DATA / STORAGE

The Unfortunate Reality about Data Pipelines

How ETL, ELT and Reverse-ETL tools get in the way of building better data products.

Nov 3rd, 2022 7:45am by Jon Fancey

Image via Pixabay.

Confluent sponsored this post.

The data mesh article by Zhamak Dehghani has made popular the concept of "data as a product." As one of the four essential principles, it describes a fundamental shift in the way organizations need to create, store and communicate important business data. While the data mesh concept is relatively new and feels simple and intuitive, the problems it highlights and proposes to solve are not.

so-called modern approaches provide a realistic solution to the problem. If anything, they amplify existing data access challenges.

Under the hood, data pipelines play a critical role. They do a lot of the heavy lifting, coordinating data movement, extracting, transforming, integrating and loading data across silos of systems to serve various operational and analytical use cases. They're essential to building trustworthy data products. And yet, despite the criticality of data pipelines without which a modern data-driven organization cannot function, they fundamentally haven't evolved in the last few decades.

In this article, we'll dive into the legacy approaches to data pipelines and the compounding data problems present in the much-touted modern data stack. We'll also explore how to reimagine your data pipelines and build better data products that can serve the real-time needs of your business and customers.

ETL (extract, transform and load) tools were originally built decades ago to extract data from siloed systems, and then transform and load it in periodic intervals into a format that matches the destination data warehouse for post-hoc analysis. Data typically has one final destination, the data warehouse, and processing is heavily governed by centralized, domain-agnostic "data product" teams, who spend a bulk of their time fixing broken data pipelines and use their remaining time to discover and understand domain data to glean meaningful insights. This approach made sense in the old world because the application of data, the analysis, was a back-office concern and had cycle times of weeks, months or even quarters. That analysis, in a highly latent fashion, was then used to steer the company.

In the 2000s, due to the advent of the internet, data growth accelerated and the market saw an emergence of cloud-based data warehouses such as Amazon Redshift, Snowflake and Google BigQuery that could accommodate any volume of data, and scale storage and processing infinitely. Unfortunately, traditional ETL software isn't able to take advantage of the native improvements the newer generation of cloud data warehouses offer. In an attempt to overcome the performance bottlenecks created by legacy ETL tools, a variation of this traditional paradigm — extract, load and transform (ELT) — has emerged.

ELT and Reverse-ETL Pipelines Reinforce Old Bad Habits

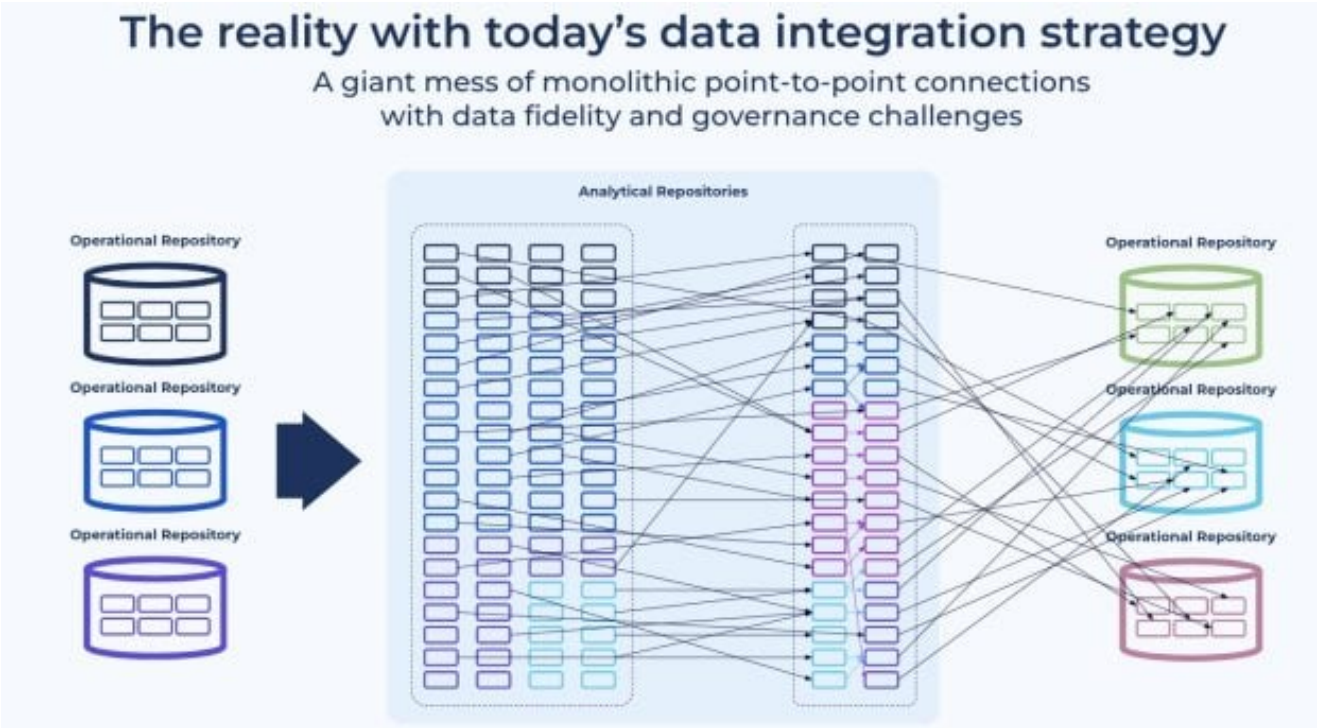
ELT (extract, load and transform) tools also focus on loading the data in a centralized cloud data warehouse or data lake, but unlike traditional ETL tools, the transformations happen in the target system, resulting in reduced physical infrastructure and intermediate staging layers. This allowed for data to be extracted and directly loaded into the data warehouse, improving load times. The final destination, yet again, being the data warehouse.

access to the data. And that's given rise to a whole new set of tools that reverse the pattern of data movement. A symptom of the centralizing force, reverse-ETL (rETL) tools have evolved to share the subsequent analysis of the data (from the data warehouse) back to operational systems, such as databases and SaaS applications such as CRM, finance and ERP systems.

While the combination of ELT and rETL tools surface the need to share data back to the operational systems and SaaS applications to power various use cases, these approaches intensify the data problem. Decodable CEO Eric Sammer wrote an excellent article on the abuse of the data warehouse and how "putting high-priced analytical database systems in the hot path introduces pants-on-head anti-patterns to supportability and ops."

Data Problems Worsen with the Modern Data Stack

ETL, ELT and reverse ETL are all approaches that aim to solve the need for unlocking the value of data. They all highlight the imperative to maximize the availability, reliability and usability of data to derive meaningful insights and power various use cases. However, all these approaches make a foundational assumption that data needs to be



If you take a moment to think about this waterfall approach, which involves centralizing data into the warehouse and then reversing that pattern to decentralize data access, it has significant drawbacks that are nontrivial.

- **Batch based:** The entire sequence of steps is built around batch data extraction, batch data processing and batch data delivery, resulting in low fidelity snapshots, frustrating inconsistencies and stale information. Any subsequent use of that data is based on outdated information.

- **Centralized data and data teams:** These approaches have really created two types of centralization problems that need to be addressed.

    - First, in the classic ETL/ELT paradigm, the data warehouse is the final destination. But like we discussed earlier, data no longer goes to just one place. Almost every system in the company needs to have access to data that is often maintained elsewhere in the organization. When you connect all your systems and applications from a centralized warehouse with ad-hoc pipelines to enable the flow of data, you end up with a giant spaghetti mess of interconnections that's complex, unreliable, risky, costly and untenable. Confluent co-founder and CEO Jay Kreps has discussed the challenges with spaghetti mess architecture at length.

developers who need access to high quality data to launch new products faster. With no clear lineage, no business-meaningful goals motivating the ingestion of data, and ownership and responsibility diluted, the promise of self-service access to data is yet to be realized, hindering the pace of innovation.

- **Immature governance and observability:** Organizations must maintain a delicate balance between centralized standards for data observability, security and compliance, while also delivering self-service data access to developers. When data flows are enabled through a patchwork of point-to-point connections, it's challenging and often impossible to scale tasks, like tracking data lineage, observability and security of multiple components of these pipelines, and have active enforcement of governance rules.

- **Infrastructure-heavy data processing:** With today's tools, data processing is tightly coupled to the underlying infrastructure. As data volumes grow, faster data processing requires more compute and storage, and inconsistent workload variations make it difficult to predict resource requirements, resulting in scale and performance challenges and high operating expenses.

- **Monolithic and inflexible design:** Developers often perceive GUI-oriented data pipelines as a black box, as they don't have access to the underlying code behind the transformation logic. When business logic needs to change, evolving these pipelines to support new requirements while avoiding breakages is quite complex and often requires manual coordination. As a result, engineers are fearful of changing existing pipelines and instead prefer to add just one more to address each new workload, increasing pipeline sprawl and technical debt.

Despite all the buzz around the modern data stack where traditional data integration tools play an important role, the reality is that the entire stack is built on a legacy paradigm, revolving around the centralization of data into the data warehouse.

However, as Sammer mentions in his article, the data warehouse was never meant to be a gateway to enable free movement of data. "Its design center is to store data at scale, and to support things like large analytical queries and visualization tools. Gateways are built for many-to-many relationships, for decoupled no-knowledge apps, for decentralization and access control." We're trying to solve the right

"We've accidentally designed our customer experience to rely on slow batch ELT processes," states Sammer.

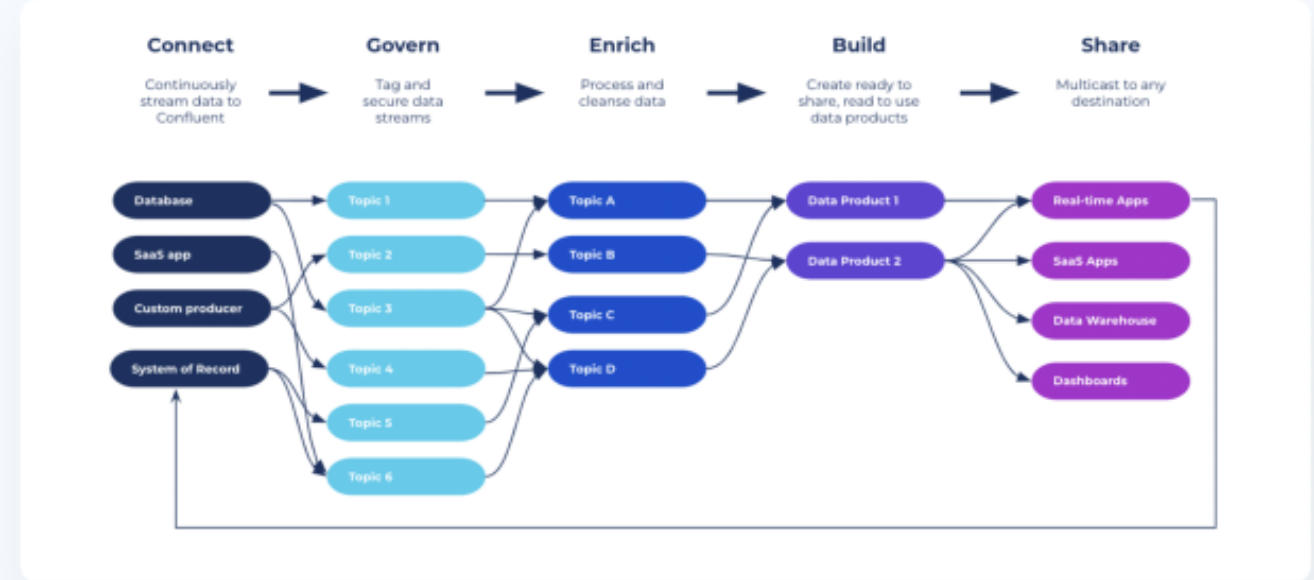
How to Implement a Product Mindset in Your Data

To put your data to work and make it discoverable, accessible, and usable, you have to stop thinking of the flow of data as a sequence of processing steps, where the next step is triggered after the previous step is completed. Instead, your data needs to be a first-class citizen.

You have to think of data as something that's continuously flowing, continuously being processed and continuously shared, so your systems and applications can react and respond to the data the moment it's created and the moment it changes.

This requires a shift in your mindset. This requires treating your data as if it were a high quality, ready-to-use product that's instantly accessible across the organization. It's consistent everywhere, which means that everyone is using the same data and taking advantage of the latest and greatest data so your operational systems can serve your customers better, your analytical systems can meet the demands of your stakeholders, and your SaaS applications are always up to date.

It's governed, which means you can track where the data is coming from, where it's going and who has access to it. You enable the discoverability of data assets through data contracts, so whoever needs access to the data in whatever format can easily subscribe and use it on demand.



When you apply this kind of product thinking to your data, you begin to accelerate use-case delivery and innovation.

How do you deliver this vision and put your data to work? In a modern enterprise, where immediate access to data is critical to being a competitive differentiator, you need to think of the data movement and access in a fundamentally different way. Your approach to data pipelines has to incorporate five essential components. They are:

- **Streaming:** Built for real-time data access instead of processing data in batches, so you can deliver a consistent view of the most recent data anywhere it's needed, in the right format, the moment it occurs. The only way to do this is to adopt data streaming to maintain real-time, high fidelity, event-level repositories of reusable data within your organization, rather than pushing periodic, low fidelity snapshots of data to external repositories. And schemas act as the data contract between the producers and consumers, ensuring data compatibility.

- **Decentralized:** Supports domain-oriented, decentralized data ownership and architecture, allowing teams closest to the data to create and publish independent data streams that can be shared and reused. At the same time, your teams are empowered with a self-serve data infrastructure platform that enables data streams to be discovered, consumed and shaped into multiple contexts on the fly, and multiplexed and shared from and to any destination, everywhere.

data with intuitive search, discovery and lineage so developers and engineers can innovate faster.

- **Declarative:** Separates the data flow logic from the low level operational details of how the data will be processed. So instead of describing the flow of data as a sequence of operational steps, you must be able to simply describe the flow of data — where the data comes from, where it's going and what it should look like along its journey, while the infrastructure automatically flexes to handle changes in data scale.

- **Developer oriented:** Brings agile development and engineering practices to your pipelines. This means enabling your teams to build modular, reusable data flows that can be tested and debugged in an iterative, automated and orchestrated fashion and independently of production environments. It also means that integrating with version control and CI/CD systems so pipelines can be versioned, forked and deployed into different environments. And lastly, it should enable people with different skills and needs to collaborate on developing pipeline components using tools that support both visual IDEs (integrated development environments) and code editors.

When you reimagine the flow of data in the context of these five foundational principles, you enable data-as-a-product thinking and maximize the usability of data, making it easy for different teams to produce, share and consume trustworthy data assets. Together, these principles reinforce one another to promote data reusability, engineering agility and greater collaboration and trust within the organization.

How Do You Get Pipelines Right?

The next time you're building an application that needs to be informed by data, ask yourself if your data infrastructure can support your data needs for real-time, ubiquitous, self-service and governed access to high quality data streams. Question why you need to default your assumption to the status quo, which is batch-oriented, centralized, ungoverned and inflexible. Identify solutions that will help you not only solve your data needs for today, but will also serve

- Learn how innovative companies are adopting a transformational next-gen approach. Netflix discusses how it is building its data mesh with a data-streaming platform. Adidas does the same in this article.
- This e-book does a fantastic job of explaining how to implement your data mesh with Apache Kafka.
- Learn more about how leading data-streaming vendors like Confluent, who pioneered the category and are the original creators of Apache Kafka, deliver a new approach to streaming data pipelines.
- Get your hands on a demo that shows how to build streaming pipelines between operational databases such as Oracle and MongoDB.

TNS

Jon Fancey leads product management for Stream Designer at Confluent and has two decades of experience working in the app and data integration space as a developer, architect, consultant and product leader. Prior to joining Confluent, Jon created Azure Integration...

Read more from Jon Fancey →

RELATED STORIES

Break Your Bottlenecks: Don't Let Kafka Challenges Hold You Back

3 Trade-offs to Consider When Deploying Apache Kafka in the Cloud

Data Mesh Requires a Change in Organizational Mindsets

Governance: Your Data Mesh Self-Service Depends on It

Bring Sanity to Managing Database Proliferation

Put Your Data to Work: Top 5 Data Technology Trends for 2023
14 December 2022

Building Event Streaming Applications in .NET
8 December 2022

What Are Apache Kafka Consumer Group IDs?
6 December 2022

How Fully Managed Connectors Make Apache Kafka Easier
6 December 2022

Modernize Your Mainframe With Confluent on IBM zSystems
5 December 2022

Data pipelines: The what, why, and how
2 December 2022

analyses.

DevOps
Kubernetes
Observability
Service Mesh
DevOps Tools

CHANNELS

Podcasts
Ebooks
Events
Newsletter
TNS RSS Feed

THE NEW STACK

About / Contact
Sponsors
Sponsorship
Contributions