ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 4, April 2016

Synchronization between Mobile Devices and Server: An Optimistic Approach

Nilesh Sonawane[1], Shahbaj Shaikh[2], Prachi Bhalerao[2], Pooja Kachroo[2], Asmita Pete[2]

Assistant Professor, Dept. of I.T., PVG's COET, Savitribai Phule Pune University, Pune, India[1]

Dept. of I.T., Pune Vidyarthi Griha's College of Engineering and Technology, Pune, India [2]

ABSTRACT: Synchronization algorithms have been used and arebeing studied in distributed systems especially client-server architecture aiming to achieve data integrity. The paper aims to throw light on previously developed synchronization techniques like Microsoft Synchronization Framework, SAMD, TLRSP and compare it with newly proposed design of a conceptual model. This bidirectional synchronizing model can be implemented and enhanced iteratively and functionality can be added according to the prioritizations in specific requirements. This synchronization algorithm is an optimistic timestamp based method. It is the key methodology to support collaboration amongst different mobile users by enabling anytime and anywhere access to shared database. In this paper, we provide solution considering the selection of data for synchronization which is obtained by keeping track of the timestamp lastly updated. The strategy is to support bidirectional sync from multiple clients to ensure data consistency whereas only incremental changes will be exchanged.Each database row has a timestamp of when it was last changed. Conflict detection is performed by comparing the timestamps in Last Modified field of both client and server's synchronization tables. A simple algorithm is proposed as an efficient solution for database synchronization when client nodes of the system are assumed to be mobile phones under the Android operating system.

KEYWORDS: Synchronization, framework, optimistic, timestamp

I. INTRODUCTION

In the real world, it happens many a times that an application needs to be available offline, allowing the users to keep working and once they go online again to synchronize their data back and forth with the server. This is known as Occasionally Connected Application (OCA). In OCA, users work with a local copy of the database and it is then updated with the server side central database. It seems to be much common task, but there is no any 'standard' solution as yet. Here, handling all the exceptions and edge cases is challenging. Synchronization is used to maintain the consistency between such client and server databases using web services. Web Service is a collection of standards or protocols for exchanging information between two devices or application. The two mainly used web services are: SOAP (Simple Object Access Protocol) and REST (Representational State Transfer protocol). REST is generally preferred over SOAP as it can handle web services in JSON format along with XML. The most important part of this paper is the synchronization between the handheld clients and the server, how to manage the inconsistencies in the database and how to keep track of the incremental changes and reduce the network traffic. Out of the two known schemas, our paper is based on Optimistic schema wherein replication allows clients, like mobile devices, to work concurrently and assumes that in the synchronization conflicts will appear and relies on conflict solving to eventually agree on a consistent state instead of a blocking conflict avoidance scheme, unlike Pessimistic techniques, which blocks other users from writing, or even accessing a data block while another user has writing privileges. Some advantages the optimistic synchronization over pessimistic are, better availability and flexibility. The components in the network can work in an offline state and then regardless of other changes commit its work. Also the partial method of synchronization is considered over full synchronization method due to its feature of synchronizing a subset of the replica depending on logs or change tracking values. The subset is updated and added objects/database rows as well as information on deleted data whereas full or slow synchronization is used when the whole replica is copied over to the client and is performed when the client synchronize with a new server or hasn't synchronized within a defined period of time and therefore has an old replica. This time period is often the lifetime of change tracking values for a database. This proposed algorithm will be explained in the next section. Further, it's comparison with the existing ones will be briefly discussed in later sections.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 4, April 2016

II. PREVIOUS METHODOLOGIES

A. *SAMD*

Synchronization Algorithm based on Message Digest undergoes synchronization between server and client database. It compares the Message Digest Tables of both client side as well as server side database to select the rows that are needed for synchronization.The SAMD algorithm allows the handheld client device to download a limited amount of replicated data from the server which is needed for its user to perform work in an offline state. It uses only standard SQL functions for Synchronization. Also actual processing is done on the server-side and not on the client. The drawbacks of this algorithm are that client sends its entire data to server whenever synchronization is performed which can be problematic in case of a bad network connection. Even a lot of extra tables are created on the server-side.

B. *TLRSP*

The Transaction Level Result-Set Propagation (TLRSP) is an algorithm modelled for optimistic database replication between handheld devices and database servers and hence, allows the handheld devices (mobiles) to access and commit changes to its local replica of the database in an offline-state. When mobile device is reconnected, the committed changes on mobile database are checked for conflicts on the server-side, if no conflicts are detected then the data is inserted into the server's replica of the database.A mobile device can be in three different stages. The consistent state is occupied when all conflicts have been resolved i.e. synchronization is completed. In the Accumulating state, the local replica of the database accommodates various changes that have been done. All read sets, write sets and result sets are logged in the mobile device which are to be used in synchronization. A read set consists of all objects that are read while write set contains all objects written to the database and result set consists of data objects from the writing and a Timestamp. The last state is Resolving state. When the mobile device reconnects to the database server a synchronization process will be carried out by sending its locally committed data, found by the logs, to the database server. The database server resolves all conflicts that may occur and incorporates the data into the database and then answers the mobile device with the updated data so it can return to initial state (i.e. consistent state). The only problem with this algorithm is conflict reconciliation (i.e. the problem of serializing potentially conflicting updates from disconnected clients on all replicas of database). Also the changes made on the mobile devices can be cancelled on the server if the data already has been changed.

C. *Microsoftsyncframework*:

*M*icrosoft Sync Framework is a closed platform. It allows limited modification in synchronization module. In this sync framework, to track changes between synchronizations so that the client only needs to download incremental changes and to detect conflicts when a client uploads data that has been changed on the server as well, MSSQL's built-in ChangeTracking functionality is used. In this functionality, a retention period is initialized, which indicates the time for which server will store the updated data. If a user wait for longer than this period between synchronization, the server cannot determine which changes has occurred and thus, all relevant data must be sent again to the client and, more importantly, changes that is uploaded from the client cannot be properly checked for conflicts and might have to be discarded. This is not the case with proposed synchronization model in this paper. There is no as such retention period for the server. Data never have to be discarded. Sync framework is neither applicable for high volume nor low latency application whereas the proposed model is.

III. PROPOSED ALGORITHM

Pre requisites:

In client-server architecture, server acts as a central repository. Server serves up resources through SOAP, REST API's etc. The scenario considered in below mentioned algorithm is the bi-directional synchronization between central server and multiple handheld clients where only the incremental changes apply on both the sides by resolving conflicts if any.

Along with the tables present in the client and server database, there is an additional tables maintained on both the sides as a provision for synchronization process.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 4, April 2016

PrimaryKey	ClientLastModified	Sync

Table 1: Client synchronization table

LastSynced

Table 2: last sync table

The above tables, table 1 and 2 represent tables on client side. When the record with particular primary key is created, modified and deleted, the sync field in table 1 is set to 1,2 or3 respectively.
The initialization of synchronization process was carried out by client either manually by pressing the Sync button or automatically after certain period or by combination of push notifications about server changes and client change triggers.

PrimaryKey	ServerLastModified

Table 3: Server synchronization table

Table 3 is stored on server side.Last Modified stores the timestamp during creation, modified and deletion of a contact. This field in both tables is used to determine highest timestamp for the table so as to download only the incremental changes.
The sync process first download the server changes then upload the client changes.The sync procedure will process the table changes in sequence and in the proper order to maintain integrity and avoid conflicts.

Initialization:
At the beginning, initialization of client side database tables will always be same as that of the server side tables.
In the client side synchronization table, sync bit will be initialized to 0 for every contact.

IV. ALGORITHM

1. Client request for data having 'lastModified' > 'LastSyncedDate&Time// Fetched data arranged in ascending order of timestamp.
2. For (i: 1 to n) // where n is the last record
Compare 'ServerLastModified' with 'ClientLastModified' of i^{th}record
3.// If the server side data is recent
If ('ServerLastModified' of records > 'ClientLastModified' of records)
If ('ServerLastModified' of record = = 31-12-9999)
If (flag = 0)
Delete contact from database Delete entry from client side sync table
Else
Get the value of record and timestamp from server.
Update these values at client side.
'LastSyncedDate&Time'='ClientLastModified'
Reset the sync bit.
4.//If client side data is recent
Else if ('ServerLastModified' field of record < 'ClientLastModified' field of record) If (sync bit =2) Modify the record on server side using PUT. Set the 'ServerLastModified' field to the record's 'ClientLastModified'. Reset the sync bit. 'LastSyncedDate&Time'='ClientLastModified' Else if(sync bit =3)
Delete the record from server side database.
Delete entry in client side contact sync table of this record.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 4, April 2016

5. Fetch the list of records with sync bit != 0 6.Iterate through each record in the collection

If(sync bit =1)

Create a record on server using POST.

If(sync bit=2)

Call relevant web service for modification.

Set the 'ServerLastModified' field to the record's latest client side timestamp.

If(sync bit =3)

Call relevant web service for deletion of a record.

Delete the entry of the record from sync table at client.

Reset the sync bit.

7. 'LastSyncDate&Time' = Max (ClientLastModified)

V. WORKING

Above mentioned algorithm is explained with the help of an example as follows. We have considered, an application having contacts database with Email id as a primary key.

A. Tables at client side

PrimaryKey	ClientLastModified	Sync
shahbaj@gmail.com	1/2/2016 13:00:20	0
pooja@yahoo.com	1/2/2016 19:00:40	2
prachi@gmail.com	1/2/2016 18:00:15	2
asmita@yahoo.com	1/2/2016 17:30:20	3
sumit@gmail.com	1/2/2016 19:30:01	1

Table 4 Client Synchronization Table

LastSynced
16:00:05

Table 5 Last Sync Table

The proposed algorithm can be clearly understood with the help of the given tables. The last synchronization time is 16.00.05. All the updates after this particular time will be taken into consideration.

B. Tables at server side

PrimaryKey	ServerLastModified
shahbaj@gmail.com	1/2/2016 16:30:20
pooja@yahoo.com	1/2/2016 18:30:15
prachi@gmail.com	1/2/2016 19:00:15
asmita@yahoo.com	1/2/2016 16:00:15
pravin@yahoo.com	31/12/9999 20:05:00

Table6

Applying the proposed algorithm to above given example:

Step 1: Only the data having timestamp greater than LastSynced field (in this case 16:00:05) is fetched from server. This fetched data is stored in ascending order as given below. This is done because the synchronization is performed according to the incremental changes i.e. one operation after another.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 4, April 2016

Step 2: Comparison of ClientLastModifiedand ServerLastModified is done for every record of table in Step 1.

Step 3: In case of first record, this step will be executed as server-side timestamp is recent also sync bit is not equal to 3. Hence the desired operation according to sync bit is performed and record is updated at client-side. 'LastSyncedDate&Time' is changed to 'ClientLastModified' and sync bit is reset.

Step 4: In case of second record this step will be executed as client-side is recent. Operation is performed as done in Step 2. The only difference is that the 'LastSyncedDate&Time' value is changed to the minimum among 'ClientLastModified' and ServerLastModified. Also sync bit is reset.

Step 5: In this way the loop is executed till all the updated records at server-side are implemented. After its successful completion those records at client-side which have sync bit greater than zero are fetched.

Step 6: These records are then implemented incrementally according to the specified operations.

Step 7: 'LastSyncDate&Time' is set to the maximum value of ClientLastModified field.

A. Table at client side

PrimaryKey	ClientLastModified	Sync
shahbaj@gmail.com	1/2/2016 16:30:20	0
pooja@yahoo.com	1/2/2016 19:00:40	0
prachi@gmail.com	1/2/2016 19:00:15	0
asmita@yahoo.com	1/2/2016 17:30:20	0
sumit@gmail.com	1/2/2016 19:30:01	0

Table 7: Client Synchronization table

LastSynced
20:05:00

Table 8:LastSync Table

Step 8: In case of last record, deletion condition is verified for proper working in a multiple client scenario. When a particular client logs in with different device, deletion of record with keyword 'pravin@yahoo.com' should be traced. With the help of the timestamp '31/12/9999', it will be immediately notified about the deletion of particular record. After performing these steps the final tables would be as above table 8 and 9.

B. Table at server side

PrimaryKey	ServerLastModified
shahbaj@gmail.com	1/2/2016 16:30:20
pooja@yahoo.com	1/2/2016 19:00:40
prachi@gmail.com	1/2/2016 19:00:15
asmita@yahoo.com	1/2/2016 17:30:20
pravin@yahoo.com	31/12/9999 20:05:00
sumit@gmail.com	1/2/2016 19:30:01

Table 9: Server Synchronization table

The table 9 denotes the final snapshot of the server side changes implemented into the Synchronization table after following the steps explained above. Now, all the timestamps are updated in an efficientmanner. It is the recursive process as same steps will be repeated when the synchronization is initialized for the next time either automatically or manually.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 4, April 2016

VI. COMPARISON

The summary of the comparisons between previously implemented algorithms and the proposed algorithm is depicted with help of table. Few important properties among many other are highlighted in the table 10.

Properties	SAMD	TLRSP	Proposed
Data sent for processing	Whole data	Categorized Data	Modified Data
Number of tables on server side	More	No tables (Sets)	Less
Conflict reconciliation in multi-client	-	Not considered	Efficiently performed

Table 10

VII. CONCLUSION

This paper has suggested an effective and efficient algorithm for synchronizing between server-side databases and handheld client database. This algorithm is performed by taking into consideration, the timestamp of every record stored in relational database which is created at client as well as server. It is been compared with well-known methodologies like SAMD, TLRSP and microsoft sync framework. The literature study clearly suggest that proposed algorithm overcomes certain drawback mentioned in the paper.

ACKNOWLEDGEMENT

This paper has suggested an effective and efficient algorithm for synchronizing between server-side databases and handheld client database. This algorithm is performed by taking into consideration, the timestamp of every record stored in relational database which is created at client as well as server. It is been compared with well-known methodologies like SAMD, TLRSP and microsoft sync framework. The literature study clearly suggest that proposed algorithm overcomes certain drawback mentioned in the paper.

REFERENCES

1. P.Kalyanakumar, A.Sangeetha, 'A synchronization algorithm for mobile database using SAMD', IRJET, Vol. 02 Issue: 02,pp. 393 May-2015.
2. Mi-Young Choi, Eun-Ae Cho, Dae-Ha Park, Chang-Joo Moon, Doo-Kwon Baik, 'A synchronization algorithm for mobile devices' IEEE Transactions on Consumer Electronics, Vol. 56, No. 2, May 2010.
3. Ding Zhiming, Meng Xiaofeng, Wang Shan, 'A transactional asynchronous replication scheme for mobile database system', J. Computer science and technology, vol. 17 No. 4, pp 389-396, July 2002