Performance Analysis of Duplicate Record Detection Techniques

Syed Hasan Adil
Department of Computer Science,
Iqra University,
Karachi, Pakistan
hasan.adil@iqra.edu.pk

Mansoor Ebrahim
Department of Computer Science,
Iqra University,
Karachi, Pakistan
mebrahim@iqra.edu.pk

Syed Saad Azhar Ali
Department of Electrical and Electronic Engineering,
Universiti Teknologi PETRONAS,
Seri Iskandar, Malaysia
saad.azhar@utp.edu.my

Kamran Raza
Department of Computer Science,
Iqra University,
Karachi, Pakistan
kraza@iqra.edu.pk

***Abstract*—In this paper, a comprehensive performance analysis of duplicate data detection techniques for relational databases has been performed. The research focuses on traditional SQL based and modern bloom filter techniques to find and eliminate records which already exist in the database while performing bulk insertion operation (i.e. bulk insertion involved in the loading phase of the Extract, Transform, and Load (ETL) process and data synchronization in multisite database synchronization). The comprehensive performance analysis was performed on several data sizes using SQL, bloom filter, and parallel bloom filter. The results show that the parallel bloom filter is highly suitable for duplicate detection in the database.**

Keywords-duplicate detection; bloom filter; SQL; database

I. INTRODUCTION

Duplicate record detection [1, 2] is a process of identifying pairs of records that belong to the same entity in one or more databases. Despite the development of many indexing techniques like ISAM, B-Tree, Bitmap, and Hash indexing, still the process of matching two records that belong to the same entity requires time which is proportional to the number of existing records. Therefore, an alternative technique is required to perform duplicate record detection. Duplicate data detection has very important applications in many critical areas including databases, distributed databases, and data warehouses. Data synchronization is a task demanded in a centralized database in case of standby after a database failure, or in a distributed database when we have to synchronize multiple remotely distributed database instances, or even in the Load part of the Extraction, Transform, and Load (ETL) process where new data have to be loaded into the database in a continuous process. Data streams like video, audio, etc. are some of the sources of big data which we want to process in real-time. In-stream processing, duplicate data detection is one of the most important tasks but at the same time it is very challenging due to the amount of data that continuously arrive at high speed. We can deal with these challenging requirements through a more robust technique like bloom filters which have the potential to perform better than the traditional duplicate detection techniques used in relational databases. Therefore, in this paper, we will deeply investigate the application of bloom filters in order to identify duplicate records in databases, distributed databases, and data warehouses. The main objective of the paper is to implement in SQL, bloom filter, and parallel bloom filter duplication detection techniques and to decide which one is the most appropriate for duplication detection.

II. RELATED WORK

Bloom filter [3] is a probabilistic data structure developed in 1970. Bloom filters are primarily based on hash functions. Bloom filters are a space-efficient data structure based on the computation of several hash functions. A Bloom filter has zero probability of false negative, but it can have more than zero probability of false positive (though it is possible to minimize the false-positive probability to zero depending on parameter selection). False-positive means that the filter may identify a new entry as already existing, even though this is not true. In addition to highly space-efficient, operations like Insert and Search are very fast in bloom filters. Deletion is generally not allowed in the bloom filters due to the additional required amount of work. While deciding about the bloom filter, one must consider a tradeoff between the space and false positive. So, if space is more important, then the bloom filter is an ideal choice (with a very little chance of false-positive). However, if even a little chance of false positives cannot be tolerated, then, in that case, one cannot use the bloom filter. Many different variants of the original bloom filters have been proposed which include but are not limited to counting bloom filter [4], d-Left counting bloom filter [5], compressed bloom filter [6], bloomier filter [7], space-code bloom filter [8], dynamic bloom

Corresponding author: Syed Hasan Adil

filter [9], etc. The applications of bloom filters [10, 11] include but are not limited to spell checking, collaboration in P2P networks, resource and packet routing, cache optimization, URL shortening, video recommendation, string matching, spam filtering, DoS and DDoS detection, anomaly detection, etc. In this research, we applied SQL, bloom filter, and parallel bloom filter to perform duplicate detection while performing bulk insertion operation in database, distributed database, and data warehouse using different numbers of tuples (i.e. from thousand to one million tuples in the tables as well as for the bulk operation) in the table.

III. PROPOSED METHODOLOGY

The discussion in the previous section acknowledged the importance of duplication detection in databases, distributed databases, and data warehouses while importing bulk data. Duplication detection in large databases is a very computational hungry task because each inserting record needs to be compared with all the exiting records in the database. It is important to note that we cannot perform a comparison based on primary keys because data are coming from various sources. In this research work, we have implemented three different approaches (i.e. SQL, bloom filter, and parallel bloom filter) to compare their performance on duplicate detection using different number of records (i.e. existing records in the table/new records to insert in the table ratio equal to 1000/1000, 10000/10000, 100000/100000, and 1000000/1000000). The overall process flow of each approach is described in Figure 3 for SQL based approach, Figure 4 for bloom filter, and Figure 5 for the parallel bloom filter. The Table used to perform duplicate detection is shown in Figure 1, while the script used to generate data is shown in Figure 2. The different steps of each approach are described below:

```
CREATE Table Students
(
 Stud_id int identity primary key,
 Stud_name nvarchar(25),
 Stud_address nvarchar(100),
 Stud_country nvarchar(25)
)
```

Fig. 1. Schema of the table used for analysis

```
Declare @Id int
Declare @TotalRecords int
Set @Id = 1
Set @TotalRecords = 1000

While @Id <= @TotalRecords
Begin

insert into Students values
('Student - ' + CAST(floor(RAND() * 100) as nvarchar(25)),
'Address - ' + CAST(floor(RAND() * 100) as nvarchar(100)),
'Country - ' + CAST(floor(RAND() * 100) as nvarchar(25)))
        Set @Id = @Id + 1

End
```

Fig. 2. Script used to generate random data

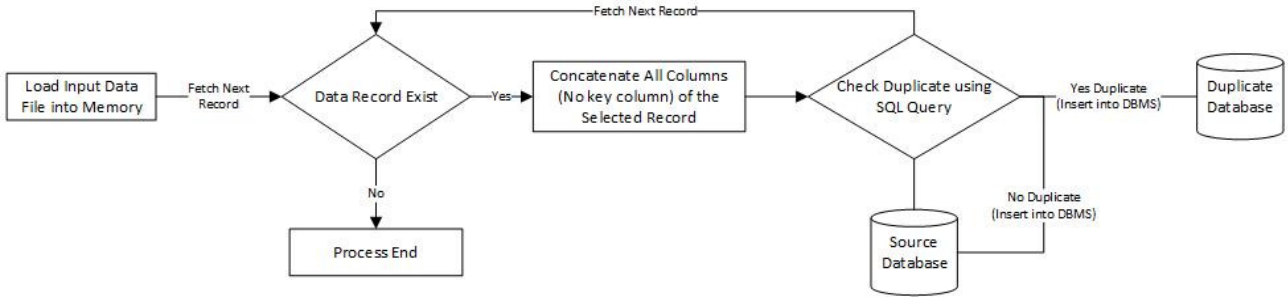

Fig. 3. The workflow of duplication detection using SQL based approach

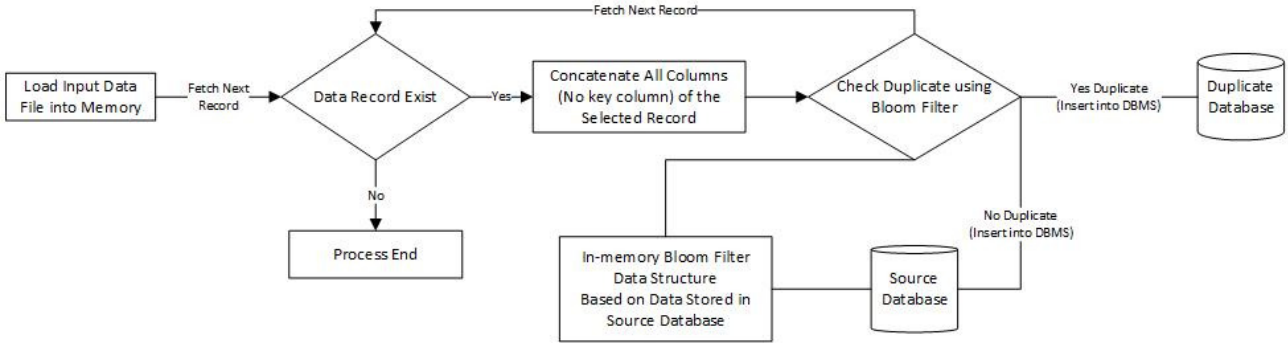

Fig. 4. The workflow of duplication detection using bloom filter-based approach

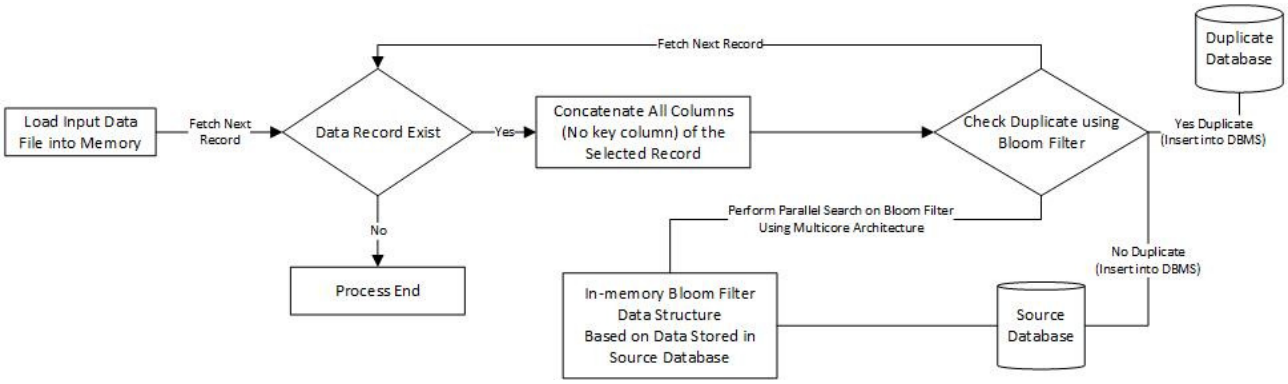

Fig. 5. The workflow of duplication detection using parallel bloom filter-based approach

A. SQL Based Approach

The different steps involved in finding duplicate records using SQL based approach are described below:

- Step 1: In this step, data from the import file are loaded by the application.
- Step 2: In this step, the next record is fetched from the file. If a record exists, then move to Step 3 otherwise end the process.
- Step 3: In this step, all columns of the record are concatenated without key column.
- Step 4: In this step the record (i.e. concatenated columns) has been matched with all existing records (i.e. each record with concatenated columns) in the table for a duplicate check using the Where clause in the Select statement. Figure 6 shows the concatenated column query.
- Step 5: In this step, if the record does not exist, then insert it into the actual table. Otherwise, insert into the duplicate database. Go back to step 2.

```
select * from Students where concat(stud_name, stud_address,
stud_country) = 'Name,Address,country'
```

Fig. 6. The select statement

B. Bloom Filter-Based Approach

The steps involved in finding duplicate records using the bloom filter-based approach are described below:

- Steps 1-3, 5 are the same as in the SQL based approach.
- Step 4: In this step the records (i.e. concatenated columns) are matched with all existing records (i.e. each record with concatenated columns) in the bloom filter without involving the source table in the search process. The bloom filter must be updated for each record inserted into the source table. So, the bloom filter always reflects the current state of the table in the database.

C. Parallel Bloom Filter-Based Approach

The steps involved in finding duplicate records using parallel bloom filter-based approach are described below:

- Steps 1-3, 5 are the same as in the SQL and bloom filter-based approaches.

Step 4: This step is like step 4 of bloom filter, but the only difference is that the records (i.e. concatenated columns) are matched in parallel with all existing records (i.e. each record with concatenated columns) in the bloom filter. This helps in utilizing multiple cores of the host machine and reduces the time required to match all the records.

IV. RESULTS AND DISCUSSION

The workflow of the three approaches used in this paper is presented in Figures 3-5. All three approaches were used to detect duplicates in four different cases. In case I, the table contains 1000 records and the bulk insert file also contains 1000 records (950 unique and 50 duplicate records). In case II, the table contains 10000 records and the bulk insert file also contains 10000 records (9800 unique and 200 duplicate records). In case III, the table contains 100000 records and the bulk insert also contains 100000 records (95000 unique and 5000 duplicate records). In case IV, the table contains 1000000 records and the bulk insert file also contains 1000000 records (850000 unique and 150000 duplicate records). The obtained results (i.e. process time, and time) after the execution of all combination of analysis are presented in Table I.

TABLE I. EXPERIMENTAL RESULTS FOR DUPLICATION DETECTION

Number of Records	Technique	Time (H:min:s.ms)	Time (ms)
Existing: 1000 New: 1000 Unique New: 950 Duplicate New: 50	BF	00:00:00.010	10
	Parallel BF	00:00:00.010	10
	Query	00:00:09.320	9320
Existing: 10000 New: 10000 Unique New: 9800 Duplicate New: 200	BF	00:00:00.050	50
	Parallel BF	00:00:00.030	30
	Query	00:02:26.880	146880
Existing: 100000 New: 100000 Unique New: 95000 Duplicate New: 5000	BF	00:00:00.450	450
	Parallel BF	00:00:00.210	210
	Query	01:14:43.170	4483170
Existing: 1000000 New: 1000000 Unique New: 850000 Duplicate New: 150000	BF	00:00:04.850	4850
	Parallel BF	00:00:01.930	1930
	Query	14:01:23.190	50483190

BF: Bloom filter

Figure 7 compares visually the performance of SQL, bloom filter, and parallel bloom filter approaches. The graph in Figure 7 is plotted using Log_{10} of time in ms instead of time in ms and the number of records to process (the reason this is the rapidly growing difference between the process execution time of SQL and the other approaches with increase in the number of rows to compare) between the processing time of SQL and bloom filter/parallel bloom filter approach. Figure 8 compares the performance of bloom-filter and parallel bloom-filter. Tabular and visual analyses clearly show the high suitability of parallel bloom filter for duplicate detection. It becomes the only viable solution when the number of rows in the table or the rows that need to insert becomes very large.

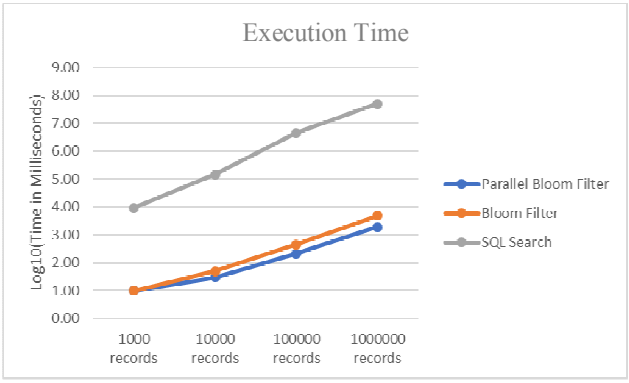

Fig. 7. Execution time for duplicate detection comparison

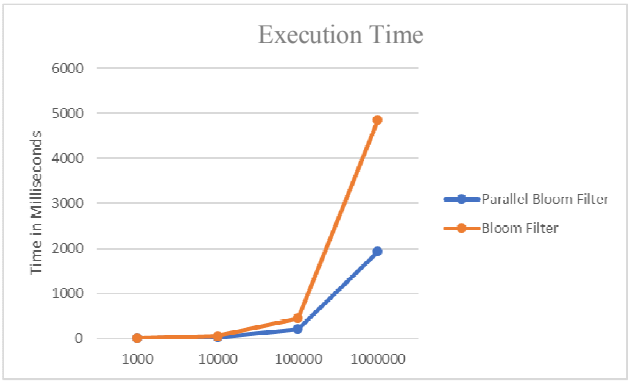

Fig. 8. Execution time for duplicate detection comparison

V. CONCLUSION

This study presented a comprehensive performance analysis of three database duplicate detection techniques. Performance analysis was conducted using different numbers of existing records in the database with bulk data insertion of different sizes. The relative time difference between SQL and bloom filter-based for duplicate detection and insertion rapidly increases with the increase in the record number. The relative time difference between the bloom filter and the parallel bloom filter also substantially increases with the increase of records, although not that rapidly. The research concludes that parallel bloom filter is the most scalable and the optimum solution for duplicate detection in databases, distributed databases, data warehouses, and in general for any application which requires duplicate detection. Due to the advent of modern highly parallel computing architecture, it is highly advisable to implement a parallel version of the algorithm which can scale on multicore and multiple processors to efficiently utilize the aggregate computing power.

REFERENCES

[1] A. K. Elmagarmid, P. G. Ipeirotis, V. S. Verykios, "Duplicate record detection: A survey", IEEE Transactions on Knowledge and Data Engineering, Vol. 19, No. 1, pp. 1-16, 2007

[2] O. H. Akel, A Comparative Study of Duplicate Record Detection Techniques, MSc Thesis, Middle East University, 2012

[3] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors", Communications of the ACM, Vol. 13, No. 7, pp. 422-426, 1970

[4] L. Fan, P. Cao, J. Almeida, A. Z. Broder, "Summary cache: a scalable wide-area web cache sharing protocol", IEEE/ACM Transactions on Networking, Vol. 8, No. 3, pp. 281-293, 2000

[5] F. Bonomi, M. Mitzenmacher, R. Panigrahy, S. Singh, G. Varghese, "An improved construction for counting bloom filters", in: European Symposium on Algorithms, Springer, pp. 684-695, 2006

[6] M. Mitzenmacher, "Compressed bloom filters", IEEE/ACM Transactions on Networking, Vol. 10, No. 5, pp. 604-612, 2002

[7] B. Chazelle, J. Kilian, R. Rubinfeld, A. Tal, "The Bloomier filter: an efficient data structure for static support lookup tables", Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, New Orleans, USA, January 11-14, 2004

[8] A. Kumar, J. Xu, J. Wang, "Space-code bloom filter for efficient per-flow traffic measurement", IEEE Journal on Selected Areas in Communications, Vol. 24, No. 12, pp. 2327-2339, 2006

[9] D. Guo, J. Wu, H. Chen, X. Luo, "Theory and network applications of dynamic bloom filters", 25th IEEE International Conference on Computer Communications, Barcelona, Spain, April, 23-29, 2006

[10] S. Geravand, M. Ahmadi, "Bloom filter applications in network security: A state-of-the-art survey", Computer Networks, Vol. 57, No. 18, pp. 4047-4064, 2013

[11] Y. Emami, R. Javidan, "An Energy-efficient Data Transmission Scheme in Underwater Wireless Sensor Networks", Engineering, Technology & Applied Science Research, Vol. 6, No. 2, pp. 931-936, 2016