Cheatography

TimeScale Cheat Sheet

by ecdk1234 via cheatography.com/60479/cs/22711/

Hypertables

Create Table- 2 steps

1. create a standard postgresql table :

```
CREATE TABLE conditions ( time TIMESTAMPTZ NOT NULL, location TEXT NOT NULL, temperature DOUBLE PRECISION NULL );
```

2. use create_hypertable command

```
SELECT create_hypertable('conditions', 'time');
```

Convert table **conditions** to hypertable with just time partitioning on column **time**

```
SELECT create_hypertable('conditions', 'time', chunk_time_interval => INTERVAL '1 day');
```

```
SELECT create_hypertable('conditions', 'time', 'location', 4, partitioning_func => 'location_hash');
```

```
SELECT create_hypertable('conditions', 'time', if_not_exists => TRUE);
```

Creat Index:CREATE INDEX ... WITH (timescaledb.transaction_per_chunk, ...);

```
CREATE INDEX ON conditions(time, device_id) WITH (timescaledb.transaction_per_chunk);
```

show_chunks()

```
SELECT show_chunks('conditions');
```

```
SELECT show_chunks(older_than => INTERVAL '3 months');
```

Compression

Utilities

Get information about hypertables

```
SELECT * FROM timescaledb_information.hypertable;
```

Get statistics about chunk compression

```
SELECT * FROM timescaledb_information.compressed_chunk_stats;
```

Get statistics about hypertable compression

```
SELECT * FROM timescaledb_information.compressed_hypertable_stats;
```

Get metadata and settings information for continuous aggregates

```
SELECT * FROM timescaledb_information.continuous_aggregates;
```

Get information about background jobs and statistics related to continuous aggregates

```
SELECT * FROM timescaledb_information.continuous_aggregate_stats;
```

Get relation size of the chunks of an hypertable

```
SELECT chunk_table, table_size, index_size, total_size FROM chunk_relation_size_pretty('conditions');
```

Get approximate row count for hypertable(s) based on catalog estimates

```
SELECT * FROM hypertable_approximate_row_count('conditions');
```

Get relation size of hypertable

```
SELECT table_size, index_size, toast_size, total_size FROM hypertable_relation_size_pretty('conditions');
```

Continuous Aggregate

Use Alter table

```
ALTER TABLE <table_name> SET (timescaledb.compress, timescaledb.compress_orderby = '<column_name> [ASC | DESC] [ NULLS { FIRST | LAST } ] [, ...]', timescaledb.compress_segmentby = '<column_name> [, ...]' );
```

```
ALTER TABLE metrics SET (timescaledb.compress, timescaledb.compress_orderby = 'time DESC', timescaledb.compress_segmentby = 'device_id');
```

add_compress_chunks_policy()

```
SELECT add_compress_chunks_policy('conditions', INTERVAL '60d');
```

Automation

add_reorder_policy()

```
SELECT add_reorder_policy('conditions', 'conditions_device_id_time_idx');
```

alter_job_schedule()

```
SELECT alter_job_schedule(job_id, schedule_interval => INTERVAL '5 minutes') FROM timescaledb_information.continuous_aggregate_stats WHERE view_name = 'conditions_agg'::regclass;
```

> reschedules the continuous aggregate job for the conditions_agg view so that it runs every five minutes.

Create View

```
CREATE VIEW continuous_aggregate_view( timec, minl, sumt, sumh ) WITH ( timescaledb.continuous, timescaledb.refresh_lag = '5 hours', timescaledb.refresh_interval = '1h' ) AS SELECT time_bucket('30day', timec), min(location), sum(temperature), sum(humidity) FROM conditions GROUP BY time_bucket('30day', timec);
```

Alter View

```
ALTER VIEW contagg_view SET (timescaledb.refresh_lag = '1h', timescaledb.max_interval_per_job = '1 week', timescaledb.refresh_interval = '30m');
```

Refresh View

```
REFRESH MATERIALIZED VIEW contagg_view;
```

Drop View

```
DROP VIEW contagg_view CASCADE;
```