International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459, ISO 9001:2008 Certified Journal, Volume 7, Issue 5, May 2017)

# A Graph Based Obfuscation Tool for Database Security

Saurabh Kulkarni
*Assistant Professor, Fr. Conceicao Rodrigues College of Engineering*

***Abstract*— In the present times, large amount of data is available from numerous sources. Organizations utilize the information and knowledge derived from the data to function and generate profit. Thus maintaining the data and its safety is vital. But the data stored in databases shouldn't be visible to every user. Therefore, to the third party users and testers the data should look real but shouldn't be the actual data. This paper attempts to describe a graph based model to create a tool for database security using techniques like data masking, obfuscation, and shuffling.**

***Keywords*—Data Obfuscation, Database Security, Data Shuffling, Graph Based Model, Substitution.**

## I. INTRODUCTION

Databases form a backbone of most of the applications. Databases may contain data which can be viewed publicly or data which must be visible to certain group of users. Development and testing in a software firm may be performed by third party developers and testers. So there is a necessity to secure data in non-production environment.

Security of the data can be achieved by techniques like encryption, access control mechanisms etc. One of such techniques can be database obfuscation. Obfuscation is the hiding of intended meaning in communication, making communication confusing, willfully ambiguous, and harder to interpret. In database obfuscation, data looks real though it's not the actual data. So even though third party developer has access to data, actual data is not revealed to him/her. Databases may contain data which can be viewed publicly or data which must be visible to certain group of users. In organizations where third party developers and testers are involved, it becomes essential to secure data in databases which is sensitive as far as the business of the organization is concerned. Even some organizations want very sensitive data to be hidden from their own employees. These developers or testers usually want data which is realistic in nature as with that they can develop and test different applications but organizations can't reveal actual data which can be sensitive from its perspective. So data presented to these users should be realistic in nature but it should not be actual data. Our proposed system uses database obfuscation to tackle this problem.

There are techniques like encryption which can hide sensitive data but it converts data into hexadecimal values which don't look realistic as far as developers and testers are concerned. So we are proposing a model for securing data in non-production environment using database obfuscation technique.

Data in databases is constrained by the relational database constraints enforced by database administrator as well as business level constraints enforced by the organization. E.g. Primary key, foreign key, unique key etc. are the constraints enforced on the database schema by the database administrator. There can be a business level constraint enforced by organization for example total sells of a product can't exceed 100 per day. In data warehouse applications may be a key is formed by combination of some attributes of different tables. In this case, key depends on many attributes of other tables. So while obfuscating such attributes, care should be taken to obfuscate dependent attributes beforehand. So in this paper, we have proposed a model to obfuscate different attributes of databases considering the constraints involved.

## II. REVIEW OF LITERATURE

Information present in databases may be sensitive from business viewpoint of an organization.

And some of the applications of the organization may require this sensitive data. The production environment of organization where the application is deployed in real time usually has some strict access control policies and not every user has access to all data in this environment. In non-production environment where usually development and testing activities take place, data is available to many developers and testers. Sometimes those developers and testers may be third party users. As the application under development and testing requires data which looks realistic, some mechanism of giving it to the user while still securing it is required. Such mechanism is often called as data obfuscation or data sanitization or data masking or data scrubbing [1].

Usually, the structure or schema of data in non-production environment is same as that of production environment.

The basic criterion for data in production environment is that it should be realistic and adhering to the constraints specified. The data here need not be actual as long as it is helpful in development and testing. [2][8]

There are broadly two architectures as far as data obfuscation is concerned.

- *On the fly architecture*- Here data is scrambled when it is transferred from one database to other. This can be typically used in data warehouse environment.
- *Situation based architecture*- In this type of architecture, data is duplicated using some technique and second database is created. Data Obfuscation is applied on this database [3].

In order to mask different types of data, variety of obfuscation techniques should be used. Some of these techniques include:

- *Substitution*- Substitution is commonly used technique in encryption. Characters in plain text are replaced by some other characters in substitution. [4][8]. In case of database obfuscation, contents of the column are substituted with some other text in the same domain of data of that column. [2][3]. For example, product “ABC” will be replaced by product “XYZ” from the set of valid values of product. There is a need to create a translation table which can be dynamic in nature which provides mapping of original text with substituted scrambled text. [5]
- *Shuffling*- As far as database obfuscation is considered, shuffling implies randomly changing data values of one of the columns with other data value in the same column. So in shuffling, data values are randomly shifted between the rows of a table.[3]. This technique is usually efficient when table is large, as large data can help to incorporate more randomness in shuffling.

In numerical data, it can be performed using rank-order data and hence there is no requirement to pass parameters in the scrambling method.[6]

- *Number and date variance*- Variance technique is suitable for date and numerical data. A small random amount of value is added or subtracted in all values of a column. So data in column adheres to its data type but values have small amount of variance. In case of dates, similar kind of variance in terms of number of days can be added. E.g. If we decide to add 10% variance in a particular column, then if one of the value is 100, it will be replaced by 110.[3]. Even random numbers can be used to provide variance in numerical or date fields. So different variance can be added to different rows of the same column. Properties of random numbers like portability, efficiency, homogeneity are helpful to generate good variance [7]
- *Gibberish Generation*- While performing data scrambling, all the references to real data should be removed. For scrambling formless data life letters, memos and notes, gibberish or random words are generated and original words are substituted by these words. [2]

These are some of the methods which can be used to scramble data. In case of databases, care should be taken to make data consistent with the business rules after scrambling. All the database level and business level constraints should be satisfied even after scrambling takes place. All these methods should be used in combination to effectively sanitize data. Following section describes our proposed model which can help to maintain the business level constraints in database scrambling.

## III. Model For Scrambling Data In Databases

Consider a simple relation in databases as shown below. From this example we can infer that Total

**TABLE I**
**SAMPLE RELATION EXAMPLE**

| Basic Salary | Tax | Total Pay |
|---|---|---|
| 50000 | 10000 | 40000 |
| 75000 | 15000 | 60000 |
| 35000 | 7000 | 28000 |

Pay is dependent on Basic Salary and Tax, Tax is dependent on Basic Salary. We can represent this structure like a tree as follows: add fig. here Thus for relational databases, following four cases exists:

### *A. Independent Node*

This is a single attribute considered in the table. The value of this node remains unaffected due to any other node. Thus data masking of such a node becomes relatively easiest to handle. Add fig. here

**Fig. 1. Independent Node**

*B. Hierarchical Dependency*

Each node depends on the succeeding node. Therefore, the root node depends on the next node and so on. We can represent it as, the (n-1) the node depends on the nth node. Thus, changes should be reflected in a similar manner.

Example: The data flows from branch offices to the main office at the city level, then state level and finally the headquarter

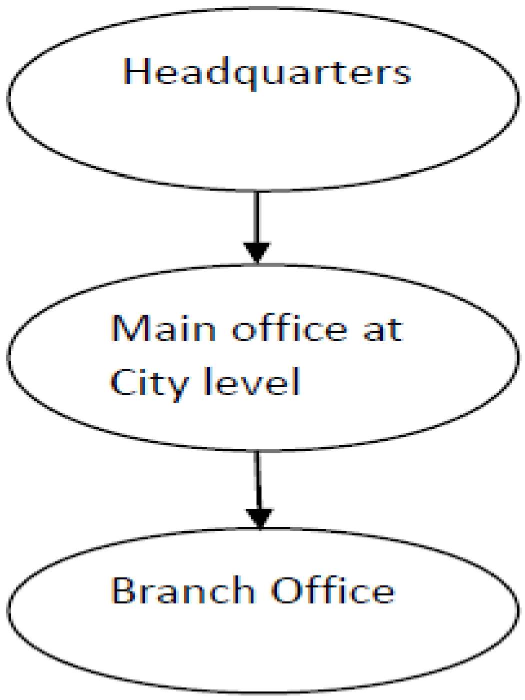

**Fig. 2. Hierarchical Dependency**

*C. One leaf node-Two parent nodes*

In this case one attribute is affected by 2 attributes directly and not in a hierarchical fashion. While masking such a leaf node it is important that the two parent nodes are handled first. Any change in the parent node directly affects the leaf node attribute.

Example: Turnover of locks department and turnover of furniture department affects the turnover business of a company.

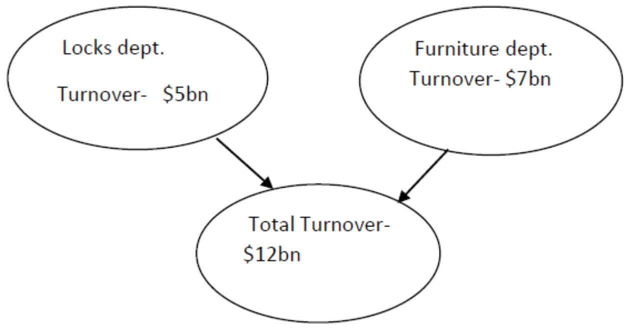

**Fig.3. One leaf-node two parent nodes**

*D. One parent-Two leaf nodes*

In this case there is one parent attribute which affects other attributes. Any change in the parent attributes affects the attributes that depend on it. While masking such a parent node, subsequent changes have to be reflected while masking the leaf nodes.

Example: In a finance budget, if the amount of money through various sources of income increases, the amount of many available for expenditure by various sectors also increases.

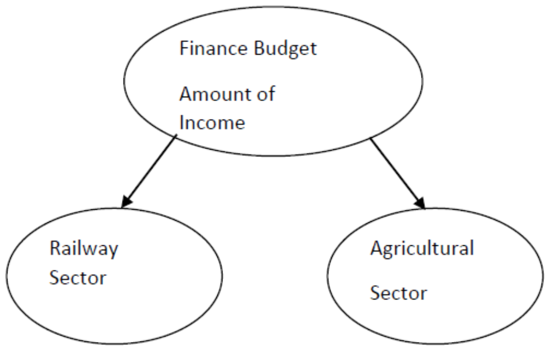

**Fig. 4. One parent two-leaf nodes**

*E. Traversal techniques*

For scrambling data modeled in tree like structure, following two traversal techniques can be followed.

*E.1 Depth first search:*

To locate the required data, the searching movement is from top-to bottom along the depth of the tree E.g.: According to Depth First Search, we traverse the graph in the following manner: A->B->D->C->E->F Add fig. here

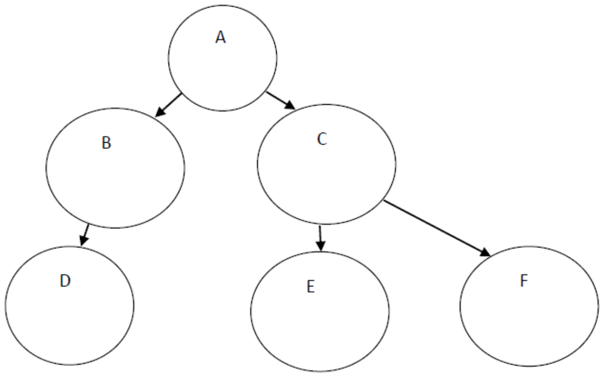

**Fig.5. Depth First Search**

*E.2. Breadth First Search:*

To locate the required data, the searching movement is from left to-right along the width of the tree E.g.: According to Breadth First Search, we traverse the graph in the following manner: A->B->C->D->E->F. Add fig. here.

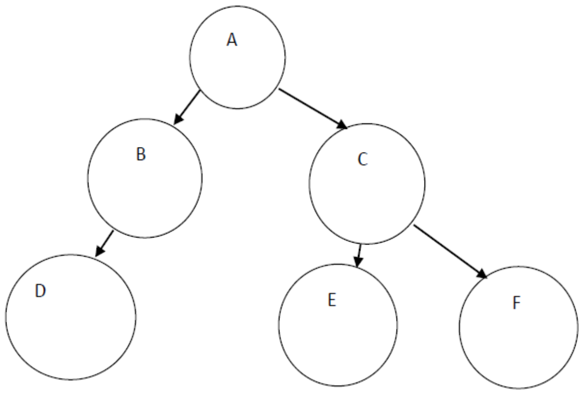

**Fig. 6. Breadth First Search**

## IV. ALGORITHM FOR DATA SCRAMBLING

There are two data structures which this algorithm uses namely

- *Dataholder*- The dataholder does the work of retrieving the data from database. This data is obfuscated based of the type of data the column holds. For eg: if the column is of type 'varchar' then it'll use the method shuffling, if data is of type integer then it will do number variance or substitution. Obfuscated data is sent to datastruct which displays in a tabular manner.
- *Datastruct*- It stores the obfuscated data of every object. It is helpful in printing obfuscated data.

**Algorithm 1** Algorithm for scrambling database

```
Declaration table holder with columns tables, no of columns, relation;
Declare arrays of table, columns, relation;
Identify type of data present;
Direct the page to dataholder;
if Data is of type integer,long then
  Use substitution, number variance
else
  Use shuffling on the array
end if
if Length of relation array >1 then
  Get relations among columns
else
  Obfuscation methods on individual nodes
end if
if Datatype is integer and length >7 then
  Redirect to dataholder
  Use substitution method here.
else
  Redirect to dataholder
  Use number variance technique
end if
Redirect page to datastruct
Display all the obfuscated data
Display time needed to obfuscate
```

Testing of this algorithm was done on a dummy bank dataset. Further there is a scope to improve the design and methods of obfuscation. Also testing can be done for real databases to test accuracy.

## V. CONCLUSION AND FUTURE SCOPE

The model detects the tables in the database and the type of data each column contains. Based on the type of data and other distinguishing parameters like length it intelligently understands which method to use for obfuscation. Changes in the existing tables are also obfuscated. If a new table has been created, then too it automatically detects it and performs obfuscation.

There is a scope to test and modify this model in various domains to make it generic. Even this model can be extended to support image obfuscation which will be crucial in hiding the identity of the people/clients. Aim will be develop image obfuscation techniques which can minimize the infringement of user sensibility and simultaneously protect private life. Therefore, it is necessary to develop image distortion techniques to minimize the infringement of user sensibility and protect their privacy.

## REFERENCES

[1] "Data scrambling issues," http://www.datamasker.com/datascramblingissues.pdf, accessed 23 February 2016.

[2] "Data sanitization techniques," http://www.datamasker.com/datasanitization whitepaper.pdf, accessed 23 February 2016.Fröhlich, B. and Plate, J. 2000. The cubic mouse: a new device for three-dimensional input. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems

[3] "Data masking: What you need to know," http://www.datamasker.com/DataMasking WhatYouNeedToKnow.pdf, accessed 23 February 2016.

[4] W. Stallings, Cryptography and Network Security Principles and Practices, 4th ed. Pearson, 2009, pp. 35.

[5] T. Ritter, "Substitution cipher with pseudo-random shuffling: The dynamic substitution combiner," Cryptologia, vol. 14, no. 4, pp. 289–303, 1990.

[6] K. Muralidhar and R. Sarathy, "Data shuffling-a new masking approach for numerical data," Management Science, vol. 52, no. 5, pp. 658–670, 2006

[7] "Properties of random numbers," http://kevscode.com/csnotes/utpa/6337meng/note/master/node37.ht ml, accessed 15 September 2014.

[8] Saurabh Kulkarni, Siddhaling Urolagin,"Review of attacks on databases and database security techniques", International Journal of Emerging Technology and Advanced Engineering,2012, Vol.2, Issue 11, pp. 253-263