How does distributed transaction work?

9.0 Distributed Transaction Requirements

Distributed transactions are managed by a middle-tier application server such as an Enterprise JavaBeans (EJB) server that works with an external transaction manager and also with a JDBC driver. These three parts of the middle-tier infrastructure provide the "plumbing" that makes distributed transactions possible. More specifically, this infrastructure includes the following:

- A transaction manager

- A driver that implements the JDBC 2.0 API, including the Optional Package interfaces XADataSource and XAConnection

- An application server that provides a connection pooling module (not required but almost always included to improve performance)

- A DataSource class that is implemented to interact with the distributed transaction infrastructure

9.1 Steps in a Distributed Transaction

Assumes a network with the following three-tier architecture:

- First tier - a client, such as a web browser

- Second tier - includes a middle-tier application server, such as an EJB server, that implements connection pooling and distributed transactions

- Third tier - two or more DBMS servers

The following list outlines the steps in a distributed transaction.

9.1.1 CLIENT SUCH AS A WEB BROWSER

9.1.1.1 Client Request

- A remote client makes a request, which goes to the middle-tier server. If work is already being done in the scope of a distributed transaction, the request includes an Xid object that identifies a distributed transaction. If not, the middle-tier server begins a distributed transaction, which produces an Xid object. The remote request is passed to the application code.

9.1.2 MIDDLE-TIER SERVER, SUCH AS AN EJB APPLICATION SERVER

9.1.2.1 Application Code to Get a Connection

- The application obtains the initial JNDI context and calls the method Context.lookup to retrieve a DataSource object from a JNDI service provider.

- The JNDI naming service returns a DataSource object to the application.

- The application calls the method DataSource.getConnection.

9.1.2.2 Getting an XAConnection

- The middle tier invokes the method lookup on the connection pool, looking for an XAConnection object that can be reused.

- If an XAConnection object is available, it is returned, and the connectionpool module updates its internal data structure. If there are none available, an XADataSource object is used to create a new XAConnection object to return to the middle tier.

- The connection pooling module registers itself as a ConnectionEventListener with the XAConnection object.

9.1.2.3 Getting an XAResource Object and Associating the XAConnection Object with a Distributed Transaction

- The middle-tier server calls the method XAConnection.getXAResource.

- The JDBC driver returns an XAResource object to the middle-tier server.

- The middle-tier server passes the XAResource object to the transaction manager.

- The transaction manager calls XAResource.start(xid,javax.transaction.xa.TMNOFLAGS).

9.1.2.4 Getting a Connection to Return

- The middle-tier server calls the method XAConnection.getConnection.

- The JDBC driver creates a Connection object that is conceptually a handle for the XAConnection object on which getConnection was called and returns it to the middle-tier server.

- The connection pool module returns the Connection object to the application.Application Code

- The application uses the Connection object to send statements to the data source but may not enable auto-commit mode or call the Connection methods commit or rollback.

- When the application is finished using the Connection object, it calls the method Connection.close.

9.1.2.5 Recycling the Connection

- The Connection object delegates the application's invocation of the method close to the underlying XAConnection object.

- The XAConnection object notifies its listeners that the method close has been called on it.

- The middle-tier server receives the event notification and notifies the transaction manager that the application is finished using the XAConnection object.

- The connection pooling module updates its internal data structure so that the XAConnection object can be reused.

9.1.2.6 Committing the Transaction

- The transaction manager calls XAResource.end(xid,javax.transaction.xa.XAResource.TMSUCCESS).

- The transaction manager calls XAresource.prepare(xid) on each resource manager to begin the two-phase commit process.

- If all resource managers vote to commit, the transaction manager calls XAResource.commit(xid, false). If not, the transaction manager calls XAResource.rollback(xid).

9.2 References

http://download.oracle.com/docs/cd/B10500_01/server.920/a96521/ds_txns.htm

http://blog.endpoint.com/2010/07/distributed-transactions-and-two-phase.html

http://www.oracle.com/technetwork/database/clustering/overview/distributed-transactions-and-xa-163941.pdf