Intelligent Automation And Soft Computing, 2019
Copyright © 2019, TSI® Press
Vol.25,no.4,683-689

SI bitmap index and optimization for membership query

Shu Gao[a,b] Zhen Wang[a] Liangchen Chen[a]
[a]School of Computer Science,Wuhan University of Technology, Wuhan, 430063, China;
[b]Hubei Key Laboratory of Transportation Internet of Things, Wuhan University of Technology, Wuhan, 430063, China.

ABSTRACT

The explosive growth of data produced by internet of things has contributed to the abundance of data. Since then, efficient indexing and querying techniques for data retrieval has become a major challenge. Bitmap index and its extension techniques, which involve a bit sequence that represents a specified property and indicates the data items that satisfies this property, are well-known methods to improve processing time for complex and interactive queries on the read-mostly or append-only data. This paper proposes an improved bitmap index technique, named Sliced-Interval Bitmap Index (SI Bitmap Index), which is efficient in both space and response time for Membership query. It also describes the method to optimize Membership query, based on SI Bitmap Index, in four steps. Experimental results indicate that SI Bitmap Index is space-saving as well as high efficiency on Membership query.

KEYWORDS: Bitmap Index, Data Mining, Interval Bitmap Index, Membership Query, Sliced Bitmap Index

1 INTRODUCTION

RAPID growth of Internet in the last decades has generated large volumes of data of all types and formats that are produced from Social media and IoT devices including sensors and embedded systems. The advent of the age of big data facilities the need for more efficient data indexing and querying techniques (Gani, Siddiqa, Shamshirband, & Hanum, 2016).To speed up the query processing, many index techniques are adopted, including B tree index, R tree index, bitmap index and so on. The bitmap index (O'Neil, 1987) uses bitmap vectors to index data based on binary bit values, 0 and 1, which is space-saving as well as high efficiency to query data because it is easier to perform logical operations, such as AND, OR, and Not IN, etc., on bit values. In most applications, bitmap indexes are shown to perform better than tree-based index schemes, such as the B-tree or R-tree and their variants. However, problems may occur if bitmap index is built on a high cardinality attribute. To exploit the advantages of bitmap index and overcome its shortcomings, this paper proposes a new bitmap index technique named Sliced-Interval Bitmap Index (SI Bitmap Index for short), which is space-time and efficiency, especially for Membership query, and also describes the method to optimize Membership query based on SI Bitmap Index. Results have shown that SI Bitmap Index is more space-time efficient than Simple Bitmap Index, Interval Bitmap Index and Sliced Bitmap Index, and the optimized Membership query processing algorithm is highly effective. Moreover, it can be also applied for both Equality query and Range query.

Structure of this paper is as follow. Section 2 reviews current bitmap index techniques and their characteristics. Section 3 introduces SI Bitmap Index and detailed method to optimize Membership query. Section 4 provides comparison between SI and other bitmap index techniques in space and query-time. Section 5 gives the conclusion and future work.

2 RELATED WORK

BITMAP index is based on bulk index data stored as sequences of bits. Compared to other index methods, bitmap index has more flexible encoding schemes and takes less time to answer the query for an attribute with low cardinality i.e., sex attribute. However, bitmap index requires more space if it is built on a high cardinality attribute. To overcome this space problem, most of researchers are interested in reducing index size while improving query processing efficiency on attribute with high cardinality. Two approaches have been developed: 1) Bitmap Index Compression, and 2) Bitmap Index Extension (Chambi, Lemire, Kaser & Godin, 2016).

In the first approach, Most of the recently proposed compressed bitmap formats are derived from Oracle's

CONTACT Shu Gao gshu418@163.com

BBC (Antoshenkov, 1995) and use run-length encoding (RLE) for compression: Enhanced Word Aligned Hybrid (EWAH) (Lemire, Kaser, & Aouiche, 2010), Compressed 'n' Composable Integer Set (Concise) (Colantonio & Pietro, 2010), Variable Length Compression (VLC) (Corrales, Chiu, & Sawin, 2011), COMPressed Adaptive indeX (COMPAX) (Fusco, Stoecklin & Vlachos, 2010), Variable Aligned Length-Word Aligned Hybrid (VAL-WAH) (Guzun, Canahuate, Chiu & Sawin, 2014), and so on. These 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 Simple Bitmap Index (O'Neil, 1987) , Interval Bitmap Index (Chan & Ioannidis, 1999), Sliced Bitmap Index (Chan & Ioannidis, 1998), Dual Bitmap Index (Wattanakitrungroj & Vanichayobon, 2006). The three indexes will be introduced in the following because SI is related to them. And our research focuses on the Bitmap Index Extension because the performance of this approach in term of space-time trade-off is better than the compression approach.

Given a data set T of N records, and T has an attribute Y with cardinality C. A bitmap index is essentially a collection of bitmaps. The size of each bitmap is equal to the cardinality of the indexed relation, and the i^{th} bit corresponds to the i^{th} record. In the simplest bitmap index design, the i^{th} bit of a bitmap associated with value v_i is set to 1 if and only if the i^{th} record has a value v_i for the indexed attribute.To reduce the space complexity and the query response time, many other bitmap index techniques have been introduced. Unlike Simple Bitmap Index, Interval Bitmap Index uses less bitmaps to store the index. It consists of $\lceil C/2 \rceil$ bitmaps $\aleph = \{ I^0, I^1, \dots, I^{\lceil C/2 \rceil - 1} \}$, where each bitmap $I^j = [j, j+m]$,and $m = \lfloor C/2 \rfloor - 1$. Benefitted by its index structure, it works efficiently for both Equality query and Range query (Chan & Ioannidis, 1999). Sliced Bitmap Index is more space-saving than other bitmap index techniques. It focuses on the method that decomposes the attribute value v by some bases. At first, the index should choose a base set $<b_1, b_2 \dots b_n>$, then present value by base set with a coefficient $<c_1, c_2 \dots c_n>$, and make sure that $b_1 * c_1 + b_2 * c_2 + \cdots + b_n * c_n = v$. Sliced Bitmap Index slices the storage space into many dimensions; any kind of bitmap index can be used into each dimension. Therefore, Sliced Bitmap Index is more space-saving and can be mixed with other bitmap index techniques. However, query efficiency needs to be optimized to make sure the advantages of bitmap index can fully be exploited (Chan & Ioannidis, 1998). Table 1 gives the *Y*'s values and indexes presentation (attribute *Y*'s cardinality is 9).

Table 1. *Y*'s Values and its Different Bitmap Indexes

(a) Values of Y

Y
0
4
7
8

(b) Simple Bitmap Index

B_0	B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8
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	1	0
0	0	0	0	0	0	0	0	1

(c) Interval Bitmap Index

B_0	B_1	B_2	B_3	B_4
1	0	0	0	0
0	1	1	1	1
0	0	0	0	1
0	0	0	0	0

(d) <3,3> Sliced Bitmap Index

First dimension index			Second dimension index		
H_0	H_1	H_2	L_0	L_1	L_2
1	0	0	1	0	0
0	1	0	0	1	0
0	0	1	0	1	0
0	0	1	0	0	1

Various bitmap indexes have been designed for different query types, including Equality queries, Range queries, Aggregation queries, and so on. However, as there is no overall best bitmap index over all kinds of queries, maintaining multiple types of bitmap indexes for an attribute may be necessary in order to achieve the desired level of performance. In this paper, we study the space-time tradeoff of bitmap indexes for Membership queries. And SI Bitmap Index

is proposed, which represents each attribute value using only two bitmap vectors, with each bitmap vector representing many attribute values. In addition, a novel Membership query processing method, which uses SI bitmap index, is introduced. This paper shows that the SI Bitmap Index is more efficient than the existing techniques for Membership queries from a space-time trade-off perspective.

3 SI BITMAP INDEX AND MEMBERSHIP QUERY OPTIMIZATION

3.1 SI Bitmap Index Structure

SI Bitmap Index combines Simple Bitmap Index, Interval Bitmap Index and Sliced Bitmap Index. Its structure is similar to that of Sliced Bitmap Index, however, using respectively Simple Bitmap index and Interval Bitmap Index in its two dimensions. As shown in Figure 1, the structure of SI Bitmap Index is a two-dimensional sliced structure. Chan C.Y (Chan & Ioannidis, 1998) has verified that the two-dimensional sliced structure can balance the storage efficiency and query performance, which is the reason why SI Bitmap Index adopts the two-dimensional structure. And it uses Interval Bitmap Index, in first dimension and simple bitmap index in the other.

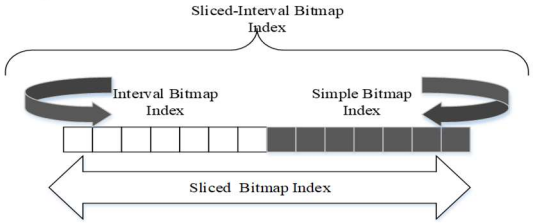

Figure 1. SI Structure

For example, <b_1,b_2> SI Bitmap Index is used on attribute Y, and b_1 and b_2 are cardinality of each dimension. For each attribute value v_i of Y, its value in each dimension is: $v_1 = v_i / b_2$, $v_2 = v_i \bmod b_2$. ($0 \le v_1 < b_1$, $0 \le v_2 < b_2$)

Besides, v_1 is indexed using Interval Bitmap Index while v_2 with Simple Bitmap Index.

SI Bitmap Index has two main features: it uses less space than Sliced Bitmap indexed because it uses Interval Bitmap Index in the first dimension, in contrast with Sliced Bitmap Index which uses Simple Bitmap Index in both the dimensions. Secondly, the experimental results have shown that SI Bitmap Index is high efficient on the Membership query. The detailed optimization methods are discussed in Section 3.2.

3.2 Optimization for Membership Query

Considering a relation R whose schema contains some attribute A taking values over a domain D, a Membership query is to retrieve all tuples in R with $A = x$ ($x \in D$), which has extensive application in science research and engineering practices.

To improve the efficiency of Membership query using SI Bitmap Index, an optimized method is proposed. The method consists of four steps, which are described as follows.

Step 1. Relationship Mining of Attribute Values

During the query, data mining technique can be used to group values' attribute that are frequently queried together. Apriori algorithm, which is a kind of association analysis technique, can produce frequent items during mining process based on the frequency of occurrence defined as *Support*. *Support* of these frequent items can be used in analyzing the relevance level of values of attribute Y.

For example, relationships of values of attribute Y need to be studied further under <d_1, d_2> SI Bitmap Index structure. Apriori algorithm is used for mining frequent d_2-items for a membership query on attribute Y. And it is ensured that mined frequent d_2-items have *Supports* bigger than a given *Support*. After mining process, items having higher correlation with each other are obtained. These d_2-items are the values of attribute Y, having stronger correlation. And then, a mapping table on attribute value and storage value is created. Therefore, Step 1 is the basis of optimization process.

Step 2. Conflict Resolution of Attribute Relationship

In Step 1, frequent d_2-items have been mined. These items have different *Supports*, and it is possible for them to overlap each other. To create the mapping table on attribute value and storage value in Step 3, it need to be ensured that these frequent d_2-items should not overlap with each other. Besides, if selected frequent d_2-items have maximum aggregated *Supports*, efficiency of Membership query will be improved vastly. The algorithm in Figure 2 describes how to get a maximum set of frequent k-items without overlaps.

In Figure 2, The algorithm MaxNoneOverlapFrequent-k-Items() has three parameters, in which word is the set of frequent k-items and *Support* is the *Support* of each k-item. And in (6), function MergeWord merges new frequent k-item into array v. For example, when new item $\{v_5,v_6\}$ merges with $\{\{v_1, v_2\},\{v_3, v_4\}\}$,a temp set container is produced which is $\{\{v_1,v_2,v_5,v_6\},\{v_3,v_4,v_5,v_6\}\}$. And in (7), function MergeList achieves the combination of original container v and new produced temp container. In above example, newly produced set container is $\{\{v_1,v_2\},\{v_3,v_4\},\{v_1,v_2,v_5,v_6\},\{v_3,v_4,v_5,v_6\}\}$.

Besides, function MergeWord should check the conflict. In case of $\{v_4,v_5\}$ merging with $\{v_1,v_2\}$ and $\{v_3,v_4\}$, merging $\{v_4,v_5\}$ with $\{v_1,v_2\}$ is non-conflict, while merging $\{v_3,v_4\}$ with $\{v_4,v_5\}$ is conflicting. So the produced temp set container is $\{\{v_1,v_2,v_4,v_5\}\}$. Hash method is used to check the conflict.

The time complexity of the algorithm MaxNoneOverlapFrequent-k-Items is O(2^n). But it can work efficiently because hash method is adopted to check conflict in MergeWord.

```
EleType[]MaxNoneOverlapFrequent-k-Items(String word [],Double support[],int n)
      // Input: Frequent k-items and their Supports
      //Output: Selected Frequent k-items
   { //The element in v[] is a set, and v[] is a container of sets
(1)   EleType v[];
(2)   v[0]= Ø;
(3)   Double maxValue=0.0;
(4)   for(int i=1;i<=n; i++)
(5)   {
   //merging new item words[i] with v[] produces new sets
(6)   EleType[] temp=MergeWord(v,i-1,words[i]);
   //merging new sets with original sets
(7)   v=MergeList(v, temp);
   //calculating the maximum set in v[]
(8)   Double value=Max(v, i);
(9)   if(value> maxValue)
(10)    maxValue=value;
(11)  }
(12)  Return v;
      }
```

Figure 2. **Algorithm Description of MaxNoneOverLapFrequent-k-Items**

Step 3. Creation of Mapping Table on Attribute Value and Storage Value (MT)

After step 2, we obtain non-conflict frequent d_2-items under $< d_1, d_2 >$SI Bitmap Index, having maximum aggregated Support. Suppose that frequent d_2-items obtained from step 2 are in set D_2, and C_{EX} is the set including all attribute values of Y, and C_{EX} is the set including the attribute values which belong to C_{ALL}, but not to D_2. Based on the hypothesis, MT is obtained by performing following steps.

Sorting frequent d_2-items in D_2 by their *Supports*.

For every frequent d_2-item in D_2, every attribute value is mapped into the storage value X (Initial X=0). X is incremented by 1 after mapping.

Repeating step 2) until the mapping process of every attribute value of every frequent d_2-item is finished.

For elements in C_{EX}, attribute values are ordered descendingly and are processed similarly as in step 2), until mapping processes of all the elements in C_{EX} is finished.

After mapping processes of attribute values of Y, MT can be created.

The following is a case study.

In this case, attribute Y has 30 different values, and <10, 3>SI Bitmap Index is established. Step1 and Step 2 is applied to get non-conflict frequent k-items, as shown in Table 2.

Table 2. **Frequent 3-items and their *Supports***

frequent 3-items	*Support*
v_1,v_4,v_5	23%
v_2,v_6,v_8	20%
v_{24},v_{25},v_{26}	18%

The mapping table of $\{v_1,v_4,v_5\}$ is shown in Table 3 and Table 4.

Table 3. **Attribute Values and their Mapping**

Attribute value	Storage value	First dimension value	Second dimension value
v_1	0	0/3=0	0 Mode 3=0
v_4	1	1/3=0	1 Mode 3=1
v_5	2	2/3=0	2 Mode 3=2
…			

Table 4. **Mapping Table of Attribute Values**

Attribute value	Storage value	First dimension indexes					Second dimension indexes		
		H_0	H_1	H_2	H_3	H_4	L_0	L_1	L_2
v_1	0	1	0	0	0	0	1	0	0
v_4	1	1	0	0	0	0	0	1	0
v_5	2	1	0	0	0	0	0	0	1
…									

As shown in the Table 3 and Table 4, the attribute values are mapped into the storage values, and then the storage values are calculated as index in the first dimension and second dimension. The index in first dimension adopts Interval Bitmap Index, and the index in second dimension Simple Bitmap Index.

Step 4. Optimization of Query Statement (SO)

In general, under <d_1, d_2>SI Bitmap Index, steps of processing Membership query $\{v_1, v_2, v_3,\ldots v_k\}$ are as follows.

1) Cleaning data.

Cleaning the query condition $\{v_1, v_2, v_3,\ldots v_k\}$, and removing repetitive elements. And making sure that attribute values in set $\{v_1, v_2, v_3,\ldots v_k\}$ are all unique.

2) Grouping the attribute values in query condition.

Based on mapping table, the attribute values are divided into different groups. The grouping method is as following.

$G(i) = \{v_a | i = v_s/d_2, v_s = \text{mapped}(v_a)\}$, where v_a is attribute value, and v_s is v_a's storage value, and d_2 is cardinality of second dimension, and i is the serial number of the group.

3) Optimizing the query according to the continuity of groups.

After grouping queried attribute values, appropriate optimization methods are chosen by analyzing continuity of groups, which are intra-group query optimization method and inter-group query optimization method, as discussed in following section.

4) Merging the query results of each group.

Query results are merged and the Membership query process is completed.

In $< d_1, d_2 >$SI Bitmap Index structure, *R(i)* is defined as the set of the attribute values in a query request which are grouped in $G(i)$, and $Q(H = i, L = j)$ as a query request which needs to query the records whose index in first dimension is *i* and in second dimension j. Query optimization includes intra-group query optimization and inter-group query optimization.

- Intra-group query optimization

Intra-group query means these groups are not "continuous" when the values in a query request are mapped into different groups, as shown in Figure 3.

Figure 3. Intra-group Query

For example, groups G_0 and G_2 in Figure 3, are not adjacent to each other. Under this condition, G_0 and G_2 are queried separately, and their results are merged in the end.

Providing the number of elements in *R(i)* equals to d_2, query method can be optimized as follows.

$Q(H = i, L = 0) \cup Q(H = i, L = 1) \cup \ldots \cup Q(H = i, L = d_2 - 1)$ i.e., $Q(H = i)$

Therefore, the response time is largely reduced because it only needs to query the first dimension index and access bitmaps vector two times, as SI Bitmap Index adopts Interval Bitmap Index structure in the first dimension.

Providing the number of elements in *R(i)* is *x*, and *x* is less than d_2, there are two query methods. The first one is to query each element in *R(i)* one by one, which needs to access bitmap vector 3**x* times. The second one is to query $G(i)$ as a whole at the first, and then query the elements in $\overline{R}(i)$ ($\overline{R}(i) = \{x | x \in G(i) \wedge x \notin R(i)\}$), i.e., using $Q(H = i) \cup \neg Q(H = i, L \in R'(i))$. In this way, the query requires to access bitmap vector 2+3*(d_2-*x*) times.

In case of 2+3*(d_2-*x*)<3**x*, i.e., *x*>(3* d_2+2)/6, the second query method ensures that less time is required, while in case of 2+3*(d_2-*x*)>3*x, i.e., *x*<(3* d_2+2)/6, the first query method is more efficient.

- Inter-group query optimization

Inter-group query means these groups are "continuous" when the values in a query request are mapped into different groups, as shown in Figure 4.

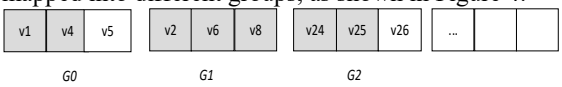

Figure 4. Inter -group Query

Providing that there is a continuous sequence $G(i), G(i+1), \ldots, G(i+k)$, and the number of vacant elements in the sequence is V ($V = \sum_{j=i}^{i+k} |\overline{R}(j)|$).There are two query methods. The first one is to query sequence $G(i), G(i+1), \ldots, G(i+k)$ as a whole at the first, and then query the vacant elements in the sequence, i.e., using $Q(i \le H \le i+k) \cup \neg Q(H = i, L \in \{x | x \text{ is the vacant element}\})$. The method needs to access bitmap vector 2+3*V times. The second one is to use the intra-group query optimization method for every group $G(i), G(i+1), \ldots, G(i+k)$ at the first, and then merge the results from every group. Providing that *M* is the times of accessing bitmap vector in this way.

In case of 2+3**V*<*M*, the first one would be more efficient, While in case of 2+3**V*>*M*, the second would be better.

Obviously, if groups are much close, inter-group query optimization will have a high performance.

4 PERFORMANCE STUDY

THIS section presents the experimental results of comparing space-time performance of four bitmap index techniques (Simple Bitmap Index, Interval Bitmap Index, Sliced Bitmap Index and SI Bitmap Index). The cardinality of the attribute *Y* is 30. The data set is about 2GB. And the experiments were run on a 2.40 GHz Intel® Xeon® with 16 GB main memory and the capacity of 1TB hard disk.

Different kinds of bitmap indexes on *Y* are created. Figure 5 shows the space comparison of different bitmap index techniques. It's obvious that SI Bitmap Index requires least space. The analysis can be seen in Table 5, which shows the space requirement of four bitmap index techniques.

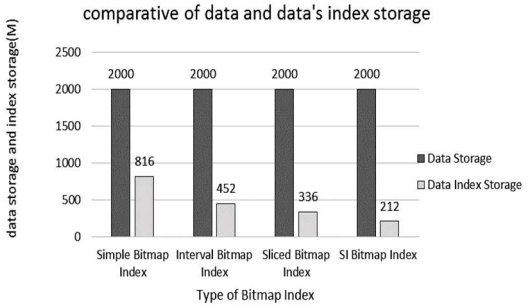

Figure 5. Space Comparisons of Different Bitmap Indexes

Table 5. A Comparative Study of Four Bitmap Indexes

Bitmap Index	Number of bitmap vectors used to represent each attribute value	Number of distinct values on 1 bitmap vector	Number of bitmap vectors used to represent an attribute with cardinality C (Space)
Simple	1	1	C
Interval	$\leq\left\lfloor\frac{C}{2}\right\rfloor$	$\left\lfloor\frac{C}{2}\right\rfloor$	$\left\lceil\frac{C}{2}\right\rceil$
Sliced	2	$\leq\left\lceil\sqrt{C}\right\rceil+1$	$\left\lceil 2\sqrt{C}\right\rceil$
SI	2	$\leq\left\lceil\sqrt{C}\right\rceil$	$\leq\left\lceil 2\sqrt{C}\right\rceil$

Figure 6 shows the response time comparison of Membership query of different bitmap index techniques, which shows that SI Bitmap Index has less response time than the other bitmap indexes. The reason is that SI Bitmap Index and the optimization method for Membership query have the following features.

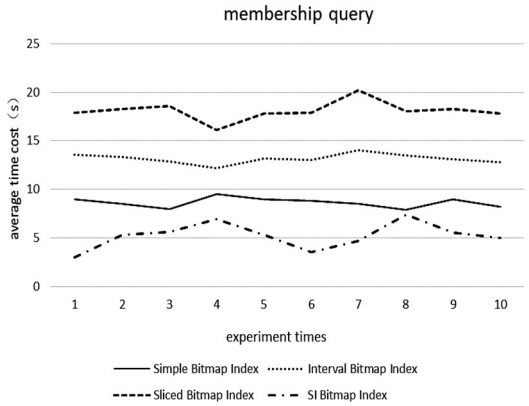

Figure 6. Running Time Comparison of Membership Query

1) Apriori algorithm is used for mining frequent d_2-items for a Membership query on attribute Y, which has an <d_1, d_2>SI Bitmap Index. After mining process, d_2-items, having higher correlation with each other, are obtained.

2) As shown in Step 3 of section 3, the mapped values of their storage values are equal in the first dimension and they are mutual exclusive and complementary in second dimension, i.e., they are 0,1,2… , if the attribute values belong to the same d_2-item.

3) As shown in Step 4 of section 3, to answer Membership query, the number of bitmap vectors scanned can be usually reduced by means of grouping the attribute values and optimizing the query based on the continuity of groups, as the number of the query on the bitmap vectors of second dimension are often decreased, even neglected.

In a word, data mining technique is used to group values' attribute that are frequently queried together, and then we take advantage of these characteristics to build a good encoding scheme to reduce the number of bitmap vectors scanned. Our experimental results confirm that the performance of SI Bitmap Index is better than those of the existing techniques for Membership query from the point of view of space-time trade-off.

5 CONCLUSION

BITMAP index techniques are widely used on read-mostly or append-only data with low cardinality. Simple Bitmap Index suits for Equality query while Interval Bitmap Index suits for Range query. But current bitmap index techniques, including Simple Bitmap Index, Interval Bitmap Index and Sliced Bitmap index, lose their advantages when confronted with large volume of data with large cardinality because of relative increase in indexing cost and query cost. The introduced Sliced-Interval Bitmap Index is useful because it is more efficient in space and time than the other three bitmap index techniques. Moreover, query optimization method, proposed in this paper, can further reduce the response time for Membership query. Experimental results in Section 4 have shown that SI Bitmap Index improves the performance and it outperforms other three bitmap indexes for Membership query from the point of space and query time.

For future work, we plan to reduce the time of data mining and investigate further applications in information retrieval.

6 REFERENCES

G. Antoshenkov. (1995 March). Byte-aligned bitmap compression. *Proceedings of Data Compression Conference.* Washington, DC, USA: IEEE Computer Society, 476-476.

S. Chambi, Lemire, D., Kaser, O., &Godin, R. (2016). Better bitmap performance with Roaring bitmaps. *Software: Practice and Experience.* 46 (5), 709–719.

C. Y. Chan & Ioannidis, Y. E. (1999). An Efficient Bitmap Encoding Scheme for Selection Queries. *Acm Sigmod Record*.28 (2), 215-226

C. Y. Chan & Ioannidis, Y. E. (1998). Bitmap index design and evaluation. *ACM SIGMOD Record*. 27(2): 355-366.

A. Colantonio, Pietro, R. D. (2010). Concise: Compressed ‘n’ Composable Integer Set. *Information Processing Letters*. 110 (16), 644-650

F. Corrales, Chiu, D., &Sawin, J. (2011 September). Variable length compression for bitmap indices. *Proceedings of the 22nd international conference on Database and expert systems applications.* Toulouse, France: Springer-Verlag. 381–395.

F. Fusco, Stoecklin, M.P, &Vlachos, M. (2010). NET-FLi: on-the-fly compression, archiving and indexing of streaming network traffic. *Proceedings of the VLDB Endowment*. 3(2), 1382–1393.

A. Gani, Siddiqa, A., Shamshirband, S., &Hanum, F. (2016). A survey on indexing techniques for big data: taxonomy and performance evaluation. *Knowledge and Information Systems.* 46(2), 241-284.

G. Guzun, Canahuate, G., Chiu, D., &Sawin, J. (2014 March). A tunable compression framework for bitmap indices. *Proceedings of IEEE 30th International Conference on Data Engineering*. Chicago, IL, USA: IEEE. 484–495.

D. Lemire, Kaser, O., &Aouiche, K. (2010). Sorting improves word-aligned bitmap indexes. *Data & Knowledge Engineering.* 69(1), 3–28.

P. E. O'Neil. (1987). Model 204 architecture and performance. *Proceedings of the 2nd International Workshop on High Performance Transaction Systems.* London, UK: Springer-Verlag. 40–59.

N. Wattanakitrungroj & Vanichayobon, S. (2006 October). Dual Bitmap Index: Space Time Efficient Bitmap Index for Equality and Membership Queries*, Proceeding of International Symposium on Communications and Information Technologies.* Bangkok, Thailand: IEEE. 568-573.

7 DISCLOSURE STATEMENT

NO potential conflict of interest was reported by the authors.

8 FUNDING

THIS work was supported by National Natural Science Foundation (No. 51479155), Natural Science Foundation of Hubei Province (No.2014CFB190).

9 NOTES ON CONTRIBUTORS

Shu Gao received the Ph.D. degree from Wuhan University of Technology, Wuhan, China, in 2001. She is currently a Professor with School of Computer Science, Wuhan University of Technology, China. Her research interests include data analysis, visual analytics and their application in Intelligent Transportation System.

Zhen Wang received the M.S. degree in computer science from the School of Computer Science, Wuhan University of Technology, Wuhan, China, in 2015. His research interests include data mining and IS development. He is now working in the Tencent.

Liangchen Chen is a PhD candidate in Computer Science and Technology from Wuhan University of Technology, China and a lecturer at China University of Labor Relations, and. His research interests include data analysis and Machine Learning.