

# Apache Iceberg Crash Course

Optimizing Apache Iceberg Tables



# Curriculum

July 11: What is a Data Lakehouse and What is a Table Format?

July 16: The Architecture of Apache Iceberg, Apache Hudi and Delta Lake

July 23: The Read and Write Process for Apache Iceberg Tables

Aug 13: Understanding Apache Iceberg's Partitioning Features

**Aug 27: Optimizing Apache Iceberg Tables**

Sep 3: Streaming with Apache Iceberg

Sep 17: The Role of Apache Iceberg Catalogs

Oct 1: Versioning with Apache Iceberg

Oct 15: Ingesting Data into Apache Iceberg with Apache Spark

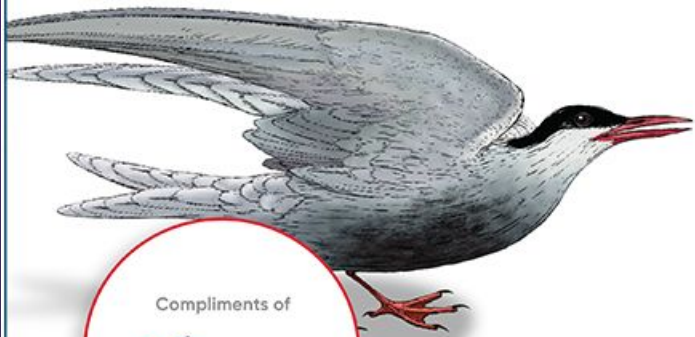
Oct 29: Ingesting Data into Apache Iceberg with Dremio

O'REILLY®

# Apache Iceberg

## The Definitive Guide

Data Lakehouse Functionality, Performance,  
and Scalability on the Data Lake



Compliments of



**dremio**

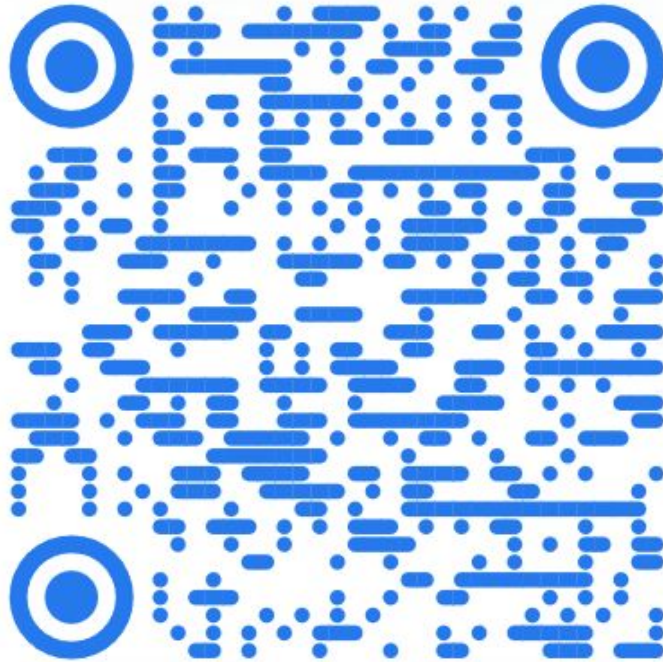
Tomer Shiran,  
Jason Hughes &  
Alex Merced

Forewords by Gerrit Kazmaier,  
Raghu Ramakrishnan & Rick Sears





[dremio.com/gnarly-data-waves](https://dremio.com/gnarly-data-waves)  
Youtube | Spotify | iTunes



**community.dremio.com**  
**Apache Iceberg Category**

# Table Optimization

# Compaction

**Problem:** Files build up after many writes resulting in more file operations than necessary slowing down queries, this is maximizing when handling “real-time” ingestion.

**Solution:** Rewrite data files so data in many files within the same partition are rewritten to fewer files.



## Compaction in Apache Spark (rewrite\_data\_files)

```
# Running Basic Compaction on a table with your
catalog
spark.sql("""
CALL nessie.system.rewrite_data_files(
  table => 'hr.employees'
);
""")
```



## Compaction in Apache Spark (rewrite\_data\_files)

```
  
# Running Compaction with Sorting  
spark.sql("""  
CALL nessie.system.rewrite_data_files(  
  table => 'hr.employees',  
  strategy => 'sort',  
  sort_order => 'id ASC NULLS LAST, name ASC NULLS FIRST'  
  );  
""")
```

## Compaction in Apache Spark (rewrite\_data\_files)

```
# Running Compaction with zorder Sorting
spark.sql("""
CALL nessie.system.rewrite_data_files(
  table => hr.employees',
  strategy => 'sort',
  sort_order => 'zorder(id,name)'
);
""")
```

## Compaction in Apache Spark (rewrite\_data\_files)

```
# Running Compaction with Custom Options
spark.sql("""
CALL nessie.system.rewrite_data_files(
  table => 'hr.employees',
  options => map('min-input-files','5', 'max-concurrent-file-group-rewrites','10')
);
""")
```

## Compaction in Apache Spark (rewrite\_data\_files)

```

● ● ●
# Running Compaction with Filters
spark.sql("""
CALL nessie.system.rewrite_data_files(
  table => 'hr.employees',
  where => 'department = "Engineering"'
);
""")
```

## Compaction in Dremio (OPTIMIZE)



```
-- Basic Compaction using OPTIMIZE in Dremio  
OPTIMIZE TABLE nessie.hr.employees;
```

## Compaction in Dremio (OPTIMIZE)

```
-- Optimization with specified minimum and maximum  
file sizes and target file size  
OPTIMIZE TABLE nessie.hr.employees  
  REWRITE DATA USING BIN_PACK (  
    MIN_FILE_SIZE_MB=100,  
    MAX_FILE_SIZE_MB=1000,  
    TARGET_FILE_SIZE_MB=512  
  );
```

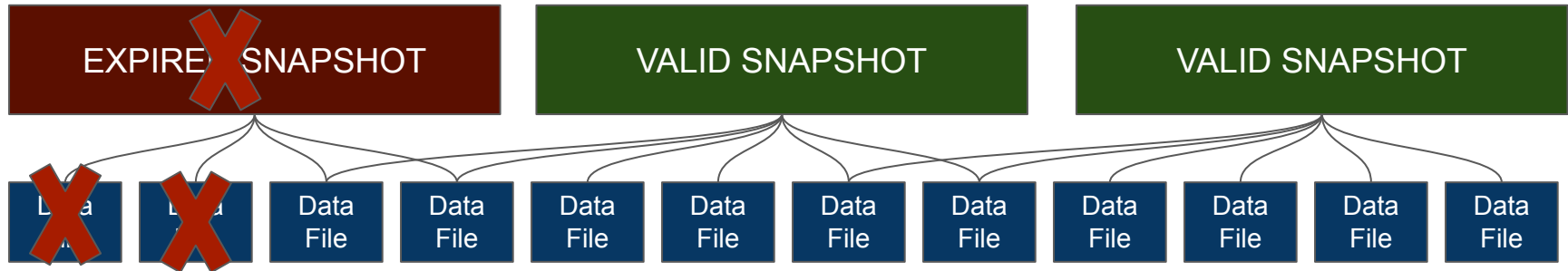
## Compaction in Dremio (OPTIMIZE)

```
-- Optimization for specific partitions
OPTIMIZE TABLE nessie.hr.employees
  REWRITE DATA USING BIN_PACK (
    MIN_FILE_SIZE_MB=100,
    MAX_FILE_SIZE_MB=1000)
  FOR PARTITIONS department = 'Engineering';
```

# Snapshot Expiration

**Problem:** As you collect more and more table snapshots the data files begin to take sizeable storage, also some data from historical snapshots may need to be deleted for regulatory snapshot.

**Solution:** Set data retention rules and expire snapshots according.





## Expiring Snapshots in Apache Spark (expire\_snapshots)

```
# Expiring Snapshots Older than Timestamp

spark.sql("""
CALL nessie.system.expire_snapshots(
  table => 'hr.employees',
  older_than => TIMESTAMP '2023-01-01 00:00:00.000'
);
""")
```

## Expiring Snapshots in Apache Spark (expire\_snapshots)

```

● ● ●

# Remove Snapshots Older Than a Specific Date and
# Retain the Last 100 Snapshots

spark.sql("""
CALL nessie.system.expire_snapshots(
  table => 'hr.employees',
  older_than => TIMESTAMP '2023-01-01 00:00:00.000',
  retain_last => 100
);
""")

```

## Expiring Snapshots in Apache Spark (expire\_snapshots)

```
# Remove Specific Snapshot IDs

spark.sql("""
CALL nessie.system.expire_snapshots(
  table => 'hr.employees',
  snapshot_ids => ARRAY(123, 456, 789)
);
""")
```

## Expiring Snapshots in Apache Spark (expire\_snapshots)

```
# Remove Snapshots Older Than 5 Days Ago  
(Default) and Retain the Last 5 Snapshots  
spark.sql("""  
CALL nessie.system.expire_snapshots(  
  table => 'hr.employees',  
  retain_last => 5  
);  
""")
```

## Expiring Snapshots in Apache Spark (expire\_snapshots)

```
# Remove Snapshots Older Than 5 Days Ago  
(Default) and Retain the Last 5 Snapshots  
spark.sql("""  
CALL nessie.system.expire_snapshots(  
  table => 'hr.employees',  
  retain_last => 5  
);  
""")
```

# The Nessie GC Cleaner



```
java -jar nessie-gc.jar sweep \  
  --jdbc \  
  --jdbc-url  
jdbc:postgresql://localhost:5432/nessie_gc \  
  --jdbc-user pguser \  
  --jdbc-password mysecretpassword \  
  --live-set-id $(cat live-set-id.txt) \  
  --max-file-modification "2023-12-31T23:59:59Z"
```

```
|
```

# The VACUUM Command (Dremio)



```
-- Remove snapshots older than a specific date, but retain at least 20 snapshots  
VACUUM TABLE catalog.hr.employees  
    EXPIRE SNAPSHOTS older_than '2023-06-01 00:00:00.000' retain_last 20;
```


## The VACUUM Command (Dremio)



```
-- Remove snapshots older than 5 days, but retain at least 100 snapshots  
VACUUM TABLE catalog.hr.employees EXPIRE SNAPSHOTS retain_last 100;
```



# The VACUUM Command (Dremio)



```
-- Remove snapshots older than 30 days, but retain at least 50 snapshots
VACUUM TABLE catalog.hr.employees
    EXPIRE SNAPSHOTS older_than CURRENT_DATE - INTERVAL '30' DAY retain_last 50;
```

## The VACUUM Command (Dremio Cloud)

A terminal window with a dark background and a light gray border. At the top left, there are three colored circles: red, yellow, and green. The text inside the terminal is as follows:

```
-- VACUUM a Nessie Catalog  
VACUUM CATALOG nessie;
```

# Metadata Tables

# Metadata Tables

**Problem:** Wanting to understand details about the table to know when you need to do maintenance and other optimizations.

**Solution:** Metadata tables allow you to use SQL to better understand the state of your tables

# Metadata Tables

Metadata Table	Purpose	Type of Data
<code>history</code>	Shows the history of the table	Timestamp the snapshot was made, snapshot ID, parent ID, whether the snapshot is part of the current history
<code>metadata_log_entries</code>	Shows metadata log entries for the table	Timestamp, file path, latest snapshot ID, latest schema ID, latest sequence number
<code>snapshots</code>	Shows the valid snapshots for the table	Timestamp the snapshot was committed, snapshot ID, parent ID, operation, manifest list, summary
<code>entries</code>	Shows the current manifest entries for both data and delete files	Status, snapshot ID, sequence number, file sequence number, data file, readable metrics
<code>files</code>	Shows the current files of the table	Content, file path, file format, spec ID, record count, file size, column sizes, value counts, null value counts, etc.
<code>manifests</code>	Shows the current file manifests of the table	Path, length, partition spec ID, added snapshot ID, added data files count, existing data files count, deleted files count

# Metadata Tables

<code>partitions</code>	Shows the current partitions of the table	Partition, spec ID, record count, file count, total data file size, delete record count, delete file count, etc.
<code>position_deletes</code>	Shows all positional delete files from the current snapshot of the table	File path, position, row, spec ID, delete file path
<code>all_data_files</code>	Shows all data files and their metadata across all snapshots	Content, file path, file format, partition, record count, file size, column sizes, value counts, null value counts, etc.
<code>all_delete_files</code>	Shows all delete files and their metadata across all snapshots	Content, file path, file format, spec ID, record count, file size, column sizes, value counts, null value counts, etc.
<code>all_entries</code>	Shows all manifest entries from all snapshots for both data and delete files	Status, snapshot ID, sequence number, file sequence number, data file, readable metrics
<code>all_manifests</code>	Shows all manifest files of the table	Path, length, partition spec ID, added snapshot ID, added data files count, existing data files count, deleted files count
<code>refs</code>	Shows the known snapshot references for the table	Reference name, type (branch or tag), snapshot ID, max reference age in ms, min snapshots to keep, max snapshot age in ms

# Use Cases for the Metadata Tables

- 1. What is the total storage footprint of my table?:** Join `files` and `all_data_files` to sum up the storage sizes of current and historical data files.
- 2. Which snapshots contributed the most data?:** Join `snapshots` and `all_data_files` to analyze the record count and file size contribution of each snapshot.
- 3. What changes have been made to the table's schema over time?:** Join `history` and `metadata_log_entries` to track schema changes and metadata file updates.
- 4. Which partitions are growing the fastest?:** Join `partitions` and `all_data_files` to compare partition sizes and record counts over time.
- 5. Which data files have been deleted recently?:** Join `history` and `all_delete_files` to identify recently deleted data files and their corresponding snapshots.
- 6. How are positional deletes affecting my table?:** Join `position_deletes` and `all_delete_files` to analyze the impact and frequency of positional deletes across snapshots.
- 7. What is the data distribution across different partitions?:** Join `partitions` and `all_entries` to examine the distribution of data files and delete files across partitions.
- 8. Which snapshots have the most metadata changes?:** Join `snapshots` and `metadata_log_entries` to find snapshots with significant metadata updates and schema changes.
- 9. What is the history of a specific snapshot?:** Join `history` and `snapshots` to trace the ancestry and operations leading to a specific snapshot.
- 10. What are the most recent changes to my table?:** Join `history` and `snapshots` to get a detailed view of the latest changes and operations performed on the table.

# Querying the Tables in Apache Spark

```
-- Query the history of the table
SELECT * FROM nessie.hr.employees.history;

-- Query the metadata log entries of the table
SELECT * FROM nessie.hr.employees.metadata_log_entries;

-- Query the valid snapshots for the table
SELECT * FROM nessie.hr.employees.snapshots;

-- Query the manifest entries for both data and delete files of the table
SELECT * FROM nessie.hr.employees.entries;

-- Query the current files of the table
SELECT * FROM nessie.hr.employees.files;

-- Query the current file manifests of the table
SELECT * FROM nessie.hr.employees.manifests;

-- Query the current partitions of the table
SELECT * FROM nessie.hr.employees.partitions;

-- Query all positional delete files from the current snapshot of the table
SELECT * FROM nessie.hr.employees.position_deletes;

-- Query all data files and their metadata across all snapshots of the table
SELECT * FROM nessie.hr.employees.all_data_files;

-- Query all delete files and their metadata across all snapshots of the table
SELECT * FROM nessie.hr.employees.all_delete_files;

-- Query all manifest entries from all snapshots for both data and delete files of the table
SELECT * FROM nessie.hr.employees.all_entries;

-- Query all manifest files of the table
SELECT * FROM nessie.hr.employees.all_manifests;

-- Query the known snapshot references for the table
SELECT * FROM nessie.hr.employees.refs;
```



# Querying the Tables in Apache Dremio

```
-- Querying the data files metadata of an Iceberg table
SELECT *
FROM TABLE(table_files('nessie.hr.employees'));

-- Querying the history of an Iceberg table
SELECT *
FROM TABLE(table_history('nessie.hr.employees'));

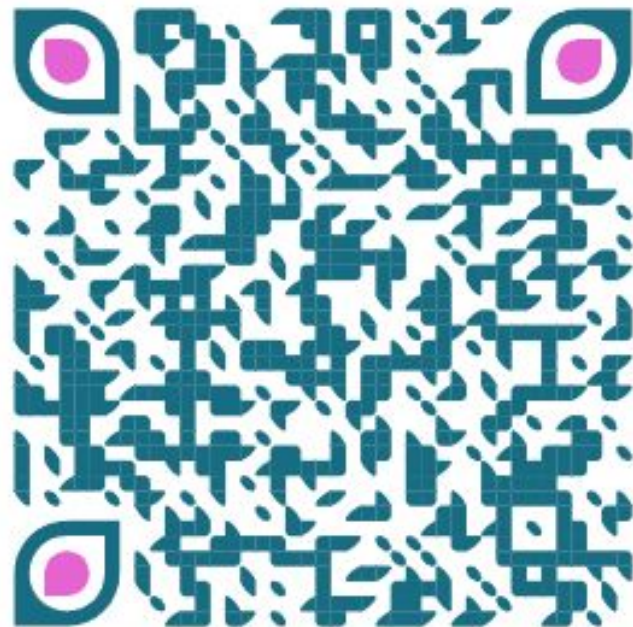
-- Querying the manifests metadata of an Iceberg table
SELECT *
FROM TABLE(table_manifests('nessie.hr.employees'));

-- Querying the partitions statistics of an Iceberg table
SELECT *
FROM TABLE(table_partitions('nessie.hr.employees'));

-- Querying the snapshots metadata of an Iceberg table
SELECT *
FROM TABLE(table_snapshot('nessie.hr.employees'));
```



A Iceberg/Dremio Lakehouse on  
your laptop exercise



Deploy Dremio Software or  
Dremio Cloud



Postgres -> Iceberg -> Dashboard

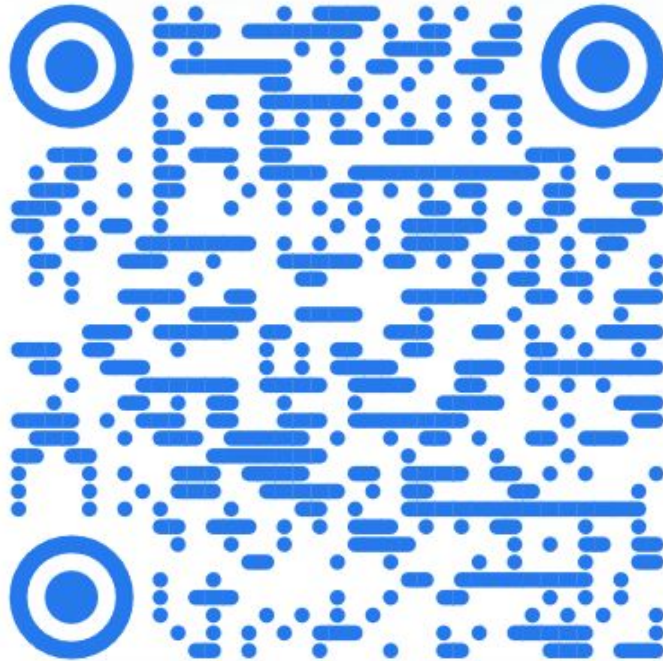


SQLServer -> Iceberg -> Dashboard



MongoDB -> Iceberg -> Dashboard

**[dremio.com/blog](https://dremio.com/blog)**



**[community.dremio.com](https://community.dremio.com)**  
**Apache Iceberg Category**