A NEW INNOVATION TECHNIQUE OF STATE TRANSITION TESTING USED FOR DBT

Dhanamma Jagli[1], Dr.(Mrs.) Sunita Mahajan[2],Akanksha Gupta[3] and Sheetal Waghmar[4]

[1,3,4]V.E.S Institute of Technology,University of Mumbai,India.

[2] M.E.T-ICS, University of Mumbai,India.

Abstract

The process of exploitation the database to ensure the correctness of data manipulation, and a tendency to accomplished associations. The transaction is a fit of logic procedure units; the data modification from one state to some other state is represented with database transaction state diagram to substantiate uniformity of data inside the database. The data manipulation ought to separate groups of logic cells, and once it all finished, data consistency can be maintained, and once a piece of this unit fails, the whole transaction ought to be absolutely thought-about an error, all succeeding operations from the starting point should all fall back to the starting state. It has become a necessary to test database transaction states; a replacement technique of state transition testing is represented and designed test cases in this paper. The database State diagram direct testing by given the states, events, actions, and transitions that ought to be tested.

Keywords

State Transition Diagram,Data base Transaction, Software Testing, Transaction Testing,ACID Properties

1.INTRODUCTION

The Transaction a is activity, or ordering of actions, carried extinct by one-on-one somebody or practical application, which reads or change contents of database. The logical unit of activity Application program - serial of transactions with non-database process in betwixt Transforms database from one conformable state to some other Consistency may be desecrated during transaction. The Transaction comprises a unit of work performed within a database management system (or similar system) against a database, and treated in a coherent and reliable way independent of other transactions. For this purpose the proper testing should be done to describe all possible ways for executing the transaction. To test database transaction state diagram state transition testing is used. The other testing techniques that have been introduced do not assistance these aspects because they do not react to the antithetical behaviour of the purpose depending on the state.

The State transition systems differ however, from finite state automata in several ways:

- In a state transition system the set of states is not needfully finite, or even countable.
- In a state transition system the set of transitions is not necessarily finite, or even countable.
- The start state and a set of special final states.
- The State transition systems can be depicted as directed graphs.

DOI : 10.5121/ieij.2015.3201

2.CONCEPT OF STATE TRANSITION

The State-Transition draw point-blank our testing endeavour by distinguishing the states, events, actions, and transitions that should be proved. Put together, these define how a scheme act with the external world, the events it processes, and the legitimate and invalidated order of these events. A state-transition diagram is not the only manner to document instrumentality behaviour. They may be easier to comprehend, simply state-transition tables may be easier to exercise in a accomplished and systematised manner. The more often than not recommended level of testing using state-transition diagrams is to make over a set of test cases so much that every transitions are exercised at most once under test. In high-risk grouping, it may neediness to appoint even more test cases, forthcoming all paths if possible. The State-Transition diagrams are an fabulous instrument to natural process convinced types of system necessitate and to written document intrinsic system arrangement. The state transition diagram documents all events that ejaculate into and are equipped aside a system as well as the system's phenomenon.

2.1 The State transition testing in software testing

The State transition testing is utilized where few characteristics of the system can be delineated in a ‘finite state machine’. This merely effectuation that the system can be in a (finite) definite quantity of assorted states, and the transitions from one state to another are discovered by the construct of the ‘machine’. This is the framework on which the system and the tests are supported. Whatever system where we get a various output for the identical input, count on what has happened ahead, is a finite state system. One of the asset of the state transition proficiency is that the exemplary can be as elaborate or as abstract as it need to be. Where a portion of the system is more than all-important a greater depth of appendage can be modeled. The system is inferior crucial definite quantity fewer testing, the model can usage a individual state to stand for what would differently be a ordering of various states. In many cases, not only the contemporary input, merely too the past times of performance or events or inputs, determinant the outputs and how the test object will act. To elaborate the dependants on account state diagrams are exploited. They are the foundation for design the test (state transition testing).The system or test objective starts from an first state and can then come into assorted states. Events initiation state transitions where an event ordinarily is a function supplication. State transitions can regard activity. In any case the initial state, the another special state is the end-state.

- A state transition model has four basic parts:
- The states that the software may occupy.
- The transitions from one state to another.
- The events that cause a transition.
- The actions that result from a transition.

3.DATABASE TRANSACTION TESTING

A database transaction, by definition, must be atomic, consistent, isolated and durable. Database practitioners’ often refer to these properties of database transactions using the acronym ACID. Transactions provide an "all-or-nothing" proposition, stating that each work-unit performed in a database must either complete in its entirety or have no effect whatsoever. Further, the system must isolate each transaction from other transactions, results must conform to existing constraints in the database, and transactions that complete successfully must get written to durable storage. Transactions in a database environment have two main purposes:

To provide reliable units of work that allow correct recovery from failures and keep a database consistent even in cases of system failure, when execution stops (completely or partially) and many operations upon a database remain uncompleted, with unclear status.To provide isolation between programs accessing a database concurrently.

program's outcome is possibly erroneous.

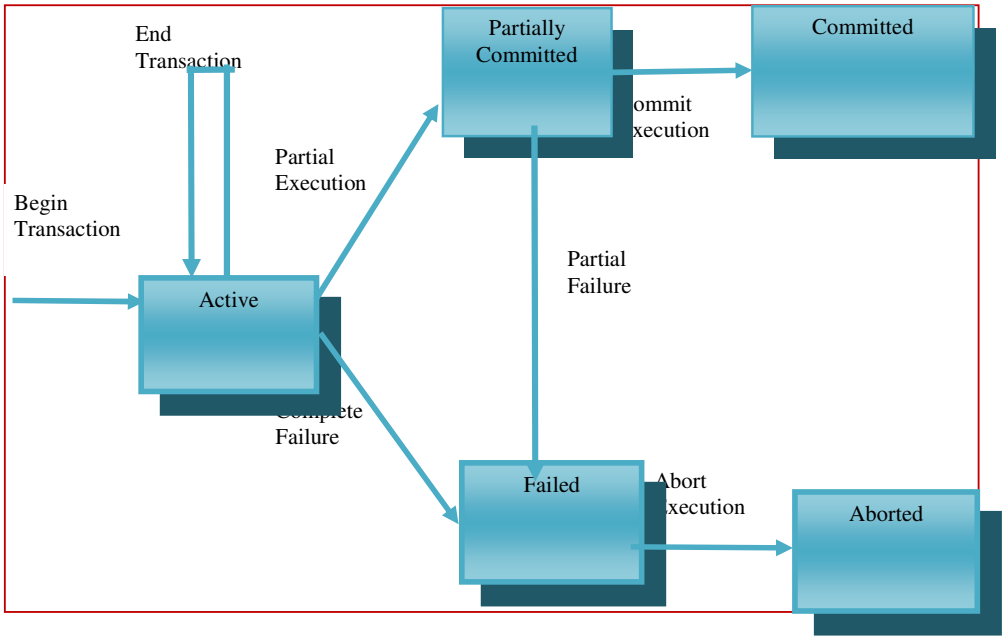


Figure 1. Transaction State Diagram

If this isolation is not provided the State-Transition draw is a superior tool to gaining control convinced kind of system necessitate and to representation internal system design. Transaction state diagram document the events that come into and are processed by a system as well as the system's responses. Test conditions can be derived from the state diagram in various ways. All states can be notable as a test status, as can all transitions. Nevertheless transaction state diagram springiness us information just about various states of transaction performance on which to arrangement several helpful tests and to inform the state transition technique. A transition does not demand to modification to a different state; it could edict in the identical state. In information, hard to input an invalidated input would be probable to bring forth an non accomplishment communication as the action,but simply the transition would be rearmost to the identical state the system was in earlier. To deriving test cases from the state transition exemplary is a black box conceptualization. The Measurement how much need to be tested (covered) is acquiring ambient to a white box orientation. Even so, state transition testing is specifically attentiveness as a black box proficiency. Nonetheless, deriving tests exclusive from the model may eliminate the antagonistic tests, where we could attempt to render invalid transitions. In order to comprehend the entire definite quantity of combinations of states and transitions, some valid and invalid, a state table can be utilized.

3.1The State-Transition Table

State-transition tables may be easier to use in a complete and systematic manner. The State-transition array consist of quaternary columns—Present State, Event, Action, and Next State and build the state-transition table as precede:

- All states of the system
- All events/triggers of the system
- Discovery the corresponding (Action, Next State) [if any]
- Document state, event, action, next state.

Table 1: Null State Transition Table

Current State	Event	Action	Next State
Null	Begin Transaction	Transaction started	Active
Null	End Transaction	-	Null
Null	Complete Failure	-	Null
Null	Partial Failure	-	Null
Null	Partial Execution	-	Null
Null	Commit Execution	-	Null
Null	Abort Execution	-	Null

Table 2: Active State Transition Table

Current State	Event	Action	Next State
Active	Begin Transaction	-	Active
Active	End Transaction	-	Active
Active	Complete Failure	Transaction Failed	Failed
Active	Partial Failure	-	Active
Active	Partial Execution	Transaction Executed	Partially Committed
Active	Commit Execution	-	Active
Active	Abort Execution	-	Active

Table 3: Partially committed State Transition Table

Current State	Event	Action	Next State
Partially Commited	Begin Transaction	-	Partially Commited
Partially Commited	End Transaction	-	Partially Committed
Partially Commited	Complete Failure	-	Partially Committed
Partially Committed	Partial Failure	Transaction failed	Failed

Partially Committed	Partial Execution	-	Partially Committed
Partially Committed	Commit Execution	Transaction committed	Committed
Partially Committed	Abort Execution	-	Partially Commit

Table 4: Failed Active State Transition Table

Current State	Event	Action	Next State
Failed	Begin Transaction	-	Failed
Failed	End Transaction	-	Failed
Failed	Complete Failure	-	Failed
Failed	Partial Failure	-	Failed
Failed	Partial Execution	-	Failed
Failed	Commit Execution	-	Failed
Failed	Abort Execution	Transaction aborted	Aborted

Table 5: Committed Active State Transition Table

Current State	Event	Action	Next State
Committed	Begin Transaction	-	Committed
Committed	End Transaction	-	Committed
Committed	Complete Failure	-	Committed
Committed	Partial Failure	-	Committed
Committed	Partial Execution	-	Committed
Committed	Commit Execution	-	Committed
Committed	Abort Execution	-	Committed

Table 6: Aborted State Transition Table

Current State	Event	Action	Next State
Aborted	Begin Transaction	-	Aborted
Aborted	End Transaction	-	Aborted
Aborted	Complete Failure	-	Aborted
Aborted	Partial Failure	-	Aborted
Aborted	Partial Execution	-	Aborted
Aborted	Commit Execution	-	Aborted
Aborted	Abort Execution	-	Aborted

The asset of a state-transition table is that it lists every achievable state-transition combinations, not rightful the valid ones. When testing fault finding, high-risk systems such as aeronautics or

medical devices, testing every state-transition couple may be needed, considering those that are not valid. Creating a state-transition table frequently excavate combinations that were not identified, authenticated, or dealt with in the need. It is highly advantageous to observe these defects ahead coding begins. Using a state-transition table can assist detect defects in implementation that modify invalid way from one state to some other. The disadvantage of such tables is that they get on very ample very rapidly as the amount of states and events increases. In addition, the tables are more often than not distributed; that is,to the highest degree of the cells are blank.

4. APPLICATION OF STATE TRANSITION TESTING

- **State transition testing**

The State transition testing is a black box testing technique and is utilized where several aspect of the system can be represented in what is called a "finite state machine". This analysable means that the system can be in a finite number of assorted states, and the transitions from one state to another are discovered by the convention of the "machine". This is the model connected the system and test cases are substantiated. Whatever system ,will acquire a various output signal for the identical input, investigation on what has come about in front, is a finite state system.

- **Consider history**

In numerous cases, not only the current input, just the history of performance, events , inputs, influences the outputs and how the test physical entity will behave. To exemplify the dependence on past state diagrams are exploited. They are the foundation for designing the test state transition testing. The scheme or test object starts from an first state and can then ejaculate into various states. Events trigger state transitions where an effect remarkably is a function supplication. State transitions can concern actions. In any case the first state, the another particular state is the end-state. Finite state machines, state diagrams, or state transition tables exemplary this activeness. The Test physical entity for state transition testing inbound state transition investigation the test object can be a completed system with several system states, as well as a class in an object-oriented system with assorted states. Whenever the history leads to take issue activeness, a state transition test must be applied**.**

- **Creating Test Cases**

1. Generate a set of test cases such that all states are visited to the lowest degree once under test. The set of three test cases shown beneath fitting this responsibility. In general this is a faint level of test coverage.
2. Generate a set of test cases such that all events are triggered at least once under test. Note that the test cases that cover each event can be the same as those that cover each state. Again, this is a weak level of coverage.
3. Generate a primed of test cases such that all paths are executed at least once under test. While this level is the most preferred because of its level of coverage, it may not be feasible.
4. If Testing of loops such as this can be important they may result in accumulating computational errors or resource loss (locks without corresponding releases, memory leaks, etc.).

If the state-transition diagram has loops, point in time the amount of achievable paths may be boundless. For illustration, presented a system with 2 states, A and B, where A transitions to B and B transitions to A. A a couple of the possible ways are:

A→B

A→B→A
A→B→A→B→A→B
A→B→A→B→A→B→A

And so on forever.

Generate a primed of test cases such that all transitions are exercised at least once low-level test. This level of testing supply a great even of extent without create big numbers of tests. This level is more often than not the one suggested.

- **Promote test cases**

For the state transition test, assorted levels of test intensiveness can be outlined. A nominal requirement is to scope every manageable states. Some other demand for the test is to evoke complete functions.

- **Test criteria**

The state transition test should slay each and every specific utility of a definite state at any rate formerly. The configuration betwixt the linguistic unit and the expected activeness of the test object can therefore be patterned. In order of magnitude to determine the required test cases, the finite state machine is changed into a transition tree, it consider convinced sequences of transitions. The whorled state transition diagram with possibly infinite sequences of states modification to a transition tree, which react to a typical amount of states without cycles. Successful doing this rendering, all states essential be reached and all transitions of the transition diagram must come about the transition tree is developed from a transition diagram in the pursuing manner:

1. The first or start state is the root of the tree.
2. For all accomplish able transition from the first state to a leading state in the state transition diagram, the transition tree acquire a branch from its root to a node, represent this next state.
3. The procedure for step 2 is continual for every leaf in the tree all freshly added node until one of the following two end-conditions is consummated:
    - The react state is already consider in the tree on the way from the root to the node. This end condition answer to one running of a cycle in the transition diagram.
    - The corresponding state is a final state, and therefore has no promote transitions to be considered.

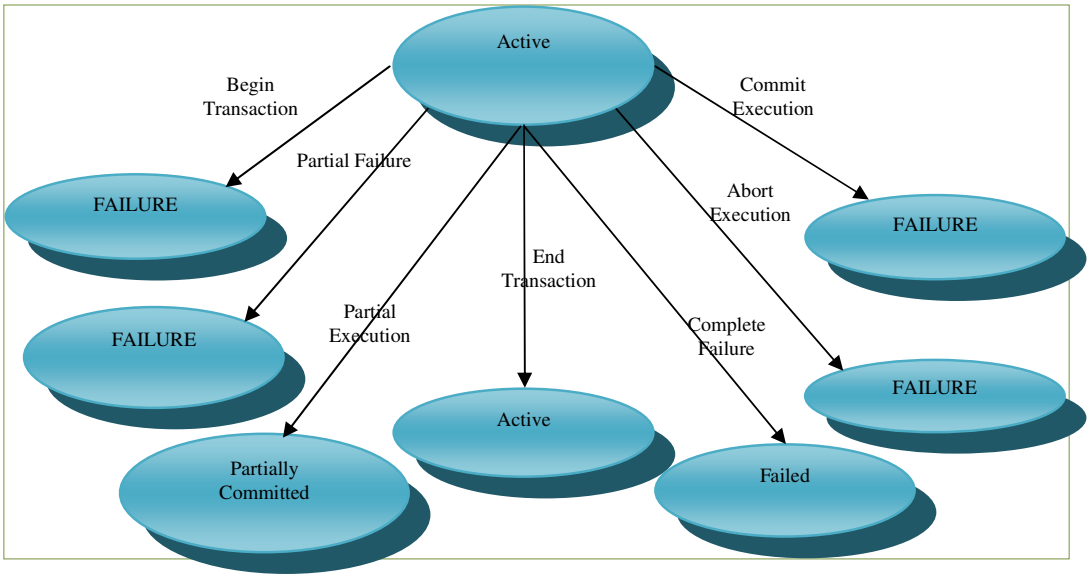
- **The State transition tree**

The Cardinal assorted routes can be produced from the root to all of the end nodes. Each of the paths represents a test case, i.e., a sequence of function calls. Therefore,every states is reached at

least at one time, and every manageable function is known as successful each state reported to the description of the state transition diagram.

- **Wrong usage of the functions**

In step-up to this, the response of the state machine intended for incorrect activity essentially patterned, which implementation that utility is called in states in which they are not expected to be called e.g., to withdraw the stack while in "full" state. This is a test of strength to substantiate how the test object enactment upon inaccurate use. Hence**,** it is established whether unanticipated transitions may come out**.** The test can be seen as an analogy to the test of unexpected input values. The transition tree diagram should be drawn-out awayconsider a subdivision for all function from every node. This mean spirited that from all sta tes, every function should be executed or at least attempted to be executed**.**



- **Test Cases**

  - The initial state of the test objective (component or
  - The inputs to the test object
  - The expected outcome or expected behaviour
  - The expected final state

Further, for for each one anticipated transition of the test case the leading aspects must be defined:

  - The state before the transition
  - The initiating event that triggers the transition
  - The expected reaction triggered by the transition
  - The next expected state
  - It is not forever effortless to determine the states of a attempt object.

Often, the state is not defined by a individual variable, merely is instead the result from a design of values of various variables. These variables may be profoundly concealed in the test object. Hence, the verification and valuation of each test case can be really high-priced.Test completion criteria is must to stop testing so where else.

- **Definition of the Test Completion Criteria**

Criteria for test intensiveness and for pass completion can also be defined for the state transition testing:

- ✓ All state has been reached at least once
- ✓ All transition has been executed at least once
- ✓ Every transition violating the specification has been checked
- ✓ Percentages can be defined using the proportion of actually executed test requirements to possible ones correspondent to the in the first place represented coverage measures.

- **Higher-level criteria**

For highly critical applications even more intensified state transition test completion criteria can be declared as follows:

- ✓ Every accumulation of transitions.
- ✓ Every transitions in whatever order of magnitude with every affirm-able states, consider multiple position in temporal arrangement.

But, achieving adequate coverage is frequently not manageable collectable to the big number of necessary test cases. Therefore, a limit to the number of consortium or sequences that essential be proven may then be intelligent.

5.CONCLUSION

The State transition testing is practical states are of the essence and where the practicality is influenced by the state of the test objective. In this paper a freshtechnique state transition testing is applied for data base transaction state diagram and designed test cases. This approach would be much more than serviceable for data base transaction to guarantee ACID properties.

REFERENCES

[1] The Text book, Software Testing Foundations: A Study Guide for the Certified Tester Exam", By Andreas Spillner, Tilo Linz, Hans Schaefer.

[2] The text book ",Database System Concepts",by

[3] Praveen Ranjan Srivastava and Km Baby, "Automated Software Testing Using Metahurestic Technique Based on An Ant Colony Optimization.", published in the 2010 International Symposium on Electronic System Design.

[4] Hideharu Kojima, Tomoyuki Ohta, Yoshiaki Kakuda, "State Transition Model for Test Sequence Generation of MANET Clustering Protocols", published in 27th International Conference on Distributed Computing Systems Workshops.

[5] Xiao Liu and Michael S. Hsiao ,“Constrained ATPG for Broadside Transition Testing ”,published in Proceedings of the 18th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT’03)
[6] Kwang –Ting ChengJing-Yang-Jou, “A single –state transition fau lt model for sequential machines”.