CSCI-B 561 ADVANCED DATABASE CONCEPTS
Assignment 4(a)
Fall 2015
Indiana University,
Bloomington, IN

Ramprasad Bommaganty
rbommaga@indiana.edu

11th October 2015

Solutions

1. The relational model is better than CODASYL due to the following reasons:

   (a) It offers physical data independence. For example, data can be stored by the administrator in such a way that a user can perform efficient queries without worrying about the physical representation of the data.

   (b) It also offers logical data independence. For example, a relational schema can be modified with much less complexity and effort when compared to the limited logical data independence in CODASYL
   which was actually poorer compared to IMS.

   (c) It does not implement the record-at-a-time data manipulation language. Set oriented data manipulation is carried out using operations such as Selection, Projection, Cross products and Joins.

   (d) A programmer does not have to navigate in a multidimensional hyperspace. It eliminates the need
   to remember the pointers from and to the record types in Owner and Member record types. Navigational
   programming is eliminated.

   (e) The data does not have to be bulk loaded all at once and load time is not as significant as CODASYL.

   (f) Data can be accessed using a high level interface with set-at-a-time DML.

   (g) The network model is too complex to implement certain simple models like marriage and other event occurences where multiple sets and record types have to be created. On the contrary, the relational model takes only a single table to implement it.

   (h) Adding new record types is very complex in CODASYL and if an instance of a record type is changed the older program version will not work until it is modified. This prompts a manual record by
   record maintenance of data in CODASYL which is time consuming. The relational model simplifies the
   procedure of adding new data and it is much more efficient.

2. In order to incur maximum I/O overhead while inserting just a single element, it is necessary to chose a leaf which is filled and whose parent nodes are also filled up. Hence, we can chose the node containing the elements : 34, 37,38, 39 and insert the values 35 or 36 to incur the maximum I/O overhead. Let us insert 35 into the B+ tree. Also, the number of disk I/O to retrieve a data entry is equal to $\log_{(fan-out)}$(no. of data entries)

   .

   The following are the steps during the insertion:

   (a) Insert an element 35 in the leaf node. The leaf node now contains 34, 35, 37, 38, 39. It has to be split into two.

   (b) The new split nodes are: 34, 35, X, X, and 37, 38, 39, X. Since 37 is the middle value, it has to be copied into the parent node.

   (c) After copying 37 into the parent node, it leads to five data entries of : 30, 34, 37, 40 , 50.Now, we have to split the internal node into two parts and move the middle value 37 into the root node.

   (d) The key 37 is moved onto the root and now the split nodes are 30, 34, X, X(three pointers to leaf nodes) and 40, 50, X, X,(three pointers to leaf nodes) and the root node is 24, 37, X, X with three pointers to the internal nodes compared to the two pointers of the initial configuration. Thus, the value 35 has been inserted with maximum overhead into the B+ tree.

   The data entries in the tree are: 24
   The number of blocks are: 36
   The fan out : 5
   Therefore, number of disk I/O = $\log_5(2/3)$
   NOTE: A JPEG image of the final B+ tree has been attached to the assignment.

3. Yes, the algorithm will still work and it will find the heavy hitters(ϵ, ϕ) if the tuple with the minumum frequency is deleted.

   Let us assume a sequence of numbers to perform the correctness analysis:
   $2\ \widehat{3}\ \widehat{4}\ \widehat{2}\ \widehat{5}\ \widehat{6}\ \widehat{2}\ \widehat{3}\ \widehat{2}$

   Let $\epsilon = 3$, then $1/\ \epsilon$ of the total count = 3.

   1. The e = 2 and f(e)= 1
   2. e = 3 and f(e) = 1
   3. e = 4 and f(e) = 1

   Here the array is full so the values are decremented.
   4. (e,f(e)) = (2,0),(3,0), (4,0)

   We delete the element e =4 from the array since its count is 0.
   5. e = 2, f(e) = 1
   6. e = 5, f(e) = 1
   7. e = 6, f(e) = 1

   Here the array is full so the values are decremented.

   8. (e,f(e)) = (2,0),(5,0), (6,0)

   We delete the element e =4 from the array since its count is 0.
   9. e = 2, f(e) = 1
   10. e = 3, f(e) = 1
   11. e =2, f(e) = 2

   Here, 2 is the most repeated item in the array. Since it has been observed that the items with frequency of 0 have been eliminated, it follows that during every iteration the items with the lowest frequency will naturally be eliminated when the array count exceeds $1/\ \epsilon$.

   Hence, even if the tuple with the lowest frequency is eliminated the Misra-Gries algoritm is successful in finding out the heavy hitters(ϕ, ϵ).

References

1. Database systems - the complete book.
2. Database Management Systems, Ramakrishnan.
3. http://www.mif.vu.lt/ algis/dsax/Index-Btree.pdf
4. http://zgking.com:8080/home/donghui/publications/books/dshandbook$_B Tree.pdf http://www.cs.umb.edu/cs634/cl$
5. http://www.cs.bu.edu/fac/gkollios/ada05/LectNotes/lect2.pdf