WorkingWithTransactions

WorkingWithTransactions

By default, the RDB DAS assumes responsibility for managing transaction boundaries by executing commit and rollback operations against the database. The boundaries of these transactions are as follows:

Reading Data

Queries to the database are always made by calling DAS command's "executeQuery()" method. The associated transaction lives for the duration of this method execution. That is, the transaction can be thought to start when control is transferred to "executeQuery" and the transaction ends just prior to return from this method.

Writing Data

Writes can be accomplished by two DAS APIs. First is calling the DAS "execute()" command against a command the will perform write operations and, in this case, the transaction boundaries are the same as read queries. That is the transaction life matches the life of the "execute" processing. If the method returns without exception then the application can assume that any changes were successfully committed to the database. If an exception is returned then the changes will have all been rolled back.

The second API used by clients is the DAS applyChanges() method (see ChangeSummaryProcessing). Again, the boundaries of the transaction match the boundaries fo the call to the method. The DAS will create a set of INSERT/UPDATE and DELETE statments to flush the graph changes to the database and all of these statements will be executed as part of a single transaction. If the method returns without exception then the application can assume that all changes were successfully committed to the database. If an exception is returned then the changes will have all been rolled back.

External Transactions

The RDB DAS provides a straightforward mechanism for working with external transactions. As mentioned above, the DAS, by default, acts as the transaction manager and commits and rolls back transactions accordingly. However, that DAS can also be configured to not issue commit/rollback and this is the behavior needed to have the DAS participate in a larger transaction since some external party is responsible for these actions. The following pseudo-code illustrates this:

```
tx.begin();
try {
     doSomeJMSStuff();
     doSomeDASStuff();
     tx.commit();
} catch (AnyException ex) {
     tx.rollback();
}
```

In the example above, the DAS has been configured to not managed transactions since this is handled by the calling program.

The DAS is set to behave this way by a piece of configuration (typically via a configuration xml file). The following config file snippet illustrates this setting:

```
<Config xsi:noNamespaceSchemaLocation="http:///org.apache.tuscany.das.rdb/config.xsd" xmlns:xsi="http://www.w3.
org/2001/XMLSchema-instance">

        <ConnectionInfo managedtx="false"/>

        <Command name="get a customer" SQL="Select * from CUSTOMER where ID = 1" kind="select"/>

</Config>
```