THE RAVIT SHOW

Data Observability, The Reality

Understand the why and how!



Foreword by:

Bill Inmon,
Father of the Data Warehouse

MONTE CARLO · Bigeye · Databand an IBM Company

Anomalo · unravel · DQLABS · TELMAI

Qualdo™ · RightData · acceldata

Data Observability, The Reality

Understand the why and how!

Foreword by:
Bill Inmon,
Father of the Data Warehouse



All rights reserved with The Ravit Show. No part of this book may be reproduced or used in any manner without the prior written permission of the copyright owner, except for the use of brief quotations in a book review.

Author/Company Information

Monte Carlo

Barr Moses -- Barr Moses is CEO & Co-Founder of Monte Carlo, a data reliability company and creator of the data observability category, backed by Accel, GGV, Redpoint, ICONIQ Growth, Salesforce Ventures, IVP, and other top Silicon Valley investors.

Lior Gavish -- Lior Gavish is CTO and Co-Founder of Monte Carlo, a data reliability company backed by Accel, Redpoint Ventures, GGV, ICONIQ Growth, Salesforce Ventures, and IVP.

Molly Vorwerck -- Molly Vorwerck is the Head of Content & Communications and a founding team member at Monte Carlo, a data reliability company.

Bigeye

Kyle Kirwan -- Kyle Kirwan is the CEO and co-founder of Bigeye, which helps data teams build trust in the data they depend on. As one of the first data scientists, data analysts, and product managers at Uber, he helped launch teams like Experimentation Platform, and products like Databook.

Databand, an IBM Company

Ryan Yackel -- Ryan is a Product Strategy Leader for Databand, an IBM Company. He's passionate about solving data quality and reliability problems through data observability. Before IBM, Ryan led GTM programs at high-growth startups in software test automation, DevOps, and IAM cybersecurity.

Anomalo

Elliot Shmukler -- Elliot Shmukler is CEO and co-founder at Anomalo. Before Anomalo, Elliot was the Chief Growth Officer at Instacart, where he led the company through a massive period of change and hyper-growth. Before that, he held senior product and growth leadership roles at fast-growing companies like Wealthfront and LinkedIn.

Paige Schwartz -- Paige Schwartz is a professional technical writer at Anomalo who has worked with clients such as Airbnb, Grammarly, and Samsara, as well as successful startups like CodeSignal, Tecton, Clerky, and Fiddler.

Unravel

Kunal Agarwal is the Co-Founder and Chief Executive Officer of Unravel Data. Previously, he led sales and implementation of Oracle products at several Fortune 100 companies. Kunal also co-founded Yuuze.com, a pioneer in personalized shopping and what-to-wear recommendations.

DQLabs

Raj Joseph -- Mr. Raj Joseph is a thought leader and innovative disruptor in the field of Modern Data Management and Enterprise-wide Data Quality Initiatives. Mr. Joseph is the CEO and founder of DQLabs.ai – a Modern Data Quality platform that harnesses the combined power of Data Observability, Data Quality, and Data Discovery to enable data producers, consumers, and leaders to turn data into action faster, easier, and more collaboratively.

Telmai

Mona Rakibe -- Mona Rakibe is the Cofounder and CEO of Telmai. She is a thought leader and has spent years in the Data field. Telmai is a real-time data observability platform — that can automatically and proactively detect issues as data is getting ingested at the attribute value level.

Qualdo

Muralidharan Krishnamoorthy is the Founder & CEO of Saturam Inc., a fast-growing advanced data engineering startup with its flagship data reliability product - Qualdo.ai. He is an acknowledged thought leader in data engineering and machine learning and an experienced entrepreneur.

Rightdata

Sridhar Bankuru -- Sridhar Bankuru serves as Vice President of Software Development for Data Quality at RightData Inc. An experienced data engineering executive, Sridhar has a comprehensive background in development, architecture, and implementation for challenging data systems and projects.

Acceldata

Rohit Choudhary -- Rohit Choudhary is the CEO and Co-Founder of Acceldata, a startup whose multidimensional Data Observability Cloud enables enterprises to observe and optimize modern data systems, maximize the return on data investments, increase data trust, and use data as a competitive advantage.

Joe Reis -- Joe Reis is a business-minded data nerd who's worked in the data industry for 20 years, with responsibilities ranging from statistical modeling, forecasting, machine learning, data engineering, data architecture, and almost everything else in between. Joe is the CEO and cofounder of Ternary Data, a data engineering and architecture consulting firm based in Salt Lake City, Utah.

Sanjeev Mohan -- Sanjeev is an established thought leader in the areas of cloud, big data, and analytics. He researches and advises on changing trends and technologies in the modern cloud data architectures. He started his data and analytics journey at Oracle, where he worked on emerging technologies. Until recently, he was a Gartner vice president known for his prolific and detailed research, and for directing the data and analytics agenda. Now a principal at SanjMo, he provides advisory and consulting services, covering modern data stacks, quality, security, governance, observability, and operations.

Acknowledgment

Writing a book is a monumental task, and I could not have done it without the support and encouragement of so many wonderful people.

Firstly, I want to express my heartfelt thanks to my family and friends for their unwavering support and encouragement throughout this journey. Your belief in me has been a constant source of motivation and inspiration, and I am grateful to have you in my life.

This book wouldn't be possible without all the Founders and Thought Leaders who generously shared their expertise and knowledge, thank you for your contributions to this book. Your insights and experiences have been invaluable, and I am honored to have had the opportunity to learn from you.

I would like to extend my sincere appreciation to my editors and designers, who worked tirelessly to bring this book to life. Your skill and dedication have been instrumental in shaping the final product, and I am grateful for your contributions.

I would also like to acknowledge the many individuals and organizations who provided feedback, reviewed drafts, and supported me throughout the process. Your insights and encouragement were critical in helping me to refine and improve this work.

Finally, I want to express my gratitude to my readers, who have invested their time and attention in this book. I hope that it provides value and insight and that it contributes to the ongoing conversation around data observability.

Thank you all for your support and encouragement. I am humbled and grateful for each and every one of you.

Foreword

Data observability is growing in importance every day. In every corporation.

The problem is that data is voluminous and complex. Making data observable can be an expensive and time-consuming thing to do.

I recommend Ravit's book as a good first step in unraveling the challenges of achieving data observability

-- Bill Inmon
Father of the Data Warehouse

TESTIMONIALS

"Data Observability might seem like another buzzword. Data Observability, the Reality demystifies the concept in an easy-to-understand way offering readers practical explanations for WHY everyone needs to consider data observability for their data ecosystems. WHAT it is, hint it's not just monitoring and alerting. The authors do a great job helping us understand HOW to implement it as well as warning us about how not to fail at it. This book is a great addition to everyone's virtual or real library and a must-read for all data professionals"

-- Veronika Durgin, VP of Data, Saks

Data Observability is a must-read for anyone interested in mastering the art of monitoring data pipelines. The book provided me with many practical insights and real-world experiences from people in the field. It's a great place to start learning about data observability in an easy-to-read format. Highly recommended!

-- Tobias Zwingmann Managing Partner of RAPYD.AI

"Would you deploy a service in production without using an observability tool like Datadog? The answer is no; that's why we should deploy data products with just as much observability. This book covers several great history lessons, best practices, and guidance on helping move data observability forward in your organization. Must read!"

-- Tommy DANGerous, Founder & CEO, Mage

Data Observability is the BEST resource on data observability and pipelines! With insights from heavy hitters from around the industry, including Bill Inmon and Joe Reis, this is a must-read for anyone interested in the emerging field of data observability. 10/10 would recommend it!

-- Carly Taylor, Machine Learning @ Call of Duty, Activision

Table of Contents

Chapter 1: Why book on Data Observability? 1

Data observability can help data professionals in several ways: 1

Chapter 2: The Past, Future, and Present of Data Observability 5

The Origins of Data Observability 7

The State of Data Observability Today 8

Incident Detection 10

Response 11

Root Cause Analysis & Resolution 11

Blameless post-mortem 13

The Future of Data Observability 14

So, Where Do We Go From Here? 15

Chapter 3:The Basics of Data Observability 17

Our dependence on data pipelines 17

Data outages 18

What is data observability? 18

What does data observability help you do? 19

Who works with data observability? 19

What use cases does data observability help with? 21

Analytics 21

Machine learning 21

Data model development 22

Data ingestion, replication, and integration 22

Data warehouse or database migrations 22

What are the core concepts? 23

Instrumentation 23
Metrics 23
Events 23
Logs 24
Lineage 24
Monitoring 24
Alerting 25
How do data observability tools enable data observability? 25
Metadata monitoring 25
Anomaly detection 25
Automation 26
Flexibility and scale 26
Summary 26

Chapter 4: Understanding Data Quality and Data Observability 29
Data Quality Key Performance Indicators(KPI) 30
Completeness 31
Uniqueness 31
Timeliness and Freshness 32
Validity 32
Accuracy 33
Consistency 34
Role of Data Lineage 34
Why do traditional data quality tools fail in the modern data stack? 35
Data Observability as the foundation of Data Quality. 36
Conclusion 37

Chapter 5: The Convergence of Data Quality and Data Observability 39

Chapter 6: Data Products Need Deep Data Observability 47

Data products and the hierarchy of data needs 48
The inevitability of data quality issues 49
Foundational data observability vs. deep data observability 49
Achieving deep data observability 50
Validation rules 51
Metrics monitoring 52
Deep data observability with ML 52
Getting started 53

Chapter 7: Real-Life Data Observability – Beyond Data Downtime 55
Real-Life Data Observability 55
But What About Data Downtime? 56
Thinking About Data Observability on a Holistic Basis 57
What's Data Observability Without Validation? 58
Data Governance is Critical to Data Observability 58
These include: 59
Putting It All Together for Data Observability 59
What's Needed Going Forward 60

Chapter 8: A Framework for Proactive Data Observability 61
Step 1: Understand Pipeline Execution Health 62
Step 2: Alert on Pipeline Latency 63
Step 3: Check Data Sanity 64
Step 4: Analyze Data Trends 65
Step 5: Identify Data Access and Business Impacts 66
Top Data Quality Metrics for Data Observability 67
Metric 1: Null counts 68
Metric 2: Schema changes 68
Metric 3: Data lineage 71

Metric 4: Pipeline failures 72
Metric 5: Pipeline duration 73
Metric 6: Missing data operations 74
Metric 7: Record count in a run 74
Metric 8: Tasks read from data set 75
Metric 9: Data freshness 76
Get started 76

Chapter 9: Automation and AI Are a Must for Data Observability 77
Let's think differently about data observability 79
What are the 4 foundational areas of data observability? 80
How data observability solves business problems 81
The unique challenges for data observability 84
What data observability should look like 84
Conclusion 86

Chapter 10: How Data Observability Optimizes Reliability, Performance, Operations, and Spend for the Modern Data Stack 89
The Data Stack Evolution 90
Data Problems Result From Quality Deficits 91
Legacy Data Monitoring Approaches Are Insufficient 91
Data is Data, But "Good" Data is Essential 92
Data Reliability 94
Spend Intelligence 96
Operational Intelligence 97
Data Observability Across the Entire Data Lifecycle 98

Chapter 11: Raising the Bar: Triangulating Observability, Quality & Reliability for all Data Stakeholders 101
Evolving expectations of data stakeholders 102
Why should you care about data quality in the cloud? 102

Who needs data observability the most? 103
Who suffers the most when data reliability is not guaranteed? 103
The interplay between data quality, data observability, and data reliability 104
Unifying data quality, data observability, and data reliability 104
Introducing a holistic view for multiple data stakeholders within an enterprise 106
The problem with SLAs 106
It takes three to tango 107
The path of our predecessors 107
Raising the bar 107

Chapter 12: Data Observability Anti-Patterns 109
Don't Get Buy-In From Stakeholders 109
Best Selling Co-author of Fundamentals of Data Engineering 109
Practice Data Observability Theater 110
Your Incident Response Shouldn't Fix the Root Cause 111
Here's to failure! 112
Links 112
Notes 113

Chapter 13: Data observability's newest frontiers: DataFinOps and DataBizOps 115
Principal, SanjMo & Former Gartner Research VP, Big Data and Advanced Analytics 115
What is DataFinOps? 116
Why do we need DataFinOps? 117
Challenges of DataFinOps 118
DataFinOps Case Studies 119
Introducing DataBizOps 121
Summary 122

Why book on Data Observability?

Chapter 1



Data observability can help data professionals in several ways:

Data observability is a crucial yet often overlooked aspect of managing data at scale. As the volume and complexity of data continue to grow, the need for effective monitoring and observability becomes increasingly important to ensure data quality, accuracy, and reliability. As a result, I felt compelled to write a book on data observability to share my knowledge and experience in this field, and to help organizations understand the importance of data observability and how to implement it successfully in their operations. Through this book, I hope to help readers gain a deeper understanding of the challenges and opportunities of data observability, and provide practical insights and strategies for achieving observability in their own data pipelines.

Monitoring and analyzing system performance: Data observability tools can collect, store and analyze data about a system's performance, such as resource utilization, response times, and error rates. This information can help data professionals identify bottlenecks and optimize system performance.

Debugging and troubleshooting: Data observability tools can provide detailed information about a system's behavior and state, which can be useful for identifying and resolving issues. Data professionals can use this information to diagnose and fix problems more quickly and efficiently.

Root cause analysis: Data observability can provide detailed information about the sequence of events that led to a problem, which can be helpful in determining the root cause of an issue.

Compliance and security: Data observability can help data professionals ensure that systems comply with regulatory requirements and industry standards. It can also provide visibility into system security, helping to detect and prevent potential threats.

Improving data quality: Data observability can help data professionals understand how data is being used, and how it is flowing and identify any data quality issues.

Optimizing Data pipeline: Data observability can be used to understand how data is flowing through the pipeline, identify any data quality issues and identify any bottlenecks that might be causing delays in the data processing.

Predictive Maintenance: Data observability can help data professionals to understand patterns of system usage, performance, and failure and use this information to predict when maintenance is likely to be needed and optimize the timing of maintenance.

Cost optimization: Data observability can help data professionals to understand the resource utilization of systems, and optimize the cost of running the systems by identifying and eliminating unnecessary resource consumption.

A few reasons why I thought the book on Data Observability would help the Data Community!

1. To share their expertise and knowledge on the topic with others in order to help educate and inform the industry.
2. To provide a comprehensive guide or resource that can be used as a reference by professionals working in the field.
3. To raise awareness of the importance of data observability and its role in ensuring the reliability, performance, and security of modern systems.

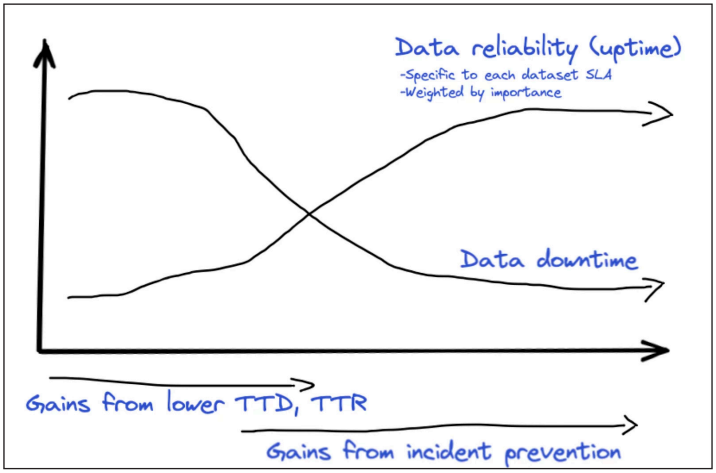
4. To provide practical advice and best practices for implementing data observability in real-world systems.
5. To showcase the latest advancements and trends in data observability and its application in various domains such as cloud, distributed systems, and more.
6. To provide a source of inspiration for further research and development in the field.
7. To provide a historical perspective on the evolution of data observability and its impact on the industry.
8. To explore the different approaches and techniques used for data observability, such as logging, monitoring, tracing, and alerting.
9. To discuss the challenges and limitations of data observability and possible solutions to overcome them.
10. To provide a comparison and evaluation of different data observability tools and frameworks currently available in the market.
11. To provide case studies and real-world examples of how data observability has been implemented and its impact on the performance and reliability of systems.
12. To provide a forward-looking perspective on the future of data observability and how it will continue to shape the industry.
13. To provide a guide to help organizations understand the importance of data observability and how to implement it for their systems.
14. To be a comprehensive guide for developers and engineers to understand the basics of data observability and help them to implement it in their systems.

We've discussed the growing importance of observability in the context of managing data at scale, and the need for effective monitoring and quality control to ensure data reliability. By providing practical insights and strategies for achieving observability in data pipelines, this book aims to empower readers to make informed decisions about their data and drive better business outcomes. In the following chapters, the amazing contributors have dived deeper into the fundamentals of data observability, explore different observability techniques and tools, and offer real-world examples of successful observability implementations. Stay tuned!

The Past, Future, and Present of Data Observability

Chapter 2

MONTE CARLO



Barr Moses, Lior Gavish, and Molly Vorwerck

If this scenario rings a bell, raise your hand (or spit out your coffee, sigh deeply, and shake your head).

Data is a priority for your CEO, as it often is for digital-first companies, and she is fluent in the latest and greatest business intelligence tools. Your CTO is excited about migrating to the cloud and constantly sends your team articles highlighting performance measurements against some of the latest technologies. Your downstream data consumers, including product analysts, marketing leaders, and sales teams rely on data-driven tools like CRMs, CXPs, CMSs, and any other acronym under the sun to do their jobs quickly and effectively.

As the data analyst or engineer responsible for managing this data and making it usable, accessible, and trustworthy, rarely does a day goes by without having to field some request from your stakeholders. But what happens when the data is wrong?

Bad data erodes trust, time, and resources from already strapped data engineering teams, with these fearless fixers of the data pipeline spending a whopping *40 percent of their time* on data quality issues. And in 2023, with budgets tightening and data workloads growing, the problem is only getting worse, only getting worse.

When we launched *Monte Carlo - and the data observability* category - in 2019, we didn't have a name for it, but we knew that the current approach to solving data quality issues (manual testing and hoping/praying) wasn't cutting it.

We also knew that by applying tried and true best practices from software engineering and DevOps - incident management, root cause analysis, automated detection and alerting, lineage, and setting SLAs and SLOs, to name a few - we could help teams prevent data incidents from becoming disasters. Although data and software have many differences, we knew there were enough similarities to codify our approach and apply it to what we believed was the next big crisis for data teams: data downtime, in other words, periods when data is missing, inaccurate, or otherwise erroneous.

We had no idea that data observability - our approach to solving *this expensive and time-consuming problem* - would resonate so broadly across the data engineering community.

Four years, hundreds of customers, and *an entire book* later, we knew we were onto something. In this chapter, we'll walk through some current and future trends informing this critical new layer of the data stack.

The Origins of Data Observability

Technical teams have been tracking — and seeking to improve — data quality for as long as they've been tracking analytical data, but only in the 2020s has data quality become a top-line priority for many businesses. As data becomes not just an output but a financial commodity for many organizations, it's important that this information can be trusted.

Companies are increasingly treating their data like code, applying frameworks and paradigms long-standard among software engineering teams to their data organizations and architectures. Developer Operations (DevOps), a technical field dedicated to shortening the systems development life cycle, spawned industry-leading best practices such as Site Reliability Engineering (SRE), CI/CD (continuous integration / continuous deployment), and microservices-based architectures. The goal of DevOps is to release more reliable and performant software through automation.

As organizations grew and the underlying tech stacks powering them grew more complicated (think: moving from a monolith to a microservices architecture), it became important for DevOps teams within the software engineering department to maintain a constant pulse on the health of their systems and deploy continuous integration and development (CI/CD) approaches.

Observability, a more recent addition to the software engineering lexicon, speaks to this need, and refers to the monitoring, tracking, and triaging of incidents to prevent software application downtime.

At its core, there are three pillars of software observability are:

- Metrics refer to a numeric representation of data measured over time.
- Logs, a record of an event at a given timestamp, also provide valuable context regarding when a specific event occurred.
- Traces represent causally related events in a distributed environment.

Taken together, these three pillars and application management (APM) solutions, like DataDog or Splunk, gave DevOps teams and software engineers valuable awareness and insights to predict future behavior, and trust their IT systems to meet SLAs.

Over the past five years, it became apparent that data engineering teams needed similar processes and tools to monitor their ETL (or ELT) pipelines and prevent data downtime across their data systems.

Since 2019, companies such as *Airbnb*, *Uber,* and *Netflix* have written prolifically about their commitment to ensuring reliable, highly available data for stakeholders across the business by applying DataOps best practices. In addition to powering analytics-based decision-making (i.e., product strategy, financial models, growth marketing, etc.), data produced by these companies powers their applications and digital services. Inaccurate, missing, or erroneous data can cost them time, money, and the trust of their customers.

As these tech behemoths increasingly shed light on the importance and challenges of achieving high data quality, other companies of all sizes and industries are taking note and replicate these efforts, from implementing more robust testing to investing in DataOps best practices like data observability.

The State of Data Observability Today

Understanding where we are today helps to begin with a formal definition. As originally defined, data observability is an organization's ability to understand the health of the data in their systems. Data observability eliminates data downtime by applying best practices learned from DevOps to data pipeline observability.

Data observability tools use automated monitoring, automated root cause analysis, field-level data lineage and data health insights to detect, resolve, and prevent data anomalies. This leads to healthier pipelines, more productive teams, better data management, and happier customers.

The five pillars of data observability are:

- Freshness
- Quality
- Volume
- Schema
- Lineage

Together, these components provide valuable insight into the quality and reliability of your data. Let's take a deeper dive.

- **Freshness:** Freshness seeks to understand how up to date your data tables are, as well as the cadence at which your tables are updated. Freshness is important in decision making, stale data is synonymous with wasted time and money.

- **Quality:** Your data pipelines might be in working order, but the data flowing through them could be garbage. The quality pillar looks at the data itself and aspects such as percent NULLS, percent uniques and if your data is within an accepted range. Quality gives you insight into whether your tables can be trusted based on what can be expected from your data.

- **Volume:** Volume refers to the completeness of your data tables and offers insights on the health of your data sources. If 200 million rows suddenly turn into 5 million, you should know.

- **Schema:** Changes in the organization of your data, in other words, schema often indicate broken data. Monitoring who changes these tables and when is foundational to understanding the health of your data ecosystem.

- **Lineage:** When data breaks, the first question is always "where?" Data lineage provides the answer by telling you which upstream sources and downstream ingestors were affected, as well as which teams are generating the data and who is accessing it. Good lineage also collects information about the data (also called metadata) that speaks to governance, business, and technical guidelines associated with specific data tables, serving as a single source of truth for all consumers.

A strong data observability approach will consider these key pillars and include these features, whether ad-hoc or as a centralized data observability platform:

- **It connects to your existing stack quickly and seamlessly** and does not require modifying your data pipelines, writing new code, or using a particular programming language. This allows quick time to value and maximum testing coverage without making substantial investments.

- **It monitors your data at rest and does not require extracting the data from where it is currently stored.** This allows the data observability solution to be performant, scalable and cost-efficient. It also ensures that you meet the highest levels of security and compliance requirements.

- **It requires minimal configuration and practically no threshold-setting.** Data observability tools should use machine learning models to automatically learn your environment and your data. It uses anomaly detection techniques to let you know when things break. It minimizes false positives by considering not just individual metrics but a holistic view of your data and the potential impact from any particular issue. You need not spend resources configuring and maintaining noisy rules within your data observability platform.

- **It requires no prior mapping of what needs to be monitored and in what way.** It helps you identify key resources, key dependencies and key invariants so you get broad data observability with little effort.
- **It provides rich context that enables rapid triage and troubleshooting** and effective communication with stakeholders affected by data reliability issues. Data observability tools shouldn't stop at "field X in table Y has values lower than Z today."
- **It prevents issues from happening in the first place** by exposing rich information about data assets so changes and modifications can be made responsibly and proactively.

Data observability isn't about the technology. An end-to-end approach to data trust and eliminating data downtime also requires cultural and process buy-in.

While not a ton of literature exists about how data teams can handle incident management for their data, there are already great resources we can leverage from *our friends in software development*. With a few adjustments, these tricks of the trade have become invaluable tools for some of the best data engineering teams.

For building a data incident management workflow for your pipelines, the 4 critical steps include incident detection, response, *root cause analysis (RCA)* & resolution, and a blameless post-mortem.

Incident Detection

You should first *test your data* before it enters production. Still, even with the most robust tests and checks in place, bad data will fall through the cracks and be pushed to prod before you can say "broken data pipeline."

When data downtime strikes, the first step is incident detection. Incidents can be detected through *data monitoring and alerting*, which can be implemented manually on your data pipelines and triggered based on specific thresholds or layered as part of *a data observability* solution and triggered automatically at regular intervals based on historical data patterns and custom rules.

One critical component of data monitoring is *anomaly detection*, or identifying when pillars of data health (i.e., volume, freshness, schema, and distribution) veer from the norm. Anomaly detection is most valuable when implemented end-to-end (across your warehouses, lakes, ETL, and BI tools) instead of only in a specific silo of your data ecosystem. Good anomaly detection will also tune algorithms to *reduce white noise* and false positives.

Incident detection can be integrated into data engineering and analytics workflows, ensuring all data stakeholders and end-users are paged when issues arise across the proper communication channels (Slack, email, SMS, carrier pigeon...)

Response

Good incident response starts — and ends — with effective communication, and fortunately, much of it can be prepared in advance and automated out through the proper workflows via PagerDuty and Slack when the time comes.

Data teams should write runbooks and playbooks that walk through standard incident response. While runbooks instruct you for how to use different services and common issues they encounter, playbooks provide step-by-step processes for handling incidents. Both will provide links to code, documentation, and other materials to help teams understand what to do when critical pipelines break.

One important part of a good runbook? Delegating roles when outages or breakages occur. In traditional site reliability engineering programs, an on-call process that delegates specific roles depending on service often segmented by hour, day, or week. In addition to an "*incident responder*," there is often an "incident commander" responsible for assigning tasks and synthesizing information as the responder, and other stakeholders troubleshoot the issue.

The incident commander is also tasked with spearheading communication to upstream and downstream consumers that might be affected, i.e., those that work with the data products powered by the broken pipeline.

In a business context, metadata is a powerful tool for understanding which teams are affected by a data downtime incident; coupled with automated, end-to-end lineage, communicating the upstream and downstream relationships between these affected assets can be painless and quick saving teams hours of manual graphing.

Once data downtime occurs, it's important to communicate its impact to upstream and downstream consumers who work with the data and those who use it. With the right approach, much of this can be baked into automated workflows using PagerDuty, Slack, and other communication tools.

Root Cause Analysis & Resolution

In theory, root-causing sounds as easy as running a few SQL queries to segment the data, but this process can be challenging. Incidents can manifest in non-obvious ways across an entire pipeline and impact multiple, sometimes hundreds, of tables.

For instance, one common cause of poor data quality is freshness – i.e., when data is unusually out-of-date. Such an incident can result from many causes, including a job stuck in a queue, a time out, a partner that did not deliver its dataset timely, an error, or an accidental scheduling change that removed jobs from your DAG.

In our experience, we've found that most data problems can be attributed to one or more of these events:

- An unexpected change in the data feeding into the job, pipeline or system
- A change in the logic (ETL, SQL, Spark jobs, etc.) transforming the data
- An operational issue, such as runtime errors, permission issues, infrastructure failures, schedule changes, etc.

Quickly pinpointing the issue at hand requires not just the proper tooling but a holistic approach that considers how and why each of these three sources could break.

As software (and data) systems become increasingly complex, it becomes more and more difficult to pinpoint one exact cause (or root) of an outage or incident.

Amazon's 5-Whys approach provides a helpful framework through which to contextualize RCA:

- Identify the problem
- Ask why the problem happened, and record the reason
- Decide if the reason is the root cause
    - Could the reason have been prevented?
    - Could the reason have been detected before it happened?
    - If the reason is human error, why was it possible?
- Repeat the process using the reason as the problem. Stop when you are confident that you have found the root causes.

There is rarely a single reason your system broke. As data engineers work to reduce manual toil with smarter processes, tests, data freshness checks, and other solutions should be able to identify the issue before it surfaces downstream. When they don't, it's a strong indication that these failsafes are inadequate. As in the world of software engineering, automated solutions, like data observability and end-to-end monitoring, are your best bet in the fight against data downtime.

To get started, we've identified the four steps data teams must take when conducting RCA on their data pipelines:

1. **Look at your lineage:** To understand what's broken, you need to find the most upstream nodes of your system that exhibit the issue – where things started and where the answer lies. If you're lucky, the root of all evil occurs in this dashboard and you will quickly identify the problem.

2. **Look at the code:** A peek into the logic that created the table, or even the particular field or fields impacting the incident, will help you come up with plausible hypotheses about what's wrong.

3. **Look at your data:** After steps 1 and 2, it's time to look at the data in the table more closely for hints of what might be wrong. One promising approach is to explore how other fields in a table with anomalous records may provide clues as to where the data anomaly occurs.

4. **Look at your operational environment:** Many data issues result directly from the operational environment that runs your ETL/ELT jobs. A look at logs and error traces from your ETL engines can provide answers. Once you identify that something has gone awry, understand its impact, determine root cause, and communicate the next steps to the proper stakeholders, it's time to fix the issue. This could be as easy as pausing your data pipelines or models and re-running them, but since data can break for millions of reasons, this often involves a fair amount of troubleshooting.

Blameless post-mortem

One of my friends, a site reliability engineer with over a decade of experience firefighting outages at Box, Slack, and other Silicon Valley companies, told me I couldn't write an article about incident management without abundantly clarifying this:

"For every incident, the system is what's at fault, not the person who wrote the code. Good systems are built to be fault and human tolerant. It's the system's job to allow you to make mistakes."

With data reliability and DataOps, the same ethos rings true. Pipelines should be fault-tolerant, with processes and frameworks in place to account for both *known unknowns and unknown unknowns* in your data pipeline.

Regardless of the incident that occurred or what caused it, the data engineering team should conduct a thorough, cross-functional post-mortem after they've fixed the problem and conducted root cause analysis.

Here are a few best practices:

- **Frame everything as a learning experience:** To be constructive, post-mortems must be blameless (or if not, *blame aware*). It's natural to assign "blame" for incidents, but it's rarely helpful for instilling trust in your colleagues or fostering a collaborative culture. By reframing this experience around the goal of "learning and improvement," it's easier to proactively take the organizational (creating better workflows and processes) and technological steps (supporting investing in new tools) necessary to eliminate data downtime.

- **Use this to assess your readiness for future incidents:** Update runbooks and adjust your monitoring, alerting, and workflow management tools. As your data ecosystem evolves (adding new, third-party data sources, APIs, and even consumers), this step will become critical with incident prevention.

- **Document each post-mortem and share with the broader data team:** As in software engineering, documenting what went wrong, how systems were affected, and what the root cause is often an afterthought. But documentation is just as important as any other step in the incident management process because it prevents knowledge gaps from accruing if engineers with tribal knowledge leave the team or aren't available to help.

- **Revisit SLAs:** In a previous article, I explained why data teams need to set SLAs for their data pipelines. Briefly, service-level agreements (SLAs) are a method many companies use to define and measure the level of service a vendor, product, or internal team will deliver—as well as potential remedies if they fail. As data systems mature or change over time, it's important to consistently revisit your SLAs, service-level indicators (SLIs), and service-level objectives (SLOs). SLAs that made sense six months ago probably don't anymore; your team should be the first to know and communicate these changes with downstream consumers.

Post-mortems are as important for data teams as they are for software engineers. As our field continues to advance (we're in the decade of data, after all), understanding how and why data downtime occurs is the only way we can improve the resiliency of our systems and processes.

The Future of Data Observability

Now that we've addressed the origins and current states of data observability, let's discuss what we see down the pipeline (no pun intended) for achieving data quality at scale and kicking data downtime once and for all.

Data quality becomes a C-suite concern.

As more data workloads move to the cloud and the economy tightens, I foresee that data will become a larger portion of a company's spend and draw more scrutiny from finance. Data teams will need to justify spending in Snowflake, Databricks, and other critical elements of data infrastructure, something that's only possible if the data they're storing, processing, and analyzing can be trusted.

Data incident management becomes codified.

We spent the last section discussing incident management, what we view as a critical component of every data engineering practice. But the fact is that incident management, root cause analysis, and several other key processes and workflows are nascent. In 2023 and beyond, we predict that data teams will spend more time and energy upfront solidifying how to fix and triage data downtime at scale, using similar muscles and tooling as our DevOps counterparts. This should bring down the time it takes to detect and resolve incidents and in the process, the money data quality costs companies.

Data observability goes mainstream.

Over the past 12 months, we've seen companies like Snowflake, Databricks, AWS, and Google launch integrations with data observability solutions or supporting features, and companies like Checkout.com, GitLab, and Intuit have built data observability platforms to support rigorous data quality initiatives. Popular architectural approaches like the data mesh and data vault mandate a single-pane-of-glass to manage visibility into data health at each stage of its life cycle. And roles focused on ensuring data quality, like data reliability engineer or analytics engineer, have become increasingly common. We predict that data observability will become a must-have layer of both cloud and on-prem stacks, and data observability best practices (like monitoring, alerting, incident management, setting SLAs, domain-driven data ownership, etc.) will enter the mainstream.

So, Where Do We Go From Here?

In the coming years, achieving data quality will become easier and harder for organizations across industries. We hope this first chapter has prepared you to go forth and conquer your data reliability goals in 2023.

Increasingly complex systems and higher volumes of data begets complication; innovations and advancements in data engineering technologies mean greater automation and improved ability to 'cover our bases,' so to speak, when preventing data downtime. Regardless of how you slice it, however, striving for some measure of data reliability will become table stakes for even the most novice of data teams.

We anticipate that data leaders will measure data quality as a vector of data maturity (if they haven't already) and build more reliable systems in the process.

In the coming months and years, we hope to see more data organizations prioritize data observability as a foundational aspect of their data architectures, workflows, and team culture. And when they do, our readers will be well prepared.

Until then, here's wishing you no data downtime!

The Basics of Data Observability

Chapter 3



Our dependence on data pipelines

Analytics and machine learning have an inspiring, awesome potential to change businesses and society. Many products and services we use today are only possible in their current forms because of data.

Packages shipped to your home depend on data about warehouse and logistics operations. Ski lifts open and close based on weather forecast data. Car interfaces get redesigned based on usage data. Large financial investments are directed based on market data. You get the idea—it's awesome when you stand back and appreciate it!

But that means we're all dependent on data more than we used to be. We become even more dependent when we add data to a product design process, add a recommender model into a product, or give analytics tools to operations teams. Each time, we add another real-world dependency on a data pipeline somewhere.

And each pipeline will fail, at some point or another.

Data outages

Failures are part of any complex system. All systems—whether hardware or software—require maintenance and repair occasionally. When changes get introduced to the system, usually to improve it, they each create the risk of disruption or failure. Over time, even the most diligent team will have something slip past them and create a problem.

If allowed to make it through the pipeline to the ultimate data users (customers, decision makers, etc.), these problems become data "outages" in the DevOps parlance. These data outages can damage the company's brand reputation, lead to lost revenue or customer churn, and in severe cases, result in fines or legal action.

What if the outages are small or only affect internal operations? Over time, these build up into a mistrust of the data that's difficult to dispel. One survey found that 77% of business leaders don't fully trust their data.* This mistrust holds back further investment in building more amazing new things with data that could push society forward.

So how can the organizations who own and operate these data pipelines systematically reduce outages' frequency and blast radius? This brings us to data observability.

What is data observability?

Data observability is the ability to understand the internal state of your data pipelines at any moment by inspecting output signals you've collected from them.

To keep pipelines reliable, the teams operating them need to understand how they're behaving and whether a change or failure has occurred that might lead to an outage. Observable pipelines don't prevent issues from occurring altogether—there will be an unexpected issue in any complex system—but it enables the team responsible for the pipeline to identify and respond before the issue becomes an outage.

Note that this differs from testing, a related concept. When data pipelines are observable, you can ask new questions about how they're working without adding additional instrumentation or checks. You can answer unknown unknowns and understand why the pipeline behaves a certain way, not just that it does so. Testing is an effective way to catch and act on known problems. Observability enables a broad understanding of system behavior.

Data observability has roots in software through practices like devops and Site Reliability Engineering. In software engineering, testing is used to validate code before it's used in production, but observability is used to understand how the system behaves ongoing.

In data engineering, testing is a good tool to employ when modifying code for a specific component in a pipeline, but observability enables a continuous understanding of how the entire pipeline works.

What does data observability help you do?

Data observability is a broad concept, but here are specific benefits:

- Identify and resolve issues before they reach business users or customers.
- Change the pipeline faster because you have a constant safety blanket.
- Identify pipeline performance bottlenecks so you can improve them.
- Automate toilsome tasks like backfills or data migration validations.

What happens without data observability? Data engineering teams are tasked with building data pipelines and new data applications; they don't have the time or resources to manage data pipeline health and quality at scale. While writing custom tests can help pinpoint specific data model issues, it doesn't scale to tackle global data health monitoring. A data team loses the trust of its stakeholders. It may also degrade long-term investments in data-adjacent initiatives like analytics, machine learning, and automation.

Data observability helps to secure a level of control around pipeline reliability and organizational trust. No matter what team you're on, you want to unlock as much value from data as possible. Data observability is tried and true way to do it.

Who works with data observability?

On some level, data observability touches everyone within an organization, from finance to HR to admin to marketing and engineering, because everyone benefits from improved data quality and pipeline reliability.

While their exact titles vary from organization to organization, here are the most direct beneficiaries of data observability:

1. **Data producers:** Data producers are responsible for how data is initially created to record things about the world. They control what is recorded, how the record is structured, and how it gets into the data pipeline. They often have the word "engineer" in their title: mobile engineer, frontend engineer, backend engineer,

etc. Observability matters to data producers because it helps them stay aware of how changes they make impact data pipelines—and the operators, consumers, and stakeholders—downstream from them.

2. **Data engineers:** Data engineers directly maintain the data infrastructure. They might have the title data engineer, data platform engineer, devops engineer, site reliability engineer, or data reliability engineer. Data operators work on pipeline logic. Observability matters to data engineers —more than any other user group— because it allows them to understand what's happening in the system they operate. Through observability, they understand the impact of changes made by producers to the logging or by other operators working on the pipeline and how those changes affect other stakeholders.

3. **Data scientists:** Data scientists focus on specific datasets. They consume the data for building machine learning models and may have a "data scientist" title or an "ML engineer" title, depending on the organization. They're concerned with outliers, duplication, and other issues that impact analysis and machine learning. They focus on getting data from the pipeline to make something that ultimately goes to a stakeholder. They care about data observability because it helps them understand the data they're working with, especially how appropriate it is for the task they plan to use it for. Knowing up front that data from a given pipeline is not reliable enough to be used in a high-profile use case can prevent major challenges with analytics or machine learning projects. Data observability helps them with freshness and completeness monitoring, duplication, outlier, and distribution shift detection. In the case of anomalies, data observability platforms will send alerts identifying potential issues. Those alerts make it easier for data scientists to keep their datasets pure and accurate.

4. **Analytics engineers:** Analytics engineers test changes within the data model. They want to use automation to reduce the hours spent writing pipeline tests. Data observability helps analytics engineers to free up time for analysis versus rote manual work. Specifically, they can do better lineage monitoring and ETL blue/green testing through data observability. They are experts in Airflow, DBT, Python, SparkSQL, and other tools for modifying data as it moves through the pipeline.

5. **Business intelligence analysts:** Business and BI analysts can do better work if data observability works. They can understand where they should spend their time, which campaigns to focus on, and when a data pipeline problem causes an anomaly. Business analysts can analyze custom business metrics, detect anomalies, and work on dimensional slicing and dicing through data observability.

6. **All other stakeholders:** All other stakeholders, from product to marketing to frontend and backend engineering, need data reliability. Often, these stakeholders benefit from the data but may not be aware of the pipeline. Customers and

stakeholders don't want data issues to bog them down or deliver false information. With data observability, these individuals and teams improve reporting and insights. They can meet deadlines faster and develop products with a better product-market fit. However, these stakeholders do not directly care about, or often know about, data observability. They feel its indirect benefits when data freshness and quality works well, giving them the freedom to trust the data they see in the dashboards or product features they interact with.

What use cases does data observability help with?

Now that you have a fundamental understanding of data observability and who uses it, you may wonder which scenarios specifically call for data observability. Here are a few of the most impactful use cases for data observability:

Analytics

When a business-critical dashboard is inaccurate or suspect, operations come screeching to a halt. Once a business leader loses trust in a dashboard they rely on to run the business, it's difficult to earn back.

Data observability tools perform deep, business-logic-driven data quality checks on the tables that power analytics dashboards and BI tools. If an anomaly occurs, data teams are notified so they can identify and resolve any issues before they impact downstream users. With proactive data management, data teams reduce time spent chasing data fires, improve the self-service BI experience, and give decision-makers the confidence they need to drive the business forward.

Machine learning

Companies of all shapes and sizes are investing in machine learning to gain more from their data and an edge on the competition. Machine learning can be a powerful tool, but data pipelines must be implemented correctly, and data must be continually verified. Training an ML model on bad data can significantly hurt a business and its customers and waste months of critical training time.

Data observability helps realize ML's full benefits and avoid costly mistakes by actively monitoring training data and feature stores for issues like freshness, missing data, and distribution changes. This process lets you find and correct issues before your models are retrained on bad data.

Data model development

As enterprise analytics requirements become increasingly sophisticated, data analytics engineers are turning to software development practices like continuous integration to manage data models at scale. To move fast and keep business users confident in the data they use to make decisions, analytic engineers must be able to detect change failures in their data models and quickly debug issues.

Many data observability platforms integrate with CI/CD pipelines to compare development and production environments and validate a successful run or pinpoint data issues. Data teams save time and move faster with immediate insight into the success or failure of a data model change. And they never have to leave the native data pipeline workstream. If a failure is detected, data observability shows what went wrong and how to fix it, eliminating hours of frustrating, manual debugging.

Data ingestion, replication, and integration

Data teams work with large volumes of complex data. While tools like Fivetran and Airbyte simplify data replication, validating that the data arrived in good shape is still often manual and time-consuming.

Whether you're ingesting third-party data or replicating data into a warehouse or lake, data observability makes it easy to compare multiple versions of a dataset and spot issues for fast, no-fuss resolution. Increase the speed and efficiency of data operations by ensuring every replication or ingestion job completes successfully.

Data warehouse or database migrations

When companies invest in a modern data warehouse, they want to get up and running and see value quickly. Unfortunately, migration complexity and manual validation processes can kill momentum and force data teams to burn time and resources chasing migration issues instead of adding value to the business.

Instant migration validation and comprehensive reporting simplifies migrations, making it easy to ensure that every row of data arrives safe and sound. Faster, smoother migrations by eliminating time-consuming manual checks. Data observability provides a reduced time to value, so data engineers can spend their time and resources adding value to the business, not babysitting a months-long warehouse migration.

What are the core concepts?

Observability is a noun that describes an attribute we wish our pipeline to have. If we have good observability, we can clearly understand our pipeline's state at any moment. In contrast, we can have poor observability, meaning it's difficult to understand what's happening in the pipeline.

Here are the core concepts that describe how data observability works in practice:

Instrumentation

This is the fundamental building block behind observability. Without instrumentation, the pipeline cannot be observable because we have no measurements of its behavior to analyze.

Full instrumentation ensures we know everything we need to understand our pipeline's behavior. Partial instrumentation creates gaps in understanding the behavior of the pipeline. Partial lineage might only capture relationships within a data source but not across sources. If metrics for pipeline health are only tracked for tables in a certain schema, we'll have blind spots about the behavior of other schemas.

Here are the building blocks of instrumentation in data observability:

Metrics

Metrics are ongoing measures of various attributes of components in your pipeline. They can measure attributes of tables (like how long they've gone without fresh data), or columns (like the percent of values matching the correct format), or even multi-column relationships (like referential integrity). They need to be aggregated over a time window so the behavior they measure can be measured over time, and changes in behavior can be detected. They can also be split out by various dimensions for further analysis, such as the percent of missing customer_id values per sales region.

Examples: hours since the last insert, number of read queries per day, percent of null values, maximum numerical value, percent of values in UUID format, etc.

Events

Events are outputs from system components when they've performed some action and can emit a record of what happened. If an Airflow job fails with a specific error status, the resulting event, presented with a table freshness metric that's out of band, will help the operator understand why the data is staler than expected.

Examples: Airflow job failures (*link to docs*), dbt build errors (*link to docs*).

Logs

Logs are the running record of all events emitted by a data pipeline component, such as the data warehouse, the ETL orchestrator, or the data transformation build environment.

Examples: Snowflake's QUERY_HISTORY view (*link to docs*), dbt logs (*link to docs*), Fivetran logs (*link to docs*).

Lineage

Lineage is the map of relationships between each element of your pipeline. Visualizing a lineage graph helps a data engineer understand the path through which data is flowing, which can help them identify the source of an issue and/or what elements in their pipeline might be affected.

Data lineage refers to the path data takes through your data system. It starts at creation, travels through databases and transformation jobs, and reaches final destinations like analytics dashboards or data stores. Data lineage provides context about your data pipeline that can help to identify the source of an issue or what data-driven applications might be affected by an issue flowing down the pipeline.

Most commonly, lineage is collected by parsing the logs of queries that write into each table. This parsing produces an *Abstract Syntax Tree (AST)* containing which tables were read from and which tables were written into.

Lineage graphs can be extended as far upstream as applications or microservices and as far downstream as users of the data products fed by data pipelines, like dashboards.

Monitoring

Monitoring is collecting and displaying the signals collected from the instrumentation, including metrics, events, lineage, etc. The visualization layer of the collected observability information makes it usable to humans. Pure instrumentation can gather and record signals that make the pipeline technically observable, but without monitoring, it would be hard for humans to do anything with them.

Monitoring also unlocks a related but different concept: *Service Level Agreements (SLAs)* which are promises about how the pipeline will behave. SLAs are made to create stability for Consumers and Stakeholders because they know what to expect from the data. Monitoring makes SLAs possible because it provides the metrics from which the promises in the SLA are measured. SLAs are not directly part of observability, but an observable data pipeline, exposed via monitoring, it is difficult to maintain the promises in an SLA.

Alerting

Alerting is another capability that makes observability useful. Alerts draw attention to the monitoring so actions can be taken. Without alerts, we must look at our monitoring every day, every hour, etc., to see if something needs our attention.

Alerts can be based on rules. Rules-driven alerts are deterministic and produce the same alert for the same conditions every time. Rule-based alerts are easy to understand but can be difficult to set such that you always get alerts for things you want to know about while avoiding false alarms.

Alerts can also be based on anomaly detection. These alerts can adapt to various behaviors inherent in the metrics to be alerted to, like seasonality or changing volatility. They can be more convenient and accurate than rule-based alerts in daily usage but are harder to reason about or explain to others, making them inappropriate for some use cases.

How do data observability tools enable data observability?

Data observability exists to help you confidently answer the question, "is my data good?" Which capabilities you'll need to answer that question depend on how you define "good" and how confident your answer needs to be. With that in mind, here are the capabilities you should look for in a data observability platform (*like Bigeye*, for example):

Metadata monitoring

Virtually all data observability tools provide some level of visibility into the general health of your warehouse. This broad and shallow approach to observability uses metadata to help you quickly determine if your data is fresh, complete, and in the right format. This is helpful for data teams who need to check the box that the data is operationally "good". Checks should also include helpful context, such as lineage, to help you understand an issue's true impacts and prioritize.

Anomaly detection

Beyond broad visibility, many companies also need to understand if the data being delivered adheres to certain business logic and is accurate enough to be fed into an analytics dashboard, AI/ML app, or to external customers for consumption. For this, you'll need to go beyond simple metadata checks and look into the data quality itself.

Unlike a traditional data quality tool or testing framework, a data observability solution should use AI to track data trends over time and surface anomalies. This helps you find and investigate "unknown unknowns," which can easily slip by as false negatives and wreak havoc downstream.

Automation

Data observability products are meant to save data engineering teams time. Any solution you consider should include robust automation and enough intelligence to take tasks off a data engineer's plate without forcing them to spend inordinate amounts of time on the product. These can include capabilities like automatic data lineage collection across the warehouse or automatic metric suggestions for tables and columns. The ability to automatically refine anomaly-driven alerts is especially important for reducing unwanted false positives, which can cripple productivity and data team effectiveness over time.

Flexibility and scale

Like IT operations before it, data pipelines are getting bigger, faster, and more complex by the day. To effectively apply data observability at scale, data engineering teams must be able to manage it at scale. This should include an intuitive UI for light users and access to a robust CLI for version-controlled CI/CD processes for programmatic management. Data observability is only useful if it can observe your data stack, so out-of-the-box integrations with platforms Snowflake and Redshift, legacy databases like SQL Server, and MDS tools like Airflow, dbt, fivetran, are a must. Finally, no two data stacks are alike, so full access to a robust REST API and SDK should allow data teams to extend data observability to address their specific technical and business requirements.

Summary

Data observability is critical for making data pipelines reliable enough to drive production applications like analytics dashboards and machine learning models. With data observability, organizations can prevent inevitable pipeline issues from reaching end users. This can prevent poor decision-making, churned users, and regulatory fines.

Organizations that use data observability can make faster improvements to their pipelines because they understand the impact of the change. They can easily undo it if it introduces unwanted behavior.

It is unique from testing or other methods that require you to know about a question in advance. That's powerful because it delivers the flexibility to ask new questions about pipelines in real-time as they come up.

Data observability is driven by instrumentation, monitoring, and alerting. This allows teams to understand how their pipelines behave and the quality of the data moving through them.

In implementing data observability, organizations improve the quality and availability of their data, ultimately driving better business outcomes and unlocking further investment in data.

Understanding Data Quality and Data Observability

Chapter 4

TELMAI

Quality is a term that often indicates the value that we are willing to associate with, be it the food we eat or the cars we drive. High quality means high trust and, therefore, high value, which is no different for Data.

Today, Data is the backbone of all modern businesses. With the increasing amount of data generated and consumed daily, companies must ensure their data is high quality.

Data quality indicators provide a way for organizations to measure the quality of their data. On the other hand, Observability helps businesses accelerate visibility into their data quality KPIs.

The exponential growth in scale and volume of data has made it extremely hard to assess the health of the data.

Data Observability is a set of techniques that can help continuously monitor data to detect anomalies. This approach goes beyond the traditional capabilities as it leverages statistical analysis and machine learning techniques to monitor various data metrics, time, and sources of the data issues and aggregate those. Such a system aims to reduce the mean time to detect (MTTD) and mean time to resolve (MTTR) data issues.

Data Observability is an approach designed for modern data architecture that scales with the data.

Data Quality Key Performance Indicators(KPI)

Data quality indicators are a set of metrics used to measure the quality of data. These metrics include completeness, accuracy, consistency, uniqueness, timeliness, and validity.

The goal of data quality indicators is to provide an objective and consistent way to measure the quality of data. Organizations use six most common data quality indicators to evaluate their data quality.



Completeness

Data completeness refers to the degree to which a data set contains the required and relevant information without missing or incomplete data. In other words, it measures whether all critical data points have been captured for a particular dataset.

Complete data is essential for accurate and reliable data analysis and decision-making. Incomplete data may lead to biased or erroneous results, particularly in statistical analysis. Additionally, missing data points can make it difficult to identify patterns or relationships within the data, which can compromise the validity of the analysis.

There are various reasons why data might be incomplete, including data entry errors, partial data load data loss, or data extraction issues.

Using Data Observability, users can automatically get the benefits of continuously monitoring the completeness of data. Unlike traditional DQ tools that measure the completeness of data against pre-defined criteria, data observability leverages historical insights to predict the thresholds of completeness. Completeness can be measured using size of data, row count, or populated attribute values.

Uniqueness

Uniqueness is an important data quality indicator that measures whether data contains duplicates. This indicator is particularly relevant when dealing with data that includes customer IDs, product codes, or other unique identifiers.

The uniqueness indicator evaluates whether the same identifier appears more than once in the dataset. For example, if a customer ID is supposed to be unique, but there are duplicate IDs in the dataset, the uniqueness indicator would be low. Sometimes even if the identifiers are unique, the data set might contain multiple records for the same customer.

Non-unique identifiers can lead to a range of issues. For example, if duplicate customer IDs exist in a customer database, it can be challenging to identify which transactions or interactions belong to which customer. This can result in inaccurate reporting, poor customer service, and incorrect billing.

Traditionally, this was solved by applying table constraints like the primary key and manual stewardship to identify and remediate duplicate records. These techniques do not scale for modern data's volume and data formats (semi-structured, streaming, columnar, etc.).

Data Observability tools will not only automatically classify the data into categorical and non-categorical, but they will also automatically detect attributes that hold unique values and flag when there is a drift in the uniqueness of the attribute.

Timeliness and Freshness

The timeliness of data refers to the degree to which data is available within an appropriate timeframe to support timely decision-making or analysis. In other words, it measures whether data is current and up-to-date.

The Freshness of data is closely related to the timeliness of data, but it specifically refers to how recently the data was collected or updated. Fresh data is data that has been collected or updated recently and is, therefore, more likely to be accurate and relevant.

Freshness can be categorized into:

- Table-level Freshness, i.e., how often a table is being updated and whether the time since the last update is anomalous
- Record-level Freshness, i.e., what is the percentage of outdated records (entities) and whether this percentage increases abnormally
- Entity-level Freshness, i.e., similar to record-level, except it takes into account that multiple versions of an entity may exist in the same table

There are different options for implementing table-level Freshness. Some use database query logs to determine the write rate; others look at the last update timestamp of the table in its metadata. In contrast, some others look at the most recent timestamp within the table (in case the record update attribute is part of the schema).

Using Data Observability, users can automatically get insights on the frequency of updates to tables and the data and configure freshness metrics relevant to business needs.

Validity

Validity is an indicator that measures whether the data conforms to predefined expectations or rules.

Often data teams ensure validity by checking for data types, formats, and ranges. Data is valid if it conforms to a particular syntax or rules.

This is often the first step in ensuring data quality.

Often data teams will write validation checks for every system in the data pipeline, like Mobile apps, DBT, data-loaders, landing storage, etc.

For example, let's say a company collects customer data through an online form. Data validity means ensuring that the data entered by the customers is accurate and complete and represents the customers' precise information. This may involve validating data against predefined rules and criteria, such as ensuring that email addresses are correctly formatted, that phone numbers are valid and complete, and that age is within an acceptable range.

Open-source tools like Great Expectations have played a key role in accelerating pipeline data validation.

Data Observability takes this approach further by first predicting and recommending validations based on the data stored in the system(statistical analysis) and then enabling users to write scalable validations.

Accuracy

Data Accuracy refers to the extent to which organizational data reflects the actual values or attributes of the objects or events it represents.

In other words, data accuracy is a measure of how closely the data matches reality. Ensuring data accuracy involves verifying data validity and checking for errors, inconsistencies, and, most importantly, anomalies. This can be achieved through various methods, such as manual data entry verification, automated data validation algorithms, and data profiling techniques.

Maintaining data accuracy also requires ongoing efforts to monitor and correct any errors or issues that arise over time.

A classic example of invalid data would be out-of-range values. Such data passes all the tests for validity, completeness, and uniqueness, yet it might not to be accurate.

Accuracy is the most challenging data quality metric as it needs context. Sometimes the context lies with source owners, and other times with data consumers.

Traditionally, this was solved using business rules and manual stewardship, and both these approaches fail to scale for the modern data ecosystem.

Data observability tools provide a sophisticated approach by aggregating historical data and data dependencies and flagging anomalies within data, like outliers and drifts(over time). Data teams can further validate/remediate anomalies using a human-in-loop approach.

Consistency

Data consistency refers to the degree to which data is uniform, coherent, and accurate across different sources or instances. In other words, data consistency means that data is the same across different systems, databases, or applications. Consistency can be evaluated by comparing the data to other sources, checking for duplicates, and evaluating naming conventions. It is essential to establish data standards and guidelines, implement data governance controls, and regularly monitor data quality to ensure data consistency. For example, if sales data is inconsistent across different regions, comparing performance accurately or identifying trends can be challenging.

Users who implement data observability across multiple systems can easily compare the data metrics to find drifts (inconsistencies) without writing code/SQLS.

Role of Data Lineage

Data lineage is related to data quality in that it is a crucial aspect of understanding and managing the data.

Data lineage refers to the ability to track the data as it flows through the system, including where it came from, how it was transformed, where it is stored, and how it is used. This information is essential for understanding the data and its quality, as it allows you to trace any issues or errors back to their source and understand how the data has been used.

Data lineage can also be an essential aspect of data governance. It allows organizations to understand the data and its quality and ensure that it is used correctly and effectively.

Data teams use lineage to improve data quality by identifying and correcting errors and ensuring that data is entered and stored in a consistent format. Additionally, understanding data lineage can help organizations identify and remove duplicate or inconsistent data and ensure that data is of sufficient quality.

Data lineage is not a data quality Indicator. Still, it is a crucial aspect of overall data governance. Most observability tools leverage lineage to find the root cause and impact of an issue.

Why do traditional data quality tools fail in the modern data stack?

Traditional data quality tools satisfy the needs of the conventional data stack that is monolithic and is designed to work with structured data sources. These tools focus on data validation, cleansing, and standardization to ensure its accuracy, completeness, and reliability. Writing validation rules in SQL or using open-source tools is one way to implement data quality. Traditional ETL tools often have data quality rules embedded in their user interface to transform the data into higher quality and work best for structured data with limited scale.

However, traditional data quality tools cannot rise to meet the needs of modern data stacks, and here are the main reasons why they fail:

- **Data Stewardship and Sampling:** In some cases, using a sample of data by data stewards rather than the entire data population for analysis, remediation, or decision-making can be helpful. But it has several drawbacks and limitations, including the risk of bias if the sampling method is not random or if the sample size needs to be more significant to be representative of the population. This can lead to inaccurate or misleading conclusions. Sampling also limits the scope of analysis to the specific sample taken—loss of granularity of data or unusual data points necessary for analysis. And the inability to account for data outliers is another limitation of this method.

- **Inability to handle semi-structured and unstructured data:** Traditional data quality tools were designed to work with structured data stored in relational data sources and cannot handle semi-structured and unstructured data that's a huge component of modern data stacks that include data from social media posts, audio and video recordings, and free-form text.

- **Lack of scalability:** Modern data stack is designed to generate and handle massive amounts of data, which can be difficult for traditional data quality tools to manage that may need help to keep up with the data volume leading to slow processing times and potential data quality issues. Most of these tools were not designed with distributed computing and faced performance bottlenecks.

- **The complexity of data sources:** Modern data stacks often consume data from multiple sources, including internal and external systems, third-party APIs, and cloud-based data stores. Traditional data quality tools may need help to handle the complexity of these data sources, leading to data quality issues and delays in processing.

- **Limited data governance capabilities:** Traditional data quality tools need more advanced data governance capabilities to manage data in modern data stacks, such as data lineage tracking, data cataloging, and data privacy management.

Traditionally, multiple techniques and tools were available to measure, monitor and improve data quality. They include data profiling, data cleansing, validation, standardization, and governance. Data quality is an ongoing process that requires continuous monitoring, updating, and maintenance to ensure that the data remains accurate, complete, and consistent over time.

This path can be accelerated using self-learning tools like Data Observability.

Data Observability as the foundation of Data Quality.

Data Observability tools are natively designed to scan the entire dataset to identify anomalies like outliers and drift. This outcome can be easily classified into data quality indicators defined above.

Data observability tools are perfect candidates to deliver superior data quality to the modern data stack as they bring versatile capabilities, as follows, to the table.

Handle Semi-structured Data: Data Observability tools are designed for modern data stacks that require the processing and analyzing of large volumes of data from various sources, including semi-structured data.

Operate in the Cloud: Data Observability is designed to operate and scale efficiently in the cloud and provide capabilities essential for managing the scale of data in modern data stacks.

Work with Streaming Data: Sophisticated data observability tools can handle data stored in a batch or streaming data. They are well-integrated platforms that can span and serve complex data pipelines in ways traditional data quality tools cannot.

Leverage Machine Learning: While data observability tools can validate data against predefined metrics from an available set of policies, they leverage ML and statistical analysis to learn about data and predict future thresholds and drifts.

Provide Interactive Visualization: Given their constant monitoring and self-tuning nature, data observability tools can automatically curate issues and signals in the data into data quality KPIs and dashboards to showcase the ongoing data quality trends visually. These tools are equipped with interactive visualizations to enable investigation and further analysis.

Conclusion

In summary, Data Observability is a set of techniques, and Data Quality(DQ) is the state that's defined using DQ metrics.

Given their constant monitoring and self-tuning nature, data observability tools can automatically curate issues and signals in the data into data quality KPIs and dashboards to showcase the ongoing data quality trends visually. These tools are equipped with interactive visualizations to enable investigation and further analysis.

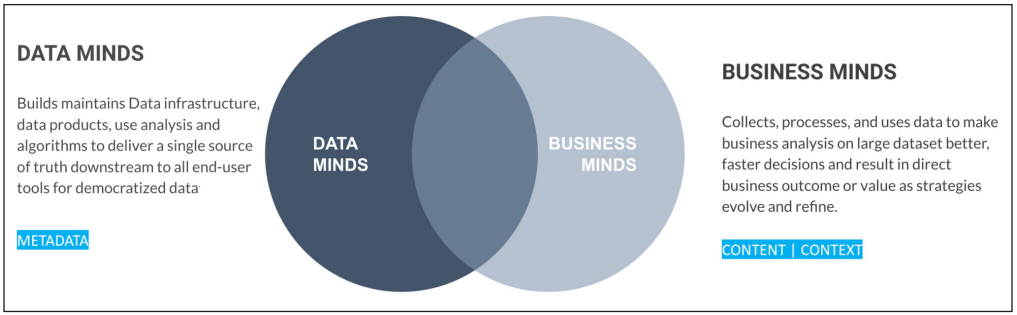
One of the outcomes of Data Observability is automatic identification and alerting on DQ metrics. Still, there is so much more to it, like business metric drift, which could be a true positive or false positive. Data Quality is one of the many use cases of Data Observability.

The Convergence of Data Quality and Data Observability

Chapter 5



With the growth in data and the adoption of modern cloud architecture or hybrid with data spread across cloud and on-premise, both producers and consumers of data are shifting away from the traditional ideologies around Centralized Data Ownership towards new principles around "Decentralized Data Ownership." Instead of flowing the data into a centrally owned data lake or platform, more domain owners are adapting the principles of "Data as a product" and serving datasets in an easily consumable way using self-service data infrastructures. Today's modern data teams use a variety of architectures ranging from Cloud Warehouse to Lakehouse, Data Virtualization to Data Fabrics or Mesh to suit their varying needs of speed and compute power. For leveraging data for improved day-to-day operations or better Strategic decisions, organizations need to look across both perspectives of minds – Data minds and Business minds as below.



Data Minds are producers (predominantly Data engineers) who build infrastructure and data products for consumption by business users or end users. Business minds are consumers of this made-available data that performs analysis to aid in business decisions or strategies. Both of these minds view data differently based on their roles, and their view of data quality is different, and so is their measurement.

Let us review different personas in an organization and how they see data quality -

- First, we look into Data Engineering, which combines three core practices around Data Management, Software Engineering, and I&O. This focuses primarily on reconstituting data into usable consumer forms by building and operationalizing data pipelines across various data and analytics platforms. *Data Engineers, aka producers of the data*, must be robust across data modeling, pipeline development, and software engineering and ensure that data always is reliable and provided per business service level agreement. This commitment requires executing a variety of checks across a large amount of data at speed and preventing any insufficient data from reaching downstream. Due to the scale of data, this engineer-focused validation is wide and shallow and includes validation at the surface or metadata levels of data. These verification methods spread from API checks preventing issues before data lands (Shift Left) to detecting problems at the warehouse or lake house using anomaly monitoring (Shift Right). Both approaches revolve around metadata or custom-level SQL checks using observability to prevent or detect unknown issues before consumers use the data or meet agreed-upon Data Contracts/ SLAs with consumers/internal/external stakeholders. **The primary focus of this persona is more around "Data Reliability" making sure the SLAs are met with the consumers of the data plus other internal/external stakeholders regarding data availability.**

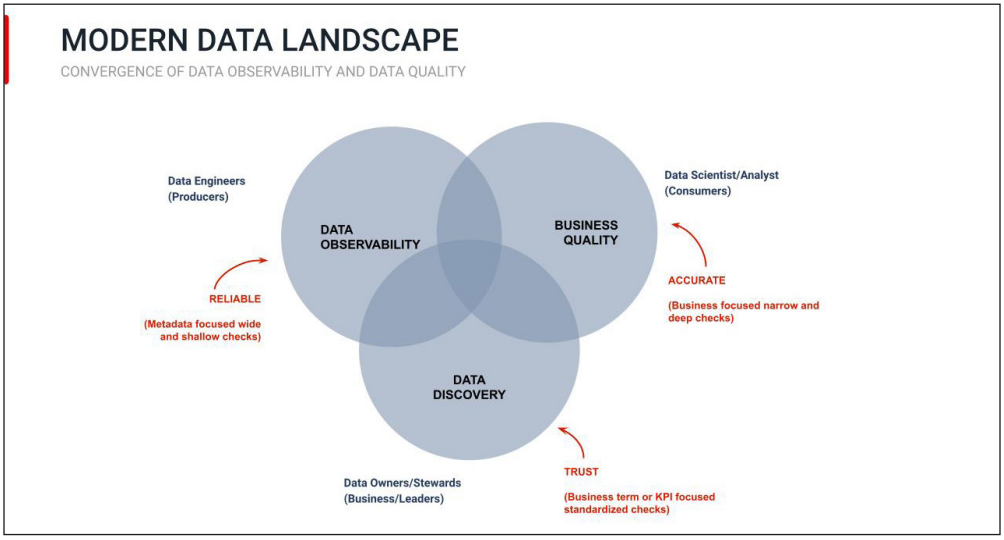
- While in the world of *Data Scientists or Data Analysts aka consumers of the data*, the most common statement one would hear is "Garbage in, garbage out". The developed model or report is only as good as the data being fed. Here the data quality assessments are defined primarily by the use case and vary for each model and report generated. The data quality checks are more business-centric and must

be measured continuously to manage and resolve any known issues using the data context. Primarily, the consumers of the data or downstream applications want to ensure that the data they use for building reports and analytical models are business fit. This requires performing narrow and deep business checks with aggregate analysis, deterministic rules, statistical measures, data accuracy, integrity, custom checks, and other dimensions of data quality, not the focus of upstream or data producers. E.g., a data scientist building the next big model for optimizing route planning for the distribution of materials produced may be interested in certain data characteristics that may not be critical for a marketing function focused on targeting the younger generation demographic for new revenue. This evaluation and need for data quality change occasionally and dictates the need for decentralized data ownership or evaluation of data quality in their context. **The primary focus of this persona is more around "Data Accuracy" making sure the context of the data meets the need of the objective on hand, i.e., reports, analytical models, etc., or fit-for-purpose**

- Now, let's not forget about the *Data Stewards*. This role is very confusing in the modern data landscape and even more confusing with decades of broken promises from Data Governance platforms. The word "Stewardship" means "supervising or taking care of something," and, yes, even in decentralized data ownership, there are a set of business terms or critical data elements (business terms) that needs to be maintained consistently to maintain business integrity, e.g., Bank Routing Number, SSN, Customer ID. This requires consistent data quality checks applied across the organization's varying data assets based on business context (semantics) despite where and how it is defined in the technical metadata. These checks require the automated discovery of semantics, and a consistent measure of quality checks is a must and a need for "Agile" Data Stewardship or Federated teams. **The primary focus of this persona is "Stewardship" to ensure the quality of the critical data elements is measured consistently and standardized.**

- Last but not least, *Data Leaders aka Business Stakeholders* are focused on what could make their next big win and want to understand how they could use the data to enable new or improvised strategies toward positive business outcomes. A Gartner study shows that by 2025, 80% of data and analytics governance initiatives shall focus on business. By 2024, 50% of organizations will adopt modern DQ solutions to support their digital business initiatives better. This clearly shows how business leaders focus on identifying key business processes, KPIs and KRIs, and underlying data or metric assets with a direct linkage to organizational mission-critical priorities. Having baselined values of business KPIs/KRIs before starting DQ initiatives and measuring continuously shall help give the leaders a top-to-bottom-down view and, more importantly, which domains and/or applications need to improve data quality. This view of "Top-down" business quality checks is essential for assessing key performance indicators and benchmark

measures for data quality against any initiative's outcomes. Also, Privacy, Security, and Compliance Leaders need at a minimum continuous validation and assurance depending on the classification or type of data to ensure 100% regulatory compliance. **The primary focus of this persona is on how to "Trust" the data, KPIs on hand, and their usefulness for the next best strategy.**

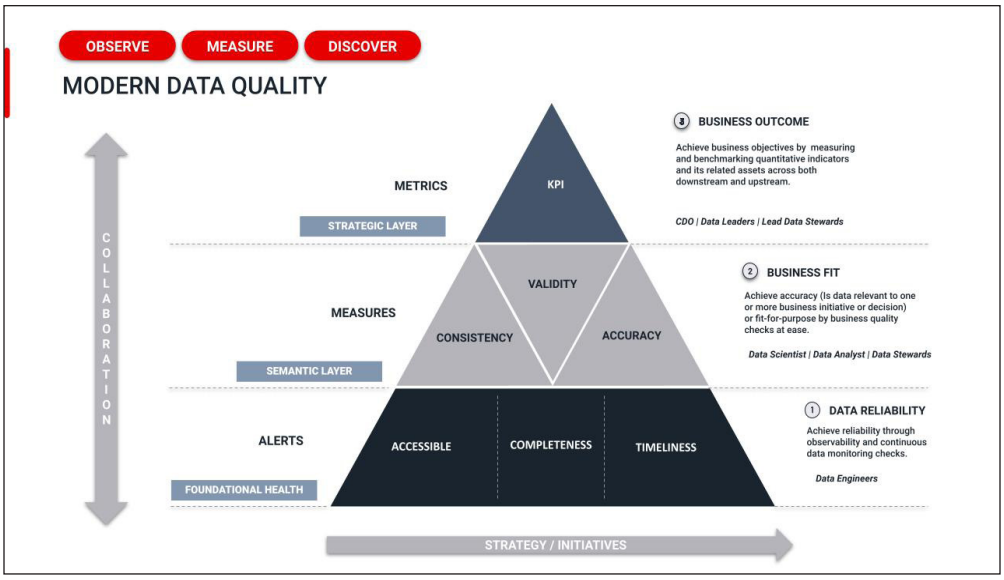


Now you can relate to how the organization's data minds and business minds view the same data but relate and measure data quality in different ways or relevant to their roles and responsibilities. Different personas (Data Engineers, Scientists, Data Stewards, Data Analysts, etc.) look at the same data through different lenses; however, they still need to come together and collaborate to win. This is foundational and critical as both these personas are essential to better outcome positive results and enable organizations to become more data-driven.

To enable more data-driven and establish "*Decentralized Data Ownership*" and facilitate collaboration more meaningfully, we should move away from traditional ideologies to more modern ones and converge multiple capabilities of observability, business quality, and discovery one platform. In these so-called Modern Data Quality platforms, various personas can collaborate meaningfully to make organizations more data-driven and deliver reliable and accurate data. This also requires shifting from traditional data quality approaches of measuring dimensions and checks via rule-based to a more automated, continuous, and semantic-powered Modern Data Quality Platform. This self-serve agile platform is critical as different roles within organizations come together to collaborate but still measure and see Data Quality in their own ways. However, being a Modern Data Quality Platform is more than that. Below is a detailed comparison of what makes the Modern Data Quality Platform different from Traditional/Legacy Data Quality and powerful.

Traditional-Legacy	Modern
Data Stewardship is centralized	Decentralized Data Ownership
No understanding of business terms	Enforces business term checks consistently
Measurement-centric	Remediation-centric
No Collaboration across personas	Collaboration focused via converging observability and quality, discovery capabilities
Manual rules-driven and not scalable	Automated, Continuous, and Auto Scalable

Let's discuss how the convergence of capabilities of observability and quality in a modern data quality platform shall help using this picture below



In the above triangle picture, the shape triangle depicts the organization hierarchy from top to bottom. On the top are the leaders making strategic decisions using key vitals, i.e., KPI or business metrics to make the next big move, while the bottom layer is the foundational team, i.e., data engineers creating and responsible for data infrastructure and getting the data ready for consumption. In the middle are the consumers of the data, i.e., data analysts and scientists either using the data for retrieving insights via data visualization, business intelligence, or building analytical models for optimizing business processes. In this layer is also the federated stewardship team i.e., data stewards responsible for managing critical data elements and standardizing data management and use of data. Using a modern data quality platform that converges capabilities, data engineers at the foundational level can measure data reliability by observing any data asset at the metadata level. They don't have to go into the details of the asset, but by merely looking at the metadata, they can understand the health of the data in the systems and, therefore, their reliability and trust around data readiness for consumption. By monitoring the metadata level measures over a period, anomalies or any data breach can be easily discovered using data observability concepts.

The metadata level measure at the Data Engineering level involves looking into

- Volume - How complete are your data tables?
- Freshness - How up to date are your data tables, and when were they last updated?
- Schema - Were any changes made to the organization of your data?
- Duplicates - Amount of duplicates in your data tables?

While at the data consumption layer, we have to go beyond the metadata and into the context of the data. That's because the business fit and needs change based on the insights being retrieved using business intelligence reports or analytical models. E.g., a data scientist may build a regression model today, and later the same scientist may build a semantic classification model for a different purpose. Also, the data scientist is involved in experimentation, sampling various algorithms which need different metrics, features, and aggregates. The data needs to be high quality not only when the model is built but also upon productionized as the model's effectiveness highly depends on the quality of the data continuously being fed. Even when the data that feeds into two models are the same, the objective and needs differ for each model from what is considered a "fit". This simply means we have to go beyond metadata and into the context of the data and check the quality against each model criteria and measure over time for any drift.

This context-level measure at the Data Scientist or Analyst level involves looking into

- Distribution - Is your data in an acceptable range, extreme value checks, minimum, maximums?
- Statistical - analyzing quantitative data analysis in terms of mean, median modalities, standard deviations, kurtosis, and quartiles
- Frequency - Is your data in the right format with its enumeration?
- Business terms - Discovering business terms and checking the validity of values against those reference values so it's consistent across the business. E.g. product name, Account number, SSN, and Phone number,
- Process Monitoring checks - quality validations of data across applications for integrity.

On the top layer are the leaders making strategic decisions using key business metrics (KPIs) like "revenue," "customer count," "churn rate" or "NRR". These metrics not only need to be observed for anomalies but also need to figure out if the actual tables contributing to this KPI are affected or not and the overall quality of the KPI being used in decision making or the usefulness. This involves tracking each asset and attribute at its level of quality assessment and bringing collectively to the KPI level measurement and its level of quality, and alerting users of any breach in benchmarks to help improve the decision-making process. Given that the KPI can spawn across multiple departments, teams, and functions, the collaboration must also be managed to improve the quality enabling organizations to be more data-driven.

As a whole, each persona or user type can now see and measure data quality in their own ways with an agile platform that processes the data faster and produces these quality benchmarks using a convergence of capabilities around observability, quality stewardship, and automated data discovery. To deliver reliable and accurate data, modern data teams need modern quality platforms and the convergence of capabilities around observability, quality, and discovery.

Data Products Need Deep Data Observability

Chapter 6

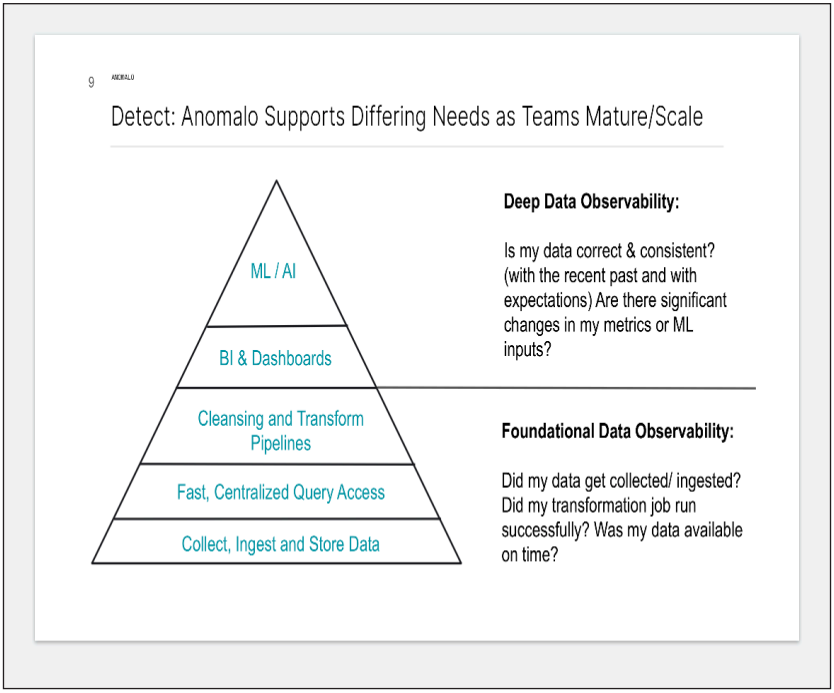
Anomalo

Data observability is an essential tool for any team that cares about identifying and resolving data issues quickly. But data observability can take many forms, from rules-based data quality to comprehensive platforms monitoring metrics and changes inside the data. Which capabilities does your team need, and why? In this chapter, we'll explain the difference between foundational and deep data observability—and why teams building data products should invest in deep data observability.

Data products and the hierarchy of data needs

The value of a data organization is in building data products that can be consumed internally by systems like AI models and BI dashboards and externally by partners or customers. Data products, much like software products or physical goods, have contracts with their users. Users expect data to be accurate, trustworthy, and reliable—often, they also need data aggregated or in a specific format. So, to build data products, raw data must be cleaned, combined, distilled, and transformed.

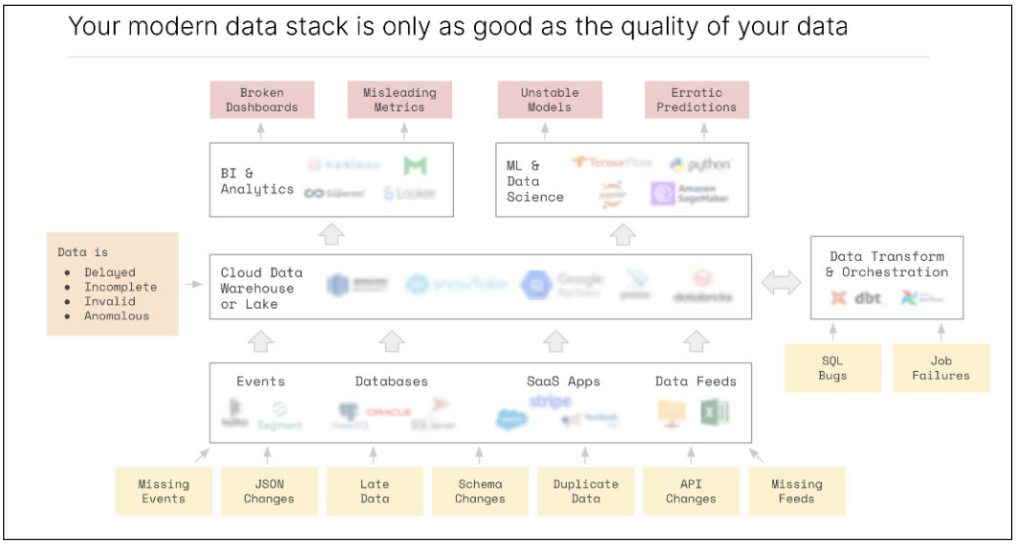
As you can't go straight from raw materials to manufactured goods, jumping straight from collecting data to releasing data products is impossible. To represent the layers of technical maturity that teams go through, we found a useful metaphor the hierarchy of data, which we first heard from data science advisor Monica Rogatti.



At the bottom of the pyramid are fundamental layers like collecting data and accessing it from a central location. Once they can send data through automated transformation pipelines, teams have created data products and are ready to use them to power BI dashboards and ML models.

The inevitability of data quality issues

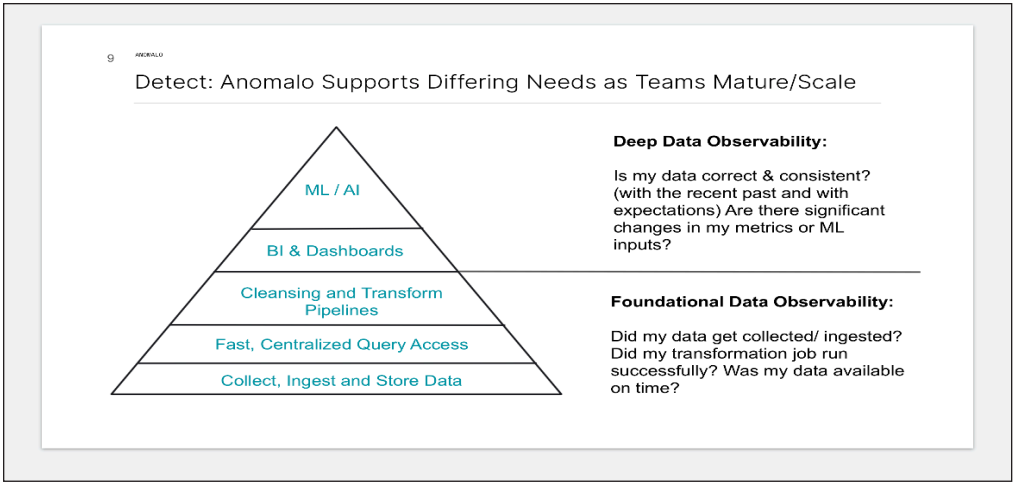
As software products inevitably accumulate bugs, data accumulates errors. The below diagram shows that for every component of the modern data stack, from data sources to ETL pipelines, there are numerous ways for errors and misunderstandings about the data to be introduced.



As teams build their pyramid of data science capabilities, they amass more data, infrastructure, and ways for things to go wrong. Data products also have additional requirements that make them especially challenging to monitor. Which brings us to the question: how should the approach to identifying and resolving data errors change as a team progresses up the data hierarchy of needs?

Foundational data observability vs. deep data observability

With the rise in data engineering, foundational data observability has become table stakes for companies building their infrastructure. Foundational data observability monitors metadata (information about the data), such as when it was ingested and whether jobs were completed successfully.

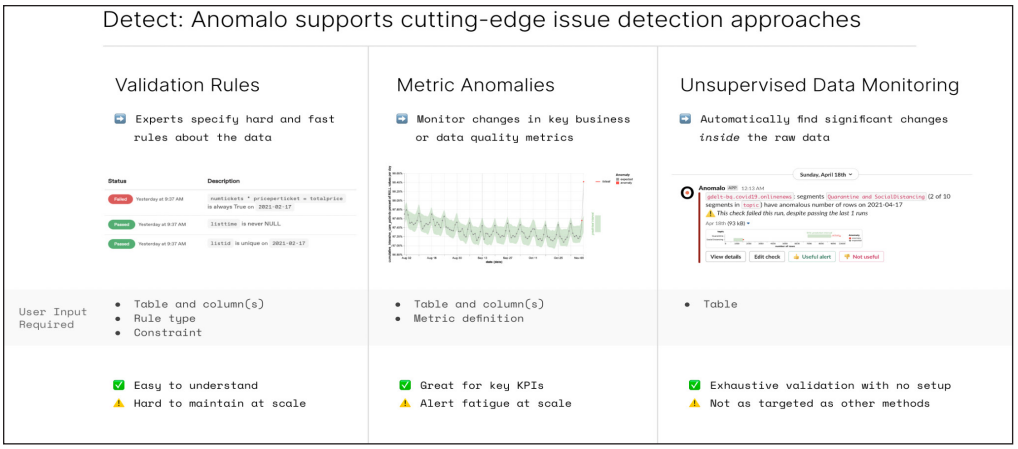


If we think of a data stack as a factory that manufactures products, foundational data observability ensures that all the machines are working and the inventory arrives on time. However, looking at the metadata only gets you so far. To know whether a product works the way it should, you must inspect the data.

That's what deep data observability does: it looks inside the data, at the values themselves, to ensure no issues or changes should be investigated. Companies with data products need deep data observability. Otherwise, while they might know that the lights are on in their factory, they have no quality assurance guaranteeing that they're upholding the contract with their data customers. Those customers might be models that behave erratically because the data looks very different from the data the model saw during training; they could be end users whose account data is corrupted, or they could be BI dashboards where a sudden change in sales metrics sends off false alarms.

Achieving deep data observability

Deep data observability looks inside the data itself to identify issues so they can be addressed before downstream consumers are affected. To implement deep data observability, what are your options? Let's look at three approaches for detecting issues inside the data itself.



Validation rules

The traditional way is to use data quality rules to observe what is happening in the data. These conditional checks enforce guarantees about the data, such as "Return false (fail) if the Name column is NULL." (This is a simple example; rules can be sophisticated SQL statements as well.)

Validation rules have several advantages:

- They are easy to read and understand.
- They allow subject matter experts to express their business requirements for the system.
- They are reliable and don't lead to false positives (where a rule says a data quality issue exists, but it does not).

However, rules are not a scalable approach to data quality for modern enterprises, for the following reasons:

- They are brittle. Data changes all the time, and to remain accurate, rules will need to be continuously maintained and updated, which is expensive and difficult.
- They require hard-coded constraints difficult to define correctly. As one example, analysts will often up setting a range for the rule too narrow, causing it to under-trigger and miss issues, or too broad, causing it to over-trigger and result in alert fatigue.

- In a practical sense, rules are impossible to scale to the size of modern datasets. A modern organization might have on the order of 10,000 tables. Even if they decide to only set up monitoring on the top 1,000 most important tables, and each table has an average of 50 columns with several rules per column, they will need to write and maintain hundreds of thousands of rules.

Rules have a place in deep data observability to encode certain key business requirements for a dataset. But, as we will see in a moment, rules should be used combined with other strategies to achieve deep data observability with minimal manual effort.

Metrics monitoring

Another way to observe what's going on inside the data is to monitor statistics about the data. These could be table-level statistics like row counts or column-level statistics like the percentage of duplicates or the mean value. To ensure that you can identify abnormalities in the data, which could point to a data quality issue, you could set up thresholds, defining what you consider a normal or acceptable range for the metric. Rather than defining this range yourself—which will be difficult to maintain—we recommend using a time-series model that can automatically learn an appropriate range for the data and adapt as the data changes.

Metrics are a great addition to deep data observability for several reasons:

- They can help you track key KPIs of your data.
- With a good time-series model, you can automatically adjust for factors like seasonality, holidays, and changes in a trend.

The downsides of a metrics-based approach are:

- Without a well-tuned time-series model, metrics-based alerts can over-trigger.
- You might miss small but important errors in the data because the data looks normal in the aggregate.

Deep data observability with ML

Also called unsupervised data quality monitoring, a final method for deep data observability is to use a machine learning algorithm to automatically detect meaningful changes in the data distribution arriving into a table over time.

These algorithms can be tuned to detect a wide range of issues, such as whether the percentage of NULL values in a set of columns has increased, a specific segment of data has disappeared, or even whether the relationship between multiple columns has changed (e.g., two columns used to sum to the result in a third column, but now no longer do that for a subset of records).

The advantages of such a method are:

- Unsupervised data monitoring "just works"—you can point it at a dataset, and it can monitor for unexpected changes with no human intervention.
- It can find a wider range of issues in the data than more narrowly scoped metrics or validation rules can find.
- An ML algorithm can learn to suppress repeated alerts automatically to help with alert fatigue, where the same alert keeps triggering and reduces the team's overall sensitivity to alerts. When the same issue occurs repeatedly, the model will treat it as the "new normal."

Deep data observability with ML is powerful, but still subject to disadvantages:

- Because it's sampling across the entire dataset, it can miss "needle in the haystack" errors that validation rules will catch.
- For the same reasons, this approach doesn't help you zero in on certain key slices of data how metrics monitoring does.
- Making this approach work well in various circumstances is a significant technical and machine-learning challenge.

Getting started

If you're building data products, we recommend investing in a deep data observability solution that includes all three of the above strategies—and we recommend investing now rather than later. When teams implement deep data observability, they often find many data issues that had gone undetected, silently affecting models and dashboards. These issues become more difficult to fix the longer they persist. To name just one problem, bad data leaves a scar in the dataset that needs to be painstakingly backfilled (or worked around for the foreseeable future).

Anomalo is a platform for deep data observability. We're always happy to help teams navigate this space and build trust in their data.

Real-Life Data Observability – Beyond Data Downtime

Chapter 7

RightData

Data Observability has dimensions beyond the cost of data downtime; it's more about a holistic approach to make data trusted and better. When data leaders combine reachable metrics with real-life practice, that creates actionable Data Observability impact, not just a buzzword. We call this Real Life Data Observability.

Real-Life Data Observability

Observability as a concept allows software and systems to be "seen" and allows questions about their behavior to be answered. When data leaders face the impact of the investment for data observability tools to meet an immense data requirement, their first question is often, "How does this really help us?"

They are asking how their data team manages real-life data because the changes and issues never stop. Data teams frequently struggle to ensure quality data due to greater volume and complexity, diverse APIs or sources, and multi-cloud load. Add fast-batch and streaming to the mix, and the need for better data to targets is a challenge every day. Data observability must answer the "so what" question and show value for real-life data problems.



Real Life – it must be simple. "You should be able to easily build data connections across various source systems, design queries or query chains, perform reconciliations, and configure batch job scheduling. One example is notifications and exception reporting used for our Sarbanes Oxley (SOX) control evidence."

Maximize. Real-life data practitioners are pressured to make quality data every day. So they look to maximize work output and how to integrate their people, technology, and processes toward various issues. If you do not catch data quality issues early – such as inaccurate data, schema issues or hidden/ duplicate data, it simply costs you real money in both finding and reconciling data, plus lost time on managing data vs. using data. It's beyond just data downtime – often only a metric of what happened, not what might be possible with great data.

The term "Data Observability" is about a parametric view across IT systems, instead of only data workflow monitoring. In real life this must include metrics and analytics to measure how investment in tools saves time and money. Beyond the concept of data downtime, Data Observability can solve real-life data problems with action and true reconciliation.

But What About Data Downtime?

Data downtime is an important metric that measures a Return on Investment (ROI) for spending money on data quality and observability. It refers to periods of incompleteness and inaccuracy in your data, and factors such as bug fixes or sudden schema changes are major causes that are detrimental. Some estimates say that for every hour of data downtime, it costs them more than $1 million. Data leaders get this.

They also know that the extent of data downtime is directly related to the complexity of the overall data system — as the organization grows and adds more applications, data downtime is likely to occur more frequently. Because data teams often approach data quality and lineage issues as they arise, so data downtime remains problematic unless data leaders collaborate closely with data quality and governance teams.

Pick any percentage from industry analysts on pain of non-accessible data – it usually tracks over 50% and millions of dollars when data quality suffers. Enter Data Observability to give meaning to real-life data management by using software tools and solutions such as automated monitoring, root cause analysis, and data lineage to help ease the pain.

Data Observability also provides the visibility to catch data issues early, minimizing data downtime and resolving the problem's root cause before it causes further issues. The investment helps keep your data up and running in the present and prevents issues from in the future.

Thinking About Data Observability on a Holistic Basis

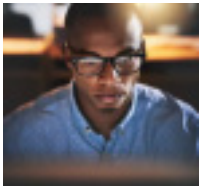
A more practical view of Data Observability is to increase visibility and monitoring holistically across the data workflow — from data ingest to transformation to consumption. Both data governance managers and data scientists can observe and monitor the state and quality of their data at all levels, but there is

more to consider for the goal of trusted data. Observability must be done at all stages throughout the enterprise.

Another deeper view of Data Observability is similar to the DevOps concept of software observability for processes that prevent data downtime. The key aspects of ensuring better data include: • Consistent monitoring of data across workflow for incidents or changes

- Identifying and tracking data using a Mean Time To Detect (MTTD)
- Resolving data errors and incidents on a Mean Time To Repair (MTTR)

Using these indicators, both an ROI and remediation plan can be put in place. This extends data monitoring into validation and reconciliation as data engineers assess what it will take to "fix" the data or accept the latest changes into the system.



Real Life – it must be throughout the enterprise. "It's important that trust in our data is shown throughout our systems, our client data, and for internal SQL processes. The platform must be technically astute – almost like it was created how I would write it myself. The impact is the data quality process does not limit our growth; it enables it."

In addition, DataOps and DevOps teams can use a continuous integration and development (CI/CD) approach to data and software engineering and provide a Return on Investment (ROI) to the cost of doing so. The payoff has a deep impact, and the tools can often automate each step through machine learning to discover and assess data problems that may otherwise remain hidden.

What's Data Observability Without Validation?

At the core of real-life implementation of Data Observability lies the combination of business and technical data validation. On the business side, you will need to check for completeness, fit for use and on the technical side, you will need to consider anomalies and other aspects outlined below. Data Teams look for validation in these areas as they perform their jobs, such as:

- Consistency or code validation ensures data styles are consistent. Validation is a core element for data quality and hence, data observability.
- Data type validation: each field, column, list, range, or file matches a specified data type and format. • Constraint validation: data field input fits a specified requirement within certain ranges. • Structured validation: ensures data matches a specified data format, structure, or schema.



Real Life – strong validation capability. "Our software partner has the same approach to flexibility and performance as we have to data. We chose our tool for all our data validation requirements and a team who provided support at every stage of the process, primarily when there were technical challenges in implementation."

Data Governance is Critical to Data Observability

THE hottest trend for people in data today is the significant role that practitioners in data governance play. Beyond access management or rules-based processes, data governors look to manage many aspects that drive Data Observability under a data governance framework.



These include:

- Defining specific use cases: Data Governance impacts the enterprise through cost, revenue, and risk. This allows stakeholders to help develop a framework and see the impact of Data Observability from a people metric viewpoint.
- Quantifying framework value: Data governance implementation must be quantifiable for the organization. KPIs can be created, and reports sent daily for impact. Data Observability tools improve those metrics by eliminating data silos, leading to productive teams and healthier pipelines.
- Enhancing data capabilities: Data Observability maximizes insights from data by enabling collaboration across organizations, identifying and resolving data issues, and helping to defining metadata such as content, classification, and origin. This is one aspect of Data Mesh as well from a domain perspective.
- Creating scalable delivery models: Data Governance frameworks must be scalable to add use cases and to show real-life ROI and new opportunities using data observability.

Without Data Observability, getting a data governance framework up and running is more challenging, especially if you plan to implement complex applications.

Putting It All Together for Data Observability

Experienced data leaders and practitioners know the most well-run data systems balance the art and science of data management daily. There's also the interplay among the data structure, the metadata context, and the impact of the final trusted data set.

In real life, a special group of data people enjoy the hard task of inspecting the data, understanding its state and deepening Data Observability beyond the data downtime metric. They know they enable success if they can bring trusted data to the decision-makers and business users.

When new code or applications move from devops to dataops, that is a critical time because teams must deploy data into a relatively stable environment and then metrically observe which changes occur to disrupt the current operation.

Without guessing, a system must be observable to the degree of the new state. This has to be deeper than dashboards and monitoring; advanced analytics and augmented data quality can be used to understand more. Observability is beyond telemetry or data downtime predictions, it's real life.



Real Life – operational value. "Our data software partner understood what it meant to go live operationally – we needed greater monitoring than before to monitor ETL logs and provide process protection and remediation if any of the data failed; it was beyond monitoring and important to the DevOps to DataOps cycle."

What's Needed Going Forward

Complex workflows and foundations of Data Quality and Data Observability implementations have been crafted to ensure reliable and consistent data based on business rules. With volumes of data into the petabytes, often continuously, management cannot continue to be manually managed. Data leaders need more.

Software suites must enable trusted data management in one easy-to-use experience. This includes:

Handling the complexities of data. With the increase in data volume, data cardinality (unique datasets) and complex pipelines managing new data transformations, a platform must create continuous profiling, reconciling, and validation to feed data models with trusted data.

Feeds to analytics and machine learning. As data models feed machine learning algorithms, verifiable data quality at source or transformed data during workflow are critical for an SME to trust the data. This has to be integrated with data wrangling that ensures better data.

A simplified user experience. With an easy-to-use user interface and dashboards and graphs that vividly show metrics, a trusted platform can help all data practitioners track performance and improvement, all the way down the workflow line. The impact – transparency and collaboration to everyone in the data enterprise.

A Framework for Proactive Data Observability

Chapter 8



For getting data observability right, the holy grail is being proactive. This proactive approach enables getting ahead of issues rather than constantly working in break-fix mode.

But where do you go from there? We've put together these five steps that can help lead the way. And for each step, we will use the analogy of a cargo train to paint a clear picture of measuring your observability success.



Step 1: Understand Pipeline Execution Health

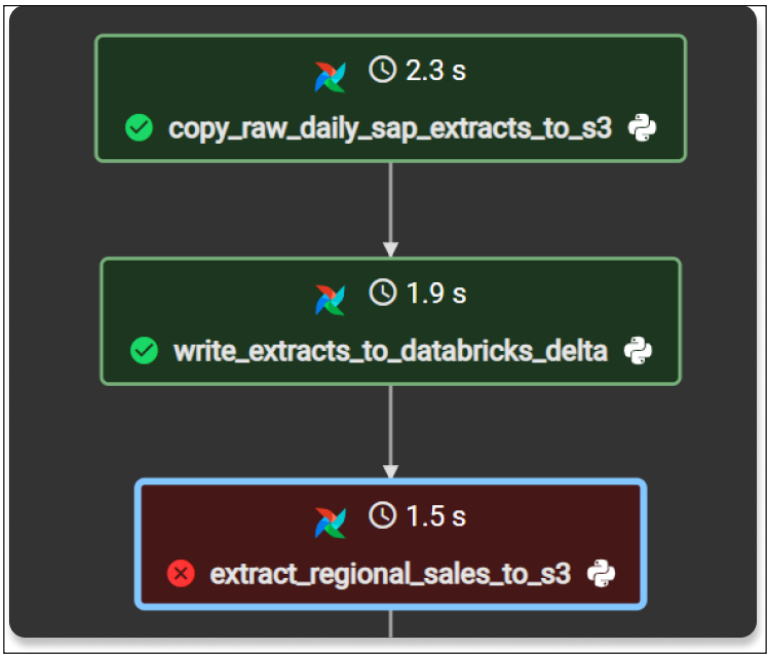
Has your cargo train left the station? If not, why hasn't it? Is it broken down or just not moving?

The first thing to understand is whether your data pipeline is executing correctly, as this impacts the flow of data. Specifically, a data orchestrator like Apache Airflow, IBM DataStage, or dbt will help create pipelines to run data through, but how do you know whether the data is moving through them as planned?

If the pipeline fails, no data can come through, and everything else that follows won't execute correctly.

This dashboard offers a clear way to visualize pipeline health and ensure your pipelines execute as expected. If they're not, it puts a full stop to all data reliability questions that follow, so having a quick and easy way to understand pipeline health is critical.

Beyond providing a clear view of pipeline health, this dashboard also makes it possible to quickly isolate where the issues are occurring so teams can immediately jump into fix mode.



Step 2: Alert on Pipeline Latency

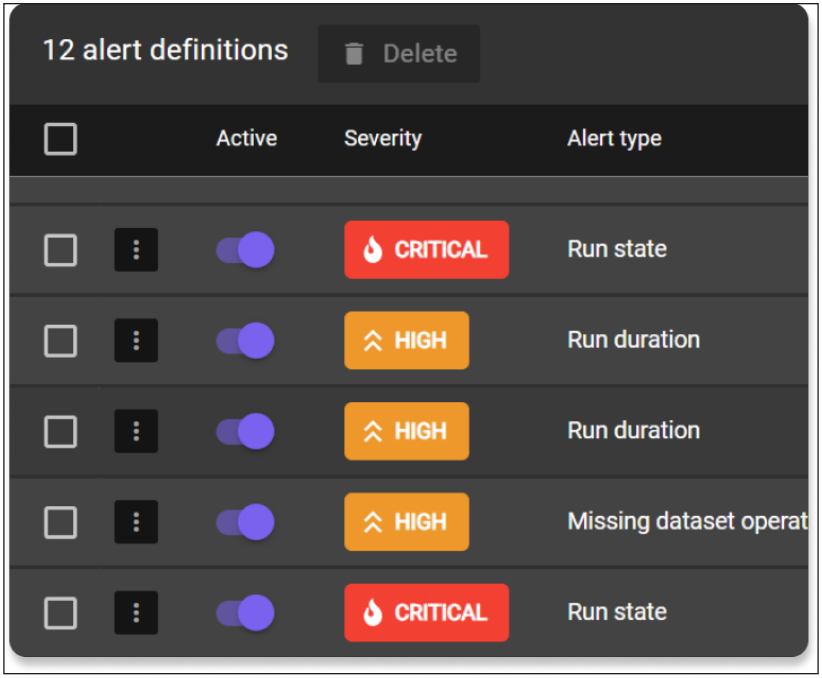
Did your train arrive on time? Or did it arrive too early, too late, or not at all?

Next, it's important to determine if data arrives in a usable time window based on when it's expected to come through. For example, a company expects to have data in hand for users at a certain time for them to make critical business decisions.

Any data that arrives late or not at all creates serious problems. Pipeline latency – or the time it should take data to arrive from point A to point B – is essential to nail down and maintain ongoing.

Monitoring pipeline stats like run count, average run duration, and total run duration can help address pipeline latency issues by keeping tabs on how often a specific process runs and how long that run actually takes versus the average run.

Comparing the two side-by-side can ensure the expected and the actual runtime align. If they don't, it clearly indicates an issue that requires further attention.



Step 3: Check Data Sanity

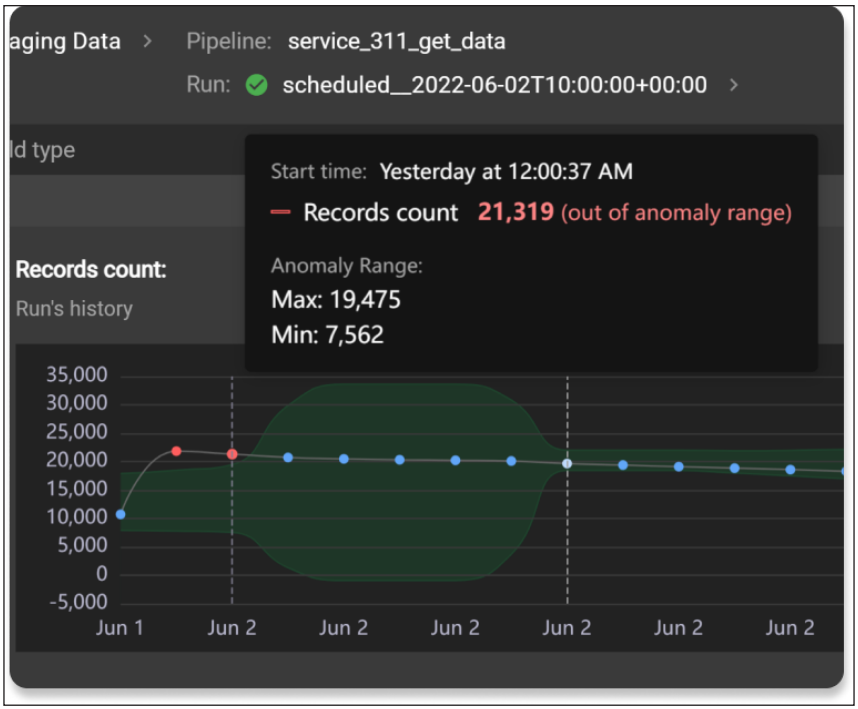
What cargo was on your train when it arrived? Was it the same cargo you put on and expected, or did you end up with more or less?

Once the data arrives, is it valid and complete, or are there any errors? This data sanity check is critical because even if data arrives on time, it does no good if it's ripe with errors.

Data sanity centers on any changes to the data schema, such as whether or not there are more or fewer columns than expected.

Checking data sanity at scale (think millions of data points moving through the pipelines) can get difficult, requiring the right team and dedicated technology.

Checking data sanity requires digging deeper into the data schema for any run. A dashboard like this provides an easy way to pull up each data schema and get a bird's eye view of the columns.



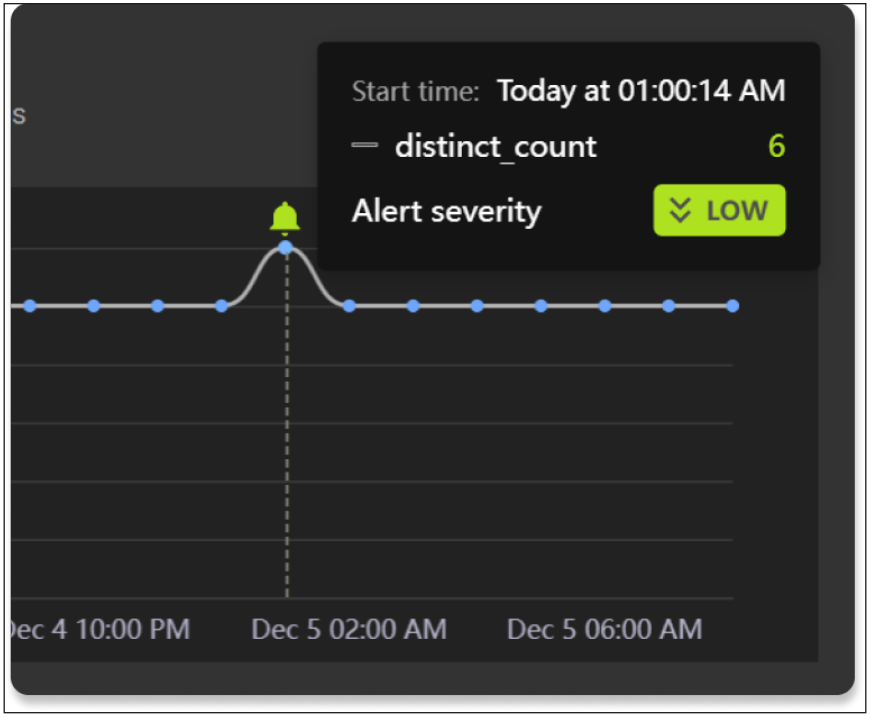
Step 4: Analyze Data Trends

Did the cargo on your train change during transit? Is it on the train as expected, but something about it differs from when it got on the train?

While data sanity looks at the data schema to confirm the data coming in is structurally okay, there's still more to consider. From there, it's essential to dig into the dataset trends to determine what causes these changes to keep happening.

For instance, what if one column is structurally right, but its data is wildly different from expected?

Narrowing in on changes to the data can help determine what within it is actually changing, including how any structural changes might affect the health of the data. This ability to look closely at not just how the data is changing but what within it is changing is important to understanding where issues are occurring and what needs to be corrected.



Step 5: Identify Data Access and Business Impacts

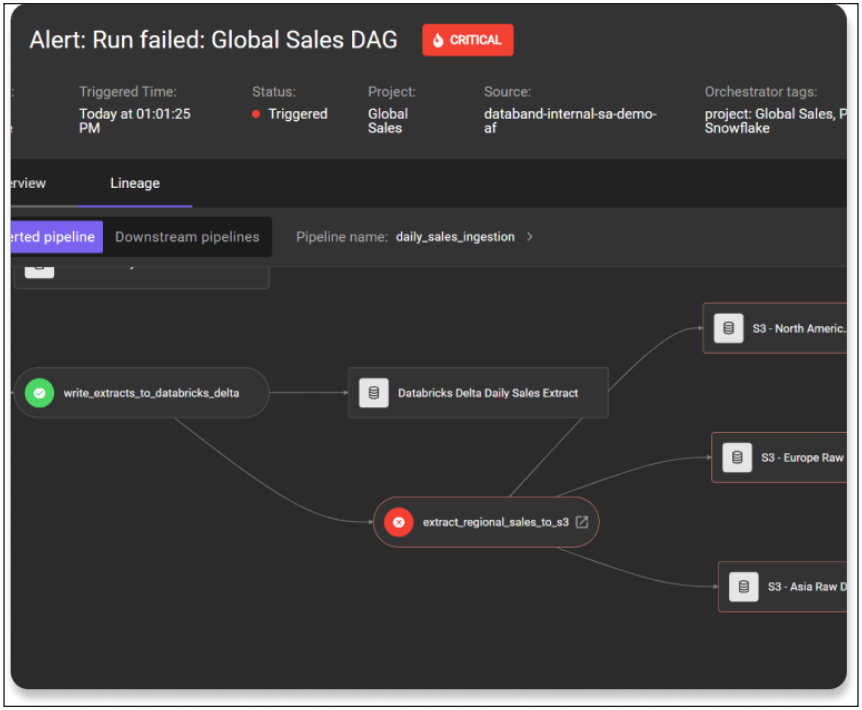
What's the final destination for the cargo on your train? How is it going to get there once the train stops, and how will any issues with the train itself impact its final arrival?

The last step brings everything together to identify the business impact of issues at any point by understanding how the data maps to key decisions.

Gaining this understanding requires investigating how the business uses each set of data. That way, it's easy to identify the downstream impacts of any issues in terms of latency, sanity, trends, or anything else.

A quick way to understand the data flow, for example, by looking at lineage, can help identify the end destination and the business impacts.

It's important to look closely at this view and related alerts to understand the different data sets affected by any issues associated with a particular pipeline or data run.



Top Data Quality Metrics for Data Observability

Now that we've discussed the five steps for being proactive, we frequently ask, "How should I measure my data observability success?"

We'll dive right in by sharing notable ways to measure the top data quality metrics for data observability. Remember, the area surrounding data quality metrics can be a touchy subject, especially within the focus of data observability. A quick Google search will show that data quality metrics involve many categories.

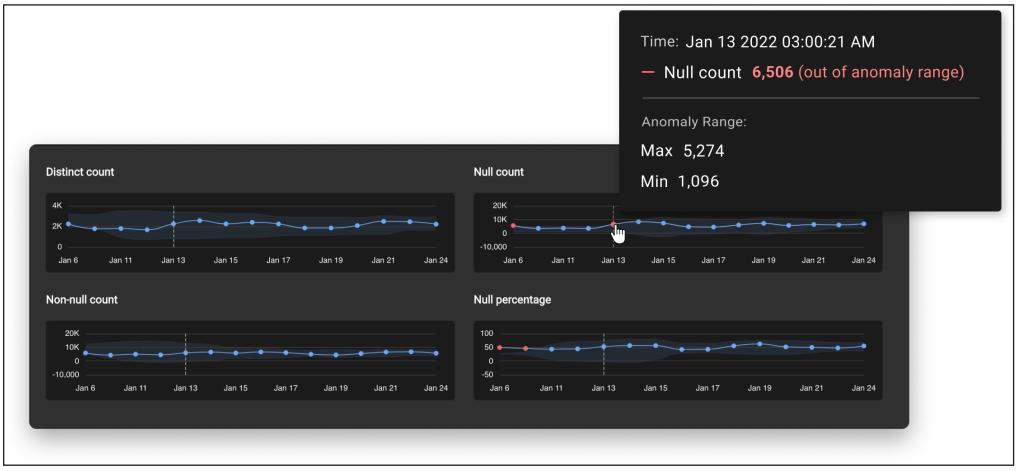
For example, completeness, consistency, conformity, accuracy, integrity, timeliness, continuity, availability, reliability, reproducibility, searchability, comparability, and probably 10 other categories all relate to data quality.

So what are the right metrics to track? Well, we're glad you asked. We compiled a list of the top data quality metrics you can use to measure the quality of the data in your environment.

Metric 1: Null counts

With null counts, Data Engineers and Data Analysts are particularly involved in it. Tracking this involves calculating the number of nulls, non-null counts, and null percentages per column so users can set an alert on those metrics. This is important because a null is the absence of value; you want to know any nulls that pass through your data workflows. For example, downstream processes might be damaged if the data used is now "null" instead of actual data.

Occasionally, the values of a column might be "dropped" by mistake when the data processes are not performing as expected. This might cause the entire column to disappear, making the issue easier to see. But sometimes, all of its values will be null.



Additionally, a column's data might slowly drift into "nullness." This is more difficult to detect than the above because the change is more gradual. Monitoring anomalies in the percentage of nulls across different columns should make it easier to see.

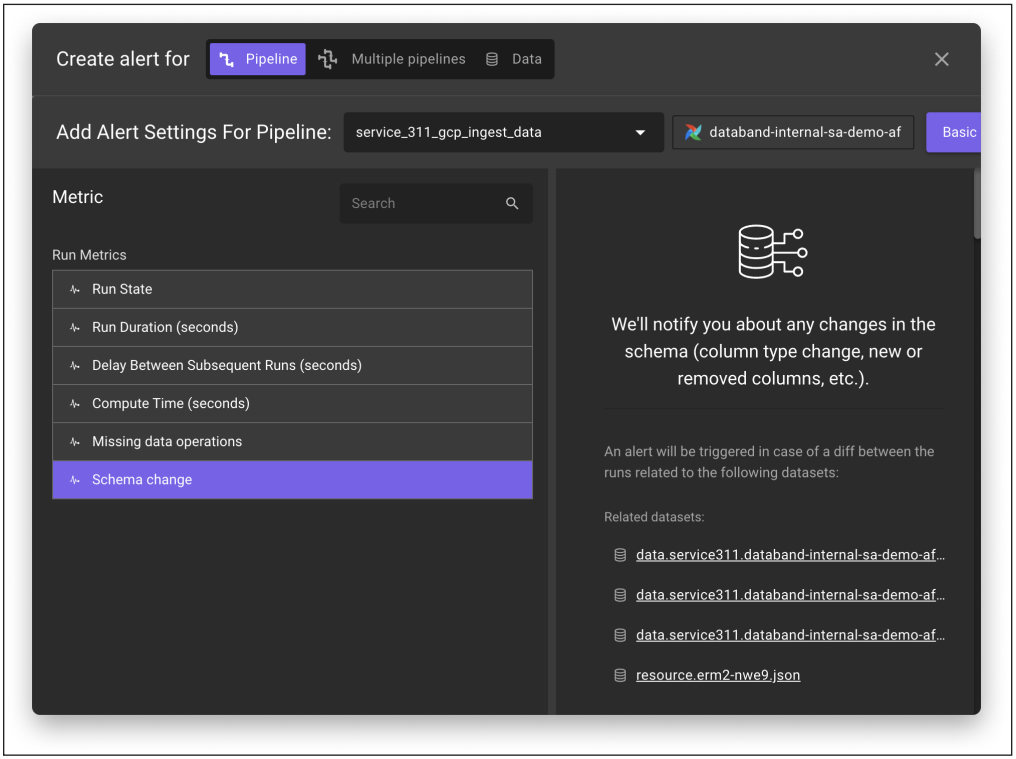
Metric 2: Schema changes

Data Engineers, Data Scientists, and Data Analysts alike manage schema changes. This involves tracking all changes in the schema for all the data sets related to a certain job. This is critical for data teams to monitor, as schema changes are key signals of poor-quality data.

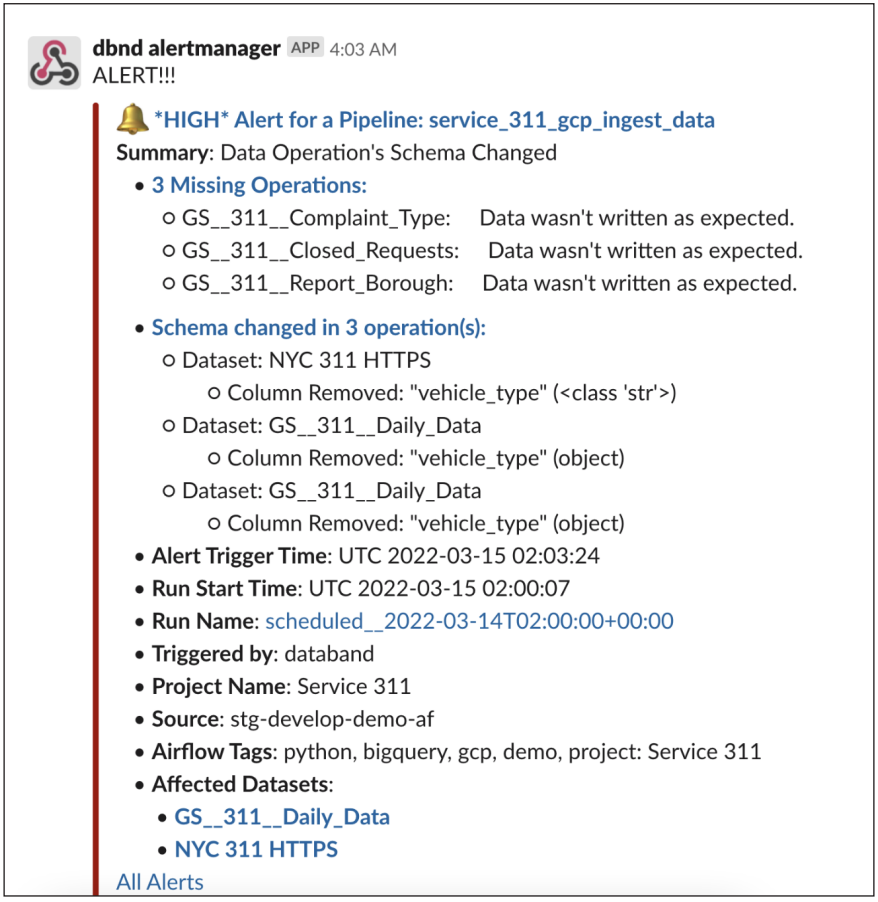
In a healthy situation, schema changes are communicated in advance. They are infrequent because many processes rely on the number of columns and their type in each table to be stable. Frequent changes might indicate an unreliable data source and problematic DataOps practices, resulting in downstream data issues.

Examples of changes in the schema can include:

- Column type changes
- New columns
- Removed columns

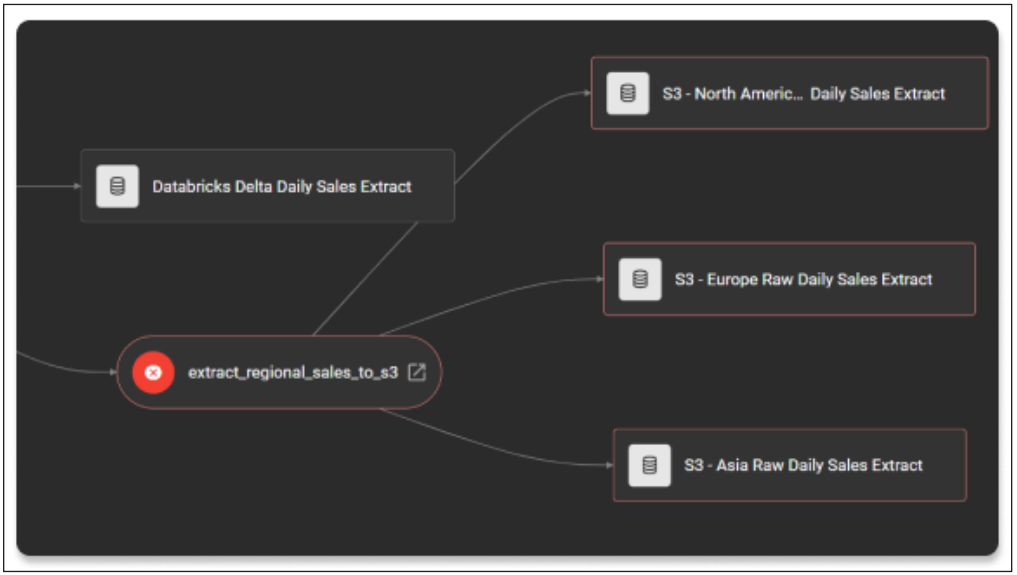


A clear way to counter this is to go beyond a good understanding of what changed in the schema and evaluate the effect the changes will have on downstream pipelines and data sets.



Metric 3: Data lineage

Data lineage is another process that Data Engineers and Data Analysts frequently monitor. For them to manage it, they must track the data lineage with assets that appear downstream from a data set with an issue. This includes data sets and pipelines that consume the upstream data set's data. This is critical because the more damaged data assets are downstream in terms of data sets and pipelines, the bigger the impact of the data issue. This metric helps data engineers understand the severity of their issue and how fast they should fix it.

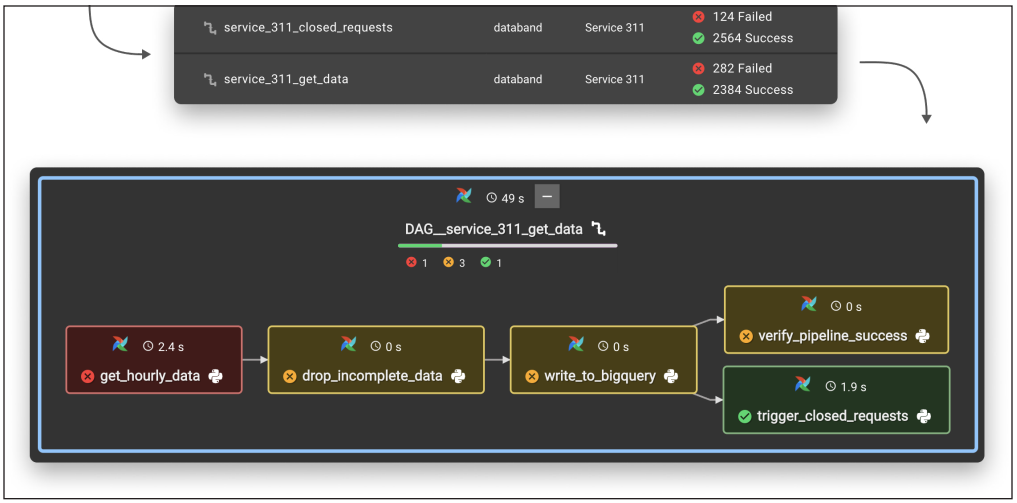


It's also an important metric for data analysts because most downstream data sets make up their company's BI reports.

Metric 4: Pipeline failures

Ask any Data Engineer or Data Executive about monitoring pipeline failures, and they'll all agree that it's one of the most important components of their jobs.

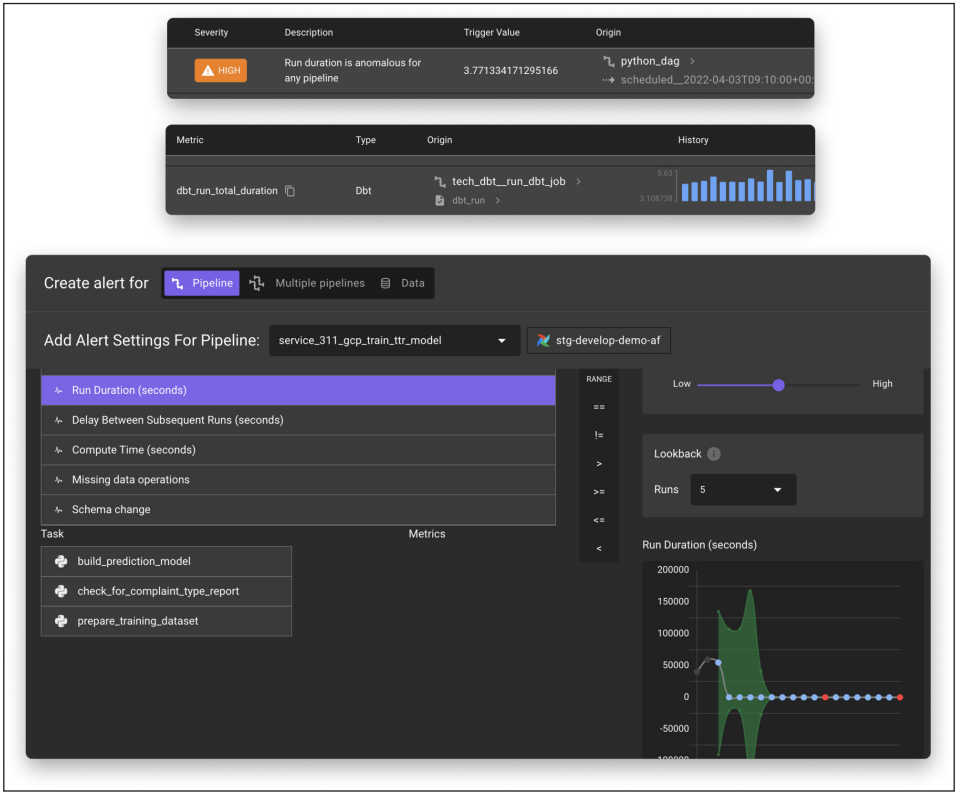
This consists of tracking the number of failed pipelines over time. To understand why pipelines fail, use tools highlighting root cause analysis and showing deep dives inside all the tasks the directed acyclic graph (DAG) contains.



Tracking this is essential because the more pipelines fail, the more data health issues you'll have. Each pipeline failure causes missing data operations, schema changes, and stale data. If you're experiencing many failures, this indicates severe problems at the root that must be addressed.

Metric 5: Pipeline duration

Data Engineers are heavily involved in measuring pipeline durations. The engineering team can track this with the Airflow syncer, which reports on the total duration of a DAG run, or by using a tracking context as part of a software development kit (SDK).



It's important to understand pipelines that work in complex data processes are usually expected to have similar duration across different runs. In these complex environments, pipelines downstream depend on upstream pipelines processing the data in certain service level agreements (SLAs). The effect of extreme changes in the pipeline's duration can be anything from processing stale data to failing downstream processes.

Metric 6: Missing data operations

Data Engineers, Scientists, Analysts, and Executives all have one thing in common: ensuring there are no missing operations. They do this by tracking all the operations related to a particular data set. A data operation combines a task in a specific pipeline that reads or writes to a table.

It's important because when a certain data operation is missing, a chain of issues in your data stack will be triggered. It can cause failed pipelines, changes in the schema, and delays.



Additionally, the downstream consumers of this data will be affected by the data that didn't arrive. A few examples include:

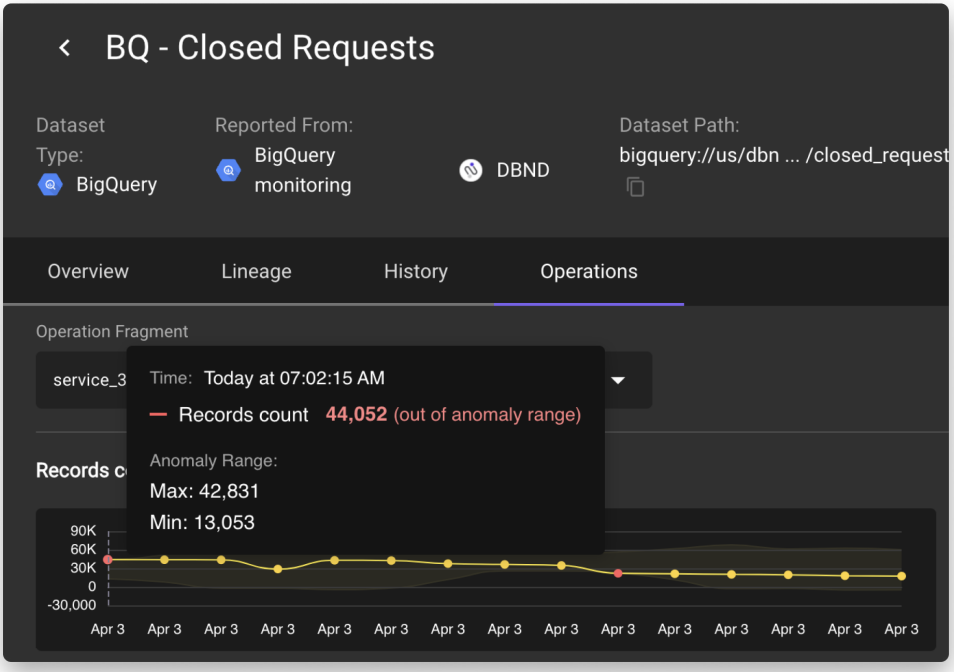
- The data analyst using this data for analysis
- The machine learning models used by the data scientist
- The data engineers in charge of the data

[Figure 8. Pipeline operations alert on Slack]

Metric 7: Record count in a run

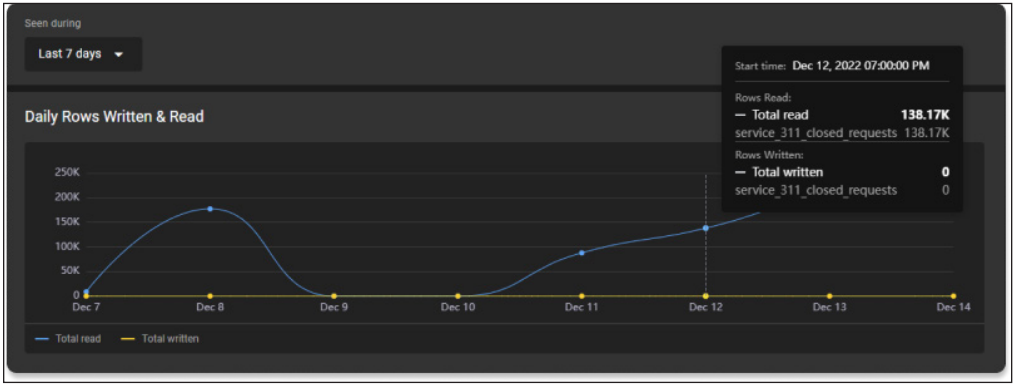
In addition to many tasks for Data Engineers and Data Analysts, one is to track the number of rows written to a data set when recording the counts in a run.

This is vital for tracking because a sudden change in the expected number of table rows signals that too much data is being written. Using anomaly detection in the number of rows in a data set provides a good way of checking that nothing suspicious has happened.



Metric 8: Tasks read from data set

Data Engineers are responsible for tracking the tasks read from data sets. The more tasks read from a certain data set, the more central and important that data set is.

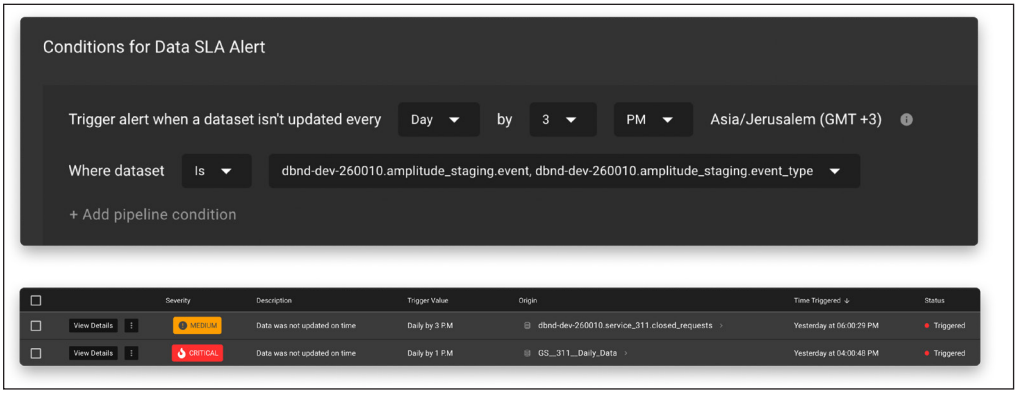


Understanding the importance of the data set is crucial for impact analysis and realizing how fast you should deal with your issue.

Metric 9: Data freshness

You can thank Data Engineers, Data Scientists, and Data Analysts for keeping data fresh. They accomplish this by tracking the number of scheduled pipelines that write to a certain data set.

Data that isn't updated as expected can wrongly feed downstream reports. This results in consuming the wrong information. A good way of knowing data freshness is to monitor your SLA and get notified of delays in the pipeline that should be written to the data set.



Get started

Implement proactive data observability today and know when there's a data health issue before your consumers do.

Discover how to get alerts on leading indicators of data pipeline health issues and drill deep into the data to implement a fix before bad data gets through.

Images: *https://drive.google.com/drive/folders/13ZzUTR_8BbUkoZTuEuEtWbYQ6Y8cujoY?usp=sharing*

Automation and AI Are a Must for Data Observability

Chapter 9

unravel™

Why isn't our data observability as intelligent as our data stack? Ironically, our modern data analytics systems can take petabytes of data from hundreds of disparate sources and turn it into practical insights, yet the way we monitor and manage those systems—data observability—doesn't do the same. Data observability as we know it today remains fractured and incomplete, taking us only halfway to where we need to be. It's the weak link in our data operations.

Automation and AI are needed to bring data observability out of the Dark Ages. The whole purpose of data observability is to help us quickly understand what's going on—and why—to either reactively fix a problem or proactively prevent an issue from becoming a problem. But truthfully, how quickly can we optimize data applications/pipelines based on those signals? Does data observability tell us "why"?

It might point us in the right direction—not enough memory has been allocated for a particular job, there's a problem with the code, the size of our dataset has doubled—but remediation can still be a lot of manual effort requiring time and expertise.

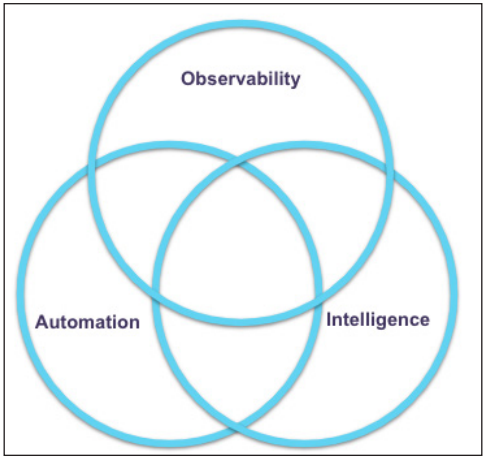


Figure 1. Data observability must be coupled with automation and intelligence (AI).

What we want to know is what we need to do next.

A more efficient approach is to automate the time-consuming manual detective work: continuous and automatic discovery of what's running in our data estate to govern costs, capturing and correlating workload profiles to choose the ideal instance type, mapping dependencies and lineage for data quality, assessing data layout and data skew for faster query performance, troubleshooting pipeline delays, and improving data processing efficiency.

AI enables us to "throw some math at the problem"—applying (hundreds of) machine learning and statistical algorithms to the observability data to identify and correlate patterns, detect anomalies, flag data issues, automate root cause analysis, and derive meaningful insights into emerging or existing problems.

Well-trained ML and highly sophisticated AI algorithms applied to a complete set of observability data can even go "*beyond observability*" to give us precise, prescriptive recommendations on what we should do to resolve the problem. For example, if there's one container in one part of one job improperly sized that is causing the entire pipeline to fail, AI can not only pinpoint the issue but tell us what the proper configuration settings would be. Or specify code changes to optimize a slow pipeline. Or optimize for cost by comparing the size, number, and resources needed to run the job vs. what is configured.

Automated AI-driven data observability cuts to the chase. Instead of relying on a handful of scarce and overwhelmed experts to tell everyone else what to do, the answers are served up to everyone automatically for an entire class of issues. This is enormously more efficient and the only way we can ever manage performance, quality, and cost at scale and speed within today's increasingly complex modern data stack.

But this approach requires a more comprehensive, unified, and holistic way to think about data observability.

Let's think differently about data observability

The term "data observability" has traditionally meant observing the data itself, but now must also include observing how the data is moving through applications/pipelines and how much it costs to do so. Briefly, data observability encompasses the dimension of quality, performance, and cost.

The term "data observability" has been around for only 4-5 years, originally coined as an umbrella term for harnessing all the information needed to understand the condition and health of data in our systems. And this is still very much a critical capability. We need visibility, context, and insights into what's going on with our data to deliver reliable results on time, every time. Nothing else really matters if the data is stale, incomplete, broken, or otherwise untrustworthy. Bad data in a good pipeline doesn't help anyone.

But neither does good data in a bad pipeline. We can have the freshest, most complete, highest-quality data in the world, but if it doesn't get where it needs to be on time—whether that's an analytics dashboard or an ML engine or another part of the application/pipeline further downstream—we've got a problem. We need to understand how the individual application/pipeline is running and how the entire data ecosystem runs end to end.

For many (if not most) problems, the issue is that there's a bad marriage between the data being processed and how it's being processed. Something somewhere changed that caused a failure or bottleneck, things aren't in sync, or the app isn't running the way it's supposed to. The dynamic nature of big data products/projects, their complex interdependencies with one another, with everything broken down into smaller parts processing in parallel, make it hard to understand what went wrong, where, and why. That's what sets off the blame game, with all the friction and finger-pointing among those who built the application and those tasked with running it (plus the frustration of people using the application's output).

These twin requirements—to deliver reliable results on time, every time—manifest themselves as SLAs revolving around performance and quality. That has been the data teams' mandate since day one.

But now there's a new kid in town: cost. As organizations increasingly move more data workloads to the cloud and their (uncontrolled) cloud data spending skyrockets and blows through budgets, the cost has become a co-equal consideration. It has become a first-class SLA, right alongside performance and quality. Good data in a good pipeline is no longer enough. Now we demand good data in a good pipeline at the most cost-effective price.

What are the 4 foundational areas of data observability?

Data observability cuts across four but interrelated contextual dimensions of data analytics. Historically, these different aspects have been treated as discrete domains, with different team members myopically focused on just one dimension or another.

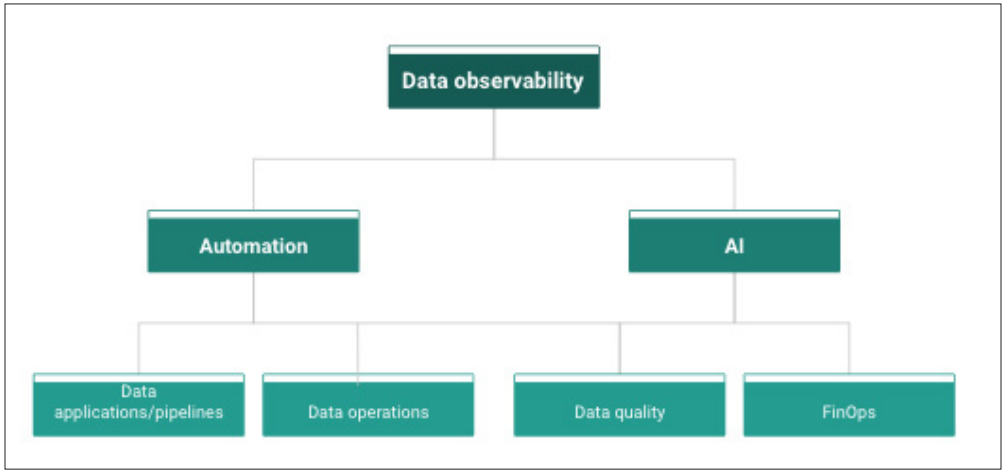


Figure 2. Data observability encompasses monitoring and managing data applications/pipelines, platforms, data quality, and FinOps.

- Data application/pipeline observability stitches together into a data-specific context millions of granular application and job-specific details that are "hidden in plain sight" but spread across disparate sources. This would be logs, metrics, traces, events, and other metadata from, say, Spark and Kafka and all the other dozens of components comprising the modern data stack, from the application down to infrastructure and everything between.

  This is basically the same observability that APM tools (Datadaog, Dynatrace, AppDynamics, New Relic) provide to software teams for web applications, but for data observability needs to be designed for the unique needs of data applications/pipelines.

- Operations observability enables data teams to understand how the entire platform, or "system of systems," is running end to end instead of pinpointing problems with one specific application. It provides a unified, holistic view of how everything is working together—both horizontally and vertically—within the Databricks, Snowflake, BigQuery, Amazon EMR, Dataproc, etc., ecosystems.

- Data quality observability looks at the health of the data making its way through the pipeline by assessing the integrity, accuracy, completeness, reliability, freshness, and consistency of data by capturing and correlating information related to data layout, schema, lineage, etc. It's observing the datasets themselves, running quality checks to ensure correct results, identifying where the data is coming from and where it's going, showing where it's used, who can access it, how it's being stored, etc.

- FinOps observability within the context of data applications/pipelines is all about using observability insights to enable engineering, business, and finance teams to work collaboratively to make data-driven decisions about their collective cloud spend. This covers everything from understanding with precision where the money is going and tracking budgets to identifying waste and inefficiencies to optimizing cloud operations for optimal ROI.

How data observability solves business problems

When considering why data observability is important—the value it delivers to the business—it's helpful to look at how observability has enabled software teams to deliver value to the business through the mature realization of DevOps.

Data teams today face many of the same generic problems that software teams faced 10+ years ago. The ever-increasing volume, velocity, and variety of work is overwhelming everyone on all fronts. There's too much work to be done and not enough hours in the day or enough expertise to go around. Troubleshooting a slow Spark or Kafka application or untangling the wires to track the lineage and dependencies of a data quality problem requires some advanced technical know-how. And there's no cavalry coming over the hill to help. The demand for skilled data professionals has always exceeded supply, and the greater complexity of the modern data stack is only exacerbating the chronic understaffing, talent shortage, and skills gaps. Data teams are spending too much time firefighting "by hand." A Gartner survey found that data teams spend only 22% of their time on innovation that delivers value. The rest is taken up by the monotonous toil of responding to trouble tickets, tracking down problems, and resolving them.

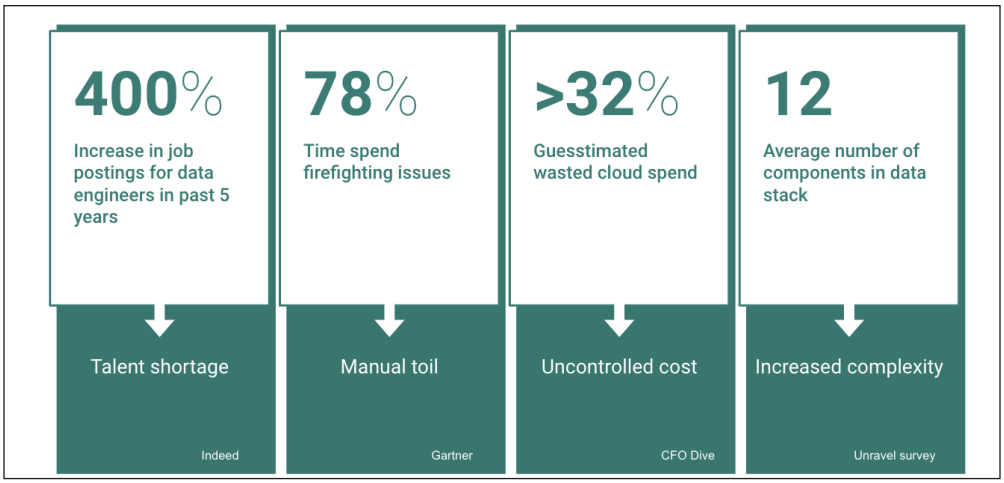


Figure 3. Business problems addressed by data observability

The combination of more work and not enough people creates a vicious cycle of complex issues landing in the lap of an already overburdened workforce without the time or expertise to deal with the problem quickly. We wind up with precious (and expensive) experts doing time-consuming break-fix work—the exact opposite of delivering strategic value. There's too much working in silos.

Data teams comprise multiple roles, and different team members use different tools for different tasks. Communication and collaboration among the team break down, and we're left operating in a Tower of Babel. Operational fragmentation and silos lead to friction and finger-pointing. Everything takes too long; trouble tickets pile up, SLAs get missed, productivity slows to a crawl.

And as more data workloads move to the cloud, the price tag gets higher and higher—seemingly with no end in sight. Modern platforms like Databricks, Snowflake, BigQuery, and Amazon EMR have greatly improved agility and speed-to-market, but it's become like the Wild West, where everybody is spinning up instances left and right but nobody has clear visibility into how efficiently they are running or control over their cost.

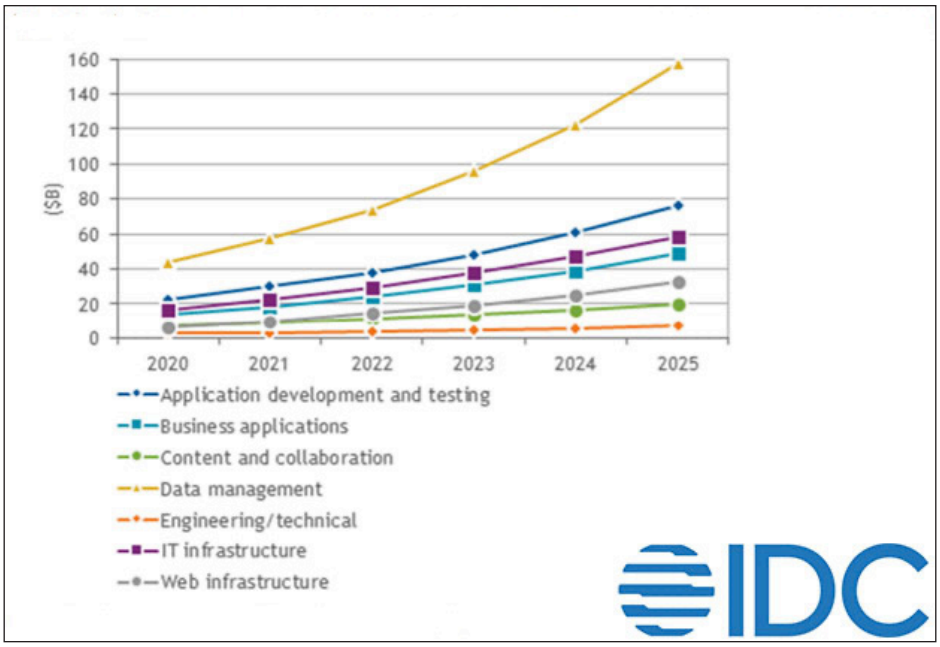


Figure 4. Data workloads are the fastest-growing cloud cost category and, if they're not already, will soon be an organization's #1 cloud expense.

The bottom line is that organizations aren't getting enough "bang for their buck" from their big data investments. The ROI from their people, processes, and technologies isn't where it should be. The goal of delivering reliable results on time, every time, in the most cost-effective manner falls woefully short.

Software teams have been able to overcome these challenges with their observability and application performance monitoring (APM) tools. Providing everyone—developers, operations, the business side—with holistic visibility into what's going on and analysis of why things happened fuels two foundational capabilities that are the bedrock of accelerating the DevOps life cycle. First, it enables a shift-left approach where individuals are empowered to self-service issues before they become problems. Second, for those issues that impact or need to involve other team members, it promotes a more collaborative methodology where everyone is looking at the same set of "facts" to tackle the task at hand.

As observability is the foundation upon which DevOps is built, data observability is the bedrock of DataOps.

The unique challenges for data observability

But while there may be generic similarities between DevOps observability and DataOps observability, data apps are a different animal from web apps. A common pitfall many organizations make is trying to "borrow" APM from their software teams to use for data observability. It seems that Datadog, Dynatrace, New Relic, AppDynamics, etc., do the same thing—but there are several reasons APM tools don't work for data observability.

First, data applications/pipelines are built on a different set of technologies, platforms, frameworks, and systems than web applications. Think Spark, Kafka, Airflow, dbt, Databricks, Amazon EMR, Dataproc, BigQuery, Snowflake.

Second, the nature of the work is different. Web applications are consistent products, and DevOps aims to accelerate the delivery of an improved version of that relatively consistent product. Data applications and pipelines start with fluid and ever-changing data and data sources, and the goal of DataOps is to accelerate the delivery of fluid and ever-changing data analytics requirements.

And there is the dimension of the data itself. In web apps, data is a static piece of information that traverses the service call chain, usually via a database API call. APM gives DevOps teams the information they care about—success/failure of the API call and how long it took—but that's about it. Data teams need to understand not only how the data flows horizontally from component to component across pipelines and vertically between data stack layers, but also the data's condition (or quality) as it flows. APM tools simply were never designed to comprehend this duality.

The different requirements for observing data applications/pipelines vs. observing software applications can be boiled down to three things:

- We need to capture very different types of telemetry data on a more granular level
- We need to visualize all this information in a context that makes sense to different data team members for their particular task at hand
- We need to do a completely different analysis to understand and fix issues

What data observability should look like

The dimensions of performance, quality, and cost are all interrelated, so it's important to view them holistically rather than as separate issues.

But make no mistake: moving to a shift-left and collaborative approach to holistically managing the performance, quality, and cost of data applications/pipelines is a seismic sea-change for data teams. We have to make it easy for every team member—and the easier it is, the greater our chances of success.

What is vitally important is that the information provided by data observability be consistent, accessible, and understandable across all data team functions: data scientists and analysts, data engineers, operations teams, data architects, data stewards, business stakeholders, C-suite executives, finance teams, etc.

What must be avoided is hopping between tools, jumping from screen to screen, manually cobbling together information from various discrete sources, with different tools giving different answers to the same problem. All the tools must integrate and play well together. Everybody needs to work from the same data while looking at things through their own lens and understand how everything fits together—a single pane of glass for a single source of truth.

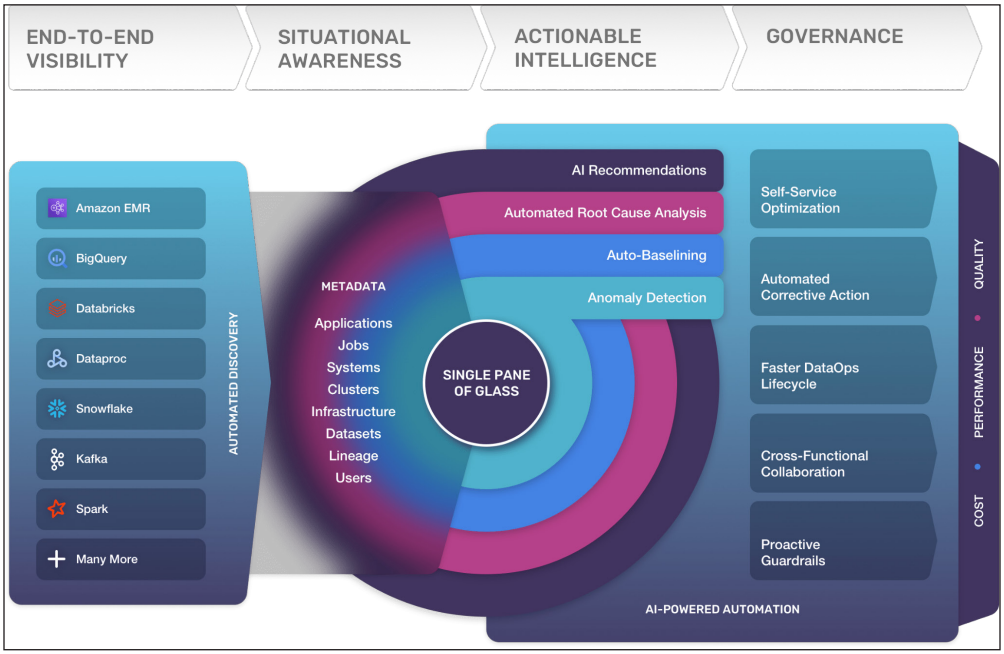


Figure 5. Automated AI-driven data observability

Automation and AI make this happen. First by providing end-to-end visibility and situational awareness, then by delivering actionable intelligence and enabling automated governance.

End-to-end visibility automatically stitches together telemetry data and metadata from the multitude of components, assets, technologies, and processes across the full data stack—both horizontally and vertically—to give a unified, in-depth understanding of the behavior, performance, cost, and health of data and data workflows.

This entails capturing information from engines, schedulers, services, and cloud providers about all data applications (pipelines, warehousing, ETL/ELT, machine learning models) and datasets (size and layout of tables, lineage, users, business units, computing resources, infrastructure), including code, configuration details, and metrics for each component in the stack.

Situational awareness puts all this aggregated information into a meaningful context for the task at hand. It's not enough simply to have a ton of granular details in dashboards, charts, and graphs, they need to be correlated, visualized, and analyzed so we can make sense of it all immediately. AI connects the dots among all the deep details about jobs, pipelines, clusters, data, infrastructure, and users into a single unified "mission control" view that shows at a glance all interrelationships (parallelism, resource contention, dependencies, lineage) and provides one-click drill-down into configuration, code, containers, resource usage, cost, data tables, quality checks, errors and logs, Gantt charts and DAGs, and more.

If we have all the details captured and correlated from everything we have running, we essentially have built a data model onto which AI/ML can be applied. But to get actionable intelligence with a high degree of confidence—i.e., to get answers, not just clues—we must have complete information about our data estate, with no blind spots. Otherwise, we're back to tackling performance, quality, and cost myopically as unrelated issues. And the observability solution needs algorithms and ML with a high degree of sophistication that have learned what the data applications are trying to do, how they behave, etc.

This actionable intelligence allows us to go from reactive to proactive—and get a level of automated governance via intelligent alerts or autonomous remediation action. Governance just converts AI-powered recommendations and insights into impact. The best way to reduce firefighting is to avoid having a fire to put out in the first place. Have the system apply the recommendations automatically. No human intervention needed. Policy-based governance rules could be as benign as sending an alert to a user (or send it up the chain of command) if some sort of data layout, performance, or cost threshold is violated—giving a heads-up that an SLA will get missed or a budget is in jeopardy of being exceeded—or as aggressive as automatically requesting a configuration change for a container with more memory or triggering preemptive corrective "circuit breaker" actions like shutting down a runaway job or rogue user.

Conclusion

Data observability is all about automation and AI. Using automation to capture and correlate millions of data points scattered across the highly complex conglomeration of technologies, platforms, frameworks, and cloud vendors provides a clear understanding of what is happening in our data estate.

Then using AI to figure out why something happened—and tell us what to do about it. And this needs to happen across the dimensions of performance, data quality, and cost. With the volume, velocity, and complexity of today's modern data stack, these aspects are increasingly interrelated—organizations can now address all three using automation and AI to realize the full potential of their data operations.

How Data Observability Optimizes Reliability, Performance, Operations, and Spend for the Modern Data Stack

Chapter 10

acceldata

Enterprise data used to be simple.

Most organizations relied on a few on-premises databases and applications, which provided all their necessary quantitative data such as sales figures, customer data, inventory levels, and financial results. While the environment wasn't exactly simple, it could be tamed, controlled, and it was generally reliable.

From that, data would be ingested — usually in the form of overnight batch jobs — into data warehouses using extract, transform and load (ETL) and other basic data pipeline tools. All the related metrics, reports, and analyses that needed to be performed were well-known and highly structured inside the data warehouse. Pipelines were built to closely align with this structure, and there was a sense of linearity to it all. Data sources didn't change much, so things didn't break.

Once complete, teams of business intelligence (BI) analysts generated weekly, monthly, or quarterly reports for their bosses. In some cases, they would tweak and update "live" executive dashboards, albeit with data that was already out of date.

The Data Stack Evolution

Things started to change fifteen years ago. It developed slowly at first, and then, over the past few years, extremely rapidly.

What happened? Data supply, tools and demand exploded. The digital economy hit its stride; ecommerce began to boom and technology evolved. SaaS applications for things like CRM and customer engagement, along with web services for targeted ads and marketing automation all drove more tools and more integration. Data sources proliferated like crazy and organizations had to now deal with getting a digital fingerprint on all their stakeholders.

All of a sudden, there were a massive number of new data sources: APIs, IT logs, IoT sensors, real-time streams, and many more. These manifested themselves as SaaS apps which were far more agile and useful, and ultimately, way more valuable in what they could deliver. Enterprise applications became more user-friendly, and as a result, it became more accessible and was distributed to more users. All of those users got more efficient at their jobs because they had control over their own work. That control was largely provided by data, both their consumption of it, and their ability to create it.

To enable this major digital transformation, new employees were brought on board specifically to caretake, analyze, and ensure smooth data operations. This formed the core of a new type or organization, the data operations, or DataOps, team. "Data" started showing up in job titles: Data architects. Data engineers. Data stewards. All of the people in these roles played a part in ensuring that data was usable and available. They had to make it show up where it was needed, and they had to ensure it was accurate.



To handle the continuous growth of their data, data teams embraced new cloud-native storage and repositories, such as data warehouses, data lakes, Amazon S3, and others. For many enterprises, the cloud was more scalable, efficient, and elastic than on-premises servers and data centers.

However, to keep existing operations functioning smoothly, new cloud implementations were integrated with legacy on-premises relational databases and warehouses. This hybrid architecture created greater complexity and continues to do so even now.

Data operations teams now oversee the building and management of a complicated, fragile set of data pipelines employing a wider variety of technologies than ever -- Extract/Transform/Load (ETL), Change Data Capture (CDC), APIs, event streaming, and more.

These all interface with an increasingly demanding set of analytics applications, including predictive analytics and AI.

Data Problems Result From Quality Deficits

This complexity and scale creates a whole host of challenges for data teams. As we know all too well, data quality can deteriorate due to structural changes, such as schema drifts or issues that form in the lifecycle of data pipelines. Add to that the proliferation of data from SaaS applications which created its own variances and uncertainties (and we're not just talking about schemas, but about the actual data itself). Those applications and all the web services and APIs kept changing as well, as per their design.

Additionally, the cost of storing and processing data can escalate beyond control or oscillate unpredictably.

To guarantee high availability and performance, data engineers came to realize that they needed visibility and control over their complex infrastructure of on-premise and cloud data storage connected to webs of data pipelines.

Now, there is no shortage of legacy data quality tools on the market. Some are bundled with or work only for a particular data platform, such as Hadoop, and only offer partial visibility into your data infrastructure. But to truly operate in the modern data stack, data engineers need a single-pane-of-glass over all data activity, but until recently, they were forced to adopt multiple monitoring solutions to deliver the insights they needed. They use these to analyze and reconcile large volumes of patchy, conflicting data, but they typically require significant manual work and time.

The bigger issue, however, is that data monitoring solutions lack the strong management, predictive analysis, and automation capabilities needed for efficient data optimization. In other words, these tools can't see and understand everything, so the "view" is incomplete. Lack of comprehensive visibility means the inability to anticipate issues and remediate as needed.

Legacy Data Monitoring Approaches Are Insufficient

Legacy approaches were built for the purpose of monitoring and delivering alerts. They don't provide much in the way of context or insights. The data quality is low and therefore can tell you that something happened, but can't include the what, where, and why of the issue. Monitoring is awareness, but it's not accompanied by much that is actionable.



Most of these legacy tools are designed for one specific layer of the stack or for a unique set of data use cases. For example, some of the APM tools are reasonably good at monitoring applications, diagnosing performance issues within those applications, and, when aided by AI and machine learning, even predict application problems before they occur. These tools, however, are not suited to meeting the needs of modern data teams, and here are a few reasons why:

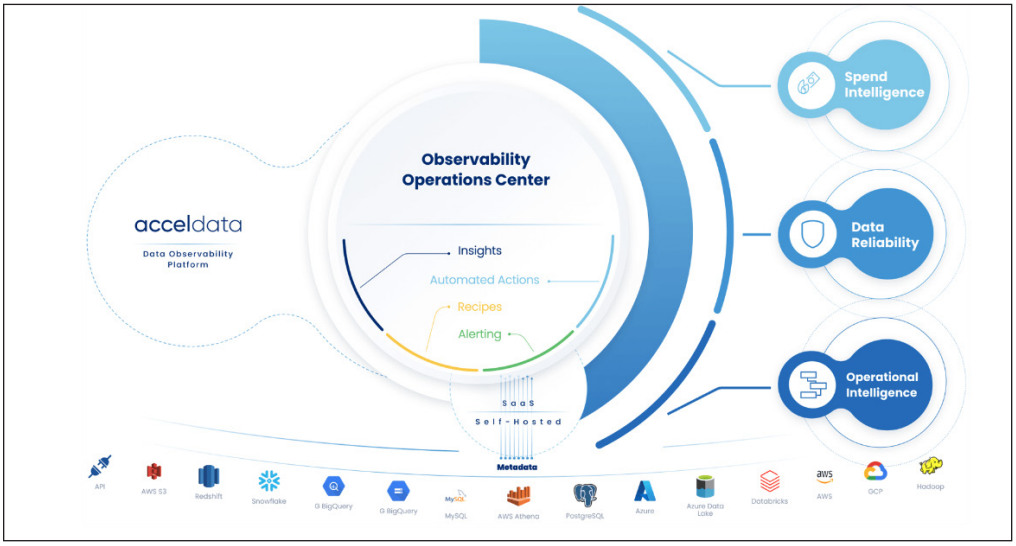
- **They create fragmentation:** Using different data technologies and solutions along the data lifecycle can cause data fragmentation. An incomplete view of data prevents data teams from understanding how the data gets transformed, thus causing broken data pipelines and unexpected data outages, which in turn requires data teams to manually debug these problems.
- **Their focus is on applications:** The ultimate focus of these legacy tools is usually on applications rather than data and data pipelines. They weren't developed to provide the in-depth visibility, control, predictive analysis, and time-saving automation that data operations teams need to efficiently maintain high data operations across the entire data environment. When enterprises rely on application-specific tools, data operations teams find themselves plagued with bottlenecks, slowdowns, and failures. Responding to performance issues becomes a daily firefighting ritual.
- **They lead to visibility gaps:** DataOps team members then find themselves in conflict with their business-side peers, for whom these analytics applications have become mission-critical. As more applications are added to the environment, and new data insight tools are used, there becomes a false sense of visibility when in reality, the focus these tools provide limit their comprehensiveness. In other words, gaps are created and those gaps lead to bad results due to inaccurate and unreliable data, and broken data pipelines.

Data is Data, But "Good" Data is Essential

The findings of a *study* that surveyed Fortune 1000 companies across 10 industries revealed intriguing insights into the relationship between data quality and business impact. The results indicated that an improvement of just 10% in the quality and usability of data could lead to a 16% increase in return on equity (ROE) and a yearly revenue boost of over $2 billion for the average Fortune 1000 company. Given the exponential growth of data collection, how can businesses enhance data quality on a large scale?

Manual efforts alone cannot address the challenge of improving data quality at scale for enterprise data teams. To achieve this goal, they require a sophisticated AI/ML-powered data observability solution that can detect and correct data drift, anomalies, and lineage issues automatically.

Data observability provides full data traceability with a single unified view (think operations command center) of your entire data pipeline. This can help data teams to predict, prevent, and resolve unexpected data downtime or integrity problems that can arise from fragmented data.



At its foundation, data observability serves as an all-encompassing approach to monitoring and managing today's complex data resources, operations, and pipelines. It helps predict, prevent and automatically resolve data performance problems and their associated AI and analytics workloads. It uses automation and machine learning to correlate – at massive scale – events across multiple servers, nodes, clusters, containers and applications.

Data observability provides a 360-degree view of data at rest and data in motion, data processing, and the pipelines through which data travels; in other words, all data, irrespective of where it exists and operates. It creates solutions and provides recommendations to optimize all aspects of enterprise data so it is accurate, consistent, and available. By assessing the accuracy and timeliness of data in transit or storage, data observability can evaluate the performance of data pipelines and their supporting infrastructure.

Consider what this all means; essentially, data observability enables DataOps and data engineers to be more effective managers of their data. Data observability allows data teams to establish and augment AI-based rules to detect schema and data drift, as well as other data quality issues arising from dynamic changes in data. This helps to avoid disruptions to data pipelines and ensures accurate data analysis. Moreover, data observability empowers data teams to automatically match data records with their sources and categorize large collections of uncategorized data.

With insights and AI-driven intelligence, these data practitioners and decision-makers can operate in a continuous state of data optimization because data observability provides them with the ability to do the following:

- **Observe:** Detect patterns of potential problems (including unknown unknowns) across complex data environments by analyzing the external outputs of data operations, including performance metrics, metadata, utilization, and more.
- **Analyze:** Infer from observations how to make data performance more reliable, scalable and cost effective.
- **Act:** Once you know what the issues are, you can manually take action or employ automated actions to fix problems before they impact performance, costs, and business results.

The most effective way to derive the benefits of data observability is with a comprehensive platform that helps enterprises optimize their usage of data as it relates to three key pillars:



Data Reliability

Historically, data processes developed to deliver data for analytics were batch-oriented and focused on highly structured data. Data teams had very limited visibility into the data processes and processing and focused their data quality efforts on the data output from the processes: data-for-consumption. These legacy data quality processes were:

- Were run in batch, performing semi-regular "data checks" weekly or monthly,
- Only performed your basic quality checks,
- Only ran on structured data in the data warehouse.
- Were sometimes performed by manual queries or "eyeballing" the data

Legacy data quality tools and processes had limitations of data processing and warehousing platforms of the time. Performance limitations constrained how often data quality checks could be performed and limited the number of checks that could run on each dataset.

To support modern analytics, data processes require a new approach that goes far beyond data quality: **data reliability**.

Data reliability is a major step forward from traditional data quality. Data reliability includes data quality but covers much more functionality that data teams need to support for modern, near-real-time data processes.

Primarily, data reliability takes into account the new characteristics of modern analytics. It provides:

- More substantial data monitoring checks on datasets such as data cadence, data drift, schema drift, and data reconciliation to support the greater volume and variety of data,
- Continuous data asset and data pipeline monitoring and real-time alerts to support the near real-time data flow,
- End-to-end monitoring of data pipeline execution and the state of data assets across the entire data pipeline to detect issues earlier,
- 360-degree insights about what is happening with data processes from information that is captured up and down the data stack to drill down into problems and identify the root cause.

Many data observability platforms with data reliability capabilities claim to offer much of the functionality of modern data reliability mentioned above. So, when looking for the best possible data reliability platform, what should you look for?

Traditional data quality processes were applied at the end of data pipelines on the data-for-consumption. One key aspect of data reliability is that it performs data checks at all stages of a data pipeline across any form of data: data-at-rest, data-in-motion, and data-for-consumption.

End-to-end monitoring of data through your pipelines allows you to adopt a "shift-left" approach to data reliability. Shift-left monitoring lets you detect and isolate issues early in the data pipeline before it hits the data warehouse or lakehouse. This prevents bad data from hitting the downstream data-for-consumption zone and does not corrupt the analytics results. Early detection also allows teams to be alerted to data incidents and remediate problems quickly and efficiently.

Spend Intelligence



In today's landscape, enterprises receive data from various sources in a variety of channels and formats. The volume of data flowing into each data warehouse or data lake can be unpredictable and fluctuate without warning, and as a result, as an organization expands its operations and adds more layers, such as customer-facing applications and verticals, data pipelines become more complex and harder to manage.

For businesses to derive value from data, they must extract actionable insights that drive better decision-making. To guarantee optimal outcomes, the entire data pipeline must be monitored meticulously. Given that each query is charged to the customer, any breakdowns or errors in data pipelines and ETL must be promptly addressed and fixed to maximize the ROI of data investments. Whether paying on demand or pre-purchasing capacity, data observability tools can help prevent waste by continuously monitoring and optimizing data systems.

Without the visibility into how data is functioning, and without understanding the cost implications of operating within a data environment, data teams run the risk of either overspending or underutilizing this crucial component of their data infrastructure.

A 2021 survey showed that 82% of organizations utilizing cloud infrastructure reported incurring unnecessary expenses. Additionally, Gartner predicts that 60% of infrastructure and operations leaders will experience public cloud cost overruns.

These surprise data-related costs occur due to the complexity of enterprise data operations, where even minor errors can rapidly escalate into costly, avoidable expenses. Data observability can help reduce downtime, software, and hardware expenses, and delivers over 80 times the return on investment compared to the annual cost of a full-time employee.

Look at it this way: *data pipelines become more fragile* as data footprint expands and data operations scale. If data teams can't predict and prevent data pipeline problems, they're just treading water, and as a result, enterprises face unexpected periods of data downtime, which contribute to customer churn and lost revenue opportunities. Data observability wires up the entire data infrastructure to predict and prevent data downtime problems.

Data observability offers a unified view of all your data systems across the entire data lifecycle. This can help you:

- Correlate errors with events across different data pipelines
- Identify data-related problems and investigate their root causes

- Improve data quality, optimize data pipelines, and reduce infrastructure costs as you scale
- Go beyond aggregate data to get better visibility into the health and reliability of data systems

When applied to the management of data costs, data teams can use data observability to optimize spending by providing visibility into how data resources are being used. This can include monitoring data storage and processing usage, identifying areas of inefficiency, and understanding the impact of changes in data usage patterns.

By using data observability to create spend intelligence, organizations can make more informed decisions about data resource allocation. The insights delivered to them about how data is being used and consumed can help them make better choices for cost-effective data storage solutions, reducing waste through improved data management practices, and identifying opportunities to reduce costs through the optimization of existing resources.

Because it is continuously assessing the performance of data-at-rest AND data-in-motion, data observability can help organizations track and manage data costs over time; it gives visibility into current usage and provides a historical view of spending patterns and trends. This allows organizations to make informed decisions about future data resource investments and prioritize initiatives that are likely to deliver the greatest return on investment.



Operational Intelligence

Data architects and engineers are tasked with creating data architectures that are high-performing, adaptable, and durable enough to meet Service Level Agreements (SLAs). To accomplish this, they must have a comprehensive understanding of their infrastructure's performance and usage patterns. For instance, inconsistently performing networks can affect the speed of ML models utilized for fraud prevention, resulting in longer wait times for customers and merchants during large transactions. That friction results in customer churn. It's a perfect example of bad data creating bad outcomes.

Data observability tools provide insights into data-related events and trends, assisting data engineers in building the appropriate infrastructure to support future analytics projects. They may opt to move merchant records to a cloud-based data store for faster fraud-check processing or establish a secondary CRM database to avoid resource conflicts between business analysts and financial analysts. Another option is to create a separate repository for inactive data. These are just some of the ways in which data observability empowers data teams with the information they need to design data architectures that will sustain their analytics projects and applications.

Data observability tools enable data architects and engineers to design infrastructures based on an analysis of performance and utilization trends. It's a critical component of data operations intelligence as it provides real-time visibility into data systems, enables organizations to make informed decisions, and supports data-driven strategies to achieve better business outcomes.

As the modern data stack has added more and more layers, ensuring optimal performance of each component, and the interplay among those components, is essential when applying data to business decision-making. Operational intelligence is the ability to monitor and analyze the performance of data systems and infrastructure, and is critical for operating the modern data stack. Data observability is an essential tool for improving performance intelligence as it provides a comprehensive view of data systems and processes, enabling organizations to gain insights into the performance of their data landscape and make informed decisions.

Data observability helps organizations monitor the performance and operations of their data pipelines, from data ingestion to transformation and delivery. It enables organizations to identify bottlenecks and performance issues in real-time, and resolve them quickly, improving the speed and reliability of data pipelines. This helps organizations improve the overall performance of their data systems, ensuring that data is delivered to stakeholders in a timely and accurate manner.

By providing real-time visibility into data systems, data observability enables organizations to identify and resolve performance issues, and supports data-driven strategies to achieve better business outcomes.

Data Observability Across the Entire Data Lifecycle

While the specifics may vary from industry to industry, all enterprise data teams need to work with several data types, sources, and technologies throughout the data lifecycle. For example, a healthcare enterprise may need to collect customer details directly via phone or their website for certain administrative tasks such as enrollment. At the same time, for billing, they may also need to work with external software, databases, and third-party payment processors. They may also need to work with social media, voice, and video customer feedback to gauge the ongoing quality of their healthcare operations.

So, enterprise data teams need to ingest different data types across a wide range of sources such as their website, third-party sources, external databases, external software, and social media platforms. They need to clean and transform large sets of structured and unstructured data across different data formats. And they need to wring actionable analysis and useful insights out of large, seemingly unrelated data sets. As a result, enterprise data teams can easily use multiple different technologies from ingestion to transformation to analysis and consumption.

Jamie Quint, a general partner at Uncommon Capital, explained his data and analytics stack in an *interview with Sylvain Giuliani*. His data lifecycle begins with getting data into a warehouse using Fivetran and Segment. He then transforms the data with Snowflake and dbt before going on to analyze the data with Amplitude and Mode. He finishes the data lifecycle by getting the data out into other platforms using Census.

Using different data technologies can help data teams handle the ever-increasing volume, velocity, and variety of data. The trade-off in using these many technologies is fragmented, unreliable, and broken data.

This is precisely why data observability is so crucial. It offers a single unified view of the entire data pipeline across different technologies through the entire data lifecycle. And it can help data teams automatically monitor data and track lineage. With a data observability approach, data teams have a foundation to ensure data reliability even after the data transforms multiple times across several different technologies.

Raising the Bar: Triangulating Observability, Quality & Reliability for all Data Stakeholders

Chapter 11

Qualdo™

The holy grail of "trust in data" from data to insights journey is not entirely new. But what does this mean in the context of evolving cloud data ecosystems? At its core, the approach to solving this complex riddle is driven equally by the strategic and operational needs of the enterprise. Success requires considering all four dimensions: **business, culture, process** and **technology**.

One common strategic goal in many enterprises is to satisfy the needs of business teams and the requirements of their applications with the right data and models "on time, every time." This strategic goal needs to be supported by:

- Structure and way of working of data and analytics teams (**Culture**)
- Data fulfillment process and measurement with SLAs (**Process**)

- Data governance and maturity (**Process**)
- Enabling stakeholders across data and business teams with tools and technologies that allow them to not just meet their expectations but also to set and share common expectations with each other (**Technology**)

Evolving expectations of data stakeholders

It is critical to evaluate the operational needs from each of the stakeholders in a team by directly considering their individual point of view. For example, the data team comprises various stakeholders with different competencies and expectations. This requires us to analyze their individual expectations on data quality, data observability, and data reliability instead of stereotyping them as one group.

Why should you care about data quality in the cloud?

Historically, data quality was a snapshot view taken periodically from a data governance point of view. However, modern data quality approaches enhance data quality, by measuring data quality performance and detecting data quality errors using statistical and machine learning approaches on a continuous basis. The advantage of using data quality metrics is that it introduces a set of standardized metrics to measure the health of data. Data quality metrics and frameworks introduce a common framework for multiple stakeholders to understand the nuances and issues in the data. Data quality performance and errors remain a relevant problem even today, especially when the data team is wall gardened from data providers (e.g., external systems or producers, such as different business lines supplying data to one global data team) in terms of what data is available or exposed. These scenarios highlight the need for a data quality framework and associated data governance as the need for understanding and fixing data quality issues can spread across different teams or business lines.

However, data quality metrics cannot acknowledge dynamic changes that occur, such as schema changes (as is normally the case) or failure paths (including pipelines) due to the complexity of the data stack.

"As the modern data stack on the cloud evolves from the Wild West to incorporating internal data standards and governance, data quality and monitoring is becoming a key enabler to align different data stakeholders."

Who needs data observability the most?

Data Observability is the next generation of data quality concepts and acknowledges the complexity and the point of failures in the data infrastructure and data pipelines. In simple terms, it is about creating visibility and traceability for the failures of a multi-component system to assist with troubleshooting. In the data ecosystem, this would extend to data pipelines, ETL jobs that do data movement to cloud, Data pipelines that assist in modelling and organizing the data. As a stakeholder, data observability serves primarily the data ops and data engineers, who are responsible for detecting and fixing data issues. In the process, this enables other data stakeholders to have visibility of the issues and understand the impact of by data issues. Data observability either includes or builds on data quality to deliver the metrics that need to be monitored from a troubleshooting point of view. Data observability is a much-needed weapon to help the data engineers tame modern data issues, However, It does not capture the expectations of the data consumers or business stakeholders.

"Cutting through the jargon, data observability delivers visibility of data issues in a complex data ecosystem and the impact of data issues."

Who suffers the most when data reliability is not guaranteed?

Let's talk about the last but most critical aspect: data reliability. The manufacturing domain has successfully used reliability concepts for decades. The concept of data reliability reflects the promises and agreements across stakeholders in the data ecosystem, especially for data consumers:

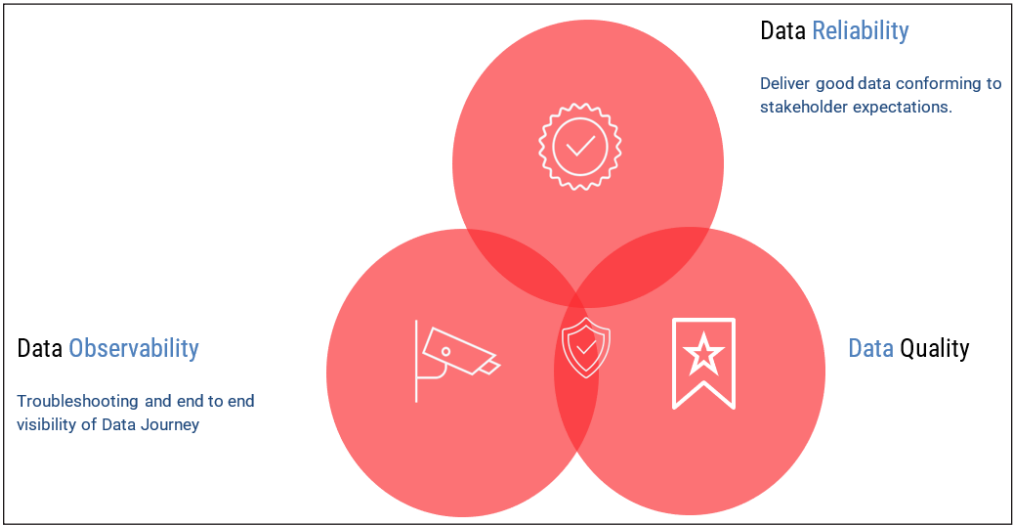
- Data reliability specifically focusses on the unknowns by actively monitoring the changes to data (schema or formats or drifts) changes to the data quality health metrics.
- Data reliability introduces specific metrics like number of incidents or issues per dataset or per consumer, or time to delivery of the latest data for the consumer.
- Data reliability focusses on the notion of an SLA that uses data quality metrics to establish and capture the expectations of data consumers and business stakeholders and measures them according to the defined data fulfillment process.

Data reliability builds on both data quality and data observability to deliver the expectations for **data consumers and business stakeholders**.

The interplay between data quality, data observability, and data reliability

Data observability and data reliability together enable the delivery and measurement of the "on time, every time" delivery.

"Data observability brings in the required intelligence needed to track and troubleshoot data issues. It enables a modern data quality approach towards governance and foundational for data reliability in managing stakeholder expectations."



Unifying data quality, data observability, and data reliability

Based on the above, we can now draw the following conclusions:

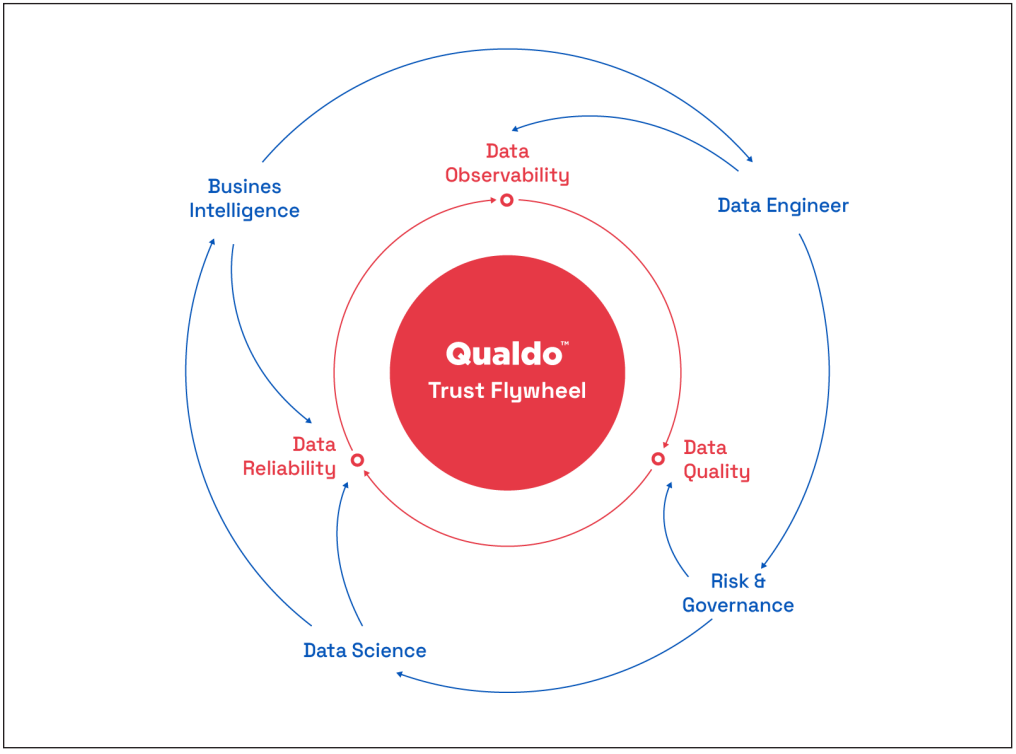
1. Any approach that does not solve problem for all the involved stakeholders holistically is doomed to fail in the long run.
2. Data observability, data quality, and data reliability are blood brothers and not just remote cousins. From all the pain points and use cases solved by these individual approaches, there is an obvious enhanced benefit in unifying all of them into one single framework.
3. This will lead to cost savings for the enterprise as a whole and will avoid fragmentation in this space, as enterprises will not want to spend on integration with multiple related technologies.

4. There will be a clear benefit to the enterprises with this merged approach and the distinction of where one jargon/use-case ends and another begins will become irrelevant. At the end of the day, data teams are only concerned about solving their day-to-day issues and other compliance/regulatory requirements.

5. A unified approach will ensure data gets delivered on-time and with high quality. This will be an ultimate bliss to all data stakeholders of an enterprise.

6. As described in the flywheel effect, it is not either one of them but bringing together all three, data quality, data observability and data reliability, that results in the required momentum in the journey towards trust.

The following trust flywheel maps all stakeholder expectations to the triumvirate of data quality, data observability. and data reliability.



Introducing a holistic view for multiple data stakeholders within an enterprise

From the stakeholder mapping, data engineering/data ops, applications, analytics/ data science, and BI/visualization are the teams that are involved or consume processed data and insights in an enterprise. Let us for a moment understand the lifecycle of what happens when something goes wrong in the production environment. Let us assume there is an issue in the production environment. If the issue is at the data layer and is not detected, it permeates other layers downstream.

Case 1: If a data pipeline or a job fails, this will involve straightforward troubleshooting and will lead to the data issue getting isolated.

Case 2: If the data issue impacts an application team, this will take certain amount of time to fix, but troubleshooting will be fairly straightforward.

Case 3: If a report on the dashboard fails due to a data issue, troubleshooting will involve BI and analytics teams responsible for the KPI as well as data teams responsible for quality. This is time consuming but will be deterministically solved as this issue is manifesting in a dashboard.

Case 4: If a model starts to go awry, this will manifest only over a period, this troubleshooting is a costly exercise. Sometimes the premise of the model itself will be questioned before the leads point to a data quality issue. Such a scenario encourages finger-pointing, leading to disastrous consequences, at times even at the organizational level.

The problem with SLAs

Most tier-1 enterprises after their initial failings on the cloud have implemented SLAs managed by the data ops team in their production environment. Due to this, data ops teams are on the radar every time something goes wrong with any critical stakeholder that is consuming data. For every issue, data ops have to review raw data and hundreds of pipelines to debug and troubleshoot the issue at hand, without any support from their various cloud databases and cloud data warehouses. The idea of data reliability is not part of this ecosystem yet. Hence, a manual scaling of data ops team is warranted, as data consumers grow. This also leads to tickets related to data issues growing with it. If only data stakeholders are empowered by tools to do their initial analysis of a reliability issue, this will continue to plague the production data ops team.

It takes three to tango

We can easily conclude that if an issue occurs with multiple data stakeholders such as Analytics or BI, to really understand and troubleshoot what has gone wrong, enterprises have to observe and track data reliability, model reliability, and dashboard reliability metrics. It really takes three to tango in this case.

The path of our predecessors

Unlike the case to merge data observability, data quality and data reliability into a framework, the case to solve data reliability challenges for multiple stakeholders should be looked at differently. Typically, this conundrum was solved by our software predecessors using the concept of 'edition' in the past. But where the problems of data teams end, the problems of other teams begin. The problems of the analytics team, for example, are their own problems and the problems of data teams as the data is their primary input. A data reliability edition targeted towards data engineering/data ops teams, a model reliability edition targeted towards analytics/data science teams, a dashboard reliability edition targeted towards BI teams had always been the need of the hour for enterprises operating data infrastructure in the cloud.

Raising the bar

This unified, and holistic approach of solving data reliability challenges across use cases and stakeholders in an enterprise environment will ultimately result in '**data trustability.**" This raises the bar not only for the organization but also for the products operating in this space.

Data Observability Anti-Patterns

Chapter 12

Joe Reis,
Best Selling Co-author
of Fundamentals of
Data Engineering

"All I want to know is where I'm going to die so I'll never go there." - Charlie Munger

Since you're reading a book on data observability, I'm guessing you have at least a passing familiarity with its purpose by now. While data observability is a key component of the health and reliability of your data, there are ways to get it wrong. Here are a few ways to fail with data observability.

Don't Get Buy-In From Stakeholders

Data observability should drive change management that increases awareness and action to improve data reliability and health. Every change management initiative's key factors include people, processes, and technology.

The most significant impacts are typically made first with people, then processes, and finally with technology. However, many data teams operate in reverse, over-indexing technology at the expense of "softer" areas like people and processes. You have a finite amount of social and political capital in any organization. You need to know how to extend this capital as far as possible and spend it where it counts. Sadly, many data teams blow their capital on a crusade to build a data stack while failing to get buy-in from stakeholders, which is almost always a fatal error.

A data team takes a technology-first approach to introduce a new data observability tool without buy-in from upstream or downstream stakeholders. The result is often a data observability tool operating in the background, but only some stakeholders outside the data team know or care. When stakeholders care, they're often annoyed and see data observability as extra work imposed on them by the data team, which is especially troublesome if your stakeholders aren't "data people." I've noticed that data is a foreign concept for most non-data people, and these people are often oblivious or skeptical about why data matters in their day-to-day work. If you implement data observability without getting buy-in from stakeholders, you're running the risk of future shelfware.

To keep failing with people and processes, avoid getting buy-in for data observability, which is the fastest way to burn through social and political capital. Don't develop rapport and communication with teams to your left and right. Do everything you can to create extra work for people. Show them no empathy.

But failing isn't about letting people down. A great head fake is pretending to be busy and useful.

Practice Data Observability Theater

Now that you've failed to get buy-in from your stakeholders, you've set up your data observability tool and collected information about your "data health." Whatever that is. It's time to practice data observability theater, meaning you go through the motions of data observability accomplishing nothing useful.

What does this look like? Monitor EVERY metric without having a good idea if they're impactful or not. Better yet, set alerts on these metrics and share them across all communication channels (email, text, chat, etc.), preferably to the entire company. People will see the data team is hard at work, doing...something. Nonstop notifications are a surefire way to annoy the living crap out of the people you worked hard to ignore in the earlier section. Extra credit if you want to be evil and set another metric for the amount of alert fatigue you're causing.

To keep up the performance art of data observability theater, when alerts flood in, ignore them and let them pile up. Set a metric for the total number of daily unresolved incidents, and bask in glory as you watch this metric grow!

Don't involve the application team who introduced the issue when your data pipelines break due to an upstream schema change. Instead, blame the application developers who created the mess you now have to duct tape. This is a great place to do this in standup and over public Slack channels.

Since you operate in a silo, ignore the cost of failures with data. Avoid silly stuff like the return on investment of fixing the root cause of a data issue. That's someone else's problem, and since nothing is your fault, it's everyone else's fault.

Finally, now that you've ignored getting buy-in, tracking the wrong things, and annoying your entire organization, you should fight yesterday's war. Pay attention to the problems that occurred. Ignore the need to anticipate problems in the future. And avoid addressing the root cause of why these incidents continue occurring.

Your Incident Response Shouldn't Fix the Root Cause

I hope you've done everything in your power to use my advice above to piss off everyone in the company to the point they're unwilling to help you. This is critical to not getting the support you need to address the root cause of the incidents your data observability tool flagged.

While the best way to fix a problem is at its source, you've got other goals. Data observability is helpful, but it's only one side of the coin. Good observability practices should include incident response. Knowing an issue occurs isn't good enough. Make every effort to solve the issue. Your data team's mandate isn't about solving problems when they arise.

Make sure your incident response includes lots of finger-pointing and anger. Don't use time-tested Lean techniques like Five Whys, as that's too effective at getting to the root cause of your issue. Instead, superficially address the problem. Again, since you don't have stakeholder support, you're free to guess the cause and solution to the issues about which your data observability tool is informing you.

Another way to fail with your incident response is to avoid seeing data observability as change management. Don't map out the value stream of your data, which would clearly show you where data adds value and bottlenecks to remove. Do the opposite and introduce as many wasteful processes and workflows as possible. While this will add to the confusion of identifying root causes, your goal is to fail with data observability.

You can't fix what you can't see! Create as much obfuscation as possible, making incident response and resolution of root causes impossible.

Here's to failure!

You've learned a few major ways to fail with data observability. I hope this has been useful for you. If you wish to fail, please follow these suggestions, and your data observability initiative will fail miserably quickly.

If you wish to succeed with data observability, study these suggestions and do the opposite. Be the rebel, take advantage of the cultural and organizational opportunities, do data observability as an active participant, and attack the root cause of issues as they arise. Do this, and you'll be in the top percentile of data practitioners. Don't tell your friends on other data teams because they might make you look bad. ;)

Links

https://lightstep.com/blog/observability-mythbusters-observability-anti-patterns

https://www.instana.com/blog/logs-are-an-anti-pattern-for-observability/

https://databand.ai/data-observability/

https://www.strongdm.com/blog/data-observability

https://towardsdatascience.com/what-is-data-observability-40b337971e3e

https://copyconstruct.medium.com/monitoring-and-observability-8417d1952e1c

^ landmark post

https://www.agileconnection.com/article/lean-anti-patterns-and-what-do-about-them

https://dzone.com/refcardz/quality-assurance-patterns-and-anti-patterns

https://www.netsolutions.com/insights/agile-anti-patterns/

https://www.apmdigest.com/skipping-application-monitoring-is-the-biggest-anti-pattern-in-application-observability

Notes

Get to the root cause of your issue

Hierarchy (databand article)

1. Operational health and dataset monitoring
2. Column level validation
3. Row-level validation

Detection, awareness, iteration (from databand article, below in the links)

DataOps shift-left

Take the practices of DevOps and DataOps.

Data observability is an extension of DataOps.

[tenents of DataOps from the book]

5 pillars (Barr's article below from 2020)

Data observability's newest frontiers: DataFinOps and DataBizOps

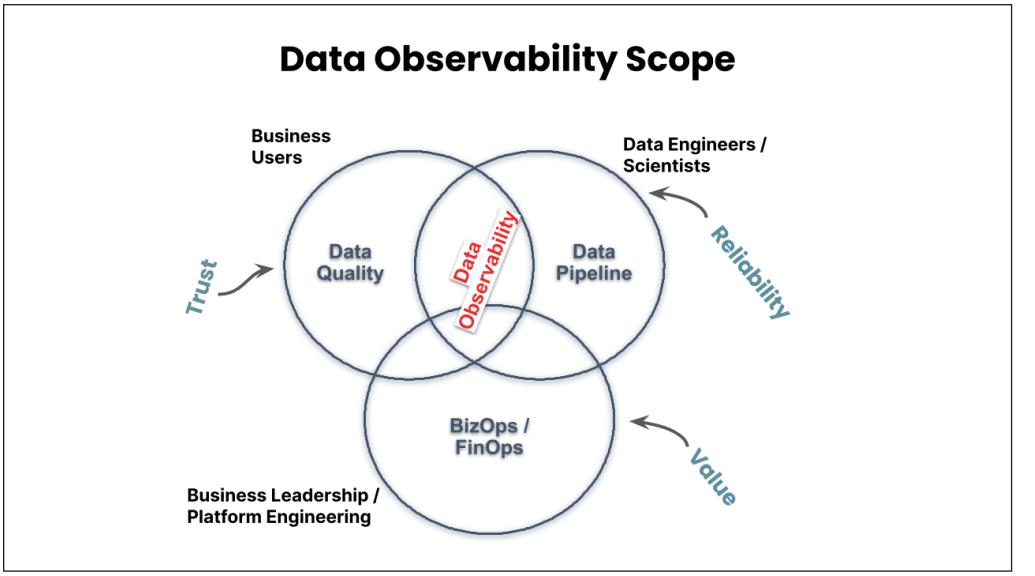
Chapter 13

Sanjeev Mohan,
Principal, SanjMo & Former Gartner Research VP, Big Data and Advanced Analytics

The sign of maturity for any new technology is when it goes from being a nice-to-have to a must-have. As we have seen in this book, data observability serves critical use cases ranging from data quality to data pipeline reliability. And, as we enter a new era defined by two major shifts – slowing economy and the rise of data products, new use cases have emerged. These pertain to helping improve business operations (DataBizOps) and control costs of our pipelines (DataFinOps).

Data observability is a key component of DataOps. It helps improve the speed and quality of data delivery, enhance collaboration between IT and business teams, and reduce the time and costs associated with data management. DataOps helps organizations make data-driven decisions faster and with more confidence. It provides them with a unified view of data across the entire organization and ensures that the data is accurate, up-to-date, and secure.

As data observability matures, we expect it will fold use cases like data security monitoring and protection into a comprehensive metadata platform. The following diagram shows the current scope of this concept.



Data observability is a multi-dimensional concept that addresses three areas: data quality, pipelines or infrastructure, and business operations including cost metrics. These three areas respectively deliver trust, reliability and value.

This chapter focuses on DataFinOps and DataBizOps.

What is DataFinOps?

The *FinOps Foundation* has defined best practices for cloud financial management (CFM) practices. When these principles are extended to data management, we call it DataFinOps. The financial metrics are tracked and monitored like data quality and performance metrics. And the data observability system recommends improving cost efficiency.

DataFinOps is a set of processes used to track costs incurred across the data stack and by different users and teams to optimize overall spending. Ideally, it provides a granular view into how costs are being incurred before they impact budgets and forecasts. Using the data observability capabilities of monitoring, alerts, and notifications, proactive actions can be taken to ensure that data pipelines run efficiently. The cost consumption is across granular or aggregate level for:

- **Resources**, like Snowflake, Databricks, Hadoop, Kafka etc.
- **Users and departments**, like business analysts, data engineers and data scientists
- **Workloads**, like data preparation pipeline, data quality, data transformations code or training models

A DataFinOps platform should provide a consolidated set of capabilities:

- **Discovery** (or observability) of the costs and spend with an ability to tag and categorize costs based on the organization's desired taxonomy. It helps to track, visualize and allocate costs. Tracking should be precise and use workload-aware context.
- **Budgeting**, **forecasting** and a **chargeback** (or optimization) of costs. It helps to eliminate inefficiencies and optimize cost. It uses AI to predict capacity rightsizing and automation to trigger preemptive corrections.
- **Recommendations** (or governance) on cost reduction strategies and opportunities. It helps to identify guardrails needed to control and prevent cost overruns. This could be across infrastructure resources, configuration, code or data. For example, datasets with low usage could be candidates for a cheaper storage tier.

When data observability is used for quality or pipeline reliability use cases, the personas are business analysts, data engineers or data scientists. For the DataFinOps use case, the primary personas are executives, such as the head of the cloud center of excellence or the CIO. These executives work with chief data officers (CDO) or the head of analytics, who manage data assets.

Interestingly, the fiscal discipline is now being pushed down to data engineers. DataFinOps practitioners inculcate a culture of financial discipline and facilitate collaboration between business, engineering, operations, and finance teams. They help improve cloud efficiency across various business teams through cost and efficiency metrics, such as resource utilization and trend analysis.

Why do we need DataFinOps?

Data is a victim of its own success. Data has gone from derided as the "exhaust" of applications to being a kingmaker. It has gone from being used by a few specialized analysts to a wide range of consumers who throw ever new use cases at it. In addition, the average number of sources that generate data has also increased significantly.

Imagine, we have gone from a few dozen data sources, like ERP and CRM, all within our firewalls to an explosion of SaaS products. *Recent studies* show that enterprises on an average used 110 SaaS products, and large companies now have close to 500.

On the one hand, we are dealing with a larger number of data sources, more data consumers, and more use cases, but on the other hand, we've also had a stampede of new tools and applications processing our data. These tools are no longer confined to our firewall-protected environments but are found on the edge, in private clouds and in public clouds. The data pipelines have grown complex and unwieldy. Organizations are now keen to understand the financial metrics of building and operating data and analytics workloads.

This scenario is the very essence for the genesis of the data observability space. Data engineers monitor the data and the pipeline to quickly detect and analyze quality and reliability issues. However, the cost of running the pipelines has never been in their scope, until now.

The economic downturn of 2023 has hit the tech sector the hardest. As infrastructure costs rise, management is keen to prioritize their spending. But how does one prioritize without clearly understanding the costs incurred at various data pipeline stages? Organizations estimated their budget and allocated capital expenditure (CAPEX) and operating expenditure (OPEX) towards their new IT initiatives. Cloud computing has upended this model by shifting most costs to OPEX. With this shift, many teams find out that the money allocated for the entire year has been consumed in the first quarter. We are now dealing with two problems - rising costs and unpredictable costs.

Studies have shown that 30% of the cloud cost is simply wasted. This is low-hanging fruit for a team looking to pare down redundant expenses. It is relatively easy to use native cloud provider cost management tools to identify unused instances and shut them down. This itself can save significant expenses. But it gets harder to pinpoint inefficiencies in complex SQL queries, ML training and data transformation workloads. Hence, DataFinOps needs a more concerted effort.

Data engineers embracing the "shift-left" approach can yield a much more proactive approach to cost containment. Sometimes rising costs are a harbinger for incorrect data or pipeline defects. So, the DataFinOps use case of data observability complements its other well-known use cases of data quality and pipeline reliability.

Challenges of DataFinOps

Cloud computing is celebrated for developing reports and dashboards infinitely faster. Within a minute, we can spin up a virtual warehouse in Snowflake and use its Zero Copy Clone to analyze the data. Similarly, we can use Terraform to spin up cloud resources on demand. But this ease results in a complex web of cloud costs.

Take the example of Accenture, which has over 1,000 teams using AWS. Its cloud bill runs into tens of millions of *lines of billing data*. The reason is so many cost elements in the cloud, such as compute instances, compute transactions, storage, storage I/O, data transfer, support, networking, monitoring, disaster recovery, and various other costs.

The second reason for cost complexity is because of the vast expansion of choices in the cloud. For example, as of February 2023, *AWS offers* 536 instance types running Linux and 427 running Windows. This leads to the challenge of calculating the cloud's total cost of ownership (TCO) and return on investment (ROI). Hence, an automated tool, such as data observability is needed.

Another challenge in data concerns the lack of ownership of data. As data journeys from the producer and goes through various transformation steps, it is unclear who owns the data. Earlier in this section, it is a cinch to spin up Snowflake compute resources, the lack of ownership leads to a lack of accountability.

Finally, cloud providers allow alerts to be set up, but sometimes by the time the administrator can act, the damage is done, and the high cost of the erroneous query is now on our books. Hence, a proactive approach is needed, which is what DataOps provides.

DataFinOps Case Studies

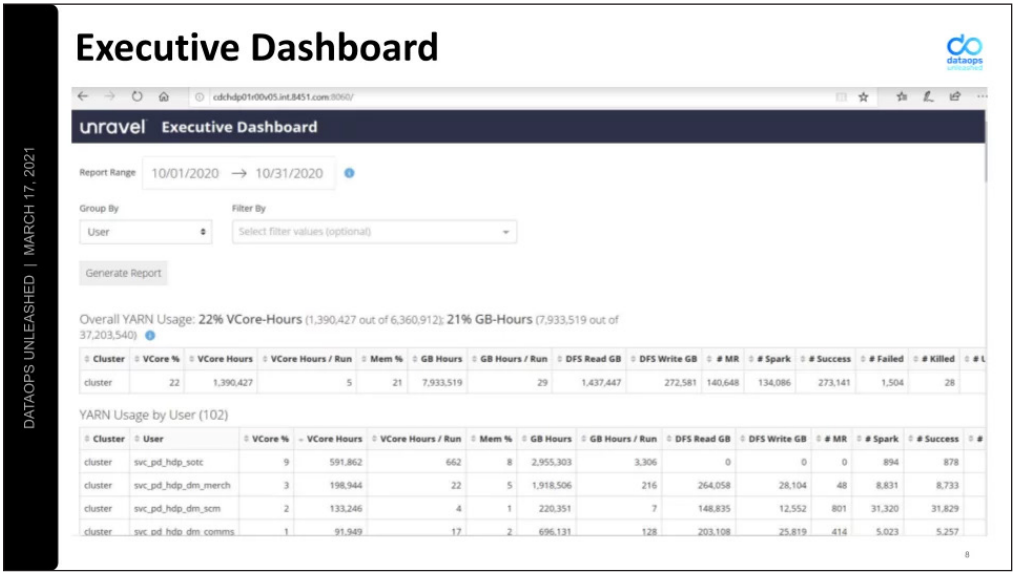
In this section, we will look at some successful customer case studies where the DataFinOps capabilities have delivered tangible value to the business. Let's first create a baseline as to what should an organization expect their DataFinOps to uncover:

- What parts of the pipeline are experiencing costs higher than the historical trends?
- What will be the impact of disrupting the job causing runaway cost?
- What are the cost trends, and when will we exceed the budget?
- Will the cost saving offset the cost of procuring and running the data observability tool?
- Will it provide a fine-grained cost breakdown into jobs and their consumption?

The first case study is for the US's largest supermarket retailer, Kroger. It has a retail data science wholly owned subsidiary called 84.51 (named after the longitude of its headquarter city, Cincinnati, Ohio). This organization provides retail analytics to its parent company, suppliers, and partners. One of its tasks was optimizing spiraling IT costs, consisting of an on-premises Hadoop ecosystem and Oracle Exadata. Using the newly gained DataFinOps capabilities, 84.51 has migrated some of its workload into the cloud. They use a combination of Google Cloud, Snowflake and Databricks for the persistence of data.

84.51 utilized Unravel Data's data observability suite to profile and identify inefficiently tuned Spark configurations. After initial success at pinpointing misconfigured clusters causing bottlenecks, the organization has now integrated Unravel into its development process. According to Jeff Lambert, VP of Data Solutions, *a key challenge* is ensuring these tools' adoption. His team had to ensure that DataFinOps capabilities were embedded into the developer ecosystem and that the notifications were handled promptly.

The figure below shows a snapshot of the executive dashboard that acts as a single pane of glass to display key metrics. According to the *case study*, it cut down aggregating workloads to just 10% of the original effort.



84.51 Data Observability dashboard powered by Unravel

Unravel Data is a data management platform that helps companies optimize their big data applications. Helped by Unravel Data, Kroger could identify performance bottlenecks in their big data applications and reduce infrastructure costs by 84%. Additionally, Kroger improved application efficiency by 51%, increasing customer satisfaction and revenue.

The second case study examines the DataFinOps features in the data observatory platform from *Acceldata*. This platform creates a spend analysis chart that displays cost consumption across services, accounts and workloads. Tags are used to categorize the data so that departmental chargebacks can be accurately calculated. The product provides a single pane of glass to see consolidated costs across all cloud and on-premises resources at individual and aggregate levels and to facilitate financial reporting to stakeholders.

Other capabilities of Acceldata are budget and recommendations. The platform provides tools to create and track budgets and generate data usage reports. Through inferred usage patterns, it also provides cost-saving recommendations.

Introducing DataBizOps

As seen in the first diagram in this document, the purpose of DataBizOps (and DataFinOps) is to demonstrate value from our data assets. It is a set of metrics that help in calculating productivity and reducing cost. For example, an organization may have built hundreds of data artifacts - reports, dashboard, views etc. However, by analyzing their usage, businesses can retire the whole set of processes leading to unused artifacts and saving costs.

The newest use case of data observability, DataBizOps, is often related to the rise in creating data products. "Data as a product" was introduced as one of the four principles of data mesh, while *Data products* turns the principle into business-outcome-driven consumable entities like a report, a dashboard, a table, a view, an ML model or a metric. DataBizOps gathers "data telemetry", like the frequency of data product releases, usage of data products and anomalies, which can lead to other indicators, such as poor data quality.

Data products take a business-first approach instead of the prevailing technology-centric bent of producing data artifacts. This approach can help mitigate challenges and frustrations of the modern data stack. For example, delivering outcomes involve a tedious and complex pipeline collection that leads to increased cost, effort and time. These hard-to-debug processes cause reliability and downtime issues, requiring data observability. Another example is the poor adoption of data catalogs, as we start by trying to collect and tag all the data available. This 'boil the ocean' approach often fails and reduces trust in the data governance initiative. If we catalog the data for the data products being built, we will have a better chance of a successful data governance outcome.

So, what is the role of DataBizOps?

DataBizOps can help establish data's ROI through metrics, like the number of data-related goals and objectives met and the level of data consumer and management buy-in. It can help support the data strategy within the organization. The best part is that the business strategy drives the data strategy. They can be in perfect alignment, maximizing the potential of corporate data assets.

An organization can benefit from DataBizOps:

- Launch a data marketplace and collect metrics to ensure that the data is being used for the intended purpose, by the intended consumers, and within the established guardrails. DataBizOps can augment the data producers' data governance efforts by extending them to data shares and exchanges.

- Strengthen the DataOps processes by providing the telemetry so intelligent decisions can be made for automation, testing and orchestration. Today, many orchestrators are rule-based and act like a complex collection of CASE statements. However, DataBizOps can feed contextual information, so the orchestrators can assess the next steps.

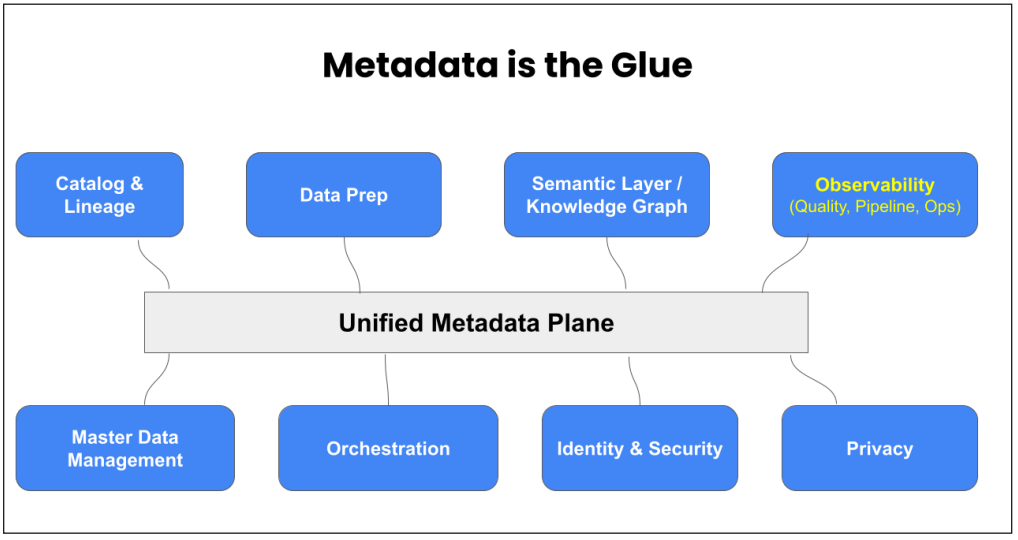
This is an overview of the potential of DataBizOps capabilities of data observability. Although some vendors are developing standalone products to provide the above-mentioned features, it should be part of the comprehensive data observability product.

Summary

DataFinOps (Financial Operations) refers to the set of practices, processes, and technologies used to assess and manage costs of the organizations' data assets. It aims to provide a streamlined and efficient approach to managing the data lifecycle, from collection and storage to analysis and decision-making. As the case studies reflect, it helps reduce costs and improve efficiency.

DataBizOps is the newest member of the data observability space. It has the potential to help establish data's ROI finally.

As we end the chapter, various use cases of metadata should be unified into a common metadata plane. These use cases are depicted in the figure below.



Metadata is the key to the success of data and analytics projects. This book has examined the observability capabilities. Organizations will benefit from a unified approach to handling metadata.

DataFinOps and DataBizOps metrics can help integrate the overall metadata management initiatives so our assets have high quality, reliability and security, and developers are more productive. A comprehensive metrics-driven metadata management plane can reform a data engineer's role from being reactive to becoming proactive. In addition, the data teams integrate into the rest of the business and deliver on the strategic imperatives. This will help with mainstream data observability adoption in an environment where data, data, and analytics workloads are constantly exploding.

Message from the Curator and Publisher

Thanks for reading the book. Please feel free to reach out to us at *Ravit@theravitshow.com* if you find an error, grammatical issues, etc.

Also, feel free to join The Ravit Show to stay updated with all the relevant news -- *www.theravitshow.com*

Thank you again for reading.

Best,
Ravit Jain
Founder & Host of The Ravit Show