iceberg.apache.org • spark.apache.org

Iceberg Spark 3.3

CATALOGS

Configure a catalog, called "sandbox"

```
spark.sql.catalog.sandbox=\
    org.apache.iceberg.spark.SparkCatalog
spark.sql.catalog.sandbox.type=rest
spark.sql.catalog.sandbox.uri=\
    https://api.tabular.io/ws
spark.sql.catalog.sandbox.warehouse=sandbox
spark.sql.catalog.sandbox.credential=...
spark.sql.defaultCatalog=sandbox
```

Working with multiple catalogs in SQL

See the session's current catalog and database

```
SHOW CURRENT DATABASE
```

Sets the current catalog and database

```
USE sandbox.examples
```

List databases and tables

```
SHOW DATABASES
SHOW TABLES
```

QUERIES & METADATA TABLES

Simple select example

```
SELECT count(1) as row_count FROM logs
WHERE event_ts >= date_add(current_date(), -7))
  AND event_ts < current_date()
```

Note: Filters automatically select files using partitions and value stats

Metadata tables

```
-- lists all tags and branches
db.table.refs
```

```
-- all known revisions of the table
db.table.snapshots
```

```
-- history of the main branch
db.table.history
```

Note: Must be loaded using the full table name

Others:

`partitions`, `manifests`, `files`, `data_files`, `delete_files`

Inspecting tables

```
DESCRIBE db.table
```

Time travel

```
SELECT ... FROM table FOR VERSION AS OF ref_or_id
```

```
SELECT ... FROM table
  FOR TIMESTAMP AS OF '2022-04-14 11:00:00-07:00'
  -- Also works with metadata tables
```

Loading a table from a metadata file

```
df = spark.read.format("iceberg").load(
  "s3://bucket/path/to/metadata.json")
```

Metadata columns

`_file`	The file location containing the record
`_pos`	The position within _file of the record
`_partition`	The partition tuple used to store the record

FUNCTIONS

Call Iceberg transform functions

```
SELECT catalog.system.truncate(10, name) FROM table
```

```
SELECT catalog.system.bucket(16, id) FROM table
```

Inspect the Iceberg library version

```
SELECT catalog.system.iceberg_version() as version
```

tabular.io • docs.tabular.io

v 0.4.4

CREATE AND ALTER TABLE

Example syntax

```
CREATE TABLE IF NOT EXISTS logs (
    level string, event_ts timestamp, msg string, ...)
USING iceberg PARTITIONED BY (level, hours(event_ts))
```

Supported types

Primitive types:

`boolean`, `int`, `bigint`, `float`, `double`, `decimal(P,S)`, `date`, `timestamp`, `string`, `binary`

Note: Spark's timestamp type is Iceberg's timestamp with time zone type

Nested types:

`struct<name type, ...>`, `array<item_type>`, `map<key_type, value_type>`

Supported partition transforms

`column`	Partition by the unmodified column value
`years(event_ts)`	Year granularity *e.g. 2023*
`months(event_ts)`	Month granularity *e.g. 2023-03*
`days(event_ts)`	Day granularity *e.g. 2023-03-01*
`hours(event_ts)`	Hour granularity *e.g. 2023-03-01-10*
`truncate(width, col)`	Truncate strings or numbers in col
`bucket(width, col)`	Hash col values into width buckets

Schema evolution (ALTER TABLE table ...)

```
ADD COLUMN line_no int AFTER event_ts
```

```
-- widen type (int to bigint, float to double, etc.)
ALTER COLUMN line_no TYPE bigint
```

```
ALTER COLUMN line_no COMMENT 'Line number'
```

```
ALTER COLUMN line_no FIRST
```

```
ALTER COLUMN line_no AFTER event_ts
```

```
RENAME COLUMN msg TO message
```

```
DROP COLUMN line_no
```

Adding/updating nested types

```
ADD COLUMN location struct<lat float, long float>
```

```
ADD COLUMN location.altitude float
```

Note: UPDATE COLUMN can't modify struct types

Alter partition spec

```
ALTER TABLE ... ADD PARTITION FIELD days(event_ts) AS day
```

```
ALTER TABLE ... DROP PARTITION FIELD days(event_ts)
```

Setting distribution and sort order

Globally sort by event_ts

```
ALTER TABLE logs WRITE ORDERED BY event_ts
```

Distribute by partitions to writers and locally sort by event_ts

```
ALTER TABLE logs WRITE DISTRIBUTED BY PARTITION
    LOCALLY ORDERED BY event_ts
```

Remove write order

```
ALTER TABLE logs WRITE UNORDERED
```

Table properties

Set table properties

```
ALTER TABLE table SET TBLPROPERTIES ('prop'='val')
```

Format version: 1 or 2

`format-version`

Note: Must be 2 for merge-on-read

Age limit for snapshot retention

`history.expire.max-snapshot-age-ms`

Minimum number of snapshots to retain

`history.expire.min-snapshots-to-keep`

Mode by command: `copy-on-write` or `merge-on-read`

`write.(update|delete|merge).mode`

Isolation level by command: `snapshot` or `serializable`

`write.(update|delete|merge).isolation-level`

Target size, in bytes, for split combining for the table

`read.split.target-size`

WRITES

INSERT

```
INSERT INTO table SELECT id, data FROM ...
```

```
INSERT INTO table VALUES (1, 'a'), (2, 'b'), ...
```

MERGE

```
MERGE INTO target_table t
USING source_changes s ON t.id = s.id
WHEN MATCHED AND s.operation = 'delete' THEN DELETE
WHEN MATCHED THEN UPDATE SET t.count =
    t.count + s.count
WHEN NOT MATCHED THEN INSERT (t.id, t.count)
    VALUES (s.id, s.count)
```

For performance, add filters to the ON clause for the target table

```
ON t.id = s.id AND t.event_ts >=
    date_add(current_date(), -2)
```

Uses write.merge.mode

`copy-on-write` vs `merge-on-read`

Note: When in doubt, use copy-on-write for the best read performance

To enable merge-on-read:

```
ALTER TABLE target_table SET TBLPROPERTIES (
    'format-version'='2',
    'write.merge.mode'='merge-on-read')
```

UPDATE

```
UPDATE table SET count = count + 1 WHERE id = 5
```

DELETE FROM

```
DELETE FROM table WHERE id = 5
```

Dataframe writes

Create a writer

```
writer = df.writeTo(tableName)
```

Note: In catalogs with multiple formats, add `.using("iceberg")`

Create from dataframe

```
df.writeTo("catalog.db.table").partitionedBy($"col").create()
```

Append

```
df.writeTo("catalog.db.table").append()
```

Overwrite

```
df.writeTo("catalog.db.table").overwrite($"report_date" === d)
```

```
df.writeTo("catalog.db.table").overwritePartitions()
```

STORED PROCEDURES

Basic syntax

```
CALL system.procedure_name(named_arg => value, ...)
```

Compaction

Compact data and rewrite all delete files

```
CALL catalog.system.rewrite_data_files(
    table => 'table_name',
    where => 'col1 = "value"',
    options => map('min-input-files', '2',
                   'delete-file-threshold', '1'))
```

Compact and sort

```
CALL catalog.system.rewrite_data_files(
    table => 'table_name',
    strategy => 'sort',
    sort_order => 'col1, col2 desc')
```

Compact and sort using z-order

```
CALL catalog.system.rewrite_data_files(
    table => 'table_name',
    strategy => 'sort',
    sort_order => 'zorder(col1, col2)')
```

Optimize table metadata

```
CALL catalog.system.rewrite_manifests(table => 'table')
```

Roll back to previous snapshot or time

```
CALL catalog.system.rollback_to_snapshot(
    table => 'table_name',
    snapshot_id => 9180664844100633321)
```

```
CALL catalog.system.rollback_to_timestamp(
    table => 'table_name',
    timestamp => TIMESTAMP '2023-01-01 00:00:00.000')
```