RESEARCH ARTICLE OPEN ACCESS

Impact of Data Distribution on Response Time in Telecom Databases

Imran Ashraf [1], Amir Shahzed Khokhar [2], Dr. Lars Lundberg [3]
Blekinge Institute of Technology, Sweden [1,2,3]

Abstract
Centralized databases are becoming bottlenecks for physically distributed organizations. They carry the disadvantages of high communication cost and high data retrieval time. The emerging distributed database technology promises increased availability, reduced cost and improved response time. However, this area needs to research many issues before implementation. Data fragmentation and allocation are very important and critical steps in distributed databases. It is a pivotal area that needs to be investigated properly especially with reference to the telecom sector. The primary focus of this research is to study the impact of fragmentation and distribution on the data retrieval time. However it will also cover the selection of the most appropriate fragmentation strategy depending upon the database architecture and data selection patterns.
Index Terms: Distributed databases, fragmentation, data distribution, data allocation

I. INTRODUCTION

The 20th century witnessed a lot of development in databases. It started with the storage of simple files which were treated as a unit and were called 'databases' in 1964 [1]. In 70's different database architectures were devised like hierarchical database system [2] and object oriented databases. However, the real revolution came with the inception of relational databases by E.F.Codd in 1970 [2, 3]. Relational databases tackled the problems of complexity of model, lack of standards and reduced flexibility present in the previous architectures.

Based on the relational model two types of databases are used today namely: centralized and distributed databases. Centralized databases are easy to manage, highly secured and concurrent. But the problem of single source bottleneck, high communication costs and high response time led to the idea of distribution of data.

Large companies need to distribute the data for many reasons e.g. for being economic and competitive (Srivastava, Shankar e Tiwari, 2012). But the main motivation behind the concept is the efficient management of data with increased availability and reduced communication cost. It has become very attractive solution for areas like online banking, e-commerce merchant, HR departments, telecommunication industry and air line ticketing etc [4].

However, there are some crucial and decisive issues that need to be researched properly before adopting distributed databases. These issues include concurrency control, backup and recovery, fragmentation [5], deadlocks [6], distribution, data replication, query processing and optimization and data allocation etc. Data fragmentation and its impact on the response time is our chief interest here.

II. RELATED WORK

Substantiating and incorporating data in distributed databases led to additional complexities like concurrency control, data allocation, data partition, query optimization, data replication, data integrity and security etc. Efforts [4, 6-14] have been put to overcome these issues in general but very little has been done with special reference to the telecom industry. Survey conducted by Mathias Jarke et al. in 2000 found only 7 papers related to distributed databases in telecom. Moreover the impact of data fragmentation and distribution on the response time in telecom databases is not very well studied.

III. FRAGMENTATION

It is the decomposition of a relation into fragments each being treated as a unit [15-17]. Fragmentation is done according to the data selection patterns of applications running on the database. It permits to divide a single query into a set of multiple sub-queries that can execute parallel on fragments.

A.Types of Fragmentation

Fragmentation is basically divided into two categories and they are mentioned as under:

- Vertical fragmentation
- Horizontal fragmentation

1. Vertical Fragmentation

It divides a relation into fragments which contains a subset of attributes of a relation along with the primary key attribute of the relation. The purpose of vertical fragmentation is to partition a relation into

a set of smaller relations to enable user applications to run on only one fragment [15-17] .

2. Horizontal Fragmentation

It divides a relation into fragments along its tuples. Each fragment is a subset of tuples of a relation. It identifies some specific tuples based on user access patterns and marks it as a fragment. Horizontal fragmentation is further divided into two types.

2.1Primary Horizontal Fragmentation

This type of fragmentation is done where the relations in a database are neither joined nor have dependencies. So, no relationship exists among the tables.

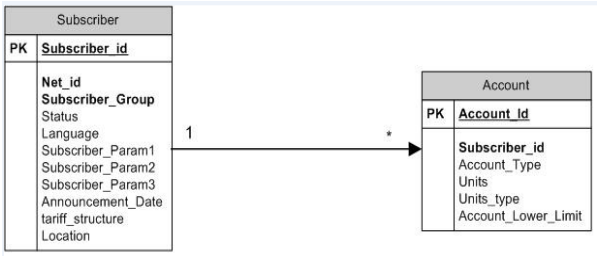
2.2Derived Horizontal Fragmentation

Derived horizontal fragmentation is used for parent relation. It is used where relations are interlinked with the help of foreign keys. It ensures that the fragments which are joined together are put on the same site. In this research work, derived horizontal fragmentation is used [15-17].

IV. DATA MODEL

A.Data Model

The data model contains two tables; subscriber and account. Subscriber table holds customer information and his connection preferences while the table account contains the details of customer accounts. The diagram depicts the attributes of relations and relationship between them.



B.Experimental Setup

The experiment was designed to compare and analyze the impact on the response time before and after data fragmentation and distribution. The setup contains 5 sites linked to a central server. Systems operate on Solaris and Oracle 9i was used as a database for the experiment. Each request is sent to the server, processed and response is then sent back. Sites do not communicate among themselves. Figure 1 shows the experiment setup.

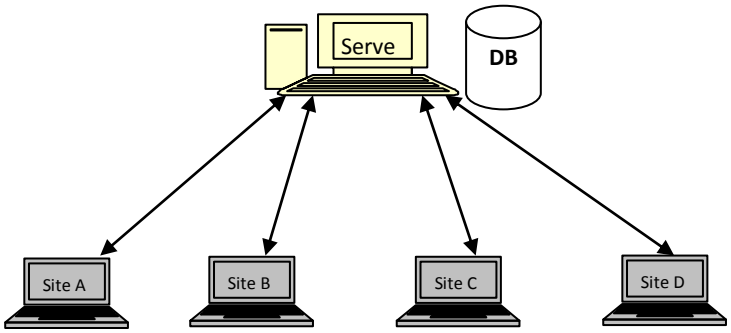


Figure 1

C.Experiment

This test was performed to take response time values for one centralized database before fragmentation and distribution was done. All sites may send queries at the same time. Response time is taken in milliseconds. Queries were executed a number of times to take as real and consistent response time values as possible.

V. STATISTICS FOR CENTRALIZED DATABASE

Table 1contains the statistics for centralized database used in our experiment. Results were taken a number of times and then their average was calculated to take best values.

Centralized Database Statistics with Workload			
Query	**Best case (ms)**	**Average (ms)**	**Worst case (ms)**
Normal call	27.5	43.6	124
Refill	28.5	43.9	130.5
Balance inquiry	20.25	32.75	152.75
Change language	2.75	6.22	116
SMS-Charge	25.75	40.612	139
Pre-activation	3	5.466	63.66
Subscriber list / subscriber group	1	1.84	44
Subscriber/ subscriber group	37.25	58.55	89.75
Units / subscriber group *	621	878.55	911.25

Figure 1

Table 1 displays the best case, worst case and average values fond in our experiment. This table has been derived from main table given in [18].

Figure 2 shows a graph based on table 1. However it does not include the marked query (Units/subscriber group). Its values are too high to be adjusted in the graph. This query checks most of the records in both the relations (in database), so its

response time is sufficiently high. Normal call and refill queries have a high response time as they perform both read and write operations on both the relations. Though, change language and pre-activation also perform data updates, but their response time is less mainly because update is performed only on one relation i.e. subscriber.

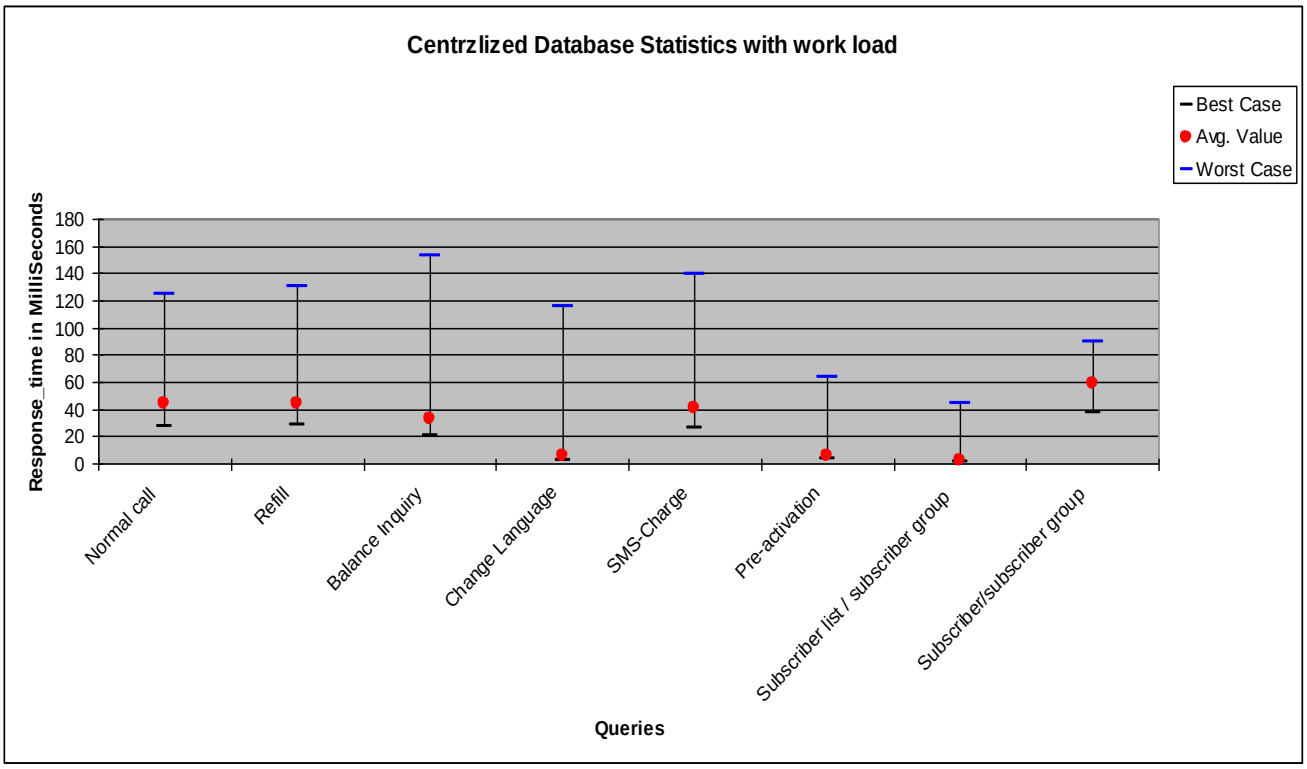


Figure 2

Statistics of Average values for all sites (milliseconds)					
Query	**Server**	**Site 1**	**Site 2**	**Site 3**	**Site 4**
Normal call	42.5	44	43.5	43.5	44.5
Refill	42.5	45	45.5	44	42.5
Balance inquiry	32.5	32.25	33	32.5	33.5
Change language	5	6	6.6	6.75	6.75
SMS-Charge	40	39.66	42.6	40.2	40.6
Pre-activation	4	4.5	7	6.5	5.33
Subscriber list/ subscriber group	1.2	2	2	2	2
Subscriber/ subscriber group	59	56.5	59	59.25	59
Units / subscriber group *	878.25	900.75	878.75	879.25	878.5

Table 2

Table 2 shows the average time for all the sites that were part of the experiment. Values were taken when all sites were sending requests to the server. One of the purposes of this measurement was to ensure that all the sites were having more or less same processing capabilities.

The graph in the **Figure 3** illustrates the average time values for each site. Response time for server deviates a little from other site pattern. Its response time is little less as compared to other sites. The possible reason for this is that the database is local to it. This difference depends upon the communication

channel speed, communication distance, server site workload and client data access approach (File approach, data transfer approach etc).

VI. DATABASE FRAGMENTATION

We discussed fragmentation and its types i.e. vertical and horizontal fragmentation, in the previous section. Fragmentation depends upon factors like the data selection patterns and database architecture etc. For the given database horizontal fragmentation is suitable. It works in databases where tables don't have much columns and mostly a complete row is selected as a response to a request.

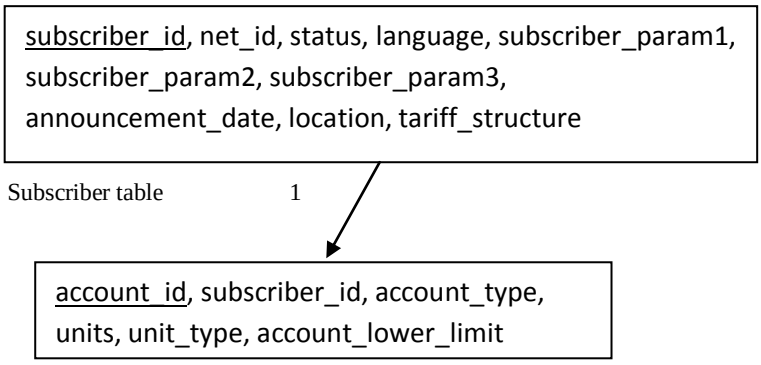
A. Information Requirements for Horizontal Fragmentation

For fragmentation two types of information is required.

1.Information Requirements for Horizontal Fragmentation

1.1Databas Information

It is very important to know the database schema and database relations prior to fragmentation. It is equally important to study how database relations are connected to one another and how joins are made [3]. The connections for the given database are shown as below.



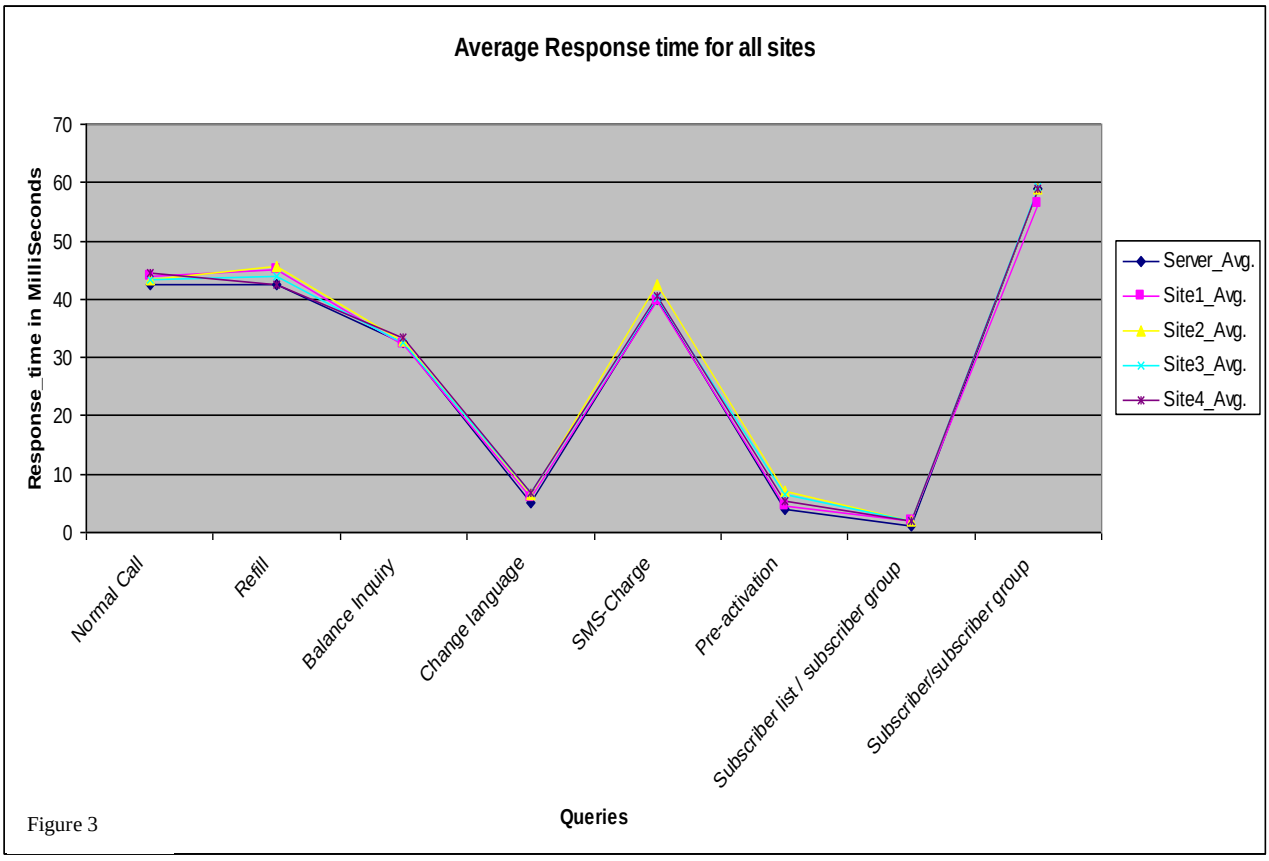


Figure 3

Subscriber in this picture is called the 'owner' while account table is called the 'member' [19]. Each subscriber may have one or more than one records in the account table (one-to-many relationship

1.2Application Information

It is fundamental to know about the predicates used in user's queries. For this purpose we need to know about all queries performed by every application. In case it is not possible, important applications queries should be analyzed at least. Wiederhold suggested 80/20 rule for this purpose. It implies that, "*the most active 20% user's queries account for 80% of the total data access*" [19]. Queries used for the given database are given in [18].

B. Fragmentation

Fragmentation depends upon user access patterns. It is always started with the owner table and then it moves toward the member table. We used two algorithms for the purpose of horizontal fragmentation; COM_MIN algorithm and P_HORIZONTAL algorithm [16]. Their detail is given in [18].

C. Distribution

Efficiency and functionality of DDBs is critically dependent on its design in terms of fragmentation and distribution [3, 20]. The prime purpose of data allocation is to deploy fragments such that it reduces the communication cost i.e. to minimize the data transmission over sites during queries execution. For this purpose we need to consider the size of the fragments, communication cost, processing cost and processing cost etc. efficient allocation means a balance between cost and performance by following the prescribed constraints [21]. Fragment allocation was done using 'Allocation Model' given in [4]. These fragments carried the same relationship and dependencies as we had in the given database and they were interconnected.

VII. EXPERIMENT FOR DISTRIBUTED DATABASE

The purpose of this execution is to get the values for minimum, maximum and average response time of distributed database. These values are collected both for local access and foreign access of data. In local case every site performs operations in its own database and response time for that particular site is taken. However, in foreign access case each site may access data from one or more other sites. So, the response time for all sites used to fetch a particular set of data is taken. In both cases all sites are performing operations at the same time.

Table 3

Response time Statistics for all sites (Milliseconds)						
	Local Access			Foreign Access		
Queries	Min	Avg.	Max	Min	Avg.	Max
Normal Call	2.4	4.3	31	3.4	6.7	47.8
Refill	1.8	3.3	23.4	2.6	5.9	33.2
Balance Inquiry	1	1.65	15.2	1.6	2.5	26
Change language	1.2	2.05	17.4	2	3.6	27
SMS-Charge	1.4	2.15	21	2	2.9	28
Pre-activation	1	1.65	16.4	1.6	2.4	21.4

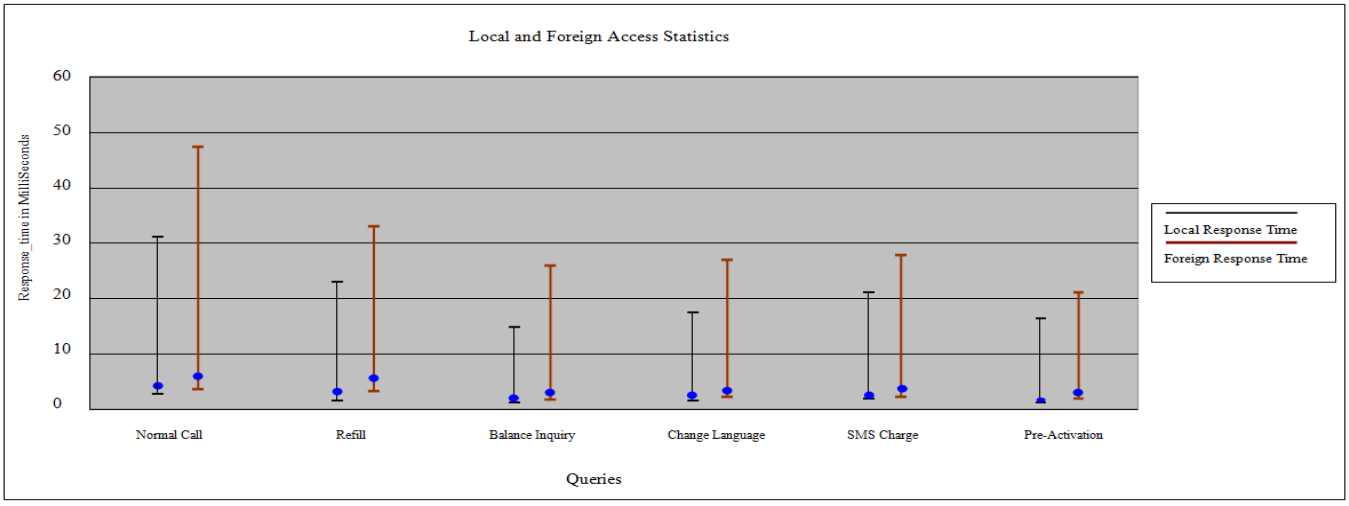


Figure 4

A. Comparison of response times

The following table describes the average values of all queries used in the centralized and the distributed environment.

Comparison between Centralized and Distributed database		
Query	**Centralized Avg.**	**Distributed**
Normal call	27	4.3
Refill	28	3.3
Balance Inquiry	20.5	1.65
Change Language	3	2.05
SMS-Charge	26	2.15
Pre-activation	4	1.65
Subscriber list / subscriber group	1.2	36.25
Subscriber/subscrib group	37.8	63
Units/subscriber group*	526	559.25

Table 4

The values in table 4 are mean values calculated for the centralized and distributed databases. Readings were taken for a number of times in order to consider the changing workload at a given time. Then from these values their mean was calculated. Graphs given below shows that average values for distributed database are less than centralized environment. In centralized databases data is accumulated at one place, so it takes long to process queries. Moreover data index tables are large to search.

Since, in distributed databases data is kept local to the site where it is needed, so response time is very good. While in distributed databases fragments are shorter, so their short index tables make searching fast. An additional point is that since fragments are based on user queries so user queries are directed to concerning fragments which also improves response time.

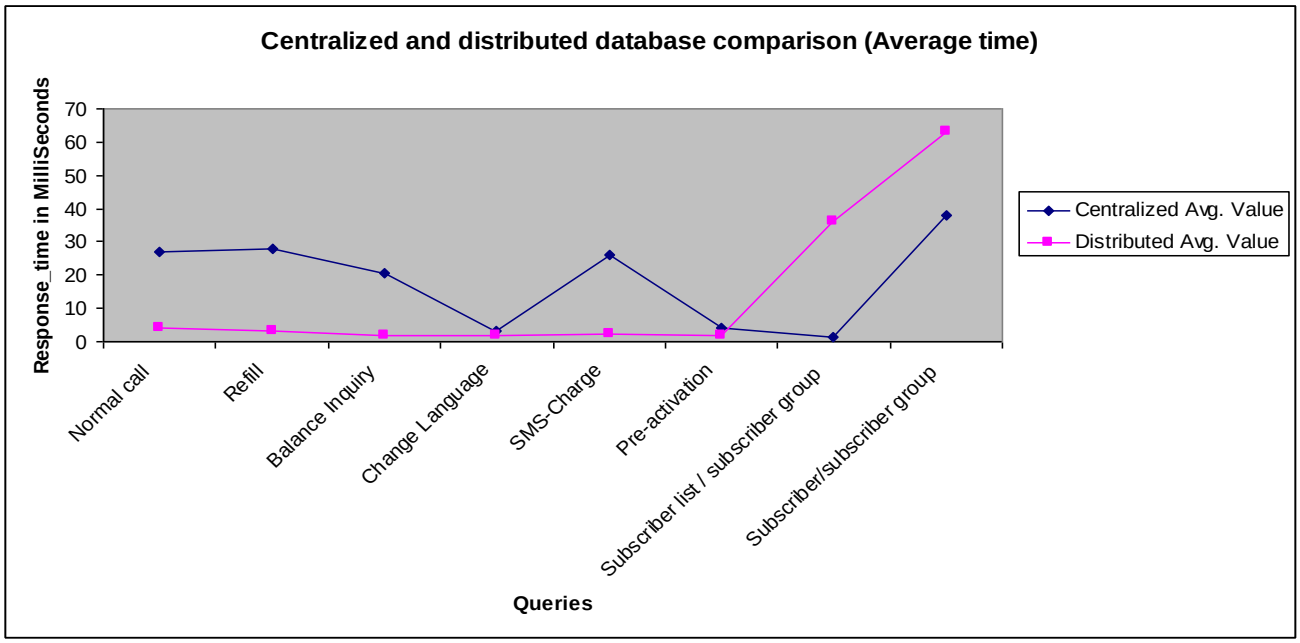


Figure 5

VIII. CONCLUSIONS

Our experiment showed that the average response time is decreased if we switch from centralized database to distributed database. In distribution we put the data to the site where it is used most frequently. This locality of data reduces the response time. Another major factor which reduces the response time is the index table. In the distributed database, data is fragmented. Since, these fragments are shorter as compared to the full relation, it makes searching fast.

However, when we need data from multiple sites for a query (report queries), the response time is increased. Accessing data from multiple remote sites and then joining those takes long time. The response time can be improved for such scenarios if we can make this search parallel. We also found out that access control, transaction management

and data allocation policy are very vital to further improve the response time.

IX. LIMITATIONS AND FUTURE WORK

As mentioned earlier that very little work is done for distributed databases in the telecom sector. So, to start the work especially concerning the telecom sector we started with a small experimental setup. The experiment was performed using 5 sites only and a small database. However, in a real environment a large number of sites are accessing the data at the same time and the database is very large. So, in future, experiments could be performed with large databases and increased number of sites. Other factors should also be considered like query optimization and concurrency.

APPENDIX A QUERIES

These queries are same for both centralized and distributed database environment. However in distributed databases we need to mention two more conditions as well i.e. location and tariff structure. Queries given below are for centralized database.

A. Normal Call

1. select s.subscriber_id, s.net_id, s.status, s.Subscriber_Param1, s.Subscriber_Param2,s.Subscriber_Param3, a.account_id, a. account_type, a.unit_type, a.units ,a.account_lower_limit,s.subscriber_group,s.tariff_ structure from subscriber s, account a where s.subscriber_id = a.subscriber_id and net_id = intial_net_id and a.account_type= account_type;
2. update account set units=new_units where account_id=acc_id;

B. Refill

1. select s.subscriber_id, s.status, a.account_id, a.account_type,a.units,a.unit_type from subscriber s, account a where s.subscriber_id= a.subscriber_id and net_id= intial_net_id and a.account_type = N_account_type;
2. update account set units=updated_units where account_id = acc_id;
3. Update subscriber set Subscriber_Param1 = subsc_id1 where subscriber_id= sub_id;
4. Update subscriber set Subscriber_Param2 = subsc_id2 where subscriber_id= sub_id;
5. Update subscriber set Subscriber_Param3 = subsc_id3 where subscriber_id= sub_id ;
2.

APPENDIX B TABLES

Account Table

C. *Balance Inquiry*

1. Sselect a.units,s.status,s.language,a.account_id,a.account_type,unit_type from subscriber s, account a where s.subscriber_id = a.subscriber_id and net_id = intial_net_id;

D. *SMS Charge*

1. Select s.subscriber_id s_Id,s.status,a.account_id a_Id,a.units, a.account_type a_type, s.Subscriber_Group sv_group from subscriber s , account a where s.subscriber_id= a.subscriber_id and s.net_id=intial_net_id and a.account_type= Sms_account_type;
2. Update account set units = actual_sms_units where account_id= account_id;

E. *Change Language*

1. Select subscriber_id S_id, language,Subscriber_Group Sv_group from subscriber where net_id= intial_net_id;
2. Update subscriber set language = lang_update where net_id = intial_net_id;

F. *Pre-Activation*

1. Select subscriber_id S_id, status, Announcement_Date a_Date from subscriber where net_id= intial_net_id;
2. Update subscriber set status =N_status, Announcement_Date=sysdate where net_id= intial_net_id;

G. *Subscriber list/ Subscriber group*

1. Select s.subscriber_id s_id,s.net_id n_id, s.Subscriber_Group Sv_group, a.account_type a_type, a.units, a.unit_type u_type from subscriber s, account a where s.subscriber_id= a.subscriber_id and a.units > 10;

H. *Subscriber/ Subscriber Group*

1. Select count (subscriber_id) sub_id , Subscriber_Group from subscriber group by Subscriber_Group";

I.Units/ Subscriber Group

1. Select s.Subscriber_Group sv_C_id , sum(a.units) units from subscriber s , account a where s.subscriber_id = a.subscriber_id group by Subscriber_Group";

Field Name	Data Type	Constraint	Description
Account_id	Number	Primary key	Unique Id for each account Value: 0-9999
Subscriber_id	Number	Foreign Key	Reference Id to subscriber
Account_type	Number	Not Null	Refers to call account, SMS account or bonus account etc
Units	Number	Not Null	Account balance gets updated due to debit or credit
Unit_Type	Number	Not Null	The charging unit type (SEK, Euro etc.)
Account_Lower_Limit	Number	Not Null	Minimum unit value allowed

Subscriber Table

Field Name	Data Type	Constraint	Description
Subscriber_id	Number	Primary key	Unique Id is used for each subscriber, random number is assigned when creating each subscriber
Net_id	Number	Unique	Unique Mobile number of subscriber
Tariff_structure	Number	Not Null	Tariff structure id used by subscriber group Value: 0-99
Subscriber_Group	Number	Foreign Key	Reference Id of subscriber group the subscriber belongs to
Status	Number	Not Null	Pre-activated or activated
Language	Number	Not Null	Language for subscriber this value is set when creating subscriber
Subscriber_Param1	Number	Null	Subscriber specific parameter
Subscriber_Param2	Number	Null	Subscriber specific parameter
Subscriber_Param3	Number	Null	Subscriber specific parameter

Announcement_Date	Number	Not Null	The last date when the special announcement was played to the subscriber
Location	Varchar	Not Null	Home location of the subscriber (e.g. Malmo, Vaxjo etc)

REFERENCES

[1] C. Mitchell "Components of a Distributed Database."
[2] Vaughn, "CSPC 343:A Sketch of Database History," 14 Nov,2003.
[3] A. Srivastava, U. Shankar, and S. K. Tiwari, "Transaction Management in Homogenous Distributed Real-Time Replicated Database Systems," *International Journal*, vol. 2, 2012.
[4] P. A. Bernstein, V. Hadzilacos, and N. Goodman, *Concurrency control and recovery in database systems*. Reading, Mass.: Addison-Wesley, 1987.
[5] R. Bhati, N. Bansal, and S. Jha, "Distributed Database System: The Current Features And Problems?."
[6] M. Alom, F.-A. Henskens, and M. Hannaford, "Optimization of Detected Deadlock Views of Distributed Database," in *Data Storage and Data Engineering (DSDE), 2010 International Conference on*, 2010, pp. 44-48.
[7] P. Lännhult and B. Lindblom, "Review of Non-Blocking Two-Phase Commit Protocols."
[8] K.-Y. Lam, T.-W. Kuo, W.-H. Tsang, and G. C. Law, "Concurrency control in mobile distributed real-time database systems," *Information Systems*, vol. 25, pp. 261-286, 2000.
[9] Y.-F. HUANG and J.-H. CHEN "Fragment Allocation in Distributed Database Design," ed, 2001.
[10] S. J. Miller "Maintaining Fully Replicated Distributed Databases in The Presence of Network or Node Failure and Repair," 1994.
[11] J. Maria Monteiro and Â. Brayner "Controlling Concurrency in Mobile ComputingEnvironments with Broadcast-Based Data Dissemination," ed, 2005.
[12] P. A. BERNSTEIN and N. GOODMAN (1981, Concurrency Control in Distributed Database Systems.
[13] A. Thomasian, "Distributed Optimistic Concurrency Control Methods for High-Performance Transaction Processing," *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING,*, vol. 10, 1998.
[14] A. B. Philip and G. Nathan "**Concurrency Control in Distributed Database Systems**," ***ACM Computing Surveys (CSUR)***, vol. 13, pp. 185-221, 1981.
[15] A. Bhardwaj, "Role of Fragmentation in Distributed database system," *International journal of networking and parallel computing*, vol. 1, pp. 6-8, 2012.
[16] M. T. Özsu and P. Valduriez, *Principles of distributed database systems*, 2. ed. Upper Saddle River, N.J.: Prentice Hall, 1999.
[17] M. T. Ozsu and P. Valduriez, "Distributed database systems: where are we now?," *Computer*, vol. 24, pp. 68-78, 1991.
[18] I. Ashraf, "Principles for Distributed Databases in Telecom Environment," 2010.
[19] A. Umar, "distributed database management systems issues and approaches ", ed, 1988.
[20] H. I. Abdalla, "A New Data Re-Allocation Model for Distributed Database Systems," *International Journal of Database Theory and ApplicationbVol*, vol. 5, pp. 45-60, 2012.
[21] N. Iacob, "Fragmentation and Data Allocation in the Distributed Environments," *Annals of the University of Craiova-Mathematics and Computer Science Series*, vol. 38, pp. 76-83, 2011.