ISSN No.: 2454- 2024 (online)

International Journal of Technical Research & Science

DATA MASKING & TECHNIQUES

Waqas Ahmed
E-Mail Id: waqaschahmed92@gmail.com
Riphah International University, Islamabad, Pakistan

Abstract-Database emerged as a big and important field in software development nowadays and use of database is increasing day by day. The use of database is increasing with the passage of time and the security & privacy issues of data in the database are also increasing so different techniques are emerging to tackle these issues. Data masking is the process of de-identifying (masking) specific data elements within data stores. The main reason for applying masking to a data field is to protect data that is classified as personal identifiable data, personal sensitive data or commercially sensitive data, however the data must remain usable. There are variety of situations in which users, developers or other personnel may not be allowed to view information that they are not authorized to see such as in development phase, training sessions, testing etc. Many methods are used for data masking such as substitution, shuffling, variance, encryption, nulling out, field masking etc. Data masking is a technique used to hide the original data from developers and end users in development process and training sessions.

Keywords: Information Security, Data Security, Data Privacy, Database Management System, Information Systems, Databases.

1. PROBLEM STATEMENT

Protection of user's sensitive data while hiding the original data from all stakeholders including developers, end users in development process and training sessions however the data must remain usable for the purpose of undertaking valid test cycles.

2. INTRODUCTION

Data masking refers to the process of changing certain data elements within a data store so that the structure remains similar while the information itself is changed to protect sensitive information. Reason of data masking is to hide the group information. Data masking ensures that sensitive customer information is unavailable beyond the permitted production environment. This is especially common when it comes to situations like user training and software testing. The purpose is to protect the actual data while having a functional substitute for occasions when the real data is not required. In data masking, the format of data remains the same; only the values are changed. The data may be altered in a number of ways, including encryption, character shuffling and character or word substitution. Masked information might be appropriated without any fear and it can moreover be imparted for analysis reason. Primary target or information masking is to breaking point the right to gain entrance of organizational information and a different essential reason for existing is that it gives the greatest handiness of information.

A key requirement for any data masking and obfuscation practice is that the data must remain meaningful at several levels. Firstly it must remain meaningful for the application logic. For example, if elements of addresses are to be obfuscated and city and suburbs are replaced with substitute cities or suburbs, then, if within the application there is a feature that validates postcode or post code lookup, that function must still be allowed to operate without error and operate as expected. There are a variety of environments in which users, developers or other personnel may not be allowed to view information that they are not authorized to see. For example, an employee in the human resources department may not be allowed to see the salary details of senior executives. In order to tackle this sort of requirement, access controls are usually placed on the application so that only users with a high enough clearance can view that information: otherwise they just see a blank field or one filled with a string of anonymous characters or they may not be presented with that field at all (so that, as far as the employee is concerned, that information might not even exist).There are many environments where it is necessary both to hide the data and to use that data for processing purposes. The most common example is in development of new or updated applications or in retro-fitting data masking to existing environments. Developers may not be allowed to see detailed personal information but at the same time they may need that data, or its equivalent, in order to test the application they are developing. The technology, or technique, used to enable this functionality is known as data masking. It is also sometimes referred to as data obfuscation, data de-sensitization or data de-identification but we shall use the term data masking. In this paper we shall discuss why data masking is important.

3. THE IMPORTANCE & IMPLEMENTING OF DATA MASKING

There are essentially three types of data you might want to mask. The first is data about individuals, which can be credit card numbers, social security numbers, health and medical information, address and postal details and so on. This is generally referred to as personally identifiable information (PII) or, in the case of healthcare, personal health information (PHI). The main reason for masking such data is to comply with data privacy regulations but also to avoid the sort of bad publicity and costs associated with data breaches by hacking and other cyber-attacks. One complication is that different jurisdictions have different data privacy regulations. Not

only does this mean that you may need to apply different approaches to masking in different countries but also there are cross-border considerations, as some countries have strict controls about securing data that is moved outside the country. This may mean two levels of masking: one for internal use and one for overseas use. This may have particular significance where software development is geographically dispersed (for example, a joint development between an in-house team and an outsourcer).

While you must obfuscate private data in order to comply with regulatory requirements there are two other classes of data that you may choose to mask. The most obvious of these is financial data. You would not want, for example, your financial results to be leaked to the market (or to individuals) prior to your official announcement of those figures. Similarly, you would not be happy for detailed financial information to be made available to competitors. Secondly, there is intellectual property. If you are Coca Cola, for example, you would not want the formula for Coke to become public knowledge or have it leaked to your competition.

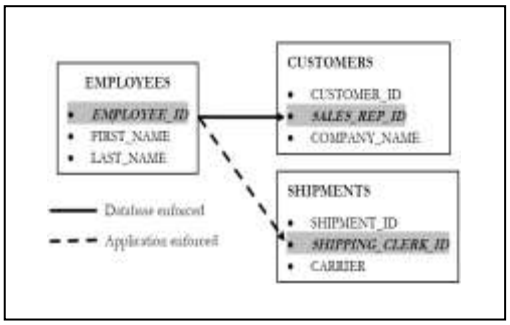
While the need to mask these three types of data is more or less obvious there is one further type of masking that may be required. Suppose that you have a fully masked database on a laptop and this gets stolen or left behind in a taxi. Then it is still possible to determine trends within the data, which may have commercial value. Special techniques are required to hide this sort of information defined. Do not use abbreviations in the title or heads unless they are unavoidable.

3.1 Oracle Masking Techniques

Oracle Data Masking Pack provides a variety of sophisticated masking techniques to meet application requirements while ensuring data privacy. These techniques ensure that applications continue to operate without errors after masking. For example,

3.2 Condition-Based Masking

This technique makes it possible to apply different mask formats to the same data set depending on the rows that match the conditions. For example, applying different national identifier masks based on country of origin.



3.3 Compound Masking

This technique ensures that a set of related columns is masked as a group to ensure that the masked data across the related columns retain the same relationship, e.g. city, state, zip values need to be consistent after masking.

3.4 Deterministic Masking

This technique ensures repeatable masked values after a mask run. Enterprise may use this technique to ensure that certain values, e.g. a customer number gets masked to the same value across all databases.

3.5 Manual Masking

It is possible, at least in theory, to do your masking manually. We do not advocate such an approach. Ultimately it will cost more money to take a manual approach and it involves significantly more risk of data privacy breaches. Indeed, where there is any degree of complexity involved then we do not believe that this approach is practical: it is time consuming, error prone and all too likely to be incomplete. This will become apparent as we discuss the requirements for data masking products in the following section.

3.6 Techniques

- Substitution
- Variance
- Shuffling
- Encryption
- Field Masking
- Cross Schema Masking
- Flat File Masking
- XML Masking
- Nulling out
- Row Internal Synchronization
- Table Internal Synchronization
- Table to Table Synchronization
- User Defined SQL Commands

3.7 Implementing

Data in the today's relational databases are too much efficient that it provides no duplication of data using some special columns called primary keys for example, an EMPLOYEE_ID generated from a human capital management (HCM) application may be used in sales force automation application using foreign key to keep a

track of their records. When applying data masking, users are mostly worried with referential integrity of the database. Some of the tools are there who automatically identifies the primary and foreign key relation in the database like if customer chooses the EMPLOYEE_ID to mask then it will find all related foreign keys in that database and enforces the mask to maintain these relations. This ensures that the relation between the tables is kept while private data is masked. These tools also apply the mask to foreign key columns.

Example: Employee-Data Masking

Based on Oracle Data Masking, Oracle has developed a comprehensive 4-step approach to implement data masking called Find, Assess, Secure, and Test (FAST). These steps are:

- **Find**: This phase involves identifying and cataloging sensitive or regulated data across the entire enterprise. Typically carried out by business or security analysts, the goal of this exercise is to come up with the comprehensive list of sensitive data elements specific to the organization and discover the associated tables and columns across enterprise databases that contain the sensitive data.
- **Assess**: In this phase, developers or DBAs in conjunction with business or security analysts identify the masking algorithms that represent the optimal techniques to replace the original sensitive data. Developers can leverage the existing masking library or extend it with their own masking routines.
- **Secure**: This and the next steps may be iterative. The security administrator executes the masking process to secure the sensitive data during masking trials. Once the masking process has completed and has been verified, the DBA then hands over the environment to the application testers.
- **Test**: In the final step, the production users execute application processes to test whether the resulting masked data can be turned over to the other non-production users. If the asking routines need to be tweaked further, the DBA restores the database to the pre-masked state, fixes the masking algorithms and re-executes the masking process.

4. PRODUCT REQUIREMENTS

4.1 Database Support

- Broad database support
- Support both distributed and mainframe data
- Support multiple database connections at once
- Support database-defined relational integrity – rely upon and use meta-data
- Maintain indexes, triggers, etc.
- Common interface regardless of database type
- Rely on database processes where appropriate
- Easily support database upgrades and changes

4.2 Application Support

- Ability to work with both common enterprise applications and custom applications
- Support application-defined relational integrity – should be easy to enforce
- Common interface regardless of application
- Rely on database processes where appropriate
- Easily support application upgrades and changes

4.3 Platform/System Support

- Broad platform support – open standards
- Avoid different versions for different platforms
- Support multiple database connections at once even if on different platforms
- Support relational integrity defined *across* databases/applications – synchronize within integrated environments
- Common interface regardless of platform
- Deployment Options: 1) Server Only; 2) Mixed Client-Server; and 3) Client Only
- Easily support platform/system upgrades and changes

4.4 Functional Requirements

- Enterprise-level solution
- Intuitive, easy to use GUI
- Same interface for all database, platform, and application types
- Repeatable process
- Pre-packaged masking methods
- Scripting ability
- Strong and secure masking methods
- Avoid manual mapping

Data masking does not simply involve the masking process itself. Before you mask the data you need to know what data needs to be masked and, once you have masked the data, you need to be able to audit your masking for compliance purposes. We will therefore discuss each of the relevant sub-tasks separately.

4.5 Discovery

Discovering what data needs to be masked isn't the simple task that it may appear to be. For example, suppose that you need to mask a social security number. This is a nine digit number in the form xxx-xx-xxxx. You have to cater for the fact that there may not be any dashes in the database where it is stored and it may just appear as a string of nine numbers or have other characters in place of dashes. The latter is likely to be a data quality issue (there is a minus sign instead of a dash, for example) but your discovery process needs to cater for that fact. You also have to allow for the possibility that the social security number may have been inadvertently incorporated into somebody's address, which means that you need to search the entire database for relevant entries not just where you expect the social security number to reside. Moreover, if the number could be just nine digits located anywhere in the database then you will get an awful lot of false positives.

A nine digit number could easily be a product code, or financial data, for example. A further consideration comes into play when you want to mask financial data because now you will typically want to apply a threshold value to what should and should not be masked. However, much the biggest issue when it comes to discovery applies to related information that needs to be masked. How important this is will depend on the environment. As an example, suppose that you are masking names and addresses. If the application you are developing requires that the address matches the post code (for instance, if you live in Nevada then you must have a valid Nevada zip code) then you must mask Nevada and the zip code in such a way that that relationship remains intact after the masking process is completed. In other words, you must understand and recognize that this relationship exists as a part of the discovery process. Conversely, if the application could care less about the relationship between the state and zip code then this will not be a factor.

Discovering relationships is not as trivial as this example might suggest. In theory, all relationships within a relational database are defined within the database schema. This obviously does not apply to databases such as No SQL databases that don't use or have only a very lightweight schema. However, even relational database schemas do not capture all the data relationships that exist within the database. This is because relationships are often implemented within application software and not within the schema itself. Such relationships can only be determined either via a detailed knowledge of all relevant applications (which is more or less impossible) or by inferring such relationships through an examination of the database content. Data discovery and profiling tools can achieve this easily but it is impractical to attempt to do so manually. The requirement to understand the relationships that exist across data elements is further complicated if your application touches multiple data sources, and especially if these sources are heterogeneous. This applies to both identities and more removed relationships. That is, you may have customer details in multiple databases.

To begin the process of masking data, the data elements that need to be masked in the application must be identified. The first step that any organization must take is to determine what is sensitive. This is because sensitive data is related to specific to the government regulations and industry standards that govern how the data can used or shared. Thus, the first step is for the security administrator to publish what constitutes sensitive data and get agreement from the company's compliance or risk officers.

To begin the process of masking data, the data elements that need to be masked in the application must be identified. The first step that any organization must take is to determine what is sensitive. This is because sensitive data is related to specific to the government regulations and industry standards that govern how the data can used or shared. Thus, the first step is for the security administrator to publish what constitutes sensitive data and get agreement from the company's compliance or risk officers.

Table-4.1 List of Sensitive Data Elements

Person Name	Bank Account Number
Maiden Name	Card Number (Credit or Debit Card Number)
Business Address	Tax Registration Number or National Tax ID
Business Telephone Number	Person Identification Number
Business Email Address	Welfare Pension Insurance Number
Custom Name	Unemployment Insurance Number
Employee Number	Government Affiliation ID
User Global Identifier	Military Service ID
Party Number or Customer Number	Social Insurance Number
Account Name	Pension ID Number
Mail Stop	Article Number
GPS Location	Civil Identifier Number
Student Exam Hall Ticket Number	Credit Card Number
Club Membership ID	Social Security Number
Library Card Number	Trade Union Membership Number

4.6 Masking

There are multiple ways in which masking can be achieved. This will depend, at least in part, on why you are doing the masking. For example, you could simply hide a credit card number by replacing each digit with an x (xxxxxxxx-xxxx), which will be fine if you are only concerned with data protection. However, if you want to test

a payment application then you will need to work with real (pseudo-) numbers in order to test your applications. Similarly, you can redact data, which is where the data is completely obscured, a typical example being where data is censored. This technique applies usually to text and it is not suitable for testing purposes since the data cannot be read. Randomization is another technique,

Whereby you simply replace a postal code (say) with random letters and numbers (as appropriate). However, this technique will not work if your application requires a valid zip code. To support application development and testing you will need to mask in such a way that the data remains valid. Perhaps the most common technique used is shuffling. As an example of how this works suppose that you have Adam Jones, Richard Hills and John Smith in your database then shuffling might result in masked records for Adam Hills, Richard Smith and John Jones: in other words you shuffle all the first names and all the last names as two separate processes. Its advantages are that you ensure realistic data and the chances of reconstitution are very small provided the initial dataset is large enough (say, 5,000+ records). For smaller datasets you will need to use some other form of masking. It should be borne in mind that masking is never perfect. In healthcare environments, for example, a determined hacker may still be able to identify individuals, precisely because of the need to retain relationships within the masked data. In addition, and as another example, your largest customer will still be your largest customer even if he, she or it is not immediately identifiable by name. Even if the whole database is masked there may still be commercial value that can be derived from trend information buried within the data, unless you have special facilities to cater for this.

Similarly, if the data is naturally skewed - for example, more people live in California than North Dakota - then this may be reflected in the masked data also, depending on how the masking is done: if states are always mapped in a consistent manner then the masked data will be similarly skewed but if you mask randomly then the skew will go away.

Data masking products will ideally be delivered with multiple seed tables, with internationalized versions of these tables where appropriate (such as names) and you should be able to add your own seed tables and customize those that are provided out-of-the-box. You will also want multi-column capability within these seed tables so that, for example, state and zip code will match. You will also require cross reference management so that you can, say, retain the same transformations across runs or databases.

File-based masking is often required in conjunction with database masking. For example, you might have scrambled the social security numbers in the target database. However, your input file could now contain non-matching social security numbers and the load will fail. Cross referencing capabilities will therefore be necessary or hash routines that you can employee as a part of the masking process so that you can ensure that this mismatch doesn't happen.

It is also worth noting that there is a distinction between static data masking, which is used in test and development environments and which we have largely been describing, and dynamic data masking. The latter is used in conjunction with operational data in real-time. As understood by most of the market this involves dynamically intercepting SQL requests to the database and masking any sensitive data. This is complementary to, or a replacement of, the sort of role-based access controls that are in use in many environments to prevent, for example, HR employees seeing the salary levels of executives. Finally, we should note that, for testing and other non-production environments, there is an alternative to data masking, which is to use synthetic data generation. In this case, once you have discovered and profiled the source data you generate (based on seed tables and parameters that you set) a purely synthetic dataset. As this contains no 'real' information it does not need to be masked.

4.7 Audit

It is important for data governance and compliance reasons that you can prove that you have appropriate processes in place to ensure the integrity of personally identifiable information and that you have actually masked what you have said you have masked. Also you will want to report on who chose what needs to be masked, why it needed to be masked and when. Audit reports should also provide details about what technology was used to provide the masking. A further additional facility that will be useful is to be able to run a comparison of the data before and after masking.

Workflow capabilities will also be useful, which allow you to define relevant procedures with, for example, checked, validated and approved stages to the identification of which data is to be masked. This lets relevant people look at the data profiling information and confirm whether the data has correctly been identified or not.

4.8 Platform

You should be able to mask across multiple heterogeneous databases and platforms such as Oracle, SQL Server, DB2 and so on. In addition, depending on your environment, you may need the ability to include legacy systems such as IMS and VSAM within your masking environment. On that topic it will also be useful if your data masking will run locally on your mainframe: using processes or ODBC based connectivity will significantly impair performance. In terms of operating systems support you would expect Windows, Linux and UNIX support as well as the ability to run in mainframe environments: either in a z/Linux partition, or directly, or both. You would want to be able to support standalone deployment, or deployment on a grid/cluster. Where relevant, you would want each geographic location to have its own deployment and to be able to synchronize metadata between

locations for distributed deployment. Cloud-based deployment (probably a private cloud) may also be advantageous.

5. BEST PRACTICE FOR DATA MASKING

There are some practices which are chosen as best for data masking. They are

- Enterprise solution
- Built-in data masking methods
- Easy to use / learn
- Content
- Data format validation
- Data consistency
- Relational integrity
- Policy simulation
- Application awareness

6. COMPARISON WITH OTHER SECURITY TECHNIQUES

Conventionally there were some techniques available that was used for security measure like encryption. Encryption protected the data while at rest but the problem was that the encrypted data can't be used for purpose of training, developing, etc. The data to be used for this purpose needed to be decrypted otherwise it's impossible to use it and decryption also removes back the security. Other conventional technique is using firewall and password but it only protects from external threats, this not fulfills our requirements because the organizational data need to be protected from both internal and external threats.

Masked data is unlike encrypted data, it maintains its usability as same as original data. It increases the protection again data theft but if somehow the data is stolen, it will be totally useless because of the masked values of attributes. It produces restriction on the use of data. Most important feature is that it gives reasonable data for development, testing, training, out sourcing, data mining, etc.

6.1 Benefits

- Increases protection against data theft
- Reduces restrictions on data use
- Provides realistic data for testing, development, training, outsourcing, data mining/research, etc.
- Enables off-site and cross-border software development and data sharing
- Supports compliance with privacy legislation & policies
- Data masking demonstrates corporate due diligence regarding compliance with data privacy legislation
- Improves client confidence
- Provides a heightened sense of security to clients, employees, and suppliers

6.2 Complications of Data Masking

- Data Utility - masked data must look and act like the real data
  - proper testing and development
  - application edits
  - data validations
- Data Relationships - must be maintained after masking
  - database level RI
  - application level RI
  - data synchronization (interrelated database RI)
  - Existing Business Processes - needs to fit in with existing processes
  - fit in with existing IT and refresh processes
  - automation of masking process
- Ease of Use - must balance ease of use with need to intelligently mask data
  - need to have usable data that does not release sensitive information
  - knowledge of specialized IT/privacy topics and algorithmic should be pre-configured and built into masking process
- Customizable - must be able to be tailored to specific needs
  - any solution/process must have the ability to be easily updated and customized
  - must have ability for masking methods and the overall solution to be customized

7. ACRONYMS

- PII Personally Identifiable Information
- PHI Personal Health Information
- ODBC Open Database Connectivity

- SQL Structured Query Language
- GUI Graphical User Interface

ACKNOWLEDGMENT

I am highly indebted to my teacher's unfailing encouragement and step to step guidance during the research period and project work. Without their help it would have been impossible to complete this task.

BIBLIOGRAPHY

[1] https://eprint.iacr.org/2010/441.pdf
[2] https://google.com/patents/US6034916
[3] https://ieeexplore.ieee.org/document/5389557/
[4] https://link.springer.com/article/10.1007/s11265-005-4153-1
[5] https://uknowledge.uky.edu/management_patents/1/
[6] http://searchsecurity.techtarget.com/definition/data-masking
[7] http://www.oracle.com/us/products/database/data-masking-best-practices-161213.pdf
[8] http://www.oracle.com/technetwork/database/security/ds-datamasking-130350.pdf
[9] https://www.academia.edu/.../A_Data_Masking_Technique_for_Data_Warehouses
[10] http://www.gridtools.com/download/Spotlight_Data_Masking.pdf
[11] http://www.oracle.com/us/products/database/data-masking/overview/index.html
[12] http://www.itpsc.org/resources/Documents/IBM%20Optim%20-%20Test%20Data%20Mgmt%200409.pdf
[13] http://nyoug.org/Presentations/2008/Donatone_Data%20Masking.pdf
[14] http://www.grid-tools.com/download/Spotlight_Data_Masking.pdf
[15] https://pubsonline.informs.org/doi/abs/10.1287/mnsc.1050.0503
[16] https://docs.oracle.com/cd/E11882_01/server.112/e41481/tdm_data_masking.htm