trino ICEBERG

iceberg.apache.org • trino.io

CATALOGS

Configure a catalog, called "sandbox"

sandbox.properties

```
connector.name=iceberg
iceberg.catalog.type=REST
iceberg.rest-catalog.uri=\
    https://api.tabular.io/ws
iceberg.rest-catalog.security=OAUTH2
iceberg.rest-catalog.session=USER
iceberg.rest-catalog.oauth2.credential=\
    ${ENV:TABULAR_CREDENTIAL}
```

Working with multiple catalogs in SQL

See the session's current catalog and database

```
SELECT current_catalog, current_schema
```

Sets the current catalog and database

```
USE sandbox.examples
```

List schemas and tables

```
SHOW SCHEMAS
SHOW TABLES
```

QUERIES & METADATA TABLES

Simple select example

```
SELECT count(1) as row_count FROM default.events
  WHERE event_ts >= date_add('day', -7, current_date)
  AND event_ts < current_date
```

Note: Filters automatically select files using partitions and column stats

Metadata tables

```
-- lists all data files
db."table$files"
```

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

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

Others:

```
$data, $properties, $manifests, $partitions
```

Inspecting tables

```
DESCRIBE db.table
```

Time travel

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

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

CREATE AND ALTER TABLE

Example syntax

```
CREATE TABLE IF NOT EXISTS logs (
    level varchar, event_ts timestamp(6), msg varchar, ...)
COMMENT 'Table description'
WITH (
  partitioning = ARRAY['level', 'hour(event_ts)'],
  sorted_by = ARRAY['event_ts'],
  format = 'PARQUET')
```

Supported types

Primitive types:

`boolean`, `int`, `bigint`, `real`, `double`, `decimal(P,S)`, `date`, `timestamp(6)`, `timestamp(6) with time zone`, `time(6)`, `varchar`, `varbinary`

Nested types:

`row(name type, ...)`, `array(item_type)`, `map(key_type, value_type)`

Supported partition transforms

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

Schema evolution (ALTER TABLE table ...)

```
ADD COLUMN line_no int
```

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

Note: Do not use SET DATA TYPE for columns with nested types

```
RENAME COLUMN msg TO message
```

```
DROP COLUMN line_no
```

Adding nested types

```
ALTER TABLE logs ADD COLUMN location row(lat real, long real)
```

Changing descriptions

```
COMMENT ON TABLE table IS 'table description'
```

```
COMMENT ON COLUMN logs.line_no IS 'Line number'
```

Compute column statistics for better query performance

```
ANALYZE table_name
```

```
ANALYZE table_name WITH (columns = ARRAY['col_1', 'col_2'])
```

Setting table properties (ALTER TABLE table SET PROPERTIES ...)

Alter partition spec

```
ALTER TABLE ... SET PROPERTIES
  partitioning = ARRAY['level', 'day(event_ts)']
```

Set the table write order

```
ALTER TABLE ... SET PROPERTIES sorted_by = ARRAY[...]
```

Remove write order

```
ALTER TABLE ... SET PROPERTIES sorted_by = ARRAY[]
```

Set file format

```
ALTER TABLE table SET PROPERTIES format = 'PARQUET'
```

Table properties

`format` *File format:* `ORC`, `AVRO`, or `PARQUET`

Note: Parquet is recommended for the broadest compatibility

`partitioning` *An array of partition expressions:*

```
ARRAY['category', 'bucket(id, 256)']
```

`sorted_by` *Write order, array of sort expressions:*

```
ARRAY['category', 'id']
```

WRITES

INSERT

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

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

MERGE

```
MERGE INTO targe_table t
USING source_changes s
ON t.id = s.id
WHEN MATCHED AND s.operation = 'delete' THEN DELETE
WHEN MATCHED THEN UPDATE SET count = t.count + s.count
WHEN NOT MATCHED THEN INSERT (id, 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('day', -2, current_date)
```

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

UPDATE

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

DELETE FROM

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

STORED PROCEDURES

Table optimization

Compact data and rewrite all delete files

```
ALTER TABLE test_table EXECUTE optimize(
    file_size_threshold => '10MB')
```

```
ALTER TABLE table_name EXECUTE optimize
    WHERE level = 'ERROR'
```

Note: Files under file_size_threshold will be compacted (default: 100MB)

Basic procedure syntax

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

Roll back to previous snapshot or time

```
CALL example.system.rollback_to_snapshot(
    schema => 'testdb',
    table => 'table_name',
    snapshot_id => 8954597067493422955)
```

Starburst

starburst.io

Tabular

tabular.io • docs.tabular.io

v 1.0.0