THE BUCHAREST UNIVERSITY OF ECONOMIC STUDIES

DATABASE SYSTEMS JOURNAL

Vol. VII, Issue 2/2016

LISTED IN

RePEc, EBSCO, DOAJ, Open J-Gate,
Cabell's Directories of Publishing Opportunities,
Index Copernicus, Google Scholar,
Directory of Science, Cite Factor,
Electronic Journals Library

BUSINESS INTELLIGENCE

ERP

DATA MINING

DATA WAREHOUSE

DATABASE

ISSN: 2069 – 3230
dbjournal.ro

Contact
Calea Dorobanţilor, no. 15-17, room 2017, Bucharest, Romania
Web: http://dbjournal.ro
E-mail: editordbjournal@gmail.com; editor@dbjournal.ro

CONTENTS

A technique for n-way joins in wireless sensor networks 3
Djail BOUBEKEUR, Hidouci Walid KHALED, Loudini MALIK

eLearning Mobile App for Android and iOS "English Grammar Learn&Test" 10
Anca-Georgiana FODOR, Bogdan Vasile COVACI

A new approach to adaptive data models 19
Ion LUNGU, Andrei MIHALACHE

Big Data Mining: Challenges, Technologies, Tools and Applications 28
Asha M. PAWAR

Efficient Partitioning of Large Databases without Query Statistics 34
Shahidul Islam KHAN

A technique for n-way joins in wireless sensor networks

Djail BOUBEKEUR, Hidouci Walid KHALED, Loudini MALIK
LCSI Laboratory National High School for Computer Science (ESI) Algiers
b_djail@esi.dz, wk_hidouci@esi.dz, m_loudini@esi.dz

The join queries are operations that require more energy for their execution. In wireless sensor networks, energy is a determinant factor for the network survival. However, high energy consumption caused by such requests needs the implementation of very appropriate techniques for their execution. Research in this field considered especially binary joins. Few studies have addressed n-way joins. In this paper, we propose 'Nway Local Join', an energy-efficient technique for n-way join operations. We adopt an in-network execution at each step of the join operation. We compare our solution with an execution at the sink. NLJ shows the best performance for low selectivity factor.
Keywords: *wireless sensor networks, communication cost, in-network join, n-way join.*

1 Introduction

A wireless sensor network is composed of a very large number of inexpensive and low energy nodes, with processing and memorizing capabilities. The nodes are deployed in a region of interest; they use radio frequencies as communication channels to provide data collection and dissemination.

A sensor is considered as a data source which generates records at different reception actions. The nodes' records of the same type in the same network compose together a distributed table. The sensors network can be regarded as a distributed database system [1]. Collecting received data can be expressed as relational queries, such as selections, projections, union operations, aggregation, and joins.

Joins queries are widely used in wireless sensor networks' applications that require data gathering from multiple nodes.

However, these requests need excessive energy consumption, while energy in wireless sensor networks is a critical factor for the survival of sensors and of the network as well. In wireless sensors network, data transmission consumes more energy than data processing in the node [2]. It is then important to reduce the quantity of transmitted data during a join operation in order to decrease the energy consumed by the sensors. The techniques proposed recently such as Distributed-Broadcast of Coman and al. [3], Mediated Join of Pandit and Gupta [4], Synopsis Join of Yu and al [5], and INJECT of Min and al [6], have been limited to binary joins queries. N-way joins (joins between more than two tables) are rarely addressed.

N-way joins are of a remarkable interest in the networks of sensors. A practical example of the application of this joins type is the traffic control of the vehicles on trajectories made up of several (more than two) zones covered by wireless sensors. Monitoring bird's migrations, through several geographical areas, is another example of the application of this joins type.

This paper presents an energy-efficiency technique for n-way joins execution in wireless sensor networks. The rest of the paper is organized as follows: section 2 provides a description of the general characteristics of joins operations in wireless sensor networks. Section 3 gives related work. Section 4 describes our proposed technique. Section 5 contains the results of the simulation performed, and a discussion of the obtained results. Finally, section 6 concludes the paper.

2 General characteristics of joins in wireless sensor networks

2.1. Basic concepts.

A join of two tables L and R consists of the concatenation of all the tuples from table L

with those of table R where a condition (join predicate) is met on some attributes of the two tuples. When an arbitrary comparison operator (≥, <, =, ..) is allowed in join predicate, the join operation is called theta-join. An equijoin is a theta-join using only the equality operator.

In a wireless sensor network, a join is generally made between two regions of the sensors network. A region consists of a set of sensors of a geographically limited area.

2.2. Joins implementation in wireless sensor networks.

A join operation in the wireless sensors networks can be accomplished according to two implementations:

- External join.

It is the simplest implementation of join operations in wireless sensor networks. The join operation is performed in the base station (sink). This implementation consumes a lot of energy, because of a large amount of transmitted data.

- In-network join.

This is the most complex implementation, but it is more efficient. Join operation is executed by internal nodes of the network before transmitting final results to the base station.

2.3. Join types in wireless sensors network

According to spatial or temporal aspect, the following types of joins operations are distinguished:

- According to a spatial aspect.
  - ✓ Inter-region joins. They represent the joins without spatial predicates; relations are easily identified by assuming that each node knows its location [7].
  - ✓ Unique region joins. Contain spatial predicates. A node receiving a query of this type, can't, in general, determine if it should participate in the query without communicating with other nodes. This may result in a difficult implementation comparing with inter-region joins.
- According to the temporal aspect.
  - ✓ One execution joins. A fixed window is specified for each of the two relations.
  - ✓ Continuous joins. The relations use sliding windows or fixed windows defined for the future.
  - ✓ Periodic joins. Based on jump windows which define a time interval for repeating a query execution.

3. Related works.

Several techniques were proposed to execute join operations in wireless sensors network. They can be divided into two main categories: without filtering and with filtering techniques.

Techniques without filtering execute join operations between all tuples of tables or data streams. This causes a great consumption of energy, due to the high number of transmitted messages. It was the first proposed techniques [1] [4] [8] [9] [10] [11] [12] [13] .

Currently, the tendency for the performance of queries joins in wireless sensor networks, is to filter tuples before the join operation realization. The objective of the filtering techniques is to minimize the number of transmitted messages. These techniques were proposed in several works [3] [5] [6] [14] [15] [16] [17].

All those techniques were suggested for the binary join. Few those developed for nway join. Stern and al. in [18] have suggested a solution allowing processing every joins type in sensors networks (including n-way joins). This solution executes the joins at the base station (sink) in three steps:

i. All join attributes are collected at the base station, to filter some values.
ii. The determined join filter is broadcast in the network.
iii. Each site uses the filter to determine all the relevant tuples and send them to the sink, where the final join is processed.

This technique consumes too much energy because the number of tuples transmitted to the sink is very high.

4. N-way Local Join description.

We propose N-way Local Join (NLJ) to reduce efficiently the consumed energy for n-way joins. We process a one-shot inter-region join having syntax like this:

```
SELECT R1.attrs,
R2.attrs,…,Rn.attrs
FROM  R1, R2, …, Rn
WHERE pred(R1)  AND
pred(R2)  …  AND pred(Rn)
AND join-exp (R1.join-attrs,
R2.join-attrs,…,  Rn.join-attrs)
```

With:
Ri is the relation of an ith region.
pred (Ri) is a predicate of selection of relation Ri, and join-exp is the join condition.

An application example of these joins is the vehicle traffic control through many geographical zones:

```
SELECT Veh1.VehId, Veh1.time,
Veh2.time, Veh3.time
FROM       Veh1, Veh2, Veh3
WHERE (Veh1.time IN i1) and
(Veh2.time in i2) and
(Veh3.time in i3) and
(Veh1.VehId = Veh2. Veh Id)
and (Veh2. Veh Id= Veh3. Veh
Id)
```

Where: i1, i2, and i3 indicate time intervals during which vehicles passed respectively through regions 1,2 and 3.
N-way Local Join (NLJ) consists of executing each join operation at a local node in the network. We also adapt the technique of left linear trees to minimize the number of n-way joins [19], and we consider geographical zone positions of participating zones to determine the execution order of joins (from the nearest to the farthest zone).
By NLJ, a join operation runs in three phases:

Phase 1.

The query is transmitted from the sink to the root node of each area, using a location routing protocol GPSR [20] [21], to ensure the arrival of the query message to the concerned regions. Each region consists of a tree of nodes. Each node must transmit its tuples to the root. We assume that each node knows its location and the locations of its neighbors, via GPS or via localization algorithms [22].

Phase 2.

The join operation is performed with the in-network principle. We also adopt the principle of the left linear trees to determine the execution order of joins. The joins are performed in pairs (Fig. 1). Where an unwinding in two steps is done for each pair of relation R_i and R_{i+1} performing a join operation:

i. The relation R_i is transmitted to the relation R_{i+1} zone.
ii. The join operation is executed in the R_{i+1} region.

Finally, the final result is transmitted to the sink.

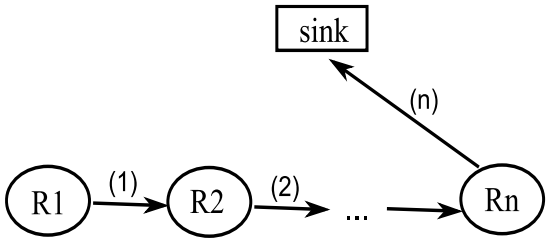

Fig. 1 -N-way local Join (NLJ) execution.

The principle of NLJ will be illustrated with an example. For this purpose, a join between 3 tables is considered.

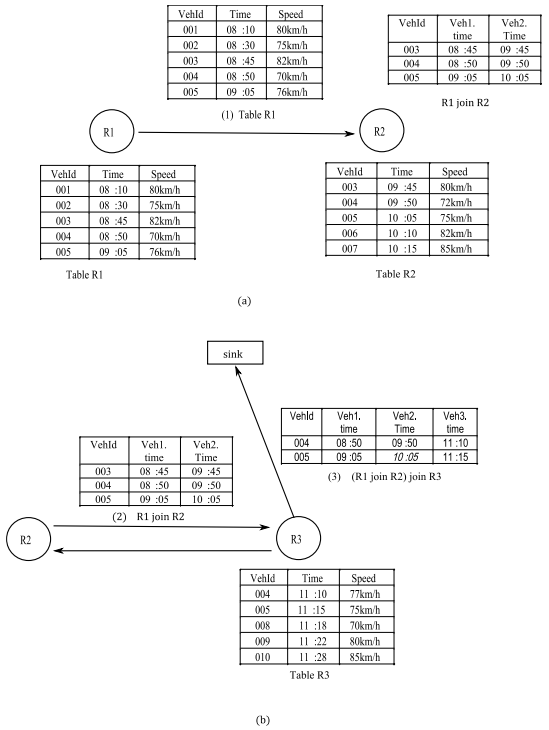

Fig. 2 - An example for N-way local Join (NLJ) execution.

The principle of NLJ will be illustrated with an example. For this purpose, a join between three tables is considered.

In a first step, the join is carried out between the relations R_1 and the relation of the nearest area (R_2). The table R_1 is transmitted to the R_2 area so that the join result between R_1 and R_2 is given (Fig 2 (a)).

Similarly, the join result will be obtained, at the second step, between the table R_3 and the join result of R_1 and R_2 (Fig 2 (b)).

At the last step, the result determined in the R_3 area is sent to the sink (Fig 2 (b)).

5. Performance analysis

5.1. Experimentation environment

To simulate the n-way join execution, we used the NS3 simulator. NLJ has been applied on static tables of 2000 tuples each. The message size is a tuple size, which is 40 bytes each. The column size is assumed to 10 bytes; the size of the result tuple is 30 bytes.

The communication cost was tested considering selectivity factors of intermediates joins in the interval $[10^{-5}, 10^{-4}]$, and then in the interval $[10^{-4}, 10^{-3}]$. Selectivity factors of intermediates joins are generated in a random way. The values of the horizontal axis of results graphs represent averages of the intermediates joins' selectivity factors.

We process separately two simulations:

- 3 tables simulation,
- 5 tables simulation.

For each simulation, we perform 2 cases:

- the extern join simulation,
- and our technique (NLJ) simulation.

5.2. Experimentation results

In the interval $[10^{-5}, 10^{-4}]$ of selectivity factors, (Fig 3 and Fig 4), N-way Local Join perform better than extern join in both simulations: 3 tables and 5 tables. In this interval, selectivity factors are very low.

In the interval $[10^{-4}, 10^{-3}]$ of selectivity factors, (Fig 5 and Fig 6), N-way local join continues to offer the best performance, but not throughout the entire interval. With the high values of selectivity factors, NLJ decrease in performance in favor of extern-join technique.

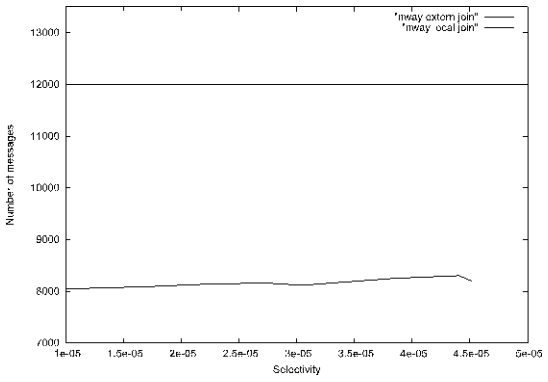

Fig 3. Communication cost for 3 tables following the average of intermediates joins' selectivity factors in the interval $[10^{-5}, 10^{-4}]$

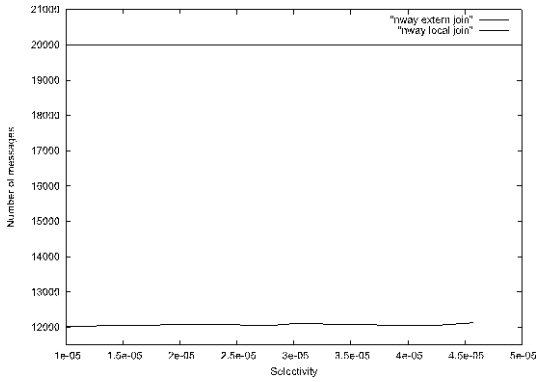

Fig 4. Communication cost for 5 tables following the average of intermediates joins' selectivity factors in the interval $[10^{-5}, 10^{-4}]$

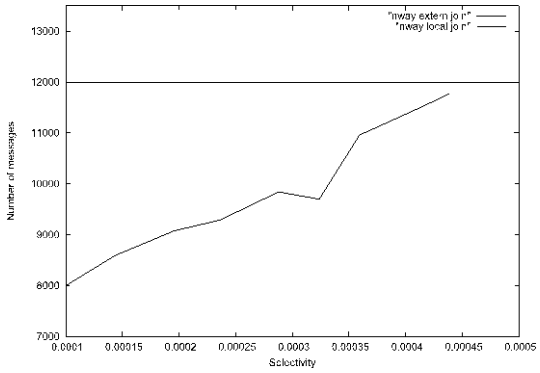

Fig 5. Communication cost for 3 tables following the average of intermediates joins' selectivity factors in the interval $[10^{-4}, 10^{-3}]$

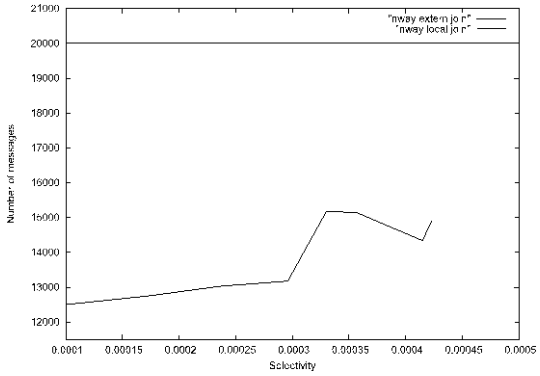

Fig 6. Communication cost for 5 tables following the average of intermediates joins' selectivity factors in the interval $[10^{-4}, 10^{-3}]$

5.3. Discussion

Because join operations are executed at sensors nodes during steps of the join execution, our approach offers the best performance.

Extern-join performs the join operation at the sink. This engenders high energy consumption, due to an important transmitted information volume.

It is important to note that these results are valid for low selectivity values.

For high selectivity values, it is ideal to realize the join query at the sink.

6. Conclusion

We have presented, in this paper, a technique for n-way join execution, in wireless sensor networks. We have adopted the principle of in-network execution to reduce the number of tuples transmitted between sensors and to sink.

Our approach shows better performance compared to extern-join principle for low selectivity factor.

In future work, we plan to a study joins operations between data streams generated by the sensors of each region. The techniques suggested in the case for distributed databases were proposed in [23] and can be adapted to the cases of the databases of sensors network.

References

[1] Y. Yao and J. Gehrke, "Query Processing in Sensor Networks," in *CIDR*, 2003, pp. 233-244.

[2] F. Zhao and L. J. Guibas, *Wireless sensor networks: an information processing approach*: Morgan Kaufmann, 2004.

[3] A. Coman, M. A. Nascimento, and J. Sander, "On join location in sensor networks," in *2007 International Conference on Mobile Data Management*, 2007, pp. 190-197.

[4] A. Pandit and H. Gupta, "Communication-efficient implementation of range-joins in sensor networks," in *International Conference*

on Database Systems for Advanced Applications, 2006, pp. 859-869.

[5] H. Yu, E.-P. Lim, and J. Zhang, "On in-network synopsis join processing for sensor networks," in *7th International Conference on Mobile Data Management (MDM'06)*, 2006, pp. 32-32.

[6] J.-K. Min, H. Yang, and C.-W. Chung, "Cost based in-network join strategy in tree routing sensor networks," *Information Sciences*, vol. 181, pp. 3443-3458, 2011.

[7] H. Kang, "In-network processing of joins in wireless sensor networks," *Sensors*, vol. 13, pp. 3358-3393, 2013.

[8] B. J. Bonfils and P. Bonnet, "Adaptive and decentralized operator placement for in-network query processing," *Telecommunication Systems*, vol. 26, pp. 389-409, 2004.

[9] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "The design of an acquisitional query processor for sensor networks," in *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, 2003, pp. 491-502.

[10] D. J. Abadi, S. Madden, and W. Lindner, "Reed: Robust, efficient filtering and event detection in sensor networks," in *Proceedings of the 31st international conference on Very large data bases*, 2005, pp. 769-780.

[11] V. Chowdhary and H. Gupta, "Communication-efficient implementation of join in sensor networks," in *International Conference on Database Systems for Advanced Applications*, 2005, pp. 447-460.

[12] S. R. Mihaylov, M. Jacob, Z. G. Ives, and S. Guha, "A substrate for in-network sensor data integration," in *Proceedings of the 5th workshop on Data management for sensor networks*, 2008, pp. 35-41.

[13] S. R. Mihaylov, M. Jacob, Z. G. Ives, and S. Guha, "Dynamic join optimization in multi-hop wireless sensor networks," *Proceedings of the VLDB Endowment*, vol. 3, pp. 1279-1290, 2010.

[14] X. Yang, H. B. Lim, T. M. Özsu, and K. L. Tan, "In-network execution of monitoring queries in sensor networks," in *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, 2007, pp. 521-532.

[15] M. Stern, K. Böhm, and E. Buchmann, "Processing continuous join queries in sensor networks: a filtering approach," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, 2010, pp. 267-278.

[16] Y.-X. Lai, Y.-L. Chen, and H. Chen, "PEJA: Progressive energy-efficient join processing for sensor networks," *Journal of Computer Science and Technology*, vol. 23, pp. 957-972, 2008.

[17] Y. Lai, Z. Lin, and X. Gao, "SRJA: Iceberg Join Processing in Wireless Sensor Networks," in *2010 2nd International Workshop on Database Technology and Applications*, 2010, pp. 1-4.

[18] M. Stern, E. Buchmann, and K. Böhm, "Towards efficient processing of general-purpose joins in sensor networks," in *2009 IEEE 25th International Conference on Data Engineering*, 2009, pp. 126-137.

[19] M. Steinbrunn, G. Moerkotte, and A. Kemper, *Optimizing join orders*: Citeseer, 1993.

[20] B. Karp and H.-T. Kung, "GPSR: Greedy perimeter stateless routing for wireless networks," in *Proceedings of the 6th annual international conference on Mobile computing and networking*, 2000, pp. 243-254.

[21] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker, "GHT: a geographic hash table for data-centric storage," in *Proceedings of the 1st ACM international workshop on*

Wireless sensor networks and applications, 2002, pp. 78-87.

[22] A. Savvides, M. Srivastava, L. Girod, and D. Estrin, "Localization in sensor networks," in *Wireless sensor networks*, ed: Springer, 2004, pp. 327-349.

[23] T. M. Tran and B. S. Lee, "Distributed stream join query processing with semijoins," *Distributed and Parallel Databases*, vol. 27, pp. 211-254, 2010.

eLearning Mobile App for Android and Ios "English Grammar Learn&Test"

Anca-Georgiana FODOR, Bogdan Vasile COVACI
LABsterzz.com
anca@labsterzz.com, bogdan@labsterzz.com

This article is aiming to present the architecture and few elements from the developing cycle of "English Grammar Learn &Test" app. This is an e-learning tool for people who want to improve their English Grammar and Vocabulary. The app was approved by Google Play and Apple Store and it is available for free on both platforms as following:
Android: https://play.google.com/store/apps/details?id=com.labsterzz.english_tests
iOS: https://itunes.apple.com/us/app/english-grammar-learn-test/id1126468980
The app already reached350.000 users, it is rated at 4.43out of maximum 5.0 in Google Play Store. Since mid-June 2016, we launched the app also in the Apple Store iOS devices.
***Keywords**: e-learning, m-Learning,English Grammar Learn & Test, Android Mobile Apps, IOS Mobile Apps, SQLite, Eclipse, XCode, Java, Swift.*

1 Introduction

The scope of launching mobile e-learning App for Android and iOS systems "*English Grammar Learn & Test*" was to help users across the world improving their English skills enhancing their vocabulary and practicing grammar.
According to [3], student reactions to grammar-focused lessons seem to be typically one of three kinds. Some students find grammar very appealing, some find it intrinsically boring, and some find it useful but really hard to comprehend. In this context, making grammar appealing for students might intrigue their curiosity and turn them back to the English learning table.
Bearing in mind this goal, we developed a simple, enjoyable and useful app for Android and iOS devices - smartphones and tablets –a kind of pocket book edition.
The concept is designed to create a daily 5 minutes education window wherever the student is: in the school-bus, waiting for the tube, in the park, etc., since the smartphone is their pocket-ready educational tool.
As content, "*English Grammar Test & Learning"* app offers 120 short grammar and vocabulary lessons, more than 1000 grammar questions, 20 tests for practice (each containing 30 random questions), Evolution Table and the wrong answers list after each test.
The app is free, ready to be used and available at the below addresses:

- Android: https://play.google.com/store/apps/details?id=com.labsterzz.english_tests
- iOS: https://itunes.apple.com/us/app/english-grammar-learn-test/id1126468980

2. E-learning world

Nowadays, during the Information Era, people are traveling a lot, are trying to optimize their time, space and life.
In the last years, we crossed over from reading paper based books in nice classical libraries to electronic learning and reading in the bus, tube or plain. We talk about big amount of information which is flying around such as Big Data which comes to the forefront in e-Learning soon.
The trend in commercial companies and education institutions [1] for example is moving very fast to e-learning, as this approach is opening a lot of opportunities for staff and students development with a proper balance between expenses (less travel costs, less time) and resources. This

will engage more learners to online education.

We hear more and more about *m-Learning* which follows the boost of mobile device development. People prefer small computing gadgets over relatively bulky laptops mainly for their portability. For this reason, m-Learning started to be a trend and we expect to rapidly evolve during next years.

As expectation [11], new apps with more expanded features in online education will flood the market in the near future. As effect, they will give access to e-Learning to much more people than before and make e-Learning more widespread as people will be able to make use of it on the go. Especially because the time seems to compress and people are traveling more and more and their natural tendency is to use the time in their benefit.

In addition, Social Networks shouldn't be neglected. This is an opportunity of live communication between people. Social Networks can become a strong amplifier as they are a means of direct communication with users and potential users.

According to [4], one of the best examples of this trend is Twitter. This social media network has been ranked # 1 in the list of Top 100 Tools for Learning for 7 years in a row since 2009.

This trend will lead to the increase of education level of the people, due to m-Learning accessibility. There are millions of apps already in Google Play and Apple App Store ready to be downloaded and used. It's true that a part of them are games, however even *gamification* which is a trend as well can be used for education. Nothing stimulates learners better than challenges, points, badges and leader boards. Number of educational apps developed as games is rapidly growing and is projected to double its quantity in the near future.

We expect that 2017 will show us a great rise of *gamification* and this will give a foundation for its implementation in augmented learning.

As a short conclusion, e-Learning, the actual trend in education which is growing, gives time a new scale enlightens travel, saves costs and gives the chance for better education to much more people.

In a specific case, m-Learning is becoming much stronger and it is a clear trend for near future because is coming with an additional value: portability and availability via mobile apps ready to be used on your smartphone or tablet.

3. Application architecture

"*English Grammar Learn & Test*" is three-tier architecture for both versions, Android and iOS, as it is presented in Figure 1.

Despite the similar architecture, the two versions of the app are developed with different tools, each of them with its specificities of development environment, types of scalability, database manipulation and programming language used.

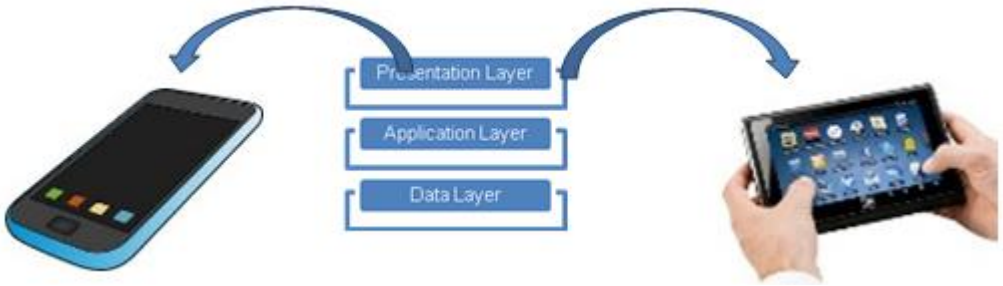

Fig. 1"*English Grammar Learn &Test*" app architecture

4. Data Layer

Data Layer was built using SQLite which is a relational database management system contained in a C programming library, according to [8].

In contrast to many other database management systems, SQLite is not a client–server database engine. Rather, it is embedded into the end program, as part of the application program–Eclipse [5] for Android operating system and Xcode [6] for iOS operating system – on our specific case.

Due to the server-less design, SQLite applications require less configuration than client-server databases. SQLite [8] is called zero-configuration because it does not require service management. This is a clear advantage in database usage as SQLite engine has no standalone processes which the application program communicates with. Instead, the SQLite library is linked n and thus becomes an integrated part of the app program.

For "*English Grammar Learn&Test*" app's data layer, the most important Table is QUESTIONS_LIST - which stores the questions for tests, together with answers options and the correct answer as well. This is created using the sequence of instructions (*Android version* sample): we will start by creating a database called "questionsDB" and set a version number as in the example below. The reason for assigning a version number for the database is to make SQLite aware when we want to update the content of the database.

```
public class E_1_DbHelper extends
SQLiteOpenHelper {

public static final String DB_NAME =
"questionsDB";
public static final int DB_VERSION =
1;
```

Afterwards we will declare the creation of QUESTIONS_LIST table. As it can be seen we use simple SQL language for the CREATE statement.

```
public static final String TABLE =
"QUESTIONS_LIST";
public static final String C_ID =
BaseColumns._ID;
public static final String C_Q = "Q";
public static final String C_R1= "R1";
public static final String C_R2= "R2";
public static final String C_R3= "R3";
public static final String C_R4= "R4";
public static final String C_RC= "RC";
public static final String C_CATEG=
"Q_CATEG";
public static final String C_SHOW=
"Q_SHOWN";
public static final String
CREATE_TABLE = "CREATE TABLE " + TABLE
+ " ("
+ C_ID + " INTEGER PRIMARY KEY
AUTOINCREMENT, "
+ C_Q+ " VARCHAR(255), "
+ C_R1 + " VARCHAR, "
+ C_R2 + " VARCHAR, "
+ C_R3 + " VARCHAR, "
+ C_R4 + " VARCHAR, "
+ C_RC + " VARCHAR, "
+ C_CATEG + " VARCHAR, "
+ C_SHOW + " INTEGER);";
```

Then the database is populated with questions (one sample below):

```
db.execSQL("INSERT INTO " +TABLE+"
(q,r1,r2,r3,r4,rc,q_categ)
values ('In ten years' time, all
students _____ their own computers in
school.','will have','had
','have','have been','will
have','Future');");
```

Two functions are called when accessing a database: onCreate and onUpgrade:

- onCreate runs only once if it does not detects a database installed on the device associated with the app (or if there is already a table in the targeted Schema with the same name) and will execute all the SQL statements included in its boundary. (E.g. create table, insert into etc.).
- onUpgrade will run every time when the database version is changed into a higher value (e.g. new value 2 instead of 1).

As there will be cases when you need to update your database with new content it is a good practice to put all DROP statements inside onUpgrade func and add your new content inside the onCreate func.

In addition to QUESTIONS_LIST table, the app uses few more tables to manage the tests done, results obtained, wrong answers list to, Evolution Table were we store the users progress, lessons attended, etc.

For *iOS version*, we used the SQL Lite Database package [6], already built in

Xcode platform, working in a similar manner at conceptual level, with some platform specificities such additional steps required to be done prior (e.g. database needs to be created in a console mode – presented in the Fig. 2).

Fig. 2. SQLite for IOS console

5. Application Layer

Application layer is dependent on the operating system platform. "*English Grammar Test & Learning*" was built on Eclipse platform for *Android* version by using the already available SQLite Database package, called from the main module of the application. As main elements used for *Android version* of the application there are packages to access SQLite Database:

```
package com.xxx.english_tests;
import
android.database.sqlite.SQLiteDatabase
;
import
android.database.sqlite.SQLiteOpenHelp
er;
```

To open and close the connection to database there are used bellow functions:

```
public void open()
{
       this.getWritableDatabase();
       // return this;
}

public void close()
{
       this.close();
}
```

Main code elements of the *IOS Version* of the app is presented below:

```
import UIKit
import SQLite
import CoreData

public class dbHelper:
UIViewController {

// opens the database
// we will use the returning
value  (db)  in our sql statements.

class func openDatabase() ->
COpaquePointer
{
 var db: COpaquePointer = nil

 let dbLocation
=  AppDelegate.copyBundledSQLiteDB()

if sqlite3_open(dbLocation, &db) ==
SQLITE_OK {
      return db
        }
else {
      print("Unable to open database.
Verify path")
      return db
        }
    }

// example of a SQL count statement
// read that we need to transform the
outcome to a Int32

class func
queryCountFromQuestiondDB()->Int32{
   let db = openDatabase()
```

```
let queryStatementString = "SELECT
count(*) FROM questionsDB;"
   var queryStatement: COpaquePointer
= nil

   sqlite3_prepare_v2(db,queryStatemen
tString, -1, &queryStatement, nil)
   sqlite3_step(queryStatement)

// declare a constant (let) to link it
to the result // 0 is the column
index. In swift we count from 0
   let countResult =
sqlite3_column_int(queryStatement, 0)

// finalize the sql statement
   sqlite3_finalize(queryStatement)

   return countResult
    }


//classic "select * from …" statement

class func queryList30questions()-
>NSArray{
    let db = openDatabase()
    let queryStatementString = "SELECT
* FROM questionsDB;"
    var queryStatement: COpaquePointer
= nil

//declare anempty array to throw the
data in
    var queryResult: [String] = [""]

    sqlite3_prepare_v2(db,
queryStatementString, -1,
&queryStatement, nil)

//sqlite will fetch one row at a time
so we need to introduce a while clause
// ==SQLITE_ROW … will make sqlite
read records until it reaches an empty
row
    while
(sqlite3_step(queryStatement) ==
SQLITE_ROW) {

// as in the example above create
constants to link them to the query
result
     let Q =
String.fromCString(UnsafePointer<CChar
>(sqlite3_column_text(queryStatement,
1)))!
     let R1 =
String.fromCString(UnsafePointer<CChar
>(sqlite3_column_text(queryStatement,
2)))!
     let R2 =
String.fromCString(UnsafePointer<CChar
>(sqlite3_column_text(queryStatement,
3)))!
     let R3 =
String.fromCString(UnsafePointer<CChar
>(sqlite3_column_text(queryStatement,
4)))!
     let R4 =
String.fromCString(UnsafePointer<CChar
>(sqlite3_column_text(queryStatement,
5)))!
     let RC =
String.fromCString(UnsafePointer<CChar
>(sqlite3_column_text(queryStatement,
6)))!
     let QCATEG =
String.fromCString(UnsafePointer<CChar
>(sqlite3_column_text(queryStatement,
7)))!

// append in the earlier created array
the query result.
     queryResult.insert(Q+"|"+R1+"|"+R
2+"|"+R3+"|"+R4+"|"+RC+"|"+QCATEG,
atIndex: 0)
        }
    sqlite3_finalize(queryStatement)
    sqlite3_close(queryStatement)
    sqlite3_close(db)

    return queryResult
    }
}
```

On iOS, SQLite does not require assigning a version number to the database.
When opening a database on iOS, SQLite will check if the database exists on the target device and if not it will copy the database created above.
This can be a problem if we need to update our database with new content.
In our case in order to overcome this we have split the database in two with one database containing the QUESTION_LIST table and one containing all user saved data (e.g. lessons attended, test scores, medals).
In case we will update the QUESTION_LIST with a new set of question we will create a new database containing the new QUESTION_LIST table.
Because we cannot send to the users a new version of our database without updating the app itself on the AppStore and in this way changing the version number of the app we will make use of this new app version to check what version the user is currently using inside the `openDatabase()` function explained above.
All English lessons are designed in a static manner, as no interactivity is required from the app users. The only database activity in

this area is related to status of each lesson, read or not-read by the user and the Summary presented per lessons category.
We also used functions to manage the tests done by local user and unlock the next test depending on accomplishments, evolution table management, wrong answer list, etc.

6. Presentation Layer

The app interface for users is the same for both Android and iOS versions. Below, in Figure 3, there are few samples of the app interfaces:

Fig.3 *English Grammar Learn & Test* app presentation layer samples

The most important part of the Presentation Layer, in this specific case of Mobile app is to insure the scalability for the entire range of devices (smartphones and tablets). As a solution for this part, we treated separately smartphones and their standards.

Basically you will need to ensure that your app will look the same no matter what device the user is using.

For this we used the built-in simulators provided by Xcode for iOS and Eclipse for Android.

If for iOS this can be an easy task as there are no more than 6-7 devices with different height and width for Android it can be very time consuming since today there are more than 10,000 Android devices on the market with tens of combinations of height and width.

One tip from us is to go to [9] and structure the devices with the biggest market share on groups of same aspect ratio and scale the screen to accommodate them.

7. Conclusions

"*English Grammar Learn & Test*" is a **simple**, user - **friendly** and appreciated app.

English Grammar Test & Learning

Google Statistics today:

- Downloads: 350.000
- No of last month sessions: 170.000
- No of Ratings: 6.000
- Rating: 4.4 (maximum is 5 and competition average is 4.1)

According to [4] in 2017 we expect to see e-learning becoming even more individual, well-timed, and technologically advanced.

In this respect *m-Learning* is earning more and more space into our educational life. The apps offer is growing fast in Google Play and Apple App Store, users are becoming selective and they are expressing their needs while choosing the most appropriate apps for them.

According to our user's feedback on Google Play platform, the app comes handy for Grammar practice, for Vocabulary development, easy to use and intuitive.

User's feedback is a very important aspect we look after as it shows us their needs and desires so we treat it very seriously.

The app is built on three-tier architecture and it is available for both Android and IOS platforms with the same presentation Layer, scalable for all types and standards of mobile devices. Differences between versions are derived from different

platforms, different programming language, database usage and calls from source code.

Scalability is as important as the user's feedback because the user has to see the app proper scaled on his own smartphone as well and in the same manner as on a tablet device, otherwise any app, no matter how life changing will be can be underestimated since the beginning.

From the content point of view, "*English Grammar Learn & Test"* app offers 120 short grammar and vocabulary lessons, more than 1000 grammar questions, 20 tests to practice (each containing 30 random questions), an Evolution Table and wrong answers after each test. The content is extremely important, to keep the user interested in making him use your app on a daily basis.

Apart from the app development cycle we will talk about entering the mobile apps market. [10]There is no other better reward for the developer than knowing his own app is part of the daily activities of users across the world. To achieve this, we tracked our app parameters on a daily basis: downloads per day, user retention rate, number of ratings and reviews received per day, average time spent by users within our app, number of sessions in a specific time period, etc. and compared them with our peers (were available). The commercial part requests a lot of time, discipline, energy and commitment.

We are also constantly monitoring the feedback our users are sending and we make sure they get a response within 1 working day.

According to [10], a marketing study focused on Mobile Apps, there are a number of interesting conclusions regarding user's behaviour:

- The average app user has 36 apps installed on his or her smartphone.
- most installed apps are not used often – 26% those apps are used daily, while 1 in 4 apps are never used;
- App discovery can occurs out of the app store–52 % of users are aware of apps from friends, family, and colleagues and 24% discover an app through it's company website, then having a website in place before you launch your app is a good idea;
- 2 out of 3 users consider the average ratings of an app as an important factor when deciding to download and the same number consider the app description an important factor. Your app description should tell the user what to expect when downloading your app as clear and simple as possible.
- 3 out of 4 users expect apps to be free but the willingness to pay is $2.17
- Making users life easier and always having new content are two of the top attributes associated with highly used apps. A good practice is to launch a new update once 3-4 weeks.

Developing, launching and keeping alive a mobile app is a balance between technology, marketing, dynamism and effort to understand and adapt to users' needs [3].

According to statistics [7] and based on our experience, the hard work only starts after the app is launched.

Acknowledgment

This paper presents the architecture and few elements from the developing cycle of "*English Grammar Learn & Test*" app, an educational tool for users across the world who want to improve their English Grammar and Vocabulary. This app was approved by Google Play and Apple Store and is available to download for free on both platforms.

References:

[1]. Velicanu Anda, Lungu Ion, Diaconita Vlad, Codrin-Florentin Nisioiu - *Cloud E-learning*, Volume of 9th International Conference"E-learning and software for education", 2013

[2]. Catalin Ionut Silvestru, Constantin Marian Matei, Codrin Nisioiu, Dragos Stefan Silvestru - *The E-Learning Systems and Platforms Role and*

Involvement in New Economy, Volume of International Conference " Knowledge Management – Projects, Systems and Technologies, 2008
[3]. http://www.onestopenglish.com/methodology/methodology/grammar-vocabulary-and-skills/grammar-and-vocabulary-seven-ways-to-help-students-enjoy-grammar/146459.article
[4]. Masolova Elena – "7 e-learning trends to keep an eye on it in 2016", on-line article https://trainingmag.com/7-e-learning-trends-keep-eye-2016
[5]. https://eclipse.org/downloads/
[6]. http://xcode-mac.en.softonic.com/mac
[7]. https://en.wikipedia.org/wiki/SQLite
[8]. http://sqllite.com/
[9]. https://design.google.com/devices/
[10]. Mobile Application Marketing Insights https://think.storage.googleapis.com/docs/mobile-app-marketing-insights.pdf
[11]. http://www.ispringsolutions.com/blog/top-e-learning-trends-in-2016-follow-or-ignore/

Anca-Georgiana FODOR graduated from the Faculty of Cybernetics, Statistics and Economic Informatics of the Academy of Economic Studies in 1998. She got the title of doctor in economy in the specialty economic informatics in 2008. At present she is working for Banca Comerciala Romana.

Bogdan Vasile COVACI graduated from the Faculty of Economics of the Dimitrie Cantemir University in 2008. At present he is attending Master courses at the Academy of Economic Studies (Databases – Support for Business) and he is working for Banca Comerciala Romana.
Together with Anca-Georgiana FODOR, they started to develop Mobile Apps as LABsterzz in 2014.

A new approach to adaptive data models

Ion LUNGU, Andrei MIHALACHE
Faculty of Cybernetics, Statistics and Economic Informatics,
Academy of Economic Studies, Bucharest, Romania
ion.lungu@ie.ase.ro , andrei@mdata.ro

Over the last decade, there has been a substantial increase in the volume and complexity of data we collect, store and process. We are now aware of the increasing demand for real time data processing in every continuous business process that evolves within the organization. We witness a shift from a traditional static data approach to a more adaptive model approach. This article aims to extend understanding in the field of data models used in information systems by examining how an adaptive data model approach for managing business processes can help organizations accommodate on the fly and build dynamic capabilities to react in a dynamic environment.
Keywords: *adaptive data model, dynamic capabilities, data models*

1 Introduction

Data dominates every information system and if data structures are properly chosen and organized things go well, any algorithm is almost always understood by itself to be optimally built to perform. Algorithms are essential in programming, but data models will always fill the central place [1].

The business environment is the sum of all those factors which are available outside the business and over which the business has no control. Some of these factors can include objects such as: clients, suppliers, competitive companies, investors and owners, improvements in technology, laws and government activities, market, social and economic trends.

2 Data models

A *model* can be defined as a simplified abstraction of a complex reality, highlighting the essentials and ignoring the details.

Data modeling is a method used to define and analyze data requirements needed in business processes deployed in companies. Software applications store data in order to use them in the future. When data is saved, most of the times a relational database is chosen due to performance and accessibility (data is understandable). The *data* term refers to facts that characterize objects or events that can be recorded and stored in a computer system and it has significance and meaning for users.

Data governance is a set of processes that ensures that important data assets are formally managed throughout the enterprise and take into account data definition and data integrity constraints in the data model [2].

3 Data model types

The 1975 ANSI/SPARC data architecture study group divided database-centric systems into three models:

- *Internal model*: describes the logical data structures and may contain logical descriptors of the collections, attributes, XML markers, etc. These models are represented in accordance with the requirements of a particular technology implementation using flowcharts.
- *Conceptual model:* represents the scope and semantics of the classes designating entities of interest to the study area and assertions about associations between these entities. They are represented by the conceptual diagram.
- *External model*: defines how data is stored. These models contain

partitions, tablespaces, indexes etc. and it is represented through the physical schema.

Looking back to the late 1960s, IBM launched the *hierarchical model*, together with data manipulation language DL/1. Hierarchical database systems organize data as a collection of trees. All recordings have an owner or root (one and only one), and thus all other records have a single parent.

To locate a certain record, you must travel the path from root parent of the tree to the level where the desired child is located. Access to data in hierarchical databases is achieved by low-level calls that programmers write to sail records from the root towards the leaves of interest. Therefore, the programmer must know the physical representation of the database.

This data model can be used in systems containing data that can be organized hierarchically, without compromising the information (e.g.: **Fig. 1**).

Hierarchical databases support two means of representing information: specific records containing data and records containing the type of parent-child links that defines the relationship 1:N between one parent and N child records.

This approach has major limitations due to restrictions of data representation. The data structures which are not represented hierarchy by default is difficult, if not artificially structured in such database.

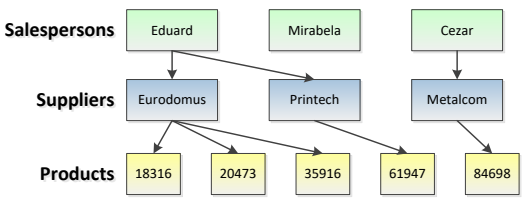

Fig. 1. Example of hierarchical data model

The *network model* was originally created by Bachman for General Electric, where he developed the first commercial database system (IDS - Integrated Data Store) in 1964.

Data modeling in a network database is different from the equivalent hierarchical approach. Networked databases arrange data in a directed graph and use a standard navigation language.

The model has brought new opportunity to move the access from a specific point of a set of data directly to another record in another data set.

Network databases provide an effective pathway to access data and are capable of representing any data structure containing simple types (such as integers, real characters and strings). This is achieved by using different types of mapping mechanisms known as *sets*.

A *set* is a container of pointers identifying the type of data set that can be accessed from the current record. These sets are defined by standard CODASYL: sets of system (single), multimembers or recursive. Using these sets, database designers and programmers can represent and navigate relationships 1:1, 1:N and N:M.

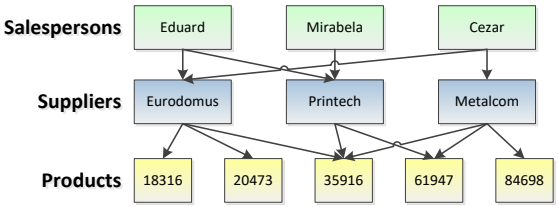

Fig. 2. Example of network data model

To access the data, the programmer must know the physical representation of data and use a low-level navigation language.

This approach to system database is more flexible than the hierarchical model, but the programmer must know the physical representation of data you can access them. Therefore applications using a network database need to be altered with every change in the database structure.

The *relational model* provides a different approach to data storage and it was first represented by Edgar F. Codd in 1970 [3]. In a relational database, all data are represented as simple tabular data structure

(relationships) that are accessed using a high-level non-procedural language.
This language is used to achieve the desired relationships and datasets. Thus, the physical implementation of the database is hidden, and the programmer does not need to know the physical implementation to access the data. In 1974 the name was proposed SEQUEL for high-level non-procedural language, which was later changed to SQL. In 1986 ANSI committee X3H2 accepted as standard ANSI SQL language.

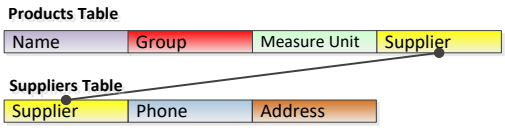

Fig. 3. Example on an instance of relational data model

Relational databases connect different data files by using key fields or some common data (e.g.: **Fig. 3**). Records are stored in different tables or files that are composed of rows and columns. In the databases terminology, tables are called "relations", the rows are called "tuples" and the columns are called "attributes".
The advantage of a *Relational Database Management System* (*RDBMS*) is that users and programmers do not have to know the data structures or pointers. Tables and rows are much more comprehensible than pointers and pointers that point to records.
The downside is that some orders retrieval requires more processing time compared to other database models.
SQL (*Structured Query Language*) is a language based on the declarative transformations as opposed to specific languages based on navigation.
The relational approach separates the physical implementation of the program database. The programs are less sensitive to changes in the physical representation data, linking data and metadata in the database. Application development is more efficient and independent of changes in the physical representation. This is the reason why SQL and relational database systems are widely used: due to the separation of physical and logical representation.
The relational model is sustained by the *entity-association model* whose purpose was not to be implemented, but to represent the data at the abstract and conceptual levels used in computer systems. It was defined in 1976 by P.S. Chen and its specification does not impose any specific data patterning or processing [4].
The *entity-association model* provides an overview and classification of terms used and their relationships, holistic for an entire system or just one area of interest.
The *Enhanced Entity-Relationship Model* includes extensions of the model of Chen and allows the definition of subtypes of a type of entities that inherit attributes from the type of entity that you extend (which in this context is called super type) and additionally attributes their significance. In terms of this model, an entity is an object that exists and can be distinguished from other similar objects [5].
In terms of standard construction of databases, an entity may correspond to a record and its attributes correspond to fields registration. This model is implemented physically, but is used in the analysis of information systems at the logical level.
The *object-oriented model* allows "objects" depositing as elements in the database. An object is composed of text, sound, images and actions that can be applied to the data. Hierarchical, network or relational data models, storage allow only numeric data and text, while object-oriented data model may additionally contain multimedia data like images or video.
The *object-oriented database management systems* provide persistent objects, including associations between objects, and methods. A basic concept of object-oriented model is defined orthogonal persistence [6] by three principles:
The principle of independent persistence: the lifespan of a program is independent of

the data they manipulate. Programs that manipulate data in the short and long term look the same.

The principle of data types orthogonality: All data objects must have full persistence, regardless their type. There are no special cases for items not to be allowed to have a long lifespan or not to be transient.

The principle of persistent identification: Choosing how to identify and supply the items is orthogonally defined to the persistent universe in the system. The identification mechanism for persistent objects is independent on the type of system.

Objects' persistence, including orthogonal persistence, is often achieved through the concept of persistence, when accessibility makes a persistent object if it can be accessed from a persistent root.

This is a completely different approaches grid / tree using language low navigation, and approaches relational or object-relational using a high level language (*HLL - High Level Language*) for navigation, query and manipulation or combined with some SQL data definition language (*DDL - data definition language*).

The *object-relational model* extends the database systems relational model to add concepts from the object-oriented approach and get more complex object structures and rules and yet remain open to other systems. An Object-Relational Database Management System - ORDBMS is most simply defined by the equation in **Table 1**:

Table 1. Defining object-relational model based on object-oriented and relational models

ORDBMS = ODBMS + RDBMS = (O + R) × DB × MS

At the logical level, an ORDBMS is a management system MS that applies methods to process data structures from the database DB and complies both an object O and relational R concepts.

An *object* is an entity with a clear role in the system, characterized by state, behavior and identity. Upon a certain object we can take action that on her own turn can trigger or perform another action. The object can be concrete: a tangible and visible entity, for example a person, place, thing; an abstract entity with it as a concept, an event, a department, marriage, idea; or an artifact of the design process, for example: user interface, control, planning [4].

Any object exposes its behavior through operations that may affect his or another object's state. The state of an object is defined by the values held by properties at a time. The behavior shows how an object acts and reacts to events.

An operation is a simple action performed by an object on another object to get an answer. Operations performed by an object or performed on an object, implemented in a programming language are called methods.

Classes which have links with specific restrictions operations and object-oriented approach, define the object-oriented data model.

The need of persistent data has evolved from sequential files to structured files, network databases, hierarchical databases, RDBMS, and recently into ORDBMS and OODBMS offering more controlled and flexible storage, interface, and transactional capabilities on complex objects and structures.

Fig 4 represents side by side all resembling and complementary concepts and characteristics in data models evolution at conceptual, logical and physical levels.

Over time, there were developed several conceptual models specific to each databases management systems, each with different capabilities, both in terms of organization, data modeling, and access.

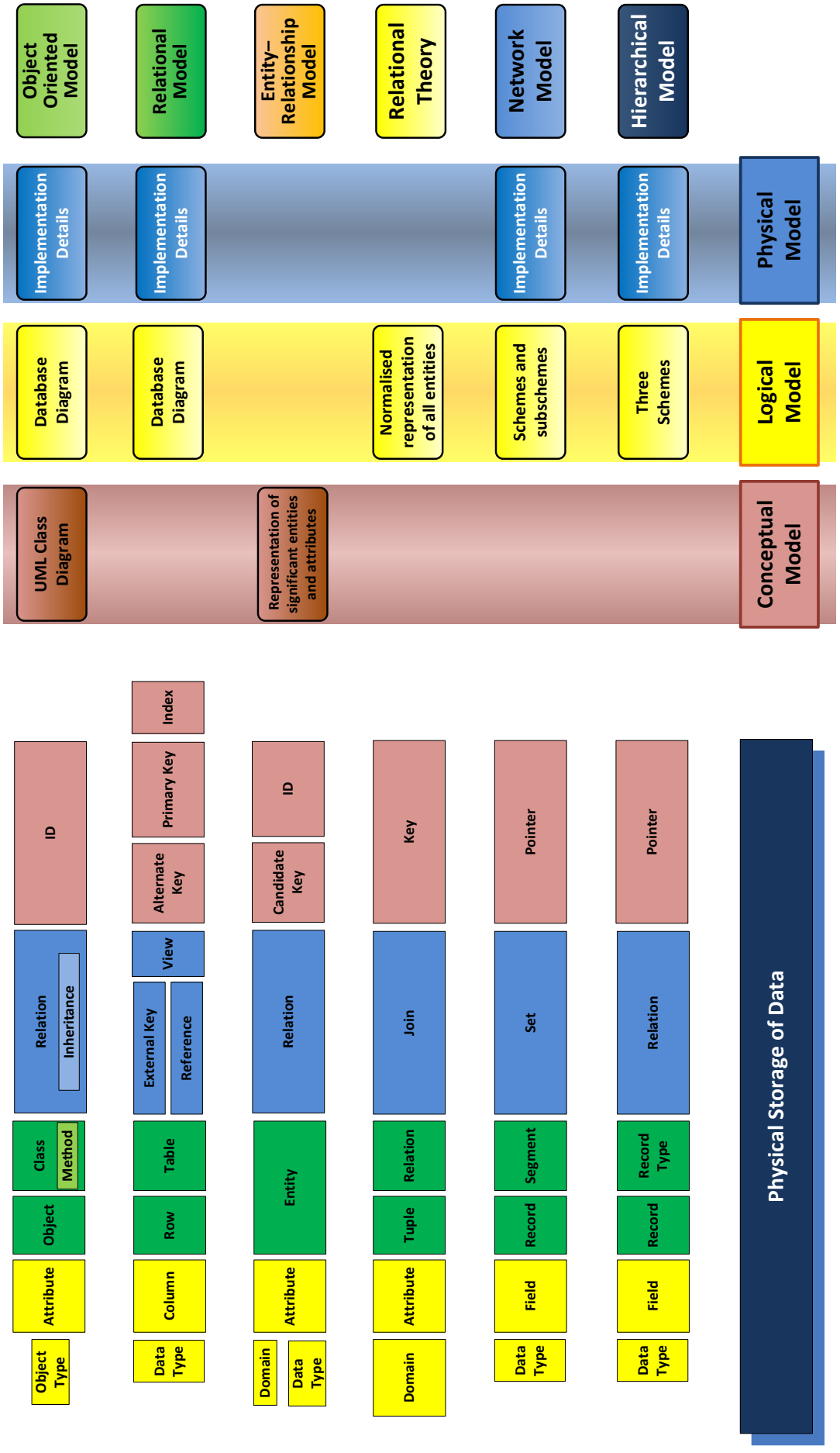

Fig. 4. Correspondence of the data models in database systems evolution

Essentially, data models should allow different applications to be able to share the same data. However, frequently achievement and maintenance of information systems cost more than would be required, and data models, as a result of weak implementation, become an obstacle for business processes rather than act as a support mechanism.

4 A new adaptable data model

The main role of data models is to ensure data compatibility, a necessary and sufficient framework that allows different applications to share the same structures, to store and access data [7].

This paper aims to define and implement an adaptable data model to assist the process models that can be modified on the fly without recompiling modules application. Thereby, at runtime the system allows adding new types, changing existing relations between defined data types and methods, functions and procedures for system processing.

To achieve this, the application must have the ability to work with metaobjects, to instantiate containers of objects that assist workflows [8].

Most applications are too inflexible to keep pace with business processes they support. Built on an architecture with three levels (e.g.: **Fig. 5.**), these applications have two major problems: "components sharing" and "application integration".

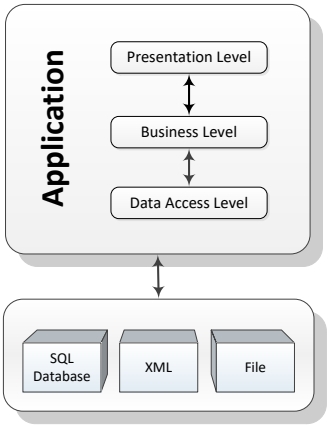

Fig. 5. Architecture based on components

Sharing components within heterogeneous platforms is difficult, if not impossible. When component-based applications need to exchange data, most of the time a designated user must take data manually from one application and put them in the other one.

Adaptability is a characteristic of a system or process. In organizational management, adaptability is generally recognized as the ability to change themselves or other objects to match the changes occurring [9].

Service Oriented Architecture – SOA is an abstract framework that turns business applications into individual business functions and processes. This collection of services is built using compliant standards for systems design integration in real time.

The basic principles of SOA: reuse, granularity, modularity, composability, and interoperability, enable creation of interconnected services between information systems that use middleware, to exchange SOAP (Simple Object Access Protocol) messages.

The *SOAP* standard represents the starting point for messages exchange and exposes behavior of objects, using web services.

SOA makes it possible to loosely couple and reuse functions from processes between different types of platforms [10].

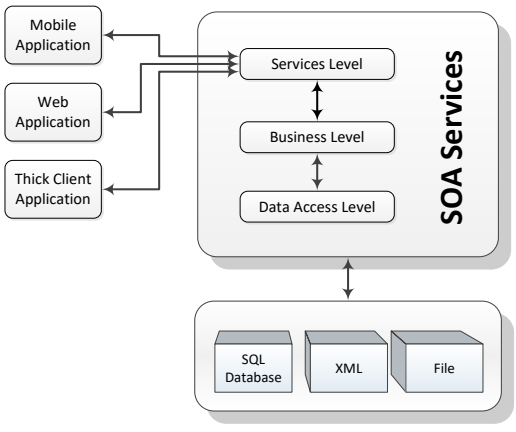

Fig. 6. Service Oriented Architecture

Figure **Fig. 6** exhibits an opposed vision to component-based applications. The Service Oriented Architecture exposes the third level using services. This level describes its interface using SOAP and WSDL (Web

Service Definition Language) and exposes a universal interface which can be used by any interface for users (presentation layer) regardless of device or platform.

Data Oriented Architecture – DOA is a result of "loosely coupled" software components with data-oriented interfaces that enable systems' integration using standards-based communication middle-ware infrastructure [1].

A high level example of WSDL of a supply process that includes the order data model and methods for orders as port types is listed below:

```
<?xml version="1.0" ?>
<definitions name="CerereOferta"
targetNamespace="http://example.com/a
     gentvanzari/wsdl"
xmlns="http://schemas.xmlsoap.org/wsd
     l/"
xmlns:bpws="http://schemas.xmlsoap.or
     g/ws/2003/03/business-process/"
xmlns:plnk="http://schemas.xmlsoap.or
     g/ws/2003/05/partner-link/">

<!-- data types and messages -->
<definitions xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/
envelope/"
xmlns:xsd="http://www.w3.org/2001/XML
Schema"
xmlns:xsi="http://www.w3.org/2001/XML
Schema-instance"
xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/
encoding/"
xmlns:tns="urn:docs_wsdl"
xmlns:soap="http://schemas.xmlsoap.or
g/wsdl/soap/"
xmlns:wsdl="http://schemas.xmlsoap.or
g/wsdl/"
xmlns="http://schemas.xmlsoap.org/wsd
l/" targetNamespace="urn:docs_wsdl">
<types>
   <xsd:schema
targetNamespace="urn:docs_wsdl">
      <xsd:import
namespace="http://schemas.xmlsoap.org
/soap/encoding/"/>
     <xsd:import
namespace="http://schemas.xmlsoap.org
/wsdl/"/>
     <xsd:complexType name="order">
       <xsd:all>
         <xsd:element name="id"
type="xsd:int"/>
         <xsd:element name="number"
type="xsd:string"/>
         <xsd:element name="dueDays"
type="xsd:int"/>
         <xsd:element name="issueDate"
type="xsd:date"/>
         <xsd:element name="delivery"
type="xsd:date"/>
        </xsd:all>
      </xsd:complexType>
      <xsd:complexType name="orders">
         <xsd:complexContent>
            <xsd:restriction base="SOAP-
ENC:Array">
               <xsd:attribute ref="SOAP-
ENC:arrayType"
wsdl:arrayType="tns:order[]"/>
            </xsd:restriction>
         </xsd:complexContent>
      </xsd:complexType>
   </xsd:schema>
</types>

<!-- port type definitions -->
<portType name="InterfataAgentVanzari">
   <operation name="CerereOferta">
      <input message="cerereOferta"/>
      <output
     message="confirmareOferta"/>
   </operation>

   <operation name="AnuleazaComanda">
      <input message="cerereAnulare"/>
   </operation>

   <operation name="LivreazaComanda">
      <input message="cerereLivrare"/>
      <output
     message="confimareLivrare"/>
   </operation>
</portType>

<portType name="InterfataMerceolog">
   <operation name="NotificareAnulare">
      <input message="mesajAnulare"/>
   </operation>

   <operation
     name="NotificareExpirareTimp">
      <input
     message="mesajExpirareTimp"/>
   </operation>

   <operation name="NotificareReceptie">
      <input
     message="receptieMateriale"/>
   </operation>
</portType>

<!-- partner link types definitions -->
<plnk:partnerLinkType
     name="SerciciuAgentVanzari">
   <plnk:role name="AgentVanzari">
      <plnk:portType
     name="InterfataAgentVanzari"/>
   </plnk:role>
   <plnk:role name="Merceolog">
```

```
      <plnk:portType
     name="InterfataMerceolog"/>
   </plnk:role>
</plnk:partnerLinkType>

<!-- properties definition -->
<bpws:property name="IDrezervare"
     type="xsd:string"/>

<!-- aliases for properties are
     omitted -->
</definitions>
```

Keeping the metadefinition of data and methods inside the application provides an adaptable data model to assist the process models that can be modified on the fly without recompiling modules application. All messages delivered using SOAP over HTTP are self-described and clients can interpret all data and methods using the contained definition.

Low coupling abstract interfaces offered by WSDL (including data types definitions, messages and port types) represent the premise and advantage for building a flexible system with a design meant to change on the fly.

5 Conclusions

Companies are performing in a dynamic economic environment and information systems assisting their business processes are limited by high degree of coupling in three major areas: systems are limited by coupling between modules and their interaction with other systems; data model is fixed at compilation time and does not allow deviations or subsequent adjustments; and workflow models do not allow improving and evolution of the business processes.

Dynamic exposure of all evolving process models and adaptive data models by WSDL interfaces creates the premise of a versatile system in which each object, rule, function, or service interface can be changed in a shorter time. The aim is to optimize the process to obtain a value or quality.

Therefore, the essence of SOA is to dynamically link resources with chain transformations, while the ground of DOA is to expose the data and hide the code.

Acknowledgment

We owe thanks to all the members of the Department of Informatics, for the sessions and Communications Conference on Informatics where we could present case studies and also published the results of our research. It's always a pleasure working with members of the Database Collective Faculty of Cybernetics, Statistics and Informatics.

References

[1] I. Lungu, A. Mihalache – *Enterprise modeling and software engineering for information systems improvement*. Studii şi Cercetări de Calcul Economic şi Cibernetică Economică, pp. 18, Nr 2/2010, ISSN: 0585 – 7511, EISSN: 1843 – 0112

[2] Data Governance Institute, *Definitions of Data Governance*, http://www.datagovernance.com/adg_data_governance_definition/.

[3] E. F. CODD, *A Relational Model of Data for Large Shared Data Banks*, IBM Research Laboratory, San Jose, California, 1970, https://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf

[4] I. Lungu, A. Bâră, C. Bodea, I. Botha, V. Diaconiţa, A. Florea, A. Velicanu, *Tratat de baze de date – Baze de date. Organizare. Proiectare. Implementare*, 2011, ISBN 978-606-505-481-3;

[5]I. Lungu, M. Velicanu, *Database Systems – Present and Future*, Informatica Economica Journal, Inforec, ISSN 1453-1305, EISSN 1842-8088;

[6]M P. Atkinson and R. Morrison, "Orthogonally Persistent Object Systems", *VLDB Journal* 4 (3), 319-401, 1995, ISSN: 1066-8888

[7]I. Lungu, Ghe. SABĂU, M. Velicanu, M. Munten, S. Ionescu, E. Posdarie, D. Sandu, *Sisteme Informatice. Analiză, proiectare şi implementare*, 2003, ISBN 973-590-830-1;

[8]I Lungu, A. Mihalache, "Adaptable

Enterprise Modeling – A New Challenge for Collaborative Data and Process-Aware Management Systems". *Proceedings of the World Multiconference on Mathematics and Computers in Business and Economics* (MCBE '10), Iaşi, România, pp. 261–266, WSEAS Press 2010, ISSN: 1790-2769

[9] K. Andersen, N. Gronau, *An Approach to Increase Adaptability in ERP Systems*, In: Managing Modern Organizations with Information Technology: Proceedings of the 2005 Information Recources Management Association International Conference, 2005, ISBN: 978-1-59140-822-2;

[10] E. A. Marks, *Service-Oriented Architecture Governance for the Service Driven Enterprise*, John Wiley & Sons, 2008, ISBN 978-0-470-17125-7;

[11] A. Mihalache, C. Vintilă, A. Cornescu – *A New Model Approach for Business Applications in Enterprise 2.0 Environment*, In Proceedings of the World Multiconference on APPLIED ECONOMICS, BUSINESS AND DEVELOPMENT (AEBD '09), June 2009, Tenerife, Canary Islands, Spain, pp. 192–195, WSEAS Press 2009, ISSN: 1790-5109.

Ion LUNGU graduated from the Faculty of Cybernetics, Statistics and Economic Informatics of the Academy of Economic Studies in 1974. He got the title of doctor in economy in the specialty economic informatics in 1983. He has been directing graduates who study towards getting a doctor's degree since 1999. At present he is a professor in the department of the faculty of Cybernetics, Statistics and Economic Informatics of the Academy of Economic Studies of Bucharest. He had documentary activity and specialization with the Eindhoven Technical University of Holland, the Economic University of Athens and Economic University of Milan. His domains of work are: informatics systems and databases. Among his books are: "Databases, organization, design and implementation", (1995), "Information Systems for Management" (1994), "SGBD Oracle Applications" (1998); "Let's learn Oracle in 28 lessons" (2003), "Database systems" (2003), "Information Systems – Analysis, Design and Implementation" (2003).

Andrei MIHALACHE has a background in computer science and is interested in database related issues and enterprise modeling techniques. Currently, he is a PhD candidate in the field of Economic Informatics at the Academy of Economic Studies. His research interests include: enterprise data model, adaptable information systems, business process improvement, and web collaborative technologies.

Big Data Mining: Challenges, Technologies, Tools and Applications

Asha M. PAWAR
Assistant Professor, SKNCOE, Computer Engineering Dept.
aashapawar20@gmail.com

Big data is a data with large size means it has large volume, velocity and variety. Now a day's big data is expanding in a various science and engineering fields. And so there are many challenges to manage and analyse big data using various tools. This paper introduces the big data and its Characteristic concepts and Next section elaborates about the Challenges in Big data. In Particular, wed discuss about the technologies used in big data Analysis and Which Tools are mainly used to analyse the data. As big data is growing day by day there are lot of application areas where we need to use any of the technology and tools discussed in paper. Mainly this paper focuses on the Challenges, Technologies, Tools and Applications used for big data Analysis.
Keywords: *big data, big data analysis, mining, heterogeneous data.*

1 Introduction

I) Data:

Data is unprocessed raw material that may or may not specify meaning. Data gives the general values to the user. E.g. Financial data, medical data and so on. It gives the collection of the things at one place. But it is very difficult to find out some meaning from those raw figures so that we need to process this data to get some meaningful information. Data may be in the form of fact, text and numbers.

II) Information:

Data can be represented in information to perform some analysis. Information gives us some meaningful information about the raw data. It is processed and interpreted in some specific format.

III) Knowledge:

Knowledge is the processed information that may give us the typical structure or value. With information it adds some experience and gives the correct analysis.

Data ⟶ Information ⟶ Knowledge

E.g. in Financial Services like Bank the data is all collected information from all stakeholders of bank, Information is separate department data which is represented in some specific format and Knowledge is withdraw process by each customer with previous experience.

Data can be represented in different kinds like Relational Data, Transaction Data and Data Cubes.

IV) Data Mining

Data Mining is one important way to analyse the data in some proper format. Data Mining is a process in which data is analysed on different criteria and summarize it for further use. In other words Data Mining is extract information from large set of data values. That means mining knowledge from large data values is Data Mining also referred as Knowledge Discovery. Data Mining can be useful in different areas like fraud detection, Market analysis, Target Analysis.

V) Big Data and its characteristic

Big Data is typically a data itself with large size and it is difficult to maintain, collect and manage by any analysis tool or user. E.g. Data stored on Facebook, twitter are Big data. The volume and velocity of this kind of data is too large. Big data includes video, photo, audio, and simulation and 3D models [3].

There are some important characteristic of big data like Volume, Velocity, Variety, Veracity, and Value. These 5V's are main characteristics of the big data [1], [2] :

1. Volume represents the size of the database in terabyte or petabyte.
2. Velocity is the speed of data in communication in a real world.
3. Veracity describes Incompleteness data.
4. Variety gives us the structure of different data types used in real world.
5. Value represents the important information from the data source.

VI) Big Data Mining

Data mining is the process to gather information in same domain. The purpose of data mining is to collect information for any particular issue which will helpful to data analyst for classification and prediction. So now todays challenge is to mine this large volume of data so that it can be analysed by any mining tool.

2 Challenges of Big Data Mining

a. Volume and Scalability.

It is the biggest challenge to deal with the size of data. As Twitter generates 7 + Terabytes of data and Facebook generate 10 + Terabytes of data every year so it becomes difficult to manage and analyse. As we are moving from Terabytes to Petabytes and from Petabytes to Zeta bytes of data it's the important task to analyse this Big data by some methodology. Scale the data in proper way is the important issue in big data mining.

b. Miss-Handling of Big Data

Data handling mainly depends on the scalability of data. And scalability depends on data size, hardware size, and concurrency. Day by day data size is increasing and format to store data is also changing and not fixed in future so it's the task of data analyst to overcome such challenge so mishandling of data by different users.

c. Privacy and Security

In Big data, data size and format are not fixed so it's difficult to maintain privacy of one user from another. And because of this volume of data security algorithms are not fixed. When size of data changes or format changes then we need to apply new security algorithms. Ones we define the security or privacy algorithms to it cannot be applicable to upgraded data. E.g. In hospital the data collected and it may upgrade daily and it may be in different format, so it becomes difficult to analyse and secure the newly added data.

As data is linked with so many formats and users it's a fear to keep privacy of data and hence it's a big challenge in data mining.

d. Speed and Velocity

Velocity refers to unique speed with timely manner. But in many cases it is difficult to maintain unique speed because of variety and size of data.

e. Heterogeneity of Data

Data analysis has first step that data must be structured in a well format. Some errors and confusion in data may lead to miss-classification of data. Machine analysis algorithm only understands homogeneous or structured data. Hence to make the data in homogeneous format is a big challenge in big data mining.

3. Big Data Mining Techniques

a. ANN

In general ANN is called as "Neural Network". NN is a non-linear statistical data modelling approach and used to manage complex relationships between I/P and O/P. As dataset used in ANN grows massively we need to analyse it automatically. It is also helpful to recognize the pattern from which it belongs. Classification, Prediction, Clustering & Association Rules are the steps of data mining and are useful in neural network to identify patterns. [8]

b. Decision Tree

It is one of most powerful tool in data mining process. Decision tree is originally implemented in decision theory and

statistics. It is used for nominal and numeric data values. Decision analysis is performed with the help of tree shaped structure. Three main components used in decision tree are 1) Square represents decision node 2) Circle represents chance node and 3) Triangle represents end node.

c. Genetic Algorithm

Genetic algorithm involves three steps selection, crossover and mutation for every gene. It follows survival of fittest law. Genetic algorithm has found applications in phylogenetic, computational science, engineering, economics and many more. GA can be used as a classifier in many areas also it can be used for prediction analysis purpose. GA uses many techniques like K-Nearest Neighbour (KNN) and Rule Induction. [9] [12].

d. Classification

Classification is a method of assigning a label to unclassified data. There are different methods in classification like Supervised and Unsupervised classification methods. There are many techniques used for classification like Bayesian Classification, ANN and Support Vector Machine (SVM).

e. Clustering

Clustering is the process of making the groups of similar objects together. While performing clustering we need to group the items on data similarity and assign the label to each group. It is more advantageous than classification because it helps to find useful features that represent different groups. Clustering has many application areas like market research, pattern recognition, data analysis, and image processing. Clustering deals with the High Dimensional Data. It has many techniques to make groups like Partitioning Method, Hierarchical Method, Density-based Method, Grid-Based Method, Model-Based Method, and Constraint-based Method.

4. Applications of Big Data

a. Healthcare and medicine

Big data creates the link between patients, doctors, diagnosis and predicting the surgery and pharmaceutical companies. Also Big data haves the computing power too high to decode DNA sequence and predict the disease or required pattern. Big data can also be helpful to monitor premature babies and sick baby unit. By analysing every heartbeat now a days it can be possible to identify the disease before its actual symptoms. [6]

b. Banking

In financial sector it will also help for predictive area in customer behaviour. Different applications in banking like credit risk analysis, customer changes, marketing policy, and historical transaction are handled by many big data tools. Business Intelligence with respect to financial banking can be handled by Big data tools.[7]

c. Telecommunication

Nowadays in telecommunication field call analysis, pricing, prediction of funds, customer loyalty and customer ratio all these can be handled by Big data mining tools.
Mobile user data mining tool is one example.

d. Marketing

Marketers and retailers uses the big data mining tools to decide whether advertisement is in correct way or not, based on customer facial recognition software. Because marketers want that advertisement is so viral so that they can be useful to improve sell of product. Also big data mining tools are helpful to prepare customer report and shopping cart analysis. [7]

e. Industry

Many Big data mining tools are helpful to prepare report on production management like quality of the product, process

optimization, store inventory and employee management on various products.
It also helpful to optimize staffing through data, reduce fraud and timely analysis of inventory.

f. Social Media

Now a day, social sites generates tremendous data every day, so many tools are used to analyse and manage it properly.

5. Tools used in Big Data

Big data deals with many data types like structure, unstructured and also volume of data is too large so to analyse these data we need some tool like Hadoop, NoSQL, MapReduce, R-Language, RapidMiner, WEKA and KNIME.

a. Hadoop

Hadoop is an open source software, that stores and process big data. It uses cluster approach to manage big data. It is java based framework allows to use inbuilt library functions and different tools. Hadoop can execute large, complex data set on different operating system like Windows, LINUX, and UNIX and so on. GOOGLE and YAHOO use Hadoop Framework to analyse their data efficiently.

b. NoSQL

NoSQL refers to not only SQL. NoSQL allows using structured data as well as un-structured data. Means data can be accessed by SQL queries as well as some new techniques can be used. NoSQL is open source software and uses DaaS (Database as a Service). As Graph databases are more popular solution for big data NoSQL is the right application who manages it efficiently.

c. MapReduce

It is processing model for distributed environment on parallel architecture. It is an open source tool to implement process and manage large data efficiently. MapReduce can be categorized in two parts Map converts the data into another type of data and divides into data tuples like key/value pair and Reduce takes input from Map and combines data tuples into small set of tuples.
MapReduce is used to sort Petabytes of data in an hour.

d. R-Language

developed by Bell Labs. R uses S Programming to handle large volume of data.
R is programming language for statistical values, complex data and graphical information. Effective data handling and storage can be done by R-Language. R provides graphical facilities for data analyst and has many tools to perform data analysis

6. Conclusions

The basic purpose of this paper is to introduce basic concepts about big data and its mining techniques. It's a survey about the big data analysis methods. Hence we introduce in first part all basic about the big data mining then we introduce the different challenges to handle and manage big data. Lateral part gives information about various techniques used in data mining to improve the accuracy and performance. Than we discuss about the tools used for the big data analysis for different techniques of big data. And finally we discuss about the applications of big data in various domains. In future, Big data and Cloud are nearer terms so we can further expand big data analysis using cloud technology.

Acknowledgment

I am really thankful to my Husband Mr. Mohan Pawar for guiding me. Also I am thankful to all my friends who encourage me all the time.

References

[1]. A Survey on Big Data, Mining: (Tools, Techniques, Applications and Notable Uses) Nour E. Oweis 1,4 , Suhail S. wais 2 , Waseem George 1 , Mona G. Suliman 3 , Václav Snášel 1,4 Springer Publication.

[2]. Kudyba, S. (2014). Big Data, Mining, and Analytics: Components of Strategic Decision Making. CRC Press.

[3]. Gupta, R. (2014). Journey from Data Mining to Web Mining to Big Data. arXiv preprint arXiv:1404.4140.

[4]. Big Data Challenges Alexandru Adrian T OLE Romanian American University ,Bucharest, Romania

[5]. A Survey on Big Data, Mining: (Tools, Techniques, Applications and Notable Uses) Nour E. Oweis 1,4 , Suhail S. Owais 2 , Waseem George 1 , Mona G. Suliman 3 , Václav Snášel 1,4 Springer Publications.

[6]. Big Data Challenges: Data Analysis Perspective Riya Lodha , Harshil Jain and Lakshmi Kurup Computer Engineering Department, D.J.Sanghvi College of Engineering, Mumbai University, Mumbai, India Accepted 10 Sept 2014, Available online 01 Oct 2014, Vol.4, No.5 (Oct 2014) International Journal of Current Engineering and Technology E-ISSN 2277 – 4106, P-ISSN 2347 – 5161 ® ©2014 INPRESSCO , All Rights Reserved

[7]. W. Raghupathi and V. Raghupathi, "Big data analytics in healthcare: promise and potential", Health Inf Sci Syst, vol. 2, no. 1, p. 3, 2014.

[8]. Applications of Big data in Various Fields Kuchipu di Sr avanthi, Tatireddy Subba Reddy, Assistant Professor, CSE Department, Assistant Professor, CSE Department, QIS Institute of Technology,. VVIT ,NAMBUR, Vengamukkapalem, Ongole, India, 523272 GUNTUR, Andhra Pradesh,

[9]. Journal of Theoretical and Applied Information Technology © 2005 - 2009 JATIT. All rights reserved. www.jatit.org

[10]. NEURAL NETWORKS IN DATA MINING DR. YASHPAL SINGH, ALOK SINGH CHAUHAN Reader, Bundelkhand Institute of Engineering & Technology, Jhansi, India Lecturer, United Institute of Management, Allahabad, India E-mail: yash_biet@yahoo.co.in , alok_sc@yahoo.co.in

[11]. International Journal of Information Management 35 (2015) 137–144 Contents lists available at ScienceDirect International Journal of Information Management jou rnal h om epage: www.elsevier.com/locate/ijinfomgt

[12]. Beyond the hype: Big data concepts, methods, and analytics Amir Gandomi ∗ , Murtaza Haider Ted Rogers School of Management, Ryerson University, Toronto, Ontario M5B 2K3, Canada

[13]. Lahoti, A. A., & Ramteke, P. L. (2014). Data Mining Technique its Needs and Using Ap-plications. IJCSMC, Vol. 3, Issue. 4, April 2014, pg.572 – 579.

[14]. http://www.ibm.com/developerworks/library/ba-data-mining-techniques/

[15]. http://thenewstack.io/six-of-the-best-open-source-data-mining-tools/

[16]. The Four V's of Big Data – IBM http://www.ibmbigdatahub.com/infograp hic/four-vs-big- data (last seen 05–April–2015).

[17]. Wu, X., Zhu, X., Wu, G. Q., & Ding, W. (2014). Data mining with big data.Knowledge and Data Engineering, IEEE Transactions on, 26(1), 97-107.

[18]. Jain, N., & Srivastava, V. (2013). DATA MINING TECHNIQUES: A SURVEY PAPER. IJRET: International Journal of Research in Engineering and Technology.

[19]. Saed Sayad, Data Mining Map, An Introduction to Data Mining, http://www.saedsayad.com/ (2012). (Last seen 05–April–2015).

Mrs. Asha M. Pawar graduated from BAMU University in 2005 from Information Technology field. She has completed her M.Tech Computer Science & Engineering from VTU Belgaum in 2012. She has total 11 Yrs of Experience as an Assistant Professor in Pune University. She has published total 6 papers in various international journals.

Efficient Partitioning of Large Databases without Query Statistics

Shahidul Islam KHAN
Department of Computer Science and Engineering (CSE)
Bangladesh University of Engineering and Technology (BUET), Dhaka, Bangladesh
nayeemkh@gmail.com, shahid@grad.cse.buet.ac.bd

An efficient way of improving the performance of a database management system is distributed processing. Distribution of data involves fragmentation or partitioning, replication, and allocation process. Previous research works provided partitioning based on empirical data about the type and frequency of the queries. These solutions are not suitable at the initial stage of a distributed database as query statistics are not available then. In this paper, I have presented a fragmentation technique, Matrix based Fragmentation (MMF), which can be applied at the initial stage as well as at later stages of distributed databases. Instead of using empirical data, I have developed a matrix, Modified Create, Read, Update and Delete (MCRUD), to partition a large database properly. Allocation of fragments is done simultaneously in my proposed technique. So using MMF, no additional complexity is added for allocating the fragments to the sites of a distributed database as fragmentation is synchronized with allocation. The performance of a DDBMS can be improved significantly by avoiding frequent remote access and high data transfer among the sites. Results show that proposed technique can solve the initial partitioning problem of large distributed databases.

Keywords*: Distributed Database, Partitioning, Fragmentation, Allocation, MCRUD matrix*

1 Introduction

A distributed database (DDB) is a collection of data that logically belongs to the same system but spreads over the sites of a computer network. It is not necessary that database system has to be geographically distributed. The sites of the distributed database can have the same network address and may be in the same room but the communication between them is done over a network instead of shared memory. DDB is an efficient way of improving the performance of applications that manipulate large volumes of data. The design of efficient distributed databases is one of the major research problems in database and information technology areas. Primary concerns of distributed database design are partitioning the relations or tables, allocating them in different sites of a distributed system, and local optimization in each site [1], [2].

Database partitioning or fragmentation is a design technique to divide a single relation or class of a database into two or more partitions such that the combination of the partitions provides the original database without any loss of information. This reduces the amount of irrelevant data accessed by the applications of the database, thus reducing the number of disk accesses. Fragmentation can be horizontal, vertical or mixed/hybrid. Horizontal fragmentation (HF) allows a relation or class to be partitioned into disjoint tuples or instances. Vertical fragmentation (VF) partitioned a relation or class into disjoint sets of columns or attributes except the repetition of primary key column. The combination of horizontal and vertical fragmentations to form mixed or hybrid fragmentations (MF/ HF) is also proposed [3]. Allocation is the process of assigning the fragments of a database on the sites of a distributed network. The replication of fragments improves reliability and efficiency of read-only queries but increase update cost. The main reasons of fragmentation of the relations are to increase locality of reference of the queries, improve reliability and availability of data and performance of the system, balance storage capacities, and minimize

communication costs [1]-[4].

1.1 Problem Definition

In distributed database design, the basis of fragmentation (horizontal, vertical or mixed) of relations is one of the follows:

- Frequency of different queries executed in a system at runtime,
- Affinity matrix of minterm predicates constructed from combination of predicates
- Attribute affinity matrix constructed based on the relationship between different attributes of a table and run time transactions that access the attributes

To know actual query frequencies or to construct above matrices sufficient experiential data are required that are not available in most cases at the initial stage of a distributed database. Moreover, almost all the previous techniques concentrated only fragmentation problem and overlooked allocation problem to reduce the complexity of the problem. But the overall performance of a distributed system fragmented by a very good fragmentation technique can be very low if proper allocation of the fragments to the sites of the distributed system cannot be ensured.

Available techniques developed by the researchers so far to support fragmentation cannot provide a solution at the initial level of a distributed system. They use frequency of queries executed in a system at runtime, affinity matrix of minterm predicates constructed from combination of predicates or attribute affinity matrix constructed based on the relationship between different attributes of a table and run time transactions that access the attributes as a basis of fragmentation of the relations. To construct these matrices sufficient experiential data are required that are not available in most cases at the initial stage of a distributed system. So using currently available techniques for fragmentation, the database administrator has to put the whole database in a single site of the system and perform fragmentation and allocation after a long period when sufficient empirical data will be available to him. During this period facilities of distributed database cannot be enjoyed. After the period the database can be fragmented correctly to some extent and allocated to the sites with a high communication cost of transferring a huge amount of data from central node to all other nodes of the system. Due to the deficiencies of fragmentation and allocation techniques existing in the literature, my research focused fragmentation and allocation in an integrated manner. Based on locality of data, partitioning the database and allocating the partitions are performed with the objective of minimizing data transmission costs and maximizing locality of data

In this paper, I have presented a fragmentation technique namely Matrix based Fragmentation (MMF) that is capable of partitioning relations of a distributed database properly at the initial stage when data access statistics and query execution frequencies are not available. Instead of using empirical data, I have developed a matrix namely Modified Create, Read, Update and Delete (MCRUD) to make fragmentation decisions. Using our technique, no additional complexity is added for allocating the fragments to the sites of a distributed database as fragmentation is synchronized with allocation. So the performance of a DDBMS can be improved significantly by avoiding frequent remote access and high data transfer among the sites. This will improve the bandwidth of the system as well.

1.2 Contributions of the Paper

- The main contribution is to develop a fragmentation technique that can partition relations without empirical data.
- Relations are fragmented initial with the help of MCRUD matrix. This overcomes initial fragmentation problem of distributed database that is not properly addressed in other fragmentation techniques.

- A very good hit rate (Approximately 90%) is achieved using my proposed technique for various kinds of insertion, selection, join, deletion and other queries.
- In our technique large amount of costly data transfer using communicational network can be avoided as fragments are correctly allocated to different sites at the initial stage of the system.

The rest of the paper is organized as follows. In Section 2, a brief review of the research in horizontal, vertical and mixed fragmentation technique of distributed database is presented and limitations of the available fragmentation techniques are also discussed. Section 3 describes the details of Matrix based Fragmentation (MMF) technique In Section 4, I have presented the results of the experiments to show the performance of my proposed technique. Finally, Section 5 concludes the paper and provides suggestions for future research.

2 Literature Review

2.1Complexity of the problem

The combined problem of fragmentation and allocation is proven NP-hard [6]. In the case of Horizontal fragmentation, if n simple predicates are considered 2^n is the number of horizontal fragments using minterm predicates. If there are k nodes, the complexity of allocating horizontal fragments is $O(k^{2^n})$.

For example, using 6 simple predicates to perform horizontal fragmentation results in $2^6 = 64$ fragments. To find the optimal allocation of the fragments in 4 sites one needs to compare all the $4^{64} \approx 10^{39}$ possible allocations.

For vertical fragmentation, if a relation has m non-primary key attributes, the number of possible fragments: Bell number $B(m) \approx m^m$. The fragment allocation is of complexity $O(k^{m^m})$. Due to the complexity of both fragmentation and allocation, allocation of the fragments are often treated independently than fragmentation of the database.

2.2 Horizontal Fragmentation

There are two types of horizontal fragmentation, primary and derived. Primary horizontal fragmentation of a relation or a class is performed using predicates of queries accessing this relation or class, while derived horizontal fragmentation of a relation or a class is performed based on horizontal fragmentation of another relation or class.

In the context of the relational data model, existing approaches for horizontal fragmentation mainly fall into following three categories [7], [1]:

- minterm-predicate-based approaches: which perform primary horizontal fragmentation using a set of minterm predicates, e.g., [1], [2], [8].
- affinity-based approaches: which, at first, group predicates according to predicate affinities and then perform primary horizontal fragmentation using conjunctions of the grouped predicates, e.g., [9] - [12].
- other approaches: approaches other than minterm predicate or predicate affinity-based approach, e.g., [13] – [16].

2.3 Vertical Fragmentation

Vertical fragmentation has been studied since the 1970s. There are two main approaches [7]:

- The pure affinity-based approach takes attribute affinities as the measure of togetherness of attributes to fragment attributes of a relation schema. Research work includes [17]-[24].
- The cost-driven approach uses a cost model while partitioning attributes of a relation schema. Research work includes [25] - [30].

Initial Vertical Fragmentation

Abuelyaman [30] provided a solution of initial fragmentation of database using vertical fragmentation technique namely

StatPart. To fragment a relation, it starts with a randomly generated matrix of attribute vs. queries called the reflexivity matrix. It then constructs symmetry matrix from the reflexivity matrix using two equations. Symmetry matrix is inputted to transitivity module which uses an algorithm to produce two set of attributes those will be used to break the relation into two binary vertical fragments.

Main two drawbacks of StatPart [30] are:

- It can suggest only two binary vertical fragments independent of the number of sites of the distributed system. So this technique is not suitable for a distributed system with more than two allocation sites.
- As it starts with a randomly generated matrix that represents the relationship among attributes and queries, optimum fragmentation decision cannot be provided using this algorithm. So it continuously shifts attributes from one fragment to another fragment trial and error basis to improve hit ratio.

Recent research on Horizontal or Vertical partitioning includes fragmentation of very large databases, cloud-based systems, multimedia databases etc. [31]-[35].

2.4 Mixed Fragmentation

Navathe et al. [3] proposed a mixed fragmentation methodology that simultaneously applies horizontal and vertical fragmentation on a relation. The input of the procedure comprises a predicate affinity table and an attribute affinity table. A set of grid cells is created first which may overlap each other. Then some grid cells are merged such that total disk accesses for all transactions can be reduced. Finally, the overlap between each pair of fragments is removed using two algorithms for the cases of contained and overlapping fragments.

Adopting some developed heuristics and algorithms in [3] to fragmentation in object oriented databases, Baiˇao and Mattoso [36] proposed a design procedure which includes a sequence of steps: analysis phase, vertical and horizontal fragmentation. In the first step, a set of classes that are needed for horizontal fragmentation, vertical fragmentation, or non-fragmentation, are identified. In the second and third steps, vertical and horizontal fragmentations are performed on the classes identified in the first step, using algorithms extended from the one in [3]. All fragmentation algorithms are affinity based. The evaluations of the resulting fragmentation are not based on any cost model. Baiˇao et al. [4] considered mixed fragmentation as a process of performing vertical fragmentation on classes first and then performing horizontal fragmentation on the set of vertical fragments.

2.5 Allocation

In the literature, allocation problems are first addressed for file allocation. Chu [37] presented a simple model for a non-redundant allocation of files. Casey [56] proposed a model which allows the allocation of multiple copies. Queries and updates are distinguished in the model. Mahmoud and Riodon [38] proposed a model for studying file allocation and the capacity of communication capacities to obtain an optimized solution which minimizes storage and communication cost. Since the early 1980s, data allocation has been studied in the context of relational databases. Due to the complexity of the problem of data allocation, different researchers make different assumptions to reduce the size of the problem. Some works do not consider replication while making a decision of allocation [39, 40] while some others do not consider storage capabilities of network nodes [6, 41].

2.6 Summary

Most of the literature about database distribution considers fragmentation and allocation as two different steps even though they are strongly related problems. Both fragmentation and allocation take the same input information to achieve the same

objectives of improving system performance, reliability, and availability. Existing approaches for primary horizontal fragmentation can be characterized into three streams, one using minterm predicates, one using predicate affinity, and a cost-driven approach using a cost model. Even though each of the approaches claims to be able to improve system performance, there is no evaluation to prove that resulting fragmentation schemata can indeed improve the system performance. Horizontal fragmentation with minterm predicates often results in a large number of fragments which will later be allocated to a limited number of network nodes. Affinity-based horizontal fragmentation approaches cannot guarantee to achieve optimal system performance because the information of data local requirement is lost while computing predicate affinities. Cost-driven approaches use cost models to measure the number of disk accesses without considering transportation cost.

For vertical fragmentation, there are two main approaches existing in the literature: affinity based and cost-based. The affinity-based vertical fragmentation approach originated for centralized databases with hierarchical memory levels, for which the number of disk accesses is the main factor that affects the system performance. Later, this approach was adapted to distributed databases for which transportation cost is the main cost that affects the system performance. Attribute affinities only reflect the togetherness of attributes accessed by applications. Vertical fragmentation based on affinities may reduce the number of disk accesses. However, there is no clear proof that affinity-based vertical fragmentation can indeed improve data local availability and thus improve system performance. The cost-driven approach performs vertical fragmentation based on a cost model that measures the number of disk accesses. The optimal solution chosen by this approach is the vertical fragmentation schema that has the fewest number of disk accesses. However, there is no fragmentation approach, for both horizontal and vertical fragmentation, taking data locality into consideration.

Due to the complexity of the allocation problem, it is infeasible to find optimal solutions. Researchers provided heuristic solutions with many assumptions to reduce the complexity of the problem. The assumption that fragmentation is completed properly is not reasonable. Because it is not possible to solve the fragmentation problem independently from the allocation problem as the optimal fragmentation can only be achieved with respect to the optimal allocation of fragments.

3 Matrix based Fragmentation Technique (MMF)

To solve the problem of taking proper fragmentation decision at the initial stage of a distributed database, I have developed a new partitioning technique based on locality precedence of the attributes. Instead of using empirical data, I have developed Modified Create, Read, Update and Delete (MCRUD) matrix to obtain fragmentation decisions. The details of the technique are discussed in the following sections.

3.1 CRUD Matrix

A data-to-location CRUD matrix is a table in which rows indicate attributes of the entities of a relation and columns indicate locations of the applications [42]. It is used by the system analysts and designers in the requirement analysis phase of system development life cycle for making a decision of data mapping to different locations [42], [43]. An example of a traditional CRUD Matrix is shown in the Fig. 1.

Entity / Use Case	Order	Chemicals	Requestor	Vendor Catalog
Place Order	C	R	R	R
Change Order	U, D		R	R
Manage Chemical Inventory		C, U, D		
Report on Orders	R	R	R	
Edit Requesters			C, U	

Fig. 1 Example of a CRUD Matrix adopted from [44]

3.2 MCRUD Matrix

I have modified the existing CRUD matrix according to our requirement of horizontal fragmentation and name it Modified Create, Read, Update, and Delete (MCRUD) matrix. It is a table constructed by placing predicates of attributes of a relation on the row side and applications of the sites of a DDBMS on the column side. I have used MCRUD matrix to generate attribute locality precedence (ALP) table for each relation. An example of an MCRUD Matrix is shown in Fig. 2. In this example, the distributed system has three sites and one application is running on each site. Entity set, attribute, and predicate are denoted by *e*, *a* and *p* respectively. If an application of a site has chances to perform create or read or update or delete operation to an attribute's certain predicate then C or R or U or D will be written in the intersecting cell of the matrix.

Site.Application / Entity.Attribute.Predicates	Site1	Site2	Site3
	Ap1	Ap1	Ap1
$e.a_1.p_1$	CRUD	R	R
$e.a_1.p_2$	RU	CRUD	CRU
$e.a_2p_1$	R	R	CRUD
$e.a_2.p_2$	R	RU	R
$e.a_3.p_1$	CRUD		R
$e.a_3.p_2$	R	R	CRUD

Fig. 2 Example of a MCRUD Matrix

3.3 Attribute Locality Precedence (ALP)

In my developed technique, a relation is fragmented according to the locality of precedence of its attributes. Attribute Locality Precedence (ALP) as the value of importance of an attribute with respect to the sites of a distributed database [45], [46]. A relation in a database contains different types of attributes those describe properties of the relation. But the important thing is that the attributes of a relation do not have same importance with respect to data distribution in different sites. For example in Fig. 2, there are three attributes a_1, a_2 and a_3. Among them, one may be more significant than others to increase data locality and to reduce remote access in the case of fragmentation. According to the above importance, we can calculate locality precedence of each attribute for each relation and construct ALP table for the relations.

3.4 ALP Table

ALP values of different attributes of a relation will be placed in a table called ALP table. ALP table will be constructed by

database designer for each relation of a DDBMS at the time of designing the database with the help of MCRUD matrix and cost functions. The algorithm that will be used to calculate ALP and to construct ALP table is given in Algorithm I. An example of ALP table for the MCRUD matrix of Fig. 2 is shown in Table 1.

Algorithm I: ALP calculation

```
Input: MCRUD of a relation
Output: ALP table of the
relation
for ( i =1; I <=
TotalAttributes; i++){
      for ( j =1; j <=
TotalPredicates[i]; j++){
           MAX[i][j] = 0;
           for ( k =1; k <=
TotalSites; k++){
for ( r =1; r <=
TotalApplications[k]; r++){  /*
Calculating sum of all
applications' cost of predicate j
of attribute i at site k */
      C[i][j][k][r] = fc*C + fr*R
+ fu*U + fd*D
      S[i][j][k] + =
C[i][j][k][r]
If S[i][j][k] > MAX[i][j] {
/*Find out at which site cost of
predicate j is maximum*/
                        MAX[i][j] =
S[i][j][k]
                        POS[i][j] =
k
                  SumOther = 0
                  Count =0
for ( r =1; r <= A[i][j][k];
r++){
If (r!=k)
SumOther + = S[i][j][r]
If S[i][j][r]>MAX[i][j]/2
/* selecting the sites where
      Replicate[Count]=r
replication of a fragment
                     Count++
                   will be
performed */
           ALPsingle[i][j] =
MAX[i][j] – SumOther
/* actual cost for predicate j
of attribute i */
ALP[i] = 0
for ( j =1; j <=
TotalPredicates[i]; j++)
/*calculating total cost for
attribute i (locality
precedence)*/
           ALP[i] + =
ALPsingle[i][j]
```

Table 1 Example of an ALP table

Entity. Attribute Name	Precedence
$e.a_1$	4
$e.a_2$	8
$e.a_3$	13

3.5 ALP Cost Functions

I treated cost as the effort of access and modification of a particular attribute of a relation to an application from a particular site. For calculating precedence of an attribute of a relation, I take the MCRUD matrix of the relation as an input and use the cost functions of [45], [46].

Using the form of Table 2, more accurate estimation of the frequency of *create*, *read*, *update* and *delete* operation by an application can be possible. This form will be used at the requirement analysis phase of a DDBMS design.

Table 2 Information Need Analysis Form

Access Statistics / Users	Site k			
	Application r			
	$attribute_i$. $predicate_j$			
	Create	Read	Update	Delete
U_1		x		
U_2		x	x	
U_3	x	x	x	x
U_4		x		
. . .				
U_n	x	x	x	

3.6 Fragmentation based on MCRUD Matrix

Here, I am describing MMF technique in details. The main functionalities of the technique are shown in Fig. 3 adopted from [46]. There are *n* numbers of relations in the database named R_1, R_2,…, R_n. First n number of MCRUD matrices will be constructed by the system designer at design time. These *n* matrices will be the input of our technique. Then using the cost functions, *n* number of ALP tables ALP (R_1), ALP (R_2), …, ALP (R_n) will be constructed. Then in the next step, *n* numbers of predicate sets named P_1, P_2, …, P_n will be generated for attributes with highest ALP value for each ALP table. Each predicate set P_i will contain *m* numbers of predicates. According to the predicate sets, each of the *n* relations R_i will be fragmented into m fragments and allocate to the *m* sites.

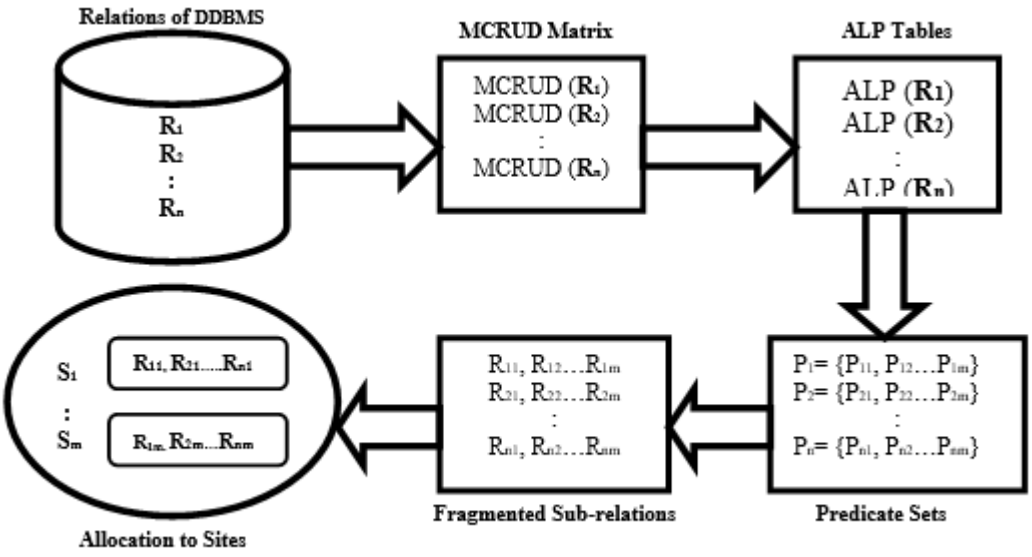

Fig. 3 Block diagram of MMF

Following algorithm, Algorithm II has been used to implement MMF technique.

Algorithm II: FragmentationAllocation

```
Input: Total number of sites: S
```

= {S_1, S_2,… ,S_n}
Relation to be fragmented: R
Modified CRUD matrix: MCRUD[R]
Output: Fragments F = {F_1, F_2, F_3,…, Fn}
Step 1: Construct ALP[R] from MCRUD[R] based on Cost functions
Step 2: For the significant highest valued attribute of ALP table
a. Generate predicate set P={ P_1, P_2, … ,P_m }
b. Fragment R using P as selection predicate

$$\forall_P \mid \sigma_P(R)$$

c. ALLOCATE F to S
*Step 3: For non-significant-highest-value (Max-Highest<1.5*2nd-Highest) in ALP[R]*

$$\sum_{j=1}^{n} S_j$$

a. REPLICATE R to *if R is an entity set*
b. Derive Horizontally Fragment R using its owner relation if R is a relationship set

---

Algorithm II takes a relation to be fragmented, MCRUD matrix of the relation and number of allocation sites as input. It finally produces fragments and allocates them in the sites of DDBMS.

3.7 Implementation of other Fragmentation Types

In this paper, I have performed the fragmentation of the relations of distributed database using horizontal fragmentation technique. This is because of improving performance significantly of a distributed database, we have to maximize locality of data or hit rate of the queries. That is query generating in one site access data of that site only. This will reduce remote access cost and cost of data transfer among the sites. The locality of data can be achieved more using horizontal fragmentation than vertical fragmentation.

MMF technique is not limited to horizontal fragmentation only. If we slightly modify the MCRUD matrix that is if we place attributes of a relation on the row side and applications of the sites of a DDBMS on the column side and modifying the cost functions we can produce vertical fragmentation using MMF technique. Modification of MCRUD matrix for vertical fragmentation is shown in Fig. 4:

Site.Application / Entity.Attribute	Site1			Site2			Site3		
	Ap1	Ap2	Ap3	Ap1	Ap2	Ap3	Ap1	Ap2	Ap3
Accounts .AccountNo	C		RU						R
Accounts.Type	CRD	RU	RUD		R				
. .									
Accounts.Balance	R		R			CRUD			R
Accounts.BrName	CRUD	RU	CRUD			R	R		

Fig. 4 MCRUD Matrix for Vertical Fragmentation

Like other Hybrid or Mixed fragmentation techniques, MF can be performed in our MMF technique by applying vertical fragmentation followed by horizontal fragmentation or vice versa. If it worth mentioning that MF is only applied in distributed databases if the relations have too many attributes and a huge number of records in the relations.

4 Results and Discussion

The objective of my experimental works is to verify the applicability and feasibility of MMF, the proposed fragmentation technique based on MCRUD matrix. The experimental evaluation has been performed with

synthetic data and a reasonable number of queries.

4.1 Experimental Environment

To validate proposed technique, I have implemented a distributed banking database system in the post-graduate lab of BUET using DELL workstations. I have used Windows XP operating system and Oracle 10g for database creation. Schema of the implemented database is shown in Fig. 5.

Customer-Schema = (Cid, Cname, Caddr, Cphn, BrNo)

Loans-Schema = (LnNo, LnType, Amount)

Accounts-Schema = (AccNo, AccType, AccBalance)

Branch-Schema = (BrNo, BrName, BrAddress)

LnCust-Schema = (LnNo, Cid)

AccCust-Schema = (AccNo, Cid)

AccofBranch-Schema = (AccNo, Opendate, Status, BrNo)

LnofBranch-Schema = (LnNo, Issuedate, Status, BrNo)

Fig. 5 Relation schema

Initially number of sites of the distributed system is three as shown in Fig. 6. In each site, three applications were executed.

- Application 1 deals with Customer related information.
- Application 2 deals with Account related information.
- Application 3 deals with Loan related information.

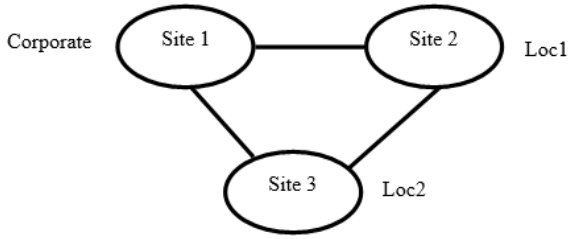

Fig. 6 Distributed banking database system

4.2 Construction of MCRUD Matrix

I have constructed the MCRUD matrix for each of the eight relations of Fig. 5 in the requirement analysis phase. An MCRUD matrix is constructed for each relation by placing predicates of attributes in the row side and applications of the sites of a DDB on the column side of a table in the requirement analysis phase of system development. Two of the matrices constructed are shown in Table 3 - 4.

Table 3 MCRUD matrix of Branch relation

Site.Application	Site1			Site2			Site3		
Entity.Attribute.Predicates	Ap1	Ap2	Ap3	Ap1	Ap2	Ap3	Ap1	Ap2	Ap3
Branch.BrNo=B01	R	R	R					R	
Branch.BrNo=B02			R	R		R			
Branch.BrNo=B03							R		R

Branch.BrName=Corporate	R	R							
Branch.BrName=Loc1				R	R			R	
Branch.BrName=Loc2	R			R			R	R	
Branch.BrAddress=?			R						

Table 4 MCRUD matrix of Loan relation

Site.Application	Site1			Site2			Site3		
Entity.Attribute.Predicates	Ap1	Ap2	Ap3	Ap1	Ap2	Ap3	Ap1	Ap2	Ap3
Loan .LnNo<10000	RU	R	CRUD	RU	R	CRUD	R	R	CRUD
Loan .LnNo>=10000	R	R	CRUD	R	RU	CRUD	R	RU	CRUD
Loan.LnType=SME	R		RU	RU	R	CRUD	R		RU
Loan.LnType=HOME	RU	RU	CRUD	R		RU	R		RU
Loan.LnType=CAR	R		RU	R		RU	RU		CRUD
Loan.Amount<50000	R		CRUD	R		CRUD	R		CRUD
Loan.Amount=50000:100000	R	R	CRUD	R		CRUD	R		CRUD
Loan.Amount>100000	R		CRUD	R		CRUD	R		CRUD

4.3 Calculation of ALP Values and Construction of ALP Tables

I have calculated locality precedence of each attribute from the MCRUD matrix of each relation using attribute locality precedence (ALP) calculation algorithm. Using the ALP values I have constructed ALP table for each relation. ALP table is a 2D array where attributes of a relation and its locality precedence is stored. For each attribute, Create, Read, Update, and Delete operation over its predicates from different applications of different sites is calculated and sum up to have locality precedence of that attribute. An attribute with the highest precedence implies that taking predicates of this attribute as selection predicate for horizontal fragmentation will maximize the hit ratio. It is depicted in Table 5.

Table 5 Precedence calculation for LnType attribute of Loan relation

Attribute Name	Predicates	Precedence in Site 1	Precedence in Site 2	Precedence in Site 3	Precedence of Predicate	ALP	Decision
LnType	LnType = SME	5	13	5	13-5-5=3	3+6+2=11	Fragment in Site 2
	LnType = HOME	16	5	5	16-5-5=6		Fragment in Site 1
	LnType = CAR	5	5	12	12-5-5=2		Fragment in Site 3

Table 6 ALP table of Loan relation

Attribute Name	Precedence
LnNo	-20
LnType	11
LnAmount	-26

4.4 Generation of Predicate Set and Fragmentation of the Relations

Predicate set was generated for the attributes with highest locality precedence of the relations respectively. These predicate sets were used to fragment the relations.

P_{Loan} ={LnType=SME, LnType=HOME , LnType=CAR }
$P_{Customer}$ ={BrNo=B01, BrNo=B02, BrNo=B03}
$P_{Accounts}$ ={AccType=Ind, AccType=Cor}
$P_{AccofBranch}$ ={BrNo=B01, BrNo=B02, BrNo=B03}
$P_{LnofBranch}$ ={BrNo=B01, BrNo=B02, BrNo=B03}

As for AccCust and LnCust relations, no attribute has significant higher precedence than other attributes, so predicate set was not generated for the relations. Instead these relations are to be fragmented derived horizontally with the help of their mother relation.

For Horizontal fragmentation of Customer relation, following queries are used:

$Q_{Customer1}$ =Select * from Customer where BrNo=B01;
$Q_{Customer2}$ =Select * from Customer where BrNo=B02;
$Q_{Customer3}$ =Select * from Customer where BrNo=B03;

For Horizontal fragmentation of Loan relation, following queries are used:

Q_{Loan1} =Select * from Loan where LnType=SME;
Q_{Loan2} =Select * from Loan where LnType= HOME;
Q_{Loan3} =Select * from Loan where LnType= CAR;

For Horizontal fragmentation of Accounts relation, following queries are used:

$Q_{Accounts1}$ =Select * from Accounts where AccType=Ind;
$Q_{Accounts2}$ =Select * from Accounts where AccType=Cor;

For Horizontal fragmentation of AccofBranch relation, following queries are used:

$Q_{AccofBranch1}$ =Select * from AccofBranch where BrNo=B01;
$Q_{AccofBranch2}$ =Select * from AccofBranch where BrNo=B02;
$Q_{AccofBranch3}$ =Select * from AccofBranch where BrNo=B03;

For Horizontal fragmentation of LnofBranch relation following queries are used:

$Q_{LnofBranch1}$ =Select * from LnofBranch where BrNo=B01;
$Q_{LnofBranch2}$ =Select * from LnofBranch where BrNo=B02;
$Q_{LnofBranch3}$ =Select * from LnofBranch where BrNo=B03;

For Horizontal fragmentation of AccCust relation, following queries are used:

$Q_{AccCust1}$ =Select AccNo, Cid from AccCust, Customer where AccCust.Cid = Customer.Cid and Customer.BrNo=B01;
$Q_{AccCust2}$ =Select AccNo, Cid from AccCust, Customer where AccCust.Cid = Customer.Cid and Customer.BrNo=B02;
$Q_{AccCust3}$ =Select AccNo, Cid from AccCust, Customer where AccCust.Cid = Customer.Cid and Customer.BrNo=B03;

For Horizontal fragmentation of LnCust relation, following queries are used:

$Q_{LnCust1}$ =Select LnNo, Cid from LnCust, Customer where LnCust.Cid = Customer.Cid and Customer.BrNo=B01;
$Q_{LnCust2}$ = Select LnNo, Cid from LnCust, Customer where LnCust.Cid = Customer.Cid and Customer.BrNo=B02;
$Q_{LnCust3}$ = Select LnNo, Cid from LnCust, Customer where LnCust.Cid = Customer.Cid and Customer.BrNo=B03;

Branch relation was not fragmented as it is a very small relation and most access to its records is by *read* operation. Instead, Branch relation was replicated to all the sites of the DBDS.

In this way all the relation schemas of the distributed banking system of Fig. 5 were fragmented using the above queries and allocated to the three computers (sites).

4.5 Queries for Performance analysis of Matrix based Fragmentation (MMF)

I have executed twenty queries in each site with a total of sixty selected queries in the distributed system according to Pareto Principle often referred as 80/20 rule [47] – [49] to see the performance of MMF. The queries were selected from the following *query domain* to accomplish enough variation of real database system:

- Insertion e.g. Insert into RRR values (xxx, yyy, zzz);
- Selection (Point) e.g. Select A_1, $A_{2...}$

A_n from RRR where xxx= P

- Selection (Range) e.g. Select A_1, A_2... A_n from RRR where xxx< BBB
- Selection (Join) e.g. Select A_1, A_2 ... A_n from R_1, R_2 where $R_1.A_i=R_2.A_j$ AND $R_1.A_k$=CCC
- Selection (Aggregation) e.g. Select Sum (AA) from RRR where P
- Update e.g. Update RRR set A_i = xxx where A_j = yyy
- Deletion e.g. Delete * from RRR where P

I have defined hit as a result of a query of any type accessed records of a local fragment of the site where the query was initiated and miss as a result of a query of any type accessed records of one or more remote fragments of other sites. Partial results of my experiments are shown in Table 7 – 8 and Fig. 7 – 8:

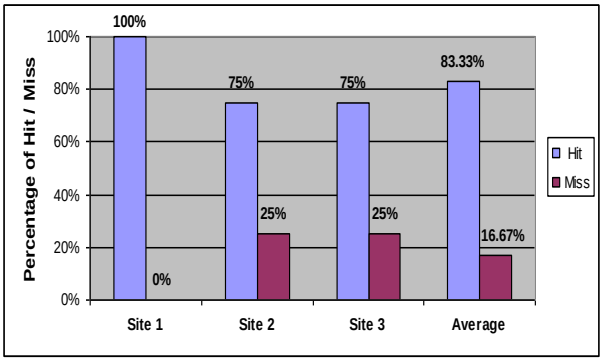

Fig. 7 Hit Miss ratio for Loan relation

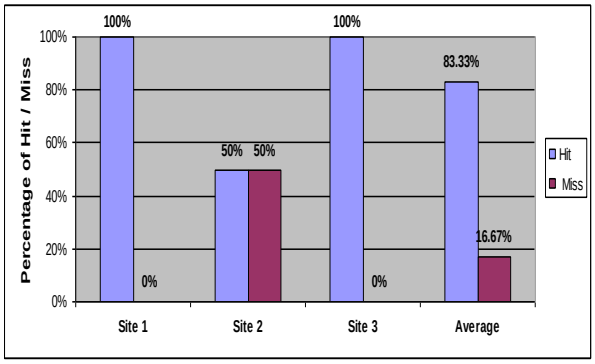

Fig. 8 Hit Miss ratio for Accounts relation

From Table 7 we can see that all the queries of Site 1 accessed records from local fragment of Loan relation. So hit ratio in Site 1 is 100%. We also see that 75% queries executed at Site 2 and Site 3 accessed records of local fragment and 25% queries accessed records of fragment stored on other (remote) site rather than query generation site. Average hit ratio for Loan relation is 83.33%.

Table 7 Hit Miss ratio for Loan

Site	Percentage of Hit	Percentage of Miss
Site 1	100%	0%
Site 2	75%	25%
Site 3	75%	25%
Average	83.33%	16.67%

From Table 8 we can see that all the queries of Site 1 and Site 3 accessed local fragment of Accounts relation.
So hit ratio in Site 1 and 3 is 100%. 50% of queries executed at Site 2 accessed records of the local fragment and 50% queries accessed records of fragment stored on other (remote) site rather than Site 2. Average hit ratio for Accounts relation is 83.33%.

Table 8 Hit Miss ratio for Accounts

Site	Percentage of Hit	Percentage of Miss
Site 1	100%	0%
Site 2	50%	50%
Site 3	100%	0%
Average	83.33%	16.67%

Table 9 shows the overall performance of the distributed system after fragmenting the relations using MMF technique. We can see that after fragmentation and allocation using MMF technique, 85.71% of the queries generated in any site accessed records of only that site and remote access reduced to 14.29%. This is definitely a significant achievement.

Table 9 Overall System Performances of MMF

Site Name	Queries executed	Accessed fragment stored in local site	Accessed fragment stored in remote site	Percentage of Hit	Percentage of Miss
Site 1	20	19	1	92.86 %	7.14 %
Site 2	20	16	3	78.57%	21.43%
Site 3	20	17	2	85.71%	14.29%
DDBMS	60	52	6	86.6 %	13.4%

4.6 Comparison with other Techniques

I have named the techniques deals with fragmentation problem of distributed database without addressing the initial stage problem as Techniques Without Initial Fragmentation (TWIF) as in [1] – [30]. TWIF first store the relations of a distributed database in a single site of the distributed system as a centralized database. The other sites where the database is not stored, access the database with various type of queries using remote network connection of the system. Information about attribute, predicate access pattern and frequencies of access by different queries from different sites are gathered in tables called Attribute Usage Matrix (AUM) or Predicate Usage Matrix (PUM) or similar tables. After a certain period when sufficient statistical data are gathered for calculating the relationship (known as affinity) of an attribute or predicate with the transaction of sites, Attribute Affinity Matrix (AAM) or Predicate Affinity Matrix (PAM) are generated using Bond Energy algorithm or similar algorithm. From AAM and PAM, vertical and horizontal fragmentation decision is made respectively. Then produced fragments are to be stored in the sites of the distributed database though almost all TWIF ignore allocation of the fragments to reduce complexity.
I have implemented the above model in the lab and execute the same forty-two queries those were used to test our technique with the assumption that at the initial stage the centralized database is stored at Site 1. Table 10 shows the overall system performance of

TWIF before DDBMS is fragmented and allocated to sites. We can see that during a long period before reasonable amount of statistical record access frequencies by transactions are available for constructing attribute affinity matrix or predicate affinity matrix and to fragment and allocate the database among the three sites, percentage of hit of the overall system is only 33.33% which is much less in comparison with our achieved 85.71% hit rate.

Table 10 Overall System Performance of TWIF

Site Name	Queries executed	Access fragment stored in local site	Access fragment stored in remote site	Percentage of Hit	Percentage of Miss
Site 1	20	20	0	100%	0%
Site 2	20	0	14	0%	100%
Site 3	20	0	14	0%	100%
DDBMS	60	20	28	33.33%	66.66%

4.7 Impact of Site Number Increase

Now we want to experiment the generalization of MMF so that we can verify if our technique is applicable to any number of sites of distributed system. I have increased a total number of sites to four at design time by adding a local branch of DBDB named Loc3 at Site 4. This situation is depicted in Fig. 9

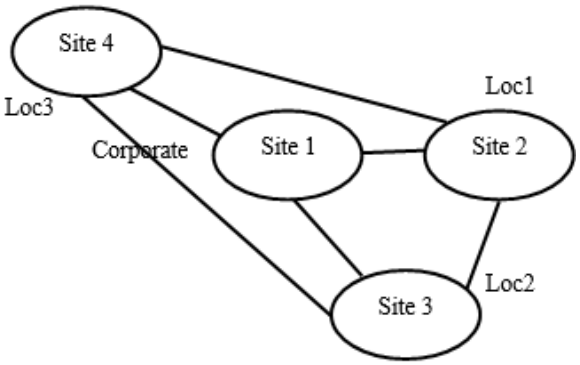

Fig. 9 DBDB with four sites

I have constructed the MCRUD matrix of Loan relation for four sites with three applications running on each site. It is shown in Table 11 below:

Table 11 MCRUD matrix of Loan relation for four sites

Site.Application	Site1			Site2			Site3			Site4		
Entity.Attribute.Predicates	Ap1	Ap2	Ap3	Ap1	Ap2	Ap3	Ap1	Ap2	Ap3	Ap1	Ap2	Ap3
Loan .LnNo<10000	RU	R	CRUD	RU	R	CRUD	R	R	CRUD	RU	R	CRUD
Loan .LnNo>=10000	R	R	CRUD	R	RU	CRUD	R	RU	CRUD	R	RU	CRUD
Loan.LnType=SME	R		RU	RU	R	CRUD	R		RU	RU		CRUD
Loan.LnType=HOME	RU	RU	CRUD	R		RU	R		RU	R		RU
Loan.LnType=CAR	R		RU	R		RU	RU		CRUD	RU	R	CRUD

Loan.Amount<50000	R		CRUD	R		CRUD	R		CRUD	RU	R	CRUD
Loan.Amount=50000:100000	R	R	CRUD	R		CRUD	R		CRUD	R	R	CRUD
Loan.Amount>100000	RU	R	CRUD	R		CRUD	R		CRUD	R		RU

From Table 11, I have calculated ALP table for Loan relation shown in Table 12. The process of how fragmentation and replication decision is made in four sites can be understood from Table 13.

Table 12 ALP table of Loan relation with four sites

Attribute Name	Precedence
LnNo	-46
LnType	-17
LnAmount	-42

Table 13 Precedence calculation and fragmentation decision for Loan relation

Attribute Name	Predicates	Precedence in Site 1	Precedence in Site 2	Precedence in Site 3	Precedence in Site 4	Decision
LnType	LnType = SME	5	13	5	12	Fragment in Site 2 Replica in site 4
	LnType = HOME	16	5	5	5	Fragment in Site 1
	LnType = CAR	5	5	12	13	Fragment in Site 4 Replica in site 3

Predicate set is generated for the attribute LnType of Loan relation.

P_{Loan} ={LnType=SME, LnType=HOME , LnType=CAR }

For Horizontal fragmentation of Loan relation, following queries were used:

Q_{Loan1} =Select * from Loan where LnType=HOME;

Q_{Loan2} =Select * from Loan where LnType= SME;

Q_{Loan3} =Select * from Loan where LnType= CAR;

$Q_{Loan4.1}$=Select * from Loan where LnType=SME;

$Q_{Loan4.2}$ =Select * from Loan where LnType= CAR;

I have executed same queries as previous in four sites of DBDS to check the impact of site addition on the hit-miss ratio. The result is shown in Table 14 and Fig 10.

We can see that average hit ratio is 81.25% that is very close to our previous result 83.33% achieved for three sites.

Table 14 Performance table of MMF for Loan relation distributed in four sites

Site	Percentage of Hit	Percentage of Miss
Site 1	100%	0%
Site 2	75%	25%
Site 3	75%	25%
Site 4	75%	25%
Average	81.25%	19.75%

Fig. 10 shows the performance of MMF and TWIF with the increase of a number of sites in the distributed system. We can see that MMF shows better and quite steady performance as sites increases from three to five. In the same time, performance of TWIF falls gradually as new sites are added to the system.

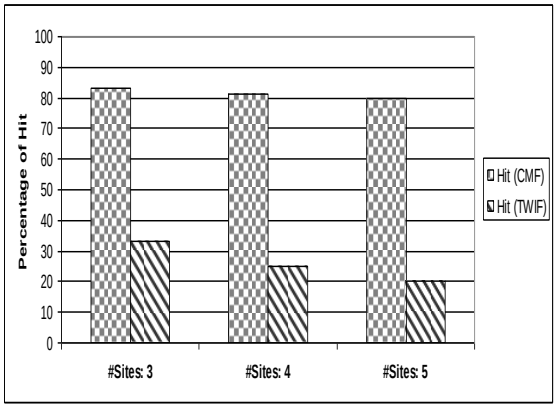

Fig. 10 Comparison of Hit ratio between MMF and TWIF with increasing number of sites

4.8 Summary

From the above result, we can see that our technique has clearly outperformed the technique stated in [30]. Our fragmentation technique achieved a very good hit rate which is approximately 84%. As other techniques described in [1] – [29] could not provide solutions for initial state of the distributed system, using TWIF initial performance (hit ratio) of the system is only 33.33%. After a long period when sufficient data for fragmenting the centralize database were available, hit rate of TWIF increased significantly as much as 91.66% but in the price of high transfer cost incurred for transferring data among the sites of the distributed system using communication network. Another thing is to mention that MMF achieves a steady hit rate over 80% and TWIF's performance falls gradually from 33.33% to 20% with the increase of a number of sites of DBDS from three to five. I have increased the number of sites in the system up to ten and found similar results.

7 Conclusions

Making proper fragmentation of the relations and allocation of the fragments is a major research area in distributed systems. Many techniques have been proposed by the researchers using empirical knowledge of data access by different queries and frequencies of queries executed in different sites of a distributed system. But proper fragmentation and allocation at the initial stage of a distributed database have not yet been addressed. In this paper, I have presented a fragmentation technique to partition relations of a distributed database properly at the initial stage when no data access statistics and query execution frequencies are available. Instead of using empirical data, I have developed a matrix namely Modified Create, Read, Update and

Delete (MCRUD) to make fragmentation decisions. Using our technique no additional complexity is added for allocating the fragments to the sites of a distributed database as fragmentation is synchronized with allocation. So the performance of a DDBMS can be improved significantly by avoiding frequent remote access and high data transfer among the sites.

Acknowledgment
The Author is grateful to Dr. Abu Sayed Md. Latiful Hoque, Professor, Dept. of Computer Science and Engineering, Bangladesh University of Engineering and Technology (BUET) for his guidance.

References

[1] Ozsu, M. T., and Valduriez, P. Principles of Distributed Database Systems. Prentice-Hall, New Jersey, 1999.
[2] Ceri, S., and Pelagatti, G. Distributed Databases Principles and System. McGraw- Hill, New York, 1984.
[3] Navathe, S., Karlapalem, K., and Ra, M. A mixed fragmentation methodology for initial distributed database design. Journal of Computer and Software Engineering 3, 4 (1995), 395–426.
[4] Bai˜ ao, F., Mattoso, M., and Zaverucha, G. A distribution design methodology for object dbms. Distributed and Parallel Databases 16, 1 (2004), 45–90.
[5] Tamhankar, A. M., and Ram, S. Database fragmentation and allocation: An integrated methodology and case study. IEEE Transactions on Systems Management 28, 3 (1998), 194–207.
[6] Blankinship, R., Hevner, A. R., and Yao, S. B. An iterative method for distributed database design. In Proceedings of the 17th International Conference on Very Large Data Bases (1991), G. M. Lohman, A. Sernadas, and R. Camps, Eds., Morgan Kaufmann, pp. 389–400.
[7] Ma, H. "Distribution design for complex value databases" Doctoral thesis, Department of Information Systems, Massey University, 2007.
[8] Ceri, S., Negri, M., and Pelagatti, G. Horizontal data partitioning in database design. In Proceedings of the 1982 ACM SIGMOD international conference on Management of data (1982), ACM Press, pp. 128–136.
[9] Zhang, Y. On horizontal fragmentation of distributed database design. In Advances in Database Research (1993), M. Orlowska and M. Papazoglou, Eds., World Scientific Publishing, pp. 121–130.
[10] Ra, M. Horizontal partitioning for distributed database design. In Advances in Database Research (1993), M. Orlowska and M. Papazoglou, Eds., World Scientific Publishing, pp. 101–120.
[11] Cheng, C.-H., Lee, W.-K., and Wong, K.-F. A genetic algorithm-based clustering approach for database partitioning. IEEE Transactions on Systems, Man, and Cybernetics, Part C 32, 3 (2002), 215–230.
[12] Mahboubi H. and Darmont J. "Enhancing XML Data Warehouse Query Performance by Fragmentation" ACM SAC, 2009, pp.1555-1562.
[13] Chang, S.-K., and Cheng, W.-H. A methodology for structured database decomposition. IEEE Transactions on Software Engineering (TSE) 6, 2 (1980), 205–218.
[14] Shin, D. G., and Irani, K. B. Partitioning a relational database horizontally using a knowledge-based approach. SIGMOD Record 14, 4 (1985), 95–105.
[15] Shin, D.-G., and Irani, K. B. Fragmenting relations horizontally using a knowledgebased approach. IEEE Transactions on Software Engineering (TSE) 17, 9 (1991), 872–883.
[16] Khalil, N., Eid, D., and Khair, M. Availability and reliability issues in distributed databases using optimal horizontal fragmentation. In Database and Expert Systems Applications (DEXA) (1999), T. J. M. Bench-Capon, G. Soda, and A. M. Tjoa, Eds.,vol. 1677

of Lecture Notes in Computer Science, Springer, pp. 771–780.

[17] Hoffer, J. A., and Severance, D. G. The use of cluster analysis in physical database design. In Proceedings of the First International Conference on Very Large Data Bases (VLDB) (1975), pp. 69–86.

[18] Navathe, S. B., Ceri, S., Wiederhold, G., and Dour, J. Vertical partitioning algorithms for database design. ACM Transactions on Database Systems (TODS) 9, 4 (1984), 680–710.

[19] Navathe, S. B., and Ra, M. Vertical partitioning for database design: A graphical algorithm. SIGMOD Record 14, 4 (1989), 440–450.

[20] Lin, X., and Zhang, Y. A new graphical method of vertical partitioning in database design. In Proceedings of the 4th Australian Database Conference (ADC) (1993), M. P. P. Maria E. Orlowska, Ed., World Scientific, pp. 131–144.

[21] Ma, H., Schewe, K.-D., and Kirchberg, M. A heuristic approach to vertical fragmentation incorporating query information. In Proceedings of the 7th International Baltic Conference on Databases and Information Systems (DB&IS), IEEE Computer Society Press, 2006, pp. 69–76.

[22] AlFares M. et al, "Vertical Partitioning for Database Design: A Grouping Algorithm", International Conference on Software Engineering and Data Engineering (SEDE), 2007, pp. 218-223.

[23] Ngo T.H., "New Objective Function for Vertical Partitioning in Database System" In Proceedings of the SYRCODIS, 2008.

[24] Runceanu A. "Fragmentation in Distributed Databases", Innovations and Advanced Techniques in Systems, Computing Sciences and Software Engineering, Springer, 2008, pp. 57–62.

[25] Cornell, D., and Yu, P. A vertical partitioning algorithm for relational databases. In International Conference on Data Engineering (1987), pp. 30–35.

[26] Cornell, D. W., and Yu, P. S. An effective approach to vertical partitioning for physical design of relational databases. IEEE Transactions on Software Engineering 16, 2 (1990), 248–258.

[27] Chu, P.-C. A transaction oriented approach to attribute partitioning. Information Systems 17, 4 (1992), 329–342.

[28] Chakravarthy, S., Muthuraj, J., Varadarajan, R., and Navathe, S. B. An objective function for vertically partitioning relations in distributed databases and its analysis. Distributed and Parallel Databases 2, 2 (1994), 183–207.

[29] Son, J. H., and Kim, M. H. An adaptable vertical partitioning method in distributed systems. Journal of Systems and Software 73, 3 (2004), 551–561.

[30] Abuelyaman E. S. "An Optimized Scheme for Vertical Partitioning of a Distributed Database" International Journal of Computer Science and Network Security, VOL.8 No.1, January 2008, pp 310-316.

[31] Raouf, A. E. A., Badr, N. L., & Tolba, M. F. (2014). Dynamic distributed database over cloud environment. In International Conference on Advanced Machine Learning Technologies and Applications (pp. 67-76). Springer International Publishing.

[32] Rodríguez-Mazahua, L., Alor-Hernández, G., Abud-Figueroa, M. A., & Peláez-Camarena, S. G. (2014). Horizontal Partitioning of Multimedia Databases Using Hierarchical Agglomerative Clustering. In Mexican International Conference on Artificial Intelligence (pp. 296-309). Springer International Publishing.

[33] Harikumar, S., & Ramachandran, R. (2015). Hybridized fragmentation of very large databases using clustering. In Signal Processing, Informatics, Communication and Energy Systems (SPICES), 2015 IEEE International Conference on (pp. 1-5). IEEE.

[34] Hababeh, I., Khalil, I., & Khreishah, A. (2015). Designing high performance web-based computing services to promote telemedicine database management system. IEEE Transactions on Services Computing, 8(1), 47-64.
[35] Hess, H. (2016). Evaluating Domain-Driven Design for Refactoring Existing Information Systems (Doctoral dissertation, Ulm University).
[36] Bai˜ ao, F., and Mattoso, M. A mixed fragmentation algorithm for distributed object oriented databases. In Proceedings of the International Conference Computing and Information (ICCI) (1998), pp. 141–148.
[37] Chu, W. W. Optimal file allocation in a multiple computer system. IEEE Transactions on Computers 18, 10 (1969), 885–889.
[38] Casey, R. G. Allocation of copies of files in an information network. In Proceedings of AFIPS SJCC (1972), vol. 40, AFIPS Press, pp. 617–625.
[39] Mahmoud, S., and Riordon, J. S. Optimal allocation of resources in distributed information networks. ACM Transactions on Database Systems (TODS) 1, 1 (1976), 66–78.
[40] Ahmad, I., Karlapalem, K., Kwok, Y.-K., and So, S.-K. Evolutionary algorithms for allocating data in distributed database systems. Distributed Parallel Databases 11, 1 (2002), 5–32.
[41] Menon, M.-S. Allocating fragments in distributed databases. IEEE Transactions on Parallel and Distributed Systems 16, 7 (2005), 577–585.
[42] Surmsuk P. "The integrated strategic information system planning methodology", IEEE Computer Society Press, 2007, pp. 467-475.
[43] Whitten J. et al. "Systems Analysis and Design Methods", 6th Ed. McGraw-Hill, 2004.
[44] Wiegers K. E. Software Requirements, 2nd Edition, Microsoft Publication, 2003.
[45] Khan, S. I., and Hoque, A. S. M. L. (2010). A new technique for database fragmentation in distributed systems. International Journal of Computer Applications, 5(9), 20-24.
[46] Khan, S. I., and Hoque, A. S.M. L. (2012). Scalability and performance analysis of CRUD matrix based fragmentation technique for distributed database. In Computer and Information Technology (ICCIT), 2012 15th International Conference on (pp. 567-562). IEEE.
[47] Pareto principle, accessed from http://www. en.wikipedia.org/wiki/Pareto_principle.
[48] Craig S. Mullins "Defining Database Performance", accessed from http://www.craigsmullins.com/cnr_db.htm.
[49] Fritchey G. and Dam S. "SQL Server 2008 Query Performance Tuning Distilled", 1st Ed. Apress, 2009.

Shahidul Islam Khan obtained his B.Sc. and M.Sc. Engineering Degree in Computer Science and Engineering (CSE) from Ahsanullah University of Science and Technology (AUST) and Bangladesh University of Engineering and Technology (BUET), Dhaka, Bangladesh in 2003 and 2011. He is now a Ph.D. candidate in the Department of CSE, BUET, which is the highest ranked technical university of Bangladesh. His current fields of research are database systems, data mining, and health informatics. He has twenty published papers in referred journals and conferences. He is also an Associate Professor (Currently in study leave) in the Dept. of CSE, International Islamic University Chittagong (IIUC), Bangladesh.