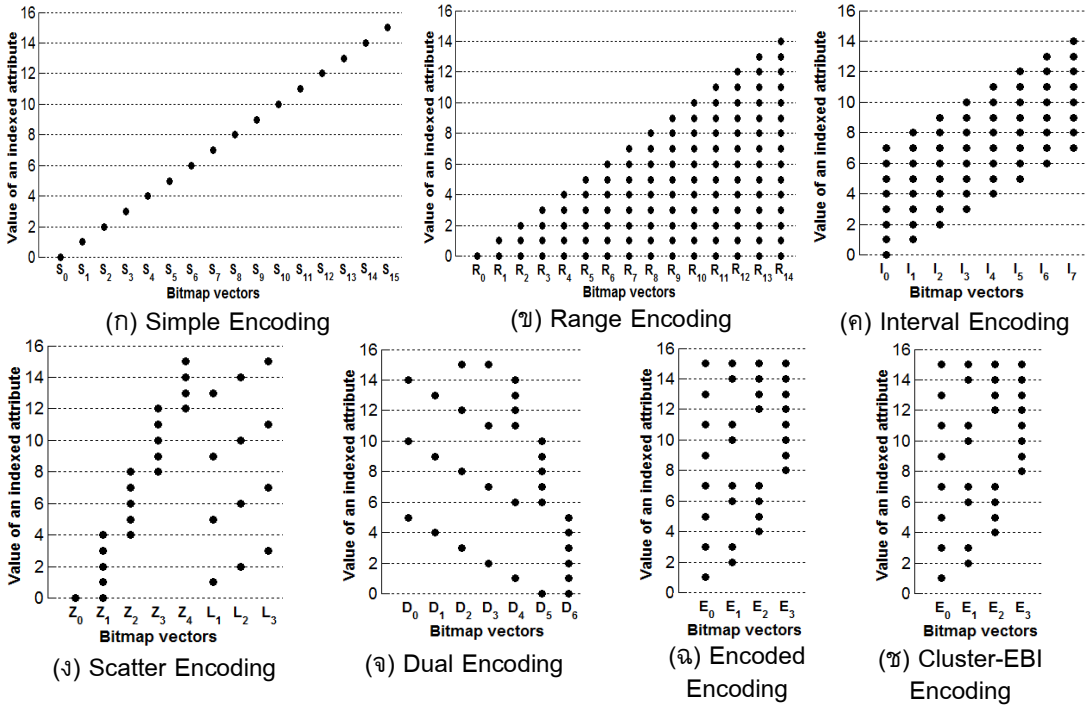
โดยทำการทดลองบนเครื่องคอมพิวเตอร์รุ่น Intel(R) Core(TM)2 Duo ที่มีหน่วยประมวลผลกลาง 2.00 GHz หน่วยความจำหลัก 2.00 GB ระบบปฏิบัติการ Window 7 Professional และใช้ภาษาซีในการเขียนโปรแกรม

ข้อมูลที่ใช้ในการทดลองเป็นชุดข้อมูลมาตรฐานจาก TPC-H Benchmark (Transaction Processing Performance Council, 2011) สำหรับการสอบถามข้อมูลแบบสมาชิกเลือกแอตทริบิวต์ที่นำมาดัชนีมีคาร์ดินอลิตี้เท่ากับ 1,000 มีจำนวนแถวเท่ากับ 5,000,000 แถว สำหรับการสอบถามข้อมูลแบบค่าเท่ากันเลือกแอตทริบิวต์ที่นำมาทำดัชนีมีคาร์ดินอลิตี้เท่ากับ 50, 150 และ 1,000 มีจำนวนแถวเท่ากับ 5,000,000 แถว

4.1 ค่าใช้จ่ายเกี่ยวกับพื้นที่ที่ใช้ในการจัดเก็บดัชนี

ดัชนีบิตแมปแต่ละแบบมีรูปแบบการลงรหัสที่แตกต่างกันจึงทำให้ประสิทธิภาพด้านพื้นที่การจัดเก็บดัชนีแตกต่างกันด้วย พิจารณารูปแบบการลงรหัสของดัชนีบิตแมปทั้ง 7 แบบแสดงดังภาพประกอบ 4.1 เมื่อกำหนดให้คาร์ดินอลิตี้ของแอตทริบิวต์ที่นำมาสร้างดัชนีเท่ากับ 16 ประกอบด้วยสมาชิกคือ {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15} Simple Bitmap Index ใช้พื้นที่มากที่สุดคือ 16 บิตแมปเวกเตอร์ดังภาพประกอบ 4-1(ก) Range Bitmap Index ใช้พื้นที่น้อยกว่า Simple Bitmap Index อยู่ 1 บิตแมปเวกเตอร์คือ 15 บิตแมปเวกเตอร์ดังภาพประกอบ 4-1(ข) Interval Bitmap Index ใช้พื้นที่เป็นครึ่งหนึ่งของ Simple Bitmap Index คือ 8 บิตแมปเวกเตอร์ดังภาพประกอบ 4-1(ค) Scatter Bitmap Index ใช้ 8 บิตแมปเวกเตอร์ดังภาพประกอบ 4-1(ง) Dual Bitmap Index ใช้ 7 บิตแมปเวกเตอร์ดังภาพประกอบ 4-1(จ) Encoded Bitmap Index ใช้ 4 บิตแมปเวกเตอร์ดังภาพประกอบ 4-1(ฉ) และ Cluster-EBI ใช้ 4

บิตแมปเวกเตอร์ดังภาพประกอบ 4-1(ช) โดยจุดทึบ คือบิตที่มีค่าเท่ากับ 1 ซึ่งใช้แทนค่าของแอตทริบิวต์ในแต่ละบิตแมปเวกเตอร์ ตัวอย่างเช่น จากรูปแบบการลงรหัสของ Range Bitmap Index ดังภาพประกอบ 4-1(ข) บิตแมปเวกเตอร์ R_2 ใช้แทนค่าของแอตทริบิวต์ที่อยู่ในช่วง [0,2] เป็นต้น พิจารณาการเปรียบเทียบพื้นที่ที่ใช้ในการจัดเก็บดัชนีบิตแมปทั้ง 7 แบบดังตาราง 4-1

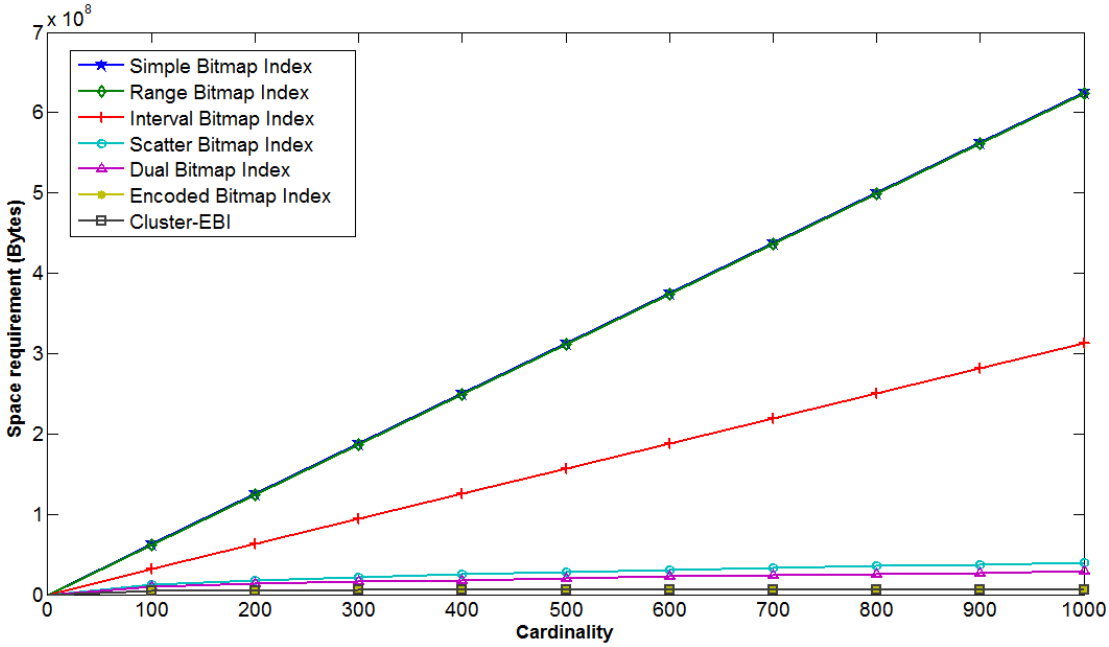


ภาพประกอบ 4-1 รูปแบบการลงรหัสของดัชนีบิตแมปทั้ง 7 แบบ เมื่อคาร์ดินอลิตี้เท่ากับ 16

ตาราง 4-1 พื้นที่ที่ใช้ในการจัดเก็บดัชนีบิตแมปทั้ง 7 แบบเมื่อ *C* คือ คาร์ดินอลิตี้ของแอตทริบิวต์ และ *N* คือ จำนวนแถวทั้งหมด

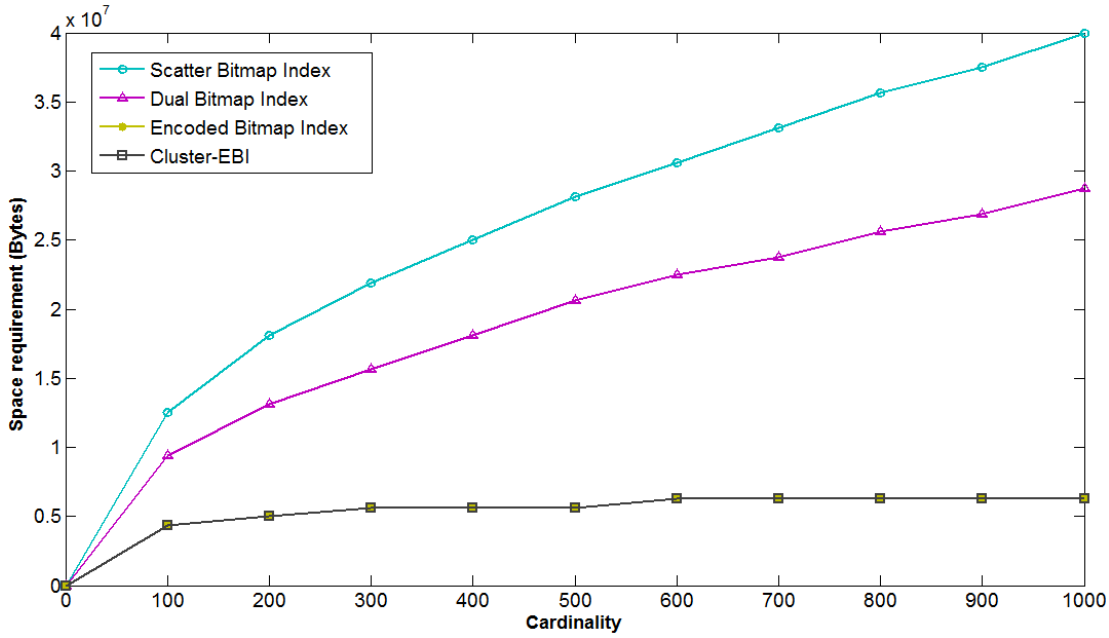
Bitmap Index	Space requirement (bits)
Simple	*NC*
Range	*N (C-1)*
Interval	$N\lceil C/2 \rceil$
Scatter	$N\lceil 2\sqrt{C} \rceil$
Dual	$N\lceil \sqrt{2C + 0.25} + 0.5 \rceil$
Encoded	$N\lceil log_2\, C \rceil$
Cluster-EBI	$N\lceil log_2\, C \rceil$

จากตาราง 4-1 จะเห็นได้ว่าการทำดัชนีบิตแมปบนแอตทริบิวต์ที่มี *N* แถว พื้นที่ที่ใช้ในการจัดเก็บดัชนีบิตแมปแต่ละชนิดจะแปรผันตรงกับคาร์ดินอลิตี้ของแอตทริบิวต์ที่นำมาทำดัชนี เมื่อเปรียบเทียบพื้นที่ที่ใช้ในการจัดเก็บดัชนีบิตแมปทั้ง 7 แบบ จะเห็นได้ว่า Simple Bitmap Index มีการใช้พื้นที่ในการจัดเก็บมากที่สุด รองลงมาคือ Range Bitmap Index, Interval Bitmap Index, Scatter Bitmap Index, Dual Bitmap Index, Encoded Bitmap Index และ Cluster-EBI ตามลำดับ



ภาพประกอบ 4-2 กราฟเปรียบเทียบพื้นที่ที่ใช้ในการสร้างดัชนีบิตแมปทั้ง 7 แบบ เมื่อแอตทริบิวต์ที่นำมาสร้างดัชนีมี 5,000,000 แถว (N = 5,000,000)

จากภาพประกอบ 4-2 แสดงความสัมพันธ์ระหว่างพื้นที่ที่ใช้ในการทำดัชนีกับคาร์ดินอลิตี้ของแอตทริบิวต์ที่นำมาทำดัชนีบิตแมปทั้ง 7 แบบ ซึ่งคาร์ดินอลิตี้มีค่าตั้งแต่ 0-1000 เมื่อเปรียบเทียบพื้นที่ที่ใช้ในการจัดเก็บดัชนีบิตแมปทั้ง 7 แบบ จะเห็นได้ว่า Simple Bitmap Index ใช้พื้นที่ในการจัดเก็บดัชนีมากกว่าดัชนีบิตแมปแบบอื่นๆ Interval Bitmap Index ใช้พื้นที่ในการจัดเก็บดัชนีน้อยกว่า Simple Bitmap Index และ Range Bitmap Index แต่ยังคงใช้พื้นที่ในการจัดเก็บดัชนีมากกว่า Scatter Bitmap Index, Dual Bitmap Index, Encoded Bitmap Index และ Cluster-EBI



ภาพประกอบ 4-3 กราฟเปรียบเทียบพื้นที่ที่ใช้ในการสร้างดัชนีบิตแมป 4 แบบ เมื่อแอตทริบิวต์ที่นำมาสร้างดัชนีมี 5,000,000 แถว (N = 5,000,000)

จากภาพประกอบ 4-3 แสดงความสัมพันธ์ระหว่างพื้นที่ที่ใช้ในการทำดัชนีกับคาร์ดินอลิตี้ของแอตทริบิวต์ที่นำมาทำ Scatter Bitmap Index, Dual Bitmap Index, Encoded Bitmap Index และ Cluster-EBI เมื่อเปรียบเทียบพื้นที่ที่ใช้ในการจัดเก็บดัชนีบิตแมปทั้ง 4 แบบ จะเห็นได้ว่าดัชนีบิตแมปที่ใช้พื้นที่ในการจัดเก็บดัชนีน้อยที่สุดคือ Encoded Bitmap Index และ Cluster-EBI ซึ่งใช้พื้นที่ในการจัดเก็บดัชนีเท่ากัน รองลงมาคือ Dual Bitmap Index และ Scatter Bitmap Index ตามลำดับ

โดยสรุป Encoded Bitmap Index และ Cluster-EBI มีประสิทธิภาพใช้พื้นที่ในการจัดเก็บดัชนีมากที่ที่สุด ซึ่งดัชนีทั้ง 2 แบบมีการใช้พื้นที่ในการจัดเก็บดัชนีเท่ากัน

4.2 ค่าใช้จ่ายเกี่ยวกับเวลาที่ใช้ในการสอบถามข้อมูลแบบสมาชิก

เนื่องจากดัชนีบิตแมปแต่ละแบบมีรูปแบบการลงรหัสที่แตกต่างกันส่งผลให้ใช้พื้นที่ในการจัดเก็บดัชนีแตกต่างกัน และขั้นตอนการสอบถามข้อมูลแบบสมาชิกแตกต่างกันด้วย รวมทั้งจำนวนบิตแมปเวกเตอร์ที่ทำการอ่านและจำนวนครั้งที่ดำเนินการตรรกะระหว่างบิตแมปเวกเตอร์ส่งผลให้เวลาที่ใช้ในการสอบถามข้อมูลแตกต่าง ซึ่งเวลาในการสอบถามข้อมูลแบบสมาชิก ประกอบด้วย เวลาที่ใช้ในการอ่านบิตแมปเวกเตอร์ หากต้องอ่านบิตแมปเวกเตอร์จำนวนมากจะทำให้ต้องใช้เวลามาก และเวลาที่ใช้ในการดำเนินการตรรกะระหว่างบิตแมปเวกเตอร์ หากมีการดำเนินการตรรกะระหว่างบิตแมปเวกเตอร์หลายครั้งก็จะทำให้ใช้เวลามาก

เช่นกัน พิจารณาจำนวนบิตแมปเวกเตอร์ที่ต้องอ่านและจำนวนครั้งในการดำเนินการตรรกะระหว่างบิตแมปเวกเตอร์เมื่อมีการสอบถามข้อมูลแบบสมาชิก ดังตาราง 4-2

ตาราง 4-2 การเปรียบเทียบจำนวนบิตแมปเวกเตอร์ที่ต้องอ่านและจำนวนครั้งในการดำเนินการตรรกะระหว่างบิตแมปเวกเตอร์ของดัชนีบิตแมปทั้ง 7 แบบ เมื่อ k คือจำนวนสมาชิกที่มีการสอบถาม

Bitmap Index	**Number of bitmap vectors scanned**		**Number of Boolean Operations**	
	best case	**worst case**	**best case**	**worst case**
Simple	k		k-1 OR	
Range	1	$2k$	0	$2k$-1 (k XOR, k-1 OR)
Interval	1	$2k$	0	$3k$-1 (k AND, k-1 OR, k NOT)
Scatter	1	$2k$	0	$2k$-1 (k AND, k-1 OR)
Dual	1	$2k$	0	$2k$-1 (k AND, k-1 OR)
Encoded	1	$\lceil log_2 C \rceil$ k	0	use mapping table, (k-1 OR)
Cluster-EBI	1	$\lceil log_2 C \rceil$ k	0	k-1 OR

จากตาราง 4-2 แสดงให้เห็นว่า การสอบถามแบบสมาชิกที่มีจำนวนสมาชิก k ค่า Simple Bitmap Index ต้องอ่านบิตแมปเวกเตอร์ k บิตแมปเวกเตอร์ และดำเนินการตรรกะ OR จำนวน k-1 ครั้ง ในกรณีที่ดีที่สุด Range Bitmap Index, Interval Bitmap Index, Scatter Bitmap Index และ Dual Bitmap Index อ่านบิตแมปเวกเตอร์เพียง 1 บิตแมปเวกเตอร์เท่านั้น และไม่มีการดำเนินการตรรกะ ในกรณีที่แย่ที่สุด Range Bitmap Index, Interval Bitmap Index, Scatter Bitmap Index และ Dual Bitmap Index ต้องอ่านบิตแมปเวกเตอร์ $2k$ บิตแมปเวกเตอร์ สำหรับ Range Bitmap Index มีการดำเนินการตรรกะจำนวน $2k$-1 (k XOR, k-1 OR) ครั้ง สำหรับ Interval Bitmap Index มีการดำเนินการตรรกะจำนวน $3k$-1 (k AND, k-1 OR, k NOT) ครั้ง สำหรับ Scatter Bitmap Index และ Dual Bitmap Index มีการดำเนินการตรรกะ $2k$-1 (k AND, k-1 OR) ครั้ง ดังนั้น Range Bitmap Index, Interval Bitmap Index, Scatter Bitmap Index และ Dual Bitmap Index ใช้เวลาในการสอบถามข้อมูลแบบสมาชิกใกล้เคียงกัน แต่ยังคงใช้เวลามากกว่า Simple Bitmap Index สำหรับ Encoded Bitmap Index มีจำนวนบิตแมปเวกเตอร์ที่ต้องอ่านมากที่สุด คือ $\lceil log_2 C \rceil$ k บิตแมปเวกเตอร์ มีการดำเนินการตรรกะ OR จำนวน k-1 ครั้ง และใช้การเทียบค่ากับตารางเทียบค่าเป็นผลให้เวลาในการสอบถามข้อมูลแบบสมาชิกบนดัชนีบิตแมปแบบเข้ารหัสทั่วไปใช้เวลามากที่สุด สำหรับ Cluster-EBI ในกรณีที่ดี

ที่สุด จะมีการอ่านบิตแมปเวกเตอร์เพียง 1 บิตแมปเวกเตอร์ และไม่มีการดำเนินการตรรกะ แต่ในบางกรณี Cluster-EBI อาจใช้เวลาในการสอบถามข้อมูลแบบสมาชิกมากกว่า Simple Bitmap Index, Range Bitmap Index, Interval Bitmap Index, Scatter Bitmap Index และ Dual Bitmap Index ซึ่งขึ้นอยู่กับประวัติของการสอบถามในอดีตที่นำมาสร้างรูปแบบการลงรหัส แต่จะใช้เวลาในการสอบถามข้อมูลแบบสมาชิกน้อยกว่า Encoded Bitmap Index

โดยสรุป Cluster-EBI ใช้เวลาในการสอบถามข้อมูลแบบสมาชิกน้อยกว่า Encoded Bitmap Index และมีโอกาสใช้เวลาในการสอบถามข้อมูลแบบสมาชิกน้อยกว่า Simple Bitmap Index, Range Bitmap Index, Interval Bitmap Index, Scatter Bitmap Index และ Dual Bitmap Index โดยแสดงให้เห็นจากผลการทดลอง

สำหรับประวัติของการสอบถามข้อมูลแบบสมาชิกในอดีตที่ใช้ในการทดลองจะทำการสุ่มการสอบถามข้อมูลแบบสมาชิกจำนวน 100 การสอบถาม โดยการหาค่าของแอตทริบิวต์ที่มีการสอบถามไปด้วยกันและสอบถามร่วมกันจากประวัติการสอบถามในอดีตดังกล่าวก่อนสร้าง Cluster-EBI ซึ่งในการหากลุ่มข้อมูลที่มีการสอบถามไปด้วยกันและสอบถามร่วมกัน จะกำหนดค่าความถี่ขั้นต่ำ 4 ค่า ได้แก่ 10% 20% 30% และ 40% และค่า Threshold 3 ค่า ได้แก่ 10 20 และ 30

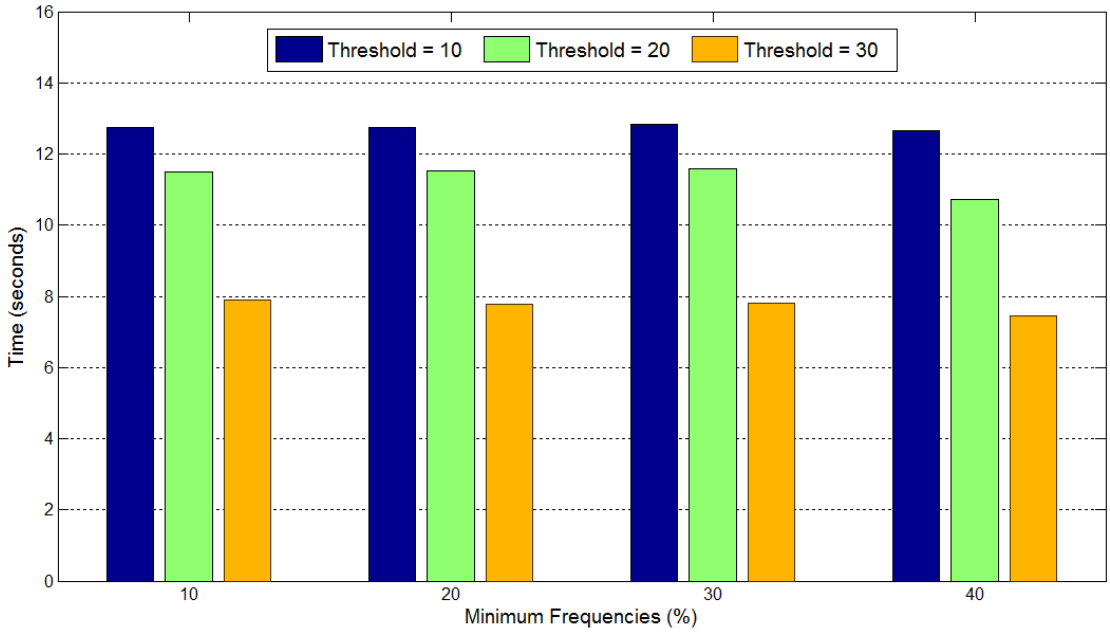
ในการทดลองกำหนดให้เวลาที่ใช้ในการสอบถามข้อมูลแบบสมาชิกบนดัชนีบิตแมปแต่ละแบบ ประกอบด้วย เวลาในการคำนวณหาบิตแมปเวกเตอร์ที่ต้องตรวจสอบ เวลาในการอ่านบิตแมปเวกเตอร์ที่ต้องตรวจสอบ เวลาในการดำเนินการตรรกะระหว่างบิตแมปเวกเตอร์ และการบันทึกคำตอบของการสอบถามลงในไฟล์ข้อมูล สำหรับ Encoded Bitmap Index และ Cluster-EBI จะรวมเวลาที่ใช้ในการค้นหารูปแบบการลงรหัสและเวลาในการคำนวณเพื่อลดรูปฟังก์ชันในการเข้าถึงข้อมูลด้วย

สำหรับการบันทึกผลการทดลองดำเนินการโดยการประมวลผลโปรแกรมเพื่อสอบถามข้อมูลแบบสมาชิกจำนวน 2 ชุดการสอบถาม แล้วบันทึกเวลาที่ใช้ในแต่ละการสอบถาม จากนั้นหาค่าเฉลี่ยของเวลาที่ใช้ในแต่ละการสอบถาม ซึ่งดำเนินการซ้ำทั้งหมด 3 รอบ ดังผลการทดลองต่อไปนี้

ตาราง 4-3 เวลาที่ใช้ในการสอบถามข้อมูลแบบสมาชิกของ Cluster-EBI บนชุดการสอบถามที่ 1

Minimum Frequency (%)	Time (seconds)		
	Threshold		
	10	20	30
10	12.7556	11.5076	7.8988
20	12.7452	11.5232	7.7896
30	12.8180	11.5700	7.8052
40	12.6568	10.7224	7.4620

จากตาราง 4-3 นำค่าเฉลี่ยของเวลาที่ใช้ในการสอบถามแบบสมาชิกของ Cluster-EBI มาเขียนกราฟได้ดังภาพประกอบ 4-4



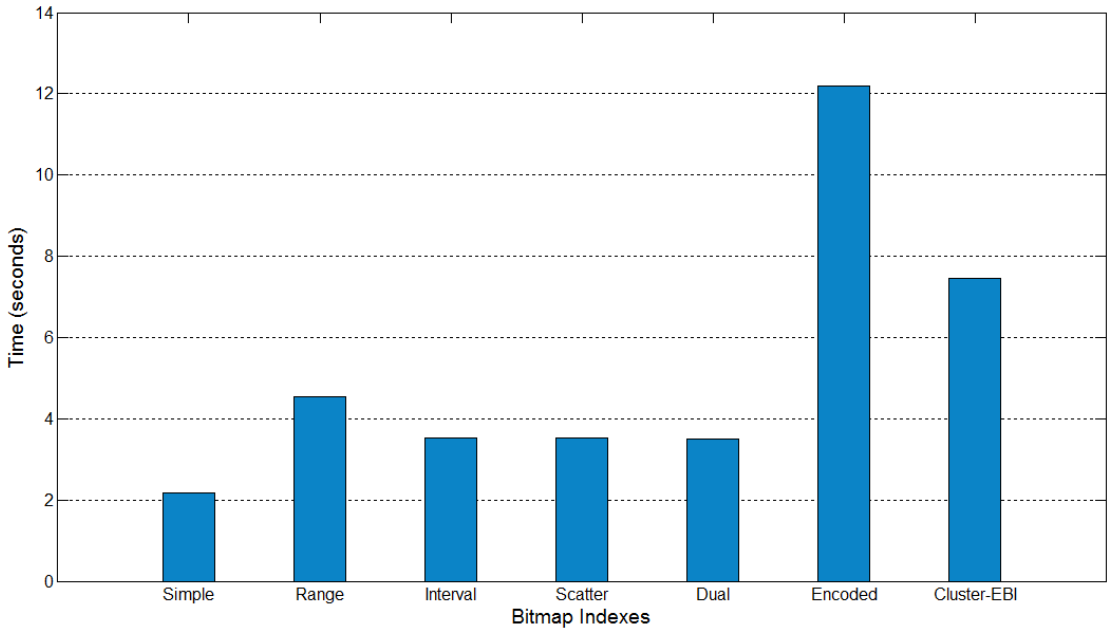
ภาพประกอบ 4-4 กราฟเปรียบเทียบเวลาที่ใช้ในการสอบถามแบบสมาชิกของ Cluster-EBI บนชุดการสอบถามที่ 1

สำหรับผลการทดลองการสอบถามข้อมูลแบบสมาชิกของ Cluster-EBI บนชุดการสอบถามที่ 1 แสดงดังตารางที่ 4-3 และภาพประกอบ 4-4 เมื่อกำหนดค่า Threshold เท่ากับ 10 Cluster-EBI จะใช้เวลาในการสอบถามข้อมูลแบบสมาชิกมาก แต่เมื่อกำหนดค่า Threshold เท่ากับ 30 Cluster-EBI จะใช้เวลาในการสอบถามข้อมูลแบบสมาชิกน้อยที่สุด และเมื่อกำหนดค่าความถี่ขั้นต่ำเท่ากับ 40% และค่า Threshold เท่ากับ 30 Cluster-EBI จะใช้เวลาสอบถาม

ข้อมูลแบบสมาชิกน้อยที่สุด การเปรียบเทียบเวลาที่ใช้ในการสอบถามข้อมูลแบบสมาชิกของดัชนีบิตแมปทั้ง 7 แบบบนชุดการสอบถามที่ 1 แสดงดังตาราง 4-4 โดยเลือกค่าความถี่ขั้นต่ำเท่ากับ 40% และค่า Threshold เท่ากับ 30 สำหรับ Cluster-EBI และนำค่าเฉลี่ยของเวลาที่ใช้ในการสอบถามแบบสมาชิกของ Cluster-EBI มาเขียนกราฟได้ดังภาพประกอบ 4-5

ตาราง 4-4 เวลาที่ใช้ในการสอบถามข้อมูลแบบสมาชิกของดัชนีบิตแมปทั้ง 7 แบบบนชุดการสอบถามที่ 1

ดัชนีบิตแมป	Simple	Range	Interval	Scatter	Dual	Encoded	Cluster-EBI
เวลาที่ใช้ในการสอบถาม (วินาที)	2.1684	4.5344	3.5356	3.5204	3.4944	12.1836	7.4620



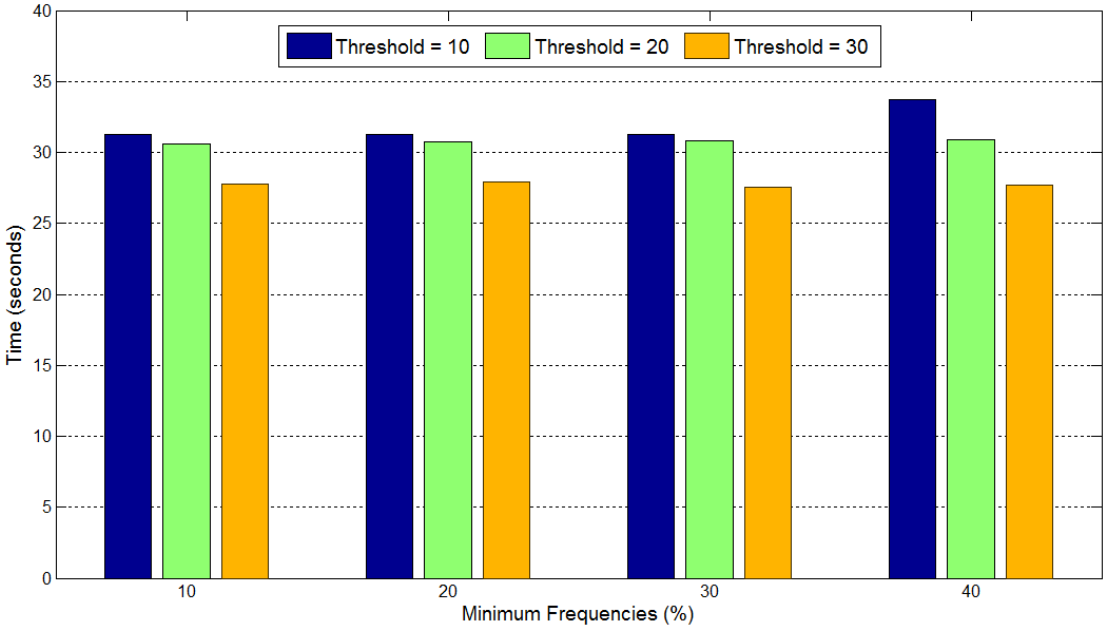
ภาพประกอบ 4-5 กราฟเปรียบเทียบเวลาที่ใช้ในการสอบถามแบบสมาชิกของดัชนีบิตแมปทั้ง 7 แบบ บนชุดการสอบถามที่ 1

จากตาราง 4-4 และภาพประกอบ 4-5 จะเห็นได้ว่า Simple Bitmap Index ใช้เวลาในการสอบถามข้อมูลแบบสมาชิกน้อยที่สุด ส่วน Range Bitmap Index ใช้เวลาในการสอบถามมากกว่า Simple Bitmap Index, Interval Bitmap Index, Scatter Bitmap Index และ Dual Bitmap Index เนื่องจากค่าข้อมูลที่สอบถามมีลักษณะเป็นช่วงข้อมูลจำนวนน้อยจึงทำให้

การดำเนินการตรรกะระหว่างบิตแมปเวกเตอร์มีจำนวนมาก สำหรับ Cluster-EBI เมื่อกำหนดค่าความถี่ขั้นต่ำในการหากลุ่มข้อมูลที่สอบถามไปด้วยกันและสอบถามร่วมกันเท่ากับ 40% และค่า Threshold เท่ากับ 30 ใช้เวลาในการสอบถามข้อมูลแบบสมาชิกน้อยกว่า Encoded Bitmap Index แต่ใช้เวลาในการสอบถามข้อมูลแบบสมาชิกมากกว่า Simple Bitmap Index, Range Bitmap Index, Interval Bitmap Index, Scatter Bitmap Index และ Dual Bitmap Index

ตาราง 4-5 เวลาที่ใช้ในการสอบถามข้อมูลแบบสมาชิกของ Cluster-EBI บนชุดการสอบถามที่ 2

Minimum Frequency (%)	**Time (seconds)**		
	Threshold		
	10	20	30
10	31.2936	30.6072	27.7626
20	31.2676	30.7372	27.8883
30	31.2468	30.7840	27.5288
40	33.7428	30.8932	27.6692



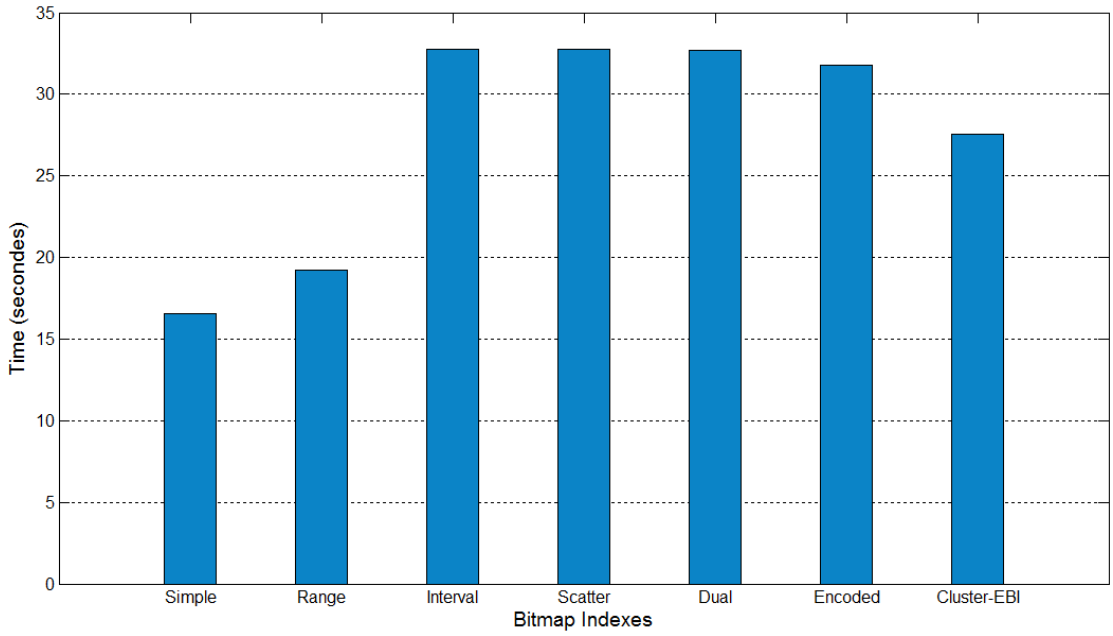
ภาพประกอบ 4-6 กราฟเปรียบเทียบเวลาที่ใช้ในการสอบถามแบบสมาชิกของ Cluster-EBI บนชุดการสอบถามที่ 2

สำหรับผลการทดลองการสอบถามข้อมูลแบบสมาชิกของ Cluster-EBI บนชุดการสอบถามที่ 2 แสดงดังตารางที่ 4-5 และภาพประกอบ 4-6 เมื่อกำหนดค่า Threshold เท่ากับ

10 Cluster-EBI จะใช้เวลาในการสอบถามข้อมูลแบบสมาชิกมากที่สุด แต่เมื่อกำหนดค่า Threshold เท่ากับ 30 Cluster-EBI จะใช้เวลาในการสอบถามข้อมูลแบบสมาชิกน้อยที่สุด และเมื่อกำหนดค่าความถี่ขั้นต่ำเท่ากับ 30% และค่า Threshold เท่ากับ 30 Cluster-EBI จะใช้เวลาสอบถามข้อมูลแบบสมาชิกน้อยที่สุด การเปรียบเทียบเวลาที่ใช้ในการสอบถามข้อมูลแบบสมาชิกของดัชนีบิตแมปทั้ง 7 แบบบนชุดการสอบถามที่ 1 แสดงดังตาราง 4-6 โดยเลือกค่าความถี่ขั้นต่ำเท่ากับ 40% และค่า Threshold เท่ากับ 30 สำหรับ Cluster-EBI และนำค่าเฉลี่ยของเวลาที่ใช้ในการสอบถามแบบสมาชิกของ Cluster-EBI มาเขียนกราฟได้ดังภาพประกอบ 4-7

ตาราง 4-6 เวลาที่ใช้ในการสอบถามข้อมูลแบบสมาชิกของดัชนีบิตแมปทั้ง 7 แบบ บนชุดการสอบถามที่ 2

ดัชนีบิตแมป	Simple	Range	Interval	Scatter	Dual	Encoded	Cluster-EBI
เวลาที่ใช้ในการสอบถาม (วินาที)	16.5256	19.2370	32.7704	32.7600	32.6820	31.7824	27.5288



ภาพประกอบ 4-7 กราฟเปรียบเทียบเวลาที่ใช้ในการสอบถามแบบสมาชิกของดัชนีบิตแมปทั้ง 7 แบบ บนชุดการสอบถามที่ 2

จากตาราง 4-6 และภาพประกอบ 4-7 จะเห็นได้ว่า Simple Bitmap Index ใช้เวลาในการสอบถามข้อมูลแบบสมาชิกน้อยที่สุด ส่วน Range Bitmap Index ใช้เวลาในการสอบถามข้อมูลแบบสมาชิกน้อยกว่า Interval Bitmap Index, Scatter Bitmap Index และ Dual Bitmap Index เนื่องจากค่าข้อมูลที่สอบถามมีลักษณะเป็นช่วงข้อมูลจำนวนมาก จึงทำให้การดำเนินการตรรกะระหว่างบิตแมปเวกเตอร์มีจำนวนน้อยส่งผลให้ Range Bitmap Index ใช้เวลาในการสอบถามข้อมูลแบบสมาชิกน้อยลงด้วย สำหรับ Cluster-EBI เมื่อกำหนดค่าความถี่ขั้นต่ำในการหากลุ่มข้อมูลที่สอบถามไปด้วยกันและสอบถามร่วมกันเท่ากับ 30% และค่า Threshold เท่ากับ 30 จะใช้เวลาในการสอบถามข้อมูลแบบสมาชิกเฉลี่ยน้อยกว่า Interval Bitmap Index, Scatter Bitmap Index, Dual Bitmap Index และ Encoded Bitmap Index แต่ใช้เวลาในการสอบถามข้อมูลแบบสมาชิกเฉลี่ยมากกว่า Simple Bitmap Index และ Range Bitmap Index

สรุปโดยภาพรวมจากการทดลองแสดงให้เห็นว่า Interval Bitmap Index, Scatter Bitmap Index และ Dual Bitmap Index ใช้เวลาในการสอบถามข้อมูลใกล้เคียงกัน ส่วน Encoded Bitmap Index ใช้เวลาในการสอบถามข้อมูลมากกว่าดัชนีบิตแมปแบบอื่นๆ แต่ในบางกรณี Encoded Bitmap Index มีโอกาสใช้เวลาในการสอบถามข้อมูลน้อยกว่า Interval Bitmap Index, Scatter Bitmap Index และ Dual Bitmap Index เนื่องจากรูปแบบการเข้ารหัสของแต่ละค่าการสอบถามสามารถลดรูปได้ จึงทำให้บิตแมปเวกเตอร์ที่ต้องตรวจสอบมีจำนวนน้อย แต่ยังคงใช้เวลาการสอบถามข้อมูลมากกว่า Simple Bitmap Index และ Range Bitmap Index สำหรับ Cluster-EBI ใช้เวลาในการสอบถามน้อยกว่า Encoded Bitmap Index และมีโอกาสใช้เวลาในการสอบถามน้อยกว่า Interval Bitmap Index, Range Bitmap Index, Scatter Bitmap Index และ Dual Bitmap Index ทั้งนี้ขึ้นอยู่กับประวัติการสอบถามที่ใช้ในการสร้าง Cluster-EBI

4.3 ค่าใช้จ่ายเกี่ยวกับเวลาที่ใช้ในการสอบถามข้อมูลแบบค่าเท่ากัน

รูปแบบการลงรหัสของดัชนีบิตแมปแต่ละแบบมีลักษณะที่แตกต่างกัน จึงทำให้ขั้นตอนการสอบถามข้อมูลแบบค่าเท่ากันของดัชนีบิตแมปแต่ละแบบแตกต่างกันด้วย ทั้งจำนวนบิตแมปเวกเตอร์ที่อ่าน และจำนวนครั้งในการดำเนินการตรรกะระหว่างบิตแมปเวกเตอร์ ปัจจัยเหล่านี้ส่งผลให้เวลาที่ใช้ในการสอบถามข้อมูลแบบค่าเท่ากันแตกต่างกัน ซึ่งเวลาที่ใช้ในการสอบถามข้อมูลแบบค่าเท่ากัน ประกอบด้วย เวลาที่ใช้ในการอ่านบิตแมปเวกเตอร์ หากต้องอ่านบิตแมปเวกเตอร์จำนวนมากจะทำให้ต้องใช้เวลามาก และเวลาที่ใช้ในการดำเนินการตรรกะระหว่างบิตแมปเวกเตอร์ หากมีการดำเนินการตรรกะระหว่างบิตแมปเวกเตอร์หลายครั้งก็จะทำให้ใช้เวลามากเช่นกัน พิจารณาจำนวนบิตแมปเวกเตอร์ที่ต้องอ่าน และจำนวนครั้งในการดำเนินการตรรกะระหว่างบิตแมปเวกเตอร์เมื่อมีการสอบถามข้อมูลแบบค่าเท่ากันของดัชนีบิตแมปทั้ง 7 แบบ ดังตาราง 4-7

ตาราง 4-7 การเปรียบเทียบจำนวนบิตแมปเวกเตอร์ที่ต้องอ่าน และจำนวนครั้งในการดำเนินการตรรกะระหว่างบิตแมปเวกเตอร์ของดัชนีบิตแมปทั้ง 7 แบบ เมื่อ C คือ คาร์ดินอลิตี้

ดัชนีบิตแมป	จำนวนบิตแมปเวกเตอร์ที่ต้องอ่าน	จำนวนครั้งในการดำเนินการตรรกะระหว่างบิตแมปเวกเตอร์
Simple	1	0
Range	2	1 XOR
Interval	2	1 AND, 1 NOT
Scatter	2	1 AND
Dual	2	1 AND
Encoded	$\lceil log_2 C \rceil$	Bit comparisons
Cluster-EBI	$\lceil log_2 C \rceil$	$\lceil log_2 C \rceil - 1$ AND, $\lceil log_2 C \rceil$ NOT

จากตาราง 4-7 แสดงให้เห็นว่า ในการสอบถามแบบค่าเท่ากัน Simple Bitmap Index ต้องอ่านบิตแมปเวกเตอร์เพียง 1 บิตแมปเวกเตอร์ และไม่มีการดำเนินการตรรกะ ดังนั้น Simple Bitmap Index จึงใช้เวลาในการสอบถามแบบค่าเท่ากันน้อยกว่าดัชนีบิตแมปแบบอื่นๆ สำหรับการสอบถามข้อมูลแบบค่าเท่ากันของ Range Bitmap Index, Interval Bitmap Index, Scatter Bitmap Index และ Dual Bitmap Index ต้องอ่านบิตแมปเวกเตอร์ 2 บิตแมปเวกเตอร์ สำหรับ Range Bitmap Index มีการดำเนินการตรรกะจำนวน 1 ครั้ง (1 XOR) Interval Bitmap Index มีการดำเนินการตรรกะจำนวน 2 ครั้ง (1 AND, 1 NOT) Scatter Bitmap Index และ Dual Bitmap Index มีการดำเนินการตรรกะ AND จำนวน 1 ครั้ง ดังนั้น Scatter Bitmap Index และ Dual Bitmap Index จึงใช้เวลาในการสอบถามข้อมูลเท่ากัน และใช้เวลาในการสอบถามข้อมูลแบบค่าเท่ากันใกล้เคียงกับ Range Bitmap Index และ Interval Bitmap Index สำหรับ Encoded Bitmap Index และ Cluster-EBI ต้องอ่านบิตแมปเวกเตอร์ $\lceil log_2 C \rceil$ บิตแมปเวกเตอร์ และ Encoded Bitmap Index ต้องใช้ตารางเทียบค่า สำหรับการสอบถามแบบค่าเท่ากันของ Encoded Bitmap Index มีการดำเนินการเปรียบเทียบรูปแบบการลงรหัสทุกบิตแมปเวกเตอร์ จึงทำให้ Encoded Bitmap Index ใช้เวลาในการสอบถามแบบค่าเท่ากันมากที่สุด ในขณะที่การสอบถามแบบค่าเท่ากันของ Cluster-EBI มีการดำเนินการตรรกะ AND จำนวนไม่เกิน $\lceil log_2 C \rceil - 1$ ครั้ง และดำเนินการตรรกะ NOT จำนวนไม่เกิน $\lceil log_2 C \rceil$ ครั้ง จึงใช้เวลาในการสอบถามข้อมูลน้อยกว่า Encoded Bitmap Index

โดยสรุป Cluster-EBI ใช้เวลาในการสอบถามข้อมูลแบบค่าเท่ากันน้อยกว่า Encoded Bitmap Index แต่ใช้เวลาในการสอบถามข้อมูลแบบค่าเท่ากันมากกว่า Simple

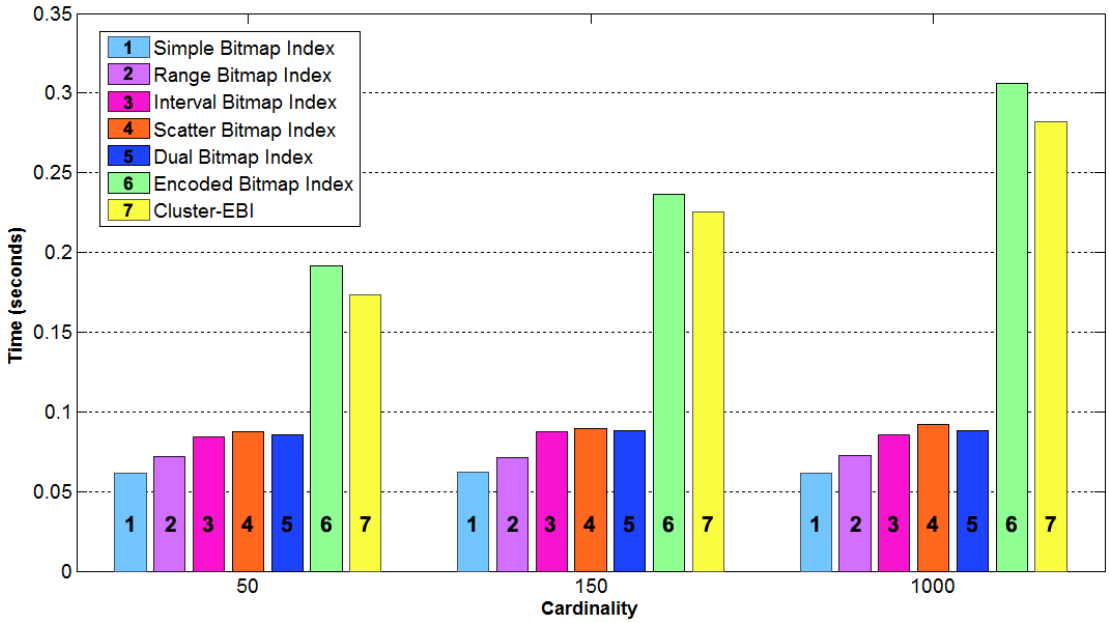
Bitmap Index, Range Bitmap Index, Interval Bitmap Index, Scatter Bitmap Index และ Dual Bitmap Index โดยแสดงให้เห็นจากผลการทดลอง

ในการทดลองกำหนดให้เวลาที่ใช้ในการสอบถามข้อมูลแบบค่าเท่ากันบนดัชนีบิตแมปแต่ละแบบประกอบด้วย เวลาในการคำนวณหาบิตแมปเวกเตอร์ที่ต้องตรวจสอบ เวลาในการอ่านบิตแมปเวกเตอร์ที่ต้องตรวจสอบ เวลาในการดำเนินการตรรกะระหว่างบิตแมปเวกเตอร์ และการบันทึกคำตอบของการสอบถามลงในไฟล์ข้อมูล สำหรับ Encoded Bitmap Index และ Cluster-EBI จะรวมเวลาที่ใช้ในการค้นหารูปแบบการลงรหัสด้วย

สำหรับการบันทึกผลการทดลองดำเนินการโดยประมวลผลโปรแกรมเพื่อสอบถามข้อมูลแบบค่าเท่ากันทีละค่าจนครบทุกค่าตั้งแต่ 0 - *C*-1 จากนั้นหาค่าเฉลี่ยของเวลาที่ใช้ในสอบถามค่าใดๆ ซึ่งดำเนินการซ้ำทั้งหมด 3 รอบแสดงดังตาราง 4-8 และหาค่าเฉลี่ยของเวลาที่ใช้ในการสอบถามแบบค่าเท่ากันของดัชนีบิตแมปทั้ง 7 แบบมาเขียนกราฟได้ดังภาพประกอบ 4-8

ตาราง 4-8 เวลาที่ใช้ในการสอบถามข้อมูลแบบค่าเท่ากันของดัชนีบิตแมปทั้ง 7 แบบ

Cardinality	**No.**	**Time (seconds)**						
		Simple	Range	Interval	Scatter	Dual	Encoded	Cluster-EBI
50	1	0.0780	0.0745	0.1040	0.1040	0.1014	0.1898	0.1742
	2	0.0572	0.0728	0.0754	0.0832	0.0780	0.1924	0.1742
	3	0.0494	0.0676	0.0728	0.0754	0.0780	0.1924	0.1716
	avg.	**0.0615**	**0.0717**	**0.0841**	**0.0875**	**0.0858**	**0.1915**	**0.1733**
150	1	0.0728	0.0676	0.1118	0.1170	0.1144	0.2340	0.2236
	2	0.0624	0.0728	0.0754	0.0728	0.0728	0.2392	0.2262
	3	0.0520	0.0728	0.0754	0.0780	0.0780	0.2366	0.2262
	avg.	**0.0624**	**0.0711**	**0.0875**	**0.0893**	**0.0884**	**0.2366**	**0.2253**
1000	1	0.0754	0.0858	0.0962	0.1118	0.1092	0.3042	0.2821
	2	0.0598	0.0676	0.0832	0.0832	0.0780	0.3042	0.2817
	3	0.0494	0.0650	0.0708	0.0806	0.0780	0.3094	0.2817
	avg.	**0.0615**	**0.0728**	**0.0858**	**0.0919**	**0.0884**	**0.3059**	**0.2818**



ภาพประกอบ 4-8 กราฟเปรียบเทียบเวลาที่ใช้ในการสอบถามแบบค่าเท่ากัน ของดัชนีบิตแมปทั้ง 7 แบบ เมื่อคาร์ดินอลิตี้เท่ากับ 50, 150 และ 1000

จากภาพประกอบ 4-8 จะเห็นได้ว่า เมื่อคาร์ดินอลิตี้เพิ่มสูงขึ้น Encoded Bitmap Index และ Cluster-EBI ต้องใช้เวลาในการสอบถามข้อมูลแบบค่าเท่ากันมากขึ้นด้วย หากพิจารณาในแต่ละคาร์ดินอลิตี้ จะเห็นได้ว่า Simple Bitmap Index ใช้เวลาในการสอบถามข้อมูลแบบค่าเท่ากันน้อยที่สุด ส่วน Interval Bitmap Index, Scatter Bitmap Index และ Dual Bitmap Index ใช้เวลาในการสอบถามข้อมูลแบบค่าเท่ากันใกล้เคียงกัน แต่ใช้เวลาในการสอบถามข้อมูลแบบค่าเท่ากันมากกว่า Simple Bitmap Index และ Range Bitmap Index สำหรับ Cluster-EBI ใช้เวลาในการสอบถามข้อมูลน้อยกว่า Encoded Bitmap Index แต่ใช้เวลาการสอบถามข้อมูลแบบค่าเท่ากันมากกว่า Simple Bitmap Index, Range Bitmap Index, Interval Bitmap Index, Scatter Bitmap Index และ Dual Bitmap Index

สาเหตุสำคัญที่ทำให้ Cluster-EBI ใช้เวลาในการสอบถามข้อมูลแบบค่าเท่ากันน้อยกว่า Encoded Bitmap Index คือขั้นตอนในการสอบถามข้อมูล กล่าวคือ เมื่อมีการสอบถามแบบค่าเท่ากัน Encoded Bitmap Index ต้องมีการอ่านบิตแมปเวกเตอร์จำนวน $\lceil log_2 C \rceil$ บิตแมปเวกเตอร์ และค้นหารูปแบบการลงรหัสในตารางเทียบค่า จากนั้นนำรูปแบบการลงรหัสมาเปรียบเทียบในดัชนี จึงทำให้ใช้เวลาในการสอบถามข้อมูลมาก แต่ Cluster-EBI มีการอ่านบิตแมปเวกเตอร์จำนวน $\lceil log_2 C \rceil$ บิตแมปเวกเตอร์ และหารูปแบบการลงรหัสจากลำดับค่าของแอตทริบิวต์ จากนั้นใช้การดำเนินการตรรกะ AND หรือ NOT บนแต่ละบิตแมปเวกเตอร์ ซึ่งตัวดำเนินการตรรกะ AND และ NOT เป็นตัวดำเนินการพื้นฐานของระบบ

คอมพิวเตอร์ สามารถทำงานได้รวดเร็ว ดังนั้น Cluster-EBI จึงสามารถสอบถามข้อมูลได้รวดเร็วกว่า Encoded Bitmap Index

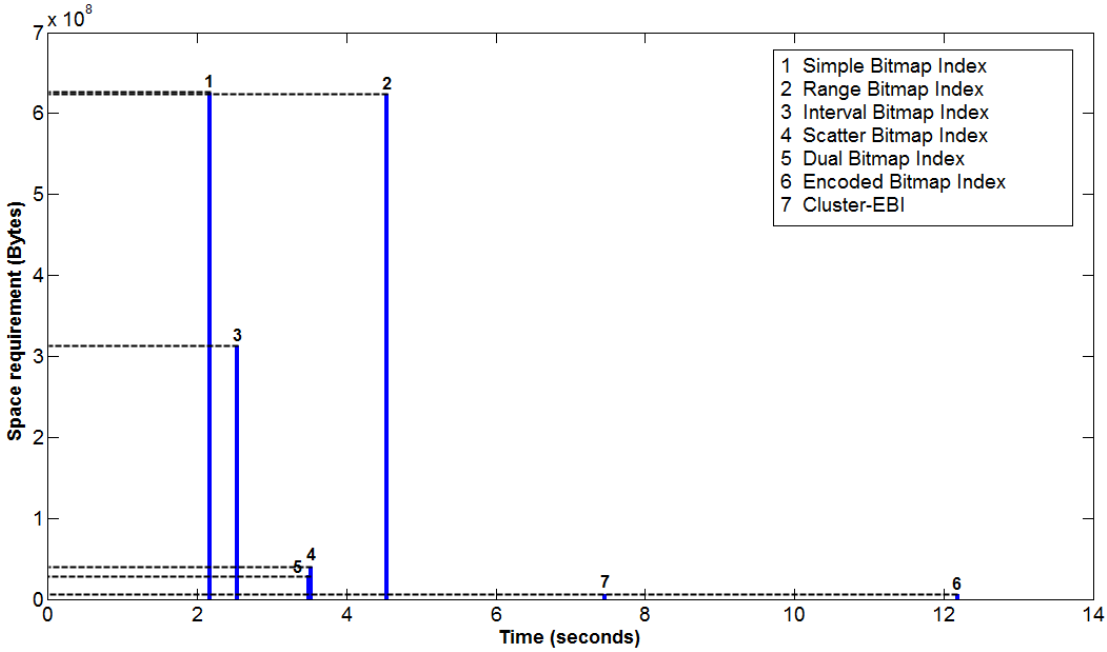
โดยสรุปแล้ว ดัชนีบิตแมปแบบเข้ารหัสที่มีการเพิ่มประสิทธิภาพการสอบถามแบบค่าเท่ากันมีประสิทธิภาพในการใช้เวลามากกว่าดัชนีบิตแมปแบบเข้ารหัสทั่วไป แต่อย่างไรก็ตาม ดัชนีบิตแมปแบบพื้นฐานเป็นดัชนีบิตแมปที่มีประสิทธิภาพในการใช้เวลาในการสอบถามข้อมูลแบบค่าเท่ากันมากที่สุด เพราะใช้เวลาในการสอบถามข้อมูลน้อยที่สุด

4.4 การแลกเปลี่ยนระหว่างประสิทธิภาพของพื้นที่กับเวลา (Space-Time Trade-off)

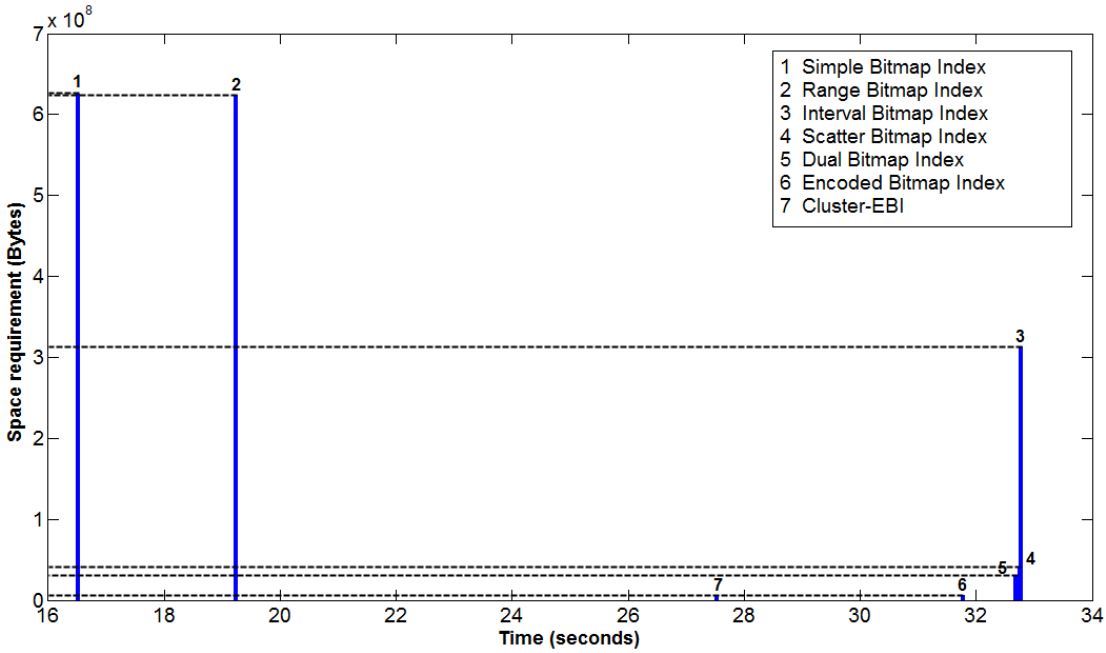
สำหรับการวิเคราะห์ประสิทธิภาพในแง่ของ Space-Time Trade-off ของดัชนีบิตแมปแต่ละแบบ โดยพิจารณาจากพื้นที่ใต้กราฟที่แสดงความสัมพันธ์ระหว่างพื้นที่ที่ใช้ในการจัดเก็บดัชนีกับเวลาที่ใช้ในการสอบถามข้อมูลของดัชนีบิตแมปแต่ละแบบ ซึ่งรูปสี่เหลี่ยมของพื้นที่ใต้กราฟมีฐานกว้างน้อย หมายความว่า ดัชนีบิตแมปดังกล่าวใช้เวลาในการสอบถามข้อมูลน้อย และรูปสี่เหลี่ยมของพื้นที่ใต้กราฟมีความสูงมาก หมายความว่า ดัชนีบิตแมปดังกล่าวใช้พื้นที่ในการจัดเก็บดัชนีมาก

4.4.1 การแลกเปลี่ยนระหว่างประสิทธิภาพของพื้นที่กับเวลาสำหรับการสอบถามข้อมูลแบบสมาชิก

จากหัวข้อ 4.1 และ 4.2 จะเห็นได้ว่า Cluster-EBI และ Encoded Bitmap Index มีประสิทธิภาพในการใช้พื้นที่จัดเก็บดัชนีดีที่สุด แต่ Cluster-EBI มีประสิทธิภาพด้านเวลาในการสอบถามข้อมูลแบบสมาชิกดีกว่า Encoded Bitmap Index และมีโอกาสมีประสิทธิภาพด้านเวลาในการสอบถามข้อมูลดีกว่าดัชนีบิตแมปแบบอื่นๆ ขึ้นอยู่กับประวัติของการสอบถามในอดีต สำหรับการวิเคราะห์ในแง่ Space-Time Trade-off ของการสอบถามข้อมูลแบบสมาชิกของดัชนีบิตแมปแต่ละแบบ โดยการคำนวณจากพื้นที่ใต้กราฟที่แสดงความสัมพันธ์ระหว่างพื้นที่ที่ใช้กับเวลาในการสอบถามข้อมูลแบบสมาชิกแต่ละแบบ แสดงดังภาพประกอบ 4-9 และ 4-10



ภาพประกอบ 4-9 กราฟแสดงความสัมพันธ์ระหว่างพื้นที่ที่ใช้กับเวลาในการสอบถามข้อมูลแบบสมาชิกของดัชนีบิตแมปทั้ง 7 แบบ บนชุดการสอบถามที่ 1



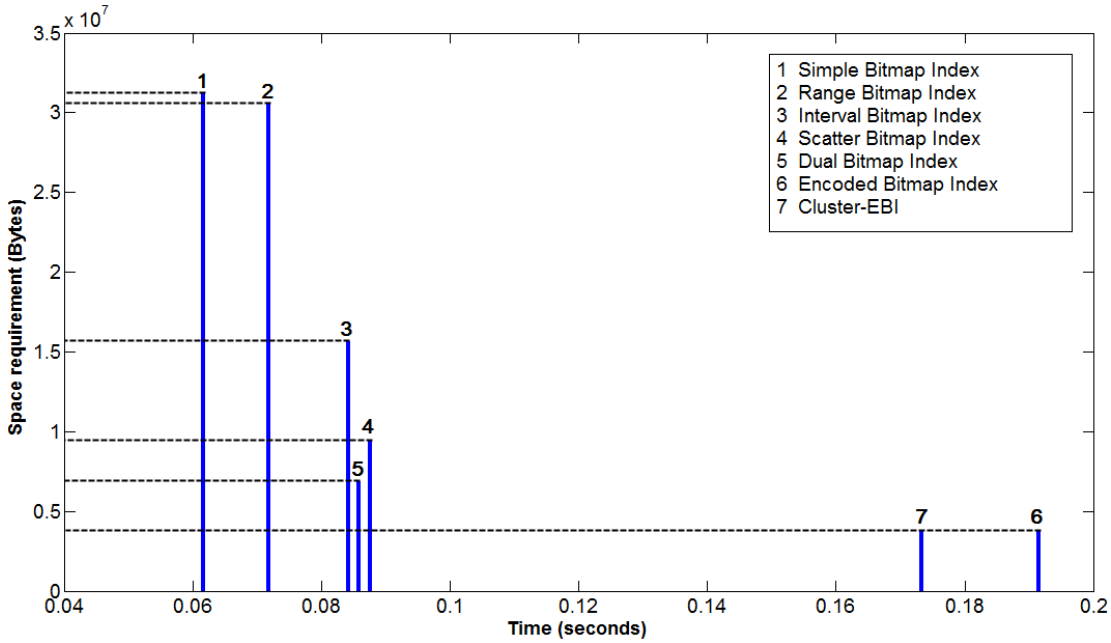
ภาพประกอบ 4-10 กราฟแสดงความสัมพันธ์ระหว่างพื้นที่ที่ใช้กับเวลาในการสอบถามข้อมูลแบบสมาชิกของดัชนีบิตแมปทั้ง 7 แบบ บนชุดการสอบถามที่ 2

จากภาพประกอบ 4-9 และ 4-10 เมื่อพิจารณาพื้นที่ใต้กราฟของดัชนีบิตแมปแต่ละแบบ พบว่า พื้นที่ใต้กราฟของ Simple Bitmap Index และพื้นที่ใต้กราฟของ Range

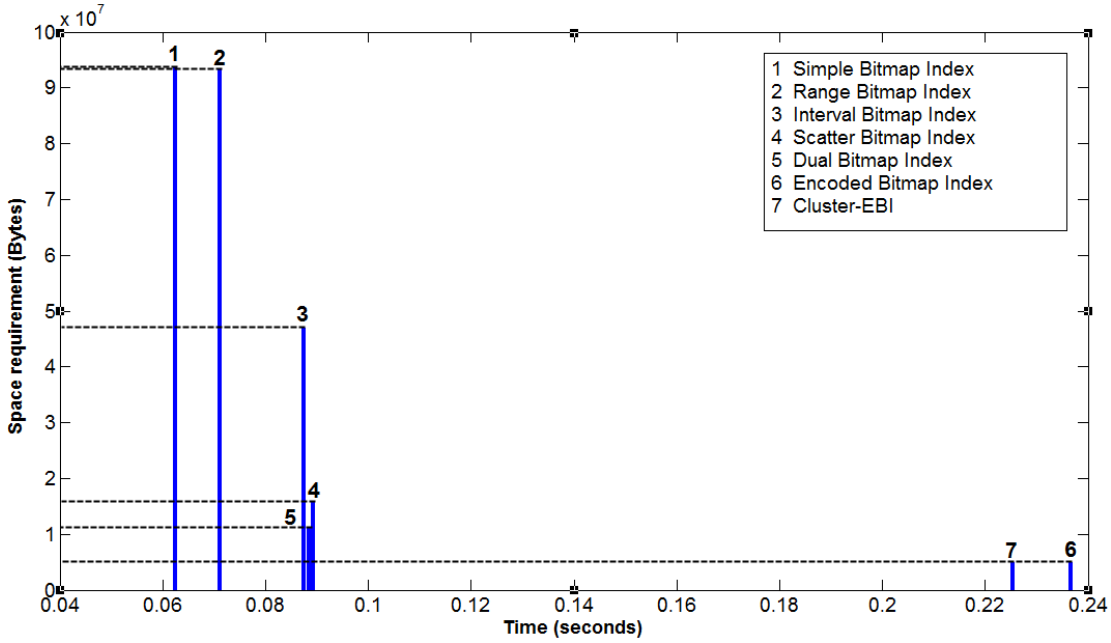
Bitmap Index มีลักษณะเป็นรูปสี่เหลี่ยมฐานแคบและมีความสูงมาก สำหรับ Simple Bitmap Index มีฐานกว้างน้อยกว่า Range Bitmap Index และมีความสูงมากกว่า Range Bitmap Index เล็กน้อย ดังนั้น Simple Bitmap Index จึงมีพื้นที่ใต้กราฟน้อยกว่า Range Bitmap Index ส่วนพื้นที่ใต้กราฟของ Interval Bitmap Index, Scatter Bitmap Index และ Dual Bitmap Index มีฐานกว้างใกล้เคียงกัน เนื่องจากดัชนีบิตแมปทั้ง 3 แบบใช้เวลาในการสอบถามข้อมูลแบบสมาชิกใกล้เคียงกัน แต่เมื่อพิจารณาจากความสูงของพื้นที่ใต้กราฟ พบว่า พื้นที่ใต้กราฟของ Interval Bitmap Index มีความสูงมากกว่าพื้นที่ใต้กราฟของ Scatter Bitmap Index และพื้นที่ใต้กราฟของ Dual Bitmap Index เนื่องจาก Interval Bitmap Index ใช้พื้นที่ในการจัดเก็บดัชนีมากกว่า Scatter Bitmap Index และ Dual Bitmap Index สำหรับพื้นที่ใต้กราฟของ Encoded Bitmap Index มีลักษณะเป็นรูปสี่เหลี่ยมที่มีฐานกว้าง เนื่องจากใช้เวลาในการสอบถามข้อมูลแบบสมาชิกมาก สำหรับพื้นที่ใต้กราฟของ Cluster-EBI มีความสูงของพื้นที่ใต้กราฟเท่ากับความสูงของพื้นที่ใต้กราฟของ Encoded Bitmap Index เนื่องจากใช้พื้นที่ในการจัดเก็บดัชนีเท่ากัน แต่พื้นที่ใต้กราฟของ Cluster-EBI มีฐานกว้างน้อยกว่า Encoded Bitmap Index เนื่องจาก Cluster-EBI ใช้เวลาในการสอบถามข้อมูลแบบสมาชิกน้อยกว่า Encoded Bitmap Index นอกจากนี้จากภาพประกอบ 4-10 พื้นที่ใต้กราฟของ Encoded Bitmap Index และ Cluster-EBI มีฐานกว้างน้อยกว่าพื้นที่ใต้กราฟของ Interval Bitmap Index, Scatter Bitmap Index และ Dual Bitmap Index เนื่องจากค่าข้อมูลที่สอบถามข้อมูลแบบสมาชิกบน Encoded Bitmap Index และ Cluster-EBI มีรูปแบบการเข้ารหัสที่สามารถลดรูปได้ดีจึงทำให้ใช้เวลาในการสอบถามข้อมูลแบบสมาชิกน้อยกว่าดัชนีบิตแมปแบบอื่นๆ และเมื่อเปรียบเทียบพื้นที่ใต้กราฟของดัชนีบิตแมปทั้ง 7 แบบจะเห็นได้ว่า พื้นที่ใต้กราฟของ Cluster-EBI น้อยที่สุด หมายความว่า Cluster-EBI มีประสิทธิภาพในแง่ Space-Time Trade-off ดีที่สุด

4.4.2 การแลกเปลี่ยนระหว่างประสิทธิภาพของพื้นที่กับเวลาสำหรับการสอบถามข้อมูลแบบค่าเท่ากัน

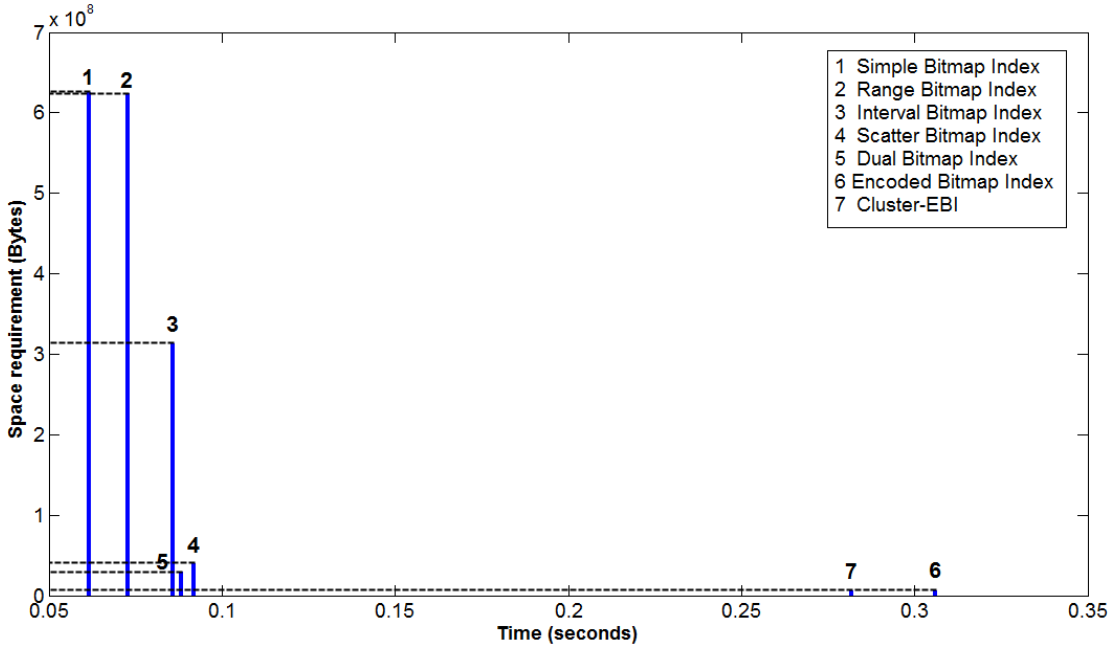
จากหัวข้อ 4.1 และ 4.3 จะเห็นได้ว่า Simple Bitmap Index มีประสิทธิภาพด้านเวลาในการสอบถามข้อมูลแบบค่าเท่ากันดีที่สุด แต่มีประสิทธิภาพทางด้านพื้นที่จัดเก็บดัชนีแย่ที่สุด ส่วน Cluster-EBI มีประสิทธิภาพในการใช้พื้นที่จัดเก็บดัชนีดีที่สุด นั่นคือใช้พื้นที่จัดเก็บดัชนีเท่ากับ Encoded Bitmap Index ซึ่งใช้พื้นที่น้อยที่สุด สำหรับการสอบถามข้อมูลแบบค่าเท่ากัน Cluster-EBI มีประสิทธิภาพด้านเวลาดีกว่า Encoded Bitmap Index สำหรับการวิเคราะห์ในแง่ของ Space-Time Trade-off ของการสอบถามข้อมูลแบบค่าเท่ากันของดัชนีบิตแมปแต่ละแบบ โดยการคำนวณจากพื้นที่ใต้กราฟที่แสดงความสัมพันธ์ระหว่างพื้นที่ที่ใช้กับเวลาในการสอบถามข้อมูลแบบค่าเท่ากันของดัชนีบิตแมปแต่ละแบบ แสดงดังภาพประกอบ 4-11, 4-12 และ 4-13 เมื่อคาร์ดินอลิตี้เท่ากับ 50, 150 และ 1000 ตามลำดับ



ภาพประกอบ 4-11 กราฟแสดงความสัมพันธ์ระหว่างพื้นที่ที่ใช้กับเวลาในการสอบถามข้อมูลแบบค่าเท่ากันของดัชนีบิตแมปทั้ง 7 แบบ เมื่อคาร์ดินอลิตี้เท่ากับ 50



ภาพประกอบ 4-12 กราฟแสดงความสัมพันธ์ระหว่างพื้นที่ที่ใช้กับเวลาในการสอบถามข้อมูลแบบค่าเท่ากันของดัชนีบิตแมปทั้ง 7 แบบ เมื่อคาร์ดินอลิตี้เท่ากับ 150



ภาพประกอบ 4-13 กราฟแสดงความสัมพันธ์ระหว่างพื้นที่ที่ใช้กับเวลาในการสอบถามข้อมูลแบบค่าเท่ากันของดัชนีบิตแมปทั้ง 7 แบบ เมื่อคาร์ดินอลิตี้เท่ากับ 1000

จากภาพประกอบ 4-11, 4-12 และ 4-13 เมื่อพิจารณาพื้นที่ใต้กราฟของ Simple Bitmap Index และ Range Bitmap Index จะเห็นได้ว่ามีลักษณะเป็นรูปสี่เหลี่ยมที่มีฐานกว้างน้อยและความสูงใกล้เคียงกัน โดยเฉพาะพื้นที่ใต้กราฟของ Simple Bitmap Index มีฐานกว้างน้อยกว่าพื้นที่ใต้กราฟของ Range Bitmap Index และมีความสูงมากกว่า Range Bitmap Index เล็กน้อย ส่วนพื้นที่ใต้กราฟของ Interval Bitmap Index, Scatter Bitmap Index และ Dual Bitmap Index มีลักษณะเป็นรูปสี่เหลี่ยมที่มีฐานใกล้เคียงกัน เนื่องจากดัชนีบิตแมปทั้ง 3 แบบใช้เวลาในการสอบถามข้อมูลแบบค่าเท่ากันใกล้เคียงกัน แต่เมื่อพิจารณาความสูงของพื้นที่ใต้กราฟของดัชนีบิตแมปทั้ง 3 แบบจะพบว่า ความสูงพื้นที่ใต้กราฟของ Interval Bitmap Index สูงที่สุด รองลงมาคือความสูงของพื้นที่ใต้กราฟของ Scatter Bitmap Index และความสูงของพื้นที่ใต้กราฟของ Dual Bitmap Index ตามลำดับ สำหรับพื้นที่ใต้กราฟของ Cluster-EBI มีลักษณะเป็นรูปสี่เหลี่ยมฐานกว้างน้อยกว่าพื้นที่ใต้กราฟของ Encoded Bitmap Index เนื่องจาก Cluster-EBI ใช้เวลาในการสอบถามข้อมูลมากกว่า Encoded Bitmap Index แต่เมื่อพิจารณาความสูงของพื้นที่ใต้กราฟของ Cluster-EBI จะเห็นได้ว่าความสูงพื้นที่ใต้กราฟของ Cluster-EBI เท่ากับความสูงของ Encoded Bitmap Index และน้อยกว่าดัชนีบิตแมปแบบอื่นๆ เมื่อเปรียบเทียบพื้นที่ใต้กราฟของดัชนีบิตแมปทั้ง 7 แบบจะเห็นได้ว่าเมื่อคาร์ดินอลิตี้เท่ากับ 50 และ 150 พื้นที่ใต้กราฟของ Cluster-EBI ใกล้เคียงกับพื้นที่ใต้กราฟของ Dual Bitmap Index ซึ่งมีพื้นที่ใต้กราฟน้อยที่สุดและเมื่อคาร์ดินอลิตี้เท่ากับ 1000 พื้นที่ใต้กราฟของ Cluster-EBI น้อย

ที่สุดหมายความว่า Cluster-EBI มีประสิทธิภาพในแง่ของ Space-Time Trade-off ดีที่สุดเมื่อแอตทริบิวต์ที่นำมาทำดัชนีมีคาร์ดินอลิตี้เพิ่มสูงขึ้น

บทที่ 5

บทสรุปและข้อเสนอแนะ

งานวิทยานิพนธ์นี้ได้นำเสนอเทคนิคการเพิ่มประสิทธิภาพของดัชนีบิตแมปแบบเข้ารหัสสำหรับการสอบถามข้อมูลแบบสมาชิกและการสอบถามข้อมูลแบบค่าเท่ากันด้วยเทคนิคการจัดกลุ่มข้อมูล (Data Clustering) มาช่วยในการหารูปแบบการลงรหัสที่เหมาะสมและการสร้างดัชนีบิตแมปแบบเข้ารหัส โดยขั้นตอนการเพิ่มประสิทธิภาพของดัชนีบิตแมปแบบเข้ารหัส ประกอบด้วย 4 ขั้นตอน คือ ขั้นตอนที่ 1 การเตรียมข้อมูล (Data Preparation) ขั้นตอนที่ 2 กระบวนการจัดกลุ่มค่าของแอตทริบิวต์ (Clustering Process) ขั้นตอนที่ 3 การลงรหัสค่าของแอตทริบิวต์ (Binary Encoding) และขั้นตอนที่ 4 การสร้างดัชนีบิตแมปแบบเข้ารหัส (Bitmap Index Creation) ผลลัพธ์ที่ได้ คือมีประสิทธิภาพในแง่ของพื้นที่ที่ใช้จัดเก็บดัชนีกับเวลาที่ใช้การสอบถามข้อมูลแบบสมาชิกและการสอบถามข้อมูลแบบค่าเท่ากันดีที่สุด เมื่อแอตทริบิวต์ที่นำมาทำดัชนีมีคาร์ดินอลิตี้สูง

5.1.บทสรุป

ระบบคลังข้อมูลเป็นฐานข้อมูลที่มีขนาดใหญ่ ซึ่งมีการเก็บรวบรวมข้อมูลจำนวนมากตั้งแต่อดีตจนถึงปัจจุบันเพื่อช่วยสนับสนุนในการวิเคราะห์และการตัดสินใจของผู้บริหาร ดังนั้นการสอบถามข้อมูลในระบบคลังข้อมูลจำเป็นต้องค้นหาข้อมูลเป็นจำนวนมากทำให้ใช้เวลาในการสอบถามข้อมูลเป็นเวลานาน ซึ่งเทคนิคการทำดัชนีบิตแมปเป็นที่นิยมอย่างสูงในการเพิ่มประสิทธิภาพการสอบถามข้อมูลที่ซับซ้อนในระบบคลังข้อมูล เนื่องจากเป็นวิธีการทำดัชนีที่สามารถดำเนินการตรรกะระดับบิตบนดัชนีได้โดยตรงก่อนเข้าถึงข้อมูลจริงจึงสนับสนุนการทำงานของฮาร์ดแวร์ทำให้การประมวลผลข้อมูลมีความรวดเร็ว โดยที่ผ่านมามีการพัฒนาขั้นตอนการสร้างดัชนีบิตแมปต่อ ๆ กันมาเพื่อให้สอดคล้องกับพื้นที่ที่ใช้จัดเก็บกับเวลาที่ใช้ในการสอบถามข้อมูลให้มีประสิทธิภาพมากยิ่งขึ้น

ในอดีตมีเทคนิคการทำดัชนีบิตแมปที่น่าสนใจ จากการศึกษา สามารถแบ่งเทคนิคการทำดัชนีบิตแมปได้เป็น 2 กลุ่ม ได้แก่ การทำดัชนีแบบบีบอัดดัชนีบิตแมปและการทำดัชนีแบบขยายแนวคิดของดัชนีบิตแมป สำหรับงานวิทยานิพนธ์นี้สนใจเทคนิคการทำดัชนีแบบขยายแนวคิดของดัชนีบิตแมป เนื่องจากเทคนิคการทำดัชนีแบบขยายแนวคิดของดัชนีบิตแมปสามารถดำเนินการตรรกะบนดัชนีได้โดยตรง การขยายแนวคิดของดัชนีบิตแมปที่น่าสนใจได้แก่ ดัชนีบิตแมปแบบพื้นฐาน ดัชนีบิตแมปแบบแถว ดัชนีบิตแมปแบบช่วง ดัชนีบิตแมปแบบกระจาย ดัชนีบิตแมปแบบคู่กัน และดัชนีบิตแมปแบบเข้ารหัส ซึ่งดัชนีบิตแมปแต่ละแบบมี

วิธีการสร้างดัชนีและการสอบถามข้อมูลที่แตกต่างกันจึงทำให้ประสิทธิภาพด้านพื้นที่ที่ใช้จัดเก็บดัชนีและเวลาที่ใช้ในการสอบถามข้อมูลแตกต่างกันออกไป นั่นคือ ดัชนีบิตแมปแบบพื้นฐานและดัชนีบิตแมปแบบแถวมีประสิทธิภาพด้านพื้นที่ที่ใช้จัดเก็บดัชนีใกล้เคียงกัน โดยดัชนีบิตแมปแบบพื้นฐานมีประสิทธิภาพด้านพื้นที่ที่ใช้จัดเก็บดัชนีแย่ที่สุด แต่ประสิทธิภาพด้านเวลาที่ใช้ในการสอบถามข้อมูลดีที่สุด ส่วนดัชนีบิตแมปแบบแถวมีประสิทธิภาพด้านเวลาที่ใช้ในการสอบถามข้อมูลใกล้เคียงกับดัชนีบิตแมปแบบช่วง ดัชนีบิตแมปแบบกระจาย และดัชนีบิตแมปแบบคู่กัน และประสิทธิภาพด้านพื้นที่ที่ใช้จัดเก็บดัชนี ดัชนีบิตแมปแบบคู่กันใช้พื้นที่จัดเก็บดัชนีน้อยกว่าดัชนีบิตแมปแบบกระจาย ดัชนีบิตแมปแบบกระจายใช้พื้นที่น้อยกว่าดัชนีบิตแมปแบบช่วง และดัชนีบิตแมปแบบช่วงใช้พื้นที่น้อยกว่าดัชนีบิตแมปแบบแถว ส่วนดัชนีบิตแมปแบบเข้ารหัสมีประสิทธิภาพด้านพื้นที่ที่ใช้จัดเก็บดัชนีดีที่สุด แต่ประสิทธิภาพด้านเวลาที่ใช้ในการสอบถามข้อมูลแย่ที่สุด

ในงานวิทยานิพนธ์นี้ได้นำเสนอขั้นตอนการเพิ่มประสิทธิภาพของดัชนีบิตแมปแบบเข้ารหัสสำหรับการสอบถามข้อมูลแบบสมาชิกและการสอบถามข้อมูลแบบค่าเท่ากัน โดยใช้หลักการที่ว่า จัดกลุ่มแต่ละค่าของแอตทริบิวต์ที่มีการสอบถามไปด้วยกันและสอบถามร่วมกันให้อยู่ในกลุ่มเดียวกัน เมื่อมีการสอบถามค่าของแอตทริบิวต์ด้วยกันอีกจะสามารถลดจำนวนบิตแมปเวกเตอร์ให้เหลือน้อยที่สุดได้ทำให้ประสิทธิภาพด้านเวลาที่ใช้ในการสอบถามข้อมูลแบบสมาชิกเพิ่มขึ้น สำหรับการสอบถามแบบค่าเท่ากันใช้ประโยชน์ของตัวดำเนินการตรรกะต้นทุนต่ำที่มีอยู่บนเครื่องคอมพิวเตอร์ปัจจุบันมาช่วยเพิ่มประสิทธิภาพด้านเวลาที่ใช้ในการสอบถามข้อมูลแบบค่าเท่ากัน ซึ่งสามารถพิจารณาเปรียบเทียบลักษณะที่สำคัญของดัชนีบิตแมปแบบเข้ารหัสที่ใช้เทคนิคการจัดกลุ่มข้อมูลที่นำเสนอกับดัชนีบิตแมปแบบอื่นๆ ที่เคยมีมา ได้ดังตาราง 5-1

ตาราง 5-1 ลักษณะที่สำคัญของดัชนีบิตแมปทั้ง 7 แบบ เมื่อ C คือคาร์ดินอลิตี้

ลักษณะสำคัญ		ดัชนีบิตแมป						
		แบบพื้นฐาน	แบบแถว	แบบช่วง	แบบกระจาย	แบบคู่กัน	แบบเข้ารหัส	ที่ใช้เทคนิคการจัดกลุ่มข้อมูล
จำนวนบิตแมปเวกเตอร์ที่ใช้ในการแทนแต่ละค่าของแอตทริบิวต์		1	$\leq C$-1	$\leq \lfloor \frac{C}{2} \rfloor$	2	2	$\lceil log_2 C \rceil$	$\lceil log_2 C \rceil$
จำนวนค่าของแอตทริบิวต์ที่ถูกแทนด้วยแต่ละบิตแมปเวกเตอร์		1	$\leq C$-1	$\lfloor \frac{C}{2} \rfloor$	$\leq \lceil \sqrt{C} \rceil + 1$	$\lceil \sqrt{2C + 0.25} + 0.5 \rceil - 1$	C	C
จำนวนบิตแมปเวกเตอร์ที่ใช้ในการแทนค่าของแอตทริบิวต์ที่มีคาร์นอลิตี้เท่ากับ C (พื้นที่)		C	$C - 1$	$\lceil \frac{C}{2} \rceil$	$\lceil 2\sqrt{C} \rceil$	$\lceil \sqrt{2C + 0.25} + 0.5 \rceil$	$\lceil log_2 C \rceil$	$\lceil log_2 C \rceil$
จำนวนบิตแมปเวกเตอร์ที่อ่าน : จำนวนครั้งในการดำเนินการตรรกะสำหรับการสอบถามแบบสมาชิก (เวลา)	กรณีดีที่สุด	*k*:*k*-1 OR	1:0	1:0	1:0	1:0	1:0	1:0
	กรณีแย่ที่สุด		2*k*:2*k*-1 (*k* XOR, *k*-1 OR)	2*k*:3*k*-1 (*k* AND, *k*-1 OR, *k* NOT)	2*k*:2*k*-1 (*k* AND, *k*-1 OR)	2*k*:2*k*-1 (*k* AND, *k*-1 OR)	$\lceil log_2 C \rceil k$: use mapping table, *k*-1 OR	$\lceil log_2 C \rceil$ *k* :*k*-1 OR

ลักษณะสำคัญ	ดัชนีบิตแมป						
	แบบพื้นฐาน	แบบแถว	แบบช่วง	แบบกระจาย	แบบคู่กัน	แบบเข้ารหัส	ที่ใช้เทคนิคการจัดกลุ่มข้อมูล
จำนวนบิตแมปเวกเตอร์ที่อ่าน : จำนวนครั้งในการดำเนินการตรรกะสำหรับการสอบถามแบบค่าเท่ากัน (เวลา)	1:0	2:1 XOR	2:2 (1 AND, 1 NOT)	2:1 (AND	2:1 (AND)	$\lceil log_2 C \rceil$: Bit comparisons	$\lceil log_2 C \rceil$: $\lceil log_2 C \rceil$ -1 AND, $\lceil log_2 C \rceil$ NOT

จากตาราง 5-1 จะเห็นได้ว่า ดัชนีบิตแมปแต่ละแบบมีลักษณะบางประการที่แตกต่างกันส่งผลให้มีข้อดีและข้อจำกัดที่แตกต่างกัน เมื่อพิจารณาการสอบถามแบบสมาชิกและการสอบถามแบบค่าเท่ากันพบว่า ดัชนีบิตแมปแต่ละแบบมีลักษณะดังนี้

- ดัชนีบิตแมปแบบพื้นฐาน แต่ละบิตแมปเวกเตอร์ใช้แทนค่าแอตทริบิวต์ได้เพียงค่าเดียว ดังนั้นในการสอบถามข้อมูลแบบสมาชิก k ค่า จะต้องตรวจสอบ k บิตแมปเวกเตอร์ และมีการดำเนินการตรรกะ OR จำนวน k-1 ครั้ง สำหรับการสอบถามแบบค่าเท่ากันจะต้องตรวจสอบเพียงบิตแมปเวกเตอร์เดียวเท่านั้นและไม่มีการดำเนินการตรรกะ
- ดัชนีบิตแมปแบบแถว ใช้จำนวนบิตแมปเวกเตอร์ในการแทนค่าแต่ละค่าของแอตทริบิวต์จำนวนมาก ($\leq C-1$ บิตแมปเวกเตอร์) เพราะในการสอบถามแต่ละค่ามีการอ่านเพียง 2 บิตแมปเวกเตอร์เท่านั้นจึงควรใช้ 2 บิตแมปเวกเตอร์ในการแทนแต่ละค่าของแอตทริบิวต์ สำหรับการสอบถามข้อมูลแบบสมาชิกจำนวน k ค่า ในกรณีที่ดีที่สุดจะต้องตรวจสอบเพียงบิตแมปเวกเตอร์เดียวและไม่มีการดำเนินการตรรกะ ในกรณีที่แย่ที่สุดจะต้องตรวจสอบ 2k บิตแมปเวกเตอร์ ดำเนินการตรรกะ XOR จำนวน k ครั้ง และดำเนินการตรรกะ OR จำนวน k-1 ครั้ง สำหรับการสอบถามแบบค่าเท่ากันจะต้องตรวจสอบบิตแมปเวกเตอร์ 2 บิตแมปเวกเตอร์ ดำเนินการตรรกะ XOR จำนวน 1 ครั้ง
- ดัชนีบิตแมปแบบช่วง ใช้จำนวนบิตแมปเวกเตอร์ในการแต่ค่าของแอตทริบิวต์จำนวนมาก ($\leq \lfloor C/2 \rfloor$ บิตแมปเวกเตอร์) เช่นเดียวกับดัชนีบิตแมปแบบแถว สำหรับการสอบถามข้อมูลแบบสมาชิกจำนวน k ค่า ในกรณีที่ดีที่สุดจะต้องตรวจสอบเพียงบิตแมปเวกเตอร์เดียวและไม่มีการดำเนินการตรรกะ ในกรณีที่แย่ที่สุดจะต้องตรวจสอบ 2k

บิตแมปเวกเตอร์ ดำเนินตรรกะ AND จำนวน *k* ครั้ง ดำเนินตรรกะ OR จำนวน *k*-1 ครั้ง และดำเนินตรรกะ NOT จำนวน *k* ครั้ง สำหรับการสอบถามข้อมูลแบบค่าเท่ากัน จะต้องตรวจสอบบิตแมปเวกเตอร์ 2 บิตแมปเวกเตอร์ ดำเนินการตรรกะ AND จำนวน 1 ครั้ง และดำเนินการตรรกะ NOT จำนวน 1 ครั้ง จึงใช้เวลาในการสอบถามข้อมูลแบบสมาชิกและการสอบถามข้อมูลแบบค่าเท่ากันใกล้เคียงกับดัชนีบิตแมปแบบแถว

- ดัชนีบิตแมปแบบกระจาย ใช้จำนวนบิตแมปเวกเตอร์ในการแทนค่าแต่ละค่าของแอตทริบิวต์จำนวน 2 บิตแมปเวกเตอร์ แต่มีบางบิตแมปเวกเตอร์ที่ถูกใช้แทนค่าของแอตทริบิวต์อย่างไม่คุ้มค่า โดยเฉพาะบิตแมปเวกเตอร์ Z_0 ที่ใช้แทนค่าของแอตทริบิวต์ได้เพียงค่าเดียวเท่านั้น ดังนั้นสำหรับการสอบถามแบบสมาชิกจำนวน *k* ค่า ในกรณีที่ดีที่สุดจะต้องตรวจสอบเพียงบิตแมปเวกเตอร์เดียวและไม่มีการดำเนินการตรรกะ ในกรณีที่แย่ที่สุดจะต้องตรวจสอบ 2*k* บิตแมปเวกเตอร์ ดำเนินการตรรกะ AND จำนวน *k* ครั้ง และดำเนินการตรรกะ OR จำนวน *k*-1 ครั้ง สำหรับการสอบถามข้อมูลแบบค่าเท่ากัน จะต้องตรวจสอบ 2 บิตแมปเวกเตอร์ และมีการดำเนินการตรรกะ AND จำนวน 1 ครั้ง จึงใช้เวลาในการสอบถามข้อมูลแบบสมาชิก และเวลาในการสอบถามข้อมูลแบบค่าเท่ากันใกล้เคียงกับดัชนีบิตแมปแบบช่วง
- ดัชนีบิตแมปแบบคู่กัน ใช้จำนวนบิตแมปเวกเตอร์ได้คุ้มค่ามากขึ้น แต่ยังคงใช้ 2 บิตแมปเวกเตอร์ในการแทนค่าแต่ละค่าของแอตทริบิวต์ สำหรับการสอบถามข้อมูลแบบสมาชิกจำนวน *k* ค่า ในกรณีที่ดีที่สุดจะต้องตรวจสอบเพียงบิตแมปเวกเตอร์เดียว ในกรณีที่แย่ที่สุดจะต้องตรวจสอบ 2*k* บิตแมปเวกเตอร์ ดำเนินตรรกะ AND จำนวน *k* ครั้ง และดำเนินการตรรกะ OR จำนวน *k*-1 ครั้ง สำหรับการสอบถามข้อมูลแบบค่าเท่ากัน จะต้องตรวจสอบ 2 บิตแมปเวกเตอร์ และมีการดำเนินการตรรกะ AND จำนวน 1 ครั้ง ดังนั้นดัชนีบิตแมปแบบคู่กันจึงใช้เวลาในการสอบถามข้อมูลแบบสมาชิกและการสอบถามข้อมูลแบบค่าเท่ากันใกล้เคียงกับดัชนีบิตแมปแบบช่วง และดัชนีบิตแมปแบบกระจาย
- ดัชนีบิตแมปแบบเข้ารหัสทั่วไป ใช้จำนวนบิตแมปเวกเตอร์ในการแทนค่าแต่ละค่าของแอตทริบิวต์ทุกบิตแมปเวกเตอร์ ($\lceil log_2 C \rceil$ บิตแมปเวกเตอร์) ทำให้การสอบข้อมูลแต่ละค่าต้องตรวจสอบทุกบิตแมปเวกเตอร์เปรียบเทียบกับตารางเทียบค่า สำหรับการสอบถามข้อมูลแบบสมาชิกจำนวน *k* ค่า ในกรณีที่ดีที่สุดจะต้องตรวจสอบเพียงบิตแมปเวกเตอร์เดียว ในกรณีที่แย่ที่สุดจะต้องตรวจสอบ $\lceil log_2 C \rceil k$ บิตแมปเวกเตอร์เปรียบเทียบกับตารางเทียบค่า และดำเนินการตรรกะ OR จำนวน *k*-1 ครั้ง สำหรับการสอบถามข้อมูลค่าเท่ากัน จะต้องตรวจสอบ $\lceil log_2 C \rceil$ บิตแมปเวกเตอร์เปรียบเทียบกับตารางเทียบค่า จึงใช้เวลาในการสอบถามข้อมูลแบบสมาชิกและการสอบถามข้อมูลแบบค่าเท่ากันมากที่สุด
- ดัชนีบิตแมปแบบเข้ารหัสที่ใช้เทคนิคการจัดกลุ่มข้อมูล ใช้จำนวนบิตแมปเวกเตอร์ในการแทนค่าแต่ละค่าของแอตทริบิวต์ทุกบิตแมปเวกเตอร์ ($\lceil log_2 C \rceil$ บิตแมปเวกเตอร์) เช่นเดียวกับดัชนีบิตแมปแบบเข้ารหัสทั่วไป สำหรับการสอบถามข้อมูลแบบสมาชิก

จำนวน k ค่า ในกรณีที่ดีที่สุดจะต้องตรวจสอบเพียงบิตแมปเวกเตอร์เดียว ในกรณีที่แย่ที่สุดจะต้องตรวจสอบ $\lceil log_2 C \rceil k$ บิตแมปเวกเตอร์เปรียบกับลำดับค่าของแอตทริบิวต์ที่ได้จากการจัดกลุ่มข้อมูล และดำเนินการตรรกะ OR จำนวน k-1 ครั้ง แต่สำหรับค่าของแอตทริบิวต์ที่เคยถูกสอบบ่อยๆ จะสามารถลดจำนวนบิตแมปเวกเตอร์ให้เหลือน้อยที่สุดได้ ในกรณีที่ดีที่สุดจะตรวจสอบเพียงหนึ่งบิตแมปเวกเตอร์ และไม่มีการดำเนินการตรรกะจึงมีโอกาสใช้เวลาในการสอบถามข้อมูลแบบสมาชิกดีที่สุด แต่ในกรณีที่แย่ที่สุด กล่าวคือกรณีเป็นการสอบถามค่าของแอตทริบิวต์ที่ไม่เคยถูกสอบถามบ่อยจะไม่สามารถลดจำนวนบิตแมปเวกเตอร์ให้เหลือน้อยที่สุดได้ ดังนั้นเวลาที่ใช้ในการสอบถามข้อมูลแบบสมาชิกบนดัชนีบิตแมปแบบเข้ารหัสที่ใช้เทคนิคการจัดกลุ่มข้อมูล จึงขึ้นอยู่กับประวัติการสอบถามในอดีต สำหรับการสอบถามแบบค่าเท่ากันจะต้องตรวจสอบ $\lceil log_2 C \rceil$ บิตแมปเวกเตอร์เปรียบเทียบกับลำดับค่าของแอตทริบิวต์และมีการดำเนินการตรรกะ AND จำนวน $\lceil log_2 C \rceil - 1$ ครั้งหรือมีการดำเนินการตรรกะ NOT ไม่เกิน $\lceil log_2 C \rceil$ ครั้งจึงใช้เวลาในการสอบถามข้อมูลแบบค่าเท่ากันน้อยกว่าดัชนีบิตแมปแบบเข้ารหัสทั่วไป

ในการคิดค้นเทคนิคการสร้างดัชนีบิตแมปแบบต่างๆ มีวัตถุประสงค์ในการนำไปใช้และความต้องการที่แตกต่างกัน ได้แก่ รูปแบบการสอบถามที่ต้องการ เช่น การสอบถามแบบสมาชิก และการสอบถามแบบค่าเท่ากัน เป็นต้น หรือความต้องการเกี่ยวกับประสิทธิภาพในแง่ต่างๆ เช่น ประสิทธิภาพในแง่เวลา ประสิทธิภาพในแง่พื้นที่ และประสิทธิภาพในแง่ Space-Time Trade-off สำหรับวิทยานิพนธ์นี้ระบุรูปแบบการสอบถามข้อมูลเป็นการสอบถามข้อมูลแบบสมาชิกและการสอบถามข้อมูลแบบค่าเท่ากัน จึงขอสรุปประสิทธิภาพที่ต้องการสำหรับการสอบถามข้อมูลแบบสมาชิก การสอบถามข้อมูลแบบค่าเท่ากัน และดัชนีบิตแมปที่ควรเลือกใช้ แสดงดังตาราง 5-2

ตาราง 5-2 สรุปประสิทธิภาพที่ต้องการสำหรับการสอบถามข้อมูลแบบสมาชิก การสอบถามข้อมูลแบบค่าเท่ากัน และดัชนีบิตแมปที่ควรเลือกใช้

รูปแบบการสอบถาม	ประสิทธิภาพที่ต้องการ	ดัชนีบิตแมปที่ควรเลือกใช้
แบบสมาชิก	ประสิทธิภาพในแง่ของเวลา	ดัชนีบิตแมปแบบพื้นฐานหรือดัชนีบิตแมปแบบเข้ารหัสที่ใช้เทคนิคการจัดกลุ่มข้อมูล
	ประสิทธิภาพในแง่ของพื้นที่	ดัชนีบิตแมปแบบเข้ารหัสทั่วไปและดัชนีบิตแมปแบบเข้ารหัสที่ใช้เทคนิคการจัดกลุ่มข้อมูล
	ประสิทธิภาพในแง่ Space-Time Trade-off	ดัชนีบิตแมปแบบเข้ารหัสที่ใช้เทคนิคการจัดกลุ่มข้อมูล เมื่อมีคาร์ดินอลิตี้สูง
แบบค่าเท่ากัน	ประสิทธิภาพในแง่ของเวลา	ดัชนีบิตแมปแบบพื้นฐาน
	ประสิทธิภาพในแง่ของพื้นที่	ดัชนีบิตแมปแบบเข้ารหัสทั่วไปและดัชนีบิตแมปแบบเข้ารหัสที่ใช้เทคนิคการจัดกลุ่มข้อมูล
	ประสิทธิภาพในแง่ Space-Time Trade-off	ดัชนีบิตแมปแบบเข้ารหัสที่ใช้เทคนิคการจัดกลุ่มข้อมูล เมื่อมีคาร์ดินอลิตี้สูง

จากผลการวิเคราะห์และผลการทดลองเปรียบเทียบกับดัชนีบิตแมปแบบเข้ารหัสที่ใช้เทคนิคการจัดกลุ่มข้อมูลที่คิดค้นขึ้นกับดัชนีบิตแมปที่เคยมีมา พบว่า ดัชนีบิตแมปแบบพื้นฐานใช้เวลาในการสอบถามข้อมูลแบบสมาชิกและการสอบถามข้อมูลแบบค่าเท่ากันน้อยแต่สุด แต่ต้องสูญเสียพื้นที่ในการจัดเก็บดัชนีมากที่สุดจึงเหมาะสำหรับการสร้างดัชนีบนแอตทริบิวต์ที่มีคาร์ดินอลิตี้ต่ำและต้องการประสิทธิภาพในแง่ของเวลา ส่วนดัชนีบิตแมปแบบเข้ารหัสที่ใช้เทคนิคการจัดกลุ่มข้อมูลใช้พื้นที่ในการจัดเก็บดัชนีเท่ากับดัชนีบิตแมปแบบเข้ารหัสทั่วไป ซึ่งน้อยกว่าดัชนีบิตแมปแบบอื่นๆ จึงเหมาะสำหรับการสร้างดัชนีบนแอตทริบิวต์ที่มีคาร์ดินอลิตี้สูง และต้องการประสิทธิภาพในแง่ของพื้นที่ โดยดัชนีบิตแมปแบบเข้ารหัสที่ใช้เทคนิคการจัดกลุ่มข้อมูลมีประสิทธิภาพในแง่เวลาที่ใช้ในการสอบถามข้อมูลแบบสมาชิกและการสอบถามข้อมูลแบบค่าเท่ากันมากกว่าดัชนีบิตแมปแบบเข้ารหัสทั่วไปและโดยเฉพาะการสอบถามข้อมูลแบบสมาชิกมีโอกาสดีกว่าดัชนีบิตแมปแบบอื่นๆ ซึ่งขึ้นกับประวัติการสอบถามในอดีต ในกรณีที่ดีที่สุด ดัชนีบิตแมปแบบเข้ารหัสที่ใช้เทคนิคการจัดกลุ่มข้อมูลมีประสิทธิภาพในแง่ของเวลาที่ใช้ในการสอบถามข้อมูลแบบสมาชิกดีที่สุด นอกจากนี้เมื่อพิจารณาในแง่

Space-Time Trade-off ดัชนีบิตแมปแบบเข้ารหัสที่ใช้เทคนิคการจัดกลุ่มข้อมูลยังมีประสิทธิภาพดีที่สุด จากที่กล่าวมา จะได้ว่า เมื่อต้องการทำดัชนีบนแอตทริบิวต์ที่มีคาร์ดินอลิตี้สูง และต้องการประสิทธิภาพทั้งในแง่ของพื้นที่ เวลา และ Space-Time Trade-off ควรเลือกสร้างดัชนีบิตแมปแบบรหัสที่ใช้เทคนิคการจัดกลุ่มข้อมูล

5.2.ข้อจำกัดของดัชนีบิตแมปแบบเข้ารหัสที่ใช้เทคนิคการจัดกลุ่มข้อมูล

ดัชนีบิตแมปแบบเข้ารหัสที่ใช้เทคนิคการจัดกลุ่มข้อมูลเหมาะกับการทำบนแอตทริบิวต์ที่มีคาร์ดินอลิตี้สูง และการสร้างดัชนีบิตแมปแบบเข้ารหัสที่ใช้เทคนิคการจัดกลุ่มข้อมูลขึ้นกับประวัติการสอบถามข้อมูลในอดีต ดังนั้นหากมีเงื่อนไขการสอบถามข้อมูลแบบสมาชิกที่ไม่เคยสอบถามมาก่อนอาจจะทำให้ประสิทธิภาพการสอบถามข้อมูลลดลง นอกจากนี้ในการจัดกลุ่มค่าของแอตทริบิวต์ที่ถูกสอบถามไปด้วยกันและสอบถามร่วมกันได้นำตาราง Bit Matrix มาใช้หากลุ่มข้อมูล หากแอตทริบิวต์ที่นำมาทำดัชนีมีคาร์ดินอลิตี้สูงจะทำให้ขนาดของตาราง Bit Matrix ใหญ่ขึ้นด้วยส่งผลให้ใช้พื้นที่ในการจัดเก็บตาราง Bit Matrix เพิ่มขึ้น

5.3.ข้อเสนอแนะและงานในอนาคต

- ประสิทธิภาพของดัชนีบิตแมปแบบเข้ารหัสที่ใช้เทคนิคการจัดกลุ่มข้อมูลขึ้นอยู่กับประวัติการสอบถามในอดีต ดังนั้นประวัติการสอบถามในอดีตที่นำมาช่วยในการสร้างดัชนีบิตแมปแบบเข้ารหัสควรสอดคล้องกับเป้าหมายที่ต้องการ
- สำหรับการหากลุ่มค่าของแอตทริบิวต์ที่มีการสอบถามไปด้วยกัน และสอบถามร่วมกัน นอกจากจะใช้เทคนิคการจัดกลุ่มข้อมูลที่นำเสนอและ ยังสามารถนำเทคนิคการทำเหมืองข้อมูลเทคนิคอื่นมาประยุกต์ใช้ได้ เช่น การจำแนกข้อมูล (Classification) การหากฎความสัมพันธ์ของข้อมูล (Association Rules)

อย่างไรก็ตามการหากลุ่มค่าของแอตทริบิวต์ที่มีการสอบถามไปด้วยกันและสอบถามร่วมกัน ผู้ใช้จะต้องกำหนดค่า Minimum Frequency และค่า Threshold ด้วยตนเอง ดังนั้นงานวิจัยในอนาคตจะเป็นการกำหนดค่า Minimum Frequency และค่า Threshold อัตโนมัติ โดยคำนึงถึงการลดจำนวนบิตแมปเวกเตอร์ที่ต้องอ่านและเวลาที่ใช้ในการสอบถามข้อมูลให้มีประสิทธิภาพมากที่สุด

บรรณานุกรม

Alam, G. R., Arafat, M. Y. and Uddin Iftekhar, M. K. 2008. A New Approach of Dynamic Encoded Bitmap Indexing Technique based on Query History. Proceeding of the 5th International Conference on Electrical and Computer Engineering. Dhaka, Bangladesh December 20-22, 2008. pp.974-979.

Bontempo, C. J. and Saracco, C. M. 2011. Accelerating Indexed Searching. http://www.fortunecity.com/skyscraper/oracle/699/orahtml/dbpd/accelind.html (accessed 17/09/2011).

Chan, C. Y. and Ioannidis, Y. E. 1998. Bitmap Index Design and Evaluation. Proceeding of the 1998 ACM SIGMOD International Conference on Management data. pp.355-366.

Chan, C. Y. and Ioannidis, Y. E. 1999. An Efficient Bitmap Encoding Scheme for Seletion Queries. Proceeding of the 1999 ACM SIGMOD International Conference on Management data. pp.215-116.

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

Goebel, M. and Gruenwald, L. 1999. A Survey of Data Mining and Knowledge Discovery Software Tools. ACM SIGKDD Vol. 1. pp.20-33.

Hamadou, A. and Yang, K. 2008. An Efficient Bitmap Indexing Strategy based on Word-Aligned Hybrid for Data Warehouse. Proceeding of the 2008 International Conference on Computer Science and Software Engineering. pp.486-491.

Han, J. and Kamber, M. 2000. Data Warehouse and OLAP Technology for Data Mining. In: Data Mining Concepts and Techniques. Chapter 2.

Han, J. and Kamber, M. 2000. Cluster Analysis. In: Data Mining Concepts and Techniques. Chapter 8.

He, Z., Xu, X. and Deng, S. 2011. Attribute value weighting in k-modes clustering. Expert System with Application, Vol. 38. pp.15365-15369.

Huang, Z. 1998. Extension to the K-Means Algorithm for Clustering Large Data Sets with Categorical Values. Data Mining and Knowledge Discovery, Vol. 2. pp.283-304.

Inmon, W. H. 2005. Building the Data Warehouse, Fourth Edition. Cannad: Wiley Publishing, Inc.

Joshi, R., Patidar, A. and Mishra, S. 2011. Scaling K-mediod Algorithm for Clustering Large Categorical Dataset and its performance analysis. Proceeding of the 3rd International Conference on Electronic Computer Technology. pp.117-121.

Kimball, R. and M. Ross. 2002. The Data Warehouse Toolkit, Second Edition. Canada: John Wiley and Sons, Inc.

Null, L. and Lobur, J. 2003. Boolean Algebra and Digital Logic. In: The Essentials of Computer Organization and Architecture. Chapter 3. London: Jones and Bartlett Publishers, Inc.

O'Neil, P. and Quass, D. 1997. Improved Query Performance with Variant Indexes. Proceeding of the 1997 ACM SIGMOD International Conference on Management data. pp.38-49.

Quine, W. V. 1952. The Problem of Simplifying Truth Functions. The America Mathematical Monthly. Vol. 59. pp.521-531.

Sainui, J., Vanichayobon, S. and Wattanakitrungroj, N. 2008. Optimizing Encoded Bitmap Index using Frequent Itemsets Mining. Proceeding of International Conference on Computer and Electrical Engineering. pp.511-515.

Sheng, W. and Liu, X. 2004. A Hybrid Algorithm for K-mediod Clusterign of Large Data Sets. Evolutionary Computation, Vol. 1. pp.77-82.

Silberschatz, A., Korth, H. F., and Sudarshan, S. 2011. Indexing and Hashing. In: Database System Concepts, Sixth Edition. Chapter 11.

Silberschatz, A., Korth, H. F., and Sudarshan, S. 2011. Data Warehousing and Mining. In: Database System Concepts, Sixth Edition. Chapter 20.

Stockinger, K., Wu, K. and Shoshani, A. 2002. Strategies for Processing ad hoc Queries on Large Data Warehouses. Proceeding of the 5th ACM International Workshop on Computer Science and Software Engineering. Virginia, USA November 8, 2002. pp.72-79.

Tomaszewski, S. P., Celik, I. U. and Antoniou, G. E. 2003. WWW-based Boolean Function Minimization. International Journal of Applied Mathematics and Computer Science. Vol. 13. pp.577-583.

Transaction Processing Performance Council (TPC). 2011. TPC-H: An Ad-hoc, Decision Support Benchmark. Standard Specification Revision 2.14.1. http://www.tpc.org/tpch/ (accessed 11/08/2011).

Vanichayobon, S., Manfuekphan, J. and Gruenwald, L. 2006. Scatter Bitmap: Space-Time Efficient Bitmap Indexing for Equality and Membership Queries. Proceeding of the IEEE International Conferences on cybernetics and Intelligent Systems. pp.1-6.

Wattanakitrungroj, N and Vanichayobon, S. 2006. Dual Bitmap Index: Space-Time Efficient Bitmap Index for Equality and Membership Queries. Proceeding of the International Symposium on Communications and Information Technologies. pp.568-573.

Weahama, W., Vanichayobon, S. and Manfuekphan, J. 2009. Using Data Clustering to Optimization Scatter Bitmap Index for Membership Queries. Proceeding of the International Conferences on Computer and Automation Engineering. pp.174-178.

Wu, K. and Yu, P. 1998. Range-based Bitmap Indexing for High Cardinality Attribute with Skew. Proceeding of the 22nd Annual International Computer Software and Applications Conference. pp.61-66.

Wu, K., Otoo, E. J. and Shoshani, A. 2006. Optimizing Bitmap Indices with Efficient Compression. ACM Transactions on Database System, Vol. 31, No. 1. pp.1-38.

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

Yoon, J. P., Raghavan, V. and Chakilam, V. 2001. Bitmap Indexing-based Clustering and Retrieval of XML Documents. Proceeding of the ACM SIGIR'01 Workshop on Mathematical/Formal Methods in IR.

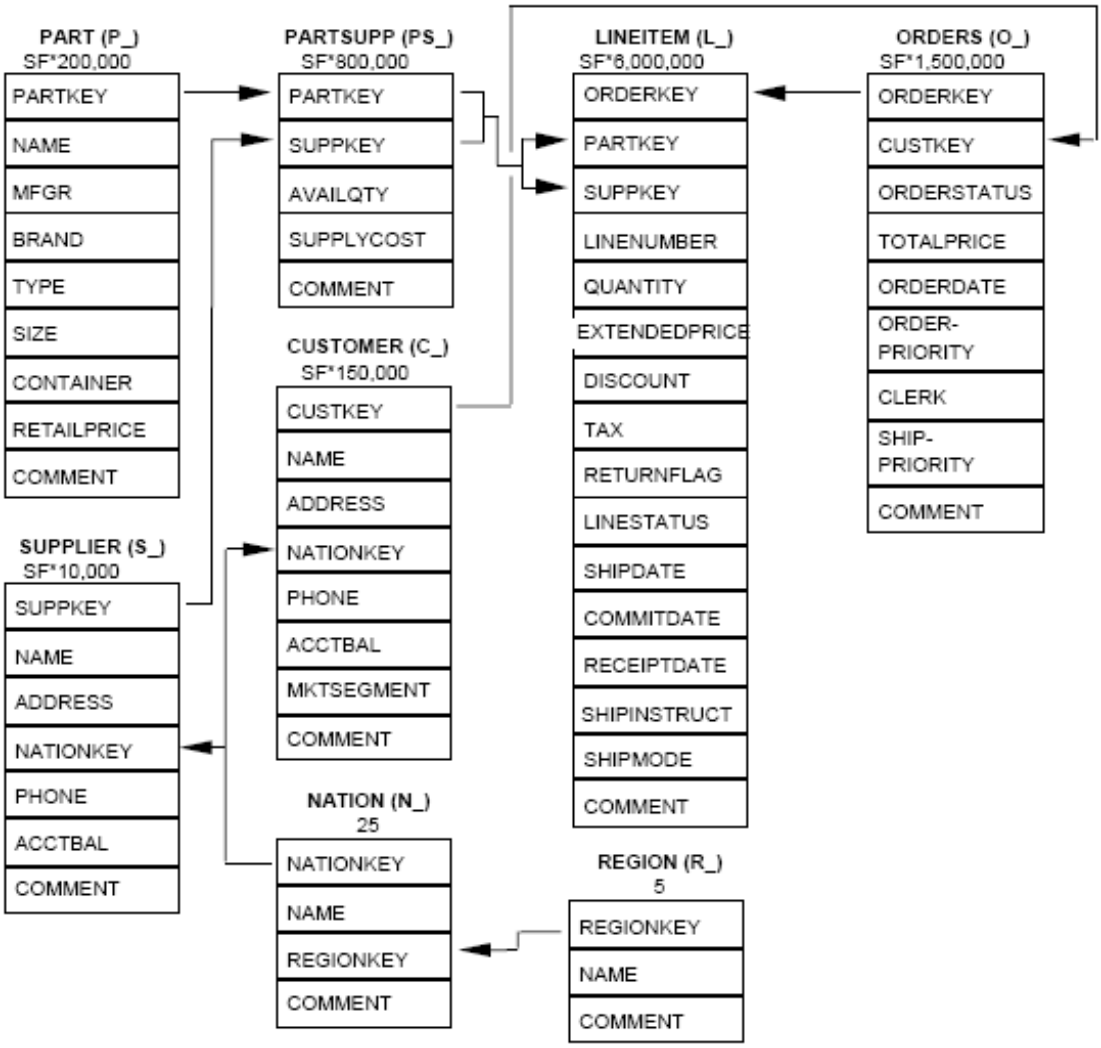
Zaman, M., Surabattula, J. and Gruenwald, L. 2004. An Auto-Indexing Technique for Database Based on Clustering. Proceeding of the 15th International Workshop on Database and Expert Systems Applications. pp.776-780.

ภาคผนวก

ภาคผนวก ก

ก.1 การวัดเปรียบเทียบสมรรถนะด้วย TPC-H Benchmark

ข้อมูลที่ใช้ในวิทยานิพนธ์นี้ เป็นข้อมูลมาตรฐานจาก TPC-H Benchmark (Transaction Processing Performance Council, 2011) ซึ่งเป็นเครื่องมือในการวัดเปรียบการประมวลผลการสอบถามที่ซับซ้อนที่ใช้ในระบบสนับสนุนการตัดสินใจ (Decision Support) โดยต้องการทดสอบกับข้อมูลที่มีปริมาณมาก โดยมีโครงสร้างดังนี้



ภาพประกอบ ก-1 โครงสร้างข้อมูล TPC-H

ก.2 ข้อมูลตารางและแอตทริบิวต์ที่ใช้ในการทดลอง

ข้อมูลที่ใช้ในการทดลองมี 3 ชุด และมาจาก 2 ตาราง นั่นคือ ข้อมูลจากตาราง PART และ ORDER ซึ่งมีรายละเอียดดังนี้

ข้อมูลชุดที่ 1 แอตทริบิวต์ SIZE บนตาราง PART
มี 5,000,000 แถว คาร์ดินอลิตี้เท่ากับ 50 (C = 50)

ข้อมูลชุดที่ 2 แอตทริบิวต์ TYPE บนตาราง PART
มี 5,000,000 แถว คาร์ดินอลิตี้เท่ากับ 150 (C = 150)

ข้อมูลชุดที่ 1 แอตทริบิวต์ CLERK บนตาราง ORDER
มี 5,000,000 แถว คาร์ดินอลิตี้เท่ากับ 1000 (C = 1000)

ก.3 โครงสร้างตาราง (Table Layouts) ของการวัดเปรียบเทียบสมรรถนะ

ฐานข้อมูลของการวัดเปรียบเทียบสมรรถนะของ TPC-H ประกอบด้วยตารางหลายตาราง เช่น ตาราง CUSTOMER เป็นตารางที่เก็บข้อมูลเกี่ยวกับลูกค้า ตาราง PART เป็นตารางที่เก็บข้อมูลเกี่ยวกับชิ้นส่วนของสินค้า เป็นต้น โดยมีโครงสร้างตารางดังนี้

โครงสร้างตาราง PART

ชื่อแอตทริบิวต์	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

โครงสร้างตาราง ORDER

ชื่อแอตทริบิวต์	ชนิดข้อมูล	หมายเหตุ
O_ORDERKEY	Identifier	SF*1,500,000 are sparsely populated
O_CUSTKEY	Identifier	Foreign Key to C_CUSTKEY
O_ORDERSTATUS	Fixed text, size 1	
O_TOTALPRICE	Decimal	
O_ORDERDATE	Date	
O_ORDERPRIORITY	Fixed text, size 15	
O_CLERK	Fixed text, size 15	
O_SHIPPRIORITY	Integer	
O_COMMENT	Variable text, size 79	

Primary Key: O_ORDERKEY

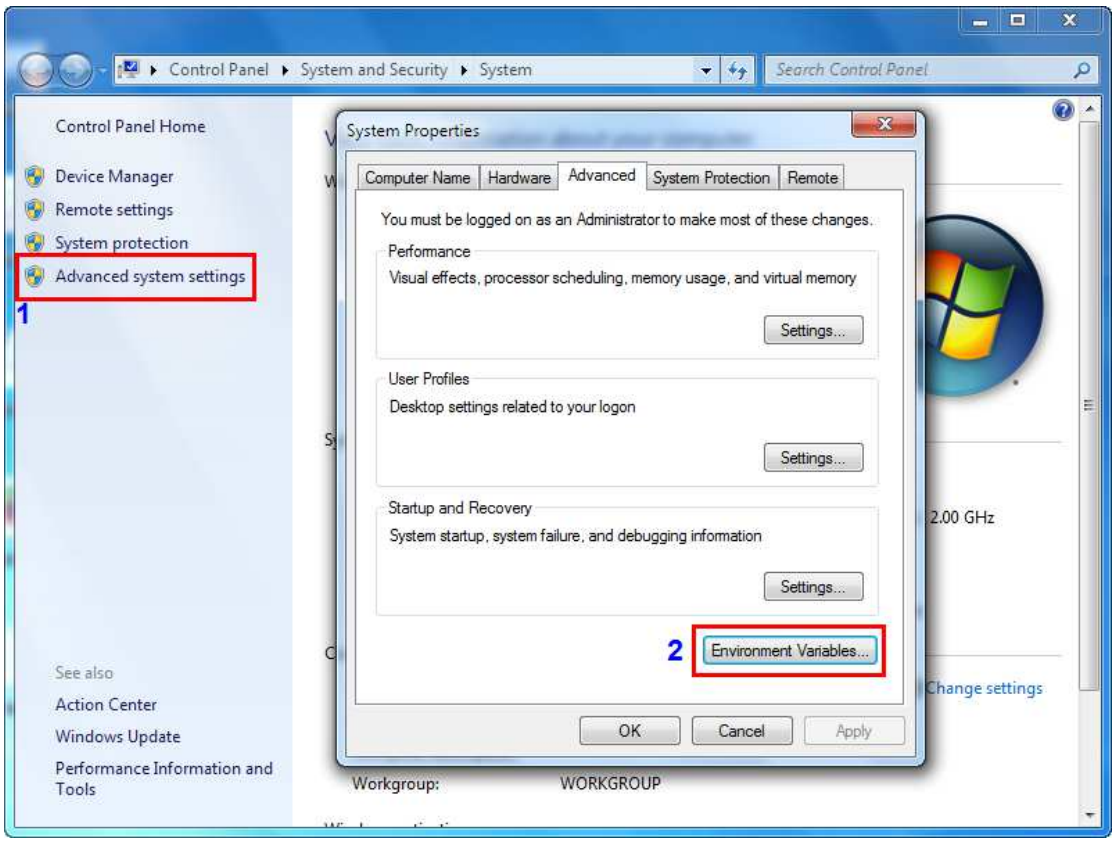
ก.4 การเตรียมข้อมูลเพื่อใช้ในการทดลอง

ในส่วนนี้เป็นการเตรียมข้อมูล เพื่อวัดเปรียบเทียบประสิทธิภาพด้านเวลาที่ใช้ในการสอบถามข้อมูลแบบสมาชิก และการสอบถามข้อมูลแบบค่าเท่ากันของดัชนีบิตแมปทั้ง 7 แบบ (Simple Bitmap Index, Range Bitmap Index, Interval Bitmap Index, Scatter Bitmap Index, Dual Bitmap Index, Encoded Bitmap Index, Cluster-EBI)

สำหรับแต่ละแอตทริบิวต์ที่เลือกมาทำดัชนีบิตแมปแต่ละแบบ จะต้องมีการเตรียมข้อมูลตามขั้นตอนต่อไปนี้

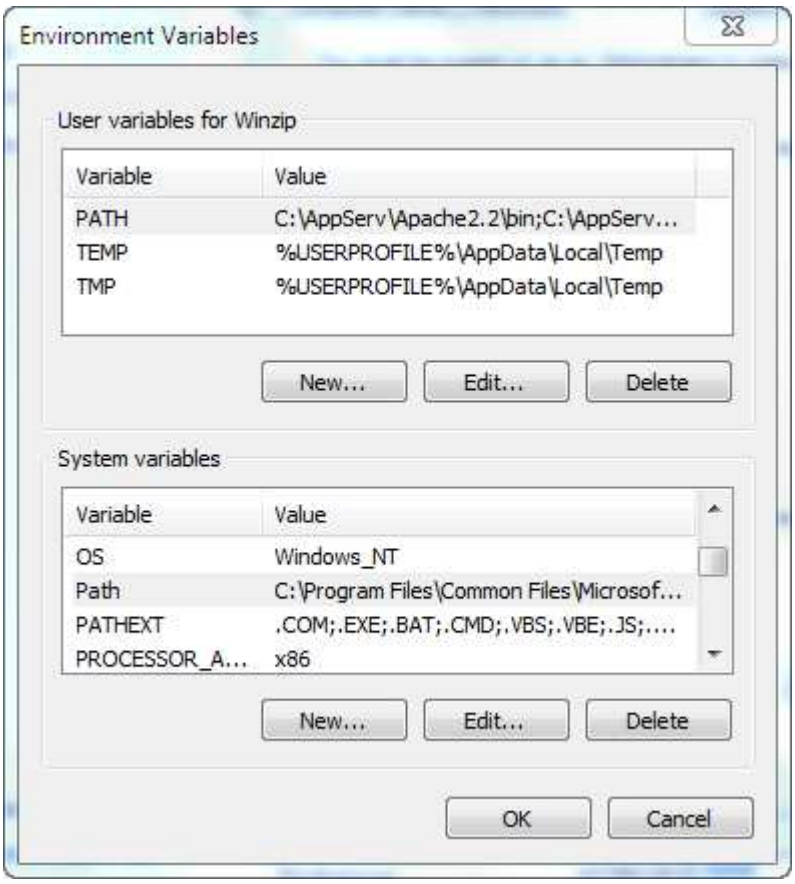
1. ติดตั้งโปรแกรม gawk-3.1.6-1-setup.exe ซึ่งเป็นโปรแกรมเสริมที่ช่วยในการเลือกแอตทริบิวต์ที่ต้องการนำมาทำดัชนีบิตแมป ซึ่งสามารถใช้บนโปรแกรม MS-DOS ในระบบปฏิบัติการ Window 7 ได้

2. ตั้งค่า Path Environment Variable เพื่อให้สามารถใช้งานโปรแกรม gawk ที่ได้ติดตั้งก่อนหน้านี้ โดยเข้าไปตั้งค่าที่ Control > System and Security > System > Advanced System Setting แสดงดังภาพประกอบ ก-2

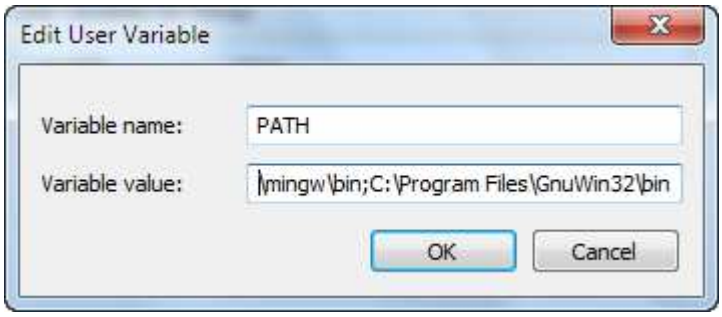


ภาพประกอบ ก-2 หน้าต่าง System Properties

3. เลือก Environment Variables… ดังแสดงในภาพประกอบ ก-2 จะได้หน้าต่าง Environment Variables แสดงดังภาพประกอบ ก-3 จากนั้นในส่วน User Variables ให้เลือก Edit เพื่อแก้ไขค่าของ Variable ที่ชื่อว่า PATH โดยการเพิ่ม Path ที่ได้ลงโปรแกรม gawk ไว้ ตัวอย่างเช่น ถ้าเราลงโปรแกรม gawk ไว้ที่ C:\Program Files\GnuWin32\bin เราจะเพิ่มรายการ “C:\Program Files\GnuWin32\bin” เข้าไปใน PATH แสดงดังภาพประกอบ ก-4



ภาพประกอบ ก-3 หน้าต่าง Environment Variables



ภาพประกอบ ก-4 หน้าต่าง Edit User Variable

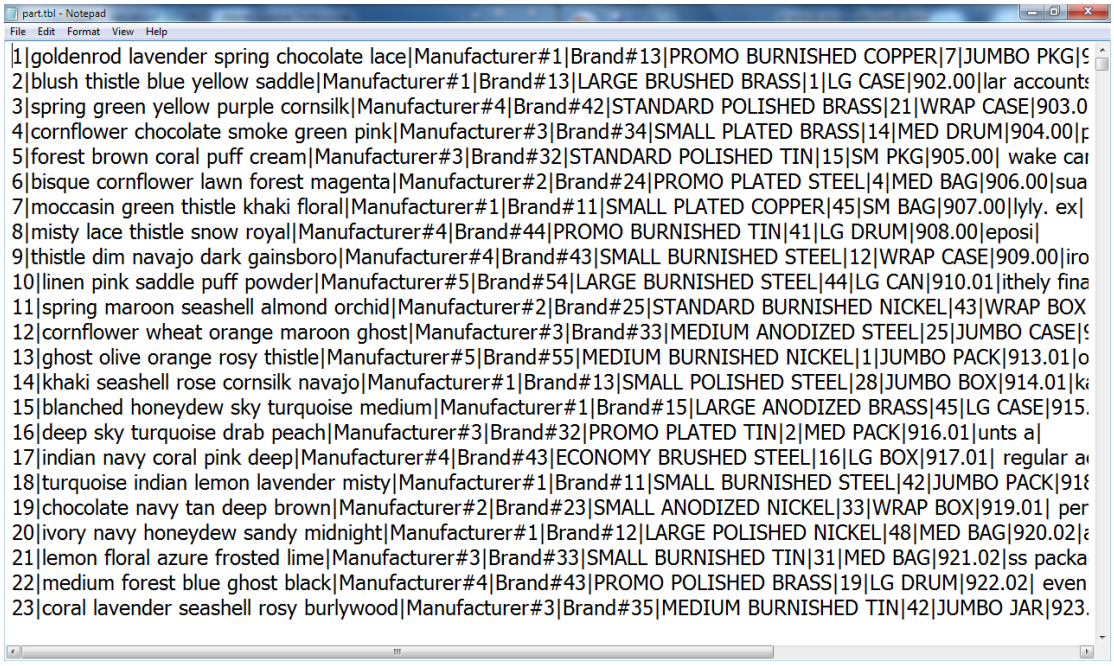
4. จากนั้นเลือกเฉพาะแอตทริบิวต์ที่จะนำมาทำดัชนีโดยใช้พิมพ์คำสั่ง

gawk –F “|” “{print $number}” input_filename > output_filename

ลงในโปรแกรม MS-DOS โดยที่ | คืออักขระที่ใช้คั่นแต่ละค่าของแอตทริบิวต์ number คือ ลำดับของแอตทริบิวต์ที่ต้องการ input_filename คือ ชื่อไฟล์ตารางที่มีแอตทริบิวต์ที่ต้องการ และ output_filename คือ ชื่อไฟล์ที่ใช้เก็บข้อมูลที่เลือก

ตัวอย่างเช่น gawk –F “|” “{print $5}” part.tbl > part_type.txt เป็นการเลือกแอตทริบิวต์ลำดับที่ 5 จากตาราง PART และเก็บผลลัพธ์ไว้ในไฟล์ชื่อ part_type.txt ตัวอย่าง

ไฟล์ part.tbl แสดงดังภาพประกอบ ก-5 และตัวอย่างไฟล์ part_type.txt แสดงดังภาพประกอบ ก-6(ซ้าย)



ภาพประกอบ ก-5 ตัวอย่างข้อมูลในไฟล์ตาราง PART (part.tbl)

5. เปลี่ยนรูปค่าของแอตทริบิวต์ให้เป็นจำนวนเต็มที่ต่อเนื่องกัน เริ่มตั้งแต่ค่า 0 ดังนั้น ค่าแอตทริบิวต์ที่ถูกเปลี่ยนค่าแล้วคือ 0, 1, 2, …, C-1 โดยการสร้าง attr.gawk ซึ่งเขียนเป็นชุดคำสั่งสำหรับเปลี่ยนค่าในไฟล์แล้วเก็บไว้ในอีกไฟล์หนึ่ง

ตัวอย่างชุดคำสั่งในไฟล์ attr.gawk

/ECONOMY ANODIZED BRASS/ {print 0}

(เปลี่ยนคำว่า ECONOMY ANODIZED BRASS เป็น 0)

/ECONOMY ANODIZED COPPER/ {print 1}

(เปลี่ยนคำว่า ECONOMY ANODIZED BRASS เป็น 1)

/ECONOMY ANODIZED NICKEL/ {print 2}

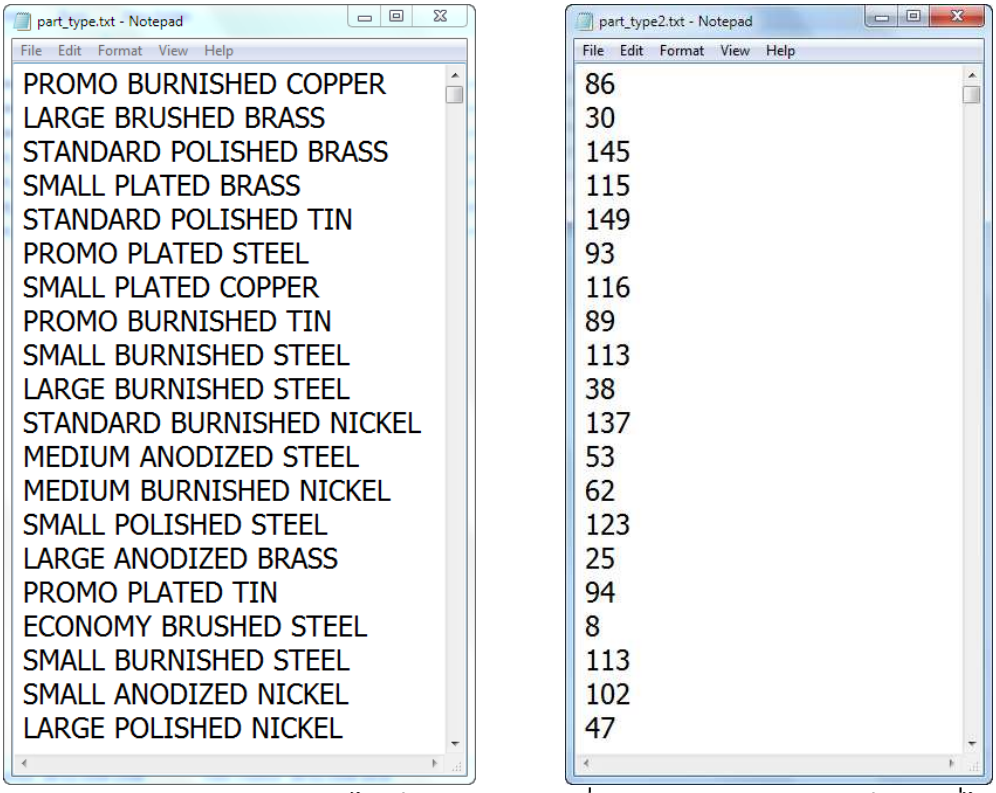
(เปลี่ยนคำว่า ECONOMY ANODIZED BRASS เป็น 2)

จากนั้นใช้โปรแกรม MS-DOS เพื่อสั่งให้คำสั่งใน attr.gawk ทำงานโดยพิมพ์คำสั่ง

gawk –f attr.gawk input_filename > output_filename

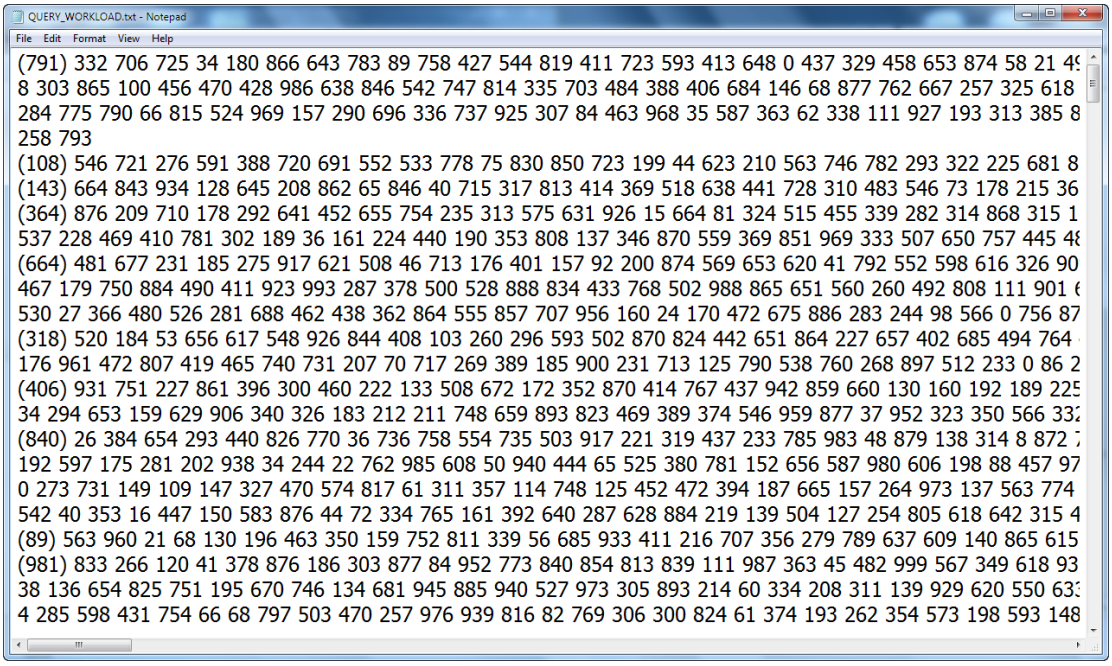
ลงในโปรแกรม MS-DOS โดยที่ attr.gawk คือไฟล์ชุดคำสั่ง input_filename คือไฟล์ที่ใช้เก็บข้อมูลของแอตทริบิวต์ที่เลือก และ output_filename คือ ไฟล์ที่ใช้เก็บข้อมูลของ

แอตทริบิวต์ที่เลือก และมีการเปลี่ยนรูปแบบเป็นจำนวนเต็มที่ต่อเนื่องกัน ตัวอย่างไฟล์ output_filename แสดงดังภาพประกอบ ก-6(ขวา)



ภาพประกอบ ก-6 (ซ้าย) ตัวอย่างไฟล์ part_type.txt ซึ่งเก็บค่าของแอตทริบิวต์ type ที่ได้จากไฟล์ part.tbl (ขวา) ไฟล์ที่ได้จากการเปลี่ยนรูปแบบค่าของแอตทริบิวต์ type เป็นจำนวนเต็ม

สำหรับประวัติการสอบถามข้อมูลแบบสมาชิกในอดีตที่นำมาใช้ในการทดลอง โดยเลือกแอตทริบิวต์ที่มีคาร์ดินอลิตี้เท่ากับ 1000 และทำการสุ่มประวัติการสอบถามข้อมูลแบบสมาชิกจำนวน 100 ทรานแซคชัน แต่ละทรานแซคชันจะทำการสุ่มจำนวนสมาชิก (จำนวนค่าของแอตทริบิวต์ในแต่ละทรานแซคชัน) และค่าของแอตทริบิวต์แต่ละค่า (เป็นจำนวนเต็ม) ที่ถูกสอบถามไปด้วยกัน และสอบถามร่วมกัน ตัวอย่างประวัติการสอบถาม แสดงดังภาพประกอบ ก-7 โดยตัวเลขที่อยู่ในเครื่องหมาย “()” ในแต่ละทรานแซคชัน หมายถึง จำนวนสมาชิกที่ถูกสอบถาม และตัวเลขที่เหลือเป็นค่าของแอตทริบิวต์ที่ถูกสอบถาม



ภาพประกอบ ก-7 ตัวอย่างไฟล์ประวัติการสอบถามในอดีต (Query Workload)

ก.5 การเขียนโปรแกรมเพื่อทดลองการสอบถามข้อมูลแบบสมาชิกและการสอบถามข้อมูลแบบค่าเท่ากันบนดัชนีบิตแมป

ในการเขียนโปแกรมเพื่อทดลองสำหรับการสอบถามข้อมูลแบบสมาชิกและการสอบถามข้อมูลแบบค่าเท่ากัน เป็นการเขียนโปรแกรมด้วยตัวแปลภาษาซี (C Compiler) โดยมีการกำหนดค่านิยาม ตัวแปร ฟังก์ชัน และไฟล์สำคัญต่อไปนี้

- **ค่าคงที่ (Define)**

CARDINALITY 1000	เป็นการกำหนดค่าให้กับคาร์ดินอลิตี้ ในที่นี้เท่ากับ 1000
NUM_RECORD 5000000	เป็นการกำหนดค่าจำนวนแถวข้อมูล ในที่นี้เท่ากับ 5,000,000
NUM_QUERY 100	เป็นการกำหนดจำนวนการสอบถาม ในที่นี้เท่ากับ 100

- **ตัวแปร (Variable)**

number	ใช้สำหรับเก็บจำนวนสมาชิกในแต่ละการสอบถาม ($1 < number < C$)
queryValue	ใช้สำหรับเก็บค่าที่ต้องการสอบถาม ($0 \leq value < C$)
vector	ใช้สำหรับเก็บค่าลำดับของบิตแมปเวกเตอร์ที่ต้องการอ่านมาเพื่อดำเนินการตรรกะ
bitVector	ใช้สำหรับเก็บค่าบิตแมปเวกเตอร์ที่อ่านจากดัชนี
ansVector	ใช้สำหรับเก็บค่าบิตแมปเวกเตอร์ผลลัพธ์จากการดำเนินตรรกะ ก่อนนำไปเขียนลงไฟล์คำตอบของการสอบถาม

- **ฟังก์ชัน (Function)**

MemberQuerySimple(number,queryValue)	ฟังก์ชันดำเนินการสอบถามข้อมูลแบบสมาชิกบนดัชนีบิตแมปแบบพื้นฐาน
MemberQueryRange(number,queryValue)	ฟังก์ชันดำเนินการสอบถามข้อมูลแบบสมาชิกบนดัชนีบิตแมปแบบแถว
MemberQueryInterval(number,queryValue)	ฟังก์ชันดำเนินการสอบถามข้อมูลแบบสมาชิกบนดัชนีบิตแมปแบบช่วง
MemberQueryScatter(number,queryValue)	ฟังก์ชันดำเนินการสอบถามข้อมูลแบบสมาชิกบนดัชนีบิตแมปแบบกระจาย
MemberQueryDual(number,queryValue)	ฟังก์ชันดำเนินการสอบถามข้อมูลแบบสมาชิกบนดัชนีบิตแมปแบบคู่กัน
MemberQueryEncoded(number,queryValue)	ฟังก์ชันดำเนินการสอบถามข้อมูลแบบสมาชิกบนดัชนีบิตแมปแบบเข้ารหัสทั่วไป
MemberQueryCluster-EBI(number,queryValue)	ฟังก์ชันดำเนินการสอบถามข้อมูลแบบสมาชิกบนดัชนีบิตแมปแบบเข้ารหัสที่ใช้เทคนิคการจัดกลุ่มข้อมูล
EqualityQuerySimple(queryValue)	ฟังก์ชันดำเนินการสอบถามข้อมูลแบบค่าเท่ากันบนดัชนีบิตแมปแบบพื้นฐาน
EqualityQueryRange(queryValue)	ฟังก์ชันดำเนินการสอบถามข้อมูลแบบค่าเท่ากันบนดัชนีบิตแมปแบบแถว
EqualityQueryInterval(queryValue)	ฟังก์ชันดำเนินการสอบถามข้อมูลแบบค่าเท่ากันบนดัชนีบิตแมปแบบช่วง
EqualityQueryScatter(queryValue)	ฟังก์ชันดำเนินการสอบถามข้อมูลแบบค่าเท่ากันบนดัชนีบิตแมปแบบกระจาย

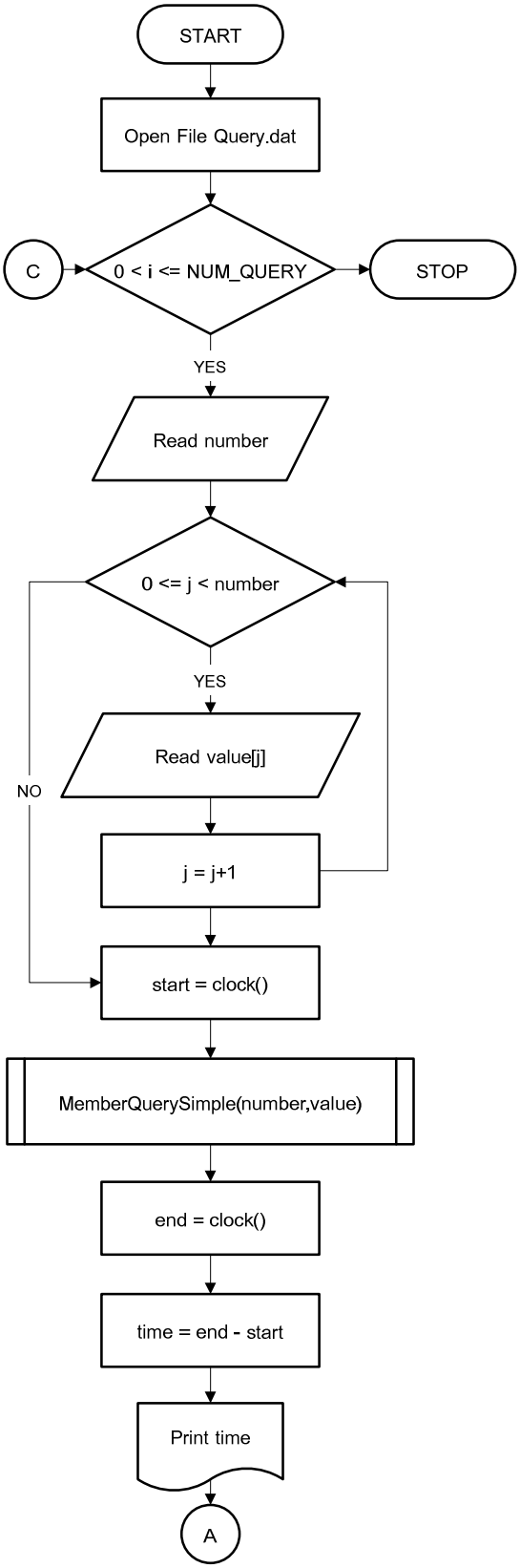
EqualityQueryDual(queryValue)	ฟังก์ชันดำเนินการสอบถามข้อมูลแบบค่าเท่ากันบนดัชนีบิตแมปแบบคู่กัน
EqualityQueryEncoded(queryValue)	ฟังก์ชันดำเนินการสอบถามข้อมูลแบบค่าเท่ากันบนดัชนีบิตแมปแบบเข้ารหัสทั่วไป
EqualityQueryCluster-EBI(queryValue)	ฟังก์ชันดำเนินการสอบถามข้อมูลแบบค่าเท่ากันบนดัชนีบิตแมปแบบเข้ารหัสที่ใช้เทคนิคการจัดกลุ่มข้อมูล
WordPerBitmap(NUM_RECORD)	ฟังก์ชันดำเนินการหาจำนวนเวิร์ด (Word) ของหนึ่งบิตแมปเวกเตอร์ ซึ่งในที่นี้ให้ 1 เวิร์ดเท่ากับ 32 (เท่ากับขนาดของ (Integer) ในตัวแปลภาษาซี

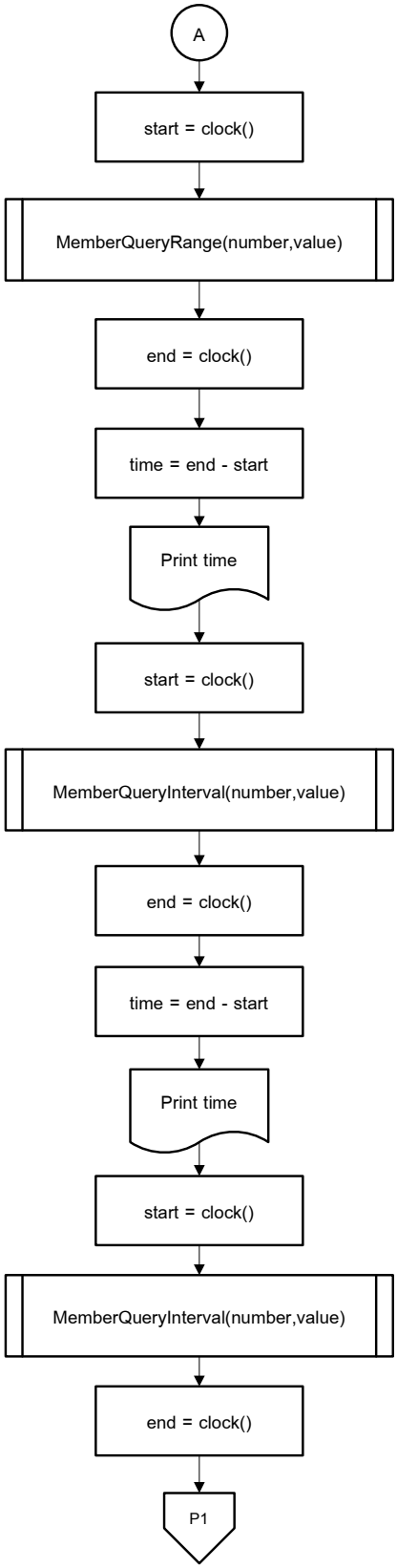
- **ไฟล์ (File)**

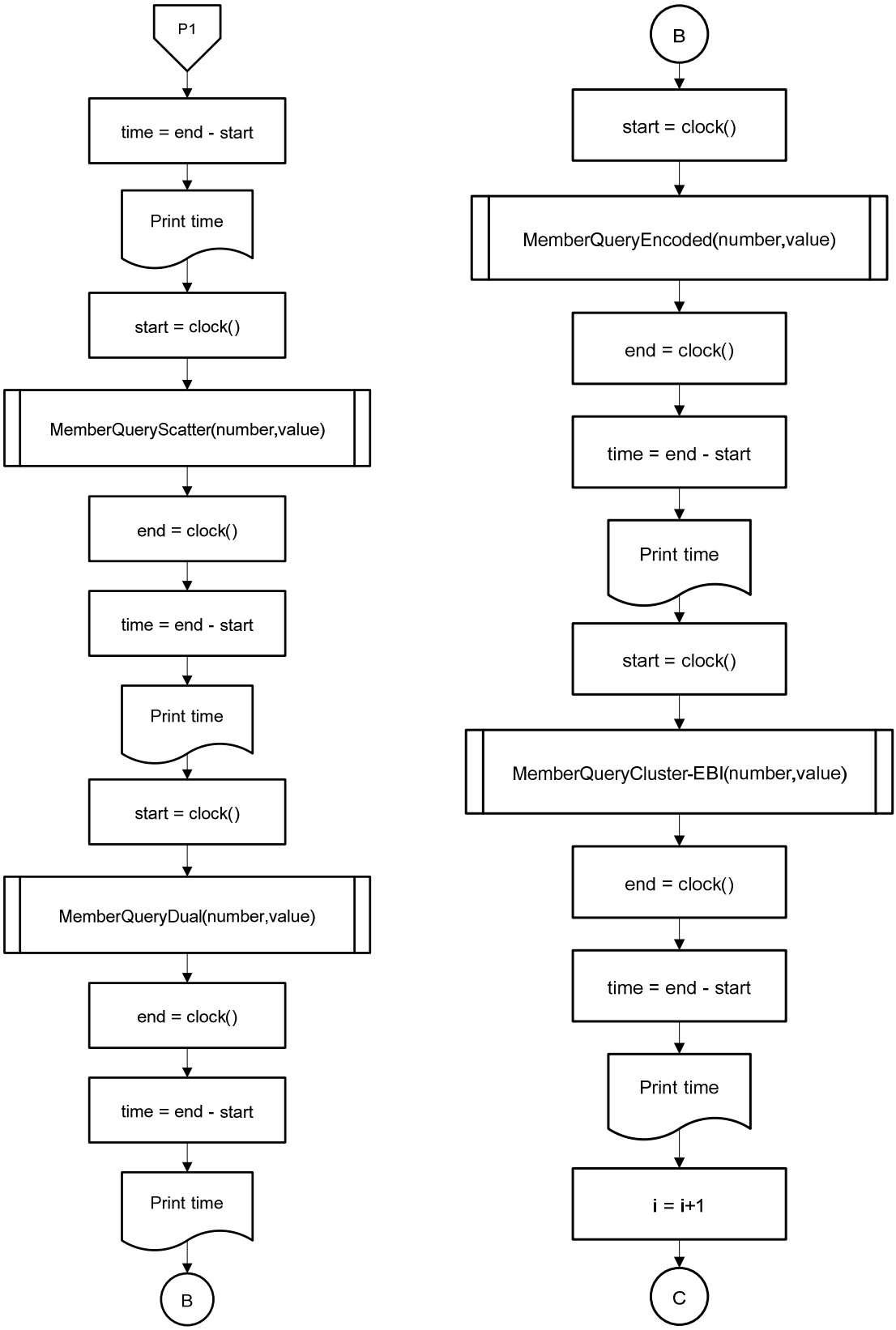
simple.dat	เป็นไฟล์ที่เก็บดัชนีบิตแมปแบบพื้นฐาน
range.dat	เป็นไฟล์ที่เก็บดัชนีบิตแมปแบบแถว
interval.dat	เป็นไฟล์ที่เก็บดัชนีบิตแมปแบบช่วง
scatter.dat	เป็นไฟล์ที่เก็บดัชนีบิตแมปแบบกระจาย
dual.dat	เป็นไฟล์ที่เก็บดัชนีบิตแมปแบบคู่กัน
encoded.dat	เป็นไฟล์ที่เก็บดัชนีบิตแมปแบบเข้ารหัสทั่วไป
cluster-EBI.dat	เป็นไฟล์ที่เก็บดัชนีบิตแมปแบบเข้ารหัสที่ใช้เทคนิคการจัดกลุ่มข้อมูล
ansSimpleMember.dat	เป็นไฟล์ที่เก็บคำตอบของการสอบถามแบบสมาชิกบนดัชนีบิตแมปแบบพื้นฐาน
ansRangeMember.dat	เป็นไฟล์ที่เก็บคำตอบของการสอบถามแบบสมาชิกบนดัชนีบิตแมปแบบแถว
ansIntervalMember.dat	เป็นไฟล์ที่เก็บคำตอบของการสอบถามแบบสมาชิกบนดัชนีบิตแมปแบบช่วง
ansScatterMember.dat	เป็นไฟล์ที่เก็บคำตอบของการสอบถามแบบสมาชิกบนดัชนีบิตแมปแบบกระจาย

ansDualMember.dat	เป็นไฟล์ที่เก็บคำตอบของการสอบถามแบบสมาชิกบนดัชนีบิตแมปแบบคู่กัน
ansEncodedMember.dat	เป็นไฟล์ที่เก็บคำตอบของการสอบถามแบบสมาชิกบนดัชนีบิตแมปแบบเข้ารหัสทั่วไป
ansCluster-EBIMember.dat	เป็นไฟล์ที่เก็บคำตอบของการสอบถามแบบสมาชิกบนดัชนีบิตแมปแบบเข้ารหัสที่ใช้เทคนิคการจัดกลุ่มข้อมูล
ansSimpleEquality.dat	เป็นไฟล์ที่เก็บคำตอบของการสอบถามแบบค่าเท่ากันบนดัชนีบิตแมปแบบพื้นฐาน
ansRangeEquality.dat	เป็นไฟล์ที่เก็บคำตอบของการสอบถามแบบค่าเท่ากันบนดัชนีบิตแมปแบบแถว
ansIntervalEquality.dat	เป็นไฟล์ที่เก็บคำตอบของการสอบถามแบบค่าเท่ากันบนดัชนีบิตแมปแบบช่วง
ansScatterEquality.dat	เป็นไฟล์ที่เก็บคำตอบของการสอบถามแบบค่าเท่ากันบนดัชนีบิตแมปแบบกระจาย
ansDualEquality.dat	เป็นไฟล์ที่เก็บคำตอบของการสอบถามแบบค่าเท่ากันบนดัชนีบิตแมปแบบคู่กัน
ansEncodedEquality.dat	เป็นไฟล์ที่เก็บคำตอบของการสอบถามแบบค่าเท่ากันบนดัชนีบิตแมปแบบเข้ารหัสทั่วไป
ansCluster-EBIEquality.dat	เป็นไฟล์ที่เก็บคำตอบของการสอบถามแบบค่าเท่ากันบนดัชนีบิตแมปแบบเข้ารหัสที่ใช้เทคนิคการจัดกลุ่มข้อมูล

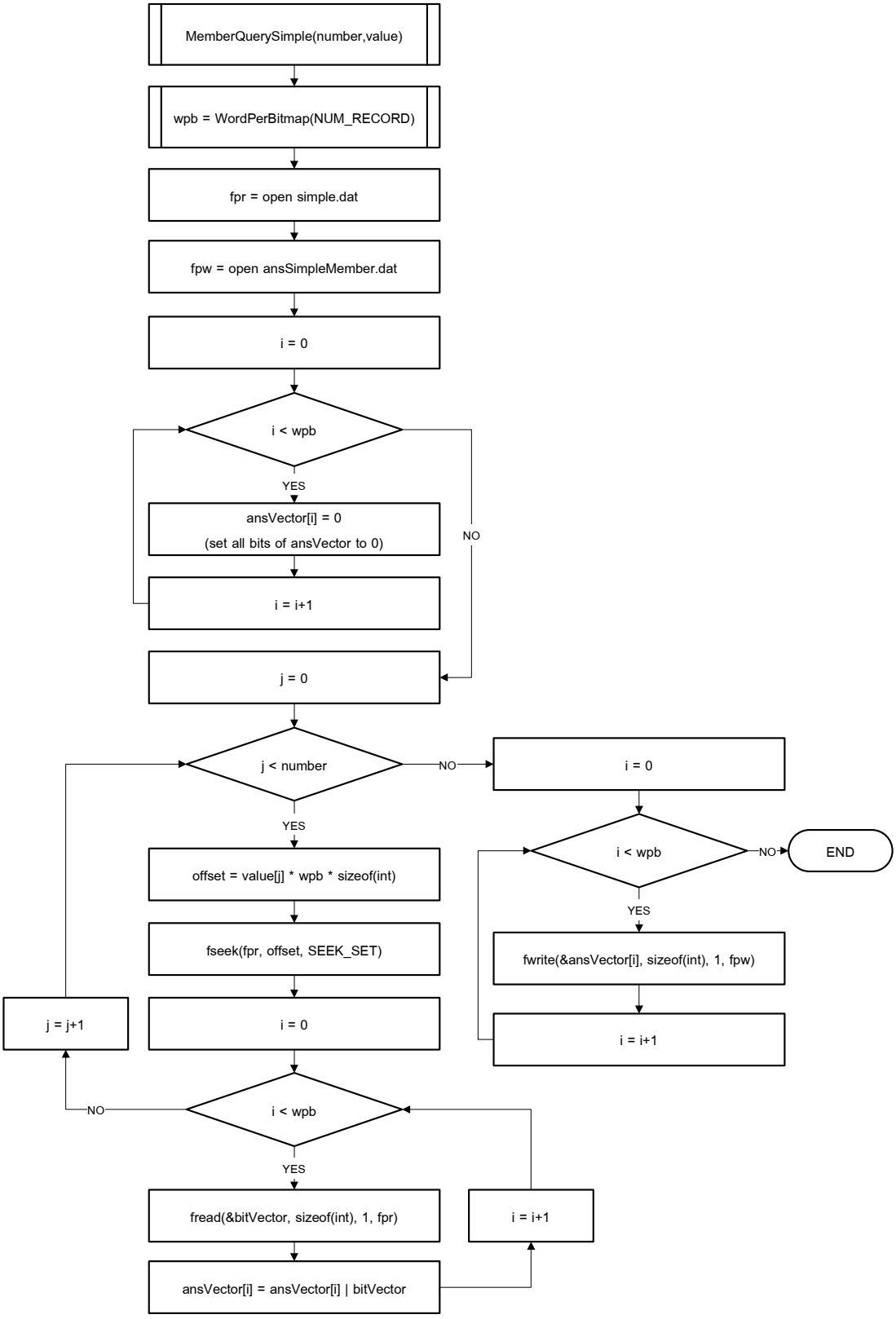
ก.5.1 ขั้นตอนวิธีการทำงานของการสอบถามข้อมูลแบบสมาชิกบนดัชนีบิตแมปทั้ง 7 แบบ



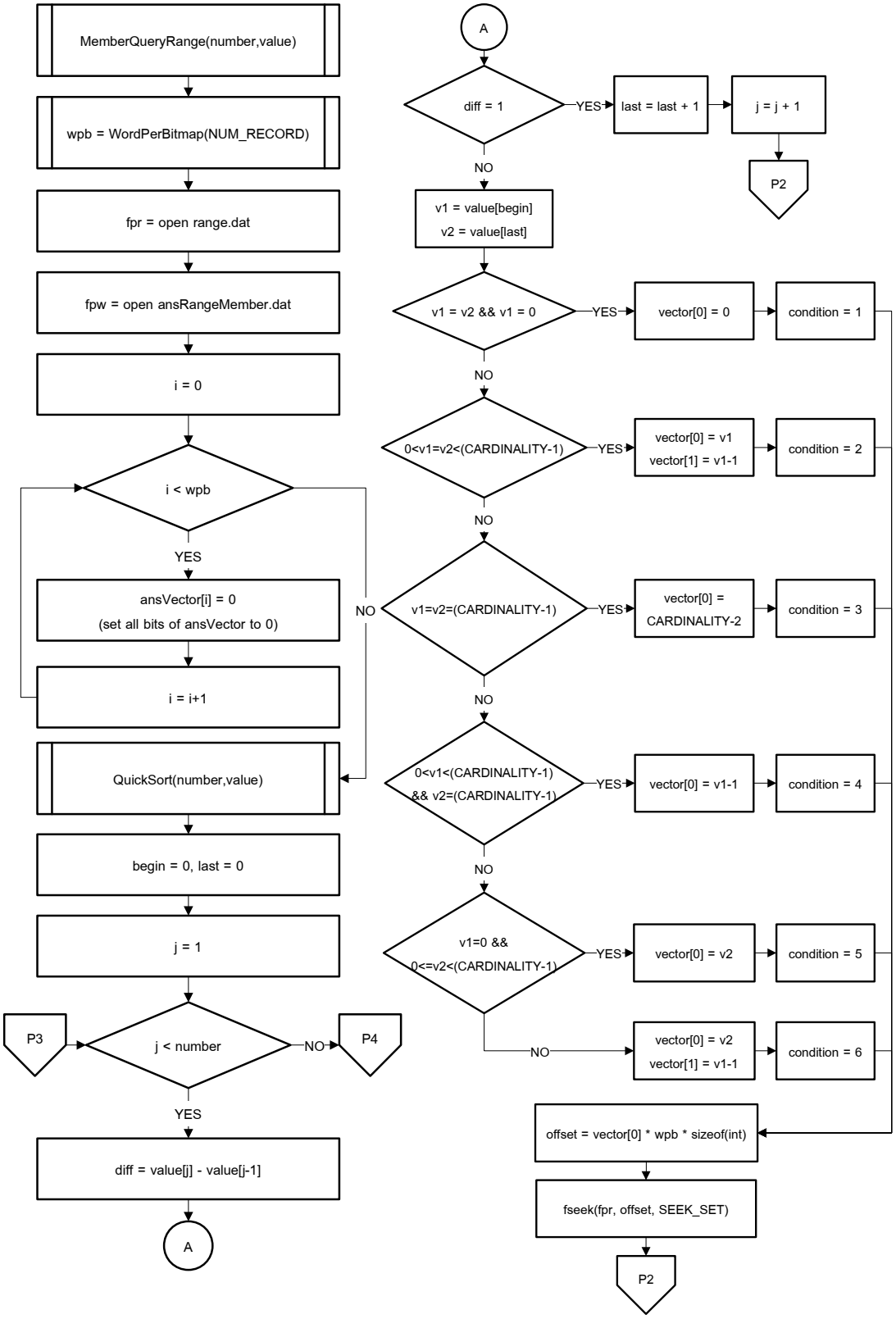


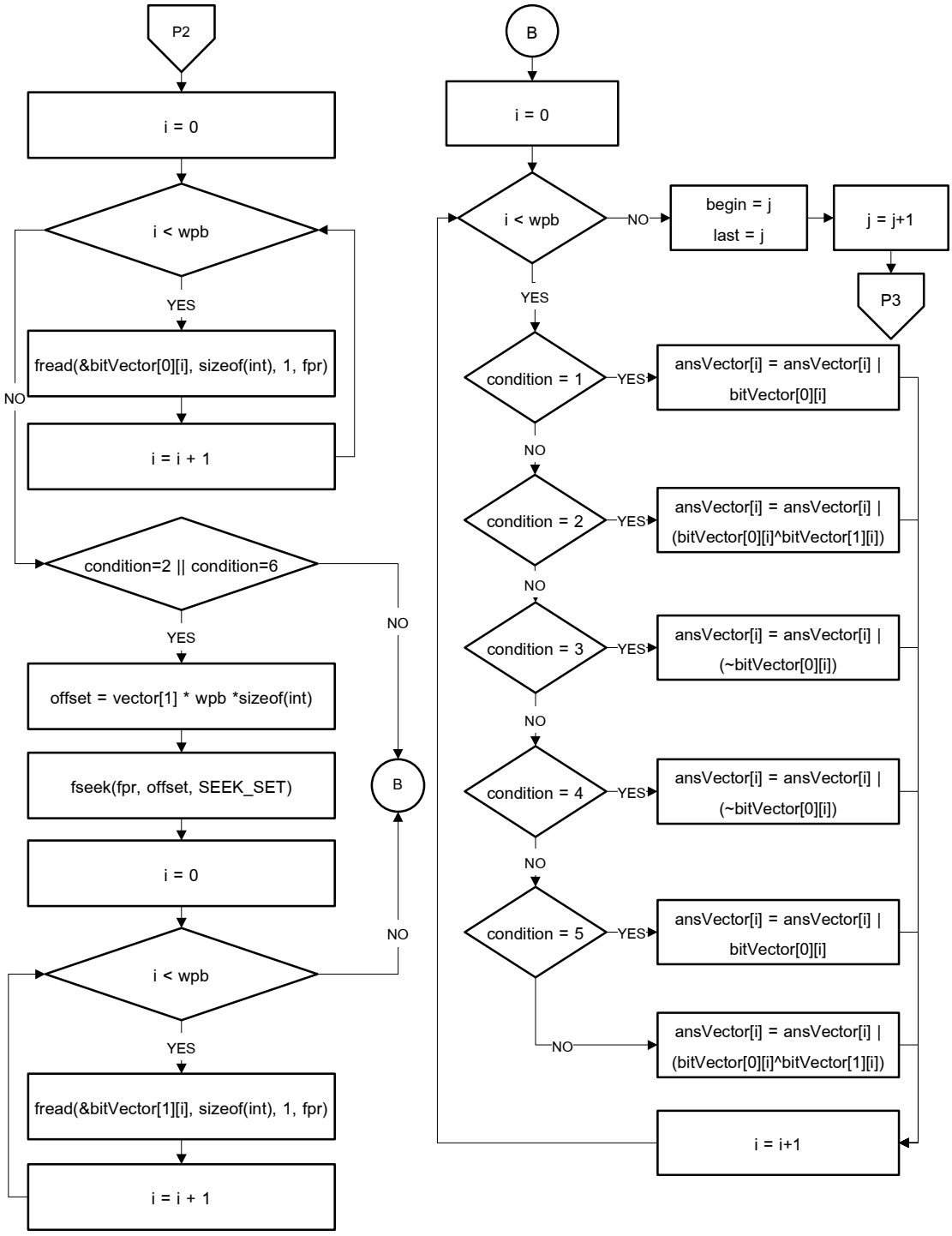

P1
time = end - start
Print time
start = clock()
MemberQueryScatter(number,value)
end = clock()
time = end - start
Print time
start = clock()
MemberQueryDual(number,value)
end = clock()
time = end - start
Print time
B
B
start = clock()
MemberQueryEncoded(number,value)
end = clock()
time = end - start
Print time
start = clock()
MemberQueryCluster-EBI(number,value)
end = clock()
time = end - start
Print time
i = i+1
C

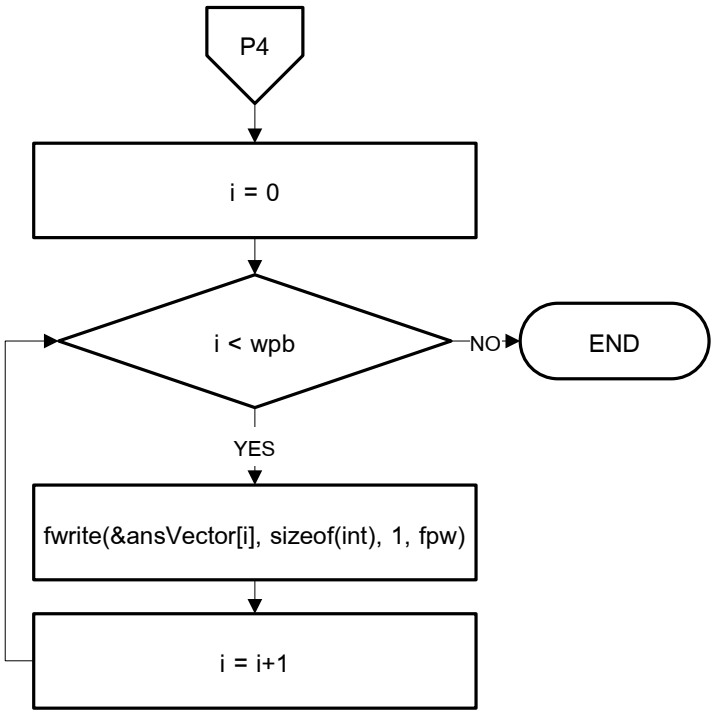
ก.5.2 ขั้นตอนวิธีการสอบถามข้อมูลแบบสมาชิกบนดัชนีบิตแมปแบบพื้นฐาน



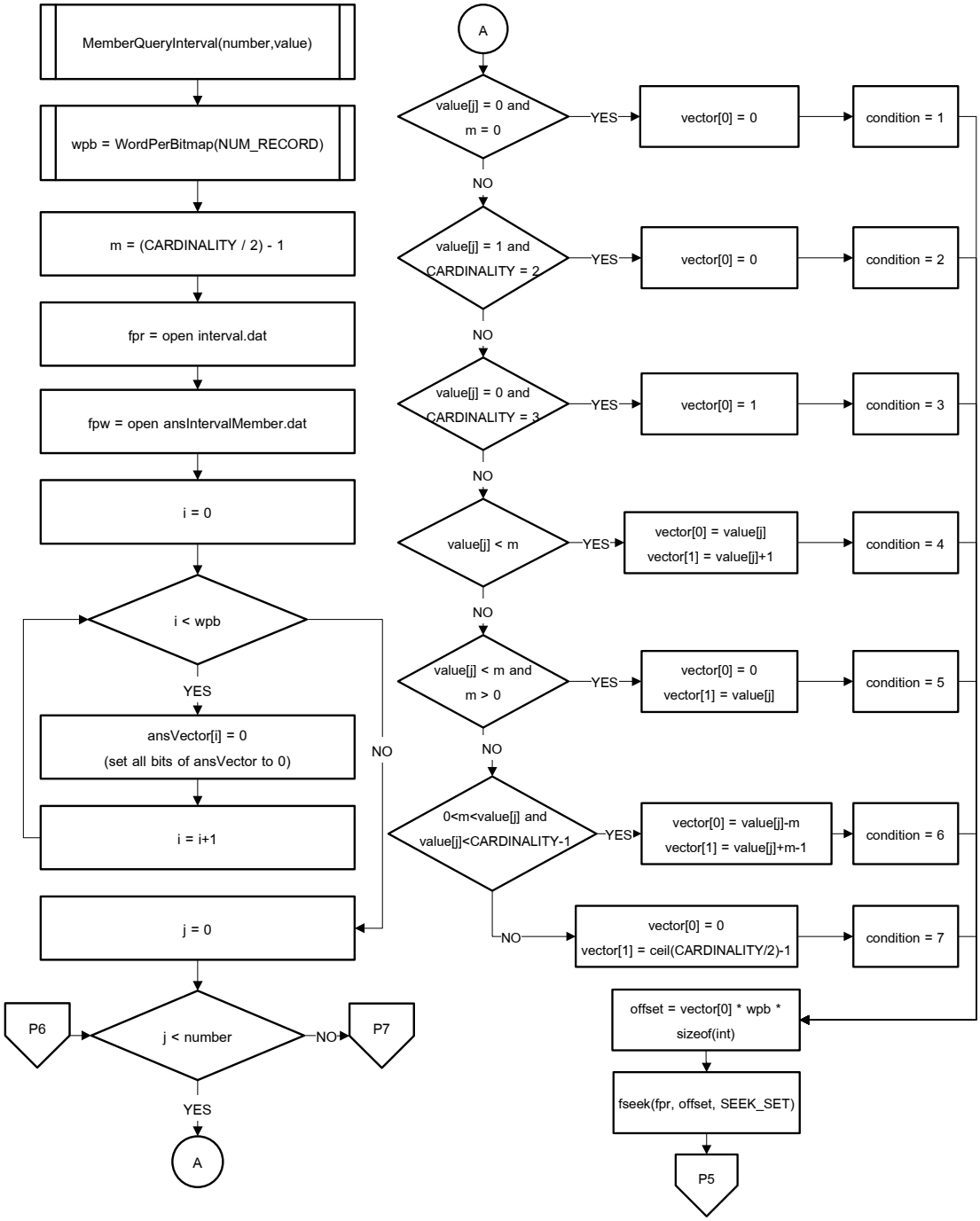
ก.5.3 ขั้นตอนวิธีการสอบถามข้อมูลแบบสมาชิกบนดัชนีบิตแมปแบบแถว

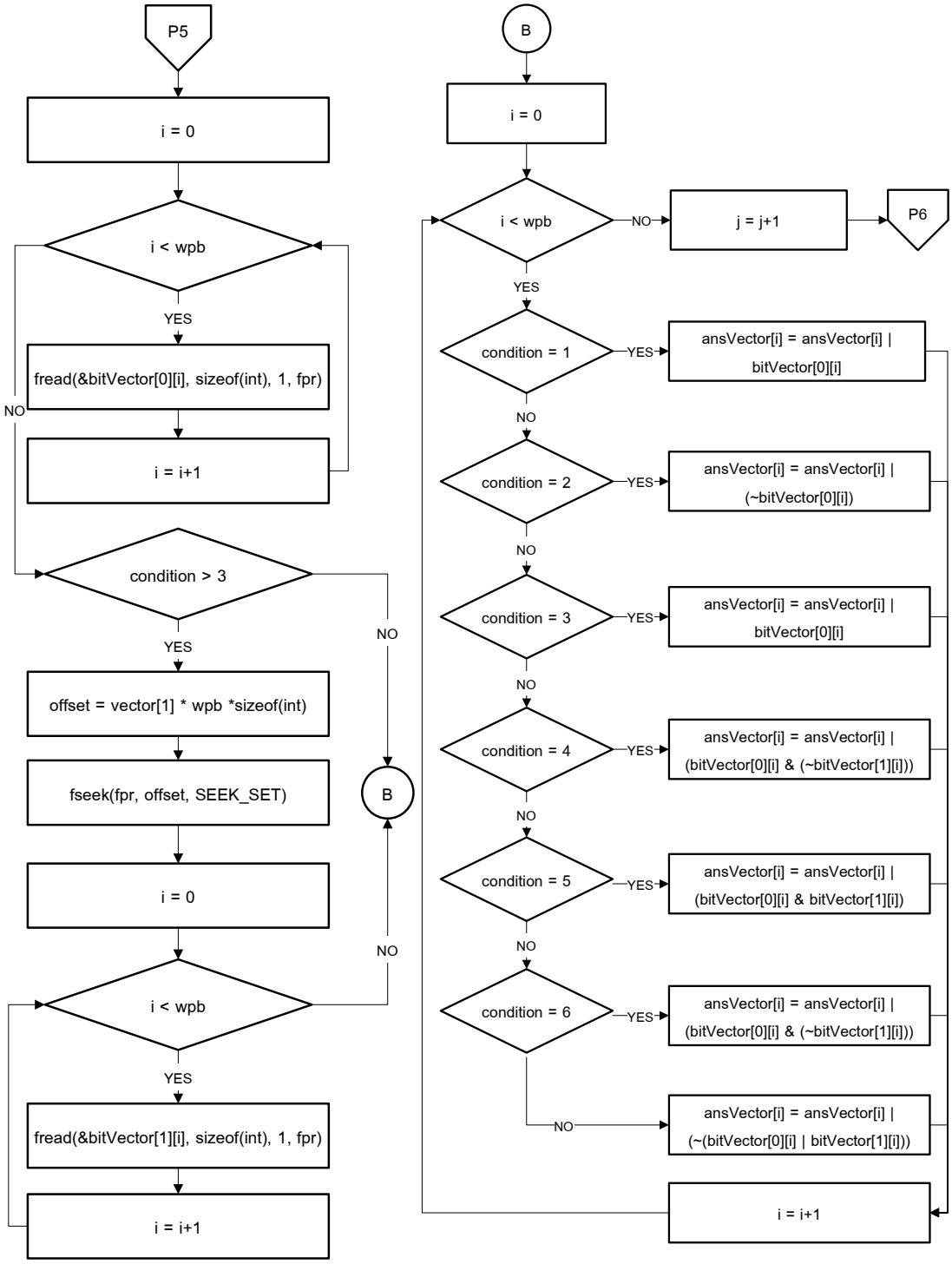


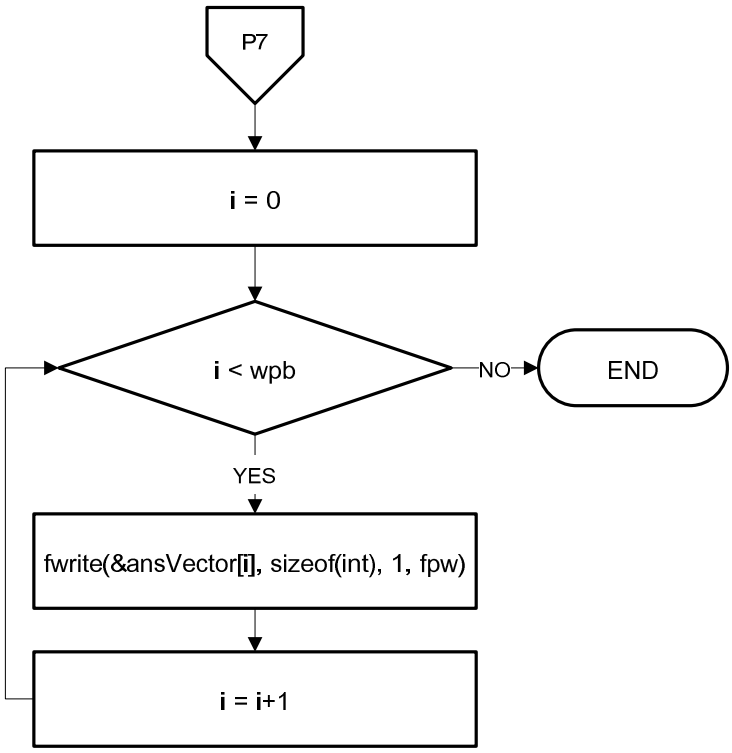

P2
i = 0
i < wpb
YES
NO
fread(&bitVector[0][i], sizeof(int), 1, fpr)
i = i + 1
condition=2 || condition=6
YES
NO
offset = vector[1] * wpb *sizeof(int)
fseek(fpr, offset, SEEK_SET)
i = 0
i < wpb
YES
NO
fread(&bitVector[1][i], sizeof(int), 1, fpr)
i = i + 1
B
B
i = 0
i < wpb
NO
begin = j
last = j
j = j+1
P3
YES
condition = 1
YES
ansVector[i] = ansVector[i] | bitVector[0][i]
NO
condition = 2
YES
ansVector[i] = ansVector[i] | (bitVector[0][i]^bitVector[1][i])
NO
condition = 3
YES
ansVector[i] = ansVector[i] | (~bitVector[0][i])
NO
condition = 4
YES
ansVector[i] = ansVector[i] | (~bitVector[0][i])
NO
condition = 5
YES
ansVector[i] = ansVector[i] | bitVector[0][i]
NO
ansVector[i] = ansVector[i] | (bitVector[0][i]^bitVector[1][i])
i = i+1


P4
i = 0
i < wpb
NO
END
YES
fwrite(&ansVector[i], sizeof(int), 1, fpw)
i = i+1

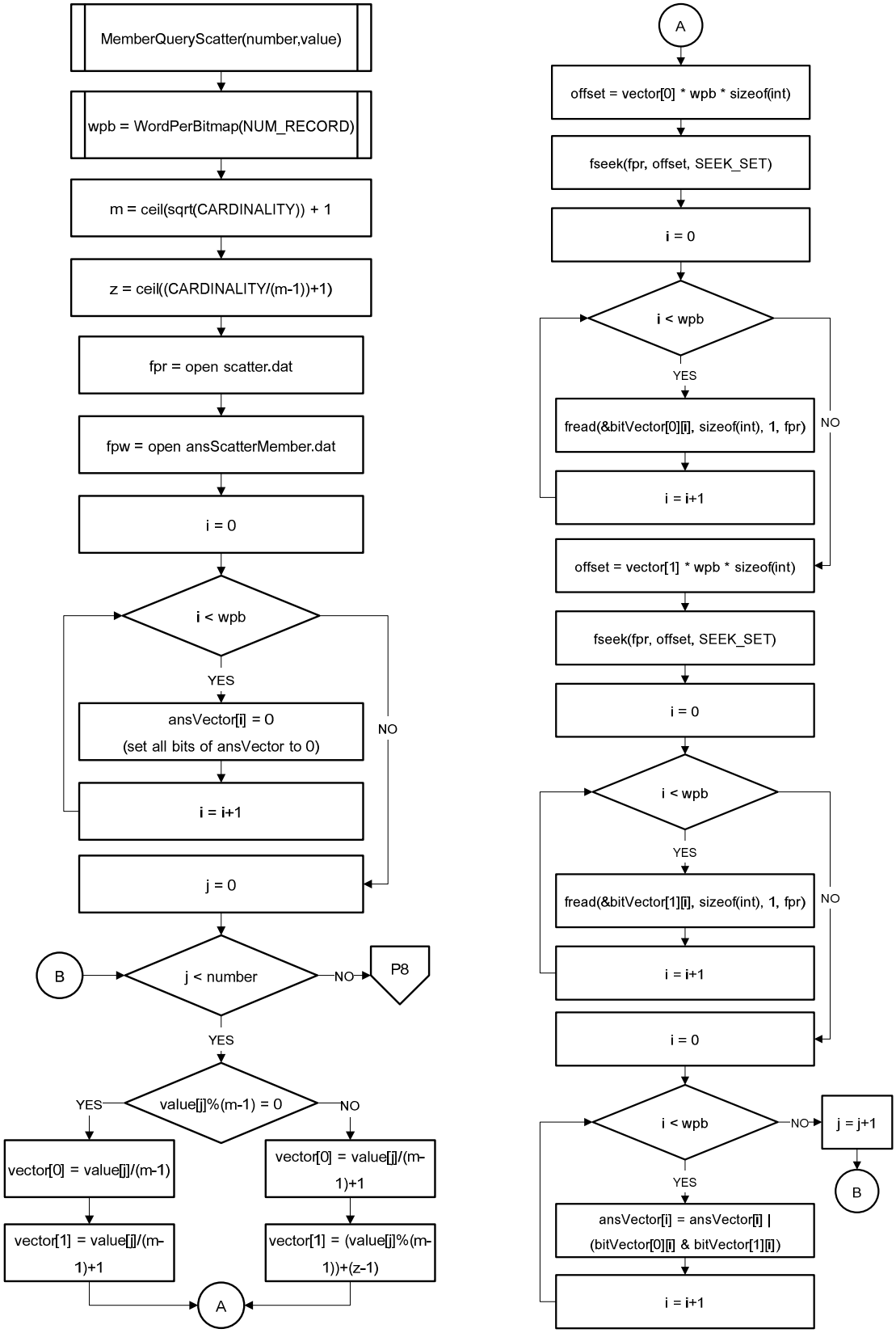
ก.5.4 ขั้นตอนวิธีการสอบถามข้อมูลแบบสมาชิกบนดัชนีบิตแมปแบบช่วง

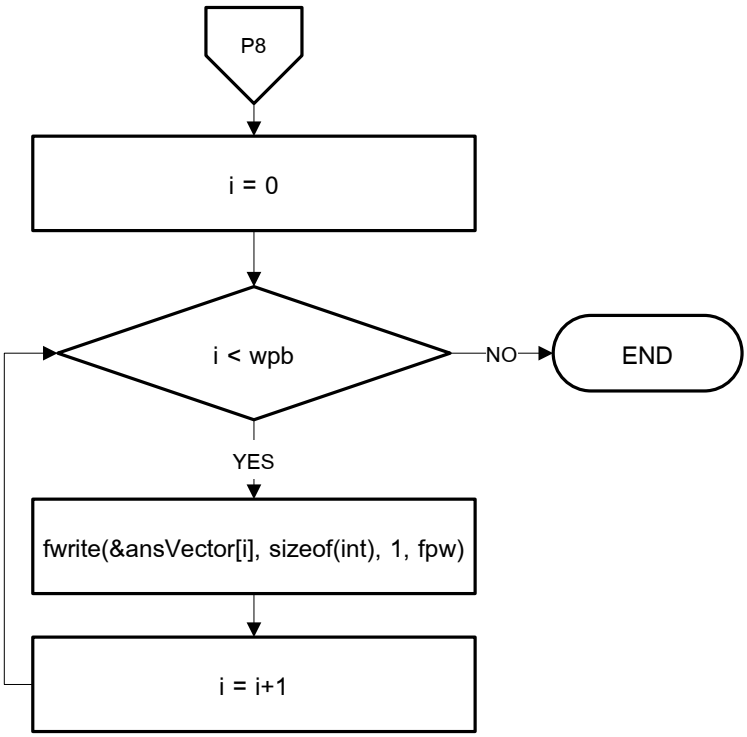



P5
i = 0
i < wpb
YES
fread(&bitVector[0][i], sizeof(int), 1, fpr)
i = i+1
NO
condition > 3
YES
NO
offset = vector[1] * wpb *sizeof(int)
fseek(fpr, offset, SEEK_SET)
i = 0
i < wpb
YES
NO
fread(&bitVector[1][i], sizeof(int), 1, fpr)
i = i+1
B
B
i = 0
i < wpb
NO
j = j+1
P6
YES
condition = 1
YES
ansVector[i] = ansVector[i] | bitVector[0][i]
NO
condition = 2
YES
ansVector[i] = ansVector[i] | (~bitVector[0][i])
NO
condition = 3
YES
ansVector[i] = ansVector[i] | bitVector[0][i]
NO
condition = 4
YES
ansVector[i] = ansVector[i] | (bitVector[0][i] & (~bitVector[1][i]))
NO
condition = 5
YES
ansVector[i] = ansVector[i] | (bitVector[0][i] & bitVector[1][i])
NO
condition = 6
YES
ansVector[i] = ansVector[i] | (bitVector[0][i] & (~bitVector[1][i]))
NO
ansVector[i] = ansVector[i] | (~(bitVector[0][i] | bitVector[1][i]))
i = i+1

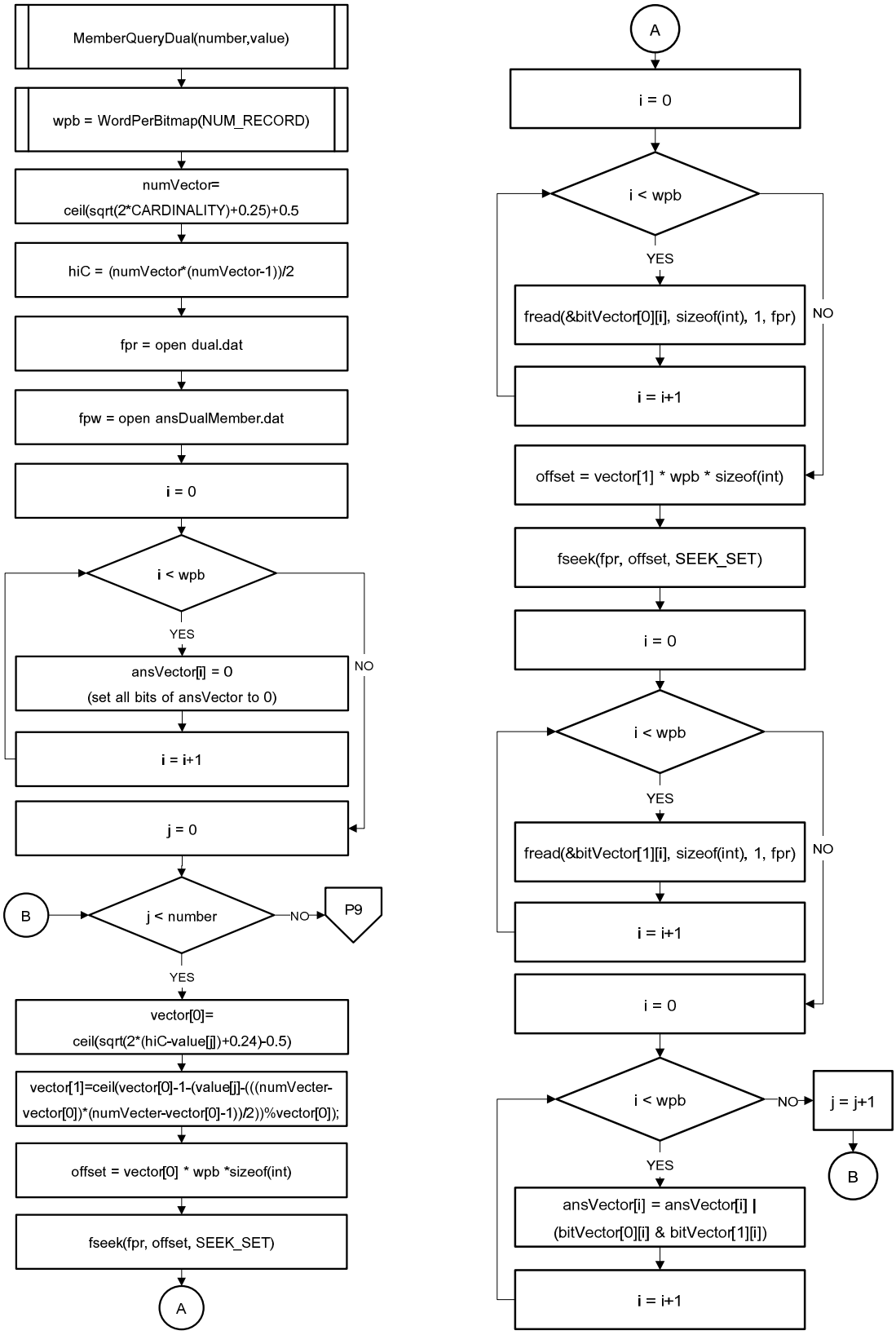

P7
i = 0
i < wpb
NO
END
YES
fwrite(&ansVector[i], sizeof(int), 1, fpw)
i = i+1

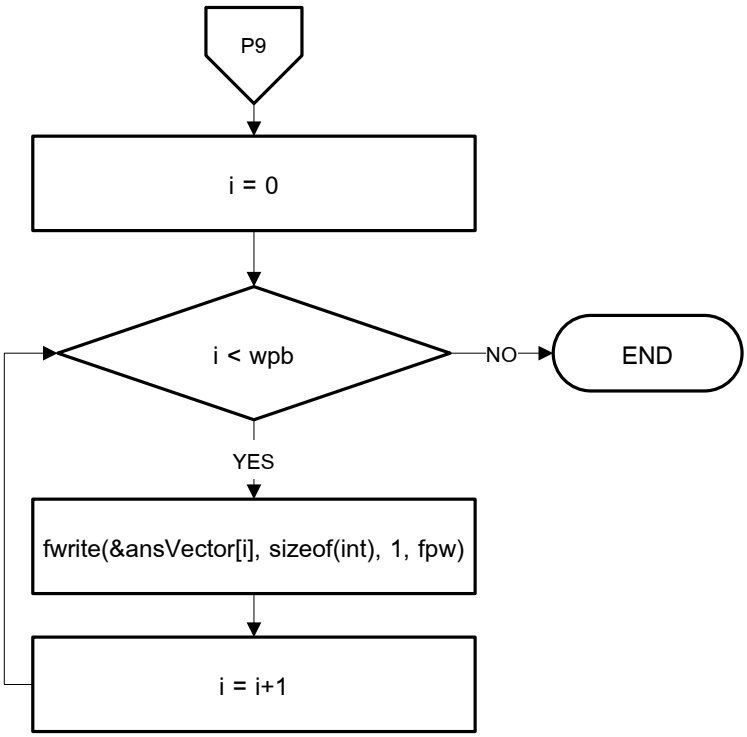
ก.5.5 ขั้นตอนวิธีการสอบถามข้อมูลแบบสมาชิกบนดัชนีบิตแมปแบบกระจาย



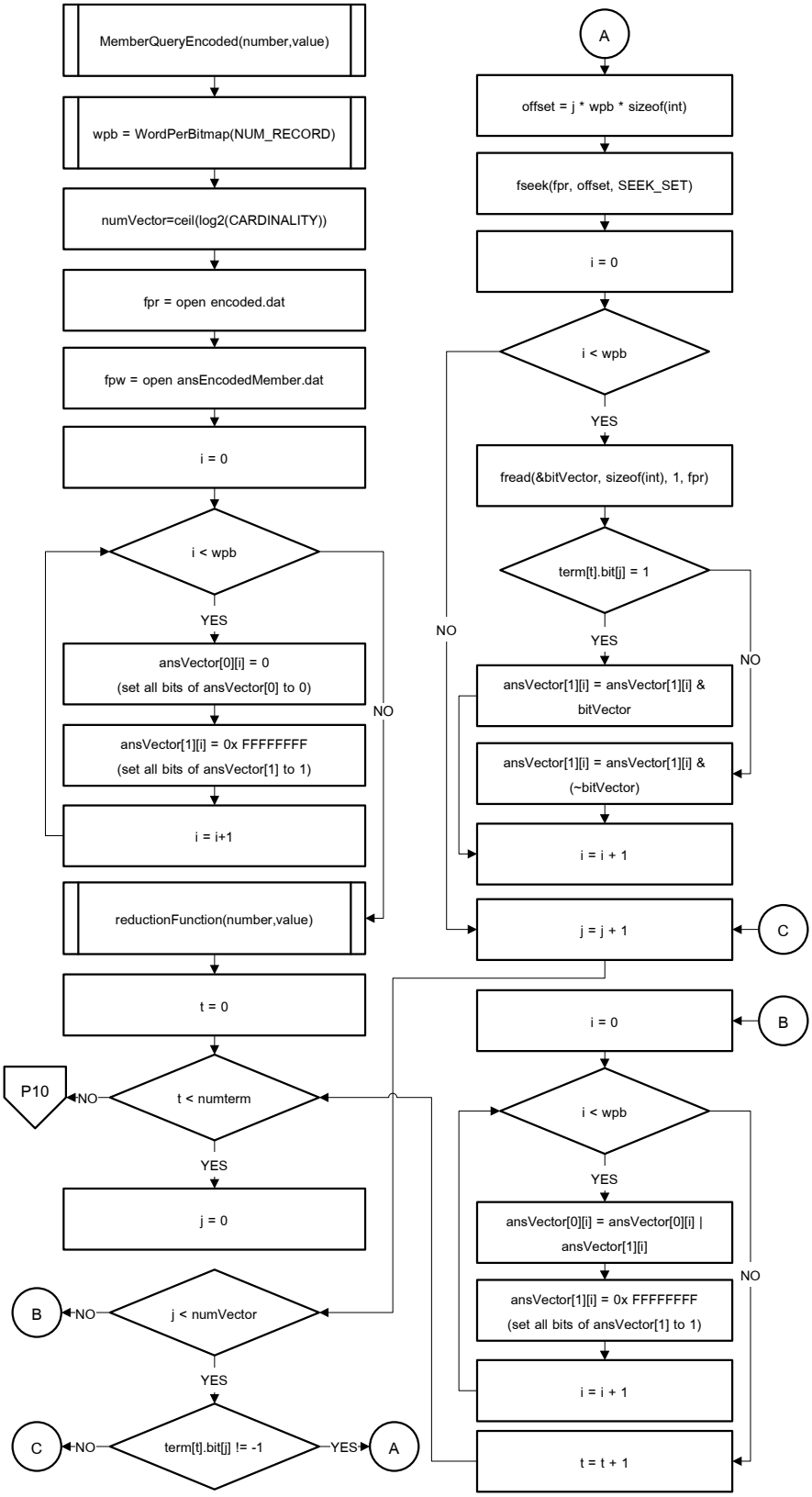

P8
i = 0
i < wpb
NO
END
YES
fwrite(&ansVector[i], sizeof(int), 1, fpw)
i = i+1

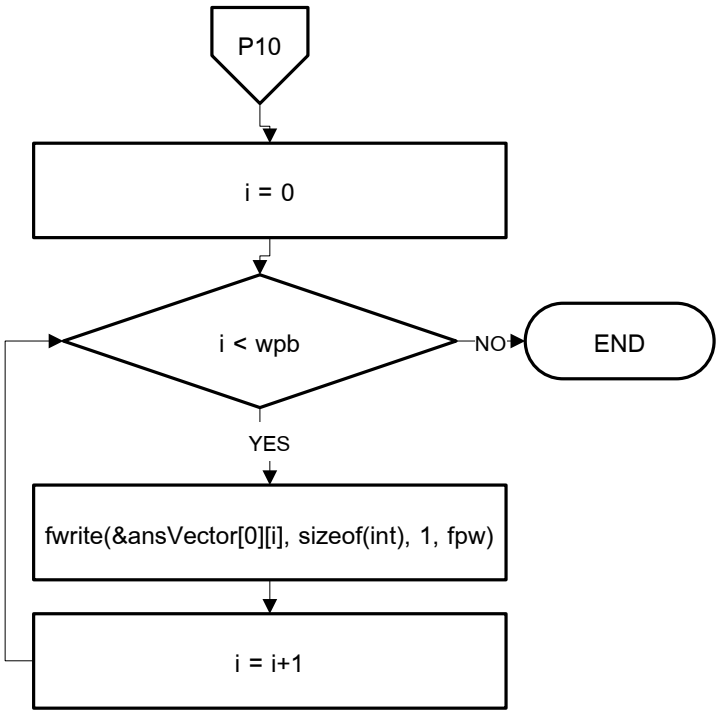
ก.5.6 ขั้นตอนวิธีการสอบถามข้อมูลแบบสมาชิกบนดัชนีบิตแมปแบบคู่กัน



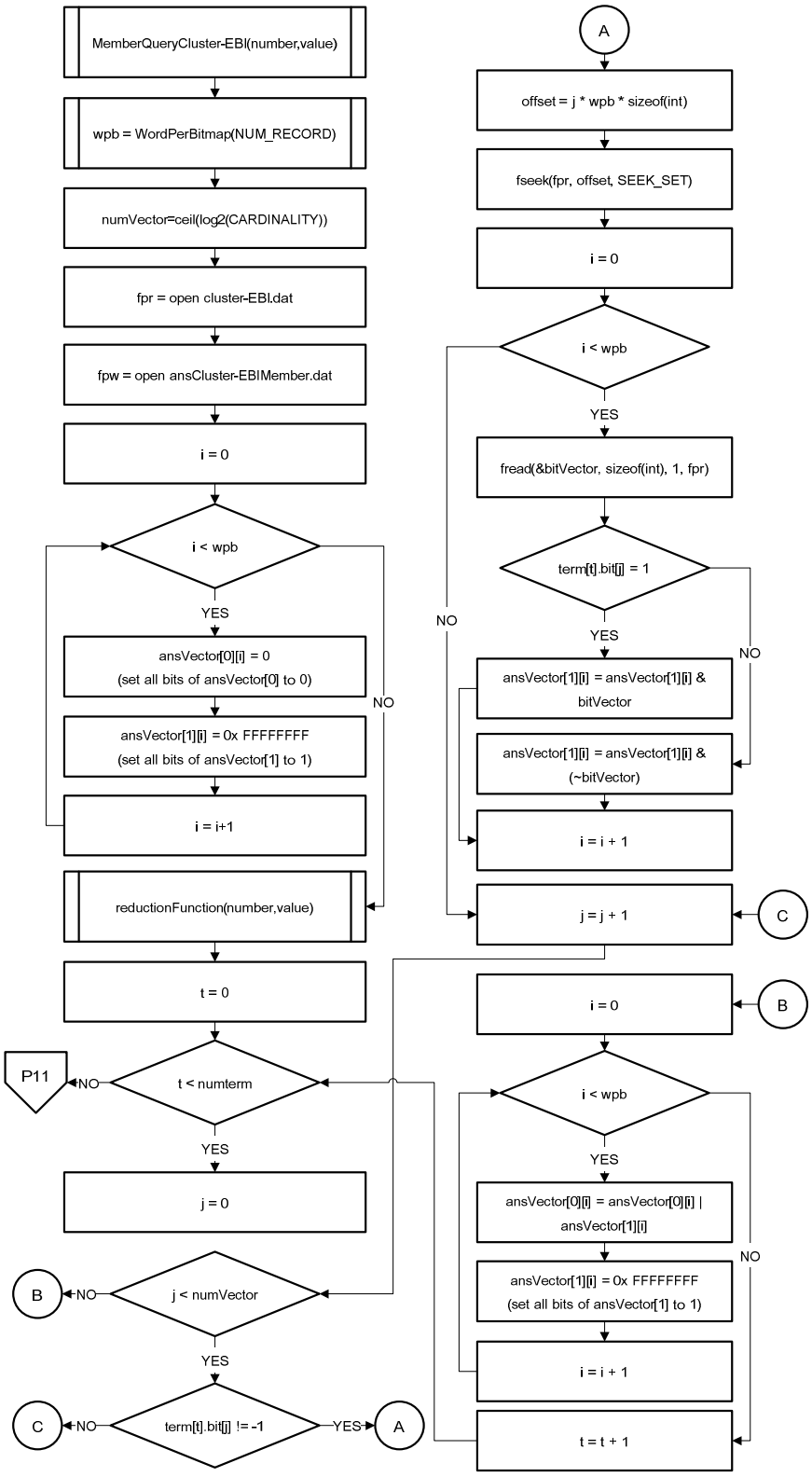

P9
i = 0
i < wpb
NO
END
YES
fwrite(&ansVector[i], sizeof(int), 1, fpw)
i = i+1

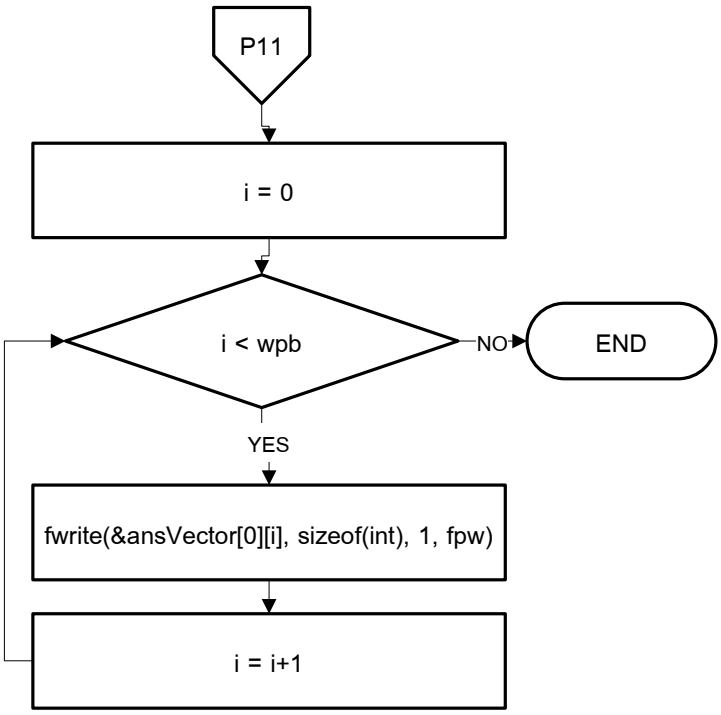
ก.5.7 ขั้นตอนวิธีการสอบถามข้อมูลแบบสมาชิกบนดัชนีบิตแมปแบบเข้ารหัสทั่วไป



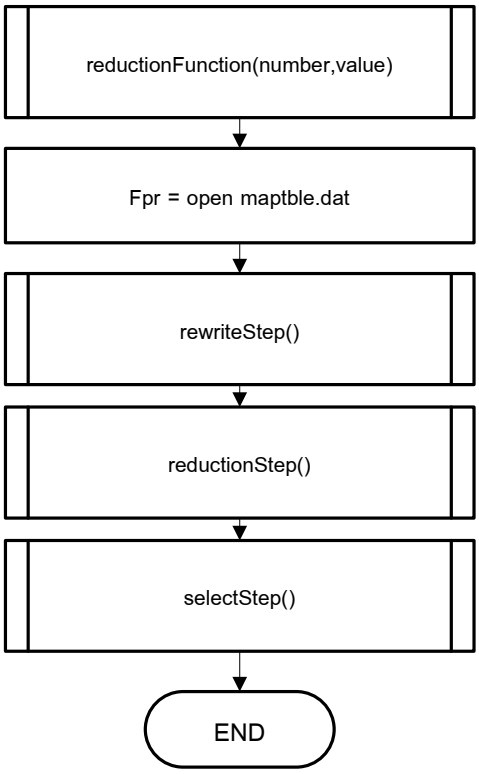

P10
i = 0
i < wpb
NO
END
YES
fwrite(&ansVector[0][i], sizeof(int), 1, fpw)
i = i+1

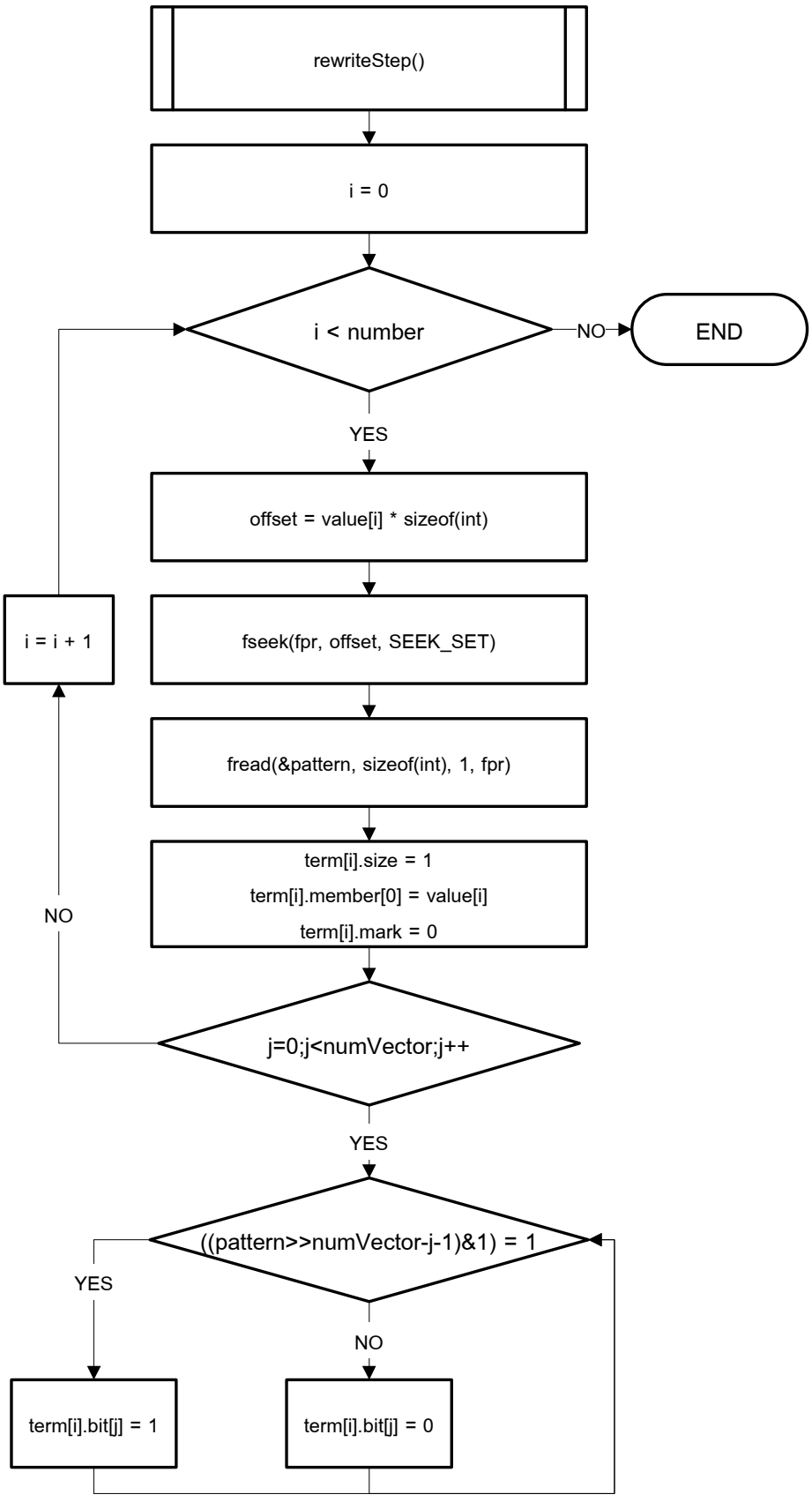
ก.5.8 ขั้นตอนวิธีการสอบถามข้อมูลแบบสมาชิกบนดัชนีบิตแมปแบบเข้ารหัสที่ใช้เทคนิคการจัดกลุ่มข้อมูล

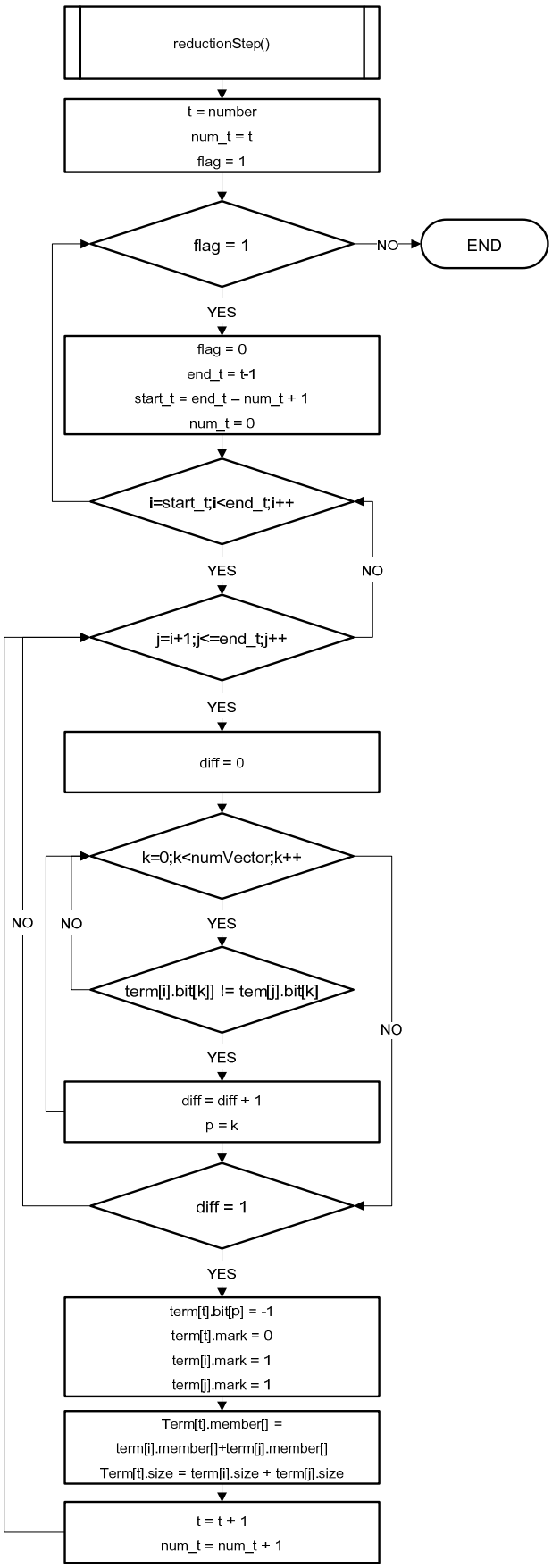


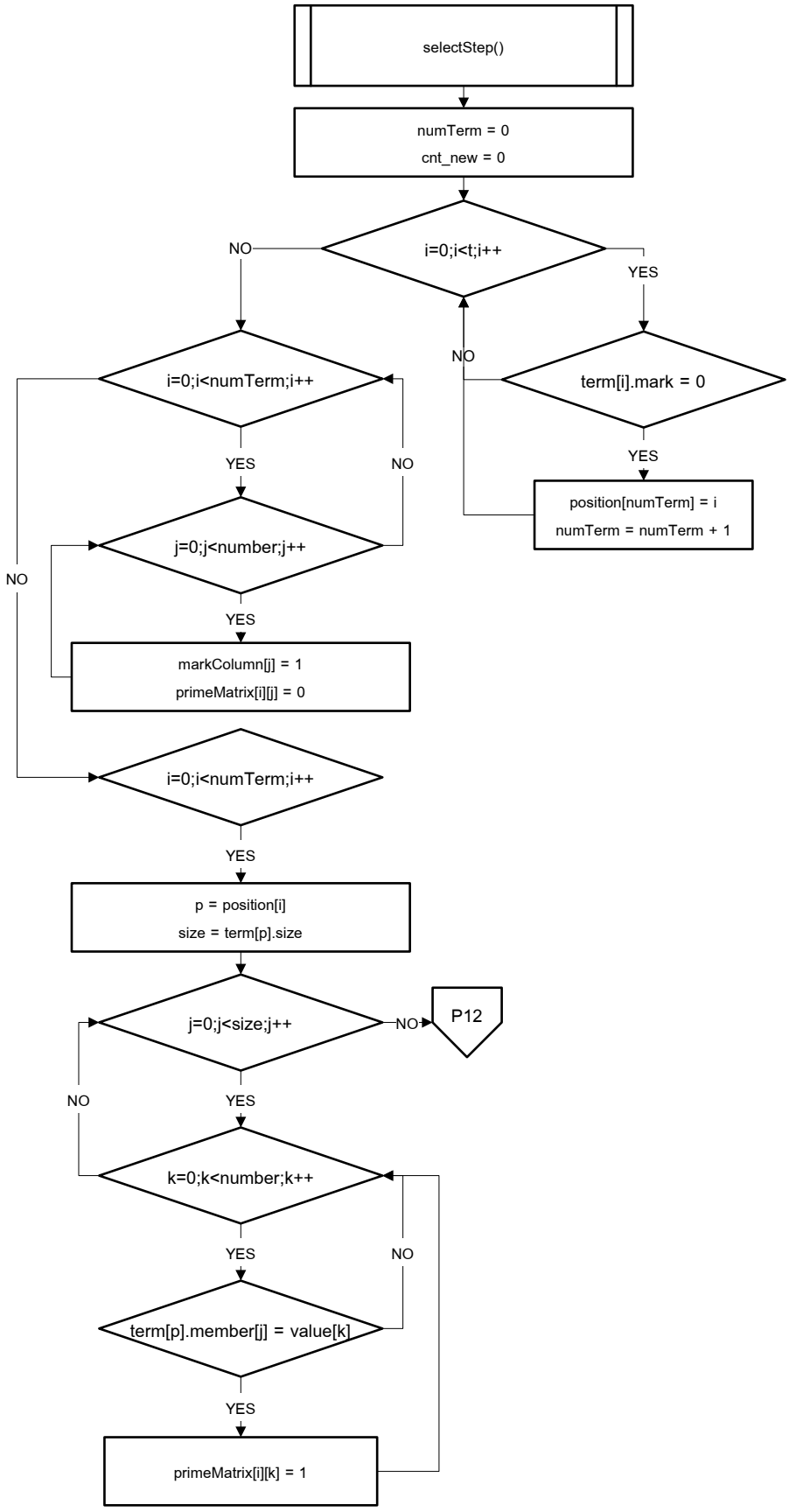

P11
i = 0
i < wpb
NO
END
YES
fwrite(&ansVector[0][i], sizeof(int), 1, fpw)
i = i+1

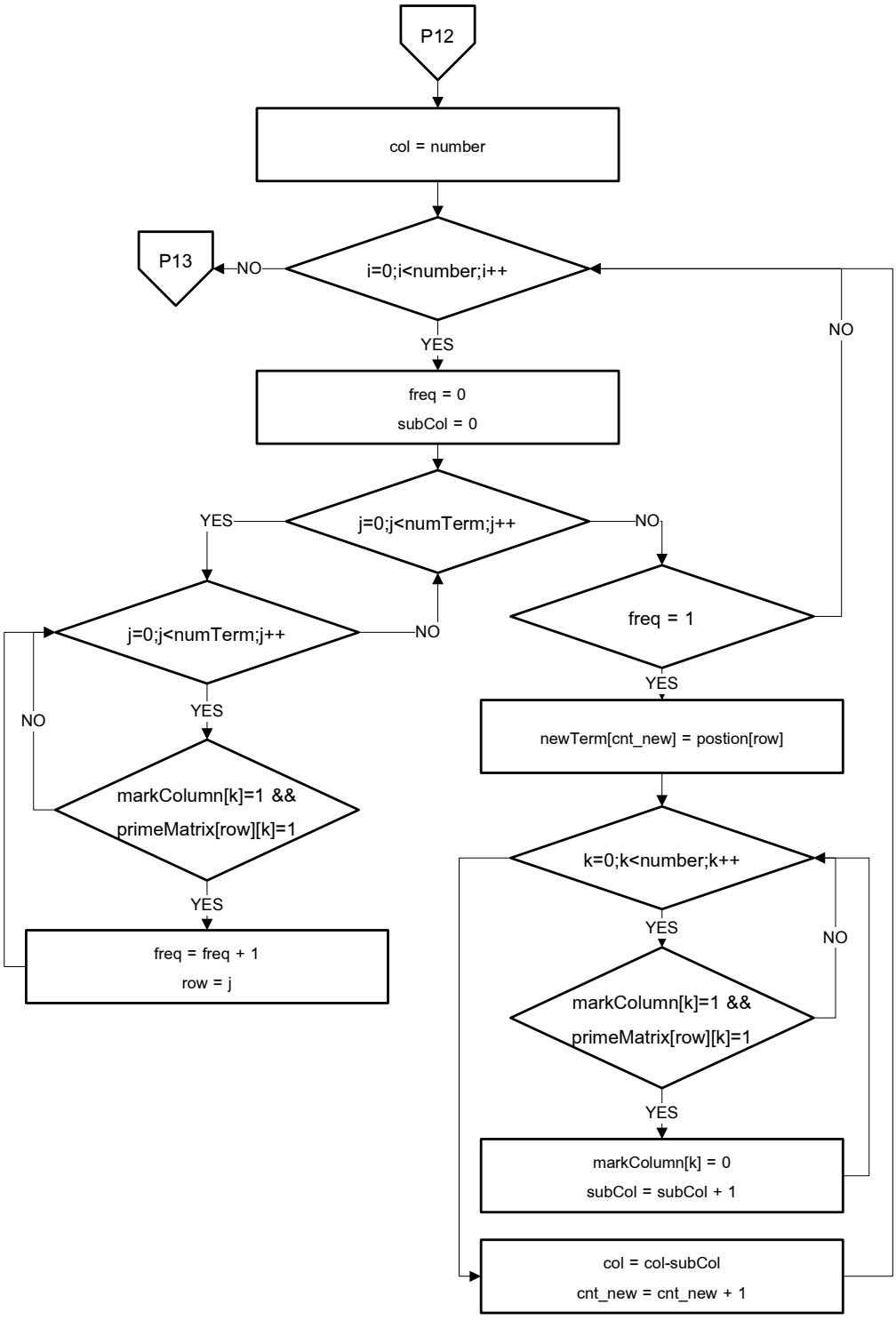
ก.5.9 ขั้นตอนวิธีการลดรูปฟังก์ชันการเข้าถึงข้อมูลบนดัชนีบิตแมปแบบเข้ารหัสทั่วไป และดัชนีบิตแมปแบบเข้ารหัสที่ใช้เทคนิคการจัดกลุ่มข้อมูล

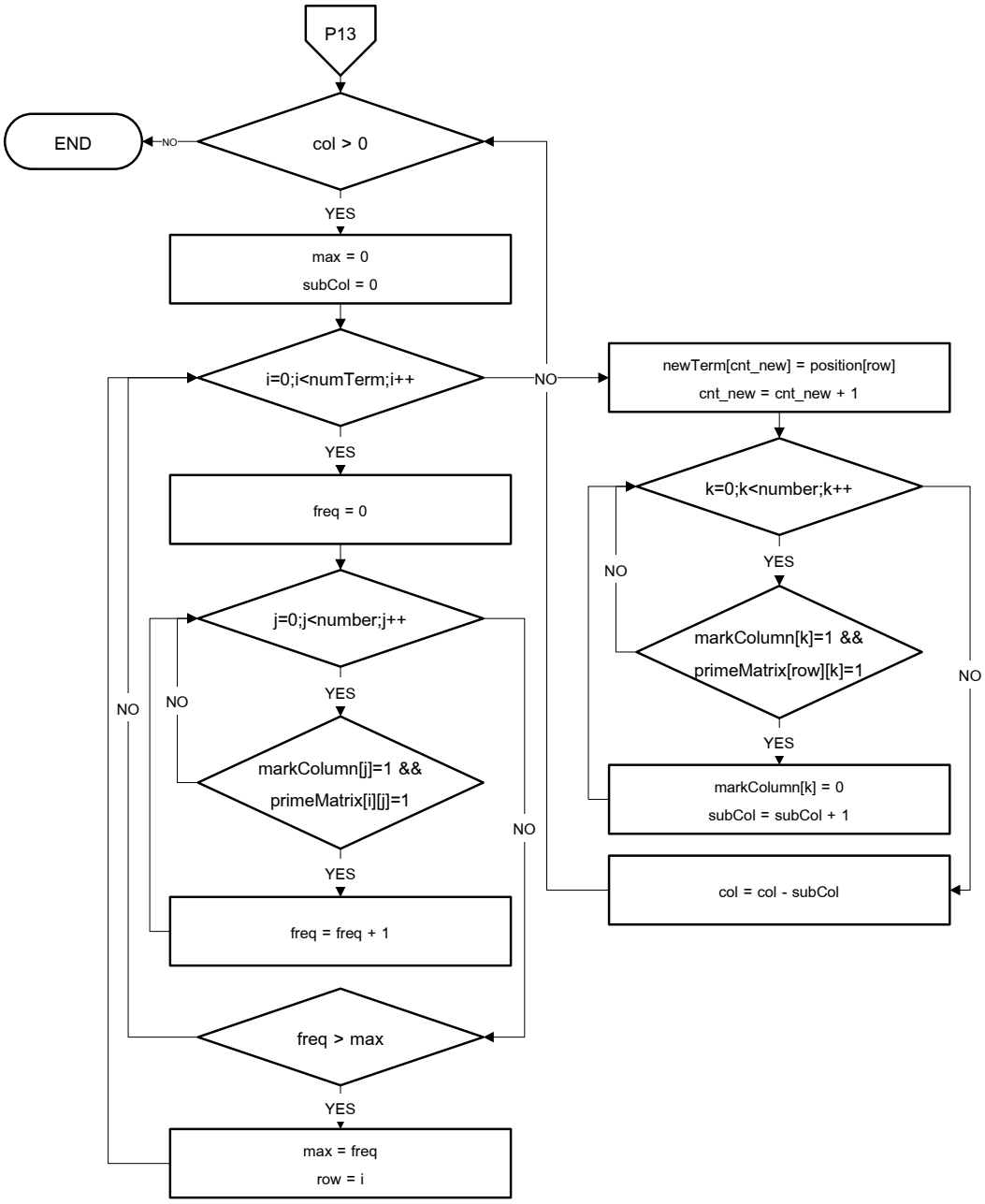



rewriteStep()
i = 0
i < number
NO
END
YES
offset = value[i] * sizeof(int)
fseek(fpr, offset, SEEK_SET)
fread(&pattern, sizeof(int), 1, fpr)
term[i].size = 1
term[i].member[0] = value[i]
term[i].mark = 0
j=0;j<numVector;j++
NO
i = i + 1
YES
((pattern>>numVector-j-1)&1) = 1
YES
NO
term[i].bit[j] = 1
term[i].bit[j] = 0

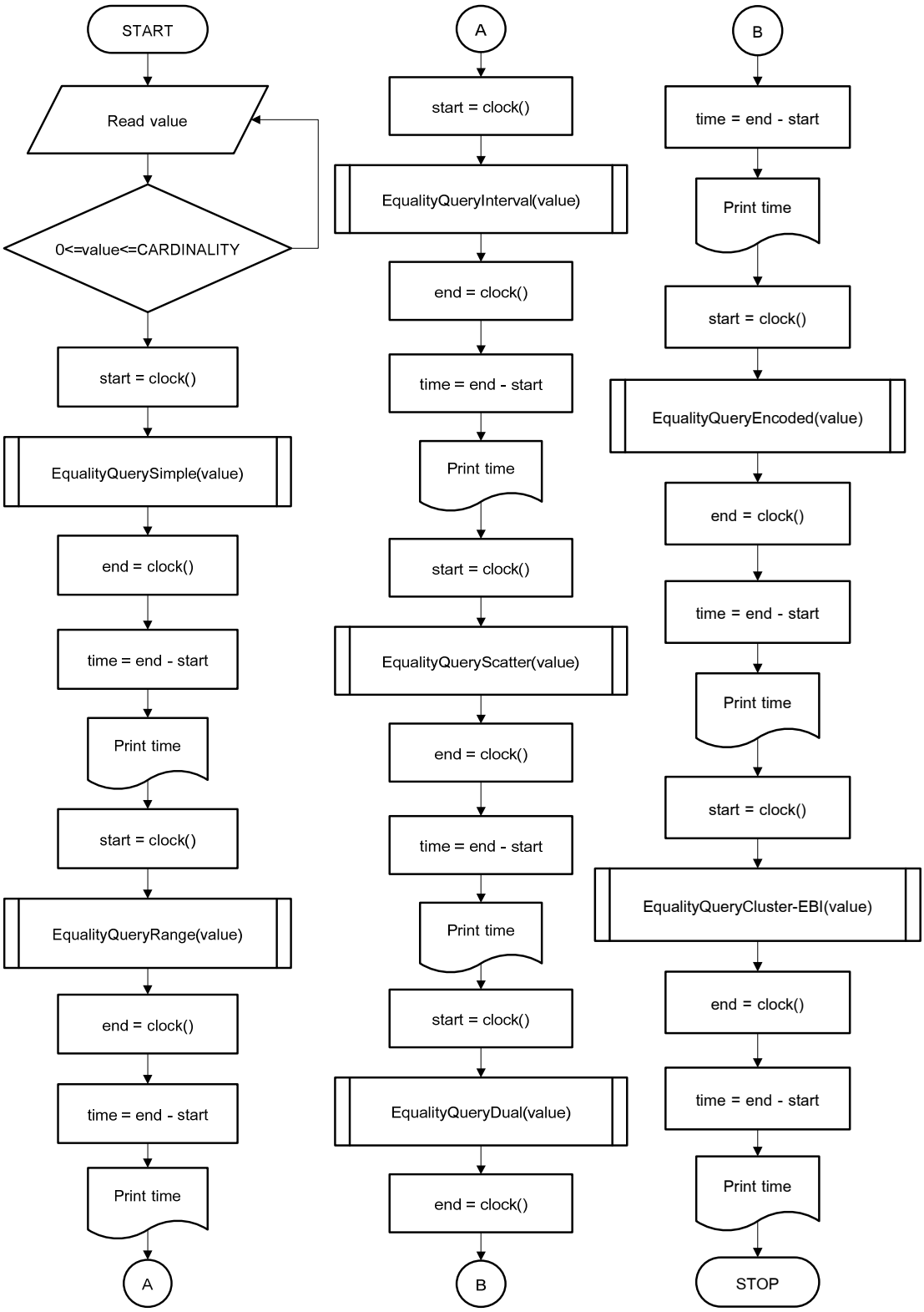

reductionStep()
t = number
num_t = t
flag = 1
flag = 1
NO
END
YES
flag = 0
end_t = t-1
start_t = end_t – num_t + 1
num_t = 0
i=start_t;i<end_t;i++
YES
NO
j=i+1;j<=end_t;j++
YES
diff = 0
k=0;k<numVector;k++
NO
NO
YES
term[i].bit[k] != tem[j].bit[k]
NO
YES
diff = diff + 1
p = k
diff = 1
YES
term[t].bit[p] = -1
term[t].mark = 0
term[i].mark = 1
term[j].mark = 1
Term[t].member[] =
term[i].member[]+term[j].member[]
Term[t].size = term[i].size + term[j].size
t = t + 1
num_t = num_t + 1


selectStep()
numTerm = 0
cnt_new = 0
i=0;i<t;i++
NO
YES
term[i].mark = 0
NO
YES
position[numTerm] = i
numTerm = numTerm + 1
i=0;i<numTerm;i++
YES
NO
j=0;j<number;j++
NO
YES
markColumn[j] = 1
primeMatrix[i][j] = 0
i=0;i<numTerm;i++
YES
p = position[i]
size = term[p].size
j=0;j<size;j++
NO
P12
NO
YES
k=0;k<number;k++
YES
NO
term[p].member[j] = value[k]
YES
primeMatrix[i][k] = 1

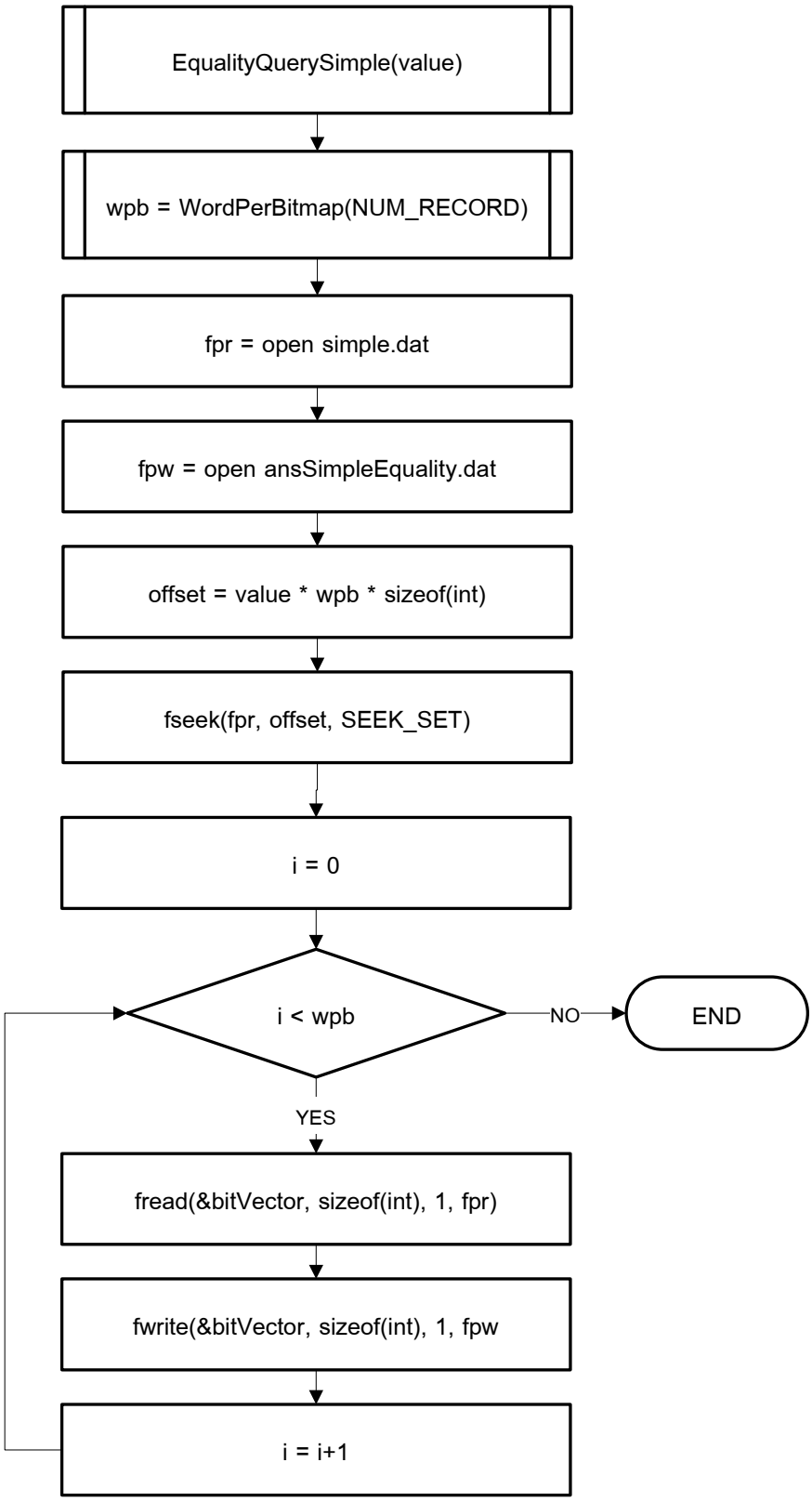

P12
col = number
P13
NO
i=0;i<number;i++
YES
NO
freq = 0
subCol = 0
YES
j=0;j<numTerm;j++
NO
freq = 1
YES
j=0;j<numTerm;j++
NO
YES
newTerm[cnt_new] = postion[row]
NO
markColumn[k]=1 &&
primeMatrix[row][k]=1
k=0;k<number;k++
YES
NO
YES
markColumn[k]=1 &&
primeMatrix[row][k]=1
YES
freq = freq + 1
row = j
markColumn[k] = 0
subCol = subCol + 1
col = col-subCol
cnt_new = cnt_new + 1


P13
col > 0
END
NO
YES
max = 0
subCol = 0
i=0;i<numTerm;i++
NO
YES
freq = 0
j=0;j<number;j++
NO
YES
markColumn[j]=1 &&
primeMatrix[i][j]=1
NO
YES
freq = freq + 1
freq > max
NO
YES
max = freq
row = i
NO
newTerm[cnt_new] = position[row]
cnt_new = cnt_new + 1
k=0;k<number;k++
NO
YES
markColumn[k]=1 &&
primeMatrix[row][k]=1
NO
YES
markColumn[k] = 0
subCol = subCol + 1
col = col - subCol

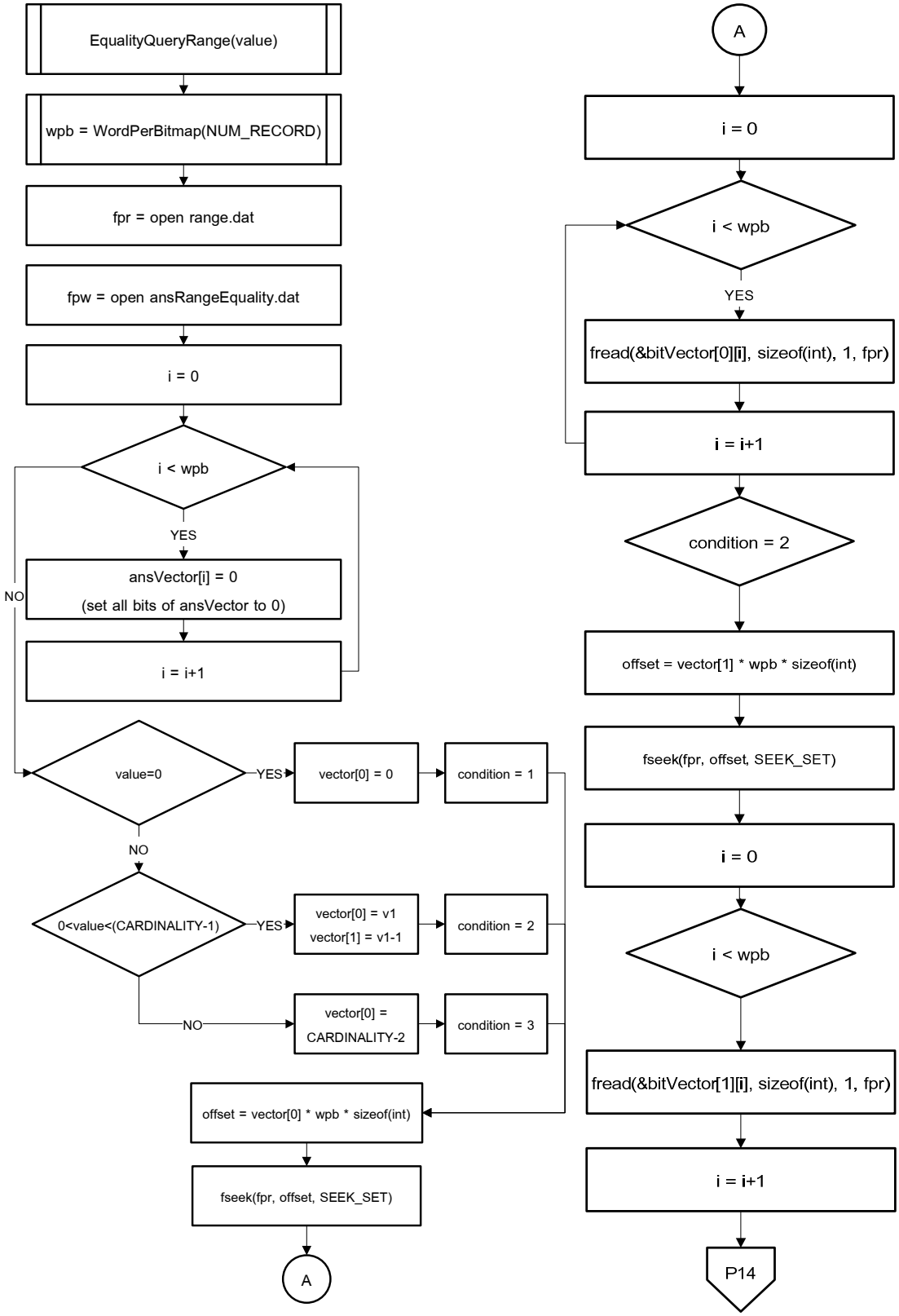
ก.5.10 ขั้นตอนวิธีการทำงานของการสอบถามข้อมูลแบบค่าเท่ากันบนดัชนีบิตแมปทั้ง 7 แบบ

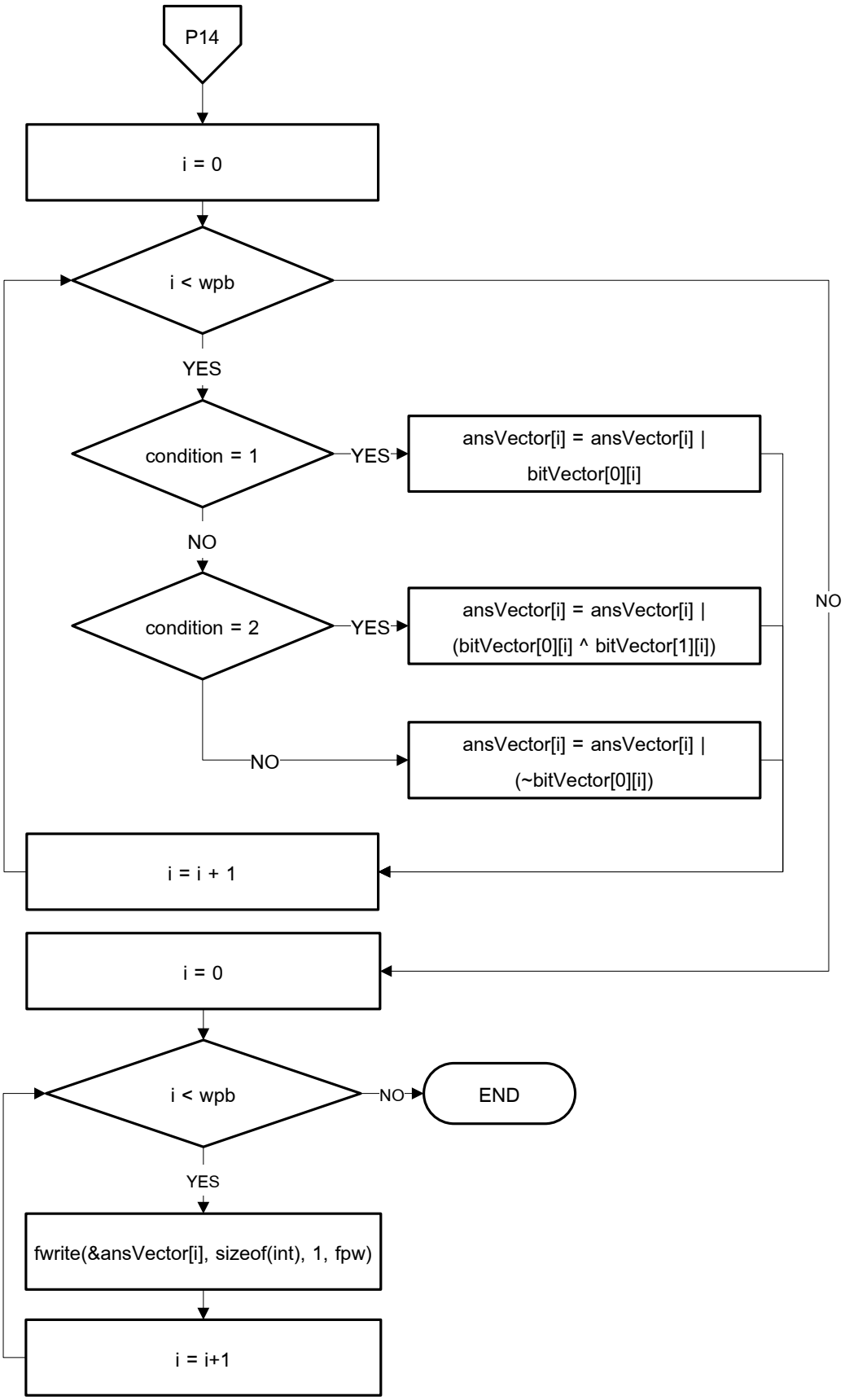


ก.5.11 ขั้นตอนวิธีการสอบถามข้อมูลแบบค่าเท่ากันบนดัชนีบิตแมปแบบพื้นฐาน

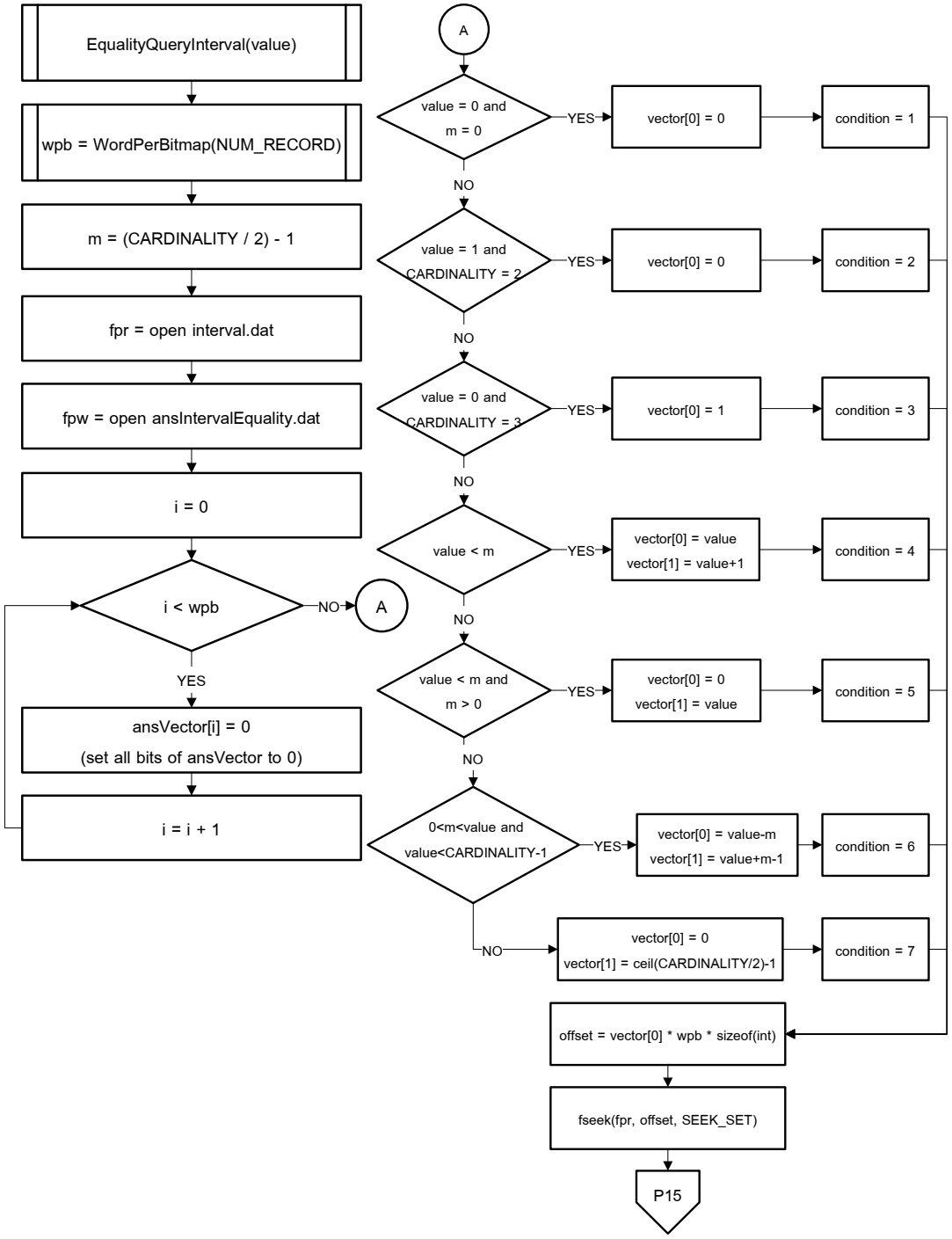


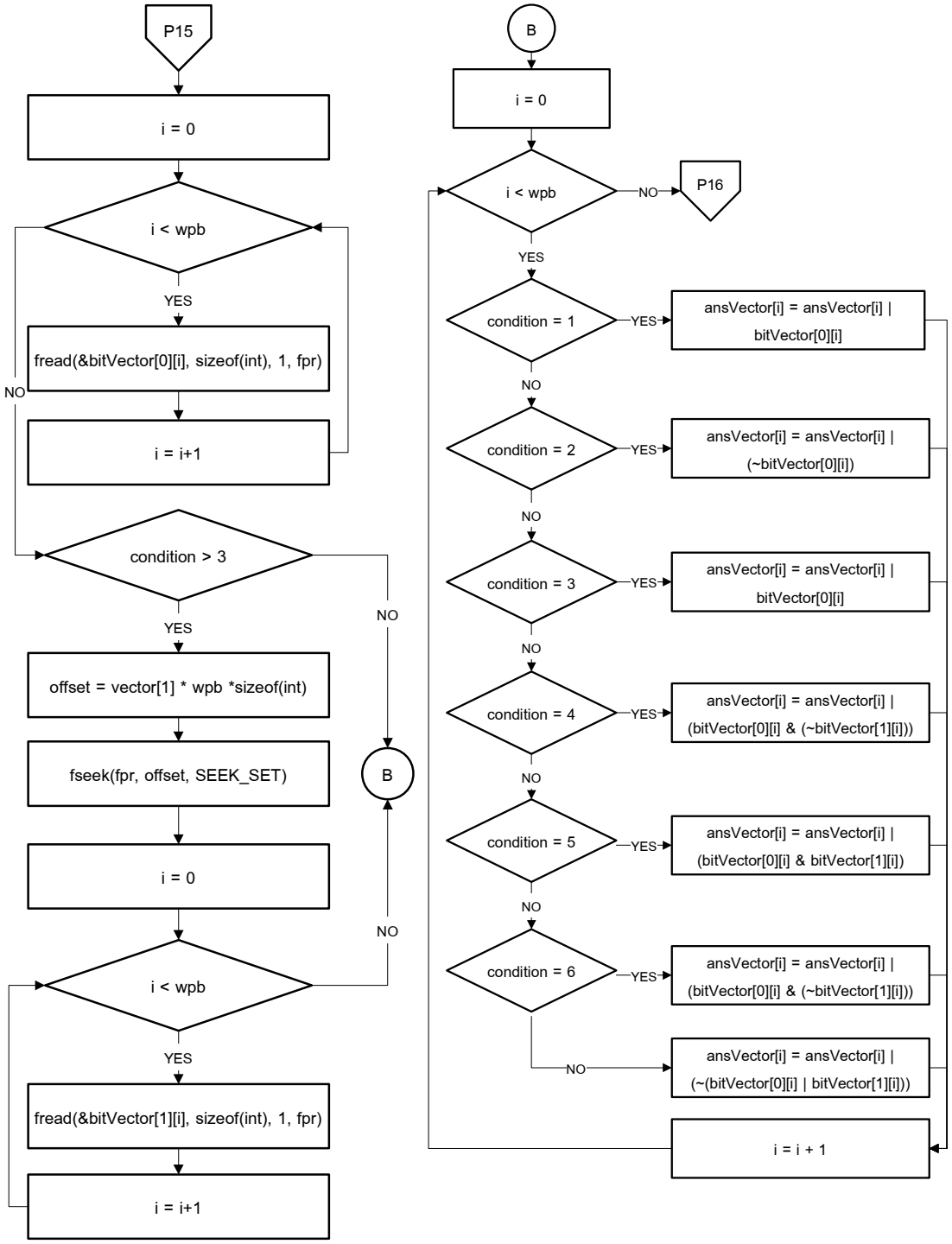
ก.5.12 ขั้นตอนวิธีการสอบถามข้อมูลแบบค่าเท่ากันบนดัชนีบิตแมปแบบแถว

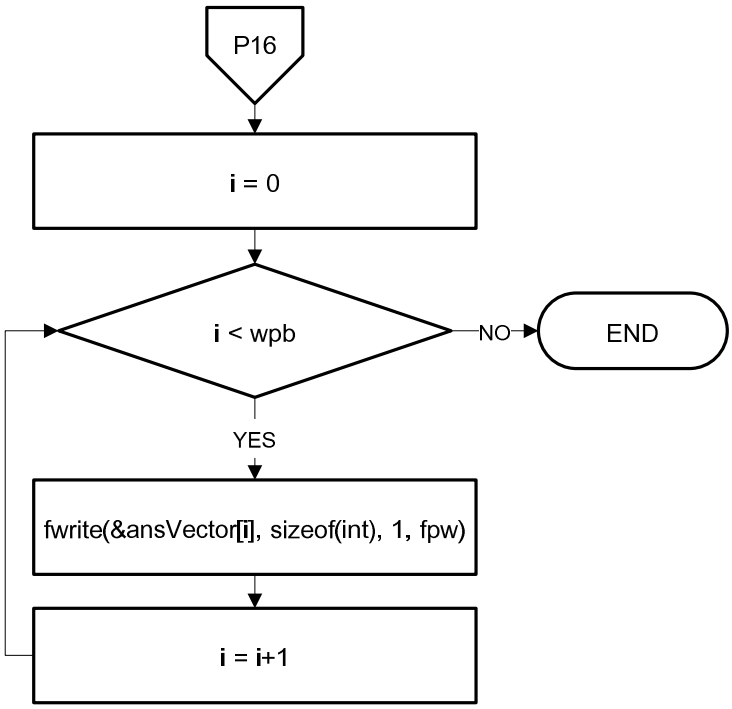



P14
i = 0
i < wpb
YES
condition = 1
YES
ansVector[i] = ansVector[i] |
bitVector[0][i]
NO
condition = 2
YES
ansVector[i] = ansVector[i] |
(bitVector[0][i] ^ bitVector[1][i])
NO
ansVector[i] = ansVector[i] |
(~bitVector[0][i])
NO
i = i + 1
i = 0
i < wpb
NO
END
YES
fwrite(&ansVector[i], sizeof(int), 1, fpw)
i = i+1

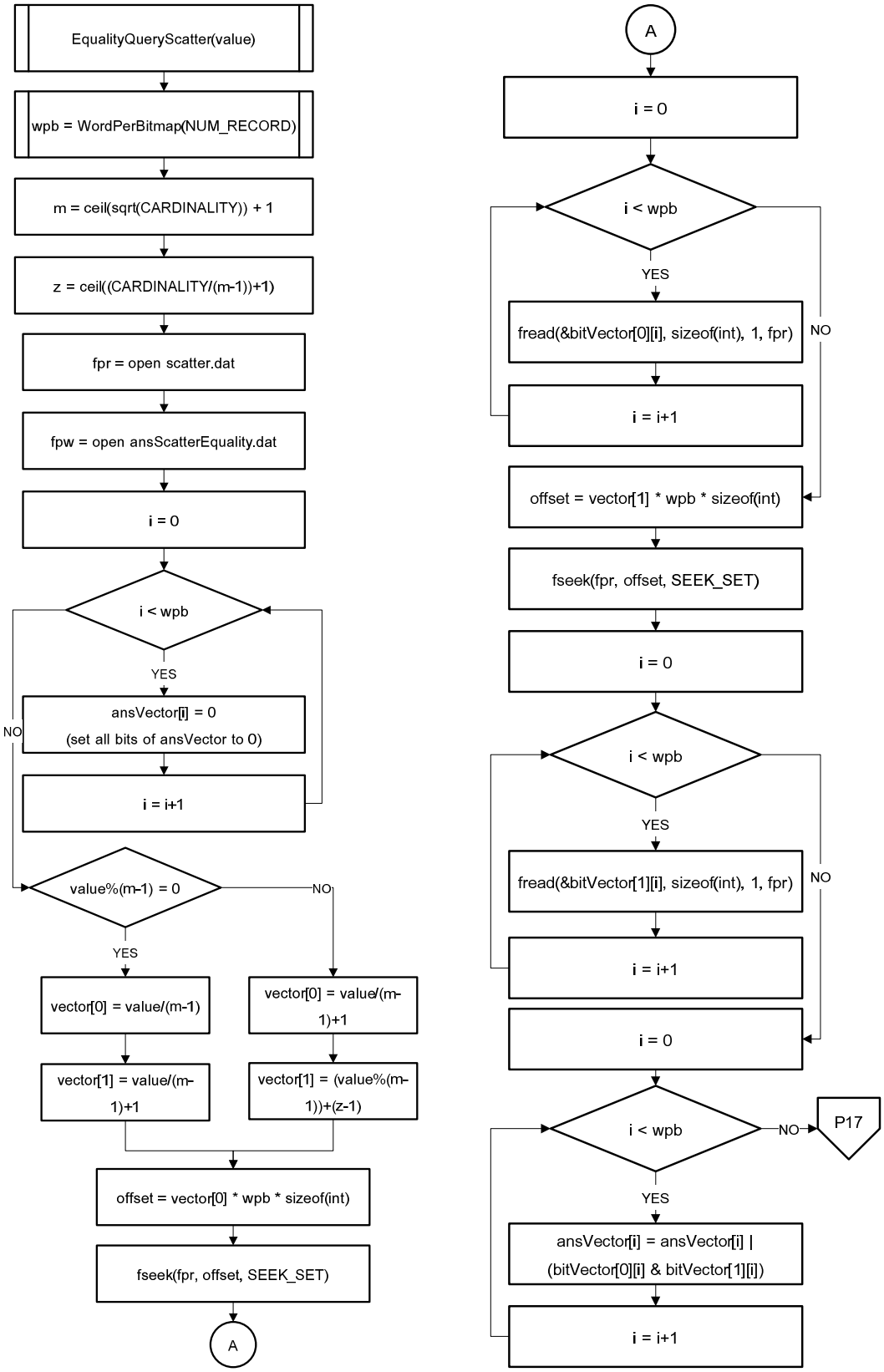
ก.5.13 ขั้นตอนวิธีการสอบถามข้อมูลแบบค่าเท่ากันบนดัชนีบิตแมปแบบช่วง

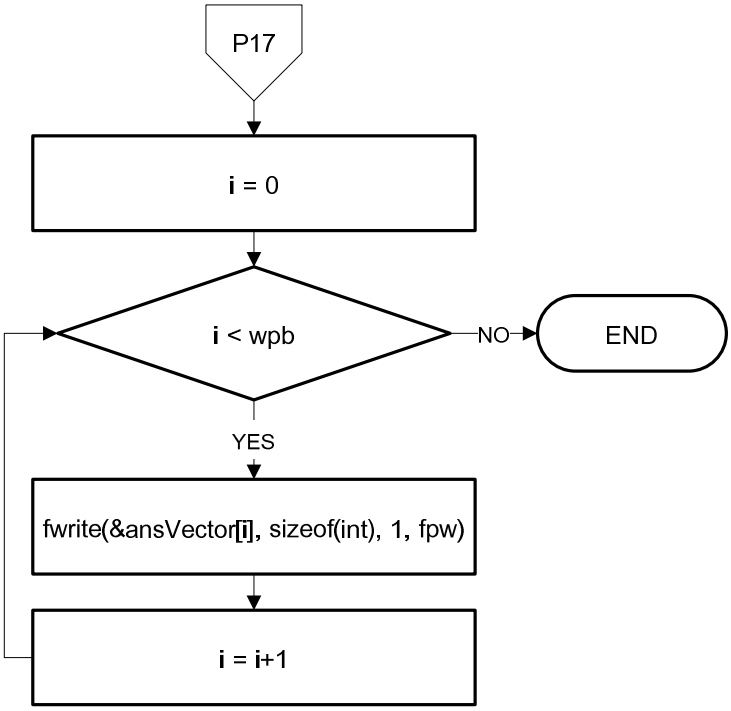



P15
i = 0
i < wpb
YES
fread(&bitVector[0][i], sizeof(int), 1, fpr)
i = i+1
NO
condition > 3
YES
NO
offset = vector[1] * wpb *sizeof(int)
fseek(fpr, offset, SEEK_SET)
i = 0
i < wpb
YES
NO
fread(&bitVector[1][i], sizeof(int), 1, fpr)
i = i+1
B
B
i = 0
i < wpb
NO
P16
YES
condition = 1
YES
ansVector[i] = ansVector[i] | bitVector[0][i]
NO
condition = 2
YES
ansVector[i] = ansVector[i] | (~bitVector[0][i])
NO
condition = 3
YES
ansVector[i] = ansVector[i] | bitVector[0][i]
NO
condition = 4
YES
ansVector[i] = ansVector[i] | (bitVector[0][i] & (~bitVector[1][i]))
NO
condition = 5
YES
ansVector[i] = ansVector[i] | (bitVector[0][i] & bitVector[1][i])
NO
condition = 6
YES
ansVector[i] = ansVector[i] | (bitVector[0][i] & (~bitVector[1][i]))
NO
ansVector[i] = ansVector[i] | (~(bitVector[0][i] | bitVector[1][i]))
i = i + 1

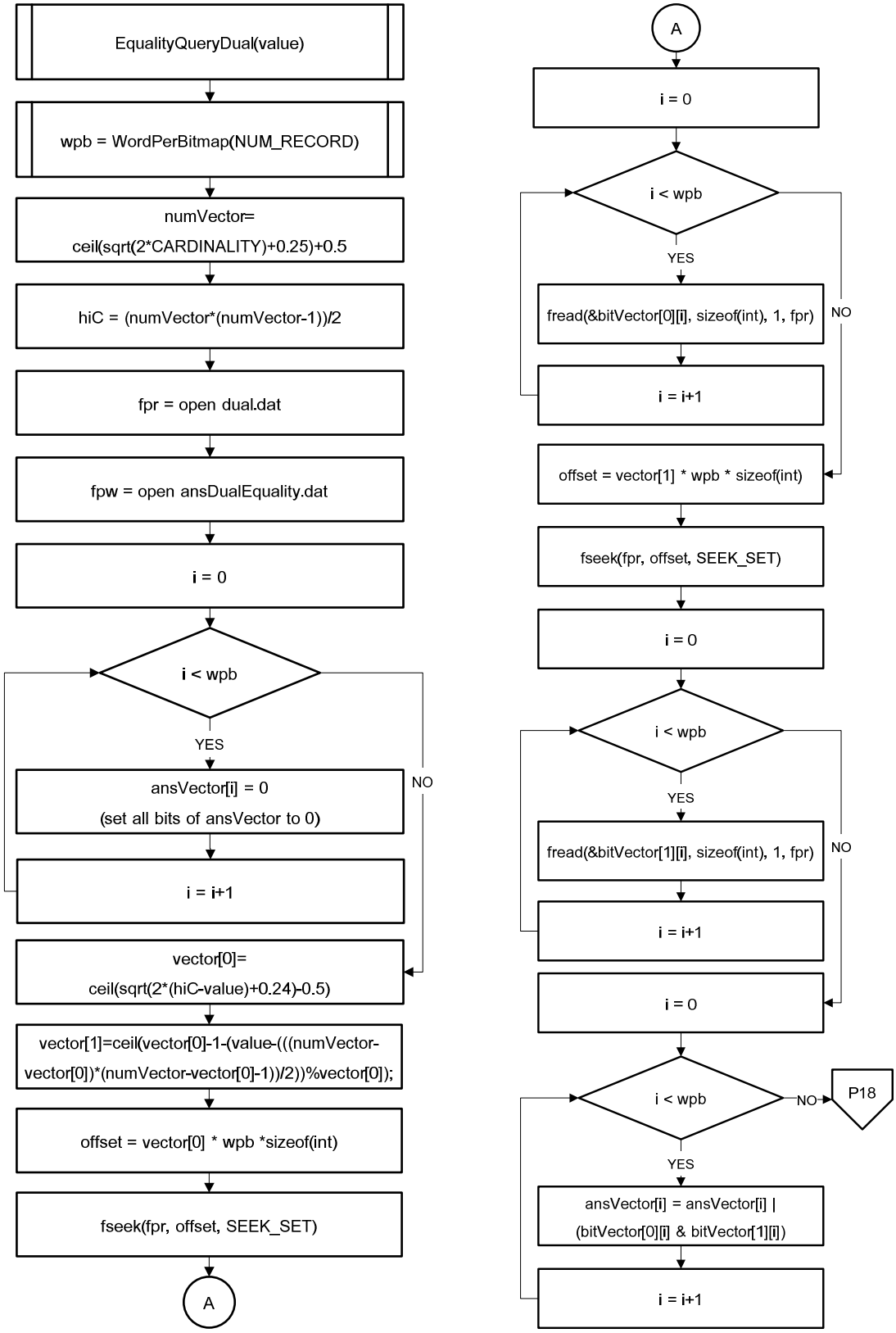

P16
i = 0
i < wpb
NO
END
YES
fwrite(&ansVector[i], sizeof(int), 1, fpw)
i = i+1

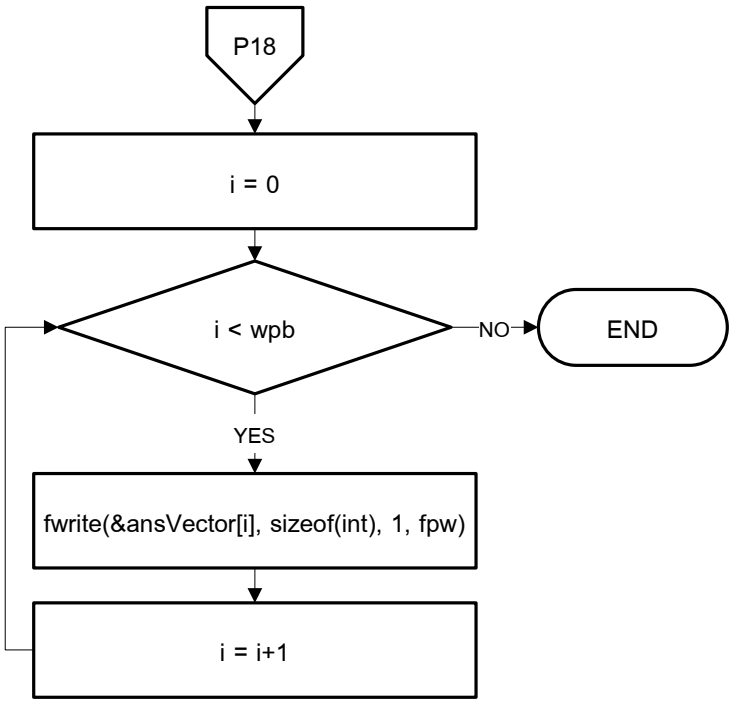
ก.5.14 ขั้นตอนวิธีการสอบถามข้อมูลแบบค่าเท่ากันบนดัชนีบิตแมปแบบกระจาย



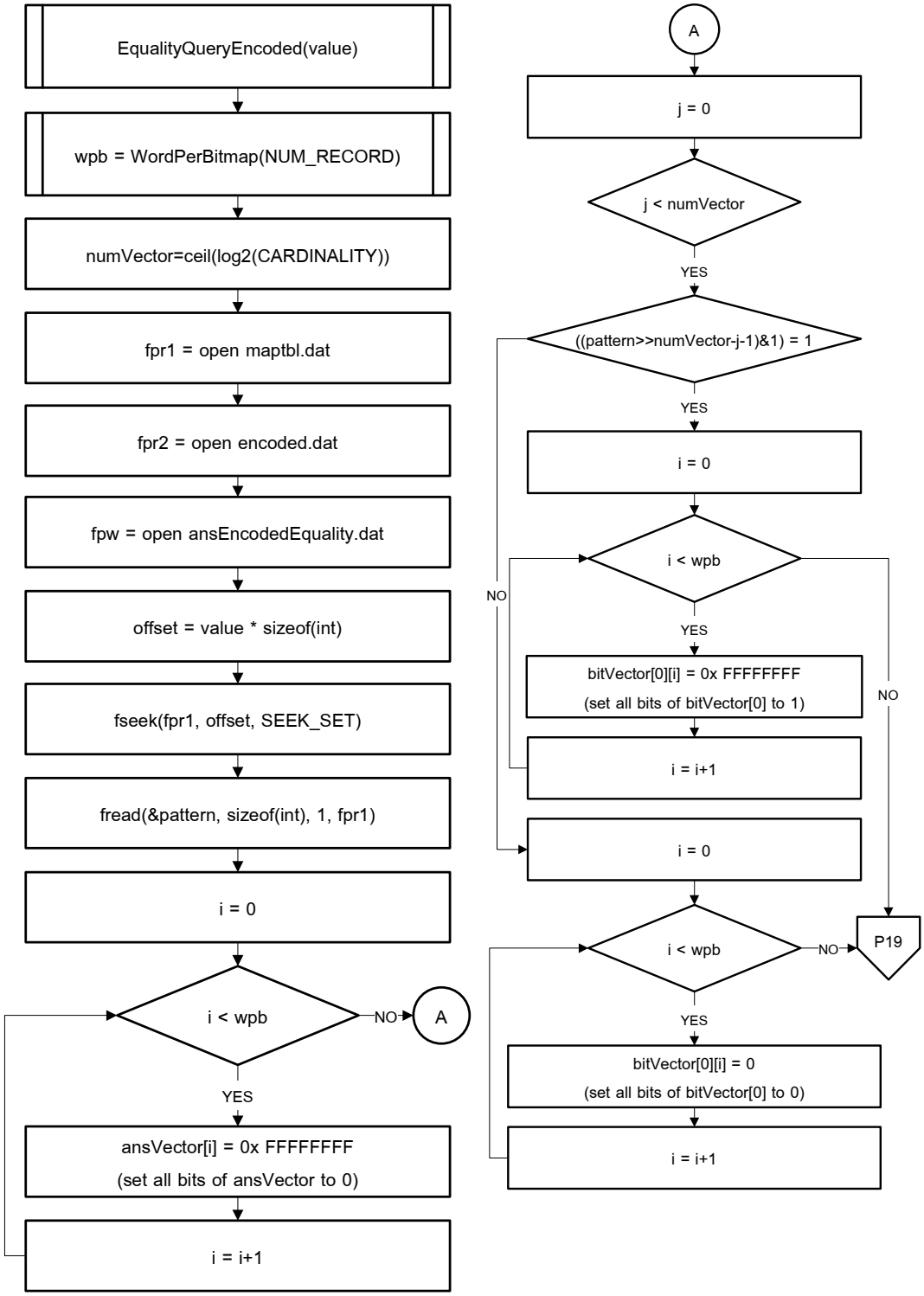

P17
i = 0
i < wpb
NO
END
YES
fwrite(&ansVector[i], sizeof(int), 1, fpw)
i = i+1

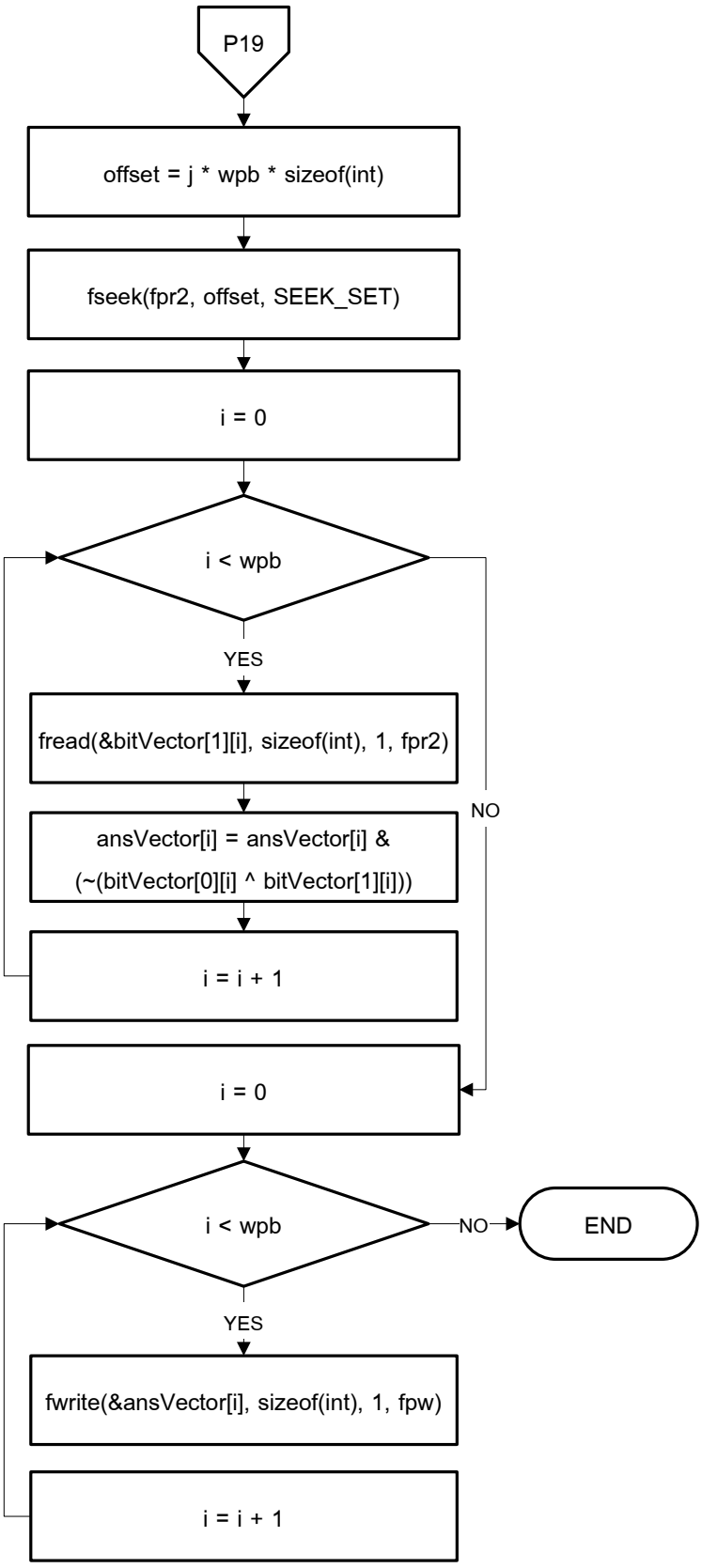
ก.5.15 ขั้นตอนวิธีการสอบถามข้อมูลแบบค่าเท่ากันบนดัชนีบิตแมปแบบคู่กัน



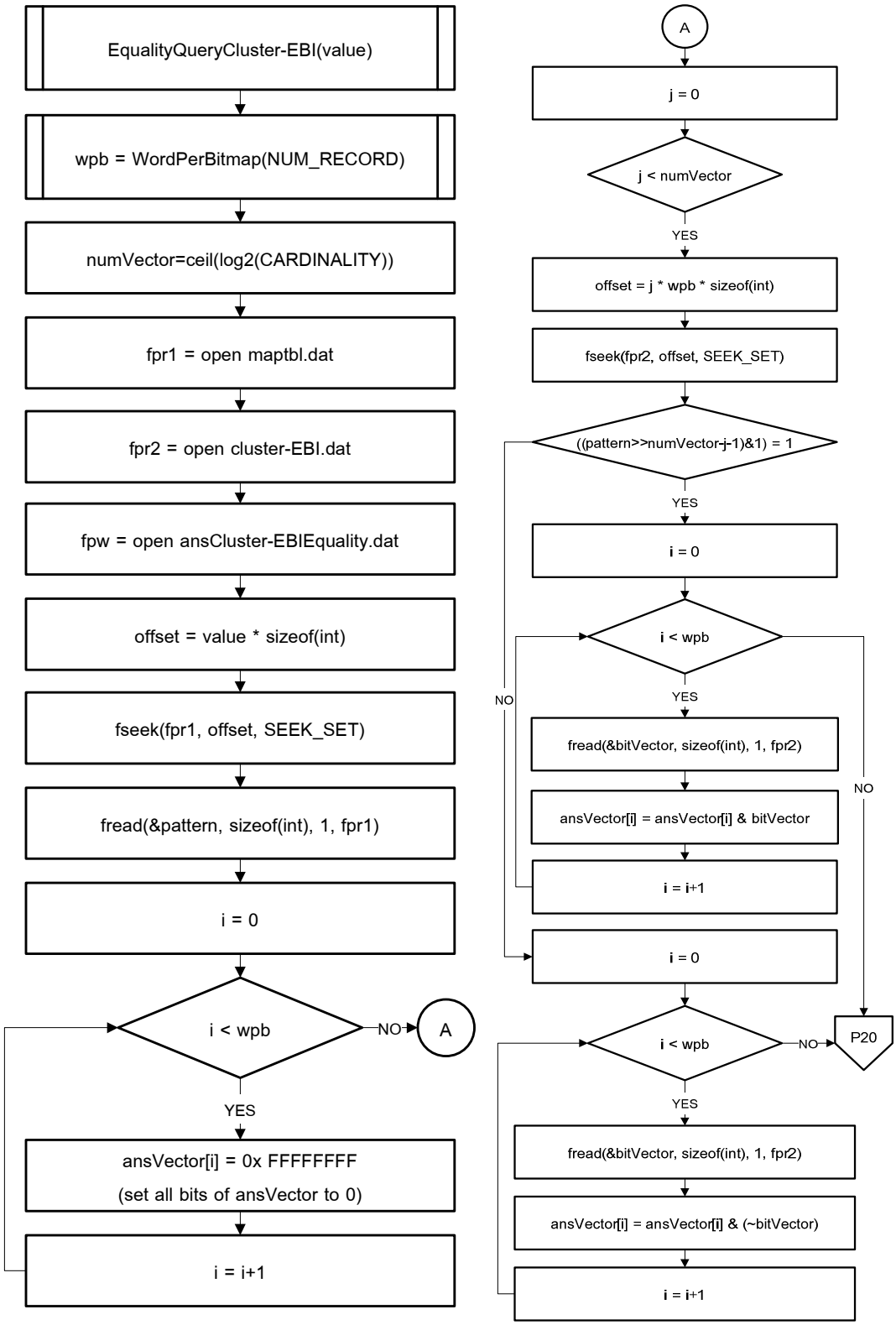

P18
i = 0
i < wpb
NO
END
YES
fwrite(&ansVector[i], sizeof(int), 1, fpw)
i = i+1

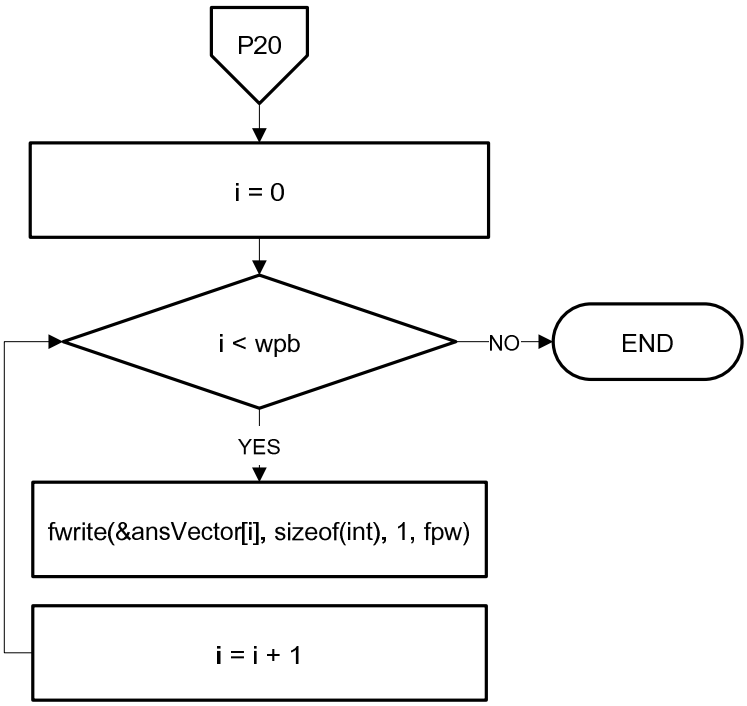
ก.5.16 ขั้นตอนวิธีการสอบถามข้อมูลแบบค่าเท่ากันบนดัชนีบิตแมปแบบเข้ารหัสทั่วไป




P19
offset = j * wpb * sizeof(int)
fseek(fpr2, offset, SEEK_SET)
i = 0
i < wpb
YES
NO
fread(&bitVector[1][i], sizeof(int), 1, fpr2)
ansVector[i] = ansVector[i] &
(~(bitVector[0][i] ^ bitVector[1][i]))
i = i + 1
i = 0
i < wpb
NO
END
YES
fwrite(&ansVector[i], sizeof(int), 1, fpw)
i = i + 1

ก.5.17 ขั้นตอนวิธีการสอบถามข้อมูลแบบค่าเท่ากันบนดัชนีบิตแมปแบบเข้ารหัสที่ใช้เทคนิคการจัดกลุ่มข้อมูล




P20
i = 0
i < wpb
NO
END
YES
fwrite(&ansVector[i], sizeof(int), 1, fpw)
i = i + 1

ภาคผนวก ข

ผลงานวิจัยที่ได้รับการตีพิมพ์

เรื่อง	Enhanced Encoded Bitmap Index for Equality Query
งานประชุมวิชาการ	2012 8th International Conference on Computing Technology and Information Management (NCM & ICNIT)
สถานที่	Grand Hilton Seoul, Seoul, Korea
วันที่	ระหว่างวันที่ 24-26 เมษายน 2555

Enhanced Encoded Bitmap Index for Equality Query

Amorntep Keawpibal, Niwan Wattanakitrungroj and Sirirut Vanichayobon
Department of Computer Science
Faculty of Science, Prince of Songkla University
Songkhla, Thailand
e-mail: p.winzip@gmail.com, niwan.w@psu.ac.th, sirirut.v@psu.ac.th

***Abstract*— Bitmap Indexes are well-known method to improve processing time for complex and interactive queries in a data warehouse. They significant improve query processing time by utilizing low-cost Boolean operations and performing predicate conditions on the index level before accessing to the primary source data. In this paper, we propose an efficient algorithm to improve equality query on traditional Encoded Bitmap Index by making use of low-cost Boolean operations. Our Comparative study shows that our proposed method is better than other existing bitmap indexing techniques for an equality query from the point of view of space-time trade-off.**

Keywords: data warehouse; bitmap indexes; encoded bitmap index; equality query

I. INTRODUCTION

A Data Warehouse (DW) is a huge data repository used for supporting a decision maker. The data stored in DW is clean, static, time variant and integrated from many different data sources including operational databases, text files, etc., over a long period of time [1]. Most of queries on DW are complex and interactive. Such complex queries could take a several minutes or hours or days to process because a large amount data in DW have to be scanned in order to answer these queries. Performance to find answers of the queries efficiently is a critical issue in DW.

Many approaches such as parallel processing, summary tables, and index [1-3] have been proposed to make query processing faster in DW. Summary tables store aggregated and summarized data for optimizing queries' performance, which is a good approach when used with predicted queries. However, when unpredicted queries are submitted, the system needs to access, retrieve and aggregate the actual data, resulting in performance decreasing. Moreover, summary tables need to update whenever source data changes and it is difficult to build all possible summary tables, choosing which ones to build is an important issue [1].

Indexing is the optimal method that improves the speed of data retrieval without additional hardware [2-4]. There are many indexing techniques such as B+ tree, Bitmap Indexes, etc. The B+ tree is the data structure which represents sorted data that allows insertion, modification, deletion and retrieval of tuples. The B+ tree is widely used in relational database environment but it cannot handle efficiently on large amount of data which causes memory overhead in complex and interactive queries. This is often queried in DW. Moreover, the data stored in DW are read-only and Bitmap Index can take benefit of this fact [1-3]. Bitmap Index is simple to represent, and can improve query processing time by applying low-cost Boolean operations such as AND, OR, NOT, etc., before accessing to the primary source data. Bitmap Index is efficient in both space and processing time for an attribute with low cardinality i.e., sex attribute (the domain of sex attribute contains only two distinct values). However, the problem is occurred if Bitmap Index is built on a high cardinality attribute which requires more space. To overcome this space problem, most of researchers on Bitmap Index are interested in reducing index size while improving query processing time on attribute with high cardinality. Two approaches have been developed: 1) Bitmap Index Compression, and 2) Bitmap Index Extension [5].

In the first approach, two well-known techniques have been studied, namely Word-Aligned Hybrid (WAH) [4,5] and Run-Length Huffman (RLH) [5]. Both techniques can reduce the size of index dramatically. However, when making a query, the techniques cannot perform a Boolean operation on index directly. They need to decompress each compressed Bitmap Index before making a query, leading to use much query processing time. In the second approach, the concept of Bitmap Index is extended in order to use less storage space while remaining efficient query processing time. In this approach, each indexed attribute value is encoded with a number of bitmap vectors. Query processing and data retrieval are supported by Boolean operation on bitmap vectors before accessing the data. The well-known techniques are Interval, Scatter, Dual, and Encoded Bitmap Index [6-9]. In this paper, we focus on the Bitmap Index Extension because the performance of this approach in term of space-time trade-off is better than the compression approach [5,10]. Among extension approach, Encoded Bitmap Index outperforms other bitmap indexes in term of space requirement [7-9]. However, its response time for equality queries is the worst. This is because, to answer an equality queries (i.e., "X=v"), it needs to spend time finding the encoded representation of the attribute value 'v' from the mapping table and comparing $\lceil log_2 C \rceil$ bitmap vectors, taking long query processing time. In this paper, we proposed a new method called Enhanced Encoded Bitmap Index to improve equality query performance of traditional Encoded Bitmap Index.

The rest of the paper is organized as follows. In Section 2 we review the five existing bitmap indexing techniques in the literature (Simple, Interval, Scatter, Dual and Encoded Bitmap Index). In Section 3 we present our algorithm to improve equality query performance of traditional Encoded Bitmap Index. In Section 4 we discuss our comparative study of five

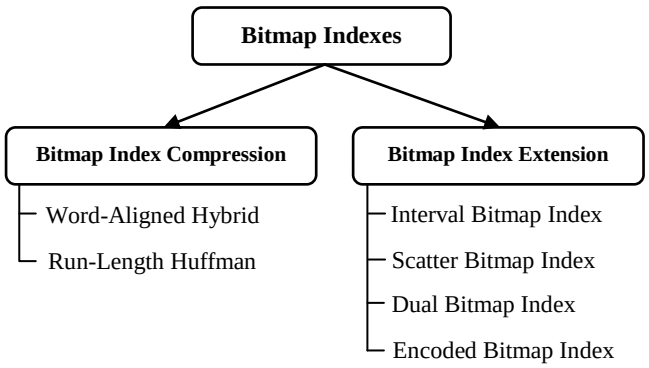


Figure 1. Bitmap Index Approaches

bitmap indexing techniques comparing with Enhance Encoded Bitmap Index. Finally, we discuss our conclusion and future work.

II. RELATED WORKS

Bitmap Index is a technique that improves query performance by using Boolean operations (AND, OR, NOT, etc.) in the selection predicate on multiple indexes and uses efficient storage space for attribute with low cardinality [1,2,6,9]. Simple Bitmap Index [3, 8-11] consists of C (the number of distinct values of an indexed attribute) bitmap vectors. Each distinct value of the indexed attribute has associated its own bitmap vector. In the bitmap vector, representing value '*v*', the i[th] bit is set to '1' if the value of the i[th] tuple on the indexed attribute equal to '*v*'. Otherwise, the bit is set to '0'. Figure 2 shows an example of Simple Bitmap Index on an indexed attribute *X* with $C = 16$. Therefore, Simple Bitmap Index on this attribute is composed of 16 bitmap vectors, says $\{S_0, S_1, S_2, S_3, \ldots S_{14}, S_{15}\}$, one for each distinct value of the attribute *X*. For example, the 1[st] bit of bitmap vector S_3 (representing values '3') is set to '0' since the value of attribute *X* of the 1[st] tuple is not equal to '3' while the 3[rd] and 7[th] bit of S_3 are set to '1' since the value of attribute *X* of the 3[rd] and 7[th] tuple are equal to '3'. To answer an equality query, a bitmap vector of the value specified in the query's condition is read into memory. Table T in Figure 2 is used as an example for the rest of the paper.

RID	…	X	…
1	…	5	…
2	…	2	…
3	…	3	…
4	…	10	…
5	…	1	…
6	…	0	…
7	…	3	…
8	…	15	…
9	…	13	…
10	…	1	…
.	.	.	.
.	.	.	.
.	.	.	.
1000	…	14	…

(a) Table T

S_0	S_1	S_2	S_3	S_4	…	S_{14}	S_{15}
0	0	0	0	0	…	0	0
0	0	1	0	0	…	0	0
0	0	0	1	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	1	0	…	0	0
0	0	0	0	0	…	0	1
0	0	0	0	0	…	0	0
0	1	0	0	0	…	0	0
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
0	0	0	0	0	…	1	0

(b) Simple Bitmap Index

Figure 2. Example of Simple Bitmap Index

However, the problem occurs if Simple Bitmap Index is built on an indexed attribute with high cardinality, requiring more storage space. To overcome this index size's problem, various Bitmap Index techniques have been proposed in order to reduce storage space requirement as well as improve query performance [4-9].

An Interval Bitmap Index [6] on an indexed attribute *X* consists of $\lceil C/2 \rceil$ bitmap vectors, $\{I_0, I_1, \ldots, I_{\lceil C/2 \rceil - 1}\}$, where $I_j = [j, j+m]$ and $m = \lfloor C/2 \rfloor - 1$. An example of Interval Bitmap Index on an indexed attribute *X* with $C = 16$ is shown in Figure 3(a). The Interval Bitmap Index on this attribute contains 8 bitmap vectors, $\{I_0, I_1, I_2, I_3, \ldots, I_7\}$. To answer an equality query, the following formula is used.

$$"X = v" = \begin{cases} I_0 & \text{if } v = 0, m = 0 \\ \overline{I_0} & \text{if } v = 1, C = 2 \\ I_1 & \text{if } v = 1, C = 3 \\ I_v \wedge \overline{I_{v+1}} & \text{if } v < m \\ I_v \wedge I_0 & \text{if } v = m, m > 0 \\ I_{v-m} \wedge \overline{I_{v-m-1}} & \text{if } m < v < C\text{-}1, m > 0 \\ \overline{I_{\lfloor C/2 \rfloor - 1} \vee I_0} & \text{if } v = C\text{-}1 \end{cases}$$

Scatter Bitmap Index [7] on an indexed attribute X consists of $\lceil 2\sqrt{C} \rceil$ bitmap vectors. The Scatter Bitmap Index is split into two groups of bitmap vectors (i.e., Z-group and L-group). Z-group contains $\left\lceil \frac{C}{m-1} \right\rceil + 1$ bitmap vectors and L-group contains $\lceil m - 2 \rceil$ bitmap vectors, where $m = \lceil \sqrt{C} \rceil + 1$. Two bitmap vectors represent one distinct value of the indexed attribute. Building Z- and L-bitmap vectors of the Scatter Bitmap Index is described below.

Z_0 bitmap vectors' Creation

1. If the i[th] tuple contains value '0' then the i[th] bit of Z_0 is set to '1'. Otherwise, it is set to '0'.

Z- and L-bitmap vectors' Creation

For each value '*v*' of the i[th] tuple on an indexed attribute

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

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

1. If (k = 0) then the i[th] bit of Z_j and Z_{j-1} is set to '1'.
   else the i[th] bit of Z_j and L_k is set to '1'.
2. Otherwise, it is set to '0'.

Figure 3(b) shows an example of the Scatter Bitmap Index on an indexed attribute *X* with $C = 16$. There are 8 bitmap vectors (i.e., $\{Z_0, Z_1, Z_2, Z_3, Z_4, L_1, L_2, L_3\}$). To answer an equality query, the following formula is calculated.

$$"X = v" = \begin{cases} Z_{\frac{v}{(m-1)}} \wedge Z_{\frac{v}{(m-1)}+1} & \text{if } v \bmod (m\text{-}1) = 0 \\ Z_{\frac{v}{(m-1)}+1} \wedge L_{v \bmod (m-1)} & \text{otherwise} \end{cases}$$

I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0
0	0	1	1	1	1	1	1
0	0	0	0	0	1	1	1
0	0	0	0	1	1	1	1
1	1	1	1	1	0	0	0
0	0	0	0	0	0	1	1
0	0	0	0	0	0	0	1
0	0	0	0	1	1	1	1
1	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0
0	0	0	0	0	0	1	1
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
1	0	0	0	0	0	0	0

(a) Interval Bitmap Index

Z_4	Z_3	Z_2	Z_1	Z_0	L_3	L_2	L_1
0	0	1	0	0	0	0	1
0	0	0	1	0	0	1	0
0	0	0	1	0	1	0	0
0	1	0	0	0	0	1	0
0	0	0	1	0	0	0	1
0	0	0	1	1	0	0	0
0	0	0	1	0	1	0	0
1	0	0	0	0	1	0	0
1	0	0	0	0	0	0	1
0	0	0	1	0	0	0	1
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
1	0	0	0	0	0	1	0

(b) Scatter Bitmap Index

D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	0	0	1
1	0	0	1	0	0	0
1	0	0	0	1	0	0
0	1	0	0	0	0	1
1	0	1	0	0	0	0
1	1	0	0	0	0	0
1	0	0	0	1	0	0
0	0	0	1	1	0	0
0	0	1	0	0	1	0
1	0	1	0	0	0	0
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
0	1	0	0	0	0	1

(c) Dual Bitmap Index

E_3	E_2	E_1	E_0
0	0	1	1
0	1	0	0
0	1	0	1
1	1	1	1
0	0	0	1
0	0	0	0
0	1	0	1
1	0	0	1
1	0	1	0
0	0	0	1
.	.	.	.
.	.	.	.
.	.	.	.
1	0	1	1

Mapping table	
0	0000
1	0001
2	0100
3	0101
4	0010
5	0011
6	0110
7	0111
8	1100
9	1101
10	1111
11	1110
12	1000
13	1010
14	1011
15	1001

(d) Encoded Bitmap Index and Mapping Table [9]

Figure 3. Exmaple of Four Bitmap Indexes

Like Scatter Bitmap Index, Dual Bitmap Index [8] uses two bitmap vectors to represent each attribute value. It consists of $\lceil\sqrt{2C+0.25}+0.5\rceil$ bitmap vectors, $\{D_0, D_1, D_2, D_3, \ldots, D_{\lceil\sqrt{2C+0.25}+0.5\rceil-1}\}$. Each i[th] bit of D_j is set to '1' if $j = r$ or $j = s$, where

$$r = \left\lceil\sqrt{2(hiC - v) + 0.24} - 0.5\right\rceil$$
$$s = \left\lceil r - 1 - \left\lfloor\left(v - \frac{(n-r)(n-r-1)}{2}\right) mod\ r\right\rfloor\right\rceil$$
$$hiC = \binom{n}{2}$$

and n is number of bitmap vectors.

In Figure 3(c) shows an example of Dual Bitmap Index on an indexed attribute X with C = 16. The Dual Bitmap Index includes 7 bitmap vectors, says $\{D_0, D_1, D_2, \ldots, D_6\}$. To answer a query, the following formula is used.

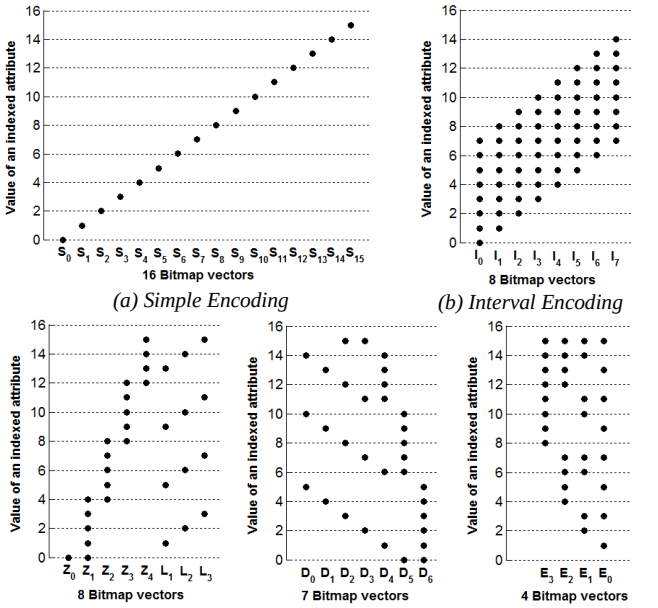
$$"X = v" = D^r \wedge D^s$$

Wu and Buchmann have been introduced Encoded Bitmap Index [9]. For an indexed attribute, the index contains $\lceil log_2 C\rceil$ bitmap vectors $(E_{\lceil log_2 C\rceil-1}, \ldots, E_1, E_0)$ and a mapping table. Each distinct value of the indexed attribute is represented with $\lceil log_2 C\rceil$ bits. Figure 3(d) shows an example of the Index on an indexed attribute X with C = 16, which is composed of 4 bitmap vectors, $\{E_3, E_2, E_1, E_0\}$ and one mapping table. For example, value '0' is encoded as '0000' (for tuples with "$X=0$". We set corresponding positions in all bitmap vectors as $E_3=0$, $E_2=0$, $E_1=0$, and $E_0=0$). To answer an equality query, encoded representations of the value in a predicate condition query is retrieved from the mapping table, and then compared with all $\lceil log_2 C\rceil$ bitmap vectors of each tuple. The result is the tuple that has the same bit pattern as the encoded representation. For example, using a mapping table in Figure 3(d), if we have the condition in a query as "$X=3$". From the mapping table, '3' is encoded as '0101' ($E_3=0$, $E_2=1$, $E_1=0$ and $E_0=1$). Then, each tuple (having 4 bits) of bitmap vectors are checked to see whether it has $E_3=0$, $E_2=1$, $E_1=0$, and $E_0=1$ or not. For the above example, an equality query could take a long time to process because the equality query on Encoded Bitmap Index needs to compare $\lceil log_2 C\rceil$ bits per tuple.

III. Improving Equality Query Performance

To index an attribute, Encoded Bitmap Index uses $\lceil log_2 C\rceil$ bitmap vectors while Simple Bitmap Index uses C bitmap vectors, Interval Bitmap Index uses $\lceil C/2\rceil$ bitmap vectors, Scatter Bitmap Index uses $\lceil 2\sqrt{C}\rceil$ bitmap vectors, and Dual Bitmap Index uses $\lceil\sqrt{2C+0.25}+0.5\rceil$ bitmap vectors (see Figure 4(a-e)). The main disadvantage of these Bitmap Indexes, especially Simple Bitmap Index, is that space usage performance degrades as the cardinality in an indexed attribute grows. Although the Encoded Bitmap Index can utilize space requirement efficiently, the complexity of an equality query retrieval functions is undesirable.

In this section we propose our indexing method, called Enhanced Encoded Bitmap Index (E-EBI) [10], to improve an equality query performance of traditional Encoded Bitmap Index. The method makes use of the benefit of low-cost Boolean operations. Low-cost Boolean operations are very fast and basic, directly supported by processor [6].



(a) Simple Encoding *(b) Interval Encoding*

(c) Scatter Encoding *(d) Dual Encoding* *(e) Encoded Encoding*

Figure 4. Encoding Scheme of Five Bitmap Indexes with C=16
(● represents bit equal to '1')

A. The Creation of E-EBI

An Enhanced Encoded Bitmap Index [10] on an indexed attribute X consists of $\lceil log_2 C \rceil$ bitmap vectors ($E_{\lceil log_2 C \rceil - 1}, \ldots, E_1, E_0$). Each distinct value of the indexed attribute is numbered in ascending order and represented with binary encoding on $\lceil log_2 C \rceil$ bits. Figure 6 shows E-EBI on an indexed attribute X with C = 16, which is composed of 4 bitmap vectors, $\{E_3, E_2, E_1, E_0\}$. From binary encoding, for example, those tuples with "*X=0*", "*X=1*", and "*X=2*" are encoded as '0000', '0001' and '0010', respectively.

B. Equality Queries

The E-EBI algorithm for answering an equality query is shown in Figure 5 and Table I describes notations used in the algorithm. The algorithm consists of eight steps. In the first step, all bits of the result bitmap vector (i.e., ANS_VECTOR), are set to '1'. In the second step, the encoded bits' representation of the value specified in the query condition (i.e., PATTERN) is calculated from the binary encoding. In 3rd-7th step, each ith bitmap vector is performed Boolean operation AND with ANS_VECTOR if E_i is equal to '1'. Otherwise, each bit in the ith bitmap vector is inverted before performing Boolean operation AND with ANS_VECTOR. In the last step, ANS_VECTOR is output as the final result. For example, if selection condition is "*X=3*". From the binary encoding, '3' is represented as '0011'. Therefore, E_3 and E_2 need to be inverted before performing Boolean operation AND with ANS_VECTOR, while E_1 and E_0 can be performed Boolean operation AND with ANS_VECTOR directly. The result of the above equality query is the 3rd and 7th tuples as shown in Figure 6(c).

TABLE I. NOTATION USED IN E-EBI QUERY PROCESSING ALGORITHM

Symbol	Description
C	The number of distinct values of the indexed attribute (i.e., cardinality).
N	The total number of tuples in a table.
QUERY_VALUE	A value specified in the query condition.
ANS_VECTOR	The result bitmap vector, sized *N* bits.
PATTERN	The binary encoding represented *QUERY_VALUE*, sized $\lceil log_2 C \rceil$ bits.

E-EBI Query Processing Algorithm

1. Initialize all bits in *ANS_VECTOR* to be '1'.
2. Find *PATTERN* using the binary encoding.
3. for i = 0 to $\lceil log_2 C \rceil$-1
4. if ($E_i = 1$) then
5. *ANS_VECTOR* = $ANS\_VECTOR \wedge E_i$
6. else
7. *ANS_VECTOR* = $ANS\_VECTOR \wedge \overline{E_i}$
8. Output *ANS_VECTOR*

Figure 5. E-EBI Query Processing Algorithm

RID	…	X	…
1	…	5	…
2	…	2	…
3	…	3	…
4	…	10	…
5	…	1	…
6	…	0	…
7	…	3	…
8	…	15	…
9	…	13	…
10	…	1	…
.	.	.	.
.	.	.	.
.	.	.	.
1000	…	14	…

(a) Table T

E_3	E_2	E_1	E_0
0	1	0	1
0	0	1	0
0	0	1	1
1	0	1	0
0	0	0	1
0	0	0	0
0	0	1	1
1	1	1	1
1	1	0	1
0	0	0	1
.	.	.	.
.	.	.	.
.	.	.	.
1	1	1	0

(b) E-EBI

$\overline{E_3}$	$\overline{E_2}$	E_1	E_0
1	0	0	1
1	1	1	0
1	1	1	1
0	1	1	0
1	1	0	1
1	1	0	0
1	1	1	1
0	0	1	1
0	0	0	1
1	1	0	1
.	.	.	.
.	.	.	.
.	.	.	.
0	0	1	0

(c) The result of query "X=3"

ANS_VECTOR
0
0
1
0
0
0
1
0
0
0
.
.
.
0

Figure 6. Example of E-EBI and its query result

IV. PERFORMANCE STUDY

This section presents experimental results by comparing the space requirement and query response time of six Bitmap Indexes (Simple, Interval, Scatter, Dual, traditional Encoded Bitmap Index and E-EBI) for evaluating equality queries on an indexed attribute. The experiments were conducted on a 2.00 GHz Intel Core2Duo with 3.00 GB main memory, using Windows 7 as an operating system. The experiments were performed on data set from TPC-H [11].

Table II shows a theoretical analysis of six Bitmap Indexes. Simple Bitmap Index is the worst space requirement (uses C bitmap vectors), while E-EBI is the best space requirement (uses $\lceil log_2 C \rceil$ bitmap vectors). Traditional Encoded Bitmap Index uses $\lceil log_2 C \rceil$ bitmap vectors and one mapping table while Interval, Scatter and Dual Bitmap Index use $\lceil C/2 \rceil$, $\lceil 2\sqrt{C} \rceil$ and $\lceil \sqrt{2C + 0.25} + 0.5 \rceil$ bitmap vectors, respectively.

Among six Bitmap Indexes, Simple Bitmap Index takes one bitmap vector and no Boolean operation when answer an equality query. Interval, Scatter and Dual Bitmap Index take two bitmap vectors and perform 2, 1, and 1 Boolean operations, respectively. Both traditional Encoded Bitmap Index and E-EBI use $\lceil log_2 C \rceil$ bitmap vectors. However, in order to answer equality query, for each tuple, the first method must find the bitmap encoding from the mapping table and then make comparison all bitmap vectors but the latter method only performs not more than $\lceil log_2 C \rceil$ low-cost Boolean operations.

Figure 7-9 highlights the above theoretical analysis. Scatter, Dual, traditional Encoded Bitmap Index and E-EBI require less space than Simple and Interval Bitmap Index as shown in Figure 7. In Figure 8 shows that E-EBI and traditional Encoded Bitmap Index use less space than Scatter and Dual Bitmap Index. However, E-EBI requires the least amount of space usage while traditional Encoded Bitmap Index uses a little more space than E-EBI due to the mapping table. Figure 9 illustrates the result of space vs. query response time of six Bitmap Indexes (Simple, Interval, Scatter, Dual, traditional Encoded Bitmap Index and E-EBI) for evaluating equality queries on the indexed attribute with cardinality 150. Among six Bitmap Indexes, traditional Encoded Bitmap Index and E-EBI outperform all other bitmap indexes in term of space requirement. Simple Bitmap Index spends a lot of time because

it performs one bitmap vector but requires maximum storage space. Interval, Scatter and Dual Bitmap Indexes take query processing time slower than Simple Bitmap Index because they perform two bitmap vectors and one low-cost Boolean operation but require storage space less than Simple Bitmap Index. Especially, Dual Bitmap Index requires space storage less than Interval and Scatter Bitmap Index, respectively. The traditional Encoded Bitmap Index takes query processing time slower than Interval, Scatter, Dual Bitmap Index and E-EBI. Moreover, by calculate each rectangular area in Figure 9, our experimental results confirm that E-EBI is better than other five Bitmap Indexes in term of space-time trade-off.

V. CONCLUSIONS

Complex and interactive queries are often in data warehouse environment. They spend a long time to access and retrieve to find answer for the queries. Various bitmap indexes focus on finding answer to these queries because they perform fast Boolean operations on the indexed level before accessing data source. Several bitmap indexes have been introduced, aiming to reduce space requirement and improve query processing time. In this paper, we proposed E-EBI, a new method to improve equality query performance for Encoded Bitmap Index. It uses the least number of bitmap vectors and query time by using the low-cost Boolean operations to answer equality query. Our comparative study shows that E-EBI can improve performance to traditional Encoded Bitmap Index and it outperforms other bitmap indexed for equality query from the point of space vs. query time.

For future work, a well-defined encoding representation of each distinct value of an indexed attribute is important point to optimize other types of queries' performance i.e., membership query. Thus we plan to apply data mining techniques, for example, Clustering and Association Rules. These techniques could be used to group values' attribute that are frequently queried together then we can use this discovery to build a good encoding scheme to reduce numbers of bitmap vectors scanned.

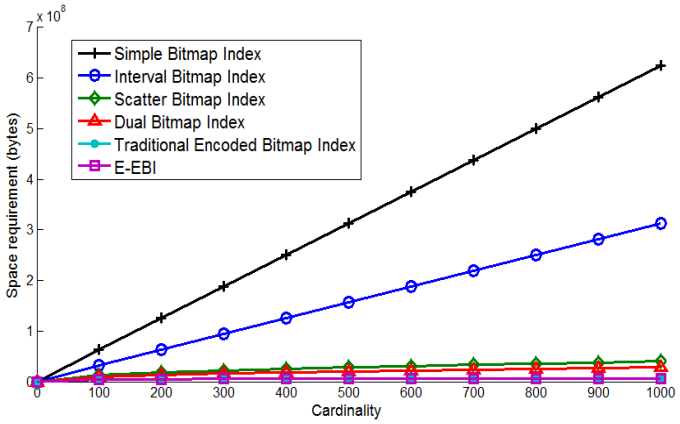


Figure 7. Comparison of Space requirement vs. Cardinality of Six Bitmap Indexing Techniques (N=5,000,000)

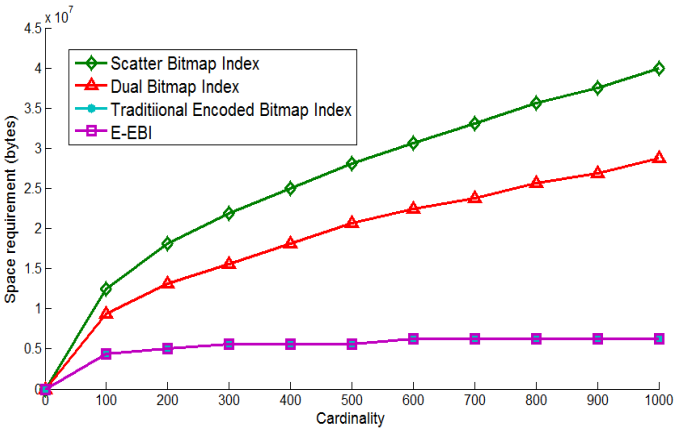


Figure 8. Comparison of Space Requirement vs. Cardinality of Scatter, Dual, Traditional Encoded Bitmap Index, and E-EBI (N=5,000,000)

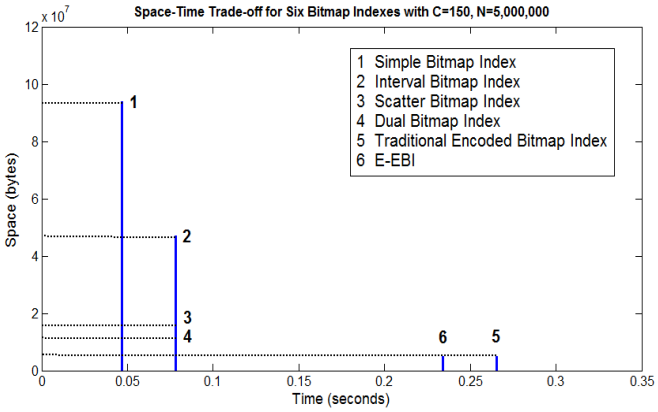


Figure 9. Space vs. Time of Six Bitmap Indexing Techniques

REFERENCES

[1] S. Chaudhuri and U. Dayal, "An Overview of Data Warehousing and OLAP Technology", ACM SIGMOD RECORD, vol. 26, pp. 65-74, 1997.

[2] P. O'Neil and D. Quass, "Improved Query Performance with Variant Indexes", Proceeding of the 1997 ACM SIGMOD International conference on Management of data, pp.38-49, 1997.

[3] C. Y. Chan and Y. E. Ioannidis, "Bitmap Index Design and Evaluation", Proceeding of the 1998 ACM SIGMOD International Conference on Management data, pp. 355-366, 1988.

[4] K. Wu, E. J. Otoo and A. Shoshani, "Optimizing bitmap indices with efficient compression", ACM Transaction on Database Systems (TODS), vol. 31, pp. 1-38, 2006.

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

[6] C. Yong Chan and Y. E. Ioannidis, "An Efficient Bitmap Encoding Scheme for Selection Queries", Proceeding of the 1999 ACM SIGMOD International Conference on Management of data, pp. 215-226, 1999.

[7] S. Vanichayobon, J. Manfuekphan and L. Gruenwald, "Scatter Bitmap: Space-time Efficient Bitmap Indexing for Equality and Membership Queries", Proceeding of IEEE International Conference on cybernetics and Intelligent Systems, pp. 1-6, 2006.

[8] N. Wattanakitrungroj and S. Vanichayobon, "Dual Bitmap Index: Space-Time Efficient Bitmap Index for Equality and Membership Queries", Proceeding of International Symposium on Communications and Information Technologies (ISCIT'06), pp. 568-573, 2006.

[9] M. C. Wu and A. P. Bucgmann, "Encoded Bitmap Indexing for Data Warehouses", Proceeding of 14th International Conference on Data Engineering, pp. 220-230, 1998.

[10] A. Keawpibal and S. Vanichayobon, "Improved Query Performance to Encoded Bitmap Index using Data Clustering", Department of Computer Science, Faculty of Science, Prince of Songkla University, Thailand, 2011

[11] Transaction Processing Performance Council (TPC), "TPC-H: An Ad-hoc Decidion Support Benchmark", Version 2.14.1, August 11, 2010, http://www.tpc.org/tpch/.

TABLE II. A COMPARATIVE STUDY OF SIX BITMAP INDEXES

Bitmap Indexes	**Space**	**Query response time**	
	Number of bitmap vectors	*Number of bimap vectors scanned*	*Operations*
Simple [2]	C	1	-
Interval [6]	$\lceil C/2 \rceil$	2	1 AND, 1 NOT
Scatter [7]	$\lceil 2\sqrt{C} \rceil$	2	1 AND
Dual [8]	$\lceil \sqrt{2C + 0.25} + 0.5 \rceil$	2	1 AND
Traditional Encoded [9]	$\lceil log_2 C \rceil$ + mapping table	$\lceil log_2 C \rceil$	Bit comparisons
E-EBI [10]	$\lceil log_2 C \rceil$	$\lceil log_2 C \rceil$	$\lceil log_2 C \rceil$ AND, $\lceil log_2 C \rceil$ NOT

C is Cardinality.

ประวัติผู้เขียน

ชื่อ สกุล นายอมรเทพ แก้วภิบาล

รหัสประจำตัวนักศึกษา 5310220126

วุฒิการศึกษา

วุฒิ	ชื่อสถาบัน	ปีที่สำเร็จการศึกษา
วท.บ. (วิทยาการคอมพิวเตอร์)	มหาวิทยาลัยสงขลานครินทร์	2552

ทุนการศึกษา (ที่ได้รับในระหว่างการศึกษา)

ทุนผู้ช่วยวิจัยคณะวิทยาศาสตร์ (RA) มหาวิทยาลัยสงขลานครินทร์ วิทยาเขตหาดใหญ่ ประจำปีการศึกษา 2553

การตีพิมพ์เผยแพร่ผลงาน

A. Keawpibal, N. Wattanakitrungroj and S. Vanichayobon, "Enhanced Encoded Bitmap Index for Equality Query", 2012 8th International Conference on Computing Technology and Information Management (NCM & ICNIT), 2012.