2017

Advanced Database Project Design Document

YI ZHANG, TAIKUN GUO

NOV. 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.

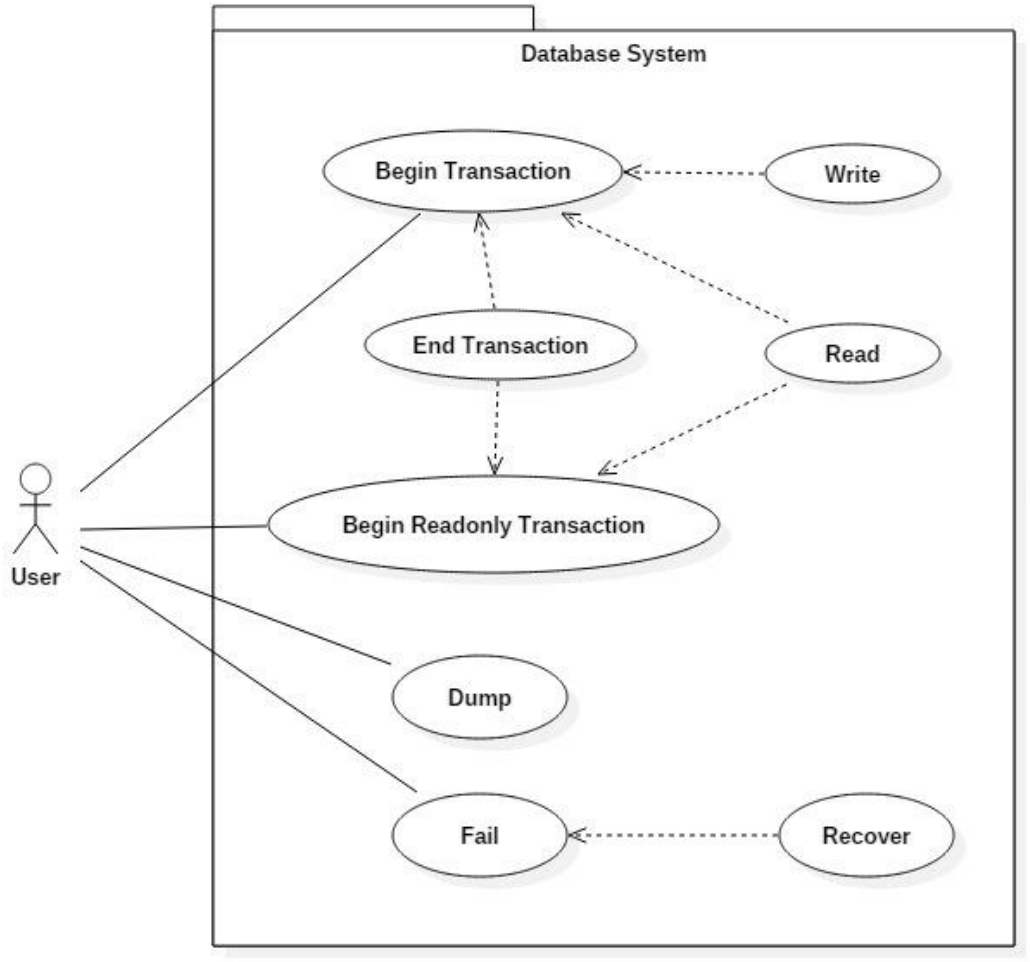

Figure 1 - Use Cases

2. Activities

2.1 Read Operation

To interact with the program, the user can type a command in the console. The program supports user to load a file or a command directly processed by this program. When the command is recognized as loading a file, the program will split the content in the file into each line and process them sequentially. When processing a command, first it will validate whether this is a legal operation. If it is, then parse the command and extract the operation and parameters to a wrapped object.

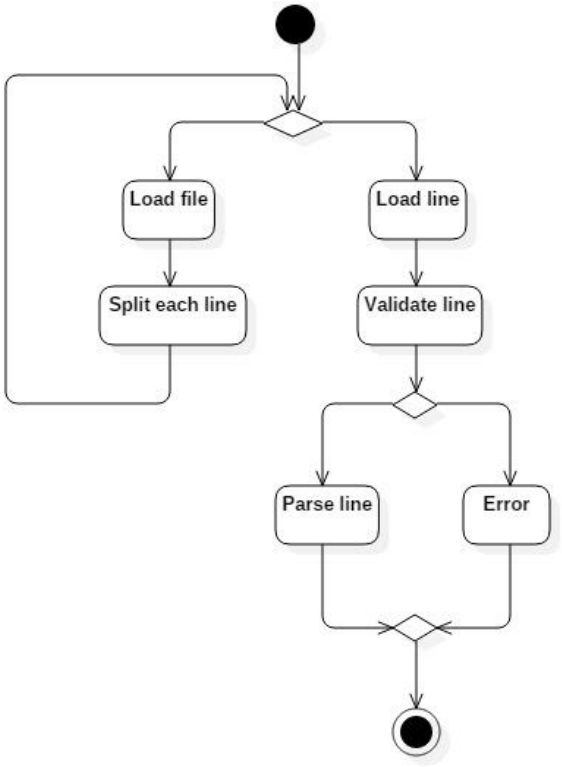

Figure 2 - Read an operation from the console.

2.2 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, which will be stored in the DM, then the transaction manager can create this 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 3 below.

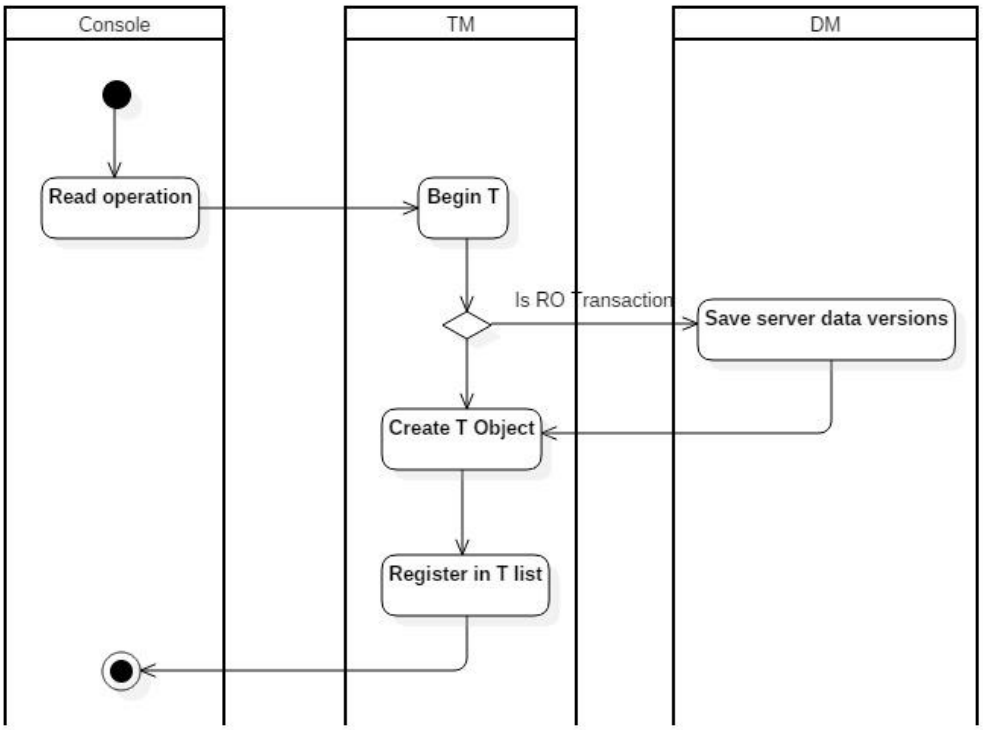

Figure 3 - Begin a transaction

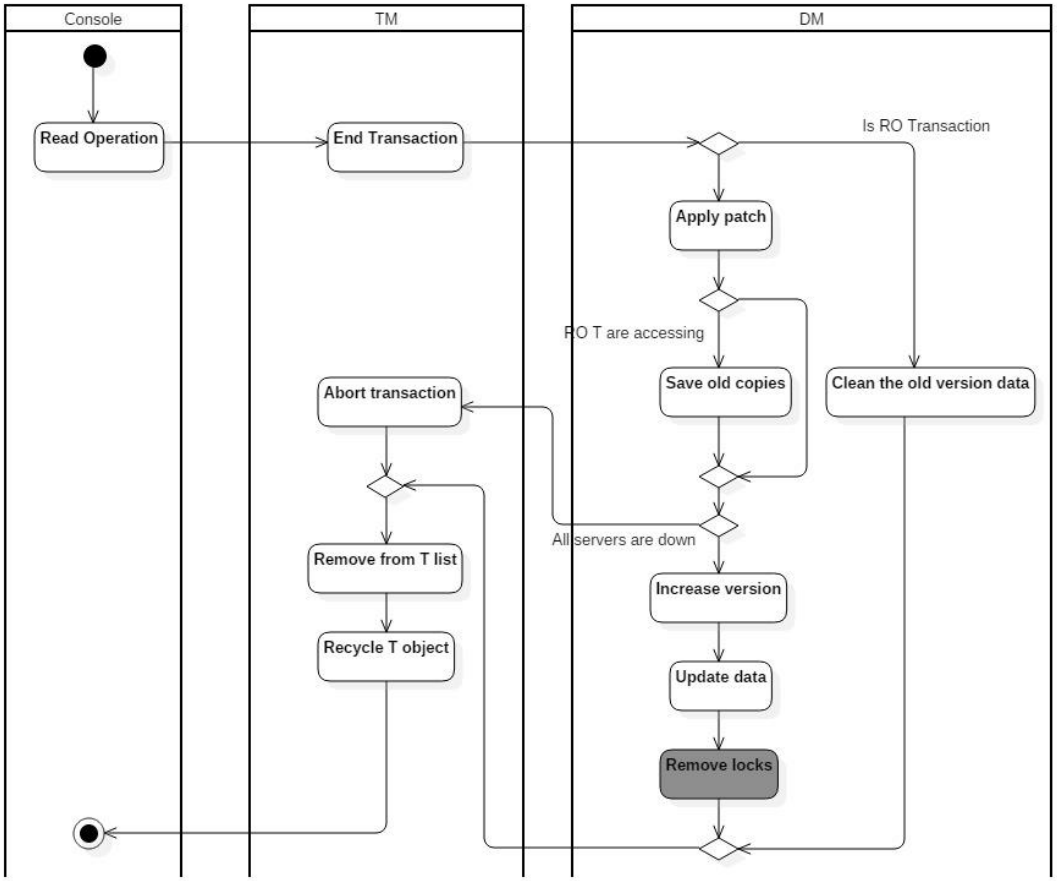

Figure 4 - End a transaction

2.3 End Transaction

When a transaction is required to be ended, if this transaction is read-only, then each server will be informed to remove the old version backup for this transaction, and then it will be removed from the list and recycle its resource. If not, the database manager will apply the changes made by this transaction, and 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. After that, DM will check whether there is server down, if there is, then abort this transaction because the update cannot be completed. Otherwise, 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 in Figure 4.

2.3.1 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.

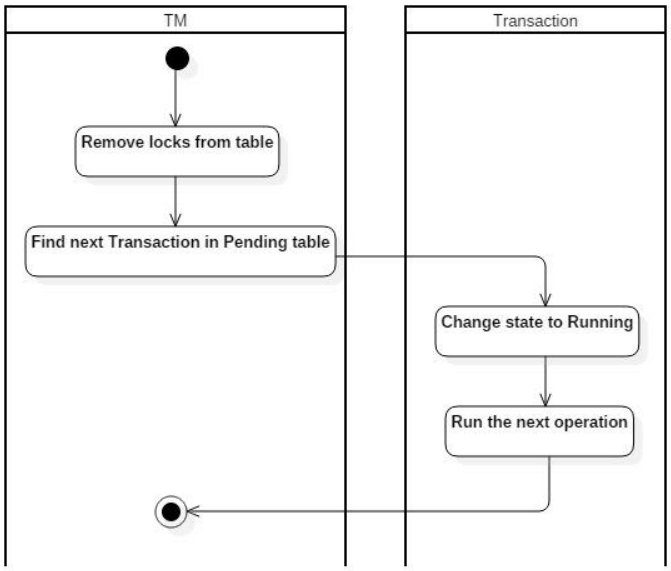

Figure 5 - Remove the locks of a transaction

2.4 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. Then the DM will inform TM to abort this transaction if there is no backup server available. 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 latest version value from DM and store the results. If fails, it changes its own state to Blocking, and return directly. Certainly, it will check whether there is a deadlock. This process is shown in Figure 6.

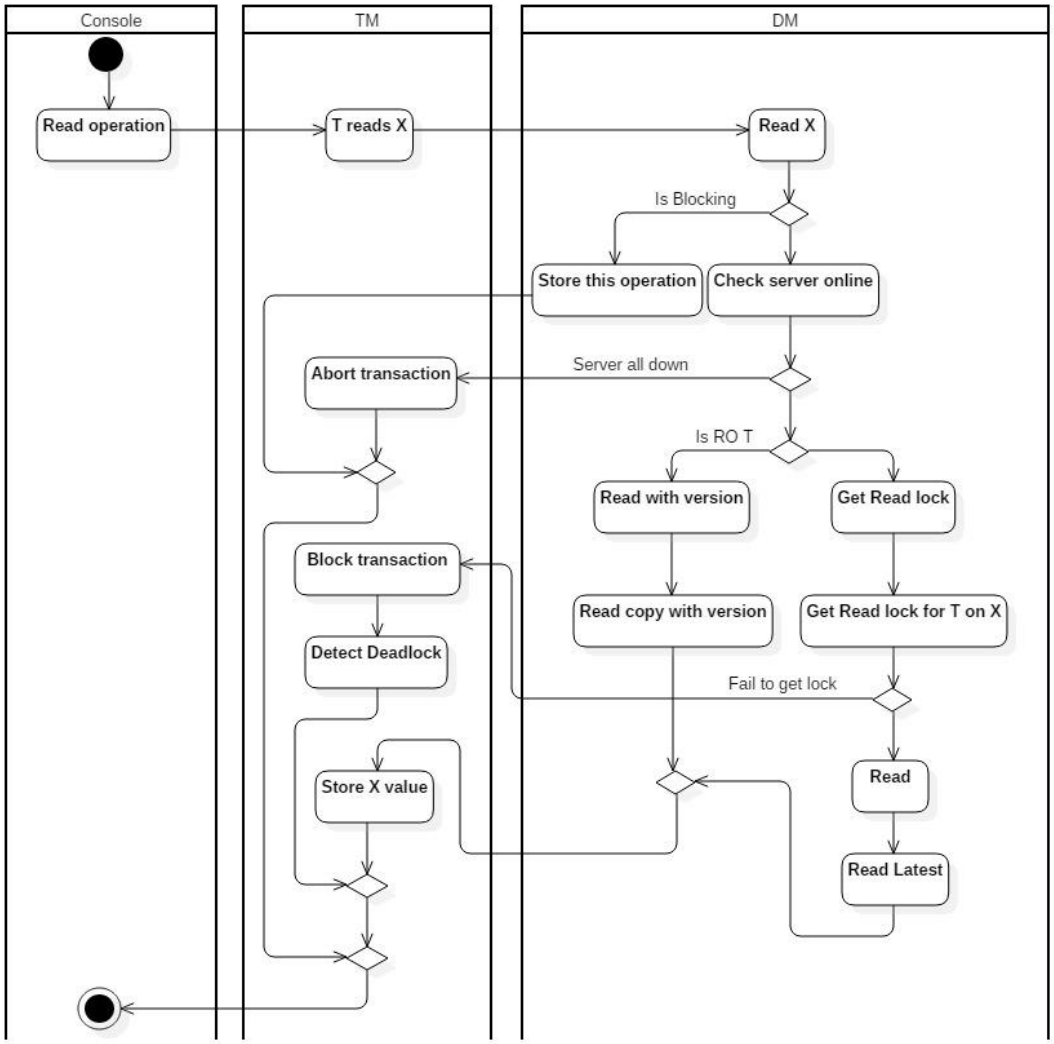

Figure 6 - Read a value in a transaction

2.5 Write Values

To write a value to a variable in a transaction, it must try to get an exclusive lock. If fail to get the lock, then this transaction will be added to the waiting list, and the transaction manager blocks this transaction until the lock of this variable is available. Especially, when the waiting lock list or lock list has been changed, the system will start to detect whether there is a dead lock. If this transaction gets the exclusive lock successfully, the transaction will write the value into its separated memory. And when this transaction is committed, its memory will be updated to all the servers.

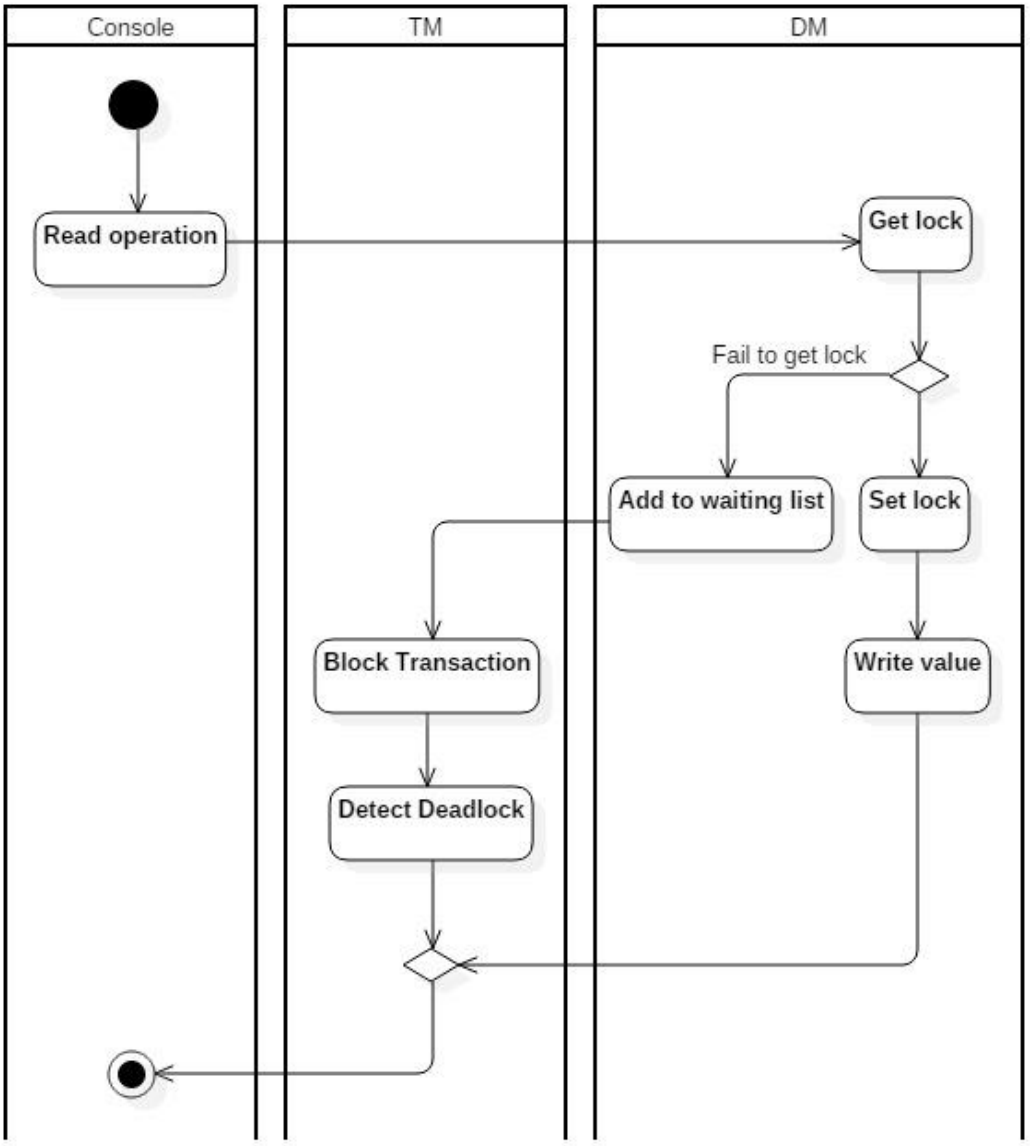

Figure 7 - Write a value in a transaction

2.6 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. The method to find such transactions is by detecting the read or write locks on this server.

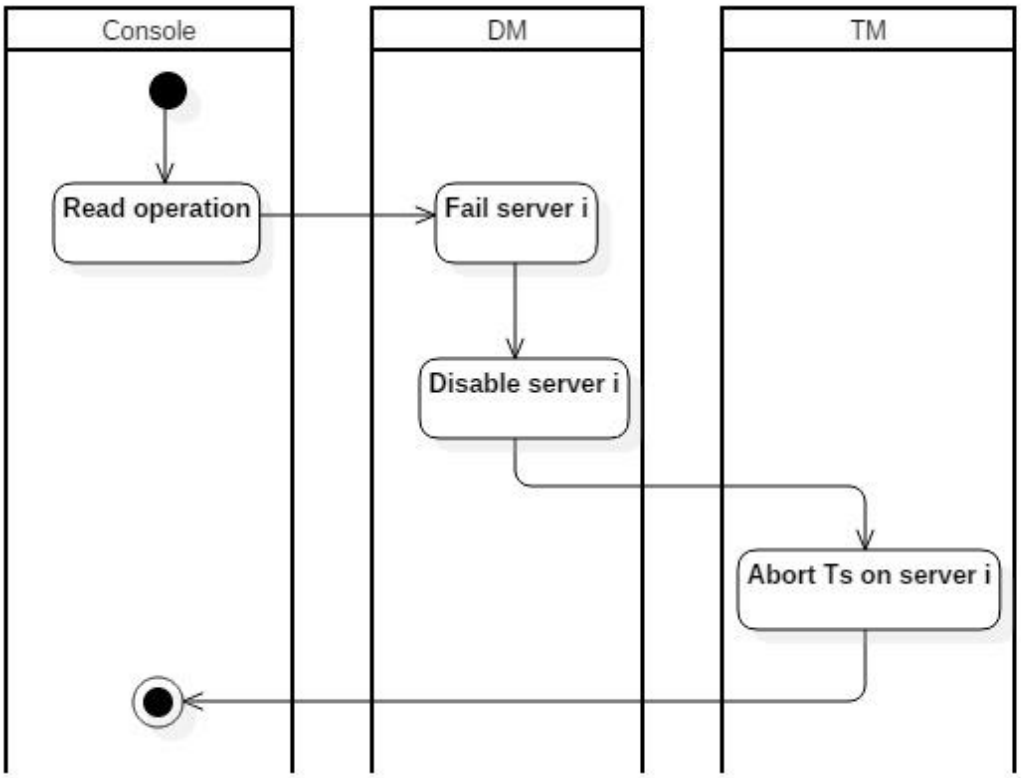

Figure 8 - Fail a server

2.7 Recover a server

To recover a server, the DM will first synchronize this server with other online servers. If finished successfully, then the status of this server will be set as available.

2.8 Detect the deadlock

To detect the deadlock, the program uses the DFS algorithm to detect whether there is a loop among the transactions. When detecting one, the transaction manager will abort the youngest transaction and continue all the other transactions. The program will detect the deadlock when the lock table or waiting lock table has been changed, especially when reading or writing variables.

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.

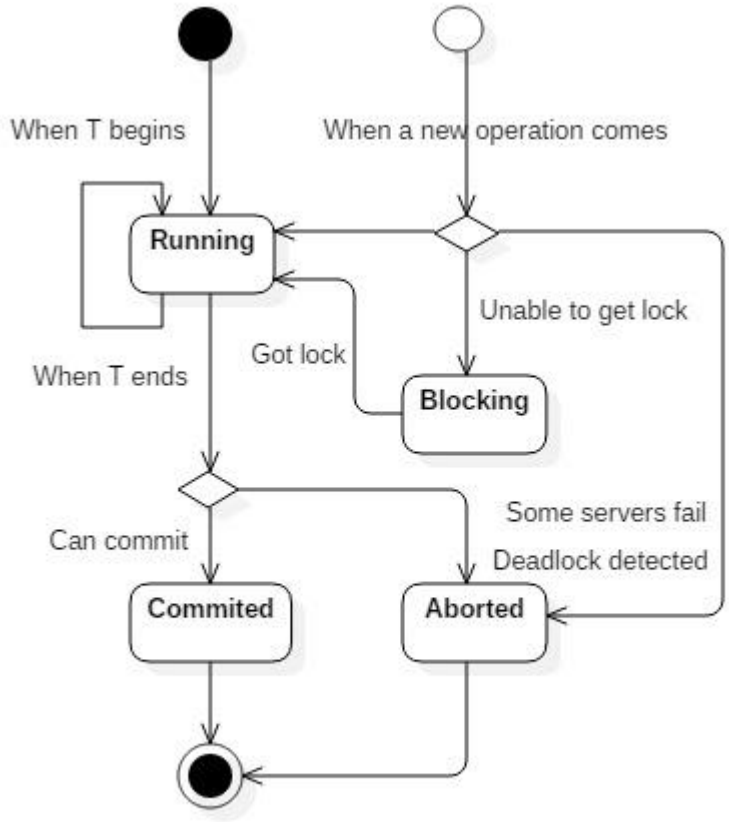

Figure 9 - Transaction States