08.10.2025

Apache Iceberg on AWS

David Greenshtein
Senior Solutions Architect – Analytics, AWS

Agenda

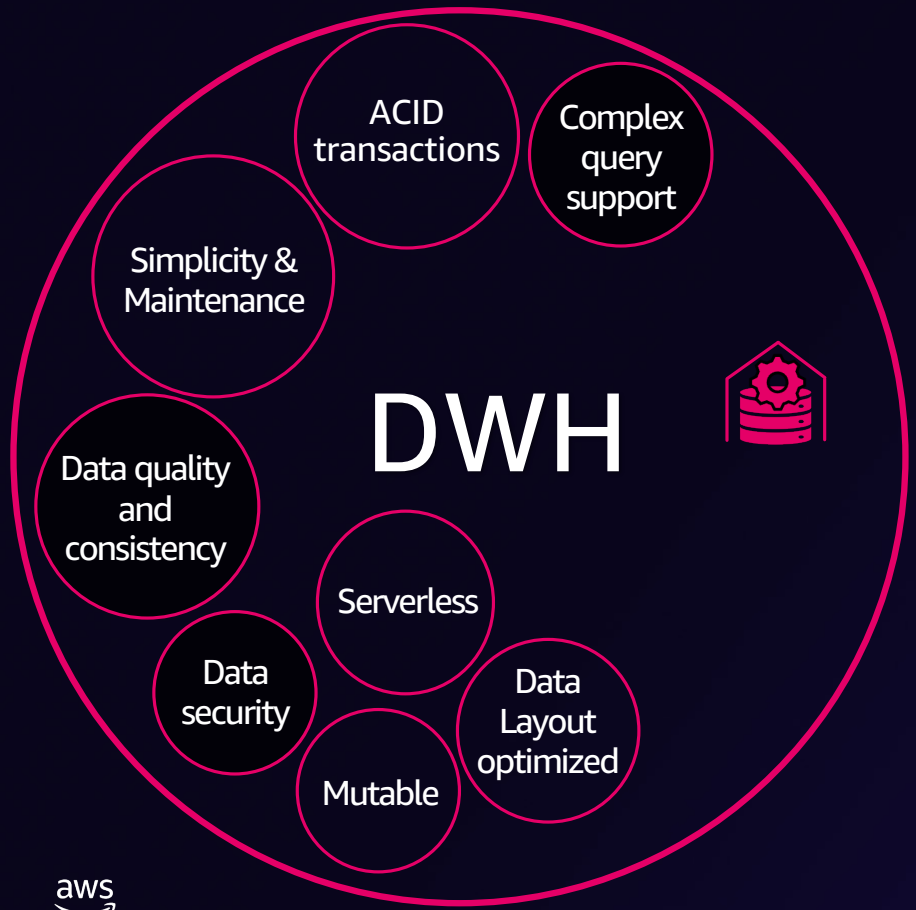
Introduction to Open Table Formats

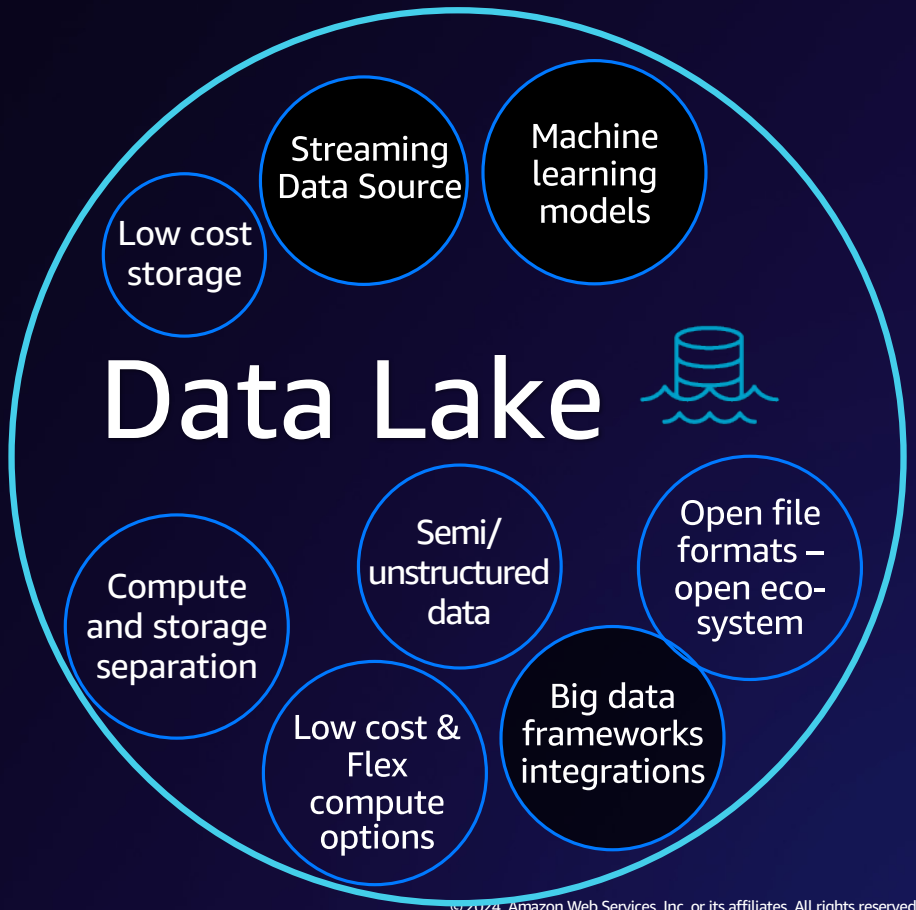
Apache Iceberg

- Overview
- Feature Highlights
- Table Strategy and table Maintenance

Open Table Formats

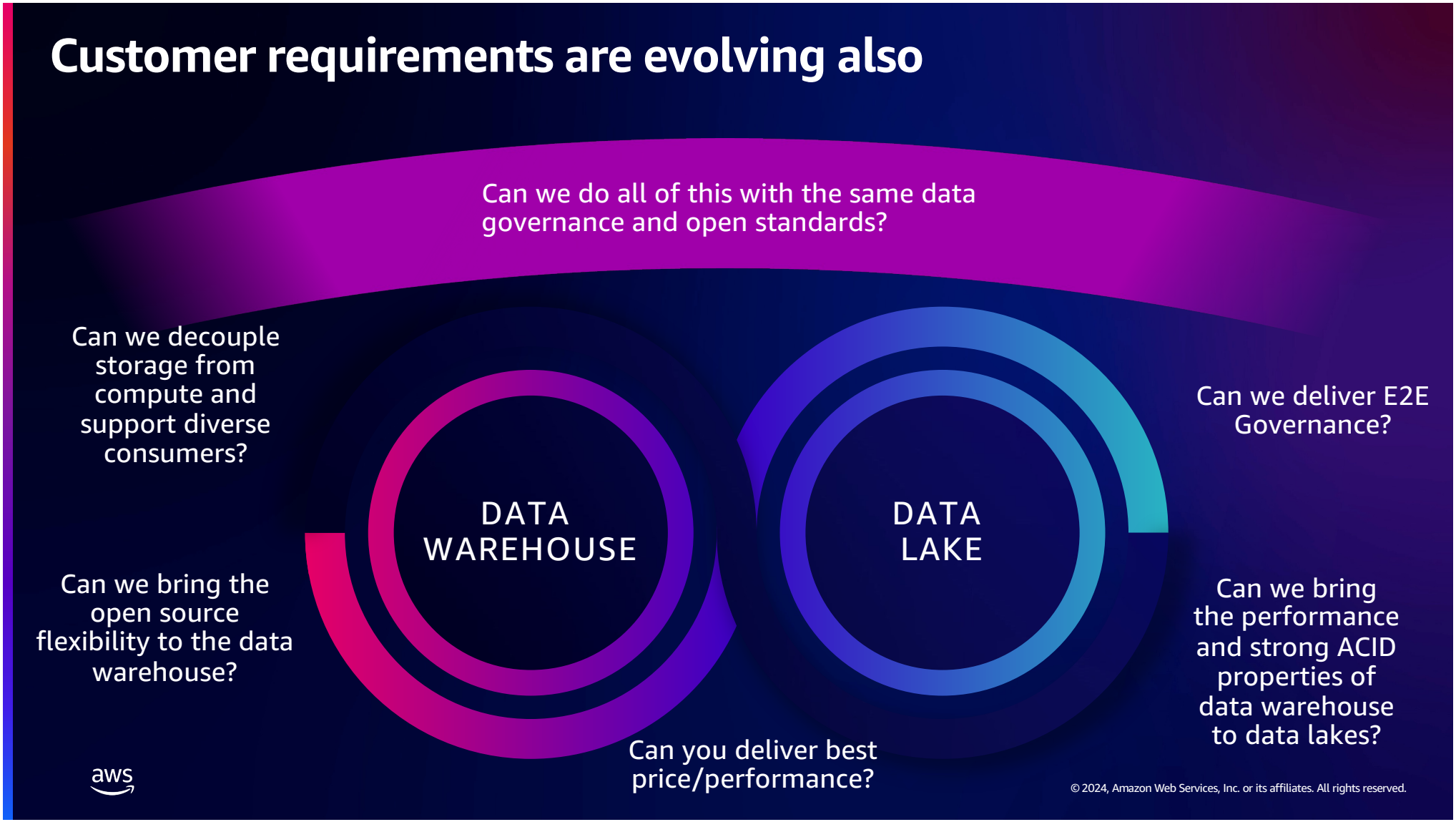
Platforms are evolving



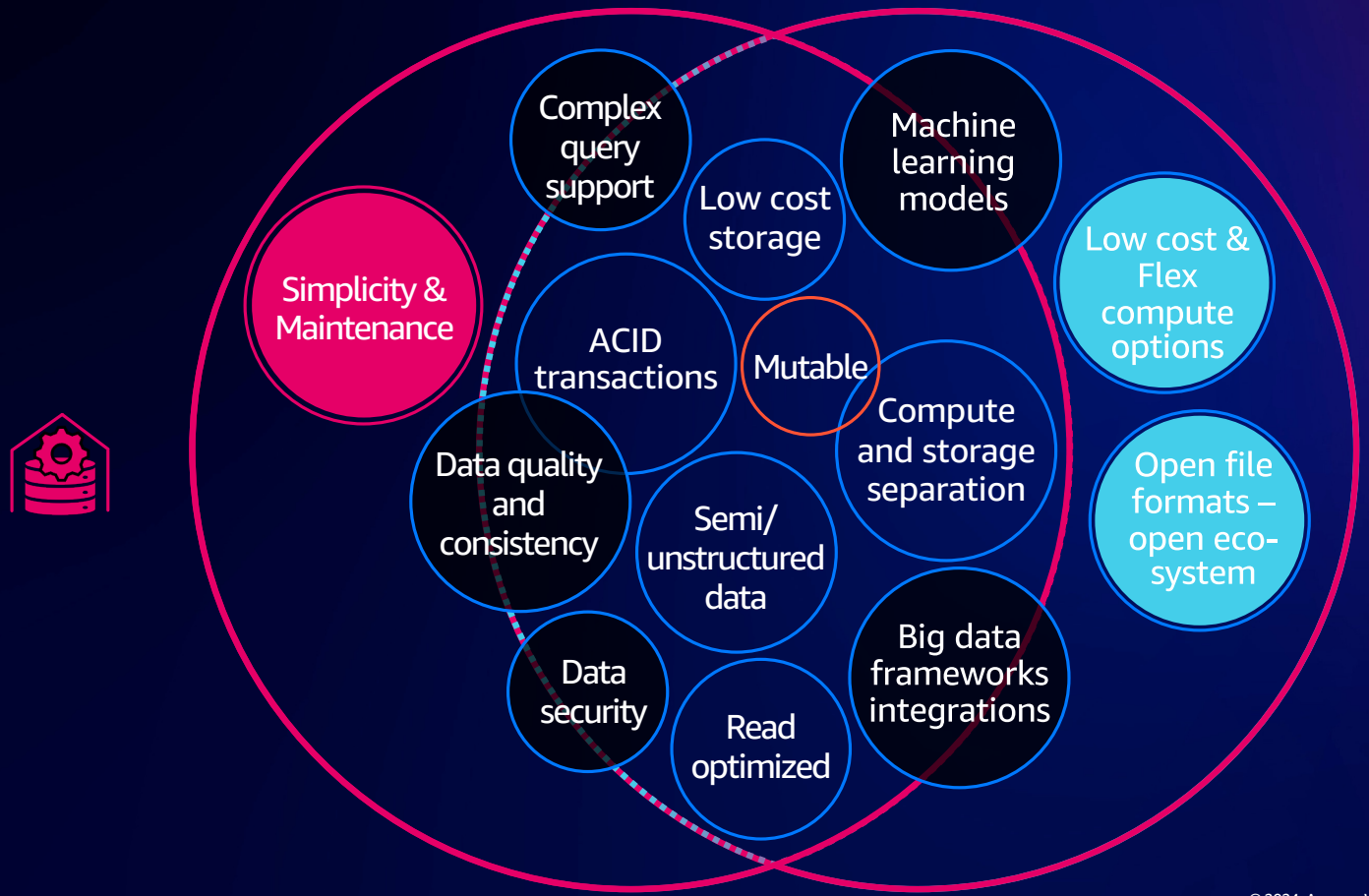


aws

Customer requirements are evolving also



Modern Data Lakes using an Open Table Format



Modern Data Lakes

Open table formats (OTFs) provide transactional support and simplify data lake optimization and management

Apache Hudi

Apache Iceberg

Delta Lake

Apache Iceberg Overview

What is Apache Iceberg?

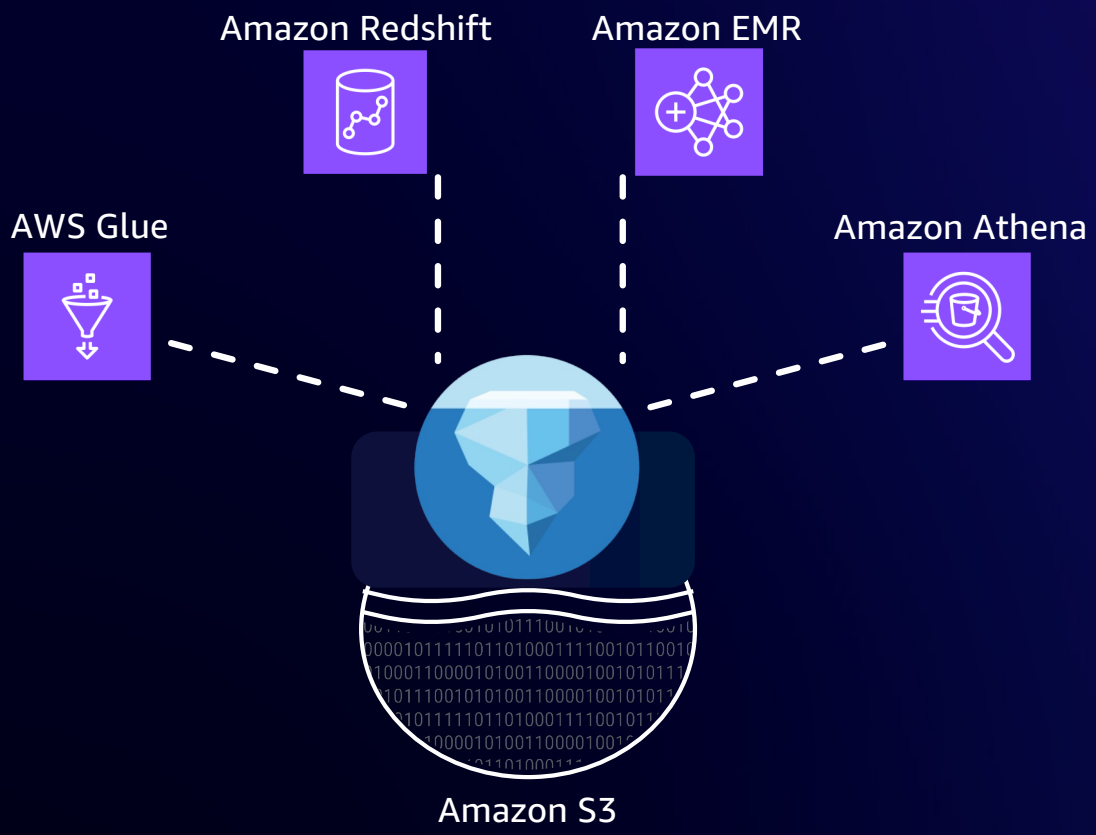
Apache Iceberg is an open table format for huge analytic datasets

- Open table format specification
- Set of libraries
- Java / Python / Rust APIs

trino

presto

AWS Apache Iceberg Integration

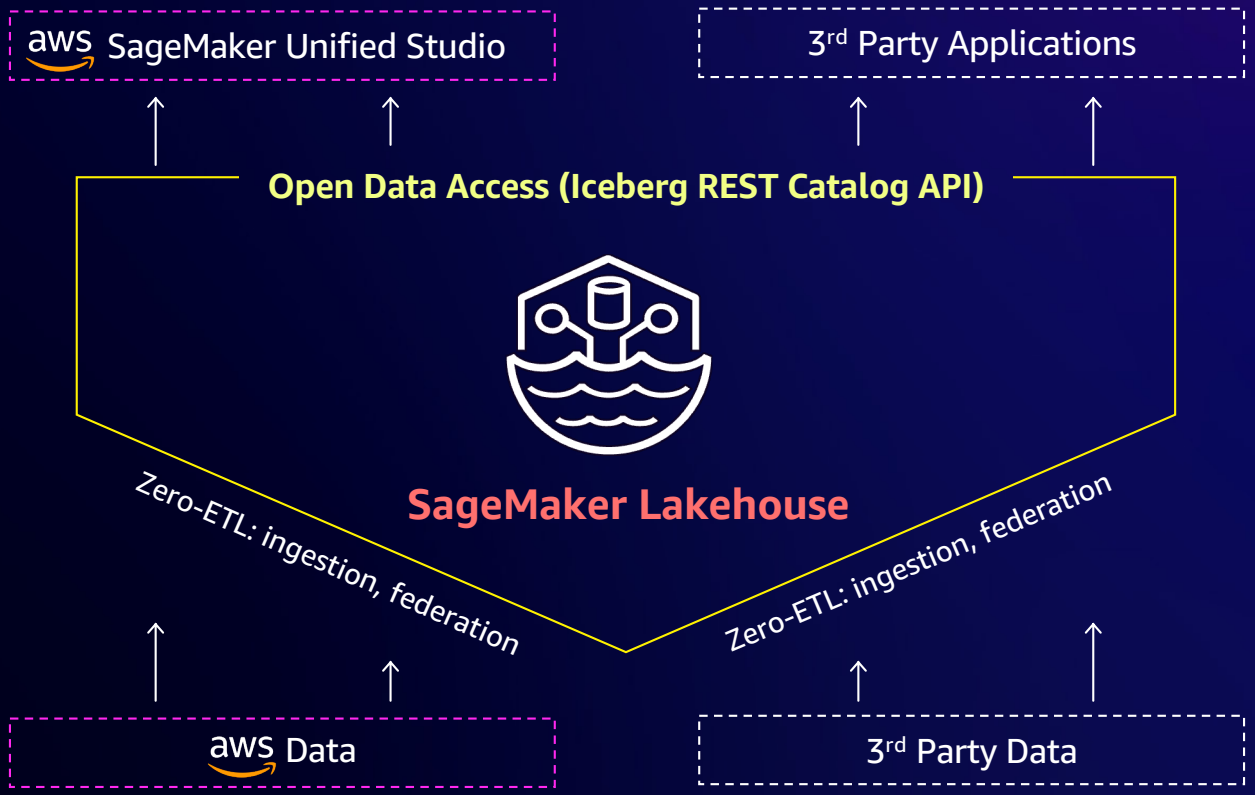


- EMR (v6.5+) - Spark, Flink, Trino, Presto
- Athena (v2 & v3)- DML, DDL, maintenance
- Glue - Interactive sessions, jobs, maintenance
- Redshift – Spectrum, Serverless

SageMaker Lakehouse

NEW

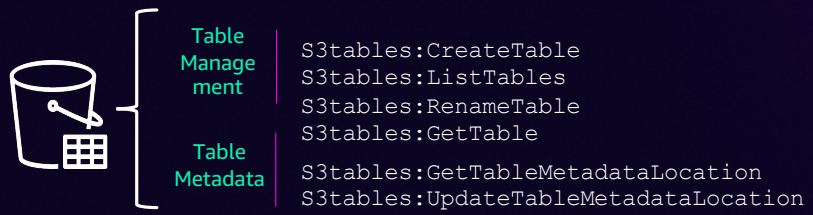
FIRST PARTY AND THIRD PARTY QUERY ENGINES AND TOOLS



Amazon S3 Tables

NEW

FULLY MANAGED STORAGE FOR APACHE ICEBERG DATA LAKE



New S3 storage class for Apache Iceberg data lakes

Amazon S3 APIs to read/write to S3 tables

Fully managed Iceberg table maintenance

Simple integration with the Lakehouse

10x Requests per second compared to standard Amazon S3 buckets

Apache Iceberg Features

Transactional updates

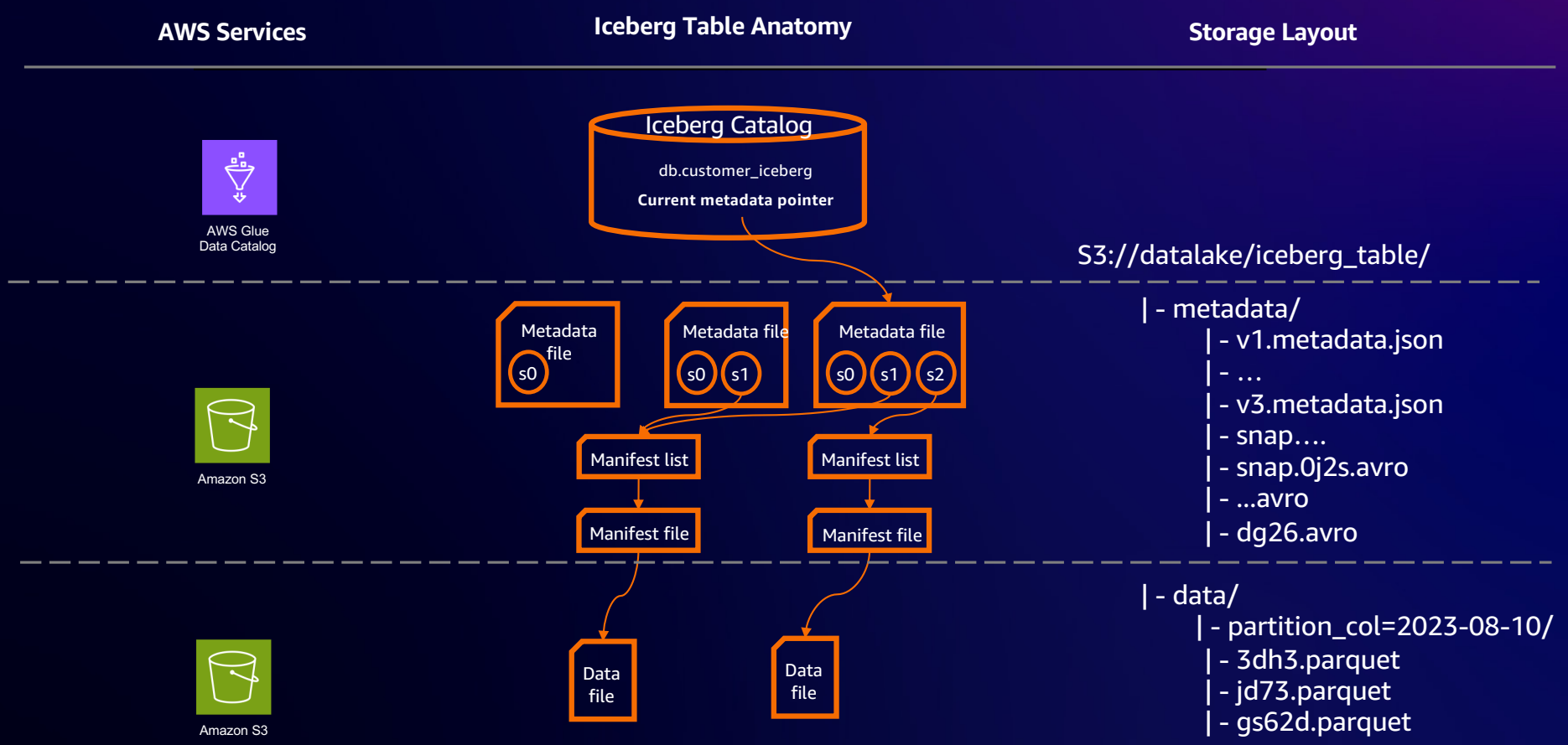
Time travel queries

Schema & partition evolution

Optimization & compaction

How Iceberg works

Iceberg Table Anatomy



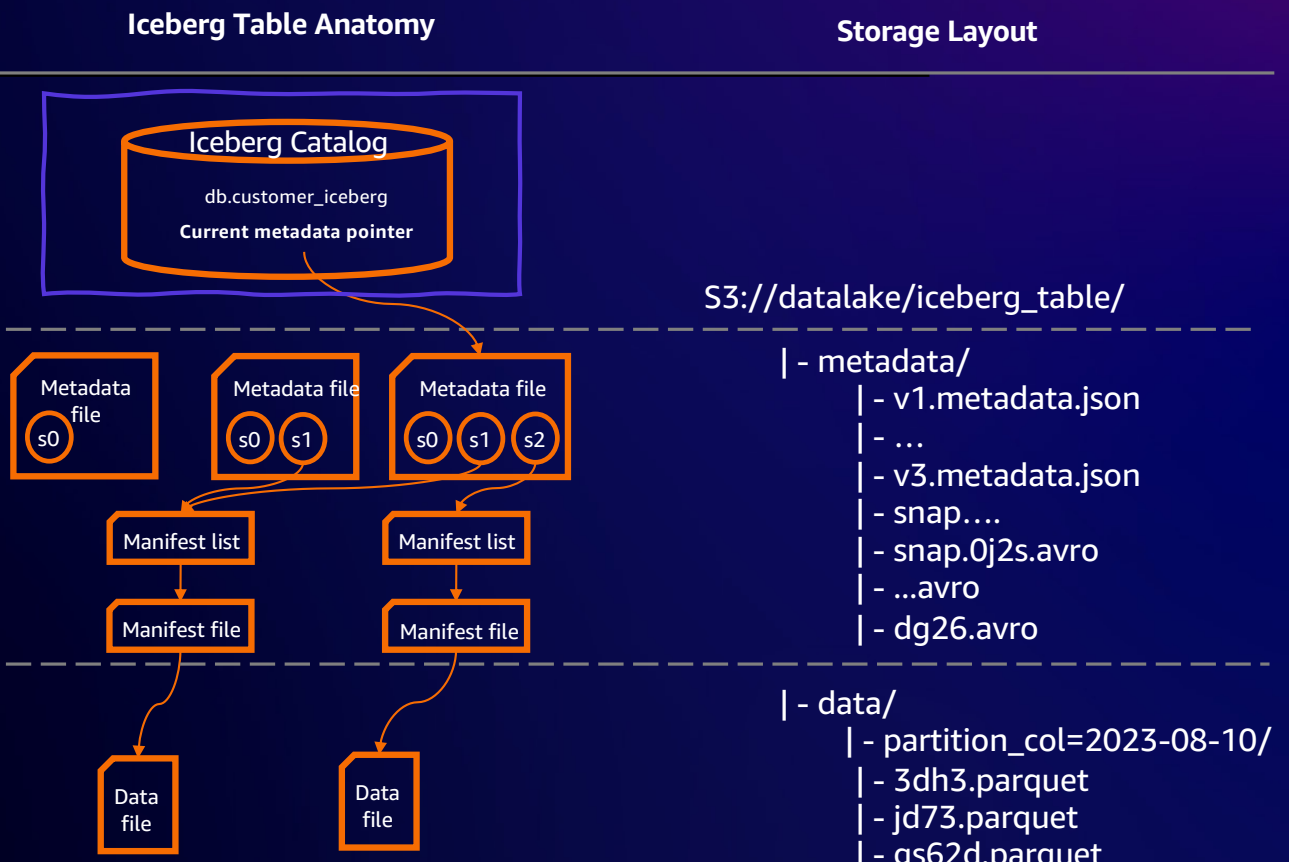
Iceberg Table Anatomy

Catalog

- Pointer to metadata file
- Unlike Hive Metastore, almost nothing else
- Supports atomic swap of the metadata files pointer

Metadata pointer

- S3 path to the latest metadata.json file



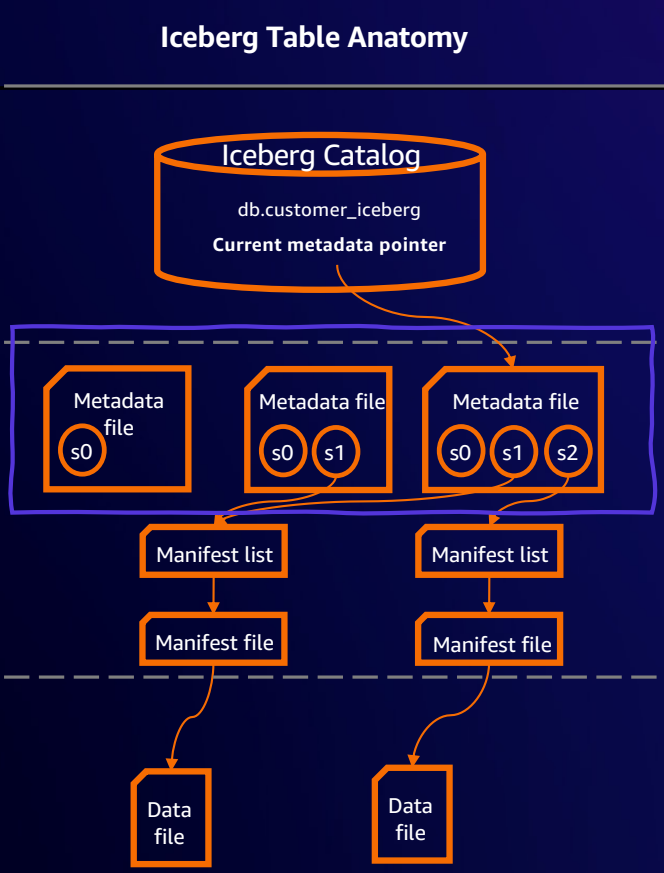
Iceberg Table Anatomy

Metadata file (json)

- "Root" of the table
- Contains table level metadata (schema, partitions, snapshots, properties)

Snapshot

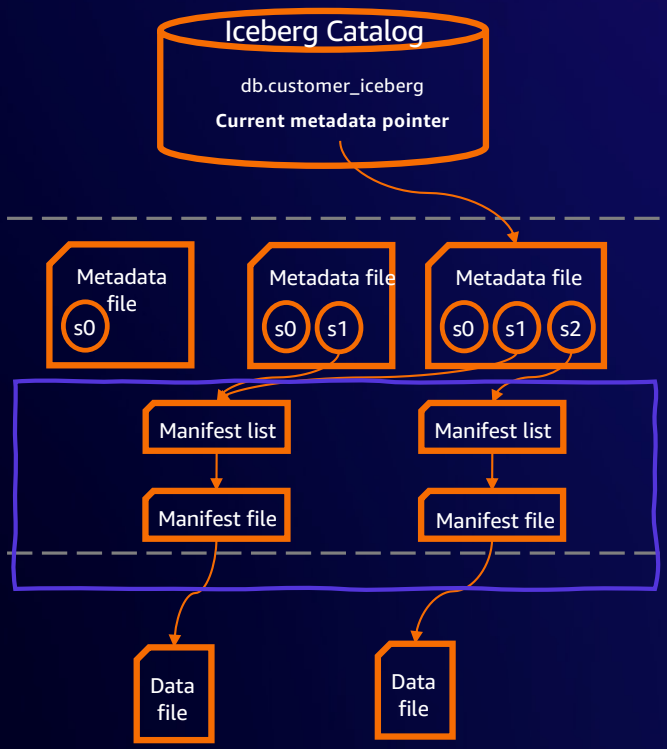
- Represents the state of the table at a given point in time
- Points to the manifest list

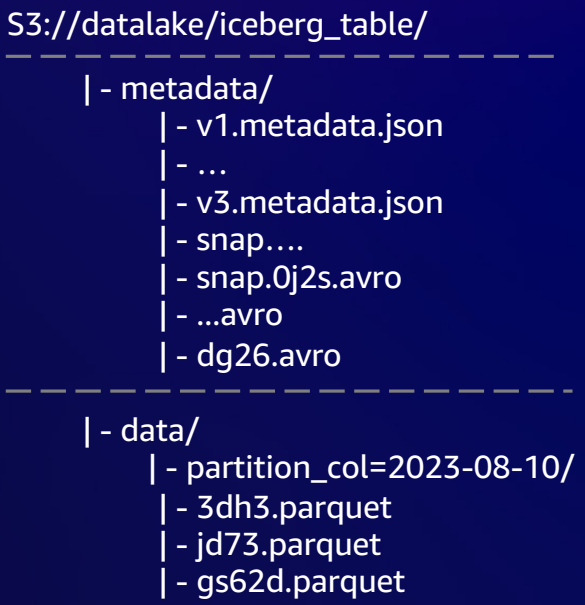


Iceberg Table Anatomy

Manifest list (avro)
- Pointers to Manifest files
- 1st level index of the table (i.e. high level statistics for query optimization)

Manifest file (avro)
- Pointers to data files
- 2nd level index of the table (i.e. data statistics for query optimization)





Quick Example

```
CREATE TABLE product_inventory (
  product_id BIGINT,
  product_name STRING,
  category STRING
)
PARTITIONED BY (category)
LOCATION 's3://my-bucket/product-inventory/'
TBLPROPERTIES (
  'table_type'='ICEBERG',
  'format'='parquet'
);
```

```
INSERT INTO product_inventory
VALUES
  (1, 'Microsoft Office', 'Software'),
  (2, 'Adobe Photoshop', 'Software'),
  (3, 'AutoCAD', 'Software'),
  (4, 'Visual Studio', 'Software');
```

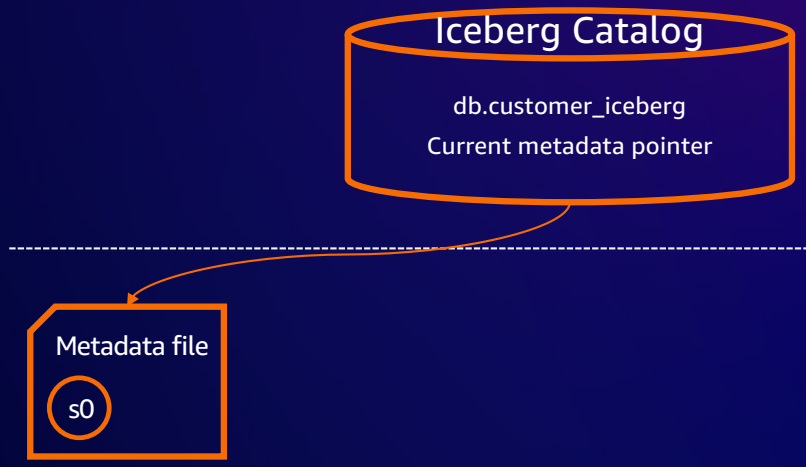
Create Table

S3://my-bucket/product-inventory/

/metadata/

/v1.metadata.json

/data/



Insert Records

S3://my-bucket/product-inventory/

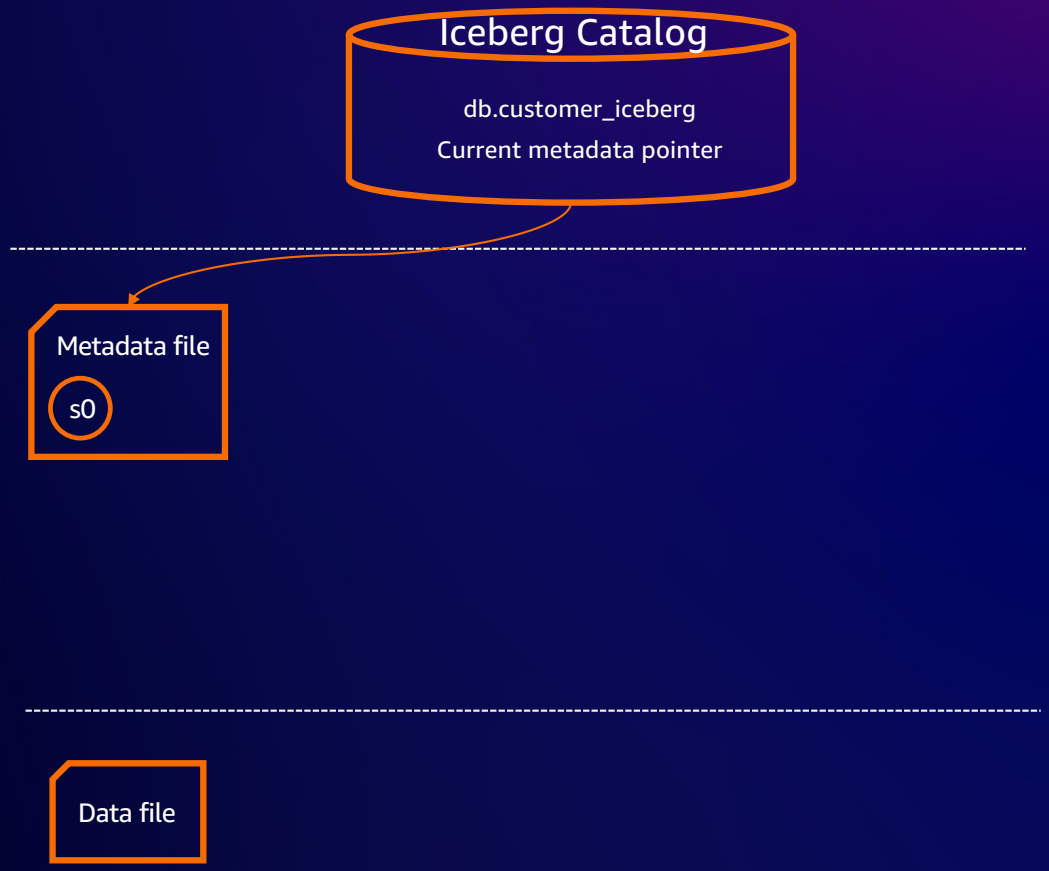
/metadata/

/v1.metadata.json

/data/

/category=Software/

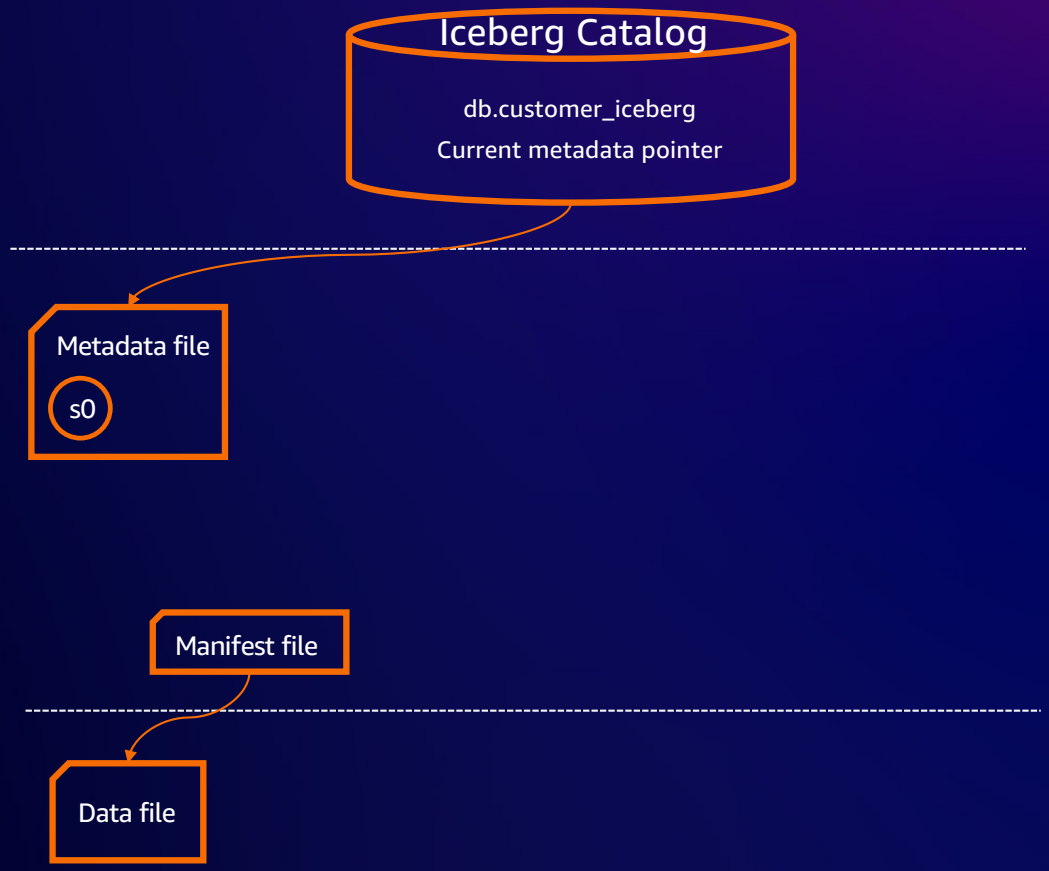
/3dh3.parquet



Insert Records

S3://my-bucket/product-inventory/

/metadata/
/v1.metadata.json
/dbda7.avro
/data/
/category=Software/
/3dh3.parquet



Insert Records

S3://my-bucket/product-inventory/

/metadata/

/v1.metadata.json

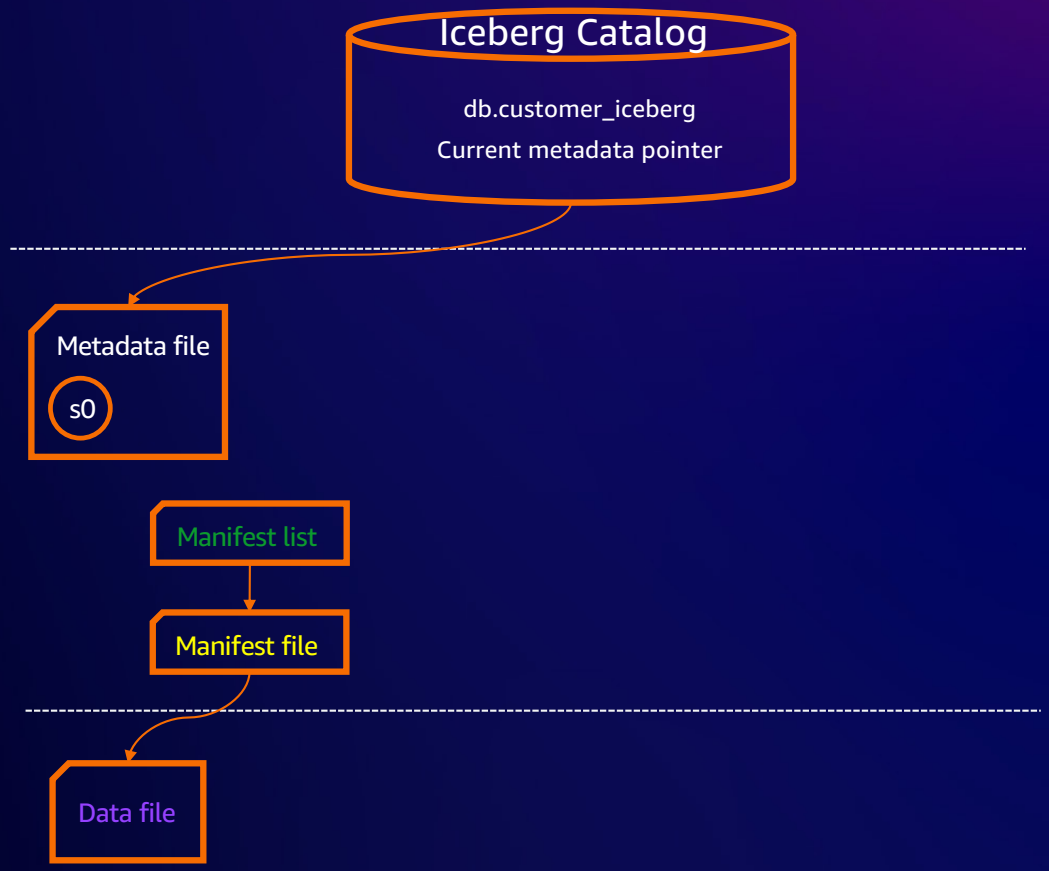
/snap.3ejn.avro

/dbda7.avro

/data/

/category=Software/

/3dh3.parquet



Insert Records

S3://my-bucket/product-inventory/

/metadata/

/v1.metadata.json

/v2.metadata.json

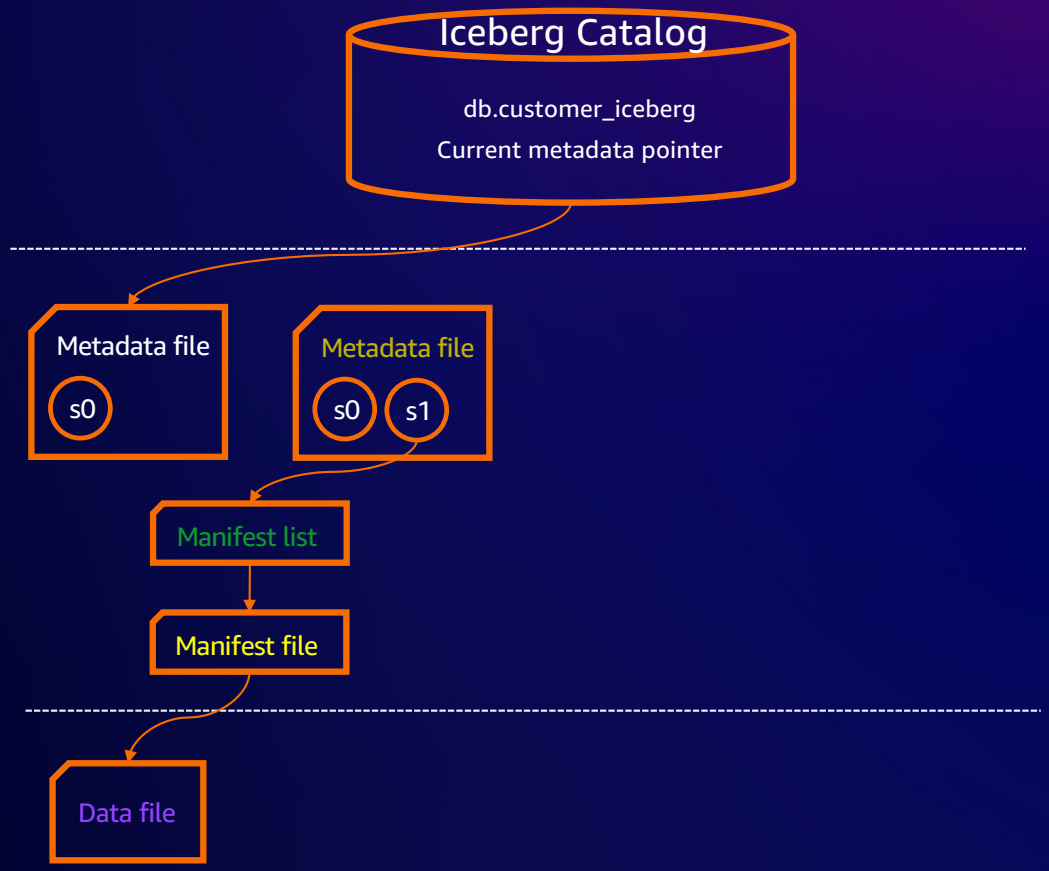
/snap.3ejn.avro

/dbda7.avro

/data/

/category=Software/

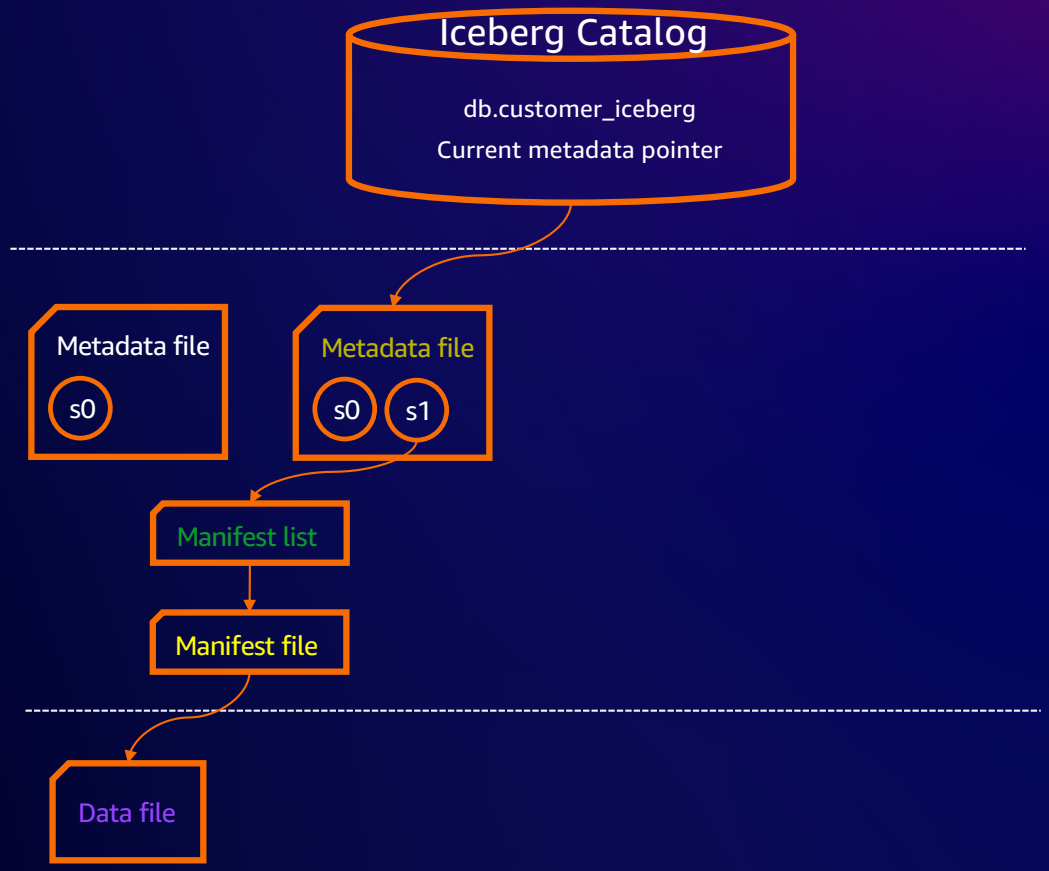
/3dh3.parquet



Insert Records

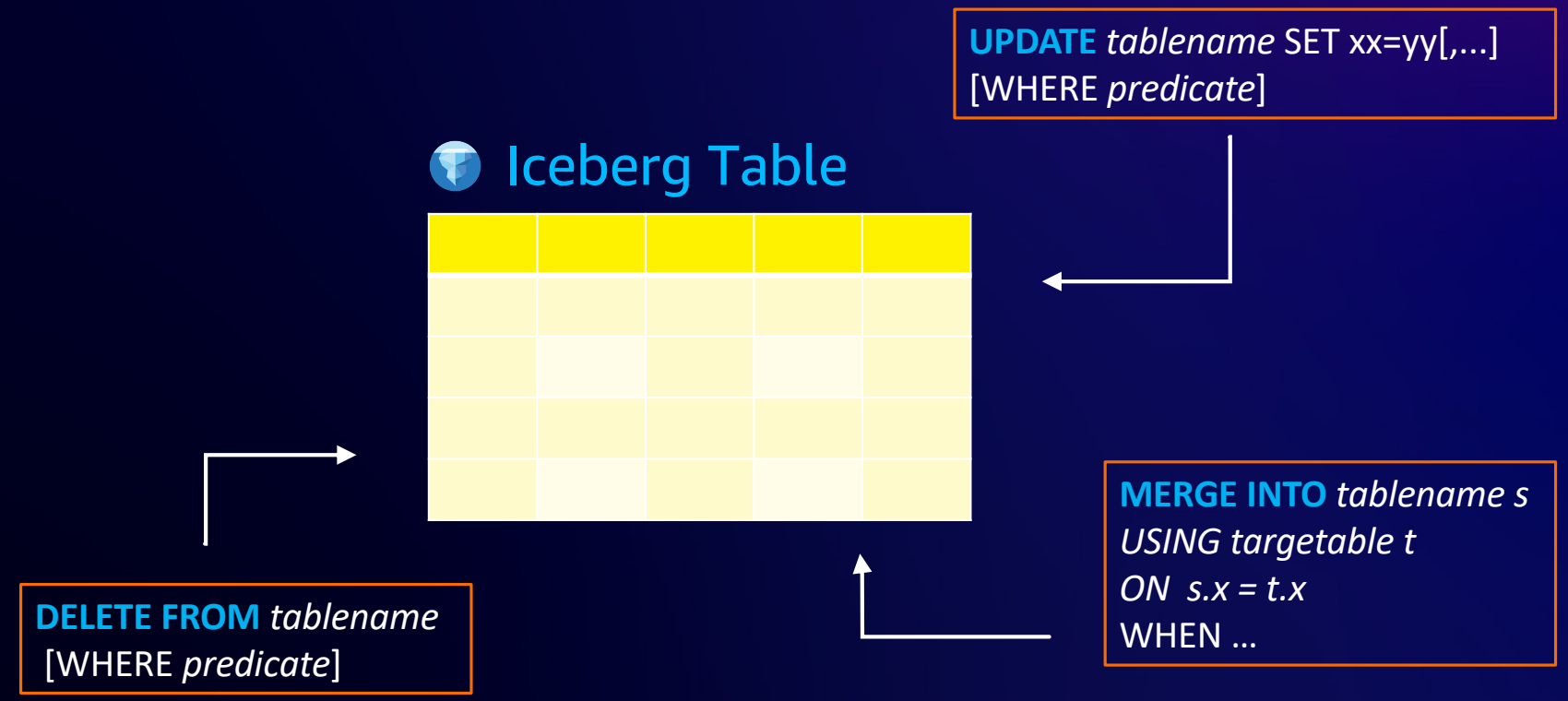
S3://my-bucket/product-inventory/

/metadata/
/v1.metadata.json
/v2.metadata.json
/snap.3ejn.avro
/dbda7.avro
/data/
/category=Software/
/3dh3.parquet



Apache Iceberg Feature Highlights

#1 Expressive SQL



#2 Schema & Partition Evolution

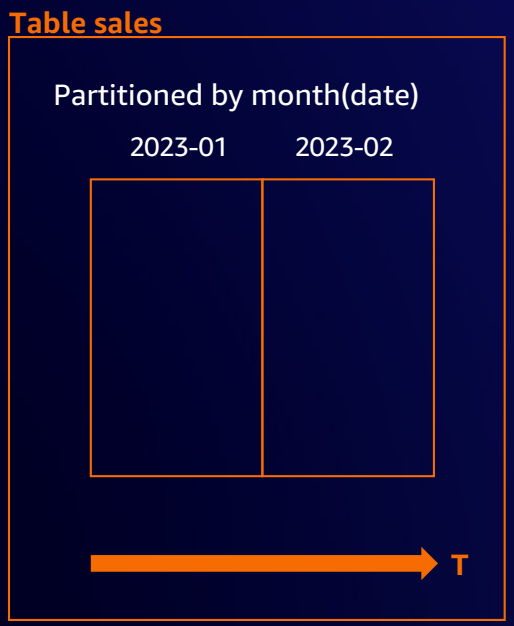
Schema evolution changes are **independent and free of side-effects** without rewriting files

Supported schema evolution changes (even in nested structures)

- Add column
- Drop column
- Rename column
- Reorder column
- Update column

#2 Schema & Partition Evolution

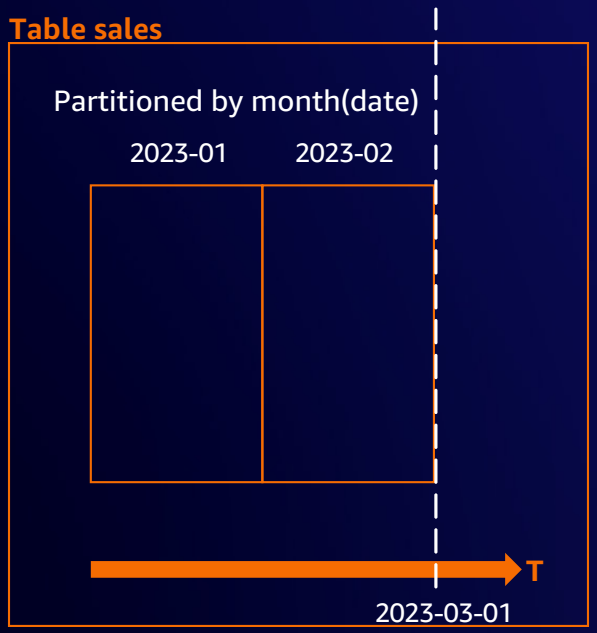
Iceberg supports changing partition layout



#2 Schema & Partition Evolution

Just a metadata operation

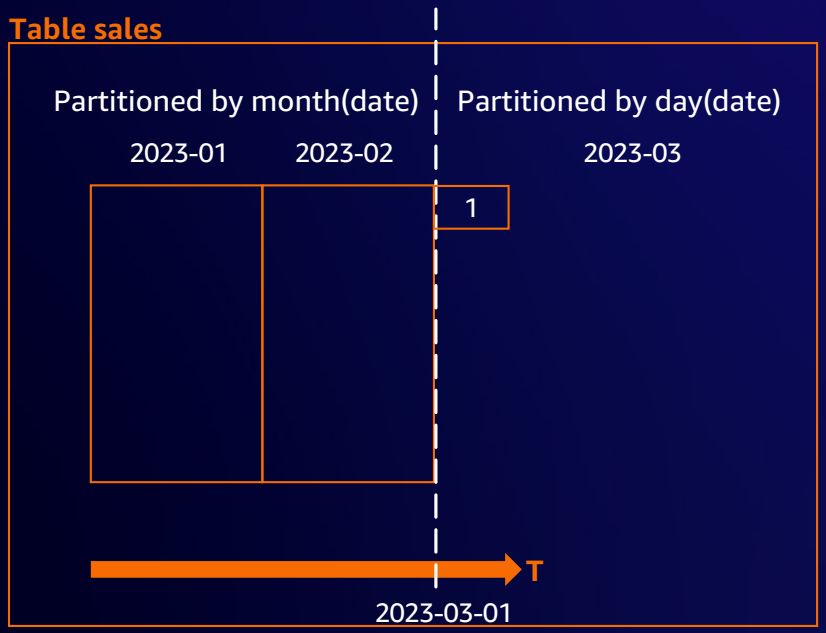
ALTER TABLE sales REPLACE FIELD
Month(date) WITH days(date)



#2 Schema & Partition Evolution

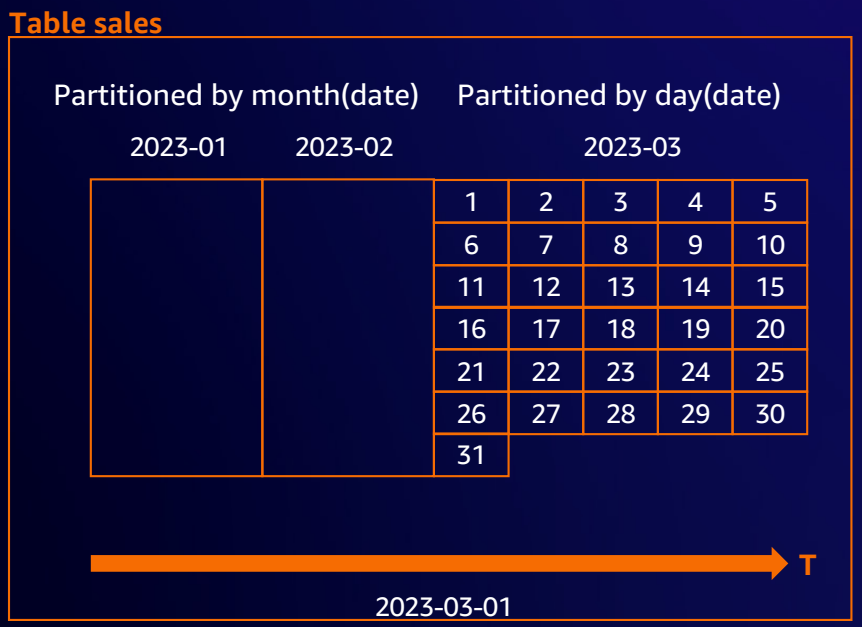
Just a metadata operation

ALTER TABLE sales REPLACE FIELD
Month(date) WITH days(date)



#2 Schema & Partition Evolution

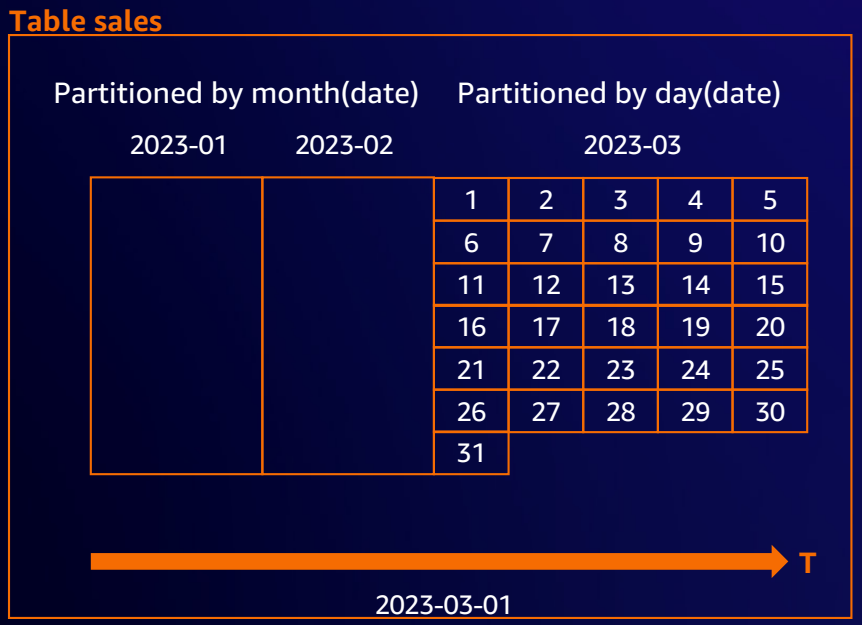
There are no additional costs such as moving data, migrating tables, or rewriting queries



#2 Schema & Partition Evolution

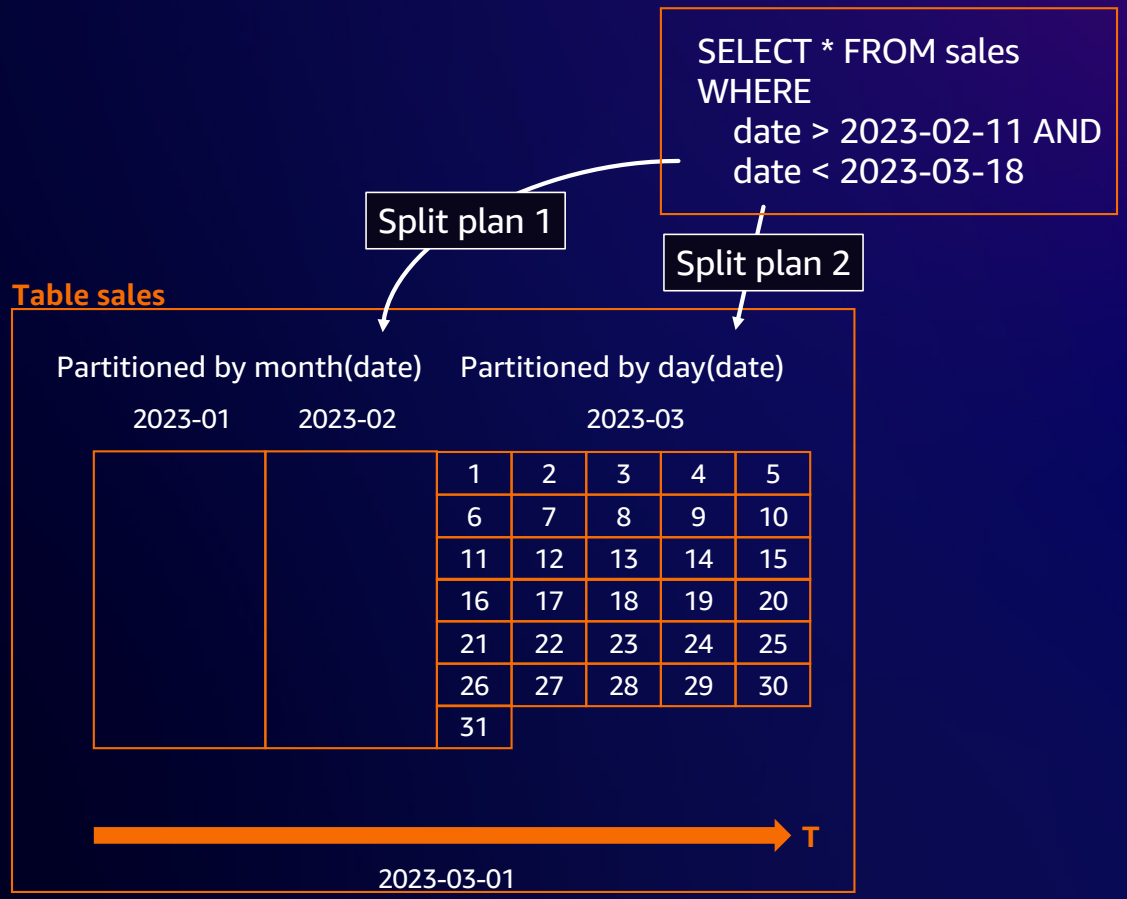
Query example

```
SELECT * FROM sales
WHERE
   date > 2023-02-11 AND
   date < 2023-03-18
```



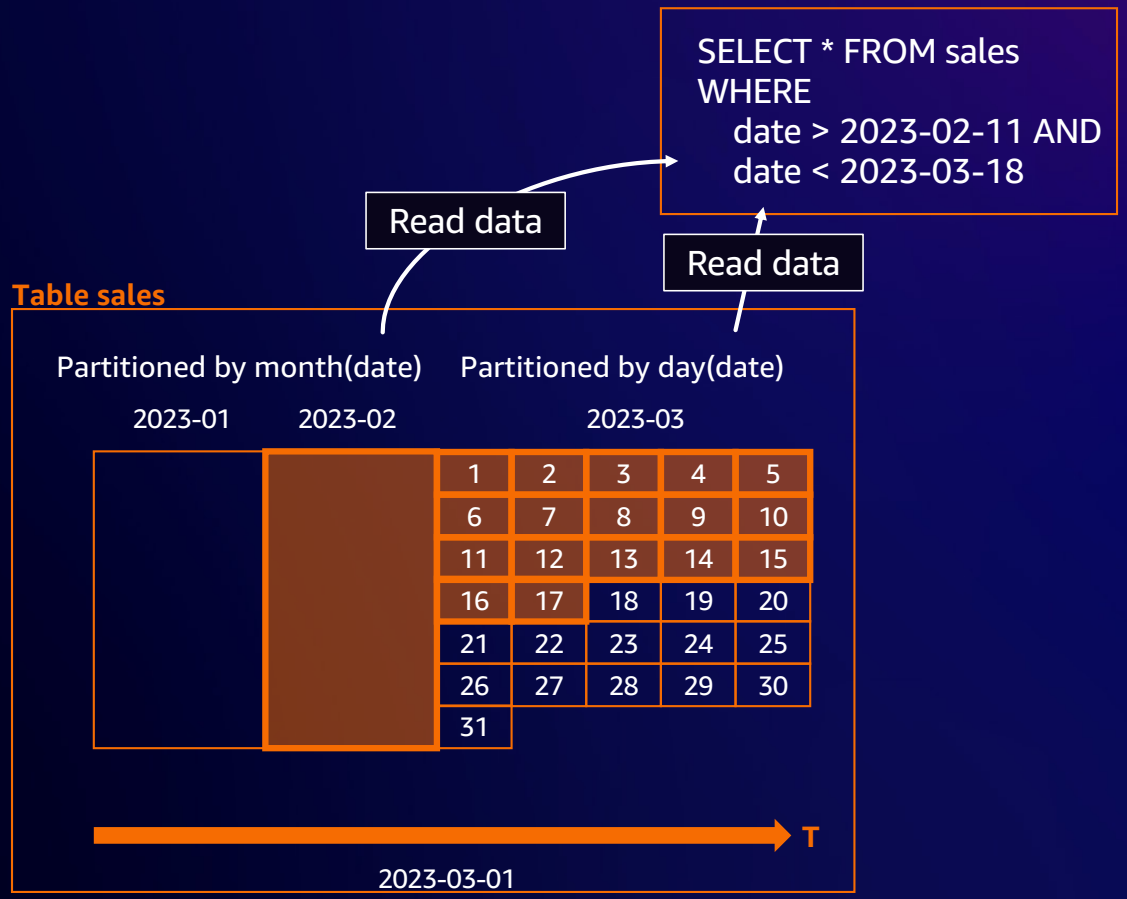
#2 Schema & Partition Evolution

Query example



#2 Schema & Partition Evolution

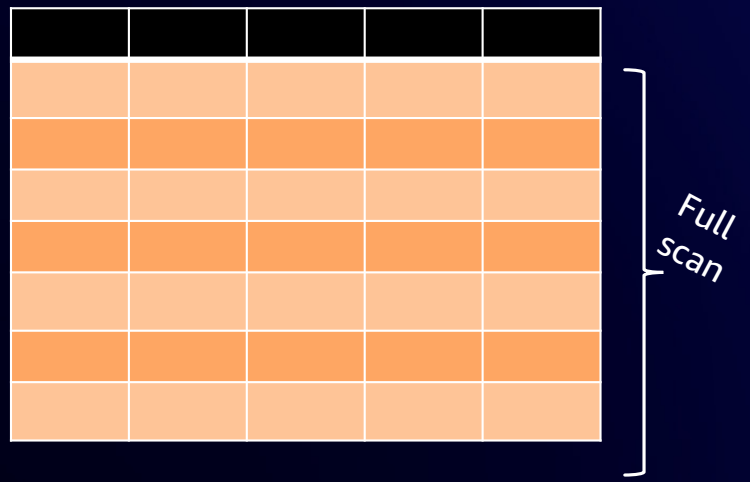
Query example



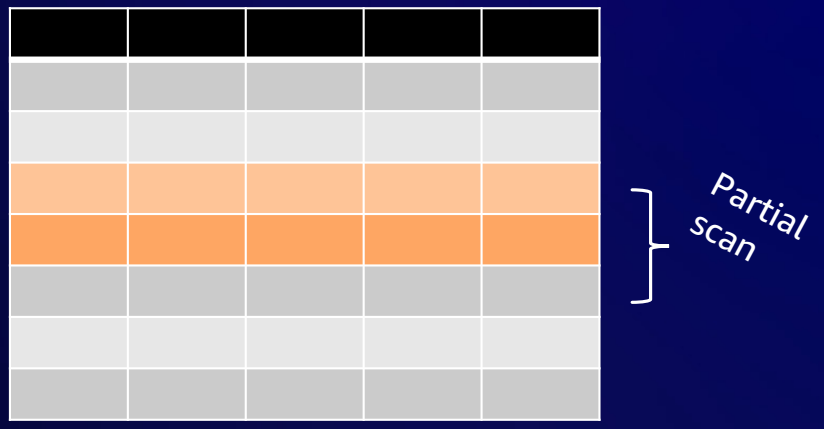
#3 Simplified Queries with Hidden Partitioning

*Example of an **Hive** query*

```
SELECT level, Count(1) as count
FROM   logs
WHERE  event_time BETWEEN
         '2018-12-01 10:00:00'
AND      '2018-12-01 10:10:00'
```




```
SELECT level, Count(1) as count
FROM   logs
WHERE  event_time BETWEEN
         '2018-12-01 10:00:00'
AND      '2018-12-01 10:10:00'
AND    event_date = '2018-12-01'
```



#3 Simplified Queries with Hidden Partitioning

Tables are created using “Partition Transforms”

```
CREATE TABLE catalog.db.logs(
    id long,
    log_event string,
    level string,
    event_time timestamp
) PARTITIONED BY (day(event_time)) USING iceberg;
```

Example of an Iceberg query

```
SELECT level, Count(1) as count
FROM   logs
WHERE  event_time BETWEEN
         '2018-12-01 10:00:00'
AND      '2018-12-01 10:10:00'
```

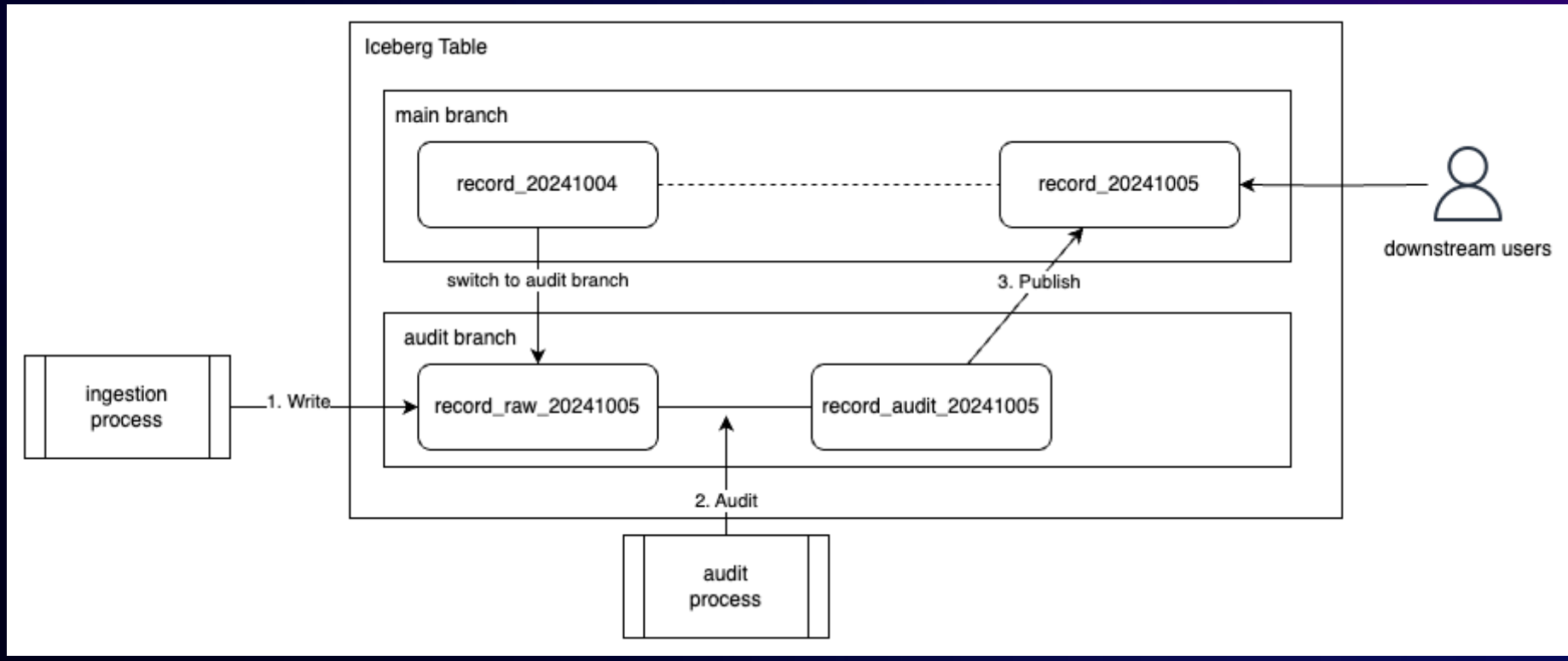
#4 Tagging and Branching for Compliance

Each transaction (insert/delete/update/compaction) creates a new table snapshot

#4 Tagging and Branching for Compliance

Keep only the most relevant Table Snapshots for the business

#4 Tagging and Branching for DataOPS
Write – Audit- Publish (WAP) Pattern

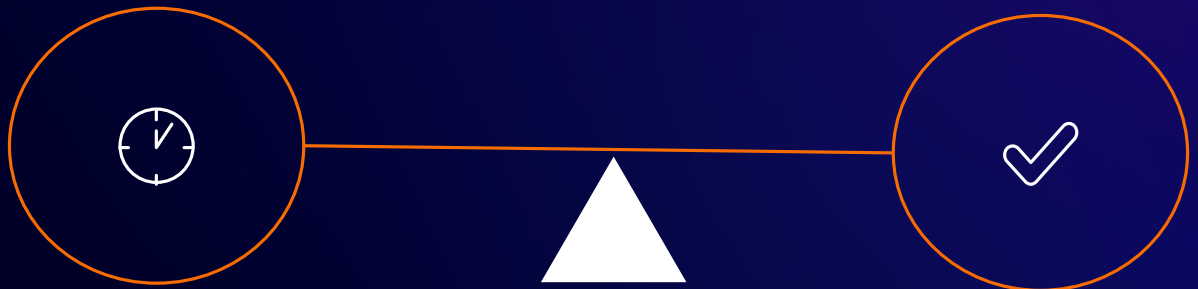


Apache Iceberg Table Update Strategy

Tables Update Strategies

Copy-On-Write

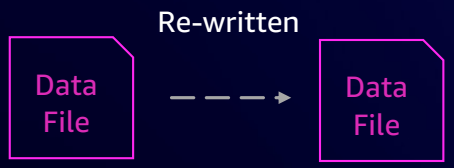
Merge-on-Read



write.update.mode | write.delete.mode | write.merge.mode

Tables Update Strategies

Copy-On-Write – COW



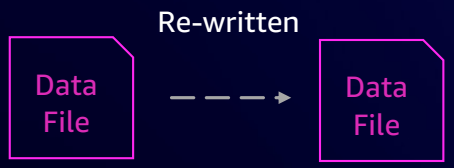
Writer/Reader Trade-off

Cost may be higher for writes

No impact on the readers side

Tables Update Strategies

Copy-On-Write – COW



Writer/Reader Trade-off

- Cost may be higher for writes
- No impact on the readers side

Merge-On-Read - MOR

Writer/Reader Trade-off

- Offloads writing cost
- Adds overhead for reads

Apache Iceberg Table Maintenance

Iceberg Table Maintenance

Iceberg tables may need additional maintenance to be “healthy” and “performant”

Iceberg provides a “Control Plane”

- Iceberg provides Procedures implemented in Spark, Flink, or Trino/ Athena → table maintenance is managed by the user
- AWS Glue Data Catalog provides a serverless experience for Iceberg tables → table maintenance is managed by AWS
- Fully managed Iceberg maintenance on a storage level using S3 Tables

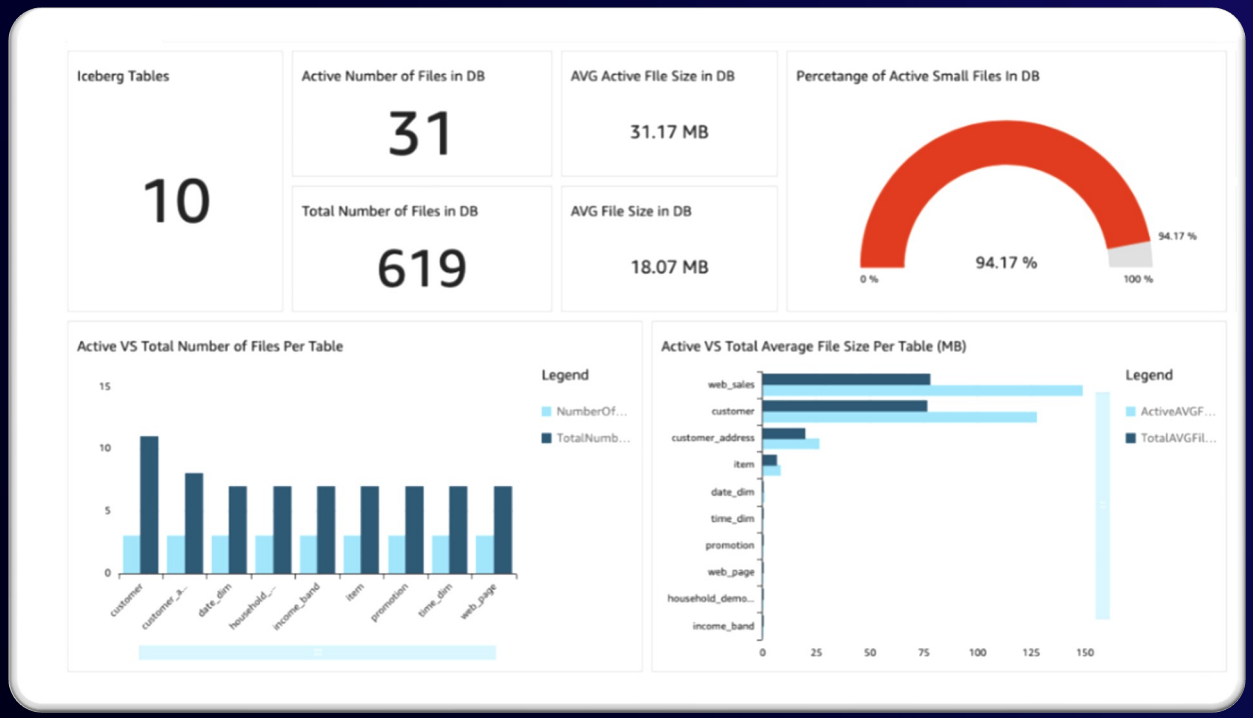
Iceberg Table Maintenance

Most important actions

- Compactions – contains multiples operations:
  - combine small objects together
  - sort the data using hierarchal sorting
  - clustering the data using z-ordering
- Expirations – removes ”old” metadata
- Orphan reclamation – removes unreferenced objects

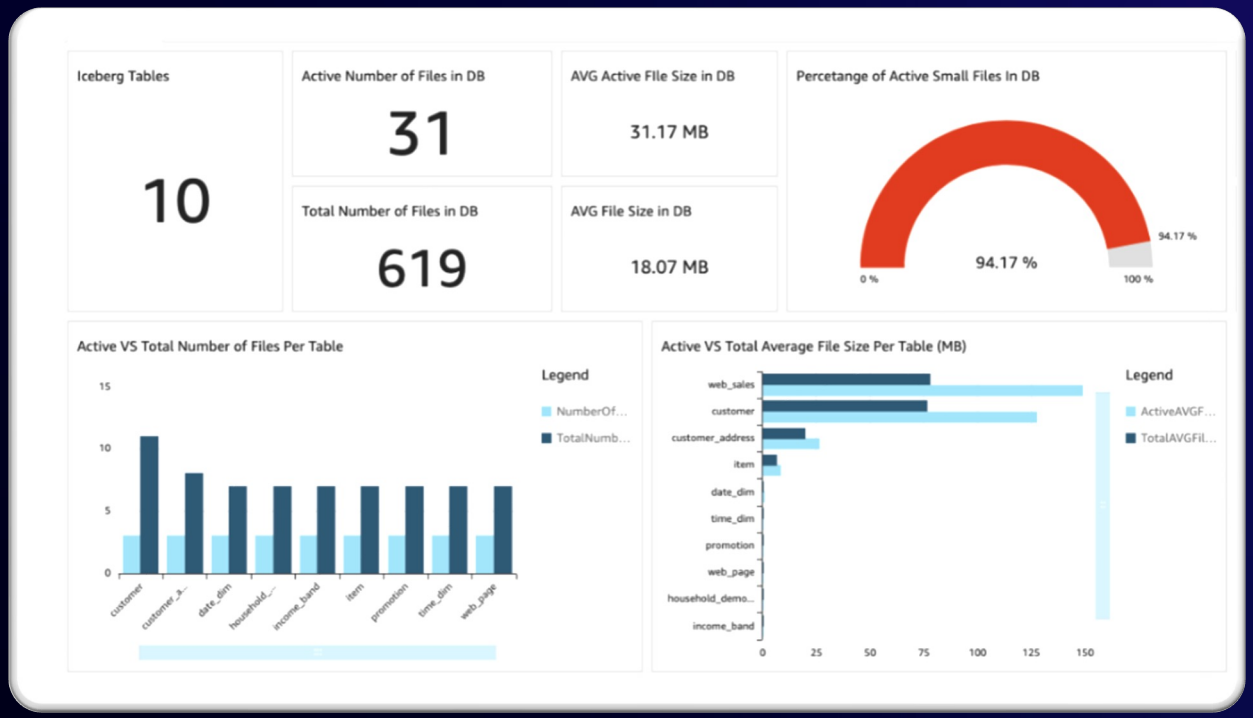
Maintaining Iceberg Tables

Monitor Iceberg table with metadata tables



Maintaining Iceberg Tables

Monitor Iceberg table with metadata tables



Key maintenance tasks:

- Rewrite Files (Data File Optimization)
- Rewrite Manifests (Metadata Optimization)
- Delete orphan files (Cleanup)

Table maintenance by S3

NEW

PERFORMANCE

Compaction: Consolidate small objects into larger ones to improve query performance.
Snapshot Retention: Remove unused snapshots

Key Takeaways

- Active Open-Source Community
- AWS Prioritizes Apache Iceberg
- Production-grade maturity

Migrate an existing data lake to a transactional data lake using Apache Iceberg

Amazon S3 Tables integration with Amazon SageMaker Lakehouse is now generally available

Thank you!

David Greenshtein

dgreensh@amazon.de