TO MASK OR NOT TO MASK: WHY WE DON’T MASK OUR DATA*

*But why it’s critical that we do

By Pete Finnigan
ABOUT THE DELPHIX EXPERTS’ SERIES

The Delphix Experts’ Series explores the technology issues that technical and business decision makers face when they’re evaluating and deploying enterprise software solutions. Our experts are subject matter authorities on a wide range of topics, including data security, regulatory and legal issues, implementation, and best practices. Papers in this series reflect their unique perspectives.

ABOUT THE AUTHOR

Pete Finnigan is widely regarded as an expert on the subject of securing data in Oracle databases. He helped to set standards now used across industries to harden and lock down Oracle databases. He created the SANS Oracle security step-by-step guide that was the original source for the Center for Internet Security standard for Oracle databases. Pete speaks regularly at conferences worldwide and is the author of a number of books on Oracle Security.

He runs his company, PeteFinnigan.com Limited, which provides training on all aspects of securing Oracle, Security health checks, consultancy and design and also sells two software products; PFCLScan to perform your own database security health check and PFCLObfuscate to protect the IPR in your deployed PL/SQL.
# TABLE OF CONTENTS

Understanding Today’s Landscape ............................................................... 1
Why Do People Not Mask Their Data? .......................................................... 2
Protecting Data on Non-Production Systems ................................................. 3
Implementing a Successful Data Masking Project .......................................... 4
Planning Your Masking ................................................................................. 5
Using Delphix ............................................................................................... 5
Handling Complexity .................................................................................... 10
Keeping Clear Data Clear ............................................................................. 11
Wrapping Up ................................................................................................. 11
Appendix—How I Wrote this Paper ............................................................... 12
UNDERSTANDING TODAY’S LANDSCAPE

I make my living assessing and measuring security risks in Oracle databases. This usually involves performing a detailed audit of a main production database and tracking the flow of that data to non-production databases. This audit may also uncover sensitive data that has been replicated from production to non-production databases.

As part of my security auditing work I constantly find data being copied to non-production databases. In almost every case, this data isn't protected or altered in any way, making it easier to steal as clear text from such non-production sources.

This paper delves into the reasons why customers don’t mask their critical data, and what steps you can take to make sure you mask any data leaving your production silo.

DEFINING DATA MASKING

Data masking is the process used to change the character of data to protect any information that may be personally identifiable, personally sensitive, or commercially sensitive.

While there are many data touch points throughout an organization that can benefit from data masking, it plays an especially relevant role in application development and testing environments as IT organizations in nearly all industries are under pressure to deliver new applications faster.

– Delphix
WHY DO PEOPLE NOT MASK THEIR DATA?

In discussions with customers about their data replication I always find recurring reasons as to why do people not mask? These could be thought of as the top four reasons, but I have no scientific basis for these being the top four or indeed the only reasons why people do not mask; they are just the reasons I come across many times. These reasons can be summarized as:

1. **Locate all the data:** Customers tell me that even if they throw a lot of budget at this, that budget is wasted if they can’t actually validate that they’ve found all data.

2. **Referential integrity:** Customers tell me they’re concerned about breaking referential integrity with data masking. Ensuring referential integrity with masking requires locating and consistently masking all data of the same value to maintain referential integrity such that column values continue to match each other after masking. Sometimes the database enforces foreign key to primary key joins and in these cases we can query the database to find the referential integrity constraints. At other times an application handles the intended table joins without enforcing the referential integrity in the database making it more difficult to find fields required for referential integrity and this increased difficulty increases the risk that all the referentially correlated data won’t be located.

3. **Validate testing accuracy:** Customers often believe that testing masked data isn’t valid since masked data is simply not the same as real data. It can be difficult to ensure that masked data retains the distribution of the unmasked data. Masking data to random data values can change the meaning of the data with respect to other application features, confuse testers, and skew the results of any testing. Customers have also shared with me the concern that summary detail reports may not be valid, particularly for financial data in terms of scale, distribution, volumes, and more.

4. **Secure adequate budget and resources:** It may be necessary to employ specialist staff, either on a permanent or contract basis to implement the masking. Customers often tell me they can’t find internal resources with the necessary skills and that licensed products are too expensive. They also express frustration with having to implement new development and processes or programs simply to extract and mask data, and update test and development systems.
TRACKING THE FLOW OF UNMASKED DATA

Rather than implementing a masking solution, customers more often than not will note the security risk in a risk register for management to sign off. The risk for them to mask is seen as greater than the risk of not masking. This should always be an invalid assumption as data loss is worse than disgruntled testers.

Unfortunately unmasked data can quickly spread around an organization—not only in non-production databases but in printed test reports, binary reports, on users screens, backups, and more.

In the rare case that some masking is done then it’s for very specific data sets that may be affected by regulatory requirements\(^1\). Customers often use simple home grown tools or scripts—the focus being to comply with regulations, not to protect all data of the same type.

But regulatory drivers aren’t the only reason you should think about masking data. Maintaining business integrity and thwarting potential espionage are also pretty good reasons to protect company-specific data, including financials, customer lists, supplier lists, product lists—any data that could have value to criminals or competitors if stolen.

PROTECTING DATA ON NON-PRODUCTION SYSTEMS

WHAT’S THE RISK OF COPYING DATA TO NON-PRODUCTION SYSTEMS WITHIN THE ORGANIZATION, OR—EVEN WORSE—TO EXTERNAL SUPPLIERS OR VENDORS TO TEST THEIR SOFTWARE BEFORE DELIVERING IT TO YOU?

The answer to this should be obvious. You may be liable for a set fine per record lost if personal details are stolen from you. Even worse: media backlash and loss of confidence from your customers and partners.

Data loss can spell the end for a company both financially and even personally for directors.

WHAT SOLUTIONS EXIST TO COMBAT DATA LOSS?

The solution to securing data is multi-faceted. It should include properly planned, structured and layered data security across both production databases and all non-production databases. Data masking should be part of this data security solution. The data masking solution you select should ideally be “point and click,” without the need for specialist knowledge, and it should be able to (a) find all the data that should be masked, (b) locate all joins between database tables, and (c) allow retention of data distributions as necessary for testing.

---

\(^1\) The US mandates specific protections for sensitive data including HIPAA and SOX. In the EU data protection laws in specific countries such as the UK mandate usage and dissemination of personal data, and PCI regulations dictate that elements of credit card details must be stored and used in a secure way.
IMPLEMENTING A SUCCESSFUL DATA MASKING PROJECT

Based on where the data is changed, methods of data masking include:

- **“In-place” masking** - extracts and changes clear text data and then replaces back into the original source database. This is the usual method used by simple retail products.

- **“Extract-and-load” masking** - manipulates data as it’s loaded.

- **“On-the-fly” masking** - a subset of “extract-and-load” that may not involve traditional SQL based solutions. The data never leaves the source in clear text and gets masked in real time.

The masked data still needs to work like the original data, and a number of algorithms exist to convert or change the data. These range from a simple mapping of values on a first-come, first-served basis from a list or conversion to completely random strings or secure lookups where an original value is hashed and then “looked up” in a list. This method enables the same source value to be mapped to another value but the mapping is not predictable or reversible. It’s also possible to create algorithms that reverse the original value from the masked value. This is limited in security if the reverse procedure is available in the masked database.

Which are the best algorithms? In my experience any algorithm that maps joins between tables based on database constraints or simply matches values to be masked based on their current value is the best. If that algorithm also is not predictable and cannot be reversed that’s ideal.

What’s the best way to ensure a successful data masking project? You want a solution that needs minimal intervention to properly mask data and that doesn’t require specialist masking skills. Data masking is a complex process and when you toss requirements such as referential integrity and data distribution into the mix it quickly becomes apparent that finding a completely automated tool that “simply masks” all data beyond recognition is an impossible task. So, a more realistic goal is a solution where you perform limited tasks—ideally point and click based tasks—that help you achieve fully masked, usable data in all test and development and downstream databases from production. This is possible without skilled masking knowledge and intervention. This is what Delphix does.
PLANNING YOUR MASKING

Before starting a masking project with Delphix it’s important to plan in advance what you want to achieve. It’s also useful to have some pre-knowledge of your schemas and data. Delphix enables you to specify the connection details of a user to the database and also the schema to be masked, so if you need to mask multiple schemas you should identify these in advance. A simple way to find schemas is to look for all database users that have tables and/or views.

Does the database use indexes for primary keys and foreign keys? You can check this easily in the database meta-data and index and constraint views.

What do you want to achieve with data values after masking? Do you need to retain relative distributions of data? If yes, then use secure lookups to map like data for like data and retain distributions. If you want to retain regional distribution—so that someone who lives in York still lives in York after masking but his name has changed, his post code still starts with YO and his phone number starts 01904—you can do it using Delphix and some configuration.

You should also decide when to mask. Should you provision a new database with Delphix and then mask in place? Or should you use on-the-fly masking where the data is masked as the database is provisioned? You could also mask in a provisioned VDB (a virtual provisioned database—or a staging database) using in-place masking and then provision new non-production databases from that staging VDB.

USING DELPHIX

Setting up Delphix is easy to do:

1. Set up the target database as an environment
2. Add the connection details for this database by creating a connection user and password
3. Identify the schema that should be used as the masking target

Once you’ve set up Delphix, you’re ready to start masking. If you need to mask data in more than one schema, create multiple connections—one for each schema that needs to be masked.
Delphix masking works in two layers.

The first layer is discovery. This is where we specify a rule set to decide which database tables to mask. This discovery level is called profiling in Delphix. The **Delphix Data Profiler** also has two elements; these are the expressions used to locate data in your database (more on this in a moment) and the target of those expressions. There are also two targets. **Delphix Masking** can check meta-data (database column names) to see if it’s likely to hold the data you’re looking for based on naming, and it can also sample data in the database itself. Delphix does this by checking all possible tables and columns in the rule set that suit the possible storage types and sampling a set number of rows of data from each column, and then applying the expression to each row.

You can create your own expressions to profile data and choose whether the expression targets meta-data (columns) or samples actual data.

After you’ve added your table names to your rule set you can create a profile job. This job runs the expressions in the Data Profiler against the tables defined for your schema in the rule set. Delphix ships with two sets of profile expressions—Profile Sets—for Healthcare and Finance. When you create the profiler job don’t choose any of these options to get all expressions.
The second layer of Delphix is masking. After profiling, you’ll see an inventory of possible columns that can be masked, identified DOMAINS for each, as well as an identified masking algorithm. This is a great start to the masking process, and as you’ve seen doesn’t require any deep masking knowledge. The profile results are visible in the inventory:

Delphix displays profile results in the inventory.

Delphix was very easy to use to get a first pass at identifying data that can and should be masked. There were false positives; this is data that’s identified to be masked but should not be. In my database all columns of my blog posts table were identified as ZIP codes but this was because the expression I used looked for columns that include the word POST. All columns in this table included the word POST. Delphix identified some columns to be masked but the algorithms weren’t suitable. For instance in my database my addresses table has a column called CITY and it was identified to be masked to a NULL string. This is not suitable for me since CITY data is a mandatory column. Because I didn’t customize the profiler expressions in advance of running it (which I would do for an actual implementation), some columns were missed, such as the address first line and the address second line.

Expect to take some time to review the inventory of profiled data. This is a first pass to help you identify masking and should not be simply used as-is. Delphix is good but not completely automatic. You don’t need to profile well as you can visit every column and review its choices and edit the inventory.
Delphix rules identify a DOMAIN—an identifier if you will—and also an algorithm to use to mask the data. Some of the available algorithms are discussed here:

- **Secure lookup**
  - Encrypt/hash/modulus algorithm that's repeatable but not reversible.
  - Lets you assign a realistic pre-defined value from a list. Use secure lookup when unique values are not needed.
  - Secure lookup maps the same value to the same value across the database, i.e. Pete => Fred everywhere, or Finnigan=>somesurname. Referential integrity is retained.

- **Segmented mapping algorithm**
  - Useful for creating unique mapping of formatted data. Each segment is masked individually.
  - You can preserve a semantically rich part of a segment and mask the other parts of the data.
  - When using these for primary=>foreign key pairs both sides must use the same segmented algorithm of course.
  - These are very powerful as parts of a number can be changed and parts retained. It's important not to generate real values in parts of the segment.

- **Mapping algorithm**
  - Sequentially maps values to a pre-populated table made from a supplied file

- **Binary lookup**
  - Similar to secure lookup algorithm but used to binary data
At this point you need to review all columns for each table in terms of the domains and algorithms identified or needing to be applied. You must edit the column to add a masking rule in the inventory, and if you need to create a new rule then it’s easy to create a new domain and algorithm. For instance for city masking I created a new secure lookup algorithm and a new domain for my CITY.

1. I created a lookup file for UK cities:

<table>
<thead>
<tr>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>Derby</td>
</tr>
<tr>
<td>Manchester</td>
</tr>
<tr>
<td>Birmingham</td>
</tr>
<tr>
<td>Brighton</td>
</tr>
<tr>
<td>Burton</td>
</tr>
<tr>
<td>Plymouth</td>
</tr>
<tr>
<td>Leicester</td>
</tr>
<tr>
<td>Dundee</td>
</tr>
<tr>
<td>York</td>
</tr>
<tr>
<td>Hull</td>
</tr>
<tr>
<td>Chester</td>
</tr>
<tr>
<td>Lancaster</td>
</tr>
<tr>
<td>Bolton</td>
</tr>
<tr>
<td>Wrexham</td>
</tr>
<tr>
<td>Scunthorpe</td>
</tr>
<tr>
<td>Scunthorpe</td>
</tr>
<tr>
<td>Droitwich</td>
</tr>
<tr>
<td>Swindon</td>
</tr>
<tr>
<td>Bedford</td>
</tr>
<tr>
<td>Bridgend</td>
</tr>
<tr>
<td>Bury St Edmunds</td>
</tr>
<tr>
<td>Guildford</td>
</tr>
<tr>
<td>Gosport</td>
</tr>
<tr>
<td>Fareham</td>
</tr>
<tr>
<td>Fareham</td>
</tr>
<tr>
<td>Netley</td>
</tr>
</tbody>
</table>

2. I attached that to a new secure lookup algorithm:

3. Then I added the algorithm to the new domain:
This process should be repeated for each table and column to ensure that the masking rules you chose are what you need—but no masking skill is needed to do this. I completed this process in about one hour for 25 database tables and, I created a number of my own algorithms and domains easily.

The final step is to create a masking job and run it to mask all the identified data in your database. I chose suitable settings for memory and number of streams to match my limited hardware but you can ramp these up for performance on industry level hardware. The masking process is completed.

**HANDLING COMPLEXITY**

In some cases we may need to deal with complexity. A good example is encrypted data. If credit cards are encrypted in a database then they would not be discovered for masking or could be incorrectly discovered by a column-based expression during profiling that identifies, YES it’s a credit card, but NO it cannot be masked as it’s actually encrypted data.

If you need to mask encrypted data, you can use the pre and post options in the masking job wizard. In this example we would decrypt the masked data in the pre-scripts, the masking rule would mask the data, and then we would re-encrypt the data in the post-script rule.

Delphix masking also enables you to define users and levels of access with 5 built in roles:

- **Analyst** - can profile and update the inventory but cannot add targets or connection details
- **Developer** - can create masking jobs and view runtime reports
- **Operator** - can execute jobs only
- **Application owner** - can define connections
- **Administrator** - can do anything
KEEPING CLEAR DATA CLEAR

New in version 5 of Delphix is the ability to connect two Delphix engines, one in production (used as the source) and one in non-production. The new virtual database (VDB) is created in production, masked in the production zone and then replicated to the Delphix engine in the non-production zone, ensuring that clear text data never leaves the production silo.

Delphix also supports certification jobs so that a masked database can be reported against using the masking rules defined for that database to ensure that no new data is added to the target that would break the masking rules thus ensuring that a masked and protected non-production database does not become un-masked.

WRAPPING UP

As you can see using Delphix is actually easy compared to hand writing code to mask data. If you choose secure lookups you can retain the distribution of data. Here's a simple example of before and after for my sample database and application:

In this paper, we’ve discussed masking for Oracle, but Delphix supports many other platforms in the same way, including SQL Server, Postgres, and even unstructured data like text files or Microsoft Excel.

Delphix is a clear winner and goes a long way to alleviate people’s fears of not masking their own data.

It's simple and point-and-click easy. Just set it up once, and provision and mask many times to every non-production database from one source. Conversely it’s not just a point and click tool as it provides easy to use functionality to create your own profiling expressions, masking algorithms, and domains. Delphix masking is simple if you don’t have masking skills, but it’s also extremely powerful if you need it to be, masking data to the same secure values across databases, web servers, and even PC-based files.

You can maintain referential integrity; You can easily locate all data. And you can easily mask that data to useable values by using either the built in algorithms and shipped data sets or creating your own.
APPENDIX—HOW I WROTE THIS PAPER

This appendix provides some additional background information detailing how I tested and used Delphix and Masking and how I wrote this paper. But there’s no need to read this appendix to appreciate and enjoy the main paper!

When Delphix asked me to write this paper I didn’t want to write about the Delphix product without actually using it for myself. The simple choice in terms of testing is to download Delphix Express and the two Landshark Virtual Machines (You can download from https://download.delphix.com/). These use pre-created images you can add to your virtualization software such as VMware or Oracle Virtual Box and start testing almost immediately. But to get a more hands on feel for Delphix, I decided to dispense with simplicity and take on real-world complexity by manually setting up Delphix and creating my own source database and target database.

I used Delphix Express version 4.3.3 with masking specially enabled for me so that Delphix and Masking could be tested. I installed the Delphix Express VM into Virtual Box and proceeded to create a new Oracle 11.2.0.3 database to act as the source database, with a new Oracle 11.2.0.3 software installation to act as the target. My source database supports two PHP based applications, both web-based and both running from separate webservers/application servers. One application supports the public facing website for a mythical company and the other supports the internal application used to manage customers, products, suppliers, sales, and more by the mythical company’s staff. I use these applications and sample data in my Oracle security training classes.

The next step was to set up the source and target databases as described in the Delphix documentation. This involves creating Unix users, nfs mounts, database users, and some additional settings. Then I added the source database and target database to Delphix using the Delphix interface. Once completed, I could provision the source database. I provisioned and refreshed many times and also took snapshots of the source database and built more than one new target database. Next I locked down my source database to a reasonably good level. I then re-created the target database again to test that the locked database replicated.

Then I was ready to take a good look at the masking product. My first test was to simply use the masking features out of the box without really changing anything or making many choices. This was followed by much more detailed tests of the complete product where I created new algorithms, new domains, new searches, and where I also edited the inventory after profiling.

The purpose of this very simple appendix was to show that I was keen to use the Delphix product myself in some depth so I could better understand how it works.
To Mask or Not to Mask: Why We Don’t Mask Our Data
March 2016

You can find the most up-to-date technical documentation at:
http://www.delphix.com/support

The Delphix Website also provides the latest product updates.
If you have comments about this documentation, submit your feedback to:
help@delphix.com

Delphix Corp.
275 Middlefield Road, Suite 210
Menlo Park, CA 94025
www.delphix.com

© 2016 Delphix Corp. All rights reserved.
The Delphix logo and design are registered trademarks of Delphix Corp.
in the United States and/or other jurisdictions.
All other marks and names mentioned herein may be trademarks of
their respective companies.