

TABLE 6-continued

| <u>SCOL_QUERY definition</u> | | |
|------------------------------|-----|--|
| Field Name | Key | Description |
| PROVIDER_CLASS | | The name of the corresponding query provider class |

[0074] As stated previously, architecture 38 includes six service provider classes (i.e., transaction 40, query 32, aspect 34, action 44, query relation 46, and locking 42) for handling requests from front end application program 12, other than requesting meta data from repository 18, which is handled by repository service provider class 30. To provide services upon request by front end application program 12, service manager 16 directly calls instances of service provider classes. These instances of service provider classes can be located on the same computer (e.g., 6) as service manager 16 or on a different computer.

[0075] The locking service provider 42 can be used to implement a generic lock manager for a single aspect or a set of aspects. Each locking service provider 42 needs to be registered with an aspect. The name of the locking service provider 42 is set in SCOL_ASPECT table in LOCKING_PROVIDER field for each aspect. Locking service provider class has two methods that can be called by service manager 16. These are LOCK and UNLOCK. LOCK takes as input a collection of keys representing business objects to be locked, a name of an aspect representing a class of the business objects, and a lock mode. There are various locking modes depending on the locking capability of the target system. Locking mode can specify "E", "S", or "SP". "E" means an exclusive lock or that only one client can obtain the lock. "S" means a shared lock indicating that any clients can lock and no lock exclusive to one client is possible. "SP" means the same as "S" but a subsequent upgrade to an exclusive lock is possible.

[0076] LOCK method outputs a Boolean value indicating if the request is rejected or not and also outputs a return code. UNLOCK takes as input a collection of keys representing business objects to be unlocked and a name of an aspect representing a class of the business objects to be unlocked. UNLOCK method also outputs a Boolean value indicating if the request is rejected or not and a return code. A call to UNLOCK is rejected if a transactional buffer is already in a "dirty" state, i.e. if any update, insert, delete operation or an action that is not marked as COL_AFFECTS_NOTHING has been issued since the last CLEANUP call. All locks are removed if the CLEANUP method (described below) of the transaction service provider class is called with reason 'END'.

[0077] A transaction is a sequence of information exchange and related work (such as database updating) that is treated as a unit for the purposes of satisfying a request from front end application program 12 to service manager 16 and for ensuring integrity of backend database 24. For a transaction to be completed and changes to database 24 to be made permanent, a transaction has to be completed in its entirety. All of the steps of a transaction are completed before the transaction is successful and the database is actually modified to reflect all of the requested changes. If

something happens before the transaction is successfully completed, any changes to the backend database 24 must be kept track of so that the changes can be undone.

[0078] To handle transactions, the transaction service provider 40 receives notifications on the various states of a transaction between service manager 16, another non-transaction service provider (e.g., 32, 34, 44, 46), and front end application program 12 (or service manager proxy 14 in some cases). These notifications are the transaction service provider 40's methods BEFORE_SAVE, CLEANUP, and SAVE that are called by the service manager 16 during transactions.

[0079] The service manager 16 calls the transaction service provider 40's method BEFORE_SAVE to check if the transactional buffer can be saved. This allows checking if the internal state of the non-transaction service provider is ready for being saved. The method BEFORE_SAVE returns falso if it is not possible to save the transactional buffer, then the transaction end is aborted. Thus, the BEFORE_SAVE method has a BOOLEAN return parameter. BEFORE_SAVE takes a Boolean as an input parameter REJECTED. The transactional service provider 16 can prevent the following save and commit operations by setting the REJECTED parameter to a non-initial value, i.e. to "true". The method BEFORE_SAVE is called within the service manager's 16's sequence of operations triggered by the front-end application 12's SAVE method.

[0080] The SAVE method finally triggers the application to save the transactional buffer to the database 24. By calling SAVE, all internal states of a non-transaction service provider are made persistent—either by direct updates or by creating appropriate calls to the update task. If all service providers in architecture 38 have received a SAVE request, service manager 16 commits the transaction.

[0081] The CLEANUP method tells all non-transaction service providers to release all their transactional buffers and enqueue-based locks. Calling CLEANUP method communicates that all service providers in architecture 38 need to clean up their internal state. CLEANUP takes a REASON string as an input parameter. The REASON field indicates the reason for the clean up operation. This can be either a 'COMMIT' due to a SAVE-operation or the 'END' of the transaction due to the system closing the transaction automatically. There is no guarantee that cleanup is called under failure conditions.

[0082] The action service provider 44 is called by service manager 16 to execute an action for an aspect. The name of action service provider 44 is set in the PROVIDER_CLASS field of SCOL_ASP_ACTION table for a row corresponding to an action. Action service provider 44 has one method EXECUTE. EXECUTE method takes as input parameters an aspect name (ASPECT), a set of keys (INKEYS) specifying which instances of the aspect are acted upon by the action, a generic input parameter (INPARAM), the name of the action (ACTION) to be executed, a set of keys (RELATION_INKEY) for an action acting on a relation, and a name of the relation (RELATION). EXECUTE method returns as output parameters the changed or newly created objects (OUTRECORDS), which have been modified by the action. The objects returned by the OUTRECORDS parameter are transported to application program 12.

[0083] The aspect service provider 34 is called by service manager 16 to provide functionality to read and modify the