Volume 2, No. 1, Jan-Feb 2011

ISSN No. 0976-5697

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Development of Performance MIS

Dhawal Seth*
B.Tech CSE
Vellore Institute of Technology
Vellore, Tamil Nadu, India
dhawal.seth@gmail.com

Mr. L. Ramanathan
Assistant Professor
School of Computing Sciences and Engineering
Vellore Institute of Technology
Vellore, Tamil Nadu, India.
lramanathan@vit.ac.in

Abhishek Pandey
B.Tech CSE
Vellore Institute of Technology
Vellore, Tamil Nadu, India.
apandey846@gmail.com

Abstract: **A management information system** (**MIS**) is a subset of the overall internal controls of a business covering the application of people, documents, technologies, and procedures by management accountants to solving business problems such as costing a product, service or a business-wide strategy. Academically, the term is commonly used to refer to the group of information management methods tied to the automation or support of human decision making, e.g. Decision Support Systems, Expert systems, and Executive information systems.

Keyword: MIS, Management Information System, Indexing, Development, C-ISAM.

I. INTRODUCTION

An 'MIS' is a planned system of the collecting, processing, storing and disseminating data in the form of information needed to carry out the functions of management. According to Philip Kotler "A marketing information system consists of people, equipment, and procedures to gather, sort, analyze, evaluate, and distribute needed, timely, and accurate information to marketing decision makers."

The terms *MIS* and *information system* are often confused. Information systems include systems that are not intended for decision making. The area of study called MIS is sometimes referred to, in a restrictive sense, as information technology management. That area of study should not be confused with computer science. IT service management is a practitioner-focused discipline. MIS has also some differences with Enterprise Resource Planning (ERP) as ERP incorporates elements that are not necessarily focused on decision support.

II. THE PROPOSED METHOD

The project implements the modified version of the b-tree for maintaining a database management system, which includes all the facilities of a basic database system. The concept used to decide the line of action was C-ISAM library. C-ISAM stands for Indexed sequential access method, it is a library used to maintain a database. However, the library was not available as it was a propriety library of IBM Informix. So, we worked out the solution, which used the concept of C-ISAM, using a modified Bayer's Tree. The project includes a backend implementation of the modified BTREE version, which handles the database storage strategies. The frontend of the projects handles the basic GUIs and the services which the application provides. The project focuses on development of MIS. This maintains the database of few tables regarding the Sales and Delivery in a Company. The fact is that a file is stored as a sequential data of records. For example, A table of Project has three fields for example, Project number, name, date, project engineer, would be like:

Serial No.	**Name**	**Date**	**Emp. Name**	**…**
302	Project 1	12/3/2009	Amit	
405	Project 2	31/01/2007	Kunal Singh	
…				

Figure.1 Showing a sample record

Storing the record in a form of struct data type, for each table, would be a solution for the problem of the mode of storage. But, the records stored in the file, would not return a reliable file pointer. Therefore, we had to store the database in the simple style of storing in a sequential format as given below:

DATABASE FILE,
storing the data in a

File pointer position

302 project1 12/3/2009 Amit
405 project2 31/01/2007 Kunal Singh

File pointer
position is 30

Figure .2 This depicting the way of storing the data in a file

The records can be identified uniquely by the primary key of the table. Hence, using the feature of the key, one can search any record in a database. Now, the problem was of storing the primary key with the record's position is the only chance of getting any desired record.

Hence, we stored the file position against the primary key of each record. This is also called indexing of each record.

As explained below:

A. Implementation Of Indexing

As the data is being stored in the file, the starting location of each record, in the sequential file, with the primary key is stored in a file named "master index".

Master index contents are:

For example,

The master index file

Each table has it's own master index file

Now, the indexes of the records are stored in a file. The question arises that in which way one should implement this search of primary key, to access the record's location in the database file.

Other than that, searching of an index in the Master Index file would result into a larger time complexity, if the records are larger in number.

[a] Proposed Solution Strategy

To solve the above stated problem, one can store the indexes in a form of data structure. This implementation is being done using the concept of BTREE (Bayer's Tree), which stores the primary keys in a form of tree, which have a root and leaves.

[b] BTREE Modification

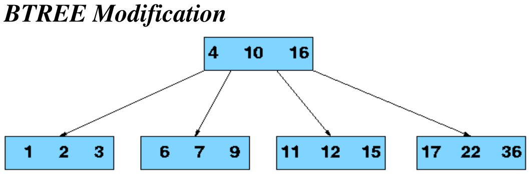

Figure. 4 Figure showing a B-TREE

The original BTREE is being modified in the project, so as to make it customized to our requirement. The concept of BTREE is to store the data values in a sorted order and search the key as per the value of the current node with the search key.

For example,

The regular BTREE has the factors of minimizing the search for a leaf node by a factor know as t, which states that the maximum nodes a leaf can have is 2t-1.

In the modified BTREE, implemented, the maximum keys stored in a leaf is set constant as 3, as the number of records being stored is assumed to be less.

Further, the searching for the required key in the BTREE is done using simple binary search, as the keys are stored in ascending (or already sorted) order.

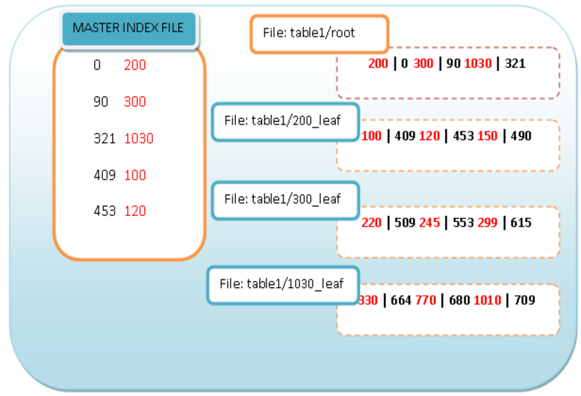

Figure.5 Implementation of the B-TREE

This method is adopted because when the amount of records exceeds a certain level, say 10MB, then the usage of RAM would increase and the system might consume more and more memory on increase of records. Instead the storage of each leaf is been on the disk and whenever the leaf is required the corresponding leaf file is accessed for that purpose.

This way the usage of RAM decreases, though it decreases a factor for I/O performance. This method is devised keeping in mind the number of records in the database to be above a million.

B. Implementation Details

The implementation of the problem was done in a simple fashion. The steps involved are:

[a] For storing the nodes, in a format of BTREE

[i] START

[ii] The user enters the record, the primary key is stored in the MASTER INDEX FILE, along with the file location.

[iii] As the record is saved, the MASTER INDEX FILE is read and the modified BTREE is formed as:

[iv] The first record is stored in the root file, as seen in the figure below.

[v] As the records are saved, the root file is filled, along with their file pointer positions, in an ascending order.

[vi] After the root is completely filled, the records are filled following the order of leaves i.e in ascending order.

[vii] The naming of the leaves are named as table/first node_leaf, which states that it contains the nodes having value less than primary key of the first node of the root.

[viii]Similar is with every leaf.
[ix] STOP

[b] For searching of the nodes, in the BTREE:

[i] START
[ii] The BTREE is stored in an ascending order, the value to be searched is first picked, for matching, in the root node.
[iii] The root leaf is searched for the match, in a binary search mode.
[iv] If the value is matched, by a node in the leaf, the result is returned with the corresponding file pointer.
[v] If the value is not found in the current leaf, the related position of the next leaf to be searched is returned to the recursive function.
[vi] The function returns if the value is not found, when the function reaches the last leaf without a match.
[vii]STOP

The root of the BTREE is been stored as a file namely “table1/root”, the keys having lesser value than the key 200 in the root is been stored as “table1/200_leaf”, and so on.

SEARCHING IN A BTREE (example)
Command: Search 250, in the BTREE.

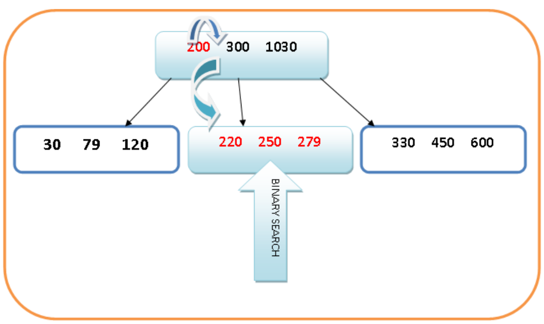

Figure .6 Searching Mechanism

The above mechanism will render the complexity of searching a key from a large database system to mere O(log n).

[c] The deletion process:

[i] START
[ii] The user gives the primary key of the record to be deleted.
[iii] The root leaf is being searched for the match.
[iv] If the value is not matched, then user is given the notification.
[v] If the value is found in the BTREE, the value of the file position and the primary key is set to -1 in the MASTER INDEX FILE.
[vi] After that, the current BTREE of the record is deleted. A new BTREE is formed with the modified MASTER INDEX FILE.
[vii]Also, the database file is modified with an invalid value for the set of record, to be deleted.
[viii]STOP

[d] The modification process:

[i] START
[ii] The user gives the primary key of the record to be modified.
[iii] The primary key value is matched in the current BTREE.
[iv] If the value is not found, the user is given notification of the result.
[v] If the value is found, the user is given a dialog box, which contains the current values of the record. Then the user, can alter or change the record details and save the new record.
[vi] As the primary key has a default file pointer stored against it in the MASTER INDEX FILE. The value is used to alter the record in the database file.
[vii]STOP

III. CONCLUSION

This method is adopted because when the amount of records exceeds a certain level, say 10MB, then the usage of RAM would increase and the system might consume more and more memory on increase of records. Instead the storage of each leaf is been on the disk and whenever the leaf is required the corresponding leaf file is accessed for that purpose. This way the usage of RAM decreases, though it decreases a factor for I/O performance. This method is devised keeping in mind the number of records in the database to be above a million.

The above problem has the time complexity of log(n). As the BTREE has an implementation in the hard disk rather than RAM. This states that, if the numbers of records are larger the size of memory used by the application will be less. But the I/O operations will render the system to slow, this problem can be dealt with the help of the BTREE itself, by increasing the number of nodes to be stored in each leaf, which is set by DEFAULT as 3 per leaf. This gives flexibility to the problem according to the number of records stored in a table.

IV. REFERENCES

[1] Beginning Linux Programming by Neil Mathew and Richard Stones. Publisher Wrox Publishing Inc. 4th Edition.
[2] “Development strategy of MIS for small and medium-sized enterprise”- Guo-Shun Lin Dong Xiang Qing Chang Shipping Manage. Coll., Dalian Maritime Univ.
[3] “Development of a comprehensive management information system for a large R&D laboratory”- Panella, R.F.; Jendrian, F.H.; ASTARS Program Branch, Wright Lab., Wright-Patterson AFB, OH.
[4] “Evolutionary database design and development in very large scale MIS”- M. C. Filteau, S. K. Kassicieh and R. S. Tripp
[5] IBM Informix C-ISAM Version 7.x for Unix and Linux, official e-book by IBM Informix. (ISBN - GC27-1491-00)
[6] “Systems Development in Information Systems Research”- Jay F. Nunamaker, JR., Minder Chen, and Titus D.M. Purdin.

