# Hybrid Query Execution

## What is a Database Client, Anyway?

Jordan Tigani

co-founder & chief duck-herder @MotherDuck

2023-08-28

MotherDuck

## WHO AM I?

- MotherDuck Co-Founder
- MemSQL/SingleStore CPO
- BigQuery PM/Eng Director
- BigQuery Storage Tech Lead
- MIcrosoft Research

Photo above: Prophesying the end of Big Data.

## About this talk

Opinionated History of Cloud Data Warehouses and Data Lakes

What is Hybrid Execution and why would you want to use it?

How does Hybrid Execution work?

What are some new things to try?

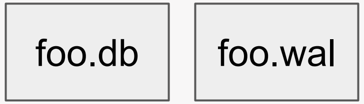
Telling stories with diagrams like the one above

## How did we get here?

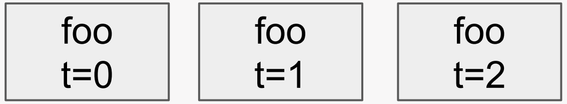
The rise of the Cloud Data Warehouse

# Rise of the CDW: It’s all about the storage

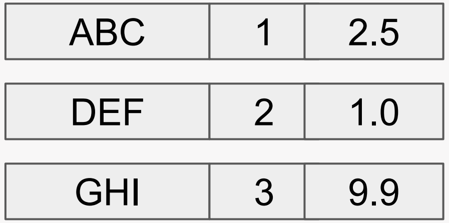
## Local Disks



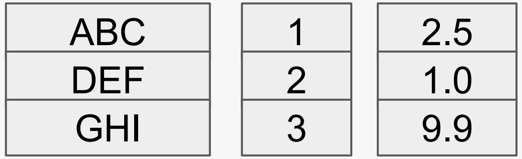
## Object Stores



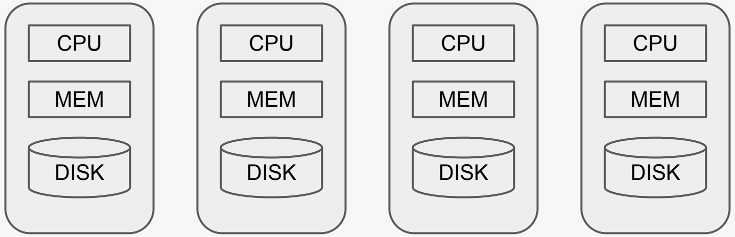
## Row Stores



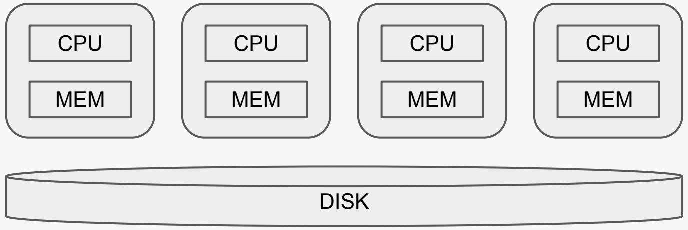
## Column Stores



## Shared Nothing

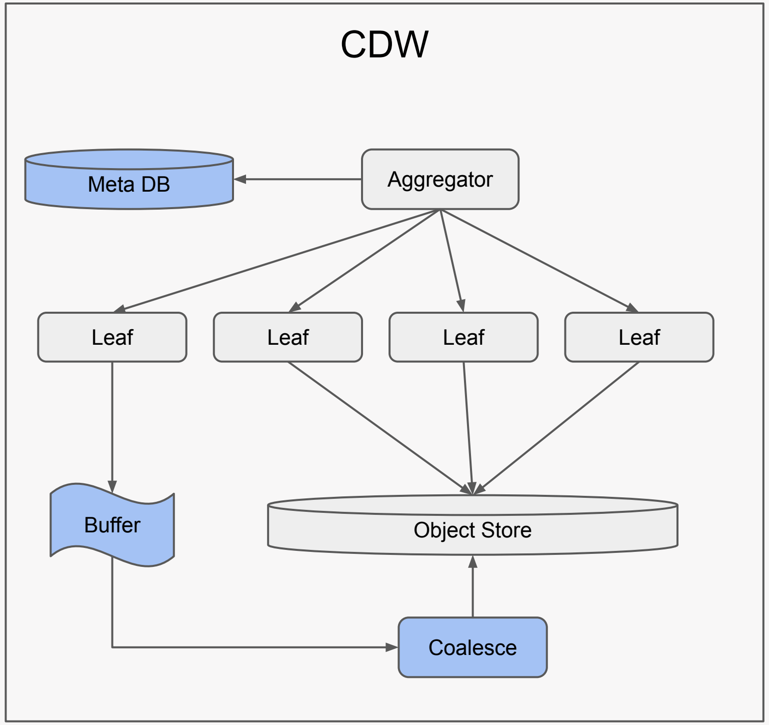


## Shared Disk



# Important features for CDW Storage

- File Management
  - Problem: Performance degrades over time
  - Solution: Background Compaction & Coalesce
- Metadata Management
  - Problem: Need atomic updates
  - Solution: Metadata points to latest data objects
- Streaming buffers
  - Problem: Column stores aren't great at frequent updates
  - Solution: Buffer recent data in a rowstore / log
- Storage Pushdown
  - Problem: Reading too much data is slow / expensive
  - Solution: Segment-level storage statistics



## Meanwhile …
## In a Parallel Universe

Opening up the Data

# Why don’t we drop our data in a lake?

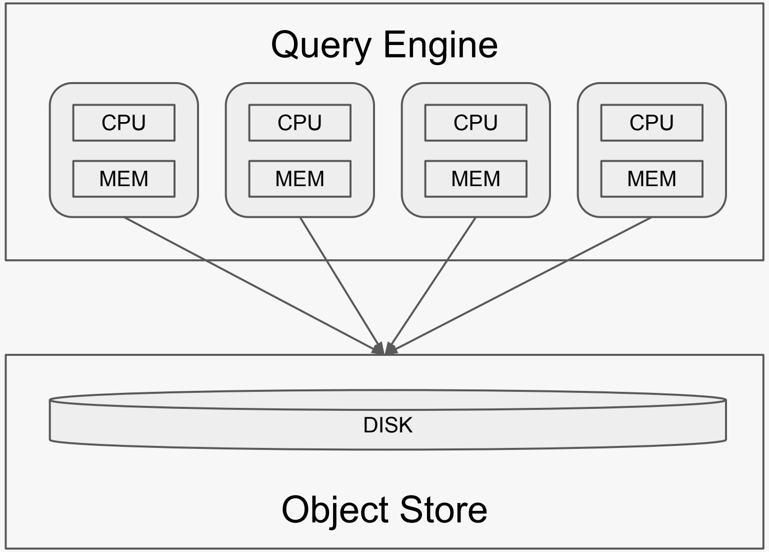
## Data Lakes are Great!

- Open data formats
- Write once, figure out out later
- Infinitely Scalable
- Inexpensive

## But… Data Lakes are Terrible!

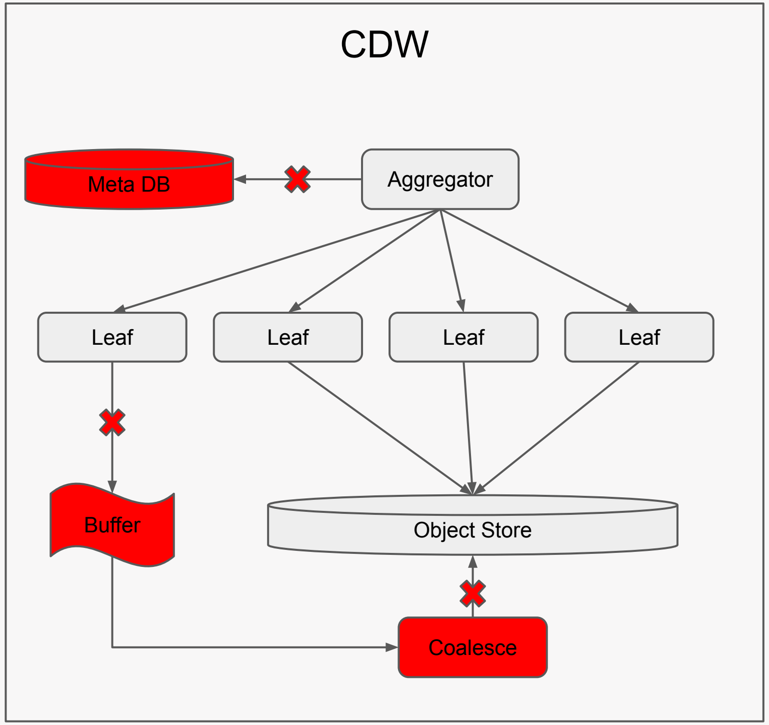
- Data Swamps
- No governance
- Do we still need this file?
- Mediocre Performance

Data Lake Architecture



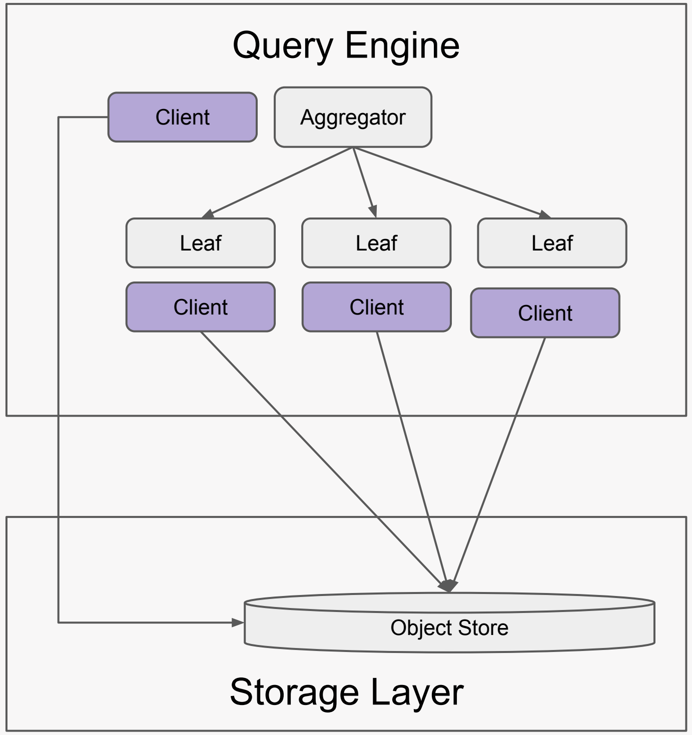
# Comparing Data Lake problems to CDW

- File Management
  - Problem: Performance degrades over time
  - Solution: ???
- Metadata Management
  - Problem: Need atomic updates
  - Solution: ???
- Streaming buffers
  - Problem: Column stores aren't great at frequent updates
  - Solution: ???
- Storage Pushdown
  - Problem: Reading too much data is slow / expensive
  - Solution: ???



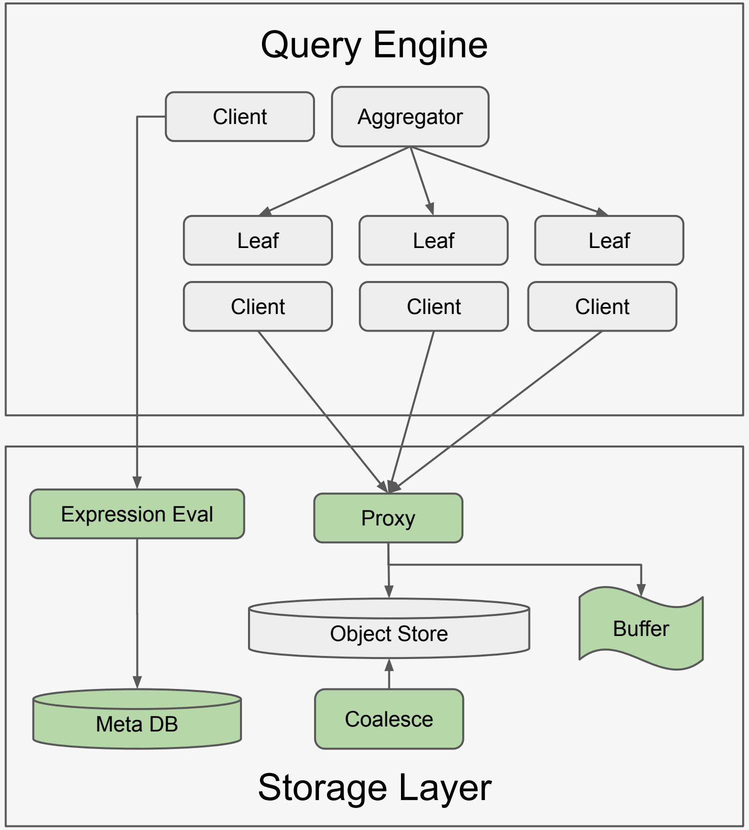
# Enter the Lake House

- File Management
  - Problem: Performance degrades over time
  - Solution: Compact-on-write or Compact-on-read
- Metadata Management
  - Problem: Need atomic updates
  - Solution: Metadata files stored in object stores
- Streaming buffers
  - Problem: Column stores aren’t great at frequent updates
  - Solution: You didn’t really want fresh data did you?
- Storage Pushdown
  - Problem: Reading too much data is slow / expensive
  - Solution: Metadata files have extent information



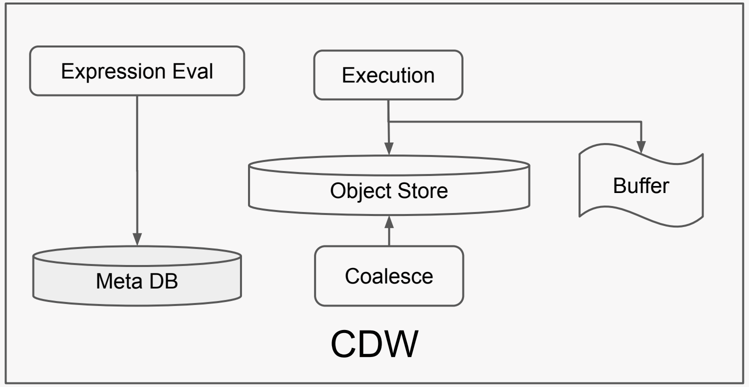
# Evolution of the Lake House

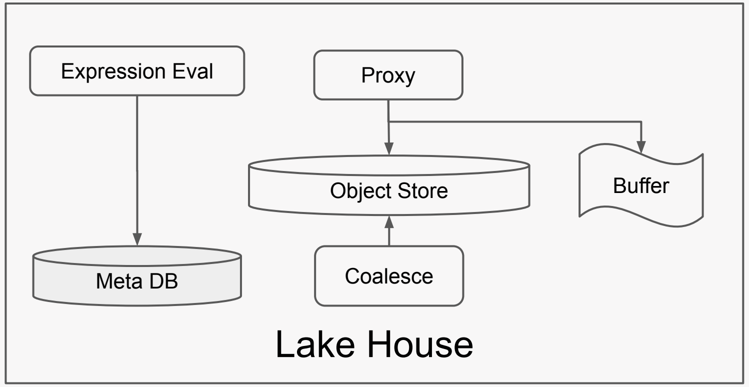
- File Management
  - Problem: Merging slows down access
  - Solution: Create coalesce service
- Metadata Management
  - Problem: Need fine grained auth
  - Solution: Proxy data
- Streaming buffers
  - Problem: Object store writes are slow
  - Solution: Streaming service
- Storage Pushdown
  - Problem: Want better segment elimination
  - Solution: Metadata service w/ Expression Evaluator



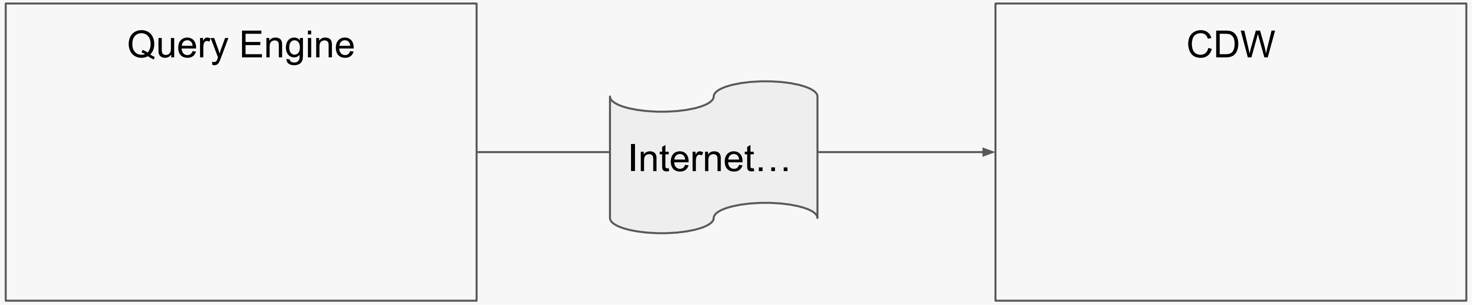
# Lakehouses Evolve Into CDWs

- Interface: Tables > Files
- Metadata: DB > Object Store
- Coalesce: Background > Foreground
- Access: Managed > Cooperative





## If your Data Lake becomes a Data Warehouse, What do you do with the query engine?



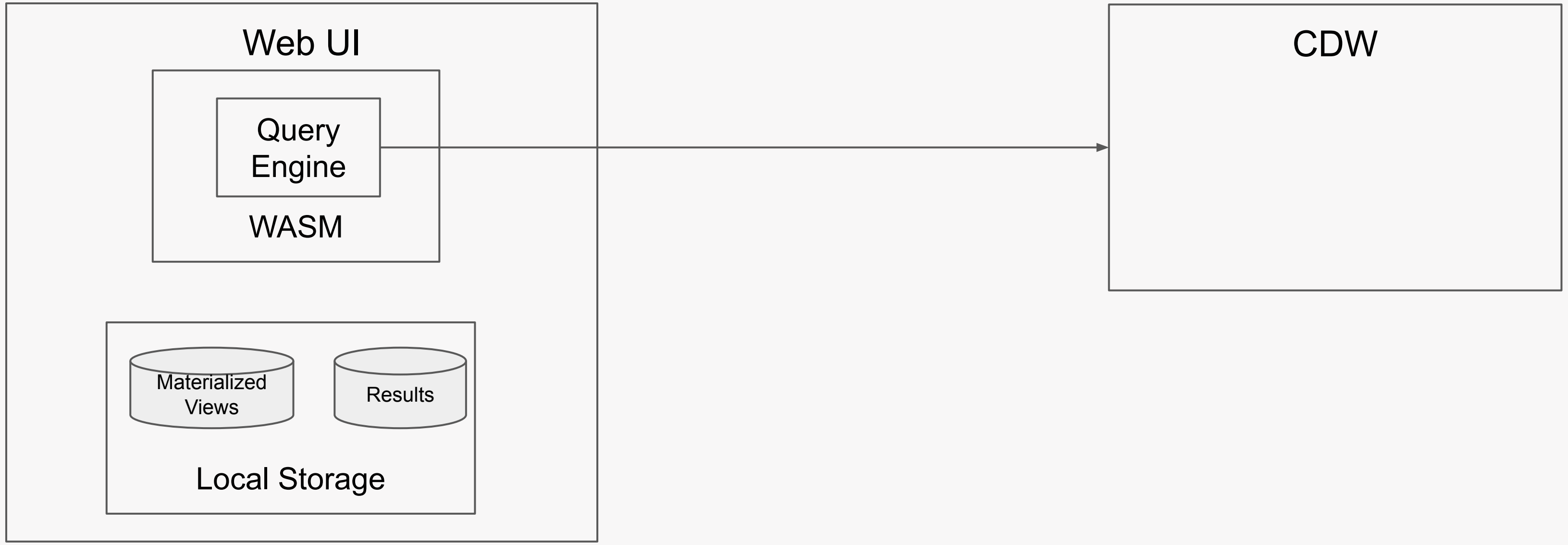
Hybrid Execution!

## Why Hybrid?

## Local Result Cache

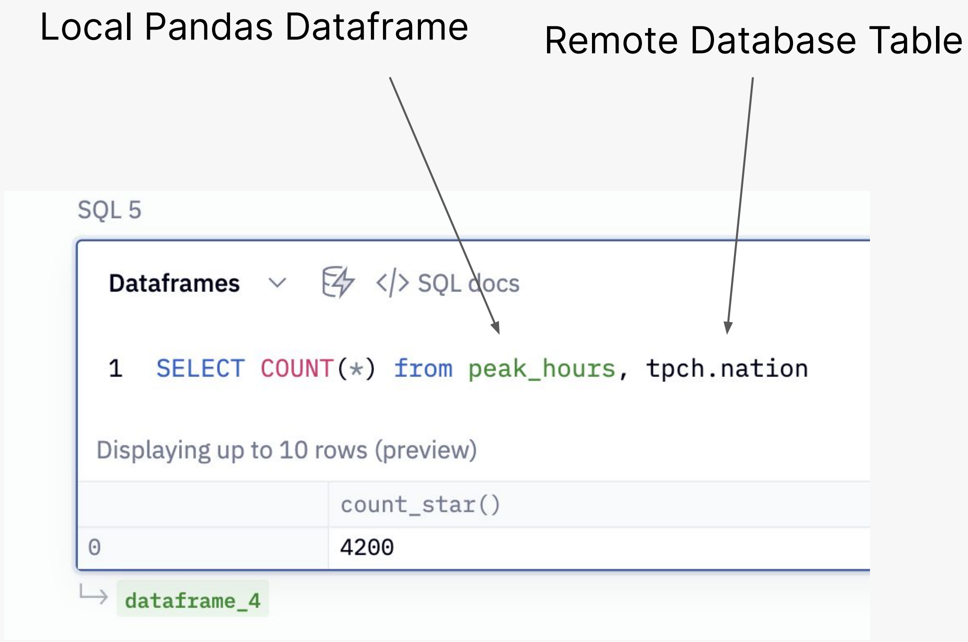
Low Latency local queries

Reactive User Experiences



# Data Science without Borders

Mix Local and Remote data seamlessly

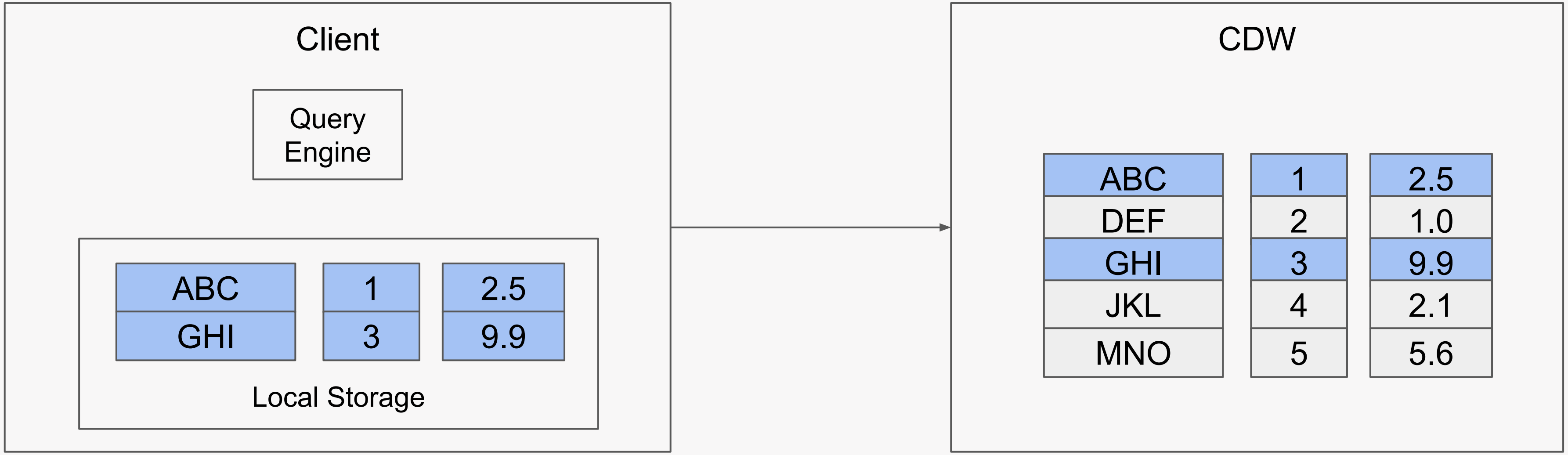


```
SELECT COUNT(*) from peak_hours, tpch.nation
```

Displaying up to 10 rows (preview)

|   | count_star() |
|---|---|
| 0 | 4200 |

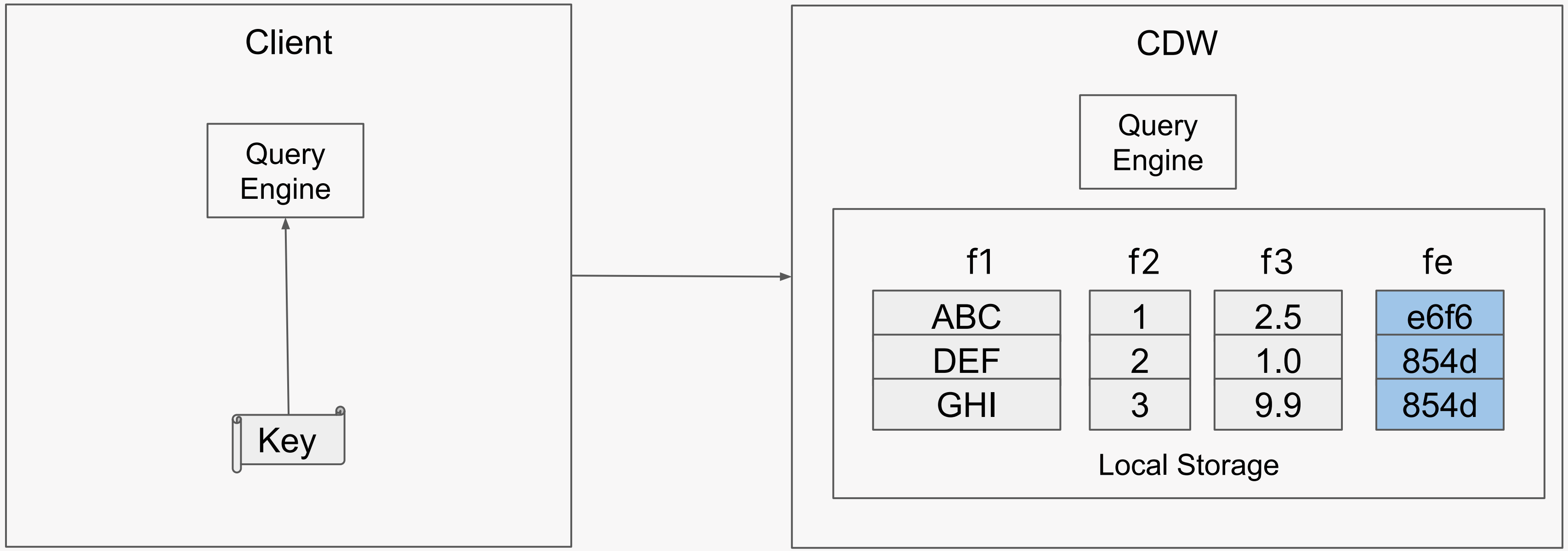
## Local Sub-Sampling



Reduce costs of running server-side

“Reasonable” size data can be run entirely client side

# Client Side Decryption



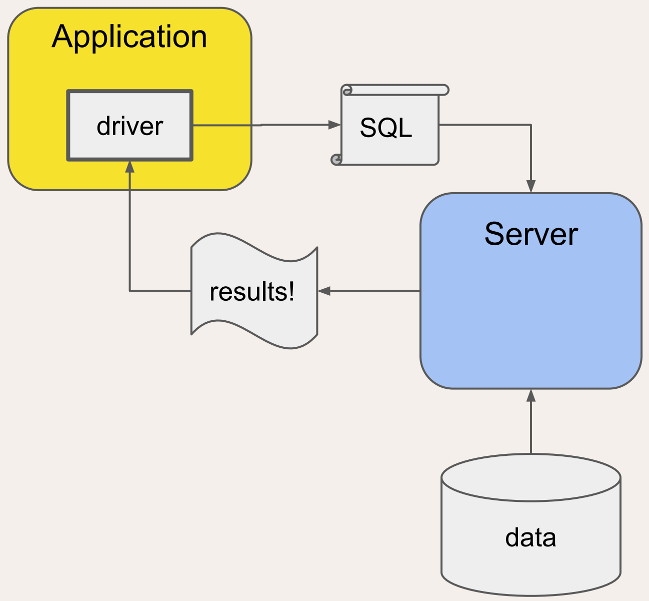
“Select f1, SUM(fe)” downloaded to client
“Select f1, f2” runs on server

## Building Hybrid Execution

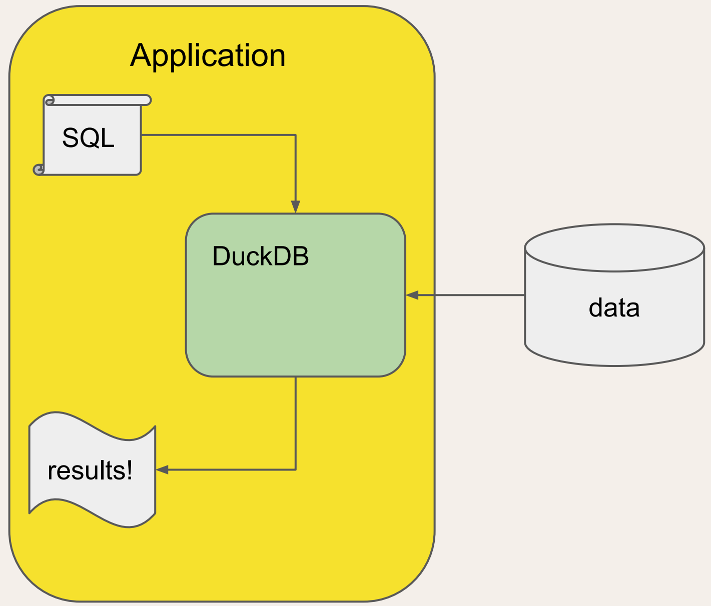
One Duck at a Time

# Embedded Database Execution w/ DuckDB

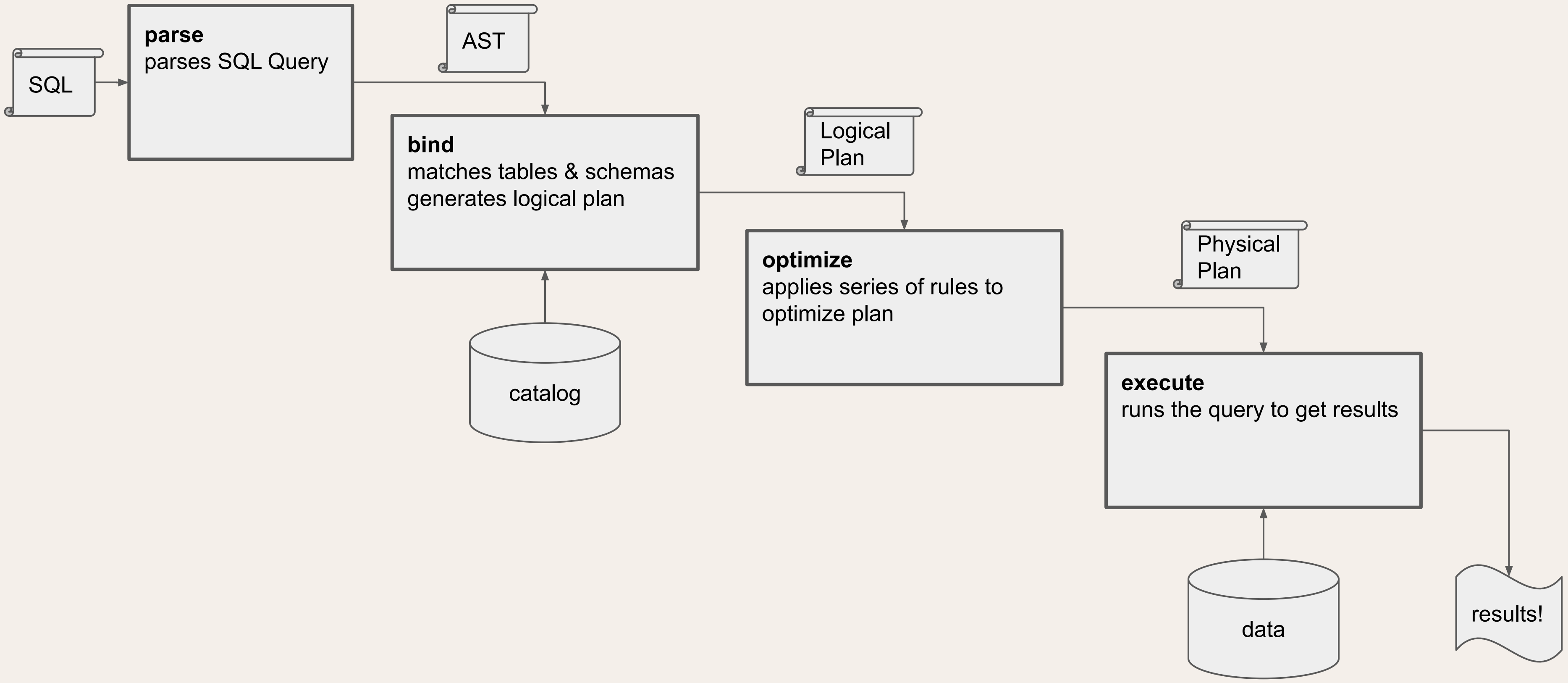
## Typical Database Execution



## DuckDB Execution

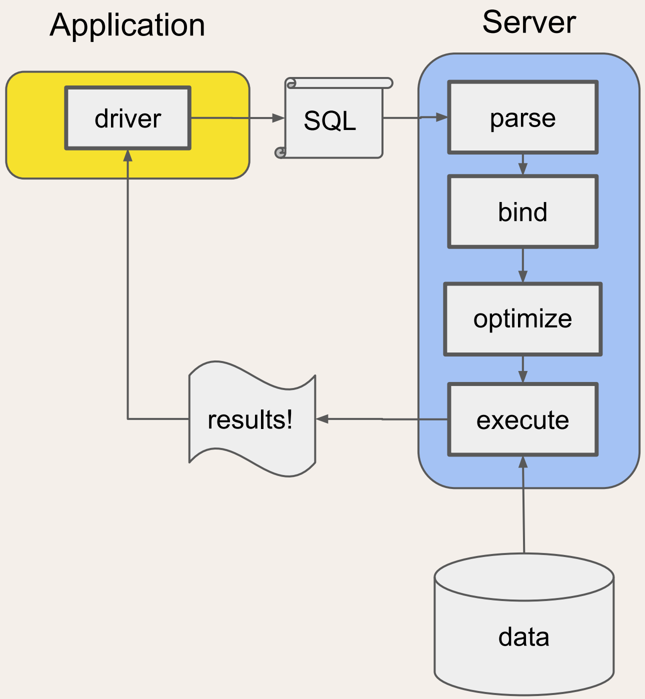


# Database Execution 101

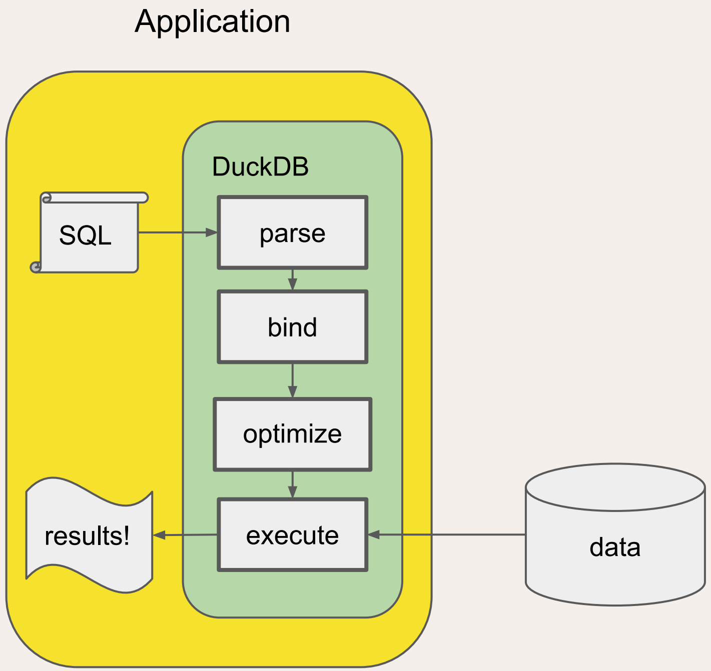


# Embedded Database Execution w/ DuckDB (II)

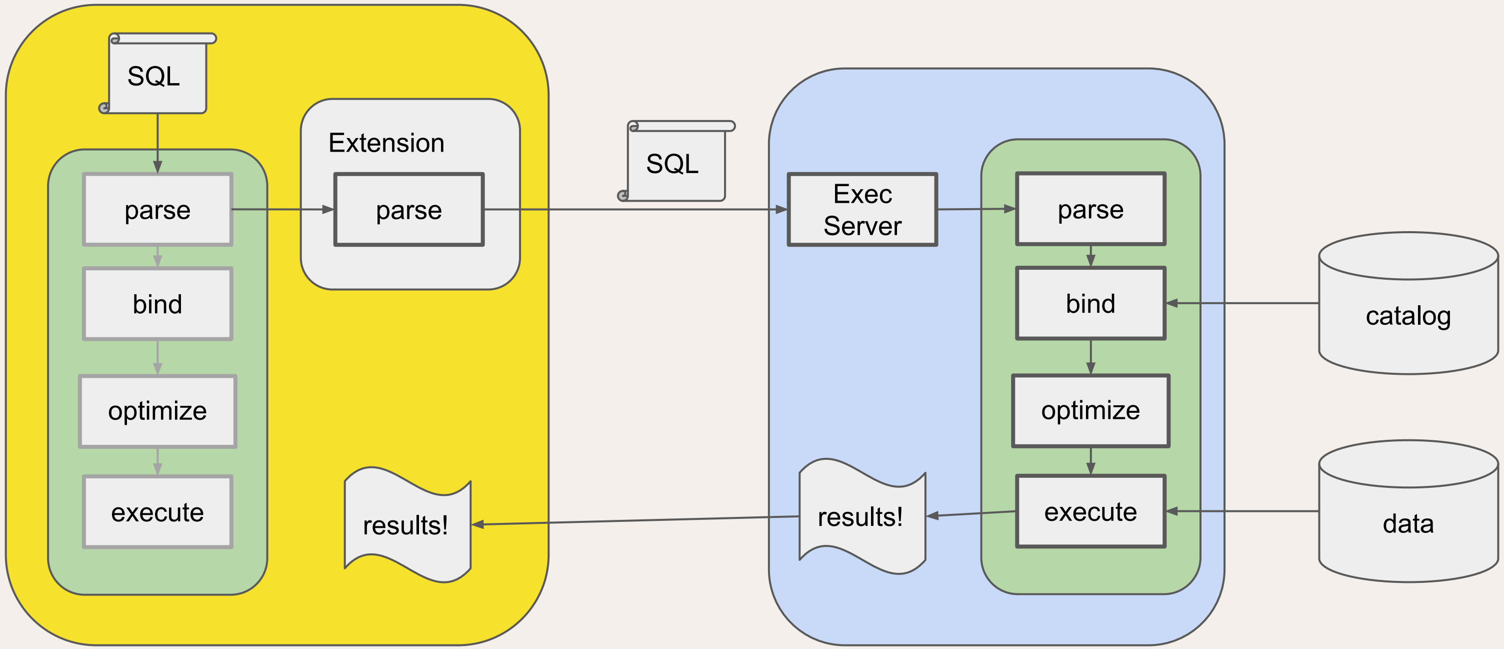
## Typical Database Execution



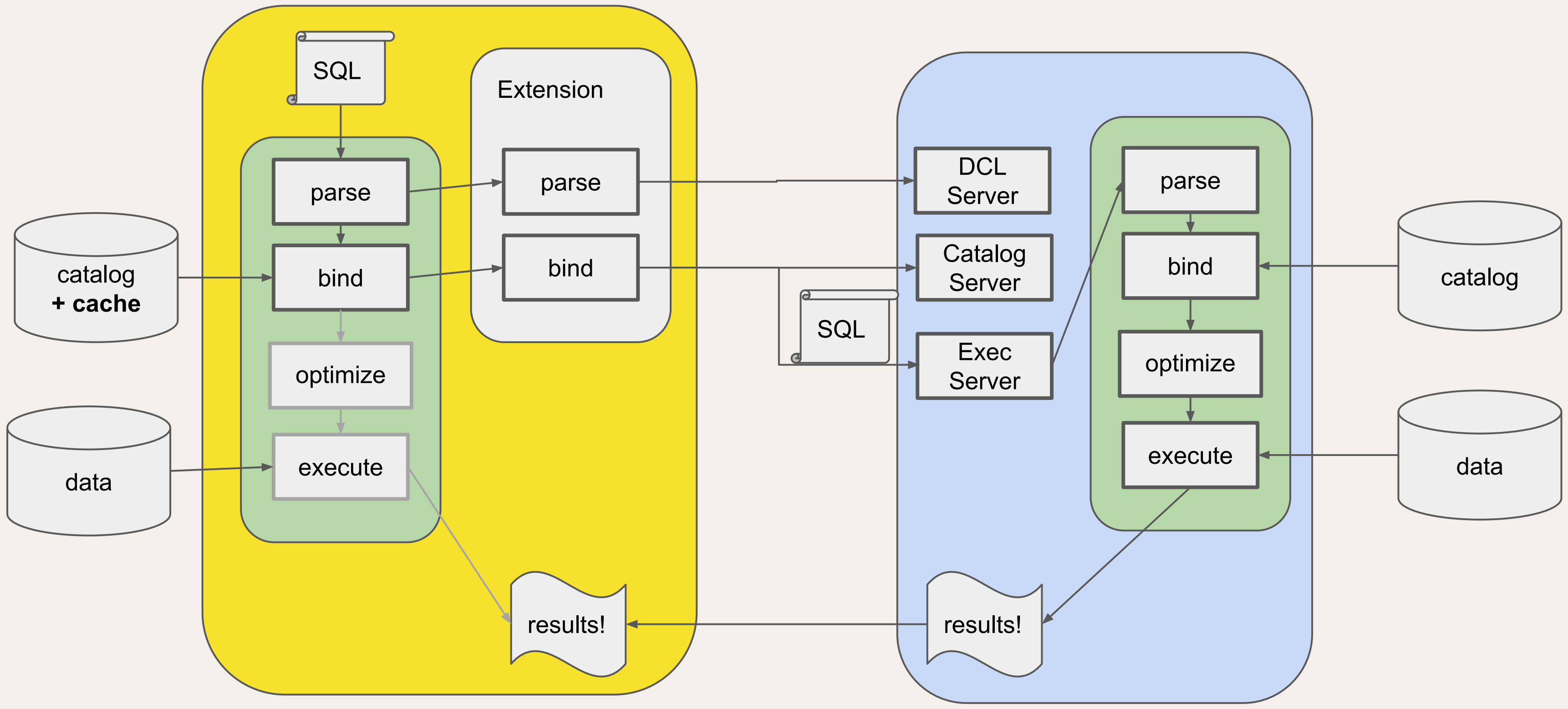
## DuckDB Execution



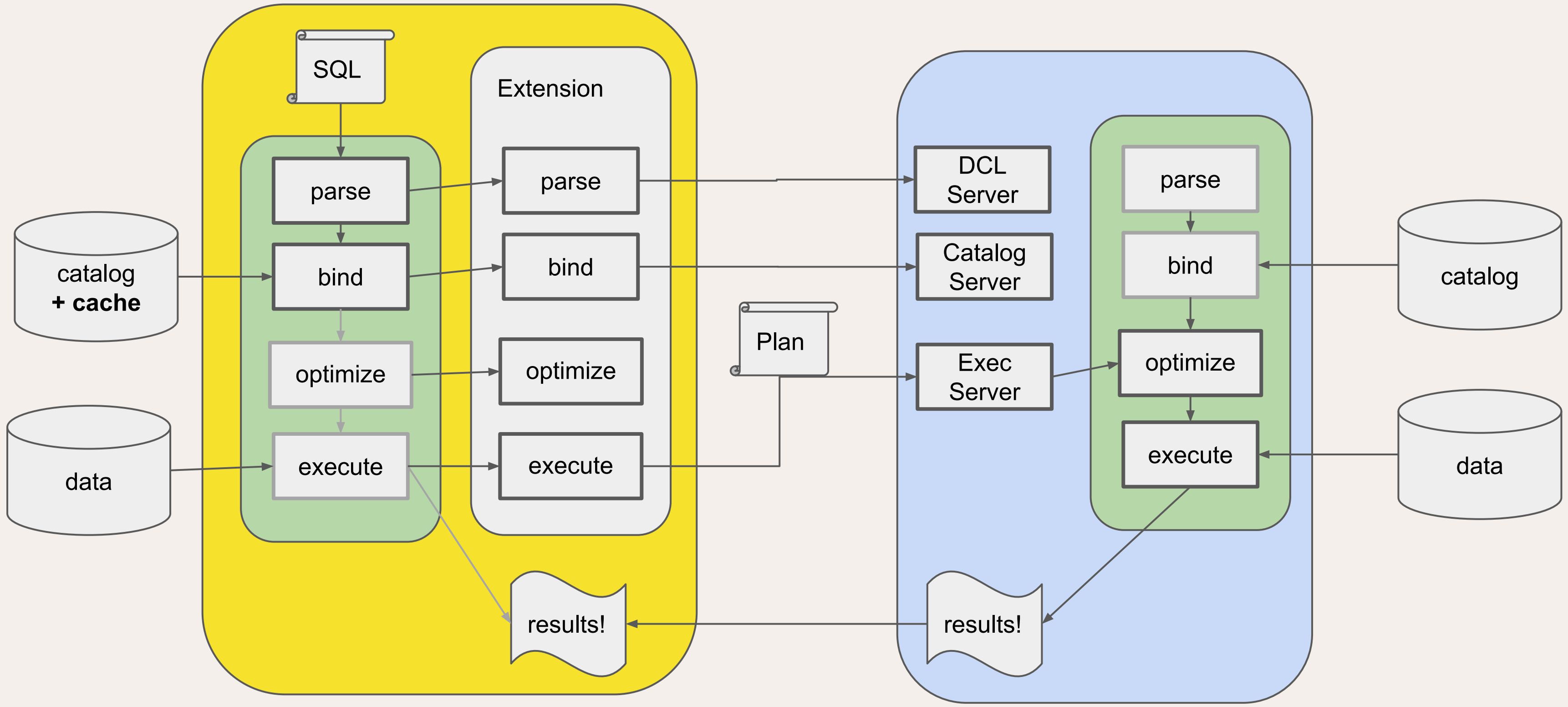
# Naive Hybrid 1: Forward immediately

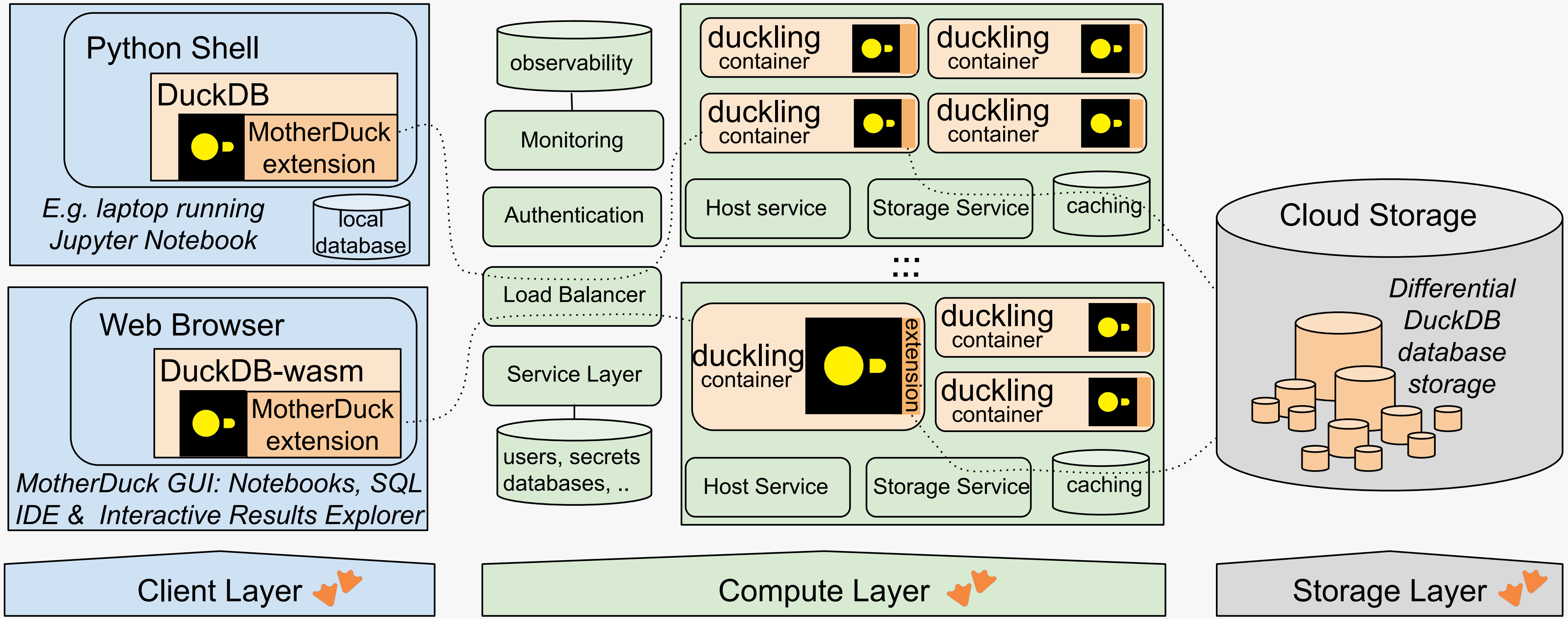


# Naive Hybrid 2: Handling Local Data



# Full Hybrid: Handling Blended Data

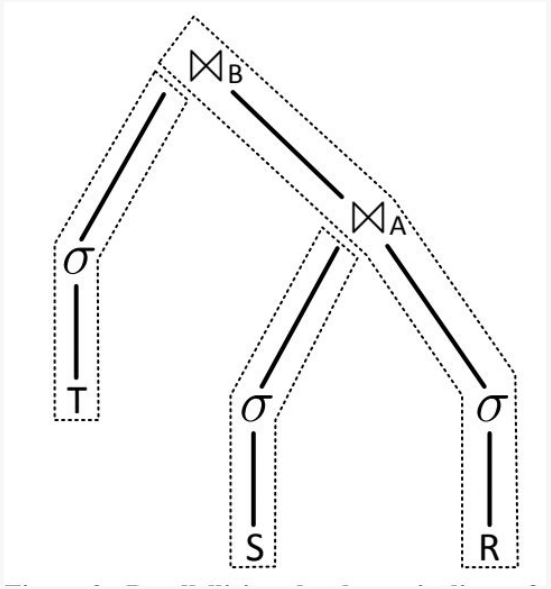




MotherDuck and the Very Hybrid Architecture

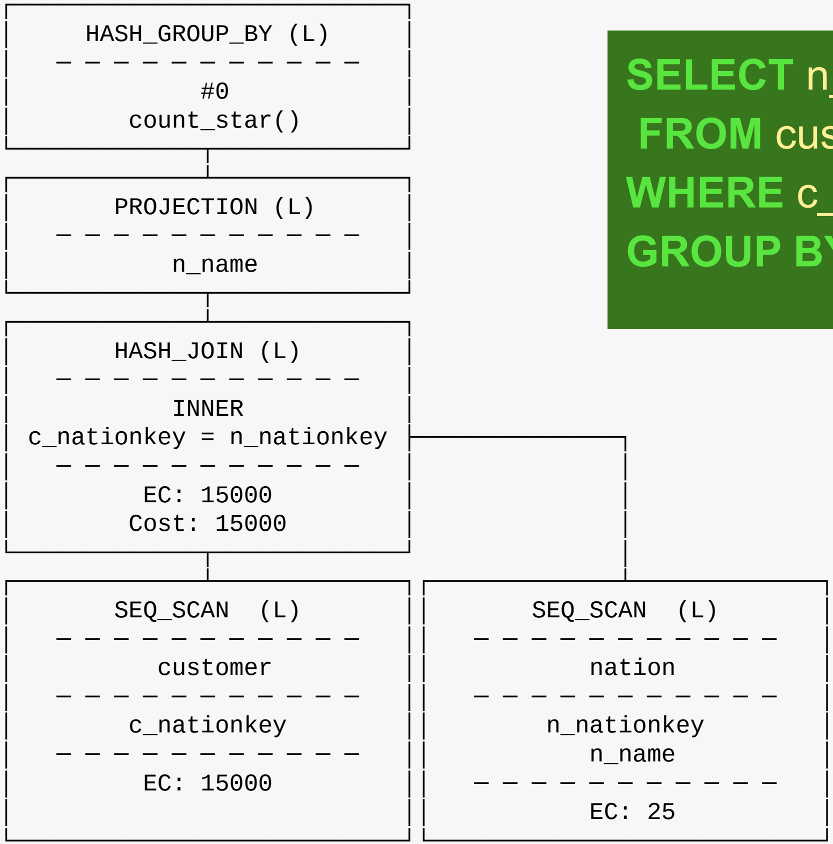
# Hybrid Query Optimization

1. Build “Pipelines” of operators
2. Pipelines terminating in local source run locally
3. Pipelines terminating in remote source run remotely
4. Insert bridges at pipeline sinks
5. If bridges transition local to remote or remote to local, add upload / download
6. Add “costs” of data movement to the bridge to determine whether to upload or download



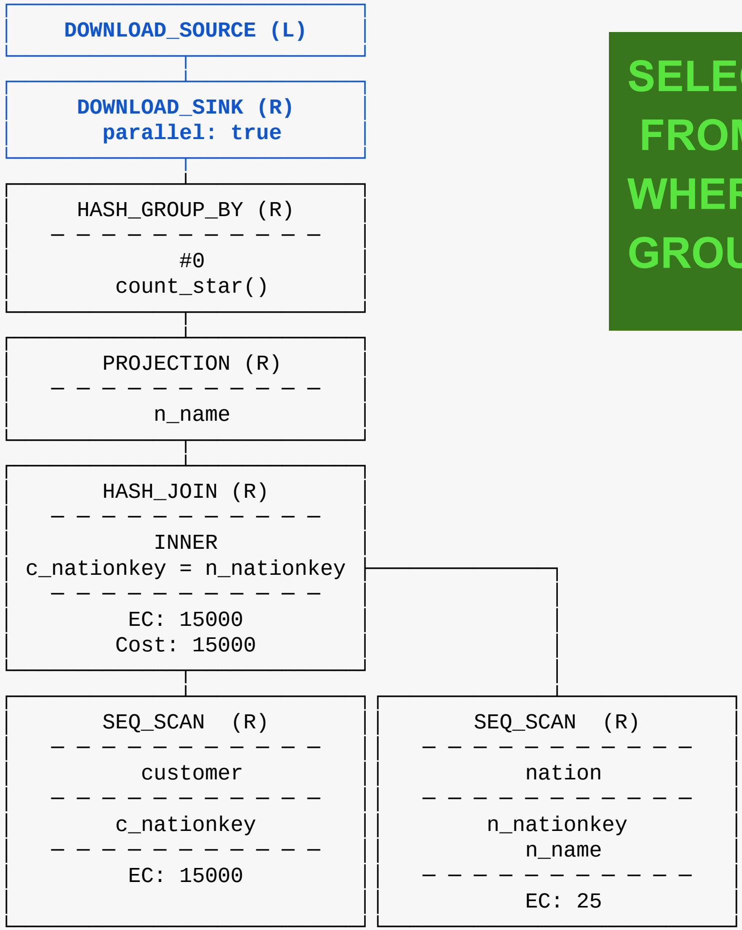
```
select *
from T,S,R
where T.id=S.id AND S.id=R.id
```

## Hybrid Query Plans: All Tables Local



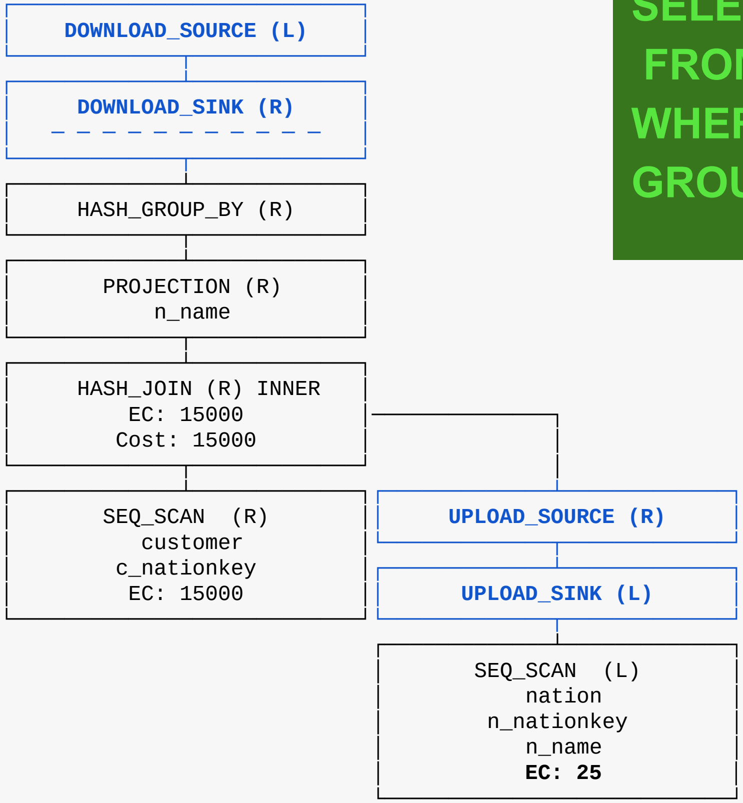
```
SELECT n_name, count(*)
 FROM customer, nation
WHERE c_nationkey = n_nationkey
GROUP BY n_name;
```

## Hybrid Query Plans: All Tables Remote



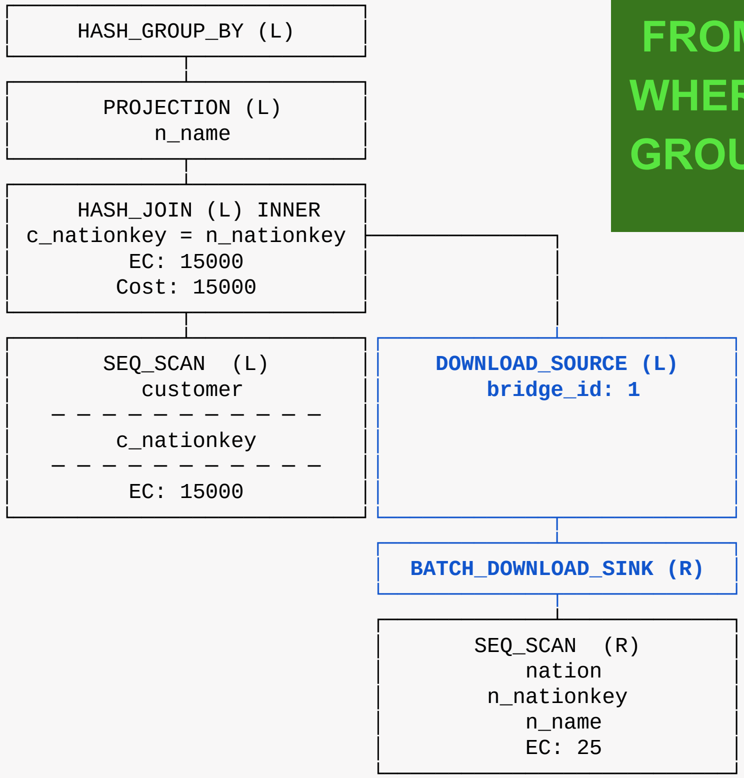
```
SELECT n_name, count(*)
 FROM customer, nation
WHERE c_nationkey = n_nationkey
GROUP BY n_name;
```

## Hybrid Query Plans: Small Local Table



```
SELECT n_name, count(*)
 FROM customer, local.nation
WHERE c_nationkey = n_nationkey
GROUP BY n_name;
```

## Hybrid Query Plans: Small Remote Table

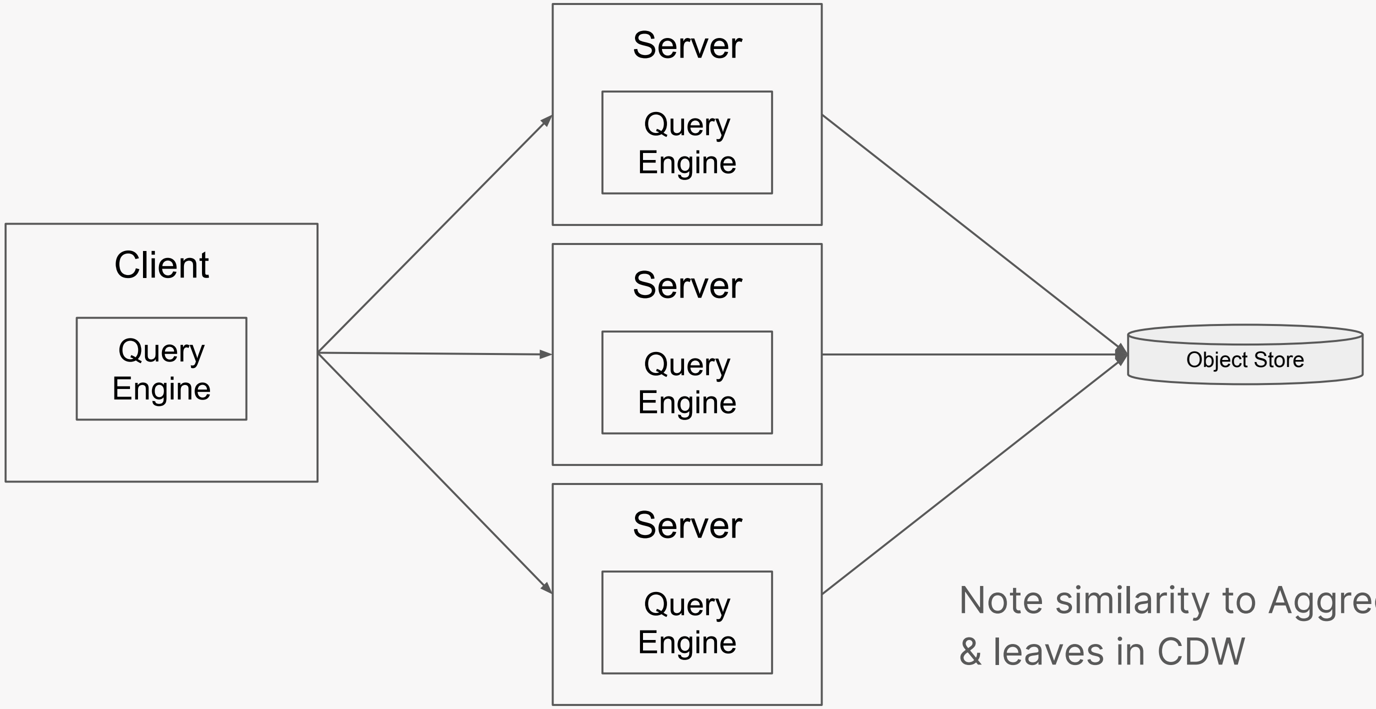


```
SELECT n_name, count(*)
 FROM local.customer, nation
WHERE c_nationkey = n_nationkey
GROUP BY n_name;
```

## Exploring Hybrid Topologies

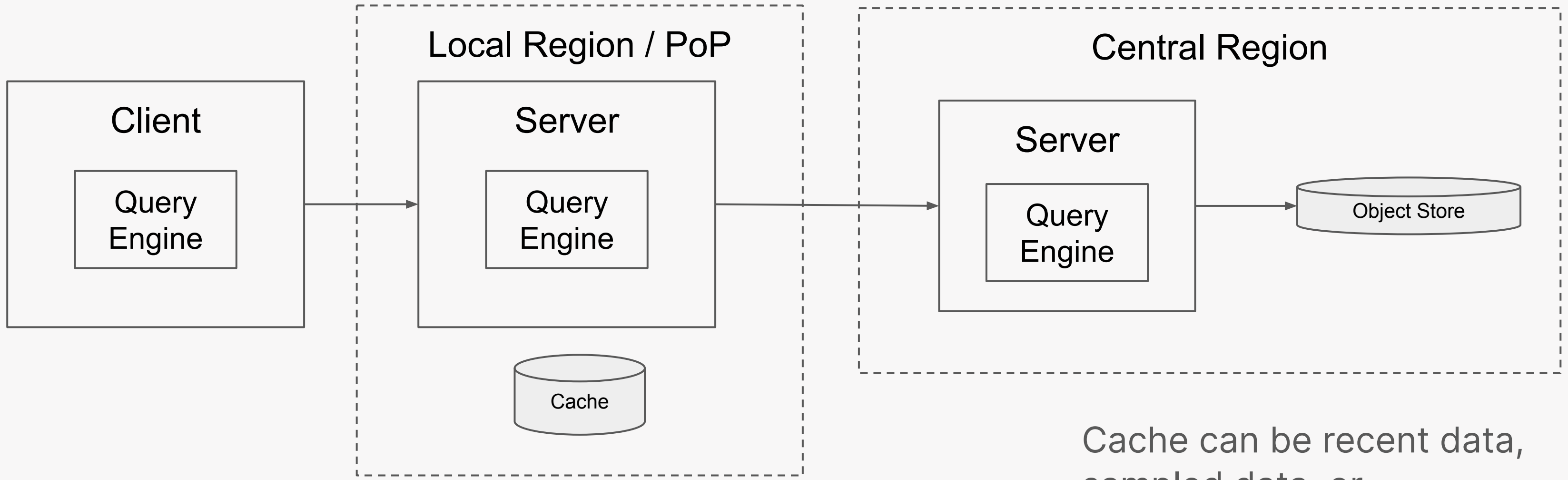
Future Work

## Parallel Scans



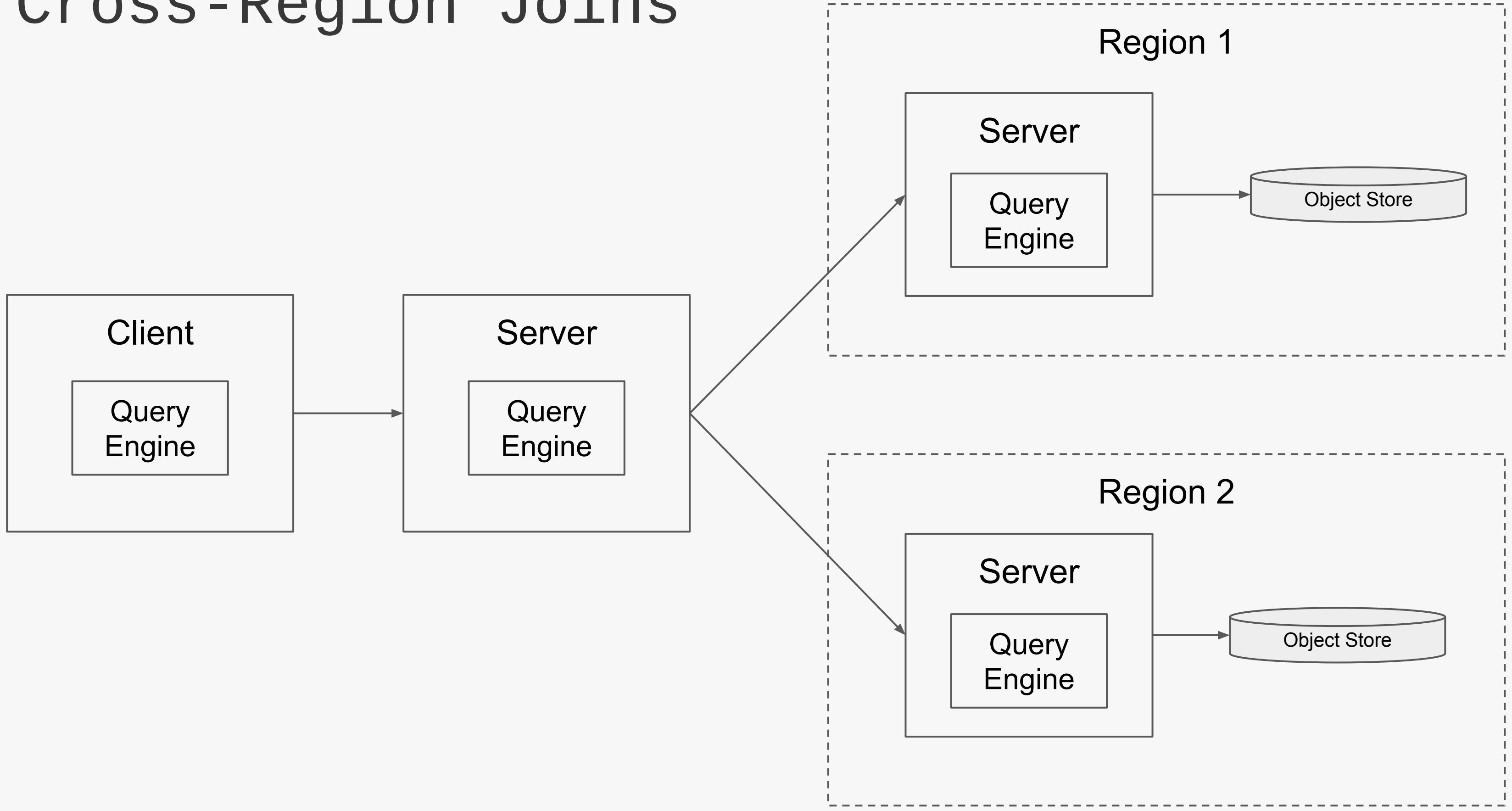
Note similarity to Aggregators & leaves in CDW

## Edge Cache

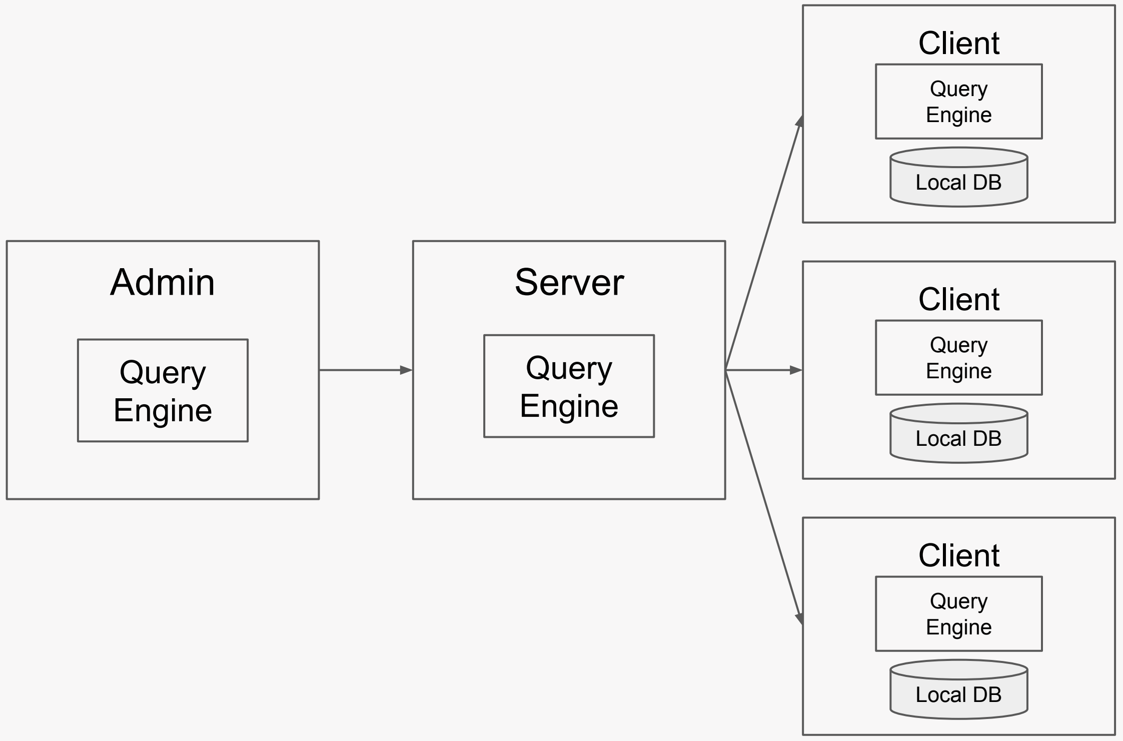


Cache can be recent data, sampled data, or region-fenced data

# Cross-Region Joins



## Reverse the polarity



“Find Security Threats”
Minimize Egress

## Conclusion

## Summary

Data Lakes becoming Data Warehouses

Query Engine + DW = Hybrid

Hybrid is useful in a bunch of different ways

Plenty of ways to push these ideas further

## Thank you!