บทที่ 3

ดัชนีแบบบิตแมป (Bitmap Indexing)

สำหรับบทนี้ จะเป็นการศึกษาและวิเคราะห์เทคนิคของดัชนีบิตแมปเดิมที่มีอยู่ การทำดัชนีที่อยู่บนพื้นฐานของบิตแมป ได้แก่ ดัชนีบิตแมปแบบพื้นฐาน ดัชนีบิตแมปแบบ Range ดัชนีบิตแมปแบบช่วง และดัชนีบิตแมปแบบเข้ารหัส

จากบทที่ 2 ที่กล่าวมา ดัชนีที่ได้รับความนิยมในการใช้งานกับระบบฐานข้อมูลดำเนินการ คือ ดัชนีแบบต้นไม้ ซึ่งดัชนีแบบนี้ก็มีใช้ในคลังข้อมูล แต่ในกรณีที่แอทริบิวต์ที่นำมาทำดัชนีในคลังข้อมูลมีคาร์ดินอลิตี้ (คือ จำนวนค่าที่เป็นไปได้ของแอทริบิวต์ที่นำมาทำดัชนี) ต่ำ ก็จะใช้ดัชนีแบบบิตแมปแทนดัชนีแบบต้นไม้ เพื่อใช้ประโยชน์ของการดำเนินการตรรกะระดับบิต โดยเฉพาะอย่างยิ่งลักษณะการสอบถามข้อมูลที่ใช้กับระบบคลังข้อมูล มักจะเป็นการสอบถามข้อมูลแบบซับซ้อน และคลังข้อมูลมักจะเป็นข้อมูลที่มีคาร์ดินอลิตี้ต่ำ

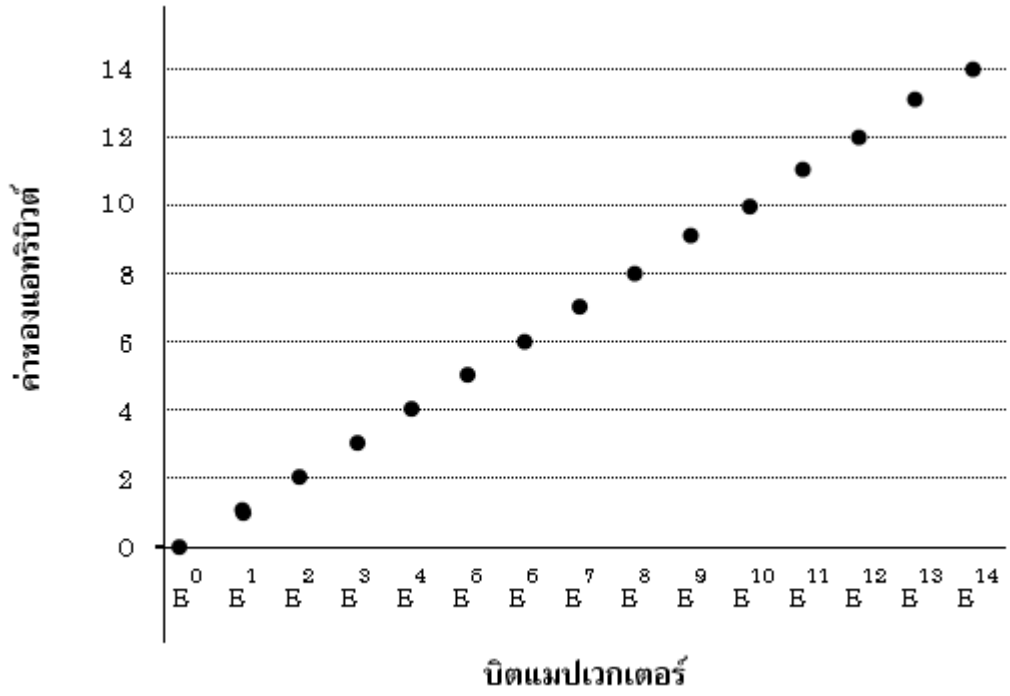
ดัชนีแบบบิตแมป เป็นดัชนีที่ได้รับความนิยมในการใช้งานกับระบบคลังข้อมูลในกรณีที่มีคาร์ดินอลิตี้ต่ำ ๆ เช่น แอทริบิวต์เพศ ก็จะมีค่าที่เป็นไปได้ 2 ค่า คือ เพศหญิง และเพศชาย ดังนั้นค่าคาร์ดินอลิตี้ของแอทริบิวต์เพศ ก็จะเท่ากับ 2 หรือ แอทริบิวต์จังหวัดในประเทศไทย ก็จะมีค่าที่เป็นไปได้ 76 ค่า ดังนั้นค่าคาร์ดินอลิตี้ของแอทริบิวต์จังหวัดในประเทศไทย ก็จะเท่ากับ 76 เป็นต้น ตัวอย่างของดัชนีที่อยู่บนพื้นฐานของบิตแมป เช่น ดัชนีบิตแมปแบบพื้นฐาน (Pure/Simple Bitmap Indexing) ดัชนีบิตแมปแบบ Range (Range Bitmap Indexing) ดัชนีบิตแมปแบบช่วง (Interval Bitmap Indexing) [30] และดัชนีบิตแมปแบบเข้ารหัส (Encoded Bitmap Indexing) [10]

3.1 ดัชนีบิตแมปแบบพื้นฐาน (Pure/Simple Bitmap Indexing)

ดัชนีบิตแมปแบบพื้นฐาน [30] ประกอบด้วย C บิตแมปเวกเตอร์ ได้แก่ บิตแมปเวกเตอร์ E^0, E^1, E^2, ..., E^i, ..., E^{C-1} โดย E^i คือ บิตแมปเวกเตอร์ที่มีค่าของแอทริบิวต์เท่ากับ i เช่น E^2 คือ บิตแมปเวกเตอร์ที่มีค่าของแอทริบิวต์ เท่ากับ 2 ซึ่งตำแหน่งบิตในบิตแมปเวกเตอร์จะมีการเทียบค่า (Map) ไปยังหมายเลขเรคอร์ด (RIDs) ของตารางเชิงความสัมพันธ์ [29] โดยจะมีฟังก์ชันการเทียบค่า แปลงตำแหน่งบิตไปเป็นหมายเลขเรคอร์ดจริง [11, 29] กล่าวคือ ในตารางดัชนีตำแหน่งบิตที่ i ของบิตแมปเวกเตอร์ใดที่มีค่าเป็น 1 คือ ค่าของแอทริบิวต์ที่ถูกทำดัชนีหมายเลขเรคอร์ดที่ i ในตารางเชิงความสัมพันธ์

ตัวอย่างเช่น เลือกเฉพาะแอทริบิวต์ A จากตารางเชิงความสัมพันธ์มาทำดัชนี โดยสมมติให้แอทริบิวต์ A มีคาร์ดินอลิตี้ เท่ากับ 15 (ซึ่งจะใช้เป็นกรณีตัวอย่างในทุก ๆ ดัชนี

ของเอกสารนี้) ดังนั้นเราจะต้องสร้างบิตแมปเวกเตอร์แบบพื้นฐาน 15 บิตแมปเวกเตอร์ โดยแต่ละบิตแมปเวกเตอร์จะแทนค่าที่เป็นไปได้แต่ละค่าของแอทริบิวต์ A ดังภาพประกอบ 3-1 และ 3-2



ภาพประกอบ 3-1 ตัวอย่างแผนภาพการทำดัชนีบิตแมปแบบพื้นฐานบนแอทริบิวต์ A (มีค่า C = 15) ใช้ 15 บิตแมปเวกเตอร์

จากภาพประกอบ 3-1 แอทริบิวต์ A มีค่าที่เป็นไปได้ คือ 0, 1, 2, ..., 14 ดังนั้นดัชนีบิตแมปแบบพื้นฐาน ประกอบด้วย 15 บิตแมปเวกเตอร์ คือ E^0, E^1, E^2, ..., E^{14} โดยที่แต่ละบิตแมปเวกเตอร์ แทนค่าแต่ละค่าของแอทริบิวต์ A ตัวอย่างเช่น บิตแมปเวกเตอร์ E^2 แทนค่าของแอทริบิวต์ A ที่มีค่าเท่ากับ 2 ดังนั้นในบิตตำแหน่งที่ 4 (RID ที่ 4 ในตารางเชิงความสัมพันธ์) ของบิตแมปเวกเตอร์ E^2 จะมีค่าเท่ากับ 1 ส่วนบิตตำแหน่งที่ 4 ของบิตแมปเวกเตอร์อื่น ๆ จะมีค่าเท่ากับ 0 ดังแสดงในภาพประกอบ 3-2(b)

ตัวอย่าง 3-1 กำหนดให้แอทริบิวต์ A มีค่าที่เป็นไปได้ 15 ค่า คือ 0 - 14 ดังภาพประกอบ 3-2

RID	แอทริบิวต์ A
1	14
2	3
3	4
4	2
5	3
6	1
7	13
8	0
9	6
10	5

(a) เลือกเฉพาะแอทริบิวต์ A

E^{14}	E^{13}	...	E^5	E^4	E^3	E^2	E^1	E^0
1	0	...	0	0	0	**0**	0	0
0	0	...	0	0	1	**0**	0	0
0	0	...	0	1	0	**0**	0	0
0	0	...	0	0	0	**1**	0	0
0	0	...	0	0	1	**0**	0	0
0	0	...	0	0	0	**0**	1	0
0	1	...	0	0	0	**0**	0	0
0	0	...	0	0	0	**0**	0	1
0	0	...	0	0	0	**0**	0	0
0	0	...	1	0	0	**0**	0	0

(b) ดัชนีบิตแมปแบบพื้นฐาน บนแอทริบิวต์ A

ภาพประกอบ 3-2 ดัชนีบิตแมปแบบพื้นฐานบนแอทริบิวต์ A ใช้ 15 บิตแมปเวกเตอร์

3.1.1 การค้นหาข้อมูลแบบค่าเท่ากัน

มีรูปแบบ "A=v" โดย v คือ ค่าของข้อมูลที่จะค้นหา สามารถค้นหาโดยการต้องการค่าใด ก็อ่านค่าบิตแมปเวกเตอร์นั้น ๆ $\left(E^{v}\right)$

ตัวอย่างเช่น จากภาพประกอบ 3-2 ถ้าต้องการค้นหา A = 2
<u>วิธีการดึงข้อมูล</u> โดยการอ่านบิตแมปเวกเตอร์ E^2 ซึ่งคำตอบที่ได้คือ บิตแมปเวกเตอร์ E^2 ที่มีค่าบิตเท่ากับ 1 (ON) ในที่นี้คือ เรคอร์ดที่ 4

3.1.2 การค้นหาข้อมูลแบบความเป็นสมาชิก

มีรูปแบบ "$A \in \{ v_1, v_2, \ldots, v_k \}$" ซึ่งคือการใช้การค้นหาแบบค่าเท่ากันที่ไม่ต่อเนื่องกันหลาย ๆ ค่าในเวลาเดียวกัน สามารถค้นหาโดยการอ่านค่าบิตแมปเวกเตอร์นั้น ๆ $\left(E^{v}\right)$ มาดำเนินการตรรกะ OR กันระหว่างบิตแมปเวกเตอร์

ตัวอย่างเช่น จากภาพประกอบ 3-2 ถ้าต้องการค้นหา

1) A = 1, 4 และ 6

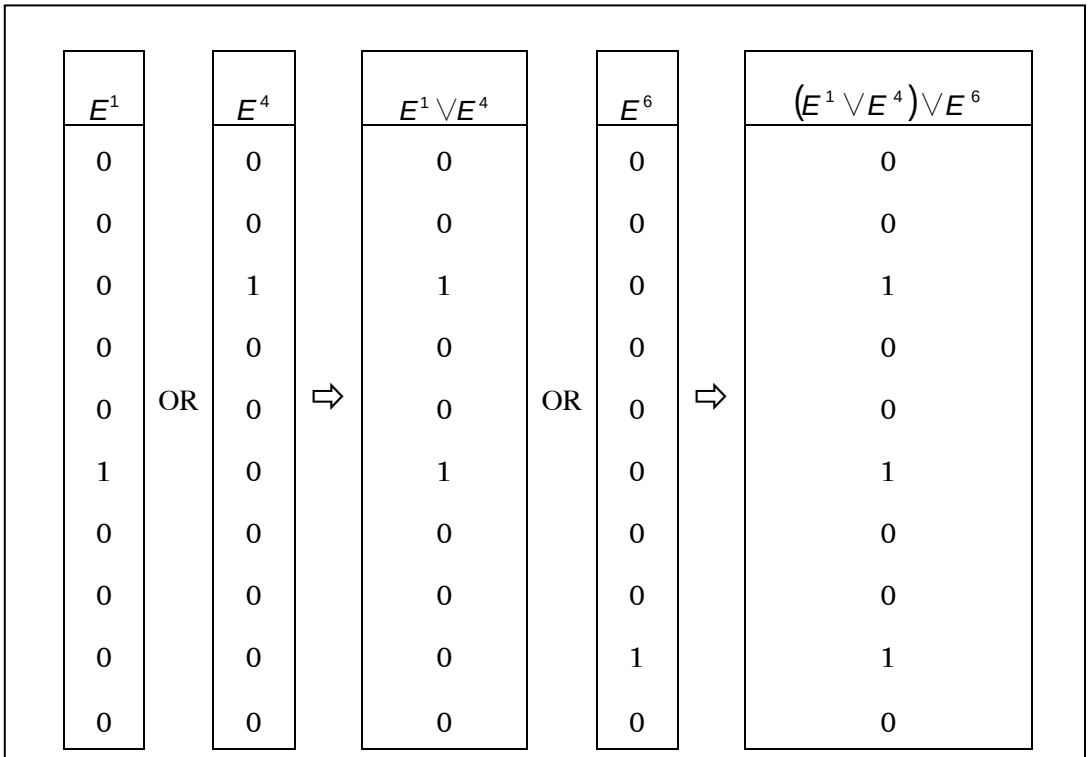
<u>วิธีการดึงข้อมูล</u>

1.1) อ่านบิตแมปเวกเตอร์ E^1 และ E^4 ดังภาพประกอบ 3-3(a)

1.2) นำผลลัพธ์จากข้อ 1.1) มาดำเนินการตรรกะ OR ในระดับบิตต่อบิต จะได้ผลลัพธ์ดังภาพประกอบ 3-3(b)

1.3) อ่านบิตแมปเวกเตอร์ E^6 ดังภาพประกอบ 3-3(c)

1.4) นำผลลัพธ์จากข้อ 1.2) และ 1.3) มาดำเนินการตรรกะ OR ในระดับบิตต่อบิต คำตอบที่ได้คือ บิตที่ 3, 6 และ 9 มีค่าบิตเท่ากับ 1 (ON) ในที่นี้คือ เรคอร์ดที่ 3, 6 และ 9 ดังภาพประกอบ 3-3(d)



(a) อ่านบิตแมปเวกเตอร์ E^1, E^4 มาดำเนินการตรรกะ OR ในระดับบิตต่อบิต

(b) ผลลัพธ์จากการ OR คือ บิตที่ 3 และ 6 มีค่าเท่ากับ 1

(c) อ่านบิตแมปเวกเตอร์ E^6

(d) ผลลัพธ์จากการนำ (b) และ (c) มาดำเนินการตรรกะ OR ในระดับบิตต่อบิต คือ บิตที่ 3, 6 และ 9 มีค่าเท่ากับ 1

ภาพประกอบ 3-3 การดำเนินการตรรกะระดับบิตของแอทริบิวต์ A ของดัชนีบิตแมปแบบพื้นฐาน

2) A = 1, 3, 5 และ 14

<u>วิธีการดึงข้อมูล</u>

2.1) อ่านบิตแมปเวกเตอร์ E^1, E^3, E^5 และ E^{14}

2.2) นำผลลัพธ์จากข้อ 2.1) มาดำเนินการตรรกะ OR ในระดับบิตต่อบิต

2.3) คำตอบที่ได้คือ ผลลัพธ์จากข้อ 2.2) ที่มีค่าบิตเท่ากับ 1 (ON) ในที่นี้คือเรคอร์ดที่ 1, 2, 5, 6 และ 10

3.1.3 การดำเนินการตรรกะระดับบิตระหว่างบิตแมปเวกเตอร์ของต่างแอทริบิวต์

นอกจากนี้ดัชนีบิตแมปแบบพื้นฐานยังมีคุณสมบัติในการดำเนินการตรรกะระดับบิตระหว่างบิตแมปเวกเตอร์ของต่างแอทริบิวต์ได้ด้วย ดังตัวอย่าง 3-2

ตัวอย่าง 3-2 กำหนดให้ตารางเชิงความสัมพันธ์ item ประกอบด้วยแอทริบิวต์ item_key, item_name, brand, type และ supplier_type และต้องการสอบถามข้อมูลว่ารายการสินค้าใดที่เป็นประเภท (type) ที่ 3 หรือ 14 เฉพาะยี่ห้อ (brand) เป็น “B” โดยในที่นี้สมมติให้สินค้ามีประเภท (ในที่นี้กำหนดให้ แอทริบิวต์ A ในภาพประกอบ 3-2(a) คือ ประเภทสินค้า) และมียี่ห้อทั้งหมด 20 ยี่ห้อ คือ “A”, “B”, “C”, “D”, …, “S”, “T” ดังภาพประกอบ 3-4(a)

ดังนั้นเราจึงเลือกเฉพาะแอทริบิวต์ type และ brand (ดังภาพประกอบ 3-4(a)) มาทำดัชนีบิตแมปแบบพื้นฐาน ดังภาพประกอบ 3-2(b) และ 3-4(b) ตามลำดับ

RID	brand
1	E
2	C
3	B
4	E
5	B
6	A
7	B
8	T
9	F
10	C

(a) เลือกเฉพาะแอทริบิวต์ brand

E^T	E^S	⋯	E^G	E^F	E^E	E^D	E^C	E^B	E^A
0	0	…	0	0	1	0	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	1	0	0	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	1	0
1	0	…	0	0	0	0	0	0	0
0	0	…	0	1	0	0	0	0	0
0	0	…	0	0	0	0	1	0	0

(b) ดัชนีบิตแมปแบบพื้นฐาน บนแอทริบิวต์ brand

ภาพประกอบ 3-4 ดัชนีบิตแมปแบบพื้นฐานบนแอทริบิวต์ brand ใช้ 20 บิตแมปเวกเตอร์

<u>วิธีการดึงข้อมูล</u>

1) ดึงข้อมูลแอทริบิวต์ type (แอทริบิวต์ A ภาพประกอบ 3-2(b)) โดยการ

1.1) อ่านบิตแมปเวกเตอร์ E^3 และ E^{14}

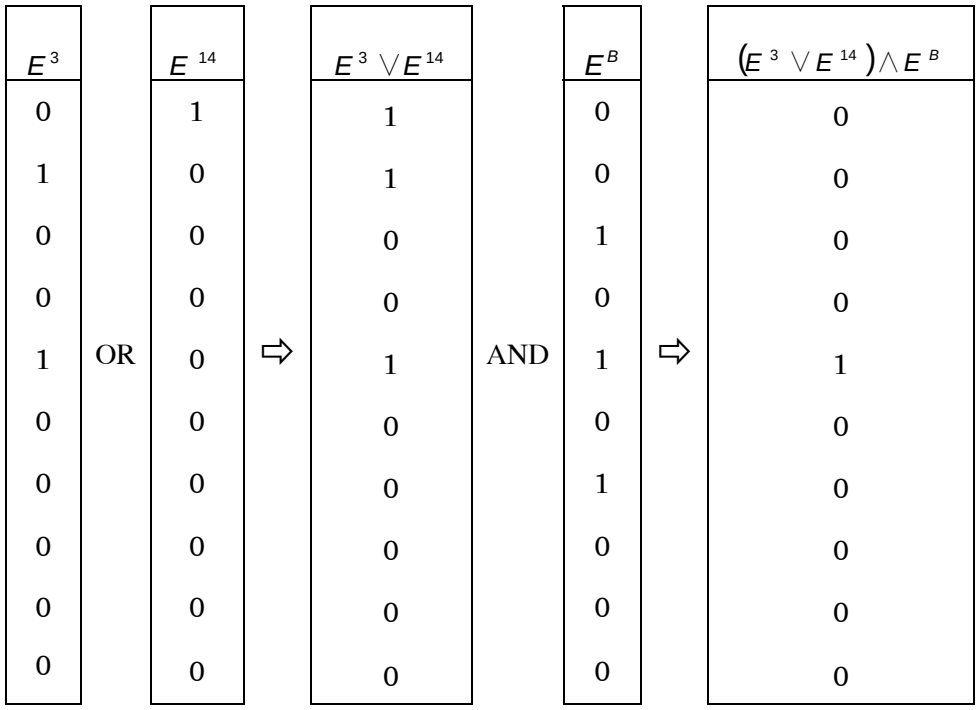
1.2) นำผลลัพธ์จากข้อ 1.1) มาดำเนินการตรรกะ OR ในระดับบิตต่อบิต ดังภาพประกอบ 3-5(a)

1.3) คำตอบที่ได้ คือ ผลลัพธ์จากข้อ 1.2) ที่มีค่าบิตเท่ากับ 1 (ON) ซึ่งก็คือ บิตที่ 1, 2 และ 5 ดังภาพประกอบ 3-5(b)

2) ดึงข้อมูลแอทริบิวต์ brand โดยการอ่านบิตแมปเวกเตอร์ E^B ดังภาพประกอบ 3-5(c)

3) นำผลลัพธ์ที่ได้จากข้อ 1.3) และ 2) มาดำเนินการตรรกะ AND ในระดับบิตต่อบิต

4) คำตอบที่ได้ คือ ผลลัพธ์จากข้อ 3) ที่มีค่าบิตเท่ากับ 1 (ON) ในที่นี้คือเรคอร์ดที่ 5 ดังภาพประกอบ 3-5(d)



(a) บิตแมปเวกเตอร์ E^3, E^{14} ดำเนินการตรรกะ OR ในระดับบิตต่อบิต

(b) ผลลัพธ์จากการ OR คือ บิตที่ 1, 2 และ 5 มีค่าเท่ากับ 1

(c) อ่านบิตแมปเวกเตอร์ E^B

(d) ผลลัพธ์จากการ AND บิตแมปเวกเตอร์ คือ บิตที่ 5 มีค่าเท่ากับ 1 (ON)

ภาพประกอบ 3-5 การดำเนินการตรรกะระดับบิตของแอทริบิวต์ type และ brand ของดัชนีบิตแมปแบบพื้นฐาน

3.1.4 ข้อดีของดัชนีบิตแมปแบบพื้นฐาน

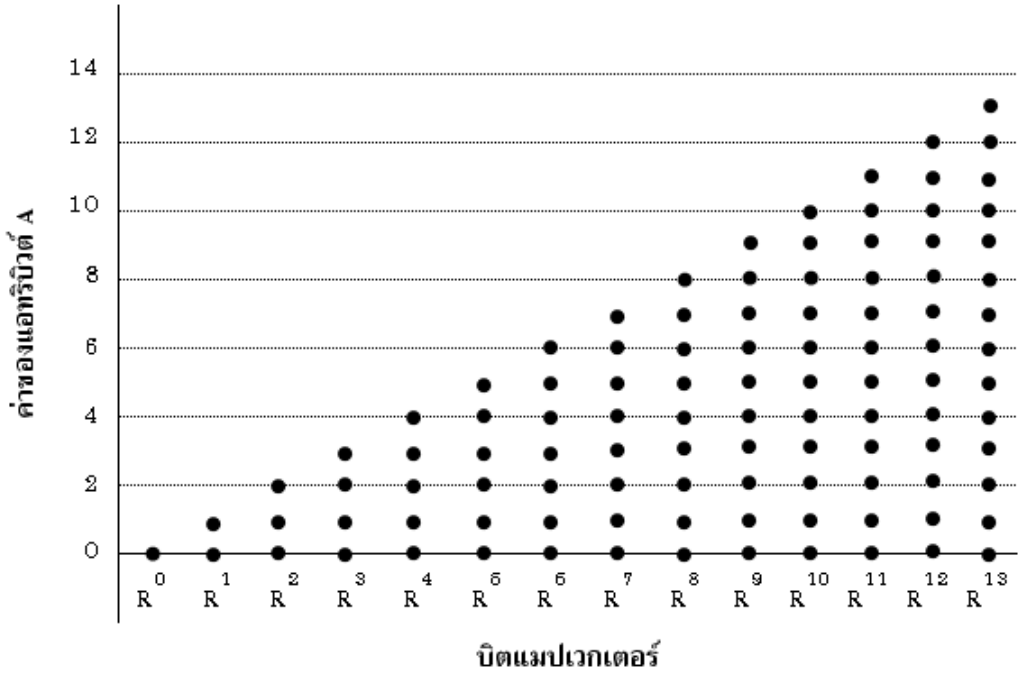
เหมาะสำหรับการสอบถามข้อมูลแบบค่าเท่ากัน เพราะมีการอ่านเพียง 1 บิตแมปเวกเตอร์ และไม่มีการดำเนินการตรรกะใด ๆ เกิดขึ้น

3.1.5 ข้อจำกัดของดัชนีบิตแมปแบบพื้นฐาน

ดัชนีบิตแมปแบบพื้นฐาน จะไม่เหมาะสำหรับแอทริบิวต์ที่มีค่าคาร์ดินอลิตี้สูง ๆ [28, 31, 32] เมตริกซ์ของบิตแมปเวกเตอร์จะเบาบาง (Sparse) คือ ประกอบด้วยบิต 0 หลายบิต จำนวนบิตที่มีค่าเป็น 0 มีจำนวนมากกว่าจำนวนบิตที่มีค่าเป็น 1 เช่น กำหนดให้คาร์ดินอลิตี้เท่ากับ C ก็จะต้องใช้พื้นที่ในการสร้างดัชนีเท่ากับ C บิตแมปเวกเตอร์ ความเบาบางของเมตริกซ์ของบิตแมปเวกเตอร์ก็จะเท่ากับ $\frac{C-1}{C}$ บิตแมปเวกเตอร์ คือ บิตที่มีค่าเป็น 0 มีจำนวนเท่ากับ C-1 บิตแมปเวกเตอร์ และบิตที่มีค่าเป็น 1 มีจำนวนเท่ากับ 1 บิตแมปเวกเตอร์เท่านั้น ซึ่งจะเห็นได้ว่าถ้าคาร์ดินอลิตี้สูง ก็จะทำให้เมตริกซ์ของบิตแมปเวกเตอร์นั้นมีความเบาบางสูง [28, 31, 32] และยิ่งทำให้เปลืองพื้นที่ในการจัดเก็บดัชนีมากขึ้น และค่าใช้จ่ายในการจัดเก็บข้อมูลก็จะสูงด้วย [33] จึงมีการใช้เทคนิควิธีการจัดการปัญหาเรื่องเมตริกซ์ของบิตแมปเวกเตอร์ที่เบาบาง เช่น โดยการบีบอัด (Compressed Bitmap) ข้อมูลที่เป็นบิต 0 [31, 32, 33]

3.2 ดัชนีบิตแมปแบบ Range (Range Bitmap Indexing)

ดัชนีบิตแมปแบบ Range [30] ประกอบด้วย $C-1$ บิตแมปเวกเตอร์ ได้แก่ บิตแมปเวกเตอร์ R^0, R^1, R^2, ..., R^i , ..., R^{C-2} โดย R^i คือ บิตแมปเวกเตอร์ที่มีค่าของแอทริบิวต์ เท่ากับ 0, 1, 2, ..., i เช่น R^0 คือ บิตแมปเวกเตอร์ที่มีค่าของแอทริบิวต์ เท่ากับ 0, R^1 คือ บิตแมปเวกเตอร์ที่มีค่าของแอทริบิวต์ เท่ากับ 0 และ 1 และจากตัวอย่างเดิม ดังภาพประกอบ 3-2(a) เมื่อนำมาทำดัชนีบิตแมปแบบ Range จะได้ดังภาพประกอบ 3-6 และ 3-7(b)



ภาพประกอบ 3-6 ตัวอย่างแผนภาพการทำดัชนีบิตแมปแบบ Range บนแอทริบิวต์ A (มีค่า C = 15) ใช้ 14 บิตแมปเวกเตอร์

จากภาพประกอบ 3-6 จะได้ว่า การทำดัชนีบิตแมปแบบนี้ ประกอบด้วย 14 บิตแมปเวกเตอร์ คือ R^0, R^1, R^2, …, R^{13}

ตัวอย่าง 3-2 กำหนดให้แอทริบิวต์ A มีค่าที่เป็นไปได้ 15 ค่า คือ 0-14 ดังภาพประกอบ 3-7

RID	แอทริบิวต์ A
1	14
2	3
3	4
4	2
5	3
6	1
7	13
8	0
9	6
10	5

R^{13}	R^{12}	…	R^5	R^4	R^3	R^2	R^1	R^0
0	0	…	0	0	0	0	0	0
1	1	…	1	1	1	0	0	0
1	1	…	1	1	0	0	0	0
1	1	…	1	1	1	1	0	0
1	1	…	1	1	1	0	0	0
1	1	…	1	1	1	1	1	0
1	0	…	0	0	0	0	0	0
1	1	…	1	1	1	1	1	1
1	1	…	0	0	0	0	0	0
1	1	…	1	0	0	0	0	0

(a) เลือกเฉพาะแอทริบิวต์ A (b) ดัชนีบิตแมปแบบ Range บนแอทริบิวต์ A

ภาพประกอบ 3-7 ดัชนีบิตแมปแบบ Range บนแอทริบิวต์ A ใช้ 14 บิตแมปเวกเตอร์

3.2.1 การค้นหาข้อมูลแบบค่าเท่ากัน ใช้เงื่อนไขดังสมการ (3.1)

$$\text{“A = } V \text{”} = \begin{cases} R^0 & \text{ถ้า } V = 0 \\ R^V \oplus R^{V-1} & \text{ถ้า } 0 < V < C - 1 \\ \overline{R^{C-2}} & \text{ถ้า } V = C - 1 \end{cases} \quad \text{- - - - - (3.1)}$$

ตัวอย่างเช่น จากภาพประกอบ 3-7 ถ้าต้องการค้นหา A = 2

<u>วิธีการดึงข้อมูล</u>

1) อ่านบิตแมปเวกเตอร์ R^2

2) หานิเสธของผลลัพธ์จากข้อ1)โดยการใช้ตัวดำเนินการตรรกะ NOT ระดับบิต

3) อ่านบิตแมปเวกเตอร์ R^1

4) หานิเสธของผลลัพธ์จากข้อ3)โดยการใช้ตัวดำเนินการตรรกะ NOT ระดับบิต

5) นำผลลัพธ์จากข้อ2) และ 3) มาดำเนินการตรรกะ AND ในระดับบิตต่อบิต

6) นำผลลัพธ์จากข้อ1) และ 4) มาดำเนินการตรรกะ AND ในระดับบิตต่อบิต

7) นำผลลัพธ์จากข้อ 5) และ 6) มาดำเนินการตรรกะ OR ในระดับบิตต่อบิต

ซึ่งคำตอบที่ได้คือ ผลลัพธ์ของบิตแมปเวกเตอร์ที่มีค่าบิตเท่ากับ 1 (ON) ในที่นี้คือ เรคอร์ดที่ 4

วิธีการดึงข้อมูลตั้งแต่ขั้นตอนที่ 1) ถึง 7) คือ การอ่านบิตแมปเวกเตอร์ R^2 ดำเนินการตรรกะ XOR กับบิตแมปเวกเตอร์ R^1 ดังสมการ (3.1) เพื่อค้นหาค่า A = 2 (ใช้สูตร $R^V \oplus R^{V-1}$ เพราะ 0 < V < C - 1) ซึ่งคำตอบที่ได้ คือ ผลลัพธ์ของบิตแมปเวกเตอร์ $R^2 \oplus R^1$ ที่มีค่าบิตเท่ากับ 1 (ON) ในที่นี้คือ เรคอร์ดที่ 4 นั่นเอง

3.2.2 การค้นหาข้อมูลแบบความเป็นสมาชิก

สามารถค้นหาโดยการอ่านค่าบิตแมปเวกเตอร์ตามเงื่อนไขการค้นหาข้อมูลแบบค่าเท่ากัน ดังสมการ (3.1) ของแต่ละค่าที่จะค้นหาขึ้นมาดำเนินการตรรกะ OR กันระหว่างบิตแมปเวกเตอร์

ตัวอย่างเช่น จากภาพประกอบ 3-7 ถ้าต้องการค้นหา

1) A = 1, 4 และ 6

<u>วิธีการดึงข้อมูล</u>

1.1) อ่านบิตแมปเวกเตอร์ R^1

1.2) หานิเสธของผลลัพธ์จากข้อ 1.1) โดยการใช้ตัวดำเนินการตรรกะ NOT ระดับบิต

1.3) อ่านบิตแมปเวกเตอร์ R^0

1.4) หานิเสธของผลลัพธ์จากข้อ 1.3) โดยการใช้ตัวดำเนินการตรรกะ NOT ระดับบิต

1.5) นำผลลัพธ์จากข้อ 1.2) และ 1.3) มาดำเนินการตรรกะ AND ในระดับบิตต่อบิต

1.6) นำผลลัพธ์จากข้อ 1.1) และ 1.4) มาดำเนินการตรรกะ AND ในระดับบิตต่อบิต

1.7) นำผลลัพธ์จากข้อ 1.5) และ 1.6) มาดำเนินการตรรกะ OR ในระดับบิตต่อบิต ซึ่งคำตอบที่ได้คือ ผลลัพธ์ของบิตแมปเวกเตอร์ที่มีค่าบิตเท่ากับ 1 (ON)

วิธีการดึงข้อมูลตั้งแต่ขั้นตอนที่ 1.1) ถึง 1.7) คือ การอ่านบิตแมปเวกเตอร์ R^1 ดำเนินการตรรกะ XOR กับบิตแมปเวกเตอร์ R^0 ดังสมการ (3.1) เพื่อค้นหาค่า A = 1 (ใช้สูตร $R^v \oplus R^{v-1}$ เพราะ $0 < v < C - 1$)

1.8) อ่านบิตแมปเวกเตอร์ R^4

1.9) หานิเสธของผลลัพธ์จากข้อ 1.8) โดยการใช้ตัวดำเนินการตรรกะ NOT ระดับบิต

1.10) อ่านบิตแมปเวกเตอร์ R^3

1.11) หานิเสธของผลลัพธ์จากข้อ 1.10) โดยการใช้ตัวดำเนินการตรรกะ NOT ระดับบิต

1.12) นำผลลัพธ์จากข้อ 1.9) และ 1.10) มาดำเนินการตรรกะ AND ในระดับบิตต่อบิต

1.13) นำผลลัพธ์จากข้อ 1.8) และ 1.11) มาดำเนินการตรรกะ AND ในระดับบิตต่อบิต

1.14) นำผลลัพธ์จากข้อ 1.12) และ 1.13) มาดำเนินการตรรกะ OR ในระดับบิตต่อบิต ซึ่งคำตอบที่ได้คือ ผลลัพธ์ของบิตแมปเวกเตอร์ที่มีค่าบิตเท่ากับ 1 (ON)

วิธีการดึงข้อมูลตั้งแต่ขั้นตอนที่ 1.8) ถึง 1.14) คือ การอ่านบิตแมปเวกเตอร์ R^4 ดำเนินการตรรกะ XOR กับบิตแมปเวกเตอร์ R^3 ดังสมการ (3.1) เพื่อค้นหาค่า A = 4 (ใช้สูตร $R^v \oplus R^{v-1}$ เพราะ $0 < v < C - 1$)

1.15) อ่านบิตแมปเวกเตอร์ R^6

1.16) หานิเสธของผลลัพธ์จากข้อ 1.15) โดยการใช้ตัวดำเนินการตรรกะ NOT ระดับบิต

1.17) อ่านบิตแมปเวกเตอร์ R^5

1.18) หานิเสธของผลลัพธ์จากข้อ 1.17) โดยการใช้ตัวดำเนินการตรรกะ NOT ระดับบิต

1.19) นำผลลัพธ์จากข้อ 1.16) และ 1.17) มาดำเนินการตรรกะ AND ในระดับบิตต่อบิต

1.20) นำผลลัพธ์จากข้อ 1.15) และ 1.18) มาดำเนินการตรรกะ AND ในระดับบิตต่อบิต

1.21) นำผลลัพธ์จากข้อ 1.19) และ 1.20) มาดำเนินการตรรกะ OR ในระดับบิตต่อบิต ซึ่งคำตอบที่ได้คือ ผลลัพธ์ของบิตแมปเวกเตอร์ที่มีค่าบิตเท่ากับ 1 (ON)

วิธีการดึงข้อมูลตั้งแต่ขั้นตอนที่ 1.15) ถึง 1.21) คือ การอ่านบิตแมปเวกเตอร์ R^6 ดำเนินการตรรกะ XOR กับบิตแมปเวกเตอร์ R^5 ดังสมการ (3.1) เพื่อค้นหาค่า A = 6 (ใช้สูตร $R^{v} \oplus R^{v-1}$ เพราะ $0 < v < C - 1$)

1.22) นำผลลัพธ์จากข้อ 1.7), 1.14) และ 1.21) มาดำเนินการตรรกะ OR ในระดับบิตต่อบิต ซึ่งคำตอบที่ได้คือ ผลลัพธ์ของบิตแมปเวกเตอร์ที่มีค่าบิตเท่ากับ 1 (ON) ในที่นี้คือ เรคอร์ดที่ 3, 6 และ 9 นั่นเอง

2) A = 1, 3, 5 และ 14

<u>วิธีการดึงข้อมูล</u>

2.1) อ่านบิตแมปเวกเตอร์ R^1

2.2) หานิเสธของผลลัพธ์จากข้อ 2.1) โดยการใช้ตัวดำเนินการตรรกะ NOT ระดับบิต

2.3) อ่านบิตแมปเวกเตอร์ R^0

2.4) หานิเสธของผลลัพธ์จากข้อ 2.3) โดยการใช้ตัวดำเนินการตรรกะ NOT ระดับบิต

2.5) นำผลลัพธ์จากข้อ 2.2) และ 2.3) มาดำเนินการตรรกะ AND ในระดับบิตต่อบิต

2.6) นำผลลัพธ์จากข้อ 2.1) และ 2.4) มาดำเนินการตรรกะ AND ในระดับบิตต่อบิต

2.7) นำผลลัพธ์จากข้อ 2.5) และ 2.6) มาดำเนินการตรรกะ OR ในระดับบิตต่อบิต ซึ่งคำตอบที่ได้คือ ผลลัพธ์ของบิตแมปเวกเตอร์ที่มีค่าบิตเท่ากับ 1 (ON)

วิธีการดึงข้อมูลตั้งแต่ขั้นตอนที่ 2.1) ถึง 2.7) คือ การอ่านบิตแมปเวกเตอร์ R^1 ดำเนินการตรรกะ XOR กับบิตแมปเวกเตอร์ R^0 ดังสมการ (3.1) เพื่อค้นหาค่า A = 1 (ใช้สูตร $R^{v} \oplus R^{v-1}$ เพราะ $0 < v < C - 1$)

2.8) อ่านบิตแมปเวกเตอร์ R^3

2.9) หานิเสธของผลลัพธ์จากข้อ 2.8) โดยการใช้ตัวดำเนินการตรรกะ NOT ระดับบิต

2.10) อ่านบิตแมปเวกเตอร์ R^2

2.11) หานิเสธของผลลัพธ์จากข้อ 2.10) โดยการใช้ตัวดำเนินการตรรกะ NOT ระดับบิต

2.12) นำผลลัพธ์จากข้อ 2.9) และ 2.10) มาดำเนินการตรรกะ AND ในระดับบิตต่อบิต

2.13) นำผลลัพธ์จากข้อ 2.8) และ 2.11) มาดำเนินการตรรกะ AND ในระดับบิตต่อบิต

2.14) นำผลลัพธ์จากข้อ 2.12) และ 2.13) มาดำเนินการตรรกะ OR ในระดับบิตต่อบิต ซึ่งคำตอบที่ได้คือ ผลลัพธ์ของบิตแมปเวกเตอร์ที่มีค่าบิตเท่ากับ 1 (ON)

วิธีการดึงข้อมูลตั้งแต่ขั้นตอนที่ 2.8) ถึง 2.14) คือ การอ่านบิตแมปเวกเตอร์ R^3 ดำเนินการตรรกะ XOR กับบิตแมปเวกเตอร์ R^2 ดังสมการ (3.1) เพื่อค้นหาค่า A = 3 (ใช้สูตร $R^v \oplus R^{v-1}$ เพราะ $0 < v < C - 1$)

2.15) อ่านบิตแมปเวกเตอร์ R^5

2.16) หานิเสธของผลลัพธ์จากข้อ 2.15) โดยการใช้ตัวดำเนินการตรรกะ NOT ระดับบิต

2.17) อ่านบิตแมปเวกเตอร์ R^4

2.18) หานิเสธของผลลัพธ์จากข้อ 2.17) โดยการใช้ตัวดำเนินการตรรกะ NOT ระดับบิต

2.19) นำผลลัพธ์จากข้อ 2.16) และ 2.17) มาดำเนินการตรรกะ AND ในระดับบิตต่อบิต

2.20) นำผลลัพธ์จากข้อ 2.15) และ 2.18) มาดำเนินการตรรกะ AND ในระดับบิตต่อบิต

2.21) นำผลลัพธ์จากข้อ 2.19) และ 2.20) มาดำเนินการตรรกะ OR ในระดับบิตต่อบิต ซึ่งคำตอบที่ได้คือ ผลลัพธ์ของบิตแมปเวกเตอร์ที่มีค่าบิตเท่ากับ 1 (ON)

วิธีการดึงข้อมูลตั้งแต่ขั้นตอนที่ 2.15) ถึง 2.21) คือ การอ่านบิตแมปเวกเตอร์ R^5 ดำเนินการตรรกะ XOR กับบิตแมปเวกเตอร์ R^4 ดังสมการ (3.1) เพื่อค้นหาค่า A = 5 (ใช้สูตร $R^v \oplus R^{v-1}$ เพราะ $0 < v < C - 1$)

2.22) อ่านบิตแมปเวกเตอร์ R^{13}

2.23) หานิเสธของผลลัพธ์จากข้อ 2.22) โดยการใช้ตัวดำเนินการตรรกะ NOT ระดับบิต เพื่อค้นหาค่า A = 14 (ใช้สูตร $\overline{R^{C-2}}$ เพราะ $V = C-1$)

2.24) นำผลลัพธ์จากข้อ 2.7), 2.14), 2.21) และ 2.23) มาดำเนินการตรรกะ OR ในระดับบิตต่อบิต ซึ่งคำตอบที่ได้คือ ผลลัพธ์ของบิตแมปเวกเตอร์ที่มีค่าบิตเท่ากับ 1 (ON) ในที่นี้คือ เรคอร์ดที่ 1, 2, 5, 6 และ 10

3.2.3 การดำเนินการตรรกะระดับบิตระหว่างบิตแมปเวกเตอร์ของต่างแอทริบิวต์

นอกจากนี้ดัชนีบิตแมปแบบ Range ยังมีคุณสมบัติในการดำเนินการตรรกะระดับบิตระหว่างบิตแมปเวกเตอร์ของต่างแอทริบิวต์ได้ด้วย เช่น จากตัวอย่างเดิม 3-2 เมื่อต้องการค้นหาข้อมูล type = 3, 14 และ brand = "B" สามารถทำได้โดยการเลือกแอทริบิวต์ type และ brand มาทำดัชนีบิตแมปแบบ Range ได้ดังภาพประกอบ 3-7(b) และ 3-8(b) ตามลำดับ โดยกำหนดให้ brand = "A", "B", "C", "D", …, "S", "T" มีลำดับที่เท่ากับ 0, 1, 2, 3, ..., 18, 19 ตามลำดับ

RID	brand
1	E
2	C
3	B
4	E
5	B
6	A
7	B
8	T
9	F
10	C

(a) เลือกเฉพาะแอทริบิวต์ brand

R^{18}	R^{17}	⋯	R^{6}	R^{5}	R^{4}	R^{3}	R^{2}	R^{1}	R^{0}
1	1	…	1	1	1	0	0	0	0
1	1	…	1	1	1	1	1	0	0
1	1	…	1	1	1	1	1	1	0
1	1	…	1	1	1	0	0	0	0
1	1	…	1	1	1	1	1	1	0
1	1	…	1	1	1	1	1	1	1
1	1	…	1	1	1	1	1	1	0
0	0	…	0	0	0	0	0	0	0
1	1	…	1	1	0	0	0	0	0
1	1	…	1	1	1	1	1	0	0

(b) ดัชนีบิตแมปแบบ Range บนแอทริบิวต์ brand

ภาพประกอบ 3-8 ดัชนีบิตแมปแบบ Range บนแอทริบิวต์ brand ใช้ 19 บิตแมปเวกเตอร์

<u>วิธีการดึงข้อมูล</u>

1) ดึงข้อมูลแอทริบิวต์ type (แอทริบิวต์ A ภาพประกอบ 3-7(b)) โดยการ

1.1) อ่านบิตแมปเวกเตอร์ R^3

1.2) หานิเสธของผลลัพธ์จากข้อ 1.1) โดยการใช้ตัวดำเนินการตรรกะ NOT ระดับบิต

1.3) อ่านบิตแมปเวกเตอร์ R^2

1.4) หานิเสธของผลลัพธ์จากข้อ 1.3) โดยการใช้ตัวดำเนินการตรรกะ NOT ระดับบิต

1.5) นำผลลัพธ์จากข้อ 1.2) และ 1.3) มาดำเนินการตรรกะ AND ในระดับบิตต่อบิต

1.6) นำผลลัพธ์จากข้อ 1.1) และ 1.4) มาดำเนินการตรรกะ AND ในระดับบิตต่อบิต

1.7) นำผลลัพธ์จากข้อ 1.5) และ 1.6) มาดำเนินการตรรกะ OR ในระดับบิตต่อบิต ซึ่งคำตอบที่ได้คือ ผลลัพธ์ของบิตแมปเวกเตอร์ที่มีค่าบิตเท่ากับ 1 (ON) ดังภาพประกอบ 3-9(a)

วิธีการดึงข้อมูลตั้งแต่ขั้นตอนที่ 1.1) ถึง 1.7) คือ การอ่านบิตแมปเวกเตอร์ R^3 ดำเนินการตรรกะ XOR กับบิตแมปเวกเตอร์ R^2 ดังสมการ (3.1) เพื่อค้นหาค่า A = 3 (ใช้สูตร $R^{v} \oplus R^{v-1}$ เพราะ $0 < v < C - 1$)

1.8) อ่านบิตแมปเวกเตอร์ R^{13}

1.9) หานิเสธของผลลัพธ์จากข้อ 1.8) โดยการใช้ตัวดำเนินการตรรกะ NOT ระดับบิต เพื่อค้นหาข้อมูลค่า A = 14 (ใช้สูตร $\overline{R^{C-2}}$ เพราะ $v = C - 1$)

1.10) นำผลลัพธ์จากข้อ 1.7) และ 1.9) มาดำเนินการตรรกะ OR ในระดับบิตต่อบิต ซึ่งคำตอบที่ได้คือ ผลลัพธ์ของบิตแมปเวกเตอร์ที่มีค่าบิตเท่ากับ 1 (ON) ดังภาพประกอบ 3-9(b)

2) ดึงข้อมูลแอทริบิวต์ brand จากภาพประกอบ 3-8(b) โดยการ

2.1) อ่านบิตแมปเวกเตอร์ R^1

2.2) หานิเสธของผลลัพธ์จากข้อ 2.1) โดยการใช้ตัวดำเนินการตรรกะ NOT ระดับบิต

2.3) อ่านบิตแมปเวกเตอร์ R^0

2.4) หานิเสธของผลลัพธ์จากข้อ 2.3) โดยการใช้ตัวดำเนินการตรรกะ NOT ระดับบิต

2.5) นำผลลัพธ์จากข้อ 2.2) และ 2.3) มาดำเนินการตรรกะ AND ในระดับบิตต่อบิต

2.6) นำผลลัพธ์จากข้อ 2.1) และ 2.4) มาดำเนินการตรรกะ AND ในระดับบิตต่อบิต

2.7) นำผลลัพธ์จากข้อ 2.5) และ 2.6) มาดำเนินการตรรกะ OR ในระดับบิตต่อบิต ซึ่งคำตอบที่ได้คือ ผลลัพธ์ของบิตแมปเวกเตอร์ที่มีค่าบิตเท่ากับ 1 (ON)

วิธีการดึงข้อมูลตั้งแต่ขั้นตอนที่ 2.1) ถึง 2.7) คือ การอ่านบิตแมปเวกเตอร์ R^1 ดำเนินการตรรกะ XOR กับบิตแมปเวกเตอร์ R^0 ดังสมการ (3.1) เพื่อค้นหา ข้อมูลค่า brand = “B” (ใช้สูตร $R^{v} \oplus R^{v-1}$ เพราะ $0 < v < C - 1$) ดังภาพประกอบ 3-9(c)

3) นำผลลัพธ์ที่ได้จากข้อ 1.10) และ 2.7) มาดำเนินการตรรกะ AND ในระดับบิตต่อบิต ดังภาพประกอบ 3-9(d)

4) คำตอบที่ได้ คือ ผลลัพธ์จากข้อ 3) ที่มีค่าบิตเท่ากับ 1(ON) ในที่นี้คือ เรคอร์ดที่ 5 ดังภาพประกอบ 3-9(e)

$\overline{R^3} \wedge R^2$	OR	$R^3 \wedge \overline{R^2}$	⇨	$(\overline{R^3} \wedge R^2) \vee (R^3 \wedge \overline{R^2})$
0		0		0
0		1		1
0		0		0
0		0		0
0		1		1
0		0		0
0		0		0
0		0		0
0		0		0
0		0		0

(a) ผลลัพธ์ $R^3 \oplus R^2$ เพื่อค้นหาค่า A=3

$(\overline{R^3} \wedge R^2) \vee (R^3 \wedge \overline{R^2})$	OR	$\overline{R^{13}}$	⇨	$((\overline{R^3} \wedge R^2) \vee (R^3 \wedge \overline{R^2})) \vee (\overline{R^{13}})$
0		1		1
1		0		1
0		0		0
0		0		0
0		0		1
0		0		0
0		0		0
0		0		0
0		0		0
0		0		0

(b) นำผลลัพธ์จากการค้นหาค่า A=3 และ A=14 มาดำเนินการตรรกะ OR ในระดับบิตต่อบิต

$\overline{R^1} \wedge R^0$	OR	$R^1 \wedge \overline{R^0}$	⇨	$(\overline{R^1} \wedge R^0) \vee (R^1 \wedge \overline{R^0})$
0		0		0
0		0		0
0		1		1
0		0		0
0		1		1
0		0		0
0		1		1
0		0		0
0		0		0
0		0		0

(c) ผลลัพธ์ $R^1 \oplus R^0$ เพื่อค้นหาค่า brand = “B”

$((\overline{R^3} \wedge R^2) \vee (R^3 \wedge \overline{R^2})) \vee (\overline{R^{13}})$	AND	$(\overline{R^1} \wedge R^0) \vee (R^1 \wedge \overline{R^0})$	⇨	ผลลัพธ์
1		0		0
1		0		0
0		1		0
0		0		0
1		1		1
0		0		0
0		1		0
0		0		0
0		0		0
0		0		0

(d) นำผลลัพธ์จากการค้นหาค่า A=3, 14 และค่า brand=“B” มาดำเนินการตรรกะ AND

(e) ผลลัพธ์จากการ AND ในระดับบิตต่อบิต คือ บิตที่ 5 มีค่าบิตเท่ากับ 1

ภาพประกอบ 3-9 การดำเนินการตรรกะระดับบิตของแอทริบิวต์ type และ brand ของดัชนีบิตแมปแบบ Range

3.2.4 ข้อดีของดัชนีบิตแมปแบบ Range

เหมาะสำหรับแอทริบิวต์ที่มีคาร์ดินอลิตี้ต่ำ ๆ

3.2.5 ข้อจำกัดของดัชนีบิตแมปแบบ Range

ไม่เหมาะสำหรับการสอบถามข้อมูลแบบค่าเท่ากัน เพราะจำนวนบิตแมปที่ถูกอ่านจะเท่ากับ 2 และจะมีการดำเนินการตรรกะ 5 ตัวดำเนินการ คือ $\left(R^v \wedge \overline{R^{v-1}}\right) \vee \left(\overline{R^v} \wedge R^{v-1}\right)$

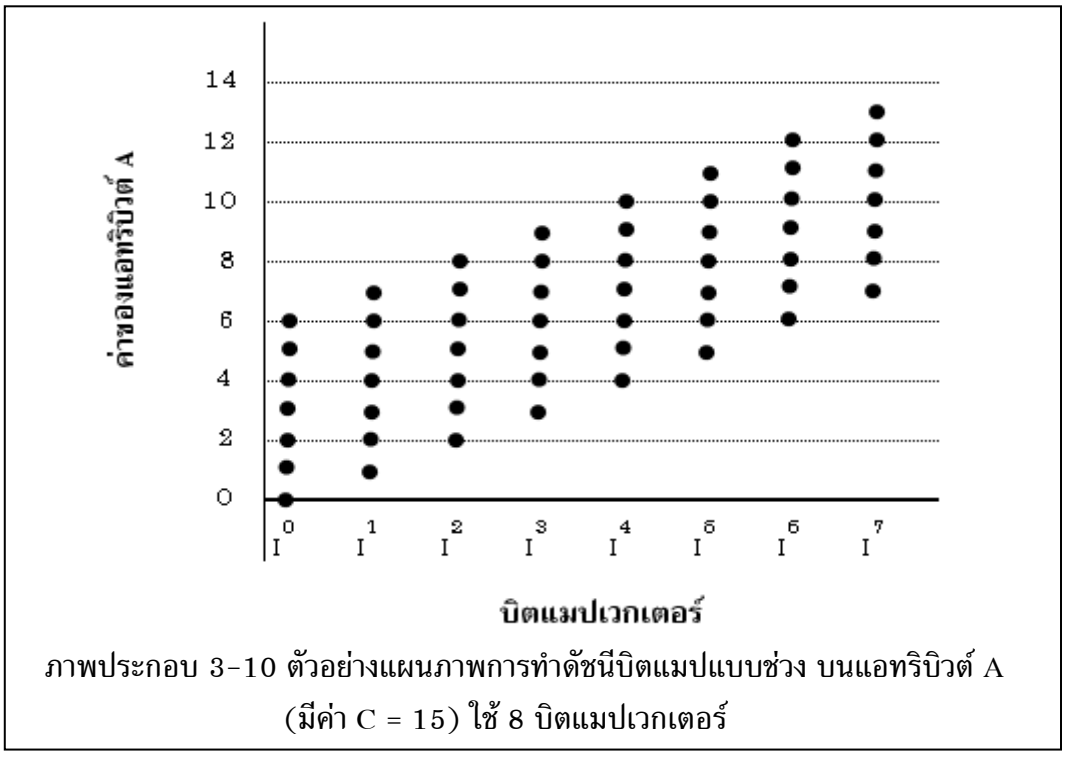
3.3 ดัชนีบิตแมปแบบช่วง (Interval Bitmap Indexing)

ดัชนีบิตแมปแบบช่วง [30] ประกอบด้วย $\left\lceil \frac{C}{2} \right\rceil$ บิตแมปเวกเตอร์ ได้แก่ บิตแมปเวกเตอร์ $I^0, I^1, I^2, \ldots, I^j, \ldots, I^{\left\lceil \frac{C}{2} \right\rceil - 1}$ โดย I^j คือ บิตแมปเวกเตอร์ที่มีค่าของแอทริบิวต์ เท่ากับ $j, j+1, j+2, \ldots, j+m$ โดย m มีค่าเท่ากับ $\left\lfloor \frac{C}{2} \right\rfloor - 1$ เช่น I^2 คือ บิตแมปเวกเตอร์ที่มีค่าของแอทริบิวต์ เท่ากับ 2, 3, 4, 5, 6, 7 และ 8 ดังตัวอย่าง 3-3

ตัวอย่าง 3-3 กำหนดให้แอทริบิวต์ A มีคาร์ดินอลิตี้ (C) เท่ากับ 15 จะได้ว่า

$$\left\lceil \frac{C}{2} \right\rceil - 1 = \left\lceil \frac{15}{2} \right\rceil - 1 = 7 \text{ และ } m = \left\lfloor \frac{C}{2} \right\rfloor - 1 = \left\lfloor \frac{15}{2} \right\rfloor - 1 = 6$$

จาก $I = \left\{ I^0, I^1, \ldots, I^{\left\lceil \frac{C}{2} \right\rceil - 1} \right\}$ จะได้ $I = \left\{ I^0, I^1, I^2, I^3, I^4, I^5, I^6, I^7 \right\}$



ภาพประกอบ 3-10 ตัวอย่างแผนภาพการทำดัชนีบิตแมปแบบช่วง บนแอทริบิวต์ A (มีค่า C = 15) ใช้ 8 บิตแมปเวกเตอร์

จาก $I^j = [j, j+m]$ จะได้ดังนี้ (ดูภาพประกอบ 3-10 และ 3-11)

I^0 = [0, 6] , I^1 = [1, 7]

I^2 = [2, 8], I^3 = [3, 9]

I^4 = [4, 10], I^5 = [5, 11]

I^6 = [6, 12], I^7 = [7, 13]

RID	แอทริบิวต์ A
1	14
2	3
3	4
4	2
5	3
6	1
7	13
8	0
9	6
10	5

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

(a) เลือกเฉพาะแอทริบิวต์ A (b) ดัชนีบิตแมปแบบช่วง บนแอทริบิวต์ A

ภาพประกอบ 3-11 ดัชนีบิตแมปแบบช่วง บนแอทริบิวต์ A ใช้ 8 บิตแมปเวกเตอร์

3.3.1 การค้นหาข้อมูลแบบค่าเท่ากัน ใช้เงื่อนไขดังสมการ (3.2)

$$\text{“A = v”} = \begin{cases} I^0 & \text{ถ้า } v = 0, m = 0 \\ \overline{I^0} & \text{ถ้า } v = 1, C = 2 \\ I^1 & \text{ถ้า } v = 1, C = 3 \\ I^v \wedge \overline{I^{v+1}} & \text{ถ้า } v < m \\ I^v \wedge I^0 & \text{ถ้า } v = m, m > 0 \\ I^{v-m} \wedge \overline{I^{v-m-1}} & \text{ถ้า } m < v < C - 1, m > 0 \\ \overline{I^{\left\lceil \frac{C}{2} \right\rceil - 1}} \vee I^0 & \text{ถ้า } v = C - 1 \end{cases} \quad \text{- - - - - (3.2)}$$

ตัวอย่างเช่น จากภาพประกอบ 3-11 ถ้าต้องการค้นหา A = 2

โดยการพิจารณาค่า v และ m ที่กำหนดให้ จะเห็นว่า v = 2, m = 6 จึงใช้เงื่อนไขที่ 4 สมการที่ 3.2 ในการดึงข้อมูล

$$\text{“A = 2”} = I^{v} \wedge \overline{I^{v+1}}$$
$$= I^{2} \wedge \overline{I^{2+1}}$$
$$= I^{2} \wedge \overline{I^{3}}$$

<u>วิธีการดึงข้อมูล</u>

1) อ่านบิตแมปเวกเตอร์ I^2

2) อ่านบิตแมปเวกเตอร์ I^3

3) หานิเสธของผลลัพธ์จากข้อ2) โดยการใช้ตัวดำเนินการตรรกะNOT ระดับบิต

4) นำผลลัพธ์จากข้อ 1) และ 3) ดำเนินการตรรกะ AND ในระดับบิตต่อบิต

5) คำตอบที่ได้ คือ ผลลัพธ์จากข้อ 4) ที่มีค่าบิตเท่ากับ 1 (ON) ในที่นี้คือเรคอร์ดที่ 4

3.3.2 การค้นหาข้อมูลแบบความเป็นสมาชิก

สามารถค้นหาโดยการอ่านค่าบิตแมปเวกเตอร์ ตามเงื่อนไขการค้นหาข้อมูลแบบค่าเท่ากัน ดังสมการ (3.2) ของแต่ละค่าที่จะค้นหาขึ้นมาดำเนินการตรรกะ OR กันระหว่างบิตแมปเวกเตอร์

ตัวอย่างเช่น จากภาพประกอบ 3-11 ถ้าต้องการค้นหา

1) A = 1, 4 และ 6

<u>วิธีการดึงข้อมูล</u>

1.1) อ่านบิตแมปเวกเตอร์ I^1

1.2) อ่านบิตแมปเวกเตอร์ I^2

1.3) หานิเสธของผลลัพธ์จากข้อ 1.2)

1.4) นำผลลัพธ์จากข้อ 1.1) และ 1.3) มาดำเนินการตรรกะ AND กันในระดับบิตต่อบิต เพื่อค้นหาข้อมูลค่า A = 1

1.5) อ่านบิตแมปเวกเตอร์ I^4

1.6) อ่านบิตแมปเวกเตอร์ I^5

1.7) หานิเสธของผลลัพธ์จากข้อ 1.6)

1.8) นำผลลัพธ์จากข้อ 1.5) และ 1.7) มาดำเนินการตรรกะ AND กันในระดับบิตต่อบิต เพื่อค้นหาข้อมูลค่า A = 4

1.9) อ่านบิตแมปเวกเตอร์ I^6

1.10) อ่านบิตแมปเวกเตอร์ I^0

1.11) นำผลลัพธ์จากข้อ 1.9) และ 1.10) มาดำเนินการตรรกะ AND กันในระดับบิตต่อบิต เพื่อค้นหาข้อมูลค่า A = 6

1.12) นำผลลัพธ์จากข้อ 1.4), 1.8) และ 1.11) มาดำเนินการตรรกะ OR ในระดับบิตต่อบิต ซึ่งคำตอบที่ได้คือ ผลลัพธ์ของบิตแมปเวกเตอร์ที่มีค่าบิตเท่ากับ 1 (ON) ในที่นี้คือ เรคอร์ดที่ 3, 6 และ 9

2) A = 1, 3, 5 และ 14

<u>วิธีการดึงข้อมูล</u>

2.1) อ่านบิตแมปเวกเตอร์ I^1

2.2) อ่านบิตแมปเวกเตอร์ I^2

2.3) หานิเสธของผลลัพธ์จากข้อ 2.2)

2.4) นำผลลัพธ์จากข้อ 2.1) และ 2.3) มาดำเนินการตรรกะ AND กันในระดับบิตต่อบิต เพื่อค้นหาข้อมูลค่า A = 1

2.5) อ่านบิตแมปเวกเตอร์ I^3

2.6) อ่านบิตแมปเวกเตอร์ I^4

2.7) หานิเสธของผลลัพธ์จากข้อ 2.6)

2.8) นำผลลัพธ์จากข้อ 2.5) และ 2.7) มาดำเนินการตรรกะ AND กันในระดับบิตต่อบิต เพื่อค้นหาข้อมูลค่า A = 3

2.9) อ่านบิตแมปเวกเตอร์ I^5

2.10) อ่านบิตแมปเวกเตอร์ I^6

2.11) หานิเสธของผลลัพธ์จากข้อ 2.10)

2.12) นำผลลัพธ์จากข้อ 2.9) และ 2.11) มาดำเนินการตรรกะ AND กันในระดับบิตต่อบิต เพื่อค้นหาข้อมูลค่า A = 5

2.13) อ่านบิตแมปเวกเตอร์ I^7

2.14) อ่านบิตแมปเวกเตอร์ I^0

2.15) นำผลลัพธ์จากข้อ 2.13) และ 2.14) มาดำเนินการตรรกะ OR กันในระดับบิตต่อบิต

2.16) หานิเสธของผลลัพธ์จากข้อ 2.15) เพื่อค้นหาข้อมูลค่า A = 14

2.17) นำผลลัพธ์จากข้อ 2.4), 2.8), 2.12) และ 2.16) มาดำเนินการตรรกะ OR ในระดับบิตต่อบิต ซึ่งคำตอบที่ได้คือ ผลลัพธ์ของบิตแมปเวกเตอร์ที่มีค่าบิตเท่ากับ 1 (ON) ในที่นี้คือ เรคอร์ดที่ 1, 2, 5, 6 และ 10

3.3.3 การดำเนินการตรรกะระดับบิตระหว่างบิตแมปเวกเตอร์ของต่างแอทริบิวต์

นอกจากนี้ดัชนีบิตแมปแบบช่วง ยังมีคุณสมบัติในการดำเนินการตรรกะระดับบิตระหว่างบิตแมปเวกเตอร์ได้ด้วย เช่น จากตัวอย่างเดิม 3-2 เมื่อต้องการค้นหาข้อมูล type = 3, 14 และ brand = "B" สามารถทำได้โดยการเลือกแอทริบิวต์ type และ brand มาทำดัชนีบิตแมปแบบช่วงได้ดังภาพประกอบ 3-11(b) และ 3-12(b) ตามลำดับ โดยกำหนดให้ brand = "A", "B", "C", "D", …, "S", "T" มีลำดับที่เท่ากับ 0, 1, 2, 3, ..., 18, 19 ตามลำดับ

RID	brand
1	E
2	C
3	B
4	E
5	B
6	A
7	B
8	T
9	F
10	C

(a) เลือกเฉพาะแอทริบิวต์ brand

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

(b) ดัชนีบิตแมปแบบช่วง บนแอทริบิวต์ brand

ภาพประกอบ 3-12 ดัชนีบิตแมปแบบช่วง บนแอทริบิวต์ brand ใช้ 10 บิตแมปเวกเตอร์

<u>วิธีการดึงข้อมูล</u>

1) ดึงข้อมูลแอทริบิวต์ type (แอทริบิวต์ A ภาพประกอบ 3-11(b)) โดยการ

1.1) อ่านบิตแมปเวกเตอร์ I^3

1.2) อ่านบิตแมปเวกเตอร์ I^4

1.3) หานิเสธของผลลัพธ์จากข้อ 1.2) โดยการใช้ตัวดำเนินการตรรกะ NOT ระดับบิต

1.4) นำผลลัพธ์จากข้อ 1.1) และ 1.3) มาดำเนินการตรรกะ AND ในระดับบิตต่อบิต เพื่อค้นหาค่า A = 3 ดังภาพประกอบ 3-13(a)

1.5) อ่านบิตแมปเวกเตอร์ I^7

1.6) อ่านบิตแมปเวกเตอร์ I^0

1.7) นำผลลัพธ์จากข้อ1.5) และ 1.6) ดำเนินการตรรกะ OR ในระดับบิต

1.8) หานิเสธของผลลัพธ์จากข้อ 1.7) โดยการใช้ตัวดำเนินการตรรกะ NOT ระดับบิต เพื่อค้นหาค่า A = 14 ดังภาพประกอบ 3-13(b)

1.9) นำผลลัพธ์จากข้อ 1.4) และ 1.8) มาดำเนินการตรรกะ OR ในระดับบิตต่อบิต ซึ่งคำตอบที่ได้คือ ผลลัพธ์ของบิตแมปเวกเตอร์ที่มีค่าบิตเท่ากับ 1 (ON) ดังภาพประกอบ 3-13(c)

2) ดึงข้อมูลแอทริบิวต์ brand จากภาพประกอบ 3-12(b) โดยการ

2.1) อ่านบิตแมปเวกเตอร์ I^1

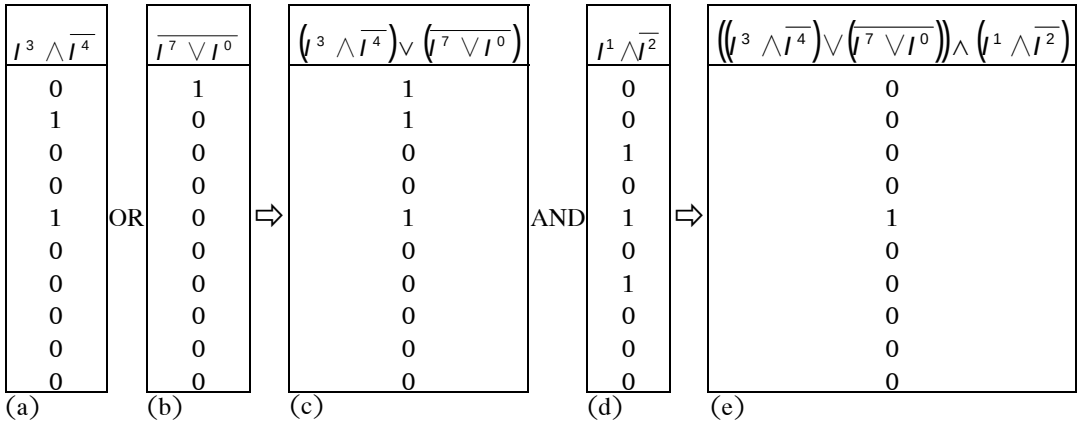
2.2) อ่านบิตแมปเวกเตอร์ I^2

2.3) หานิเสธของผลลัพธ์จากข้อ 2.2) โดยการใช้ตัวดำเนินการตรรกะ NOT ระดับบิต

2.4) นำผลลัพธ์จากข้อ 2.1) และ 2.3) มาดำเนินการตรรกะ AND ในระดับบิตต่อบิต เพื่อค้นหาค่า brand = "B" ซึ่งคำตอบที่ได้คือ ผลลัพธ์ของบิตแมปเวกเตอร์ที่มีค่าบิตเท่ากับ 1 (ON) ดังภาพประกอบ 3-13(d)

3) นำผลลัพธ์ที่ได้จากข้อ 1.9) และ 2.4) มาดำเนินการตรรกะ AND ในระดับบิตต่อบิต

4) คำตอบที่ได้ คือ ผลลัพธ์จากข้อ 3) ที่มีค่าบิตเท่ากับ 1 (ON) ในที่นี้คือเรคอร์ดที่ 5 ดังภาพประกอบ 3-13(e)



โดยที่ (a) ผลลัพธ์จากการ $I^3 \wedge \overline{I^4}$ เพื่อค้นหาค่า A=3

(b) ผลลัพธ์จากการ $\overline{I^7 \vee I^0}$ เพื่อค้นหาค่า A=14

(c) ผลลัพธ์จากการค้นหาค่า A=3, 14 มาดำเนินการตรรกะ OR ในระดับบิตต่อบิต

(d) ผลลัพธ์จากการ $I^1 \wedge \overline{I^2}$ เพื่อค้นหาค่า brand = "B"

(e) ผลลัพธ์จากการค้นหาค่า A=3, 14 และค่า brand="B" มาดำเนินการตรรกะ AND ในระดับบิตต่อบิต คือ บิตที่ 5 มีค่าบิตเท่ากับ 1

ภาพประกอบ 3-13 การดำเนินการตรรกะระดับบิตของแอทริบิวต์ type และ brand ของดัชนีบิตแมปแบบช่วง

3.3.4 ข้อดีของดัชนีบิตแมปแบบช่วง

ประหยัดพื้นที่ได้มากกว่าดัชนีบิตแมปแบบพื้นฐาน กล่าวคือ ดัชนีบิตแมปแบบช่วงใช้พื้นที่ในการจัดเก็บดัชนีเพียงครึ่งหนึ่งของดัชนีบิตแมปแบบพื้นฐาน

3.3.5 ข้อจำกัดของดัชนีบิตแมปแบบช่วง

ไม่เหมาะกับแอทริบิวต์ที่มีค่าคาร์ดินอลิตี้สูง ๆ เพราะยังจัดว่าใช้พื้นที่มากอยู่ เมื่อเปรียบเทียบกับดัชนีบิตแมปแบบถัดไปที่จะกล่าวถึง คือ ดัชนีบิตแมปแบบเข้ารหัส

3.4 ดัชนีบิตแมปแบบเข้ารหัส (Encoded Bitmap Indexing)

ดัชนีบิตแมปแบบเข้ารหัส [10] ประกอบด้วย $\lceil \log_2 C \rceil$ บิตแมปเวกเตอร์และตารางการเทียบค่า โดยที่ดัชนีบิตแมปแบบเข้ารหัสจะมีการเข้ารหัสค่าของแอทริบิวต์ที่จะนำมาทำดัชนีเหมือนการลงรหัสแบบ Huffman Coding ที่มีความยาวเท่ากัน การสร้างดัชนีบิตแมปแบบเข้ารหัสจะต้องมีสิ่งต่อไปนี้คือ

RID	แอทริบิวต์ A
1	14
2	3
3	4
4	2
5	3
6	1
7	13
8	0
9	6
10	5

(a) เลือกเฉพาะแอทริบิวต์ A

$B^3B^2B^1B^0$
1110
0011
0100
0010
0011
0001
1101
0000
0110
0101

(b) ตัวอย่างตารางดัชนี

ค่าของแอทริบิวต์ A	$B^3B^2B^1B^0$
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

(c) ตัวอย่างตารางการเทียบค่า (Mapping Table)

ภาพประกอบ 3-14 ตัวอย่างดัชนีบิตแมปแบบเข้ารหัส บนแอทริบิวต์ A (มีค่า C = 15)

1. ตารางการเทียบค่า (Mapping Table) เป็นตารางที่มีไว้ เพื่อเก็บบันทึกข้อมูลการเข้ารหัสค่าที่เป็นไปได้ทั้งหมดของแอทริบิวต์ที่ถูกนำมาทำดัชนี ซึ่งแต่ละค่าจะแตกต่างกัน โดยจำนวนเรคอร์ดในตารางการเทียบค่า จะเท่ากับคาร์ดินอลิตี้ของแอทริบิวต์ที่ถูกนำมาทำดัชนี จำนวนบิตแมปเวกเตอร์ที่ใช้ในการเข้ารหัสแต่ละค่า จะเท่ากับ $\lceil \log_2 C \rceil$ และรูปแบบการเข้ารหัส คือ $B^{\lceil \log_2 C \rceil - 1} \ldots B^2 B^1 B^0$ เช่น จากตัวอย่างเดิม ภาพประกอบ 3-2(a) สามารถสร้างตารางการเทียบค่าได้ ดังภาพประกอบ 3-14(c)

2. ตารางดัชนี (Indexed Table) เป็นตารางที่มีไว้ เพื่อเก็บบันทึกค่าของแอทริบิวต์ที่ถูกทำดัชนี ในรูปแบบของการเข้ารหัส จำนวนเรคอร์ดของตารางดัชนี จะเท่ากับจำนวนเรคอร์ดของข้อมูลจริง ดังภาพประกอบ 3-14(b)

จากภาพประกอบ 3-14(b) แอทริบิวต์ A มีคาร์ดินอลิตี้เท่ากับ 15 ดังนั้นการเข้ารหัสแต่ละค่าจะต้องใช้บิตแมปเวกเตอร์ เท่ากับ $\lceil \log_2 15 \rceil = 4$ ซึ่งรูปแบบการเข้ารหัส คือ $B^3 B^2 B^1 B^0$ และค่าแต่ละบิตของบิตแมปเวกเตอร์จะใช้สัญลักษณ์ B^i แสดงค่าของตำแหน่งบิตที่ i ที่มีค่าเท่ากับ 1 และสัญลักษณ์ $\overline{B^i}$ แสดงค่าของตำแหน่งบิตที่ i ที่มีค่าเท่ากับ 0 เพื่อใช้เป็นฟังก์ชันในการเข้าถึงข้อมูล (Retrieval Boolean Function) เช่น แอทริบิวต์ที่มีค่าเป็น “2” มีการเข้ารหัสเป็น 0010 และฟังก์ชันในการเข้าถึงข้อมูลคือ $\overline{B^3}\,\overline{B^2} B^1 \overline{B^0}$

3.4.1 การค้นหาข้อมูลแบบค่าเท่ากัน

โดยการอ่านค่าที่จะค้นหานั้นจากตารางการเทียบค่า ว่ามีการเข้ารหัสบิตแมปเวกเตอร์ในรูปแบบอะไร แล้วไปอ่านบิตแมปเวกเตอร์ทั้ง $\lceil \log_2 C \rceil$ บิตแมปเวกเตอร์จากตารางดัชนี ตัวอย่างเช่น ต้องการค้นหาค่า A = 2 สามารถทำได้โดยการอ่านค่าการเข้ารหัสจากตารางการเทียบค่า ซึ่งจะมีการเข้ารหัสเป็น 0010 หลังจากนั้นทำการอ่านบิตแมปเวกเตอร์ $B^3 B^2 B^1 B^0$ เข้ามา เพื่อทำการค้นหาเรคอร์ดที่มี B^3, B^2 และ B^0 มีค่าเป็น 0 และ B^1 ที่มีค่าเป็น 1 ซึ่งผลลัพธ์ที่ได้ คือ เรคอร์ดที่ 4

3.4.2 การค้นหาข้อมูลแบบความเป็นสมาชิก

ดัชนีบิตแมปแบบเข้ารหัส ไม่สามารถดำเนินการตรรกะระดับบิตระหว่างบิตแมปเวกเตอร์ได้โดยตรง จะต้องทำการแปลงให้อยู่ในรูปของดัชนีบิตแมปแบบพื้นฐานก่อนจึงจะสามารถดำเนินการตรรกะระดับบิตระหว่างบิตแมปเวกเตอร์ได้ กล่าวคือ หลังจากค้นหาข้อมูลแบบค่าเท่ากันแล้ว ให้นำผลลัพธ์ไปเก็บไว้ในบิตแมปเวกเตอร์ชั่วคราวในรูปของดัชนีบิตแมปแบบพื้นฐาน แล้วหลังจากนั้นค่อยนำผลลัพธ์ของการค้นหาแต่ละค่ามาดำเนินการตรรกะ OR ในระดับบิตระหว่างบิตแมปเวกเตอร์ชั่วคราว

3.4.3 การดำเนินการตรรกะระดับบิตระหว่างบิตแมปเวกเตอร์ของต่างแอทริบิวต์

ดัชนีบิตแมปแบบเข้ารหัส ไม่สามารถดำเนินการตรรกะระดับบิตระหว่างบิตแมปเวกเตอร์ของต่างแอทริบิวต์ได้โดยตรง ต้องดำเนินการเก็บผลลัพธ์ของแต่ละแอทริบิวต์ไว้ในบิตแมปเวกเตอร์ชั่วคราวในรูปของดัชนีบิตแมปแบบพื้นฐานก่อน ทำนองเดียวกันกับการค้นหาข้อมูลแบบความเป็นสมาชิก

3.4.4 ข้อดีของดัชนีบิตแมปแบบเข้ารหัส

ประหยัดพื้นที่ในการจัดเก็บดัชนี คือ เท่ากับ $\lceil \log_2 C \rceil$ บิตแมปเวกเตอร์

3.4.5 ข้อจำกัดของดัชนีบิตแมปแบบเข้ารหัส

จำนวนบิตแมปเวกเตอร์ที่ใช้มีค่าเท่ากับ $\lceil \log_2 C \rceil$ ของแอทริบิวต์ที่จะทำการสร้าง จะเห็นว่าดัชนีแบบนี้ไม่เหมาะสำหรับการสอบถามข้อมูลแบบค่าเท่ากัน เพราะต้องอ่านมาทุกบิตแมปเวกเตอร์ ($\lceil \log_2 C \rceil$) ทุก ๆ ครั้งที่มีการสอบถามข้อมูล โดยเฉพาะอย่างยิ่งแอทริบิวต์ที่มีค่าคาร์ดินอลิตี้สูง ๆ นอกจากนี้ดัชนีบิตแมปแบบเข้ารหัสไม่สามารถดำเนินการตรรกะระดับบิตระหว่างบิตแมปเวกเตอร์ของต่างแอทริบิวต์ได้

จากที่กล่าวมาข้างต้น จะเห็นได้ว่า ดัชนีบิตแมปแต่ละแบบจะมีข้อดี ข้อจำกัด และเหมาะสำหรับการสอบถามข้อมูลที่แตกต่างกัน ดังตาราง 3-1

ตาราง 3-1 สรุปข้อดีข้อจำกัดของดัชนีบิตแมปแต่ละชนิด

ชนิดของดัชนีบิตแมป	ข้อดี	ข้อจำกัด
แบบพื้นฐาน	- เหมาะสำหรับแอทริบิวต์ที่มีค่าคาร์ดินอลิตี้ต่ำ ๆ - เหมาะสำหรับการสอบถามข้อมูลแบบค่าเท่ากันมากที่สุด	- เปลืองพื้นที่ในการจัดเก็บดัชนีมากที่สุด
แบบ Range	- เหมาะสำหรับแอทริบิวต์ที่มีค่าคาร์ดินอลิตี้ต่ำ ๆ	- เปลืองพื้นที่ในการจัดเก็บดัชนีรองลงมาจากดัชนีบิตแมปแบบพื้นฐาน
แบบช่วง	- ประหยัดพื้นที่ในการจัดเก็บดัชนีลงได้ครึ่งหนึ่งของดัชนีบิตแมปแบบพื้นฐาน	- ไม่เหมาะกับแอทริบิวต์ที่มีค่าคาร์ดินอลิตี้สูง ๆ เพราะจัดว่ายังใช้พื้นที่มากอยู่
แบบเข้ารหัส	- ประหยัดพื้นที่ในการจัดเก็บดัชนีได้มากที่สุด	- ใช้เวลานานในการค้นหาข้อมูล

จากตาราง 3-1 จะเห็นได้ว่า ดัชนีบิตแมปแบบพื้นฐานจะเหมาะสำหรับแอทริบิวต์ที่มีคาร์ดินอลิตี้ต่ำ ๆ และเหมาะสำหรับการสอบถามข้อมูลแบบค่าเท่ากันมากที่สุด แต่ก็เปลืองพื้นที่ในการจัดเก็บ ดัชนีบิตแมปแบบเข้ารหัส ประหยัดพื้นที่ในการจัดเก็บดัชนีได้มากที่สุด แต่ใช้เวลานานในการค้นหาข้อมูล ดัชนีบิตแมปแบบช่วงประหยัดพื้นที่ในการจัดเก็บดัชนีลงได้ครึ่งหนึ่งของดัชนีบิตแมปแบบพื้นฐาน แต่ไม่เหมาะกับแอทริบิวต์ที่มีค่าคาร์ดินอลิตี้สูง ๆ เพราะจัดว่ายังใช้พื้นที่มากอยู่

จากที่กล่าวมาข้างต้นทั้งหมด จะเห็นได้ว่า ดัชนีบิตแมปที่ใช้เวลาน้อยที่สุดในการค้นหาข้อมูลทั้งแบบค่าเท่ากันและแบบความเป็นสมาชิก แต่จะเปลืองพื้นที่มากที่สุดในการจัดเก็บดัชนี ซึ่งก็คือ ดัชนีบิตแมปแบบพื้นฐาน ในทางกลับกัน ดัชนีบิตแมปที่สามารถประหยัดพื้นที่ได้มากที่สุดในการจัดเก็บดัชนี แต่จะใช้เวลานานที่สุดในการค้นหาข้อมูลทั้งแบบค่าเท่ากันและแบบความเป็นสมาชิก ซึ่งก็คือ ดัชนีบิตแมปแบบเข้ารหัส

สำหรับดัชนีบิตแมปแบบช่วง จะประหยัดพื้นที่ในการจัดเก็บดัชนี และเวลาที่ใช้ในการค้นหาข้อมูลได้ระดับปานกลาง แต่จะไม่เหมาะกับแอทริบิวต์ที่มีค่าคาร์ดินอลิตี้สูง ๆ เพราะจัดว่ายังใช้พื้นที่มากอยู่

ดังนั้น เพื่อขจัดปัญหาเหล่านี้ไป งานวิทยานิพนธ์นี้จึงได้เสนอดัชนีบิตแมปแบบใหม่ เรียกว่า ดัชนีบิตแมปแบบกระจาย ซึ่งสามารถประหยัดพื้นที่ในการจัดเก็บดัชนีได้มากกว่าดัชนีบิตแมปแบบช่วง แต่ยังคงรักษาเวลาในการค้นหาข้อมูล ซึ่งรายละเอียดจะกล่าวต่อไปในบทที่ 4