

# METHOD, DATA STRUCTURE, AND COMPUTER PROGRAM PRODUCT FOR OBJECT STATE STORAGE IN A REPOSITORY

## BACKGROUND OF THE INVENTION

### 1. The Field of the Invention

The field of this invention is database storage software that preserves the state of software objects for later use by the creating program or some other client. More particularly, the present invention is related to such object persistence databases that have extensibility characteristics to facilitate implementation of custom objects.

### 2. Present State of the Art

With the advent of object-oriented design techniques, many changes have occurred in the software industry. These techniques have influenced virtually every aspect of application development and implementation and currently many different object oriented products and tools exist. This runs the gamut from languages supporting the creation of objects such as C++, to system level mechanisms allowing availability of objects registered thereto, to the operating system itself, and to any and all applications or other clients desiring to access such.

Many times, it is desirable to store an object's state as it is found in an active object so that another active object may be created at a later point in time long after the original active object has been destroyed. Thus, an application may instantiate a number of objects, operate on those objects to access and modify their state, save or persist the object's state in a more permanent storage such as a disk drive, and terminate execution. When the user desires to commence the application program at a later time, the particular stored state may be recalled and placed into newly instantiated objects so that the application may operate on the objects as they were originally stored. Furthermore, a different application or other client may recreate active objects from the persistent store thus allowing the capability and state of the persisted objects to be shared across different program entities.

As mentioned, having a database or repository of such persisted object states allows different clients (i.e., application programs, other objects, other software components) access to such objects so that they may be shared. By placing a high degree of functionality into the objects themselves, much of the development effort is centralized around object design and implementation so as to allow a decreasing amount of effort to be focused in the actual application development, reduce unnecessary duplication of effort, and avoid inconsistent interpretations of and updates to the object state by different applications.

For example, in a software development environment different software tools are used in the process of designing and creating a particular application program. Such tools, however, are related in that they may operate on the same types of information. A compiler and a source code analysis tool will both operate on application source code files and have overlap in certain behaviors for opening and parsing such files. By placing such behavior in a common software object, duplication of effort is eliminated, the individual application development time reduced, and the tools are assured of having the same interpretation of the source code files.

In a multi-client environment, with each client accessing the same repository of persistently stored object states, a

number of problems arise that make it difficult to implement a general purpose common repository. For example, each different application may have application-specific requirements which will necessitate extending the description of the object state in the repository. For this reason, it is advantageous to allow extension of objects that have the object state persistence characteristics.

Extensibility also allows the use of a general object-oriented persistence implementation. In other words, an object may be based through inheritance or other extension mechanism on another object that has the object state persistence characteristics the inherited object needs. In this manner, a general object persistence functionality may be implemented a single time and the system mechanisms for extensibility can be used to propagate such functionality to other custom objects. There are different underlying object systems upon which to base an object-oriented object state repository that each have different configuration and extensibility mechanisms.

While class inheritance can be used in a compiled language environment to propagate the persistence characteristics to other more specific objects, a language inheritance mechanism for implementing such inheritance requires publication of the source code as well as an added compilation step. This occurs because language inheritance is implemented by a source code compiler which has an inherent need to use the source code of the underlying object state implementation in order to make that implementation available to the more specific object. The above is an example of a language based object system with a language extensibility model.

A binary based or executable code object system will overcome part of this problem and allow objects created by different source code implementations to coexist. Ideally, a binary based object system should allow some form of binary extensibility thereby allowing only an operational specification of the object to suffice in accessing and extending capabilities. In practice, however, many systems require that the extension of the object require client understanding of all underlying object functionality and changes therein to add state persistence can require a class to expose its proprietary source code to the extending class, which is undesirable for the developers of the class being extended.

An interface-based object system that supports multiple interfaces allows incremental extension without access to source code being extended as long as the interfaces used by the client operate the same on a binary level. State persistence functionality may be added easily and can provide greater functionality and extensibility to custom objects.

It is also advantageous to access an existing stored object state through an object that has greater functionality than the object whose state was originally stored. When this can be accomplished, a more dynamic development environment results wherein simultaneous development between different parts using the same object repository can occur.

## SUMMARY AND OBJECTS OF THE INVENTION

It is an object of the present invention to provide permanent storage of a software object state in a database so that an object may be recreated at a later time having the stored state used therein.

It is another object of the present invention to place object persistence capabilities on interfaces rather than class templates so as to facilitate expanded extensibility.

Another object of the present invention is to allow easy extensibility of the persistent capabilities to custom objects