semele

The Benefits of Obfuscated Vs. Manufactured Data

A Complete Solution for Complex Data Testing Environments
semeledata.com

It is widely understood that manually generated test data is inadequate for all but the most trivial of testing. One reason is labor cost. Not only does it require an extensive amount of manual labor, but also requisite knowledge for producing meaningful test data is often high. Consequently, high cost personnel are often relied upon to produce this test data. For the business, it is more cost effective to focus them on activities that are more profitable. To control labor costs, a form of automatic generation of test data is clearly needed.

Using production data for testing is faster and less expensive, while providing the complexity needed for truly robust testing.

One way to automatically generate test data is to simply manufacture the data, using randomization or iterative data generation techniques. This is less labor-intensive than manually generated data, and there is the added benefit that large volume can easily be produced for stress and performance testing. However, the effectiveness of this technique is limited. How well documented and understood are these production systems? There are complexities of data relationships (referential integrity) within databases that demand a lot of time to understand and account for. There are also often intricate linkages to other systems that come into play. It is a daunting task to develop software to properly create data adhering to these relationships even if the linkages are understood. To further complicate matters, there are the vagaries of corner cases and other anomalies that inevitably occur in real world production systems. Manufactured data can be helpful for simple data that does not contain these complexities, but it is generally insufficient on its own.

The Advantages of Using Production Data for Testing

A far more effective way of automatically generating robust test data is by using production data. There are several advantages, including:

- **Legitimacy**. Manufactured data are often composed of lightly distinguishable values (e.g. “FirstName1” and “LastName1”) or inscrutable randomized gibberish (e.g. “K3vt6yrus1df” and “J2nmx9675xh9”). Where human interaction is needed, such as User Acceptance Testing (UAT), such data appears illegitimate and therefore tends to impede human interaction. Imagine, for example, a user in UAT attempting to validate that some action was properly performed in the system by visually inspecting a related screen in the user interface, looking for the name “K3vt6yrus1df J2nmx9675xh9”. It would take a person far longer to locate and validate this than it would if they were looking for “John Smith”. Unlike computers, people more readily lock in on data that look legitimate. Illegitimate data diminish productivity of test personnel.

- **Validity**. Randomly generated values are not always valid. For example, if the application being tested requires valid credit card numbers for test processing, the generated values might be required to adhere to the checksum calculation of the Luhn algorithm. There are also cases where the values of multiple fields have validity dependent on their combination. For example, if the application requires valid address information, such as the combination of city, state, and zip code, simple random values will not suffice. An even more complex example is medical history records. Depending, for example, on the types of past exams for a patient, there may be restrictions on the sequence of dates for follow-on exams of a particular type. Purely random data cannot adhere to this requirement. Software for generating meaningful and valid combinations can be costly to configure and maintain over time, especially as new combinations are introduced or discovered.
- **Complexity**. Manufactured test data tend to be somewhat simplistic, since developers and test personnel are often pressed for time. As previously mentioned, complex systems tend to have complex referential integrity and inter-system linkages between many different subject areas and sometimes even disparate database platforms. Specific test scenarios can (and should) be created ahead of time for feature-specific functional testing. However, unlike statically created test data, actual production data tends to grow more and more combinatorically complex over time. Such complexity is generally unpredictable, often resulting in new combinations not accounted for in feature scenario testing but uncovered in the production system because of software or configuration changes. Maintaining use cases and their traceability to features and configuration parameters tend to become more complex and costly over time.
- **Shape**. Real world data have particular statistical distribution (sometimes referred to as “shape”) of values and combinations of data relationships. These can be extremely important for such applications as demographic analytics. For example, certain genders, ethnicities or geographic locations might exhibit higher correlation to certain diseases than others in medical data. Maintaining this distribution can be vital for successful functional testing of the application. It can also affect performance. Once again, maintaining the software to generate these conditions becomes increasingly difficult and costly over time.

Testing cycles can be accelerated since less time is required by test developers to create quality data.

- **Timeliness**. Since data validity and complexity are inherently preserved and there is less need for developers to spend time accounting for it, production data can be more quickly obtained. This allows for more quickly tapping into the infamous corner cases and unexpected anomalies resulting from software and configuration changes. Since such data can be more quickly obtained from the production system, multiple iterations of testing are practical, allowing for a wider range of scenarios and for faster resolution of bugs before the cost of impact becomes much higher.

Although the advantages are compelling, there are also significant challenges associated with the use of production data for testing in lower environments.

The Challenge of Excessive Volume

One challenge with using production data for testing is that it is technically impractical to copy all of the data from production to lower level development and test environments. Lower environments do not typically have sufficient storage and computing capacity to handle the staggering volumes of data contained in production systems. Fortunately, this limitation can be overcome by using a subset engine, such as is available in Semele's subsetting solution. A separate white paper explores this issue in greater detail.

Using production data for testing is dangerous due to the risk of exposure of sensitive data.

The Challenge of Sensitive Data

The far more important challenge with using production data for testing is the existence of sensitive data. This is a fear that keeps executive management and the legal team in companies awake at night. This is an issue that – if not properly handled – can have severe financial, marketing, and possibly even legal impact. We have all heard nationally televised stories of data breaches that have occurred in this prominent financial organization's or that health insurance company's system. The ensuing fines and scandal attending the wake of their revelation are shocking. The recent Equifax data breach and scandal is one of the more infamous examples. This event will likely have far reaching impact well beyond the life of the story.

To safeguard against the threat imposed by sensitive data, organizations invest a lot of money on hardware and software security devices and associated procedures for their production systems. This is especially true of externally facing, Internet-accessible systems. But the cost and complexity of these security measures are seldom justifiable for internal lower environments. The temptation might be to reason that, since these environments are internal, they do not impose the same threat as external systems. This assumption is both dangerous and incorrect. According to research by Forrester, Verizon, and others, internal attacks account for anywhere between 15%-40% of all data security breaches. That's a very significant amount, and it's probably understated, since many internal attacks are likely never reported, either to avoid the publicity or because they simply remain undetected. The catastrophic potential of internal attacks is enormous, since rogue actions or egregious safety oversights of internal personnel would effectively unlock the doors to the sensitive data contained in these less secure internal systems. It is simply too dangerous to risk exposure of sensitive, production data to use it for testing. Or is it…?

Safe, Production Quality Test Data Using Obfuscation

It is true that we never want to *expose* sensitive production data in lower environments. However, production *quality* data is absolutely needed for the numerous reasons cited above. What if we

could remove the sensitive nature of the data, allowing us to capitalize on the advantages? That is where data obfuscation comes in.

Data obfuscation (also sometimes referred to as de-identification, scrubbing, sanitizing, and or other such terms) is a process for transforming sensitive data so that it can be protected from misuse and safely used for testing. Data such as Protected Heath Information (PHI), Payment Card Industry (PCI), and other Personally Identifiable Information (PII) fall into this category of sensitive data. Protection is governed by various standards, including the Healthcare Insurance Portability and Accountability Act (HIPAA), Payment Card Industry Data Security Standard (PCI-DSS) and the Gramm-Leach-Bliley Act (GLBA).

Semele provides a library of pre- built transformation algorithms and technological capabilities for data obfuscation that meet this challenge and then some. Highlights include:

- Random strings with character pattern and keep/overlay masking
- Hash values, using various algorithms (SHA-1, MD5, etc.), lengths, and output formats
- Domain list, chosen from a list of input values provided in a file or dynamically generated by "shuffling" the values of the field from the input records
- Multi-Column Domain, a special case of the Domain with multiple columns that are collectively transformed as a unit, such as often required for addresses
- Social Security Numbers, optionally including/excluding reserved and test number ranges as specified by the Social Security Administration
- Credit Card numbers, with controls over length, prefix/suffix preservation, and guaranteeing the generated number has a valid checksum digit per the Luhn algorithm
- Numeric Sequences, Random Numbers, and Percent Variances
- Random Dates and Date Variances, along with date format designations
- General capabilities available to many of the transformation algorithms include maintaining statistical distribution of output values based on input values, unique output values, non-null output values, guaranteeing output is different than input, and several others
- There is also support for custom transformation algorithms, using a built-in Lua script interpreter engine
- Transformations can be grouped, allowing for consistent obfuscation across multiple database tables and/or files and even multiple database platforms

Data obfuscation transforms sensitive data and protects it for use in testing.

Using obfuscation techniques, **Semele** can protect sensitive data while simultaneously tapping into robust production data, providing the legitimacy, validity, complexity, shape and timeliness required for robust testing of real world systems.

The Bottom Line

- Using production data to generate test data is less expensive than manual methods and it provides the legitimacy, validity, complexity, shape and timeliness that are necessary for truly robust testing of mission-critical systems.
- Obfuscation is an effective way to protect sensitive, production-quality data and make it available for testing.

Semele was developed by Meridian Technologies, a leading consulting, staffing and technology firm, as a solution to a challenge facing many of its clients: the need for a complete enterprise solution for fast, efficient and secure sourcing and masking of production data for testing. Now you can leverage that expertise for your business's own test data challenges.

The Semele Advantage

Semele is an enterprise solution for the fast, efficient, and secure subsetting and masking of data for testing. It was designed specifically to work within complex data environments. It is sophisticated yet easy-to- install, easy-to-use and supports all major databases.

- **Semele Test Data Subsetting & Obfuscation:** Simultaneously subset, transform and protect data for testing using an automated, repeatable solution.
- **Semele Test Data Comparison:** An innovative comparison utility that allows testers to compare data across any platform to quickly identify differences.
- **Semele Audit Solution:** An automated solution that compares values in the lower environments with those in production to identify the location of toxic data.

Contact:

Don Kiernan
Vice President, Sales
904-512-7917
don@semeledata.com

ACCELERATE YOUR TESTING PROCESS

- Subset utilizing a rules engine to create smaller, purposeful test data sets
- Easily create referentially intact subsets of data
- Obfuscate data "in-flight"
- Maintain referential integrity *across* databases without storing any PII