IMPALA learned some new tricks while living on ICEBERG

Zoltán Borók-Nagy

CLOUDERA

Agenda

- Legacy Hive tables
- Iceberg
- Impala
- Iceberg + Impala
- Future work

The Hive table format

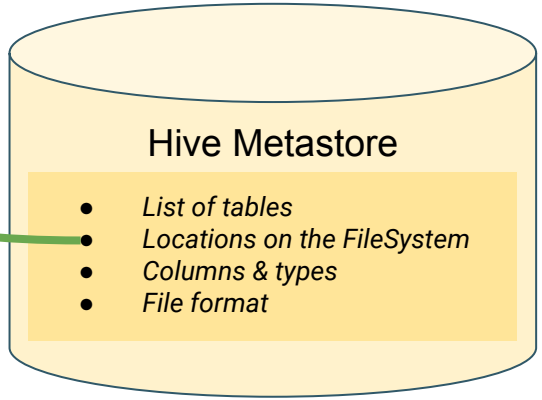
Files in a directory

The table is just a directory on HDFS

Files have a common schema

The schema is stored in the Hive Metastore

```
└── my_table
    ├── 0.csv
    └── 1.csv
```



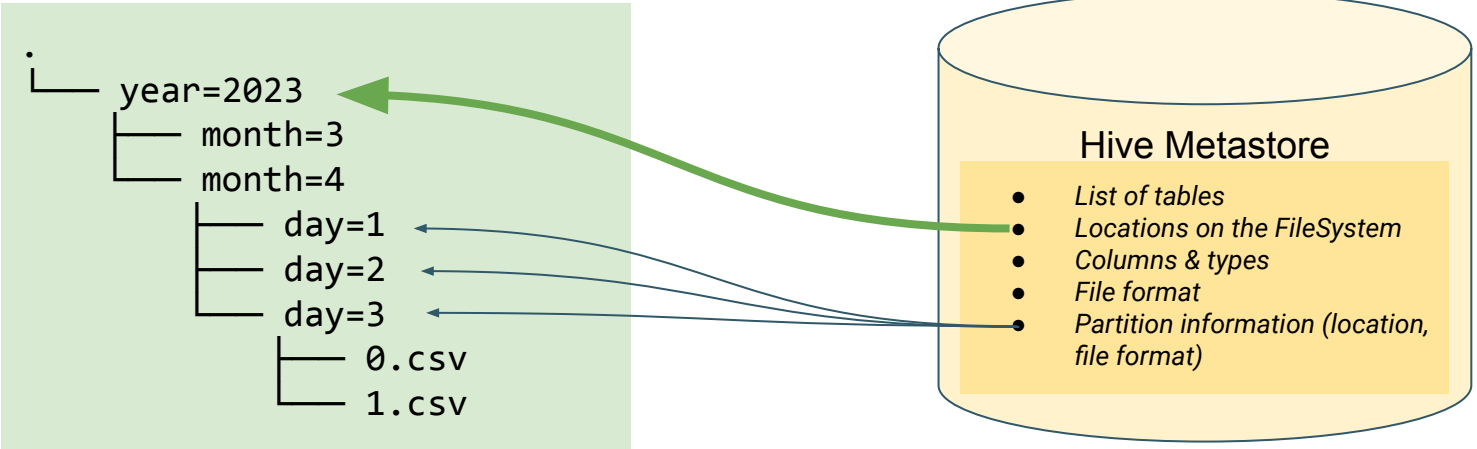
The file list is not stored, it is retrieved from the file system

Tables can be partitioned

Files in a directory

Partition values are encoded in the directory paths

The schema is stored in the Hive Metastore



Problems with Hive tables

Nothing is perfect

- Atomicity & consistency
- Schema evolution is limited
- Partitioning
  - Only value-based partitioning is supported
    - year=2023
  - Partition layout is set in stone
    - To change the partitioning, one needs to rewrite everything
  - Scalability issues with high number of partitions
- Cannot ROLLBACK table state
- No time travel, i.e. no replayable queries
- Row-level modifications are not really feasible
- Small files issue is hard to tackle

Object stores introduced new problems

Like we haven't had enough problems

- Atomic writes/renames are not supported
  - Readers can observe writes in-progress
  - Aborted operations might leave half-written data
- Eventual consistency (*this has been resolved since*)
- Directory listings are expensive

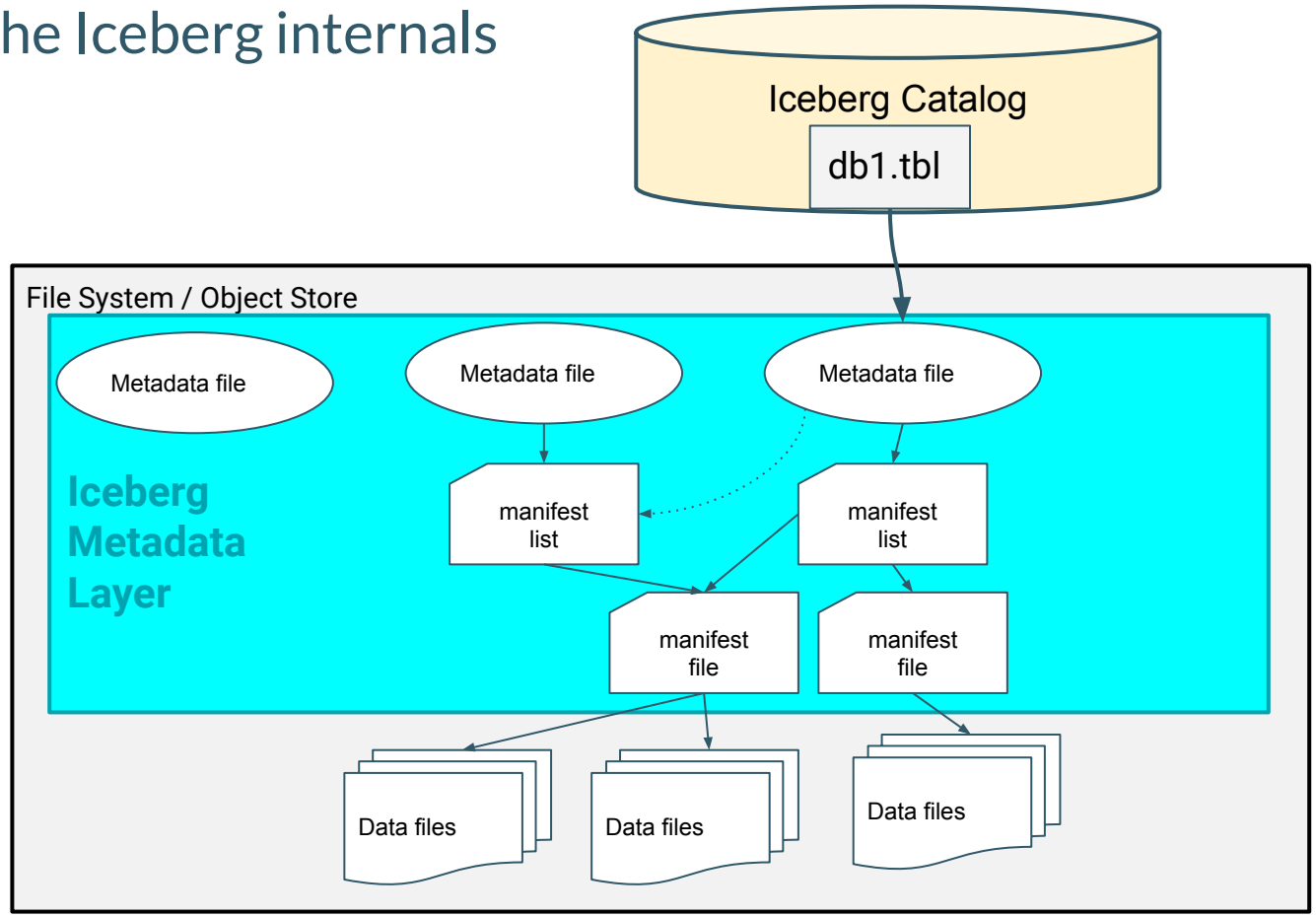
What is Iceberg?

ICEBERG

- It is a table format for huge data sets. So it defines how
  - metadata is stored
  - data files can be organized

- It is also a library
  - Compute engines can use this library to directly read/write the table
  - No mediator component required

Apache Iceberg internals

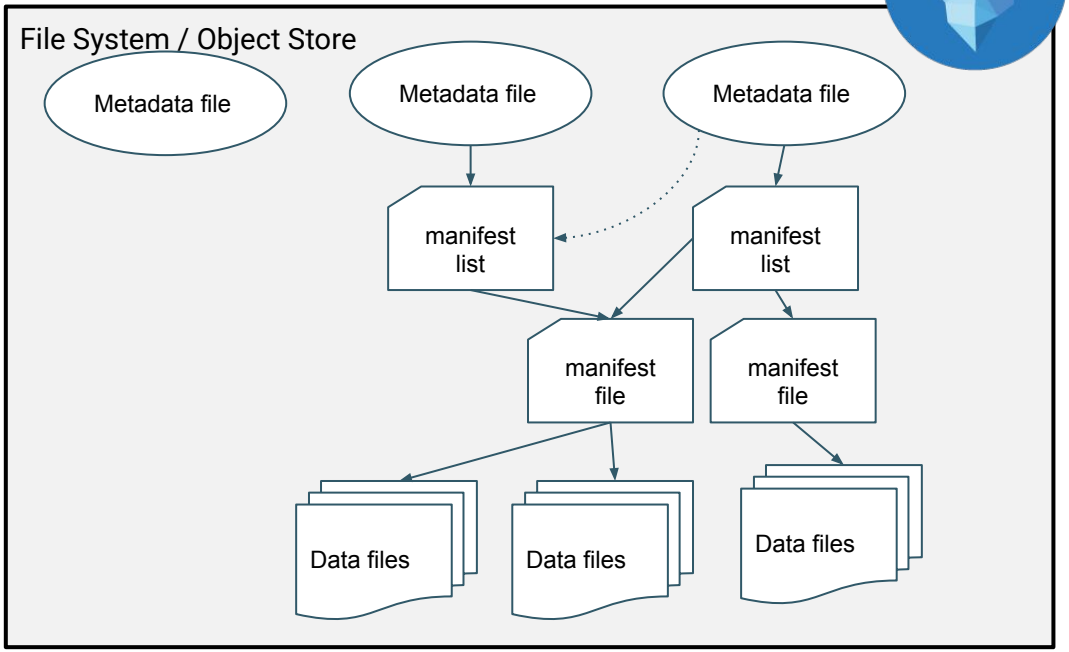


ACID guarantees

ICEBERG

Snapshot isolation

- All files are immutable
- Readers always read a consistent snapshot
- Updates create a new snapshot
  - Atomic operation, using optimistic concurrency
- Time-travel queries



Schema evolution

ICEBERG

- Schema elements get unique field ids:
  - “ID”: INT : **1**
  - “First Name” : STRING : **2**
  - “Last Name” : STRING : **3**
- Iceberg field ids are written to the data files’ metadata as well
  - File scanners retrieve columns from data files based on their field id
- Field ids remain unchanged on schema evolution
  - Columns can be added / dropped / renamed / reordered

Flexible partitioning

ICEBERG

- Partition by transformations
  - *IDENTITY, TRUNCATE, BUCKET, YEAR, MONTH, DAY, HOUR*
  - Now it's possible to partition based on high-NDV columns

- Partitioning is "hidden"
  - Partition information is stored in the Iceberg metadata layer
  - No need to explicitly write partition columns (YEAR, MONTH, etc.)
  - No need to add extra predicates to queries for partition pruning

```
SELECT * FROM tbl WHERE ts = '2023-04-21 20:56:08'
AND YEAR = 2023 AND MONTH = 4 AND DAY = 21;
```

    - All this can be automatically extracted from **'2023-04-21 20:56:08'** If the table is partitioned by **DAY(ts)**

Partition evolution

No other table format can do this trick

- Partition information is stored in Iceberg metadata layer
  - Not in the directory structure
- It's possible to just change the partitioning completely
  - Or just refine existing partitioning
- And write new data based on the new partition layout

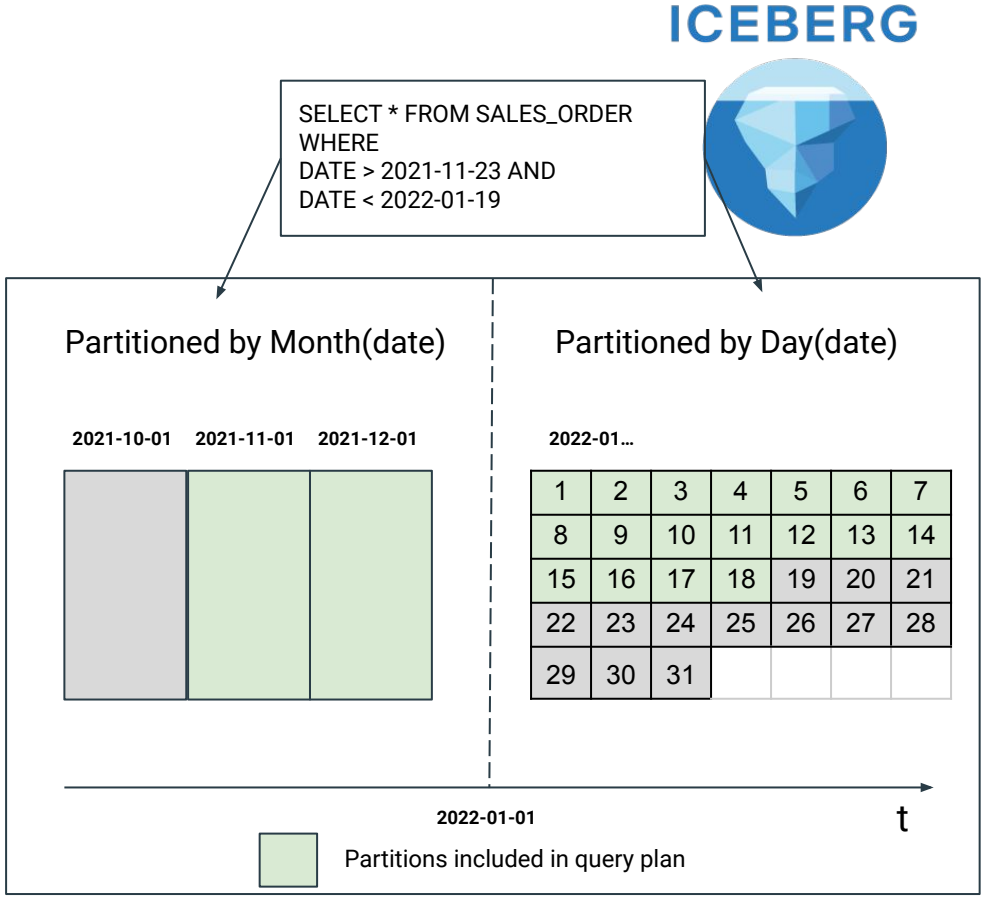


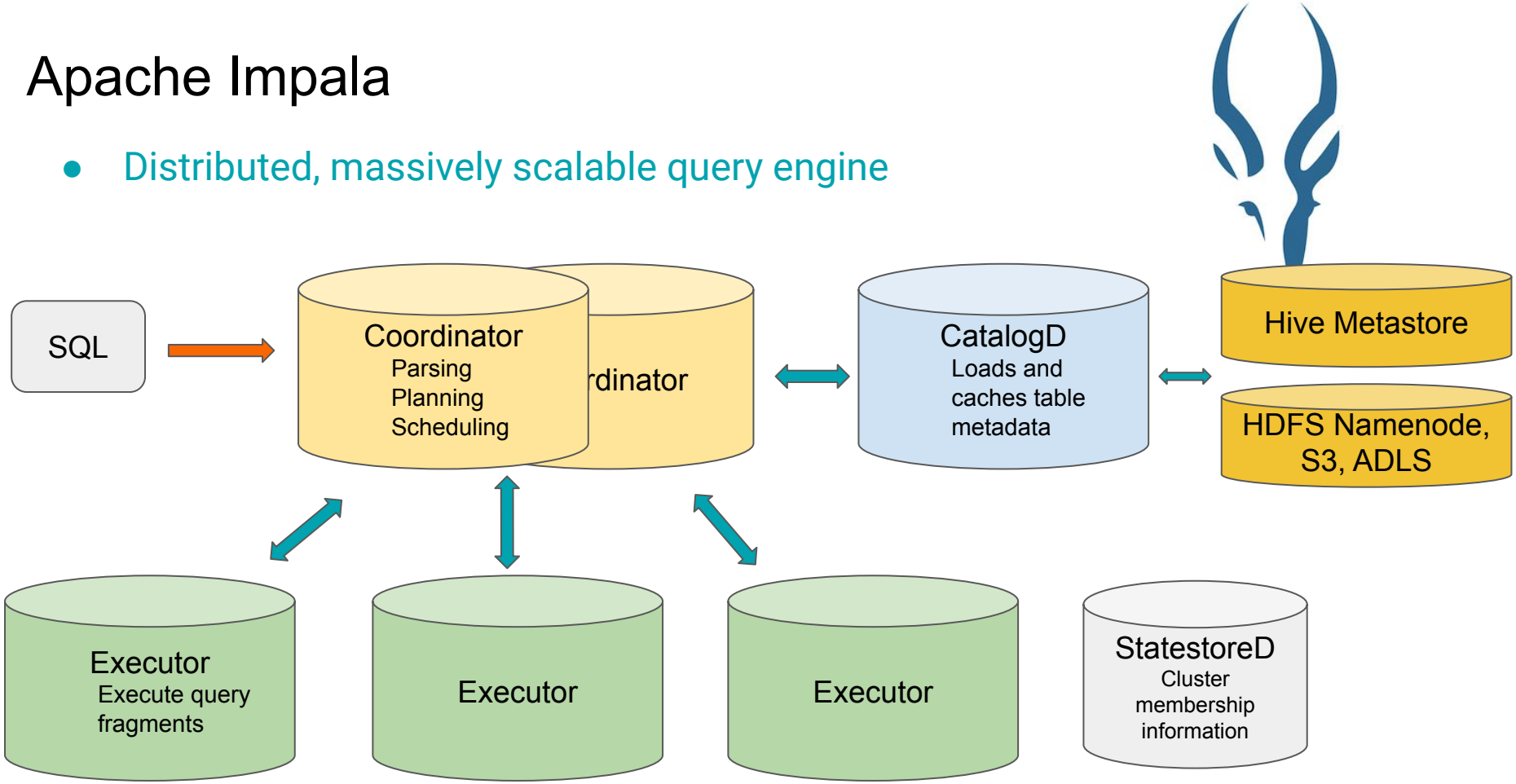
Table maintenance

ICEBERG

- Rollback
  - Restore earlier state
- Expire old snapshots (GDPR, CCPA)
- Compaction
  - Fix small files issues
  - Eliminate delete delta files

Apache Impala

- Distributed, massively scalable query engine



About Impala

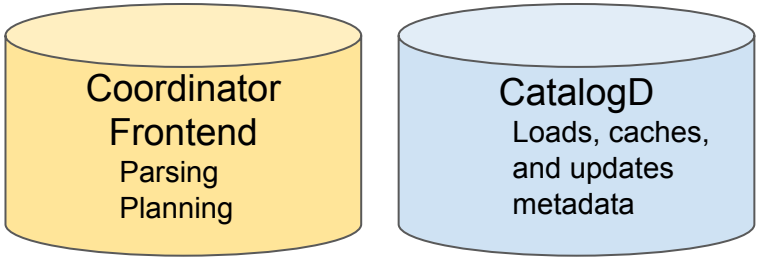
- Frontend (query parsing, planning) is written in **Java**
- Backend (query execution) is written in **C++**
- Optimized for query performance
  - Completely pipelined, no checkpointing
  - Caches table metadata
  - Caches remote reads
  - Code generation via LLVM to speed up queries
- Limited write capabilities **in the past**

About Impala

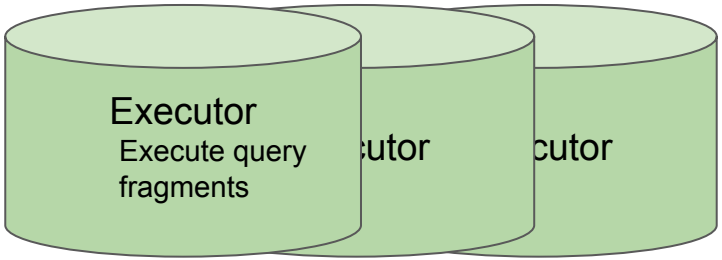
- Various storage systems and file formats
  - HDFS, Ozone, S3, ADLS, basically anything HDFS-client compatible
  - Parquet, ORC, Avro, JSON, text
- Different authentication methods
  - LDAP, Kerberos, SAML, JWT
- Fine-grained authorization policies (row filtering, column masking)
  - Via Apache Ranger
- Admission control to limit the number of concurrent queries
- Spilling operators to execute queries in a given memory limit
- And a lot more…

ICEBERG

Why Impala + Iceberg integration is special?



- Written in Java
- Interacts with Iceberg API
  - loadTable()
  - planFiles() - predicate pushdown
  - appendFiles(), commit(), etc.



- Highly optimized C++ code
- Uses Impala's own scanners / writers
  - Field-id resolution
- Specific operators to deal with V2 tables
  - Reading/writing position delete files

CREATE Iceberg tables


```
CREATE TABLE ice_t (i INT, s STRING)
STORED BY ICEBERG;
```

```
CREATE EXTERNAL TABLE ice_t STORED BY ICEBERG
TBLPROPERTIES ('iceberg.catalog'='my_catalog',
               'iceberg.table_identifier'='my.tbl');
```

CREATE partitioned Iceberg tables


```
CREATE TABLE ice_t (i INT, s STRING)
PARTITIONED BY SPEC (i)
STORED BY ICEBERG;
```

```
CREATE TABLE ice_t (i INT, s STRING)
PARTITIONED BY SPEC (TRUNCATE(3, s))
STORED BY ICEBERG;
```

- IDENTITY
- BUCKET(N, col)
- TRUNCATE(N, col)
- YEAR(col)
- MONTH(col)
- DAY(col)
- HOUR(col)

Reading Iceberg tables

- **Arbitrary SELECT queries**

```
SELECT * FROM ice_t;

SELECT * FROM ice_t JOIN non_ice_t ON(<cond>);
```

Predicates are pushed down to Iceberg

- **Time-travel**

Uses table schema and table data of the specified time / version

```
SELECT * FROM ice_t FOR SYSTEM_TIME AS OF '2023-09-27 11:48:12';
SELECT * FROM ice_t FOR SYSTEM_TIME AS OF now() - interval 5 days;

SELECT * FROM ice_t FOR SYSTEM_VERSION AS OF 123456;
```

Ingesting data into Iceberg tables

- **INSERT INTO**

```
INSERT INTO ice_t VALUES (1, 2);
INSERT INTO ice_t SELECT * FROM other_t;
```

- **INSERT OVERWRITE**

```
INSERT OVERWRITE ice_t VALUES (1, 2);
INSERT OVERWRITE ice_t SELECT * FROM other_t;
```

- **LOAD DATA**

```
LOAD DATA INPATH 'file_or_directory_path' [OVERWRITE]
INTO TABLE tablename;
```

- **Hidden partitioning**
  - No need for PARTITION() clause
- **Impala writes minimum number of data files via shuffling data based on partitioning**

TRUNCATE Iceberg tables

- **Delete all records:**

  `TRUNCATE` creates new empty snapshot

  ```
  TRUNCATE TABLE ice_t;
  ```

  One can still retrieve old data via time travel

Row-level modifications

ICEBERG

DELETE / UPDATE existing records (GDPR, Data correction)

Merge-on-read

- Iceberg V2-only
- Writes delete files
  - Contain information about deleted rows
- Low write amplification
- High read amplification
- Table readers need to exclude deleted rows from result
- Useful for small amount of deletes

Copy-on-write

- Replace old data files with new ones
- High write amplification
- No read amplification
- Useful for rewriting lots of data

DELETE FROM Iceberg tables

Only for Iceberg V2 tables

```
TBLPROPERTIES ('format-version'='2');
```

Only Merge-on-read is supported

Position deletes

```
DELETE FROM ice_t WHERE c1 = 100;

DELETE t1 FROM ice_t t1 JOIN other_t t2 ON t1.x = t2.x;
```

Impala writes as few delete files as possible, typically one per partition

ALTERing Iceberg tables

- **Rename the whole table:**

```
ALTER TABLE ... RENAME TO ...
```

- **Schema evolution**

```
ALTER TABLE ... CHANGE COLUMN ...

ALTER TABLE ... ADD COLUMNS ...

ALTER TABLE ... DROP COLUMN ...
```

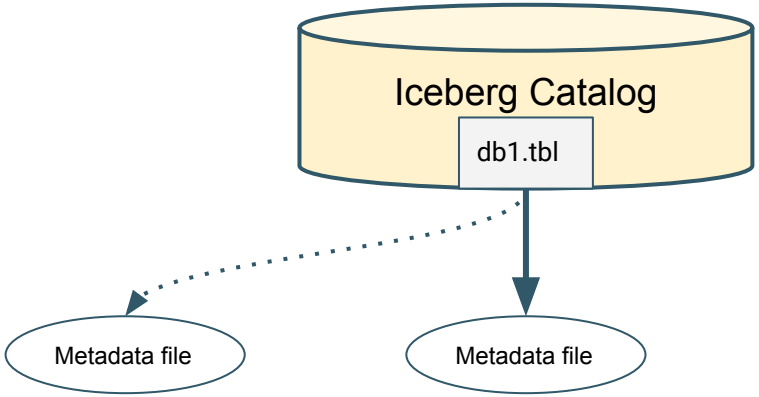
- **Partition evolution**

```
ALTER TABLE ice_p SET PARTITION SPEC (TRUNCATE(3, s), HOUR(t), i);
```

ROLLBACK

```
ALTER TABLE ice_t EXECUTE ROLLBACK(3088747670581784990);
```

- If the older snapshot is available, it just restores that state of the table

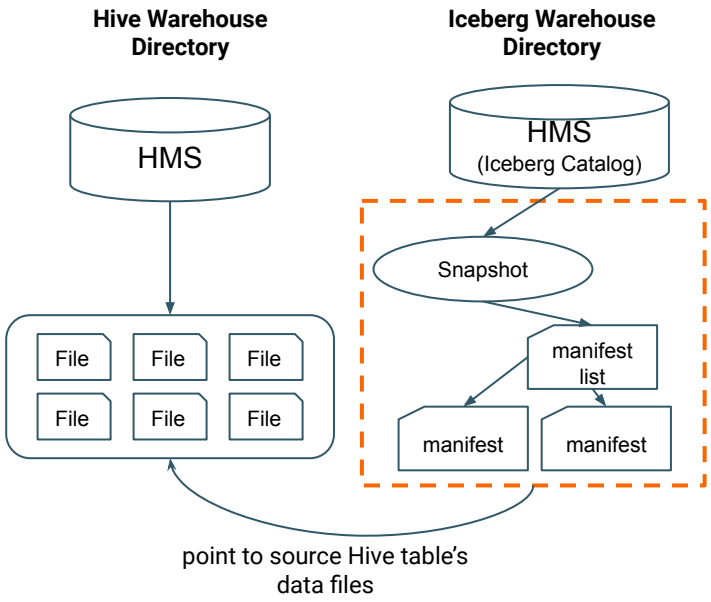


EXPIRE SNAPSHOTS


```sql
ALTER TABLE ice_t
EXECUTE expire_snapshots('2022-01-04 10:00:00');
```

- Expires old snapshots, i.e. removes metadata files and data files that are only pointed by them
- respects the minimum number of snapshots to keep: history.expire.min-snapshots-to-keep table property.

Table Migration In-Place

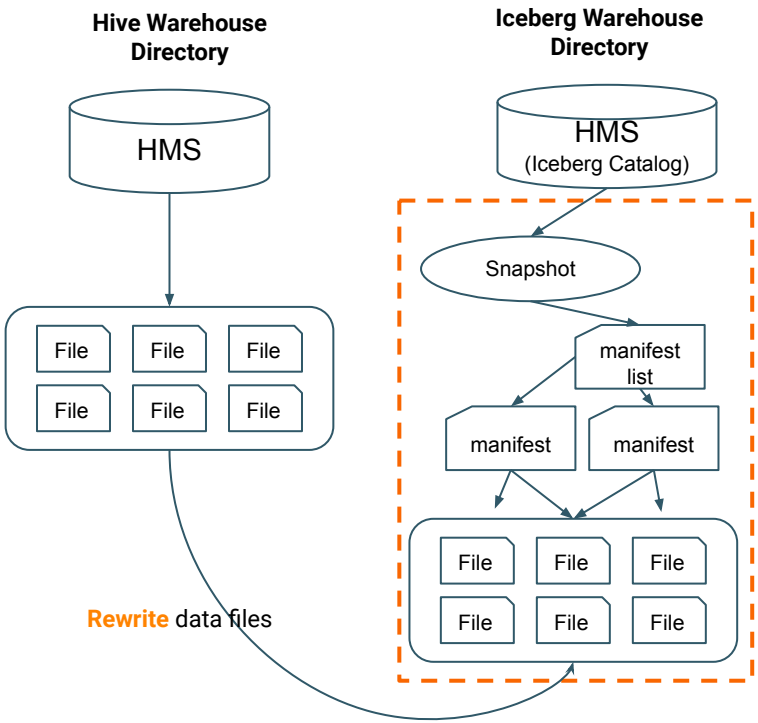


Avoids rewriting data files, just **write** the metadata.

Partition layout remains unchanged.

```
ALTER TABLE tbl CONVERT TO ICEBERG;
```

Table Migration **CTAS**



Data files **recreated** in addition to creation of Iceberg tables and corresponding metadata

```
CREATE TABLE ctas
PARTITIONED BY SPEC(z)
STORED BY ICEBERG AS
SELECT x, y, z FROM t;
```

Near-future work

- **UPDATE** statement (then eventually **MERGE**)
- Querying metadata of tables (history, files, partitions, snapshots)
- **OPTIMIZE** - compaction!
- **Reading** tables with equality deletes
  - *Writing equality delete files is not planned*
- Branching / tagging
- Column stats in Puffin

Limitations (at the time of writing)

- Copy-on-write
  - OTOH our merge-on-read is *very efficient*
  - OPTIMIZE (compaction) is coming soon
- Can only write Parquet files
- Following types are not supported currently:
  - TIMESTAMPTZ type
    - But Impala is able to correctly read such tables
  - UUID type
  - TIME
  - Fixed(L)
- **We might eventually add support for the above**
  - *Contributions are welcome!*

Questions?