

absolutely necessary, i.e. when a property value in a particular object has changed. In lieu of copying objects, a property table maintains a range of versions for which the property value is the same.

A further aspect of the system is that the propagation of relationships to a new version is controlled by the data model. A flag on the relationship is used to determine whether or not the particular relationship should be copied.

A still further aspect of the system is that resolution of objects during a relationship traversal can be customized depending on whether or not an application accessing the objects is version-aware. If the application is version aware, the traversal resolves to a collection of objects versions related to the origin object. If the application is not version aware, a means for resolving the relationship to a particular object is provided.

A still further aspect of the system is that merge behavior is parameterized. When two versions of an object are merged, flags control how conflicts in property values and relationship contents are managed.

Finally, the system provides a workspace that acts as a virtual repository session and provides workspace context and scope to repository objects.

The present invention describes systems, clients, servers, methods, and computer-readable media of varying scope. In addition to the aspects and advantages of the present invention described in this summary, further aspects and advantages of the invention will become apparent by reference to the drawings and by reading the detailed description that follows.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 shows a diagram of the hardware and operating environment in conjunction with which embodiments of the invention may be practiced;

FIG. 2 is a diagram illustrating a system-level overview of an exemplary embodiment of the invention;

FIG. 3 is an exemplary object hierarchy demonstrating various object and attribute relationships operated on by an exemplary embodiment of the invention;

FIG. 4 is a diagram illustrating an exemplary sequence of version creation and version merging;

FIGS. 5A and 5B are diagrams illustrating data structures supporting object versioning according to an embodiment of the invention;

FIG. 6 is a flowchart illustrating a method for updating a property in a versioned object according to an embodiment of the invention;

FIG. 7 is a diagram illustrating an exemplary object relationship;

FIG. 8 is a diagram illustrating interfaces for traversing relationships in version-aware and non-version-aware applications according to an embodiment of the invention; and

FIG. 9 is a system level overview of an exemplary system according to an embodiment of the invention that provides workspaces in an object repository.

DETAILED DESCRIPTION OF THE INVENTION

In the following detailed description of exemplary embodiments of the invention, reference is made to the accompanying drawings which form a part hereof, and in which is shown by way of illustration specific exemplary embodiments in which the invention may be practiced.

These embodiments are described in sufficient detail to enable those skilled in the art to practice the invention, and it is to be understood that other embodiments may be utilized and that logical, mechanical, electrical and other changes may be made without departing from the spirit or scope of the present invention. The following detailed description is, therefore, not to be taken in a limiting sense.

The detailed description is divided into multiple sections. In the first section, the hardware and the operating environment in conjunction with which embodiments of the invention may be practiced are described. In the second section, a system level overview of an embodiment of the invention is presented. In the third section, systems, methods and data structures according to embodiments of the invention are described that support versioning of objects in a repository. In the fourth section, system and methods of various embodiments of the invention that provide workspaces in a repository are presented. The fifth section is a conclusion of the specification.

Definitions

Throughout this application, reference will be made to objects that are created or instantiated by computer software. Such objects will have a data portion associated therewith for storing information, and have methods or functionality associated therewith to provide desired functionality to a client accessing the object. Typically, the methods of the object will be directed in part to manipulating the object's data. Such an abstract object has an associated state that is the cumulative effect of methods operating on the data. It is this state that will be stored by the innovative object state repository as explained in this application.

As used herein, the term "objects," refers to software objects pertaining to a binary object model and that have binary extensibility through wrapping. Furthermore, such objects are interface-based meaning that an object can be used or operated through specific "interfaces" as defined hereafter and an interface-based binary object model will entail objects having multiple interfaces. In this sense, an object is exposed through its interface.

An object may be active or loaded meaning that it is a functional part of a software program or system. An object is said to be persisted when the data portion or properties are stored, though it is more accurate to refer to the state of an object as being persisted. At a later time, an object of the same class may be instantiated and restored to the same state as the original object using the previously persisted object state.

One implementation of a binary object model and system that follows the characteristics of objects used throughout this application and as described above is the Component Object Model or COM as provided by Microsoft Corporation as part of their Object Linking and Embedding (OLE) and ActiveX™ software technology. Reference to COM objects will be made as part of a specific and exemplary embodiment of the present invention. The invention, however, would fit any object model having the relevant characteristics of COM, namely, being an interface-based, binary object model supporting binary extensibility. As an example, the systems and methods detailed below and their equivalents could be adapted for use in a CORBA (Common Object Request Broker Architecture) environment.

As used herein, the term "interface" refers to a specification for a particular and related subgroup of behavior and properties. Behavior or methods are typically a set of software subroutines, with each subroutine having a signa-