

## METHOD AND DISTRIBUTED DATABASE FILE SYSTEM FOR IMPLEMENTING SELF- DESCRIBING DISTRIBUTED FILE OBJECTS

### FIELD OF THE INVENTION

The present invention relates to a distributed database file system and method for implementing self-describing file objects.

### DESCRIPTION OF THE PRIOR ART

Several known distributed database, or parallel database implementations exist. In known implementations, external constructs must be accessed for operations including hashing, communications information and partitioning. When implementing a distributed database file, or any object that spans multiple systems, there is a need for each piece of the distributed object to have information about the other pieces. In the instance of a distributed database file, each system must have information about the other systems that the file is distributed across, as well as have information about how the data itself is partitioned. If a piece of the distributed file is accessed, and the user wants to retrieve all of the data in the entire distributed file, then the local system has to determine which remote systems to access in order to retrieve all the data. Furthermore, when data is inserted into the file, the file needs to know the partitioning scheme for the data, in order to determine where to store the newly inserted data.

A need exists for an improved method for implementing a distributed database, or any cross-system file object.

### SUMMARY OF THE INVENTION

An important object of the present invention is to provide an improved method and apparatus for implementing self-describing file objects.

In brief, a method and apparatus are provided for implementing self-describing file objects. A node group is created for defining multiple computer systems for storing data. A hash algorithm for applying to data records is identified. A partition distribution map for distributing data to each of the multiple computer systems utilizing a set of predetermined hash algorithm results and remote system information for each of the multiple computer systems are identified. A file object is created in each of the multiple computer systems. Each the file objects includes the hash algorithm, the partition distribution map, and the remote system information.

In accordance with a feature of the invention, a data record is inserted into one of the distributed file objects by receiving the data record, applying the hash algorithm to the received data record, comparing the hash algorithm result with the partition distribution map to identify the particular computer system for the data record, and utilizing the system information to establish connection to that system. The file objects are fully self-describing eliminating the need for additional objects to be addressed, opened, paged into memory or the like.

### BRIEF DESCRIPTION OF THE DRAWINGS

The present invention together with the above and other objects and advantages may best be understood from the following detailed description of the preferred embodiments of the invention illustrated in the drawings, wherein:

FIG. 1 is a block diagram representation of a computer or data processing system of the preferred embodiment;

FIG. 2 is a block diagram representation illustrating a distributed file object of the preferred embodiment;

FIGS. 3 and 4 are logic flow diagrams illustrating the method and apparatus of the preferred embodiment; and

FIG. 5 a block diagram illustrating a computer program product in accordance with the preferred embodiment.

### DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Having reference now to the drawings, in FIG. 1 there is shown a distributed computer or data processing system of the preferred embodiment generally designated by the reference character 10. Distributed computer system 10 includes multiple computer systems 12 labeled #1, #2, and #3, as shown in FIG. 1. Each computer system 12 includes a processor 14, a memory 16 containing a distributed file object 18 of the preferred embodiment, a user input 20, a user interface 22 and a network adapter interface 24. The multiple computer systems 12 are connected to a network 26 via the network adapter interface 24.

In accordance with the preferred embodiment, the database file or distributed file object 18 is arranged for storing data portions distributed over the multiple computer systems 12. Each distributed file object 18 contains enough information to locate all of the other distributed file objects 18. A method of the preferred embodiment for locating data uses a hashing algorithm and predetermined partitioning data. The distributed file objects 18 themselves are fully self-describing and eliminate the need for any additional descriptive or operational information.

In the distributed computer system 10, each of the multiple computer systems 12 can be implemented with various different commercially available computers, such as an IBM PS/2 including a storage file system. Also other types of computer system, whether it be another microcomputer such as an Apple Macintosh, a minicomputer such as an IBM System/390, or a microcomputer connected to a larger computer system such as an IBM AS/400 can be used for any of the multiple computer systems 12 and fall within the spirit and scope of this invention. It should be understood that the computer system 10 includes multiple computer systems 12 with the communication network 26 defined by an local area network, a wide area network, or other communication network between the multiple computer systems 12.

Having reference to FIG. 2, each distributed file object 18 contains a hash algorithm 31 that is applied to new data to determine its hash value and a partitioning scheme or partitioning data 32 that correlates each of possible hash values to a specific one of the systems 12. The hash algorithm 31 is generated in the file object 18 at creation time of the file object 18 and can be customized for the specific attributes of the file's data types. The hash algorithm 31 yields a hash value, which is simply a number from a predefined set of possible numbers. The partitioning data 32 that is stored in each file object 18 is an array of values that maps each of the possible hash values to one of the systems 12 that will hold a portion of the distributed file data 36. Each distributed file object 18 contains communications information 34 for each of the remote systems across which the distributed file object 18 is distributed. Each distributed file object 18 is complete entity. There is no dependency on related objects on the systems 12 that perhaps could get destroyed, corrupted or get out of synch. Everything needed is in the file. There is also the secondary benefit of performance. Once the file is addressed, there is no need for additional objects to be addressed, accessed, opened, paged into memory, or the like. Each of the distributed file objects