
AWS
re:Invent
NOV. 28 – DEC. 2, 2022 | LAS VEGAS, NV

ANT328

Build transactional data lakes using open-table formats in Amazon Athena

Vijay Jain

Senior Product Manager, Athena
Amazon Web Services

Jack Ye

Sr. Software Development Engineer
Amazon Web Services

Agenda

Customer needs

Serverless Amazon Athena for analytics

Federated and native connectors with Athena

File formats and table formats with Athena

Transactional data lake use cases

Transactions in Athena – Deep Dive

Summary

Voice of customer

What are we hearing from customers?

"Managing a data pipeline is difficult and costly."

"How do we keep data consistent between our data lake and data warehouse?"

"We want to reduce the time it takes in ETL/ELT to get data ready for analytics."

"I wish to focus on innovating with data, not on maintaining and administering a data lake."

"My BI team needs access to the same data my data science team is looking at with no time lag."

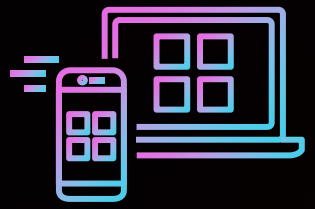
"How do I reduce bugs in ETL and improve data quality for our data analysts?"

"It's difficult to enforce governance policies consistently."

"How do I ensure data governance policies are enforced consistently between my data warehouse and data lake?"

Common themes

Growing exponentially



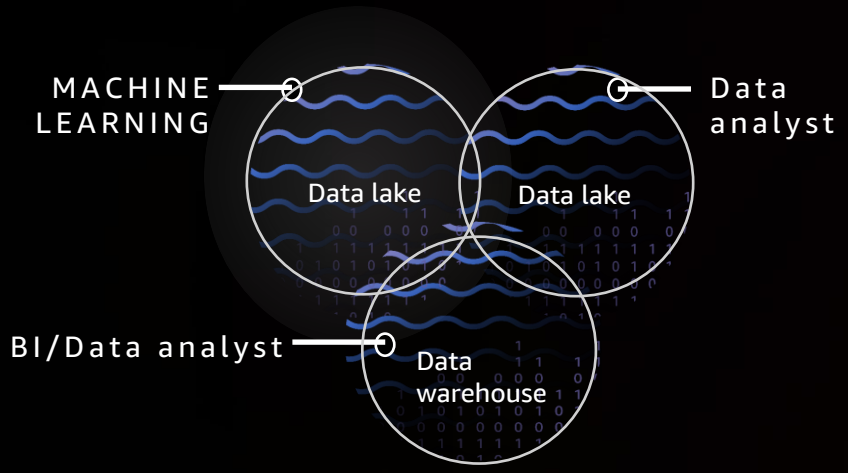
From new sources

Increasingly diverse formats

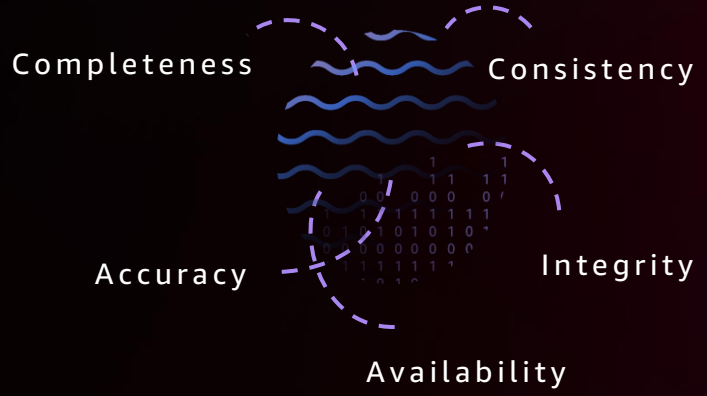
Used by many internal roles

Analytics and machine learning

Business needs



Faster time to analytics across different systems/end users



Data quality

Data privacy and compliance

Athena for data lake analytics

Athena

QUERY DATA IN AMAZON S3 USING SQL

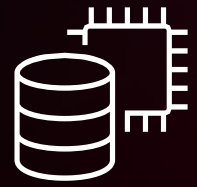
Easy to get started

Serverless, no setup

Near instant startup

Interactive analytics

Run queries across Amazon S3 and ***25+*** data sources



Open and flexible

Based on open-source framework, optimized for AWS

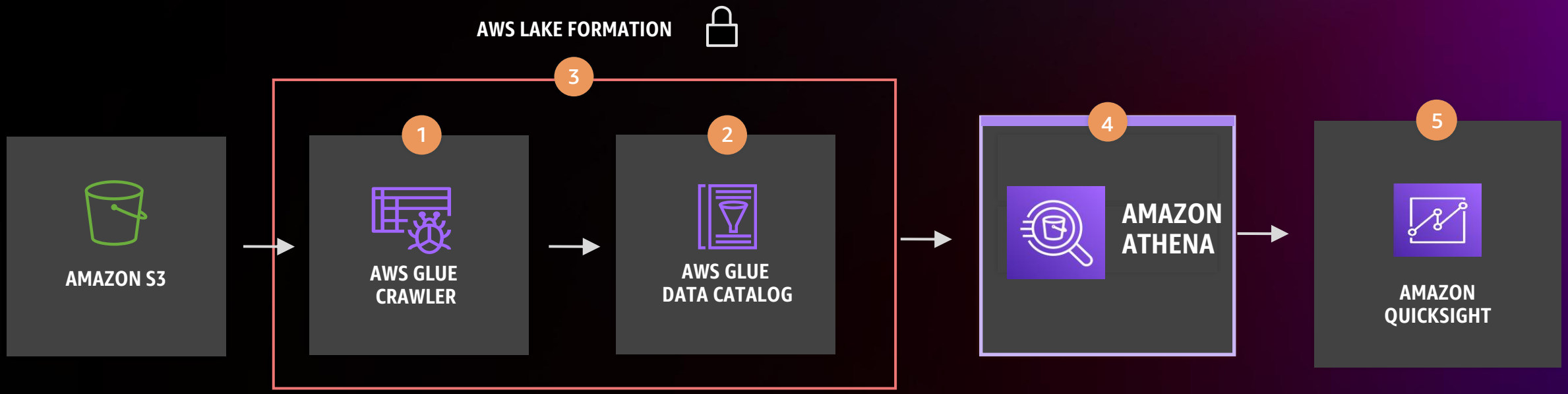
Open-table format
Open-file format

Cost effective

Pay only for what you use

Reduce total cost of ownership

Use case: Serverless data lake and analytics

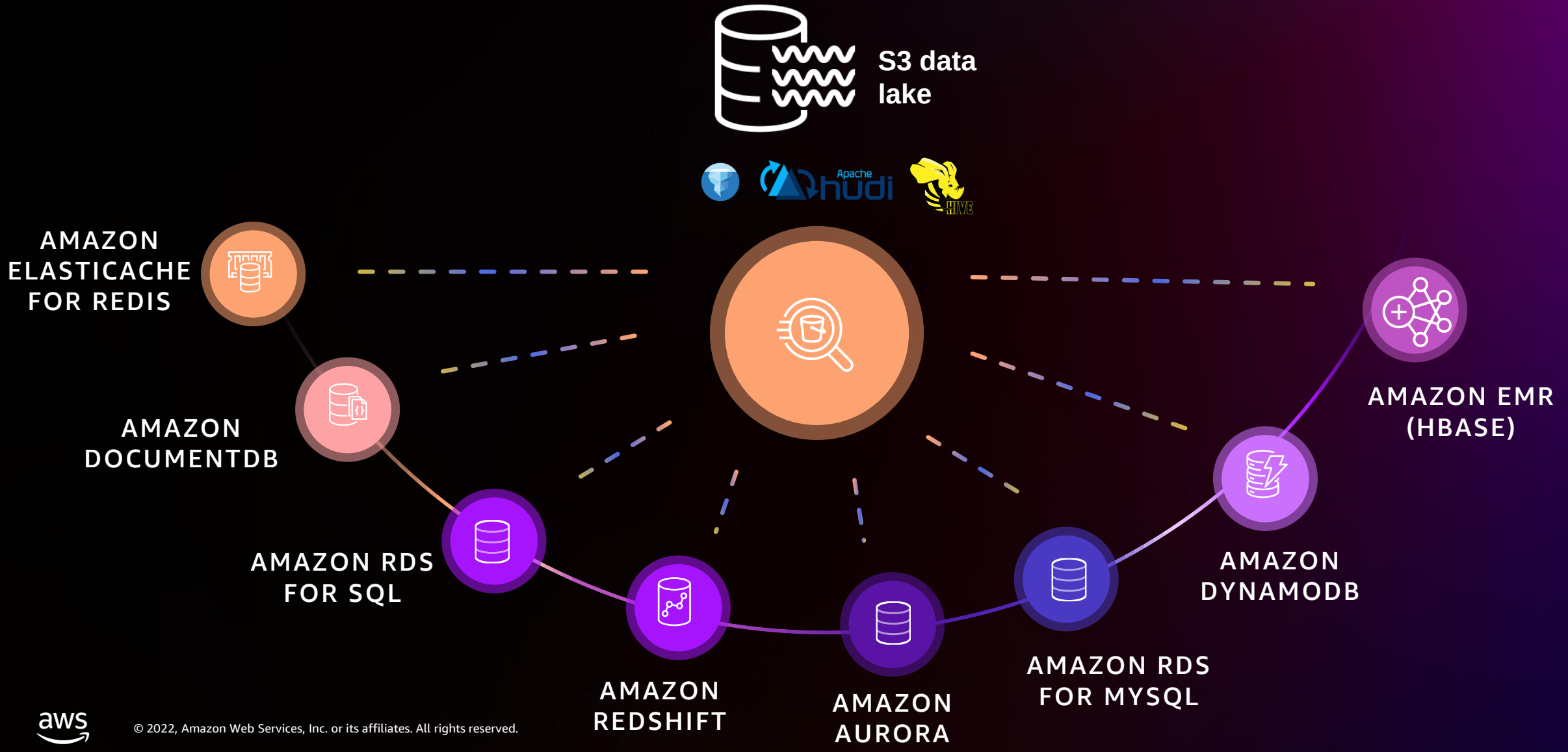


1. Crawlers scan your data sets and populate the Data Catalog
2. The Data Catalog serves as a central metadata repository
3. Once cataloged, apply security and permissions
4. Query your data immediately for analytics
5. Visualize with QuickSight

https://github.com/aws-samples/serverless-data-analytics

Connectors with Athena

Athena – Connectors



Transactions with Athena

Table and file formats supported in Athena

File formats

- Raw formats: Service logs, machine logs
- Semi-structured formats: CSV, JSON
- Analytics formats: Parquet, Avro, ORC

Table formats

- Apache Hive
- Apache Iceberg
- Apache Hudi

Transactions with Athena – use cases

ETL/ELT

Many organizations prefer SQL for data preparation. With transaction support in Athena, customers manage ETL/ELT operations in Athena at scale in an efficient and cost-effective way.

At-scale ACID transactions

No data duplication

Better data quality

Faster time to analytics

Data pipeline management

Data pipelines are set up to convert source data to an analytics-ready state by copying data, moving it from an on-premises location into the cloud, and reformatting it or joining it with other data sources. With support for transactions, Athena can now simplify data pipeline management.

Robust and agile pipeline management

No need for data movement

Optimize data for end use case

Reduce costs

GDPR/CCPA

Per GDPR/CCPA, a company must remove personal data from its storage systems upon consumer's request. This is difficult when dealing with petabytes of data in data lakes and ensuring data consistency across different systems. With Athena and Iceberg, all personal data can be managed effectively in an Iceberg-based S3 data lake.

No data duplication for sharing

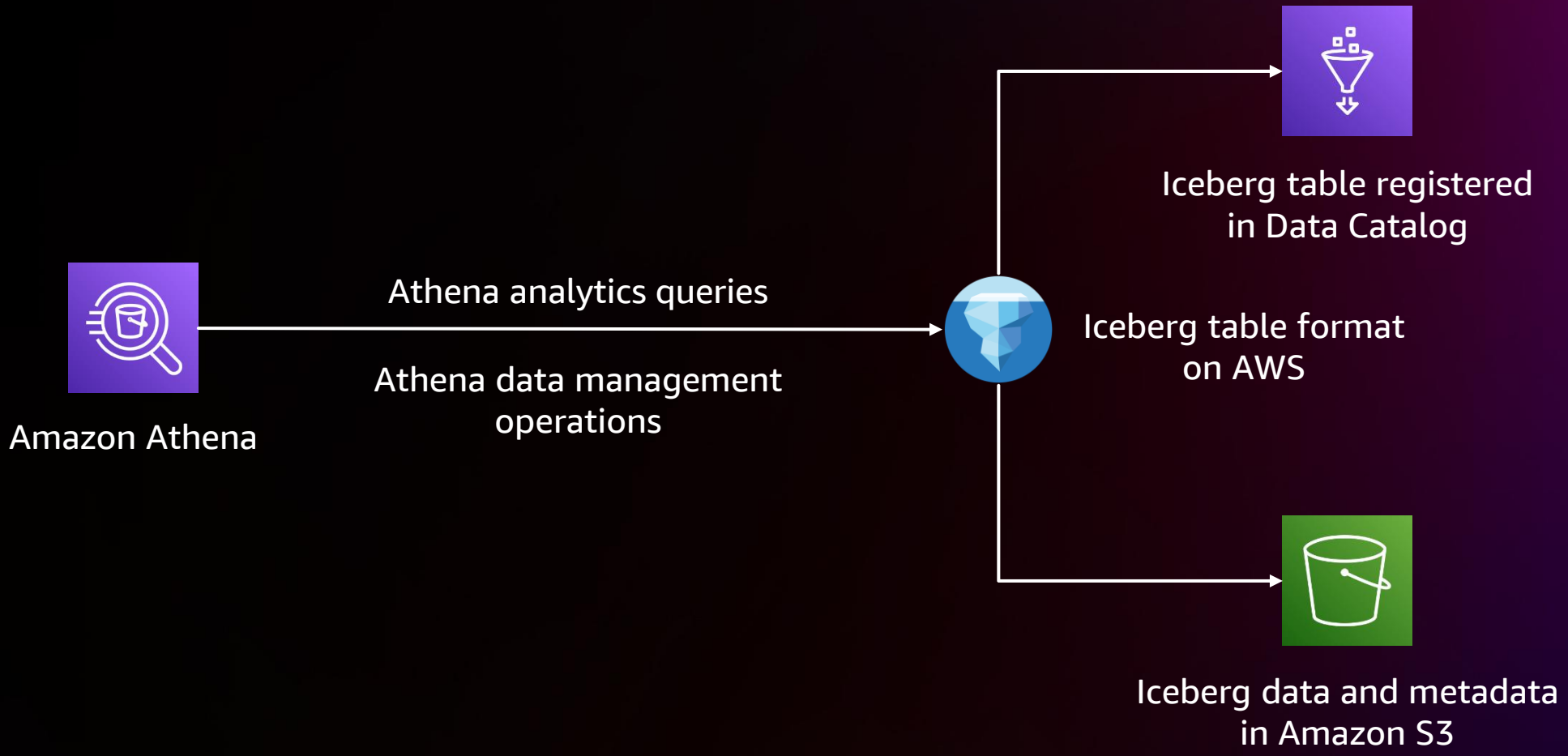
Transaction logs for history of changes

Permanent deletes

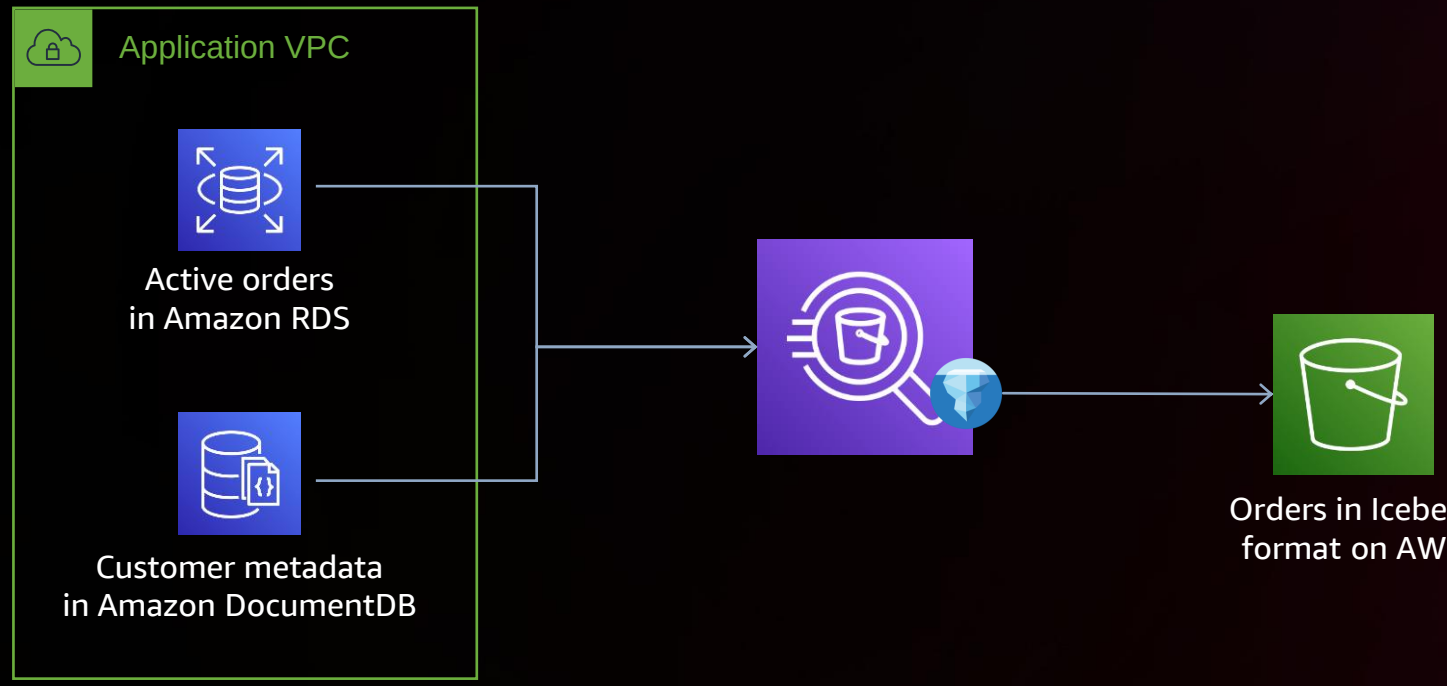
Effective data governance with fine-grained access controls

Transactions with Athena – deep dive

Apache Iceberg data lake using Athena

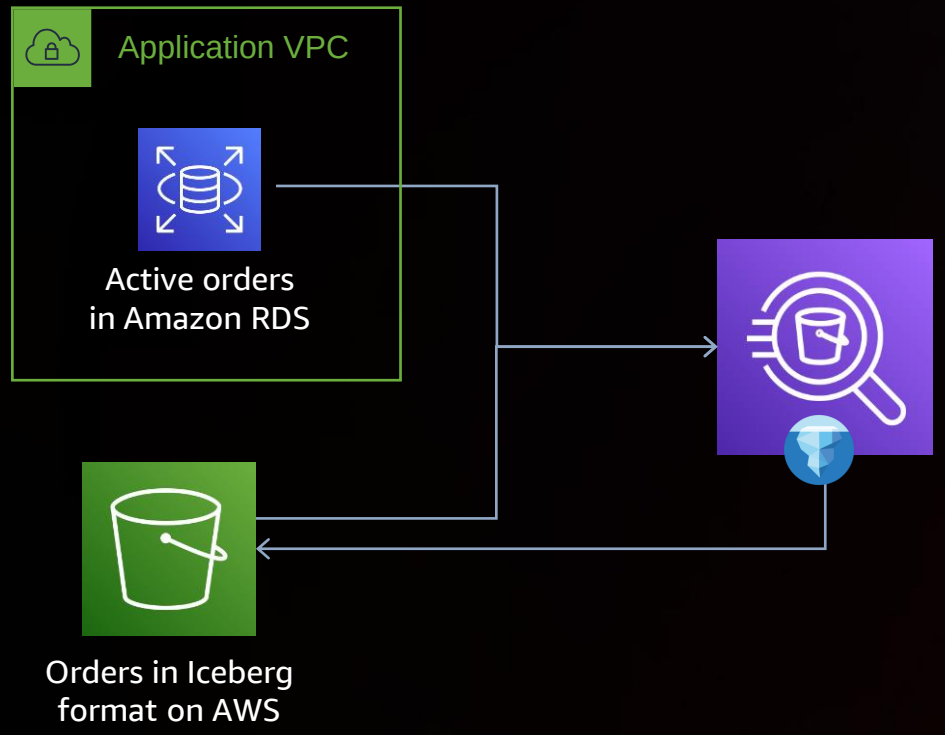


Loading data in Iceberg format



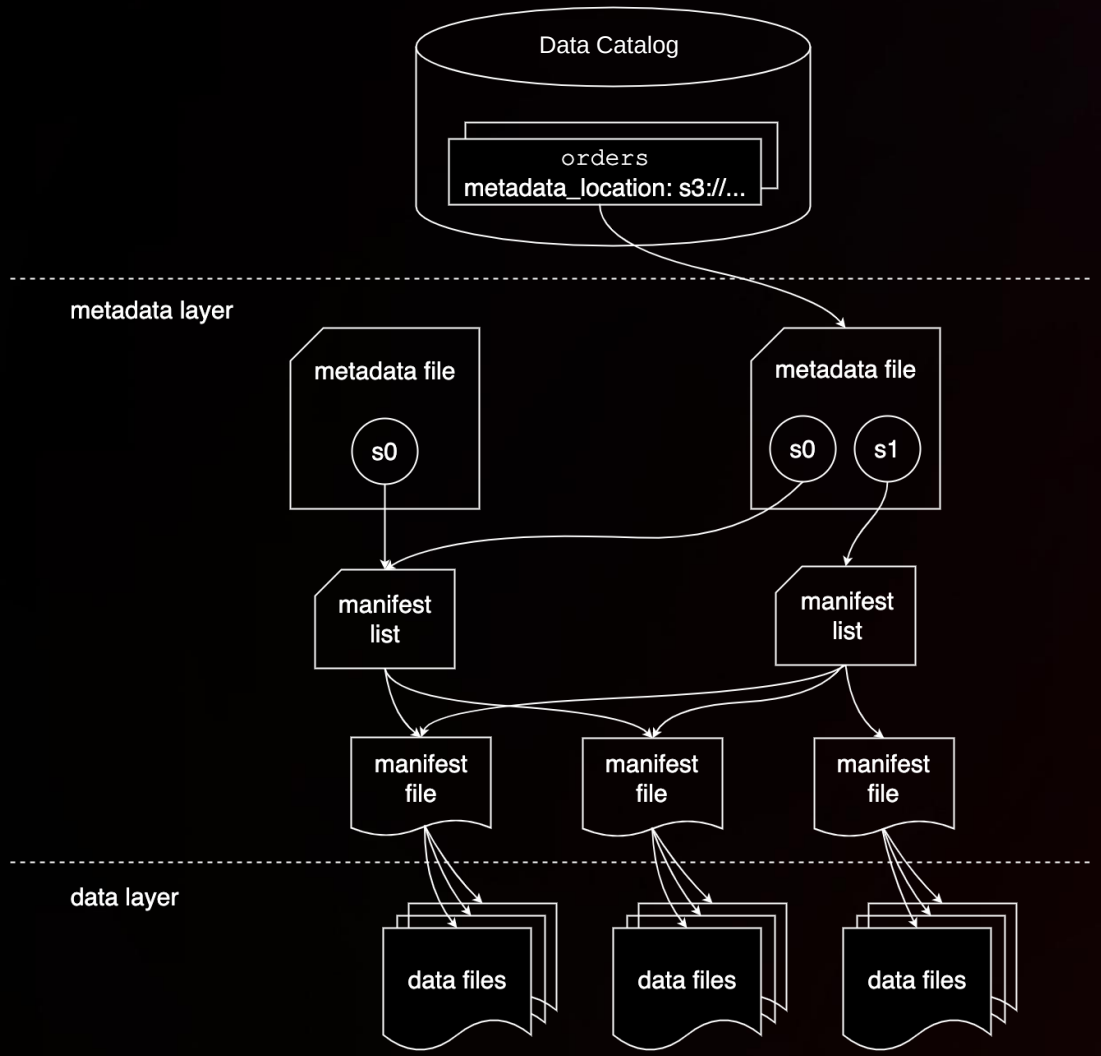

```
CREATE TABLE orders
WITH (
  format = ‘parquet',
  location = 's3://…’,
     table_type = 'ICEBERG’,
  partitioning = ARRAY[’day(ship_date)’]
)
AS SELECT *
FROM rds.orders.active_orders ao
  LEFT JOIN docdb.customer.metadata cm
  ON ao.customer_id = cm.id
WHERE status <> 'pending’
```

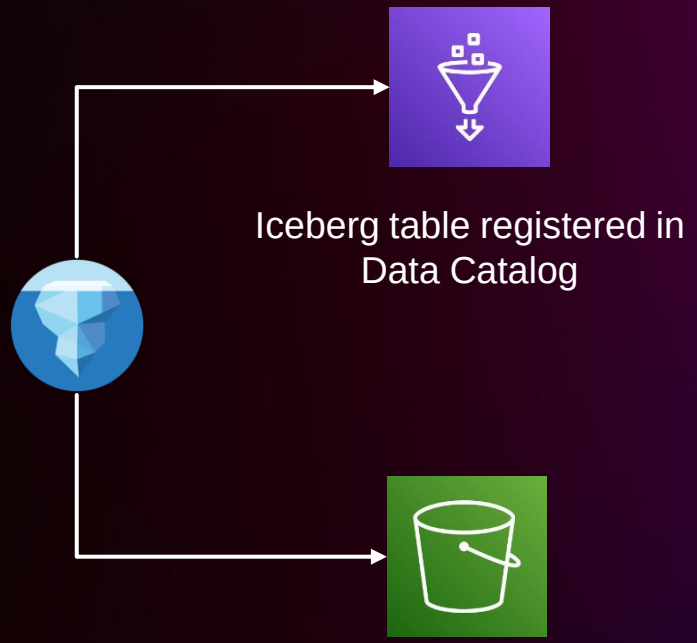
Merge changelog




```
MERGE INTO orders o USING rds.orders.active_orders ao
    ON (o.order_id = ao.order_id)
    WHEN MATCHED AND ao.status = 'Cancelled'
        THEN DELETE
    WHEN MATCHED
        THEN UPDATE
            SET ship_date = ao.ship_date
    WHEN NOT MATCHED
        THEN INSERT (order_id, customer_id, ship_date)
            VALUES(ao.order_id, ao.customer_id, ao.ship_date)
```

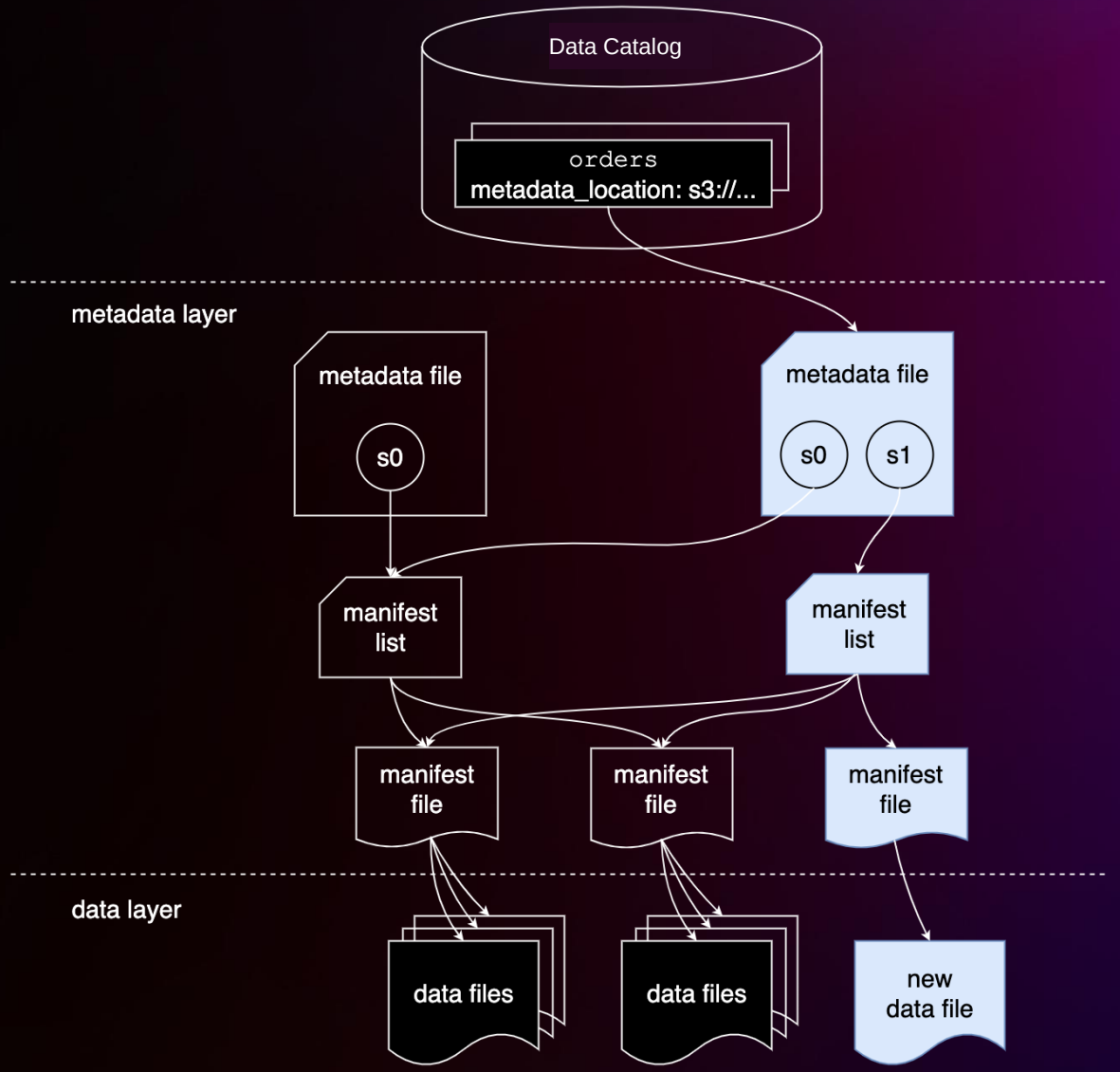
Dive deeper into the format





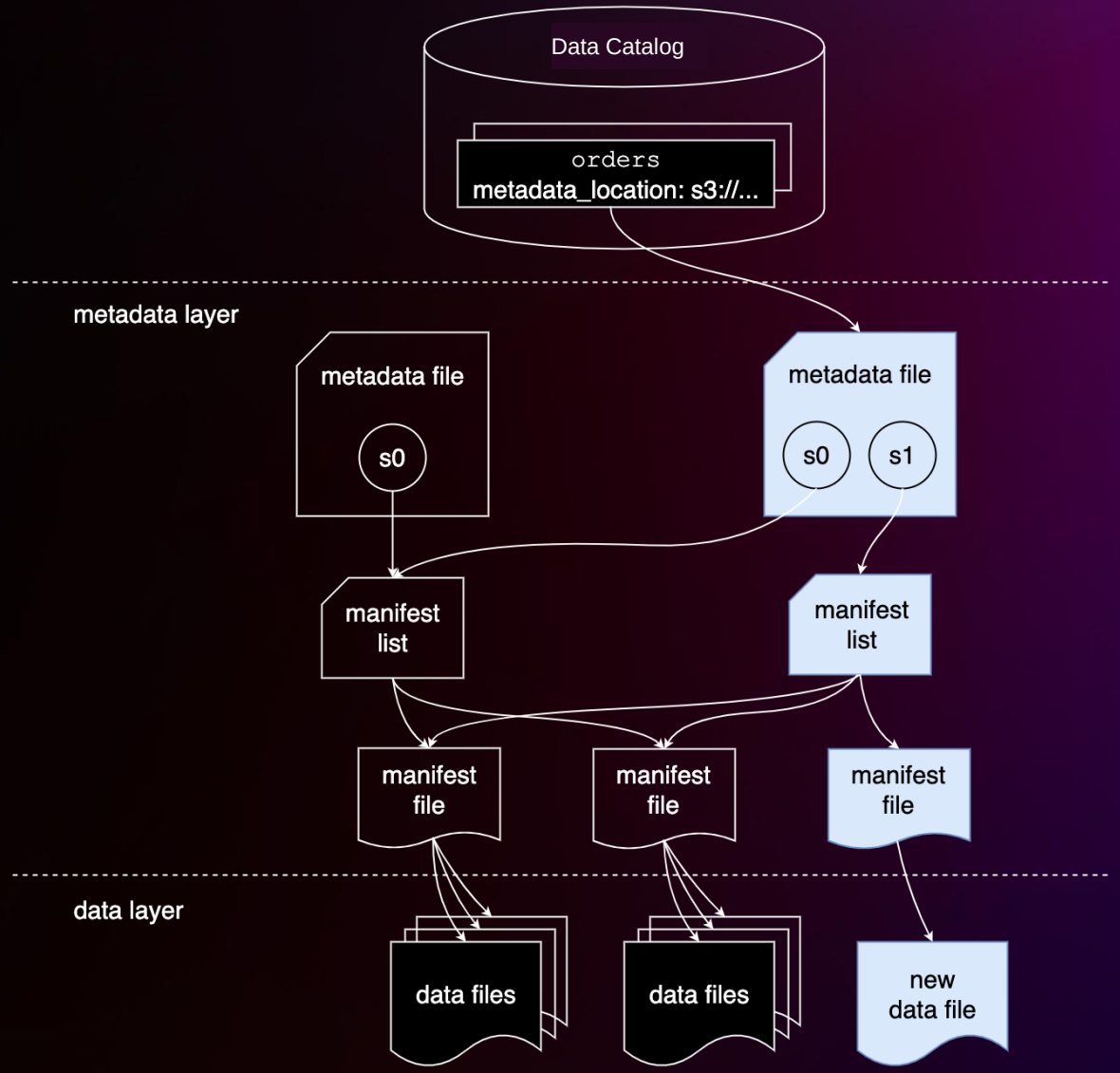
INSERT transaction

```
INSERT INTO orders
AS SELECT *
FROM rds.orders.active_orders ao
  LEFT JOIN docdb.customer.metadata cm
  ON ao.customer_id = cm.id
WHERE status <> 'pending'
```



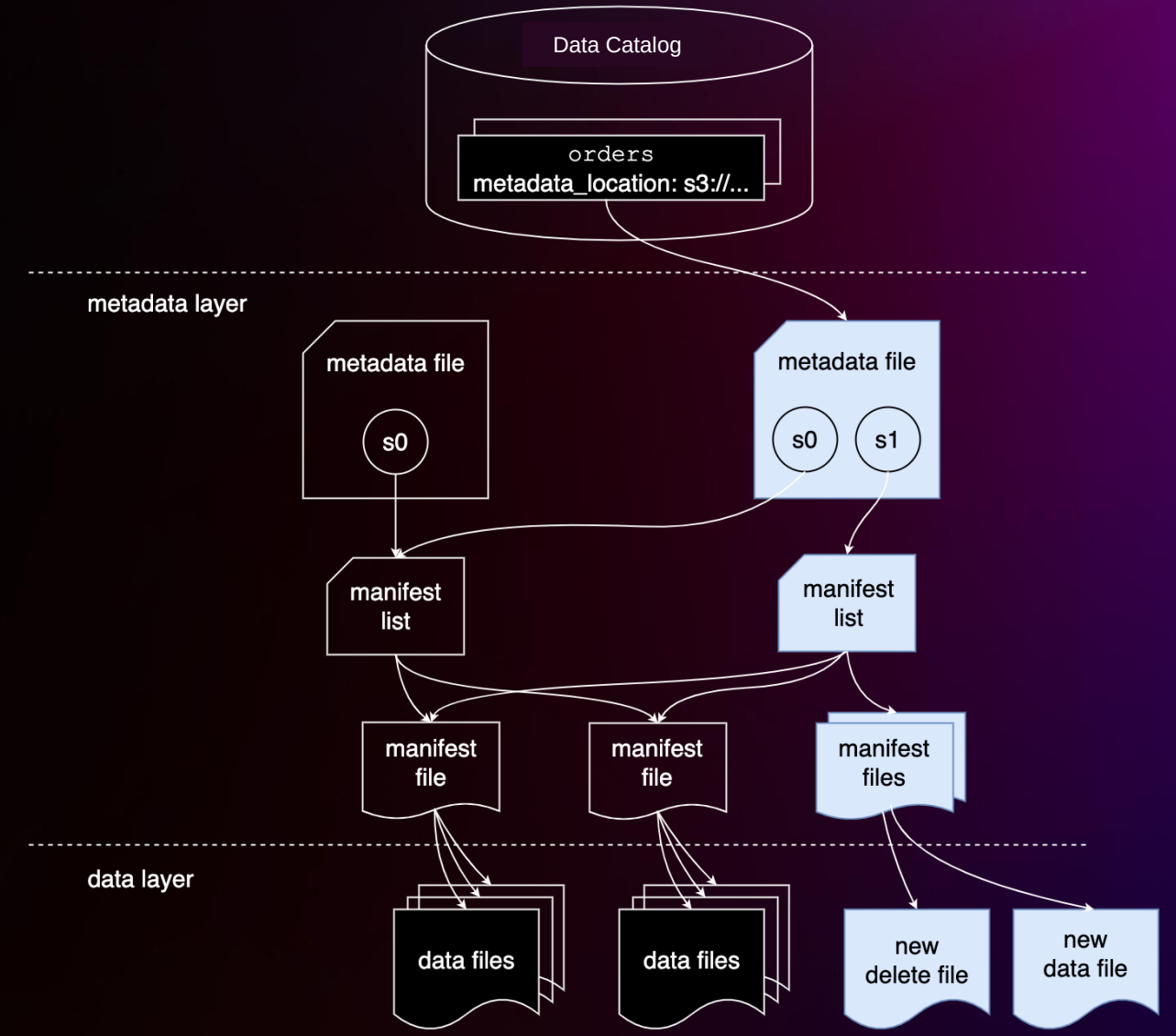
DELETE transaction

```
DELETE FROM orders
WHERE order_id IN (
  SELECT order_id FROM active_orders
  WHERE priority = 'LOW');
```

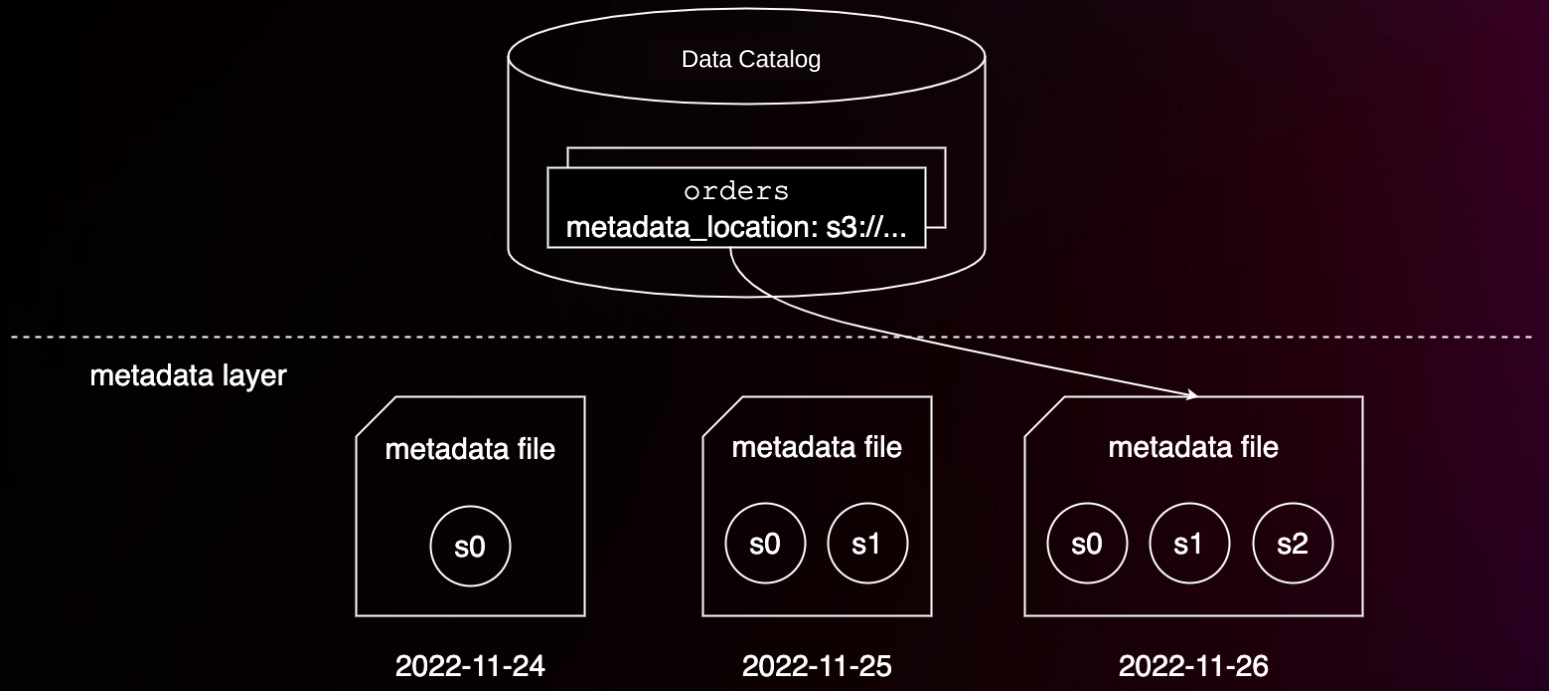


UPDATE transaction

```
UPDATE
  orders
SET
  status = 'OVERDUE'
WHERE
  ship_date IS NULL;
```



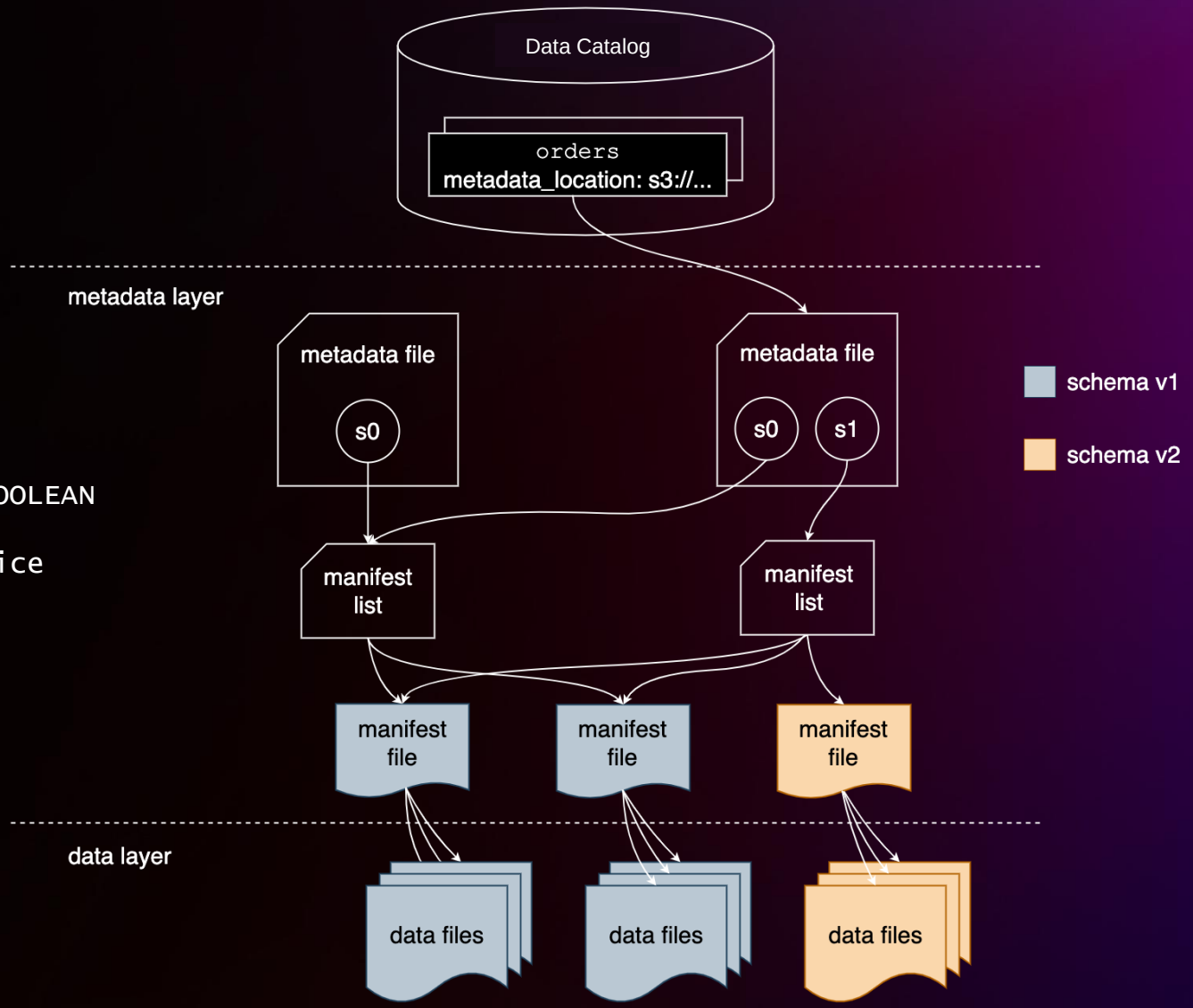
Time travel with transactions




```
SELECT * FROM orders FOR TIMESTAMP AS OF DATE '2022-11-25'

SELECT * FROM orders FOR VERSION AS OF 1
```

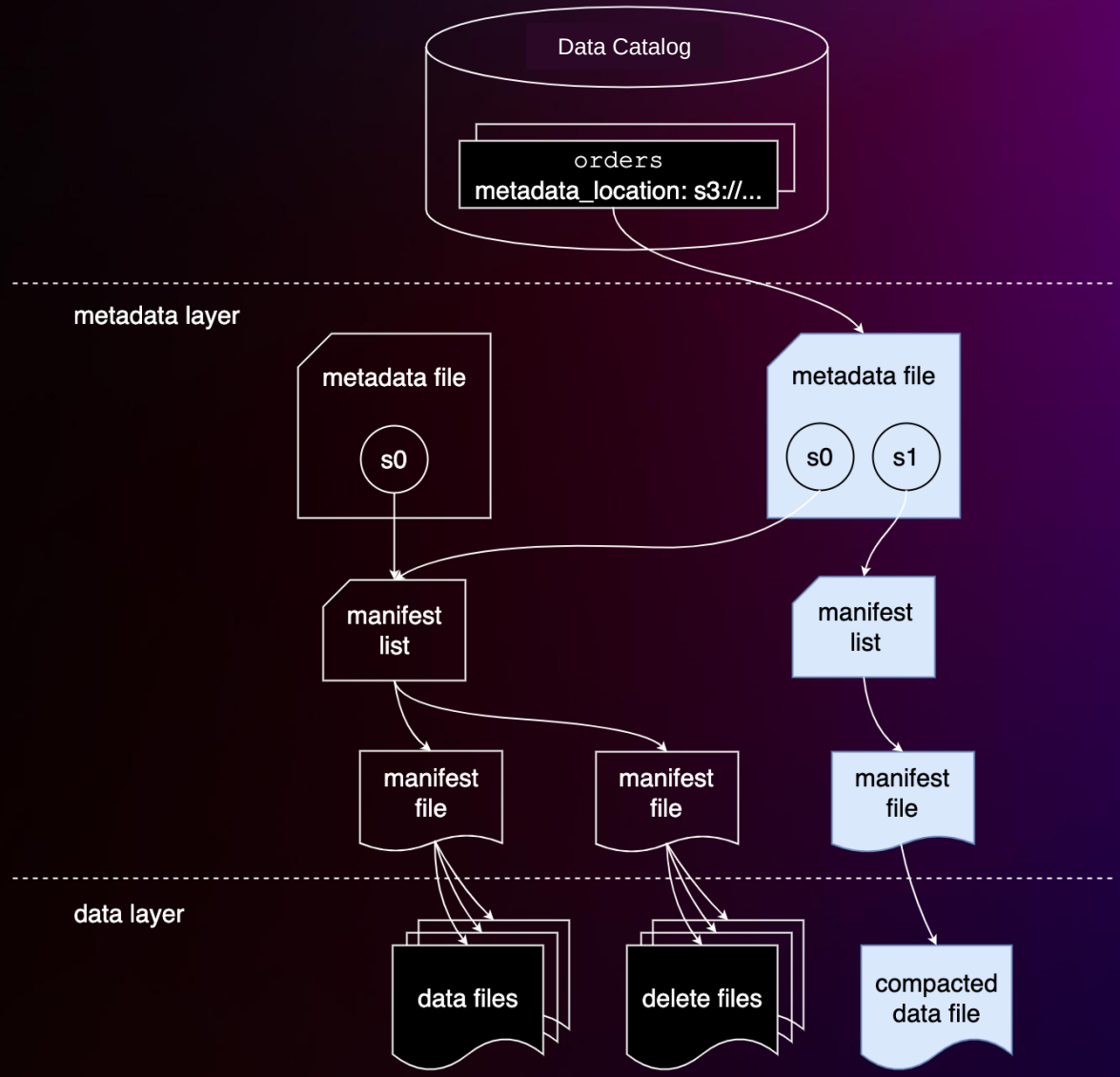
Schema evolution

```
ALTER TABLE orders ADD COLUMN is_prime BOOLEAN

ALTER TABLE orders CHANGE COLUMN item_price
  item_price_double DOUBLE
  COMMENT 'item price in double type'
  AFTER order_id
```



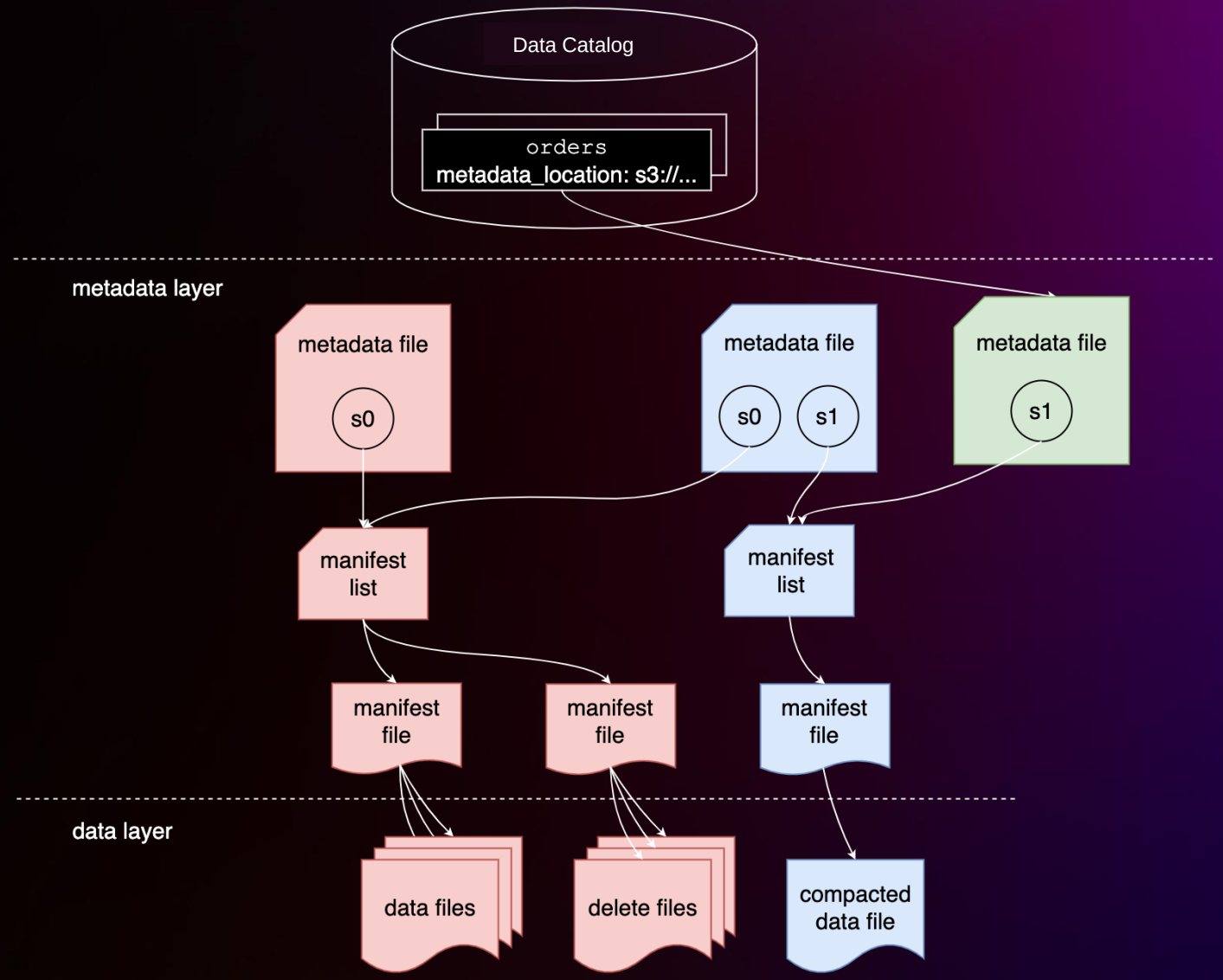
OPTIMIZE operation

```
ALTER TABLE orders SET TBLPROPERTIES (
  'optimize_rewrite_delete_file_threshold' = '2',
  'optimize_rewrite_data_file_threshold' = '10',
  'write_target_file_size_bytes' = '536870912'
)

OPTIMIZE orders REWRITE DATA USING BIN_PACK
WHERE ship_date = TIMESTAMP '2022-11-28'
```



Permanent deletes by vacuum

```
ALTER TABLE orders SET TBLPROPERTIES (
  ‘vacuum_min_snapshots_to_keep’ = ’1’,
  ‘vacuum_max_snapshot_age_seconds’ = ‘86400’
)


VACUUM table
```



Thank you!

Vijay Jain

Jack Ye

General Athena contact

athena-feedback@amazon.com

Please complete the session survey in the **mobile app**