Crunch Budapest October 2022

Apache Iceberg Merge-on-Read Streaming CDC

Victoria Bukta (Staff Data Eng)

shopify

AGENDA

Lakehouse Team’s Mission

“Provide Shopify developers with an interoperable,

performant, and standards-based data lakehouse

***where 1st-** and 3rd-party **Shopify data can be ingested**”*

Table Snapshot

The state of a datasource at a specific moment in time.

- One row per a primary key
- Latest version of a primary key

Legacy System

Batch data ingestion

*via **statement based replication***

Legacy System

"statement based replication"

- ➔ Query DB for data
  - **Does not scale as tables get larger**
  - Long running queries (volatile)

SELECT * FROM my_table

Legacy System

"statement based replication"

- ➔ Query DB for data
  - ◆ **Does not scale as tables get larger**
  - ◆ Long running queries (volatile)
- ➔ Done incrementally by keeping position
- ➔ Depends on app devs updating updated_at
  - ◆ **Possible to miss updates**

```
SELECT * FROM my_table
    WHERE updated_at > 2020/10/01 09:00:00
```

Legacy System

"statement based replication"

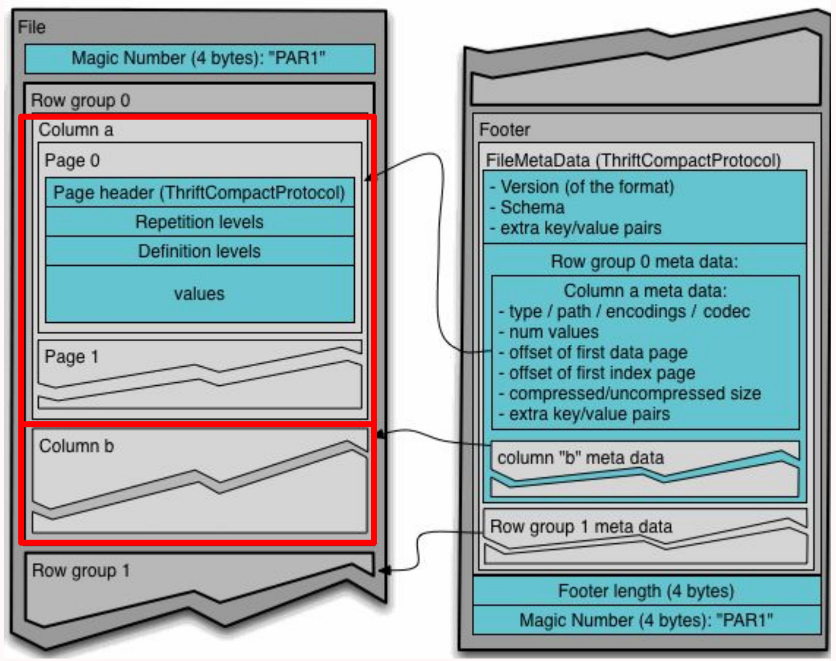
- ➔ Query DB for data
  - ◆ **Does not scale as tables get larger**
  - ◆ Long running queries (volatile)
- ➔ Done incrementally by keeping position
- ➔ Depends on app devs updating updated_at
  - ◆ **Possible to miss updates**
- ➔ Smaller queries using bucketing
  - ◆ Increased network requests increase time
- ➔ **Can't capture deletes**

```
SELECT * FROM my_table
    WHERE updated_at > 2020/10/01 09:00:00
    AND primary_key >= last_seen_key
    ORDER BY primary_key
LIMIT 10,000
```

Legacy System

“statement based replication” contd.

- ➔ **Spark process to apply update (⏳💰)**
  - ◆ **Due to columnar file format**
  - ◆ Optimizing for aggregation analytics over a subset of columns
  - ◆ Efficient compaction (schematized data)
  - ◆ **Columnar files are immutable (overwrite)**
    - ● Rewrite is an expensive operation



File
Magic Number (4 bytes): "PAR1"
Row group 0
Column a
Page 0
Page header (ThriftCompactProtocol)
Repetition levels
Definition levels
values
Page 1
Column b
Row group 1
Footer
FileMetaData (ThriftCompactProtocol)
- Version (of the format)
- Schema
- extra key/value pairs
Row group 0 meta data:
Column a meta data:
- type / path / encodings / codec
- num values
- offset of first data page
- offset of first index page
- compressed/uncompressed size
- extra key/value pairs
column "b" meta data
Row group 1 meta data
Footer length (4 bytes)
Magic Number (4 bytes): "PAR1"

Design Goals

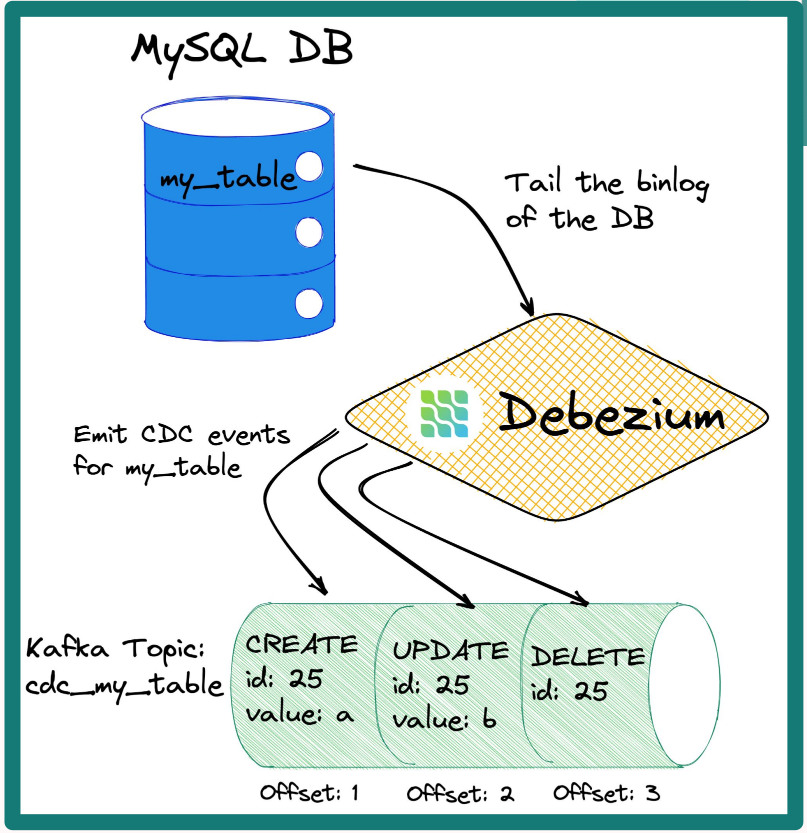
- ❏ Scalable ingestion
- ❏ Scalable **snapshot production**
- ❏ More accurate updates
- ❏ Delete capture
- ❏ Snapshot SLO of under an hour

Design

*Streaming data ingestion of **change data capture** via **Kafka** written in **Iceberg V2 storage format***

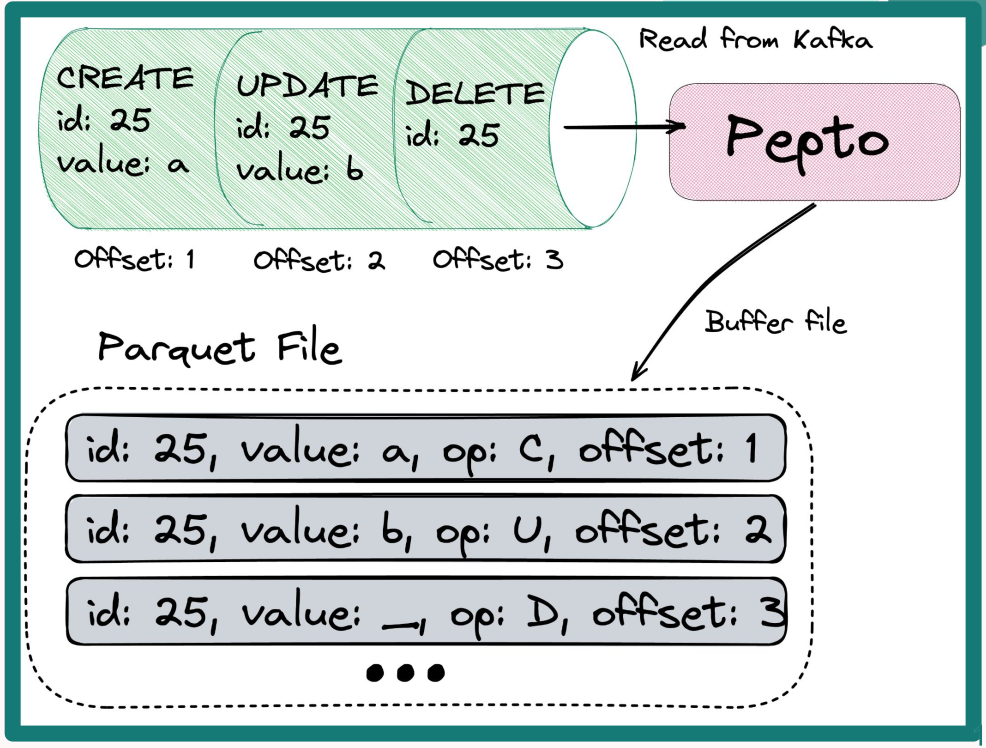
Design : Kafka ingestion + CDC

- Capturing changes from our source
  - Binlog from MySQL captures every transaction
  - **Every CREATE, UPDATE, DELETE**
    - **Requires performing an upsert**
  - Ordered based on when the event happened
  - Binlog is also used for DB replication
- State changes are emitted to Kafka for future ingestion



Design : Kafka ingestion + CDC

- Consume the CDC events from Kafka
- Buffer events into Parquet files
  - Registered in an Iceberg Table
- Achievements
  - **Scalability w/ Kafka partitions + multiple consumers / writers**
  - **More accurate updates**
  - **Delete capture**
  - **~7 min SLO data ingest**

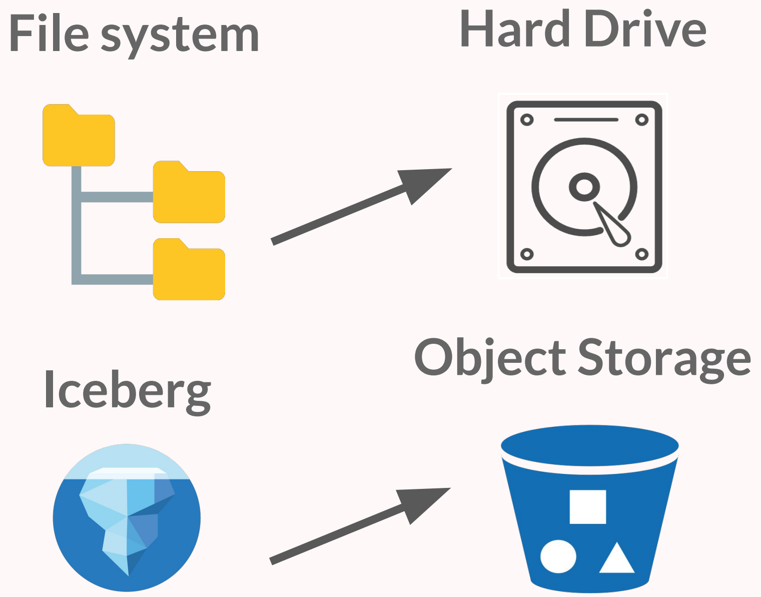


Goals

- ✓ Scalable ingestion
- ❑ Scalable **snapshot production**
- ✓ More accurate updates
- ✓ Delete capture
- ❑ Snapshot SLO of under an hour

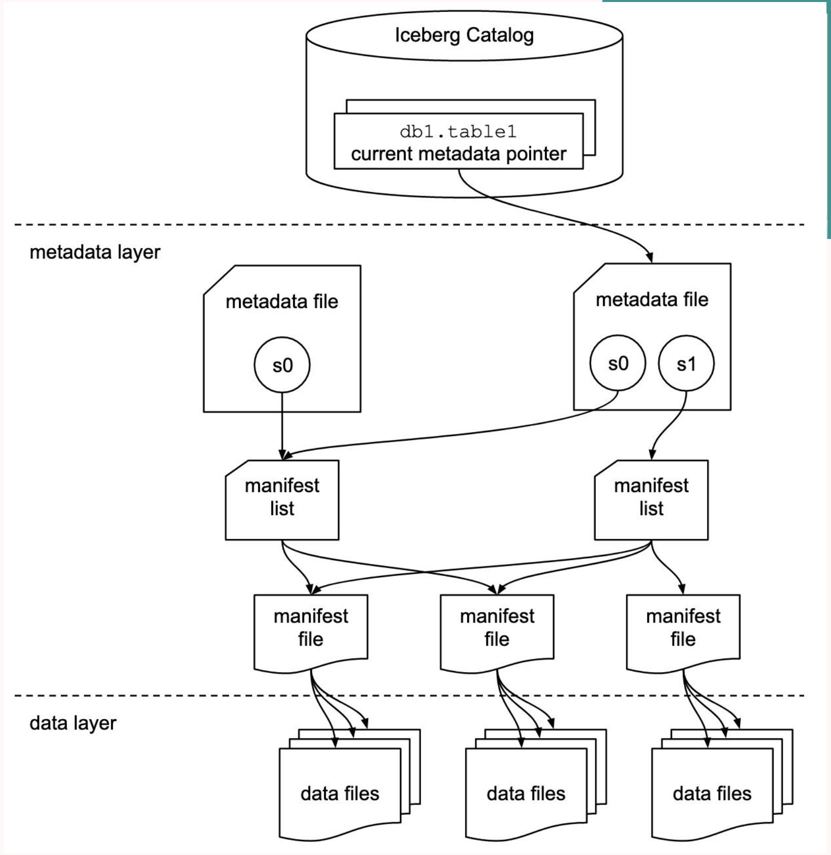
Design : Iceberg

- Iceberg is a table format
- Just a library
- Contents of a table are identified by traversing through metadata files



Design : Iceberg

- Iceberg is a table format
- Just a library
- Contents of a table are identified by traversing through metadata files



Design : Iceberg

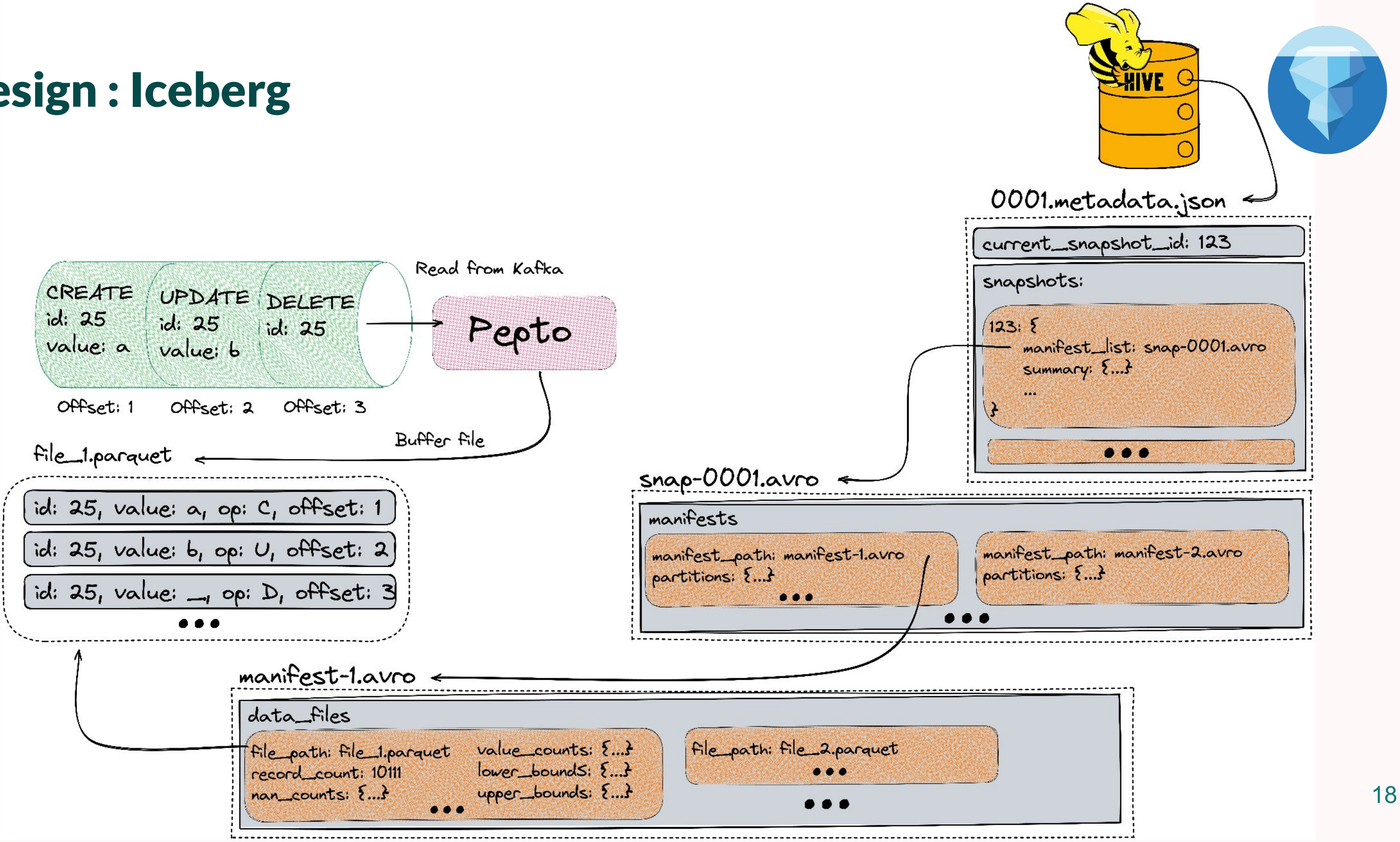
- Iceberg is **metadata rich table format** on top of parquet files
- Enables more **efficient file pruning**
  - More scalable reads
  - More scalable writes

efficient read & writes

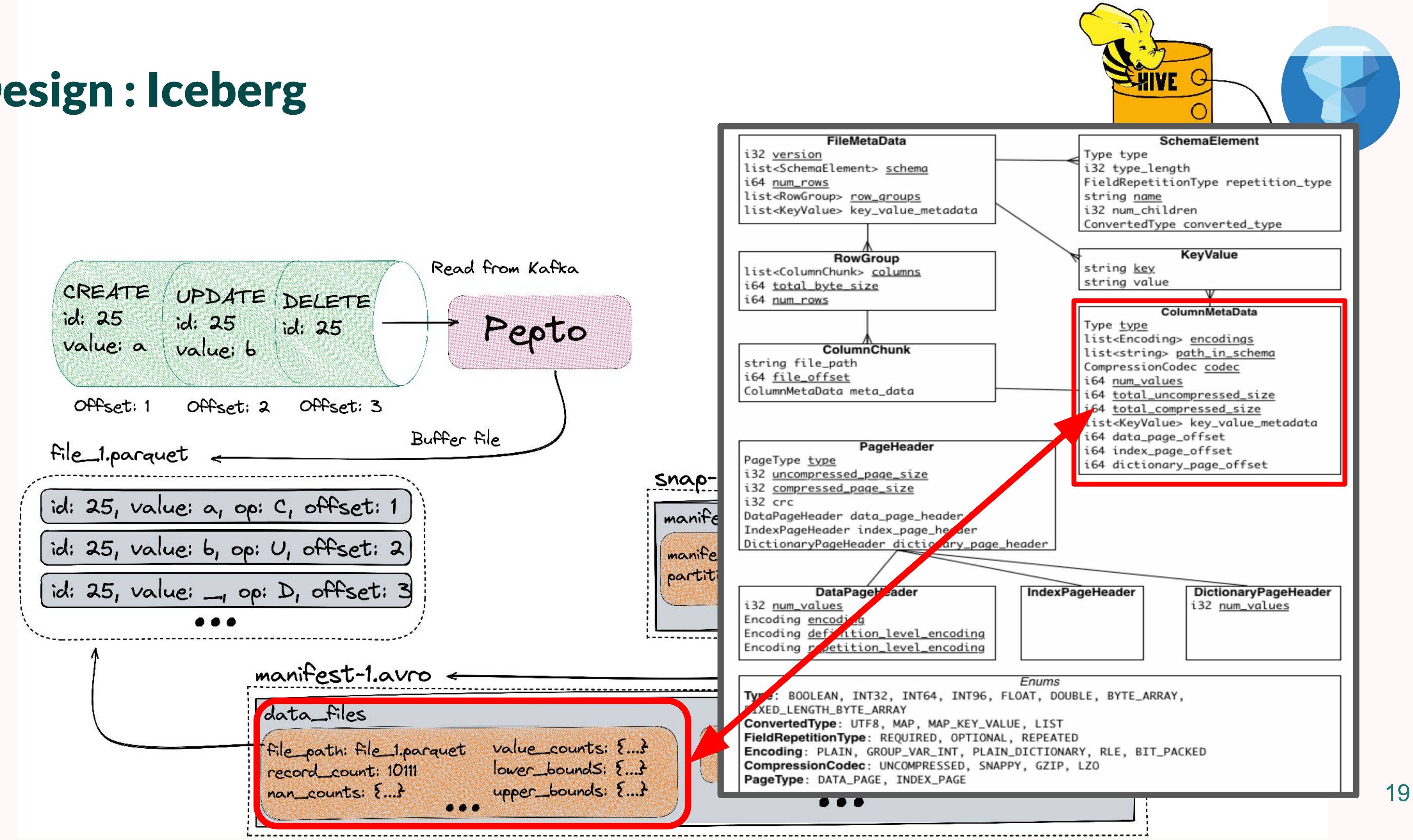
$\longrightarrow$

scalable snapshot production & 1h< SLO

Design : Iceberg



Design : Iceberg



```
FileMetaData
i32 version
list<SchemaElement> schema
i64 num_rows
list<RowGroup> row_groups
list<KeyValue> key_value_metadata

SchemaElement
Type type
i32 type_length
FieldRepetitionType repetition_type
string name
i32 num_children
ConvertedType converted_type

RowGroup
list<ColumnChunk> columns
i64 total_byte_size
i64 num_rows

KeyValue
string key
string value

ColumnChunk
string file_path
i64 file_offset
ColumnMetaData meta_data

ColumnMetaData
Type type
list<Encoding> encodings
list<string> path_in_schema
CompressionCodec codec
i64 num_values
i64 total_uncompressed_size
i64 total_compressed_size
list<KeyValue> key_value_metadata
i64 data_page_offset
i64 index_page_offset
i64 dictionary_page_offset

PageHeader
PageType type
i32 uncompressed_page_size
i32 compressed_page_size
i32 crc
DataPageHeader data_page_header
IndexPageHeader index_page_header
DictionaryPageHeader dictionary_page_header

DataPageHeader
i32 num_values
Encoding encoding
Encoding definition_level_encoding
Encoding repetition_level_encoding

IndexPageHeader

DictionaryPageHeader
i32 num_values
```

Enums

Type: BOOLEAN, INT32, INT64, INT96, FLOAT, DOUBLE, BYTE_ARRAY, FIXED_LENGTH_BYTE_ARRAY
ConvertedType: UTF8, MAP, MAP_KEY_VALUE, LIST
FieldRepetitionType: REQUIRED, OPTIONAL, REPEATED
Encoding: PLAIN, GROUP_VAR_INT, PLAIN_DICTIONARY, RLE, BIT_PACKED
CompressionCodec: UNCOMPRESSED, SNAPPY, GZIP, LZO
PageType: DATA_PAGE, INDEX_PAGE

Design

Kafka Injection

+

CDC

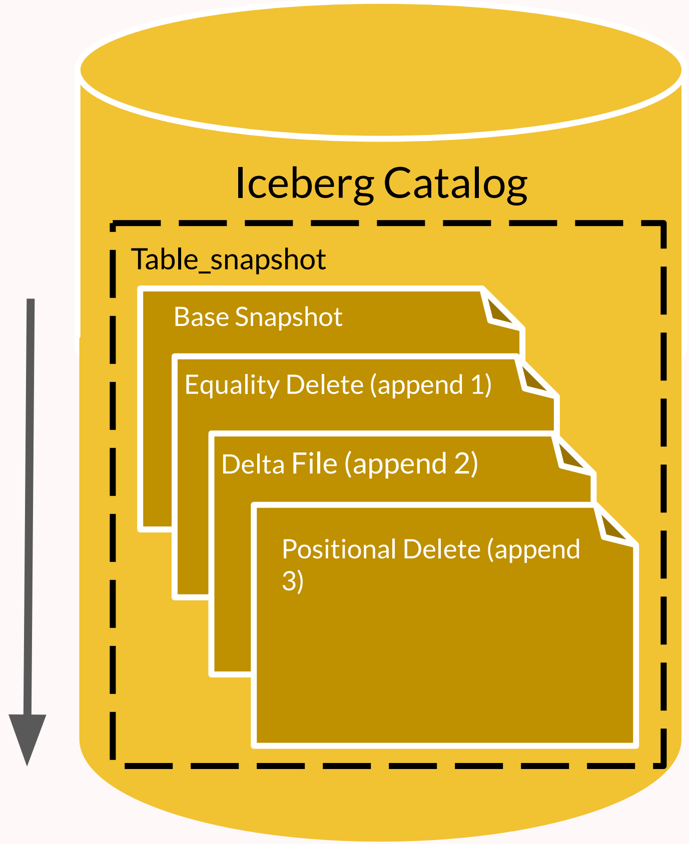
+

Merge-on-Read Iceberg

(aka Iceberg V2 storage)

Design : Iceberg V2 storage

- V2 Spec introduces delete files
  - Positional Delete
  - Equality Delete
- Act as filters at query time
- **Procrastinate rewriting files**
- Targeted rewrite (via Iceberg metadata)
  - **Reduced compute**



Design : Iceberg V2 storage

- **Procrastinate rewriting files**
- Targeted rewrite (via Iceberg metadata)
  - **Reduced compute**

Targeted upsertes

→

scalable snapshot production & 1h< SLO

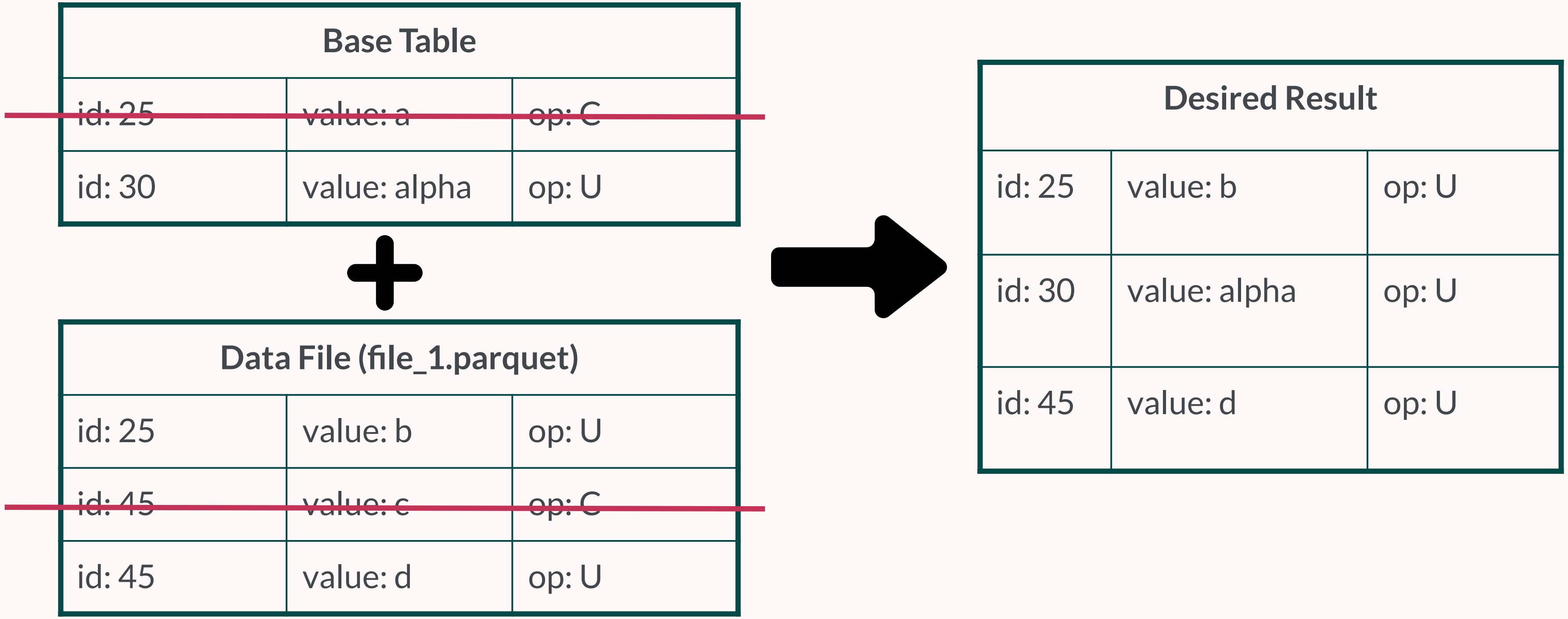
Design : Iceberg V2 storage – Our Goal

Base Table		
id: 25	value: a	op: C
id: 30	value: alpha	op: U

Data File (file_1.parquet)		
id: 25	value: b	op: U
id: 45	value: c	op: C
id: 45	value: d	op: U

Desired Result		
id: 25	value: b	op: U
id: 30	value: alpha	op: U
id: 45	value: d	op: U

Design : Iceberg V2 storage – Our Goal



Base Table		
~~id: 25~~	~~value: a~~	~~op: C~~
id: 30	value: alpha	op: U

+

Data File (file_1.parquet)		
id: 25	value: b	op: U
~~id: 45~~	~~value: c~~	~~op: C~~
id: 45	value: d	op: U

$\rightarrow$

Desired Result		
id: 25	value: b	op: U
id: 30	value: alpha	op: U
id: 45	value: d	op: U

Design : Iceberg V2 storage – Step 1: Append Equality Delete

Base Table		
id: 25	value: a	op: C
id: 30	value: alpha	op: U

+

Append - Equality Delete File
id: 25
id: 45

→

Result		
id: 30	value: alpha	op: U

Design : Iceberg V2 storage – Step 1: Append Equality Delete

Base Table		
~~id: 25~~	~~value: a~~	~~op: C~~
id: 30	value: alpha	op: U

+

Append - Equality Delete File
id: 25
id: 45

→

Result		
id: 30	value: alpha	op: U

❖ **Row gets filtered out at query time**

Design : Iceberg V2 storage – Step 2: Append Data File

Previous Result		
id: 30	value: alpha	op: U

+

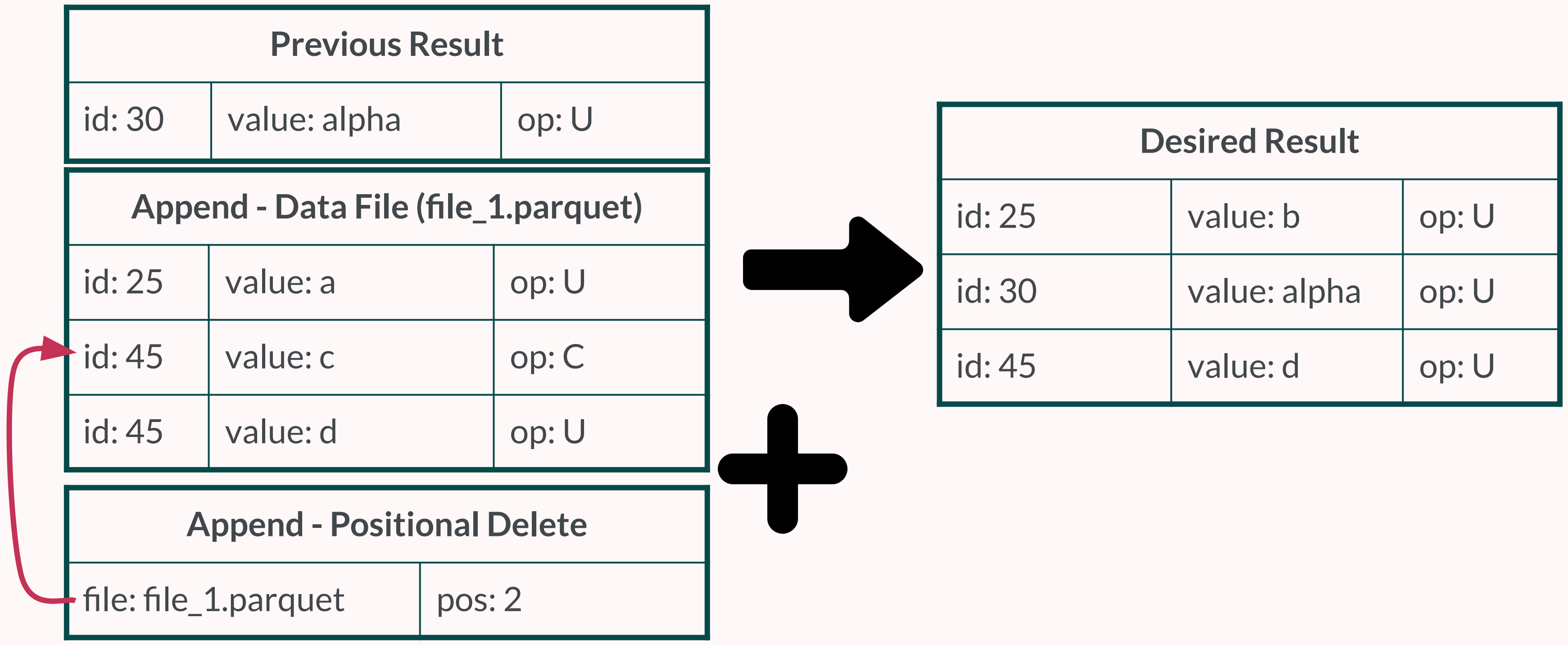
Append - Data File (file_1.parquet)		
id: 25	value: a	op: U
id: 45	value: c	op: C
id: 45	value: d	op: U

Previous Result		
id: 30	value: alpha	op: U

Append - Data File (file_1.parquet)		
id: 25	value: a	op: U
id: 45	value: c	op: C
id: 45	value: d	op: U

❖ **Duplicates within the datafile remain**

Design : Iceberg V2 storage – Step 3: Append Positional Delete



Previous Result		
id: 30	value: alpha	op: U

Append - Data File (file_1.parquet)		
id: 25	value: a	op: U
id: 45	value: c	op: C
id: 45	value: d	op: U

Append - Positional Delete	
file: file_1.parquet	pos: 2

Desired Result		
id: 25	value: b	op: U
id: 30	value: alpha	op: U
id: 45	value: d	op: U

Design : Iceberg V2 storage – Step 3: Append Positional Delete

Previous Result		
id: 30	value: alpha	op: U

Append - Data File (file_1.parquet)		
id: 25	value: a	op: U
~~id: 45~~	~~value: c~~	~~op: C~~
id: 45	value: d	op: U

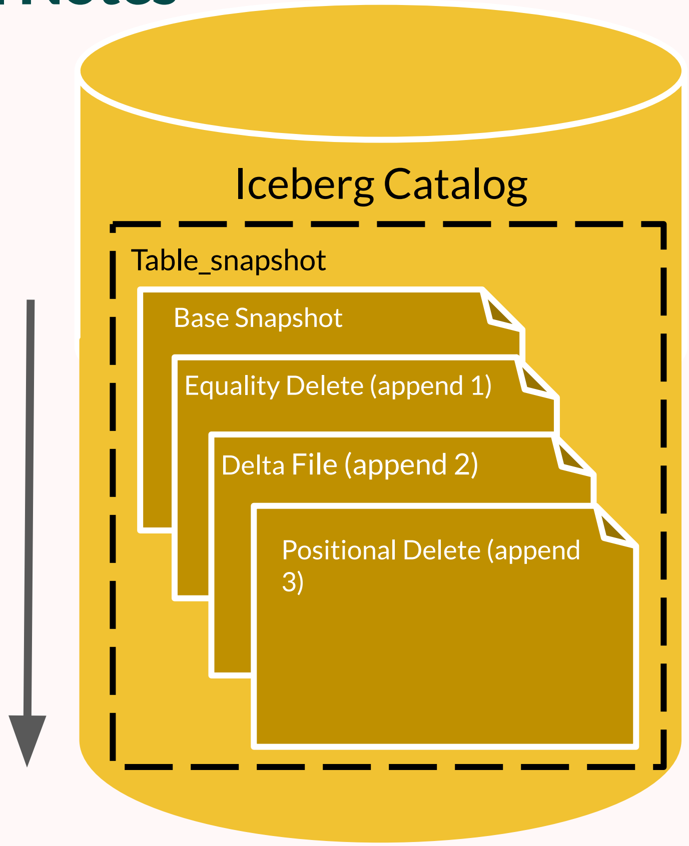
Append - Positional Delete	
file: file_1.parquet	pos: 2

Desired Result		
id: 25	value: b	op: U
id: 30	value: alpha	op: U
id: 45	value: d	op: U

❖ **Duplicates within the datafile are filtered out**

Design : Iceberg V2 storage – Additional Notes

- Delete files effects performance
  - Positional Deletes → fast
  - Equality Deletes → slow
- Lots of small files
- **Regular maintenance** required to optimize the table
  - **Rewrite data + delete files**



Results

< 7min

Ingestion Time

3-40 min

Snapshot Compaction Time

Results

<7min

Ingestion Time

3-40 min

Snapshot Compaction Time

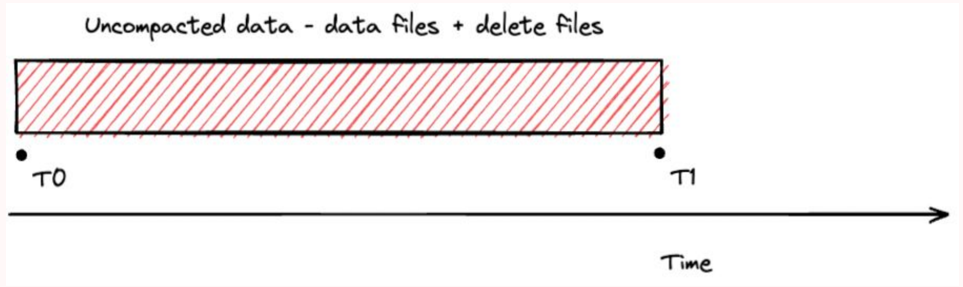
- **Lots more levers** (weigh cost vs performance)
  - Parallel partition rewrites
  - Max file group size (limit the input data to rewrite)
  - Machine size

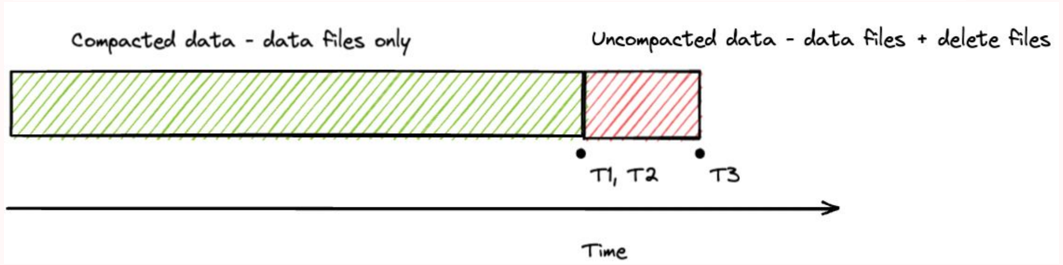
Goals

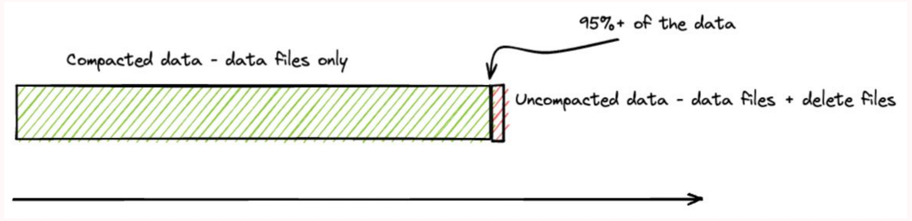
- ✓ Scalable ingestion
- ✓ Scalable **snapshot production**
- ✓ More accurate updates
- ✓ Delete capture
- ✓ Snapshot SLO of under an hour

Reflection

- **Two ingestion modes**
  - Initial full state dump
  - Steady state continuous ingestion
- Two categories of tables
  - **Fast** to query the **uncompacted**
  - **Slow** to query the **uncompacted**
- Performance cost of keeping delete files around
- **Aggressive rewriting**




Compacted data - data files only
Uncompacted data - data files + delete files
T1, T2
T3
Time


95%+ of the data
Compacted data - data files only
Uncompacted data - data files + delete files

Reflection

- Very happy with compaction times + performance
  - **Having more levers is important**
- **Fast vs Slow uncompacted tables**
  - Different tiers of tables
  - Hiding uncompacted tables behind a view or tag
- **Iceberg's abstraction hides how data is represented on disk**, which makes this possible
- Increased complexity of moving to streaming

Thank You!

shopify