Volume 5, Issue 2, February 2015 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Polynomial Based Database Transfer Technique

[1]Siddharth Srivastava*, [2]Shashank Saxena, [3]Rishabh Trikha

[1],Lecturer, Department of Computer Science, Central Institute of Plastics Engineering and Technology, Lucknow. Uttar Pradesh Technical University, Lucknow, U.P., India

[2, 3] Student, Department of Computer Science, National P.G. College, Lucknow, An Autonomous College of Lucknow University, Lucknow, U.P., India

Abstract: In this journal we will describe a technique entitled "Polynomial Based Database Transfer Technique". As we know that database contains large volume of data in it and sometimes this volume may increase to gigabytes or terabytes or more. So in some scenario if we have to transfer such huge amount of data from one system to another then it is not an armchair job. Here in this journal we will describe a technique using which this problem can be minimized drastically. We have also developed software regarding the same but we tested it for small databases. Actually accuracy and efficiency of this technique is based on mathematical theorems coded to construct polynomial using which the final database can easily be coded in polynomial form and on receivers side using this polynomial database can easily be regained. Using this technique database can easily and securely be transferred from one system to another that to in encrypted form.

Keywords:- Mapping Function, LZ77, Compression, Index Number.

I. INTRODUCTION

In this technique entitled "Polynomial Based Database Transfer Technique" we convert data present in database in form of polynomial equation and this equation is transferred from sender system to destination system. Once polynomial equation reaches destination system than its roots are extracted and replica of original database is reconstructed again on destination system.

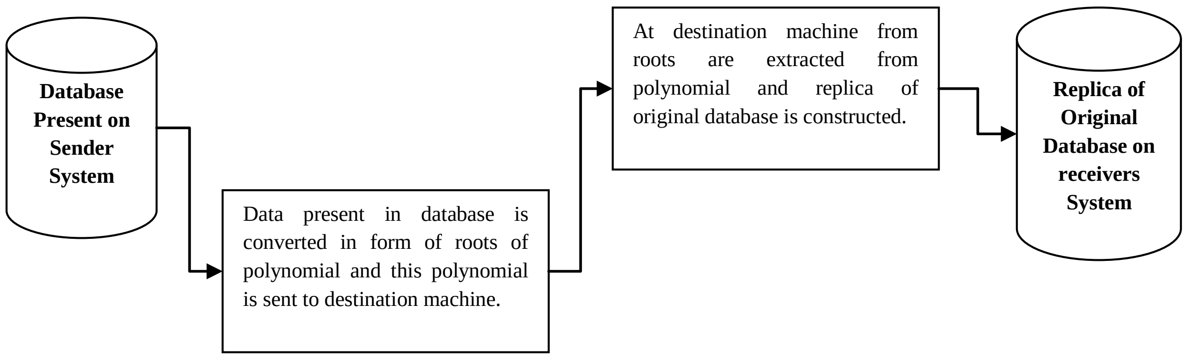
Database Present on Sender System
Data present in database is converted in form of roots of polynomial and this polynomial is sent to destination machine.
At destination machine from roots are extracted from polynomial and replica of original database is constructed.
Replica of Original Database on receivers System

Figure. 1 Showing abstract concept of Polynomial Based Database Transfer Technique.

Figure 1.shown above shows the abstract concept of our technique. In reality we have designed this technique piece by piece where each piece is devoted to some particular issue of database. We will describe these pieces or phases later in this journal.

II. PROBLEMS WITH DATABASE TRANSFER

As we all know that this age is known as information age. So in this age information is everything and we need database to store this tremendous amount of data from which we can easily infer desired information. Now in real world scenario conditions may arise when we want to transfer some data or entire database from one system to another.

If only few amount of data is to be transmitted between systems than no issues regarding transfer arises assuming that data to be transmitted do not require any security. But if we want to transmit entire database, than time is the biggest issue. Since databases may occupy gigabytes of space in memory so transferring entire database may be very time consuming task and if we want to securely transfer this database than scenario becomes much worse since now we have to perform encryption on database first, than we can transfer it from one system to another. In order to eliminate these two issues namely security and time we have constructed this technique.

III. INTRODUCTION TO POLYNOMIAL BASED DATABASE TRANSFER TECHNIQUE

Now let's start off with our core technique. This technique is basically developed phase by phase where each phase is devoted towards one specific issue of database. Here we will explain our technique phase by phase. The abstract view of each phase is given in figure shown below.

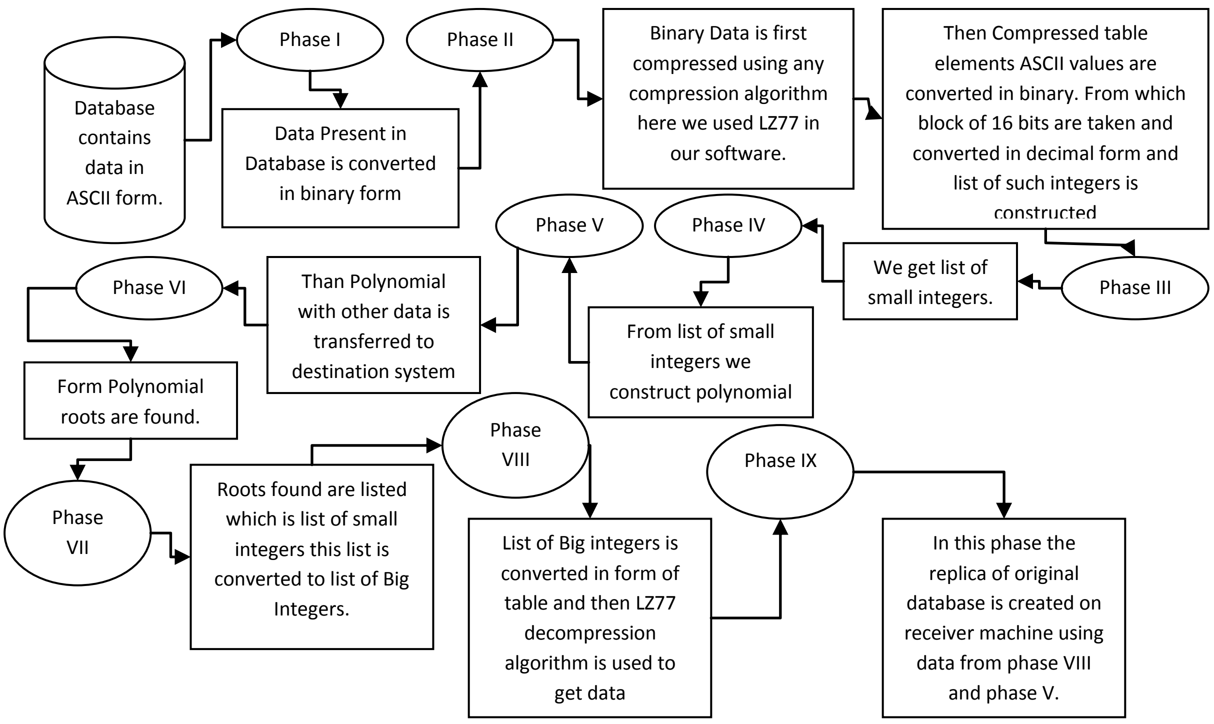
Database contains data in ASCII form.
Phase I
Data Present in Database is converted in binary form
Phase II
Binary Data is first compressed using any compression algorithm here we used LZ77 in our software.
Then Compressed table elements ASCII values are converted in binary. From which block of 16 bits are taken and converted in decimal form and list of such integers is constructed
Phase III
We get list of small integers.
Phase IV
From list of small integers we construct polynomial
Phase V
Than Polynomial with other data is transferred to destination system
Phase VI
Form Polynomial roots are found.
Phase VII
Roots found are listed which is list of small integers this list is converted to list of Big Integers.
Phase VIII
List of Big integers is converted in form of table and then LZ77 decompression algorithm is used to get data
Phase IX
In this phase the replica of original database is created on receiver machine using data from phase VIII and phase V.

Figure 2. Showing details of each phases involved in polynomial based database transfer technique.

A. Phase I (Conversion)

In this phase data present in database is converted to its binary equivalent. As we know that data present in database is composed of alphabets, digits and special symbols generally and these characters are stored using ASCII codes. So in this phase the binary equivalent of each character is regained. Once this much is done than we move to second phase which is compression.

B. Phase II (Compression)

In this phase the data present in binary form is compressed using standard or user defined compression technique. This we did because binary lossless compression is more effective than compression of characters. Moreover in our software we have used LZ77 compression algorithm. Once compression is done than we get table at the end, we than consider that elements of table are in ASCII code and accordingly convert them in binary again. Once we did this than we divide binary elements in 16 bit blocks and convert them in form of integers.

C. Phase III (Mapping Function)

The list of integers formed above contains big integers. When this phase is executed than big integers are mapped to small and unique integers. So output of this phase is list of small integers using which polynomial can easily be formed.

D. Phase IV(Polynomial Function)

Using mapped integers we create a polynomial where all mapped integers becomes the root of the polynomial. This polynomial is passed to destination machine where the above process is reversed and data is regained.

E. Phase V (Transfer)

The polynomial formed in above phase is transferred to destination system. Some other data is also transferred during this phase which we will tell later.

F. Phase VI (Decrypting Polynomial)

In this phase on destination system the received polynomial is decrypted and all roots are found. These roots are nothing but the mapped integers which are formed in phase III.

G. Phase VII (Reverse Mapping Function)

When this phase is executed than first of all the list of small integer is constructed from roots than this list is converted to list of big integers.

H. Phase VIII (Decompression)

In this phase the table formed in above phase is decompressed and the data which is in binary form is regained.

I. Phase IX (Regeneration)

In this phase, binary data regained from above phase is converted to ASCII form and real data is regained from which the database is regenerated. This is how the database is transferred from one system to another.
These are nine phases which collectively form our algorithm. Now let's explain our algorithm in depth.

IV. EXPLANATION OF ALGORITHM

A. Phase I (Conversion)

Suppose given database is known as **Test** and it contains only one table say **Student** with records as shown below.

Table 1. Showing table "*Student*" present in database "*Test*".

Student	
Name	**Age**
AB	21
CD	4

Nowwhen phase one is executed on this table than all data present in table is converted to its binary equivalent as shown below. This conversion is accomplished using ASCII values assigned to characters which are stored in database. For example A's ASCII value is 65 so A's binary equivalent is 10000001 which is binary equivalent of A. So using this concept binary data which we get at the end of this phase is as shown below.

Table 2. Showing data of above table in binary form.

0100000101000001001110110011001000110001001110110010000110100010001110 1100110100001111110

One can easily note that binary data shown above contains more bits than its binary equivalent. This happens because we have embedded some special characters in data using which data can easily be regained at destination machine.
Logically we can say that our table is converted in binary form as shown below.

Table 3. Logical representation of table "*Student*" after execution of phase I.

Student	
Name	**Age**
0100000101000010	***0110010001100010***
0100001101000100	***01101000***

B. Phase II (Compression)

The data which we get in above phase is compressed using LZ77 algorithm and table is formed, this table holds data in compressed form. Compressed table is shown below in Table 4. Any compression algorithm can be used but to make our software simple we have implemented LZ77 the most basic compression algorithm.

Table 4. Compressed Table after application of LZ77 compression algorithm.

Steps	**Input Bit**	**Tokens**
1	0	(0,0)
2	1	(0,1)
3	0	(1,0)
4	0	(3,0)
5	10	(2,0)
6	100	(5,0)
7	1	(3,1)
8	11	(7,1)
9	101	(5,1)
10	1001	(6,1)
11	10010	(10,0)
12	110	(8,0)
13	10	(7,0)
14	1	(1,1)
15	11	(2,1)
16	11	(14,1)
17	100	(13,0)
18	1101	(12,1)
19	1	(4,1)
20	11	(19,1)
21	1011	(9,1)
22	11010	(18,0)
23	111	(20,1)
24	111	(15,1)
25	0	(0,0)

Once we get archived data in tabular form than we take ***Tokens***column of table and we find ASCII value of each element present in column under consideration. Than convert it into binary form as shown below (data shown below represents only first two rows of Token column).

Table 5. Showing partial data of Tokens column of compressed table in binary form.

01010000011000000101100001100000010100100101000001100000010110000110001001010010.......... and so on.

Then we make block of 16 bits and convert it to integer as shown below.

Table 6. Data shown in Table 5 is packed into block of 16 bits and then decimal equivalent of each block is found.

$\mathbf{0101000001100000_{(2)}=20576_{(10)}}$

$\mathbf{0101100001100000_{(2)}=22624_{(10)}}$

$\mathbf{0101001001010000_{(2)}=21072_{(10)}}$

$\mathbf{0110000001011000_{(2)}=24664_{(10)}}$

$\mathbf{0110001001010010_{(2)}=25170_{(10)}}$

.... and so on.

B 1) Index number assignment

The integers formed above are allotted number which is known as index number using which destination machine can easily regain the replica of original database. These numbers are allotted as shown below.

Table 7. Data shown in Table 6 is packed into block of 16 bits and then decimal equivalent of each block is found.

Binaries	**Integers**	**Index Number**
0101000001100000	**20576**	**1**
0101100001100000	**22624**	**2**
0101001001010000	**21072**	**3**
0110000001011000	**24664**	**4**
0110001001010010	**25170**	**5**
and so on		

C. Phase III (Mapping Function)

In this phase we construct a mapping function using which big integers formed in above step are mapped to small and unique integers. To do this we follow following algorithm which we named mapping function. **This algorithm uses structure which isshown below.**

```
structmapping_function_node
{longint integer;
intindex_number;
intrecursive_step;
};
```

```
Algorithm_mapping_function()
{Step 0: Construct list which contains all the integers created in phase II.Take variable i=0;
 Step 1: From list select the minimum value integer say $min_i$ now subtract each integer in list with this $min_i$.
Step 2: For each row in list where value is zero construct a node of structure mapping_function_nodemfn; and assign
mfn->integer=$min_i$;
mfn->index_number=index_number_of_integer as per phase II;
mfn->recursive_step=i;
and make it the node of doubly linked list. Increment i by one.
step 3: now for all rows present in the list except rows where integer values are zero repeat step 1 to 3 until small integers are not gained.
Step 4: Once we get integers of required size than goto phase IV.
}
```

D. Phase IV (Polynomial Function)

In this phase the integers which we obtained in phase III are used to form polynomial function see equation 1 given below.

We use simple mathematical concept for constructing polynomial as shown below. Suppose from phase III we get integers 7, 2, 5 and 9 than polynomial is constructed as

$(x-7)*(x-2)*(x-5)*(x-9)*(x-41) \rightarrow$ (equation 1)

and we get polynomial of degree four.

E. Phase V (Transfer)

In this phase the polynomial is transferred to receiver machine. During transfer not only polynomial but some metadata is also transferred which help in construction of replica on destination machine. In our software we have passed *"create table"* query which creates the table in destination's database. In phase III we have created doubly linked list, the nodes of that linked list are also transmitted.

So by the end of this phase three things are transmitted to the destination system, namely polynomial, metadata and nodes of linked list.

F. Phase VI (Decrypting Polynomial)

This phase is carried out at destination system. First of all the roots of polynomial are found using any standard method like ***Newton Raphson***and than a list is constructed with zero at locations equivalent to ***index_number*** of each node of transferred linked list. Than we use transferred linked list to construct correct integers. This process is carried out in next phase.

G. Phase VII (Reverse Mapping Function)

When this phase is executed than using doubly linked list we reconstruct list of integers. The algorithm which is carried out during this phase is given below.

```
Revers_Mapping_Function(Doubly_Linked_ListDLL,Roots roots)
{Step 0: After finding roots arrange them in ascending order and then put zero at every index number present in doubly linked list.
 Step 1: Now scan DLL for node having highest recursive_step. store data of that node in structure variable temp and delete the node from DLL
 Step 2: To list of small integers formed in step 0 add temp->mini and update the small integer list obtained in step 0 accordingly.
 Step 3: Repeat step 1 and 2 until DLL is not empity.
}
```

Once above algorithm gets executed, we regain list of large integers which is analogous to list obtained after accomplishment of phase II.

H. Phase VIII (Decompression)

In this phase first of all integers obtained after phase VII are converted into their binary form. Then block of eight bits are taken and converted into decimal. This decimal number is considered as ASCII code and all data is converted in ASCII form. This data is last column of compression table shown in phase II. We use LZ77 decompression algorithm to regain binary data.

I. Phase IX (Regeneration)

When this phase is executed than binary data is converted into original data by taking block of eight bits and converting it to ASCII form. Then using metadata which is transferred to receiver’s system this data is inserted into table which is formed using metadata. Hence replica of original database is created on receiver's system.

V. CONCLUSION

Using this technique entitled “Polynomial Based Database Transfer Technique” we can easily transfer database from one machine to another in form of polynomial that too securely and in short time. We have tested this technique with small databases and it works well but with large databases this technique is to be modified a bit. In that case instead of transferring only one polynomial a matrix of polynomials needs to be transferred. Due to which we have to transfer list of metadata’s. Using which we can regenerate the desired database.

REFRENCES

[1] Siddharth Srivastava, “Virtual Technology”, 2012, Lambert Academic Publication.

[2] Siddharth Srivastava and Umesh Chandra Jaiswal; (November 2011) Virtually Modified Multi Protocol Label Switching, International Journal of Engineering Research &Indu. Appls. (IJERIA). ISSN 0974-1518, Vol.4, No. IV (November 2011), pp. 165-192

[2] Siddharth Srivastava and Umesh Chandra Jaiswal; (July 2011) Study of State Load Dispatch Center and EDI based Software Tree List, International Journal of Engineering Science and Technology (IJEST), ISSN : 0975-5462, Vol. 3 No. 7 July 2011, pages 5818-5824

[3] Anchao Cheng, Xin Yao and LinguoDuan, “Design and Implementation of Integrated Services Switch Simulation Model Based on ATM-MPLS”, vol. 2, pp. 223-226, 2008

[4] Alouneh, Sahel En-Nouaary, Abdeslam Agarwal, Anjali, “Securing MPLS network with multi path routing”, IT, 2007. ITNG ’07, pp. 809-814, April 2007.

[5] Chen, T.M. Oh and T.H., “Reliable Services in MPLS”, Communication Magazine, IEEE, vol. 37 on 12, pp. 58-62, Dec. 1999

[6] Acharya, A. Ganu, S. Misra, A., “DCMA A Label Switching MAC for Efficient packet Forwarding in Multihop Wireless Network”, Selected Areas in Communications, IEEE Journal, vol. 24 on 11, pp. 1995-2004, Nov. 2006

[7] Na Lin, Li-xue Song, Tao Yang, “An Improved Mobile Node Fast Handover Algorithm”, Computational Intelligence and Software Engg. 2009, pp. 1-4, Dec. 2009.

[8] Jamali A., Naja N., El Ouadghiri, Benaini R., “Improving QoS in MPLS module”, MMS, 2009 Mediterrannean, pp. 1-4, 2009.

[9] Romdhani L., Bonnet C., “Cross Layer QoS Routing Framework for Wireless Mess Networks”, Wireless and Mobile Communications, 2008. ICWMC 08, pp. 382-388, 2008.

[10] Zhangwei He and Yong Jiang, "Improve reliability of scalable VPN routing via Relaying", Network Infrastructure and Digital Content, 2010 2nd IEEE, pp. 1061-1066, Sep. 2010.

[11] Van den Berghe, De Turck F. and Demeester P., "SONAR: a plateform for flexible DiffServ / MPLS traffic engg.", Performance, Computing and Communication, 2004 IEEE., pp. 195-201, 2004.

[12] Polycarpou M., "Intelligent monitoring and fault tolerance in large scale distributed systems", control and fault tolerant systems (SysTol 2010 conferance). Pp. 480 onwards, October 2010.

[13] Al-Jaroodi J., Mohamed N., Hong Jiang, "Distributed System middleware architecture from a software engg. perspective", Information reuse and integration-2003 IEEE. pp. 572-579, October 2003.

[14] Tally G., Whitmore B., Sames D., Matt B., Niebuhr B., Bakken D.,"Intrusion tolerant distributed object system: project summary", DARPA Information survivability Conference and Exposition 2003. Vol. 2, pp. 149-151, 2003

[15] Herbert Schildt, "The Complete Reference Java", Seventh Edition-2007, Tata McGraw-Hill Edition.

[16] George Coulouris, Jean Dollimore and Tim Kindberg, "Distributed Systems concepts and Design" Fourth Edition- patentact 1988 (LPE) Pearson Edition.