WHITEPAPER

Test data provisioning for development

How to efficiently deliver realistic and compliant data

Compliant
Database DevOps

redgate

Test data provisioning for development

Contents

Introduction

Database teams are under growing pressure to deliver more value to the business and speed up releases, while ensuring the ongoing protection of sensitive data. This means that having a robust and efficient process for managing the delivery of test data to development environments is more important than ever.

The aim of database testing is to improve the quality of software before release, so that bugs are caught earlier, where it's cheaper and faster to fix them, and to increase confidence when updates are deployed to customers. Nothing slows release cycles down quite as severely as reworks, and nothing's more frustrating than pinpointing poor-quality test data as the root cause.

But provisioning great test data efficiently, cost-effectively, and safely is a challenge.

Testing requirements vary across the different environments and stages of the release pipeline. The closer we get to production, the more pressure we want to apply to our work through performance testing and by running through scenarios that, although rare, could feasibly occur. Significant investment in infrastructure is often required to facilitate this, which isn't practical for development environments earlier in the release pipeline.

The goal of the development team at the start of the testing process is to uncover as many data-related issues as possible as early as possible. In order to do that, developers need data that accurately reflects the production environment's nuanced structure, complexity, and referential integrity.

'Shift-left testing' is a useful concept here. The idea is to shift production behaviors and processes to the left, in the context of a left to right workflow eg Dev > Test > Staging > Production. That's not just to staging environments, but as far as possible into development. If an application is never fully tested against real data until it reaches production, it will likely encounter data-related issues and suffer performance problems. By shifting left, code can be tested against a range of values, volume, and distribution of data comparable to production, so that data-related problems are caught earlier.

However, the shift left concept also introduces security and administration challenges for the people tasked with managing production data. Without the right approach, provisioning copies of production or production-like data to development teams can be difficult and resource-heavy, resulting in escalating storage costs and long wait times for up-to-date data.

And then there's the question of how to handle sensitive data. Development teams have always had a duty to safeguard this information. But the growth in data privacy regulations the world over, spearheaded by the GDPR, requires teams to assess whether they're really doing everything in their power to protect it.

In this paper, we'll review the most common approaches to provisioning test data to development, and assess their capabilities for delivering realistic data, removing bottlenecks, and meeting data privacy and protection regulations. We'll also share cases studies about organizations who have implemented successful processes.

> **"Within the space, we also see an increased emphasis on test data provisioning, as opposed to merely test data management. This is likely due to continued interest in DevOps practices, as well as Agile and continuous testing."**
>
> **Bloor 2019 Market Update on Test Data Management**

Test data provisioning approaches

Badly designed processes for delivering test data downstream can grind things to a halt. Developers waiting for a refresh or a new dataset are blocked from progressing their changes upstream. Data that doesn't resemble production makes testing less reliable but using actual production data leads to expensive storage costs and risks non-compliance with data privacy legislation. In this section, we'll explore how the most common approaches to test data provisioning stack up.

Restore from backup

Realistic data **Removing bottlenecks** **Data privacy**

According to the State of Database DevOps report, 65% of organizations copy down raw production data for development, typically taken from a backup. While the data is certainly realistic – you can't get more production-like than a copy of production – this method is flawed when it comes to data privacy and causes significant bottlenecks when refreshing environments.

Restoring from a backup is a slow process. The datasets are often large, causing a bottleneck for developers waiting on fresh data. An automated, overnight process can help, but it does little for ad hoc requests or emergency situations. Teams may be tempted to just work with the dataset they currently have, but an out-of-date dataset sacrifices quality and risks errors later. A slow process for provisioning test data invariably means a slow response to reworks and impacts an organization's ability to deploy updates and outpace the competition.

Copying production data is also expensive. Storage must be allocated for each copy. Depending on the size of the data and the number of development environments, this could incur additional hardware, licensing, and support costs. For some organizations, cost is the main blocker to a transition from developing on a shared database to developers working on their own local sandboxes.

A dataset that's simply an untreated copy of production will, in most cases, contain sensitive data. Development environments, by necessity, don't have production-grade security controls – they're open to a much wider group of users and have lower levels of perimeter defense. When you move sensitive data into development environments you increase your risk of a data breach, whether through malicious or accidental means, and face the threat of noncompliance. Stiff fines from regulators and negative press could have a significant impact on company finances.

As a method for delivering test data to development teams, restore from backup simply doesn't meet today's requirements for speed and safety.

Data subsetting

Realistic data

Removing bottlenecks

Data privacy

Provisioning full copies of production data for development environments is inefficient, expensive, and leaves organizations at risk of noncompliance in the event of a data breach. The aim of data subsetting is to reduce the overall size of the dataset so that it's faster and easier to distribute and test with. While this approach helps to remove some bottlenecks, it introduces challenges around ensuring the data is representative and referentially intact, and it doesn't automatically alleviate compliance concerns.

Development teams want to work with data that accurately reflects the real world, to better test their work before pushing it upstream. Providing a copy of the entire production database is too costly in time and disk space, especially when refreshing multiple developer environments. Data subsetting involves cutting the dataset, for example from several hundred million records to several hundred thousand records. This enables developers to execute their tests faster. Reducing the byte size, for example from 1TB to 1GB per copy, also reduces storage and infrastructure costs, leading to faster refresh times.

However, excluding such a large chunk of records makes it difficult to accurately reflect the demographic shape, distribution, and characteristics of production. If you reduce your dataset by 95%, for example, you may break foreign key constraints or end up with a statistically non-representative data sample. Shift-left testing aims to catch bugs where it's cheapest to fix them, but a dataset that isn't realistic enough to cover an adequate range of test cases leads to fewer bugs being caught in development, not more.

Further, a slice of production data is likely to include confidential information. One option would be to remove all the sensitive bits as the subset gets created, but this only adds to the challenge of arriving at a realistic dataset.

The big plus for data subsetting as an approach is the small size, which makes it faster to deliver test data to teams while keeping infrastructure costs to a minimum. But these gains in efficiency are cancelled out by the work required to ensure that the eventual dataset is also realistic and compliant.

> **"You must have a way of ensuring that your subset is representative of your entire dataset, and it must be referentially intact."**
>
> **Bloor 2019 Market Update on Test Data Management**

Data virtualization

Realistic data **Removing bottlenecks** **Data privacy**

Another approach that results in lightweight and easy to provision datasets is data virtualization. Like data subsetting, the aim is to take a copy of production and shrink it, removing bottlenecks and reducing infrastructure and storage costs. However, unlike data subsetting, data virtualization enables complete datasets to be provisioned so that development teams receive a mirror copy of production to work with.

Virtual copies work just like a normal database – they can be developed on and tested, and their updates migrated upstream – but their tiny footprint makes them incredibly agile and cost effective. Their small size is due to each copy referencing a master dataset, rather than physically containing the actual data. As the master dataset can be a complete copy of the production environment, each clone created from it is fully representative and referentially intact. With realistic test data, developers can take greater ownership of their changes, reducing reliance on others to pick up bugs later.

Data virtualization leads to short wait times for fresh data, reducing the time it takes to provision the latest update from hours or days to seconds. The minimal disk space requirement also removes blockers to sandboxed development caused by concerns around mounting storage costs. Sandboxed database development is a recommended DevOps practice, as it allows for increased experimentation and aligns application and database team processes.

Data virtualization does much to remove bottlenecks and provide a fast and efficient way of provisioning production data to development environments. However, without additional treatment, the master dataset is likely to contain confidential information. As such, organizations face the same data privacy concerns as outlined above.

> **“With data virtualisation, you never have to worry whether your test dataset is representative, because it consists of your entire dataset.”**
>
> **Bloor 2019 Market Update on Test Data Management**

Synthetic data generation

Realistic data **Removing bottlenecks** **Data privacy**

Generating fake data is a guaranteed way of ensuring that no confidential information is present in test datasets. This enables development teams to carry out their work without worrying about compliance and helps to protect the business in the event of a data breach. But as the balance tips in favor of data privacy, there are challenges to contend with around producing realistic datasets and provisioning them efficiently.

Producing synthetic data through automation, as opposed to by hand, allows for the creation of quite complex datasets. Without too much effort, it's possible to generate large volumes of perfectly reasonable data, which on viewing appears realistic, at least in terms of the type of data expected to be present in each column, eg valid email addresses. The difficult thing is being able to generate test data that accurately reflects the shape and size of production. Without the correct spread of demographic information, for example, the dataset is unlikely to cover enough test cases to make it truly useful for developers to work with. The effort involved in generating accurate data is often so high it may actually be easier to do it by hand.

The size of the generated dataset may also present similar challenges to before. If it's simply a synthetic representation of a full production database, then additional disk space will be required for each environment. Updating the dataset is also no mean feat. As new data enters and changes the shape of production, it's critical that test data keeps up, so the data model needs constant attention. If production changes frequently, this could quickly become a bottleneck.

Synthetic data generation is the safest way to prevent sensitive information from being spread around the data estate when provisioning test data. However, the trade-offs are a set of significant challenges around ensuring test data is representative, and around allocating storage.

> **"One of the chief advantages of synthetic data is that it is fundamentally not real, and therefore cannot be sensitive."**
>
> **Bloor 2019 Market Update on Test Data Management**

Data masking

Realistic data **Removing bottlenecks** **Data privacy**

An alternative approach to generating synthetic data is to mask the data you already have. Whereas synthetic data generation replaces all the data with fake values, data masking only replaces the sensitive data, resulting in a sanitized copy of production. Development teams can now be confident that the data they're working with is both realistic and safe. That said, simply swapping out the sensitive information means the dataset is identical in size to the original copy, so this approach has bottlenecks similar to restore from backup and synthetic data generation.

With data masking, sensitive data is replaced through executing a masking script against the databases. Apart from the values that have been masked, the data is intact, so it's practically identical to the real thing in terms of its size, shape, and demographic makeup. Data masking also benefits from automation. As new environments are refreshed with data from the most recent backup, a masking script is immediately run to sanitize them. Reducing the time period where sensitive data exists in non-production environments is crucial in mitigating the impact of a data breach. Wherever confidential information lives, even for a short while, there's the potential for unauthorized access or an accidental leak to occur. A safer approach is to mask the data before it's provisioned to development environments, which in turn reduces the number of times the masking script needs to be executed.

With data masking, the same difficulties exist as with distributing full copies of production data, with the additional step of masking. Bottlenecks around checking the data for sensitive values have been removed, offering some improvement, but data masking alone is not going to satisfy the requirements of agile development teams, even if it does appease data protection officers – pseudonymization is a recommended technique in the GDPR, for example.

> **"Sensitive data is of extreme importance right now, thanks to GDPR as well as upcoming regulations. This has produced a great deal of interest in the data security space, and this has had a knock-on effect on test data management and particularly data masking."**
>
> **Bloor 2019 Market Update on Test Data Management**

Data masking and data virtualization

Realistic data **Removing bottlenecks** **Data privacy**

All the approaches above have plusses and minuses when it comes to provisioning realistic and compliant test data to development efficiently. Data virtualization surpasses all other methods in terms of speed and efficiency and removes all but a few bottlenecks. Those bottlenecks that remain lie with what the master dataset contains, rather than with the technology itself. If the clones simply reference a copy of production, then it's likely that sensitive data will be distributed and exposed to developers. But if the clones reference a sanitized dataset, such as can be achieved with data masking, then we immediately eliminate data privacy concerns.

A combined data masking and data virtualization approach enables lightweight, realistic, and compliant test data to be provisioned to development teams efficiently and with ease. Infrastructure and storage concerns go away, and development environments can be refreshed in seconds with trustworthy data. Each clone is an exact replica of the masked database, delivered in the same way, every time.

Data masking and data virtualization together solve virtually all the major challenges for provisioning test data, which creates new opportunities for database development teams. It enables a transition from shared to sandboxed development environments, thanks to the storage savings of data virtualization. Lightweight clones containing sanitized data allow for self-service and automated provisioning processes. Shift-left testing is made possible, as developers can receive a steady stream of realistic, up-to-date, and safe data to work with, driving up the quality and reliability of database updates.

> **"Test data provisioning particularly benefits from a data virtualisation capability."**
>
> **Bloor 2019 Market Update on Test Data Management**

> **"Test data (or copy data) virtualization is a technology that is increasingly popular, when used in combination with Static Data Masking, to speed up the provisioning of and updates to target environments, in addition to significantly reducing the amount of storage required by these environments."**
>
> **Gartner 2018 Market Guide for Data Masking**

Case studies

PASS

With over 300,000 members and more than 250 local chapters around the world, PASS (Professional Association for SQL Server) has a large database that is constantly being accessed and updated through its various websites.

To meet the requirements of global data privacy regulations, PASS implemented a Compliant Database DevOps process to improve the way they developed their database and ensure the protection of sensitive information.

By implementing SQL Provision, PASS was able to move away from working on a shared database and transform the way test data was delivered to development environments.

Developers now test their changes against their own copies of the production database which, while masked, are fully representative of the real database and can be provisioned to them in seconds.

Read the full case study at
www.red-gate.com/solutions/entrypage/pass-compliant-database-devops

KEPRO

KEPRO, a leading US healthcare provider, encountered difficulties ensuring that the data supplied to development was HIPAA compliant and updated regularly.

The old method of restore from backup was slow and limited data refreshes to once a quarter, holding back development teams from receiving up-to-date data.

The team was also under pressure to speed up development cycles and get updates out as quickly as possible, so they needed a process that could match up.

Since implementing Redgate SQL Provision, KEPRO have been able to get a new offshore development team up and running and compliant and save 15-20 hours a week in provisioning test data.

Read the full case study at
www.red-gate.com/products/dba/sql-provision/resources/kepro-ensuring-hipaa-compliance-with-sql-provision

Summary

As database teams grapple with shortening release cycles and tightening data protection laws, the need to deliver realistic and compliant test data to development quickly and safely is greater than ever. Choosing the wrong test data provisioning approach can hamper development, drive down quality, risk non-compliance, and lead to escalating infrastructure costs.

Traditional approaches like restore from backup and data subsetting are unable to meet the demands of contemporary database development. And while synthetic data generation provides a failsafe way of preventing sensitive information from reaching development environments, it involves significant bottlenecks.

The answer is to implement a combined data masking and data virtualization approach. Data masking plus data virtualization solves virtually all the major challenges for provisioning realistic and compliant data and opens new opportunities for database development teams. It enables teams to transition from shared to dedicated development environments, shift testing left, and refresh data on-demand through self-service and automated processes.

Organizations such as PASS and KEPRO are implementing data masking and data virtualization with SQL Provision to streamline test data provisioning, transform their approach to database development, and deliver value to their customers faster.

SQL Provision combines data virtualization with data masking to enable realistic and compliant test data to be delivered to developers at speed.

The web app provides a single pane of glass for managing test data provisioning processes, and provides oversight on where clones are located, how much disk space they're using, when they were created, and by who.

SQL Provision empowers teams to adopt self-service and automation to streamline test data provisioning. By reducing storage requirements by up to 99%, it makes transitioning from shared to dedicated development environments easy.

Learn more at www.red-gate.com/products/dba/sql-provision/