Machines and Algorithms
http://www.knovell.org/mna

Review Article

It's all about Mobile Databases - Survey

Sheikh Abdul Hannan [1], *

[1] Virtual University, Lahore, 54000, Pakistan

*Corresponding Author: Sheikh Abdul Hannan. Email: ms090400008@vu.edu.pk

Received: 02 November 2022; Revised: 02 December 2022; Accepted: 30 January 2023; Published: 6 March 2023

AID: 002-01-000016

Abstract: As the use of mobile databases is increasing and demanding now a days, so there is need of efficient mobile DBMS. This paper presents useful work done in mobile database systems. Introduction of Mobile DBMS is discussed in this paper. Important characteristics and limitations of any Ubiquitous DMBS have been explored. This survey focuses on various types of mobile DBMS architectures discussed by different authors in different literature. It also addresses different mobile transaction management techniques and models. There is an overview of different concurrency control techniques included, so that data can be managed concurrently. There are different challenges and future concerns in this field and those should be addressed properly.

Keywords: Mobile DBMS; Mobile Transaction; Concurrency Control; Mobile Database Architecture; Ubiquitous DBMS; Mobile Transaction Models;

1. Introduction

Mobile technology and wireless networks are progressing rapidly and a lot of databases driven mobile applications are in the market. People used these mobile applications for ease of their everyday life. Like a manager can manage his resources and information to accomplish the tasks while travelling. A sales representative can update his/her recent sale before leaving the client's place. People can maintain their bank accounts, businesses, social activities and personal data with the help of these applications. Now several companies are developing conventional business-oriented applications as mobile systems to use on handheld devices.

Due to this vast advancement and usage of mobile applications, there is a great need of effective resourceful mobile databases. These databases should handle huge market data and meet the market necessities and challenges efficiently. In most applications, database systems are being used behind; even in the applications like web browsers, antivirus software etc. These mobile database systems are not being used only in these types of applications, these can be useful in small operative gadgets or ubiquitous devices such as smart watches, smart phones, tablets, GPS navigators, sensors, MP3 players etc. As these databases are for devices with mobility options, so these can be termed as Ubiquitous Database Management Systems (UDBMS) [1].

These mobile databases are different in the structure, transaction management, query optimization and other aspects as compared to immobile or traditional databases. The challenges and considerations are also dissimilar and complex for mobile databases. The mobile devices face disconnection frequently while roaming between different cells. The bandwidth of wireless network channels, is also low. On the other hand, these types of devices have limited computation power and small storages. The energy resource like

battery of these devices is also limited; and it may cause for long disconnections [2-3]. So, there is a need of light weight and strong DBMS, which should meet these challenges [4-5].

The rest of the paper is organized as follows. The next section provides the background and related work information. The section after that discussed the architecture of mobile database. Transaction management and Concurrency control techniques will be discussed in the same section. In the next section, some Mobile DBMSs in the market will be discussed. The subsequent section, summarizes and concludes the paper along with future research horizons.

2. Related work

As per study of different work done already, referenced in the end, Badrinath has presented a new architecture for mobile database which addressed the issues like disconnection, concurrency control and replication management etc [6], [7]. An object-oriented design architecture which supports conflict-free and context-aware data and transaction management, is proposed by Steffen [8]. A way to transform a conjunctive projection-selection query into a unified form, is described by Lee [9]. This transformation proposed two cache partitioning schemes to reduce the granularity of a caching unit and described the semantics of a cache fragment. Ziyad adapted Encryption and Decryption algorithms between base station (BS) and mobile host (MH) for secured data transmission [11]. He worked on security management for mobile database transaction. The same work is further carried and explored by Priya [10].

Sheini worked on the algorithm to predict data in mobile databases [12], [17]. As mobile systems can be in disconnected mode while a transaction, so he proposed Memetic algorithm to predict till to connect to the next mobile network. Yendluri enhanced the conventional optimistic concurrency control (OCC) algorithm through the use of invalidation reports [13]. With the help of these invalidation reports, conflicting transactions can be identified timely and terminated before reaching the validation phase.

Imam proposed Heterogeneous Mobile Database Synchronization Model (HMDSM) for data synchronization between Mobile Clients and Server-side database [14]. In this approach, a listener is added to listen any operation performed on either side. IDBSync (Improved Database Synchronization) was produced by Kottursamy [16], in which the control plane of Software Defined Networking (SDN), synchronizes information across the data planes of mobile hosts and server. It uses BaSyM (Batch Level Synchronization Methodology) table to synchronise different data planes of mobile clients.

Malladi applied multiple query optimization algorithms on mobile databases and analyse the comparative results for optimal solution [15]. Issues and research direction have been addressed by Bernard [2].

2.1. Mobile Database Management System

The applications for ubiquitous devices required totally different ways of data access and management as compared to the traditional DBMSs.

2.1.1. Characteristics of Mobile DBMS

Different characteristics of Mobile DBMS has been discussed before. The authors and researchers have reviewed the main characteristics and stated as the basic requirements of any Mobile DBMS's development. Based on these existing surveys [1], [18-19], the important requirements for ubiquitous DBMS are as follows.

- Streamlined DBMS Integration - These databases functionalities should be integrated with in the application infrastructure, so that there will be no need for further configuration or administration. These databases must be incorporated as DLLs in applications and implemented as a single stand-alone DBMS capable of handling many transactions and applications.
- Mobile Database Footprint - As these databases will be developed for ubiquitous devices with limited storage capacity and does not need continuous connection with the server, so these devices download the required small chunk of information from server and uploads the manipulated data

on server. This small chunk of data downloaded and maintained by mobile database, is called footprint of database [20]. It is important to minimize the size of footprint, so the efficient and simple query execution can be achieved.

- Universal Mobile Compatibility - There is a variety of mobile devices with memory, disk, processor and operating system limitations; so, the mobile DBMS must be able to run on all devices and can be synchronized with back-end systems efficiently.
- Componentized DBMS - As these DBMSs must contain support for small footprints, thus, it is preferable to offer only the functionality necessary by the application. This can be achieved with the help of componentization of DBMS. For example, Simple apps do not require a query processor since they just require record-oriented access. So, these types of DBMSs should support component-based development.
- Self-Sufficient Database System - This DBMS in not visible to end users and usually there is no database administrator (DBA) to manage the database operations. This DBMS should be self-manageable, so that it can perform maintenance operations (backups, recovery, indexing, tuning etc.) by itself or through application. This DBMS should be installed automatically once the application is installed and there will be no need to install this DBMS explicitly or separately.
- In-Memory DBMS Techniques - These DBMSs should serve the applications which need high performance on the data, which will be small enough in the main memory. There should be in-memory DBMSs techniques for optimal query processing and indexing for the small main memory data that may never get persisted.
- Portable Mobile Databases - The mobile applications have to be deployed quickly. When you install these applications, the related database should be installed automatically. So, the mobile database should be highly portable and should be ported with the application as a single file. It means copying the database file completes the migration process of application and there should be no need to install database separately.
- Code-Free DBMS - These DBMSs should be portable and this portability may be act as a carrier of virus or other risks, so there should be no executable code stored in the database. These DBMSs should prevent to store any code in the database.
- Efficient Data Synchronization - These DBMS should facilitate efficient synchronization between mobile hosts and back-end server databases. The data may be retrieved from the back-end database and synced back to the server database after alteration.
- Centralized DBMS Control - As mobile DBMS must be self-managed; so, it should be managed remotely also. The centralized remote management is necessary to configure and manage the mobile databases according to company standards.
- Adaptable Query Systems - There should be facility for custom programming interfaces. As there is a variety of query languages and data models, such as relational, XML, Object-oriented, Streams; it is therefore necessary to modify and componentize these DBMSs in order to provide domain-specific programming interfaces and query languages.

2.1.2. Architecture of Mobile Database

Badrinath consider the centralized database and extended it as Mobile Database, such that mobile devices (hoard clients) can also use this database [6-7]. Hoard clients are the ubiquitous devices which can download (hoard) data and reintegrate the manipulated data back to the server. According to him, there is one centralized database, which can be accessible by traditional wired clients and hoard clients simultaneously.

Traditional clients should be connected always to the server to perform any single database related task. If these clients get disconnection, these cannot perform any operation on the data. On the other hand, the hoard clients get data (footprint) from the centralized server as local database. These clients hoard the small

required fragments only, not the entire data. These clients can be disconnected from the server and perform any operation on local hoarded database while disconnected. Once these clients are connected, these behave like traditional clients and have full access over centralized database. These clients reintegrate the data with centralized server; such as update the manipulated data on the server or hoard updates from the server to refresh its local database etc.

Kumar presented a reference architecture for mobile database that a fixed traditional host should be interconnected through a wired network and mobile hosts should be entertained by a mobile switching centre (MSC) [21-22]. Selvarani further categorized the types of mobile database architecture; like Client/Server architecture, Server/Agent/Client architecture and Peer to Peer architecture.

- Client/Server Architecture

There is one centralized database server. The fixed clients and mobile clients communicate with this server for database operations [23-24]. Yang presented Publisher/Subscription framework in a client/server fashion; where the Publisher objects acts as server and Subscriber objects treated as clients [25]. Zhang also described the 3-tier architecture as client/server architecture; this architecture addressed the synchronization or replication of data between server and mobile hosts [26].

As mobile hosts can hoard data in local database, manipulate the data on local site and update the server database. So, it may lead 3 more architectures. If the server is responsible for all computations and clients with limited resources, need to run operations on the server, then this is Thin Client Architecture. If the clients have complete access and can conduct all server functions while disconnected, then it is Full Client Architecture. Once the responsibilities are divided between clients and server, this is Flexible Client Server Architecture.

- Server/Agent/Client Architecture

An agent is added in this design to execute specified activities. It is a computer programmer who can work on either the server or client side. So, there are three layers in this mobile database architecture; Mobile Terminal Layer, Mobile Agent Layer and Server Layer [27].

- Peer to Peer architecture

Clients in this architecture can communicate with one another. These customers can exchange the information; due to this higher degree of availability of data can be achieved. The consistency should be considered and may be compromised if nodes (clients) update any data.

2.1.3. Transaction Management

Mobile Transaction is a set of multiple operations for a single unit of work in a mobile environment. The main characteristics of the mobile transaction are provided as follows [21-22], [28]:

- As there are frequent disconnection and mobility of data and users exist in this mode, so the mobile transactions should be long-lived.
- A mobile transaction can be divided into several sub-operations and can be run in the mobile host or other hosts on the mobile service station (MSS). This shares its state and partial results with other sub-transactions, so the disconnection and mobility issues can be overcome.
- The computations and communication standards, needed by a mobile transaction, should be compatible with mobile service station (MSS).
- The state of transaction, states of accessed data objects and location information should also move along with mobile host during the mobility of mobile host from one cell to another.
- The mobile transaction should manage the data concurrency and recovery. It should handle disconnection and mutual consistency of replicated objects.

A mobile transaction model is quite complex because of the mobile environment limitations; such as low bandwidth, frequent disconnections and limited resources of mobile hosts. Mobile host can be operated

in fully connected mode, disconnected mode or partially connected mode. Another mode is doze mode, in which it saves energy to minimize the connection cost.

Following are the main models for mobile transactions have been proposed.

- Pre-Write Transaction Model - This model supports once the mobile host initiated the transaction and executes it. It means mobile host gets full autonomy. This concept has offered two data versions, pre-write and write, to boost data availability. [29].
- High Commit Mobile (HICOMO) Transaction Model - HICOMO model is also used to handle mobile transactions in fully autonomous mobile hosts. Aggregate table is used in this model [30].
- Moflex Transaction Model - This model is used in the scenario when mobile transaction is initiated by mobile host but fully executed by data server or fixed host. It supports sub transactions or queries which are location dependant [31].
- Pre-Serialization Transaction Model - This model also suitable for the scenario when mobile transaction initiated by mobile host but executed by data server or fixed host. This model allows local site transactions to commit independently of the global transaction. Local resources can be released in a timely fashion [32].
- Multiversion Transaction Model - In this model, mobile transaction may access stale data at fixed host or database server. This model is suitable once the mobile transaction is executed between mobile hosts and database server or fixed host [33].
- Twin Transaction Model - This model satisfies the both modes of operation, connected and disconnected. Resynchronization mechanism makes sure the consistent state of data [34].
- P system-based Mobile Transaction Model (PMTM) - This model is proposed by Zheng [35]. There are two transitions rules (Membrane Rule and Object Rule) are introduced. Membrane rule describes the structure of membrane and object rule describes the transitions.
- Adaptable Mobile Transaction Model - Alvarado presented this approach in which the transaction characteristics of atomicity and isolation are reduced yet conflict serializability is preserved. [36]. It improves the commit possibilities cost effectively.
- Shadow Paging Transaction Model - Hegazy discussed the Mobile-Shadow technique for mobile transactions [37]. This classifies the actions to be performed during validation phase of a transaction.
- Surrogate Object Based Mobile Transaction Model - Ravimaran proposed a methodology to facilitate data caching at surrogate objects in order to retrieve data more quickly and conduct database operations more efficiently [38].
- Connection Fault-Tolerant Model - This architecture decreases resource blocking time at the Server or fixed host. It allows for quick connection recovery and increases the number of committed mobile transactions. Dimovski proposes this concept [39].
- Kangaroo Transaction Model - In this concept, a new layer (data access agent) is added to all base stations on top of the current global transaction manager. It addresses the mobility issue during mobile transactions [40].
- Clustering Transaction Model - In this model, the mobile transaction is divided into two types, strict transaction and week transaction. Strict transactions are performed in connected mode only whereas week transactions execute operations during disconnected mode and synchronize the database once connected [40].
- Two-Tier Transaction Model - Lazy replication mechanism has been used in this model. Tentative transactions are performed in disconnected mode and on reconnect these tentative transactions are converted to base transactions (transactions performed on the master version of data).

2.1.4. Concurrency Control

Concurrency control can be achieved through implementation of two phases locking scheme. This locks all the data and transaction, for a specific time period. Another technique is pessimistic concurrency control, which assumes that the data utilized by one transaction may be needed by others and hence locks the data. Once the transaction committed then the lock should be released.

Optimistic concurrency control technique does not lock the data and same data can be accessed by different transactions concurrently. On commit time conflict should be checked and resolved. Another strategy is to use as low an isolation lever as feasible in order to archive good performance and availability. Shot Duration Locks are also implemented to increase the concurrency among transactions. These locks will be released as soon as possible.

2.2. Mobile DBMSs in market

There are many mobile DBMSs are already in the mobile application's market serving the mass. Some of the famous DBMSs are following:

- SQLite – an open-source embedded lightweight database engine suitable for embedded applications. Size of its footprint is just 500KB. It does not need any configuration or any installation. It also restores automatically after system collapse as it does not need any DBA also. It is plate-form independent and supporting multiple tables and indexes, transactions and ACID property, views, triggers etc.
- SQL Anywhere – an embedded database system and can be deployed with the application installation. It has less than 275KB footprint. It can be called through a simple API. Its transaction log and data are stored in separate files. It provides 128-bit encryption security for data objects.
- IBM DB2 EVERYPLACE – is consisting of shared libraries and used as APIs. Its footprint is 350KB. Compiler and Data Manager Service is used to abstract, parse, optimize and execute query plans.
- ORACLE DATABASE LITE – includes small footprint, 2MB, and used for small business environments. It has data synchronization capabilities and can be installed via self-extracting setup.exe files. It is intended for use in embedded devices and has a compact footprint. There is no involvement of DBA.
- SQL SERVER COMPACT – is a redistribute compact relational database for embedded applications. OLE DB allows applications to connect to the database. It provides ACID properties and 128-bit encryption.

Table 1: Summary of the Key Finding

Aspect	Findings
Architectures & Approaches	- Badrinath's architecture addresses disconnection, concurrency control, and replication management. - Steffen's object-oriented design supports conflict-free, context-aware data, and transaction management. - Lee's query transformation and cache partitioning. - Ziyad and Priya's work on encryption for secure data transmission. - Sheini's algorithm using Memetic algorithm for predicting data in disconnected mobile transactions. - Yendluri's enhancement of optimistic concurrency control. - Imam's Heterogeneous Mobile Database Synchronization Model (HMDSM). - Kottursamy's IDBSync with SDN for synchronization. - Malladi's multiple query optimization algorithms. - Bernard's addressing of issues and research directions.
Characteristics of	- Integration with application infrastructure. - Minimal footprint (small chunk of information downloaded and maintained). - Support for various devices. -

Mobile DBMS	Self-management for maintenance operations. - Component-based development for functionality. - In-memory DBMS techniques for optimal query processing. - Portability with quick deployment. - No executable code stored to prevent risks. - Efficient synchronization between mobile hosts and back-end server databases. - Remote management for configuration according to company standards. - Custom programming interfaces for various query languages and data models.
Architecture of Mobile Database	- Different architectures: Client/Server, Server/Agent/Client, Peer-to-Peer. - Centralized database with hoard clients (Badrinath). - Reference architecture with fixed traditional hosts and mobile hosts (Kumar). - Categorized types: Client/Server, Server/Agent/Client, Peer-to-Peer (Selvarani).
Transaction Management	- Various mobile transaction models with different characteristics and complexities: Pre-Write, HICOMO, Moflex, Pre-Serialization, Multiversion, Twin, P system-based, Adaptable, Shadow Paging, Surrogate Object-Based, Connection Fault-Tolerant, Kangaroo, Clustering, Two-Tier. - Characteristics include long-lived transactions, compatibility with mobile service stations, handling of disconnection and mobility issues, and management of data concurrency and recovery.
Concurrency Control	- Two-phase locking scheme. - Pessimistic concurrency control. - Optimistic concurrency control. - Low isolation levels for performance and availability. - Short Duration Locks for increased concurrency.
Mobile DBMSs in the Market	- SQLite: open-source, lightweight, embedded, platform-independent. - SQL Anywhere: embedded, less than 275KB footprint, 128-bit encryption. - IBM DB2 EVERYPLACE: shared libraries, APIs, 350KB footprint. - ORACLE DATABASE LITE: small footprint (2MB), embedded, intended for embedded devices. - SQL SERVER COMPACT: redistributable compact relational database, supports ACID properties, and 128-bit encryption.

Summary

The referenced works present various approaches and solutions in the domain of mobile database management systems (DBMS). Badrinath introduces a novel architecture addressing disconnection, concurrency control, and replication management issues in mobile databases. Steffen proposes an object-oriented design architecture supporting conflict-free and context-aware data and transaction management. Lee focuses on transforming conjunctive projection-selection queries into a unified form, presenting cache partitioning schemes to optimize caching unit granularity. Ziyad and Priya contribute to security management in mobile database transactions through encryption and decryption algorithms. Sheini develops an algorithm using a Memetic approach to predict data in mobile databases, considering disconnected modes. Yendluri enhances optimistic concurrency control using invalidation reports to identify and terminate conflicting transactions timely. Imam introduces the Heterogeneous Mobile Database Synchronization Model for efficient data synchronization between mobile clients and server-side databases. Kottursamy's IDBSync employs Software Defined Networking (SDN) control planes for information synchronization across mobile hosts and servers. Malladi explores multiple query optimization algorithms for mobile databases, analysing comparative results for optimal solutions. Bernard addresses issues and research directions in the field. The subsequent section details the characteristics, architecture, transaction management models, concurrency control strategies, and existing mobile DBMSs in the market. Key findings include the importance of self-manageability, efficient synchronization, and footprint minimization for ubiquitous DBMSs, as well as diverse transaction management models to address the challenges posed by the mobile environment. Additionally, various architectures, such as Client/Server, Server/Agent/Client, and Peer to Peer, cater to different communication and data exchange scenarios in

mobile databases. The market section lists several mobile DBMSs like SQLite, SQL Anywhere, IBM DB2 EVERYPLACE, ORACLE DATABASE LITE, and SQL SERVER COMPACT, each with unique features catering to specific needs in the mobile application market.

3. Conclusions

In this paper, introduction of mobile databases has been provided. Important characteristics of a mobile DBMS are discussed in detail which leads to a better design of M-DMBS. Architecture of M-DBMSs is different than traditional DBMS. As M-DBMSs can work in disconnected mode also, so it should hoard data locally in form of footprints. After manipulation it should reintegrate the data on server.

Transaction management in mobile environment is also a challenge as mobile devices can be disconnected from the network, moved from once cell to another. Different models to handle the mobile transaction have been discussed and provided the techniques for concurrency control.

There are still many future concerns and research challenges should be addressed. Efficient Data replication and synchronization techniques are needed. Transaction management and concurrency control, is a hot topic for future research. There is a need of efficient light weight DBMS with small footprints. How to get data confidentiality, privacy and security over the wireless network.

References

[1] Whang, Kyu-Young, Il-Yeol Song, Taek-Yoon Kim, and Ki-Hoon Lee. "The ubiquitous DBMS." *ACM SIGMOD Record* 38, no. 4 (2010): 14-22.

[2] Bernard, Guy, Jalel Ben-Othman, Luc Bouganim, Gérôme Canals, Sophie Chabridon, Bruno Defude, Jean Ferrié et al. "Mobile databases: a selection of open issues and research directions." *ACM Sigmod Record* 33, no. 2 (2004): 78-83.

[3] Barbará, Daniel. "Mobile computing and databases-a survey." *IEEE transactions on Knowledge and Data Engineering* 11, no. 1 (1999): 108-117.

[4] Araujo, Mozart, Clauirton A. Siebra, Fabio QB da Silva, and Andre LM Santos. "A lightweight DBMS solution for mobile devices." In *2009 IEEE International Conference on Communications Technology and Applications*, pp. 946-951. IEEE, 2009.

[5] Tavakkoli, F., A. Andalib, A. Shahbahrami, and R. Ebrahimi Atani. "A Comparison of Lightweight Databases in Mobile Systems." (2011).

[6] Badrinath, B. R., and Shirish Hemant Phatak. "A Database Architecture for Handling Mobile Clients1." (1998).

[7] Badrinath, B. R., and Shirish Phatak. *An architecture for mobile databases*. Rutgers University, 1998.

[8] Vaupel, Steffen, Damian Wlochowitz, and Gabriele Taentzer. "A generic architecture supporting context-aware data and transaction management for mobile applications." In *Proceedings of the International Conference on Mobile Software Engineering and Systems*, pp. 111-122. 2016.

[9] Lee, Ken CK, Hong Va Leong, and Antonio Si. "Semantic query caching in a mobile environment." *ACM SIGMOBILE Mobile Computing and Communications Review* 3, no. 2 (1999): 28-36.

[10] Priya, A. "Security Management System for Mobile Database Transaction Model using Encryption and Decryption Algorithm." *International Research Journal of Engineering and Technology* 2, no. 05 (2015).

[11] Abdul-Mehdi, Ziyad T., and Ramlan Mahmod. "Security Management Model for Mobile Databases Transaction Management." *2008 3rd International Conference on Information and Communication Technologies: From Theory to Applications*. IEEE, 2008.

[12] Sheini, Leila Alaei, Khomein Branch Engineer, Hamid Paygozarh, and Mohammad Khalily Dermany. "New Genetic Algorithm to Predict Data In Mobile Databases."

[13] Yendluri, Anand, Wen-Chi Hou, and Chih-Fang Wang. "Improving concurrency control in mobile databases." In *Database Systems for Advanced Applications: 9th International Conference, DASFAA 2004, Jeju Island, Korea, March 17-19, 2003. Proceedings, 9*, pp. 642-655. Springer Berlin Heidelberg, 2004.

[14] Imam, Abdullahi Abubakar. "Data Synchronization Model for Heterogeneous Mobile Device Databases And Server-Side Database." Phd Diss., Universiti Teknologi Petronas, 2015.

[15] Malladi, Rajeswari, and Karen C. Davis. "Applying multiple query optimization in mobile databases." In *36th Annual Hawaii International Conference on System Sciences, 2003. Proceedings of the*, pp. 10-pp. IEEE, 2003.

[16] Kottursamy, Kottilingam, Gunasekaran Raja, Jayashree Padmanabhan, and Vaishnavi Srinivasan. "An improved database synchronization mechanism for mobile data using software-defined networking control." *Computers & Electrical Engineering* 57 (2017): 93-103.

[17] Sheini, Leila Alaei, Khomein Branch Engineer, Hamid Paygozarh, and Mohammad Khalily Dermany. "A Multi-Objective Optimization Algorithm to Predict Information in Mobile Databases."

[18] Nori, Anil. "Mobile and embedded databases." In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of data*, pp. 1175-1177. 2007.

[19] Wankhade, Nitin V., and S. P. Deshpande. "A Review on Databases for Mobile Devices." *International Journal of Electronics, Communication and Soft Computing Science & Engineering (IJECSCSE)* (2015): 276.

[20] Birari, Nilesh. "Small footprint of Database on tablet." PhD diss., Indian Institute of Technology, Bombay, 2013.

[21] Kumar, Vijay. *Mobile database systems*. John Wiley & Sons, 2006.

[22] Selvarani, D. Roselin, and T. N. Ravi. "A survey on data and transaction management in mobile databases." *arXiv preprint arXiv:1211.5418* (2012).

[23] Xia, Yanli, and Abdelsalam Helal. "A dynamic data/currency protocol for mobile database design and reconfiguration." In *Proceedings of the 2003 ACM symposium on Applied computing*, pp. 550-556. 2003.

[24] Chan, Darin, and John F. Roddick. "Summarisation for mobile databases." *Journal of Research and Practice in Information Technology* 37, no. 3 (2005): 267-284.

[25] Li, Yang, Xuejie Zhang, and Yun Gao. "Object-Oriented Data Synchronization for Mobile Database over Mobile Ad-hoc Networks." In *2008 International Symposium on Information Science and Engineering*, vol. 2, pp. 133-138. IEEE, 2008.

[26] Zhang, Yun, Mu Zhang, and Fuling Bian. "A tiered replication model in embedded database based mobile geospatial information service." In *2008 4th International Conference on Wireless Communications, Networking and Mobile Computing*, pp. 1-4. IEEE, 2008.

[27] Li, Jing, and Jianhua Wang. "A new architecture model of mobile database based on agent." In *2009 First International Workshop on Database Technology and Applications*, pp. 341-344. IEEE, 2009.

[28] Mobarek, Adil, Siddig Abdelrhman, Areege Abdel-Mutal, Sara Adam, Nawal Elbadri, and Tarig Mohammed Ahmed. "Transaction processing, techniques in mobile database: An overview." *Int. J. Comput. Sci. Appl* 5 (2015): 1-10.

[29] Madria, Sanjay Kumar, and Bharat Bhargava. "A transaction model to improve data availability in mobile computing." *Distributed and Parallel Databases* 10 (2001): 127-160.

[30] Lee, Minsoo, and Sumi Helal. "Hicomo: High commit mobile transactions." *Distributed and Parallel Databases* 11 (2002): 73-92.

[31] Ku, Kyong-I., and Yoo-Sung Kim. "Moflex transaction model for mobile heterogeneous multidatabase systems." In *Proceedings Tenth International Workshop on Research Issues in Data Engineering. RIDE 2000*, pp. 39-45. IEEE, 2000.

[32] Dirckze, Ravi A., and Le Gruenwald. "A pre-serialization transaction management technique for mobile multidatabases." *Mobile Networks and Applications* 5 (2000): 311-321.

[33] Lam, Kam-Yiu, Guo Hui Li, and Tei-Wei Kuo. "A multi-version data model for executing real-time transactions in a mobile environment." In *Proceedings of the 2nd ACM international workshop on Data engineering for wireless and mobile access*, pp. 90-97. 2001.

[34] Cuce, Simon, Arkady Zaslavsky, Bing Hu, and Jignesh Rambhia. "Maintaining consistency of twin transaction model using mobility-enabled distributed file system environment." In *Proceedings. 13th International Workshop on Database and Expert Systems Applications*, pp. 752-756. IEEE, 2002.

[35] Zheng, Baihua, Wang-Chien Lee, and Dik Lun Lee. "On semantic caching and query scheduling for mobile nearest-neighbor search." *Wireless Networks* 10 (2004): 653-664.

[36] Serrano-Alvarado, P., Roncancio, C., Adiba, M., & Labbé, C. (2005). An Adaptable Mobile Transaction Model. *Computer Systems Science and Engineering (IJCSSE)*, *20*(3), ISSN-0267.

[37] Hegazy, Osman Mohammed, Ali Hamed El Bastawissy, and Romani Farid Ibrahim. "Handling Mobile Transactions with Disconnections Using a Mobile-Shadow." *Cairo University, Egypt* (2008).

[38] Ravimaran, S., and MA Maluk Mohamed. "An improved kangaroo transaction model using surrogate objects for distributed mobile system." In *Proceedings of the 10th ACM International Workshop on Data Engineering for Wireless and Mobile Access*, pp. 42-49. 2011.

[39] Dimovski, Tome, and Pece Mitrevski. "Connection Fault-Tolerant Model for distributed transaction processing in mobile computing environment." In *Proceedings of the ITI 2011, 33rd International Conference on Information Technology Interfaces*, pp. 145-150. IEEE, 2011.

[40] Serrano-Alvarado, Patricia, Claudia Roncancio, and Michel Adiba. "A survey of mobile transactions." *Distributed and Parallel databases* 16, no. 2 (2004): 193-230.