Agent Mobile Transaction Model

Romani Farid Ibrahim

***Abstract*—Transaction is an important concept in the database systems, it moves the database from consistent state to a new consistent state, so that users trust the information retrieved from information systems. A mobile user usually works offline, when connecting to the system, he needs to synchronize his work with the primary database server and updates his local copy, but database may be changed while he is working offline. In this paper we introduce the M-Shadow-Agent model which is a modification of the M-Shadow technique to avoid mobile disconnection problems and handling data changing problem while the mobile user is disconnected or working offline, by using the actionability and the business rules. It increases the transaction success probability, and reserves database consistency. In this paper, the solution concerns on avoiding disconnection, and transaction is not divisible and transaction is one compound transaction only.**

***Index Terms*—Concurrency control, mobile database, transaction, shadow paging, nested transaction, saga, caching, group transaction, Vital subtransaction, cloud computing.**

I. Introduction

The values of all information systems are based on the accuracy and consistency of their databases which are based on the transaction concept. We can say that the benefits of computer systems are appeared after the appearance of the transaction concept, and it is the reason for the diffusion of information systems and computer systems. Accessing data anywhere-anytime-anyway it becomes real events, but this should not violate the database consistency. The mobile database, or embedded database on a mobile device is starting to become an important player in all practical fields, for example, business, travelling, police, military, medical, etc. The data is entered approximately in its real time, no delay between the events time and the entering time to the database.

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) [1]. 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 [2].

This paper concerns the mobile transaction, which is a transaction performed with at least one mobile host takes part in its execution [3]; 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 [4].

As an example of mobile transactions, we are considering mobile hosts are laptop computers belonging to members of a big salespersons team. The salesperson performs a transaction that handles a customer big order which consists of a group of subtransactions represent sub-orders that can be dependent or independent or partially dependent.

We view a transaction as a program in execution in which each write-set satisfies the ACID properties [5], and the program that updates the database as a three folds module (phases): reading phase, editing phase, and validation and write phase. The main questions we attempt to answer in this paper are: 1- Is there a way to avoid the disconnection problems while the mobile transaction is being executed? 2- If the data on the primary server has been changed while the mobile user is disconnected or working offline, how can the transaction continues its work? 3- What are the effects of business logic on the transaction behavior?

This work modifies the M-Shadow technique which is described in [5]-[7] to avoid disconnection problems while the mobile transaction is being executed, and is called the M-Shadow-Agent model. It is an optimistic concurrency model constructed on the shadow paging technique that is used in deferred database recovery and other OS techniques. Shadow paging technique uses two copies of data items, the shadow copy (original), and the edited copy (current). When a transaction commits, the edited copy becomes the current page, and the shadow copy is discarded, otherwise, the edited copy is discarded and the shadow copy is reinstated to become the current page once more.

The remainder of this paper is organized as follows. Section II reviews related work. Section III presents the important points we considered to propose the M-Shadow-Agent model. Section IV presents the M-Shadow-Agent model for transaction processing. Section V presents a summary of the evaluation and implementation of the proposed model. Section VI summarizes the paper and list future research.

II. Related Work

The authors of [8]-[10] assume that the mobile transaction manager (coordinator) resides at the base station. But, adding transaction management functionality to base stations is likely to overload them, which would not be recommended by wireless service providers. Theoretically, this may be the best choice, and many researchers have selected base stations for incorporating database functions; however, in reality this is not an acceptable solution [11]. In our model, base stations are gateways to the primary server, and the transaction agent

Manuscript received November 4, 2016; revised March 2, 2017.

Romani Farid Ibrahim is with the High Institute of Computer Science and Information, City of Culture and Science- 6 October City, Egypt (e-mail: romanifarid@hotmail.com).

doi: 10.18178/ijiee.2017.7.2.660

resides at the sever unit, and it submits the transaction data after editing at the mobile unit to a stored procedure at the primary server to execute the transaction, and receives the execution result message from the primary server and sends it to the mobile unit.

The authors of [4], [9], [12]-[14] assume long disconnection and working offline, rarely changing data (Insurance data, Patients data, etc) and the mobile unit has replica or caching subsystem. Our model avoids disconnections, uses normally changing data, and no replica or database system on the mobile unit only the application program.

In [14]-[16], the authors assume that mobile replica is logically removed from the master copy of the object and is only accessible by the transaction on the mobile unit. In [8], [9], [12], [15]-[18], the authors did not consider the case of changing data on the primary server while the transaction is being executed so that the transaction will abort.

III. Important Consideration

In this section, we present the important points we considered to propose the M-Shadow-Agent model: compound transaction, transactions and grouping, attributes and transaction behavior (actionability), and transaction mathematical functions.

A. Compound Transaction

In the previous example, salespersons mobile transaction application, a transaction is a compound transaction consists of group of independent or groups of dependent which may include partially-dependent subtransactions.

Fig. 1 shows a compound transaction that consists of six subtransactions, they are executed in sequence according to their numbers. The relationships between subtransactions can be dependent or independent or partially dependent according to the business logic.

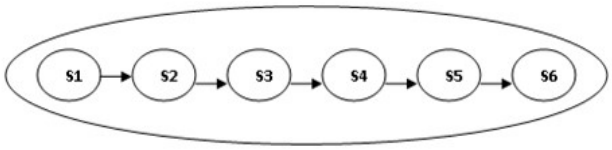

Fig. 1. Compound transaction.

B. Transactions and Grouping

Simple transaction is 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.

Simple transaction by nature is independent, but when it is grouped with other subtransactions in a compound transaction (CT), it has three cases:

- It does not lose its independency property, so it can commit alone.
- It loses its independency property, and it has a dependency relationship with its CT. IF it fails, the CT fails, if the CT fails for any reason, the subtransaction fails also.
- If it is a non-vital subtransaction, it can abort alone and doesn't effect on vital subtransactions of the CT and the CT can commit without it.

C. Attributes and Transactions Behavior

Business rules or constrains are additional constrains specified by users or database administrator that must be satisfied in a database to be consistent. For example, a sales applications have a rule sold-amount ≤ amount-on-stock that should be satisfied at all times. Usually, business constraints include relational operators as (<>, <,>, ≤, ≥), which has a range of values that the data-item can be changed within it, this is called the acceptance range. In the previous example, the sold-amount value can be assigned any value from a range of values which less than or equal to the amount-on-stock value.

In the papers [5]-[7], the actionability concept is introduced which describes how a transaction behaves if a value change is occurred on one or more of its attributes during its processing time by other transactions. Other than Key attributes (K), actionability classifies the data items uses by a transaction into five types: change-accept, change-aware, change-reject, change-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.

The previous types of attribute are to be declared for each transaction type. If omitted, the complete set of attributes will be handled as Change-Reject type (the default actionability type), a case in which the M-Shadow-Agent works like the

traditional shadow paging technique.

Table I illustrates the applied validation rules. If the Change-Accept attribute and the Change-passing attributes are changed or not, it doesn't have any effect on the transaction behavior that updates the Change-Aware attributes. Also, Change-Accept attributes are very rarely changing attributes, for example, item-description, employee-name; Birth-Date, etc., are approximately fixed value attributes.

TABLE I: ACTION ABILITY TRUTH TABLE

Change in				Integrity Constrains Violation	T Succeed
Change-Accept Attribute	Change-Reject Attribute	Change-Aware Attribute	Change-passing Attribute		
Y / N	N	N	Y / N	NA*	Y
Y / N	Y	N	Y / N	NA	N
Y / N	N	Y	Y / N	N	Y
				Y	N

*NA means Not Available

D. Transaction Mathematical Functions

Transactions that update numeric attributes (e.g., change aware attributes) of business applications can be classified into two types according to their mathematical functions:

- Cumulative transactions: which include only add or/and subtract operations (+, -). It changes the value of change aware attribute by adding or subtracting Δ(w), which is the difference between the new value and the old value. Δ(w) = New_Value (w) - Old_Value (w). The function can be written as f(w) = w ± Δ(w).
- Non-Cumulative transactions: which use other mathematical functions such as Power, Div, Multiply, Len, Log, Sqrt, Sin, Cos, Tan, etc. The business constrains of these types of applications can affect on the transaction behavior in three different ways:
  - The business logic accepts the change of the attributes and continues with the transaction as in the case of cumulative transactions, and adding or subtracting Δ(w) to the attribute and commit transaction, or
  - recalculating f(w) according to the current (w) at the validation and write phase at the primary server and commit transaction, or
  - abort transaction.

By using the properties of business constraints, actionability rules and the characteristics of transaction mathematical functions, the M-Shadow-Agent model is built that allows transactions to continue their works and commit even if the shared data items at the primary server have been changed. So that roll-backing of transaction is avoided which raises the performance of the system.

IV. THE M-SHADOW-AGENT MODEL

This section introduces the architecture of the M-Shadow-Agent model, description of the validation test, summary of Shadow-Agent model steps, determination of transaction type, a pseudo code for the validation-write procedure that can be written as a part of the DBMS or as a stored procedures at the primary server side, and the M-Shadow-Agent model and cloud environment.

A. M-Shadow-Agent Model Architecture

Fig. 2 shows the proposed M-Shadow-Agent model architecture. The architecture is composed of mobile units or stations as a client, the basic components of the wireless communication networks are BSS and MSC, and a server unit that includes a database server and the transaction agent module. The system is a distributed system and it is repeated for each cell, but for simplification the architecture for one cell only of the system is shown.

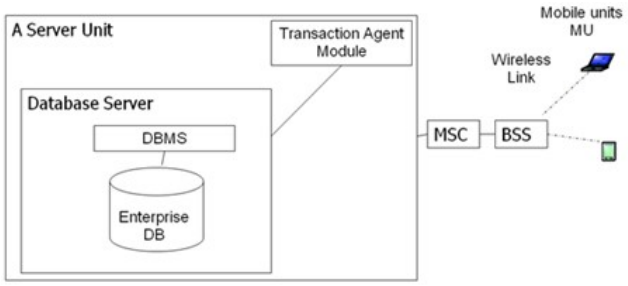

Fig. 2. M-Shadow-Agent model architecture.

The Base Station Subsystem (BSS) is the physical equipment that provides radio coverage to prescribed geographical areas, known as the cells. It contains equipment required to communicate with the mobile units. Functionally, a BSS consists of a control function carried out by the Base station Controller (BSC) and a transmitting function performed by the Base Transceiver Station (BTS). The BTS is the radio transmission equipment and covers each cell. A BSS can serve several cells because it can have multiple BTSs.

The Mobile Service Switching Center (MSC) performs the necessary switching functions required for the MUs located in an associated geographical area. It monitors the mobility of its subscribers and manages necessary resources required to handle and update the location registration procedures and to carry out the handover functions. The MSC is involved in the interworking functions to communicate with other networks such as PSTN and ISDN.

A server unit is a computer that includes a database server and the transaction agent module. The database server accesses the local databases and performs transactions processing. The transaction agent module works as a middle ware between mobile unit and database server.

B. Description of Validation Test

The validation test compares the original values of transaction data items (old data) with its current values on the primary server, the transactions succeed and commit in three cases:

- No change, which means that the original values are equal to the current values on the primary server.
- 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 don't affect on the current transaction.

- Constrained change, which means that some Change-Aware attributes has been changed by other transactions while working offline, but still these changes within the integrity constraint acceptance range and are acceptable according to business rules.

The validation test fails and transactions rollbacks in the following two cases:

- Significant change, which means that some Change-Reject data items have been changed during the transaction processing and/or working offline.
- Out-of-Constraints change which means 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.

C. The M-Shadow-Agent Model Steps

1. The mobile unit sends the query to retrieve the current dataset to transaction agent which passes it to the primary database server (Reading phase).
2. The primary server sends the current dataset to the transaction agent which passes it to the mobile unit. If the mobile unit is unavailable, the transaction agent drops the dataset to decrease the traffic on the network.
3. The mobile user edits the dataset [modify, add, delete], and the mobile application determines the type of transaction mathematical functions and save a copy of original and edited datasets at the mobile unit as XML file. (Editing phase)
4. The mobile unit sends the original dataset and the edited dataset to the transaction agent.
5. The transaction agent submits the original dataset and the edited dataset to the primary database server.
6. The primary database server call the Validation-Write-Procedure (section 4.4) to perform the validation test which uses exclusive lock mode.
7. If the validation test is succeeded, the primary server commits the transaction and sends a state message to the transaction agent which passes it to the mobile unit, If the mobile unit not exists in the transaction agent cell, the message is sent through the network tunneling process to the mobile unit in its current location.
8. If the validation test is failed because of integrity constrains, the primary server rollbacks the transaction and sends a state message to the transaction agent which passes it to the mobile unit in its current location.
9. If the system failed while the validation test is being executed, the recovery mechanism will be performed and restores the database to a consistent state.
10. If the mobile unit is disconnected or failed after sending its data (original and edited) to the transaction agent, then when it connects to the system will receive the transaction state message in its current location.
11. If the mobile unit is disconnected or failed before or while sending its data (original and edited) to the transaction agent, when it connects to the system, if its data is available on it, the mobile unit sends the data to the transaction agent, otherwise it initiates a new transaction. Mobile unit may be disconnected because of crossing boundaries between cells.

Group transaction

12. In independent group transaction, the mobile unit receives a message state for each subtransaction (committed or aborted), then it removes the subtransaction data from the edited dataset. If the system failed while the transaction is being executed, the mobile unit sends the remaining part of the independent group transaction datasets to the transaction agent to be processed.
13. In dependent group transaction, the mobile unit receives one message only at the end of transaction execution. If the system failed while the transaction is being executed, the mobile unit resends all the dependent group transaction datasets to the transaction agent to be executed. If any subtraction fails because of integrity constraints the transaction is aborted.
14. When the transaction completes (committed or aborted), the mobile application on the mobile unit deletes the xml files, and transaction agent removes the transaction data from the memory.

We assume lazy replication protocol to update other secondary replicas of the database.

D. Determination of Transaction Type

The mobile unit application determines the type of the transaction based on the mathematical function that is used to calculate the new value of the change aware attribute. If the mathematical function includes only addition and/or subtraction operators (+, -) then the transaction is cumulative and set the flag *lflag = 0*, otherwise it is non-cumulative and set *lflag = 1*. Also the parameter *br* (abbreviation of business rules) takes one of three values: 0 means the business logic accepts the case as cumulative function, 1 means the business logic requires recalculation of the value of the change aware attribute according to the current value at the primary server, and 2 means the business logic rejects the changes and aborts the transaction.

The following part of the algorithm shows a pseudo code for the update of the change aware attributes. Section *B* (Description of validation test) is an overview for the logic used in this pseudo code.

```
Validation-Write-Procedure (Record original, Record
shadow, String read-query, String update-query, integer Lflag,
integer br)  --- using exclusive lock
Aware-Update ()
{ integer flag=0
    For each change-reject-attribute(i)  in shadow-rec
If   Current.R(i) <>  Shadow.R(i)     then
      Flag = -1
     Return (flag)
    End If
 Next-For
For each change-aware-attribute(i)   in shadow-rec
 If   Current.W(i) =  Original.W (i)    then
  Current.W(i)  = shadow(W(i))
 Else
   If Lflag = 0 then
```

```
    ΔW(i)  = Shadow.W(i)  - Original.W (i)
    Current.W(i)  = ΔW(i)  +  Current.W(i)
  Else
    Call non-cumulative (current.w(i))
End if
 End if
  If (check-constraints(current.W(i)  ) = False )
then
  Flag = -2
     Return (flag)
 End if
 Next-For
}
```

The above function takes 6 inputs: the original record and the modified record, the read query, the update query, *lflag* denotes to type of transaction function (0= cumulative, 1=non-cumulative), and *br* parameter denotes to business rules (0=accept as cumulative, 1=recalculate, 2= abort transaction). Variable *flag* returns -1, or -2 in case of transaction failure. The algorithm checks the status of each change reject attribute, if the value of any change reject attribute is changed it returns *flag = -1* and transaction abort. Then it checks the status of each change aware attribute, if it is not changed, it accepts the shadow values as the current value of the attribute. If a value of a change aware attribute is changed and *lflag = 0* then it recalculates the value of the change ΔW(i) and adds it to the current value. If *lflag =1* then it calls non-cumulative function to handle the recalculation of the new current value according to the business rules (*br*) value. Then it checks that the new current value doesn't violate the integrity constrains of the database.

TABLE II: EXAMPLES OF CUMULATIVE AND NON-CUMULATIVE TRANSACTIONS

T1 (Cumulative)	T2 (Non- Cumulative
Read Phase:	**Read Phase:**
Xoriginal = 200, X >= 0	Xoriginal = 200, X >= 0
Edit Phase:	**Edit Phase :**
Xshadow = X-40= 160	Xshadow = X* 8/10 = 160
Validation Phase:	**Validation Phase:**
Xcurrent = 50	Xcurrent = 50
Δ(x) = Xshadow – Xoriginal	**Calculate (NewXshadow)**
- 40 = 160 - 200	50 *8/10= 40
Xcurrent = Xcurrent + Δ(x)	Xcurrent = Xcurrent - NewXshadow
10 = 50 -40	10 = 50 - 40
Check-Constrains (10)	Check-Constrains(10)
Commit	Commit

The following example shows how the M-Shadow-Agent model handles both cumulative and non-cumulative transactions. In Table II, T1 is applied as a cumulative transaction and T2 as a non-cumulative transaction. Both transactions decrease the value of the change-aware attribute (X) by 40 units, and both cases recalculate the new value (X). In the cumulative case, the recalculation is done by applying Δ(x). But in the non-cumulative case, the recalculation is done by retrieving the current value of (X) and applies the mathematical function to calculate the new shadow value and the new value of the change aware attribute (X). In the later case, the mobile user should pass a parameter that determines if the new shadow value will be incremented or decremented to/from the current value, to produce the final current value of the change aware attribute (X), or it can be implemented in separate modules.

E. M-Shadow-Agent Model and Cloud Environment

Cloud computing is defined by National Institute of Standards and Technology (NIST) as a model for enabling ubiquitous, convenient and on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [19]. The main advantages of the cloud are cost saving, high availability and elasticity. Cloud computing is a stateless system, therefore, to impose transactional coherency upon the system, additional overhead in the form of service brokers, transaction managers, and other middleware must be added to the system. This can introduce a very large performance hit into some applications [15]. For business application that applying short duration transactions with small number of attributes and normal database size, the M-Shadow -Agent model can be used to implement transaction processing, because it avoids the disconnection problems and handles data changing on the primary server while the mobile unit is disconnected. M-Shadow -Agent model implements the core of the transaction processing on the wired network between the agent and database server, only the initiation of the transaction from the mobile unit and the result of the transaction processing is sent by the transaction agent to the mobile unit.

V. EVALUATION AND IMPLEMENTATION OF THE M-SHADOW-AGENT MODEL

We implemented a prototype system to test the electiveness of our model on the success rate of transaction processing in cases of disconnection and data changing on the primary server. We simulated the disconnection state by changing the wireless connection to disconnected state.

We implemented the transaction agent as a program at the primary server. The mobile unit saves the transaction data (Old and edited data) at the primary server as XML files, the transaction agent loads the XML files and submits it to a stored procedure at the primary server to execute the transaction, and receives the execution result message from the primary server and sends it to the mobile unit. We used VB.Net 2010 and SQL Server 2008.

We compared between traditional online transaction processing and our model. Traditional online transaction processing is based on the availability of data and locking data from the beginning of the transaction. But if data is unavailable because it is locked by other transactions, then the delay time depends on the period of locking and number of active transactions. If disconnection happened while transaction is being executed, then transaction fails and

should be resubmitted. In this case, data changing is not applicable since transaction locks data from the beginning.

Our model avoids disconnection, since transaction starts its execution at the primary server and it fails only in case of primary server failure and in this case the recovery manager restores the database to a correct state.

To test our model in the case of data changing on the primary server, we developed a program that uses regular optimistic concurrency control technique and a program that uses our actionability and business logics rules. Both are equals in case of changing the value of a change reject attribute. But in case of changing the numeric value of change aware attributes, transactions that uses regular concurrency control fails, but transactions that uses our models succeed if data changed on the primary server but still within the integrity constrain acceptance range.

Table III shows a comparison between the transaction success rates in disconnection case of the M-shadow-Agent model and the traditional transaction processing model. Transactions uses M-shadow-Agent model usually succeed.

TABLE III: COMPARISON BETWEEN TRANSACTION SUCCESS RATE IN DISCONNECTION CASE

	M-Shadow-Agent mobile transaction model		Traditional transaction processing model	
No of disconnection	Succeeded	Failed	Succeeded	Failed
100	100	0	0	100

Table IV and Fig. 3 show a comparison between the transaction success rates in case of changing the value of Change-Reject attributes of the M-shadow-agent model and the regular Optimistic concurrency control model. Both are equals.

TABLE IV: COMPARISON IN CASE OF CHANGING THE VALUE OF CHANGE-REJECT ATTRIBUTES

	M-Shadow-Agent mobile transaction model		Optimistic Concurrency Control model	
Change Rate	Succeeded	Failed	Succeeded	Failed
10%	90	10	90	10
25%	75	25	75	25
40%	60	40	60	40
50%	50	50	50	50

Table V and Fig. 4 show a comparison between the two models of the transaction success rates in case of changing the value of Change-Aware attributes, but still within the integrity constrain acceptance range. The result shows that the number of failed transactions in regular Optimistic concurrency control model is equal to the ratio of the change. In the M-Shadow-Agent model the transaction usually succeed. If the values of Change-Aware attributes violate the integrity constrain of the database, then in both models transactions will fail.

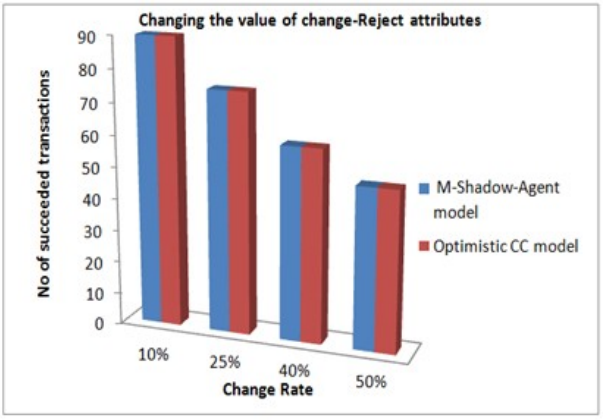

Fig. 3. Comparison between the transaction success rates in case of changing the value of Change-Reject attributes.

TABLE V: COMPARISON IN CASE OF CHANGING THE VALUE OF CHANGE-AWARE ATTRIBUTES

	M-Shadow-Agent mobile transaction model		Optimistic Concurrency Control model	
Change Rate	Succeeded	Failed	Succeeded	Failed
25%	100	0	75	25
50%	100	0	50	50
75%	100	0	25	75
90%	100	0	10	90

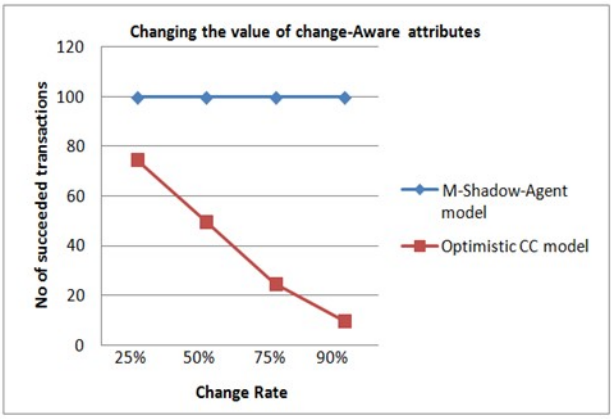

Fig. 4. Comparison between the transaction success rates in case of changing the value of Change-Aware attributes.

VI. CONCLUSION AND FUTURE WORK

In this paper, we introduced the Mobile-Shadow-Agent model which avoids disconnection problems while transaction is being executed, and handling data changing on the primary server while the mobile unit is disconnected or the mobile user is working offline, by using the actionability rules and business logic constrains. M-Shadow-Agent model increases the transaction success probability, and this by consequence, raises the performance of the system. It reserves database consistency.

In future work, we will improve the model and measure the performance under different workloads. Also, we will extend this work to support complex business applications that include a big number of shared data items and complex computations, and we will investigate how a transaction

model can be build for other types of applications that have different characteristics than business applications. Also we will investigate how a transaction model can be build to deal with big data on cloud computing and parallel processing.

REFERENCES

[1] P. O'neil and E. O'nell, *Database Principles Programming Performance*, 2nd ed. Academic press, 2001, ch. 9, pp. 645-646.

[2] T. M. Connolly and C. E. Begg, *Database Systems- A Practical Approach to Design, Implementation, and Management,* 4th ed. Addison Wesley, 2005, ch. 20, pp. 616-618.

[3] P. Serrano-Alvarado, C. L. Roncancio, and M. Adiba, "Analyzing mobile transactions support for DBMS," in *Proc. the 12th International Workshop on Database and Expert Systems Applications (DEXA'01)*, 2001.

[4] P. K. Chrysanthis, "Transaction processing in a mobile computing environment," in *Proc. the IEEE Worskhop on Advances in Parallel and Distributed Systems*, 1993, pp. 77-82.

[5] O. Hegazy, A. E. Bastawissy, and R. F. Ibrahim, "Handling mobile transactions with disconnections using a mobile-shadow technique," in *Proc. the 6th International conference of Informatics and Systems (INFOS 2008)*, Faculty of Computers & Information-Cairo University, March 2008.

[6] R. F. Ibrahim, "Adaptive M-shadow model for handling mobile database transaction processing," *International journal of Advanced Research in Computer and Communication Engineering (IJARCCE)*, vol. 1, Issue 9, pp. 700–709, November 2012.

[7] R. F. Ibrahim, "Business logic and long lived transactions processing," *International journal of Advanced Research in Computer and Communication Engineering (IJARCCE*), vol. 2, issue 3, March 2013.

[8] N. Nouali, A. Doucet, and H. Drias, "A two-phase commit protocol for mobile wireless environment," in *Proc. the 16th Australasian Database Conference (ADC 2005)*, Australian Computer Society, Inc, 2005.

[9] M. Dunham and A. Helal, "A mobile transaction model that captures both the data and movement behaviour," *Mobile Networks and Application (MONET)*, pp. 149–162, 1997.

[10] Y. H. Loh, T. Hara, M. Tsukamoto, and S. Nishio, "Moflex transaction model for mobile heterogeneous multidatabase systems," in *Proc. the symposium on Applied computing*, ACM, March 2000.

[11] V. Kumar, *Mobile Database Systems,* John Wiley & Sons, 2006, ch. 7, pp 175- 176.

[12] S. Vaupel, D. Wlochowitz, and G. Taentzer, "A generic architecture supporting context-aware data and transaction management for mobile applications," in *Proc. the International Conference on Mobile Software Engineering and Systems (MobileSoft)*, May 2016.

[13] G. D. Walborn and P. K. Chrysanthis, "PRO-MOTION: Management of mobile transactions," in *Proc. ACM symposium on Applied computing*, April 1997, pp 101-108.

[14] J. Holliday, D. Agrawal, and A. E. Abbadi, "Disconnection modes for mobile databases," *Wireless Networks*, July, 2002, vol. 8, issue 4, pp. 391-402.

[15] J. Holliday, D. Agrawal, and A. E. Abbadi, "Exploiting planned disconnections in mobile environments," in *Proc. the 10th IEEE Workshop on Research Issues in Data Engineering*, February 2000, pp. 25–29.

[16] G. D. Walborn and P. K. Chrysanthis, "Supporting semantic-based transaction processing in mobile database applications," in *Proc. the 14th IEEE Symposium on Reliable Distributed Systems*, September, 1995.

[17] N. Prabhu, V. Kumar, I. Ray, and G. C. Yang, "Concurrency control in mobile database systems," in *Proc. the 18th International Conference on Advanced Information Networking and Application (AINA'04)*, IEEE, 2004, pp. 83-86.

[18] N. Krishnakumar and R. Jain, "Escrow techniques for mobile sales and inventory applications," *Wireless Networks*, August, 1997, vol. 3, issue 3, pp 235–246.

[19] P. Mell and T. Grance, "The NIST definition of cloud computing," NIST special publication, pp. 800-145, 2011.

Romani Ibrahim received the Ph.D. degree in information systems from Cairo University, Egypt. He works in the High Institute of Computer Science and Information, City of Culture and Science, 6 October City, and as a part time lecturer in the Institute of Statistical Studies & Research - Cairo University. He is a member of ACM, his research interests include distributed and mobile database systems, transaction processing, data warehousing, Cloud Computing, and information security.