International Journal of Soft Computing 12 (5-6): 343-350, 2017
ISSN: 1816-9503


# Transactional Workflow Technique for Distributed Transaction Processing

Romani Farid Ibrahim
High Institute of Computer Science and Information, City of Culture and Science,
6-October City, Egypt

**Abstract:** Two Phase Commit Protocol (2PC) is an atomic and synchronous protocol and it is the common protocol uses to coordinate the commitment of transactions in distributed database systems. Most current database systems use page, level and locks technique to lock data items while transactions processing, other transactions can't access all records in the page while the update of a record in that page. If locks are released quickly from data items, other transactions can access these records. In this study, we propose a modification to the standard two phase commit protocol to release data items locks quickly after the preparation phase. Participants doesn't locks data items until the end of the transaction. We propose a two Phase Commit Protocol with Incomplete state (2PC-I) which avoids the system blocking problem and ensures semantic ACID properties. We propose a transactional workflow technique as an optimistic concurrency control technique that uses (2PC-I) and actionability rules to handle the disconnection in transactions processing and increase the success rate of transactions. We implemented a simulation prototype for the 2PC-I protocol and transactional workflow technique to test the applicability of the 2PC-I protocol and measure the success rate of transactions.

**Key words:** Transaction, concurrency control, two phase commit protocol, long-lived transaction, distributed database, mobile database, workflow, shadow paging, saga, caching, compensation

## INTRODUCTION

Most of business application that run in different organizations are based on the relational database model which uses the transaction concept and both their correctness are proved. Relational data model is based on the set theory rules and transaction concept is proposed to avoid race condition problems and it is based on the transaction properties (ACID). Serializability theory is used to prove the correctness of transactions schedules. Also, transaction moves the database from consistent state to a new consistent state so that users trust the information retrieved from information systems.

A Distributed Data Base (DDB) is a collection of multiple logically interrelated databases distributed over a computer network and a distributed Data Base Management System (DDBMS) as a software system that manages a distributed database while making the distribution transparent to the user. A Data Base Management Systems (DBMS) are classified according to the number of sites over which the database is distributed. A DBMS is centralized if the data is stored at a single computer site. A centralized DBMS can support multiple users but the DBMS and the database reside totally at a single computer site. A distributed DBMS (DDBMS) can have the actual database and DBMS software distributed over many sites, connected by a computer network. Homogeneous DDBMSs use the same DBMS Software at all the sites whereas heterogeneous DDBMSs can use different DBMS Software at each site (Elmasri and Navathe, 2011).

There are many types of transaction models we reviews the models related to our work only. Flat transaction (or simply transaction) is defined as a means by which an application programmer can package together a sequence of database operations so that the database can provide a number of guarantees, known as the ACID (Atomicity, Consistency, Isolation and Durability) (O'neil and O'nell, 2001). Nested transaction is a collection of related subtasks or subtransactions, each of which may also contain any number of subtransactions as a tree structure and only the leaf-level subtransactions are allowed to perform the database operations (Thomas and Carolyn, 1999).

We viewed a transaction as a program in execution in which each write-set satisfies the ACID properties (Hegazy *et al*., 2008) and the program that updates the database as a three folds module (phases): reading phase, editing phase and validation and write phase.

A distributed transaction is a database transaction in which two or more network hosts are involved. Usually, hosts provide transactional resources while the Transaction Manager (TM) is responsible for creating and managing a global transaction that encompasses all operations against such resources. Distributed transactions as any other transactions, must

satisfy ACID properties. A distributed transaction is composed of several subtransactions, each running on a different site (Chandra, 2017).

A distributed transaction is a transaction that includes one or more statements that individually or as a group, update data on two or more distinct nodes of a distributed database (Kumari and Chanderkant, 2012).

A Long Lived Transaction (LLT) is a transaction whose execution, even without interference from other transactions, takes a substantial amount of time, possibly on the order of hours or days. A long lived transaction has a long duration compared to the majority of other transactions either because it accesses many database objects, it has lengthy computations, it pauses for inputs from the users or a combination of these factors (Molina and Salem, 1987).

A compensating transaction is a transaction with the opposite effect of an already committed transaction. It is intended to undo the visible effects of a previously committed transaction, e.g., cancel car is the compensating transaction for rent car. A contingency transaction is invoked upon the occurrence of some failure condition and before commit of the transaction for which it is an alternative. It is intended to accomplish a similar goal as the original transaction as opposed to the compensating transaction which is intended to undo a committed (sub) transaction (Elmagarmid, 1992). A saga is a long-lived transaction that consists of a set of relatively independent subtransactions associated with them their compensating subtransactions. To execute a saga, the system must guarantee that either all of the subtransactions in a saga are complete or any partial execution is undone with their compensating subtransactions (Molina and Salem, 1987). A vital transaction is a transaction that must be executed successfully (i.e., it has to commit) for its parent transaction to commit. A non-vital transaction may abort without preventing the parent transaction from committing (Elmagarmid, 1992).

A mobile transactions is a transaction performed with at least one mobile host takes part in its execution (Gary and Pamos, 1997); also, it may be defined with perspective of its structure as a set of relatively independent (component) transactions which can interleave in any way with other mobile transactions.

Workflow is a collection of tasks organized to accomplish some business process (e.g., processing purchase orders over the phone, provisioning telephone service and processing insurance claims). A task can be performed by one or more software systems, one or a team of humans or a combination of these (Georgakopoulos *et al*., 1995).

Two Phase Commit (2PC) protocol is synchronous in nature and thus not well suited for long-lived transactions. Although, 2PC provides autonomy of a transaction, the required processing load is rather heavy. The transaction speed is always limited by the resource manager with the slowest response and the network traffic and latency is double that of a normal transaction (Wilson, 2003).

In this study, we propose a 2-Phase Commit protocol with Incomplete state (2PC-I) which avoids the system blocking problem and ensures semantic ACID properties. We proposed a transactional workflow technique as an optimistic concurrency control model that uses (2PC-I) and actionability rules to handle disconnection and increase the success rate of transactions.

**Literature review:** Salem *et al*. (1989), the researcher proposed a model called altruistic locking which allows LLTs to release their locks early based on transaction's data access pattern. Once LLT is determined that the data the locks protect will no longer be accessed, it issues release operations and allows other transactions to access this data concurrently.

Molina and Salem (1987), the researcher proposed the Sagas Model which handle LLT as a sequence of sub-transactions that can be interleaved in any way with other transactions. Each sub-transaction in the saga guarantees that the ACID properties on the database are preserved. Either all of the subtransactions in a saga are complete or any partial execution is undone with compensating subtransactions. Sagas relax the property of isolation by allowing a saga to reveal its partial results to other transactions before it is complete, consistency may be compromised.

Elmagarmid *et al*. (1990), the researchers proposed the flexible transaction model for multidatabase environment. It consists of a set compensable or non-compensable subtransactions and a setof execution dependencies among subtransactions. It relaxes isolation by using compensation. It relaxes global atomicity by allowing the transaction designer to specify acceptable state for termination of the flexible transactions.

Taft *et al*. (2014), NoSQL systems such as Cassandra and Amazon's Dynamo DB are able to scale in/out a DBMS cluster easily because they do not support full SQL transactions.

Holt (2011), mapped data in CouchDB is stored in a B+ tree index, effectively making it impossible to query nonindexed data. Using a relational database you can index your data to make your queries more efficient but you can also query against non-indexed data.

Holt (2011), the core of Paxos is a consenus algorithm called Syned. Since, consensus is unsolvable in asynchronus systems with failures, the Syned protocol while guranteeing always to be safe, ensures progress when the system is stable so that an accurate leader election is possible. In order to gurantee safety even during instability period, the Syned algorithm employs a 3-phase commit like protocol where unique balllots are used to prevent multiple leaders from committing possibly inconsistent value and to safely choose a possible decision value during the recovery phase (Chockler *et al*., 2003).

Wikipedia concurrency control for editing is optimistic, allowing editors concurrent access to web pages in which the first write is accepted and a user making a subsequent write is shown an 'edit conflict' screen and asked to resolve the conflicts (Coulouris *et al*., 1994).

## MATERIALS AND METHODS

In this study, we describe the important points we considered to to propose the 2PC-I protocol and the transactional workflow technique which are: transactions dependency, actionability rules and description of validation test.

**Transactions dependency:** We classified transactions based on their structures as simple transactions and compound transactions (Chandra, 2017). Simple transaction s a transaction that cannot be divided into subtransactions and all ACID properties are achieved. Compound transaction consists of two or more simple transactions (called subtransaction) and theses subtransactions may be nested, it can be ACID or non-ACID. Examples of compound transactions are nested transactions, sagas, long duration transactions (LLT), kangaroo transaction, etc.

We define transaction as a unit of work which each write-set satisfies the ACID properties. Unit of research means the structure of the transaction depends on the business logic (rules). The programmer constructs the transaction by collecting the suitable instructions according to his/her view of the business process. For example, if a programmer writes an application to calculate the employee salaries of an organization, based on their working hours or basic salary, total deductions (absence, tax, loans, etc.) and additions (bonuses, overtime, commission, etc.). The programmer can write the calculating salaries transaction by different ways as a flat transaction or as a compound transaction consists of group of subtransactions for calculating deductions and additions. If employee salaries data are distributed among different branches of an organization he/she writes it as a distributed transaction which can be flat or compound transaction.

We classified dependency among transaction according to the relation between them to independent, dependent and partially dependent (Chandra, 2017). The success (commitment) or failure (abortion) of an independent subtransactions doesn't depend on the success or failure of other subtransactions in the compound transaction. Success or failure of a dependent subtransaction depends on the success of the other subtransactions in the compound transaction, if any subtransaction fail, the entire transaction fails because all subtransactions are vital subtransactions. Partially dependent transaction includes some non-vital subtransactions in the compound transaction, transaction can commit without them. Also, we consider another two types of dependency based on the return values from transactions, sequence dependency and value dependency. Sequence dependency means a subtransaction doesn't return any value to the next subtransaction. For example, subtransactions that calculate the employee salaries doesn't return any value to each other because salary of an employee doesn't depend on the salaries of other employees. Value dependency means the next subtransaction depends on the return value from the current and or previous subtransactions and can't complete its work without it. example, calculating the salary of an employee is based the return values from deduction subtransaction addition subtransaction to the net salary subtransaction. From this dependency analysis, we notice that sequence dependency can be implemented as independent, dependent or partially dependent according to the business rules. Value dependency relation should be implemented as dependent or partially dependent. This analysis programmers in designing the structure of transactions transactions processing.

We classified applications according to the division their compound transaction to atomic compound applications which their compound transaction divisible and satisfies ACID properties and Transactional Workflow (TW) applications which their compound transaction is divisible and satisfies semantic ACID.

**Motivating example:** As an example of applications that can be applied as atomic transaction application or as a transactional workflow application, we are considering a big salespersons that can connect to the system through wired network using fixed hosts. Or through wireless network by mobile units. The salesperson performs a task

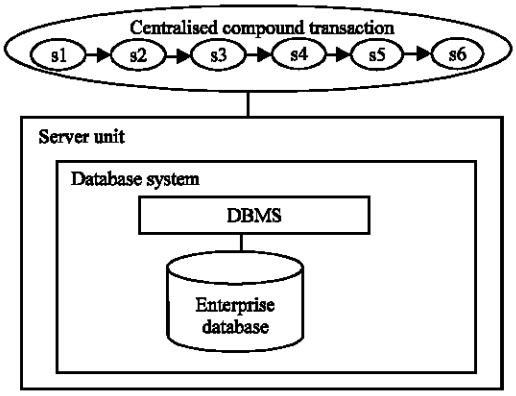


Fig. 1: Centralized compound transaction

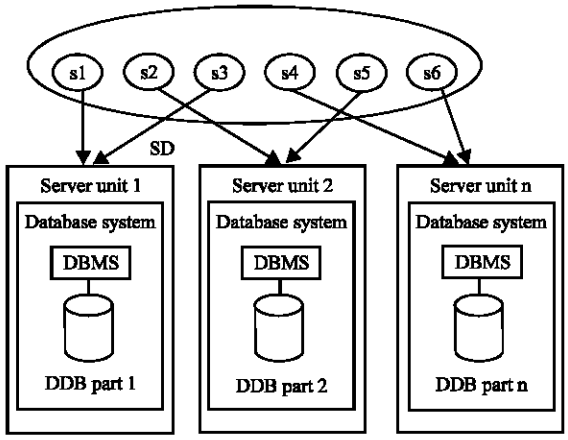


Fig. 2: Distribution compound transaction

that handles customer big order which consists of groups of sub-orders. The relations among suborders depends on the business logic rules.

**Case 1:** Compound transaction is a centralized transaction and the relation among its subtransactions is independent.

**Case 2:** Compound transaction is a centralized transaction and the relation among subtransactions is dependent partially dependent.

**Case 3:** Compound transaction is a distributed transaction and the relation among its subtransactions is independent.

**Case 4:** Compound transaction is a distributed transaction and the relation among subtransactions is dependent partially dependent.

In Case 3 and 4, we assume that the data are distributed among database partitions on different sites. Figure 1 shows a centralized compound transaction that consists of six subtransactions which access a centralized database that is stored on only one machine (the server). The relationship among subtransactions can be independent, dependent or partially dependent.

Figure 2 shows a distributed compound transaction that consists of six s ubtransactions. It accesses a distributed database system which can be homogenous or heterogeneous databases. The figure shows that subtransactions s1 and s3 is executed on the server 1, subtransactions s2 and s5 is executed on the server 2, subtransactions s4 and s6 is executed on the server 3. The relationship among subtransactions can be independent, dependent or partially dependent.

**Actionability rules:** In this study, we summarize the actionability concept and rules that is used in the M-Shadow Model (Ibrahim, 2017) to describe how a transaction behaves if a value-change is occurred on one or more of its attributes during its processing time and by other transactions. Other than key attributes (K) actionability classifies the data items used by a transaction into five types: change-accept, aware, reject, passing and location-time attributes.

**Change-Accept (A):** Any attribute retrieved during the read phase to complete and explain the meaning of the transaction. If it is potentially changed (by another transaction) while the transaction is processing, it does not have any effect on the transaction behavior.

**Change-Reject (R):** This type of attributes is subject of periodical changes (e.g., currency values, tax rates, etc.). the value of such attribute remains constant for long period. But once it is changed during the transaction life time (by another transaction), it affects severely the transaction behavior.

**Change-aware (W):** This type of attributes is subject to change more frequently by different concurrent transactions. A modification on the value of this type of attributes may be accepted if the new value still in the acceptance range. Otherwise, the transaction aborts.

**Change-Passing (P):** This type of attributes is not basically part of the transaction data but the result of the transaction processing is passed to this type of attributes. For example, in an insurance company (or many other applications) all different departments are related through the financial department so that all insurance transactions in all departments should pass their financial values to the

financial attributes. Usually this subtransaction is succeeded because it only increases the financial attributes by the new amounts and the previous change and the current values of this type of attributes doesn't effect on the transaction data or behavior. But if the subtransaction that changes their values is failed for any reason, it causes the main transaction to fail.

**Location-Time (L):** This type of attributes is for handling Location dependent transaction processing.

**Description of validation test:** The validation test for subtransactions is performed as an optimistic concurrency control technique by comparing the original values of some data items with its current values on the primary server which succeeds in three cases:

- No change which means that the original values are equal to the current values on the primary server
- Constrained change which means that some Change-aware attributes has been changed by other transactions during disconnection (working offline) time but still these changes within the integrity constraint acceptance range
- Insignificant change which means that some Change-accept attributes has been changed by other transactions during disconnection (working offline) time or during the execution of the transaction but these change-accept data items does not effect on the current transaction

The validation test fails in the following two cases:

- Significant change In which we detect that some change-reject data items have been changed during the transaction processing and/or disconnections
- Out-of-constraints change, in which we detect that one or more change-aware data items have been updated in such a way that the global changes put the stored values out of the acceptance ranges

**The 2-Phase Commit Protocol with Incomplete state (2PC-I):** The two Phase Commit Protocol (2PC) is commonlyused used to coordinate the commitment of transactions in distributed database systems. It is an atomic commitment protocol, it has only two states commit or abort transaction. The coordinator issues commit state message if all participations vote ready to commit and issues abort state message if at least one participation decides to abort its subtransaction. It locks data items while transaction 5 processing, until the final decision (commit or abort) of coordinator is received.

Page-level-locks are currently the most frequently used locking method for multiuser DBMSs (Coronel and Steven, 2016). If locks is released quickly from data items, other transactions can access these records.

We propose a modification to the standard two phase commit protocol to release data items locks quickly after the preparation phase. Participants doesn't locks data items until the end of the transaction.

We suggest to add a new state for the subtransactions at participant database sites which is Incomplete state (I). Also, data items (records) include an attribute contains the state of the last transaction (last_trans_state) that accessed the data item.

If a participant votes ready to commit, it put the state of subtransaction and last_trans_state attribute to incomplete state and locks are released.

When it receives the final decision (commit or abort) of the coordinator, it changes the transaction state and data item state to committed or aborted and undo the effects of its subtransaction on the database items (restore before image).

If the coordinator fails before sending its message when it recovers, it checks the log file. If it found the initialization of a transaction and doesn't find a decision (commit or abort) or end of the transaction, it sends a state request message to all participant sites.

When the coordinator receives the state messages from all participants, it takes a new decision based on the returned messages. If all participants send ready to commit, the coordinator commits the transaction otherwise, it aborts the transaction and sends its final decision to the participants.

If the coordinator fails after sending its decision to all participants, then when it recovers, it checks the log file. If it found the end of the transaction or commit or abort record which means transaction is completed then, it will do nothing but if it didn't find the end of the transaction record in the log file, it sends a request state message to all participant to checks the correctness of the subtransactions states on all participated sites.

If a participant still in the incomplete state, the coordinator send its final decision to that participant, then the coordinator write end of the transaction in the log file.

If a participant fails before sending its vote message to the coordinator, the coordinator waits for the predefined time period and then send abort message to all active participants.

If a participant fails after sending its vote message to the coordinator when the participant is recovered, it send a request state message to the coordinator to know the final decision of the transaction and apply it. If a new transaction requests to update data items from a

participant (calling participant) which its current transaction state is incomplete and its coordinator still fails, the participant checks the predefined timeout period if it passed, it sends a request state message (call) to all participants.

If at least one participant has the final decision of the coordination, it acts as a coordinator and sends the decision to the other participant and end transaction.

If no participant has the final decision of the coordinator and at least one participant is decided to abort its subtransaction, this participant acts as a coordinator and sends the abort decision to all other participants. All participants undo the effects of their subtransactions on the database items.

If no participant has the final decision of the coordinator and the state of all current subtransactions at all participants sites is incomplete state, then the calling participant acts as a coordinator for the current transaction and send a commit state message to all other participants and end the transaction. Then it allows the new transaction to access the requested data items.

If the coordinator and one or more participant fails, the calling participant acts as a coordinator for the current transaction and send an abort state message to all other active participants to avoid system blocking. When the coordinator is available, it send a request state message to all participants. If it found that participants are aborted the transactional workflow and the coordinator is committed it, it write a correction record in the log file, to indicate the failure of the transactional workflow task and send a message to the user so, he/she can execute it again.

If a new transaction request to read data item which hold incomplete state and the coordinator and other participants are active, this participant transfers the request to another available replica. We assume using the lazy replication protocol to update replicas of the databases. Update of replicas is only allowed for committed transactions.

**Transactional workflow technique:** We assume wired distributed database system and all sites are available while the transaction processing. Mobile units can delegates its transaction data to a fixed host agent using a model similar to M-Shadow Agent Model (Ibrahim, 2017).

The transactional workflow task follows the optimistic concurrency control model and it consists of three phase: read phase, edit phase and validation and write phase. All subtransactions of transactional workflow task are vital subtransactions.

The read phase and the edit phase are performed without locking, only the validation phase is under locking until the end of the preparation phase.

Each transactional workflow has an transaction-id which referencing its issue site and each subtraction has a sub-id. The transactional workflow coordinator sends transactional workflow id and sub-ids to all participants in the transactional workflow, to be used in the recovery case. Participants add their local subtransaction-id to its subtransactions.

Fixed hosts or mobile units retrieve sales orders data (transactional workflow data) we call it original data, from the their primary servers or from the nearest replicas and the user edit it (edited data).

The transactional workflow coardinator starts the validation and write phase by sending its subtransaction data (original and edited) to their primary servers. It uses 2PC-I to decide the final decision of commitment or rollbacking of the transactional workflow.

The primary server compares the current data with the orignal data. If the original data is not changed or the change-aware attributes are changed but these changes within the integrity constraint acceptance range, it accepts the edited data as a new current values of the data items and the validation test is considered success otherwise, it is considered fail.

If a subtransaction validation is succeeded, the participant sends ready to commit message to the transactional workflow coordinator, set the subtransaction state and the last-transaction-state attribute to incomplete state and releases locks of its data items.

When participants receive the final decision of the transactional workflow coordinator, the participants change their subtransaction state and the last-transaction-state attribute to committed. or aborted and undo the effects of its subtransaction on the database items (restore before image).

**Advantages and limitations of the transactional workflow technique:** The advantages of using the transactional workflow technique are:

- Only local locks are used for subtransactions on different participant sites and for short time at the validation phase and to the end of the preparation phase of the standard two phase commit protocol
- Releasing looks quickly from data items
- Reserve database semantic consistency
- Transactions complete its processing, even if case of failure of its coordinator
- If the network is disconnected while the transaction processing, the transaction not fail
- No need for using compensation
- It can be used for homogenous or heterogeneous distributed database

- It can be used for applications that moved to the cloud environment and need to use distributed short transactions, regular database size and semantic consistency

The limitations of the transactional workflow technique are: it is designed for commercial applications that have a few shared data items among transactions and the validation test is not suitable for some real-time applications. Also, it is not designed for handling huge or big data.

## RESULTS AND DISCUSSION

**Implementation and performance evaluation:** The standard two Phase Commit Protocol (2PC) aborts transaction if disconnection is happened while any time of the transaction is processing period. To evaluate the applicability of 2PC-I protocol and the success probability of transactions we used three database systems: Microsoft Access 2010, SQL Server 2008 and Oracle 10 to create the homogenous and heterogeneous DDBS environments. Application program is written by VB.net 2010. We simulated the coordinator by the application program. Participants are simulated by stored procedures at each database system. Log file and transaction state are simulated by tables include trans_id, subtrans_id as primary key and last_trans_state attribute is added to data tables.

State message is simulated by a query reads the state from the transaction state table. Commit state is simulated by changing the transaction state and last_trans_state attribute to committed and appending a record in the log_file table to indicate the commit event. Abortion state is simulated by changing the transaction state and last_trans_state attribute to aborted, undo the effects of its subtransaction on the database items (restore before image) and appending a record in the log_file table to indicate the abort event.

Validation test is implemented according to the actionability rules. The primary server compares the current data with the orignal data. If the original data is not changed or the change-aware attributes are changed but these changes within the integrity constraint acceptance range, it accepts the edited data as a new current values of the data items.

If the coordinator is failed and a new transaction request to access a record which last_trans_state attribute is incomplete a trigger is activated that checks the active participant state and decides whether to commit or abort the transaction. We measured the success rate of 2PC and 2PC-I protocols in the short disconnection case

Table 1: Comparison between success rates of 2PC and 2PC-1

| Disconnection rate (%) | 2PC-1 | | 2PC | |
|---|---|---|---|---|
| | Succeeded | Failed | Succeeded | Failed |
| 5 | 99 | 1 | 95 | 5 |
| 10 | 97 | 3 | 90 | 10 |
| 20 | 95 | 5 | 80 | 20 |
| 30 | 93 | 7 | 70 | 30 |
| 50 | 91 | 9 | 50 | 50 |

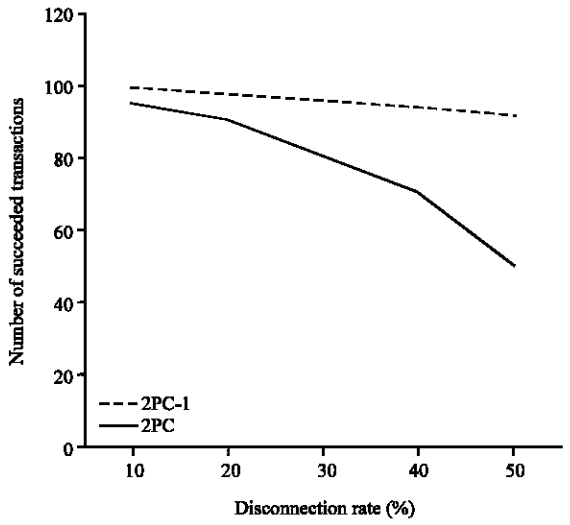


Fig. 3: Comparison between success rates of 2PC and 2PC-1

(Table 1). 2PC-I protocol success rate is higher than the standard 2PC protocol because it is implemented as an optimistic concurrency control technique. While 2PC is an atomic technique. Transactions fails in 2PC-I, in case of failure of participant while its processing in the validation test phase or in case of a new transaction requests to update data items from a participant which its current transaction state is incomplete and the coordinator and one or more participant fails. Figure 3 shows a graphical representation for the result.

## CONCLUSION

In this research, we define transaction as a unit of work which each write-set satisfies the ACID properties. We propose a two Phase Commit Protocol with Incomplete state (2PC-I) which avoids the system blocking problem and ensures semantic ACID properties. We propose a transactional workflow technique as an optimistic concurrency control techniqu that uses (2PC-I) and actionability rules to handle the disconnection in transactions processing and increase the success rate of transactions. We implemented a simulation prototype for the 2PC-I protocol and transactional workflow technique

to test the applicability of the 2PC-I protocol and measure the success rate of transactions. 2PC-I is designed to handle transaction processing of regular database size and normal business environment but when database size becomes very large, the system will be very slow because of long time for reading and updating data.

## RECOMMENDATION

In our future research, we will investigate how 2PC-I can handle transaction processing for very large size database and big number of attributes. We tested the applicability of the transactional workflow technique and the success rate of its transactions, we will tests the performance of 2PC-I compared to the standard 2PC.

## REFERENCES


Chandra, G.D., 2017. NoSQL: Database for Storage and Retrieval of Data in Cloud. CRC Press, Boca Raton, Florida, USA., ISBN:9781498784368, Pages: 455.

Chockler, G., D. Malkhi and D. Dolev, 2003. A Data-Centric Approach for Scalable State Machine Replication. In: Future Directions in Distributed Computing, Schiper, A., A.A. Shvartsman, H. Weatherspoon and B.Y. Zhao (Eds.). Springer, Berlin, Germany, ISBN:978-3-540-00912-2, pp: 159-163.

Coronel, C. and M. Steven, 2016. Database Systems: Design, Implementation and Management. Cengage Learning, Boston, Massachusetts, USA.,.

Coulouris, G., J. Dollimore and T. Kindberg, 1994. Distributed Systems: Concepts and Design. 2nd Edn., Addison-Wesley, USA., ISBN-13: 9780201624335, Pages: 644.

Elmagarmid, A.K., 1992. Database Transaction Models for Advanced Applications. M. Kaufmann Publishers, ?Burlington, Massachusetts, USA., ISBN: 9781558602144, Pages: 611.

Elmagarmid, A.K., Y. Leu, W. Litwin and M.E. Rusinki, 1990. A multidatabase transaction model for interbase. Proceedings of the 16th International Conference on Very Large Data Bases, August 13-16, 1990, Morgan Kaufmann Publishers Inc., San Francisco, California, USA., ISBN:1-55860-149-X, pp: 507-518.

Elmasri, R. and S. Navathe, 2011. Fundamentals of Database Systems. 6th Edn., Addison-Wesley, Boston, Massachusetts, USA., ISBN:9780136086208, Pages: 1172.

Gary, D.W. and K.C. Panos, 1997. PRO-MOTION: Management of mobile transactions. Proceedings of the 1997 ACM Symposium on Applied Computing, April 1, 1997, ACM, New York, USA., pp: 101-108.

Georgakopoulos, D., M. Hornick and A. Sheth, 1995. An overview of workflow management: From process modeling to workflow automation infrastructure. Distrib. Parallel Databases, 3: 119-153.

Hegazy, O.M., A.H. El-Bastawissy and R.F. Ibrahim, 2008. Handling mobile transactions with disconnections using a mobile-shadow. Proceedings of the 6th International Conference of Informatics and Systems, March 27-29, 2008, Cairo University, Giza, Egypt, pp: 1-8.

Holt, B., 2011. Writing and Querying MapReduce Views in CouchDB: Tools for Data Analysts. O'Reilly Media, Sebastopol, California, ISBN:978-1-449-30312-9, Pages: 63.

Ibrahim, R.F., 2017. Agent mobile transaction model. Intl. J. Inf. Electron. Eng., 7: 48-54.

Kumari, A. and Chanderkant, 2012. New approaches of transaction processing in distributed database system. Intl. J. Comput. Sci. Commun., 3: 101-104.

Molina, G. and K. Salem, 1987. Sagas. ACM. SIGMOD Rec., 16: 249-259.

O'neil, P. and E. O'nell, 2001. Database Principles Programming Performance. Morgan Kaufmann Publisher, Burlington, Massachusetts, ISBN:9781558604384, Pages: 870.

Salem, K., H. Garcia-Molina and R. Alonso, 1989. Altruistic locking: A strategy for coping with long lived transactions. High Perform. Trans. Syst., 1: 175-199.

Taft, R., E. Mansour, M. Serafini, J. Duggan and A.J. Elmore *et al*., 2014. E-store: Fine-grained elastic partitioning for distributed transaction processing systems. Proc. VLDB. Endowment, 8: 245-256.

Thomas, M.C. and E.B. Carolyn, 1999. Database Systems: A Practical Approach to Design, Implementation and Management. Addison-Wesley, Boston, Massachusetts, USA., ISBN:9780201342871, Pages: 1093.

Wilson, A., 2003. Distributed transactions and two-phase commit. SAP SE, Walldorf, Germany.