

specified including a structure describing various options (OPTIONS) of the queries (e.g., for paging) and SELECTIONS.

[0092] Parameters returned by EXECUTE method include a description (DESCRIPTION) of the query, the query result (OUTRECORDS), and a Boolean flag (REJECTED) indicating if the request for the EXECUTE method was rejected or not

[0093] The EXECUTE method returns the results specified by the query parameters. If the INKEYS table parameter is not empty, the result is restricted to the objects that fulfill the query parameters. INKEYS and INPARAM both restrict the query, but are used in different ways. For example, a query can be defined that returns a list of orders not yet delivered. In such an example, the structure INPARAM can specify that only orders from customers with last names from A-D are to be returned. The INKEYS is a table of all orders that have not yet been delivered. OUTRECORDS contains all orders from the relevant customers, in this case with last names A-D, that have not been delivered yet. In one example, the OUTRECORDS result of a query is a disconnected aspect, that is, the aspect is always read-only. No further backend operations can be performed on this aspect. In this example, the received keys can be used as parameters to select other aspect rows using the aspect service provider 34 and, for example, its SELECT method.

[0094] The query relation service provider 46 implements a routine in a service provider (e.g., aspect service provider 34) for an aspect that is the target of a relation. Methods of query relation service provider 46 are indirectly called from the aspect service provider 34 of the source aspect, if the relation is marked as SOURCE_KEYS or ATTRIBUTES.

[0095] Query relation service provider 46 has a SELECT_TARGET method. The method SELECT_TARGET has input parameters as follows. Input parameters include the name (SOURCE_ASPECT) of the source aspect. Optionally, the method also includes an input parameter defining a proxy interface (TARGET) to the target aspect's SELECT method. Specifying the TARGET parameter allows calling the SELECT method of the aspect service provider 34 for the target aspect without directly knowing the aspect service provider 34 for the target aspect. This enables a query relation service provider 46 to be added to a service module without knowledge of the aspect service provider 34 for the target aspect.

[0096] Another input parameter for the SELECT_TARGET method is the relation (RELATION). Another input parameter is a table of fields (INPARAMS) to describe the relation. To allow mass selection, INPARAMS is a table where every row describes a single selection. An INDEX parameter is used to relate the various rows of the INPARAMS structure to the OUTRECORDS rows. Another optional input parameter is a structure (OPTIONS) describing various options of the queries (e.g., for paging).

[0097] The SELECT_TARGET method returns parameters that include the result encoded with the structure of the target aspect (OUTRECORDS), a description of the query result (DESCRIPTION), and a proxy interface to the target aspects SELECT method. Other output parameters include an index (INDEX) to describe the relation between the IPARAMS records and the OUTRECORDS parameter, a

Boolean flag (REJECTED) indicating if the request for the SELECT_TARGET method was rejected or not and return codes (RETURN_CODES).

[0098] The service providers 32, 34, 40, 42, 44, and 46, as described above, enable the following transactional model for the architecture 38. Executing method SELECT of aspect service provider 34 reads from the backend database 24 or reads from a transactional buffer stored in the back-end. Aspect service provider 34 merges data from both sources—the database and its transactional buffer—in a consistent way so that the merge data reflects the updates made so far in this transaction. Next, executing UPDATE, INSERT, MODIFY, or DELETE methods of aspect service provider 34 builds up a transactional buffer. Before actually changing data in the transactional buffer, the service manager 16 has to acquire a transactional lock on the data and read the data under the protection of a lock. There are exclusive, shared, and shared promotable lock modes available using locking service provider 42 as described previously. Locking has to be accompanied by selecting the locked data again under the protection of the lock. Applications can support optimistic locking by providing time-stamped or otherwise versioned data, and merging actual and modified data on the front-end in case of conflicts.

[0099] The BEFORE_SAVE method of the transaction service provider 40 enables all participating service providers to declare if they are ready for saving the transactional buffer. The SAVE method of the transaction service provider 40 finally triggers service manager 16 to save the transactional buffer to the backend database 24.

[0100] The CLEANUP method of the transaction service provider 40 notifies all service providers (e.g., aspect service provider 34) to release all their transactional buffers and enqueue-based locks. If CLEANUP is called with reason 'END', all locks have to be released. If reason is set to 'COMMIT', each service provider can chose to keep its locks. Aspect service provider 34 must not call COMMIT WORK or ROLLBACK WORK internally on its own. The service manager 16 enforces this by automatically aborting the transaction if aspect service provider 34 is trying to commit a transaction.

[0101] The supported locking models and lock policies are as follows. Using policy S, many participants can obtain a shared lock. If a shared lock is obtained on an object, no exclusive lock or SP lock can be obtained. Shared locks can only be used to achieve a consistent view on a larger set of data during read operations. Using policy E, only a single participant can obtain a lock. Using policy SP (shared promotable), many participants can obtain the lock. If a SP lock exists, exclusive locks can only be obtained by participants already having a SP lock on the object. Only one of the participants can upgrade the lock to an exclusive lock. No other participant, who did obtain a lock prior to the upgrade, can upgrade to exclusive even if the first participant did release its lock.

EXAMPLE

[0102] The architecture 38 (of FIG. 3) implements a simple task of creating a new customer, receiving the customer's order of one or more products via GUI 28 and submitting the order to a business process. To support this example, backend database 24 can be implemented using a