

TABLE 2

DEPARTMENT Table		
Dept	Manager	Valid_Time
A	Smith	[0,3]
A	Thomas	[4,9]
A	Chang	[10,now]
B	Cannata	[0,6]
B	Martin	[7,now]
C	Roberto	[0,now]

The effect of the operation is to join each DEPARTMENT object with the appropriate EMPLOYEE objects during the time periods when the employees worked for that department. Using the described two-level time index on the Dept attribute of EMPLOYEE retrieves the employees working for each department during specific time periods. The JOIN operation would effectively be as follows:

```

for each DEPARTMENT object do
  begin
    for each version of the DEPARTMENT object to
      begin
        retrieve the Dept value, and valid_time
          [t1,t2] of the version;
        use the EMPLOYEE top-level index to locate the
          time index for the Dept value;
        use the time index to retrieve EMPLOYEE versions
          whose time interval overlaps [t1,t2];
        join each EMPLOYEE version to the DEPARTMENT
          version;
      end;
    end;
  end;

```

The result of this operation would appear as in the following Table 3:

TABLE 3

EMPLOYEE/MANAGER Table			
Name	Dept	Valid_Time	Manager
emp1	A	[0,3]	Smith
emp1	B	[4,6]	Cannata
emp1	B	[7,now]	Martin
emp2	B	[0,5]	Cannata
emp3	C	[0,7]	Roberto
emp3	A	[8,9]	Chang
emp4	C	[2,3]	Roberto
emp4	A	[8,9]	Thomas
emp4	A	[10,now]	Chang
emp5	B	[10,now]	Martin
emp6	C	[12,now]	Roberto
emp7	C	[11,now]	Roberto

The simple data processing computer arrangement depicted in FIG. 5 is typical of database management systems in general and is suitable for practice of the present invention. In the usual manner, the system is under the control of CPU 502 which, operating over bus 503 and utilizing application programs in memory (MEN) 504, directs the addition, deletion, search, and retrieval of data located on disks in database (DB) 508. Object version updates and searches requested at input/output means (I/O) 506, e.g., keyboard and CRT monitor screen, follow the time index structure set out in the present invention to rapidly and efficiently locate, revise, and retrieve the desired data on appropriate disks of DB 508.

A simulation of the performance of the time index was conducted in order to compare it with traditional temporal access structures. The database had 1000 objects, and versions were added based on an exponential distribution for interarrival time. New versions were

assigned to objects using a uniform distribution. Objects were also inserted and deleted using an exponential distribution with a much larger interarrival time than that for version creation. The comparison of the performance of a time index was based on traditional access structures of clustering (all versions of an object are clustered on disk blocks) and using an accession list (each object has an accession list to access its versions based on time), and the number of block accesses needed for an interval query was calculated (an interval query retrieves all versions valid during a particular time period). The results of the comparison indicated that performance for clustering and accession list deteriorates as the number of versions per object grows, whereas using a time index maintains a uniform performance.

The temporal selection query employing the two-level time index of FIG. 4 showed the most dramatic improvement over traditional access structures, since only 16 block accesses were needed compared to over 1000 block accesses with traditional structures. It was also observed that the storage requirements for the two-level index are considerably less than for a regular time index because the versions are distributed over many time trees resulting in smaller buckets for leading entries in the leaf nodes.

The procedures described and variants suggested herein for the practice of this time indexing process and the various other embodiments which will become apparent to the skilled artisan in the light of the foregoing description are all nonetheless to be included within the scope of the present invention as defined by the appended claims.

What is claimed is:

1. A computer-based temporal database management system including a time index which comprises:

a) an ordered series of indexing time points defining time intervals during which at least one of a plurality of concurrent object version in said database is valid; and

b) associated with each indexing point, pointer means identifying all database object versions that are valid at the time represented by said each indexing point.

2. A system according to claim 1 wherein said each indexing time point defines a change of state of said database with respect to an object version.

3. A system according to claim 1 wherein said indexing time points are situated on a scale of regularly occurring time points and represent those scalar time points:

a) at which a database object version begins; and
b) next following the scalar time point at which a database object version terminates.

4. A system according to claim 1 wherein said indexing time points populate the leaf nodes of a B⁺-tree index structure.

5. A system according to claim 4 wherein each said indexing time point has an associated pointer to a bucket of pointers identifying said valid database object versions.

6. A system according to claim 5 wherein each said valid database object version is identified by a bucket pointer.

7. A system according to claim 5 wherein the bucket pointers of selected indexing points identify only beginning and terminating database object versions.