What is Distributed Transaction?

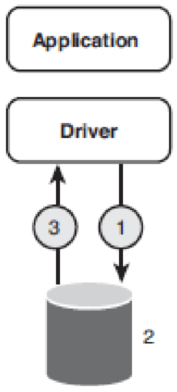
7.0 Distributed Transactions

Transactions can be classified into two groups as Local and Distributed transactions

7.1 Local Transaction

A local transaction is a transaction that accesses and updates data on only one database. Local transactions are significantly faster than distributed transactions because local transactions do not require communication between multiple databases, which means less logging and fewer network round trips are required to perform local transactions.
Use local transactions when your application does not have to access or update data on multiple networked databases.

The following occurs when the application requests a transaction:



1. The driver issues a commit request.

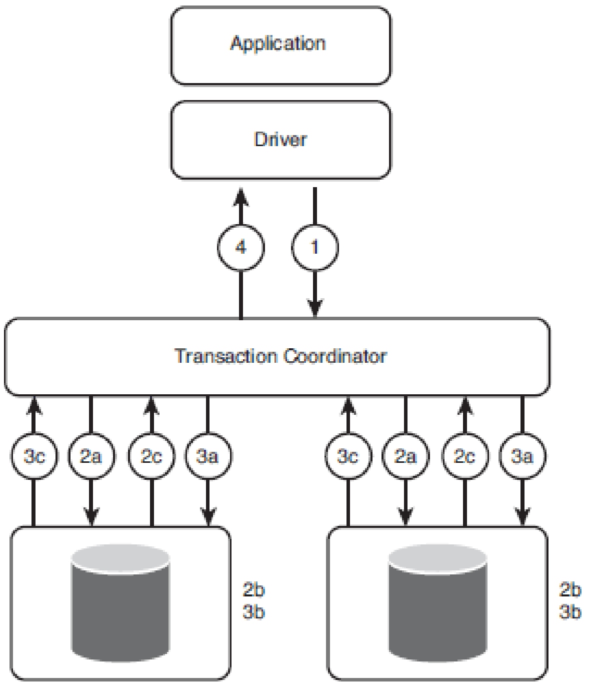
2. If the database can commit the transaction, it does, and writes an entry to its log. If it cannot, it rolls back the transaction.

3. The database replies with a status to the driver indicating if the commit succeeded or failed.

7.2 Distributed Transaction

In a distributed transaction, commands are distributed, that is, they are sent via two or more connections to one or more database servers. The commands sent to each DBMS server, or resource manager, constitute a branch of the transaction.
In order for a distributed transaction to be committed, all resource managers must vote to commit. If even one of the resource managers does not agree to commit, the transaction coordinator will roll back the results of command execution on all of the transaction branches.
Distributed transactions are substantially slower than local transactions because of the logging and network round trips needed to communicate between all the components involved in the distributed transaction.

The following occurs when the application requests a transaction:



1. The driver issues a commit request.

1A. The transaction coordinator sends a precommit request to all databases involved in the transaction.

2. The transaction coordinator sends a commit request command to all databases.

a. Each database executes the transaction up to the point where the database is asked to commit, and each writes recovery information to its logs.

b. Each database replies with a status message to the transaction coordinator indicating whether the transaction up to this point succeeded or failed.

3. The transaction coordinator waits until it has received a status message from each database. If the transaction coordinator received a status message from all databases indicating success, the following occurs

a. The transaction coordinator sends a commit message to all the databases.

b. Each database completes the commit operation and releases all the locks and resources held during the transaction.

c. Each database replies with a status to the transaction coordinator indicating whether the operation succeeded or failed.

4. The transaction coordinator completes the transaction when all acknowledgments have been received and replies with a status to the driver indicating if the commit succeeded or failed.