'97 International Conference Multimedia Databases on Internet
October 10, 1997, Seoul Korea

Performance Analysis for Maintaining Distributed Views

Wookey Lee
Department of Computer Engineering,
Sungkyul University, Anyang, Korea., 430-742
wook@hana.sungkyul.ac.kr

abstract

Maintaining materialized views and/or replica can basically be considered as a client-server architecture that extracts the changes of the distributed source data and transfers them to the relevant target sites. View maintenance and materialized views are considered to be important for suggesting solutions to the problems such as a decision support, active databases, a data warehouse, temporal databases, internet applications, etc. In this paper an analysis is addressed that formulates the cost functions and evaluates them as the propagation subjects, objects, and update policies. The propagation subject can be the client side, server side, and the third; and the objects can be base tables, semi-base tables, and delta files; And the update policies can be the immediate, deferred, and periodic ones, respectively.

1. Introduction

Maintaining materialized views and/or replica can be analyzed as a client-server architecture that extracts the changes of the distributed source data and transfers them to the relevant target sites. View maintenance and materialized views are considered to be important for suggesting solutions to the problems such as a decision support [TGH95], active databases [MZ97], a data warehouse [LG96, ZG+95], temporal databases [OS95], mobile computings [WSD95], concurrent engineering with cooperative works [GL96], internet applications, etc. [RO91].

In this paper an analysis is addressed that formulates the cost functions and evaluates them according to the propagation subjects, objects, and update policies. At first, the propagation subject can be the client side, server side, and the third. (1)Server side: the changes are detected and delivered in the server side. It is called as a push type method. (2)Client side: The changes and processes are made by client's requests. It is called as a pull type method. (3)The third: All the processes are performed by the other side such as a rule trigger, and is called as a hybrid method. R.Goldring discussed that data replicators described above adopt push and pull type data propagation techniques respectively [GO94].

The extracted and/or the transferred objects that can be classified as three ways such as the base tables themselves, the indices that we call them 'semi-base tables', and the delta files that denotes the changed portions of the relevant base tables as follows: (1)Base tables: The view definitions are re-executed every time. (2)Semi base tables: The indices of tables are gathered and are sent incrementally. (3)Delta files: The delta files or called differential files are delivered differentially. W.Lee et. al. in [LPS96] and [LP98] showed that 'the differential file scheme' is superior to base table or semi-base table scheme in normal situations such as about less than 0.5 screening, under 1Mbps communication speed, and when the differential file size is less than that of base table.

Three view maintenance policies have been suggested: (1)Immediate updates: The view is updated immediately that the transactions include the view maintenance operations. (2)deferred updates: The view is updated by the view invoke transaction that is separate from the update transactions. (3)Periodic updates: The view is updated periodically such as every 5 minutes, or a day, or bimonthly, etc. [CK+97] suggested several algorithms for multiple view maintenance and compared a number of view maintenance policies. In this paper an analysis is addressed that formulates the cost functions and evaluates them according to the three criteria and policies.

2. A View Maintenance Architecture

We assume here that the subject that who has the update initiative is the place where the tables are stored. The server side update means that all the tables and/or the related projection and join operations are made in the server site whereas the client makes the request queries and gets the results. The client side update means that all the tables are replicated in the client site, whereas they are updated synchronously as in the data replicators such as *Data Distributor (DEC), Data Propagator Relational (IBM), Replication Server (Sybase), Symmetric Replicator (ORACLE)*, etc. The hybrid update one has another update schema fundamentally in another sites that has a role to update views and replicas such as the Global Update File [SP89].

The views that to be updated immediate have to be consistent with the relations they are defined over, whereas the deferred views need not to be consistent with the base relation transactions but have to be consistent at query time. Periodic views are requested to be consistent with the state of the derived relations at the time of predetermined update periods.

It is necessary that the view refresh requires extra works to be done in response to those update needs. If the views are an immediate views, then the transactions should have additional loads. In case of deferred views, an additional index maintenance mechanism such as view caches [RO91] will be needed. In case of periodic updates, the additional work is needed in the maintenance of logs such as called the relevant logs [DGA96] or the differential files in [SP89] and [LP98].

3. View Update Algorithms

Here several algorithms that update the views that have update policies such as immediate, deferred, and periodic, and the update strategies such as base table method as Semi-join and Delta File method, respectively. It is assumed that relation R_i^A named A is located in site i and R_j^B named B in site j are to be jc .ed at site k where the *Join Master File* (or the views) locates, normally for $i \neq j \neq k$. Here, for the sake of convenience, we set A_f as a foreign key of relation R_i^A so that it can be relevant to the primary key of relation R_j^B. The algorithm *DeltaFile,* therefore, is as follows.

```
Algorithm: update deferred view
 Input: Update command U to the view V
 Output: Consistent view V after the update command U
 Method:
 If V needs to be updated
        compute the incremental changes of V;
        update the changes to the relevant base relations;
        update the view V;
 end.
△
```

```
Algorithm: semi join
 Input: Update command U to the View V
 Output: Consistent View after the update command U
```

```
Method:
  If U is the update(insert, delete, and modify) of base tables
        for each view V to be updated do
                If ever, then send the primary key of table $R_i$ to site j where $R_j$ is located.
                If ever, then send the tuples of $R_j$ that are matched with that of $R_i$
                Join $R_i$ and the tuples sent from $R_j$
                send them to the sites where materialized views are located.
                Append the systime;
        endfor
  Else
        Append the systime;
end.
△
```

```
Algorithm: update DeltaFile
 Input: Update command U to the view V
 Output: Consistent view V after the update command U
 Method:
  If V needs to be updated
  Begin
    /* In site i */
    CREATE $DeltaFile_{ij}^{v}$ AND APPEND as
    $DeltaFile_{ij}^{v}[TID] \leftarrow T_i^d[TID]$
    $DeltaFile_{ij}^{v}[A_u] \leftarrow T_i^d[A_u]$
    $DeltaFile_{ij}^{v}[TS] \leftarrow T_i^d[TS]$
    $DeltaFile_{ij}^{v}[OP] \leftarrow T_i^d[OP]$
    Stop;
    If $T_i^d[OP]$ = ins or mod
        THEN Send $DeltaFile_{ij}^{v}$ to relevant join site
    If $T_i^d[OP]$ = del
        THEN
        Select all the tuples that were deleted as $DeltaFile_{ij}^{v}$ for
        $DeltaFile_{ij}^{v}[TID] \leftarrow T_i^d[TID]$
        $DeltaFile_{ij}^{v}[TS] \leftarrow T_i^d[TS]$
        $DeltaFile_{ij}^{v}[OP] \leftarrow T_i^d[OP]$
        Send $DeltaFile_{ij}^{v}$ to site $k$  /* Join Master File site */
    Else
        $DeltaFile_{ij}^{v}[TS] \leftarrow T_i^d[TS]$
        Send the $DeltaFile_{ij}^{v}[TS]$ to site $k$
        /* The time-stamp only will be sent to the Join Master File */
  Stop;
  /* In site $j$ */
  Begin
    If $DeltaFile_{ij}^{v}[A_R] = T_j^d[TID]$
    /* If the foreign keys will be matched to the differential file in site $j$ */
        $J1 = DeltaFile_{ij}^{v}[TID] \bowtie T_j^d$
        /* Join with the differential file */
    Else
        $J2 = DeltaFile_{ij}^{v}[TID] \bowtie R_j$
        /* Join with the base Table in site $j$ */
        Set $JF = J1 \cup J2$
        Send $JF$ to site $k$
  Stop;
  /* In site $k$ */
  Begin
    If $DeltaFile_{ij}^{v} \cup J \neq \emptyset$
    THEN Update the Join Master File (or the join view) with $DeltaFile_{ij}^{v}$ and $JF$
    ELSE Update the time-stamp with $DeltaFile_{ij}^{v}[TS]$
  Stop;
End.
```

△

3. Parameters and Cost functions

3.1 Cost functions

General parameters for distributed databases:

Ω	: the set of site index, for $i \in \Omega = \{1, 2, \ldots, n\}$

B	: Page size (bytes)
$\bowtie$	: the join operator
SF	: *Semi-join* factor
C_{IO}	: the input and output cost(ms/block)
C_{comm}	: the transmission cost(bits/s)
$H_{B\ Si}$	: the height of B^+ tree at S_i site
R_i, $\acute{R}_i$	: the base table in site i and the screened one respectively.
dR_i, $d\acute{R}_i$	: the delta file of the base table R_i and that of the screened one respectively.

Parameters for the number of tuples:

Pr_{DF_Rj}	: the probability that all the tuples needed to join in DF_{Rj}
$N(R_i)$, $N(R_j)$	: the number of R_i tuples per page ($=B/W_{Ri}$) and that of R_i
$N(d\acute{R}_i)$, $N(d\acute{R}_j)$	: the number of delta file of the screened R_i and R_j respectively.
$N(R_i[PK])$,$N(R_j[PK])$	: the number of primary key's the R_i and R_j respectively.

Parameters for the reduction procedure:

α_s	: screen factor for view predicate
W_{ins}, W_{del}, W_{mod}	: the width of *MF* tuples with OP = *ins*, *del*, and *mod* respectively
W_{Ri} , W_{DDJj} , W_v	: the width of R_i, DDJ_j, and materialized view V_i respectively.
W_B	: the width of the B^+ tree.

3.2 Related Cost functions: Yao's and Referential function

We assumed here that R_i and R_j have clustered indexes on the attributes to be joined. The communication costs and the I/O costs without computing costs are considered. (DB relations are generally stored in the costless Disks, and the portion of computation time at the main memory is known to below 1% in the total cost)

3.3.1. Yao's Function

Yao suggested a cost function that for accessing k records randomly distributed in a file of n records stored in k pages, a formula for the expected optimal number of page accesses is given in [YAO77]:

$$\Phi(k,n,m) = m\left[1 - \sum_{i=1}^{k} \frac{(nd-i+1)}{(n-i+1)}\right], \; where \; d = 1 - 1/m \qquad (1)$$

This formula assumes that the scheduling of page accesses is optimal, that is, the same page is not accessed more than once.

3.2.2. Referential Function

The expected number of blocks to be searched in a referenced file is· as follows where the number of records related is k. The proof of this function is abbreviated and will be made next time.

$$\Psi(k,n,m) = \sum_{j=1}^{n} jp(1-p)^{j-1}, \text{ for } p = \sum_{i=1}^{k} \frac{(1-p)^{i-1}}{n^{i}} \qquad (2)$$

3.3 The Cost functions:

Cost functions and their explanations are denoted as follows. The functions are denoted by NIO's and NCOM's in the DeltaFile method, DIO's and DCOM's in Deferred view method, and semijoin by SIO's and SCOM's, respectively.

Cost functions of the DeltaFile method:

NIO1 = Cost of reading delta file tuples and sort them from dR_i = $3*C_{I/O}N(d\acute{R}_i)*W_R/B$

NIO2 = Cost of reading and sorting $d\acute{R}_j$ tuples = $3*C_{I/O}N(d\acute{R}_i)W_R/B$

NIO3 = Cost of reading and sorting DeltaFile = $3*C_{I/O}*N(d\acute{R}_i)W_{DeltaFilej}/B$

NIO4 = Cost of reading $N(R_j)$ tuples and sort them in DeltaFile = $3*C_{I/O}*N(R_j)*W_R/B$

NIO5 = Cost of accessing the B^+ tree and view table at the view site = $C_{I/O}[(H_{B\_SMi} - 1) + \Phi[a_sN(R_i), a_sN(R_i)W_B/B, N(R_i)]] + 3*C_{I/O}*\Phi[a_sN(d\acute{R}_i), a_sN(R_i)W_R/B, N(R_i)]$

NCOM1 = Cost of transmitting tuples to site i to site j = $8N(d\acute{R}_i)*W_R/C_{comm}$

NCOM2 = Cost of sending joined tuple to View site + Cost of sending the change of Relation Rj to the view = $8*N(d\acute{R}_i)*Wmvi/C_{comm}$ + $8[N(R_{j.del})W_{del} + N(R_{j.mod})W_{mod}]/C_{comm}$

The costs of the algorithm *DeltaFile* method can be $\sum_{i=1}^{5} NIOi + \sum_{j=1}^{2} CCOMj$.

6. Summary

View maintenance and materialized views are considered to be important for suggesting solutions to the problems such as a decision support, active databases, a data warehouse, temporal databases, internet applications, etc. Maintenance of materialized views is applied to the replica update problem without extra-efforts. It can be viewed to the client-server architecture that extracts the changes of the distributed source data and transfers them to the relevant target sites. In this paper an analysis is addressed that formulates the cost functions and evaluates them as the propagation subjects, objects, and update policies. The propagation subject can be the client side, server side, and the third; and the objects can be base tables, semi-base tables, and delta files; And the update policies can be the immediate, deferred, and periodic ones, respectively.

In this paper a join query is a testbed not only because it can cover almost all operations in relational database systems but also because it is one of the most time-consuming and data intensive operations, especially for a distributed environment. The cost functions and their performance analyses indicate that the delta file method with periodic update scheme is superior to base table methods in normal situations, but is closely dependent on the number of differential tuples, the screen factor, and the transmission speed. Immediate update is not scalable with respect to the number of views, so a system cannot define many immediate views. Deferred update is, however, distinguished as a base table method at the point of the updating of the relevant views, so it can appropriately be called a 'semi base table' method. If one can tolerate asynchronous updated data and stale data set, then periodic update with delta file will be the best alternative.

References

[BLT86] J.A.Blakeley, P.Larson, and F.W.Tompa, "Efficiently updating materialized views," *in Proc. ACM-SIGMOD Conf. Management of Data*, Washington D.C., May 1986

[CK+97] L.Colby, A.Kawaguchi, D.Lieuwen, I.Mumick, and L.Ross, "Supporting Multiple View Maintenance Polices," *Proceedings of the ACM SIGMOD*, 1997, pp.405-416.

[DGA96] D.Chao, G.Diehr, A. Saharia, "Maintaining Remote Join Materialized Views," *Working Paper*, San Francisco State University, 1996.

[GL96] T.Griffin, and L.Libkin, "Incremental Maintenance of Views with Duplicates," *ACM SIGMOD'95*, San Jose, 1996, pp. 328-339.

[GO94] R.Goldring, "A Discussion of Relational Database Replication Technology," *InfoDB*, Spring

1994.

[LG96] W.J.Labio, and H. Garcia-Molina, "Efficient Snapshot Differential Algorithm for Data Warehousing," Proceedings of the 22nd *VLDB Conference*, Bombay, India, 1996, pp. 63-74.

[LP98] W.Lee. and J.Park, "An Asynchronous Differential Join in Distributed Data Replications," to appear in *the Journal of Database Management*, 1998.

[LPS96] W.Lee., J.Park, and S.Kang, "Replication Server Scheme in Distributed Database Systems," *Proceedings of `96APDSI conference,* Vol. 3, 1996, pp. 1275-1282.

[MZ97] I.Motakis and C.Zaiolo, "Temporal Aggregation in Active Database Rules," in *Proc. ACM-SIGMOD*, June 1997, pp. 440-451.

[OS95] G.Ozsoyoglu, and R.T.Snodgrass, "Temporal and Real-Time Databases: A Survey," *IEEE Transaction on Knowledge and Data Engineering*, Vol. 7, No. 4, Aug. 1995, pp. 513-532.

[RO91] N.Roussopoilos, "An Incremental Access Method for ViewCashe: Concepts, Algorithms, and Cost Analysis, *ACM Transactions on Database Systems*, Vol. 16, No. 3, Sept. 1991.

[SP89] A.Segev, and J.Park, "Updating Distributed Materialized Views," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 1 , no. 2, June 1989.

[TGH95] V.J.Tsotras, B.Gopinath, and G.W.Hart, "Efficient Management of Time-Evolving Databases," *IEEE Transaction on Knowledge and Data Engineering*, Vol. 7, No. 4, Aug. 1995, pp.591-608.

[WSD95] O.Wolfson, P.Sistla, S. Dao, K.Narayanan, and R. Raj, "View Maintenance in Mobile Computing," *ACM SIGMOD Record*, Vol. 24, No. 4, December 1995. pp. 22-27

[YAO77] S.B.Yao, "Approximating block accesses in database organizations," *Communications of the ACM*, Vol. 20, April 1977.

[ZG+95] Y.Zhuge, H.Garcia-Molina, J.Hammer, J.Widom, "View Maintenance in a Warehousing Environment," *ACM SIGMOD'95*, San Jose, 1996, pp. 316-327.