MANTA

The Ultimate Guide to Data Lineage in 2022

The Ultimate Guide to Data Lineage

Given the high volume of data your organization collects, it should be easy to take informed action and make decisions. But today's data systems are so complex and chaotic that even asking a simple question is complicated unless you have the right tools to help you paint a complete, comprehensive picture of your data landscape, tame the data chaos, and turn it all into valuable insights.

Key Takeaways

1. CTOs and CIOs of data-driven companies are facing one major challenge: the skyrocketing complexity of their data stack (data pipelines). Combined with a shortage of engineering talent, it limits their ability to cope with the fast pace of changes, negatively impacts innovation initiatives, increases the risk of data incidents, and causes reputation issues and non-compliance with regulatory requirements.

2. Successful organizations are adopting augmented data management to deal with the complexity, maintain the ability to innovate and iterate quickly, and stay efficient, thanks to higher levels of automation across all data management and data processing practices. Active metadata and data lineage are key cornerstones of any such effort.

3. Data lineage started as a simple way to describe the data journey, but now it has evolved and become the main tool for organizations to map, understand, and gain insights into their data pipelines.

4. There are multiple very different views of data lineage and several linked approaches to its discovery, each with its advantages and disadvantages.

5. Traditional static metadata is only one piece of the puzzle here. Dynamic metadata can and should be used not only to control data behavior, but, primarily , to better understand and manage the heterogeneous and increasingly complex data pipelines in today's enterprises.

Table of Contents

1. Why Are We Talking about Data Lineage in 2022?

Data management has undergone a massive transformation in the past decade. From the early days of business intelligence, using historical data to get business insights, we have moved to the big data era, collecting data without thinking about why and training modern AI/ML algorithms to predict the future of our businesses. And then, we started asking questions about privacy, trying to establish boundaries for what is ethical and what is not. And all that time, our data infrastructure was growing in complexity—from batch-processed structural data, we evolved to a crazy ecosystem with thousands of components aimed at one goal: to derive more value from data in an ethical way. Such an ecosystem is too much for a human brain to handle, too diverse, too interconnected.

And there are consequences:

Slower delivery of new analytical/predictive insights due to limited understanding and control of the environment. Statistics from our customers show that up to 40% of data engineering resources are spent on unproductive impact analysis, just assessing the impact of new development requirements. It distracts developers and slows down the delivery of new features by almost 100%.

Decreasing level of trust in reports, dashboards, and insights as we cannot fully explain how the numbers we present were calculated and what their origins and associated data quality or data privacy attributes are. It also leads to frustration both on the business side (depending on data they don't trust, waiting a long time to get even basic questions answered) and the technology side (time wasted searching for the same answers about the origin of data again and again).

Growing number of data incidents due to our limited ability to assess the end-to-end impact of to-be implemented changes. And with our dependency on data, one single incident can cause incalculable damage to the business. We may have modern data quality / data observability tools, but that is still a very reactive approach. (We find bugs, we don't prevent them.) According to research, it is 20 times more expensive to fix a bug in production than to fix the same bug in implementation, and it is 100 times more expensive than fixing it in the design phase. (Check out this amazing paper from the IBM System Science Institute.) And that is something!

Severe shortage of data engineering talent caused by two major factors: (i) data engineers have become, due to the growing complexity of the data stack, more critical than any other role as they enable the data pipelines and integrated data structures, and (ii) due to the variety of data technologies and strategies, the data engineering role has evolved and now requires a larger skill set, which makes good data engineers harder to find. Demand for data engineers goes up by 50% every year, and their salaries are skyrocketing. Yet, we waste their valuable time on manual, routine tasks (like manual impact analysis or incident investigation), increasing their frustration and the risk of them leaving.

Increasing risk of non-compliance and regulatory penalties with the growing number of regulations, policies, and societal expectations, especially related to the ethical use of data and data privacy (GDPR, CCPA/CPRA, HIPAA, etc.) and data quality (BCBS, SOX, etc.). There have been many cases where one incident caused damage to a company totaling hundreds of millions (fees, reputation damage, etc.).

There are endless business opportunities if we manage to tap into the full potential of data. To do that, we first need to get our data systems under control and find a way to stay efficient despite skyrocketing complexity. We at MANTA believe that **metadata is the key**.

2. Not All Metadata Is the Same

Metadata management solutions have been helping us manage data about data for quite some time now. And while some are bold enough to claim that the first metadata management solutions were the scroll tags in the Great Library of Alexandria, today both people and companies deal with much more data chaos than any of those ancient librarians in 280 BC.

But the Great Library of Alexandria had one advantage—they only needed very **static metadata** about the books they were managing. But that is not true for data systems today. They consist of two equally important parts: the data itself and the "system" component responsible for data storage, processing, transformations, etc. We have done a pretty good job so far in cataloging and profiling data, but we keep failing to understand the **dynamic aspect** of it.

- How does a data element move from one location to another?
- How do two (or more) data elements depend on each other?
- How does the profile of a data element evolve along its journey through the environment?

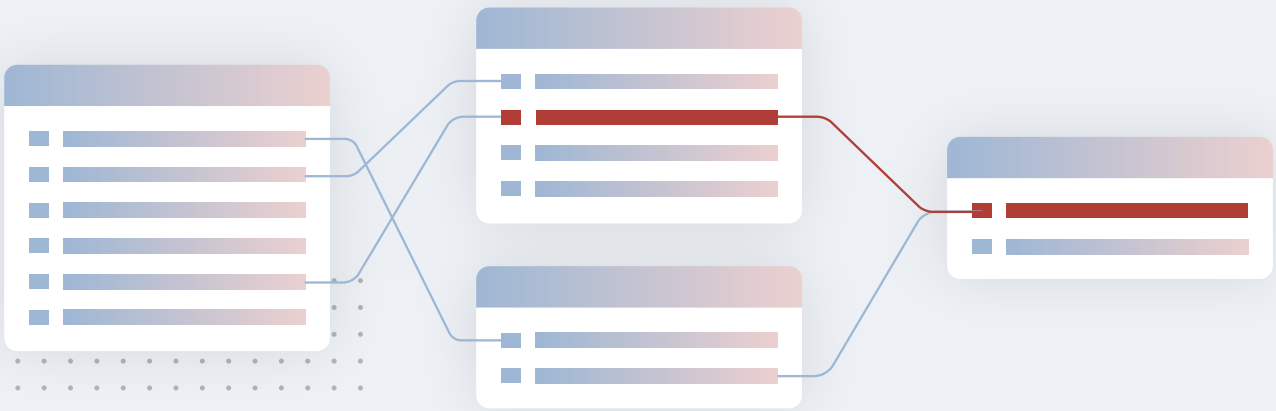
These are just a few of the questions we should be looking for answers to.

And the answer is Data Lineage.

3. What Is Data Lineage?

Traditionally, data lineage has been seen as a way of understanding the data journey through all your data processing systems:

what sources the data comes from, where it is flowing to in the environment, and—last but not least—what happens to it along the way.

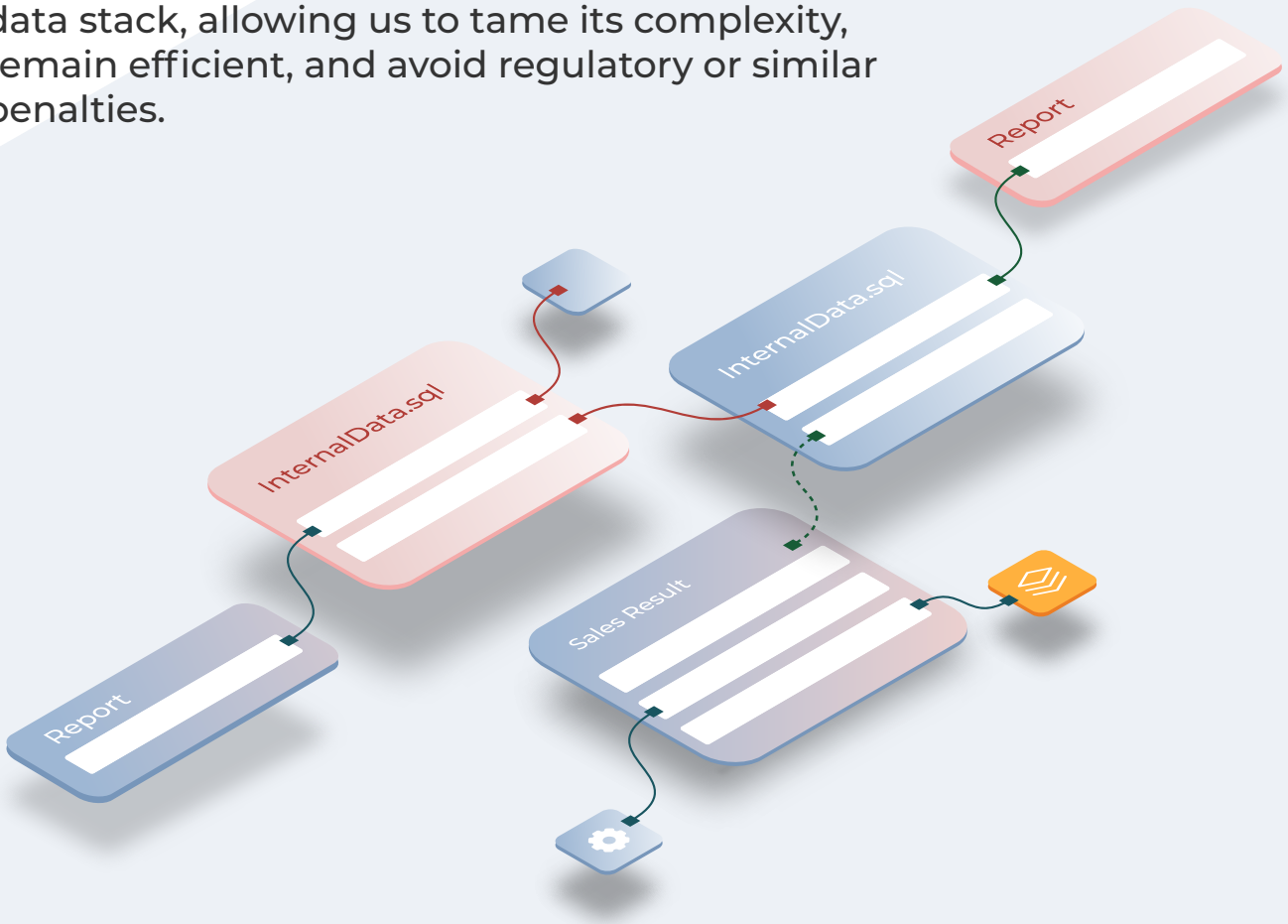


But real data lineage is far more than that. Data lineage represents a **detailed map of all direct and indirect dependencies between data** entities in the environment. Why is that so important? Here are a few examples that go beyond the traditional definition of data lineage.

A detailed dependency map will tell you:

- **How changing a bonus calculation algorithm in the sales data mart will affect your weekly financial forecast report (and if you are going to like it)**
- **What the best subset of test cases is that will cover the majority of data flow scenarios for your newly released pricing database app**
- **How to divide a data system into smaller chunks that can be migrated to the cloud independently without breaking other parts of the system**

Such a map is the core component of a modern data stack, allowing us to tame its complexity, remain efficient, and avoid regulatory or similar penalties.



4. Activating Metadata Is Key

But no matter how powerful data lineage (or metadata in general) is, if it just sits somewhere, no organization will ever be able to maximize its potential. That is also one reason why historically metadata was not very successful. We spent decades focused on metadata collection and forgot about the important part—making it useful and worth the investment. This is changing now with the focus on **active metadata**.

According to Gartner:

> **“Through 2024, active metadata capabilities will expand to include monitoring, evaluating, recommending design changes and orchestrating processes in third-party data management solutions.”**

> **“Through 2024, organizations that adopt aggressive metadata analysis across their complete data management environment will decrease time to delivery of new data assets to users by as much as 70%.”**

To better frame it, think about our experience today when driving a car compared to how it was in the past. Nowadays, thanks to augmented reality in our cars, we get a heads-up when the windshield washer fluid should be refilled or when the oil should be changed, and so on. Breaks even activate themselves if there is an object in front of the car! All this is possible because of a "metadata processor" in your car that scans the car, its parts, and then the environment and uses advanced intelligence to help you with various routine yet critical tasks.

Sounds promising, but what does it mean in the context of data management? Here are a few examples relevant to data lineage.

- **Continuous detection of "dead tables" where potentially sensitive information is stored but not accessed or used** (like when we create a helper table to process large amounts of data and we forget to drop it or when we decommission part of a system without actually knowing what all its components are)
- **Instant alerts if anything is changed in the environment that has a negative impact on tactical management reports or key data features used by the data science team** (like if you change a gender column in a database from male/female to male/female/other without realizing that the data from that column is used as an important filtering condition down the road in a Python credit scoring algorithm)
- **Proactive notifications to inform us of overly complex parts of our data pipelines where refactoring or redesign would help to reduce the risk of failure**
- **Warnings if we design a data pipeline moving data between locations where no data should ever be moved** (like if we accidentally build a data flow between production and test data lakes or between two countries with incompatible data protection policies)

The main idea here is to combine existing metadata with the automation, intelligence, and processing power that only computers can offer; switch from reactive mode to proactive; and do all we can to help people better manage and use data, even in overly complex data environments.

5. Business Benefits of Data Lineage

There are several key benefits of having data lineage and associated usage patterns that help organizations achieve their business goals.

Zero Incidents with Impact Analysis

Organizations with better incident prevention strategies achieve higher productivity and significant cost reductions. One key technique of the most successful companies is the extensive use of impact analysis for all planned changes early in the process in the design phase. Our customers report a significant decrease in the number of erroneous releases (below 1%) and improved productivity. (Recovering from an incident in production is about 100 times more expensive than fixing it in the design phase.)

There is also a strategic aspect. Properly done impact analysis allows businesses to look ahead and determine how changes will impact the organization. When done incorrectly, changes can result in delayed and low-quality deliveries, the need for emergency fixes, and extra work. Nevertheless, most organizations struggle with impact analysis as it requires significant resources when done manually. Thanks to its automated capabilities, MANTA enables data engineering teams to increase their productivity by 30 - 40%.

Data Pipeline Observability and Faster Incident Resolution

Data lineage expands the scope of traditional data quality and observability from data only to data processing infrastructure (data pipelines). Most data incidents don't originate from source data of questionable quality (even if that is a very common issue) but from data pipeline problems (like API calls not matching database column type due to recent changes in the system).

Thanks to data lineage, these incidents can be prevented in the design phase (see the previous section) or identified in the implementation and testing phase to achieve higher productivity and reduce maintenance costs. (Fixing an issue in production is about 20 times more expensive and labor-intensive than fixing it in the implementation phase and 100 times more expensive than in the design phase.) Complete lineage also allows organizations to easily trace any data-related issues back to the source 90% faster compared to the traditional manual approach, so the teams responsible for particular systems can fix any issue in a matter of minutes.

Regulatory Compliance

The number of regulations that require data lineage has increased rapidly over the past few years, and we can suppose that there are more coming, including BASEL, HIPAA, GDPR, CCPA/CPRA, and CCAR, just to name a few. All of these have one thing in common—the company's stakeholders (customers, auditors, employees, control authorities) require accurate tracking of the data we report.

- Where does it come from?
- How did it get there?
- Are we capable of proving it with up-to-date evidence whenever necessary?
- Do we need weeks or months to complete a report, which ultimately is not even entirely reliable?

Faster and More Efficient (Cloud) Migrations

New cloud data platforms (Snowflake, Amazon Redshift, Google Cloud, Microsoft Azure, IBM Cloud Paks, etc.) have become enablers of digital transformation. Organizations move their operations to the cloud to benefit from performance and scalability and to stay competitive. However, anyone who has ever witnessed the migration of a data system knows how complex a process it is. One of the key challenges is to scope the project. Every system consists of thousands or millions of interconnected parts, and it is impossible to migrate it in a single step.

A successful strategy is to divide the system into smaller chunks of objects (reports, tables, workflows, etc.), which poses other challenges—how to migrate one part without breaking another, and how do we even know what pieces can be grouped together to minimize the number of external dependencies? A migration project also represents an opportunity to consolidate or entirely remove components that are no longer in use to avoid migrating worthless data. But even that is tricky, as it requires the ability to view all dependencies within the environment prior to migration. Successful organizations use data lineage to complete their migration projects 40% faster with 30% fewer resources.

Data Consolidation and Virtualization

Data continues to grow and increase in complexity. Many enterprises are consolidating their data from multiple sources in one place or exploring data virtualization technologies that make it appear that the data is in one place. Whether it is being called a data lake, a central repository, or another term is less important to this discussion than being able to identify the original sources of the data and how it arrived at its current location, or where it is actually located if data virtualization or replication has been implemented.

Addressing the Data Engineering Talent Crisis with Self-Service

The shortage of data engineering talent has been a big thing the past couple of years, turning from a problem into a crisis as the demand for data engineers explodes due to the increasing complexity of our data stack. The very last thing we want is to waste the time of a data engineer worth $200k to $500k per year on routine, manual, frustrating tasks like chasing data incidents, assessing the impacts of planned changes, or patiently answering the same questions about the origins of data records again and again. Quite similarly, we also face a lack of data scientists.

To protect these valuable assets, successful organizations use data lineage to automate routine tasks and enable self-service wherever possible. Armed with the right solution, data scientists and other data users have the power to retrieve up-to-date information about all the details surrounding lineage and data origin on their own whenever they need it. A detailed data lineage map also enables faster on-boarding of new data engineers and allows organizations to hire less experienced people for the role without jeopardizing the stability and reliability of their data environment.

Trust in Data and Understanding It

If data lineage is ignored or mapped inaccurately, your decision-makers will lose faith in their reports and analytic models. Report developers, data scientists, and data citizens, as they are often called, deserve data that inspires accurate, timely, and confident decision making. Only when you have a complete understanding of your data can you really rely on it and make the most of it, increasing your overall efficiency.

6. Let's Get Technical:

How to Create Data Lineage and Keep It Up to Date

Now that we know what data lineage is, its importance to the company, and its benefits, it is time to talk about how data lineage can actually be delivered.

When we talk about metadata, we very often think of simple things such as tables, columns, and reports. But lineage metadata is more about logic—instructions or code in any form. It can be an SQL script, a database stored procedure, a job in a transformation tool, a Java API call, or a complex macro in an Excel spreadsheet. Data lineage, specifically, can be anything that moves your data from one place to another, transforms it, or modifies it. So, what are your options for outlining, diagraming, and understanding that logic? There are two dimensions to be considered when answering these questions: **which source we get information from to build the data lineage map** and **how we build it**.

Question 1: Source

Data lineage information can be derived from three major sources: data, logs, and the code.

Data as a Source aka Pattern-Based Lineage

This technique expands current data profiling algorithms. It reads metadata about tables and columns and uses information about data profiles to create links representing possible data flows based on common patterns or similarities. Tables or columns with similar names and columns with very similar data values are examples of such similarities. And, if you find a lot of them between two columns, you link them together in the data lineage diagram.

There is one big advantage to this approach—if you only watch data, not algorithms, you do not have to worry about technologies and it is no big deal if a site uses Teradata, Oracle, or MongoDB with Java on top. But this approach is usually not accurate. The impact on performance can be significant (they work with data), and data privacy is at risk (they work with data). There are also a lot of details missing like:

- So-called "transformation logic"—"how and where is data actually being transformed and modified?"
- The lineage is typically limited to the database world, ignoring the application part of your environment.

On the other hand, this approach may be sufficient for some cases, especially when it is impossible to read the logic hidden in your programming code because the code is unavailable or proprietary and cannot be accessed. Pattern-based lineage is also the only approach with slim chances to identify manual data flows happening outside of any system (like copying data to a flash drive, modifying it on another computer, and storing it on a different part of the system).

Logs as a Source aka Run-Time Lineage

This technique relies on run-time information extracted for the data environment. It can be log files, execution workflows exported by ETL/ELT tools, or any other source with sufficient run-time details. Some data processing engines use a trick called data tagging, where each piece of data being moved or transformed is tagged/labeled by a transformation engine which then tracks that label all the way from start to finish. Unlike pattern-based lineage, run-time lineage must take into account different technologies in the data stack as the format and structure of the logging information vary significantly. Regular expressions, rules, or AI/ML can be deployed to identify relevant parts of log files and derive data flow information.

The operational nature of run-time lineage is also valuable for incident resolution as it gives us very accurate information about the flow of a specific data element that has been identified

as erroneous. But this strength also represents its biggest weakness. Run-time lineage only captures information about recently executed data flows. Imagine an algorithm for product pricing—usually a very complex calculation with many variables and tens or even hundreds of different scenarios. Each scenario represents one possible flow of data. But these scenarios are not executed equally often as we have more and less frequent input conditions. One may argue that if we wait long enough, even less frequent scenarios will run, but only if the system is never changed. In a traditional environment, around 20% to 40% of it is changed in just one year. That leads to inconsistent and inaccurate data lineage as some parts are either missing or are no longer valid because the underlying logic was changed even if the scenario was not yet executed and, therefore, its lineage not yet captured.

Another limitation of run-time lineage is the absence of transformation details. Not everything is or can be logged, especially in the case of more complex algorithms and calculations. The same is true of processing done outside the database/ETL/ELT world (like in Java, Python, etc.). As a result, run-time lineage can often capture only very high level and generic table to table mappings or not even that.

Blindly using such metadata poses a big risk for an organization.

- If used by a data engineer to run impact analysis, it leads to a high probability of incidents when designing and implementing changes in the system and new requirements.
- Similarly, if used by a risk analyst to prepare a regulatory report, it leads to inaccuracies in the report and increased risk of (public) incidents and penalties.
- If used by a data scientist to analyze and prepare data to train a new model, it leads to inherent inequality encoded into the AI/ML algorithm developed.

Code as a Source aka Design Lineage

Instead of using log files to identify data flows, we can look directly into the code that processes and transforms data records. The word "code" is meant in the broadest possible way here. Code can be an SQL script, a PL/SQL stored procedure, an ETL/ELT workflow encoded in a proprietary XML format, a macro in an Excel spreadsheet, a mapping between a field in a report and a database column or table, a Java API, a Kafka stream definition, an XSLT transformation, or a Python algorithm in a Jupyter notebook. This variety poses a major challenge because parsing and reverse engineering the code is much tougher than parsing log files, and it requires specialized scanners for all supported technologies.

However, this approach offers several critical, hard-to-beat advantages, which is why it is the main strategy used by the most successful vendors and organizations.

- It is a very accurate approach with no or only a few false positives (if data lineage is incorrectly identified between two columns), which is critical for incident management as it narrows down the scope of the investigation and makes change management and impact analysis more efficient.
- It is the only approach that can reliably detect indirect data flows where one data element influences another one even without a

direct data lineage connection (for example, a database column used as a filtering condition for data presented in a report). That is essential for change management, impact analysis, incident management, migration projects, and regulatory reporting.

- It has an extremely low, close to zero chance of missing any, even rarely used (or not used at all), data flows. That is critical for change management processes and impact analysis as well as for migration projects, privacy programs, and regulatory reporting.
- It is the only approach that records details about transformations and calculations used to process data, which is especially important for compliance and regulatory reporting. It also allows vendors utilizing this approach to deduce the semantics of analyzed transformations.

Question 2: Process

Today, we see three major approaches on the market.

Manual Data Lineage Analysis

Manually resolving lineage usually starts at the top by mapping and documenting the knowledge in people's heads. Interviews with application owners, data stewards, and data integration specialists will give you a fair amount of information about the movement of data in your organization. From there, the lineage can be defined, usually in spreadsheets or other straightforward mapping mechanisms, to reflect the subject matter the experts have described.

Of course, one downside to this approach is that the information may (and probably will) be contradictory, or if you miss talking to someone you simply don't know about, a piece of the flow might be missing! This often results in a dangerous situation where you have lineage but are unable to use it for real case scenarios. The resulting lineage cannot be trusted.

Manual data lineage analysis is inherently done using code as a source (see Code as a Source above), where the code is analyzed by its authors or external resources. Regardless of how it is done, manually examining the code, comparing column names, and reviewing tables and file extracts by hand is tedious! It may not even be worth attempting unless you have team members with the requisite skills and expertise in the programs and modules you need to examine. Due to code volumes, complexity, and the rate of change, this method quickly becomes unsustainable. Sooner or later, such manually managed lineage will fall out of sync with the actual data transfers in the

environment, and once again, you will have lineage you cannot actually trust.

Despite these concerns, this approach cannot be sidelined completely, as this is where we all need to start to be able to gain insight into what is actually going on across the entire environment. Sometimes there isn't any code at all or any permissions to access and profile the data directly (especially with legacy systems) and domain experts are your only source of lineage.

Self-Contained Data Lineage Analysis

This approach, very popular among ETL/ELT vendors, is fully automated. Its core concept is that a tool that fully controls every data movement, every change in data, and the whole data processing workflow has full insight into it (unlimited access to internal logs, details about executed workflows and processing instructions, etc.). Vendors and tools adopting this approach to data lineage analysis typically use logs as a source (see Logs as a Source above) with all its advantages and disadvantages.

The key disadvantage specific to self-contained analysis is the fact that data lineage is limited to the controlling platform (usually an ETL/ELT tool). Anything that happens outside the controlled environment is invisible, like it does not exist. Sometimes, even more complex components of the platform pose a challenge, like a component with a custom Python/Java/SQL mapping. Organizations trying to use this approach sometimes enforce a single data processing platform or prohibit the use of its more complex components, just to quickly find out how limiting it is for the vast majority of data engineering tasks and how much it slows down new development (not even considering the frustration of data engineers).

External Automated Data Lineage Analysis

As the name indicates, this approach also offers fully automated data lineage analysis. Unlike the previous option, external analysis is inherently designed to cope with the heterogeneity of the data stack and does not require all data processing to happen in one tool or platform. This is why it is the leading option among all vendors offering above-average data lineage capabilities.

Most vendors that adopt external analysis use either logs (see Logs as a Source above) or the code (see Code as a Source above) as a source for data lineage discovery. But some vendors primarily use data (see Data as a Source above) or try to combine several approaches.

7. Data Lineage Tools and Solutions on the Market

For historical reasons, data lineage is part of the very broad metadata management market. And that is a very crowded space. Why? Simply because there are too many types of metadata, too many ways to use them, and too many different personas/roles in the organization to help. (We recommend reading the short, funny Time To Kill Metadata.)

The traditional type is **static metadata**—information about tables, fields, columns, business terms, and their data types, location, quality attributes, tags they have, etc. This type of metadata is very useful for data search and discovery tasks. Typical capabilities are **data search, discovery, profiling, classification, tagging,** and **labeling**. Static metadata helps data professionals to find and better understand the nature of the data they work with.

Static metadata is the primary domain of metadata managers (legacy name) and data catalogs (modern rebranding). There are many "native" data catalogs, but we also see data quality, data security, ETL/ELT, data preparation, data wrangling, and data virtualization vendors entering the space of data cataloging as well.

The other type of metadata, overlooked for many years due to its complexity, is **dynamic metadata**—information about the journey of the data from its source to its target, including all changes, transformations, and calculations. Dynamic metadata is very useful for controlling the behavior of data and, even more so, for better understanding and managing the heterogeneous and increasingly complex **data pipelines** in today's enterprises. There are several "native" data lineage players, but we also see some data catalogs, data preparation, data virtualization, and ETL/ELT vendors entering the space.

Some traditional vendors have been offering data lineage solutions for more than a decade. But as of yet, none of them have managed to fully tap into the true potential of activated data lineage to get rid of manual processes, increase the productivity of an organization, and enable us to regain trust in the data, reports, and insights we use. To achieve that goal, the following key data lineage ingredients described in subsequent sections must be present: (1) **accurate and detailed metadata**; (2) **semantics and AI**; and (3) **activating integrations**.

Accurate and Detailed Metadata

Successful organizations understand what value is hidden in their data, and it is no surprise that their main focus is on understanding data. That leads to collecting structural metadata and capturing how data is organized and sorted. By doing that, organizations overlook dynamic aspects of the data and its dependencies, which are best represented by data lineage. That said, without understanding and controlling data lineage, data management will remain in the industrial age.

But dependencies are everywhere and are usually well hidden. Every single transformation, simple calculation, and data movement represents a type of dependency. There are even indirect dependencies like filtering conditions. **Automated discovery** is an absolute must. Another challenge is that dependencies must be mapped very accurately, in detail; otherwise, the resulting map will contain too many false positives and/or will miss several critical relations among the data. Without **detail and accuracy**, any attempt to control dependencies is destined to fail.

Semantics and AI

A very common challenge when automating manual tasks is how to train a computer to respond to various, sometimes unexpected scenarios. The traditional approach that has been used for decades, called imperative programming, is basically a set of rules a computer can follow. “If A happens, do B; else do C”. Due to the complexity of data systems and the large volume of (hidden) dependencies, that approach has its clear limits on our way to automation. With the expansion and wider adoption of AI, a different approach is blooming called declarative programming. Without diving deeper into the subtypes of programming paradigms and the differences between imperative and declarative, we will focus on what is needed to fully deploy AI and other

advanced techniques to better support automation in the data management context—**Semantics**.

Just mapping dependencies is not enough. The core information about the flow of data and the data journey has to be enriched by its “meaning”. What is really happening to the data, or what does a specific transformation mean and how does it affect the data? The ability to answer such questions gives us more power and control over dependencies and allows us to deploy more advanced techniques for automation. This layer manifests through various capabilities—the ability to differentiate between

various types of dependencies (direct, indirect), the ability to understand the evolution of data lineage over a period of time (time slicing, revisions), or the ability to translate the real data processing code into more high-level, user-friendly expressions.

Activating Integrations

Although, historically, metadata catalogs focused on passively storing metadata, the key task is actually how to integrate it into all data management processes and how to pro-actively use this extra layer of knowledge to speed up everything and reduce the amount of manual labor.

Strategies for activating data lineage metadata differ based on the domain where we are integrating it. For every domain, we ask the same set of questions.

- What processes and tools are currently in use?
- What is still being done manually? And why hasn't it been automated yet?
- How can accurate, detailed, semantically rich metadata help with automation?
- Is there anything that would have a major impact that we are not doing today but we could do if it were automated?
- Is there a way to use automation to redesign and improve an existing process?

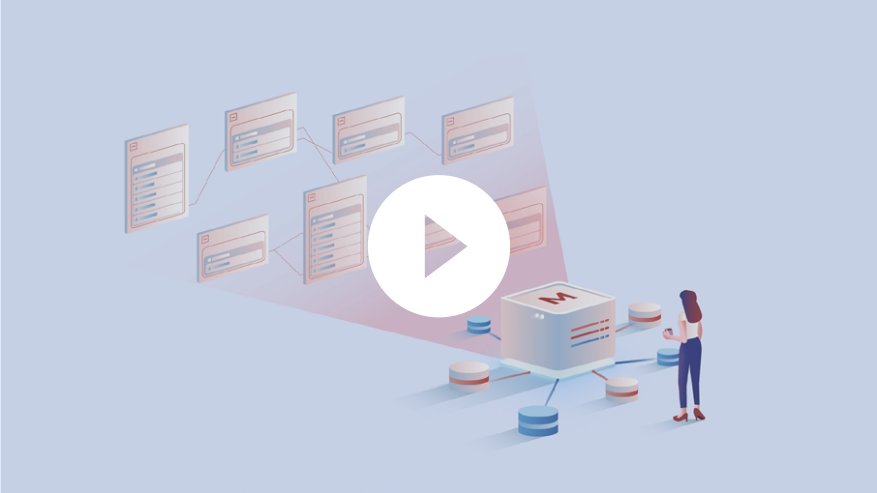
Data lineage has tremendous potential. It can be activated and used to deliver direct benefits to data professionals in any organization (see the examples dealing with data lineage in the section Activating Metadata Is Key).

Data lineage also enables the activation of other forms of metadata. In other words, if, for example, a data catalog, data privacy, or ETL/ELT tool has access to detailed, accurate, semantically rich data lineage, it opens new doors as to what can be activated when and how. That is why successful organizations deploy enterprise-wide data lineage platforms and integrate them with other parts of their data infrastructure.

Back to summary

How MANTA Can Help with Data Lineage

MANTA



MANTA creates complete end-to-end data lineage to provide organizations with full understanding, observability, and control of their data pipelines; give them a clear line of vision to avoid any blind spots; and work as a cornerstone in their active metadata initiatives.

BOOK A DEMO

getmanta.com