TimescaleDB

Using TimescaleDB

TimescaleDB scales PostgreSQL for time-series data via automatic partitioning across time and space (partitioning key), yet retains the standard PostgreSQL interface.

In other words, TimescaleDB exposes what look like regular tables, but are actually only an abstraction (or a virtual view) of many individual tables comprising the actual data. This single-table view, which we call a hypertable, is comprised of many chunks, which are created by partitioning the hypertable's data in either one or two dimensions: by a time interval, and by an (optional) "partition key" such as device id, location, user id, etc. (Architecture discussion)

Virtually all user interactions with TimescaleDB are with hypertables. Creating tables and indexes, altering tables, inserting data, selecting data, etc., can (and should) all be executed on the hypertable.

From the perspective of both use and management, TimescaleDB just looks and feels like PostgreSQL, and can be managed and queried as such.

Before you start

PostgreSQL's out-of-the-box settings are typically too conservative for modern servers and TimescaleDB. You should make sure your `postgresql.conf` settings are tuned, either by using timescaledb-tune or doing it manually.

Creating a hypertable

```
-- Do not forget to create timescaledb extension
CREATE EXTENSION timescaledb;

-- We start by creating a regular SQL table
CREATE TABLE conditions (
  time        TIMESTAMPTZ       NOT NULL,
  location    TEXT              NOT NULL,
  temperature DOUBLE PRECISION  NULL,
  humidity    DOUBLE PRECISION  NULL
);

-- Then we convert it into a hypertable that is partitioned by time
SELECT create_hypertable('conditions', 'time');
```

- Quick start: Creating hypertables
- Reference examples

Inserting and querying data

Inserting data into the hypertable is done via normal SQL commands:

```
INSERT INTO conditions(time, location, temperature, humidity)
  VALUES (NOW(), 'office', 70.0, 50.0);

SELECT * FROM conditions ORDER BY time DESC LIMIT 100;

SELECT time_bucket('15 minutes', time) AS fifteen_min,
    location, COUNT(*),
    MAX(temperature) AS max_temp,
    MAX(humidity) AS max_hum
  FROM conditions
  WHERE time > NOW() - interval '3 hours'
  GROUP BY fifteen_min, location
  ORDER BY fifteen_min DESC, max_temp DESC;
```

In addition, TimescaleDB includes additional functions for time-series analysis that are not present in vanilla PostgreSQL. (For example, the `time_bucket` function above.)

- Quick start: Basic operations
- Reference examples
- TimescaleDB API

Documentation associated with Apache licensed "community edition" of TimescaleDB provided for convenience and built from Apache licensed source code available here.