Temporal Query Processing using Spatially-Partitioned Method

Duk-Ho Chang
Electronics and Telecommunication Research Institute
YuSung PO Box 106, Taejon, Korea, 305-333
email : dhchang@etri.re.kr

Jong Soo Kim and Myoung Ho Kim
Department of Computer Science
Korea Advanced Institute of Science and Technology
373-1, KuSung-Dong, YuSung-Gu, Taejon, Korea, 305-701
email : {jskim,mhkim}@dbserver.kaist.ac.kr

Abstract

In this paper, we propose a general spatially-partitioned temporal operation model, called 3D-TOM, and partition-based join algorithms for various types of temporal join in the proposed model. The temporal operators addressed in this paper include overlap, precede, follow, and equal. Some factors which affect the performance of the proposed method are also discussed.

1 Introduction

Conventional databases that reflect snapshots of the constantly evolving real world, are not appropriate for applications in which past and/or future data are also required. In these circumstances, what is needed is a database system that fully supports the storage and querying of information that varies over time. A Database that maintains past, present, and future data is called a *temporal database*[TCG+93].

Research in temporal databases has largely focused on extensions of existing data models to handle temporal operations[Gar88]. An important aspect of these extensions is the introduction of new temporal operators. Temporal operators can significantly enhance the expressive power of a database while query optimization is a critical issue in practicality of a temporal database management system(TDBMS).

So far a number of strategies to process temporal operators for *temporal join*, which is one of the most expensive operations in TDBMS, were proposed[GS91, SSJ94, LOT94]. In general, join evaluation algorithms fall into three basic categories, nested-loops, sort-merge, and partition-based. Among them a partition based approach, which has a great potential for effective temporal join handling, is our main concern. However, without some replications, partition-based temporal join methods needs an efficient mechanism to identify appropriate sets of partitions. In this case, there can be a performance bottleneck because a partition of one relation needs to be compared with more than one partition of the other relation for most temporal operators. Therefore, a mechanism which can identify the *counterparts* of a partition – the partitions of a joining relation which need to be compared with a partition of the other joining relation – is essential to a partition-based temporal join method.

We observe that the two dimensional space proposed in [LOT94] needs to be extended to three dimensional space to cover various types of temporal join. In this paper, we propose a three dimensional spatially-partitioned temporal operation model (3D-TOM), which can be applied to process a variety of temporal operations. Since join operation is generally considered as the most expensive one, we will mainly focus on *time join* and *time equijoin* method. We also give algorithms for a number of temporal comparison operators which are specified in the join process. The operators addressed here include *overlap, contain, precede, follow,* and *equal*. Formal definitions of temporal join in which the operators are used can be found in [TCG+93].

The rest of the paper is organized as follows. In section 2, we propose three dimensional spatially-partitioned temporal operation model(3D-TOM). Section 3 describes temporal join processing strategies under 3D-TOM. In section 4, we discuss the performance issues of our study. We conclude with a summary of our contribution and further research area in section 5.

2 3D-TOM

3D-TOM is a general operation model based on the spatial partitioning of underlying data. The model provides a framework in which various temporal operations can be processed effectively. We present the structure of 3D-TOM and data mapping scheme used in 3D-TOM in this section. We also discuss the par-

titioning of the three dimensional space.

First we give the definition of 3D-TOM. We assume that each temporal data can provide the start time(Ts), interval length($Te - Ts$), and the join attribute value(J) as in [TCG+93]. We also assume that the time dimension is a sequence of discrete time instants $T_1, T_2, \ldots T_{now}$

Definition 1 *3D-TOM is an operation model which can be used for various temporal operators. It is based on a three dimensional space on which temporal data are mapped along the three axes, i.e. Ts, Te, and J.*

Fig. 1 illustrates the conceptual structure of 3D-TOM.

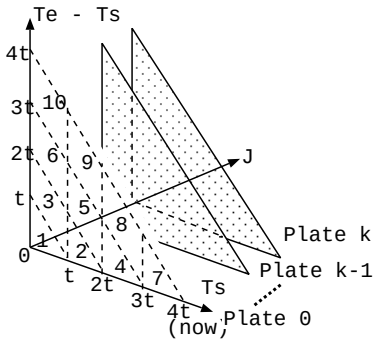

Figure 1: The concept of 3D-TOM

In 3D-TOM, we call a two dimensional plane which can be obtained for a fixed J value as *a plate*. In other words, 3D-TOM is a set of consecutive plates. Based on the previous work in [LOT94], each plate of 3D-TOM has the following properties:

- The temporal data are mapped into a plate according to their T_s and $T_e - T_s$.
- Each plate is split into partitions by the lines $x = 0, y = 0, x + y = T_i, x = T_i, i = 1...now$.
- The *partition_id* of a partition to which a data belongs can be identified using the following two mapping function[LOT94]:

$$f : (Ts, Te) \rightarrow (\lceil \frac{Te}{t} \rceil, \lfloor \frac{Ts}{t} \rfloor)$$

$$g : (a, b) \rightarrow k \ where \ k = \frac{a \cdot (a+1)}{2} - b$$

The partition_id of a data whose time interval is [Ts, Te] is given by $g(f(Ts, Te))$. In the mapping function, t means the time unit.

The plate number of a temporal data can be identified easily by introducing a hash function on the join attribute. Therefore, we can describe a point to which a temporal data is mapped by providing the plate number and the partition_id of the data in 3D-TOM.

3 Processing of Temporal Join in 3D-TOM

In this section, we describe various temporal join processing algorithms in the proposed model, 3D-TOM. Since time equijoin can be considered as time join on a specific plate in 3D-TOM, we first discuss time join processing in 3D-TOM. We also present a general temporal join processing algorithm for both time equijoin and time join in 3D-TOM.

The proposed method has the following features:

- Data objects are mapped to points in a three dimensional space according to their time interval and join attribute value.
- Temporal join is performed by partition-based approach. A partition of one relation is only compared with certain partitions of the other relation to find matches.
- The partitions to be compared with a given partition can be identified easily through simple calculation.

3.1 Time Join on a 2-dim. Plate

Each plate in 3D-TOM is conceptually identical to the two dimensional time space which is proposed in [LOT94]. Therefore, we can construct algorithms for various types of time join on a plate based on the time-intersection join algorithm in [LOT94].

Assume that two joining relations R and S are partitioned on the time attributes using the mapping scheme presented. As we mentioned earlier, partition R_i of a relation R needs to join with more than one partition of relation S depending on the operator used. Therefore, a time join algorithm requires mechanisms which can identify the counterparts of a joining partition for a temporal comparison operator used. Table

Table 1: 5 temporal comparison operators

Operator	Predicate
d1 overlap d2	$d1.Ts \leq d2.Te$ $\wedge$ $d1.Te \geq d2.Ts$
d1 precede d2	$d1.Te < d2.Ts$
d1 follow d2	$d1.Ts > d2.Te$
d1 contains d2	$d1.Ts \leq d2.Ts$ $\wedge$ $d2.Te \leq d1.Te$
d1 equal d2	$d1.Ts = d2.Ts$ $\wedge$ $d1.Te = d2.Te$

1 shows that each temporal comparison operator can be described in a form of predicate involving time attributes of data. Since all the information we have is a partition_id of a partition, we need a way to represent the specific time attribute values of a partition using the given partition_id. To do this, we first define *level* and *height* of a partition as follows:

Definition 2 *The level l of a partition having partition_id k is defined as the largest integer that satisfies $k - \frac{l \cdot (l+1)}{2} > 0$. The height h of a partition k is defined as $h = k - \frac{l \cdot (l+1)}{2}$.*

Fig. 2 shows an example of levels and heights of partitions.

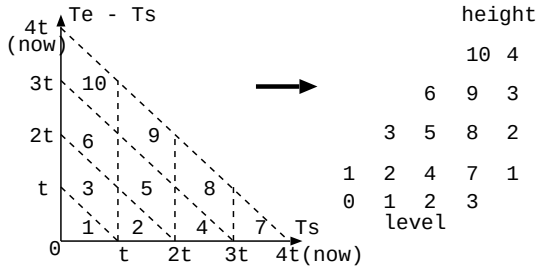

Figure 2: Level and height of partitions

We can represent the maximum and minimum values of time attribute in a partition with the level and the height of a partition. This representation is given in table 2. In table 2, Max(T)/Min(T) means the largest/smallest possible value of T in a given partition and l, h mean the level and the height of the partition. The result presented in table 2 can be derived easily. If we consider the two dimensional space as a Cartesian coordinate, the problem can be transformed into a problem of finding minimum/maximum of x (Ts) and $x + y$ (Te) on the region occupied by a partition.

Table 2: Representation of time attribute values

Value	Representation
Min(Ts)	$l - h + 1$
Max(Ts)	$l - h + 2$
Min(Te)	l
Max(Te)	$l + 1$

Now we are ready to identify the counterparts of a partition for an operator using the level and the height. We first present special conditions,called *counterpart condition*, that a partition should satisfies to be a counterpart of the given partition for each operator. These conditions can be constructed from the predicates in table 1 without much effort. For example, given two partitions R_i and S_j of two joining relations R and S, S_j needs to satisfy the following condition to be a counterpart of R_i for *precede* operator:

$$S_j.Min(Te) < R_i.Max(Ts)$$

In other words, we have to compare S_j with R_i if there is any possibility of existing data in each partition which can satisfy the counterpart condition.

In the same manners, we can construct the counterpart conditions for other operators. The results are given in table 3. In table 3, R_i and S_j are partitions of two joining relation R and S. We omit the case of *equal* operator because we can present a condition involving the level and the height directly.

Table 3: Counterpart conditions for 5 temporal operators

Operator	Condition
precede	$S_j.Min(Te) < R_i.Max(Ts)$
follow	$S_j.Max(Ts) > R_i.Min(Te)$
overlap	$S_j.Max(Te) \geq R_i.Min(Ts)$ $\wedge\ S_j.Min(Ts) \leq R_i.Max(Te)$
contain	$S_j.Max(Ts) \geq R_i.Min(Ts)$ $\wedge\ S_j.Min(Te) \leq R_i.Max(Te)$
equal	–

Table 4: Counterpart conditions of partition s for partition r

Operator	Condition
precede	$l_s < (l_r - h_r + 2)$
follow	$(l_s - h_s + 2) > l_r$
overlap	$(l_s + 1) \geq (l_r - h_r + 1)$ $\wedge\ (l_s - h_s + 1) \leq (l_r + 1)$
contain	$(l_s - h_s + 2) \geq (l_r - h_r + 1)$ $\wedge\ l_s \leq (l_r + 1)$
equal	$(l_s = l_r) \wedge (h_s = h_r)$

From the results presented in table 2 and table 3, we can derive the counterpart conditions involving the length and the height of partitions for each temporal comparison operator. Table 4 shows these conditions. In table 4, l_p and h_p mean the length and the height of a partition whose partition_id is p.

The time join algorithm which can handle the five temporal comparison operators is described as follows. In the algorithm, we assume that two joining relations are partitioned on time attribute and their partition_ids are calculated in advance. Moreover we assume that the number of partitions of R and S are all n.

Algorithm *SpatiallyPartitionedTimeJoin*
in R,S : joining relations ;
OP : a temporal comparison operator
begin
for each partition i **in** R **do**
load partition i;
compute level and height of i;
Set := $\emptyset$;
for j := 1 **to** n **do**
compute level and height of j;
if j satisfies the counterpart
condition of S for i **then**

```
            Set := Set ∪ {j};
         end if
      end for
      for each partition j in Set do
         load partition j;
         for each pair of data in i and j do
            time join of the pair of data;
         end for
      end for
   end for
end
```

3.2 Time Equijoin in 3D-TOM

So far we have presented the join algorithm for time join which can be used for a variety of temporal comparison operators. The time equijoin in 3D-TOM can be processed easily using the proposed time join algorithm. The general temporal join algorithm for various temporal operators is given below.

```
Algorithm TemporalJoin_In_3D-TOM
in R,S : joining relations ;
   OP : a temporal comparison operator
   J : join type
begin
   if J is "time join" then
      PlateSet := all plates in the space;
   else /* Case of "time equijoin" */
      PlateSet := a plate chosen by the
            join attribute value;
   end if
   for each plate p in PlateSet do
      SpatiallyPartitionedTimeJoin() on p;
   end for
end
```

4 Performance Consideration

As 3D-TOM is an extension of two-dimensional spatially partitioned method, we can analyze the performance of the proposed algorithms based on the extensive performance study of [LOT94]. The performance of spatially partitioned method can be summarized as follows:

- The partition-based join reduces the number of partition pairs to be joined to about 70compared to nested-join.
- The partitioning process of relation has to read and write the whole relation at least once, which is some overhead. Partition-based algorithms are sensitive to memory size [SSJ94, LOT94]. The partition-based algorithm performs better than the nested-loops join at less than about 20 M bytes. At a larger memory size, the nested-loops join slightly outperforms.
- The lifespan of tuples also affects the performance. As the lifespan increases, the tuples overlap more buckets. The number of partitions is also related to the performance of the algorithm. When two relations are partitioned with the same interval gives the best performance.

Some additional overhead caused by the 3D- TOM is the generation of plates. In algorithm *TemporalJoin_In_3D_TOM*, getting PlateSet needs additional computation, however, it can be gotten easily by introducing a hash function on the join attribute. So the additional overhead is negligible and the overall performance of the proposed algorithm is expected to be almost same with the result of [LOT94].

5 Conclusion

In this paper, we have discussed several issues of temporal join processing. The contribution of this work can be summarized as follows:

- We have proposed 3D-TOM that can handle various temporal operations effectively. In 3D-TOM, temporal data can be mapped into a point in three dimensional space by using their time attributes and join attribute value.
- We have presented a general algorithm for temporal join, which is the most expensive operation, in the proposed model. In the proposed algorithm, the inherent problem of partition-based temporal join methods can be resolved through simple computation.

Though we have addressed some aspects that affect the performance of the proposed model, intensive performance study remains as future work. Spatial indexing mechanisms for the proposed model also needs to be investigated.

References

[Gar88] S. Gardia. A Homogeneous Relational Model and Query Language for ER Databases. *ACM Transactions on Database Systems*, 13(4), 1988.

[GS91] H. Gunadhi and A. Segev. Query Processing Algorithms for Temporal Intersection Join. In *Proceeding of the 7th ICDE*, 1991.

[LOT94] H. Lu, B. Ooi, and K. Tan. On Spatially Partitioned Temporal Join. In *Proceedings of the 20th VLDB Conference*, 1994.

[SSJ94] M. Soo, R. Snodgrass, and C Jensen. Efficient Evaluation of Valid-time Natural Join. In *Proceeding of the 10th ICDE*, 1994.

[TCG+93] A. Tansel, J. Clifford, S. Gadia, S. Jajodia, A. Segev, and R. Snodgrass. *Temporal Databases:Theory, Design, and Implementation*. Database Systems and Application Series. Benjamin/Cummings, 1993.