AN ANALYTICAL MODEL OF A DEFERRED AND INCREMENTAL UPDATE STRATEGY FOR SECONDARY INDEXES

Edward Omiecinski, Wei Liu and Ian Akyildiz
School of Information and Computer Science
Georgia Institute of Technology
Atlanta, GA 30332 U. S. A.

1. Introduction

The advantages of using an index for the retrieval of tuples from a relational database is well known. The benefit of having indexes is offset by the cost of maintaining the indexes in the face of updates. For some updates, the cost of maintaining the index may be greater than the cost of updating the desired set of tuples in the relation. In this paper, we describe a strategy for updating secondary indexes which can be classified as deferred and incremental. By deferred, we mean that the index is not updated when the user's update statement is executed. Rather, the appropriate changes are simply recorded for later update of the index. By incremental, we mean that only certain recorded changes will be applied to the index at a given time. We stress that only the index updates are deferred and that the tuple updates are performed by the user's update statement.

2. Previous Research

In [5], the use of a differential file is proposed. The differential file stores all updates, leaving the main file unchanged. Eventually, the differential file and main file are merged. In [5], they were interested in designing a Bloom filter which would indicate whether a given record could be found in the differential file.

In [3,6], the use of a differential file for updating a main file which is organized as a B+ tree is presented. In both [3] and [6], the approach is to perform an efficient batch update of the tree structured main file. In [3], they are concerned with determining the optimal time at which the batch update should take place. In [6], they use a tree structure for the differential file and assume that it can reside in main memory.

Instead of performing a batch update, others [1] have proposed to do incremental updating of the main file. Usually an update will be triggered by a query. In [1], the differential file does not contain the updated records but the update procedure.

In [2], an incremental and deferred update strategy is presented for the maintenance of text indexes. Their emphasis is on concurrency control methods and efficient data structures for the deferred update information.

3. Deferred and Incremental Index Update

Our deferred and incremental index update approach uses a differential file which grows as updates are executed and shrinks as queries are executed. A record in the differential file consists of the following attributes: a *TID*, the *old-key value* and the *new-key value*. For our purpose, we assume that the differential file is a simple sequential file with an update inplace capability. In addition, a record with a particular *TID* value occurs only once in the differential file.

The update procedure is simple and is shown below.

```
1. Access tuples to be modified (via a single index or file scan). If access is through
   the secondary index, then the query procedure (which follows) must be invoked.
2. Modify tuples.
3. For each indexed attribute, of the modified tuples, that has been modified:
   if the TID for this tuple does not appear in the differential file
       then write a new record to the differential file
   else update the new-key value for an existing record in the differential file
```

A naive query procedure, which uses the secondary index as the access method, is shown below.

```
1. Follow the appropriate path from the root to a leaf page of the index, (the leaf
   page contains key values and their associated TID lists).
2. Search the differential file for key values which match the requested key values
   contained on the leaf page.
   2.1 If the old-key value, from a record in the differential file, matches then delete
       the corresponding TID value from the TID list for the key and set the old-key
       value to null.
   2.2 If the new-key value, from a record in the differential file, matches then
           if the old-key value is null
             then
                 insert the corresponding TID value in the TID list for the new
                 key and set the new-key value to null
             else
                 save the TID value in a temporary TID list.
   2.3 If both the new-key value and the old-key value, for a record in the differential
       file are null, then delete that record from the differential file.
3. If necessary, access the next leaf page and go to step 2.
4. Retrieve the tuples for the TID lists which have been found (including the tem-
   porary TID list). The TID lists contain any modifications as performed in step 2.
```

In [4], we prove that only one record in the differential file is needed per tuple, regardless of the number of key value changes that have taken place on any given tuple. We also prove in [4], that any *TID* will occur in the index at most one time and possibly be absent from the index during the deferred update.

4. Analytical Model of Differential File Size

Let us denote $P[size=s]$ as the probability that the differential file size is s, for

$s = 0,1,\ldots,S_{max}$ and $P[i \rightarrow s]$ as the probability that the size of the differential file before the transaction is *i*, and after the transaction it is *s*. For a single update, a new record will be added to the differential file if and only if the *TID* of the record being updated is not contained in any of the existing differential file records. For a search transaction (single key or range of keys), if the *old-key value* for a differential file record is matched, then this value is set to $\varnothing$ and the indexed file is modified. A match on the *new-key value* will result in the record being deleted if and only if the *old-key value* in that record is $\varnothing$. Therefore, $P[i \rightarrow s]$ is determined by the number of records with the *old-key value* of $\varnothing$.

To characterize the above phenomenon, we use $P[empty \mid s]$ to denote the proportion of records with an empty *old-key value* when the differential file contains *s* records. It is obvious that for each record in the differential file of size *s*, the probability that its *old-key value* is empty is also $P[empty \mid s]$. The search transaction is treated as if it consisted of two steps. First, when the *old-key value* is matched, it is set to $\varnothing$. This results in $P[empty \mid s]$ being changed to a new value, $N(s,j)$, where $j = 1$ denotes a single key search and $j = 2,\ldots,R_{max}$ denotes a range search consisting of 2 keys through R_{max} keys. The variable R_{max} denotes the maximum number of keys in a range query. The calculation of $N(s,j)$ will be given later. The second step is to match the *new-key value*. It should be noted that $P[empty \mid s]$ changes from transaction to transaction. Another consideration is, that if $P[size = s] = 0$ then $P[empty \mid s]$ is undefined. The above two considerations direct us in the calculation of $P^{(t)}[empty \mid s]$ where *t* denotes that *t* transactions have been processed. We use $P^{(t-1)}[empty \mid s']$ for all states s' that can reach *s*. That is, the differential file can change from a state having s' records to a state having *s* records. Also, $P^{(t-1)}[empty \mid s']$ is undefined if $P^{(t-1)}[size = s'] = 0$.

Next, we develop an iteration model to calculate $P[size = s]$ based on $P[empty \mid s]$. Also, the maximum size of the differential file after *t* transactions have been processed can be easily predicted from the size distribution. We use the following notation in our model:

$Prob[up]$ =	probability of update	K_{max} =	number of unique key values
$Prob[ss]$ =	probability of single key search	S_{max} =	maximum size (in records)
$Prob[rs]$ =	probability of range search	R_{max} =	maximum number of keys in range search

Initially, we set $P^{(0)}[size = 0] = 1$ and $P^{(0)}[size = i] = 0$, for $i = 1,\ldots,S_{max}$. All $P^{(0)}[empty \mid s]$ are undefined for $s = 1,\ldots,S_{max}$. Iterate over equations (1 - 5), i.e., $t = 1,\ldots,ceil$, until the difference between the average size of the differential file in two successive iterations is less than an epsilon ($\varepsilon = 10^{-4}$) value. The probability that after *t* transactions, the differential file size is *s*, is shown in equation 1.

$$P^{(t)}[size = s] = \sum_{i=0}^{S_{max}} P^{(t-1)}[size = i] \cdot P^{(t-1)}[i \rightarrow s] \qquad (1)$$

where $P^{(t-1)}[i \to s]$ *is computed by equation 4*

$P^{(t)}[empty\,|\,s]$, in equation 2, is defined in a somewhat similar manner, except now, we have to take into account $Q^{(t\text{-}1)}[i \to s]$ for every i value that can lead us to a state with s records. $Q^{t}[i \to s]$, is defined in equation 5.

$$P^{(t)}[empty\,|\,s] = undefined, \quad \text{if } P^{(t)}[size=s]=0 \tag{2a}$$

$$P^{(t)}[empty\,|\,s] = \frac{\sum_{i=0}^{S_{\max}} P^{(t-1)}[size=i]\cdot P^{(t-1)}[i \to s]\cdot Q^{(t-1)}[i \to s]}{\sum_{j=0}^{S_{\max}} P^{(t-1)}[size=j]\cdot P^{(t-1)}[j \to s]} \tag{2b}$$

where $Q^{(t-1)}[i \to s]$ *is defined in equation 5*

The following equation yields the new proportion of records with an *old-key value* of Ø. This is relative to the differential file size and the number of keys which appear in a search request. This formula will be used in defining $Q^{t}[i \to s]$ as shown shortly.

$$N^{(t-1)}(s,j) = P^{(t-1)}[empty\,|\,s] + (1 - P^{(t-1)}[empty\,|\,s])\cdot\frac{j}{K_{\max}} \tag{3}$$

for $s=1,..,S_{\max}$ *such that* $P^{(t-1)}[size=s] > 0 \;\&\; j=1,..,R_{\max}$

Equations 4(a - d) depict the 4 possibilities for a transition in the differential file size.

$$P^{(t-1)}[s \to s+1] = Prob[up]\cdot(1 - \frac{s}{S_{\max}}) \qquad (if\ P^{(t-1)}[size=s] > 0) \tag{4a}$$

$$P^{(t-1)}[s \to s] = Prob[up]\frac{s}{S_{\max}} + Prob[ss]\left[1 - N^{(t-1)}(s,1)\cdot\frac{1}{K_{\max}}\right]^{s} \tag{4b}$$

$$+ \; Prob[rs]\cdot\sum_{j=2}^{R_{\max}}\frac{1}{R_{\max}-1}\left[1 - N^{(t-1)}(s,j)\cdot\frac{j}{K_{\max}}\right]^{s} \qquad (if\ P^{(t-1)}[size=s] > 0)$$

$$P^{(t-1)}[s \to s-\delta] = Prob[ss]\cdot\left[\binom{s}{\delta}\right]\left[N^{(t-1)}(s,1)\cdot\frac{1}{K_{\max}}\right]^{\delta}\cdot\left[1 - N^{(t-1)}(s,1)\cdot\frac{1}{K_{\max}}\right]^{s-\delta} \tag{4c}$$

$$+ \; Prob[rs]\cdot\sum_{j=2}^{R_{\max}}\frac{1}{R_{\max}-1}\left[\binom{s}{\delta}\right]\left[N^{(t-1)}(s,j)\cdot\frac{j}{K_{\max}}\right]^{\delta}\cdot\left[1 - N^{(t-1)}(s,j)\cdot\frac{j}{K_{\max}}\right]^{s-\delta}$$

$$(if\ P^{(t-1)}[size=s] > 0 \;\&\; \text{for } \delta=1,..s)$$

$$P^{(t-1)}[i \to j] = 0 \quad \text{for } all\ other\ cases \tag{4d}$$

Again, we consider the possible state transitions for the differential file in equations 5(a - e). Except, that here we calculate the proportion of records in the differential file whose *old-key value* is ∅.

$$Q^{(t-1)}[0 \to 0] \textit{ is not defined and is never used} \tag{5a}$$

$$Q^{(t-1)}[0 \to 1] = 0 \tag{5b}$$

$$Q^{(t-1)}[s \to s] = Prob[up] \cdot P^{(t-1)}[empty \mid s] + Prob[ss] \cdot N^{(t-1)}(s,1) \tag{5c}$$

$$+ \; Prob[rs] \cdot \sum_{j=2}^{R_{\max}} \frac{1}{R_{\max}-1} N^{(t-1)}(s,j) \qquad (if \;\; P^{(t-1)}[size=s] > 0)$$

$$Q^{(t-1)}[s \to s-\delta] = \frac{Prob[ss]}{Prob[ss]+Prob[rs]} \frac{s \cdot N^{(t-1)}(s,1)-\delta}{s-\delta} \tag{5d}$$

$$+ \; \frac{Prob[rs]}{Prob[ss]+Prob[rs]} \cdot \sum_{j=2}^{R_{\max}} \frac{1}{R_{\max}-1} \frac{s \cdot N^{(t-1)}(s,j)-\delta}{s-\delta} \qquad (if \;\; P^{(t-1)}[size=s] > 0)$$

$$Q^{(t-1)}[s \to s'] = 0 \qquad \text{for } \textit{all other cases.} \tag{5e}$$

In [4], we compare our analytical model with a simulation model. We show that the expected maximum differential file size (in number of records) is only 16% of the number of records in the relation. In addition, we show in [4] that the model differs from the simulation by approximately 15% for the maximum differential file size.

References

1. Cammarata, S., "Deferring Updates in a Relational Database System," 1981 VLDB Conference Proceedings, 1981, 286-292.

2. Dadam, P., Lum, V., Praedel, U. and Schlageter, G., "Selective Deferred Index Maintenance & Concurrency Control in Integrated Information Systems," 1985 VLDB Conference Proceedings, 1985, 142-149.

3. Lang, S., Driscoll, J. and Jou, J., "Improving the Differential File Technique via Batch Operations for Tree Structured File Organizations," 1986 Data Engineering Conference Proceedings, IEEE, 1986, 524-532.

4. Omiecinski, E., Liu, W. and Akyildiz, I., "Analysis of a Deferred and Incremental Update Strategy for Secondary Indexes," Georgia Institute of Technology, Technical Report GIT-ICS-88-41, November 1988.

5. Severance, D. and Lohman, G., "Differential Files: Their Application to the Maintenance of Large Databases," ACM TODS, 1, 3, Sept. 1976, 256-267.

6. Srivastava, J. and Ramamoorthy, C., "Efficient Algorithms for Maintenance of Large Database Indexes," 1988 Data Engineering Conference Proceedings, IEEE, 1988, 402-408.