# HyBiX: Hybrid Encoding Bitmap Index for Efficient Space and Query Processing Time

**Naphat Keawpibal**

**A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Computer Science (International Program)**
**Prince of Songkla University**
**2018**

# HyBiX: Hybrid Encoding Bitmap Index for Efficient Space and Query Processing Time

**Naphat Keawpibal**

**A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Computer Science (International Program)**
**Prince of Songkla University**
**2018**

**Thesis Title** HyBiX: Hybrid Encoding Bitmap Index for Efficient Space and Query Processing Time

**Author** Mr. Naphat Keawpibal

**Major Program** Computer Science (International Program)

---

| **Major Advisor** | **Examining Committee:** |
|---|---|
| ............................................................... | ............................................... Chairperson |
| (Asst. Prof. Dr. Sirirut Vanichayobon) | (Assoc. Prof. Dr. Ohm Sornil) |
| | ................................................. Committee |
| **Co-advisor** | (Asst. Prof. Dr. Sirirut Vanichayobon) |
| ............................................................... | ................................................. Committee |
| (Asst. Prof. Dr. Ladda Preechaveerakul) | (Asst. Prof. Dr. Ladda Preechaveerakul) |
| | ................................................. Committee |
| | (Asst. Prof. Dr. Wiphada Wettayaprasit) |

The Graduate School, Prince of Songkla University, has approved this thesis as partial fulfillment of the requirements for the Doctor of Philosophy Degree in Computer Science (International Program).

............................................................

(Prof. Dr. Damrongsak Faroongsarng)

Dean of Graduate School

This is to certify that the work here submitted is the result of the candidate's own investigations. Due acknowledgement has been made of any assistance received.

..................................................Signature
(Asst. Prof. Dr. Sirirut Vanichayobon)
Major Advisor

..................................................Signature
(Asst. Prof. Dr. Ladda Preechaveerakul)
Co-advisor

..................................................Signature
(Mr. Naphat Keawpibal)
Candidate

I hereby certify that this work has not been accepted in substance for any degree, and is not being currently submitted in candidature for any degree.

..................................................Signature
(Mr. Naphat Keawpibal)
Candidate

| | |
|---|---|
| **ชื่อวิทยานิพนธ์** | HyBiX: การลงรหัสดัชนีบิตแมปแบบไฮบริด สำหรับประสิทธิภาพด้านพื้นที่และเวลาการประมวลผลสอบถาม |
| **ผู้เขียน** | นายณภัทร แก้วภิบาล |
| **สาขาวิชา** | วิทยาการคอมพิวเตอร์ (นานาชาติ) |
| **ปีการศึกษา** | 2561 |

## บทคัดย่อ


การพัฒนาของเทคโนโลยีในปัจจุบันได้สร้างข้อมูลจำนวนมหาศาลขึ้นมา ซึ่งได้ก่อให้เกิดปัญหาในการจัดเก็บและเข้าถึงข้อมูลจำนวนมหาศาลดังกล่าว ดังนั้นเทคนิคในการจัดเก็บข้อมูลและการเข้าถึงข้อมูลจึงได้รับความสนใจและศึกษาเพื่อให้มีการจัดเก็บและเข้าถึงข้อมูลอย่างมีประสิทธภาพ ดัชนีบิตแมปเป็นวิธีการจัดทำดัชนีที่มีประสิทธิภาพและประสิทธิผลในการเรียกดูข้อมูลบนระบบที่มีสภาวะแวดล้อมแบบอ่านอย่างเดียว เนื่องจากสามารถดำเนินการการค้นหาได้รวดเร็วโดยใช้ตัวดำเนินการบูลีนต้นทุนต่ำบนดัชนีได้โดยตรงก่อนเข้าถึงข้อมูลจริง อย่างไรก็ตามข้อเสียของดัชนีบิตแมปคือขนาดของดัชนีที่มีขนาดใหญ่ขึ้นเมื่อสร้างบนแอตทริบิวต์ที่มีคาร์ดินอลิตี้สูง วิทยานิพนธ์นี้เสนอดัชนีบิตแมปที่มีการลงรหัสรูปแบบใหม่ซึ่งเรียกว่า ดัชนีบิตแมปแบบไฮบริด (ดัชนีบิตแมป HyBiX) แนวคิดพื้นฐานของการสร้างดัชนีบิตแมปแบบไฮบริดคือการจัดกลุ่มค่าของแอตทริบิวต์ และการใช้แนวคิดพื้นฐานการลงรหัสของดัชนีบิตแมปรูปแบบอื่น ๆ ที่มีอยู่ เพื่อปรับปรุงประสิทธิภาพทั้งด้านเนื้อที่และเวลาที่ใช้ในการประมวลผลสำหรับการสืบค้นข้อมูลในลักษณะต่าง ๆ การจัดกลุ่มค่าข้อมูลของแอตทริบิวต์ช่วยอำนวยความสะดวกในการตอบแบบสอบถามที่มีการค้นหาช่วงของค่าข้อมูลที่ต่อเนื่องกัน จากผลการวิเคราะห์และทดลองเปรียบเทียบระหว่างดัชนีบิตแมปแบบไฮบริดกับดัชนีบิตแมปอื่น ๆ แสดงให้เห็นว่า เวลาที่ใช้ในการตอบแบบสอบถามแบบค่าเท่ากันเร็วขึ้น 79% และการตอบแบบสอบถามแบบช่วงเร็วขึ้น 82% นอกจากนี้ประสิทธิภาพของดัชนีบิตแมปแบบไฮบริดในแง่ของการแลกเปลี่ยนระหว่างประสิทธิภาพของพื้นที่กับเวลา (Space vs. time trade-off) อยู่ในลำดับที่สามที่ดีที่สุดสำหรับการสอบถามแบบค่าเท่ากัน และลำดับแรกที่ดีที่สุดสำหรับการสอบถามแบบช่วง เมื่อเปรียบเทียบกับดัชนีบิตแมปแบบอื่น ๆ

**Thesis Title** HyBiX: Hybrid Encoding Bitmap Index for Efficient Space and Query Processing Time
**Author** Mr. Naphat Keawpibal
**Major Program** Computer Science (International Program)
**Academic Year** 2018

## ABSTRACT

With an increasing availability of technology, an enormous amount of data has been generated. The problems in the storage and access have emerged. The consequent need for efficient techniques to store and access the information has been a strong resurgence of interest in the area of information retrieval. A bitmap-based index is an effective and efficient indexing method for operating information retrieval in a read-only environment. It offers improved query execution time by applying low-cost Boolean operators on the index directly, before accessing raw data. However, a drawback of the bitmap index is that the index size increases with the cardinality of indexed attributes. This dissertation then proposes a new encoding bitmap index, called HyBiX bitmap index. The basic concept of HyBiX bitmap index is the use of grouping idea with attribute values and the encoding design of existing encoding bitmap indexes in order to improve both storage demanded and execution time consumed with various queries. Particularly, the grouping of attribute values facilitates in answering a continuous range of query values. The experiment show that the HyBiX bitmap index takes 79% and 82% faster execution times than the Encoded bitmap index, for equality and range queries, respectively. Furthermore, the performance of HyBiX bitmap index in terms of space and time trade-off achieves the third-best and first-best as compare to existing encoding bitmap index, for equality and range queries, respectively.

# ACKNOWLEDGEMENTS

First of all, I would like to express my deep gratitude to my advisor, Assistant Prof. Dr. Sirirut Vanichayobon, for her support and guidance in the discussion on different opinions, for her patient and generously spent hours in correcting both my stylistic and scientific mistakes, and for a completed final research. She has helped me become better in conducting scientific research and scientific writing.

I would like to express my sincere thanks to my co-advisor, Assistant Prof. Dr. Ladda Preechaveerakul. The door to Assistant Prof. Dr. Ladda Preechaveerakul office always opens whenever I ran into a trouble spot or had a question about my study.

Besides my advisor, I would like to thank the rest of my dissertation committee: Associate Prof. Dr. Ohm Sornil and Assistant Prof. Dr. Wiphada Wettayaprasit, for their insightful comments and encouragement to accomplish my study.

My thanks go to PSU Ph.D scholarship financially supported by the Graduate School, Prince of Songkla University (PSU). I also gratefully acknowledge the PSU.GS. Financial Support for Thesis, Fiscal Year 2017 from Graduate School, PSU.

I also thank all lecturers and officers at the Department of Computer Science, Faculty of Science, PSU, who gave access to the Ph.D. office and research facilities. Without their precious supports, it would not be possible to conduct this research.

I would like to thank my friends studying in Master and Ph.D. degree at Department of Computer Science, Faculty of Science, PSU, who helped and encouraged me to overcome difficulties during my study.

Last but not least, my deepest thanks are going to my parents who always encourage me, believe in me, and give me the willpower to keep walking, and to my brother who always cheers me up and stands by me.

Naphat Keawpibal

# TABLE OF CONTENTS


| | Page |
|---|---|
| ABSTRACT (Thai) | **V** |
| ABSTRACT (English) | **VI** |
| ACKNOWLEDGEMENTS | **VII** |
| TABLE OF CONTENTS | **VIII** |
| LIST OF TABLES | **IX** |
| LIST OF FIGURES | **X** |
| LIST OF ALGORITHMS | **XI** |
| **CHAPTER** | |
| **1 EXISTING ENCODING BITMAP INDEXES** | **1** |
| 1.1 Range bitmap index | 1 |
| 1.2 Interval bitmap index | 3 |
| 1.3 Encoded bitmap index | 5 |
| 1.4 Scatter bitmap index | 7 |
| 1.5 Dual bitmap index | 10 |
| **BIBLIOGRAPHY** | **17** |
| **VITAE** | **19** |

# LIST OF TABLES

**Table** **Page**

1.1 A summarization of the number of bitmap vectors used for six encoding bitmap indexes . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12
1.2 A summarization of encoding bitmap index algorithms . . . . . . . . . . . 14

# LIST OF FIGURES

**Figure** **Page**

1.1 An example of the Range bitmap index: encoding of attribute $A$ with $C = 15$. 2

1.2 An example of the Interval bitmap index: encoding of attribute $A$ with $C = 15$. 3

1.3 An example of the Encoded bitmap index: encoding of attribute $A$ with $C = 15$. 6

1.4 An example of the Scatter bitmap index: encoding of attribute $A$ with $C = 15$. 8

1.5 An example of the Dual bitmap index: encoding of attribute $A$ with $C = 15$. 10

1.6 Six encoding schemes with $C = 15$, (• represents bit 1). . . . . . . . . . . . 12

# LIST OF ALGORITHMS

**Algorithm** **Page**

1.1 The creation of Scatter bitmap index . . . . . . . . . . . . . . . . . . . . . . 8

# CHAPTER 1

## Existing Encoding Bitmap Indexes

The encoding strategy is regarded as being one of the preferable strategies for improving bitmap index if the small numbers of bitmap vectors are generated and these bitmap vectors are primarily used in the bitwise operations to precisely answer queries without any decompression and additional processing. This chapter describes the existing encoding bitmap indexes, including Range bitmap index, Interval bitmap index, Encoded bitmap index, Scatter bitmap index, and Dual bitmap index.

### 1.1 Range bitmap index

Let $C$ be the attribute cardinality, which is the number of distinct values of that indexed attribute. The Range bitmap index formed on the range encoding scheme [1] produces a set of $C - 1$ bitmap vectors, says $R = \{R^0, R^1, \ldots, R^{C-2}\}$. The attribute values represented by bitmap vector $R^j$ are ranging from 0 to $j$. The encoding function for this bitmap index, for attribute value $v$, is given in Eq. (1.1)

$$R^j = \begin{cases} 1 & v \leq j \leq C - 2 \\ 0 & \text{Otherwise.} \end{cases} \tag{1.1}$$

Assume a domain of attribute $A$ given by table $T$ is $\{0, 1, 2, \ldots, 14\}$, as shown in Figure 1.1(a). The Range bitmap index therefore consists of 14 bitmap vectors since the cardinality of attribute $A$ is 15, says $\{R^0, R^1, R^2, \ldots, R^{13}\}$. Using Eq. (1.1) to represent attribute value '3', all bits from $R^3$ to $R^{13}$ are set to bit value 1; otherwise, the bits remained are set to bit value 0, and these are highlighted in Figure 1.1(b).

Querying both equality and range on the Range bitmap index uses the retrieval function in Eq. (1.2). Clearly, the equality queries deploy the first three conditions in Eq. (1.2), and the range queries therefore deploy the remaining conditions.

| | **A** |
|---|---|
| 1 | 3 |
| 2 | 9 |
| 3 | 14 |
| 4 | 8 |
| 5 | 10 |
| 6 | 3 |
| 7 | 4 |
| 8 | 0 |
| 9 | 12 |
| 10 | 5 |
| . | . |
| . | . |
| . | . |
| 100,000 | 2 |

**(a)** Table $T$

| $R^0$ | $R^1$ | $R^2$ | $R^3$ | $R^4$ | $R^5$ | $R^6$ | $R^7$ | $R^8$ | $R^9$ | $R^{10}$ | $R^{11}$ | $R^{12}$ | $R^{13}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 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 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**(b)** Range bitmap index

**Figure 1.1** An example of the Range bitmap index: encoding of attribute $A$ with $C = 15$.

For $0 \leq v_1 < v_2 \leq C - 1$,

$$v_1 \leq A \leq v_2 = \begin{cases} R^0 & v_1 = v_2 = 0, \\ R^{v_1} \oplus R^{v_1 - 1} & 0 < v_1 = v_2 < C - 1 \\ \overline{R^{C-2}} & v_1 = v_2 = C - 1 \\ \overline{R^{v_1 - 1}} & 0 < v_1 < C - 1, v_2 = C - 1 \\ R^{v_2} & v_1 = 0, 0 \leq v_2 \leq C - 1 \\ R^{v_2} \oplus R^{v_1 - 1} & \text{Otherwise.} \end{cases} \tag{1.2}$$

For example, to evaluate the equality query in the form of $A = 3$, using Eq. (1.2), the bitmap vector $R^2$ and $R^3$ were scanned, and then the bitwise-XOR operator is performed on them to answer this equality query, yields $R^2 \oplus R^3$. This query results the 1st and 6th rows.

To evaluate range query $1 \leq A \leq 4$, using Eq. (1.2), the bitmap vector $R^0$ and $R^4$ were scanned, and then the bitwise-XOR operator is performed on them to answer this range query, yields $R^4 \oplus R^0$.

**Advantage**

The Range bitmap index offers a good query performance for equality and range queries with low cardinality. Sometimes, the Range bitmap index scans only one bitmap vector to answer equality queries if the query value is equal to 0 or $C - 1$.

**Limitations**

The Range bitmap index decreases one bitmap vector from the Basic bitmap index, which still suffered storage problem with high cardinality attributes. The query performance of Range bitmap index is degraded with high cardinality attributes for both equality and range queries in point of views space and time trade-off.

## 1.2 Interval bitmap index

The Interval bitmap index based on the interval encoding scheme [2] reduces the number of bitmap vectors by half to $\{I^0, I^1, \ldots, I^{\lceil \frac{C}{2} \rceil - 1}\}$. Each bitmap vector $I^j$ represents the range of values between $j$ and $j + m$, where $m = \lfloor \frac{C}{2} \rfloor - 1$. The encoding function for the Interval bitmap index can be written as in Eq. (1.3). Figure 1.2 shows as an example of the Interval bitmap index for the data in Figure 1.2(a), with the 8 bitmap vectors, $\{I^0, I^1, \ldots, I^7\}$. On encoding attribute value '3', all bits between $I^0$ and $I^3$ are set to 1 and the remaining bits are set to 0.

| | $A$ |
|---|---|
| 1 | 3 |
| 2 | 9 |
| 3 | 14 |
| 4 | 8 |
| 5 | 10 |
| 6 | 3 |
| 7 | 4 |
| 8 | 0 |
| 9 | 12 |
| 10 | 5 |
| . | . |
| . | . |
| . | . |
| 100,000 | 2 |

**(a)** Table $T$

| $I^0$ | $I^1$ | $I^2$ | $I^3$ | $I^4$ | $I^5$ | $I^6$ | $I^7$ |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| . | . | . | . | . | . | . | . |
| . | . | . | . | . | . | . | . |
| . | . | . | . | . | . | . | . |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |

**(b)** Interval bitmap index

**Figure 1.2** An example of the Interval bitmap index: encoding of attribute $A$ with $C = 15$.

$$I^j = \begin{cases} 1 & v \leq j \leq j + m \\ 0 & \text{Otherwise.} \end{cases} \tag{1.3}$$

The retrieval functions in Eq. (1.4) – (1.6) checks for equality, one-side range, and two-side range queries, respectively.

For example, to evaluate the equality query in the form of $A = 3$, then the retrieval function of this query is $I^3 \wedge \overline{I^4}$. To evaluate range query in the form of $1 \leq A \leq 4$, the bitmap vector $I^1$ and $I^5$ were scanned, and then the bitwise-OR operator is performed on them, yields $I^1 \wedge \overline{I^5}$.

$$A = v = \begin{cases} I^0 & v = 0, m = 0 \\ \overline{I^0} & v = 1, C = 2 \\ I^1 & v = 1, C = 3 \\ I^v \wedge \overline{I^{v+1}} & v < m \\ I^1 \wedge I^0 & v = m, m > 0 \\ I^{v-m} \wedge \overline{I^{v-m-1}} & m < v < C - 1, m > 0 \\ \overline{I^{\lceil \frac{C}{2} \rceil - 1} \vee I^0} & v = C - 1. \end{cases} \tag{1.4}$$

For $0 < v < C - 1$,

$$A \leq v = \begin{cases} I^0 \wedge \overline{I^{v+1}} & v < m \\ I^0 & v = m \\ I^0 \vee I^{v-m} & m < v < C - 1. \end{cases} \tag{1.5}$$

For $0 < v_1 < v_2 < C - 1$,

$$v_1 \leq A \leq v_2 = \begin{cases} I^{v_1} \wedge \overline{I^{v_2} + 1} & v_2 < m \\ I^{v_1} \wedge I^0 & v_2 = m \\ I^{v_1} \wedge I^{v_2 - m} & v_2 < v_1 + m, v_1 < n \\ I^{v_1} & v_2 = v_1 + m, v_1 < n \\ I^{v_1} \vee I^{v_2 - m} & v_2 > v_1 + m, v_1 < m \\ I^{v_1} \vee I^{v_1 + 1} & v_2 = v_1 + m + 1, v_1 = m \\ I^{v_2 - m} \wedge \overline{I^{v_1 - m - 1}} & v_1 \geq n. \end{cases} \tag{1.6}$$

**Advantage**

The index size used by the Interval bitmap index is much smaller than that used by the Basic and Range bitmap index. In addition, the Interval bitmap index offers improved query performance against the Range bitmap index for both equality and range queries with low cardinality.

**Limitations**

The Interval bitmap index still produces many bitmap vectors with high cardinality, which is an impact on storage requirement. The performance of Interval bitmap index is degraded with high cardinality attributes for both equality and range queries in point of views space and time trade-off.

## 1.3 Encoded bitmap index

To our knowledge, the Encoded bitmap index [3] produces the smallest number of bitmap vectors, which consists of $\lceil \log_2 C \rceil$ bitmap vectors, say $\{E^0, E^1, \ldots, E^{\lceil \log_2 C \rceil - 1}\}$, and a mapping table which stores the binary patterns of all distinct attribute values. The attribute values are encoded with $\lceil \log_2 C \rceil$ bits in corresponding position of the bitmap vectors. Figure 1.3 shows an example of the Encoded bitmap index for the attribute with cardinality 15, which consists of 4 bitmap vectors, $E^0$, $E^1$, $E^2$, and $E^3$. Let us consider encoding the attribute value ‘3’. The binary pattern for attribute value ‘3’

| | $A$ |
|---|---|
| 1 | 3 |
| 2 | 9 |
| 3 | 14 |
| 4 | 8 |
| 5 | 10 |
| 6 | 3 |
| 7 | 4 |
| 8 | 0 |
| 9 | 12 |
| 10 | 5 |
| . | . |
| . | . |
| . | . |
| 100,000 | 2 |

**(a)** Table $T$

| $E^0$ | $E^1$ | $E^2$ | $E^3$ |
|---|---|---|---|
| 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| . | . | . | . |
| . | . | . | . |
| . | . | . | . |
| 0 | 0 | 1 | 0 |

| 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 |

**(b)** Encoded bitmap index with a mapping table

**Figure 1.3** An example of the Encoded bitmap index: encoding of attribute $A$ with $C$ = 15.

in the mapping table is '0011'. Therefore, the bitmap vectors are set, for this item, as $E^0 = 0$, $E^1 = 0$, $E^2 = 1$, and $E^3 = 1$, respectively.

To evaluate equality queries, the binary pattern of the query value is retrieved from the mapping table, and then the $\lceil \log_2 C \rceil$ bitmap vectors are jointly checked for this pattern in each row. Rows matching the target pattern across the $\lceil \log_2 C \rceil$ bits are returned as the answer to the query. Unfortunately, the traditional equality query processing used by the Encoded bitmap index takes a long time to answer the query because of the comparison of all $\lceil \log_2 C \rceil$ bitmap vectors. Accordingly, the E-EBI [4] is introduced to improve the traditional equality query processing in the Encoded bitmap index without comparison all bitmap vectors. In this algorithm, the bitmap vector can be performed bitwise-AND operation directly if the corresponding bit is set to 1. Otherwise, the negation of the bitmap vector is required before performing bitwise-AND operation. For example, to evaluate the equality query $A = 3$, the binary pattern for attribute value '3' is '0011', given by the mapping table in Figure 1.3(b). Therefore, the negation of bitmap vector $E^0$ and $E^1$ is required before performing bitwise-AND operation. As a result, the retrieval function of this query is simply created as $\overline{E^0 E^1} E^2 E^3$.

For evaluating range queries, the retrieval function for the query is formed of Boolean expressions that apply bitwise-OR operators on the expressions to get the final result. For example, to evaluate range query $1 \leq A \leq 4$, the binary patterns for each item are retrieved from the mapping table, and transformed to Boolean expressions, yields $\overline{E^0E^1E^2}E^3$ for value 1, $\overline{E^0E^1}E^2\overline{E^3}$ for value 2, $\overline{E^0E^1}E^2E^3$ for value 3, $\overline{E^0}E^1\overline{E^2E^3}$ for value 4. Then, the bitwise-OR operators are used on them, yields $(\overline{E^0E^1E^2}E^3) \vee (\overline{E^0E^1}E^2\overline{E^3}) \vee (\overline{E^0E^1}E^2E^3) \vee (\overline{E^0}E^1\overline{E^2E^3})$. Furthermore, the generated retrieval function can be further reduced to optimize range query performance by utilizing Boolean minimization method, such as Quine-McCluskey algorithm [5, 6]. The reduced retrieval function by Quine-McCluskey algorithm is generated as $(\overline{E^0E^1}E^3) \vee (\overline{E^0E^1}E^2\overline{E^3}) \vee (\overline{E^0}E^1\overline{E^2E^3})$. Additionally, the improved algorithms for Encoded bitmap index were introduced for querying equality and range queries by using data mining techniques and parallel processing over large dataset [4, 7–9]. However, both equality and range queries on the Encoded bitmap index take long execution times, even though this bitmap index is effective from the space requirement point of view.

**Advantage**

The Encoded bitmap index requires the smallest number of bitmap vectors for all cardinalities, which is an efficiency in space requirement.

**Limitations**

The query execution time taken by Encoded bitmap index is undesirable for both equality and range queries. Even though the Encoded bitmap index uses the Boolean minimization method in range queries to reduce the complexity of the retrieval function, it considerably takes long processing times with range queries.

## 1.4 Scatter bitmap index

For Scatter bitmap index [10], the bitmap vectors are split into two groups, namely Z-group and L-group. The Scatter bitmap index uses $\lceil 2\sqrt{C} \rceil$ bitmap vectors. The Z-group contains $\lceil \frac{C}{m-1} \rceil + 1$ bitmap vectors, says $\{Z^0, Z^1, \ldots, Z^{\lceil \frac{C}{m-1} \rceil}\}$, while the L-group contains $m - 2$ bitmap vectors, say $\{L^1, L^2, \ldots, L^{m-2}\}$, where $m = \lceil \sqrt{C} \rceil + 1$.

The algorithm for creation Scatter bitmap index is shown in Algorithm 1.1. If the value '$v$' at i$^{\text{th}}$ row relates to $Z^{j-1}$ and $Z^j$ (or $L^k$ and $Z^j$), the bits in $Z^{j-1}$ and $Z^j$ (or $L^k$ and $Z^j$) at i$^{\text{th}}$ row are set to 1. Otherwise, they are set to 0. Figure 1.4 depicts

**Algorithm 1.1** The creation of Scatter bitmap index

**INPUT:** The cardinality and values of of the indexed attribute
**OUTPUT:** The scatter bitmap index

1: $m \leftarrow \lceil \sqrt{C} \rceil + 1$
2: **for** a value $'v'$ in each row **do**
3:     Initial all bits in the bitmap vectors of Z- and L-group to be 0
4:     $j \leftarrow \lfloor \frac{v}{m-1} \rfloor + 1$
5:     $k \leftarrow v \mod m - 1$
6:     **if** $k = 0$ **then**
7:         Set bit of $Z^{j-1}$ and $Z^j$ to be 1
8:     **else**
9:         Set bit of $L^k$ and $Z^j$ to be 1
10:     **end if**
11: **end for**

an example of Scatter bitmap index for an attribute of cardinality 15, which consists of 8 bitmap vectors, say $\{Z^0, Z^1, Z^2, Z^3, Z^4, L^1, L^2, L^3\}$, and $m = 5$. On encoding attribute value '3', the bits in $Z^1$ and $L^3$ are set to 1 and the remaining bits are set to 0, due to the values of $j$ and $k$ are 1 and 3, respective, corresponding to the 2nd and 11th steps in Algorithm 1.1. Then, the bits in $Z^1$ and $L^3$ are set to 1 and the remaining bits are set to 0.

| | $A$ |
|---|---|
| 1 | 3 |
| 2 | 9 |
| 3 | 14 |
| 4 | 8 |
| 5 | 10 |
| 6 | 3 |
| 7 | 4 |
| 8 | 0 |
| 9 | 12 |
| 10 | 5 |
| . | . |
| . | . |
| . | . |
| 100,000 | 2 |

**(a)** Table $T$

| $Z^0$ | $Z^1$ | $Z^2$ | $Z^3$ | $Z^4$ | $L^1$ | $L^2$ | $L^3$ |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| . | . | . | . | . | . | . | . |
| . | . | . | . | . | . | . | . |
| . | . | . | . | . | . | . | . |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |

**(b)** Scatter bitmap index

**Figure 1.4** An example of the Scatter bitmap index: encoding of attribute $A$ with $C = 15$.

Equality queries with Scatter bitmap index use the retrieval function in Eq. (1.7). For example, to answer the equality query $A = 3$. Using Eq. (1.7), the retrieval function of this query is $Z^1 \wedge L^3$.

$$A = v = \begin{cases} Z^{j-1} \wedge Z^{j} & k = 0 \\ Z^{j} \wedge L^{k} & \text{Otherwise.} \end{cases} \tag{1.7}$$

where: $m = \left\lceil \sqrt{C} \right\rceil + 1$

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

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

For evaluating range queries, two bitmap vectors associated with each query value can be dynamically created by using Eq. (1.7), and the retrieval is performed with bitwise-OR operations. For example, to answer the range query $1 \leq A \leq 4$, by using Eq. (1.7), the retrieval functions for representing each item are dynamically generated as $Z^1 \wedge L^1$ for value 1, $Z^1 \wedge L^2$ for value 2, $Z^1 \wedge L^1$ for value 3, $Z^1 \wedge Z^2$ for value 4. Then, the bitwise-OR operators are used on them, yields $(Z^1 \wedge L^1) \vee (Z^1 \wedge L^2) \vee (Z^1 \wedge L^2) \vee (Z^1 \wedge Z^2)$. Furthermore, the retrieval function can be further minimized, which impacts the numbers of bitmap vectors accessed and the numbers of Boolean operations used. The basis idea of Dual-simRQ [11] is modified and applied to improve query processing, especially range queries. Therefore, the final reduced retrieval function is generated as $\left(Z^1 \wedge (L^1 \vee L^2 \vee L^3)\right) \vee (Z^1 \wedge Z^2)$. Additionally, the data clustering technique was employed to optimize the query processing on the Scatter bitmap index by grouping the attribute values which is frequently queried [12], to improve query execution time used by Scatter bitmap index.

**Advantage**

The Scatter bitmap index requires the less space than the Basic, Range, and Interval bitmap indexes, except the Encoded bitmap index. The Scatter bitmap index is suitable for equality queries because of scanning two bitmap vectors.

**Limitations**

The query execution time used by Scatter bitmap index is slower than the Basic bitmap index for equality queries. For range queries, the query execution time used by Scatter bitmap index is undesirable. Therefore, the performance of Scatter bitmap index with range queries is poor in space vs. time trade-off point of view.

## 1.5 Dual bitmap index

In the Scatter bitmap index, one bitmap vector (i.e., $Z^0$) is used to represent one value, which wastes space. Improving the Scatter bitmap index, the Dual bitmap index [13] efficiently represents attribute values while using two bitmap vectors. The Dual bitmap index consists of $\lceil\sqrt{2C+0.25}+0.5\rceil$ bitmap vectors, say $\{D^0, D^1, \ldots, D^{\lceil\sqrt{2C+0.25}+0.5\rceil-1}\}$. The dual encoding function is given in Eq. (1.8).

$$D^j = \begin{cases} 1 & j = r \text{ and } j = s \\ 0 & \text{Otherwise.} \end{cases} \tag{1.8}$$

where: $hiC = \frac{n(n-1)}{2}$

$$r = \left\lceil \sqrt{2(hiC - v) + 0.25} + 0.5 \right\rceil$$

$$s = \left\lceil r - 1 - \left\lceil \left(v - \frac{(n-r)(n-r-1)}{2}\right) \bmod r \right\rceil \right\rceil$$

Figure 1.5 depicts an example of the Dual bitmap index for an attribute with cardinality 15, with 6 bitmap vectors, say $\{D^0, D^1, D^2, D^3, D^4, D^5\}$. Using Eq. (1.8), to represent attribute value '3', the bits in $D^1$ and $D^5$ are set to 1, while the remaining bits are set to 0.

| | A |
|---|---|
| 1 | 3 |
| 2 | 9 |
| 3 | 14 |
| 4 | 8 |
| 5 | 10 |
| 6 | 3 |
| 7 | 4 |
| 8 | 0 |
| 9 | 12 |
| 10 | 5 |
| . | . |
| . | . |
| . | . |
| 100,000 | 2 |

**(a)** Table $T$

| $D^0$ | $D^1$ | $D^2$ | $D^3$ | $D^4$ | $D^5$ |
|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 |
| . | . | . | . | . | . |
| . | . | . | . | . | . |
| . | . | . | . | . | . |
| 0 | 0 | 1 | 0 | 0 | 1 |

**(b)** Dual bitmap index

**Figure 1.5** An example of the Dual bitmap index: encoding of attribute $A$ with $C = 15$.

Evaluation of equality queries with the Dual bitmap index uses the retrieval function in Eq. (1.9). For example, to answer the equality query $A = 3$. Using Eq. (1.9), the retrieval function of this query is $D^5 \wedge D^1$.

$$A = v = D^r \wedge D^s \tag{1.9}$$

To answer range queries, the retrieval function can be dynamically created and performed bitwise-OR operators, similar to the case with Scatter bitmap index. For example, to answer the range query $1 \leq A \leq 4$, by using Eq. (1.9), the retrieval functions for representing each item are dynamically generated as $D^5 \wedge D^3$ for value 1, $D^5 \wedge D^2$ for value 2, $D^5 \wedge D^1$ for value 3, $D^5 \wedge D^0$ for value 4. Then, the bitwise-OR operators are used on them, yields $(D^5 \wedge D^3) \vee (D^5 \wedge D^2) \vee (D^5 \wedge D^1) \vee (D^5 \wedge D^0)$. In addition, the retrieval function can be minimized to reduce the scanning of bitmap vectors as well as the number of Boolean operations, which impacts the query execution time taken. Therefore, Dual-simRQ [11] was proposed to improve the query execution time with range queries. The reduced retrieval function generated by Dual-simRQ is $D^5 \wedge (D^3 \vee D^2 \vee D^1 \vee D^0)$.

**Advantage**

The Dual bitmap index requires the less space than the Basic, Range, Interval, and Scatter bitmap indexes, except the Encoded bitmap index. The performance of Dual bitmap index is better than the existing bitmap indexes in terms of space and time trade-off for equality queries.

**Limitations**

The query execution time used by Dual bitmap index is slower than the Basic bitmap index for equality queries. Furthermore, the query execution time with range queries used by Dual bitmap index is undesirable. Therefore, the performance of Dual bitmap index is degraded in space vs. time trade-off for range queries.

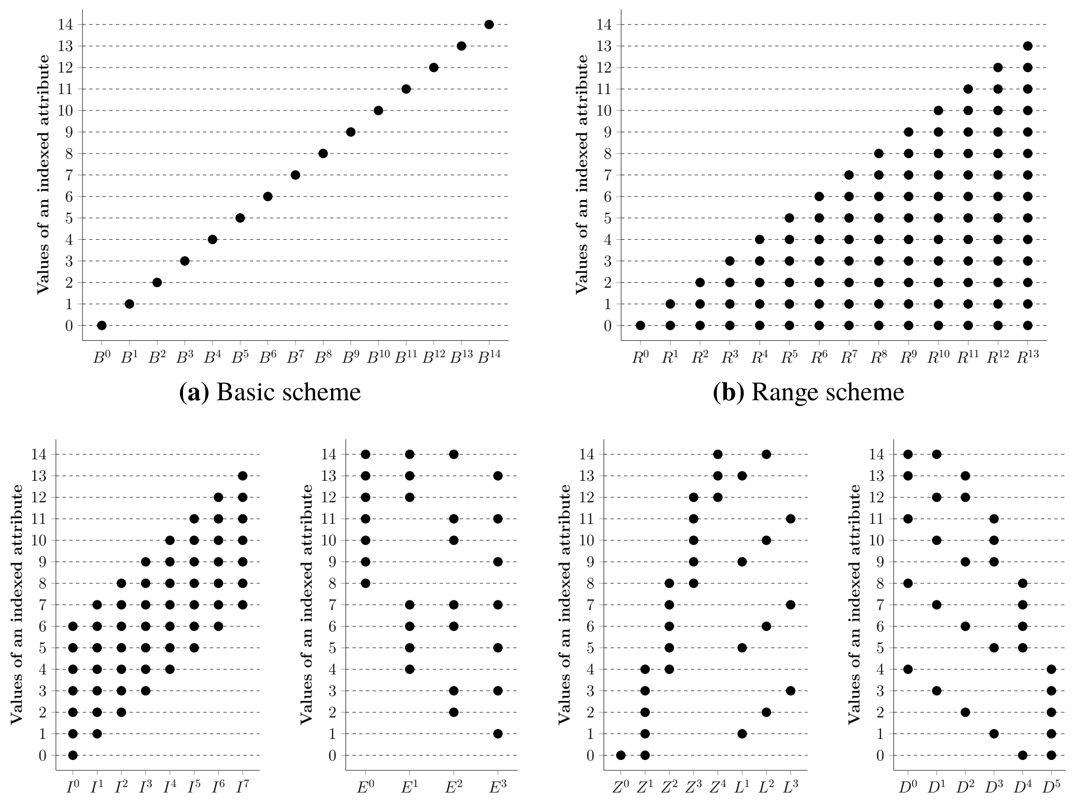
The numbers of bitmap vectors used for encoding bitmap indexes are summarized in Table 1.1. The Basic bitmap index uses $C$ bitmap vectors while the Range and Interval bitmap indexes decrease the number of bitmap vectors by one and half, respectively. The Encoded bitmap index uses $\lceil \log_2 C \rceil$ bitmap vectors. The Scatter and

Dual bitmap indexes utilize $\lceil 2\sqrt{C} \rceil$ and $\lceil \sqrt{2C + 0.25} + 0.5 \rceil$ bitmap vectors, respectively.

**Table 1.1** A summarization of the number of bitmap vectors used for six encoding bitmap indexes

| Bitmap index | The number of bitmap vectors used |
|---|---|
| Basic | $C$ |
| Range | $C - 1$ |
| Interval | $\lceil \frac{C}{2} \rceil$ |
| Encoded | $\lceil \log_2 C \rceil$ |
| Scatter | $\lceil 2\sqrt{C} \rceil$ |
| Dual | $\lceil \sqrt{2C + 0.25} + 0.5 \rceil$ |

Figure 1.6 illustrates the six encoding schemes with $C = 15$. Note that the black dots denote bit value 1. The Basic bitmap index uses 15 bitmap vectors, shown in Figure 1.6(a). The Range bitmap index uses 14 bitmap vectors while the Interval bitmap index uses 8 bitmap vectors to represent the attribute values, shown in Figure




**(a)** Basic scheme **(b)** Range scheme

**(c)** Interval scheme **(d)** Encoded scheme **(e)** Scatter scheme **(f)** Dual scheme

**Figure 1.6** Six encoding schemes with $C = 15$, (● represents bit 1).

1.6(b) and 1.6(c), respectively. The Encoded bitmap index uses 4 bitmap vectors, shown in Figure 1.6(d). Figures 1.6(e) and 1.6(f) show the encoding schemes for the Scatter and Dual bitmap indexes, which use 8 and 6 bitmap vectors, respectively.

This chapter described the characteristics of five encoding bitmap indexes, including Range, Interval, Encoded, Scatter, and Dual bitmap indexes. The Range and Interval bitmap indexes are efficient for equality and range queries by setting bit value 1 in the consecutive bitmap vectors to represent attribute values. However, those bitmap indexes suffer from high storage requirements, due to the large numbers of bitmap vectors. While the Encoded bitmap index uses the smallest number of bitmap vectors, it is inefficient with equality and range queries. The Scatter and Dual bitmap indexes can improve the performance for equality query processing, in terms of space and time trade-off, by accessing two bitmap vectors. Unfortunately, the range query processing is unsatisfactory. Table 1.2 summarizes the advantages and limitations of six encoding bitmap index algorithms.

As aforementioned, the existing encoding bitmap indexes are not be able to fully solve the problems of both space requirements and execution times with a variety of submitted queries. Therefore, the proposed encoding bitmap index, called HyBiX for Hybrid Encoding Bitmap Index, will be explained in the next chapter, to deal with those problems.

**Table 1.2** A summarization of encoding bitmap index algorithms

| Algorithm | Year | Method | Pros. | Cons. |
| --- | --- | --- | --- | --- |
| Basic bitmap index [14] | 1997 | - Formed on equality encoding<br>- Use one bitmap vector for representing one attribute value | - Easy to represent the data<br>- Suited for equality queries<br>- Suited for attribute with low cardinality | - Require a massive storage when build on attribute with high cardinality<br>- Consume long times for answering range queries |
| Range bitmap index [1] | 1998 | - Formed on range encoding<br>- Each attribute value is represented by the specific consecutive bitmap vectors<br>- Access 2 bitmap vectors to answer the queries | - Suited for equality and one-side range queries<br>- Suited for attribute with low cardinality | - Index size is dramatically increased when cardinality of indexed attribute is high |

Continued on next page

| Algorithm | Year | Method | Pros. | Cons. |
|---|---|---|---|---|
| Interval bitmap index [2] | 1999 | - Formed on interval encoding<br>- Each attribute value is represented by the specific consecutive bitmap vectors<br>- Access 2 bitmap vectors to answer the queries | - Index size is smaller than Basic and Range bitmap indexes<br>- Suited for equality queries, one-side, and two-side range queries<br>- Suited for attribute with low cardinality | - Index size is dramatically increased when cardinality of indexed attribute is high |
| Encoded bitmap index [3] | 1998 | - Formed on binary encoding<br>- Use a mapping table | - Index size is the smallest comparing with existing other encoding bitmap indexes<br>- Suited for attribute with high cardinality | - Take a long query execution time with both equality and range queries<br>- Need to look up at a mapping table and access all bitmap vectors |
| Scatter bitmap index [10] | 2006 | - Divide bitmap vectors into 2 groups<br>- Each indexed value is calculated and place into the group<br>- Use 2 bitmap vectors to represent each attribute value | - Index size is the smaller than the Basic, Range, and Interval bitmap indexes<br>- Suited for equality queries<br>- Use 2 bitmap vectors to answer equality queries | - Waste one bitmap vector to represent one value (i.e., $Z^0$)<br>- Take long times to answer range queries |

Continued on next page

| Algorithm | Year | Method | Pros. | Cons. |
|---|---|---|---|---|
| Dual bitmap index [10] | 2006 | - Improve space requirement of Scatter bitmap index<br>- Use 2 bitmap vectors to represent each attribute value | - Index size is smaller than the Basic, Range, Interval and Scatter bitmap index<br>- Suited for equality queries<br>- Use 2 bitmap vectors to answer equality queries | - Take long times to answer range queries |

# BIBLIOGRAPHY

[1] K.-L. Wu and P. S. Yu, "Range-based Bitmap Indexing for High Cardinality Attributes with Skew," in *Proceedings of The Twenty-Second Annual International Computer Software and Applications Conference (COMPSAC '98)*, Vienna, Austria, 1998, pp. 61–66.

[2] C.-Y. Chan and Y. E. Ioannidis, "An Efficient Bitmap Encoding Scheme for Selection Queries," *ACM SIGMOD Record*, vol. 28, no. 2, pp. 215–226, Jun. 1999.

[3] M.-C. Wu and A. P. Buchmann, "Encoded Bitmap Indexing for Data Warehouses," in *Proceedings of 14th International Conference on Data Engineering*, Florida, USA, 1998, pp. 220–230.

[4] A. Keawpibal, N. Wattanakitrungroj, and S. Vanichayobon, "Enhanced Encoded Bitmap Index for Equality Query," in *Proceedings of 2012 8th International Conference on Computing Technology and Information Management (ICCM)*, Seoul, South Korea, 2012, pp. 293–298.

[5] W. V. O. Quine, "The Problem of Simplifying Truth Functions," *The American Mathematical Monthly*, vol. 59, no. 8, pp. 521–531, 1952.

[6] E. J. McCluskey, "Minimization of Boolean Functions," *Bell Labs Technical Journal*, vol. 35, no. 6, pp. 1417–1444, 1956.

[7] G. R. Alam, M. Y. Arafat, M. Kamal, and U. Iftekhar, "A New Approach of Dynamic Encoded Bitmap Indexing Technique based on Query History," in *Proceedings of the 5th International Conference on Electrical and Computer Engineering (ICECE '08)*, Dhaka, Bangladesh, 2008, pp. 20–22.

[8] J. Sainui, S. Vanichayobon, and N. Wattanakitrungroj, "Optimizing Encoded Bitmap Index Using Frequent Itemsets Mining," in *Proceeding of 2008 International Conference on Computer and Electrical Engineering (ICCEE '08)*, Phuket, Thailand, 2008, pp. 511–515.

[9] N. Keawpibal, J. Duangsuwan, W. Wettayaprasit, L. Preechaveerakul, and S. Vanichayobon, "DistEQ: Distributed Equality Query Processing on Encoded Bitmap Index," in *Proceedings of the 2015 12th International Joint Conference on Computer Science and Software Engineering, (JCSSE)*, Songkhla, Thailand, 2015, pp. 309–314.

[10] S. Vanichayobon, J. Manfuekphan, and L. Gruenwald, "Scatter Bitmap: Space-Time Efficient Bitmap Indexing for Equality and Membership Queries," in *Proceedings of 2006 IEEE Conference on Cybernetics and Intelligent Systems*, Bangkok, Thailand, 2006, pp. 6–11.

[11] N. Keawpibal, L. Preechaveerakul, and S. Vanichayobon, "Optimizing Range Query Processing for Dual Bitmap Index," *Walailak Journal of Science and Technology (WJST)*, vol. 16, no. 2, pp. 133–142, Feb. 2019.

[12] W. Weahama, S. Vanichayobon, and J. Manfuekphan, "Using Data Clustering to Optimize Scatter Bitmap Index for Membership Queries," in *Proceedings of 2009 International Conference on Computer and Automation Engineering (ICCAE 2009)*, Bangkok, Thailand, 2009, pp. 174–178.

[13] N. Wattanakitrungroj and S. Vanichayobon, "Dual Bitmap Index: Space-time Efficient Bitmap Index for Equality and Membership Queries," in *Proceedings of 2006 International Symposium on Communications and Information Technologies (ISCIT)*, Bangkok, Thailand, 2006, pp. 568–573.

[14] P. O'Neil and D. Quass, "Improved Query Performance with Variant Indexes," *ACM SIGMOD Record*, vol. 26, no. 2, pp. 38–49, Jun. 1997.

# VITAE

**Name** Mr. Naphat Keawpibal

**Student ID** 5710230025

**Educational Attainment**

| Degree | Name of Institution | Year of Graduation |
|---|---|---|
| Master of Science (Computer Science) | Prince of Songkla University | 2012 |
| Bachelor of Science (Computer Science) with first class honor | Prince of Songkla University | 2010 |

**Scholarship Awards**

PSU.GS financial support for thesis, Fiscal year 2017 from Graduate School, Prince of Songkla University, 2017 – 2018.

PSU-PhD scholarship from Prince of Songkla University, 2014 – 2019.

Research Assistant Scholarship supported by Faculty of Science, Prince of Songkla University, 2010 – 2012.

**Work – Position and Address**

Lecturer at Department of Information Technology Business, Prince of Songkla University, Surat Thani campus, Surat Thani, Thailand, May 2013 – October 2013.

Lecturer at Department of Computer Science, Suratthani Rajabhat University, Surat Thani, Thailand, May 2012 – April 2013.

**List of Publications**

- Naphat Keawpibal, Ladda Preechaveerakul, Sirirut Vanichayobon, "HyBiX: A Novel Encoding Bitmap Index for Space- and Time-Efficient Query Processing", *Turkish Journal of Electrical Engineering & Computer Sciences*, Vol. 27, No. 2, 2019, pp. 1504–1522. doi: 10.3906/elk-1807-277.

- Naphat Keawpibal, Ladda Preechaveerakul, Sirirut Vanichayobon, “Optimizing Range Query Processing for Dual Bitmap Index”, *Walailak Journal of Science and Technology (WJST)*, Vol. 16, No. 2, 2019, pp. 133–142.

**List of Proceedings**

- Naphat Keawpibal, Jarunee Duangsuwan, Wiphada Wettayaprasit, Ladda Preechaveerakul, Sirirut Vanichayonon, “DistEQ: Distributed Equality Query Processing on Encoded Bitmap Index”, in *2015 12th International Joint Conference on Computer Science and Software Engineering (JCSSE 2015)*, Hat Yai, Thailand, 2015, pp. 309–314. doi: 10.1109/JCSSE.2015.7219815.

- Amorntep Keawpibal, Niwan Wattanakitrungroj, Sirirut Vanichayonon, “Enhanced Encoded Bitmap Index for Equality Query”, in *2012 8th International Conference on Computing Technology and Information Management (NCM and ICNIT)*, Seoul, South Korea, 2012, pp. 293–298.