Database Security Enhancement by Eliminating the Redundant and Incorrect Spelled Data Entries

Rupali Chopade* and Vinod Pachghare

Department of Computer Engineering and IT, College of Engineering Pune, Savitribai Phule Pune University, India
E-mail: rmc18.comp@coep.ac.in; vkp.comp@coep.ac.in
**Corresponding Author*

Received 29 September 2020; Accepted 25 November 2020;
Publication 29 March 2021

Abstract

Database is used for storing the data in an easy and efficient format. In recent days large size of data has been generated through number of applications and same has been stored in the database. Considering the importance of data in every sector of digitized world, it is foremost important to secure the data. Hence, database security has been given a prime importance in every organization. Redundant data entries may stop the functioning of the database. Redundant data entries may be inserted in the database because of the absence of primary key or due to incorrect spelled data. This article addresses the solution for database security by protecting the database from redundant data entries based on the concept of Bloom filter. This database security has been obtained by correcting the incorrect spelled data from query values with the help of edit distance algorithm followed by the data redundancy check. This article also presents the performance comparison between proposed technique and MongoDB database for document search functionality.

Keywords: Database security, redundancy, spell checker, Bloom filter, Edit distance.

Journal of Cyber Security and Mobility, Vol. 10_2, 403–420.
doi: 10.13052/jcsm2245-1439.1024

1 Introduction

The primary key has the important concept in database management systems [1]. This key plays an important role while storing data in the database and is beneficial to avoid the data redundancy. Data redundancy is storing same data multiple times [2, 3]. Minimal data redundancy will keep the database consistent and correct execution of data manipulation operations [4]. Data redundancy has always given a thought from the perspective of data availability. If one copy of database contains redundant data then same data will be replicated in distributed environment. Can this data redundancy affect the database security? Yes, database denial of service attack may be possible [5, 6]. Attackers are moving their targets from networks to servers, servers to application layers, towards the databases. Though denial of service attack may not damage the database or may not expose the sensitive data, but the mission of attacker is to slow down the system, or to crash the database server or may be to hide the malicious activities happening behind. Normalization is the primary concept in relational database management systems [7] to avoid the data redundancy problem. In normalization different normal forms have been used based on functional dependency, which involves the concept of key. But, while creating tables in databases like MySQL, MongoDB it is not mandatory to specify the primary key. In such scenario, it has the possibility of inserting redundant data in the database, which will also lead to the problems of storage space and associated cost. This type of problem may occur majorly in NoSQL databases. In relational database, the structure of the database is fixed so the chances of redundant data entries are less. But in NoSQL database the schema is unstructured which allows to insert all types of documents. The other problem associated with database is the repetition of similar data [8] due to spelling mistakes while inserting values (especially string data). Similar data is not exactly the same data, but due to spelling mistakes similar type of data may get inserted in the database. When the data is inserted using web based applications, this problem may not occur for the key string, but if the data is inserted from database shell then there are chances of similar data insertion for key as well as for values in key-value based databases. In this article a technique has been proposed by developing application which consists of two modules namely spell checker and redundancy checker. The redundancy checker module has used to avoid the duplicate data insertion problem, in absence of primary key. The spell checker module has used to solve the problem of similar value insertion by suggesting the corrected query to the user. The proposed technique has been

evaluated on MongoDB database, but it is useful to any database whenever there are chances of redundant data entries.

Paper Layout

The rest of this article is organized as follows. The related work has mentioned in Section 2 and data redundancy problems has been discussed in Section 3 for MongoDB database. A proposed technique for avoiding data redundancy and spell checker for database has explained in Section 4. The mathematical model for this implementation has shown in Section 5. The results have been discussed in Section 6 and finally, we conclude this article in Section 7.

2 Related Work

Hong has analyzed data redundancy problem in object-oriented databases [9]. The proposed method has extended data normalization concept to analyze redundancy within classes as well as among the classes. The method progresses by finding abstractions among the class. The cohesion (software engineering term) within the abstraction has captured by using two relationships namely a-dependency and m-dependency. The a-dependency shows the connection among attributes and m-dependency links the method to attributes in an abstraction. Further these dependencies have been used to define inference rules. The degree of data redundancy in a class has measured using first, second and third well-defined forms, which are similar to normal forms used in relational databases.

The data redundancy impact on intrusion detection system [10] has been explored by Al-Rawi et al. The observation presented by researchers: if the dataset contains data redundancy then the intrusion detection systems may shows the unstable or fluctuated performance. They proved that the original KDDCUP99 dataset contains the redundant data. The researchers deleted redundant data from KDDCUP99 dataset and derived new dataset, KD9. This new dataset contains the data size almost one fifth of the original dataset. The performance comparison among KDDCUP99 and KD9 dataset has presented for more than 800 IDS training/testing experiments. The performance of test accuracy is measured using F1 score. The variation in the results for these two datasets has been presented. The researchers working in the domain of IDS can use this as a guideline by selecting the dataset carefully.

The storage allocation system on cloud has been presented by Huang et al. [11] to reduce the data redundancy problem. The redundant storage of data on cloud may lead to the problems like system overhead, overall increase in cost and energy consumption at the cost of increasing reliability. The proposed system gives high reliability with minimum data redundancy. The properties of proposed system are analyzed using various theorems and prepositions. Minimizing redundancy has based on two schemes. In the first scheme find minimum n with fixed k and in the second one, find maximum k with fixed n. The k and n are terms used from replication and erasure schemes.

Altarawneh has presented [12] survey on spelling error detection techniques for Arabic language. Many researchers have followed the approach of rule-based, N-gram score, Radix search tree, direct detection and detection based on language dictionary.

Kukich has presented solution for word correction in text [13]. The first solution has based on pattern matching and n-gram analysis to detect non-word error. For the correction of isolated word, second solution has developed based on application specific correction techniques. For correction of isolated word problem, language processing tools have been used.

Jayalatharachchi et al. explored the research work for spelling error detection and correction for Sinhala, which is the most preferred language of Shri Lanka [14]. They have shown an improvement for an existing n-gram based approach called as Subasa-data driven spell checker. The extension of edit distance algorithm dictionary (by adding more words to it) and ranking of spelling error correction for subasa will remain in the future scope.

Turchin et al. evaluated algorithm to identify misspelled words from medical records [15]. The prototype implementation has used three text files. The first one has representation of narrative document containing one or many plain text files having unlimited size, the second one contained text file for which words has been identified for misspelling. The third word contains English vocabulary text file. The algorithm has evaluated with recall, specificity, false positive fraction, positive predictive value and F-measure.

3 Data Redundancy Problems

Existing research has presented that there are significant problems associated with data redundancy [16, 17]. In this section, how the repeated data may get stored in the database has shown with the examples from MongoDB database. In this database, if _id is used for key field, then that field will be used as

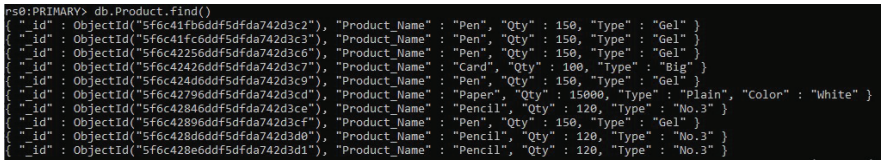


Figure 1 MongoDB redundant data.

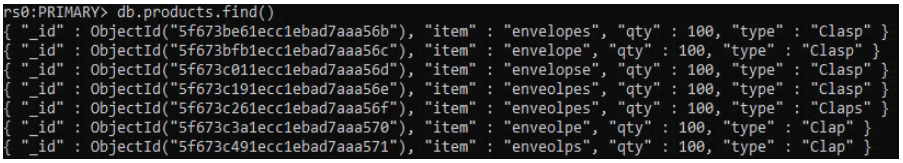


Figure 2 MongoDB redundant data due to spelling mistakes.

primary key [18]. If no key has specified, then it will insert _id as primary key and same data may get inserted multiple times. This problem is depicted with an example in Figure 1.

From this example it can be observed that, same product name, qty and type have been inserted multiple times, because _id field is not marked as primary key, while creating collection or during insert query. MongoDB has internally assigned primary key in the form of _id, but the remaining values of key-value pair have repeated unnecessarily.

In the example from Figure 2, key field "item" consists of same string "envelopes" which has inserted repeatedly with spelling mistake by replacement of characters. This may happen due to typing errors. The same case may be observed for key "type" and value is "clasp" The spell checker module will identify such kind of errors and will notify user about it.

4 Proposed Technique

The architectural flow for the proposed technique is shown in Figure 3. The architecture consists of two modules namely "Spell Checker" and "Redundancy Checker". The spell checker module checks that if data from insert query issued by user has any data spelling corrections. If user query contains any spelling corrections, this module suggest the new query to the user by correcting spelling of values. User has choice to keep old query or updated query. Depending on user's input, either user query or corrected query will be given to Redundancy Checker module. This module checks if queried data already exists. If data is already available in the database then user query will

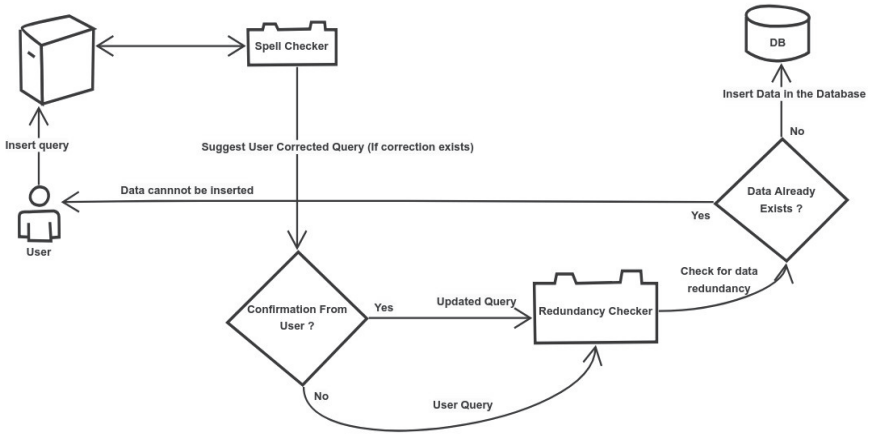


Figure 3 Proposed technique architecture.

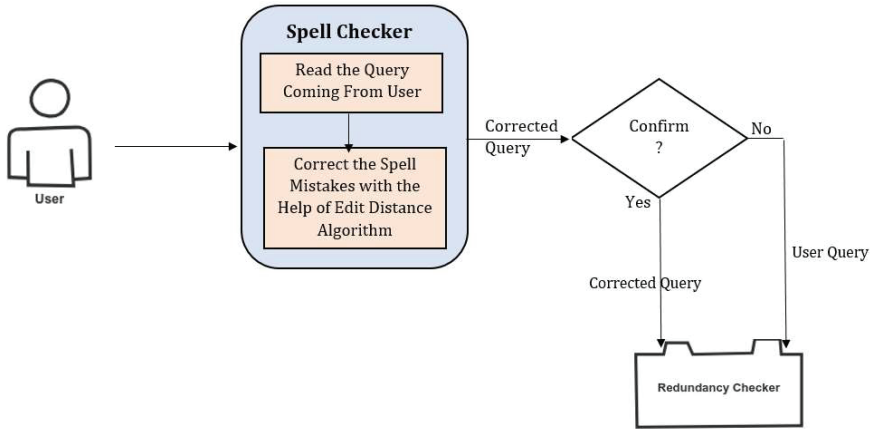


Figure 4 Spell checker module.

be denied else it will get inserted in the database. The detailed working flow of these modules is shown in Figures 4 and 5 respectively.

Spell Checker

The working of this module (Figure 4) is based on edit distance algorithm [19]. The algorithm uses dynamic programming approach to find the corrected word matching from stored words dictionary. The corrected word

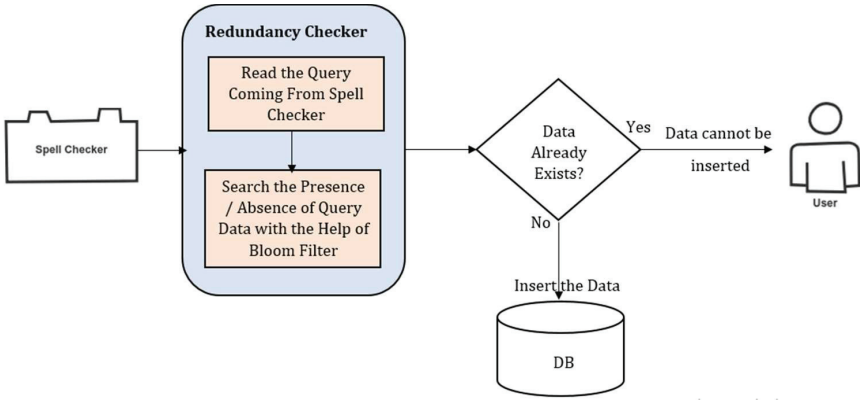


Figure 5 Redundancy checker module.

has derived by matching query string and dictionary string with minimum edit distance. The calculation of minimum edit distance has explored in Section 5. Depending on outcome of this algorithm the corrected query (If corrections exists and these corrections performed by module) has shown to user for confirmation. If user agrees to keep the corrected query or wish to keep the existing query, the respective query will be forwarded to Redundancy Checker module.

Redundancy Checker

The redundancy checker module shown in Figure 5, accepts the query coming from spell checker module. The working of this module is to verify that the issued data has already available in the database or not. For search purpose this module has used the functionality of Bloom filter [20]. The Bloom filter is memory efficient data structure for checking existence of member in a set. This data structure is preferred due to its fast working. Here it will verify whether the values are already present. If yes, then insert query will be discarded by informing the query status to the user. If the values are not already available then insert query will succeed by inserting the data in the database.

5 Mathematical Model

The mathematical model behind the working of data redundancy and spell checker has explained in this section. Bloom filter is one of the efficient data

structure for checking whether data member is already available in set or not [21]. By using this concept, the insertion of redundant data entries in the database can be stopped. The Bloom filter is a bit vector array, in which all bits index is initialized to zero. When any string has to be stored to an array, it will pass through few number of hash functions (by calculating optimal value of hash − k) and based on resultant hash value, the index is marked to one. While storing new string, it will pass through same hash functions and if calculated hash values are same [22], it will notify that the string is already present. Otherwise new string will gets stored into vector array. The working of Bloom filter is shown with an example.

The optimal value of k (number of hash functions) is given by Equation (1),

$$k = \left(\frac{m}{n}\right) \ln(2) \tag{1}$$

The Bloom filter may suffer from the false positive probability. If element is not present in the array but its existence is returned as present, is termed as false positive probability. The false positive probability *P* is calculated using Equation (2),

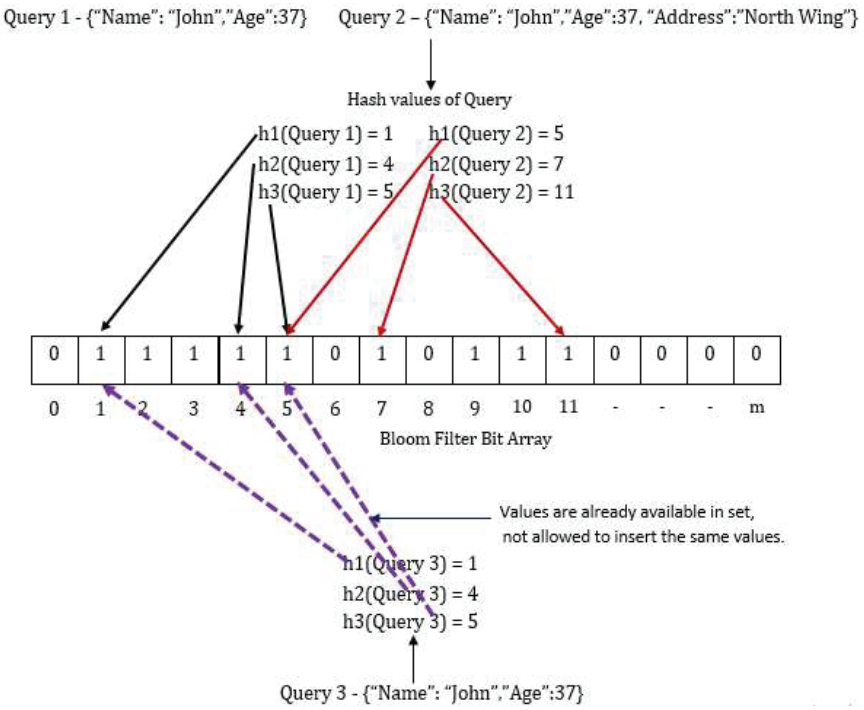
$$P = \left[1 - \left(1 - \frac{1}{m}\right)^{k.n}\right]^{k} \tag{2}$$

Where,

p – False positive probability
m – Length of Bloom Filter
k – Number of hash functions
n – Number of elements to be inserted

Following example shows the working of Bloom filter used in the proposed technique. When first document is inserted in the array, it will pass through three hash functions namely h1, h2 and h3. Depending on return values for hash, the respective index from bit array will be set to 1. The same process has repeated for second document coming from query 2. When query 3 has been issued, which contains the same document as query 1, it will have same hash values. Hence, here bit array will return the result as data already present and not allowed to insert the same values.

The String spell corrector is based on Levenshtein distance (edit distance) algorithm. The algorithm calculates minimum edit distance between two strings, which uses the concept of dynamic programming. The new string gets replaced by original string with any three types of edit operations which are



replacement, insertion or deletion. For the two strings $s(s_1,s_2,s_3\ldots,s_m)$ and $t(t_1,t_2,t_3\ldots,t_n)$, the cost associated with changing s_i into t_j, is given by $C(s_i,t_j)$ Calculation of minimum edit distance is given by $sd(s,t) = D(m,n)$ using adjacency matrix [19, 23]. The conversion of string "syeeet" into corrected string "street" is shown by generating adjacency matrix. The string to be verified has written in the first row and the corrected string has written in the first column. Then each character from string (present in the row) will be scanned to find the number of changes required to obtain the corrected string.

$s \rightarrow s$ (no change, so value is 0)
$s \rightarrow st$ (insert t after s, so value is 1)
$s \rightarrow str$ (insert tr after s, so value is 2)

By continuing the same process, following matrix has generated. The value obtained at the intersection of last row and last column (corner value) is the minimum edit distance.

Adjacency Matrix for Minimum Edit Distance

	" "	**s**	**y**	**e**	**e**	**e**	**t**
" "	0	1	2	3	4	5	6
s	1	0	1	2	3	4	5
t	2	1	1	2	3	4	4
r	3	2	2	2	3	4	5
e	4	3	3	2	2	3	4
e	5	4	4	3	2	2	3
t	6	5	5	4	3	3	**2**

The similarity among two strings is given by Equation (3),

$$1 - \frac{sd(s,t)}{\max(m,n)} \tag{3}$$

Where,

sd(s,t) – minimum edit distance among two strings s and t
m – The length of first string
n – The length of second string

Where sd(s,t) $= 2$ (marked with circle in Adjacency matrix)
Similarity among two strings $= 1 - \frac{2}{6} = 0.66$

6 Results and Discussion

A system with specifications: windows 10 64-bit OS, 3 GB RAM with Intel Core 2.40 GHz i3 processor has been used for implementation. Application has been designed with python 3.7, MongoDB version 4.0. The operations are performed on sample collections to verify the result.

The corrected query, suggested by spell checker module has shown in Figure 6. This module works for spell correction of single document as well as multiple documents. The result for multiple documents has shown in Figure 7.

The limitations of edit distance algorithm [24] can be observed from Figure 8. Here for key "wing" the value is given as "sout" and expected correction is "south" but the suggestion given by edit distance is "out", though minimum edit for both the cases is 1.

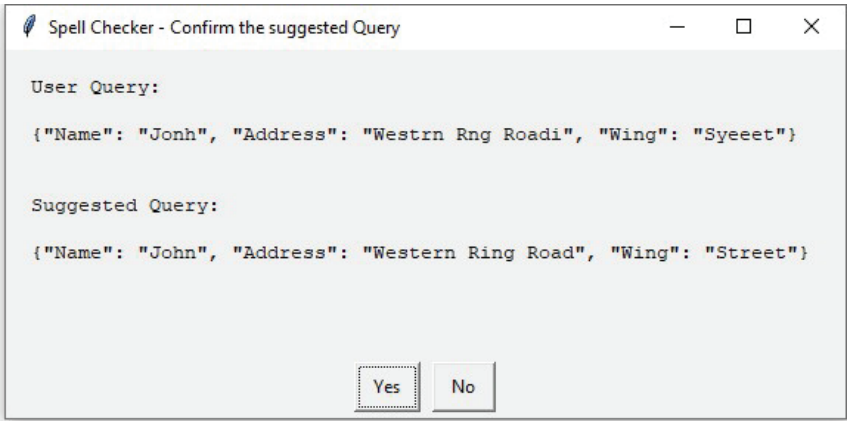


Figure 6 Spell correction for single document query.

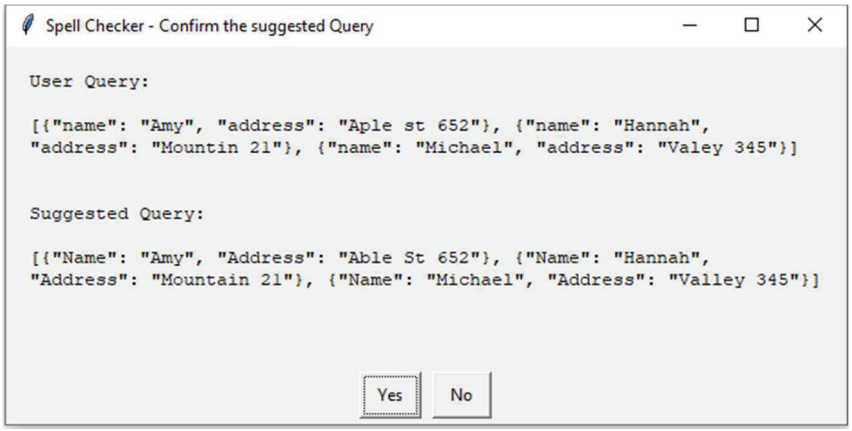


Figure 7 Spell correction for multiple documents query.

Sout → south (Insertion of h at the end)
Sout → out (Deletion of s from beginning)

The result of redundancy checker module is shown in Figure 9.

For experimentation purpose, sample collection has created and the data sets for experimentation were downloaded from GitHub [25] for the MongoDB database. The MongoDB WiredTiger storage engine stores the data in compressed form. So the experimentation was carried out on the 336 MB as the compressed database size. By knowing the number of elements to be added in the Bloom filter and setting the desired false positive probability, the

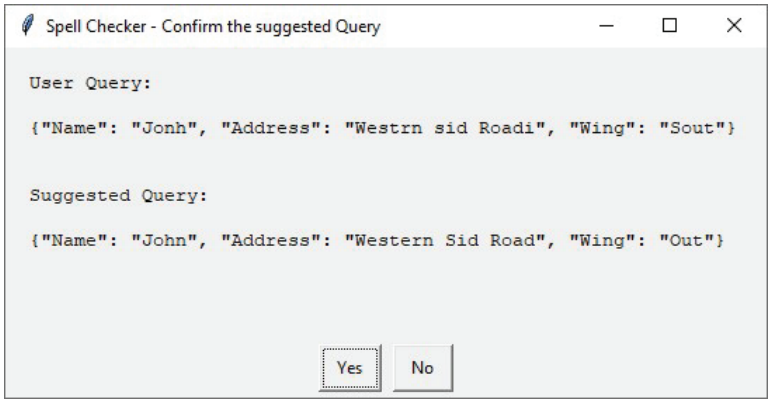


Figure 8 Spell correction result.

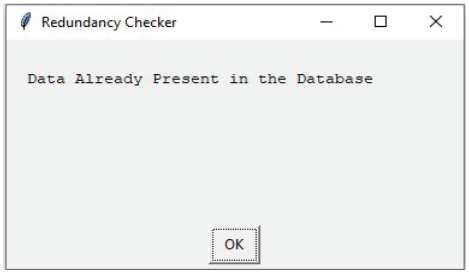


Figure 9 Presence of data in the database.

size of bit array can be calculated as shown in Equation (4) [26].

$$m = -\frac{n \ln P}{(\ln 2)^2} \tag{4}$$

Where,

m – The size of bit array
n – Number of elements
p – False positive probability

The probability of false positive [27] is the element that we wish to insert in the array and status shown by array that element has already present but that particular element has never added to the array. This happens because the status is shown by checking the index position of bit array. If that index is updated to 1 by other elements, the status is shown as already present. By knowing n = 111239 (Number of documents inserted in the bit array) and desired false probability P = 0.05, the calculated bit array size is m = 693600.

The optimal value of hash function (k) in the Bloom filter has calculated using Equation (1) (Section 5).

$$\mathrm{k} = \left(\frac{693600}{111239}\right) \ln(2)$$

$$\mathrm{k} = 4$$

To check the performance, the desired false probability has been set to 0.01 from 0.05 and n is the same which is $\mathrm{n} = 111239$.

Using these values, the bit array size (m) and number of hash functions (k) have calculated as,

$$\mathrm{m} = 1066232$$

$$\mathrm{k} = 6$$

The benefit of Bloom filter is that it is faster as compared to database search method and this can be observed from Table 1. The change in performance by using different values for false positive probability has been presented here.

The number of documents available in Bloom filter and database are around 111239. The graphical analysis of the same is shown in Figure 10.

Table 1 Document search execution time comparison between proposed technique and MongoDB database

No. of Documents Search	Document Search Execution Time (Seconds)		
	Proposed Technique		
	False Positive Probability = 0.05, k = 4	False Positive Probability = 0.01, k = 6	MongoDB Database
1	0.0009	0.0009	0.0009
10	0.001	0.001	0.002
100	0.001	0.003	0.03
500	0.01	0.01	0.03
1000	0.02	0.02	0.06
5000	0.12	0.13	0.32
10000	0.26	0.28	0.7
50000	1.44	1.56	3.46
100000	2.8	3.18	6.17
500000	14.35	16.15	32.64

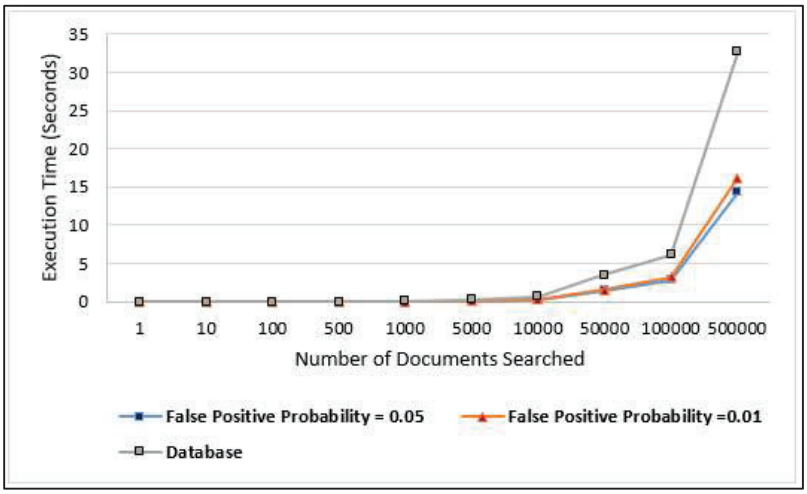


Figure 10 Execution time comparison between proposed technique and MongoDB database for document search.

It can be observed from the graph that, up to 10k documents search, the time requirement is almost similar for both the methods. But from 50k documents onwards, the time requirement for proposed technique is less as compared to database search method. The performance of proposed technique is good when the number of documents to be inserted are more. Even the better performance has been observed when the number of hash functions are 4 and false positive probability set to 0.05 in comparison with hash functions 6 and false positive probability as 0.01.

7 Conclusion

Security of database from external attacks is considered as the challenging task. The internal security of the database is equally important. The storage of same data multiple times may make the database inconsistent, when primary key has not defined for the records. Similarly due to typing mistakes there are chances of storing incorrect data in the database multiple times. A technique has been proposed to solve these two problems related to internal security of database. The proposed technique will work for all the databases, whenever tables or collections are created without mentioning the primary key. The redundancy checker module has based on the concept of Bloom filter. For spell checker module the edit distance algorithm by Levenshtein has refereed.

Due to accuracy limitations of edit distance algorithm, improvement to this algorithm will remain in the future scope.

References

[1] Jiang L, Naumann F (2019) Holistic primary key and foreign key detection. J Intell Inf Syst 1–23
[2] Date CJ (2019) What Is Database Design, Anyway? In: Database Design and Relational Theory. Springer, pp 393–406
[3] Link S, Prade H (2019) Relational database schema design for uncertain data. Inf Syst 84:88–110
[4] Vighio MS, Khanzada TJ, Kumar M (2017) Analysis of the effects of redundancy on the performance of relational database systems. In: 2017 IEEE 3rd International Conference on Engineering Technologies and Social Sciences (ICETSS). pp 1–5
[5] (2013) Dealing with Database Denial of Service Licensed by DB Networks
[6] DoS-in Your Database. https://www.darkreading.com/application-security/database-security/dos-in-your-database/d/d-id/1139881
[7] Bahmani AH, Naghibzadeh M, Bahmani B (2008) Automatic database normalization and primary key generation. In: 2008 Canadian Conference on Electrical and Computer Engineering. pp 11–16
[8] Zhu M, Shen D, Nie T, Kou Y (2009) An Adjusted-Edit Distance Algorithm Applying to Web Environment. In: 2009 Sixth Web Information Systems and Applications Conference. pp 71–75
[9] Hong S (1995) A method for analyzing and reducing data redundancy in objectoriented databases
[10] Al-Rawi M, Al-Zuqary Y, Saghezchi FB, et al (2017) Data redundancy may lead to unreliable intrusion detection systems. In: 2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC). pp 1897–1902
[11] Huang Z, Chen J, Lin Y, et al (2015) Minimizing data redundancy for high reliable cloud storage systems. Comput Networks 81:164–177
[12] Altarawneh R (2017) Spelling detection errors techniques in NLP: A survey. Int J Comput Appl 172:1–5
[13] Kukich K (1992) Techniques for automatically correcting words in text. Acm Comput Surv 24:377–439
[14] Jayalatharachchi E, Wasala A, Weerasinghe R (2012) Data-driven spell checking: the synergy of two algorithms for spelling error detection

and correction. In: International Conference on Advances in ICT for Emerging Regions (ICTer2012). pp 7–13
[15] Turchin A, Chu JT, Shubina M, Einbinder JS (2007) Identification of misspelled words without a comprehensive dictionary using prevalence analysis. In: AMIA Annual Symposium Proceedings. p 751
[16] DeCastro-García N, Muñoz Castañeda ÁL, Fernández Rodríguez M, Carriegos M V (2018) On detecting and removing superficial redundancy in vector databases. Math Probl Eng 2018
[17] Nalini M, Anbu S (2016) Elimination of Data Redundancy before Persisting into DBMS using SVM Classification. Int J Eng Res 5
[18] Mongodb Primary Key: Example to set _id field with ObjectId(). https://www.guru99.com/mongodb-objectid.html
[19] Zhang S, Hu Y, Bian G (2017) Research on string similarity algorithm based on Levenshtein Distance. In: 2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC). pp 2247–2251
[20] Patgiri R, Nayak S, Borgohain SK (2019) Role of bloom filter in big data research: A survey. arXiv Prepr arXiv190306565
[21] Doniparthi G, Mühlhaus T, Deßloch S (2020) A Bloom Filter-Based Framework for Interactive Exploration of Large Scale Research Data. In: European Conference on Advances in Databases and Information Systems. pp 166–176
[22] Jing C, Zhengang N, Liying L, Fei Y (2009) Research and application on Bloom filter in routing planning for indoor robot navigation system. In: 2009 Pacific-Asia Conference on Circuits, Communications and Systems. pp 244–247
[23] Rani S, Singh J (2017) Enhancing Levenshtein's edit distance algorithm for evaluating document similarity. In: International Conference on Computing, Analytics and Networks. pp 72–80
[24] Greenhill SJ (2011) Levenshtein distances fail to identify language relationships accurately. Comput Linguist 37:689–698
[25] GitHub – ozlerhakan/mongodb-json-files: A curated list of JSON / BSON datasets from the web in order to practice/use in MongoDB. https://github.com/ozlerhakan/mongodb-json-files
[26] Calculate the required bloom filter size and optimal number of hashes from the expected number of items in the collection and acceptable false-positive rate GitHub. https://gist.github.com/brandt/8f9ab3ceae37562a2841
[27] Bose P, Guo H, Kranakis E, et al. (2008) On the false-positive rate of Bloom filters. Inf Process Lett 108:210–213

Biographies

Rupali Chopade is a full time Research Scholar under AICTE-QIP scheme, at Department of Computer Engineering and IT, College of Engineering Pune, India. She is working as Assistant Professor at Department of Information Technology, Marathwada Mitra Mandal's College of Engineering Pune, India. She has 17 years of teaching experience. Her research interest includes database forensics and database security. She has received "Distinguished HOD "Award by Computer Society of India (CSI) in 2017.

Vinod Pachghare is Associate Professor in the Department of Computer Engineering and Information Technology, College of Engineering, Pune (An autonomous institute of Government of Maharashtra), India. He has 29 years of teaching experience and has published the books on Cloud Computing and Computer Graphics. Dr. Pachghare has over 37 research publications in various international journals and conferences. His area of research is network security. Also he is a member of Board of studies in Computer

Engineering/Information Technology of a number of Autonomous Institutes. He is an Investigator for the Information Security Education and Awareness [ISEA] Project, Ministry of Information Technology, Govt. of India. He was a Principal Investigator for a research project "Wireless IDS", sponsored by AICTE, New Delhi. He delivered lectures on recent and state of the art topics in Computer Engineering and Information Technology as an invited speaker. He has received "Best Faculty Award" 2018 by CSI, Mumbai Chapter.