



# Test Data 101

**An Entirely New Approach to  
Creating Quality, Safe Data for  
Testing and Development**

Spring 2021

# Overview

## **4     Part I: Definitions and the Value of Test Data**

- 5     Why Generate Test Data Instead of Using Production Data?
- 6     Defining PII
- 7     Defining PHI
- 8     5 Traditional Approaches to Generating Test Data
- 12    Applications of Test Data
- 14    Coming Up in Part II: Modern Approaches to Generating Test Data

## **15    Part II: Modern Approaches to Generating Test Data: Anonymization, Synthesis and Mimicking**

- 17    Anonymization: Transforming Real Data into Test Data
- 22    Synthesis: Generating New Test Data
- 23    Mimicked Data: The Crossroads of Anonymization and Synthesis

## **25    Part III: Anonymization, Synthesis, and Mimicked Data: The Pros and Con**

- 28    4 Reasons Why Mimicked Data Is Safer and More Useful than Traditional Test Data
- 30    Build vs. Buy: Should You Create or Outsource Your Mimicked Data Generator?
- 32    Better Testing Comes from Better Data

## Part I

# Definitions and the Value of Test Data

Testing is the foundation of development. Successful tests mark the boundary from concept to code, from idea to prototype or from a minimum viable product (MVP) to a market leader. The problem is that generating truly productive test data is extremely difficult. No one can imagine everything that could go wrong. Great test data starts with thoughtful design.



## Why Generate Test Data Instead of Using Production Data?

“More than the act of testing, the act of designing tests is one of the best bug preventers known. The thinking that must be done to create a useful test can discover and eliminate bugs before they are coded – indeed, test-design thinking can discover and eliminate bugs at every stage in the creation of software, from conception to specification, to design, coding and the rest.”

**Dr. Boris Beizer**

UPenn Prof. of Computer Science, author of *Software Testing Techniques*

Test data includes any data that application developers and testers use in order to test software functionality and identify bugs and performance issues.

If the data is not sensitive, confidential, or personally identifiable (e.g. local forecasts and temperatures for a weather mobile app), developers can use real-world information as their test data. For example, an app answering questions can use actual user inputs as long as they don't contain identifying information.

But whenever privacy and security are a concern, which is true for almost all data involving people, then test data should not include information on real-world individuals or events in order to reduce the risks of exposure, both for the people involved and for the company. Over the course of this primer, we will look at the consequences of using real-world data, even when partially anonymized, and outline best practices for generating test data that is safe to use while fully preserving the utility of an original dataset.

Test data can be generated by anonymizing fields within a real dataset or by creating a new dataset from scratch. In part II of this series, we will go into detail on anonymization, synthesis and an entirely new approach incorporating both, which we call “mimicking.” This new approach is designed specifically to create high-quality, safe test data matched to the composition of your existing production data.

The goal is to protect the PII and PHI of anyone in your databases while equipping your developers with the data they need to get their work done effectively.

## Defining PII

The most sensitive data is personally identifiable information (PII). The fields most commonly targeted by hackers and other malicious actors include:

- Name
- Address, including zip code
- Phone number
- Email address
- Account numbers
- Social Security number, even just the last 4
- Credit card numbers
- Bank information
- Birth date
- Age
- Nationality
- Job history
- Educational history
- Partial data that can be combined to identify an individual

It's important to realize that companies have a legal obligation to not only protect this data, but also prevent partially anonymized data from being re-identified. Learn more in this article on [Reverse Engineering Your Test Data](#).

## Defining PHI

As defined by the Health Insurance Portability and Accountability Act (HIPAA), protected health information (PHI) is also another important target for de-identification. PHI refers to a wide swath of data about a person's medical history. The types of PHI that may need to be replaced with test data include

- Diagnoses
- Medical conditions
- Medical history
- Biometric data
- Test and laboratory results
- Prescriptions
- Health insurance
- Demographics (birthdate, gender, race/ethnicity, etc.)
- Emergency contact information
- Medical images

While companies already face heavy fines and a devastating loss of consumer confidence with breaches involving PII data, PHI loss can be even more serious. The bare minimum fine for a violation of HIPAA guidelines is \$50,000. Breaches are treated as a criminal offence with penalties of up to \$250,000 for a single individual plus restitution for the victims. In some cases the penalty involves a jail term. Take the time to investigate What Are the Penalties for HIPAA Violations.

# 5 Traditional Approaches to Generating Test Data

“As a rule, software systems do not work well until they have been used, and have failed repeatedly, in real applications.”

Dr. Dave Parnas  
Carnegie Mellon Prof. of Electrical Engineering

Historically, developers have used a number of approaches to create test data when they didn’t have or were unable to use production data in their lower environments.

Below, we’ll discuss five of the most common methods used to date to generate data and how they stack up for use in testing and development.

1

## Dummy Data

“Dummy data,” as the name suggests, acts as a placeholder for real data during development and testing. Using dummy data ensures that all of the data fields in a database are occupied, so that a program doesn’t return an error when querying the database.

Dummy data is usually nothing more than data or records generated on the fly by developers in a hurry. However, dummy data can also refer to more carefully designed tests, e.g. in order to produce a given response or test a particular software feature.

This is not ideal for large scale testing because dummy data can lead to unintended impacts and bugs unless it has been thoroughly evaluated and documented.

user_id	first_name	last_name	job_title	company
1	kahsdkjahdc	lajkdsclkjd	asdc	ooqweiasdc
2	asdc	ad	dafadf	adc
3	adfvdafv	dfv	adffgb	klajsd
4	bsbff	vsdfsd	bsdfd	lkajdscsd
5	lkajdc	lkajdc	ljac	ascdoiec
6	sfgbsf	svfsdfv	sfv	poqwcldkasc

Dummy Data Example

2 Mock Data

Mock data is similar to dummy data, but generated more consistently, usually by an automated system, and on a larger scale. The purpose of mock data is to simulate real-world data in a controlled manner, often in uncommon ways. For example, mock data can be generated for database fields corresponding to a person’s first and last names, testing what happens for inputs such as:

- Very long or very short strings
- Strings with accented or special characters
- Blank strings
- Strings with numerical values
- Strings with reserved words (e.g. “NULL”)

Mock data can be useful throughout the software development process, from coding to testing and quality assurance. In most cases, mock data is generated in-house using various rules, scripts, and/or open-source data generation libraries.

The more complex the software and production data, the harder it is to mock data that retains referential integrity across tables.

user_id	first_name	last_name	job_title	company
1	David	Smith	VP of Operations	Acme Corporation
2	Amanpreet	Singh	Director of Engineering	123, LLC
3	Jane	Doe	Customer Success	ABC, Inc.
4	Mohammad	Akbar	VP of Engineering	MBI, Inc.
5	Long	Nguyen	QA Engineer	Orange Co
6	Mitesh	Gandhi	Software QA	9876, LLC

Mock Data Example

3 Anonymized (De-identified) Data

Real data that has been altered or masked is called “anonymized” or “de-identified.” The purpose is to retain the character of the real dataset while allowing you to use it without exposing PII or PHI.

More specifically, anonymization involves replacing real data with altered content via one-to-one data transformation. This new content often preserves the utility of the original data (e.g. real names are replaced with fictitious names), but it can also simply scramble or null out the original data with random characters.

Data anonymization can be performed in-house but it places an immense responsibility on the in-house development team to ensure that it’s done safely. Increasingly, companies are turning to third-party vendors to provide proven software solutions that offer privacy guarantees.

user_id	first_name	last_name	social_security	date_of_birth
1	Jon	Doe	123-45-6789	1-1-1111
2	Jane	Smith	111-111-1111	01-02-2099
3	Mark	Thomas	098-765-4321	01-01-01
4	Karl	White	xxx-xxx-xxxx	34-12-3421
5	Daniel	Green	000-000-0000	1-1-1001
6	Brad	Williams	445-231-2322	4-2-1212

Anonymized Data Example



## 4

**Subsetted Data**

Subsetting is the act of creating a smaller, referentially-intact version of a dataset by:

- pulling a percentage of that dataset, or
- limiting it through a “select where” clause to target individual records as needed.

For example: give me 5% of all transactions or pull all data associated with customers who live in California.

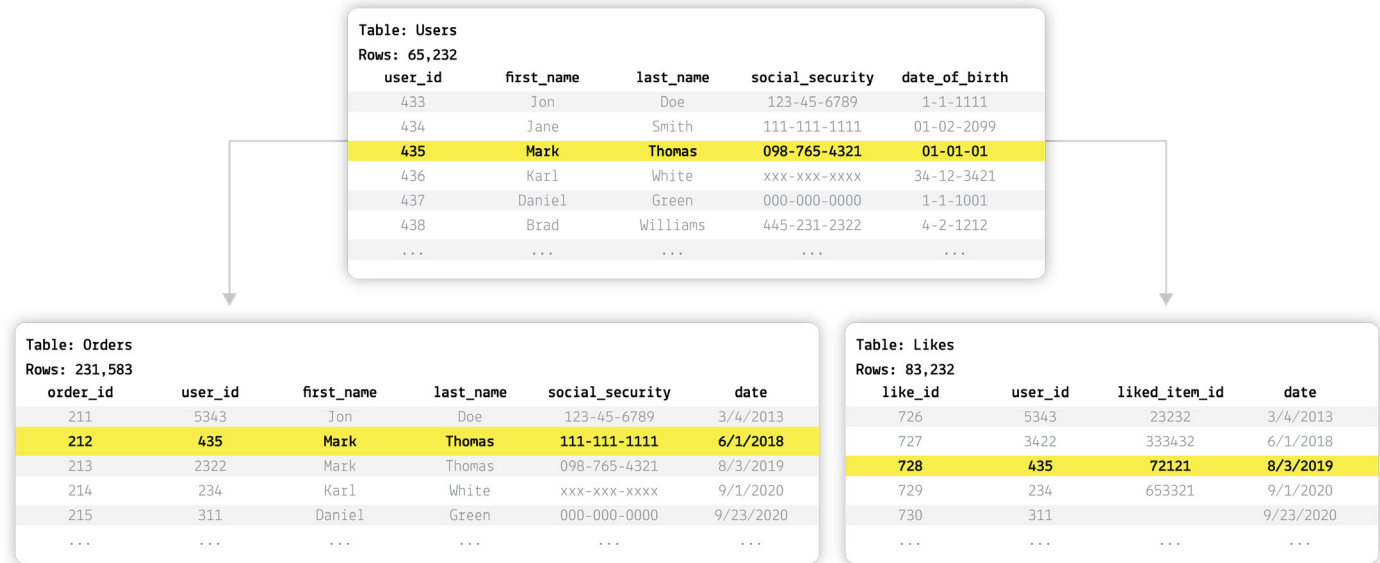
Subsetting allows for creating a dataset sized specifically to your needs, environments, and simulations without unnecessary data and often constructed in order to target specific bugs or use cases. Using data subsetting also helps reduce the database’s footprint, improving speed during development and testing.

Data subsetting must be handled with care in order to form a consistent, coherent subset that is representative of the larger dataset without becoming unwieldy. Given the growing complexity of today’s ecosystems, this is fast becoming an impressive challenge. At the same time, demand for subsetting is increasing with the growing shift toward microservices and the need for developers to have data that fits on their laptops to enable work in their local environments.

On its own, subsetting does not protect the data contained in the subset; it simply minimizes the amount of data at risk of exposure.

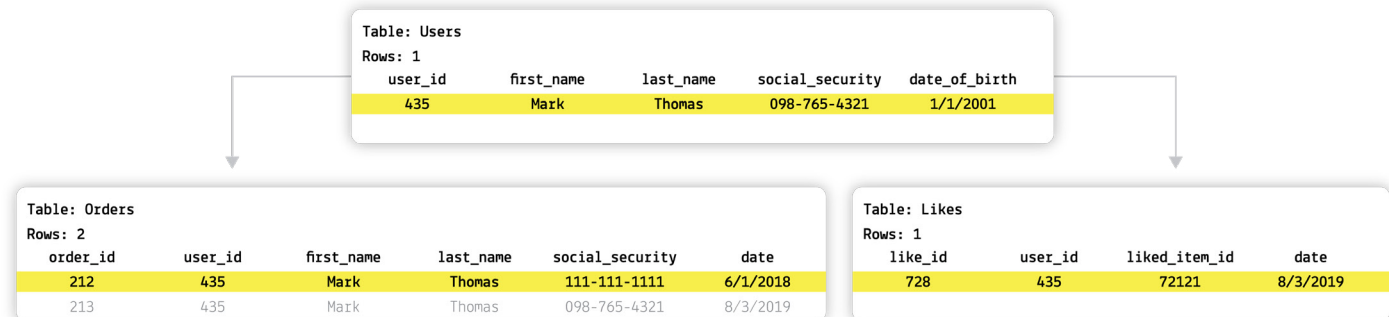
# Original Dataset

**Rows: 65,232**



# Subset

**Rows: 1**



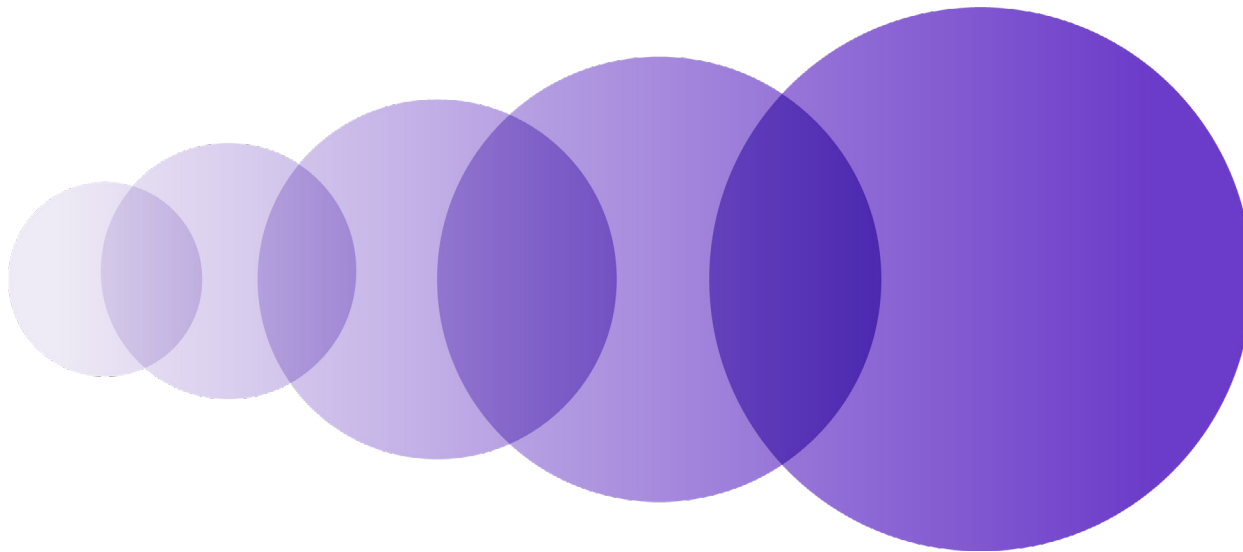
## 5

**Data bursts**

Automated production of generated data in large batches are called data bursts. This is most useful when you care less about data itself, and more about the volume and velocity of data as it passes through the software. This is particularly useful in simulating traffic spikes or user onboarding campaigns.

For example, data bursts can be used in the following testing and QA scenarios:

- **Performance testing:** Testing various aspects of a system's performance, e.g. its speed, reliability, scalability, response time, stability, and resource usage (including CPU, memory, and storage).
- **Load testing:** Stress testing a system by seeing how its performance changes under increasingly heavy levels of load (e.g. increasing the number of simultaneous users or requests).
- **High availability testing:** Databases should always be tested to handle bursts in incoming requests during high-traffic scenarios, but testing for high-data scenarios is also important. Preparing for data burst scenarios helps ensure that queries continue to run optimally with zero impact on the application performance and user experience.



# Applications of Test Data

“Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it.”

**Dr. Brian Kernighan**  
Princeton Prof. of Computer Science

Developers and QA engineers know they need test data to build, test, and debug their applications, but the quality of the test data they use has a big impact on their success in the long run.

Test data that looks, acts, and behaves like production data will provide more accurate development environments, higher quality testing, and rapid debugging.

When test data is generated without an adequate understanding of how the software was built or how data passes between functional areas of the code or how outliers in a dataset can cause bad behaviors in the product, the results will be disappointing to say the least.

Here are six areas where high quality test data can have a huge impact on development and testing.

**1**

## QA Environments

By identifying and maintaining various requirements and standards, QA ensures that the development team delivers the most bug-free and useful software products possible. QA environments can perform many types of functional and non-functional testing, but all require well-designed datasets that can stand up to rigorous testing.

**2**

## Debugging

Test data generation tools can create custom data specifically for testing and debugging, which means:

- More accurate testing environments to reproduce production bugs
- Datasets with a specific characteristic or feature for unit testing (i.e. testing an individual unit or component of a software application)
- Datasets that are very diverse in order to test the limits of the system
- Extremely large datasets for load testing and performance testing

Another valuable tool in debugging is data subsetting, which can narrow down a large dataset to the specific rows or entries that are causing an issue. For more information, see the definition of subsetting in the previous section.

### 3 CI/CD and DevOps

CI/CD and DevOps are two popular trends in software development accelerating software delivery pipelines and quality.

Automated testing is a crucial step for CI/CD. Modern pipelines are built with automation baked in, from code check-in to deployment. Throughout a deployment pipeline, various stages can trigger automated tests. Test data that mimics the real-world ensures higher quality tests, fewer breaks in automation, and improved MTTR (mean-time-to-release) of features.

### 4 Product Demos

Software demos are one of the best ways for you to show off the results of what you've built. Yet for applications that rely heavily on data, how can you give a successful demonstration of the product's capabilities without sharing a real dataset with untrusted third parties?

Realistic synthetic data is the answer. By fully anonymizing existing data or generating new test data, you can make use of synthetic datasets in your product demos, or share them with potential customers to use in a sandbox, without fear of exposing sensitive or confidential information.

### 5 Customer Support

Customer support teams are both tasked with resolving bugs and challenged with not having full access to production data. High quality test data provides three key advantages to support teams:

- A subset of the data with a custom select where clause that only selects the necessary data to triage a bug.
- More accurate data environments allow for more realistic behaviors in order to replicate bugs that customers are experiencing.

- Multiple datasets for user “personas” that represent customer segments based on demographics and behaviors. You can then analyze this information to better understand your end user without violating privacy concerns.

### 6 Machine Learning

High quality test data is essential for machine learning, and synthesized data has been found to be as good as real data for use as ML training data. [Research by MIT](#) found that in 70 percent of experiments, machine learning models trained on artificial datasets resulted in no significant performance loss vs. models that were trained on real data.

Realistic test data can also be put to use in testing machine learning pipelines, which often get very complicated and need testing of their own. The pipeline that takes real world data and cleans it and prepares it for ML takes a great amount of time due to the complicated nature of the process. Providing realistic data to test the pipeline can significantly streamline the process.

What's more, a quality test data generator can readily expand existing datasets to create additional training data in cases where the existing data is limited. Noise can also be added to existing data, to create better, more comprehensive testing overall.

Recently, concerns about bias in machine learning due to skewed or selective datasets have permeated the field. According to Gartner, [85 percent of AI projects will suffer from errors by 2022](#) due to biases baked into the data, the biases programmed into algorithms, or bias in the teams responsible for managing AI. Removing human bias from the equation as much as possible is a growing field of study within data generation, the caveat, of course, being that machine learning practitioners need to be sure that the generation process itself is unbiased.

## Coming Up in Part II: Modern Approaches to Generating Test Data

To conduct a real test, yesterday's approaches of random data generation or de-identification to the point of uselessness won't work. Your test data must feel as close as possible to production data if you want to mirror real-world problems.

That said, you also can't afford to expose the identity or private information of individuals, even when you are sharing data within the company. The legal bar for exposure includes data that is only partially anonymized and could theoretically be used to re-identify individuals.

Today, all test data must be robust enough to work the same way as production data does and have safeguards in place to guarantee data privacy. As long as you can ensure privacy, the closer your test data comes to its production counterpart, the better your outcomes will be all down the line.

Next, we'll look at three modern approaches to generating test data based on the complexity of your development project and what the data needs to do for you. The three approaches we will examine in detail are a) Anonymization (or De-identification); b) Synthesis; and c) a new approach that combines elements of both, Mimicking Data.

Test data that looks, acts, and behaves like production data will provide more accurate development environments, higher quality testing, and rapid debugging.

## Part II

# Modern Approaches to Generating Test Data: Anonymization, Synthesis and Mimicking

In part I of this series, we introduced essential terms like PII and PHI. We also outlined traditional approaches to generating test data and their use cases throughout software development.

Now, we will dive deeper into the recent advances in data generation that allow you to produce privacy-protecting and utility-preserving test data that can stand up to today's complex data ecosystems. At a high level, these approaches include:



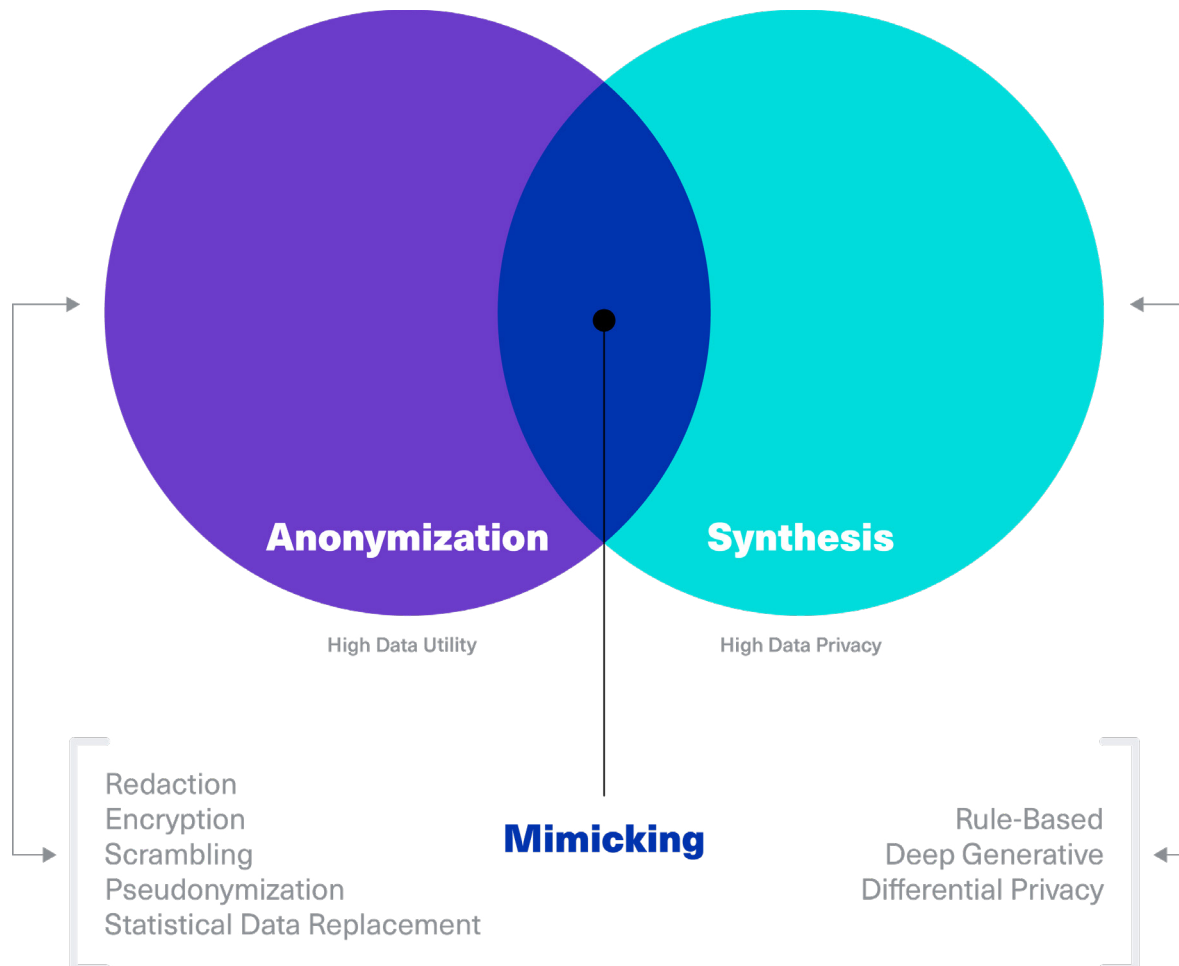
**Anonymization: Transforming  
Real Data into Test Data**



**Synthesis: Generating  
New Test Data**



**Mimicked Data: The Crossroads  
of Anonymization and Synthesis**



**Anonymization:** Also known as de-identification, this is the process of starting with a real dataset and transforming or truncating data in specific fields so the test data can no longer be used to re-identify a real world individual.

**Synthesis:** This is the process of creating new data from scratch, using predefined rules and schemas, or from existing data, by learning that data's rules and schemas. Machine learning can also be used to perform data synthesis.

**Mimicking:** This approach represents the next generation in data generation, combining anonymization and synthesis in a way that most effectively mimics real production data without exposing sensitive or private information.



# Anonymization: Transforming Real Data into Test Data

There are many pathways to anonymizing (or de-identifying) real data. These represent five of the most common ones.

## 1 Redaction

Redaction may be the simplest data anonymization technique. This simply means removing or replacing part of a field's contents. For example, PII in a dataset might be deleted or written over with dummy characters (such as Xs or asterisks) in order to "mask" the data.

Redaction is a good option when dealing with sensitive data that is no longer actively used by your organization. The rest of the dataset which doesn't involve personal information can still be used for other purposes.

However, data redaction is also a drastic measure. Normally, data redacted can't be restored or recovered and loses most of its utility for real-world use cases.



Users	
user_id	34
first_name	David
last_name	Smith
job_title	VP of Operations
credit_card	378282246310005

Users	
user_id	34
first_name	David
last_name	S*****
job_title	XXXXXXXX
credit_card	*****

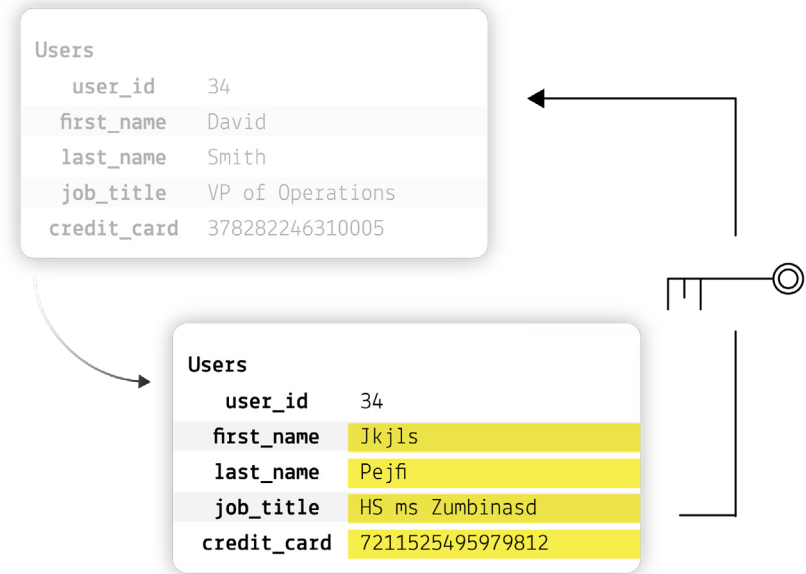
## 2

**Format-Preserving Encryption (FPE)**

Data encryption is the process of encoding information using a secret called the encryption key. More specifically, format-preserving encryption encodes the information in a way that preserves the format of the original input in the output. E.g. a 16-digit credit card number will be encoded to another 16-digit number.

Using data encryption is a strong layer of defense against security breaches, although not a foolproof one. If a malicious actor manages to exfiltrate the encrypted data from your internal network, this information will be useless without the corresponding decryption key. Of course, this means that the key itself needs to be fiercely safeguarded from prying eyes.

Encrypted data generally retains some of its utility—at least for those with the decryption key, which allows the encryption process to be reversed. This is in contrast with one-way hash functions, which also convert information into an indecipherable format but which are computationally infeasible to invert. Whereas data encryption is more often used to protect data in transit or rest, hash functions are used to create a digital fingerprint of data.



## 3

### Scrambling

Like data encryption, data scrambling seeks to convert an input dataset into an entirely new format to preserve the information's confidentiality. However, data scrambling and data encryption are not the same thing: whereas encryption is reversible if you possess the decryption key, data scrambling is permanent and irreversible.

Encryption and scrambling are also different in terms of their methodologies. Data encryption usually makes use of the Advanced Encryption Standard (AES), a specification for the encryption of electronic data established by the U.S. National Institute of Standards and Technology in 2001. Data scrambling, on the other hand, depends on each organization's implementation: it has no predefined rules or guarantees about how the data will be scrambled. For example, data scrambling might replace all but the leading characters in a person's name (e.g. "John Smith" becomes "Jxxx Sxxxx"), or it might randomize the order of the interior characters (e.g. "John Smith" becomes "Jhon Stmih").

Data scrambling is often used when cloning a database from one environment to another. Databases that contain sensitive business information, such as data on human resources, payroll, or customers, must be treated with care during the cloning process. This scrambled database can then be used for purposes such as stress tests and integration tests, which don't depend on the accuracy of the information within the database.

#### Users

user_id	34
first_name	David
last_name	Smith
job_title	VP of Operations
credit_card	378282246310005

#### Users

user_id	34
first_name	Jhon
last_name	Stmih
job_title	VP of Saeiotropn
credit_card	3*****

## 4

**Pseudonymization**

Data pseudonymization is a less comprehensive data anonymization technique that focuses specifically on personally identifiable information (PII), replacing a person's information with an alias. The GDPR formally defines pseudonymization as “the processing of personal data in such a manner that the personal data can no longer be attributed to a specific data subject without the use of additional information.” However, pseudonymized data may retain less identifiable information (e.g. the dates of a student's attendance) in order to maintain the data's statistical utility.

Like encryption, a key feature of data pseudonymization is that it can be reversed when necessary. However, great care must be taken with pseudonymization: data elements that may not seem to be personally identifiable in isolation can become identifiable when viewed together. As such, pseudonymizing data is one of the less safe methods of anonymization, and one that can lead to unintentional re-identification.

Using pseudonymized data is risky because surprisingly few pieces of information are often necessary in order to uniquely identify a person. In many cases, even two or three identifiers are sufficient to narrow the search pool. According to a study from Harvard's Data Privacy Lab, for example, an estimated 87 percent of Americans can be uniquely identified from three pieces of information: their gender, their date of birth, and their 5-digit ZIP code.

**Employees**

employee_id	34
first_name	Kevin
last_name	Mitnick
job_title	Director of Security
salary	50,000,000
zip	90210
date_of_birth	02-10-1975
gender	m
t_shirt_size	L
hire_date	01-01-2021
manager_id	23

**Employees**

employee_id	34
first_name	Jon
last_name	Doe
job_title	Director of Security
salary	50,000,000
zip	90210
date_of_birth	null
gender	m
t_shirt_size	L
hire_date	01-01-2021
manager_id	23

## 5

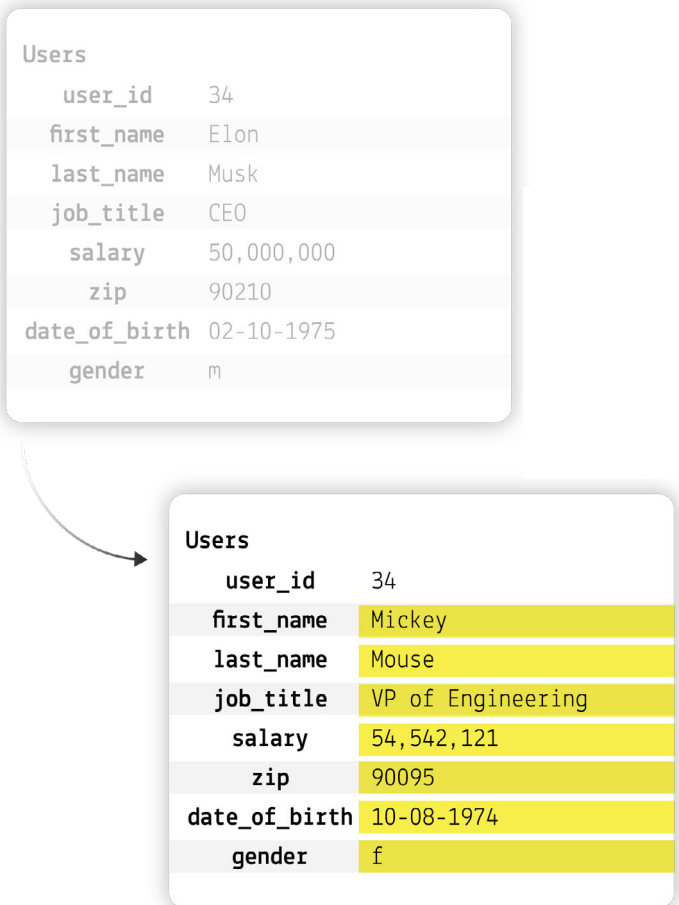
**Statistical Data Replacement**

Commonly known as data masking or obfuscation, this is perhaps the strongest and most sophisticated option when it comes to preventing reidentification with data anonymization. The approach involves learning statistics about the underlying data and then using those statistics to replace the existing data realistically. The information is still obfuscated or obscured, as with data redaction; however, the true records still exist in the underlying database, and can be hidden or revealed as necessary.

Statistical data replacement generates a new dataset by replacing the original with altered content via one-to-one data transformation. This new content often preserves the utility of the original data (e.g. real names are replaced with fictitious names), but it can also simply scramble or null out the original data with random characters when preserving utility isn't required.

Provided that a dataset is masked well, it's typically difficult to reverse engineer or re-identify the underlying data involved, making this a good choice for safeguarding privacy along with utility. That said, not all anonymization is performed equally.

Sometimes it looks a lot more like pseudonymization—so much so that the approach has made headlines recently for the ease with which individuals in “anonymized” datasets have been re-identified. To ensure privacy, it's important to use a powerful masking tool that makes it easy to detect PII, alert users to schema changes that require new masking, and provide comprehensive reports on the PII found and the masking performed.



The diagram illustrates the process of statistical data replacement. It shows two tables, both titled 'Users'. The top table contains real data for a user with ID 34, named Elon Musk, who is a CEO with a salary of 50,000,000, living at zip 90210, born on 02-10-1975, and is male. An arrow points from this table to a second table below it. The second table shows the same user record but with all personal and identifying information masked with fictional or randomized values: the name is now Mickey Mouse, the job title is VP of Engineering, the salary is 54,542,121, the zip is 90095, the date of birth is 10-08-1974, and the gender is female.

Users	
user_id	34
first_name	Elon
last_name	Musk
job_title	CEO
salary	50,000,000
zip	90210
date_of_birth	02-10-1975
gender	m

Users	
user_id	34
first_name	Mickey
last_name	Mouse
job_title	VP of Engineering
salary	54,542,121
zip	90095
date_of_birth	10-08-1974
gender	f

# Synthesis: Generating New Test Data

Data synthesis is a newer approach to creating realistic data, by way of several possible techniques. Here are the two primary methods of data synthesis used today, along with a property to heighten the degree of privacy their outputs provide.

## 1 Rule-Based Data Generation

In rule-based data generation, users define the schema of the dataset that they want to create, and then give the system free rein to generate it (usually by randomly generating values). There are many different open-source libraries and tools that you can use to synthesize realistic test data, such as Mockaroo and [pydbgen](#).

If you want to generate a synthetic dataset of students at a university, for example, you might begin by defining the various fields that you want the dataset to incorporate. This might include student names, genders, birthdates, age, addresses (school and home), email addresses, passwords, GPA, coursework, activities and memberships, etc. Each one of these fields would have an associated type: the name field would be composed of text, the age field would be a number, etc. The synthetic data generation system would then draw upon its existing resources and knowledge (such as a list of common first and last names) to create a realistic-looking dataset that obeys these rules.

## 2 Deep Generative Models

Machine learning architectures such as generative adversarial networks (GANs) are also capable of generating extremely high-quality synthetic data. GANs are a class of deep learning model that consist of two competing networks: a “generator” which tries to create realistic synthetic

data, and a “discriminator” which tries to distinguish between the real data points and the synthetic ones.

In recent years, GANs have made tremendous strides in their capabilities and accuracy. If you need to generate synthetic photographs of people’s faces, for example, you can use a tool such as [thispersondoesnotexist.com](#), while SDV’s open-source [CTGAN](#) model can synthesize tabular databases. Of course, training your own GAN will require a good deal of machine learning knowledge, plus a dataset that’s large enough so that the network doesn’t simply replicate the original information.

### Differential Privacy

An important property that can be applied to either of the above two approaches to synthesizing data is that of making them differentially private. The term differential privacy refers to the practice of adding statistical noise to datasets to prevent malicious actors from detecting the presence of a given record in the dataset. It provides a mathematical guarantee against data re-identification, hardening the security of a synthetic output dataset.

For example, given a dataset, an algorithm might perform some statistical analysis on it (such as the dataset’s mean, variance, mode, etc.). The algorithm is then said to be “differentially private” if, looking at the output, it is impossible to tell whether a given record was included in the calculation, regardless of how typical (or atypical) that record is. Essentially, differential privacy is a way of guaranteeing an individual’s privacy when analyzing a larger dataset in aggregate.

# Mimicked Data: The Crossroads of Anonymization and Synthesis

Anonymization and synthesis are based on different conceptions of how to work with data safely that each come with their own advantages and challenges:

- The goal for best-in-class **Anonymization** is to come as close as possible to production data for high quality testing, while also protecting private information. The process de-identifies individuals in an existing dataset by eliminating or transforming PII and anything that could be used to reconstruct PII.
- In **Synthesis**, the priority is to generate a realistic dataset that is not so closely tied to existing individual records, with a great deal of planning and evaluation put into predefined generation rules, using the requirements of the data's use case as a guide.

**Mimicked** data is a new concept that combines the best aspects of data anonymization and synthesis into an integrated set of capabilities.

1

## Preserving Production Behavior

The goal of data mimicking is to allow developers to finetune the dials and achieve the balance they need between utility and privacy. This is done by providing developers with a single platform in which all of the required features work together, while focusing as heavily on realism as on privacy in the data generation techniques employed.

2

## Hundreds of Databases, Thousands of Tables, PBs of Data

Data mimicking is designed specifically for creating realistic data from existing data. Moreover, it is an approach built for today's complex data ecosystems—tens of thousands of rows and hundreds of tables, spread across multiple databases of different types. To truly mimic today's data, you can't work one table at a time, or even one database. You need to be able to capture relationships throughout your entire ecosystem.

Test data that looks, acts, and behaves like production data will provide more accurate development environments, higher quality testing, and rapid debugging.

3

## Infinite Scenarios

Machine learning may be employed to preserve ratios, relationships, and dependencies within certain data. Differential privacy can be applied during transformations, to muffle the impact of outliers and provide mathematical guarantees of privacy. Columns can be linked and partitioned across tables or databases to mirror the data's complexity, and consistency can be set to ensure that certain inputs always map to the same outputs regardless of their transformations.

The best mimicking infrastructure will also flag PII as sensitive and in need of protection, and alert you to schema changes before any data leaks through to your lower environments. Your data is constantly changing; your test data should too, through the generation of refreshed datasets on demand.

A data mimicking platform's focus isn't to build one model to rule them all, but to enable its users to create a heterogeneous super model containing a multitude of smaller models within it, specific to the subset of data at hand. In this way, it allows for handling different parts of a database in different ways. Differential privacy here, consistency there, machine learning over there, etc. The system can pull in what's needed where it's needed or most appropriate.

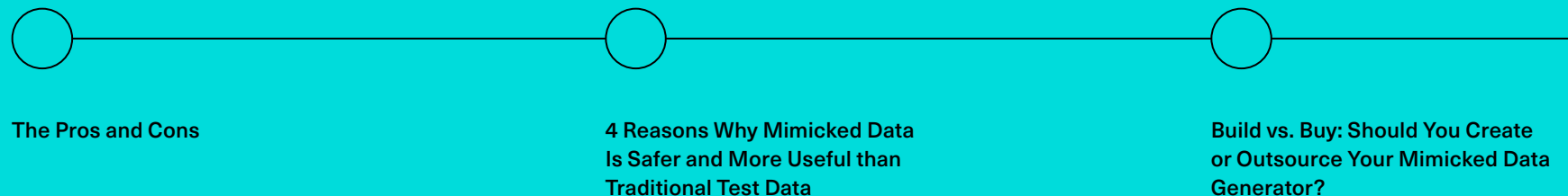
Mimicked data is a new concept that combines the best aspects of anonymization and synthesis. The goal of data mimicking is to allow developers to finetune the dials and achieve the balance they need between utility and privacy.



## Part III

# Anonymization, Synthesis, and Mimicked Data: The Pros and Cons

We'll complete our three-way analysis of anonymization (or deidentification), synthesis, and mimicking by briefly evaluating the benefits and drawbacks of each, then expanding on uses for the new approach of creating mimicked data.



## The Pros and Cons of Anonymization, Synthesis, and Mimicked Data

### + Pros

### − Cons

#### Anonymization

Accurate  
Better at preserving utility

Extremely difficult to preserve privacy  
High risk

#### Synthesis

Increased privacy and security  
Aligned to your needs

Extremely difficult to preserve utility  
Imaginary patterns

#### Mimicking

Maximizes utility and privacy  
The most accurate environment resembling production

Extremely difficult to build

### Anonymization Pros

**More accurate:** There's no question that anonymized data is sophisticated enough to accurately represent the underlying reality. Anonymization preserves the look and feel of the production dataset, but de-identifies the people behind the data.

**Better at preserving utility:** When done well, anonymized data is highly realistic and mirrors the original data with precision. Some anonymization techniques, such as encryption and pseudonymization, are reversible as necessary.

### Anonymization Cons

**Harder to preserve privacy:** The nature of anonymization's one-to-one transformations of real world data entails an increased risk of data re-identification and breach. The consequence of preserving a high degree of utility is that it's hard to do so while also ensuring privacy.

**High risk:** Every environment is less secure than production. When you share data, even internally, you lose control of it and can't say with certainty where it will end up. There are many severe penalties associated with a breach and re-identification of anonymized data gets easier every day.

## Synthesis Pros

**Increased privacy and security:** Because data masking starts from real datasets, there's always the risk that you haven't done enough work to obfuscate the original information properly. Data synthesis creates a completely artificial dataset, making it more secure and preserving the privacy of real individuals.

**Aligned to your needs:** The real data that you have on hand may not be enough or may be lacking in one or more respects. With data synthesis, you can generate as much data as you need, or create unique situations and outliers to fill in holes in your existing datasets.

## Synthesis Cons

**Decreased utility:** Once you've got a system in place, generating endless rows of data is easy, but making that data useful is not. Your project may be so complex and data dependencies so intricate that you spend more time designing and refining the synthetic data algorithm than actually putting the data to use.

**Imaginary patterns:** Algorithms are excellent at identifying patterns, but sometimes they don't know when to stop. Humans can usually identify outliers that skew the data but algorithms have to be perfectly tuned so they are not susceptible to statistical noise.

## Mimicking Pros

**Maximized utility and privacy:** By combining the most powerful tools and techniques of anonymization and synthesis, data mimicking pulls in the pros of each approach in a way that negates their cons. The result is highly realistic, highly secure data that can be used in testing and development with confidence both in the validity of the testing being performed and the privacy of the data involved. Simply put, it's the best of both worlds.

**A mirror of production:** When mimicking data, ratios, relationships, and referential integrity are preserved throughout the generated test database without revealing any real-world information about the production database on which it is based. It creates a mirror of production in the safety of a developer landscape.

## Mimicking Cons

**Extremely difficult to build:** Creating a system for mimicking data that enables all the functionality of the most powerful anonymization and synthesis techniques can be a monumental task. With the growing complexity of today's data, it is becoming less and less feasible for an in-house team to tackle on its own. Not only do all the tools and techniques need to be built out, they need to be built to interface and work with one another. What's more, they need to work with a growing landscape of data types, from SQL to NoSQL across a host of different databases. The complexity of the work required cannot be understated.

## 4 Reasons Why Mimicked Data Is Safer and More Useful than Traditional Test Data

1

### Regulatory Compliance

Data can be tremendously valuable: access to users' personal data is the foundation of business plans for tech giants such as Facebook, Twitter and Google. However, organizations must be sure to use this access responsibly or risk controversies such as the [Exactis scandal](#), which exposed the PII of more than 300 million people and businesses on a publicly accessible server.

Privacy regulations such as the European Union's [GDPR](#) and California's [CCPA](#) are intended to counteract the risk of data misuse. These laws place significant restrictions on how organizations can use, store and retain individuals' personal information. (That's not to mention other standards such as SOC 2, and ISO/IEC 27001.) And although these strong regulations are already in place, there are likely more on the horizon.

The good news is that the GDPR's provisions are "[not applicable to anonymous data](#)," i.e. "information which does not relate to an identified or identifiable natural person or to personal data rendered anonymous in such a manner that the data subject is not or no longer identifiable." The International Association of Privacy Professionals agrees, writing: "Once personal data has been fully anonymized, it is no longer personal data, and subsequent uses of the data are no longer regulated by the GDPR." (Of course, this depends on the assumption that the process is not reversible.)

2

### Security Breaches

Mimicked data allows organizations to use and experiment with valuable datasets, without the constraints imposed by [GDPR guidelines](#) and the [California Consumer Privacy Act \(CCPA\)](#).

There's mounting evidence that [2021 will be the worst year for data breaches yet](#). With new data breaches in the headlines almost constantly, organizations who don't take data security seriously enough risk becoming the next big story. The attackers often aren't after the data for their own uses. They are looking for valuable information that they can sell to the highest bidder.

According to IBM's [2020 Cost of a Data Breach report](#), the average cost of a data breach is \$3.9 million, a figure which rises to \$8.6 million for U.S.-based companies. What's more, this cost breaks down to roughly [\\$150 per record stolen](#). Data breaches cause a tremendous amount of organizational harm—not only the immediate damage as you race to repair the vulnerability but also the long-term economic and reputational damage in the eyes of your investors and customers.

By using mimicked data in the place of real data, organizations dramatically lower the risk they face in the event of a breach. Any information stolen during the attack will be useless in the hands of a malicious actor.

### 3 Data Sharing

Over the course of its useful lifetime, enterprise data may be shared with different teams, departments and organizations, including third-party vendors and offshore partners. Data is shared for reasons such as:

- Collaborating on projects and research
- Breaking down organizational “silos” that prevent synergy
- Distributing analytics insights, reports, and dashboards
- Giving product demonstrations to customers and prospects

The problem, of course, is that as the number of people with access increases, the risk of a data breach (accidental or malicious) also rises exponentially. While you might be able to trust PII in the hands of a few select individuals, this blind trust becomes far more difficult when there are dozens, hundreds or thousands of people in the equation.

The key to safely sharing data internally is designing and building a data de-identification infrastructure. In many cases the precise details of individuals' personal information isn't necessary for data sharing to be successful so mimicked data can be a better alternative to sharing real data. Mimicked data performs as needed and doesn't leave the door open to the possibility of data theft and insider threats.

### 4 Data Proliferation and Distributed Teams

Not only do you need to consider data sharing arrangements with external third parties, you also have to control the internal spread of data within your own organization. According to a 2016 IDG study, the average company now manages 163 terabytes (163,000 gigabytes) of information. This data is often scattered across multiple tools and databases, as well as cloud, hybrid, and on-premises environments: 76 percent of IT executives say that they're “concerned” about the issue of unstructured data sprawl.

Using your own internal data as test data further complicates the issues of data sprawl and data governance, and can run afoul of regulations such as GDPR and CCPA. However, you need to balance these issues with the need to give your developers autonomy to complete their tasks independently, especially in distributed teams, by supplying them with the data they need.

Keeping track of how your data is collected, stored, and shared is difficult enough when everyone is working in the same place. Distributed teams (i.e. teams with members working from different locations) can make data management and data governance more challenging—but they don't have to.

As discussed above, your developers, testers, and QA professionals need high-quality, targeted datasets for use cases such as CI/CD, debugging, QA, product demos, and more. Generating mimicked test datasets gives distributed teams the tools they need to do their jobs no matter where they're located, while ensuring that real data isn't shared with people who don't need it—thereby enforcing your organization's data governance and data access policies.

# Build vs. Buy: Should You Create or Outsource Your Mimicked Data Generator?

From new regulations and data breaches to the ever-increasing complexity of your organization's data footprint, there are many reasons why you need a secure way to generate realistic test data. But should you build your own in-house solution, or purchase a pre-built solution from a data synthesis vendor? Below are three issues to consider.

1

## Technical complexity

Building best-in-class algorithms from anonymization and synthesis techniques to achieve data mimicking capabilities suited to today's data is a massive undertaking. Not only does it require building all of the capabilities addressed herein, it also requires implementing them so that they can work together. CTGAN will need to work with subsetting capabilities which will need to work with format-preserving encryption, etc.

Depending on your data ecosystem, these tools will also need to be built in a way that enables them to work across multiple types of databases. The big division is between SQL and NoSQL databases; NoSQL databases are themselves divided into graph databases, key-value stores, columnar databases, and document databases. You'll need to build an adaptor for each type of database you want to connect to, a challenge which can quickly surpass an organization's in-house technical abilities.

Beyond the database architecture itself, another complication is the sheer scale of enterprise data. According to a 2016 study by market intelligence firm IDC, the average company manages 163 terabytes of information—and the number today is no doubt even higher. Businesses

may need to oversee petabytes of data scattered across millions of rows, thousands of tables, and scores of databases. This makes creating a realistic version of even a subset of a data ecosystem a tremendously involved task. ([See our blog post](#) for more info on what's required.)

2

## Privacy concerns

Alongside the engineering difficulties, you need to be very careful about how you handle and anonymize the data in your possession. Guaranteeing user privacy will require deep knowledge of mathematics and computer science, and few businesses have the budget or need to maintain these experts in-house.

In addition, trying to create your own test data solution can expose your business to financial penalties and reputational damage if real data is exposed during a security breach. Laws such as GDPR and CCPA may require the offending organization to pay hundreds of dollars for each consumer record exposed—so when the dust settles, you could be on the hook for a multimillion-dollar fine.

## 3

**Support and maintenance obligations**

Of course, like any non-trivial software product, your solution for generating test data will need to be kept up to date and maintained over time, to ensure that it still fits your needs and preserves user privacy. This involves a great deal of work: datasets are constantly changing, and a script that works one day may not work a month or year down the road. Even if it does work, a change to the underlying schema may cause real data to leak through accidentally.

Building an in-house solution for generating test data, and keeping it functional, will divert focus from your core business functionality, requiring you to pull time, money, and resources from other areas. Instead of saddling developers with the additional work of building and maintaining an in-house solution, a third-party solution enables them to get their actual job done faster.

## Better Testing Comes from Better Data

The need for safe, realistic test data is intense, but getting there is not easy. Many organizations are using their own in-house custom scripts or running legacy anonymization software. Some don't know the risks involved in using improperly de-identified data, while others don't realize how much QA is missing due to poorly designed synthetic data.

For company leaders, it soon becomes evident that partial methods of generating test data are not sustainable in the long run. In-house scripts are hard to maintain and keep up-to-date in accordance with best practices for synthetic data. What's more, the generated data is often not complex, high-quality, or secure enough, exposing the organization to legal and financial jeopardy.

Meanwhile, synthetic data generation software has its own set of issues. The test data may lack complexity or operate with limited functionality, which prevents it from properly simulating large, complicated real-world datasets.

To be clear, anonymization and synthesis are both useful methods of generating test data within a limited set of environmental conditions. The more a company grows, and the more geographically distributed the teams are, the greater the need for more secure, more complex test datasets.

Here's one last thought on the nature of testing and quality control:

“Some people mistakenly refer to software defects as bugs. When called bugs, they seem like pesky things that should be swatted or even ignored. This trivializes a critical problem and fosters a wrong attitude. Thus, when an engineer says there are only a few bugs left in a program, the reaction is one of relief. \*Supposed, however, that we called them time bombs instead of bugs.\* Would you feel the same sense of relief if a programmer told you that he had thoroughly tested a program and there were only a few time bombs left in it? Just using a different term changes your attitude entirely.”

**Software pioneer Watts Humphrey**

Reflections on Management: How to Manage Your Software Projects, Your Teams, Your Boss, and Yourself

The clock is ticking. Do you know where your data is?



## Contributing Authors



**Ian Coe** is the Co-founder and CEO of Tonic. For over a decade, Ian has worked to advance data-driven solutions by removing barriers impeding teams from answering their most important questions. As an early member of the commercial division at Palantir Technologies, he led teams solving data integration challenges in industries ranging from financial services to the media. At Tableau, he continued to focus on analytics, driving the vision around statistics and the calculation language. As a founder of Tonic, Ian is directing his energies toward synthetic data generation to maximize data utility, protect customer privacy, and drive developer efficiency.



**Omed Habib** is the VP of Marketing at Tonic. A developer at heart, Omed fell in love with fake data as a way to improve developer productivity. He formerly led Product Marketing teams at AppDynamics, Harness.io, and helped launch startups from inception to unicorns. When not faking data, Omed keeps busy geeking out on all things tech, photography, and cooking.



**Andrew Colombi, PhD**, is the Co-founder and CTO of Tonic. Upon completing his PhD in Computer Science and Astrodynamics at UIUC in 2008, Andrew joined Palantir, leading the team of engineers that launched the company into the commercial sector. Later he began Palantir's Foundry product, growing it into a keystone of the company's offering. His extensive work in analytics across a full spectrum of industries gives him an in-depth understanding of the complex realities captured in data. He is building Tonic's platform to engineer data that parallels those realities and supply development teams with the resource-saving tools they most need.



**Chiara Colombi** is the Content Strategist at Tonic. A bilingual wordsmith dedicated to the art of engineering with words, Chiara has over a decade of experience supporting corporate communications at international companies, including Finmeccanica, Enel, and RAI. She once translated for the Pope; it has more overlap with translating for developers than you might think.



### About Tonic

Tonic empowers developers while protecting customer privacy by enabling companies to create safe, synthetic versions of their data for use in software development and testing.

Founded in 2018 with offices in San Francisco and Atlanta, the company is pioneering enterprise tools for database subsetting, de-identification, and synthesis. Thousands of developers use data generated with Tonic on a daily basis to build their products faster in industries as wide ranging as healthcare, financial services, logistics, edtech, and e-commerce. Working with customers like eBay, Flexport, and PwC, Tonic innovates to advance their goal of advocating for the privacy of individuals while enabling companies to do their best work.

For more information, visit [Tonic.ai](https://Tonic.ai).