International Journal of Advanced Technology and Engineering Exploration
ISSN (Print): 2394-5443 ISSN (Online): 2394-7454
Volume-2 Issue-5 April-2015

An Effective Approach for Indexed Data Access based on Linear Search Technique Using Reference Key Algorithm

Ashwini* **and Dileep M. R**
Department of Computer Science, Alva's College, A unit of Alva's Education Foundation, Moodbidri, Karnataka, India

Abstract

A robust and simple algorithm is implemented to search a record using the index dataset. In an RDBMS, the required dataset or records are accessed using the database name. In this proposed algorithm, the database name is used only once during the creation of index table, and hence forth onwards the database is accessed using the index table. Thus, database hiding concept is implemented to certain extent. The usage of index table reduces the access time complexity to a greater extent when applied to a large database. Security is provided to the database by checking for the authorization before the index table is accessed. The records are searched from database through the index table using the linear search. In this paper, a bi-level linear search is implemented using parallel processing system efficiently at a faster rate. Thus, time complexity is less compared to normal search method which is applied to a large database. Data retrieval is too efficient since the proposed approach operates on the address rather than the data records. The second attribute of the index table uses the concept of pointers. Pointers are the variables that stores the memory address of other variables. Using pointers in index table any record is accessible with time less than a second. Many factors have contributed to decrease the time complexity in accessing the database using index table. Initial processing is done using index table without database access, then the range of records related to the corresponding index is read from memory directly using the pointer variable. Thus, the proposed algorithm not only contributes in data hiding, implementing security, reduced time complexity and but also efficient memory usage.

Keywords

Database access, Indexed database access, linear search, pointers, reference key, time complexity.

*Author for correspondence

1. Introduction

Quick, flexible and efficient access to information collected from a long period of time is essential for increasing the effectiveness of an organization. A lot of data mining tools are available, it is the choice of the organization to decide which tool will best applicable to manage their data and achieve this goal. Out of many options are available, Relational Database Management Systems (RDMS) provide the only sensible choice for data management.

What is a relational database? A relational database is a collection of related information edited to required data, organized into separate tables. This will provide storage and easy access of related information in one place.

RDBMS stores the collection of related information in a continuous memory location. To query such a database requires to load the entire table onto the main memory, which may be inefficient to manage the data within the limited memory. As a solution being proposed in this paper is to access the database using indexed reference keys to load only the required part/ section of database for query processing. Thus efficiently managing the main memory with a cost of extra processing for managing the index table to the database.

In this paper, the algorithm has been proposed to access the database using pointers. The considered database is a table with large set of records stored in the central server. The records are arranged in ascending order using the Primary key. Thus, as the time passes the dataset keeps on growing. The primary key is allotted uniquely to each record in increasing order.

1.1 Literature Survey

Database application is a thrust research area in security management and enforcement applications as a prerequisite for database applications. Until now much research work has been done on database

applications and example-based techniques. Data hiding is one of the important feature that provides the security. The accessing are simple and easy to understand and implement. The only complication present in this methodology is that it requires extra space for storing index table. However, some of these methods are computationally expensive**.**

A. Fikes[1], 2010, mentioned regarding the Storage architecture and challenges. A. Labrinidis, et al [2], 2010, proposed the methodology for Caching and Materialization for Web Databases. Anup K. Sen et al. [3], 1989, devised method for Fast recursive formulations for best-first search that allows controlled use of memory. Hacigumus, et al [4],2004, proposed a methodology for Ensuring the Integrity of Encrypted Databases in the Database-as-a-Service Model. Hermann Keindl, et al [5], 1994, discovered the algorithm for Improvements on linear space search algorithms. J. Baker et al [6], 2011, invented a methodology for Megastore: Providing scalable, highly available storage for interactive services. J. Dean [7], 2010, described a procedure for the Evolution and future directions of large-scale storage and computation systems . Lohman, G., et al [8], 1984, developed the methods for Query Processing . Markl et al [9], 2003, proposed an An autonomic query optimizer for db2. M. Stonebraker [10], 2010, described the differences between SQL databases v. National Science Foundation [11], 2012, provided the information about the Core Techniques and Technologies for Advancing Big Data Science &Engineering. Oracle [12], 2011, gave the information on relational dbass . Pizzete, L. et al [13], 2012, provided the information on database in regards to Database as a Service: A Marketplace Assessment. S. A. Wright, et al [14], 2011,specified a methodology in Light-weight Parallel I/O Analysis at Scale. T. Kraska et al [15], 2013, proposed a technique in to maintain data consistency in MDCC: Multi-data center consistency. W. Hsu et al [16], 2004, provided the information on "The Performance Impact of I/O Optimizations and Disk Improvements.

In this paper, we present a systematic representation of the indexed dataset access with reference key using the linear search methodology as a searching technique. The rest of this paper is being organized as follows. Section 2 describes the Proposed Methodology. Section 3 presents the Proposed Algorithm. Section 4 provides the Experimental Results. Finally, we draw the conclusions in section 5.

2. Proposed Methodology

In the proposed methodology a separate index table is maintained with two attributes, a key which represents the primary key of the database table and the second attribute is a pointer variable which stores the address of the record in memory corresponding to the key. The entire database is divided to equal intervals of set of records, and the first record in the presented interval will be the key in the index table. For example a sample database with 100 records is divided into 10 intervals, thus every 1st record primary key value will be an entry in the index table. Thus the range of records for search criteria from the main table with n records will be reduced to m records (i.e from key to key + m (interval size) - 1 to key).This will point to the first record in the interval. Hence, by reducing the memory storage space required to load the data records to be searched or accessed to 1/mth of the whole database. Once the specified set of records are loaded into the primary memory, further the desired record is obtained by performing a linear search with the range of key to key + m (interval size) - 1 . The linear search algorithm is used in this proposed research, since it is simple to understand and implement.

Linear search or sequential search is one of the most basic algorithms used to perform a search operation by comparing each element with the search key. The search is performed beginning from to first record in sequence until it the desired record is found or the list is exhausted.

For a list with *n* items, the different cases are as follows:

(i) The best case is when the value is equal to the first element of the list (i.e., in the first comparison itself the desired record is found).

(ii) The worst case is when the value is not in the list (or occurs only once at the end of the list), in which case *n* comparisons are needed.

(iii) The time complexity for linear search to be performed on index table is O(n) and the second search requires O(m). Therefore overall time complexity required to search a record is O(n + m).

The database is accessed first through the index table of size *n*, thus expected no of comparisons at worst case is *n* and at the best case is *1*.
Next the dataset related to the index is searched for the search key record, therefore the no.of comparisons a worst case is *m* and at best case is again *1*.
Thus the overall no.of comparisons required are:

Best case → 1+1 = 2 comparisons (1)

i.e., first record is matched both at the index table and the dataset retrieved from the database.

Worst case → *n+m comparisons* (2)

i.e., total no. of records being compared in the tables, index and database.

In a DBMS the dataset records are accessed directly by using the table name. Once the database table name is known all the detailed records can be accessed. To implement the characteristics of the database hiding and security, the entire record access procedure is maintained by using the reference pointers i.e., direct memory location of the index key record. Pointers are the variables who store the address of other variables. Thus by using reference to direct memory location only the part of the database is accessed through the index table. Since through the reference variables are used to access the original location of the dataset and the copy of dataset is displayed, all operations which can be performed on a database can also be implemented through reference variables.

The linear search algorithm is used to search for the desired record in the provided, set by incrementing or decrementing the pointer value. Thus the name of the database is used only once, in the beginning during the initialization of pointers in the index table, rest of task is carried out using the attributes of main index table. Hence the database hiding quality is implemented. The reference Key used in the database is given in Figure 1.

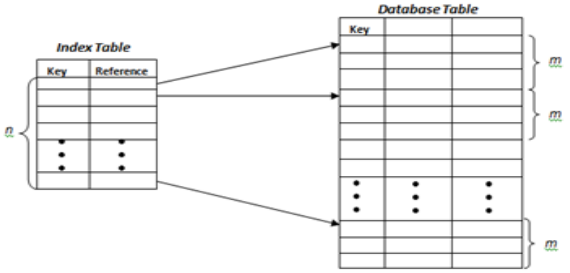

Fig.1: Reference Key used in Database Access

3. Proposed Algorithm

A huge Relational database with a incremental unique primary key. The proposed research is implemented as shown in the below algorithm.

Step 1: Create a database with primary key.
Step 2: Create an index database with two attributes key and reference.
Step 3: Dividing the database into *m* partitions by assigning the first key record value of each partition to the index database.
Step 4: The primary key of the reference table is stored in key attribute and the address of the corresponding record is stored in the reference attribute of the index table.
Step 5: once the index table is created it is updated internally whenever the any updations to the original database are updated (i.e., the index table grows along with the growing database).
Step 6 : Read the key value as the search key to search the database.
Step 7: First the index table searched for the key attribute for the corresponding value it falls in as shown in Equation .1
Step 8: The dataset to which the search key is loading into memory using reference value + m (interval value) - 1 i.e., the first record in the interval to key i.e., the first attribute value of index table.
Step 9: A linear search is performed to search the loaded database for the required record with the no. of comparisons for best case as shown in Equation 1 and in worst case as shown in Equation 2.

4. Experimental Results

In this research, there are 100 records for experimental purpose. The interval selected is one key for every 5 records. The first record of every five records is selected as the key index value in the index table. Thus the index table has 20 records with range of each equivalent to 5 records. The interval size can be changed as per user desires depending on the database size. The interval size range should not be too large or too small, as it might affect the time complexity for the data search method. Figure 2

shows the sample Dataset considered in the proposed approach.

```
Entered employee details:
-------------------------
¦ 101          Aruna     15000
¦ 102         Sanjay     13500
¦ 103         Sumaya     12580
¦ 104       Satheesh     10590
¦ 105         Anjali     13500
¦ 106          Ramya      9850
¦ 107         Lokesh     10000
¦ 108         Ramesh     12550
¦ 109       Venkatesh    15000
¦ 110   RaghupathiPFo    18000
¦ 111        Vanitha     13580
¦ 112       Sangeetha     7580
¦ 113         Mangala     13575
¦ 114        Shailaja     14080
¦ 115         Sunaina     9500
¦ 116          Dileep     18000
```

Fig. 2: Sample Dataset Used in the proposed algorithm

Index table contains two attributes, the first attributes corresponds to the Primary key of the linked database. The second attribute references to the memory location of the corresponding key attribute. The first record of the index table it refers to the first record of the database. It has an access range of about five records from the first record in this experiment. Followed by second record of the index table consists of key value equivalent to the next record followed by previous record access range i.e., sixth record and reference stores the memory location of the corresponding sixth record. Thus continues till the end of the database. In future as the database grows the index table grows respectively. Figure 3 gives the representation of Sample Index table for the database used in the proposed algorithm.

```
Index Table:
---------------
  Key     Reference
101       2038
106       2108
111       2178
116       2248
```

Fig. 3: Sample Index table for the database used in the proposed algorithm

The experimental results for searching the desired record as shown in Figure 4.

```
Enter the Record keyto be searched: 113

Record found:
-------------

The Record key 113 is in 3 Index
 Employee no : 113
 Employee name : Sangeetha
 Salary : 7580
```

Fig. 4: Experimental Results obtained for desired record.

However, the proposed method fails in terms of time complexity when the searching technique at both the levels is replaced by Binary search method. There by reducing the access time by $\frac{(n+m)}{2}$.

The proposed research is compared with normal database access procedure proposed by other researchers viz Anup K. Sen, G.Mohan, W..E Boebert in and as . Anup K. Sen et al. [3], 1989, mentioned a Fast recursive formulations for best-first search that allows controlled use of memory. Hermann Keindl et al.[5], 1994, described the Improvements on linear space search. Time taken to search each record is less than 1 second on a Quad core Processor with 2 GB RAM and found higher success rate of 92%. Figure 5 shows the success rate of the proposed approach.

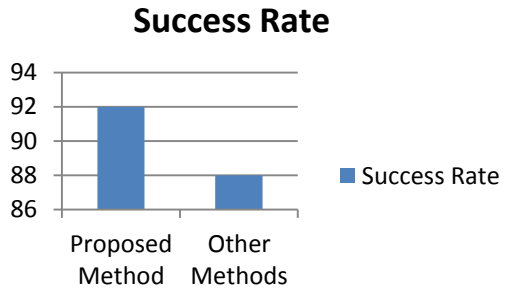

Fig. 5: Success rate of the proposed method with the other methods

5. Conclusions and Discussions

The large database is divided into equal intervals initially by creating an index table. Once the index table is created, from that point onwards only index table is used by the user access the database. Only the

administrator/programmer will know which database name is connected with the index table. Hence database hiding is implemented here. Also the entire database need not be loaded into the primary memory for processing, only the partition .i.e., and the intervals records will be loaded into memory through index table. Thus access time to the database is reduced to greater extent here.

Again it is based on the size of the partition made, if the size is too large, to process the interval partition may be time consuming. Hence it is preferred to the interval size to the considerable limit with reference to the size of the primary memory and processing power the system used.

In future studies, efficiency of the search algorithm is increased by using binary search method for index table and the database, for further improvement in the proposed system so that it becomes more pertinent to the design of real time system.

References

[1] A. Fikes. Storage architecture and challenges. Google Faculty Summit, July 2010.

[2] A. Labrinidis, Q. Luo, J. Xu, and W. Xue. Caching and Materialization for Web Databases. Foundations and Trends in Databases, 2010, 2(3):169–266.

[3] Anup K. Sen and A. Bagchi, 'Fast recursive formulations for best-first search that allows controlled use of memory', in Proceedings IJCAI-89, pp. 297–302.

[4] Hacigumus, H., Iyer, B. and Mehrotra, S. 2004. Ensuring the Integrity of Encrypted Databases in the Database-as-a-Service Model. Retrieved 7th March 2015 from http://link.springer.com/chapter/10.1007%2F1-4020-8070-0_5?LI=true.

[5] Hermann Keindl, Angelika Leeb, and Harald Smetana, 'Improvements on linearspace search algorithms', in Proceedings ECAI-94, pp. 155–159, (1994).

[6] J. Baker et al. Megastore: Providing scalable, highly available storage for interactive services. In CIDR, pages 223–234, 2011.

[7] J. Dean. Evolution and future directions of large-scale storage and computation systems at Google. In SOCC, 2010.

[8] G. Lohman, C. Mohan, L. Haas, D. Daniels, B. Lindsay, P. Selinger, P. Wilms. Query Processing in R*. In Query Processing in Database Systems. W. Kim, D. Reiner, D. Batory. (Eds.) NY: Springer-Verlag, 1985.

[9] Markl, G. Lohman, and V. Raman, "Leo: An autonomic query optimizer for db2," IBM Systems Journal, vol. 42, pp. 98–106, 2003.

[10] M. Stonebraker. SQL databases v. NoSQL databases. , 2010, CACM, 53(4).

[11] National Science Foundation. Core Techniques and Technologies for Advancing Big Data Science &Engineering (BIGDATA) 2012.

[12] Oracle. 2011. Retrieved 13th March 2015 from http://www.oracle.com/technetwork/topics/entarch/oes-refarch-dbaas-508111.pdf.

[13] Pizzete, L. and Cabot, T. (Authors). 2012. Database as a Service: A Marketplace Assessment.

[14] S. A. Wright, S. D. Hammond, S. J. Pennycook, and S. A. Jarvis, "Light-weight Parallel I/O Analysis at Scale," Lecture Notes in Computer Science (LNCS), October 2011 vol. 6977, pp. 235-249.

[15] T. Kraska et al. MDCC: Multi-data center consistency. In EuroSys, 2013.

[16] W. Hsu and A. J. Smith, "The Performance Impact of I/O Optimizations and Disk Improvements," IBM Journal of Research and Development, March 2004, vol. 48, no. 2, pp. 255-289.

Ms. Ashwini is currently working as Lecturer in the Dept. of Computer Science, Alva's College, an unit of Alva's Education Foundation, Moodbidri, Karnataka, India. Research interests include Image Processing, Data Mining and Database Applications. He has Completed Master of Computer Applications (MCA) from Indira Gandhi National Open University, New Delhi, India in the year 2010.
Email: aashindia023@gmail.com

Mr. Dileep M R is currently working as Lecturer in the Dept. of Computer Science, Alva's College, an unit of Alva's Education Foundation, Moodbidri, Karnataka, India. Research interest includes Image Processing and Database Applications. He has Completed Master of Computer Applications (MCA) from Visvesvaraya Technological University, Belgaum, Karnataka, in the year 2013.