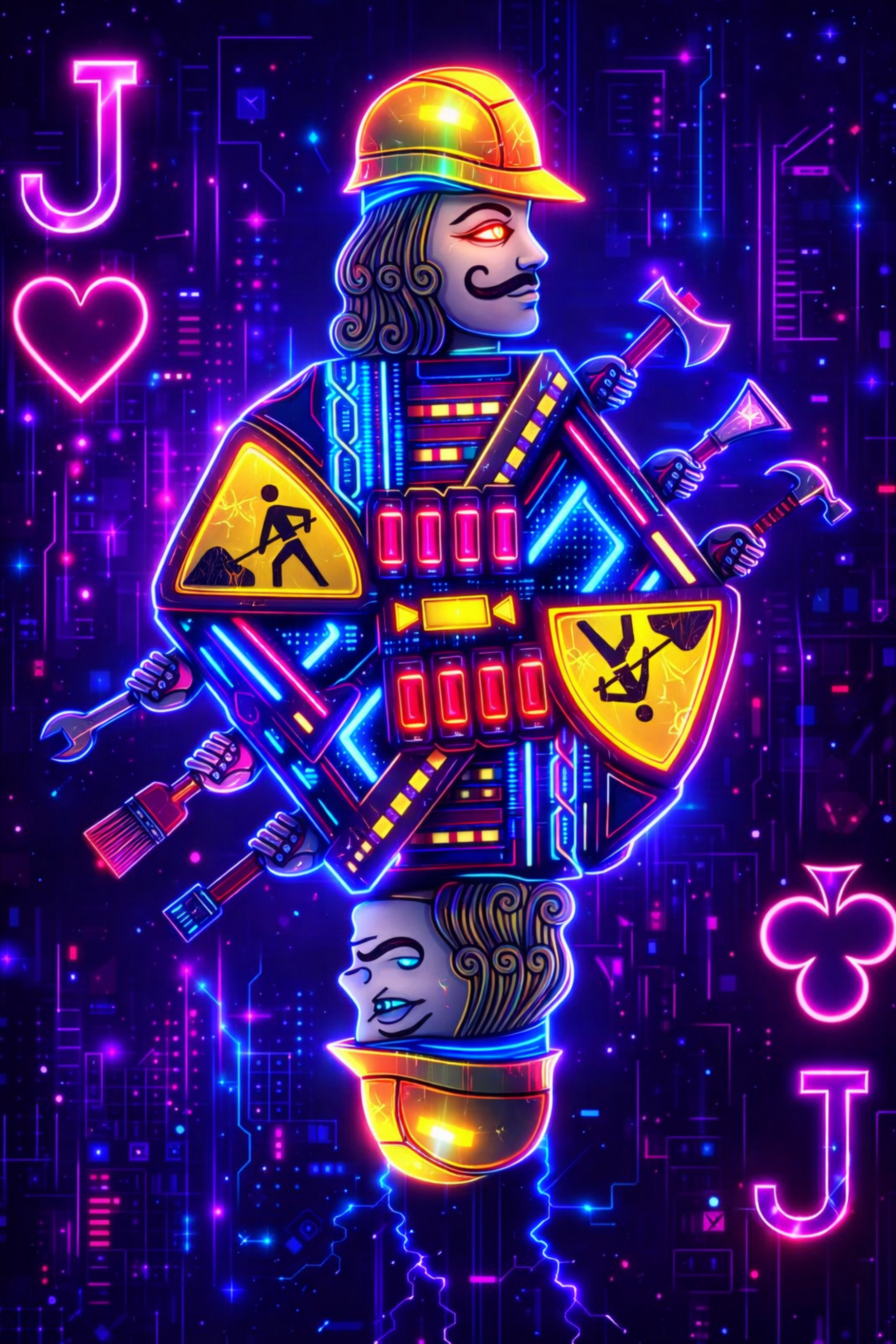




Jack of all trades

*Query Federation in modern
OLAP databases*

Agenda

1. What is query federation?
2. Two philosophies
3. Why is federation important?
4. Iceberg federation
5. Challenges of querying a data lake
6. Building a unified lakehouse architecture
7. Key takeaways

Query Federation

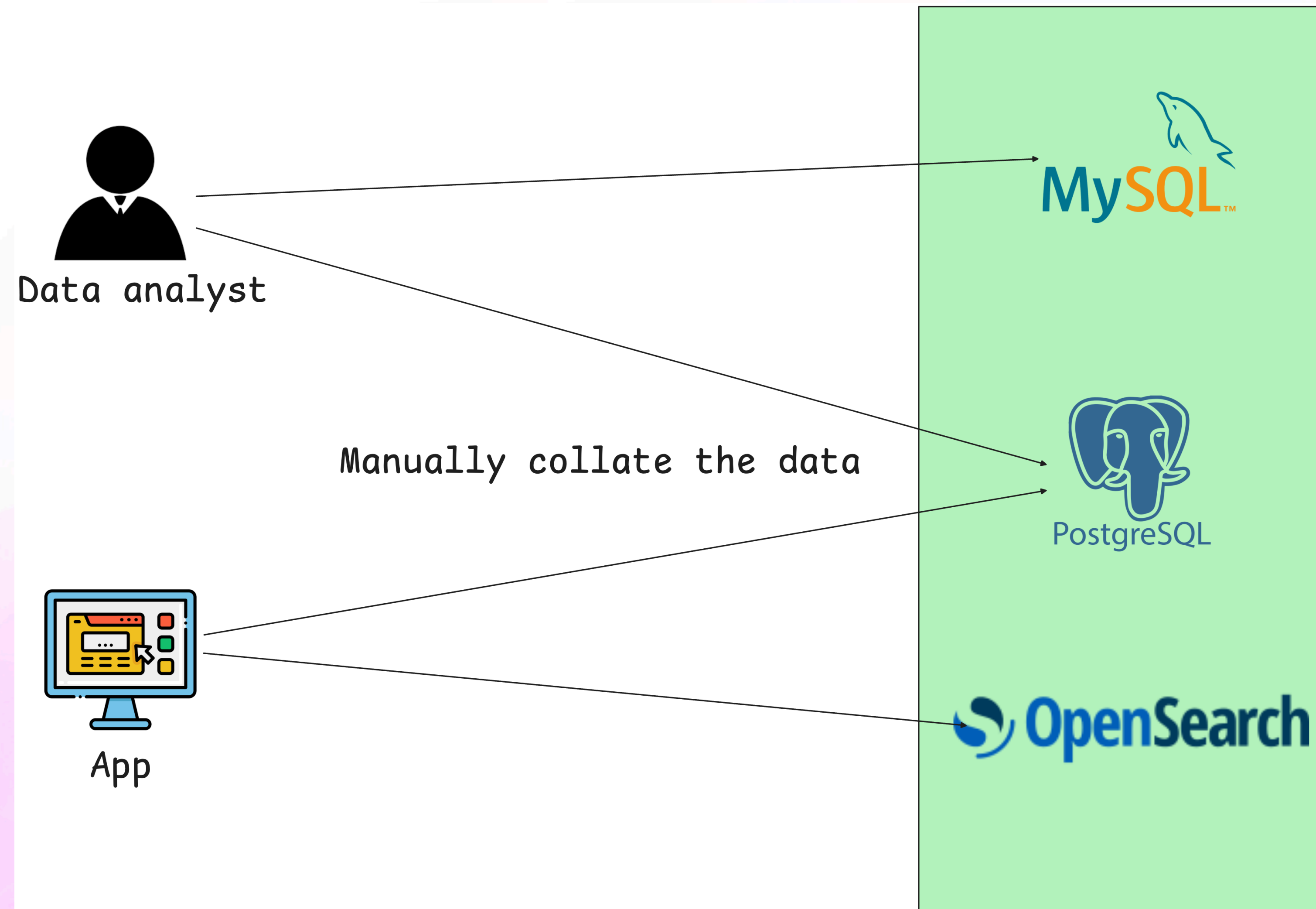
Query data across different data sources

No need to physically copy or replicate data

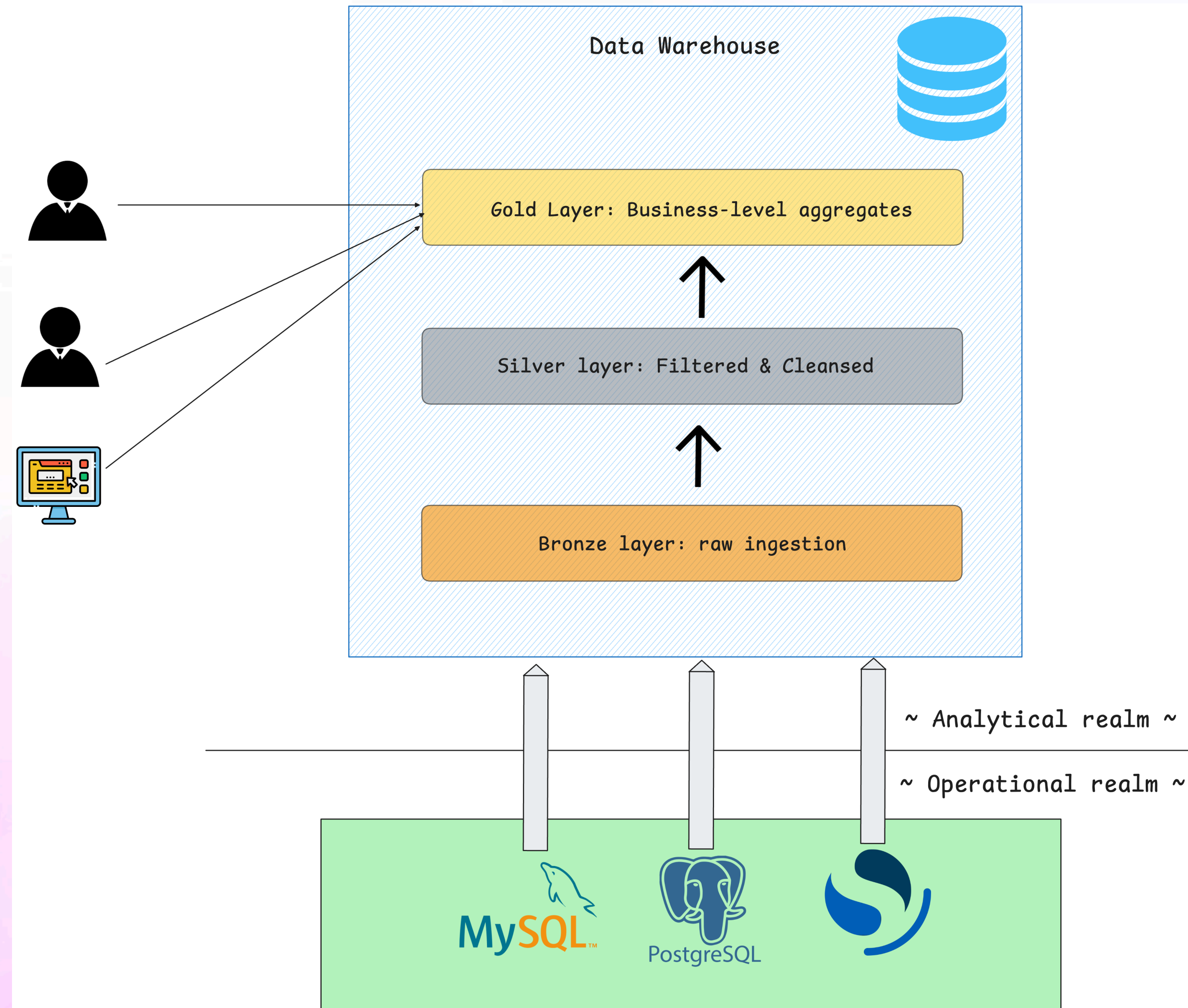
Democratise access to data

Simplify data architectures

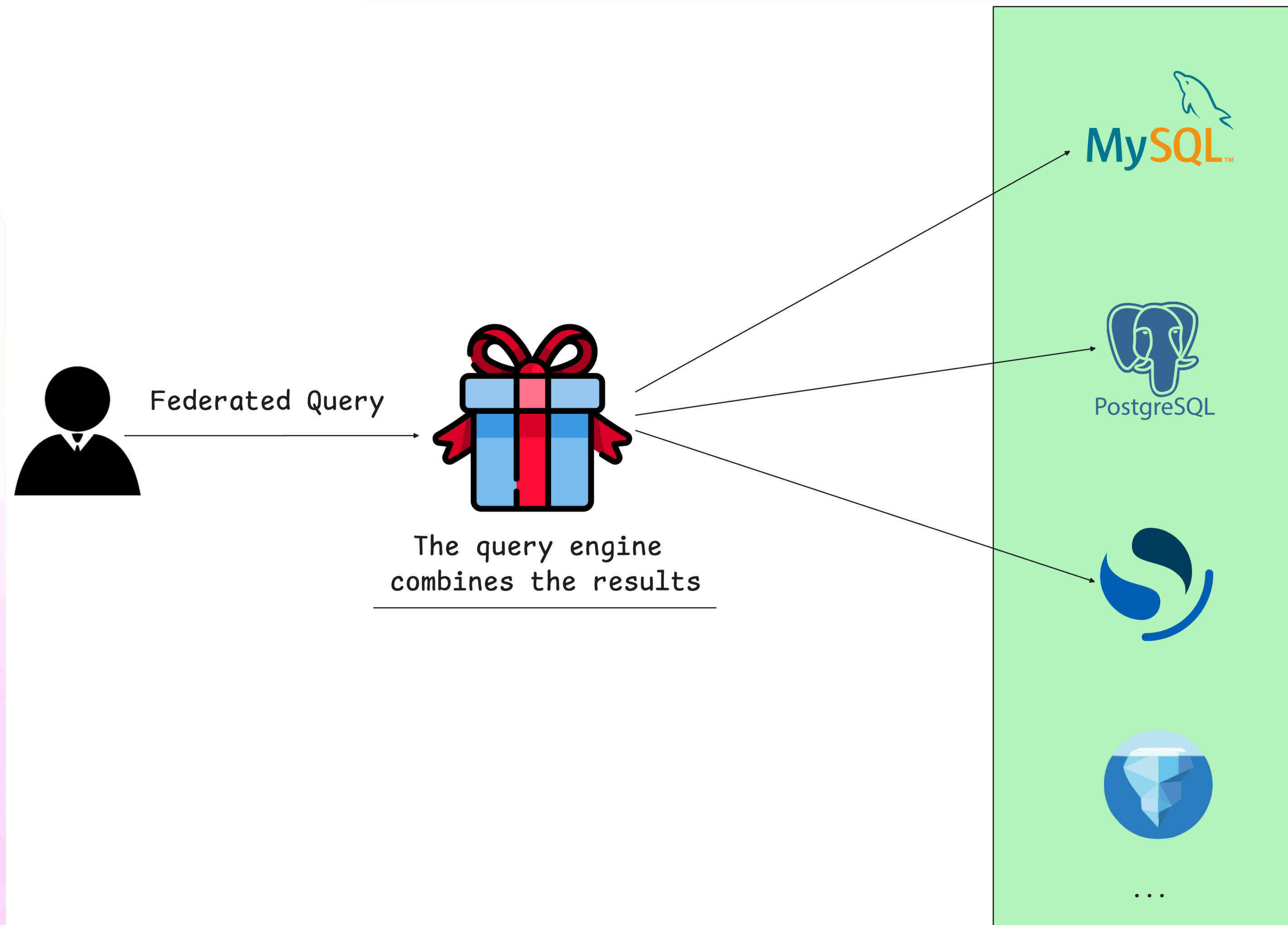
Without Query Federation



Without Query Federation



Query Federation



Federation Challenges

Performance driven by the slowest source

... And also difficult to predict

Data access patterns and Security policies

Network limitations

Querying Data Lakes

Two philosophies

Federation as a First-Class
Global Query Layer

Federation as an Extension
to an OLAP engine

General purpose federation layer

Data is queried where it lives

Optimised for flexibility

Handle as many data sources as possible



Complementary capability

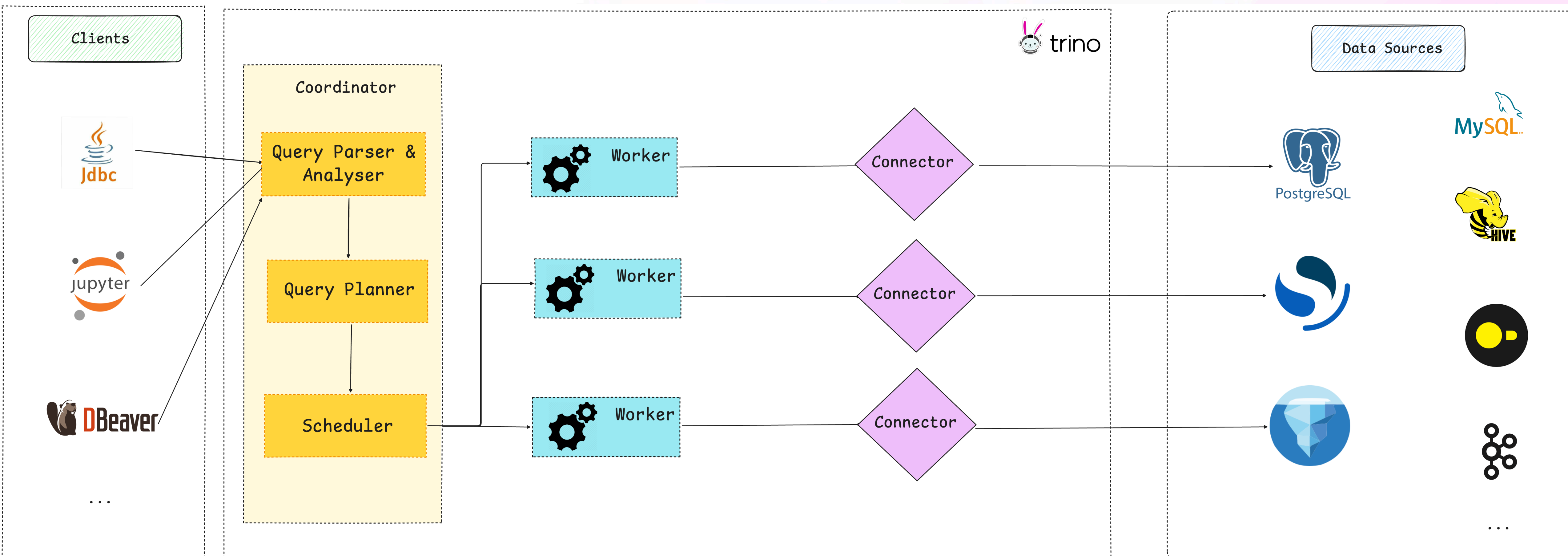
External data integrated into the engine

Optimised for predictability

Handle few important sources well



Trino

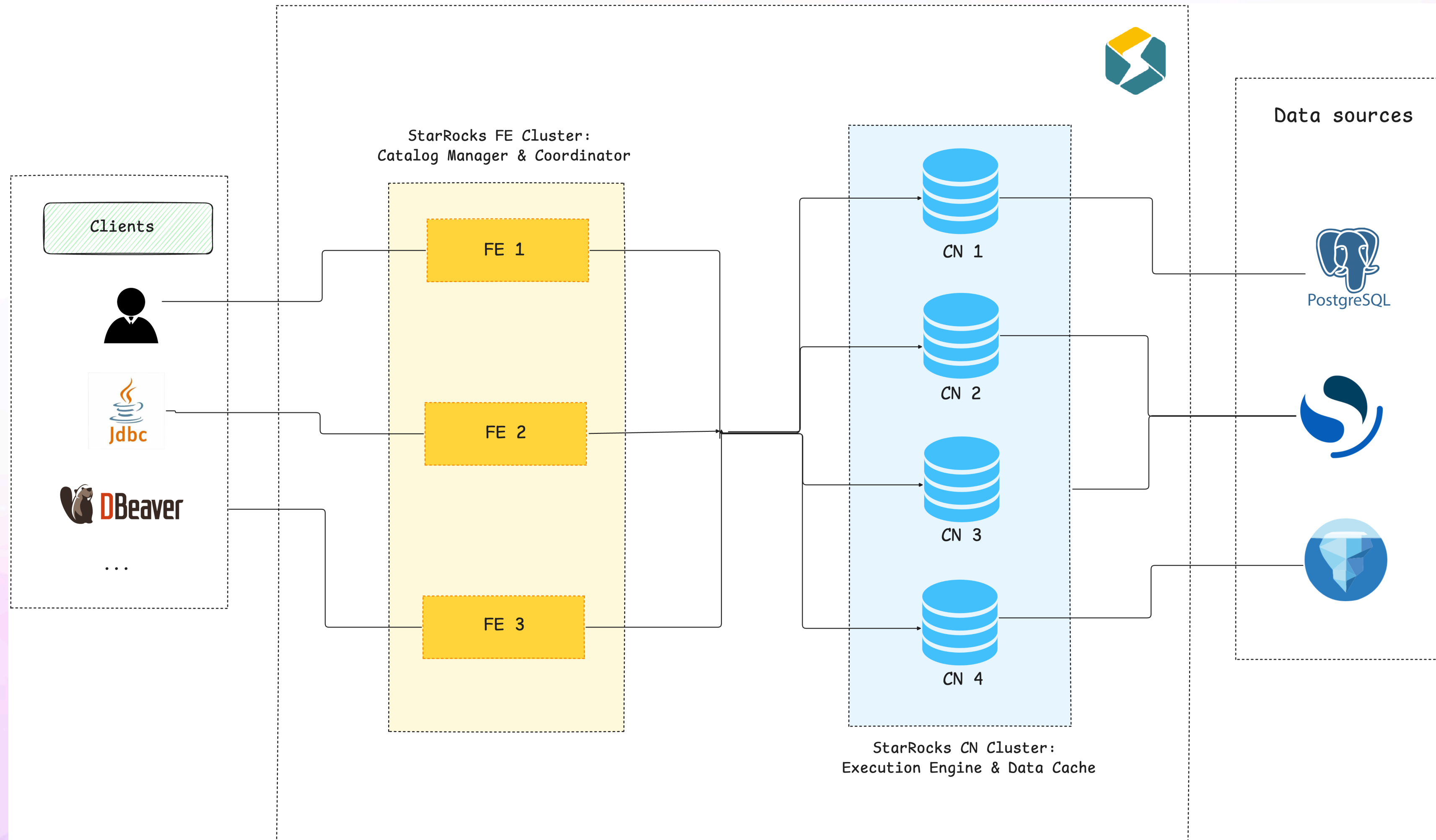


StarRocks

- OLAP Database
- Open Source
- Vectorized, MPP processing engine
- **Federated query engine**
- Lake-house integrations
- Cost Based Optimizer for query acceleration
- MySQL protocol compatibility



StarRocks



Trino vs. StarRocks

Concern	Trino	StarRocks
Optimisation goal	Global efficiency	Latency predictability
Flexibility	Very high	Moderate
Statistics reliance	Low	High
Adaptability	High	Moderate
Performance	The slowest data source	OLAP engine optimisations, caching
Concurrency	Moderate	Very high
Best workload	Exploratory federation, ad-hoc queries	Production OLAP

Why is Federation Important?



Why is Federation Important?

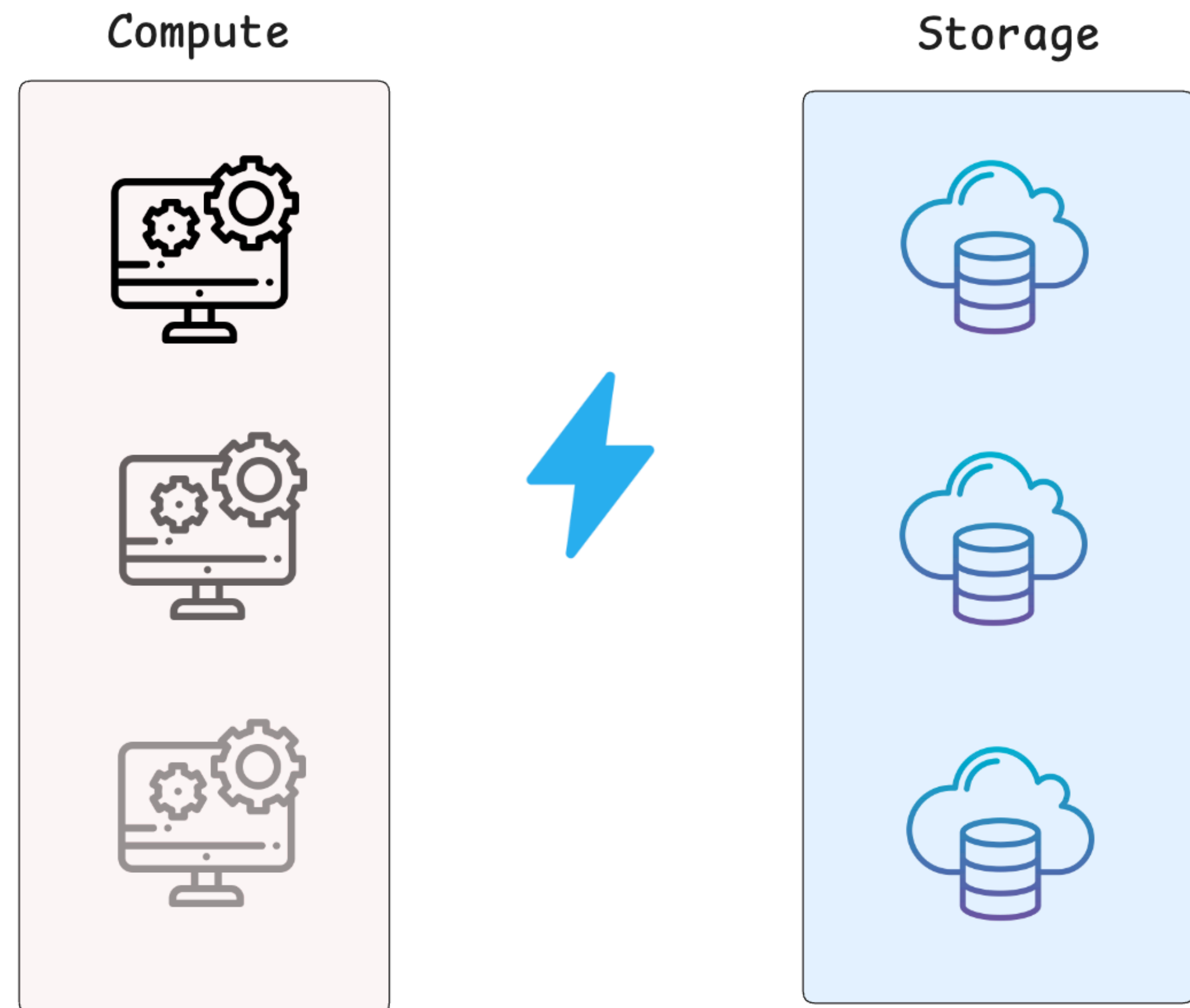
Compute Storage separation

Data is persisted in cheap cloud storage

Compute is provisioned on demand

Lakehouse architectures

on top of Open Table Formats



Challenges Federating to Data Lakes

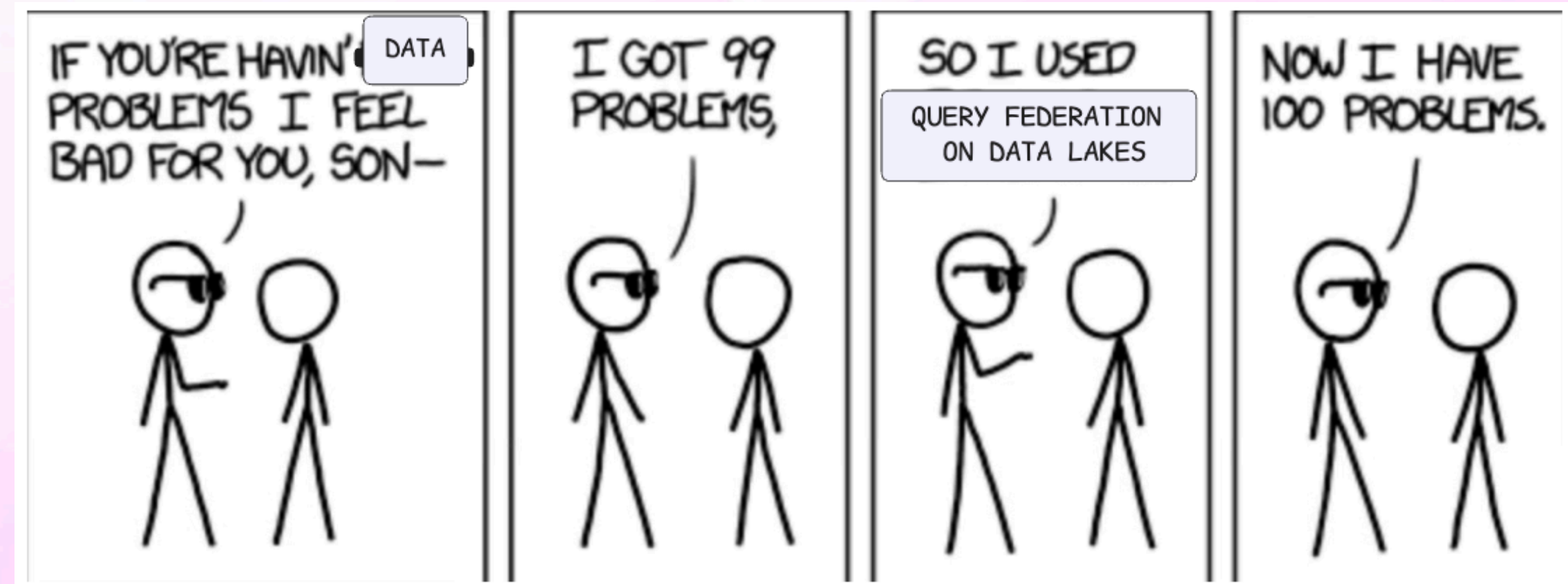
Object Storage Latency

Semantic load on engines

Data Freshness

File Fragmentation

Schema evolution



Use case: Apache Iceberg

Open Table Format, Open Spec

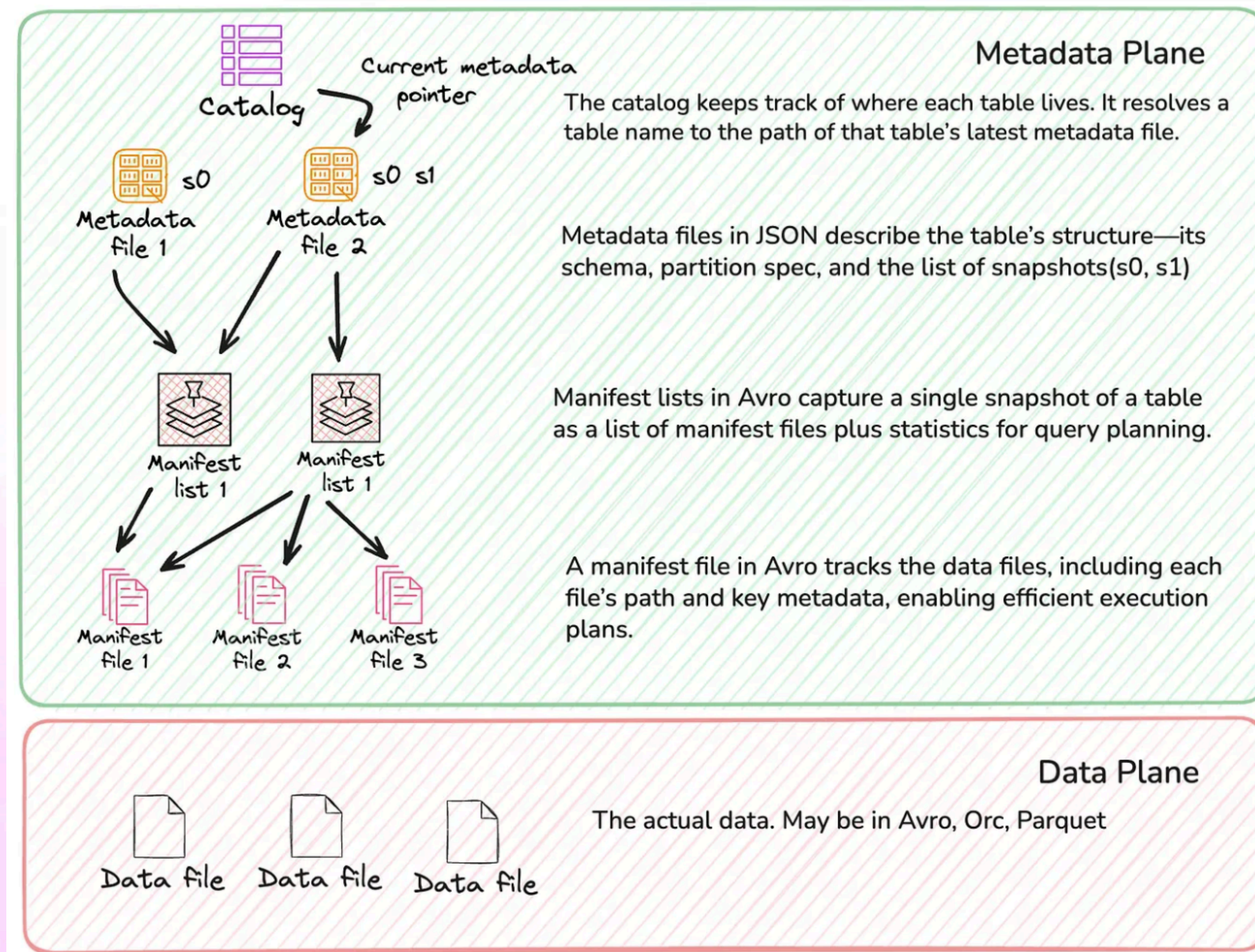
Wide Support in open source

**Just a series of manifest lists and manifest files
that point to the actual data files**

Readers rely on statistics to improve performance



Use case: Apache Iceberg



Federating to Iceberg



Scan Tables directly in object storage

Pay the price of opening files at query time

Highly sensitive to table maintenance



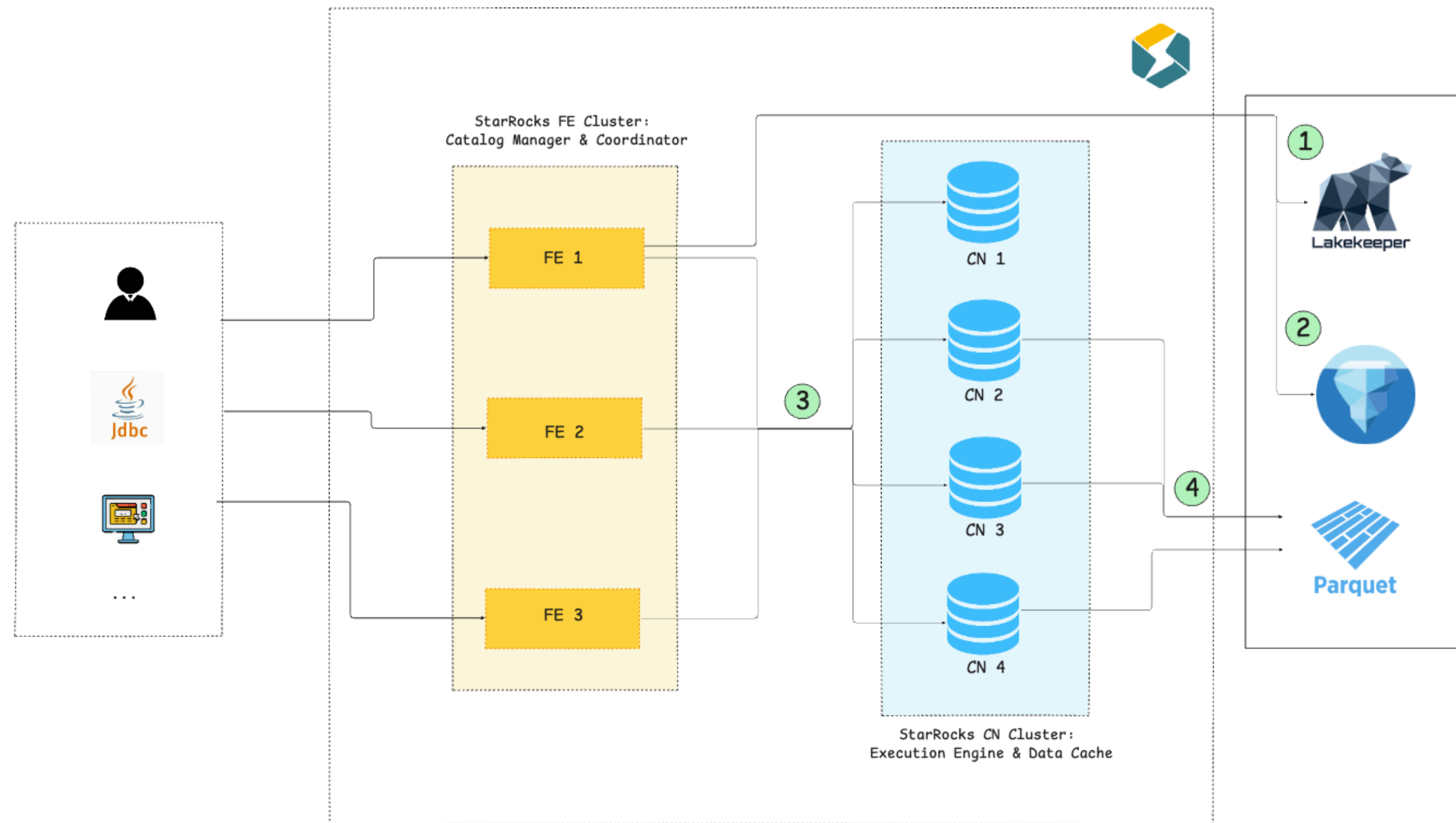
Bring external data into the OLAP Engine

Cache data, metadata, results

Optionally Materialise results

Minimise remote reads

StarRocks Federation to Iceberg

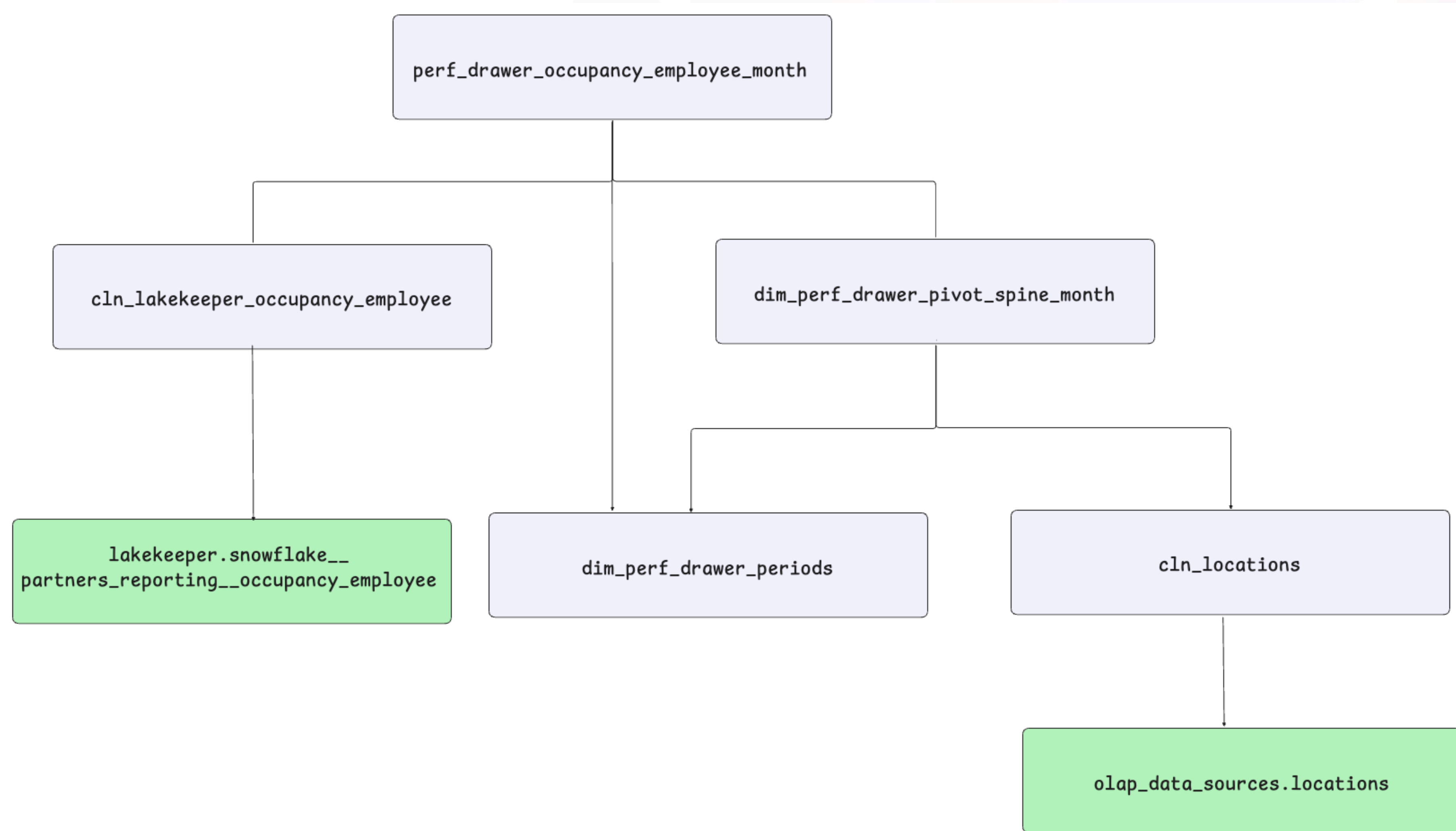


How it looks in practice

```
mysql> SELECT *  
-> FROM perf_drawer_occupancy_employee_tab_month  
-> WHERE provider_id = 604262;
```



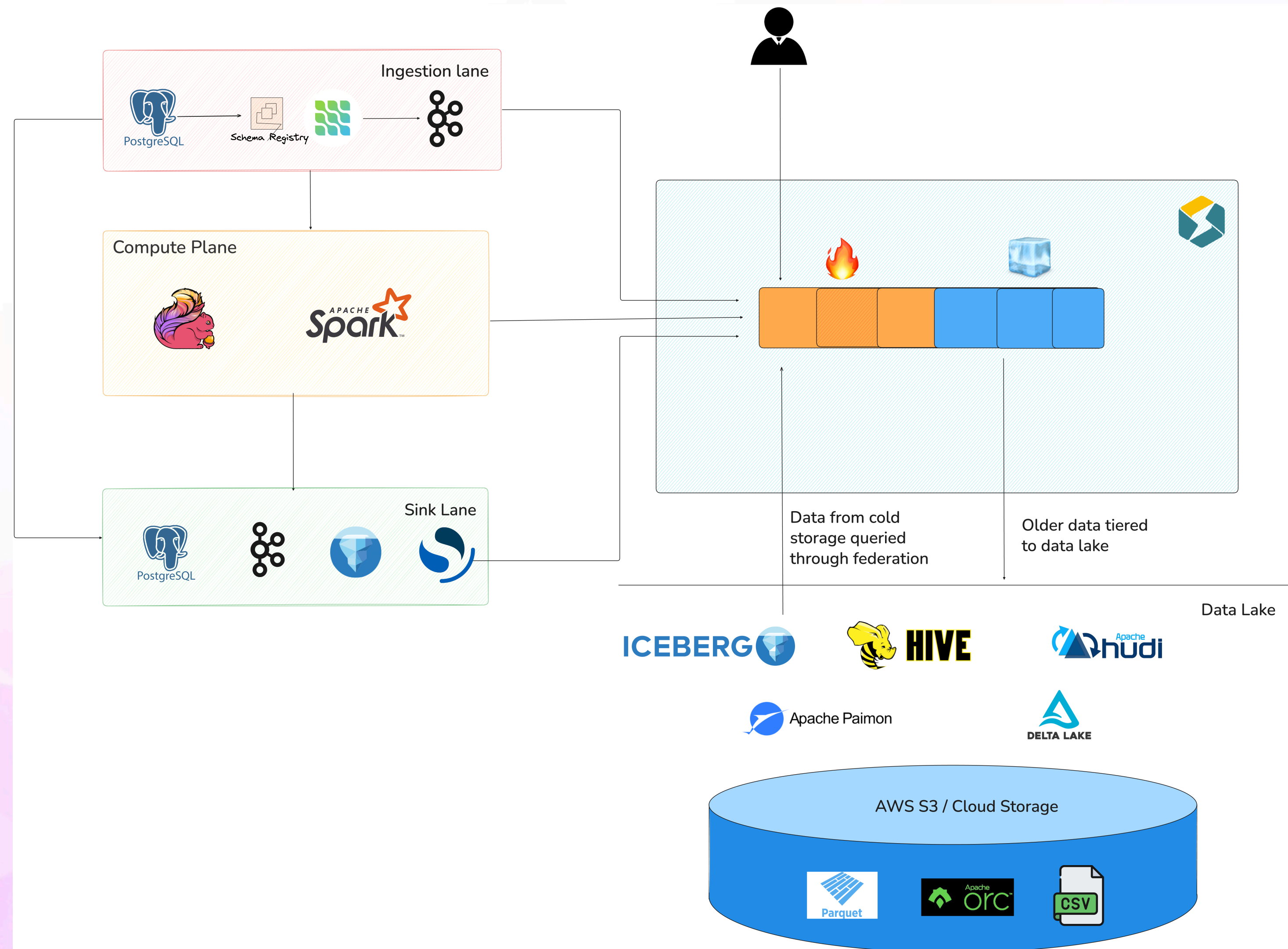
How it looks in practice



Addressing the challenges

Object Storage Latency	Metadata + Data Caching
Data Freshness	Hot/Cold Data Separation
File Fragmentation	Caching + Materialized Views
Schema Evolution	Asynchronously refresh metadata

Putting it all together



Putting it all together

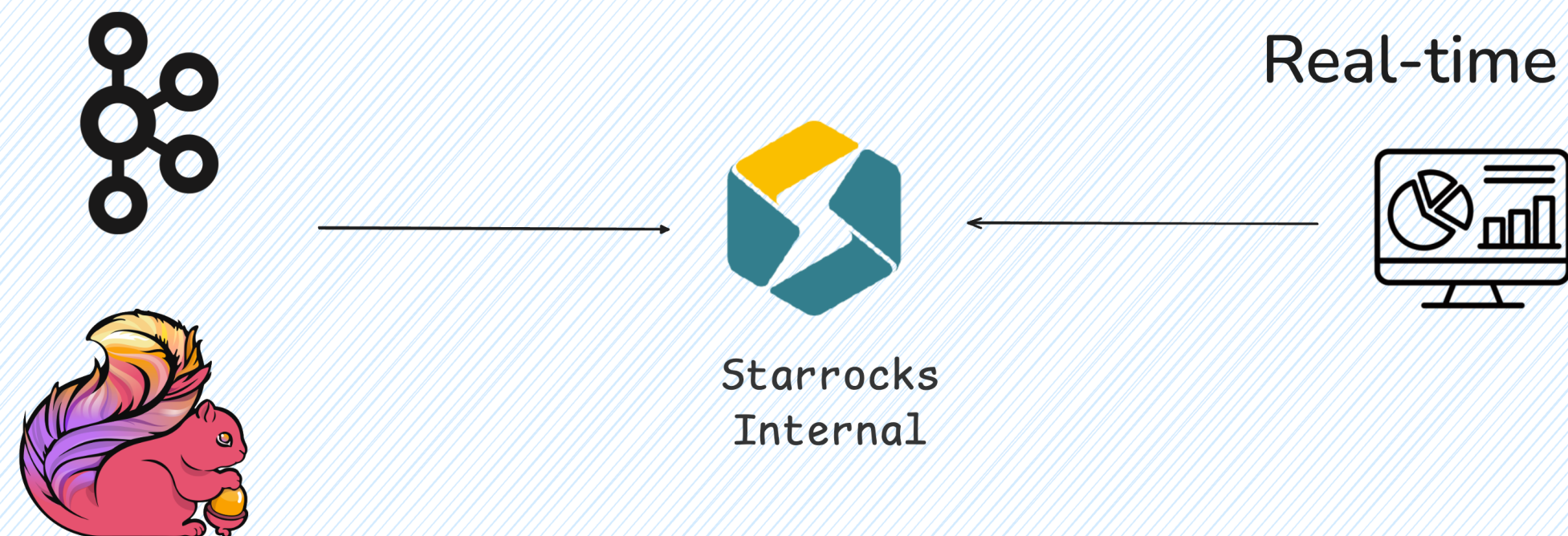
Ingestion lane



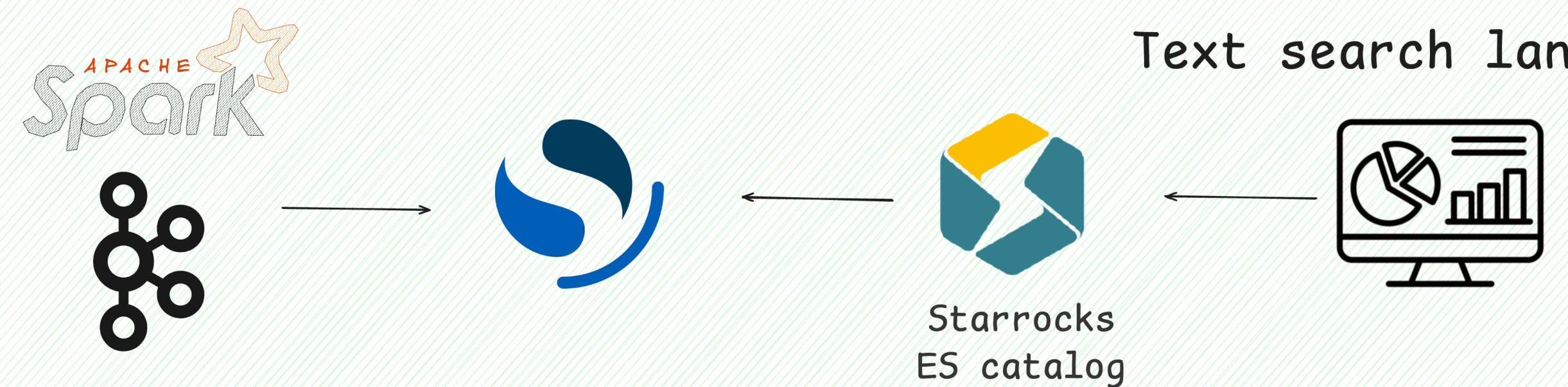
Historical lane



Real-time lane



Text search lane



Key Takeaways

Two philosophies: Orchestration-first vs. engine-first

They come with tradeoffs in terms of performance, cost and use cases

Industry shifts towards cheap, reusable, universal storage

Query engines need to adapt to the new lakehouse architectures

Q&A

Thank you for listening!

