International Journal of Engineering In Advanced Research Science and Technology(IJEARST)
ISSN 2278-2566 Volume 2 –JULY 2016.

Chunk -Stage File Replication Scheme: Detection OF Replica Data In Scattered-Distributed method

Guntur.Susmitha[#1] and Subhani.Shaik[*2]

[#]*M.Tech, Department CSE, St. Mary's Women's Engineering College,Budampadu, Guntur, India*

[*] *Assistant Professor,M.Tech, Department CSE, St. Mary's Women's Engineering College,Budampadu, Guntur,India*

***Abstract*- Nowadays, a large of amount of data is being generated. The dataset contain a large amount of data where there might be possibility of storing duplicate records. The formation of duplicate records is possible only if the data are arranged in heterogeneity mannerism. The discovery of duplicate records from dataset is higher time consuming process. It is an important process in data cleaning and data integration techniques. In some scenario, a web page may display according to the query with its relevant advertisements. The data schema comprised on the association between any real entities. To overcome from this issue, we propose a novel block level deduplication system, which disposes the redundant records from database. It efficiently handles the search queries. The main contribution is the removal of redundant records from the level of parent. By doing so, the effectiveness of the system is enhanced.**

Keywords: ***Data cleaning, Data Integration process, Duplicate records, block-level system and effectiveness.***

I. INTRODUCTION

Data are among the most important assets of a company. But due to data changes and sloppy data entry, errors such as duplicate entries might occur, making data cleansing and in particular duplicate detection indispensable. However, the pure size of today's datasets renders duplicate detection processes expensive. Online retailers, for example, offer huge catalogs comprising a constantly growing set of items from many different suppliers. As independent persons change the product portfolio, duplicates arise. Although there is an obvious need for de duplication, online shops without downtime cannot afford traditional de duplication. Progressive duplicate detection identifies most duplicate pairs early in the detection process. Instead of reducing the overall time needed to finish the entire process, progressive approaches try to reduce the average time after which a duplicate is found. Early termination, in particular, then yields more completes results on a progressive algorithm than on any traditional approach.

Databases play an important role in today's IT based economy. Many industries and systems depend on the accuracy of databases to carry out operations. Therefore, the quality of the information stored in the databases, can have significant cost implications to a system that relies on information to function and conduct business. In an error-free system with perfectly clean data, the construction of a comprehensive view of the data consists of linking -- in relational terms, joining-- two or more tables on their key fields. Unfortunately, data often lack a unique, global identifier that would permit such an operation. Furthermore, the data are neither carefully controlled for quality nor defined in a consistent way across different data sources. Thus, data quality is often compromised by many factors, including data entry errors (e.g.,studet instead of student), missing integrity constraints (e.g., allowing entries such as Employee Age=567), and multiple conventions for recording information To make things worse, in independently managed databases not only the values, but the structure, semantics and underlying assumptions about the data may differ as well. Progressive duplicate detection identifies most duplicate pairs early in the detection process. Instead of reducing the overall time needed to finish the entire process, progressive approaches try to reduce the average time after which a duplicate is found. Progressive techniques make this trade-off more beneficial as they deliver more complete results in shorter amounts of time. Progressive Sorted Neighborhood method take clean dataset and find some duplicate records and Progressive Blocking take dirty datasets and detect large duplicate records in databases.

II. RELATED WORK

Steven Euijong Whang, David Marmaros,[1] Entity resolution (ER) is the problem of identifying which records in a database refer to the same entity. In practice, many applications need to resolve large data sets efficiently, but do not require the ER result to be exact. For example, people data from the Web may simply be too large to completely resolve with a reasonable amount of work. As another example, real-time applications may not be able to tolerate any ER processing that takes longer than a certain amount of time. This paper investigates how we can maximize the progress of ER with a limited amount of work using "hints," which give information on records that are likely to refer to the same real-world entity. A hint can be represented in various formats, and ER can use this information as a guideline for which records to compare first. We introduce a family of techniques for constructing hints efficiently and techniques for using the hints to maximize the number of matching records identified using a limited amount of work. Using real data sets, we illustrate the potential gains of our pay-as-you-go approach compared to running ER without using hints. An ER process is often extremely expensive due to very large data sets and compute-intensive record comparisons.

The proposed a pay-as-you-go approach for Entity Resolution (ER) where given a limit in resources (e.g., work, runtime) we attempt to make the maximum progress possible. In [11] the World Wide Web is witnessing an increase in the amount of structured content – vast heterogeneous collections of structured data are on the rise due to the Deep Web, annotation schemes like Flickr, and sites like Google Base. While this phenomenon is creating an opportunity for structured data management, dealing with heterogeneity on the web-scale presents many new challenges. In this paper, we highlight these challenges in two scenarios – the Deep Web and Google Base. We contend that traditional data integration techniques are no longer valid in the face of such heterogeneity and scale. We propose new data integration architecture, PAYGO, which is inspired by the concept of data spaces and emphasizes pay-as-you-go data management as means for achieving web-scale data integration.

[13] Similarity join is a useful primitive operation underlying many applications, such as near duplicate Web page detection, data integration, and pattern recognition. Traditional similarity joins require a user to specify a similarity threshold. In this paper, we study a variant of the similarity join, termed top-k set similarity join. It returns the top-k pairs of records ranked by their similarities, thus eliminating the guess work users have to perform when the similarity threshold is unknown. An algorithm, topk-join, is proposed to answer top-k similarity join efficiently. It is based on the prefix filtering principle and employs tight upper bounding of similarity values of unseen pairs. Experimental results demonstrate the efficiency of the proposed algorithm on large-scale real datasets. Given a similarity function, a similarity join between two sets of records returns pairs of records from two sets such that their similarities are no less than a given threshold. In this paper, we study the problem of answering similarity join queries to retrieve top-k pairs of records ranked by their similarities. Existing approaches for the traditional similarity joins with a given threshold will have to make guesses on the similarity threshold and incur much redundant calculation. We propose an efficient algorithm that computes the answers in a progressive manner.

[2] Ahmed K. Elmagarmid, Panagiotis G.Ipeirotis, Vassilios S.Verykios often, in the real world, entities have two or more representations in databases. Duplicate records do not share a common key and/or they contain errors that make duplicate matching a difficult task. Errors are introduced as the result of transcription errors, incomplete information, lack of standard formats, or any combination of these factors. In this paper, we present a thorough analysis of the literature on duplicate record detection. We cover similarity metrics that are commonly used to detect similar field entries, and we present an extensive set of duplicate detection algorithms that can detect approximately duplicate records in a database. We also cover multiple techniques for improving the efficiency and scalability of approximate duplicate detection algorithms. We conclude with coverage of existing tools and with a brief discussion of the big open problems in the area. The problem that we study has been known for more than five decades as the record linkage or the record matching problem in the statistics community. The goal of record matching is to identify records in the same or different databases that refer to the same real-world entity, even if the records are not identical.

In paper [9] Duplicate detection is the task of identifying all groups of records within a data set that represent the same real-world entity, respectively. This task is difficult, because (i) representations might differ slightly, so some similarity measure must be defined to compare pairs

of records and (ii) data sets might have a high volume making a pair-wise comparison of all records infeasible. To tackle the second problem, many algorithms have been suggested that partition the data set and compare all record pairs only within each partition. One well-known such approach is the Sorted Neighborhood Method (SNM), which sorts the data according to some key and then advances a window over the data comparing only records that appear within the same window.

III. INTENSIFIED AUTHENTIC PROGRESSIVE DISTRIBUTED DEDUPLICATION SYSTEMS

In an error-free system with perfectly clean data, the construction of a comprehensive view of the data consists of linking—in relational terms, joining—two or more tables on their key fields. Unfortunately, data often lack a unique, global identifier that would permit such an operation. Furthermore, the data are neither carefully controlled for quality nor defined in a consistent way across different data sources. Thus, data quality is often compromised by many factors. In this proposed system two novel, progressive duplicate detection algorithms namely progressive sorted neighborhood method (PSNM), which performs best on small and almost clean datasets, and progressive blocking (PB), which performs best on large and very dirty datasets. Both enhance the efficiency of duplicate detection even on very large datasets. In this project genetic programming algorithm is used to detect the duplication of text document. Text document which has same content with different name is detected and saved as duplicate. If any document has same name with different content is saved without overwrite.

The main objectives of the progressive deduplication system are listed as:

- To detect the duplicate documents.
- To heightens the approach for discovering duplicate documents.
- To maximize the effect of determining the duplicate data in a stipulated time.
- To develop the support for detecting file-level and block-level deduplication system.
- To prove that our system improve the duplicate detection in stipulated time.

The algorithm takes five input parameters: D is a reference to the data, which has not been loaded from disk yet. The sorting key K defines the attribute or attributes combination that should be used in the sorting step. W specifies the maximum window size, which corresponds to the window size of the traditional sorted neighborhood method. When using early termination, this parameter can be set to an optimistically high default value. Let us consider, it has the default value 1. The last parameter N specifies the number of records in the dataset. This number can be gleaned in the sorting step, but listed as a parameter for presentation purposes.

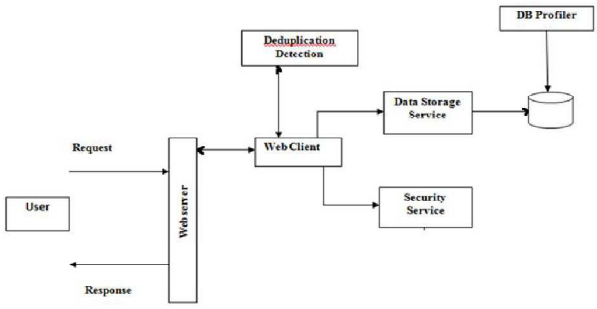

Fig.1. Proposed Workflow of distributed deduplication system

A Block -Level Distributed deduplication algorithm is used in this paper to detect the duplicates of progressive data in a distributed approach. The pseudo code is given below:

Initialize the User U_j id: IDU_j j=1….N (No.of users).

File identifier: F_{idj} j= 1….N (No. of files).

Blocks of files: B_i i=1… N (No. of blocks)

For each block B_i

For each file F_{idj}

Compute hash value of the file F_{id1}

Stores the F_{id1} as H (User Id | File Name) to the multiple servers

Obtain the unique Block identifier B_{id} for each file in the server.

End for

End for

User $U_{j\,new}$ is found,

Compute the F_{idj}, B_{idj} for the Server j

If (present value F_{idj}, B_{idj} ≠ old value F_{idj}, B_{idj})

The file is stored to the server

Else

Detects the duplicated blocks of a file.

IV. EXPERIMENTAL RESULTS

We consider restaurant information from an internal operational data warehouse and introduce errors. Because we start from real data all characteristics of real data: variations in the lengths of strings, numbers of tokens in and frequencies of attribute values, co-occurrence patterns, etc. are preserved. Since, we know the duplicate tuples and their correct counterparts in the erroneous dataset; we can evaluate duplicate elimination algorithms.

Fig.2. Start page of progressive deduplication process

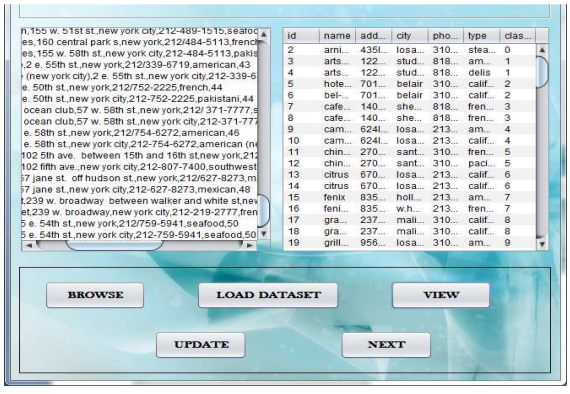

Fig.3. Loading of the dataset

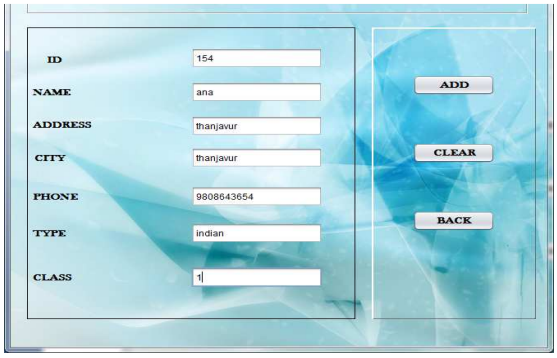

Fig.4. Successful Insertion of new records

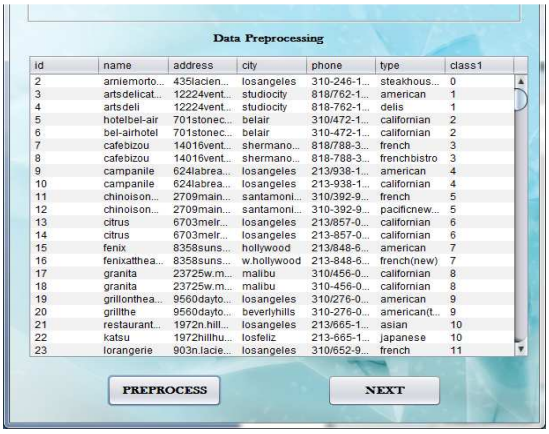

Fig.5. Data preprocessing with the updated data records

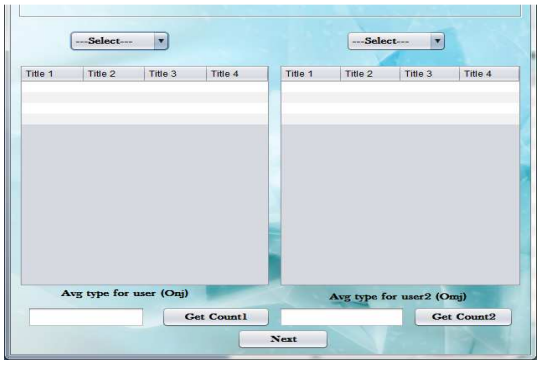

Fig.6. Process of selection technique

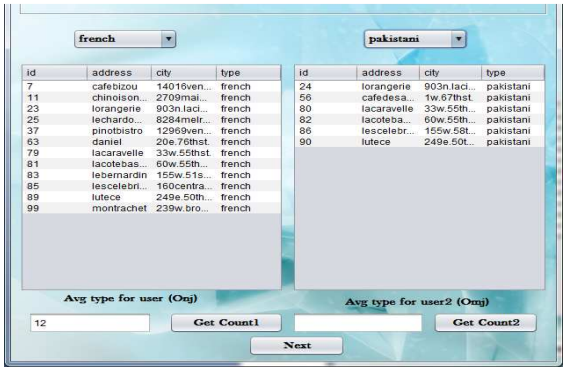

Fig.7. Categorizing the data based on its type attributes

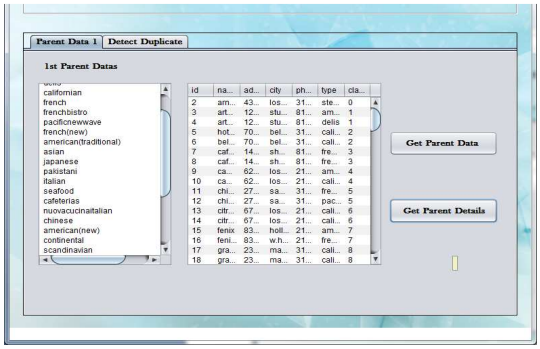

Fig.10. Based on its type categorization, parent data is assigned and viewed.

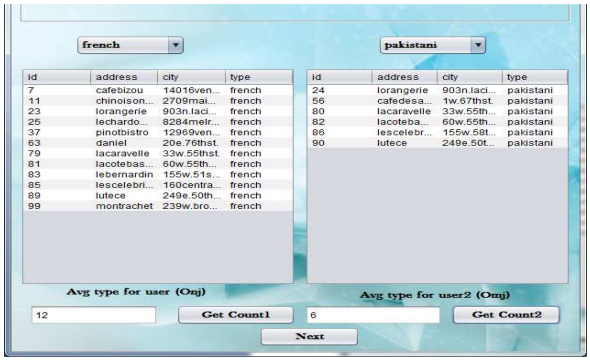

Fig. 8. Calculating the data counts.

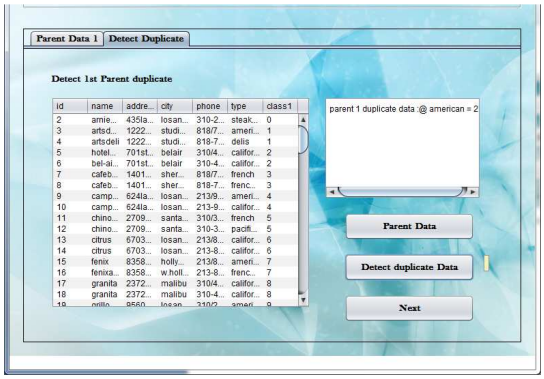

Fig.11. Finding the duplicate data from the parent node 1

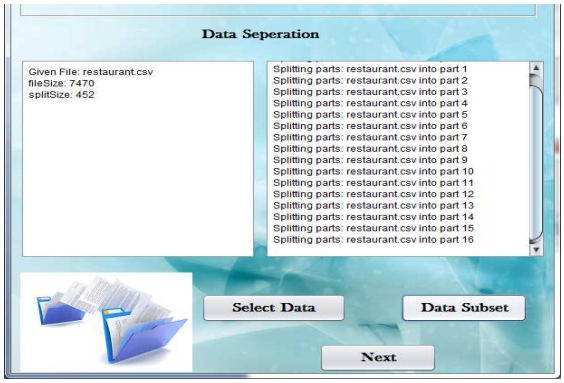

Fig. 9. Splitting the data into several parts with restricted file size.

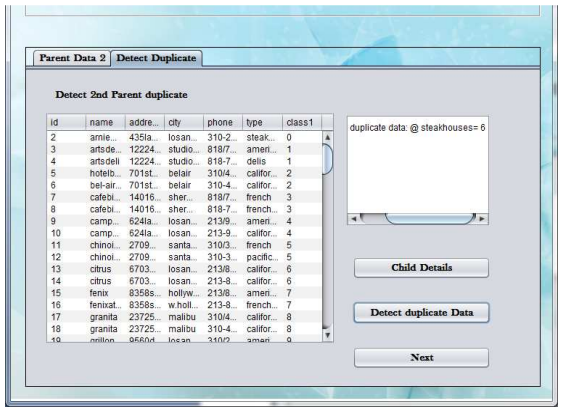

Fig.12. Finding the duplicate data from the parent node 2.

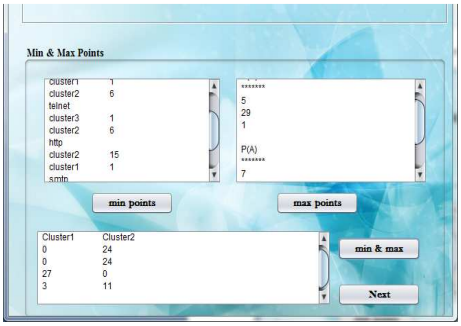

Fig.13. Calculating min and max points for every clusters

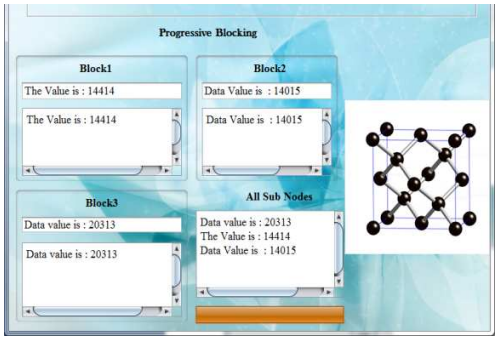

Fig.14. Progressive Block-level deduplication system

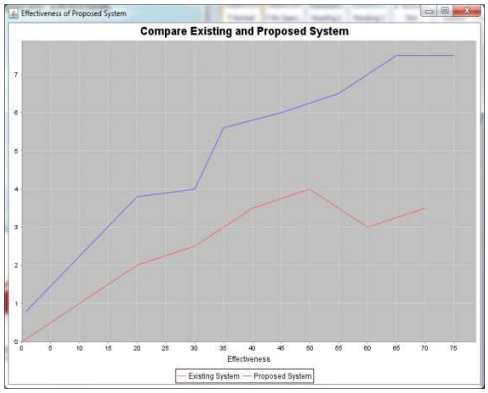

Fig.15. Based on the effectiveness metric for our proposed work with existing work.

V. CONCLUSION

This project deals about the data duplication in the large scale databases. Owing to data modification and mistyping of the data may lead to error or duplicate records. In this paper, a novel block-level file deduplication system presents the removal of duplicate records in progressive manner. In order to develop high quality data, the data are analyzed in the progressive tree structure. Our approach is able to automatically suggest duplication functions based on evidence present in the data repositories. The suggested functions properly combine the best evidence available in order to identify whether two or more distinct record entries are replicas (i.e., represent the same real-world entity) or not. It is very much useful for the best set of all users.

REFERENCES

[1] S. E. Whang, D. Marmaros, and H. Garcia-Molina, ―Pay-as-you-go entity resolution, IEEE Trans. Knowl. Data Eng., vol. 25, no. 5, pp. 1111– 1124, May 2012

[2] F. Naumann and M. Herschel, An Introduction to Duplicate Detection. San Rafael, CA, USA: Morgan & Claypool, 2010.

[3] H. B. Newcombe and J. M. Kennedy, ―Record linkage: Making maximum use of the discriminating power of identifying information,‖ Commun. ACM, vol. 5, no. 11, pp. 563–566, 1962.

[4] M. A. Hernandez and S. J. Stolfo, ―Real-world data is dirty: Data cleansing and the merge/purge problem,‖ Data Mining Knowl. Discovery, vol. 2, no. 1, pp. 9–37, 1998.

[5] X. Dong, A. Halevy, and J. Madhavan, ―Reference reconciliation in complex information spaces,‖ in Proc. Int. Conf. Manage. Data, 2005, pp. 85–96.

[6] O. Hassanzadeh, F. Chiang, H. C. Lee, and R. J. Miller, ―Framework for evaluating clustering algorithms in duplicate detection,‖ Proc. Very Large Databases Endowment, vol. 2, pp. 1282– 1293, 2009.

[7] O. Hassanzadeh and R. J. Miller, ―Creating probabilistic databases from duplicated data,‖ VLDB J., vol. 18, no. 5, pp. 1141–1166, 2009.

[8] U. Draisbach, F. Naumann, S. Szott, and O. Wonneberg, ―Adaptive windows for duplicate detection,‖ in Proc. IEEE 28th Int. Conf. Data Eng., 2012, pp. 1073–1083.

[9] Shen H, Zhang Y, Improved approximate detection of duplicates for data streams over sliding windows, Journal of Computer Science and Technology, Volume 23(6), 2008, pp. 973-987.

[10] Thorsten Papenbrock, Arvid Heise, and Felix Naumann, "Progressive Duplicate Detection", IEEE Transactions on Knowledge and Data Engineering, Vol. 27, No. 5, May 2015.