

action includes a name of the specialized action, a name of a data structure for input data for the specialized action, and a name of the collection of data elements. The set of operations correspond to methods of a service provider class. In some cases, the set of operations includes select, delete, select by relation, and update operations.

[0012] Embodiments may include one or more of the following. The repository further includes descriptions of relations between pairs of collections of data elements. The first collection has a relation with a second collection of data elements, a description of the relation is stored in the repository, and the relation enables the client program to request the retrieval of data elements of the second collection by specifying data elements of the first collection. The repository is a database. Executing the first operation includes reading the one or more of the attributes of the first collection of data elements from memory storage and sending the attributes to the client program. In some cases, executing the first operation further includes calculating the one or more of the attributes of the collection of data elements and sending the attributes to the software entity.

[0013] Embodiments may include one or more of the following. A description of a collection and common attributes for the collection includes a name of the collection and a description of a data structure defining the attributes. The first operation is a query and executing the first operation further includes searching for individual data elements within the first collection and returning keys representing the individual data elements. In some cases, the repository includes a definition of the query that includes a search parameter structure of the query and an input name defining a key that is used for filtering one or more data from the collection of data. The system further includes enabling the client program to request a service represented in the repository, the service representing the first operation on one or more data elements in a second collection from the collections, and executing the first operation on the one or more data elements in the second collection.

[0014] These and other embodiments may have one or more of the following advantages. There can be independent software lifecycles for service providers and service consumers. Services provided by a service-based software architecture can be reused for different situations. Generic engines in the software architecture can combine services for new applications.

[0015] The details of one or more implementations of the invention are set forth in the accompanying drawings and the description below. Further features, embodiments, and advantages of the invention will become apparent from the description, the drawings, and the claims.

#### BRIEF DESCRIPTION OF THE DRAWINGS

[0016] FIG. 1 is a block diagram of an example logical representation of a business software application.

[0017] FIG. 2 is a view of a network configuration for a business software application.

[0018] FIG. 3 is a block diagram of the business software application of FIG. 1.

[0019] FIG. 4 is a Unified Modeling Language (UML) representation of a structure of a meta model repository.

[0020] FIG. 5 is a flow diagram of a business process.

[0021] FIG. 6 is a diagram showing relations between different aspects for a business software application.

#### DETAILED DESCRIPTION

[0022] FIG. 1 illustrates an overview logical representation of a business software architecture 2, which includes a client 3, a separation layer 5, a repository 7 and backend data 9 and 9'. Client 3 provides a user interface (UI) that enables a user to interact with the backend data 9 and/or 9'. Backend data 9 and 9' can be associated with different backend applications and/or can be arranged and formatted differently from each other. Separation layer 5 separates the front end user interface provided by client 3 from the back end data 9 and 9'. This separation enables client 3 to interact with backend data 9 and 9' in a consistent and similar manner, regardless of the formatting or application-associated differences between backend data 9 and 9'. In other words, separation layer 5 provides a canonical interface to backend data 9 and 9' so that client 3 is configured to interact with separation layer 5 and only needs to be updated if separation layer 5 changes. Changes to backend data 9 and 9' do not necessitate an update to client 3. Further, separation layer 5 is scalable and configured to handle changes and growth to backend data 9 and 9' and any other disparate backend data and backend services that are further connected to separation layer 5.

[0023] As described in more detail below, separation layer 5 is based on a meta model that defines how backend data (e.g., 9 and 9') are represented in separation layer 5. Meta data is stored in repository 7 that describes how the backend data 9 and 9' fit into the meta model representation. Client 3 interacts with backend data 9 and 9' using a generic command set defined by separation layer 5. As described in more detail below, separation layer 5 accesses service providers that perform the generic commands from client 3, using the meta data in repository 7, to effect the requested manipulation of backend data 9 and 9'. The service providers are configurable so that different service providers can be used for different backend data 9 and 9'. Separation layer 5 includes an interface (e.g., a service manager) that hides the characteristics of the corresponding backend data 9 and 9' and also the granularity and distribution of the implementation (i.e., the service providers).

[0024] FIG. 2 illustrates an example implementation of the business software architecture 2. As shown in FIG. 2, the business software architecture 2 includes a first computer 4 and a second computer 6. The computers 4 and 6 each can include a processor, a random access memory (RAM), a program memory (for example, a writable read-only memory (ROM) such as a flash ROM), a hard drive controller, a video controller, and an input/output (I/O) controller coupled by a processor (CPU) bus. The computers 4 and 6 can be preprogrammed, in ROM, for example, or the computers 4, 6 can be programmed (and reprogrammed) by loading a program from another source (for example, from a floppy disk, a CD-ROM, or another computer) into a RAM for execution by the processor. The hard drive controller is coupled to a hard disk suitable for storing executable computer programs, including programs embodying the present invention, and data. The I/O controller is coupled by an I/O bus to an I/O interface. The I/O interface receives and