Performance Analysis of Data Organization of the Real-time Memory Database Based on Red-Black Tree

Liang Qiaoyu,Li Jianwei,Xu Yubin
Institute of System Emulation and Computer Application
Taiyuan Science and Technology University 030024
Taiyuan, China
Qiaoyu831230@163.com
Ghhong2004@163.com

Abstract: A scheme of the real-time memory database data organization based on Red-Black tree is presented and the performance is compared with other structure used in the real-time database referring to the practical application in the paper. According to the theory analysis and experiment, it is shown that the method has great superiority on inserting and deleting the magnanimous data.
Key words: real-time memory database; Red-Black tree; querying; accessing

I. INTRUDUCTION

Along with the modern database developing, the real-time memory database becomes the researching hot spot in the database and the industrial control domain day by day. Since 1980s, the real-time database has been application more and more widespread, the domestic and foreign companies already has carried on the real-time database in view of B tree, AVL tree. In real-time memory database system, the work-edition of the database is always resided in memory, so the B tree data organization which filling data in its each node is about 60% has not already suited any more. It may not be accepted by the memory database which is strict to the space. It is discovered that regarding the memory database the AVL tree is better than B tree[1]. Querying data quickly is the greatest character of AVL tree, but to maintain balance while inserting and deleting, the AVL tree need to rotate itself. And the rotation is possible frequent sometimes, which is too low efficient to renew data in the real-time memory database which emphasizes on the performance of the real-time.

A real-time memory database model is designed and realized on the Red-Black tree. It is indicated by the experiment that the Red-Black tree is not only one kind of highly effective querying tree but more importantly it provides one kind of real-time's manipulation of inserting and deleting, which provides the support to the real-time memory database's timeliness.

II. THE RED-BALCK TREE

The Red-Black tree proposed by Rudolf Bayer is one kind of self regulation and improved binary tree. An effective Red-Black tree has all properties of binary tree and also has the requirements as following:

[1] *1)* Each code is colored either red or black;
2) Root is black;
3) If a node is red, both of its children are black;
4) Leaf code is black;
5) All offspring nodes of each code include the same number of black nodes.

The key property of Red-Black tree is decided by these requirements: the longest path length from the root node to the leaf node is less than twice the shortest path. This limitation to the height guarantees that the Red-Black tree is all highly effective in inserting, deleting and querying some value at the worst situation, which surpasses binary tree by far. Moreover these requirements also suggested that if a red node has two sub-children, these two sub-nodes must be the black ones, and if a black node has a sub-child, then this sub-child must be the red one. Figure 1 is a typical Red-Black tree.

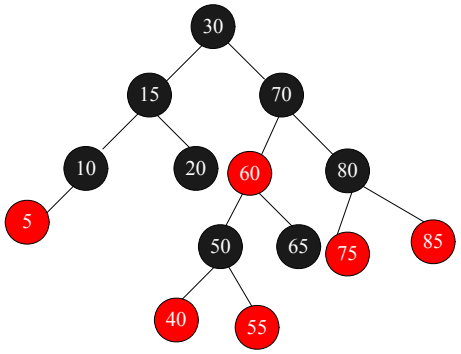

Figure 1 The Red-Black tree

III. RELIZING THE REAL-TIME MEMORY DATABASE MODEL

A. *Designing the model*

In real-time memory database, the efficiency of updating data will be very low with accessing vast amounts of data so it is necessary to optimize the database. In order

Fund project:The colleges and universities science and technology research and development project of the Shanxi Province education's department (200758),Shanxi Province youth fund(20081020)

978-0-7695-4026-9/10 $26.00 © 2010 IEEE
DOI 10.1109/CCIE.2010.113

to enhance the accessing performance of the database, a real-time database based on the Red-Black tree is adopted.

This model is mainly used for testing the performances of the real-time database based on the Red-Black tree therefore some basic functions of the database such as adding, querying, updating, deleting and so on are realized. The system is designed with OOA and programmed with C++, where the Red-Black tree and the nodes are defined class. The relationship between the tree and the nodes are shown in figure 2.

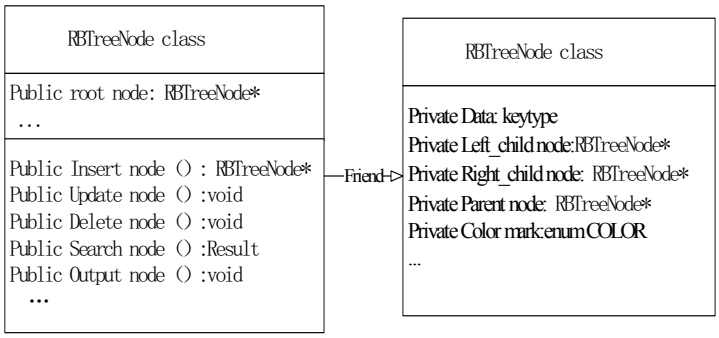

Figure 2 The Red_black tree class and the nodes class

B. *Realizing the programme*

Take inserting data as example to describe the realization of the program:

1) According to the general binary tree, inserting a node.

```
RBTreeNode* RBTree::RB_tree_insert(KeyType key);
//inserting node
        if(key<down->key)
                down=down->lchild;
        else
                down=down->rchild;
```

2) Test the tree which added a new node whether it conforms to the Red-Black tree rule and then update some nodes and rebalance this tree according to the rule.

```
void RBTree::RB_insert_fixup(RBTreeNode* &change);
    /*the grandparent node has two children, if uncle
    node and parent node are //red, just swap the color.
    */
                        father->color=BLACK;
                        uncle->color=BLACK;
                        father->parent-
>color=RED;
                        change=father->parent;

     /*the grandparent node has one child and the uncle
    code is black, this case can be fixed with rotations.*/

                        Left_rotate(change);
                        father->color=BLACK;
                        father->parent-
>color=RED;
                        Right_rotate(father-
>parent);
```

3) Traversal the Red-Black tree in order

```
void PrintTree(RBTreeNode* &pRoot);
```

IV. ANALYZING AND CONFIRMING THE PERFORMANCE

A. *Analyzing the performance*

A little manipulation shows that the worst-case height simplifies to hworst = 2 lg (n + 2) – 3, the certificate is as follows:

1) Let k represent the number of black nodes per null path and $h = 2k - 1$ represent the height of the worst-case tree.

2) To determine F (k), the number of nodes in the worst-case tree, we note that

F (1) = 2 and F (k) = F (k – 1) + 2 + 2(2k – 1 – 1).

3) Use maple:

```
Rsolve ({F (1) = 2,   F (k) = 2 + F (k - 1) + 2*(2^ (k
– 1) – 1)}, F (k));
                    2 *2^k – 2
Solve ({h = 2*k - 1}, k); //solve equation for k
                     k=h/2+1/2
Eval (2*2^k - 2, %); //evaluate at k = h/2+1/2
                2 *2^k – 2=2*2^(h/2+1/2)-2
Solve ({n = %}, h); //solve for h
          h=(2ln(n/2+1)-ln(2))/ln(2)                    (1)
```

The height (1) above on implies that the tree is effective to query data. Although the time complexity of operation algorithm of both the Red-Black tree and other binary trees are all O(logn)[2], the operation efficiency of inserting and deleting of the Red-Black tree are higher than the common used tree structure. So it is assumed that the data access efficiency is enhanced in the real-time memory database based on the Red-Black tree.

B. *Testing and confirming*

The testing platform is made up of Pentium 4 3.00GHz processors, 512MB memory, WindowsXP operation system, and VC++ 6.0. While searching and inserting and the number of records from 500,000 to 3,000,000 in the AVL tree and the Red-Black tree separately, write down the time. Regarding each kind of operation, derive the average from 100 times executions. The average is shown in Figure 3 and Figure 4.

Figure 3 shows while searching n random data, the average querying time of AVL tree is shorter than Red-Black one, but their disparity is very small. Along with the querying data increasing, the growth rates of querying time of both trees are almost same. So it is thought that Red-Black tree is another fast querying data structure besides AVL tree. Figure 4 indicates that the average inserting time of AVL tree is more than the Red-Black one while inserting n random. Along with the inserting data increasing, the growth rate of inserting time of AVL tree is much more than of Red-Black tree. It is indicated that the real-time memory database based on Red-Black tree sacrifices the less

querying time, but saves much more time to access data, which is a quite good optimized scheme.

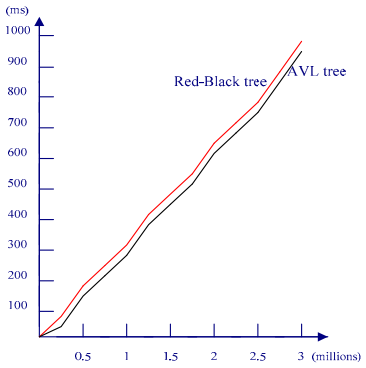

Figure 3 Searching time

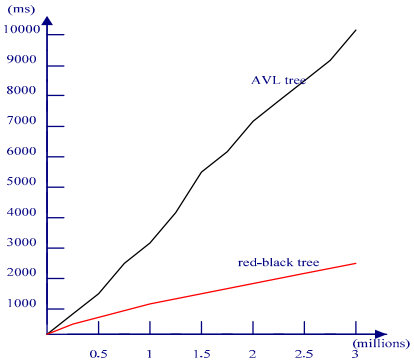

Figure 4 Inserting time

V. CONCLUSIONS

The Red-Black tree does not pursue perfect balance; therefore its querying efficiency is lower than the AVL tree. But because of partial balance the Red-Black tree decreases requirements for rotation, thereby the performance of accessing data are enhanced. Testing through the experiment, it is concluded that the performance of accessing data is enhanced with Red-Black tree in the real-time database. Due to higher accessing efficient of Red-Black than AVL tree, which satisfies the real-time performance of the real-time database.

REFERENCES

[1] Zhao-Yuan Li,Xiao-Pei Luo. The database technology progresses newly[M]. Qinghua University publishing house, 2007.

[2] Rui-Ying Cai. Modern computer commonly used construction of data and algorithm[M]. Nanjing: Nanjing University publishing house, 1994.

[3] COLLINSW J . Data Structures and the Standard Template Library[M]. Boston：McGraw-Hill，2003.

[4] H.Garcia-Mollina, K.Salem.Main memory database systems: An overview.IEEE rans.on Knowledge and Data Engineering,1992,4(6):509～516.

[5] QingGao,FanJiang.ApplicationofRBtreealgorithm[J].Software Guid,2008(9).

[6] Dong-Jun Luo.A List Structure Capable of Fast Searching Based on the Red-Black tree.Journal of Yunnan Nationlities University.2008(3).

[7] Qiang-Zhang Chen.An efficient Binary Search Tree Read-Black tree.Journal of East China Normal University,2000(3).

[8] Azer Bestavros and Sue Nagy,"Value-cognizant Admission Control for RTDB Systems,"IEEE 16th Real-Time Systems Symposium,December 1996.