Research on Composite Index Construction Method Based on Master-Slave Blockchain Structure

Guiyue Zhang, Yu Sui, Jiacheng Zhang, Chen Liu, Wenlong Hu, and Tingwei Chen(✉)

School of Information, Liaoning University, Shenyang 110036, China
twchen@lnu.edu.cn

Abstract. Blockchain is a new information processing technology that uses efficient cryptographic principles for the trustworthy storage of big data. With the exponential growth of the scale of data on the chain, the problems of low query efficiency and long traceability time of the existing blockchain system become more and more serious. To solve the above problems, this paper proposes a composite index construction method based on a master-slave blockchain structure. Firstly, a weight matrix is introduced to slice the whole master-slave blockchain based on the master chain structure; secondly, a master index construction method based on jump consistency hash is proposed for the master blockchain within each slice; finally, based on an improved Bloom filter, a slave composite index is constructed for each master block corresponding to the slave blockchain. Experimental results show that, compared with the existing methods, the proposed method can reduce the index construction time by 4.63% on average, improve the query efficiency by 6.71%, and reduce the memory overhead by 24.4%.

Keywords: Blockchain · Indexing · Slice

1 Introduction

Blockchain stores and verifies data through a blockchain data structure, and ensures the security of data transmission and access with cryptography [1]. It has the characteristics of high credibility, traceability, and decentralization [2], and can solve the problem of trust of the third party in data storage [3]. With the development of blockchain technology and the accumulation of data in various industries, the traditional single-chain blockchain system has been unable to meet the increasingly complex application scenarios. Master-Slave Blockchain (MSBC) structures (such as Spark Chain, etc.) have begun to attract the attention of experts and scholars in the field, and are gradually widely used in education, medical care, security [4], and other fields [5, 6].

Master-slave blockchain usually includes a master chain and slave chain, which are composed of a master block and slave block respectively, and each master block has only one slave chain. Master-slave blockchain structure can cope with the application of complex classification scenarios. For example, in the financial field, the master-slave

S. Yang and S. Islam (Eds.): APWeb-WAIM 2022 Workshops, CCIS 1784, pp. 75–85, 2023.
https://doi.org/10.1007/978-981-99-1354-1_8

blockchain is used to build a company blockchain system that generates financial activities. The master chain stores the information of financial institutions, and the corresponding slave chains store their transaction events, financial activities, and other data. Through the consensus mechanism of blockchain [7], the data cannot be tampered with.

With the continuous increase of data scale, the problems of low query efficiency and long tracing time of blockchain systems [8, 9] become more and more serious. However, the existing blockchain indexes are only suitable for a single chain structure, and the query efficiency and dynamic maintainability are poor. Therefore, how to establish an efficient and dynamically maintainable index structure based on a master-slave blockchain has become a hot and difficult point in the field of research.

To solve the above problems, this paper proposes a composite index construction method (MSCI) based on master-slave blockchain structure. The main contributions of this paper are as follows:

(1) A weight matrix is constructed by integrating node load, node credit, and network quality, and a WMS master-slave blockchain fragmentation algorithm is proposed to realize the dynamic fragmentation of the master-slave blockchain structure based on the characteristics of the main chain.
(2) On this basis, aiming at the problem that the existing blockchain index is not suitable for the master-slave chain structure, an index construction method based on jump consistency hash is proposed on the master chain. The key value of the master chain node is mapped with the index slot to realize the efficient query of the master chain information.
(3) On the slave chain, combining the data characteristics of blockchain, an index construction method based on an improved Bloom filter is proposed.
(4) Compared with the existing methods on different constraints and data sets, the effectiveness of the proposed method is verified.

2 Related Work

At present, many scholars have conducted in-depth research on the index construction of blockchain and achieved certain research results.

Vikram Nathan et al. [10] proposed a multi-dimensional memory read optimized index (Flood) method, which automatically adapts to specific data sets and workloads by jointly optimizing the index structure and data storage layout. However, this method needs to evaluate the query cost of the current layout regularly, and completely rebuild the index for different workloads. Huang et al. [11] proposed the B-tree-based indexing (EBTree) method, which supports real-time top-k, range, and equivalent search of Ethereum blockchain data. However, the index nodes of this method are stored separately in Level DB, and the query efficiency is greatly affected by the node size. Andreas Kipf et al. [12] proposed a single-channel learning index RS (RadixSpline) method, which can be constructed in a single transfer of sorted data, and it is friendly to most data sets, but this method will reduce the performance with the increase of data sets. Xing et al. [13] proposed an index method of account transaction chain based on subchain (SCATC), which divided the transaction chain into sub-chains, and converts the query

mode traversing the transaction chain into sub-chain query through pointers to reduce the computational overhead, but this method only improved the query efficiency of the longer account transaction chain and only optimized the query in plaintext. Noualhamdi et al. [14] put forward an extensible distributed index (ChainLink) method for large time-series data, designed a two-layer distributed index structure, and realized query operation with partition-level data reorganization. However, this method requires data reorganization locally, which can't guarantee data security. Gao Yuanning et al. [15] put forward a scalable learning index model (Dabble) based on the middle layer, which uses the K-Means clustering algorithm, divides the data set into K regions according to the data distribution, and uses a neural network to learn and train, respectively, and predicts the data position through the neural network model. However, this method requires retraining the model when updating the data set, and the timeliness of the model is poor, and the value of K has a great influence on the accuracy of the model.

To sum up, this paper makes an in-depth study on the index construction of the master-slave blockchain. Aiming at the shortcomings of existing index methods, and considering the space-time complexity and data security of index construction, a composite index construction method based on master-slave blockchain structure is proposed.

3 Blockchain Slice Method Based on Master-Slave Structure

To achieve efficient index construction and query processing of the master-slave blockchain structure, firstly, the whole master-slave blockchain is sliced based on the characteristics of the master chain, and each segment is given a weight. Based on this, the segmentation weight matrix of the whole master-slave blockchain structure is constructed, and the number of nodes in the slice is determined based on the weight matrix, which provides support for the indexing of the master chain and slave blockchain.

3.1 Construction of Weight Matrix

Let the number of nodes in the master-slave blockchain be x, the master-slave blockchain is divided into $y(y \ll x)$ slices, the *i-th* slice is $fi(i = 0, 1, 2, \cdots, y-1)$, and the weight of each slice is ω_i. The slice weight is determined by the weights of three dimensions: node load, node credit, and network quality, in which node credit and network quality are positively correlated with slice weight. Before calculating the slice weight, the units of the above three dimensions are not uniform, so normalization is needed.

Among them, the normalization formula of node load is:

$$d_{ij} = \left\lfloor \frac{1}{\log(d'_{ij} + 1) + 1} \times \frac{x}{y} \right\rfloor \tag{1}$$

The normalization formula of node credit and network quality is:

$$d_{ij} = \left\lfloor \frac{d'_{ij} + 1 - min}{max + 1 - min} \times \frac{x}{y} \right\rfloor \tag{2}$$

The weight of the *i-th* slice is ω_i, and the calculation formula is shown in formula (3).

$$\omega_i = \sum_{j-1}^{3} \omega_{ij} d_{ij} \tag{3}$$

d'_{ij} in formulas (1) and (2) represents the original values of node credit and network quality, d_{ij} is the normalized value, and *max* and *min* are the maximum and minimum values of this component, respectively. The ω_{ij} in formula (3) represents the weight of each component.

Let each element of the two-dimensional weight matrix $\boldsymbol{M}$ be composed of slice weights. After the weight of each slice is obtained, a two-dimensional weight matrix $\boldsymbol{M}[p][q]$ is constructed by using each slice's weight ωi (where $p \leq \sqrt{y}$, and p and q are integers). For any slice f, there is $M[f \,/\, p][f\,\%q] = \omega_f$, and the empty position in the matrix is set to 0.

3.2 Determining the Number of Nodes in a Slice Based on Weight Matrix

Determine the number of nodes in each slice based on the weight matrix in Sect. 3.1. Firstly, the slice weights in the matrix are linearly normalized. Secondly, all normalized slice weights are dispersed in proportion, and the interval of discrete proportional weights is set as [1, Q]. The final slice proportional weight Q-1 is obtained, that is, the slice will correspond to Q-1 nodes.

Number the nodes in the order of $\{0, 1, 2, \ldots, \sum_{i}^{x-1} \omega_i - 1\}$, and the node number corresponding to the *kth* slice is $[\sum_{i=0}^{k} \omega_i, \sum_{i=0}^{k+1} \omega_i - 1]$. After obtaining the node number, the slice number can be obtained by looking up the table.

Among them, the linear normalization formula is:

$$\omega'_i = \frac{\omega_i - \omega_{min}}{\omega_{max} - \omega_{min}} \tag{4}$$

ω_{min} in formula (4) is the minimum value of all slice weights, ω_{max} is the maximum value, and ω'_i is the normalized slice weight.

Based on the weight matrix in Sect. 3.1, a weighting matrix-based slice algorithm (WMS) is proposed, which is shown in Algorithm 1 and realizes the slice of the master-slave blockchain structure.

Algorithm 1. WMS Algorithm

```
WMS(key,r,M[p][q]){
    int i ,j ;
    for i = 0 to p-1
      for j = 0 to q-1
       num_row = sum(M[i][j])
       row = jumpSearch(key,r,num_row);
       /*Call the jump search function*/
     row = row*q;
     for i = 0 to q-1
       num_col = sum(M[i][j])
       colmun = jumpSearch(key,r,col);
       /*Call the jump search function*/
     return(row,colmun);
}
```

4 Methods for Constructing Composite Index

Because the data scale and information types stored in the master chain and slave chain of the master-slave blockchain structure are different, after slicing by the WMS algorithm, a composite index construction method based on jump consistency hash and improved Bloom filter is proposed to meet the query requirements of the master chain and slave chain. The schematic diagram of composite index construction is shown in Fig. 1.

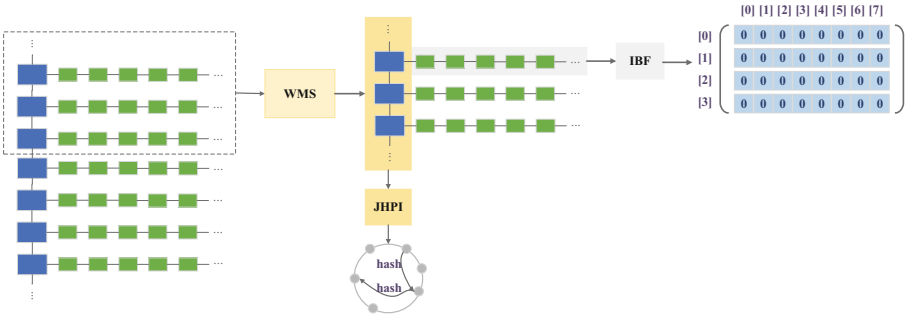

Fig. 1. Composite index construction schematic

4.1 Construction of Master Chain Index Based on Jump Consistency Hash

Based on the characteristics of data stored in the master chain, this paper introduces the jump consistent hash algorithm and proposes a jump consistent hashing-based master chain index construction method (JHPI) to realize the fast construction of the master chain index. Firstly, the number of index slots is determined according to the number of nodes in each master chain slice; secondly, the key value of each node is determined

according to the hash value of the master chain stored data; finally, the key value of each node and the number of index slots are input, and the master chain index is output.

When the number of nodes in a slice changes, the nodes jump in the index, and some nodes are remapped. Let the hash mapping function that produces the jump change be *ch*(*key*, *num_buckets*), *key* be the key value of nodes, and *num_buckets* be the number of slots, when the number of slots changes from n to $n + 1$, the slot where the node with $n/(n + 1)$ is located remains the same, that is, *ch*(*key*, *num_buckets*) = n-1, and $1/(n + 1)$ keys need to be remapped, that is, *ch*(*key*, *num_buckets*) = n.

The b is the result of the last jump of the node, and j is the result of the next jump. For any $i(i \in [b + 1, j - 1])$, the probability that the number of nodes does not jump is shown in formula (5).

$$P(i) = \frac{b+1}{b+2} \times \frac{b+2}{b+3} \times \cdots \times \frac{i-1}{i} = \frac{b+1}{i} \tag{5}$$

The construction method of the master chain index based on jump consistent hash is shown in Algorithm 2.

Algorithm 2. JHPI Algorithm

```
int JumpHashIndex(int key, int num_buckets,int[] s) {
      int b = 1, j = 0;
      while (j < num_buckets) {
        b = j;
        key = key * 2862933555777941757ULL + 1;
        /* Generate random numbers with key and r as
seeds*/
        j = (b + 1) * (double(1LL << 31) / double((key >>
33) + 1));
      }
      return b;
}
```

4.2 Construction of Slave Chain Index based on Improved Bloom Filter

In the process of constructing the dependent chain index, the data structure of the Bloom filter is reconstructed, and the column-based selection function is proposed to realize the dependent chain index construction method (IBF) based on the improved Bloom filter.

Firstly, the data structure is a two-dimensional array $A[p][q]$, where $p = 2^n$ (n is a positive integer), and the data length in q is lq, assuming that $lq = 32/64$ bits. The value is determined by the cache line length of general registers in the CPU to reduce memory access and improve query performance. Let the K hash functions of the improved Bloom filter be Hash(*key*), where the *key* is the key value of the node, and let the length of the elements that can be stored in an improved Bloom filter be *len*, and the calculation result of *len* is shown in formula (6).

$$len = p \times q = 2^n \times lq \tag{6}$$

After the data structure of the index is constructed, the index of the slave chain corresponding to each main block is constructed, and the specific steps are as follows:

Step1: Use the function of selecting columns, first map the element to the corresponding column, and the element will be at the position of the corresponding column;

Step2, obtaining the locus through K hash functions;

Step3: Set the corresponding site to 1, and the rest sites to 0.

Among them, the column selection function of Step1 is based on the SHA256 hash function, and the column selection function is obtained by modular operation. As shown in formula (7):

$$qv = v\%2^n \tag{7}$$

In Step2, the K hash functions are composed of K bitwise AND operations, which can be expressed as:

$$pv^i = v\& \sum_{j}^{i\times\frac{k'}{k}-1} 2^{i-1}(0 < i \leq k) \tag{8}$$

The v in formula (7) is the element in the Bloom filter, and qv is the column number obtained after column selection. After obtaining the column in which the element is located, it is determined that the element will be limited to the corresponding column during construction and query. The pv^i in formula (8) is the row number obtained by K times of operation of formula (8) in the column after the column number is obtained, that is, the corresponding position, and k' is the array length in the general Bloom filter. The slave chain index construction algorithm based on the improved Bloom filter is shown in Algorithm 3.

Algorithm 3. IBF_Construction

```
IBF_Construction(v,k, ) k'{
    int p = 0,q = 0;
  for(v){
    q = v %  2^n ;
    p = v &  Σ_j^{i×k'/k−1} 2^{i−1} ;
    output p,q;
  }
}
```

5 Experiments

The experimental environment of this paper is 16 servers with 1T storage space, 8G RAM, and a 4-core CPU. The servers communicate with each other through a high-speed network, and each server is equipped with an ubuntu 18.04 operating system. Two

different data sets are used for experimental verification. The first data set is the first 3,000,000 blocks in the public Ethereum network, and there are 15,362,853 transactions in the data set. The second data set is the Lognormal artificial data set. Lognormal data set samples 5 million pieces of non-duplicate data according to lognormal distribution (mean value is 0, variance is 2). In this section, we will verify the high efficiency and low memory advantages of MSCI from three aspects: index building time, query time, and memory consumption.

5.1 Comparison of Index Construction Time

To verify the efficiency of the MSCI method proposed in this paper in terms of index construction time, the EBTree method with improved blockchain structure and Dabble model method with the neural network will be compared respectively. In the EBTree method, the capacity of internal nodes is set to 128, and the capacity of leaf nodes is set to 16. The value of k in Dabble model is 100, and the number of nodes in a slice of the MSCI method is set to 100. In this section, the experiment will be divided into three specific situations as shown in Table 1.

Table 1. Index build time comparison.

	Dataset One		Dataset Two	
	MSCI	EBTree	MSCI	Dabble
Case 1	The slave block data is empty, and the number of master blocks is 500k, 1000k, 1500k, 2000k, 2500k, and 3000k respectively		The slave block data is empty, and the master block stores 2, 4, 6, 8, and 100,000 pieces of data	
Case 2	The master block data is empty, and the number of slave blocks is 500, 1000, 1500, and 2000		The master block data is empty, and the slave blocks store 500, 1000, 1500, and 2000 pieces of data	
Case 3	The data of master and slave blocks are not empty, the number of master blocks is 500k, 1000k, 1500k, 2000k, 2500k, and 3000k respectively, and the number of slave blocks is 1000		The data of master and slave blocks are not empty. The master block stores 2, 4, 6, 8, and 100,000 pieces of data, and the slave block stores 1,000 pieces of data	

As shown in Table 1, the experimental results of index construction time comparison are shown in Fig. 2 and Fig. 3.

As can be seen from Fig. 2 and Fig. 3, with the increase in data volume, compared with the existing methods, the index construction time of MSCI is optimized by about 4.79% and 4.47% respectively.

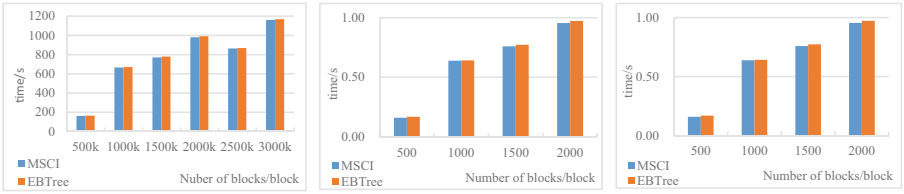

Fig. 2. Index build time comparison 1

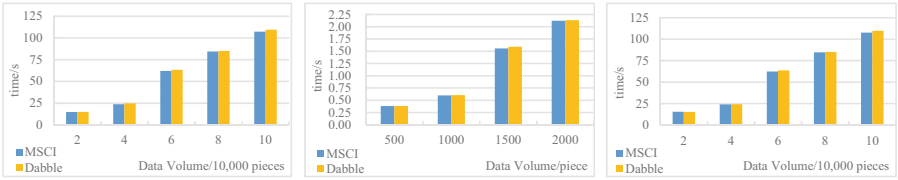

Fig. 3. Index build time comparison 2

5.2 Comparison of Query Time

In this experiment, to test the query performance of MSCI, we will compare the query response time of the EBTree method when the number of main blocks is 500k, 1000k, 1500k, 2000k, 2500k, and 3000k, and the number of subordinate blocks is 1000. Compare the query response time when Dabble method stores 10, 20, 30, 400, 500,000 pieces of data in the main block and 1000 pieces of data in the subordinate block. The experimental results are shown in Fig. 4.

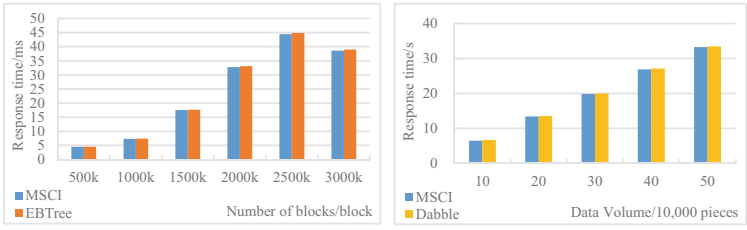

Fig. 4. Index query time comparison.

As can be seen from Fig. 4, compared with the existing methods, the index construction time of MSCI is optimized by about 7.44% and 5.89%, respectively, and when the data volume increases, the advantage of MSCI is more obvious than that of EBTree.

5.3 Comparison of Memory Consumption

The weight matrix constructed by MSCI in the slicing stage hardly takes up memory, and the master chain builds the index based on the jump consistent hash algorithm. Compared with the classical consistent hash, the jump consistent hash has almost no additional memory consumption, so the memory overhead in MSCI mainly considers the index construction of the slave blockchain. The false-positive of IBF is set to 0.0137 bits. The EBTree method rewrites the blockchain structure, and the memory consumption is mainly blocked data. Therefore, this section will compare the Dabble method. The experimental results are shown in Table 2.

Table 2. Memory Consumption Comparison.

Method	Memory Usage
Dabble	24 MB + 4 KB
MSCI	24 MB + 2.048 KB

As can be seen from Table 2, the Lognormal data set takes up 24MB of memory, while the neural network takes up 4KB of memory in Dabble method, while the IBF in MSCI still only takes up about 2.048KB of memory within the allowable range of false positives. Even if IBF and BF are in the same false positives, they can still keep the same order of magnitude.

6 Conclusion

With the wide application of blockchain technology, the traditional single-chain structure has gradually failed to meet the needs. Adding slave blockchain on the basis of the single-chain will make blockchain be applied to a wider range of fields, such as education chain and food chain. This paper proposes a composite index construction method for master-slave blockchain. Specifically, the whole master-slave blockchain structure is sliced according to the master chain, and the weight matrix is used to improve the maintainability of segmentation, which provides support for index construction. On this basis, aiming at the characteristics of different data scales between master chain and slave chain, the index construction method based on jump consistency hash algorithm and the index construction method of slave chain based on IBF is proposed to improve the query efficiency of the master-slave blockchain. Experimental results show that the proposed method has great advantages over the existing methods in terms of construction time, query efficiency, and memory consumption.

Acknowledgment. This study was supported by the Applied Basic Research Program of Liaoning Province (No. 2022JH2/101300250); the Digital Liaoning Smart Building Strong Province (Direction of Digital Economy) (No.13031307053000568); the National Key R&D Program of China (No. 2021YFF0901004); the Central Government Guides Local Science and Technology Development Foundation Project of Liaoning Province (No. 2022JH6/100100032); the Natural Science Foundation of Liaoning Province (No. 2022-KF-13-06).

References

1. Zhu, J.M., Zhagn, Q.N., Gao, S., Ding, Q.Y., Yuan, L.P., et al.: Blockchain-based trusted federated learning model for privacy protection. Chin. J. Comput. **44**(12), 2464–2484 (2021)
2. Li, C., Li, P., Zhou, D., et al.: A decentralized blockchain with high throughput and fast confirmation. In: 2020 {USENIX} Annual Technical Conference ({USENIX}{ATC} 2020), pp. 515–528 (2020)
3. Połap, D., Srivastava, G., Jolfaei, A., et al.: Blockchain technology and neural networks for the internet of medical things. In: IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 508–513. IEEE (2020)
4. Wei, S.J., Li, S.S., Wanag, J.H.: Cross-domain authentication protocols based on identity cryptosystems and blockchains. Chin. J. Comput. **44**(05), 908–920 (2021)
5. Bao, J., He, D., Luo, M., et al.: A survey of blockchain applications in the energy sector. IEEE Syst. J. **15**(3), 3370–3381 (2020)
6. Kalodner, H., Möser, M., Lee, K., et al.: {BlockSci}: design and applications of a blockchain analysis platform. In: 29th USENIX Security Symposium (USENIX Security 2020), pp. 2721–2738 (2020)
7. Huang, J., Kong, L., Chen, G., et al.: Towards secure industrial IoT: blockchain system with credit-based consensus mechanism. IEEE Trans. Industr. Inf. **15**(6), 3680–3689 (2019)
8. Sui, Y., Wang, W., Deng, X.: High throughput verifiable query method for blockchain-oriented off-chain database. J. Chin. Comput. Syst. **42**(6), 1304–1312 (2021)
9. Cai, L., Zhu, Y.C., Guo, Q.X., Zhang, Z., Jin, C.Q.: Efficient materialized view maintenance and trusted query for blockchain. J. Softw. **31**(3), 680–694 (2020)
10. Nathan, V., Ding, J., Alizadeh, M., Kraska, T.: Learning multi-dimensional indexes. In: 2020 ACM SIGMOD International Conference on Management of Data (SIGMOD 2020), 14–19 June 2020, Portland, OR, USA, 16 p. ACM, New York (2020)
11. XiaoJu, H., XueQing, G., ZhiGang, H., et al.: Ebtree: a b-plus tree based index for Ethereum blockchain data. In: Proceedings of the 2020 Asia Service Sciences and Software Engineering Conference, pp. 83–90 (2020)
12. Kipf, A., Marcus, R., van Renen, A., et al.: RadixSpline: a single-pass learned index. In: Proceedings of the Third International Workshop on Exploiting Artificial Intelligence Techniques for Data Management, pp. 1–5 (2020)
13. Xing, X., Chen, Y., Li, T., et al.: A blockchain index structure based on subchain query. J. Cloud Comput. **10**(1), 1–11 (2021)
14. Alghamdi, N., Zhang, L., Zhang, H., et al.: ChainLink: indexing big time series data for long subsequence matching. In: 2020 IEEE 36th International Conference on Data Engineering (ICDE), pp. 529–540. IEEE (2020)
15. Gao, Y.N., Ye, J.B., Yang, N.Z., Gao, X.F., Chen, G.H.: Middle layer-based scalable learned index scheme. J. Softw. **31**(3), 620−633 (2020)