A.A. 2021 - 2022

M. CECCARELLI - E. D'ANGELIS - P. ZARRI

LIBRARY SEAT RESERVATION

TIMESCALEDB

LIBRARY SEAT RESERVATION

INTRODUZIONE

TimescaleDB è un database relazionale open-source **Full SQL** per i dati time-series, che promette di "*scalare*" con performance che prima erano raggiungibili solo dai database NoSQL.

TimescaleDB è sostanzialmente una estensione di **PostgreSQL** in grado di offrire un insieme di operazioni relative ai *time-series data*, cioè ad un set di dati annotato *temporalmente*. In quanto derivato da PostgreSQL, TimescaleDB risulta supportato da molti linguaggi di programmazione, tra cui **Java** e **Python**.

HYPERTABLES

Le **hypertables** sono ciò che consente a TimescaleDB di lavorare in modo così efficace con i dati delle serie temporali, rendendo l'**archiviazione** e l'**interrogazione di dati** operazioni estremamente veloci su scala peta-byte.

TimescaleDB infatti partiziona automaticamente i dati delle *time-series* in **blocchi** o **sotto-tabelle**, sia in base al tempo che allo spazio.

Una **hypertable** è quindi un livello di astrazione che consente di interrogare e accedere ai dati da **tutti i blocchi**, come se fossero in un'unica tabella: i comandi inviati all'hypertable vengono infatti applicati a tutti i blocchi che appartengono a quella hypertable.

La creazione di una hypertable è un processo suddiviso in due fasi:

1. Inizialmente si utilizza un'istruzione SQL **`CREATE TABLE`** per creare una tabella relazionale;

2. Successivamente è possibile usare l'istruzione **`SELECT`** con la funzione `create_hypertable()` per convertire la tabella in una hypertable (vedi esempio).

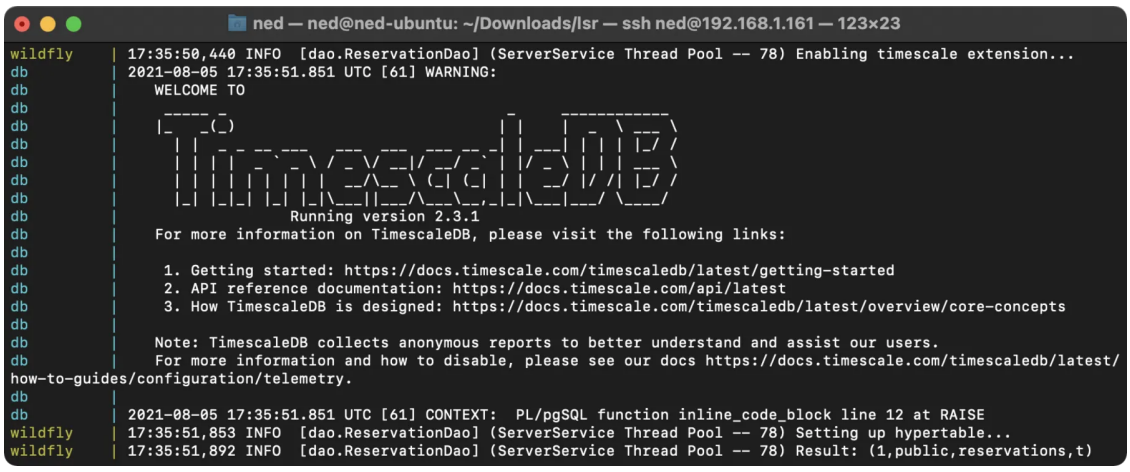
```
SELECT create_hypertable('reservations', 'datetime')
```

Esempio di creazione di hypertable per la tabella `reservations` e colonna `datetime`

L'istruzione **`SELECT`** richiede sia il nome della tabella da convertire che il nome della colonna *temporale* presente in quella tabella.

I vantaggi delle hypertables sono:

- miglioramento nelle performance nella **scrittura dei dati**, poiché questi vengono inseriti solo nel blocco corrente, mentre i dati negli altri blocchi rimangono intatti. Utilizzando invece una singola tabella relazionale, non essendoci alcuna partizione, dopo ogni inserimento aumenterà di dimensioni fino a ridurne le prestazioni;
- migliori prestazioni nei **tempi di risposta** per le query, poiché con queste partizioni vengono interrogati solo blocchi specifici, grazie all'**indicizzazione automatica** per tempo o spazio;
- il modo in cui i dati vengono partizionati su disco: nella tabella è infatti presente un **indice** che viene gestito automaticamente dalla dipendenza temporale dei dati, in modo da consentire un uso più mirato della memoria. In altri sistemi di gestione di database relazionali, il recupero avviene invece direttamente del livello fisico (memoria o disco), che non sempre si traduce in un uso efficace della memoria e delle risorse.



Esempio log di attivazione di Timescale e creazione di una hypertable

VANTAGGI DI TIMESCALEDB

Uno dei maggiori vantaggi di TimescaleDB è il fatto che **supporta il linguaggio SQL** in modo nativo, riducendo molto la curva di apprendimento, pur offrendo una **serie di funzioni** che non si trovano nei database relazionali tradizionali.

Queste funzioni hanno lo scopo di fornire due vantaggi chiave:

- maggiore facilità d'uso per l'analisi delle *time-series;*
- prestazioni migliorate.

Una di queste funzioni critiche per le serie temporali è `time_bucket()`.

`time_bucket()` viene utilizzato per **aggregare periodi di tempo** di dimensioni arbitrarie (es. 5 minuti, 1 giorno). Essenzialmente, `time_bucket()` è una versione più potente della funzione standard `date_trunc()` di PostgreSQL, in quanto consente anche la scelta di intervalli di tempo arbitrari.

Oltre a consentire query di serie temporali più flessibili, `time_bucket()` consente anche di scriverle in modo più semplice a livello sintattico:

```sql
SELECT time_bucket('5 minutes', time) AS five_min, avg(cpu)
  FROM metrics
  GROUP BY five_min
  ORDER BY five_min DESC;
```

Esempio di query SQL con `time_bucket` per estrarre la media temporale (colonna `time`) della colonna `cpu` ad intervalli di 5 minuti.

`time_bucket()` risulta molto utile per creare dashboard o visualizzazioni di dati *time-series*, e in generale per trasformare le osservazioni "*raw*" in aggregati di livello superiore con intervalli di tempo prefissati.