

WHITE PAPER

THE HIDDEN COSTS OF NEGLECTED TEST DATA MANAGEMENT

How poor practices and workarounds cost
\$420,000 and how to fix them.

Savings in percentage

80%

Including the costs of subsetting

Savings in dollars

\$360,000

per year, conservatively

TEST DATA SIMPLIFIED

Cut test data preparation time by up to 90%

From 4 hours to 20 minutes

TABLE OF CONTENTS

EXECUTIVE SUMMARY THE COST OF DOING NOTHING	04
1. THE ROOT CAUSE: WHY TEAMS STOP WHEN FRICTION BECOMES A HABIT	06
2. QUANTIFYING THE HIDDEN COSTS OF WORKAROUNDS THE NUMBERS BEHIND THE NOISE	07
3. THE SOLUTION: SUBSETTING AND VIRTUALIZATION RESETTING FROM HOURS TO MINUTES	13
4. RETURN ON INVESTMENT A CONSERVATIVE CASE FOR CHANGE	17
5. CONCLUSION THE QUESTION IS HOW LONG YOU WAIT	22
ABOUT THE AUTHOR	23

EXECUTIVE SUMMARY

Poor test data management (TDM) is one of the most underestimated sources of waste in software delivery.

When resetting a test database takes 4–6 hours, teams simply stop doing it. In its place, a web of workarounds emerges: testers search for “close enough” data, developers recreate scenarios by hand, chain tests get skipped, and bugs slip through to production.

This paper quantifies the true cost of that behavior for a team of 30. The numbers are based on conservative, low-end assumptions, and yet they add up to over \$421,800 per year.

COST COMPONENT	TIME LOST PER TEST CASE	ANNUAL COST (30-PERSON TEAM)
Searching for test data	45 min	\$280,800
Manually creating data	40 min	(included in total) 12 min
Rework from incomplete tests	12 min	\$65,000
Production incidents	-	\$40,000
Release delays	-	\$36,000
TOTAL ANNUAL COST OF POOR TDM	~1.5 HRS / TEST CASE	\$421,800 / YEAR

The solution is not only a technical one. It is also a process and culture change. Database subsetting reduces reset time from hours to 20 minutes. When combined with virtualization, resets drop to seconds. But technology alone is not enough: teams need to adopt resetting as a natural part of their workflow. When they do, nearly all workarounds disappear and over \$359,400 in annual waste is recovered.

Key finding:

Switching to subsetting reduces annual TDM-related costs from \$421,800 to \$62,400. A saving of \$359,400 per year, on conservative assumptions.

1. THE ROOT CAUSE: WHY TEAMS STOP RESETTING

In most organizations, testing and acceptance still rely on full copies of production databases. Resetting such an environment requires restoring that entire dataset - a process that typically takes 4 to 6 hours for the technical execution alone.

When scheduling, coordination, and infrastructure dependencies are included, the total elapsed time increases significantly. As a

result, many organizations delay refreshes - sometimes for years. The consequence is predictable. Because resets or refreshes are perceived as difficult and costly and disruptive, teams avoid them. Instead, they improvise. And that improvisation, repeated thousands of times across an organization, becomes far more expensive than the reset they were trying to avoid.

1.1 THE WORKAROUNDS THAT FILL THE GAP

When test data cannot be reliably reset, teams develop their own coping strategies, at the office often called workarounds:

- Searching for “roughly comparable” data in a polluted database
- Manually rebuilding test scenarios screen by screen
- Executing tests partially, skipping edge cases and chain tests
- Debugging against data that no longer reflects the intended state
- Developers creating stubs to work around missing or unusable data

These workarounds feel pragmatic in the moment. In reality, they are the most expensive things a QA team can do. However, the costs of these workarounds often stay hidden.

This paper sheds a light on the work arounds, their costs and how work can be organized differently. Saving both time and money.

2. QUANTIFYING THE HIDDEN COSTS OF WORK AROUNDS

The costs of poor TDM fall into two categories: operational waste (time lost to workarounds) and quality losses (defects found late, incidents, and delays). Both are measurable and both are avoidable.

To make this tangible, we quantify these costs using a simple model based on a set of clearly defined assumptions.

All assumptions in this model are intentionally conservative. We use a reset frequency of twice per week per employee, a blended hourly rate of \$65, and rework rates at the low end of what industry research shows. In practice, both the frequency and the rates are typically higher.

2.1 OPERATIONAL WASTE: TIME LOST TO WORKAROUNDS

Three structural workarounds account for the bulk of operational waste. Each has been modeled based on desk research, industry standards (ISTQB, TMAP, Capers Jones, IBM), and direct customer experience.

In practice, these workarounds take three recurring forms:

Searching for comparable test data

Manually creating test data

Executing incomplete tests due to data limitations

Each of these contributes measurable time loss or quality impact, as detailed below.

2.1.1 WORKAROUND 1: SEARCHING FOR COMPARABLE TEST DATA

When a full reset is not feasible - due to time, cost, or operational constraints - teams are forced to work with whatever data is already present in the environment. However, that data is often no longer reliable. Previous tests may have altered it, invalidated key scenarios, or left it in an inconsistent state.

As a result, testers have to search for data that “closely enough” approximates the scenario they need. This means navigating large and complex datasets, writing queries, comparing multiple records, validating against test scripts, and often dealing with access restrictions or missing context.

What seems like a small task quickly turns into a time-consuming and repetitive effort - as reflected in the time ranges in the table below.

ACTIVITY	TIME RANGE	MIDPOINT
Writing queries, filtering, checking logs	15–30 min	
Comparing 2–4 datasets	10–15 min	
Validating against test script	5–10 min	
Coordination / waiting / access issues	5–10 min	
TOTAL: SEARCHING FOR TEST DATA	40–55 MIN	45 MIN

For end-to-end chain tests, this search can take 1–3 hours. The 45-minute midpoint used here is deliberately modest.

Industry research shows that a substantial share of testing effort (often 30–50%) is spent on test preparation, including test data and environment setup.

2.1.2 WORKAROUND 2: MANUALLY CREATING TEST DATA

When no suitable data exists, testers and developers often recreate the required scenario by hand. This means navigating through application screens, filling in fields, creating linked entities, and correcting validation errors until the desired state is reached.

The problem is not just the time involved, but the fact that this approach does not scale in complex enterprise environments. In sectors such as banking, insurance, and pensions, realistic test scenarios depend on years of historical data, process context, linked records, and specific business states. These conditions cannot easily be recreated manually - yet teams still attempt to do so because they have no practical alternative.

As a result, what seems like “just creating a bit of data” often takes far more time than expected.

The table below reflects the effort required to manually create the data needed for a single business test scenario or test case.

ACTIVITY	TIME RANGE
Navigating through screens	3–5 min
Manually filling in fields/pages	10–20 min
Creating sub-entities (orders, contracts)	10–15 min
Correcting validation errors	5–10 min
TOTAL: MANUAL DATA CREATION	28–50 MIN

In enterprise environments (ERP, insurance, banking), manual data creation routinely takes 1–2 hours.

The 28-50 minute range used here reflects simple-to-moderate complexity.

2.1.3 WORKAROUND 3: INCOMPLETE TEST EXECUTION

When the data is not right, tests get shortened. Edge cases are dropped. Regression suites become patchy. The cost here is indirect: defects that should have been caught in testing are instead found in production.

Based on research by Capers Jones, IBM Systems Sciences, and TMAP, incomplete test data increases defect leakage by 15 - 40%.

We apply a conservative figure: 20% extra rework on a 60-minute average test case.

60 minutes × 20% = **12 minutes of extra rework per test case**

2.2 TOTAL OPERATIONAL WASTE PER TEST CASE DUE TO WORKAROUNDS

WORKAROUND	TIME LOST PER TEST CASE
Searching for comparable data	45 min
Manually creating test data	40 min
Extra rework from incomplete tests (20%)	12 min
TOTAL WASTE PER TEST CASE	97 MIN (~1.5 HOURS)

Annual calculation for 30 employees:
1.5 hrs × 2 situations/week × 30 employees × \$65/hr × 48 weeks = **\$280,800 per year**

Each test case that requires a workaround cost approximately 1.5 hours of wasted effort. At just 2 occurrences per employee per week, this compounds rapidly.

2.3 QUANTIFYING DOWNSTREAM COSTS: DEFECTS, INCIDENTS AND DELAYS

Beyond the operational waste, poor test data degrades software quality in ways that generate their own operational costs downstream. In practice, these costs manifest in three areas:

1. Increased defect leakage and rework
2. Production incidents and their associated business impact
3. Delays in delivery and reduced release confidence

Each of these effects can be quantified and traced back to limitations in test data.

2.3.1 DELAYED DEFECT DETECTION

When test data is incomplete or unrepresentative, defects escape testing and surface later in the cycle. Late-stage defects are consistently more expensive to fix than early ones. Based on conservative assumptions:



30 employees,
with 70% involved in
testing/quality work



3 hours per day
on test or quality
activities



20% rework rate
(low end of industry
literature)

$3 \text{ hrs/day} \times 200 \text{ days} \times 30 \text{ employees} \times 70\% \times 20\% \text{ rework} = \sim 1,000$
hours extra rework/year

$1,000 \text{ hours} \times \$65 = \text{\$65,000 per year}$

2.3.2 PRODUCTION INCIDENTS

When chain tests are incomplete and data is not representative, defects reach production. According to Gartner and Forrester, production incidents cost between \$10,000 and \$250,000 each.

This model uses a conservative \$40,000 per year equivalent to a single modest incident.

2.3.3 RELEASE DELAYS

Incomplete or unavailable test data disrupts the flow of testing. Test cases cannot be executed as planned, scenarios are delayed, and teams are forced to wait, rework, or reprioritize. Because of these disruptions, sprints are delayed.

Or as a result, work frequently spills over into the next sprint. Creating longer or extra sprints. What should have been completed within a sprint is deferred, creating cascading

delays across teams and environments. In some cases, this leads to additional regression cycles or even an extra sprint to complete the planned scope.

We do not assume that every missed or delayed sprint is caused by test data. Industry practice shows that incomplete sprint delivery is common, but in this model, we apply a much more conservative assumption: one test-data-related sprint delay per quarter, so 4 delays.

And even one sprint slip per quarter is costly:

$4 \text{ delays/year} \times 1\text{--}3 \text{ days} \times \$3,000\text{--}5,000/\text{day} = \textbf{\$36,000 per year}$

2.4 TOTAL HIDDEN ANNUAL COST OF POOR TDM

COST COMPONENT	ANNUAL COST
Operational workarounds	\$280,800
Extra rework (quality losses)	\$65,000
Production incidents (min. 1/year)	\$40,000
Release delays	\$36,000
TOTAL ANNUAL COST OF POOR TDM	\$421,800

3. THE SOLUTION: SUBSETTING AND VIRTUALIZATION

The underlying problem is simple: resets take too long, so teams avoid them. The solution is equally straightforward: make resets fast enough that teams want to use them.

Two complementary techniques achieve this: database subsetting (which reduces volume) and database virtualization (which eliminates restore time altogether). Together they transform test data resets from a multi-hour event into something that takes seconds.

3.1 DATABASE SUBSETTING: FROM HOURS TO 20 MINUTES

A subset contains only the data relevant to testing, typically less than 10% of the production database size.

A 4 TB database becomes a 400 GB subset. Because restore time scales almost linearly with data volume, **that 10x size reduction translates directly into a 10x speed improvement.**

Conservative calculation based on enterprise SAN environments (approximately 350 MB/s effective I/O throughput during database restore operations):

Full 4 TB restore:

3 hours 20 minutes

Subset 400 GB restore:

19.5 minutes

Application reconnection and warm-up:

25 minutes

Without subsetting

3 hours 20 minutes

10x faster



With subsetting

19.5 minutes

Total practical reset time with subsetting: 20–25 minutes
minutes Modern environments with higher throughput (600 MB/s 1+ GB/s) achieve even faster results. This estimate is deliberately conservative.

WHY 20 MINUTES CHANGES EVERYTHING

The behavioral shift is the key insight. A **4-hour** reset requires planning, coordination, and a dedicated maintenance window. A **20-minute** reset fits within a morning, a sprint cycle, or even a single debugging session.

Once the friction is low enough:

- Teams reset as part of their normal workflow
- Eliminates day-to-day dependency on DBAs and infrastructure teams
- Chain tests, regression tests, and debugging all become consistent
 - Workarounds disappear because they are no longer necessary

3.2 DATABASE VIRTUALIZATION: FROM MINUTES TO SECONDS

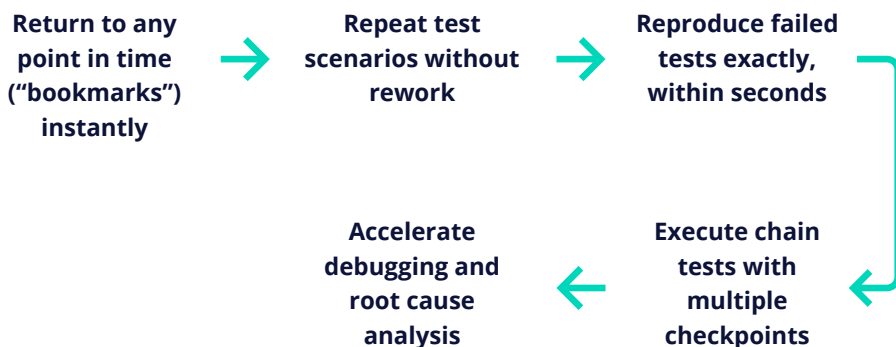
Virtualization takes this further. Rather than copying data, a virtualized database uses copy-on-write snapshots and pointer switching. There is no restore, no full copy, no heavy I/O. A rollback is literally moving from a pointer to an earlier snapshot - an operation that takes 5 to 20 seconds.

However, virtualization alone is not enough. Virtualizing full production-sized databases (often

multiple terabytes) still introduces significant overhead in storage, management, and performance.

The real advantage emerges when virtualization is combined with intelligently defined data subsets. By working with smaller, scenario-specific datasets, teams can fully leverage the speed and flexibility of virtualization without the burden of large-scale data volumes.

This fundamentally changes how teams work with test data:



3.3 WHY BOTH ARE NEEDED

Subsetting and virtualization solve different parts of the same problem:

	SUBSETTING ONLY	SUBSETTING + VIRTUALIZATION
Reset time	~20 minutes	5–20 seconds
Reproducibility	Improved, not instant	Exact replay, any point in time
Team autonomy	High	Complete

Subsetting alone solves the speed problem

-> but leaves reproducibility incomplete.

Virtualization alone removes restore time

-> but does not address large data volumes.

Only the combination of both eliminates these limitations entirely.

4. RETURN ON INVESTMENT

The previous sections quantified what poor test data management costs in its current state: over \$421,800 per year for a team of 30, driven by workarounds, late defect discovery, production incidents, and release delays.

This section translates that baseline into a concrete return on investment.

The ROI case is straightforward. Subsetting reduces reset time

from hours to 20 minutes — short enough that teams actually use it. That single change eliminates nearly all the waste calculated above. The only remaining cost is the time spent during the reset itself.

The model below uses conservative assumptions throughout. The calculated savings represent a floor, not a ceiling.

4.1 COST MODEL AND ASSUMPTIONS

The ROI calculation in this document is intentionally conservative. All values are based on low-end assumptions to avoid overstating the business case. The model uses the following parameters:



Relatively low test frequency

(2 resets per week, while 5–20 is common in practice)

\$65

Average blended hourly rate

48

working weeks per year

30

employees
involved in testing, development, analysis, or debugging

20%

Minimal rework percentages
(20% even though most literature shows higher values)



Modest chain complexity

\$

Minimal incident costs

This conservative foundation ensures that all calculated savings can be achieved even in organisations with relatively simple test landscapes or modest test volumes.

4.2 ROI SCENARIO: CURRENT STATE

In the current state, organisations rely on full-size database copies that take **4-6 hours** to refresh. Because resets are slow, disruptive and require coordination across multiple teams, they are performed infrequently, sometimes not for months or even years.

This leads to workarounds:

- search effort for “usable” test data
- manual data creation
- incomplete test execution
- polluted data
- late defect detection
- repeated debugging and retesting
- skipped chain tests

These patterns create substantial hidden costs:

Operational waste:

\$280,800 per year

Quality-related losses:

\$103,000 per year

Total annual cost of poor TDM:

\$421,800

This represents the baseline cost of maintaining the status quo.

4.3 ROI SCENARIO: SUBSETTING (20-MINUTE RESETS)

Subsetting reduces the database to roughly 10% of its original size, decreasing reset time from hours to approximately **20 minutes**. This is short enough to fit within a testing session, a sprint, or even the same morning.

This behavioural shift eliminates nearly all workarounds:

- **Testers no longer search for comparable data**
- **Manual test data creation stops**
- **Tests become complete and repeatable**
- **Debugging becomes cleaner and faster**
- **Defects surface earlier in the cycle**

As a result, nearly all structural waste is removed. The only remaining cost is the short pause during a 20-minute reset.

4.3.1 UNDERPINNING CALCULATION OF \$124.800

The \$124.800 annual cost for the subsetting scenario is calculated as follows: **Assumptions**

20 minutes (0.33 hours) Reset duration

2 times per week, Reset frequency

30 Employees affected

\$65 Hourly rate

48 Working weeks per year

Annual reset cost

$0.33 \text{ hours} \times 2 \times 30 \times \$65 \times 48 = \$124.800$

This represents the complete operational cost of resetting in a subsetting-only solution. All other forms of waste are eliminated.

SCENARIO	ANNUAL COST	SAVINGS VS. CURRENT STATE	CONTEXT
Current state (full copy, rare resets)	\$421,800	-	Workarounds, late defects, incomplete tests
Subsetting (reset in ~20 min)	\$62,400	\$359,400 saved	Workarounds eliminated, tests complete

Subsetting cost calculation

$0.33 \text{ hrs} \times 2 \text{ resets/week} \times 30 \text{ employees} \times \$65/\text{hr} \times 48 \text{ weeks} = \$62,400/\text{year}$

Fully reproducible tests across teams and environments

- Eliminate DBA and infrastructure dependencies for routine resets
- Shorten release cycles and reduced coordination overhead
- Contribute to a higher team autonomy and morale

5. CONCLUSION

The hidden costs of poor test data management are not a QA problem. They are a business problem. Every hour a tester spends searching for usable data, every chain test that gets skipped, every defect that slips into production - all of it traces back to one root cause: resets take too long, so teams stop doing them.

The fix is technical, AND organizational. Database subsetting reduces a 4-to-6-hour

reset to 20 minutes. That single change is sufficient to eliminate the workaround behavior that costs \$421,800 per year. With virtualization on top, resets drop to seconds, and reproducibility becomes complete.

The numbers in this paper are conservative by design. They represent the floor of what most organizations are losing. The actual savings, and the actual opportunity, are typically larger.

The question is not whether better test data management delivers ROI. It does, reliably and measurably. The question is how long an organization can afford to wait.



Learn more about DATPROF Subset:

<https://www.dataprof.com/products/datprof-subset/>



Learn more about DATPROF Virtualize:

<https://www.dataprof.com/products/datprof-virtualize/>

ABOUT THE AUTHOR

Maarten Urbach is Chief Sales Officer at DATPROF, where he leads initiatives to help organizations optimize their test data management practices. With extensive experience in software testing, data privacy, and enterprise IT environments, Maarten has a deep understanding of the operational and quality challenges organizations face when managing test data.

In this white paper, he draws on both industry research and real-world customer experience to highlight the hidden costs of poor test data management and the practical strategies to address them.



Maarten Urbach
– DATPROF

Connect with
Maarten Urbach
scan the QR



Or search:

<https://www.linkedin.com/in/maartenurbach/>

