DATS-6202:TimescaleDB

Arshiful Islam Shadman
Amy Yang
Xi Qian
Ruijin Jia

Introduction - TimescaleDB

- An extension of PostgreSql
- An open-source database built for analyzing time-series data with the power and convenience of SQL — on premise, at the edge or in the cloud.

Characteristics of Time Series Data

Time-series data is data that collectively represents how a system, process, or behavior changes over time.

- **Time-centric**: Data records always have a timestamp.
- **Append-only**: Data is almost solely append-only (INSERTs).
- **Recent**: New data is typically about recent time intervals, and we more rarely make updates or backfill missing data about old intervals.

Time Series Data is Everywhere

- **Financial trading systems**: Classic securities, newer cryptocurrencies, payments, transaction events.
- **Eventing applications**: User/customer interaction data like clickstreams, pageviews, logins, signups, etc.
- **Business intelligence**: Tracking key metrics and the overall health of the business.
- ……

Why we need TimescaleDB?

These databases don't work for time series.

Relational Databases

MySQL

PostgreSQL

NoSQL Databases

InfluxDB

mongoDB

cassandra

Comparison - Relational DBs

- **Much Higher Ingest Rates:**TimescaleDB achieves a much higher and more stable ingest rate than PostgreSQL for time-series data.
- **Superior or Similar Query Performance:** for almost every query, TimescaleDB achieves either similar or superior (or vastly superior) performance to vanilla PostgreSQL
- **Time-oriented Features:** Time-oriented Analytics,Time-oriented Data Management

Comparison - NoSQL DB

- **Operational simplicity:** With TimescaleDB, you only need to manage one database for your relational and time-series data. Otherwise, users often need to silo data into two databases: a "normal" relational one, and a second time-series one.
- **JOINs** can be performed across relational and time-series data
- **Faster Query performance** is faster for a varied set of queries. More complex queries are often slow or full table scans on NoSQL databases, while some databases can't even support many natural queries.
- ………….

Timescale DB - Advantages

1. Easy to use

- **Full SQL interface** for all SQL natively supported by PostgreSQL (including secondary indexes, non-time based aggregates, sub-queries, JOINs, window functions).
- **Connects** to any client or tool that speaks PostgreSQL, no changes needed.
- **Time-oriented** features, API functions, and optimizations.
- Robust support for **Data retention policies**.

TimescaleDB - Advantages

2. Scalable

- **Transparent time/space partitioning** for both scaling up (single node) and scaling out (forthcoming).
- **High data write rates** (including batched commits, in-memory indexes, transactional support, support for data backfill).
- **Right-sized chunks** (two-dimensional data partitions) on single nodes to ensure fast ingest even at large data sizes.
- **Parallelized operations** across chunks and servers.

TimescaleDB - Advantages

3. Reliable

- **Engineered up** from PostgreSQL, packaged as an extension.
- **Proven foundations** benefiting from 20+ years of PostgreSQL research (including streaming replication, backups).
- **Flexible management options** (compatible with existing PostgreSQL ecosystem and tooling).

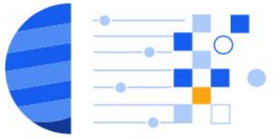
Key customers

Bloomberg

COMCAST

Convenience of SQL

Get started right away using the query language your developers and business analysts already know.



Rich time-series analytics

Built-in tools to perform common time-series data analysis, including buckets, gap filling, aggregations, and more.

Massive scale and performance

Write millions of data points per second. Store 100s of billions of rows and 10s of terabytes of data.

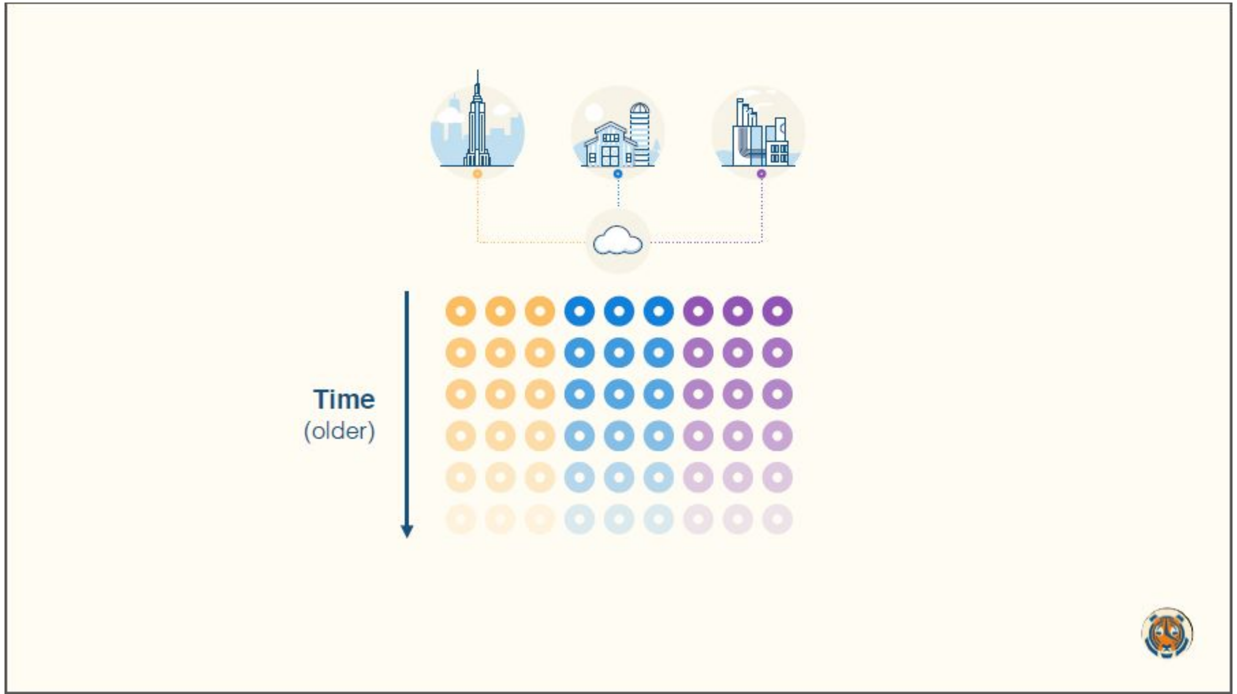
Product: TimescaleDB Open Source, TimescaleDB Community, TimescaleDB Enterprise

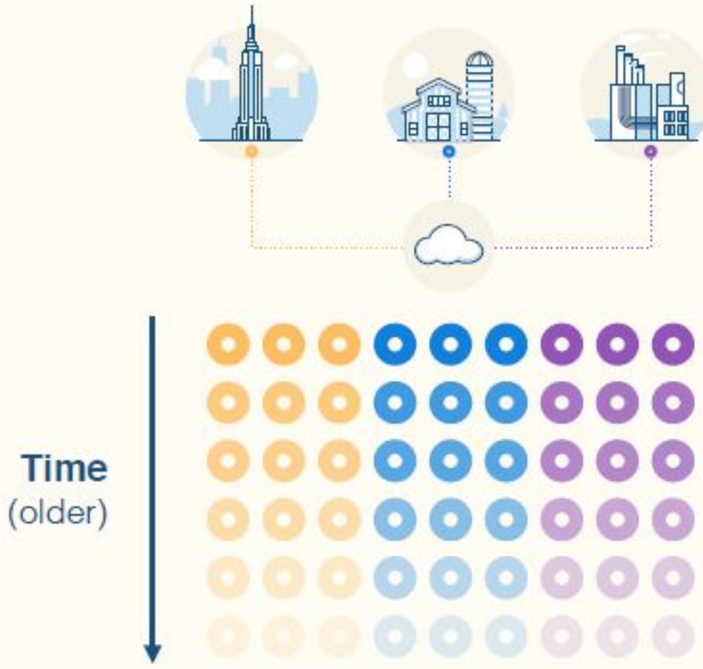
Time-series analytics	Open-source	Community	Enterprise
Hypertable abstraction layer	✓	✓	✓
Automatic chunking / partitioning	✓	✓	✓
Optimized time-based constraint exclusion	✓	✓	✓
Joins across time-series and relational tables	✓	✓	✓
Built-in flexible time bucketing / windowing	✓	✓	✓
Advanced analytical functions (gap filling, LOCF, interpolation)		✓	✓
Automatic continuous aggregations		✓	✓

Pricing Models

1. Choose Cloud (AWS/Microsoft Azure/Google Cloud)
2. Choose Region
3. Choose Plan (Dev/Basic/Pro)
4. Configuration: Storage/RAM/vCPUs

How does TimescaleDB optimizes for time-series data



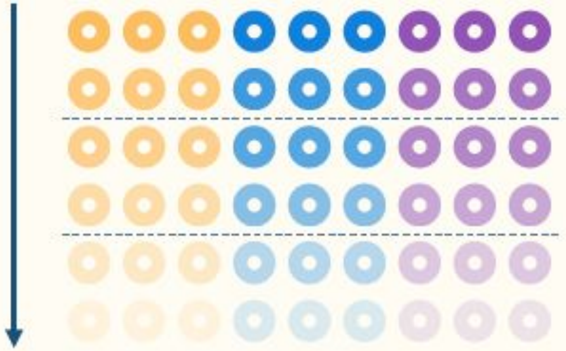

Time
(older)

Time-space partitioning

(for both scaling up & out)

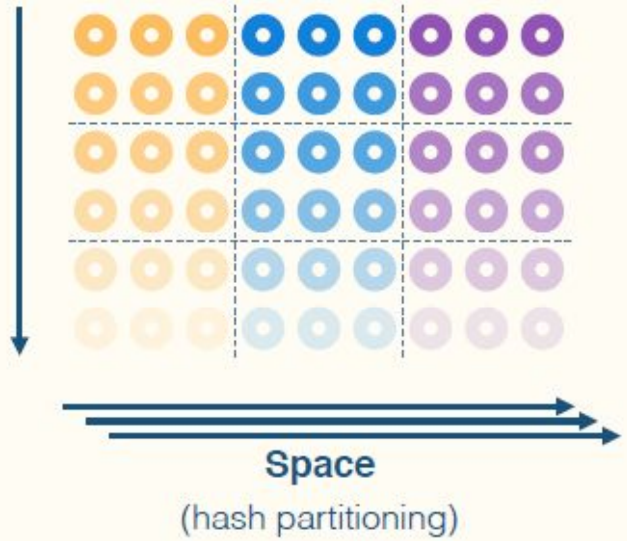
Time
(older)

Intervals
1) manually specified
2) automatically adjusted



Time-space partitioning

(for both scaling up & out)



Time

(older)

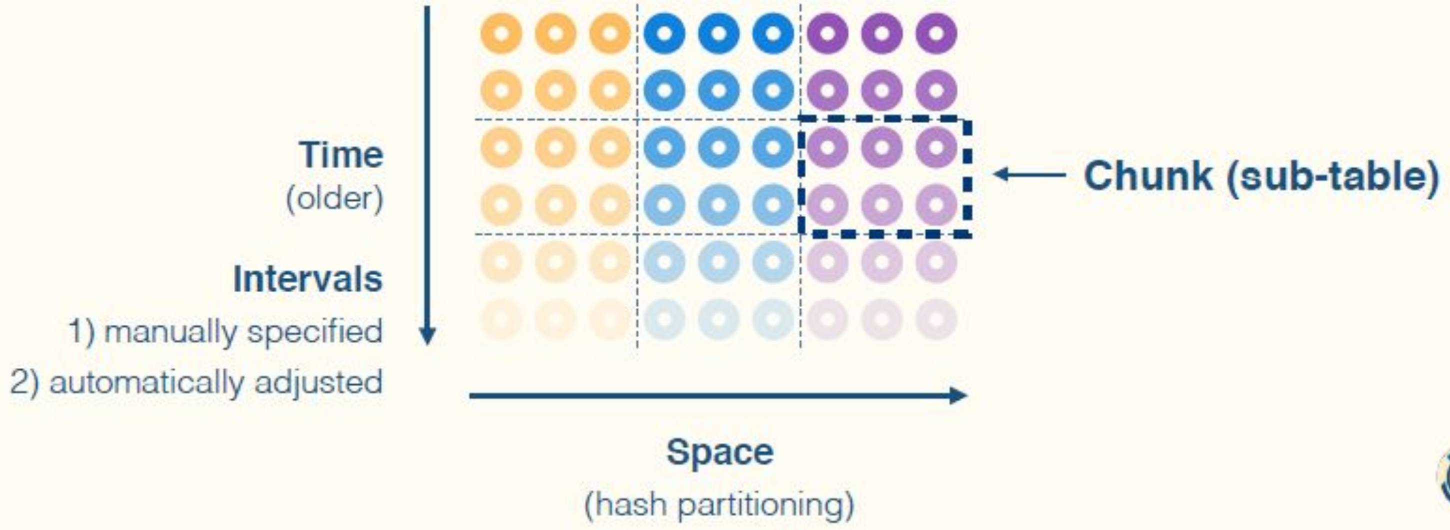
Intervals

1) manually specified

2) automatically adjusted

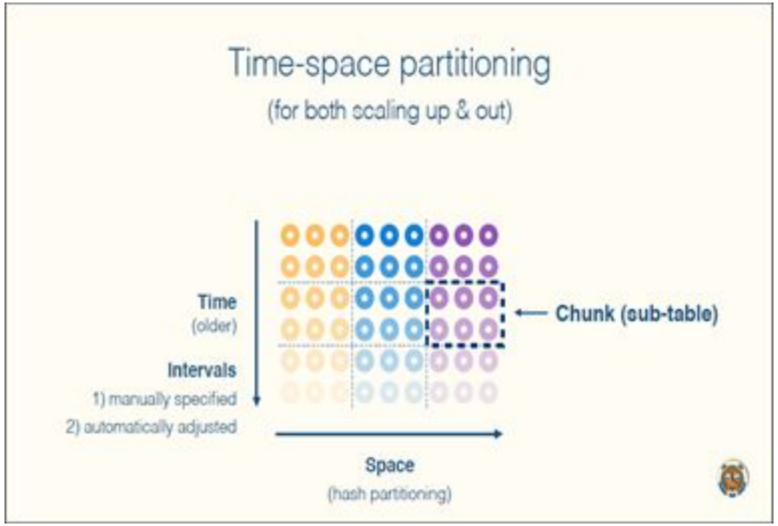
Time-space partitioning

(for both scaling up & out)

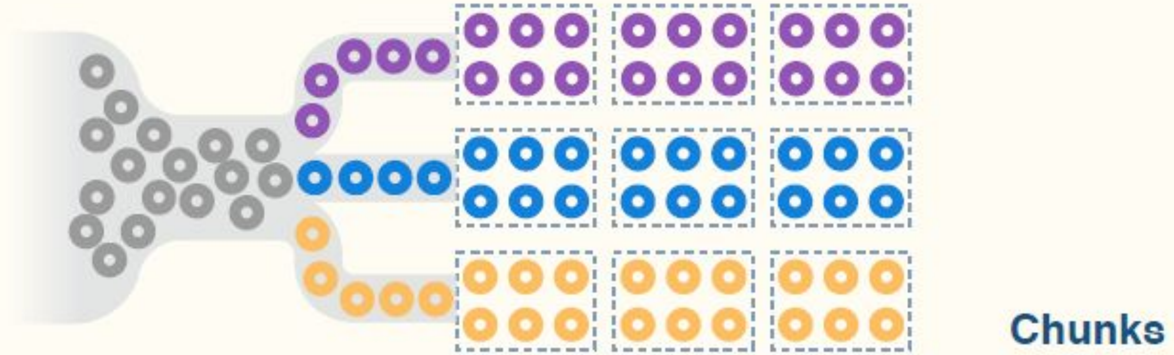


Chunks

- Chunking offers a wide set of capabilities: scaling-up (on the same node) and scaling-out (across multiple nodes), elasticity, partitioning flexibility, data retention policies, data tiering, and data reordering.
- Each of the chunk (and their B-trees) is stored as a separate table in the PostgreSQL database.
- Chunks are right sized to fit into memory.
- By avoiding overly large chunks, we can avoid expensive "vacuuming" operations when removing data.

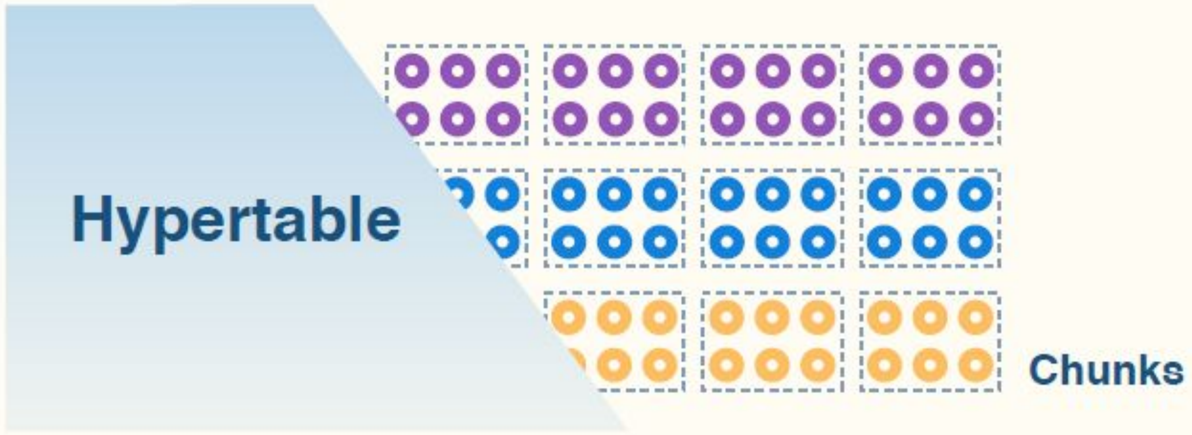


Automatic Space-time Partitioning



But treat it like a single table

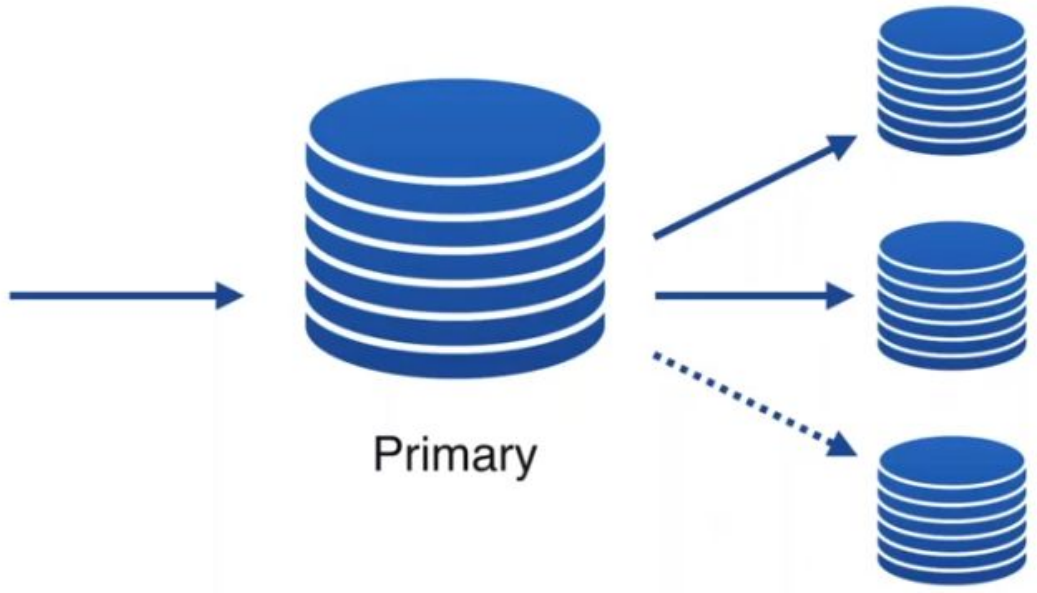
- Indexes
- Triggers
- Constraints
- Foreign keys
- UPSERTs
- Table mgmt



Consistency

- **Tunable consistency**: has several options with varying consistency/performance tradeoffs.
- **Streaming replication**: TimescaleDB writes all data to a single primary node which then replicates that data to any connected replicas through streaming replication.
- **Asynchronous and synchronous replication**:
  - The most common streaming replication use case is asynchronous replication with one or more replicas.
  - In cases where you need stronger consistency on the replicas or where your query load is heavy enough to cause significant lag between the primary and replica nodes in asyncronous mode, you may want to consider one of the synchronous replication configurations.
- Third party solutions: Kubernetes, Patroni

Clustering via PostgreSQL replication



Synchronous replicas
for high availability
and scaling queries

Asynchronous replicas
for data science

Consistency

- **Tunable consistency**: has several options with varying consistency/performance tradeoffs.
- **Streaming replication**: TimescaleDB writes all data to a single primary node which then replicates that data to any connected replicas through streaming replication.
- **Asynchronous and synchronous replication**:
  - The most common streaming replication use case is asynchronous replication with one or more replicas.
  - In cases where you need stronger consistency on the replicas or where your query load is heavy enough to cause significant lag between the primary and replica nodes in asyncronous mode, you may want to consider one of the synchronous replication configurations.
- Third party solutions: Kubernetes, Patroni

DEMO

1. Installation
2. Setting up
3. Configuration
4. Creating Hyper Tables
5. Simple Queries
6. Migrating Data

https://docs.google.com/document/d/1cfKNe0uM5aSlzbjyfmhpAvND7wSB5chLt4lbp_POShQ/edit?usp=sharing

THANK YOU