A Comparative Study of Mobile Database Transaction Models

Ramzi A. Haraty
Department of Computer Science and Mathematics
Lebanese American University
Beirut, Lebanon
Email: rharaty@lau.edu.lb

Abstract--**As communication becomes more and more an integral part of our daily lives, so does our need to access more and more information. Mobility is currently one of the most important factors to consider in our aim to achieve ubiquitous computing, and with it rises the problem of how to manipulate data while maintaining consistency and integrity. This paper presents a comparative analysis of mobile database transaction models.**

keywords—mobility; database transaction models; and ACID properties.

I. INTRODUCTION

Pervasive computing is a term loosely used to describe the current state of computer technology in modern life [1]. Our reliance on computing mediums increases with the need for mobility, connectivity and data availability. We often find that the data we need located on multiple devices and in various locations, is inaccessible directly most of the time. Pervasive computing also encompasses the concepts of data, connectivity and their ubiquitous presence in an individual's daily life. As an example, in a single day, the average individual can go through a minimum of three different devices to perform various everyday tasks such as checking his/her email account on the desktop computer, calling a family member on the cellphone, listening to some music in the background on his/her personal laptop and syncing all appointments from his/her palm-pilot to his/her email client. Current tools, such as Google's plethora of desktop search tools, have reduced the divide by centralizing data management, but they do not address issues such as unrelated data repositories, data safeguard and integrity. In addition, the problem of intermittent connectivity through wireless enabled devices is also a major issue in mobility. As such, maintaining data integrity and consistency in such mobile environments is a challenge - given the diverse factors that influence connectivity, from geography to battery life. Current trends in mobile databases suggest the adoption, among other techniques, of the quorum approach in which multiple mobile hosts perform reads and writes based on the majority vote of hosts selected to the quorum [2]. Although the quorum algorithm has been extensively studied since its earlier days, adapting it to mobile devices with connectivity issues and providing a solid quality of service (QoS) for quorum members is still in its infancy. The adaptation of quorum consensus to mobile environments to insure a high level of service is one of the main challenges to tackle.

This paper presents a comparative analysis of mobile database transaction models. The rest of the paper is organized as follows. Section 2 presents the mobile database systems architecture. Section 3 presents the comparative analysis of the mobile transaction models; and section 4 concludes the paper.

II. THE MOBILE ENVIRONMENT

A. Mobile Database System

Mobile database systems can essentially be summarized as large distributed database systems, with the added property of catering for mobile units that may experience connectivity outages depending on their geographical location or data processing capabilities. The acronym MDS (Mobile Database System) is used to refer to them. An MDS comprises of interconnected computers and communication systems, both wired and wireless (typically GSM or 802.11), which allow users to connect to the systems that host the requested data. Typically, an MDS comprises of Fixed Host (FH) units, interconnected through a high speed wired network, Base Stations (BS) and Mobile Units (MU) or Mobile Hosts (MH), which typically describe a portable computing device ranging from a laptop to a PDA. Figure 1 depicts the various components of an MDS and the interconnectivity among its various components. The mobile unit refers to any hand held device that can be carried by its owner and is able to communicate with other computing devices. These mobile units are typically considered outlying components of the system and are inter-networked through the wireless network infrastructure. This wireless infrastructure can be a typical 802.11 wireless network, a GSM network or a hybrid of both, and comprises of a BS with which the MU communicates directly. The BS, in turn, communicates with a Base Station Controller (BSC), which work in tandem to control and coordinate traffic among the various BS in a given geographical region. Communications from the MU are then routed either through a standard wired network infrastructure or directly to the database server, which handles fetching requested information and sending information back to the MU that originated the request.

It is also worth noting that an MDS would comprise multiple DBSs and various DB configurations, which could be in various geographical regions and contain either replicated data or spatial data pertinent to the geographical location where that DBS is located. The architecture is also dynamic enough to allow semantically related data to be clustered together. The above three DBS types (replicated, spatial, semantic) may also coexist together and provide a hybrid MDS incorporating all of these features. The choice for such configurations is usually related to data availability and redundancy considerations. In the case of geographically related data, this distribution could also serve to provide users with data relevant to their current geographical location as well [3]. As such we can define two broad categories of replication: spatial replication for location dependent data or temporal replication for traditional databases. The main difference being that temporal replication provides a single, unique, consistent value for any accessed data, from any replication site, whereas spatial replication (although provides the same DB structure on all sites) provides a single unique value of requested data, depending on the geographical location where that request has been made. Thus, in spatial replication, data in various geographical locations may have different values that are unique to a particular geographical context. Note that temporal replicas of spatial data, in a particular geographical location, are also possible.

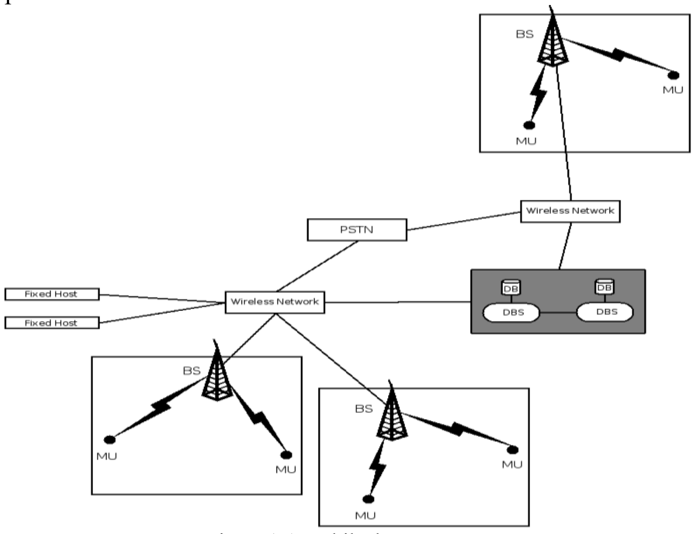

Figure 1.A mobile data system.

B. Transaction Execution in MDS

Transactions are the basic atomic units that carry requests and data between the DBS and the client (MU or FH). In an MDS, the distributed nature of data and the nature of requests made by clients, involves a lot of parallel processing, both to improve system performance as well as provide the necessary data. Transactions are no longer treated as atomic units in distributed databases, but are themselves amenable to being divided into sub-transactions, that are spread and sent to the DBS containing the requested data. The entity responsible for breaking down a transaction into subcomponents is called the coordinator. A coordinator is usually a system that is aware of the network within its coverage zone; spatial location of requested data and geographical location of associated DBSs, making it a crucial component in request dissemination (see Figure 2). A coordinator is a system that should satisfy the following two properties:

- Continuous Connectivity: A coordinator should (in theory) maintain connections with the rest of the system with no downtime or intermittent failures.
- Continuous Availability: It should also be accessible at any time with no downtime and provide comparatively large storage capabilities for cached data.

According to [4] the most suitable entity in Figure 2 satisfying the above requirements, would be the BS acting as the coordinator, as its features include, in addition to the points mentioned above, direct communication with MUs in its coverage zone. Communication between the MU and the DBS typically involves transaction exchanges that can be initiated by the DBS, the MU or both. The processing of these transactions can also take place exclusively at the MU where it was instantiated, exclusively at the DBS, or at both locations simultaneously.

Given that mobile units may move from one coverage area to another, changing the BS/Coordinator to which it is attached, a transaction may complete in one of the following scenarios [5]:

- Static MU: the MU establishes a connection with a given BS, which is designated as its coordinator. If the MU does not move, then the transactions initiated by the MU will complete through the same coordinator it was initiated from.
- Dynamic MU: In this scheme the designated coordinator of an MU can change depending on the location of the MU and the coverage area of the BS.

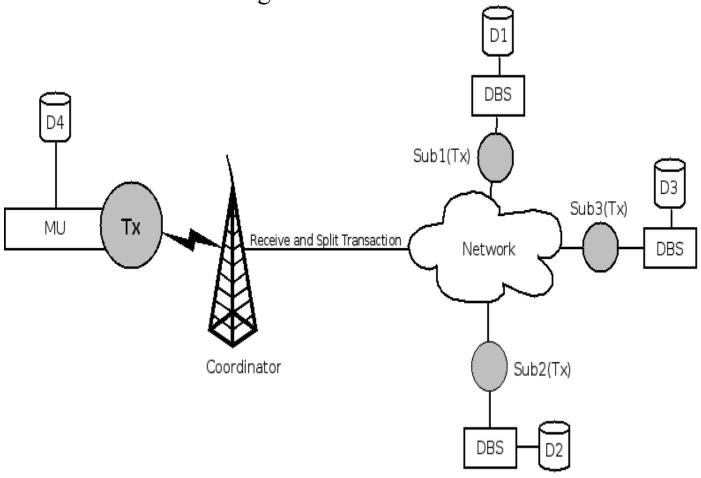

Figure 2. Transaction execution in an MDS.

III. MOBILE TRANSCTION MODELS

This section covers the most recent mobile transaction management and execution models by giving an informative overview of their features and mode of operation, as well as comparing their high-points and low-points. All discussed models adhere (to varying extent) to, or extend, the ACID transaction execution model [6]. This model has been the generic underlying of all transaction execution models because of its proven reliability, insofar as the properties it provides guarantee that transactions are processed in a reliable fashion [7]. A brief explanation of ACID properties follows:

- Atomicity: This property guarantees that any executing transaction is treated as an entity that may not be fragmented into any smaller components.
- Consistency: Ensures that all integrity constraints (regardless of granularity) are maintained to provide an accurate and timely view of data.
- Isolation: This property ensures that no data may be viewed in an intermediate state by one transaction while another transaction is operating on that data. The formal adjective describing this state is referred to as a serializable state [8].
- Durability: Refers to the persistence of the operations of a transaction after successful execution, so that any modifications carried out by that transaction on a data item will not be rolled back.

Mobility introduces new challenges to the way data is handled and presented, because of its spatial quality; the same data in two different geographical locations will have different values. Therefore, maintaining consistency also becomes more complex as the spatial component gets factored in. Most MDS transaction execution models introduce the concept of spatial consistency [9]. The idea consists of providing an MU consistent data, based on its current location, in a way that the owner of the MU can use. As an example, assume user requests information about a particular restaurant which is a mile from his current location. If the returned answer was given back after that user has passed the restaurant, then that request is no longer relevant; the user having left the geographical area where this information would have been useful. The reason for getting a belated response could be due to factors such as bandwidth limitation, MU disconnections, or query processing time.

The following are the current mobile transaction models, each which will be described in detail with accompanying figures, where applicable.

A. Clustering

The clustering model, depicted in figure 3, was introduced by Pitoura et al. in [10] and extended in [11]. It assumes a fully distributed system where data is clustered based on a set of dynamic semantic proximity. These clusters are created and merged dynamically based on either global conditions, or on conditions set by mobile users, which would allow them to cluster frequently accessed data in a way that minimizes access time. Data in a specific cluster is required to be fully consistent, in so far as various versions of the same data cannot co-exist. Different clusters may exhibit what is referred to as bounded inconsistency wherein data items may have different values in different clusters based on a certain set of predefined metrics. The metrics include the number of different copies of the data in all clusters. Once this limit is reached, a reconciliation function takes care of minimizing the value of that metric back to a lower threshold. The model also defines two types of consistency, an inter-cluster consistency and an intra-cluster consistency. Inter-cluster consistency is the equivalent of global consistency in traditional database models, with the difference that inter-cluster states may contain irregularities that fall within the values of the bounded inconsistency threshold. The cluster is referred to as being *m*-degree consistent, where the degree refers to the divergence in the value of the chosen bounded inconsistency. Intra-cluster consistency, on the other hand, is equivalent to strict consistency in traditional database models, whereas no data in the specified cluster may have more than one value associated with it. To achieve this, the model introduces two types of transactions, weak transactions and strict transactions. Weak and strict operations are also introduced in terms or reads and writes. As a rule, strict transactions may apply to a single cluster or be inter-cluster operations leaving the database in a globally consistent state. On the other hand, weak transactions may only be executed within a particular cluster only and modifications by a write operation become permanent once the scheduled reconciliation function is run. In general, strict reads will only read values written by strict writes, and weak reads will read values written by a weak write. A strict transaction becomes a set of strict operations (reads and writes), whereas weak transactions refer to a set of weak reads and writes. Only weak transactions are allowed to be performed by the MH on its dataset. Table 1 summarizes properties and mechanism of the clustering model to maintain ACID properties. Two of the main drawbacks in this approach involve issues with the architecture itself. First, the fact that clustering maintains two different types of data makes replication a very complex operation. Compounded to that, the model does not fully adhere to the durability property of the ACID model, as locally committed transactions may be rolled back due to reconciliation conflicts, leading to a higher degree of cascaded aborts.

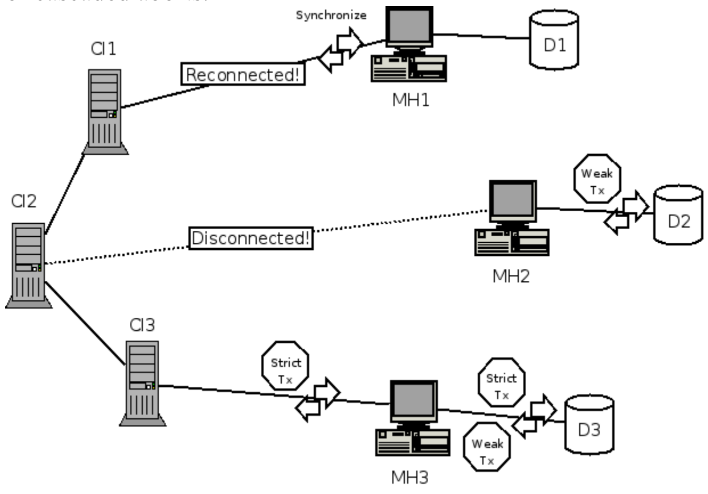

Figure 3. The Clustering Model Architecture.

B. Two-Tier Replication

The two-tier replication model [12], illustrated in figure 4, relies on a lazy replication mechanism geared towards mobile environments. The model introduces the concept of master copies to which fully replicated copies are associated. As with the clustering model, it also classifies transactions in two categories, base transactions, which operate on master copies of data, and tentative transactions, which operate on the replicated copies when a MH is disconnected. As long as the mobile host is connected, it participates in all operations using base transactions to modify stored master copies of the data and propagate these changes through a lazy replication scheme that guarantees one-copy serializability. When an MH is disconnected, it no longer has access to stored master data and may only operate on tentative copies of the data instead, using tentative transactions. Once the connection is re-established, the MH re-executes tentative transactions as base transactions to update its master copy and propagate the data changes. The acceptance of the re-executed operation for final commit is dependent on the predefined acceptance criteria. In case of conflicts, the initiating tentative transaction is aborted. This model allows for semantic divergence between tentative and base data, and reconciliation is done by the re-execution of tentative transactions as base transactions. The adherence of this model to the ACID properties can be found in Table 2.

C. HiCoMo

Introduced by [13], HiCoMo is yet another novel approach to managing transactions in highly mobile environments (see figure 5). Like the two preceding models, it distinguishes between two types of transactions, base and HiCoMo transactions. The difference being that HiCoMo mobile hosts do not operate on base data but on aggregate data (summation, counts, minimum, etc...) which are obtained from base tables. HiCoMo transactions are executed when the MH is operating in disconnected mode. Upon reconnection any modifications performed on the aggregate tables is then re-synced with the base tables using commutative inference and semantics functions, which allow HiCoMo transactions to be transformed into base transactions. A divergence threshold is also tolerated between base and HiCoMo transactions.

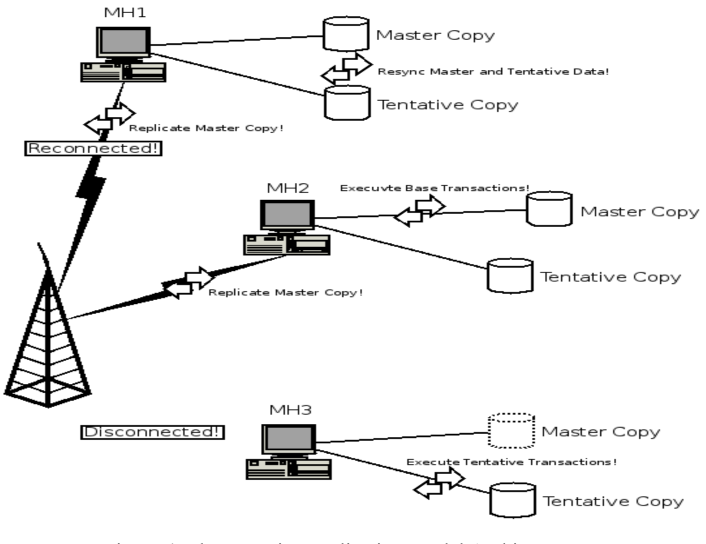

Figure 4. The Two-tier Replication Model Architecture

The ACID properties are maintained through the following features: Atomicity is guaranteed in the use of an extended nested transaction model in which base transactions are treated as sub-transactions and organized in a tree like structure with parent-child dependencies. What is most striking about this approach is its flexible commit model that allows base transactions (and HiCoMo transactions) to be re-executed within a preset error margin (divergence criteria). Once that error margin is exceeded the transaction is aborted. The triggering of the error margin retry, due to a conflict between base transactions, results from the transformation of the HiCoMo transactions into the original base table data (i.e., a constraint on the maximum value of a data field). The re-execution of transactions is allowed in this model due to the commutative nature of the original aggregate data made available to the MH. This commutative feature also provides the model with a high level consistency. In terms of isolation, intermediate values of data are only made visible to local transactions on the MH, whereas concurrency is maintained using an optimistic concurrency control strategy that uses timestamps to order operations and avoid conflict between base and HiCoMo transactions. In terms of replication, correctness is maintained through a convergence scheme that guarantees that data between HiCoMo and base tables always remains within the specific error margin. This condition is guaranteed by the data commitment process. One of the main drawbacks of the HiCoMo model is inherent in its design. The use of aggregate data, although improving local transaction commit times, also makes the conversion process (from HiCoMo to base transactions) rather complex, and limits the possibilities of data manipulation to commutative operations only. Table.3 summarizes the ACID properties of the HiCoMo model.

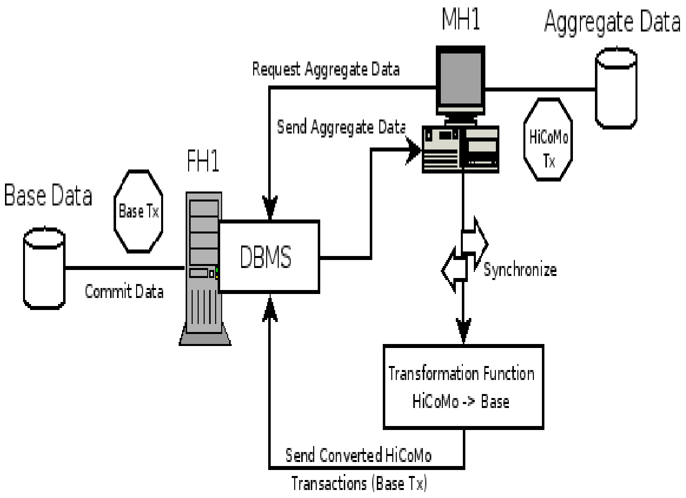

Figure 5. The HiCoMo Replication Model Architecture

D. Pro-Motion

One of the most innovative approaches to mobile databases was introduced by Walborn and Chrysanthis in [14] and is mainly a data caching scheme that allows for consistent local transaction processing. The main innovation of this scheme is the introduction of the concept of compacts. Compacts, as the name suggests, are an agglomeration of data, operations and constraints, which form the basic caching and control mechanisms among MHs and FHs, and considers all operations executed on mobile systems as a very long transaction executed on the server. In other words, all operations are executed on the MH and are synchronized later with the server. A compact manager takes care of the generation and management of compacts on the server, whereas a compact agent takes care of caching and processing transactions on the MH. Interaction between various MHs and FHs are done through the mobility manager in charge of data exchange between various compact agents. The model extends the standard transaction execution model with some specialized methods pertaining directly to the manipulation and querying of compacts to support data manipulation and concurrency schemes. As such, compacts can be inquired about using the “Inquire” method and notified of any changes in the state of the MH using the “Notify” method. Commit and abort operations are also used to validate or invalidate data changes performed by compact transactions. A special “Dispatch” method is used to process operations initiated by the compact agent, which are to be locally committed. The interaction between the compact agent and the manager is confined to four types of transaction processing activities, which involves the agent and/or manager. When the MH is connected to the network, it always attempts to store compacts for an eventual disconnection. This constant storage of compacts is referred to in the model as hoarding. Compacts are stored in a compact registry. While connected and performing its hoarding function, the MH will continually process transaction with the compact manager. Although the model does not differentiate between connected and disconnected modes, when interacting directly with the compact manager, the MH will be operating at a more optimized level than in disconnected mode due to the quick turnaround of operations. This is referred to as connected execution. When disconnected, the MH will revert to local processing of transactions and maintains a log of operations that can be replayed later for either recovery or resynchronization. An overview of the PRO-MOTION architecture is depicted in the Figure 6.

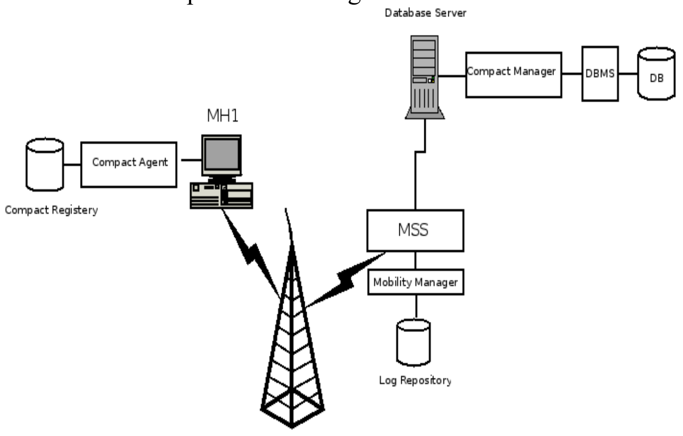

Figure 6. The PRO-MOTION Replication Model Architecture.

After the MH re-establishes a connection with the network, a synchronization process takes place to reconcile the locally committed transactions with the permanent data store on an FH. If no conflicts are detected, all updates are performed. In the eventuality of a conflict or data expiry, the following venues may be undertaken by the system: In case of data expiry, the compact agent will attempt to get a renewal on the data item from the compact manager, pending no other transaction (from another MH) has modified that data after the expiry date. In case both of these operations fail, compacts that have failed to reinstate their changes with the compact manager and incorporate their changes with the permanent data store, are aborted, and all associated compacts are invalidated. Once a list of valid compacts is generated, these are allowed to issue a dispatch event to the server, replaying the operations that have been locally committed on the database server for final commitment. As far as the model's adherence to ACID properties, Table 4 summarizes the various mechanisms and schemes used by PRO-MOTION.

IV. CONLCUSION

This paper presented an overview of the various database transaction models currently available for mobile computing environments. The paper gave an informative overview of the models, their features, and their mode of operation. One of the main points to focus on in future studies, would include studying the impact of building historical track record of mobile hosts based on elaborate regression models (such as a Bayesian regression model [15]). Future work will also entail classifying models based on metrics such as the one in [16] to further refine the classifications of the hosts. This would require that real performance data be made available to the system in order to allow the Bayes engine's learning process to evaluate its current state, based on measurements that reflect the reality of the system.

References

[1] M. Satyanarayanan, Pervasive Computing: Vision and Challenges, IEEE Personal Communications, vol. 6, no. 8, pp. 10–17, August 2001.
[2] S. Younes and R. A. Haraty, An Enhanced Quorum Selection Algorithm, Journal of Computers, vol. 4, no. 7, pp. 654-662, July 2009.
[3] R. A. Haraty and Roula C. Fany, Query Acceleration in Distributed Database Systems. Revista Colombiana de Computación, Volume 2, Issue 1, pp. 19-34, 2001.
[4] V. Kumar, Mobile Database Systems. J. Wiley & Sons Inc., 2006.
[5] P. Krishna, N. Viadya, and D. Pradhan. Static and Dynamic Location Management in Distributed Mobile Environments. Technical Report, Department of Computer Science, Texas A&M University, June 1994.
[6] J. Gray and A. Reuter. Transaction Processing: Concepts and Techniques. Morgan Kaufmann Publishing, San Mateo, CA, USA, 1992.
[7] Serrano-Alvarado, P., et al., (2004). A Survey of Mobile Transactions. Journal of Distributed and Parallel Databases. 16, pp. 193-230, 2004.
[8] R. Elmasri and S. Navathe. Database Systems: Models, Languages, Design, and Application Programming. Pearson, 2011.
[9] S. Servigne, T. Ubeda, A. Puricelli, and R. Laurin. A Methodology for Spatial Consistency Improvement of Geographic Databases. GeoInformatica 4:1, pp. 7-34, 2000.
[10] E. Pitoura and B. Bhargava. Maintaining Consistency of Data in Mobile Distributed Environments. Proceedings of the 15th International Conference on Distributed Computing Systems, pp. 404-413,

Vancouver, Canada, 1995.

[11] E. Pitoura and B. Bhargava. Data Consistency in Intermittently Connected Distributed Systems. IEEE Transactions on Knowledge and Data Engineering (TKDE), vol. 11, no. 6, 1999.

[12] D. Golovin, A. Gupta, B. M. Maggs, F. Oprea and M. K. Reiter. Quorum Placement in Networks: Minimizing Network Congestion. Proceedings of the Twenty-fourth Annual ACM Symposium on Principles of Distributed Computing, Denver, Colorado, USA, pp. 16–25, 2005.

[13] M. Lee and S. Helal. HiCoMo: High Commit Mobile Transactions. Kluwer Academic Publishers Distributed and Parallel Databases, vol. 11, no. 1, 2002.

[14] G. D. Walborn and P. K. Chrysanthis. Transaction Processing in PRO-MOTION. Proceedings of the ACM Symposium on Applied Computing, San Jose, USA, pp. 389-398, 1999.

[15] C. M. Bishop and E. M. Tipping. Bayesian Regression and Classification. Advances in Learning Theory: Methods, Models and Applications, J.A.K. Suykens et al. (Editors), IOS Press, NATO Science Series III: Computer and Systems Sciences, volume 190, 2003.

[16] S. Abiad, R. A. Haraty and N. Mansour. Software Metrics for Small Database Applications. Proceedings of the 15th ACM Symposium on Applied Computing. Como, Italy, pp. 886-870, March 2000.

Table 1. Clustering Model ACID Properties

Model	Atomicity	Consistency	Isolation	Durability
Clustering	MH Disconnected: weak operations are performed and locally committed. MH Connected: strict operations are performed using 2 Phase Commit (2PC) DB Server: reconciliation function commits or rolls back transactions in case of conflict	Intra-cluster consistency: For the two types of data values either weak or strict, a single value for each may exist for each no data my value may have multiple values for a specific type. Inter-cluster consistency: Is maintained by insuring that divergence doesn't exceed the specified degree of inconsistency.	Uses Strict 2PL for concurrency control and introduces 4 lock tables one for each operation type (WR, WW, SR, and SW). Intermediate values are not visible to transactions, but locally committed values are visible to local transactions on a specific MH. Strict versions of data are replicated using a quorum consensus protocol, whereas weak versions are propagating according to the degree of inconsistency.	No guarantees on durability. Dependent on the degree of inconsistency in the cluster.

Table 2. Two-Tier Replication Model ACID Properties

Model	Atomicity	Consistency	Isolation	Durability
Two-tier Replication	MH Disconnected: tentative operations are performed and locally committed. MH Connected: Base operations are performed using an atomic commit protocol DB Server: reconciliation is performed by re-executing tentative transactions as base transactions.	Is maintained through acceptance criteria and commutative tentative transactions.	Uses a 2PL variant for concurrency control. Intermediate values are not visible to transactions. Locally committed values are visible to local transactions on a specific MH. Base versions of data are replicated through a lazy replication scheme. Tentative versions remain local to the MH that generated them.	No guarantees on durability. Dependent on acceptance criteria during re-execution.

Table 3. HiCoMo Model ACID Properties

Model	Atomicity	Consistency	Isolation	Durability
HiCoMo	MH: HiCoMo transactions are locally committed and manipulate only aggregate data. DB Server: reconciliation is performed by re-executing HiCoMo transactions as base transactions, taking into account the predefined error margin. Transformed HiCoMo transactions are aborted if, during re-execution, the error margin is not exceeded.	Is maintained through the generation of aggregate tables, commutative transactions, and predefined error margins.	Uses an optimistic timestamp ordering concurrency control. Intermediate values are not visible to transactions. But locally committed values are visible to local HiCoMo transactions on a specific MH. A convergence scheme maintains consistency among replicated aggregate and base tables.	Data items are committed locally directly after the execution of a HiCoMo transaction. However final commit is dependent on the successful re-execution of HiCoMo transactions as Base transactions.

Table 4. PRO-MOTION Model ACID Properties.

Model	Atomicity	Consistency	Isolation	Durability
PRO-MOTION	MH: Local commit using 2PC DB Server: synchronization operation to reconcile the MH's data store with the permanent data store.	Is maintained through constraints and state information in the compact upon dispatch to the MH.	Uses isolation levels (0-10) applied individually to each generated compact	Data expiry and reconciliation conflicts may rollback locally committed transactions