2017

Advanced Database Project Design Document

YI ZHANG, TAIKUN GUO

OCT. 23 2017

Content

1. Use Cases

In this distributed database system, users can begin a Read-Write or Read-only transaction and end it. During a transaction, a user can write on a variable or read the value from one variable. Besides, users can fail or recover a server, and he can also view the values of all variables on all server by dump operation. As shown in Figure 1.

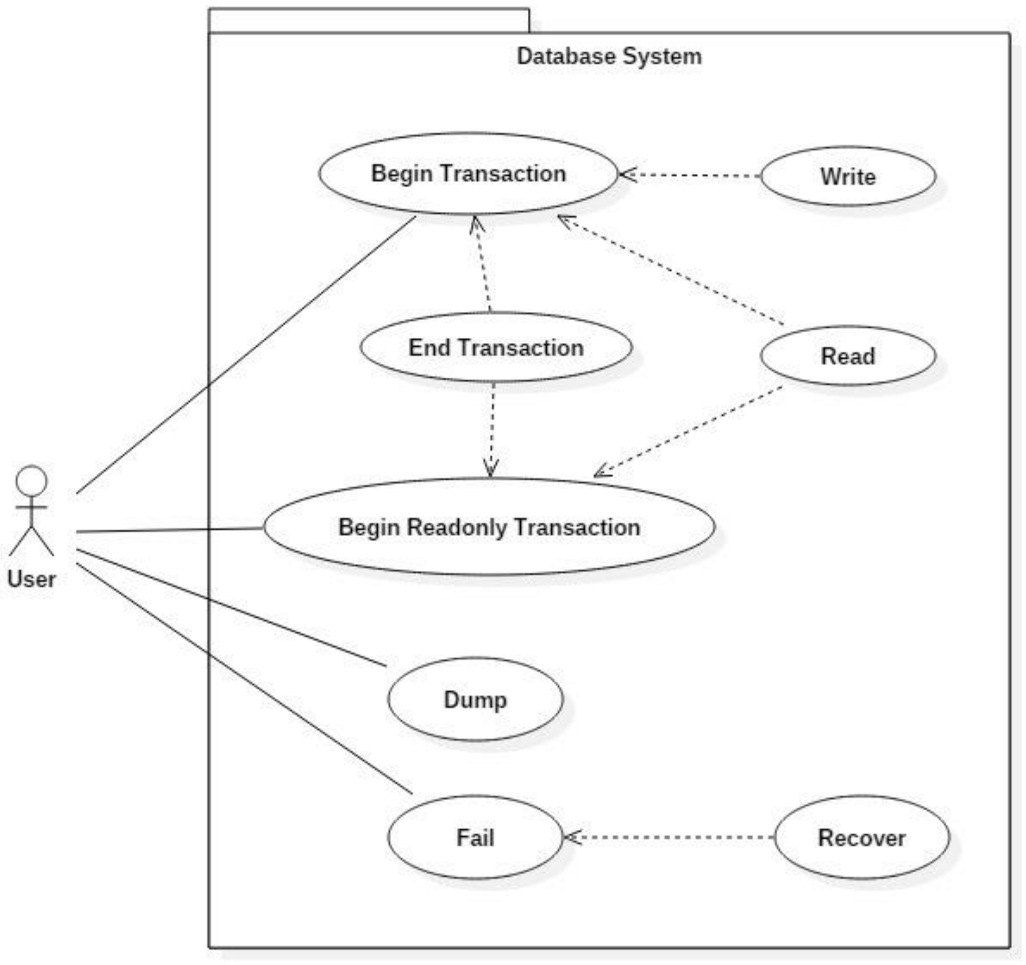

Figure 1 - Use Cases

2. Activities

2.1 Begin Transaction

When a user wants to begin a new transaction, he types in the console or from the file. The console thread will read the command and transfer to the transaction manager. If the transaction is a read-only transaction, then it will fetch the version data from all the servers, then with this version data, the transaction manager will create a new transaction, or without the version data if it is not read-only. After that, register this new transaction in the transaction list which is maintained by the transaction manager. As shown in Figure 2 below.

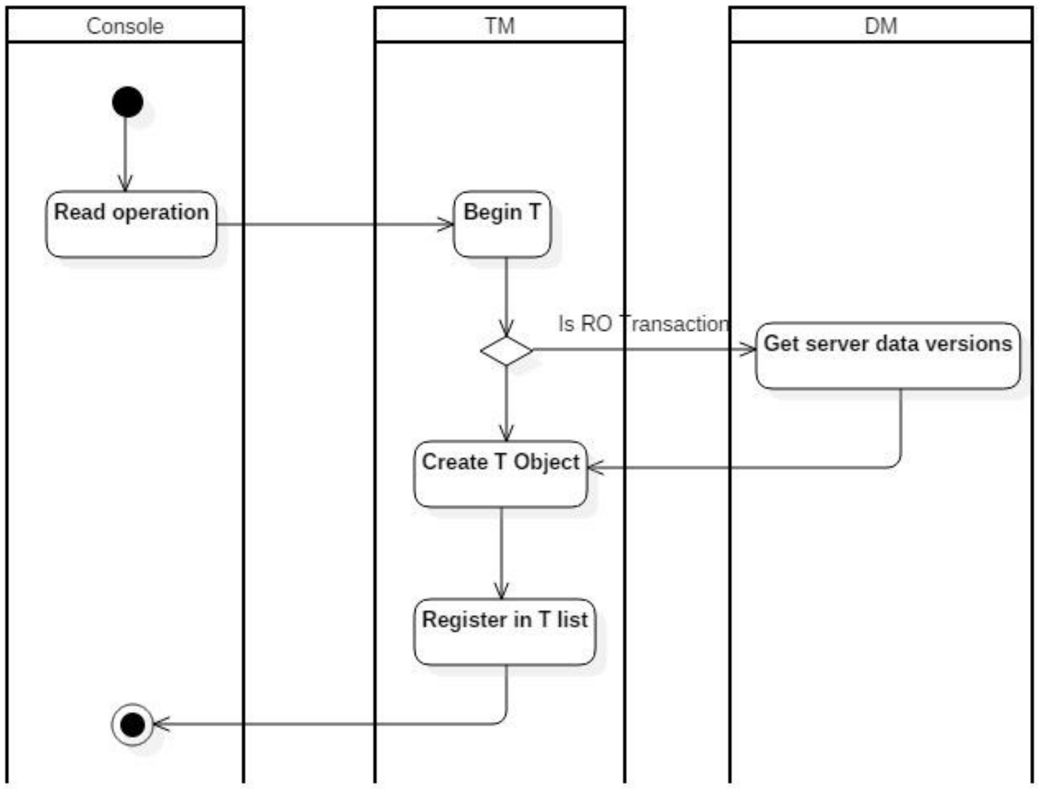

Figure 2 - Begin a transaction

2.2 End Transaction

When a transaction is required to be ended, if this transaction is read-only, then there is nothing left needed to be handled, and it will be simply removed from the list and recycle its resource. If not, the transaction manager will apply the changes made by this transaction to the database manager. When DM receive this update request, it will first check whether there is any server currently being accessing by a read-only transaction. If there is, then it will reserve an old version copy, and then update the data on the servers with increasing the version on the server. Then TM will remove the locks occupied by this transaction, and the details of removing locks will be discussed later. The activity flow is shown below.

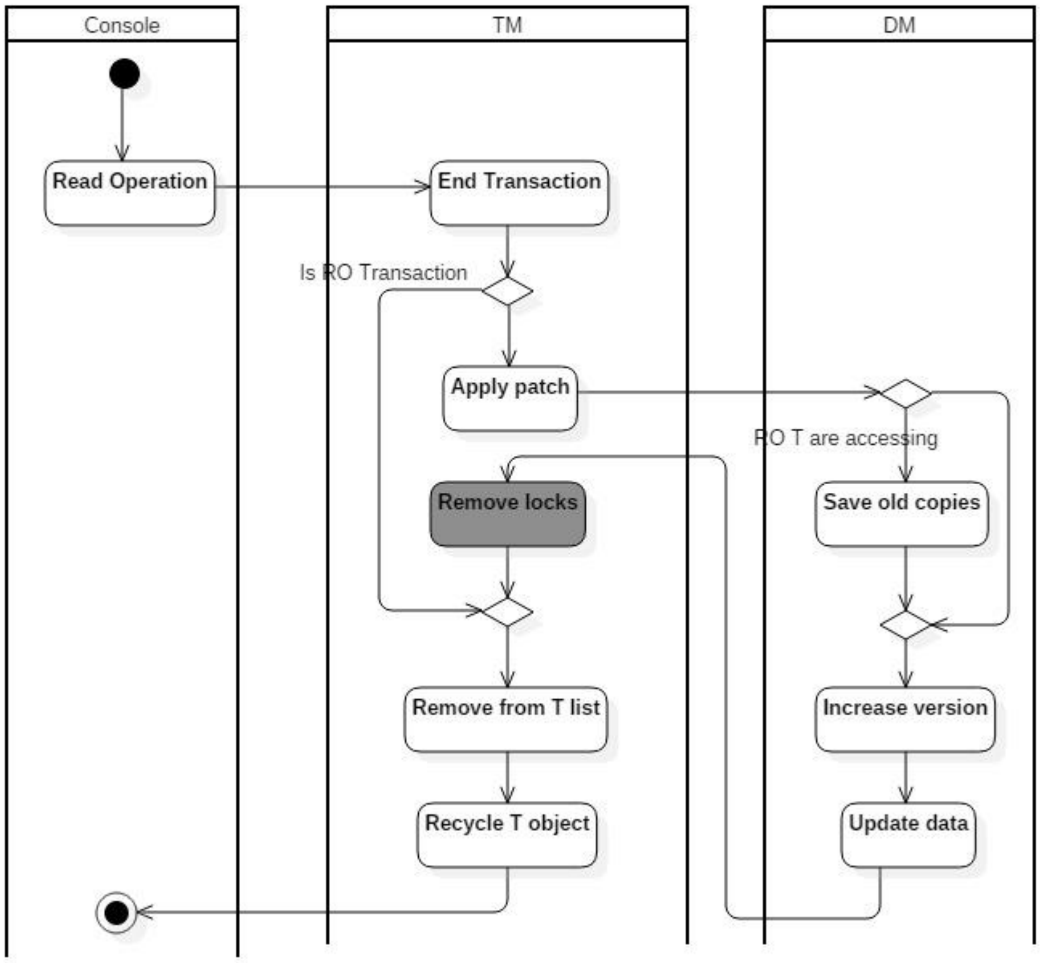

Figure 3 - End a transaction

2.2.1 Remove Locks

When TM needs to remove all the locks of a transaction, it will first check which variable it locks, then remove them from the lock table. After that, for those variables whose lock is just released, the TM will check whether there is another transaction waiting for it. If there is, then locks it and change the state of the transaction to Running, and continues the operation of that transaction.

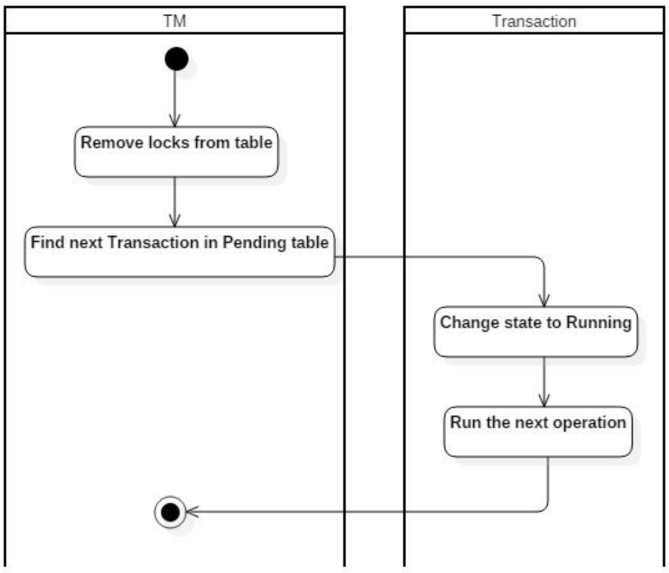

Figure 4 - Remove the locks of a transaction

2.3 Read Values

Reading a value is a complicated operation. When a transaction needs to read a variable's value, if the transaction is now blocking, then store this operation into the logs and return directly. If the transaction is a read-only transaction, it will ask the DM for the value of the specified version when the transaction starts, and store the value into its own cache (These results will be shown when the transaction ends.). If it's a read-write transaction, then it will ask the TM for a read lock. If getting the read lock successfully, then it gets the value from DM and store the results. If fails, it changes its own state to Blocking, and return directly.

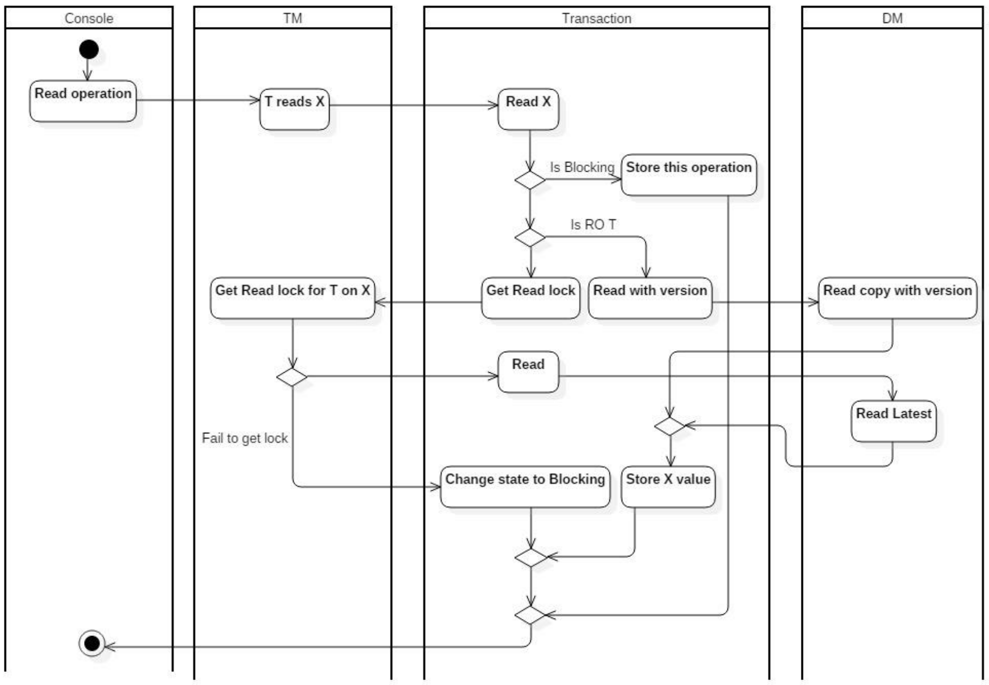

Figure 5 - Read a value in a transaction

2.4 Fail a Server

To test the database system, users can fail a server manually. When a server is failed, the DM will mark this server as failed or disabled, then it will inform TM to abort all those transactions that has read or written on this server.

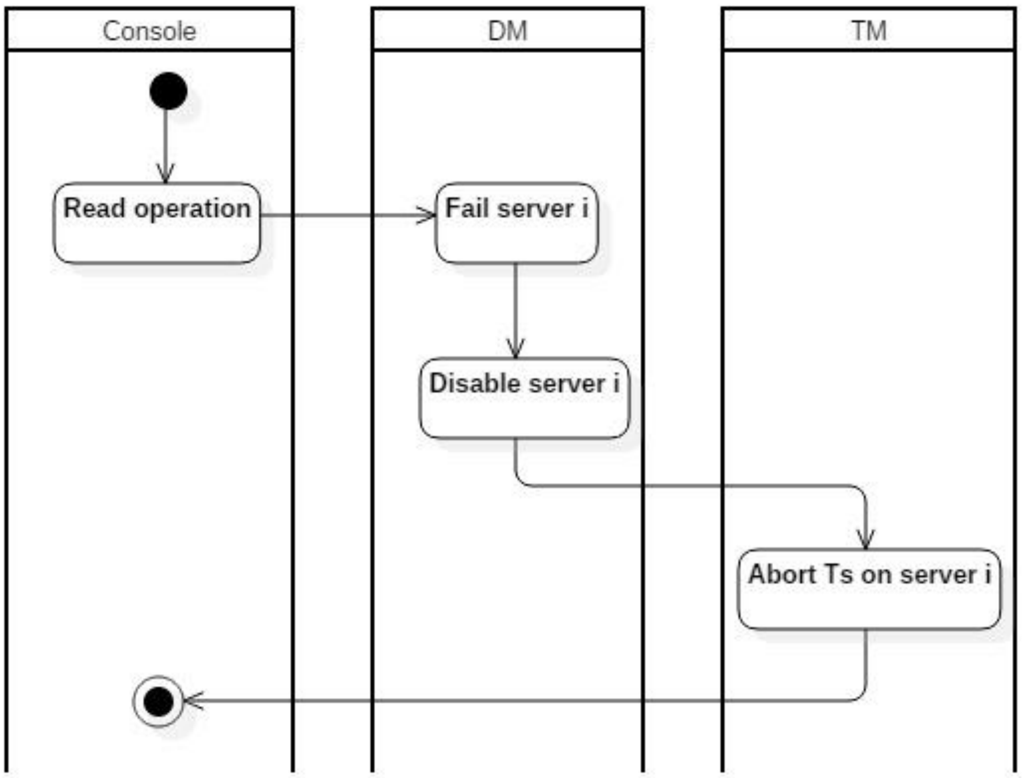

Figure 6 - Fail a server

3. State Changes

3.1 Transaction States

When a transaction is created, its state will be initialized as Running. When an operation in this transaction is successfully processed, it will keep Running until it ends. If a new operation comes, and it cannot get a lock from TM, then this transaction will become Blocking. And it will go back to Running when it gets its lock. If a deadlock cycle is detected or some servers fail, then this transaction will be Aborted. If this transaction can be normally ended and committed, then it will become Committed.

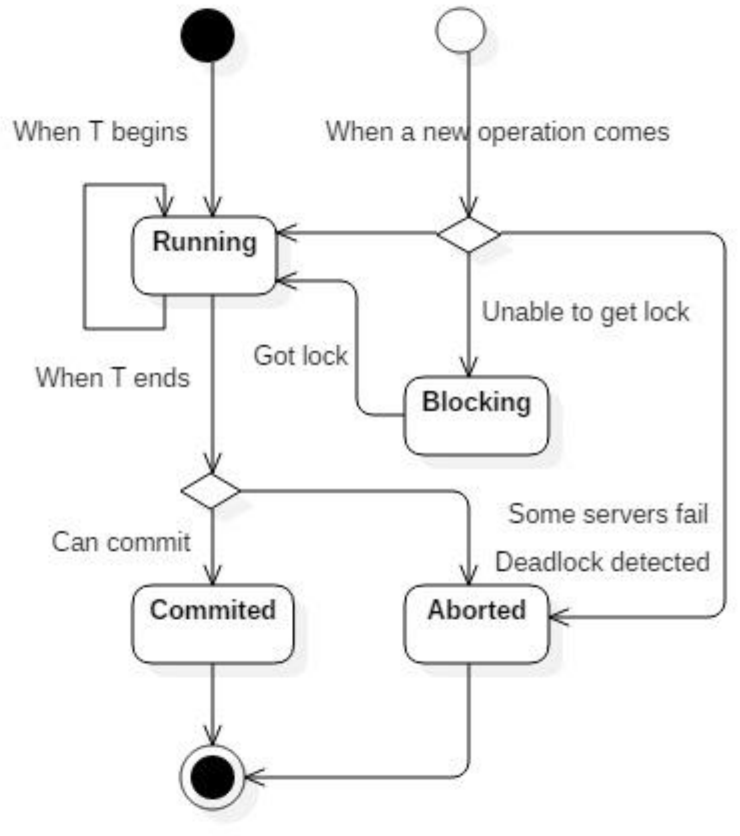

Figure 7 - Transaction States