

Gestione di Serie Temporali con TimescaleDB: un'analisi sperimentale

Relatore:
Chiar.mo Prof.
Marco Di Felice

Presentata da:
Andrea Serrano

Obiettivi della tesi

- ➔ **Panoramica generale sulle serie temporali**
- ➔ **Architettura e funzionalità di TimescaleDB**
- ➔ **Analisi di performance di TimescaleDB e confronto con PostgreSQL**
- ➔ **Progettazione e realizzazione di una sperimentazione pratica per applicazione IoT**
- ➔ **Sviluppi futuri**

Obiettivi della tesi

- ➔ **Panoramica generale sulle serie temporali**
- ➔ **Architettura e funzionalità di TimescaleDB**
- ➔ **Analisi di performance di TimescaleDB e confronto con PostgreSQL**
- ➔ **Progettazione e realizzazione di una sperimentazione pratica per applicazione IoT**
- ➔ **Sviluppi futuri**

1)

Le serie temporali (o *time series*) sono collezioni di dati ordinati cronologicamente, dove ogni osservazione è associata a un preciso istante di tempo



2)

**Sensori IoT
(IoT)**



**Monitoraggio
ambientale**



**Servizi
finanziari**



**Time series
analysis**



3) Caratteristiche e criticità

01

Alto volume
di dati dovuto
all'alta frequenza
di registrazione

02

Rapida crescita
del volume dei
dati

03

Scalabilità,
efficienza delle
interrogazioni e
persistenza nel
lungo periodo

04

Le **basi di dati**
tradizionali non
sempre sono
ottimali

4) Time Series Database



5)

TimescaleDB:
nato di recente,
approccio relazionale



Timescale

Estensione open source di
PostgreSQL

Progettato per l'elaborazione e
l'analisi di dati temporali su
larga scala

Soluzioni architetturali dedicate

Obiettivi della tesi

- ➔ **Panoramica generale sulle serie temporali**
- ➔ **Architettura e funzionalità di TimescaleDB**
- ➔ **Analisi di performance di TimescaleDB e confronto con PostgreSQL**
- ➔ **Progettazione e realizzazione di una sperimentazione pratica per applicazione IoT**
- ➔ **Sviluppi futuri**

6)

Innovazioni architettonali chiave

01

Utilizzo di
hypertable e
chunking
automatico

02

Hypercore: motore
ibrido riga/colonna

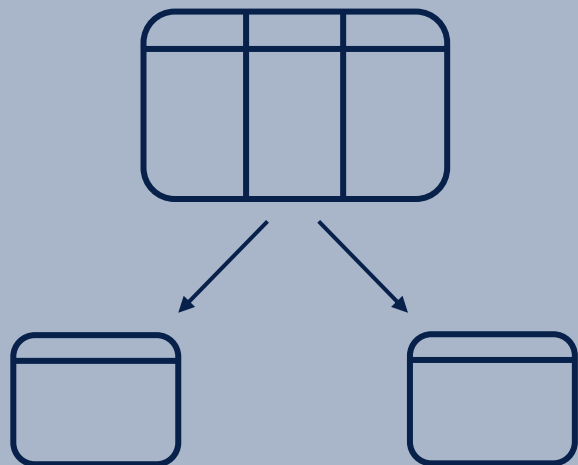
03

Tecniche di
compressione
dei dati

7) HYPERTABLE & CHUNK

Una hypertable è una tabella composta da più **chunk** (sottotabelle)

Permette la gestione scalabile dei dati temporali



Ogni **chunk** contiene dati relativi ad uno **specifico intervallo temporale**



Le query analizzano **solo i chunk rilevanti**
→ **miglior performance**



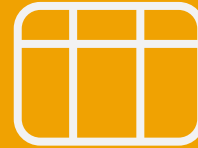
Vantaggi:

- 📌 Indici piccoli e rapidi
- 🔄 Migliore località della cache
- 🔧 Meno manutenzione
- 🖌️ Meno table bloat

8) HYPERCORE: UNA SOLUZIONE IBRIDA

E' una **strategia di storage ibrido** che adatta dinamicamente il formato di memorizzazione in base al ciclo di vita del dato




Column-store: i chunk più vecchi vengono compressi in formato colonnare



Row-store per i dati più recenti, funge da writethrough per lo storage colonnare



Vantaggi:

-  Superamento del compromesso
-  Ingestione e accesso ai dati recenti ottimizzati
-  Analisi su larga scala ed efficienza di storage


9) TECNICHE DI COMPRESSIONE


TimescaleDB applica tecniche avanzate di compressione per ridurre il footprint fino al **95%** e migliorare le prestazioni I/O

Run-length encoding:
salva il valore e quante volte si ripete consecutivamente

AAAABB
↓
4A2B

Vantaggi:

 RLE: ideale per dati costanti

 Gorilla: perfetta per serie temporali dense con variazioni minime

Gorilla compression:
salva solo la differenza tra i timestamp e i bit variati nei valori numerici



Obiettivi della tesi

- ➔ **Panoramica generale sulle serie temporali**
- ➔ **Architettura e funzionalità di TimescaleDB**
- ➔ **Analisi di performance di TimescaleDB e confronto con PostgreSQL**
- ➔ **Progettazione e realizzazione di una sperimentazione pratica per applicazione IoT**
- ➔ **Sviluppi futuri**

10) TimescaleDB vs PostgreSQL

Valutazione qualitativa —> quantitativa

- **Dataset crescenti:** 1.000 -> 10.000.000 record
- **100 esecuzioni**
- **Calcolo media & intervallo di confidenza**



matplotlib

 SciPy

1 1) TimescaleDB vs PostgreSQL

Pipeline ambiente di test



```
[0, 30]  
1000, 10000, 100000, 1000000, 10000000  
SELECT * FROM sensor_data;  
SELECT * FROM sensor_data WHERE etc...
```

```
.  
. .
```


12) TimescaleDB vs PostgreSQL

- Le query analizzate:

Q1

Full scan (tutti i record)

Q2

Range Filter (intervallo temporale)

Q3

Future Average (media da data futura)

Q4

Daily Average (media giornaliera)

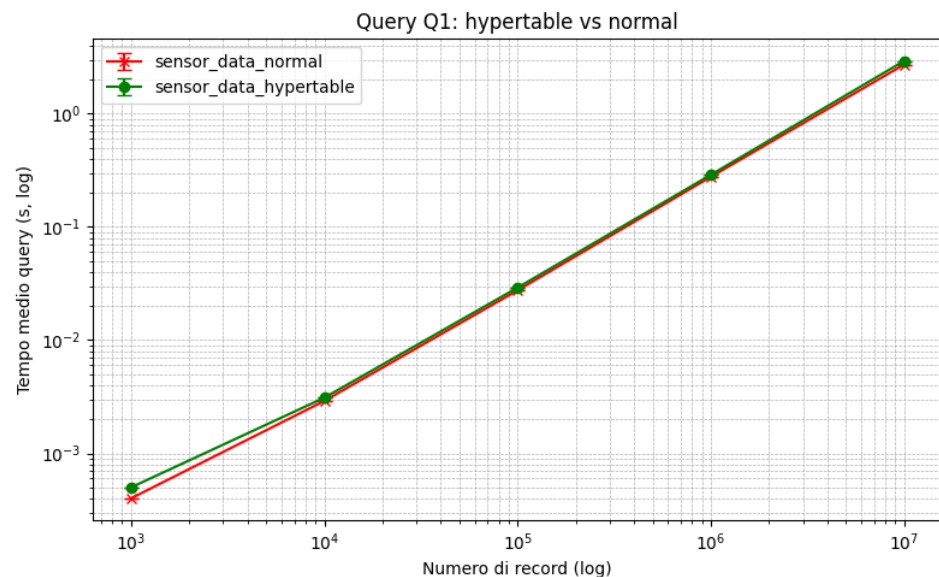
Q5

Rolling Window Avg (media mobile su finestra)

13) TimescaleDB vs PostgreSQL

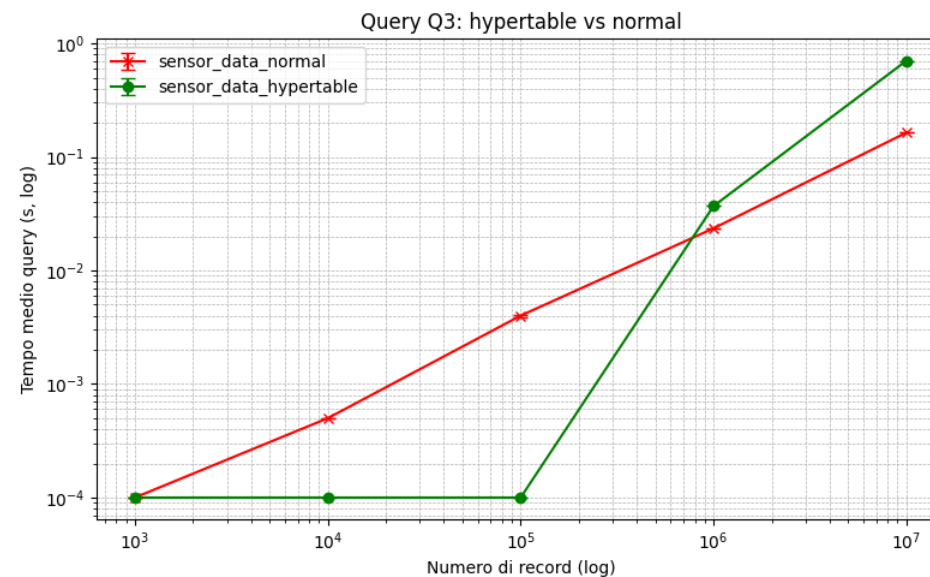
Q1

Full scan (tutti i record)



Q3

Future Average (media da data futura)

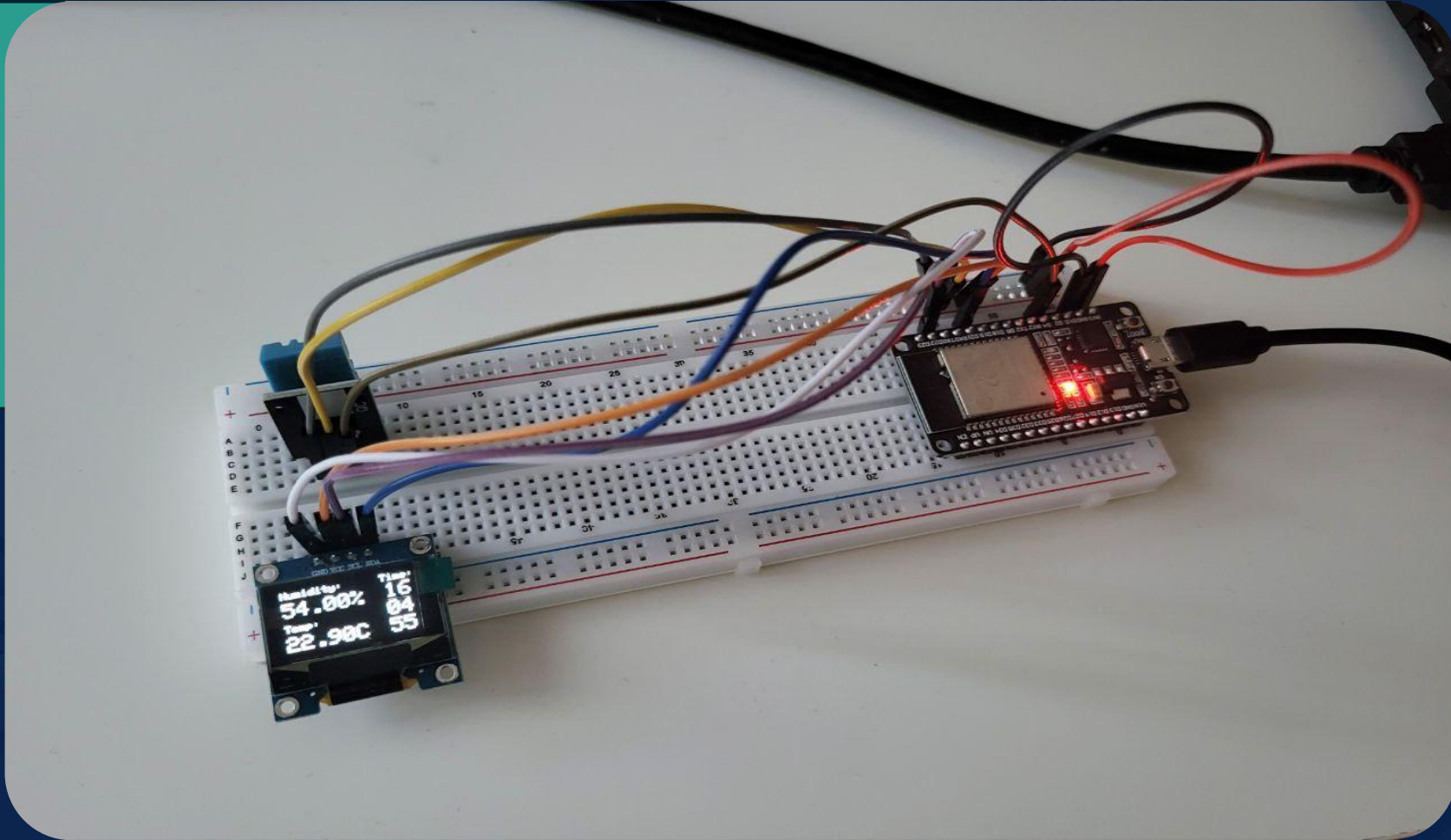


Obiettivi della tesi

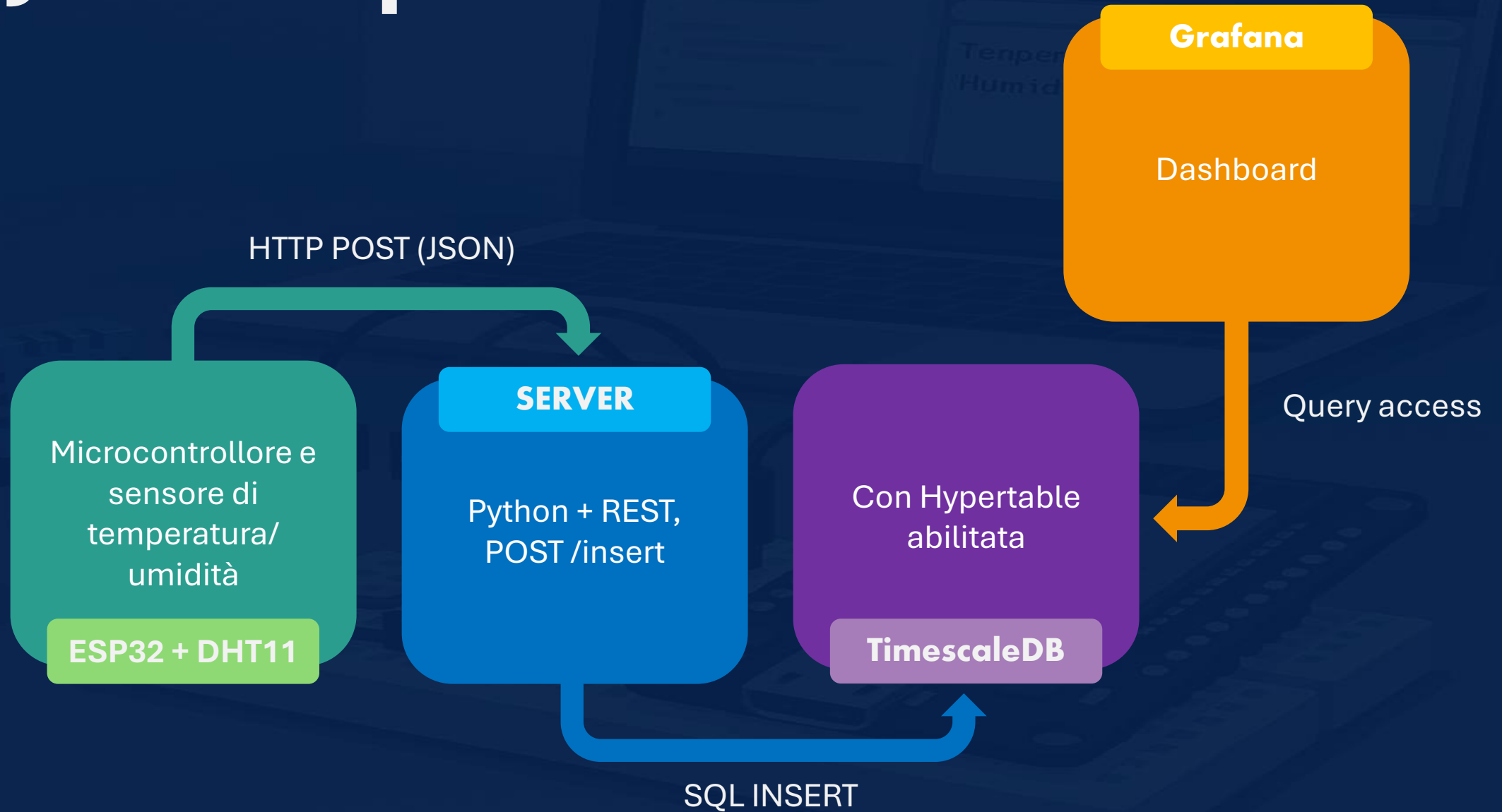
- ➔ **Panoramica generale sulle serie temporali**
- ➔ **Architettura e funzionalità di TimescaleDB**
- ➔ **Analisi di performance di TimescaleDB e confronto con PostgreSQL**
- ➔ **Progettazione e realizzazione di una sperimentazione pratica per applicazione IoT**
- ➔ **Sviluppi futuri**

14) Prototipo IoT

ESP32 + DHT11



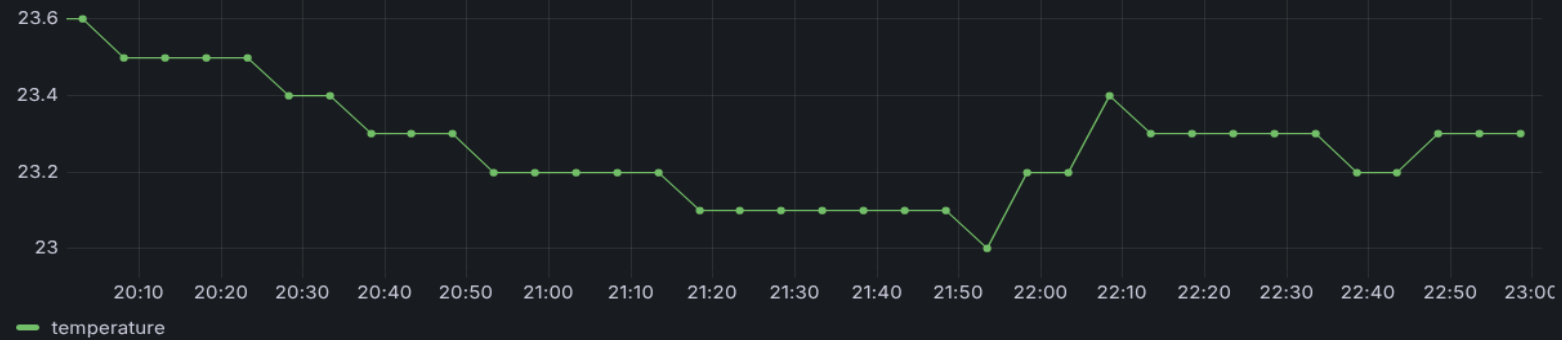
15) Prototipo IoT



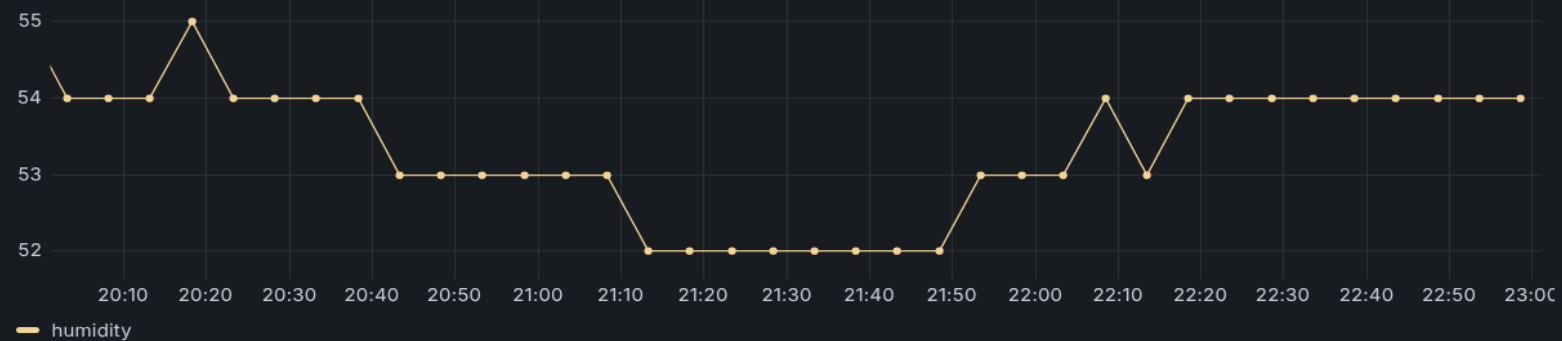
16) Prototipo IoT

Grafana

Temperature



Humidity



Obiettivi della tesi

- ➔ **Panoramica generale sulle serie temporali**
- ➔ **Architettura e funzionalità di TimescaleDB**
- ➔ **Analisi di performance di TimescaleDB e confronto con PostgreSQL**
- ➔ **Progettazione e realizzazione di una sperimentazione pratica per applicazione IoT**
- ➔ **Sviluppi futuri**

17) Sviluppi futuri

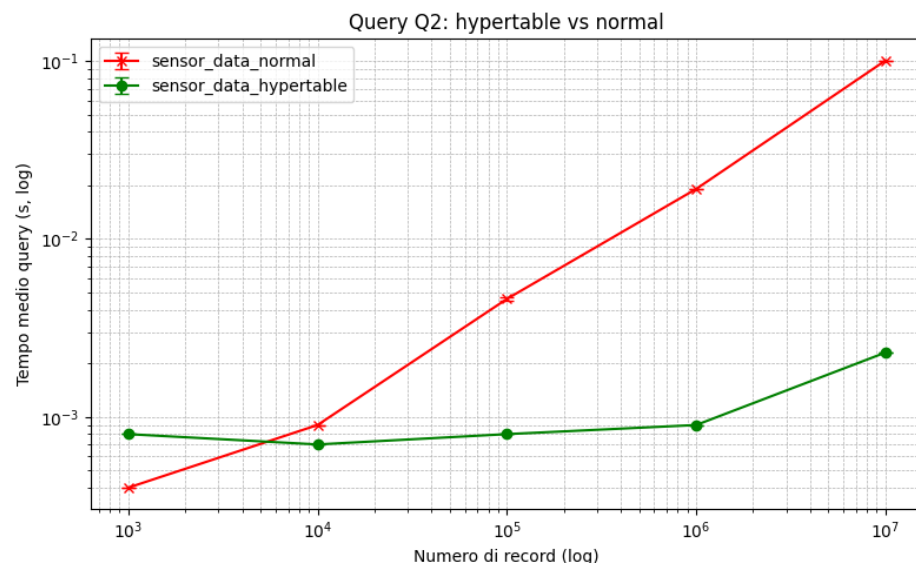
- ➔ **Estensione dell'analisi delle prestazioni rispetto ad altri TSDB**
- ➔ **Integrazione con pipeline di data engineering e Time Series Analysis**
- ➔ **Valutazione delle estensioni edge computing**
- ➔ **Implementazione di architetture peer-to-peer (P2P)**

Grazie a tutti

18a) Le altre query

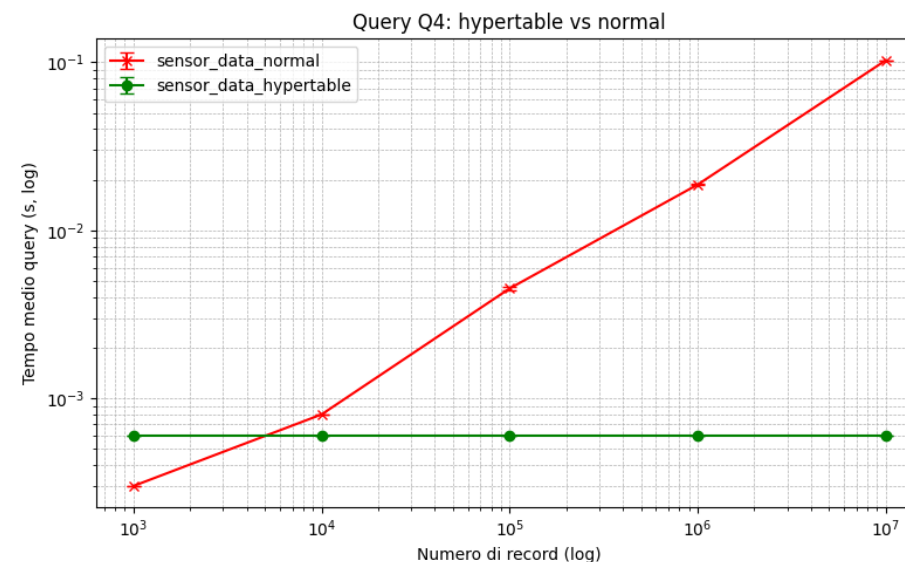
Q2

Range Filter (intervallo temporale)



Q4

Daily Average (media giornaliera)



18b) Le altre query

Q5

Rolling Window Avg
(media mobile su finestra)

