Handling Disconnection in Mobile Database Transaction

Romani Farid Ibrahim
High Institute of Computer Science and Information - City of Culture and Science-6 October City, Egypt. romanifarid@hotmail.com

Abstract- In this paper, our concern is the management of disconnection in mobile transaction that access temporal database and/or spatial database. We improve and extend the Mobile-shadow (M-Shadow) technique which handle disconnection in mobile transaction that access temporal database. M-Shadow uses a notation of actionability, which differentiates the actions to be taken during the transaction’s validation phase according to the types of affected attributes. The M-Shadow technique increases the success probability of transactions processed under optimistic concurrency techniques.

Keywords: Concurrency Control, Mobile Database, Transaction, Shadow paging, Saga, Caching, Compensation, Temporal database, Spatial database, GPS.

1 Introduction

We are in the age of mobile devices and wireless networks, accessing data anywhere-anytime-anyway will be real but this must be consistent. The mobile database, or embedded database on a mobile device, is starting to become an important player in all practical fields, for example, business, traveling, police, military, medical, etc. The data will be entered approximately in its real time, no delay between the events time and the entering time to the database.

Mobile transaction is a transaction performed with at least one mobile host takes part in its execution [4]; 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 [5]. The mobile user, by nature, is moving from one place to another so the mobile transaction should follow the user anywhere, which is not supported in distributed database transactions.

We view a transaction as a program in execution in which each write-set satisfies the ACID properties [1], and the program that updates the database as a three folds module (phases): reading phase, editing phase, and validation and write phase. The main question we attempt to answer in this paper is, if the data on the primary server has been changed while the mobile unit (MU) is disconnected or working offline, how can the transaction continue its work?

The proposed M-shadow technique is an optimistic concurrency technique constructed on the shadow paging technique that is used in deferred database recovery and other OS techniques. In this article, we illustrate a survey of related efforts in section 2, section 3 is dedicated to describe the important points we considered to propose the new technique, however the description of the M-Shadow technique, its relative usages, and advantages are detailed in section 4, and we conclude this work and think about the future in section 5.

2 Related Work

Most of the work handling mobile transactions as (Kangaroo, Reporting and Co, Moflex, Escrow techniques…) assume that the handoff process is under the mobile support station (MSS) responsibility [7], and the mobile support stations has the capability to transfer control and transaction history among servers while handoff procedure as [3], [5], [6]. However, this approach has many limitations, such as, if the mobile unit moves relatively slow such that the probability of the commitment protocol terminating at the same cell is high. If it is fast moving then a frequent migration of the control may increase the protocol latency and thus its vulnerability [7]. In addition, if a big number of MUs move among cells, so that most of the response time is spent in transferring data among cells.

Most of the papers assume rarely changing data (Insurance data, Patients data, etc); the mobile unit has replica or caching subsystem. And, the mobile replica is logically removed from the master copy of the object and is only accessible by the transaction on the mobile unit [9], so that they do not consider the case of changing data on the primary server while the transaction processing. In addition, they assume long disconnection or working offline and do not consider short disconnection case.

3 Important Considerations

In optimistic methods using shadows, transactions are dependent on all data items by the same degree. A minor change in an item is sufficient to abort a transaction handling hundreds of actions on thousands of data items. Consequently, the probability of a transaction to fail is very high. This failure probability increases with the increase of the number of data items, the disconnection time, and the number of concurrent transactions. This is why the shadow technique is not frequently used in transactions management.
The m-shadow technique we propose offers a solution for the preceding problem and gives the opportunity to widely manage transactions of difficult types such as long and/or mobile transactions. In m-shadow technique, transaction's validation is not tightly coupled to the eventuality of encountering modifications (done by other transactions) on the values of one or more of its data items.

In this section, we describe the important points we considered to design our technique for handling mobile transaction with disconnection. Which are: the enterprise constraints acceptance range, the effects of attributes types on the transaction behavior (actionability), and linear and non-linear applications.

3.1 Enterprise Constraints Acceptance Range

Enterprise constraints also known as business rules, which are additional rules specified by users or database administrators that the data must satisfy. Usually, enterprise constraints include relational operators as (\(<\), \(\leq\), \(\geq\), \(\geq\)), which has a range of values that the data-item can be changed within. For example, in the above constrain, sold-amount value can be assigned any value from a range of values which \(\leq\) amount-on-stock value.

By using this property of enterprise constraints, in addition to, the characteristics of the attributes, we can build an algorithm that allows transactions to continue their works even if the shared data items at the primary server have been changed. So that we avoid roll-backing of transactions, if the changes are within the acceptable range of the data-item.

3.2 Actionability and Transactions Behavior

We use the new notion actionability 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 three types: change-accept, change-aware, and change-reject.

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 behaviour.

Change-Aware (W): The 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.

The previous three types of attribute actionability 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 works like the traditional Shadow technique. It is also important to note that a transaction may generate a new data item (G) as a function of the three previous types of attributes.

Table (1) illustrates the applied validation rules. If the Change-Accept attribute is 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>
<thead>
<tr>
<th>Change in Attribute(s)</th>
<th>Change in Attribute(s)</th>
<th>Integrity Constraints Violation</th>
<th>T Succeeded</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y / N</td>
<td>N</td>
<td>NA*</td>
<td>Y</td>
</tr>
<tr>
<td>Y / N</td>
<td>Y</td>
<td>NA*</td>
<td>N</td>
</tr>
<tr>
<td>Y / N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
</tbody>
</table>

*NA means Not Available

Rule: If T1, T2 are concurrent transactions, T1 changes a shared Change-Reject attribute and T2 changes a shared Change-Aware attribute that belong to a normalized database then:

- If T1 commits before T2 then T2 must abort.
- If T2 commits before T1 then T1 can continue its processing.

The reasons behind using the actionability include:

- A transaction usually update a part of the data set it uses, the other part of the data elements is asked by the transaction to control the transaction. These data items are read only items and a change in such elements should not prevent the execution of the transaction.
Our concern is on the transactions that update Change-Aware attributes, which have acceptable range. An encountered change in these attributes may affect the outcomes of the transaction but not aborts entirely its execution.

The usage of mobile transactions is still limited to salesman and inventory applications which are, by nature, applying short transactions with little attributes. This fortunately comply well with the M-Shadow concept.

The main advantages of using actionability are: The DBMS can perform the update process automatically based on the actionability type of the data for all applications and a reasonable increase in the succeeded transactions ratio.

3.3 Linear and Non-Linear Applications

Most of papers that handle transaction processing assumed implicitly that all the applications increase or decrease the attributes (in our notation change aware attributes \(W\)) by \(\Delta\) value. But the applications that update a change aware attributes \(w\) can be classified based on the mathematical functions that is used to calculate the new values of the change aware attributes into two types:

- **Linear transactions**: that use the linear function \(f(x) = mx + b\) for the calculation of the new value of change aware attribute. We assume that the \(m\) value usually equals 1, and the \(b\) is the value of the changes that transaction are performed on the \(w\) attribute, which is known as delta \(\Delta\)(\(w\)), and the function can be written as \(f(w) = w \pm \Delta\)(\(w\)), where \(w\) is the original value of the change aware attribute. The function includes only add or/and subtract operations (+, -).

- **Non-linear transactions**: that use other functions that differ from the form of \(f(w) = w \pm \Delta\)(\(w\)). They can include the functions: Power, Div, Multiply, Len, Log, Sqrt, Sin, Cos, Tan, etc. The semantic of these types of applications require to recalculate \(f(w)\) according to the current \(w\) at the validation and write phase at the primary server.

4 Summary of the M-Shadow Technique Steps

4.1 For linear transactions

At Mobile Unit side:

1. Retrieve the current dataset from the primary server (Reading phase)
2. Copy the retrieved dataset as a shadow copy.
3. The user edits the dataset on the shadow copy [modify, add, delete] (Editing phase)

4. Send the original read-set, the edited-set (shadow copy changes), the read-query and, and the update query to the primary server (subtransaction by subtransaction).

At Primary Server Side:

5. Implement the validation and write phase (which can be implemented as a as a part of the DBMS or as a stored procedure at the primary server.
   - Call validation-write-1 procedure (as a part of the DBMS)

   a) Independent Case

6. If one subtransaction fails (disconnection, integrity constraints, etc.)

At Primary Server Side:

- Discard the current write-set subtransaction.

At Mobile Unit side:

- Removes the subtransaction shadow dataset from the shadow copy.
- Send next subtransaction data to the primary server.
- Short disconnection (the user doesn't close the program which means all variables and data-sets still available in the main memory): Try to reconnect.
- Long disconnection( the user wants to close the program): The program saves the data-sets (the original data-set and the remaining elements of the shadow data-set) as XML files on the mobile unit secondary storage to be retrieved at the reconnection time.

When reconnection with the primary server is available:

After short disconnection:

The program resends the write-set data for the subtransaction, which the disconnection happened through its update only. The primary server restarts the write-set subtransaction as in step 5.

After long disconnection:

The program loads the XML files and starts a new independent write-set group transaction for the loaded data-sets (original and shadow) as in step 4.

b) Fully Dependent Case

7. If one subtransaction fails:

At Primary Server Side:
• Rollback the current and all the previous write-set subtransactions of the group.

At Mobile Unit side: because of
• **Integrity constraints violation:** Drops its data-sets and clears the memory to start a new transaction.
• **Short disconnection:** Try to reconnect.
• **Long disconnection:** The program saves the data-sets (the original data-set and the shadow data-set) as XML files on the mobile unit secondary storage.

When reconnection with the primary server is available

**After short disconnection:**
The program reissues the dependent-write-set group transaction as a new transaction as in step 4.

**After long disconnection:**
The program loads the XML files and starts a new fully dependent write-set group transaction for the loaded data-sets (shadow and original) as in step 4.

**Validation-Write Procedure-1 (A General Validation)**

**Algorithm to Be Put as a Part of the DBMS**

**Validation-Write-Phase (Record original, Record shadow, String read-query, String update-query)**

**Aware-Update (integer flag)**

```plaintext
{ 
  For each change-reject-attribute(i) in shadow-rec
    If Current.R(i) <> Shadow.R(i) then
      Flag = -1
      Goto par-out
    End if
  Next-For

  For each change-attribute(i) in shadow-rec
    ΔW(i) = Shadow.W(i) - Original.W (i)
    Current.W(i) = ΔW(i) + Current.W(i)
    If (check-constraints(current.W(i))) = False ) then
      Flag = -2
    Goto par-out
  Next-For
  Par-out:
  Return (flag)
}
```

Table 2 shows an example that describes how the validation and write phase can be applied and assume linear transactions for simplicity. The example shows a bank transaction that transfers $400 from account X to account Y. We use the notations of actionability, K denotes the Key attribute, A denotes a Change-Accept attribute, R denotes a Change-Reject attribute, W denotes a Change-Aware attribute, G denotes a generated attribute, and the subindexes o denotes the original value, s denotes the shadow value and c denotes the current value at the primary sever.

**4.2 Determination of transaction type (Linear or Non-Linear)**

To determine if the transaction is linear or non-liner, we need the program at the mobile unit to be more
intelligent and performs more operations than data entry validation. The mobile unit 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 linear otherwise it is non-linear.

The validation-write procedure will be as follows:

**Validation-Write-Procedure (Record original, Record shadow, String read-query, String update-query, integer Lflag)**

```
Aware-Update (integer flag)
{
  For each change-reject-attribute(i) in shadow-rec
    If Current.R(i) <> Shadow.R(i) then
      Flag = -1
      Goto par-out
    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
        ∆W(i) = Shadow.W(i) - Original.W(i)
        Current.W(i) = ∆W(i) + Current.W(i)
      Else
        Return (current.W(i))
        Call non-linear(current.w(i))
        Goto par-out
      End if
    End if
    If (check-constraints(current.W(i)) = False ) then
      Flag = -2
      Goto par-out
    End if
  Next-For
  Par-out:
  Return (flag)
}
```

Figure (2) shows a graphical representation for the transaction processing between the mobile unit and the primary server. The primary server performs the validation phase under exclusive-lock. If the data does not change at the primary server, it accepts the shadow data and copies it as a current data. If the data at the primary server changed, it checks the linear-flag, if its value is 0, which means linear transaction; it calculates the new value of the change aware attribute by calculating the difference between shadow and original values and add it to the current. If linear-flag is 1, which means non-linear transaction, it returns the current-value of the change aware attribute to the mobile unit to be recalculated at the mobile unit and re-passed to the server after recalculation. In this case, the new value of the change-aware attribute will be validated that it doesn’t violates the enterprise constrains only, because validation phase is under exclusive-lock.

The following example shows how the M-Shadow technique handles both linear and non-linear transactions.

In Table 3, T1 is applied as a linear transaction and T2 as a non-linear 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 linear case, the recalculation is done by applying ∆(x). But in the non-linear case, the recalculation is done by passing the current value (X) at the primary server to the mobile unit which applies the mathematical function of the application that uses to calculate the new value of the change aware attribute (X). Both cases, linear and non-linear are serializable.

<table>
<thead>
<tr>
<th>T1 (Linear)</th>
<th>T2 (Non-Linear)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Read Phase:</strong></td>
<td><strong>Read Phase:</strong></td>
</tr>
<tr>
<td>Xoriginal = 200, X &gt;= 0</td>
<td>Xoriginal = 200, X &gt;= 0</td>
</tr>
<tr>
<td><strong>Edit Phase:</strong></td>
<td><strong>Edit Phase:</strong></td>
</tr>
<tr>
<td>Xshadow = X-40= 160</td>
<td>Xshadow = X * 8/10 = 160</td>
</tr>
<tr>
<td><strong>Validation Phase:</strong></td>
<td><strong>Validation Phase:</strong></td>
</tr>
<tr>
<td>Xcurrent = 50</td>
<td>Xcurrent = 50</td>
</tr>
<tr>
<td>∆(x) = Xshadow - Xoriginal</td>
<td>Send (Xcurrent)</td>
</tr>
<tr>
<td>-40 = 160 - 200</td>
<td>Accept (new-Xcurrent) from the client-agent</td>
</tr>
<tr>
<td>10 = 50 -40</td>
<td>(50 * 8/10= 40 )</td>
</tr>
<tr>
<td>Check-Constrains (10)</td>
<td>Check-Constrains(40)</td>
</tr>
<tr>
<td>Commit</td>
<td>Commit</td>
</tr>
</tbody>
</table>

For simplicity in the previous example, we used \( f(X) = X^{8/10} \), but it can be any function of the non-linear functions as \( f(X) = X^2 \) or \( f(X) = \log(X) \), etc. The logic that
has been applied on this function can be applied using any other non-linear function.

4.3 Supporting Location Dependent Data Access in M-Shadow Model

M-shadow model is designed to handle the processing of transactions that access location independent data (Temporal database). It can be extended to support transactions that access location dependent data (Spatial database). To do that, the following points should be considered:

1. The database should be a distributed database and data are distributed based on the geographical location.
   • For our sales application, we assume that the company has many branches. Each branch represents an area (cell), and it has a local database server which stores available data items for sale in this branch, and works as a primary server for this area. The salesman moves among cells and has access to all local servers, but customer orders should be issued to the local server of each cell which serves the customers of its area.

2. We assume that the mobile database system of the company can connect to a Global Positioning System (GPS) that determines the location of the mobile unit (latitude/longitude) and answers the location dependent query as: Find the nearest hospital or restaurant from the current location of the mobile unit?

3. Transactions table should include attributes to store the X, Y (latitude/longitude) location of the mobile unit and the time of issuing the transactions. So that, we extend the actionability classification of attributes to include this type of attributes by adding a new type of attributes called location-time-attribute (L).

4.3.1 Spatial Queries based on Temporal Data

Most of researchers as [6], [14] assume that the queries that require spatial data is retrieved from a GPS that includes a spatial database and a spatial database management system that has the capabilities to handle this type of queries as: find the nearest hospital or restaurant from the current location of the mobile unit, list the local weather, etc. But, in our example, if the salesman decided to visit his customers according to their nearest from his current location and typed the query: Find the nearest customer (address) to my current location? The problem here is that the customer address is in the temporal database not in the spatial database. To answer this query, the query analyzer should perform the following steps:

   • Retrieve the customers primary keys and addresses (assume streets) of the salesman.
   • Pass it to the GPS.
   • The GPS compares these data with its database and arranges the data according to the current location of the mobile unit and returns the result to the mobile database system.
   • Based on the returned result from the GPS, the customer details are retrieved from the mobile database system and are passed to the mobile user.

4.3.2 M-Shadow Transaction Model Steps for Location Dependent Data

• **Query Phase**
  1. Get mobile unit location data (x_loc, y_loc)
  2. Get the server identification data
  3. Retrieve datasets from the current server of the current cell (the home-server of the transaction).

• **Edit Phase**
  ♦ As the edit phase of the M-Shadow model for location independent data.

• **Validation and Write Phase**
  1. In general, as the validation and write phase of the M-Shadow model for location independent data.
  2. In addition, in long disconnection case, The mobile unit saves the transaction home-server identification data as XML file.
  3. At reconnection time
    ♦ After long disconnection case, the mobile unit loads the datasets and the transaction home-server identification data from the XML files.
    ♦ Connect with the home-server of the transaction from the current cell (we assume logging to the system is performed automatically using a Kerberos subsystem)
    ♦ Perform the transaction processing as the M-Shadow model at reconnection time for location independent data.

4. After terminating the transaction (commit or abort)
   • If the current cell is not the home-server cell then
     o Retrieve datasets from the primary server of the current cell.
     o Start new transaction.
   Else
o Continue and start a new transaction with the current home-server.
End if

4.4 Advantages and Limitations of the M-Shadow technique

The main advantages of using the shadow technique in general and the M-shadow technique are:

1. Increase the performance of the system, by increasing the success probability of transaction by allowing transaction to continue its work even after disconnection and changing data on the primary server.
2. No transfer of logs or transaction history among sites. Only external files (XML files) would be saved on the mobile unit and will be deleted when the transaction finished.
3. Recovery for active transactions at failure time, which DBMS recovery manager does not do.
5. Decrease the programming time for applications, because the DBMS performs the update process.

The limitations of the M-Shadow 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.

We implemented a sales application that uses the M-Shadow technique using Visual Basic.Net and SQL Server 2005 because they support many new features as writing and reading XML files. We assume that the replication handling is solved as a distributed database problem using the lazy replication technique among fixed hosts.

5 Conclusion and Future Works

In M-Shadow we increase the transaction success probability, this by consequence, raises the performance of the system. Actionability classifies the data elements handled by a transaction according to how much a change on these elements affect the transaction behaviour. It doesn't transfer logs or transaction history among sites and it isn't based on compensation concept. It differentiates between short disconnection and long disconnection. It decreases the programming time for applications. So, it is suitable for handling mobile transaction with disconnection. Future research will extend this work to support complex business applications that include a big number of shared data items and complex computations, and parallel processing and real-time environments. Also, studying how to find the optimal solution for selecting the next server in a shared area among many servers to decrease the number of disconnection.

References


