

## TIME INDEX ACCESS STRUCTURE FOR TEMPORAL DATABASES HAVING CONCURRENT MULTIPLE VERSIONS

### BACKGROUND OF THE INVENTION

Research in temporal computer databases has been mostly concerned with defining data models and operations that incorporate the time dimension. For example, extensions to the relational data model and its operations for handling temporal data have been discussed by Snodgrass and Ahn (R. Snodgrass and I. Ahn, "A Taxonomy Of Time In Databases", *ACM SIGMOD Conference*, May 1985) and Gadia and Yeung (S. Gadia and C. Yeung, "A Generalized Model For A Temporal Relational Database", *ACM SIGMOD Conference*, June 1988). In addition, some work has been presented by Segev and Shoshani (A. Segev and A. Shoshani, "Logical Modeling Of Temporal Data", *ACM SIGMOD Conference*, June 1987) and Elmasri and Wu (R. Elmasri and G. Wu, "A Temporal Model And Language For ER Databases", *IEEE Data Engineering Conference*, February 1990) defining temporal extensions to conceptual data models and query languages. Such temporal data models define powerful operations for specifying complex temporal queries. Although there has been some research in the area of defining storage structures and access paths for temporal data, for example by Lure (V. Lum, "Design Dbms Support For The Temporal Dimension", *ACM SIGMOD Conference*, April 1984), Rotem and Segev (D. Rotem and A. Segev, "Physical Organization Of Temporal Data", *Proceedings Of IEEE Data Engineering Conference*, 1987), and Kolovson and Stonebraker (C. Kolovson and M. Stonebraker, "Indexing Techniques For Historical Databases", *Proceedings Of IEEE Data Engineering Conference*, February 1989), these works do not provide indexing schemes for supporting high-level temporal operators such as described by Gadia et al. and Elmasri et al., cited above.

Storage techniques for temporal data, such as proposed by Lum, index or link the versions of each individual object separately. In order to retrieve such object versions that are valid during a certain time period, it has been necessary to first locate the current version of each object, and then search through the version index (or list) of each object separately. A method proposed by Rotem et al., noted above, allows a search based on time using a multi-dimensional partitioned file, in which one of the dimensions is the time dimension. However, in such a scheme temporal data items are associated with a time point rather than a time interval, and hence it is not useful when a search involving time intervals is required.

In order to conduct an efficient computer search operation in a temporal database, some effective form of indexing is required. However, since conventional indexing schemes assume that there is a total ordering on the index search values, the properties of the temporal dimension make it difficult, for a number of reasons to use traditional indexing techniques for time indexing. First, the index search values, i.e. the valid\_time attribute, are intervals rather than points, because each version of an object is typically valid during a time interval  $[t_1, t_2]$ , and the valid\_time intervals of various object versions will overlap in arbitrary ways. Because one cannot define a total ordering on the interval values, a conventional indexing scheme cannot be used. Second, because of the nature of temporal databases, most up-

dates occur in an append mode, since past versions are kept in the database. Hence, deletions of object versions do not generally occur, and insertions of new object versions occur mostly in increasing time value. In addition, the search condition typically specifies the retrieval of versions that are valid during a particular time interval.

Although the interval-based search problem is similar in many respects to the k-dimensional spatial search problem, the various index methods proposed for k-dimensional spatial search, for example by Ooi et al. (K. Ooi, B. McDonell, and R. Sack-Davis, "Spatial Kd-tree: Indexing Mechanism For Spatial Database", *IEEE COMPSAC 87*, 1987), are not suitable for the time dimension. While these spatial index methods might be adapted to a single dimension, for the most part they support spatial search for two-dimensional objects in CAD or geographical database applications. The index algorithms, such as suggested by Ooi et al., use the concept of a region to index spatial objects, wherein a search space is divided into regions which may overlap with each other, and a sub-tree in an index tree contains pointers to all spatial objects located in a region. Since spatial objects can overlap each other, handling the boundary conditions between regions is quite complex in these algorithms. In temporal computer databases there can be a much higher degree of overlapping between the valid\_time intervals of object versions. For instance, a large number of long or short intervals can exist at a particular time point. Furthermore, the search space is continuously expanding whereas most spatial indexing techniques assume a fixed search space. In addition, temporal objects are appended mostly in increasing time value, making it difficult to maintain tree balance for traditional indexing trees. Thus, because of these added requirements of the temporal over the spatial search, the spatial index algorithms are not suitable for temporal data even where they are directly adapted from two dimensions to a single dimension.

### SUMMARY OF THE INVENTION

The present invention provides a time indexing procedure which is particularly useful with object versioning structured temporal computer databases for the efficient processing of temporal operations requiring reference to time intervals. For example, where it is desired to retrieve object versions that are valid during a given time period, e.g. the names of all employees who worked for the company during 1985, this time index will lead directly to the desired versions, i.e. the names, without requiring the search of a version index for each individual object, i.e. employee, separately. In addition, the time index may be used to efficiently process temporal aggregate functions, as well as temporal WHEN, SELECT, and JOIN operators of Gadia et al., and temporal projection suggested in the earlier-noted work of Elmasri et al.

In a temporal database, the time dimension is usually represented, as described in Gadia et al., using the concepts of discrete time points and time intervals. A time interval, denoted by  $[t_1, t_2]$ , is defined as a set of time instants (points) on a scale of consecutive, regularly occurring time points, where  $t_1$  is the first time instant and  $t_2$  is the last time instant of the interval. The time dimension is represented as a time interval  $[0, \text{now}]$ , where 0 represents the starting time of a database mini-world application, and now is the current time, which is