

Dynamic Data Masking Introduction

Overview and implementation best practice

Introduction to Dynamic Data Masking

Dynamic Data Masking is the process of obscuring (masking) specific data elements on-the-fly without touching applications or physical production data stores (databases). It ensures that sensitive and personal identifiable information (PII) accessed from application screens, packaged reports, development and DBA tools is replaced with credible but not real data to selective end-users, IT support teams, part-time or outsourced personnel who do not require it to perform their job.

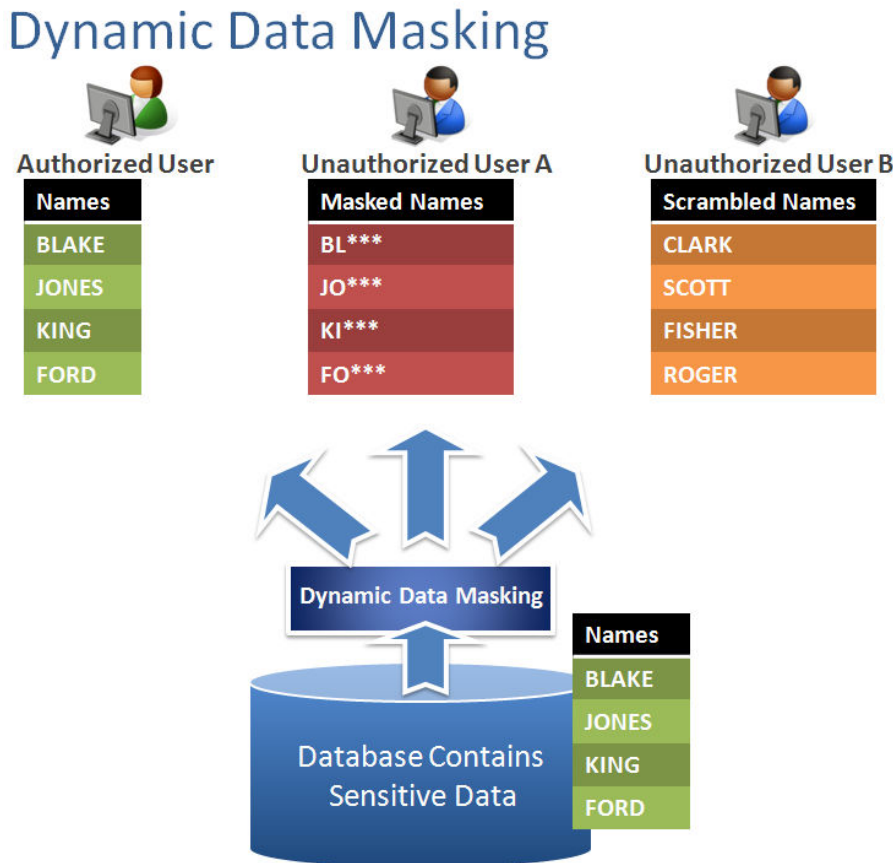


Figure 1: Dynamic Data Masking masks PII to unauthorized users, while maintaining authorized user productivity

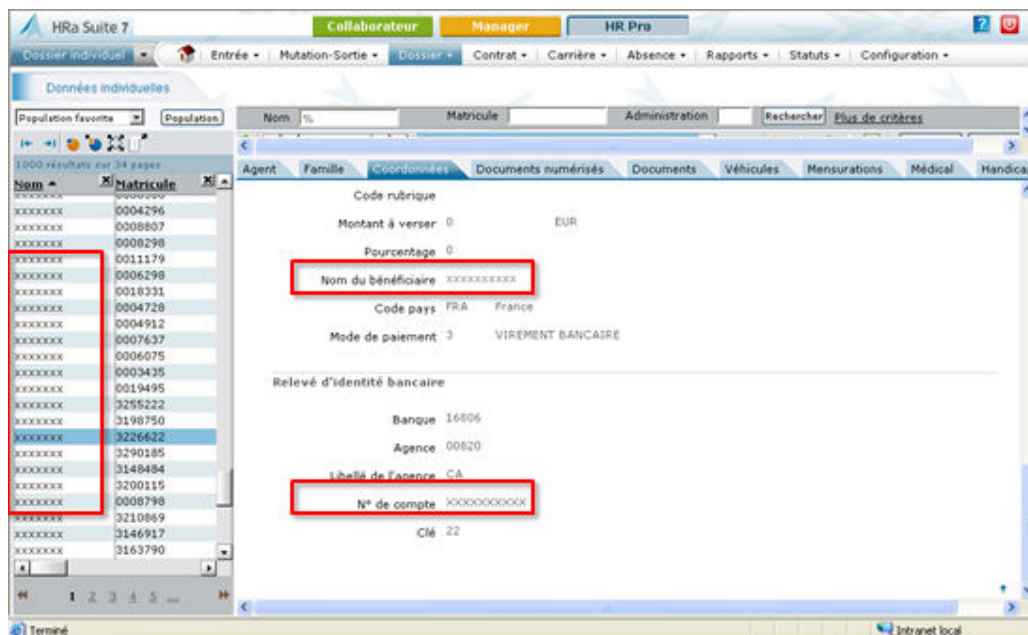
A New Paradigm

Traditional data masking solutions perform Static Data Masking, where the obfuscated values are physically stored in the database. For obvious reasons, Data Masking has been limited to Development and QA environments. Now, for the first time, organizations' production environments can be DYNAMICALLY masked without enhancing existing applications and building new security layers to comply with ever increasing regulatory requirements to protect PII. Moreover, building new application security is not possible in most packaged and proprietary applications, reporting and DBA tools.

The only way remaining for organizations to protect PII is to enhance the existing application security with Dynamic Data Masking security layer, which enables the application of various security actions, including masking, hiding or blocking incoming requests.

These actions are selectively applied according to specific application requests, modules, on-line screens, batches, transactions or end-users, applying additional contextual access restrictions (e.g., row-level, column or object level) without touching application code or databases.

Dynamic Data masking is an effective strategy in reducing the risk of data exposure to insiders and outsiders in organizations and is a best practice for securing production databases.



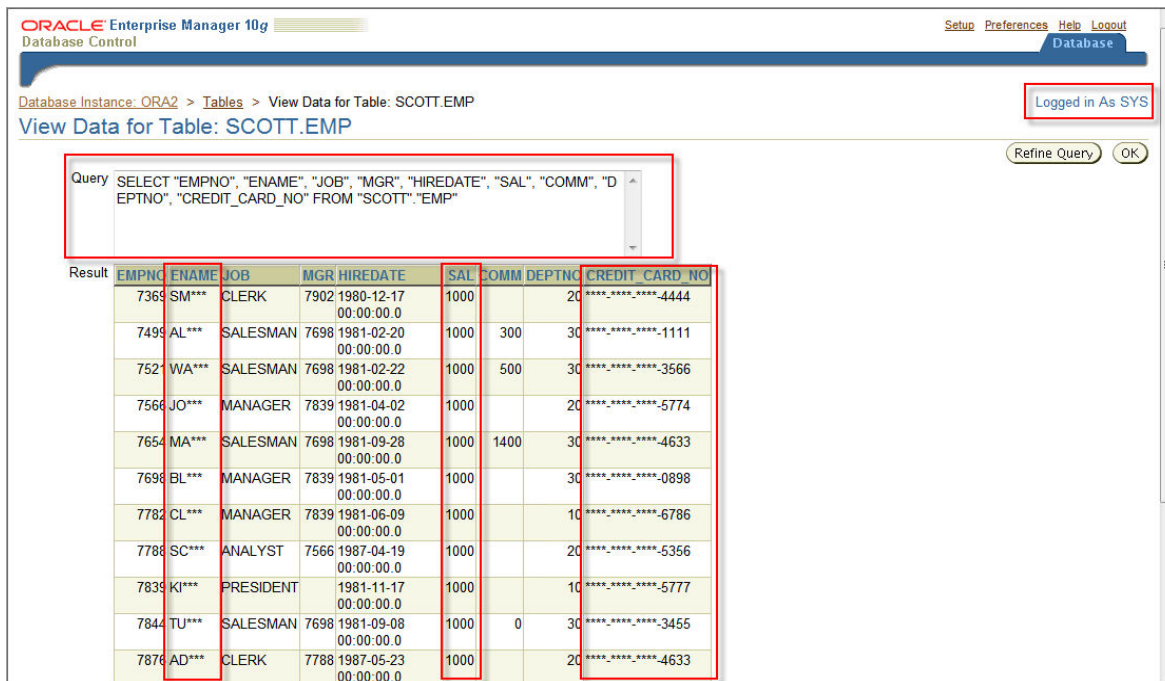


Figure 2: ActiveBase Security™ Dynamic Data Masking is transparently masking name, salary and credit card number in application screens, packaged reports and development tools accessing production databases, including users who have the highest database privileges.

Benefits

1. ActiveBase quickly protects personal financial information by dynamically masking it from end-users, IT support teams and outsourced personnel who do not require it to perform their job.
2. Quickly comply with new security requirements with no application modifications, no database or physical data changes.
3. Dynamic Data Masking is 1/10 the cost and time of applying other solutions.
4. Enforces security policies across different applications and tools within days.
5. Can be easily deployed and maintained with little to no involvement from network operations, DBAs or application managers, and does not require dedicated headcount.

How does our Dynamic Data Masking solution work?

ActiveBase Dynamic Data Masking solution (ActiveBase Security™) is built on a patented SQL*Net software in-line proxy, transparently installed between applications and databases, acting as a database listener. All inbound application requests coming from application screens, canned reports, development tools, batches, transactions and interfaces travel through the proxy, analyzed, actions performed and then sent to the database for prompt execution.

Through a simple, yet elegant, rules engine, criteria can be specified to identify which SQL statements are to be acted upon. When there is a match, one or more actions can be applied, including Mask, Re-write, Block, Re-Direct or simply to Audit the action creating a Database Access Audit Trail.

ActiveBase Security™. transparently intercepts incoming SQL requests and applies security rules in real-time on them. One of the simplest illustrations:

- a.) an application sends in 'select name, ..'
- b.) which dynamically gets re-written to 'select substr(name,1,2) || '***'...'
- c.) which will take the name 'Obama' and display it as "Ob****"
- d) The following expression is used to mask credit card numbers in productions:

```
nl2 (CREDIT_CARD_NO, ('****-****-****-  
  ' || substr (CREDIT_CARD_NO, length (trim (CREDIT_CARD_NO)) -  
  3), ''))
```

ActiveBase can use any existing functions in your database portfolio. Some of the frequently used masking, scrambling and blocking algorithms include:

- Data substitution, replacing a value in the column with fictionalized data
- Truncating, scrambling, hiding or nullifying, which replaces column values with NULL or "****"
- Randomization, replacing the value with random data
- Skewing, which alters the numeric data by a random variance
- Using masking or scrambling functions with fictitious values
- Character substring masking, replacing a particular substring with a custom mask
- User extensibility is available with custom PL./SQL functions and/or a Java API

ActiveBase is "application agnostic". It has already been shown to work with any packaged or home-grown application, including Siebel, PeopleSoft, Amdocs and LHS Billing systems, Oracle Apps ERP suite, Clarify, HR Access, Business Objects, Cognos, Crystal Reports, Brio, SQRIBE, ESRI and many more. ActiveBase Security™ intercepts user requests before they reach the database, applying predefined and customized security policies in real-time. ActiveBase Security™ is easy to deploy and IS fully transparent to clients, applications and databases.

Where does it fit?

Dynamic Data Masking is the only application security approach that enables the protection of information in ANY environment from users who do not require it for performing their jobs, including but not limited to:

- 1. Production environments:** the Achilles Heel of most Data Security Strategies. Privileged Users, such as external workforce, part time employees, production DBAs, tech support, help desk, QA and others are exposed to PII as a natural by-product of doing their jobs. Dynamic Data Masking shuts down this exposure without mitigating these users in performing their work. This is especially critical when these users are outsourced, and even more offshore.
- 2. Training environments:** used by customers for internal and external end-user training: Training systems are infested with PII, and used daily by untrained personnel. Due to the complexities of these applications and the flow of data between applications, physical masking of data through one application is impossible to apply, as masked data will now flow to other applications, thus losing all referential integrity. As Dynamic Data Masking keeps the original value intact, the data stored in the database is not changed –keeping data referential integrity is consistent when flowing to other applications.
- 3. Pre-production, QA and test environments (non-production):**
Used by internal IT personnel for development and pre-production testing. Non-production environments have always been created by copying production data, including all PII.

Dynamic Data Masking can assist in securing non-production environments in several ways:

1. Combining Physical Masking with Dynamic Data Masking, offering a Hybrid security solution: Physical masking is complex to define and execute, while requiring an additional processing step for physically masking the PII before releasing a new testing environment. In many cases, physical masking requires a long processing time, prolonging the release of the new testing environment.

Adding a Dynamic Data Masking solution complements the physical masking process. Dynamic Data Masking helps to secure data that is too complex to physically mask (such as data that is referential in other applications).

2. Dynamic Data Masking solution can be implemented transparently when Exporting PII from production environments. Dynamic Data Masking can intercept the Export SQL command and rewrite it in order to extract only masked PII. This solution ensures that PII does not

leave the production servers, shortens the time requires for creating new non-production environments and simplifies the physical masking process by eliminating the physical masking required.

3. ActiveBase Security™ supports database extracts (using database connections - DBlinks) from all major ETL tools such as Informatica or DataStage as well as in-house developed scripts using Export/Import.

Incremental Implementation

The implementation of our Dynamic Data Masking solution is done using a unique methodology that accelerates implementation time from months into weeks. The methodology includes the following steps:

Step 1: Test Scenarios

The best way to get a full scope of the project is to start by defining test scenarios that will also be used for final acceptance testing. These scenarios should document:

1. The on-line application screens, packaged reports and batch processes impacted by the masking
2. The actual data to be masked, as presented in the application screens and packaged reports
3. The fields masking requirements
 - Masking functions
 - Business rules and constraints on masked data to be respected
 - Expected behavior of queries on masked fields
 - Expected behavior of update on masked fields
4. Map processes that should not be impacted by the masking (e.g. ETL...), bypassing ActiveBase Security™

In many cases, such test scenarios already exist for application testing purposes.

Step 2: Mapping Data

Once data to be masked has been clearly identified at the application level, the first effort will be to map this data to actual technical objects such as tables, view, stored procedures...

This mapping can be achieved through multiple sources, which will most probably be used jointly:

- Application expertise, Database knowledge and naming standards
- ActiveBase Security in logging mode can be used while running a specific scenario in order to see the complete interaction with the database.
- Database integrity constraints and object dependencies.

Mapping data is the most time-consuming part of the project

Step 3: Developing Masking Rules

Once a masked field has been mapped to the database, rules are easily created in the Dynamic Data Masking solution in order to implement the masking.

Most of the work required to handle masking of select statements with the Masking Rule is straightforward. However, special cases may have to be handled by specific rules:

Behavior of queries on masked fields: The default behavior is to query with the entered value against the actual database value.

Behavior of update of masked fields: This phase ends when the acceptance test scenarios can be run successfully.

Step 4: Acceptance Test

Using the masking rules defined in step 3, all test scenarios defined in step 1 are rerun, this time with masked results.