Journal of Theoretical and Applied Information Technology
20th November 2015. Vol.81. No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

DATA SYNCHRONIZATION BETWEEN MOBILE DEVICES AND SERVER-SIDE DATABASES: A SYSTEMATIC LITERATURE REVIEW

[1]ABDULLAHI ABUBAKAR IMAM, [2]SHUIB BASRI, [3]ROHIZA AHMAD
[1,2,3]Department of Computer and Information Science, Universiti Teknologi PETRONAS Malaysia.
Email: [1]aiabubakar3@gmail.com, [2]shuib_basri@petronas.com.my, [3]rohiza_ahmad@petronas.com.my

ABSTRACT

It is progressively important for mobile device databases to achieve additional coordination of diverse computerized operations. To do so, many research works have been conducted to attain more companionable solutions that can be used to synchronize data between the mobile device and the server side databases. The objective of this study is to survey the state of the art in various aspects of mobile devices with respect to data sharing and synchronization between mobile device databases and server-side databases as opposed to the server-to-server synchronizations. To achieve this feat, 5 different electronic databases were used to identify primary studies using the relevant keywords and search terms related to mobile data synchronization classified under journals, conferences, symposiums and book chapters. The study produced interesting results in different areas such as data synchronization, data processing, vendor dependency, data inconsistency and conflicts resolutions where 19 primary studies were selected from the search processes. In our conclusion, mobile data synchronization has been significantly discussed in the domain of databases. In general however, it was discovered that, existing synchronization solutions suffer from a number of limitations such as lack of consistency among data, resolving conflicts, data processing responsibility, vendor dependency, data type, bandwidth utilization and network fluctuation during data transmission. Also the applicability of the existing solutions has not been reported yet.

Keywords: *Data Synchronization, Mobile device, Mobile Databases, DBMS, SLR, Heterogeneous.*

1. INTRODUCTION

The advancement in the area of computing and mobile technology have led to the occurrence of a new computing atmosphere and different categories of small sized mobile devices such as smart phones, Personal Digital Assistants (PDA), Handheld PCs (HPC) and Pocket PCs. Mobile devices have rapidly evolved from simple devices that merely make and receive calls and sends messages to bigger and more sophisticated devices that can also be used as tools to manage and store personal data. As various network technologies and enterprise applications are progressively being associated with such devices, the data management, processing as well as manipulation of enterprise business information can be handy and obtainable using these mobile devices. As a result, more business models that are solely dependent on mobile technologies begin to emerge.

According to [1], [2] and [3], mobile computing offers the chance to users to access data stored in a data repository or stationary database of the mobile devices at anytime and anywhere. Therefore, it is necessary to establish an effective data sharing and synchronization between these devices and the server with maximum consideration of the following mobile devices limitations [4]:

- Restricted bandwidth of wireless networks.
- Limited resources, e.g, memory.
- Mobility (two types of mobility [2]).
  - Terminal (also known as micro) Mobility
  - Network (also known as macro) Mobility
- Disconnections.
- Limited power supply.

Mobile devices are battery dependent and do not have much computing power. In addition, continuous network access is difficult due to the narrow bandwidth [2][5][6]. Therefore, processing large size of data as well as maintain a persistent and uninterrupted connection to the server-side database becomes difficult. For these reasons among others, mobile devices are equipped their

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

own databases so that more stable data processing can be achieved. Various data processing tasks are handled in an offline mode. For mobility support, the work becomes crucial on the network disconnected condition. However, in a disconnected environment, some inconsistencies between the mobile database and the server-side database are inevitable. These discrepancies are sometime detected using *message digest* [5][7][8][9] [10][11] [12] and can also be resolved by synchronization techniques in order to ensure data integrity.

This brings the need for database synchronization between mobile database and server-side database. Based on these, varieties of data synchronization solutions are provided to solve the problem. This paper aims to review all previous related work (with putting much emphasis to 2009-2015 papers) by outlining their strength and weaknesses in order to achieve better synchronization solution for all platforms despite their individual differences. The exploration of this area is necessary as we are heading to the environment where mobile devices are further diversified and their databases are heterogeneous in nature.

The remainder of the paper is organized as follows. Section 2 explains the research method followed in this review, while Section 3 discusses the threats to validity. Section 4 provides the results and the general discussion of the reviewed papers. Section 5 enumerates the research findings and outlines some recommendations for further research, and finally Section 5 which concludes the study.

2. RESEARCH METHOD

According to Kitchenham et al. [13], both research and practice in Software Engineering require evidence based approach which is the synthesis of scientific studies correlated to a question or topic of the research. In addition, it is also being agreed by [14] that, combining empirical studies on a particular topic greatly ensures the chances of reliability. Therefore, a secondary study known as Systematic Literature Review (SLR) is recommended for aggregating evidences.

The aim of Systematic Literature Review (SLR) is to institute a formal process for conducting a literature review, making sure that no biasness and other eventualities such as thorough investigation and analysis are administered [15]. SLRs allow the identification, evaluation and interpretation of all available and relevant information with respect to the topic of research [16].

A tertiary study was conducted in order to evaluate the actual state of Mobile Data Synchronization. This study was planned and executed based on the methodology in [17] and the protocol presented in [13] [14][17] in which the impact of Systematic Reviews in SE is properly evaluated.

2.1 Review Protocol Phases

Basing on Kitchenham et al. [13] review protocols, research questions were defined, search strategy was outlined and resource/materials to be studied were identified and selected. These were later followed by data synthesis to conduct the findings. Fig. 1 shows the SLR review protocol phases.

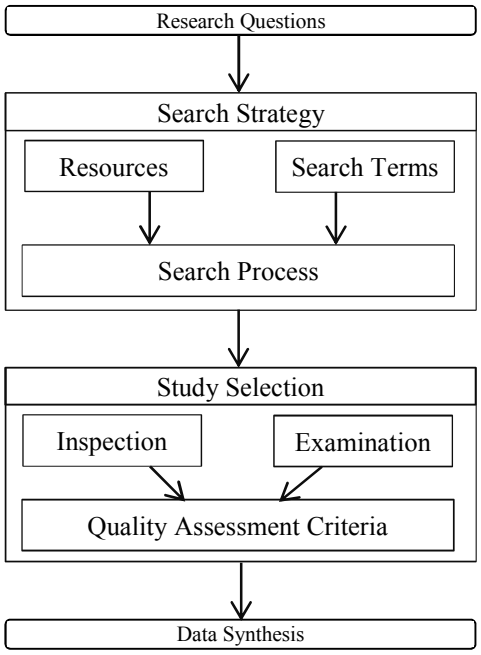

Fig. 1: Review Protocol Phases

2.2 Research Questions

To initiate the study, several research questions were formulated as follows:

RQ1. What are the existing solutions to Data Synchronization with respect to Mobile devices?
RQ2. What are the techniques used to resolve data inconsistencies and records conflicts?
RQ3. What are the methods used to manage Data processing vendor dependency?
RQ4. What are the processes involved in implementing synchronization solutions?

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

2.3 Search Strategy

In this section, search terms, literature resources and search process were elaborated. The following sub-sections describe each process:

2.2.2 Search Strings

Based on [17][18], we constructed our search terms as follows:

i. Terms from the research topic.
ii. Major terms from the research questions.
iii. Derivation of keywords from the relevant books and papers.
iv. Alternative synonyms and spellings from the major terms.
v. Boolean operator "AND" was incorporated as a linker between the major terms.
vi. Boolean operator "OR" was incorporated to join the alternative synonyms and terms. Fig. 2 shows sample of our research string.

(("Mobile Database synchronization" OR "mobile data synchronization" OR "mobile data sharing" OR "distributed data synchronization" OR "mobile data exchange" OR "mobile database communication" OR "mobile contacts synchronization" OR "mobile device databases") AND ("algorithm" OR "model" OR "message digest" OR "systematic review" OR "systematic literature review" OR "cloud" OR "global schema" OR "survey" OR "tertiary study"))

Fig. 2: Sample search string

2.2.3 Literature Resources

In this research, we used comprehensive and detailed electronic libraries such as IEEE Xplore, Science Direct, ACM, Springer, Web of Science, and Google Scholar to search for the relevant materials. Out of these libraries we managed to retrieve journals papers, conference proceedings, books chapters as well as symposiums.

2.2.4 Search Process

In SLR approach, it becomes mandatory for a complete systematic literature review to undergo a comprehensive search of all related sources about the topic of discussion[19].

For that, we systematically searched through the databases and scrutinized the results at different levels as illustrated in Fig. 3 below:

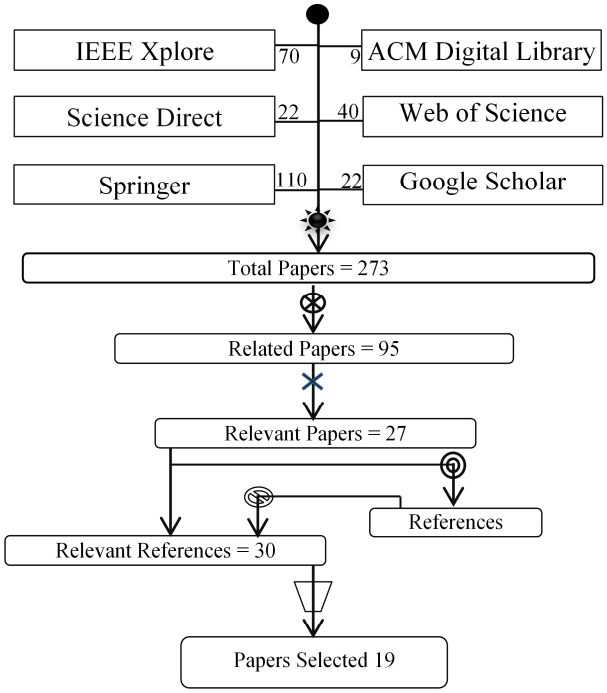

Fig. 3: Process of Search and Selection

In the figure there are several symbols used on the arrows. The symbols can be interpreted as follows:

- Marks the starting point of the search process
- Duplicate papers removal process
- Scrutiny at title level
- Scrutiny at abstract and conclusion level
- Derive references from relevant papers.
- Additional relevant papers search process
- Application of Assessment Criteria

2.4 The Study Selection

Once the process of search is completed, a set of rules are applied for selection. The publications were selected for review if they were:

- ✓ Tackling any of the research key words.
- ✓ Tackling any of the research questions or attempt to describe its nature.
- ✓ Published in, or submitted to, a conference or journal or were technical reports or book chapters.
- ✓ Written in English.
- ✓ Related to topics such as evaluating installed systems and applications, IT services, software applications and IT operations, open source software development, teaching and education.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

On the contrary, the publications were excluded if they were:

- ✓ Papers discussing completely different area from the research topic.
- ✓ Masters and PhD studies which were not published in refereed conferences or journals.
- ✓ Informal literature surveys (no defined search questions, no search process, no defined data extraction or data analysis process)
- ✓ Papers with no answer relevant to any of the questions in the research questions.

2.5 Quality Assessment

In this section we adopted the same set of criteria defined by the Centre for Reviews and Disseminations (CDR) Database of Abstracts of Reviews of Effects (DARE), of the York University [20]. This criteria was also adopted by [13][14] in their SLR.

The above mentioned criteria are governed by the following quality assessment questions:

Q1. Are the objectives of the research clear?
Q2. Is the proposed algorithm/model/technique clearly described?
Q3. Is the experimental setup appropriately designed?
Q4. Were there enough data sets or adequate case study for the experiment?
Q5. Does the academia or industrial community benefited from the research outcome?

We scored the questions by using weighting and scoring technique proposed by Kitchenham et al. [13] so as to obtain relevant studies capable of addressing our research questions. Each question (listed above) has three options: "Y", "P" and "N" which stands for 1, 0.5 and 0 respectively. As a result, the value of each question for a particular study are computed and summed to have the average score which determines whether to include the study or not. The actual pass marks for each study is 50% which is 2.5points (since maximum score possible will be 5). Consequently, in the process of attaining the maximum credibility, completeness and relevance of the selected studies, 57 papers were excluded and we were left with 19 papers. Table 1 depicts the quality scores of the selected study.

3. THREAT TO VALIDITY

According to [21], "there may be at least as many threats to validity as there are sources of validity evidence. Any factors that interfere with the meaningful interpretation of assessment data are a threat to validity". However, in this research we only consider the major sources of validity threats such as Construct Under-representation (CU) and Construct-Irrelevant Variance (CIV) [21].

A. Construct under-representation (CU)

Construct under-representation refers to the undersampling or biased sampling of the content domain by the assessment instrument or research papers. Below are the checklists that need to be fulfilled.

- ✓ The previous study was enough to sample the domain adequately.
- ✓ The domain is either directly or indirectly represented in the retrieved papers and none of the reported papers was in any way bias to the idea, instead they always encourage further researches as summarized in **Error! Reference source not found.**
- ✓ The journal/conference proceedings and books though not too many but they are found to be related to the domain and helpful.

B. Construct-irrelevant variance (CIV)

On the other hand, construct-irrelevant variance refers to systematic error (rather than random error) introduced into the assessment data by variables unrelated to the construct being measured [21]. For CIV, the following are the checklists:

- ✓ The papers almost have the same approach and format related to the proposed domain.
- ✓ The papers are highly secured because almost all of them were published by IEEE, Science Direct, ACM, Springer, Web of Science, and Google Scholar which are all international journals.
- ✓ Score method: Actually our score method is based on how a paper is related to the domain, methodology, application, impact of the research and the percentage at which the problem was tackled.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

- ✓ Teaching: We learnt a lot from the previously written papers which led us having this SLR prepared properly.

In our study, we perceived that validity is related to obtaining the same results after the possible replication of our experiment. It should however be noted that, Kitchenham Charters [17] proposed procedure was adopted in this research, which systematized data selection and extraction and also possesses well defined stages. On the topic of validity, we have the following limitations:

a) The entire process of extracting data was conducted by a single researcher. Although other authors of this paper as well as some experienced researchers reviewed the paper, some eventualities such as biases could have been included in the process.
b) Our search did not go deep in the technical aspects of the selected papers, thereby providing the possibility of missing some relevant information that could have a positive impact in the final outcome.

Readers should bear in mind that, as stated by [22], SLR is limited by: sources and terminologies used in the search, and the search date. Consequently, there is every possibility to add more papers when replicating this study in the future. Our final outputs of this research are limited by the advancement of Mobile data Synchronization area and also by the preceding features.

4. RESULTS AND DISCUSSION

4.1 Overview of the Selected Study

The sub-section presents the results of the search process and study selection which have been conducted.

Table 1 shows the quality assessment score given to 19 research materials. For example, the first row says that for paper [1], the objectives of the research was clear and related, the description of the algorithm/model was clearly articulated, the experimental set up was properly designed, the data sets or case study was not enough, and it was not for academic purposes nor benefited any industry. Hence the overall score of the paper was 3.5.

It should be noted that, "Y = 1", "P = 0.5", "N = 0" (refer to the quality assessment in sub-section 2.5 for detail explanation)

Table 1: Quality Evaluation Score Results

PID	QA1	QA2	QA3	QA4	QA5	Total
S[1]	Y	Y	Y	P	N	3.5
S[2]	Y	P	P	P	N	2.5
S[3]	Y	Y	Y	Y	N	4
S[5]	Y	Y	P	P	Y	3
S[7]	Y	Y	Y	P	Y	3.5
S[8]	Y	P	P	P	N	2.5
S[9]	Y	Y	Y	P	N	3.5
S[10]	Y	Y	Y	Y	N	4
S[11]	Y	Y	Y	P	Y	4.5
S[12]	Y	Y	Y	P	Y	4.5
S[23]	Y	Y	Y	Y	N	4
S[24]	Y	Y	P	P	N	3
S[25]	Y	Y	Y	P	N	3.5
S[26]	Y	Y	Y	P	N	3.5
S[27]	Y	Y	Y	Y	N	4
S[28]	Y	Y	Y	Y	N	4
S[29]	Y	P	Y	P	N	3
S[30]	Y	P	P	P	Y	2.5

The graph below categorized the selected papers per year of publication as it can be seen that 2002 has the highest number of publication while 2003, 2005 and 2009 have the least number of publications.

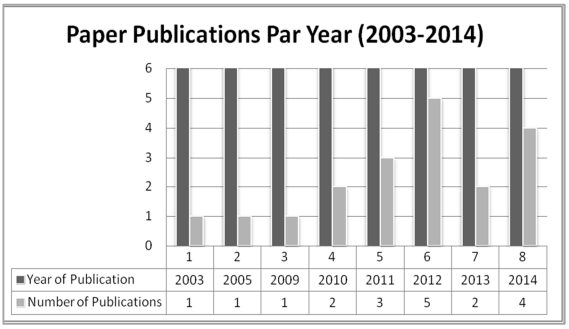

Fig. 4: Number of papers per year of publications

Using the above results, the papers which scored more or equal to 2.5 were selected for review, and Table ***2*** shows some of the results from the review done. Basically three aspects were reviewed which are the solution proposed, the strength and the weaknesses of each of the selected paper.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 2: Meta-Analysis of some of the selected papers

Author & Year	Solution	Strength	Weaknesses
(Domingos, Simões, Pereira, Silva, & Marcelino, 2014)	Synchronization Model	No Standard SQL queries	RDBMS Only. Poor extensibility
(Zaia, Messias, Eduardo, & Olivete, 2014)	MySQLite middleware		Based on Timestamp, Trigger
(Sethia, Mehta, Chodhary, Bhatt, & Bhatnagar, 2014)	Synchronization Algorithm	Consistency improved	Based on Timestamp
(Gopta, Kumar, & Loothra, 2014)	Google Account	Personal goggle accounts	Phone contacts only
(Alhaj, Taha, & Alim, 2013)	Algorithm for Wireless Communication	Wireless based	RDBMS Only
(Balakumar & Sakthidevi, 2012)	Synchronization Algorithm Using MD	Uses standard SQL queries only	Difficult Access. RDBMS Only.
(Sedivy, Barina, MOrozan, & Sandu, 2012)	Cloud Synchronization	Uses cloud as the storage area.	Based on Timestamp. Fixed schemas.
(Choi, Cho, Park, Moon, & Baik, 2010)	SAMD Algorithm Using MD	Based on only standard (ISO) SQL queries	Difficult Access. RDBMS Only.
(Segu, 2011)	Contacts Synchronization Algorithm	Conflict Resolution Strategies	• Contacts only. • User Intervention. • Several external support required
(Ahluwalia, Gupta, Gangopadhyay, & Mcallister, 2010)	Target-Based Synchronization Algorithm	• Interoperability • Scalability.	• Hash collision. • Limited hash capacity.

4.2 Detail Review on Selected Papers

4.2.1 Data synchronization *(RQ1)*

Data Synchronization can be defined as record exchange between two different databases or coherently keeping replicated copies of a data-set [23]. In the context of mobile applications, it is the system that establishes the movement of data between the mobile device and the server-side databases [25]. Database synchronization can be a one-way or a two-way program, and can also be real time or periodic mode, namely *Synchronous and Asynchronous* [31]. *Asynchronous* is periodic in nature and happens to be most advantageous for the disconnected environment where clients are allowed to disconnect from the network and still not miss out any item when reconnected. This technique is the most effective way for bandwidth utilization.

In the area of communication such process of exchanging data is characterized as an inherent asymmetry [32]: where downstream direction (servers-to-clients) is considered to be much greater on bandwidth utilization than upstream direction (clients-to-servers). Since there is usually inability for data transmission at a very high speed (such capacity is sometime not even available), this model was found to fit mobile system excellently. In addition, these same devices can at a very high rate receive data.

Fig. 5 shows a typical synchronization model in the *Database Management System* (*DBMS*) works with a number of clients connected to a

Journal of Theoretical and Applied Information Technology
20th November 2015. Vol.81. No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

server via a network, in which there is usually a server and mobile devices that function as standard customers.

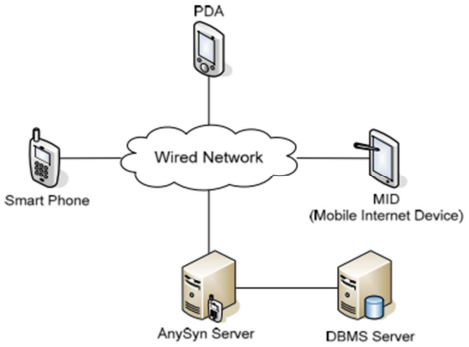

Fig. 5: Typical Framework of Network Synchronization [11] [12]

Based on the layout in Fig. 5 above, [11] and [12] introduced a synchronization algorithms based on secured message digest (SAMD) as a means of solving the synchronization problems. The framework was used to depict synchronization process in a mobile business environment. It consists of multiple mobile devices with internal mobile databases, a database on the server-side, and also a synchronization server (AnySyn).

The synchronization server (AnySyn) is sited between the server and the mobile databases to enable data synchronization as well as manage some essential information that may be used for an effective synchronization. Connection pool is used to minimize the server-side access load on AnySyn server where policies for synchronization are established. For mobile device to perform synchronization, individual toolkits are assigned to each device to access the AnySyn server via a wired network. For this type architecture, we foresee that it may not work perfectly with mobile devices since mobility is one of the characteristics of mobile devices and the users store their personal data on the move.

Some data synchronization solutions adopted the above technique in solving the issues of data exchange between mobile device and server-side databases [5][7]. However, studies such as [9] proposed slightly different approach by establishing wireless connection between the mobile device and the middle tier (sync server). With this, the synch server can be accessed by mobile device via wireless connection. A scenario is shown in Fig. 6.

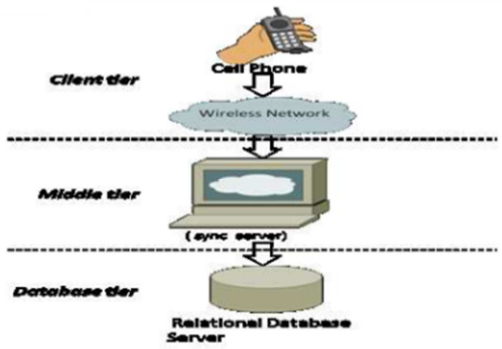

Fig. 6: SWAMD Layout [9]

In the above architecture, all necessary data are located on the server that hosts the relational database. On the other hand, the data on the server is duplicated on mobile databases and finally the sync server comes in between to do the synchronization based on the predefined algorithm (SWAMD). This architecture is good considering the fact that mobile devices cannot maintain connection to the server and require databases for offline storage. The data stored offline, can be synchronized when the network is back in operation.

In [25], an API was introduced to primarily restore the data after a damage. The API is considered general enough to be accessible from mobile phone platform and desktop computers. The API supports all systems that use Google App Engine (GAE) as back-end.

Although the solution is cloud based, it is based on the architecture that is similar to those earlier discussed. However, MCSync is able to synchronize data between web application that resides on GAE and mobile platform. In this approach, there are two databases, one on the client side and another one on the server-side which is considered to have the "correct" value of the database records. So the mobile client can alter the records on the local copy only based on the aforementioned approach as depicted in Fig. 7.

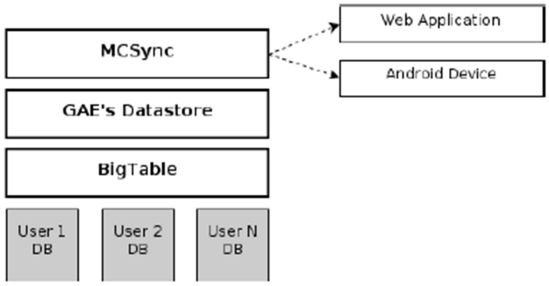

Fig. 7: MCSynch Layout [25]

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

In [23], only two components are used unlike the previously discussed studies where three components are adopted. The two designated components connect and share information using the technology known as JSON (see Fig. 8). The process assumes that for an effective implementation of the algorithms, all devices have their respective times. In this regard, a standard UNIX timestamp is used on either side. During synchronization, these two timestamps are compared, anyone that has the most current time is considered to be the updated version and it's data will be transferred to the party that requires it.

This solution is very effective because it uses only the devices involved without any middle tier, but the fact that it uses timestamp, it becomes vendor specific which causes a great limitation to the databases that do not support timestamp.

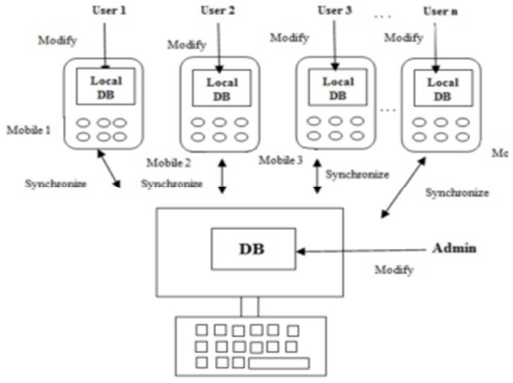

Fig. 8: MRDBMS Synchronization Architecture [23]

Even though the framework introduced by [9], [11] and [12] are similar to the one adopted by [8], the synchronization flow differed significantly. In [8], two methods are outlined, namely One-Way Sync and Two-Way Sync in which the second option is adopted where in each communication there is a controller that controls the operation. The operations are either download from the server or upload to the server. Using such method, we can say bidirectional mode of synchronization is solved although what triggers the event (timestamp) may not be available with other databases.

In [28], the authors proposed a stateful synchronous scheme of mobile devices and also claim that there is no need to compute message digest values of the databases as the server maintains the state of clients. According to the authors, their approach is more effective because whenever a modification is made at the server side of a particular client's data, the client receives an alert or invalidation message to be able to know that there is need for synchronization.

On the contrast, in [29], data synchronization problem is solved using the mobile agents for distributed databases. The communication between the systems is done via mobile agents (see Fig. 9) where aglet is appended with different queries and routes them to the intended destination. In this system although the queries are from different database vendors, they are considered to be the same when retrieving the data. However, this approach is for server-to-server communication that might be hosting different types of databases. It can also be considered helpful mainly for organizations that have more than one server with different database vendors but need their staff to fetch information from all the different databases. As for systems which are of different stage or mobility characteristics, the approach will not be able to be implemented.

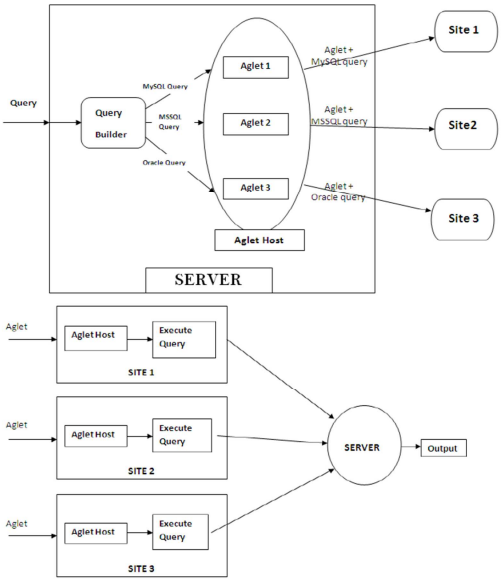

Fig. 9: Server-to-server agent based process [29]

The study in [26] introduced the concept of contact synchronization between the mobile device and the server-side database. The system uses Distributed Version Control System (DVCS) to communicate and exchange information between mobile device and the server, although it is also acknowledged that the DVCS can be used to share data between mobile device without necessarily using the server [26]. However, in [27], a different solution for the same synchronization purpose is provided where the server-side is considered to host OLAP data warehouse. Moreover, the main aim is to reduce

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

tuples retrieval when both the records on either side are equivelent. The system partitions the tuples in both the target's and source tables horizantally and make hashed summaries of each. The two hashes for both partitions are then compared. If they are the same, then identical asumption is colcluded otherwise both the summaries are retrieved and synchronized. Hence make this solution more suitable for server-to-server communication since it involves many proceses of creating images, partitions as well as image comperision which may not be handy for mobile devecies.

4.2.2 Cloud based solutions *(RQ1)*

In Mobile Cloud Computing (MCC) the main activities happen outside of the mobile device such as data processing and data storage. In MCC, things like Data storage and other computing powers are shifted to the mobile cloud as shown in Fig. 10.

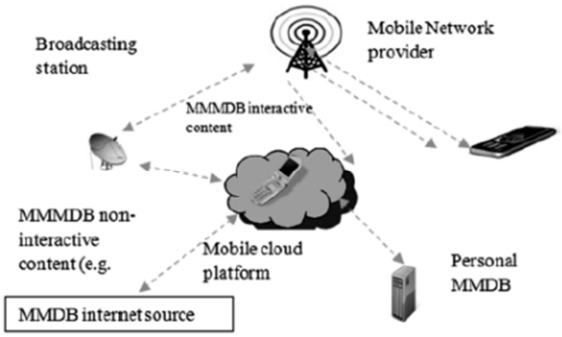

Fig. 10: Mobile Cloud [2]

To relate with data synchronization, some cloud based solutions are provided in which most of the applications use centralized cloud storage. However, in [25], Mobile Cloud Synchronization (MCSync) was introduced in contrast, as a framework to provide the developers with the ability to create and maintain distributed databases. It also allows reliable data storage and synchronization between mobile device and the cloud. In addition to functionalities, some technologies were used such as Android and SQLite. Android provides full relational database capabilities through the SQLite library, without imposing any additional limitations. SQLite, is a relational database management system found to be the ideal system for embedded devices like smartphones.

Using the cloud, applications are transformed into distributed applications that are partially running on the cloud and also partially on the mobile device. This can be considered a drawback [25] due to the dynamic nature in network connectivity and mobile devices; the variations ranges from Wi–Fi to 4G or even to no connectivity at all. Therefore, it becomes necessary to make each application configurable to be able to run locally too, thus; adding more load to the mobile devices even though data integrity is maintained between the Web applications and the mobile. In addition, it also dependents on Google's distributed file system which makes it compulsory to adopt the use of Google app engine server as a server that hosts the databases. In this case, the use of the protocol becomes controlled and constrained.

In [28], the solution is not on the cloud but uses the server that is on cloud in which the data from the client and the server are pushed into its database. The main reason for including additional server in this approach is that, the cloud server uses a static IP while both the client and the server use dynamic IP. To further explain the system, the application on the server is a java socket program that has the capability to detect an active port anytime mobile phone gets connected with GPRS. The server then uses the state of the port and responds to the client's request. The communication between the program and MySQL database is through JDBC driver. To communicate with the cloud, SSH (putty) is used after which FTP can transfer any intended data or file. This approach achieves the ultimate goal in data synchronization. However, it has many stages and many external dependent technologies involved. This contradicts the idea of reducing stages as many as possible to fasten and ease the synchronization process.

A research conducted by ABI [2] predicted that by 2014 the end users that access "mobile cloud" will be close to one billion. But the cloud still faces the following challenges:

1) Network Availability and Intermittency: All services are provided via the Internet [2].

2) Environmental issues: it refers to the spaces in which the communication is established such as delays and connectivity. To be precise they are as follows:

1D Metric Space: Mobile Target Server or Client on the road network. *2D Metric Space:* Mobile Target Server or Client on the plane. *3D Metric Space:* "System involving control of aircraft or submarine". *4D Metric Space:* This involves the

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

tracing of device movement where both devices and environment are mobile such as the sea, in the air, fast moving wild. This areas are known as incessant areas [33].

3) Green Cloud: the core concept here is to be able to perform all computations related to cloud in an economically and energy efficient environment. However, it still requires improvements despite all the proposed architectures in place [33].

4.2.3 Data inconsistency *(RQ2)*

Data inconsistency refers to a state in which the subscribed data in the mobile database and the published data in the server-side database hold different values due to some modifications at either side. Operations such as addition, deletion and modification of data occur on one database independent of the other database; this makes inconsistency inevitable.

Table 3 below shows inconsistencies on every case for a single row as discussed in the research conducted by [11] and [12].

Table 3: Inconsistency Analysis [11] [12]

C	Mobile DB	DB Server	C	Mobile DB	DB Server	C	Mobile DB	DB Server	C	Mobile DB	DB Server
1	UC	UC	5	UC	ADD	9	UC	MOD	13	UC	DEL
2	ADD	UC	6	ADD	ADD	10	ADD	MOD	14	ADD	DEL
3	MOD	UC	7	MOD	ADD	11	MOD	MOD	15	MOD	DEL
4	DEL	UC	8	DEL	ADD	12	DEL	MOD	16	DEL	DEL

(C:Case, UC:Unchanged, ADD:Added, MOD:Modified, DEL:Deleted)

In [11] and [12], the inconsistency check applied was Row Based Inconsistency Check (RBIC). If we critically look at all the 16 cases in Table 3, ADD operation is present in Cases 2, 5, 6, 7, 8, 10 and 14 indicated above, which cannot happen for a single row. These five cases are not considered in this solution. For example, in Case 7 the modified row at the client is different from the added row at the server side; therefore, such scenario cannot be considered an inconsistency. Case 3 and Case 5 also occur independently as Case 7. Likewise for Cases 6, 8, 10 and 14 some interpretation can be made using the synchronization algorithm such as SAMD [11][12], ISAMD [5], and SWAMD [9].

Unlike the above inconsistency checks where the checking of consistency is between the client and the server-side databases at all times, the solution reported in [23] considered inconsistency on the server-side first during insertion where multiple devices try to insert records at the same time. It is also assumed that a new primary key is created only to the considered connected mobile devices. Thus, the insertion process begins as follows:

The server receives request for a new primary key, the server retrieves the highest primary key and it returns the highest + 1 to the requester, the server prerecord the returned primary key, then the device likewise use the received primary key to create a record and finally the required information is synchronized and can be used anytime in the future for the same record.

Moreover, the actual inconsistency between the server and the mobile device is checked using timestamp in this solution. Each client device keeps a log of the *last_sync_timestamp* that is instantiated with 0 and it continues incrementally on any successful synchronization. Anytime the user tempers with the Local Database (LDB), the timestamp gets updated. If any change is made on Local Database LDB for any cell, the timestamp matrix for the cell and *last_sync_timestamp* is appended with the content and sent to the server. The server compares the two timestamps for both ends, if *servre_cell_timestamp* > *client_cell_timestamp,* means the client has the old record, thus; the client will receive the updated version from the server, otherwise the server gets updated with the client value. Similarly, the reverse also holds. Fig. 11 shows the sample of the timestamp.

last_sync_time:

[[[“3”, t1, t1, 0, 0] , [“5”, t2, 0, t2, 0]]
, null , [[“4”, t1, 0, 0, t1, 0, 0]]]

current_time_at_server:

[[[“3”, “#”, “klm”, “@”, “@”],
[“5”, “ghi”, “@”, “#”, “@”]], null,
[[“4”, “#”, “@”, “@”, “#”, “@”, “@”]]]

Fig. 11: time stamp algorithm [23]

In [7], a registry in the device is registered with the new, modified and removed content for the purpose of sending the fundamental data only. Also the entire synchronization solution is considered to be offline by default. Consequently, a variable is created to store the different state of the data such as State A: means the record is available on both side but there is need to remove the one on the

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

client; State B: new record is inserted by the client but not in the server; State C: means the client modified the same copy that is on the server; and State D: means both records are consistent.

In addition, during synchronization, the state of the data is put forward to compare with the record presented for the inconsistency check. It should be noted that the inconsistency check is not from the row data or its ID only, *Actual Message Digest (*hashes business data from the client*)* is compared with *Last Sync Message Digest* (latest hashed business data from the server). Apart from inconsistency check, this method helps in checking the data integrity (data may be lost during transmission) and also lighten the data for faster transmission.

Another way of ensuring data consistency is by adding "transactions". Using this method, atomicity is granted to each database content on any operation [25] which clearly indicate that transactions must be fully applied. Situation where parts of the operations are considered is entirely not welcomed in such approach.

4.2.4 Conflict resolution *(RQ2)*

One of the main challenges of disconnected environment is resolving conflicts [30]. For example, if an upload or download is to be made to/from the server by many clients, alteration of data may occur individually in several ways which makes the data in both sides different. Conflict comes when two or more clients want to synchronize with the server at the same time, i.e., which data is going to be considered the most recent data because all the data cannot be applied correctly into the server, such situation is called conflict [7][11][12][30].

In [12], several synchronization conflicts are considered although AnySyn is adopted. AnySyn detects and resolves possible conflicts conventionally [34]. According to [7], the main type of conflict occurs during synchronization of client devices where different operations must be performed for the same record. Furthermore, the conflicts are classified into the following operations: insert, remove and update of data. Table 4 [11] shows the possibility of creating potential conflicts on each operation that can be performed to a single record on online database server.

To make the process work perfectly, conflicts resolution criteria are predefined by the user (client device) or by the administrator which would be used to resolve each conflict that is detected using the sync manager (AnySyn). The conflict resolution happens when synchronization process begins to keep the copy of the data in database of the remote server. Usually the conflicts occur first then the action of resolving the conflicts is taken afterwards. Although this method is found to be effective, but there is need to enhance the detection mechanism adopted. This could be done by predicting the possibility of the conflicts and applying a certain technique to avoid them to a certain level if not entirely before they occur. In case of those that escape the prediction stage, the second method can be applied to resolve the conflicts using AnySyn.

Generally AnySyn uses automatic conflict resolution mechanisms in DBMSs such as modification date or message digest. A message digest [12][11] is a hash function that generate a fixed length value from a random length message; so that if there is any changes in any bit the entire message digest changes. This message digest is discussed in section 4.2.7 of this paper.

Table 4: Possible Conflicts [11]

C	DB Mobile	DB Server	C	DB Mobile	DB Server	C	DB Mobile	DB Server	C	DB Mobile	DB Server
1	UC	UC	5	UC	ADD	9	UC	MOD	13	UC	DEL
2	ADD	UC	6	ADD	ADD	10	ADD	MOD	14	ADD	DEL
3	MOD	UC	7	MOD	ADD	11	MOD	MOD	15	MOD	DEL
4	DEL	UC	8	DEL	ADD	12	DEL	MOD	16	DEL	DEL

(C: Case, UC: Unchanged, ADD: Added, MOD: Modified, DEL: Deleted)

In [35], the rapid growth of source code in decentralized management system (*DSCM*) can be verified. A system (Git) that is used in high-profile projects is considered in this paper which is a very popular DSCM system. This system is considered to have prioritized speed in keeping history and version tracking. On the subject of conflict, this system uses operations that search the differences between the local repository and the current system after which the edited files are synchronized with the remote server. However, this system is used to solve the conflicts among files, and also uses Version History where the users have to intervene and choose the preferred version [35].

In [26], two getter methods are used. One is to get the contacts stored on mobile and the other is to get the contacts stored on the server. The two (with their differences and the creation or modification dates) are then presented to the user for the appropriate action. Fig. 12 depicts this case:

Journal of Theoretical and Applied Information Technology
20th November 2015. Vol.81. No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Fig. 12: User Resolving Conflict [26]

4.2.5 Data processing *(RQ3)*

In mobile wireless computing environment, databases will be queried by massive low powered machines via wireless communication channels in the nearby future [5]. In addition, mobile devices have limited resources, therefore, minimizing the load on these devices during synchronization become necessary[5][7][11][12].

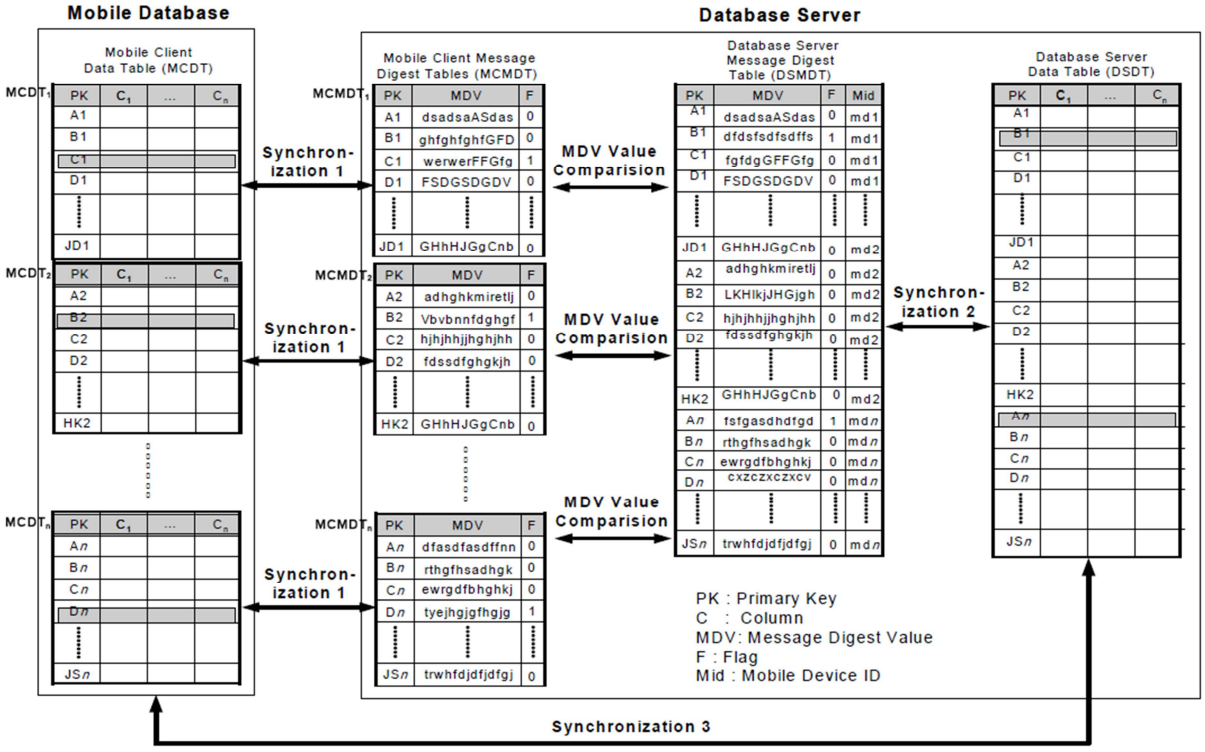

Fig. 13: Data Processing Architecture [11][12]

Correspondingly, to economize storage space of the mobile device, the server-side database hosts all the message digest tables and the actual data table, as depicted in the Fig. 13 above. In this solution, the data required for business are maintained by the server-side database, and the user (mobile database) can download copies of data from the server-side.

Despite the fact that there is additional load caused by network access in the Synchronization 1 in Fig. 13, is considered to be negligible due to the data size of Mobile client data table (MCDT) which is usually smaller than the data from the server. Moreover, the data required for Synchronization 1 (MCDT data) is sent to the server-side database in a single transmission using an SQL query capable of batch processing over a wired network. From this stage, the mobile device becomes relieved and the major part of the operations are performed at the server-side database, thus, making the server to shoulder the data processing responsibility [5][7][8][9] [11][12].

According to [23], data processing can be minimized by adopting read-any/write-any replication scheme. In other words, both the server and client can read and modify the database, thereafter, the records are synchronized but the

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

major computations are performed on the server as well. However, it is believed that data processing can be handled in another way by utilizing the Google App Engine that has many features such as load balancing, caching, and traffic peak [25]. The GAE is located on the server which makes the server to be the main coordinator of the synchronization process. While in [7], a synchronization model was introduced in which its main aim is to minimize the use of resources of the mobile device.

Eventually, the following restrictions are found to be embraced by majority of the researchers to have the main operations handled at the server-side so as to enhance productivity and minimized usage of resources on the mobile device.

(1) Primary key must be available for every database. (2) The data table and the message digest table primary keys have a row identical value. (3) The primary key of the new row that is inserted into the mobile database cannot be identical with another primary key that is newly inserted into the server-side database. According to [11][12], the relational database model involves every table having restrictions (1) and (2). Restriction (3) implies that between the mobile database and the server-side database there is no primary key integrity collision during the synchronization process. Although both ends can have identical primary key values, it is not worthy spending too much time since this problem can be resolved by synchronization policy or even the application-level processing.

4.2.6 Vendor dependency *(RQ3)*

In distributed databases systems and mobile databases, a solution is considered to be vendor specific if it is based on a particular functionality or feature that is not standard across all vendors that may wish to participate (in data exchange) at any given time.

On top of that, databases are considered to be heterogeneous if the local nodes have different types of operating systems (OS) and computers, even if all local databases are of the same DBMS and are based on the same data model [36].

In consideration of the above, several approaches that are vendor specific as well as database category specific such as RDBMS only are described below:

In [5] [9] and [12], the standard SQL query as certified by the ISO (International Organization for Standardization) was adopted in their solutions to enable cross platform synchronization without having any limitation. However, this does not make it fully independent to all vendors because it is applicable only to RDBMS category of databases. Other databases, such as Analytical Databases, Operational Databases, FlatFile, XML etc. are not included in the solution. Whereas in [7], a model was developed to independently establish communication between the mobile devices and the server, the model's independence makes it adoptable by any system or platform. Nevertheless, the solution has some table structure that must be adopted by both parties that wish to communicate. In addition, a given function is used to generate message digest that must be the same for both side to be able to decode the encoded data.

However, many solutions for mobile data synchronization happened to be vendor specific such as the solution in [30] which is based on Microsoft SQL Server and [8] whose solution is solely dependent to MySQLite. Furthermore, others like [23], [25] and [8] voted timestamp database feature as a means of determining the most current state of the data on either side of the databases. So if the timestamp of A is higher than the timestamp of B, A is considered the most up-to-date data and it is synchronized with B. Another database feature that is used by [8] is Trigger which is used to trigger an event in case of any inconsistency that is discovered using the timestamp database feature and thus making all the above not suitable for databases that are fully heterogeneous in nature.

4.2.7 Message digest *(RQ4)*

The message digest concept is defined using the following formula:

h = H (M) where:
h = the message digest
H = the hash function
M = the message of any length

A fixed length message digest h will be generated after subjecting the message M of any length through the formula. If the value of M changes, then it means there is also a corresponding change in the value of h (message digest) [28].

In [11] and [12], a synchronization algorithm based on secured message digest (SAMD) was introduced as a means of solving the

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

synchronization problems. The solution only based on certified ISO (International Organization for Standardization) SQL queries. Furthermore, the design of the algorithm was done bearing in mind the ubiquitous environment; therefore it is considered to be an effective solution for synchronization problems in ubiquitous mobile databases.

In [5], the study proposed Improved Synchronization Algorithms based on Message Digest (ISAMD) as an improved way of giving solutions to synchronization problems. Similarly, it uses only standard SQL queries certified by the ISO. The ISAMD makes the images of the table of the server-side database and the mobile database using a message digest algorithm; then the images and the message digest values are saved in the message digest tables on both sides. The solution algorithm then makes comparison between the two stored images to identify and select the rows needed for synchronization. If the images of both side are different (value of message digest), it means there is change in the duplicated rows which calls for synchronization using ISAMD.

In [9], Synchronization Wireless Algorithm based on Message Digest (SWAMD) is presented as an aid to improve and assist data synchronization between the server-side database and the mobile device database. Additionally, SWAMD embed the concept of wireless connectivity, where the two databases share information wirelessly. SWAMD proposes message digest table that will be used to store images from the server-side database and the mobile device database in order to make comparison between the two images so as to isolate the rows that need synchronization.

Although the solutions discussed above have so many advantageous functionalities such as independence of database vendor, avoiding timestamps, triggers as well as stored procedures, they are all based on the standard SQL queries or function which causes a greater operations on the client because the device need a prior knowledge of the operation of SQL commands [7]. Mobile devices are attributed to be battery dependent, and do not have much computing power. In addition, continuous network access is difficult due to narrow bandwidth [5]. Therefore, the above discussed solutions might be better but not the best for an effective synchronization because they require mobile device to collate data (using temporary table) from various data sources as it prepares the data for synchronization, thereby placing more operations on the mobile device.

4.2.8 Governing Algorithms and Implementation Technologies *(RQ4)*

In this section, we present and discussed the most commonly adopted algorithms and also the technologies or tools used in implementing the existing synchronization solutions.

4.2.8.1 Algorithms

In 2009, Choi et al. [11] introduced a synchronization algorithm that is based on message digest discussed in section 4.2.7 of this paper. The algorithm is discussed as follows (*Note: we will use Fig. 13 and Fig. 14 to explain the algorithm*):

Looking at the algorithm in *Fig. 14,* steps S1~S3 substitute the stage of Synchronization 2 depicted in Fig. 13 where dangling rows are used to identify the DSDT rows for which the application of Steps S1, S2 and S3 occur. This is when the DSMDT and DSDT are FullyOuterJoined.

Steps S4~S6 Synchronization 1 steps are shown in this stage. This is where MCMDT and MCDT synchronization occurs, and Steps S1~S3 are indistinguishable with the basic algorithm. However, FullOuterJoin, as in Steps S1~S3, is not practicable in the internal data table (MCDTT and MDMDT) because it contains different tables of different vendors that are separated physically. To unify the data for MCDT, a temporary table is created at this level to enable Synchronization 1 copy the data to the server-side. The created table and its data are later deleted when Synchronization 1 is completed. This is where the independence to database vendors comes in, using this solution. But it should be noted that the temporary table must be able to either receive or retrieve data from various database vendors that might have different data structure and different connection interfaces to which the temporary table may not conform.

Steps S7~S12 indicate the Synchronization 3 stage. This is where the inconsistencies and the rows that require synchronization are identified using the dangling rows. Also the flags of MDCMDT and DSMDT are verified to be able to synchronize between the DSDT and MCDT. This happens when FullOuterJoin is applied on DSMDT and MCMDT.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The algorithm was found to be efficient and effective in the area of mobile database synchronization. Thus, it is adopted by many subsequent solutions such as [12] in 2010, [5] in 2012, [9] in 2013 respectively. In 2014 two different approaches came in view.

1) [28] introduced the concept of a stateful synchronization where data synchronization between the mobile and the server occur based on the state of the data in either side. It eliminates the issue of message digest. According to the study, using message digest is not required since the server keeps the state of the client's data. So in case of any updates, the server will send the invalidation message to the relevant client. This means some client's information has to be stored in the database of the server to be able to locate which client should be targeted. We believe that this should be avoided, thus, we aim to have a complete independent solution for everyone everywhere and every platform.

2) [7] adopted the use of message digest and a new model with different algorithm was proposed with the aim of minimizing data before transmission. This will reduce the bandwidth consumption and improve the speed of data transfer. Nonetheless, this model is a bit slow as compared to the earlier discussed algorithm with approximately 1.0 seconds in the processing time.

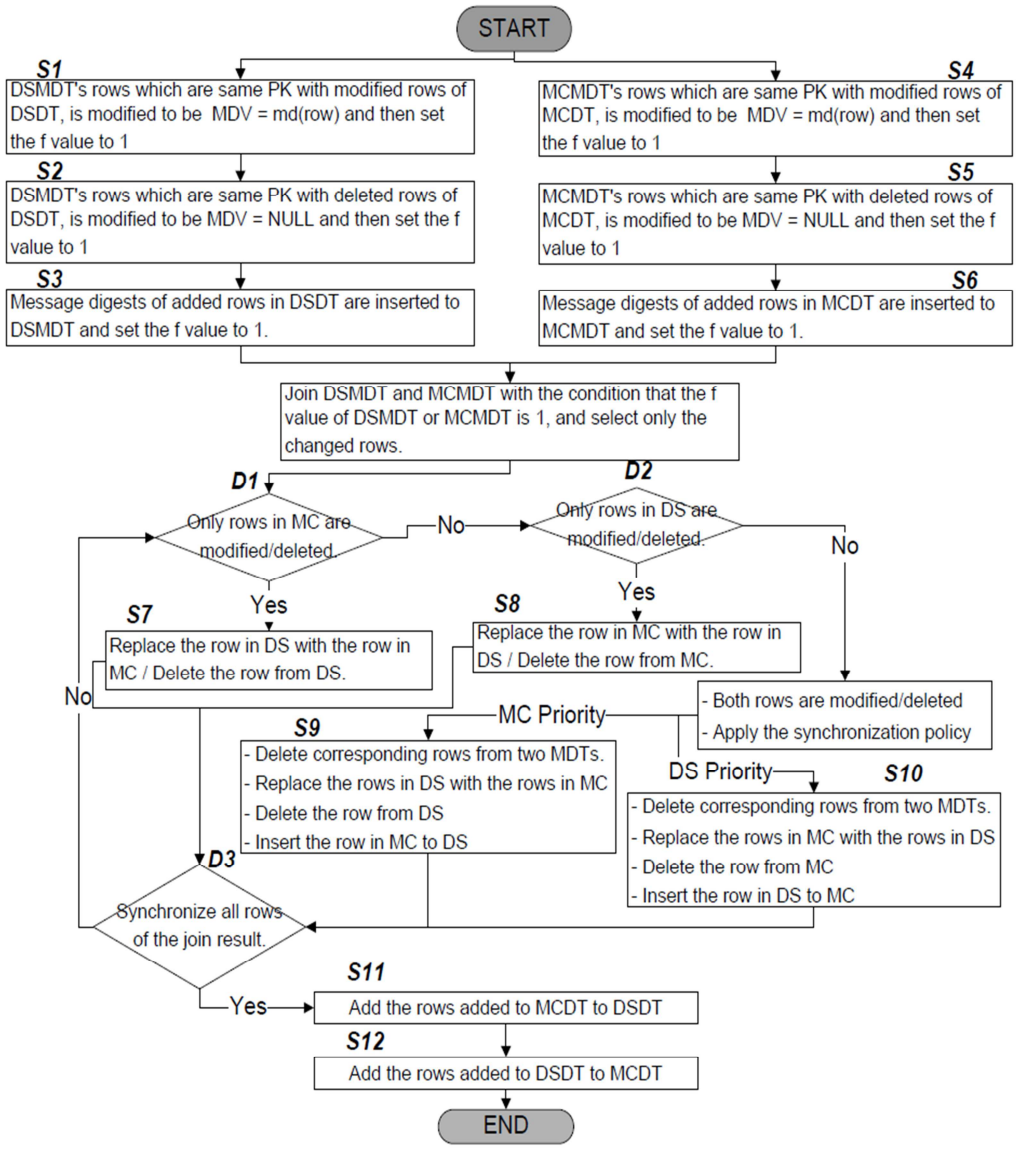

Fig. 14: Synchronization Algorithm [11][12]

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

4.2.8.2 Implementation technologies

The most common technologies adopted to implement the discussed approaches used in solving the problem of data synchronization between mobile device and server-side databases are categorized and discussed as follows:

4.2.8.3 Mobile-device

In mobile device, Android platform was used to develop the mobile based application in Java programming language [5][7][12][24], while [30] considered a different platform which is .Net 2.0 framework. On the other hand, SQLite was chosen as a mobile database technology in [23] and [25], while MySQLite was used in [8]. To add on [25], Cloud to device messaging (C2DM) protocol was considered since the central server is located on cloud. The C2DM is provided by Google. In [28], GPRS technology is embedded, while in [9], Java 2 Micro Edition (J2ME) was used as a platform for solution implementation.

4.2.8.4 Server-side

On the server side, the most used databases are RDBMSs. SQLite is used in [5][7][12][8][25], and Mircrosoft SQL Server in [30]. In addition, MySQL server is considered by [9], [23] and [28]. Again in [25], as there are two servers apart from the client, Google App Engine was used in the second centralized server located on cloud. Similarly in [28], additional server is used that is located on cloud which receives data from the MySQL database on the local server. When all computations are completed the server prepare the data in JSON file format and return it to the client [25].

4.2.8.5 Connections

For an effective data synchronization there must be proper communication. In such, several approaches were used to establish the communication between mobile application and its local database (internal) and also between the mobile device and the server (external). In local connection JDBC (Java Database Connectivity) was selected in [5], [12] and [28] while .Net Connector in [30]. In the external connection, [28] uses SSH (putty) for the connection and ftp for the transfer of data in which the main server is cloudily located. In [25], MCSync REST API was used with a client request like (GET, POST, PUT, DELETE), and formatted JSON file is returned to the client. Whereas in [8], [9] and [23], Hyper Text Transfer Protocol (HTTP) is adopted. In [7], the communication is completely via JSON technology.

4.2.8.6 Hashing functions

Some solutions believed that the data to be transferred should be hashed for many reasons such as security, data transfer speed and data integrity. Therefore, [5] and [12] selected JCE (Java Cryptography Extension) to enable them to easily generate message digest, while JSON file format is adopted in [25] for the same purpose. Furthermore, in [8], mysql2sqlite (Base64) converter is used. Conclusively, [7], [9] and [23] chose to use MD5 technology to produce the message digest in which every single bit must remain in its original position (fixed length). Section 4.8 of this paper explains the concept of message digest.

5. FINDINGS & RECOMMENDATION

Mobile computing has proven to be a fruitful area of concentration for researchers most especially in the field of distributed databases and data management. The fundamental limitations of mobile devices as outlined in section 1 create challenges to the current problems of data management in the area of distributed databases.

Given the amount of work conducted in the last five (5) years in this area, it can be concluded that there are confounding and incredible solutions. However, many problems remain opened for research. Hence need to have better approaches and protocols in the area of data management and synchronization, good bandwidth utilization, better connection interfaces, best conflict resolution, better data processing responsibility assignment and clever algorithms that exploit the vicinity so as to have answers to existing queries. Undoubtedly, a balanced number of research contributions will continue to flow in the future.

Even though more and more incredible researches have been done in this area, many challenges still remain. In this section some of them are pointed:

i. Bandwidth utilization and network fluctuation: Although more sophisticated mobile devices are technologically advanced, some areas mobile network infrastructures are often still under-developed or prone to some unavoidable disturbances such as severe weather, etc. This brings about the issue of

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

poor bandwidth over most networks [37]. Therefore, there is a need to lighten data for easy and fast transfer.

It is also important to determine whether data is 100% or partly delivered to the destination due to possible network breakage at some point. The data transfer starts over again when the network comes back in operation, thus make the process tedious.

ii. Data Inconsistency: More work is needed in studying real cases where the data in mobile database become different from the data in the server-side databases most especially in the distributed network and databases. In solving that, many techniques have been suggested, however, most of them use row based inconsistency check which is cumbersome when there are so many columns in a given row and only one or two columns require update. Therefore, there is need to enhance the inconsistency checking by handling the affected data only.

iii. Conflict Resolution: in this area we can see that the resolving of conflicts happens after the occurrence of the conflicts. In the future two things are needed. 1) To have a solution that will predict the possible occurrence of conflicts and mitigate them before they occur. And 2) to develop a model that will explore all the sources of conflicts and address them in the solution.

iv. Data processing: A key issue is to effectively reduce the data processing such as calculations, manipulations, version comparison etc. done by the mobile application. Therefore, there is a need to migrate the highest percentage of the operations to either the server or the middle tier so that mobile device power can be further utilized, memory can be economized, and unstable network connection can be optimized.

v. Vendor Dependency: Very little has been done in discovering the best ways to generalize an approach such that any mobile device (that is on different platform, versions, operating system, databases etc) can communicate with any server or collection of servers with heterogeneous databases and share information seamlessly.

vi. Implementation Technologies: SQLite and MySQLite are the commonly used technologies on mobile devices while RDBMS category is most considered on the server side. However to make the solutions more heterogeneous, other databases (that use different structure, schema and NoSQL) should be explored.

vii. Prototyping: we have discovered many theoretical studies, some implemented proprietary works, experimented or simulated solutions. However, a full-fledged sample that incorporates the major area of strength and the main ideas is still missing.

viii. Data type: this problem is common yet often overlooked, we can see almost all the solutions are concerned with the structured data while there are other types of data such as images, videos and audios [6] and unstructured or semi structured data that are found to be very vital to the users. Clients may need to keep and share such information as mobile device have the capabilities of taking pictures, recording audio as well as video. Some specific issues here are:
   a. Database object representation
   b. The model and the architecture of Database
   c. Retrieval of data efficiently

6. CONCLUSION

Using the approach of SLR, we have managed to conduct literature study on synchronization between mobile device and server-side databases effectively. Relevant literatures have been chosen for reviews and several aspects have been identified to be either directly or indirectly involved in synchronization.

The aspects are data synchronization between mobile device and server-side databases, cloud based solutions, data inconsistency, conflict resolution strategies, data processing, vendor dependency, message digest and the algorithms used as well as the common tools adopted.

Out of the review, several findings and potential future works have been discussed. As future work, we intend to conduct research on data synchronization, data processing responsibility, data inconsistency, and vendor dependency.

REFRENCES:

[1] A. Stage, “Synchronization and replication in the context of mobile applications,” 2005, pp. 1–16.

[2] A. Khan and K. Ahirwar, “Mobile cloud computing as a future of mobile multimedia database,” vol. 2, no. 1, pp. 219–221, 2011.

[3] A. Stage, “Synchronization and replication in the context of mobile applications,” in *ICFN*

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

'10. Second International Conference on, 2012, p. 98,101.

[4] N. Banivaheb, "Mobile Databases," *Slide Presentation*, 2012. [Online]. Available: http://www.cse.yorku.ca/~jarek/courses/6421/F12/presentations/Mobile-Databases_Presentation.pdf.

[5] V. Balakumar and I. Sakthidevi, "An Efficient Database Synchronization Algorithm for Mobile Devices Based on Secured Message Digest," *2012 Int. Conf. Comput. Electron. Electr. Technol. [ICCEET] Messag.*, pp. 937–942, 2012.

[6] V. Friderikos, "Balancing Transmission and Storage Cost for Reducing Energy Consumption in Mobile Devices," *IEEE 3013*, 2013.

[7] J. Domingos, N. Simões, P. Pereira, C. Silva, and L. Marcelino, "Database Synchronization Model for Mobile Devices," 2014.

[8] G. P. Zaia, C. R. C. Messias, R. G. Eduardo, and C. J. Olivete, "MySQLite Sync : Middleware for stored data synchronization in mobile devices and DBMSs," *2014 XL Lat. Am. Comput. Conf. 2 agente*, pp. 1–7, 2014.

[9] T. A. Alhaj, M. M. Taha, and F. M. Alim, "Synchronization Wireless Algorithm Based on Message Digest (SWAMD) For Mobile Device Database," *2013 Int. Conf. Comput. Electr. Electron. Eng. Synchronization*, pp. 259–262, 2013.

[10] E. Mohan Das and S. Suresh, "A Synchronization Algorithm of Mobile Database by Using SAMD Algorithm," *Int. Conf. Comput. Control Eng. (IEEEC 2012)*, no. 12 & 13, 2012.

[11] M. Choi, E. Cho, D. Park, J. Bae, C. Moon, and D. Baik, "A Synchronization Algorithm of Mobile Database for Ubiquitous Computing," *Fifth Int. Jt. Conf. INC, IMS IDC, NCM 2009.*, pp. pp.416,419, 25–27, 2009.

[12] M. Choi, E. Cho, D. Park, C. Moon, and D. Baik, "A database synchronization algorithm for mobile devices," *IEEE Trans. Consum. Electron.*, vol. 56, no. 2, pp. 392–398, May 2010.

[13] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering – A tertiary study," *Inf. Softw. Technol.*, 2010.

[14] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering – A systematic literature review," *Inf. Softw. Technol.*, 2009.

[15] P. Achimugu, A. Selamat, R. Ibrahim, and M. Naz, "A systematic literature review of software requirements prioritization research," *Inf. Softw. Technol.*, vol. 56, no. 6, pp. 568–585, 2014.

[16] J. Biolchini, P. Mian, A. Natali, T. e Conte, and G. H. Travassos, "Scientific research ontology to support systematic review in software engineering," *Adv. Eng. Informatics*, pp. 133–151, 2007.

[17] B. e Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," *EBSE Tech. Rep. EBSE-2007-01*, 2007.

[18] G. K. Hanssen, D. Šmite, and N. B. Moe, "Signs of Agile Trends in Global Software Engineering Research: A Tertiary Study," in *6th Int. Conf. on Global Software Engineering*, 2011, pp. 17–23.

[19] A. B. Marques, R. Rodrigues, and T. Conte, "Systematic Literature Reviews in Distributed Software Development : A Tertiary Study," 2012.

[20] T. Conte, V. Vaz, J. Massolar, E. Mendes, and G. H. Travassos, "Improving a Web Usability Inspection Technique Using Qualitative and Quantitative Data from na Observational Study," in *XXIII Brazilian Symposium on Software Engineering*, 2009, pp. 227–235.

[21] S. M. Downing and H. M. Thomas, "Validity Threats: Overcoming Interference with Proposed Interpretations of Assessment Data," *Downing. N.p., 30 Jan. 2004*, 2015. [Online]. Available: http://onlinelibrary.wiley.com/doi/10.1046/j.1365-2923.2004.01777.x/full#ss2. [Accessed: 05-Jun-2015].

[22] D. Šmite, C. Wohlin, T. Gorschek, and R. Feldt, "Empirical evidence in global software engineering: a systematic review," *Empir. Softw. Eng.*, 2010.

[23] D. Sethia, S. Mehta, A. Chodhary, K. Bhatt, and S. Bhatnagar, "MRDMS-Mobile Replicated Database Management Synchronization," *2014 Int. Conf. Signal Process. Integr. Networks*, pp. 624–631, 2014.

[24] K. Gopta, R. Kumar, and S. Loothra, "Smartphone security and contact synchronization," *2014 Fourth Int. Conf. Commun. Syst. Netw. Technol. SMARTPHONE?SECURITY?AND?CONTAC*

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

T?SYNCHRONIZATION?, pp. 621–625, 2014.

[25] J. Sedivy, T. Barina, I. MOrozan, and A. Sandu, "MCSync – Distributed , Decentralized Database for Mobile Devices," *IEEE 2012*, pp. 0–5, 2012.

[26] V. Segu, "Contact synchronization for the Android platform," 2011.

[27] M. Ahluwalia, R. Gupta, A. Gangopadhyay, and M. Mcallister, "Target-Based Database Synchronization," pp. 1643–1647, 2010.

[28] B. S. Ramya, S. B. Koduri, and M. Seetha, "A Stateful Database Synchronization Approach for Mobile Devices," no. 3, pp. 316–320, 2012.

[29] N. Gajjam and S. S. Apte, "Mobile Agent based Communication Platform for Heterogeneous Distributed Database," vol. 2, no. 9, pp. 203–207, 2013.

[30] S. A. Ajila and A. Al-asaad, "Mobile Databases - Synchronization & Conflict Resolution Strategies using SQL Server," *IEEE IRI 2011, August 3-5, 2011, Las Vegas, Nevada, USA*, pp. 487–489, 2011.

[31] L. Zhenyu, C. Zhang, and L. Zunfeng, "Optimization of Heterogeneous Databases Data Synchronization in WAN by Virtual Log Compression," *Futur. Networks, 2010. ICFN '10. Second Int. Conf.*, pp. 98–101, Jan. 2010.

[32] D. Barbará, "Mobile Computing and Databases A Survey," vol. 11, no. 1, pp. 108–117, 1999.

[33] L. Gruenwald and F. Olken, "Mobile Database Research: What Is To Be Done?," *DOI=web.mst.edu/ ~cswebdb/Workshop-AFRL/Paper3209559.pdf.* .

[34] Thomas Fanghänel, J. S. Karlsson, and C. Leung, "DB2 Everyplace Database Release 8.1: Architecture and Key Features," *Datenbank- Spektrum*, pp. 1–15, 2003.

[35] C. Bird, P. C. Rigby, E. T. Barr, D. J. Hamilton, D. M. German, and P. Devanbu, "The promises and perils of mining git," *Min. Softw. Repos. 2009. MSR'09. 6th IEEE Int. Work. Conf. (pp. 1-10). IEEE*, 2009.

[36] G. Thomas, G. R. Thompson, C.-W. Chung, E. Barkmeyer, F. Carter, M. Templeton, S. Fox, and B. Hartman, "Heterogeneous Distributed Database Systems for Production Use," *ACM Comput. Surv. - Spec. issue Heterog. databases*, vol. 22, no. 3, pp. 237–266, 1990.

[37] M. Fathy, M. Soryani, A. Z. Zonouz, Asad, and A. Seyrafi, "Some Enhanced Cache Replacement Policies for Reducing Power in Mobile Devices," *Int. Symp. Telecommun.*, no. IST 2008, pp. 230–234, 2008.