Volume 3, Issue 11, November 2013 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Implementation of Fastest Searching & Retrieving Algorithm

Prof.Ajanthaa Lakkshmanan*, R.Krishna Priya, R.Gayathri, A.Lokesh Vishal, R.Rahul

SCSE & SITE / VIT University,

Vellore, India

Abstract*— Searching and indexing of complex or simple databases are in popular demand. Different approaches for organizing the data are proposed and implemented, such as full text searching, signature files, sequential signature files and its different variants. A signature tree is a data structure constructed over a signature file to speed up searching all those signatures, which match a given query signature. In this project, we propose the structure of signature trees to speed up the query processing. We take a string as an input and use a suitable hash function to convert into its byte (binary) equivalent value. This forms the basis of construction of signature trees. A new concept of signature identifier is introduced here which is the path from the root node to the leaf node which contains the index value. Searching in the tree is done by matching the bit pattern with the corresponding query given for search. The leaf nodes point to the database containing related details of a successful match. Both the signature tree and the query details are stored in separate files. It helps in faster retrieval of the details for the query given for the search and also there is a low overhead consumption.

***Keywords*— signature trees, query, hash function**

I. INTRODUCTION

Indexing has been very popular in complex database applications. Keeping in mind the response time and the space overhead many older technologies like signature files, bit-slice files have been proved inefficient. Since the modern database is very huge and complex, we need to care about the time and the cost too. So signature trees are good enough to retrieve details quickly and efficiently satisfying all the above constraints. The basic principle of the signature trees is that it uses bit values instead of strings and the leaf nodes directly point to the database.

II. LITERATURE SURVEY

A. Old Indexing Method

The signature file method has been widely advocated as an efficient index schema to handle large volumes of textual databases and recently extended to support a wide range of applications, such as multi-media hypertext systems, relational and object-oriented databases, as well as data mining. In comparison with the other index structures, it has mainly the following advantages:

- can be used to efficiently evaluate set-oriented queries;
- can handle insertion and update operations easily.

Intuitively, a signature file can be considered as a set of bit strings, called *signatures*. A typical query processing with the signature file is as follows: when a query is given, a query signature (a bit string) is formed from the query values. Then each signature in the signature file is examined over the query signature. If a signature in the file matches the query signature, the corresponding data object becomes a candidate that may satisfy the query. Such an object is called a drop.

The next step of the query processing is the false drop resolution. Each drop is accessed and examined whether it actually satisfies the query condition. Drops that fail the test are called false drops while the qualified data objects are called actual drops. In general, for each query processed, the entire signature file needs to be searched. Consequently, the signature file method involves high processing and I/O cost.

In a signature file, a set of signatures is sequentially stored, which is easy to implement and requires low storage space and low update cost. However, when a query is given, a full scan of the signature file is required. Therefore, it is generally slow in retrieval. If more than one objects share a same signature, that signature will be associated with the identifiers of all those objects.

Today, information retrieval implementations utilize one, or more, of the following techniques: full text scanning, inversion, and the signature file. Full text scanning introduces zero space overhead, but involves long response times. In the cases of inversion and the signature file, an intermediary representation structure (index) is utilized providing direct links to relevant data. Concentrating on the inverted index and the signature file, the former excels in query processing efficiency, whereas the latter involves a simpler structure and utilizes significantly less secondary storage. Inverted index structures are usually implemented as B+tree variants, whereby each real text block address is stored more than once. The scheme needs to frequently undergo re-organization under intensive information insertion/updating procedures. The method is also reported to perform poorly for multiple term user queries.

B. New Indexing Method

This problem is mitigated by partitioning a signature file, by introducing auxiliary data structure, as well as by exploiting parallel computer architectures. Recently, a new data structure, called a **signature tree**, is proposed to get rid of useless signature comparisons. A new method was proposed to organize signature files to speed up a signature file scanning. Using this method, a tree over a signature file *S*, called a signature tree, is constructed with the following properties.

(1) Each node is associated with a number to tell which bit in query to check when node is encountered during the tree searching.
(2) For each node, its left outgoing edge is labelled with 0 and its right outgoing edge is labelled with 1.
(3) Each path from the root to a leaf represents a signature identifier that uniquely identifies a signature in *S* just as a position identifier used to identify a substring.

III. PROPOSED SYSTEM

The signature tree is indeed a faster indexing method to retrieve details from the database. The system basically involves following steps to searching for a particular detail:-

- Firstly the tree is constructed using the index pointers of the database (basically a binary tree). The following rules are applied :-

  - Each node is associated with a number to tell which bit in query to check when node is encountered during the tree searching.
  - For each node, its left outgoing edge is labelled with 0 and its right outgoing edge is labelled with 1.
  - Each path from the root to a leaf represents a signature identifier that uniquely identifies a signature in *S* just as a position identifier used to identify a substring.
- Then the query string is taken as an input which is basically an index (user's choice).
- The corresponding bit pattern of the string is compared with the path from the root to the index node.
- When the path matches the pattern the index node which is a leaf node is used to retrieve data from the database.

IV. DESIGN AND IMPLEMENTATION

A. Module Description

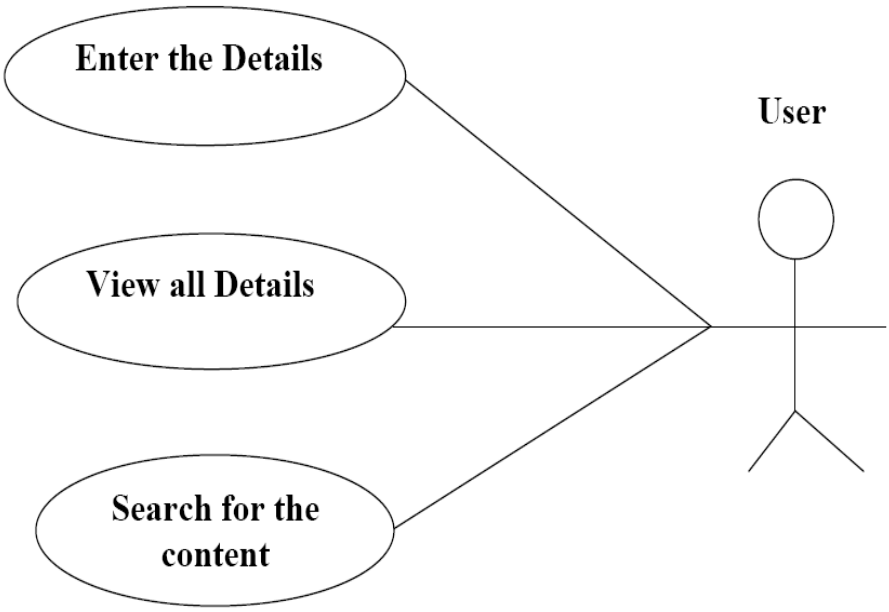

Fig. 1 Use case Model

The description of the different actors and their roles involved in the use case model are as follows:

User: The user of the system whose functions are:-

a. Enter the details: Responsible for entering the detailed information of the user in the database.

b. View User Details: Responsible for checking the details of different users stored in the database thus maintaining the integrity of the system.

c. Search using a query: The user searches for a particular string (index) by posting a query string as an input and gets the response based on the tree search.

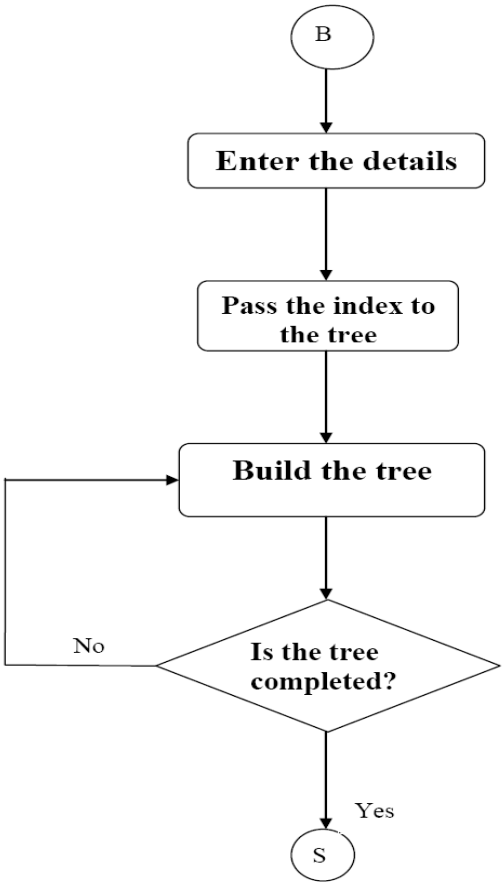

Fig. 2 Module 2

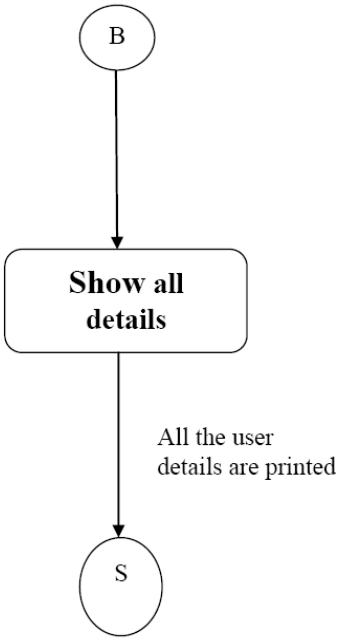

Fig.3 Module 2

1) Module1 – (Enter the details)

- Name of the use case: Enter the details
- Description: Responsible for entering the detailed information of the user in the database.
- Algorithm:
  - Enter the name and the details which are to be stored in the database.
  - The index is passed to the tree.
  - Build the tree.
  - Stop(S)

2) Module 2 – (Viewing of Details)

- Name of the use case: View all details.
- Description: Responsible for checking the details of different users stored in the database thus maintaining the integrity of the system.
- Algorithm:
  - Show all details.
  - Stop

3) Module 3 – (Searching using a query)

- Name of the use case: Search using a query
- Description: The user searches for a particular string (index) by posting a query string as an input and gets the response based on the tree search
- Algorithm:
  - Take a string as an input for search
  - Convert it into a binary value using hash function
  - Search the pattern in the tree which corresponds to the given string.
  - If the pattern is found mark the leaf node
  - Using the index in the leaf node locate the details in the database
  - Print the details

- Alternate flow of events:

i. Take a string as an input for search

Note: (Tree Search)

- ✓ While searching for the pattern each bit of the input string is checked the tree node
- ✓ The path is identified sequentially which represents a *signature identifier*
- ✓ The leaf node is identified

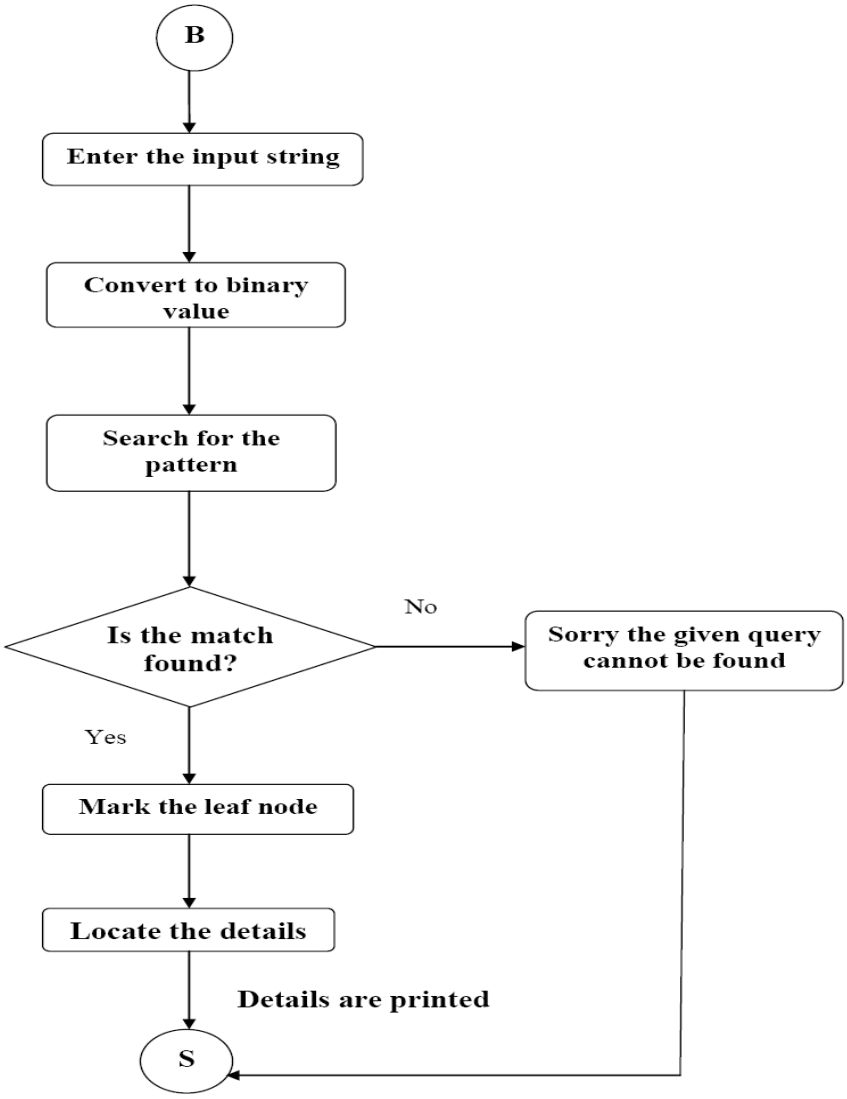

Fig. 4 Module 3

V. CONCLUSIONS AND FUTURE ENHANCEMENTS

In this project, a new concept of signature identifiers has been introduced, which can be used to identify the pattern in the signature tree. Based on this concept, a tree structure, called a signature tree, is proposed in which each path from the root to a leaf node corresponds to a signature identifier which in turn leads to the correct signature.

The traversal of a signature tree has improved:-

- ➢ the query processing time in complex databases.
- ➢ the response time

This code can be used as a package for any of the database applications provided the index value (here indexing is done by NAME). Today indexing a database has been very popular so this is one of the implementations which can serve as a faster and an efficient method.

Thus signature tree is a demanding data structure for retrieving details from the database. It has applications in various fields like:

- ➢ Hospital information system
- ➢ University management system

- Banking issues
- Sensex calculations
- Library system

ACKNOWLEDGMENT

The authors would like to thank the School of Computing Science and Engineering, VIT University, for giving them the opportunity to carry out this project and also for providing them with the requisite resources and infrastructure for carrying out the research.

REFERENCES

[1] Signature files and signature trees - YANGJUN CHEN ,Volume 82 , Issue 4 (May 2002) Pages: 213 - 221 , Year of Publication: 2002

[2] On the cost of searching signature trees - YANGJUN CHEN, Department of Applied Computer Science, University of Winnipeg, Available online 11 April 2006

[3] Comparison of signature file models with superimposed coding - D. Dervos , Y. Manolopoulos , P. Linardis , a. Department of Informatics, Aristotle University 540 06 Thessaloniki, Greece, b. Department of Informatics, Technology Educational Institute, 541 01 Thessaloniki, Greece, Received 26 January ; revised 16 June 1996

[4] W. Lee, D.L. Lee, Signature file methods for indexing object-oriented database systems, in: Proc. ICIC'92 — 2nd Internet. Conf. on Data and Knowledge Engineering: Theory and Application, Hongkong, December 1992, pp. 616–622.

[5] Key-based partitioned bit-sliced signature file - Volume 29, Issue 2 , Pages : 20 - 34 , Year of Publication: 1995

