ISSN 2278-5612

BST Algorithm for Duplicate Elimination in Data Warehouse

Payal Pahwa[1] , **Rashmi Chhabra**[2]

[1]**Bhagwan Parshuram Institute of Technology, I.P.University**
Delhi 110085, India
pahwapayal@gmail.com
[2]**GVM Institute of Technology and Management, DCRUST ,Murthal**
Sonipat, Haryana 131001, India
rashmidahra@gmail.com

ABSTRACT

Data warehousing is an emerging technology and has proved to be very important for an organization. Today every business organization needs accurate and large amount of information to make proper decisions. For taking the business decisions the data should be of good quality. To improve the data quality data cleansing is needed. Data cleansing is fundamental to warehouse data reliability, and to data warehousing success. There are various methods for data cleansing. This paper addresses issues related data cleaning. We focus on the detection of duplicate records. Also an efficient algorithm for data cleaning is proposed. A review of data cleansing methods and comparison between them is presented.

KEYWORDS: Data cleansing; data warehouse; data cleansing techniques; binary search tree.

1. INTRODUCTION

A data warehouse is a subject-oriented, integrated, time variant, non-volatile collection of data in support of management's decision-making process [1]. Many organizations today heavily rely on data for their business and decision making processes. The data collected from various sources may have data quality problems in it. These kinds of issues become prominent when various databases are integrated. The integrated databases inherit the data quality problems that were present in the source database. Data quality plays an important role in the success of the analysis. Therefore data quality is important for data warehouse. There are different criteria's on which the quality of data depend[11]. Therefore The data in the integrated systems need to be cleaned for proper decision making. Cleansing of data is one of the most crucial steps. In our paper, focus is on one of the major issue of data cleansing i.e. "duplicate record detection" which arises when the data is collected from various sources. The algorithm like standard duplicate elimination algorithm , sorted neighborhood algorithm , duplicate elimination sorted neighborhood algorithm, and priority queue algorithm already exist for data cleaning[3]. All these algorithms follow one common phase i.e. sort the file to bring all duplicate records together. This extra phase increases the time complexity of the algorithm.

In this paper we propose an alternative approach for eliminating records from a file by creating binary search tree. This algorithm reduces the execution time and complexity by excluding the extra phase i.e. sorting the file.

2. RELATED LITERATURE

Sorted neighborhood method focuses on approximate duplicates detection. This method consists of three steps: First, a key is computed by extracting the relevant fields or portion of fields from the database. The choice of the key depends on domain knowledge. Efficiency of algorithm heavily depends on the key chosen. The second step is to sort the database using the key calculated in first step[9]. Finally a fixed size window is moved to compare the records.. Every new record entering in the window is compared with previous records in the window. Suppose the size of the window is w records, then every new record entering the window is compared with the previous $w-1$ records to find "matching" records. The complexity of this algorithm is calculated at different phases of the algorithm, the key creation phase requires O(n) operations, the sorting phase requires O(n LOG n), merging requires O(wn) where N is no. of records in database. So total complexity of sorted neighborhood method is O(n log n) if $w<\log n$ otherwise O(w n)[2].
The drawback of the method is that it depends heavily on the proximity of duplicate records after sorting. If duplicate records are far apart after sorting, it is unlikely that they will appear in the same window during the scanning process, and hence, will be missed.

As discussed in token based approach instead of taking all the fields as a key we use smart tokens. Tokens are formed using the most important fields of records. This heavily depends on domain knowledge. This is an important step because performance of the whole algorithm depends on it. Then the tokens are sorted and compared to identify duplicate records. Token-based technique achieves a better result than the record-based techniques of comparable algorithms. The sorting phase requires O(n log n) operations where N is maximum number of atomic strings in either field[6].

Priority queue algorithm focuses on approximate duplicate detection. It uses a priority queue of sets of records belonging to the last few clusters detected [5]. The algorithm sequentially scans the database and find out whether each record belongs to a cluster represented in the queue or not. If the record is an existing member of cluster then next record in database is scanned otherwise the record is kept in the priority queue. This record is then again checked against existing clusters. If it is not the member of any cluster of them then it is assumed as a new cluster in priority queue. With this approximate similar records are kept in same cluster. Hence we identify the duplicate records easily. The complexity is calculated For fixed T, K, L _and M where T is number of records in database each of length L and K is maximum number of sets in the queue with maximum size of each set M , all union-find and priority queue operations take small constant amounts of time. For one pass over the database, the number of invocations of the Smith-Waterman algorithm is bounded above by TKM . The overall time complexity of one pass is therefore O(TL log TL+TKLM) =O(TL log TL) as there is small fixed number of passes, the total time complexity is the same[4].

Duplicate record elimination using external merge sort method uses modified two way merge sort which removes duplicate records along with sorting. If two input runs contain duplicate records, then the output run produced by merging them should retain only one copy of each record. Whenever two input tuples are compared and found to be identical, only one of them is written to the output run and the other is discarded by advancing the appropriate pointer to the next tuple[7]. The complexity of the duplicate elimination process using modified merge-sort is determined by two factors: the number of phases and the size of the output runs produced at each phase. The number of phases/operations required to sort a file along with duplicate elimination is $O(\log_2 N)$ where N is no. of records in file regardless of the number of duplicate tuples.

All the above discussed algorithms are used to detect duplicate records and each of them sorts the list as their initial phase. Time is wasted in this extra phase as the best time complexity of a sorting algorithm is O(n logn).We proposed an algorithm that uses binary search tree and reduces the overhead of sorting the list.

3. COMPARISON OF OUR APPROACH AND EXISTING APPROACHES

The comparison of all the above discussed techniques is given in the Table 1[10]. We compare them on the basis of what are the key steps followed by the technique, whether the technique find approximate or exact duplicate of records, and there complexities.

Table 1: Comparison of Existing Techniques

Approach	Key Steps	Duplicate Detection	Complexity
Sorted Neighborhood method	*Key creation + sorting + Merging using sliding window*	*Approximate duplicate detection*	*O(nw) where ' n' is no. of records and 'w' is window size.*
Token Based Algorithm	*Selection of fields + formation of tokens + sorting + duplicate detection and elimination*	*Exact duplicate detection*	*O(n log n) Where 'n' is maximum number of atomic strings in either field.*
Priority Queue Algorithm	*Sorting + merging +duplicate elimination using priority queue*	*Approximate duplicate detection*	*O(tl log tl)* *Where 't' is no. of records with Length 'l'*
Proposed algorithm	*Selection of fields + formation of tokens +BST creation*	*Approximate duplicate detection*	*O(log n) where n is total no. of records*

4. BINARY SEARCH TREE

A binary tree T is a binary search tree if each node N of T has the following property: The value of N is greater than every value in the left subtree of N and is less than every value in the right subtree of N. Left and right subtree are again binary search tree. The main property of binary search tree is that all the nodes have distinct values. To insert an element in a BST first of all we find the location of the item if item is not in the tree then it is inserted at that location. Suppose there are n data items to be inserted in BST then the average depth of tree is n! that is approximately c $\log_2 n$, where c = 1.4. Accordingly the average running time f(n) to search an element in BST with n elements is proportional to $\log_2 n$ that is f(n) = O($\log_2 n$). We will use this concept in our algorithm to clean data. We propose a framework for the same as discussed below.

5. PROPOSED FRAMEWORK

The framework of our proposed algorithm is

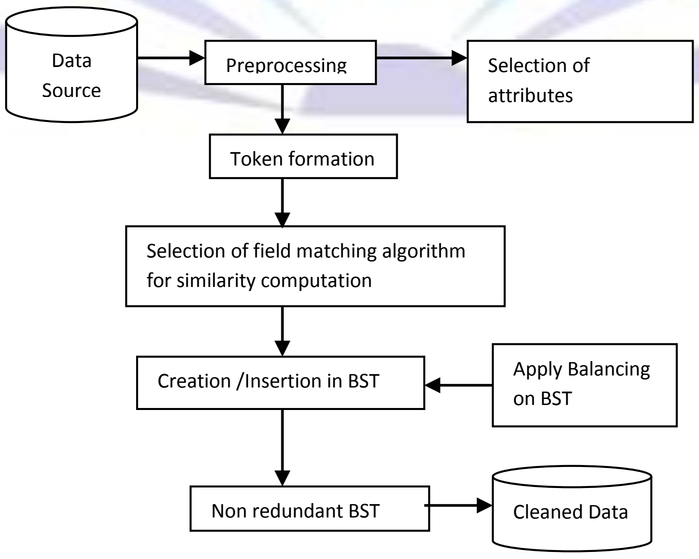


Figure 1: Proposed framework

5.1 Selection of attributes

Data warehouse can have hundreds of attributes. All of them cannot be used for records matching. Therefore attribute selection is very important when two records are compared. This is the foundation step for all the other steps. Those attributes which are useful for analysis are selected. This requires the domain knowledge. Therefore the key chosen is domain dependent.

5.2 Token formation

This step makes use of the selected attribute field values to form a token. The tokens can be created for a single attribute field value or for combined attributes. For example, contact name address and telephone number attributes are selected to create a combine token for further cleaning process. Tokens are formed using numeric values, alphanumeric values and alphabetic values. Suffix of the attributes are removed like Mr., Dr. and so on. Numeric tokens comprise only digits [0 – 9]. Alphabetic tokens consist of alphabets (aA - zZ).

5.3 Selection of field matching algorithm for similarity computation

After token formation the next thing is to select the field matching algorithm.These methods consider records as strings and compute record similarity using string comparison algorithms. Basically three field matching algorithms exist for identification of duplicates[8] : Basic Field matching algorithm, Recursive field matching and Smith-Waterman algorithm. In basic field matching algorithm, the degree to which two fields match is the no. of their matching atomic strings divided by their average no. of atomic strings. In recursive field matching algorithm ,two strings match with degree 1.0 if they are the same atomic string or one abbreviates the other else their degree of match is 0.0 .The Smith Waterman algorithm computes an n x m matrix with given two strings of length n and m whose [i, j] component is the edit-distance between the prefix 1..i of one string and the prefix 1.. j of the second string. When the prefixes (or the entire strings) match exactly, then the optimal alignment follows the main diagonal. For approximate matches, the optimal alignment is within a small distance of the diagonal.

5.4 Creation of BST/ duplicate elimination

In this step the tokens are used to create a binary search tree. The first token is taken as the root of the tree. Whenever the next record is added to the tree it is compared with the root of the tree. If they match then that record is not inserted other wise the record is inserted according to the value of token whether it is smaller or greater than the token value of the root node. The above process is repeated with all the tokens and moved to left or right subtree for further matching. The record is inserted only in the case when there is no match. Therefore all the matched records are eliminated.

6. PROPOSED ALGORITHM

```
BSTdeduplication( token p, BST T) //insert a token p in BST
{
      if   T=NUL
              Root(T) ← p //p becomes the root of the tree
          else if (p<t->element)
          BSTdeduplication(p,t->left);
          else if (t->element<p)
              BSTdeduplication(p,t->right);
          else ; // the token is duplicate token
}
```

7. COMPLEXITY OF PROPOSED ALGORITHM

Suppose there are n data items to be inserted in BST then the average depth of tree is n! that is approximately c $\log_2 n$, where c = 1.4. Accordingly the average running time f(n) to search an element in BST with n elements is proportional to $\log_2 n$ that is $f(n) = O(\log_2 n)$. The complexity of the algorithm depends on the height of the tree. Cost of each operation described above is O(h). Where h is the height of the binary search tree. Thus we see that the complexity of this algorithm is much less than the existing algorithms. We have also implemented the discussed approach.

8. IMPLEMENTATION DETAILS

We have developed a prototype for De Duplication of database records extracted from multiple operational systems. It is called as BSTdeduplication It has three components, each executed at different level:

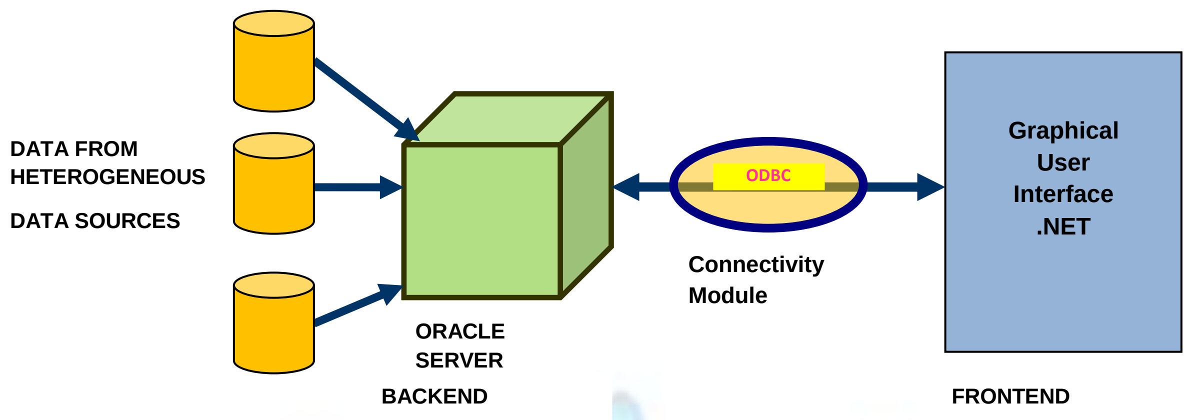


Figure2 : BSTdeduplication components

FRONT-END is the interface/server between the user and the database or data warehouse. This is the highest level of our framework which hides out all the background details from the user. Here, we use .NET at the front end. It interacts with the user with the help of GUI features and accesses the records in the warehouse.

BACK-END is the lowest level in our framework that deals with the databases present in the data warehouse. ORACLE 10g was used at the back-end server for creating the database records or managing the data warehouse. It performs all the querying, analysis and house-keeping tasks necessary for backup and maintenance of the warehouse.

MIDDLEWARE between the two levels of our framework is ODBC connectivity. It is necessary to bridge the database records with the algorithms as both are implemented in different applications. For our design this middleware is of prime importance because it causes smooth transition between the above two levels.

When the user tries to enter any new record in the database, the algorithms creates its token and insert it into the binary search tree . If token is already found in the binary search tree then, the record is discarded and a message is displayed on the screen informing the user about the redundancy that exist. Otherwise, the record is successfully added in the data warehouse.

Some of the screenshots of the BST deduplication prototype are shown below:

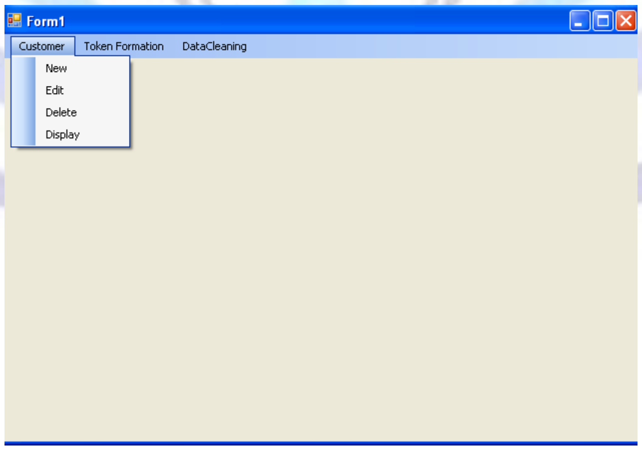


Figure3: Initial screen

The database relations are implemented as menus and the operations that can be performed on those databases are shown in the drop-down window. User can select the desired relation and perform the desired operation by selecting from the drop-down menu.

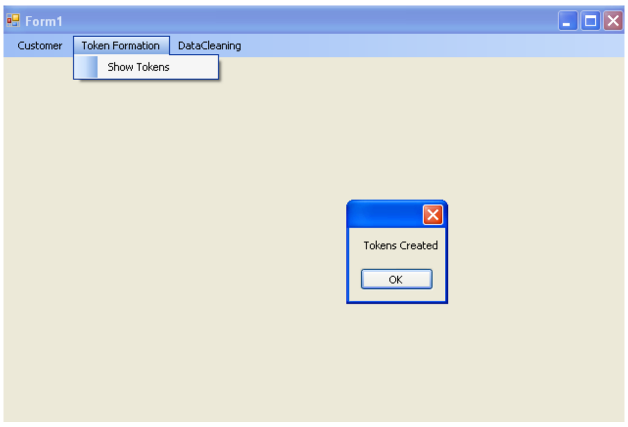
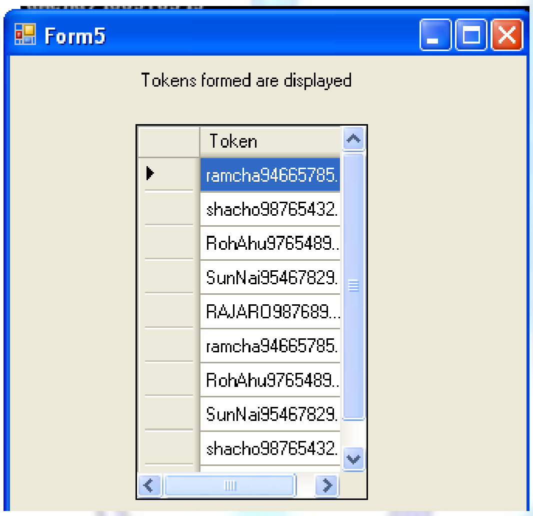


Figure4: Token Created

Whenever user enters a new record its token is created. If we have existing database then to form the tokens we can select token formation button to create tokens. T he tokens can be displayed



Tokens formed are displayed

	Token
▶	ramcha94665785.
	shacho98765432.
	RohAhu9765489..
	SunNai95467829.
	RAJAR0987689...
	ramcha94665785.
	RohAhu9765489..
	SunNai95467829.
	shacho98765432.

Figure5: Token Shown

Then this token is inserted into the binary search tree. If there is duplication then a proper message is displayed that token already present otherwise it is simply added into the binary search tree.

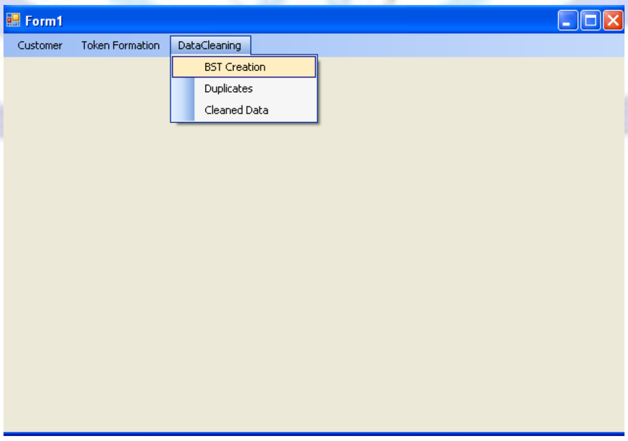
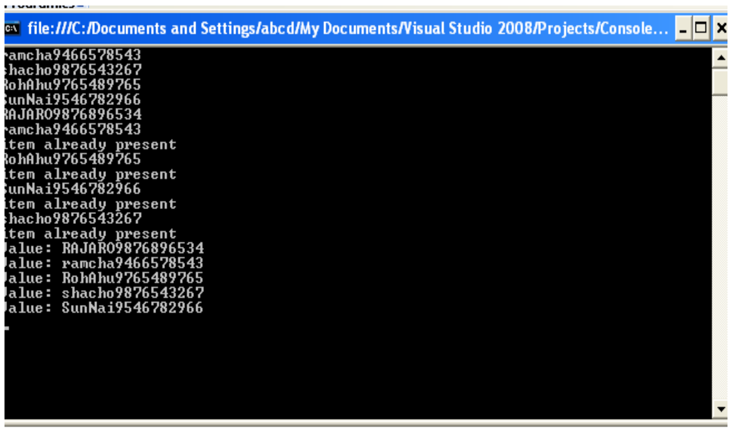


Figure6: BST Creation

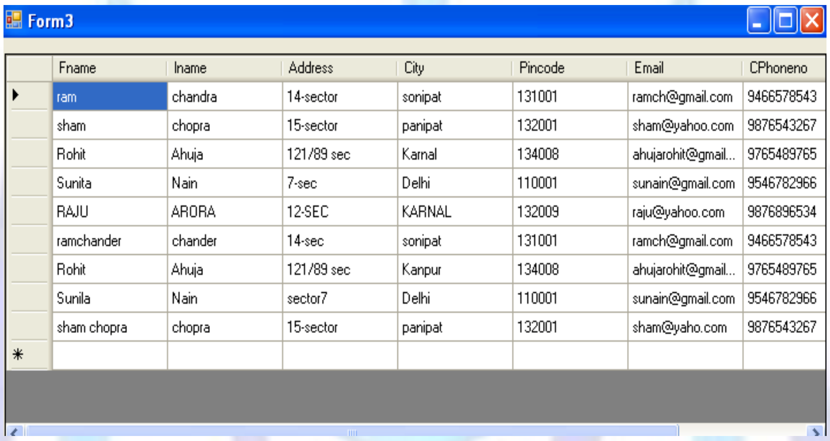


file:///C:/Documents and Settings/abcd/My Documents/Visual Studio 2008/Projects/Console...

```
amcha9466578543
hacho9876543267
ohAhu9765489765
unNai9546782966
AJARO9876896534
amcha9466578543
item already present
ohAhu9765489765
item already present
unNai9546782966
item already present
hacho9876543267
item already present
alue: RAJARO9876896534
alue: ramcha9466578543
alue: RohAhu9765489765
alue: shacho9876543267
alue: SunNai9546782966
```

Figure7: Duplicate record

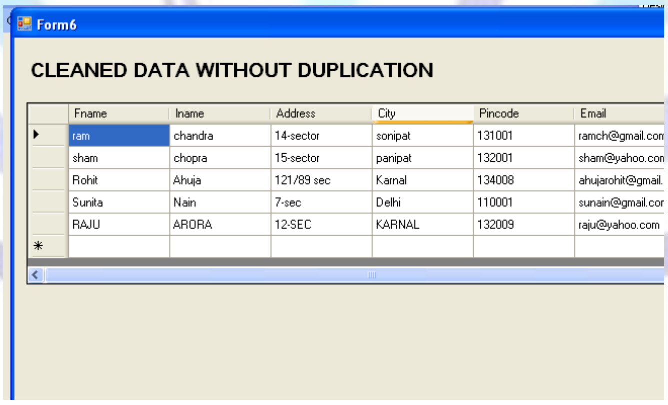
The Figure8 shows the existing records that have duplicate entries before applying the deduplication after applying the deduplication we get cleaned data shown in figure 9.



Form3

Fname	lname	Address	City	Pincode	Email	CPhoneno
ram	chandra	14-sector	sonipat	131001	ramch@gmail.com	9466578543
sham	chopra	15-sector	panipat	132001	sham@yahoo.com	9876543267
Rohit	Ahuja	121/89 sec	Karnal	134008	ahujarohit@gmail...	9765489765
Sunita	Nain	7-sec	Delhi	110001	sunain@gmail.com	9546782966
RAJU	ARORA	12-SEC	KARNAL	132009	raju@yahoo.com	9876896534
ramchander	chander	14-sec	sonipat	131001	ramch@gmail.com	9466578543
Rohit	Ahuja	121/89 sec	Kanpur	134008	ahujarohit@gmail...	9765489765
Sunila	Nain	sector7	Delhi	110001	sunain@gmail.com	9546782966
sham chopra	chopra	15-sector	panipat	132001	sham@yaho.com	9876543267

Figure8: Showing records with duplication



Form6

CLEANED DATA WITHOUT DUPLICATION

Fname	lname	Address	City	Pincode	Email
ram	chandra	14-sector	sonipat	131001	ramch@gmail.co
sham	chopra	15-sector	panipat	132001	sham@yahoo.co
Rohit	Ahuja	121/89 sec	Karnal	134008	ahujarohit@gmail.
Sunita	Nain	7-sec	Delhi	110001	sunain@gmail.co
RAJU	ARORA	12-SEC	KARNAL	132009	raju@yahoo.com

Figure9: Cleaned data

Hence, our design approach has taken care of both the aspects that duplicate elimination in existing database is removed and as well as new data that is inserted in the data warehouse is not redundant . Thus we get a clean data warehouse .

9. CONCLUSION

In this paper we have tried to highlight the major problems encountered when data sets are integrated from multiple sources. Also we have discussed some of the existing algorithms that have been designed to perform identification of exact as well as approximate duplicate records before loading them in the data warehouse. But, we have analyzed that they deal with only a limited portion of the problem and do not yield the desired results. In the existing techniques the token or records are first of all sorted and then detection /elimination process goes on. The complexity of sorting the records or tokens is O(n log n) if best sorting technique is used. In the proposed approach there is no need to sort the tokens therefore the complexity reduces. The algorithm removes the duplicate record when it is added to the binary search tree. We have implemented these operations using Oracle server at the back-end and .NET at the front-end. The algorithm is also implemented in c# language.

10. ACKNOWLEDGEMENTS

I want to express my deepest gratitude for Dr. Payal Pahwa and Ms. Gurpreet Bansal for their major help and support through all the phases of research and development.

REFERENCES

[1] W.H.Inmon , Building the Data Warehouse, John Wiley and Sons, Second Edition 1996.

[2] Hernández M.A., Stolfo S.J. “The merge/purge problem for large databases”. Proceedings of the International Conference on Management of Data 1995 (ACM SIGMOD- 1995), San Jose, California, p. 127–138.

[3] J. Jebamalar Tamilselvi and V. Saravanan “A Unified Framework and Sequential Data Cleaning Approach for a Data Warehouse”, IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.5, May 2008

[4] Alvaro E. Monge and Charles P. Elkan “Efficient domain-independent detection of approximately duplicate database records” Department of Computer Science and Engineering 92093–0114 February 20,1997.

[5] Monge A.E., Elkan C.P. “An efficient domain-independent algorithm for detecting approximately duplicate database records” : Proceedings ACM-SIGMOD-97, 1997.

[6] C.I. Ezeife and Timothy E. Ohanekwu, Use of Smart Tokens in Cleaning Integrated Warehouse Data, the International Journal of Data Warehousing and Mining (IJDW), Vol. 1, No. 2, pp. 1-22, Ideas Group Publishers, April-June 2005.

[7] Bitton D.and Dewitt D.J. “ Duplicate Record Elimination in Large Data Files” ACM Transactions on Database Systems, June 1983, Vol. 8, no. 2, p 255 – 265.

[8] Monge A.E and Elkan C.P. “The Field Matching Problems: Algorithms and applications.” Proceedings of the 2nd Int’l Conference on Knowledge and Data Mining , 1996, p 267 – 270

[9] T.E. Ohanekwu, C.I. Ezeife, A token-based data cleaning technique for data warehouse systems, IEEE Workshop on Data Quality in Cooperative Information Systems, Siena,Italy, January 2003

[10] Rashmi Chhabra and Payal Pahwa, Domain Dependent and Independent Data Cleansing Technique, International Journal of Computer Science & Technology, Vol 2, September 2011

[11] Rashmi Chhabra , Payal Pahwa “Data Quality and Cleansing Tools: A Comparitive Study”, International Journal of advanced engineering and applications, volume 1,june 2010. Page No 76-79

Authors’ biography with Photo

Ms. Payal Pahwa received her M.S. degree from BITS Pilani in 1997 and the Ph.D. degree in Datawarehousing from J.N.U. Delhi. Presently she is working as a Professor in Bhagwan Parshuram Institute of Technology, New Delhi.

Ms. Rashmi Chhabra received her MCA degree from Kurukshetra University in 1998, the M.Tech degree in 2005 from IASE Rajasthan, M.Phil degree in 2007 from Madurai Kamraj University and presently pursing Ph.d. From NIMS University, Rajasthan. She is working as an Asstt. Professor in GVM Institute of Technology and Management, Sonepat, Haryana