DZone

DZONE TREND REPORT

MAY 2022

Data Pipelines

Ingestion, Warehousing, and Processing

BROUGHT TO YOU IN PARTNERSHIP WITH

dremio

rudderstack™

hightouch

Table of Contents

Welcome Letter

By Caitlin Candelmo, Head of Publications at DZone

Data. It's everywhere and almost everything we touch collects it. Even more, the way that organizations collect, store, and interpret data today is under a microscope. The continued rise of data-driven applications provides a fascinating yet highly debated-upon topic: What is the value of data?

The answer to that question depends on the perspective.

I consider myself to be a (very) amateur gardener. Each year when the weather gets warmer, I find myself planning out my garden. I meticulously measure the distance between each seed planted, water it every day, and wait for the seeds to sprout. At the end of the growing season, I will be happy if I'm able to gather a few handfuls of vegetables.

My success is found in the relaxing practice of getting my hands dirty and tending to the plants.

A professional farmer must also plan out their garden, but on a much larger scale. And their success is based on how much is grown and sold. To me, as an amateur gardener, the value of gardening is in the practice, not the yield. For someone such as a farmer, whose livelihood depends on their yield, they would place a different value on the act.

While we both value gardening, *how* we ascribe that worth and our perspectives on the definition of success greatly differ. The value of a single seed being planted depends on the desired end result.

Much is similar in how we define the value of data. An individual contributor can use data to better improve their personal work habits, but an organization relies on data to make high-level decisions that have the potential to impact thousands, if not millions, of people.

In DZone's 2022 "Data Pipelines" Trend Report, we aim to provide some perspective on the value of data with a particular focus on the pipeline itself. We take a step back from the end result of the data point and focus on the methods of extracting, storing, and processing it.

Featured articles written by DZone community members cover the key components of a data pipeline, ideas for how to overcome challenges with designing data pipelines at scale, and the differences between ETL, ELT, and reverse ELT.

You will also find our contributors' expert insights into how to modernize testing with data pipelines, data security considerations for cloud data warehouses, and engineered decision intelligence. Our research delves into the role of data pipelines in the software profession and provides community-offered advice on ETL and ELT techniques along with recommended data pipeline design.

All of the well-tilled content in this Trend Report is geared toward providing the proper tools to help you define the value of data and determine how to leverage your data pipeline to best harvest it.

Happy gardening!

Caitlin Candelmo

ABOUT

DZone Publications

Meet the DZone Publications team! Publishing Refcards and Trend Reports year-round, this team can often be found reviewing and editing contributor pieces, working with authors and sponsors, and coordinating with designers. Part of their everyday includes collaborating across DZone's Production team to deliver high-quality content to the DZone community.

DZone Mission Statement

At DZone, we foster a collaborative environment that empowers developers and tech professionals to share knowledge, build skills, and solve problems through content, code, and community.

We thoughtfully — and with intention — challenge the status quo and value diverse perspectives so that, as one, we can inspire positive change through technology.

Meet the Team

Caitlin Candelmo, Head of Publications at DZone

@CCandelmo on DZone | @caitlincandelmo on LinkedIn

As the Head of Publications, Caitlin oversees the creation and publication of all DZone Trend Reports and Refcards. She helps with topic selection and outline creation to ensure that the publications released are highly curated and appeal to our developer audience. Outside of DZone, Caitlin enjoys running, DIYing, living near the beach, and exploring new restaurants near her home.

Lindsay Smith, Publications Manager at DZone

@DZone_LindsayS on DZone | @Smith_Lindsay11 on Twitter

Lindsay is a Publications Manager at DZone. From working with expert contributors on editorial content to assisting Sponsors with report materials, Lindsay and team oversees the entire Trend Report process end to end, delivering insightful content and research findings to DZone's global developer audience. In her free time, Lindsay enjoys reading, biking, and walking her dog, Scout.

Melissa Habit, Publications Manager at DZone

@dzone_melissah on DZone | @melissahabit on LinkedIn

As a Publications Manager, Melissa co-leads the publication lifecycle for Trend Reports and Refcards — from coordinating project logistics like schedules and workflows to conducting editorial reviews with authors and facilitating the design stage. She often supports Sponsors alongside her Client Success teammates. At home, Melissa passes the days reading, sewing, woodworking, and (most importantly) adoring her cats, Bean and Whitney.

John Esposito, Technical Architect at 6st Technologies

@subwayprophet on GitHub | @johnesposito on DZone

John Esposito works as technical architect at 6st Technologies, teaches undergrads whenever they will listen, and moonlights as research analyst at DZone.com. He wrote his first C in junior high and is finally starting to understand JavaScript NaN%. When he isn't annoyed at code written by his past self, John hangs out with his wife and cats Gilgamesh and Behemoth, who look and act like their names.

ORIGINAL RESEARCH

Key Research Findings

An Analysis of Results from DZone's 2022 Data Pipelines Survey

By John Esposito, PhD, Technical Architect at 6st Technologies

From March to April 2022, DZone surveyed software developers, architects, site reliability engineers, platform engineers, and other IT professionals in order to understand what problems data pipelines pose and how they are solved.

Major research targets were:

1. ETL and ELT techniques
2. The role of data pipelines in the software profession
3. Data pipeline design

Methods: We created a survey and distributed it to a global audience of software professionals. Question formats included multiple choice, free response, and ranking. Survey links were distributed via email to an opt-in subscriber list, popups on DZone.com, the DZone Core Slack Workspace, and LinkedIn. The survey was open from March 21–April 8, 2022, and it recorded 317 responses.

In this report, we review some of our key research findings. Many secondary findings of interest are not included here. Note that this is the first time we have asked many of these questions: We expect better trend analysis using future survey data.

Research Target One: ETL and ELT Techniques

Motivations:

1. Data, in itself, is unexplained and therefore useless, and the intrinsic variety and extrinsic utility of data are so massive that data pipelines cannot be generically solved. In such wide-open cases, it is especially helpful to have a flexible and broad toolkit of approaches (particularly starting points) top of mind. We wanted to get a better sense of what these approaches and starting points are.

2. Key decisions about which transformations will take place where and how may be hard to rearrange later. This is truer of linear processes with mostly defined control flows (line data pipelines) than modular processes with deliberately open control flows. To gain a better understanding of how these hard-to-untangle decisions are made, we wanted to understand which ETL/ELT techniques are applied at which stages of data pipelines.

HIGH-LEVEL APPROACHES TO ETL/ELT

Most mature software systems offer relatively straightforward proximate extract/load mechanisms. That is, in our experience, *E* and *T* tend to be simpler "at the edge." We wanted to know how these start-/end-of-pipeline tasks compare with in-between tasks, so we asked:

How often do you extract, transform, and load data using each of the following approaches? Rank from most frequently used (1) to least frequently used (9).

Results (n=228):

SEE TABLE 1 ON NEXT PAGE

Table 1

RANKING OF APPROACHES TO ETL			
Approach	**Overall rank**	**Score**	**n=**
Manual database imports/exports (csv, text, etc.)	1	1,137	213
Ad hoc batch scripts run on a schedule	2	1,129	204
Reusable but manually run SQL scripts	3	1,117	214
Specialized data pipelining tool	4	1,106	214
General-purpose integration tool	5	1,047	218
Specialized ETL tool	6	1,026	205
Custom-built middleware	7	1,022	228
Ad hoc batch scripts run on demand (manually initiated)	8	1,004	200
Ad hoc streaming scripts	9	998	211

Observations:

1. Manual database exports — the least automated, most edge-proximate approach — is the most common approach used, albeit by a small margin. This is consistent with our impression of the maturity of basic exporting tools.
2. Ad hoc batch scripts run on a schedule are a very close second, significantly more common than ad hoc batch scripts run on demand or ad hoc streaming scripts. We take scheduled batch scripts to be the middle ground of bespoke data pipelining: more "designed" than manually initiated scripts and less "designed" than streaming scripts, which we take to be rarely dispreferred to batch scripts if we abstract from complexity and performance — problems that must be solved by lower-level design.
3. We were somewhat surprised that custom-built middleware was used third least commonly for ETL, but we take this to be a reflection of our (experience) bias toward sometimes-antiquated enterprise systems and the growing maturity of off-the-shelf middleware solutions, including specialized data pipelining solutions.
4. Significant differences in ETL approach rankings obtain between senior (>5 years of experience as a software professional) and junior (≤ 5 years) respondents. Among junior respondents, specialized data pipelining tools and manual database imports/exports are ranked first and second respectively, while among senior respondents, ad hoc batch scripts run on a schedule and reusable but manually run SQL scripts are ranked first and second.

TRANSFORMATION LOCI AND DIRECTIONS

Presumably, data pipeline activities often comprise a non-commuting ring, or the ETL/ELT distinction would be a DINO (distinction in name only). Of course, this is not universally the case: Many data pipelines include non-serial transformations. We knew our survey could not dig deep enough into the weeds to capture a good low-level picture of data pipeline commutativity/parallelizability. However, we did want to know, at a high level, where the transformation logic is located and which direction the data flowed, so we asked:

What percent of your organization's pipelined data is transformed (a) before loading into the target system vs. (b) within the target system?

And:

What percent of your time spent working on data pipelines is spent moving data (a) FROM an operational system (that uses the data directly) TO a central repository (e.g., a data warehouse) vs. (b) FROM a central repository (e.g., a data warehouse) TO an operational system (that uses the data directly)?

Results (n=284 and n=286, respectively):

SEE TABLE 2 ON NEXT PAGE

Table 2

DATA TRANSFORMATION LOCALITY AND DIRECTION			
Percent of Pipelined Data Transformed			
	Avg.	**Sum**	**n=**
Transform data before loading into target system	60.1	17,060	284
Transform data within target system	40.6	11,297	278
Percent of Time Spent Moving Data			
	Avg.	**Sum**	**n=**
From operational system to central repository	55.4	15,845	286
From central repository to operational system	44.2	12,117	274

Observations:

1. ETL beats ELT by a 60/40 split of arithmetic means. That is, modern data pipelines retain a majority of the responsibility for transformation *within* the pipeline. We take this to be consistent with the ongoing increase of complex off-the-shelf ETL solutions, especially low-code offerings.
2. Movement of data from operational systems to central repositories occupies somewhat more time than movement from central repositories to operational systems — but not by a wide margin (55.4% vs. 44.2%, on average). We take the difference to suggest that: (a) ETL work is still often done for the sake of business decision-making (i.e., central repositories are naturally suited to data warehousing), and (b) data architecture may retain a degree of centralization greater than modern, web-first, distributed application architectures.

We hypothesize that the relatively small gap in time spent may derive from the increasing distributedness of application architectures, where operational data locality may be achieved more efficiently without running through a central hub — but we do not currently have any correlates to support this hypothesis.

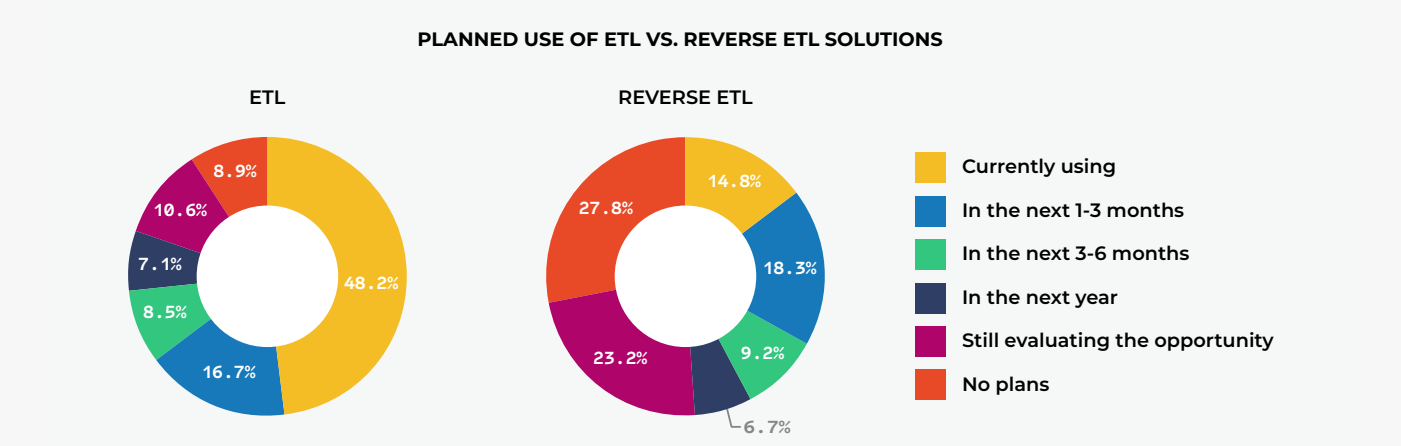
USE OF ETL AND REVERSE ETL SOLUTIONS

Given that in-pipeline work remains dominant, and that junior software professionals are highly likely to use a data pipelining solution, we might suppose that a large number of respondents currently use an ETL solution and that more respondents will use an ETL solution over time. The latter hypothesis we reserve for future surveys; the former we can address based two questions we asked in this survey:

When (if ever) are you planning to use an ETL solution? And: When (if ever) are you planning to use a reverse ETL solution?

Results (n=282 and n=284, respectively):

Figure 1



Observations:

1. Almost half of respondents (48.2%) are currently using an ETL solution. This is consistent with our interpretation of the two questions discussed above and with our experience in enterprise consulting.
2. 16.7% of respondents reported that they are planning to use an ETL solution in the very short term (1-3 months). Together with the "current" responses, this suggests that a significant majority of software professionals use or will soon use an ETL solution. This is good news for developers who disprefer data pipelining activities (see discussion below) as well as data administrators and architects who enjoy learning ETL tooling — and, of course, for ETL consultants and solution vendors.
3. ETL solutions were reported to be used far more commonly than reverse ETL solutions (48.2% vs. 14.8%). We imagine that this may be because movement toward operational systems is more coupled to the multiform details of at-the-edge interface directly with those systems and, hence, are harder to generalize than movement toward less-multiform centralized repositories. In other words, ETL solutions are easier to develop than reverse ETL solutions.
4. Less than a tenth (8.9%) of respondents reported no plans to use an ETL solution. This suggests that ETL remains a pain point nontrivial enough for a large majority of software professionals to feel some need for an ETL solution.
5. ETL use and future plans vary significantly by company size. Respondents at large (>1k employees) and small (<20 employees) companies were most likely to be using an ETL solution right now, and respondents at the smallest companies (1-4 employees) were most likely to report no plans to use an ETL solution.

Research Target Two: The Role of Data Pipelines in the Software Profession

Motivations:

1. Some people love messing with data. Others do not. *Prima facie*, it seems there is something admin-like about preferring data pipelines and something creator-like about dispreferring data pipelines. We wanted to understand how software professionals' preferences relate to data pipelining at a less intuitive and more granular level.
2. Although end-to-end data pipelining does not have a generic solution, many kinds of extraction, transformation, and loading operations done within a data pipeline are similar enough to one another that Turing-completeness is overkill for expressing these transformations. Automation naturally flows into such a complexity vacuum, but in our experience, data pipelines are not automated as much as they (theoretically, fruitfully) might be. We wanted to understand why.

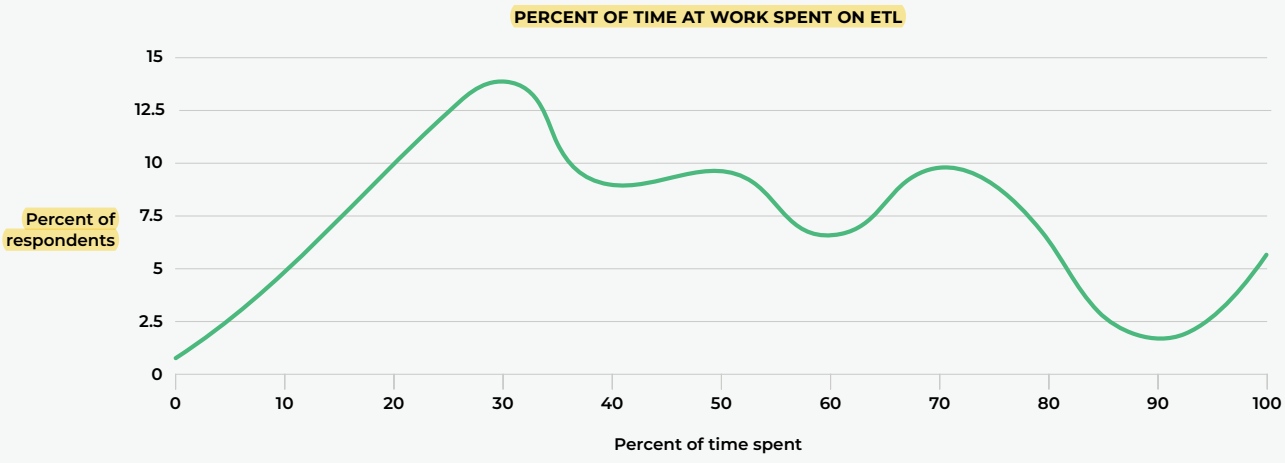
TIME SPENT ON ETL/ELT

Von Neumann and DNA both declare: There is information and there is doing stuff. But how much of computation (or life) is spent on each? And how does the whole machine move back and forth between the two? In the biological world, the relevant topic is (perhaps) the ribosome; in the software world, the relevant topic is the data pipeline. We wanted to know how much effort, measured as time, software professionals spend on the movement from storage to use and vice versa, so we asked:

What percent of your time at work is spent on data extraction, loading, and/or transformation?

Results: (n=281):

Figure 2



Observations:

1. On average, respondents reported spending almost half (44.7%) of their time at work on data extraction, loading, and/or transformation. We interpret this ambivalently: On the one hand, we expected this number to be lower; on the other, the consulting projects that took far more time than anticipated did so mainly because of data pipelining issues.

   Note, however, that the standard deviation was quite high (26.9), and that the response curve is fairly broad but skewed left. So the arithmetic mean probably makes the amount of time spent on ETL look higher than it really is. The peak of the curve (20-30% of time spent at work) fits better with our subjective experience, but in future surveys, we hope to articulate this question further.

2. Of time spent on data pipelines, respondents reported spending significantly more time on batch (48.6%*) vs. stream (33.9%*) processing — with similar results for time spent on offline (51.9%*) vs. online (44.8%*) data processing. These time spent data are consistent with the techniques-used findings (discussed above). **Indicates average.*

IMPACT OF DATA PIPELINING ON SKILL ACQUISITION

Unfamiliar and distinct problem domains are more likely to demand upskilling than familiar or poorly defined problem domains — the former because current skills are evidently adequate, and the latter because applying existing skills to a poorly defined domain is more likely to generate some success than trying to find the skill perfectly suited to something amorphous.

Because both sources and sinks of data pipelines are likely to be reasonably well coupled to business rules, we take data pipelining to benefit more from domain knowledge than, for example, general-purpose enterprise application architecture — hence, the popular "patternization" of the latter. For this reason, we took data pipelining to be highly stimulating of upskilling. To test this supposition, we asked:

Have you ever learned a new technology (language, library, platform) in order to implement a data pipeline?

Results (n=280):

Table 3

SKILL ACQUIRED TO IMPLEMENT DATA PIPELINES		
	Percent	**n=**
Yes	60.4%	169
No	39.6%	111

Observations:

1. Our guess was correct: A significant majority (60.4%) of respondents reported that they learned a new technology in order to implement a data pipeline. This suggests that data pipelines are technically challenging enough to require upskilling to a significant degree. However, in future surveys, we intend to compare this upskill-stimulating response rate with other technical problem domains (e.g., rearchitecting for distributedness, adopting CD, handling scaling problems) to see whether data pipelining is more or less likely to require upskilling than other software development tasks.

2. Senior respondents were significantly more likely than junior respondents to have learned a new technology in order to implement a data pipeline (65.2% vs. 45.5%, respectively), but we cannot tell how much of this is due to more time in which to encounter data pipelining work vs. a need to absorb new skills related to data pipelining changing over time.

3. The most commonly reported new technologies learned for data pipelining were Python and Apache Spark. We take Python to be especially useful for data pipelining because it combines ease of use with excellent performance — bash on steroids, for the purpose of data pipelining.

PREFERENCE FOR DATA-PIPELINING-RELATED WORK

The strong labor market for software developers means that many developers can afford to quit if they are bored, so it is important for employers to provide developers interesting work. Further, for a software professional in a high-opportunity market, it is important to know what kind of work is enjoyable and sustainable. Sympathy with both of these perspectives led

us to want to know how much developers prefer or disprefer data pipelining compared with other kinds of software work, so we asked:

Rank the following software-development-related activities from most preferred (1) to least preferred (14).

Results (n=252):

Table 4

RANKING OF PREFERENCES FOR WORK RELATED TO SOFTWARE DEVELOPMENT			
Task	**Overall rank**	**Score**	**n=**
Designing multi-application system architecture	1	1,827	200
Gathering business requirements	2	1,823	238
Designing application architecture	3	1,804	200
Analyzing data	4	1,635	201
Designing algorithms for performance	5	1,630	204
Designing algorithms for future maintainability	6	1,622	199
Physically written code	7	1,576	192
Data extraction, transformation, and loading (in any order)	8	1,559	211
Reviewing code written by others	9	1,523	222
Analyzing systems apart from data	10	1,504	202
Mentoring junior engineers	11	1,490	225
Reading documentation and tutorials	12	1,456	221
Optimizing user experience	13	1,415	201
Refactoring code	14	1,388	204

Observations:

1. Of data-pipeline-related activities, analyzing data (the least data-pipeline-specific activity) ranked highest (no. 4). ETL proper ranked much lower (no. 8), slightly below the midpoint of the activities list.
2. Designing multi-application system architecture is the most preferred activity but gathering business requirements comes in a close second. This is consistent with a respondent pool skewed toward those more experienced.
3. Indeed, preferences significantly differ by experience level. A few differences of note:
    - Senior respondents slightly preferred gathering business requirements to designing application architecture (!). From our own experience, we suppose that this may be due to the mental toll taken by building and rebuilding systems that turn out not to help anyone because the requirements were wrong in the first place — a toll that accrues painfully over time.
    - Junior respondents ranked physically writing code at number 3 vs. 5 among senior respondents. This may reflect greater complexity of coding tasks assigned to senior respondents, which may require more forethought before fingers-to-keyboard, independently of "raw" preference.
4. Senior respondents preferred designing algorithms for performance (no. 4) more than designing algorithms for maintainability (no. 6) — the inverse of junior respondents' preferences (no. 4 for maintainability vs. no. 7 for performance). This may reflect the relatively low need for algorithmic performance optimization in much entry-level work.

Research Target Three: Data Pipeline Design

Motivations:

1. Linear processes with non-obvious sequencing (like many data pipelines) sometimes make design trade-offs harder to work around later — i.e., the contract for any given stage depends on the previous stages in a way that is not always obviously necessary. We wanted to understand how software professionals use formal and informal abstraction methods to gain a higher-level understanding of data pipelines to facilitate trade-offs that result in less brittle design.

2. Data persistence approaches are determined by more considerations than ETL/ELT pipelining. But people who move data around offer a "data-in-motion" perspective on data persistence design that pure data architects or application developers (as such) do not automatically enjoy. We wanted to shed light on the broader problem of data storage by way of the data-pipelining perspective.

MODELING DATA PIPELINES

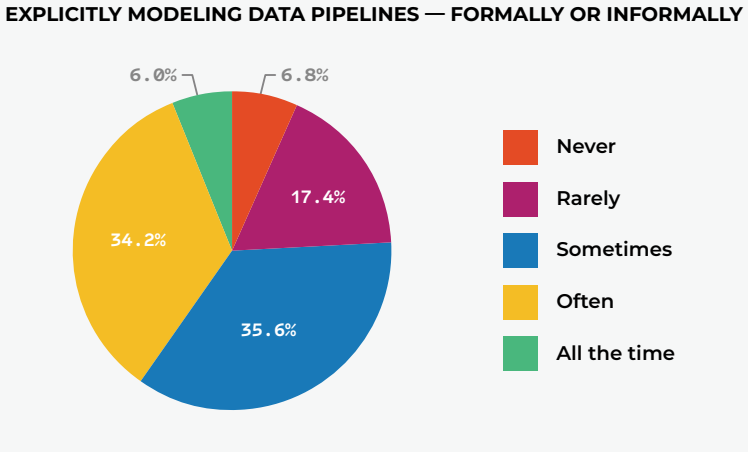
Procedural software design is sometimes frowned upon (e.g., by object-oriented and functional programmers) because it can be brittle, hard to understand, and a poor representation of the problem space. But because data pipelines both span multiple systems and can be represented as functional transformations, data pipeline design often requires both procedural and functional thinking. Explicit modeling can help where paradigms are mixed, and especially where the solution is potentially brittle and problem-anisomorphic. However, as with formal verification of software, a fully explicit model is not always (judged to be) worth the effort.

We wanted to know how often software professionals do model data pipelines, so we asked:

How often do you explicitly model data pipelines (formally or informally)? Note: Take "explicitly" in the broadest sense. Examples: Drawing data flow diagrams, sequencing vector space homomorphisms, poking around TensorFlow, and writing data-transforming pseudocode all count as "explicit modeling."

Results (n=281):

Figure 3



Observations:

1. The distribution of responses appears broad enough to seem noncommittal. Roughly one third of respondents reported often explicitly modeling data pipelines, and another third reported sometimes, while only 6.8% reported that they never explicitly model data pipelines.

2. No significant differences in use of explicit modeling of data pipelines were observed when segmenting responses by experience (senior vs. junior) or company size.

Note: In future research, we plan to investigate what sorts of formal or informal modeling is done for both data pipelining and other software analysis/understanding tasks.

MANUAL IMPLEMENTATION OF BASIC DATA TRANSFORMATIONS

A few transformations appear commonly across many data pipelines, presumably determined by intended use. For instance, an ETL pipeline might sort data before load so that the target system can select a sort-optimized aggregation algorithm, or row-level timestamped data might be aggregated to bucketed or unbucketed totals for presentation-layer access without expensive relational joins. Sorting and summing are common data pipelining tasks and often taught in undergraduate data structures and algorithms courses. But the most commonly required and thoroughly studied tasks are also the most likely to be automated enough to require little or no manual work by developers; programmers may not often need to implement such tasks manually.

We wanted to know how often developers actually implement these two basic transformations manually, so we asked:

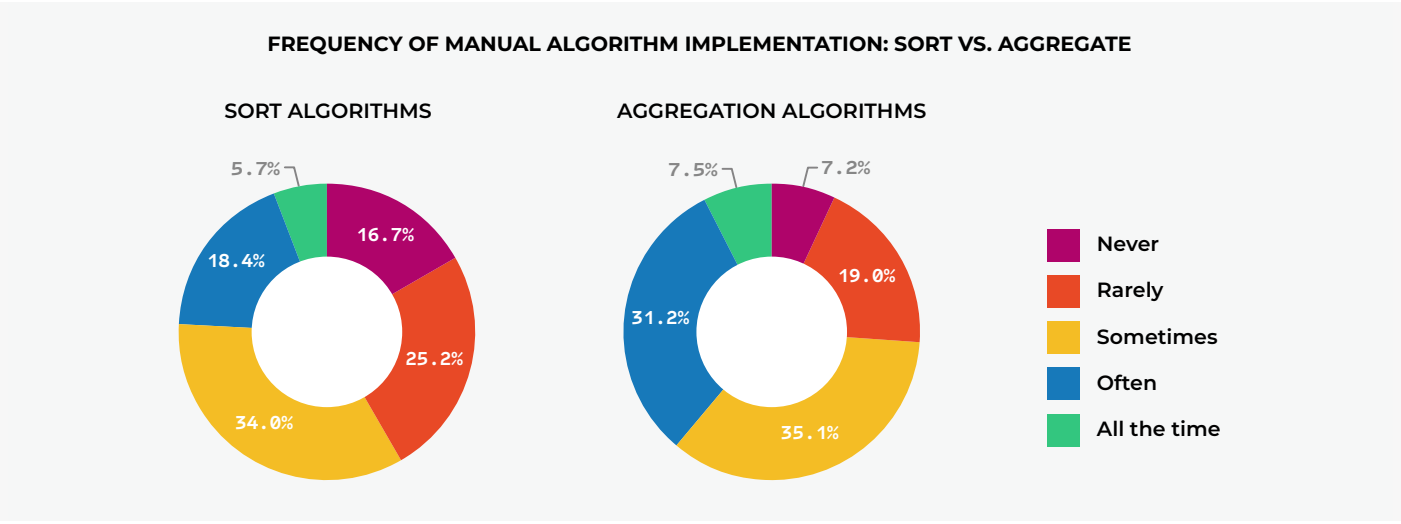
How often do you manually implement sort algorithms?

And:

How often do you manually implement aggregation algorithms? Note: By "aggregation," we mean any operation that reduces individual data points to higher-level, more abstract representations — i.e., anything from arithmetic summing to Petri nets to garbage collection.

Results (n=282 and n=279, respectively):

Figure 4



Observations:

1. We were surprised to see that almost a fifth (18.4%) of respondents reported that they often manually implemented sort algorithms. Because sort algorithm efficiency is highly sensitive to initial data state, in retrospect, we suppose that titling this survey "Data Pipelines" might bias respondents toward stronger interest in decisions that depend on initial data state.

2. Less surprisingly, junior respondents were more likely than senior respondents to report they often or always implement sort algorithms manually (19.3% vs. 17.7% and 8.8% vs. 4%, respectively). We imagine this is because junior respondents are closer to data structures and algorithms coursework, on the one hand, and because senior respondents are more likely to work at higher levels of software design than (presumably swappable) sort algorithm implementation.

   Still, the overall frequency of sort algorithm implementation is higher than we expected, and we intend to follow up with more data-structure/algorithm-focused questions in the future.

3. We were less surprised by how frequently respondents implement their own aggregation algorithms, especially given the topic of this survey. These numbers varied by experience level somewhat inversely: Senior respondents were slightly more likely to report they often implement their own aggregation algorithm (33.2% vs. 29.8%), though also less likely to report all the time (6.6% vs. 10.5%).

SLOWLY CHANGING DATA WORRIES

One particularly frustrating problem with both data architecture in general and data pipelines in particular: dealing with data that may never change, or if it ever changes, then will only change a few times but cannot be absolutely relied upon not to change (e.g., a human's full name or driver's license number). Certain specialized storage and retrieval strategies may assume that slowly changing data is never-changing — a risk of occasional incorrectness measured against a benefit of design simplicity and (often) runtime performance.

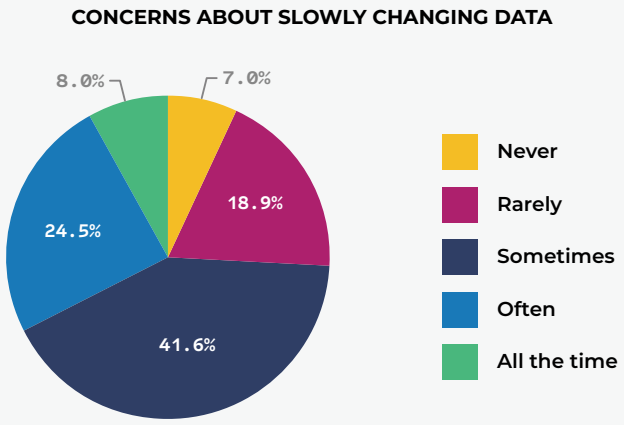
The same is true of data pipelines more broadly: For instance, the data pipeline designer may suppose that certain datapoints (e.g., technical attributes of a manufactured product) need not be queried for every load, but they may build in additional mechanisms (e.g., less frequently run "deep" queries) to avoid too much data drift.

We wanted to understand how much concern about slowly changing data factors into data pipeline design, so we asked:

When building data pipelines, how often do you worry about slowly changing data (e.g,. a user's job title)? Note: By "slowly changing data," we mean data that changes infrequently enough and in such a way that it seems too costly in terms of design complexity and/or performance to bake representation of these changes into the schema at a deep level.

Responses (n=286):

Figure 5



Observation: Responses distributed somewhat less blandly than for the previous two questions. 41.6% of respondents reported sometimes worrying about slowly changing data, and 24.5% reported often doing so. We take this to mean that slowly changing data is a highly significant problem that should not be assigned exclusively to magic data architects or DBMS administrators.

DATA-CODE ISOMORPHISM

Object-relational mapping (ORM) and the data access object (DAO) pattern both try to decouple data structure from business logic. But this decoupling may also come with performance hits and/or a fuzzy understanding of the problem domain. The degree to which data and code should share structure is a matter of particular judgment, and the kind of transformations done in a data pipeline depend on these application-level decisions. Still, as in all matters of judgment, preferences affect the default, uncorrected approach to relating data and code.

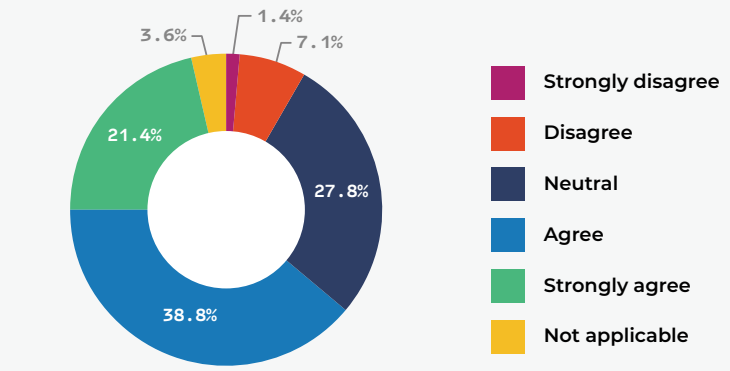
We wanted to know where software professionals' data-code isomorphism biases lie, so we asked:

Agree/disagree: Application code and the data it directly ingests should be as isomorphic as possible.

Results (n=281):

SEE FIGURE 6 ON NEXT PAGE

Figure 6



Observations:

1. Data-code isomorphism is generally desirable. Just over fifth of respondents (11.4%) strongly agreed with maximal data-code isomorphism, and an additional 38.8% agreed. However, over a quarter of respondents are neutral (27.8%).

   We expected this number to be higher: The whole point of ORMs and DAOs, which are nontrivial to implement well, is to keep data and code forms decoupled. Because more respondents prefer data-code isomorphism than we expected — and because, anecdotally, ORMs have come increasingly under fire, even in enterprise contexts — we intend to compare these results with future questions about object-relational and other "impedance mismatches" that affect both application architecture and data pipeline design.

2. No significant difference w.r.t. data-code isomorphism preference obtained among respondents bucketed by company size.
3. Junior respondents were slightly more likely to prefer data-code isomorphism, but again, results were mixed: More junior respondents strongly agreed, but fewer agreed and fewer disagreed (by almost a 3:1 margin).

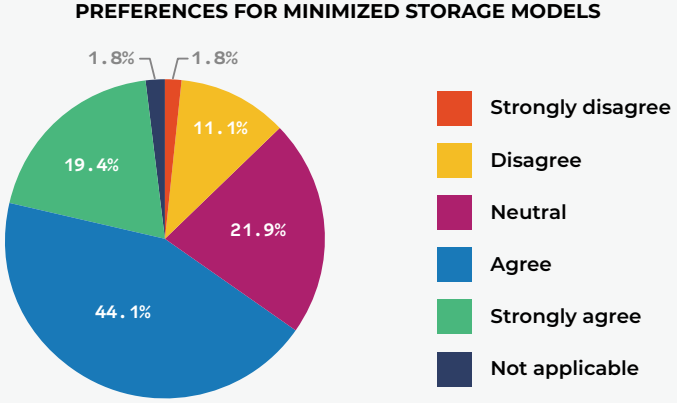
VARIETY OF STORAGE MODELS IN DATA PIPELINES

Perhaps data must be transformed from one relational schema to another, but perhaps the transformation must be more radical. Some transformations from one storage model to another are straightforward (e.g., rolling up daily sales into monthly sales), while others are not (e.g., transforming hypergraphs into relational tables without intractable schema complications, information loss, or awkward SQL). The best decision is, of course, a case-by-case judgment call, and again, we wanted to know where software professionals' biases (or experience-hardened lenses) lie. So we asked:

Agree/disagree: A data pipeline should include as few distinct storage models (relational, columnar, document-oriented, etc.) as possible.

Results (n=279):

Figure 7



Observations:

1. Preference for minimized storage model counts per data pipeline was stronger than preference for data-code isomorphism. Almost half (44%) of respondents agreed that a data pipeline should include as few storage models as possible, and another fifth (19.4%) strongly agreed. Only a fifth (21.9%) of respondents were neutral.

2. Responses varied significantly by whether respondents have previously learned a new technology for the purpose of implementing a data pipeline. Those who had learned a new technology for such purpose were almost three times more likely to disagree than respondents who had not (15.1% vs. 5.5%). This might mean that more flexible-minded respondents (with respect to data pipeline development) are more likely to accept multiple storage models within a data pipeline, but we cannot tell whether this flexibility caused or resulted from experience implementing data pipelines that contain multiple storage models.

Further Research

This was our first survey focused on the topic of data pipelining. In future research, we aim to examine lower-level data pipelining techniques in relation to organizational role, programming language preference, and attitudes toward software architecture. Our survey included material not published in this article, including: the relation of data warehousing responsibilities to data pipeline design preferences, attitudes toward normalized relational databases as systems of record (a common pattern), involvement in data warehousing activities, version control of data, impact of time pressure on data pipeline formalization, and more. Please contact publications@dzone.com if you would like to discuss any of our findings or supplementary data.

John Esposito, PhD, Technical Architect at 6st Technologies

@subwayprophet on GitHub | @johnesposito on DZone

John Esposito works as technical architect at 6st Technologies, teaches undergrads whenever they will listen, and moonlights as research analyst at DZone.com. He wrote his first C in junior high and is finally starting to understand JavaScript NaN%. When he isn't annoyed at code written by his past self, John hangs out with his wife and cats Gilgamesh and Behemoth, who look and act like their names.

CONTRIBUTOR INSIGHTS

An Overview of Key Components of a Data Pipeline

Key Components, Architecture Options, and Best Practices for Designing Data Pipelines

By Sudip Sengupta, Technical Writer at Javelynn

With the consistent growth of data-driven applications, the complexities of consolidating data from multiple sources for streamlined decision making is often considered a key challenge. While data forms the foundation of analytics and operational efficiency, processing big data requires holistic data-driven strategies for real-time ingestion and processing. To help with this, data pipelines enable organizations to aggregate and analyze huge datasets by defining a series of activities that convert raw data into actionable insights.

In this article, we dive into how a data pipeline helps process enormous amounts of data, key components, various architecture options, and best practices to achieve the maximum benefits.

What Is a Data Pipeline?

A data pipeline is the collection of tasks, tools, and techniques used to process raw data. Pipelines consist of multiple interrelated steps connected in series, which enable the movement of data from its origin to the destination for storage and analysis. Once data is ingested, it is taken through each of these steps, where the output of one step acts as the input for the subsequent step.

In the modern technology landscape, big data applications rely on a microservice-based model, which allows monolithic workloads to be broken down into modular components with smaller codebases. This encourages data flow across many systems, with data generated by one service being an input for one or more services (applications). An efficiently designed data pipeline helps manage the variety, volume, and velocity of data in these applications.

BENEFITS OF A DATA PIPELINE

Primary advantages of implementing an optimally designed data pipeline include:

IT RESOURCE OPTIMIZATION

When building infrastructure for data processing applications, a data pipeline enables the use of replicable patterns — individual pipes that can be repurposed and reused for new data flows, helping to scale IT infrastructure incrementally. Repeatable patterns also blend security into the architecture from the ground up, enabling the enforcement of reusable security best practices as the application grows.

INCREASE APPLICATION VISIBILITY

Data pipelines help extend a shared understanding of how data flows through the system along with the visibility of the tools and techniques used. Data engineers can also set up telemetry for data flows within the pipeline, enabling continuous monitoring of processing operations.

IMPROVED PRODUCTIVITY

With a shared understanding of data processing operations, data teams can efficiently plan for new data sources and flows, reducing the time and cost for integrating newer streams. Offering analytics teams comprehensive visibility of the data flows also enables them to extract meaningful insights, thereby helping to enhance the quality of data.

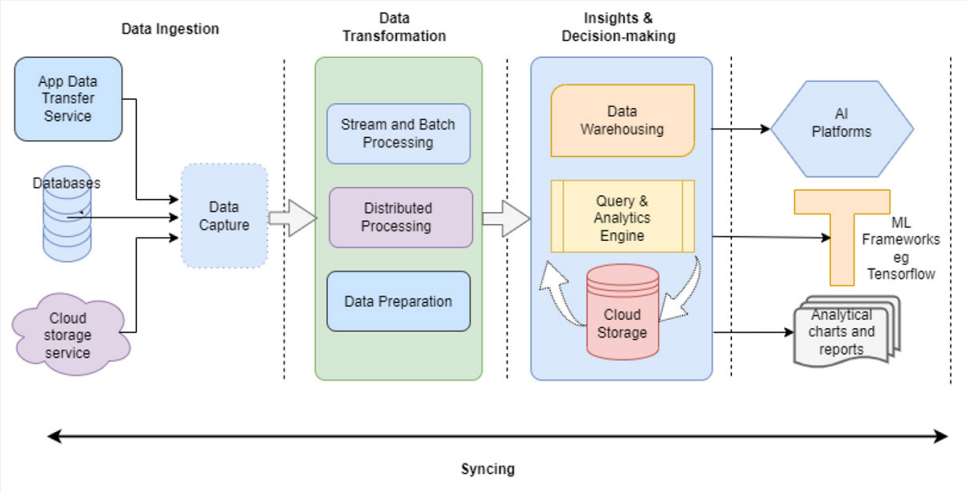
Key Components of a Data Pipeline

Data pipelines enforce the enrichment of data by moving it from one system to another, typically with different storage implementations. These pipelines enable the analysis of data from disparate sources by transforming and consolidating it into a unified format. This transformation consists of various processes and components handling diverse data operations.

PROCESSES OF A DATA PIPELINE

Although different use cases require different process workflows, the following are some common processes of a data pipeline:

Figure 1: Processes of a data pipeline



STAGES OF A DATA PIPELINE

Though the complexity of a data pipeline differs based on use case, amount of data to be churned, and the frequency of processing data, here are the most common stages of a data pipeline:

EXTRACTION/INGESTION

This stage involves the ingestion of data from its origin, also known as the source. Data entry points include IoT sensors, data processing applications, online transaction processing applications, social media input forms, public datasets, APIs, etc. Data pipelines can also extract information from storage systems, such as data lakes and warehouses.

TRANSFORMATION

This stage encompasses the changes that are made to the data as it moves from one system to another. The data is transformed to ensure it fits the format supported by the target system, such as an analytical application.

PROCESSING

This stage includes all the activities involved in ingesting, transforming, and loading data to the target destination. Some data processing activities include grouping, filtering, aggregating, and augmenting.

SYNCING

This process ensures the syncing of data across different data sources and the pipeline's endpoints. The stage essentially involves the updating of data libraries to keep data consistent across all stages of the pipeline's lifecycle.

Data Pipeline Architecture Options

The three primary design options for building data processing architecture for big data pipelines include stream processing, batch processing, and lambda processing.

STREAM PROCESSING

Stream processing involves ingesting data in continuous streams with data being processed in parts. This architecture's objective is a rapid processing approach that is meant primarily for real-time data processing with use cases such as fraud detection, log monitoring and aggregation, and user behavior analysis.

BATCH PROCESSING

With batch processing, data is collected over time and later sent for processing in batches. In contrast to stream processing, batch processing is a time-consuming approach and is meant for large volumes of data that are not needed in real time. Batch processing pipelines are commonly deployed for applications such as customer orders, billing, and payroll.

LAMBDA PROCESSING

Lambda processing is a hybrid data processing deployment model that combines a real-time stream pipeline with a batch processing data flow. This model divides the pipeline into three layers: batch, stream, and serving.

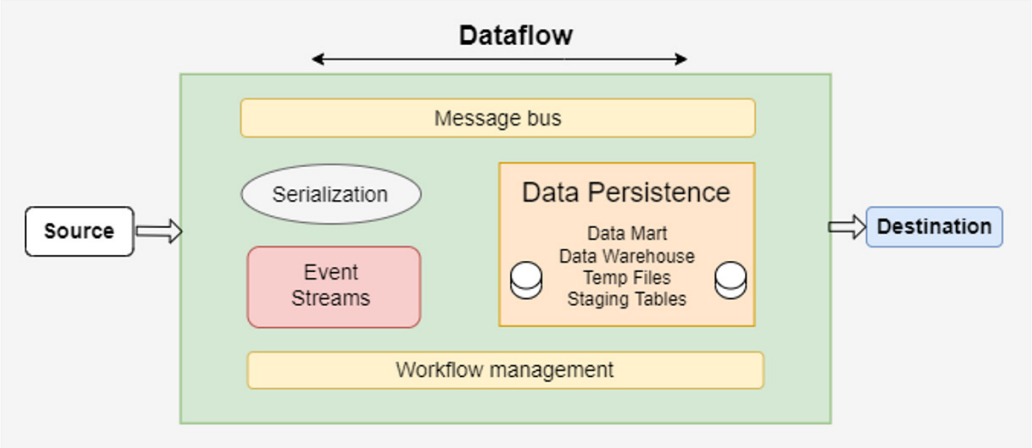
In this model, data is ingested continuously and fed into both batch and stream layers. The batch layer precomputes batch views and hosts the primary dataset. The stream layer handles data that has not been loaded to the batch view since batch operations are time consuming. The serving layer creates an index of batch views so that they can be occasionally queried in low latency.

Components of a Data Pipeline

Key components of a data pipeline include:

- **Data serialization** – Data serialization defines standard formats that make data conveniently identifiable and accessible and is responsible for converting data objects into byte streams.
- **Event frameworks** – These frameworks detect actions and processes that lead to changes in the system. Events are logged for analysis and processing to assist in decisions based on application and user behavior.
- **Workflow management tools** – These tools help structure tasks within a pipeline based on directional dependencies. These tools also simplify the automation, supervision, and management of pipeline processes.
- **Message bus** – Message buses are one of the most crucial components of a pipeline, allowing for the exchange of data between systems and ensuring compatibility for disparate datasets.
- **Data persistence** – The storage system where data is written onto and read from. These systems enable the unification of various data sources by enabling a standard data access protocol for different data formats.

Figure 2: Common components of a data pipeline



Best Practices for Implementing a Data Pipeline

To build effective pipelines, recommended practices for teams include enabling the execution of concurrent workloads, using extensible tools with inbuilt connectivity, investing in appropriate data wrangling tools, and enforcing data cataloging and ownership.

ENABLE THE EXECUTION OF CONCURRENT WORKLOADS

Most big data applications are required to run multiple data analysis tasks simultaneously. A modern data pipeline should be built with an elastic, multi-cluster, and shared architecture that can handle multiple data flows concurrently. A well-architected pipeline should load and process data from all data flows, which downstream DataOps teams can analyze for further use.

USE EXTENSIBLE TOOLS WITH INBUILT CONNECTIVITY

Modern pipelines are built on multitudes of frameworks and tools that communicate and interact with each other. Tools with inbuilt integration should be utilized to reduce the time, labor, and cost to build connections between various subsystems in the pipeline.

INVEST IN APPROPRIATE DATA WRANGLING TOOLS

Because inconsistencies often lead to poor data quality, it is recommended that the pipeline leverages appropriate data wrangling tools to fix inconsistencies in distinct data entities. With cleaner data, DataOps teams can gather accurate insights for effective decision-making.

ENFORCE DATA CATALOGING AND OWNERSHIP

It is important to keep a log of the data source, the business process owning the dataset, and the user or process accessing those datasets. This offers comprehensive visibility of the datasets that are safe to be used, enforcing trust in data quality and veracity. Cataloging also traces the data lineage, making it easy to establish the path of data flow across the pipeline.

Conclusion

Gartner predicts that the significance of automation will continue to rise to the extent that, "By 2025, more than 90% of enterprises will have an automation architect." Additionally, Gartner projected that, "by 2024, organizations will lower operational costs by 30% by combining hyperautomation technologies with redesigned operational processes."

As data sits at the core of achieving operational efficiency, processing enormous amounts of data is one of the key challenges modern organizations deal with. Data pipelines help organizations build reliable infrastructure to automate the aggregation and management of big data for analytics and business operations support. But before building a pipeline, it is recommended that organizations diligently analyze their business objectives, data sources, targets, and resources available.

Sudip Sengupta, Technical Writer at Javelynn

@ssengupta3 on DZone | @ssengupta3 on LinkedIn | www.javelynn.com

Sudip Sengupta is a TOGAF Certified Solutions Architect with more than 17 years of experience working for global majors such as CSC, Hewlett Packard Enterprise, and DXC Technology. Sudip now works as a full-time tech writer, focusing on Cloud, DevOps, SaaS, and cybersecurity. When not writing or reading, he's likely on the squash court or playing chess.

Make Cloud Storage Your Data Warehouse in Minutes with the Dremio Open Lakehouse Platform

Data is needed everywhere. But with current data architectures, it's still too difficult to go from data to decision making. Data warehouses are expensive, lock your data into proprietary formats, lack flexibility of different engines and make only a subset of data available. Data lakes are flexible but difficult to manage.

Data teams love Dremio because the open lakehouse delivers the best of both worlds.

Dremio is the *only* open lakehouse platform with a forever-free edition.

Get started now at Dremio.com

SPONSOR OPINION

Simplify Data Architectures With an Open Lakehouse

By Jeremiah Morrow, Director, Product Marketing at Dremio

Data teams today are facing an impossible set of circumstances. They are being inundated with new sources of customer and operational data. At the same time, as leaders catch on to the transformative potential of analytic insights across the business, there are a growing number of requests for access to that data from a variety of data consumers.

Many organizations have made large investments in proprietary data warehouses that struggle to scale with the volume and variety of modern datasets. That deficiency led to the adoption of data lakes — first HDFS and then cloud and on-premises object storage — as the destination for these new data volumes.

To satisfy data requests within the constraints of their current data architectures, data teams rely on a series of workarounds, including building complex Extract, Transform, and Load (ETL) processes to move data into proprietary data warehouses, and creating and managing multiple copies of data to use. These workarounds have introduced complexities, risk, and cost, and created a bottleneck so that new data requests can take weeks to fulfill. There is a simpler solution: the open lakehouse.

The open lakehouse combines the flexibility and scalability of the data lake with the data management, data governance, and ANSI-SQL functionality of the data warehouse, all directly on the data lake. It eliminates the complexities associated with moving, transforming, and copying data, provides broad access to the data lake, and reduces the data management and governance burden on data teams.

An open lakehouse enables companies to store data in a truly open format and process it with their choice of engines, thereby supporting all analytics use cases, including low-latency BI dashboards, ad hoc SQL queries, batch, and data science.

The open lakehouse gives businesses several advantages over traditional data architectures. It reduces time to insight by eliminating bottlenecks in the data pipeline, accelerating query performance, and providing self-service access to data for BI and ad hoc exploration. It makes data teams more productive by reducing the data management and data governance burdens. And it reduces the costs and complexities of traditional data architectures.

> ***eMag*** *is a Romania-based online retailer that has experienced significant growth in their digital business over the last several years. It quickly outgrew its IT infrastructure, and business users and data scientists who wanted access to data relied on reports that often arrived too late to be relevant. Data consumers who wanted ad hoc reports had to go through the BI group, which created a bottleneck.*
>
> *By implementing an open lakehouse solution, eMag was able to easily access and query data from multiple sources, including HDFS, MySQL, Microsoft SQL Server, and Amazon S3. They reduced report creation time from weeks to just hours, provided access to data for ad hoc analysis to data scientists and business analysts, and got insights into their marketplace in minutes instead of the next day.*

Get Started With Dremio Cloud

Dremio provides a SQL-first platform that makes it as easy to get started with an open lakehouse architecture as it is to get started with a cloud data warehouse. Visit dremio.com/get-started to register for Dremio Cloud for free, connect to your data lake, and start exploring your data in just a few minutes.

CONTRIBUTOR INSIGHTS

Challenges to Designing Data Pipelines at Scale

Assessing Key Challenges and Solutions to Effective Data Management

By Miguel Garcia, Head of Engineering at Nextail

The current challenge in the world of data is no longer in the processing capacity of volumes of data. Because of the performance of modern current streaming platforms and owing to a new generation of data repositories that allow decoupling computation from the storage layer, we can improve scalability with very low operational effort.

However, if we remember the famous 5 Vs of big data (volume, value, variety, velocity, and veracity), veracity and value are still a challenge for most companies today.

Data pipelines are fundamental to solve that challenge. Designing and building data pipelines at scale not only leads to more speed but doing so is maintainable and comprehensible through whole teams.

Processing Capacity for Data Volumes

A few years ago, the main technological challenges were in the following areas:

- **Data repositories** (volume and velocity) – Ingesting and managing huge volumes of data.
- **Data processing compute layer** (velocity) – Having a high-performance compute layer to launch thousands of data pipelines that allow ingesting a huge amount of data in a very fast way.
- **Integration adapters** (variety) – Developing adapters to integrate with the different components.

IT teams spent years building data platforms that allowed for those capabilities before starting to develop the first data pipelines. After that, all efforts went into ingesting huge volumes of data in a short period of time. But it all happened without focusing on real business value. Additionally, operating these data platforms required huge amounts of effort.

Now, with cloud adoption, the open-source community, the latest generation of cloud software, and new data architecture patterns, implementing those features is no longer a challenge.

DATA REPOSITORIES

There are multiple ways of providing a high-performance data repository to manage a huge volume of data in real time with low operational effort.

- **Data warehouses** – The new generation of data warehouses decouple storage from the compute layer, providing new, scalable capabilities based on different technologies, such as NoSQL, data lakes, or relational data warehouses with AI features.
- **Relational databases** – The latest relational databases that use sharding and in-memory features, providing OLTP and OLAP capabilities.
- **Streaming platforms** – Platforms such as Apache Kafka or Apache Pulsar provide the capacity to process millions of events per second and build real-time pipelines.

In this case, the key to success is to choose the best option for your use case.

COMPUTE LAYER

The following capabilities enable the execution of a large number of concurrent pipelines:

- **Serverless** – Cloud platforms provide easy, on-demand scaling.
- **New data repositories' computation layer** – Decoupling from storage allows delegating a large part of operations to be performed on the database without risking impacting other loads or scaling limitations, as with older, on-premises systems.
- **Kubernetes** – We can use the Kubernetes API to dynamically create job containers as Kubernetes pods.

INTEGRATION ADAPTERS

The open-source community, together with new cloud software vendors, provides a variety of data adapters to quickly extract and load data. This simplifies the integration between different software components, such as data repositories, ERP, and many others.

Current Data Challenges

The current challenge is providing data in a more comprehensive and reliable way. Data processes are one of the main factors of complexity. As a data or business analyst, in addition to having the information at the right time, we need to know the metadata about the data we are analyzing in order to make decisions.

- What does the data mean and what value does it provide? (Value)
- How is the data calculated? (Value)
- What are the sources of the data? (Value)
- How up to date is the data? (Veracity)
- What is the quality of the data? (Veracity)

In large enterprises, there is a lot of data, but also a lot of departments that sometimes work with similar, but different, data. For example, retail companies have two kinds of stock:

- **Countable stock** – This stock is theoretical and based on purchase orders and delivery notes.
- **Real stock** – This stock represents the items they have in their warehouses and stores.

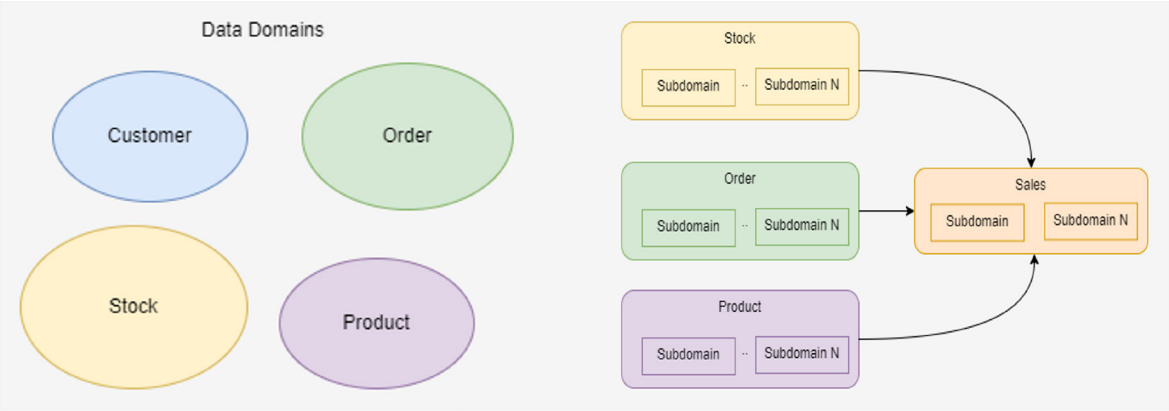
Usually, these values do not match for many reasons, such as delivery delays or other human errors. However, the stock data is the source for calculating indicators, such as sales, sales forecasts, or stock replenishment. The decision to purchase a new shipment of items based on the theoretical stock can cause thousands of dollars in losses and lead to a less sustainable world. We can only imagine the impact of these decisions on other sectors, such as healthcare.

When analysts make a decision, they need to know what data they are working with to assess the risks and make conscious choices. In the end, we are talking about transforming data into business value.

What Role Do Data Pipelines Play?

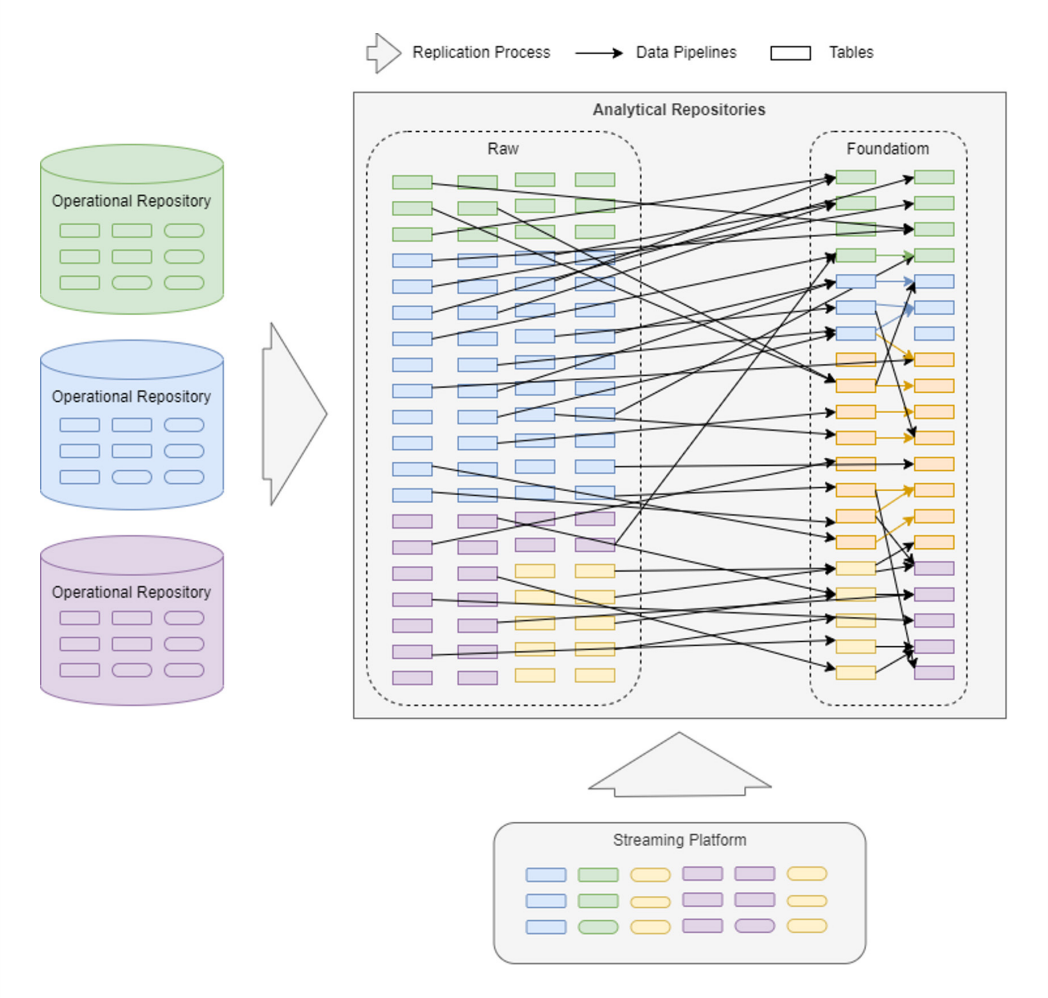
When we imagine data or read high-level articles, we visualize a very simple world with a few data domains:

Figure 1



But the reality is more complex than that:

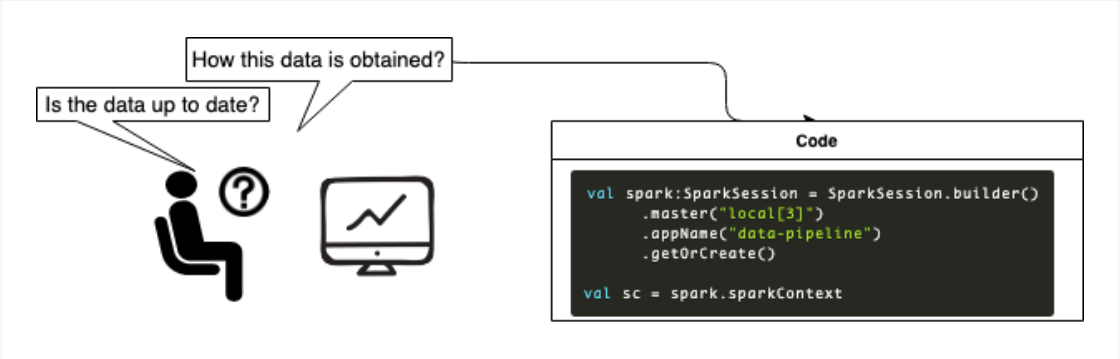
Figure 2



In big data scenarios, there are thousands of data pipelines at different levels that are continuously ingesting and consolidating data between data domains or data repositories. Data pipelines are among the most important components to provide a successful data platform, but it seems that, even today, we have some of the same challenges of building data pipelines at scale that we did many years ago, including:

- Providing metadata, such as the data lineage, in a dynamic and agile way
- Enabling easy collaboration between data analysts, data engineers, and business stakeholders
- Providing information to data analysts and stakeholders about data quality and freshness easily

Figure 3



What Is the Challenge of Building Data Pipelines at Scale?

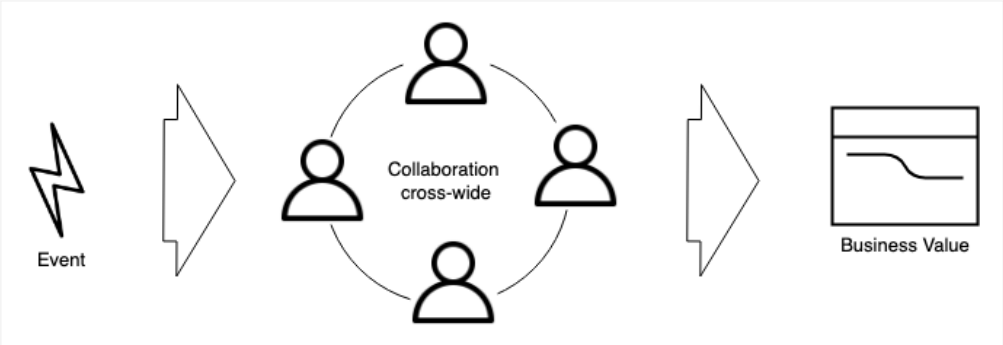
It seems the goals of data pipelines are focused on four main areas:

- Team collaboration and comprehension
- Dynamic data lineage
- Observability
- Data quality

TEAM COLLABORATION AND COMPREHENSION

The rules change quickly, and companies have to adapt fast. Data and information are more important than ever.

Figure 4



To provide value quickly, we need a heterogeneous team composed of data scientists, data engineers, analysts, and business stakeholders working together. They need to talk a similar language to be agile. When all our data pipelines are built with technologies such as a Spark or Kafka streams, it is too complex to have agile communication between technical and non-technical people.

The journey from data to information starts with data pipelines.

DATA LINEAGE

Metadata is important to allow the transformation of the data into business information. It provides a summary lineage accessible to everyone, increasing visibility, comprehension, and confidence in the data. We have to provide this information dynamically and not through a static documentation that never changes or a complex inspection process that never ends.

OBSERVABILITY

The observability of data pipelines is a major challenge. Usually, observability is oriented toward technical teams, but we need to raise this visibility to all stakeholders (business analysts, data analysts, data scientists, etc.).

DATA QUALITY

We need to evolve how we work with data and start applying the best practices of traditional software development. From data to Data as Code, we can use methodologies such as version control, continuous integration, or testing.

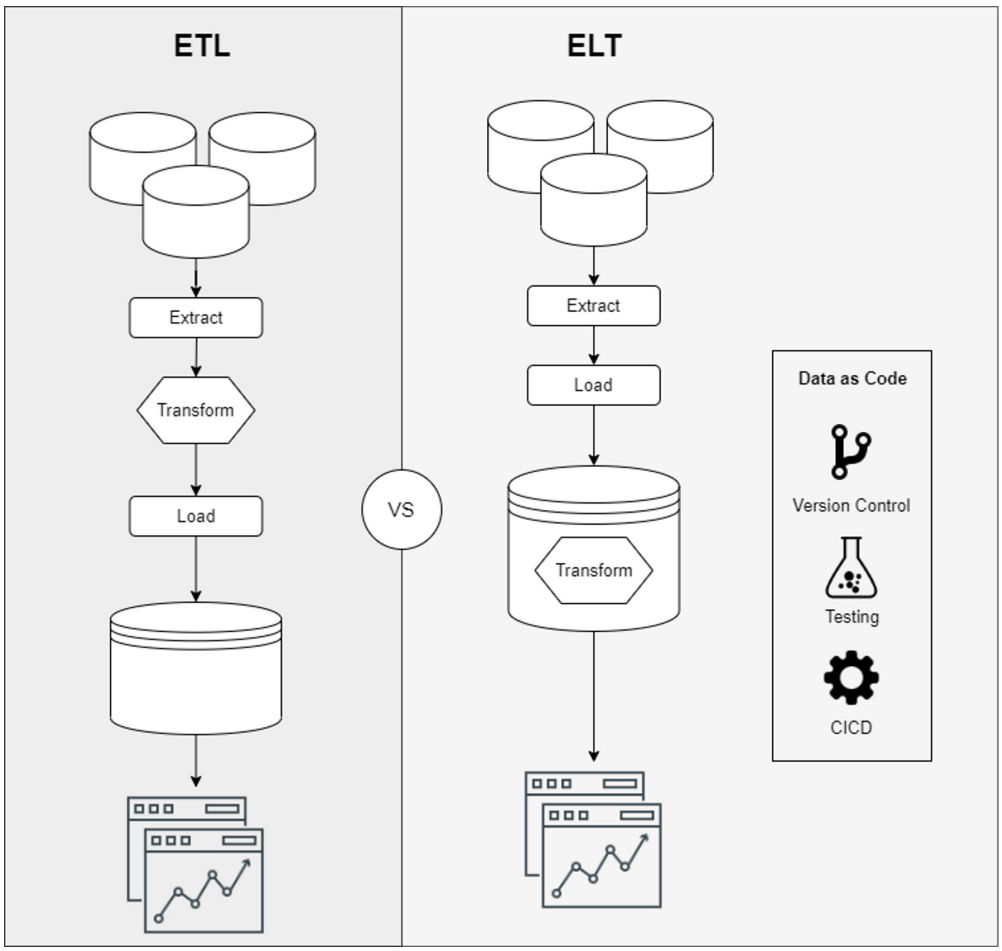
How Are Data Pipelines Evolving?

When we observe new data trends, we can see initiatives in the open-source community or startups with commercial software, such as Airbyte, Meltano, dbt-labs, DataHub, or OpenLineage.

These examples show how the world of data processing is evolving to:

- Take advantage of new data repository enhancements and move from an ETL (extract, transform, load) to an ELT (extract, load, transform) model
- Provide capabilities to manage Data as Code
- Improve data comprehension and observability

Figure 5



These capabilities are the keys that will allow us to build data pipelines at scale while also providing business value.

The Journey From Data Preparation to Data Comprehension

Comprehension is the first step to convert data into information. The transformation layer plays a key role in this process. Usually, the transformation layer is the bottleneck because data engineers and business analysts do not talk the same language, so collaborating is complex. However, new tools such as dbt are trying to improve that.

WHAT IS DBT?

dbt (data build tool) is an open-source CLI tool for the transformation layer that enables collaboration between business data analysts and engineers during the data lifecycle using a common language as well as a software development best practices, such as version control, continuous integration/deployment, and continuous testing.

Figure 6

COMMON LANGUAGE COMPANY-WIDE

SQL is a common language throughout companies that allows all stakeholders to engage in discussion throughout the entire data lifecycle. SQL enables data and business analysts to collaborate on writing or modifying transformation through SQL statements using version control tools.

ELT

The world of data has changed a lot. Right now, most data repositories are performant and scalable. New scenarios have changed the rules of the transformation process. In many cases, data repositories are better suited for the work than external processes. dbt performs the transformation of the ELT process, taking advantage of new data repositories' features and running data transformation queries inside your data warehouse.

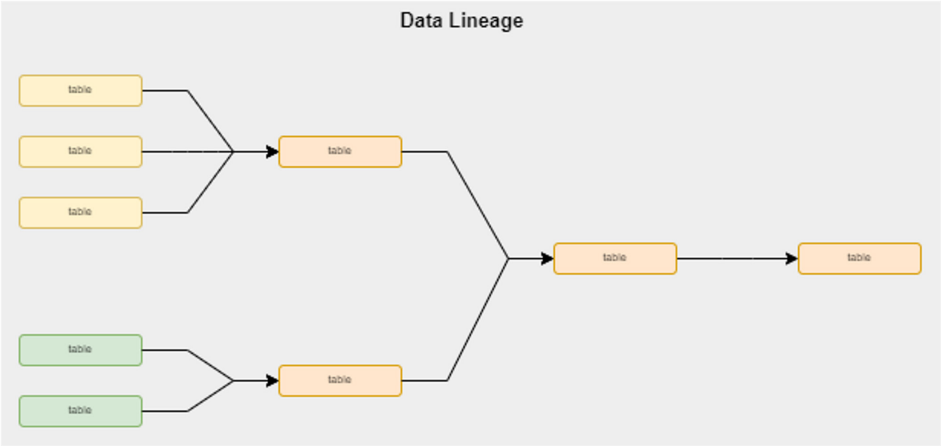
DATA AS CODE

dbt allows managing data transformation as code in your Git repository and applying continuous integration best practices. It provides testing capabilities to include unit testing modules based on SQL queries or extending them with macros to increase coverage in more complex scenarios.

LINEAGE AND METADATA

dbt allows us to generate data lineages and metadata dynamically in each run of data transformations. There are many integrations with platforms, such as DataHub or OpenLineage, that provide enterprise visibility.

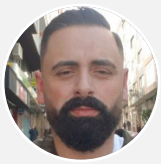
Figure 7



Conclusion

In recent years, a great deal of effort has been put into improving data processing performance and ingestion capacity, but quality and comprehension are still problematic areas that make decision-making difficult. It doesn't matter if we receive a lot of data very quickly if we don't understand it, or the data is of poor quality. Having a layer of data pipelines that is scalable, maintainable, and comprehensible is a *must* in order to provide information that adds business value.

Remember: Not making a decision is better than making a decision based on inaccurate data — especially when you are not aware that the data is wrong.



Miguel Garcia, Head of Engineering at Nextail

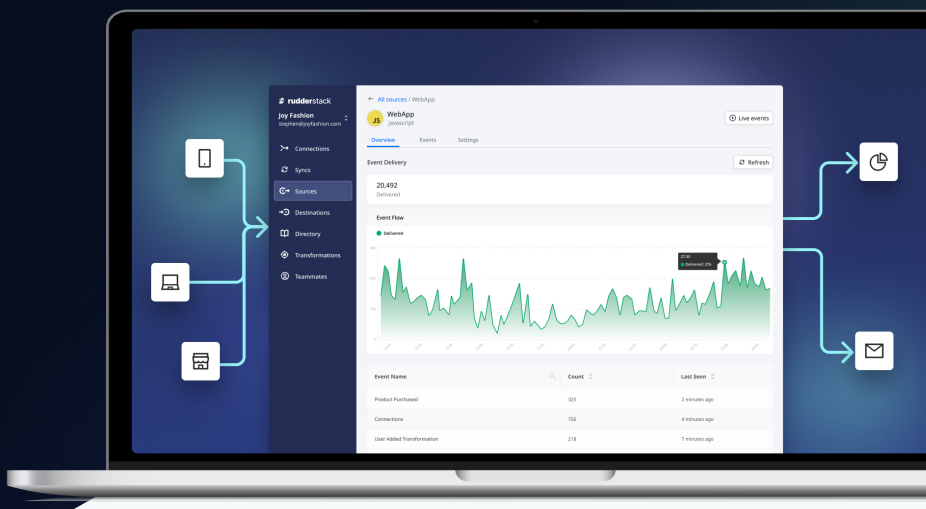
@miguelglor on DZone | @mgarlorenzo on LinkedIn | @mgarlorenzo on GitHub

With more than 10 years of experience leading data platforms and technical teams in high-performance and real-time environments, he has worked in companies like Inditex and Oracle, among others. He is a believer in work based on three fundamental principles: team, architecture, and automation. He enjoys sharing his knowledge through articles and conferences.

rudderstack™

All your customer data pipelines in one platform

Effortlessly collect customer data from any source and send it to any destination in your stack with RudderStack and your cloud data warehouse.



Kill Data Silos

With unified real-time streaming, ETL, and Reverse ETL pipelines, you can finally have complete visibility into your customer data flows.

Future-Proof Your Stack

With your warehouse at the center and RudderStack connecting your entire stack, your infrastructure can quickly adapt to changing business needs.

Build Advanced Data Products

RudderStack delivers the integrations and infrastructure, so you can focus on building data products, not plumbing.

Sign up at rudderstack.com today and send up to 5 million events per month for free.

CASE STUDY

Case Study: Joybird

How Joybird Reduced Time Spent on Data Integrations by 93 Percent

Joybird is an online-first, direct-to-consumer manufacturer of high-quality, customizable furniture inspired by mid-century modern designs. Now a La-Z-Boy subsidiary, the company operates showrooms in Brooklyn, Washington, D.C., Chicago, and Los Angeles, but the bulk of its sales go through its website.

CHALLENGE

Joybird wanted to gain a robust understanding of their customer journey and connect user actions to downstream platforms to trigger automated emails and operational actions. The company was already tracking events, but integration issues hampered their progress.

Joybird's engineers found themselves custom-coding integrations that broke whenever something on the front end or back end changed. Moreover, the data team did not have confidence in the system. They had a glut of customer data from different sources in their Snowflake warehouse, but it wasn't reliable. Moving it from the front end to the back end and shuffling it between platforms and applications yielded inconsistent data definitions and schemas.

SOLUTION

Joybird used RudderStack Event Stream and Reverse ETL data pipelines to take control of its customer data, engineering a warehouse-first approach atop event-driven architecture. This transformed its Snowflake data warehouse from a silo that hampered the free flow of customer data to a single source of truth that houses front- and back-end event data and routes it to downstream destinations. To ensure data consistency regardless of destination, the team cleaned up the website code, replacing multiple SDKs and Pixels with RudderStack's SDK, including GTM snippets and calls to random Facebook Pixels.

RESULT

Joybird's data warehouse is no longer a liability but a powerful reservoir of actionable information. "We ingest a lot of data, put it into Snowflake, and use RudderStack to get it downstream," explains Brett Trani, Director of Analytics at Joybird. "More importantly, we can guarantee that the data is clean, consistently defined across our entire stack, and compatible with all existing and future platforms."

Teams company-wide benefit from Joybird's retooled stack:

- Data Engineering reduced the time spent on building new integrations and managing data pipelines by 93 percent.
- Marketing can now spin up new campaigns in an hour vs. two weeks.
- Front End reduced the time spent implementing new event tracking dimensions from 15 percent to one percent.

JOYBIRD

COMPANY
Joybird

COMPANY SIZE
200 employees

INDUSTRY
E-Commerce

PRODUCTS USED
RudderStack Event Stream, RudderStack Reverse ETL

PRIMARY OUTCOME
Data Engineering reduced the time spent on building new integrations and managing data pipelines by 93 percent.

"With RudderStack, we activate event data streams in real time. We also send event data to Snowflake where we join it with CRM data to create richer customer profiles that we send downstream via Reverse-ETL."

— **Brett Trani**,
Director of Analytics, Joybird

CREATED IN PARTNERSHIP WITH

rudderstack

CONTRIBUTOR INSIGHTS

ETL, ELT, and Reverse ETL

Differences and Use Cases Among Three Primary Data Integration Models

By Wayne Yaddow, Data Quality Analyst/Tester & Freelance Writer

ETL (extract, transform, load) has been a standard approach to data integration for many years. But the rise of cloud computing and the need to integrate self-service data has led to the development of new methodologies such as **ELT** (extract, load, transform) and **reverse ETL**.

What are the advantages of ETL? How do these three data integration approaches differ? Is ETL preferable to ELT for your data pipeline use cases? Why and when is reverse ETL valuable for your data warehouse (DW) and a data lake? Why is reverse ETL not an optional choice for data pipeline ETL or ELT, but rather another process opportunity for a DW and data lake?

To help you choose the data integration method for your data pipeline projects, we briefly explore ETL and ELT — their strengths and weaknesses and how to exploit both technologies. We describe why ETL is an exceptional choice if you need to transform to support business logic, granular compliance on data in flight, and in the case of ETL streaming, low latency. We also explore how ELT is a better option for those who need fast data loading, minimized maintenance, and highly automated workflows.

General Concepts of ETL and ELT

A common challenge for organizations is capturing data from multiple sources in multiple formats and then moving it to one or more data targets. It is possible that the target is not the same kind of data storage as the source. If the format is different, the data must be refined or cleaned before being loaded into the final target. Many tools, services, and processes have been developed to help address these challenges. Regardless of the process used, there is a shared need to coordinate work and apply data transformations in the data pipeline.

Most data movement projects gather multiple sources of data. They need a well-defined data pipeline (i.e., ELT/ELT). But what is a data pipeline? It's the path (or workflow) that the information goes through from the origin to the end.

THE ETL PROCESS

ETL is a data integration process that enables data pipeline projects to extract data from various sources, transform the data, and load the resulting data into a target database. Regardless of whether it's ETL or ELT, the data transformation/integration process involves the following three phases (Figure 1):

1. **Extraction** – Data is extracted from source systems (e.g., SAS, online, local) using database queries or change data capture (CDC) processes. Following extraction, the data is moved into staging areas for further processing.
2. **Transformation** – Data is cleaned, processed, transformed, enriched, etc., then converted to a required format to be consumed by a target data pipeline, data warehouse, database, or data lake.
3. **Loading** – The original and converted data are loaded into the target system. This process may involve writing to a delimited file, creating schemas in a database, or overwriting existing data with cumulative or aggregated data.

SEE FIGURE 1 ON NEXT PAGE

Figure 1: The ETL process

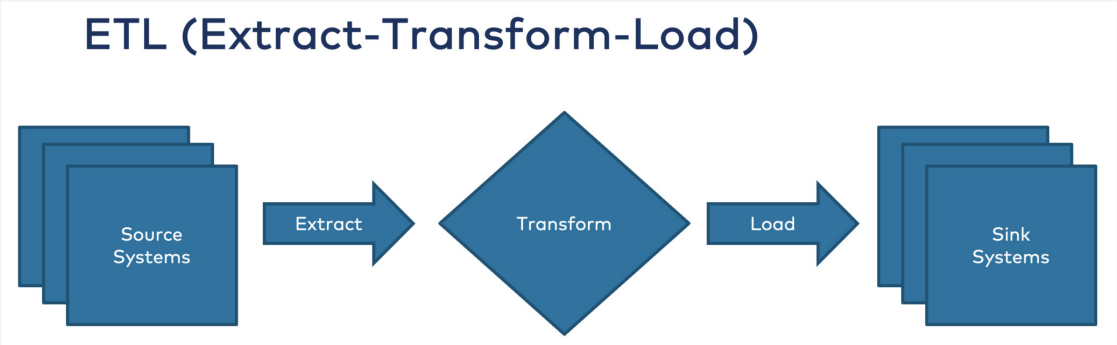


Image source: "When to Use Reverse ETL and When It Is an Anti-Pattern," Kai Waehner

ETL and ELT processes perform these steps in a different order. The data pipeline team must decide whether to transform data before or after loading it into the target data repository.

THE ELT PROCESS

ELT is a method for integrating data from across an organization to prevent data silos. The data is extracted from its source(s), loaded into a data warehouse, and then transformed on demand later. Transformations are typically applied on an as-needed basis, whereas in an ETL process, data is transformed before being stored (Figure 2).

1. **Extraction** – Same as ETL.
2. **Loading** – Unlike ETL, data is delivered directly (i.e., without cleaning, enrichment, transformations) to the target system — typically with the target schema and data type migrations factored into the process.
3. **Transformation** – The target platform where data was loaded can be transformed for business reporting purposes. Some companies utilize tools like dbt to transform their target data. Therefore, in the ELT pipeline, transformations are performed on the target data as needed.

Figure 2: The ELT process

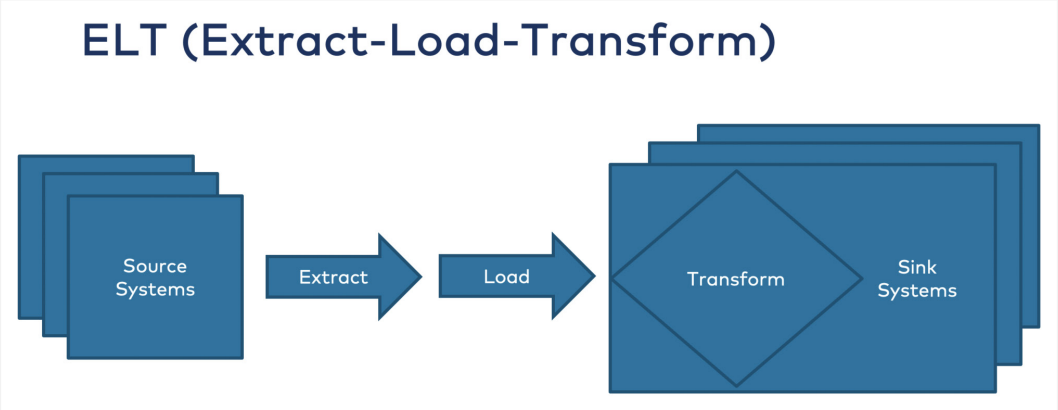


Image source: "When to Use Reverse ETL and When It Is an Anti-Pattern," Kai Waehner

ELT reorders the steps involved in the integration process, with transformations occurring at the end instead of in the middle. ELT processes can switch the order of phases to load the data into a data lake (for example) that accepts raw data, no matter its structure or format. This allows for instant data extraction and loading.

A further factor contributing to the adoption of ELT is the widespread implementation of cloud-based data warehouses. Most cloud-hosted DWs are now managed, which means enterprises don't need to purchase or manage any hardware or storage, install software, or think about scaling — the cloud provider tool manages everything. For instance, a cloud-hosted DW can be provisioned in a short period of time. Cloud data warehouse solutions offer complete separation between computing and storage and the ability to store unlimited data.

ETL vs. ELT: Attributes, Functions, and Use Cases

There is no clear front-runner in the ETL vs. ELT use case discussion. The table below outlines their distinguishing attributes:

Attributes	ETL	ELT
Best for...	Structured data, legacy systems, and relational DBs; transforming data before loading to DW	Quicker, timely data loads, structured and unstructured data, and large and growing data; transforming data as needed
Support for unstructured data	Used chiefly for on-premises relational data	Support for unstructured data is readily available
Support for data lakes	Does not support data lakes	Supports data lakes
Lookups	Fact data, as well as dimensions, must be available for the staging area	All data is available because extracts and loads occur in one single action
Time to load	Data is initially loaded into staging, then the target system	Data is loaded into a target system once
Data output	Often used for on-premises data that should be structured before uploading to a relational DW	Structured, semi-structured, and unstructured data; best suited for large data volumes to be implemented in cloud environments that offer large storage and computing power, enabling the data lake to store and transform data quickly as needed
Performance of data loads	Data load time is longer than its alternative since it's a multi-stage process	Data loading happens faster since there's no wait time for transformations, and the data is only loaded once into the target database
Performance of transformations	Data transformations can be slow	Data transformations complete faster since they are done after load and on an as-needed basis
Aggregations	Complexity increases with a greater amount and variety of data	The power of the target platform can process a significant amount of data quickly
Data deployment	On-premises or cloud-based	Often cloud-based
Analysis flexibility	Use cases and report models are well defined	Data may be added at any time as the schemas evolve; analysts can build new views of the target warehouse
Compliance	Better suited for compliance with GDPR, HIPAA, and CCPA standards; users can omit sensitive data before loading it into the target system	Carries more risk of exposing private data and not complying with GDPR, HIPAA, and CCPA standards
Implementation	Easier to implement given the wide variety of tools and support skills	Requires niche skills to implement and maintain

General Concepts of Reverse ETL

Reverse ETL is an architecture for extracting cleaned and processed data, which involves copying data from a data warehouse (or data lake/mart) to one or more operational systems. Data can be ingested back into other applications, such as Salesforce, which can be used for business operations and forecasting. Operationalizing extracted data sources enables all users to access the data and insights needed for tools they frequently use. As a modern data technology stack component, reverse ETL allows companies to conduct more complex analyses than they could with business intelligence (BI) tools alone.

Reverse ETLs have become a strategic new integration process in the modern data stack to reduce the time between analytics and data that's needed in fast-moving enterprises. Reverse ETL processes focus on activating the data within the data warehouse by syncing it back into the operational tools of business users. Users must define the data and map it to the appropriate columns/fields in the end destination.

Figure 3: The reverse ETL process

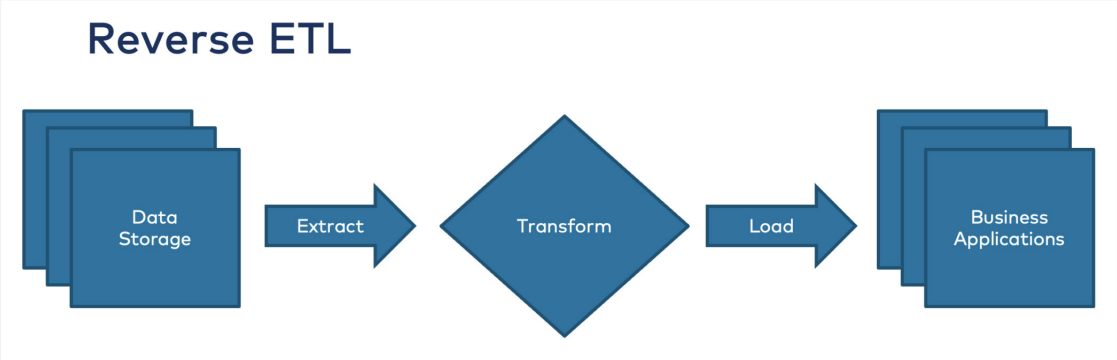


Image source: "When to Use Reverse ETL and When It Is an Anti-Pattern," Kai Waehner

Reverse ETL may be required because your data store (e.g., DW, relational DB) has become a repository that is not fully accessible to everyone who needs the data. Without a reverse ETL on your data warehouse, much of the basic data your business needs is only in the DW (Figure 3).

REVERSE ETL USE CASES

Instead of seeing the data warehouse as the end component in a data pipeline, reverse ETL users can leverage the already cleaned and prepared information available from their DW. They can do so by using connectors to read the data warehouse (e.g., SAP, SASS). The objective is to make the DW data actionable by operational systems.

Data lake teams are embracing a new set of advanced engineering analytics skills. They should be freed and allowed to use those skills and data using their modern analytic tools. With reverse ETL solutions available, modern data teams can extract data from data warehouses to power email marketing, customer support, sales, or financial models. This means that more successful business teams can make self service deep, valuable, and more efficient.

What can you accomplish with reverse ETL? Among other things:

- **Business analysis** - Quickly track and react to changes in business applications and data.
- **Business analytics** – Deliver insights to business teams for their analytical workflows, so that they can make more data-informed decisions.
- **Data infrastructure** – With an increasing number of source systems, reverse ETL is now an essential tool to quickly and efficiently operationalize data in DWs and data lakes.
- **Replicating data** for modern cloud applications – To enhance reporting capabilities and find timely information.

BUY VS. BUILD REVERSE ETL

When data teams adopt third-party reverse ETL tools, they can implement operational analytics quickly, but whether to buy or build might be difficult to determine.

Three factors should be considered before deciding to build a reverse ETL process and platform:

1. **Build data connectors** – Transferring data from your warehouse to downstream operational systems requires the complicated work of integrating API connectors. If you choose to design and build the reverse ETL and associated process, you will most likely allocate the ETL pipeline construction process to a team of developers.
2. **Prepare for long-term maintenance** – Once your development team rolls out data connectors, the work doesn't end there. Since API specifications often change, keeping connectors up to date becomes an additional burden.

3. **Design in extendibility and reliability** – Data engineers need to ensure that reverse ETL pipelines can evolve rapidly to manage data surges effortlessly as your enterprise grows. Reverse ETL pipelines must be exceedingly reliable, leaving no room for performance or data delivery issues.

Conclusions

The most burdensome and time-consuming step in creating a data pipeline is extracting data from various sources and then testing the entire process. Each source needs its synchronization, and you can have many sources for data collection. This can rapidly become a huge lift for a technical team. Ultimately, data integrations can become very complicated as they require significant expertise at all levels of the process. However, the time and effort needed to learn more about the options, variations, and products available for the job are always worthwhile.

References:

- Gartner Research (2021), "Gartner Magic Quadrant for Data Integration Tools"
- Gartner Research (2020), "Critical Capabilities for Data Integration Tools"
- Kai Waehner, "When to Use Reverse ETL and When It Is an Anti-Pattern"
- Stephen Roddewig, "ETL vs. ELT: What's the Difference & Which Is Better?"

Wayne Yaddow, Data Quality Analyst/Tester & Freelance Writer

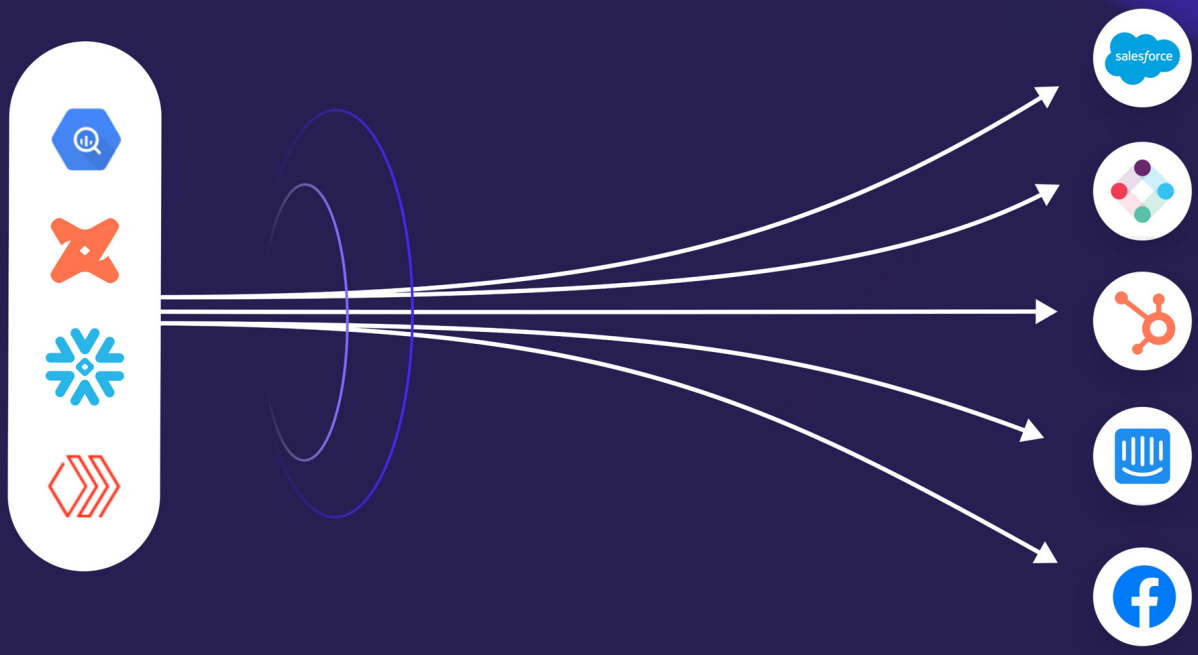
@wyaddow on DZone | @wayne-yaddow-7839624 on LinkedIn

Wayne Yaddow is a freelance writer focusing on data quality and testing issues. He spent 14 years as a consultant managing ETL data integration testing projects with J.P. Morgan Chase, Standard and Poor's, and IBM. Wayne has developed and taught data warehousing, ETL, and data integration testing courses. His other writings are found on DZone, Dataversity, TDWI, and Tricentis websites.

What's After Analytics? Activation.

Activate Your Warehouse Data with **Hightouch**

After ingesting, warehousing, and processing, the power in your data doesn't stop at analytics. Data Activation unlocks the knowledge sorted within your data warehouse and makes it actionable for your business users directly within the tools they use every day.



"Data Activation is the biggest value-add that we do. Hightouch enables us to take our insights and feed them back to our key stakeholders so that they can be informed about the business."

blend

"Data Activation is really important to us. Not everyone is doing their work in a BI tool—some of our employees are doing it in a CRM, and that is why Hightouch is valuable."

vendr

Hightouch supports more than 100 integrations all with SQL—no scripts or APIs required.

Sign up for Free at Hightouch.io

SPONSOR OPINION

Why Modern Data Teams Need Data Activation

By Pedram Navid, Head of Data at Hightouch

How many times has your data team delivered reports that nobody reads or uses? While today's modern data stack can easily deliver on insights and reports, data teams still struggle to show the business impact of their work. Imagine your CEO or CTO being able to draw a direct line between data teams and revenue — or your marketing and sales teams finally being able to self-service *action* on said insights right from the tools they use every day. All you need to do is **activate your data**.

Today, organizations are all facing the same last-mile problem: The data in the warehouse isn't accessible or usable to people who don't know how to write SQL — i.e., key stakeholders and business decision-makers — therefore, leaving the true business impact of the data team unrealized.

Data activation utilizes **reverse ETL** to put the power of warehouse data directly into the hands of sales, marketing, customer success, and finance by keeping them inside the tools they already love — but enriched with the data they need to optimize their work.

Data activation unlocks the knowledge sorted within your data warehouse and makes it actionable for business users within the tools they use every day, like Salesforce, Hubspot, and Slack.

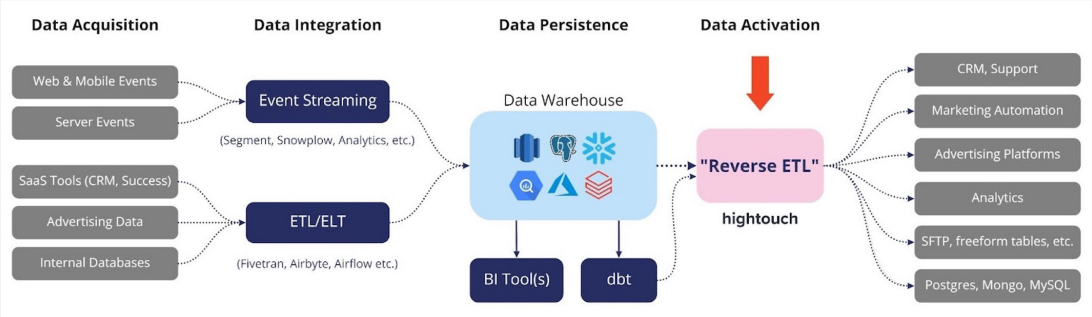
Data Activation Centers Data Teams Across the Org

There are many ideas about how a data team's work should be measured, but they've all either focused on activities over outcomes (the tables, reports, and models they created) or on the immeasurable (how quickly they enable others to make decisions). What if, instead, data teams could be measured by their contributions to tangible outcomes?

Data activation means:

- You no longer have to write data integrations yourself
- All teams are aligned with business results
- Drastically reduced time to implement or make changes to delivering data to destinations
- Data teams aren't dependent on engineering resources to get their work done

In this way, data activation is the next piece of the modern data stack puzzle, because it's what comes *after* data analytics.



The new map of the modern data stack with data activation

What if your data teams could cut the time to ingest data into ad campaign platforms with automation for faster optimization? What if your data team could improve conversions and close rates for your sales team by prioritizing incoming leads by syncing lead scores directly within the CRM?

It's time to stop asking "What if...?" Because now you really can: with data activation. Getting started with Hightouch is fast, easy, and your first destination is free.

CONTRIBUTOR INSIGHTS

Modernizing Testing With Data Pipelines

By Eric Schabell, Portfolio Architect Technical Director

Organizations today are buried in data. They collect data from a vast array of sources and are attempting to find ways to leverage this data to advance their business goals. One way of approaching this is to use data pipelines as a way of connecting to data sources and transforming the date through a pipeline into some form that is usable at the end point.

While this is part of the ongoing struggle to operationalize data for an organization, there is always a continuous need to find ways to provide good datasets for testing. Organizations need these datasets for testing both applications and systems throughout their architecture landscape. They also have needs for datasets to focus on testing aspects of their organization such as security and quality assurance.

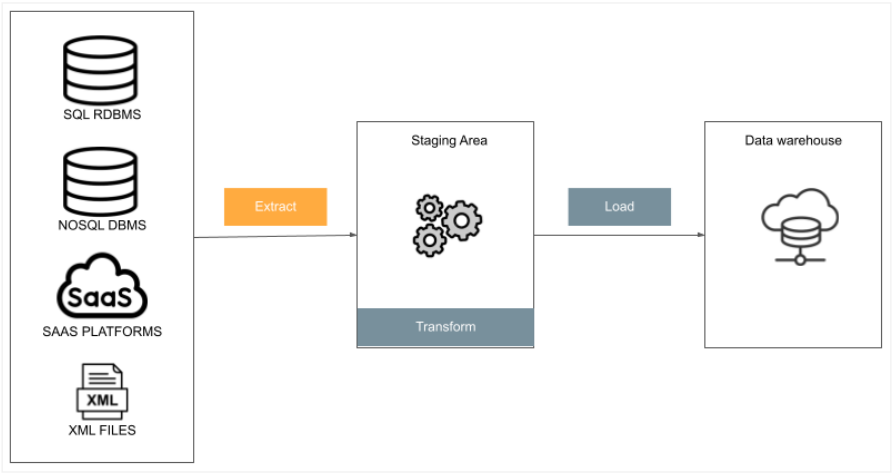
There is a very real need for creating synthetic data. What this really means, in simple terms, is that we need to find a way to create fictitious or fake data. We want to create consistent data resembling real world needs for testing systems. Let's take a look at data pipelines and explore how this can be used to start creating your own synthetic data for testing in your organization.

Data Pipelines and Testing

A very simple definition of a data pipeline is: a set of data processing elements connected in series, where the output of one element is the input of the next one.[1] More simply put, these are the fundamental connections used to bring data from a data source, back to the level where it can be analyzed, transformed, and then used by an organization. Data pipelines start by retrieving data. They can pull the needed data from within a platform, such as SQL (DB) data sources, through a programmable interface such as an application programming interfaces (APIs), or through data streaming and event processing interfaces.

Once the data has been retrieved, a decision can be made to transform the data for end user needs. This can be done with data generation APIs, data building by cleaning or changing the structure of the data retrieved, and finally the data can be anonymized for security reasons before being presented to the end users. These are but a few examples of what a data pipeline can be used for as part of the testing process, and the figure below[2] is a simple example of a data pipeline from sources to a final data warehouse location for further use.

Figure 1



[1] Pipeline (computing): https://en.wikipedia.org/wiki/Pipeline_(computing)
[2] Image adapted from: https://www.qlik.com/us/data-integration/data-pipeline

Testing requires us to provide a dataset to a system, application, or piece of code that is being tested. This set of data can be created manually, copied from an existing set of data, or generated for use by the testing team.

Manually creating test data can be useful when dealing with very small datasets, but it becomes very cumbersome when the need for huge datasets arises. Copying datasets from existing (production to test) environments comes with security and privacy issues if data contains sensitive elements. Generating data based on existing data can provide a good outcome.

What if we want to generate data at scale, with security in mind to provide for anonymized results, and ensure flexibility with regards to what is being generated? This is where data synthesis comes in. It allows you to generate data with the flexibility you might need.

Data Synthesis for Beginners

Generating synthetic data can provide massive amounts of data while addressing sensitive data elements. Synthetic data can be based on critical data dimensions like names, address, phone numbers, account numbers, social security, credit cards, identifiers, driver's license numbers, and more.

Synthetic data is defined as fake or created data, but it's often based on real data that is used to expand to create larger, realistic datasets for testing. The data generated for testing is then made available across the organization for business users and developer needs in a secure and scalable manner.

There is a vast array of uses for this synthetic data across any organization, such as healthcare, finance, manufacturing, and any other domains that are adopting new technologies for various business needs. Its direct usage is in ongoing testing, security, and quality assurance practices to help with implementation, application development, integration, and data science efforts.

Not only are you able to provide datasets at scale with data synthesis, but it also ensures support for data consistency across multiple domains in an organization while providing viable data representing real-world formats. It provides a consistent approach across any organization for developers, architects, and data architects to leverage data for testing.

Getting Started With Data Synthesis

The best way to discover what data synthesis can do for you is to explore the most common usage patterns and then dive into an open-source project to jump-start your experience. There are two simple patterns for getting started with data synthesis: in cloud-native environments and with cloud-native APIs, as shown in the figure below:

Figure 2

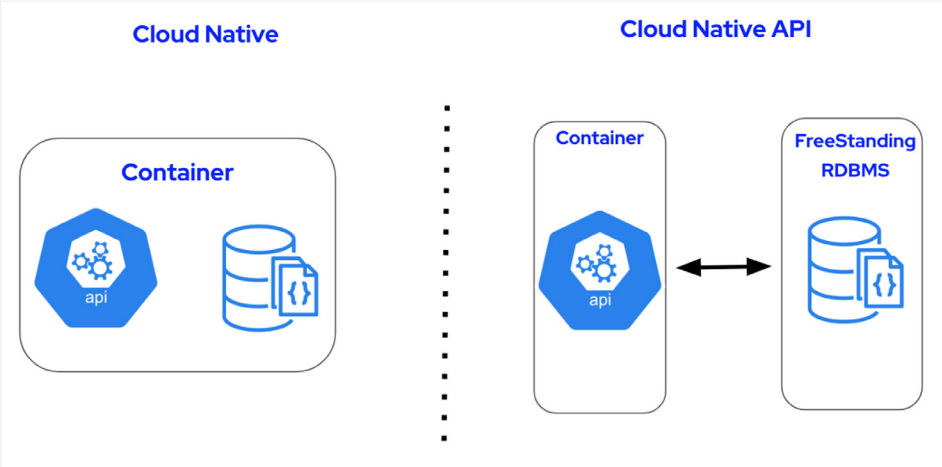


Image source: self-designed diagram

The first pattern is where you run the data synthesis platform within a single container on a cloud platform of your choice and leverage the API to pull data from sources in the container, such as applications or databases. The second is where you can deploy the data synthesis platform on a cloud platform of your choice and leverage cloud-native APIs to pull data from any source, such as external free-standing data sources.

Data synthesis shines in the following use cases:

- Retrieving needed data within the platform (SQL)
- Data retrieval (APIs)
- Data generation (APIs)
  - Building more fictitious data on demand or scheduled
- Data building
  - Building more structured data based on fictitious data on-demand or scheduled
  - Creating structured or unstructured data that meets your needs
- Streaming industry-centric data
  - Using the data pipeline to process various industry-standard data
  - De-identification and anonymization by parsing and populating from real-time systems to provide real-world attributes

Covering each of these use cases goes beyond the scope of this article, but this list gives the reader a good idea of the fields of applicability for data synthesis and testing. For an overview of the data synthesis data tier, we present the following figure:

Figure 3

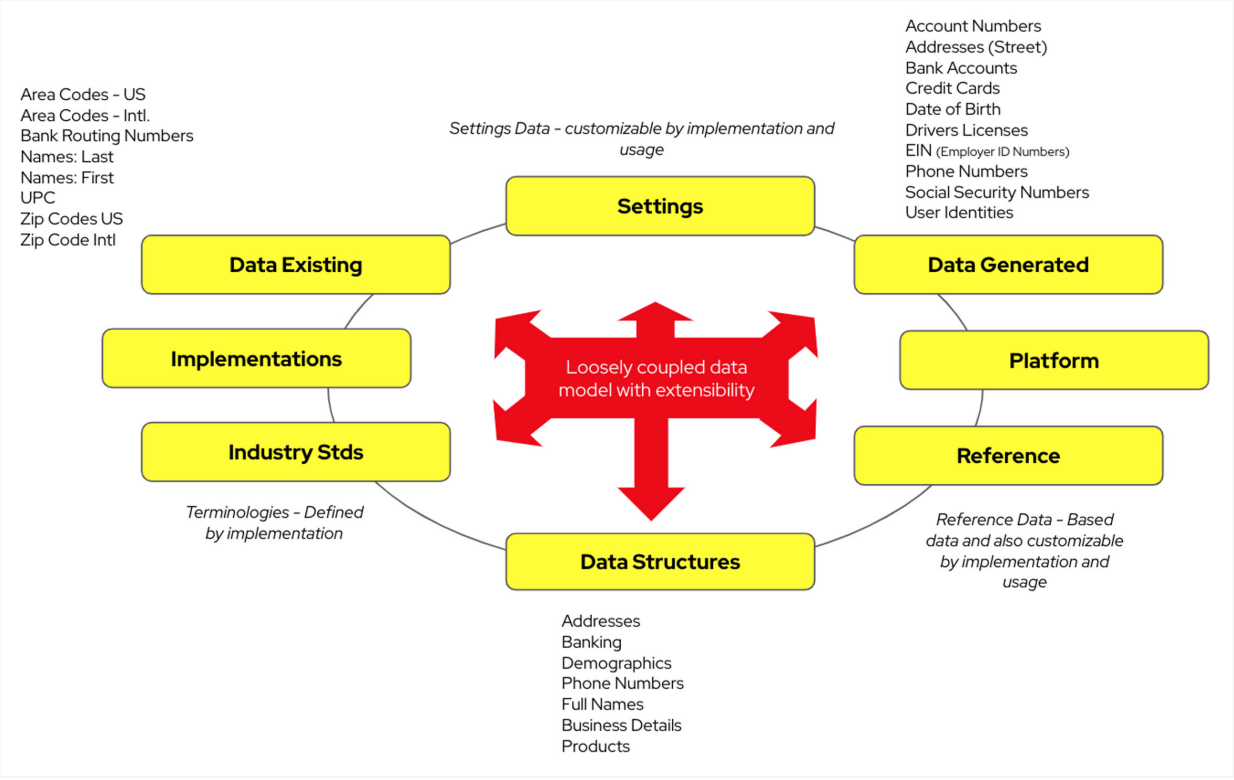


Image source: self-designed diagram

This is an overview of the data tier and how the platform ties it all together using United States location data fields (zip codes and area codes) as an example. In the center, you can see your loosely coupled data models that can be extended as needed. They form the core basis for accessing from existing data, implementation data, and industry standard data. This can be adjusted using data settings and based on existing data structures in your organization. The output is the generated data, reference data, and platform-specific data.

After this short tour of data synthesis, the next step is to start exploring the open-source project known as Project Herophilus,[3] where you can get started with the data synthesis platform.

[3] Project Herophilus: https://github.com/Project-Herophilus/DataSynthesis

You will find the key beginning areas for data synthesis:

- Data tier – Designed to be extensible and support all the needs for the platform.
- Data tier APIs – Supporting the needs of user requests are the data tier APIs; this API set is about being able to both generate data and persist it to the data tier.
- Web UI(s) – Intended to be minimum viable usable products that can be used to look at the data synthesis data tier you implement.

The three modules in the data synthesis project should get you well on your way as a quick start to developing your testing datasets.

Conclusion

As organizations collect, explore, transform, and try to leverage their data, testing is a constantly growing challenge. While generating testing datasets can solve some of these issues, it often fails when the process needs to scale. Data synthesis, together with data pipelines, can offer a scalable solution to create consistent data resembling real-world needs for testing systems. You can get started by exploring the open-source project known as Herophilus, which offers three modules to kick-start your first data synthesis project.

Eric D. Schabell, Portfolio Architect Technical Director

@eschabell on DZone | @ericschabell on Twitter | @ericschabell on LinkedIn

Eric is Red Hat's portfolio architect director. He's renowned in the development community as a speaker, lecturer, author, and baseball expert. His current role allows him to share his deep expertise of Red Hat's open-source technologies and cloud computing. He brings a unique perspective to the stage with a professional life dedicated to sharing his deep expertise of open-source technologies and organizations. Follow on https://www.schabell.org.

CONTRIBUTOR INSIGHTS

Data Security Considerations in Cloud Data Warehouses

By Ben Herzberg, Chief Scientist at Satori

Cloud data warehouses (CDWs) are making rapid growth in the way organizations are analyzing data at scale. As cloud storage is elastic and cheap, and as modern data pipelines simplify ETL processes, they commonly scale to store more data than on-premises data warehouses. This obviously includes sensitive data, leading to challenges in security, privacy, governance, and compliance. In this article, we will discuss the move to CDWs and security considerations when using them.

The Move to Cloud Data Warehouses

Over a decade ago, public cloud companies started releasing data warehouses in a Platform-as-a-Service (PaaS) model. Significantly, Google BigQuery in 2010 and Amazon Redshift in 2012 enabled organizations to deploy a CDW in minutes, without the need to install the databases or configure the servers. This was followed by the launch and gradual growth of other vendors, later becoming the largest software IPO with its data cloud.

The move to a CDW (which is considered part of the modern data stack) has significant implications for the ease for data consumers and producers to access data. Cloud computing allows elasticity, storage becomes cheaper and easier to increase (or decrease), and database administration becomes much simpler (for example, in some of the CDW platforms, there are no indexes that DBAs need to maintain).

Considerations in Cloud Data Warehouses Security

When organizations use CDWs, the basics of information security remain the same, but some things are different versus having an on-premises data warehouse, or even a data warehouse you manually install on a cloud infrastructure. Part of the difference is because of the shared responsibility model. Your provider assumes full responsibility for certain things, such as physical security, operating system security and patching, and even maintaining the database software. This leaves you narrower areas of focus.

However, in many cases, the move to the cloud also allows more users to access and make value from your data (in what's referred to as "data democratization"). The move from specific teams using data to many teams accessing it and having data that keeps changing brings more challenges.

In addition, some assumptions of on-premises data storage (including that it's disconnected from the internet by default) are not always correct in cloud data storage. The complexity in applying data security to CDWs is also in the people managing them in organizations. These are mostly data engineers and not security professionals, and in many companies, this creates a situation where the CDW is a "black box" for the security teams who don't have full control over the security policies.

Let's discuss some of the main security considerations.

Encryption

Organizations must make sure that both stored data (at rest) and data being connected to (in transit) are encrypted. This is important from a security point of view (reducing risks, such as MITM attacks, and access to the data stored). It is also important when facing compliance audits. In some CDW platforms, data is encrypted by default at rest and in transit. In other platforms, you need to configure it to encrypt stored data and enforce only encrypted protocols when accessing the data.

Access Control in Cloud Data Warehouses

A large part of meeting security — as well as compliance and privacy — requirements is to establish an effective access control to the data you're storing. This depends a lot on your company, the security policies you have, the types of data, etc. However, let's discuss some of the main aspects of access control in CDW.

NETWORK ACCESS CONTROL

Setting network access policies are, in most cases, a simple, effective way to reduce the risk level of your CDW. Some platforms come without public internet access by default, and in some platforms, you need to configure network policies. In some of the cases, you will also want to set more specific network access policies for specific users or groups of users. Using Snowflake as an example, here is how to set a network access policy and apply it to a specific user:

```
CREATE OR REPLACE NETWORK POLICY us_employees
   ALLOWED_IP_LIST = ( '1.1.1.0/24', '2.2.2.0/24', '3.3.4.5' )
   BLOCKED_IP_LIST = ( '1.1.1.128', '2.2.2.128' )
   COMMENT = 'US employees offices, excluding guest WiFi gateways';

/* Assigning the policy to a user */
ALTER user us_marketing_analysts SET NETWORK_POLICY=US_EMPLOYEES;
```

AUTHENTICATION

Authentication of users to CDWs differs by platform. For example, not all of them support OAuth for individual user authentication in BI tools. Furthermore, it is quite common not to use the most secure authentication options available. In many companies, for example, applications connect to the data warehouse with a username and password when they could use stronger key-based authentication.

This usually needs to be addressed in collaboration between the data and security teams (and sometimes, the IT identity teams as well) to make sure that there are clear security guidelines as to which authentication type to use. A good example is, when possible, to use Identity Provider integration to make human users abide by the organization's authentication policy (including using two-factor authentication).

AUTHORIZATION

Perhaps the hardest part of data warehouse security is the authorization (i.e., once users are authenticated to use the data warehouse, what data can they access and at what level?). Different CDWs have different authorization mechanisms. For example, Snowflake has a strict role-based access control model (RBAC), and Amazon Redshift recently introduced an RBAC model of its own.

Here are some of the common security challenges in CDW authorization:

- Many users require changes in their data access, which, in many cases, has to be done by data engineering teams, creating a bottleneck.
- There is often no good process to revoke access to data that is no longer needed by users.
- Tracking access to sensitive data by users (which is often needed for compliance and security reasons) is hard to apply.
- Users often get overly broad access rights.
- After a while, without a clear access design, access permissions may become complicated and harder to manage.

There are solutions to these problems, such as enabling self-service data access, creating and enforcing clear security policies, and applying security access policies in a separate environment than the data warehouse itself.

FINE-GRAINED ACCESS CONTROL

In addition to managing data access to "coarse" objects such as tables, views, schemas, or databases, in many cases, there is also a need to apply fine-grained security. This may mean that you want certain users to be able to access only specific rows from the tables (e.g., row-level security), or perform dynamic masking on data based on the user (or their role). In these capabilities, different CDWs have different capabilities. In some cases, you may need to engineer your way to such policies by using functions and views, and on other platforms, you may need to create policies and apply them to data objects.

Note that such policies have different implementations in each of these platforms. In addition, it is often hard to manage these capabilities at scale. Oftentimes, when this is done at scale, the company either builds an in-house overlay to automate access control or uses a data access solution.

Auditing and Monitoring

Auditing and monitoring your data access is an intrinsic part of your security — and a compliance requirement. Once again, different CDWs offer different levels of audit logs, as well as different steps needed to enable them. For example, in Snowflake, data access logs are available out-of-the-box of the `snowflake.account_usage` schema (available using SQL select queries), while with Amazon Redshift, you need to configure query logs export to S3 buckets.

Prioritizing and Protecting Sensitive Data

Ensuring data security for CDWs requires resources and collaboration between data and security teams, which are already very busy. In many cases, one of the most important steps is knowing where your sensitive data is, and prioritizing resources towards its security as a top priority item. These are a few examples of how different security controls are done in different CDWs. Obviously, this is a summary only, and we recommend checking for the specific requirements you require on your own:

	Amazon Redshift	Snowflake	Azure Synapse	Google BigQuery
Network access control	As part of the AWS account network policies	Defined using SQL	As part of the Synapse workspace configuration in Azure	Defined in the G-Suite admin panel
Access control	Configured using SQL, authorization to users, groups, or roles	Configured using SQL, authorization using roles only (strict RBAC)	Configured using both Azure (for adding users), and SQL (for granting access to specific objects), to users or roles	Configured using GCP UI/ API to give access to users or groups
Fine-grained security	Column-level access as part of the platform	Dynamic masking and row-level policies as part of the platform	Dynamic masking, column-level, and row-level policies as part of the platform	Column-level and row-level policies as part of the platform
Auditing and monitoring	Required configuration to export audit logs to S3	Automatic access and query logs in a virtual DB schema (`snowflake.account_usage`)	Required configuration to enable audit to Azure Storage	Automatic access and query logs as part of GCP (can be pulled using REST API)

Conclusion

CDW security — and securing access to data in an age where more users are accessing more data that keeps on changing — is challenging. On the other hand, some areas of focus are taken out of the equation, allowing data teams to concentrate on increasing data-driven value in the company. All modern CDWs allow companies to manage data in a secure way. When evaluating CDWs, it's important to bring security teams into the discussion and understand how the different security capabilities will come into play. In addition, some of the security elements may be handled outside of the CDW itself (in BI tools or data access platforms).

Regardless of the CDW you choose (which has many other factors to consider, not only security capabilities), by having clear security policies, a collaboration between data and security teams, and a plan to reduce risk continuously, a company can set itself for success in its CDW security.

Ben Herzberg, Chief Scientist at Satori

@benherzberg on DZone | @KernelXSS on Twitter | @sysadmin on LinkedIn | satoricyber.com

Ben is an experienced tech leader and book author with a background in endpoint security, analytics, and application and data security. Ben filled roles such as the CTO of Cynet and Director of Threat Research at Imperva. Ben is the Chief Scientist for Satori, the DataSecOps platform.

CONTRIBUTOR INSIGHTS

Data Pipelines for Engineered Decision Intelligence

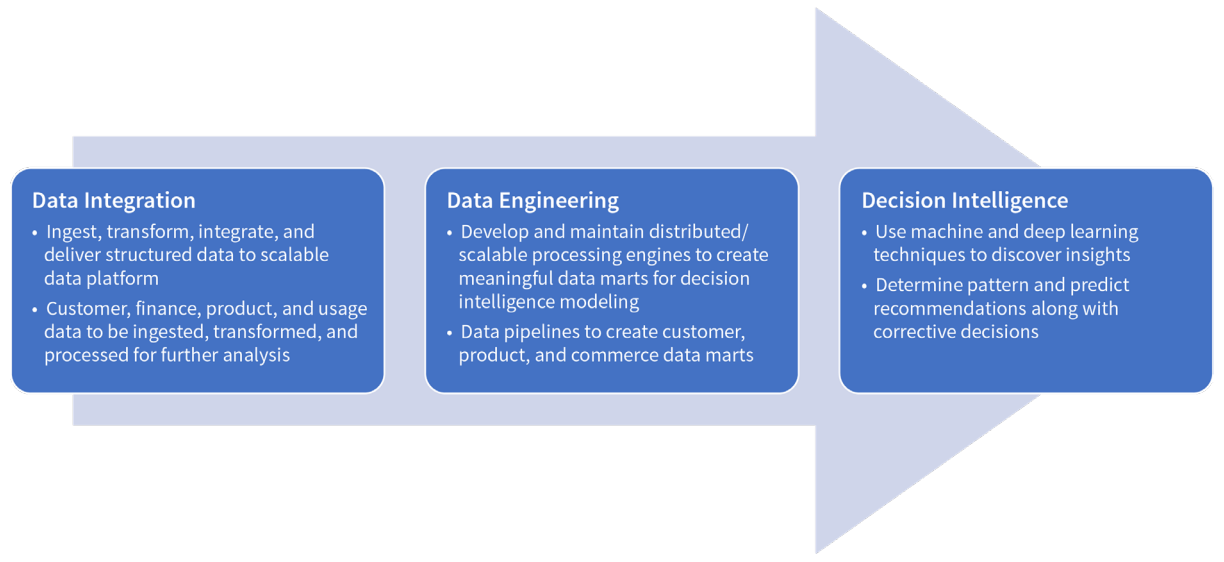
By Dr. Tuhin Chattopadhyay, Founder & CEO of Tuhin AI Advisory

Data science has reached its peak through automation. All the phases of a data science project — like data cleaning, model development, model comparison, model validation, and deployment — are fully automated and can be executed in minutes, which earlier would have taken months. Machine learning (ML) continuously works to tweak the model to improve predictions. It's extremely critical to set up the right data pipeline to have a continuous flow of new data for all your data science, artificial intelligence (AI), ML, and decision intelligence projects. Decision intelligence (DI) is the next major data-driven decision-making technique for disruptive innovation after data science. It is:

- **Futuristic** – Models ML outcomes to predict social, environmental, and business impact.
- **Holistic** – Meaningfully integrates both managerial and behavioral perspectives.
- **Realistic** – Models all contextual variables and real-life constraints.

So it's more important for DI projects to have a robust data pipeline. They need a continuous inflow of the right data with the right velocity to get stored in the right container and subsequently processed correctly for model development to generate actionable insights.

Figure 1: Enterprise decision intelligence

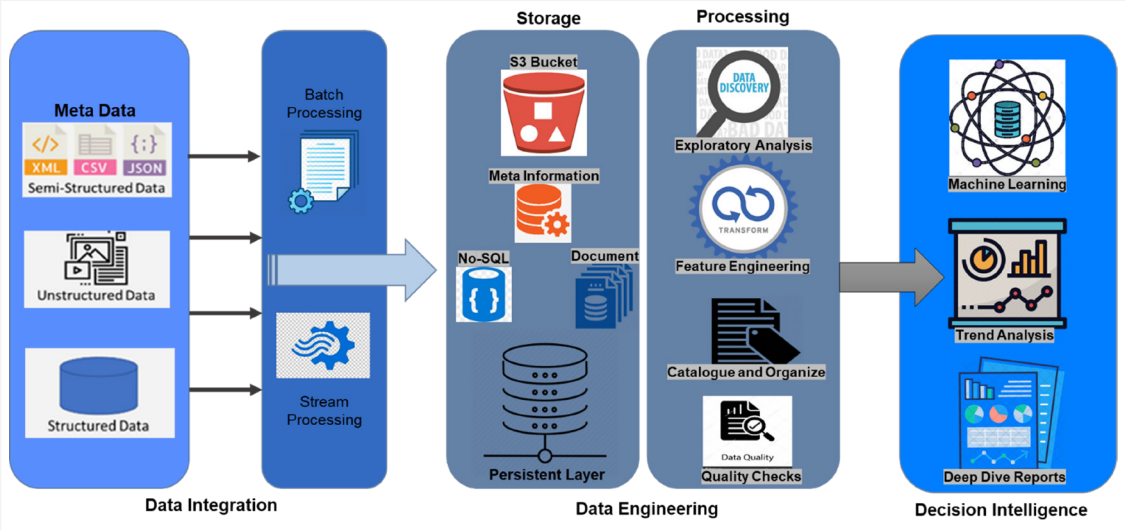


Developing a Data Pipeline

The first phase of developing a data pipeline is **data integration**: ingests various customer, product, and usage data for processing and analysis. There are two stages of data integration. The most fundamental step is to identify the right sources of both internal data — comprising IoT, CRM, ERP, OLAP, Excel reports, etc. — and external data, like Facebook, Twitter, and statistical databases. The second step of data integration is to gather semi-structured, unstructured, and structured data through batch processing and stream processing.

After obtaining the data through integration, the next phase is **data engineering**, which involves storing and processing data for further model development. The objects comprising the files and metadata can be uploaded in any container that can store diverse unstructured, hierarchical, and structured data. Processing stored data includes data sanitization, feature engineering, and splitting the data for training and testing before sending it for model development through ML, deep learning, and natural language processing techniques.

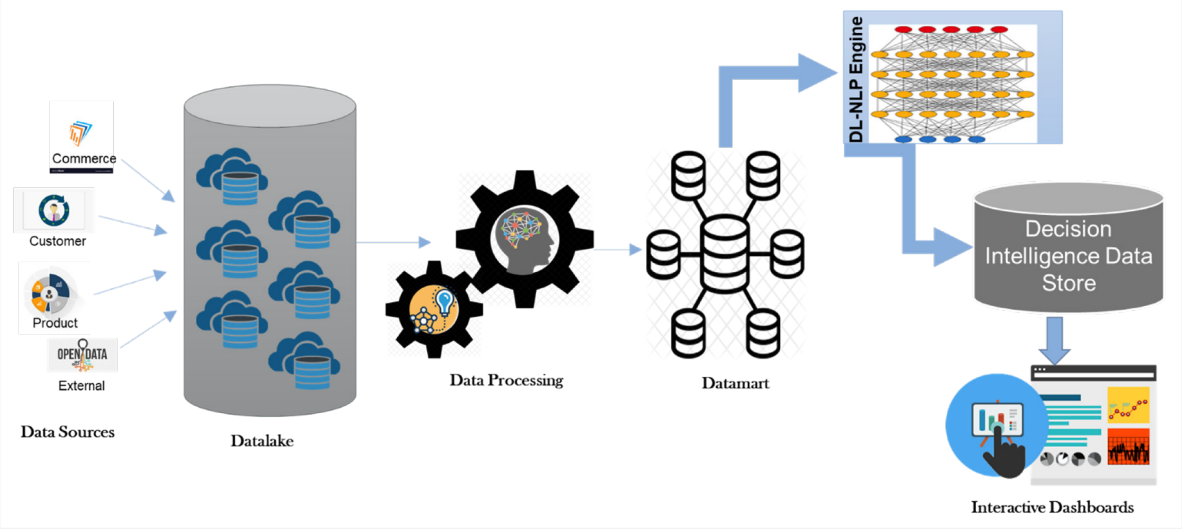
Figure 2: Data engineering framework for decision intelligence



In the last stage, data is sent for **decision intelligence** model development. Some of the most popular modeling techniques for DI include decision modeling and simulation, optimization and game theory, system dynamics and systems modeling, sensitivity and scenario analyses, knowledge management, hidden Markov models, and Markov Chain Monte Carlo. Advanced modeling techniques like quantum Bayesian networks (QBNs) with directed acyclic graphs, data-driven predictions of the Lorenz attractor, and intelligence augmentation work on top of ML outcomes to figure out the decision impact on society, business, and environment. Last but not least, the final outcome can be presented through interactive dashboards that can easily be used for managerial decisions.

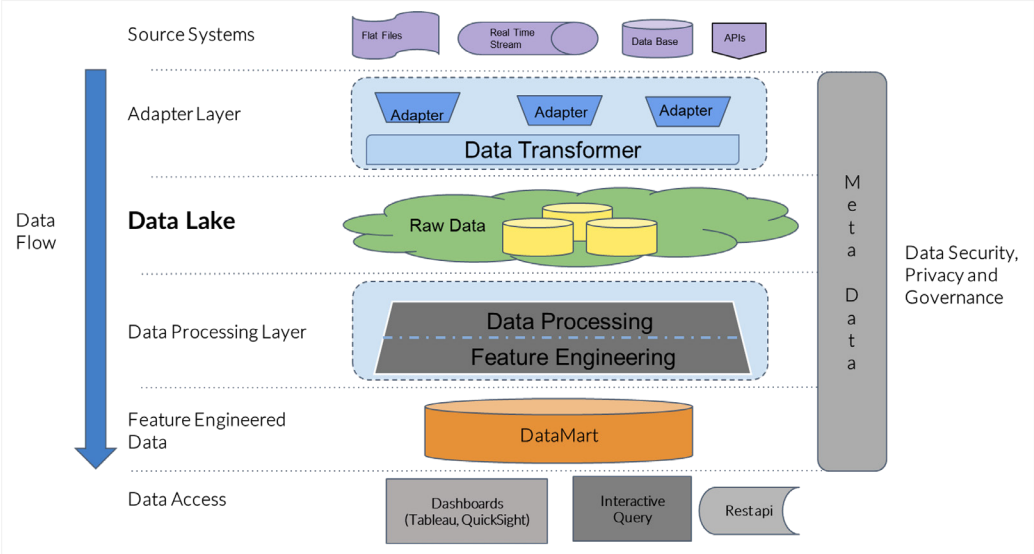
The data architecture is critical in setting up the data pipeline for DI. Traditionally, structured data is stored in a data warehouse for data discovery and querying. With the advent of semi-structured clickstream data, a data lake became the natural choice to hold vast amounts of raw data. A data lakehouse is a hybrid approach, in which a warehouse layer resides on the top of a data lake to store both structured and unstructured data. After processing the data, the feature-engineered data gets stored in a data mart before it finally flows to the DI engine for model development.

Figure 3: Data architecture



Below is an instance of platform architecture that demonstrates an increased level of abstraction for applied solution development and usage scenarios.

Figure 4: Data platform architecture



To illustrate this architecture in the cloud, Figure 5 represents a data pipeline over an AWS ecosystem, which shows the applicable AWS components in every step from data extraction to dashboarding. The dashboarding can be accomplished through any non-AWS tools like Tableau.

Figure 5: Key data engineering components in AWS

Considerations for Data Quality, Governance, Privacy and Security

While setting up the data pipeline for any DI project, it's critical to prioritize data quality, data governance, data privacy, and security issues.

DATA QUALITY

Reliable and consistent input data is critical for eliminating errors and biases. Therefore, impeccable quality of the data is sacrosanct for any DI project.

The following checkpoints can be used to measure data quality:

- **Completeness** – Is all necessary data available and accessible?
- **Consistency** – How consistent is the data across different systems that hold instances of the data?
- **Validity** – Measures whether a value conforms to a pre-defined standard.
- **Accuracy** – How correctly and accurately is the data presented?
- **Uniqueness** – A discrete measure of duplication of identified data items.
- **Timeliness** – A measure of time between when data is expected against when data is made available.

DATA GOVERNANCE

The cornerstone of any successful DI project is collaboration in correctly framing the problem and estimating the implications of actionable insights. A data governance framework enables this by assigning responsibilities to people, processes, contributors, and technology that make decision-making easier. As far as the command and control for data governance is concerned, the framework designates a few employees as data stewards. Their responsibilities include determining answers to the following questions:

- **Where** is the data?
- Who should **access** it?
- What does the data **contain**?
- What is the data **quality**?
- How can the data adhere to **compliance**?
- How **secure** is the data?

DATA PRIVACY AND SECURITY

Data privacy is critical for any DI project, and businesses must ensure compliance with all relevant data protection laws and regulations, such as PDP and GDPR (intended to protect security, privacy, PII, etc.). Privacy should also be guaranteed across data collection, processing, sharing, and deletion. Data security issues can be addressed through:

- **Authorization** – Implement measures to prevent any unauthorized access by third parties.
- **Encryption** – Encrypt data at flight and at rest, masked for PII.
- **Penalty** – Adopt and enforce huge penalties for breaches of data security.

Parting Thoughts

As we move forward, decision intelligence will connect the ML outcomes of data science projects with businesses at first, and then with society at large. Data integration and data engineering are the key components of an enterprise DI project. Both the data lake and data lakehouse have become the industry's natural choice as they can store semi-structured and unstructured data that is easy to retrieve and model. Besides the traditional ML model, many sophisticated optimization techniques are used for developing DI models. Cloud-native computing seamlessly drives the entire operations from data integration and model development to interactive dashboards for visualization and decision-making.

Ultimately, the key to a successful decision intelligence project boils down to ensuring data quality, governance, privacy, and security remain priorities at every step of the process.

Dr. Tuhin Chattopadhyay, Founder & CEO of Tuhin AI Advisory

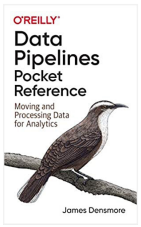
@tuhinc on DZone | @tuhinai on LinkedIn | tuhin.ai

Tuhin spent the first 10 years of his career in academia and research, teaching business statistics, analytics, and technology at several reputed B-Schools of India. As a corporate practitioner, Tuhin has a proven record of accomplishment as a transformational leader in organizations like The Nielsen Company. Currently, he runs his own consultancy for providing a full suite of AI, analytics, CTO/CAO-as-a-Service, and digital transformation services to clients.

ADDITIONAL RESOURCES

Diving Deeper Into Data Pipelines

BOOKS



Data Pipelines Pocket Reference

By James Densmore

Do you want data pipelines wherever you go? Look no further than the "Data Pipelines Pocket Reference: Moving and Processing Data for Analytics."

Author, James Densmore, explores the most critical considerations and decisions to be made around data processing and implementing pipelines.

Architecting Modern Data Platforms

By Jan Kunigk, Ian Buss, Paul Wilkinson, and Lars George

Take your data skills to the next level. In the book, "Architecting Modern Data Platforms," the authors guide readers through important concepts for building enterprise-scale data platforms.

This book also explores Apache Hadoop and how to set up applications in the cloud to ensure performance, security, and high availability.

TREND REPORTS

Data Persistence

At the core of every modern application is an unending, diverse stream of data and with it comes a fundamental need for better scalability, speed, performance, and security. This Trend Report examines effective tools and strategies for data storage and persistence, as well as topics like managing global data in microservice polyglot persistence scenarios and how to use tree structures for relational database management systems.

Readers will also find observations and analyses of survey results from our research — plus an interview with industry leaders Jenny Tsai-Smith and Tirthankar Lahiri.

REFCARDS

Multi-Cloud Data Layer Essentials

A multi-cloud data layer serves to handle all cloud workloads across all locations and regions seamlessly — it delivers functionality and data services for both existing applications and new cloud-native apps and microservices. This Refcard covers multi-cloud data layer capabilities, workloads, and considerations for various use cases.

Data Pipeline Essentials

Data pipelines allow organizations to automate information extraction from distributed sources while consolidating data into high-performance storage for centralized access. In this Refcard, you will explore the fundamentals of data pipelines, including common pipeline processes and core components, deployment considerations, implementation challenges, and advanced strategies.

PODCASTS

#idataengineer

Coming to you from the Pipeline Data Engineering Academy, listen to interviews with data engineers, hear their stories, and learn why they chose this profession and where they see it going.

Pipeline Conversations

Hear discussions with industry leaders about the latest developments in areas including open-source MLOps, the modern data stack, trustworthy machine learning, AI, deep learning, and many others.

The Data Engineering Podcast

Engineers and industry experts discuss all things data management. Listen to episodes on topics like building data lake platforms, hear insights about careers in data, learn about the role of a data engineer, and more!

INTRODUCING THE

Big Data Zone

The Big Data Zone is a prime resource and community for big data professionals of all types. We're on top of all the best and latest tips and news for Hadoop, R, and data visualization technologies. Not only that, but we also share advice from data science experts on how to understand and present that data.

Keep a pulse on the industry with topics such as:

- Data visualization
- Stream processing
- Analytics dashboards
- Data warehousing

VISIT THE ZONE

TUTORIALS

CASE STUDIES

BEST PRACTICES

CODE SNIPPETS