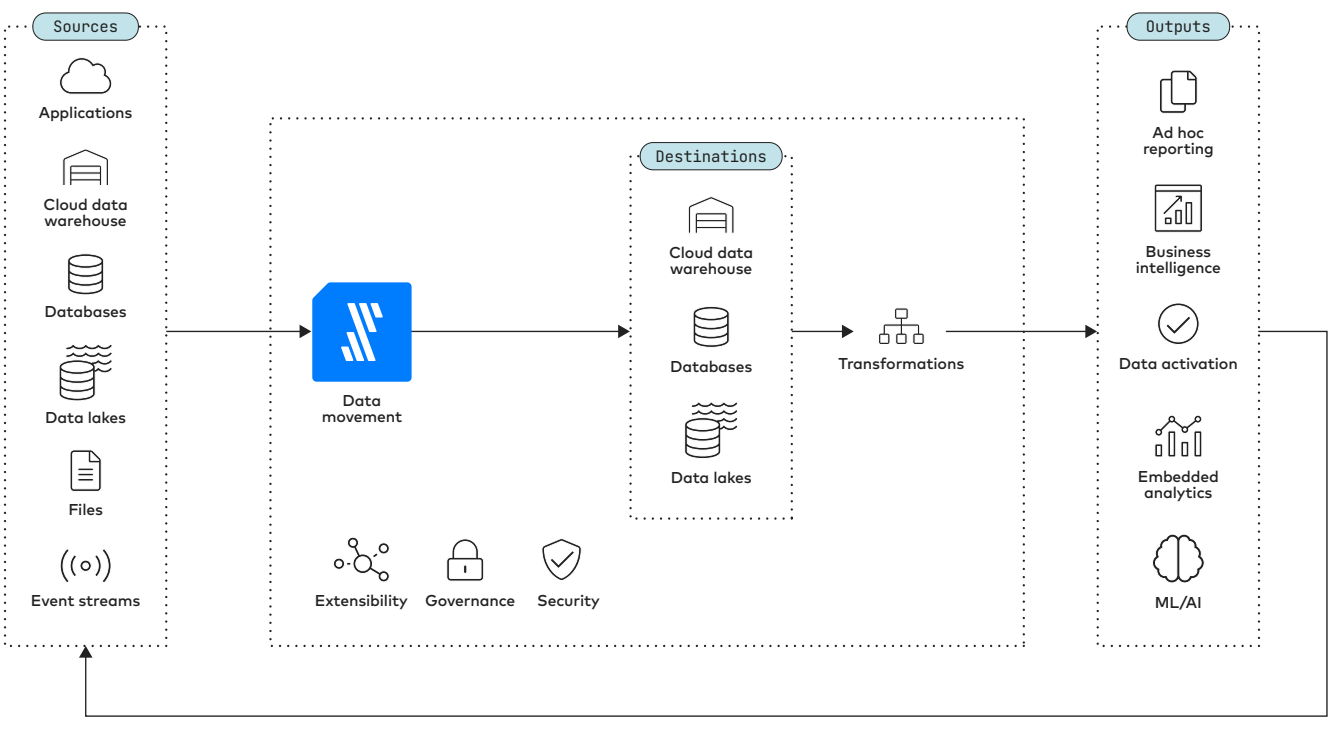
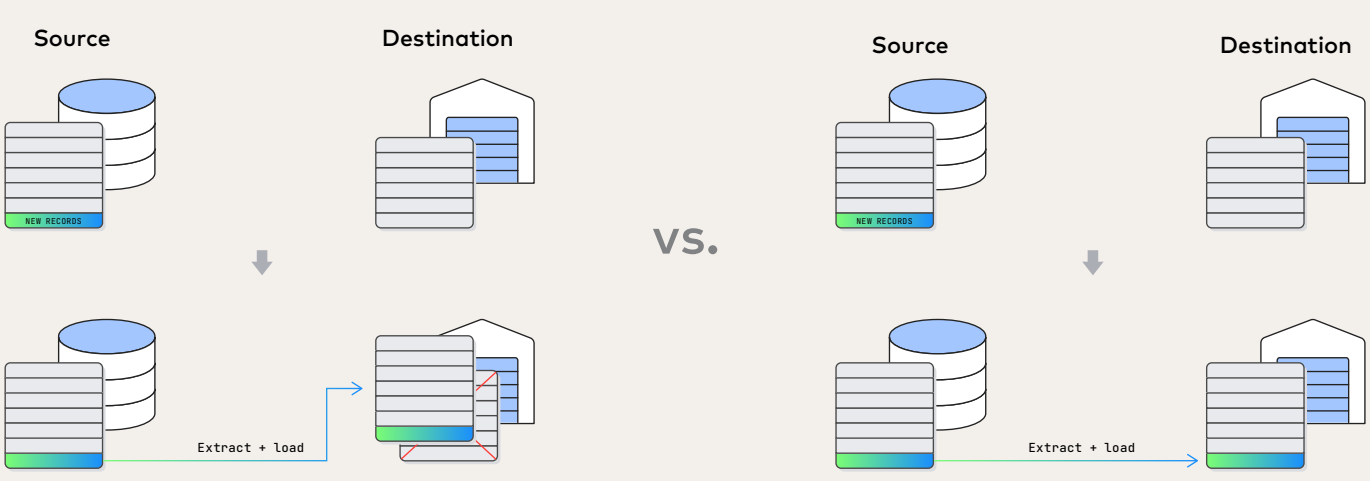
Key data engineering principles for data movement



Modern organizations use data pipelines to move data between platforms of all kinds to support operations and analytics.

Moving data from a source to a destination is deceptively complex. The following are critical features of highly performant data pipelines.

Incremental updates



Full syncs capture full data sets during an initial sync or to recover from systemic data integrity issues. However, full syncs necessitate copying entire data sets and:

1- Take too long for routine use
2- Use compute at the source and destination needed for operations and analytics
3- Consume too much bandwidth

Whenever possible, updates from a data source should be made incrementally, changing, adding or removing records only as needed.

Incremental updates require the ability to identify changes made to a previous state of the source. This is often referred to as change data capture (CDC).

Idempotence

```
Mathematically, idempotence can be
expressed thusly:

f(f(x)) = f(x)

The absolute value function abs()
is idempotent

abs(abs(x)) = abs(x)
```

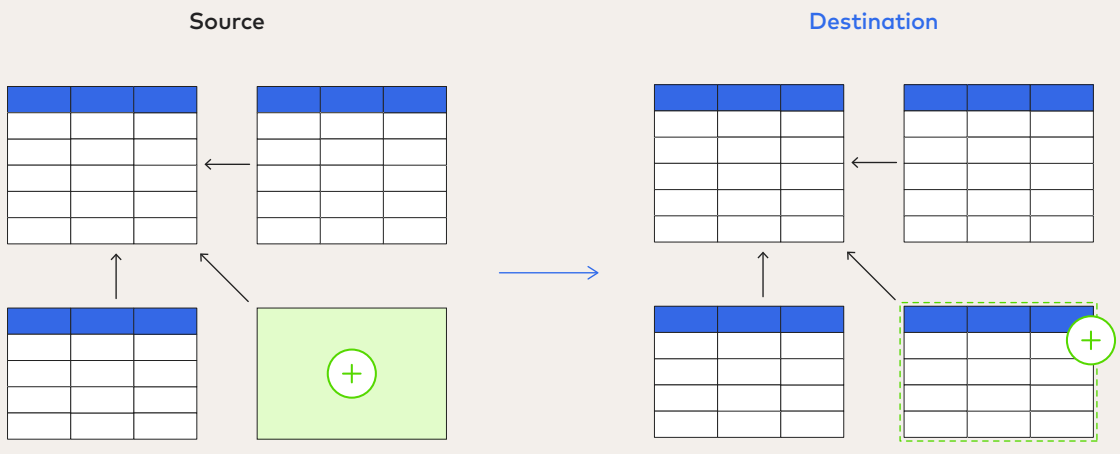
Idempotence means that if you execute an operation multiple times, the final result will not change after the initial execution.

Consider elevators; pushing the button for a particular floor will always send you to that floor no matter how many times you press it.

For data movement, idempotence ensures that if you apply the same data to a destination multiple times, you will ge the same result. Without idempotence, you must sift through the data to identify duplicate records and develop a custom recovery procedure whenever a sync fails and must be repeated.

Schema drift handling

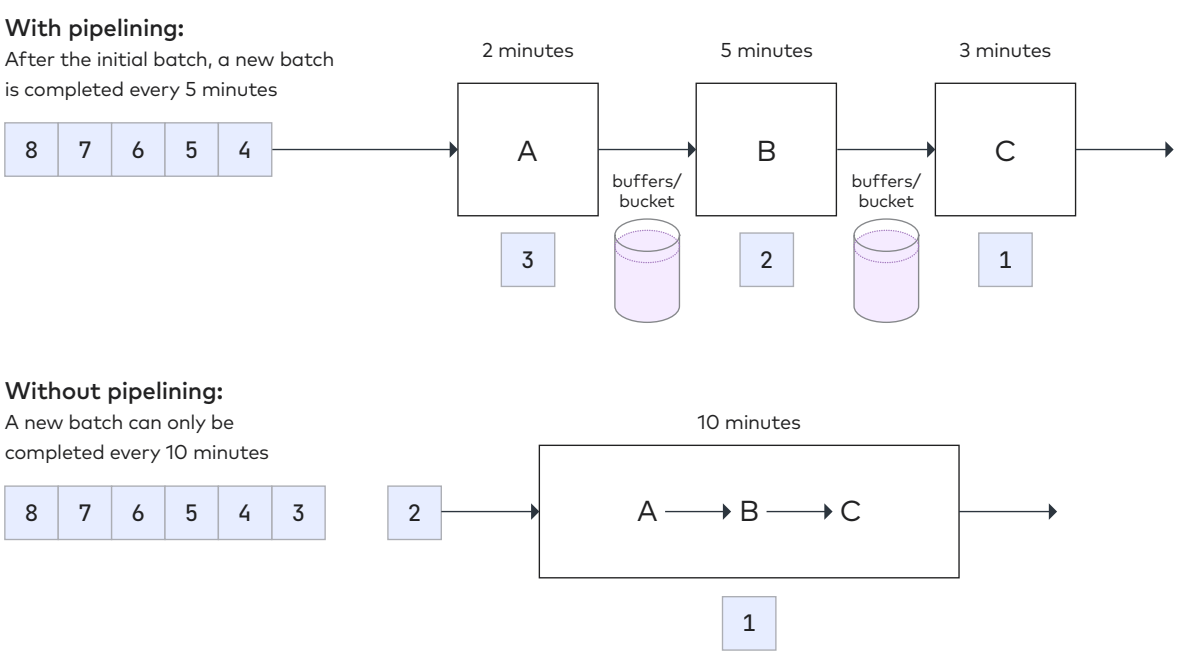
When upstream schemas change, data pipelines that depend on specific configurations of data may fail. Schema drift handling is essential to reliable pipeline functioning as well as faithful replication of the source data. A straightforward method is live updating, in which the source data model is perfectly replicated at the destination.



Pipelining and network performance

Data pipelines can experience many kinds of performance bottlenecks. This is further complicated by the movement of data between sources, nodes and destinations across a network. There are generally three basic methods to address these:

1- **Algorithmic improvement** and code optimization
2- **Architectural changes** to enable parallelization
3- **Pipelining** to separate data movement processes into sequential stages that can be simultaneously active, like an assembly line



Check out our ebook, *The ultimate guide to modern data movement* to learn more!