



# Test Data Management: A Best Practice Guide



[originalsoftware.com](https://originalsoftware.com)

This guide explores the key principals and techniques as they relate to the creation, maintenance, validation, use, and re-use of test data environments on the IBM i, IBM iSeries & IBM AS/400.

TestBench for IBM i is the definitive product for test data management and in each section, we will highlight 'How TestBench Does It' and share some of the secrets behind its success.

## Overview

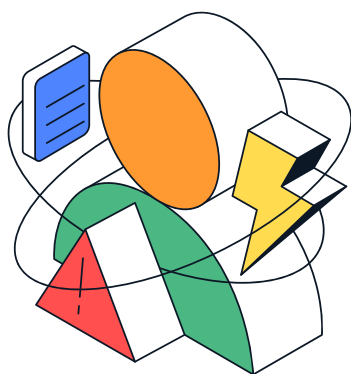
What are the key challenges of test data management?

### Test Data

A copy of the live database is unwieldy, slowing testing and making validation almost impossible.

The ideal test database contains a representative cross-section of the data, which maintains referential integrity.

If this can be achieved, many more test environments can be stored concurrently to better support multiple teams and scenarios.





## Data Confidentiality

GDPR legislation changed the rules for any organization processing data on EU citizens and extended the requirements of industry-specific initiatives such as PCI compliance and HIPAA in the US. What was best practice, became an unavoidable necessity. However, the need for meaningful test data still exists and simply replacing sensitive data with random characters does not work, especially in system and UAT testing.

## Data Re-use

Creation of test data environments is often difficult, so once complete, there is a reluctance to repeat the process. While understandable, the downside is that the data loses connection to current processing dates and is inevitably corrupted as testing is performed.

## Validation

IBM i shops know that, while errors in the UI are painful, the real problems start when the database gets corrupted. Such errors require the affected applications to be taken offline, and in the worst-case scenario, the most recent un-corrupted backup must be restored and any subsequent transactions repeated.

You will find a section on each of these areas in this guide to facilitate your research.

# Test Data

Developers, Project Managers, QA Test Teams, UAT Teams, and Trainers all need test data and ideally in an environment dedicated to them.

So, what do you need to consider?

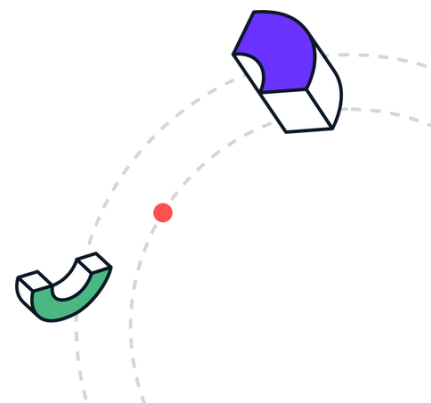
## What do I need?

Everything sounds like a good starting point, and that's where every IT shop started. It is an approach that has worked in the past. But the chances are that you will run out of disk space, and even if the disk is available, you will have only created one test environment when many are needed.

Anyway, big test databases are a bad idea. Each test will take too long – you don't want to wait five hours every time you test the 'end of day' run. Big data sets are also impossible to validate – how can you hope to check the database state if thousands of rows have been affected?

So, a sub-set is the only way to go, but there's a lot more to it. The first 100 rows from every table will result in a smaller database but will be useless as it ignores the need to maintain referential integrity.

Once you've solved referential integrity and remembered to include the foreign keys, you're still not there. To be useful the test environment needs to include a representative selection of master records and transaction data to support all the test scenarios.

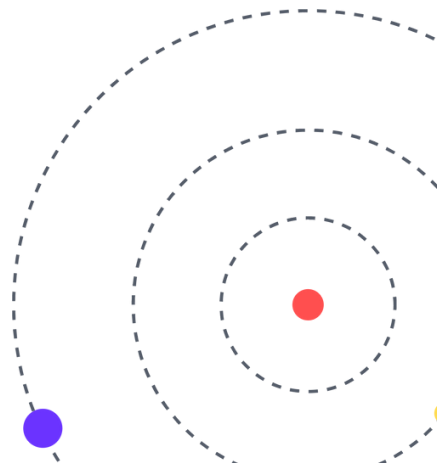
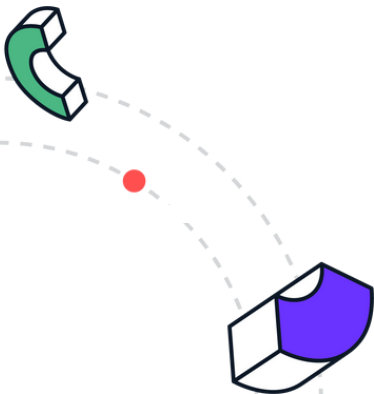




# Test Data

## Your Options

1. Create a series of programs to populate the files from a data source, such as a spreadsheet. This can work, but the workload to create and maintain such a suite of programs is significant.
2. Use an automation tool such as Original Software's TestDrive to create master data and then drive transactions through the user interface or APIs. This is a valid approach but is slow and only suitable for smaller data volumes.
3. Ask the testers to create their own. This may work for small data volumes and as an occasional approach, but is a poor use of the testers' time.



# Test Data

## How TestBench does it...

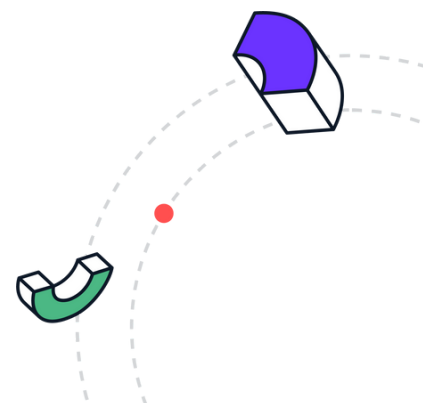
TestBench helps you create what you really want – a representative subset of your live data that respects all aspects of referential integrity. The starting point is a data model that TestBench helps you create using your existing database standards.

Once in place, you can select which records to include at any level of the model to give you total control of the database size and populations.

Alternatively, you can ask TestBench to sample the data held in a file so that for example, you can have 50 records from each combination of state, type, status, method etc., or whatever fields drive the business processes. This is the way to truly representative data.

TestBench will also take care of the creation of a new environment, handling physical, logical, join logical files, triggers, constraints, and SQL tables with ease.

You can extract from IASPs, the local machine, or even remote IBM i servers and you can even modify the data while the extraction is executed.





## Data Confidentiality

GDPR, HIPAA, or simply good practice? It does not matter why, what matters is that you can no longer allow users to access personal or sensitive information held in your live environment. So, what do you need to consider? It takes time and effort to create a test environment but what can you do when that environment becomes corrupted through a failed test or you simply run out of data at the correct initial state?



### Sensitive or What?

GDPR is the broadest piece of data protection to date and has sharp teeth in the shape of significant fines for non-compliance. In broad terms, if you hold any personal data on a citizen of the European Union, you must comply with GDPR, and that means the data must be kept confidential.

In the US, the Privacy Act of 1974 covers the confidentiality of personal data, and more specifically, HIPAA addresses the Health Insurance industry, the Gramm-Leach-Bliley Act, and FSA (UK) addresses Financial Services.

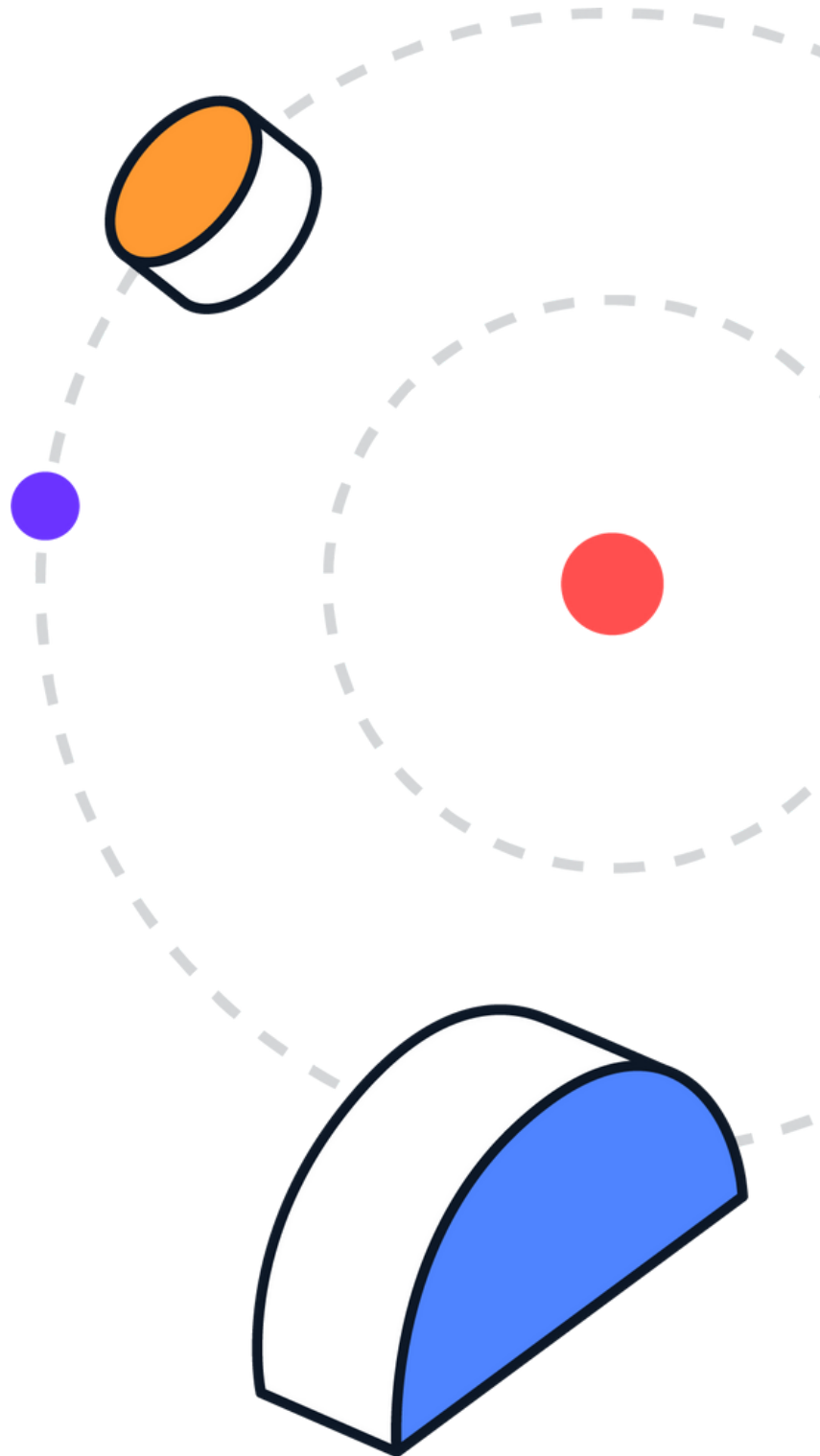
In light of these regulations, internal and external auditors are increasingly focused on data confidentiality, and IT shops need to demonstrate that they have followed best practice.

# Data Confidentiality

## Your Options

It is difficult to effectively mask test data without destroying its usefulness:

1. First, identify all fields in all tables that contain sensitive information. For those of you with program-described files, this task is an order of magnitude more difficult.
2. For each field, decide how the data should be masked.
3. Determine if there are relationships between the fields that must be maintained, and if so, ensure that the masking is synchronized.





# Data Confidentiality

## How TestBench does it...



TestBench provides several options to mask data and respect data confidentiality requirements.

Vertical Scrambling is the starting point for many accounts and may be sufficient for your needs. This moves data between the rows in a table so that the information remains meaningful but untraceable. For example, after scrambling, a row may contain a First Name of 'David', a Surname of 'Smith', and a City of 'New York', however, in the original data all these items came from different rows.

For more specific needs, TestBench provides several custom masking routines such as US Social Security Numbers and Credit Card Numbers. Finally, you can also develop your own masking algorithms and embed them within the TestBench framework.

Uniquely, these masking choices can be synchronized across multiple tables so that the same data is always masked in the same way wherever it occurs.

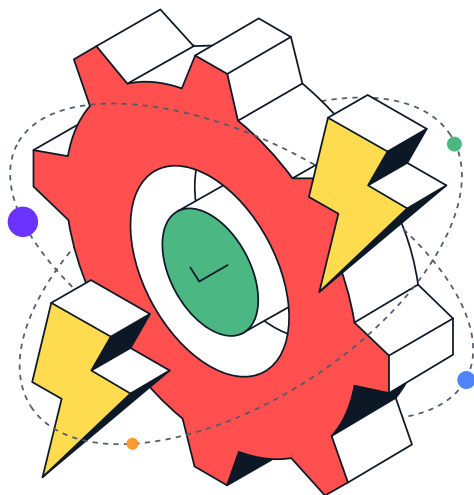


## Data Re-Use

It takes time and effort to create a test environment but what can you do when that environment becomes corrupted through a failed test or you simply run out of data at the correct initial state?

### Need an 'Undo' Button?

What you need is the ability to take your data back to a previous state so that the same test can be re-run. The importance of knowing the state of the data at the start of a test cannot be underestimated. Without a known starting point, the expected state of the data at the end of the test cannot be defined, and without that definition, it is impossible to verify the database-level activity.



# Data Re-Use

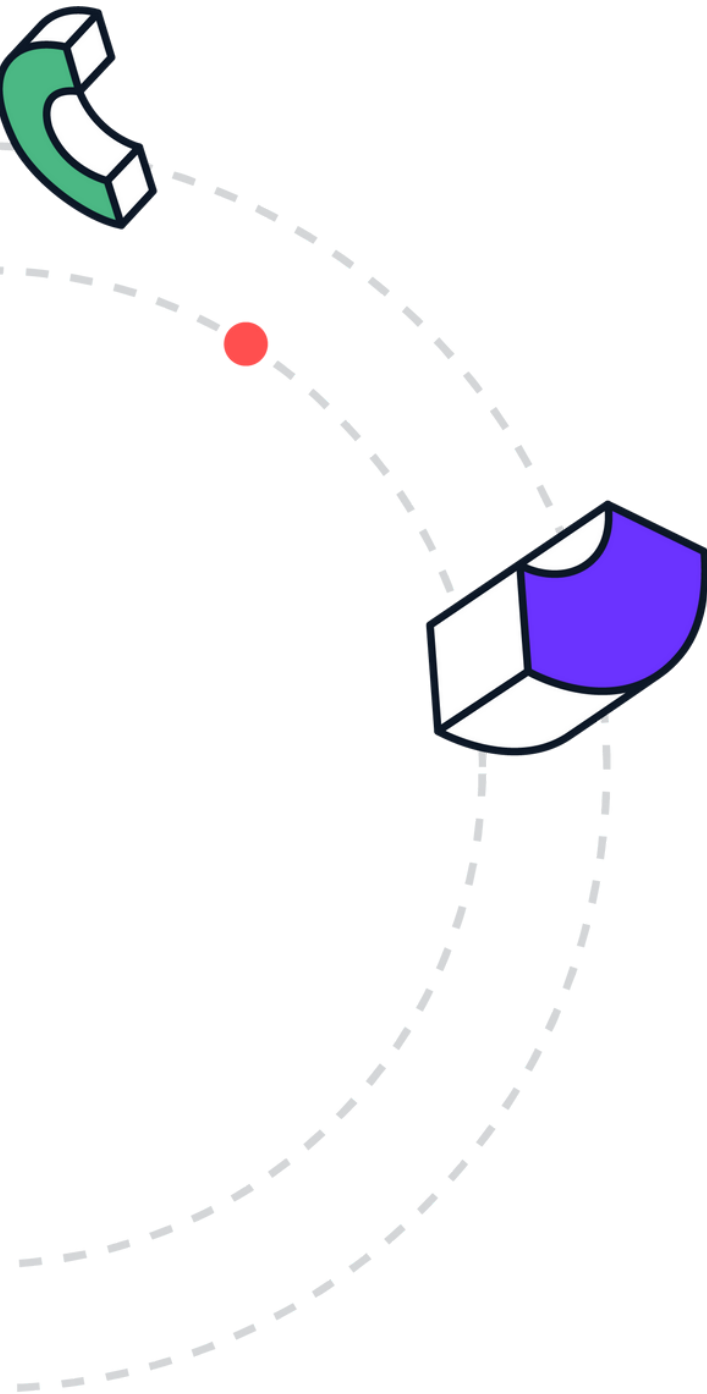
## Your Options

The reality for many IT shops is that they live with test environments which they know are corrupted to a greater or lesser extent. Due diligence at the database level is therefore, significantly compromised or simply impossible. At best, allowances must be made for the final state of the data given the less-than-perfect state of the start data.

1. Take regular back-ups of the test environment and restore it on demand
2. Regularly re-create the test environment

They are the only options, but they take time, and given that the test environment is a shared resource, it is difficult to find opportunities to execute.

This explains why many test environments are frequently out of date and have a level of data corruption.



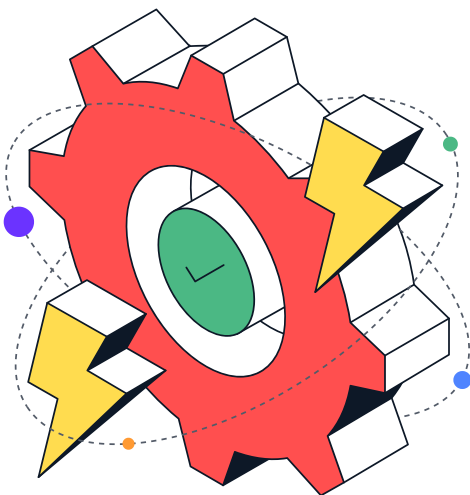
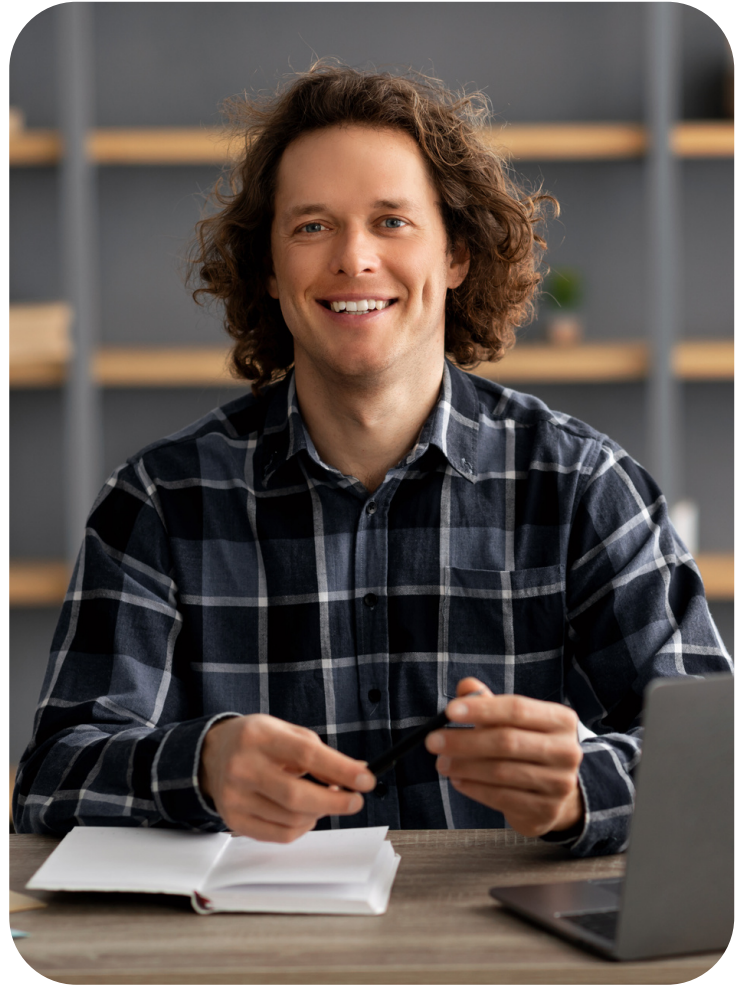


# Data Re-Use

## How TestBench does it...

TestBench provides an all-encompassing, 'Undo' button. This capability is based on the IBM journals but extended to cover other data objects such as data areas. In addition, the IBM journaling support does not cover operations such as Clear Physical File which are handled automatically by TestBench.

You can create as many Checkpoints as required during your use of the test environment, perhaps setting one each time a significant part of the testing has been completed. Once created, you can simply return the test environment to the state that existed at the time the Checkpoint was taken.



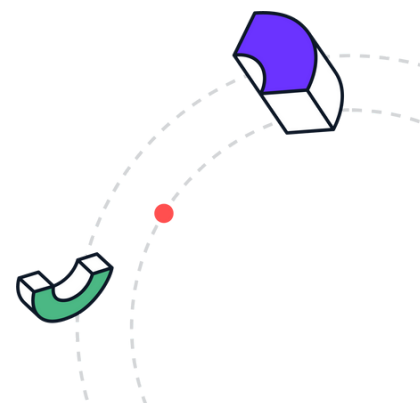
# Validation

IBM i IT shops cover a wide spectrum, from pure green screen, through mixed UI technologies to those for who the IBM i is purely a server supporting a range of APIs. Wherever you are on that spectrum, the IBM i is holding data critical to your company, and it is self-evident that database validation should be a requirement of every test.

## Getting Under the Covers

IBM i applications are rich in database activity and much of that processing is performed by APIs or batch processes that do not have any form of user interface.

Given the damage that a poorly written program can inflict on this key corporate asset, validating that every database activity is correct is difficult and IBM provides few tools to help.

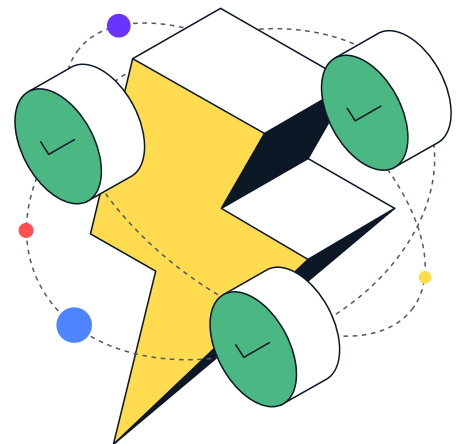
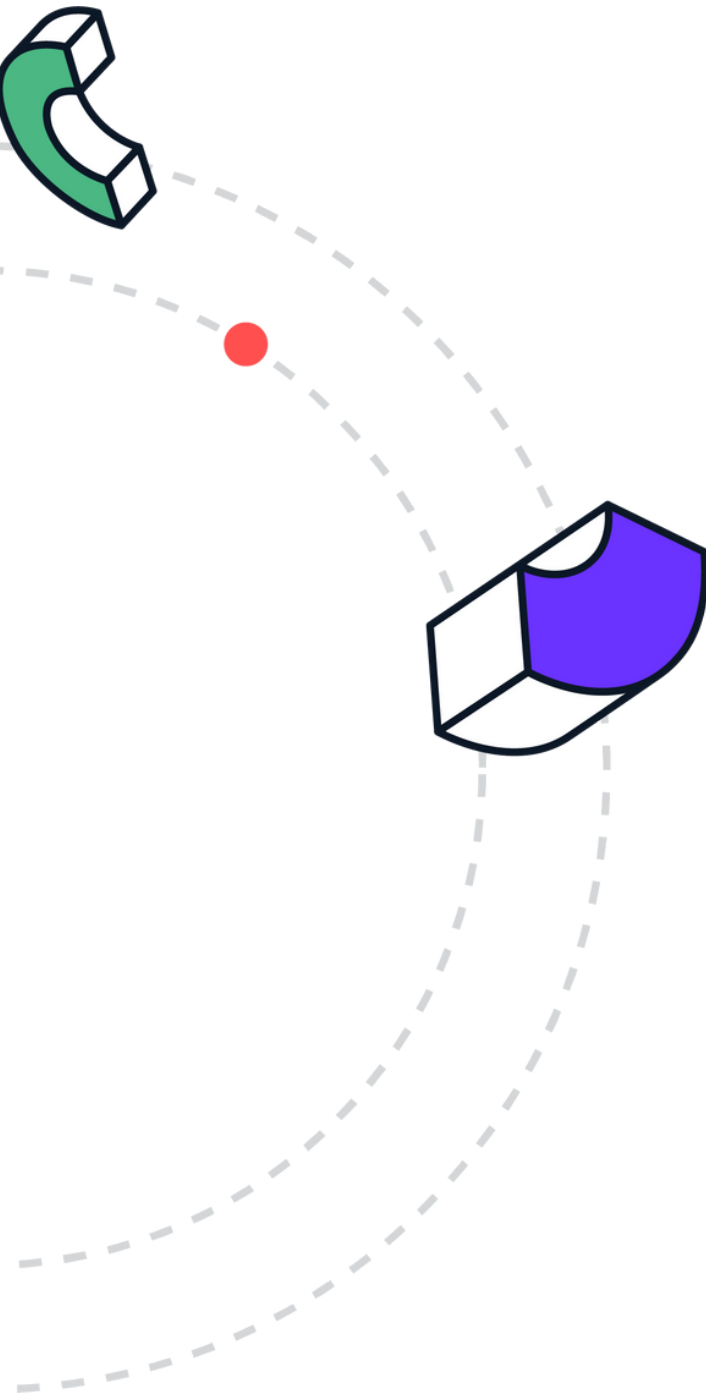


# Data Validation

## Your Options

This is a challenging area for any IT shop and the lack of technical knowledge in your test teams may be an issue with any of the following techniques:

1. Identify every table which could be affected during a test. It is important to differentiate between those that could be affected and those that should be affected otherwise collateral impacts will be missed.
2. Create numerous and complex SQL statements to ensure that every field in every table contains the expected data. This can also be achieved by creating a custom suite of validation programs.
3. Extend the step above to ensure that summary information is correctly based on the total value of supporting rows.





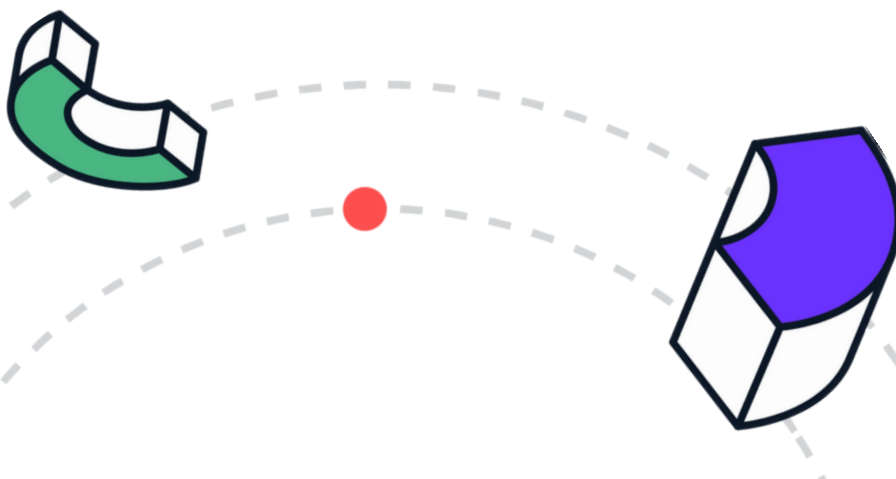
# Data Validation

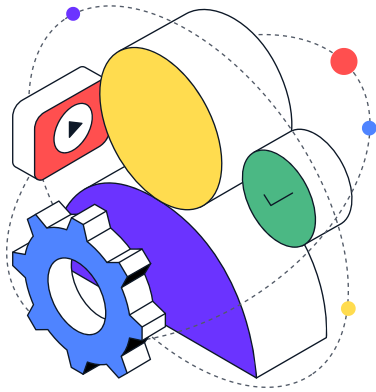
## How TestBench does it...

TestBench monitors every update to every table and data area, including multiple updates to a single row, so that it is easy to see at what point in a test a data row became corrupted. This can be a large volume of information, so TestBench provides 3 key analysis techniques.

1. Data Rules can be created that specifies the expected state of each field after every update, dependent upon its status.
2. TestBench can compare every element of the database activity monitored during a test to a baseline set of results and highlight the differences.

A static comparison can be run of the resultant file against an expected version of that same file and the differences highlighted. A similar comparison is available for reports.





# Mask, Use, Check, Reset & Re-use

In summary, the creation and management of test data environments is a challenge but let us finish this paper by reviewing the key steps to success:

## Create

- Create a representative linked subset of live data
- Ensure referential integrity is respected

## Mask

- Identify every column containing sensitive data.
- Mask that data while maintaining meaning for the user

## Use

- Ensure the test executes in the correct environment
- Consider UI, batch & API testing

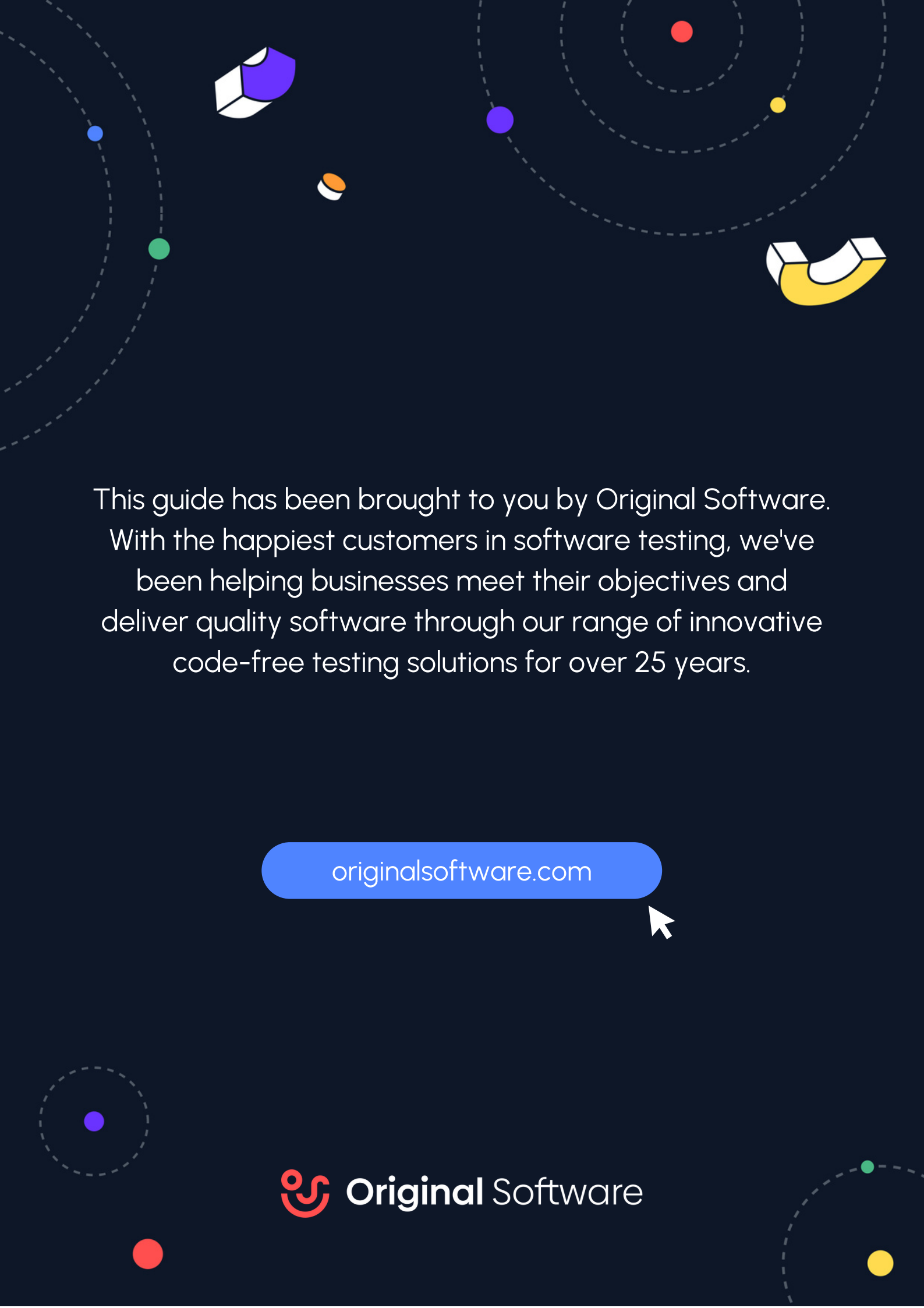
## Validate

- Ensure every action on every column in every row is correct
- Ensure parent/child summaries are correct

## Reset

- Set multiple Checkpoints as critical steps are achieved
- Reset rapidly every element of your test data environment.



The background is a dark navy blue. It features several abstract elements: dashed white lines forming concentric circles and arcs; solid colored dots in blue, green, red, yellow, and purple; and 3D geometric shapes including a purple and white cube, an orange and white sphere, and a yellow and white curved block.

This guide has been brought to you by Original Software.  
With the happiest customers in software testing, we've  
been helping businesses meet their objectives and  
deliver quality software through our range of innovative  
code-free testing solutions for over 25 years.

[originalsoftware.com](https://originalsoftware.com)



**Original Software**

