# Data Synchronization Model for Heterogeneous Mobile Databases and Server-side Database

[1]Abdullahi Abubakar Imam, [2]Shuib Basri, [3]Rohiza Ahmad, [4]Abdul Rehman Gilal
[1,2,3]Department of Computer and Information Sciences, Universiti Teknologi PETRONAS Malaysia
[1]Computer Science Department, Ahmadu Bello University, Zaria-Nigeria
[4]Department of Computer Science, Sukkur IBA University, Pakistan

***Abstract*—Mobile devices, because they can be used to access corporate information anytime anywhere, have recently received considerable attention, and several research efforts have been tailored towards addressing data synchronization problems. However, the solutions are either vendor specific or homogeneous in nature. This paper proposed Heterogeneous Mobile Database Synchronization Model (HMDSM) to enable all mobile databases (regardless of their individual differences) and participate in any data synchronization process. To accomplish this, an experimental approach (exploratory and confirmatory) was employed. Also existing models and algorithms are classified, protracted and applied. All database peculiar information, such as trigger, timestamp and meta-data are eliminated. A listener is added to listen to any operation performed from either side. To prove its performance, the proposed model underwent rigorous experimentation and testing. $X^2$ test was used to analyze the data generated from the experiment. Results show the feasibility of having an approach which can handle database synchronization between heterogeneous mobile databases and the server. The proposed model does not only prove its generic nature to all mobile databases but also reduces the use of mobile resources; thus suitable for mobile devices with low computing power to proficiently process large amount of data.**

***Keywords*—Heterogeneous databases; data synchronization; mobile databases; mobile devices; NoSQL database; relational databases**

## I. Introduction

Heterogeneity of mobile databases, complexity in mobile applications development [1] as well as the mobile devices themselves has engineered several obstructions in data synchronization. Data Synchronization (DS) can be defined as record exchange between two different databases [2]. It is the system that establishes the movement of data between the mobile device and the server-side databases [3]. On the other hand, A heterogeneous database is an automated (or semi-automated) system that has disparate data model for the local nodes, Operating System, DBMS to present user with a single, unified query interface [4], [5]. Based on this, numerous works have been conducted to address the DS concerns with different techniques. Amongst them are [6], [7] who proposed Synch Algorithm using Message Digest (SAMD) and [8] who introduced a stateful DS, all for the purpose of minimizing the load on the mobile devices. Also, [9] suggested a target based algorithm which always initiate the synchronization process from the target database.

In this paper we consider a variation of the DS problem with slightly different approach from the above. It focuses on the heterogeneity concept where several databases (regardless of their individual differences) connect and exchange data seamlessly. These differences do not stop at only the database versions or vendors but also different DBMS and data model.

At first the approach eliminates the use of any database dependent information such as timestamp, trigger and meta-data. It also pushed the highest percentage of operation (computations) to the server for calculations and conflict resolution, thus relieving the mobile devices. In addition, JSON technology was considered for data packaging and transfer as a flat file which has no bond to any mobile database. Moreover, the synchronization process is always initiated by the mobile device. This is because mobile devices cannot stay connected to the network all the time so, the server cannot know which device is online before engaging on any synchronization process. On the side of starting the synch event, anything in the mobile device can be set to trigger the synchronization event such as on-boot-up, on-button-click, and on-application-start. It is worthy to mention here emphatically and unequivocally that our approach is flexible, customizable and extendable, it is not close-ended solution, rather, it gives a blue print on how to setup a synchronization environment for the heterogeneous mobile databases and server-side database. The approach adopts the use of message digest to encode messages before transmission and decode upon arrival at the destination.

Message Digest (MD) is also called cryptographic hash, hashes or hash function [10]. It takes a message as input and produces a fixed-length output. The output is normally smaller in size than the input (original message) which is generally referred to as message digest, fingerprint or hash value [11].

The rest of the paper is structured as follows. Section II discusses the related works. Section III explains the adopted method. In Section IV, the proposed model is presented and elucidated. The results are discussed in Section V. Finally, Section VI concludes the paper with future focus.

## II. Related Works

As said in the introduction of this paper that, Data Synchronization is a record exchange between two different databases or coherently keeping replicated copies of a data-set [2]. Database synchronization on the other hand, can be a one-way or a two-way process, and can be real time or periodic mode, namely, *Synchronous and Asynchronous* [12]. Based on these, varieties of data synchronization solutions are provided to enable mobile device databases seamlessly communicate with the server database. Some of these solutions are discussed below, starting with factors that negatively affect the synchronization process.

Since synchronization process occur frequently for mobile devices that house variety of unalike databases with dissimilar data-models and also have a number of limitations such as storage space and processing capacity, the factors that influence the processing speed, allow conflicts, as well as prevent solution generalization in terms of database vendors need to be carefully explored and addressed. Therefore, we retrieved and compartmentalized factors from existing works which we are believed to have significant negative impact to an effective synchronization. The factors and their dependencies are illustrated in Fig. 1.

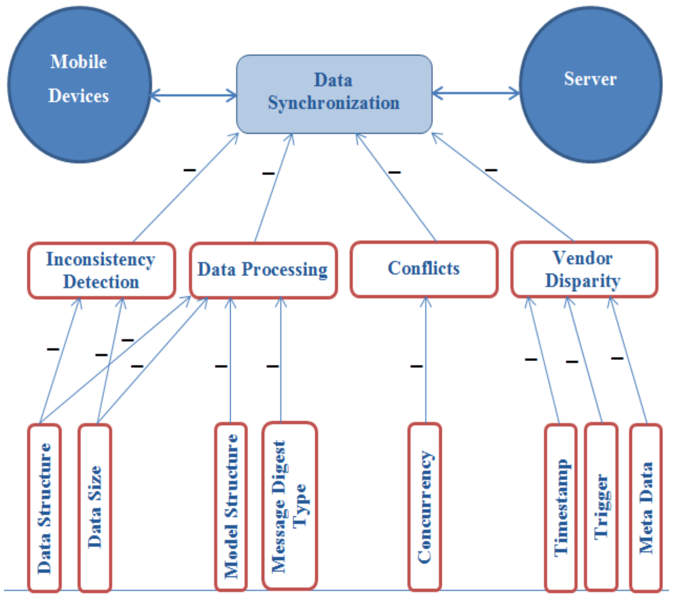

Fig. 1. Factors influencing data synchronization.

There are three layers in the figure above (Fig. 1). Starting from the bottom, the lowermost layer contains the factors that directly influence the main factors shown in the second layer, which in the end persuade or induce data synchronization process as a whole. Looking at the factors above, it is believed that, level two (second layer) has the potential to directly affect, negatively, the data synchronization process. Several approaches have been provided to subside the effect of these factors. Based on the scope of this study, only the factors that affect mobile database heterogeneity will be our focus of this research. The approaches that focus on the same or related concept are painstakingly selected and discussed below in accordance with the main factors (level 2, Fig. 1. above).

Referring to vendor disparity factor (in Fig. 1), 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 database vendors that may wish to participate in data synchronization at any given time [5].

In consideration of the above, several approaches that are vendor specific as well as database category specific such as RDBMS only are described. In [7], [13], and [14], the standard SQL query as certified by the ISO 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 [15], 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 is based on RDBMS only which operate on a particular data model. It also has some table structures that must be adopted by both parties that wish to communicate. In addition, a given function (M=h(H)) 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 [14] which is based on Microsoft SQL Server and [16] whose solution is solely dependent to MySQLite. Furthermore, others like [2], [3], and [16] 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 [16] 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. Other solutions such as [17] and [18], have great synch techniques suitable for server to server communication only.

Having said that it can be concluded that the above solutions are vendor specific or peculiar to RDBMS only. This is because some proposed solutions use vendor dependent functions such as time stamp, trigger or database dependent information like metadata. To be precise, both vendors of the mobile database and the server-side database should be identical or the same entity.

Furthermore, some solutions are dependent to a particular mobile database vendor only. Such solutions are in most cases independent of the server-side database vendor and operate on a separate synchronization server [7], [13]. That is to say, the solution must match the mobile databases for a synchronization to take place. For example, when a programmer is to develop or modify existing mobile application, some particular vendor's libraries for mobile device databases have to be embedded to the solution that resides in the synch server (like AnySyn in Fig. 2) for an effective synchronization.

As a result of these constraints, the flexibility, adaptability as well as extensibility of mobile business systems have been noticeably declined. The problems above need to be tackled as we are heading to the environment where mobile devices will be further diversified and their databases (DBMS) will be heterogeneous in nature [19].

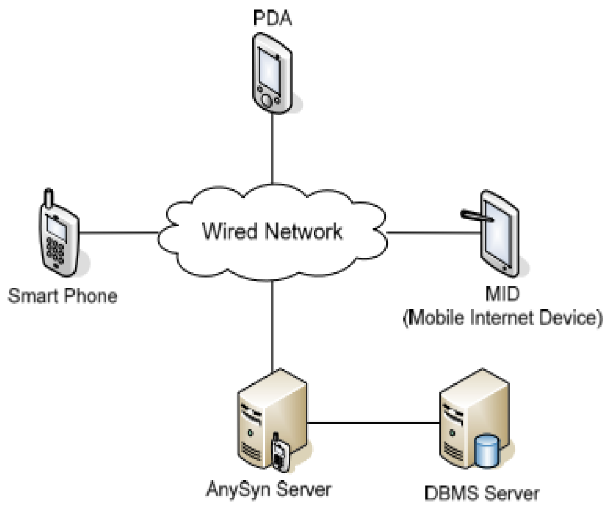

Fig. 2. Model development steps.

## III. Methodology

The purpose of this research is to study the state of the art in the context of mobile databases with respect to data synchronization as well as to propose a generic model that can be adopted by heterogeneous mobile databases. The structural flow and factors that persuade and affect the generality of any synchronization solution need to be painstakingly identified and empirically validated. As a result, it is necessary to adopt a method that allows studies to be carried out in real life context. Out of the five software engineering methods discussed by [20], a case study method was found to be the most suitable for this research.

According to [21], a case study is "an empirical inquiry that investigates a contemporary phenomenon within its real-life context, especially when the boundaries between phenomenon and context are not clearly evident." This type of method explains, comprehensively, how and why certain phenomena occur. To derive new hypotheses and build theories, exploratory case study is adopted as initial investigations of some phenomena while confirmatory case study is used to test existing theories [22]. The clear understanding that confirmatory case study reveals can be useful in reconciling between rival theories.

The results obtained when Case Study Method (CSM) is applied are more valid than when controlled experiment is applied [20]. This is because the variables under study are measured from the real world context.

### A. Invention Method

In our proposed synchronization method, although the communication is bidirectional, we consider mobile devices as the clients and the server as the master. This implies that both *Send-In* and *Send-Out* process are initiated at the client side. This is because the clients cannot stay connected all the time [13], thereby making it difficult for the server to know which among the numerous clients is actually online and ready to receive a package.

Verifications are done at the beginning of the synchronization process to confirm whether there is need for the synchronization and also at the end to verify the successful completion of the synchronization process; we aim to decrease the number of tuples retrieved from the source database that already match their counterpart in the target database.

For each successful transmission, a copy of the hashed data is saved in the temporary repository which can later be used to know whether synchronization is fully or partially completed.

In any synchronization process, the source database horizontally organizes the total order of records in the source database, summarizes the tuples using the hash function, and for the same range of tuples, retrieves the equivalent hash summary from the target database. If the summaries match then we assume both the target and source databases have the same content for the selected range. If the summaries do not match then the same range of records are retrieved from the source database, summarize and send to the target database.

In comparison, this method differs from the existing methods in the following ways:

*1)* Temporary Repository (TR): For each successful transmission a copy of the generated message digest is saved in TR until the process is successfully completed and is removed thereafter.

*2)* Embedding Data Extraction Formula (DEF) that was proposed by [23] into the proposed solution: Network might fluctuate during the synchronization process and the data might have been partly transmitted. To avoid starting the process a fresh, the DEF is used to extract only the records that failed.

*3)* Process Initiator: In some methods, synchronization process begins from the target database. The target database can be either client or master database [9]. Whereas in others such as [2], [14] and [15], the process is initiated by the owner of changes, i.e. the database that is altered would be the initiator of the process. While in our method, considering the fact that mobile devices have no stable connection [24], they are given the responsibility to initiate the synchronization process when they are online. The process can be sending to the server or receiving from the server.

*4)* Data Bank: Keeps the synchronization history. This addresses many issues such as resolving conflicts and comparing whether the source and target databases have identical content before initiating the synchronization process.

### B. Model Development Process

Based on our synchronization procedure, after analyzing the information retrieved from the literature as well as outlining the major strength and weaknesses of each of the existing solutions, the model that aims to mitigate those weaknesses is developed following the three steps as depicted in Fig. 3.

At first, we identified the relevant elements for the proposed model such as entities and attributes from the existing solutions some which strongly guided our selection criteria for the appropriate model structure. Also properties that are unique and common are identified and aggregated.

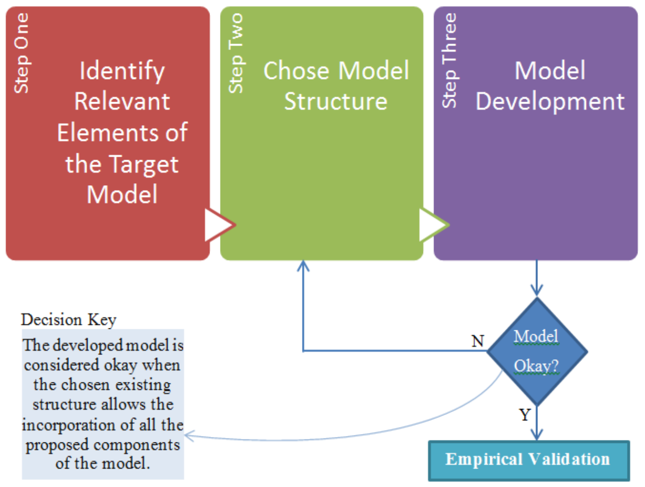
Step One
Identify Relevant Elements of the Target Model
Step Two
Chose Model Structure
Step Three
Model Development
Model Okay?
N
Y
Empirical Validation
Decision Key
The developed model is considered okay when the chosen existing structure allows the incorporation of all the proposed components of the model.

Fig. 3. Model development steps.

Secondly, to choose the appropriate model structure to be adopted, several factors were painstakingly considered such as the most adopted structure, the one that is closely independent to database vendors, the one that considered the utilization of mobile resources like CPU and Memory. Also, date of release is one of our major factors which determine the most appropriate model to adopt. Considering the trend in publications, each proposed model is an upgrade of the existing ones; therefore, the most recent (latest) model would have covered some of the loopholes of its predecessors, thereby providing a strong guide for the development of the proposed model structure.

Thirdly, reconciliation and construction of the proposed model is considered for heterogeneous mobile databases. It should be noted that, the construction of the proposed model that based on the existing models is iterative in nature, therefore, this process involves going back to step two (choose model structure) until we find the structure that best suites our approach or answer our research questions.

## C. *Data Analysis*

To effectively analyze the data generated from both the proposed model and the existing model, two different mobile devices with different specifications were used for the proposed model as well as the existing model. Besides, a single computer was considered as a server to house the central repository. Additionally, the same network was used and at almost the same time, which means, there was no big interval in the network speed for all the trials. Having big interval may result to network inconsistencies which will in the end affect the accuracy of our measurements. The specifications of the devices involved as well as the network itself are as follows.

### *1)* Empirical Validation Tools

In this section, the devices used for validating the proposed model are described. 1) First Mobile Device: ASUS phone brand was utilized with Wi-Fi of 7.10 and battery of 2000mAp. It runs on Android 4.4. 2) Second Mobile Device: iPhone 6 was employed which runs on iOS 10.3.2 operating system with Wi-Fi version of 802.11 and battery of 1810mAh. It also has 32 GB of memory and 1 GB of RAM. 3) Server-side Computer: HP laptop intel® processor, Core™ i7 was deployed which runs on windows 10 with CPU speed of 4.20GHz and 8.00GB of RAM. 4) Network: Ralink wireless network was used with 802.11bgn the network uploads at 3,364,303 and downloads at 58,105,411. Also its speed was at 54.0mbps and 98% of signal quality.

### *2)* Statistical Tool

In this study, we adopted the use of Chi$^2$ test as our analytical tool to analyze the data generated from both the proposed model and the existing model. As for the level of significance, α = 0.05 (5%) was used to indicate a 5% risk of concluding that a difference exists when there is no actual difference. the following section presents the proposed model.

## IV. Proposed Model

This chapter extensively presents the proposed model. Based on the findings and the shortcomings identified from the literature and a review conducted by [25], a generic model is proposed to address some of the untouched areas with respect to data synchronization between mobile device databases and the server-side database.

The ultimate goal of the model is to synchronize server's database with mobile devices heterogeneous databases that are remotely interlinked with utmost consideration on the usage of resources of the mobile device. Primarily, the specific concern is to avoid database vendor dependency approach and provide a solution that is heterogeneous in nature which can be used to synchronize data between mobile databases and server-side database, such that all categories of mobile databases can participate in the synchronization process regardless of their individual peculiarities.

The proposed model comprises of several components which when combined produce a complete working model. In this section, we started by introducing the three (3) architectures, followed by the overall concept and finally elaborate the major components of the architectures one after the other where applicable.

## A. *Architectures*

The architecture section of the proposed model is categorized into three different sections. Each section performs different tasks or responsibilities from the other. First section presents *Send-In Synchronization Architecture*, while section two put forward the *Send-Out Synchronization Architecture*. Finally, Server-side Synchronization Architecture is introduced which illustrates the activities of record bank entity situated on the synchronization participating server.

### *1)* Synchronization Architecture (Send-In)

The process of *Send-In* begins from the mobile devices to avoid information broadcast from the server as the current practice (see Fig. 4 below).

Referring to Fig. 4 above, the mobile devices listen for any changes made on the server, if there is any, the mobile sends request for update along with relevant parameters (authentication, data required). At this point the server takes the charge, thus reducing the burden on the mobile. The server receives request, do the comparison between the record bank and the server private repository. The latest version of the records is then sent to the mobile device as requested.

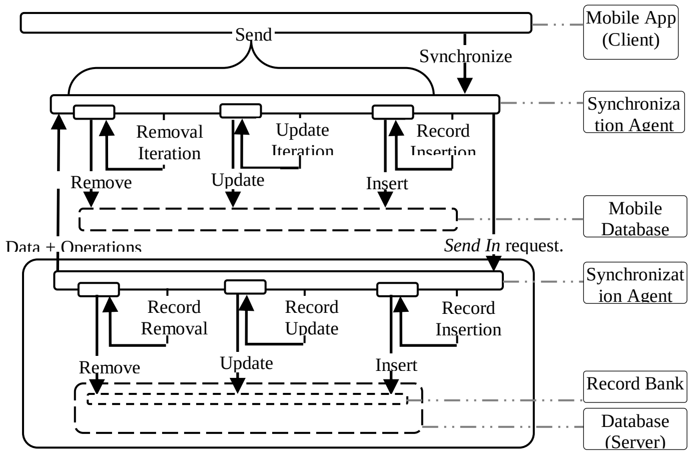

Fig. 4. Send in sequence diagram.

*2)* Synchronization Architecture (Send-Out)

In the *send-out* phase of the architecture, the mobile devices create, modify or delete records and need to notify the server about the changes made. The scenario is depicted in Fig. 5 as follows:

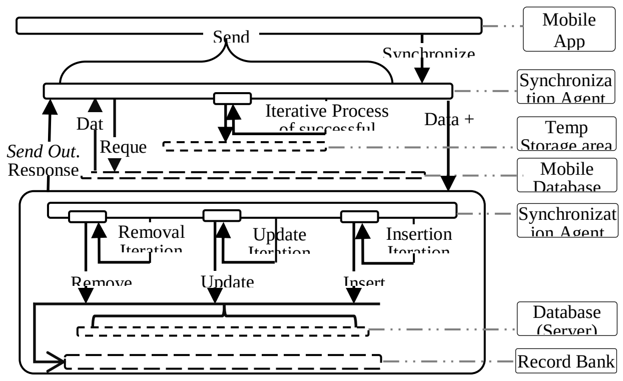

Fig. 5. Send out sequence diagram.

To start the send-out process, the mobile device retrieves the affected data using the formula proposed by [23]**.** The content is then hashed and sent to the server. Each successful transfer of record is cataloged in the mobile device temporary storage area [23] to monitor the synchronization status. The server does the comparison upon receipt of the data and applies appropriate operation.

*3)* Server-side Synchronization Architecture

One server can have multiple clients (mobile devices), each of them sends and receives records from the record bank of the server. For the server to have most up-to-date records in its private repository, there is need to (from time-to-time) synchronize with the record bank since clients communicate with the record bank regularly. The process runs periodically to check for any discrepancies between the data in the records bank and the records of the server. If there are changes, an update operation is applied to the server. Consequently, other clients that require the same updates will eventually see the alert and proceed for synchronization. The process is illustrated in Fig. 6 below.

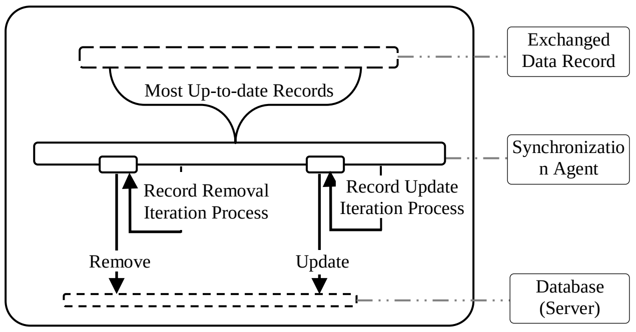

Fig. 6. Record bank to server synchronization.

## B. *Overall Concept*

The overall concept of this model is similar to the existing synchronization solutions; where unlimited client devices connect with the database of the server in order to synchronize data as shown in Fig. 7 below:

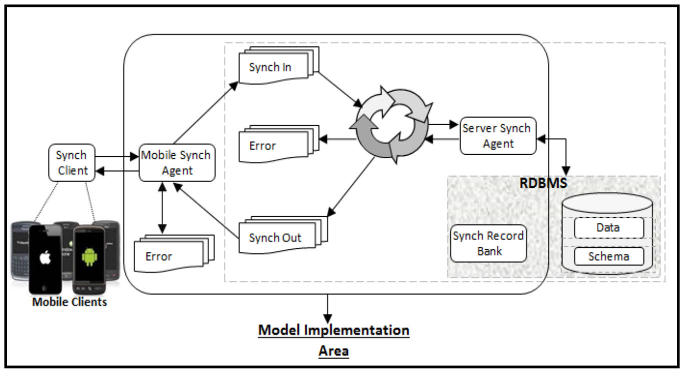

Fig. 7. Overall concept.

With regards to the clients, they are the mobile devices consisting of different types of mobile databases with a light weight storage area, mobile applications that are used to create and manipulate records on the move, and a module where the proposed model is implemented for an effective synchronization. Conversely the server is a computer system that consist of an agent where part of the synchronization model is implemented, Synchronization Records Bank (SRB) where the histories of all synchronization processes are kept regardless of their status (success, failure or removed) in order to track history and to resolve conflict in the future.

## C. *Components of the Architecture*

Regardless of the architectural categorization, each of the earlier discussed architecture cannot work alone. Meaning, they must be merged together to form a complete architecture. Therefore, the merged architecture has the following components:

*1)* Table Structure (Record Bank)

As one of the synchronization staging areas, record bank is a server-side located repository. It keeps the history of data exchanged between the devices and the server. Due to its size and computations (comparisons) involved, it's placed on the server, since the mobile devices have limited storage space. The structure of the repository is as follows:

*a) Record Owner:* It is an attribute that uniquely identifies the actual client that created the record. It is used to differentiate who amongst clients have the most up-to-date record in case one record is being used by many clients.

*b) Private Key:* It is the primary key of a record on the mobile. It is used to differentiate records on the client side.

*c) Public Key:* It is a unique identifier of a record on the server. That is to say, is the primary key of the records on the server. It is used to determine which record is the most recent and also resolve conflicts between the data in record bank and the actual data on the server.

*d) Records Message Digest:* It's the message digest generated by the hash function at run time where the business data is the input. Because it is produced at run time, is considered to be the most recent version of the record.

*e) Flag:* It's an attribute that records the synchronization outcome (success or failure). The flag is 0 when there I no error in the process and 1 otherwise.

*f) Active:* This is where the status of a record is stored. It records 0 if a given entry is no longer in use (removed) or 1 if it's still active. Note that, an entry is not completely deleted, instead, is archived for future reference.

*2)* Data Extraction Formula

Data Extraction Formula (DEF), is a fomula prposed by [23] for the purpose of extracting the records that only matter for synchronization. This formula does the comperison between records and retrive only the affected records which will be used as an input of the following hashing formula.

*3)* Message Digest Formula

Message digest formula is a formula that is used to compute and produce message digest for transmission to the target database.

$$h = H(M)$$

The input of this formula is the output of the DEF explained above. But for the DEF to be able to extract data correctly it requires the following storage space on the mobile device.

*4)* Temporary Storage Area (Client)

Temporary storage area is part of the DFD explained above. It save any successfully transmitted record so as to ease the process of locating a starting point for the DEF. it is also explained in [23].

### *D. Synchronization Proceedure*

In this section, we explain and demonstrate the procedure that our proposed model follows to synchronize data between one database to another. Each of the following figures (Fig. 8, 9, 10 and 11) depicts a particular task that might be assigned to it during the synchronization process. The first (Fig. 8) is the main procedure that, at some point, branches to link to its sub procedures for a separate responsibility.

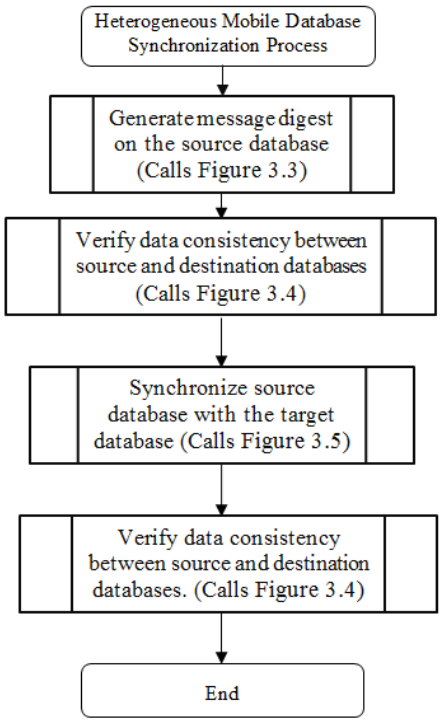
Heterogeneous Mobile Database Synchronization Process
Generate message digest on the source database (Calls Figure 3.3)
Verify data consistency between source and destination databases (Calls Figure 3.4)
Synchronize source database with the target database (Calls Figure 3.5)
Verify data consistency between source and destination databases. (Calls Figure 3.4)
End

Fig. 8. Data synchronization using HMDS model.

There are four significant procedures involved in the synchronization process which starts with generation of message digest, verification of the inconsistencies between the source and the target databases, perform synchronization, and finally verify the consistencies between the two databases. At the outset, message digest generation is explained.

*1)* Message Digest Generation

The process of generating the message digest is the same for both the source and the target databases. However, to minimize the burden on the mobile devices and also keep the history of all performed synchronizations, a hashed copy of each dataset is kept in the record bank of the server after any successful synchronization. The saved hash of any completed synchronization can be later used to resolve conflicts between two or more different clients that are meant to manipulate the same dataset. Please refer to Fig. 9 below:

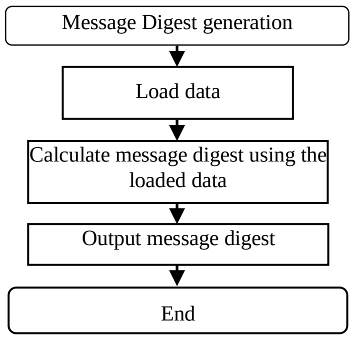
Message Digest generation
Load data
Calculate message digest using the loaded data
Output message digest
End

Fig. 9. Message digest generation.

In this phase, the first activity is to load the data that is to be hashed, after that, a hashing formula is applied to the loaded data which calculates the message digest, and finally the computed message digest is produced for the first and final verifications as well as synchronization process.

*2)* Verification of inconsistencies between the target and the source databases

This is the second procedure which the proposed model follows to verify whether the records of both the source and the target databases have identical values. This process confirms if the data to be synchronized is not available in the target databases. The figure below depicts the verification process:

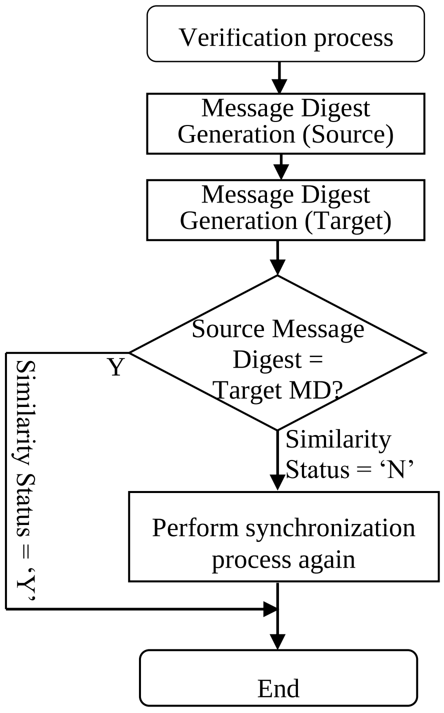
Verification process
Message Digest Generation (Source)
Message Digest Generation (Target)
Source Message Digest = Target MD?
Y
Similarity Status = 'Y'
Similarity Status = 'N'
Perform synchronization process again
End

Fig. 10. Verification process.

When the verification process starts, a cryptographic representation of the data in both databases (source and target) are produced which are further compared to see whether the two defined data ranges are different. If they are the same, the process ends there, otherwise, the synchronization is performed to fill the identified missing information in the target databases as explained in the next section below.

*3)* Synchronization Process

In this phase, after all necessary verifications have been made and confirmed that, there is need for the synchronization to take place, the following process is called to administer the changes accordingly.

The process begins with comparing the two generated hashes (the source and the target) if the verification was not called in the main procedure. This might be possible when there is more data to be synchronized immediately after the first assignment. After the comparison, if the records are the same, it calls for more data, otherwise the synchronization process continues.

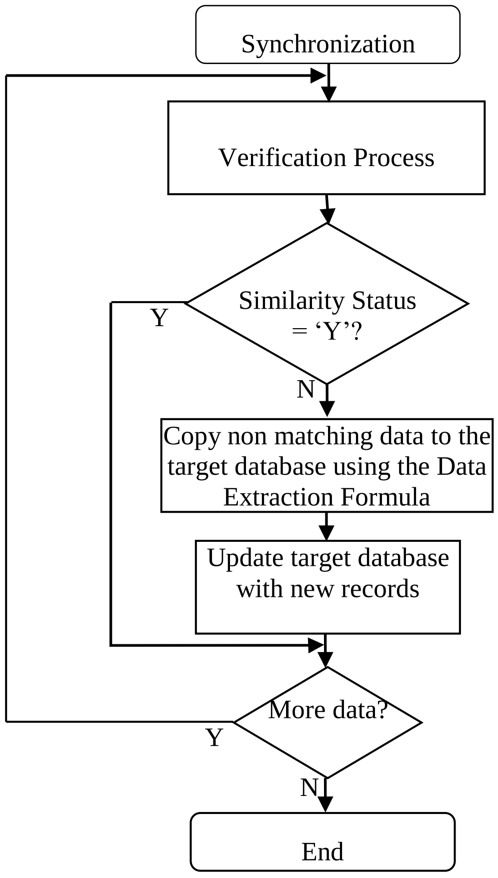
Synchronization
Verification Process
Similarity Status = 'Y'?
Y
N
Copy non matching data to the target database using the Data Extraction Formula
Update target database with new records
More data?
Y
N
End

Fig. 11. Synchronization process.

Using data extraction formula, the nonmatching records are copied to the target database, thereafter; appropriate action is applied to the copied records. If there are more data to synchronize the process is repeated, else, the process is ended. The verification is repeated in this phase because there is need to localize the verification at some points such as when there is more data to be synchronized immediately after the completion of the assigned tasks. Meaning the loop within the phase should be maintained until the data to be synchronized is exhausted.

*4)* Final Verification Process

This time, the verification process is to confirm the status of the most recent (just completed) synchronization whether is successfully completed or an error occurred during the process. If there was an error, the synchronization process is repeated, otherwise, the process is tagged to at rest, which means the process is disabled for now until there are more changes from either side.

The process uses the same diagram as shown in Fig. 10. Checking and comparison of data is repeated at multiple points due to the need to confirm that there is no more data to be synchronized before putting the entire process at rest. Putting the process at rest after a successful synchronization greatly minimizes the consumption of batteries and other mobile valuable resources such as CPU and Memory.

## V. Results and Discussion

This section presents and discussed in details the results of the proposed model and its counterpart. After subjecting the proposed model to a proper implementation, thorough testing was conducted to ascertain its capability and reliability in different aspects. These aspects are in-line with our goal in making data synchronization possible between the different mobile database vendors and the server-side database. At first the hypothesis are presented and tested, and results are produced for the null and alternative hypothesis. For clarity and easy reference, results are discussed immediately after they are presented.

The data obtained from the proposed and existing model were analyzed using the Chi$^2$ test which produces the $P$ value used to measure whether the null hypothesis ($H_o$) should be accepted or rejected. As for the level of significance, $\alpha = 0.05$ (5%) was used to indicate a 5% risk of concluding that a difference exists when there is no actual difference. Considering the formulated hypothesis, two-tailed comparison was considered. The tow-tailed test allows the comparison process to be fair to all the participating groups. Meaning, the proposed model could be better than the competitor's model or vice versa.

After examining the process repeatedly, the proposed model yields outstanding, profound and remarkable improvements from the existing solutions, mostly in the utilization of mobile resources due to the incorporation of DEF [23]. It also showed the prospect in multiple mobile database vendors' involvement in the synchronization process. The results are categorized and presented based on the aim of this study (mobile database heterogeneity).

Database heterogeneity refers to having different databases of different data model, DBMS, Vendors, and OS. Since there are varieties of mobile database vendors, a solution that neglect their individual differences and permit standard uniformity is required to be able to synchronize data limitlessly. The results of the proposed model in this regard are thereby presented in two scenarios: the first being the records exchange from Mobile Device Databases (MDD) to Server Side Database (SSD) while the second takes the opposite direction. In this context therefore, we aim to provide a general solution that can be used for data synchronization regardless of the aforementioned individual differences. One (latest) of the existing solutions was used to measure our proposed solution based on the following hypothesis.

***$H_O$*:** Database dependent information such as time-stamp, trigger or stored procedure have no impact in making any solution heterogeneous, i.e. vendor specific.

***$H_1$*:** Database dependent information such as time-stamp, trigger or stored procedure have impact in making any solution heterogeneous, i.e. vendor specific.

To answer these hypotheses, experiments are conducted and results were analyzed in two different scenarios. Scenario one (1) shows and discussed the results obtained from both the models when records are sent from Mobile Device Database (MDD) to Server Side Database (SSD), while scenario 2 present and discussed the results when records are sent from SSD to MDD. At first we begin by presenting scenario 1.

### A. Scenario 1: Data Exchange Possibility from MDD to SSD

Since our proposed solution is bidirectional (send to the server and receive from the server) we started answering the hypothesis by initiating a communication from the Mobile Device Databases (MDD) to Server Side Database (SSD). Table I summarizes the trials that has the highest available number of records.

TABLE. I. Scenario 1 MDD (SQLite & XML) To MySQL SSD

| | **SQLite** | **XML** |
|---|---|---|
| Number of Records | 10000 | 10000 |
| Proposed Model AST (ms) | 5.74 | 35.46 |
| Competitor Model AST (ms) | 9.93 | - |
| Data received by SSD using the PM | **True** | **True** |
| Data received by SSD using the CM | **True** | **False** |

Table I shows the possibility of receiving data and the Average Synchronization Time (AST) of both the proposed model and the competitor model. Looking at the Proposed Model (PM), apart from being able to receive the data sent from different mobile database vendors, it is also faster. While the Competitor Model (CM) was only able to receive data sent from SQLite because both SQLite and MySQL have the same data model and use the same DBMS, but for the case of XML, the process couldn't be completed. Table II below shows the trails at multiple levels.

TABLE. II. Scenario 1 MDD (SQLite & XML) To MySQL SDD

| | **SQLite** | **XML** |
|---|---|---|
| Trial 1 Number of Records | 500 | 500 |
| Trial 1 Proposed Model TST (ms) | 39010.1 | 181052.7 |
| Trial 1 Competitor Model TST (ms) | 31011.3 | - |
| Trial 1 data received by SSD using PM | **True** | **True** |
| Trial 1 data received by SSD using CM | **True** | **False** |
| Trial 2 Number of Records | 2000 | 2000 |
| Trial 2 Proposed Model TST (ms) | 48035.7 | 232311.2 |
| Trial 2 Competitor Model TST (ms) | 45501.1 | - |
| Trial 2 data received by SSD using PM | **True** | **True** |
| Trial 2 data received by SSD using CM | **True** | **False** |
| Trial 3 Number of Records | 5000 | 5000 |
| Trial 3 Proposed Model TST (ms) | 63210.3 | 291492.9 |
| Trial 3 Competitor Model TST (ms) | 69224.8 | - |
| Trial 3 data received by SSD using PM | **True** | **True** |
| Trial 3 data received by SSD using CM | **True** | False |
| Trial 4 Number of Records | 7000 | 7000 |
| Trial 4 Proposed Model TST (ms) | 71563.4 | 325143.4 |
| Trial 4 Competitor Model TST (ms) | 72100.0 | - |
| Trial 4 data received by SSD using PM | **True** | **True** |
| Trial 4 data received by SSD using CM | **True** | **False** |
| Trial 5 Number of Records | 10000 | 10000 |
| Trial 5 Proposed Model TST (ms) | 79461.3 | 364660.7 |
| Trial 5 Competitor Model TST (ms) | 91433.2 | - |
| Trial 5 data received by SSD using PM | **True** | **True** |
| Trial 5 data received by SSD using CM | **True** | **False** |

Talking of the Table II above, scenario 1 shows the possibility of synchronizing data to the server with a number of trials in both the

SQLite and XML databases using the Proposed Model (PM). In trial 1 of the scenario 1, the Mobile Device Database (MDD) was able to effectively synchronized data to Server

Side Database (SSD) for both the SQLite and XML databases. This achievement did not stop in trial 1 only, but across the remaining trails with different number of records, whereas, in the same trials, the Competitor's Model (CM) was able to synchronize data with SQLite database only. This is a clear indication that, the proposed model can be embraced by several database vendors, regardless of their individual difference because the model considers the interception areas rather than focusing on their individual differences.

Furthermore, the Total Synchronization Time (TST) taken for the proposed model to synchronized data to the server was 39010.1(ms) at first trial and 31011.3 (ms) for the competitor model in the same trial. The increase continued to correspond to the number of records in the trials diagonally with around 5.5(s).

## B. *Scenario 2 Data Exchange Possibility from SSD to HMDD*

In scenario 2, the opposite direction of the synchronization was considered where Server Side Database (SSD) sends records to Mobile Device Databases (MDD). Table III shows the summary of the trials conducted in Table IV.

TABLE. III. SCENARIO 2 MYSQL SDD TO MDD (SQLITE & XML)

| | **SQLite** | **XML** |
|---|---|---|
| Number of Records | 10000 | 10000 |
| Proposed Model ADAT (ms) | 3.98 | 4.73 |
| Competitor Model ADAT (ms) | 4.23 | - |
| Data received by MDD using the PM | **True** | **True** |
| Data received by MDD using the CM | **True** | **False** |

Table above shows that, the records sent by the server was received by Mobile Device Databases (SQLite and XML) using the Proposed Model (PM). While using the Competitor Model (CM), only SQLite was able to receive the data. Also, the Average Data Arrival Time (ADAT) was lower using the PM. Table below presents the results of the 5 trials.

As the case of scenario 2, the second direction of the synchronization is considered where the server sends records to its clients. Using the proposed model, both SQLite and XML databases were able to receive data composed and sent by the server crosswise, in all trials. While the competitor model behaved in contrast, where only the SQLite did received the records. This is because the competitor's model was based on SQL queries while others use database dependent information such as timestamp and trigger in the cause of synchronization, which eliminates some database vendors that do not have such techniques or mechanisms embedded or do not belong to RDBMS category at all.

In addition, it can be seen in both the scenarios 1 and 2 above that, using the Proposed Model (PM), the average time taken to synchronize records using SQLite is way less than the time taken with XML database even though they both send and receive data. This is because, in SQLite, multiple rows carry a fixed number of columns identifiers unlike in XML where multiple tags are used to wrap each record and group of records [26].

TABLE. IV. SCENARIO 2 MYSQL SDD TO MDD (SQLITE & XML)

| | **SQLite** | **XML** |
|---|---|---|
| Trial 1 Number of Records | 500 | 500 |
| Trial 1 Proposed Model DAT (ms) | 21023.4 | 24663.8 |
| Trial 1 Competitor Model DAT (ms) | 26001.3 | - |
| Trial 1 Data received by MDD using PM | **True** | **True** |
| Trial 1 Data received by MDD using CM | **True** | **False** |
| Trial 2 Number of Records | 2000 | 2000 |
| Trial 2 Proposed Model DAT (ms) | 30331.2 | 32415.9 |
| Trial 2 Competitor Model DAT (ms) | 34311.2 | - |
| Trial 2 Data received by MDD using PM | **True** | **True** |
| Trial 2 Data received by MDD using CM | **True** | False |
| Trial 3 Number of Records | 5000 | 5000 |
| Trial 3 Proposed Model DAT (ms) | 38096.9 | 39736.1 |
| Trial 3 Competitor Model DAT (ms) | 44709.7 | - |
| Trial 3 Data received by MDD using PM | **True** | **True** |
| Trial 3 Data received by MDD using CM | **True** | **False** |
| Trial 4 Number of Records | 7000 | 7000 |
| Trial 4 Proposed Model DAT (ms) | 46543.1 | 48280.7 |
| Trial 4 Competitor Model DAT (ms) | 47016.3 | - |
| Trial 4 Data received by MDD using PM | **True** | **True** |
| Trial 4 Data received by MDD using CM | **True** | **False** |
| Trial 5 Number of Records | 10000 | 10000 |
| Trial 5 Proposed Model DAT (ms) | 52206.4 | 55113.9 |
| Trial 5 Competitor Model DAT (ms) | 59236.8 | - |
| Trial 5 Data received by MDD using PM | **True** | **True** |
| Trial 5 Data received by MDD using CM | **True** | **False** |

For example, in SQLite, if you have 10000 rows of records and have 5 columns then you would have 5 columns identifies, one for each column. However, for the same number of records using XML, you would have 100,000 wrappers (that is to say, 10,000 rows * 5 records par row * 2 opening and closing tags). This adds so much load to the data, thus make heavy for mobile devices to manipulate easily.

Chi$^2$ test was used to analyze the above data that states the possibility of synchronizing records between mobile heterogeneous databases. Since our data in this case is TRUE or FALSE, a statistical tool that will allow the probabilities to be counted and aggregated is selected which works as follows.

TABLE. V. CHI$^2$ TEST DATA FROM SCENARIO 1 AND 2

| | **Number of True** | **Number of False** | **Grant Total** |
|---|---|---|---|
| Proposed Model | 20 | 0 | 20 |
| Competitor Model | 10 | 10 | 20 |
| Grant Total | 30 | 10 | 40 |

Table V shows the data (True and False count) retrieved from scenario 1 and scenario 2 as presented in Table II and Table III. Therefore, the (column total * row total) /grant total formula was used to calculate the expected values for the data presented in Table V. The results of the computation are as shown in Table VI.

TABLE. VI. EXPECTED VALUE RESULTS

| | **Number of True** | **Number of False** | **Grant Total** |
|---|---|---|---|
| Proposed Model | 15 | 5 | 15 |
| Competitor Model | 15 | 5 | 10 |
| Grant Total | 20 | 5 | 25 |

After computing the Expected Values (EV) as indicated above, the Actual Values (AV) and the EV were included in the Chi$^2$ test formula to obtain the probability value. On the other hand, 0.05 was set to be the alpha (α) value. These values can be used to either accept or reject the null hypothesis. $H_o$ can be only rejected if the probability value is less than the alpha value. Results of the analysis shows that, the *p* value is 0.00026073 for both the two scenarios, which is less than the alpha (α) value of 0.05.

Based on this therefore, the null hypothesis is fully rejected since the probability value is less than the alpha value. The outcome shows the possibility of sharing data across multiple mobile databases when database dependent information such as timestamp, triggers and Meta data are excluded in the solution. Solutions that adopt any of these techniques are thereby considered vendor specific or solution that is homogenous in nature.

## VI. Conclusion

We have presented a model for purpose of addressing the problem of data synchronization between the heterogeneous mobile devices databases and server-side database with significant consideration to the limitations of the mobile devices such as memory, CPU, power supply and continuous network fluctuations. Based on the goals of this study, experimental method which allows study to be carried out in a real life context was considered to be the most suitable for this research. This method was selected out of the five methods discussed by Easterbrook [20] for the empirical software engineering research. The study explored and investigated numerous solutions from the existing literature where various incredible research contributions were found. However, mobile database heterogeneity was uncared for in spite of its great importance. Thus hinders other types of databases to participate in the synchronization process since they were not considered as part of the solution in the first place.

Based on the review outcome, existing solutions properties were identified which guided the construction of the proposed model. To empirically validate the proposed model, a prototype was developed which implemented the model in a real-life context. Also one latest existing solution was implemented for the purpose of performance analysis.

The proposed model further weighs against the existing model to mark the improved areas. Results indicate that the objectives of this study have been achieved where the proposed model proved feasibility of engaging multiple mobile databases in a synchronization process; thus delivering substantial evidence to repudiate the null hypothesis. Moreover, the proposed model displayed some strength in the synchronization speed and also the utilization of the mobile resources.

Looking at the unique intensity that the competitor model and proposed model offer, there is need to consider the significance of heterogeneity and resource consumption when making the decisions between the models. The actual potency of the proposed model lies in the aforementioned variables. The study has provided a clear benchmark that can be used to compare these models when adopting a synchronization solution for mobile devices. Unstructured data are another key important component that will be given due consideration in the nearby future since mobile devices are now one of the major sources of big data [27].

## Acknowledgement

This paper/research was fully supported by Ministry of Higher Education Malaysia, under the Fundamental Research Grant Scheme (FRGS) with Ref No of: FRGS/1/2015/ICT01/UTP/02/1. Any opinions, findings, and conclusions stated in this paper are those of authors and do not necessarily reflect those of the MOHE.

## References

[1] M. Nayebi, B. Adams, and G. Ruhe, “Release Practices for Mobile Apps -- What do Users and Developers Think?,” 2016 IEEE 23rd Int. Conf. Softw. Anal. Evol. Reengineering, pp. 552–562, 2016.

[2] 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.

[3] J. Sedivy, T. Barina, I. MOrozan, and A. Sandu, “MCSync – Distributed , Decentralized Database for Mobile Devices,” IEEE 2012, pp. 1–5, 2012.

[4] M. F. Qaisrani, “Types of Distributed Database Management System,” Benazir Bhutto Shaheed University, 2014. [Online]. Available: http://www.slideshare.net/TAHAROC/types-of-data.

[5] 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.

[6] 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., p. pp.416,419, 25-27, 2009.

[7] 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.

[8] B. S. Ramya, S. B. Koduri, and M. Seetha, “A Stateful Database Synchronization Approach for Mobile Devices,” Int. J. Soft Comput. Eng., vol. 2, no. 3, pp. 316–320, 2012.

[9] M. Ahluwalia, R. Gupta, A. Gangopadhyay, and M. Mcallister, “Target-Based Database Synchronization,” in Proceedings of the 2010 ACM Symposium on Applied Computing, 2010, pp. 1643–1647.

[10] H. Preston and M. Narayan, “Message digest based data synchronization,” US 09/896,321.

[11] P. Bottorff, C. L. Allen, A. Hudson, and M. R. Krause, “Distributed database synchronization,” US 12/911,356.

[12] 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.

[13] 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.

[14] 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.

[15] J. Domingos, N. Sim??es, P. Pereira, C. Silva, and L. Marcelino, “Database synchronization model for mobile devices,” in Iberian Conference on Information Systems and Technologies, CISTI, 2014.

[16] 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.

[17] A. A. Imam, S. Basri, and R. Ahmad, "Synchronization Algorithm for Remote Heterogeneous Database Environment.pdf," in Advances in Intelligent System and Computing, 2014, pp. 55–65.

[18] A. Stage, "Synchronization and replication in the context of mobile applications," in ICFN '10. Second International Conference on, 2012, p. 98,101.

[19] H. Chen, J. Yu, C. Hang, B. Zang, and P. Yew, "Dynamic software updating using a relaxed consistency model," Softw. Eng. IEEE Trans. on, Vol. 37(5), pp. 679–694, 2011.

[20] S. Easterbrook, J. Singer, M.-A. Storey, and D. Damian, "Selecting Empirical Methods for Software Engineering Research," Guid. to Adv. Empir. Softw. Eng., pp. 285–311, 2008.

[21] R. K. Yin, "Case Study Research: Design and Methods.," Sage, 2002.

[22] D. Perry, A. Porter, and L. Votta, "Empirical Studies of Software Engineering: A Roadmap," Int. Conf. Softw. Eng., pp. 345–355, 2000.

[23] A. A. Imam, S. Basri, and R. Ahmad, "Data Extraction Formula for Efficient Data Synchronization between Mobile Databases and Server-side Database," in International conference on Computer and Information Science (IEEEC 2016), 2016.

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

[25] A. A. Imam, S. Basri, and R. Ahmad, "Data Synchronization Between Mobile Devices and Server-side Databases : A Systematic Literature Review," J. Theor. Appl. Inf. Technol., vol. 81, no. 2, pp. 364–382, 2015.

[26] J. Fong, H. K. Wong, and Z. Cheng, "Converting relational database into XML documents with DOM," Inf. Softw. Technol., vol. 45, no. 6, pp. 335–355, 2003.

[27] B. B. Mehta, "First Credit Seminar Presentation on " Privacy and Big Data : Issues and Challenges", 2014.