# PostgreSQLで時系列データベースを実現、TimescaleDBの紹介

2022年11月17日
db tech showcase 2022

**SRA OSS LLC**
佐藤 友章
sato@sraoss.co.jp

# 自己紹介

- 佐藤 友章 (さとう ともあき)
- SRA OSSで技術部門の責任者を務める
- PostgreSQLのサポートやトレーニング、コンサルティングなど、今もなおエンジニアとして活動中
- PostgreSQLとの出会いは大学院時代、当時のバージョンは7.2
- 趣味はお酒と海外旅行

# 時系列データベースとは

What's Time-Series Database

# 時系列データベースとは

- 時系列データに最適化されたデータベース
- 時系列データとは
  - 時間情報 (タイムスタンプ) をもった一連の値
  - 例えば、IoTセンサーやDevOps監視、株価・為替のデータ、など
- 時系列データベースに求められるのは

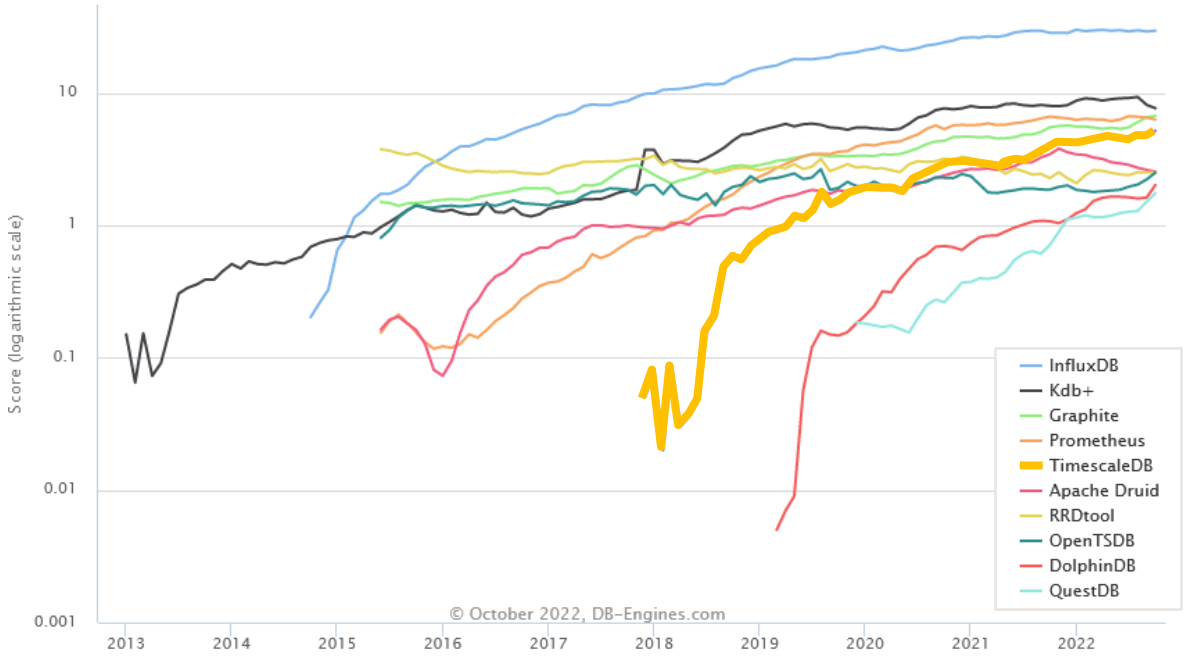
**高速な**
**同時書き込み**

**対話的な**
**集計問い合わせ**

**大量データの**
**保存**

# おもな時系列データベース製品

- 1番人気はInfluxDB、TimescaleDBは5番手 (全38製品中)




DB-Engines Ranking - Trend of Time Series DBMS Popularity
https://db-engines.com/en/ranking_trend/time+series+dbms

1. InfluxDB: 独自のストレージエンジンをもつ時系列データベース
2. Kdb+: インメモリ機能をもつ列ベースの時系列データベース
3. Graphite: シンプルな時系列データベースを含むグラフ化ツール
4. Prometheus: HTTPプル型の時系列データベースを含む監視ツール
5. TimescaleDB: PostgreSQLベースの時系列データベース

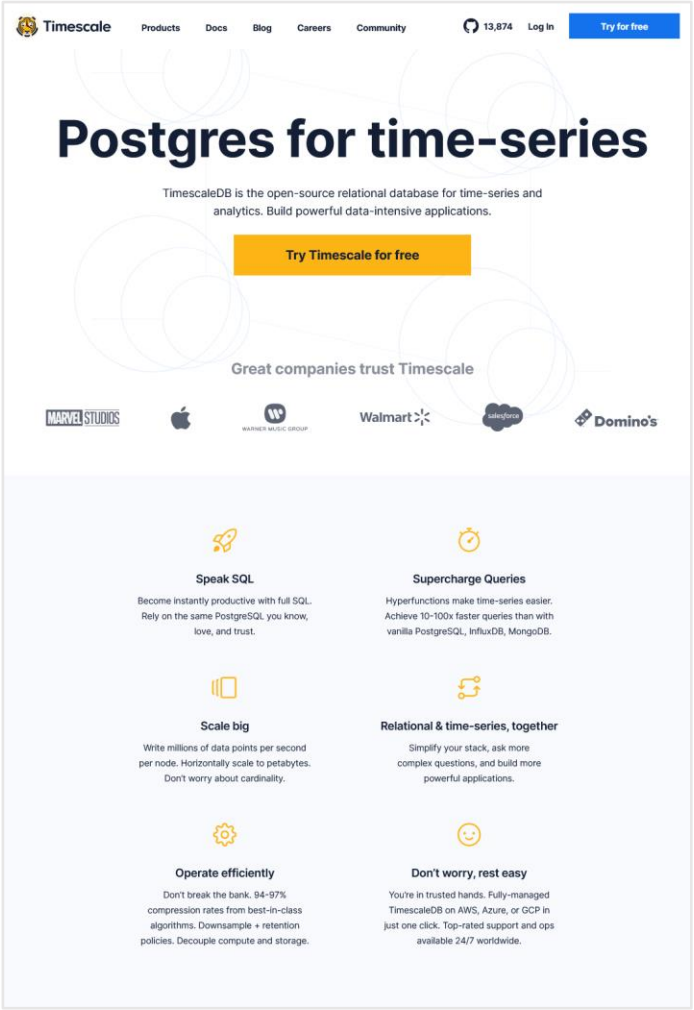
など

# TimescaleDBとは

What's TimescaleDB

# TimescaleDBとは

- 種別: 時系列データベース
- ウェブサイト: https://www.timescale.com
- 開発元: 米国Timescale社
- 初版リリース: 2017年
- ライセンス: Apache-2 (一部TSL)
- 実装言語: C言語
- コード量: 約14万行



# TimescaleDBのライセンス

自由に使える

**オープンソース機能**
(Apache License 2.0)

**コミュニティ機能**
(Timescale License)

DBasS提供除き
自由に使える

**コミュニティ機能**

- チャンクの並び替え・移動
- 分散ハイパーテーブル
- 圧縮
- 継続的集計
- データ保持
- ユーザ定義アクション
- 欠損値の補間

※ライセンスは各自で確認すること

# TimescaleDBの特長

## 使いやすさ

- SQLに完全対応
- PostgreSQL用アプリケーションがそのまま動作可能
- 時系列データ向け分析・管理機能あり

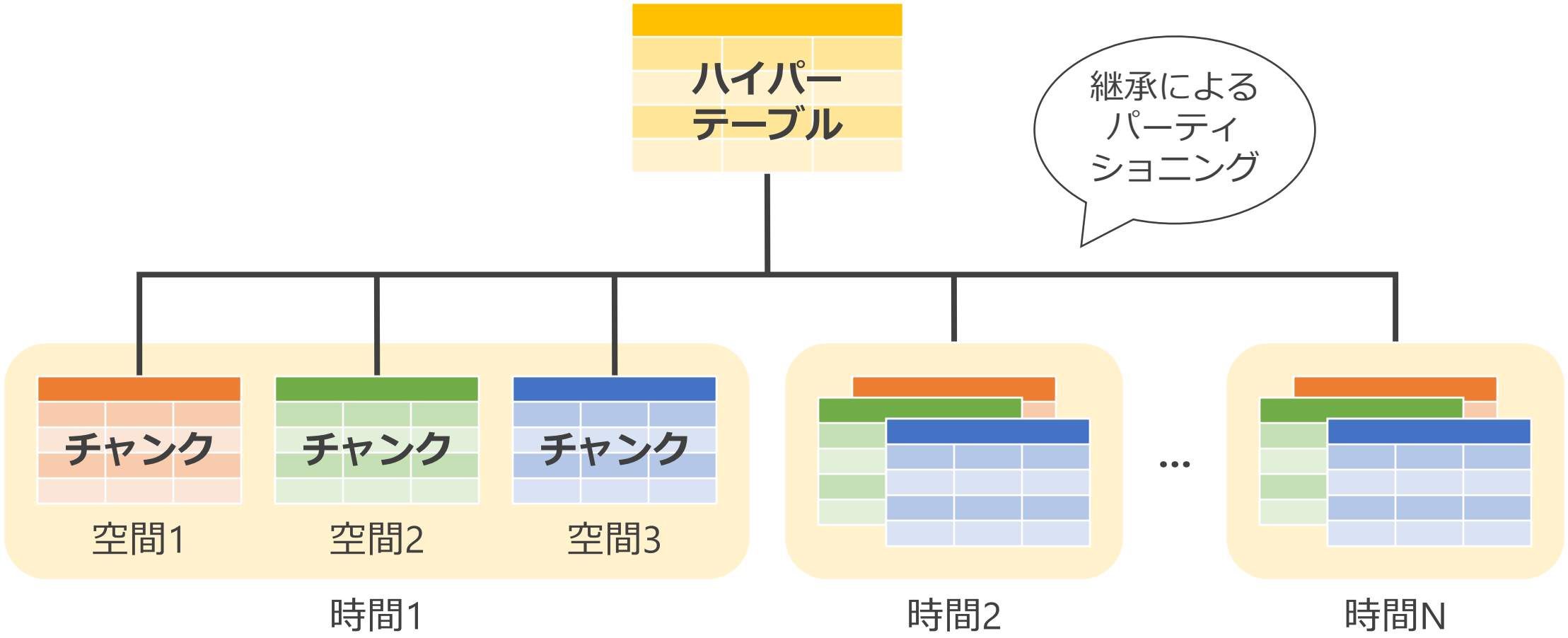
## スケーラビリティ

- データを適切なサイズに分割して格納
- データ量が増えても安定して高速に書き込み可能
- 分散処理に対応

## 信頼性

- 長い歴史をもつPostgreSQL上で動作
- バックアップやレプリケーションなど、PostgreSQLの機能・ツールが利用可能

# TimescaleDBのアーキテクチャ



# TimescaleDBのインストール

Install TimescaleDB

# TimescaleDBの利用方法

- Timescale Cloudを使う
  - Timescale社提供のマネージドサービス
  - AWSとAzure、GCPに対応
  - 0.032米ドル/時間～
- 自前で構築して使う
  - Debian/Ubuntu、RHEL/CentOS、Windows、macOS向けパッケージ、Dockerイメージ、AMIが利用可能
  - ソースコードからもビルド可能

**TimescaleDB/PostgreSQL バージョン対応**

| TimescaleDB バージョン | PostgreSQL バージョン |
|---|---|
| 2.5以降 | 12、13、14 |
| 2.4 | 12、13 |
| 2.1～2.3 | 11、12、13 |
| 2.0 | 11、12 |
| 1.7 | 9.6、10、11、12 |

※最新情報はPostgreSQLアップグレードを参照

# TimescaleDBのインストール RHEL/CentOSの場合 (1/4)

- PostgreSQLインストール済み、サーバ稼働中の状態が前提
- TimescaleDBのYumリポジトリを登録

```
$ sudo tee /etc/yum.repos.d/timescale_timescaledb.repo <<EOF
[timescale_timescaledb]
name=timescale_timescaledb
baseurl=https://packagecloud.io/timescale/timescaledb/el/¥$releasever/¥$basearch
repo_gpgcheck=1
gpgcheck=0
enabled=1
gpgkey=https://packagecloud.io/timescale/timescaledb/gpgkey
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
metadata_expire=300
EOF
```

# TimescaleDBのインストール RHEL/CentOSの場合 (2/4)

- TimescaleDBのRPMパッケージをインストール
  - コミュニティ機能を含まないパッケージtimescaledb-ossもあり

```
$ sudo yum -y install timescaledb-2-postgresql-14
(省略)
Installed:
  timescaledb-2-loader-postgresql-14-2.8.1-0.el8.x86_64   ← ライブラリローダ
  timescaledb-2-postgresql-14-2.8.1-0.el8.x86_64          ← 本体
  timescaledb-tools-0.14.1-0.el8.x86_64                   ← クライアントツール

Complete!
```

# TimescaleDBのインストール RHEL/CentOSの場合 (3/4)

- TimescaleDB向けにPostgreSQLの設定を行う

```
$ sudo timescaledb-tune --pg-config=/usr/pgsql-14/bin/pg_config
Using postgresql.conf at this path:
/var/lib/pgsql/14/data/postgresql.conf

Is this correct? [(y)es/(n)o]: y
Writing backup to:
/tmp/timescaledb_tune.backup202210171441

shared_preload_libraries needs to be updated
Current:
#shared_preload_libraries = ''
Recommended:
shared_preload_libraries = 'timescaledb'
Is this okay? [(y)es/(n)o]: y
success: shared_preload_libraries will be updated

Tune memory/parallelism/WAL and other settings? [(y)es/(n)o]: y
(省略)
Saving changes to: /var/lib/pgsql/14/data/postgresql.conf
```

- timescaledb-tuneはCPU数やメモリ量をもとに対話的に設定を行うツール
- 最低限必要な設定はカンマ区切りでshared_preload_librariesにtimescaledbを追加するだけ

# TimescaleDBのインストール RHEL/CentOSの場合 (4/4)

- PostgreSQLのサービスを再起動し、TimescaleDBの拡張モジュールをインストール

```
$ sudo systemctl restart postgresql-14
$ sudo su - postgres
$ createdb test
$ psql -c "CREATE EXTENSION timescaledb" test
WARNING:
WELCOME TO
 _____ _                               _     _____________
|_   _(_)                             | |    |  _  \ ___ \
  | |  _ _ __ ___   ___  ___  ___ __ _| | ___| | | | |_/ /
  | | | |  _ ` _ \ / _ \/ __|/ __/ _` | |/ _ \ | | | ___ \
  | | | | | | | | |  __/\__ \ (_| (_| | |  __/ |/ /| |_/ /
  |_| |_|_| |_| |_|\___||___/\___\__,_|_|\___|___/ \____/
               Running version 2.8.1
(省略)
CREATE EXTENSION
```

# TimescaleDBの基本的な使い方

Getting Started with TimescaleDB

# ハイパーテーブルへの変換

- 通常のテーブルを作成し、create_hypertable関数を実行
  - チャンクは7日間ごとに分割
  - 分割の間隔はchunk_time_interval引数で変更可能

```
=# CREATE TABLE conditions (
     time timestamp NOT NULL,
     temperature double precision
);
CREATE TABLE
=# SELECT create_hypertable('conditions', 'time');
    create_hypertable
-------------------------
 (1,public,conditions,t)
(1 行)
```

# ハイパーテーブルの制限事項

- ハイパーテーブルを参照する外部キー制約は作成できない
- 分割条件の時間列にNULLは格納できない
- 一意性インデックスに分割条件の全列を含める必要あり
- データをチャンク間で移動できない

# ハイパーテーブルのデータ操作

- 通常のテーブルと基本的に変わらない
  - データ登録時に格納先のチャンクがなければ作成される

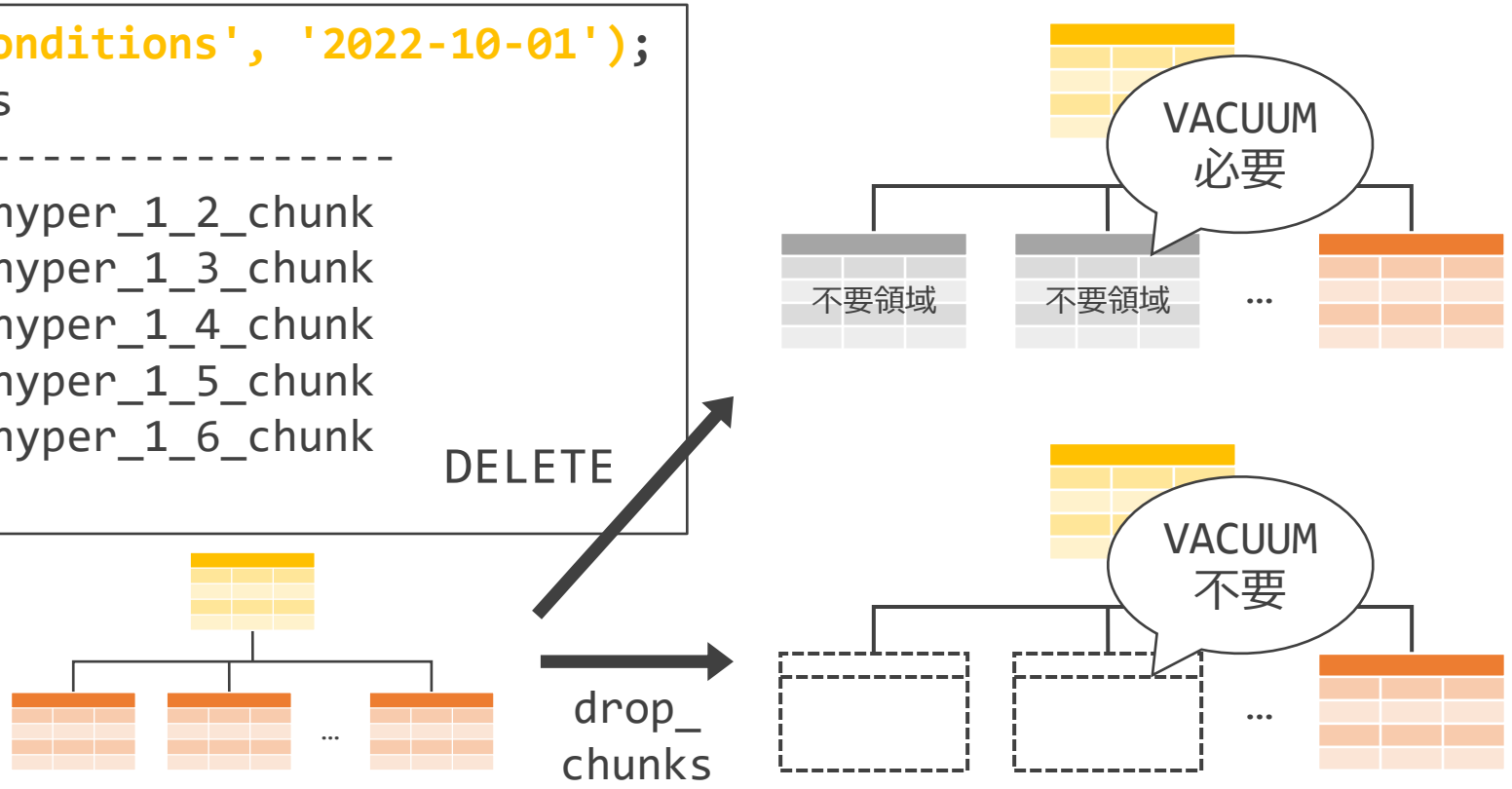
```
=# INSERT INTO conditions VALUES (current_timestamp, 21.7);
INSERT 0 1
=# SELECT * FROM conditions;
         time          | temperature
-----------------------+-------------
 2022-10-17 10:00:00 |        21.7
(1 行)

=# UPDATE conditions SET temperature = temperature + 0.1
   WHERE time = '2022-10-17 10:00:00';
UPDATE 1
```

# チャンクの削除

- チャンクごと削除すれば、不要領域が発生せずに済む

```
=# SELECT drop_chunks('conditions', '2022-10-01');
              drop_chunks
----------------------------------------
 _timescaledb_internal._hyper_1_2_chunk
 _timescaledb_internal._hyper_1_3_chunk
 _timescaledb_internal._hyper_1_4_chunk
 _timescaledb_internal._hyper_1_5_chunk
 _timescaledb_internal._hyper_1_6_chunk
(5 行)
```



# TimescaleDBによるデータ分析

Data Analysis with TimescaleDB

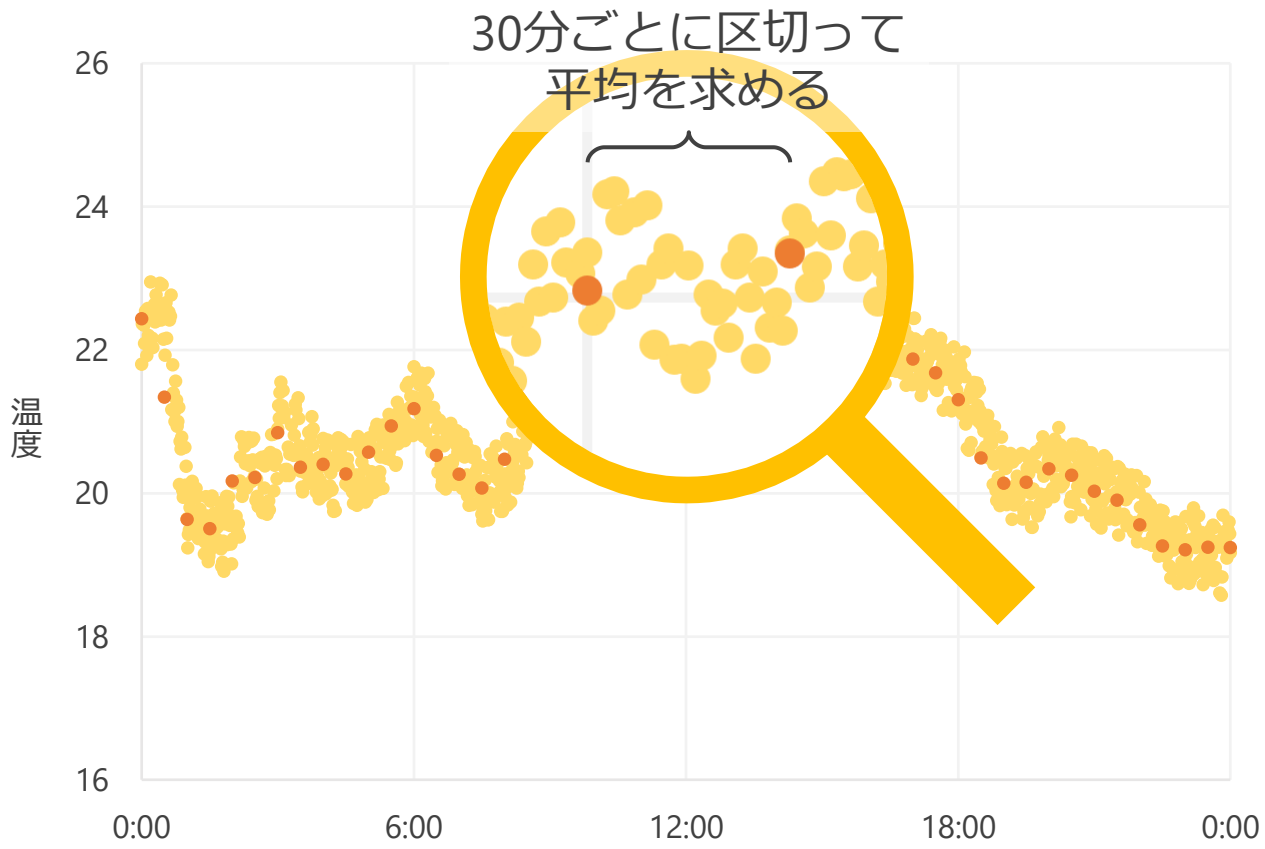
# time_bucket関数

- 時間を任意の間隔で丸める

```
=# SELECT time, temperature
   FROM condition
   ORDER BY time;
         time          |    temperature
-----------------------+--------------------
 2022-10-17 00:00:00 |  21.80233243974021
 2022-10-17 00:01:00 |  22.36631915597042
           :           |          :
 2022-10-17 00:29:00 | 22.152689798625406
(省略)

=# SELECT time_bucket('30min', time) AS time_30min,
     avg(temperature)
   FROM conditions
   GROUP BY time_30min
   ORDER BY time_30min;
     time_30min       |        avg
----------------------+--------------------
 2022-10-17 00:00:00 | 22.438319603197666
(省略)
```

平均

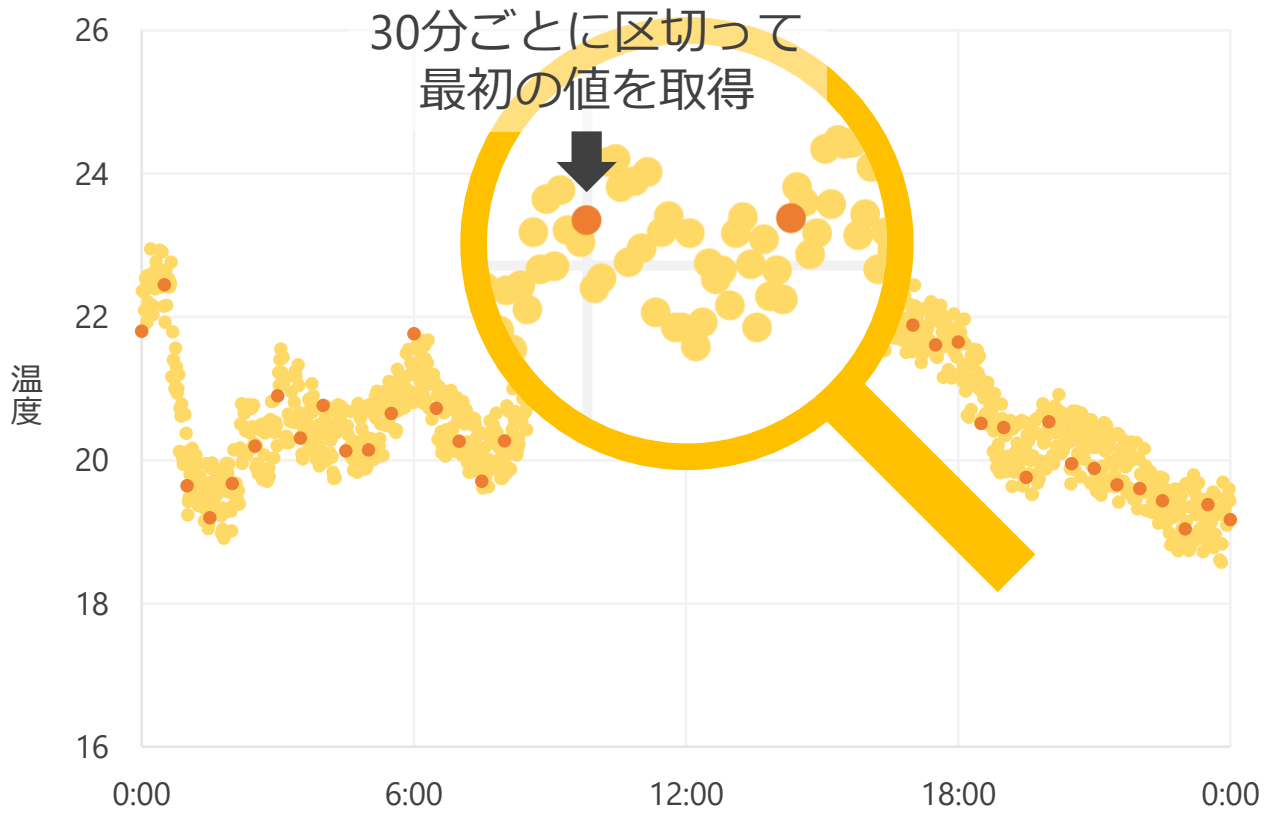


# first/last関数

- 別の列をキーに並び替え、最初・最後の値を取得

```
=# SELECT time, temperature
   FROM condition
   ORDER BY time;
         time          |    temperature
-----------------------+--------------------
 2022-10-17 00:00:00 |  21.80233243974021   ← 最初
 2022-10-17 00:01:00 |  22.36631915597042
           :           |          :
 2022-10-17 00:29:00 | 22.152689798625406
(省略)

=# SELECT time_bucket('30min', time) AS time_30min,
     first(temperature, time)
   FROM conditions
   GROUP BY time_30min
   ORDER BY time_30min;
     time_30min       |       first
----------------------+--------------------
 2022-10-17 00:00:00 |  21.80233243974021
(省略)
```

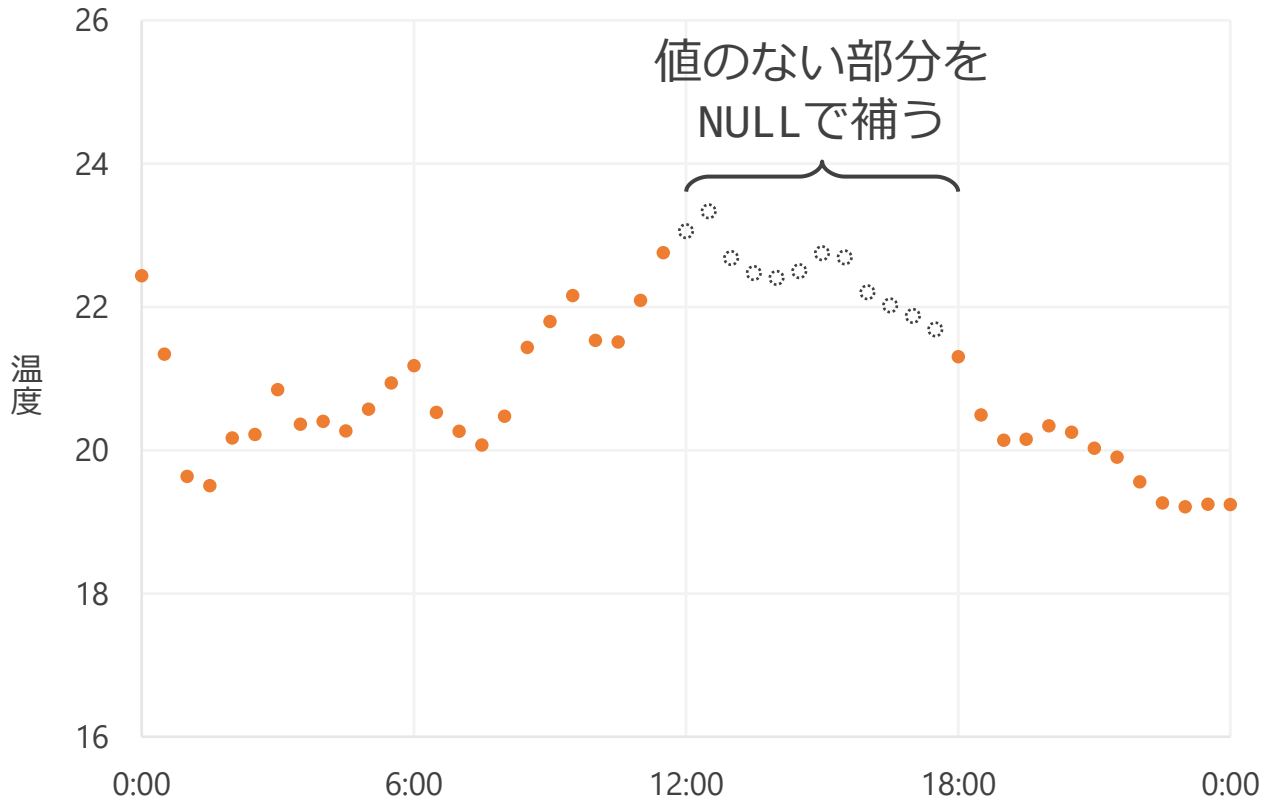


# time_bucket_gapfill関数

コミュニティ機能

- 欠損値を補って時間を任意の間隔で丸める

```
=# DELETE FROM conditions
   WHERE time >= '2022-10-17 12:00:00'
     AND time < '2022-10-17 18:00:00';
DELETE 360
=# SELECT time_bucket_gapfill('30min', time)
       AS time_30min,
     avg(temperature)
   FROM conditions
   WHERE time BETWEEN '2022-10-17' AND '2022-10-18'
   GROUP BY time_30min
   ORDER BY time_30min;
     time_30min       |        avg
----------------------+--------------------
 2022-10-17 00:00:00 | 22.438319603197666
          :          |          :
 2022-10-17 11:30:00 | 22.760572405493722
 2022-10-17 12:00:00 |                          } NULL補完
          :          |          :               }
 2022-10-17 17:30:00 |                          }
 2022-10-17 18:00:00 | 21.307451267267307
(省略)
```



# locf関数

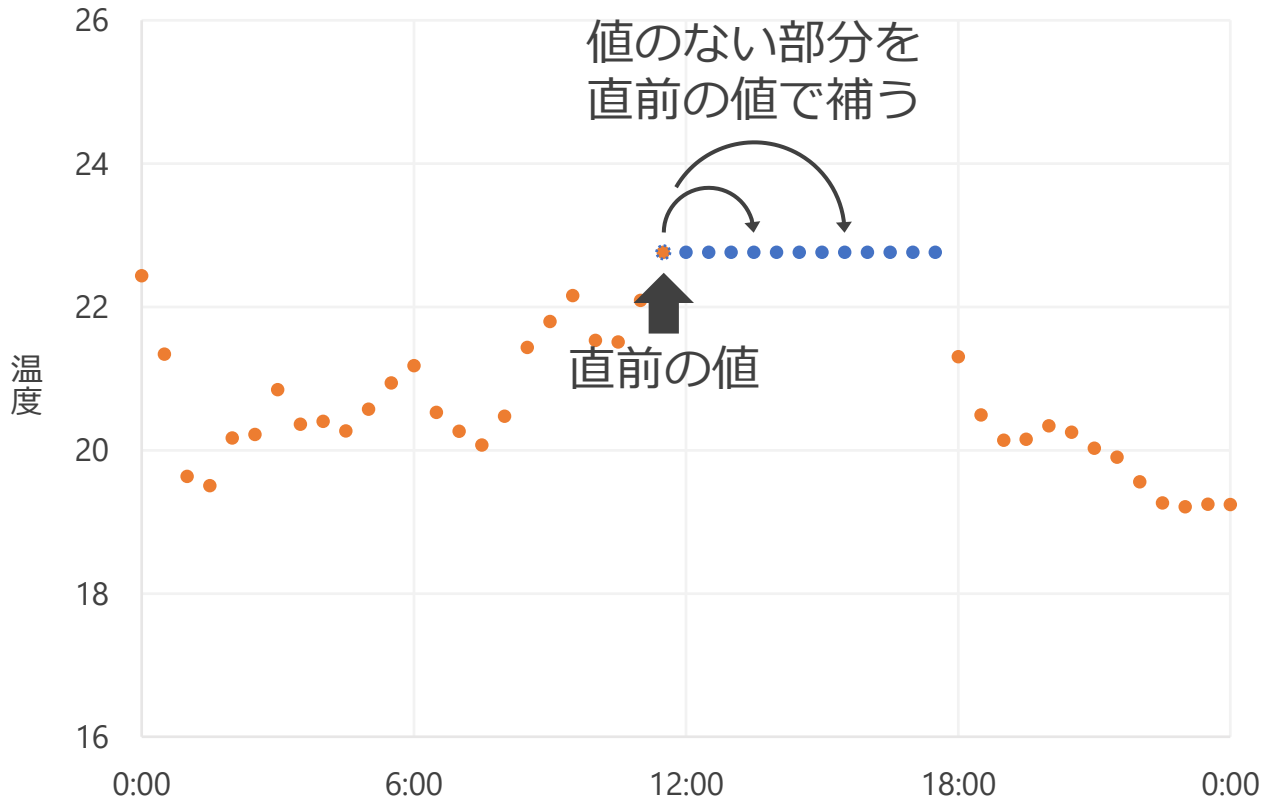
コミュニティ機能

- 直前の値で欠損値を補う (last observation carried forward)

```
=# SELECT time_bucket_gapfill('30min', time)
       AS time_30min,
     locf(avg(temperature))
   FROM conditions
   WHERE time BETWEEN '2022-10-17' AND '2022-10-18'
   GROUP BY time_30min
   ORDER BY time_30min;
     time_30min      |        avg
---------------------+--------------------
 2022-10-17 00:00:00 | 22.438319603197666
          :          |         :
 2022-10-17 11:30:00 | 22.760572405493722
 2022-10-17 12:00:00 | 22.760572405493722
 2022-10-17 12:30:00 | 22.760572405493722
 2022-10-17 13:00:00 | 22.760572405493722
          :          |         :
 2022-10-17 17:30:00 | 22.760572405493722
 2022-10-17 18:00:00 | 21.307451267267307
(省略)
```

直前

LOCF補完

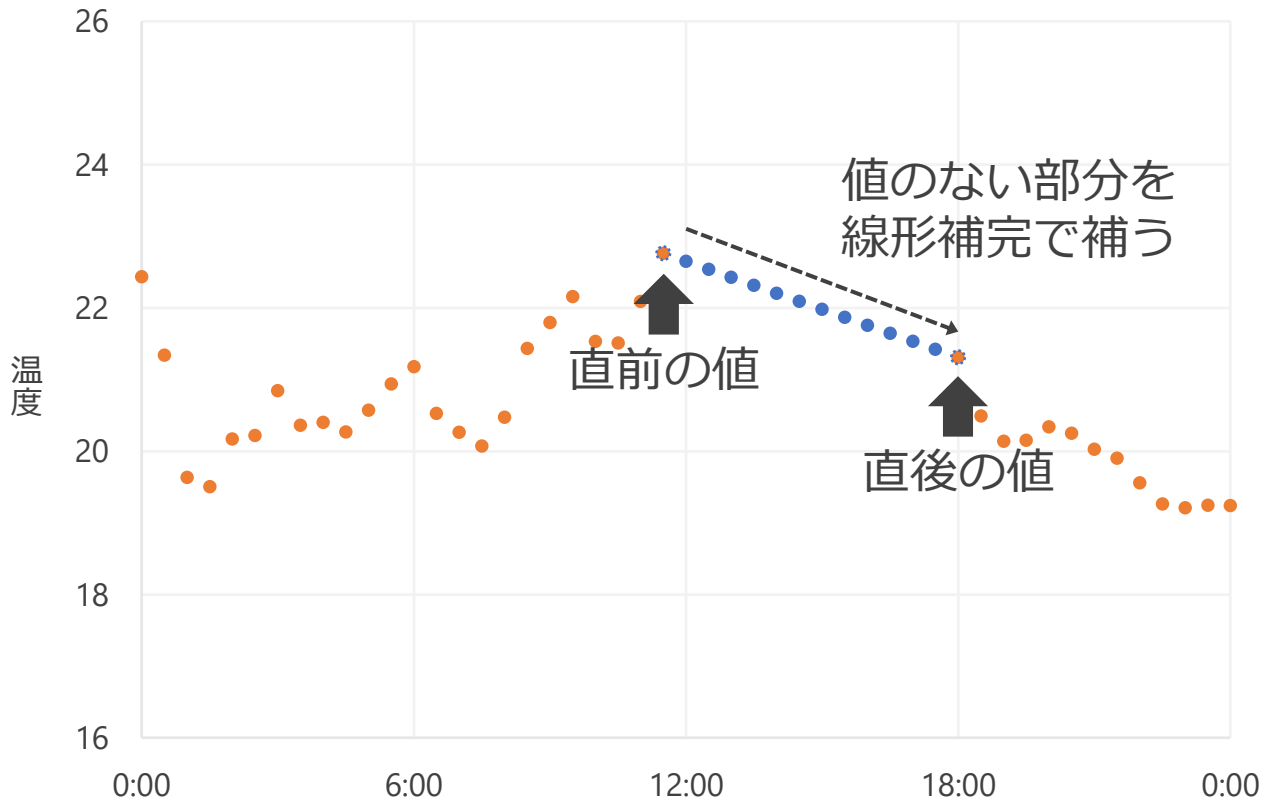


# interpolate関数

コミュニティ機能

- 直前・直後の値の線形補完で欠損値を補う

```
=# SELECT time_bucket_gapfill('30min', time)
       AS time_30min,
     interpolate(avg(temperature))
   FROM conditions
   WHERE time BETWEEN '2022-10-17' AND '2022-10-18'
   GROUP BY time_30min
   ORDER BY time_30min;
     time_30min      |        avg
---------------------+--------------------
 2022-10-17 00:00:00 | 22.438319603197666
          :          |         :
 2022-10-17 11:30:00 | 22.760572405493722   ← 直前
 2022-10-17 12:00:00 | 22.648793856399383
 2022-10-17 12:30:00 | 22.537015307305044   線形補完
 2022-10-17 13:00:00 | 22.425236758210705
          :          |         :
 2022-10-17 17:30:00 | 21.419229816361646
 2022-10-17 18:00:00 | 21.307451267267307   ← 直後
(省略)
```



# SRA OSS そのほかのデータ分析関数

- histogram関数
  - ヒストグラムを取得
- approximate_row_count関数
  - テーブルのおおよその行数を取得

```
=# SELECT time_bucket('30min', time) AS time_30min,
     histogram(temperature, 18, 23, 5)
   FROM conditions
   WHERE time BETWEEN '2022-10-17'
     AND '2022-10-18'
   GROUP BY time_30min
   ORDER BY time_30min;
     time_30min      |     histogram
---------------------+-------------------
 2022-10-17 00:00:00 | {0,0,0,0,2,28,0}
 2022-10-17 00:30:00 | {0,0,1,10,10,9,0}
 2022-10-17 01:00:00 | {0,0,26,4,0,0,0}
 2022-10-17 01:30:00 | {0,2,28,0,0,0,0}
 2022-10-17 02:00:00 | {0,0,10,20,0,0,0}
 2022-10-17 02:30:00 | {0,0,10,20,0,0,0}
(省略)
```

```
=# ¥timing
タイミングは on です。
=# SELECT count(*) FROM conditions;
  count
---------
 1455480
(1 行)

時間: 246.751 ミリ秒
=# SELECT approximate_row_count('conditions');
 approximate_row_count
-----------------------
                1455840
(1 行)

時間: 4.144 ミリ秒
```

# そのほかの機能

コミュニティ機能

- チャンクの並び替え・移動
- 分散ハイパーテーブル
- 圧縮
- 継続的集計 (continuous aggregates)
- データ保持 (data retention)
- ユーザ定義アクション

# まとめ

- 時系列データベースは時間情報をもった一連のデータを扱うのに最適化されたデータベース
- TimescaleDBはPostgreSQLベースの時系列データベース
  - DBasS提供をしなければ、すべての機能が自由に使える
  - NoSQLとリレーショナルデータベースの特長をあわせもつ
  - データは時間で分割されたチャンクに格納され、ハイパーテーブルを通じてアクセスされる

# 参考URL

- SRA OSS Tech Blog (日本語)


  - TimescaleDBの紹介
    - https://www.sraoss.co.jp/tech-blog/pgsql/timescaledb-intro/
  - TimescaleDBによるデータ分析
    - https://www.sraoss.co.jp/tech-blog/pgsql/timescaledb-analysis/
  - TimescaleDBによるZabbixの性能改善
    - https://www.sraoss.co.jp/tech-blog/zabbix/zabbix-timescaledb/
- Timescale Blog (英語)
  - TimescaleDB vs. PostgreSQL for time-series
    - https://blog.timescale.com/blog/timescaledb-vs-6a696248104e/
  - TimescaleDB vs. InfluxDB
    - https://blog.timescale.com/blog/timescaledb-vs-influxdb-for-time-series-data-timescale-influx-sql-nosql-36489299877/

SRA OSS

www.sraoss.co.jp sales@sraoss.co.jp 03-5979-2701