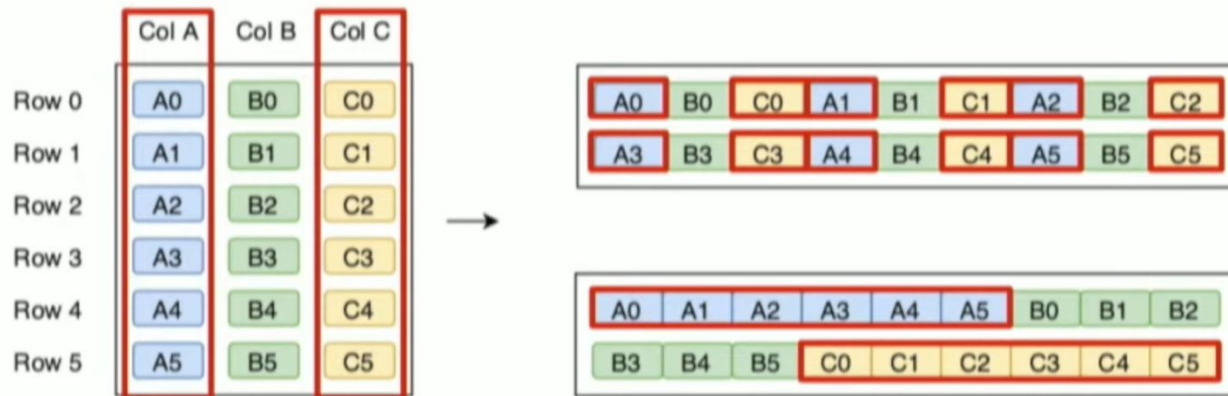


INTRO TO Apache Iceberg

Recap Parquet

Row-wise vs Columnar



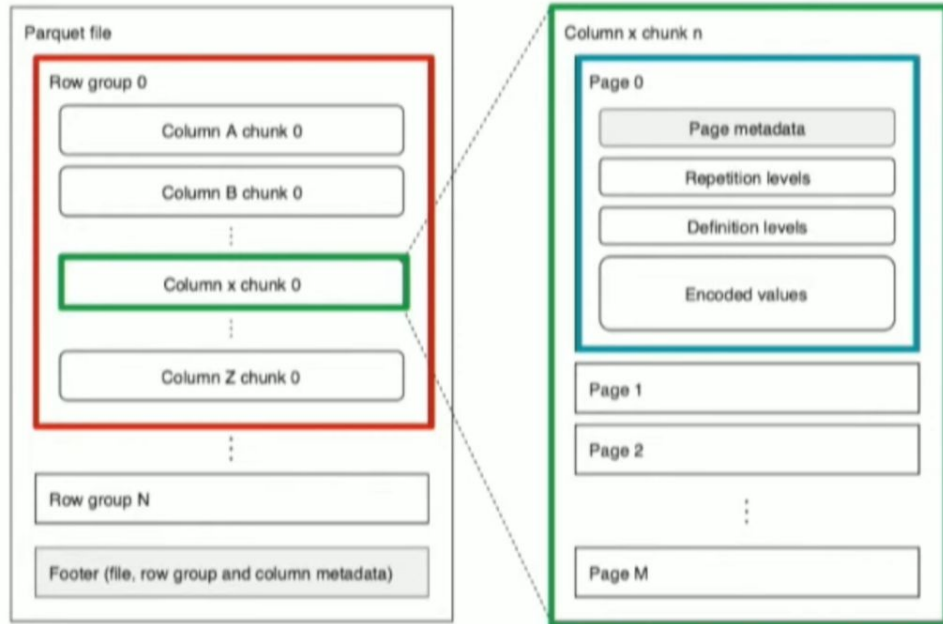
	Col A	Col B	Col C
Row 0	A0	B0	C0
Row 1	A1	B1	C1
Row 2	A2	B2	C2
Row 3	A3	B3	C3
Row 4	A4	B4	C4
Row 5	A5	B5	C5



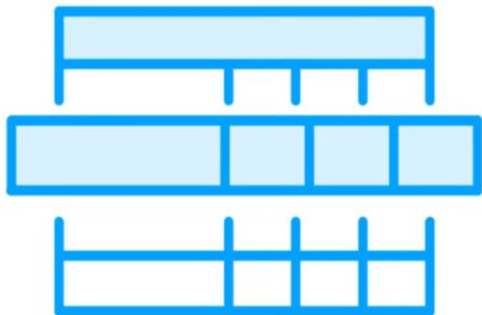
A0	A1	A2	B0	B1	B2	C0	C1	C2
A3	A4	A5	B3	B4	B5	C3	C4	C5

Parquet: data organization

- Data organization
 - Row-groups (*default 128MB*)
 - Column chunks
 - Pages (*default 1MB*)
 - Metadata
 - Min
 - Max
 - Count
 - Rep/def levels
 - Encoded values



How Parquet “Knows” Which Row Group to Scan?



- **Parquet contains metadata**
 - Data about data
- **Min and max values**
- **Footer**
 - Format version
 - Schema information
 - Column metadata

Parquet Storage

	Column 1	Column 2	Column 3	Column 4	Column 5
	Product	Customer	Country	Date	Sales Amount
Row group 1	Ball	John Doe	USA	2023-01-01	100
	T-Shirt	John Doe	USA	2023-01-02	200
Row group 2	Socks	Maria Adams	UK	2023-01-01	300
	Socks	Antonio Grant	USA	2023-01-03	100
Row group 3	T-Shirt	Maria Adams	UK	2023-01-02	500
	Socks	John Doe	USA	2023-01-05	200

Projection and Predicate(s)

Projection = SELECT

Predicate(s) = WHERE

	Column 1	Column 2	Column 3	Column 4	Column 5
	Product	Customer	Country	Date	Sales Amount
Row group 1	Ball	John Doe	USA	2023-01-01	100
	T-Shirt	John Doe	USA	2023-01-02	200
Row group 2	Socks	John Doe	UK	2023-01-02	500
	Socks		USA		
Row group 3	T-Shirt	Maria Adams	UK	2023-01-02	500
	Socks	John Doe	USA	2023-01-05	200

The engine will skip scanning these records!

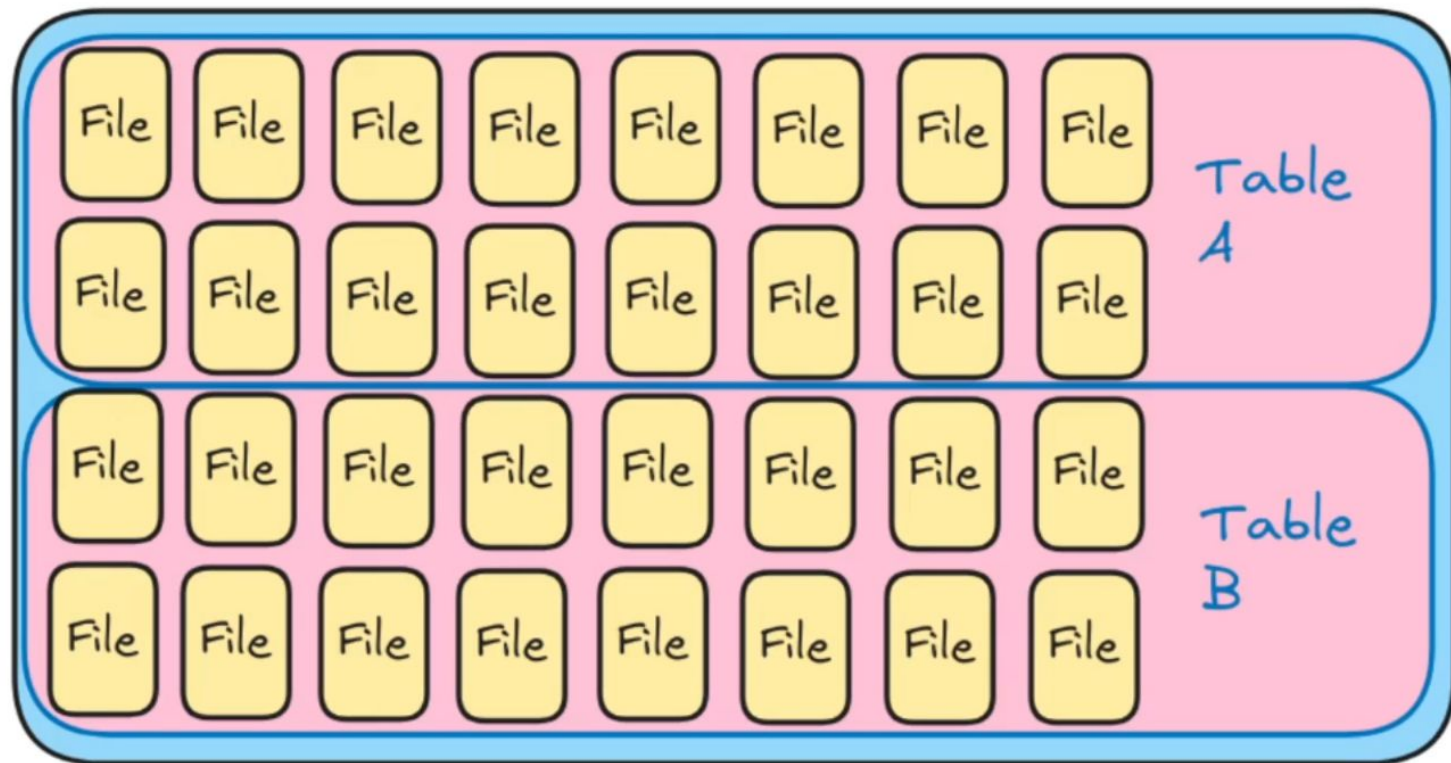


@DataMozart



What is a Table Format?

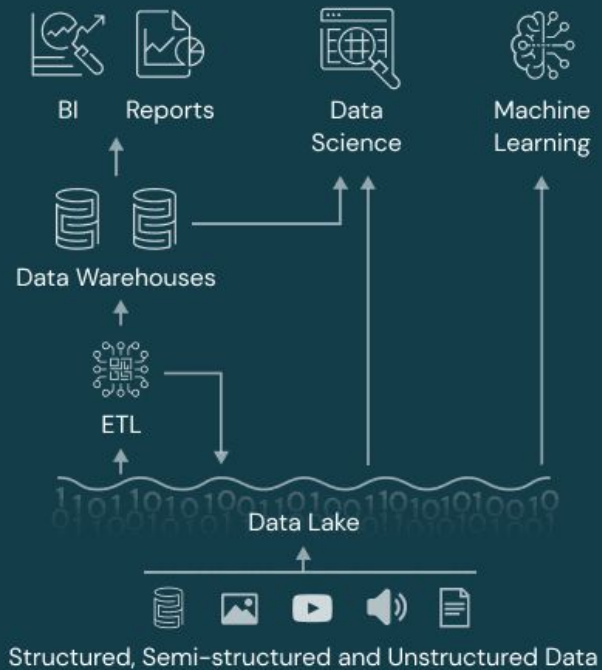
An abstraction to allow multiple files on the lake to be seen as a single table.



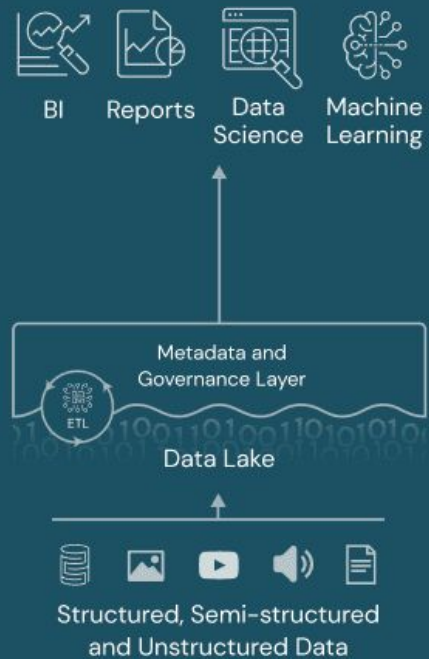
Data Warehouse



Data Lake



Data Lakehouse





Analytics

Storage
Engine



Compute
engine



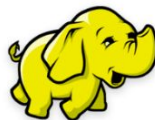
Catalog



Table formats

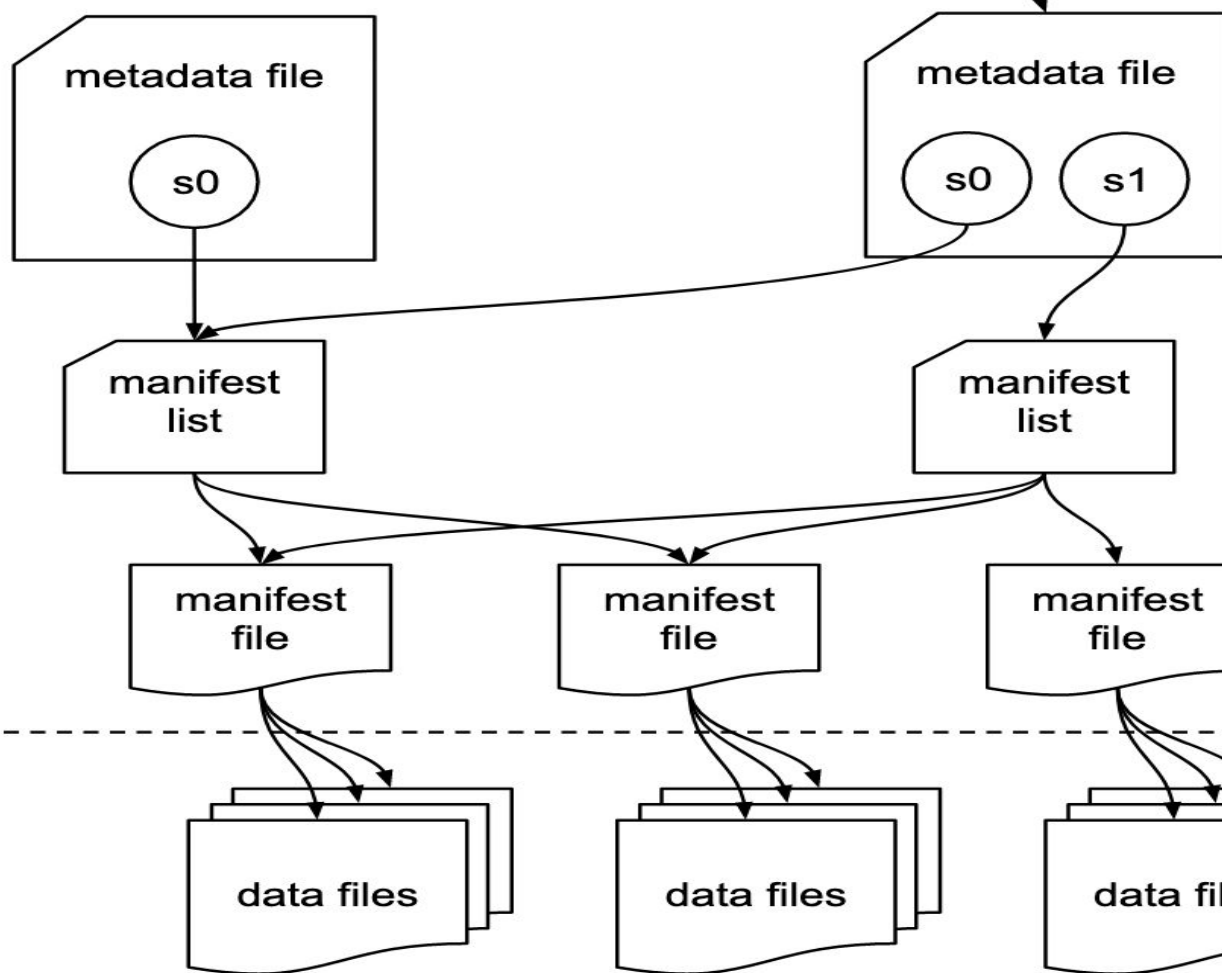


File formats

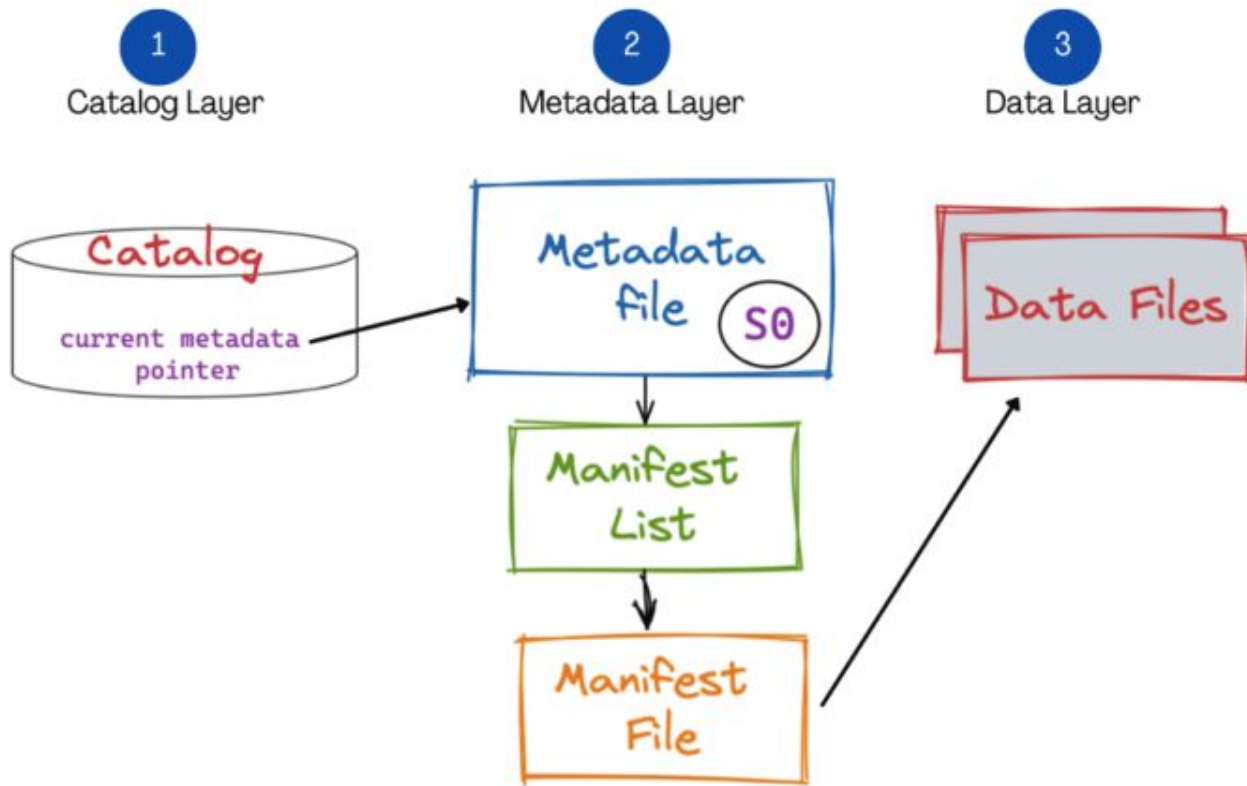


Lake Storage

metadata layer



data layer



Iceberg Table Metadata File (metadata.json)

```
{
  "format-version": 2,
  "table-uuid": "b3d6d8b8-62f1-4fa6-9a8d-08a9f7d4aa31",
  "location": "s3://my-bucket/warehouse/db/sales",
  "current-snapshot-id": 3055729220031123456,
  "snapshots": [
    {
      "snapshot-id": 3055729220031123456,
      "timestamp-ms": 1736940000123,
      "operation": "append",
      "manifest-list": ".../metadata/snap-3055729220031123456-1.avro"
    }
  ],
  "schemas": [
    { "schema-id": 0, "fields": [ ... ] }
  ],
  "partition-specs": [
    { "spec-id": 0, "fields": [ { "transform": "day(order_ts)" } ] }
  ]
}
```

Iceberg Manifest List (Avro)

Manifest List (decoded view) – one record per manifest file

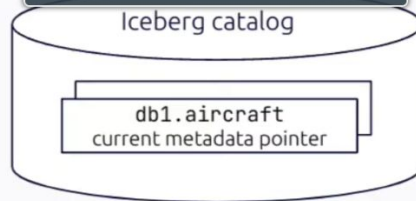
```
[
  {
    "manifest_path": ".../metadata/0d3c2b2f-...-m0.avro",
    "content": "DATA",
    "partition_spec_id": 0,
    "sequence_number": 17,
    "added_snapshot_id": 3055729220031123456,
    "added_files_count": 2,
    "existing_files_count": 0,
    "deleted_files_count": 0,
    "partitions": [
      { "lower_bound": "2026-01-15", "upper_bound": "2026-01-16" }
    ]
  }
]
```

Iceberg Manifest File (Avro)

Manifest File (decoded view) – entries point to data/delete files

```
[
  {
    "status": "ADDED",
    "snapshot_id": 3055729220031123456,
    "sequence_number": 17,
    "data_file": {
      "content": "DATA",
      "file_path": ".../data/order_day=2026-01-15/00000-0-....parquet",
      "file_format": "PARQUET",
      "partition": { "order_day": "2026-01-15" },
      "record_count": 60000,
      "file_size_in_bytes": 134217728,
      "lower_bounds": { "order_id": "100000" },
      "upper_bounds": { "order_id": "159999" }
    }
  },
  {
    "status": "ADDED",
    "data_file": {
      "file_path": ".../data/order_day=2026-01-16/00001-0-....parquet",
      "record_count": 60000
    }
  }
]
```

```
CREATE TABLE aircraft (  
    tail_number varchar(15),  
    description varchar(150),  
    class varchar(50),  
    year integer  
)  
WITH  
    (type = 'iceberg');
```

metadata layer



data layer

Objects (1) [Info](#)

Copy S3 URI



Copy URL



Download

Open

Delete

Actions ▼

Create folder



Upload

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)



Find objects by prefix



1



Name ▲



Type ▼



Last modified ▼



Size ▼



Storage class ▼

 [metadata/](#)

Folder

-

-


-

Objects (2) [Info](#) Copy S3 URI Copy URL DownloadOpen 

Delete

Actions ▼

Create folder



 Upload

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#)  to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#) 

 Find objects by prefix

< 1 >



<input type="checkbox"/>	Name ▲	Type ▼	Last modified ▼	Size ▼	Storage class ▼
<input type="checkbox"/>	 00000-e6b05edc-87c5-44c0-91ed-504767cd1107.metadata.json	json	May 28, 2024, 15:06:23 (UTC-04:00)	2.1 KB	Standard
<input type="checkbox"/>	 snap-425416382669527773-1-189af56b-cb7a-40b0-a0d1-6a94c3796ca0.avro	avro	May 28, 2024, 15:06:23 (UTC-04:00)	4.0 KB	Standard

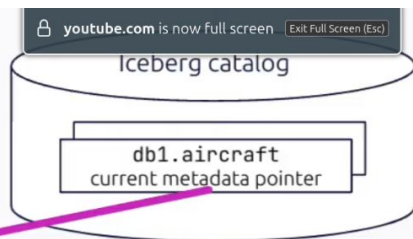
```
INSERT INTO
```

```
  aircraft (tail_number, description, class, year)
```

```
VALUES
```

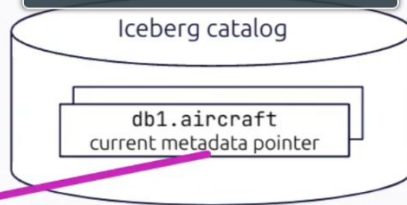
```
  ('N535NA', 'NASA', 'Helicopter', 1969),
```

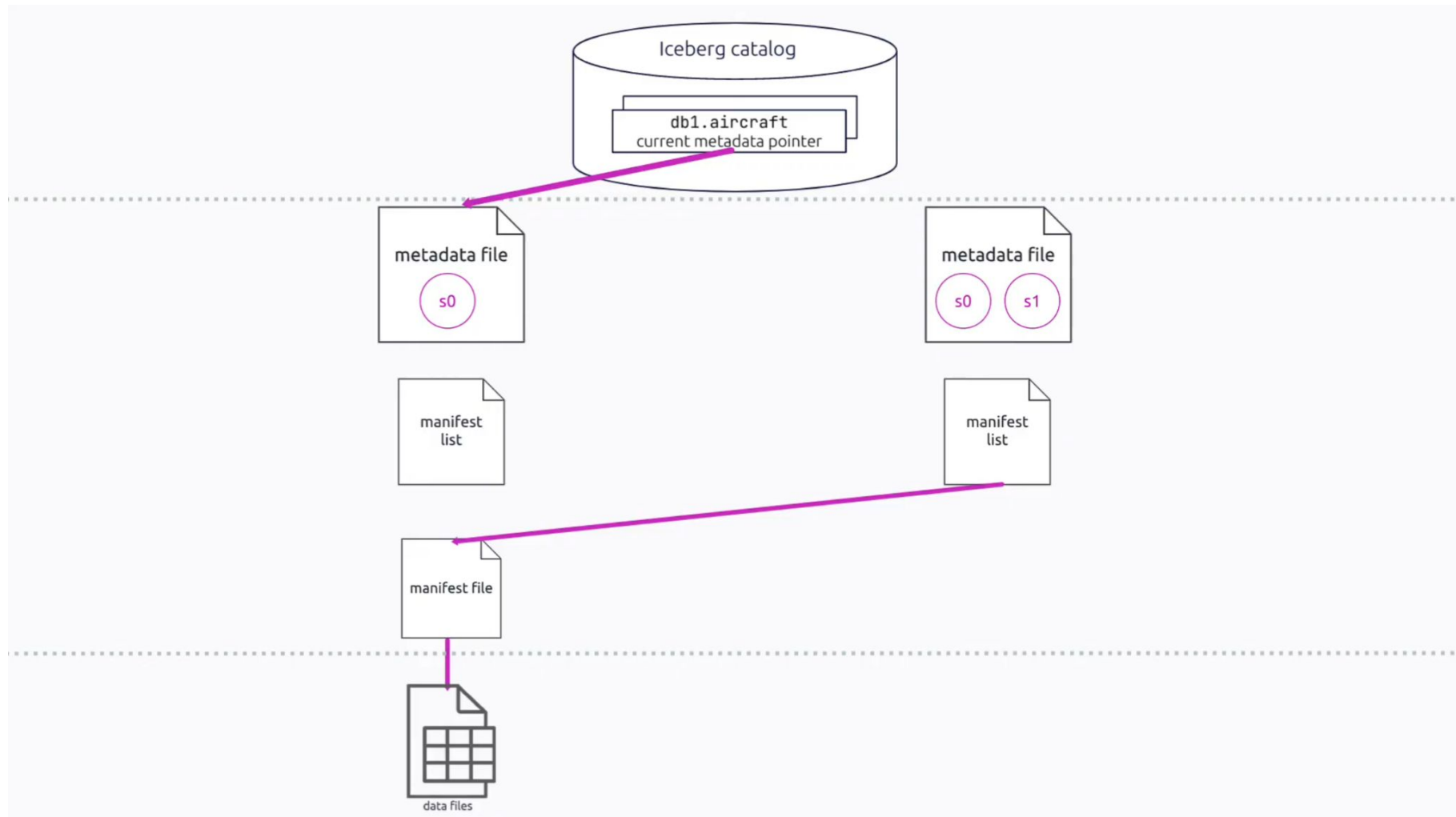
```
  ('N611TV', 'COOL', 'Jet', 1983);
```



data files







Iceberg catalog

db1.aircraft
current metadata pointer

metadata file

s0

manifest
list

manifest file



Data File

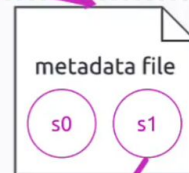
metadata file

s0

s1

manifest
list





Objects (2) [Info](#)

Copy S3 URI

Copy URL

Download

Open

Delete

Actions ▼

Create folder

Upload

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Find objects by prefix

< 1 >

<input type="checkbox"/>	Name ▲	Type ▼	Last modified ▼	Size ▼	Storage class ▼
<input type="checkbox"/>	data/	Folder	-	-	-
<input type="checkbox"/>	metadata/	Folder	-	-	-

Objects

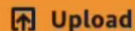
Properties

Objects (1) [Info](#) Copy S3 URI Copy URL DownloadOpen 

Delete

Actions ▼

Create folder



Upload

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#)  to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#) 

 Find objects by prefix

< 1 >



Name



Type




Last modified



Size



Storage class


[2024-0529_152110_2198](#)
[2_9r62a-588d9021-](#)
[5556-437f-a2f0-](#)
[d0f00600f748.parquet](#)

parquet

May 29, 2024, 11:22:37
(UTC-04:00)

658.0 B




Standard

Objects (7) [Info](#)

[Copy S3 URI](#)[Copy URL](#)[Download](#)[Open](#)[Delete](#)[Actions](#)[Create folder](#)[Upload](#)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

[< 1 >](#)

<input type="checkbox"/>	Name ▲	Type ▼	Last modified ▼	Size ▼	Storage class ▼
<input type="checkbox"/>	20240529_152110_2198_2_9r62a-740c70f1-774b-4b49-9067-03e1771f37a8.stats	stats	May 29, 2024, 11:22:38 (UTC-04:00)	1017.0 B	Standard
<input type="checkbox"/>	 29fa1a6e-e0aa-4aab-b9c3-58fa79dcfa44-m0.avro	avro	May 29, 2024, 11:22:38 (UTC-04:00)	6.7 KB	Standard
<input type="checkbox"/>	 snap-2724582809466504793-1-29fa1a6e-e0aa-4aab-b9c3-58fa79dcfa44.avro	avro	May 29, 2024, 11:22:38 (UTC-04:00)	4.2 KB	Standard
<input type="checkbox"/>	 snap-425416382669527773-1-189af56b-cb7a-40b0-a0d1-6a94c3796ca0.avro	avro	May 28, 2024, 15:06:23 (UTC-04:00)	4.0 KB	Standard

Objects (7) [Info](#)

Copy S3 URI

Copy URL

Download

Open

Delete

Actions ▼

Create folder

Upload

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Find objects by prefix

<input type="checkbox"/>	Name ▲	Type ▼	Last modified ▼	Size ▼	Storage class ▼
<input type="checkbox"/>	20240529_152110_2198_2_9r62a-740c70f1-774b-4b49-9067-03e1771f37a8.stats	stats	May 29, 2024, 11:22:38 (UTC-04:00)	1017.0 B	Standard
<input type="checkbox"/>	29fa1a6e-e0aa-4aab-b9c3-58fa79dcfa44-m0.avro	avro	May 29, 2024, 11:22:38 (UTC-04:00)	6.7 KB	Standard
<input type="checkbox"/>	snap-2724582809466504793-1-29fa1a6e-e0aa-4aab-b9c3-58fa79dcfa44.avro	avro	May 29, 2024, 11:22:38 (UTC-04:00)	4.2 KB	Standard
<input type="checkbox"/>	snap-425416382669527773-1-189af56b-cb7a-40b0-a0d1-6a94c3796ca0.avro	avro	May 28, 2024, 15:06:23 (UTC-04:00)	4.0 KB	Standard

Apache iceberg advantages

1) Reliable schema evolution

- add/drop/rename columns safely
- type promotion (where supported)

2) ACID-like consistency for the lake

With “just Parquet”, you typically don’t have atomic commits: readers can see partial writes, and concurrent writers can stomp on each other. Iceberg uses **atomic snapshots**:

- writes commit as a new snapshot or not at all
- readers always see a consistent version
- concurrent writes are coordinated via metadata + catalog

3) Time travel and reproducibility

Iceberg keeps table history (snapshots), so you can:

- query “as of” a timestamp/snapshot
- rollback to a previous version
- reproduce ML features / reports exactly

4) Upserts, deletes, and MERGE that actually work

Parquet files are immutable; “updates” usually mean rewriting partitions and hoping you don’t break consistency.

Iceberg supports:

- row-level deletes (delete files / position deletes)
- MERGE INTO / UPDATE / DELETE patterns (engine support varies but the table format enables it)

5) Faster reads via metadata-driven pruning

Parquet has row-group stats, but without a table layer you still often end up listing tons of files and guessing what to scan.

Iceberg maintains metadata that enables:

- **partition pruning** without directory conventions
- **file pruning** via stats (min/max, null counts, etc.)
- less expensive “list” operations on object stores

8) Multi-engine interoperability (more predictable)

A Parquet directory structure might “work” in Spark but not in Trino/Presto the same way, especially with schema evolution and partition inference.

Iceberg’s spec makes behavior more consistent across engines that support it (Spark, Flink, Trino, Dremio, Athena, etc.).

When “pure Parquet” can be enough

- append-only pipelines with no updates/deletes
- single writer, low concurrency
- you don't need time travel, rollback, or evolving partition strategies