VIŠJA STROKOVNA ŠOLA ACADEMIA

MARIBOR

Primerjava podatkovnih baz časovnih vrst: NoSQL InfluxDB in SQL TimescaleDB

Kandidat: Gregor Javor

Vrsta študija: študent izrednega študija

Študijski program: Informatika

Mentor predavatelj: mag. Ervin Schaff

Mentor v podjetju: Mitja Turnšek dipl. inž. rač. in inf. tehnol. (UN)

Lektor: Katja Soršak Godec prof. slov. in ang.

Maribor, 2023

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Podpisani, Gregor Javor, sem avtor diplomskega dela z naslovom Primerjava podatkovnih baz časovnih vrst: NoSQL InfluxDB in SQL TimescaleDB, ki sem ga napisal pod mentorstvom mag. Ervina Schaffa.

S svojim podpisom zagotavljam, da:

- je predloženo delo izključno rezultat mojega dela,
- sem poskrbel, da so dela in mnenja drugih avtorjev, ki jih uporabljam v predloženi nalogi, navedena oz. citirana skladno s pravili Višje strokovne šole Academia Maribor,
- se zavedam, da je plagiatorstvo – predstavljanje tujih del oz. misli kot moje lastne - kaznivo po Zakonu o avtorski in sorodnih pravicah (Uradni list RS, št. 16/07 – uradno prečiščeno besedilo, 68/08, 110/13, 56/15 in 63/16 – ZKUASP), prekršek pa podleže tudi ukrepom Višje strokovne šole Academia Maribor skladno z njenimi pravili,
- skladno z 32.a členom ZASP dovoljujem Višji strokovni šoli Academia Maribor objavo diplomskega dela na spletnem portalu šole.

Maribor, november 2023 Podpis študenta:

ZAHVALA

Zahvaljujem se mentorju mag. Ervinu Shaffu za mentorstvo pri diplomskem delu. Prav tako bi se rad zahvalil mentorju v podjetju Mitji Turnšku za vso pomoč in nasvete. Posebej bi se rad zahvalil svoji družini, ki me je podpirala v času študija.

POVZETEK

V zadnjem desetletju so se pojavile številne podatkovne baze časovnih vrst, ki uporabljajo različne tehnologije in pristope. Med njimi se bomo osredotočili na dve priljubljeni podatkovni bazi, in sicer NoSQL InfluxDB in SQL TimescaleDB. Glavni cilj je bil razumeti njihovo zmogljivost, obnašanje in skalabilnost, zlasti v kontekstu Big Data.

Pri zapisovanju podatkov je TimescaleDB jasno prevladovala s hitrejšimi odzivnimi časi, kar je ključnega pomena za aplikacije v realnem času. Vendar pa je bilo branje podatkov bolj zapleteno. InfluxDB je v nekaterih primerih dosegla zelo visoke odzivne čase, zlasti pri večjih naborih podatkov, medtem ko je bila pri manjših naborih podatkov izredno učinkovita.

Pri testiranju različnih časovnih intervalov je TimescaleDB dosledno izkazala hitrejše odzivne čase. Kljub temu je TimescaleDB pokazala večjo variabilnost v odzivnih časih, kar dokazuje višja standardna deviacija v primerjavi z InfluxDB. Obe bazi sta pokazali izjemno zanesljivost z 0% napakami skozi vse teste.

Ko smo preučevali možnosti skaliranja, je InfluxDB v vseh testih dosledno dosegala hitrejše odzivne čase kot TimescaleDB. Hkrati je TimescaleDB pri večjem številu uporabnikov pokazala visok odstotek napak, kar je zaskrbljujoče. V smislu podpore sta obe podatkovni bazi močni rešitvi z obsežno podporo. InfluxDB se ponaša z integriranim ekosistemom, medtem ko TimescaleDB združuje moč relacijskega sistema za upravljanje z bazami podatkov s funkcionalnostjo za časovne serije.

Glede namestitve je InfluxDB morda nekoliko lažja za namestitev kot "plug-and-play" rešitev, medtem ko TimescaleDB zahteva dodaten korak namestitve PostgreSQL.

Izbira med InfluxDB in TimescaleDB je odvisna od specifičnih potreb in okoliščin uporabe. Obe podatkovni bazi imata svoje prednosti in slabosti, zato je ključnega pomena, da uporabniki ali organizacije skrbno pretehtajo svoje potrebe pred izbiro ene ali druge.

Ključne besede: *časovne vrste, Big Data, skalabilnost, zmogljivost, zanesljivost, podatkovne baze, NoSQL, SQL*

ABSTRACT

Comparison of NoSQL InfluxDB and SQL TimescaleDB time series databases.

Within the framework of this thesis, we focused on comparing two popular time series databases: NoSQL InfluxDB and SQL TimescaleDB. The main goal was to understand their performance, behavior, and scalability, especially in the context of Big Data.

When writing data, TimescaleDB clearly prevailed with faster response times, which is crucial for real-time applications. However, reading data was more complex. InfluxDB achieved very high response times in some cases, especially with larger data sets, while it was extremely efficient with smaller data sets.

When testing different time intervals, TimescaleDB consistently showed faster response times. Nevertheless, TimescaleDB showed greater variability in response times, indicating its higher standard deviation compared to InfluxDB. Both databases showed exceptional reliability with 0% errors throughout all tests.

When examining scaling options, InfluxDB consistently achieved faster response times in all tests than TimescaleDB. At the same time, TimescaleDB showed a high percentage of errors with a larger number of users, which is concerning. In terms of support, both databases are strong solutions with extensive support. InfluxDB boasts an integrated ecosystem, while TimescaleDB combines the power of a relational database management system with time series functionality.

Regarding installation, InfluxDB might be slightly easier to install as a "plug-and-play" solution, while TimescaleDB requires an additional step of installing PostgreSQL.

Choosing between InfluxDB and TimescaleDB depends on specific needs and use case circumstances. Both databases have their advantages and disadvantages, so it's crucial for users or organizations to carefully weigh their needs before choosing one over the other.

Keywords: *Time series, Big Data, scalability, performance, reliability, databases, NoSQL, SQL.*

Kazalo vsebine

Kazalo slik

Kazalo tabel

Kazalo grafov

SEZNAM UPORABLJENIH KRATIC

Kratica	**Angleško**	**Slovensko**
IOT	Internet of Things	Internet stvari
SQL	Structured Query Language	Strukturirani poizvedovalni jezik
NoSQL	Not Only SQL	Ne samo SQL
TSDB	Time Series Database	Podatkovna baza časovnih vrst
CLI	Command Line Interface	Vmesnik ukazne vrstice
UI	User Interface	Uporabniški vmesnik
GUI	Graphical User Interface	Grafični uporabniški vmesnik
UTC	Coordinated Universal Time	Koordinirani univerzalni čas

SLOVAR STROKOVNIH IZRAZOV

Angleško	**Slovensko**	**Opis**
Time series	Časovna vrsta	Časovna vrsta je zaporedje podatkov, ki so organizirani glede na časovne žige
Bucket	Vedro	Vedro se uporablja za shranjevanje in organizacijo časovnih vrst
Organization	Organizacija	Strukturna enota, ki omogoča upravljanje in nadzor nad podatki ter uporabniki v sistemu.
Timestamp	Časovni žig	Oznaka, ki predstavlja točen čas in datum. Lahko je v različnih formatih, najpogosteje pa je predstavljen kot število sekund, ki so pretekle od začetka Unixovega časa (1. januar 1970, 00:00:00 UTC).
Windows PowerShell	Windows PowerShell	Orodje za avtomatizacijo in upravljanje sistema v okolju Windows.
Dashboard	Nadzorna plošča	Grafični vmesnik, ki na enem mestu prikazuje ključne informacije ali podatke.
Extension	Podaljšek	Podaljšek se nanaša na dodatno programsko funkcionalnost ali komponento, ki poveča zmogljivost programa.
Tag	Oznaka	Oznaka se uporablja za označevanje ali kategorizacijo elementov za lažjo organizacijo ali iskanje.
Field	Polje	Je ena od vrednosti ali lastnosti v strukturi podatkovne baze časovne vrste.
Measurement	Meritev	Uporablja se za opisovanje kategorije podatkov. Meritve predstavljajo podatke, povezane z istim tipom entitete ali dogodka.

Sharding	Reženje	Metoda razdeljevanja velikih baz podatkov na manjše enote za izboljšanje učinkovitosti.
New Line Protocol	Protokol nove vrstice	Protokol nove vrstice se nanaša na način, kako so podatki strukturirani in prenašani v besedilni obliki, običajno v več vrsticah.

1. UVOD

V dobi digitalizacije, kjer se podatki ustvarjajo, zbirajo in obdelujejo z izjemno hitrostjo, je upravljanje s podatki postalo ključno za uspeh v skoraj vsakem sektorju. Podatki so postali dragocen vir informacij, ki lahko podjetjem in organizacijam pomagajo pri odločanju, načrtovanju in optimizaciji svojih operacij. V tem kontekstu podatkovne baze časovnih vrst postajajo vse bolj pomembne, saj omogočajo učinkovito shranjevanje in obdelavo časovno označenih podatkov.

1.1. Opis področja in opredelitev problema

Podatkovne baze časovnih vrst so specializirane podatkovne baze, ki so zasnovane za shranjevanje in obdelavo časovno označenih podatkov. Te podatkovne baze so optimizirane za hitro vstavljanje in poizvedovanje podatkov, ki so urejeni po času (Noor Zehra & Yfantidou, 2017). Časovne vrste so pogoste v mnogih aplikacijah, vključno z borznimi tečaji, meritvami senzorjev, podatki o prodaji in tako naprej.

V zadnjem desetletju so se pojavile številne podatkovne baze časovnih vrst, ki uporabljajo različne tehnologije in pristope. Med njimi se bomo osredotočili na dve podatkovni bazi, in sicer NoSQL InfluxDB in SQL TimescaleDB. InfluxDB je NoSQL podatkovna baza, ki je zasnovana za obdelavo visokozmogljivih časovnih vrst, medtem ko je TimescaleDB razširitev za podatkovno bazo PostgreSQL, ki omogoča učinkovito shranjevanje in obdelavo časovnih vrst v relacijski podatkovni bazi.

V tem diplomskem delu se bomo osredotočili na primerjavo zgoraj omenjenih podatkovnih baz časovnih vrst. Cilj je razumeti njihove značilnosti, prednosti in slabosti ter ugotoviti, v katerih scenarijih je ena podatkovna baza bolj primerna kot druga.

Rezultati te diplome bodo pomagali podjetjem in organizacijam pri izbiri prave podatkovne baze časovnih vrst za njihove potrebe. Poleg tega bo ta diploma prispevala k boljšemu razumevanju podatkovnih baz časovnih vrst in njihove vloge v sodobni tehnologiji in poslovanju.

1.2. Namen, cilji in hipoteze

Namen tega diplomskega dela je primerjati dve priljubljeni, vendar zelo različni podatkovni bazi časovnih vrst, InfluxDB in TimescaleDB, z namenom, da ugotovimo, katera je bolj primerna za različne scenarije uporabe. Ta primerjava bo temeljila na več kriterijih, vključno z zmogljivostjo, podporo, enostavnostjo namestitve in primernostjo za uporabo z Big Data.

Cilji tega diplomskega dela so naslednji:

- Primerjati zmogljivost InfluxDB in TimescaleDB.
- Oceniti raven podpore, ki je na voljo za vsako podatkovno bazo.
- Oceniti enostavnost namestitve vsake podatkovne baze.
- Ugotoviti, katera podatkovna baza je bolj primerna za uporabo z Big Data.

Hipoteze, ki jih bo ta diploma preverila, so naslednje:

Hipoteza 1: InfluxDB podatkovna baza je hitrejša od TimescaleDB podatkovne baze. Ta hipoteza bo preverjena z izvajanjem različnih benchmark testov na obeh podatkovnih bazah.

Hipoteza 2: InfluxDB podatkovna baza ima večjo podporo. Ta hipoteza bo preverjena z analizo razpoložljivih virov kot so dokumentacija, skupnosti uporabnikov, forumi in tako naprej.

Hipoteza 3: TimescaleDB podatkovna baza je lažja za namestitev. Ta hipoteza bo preverjena z ocenjevanjem postopka namestitve za vsako podatkovno bazo.

Hipoteza 4: InfluxDB podatkovna baza je primernejša za uporabo z Big Data. Ta hipoteza bo preverjena z analizo zmogljivosti in funkcionalnosti vsake podatkovne baze v kontekstu obdelave velikih količin podatkov.

S tem diplomskim delom želimo prispevati k boljšemu razumevanju podatkovnih baz časovnih vrst in njihove vloge v sodobni tehnologiji in poslovanju.

1.3. Predpostavke in omejitve

V tem diplomskem delu se soočamo z nekaterimi predpostavkami in omejitvami, ki jih je treba upoštevati.

Predpostavke:

- Predpostavljamo, da so vse meritve in testi, ki se izvajajo za primerjavo zmogljivosti InfluxDB in TimescaleDB, pravični in nepristranski. To pomeni, da so testi zasnovani in izvedeni na način, ki ne favorizira nobene od podatkovnih baz.
- Predpostavljamo, da so vse informacije, ki se uporabljajo za oceno ravni podpore in enostavnosti namestitve za vsako podatkovno bazo, točne in ažurne. To pomeni, da so informacije pridobljene iz zanesljivih virov in so relevantne za trenutno različico vsake podatkovne baze.

Omejitve:

- Omejitve zmogljivosti in funkcionalnosti vsake podatkovne baze so odvisne od specifičnih zahtev in scenarijev uporabe. Zato rezultati te diplome morda ne bodo veljali za vse scenarije uporabe.
- V tem diplomskem delu se bomo osredotočali samo na dve podatkovni bazi časovnih vrst, InfluxDB in TimescaleDB. Obstaja veliko drugih podatkovnih baz časovnih vrst, ki jih ta diploma ne obravnava.
- Rezultati te diplome so odvisni od specifičnih testov in meritev, ki se izvajajo. Drugačne metode testiranja ali merjenja bi lahko privedle do drugačnih rezultatov.
- Ne bomo obravnavali vseh možnih vidikov vsake podatkovne baze, kot so varnost, stroški, združljivost z drugimi tehnologijami in tako naprej. Fokus je na zmogljivosti, podpori, enostavnosti namestitve in primernosti za uporabo z Big Data.

1.4. Uporabljene raziskovalne metode

Pri izdelavi tega diplomskega dela smo uporabili več raziskovalnih metod, ki so nam pomagale pri primerjavi podatkovnih baz časovnih vrst InfluxDB in TimescaleDB. Te metode vključujejo:

- **Pregled literature**: Izvedli smo temeljit pregled literature, ki vključuje znanstvene članke, tehnično dokumentacijo, blogovske objave in druge vire, kakor tudi pregled lastnih zapiskov, da bi pridobili globlje razumevanje obeh podatkovnih baz. Ta pregled literature nam je pomagal razumeti ključne značilnosti, prednosti in slabosti vsake podatkovne baze.
- **Eksperimentalno testiranje**: Izvedli smo praktično namestitev obeh podatkovnih baz, da bi ocenili njihovo enostavnost namestitve. Ta postopek nam je pomagal razumeti, katera podatkovna baza je lažja za namestitev. Nato smo na obeh podatkovnih bazah izvedli serijo benchmark testov, da bi primerjali njihovo zmogljivost. Ti testi so vključevali operacije, kot so vstavljanje, poizvedovanje in brisanje podatkov. Rezultati teh testov so nam pomagali ugotoviti, katera podatkovna baza je hitrejša in bolj učinkovita.
- **Analiza**: Ocenili smo raven podpore, ki je na voljo za vsako podatkovno bazo, z analizo razpoložljivih virov, kot so dokumentacija, skupnosti uporabnikov, forumi in tako naprej. Ta analiza nam je pomagala ugotoviti, katera podatkovna baza ima večjo podporo. Nato smo naredili analizo rezultatov benchmark testiranja, ki je ključna za razumevanje zmogljivosti in učinkovitosti podatkovnih baz časovnih vrst. Ta analiza vključuje pregled in interpretacijo rezultatov testov, ki so bili izvedeni na obeh podatkovnih bazah. Hkrati smo preučili, kako vsaka podatkovna baza obvladuje obdelavo velikih količin podatkov, da bi ugotovili, katera je bolj primerna za uporabo z Big Data.

Te metode so nam omogočile, da smo izvedli celovito in objektivno primerjavo med InfluxDB in TimescaleDB. Rezultati te primerjave bodo pomagali podjetjem in organizacijam pri izbiri prave podatkovne baze časovnih vrst za njihove potrebe.

2. PODATKOVNE BAZE

2.1. Definicija in opis podatkovnih baz

Podatkovna baza je zbirka podatkov, ki so organizirani na strukturiran način, običajno z uporabo različnih podatkovnih modelov, in je zasnovana tako, da omogoča učinkovito in zanesljivo shranjevanje, upravljanje in dostopanje do informacij. Temeljna ideja podatkovne baze je omogočiti enostaven in hiter dostop do podatkov ter zagotoviti doslednost in celovitost podatkov med različnimi uporabniki. Podatkovne baze so temeljni gradniki mnogih modernih informacijskih sistemov. Omogočajo nam varno hranjenje, upravljanje in analizo velikih količin podatkov. Podatki so strukturirani in razvrščeni v tabele, kjer vsaka tabela predstavlja različne tipe podatkov. Podatkovne baze delujejo na osnovi sistema za upravljanje podatkovnih baz (DBMS), ki uporabnikom omogoča interakcijo s podatki. SQL je najpogosteje uporabljen jezik za delo z relacijskimi podatkovnimi bazami, omogoča pa kreiranje, brisanje, pridobivanje in posodabljanje zapisov (Silberschatz, Korth, & Sudarshan, 2019).

Obstajajo različni tipi podatkovnih baz. Relacijske podatkovne baze, kot sta MySQL in PostgreSQL, uporabljajo strukturirane tabele za shranjevanje podatkov. NoSQL baze, kot so MongoDB in Cassandra, pa so fleksibilne in prilagodljive, omogočajo pa hranjenje nestrukturiranih podatkov.

Danes podatkovne baze igrajo ključno vlogo v podjetjih vseh velikosti. Uporabljajo se za različne aplikacije, vključno z e-poslovanjem, upravljanjem odnosov s strankami (CRM), upravljanjem verige oskrbe, socialnimi mediji in več. V dobi velikih količin podatkov so podatkovne baze bistvene za analizo in izvajanje vpogledov iz obsežnih nizov podatkov.

2.2. Opis podatkovnih baz časovnih vrst

Podatkovne baze časovnih vrst (TSDB) so poseben tip podatkovnih baz, ki so optimizirane za shranjevanje in obdelavo podatkov, urejenih po času. Časovne vrste so pogosta vrsta podatkov v številnih aplikacijah, kot so finančne storitve, telekomunikacije, energetika, industrijska avtomatizacija in internet stvari (IoT).

Časovna vrsta je časovno urejeno zaporedje številčnih podatkov, zabeleženih ob določenem času, ki izražajo vrednost neke spremenljivke. Tipičen primer je merjenje temperature na

določenem mestu vsakih 10 minut. Vsaka meritev (ali zapis) je sestavljena iz časovne oznake (timestamp) in vrednosti. V tem primeru je časovna oznaka čas merjenja, vrednost pa izmerjena temperatura.

Podatkovne baze časovnih vrst so optimizirane za obvladovanje velikih količin podatkov v časovnem zaporedju. Pogosto imajo funkcionalnosti, ki omogočajo visoko učinkovitost pri pisanju, branju in obdelavi podatkov. To vključuje stiskanje podatkov, da zmanjšajo prostorske zahteve, in indeksiranje podatkov, da omogočijo hitro iskanje in dostop (Noor Zehra & Yfantidou, 2017).

TSDB ponujajo tudi napredne funkcije za analizo podatkov. To lahko vključuje povprečenje vrednosti čez določeno časovno obdobje, iskanje maksimalnih ali minimalnih vrednosti, interpolacijo manjkajočih podatkov in izračunavanje trendov ali napovedi. Vse to pomaga uporabnikom razumeti oblike, trende in vzorce v svojih podatkih.

InfluxDB, TimescaleDB in OpenTSDB so nekateri priljubljeni sistemi za upravljanje podatkovnih baz časovnih vrst. Vsak ima svoje prednosti in slabosti, vendar vsi ponujajo visoko zmogljivost in fleksibilnost.

Kot vse podatkovne baze, morajo tudi podatkovne baze časovnih vrst zagotavljati varnost, zanesljivost in razpoložljivost. To vključuje zaščito podatkov pred nepooblaščenim dostopom, varnostno kopiranje in obnovo podatkov ter zagotavljanje, da so podatki vedno na voljo za obdelavo in analizo.

Uporaba podatkovnih baz časovnih vrst se hitro povečuje, saj podjetja in organizacije iščejo boljše načine za obdelavo in analizo časovno občutljivih podatkov.

2.3. Razlika med SQL in NoSQL podatkovnimi bazami

SQL in NoSQL sta dva glavna tipa podatkovnih baz. SQL podatkovne baze so strukturirane in za shranjevanje podatkov uporabljajo tabele. NoSQL podatkovne baze so nestrukturirane in lahko uporabljajo različne strukture za shranjevanje podatkov, kot so dokumenti, ključ-vrednost in graf. SQL podatkovne baze so bolj tradicionalni tip podatkovnih baz in se uporabljajo že več desetletij. NoSQL podatkovne baze so relativno nov tip podatkovnih baz, ki postajajo vse bolj priljubljene, saj se uporabljajo za shranjevanje velikih količin nestrukturiranih podatkov (Coursera, 2023).

Ključne razlike so naslednje:

1. Struktura podatkov: SQL podatkovne baze uporabljajo relacijski model, kjer se podatki shranjujejo v strukturirane tabele. Vsak zapis (vrstica) ima določen nabor atributov (stolpcev). Nasprotno pa NoSQL podatkovne baze podpirajo različne modele podatkov, vključno z dokumenti, širokim stolpcem, grafi in ključ-vrednost, kar omogoča večjo prilagodljivost pri shranjevanju in manipulaciji nestrukturiranih in polstrukturiranih podatkov.
2. Shema: SQL podatkovne baze uporabljajo fiksno shemo, kar pomeni, da je struktura tabele (vključno s tipi podatkov za vsak stolpec) določena vnaprej. Če želite spremeniti shemo, morate običajno spremeniti celotno bazo podatkov. NoSQL podatkovne baze na drugi strani ponujajo fleksibilne sheme, kar pomeni, da lahko zapis vsebuje različne nize atributov.
3. Skalabilnost: SQL podatkovne baze običajno podpirajo vertikalno skalabilnost, kar pomeni, da lahko povečate zmogljivost s povečanjem specifikacij strežnika (npr. več RAM-a, hitrejši CPU). NoSQL podatkovne baze so zasnovane za horizontalno skalabilnost, kar pomeni, da lahko zmogljivost povečate z dodajanjem več strežnikov v bazo podatkov.
4. Transakcije: SQL podatkovne baze ponavadi podpirajo ACID transakcije (Atomicity, Consistency, Isolation, Durability), kar zagotavlja zanesljivost v več korakih operacij. NoSQL podatkovne baze običajno ne podpirajo popolnih ACID transakcij, čeprav nekatere ponujajo omejeno podporo.
5. Jezik za poizvedbe: SQL podatkovne baze uporabljajo standardiziran jezik SQL za poizvedbe, kar omogoča močne in raznolike možnosti poizvedb. NoSQL podatkovne baze uporabljajo različne jezike za poizvedbe, ki se lahko razlikujejo glede na tip baze podatkov (npr. dokument, graf, itd.).

2.4. Pregled obstoječih podatkovnih baz časovnih vrst

Izbira podatkovne baze je odvisna od različnih dejavnikov, vključno z vrsto in količino podatkov, ki jih potrebujete za shranjevanje, potrebno učinkovitostjo, razpoložljivostjo in stroški ter zahtevami za poizvedbe in analizo. Poleg tega lahko izbira med odprtokodnimi in popolnoma upravljanimi rešitvami vpliva na zahteve za administracijo in vzdrževanje.

- InfluxDB: InfluxDB, ki jo je razvil InfluxData, je odprtokodna podatkovna baza, zasnovana za obdelavo časovnih vrst. Izstopa zaradi svoje hitrosti in učinkovitosti pri shranjevanju ter

pri poizvedovanju po časovnih vrstah. Podpira SQL-u podoben jezik za poizvedbe, imenovan InfluxQL, ki omogoča enostavno filtriranje in analizo podatkov. Poleg tega je zasnovana za delovanje z minimalno administracijo, kar olajša uporabo. InfluxDB se pogosto uporablja v aplikacijah, kot so spremljanje performans, IoT, telemetrija in analitika v realnem času.

- TimescaleDB: TimescaleDB je odprtokodna podatkovna baza časovnih vrst, ki temelji na PostgreSQL, zato lahko izkoristi vse obstoječe funkcionalnosti PostgreSQL. Njena glavna prednost je, da združuje možnosti relacijske podatkovne baze (kot so transakcije in združevanje podatkov) z optimizacijami, specifičnimi za časovne vrste. TimescaleDB omogoča shranjevanje in obdelavo velikih količin časovnih vrst, hkrati pa ohranja visoko zmogljivost in prilagodljivost (Getting started with Timescale, n.d.).
- OpenTSDB: OpenTSDB (The Open Time Series Database) je razširljiva in odprtokodna podatkovna baza časovnih vrst. Zasnovana je za zbiranje, shranjevanje in obdelavo metričnih podatkov. OpenTSDB je zasnovana za delovanje na vrhu HBase (NoSQL podatkovna baza), kar omogoča zbiranje in shranjevanje milijard zapisov brez izgube zmogljivosti. OpenTSDB se pogosto uporablja za sistemsko spremljanje in spremljanje aplikacij.
- Prometheus: Prometheus je odprtokodni sistem za spremljanje in opozarjanje, ki vključuje podatkovno bazo časovnih vrst. Zasnovan je za zbiranje metrik iz različnih ciljev in jih shranjuje kot časovne vrste. Ponuja močan jezik za poizvedbe, imenovan PromQL, ki omogoča analizo časovnih vrst podatkov. Prometheus je pridobil popularnost predvsem v okoljih za obvladovanje računalniških virov (kot je Kubernetes) za spremljanje stanja in uspešnosti sistema (Overview, n.d.).
- Graphite: Graphite je odprtokodno orodje za spremljanje, ki vključuje podatkovno bazo. časovnih vrst Sestavljen je iz treh komponent: Whisper (podatkovna baza), Carbon (zbiralec podatkov) in Grafana (vizualizacijsko orodje). Graphite se pogosto uporablja za spremljanje performans strežnikov, aplikacij in omrežij.
- Amazon Timestream: Amazon Timestream je popolnoma upravljana podatkovna baza časovnih vrst, ki jo je razvil Amazon Web Services (AWS). Timestream je zasnovan za učinkovito analizo časovnih vrst in je idealen za IoT aplikacije, aplikacije za spremljanje v realnem času in telemetrijo. Kot storitev AWS, Timestream ponuja avtomatsko skaliranje, visoko razpoložljivost in varnost na ravni podjetja, kar omogoča enostavno uporabo in upravljanje.

2.5. Opis izbranih podatkovnih baz za primerjavo

V tem diplomskem delu bomo primerjali podatkovni bazi časovnih vrst InfluxDB in TimescaleDB. InfluxDB in TimescaleDB smo izbrali, ker sta oba predstavnika širokega spektra podatkovnih baz, ki so na voljo pri delu s časovnimi vrstami. InfluxDB je odprtokodna podatkovna baza, ki je bila od začetka zasnovana za delo s časovnimi vrstami. Zaradi svoje hitrosti, učinkovitosti in enostavnosti uporabe se je uveljavila kot priljubljena izbira za različne uporabe, kot so monitoring sistemov, IoT in analitika v realnem času.

Na drugi strani TimescaleDB predstavlja drugačen pristop k časovnim vrstam. Zgrajen na osnovi PostgreSQL, prinaša vse zmogljivosti, zanesljivost in priljubljenost relacijske podatkovne baze ter jim dodaja optimizacije, specifične za časovne vrste.

Obe podatkovni bazi ponujata močne in edinstvene funkcionalnosti, vendar se razlikujeta v načinu njune uporabe in v tem katere vrste problemov najbolje rešujeta. Zato se mi je zdelo zanimivo in pomembno raziskati razlike med njima, njune prednosti in slabosti, ter v katerih scenarijih je ena ali druga najbolj primerna. Z usmeritvijo na dve vodilni tehnologiji v prostoru podatkovnih baz časovnih vrst sem želel zagotoviti, da bo moja raziskava relevantna in uporabna za široko paleto bralcev, ki se ukvarjajo z uporabo časovnih vrst.

2.5.1. TimescaleDB

TimescaleDB je odprtokodna podatkovna baza časovnih vrst, ki temelji na PostgreSQL, kar zagotavlja kombinacijo visoko zmogljive obdelave časovnih vrst in zmogljivosti relacijske podatkovne baze. TimescaleDB se uporablja predvsem za analizo velikih količin časovnih vrst, kar je zelo pomembno v mnogih aplikacijah, kot so IoT, finančne storitve, spremljanje sistemov in drugo.

TimescaleDB je bila razvita z namenom, da bi obvladala specifične izzive, ki jih prinašajo časovne vrste. Značilno je, da se te vrste podatkov kopičijo zelo hitro in lahko hitro narastejo na velike količine.

Zasnova TimescaleDB je optimizirana za te vrste podatkov. Vsaka vrstica podatkov je shranjena skupaj z njenim časovnim žigom, kar omogoča hitro in učinkovito poizvedovanje po podatkih. TimescaleDB je zasnovana tako, da podpira visoko hitrost vstavljanja podatkov, hkrati pa ohranja visoko zmogljivost pri poizvedovanju po podatkih.

Ena od glavnih prednosti TimescaleDB je njena tesna integracija s PostgreSQL. Zaradi tega TimescaleDB podeduje veliko zmogljivosti PostgreSQL, vključno z robustnostjo, zanesljivostjo, podporo za ACID transakcije, podporo za kompleksne poizvedbe SQL, kot so JOINs in sub-queries, ter široko paleto indeksov. Zaradi tega TimescaleDB ne omogoča le učinkovitega delovanja s časovnimi vrstami, temveč tudi visoko funkcionalnost in prilagodljivost, ki jo pričakujemo od relacijske podatkovne baze.

Poleg tega TimescaleDB vključuje tudi nekatere funkcije, ki so specifične za časovne vrste, kot je avtomatsko particioniranje podatkov po času (t.i. time-partitioning). To pomeni, da TimescaleDB samodejno razdeli podatke na več "časovnih kosov" (ang. time chunks), kar omogoča boljše upravljanje in učinkovito poizvedovanje velikih količin podatkov.

TimescaleDB je zasnovana tudi za horizontalno skaliranje. To pomeni, da lahko za povečanje zmogljivosti sistema dodate več strežnikov. To je zelo pomembno za velike aplikacije, kjer količina podatkov nenehno narašča. Sistem lahko enostavno prilagajate, da se prilega vašim potrebam, brez da bi morali žrtvovati hitrost ali učinkovitost (Getting started with Timescale, n.d.).

Na splošno je TimescaleDB zmogljiva in prilagodljiva rešitev za obdelavo časovnih vrst. S svojo tesno integracijo s PostgreSQL omogoča združevanje najboljših lastnosti podatkovnih baz časovnih vrst kakor tudi relacijskih podatkovnih baz, kar prinaša visoko zmogljivost, robustnost in prilagodljivost. Z možnostjo horizontalnega skaliranja in optimizacijami za časovne vrste je idealna rešitev za obdelavo in analizo velikih količin časovnih vrst.

2.5.2. InfluxDB

InfluxDB je odprtokodna podatkovna baza časovnih vrst, ki je bila razvita s strani InfluxData, podjetja, ki se osredotoča na razvoj rešitev za upravljanje in analizo časovnih vrst. InfluxDB je bila zasnovana z namenom obvladovanja velikih količin časovno označenih podatkov, ki se pogosto pojavljajo v aplikacijah, kot so IoT, spremljanje sistemov, analitika v realnem času in druge.

Za razliko od tradicionalnih relacijskih podatkovnih baz ali podatkovnih baz, ki temeljijo na SQL, InfluxDB uporablja lasten poizvedbeni jezik, znan kot InfluxQL, ki je zasnovan posebej za delo s časovnimi vrstami. InfluxQL podpira številne funkcije, ki so koristne pri delu s

časovnimi vrstami, kot so agregacija, grupiranje po časovnih intervalih in matematične funkcije, ki delujejo na časovnih serijah.

InfluxDB je zasnovana za visoko zmogljivost in učinkovitost, kar omogoča hitro vstavljanje in poizvedovanje podatkov. To je še posebej pomembno pri delu z velikimi količinami časovnih vrst, kjer se hitro kopičijo nove podatkovne točke in je potrebna hitra analiza podatkov.

Ena od ključnih značilnosti InfluxDB je njena fleksibilnost pri modeliranju podatkov. Vsak podatek v InfluxDB je povezan z nizom ključ-vrednost parov, znanih kot oznake (tags), ki omogočajo visoko fleksibilnost pri poizvedovanju in organizaciji podatkov. Oznake so indeksirane, kar pomeni, da lahko InfluxDB hitro izvaja poizvedbe, ki temeljijo na oznakah (Introduction to InfluxDB, n.d.).

InfluxDB vključuje tudi funkcionalnosti, kot so stalno shranjevanje in replikacija podatkov, ki pomagajo zagotavljati zanesljivost in razpoložljivost podatkov. Poleg tega InfluxDB podpira funkcije kot je sharding, ki omogoča razdeljevanje podatkov med več strežnikov, da se izboljša zmogljivost in zanesljivost.

InfluxDB ne vključuje le podatkovne baze, ampak tudi celovit okvir za obdelavo časovnih vrst. To vključuje podporo za analizo podatkov, vizualizacijo, obveščanje in drugo. To pomeni, da lahko uporabniki uporabljajo InfluxDB ne le za shranjevanje podatkov, ampak tudi za izvajanje kompleksnih analiz in izdelavo poročil.

Na splošno je InfluxDB zmogljiva in prilagodljiva rešitev za delo s časovnimi vrstami. Njena zmožnost visoko zmogljivega vstavljanja in poizvedovanja podatkov, skupaj z obsežno podporo za modeliranje podatkov in analizo, jo naredi idealno za številne uporabe, ki zahtevajo hitro in učinkovito obdelavo časovnih vrst.

3. BIG DATA

3.1. Kaj je Big Data?

Izraz Big Data se nanaša na ogromne količine podatkov, ki so tako obsežni in kompleksni, da jih tradicionalne metode obdelave podatkov ne morejo učinkovito obdelati. Big Data je postal ključni del sodobne tehnologije in poslovanja, saj se podatki uporabljajo za oblikovanje boljših izdelkov, boljših odločitev in boljših strategij. Big Data se lahko nanaša na katero koli vrsto podatkov, vključno s strukturiranimi podatki, kot so podatkovne baze, nestrukturiranimi podatki, kot so besedila, slike, video posnetki, in polstrukturiranimi podatki, kot je e-pošta. Vendar pa se izraz najpogosteje uporablja za opis podatkov, ki so tako obsežni, da jih je težko obdelati z običajnimi orodji za obdelavo podatkov (Characteristics of Big Data: Types & Examples, 2020).

Big Data se uporablja v številnih sektorjih, vključno z zdravstvom, trgovino, financami, znanostjo in vlado. Analiza Big Data lahko pomaga pri napovedovanju trendov, izboljšanju storitev, zmanjševanju tveganj in ustvarjanju novih priložnosti. V zdravstvu se na primer lahko Big Data uporablja za napovedovanje izbruhov bolezni, v trgovini pa za prilagajanje ponudbe in povpraševanja.

3.2. Značilnosti Big Data

Značilnosti Big Data se pogosto opisujejo s petimi "V-ji" in sicer Volume (Obseg), Velocity (Hitrost), Variety (Raznolikost), Veracity (Verodostojnost) in Value (Vrednost).

- **Volume (Obseg)** se nanaša na količino podatkov, ki jih je treba obdelati. Big Data vključujejo ogromne količine podatkov, ki se merijo v terabajtih ali celo petabajtih. Ta obseg podatkov je posledica naraščajoče digitalizacije vseh vidikov našega življenja, od socialnih medijev do e-poslovanja in IOT.
- **Velocity (Hitrost)** se nanaša na hitrost, s katero se podatki ustvarjajo, obdelujejo in analizirajo. V digitalni dobi se vsak dan ustvari ogromna količina podatkov. Hitrost je ključna za podjetja, ki želijo izkoristiti podatke v realnem času.
- **Variety (Raznolikost)** se nanaša na različne vrste podatkov, ki jih je treba obdelati. Podatki Big Data prihajajo iz različnih virov in so v različnih formatih, vključno s tekstovnimi,

slikovnimi, zvočnimi in video datotekami. Ta raznolikost lahko predstavlja izziv, saj je treba podatke standardizirati, preden jih je mogoče analizirati.

- **Veracity (Verodostojnost)** se nanaša na kakovost in točnost podatkov. Zaradi raznolikosti virov in hitrosti, s katero se podatki ustvarjajo, je lahko kakovost podatkov neenakomerna. To pomeni, da je treba podatke preveriti, preden se jih uporabi.
- **Value (Vrednost)** se nanaša na uporabnost podatkov. Kljub izzivom, ki jih predstavljajo obseg, hitrost, raznolikost in verodostojnost, imajo Big Data podatki veliko vrednost za organizacije, ki jih znajo pravilno uporabiti. S pravilno analizo lahko Big Data ponudi dragocene vpoglede, ki lahko pomagajo pri odločanju (Characteristics of Big Data: Types & Examples, 2020).

Big Data se lahko uporablja v različnih panogah, kot so finančni sektor, zdravstvo, trgovina na drobno, proizvodnja in transport. Nekateri primeri uporabe Big Data vključujejo:

- **Identifikacijo prevar**: Big Data se lahko uporablja za identifikacijo prevar v finančnih transakcijah, zdravniških preiskavah in prodaji.
- **Izboljšanje storitev za stranke**: Big Data se lahko uporablja za boljše razumevanje potreb strank in za ustvarjanje bolj personaliziranih izkušenj.
- **Napovedovanje**: Big Data se lahko uporablja za napovedovanje prihodnjih dogodkov, kot so prodaja, povpraševanje in napake.
- **Optimizacijo procesov**: Big Data se lahko uporablja za optimizacijo poslovnih procesov, kot so proizvodnja, distribucija in logistika.

Pomembno je, da se zavedamo izzivov, povezanih z Big Data, kot so varnost, zasebnost in etičnost.

3.3. Big Data in podatkovne baze časovnih vrst

Big Data in podatkovne baze časovnih vrst sta dve ključni področji, ki sta v središču sodobne tehnologije in poslovanja. Oba koncepta sta tesno povezana in se pogosto uporabljata skupaj za reševanje kompleksnih problemov in izboljšanje odločanja.

Pogosto se uporabljata skupaj v različnih aplikacijah. Na primer, podjetje bi lahko zbralo velike količine podatkov iz različnih virov (npr. prodajne transakcije, podatki o uporabnikih, podatki o izdelkih), jih shranilo v podatkovno bazo časovnih vrst in nato uporabilo različne analitične

tehnike za pridobivanje vpogledov iz teh podatkov. Ti vpogledi bi lahko podjetju pomagali pri odločanju, načrtovanju in optimizaciji svojih operacij.

Vendar pa uporaba Big Data in podatkovnih baz časovnih vrst prinaša tudi izzive. Na primer, obdelava in analiza velikih količin podatkov zahteva močne računalniške vire in napredne analitične tehnike. Poleg tega je treba zagotoviti, da so podatki točni, zanesljivi in varni. Kljub tem izzivom pa Big Data in podatkovne baze časovnih vrst ponujajo velike priložnosti za izboljšanje odločanja in operacij v različnih sektorjih.

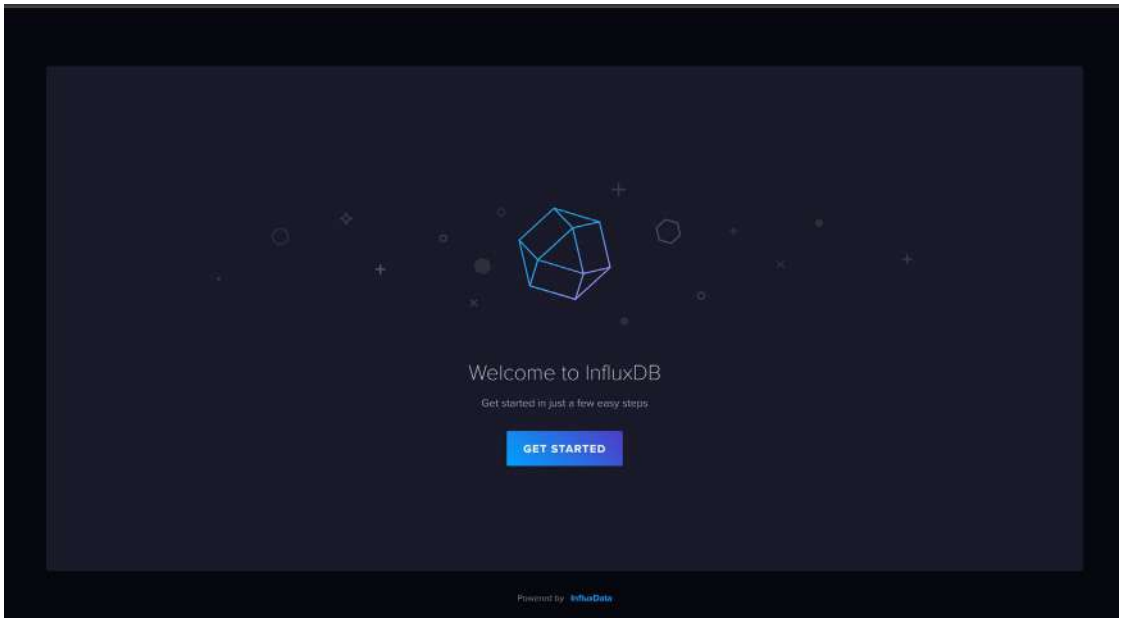
4. Primerjava InfluxDB in TimescaleDB podatkovne baze časovnih vrst

4.1. Vzpostavitev okolja

4.1.1. Namestitev InfluxDB

Zadnja verzija InfluxDB je izšla v mesecu aprilu 2023. Uveden je bil nov model podatkov, nov jezik poizvedb in nova arhitektura. Vendar je trenutno verzija 3.0 na voljo samo kot storitev v oblaku in ker brezplačna verzija ponuja zelo omejene možnosti, mi pa želimo naše podatkovne baze primerjati v istem okolju, smo se odločili za lokalno namestitev odprtokodne verzije 2.7. Ob odprtju spletne strani lahko opazimo zavihek >>Community<<, kjer najdemo povezavo do uradnega Slack kanala, Influx univerze, dokumentacije in njihovega foruma. Med veliko količino dokumentacije se nahaja tudi dokumentacija, ki obsega namestitev podatkovne baze. V primeru, da se iz dokumentacije ne znajdemo, so nam na voljo tudi video navodila znotraj Influx univerze. Sama namestitev je zelo enostavna.

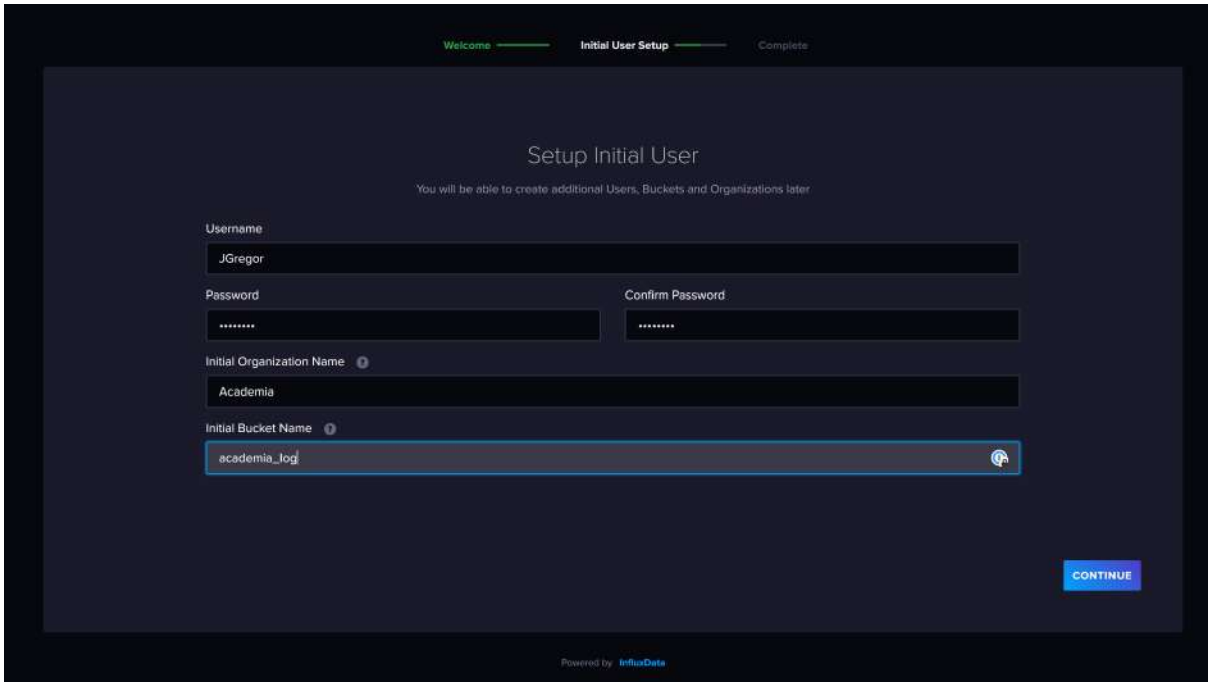
1. Najprej iz influxData spletne strani prenesemo influxdb. Ko se prenos zaključi, je potrebno ekstrahirati mapo, saj je le ta trenutno v zip formatu. Ekstrahiramo jo na poljubno mesto, kjer želimo imeti influxdb.
2. Nato zaženemo Windows PowerShell in z ukazom influxd.exe zaženemo InfluxDB.
3. V brskalniku odpremo http://localhost:8086, kjer se povežemo z InfluxDB UI in pričnemo z začetno nastavitvijo.



Slika 1: Vhodni meni InfluxDB

Vir: (lasten)

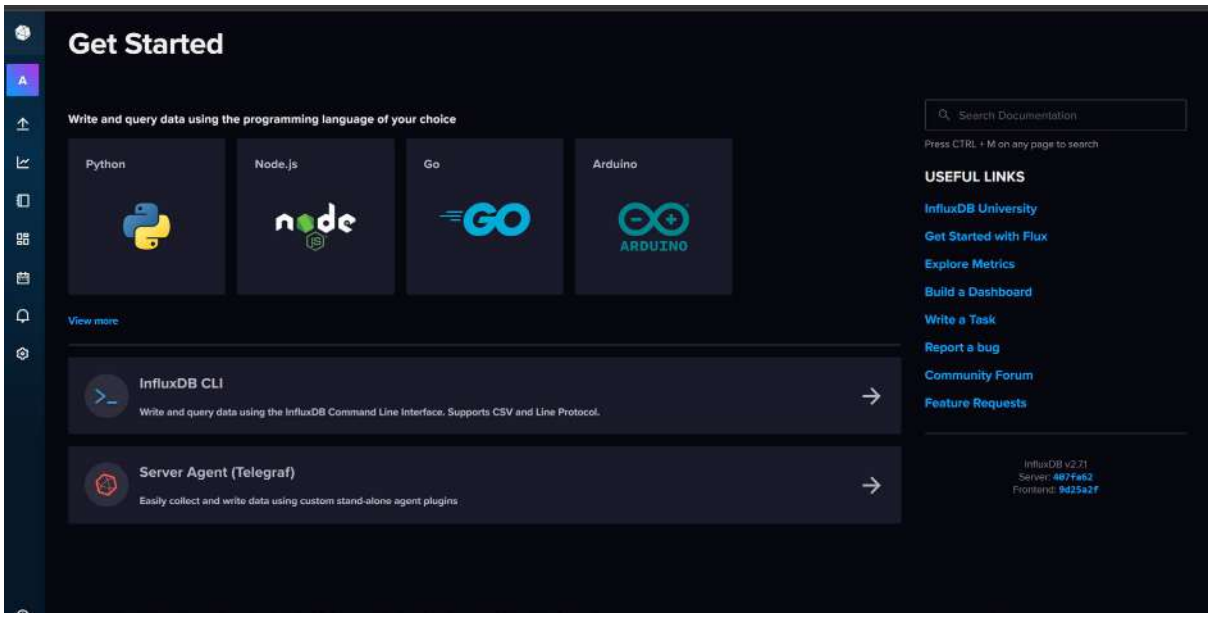
4. Nato nas InfluxDB po korakih vodi skozi osnovne nastavitve. Najprej je potrebno vnesti uporabniško ime in geslo začetnega uporabnika in nato še začetno ime organizacije in začetno ime bucket-a. Koncepta "organization" (organizacija) in "bucket" sta ključna za razumevanje kako so podatki strukturirani in dostopni. Organizacija je koncept, ki se uporablja za združevanje uporabnikov in njihovih pripadajočih virov, kot so "buckets", "dashboards", "tasks" in druge entitete. Vsak uporabnik je del ene ali več organizacij in ima določene pravice ter dovoljenja znotraj teh organizacij. Organizacije omogočajo lažje upravljanje in segregacijo podatkov in virov v večjih okoljih ali v okoljih, kjer je potrebna večja stopnja nadzora. Bucket je osnovna enota za shranjevanje podatkov v InfluxDB. Vsak bucket predstavlja lokacijo, kjer so shranjeni časovno označeni podatki. Bucket je nekako primerljiv s konceptom tabele v relacijskih podatkovnih bazah, vendar je prilagojen za potrebe časovno označenih podatkov. Vsak bucket je povezan z določeno organizacijo in ima politiko zadrževanja, ki določa, kako dolgo se podatki hranijo, preden se samodejno izbrišejo.



Slika 2: Osnovne nastavitve InfluxDB

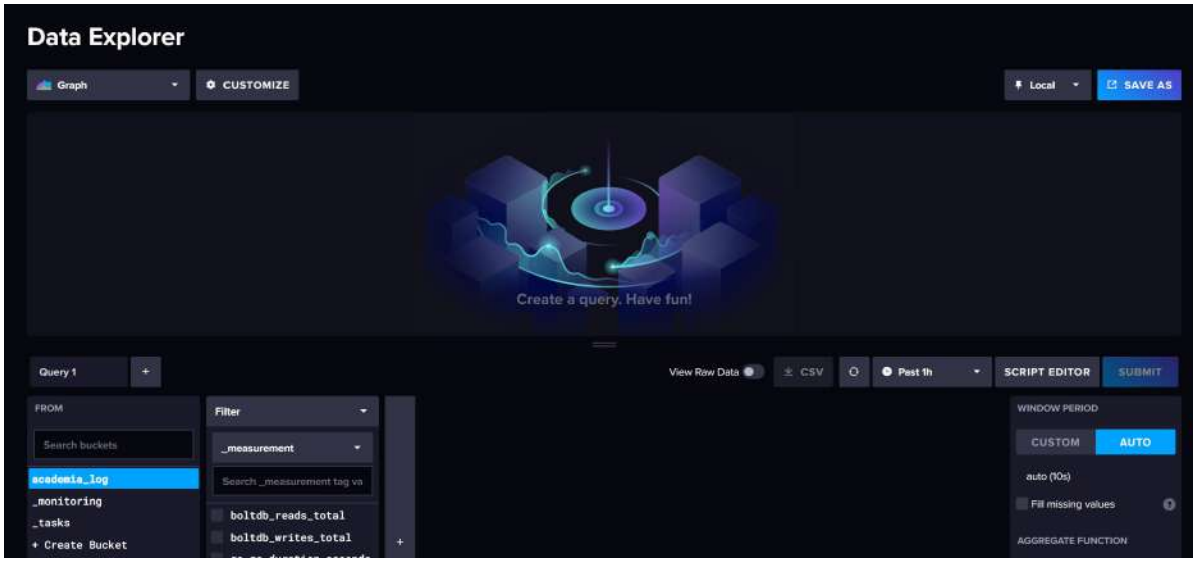
Vir: (lasten)

Po vnešenih nastavitvah lahko dostopamo do nadzorne plošče, ki je namenjena upravljanju naše podatkovne baze, ki je že pripravljena za uporabo. Za branje in pisanje podatkov lahko uporabljamo InfluxDB UI ali InfluxDB CLI, ki ga lahko uporabljamo preko ukazne vrstice z orodjem kot je Windows PowerShell.



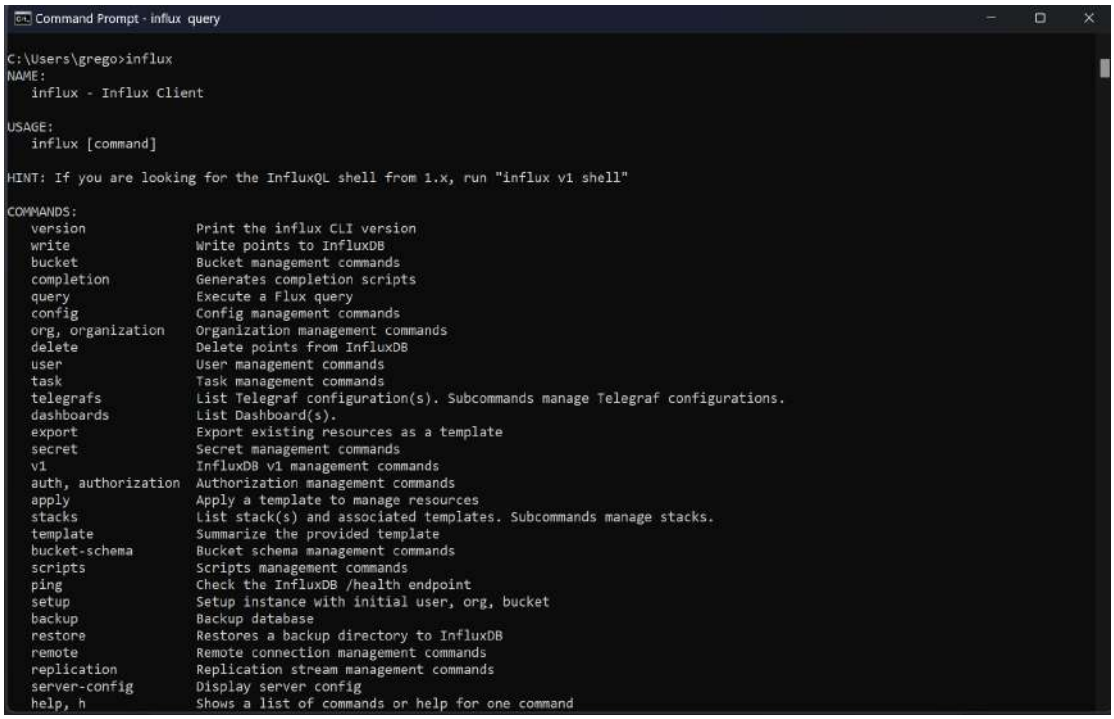
Slika 3: Osnovni meni InfluxDB

Vir: (lasten)



Slika 4: Iskanje podatkov InfluxDB

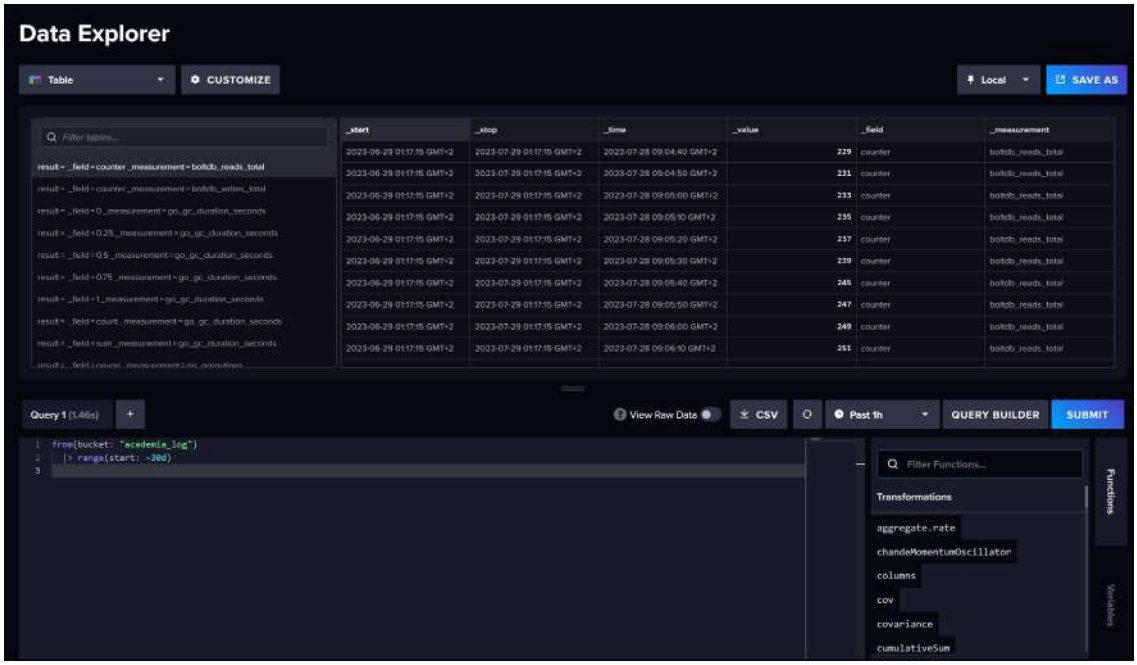
Vir: (lasten)



Slika 5: CLI InfluxDB

Vir: (lasten)

Ker ima naša podatkovna baza že naložene testne podatke, bomo poskusili izvesti poizvedbo. Uporabljali bomo InfluxDB UI, ki je uporabniku prijazen in enostaven za uporabo. Izvedli bomo poizvedbo za vse podatke znotraj academia_log za zadnjih 30 dni.



Slika 6: GUI iskanje InfluxDB

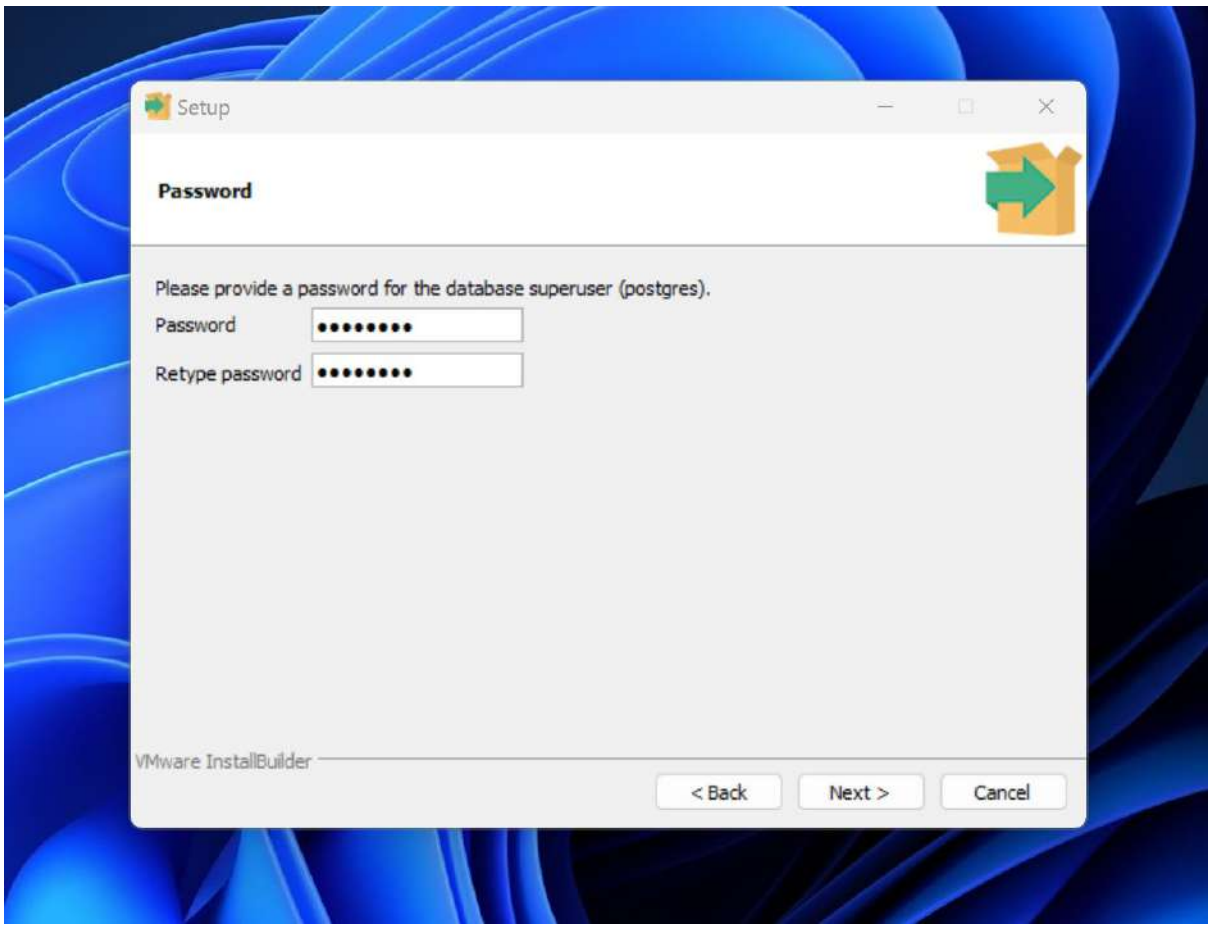
Vir: (lasten)

Znotraj InfluxDB UI imamo na voljo več formatov prikaza podatkov. Osnovni prikaz je v obliki tabele, poleg tega imamo na voljo:

- Grafikon. To je najpogostejši format za prikaz časovno označenih podatkov. Grafikon prikazuje podatke v obliki linij, stolpcev ali točk, odvisno od izbrane vrste grafikona. Grafikon je zelo prilagodljiv in omogoča spreminjanje barv, dodajanje več serij podatkov, prilagajanje osi in druge nastavitve.
- Single Stat. Ta format prikazuje eno samo statistično vrednost, kot je povprečje, vsota, minimum, maksimum ali druga agregirana vrednost. Single Stat je uporaben za prikaz ključnih kazalnikov uspešnosti ali drugih pomembnih meritev.
- Gauge prikazuje eno samo vrednost na merilniku. Gauge je uporaben za prikazovanje vrednosti, ki se spreminjajo v času in so znotraj določenega obsega.
- Heatmap. Prikazuje podatke v obliki matrike, kjer barve predstavljajo različne vrednosti. Heatmap je uporaben za prikazovanje porazdelitve vrednosti ali za prikazovanje odnosov med več spremenljivkami.
- Histogram. Prikazuje porazdelitev vrednosti v obliki stolpčnega grafikona. Histogram je uporaben za prikazovanje porazdelitve vrednosti ali za prikazovanje frekvence različnih razredov vrednosti.

4.1.2. Namestitev PostreSQL TimescaleDB

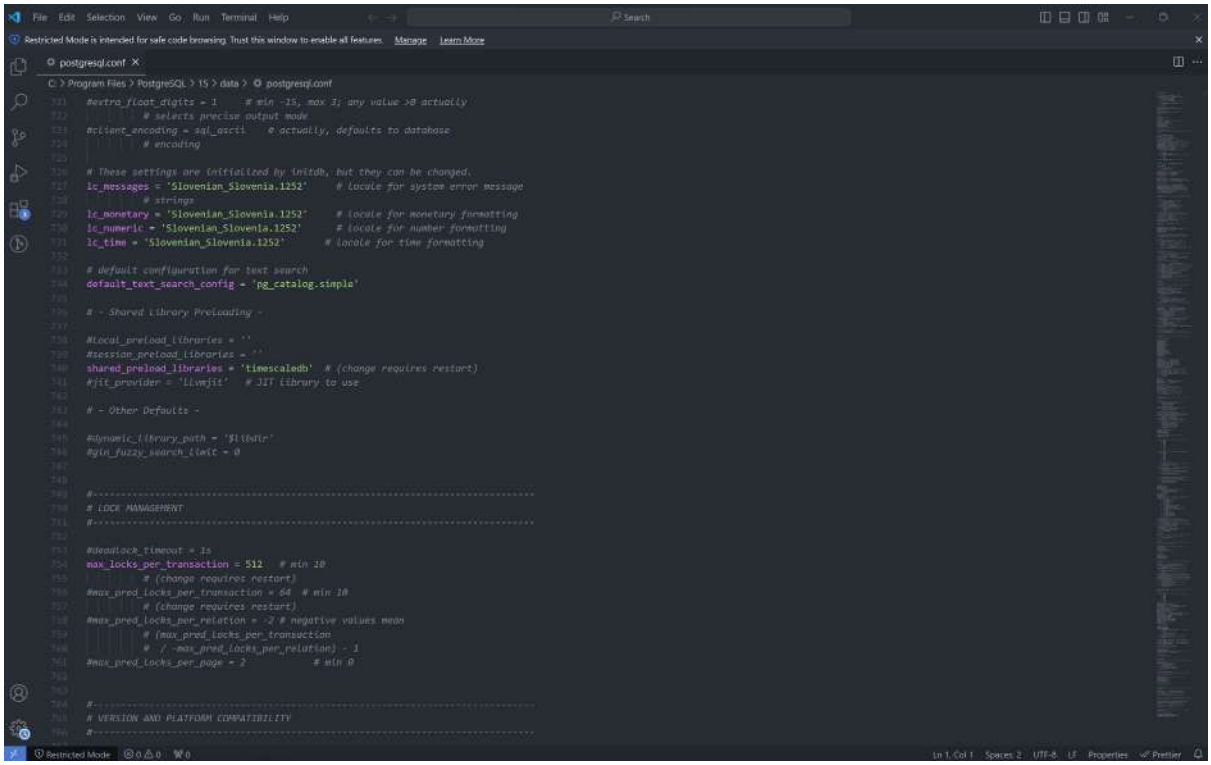
Preden namestimo TimescaleDB, je potrebno namestiti podatkovno bazo PostgreSQL, saj je TimescaleDB le podaljšek PostgreSQL. Za namestitev PostgreSQL najprej z uradne spletne strani prenesemo inštalacijski paket in zaženemo postgres.exe datoteko, ki nam odpre čarovnika za namestitev, kjer izberemo komponente, ki jih želimo namestiti. Med namestitvijo je potrebno nastaviti tudi geslo za super uporabnika >>postgres<< in si geslo zapomniti, saj ga potrebujemo za dostop do podatkovne baze.



Slika 7: Namestitev PostgreSQL

Vir: (lasten)

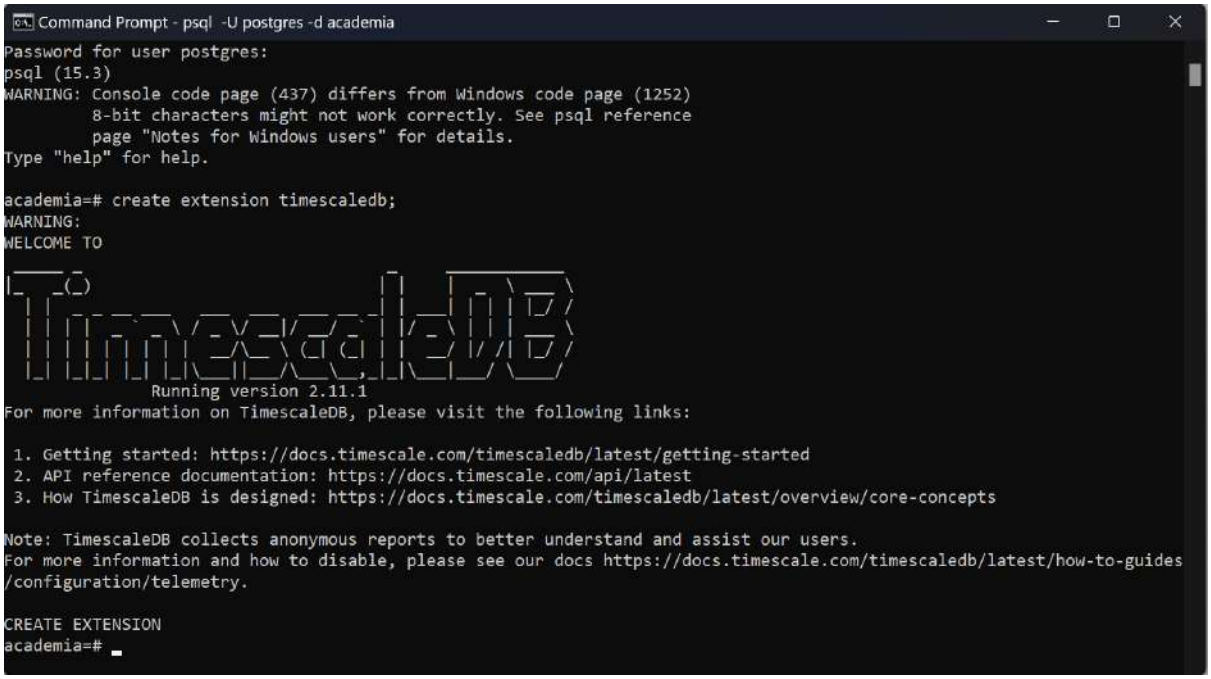
Za podatkovno bazo PostgreSQL je priporočljivo tudi branje dokumentacije o konfiguraciji podatkovne baze, saj PostgreSQL dopušča ogromno konfiguracije po meri.



Slika 8: Skripta za urejanje konfiguracije PostgreSQL

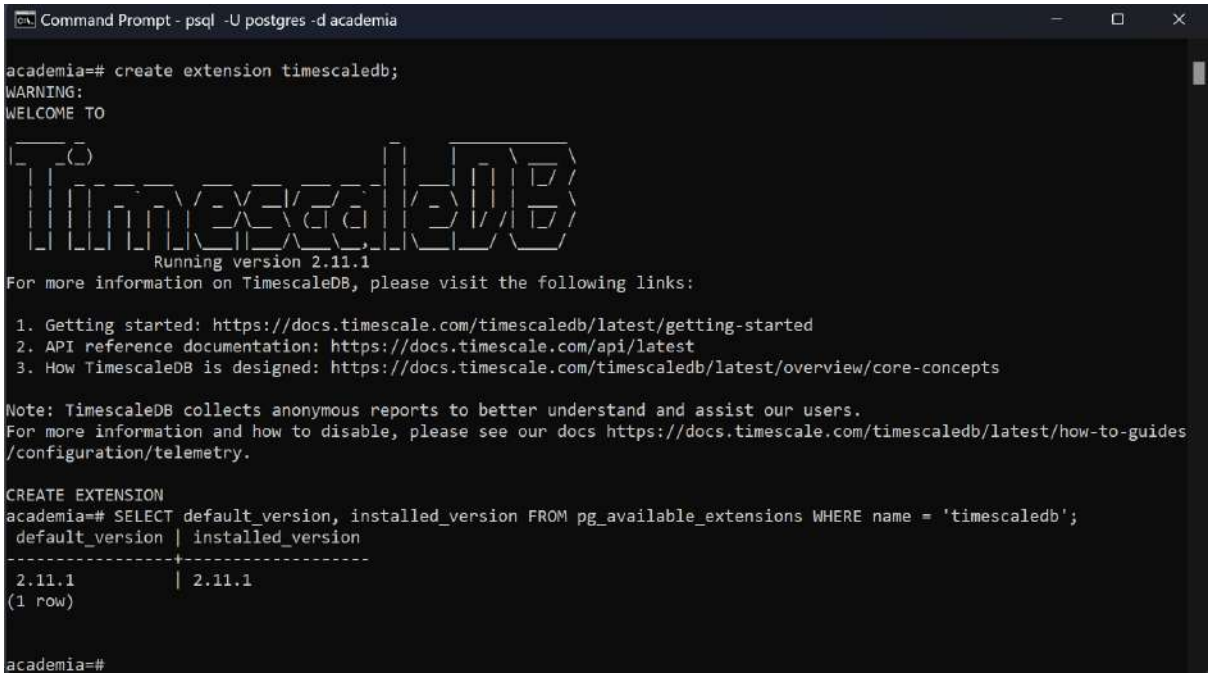
Vir: (lasten)

Ko imamo nameščeno podatkovno bazo PostgreSQL, je potrebno namestiti še podaljšek TimescaleDB. Le tega najprej prenesemo z uradne spletne strani, kjer izberemo našo verzijo podatkovne baze. Ko ga lokalno prenesemo na računalnik, je potrebno najprej zagnati Setup.exe datoteko, s katero bomo TimescaleDB podaljšek namestili in nato Timescaledb-tune.exe datoteko, ki nam bo pripravila konfiguracijo za uporabo z našo podatkovno bazo. Ko izvedemo te korake, se lahko prijavimo v bazo in namestimo podaljšek na želeno podatkovno bazo, v našem primeru bo to podatkovna baza academia.



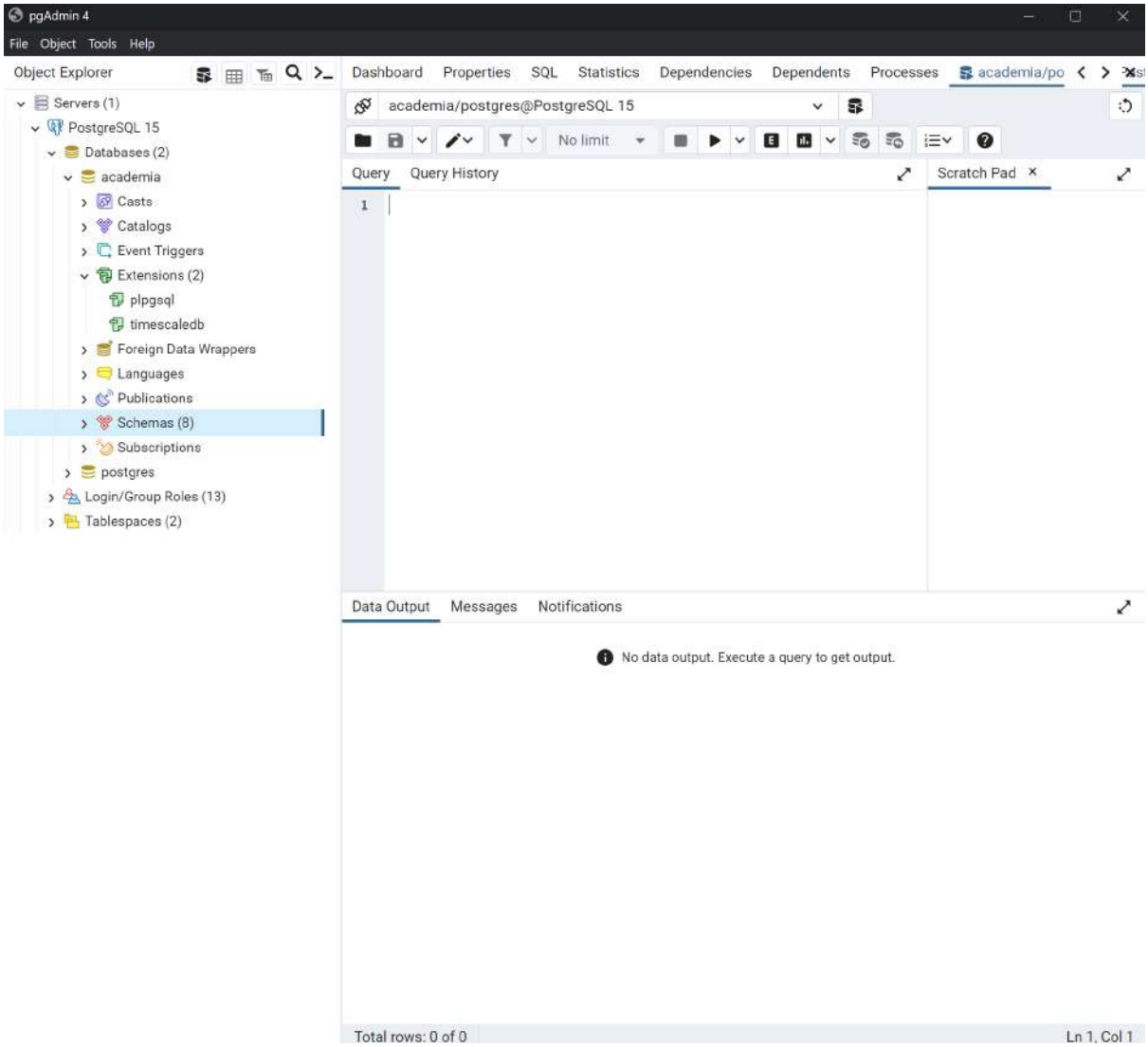
Slika 9: Namestitev TimescaleDB

Vir: (lasten)



Slika 10: Prijava v TimescaleDB

Vir: (lasten)



Slika 11: PostgreSQL GUI

Vir: (lasten)

Na uradni spletni strani TimescaleDB najdemo tudi vso dokumentacijo, povezavo do uradnega foruma, njihovega vodiča za uporabo, Slack kanala in bloga.

4.1.3. Ugotovitve

InfluxDB je namensko zgrajena za časovne vrste in nudi ločen ekosistem orodij za delo s podatki časovnih vrst. Med namestitvijo smo opazili, da InfluxDB praktično nudi storitev "plug-and-play", kar pomeni, da lahko izvedemo namestitev in vzpostavitev okolja brez veliko konfiguracije in lahko hitro začnemo z zbiranjem in poizvedovanjem podatkov.

TimescaleDB je razširitev za PostgreSQL in je zasnovana tako, da izkoristi prednosti relacijskega sistema za upravljanje z bazami podatkov (RDBMS), medtem ko dodaja funkcionalnost za časovne serije. Postopek namestitve je bil precej preprost, saj smo poleg namestitve PostgreSQL morali samo še dodati razširitev TimescaleDB. Ker je TimescaleDB integrirana v PostgreSQL, smo lahko izkoristili vse napredne funkcionalnosti in orodja, ki jih ponuja PostgreSQL.

Sama namestitev InfluxDB je bila sicer lažja, vendar je potrebno izpostaviti, da lahko obe podatkovni bazi namestimo in vzpostavimo v zelo kratkem času brez večjih težav.

4.2. Podpora

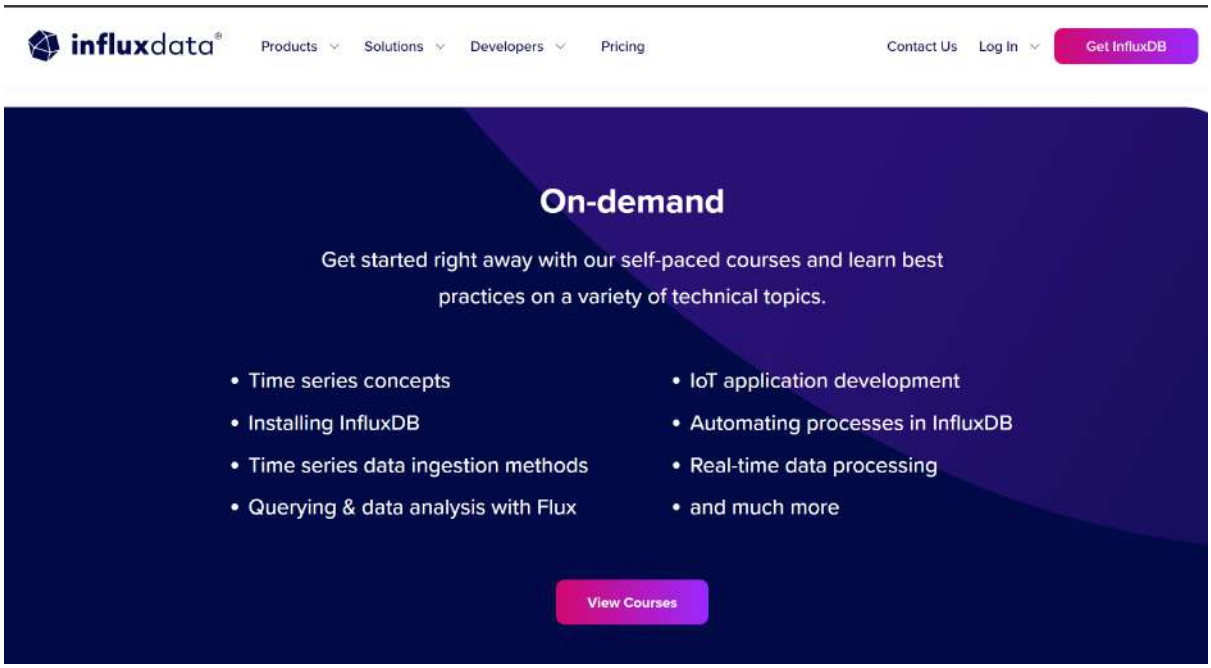
4.2.1. Podpora InfluxDB

Kot smo že omenili v prejšnjem poglavju lahko že ob odprtju uradne spletne strani opazimo zavihek >>Community<<, kjer najdemo povezavo do uradnega Slack kanala, Influx univerze, dokumentacije in njihovega foruma.

Dokumentacija, ki se nahaja na strani je zelo obsežna in pokriva vse aspekte InfluxDB in vse kar sodi zraven. Slack kanal in forum sta namenjena neposredni izmenjavi znanja, podpore in idej med člani skupnosti. Kanal in forum sta razdeljena na specifične teme, kot so osnovna uporaba, konfiguracija, razvoj API-jev, integracije, napovedi izdaj in še več. To članom omogoča, da se pridružijo razpravam, ki so najbolj relevantne za njihove interese in potrebe. Hkrati Slack kanal pogosto gosti strokovnjake iz področja InfluxDB, vključno z razvijalci, tehničnimi svetovalci in celo člani vodstvenega tima InfluxData. To uporabnikom omogoča, da postavljajo vprašanja in prejemajo nasvete neposredno od ljudi, ki so globoko vključeni v razvoj in vzdrževanje teh tehnologij.

Influx univerza je izobraževalna platforma, ki jo je ustanovilo podjetje InfluxData. Namen te platforme je zagotoviti strukturirano, modulno in interaktivno učno izkušnjo za vsakogar, ki želi postati vešč v uporabi InfluxDB in sorodnih tehnologij. Udeleženci lahko dostopajo do širokega nabora tečajev in učnih materialov, ki pokrivajo vse od osnovnih principov delovanja InfluxDB, do naprednih tehnik za poizvedovanje, analizo in upravljanje podatkov. Tečaji so zasnovani tako, da postopoma vodijo učence skozi različne funkcije in zmogljivosti sistema,

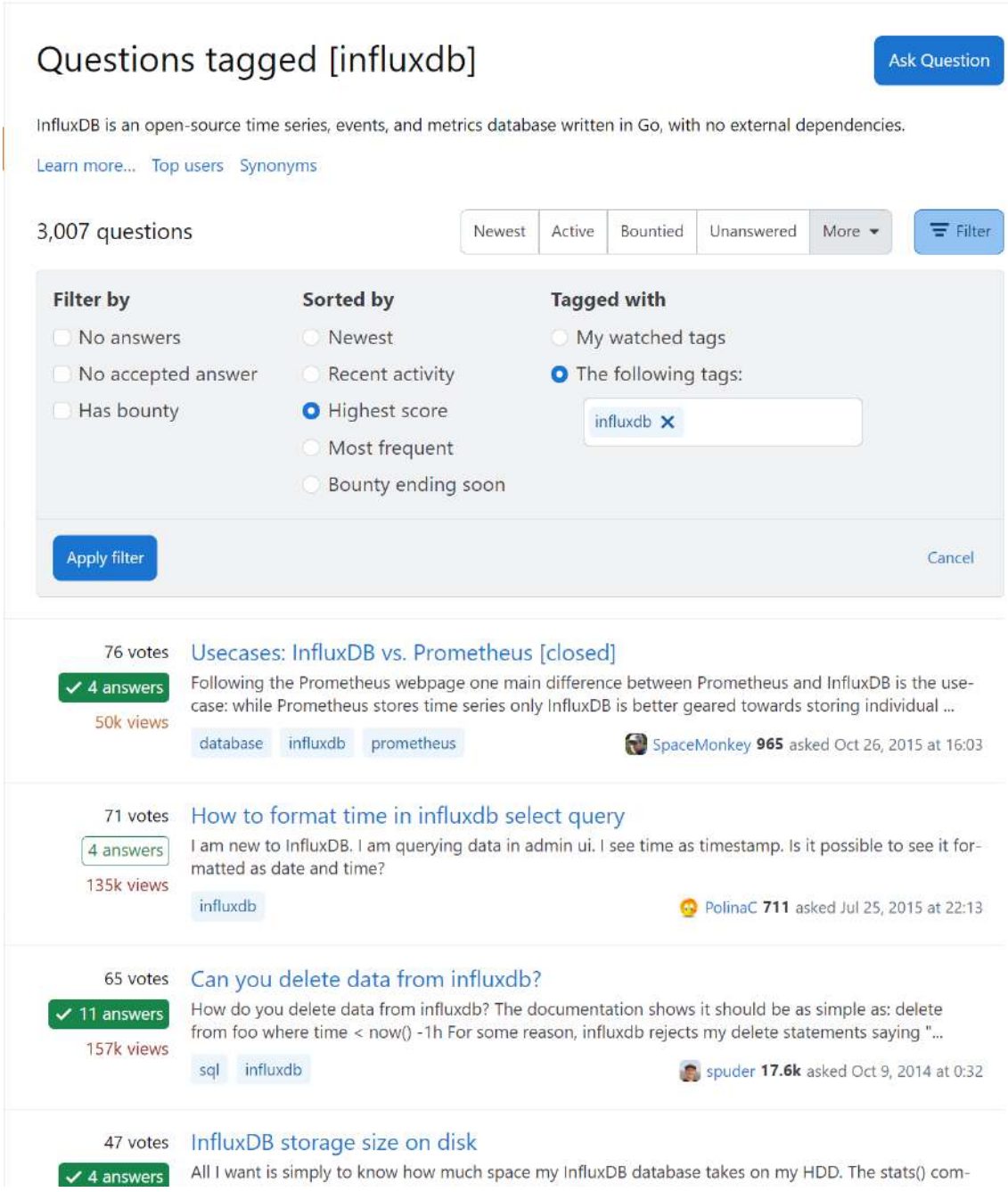
pri čemer poudarjajo praktične scenarije uporabe. Tečaji so brezplačni hkrati pa ponujajo tudi seminarje, ki se odvijajo v živo in jih vodijo njihovi strokovnjaki.



Slika 12: Influx univerza

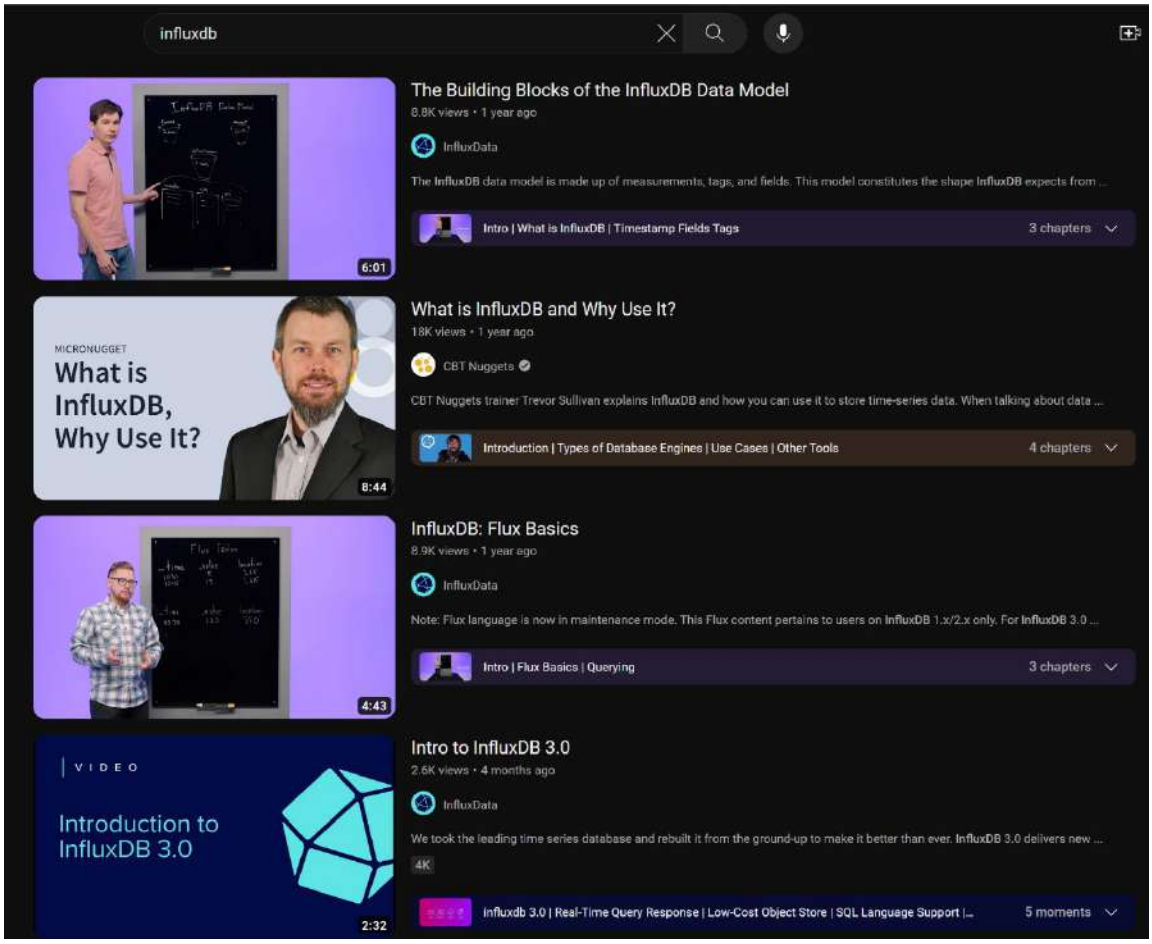
Vir: (lasten)

Podpora za InfluxDB pa ni omejena samo na uradno spletno mesto. Če pogledamo še izven okolja InfluxDB lahko vidimo, da ima podporo tudi na večih znanih platformah, kot sta Stack Overflow in Youtube. Na tehničnem forumu Stack Overflow lahko vidimo, da je bilo odprtih že več kot 3000 tem s ključno besedo >>influxdb<<. Tudi Youtube nam ponuja ogromno vsebin na temo InfluxDB.



Slika 13: InfluxDB Stack Overflow

Vir: (lasten)



Slika 14: InfluxDB Youtube

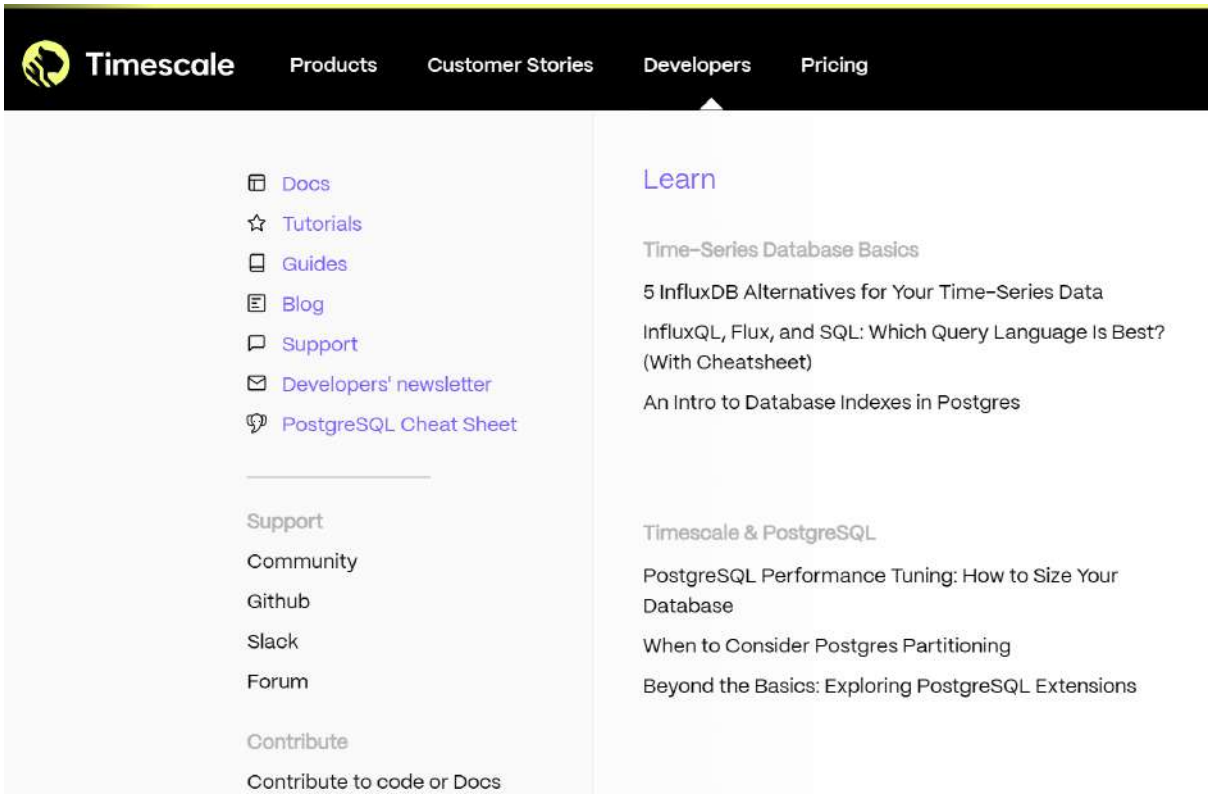
Vir: (lasten)

Skupaj gledano InfluxDB uživa v podpori živahne skupnosti, bogati dokumentaciji in obsežnih možnostih učenja. Ta trdna infrastruktura podpore, ne le olajša implementacijo in uporabo te tehnologije, ampak tudi spodbuja inovacije in sodelovanje, kar vodi k nenehnim izboljšavam in razširitvam platforme. Tako novinci kot izkušeni strokovnjaki lahko najdejo potrebne vire za učinkovito delo z InfluxDB, kar zagotavlja, da ostaja ena najbolj priljubljenih in dobro podprtih podatkovnih baz za upravljanje časovnih vrst.

4.2.2. Podpora TimescaleDB

TimescaleDB se ponaša z močno in aktivno skupnostjo, ki obsega razvijalce, administratorje podatkovnih baz, analitike in poslovne uporabnike. Ta skupnost je vitalen vir informacij in nasvetov za delo s TimescaleDB. Na voljo imamo uraden forum, Slack kanal in GitHub projekt. Skupinska spletna mesta so polna razprav, primerov uporabe in rešitev za izzive, povezane s to

podatkovno bazo. Poleg tega Timescale redno organizira in sodeluje na dogodkih, kot so srečanja skupnosti, webinarji in konference, kjer lahko uporabniki neposredno komunicirajo z ekipo Timescale. Na uradni spletni strani projekta imamo na voljo bogato dokumentacijo. Dokumentacija obsega vse od osnovnega uvoda v TimescaleDB, vodnikov za začetnike, do naprednih tem, kot so optimizacija zmogljivosti, replikacija in varnost. Dokumentacija je jasna, primeri so relevantni, kar uporabnikom omogoča hitro razumevanje in učinkovito uporabo tehnologije. Poleg klasične dokumentacije Timescale ponuja različne izobraževalne vire, vključno z video vsebinami, spletnimi seminarji, blogi in študijami primerov. Te vsebine so namenjene uporabnikom različnih ravni strokovnosti, od začetnikov do strokovnjakov. Za tiste, ki želijo poglobiti svoje znanje in veščine, so na voljo tudi specializirani tečaji in usposabljanja, ki se osredotočajo na specifične funkcionalnosti ali scenarije uporabe TimescaleDB.

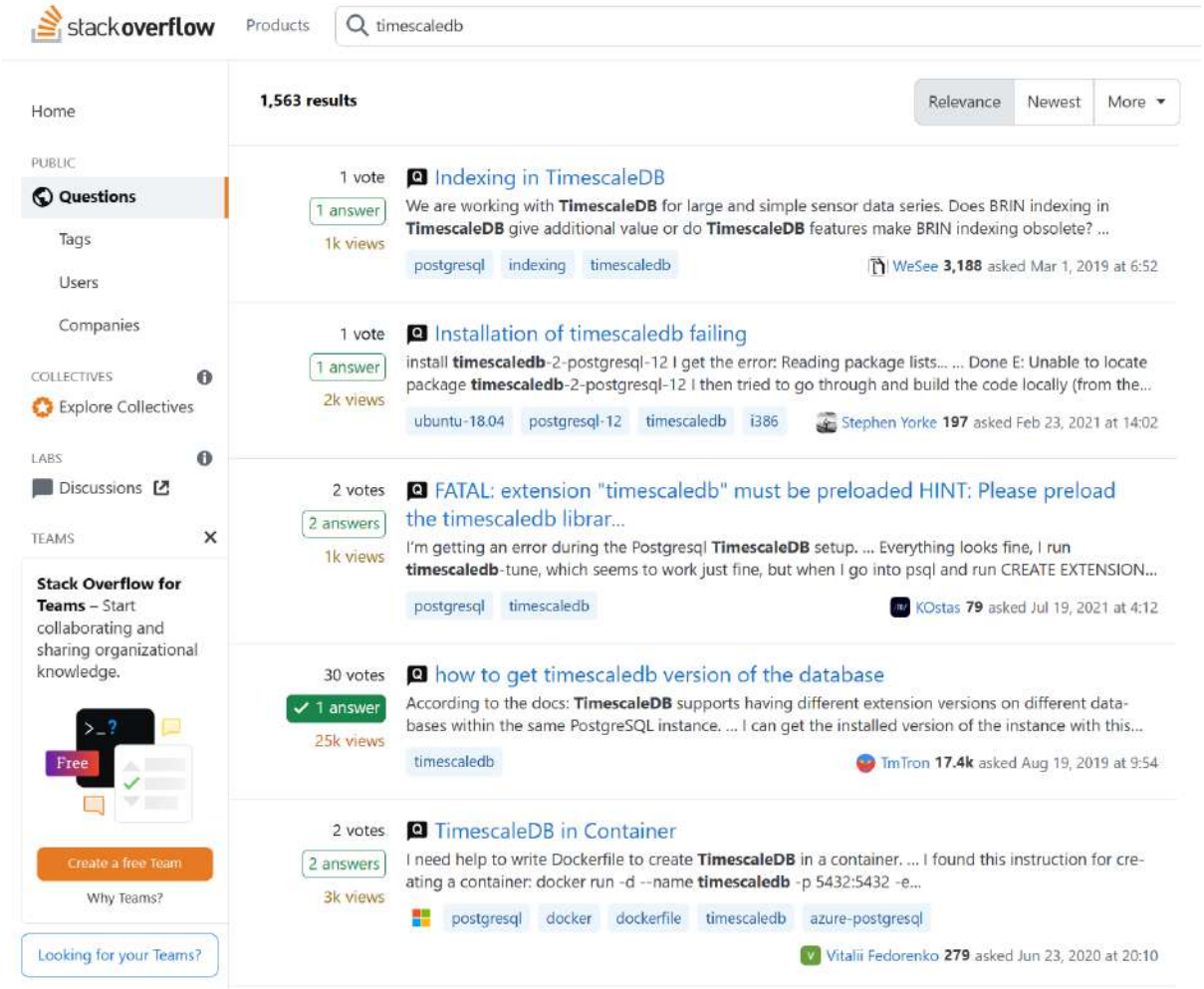


Slika 15: Uradna spletna stran Timescale

Vir: (lasten)

TimescaleDB skupnost ni le omejena na uradna spletna mesta in forumsko podporo. Dejavnost okoli te podatkovne baze je mogoče najti na različnih platformah po spletu. Za ilustracijo tega lahko pogledamo na priljubljen tehnični forum, Stack Overflow. Na spodnji sliki lahko vidimo številna vprašanja, odgovore in razprave povezane s TimescaleDB. Ta aktivnost dokazuje, da

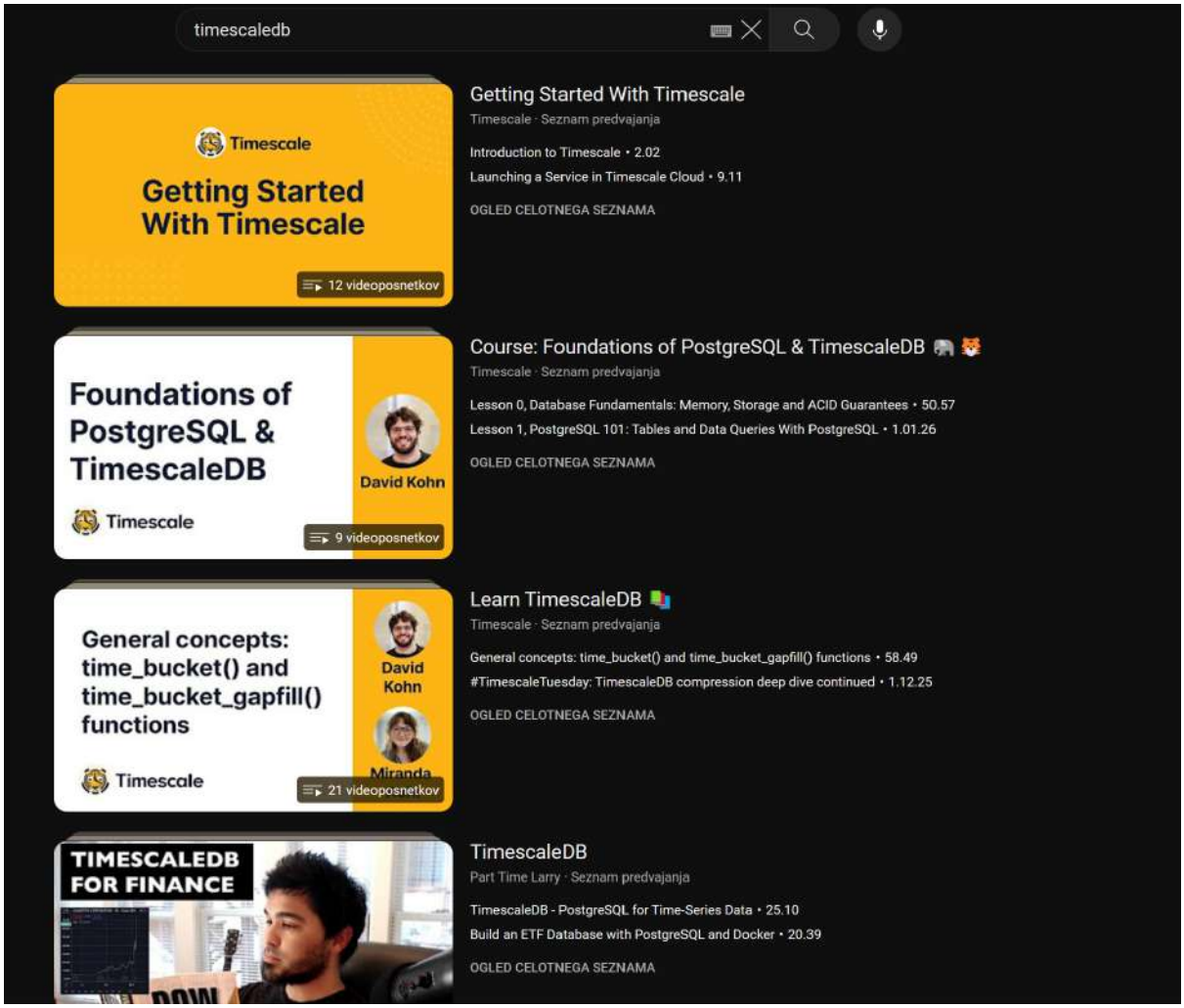
obstaja močna in angažirana skupnost strokovnjakov, ki so pripravljeni pomagati in deliti svoje znanje z drugimi.



Slika 16: TimescaleDB Stack Overflow

Vir: (lasten)

Prav tako lahko najdemo podporo za TimescaleDB na platformah, kot je YouTube. Iskanje po ključni besedi "TimescaleDB" razkrije številne videoposnetke in predstavitve, ki jih je skupnost ustvarila za pomoč novincem in naprednim uporabnikom.



Slika 17: TimescaleDB Youtube

Vir: (lasten)

Skupno gledano je TimescaleDB podprt s trdnimi stebri podporne strukture, ki zagotavljajo, da so uporabniki opremljeni z znanjem in veščinami, potrebnimi za uspeh. Aktivna skupnost, obsežna dokumentacija in širok nabor izobraževalnih virov zagotavljajo, da so uporabniki TimescaleDB vedno dobro informirani in podprti pri svojih projektih. To ne le izboljšuje uporabniško izkušnjo, ampak tudi spodbuja inovacije in širjenje te tehnologije v različnih industrijah in aplikacijah.

4.3. Priprava podatkov

Kot smo že omenili v prejšnjih poglavjih, smo obe podatkovni bazi namestili lokalno, da lahko zagotovimo iste pogoje in sistemsko opremo za testiranje. Po vzpostavitvi okolja sledi priprava podatkov. Da bomo lahko izvajali teste, potrebujemo pripravljene podatkovne strukture in veliko količino podatkov.

4.3.1. Podatkovne strukture

Podatkovne strukture bomo pripravili na podlagi finančnih podatkov, bolj specifično na gibanju cen delniških trgov. Za finančne podatke smo se odločili, ker so po naravi časovno zaporedni podatki, saj se nenehno spreminjajo in vsaka cena je povezana s specifičnim časovnim žigom, kot je datum in čas. Podatkovne baze časovnih vrst so idealne za analizo trendov, saj so zasnovane posebej za delo s takšnimi podatki. Trgovalci in analitiki pogosto uporabljajo te baze za iskanje vzorcev v cenah delnic skozi čas, da bi napovedali prihodnje gibanje cen. Še več, podatkovne baze časovnih vrst so optimizirane za obvladovanje podatkov z visoko frekvenco, kar je še posebej pomembno v segmentih kot je visokofrekvenčno trgovanje, kjer se cene delnic lahko spreminjajo večkrat na sekundo. Hitrost je v takšnih situacijah ključna in te baze so zasnovane tako, da omogočajo hitro zapisovanje in poizvedovanje podatkov, kar zmanjšuje zaostanek. Poleg tega lahko borzni trgi generirajo obsežne količine podatkov v kratkem času, zato baze časovnih vrst ponujajo učinkovite mehanizme kompresije za zmanjšanje potrebne shrambe. V finančnem svetu so natančnost, razpoložljivost in integriteta podatkov ključnega pomena. Majhne napake ali zamude lahko povzročijo velike izgube ali kazni, zato je preizkušanje teh vidikov ključnega pomena za zagotavljanje, da sistem izpolnjuje te stroge zahteve. Finančni podatki tudi zahtevajo različne vrste analiz, od preprostih statističnih izračunov do naprednih algoritmov za strojno učenje. To vključuje različne analitične pristope, kot so trendna analiza, regresijska analiza in simulacije tveganja, ki lahko zahtevajo veliko računalniške moči in spomina. Testiranje teh vidikov nam omogoča oceno ali so naši sistemi kos tem zahtevam. Kompleksnost in večplastnost teh podatkov zagotavljata bogat in zahteven nabor testnih primerov, ki odražajo resnične izzive.

V podatkovni bazi InfluxDB ni potrebe po predhodnem definiranju meritev ali stolpcev, saj jih InfluxDB dinamično določi, ko se podatki vstavljajo, torej nam v InfluxDB ni treba v naprej pripravljati podatkovnih struktur. Ustvarjanje shem vključuje vstavljanje podatkov s pravilno strukturo oznak in polj.

Ker je TimescaleDB relacijska baza, bo za to podatkovno bazo potrebno predhodno pripraviti podatkovne strukture. Pripravili smo sledeče podatkovne strukture:

- **Tabela exchanges**: Shranjuje informacije o borznih trgih, vključno z ID borze, imenom, lokacijo in kodo valute. Primarni ključ je exchange_id.

- **Tabela stocks**: Vsebuje informacije o posameznih delnicah, vključno z ID delnice, ID borze, simbolom in imenom podjetja. Primarni ključ je stock_id, in tuj ključ exchange_i, ki se sklicuje na tabelo exchanges.
- **Tabela prices**: Vsebuje točkovne cenovne podatke za delnice, vključno s časovnim žigom, ID delnice in ceno. Primarni ključ je sestavljen iz timestamp in stock_id. Stolpec stock_id je tuj ključ, ki se sklicuje na tabelo stocks. Ta tabela je hypertable za učinkovito obdelavo časovnih vrst in jo bomo uporabljali za izvajanje naših testov.

Za tabele exchanges in stocks smo pripravili skripto za generiranje naključnih podatkov, ki jih bomo uporabili v samem testiranju, kjer bomo izvajali teste na tabeli prices.

4.3.2. Orodje za testiranje

Apache JMeter je odprtokodno orodje za testiranje zmogljivosti, ki ga je razvila Apache Software Foundation. Uporablja se za simuliranje večkratnih in sočasnih uporabnikov, za obremenitveno testiranje različnih objektov in scenarijev, aplikacij, omrežnih storitev in protokolov (Apache JMeter, n.d.).

Za Apache JMeter smo se odločili, saj nam omogoča široko paleto testiranj, ima grafični vmesnik, ki omogoča enostavno izdelavo in spreminjanje testnih načrtov, kar nam lahko olajša izdelavo kompleksnih testov. Hkrati nam JMeter omogoča podrobno poročanje v realnem času z grafi in grafikoni, ki nam pomagajo razumeti, kako se sistem obnaša pod obremenitvijo. To je ključnega pomena pri analizi zmogljivosti. Ima možnost simulacije na tisoče uporabnikov, kar je ključnega pomena pri testiranju zmogljivosti in nam omogoča realistično testiranje scenarijev Big Data. JMeter je tudi stroškovno učinkovita izbira, saj je odprtokodno orodje, torej ni nobenih licenčnih stroškov.

4.4. Primerjava zmogljivosti pri uporabi Big Data

Testiranje Big Data se osredotoča na preverjanje kakovosti in zmogljivosti velikih količin podatkov v realnem času.

4.4.1. Pisanje

Najprej si z uporabo procedur iz skripte, ki smo si jo pripravili, zgeneriramo podatke za tabeli exchanges in stocks. V skripti smo za tabeli exchanges in stocks pripravili proceduro, ki kot vhodni parameter sprejme število in glede na ta parameter pripravi takšno število zapisov v tabeli. Dodatno smo pripravili tudi proceduro, ki nam pobriše vse podatke iz teh tabel. Ker InfluxDB dinamično določi podatkovne strukture, ni potrebe po vnaprejšnjem definiranju le teh.

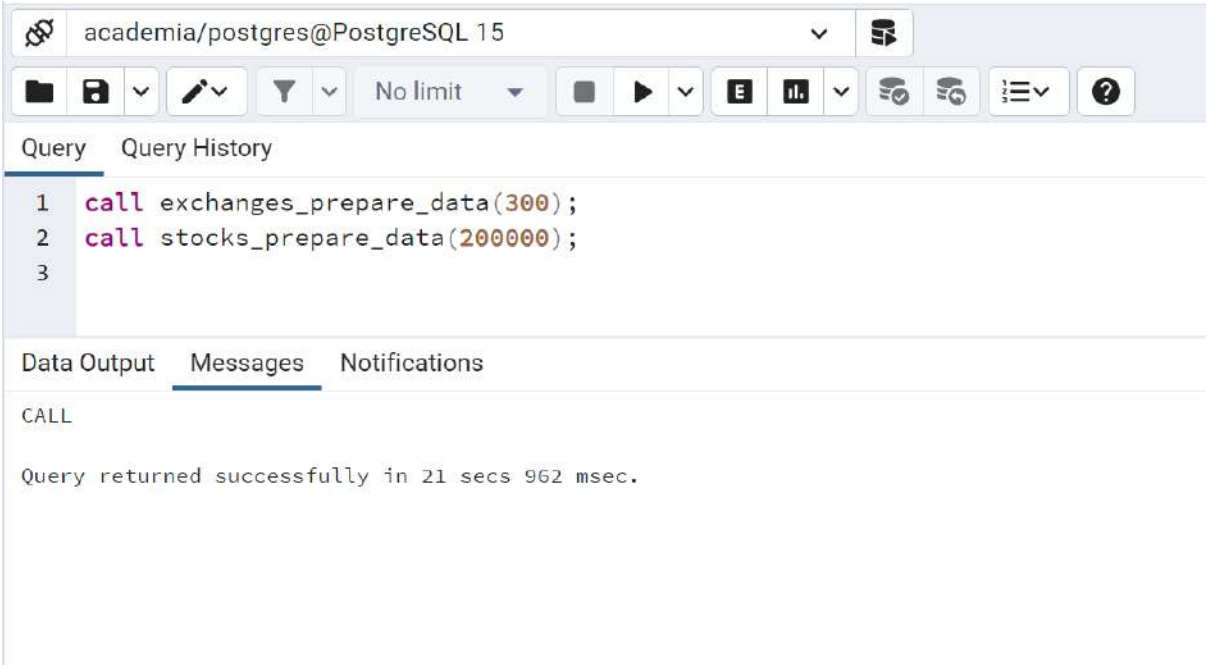
Za generiranje podatkov bomo izvedli klice teh procedur:

call exchanges_prepare_data(300);

call stocks_prepare_data(200000).

Ti dve proceduri nam bosta zgenerirali 300 zapisov za tabelo exchanges in 200000 zapisov za tabelo stocks.

Proceduri sta za izvedbo, torej generiranje 200300 zapisov, potrebovali 21 sekund.



Slika 18: Priprava podatkov TimescaleDB

Vir: (lasten)

Za testiranje zmogljivosti pri uporabi Big Data smo uporabili tabelo prices v TimescaleDB in meritev prices v InfluxDB. Za samo izvedbo testiranja smo uporabili prej opisano programsko opremo JMeter.

Za polnjenje podatkov v podatkovno bazo TimescaleDB smo si pripravili proceduro new_price, ki nam izvede zapis v tabelo prices, za zapisovanje v podatkovno bazo InfluxDB bomo uporabili New line protocol.

V programu Jmeter smo si pripravili dva testa, kjer smo najprej konfigurirali povezavo z našo podatkovno bazo in nato dodali naš testni primer. S testom vstavljamo zapise v tabelo prices in meritev prices. Dodali smo konfiguracijo, da tale klic izvaja 500 uporabnikov hkrati v sekundnih intervalih. Skupno smo na bazo poslali 1.000.000 zahtevkov. Na koncu smo še dodali, katera poročila želimo, da nam Jmeter pripravi.

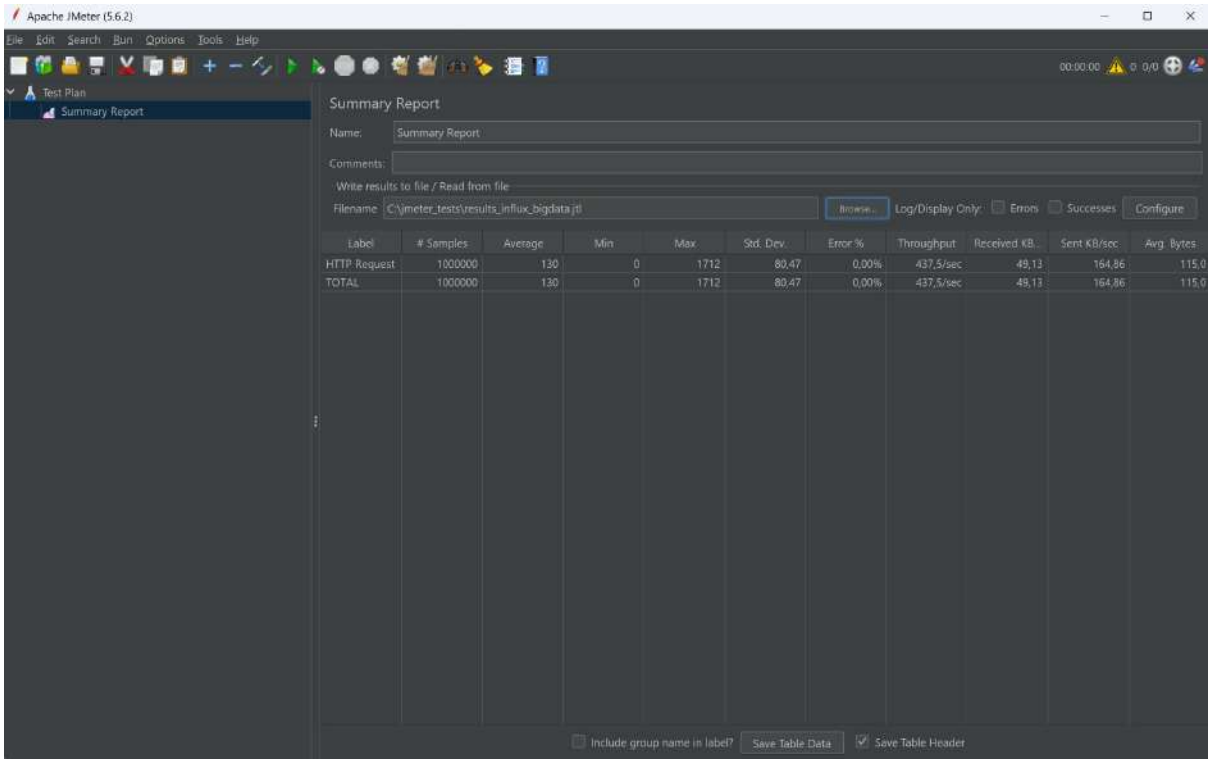
Poročilo, ki nam ga pripravi Jmeter, obsega naslednje podatke:

- **Label**: Ime vzorca. Ta stolpec vsebuje ime HTTP zahteve ali ime vzorca, ki ga določite v JMeter.
- **# Samples**: Skupno število zahtev, ki so bile poslane na strežnik.
- **Average**: Povprečni čas odziva v milisekundah.
- **Min**: Najkrajši čas odziva v milisekundah.
- **Max**: Najdaljši čas odziva v milisekundah.
- **Std. Dev**.: Standardna deviacija časa odziva. To je merilo variabilnosti časov odziva. Večja kot je standardna deviacija, bolj razpršeni so časi odziva.
- **Error %**: Procent neuspešnih zahtev.
- **Throughput**: Hitrost obdelave, izražena kot število zahtev na sekundo, ki jih je JMeter uspel poslati strežniku.
- **Received KB/sec**: Prejeti KB na sekundo.
- **Sent KB/sec**: Poslani KB na sekundo.
- **Average Bytes**: Povprečna količina bajtov, prejetih po vsaki zahtevi.

Tabela 1: Test zapisovanja podatkov

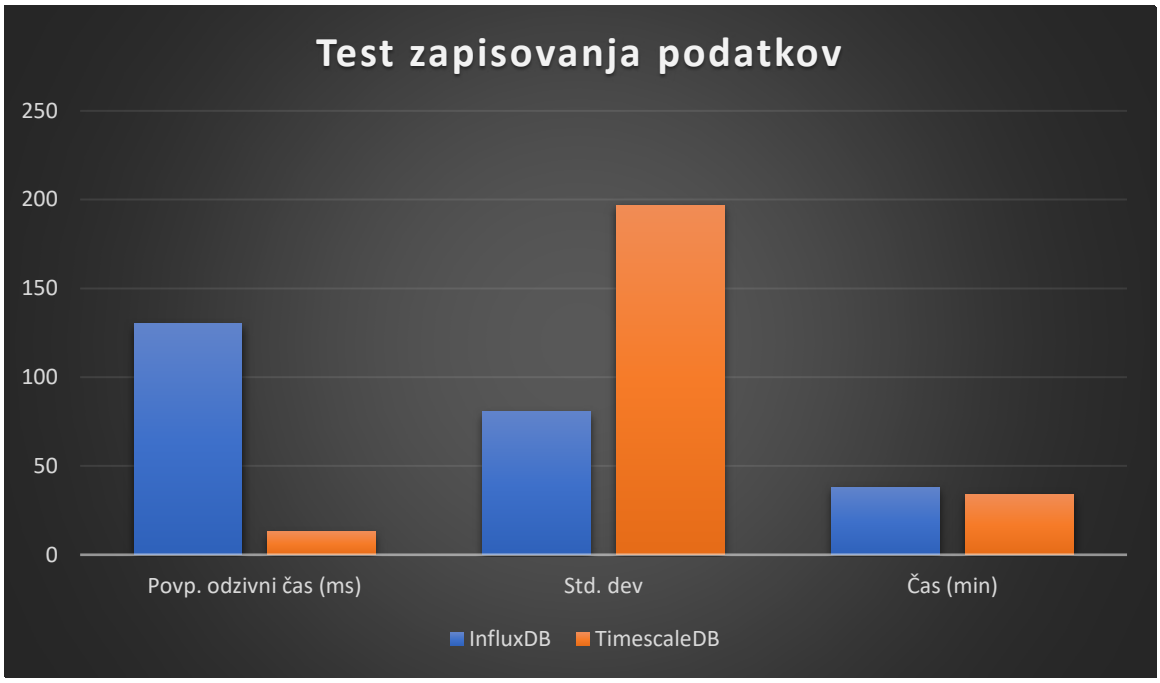
Podatkovna baza	Število uporabnikov	Povp. odzivni čas	% napak	Std. Dev.	CPU	Memory	Čas
TimescaleDB	500	13ms	0,72%	196,98	18%	74%	34:17
InfluxDB	500	130ms	0,00%	80,47	21%	68%	38:07

Vir: (lasten)



Slika 19: JMeter test zapisovanja podatkov InfluxDB

Vir: (lasten)



Graf 1: Test zapisovanja podatkov

Vir: (lasten)

4.4.2. Branje

Naše strukture trenutno vsebujejo približno 2 milijona zapisov, da lahko učinkovito testiramo branje podatkov. Za testiranje branja podatkov smo uporabili različne poizvedbe.

Uporabili smo sledeče poizvedbe:

1. Preprosto poizvedbo.
2. Poizvedbo z razponom časa.
3. Poizvedbo, ki uporablja indeks.
4. Poizvedbo brez uporabe indeksa.
5. Poizvedbo, ki uporablja agregatne funkcije.
6. Poizvedbo, ki uporablja podpoizvedbo.

Tabela 2: Test branja podatkov TimescaleDB

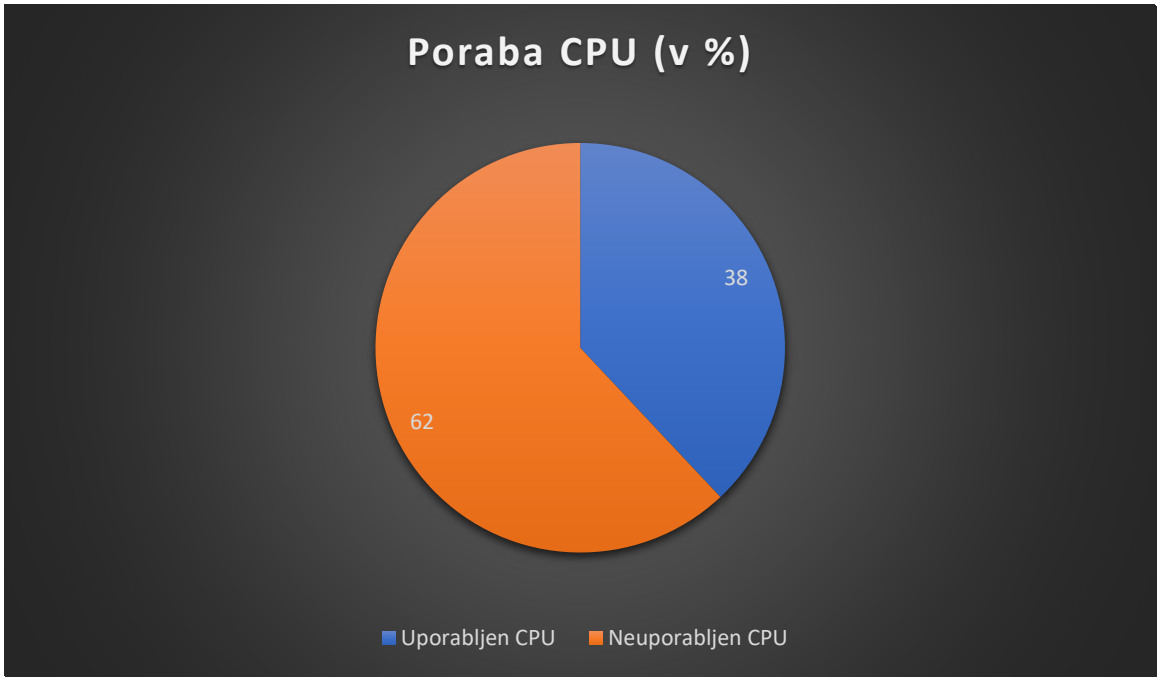
Št. poizvedbe	Povp. odzivni čas (ms)	% napak	Std. Dev.	CPU	Memory	Št. vrnjenih vrstic
1	3172	0,00%	964,54	38%	61%	1.967.262
2	1305	0,00%	611,42	38%	61%	799.189
3	457	0,00%	205,52	38%	61%	20
4	2375	0,00%	850,14	38%	61%	809.478
5	490	0,00%	181,45	38%	61%	1
6	314	0,00%	173,67	38%	61%	1

Vir: (Lasten)

Tabela 3: Test branja podatkov InfluxDB

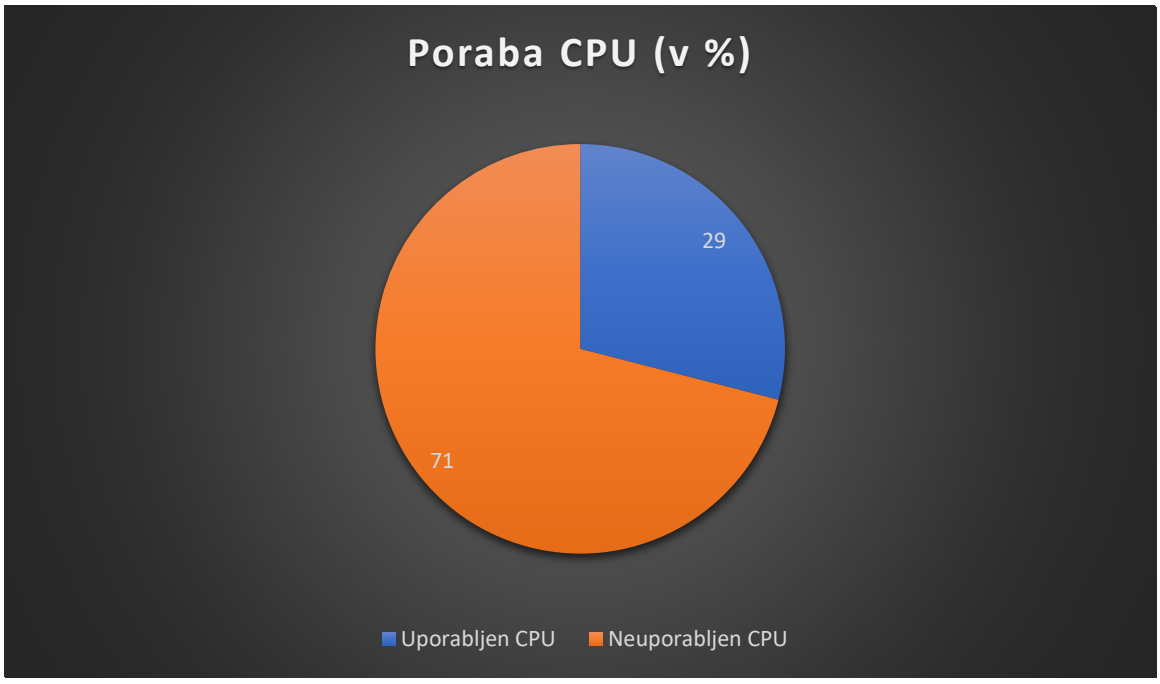
Št. poizvedbe	Povp. odzivni čas (ms)	% napak	Std. Dev.	CPU	Memory	Št. vrnjenih vrstic
1.	237150	0,00%	81459.28	29%	72%	1.782.798
2.	57938	0,00%	20369,45	29%	72%	819.372
3.	83	0,00%	23,47	29%	72%	20
4.	231848	0,00%	28496,27	29%	72%	890.934
5.	24	0,00%	14,56	29%	72%	1
6.	42	0,00%	7,11	29%	72%	1

Vir: (Lasten)



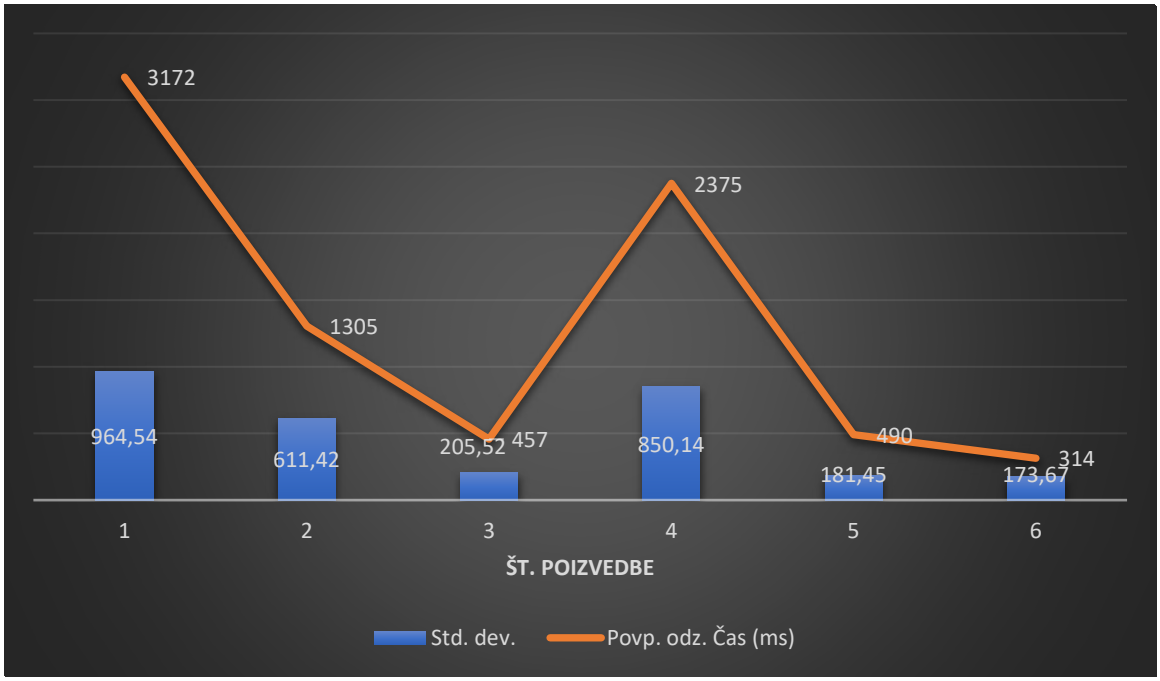
Graf 2: Poraba CPU ob branju podatkov TimescaleDB

Vir: (lasten)



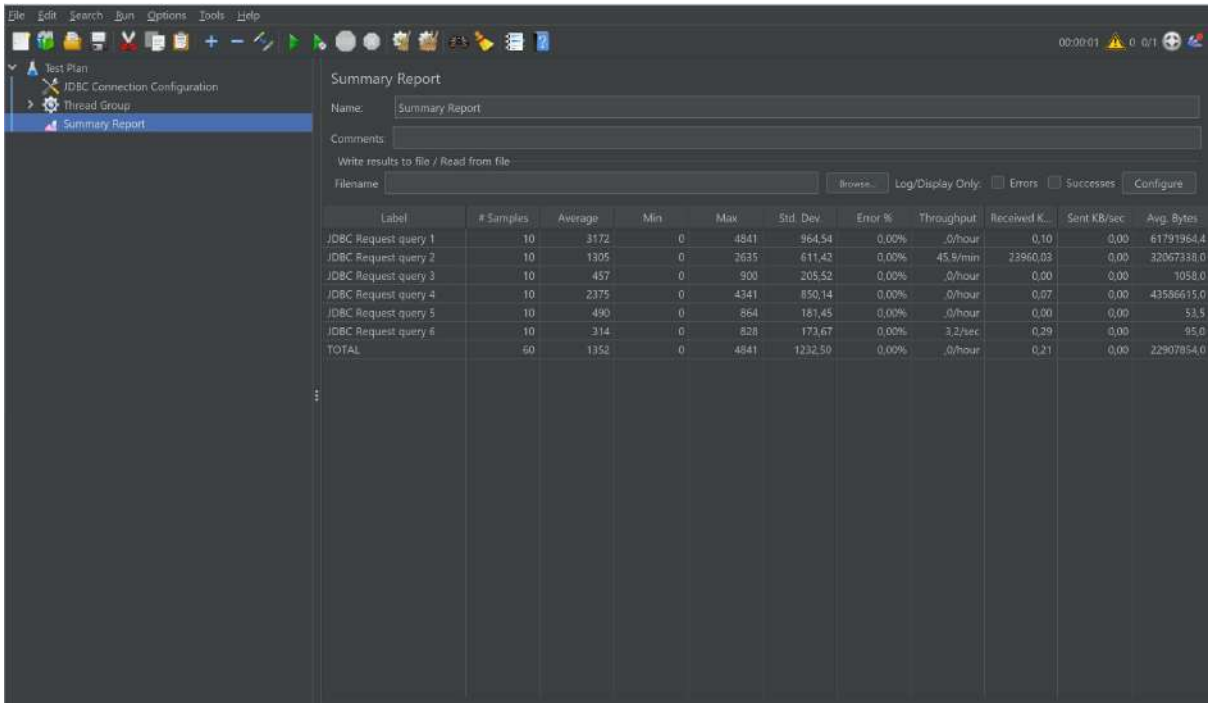
Graf 3: Poraba CPU ob branju podatkov InfluxDB

Vir: (lasten)



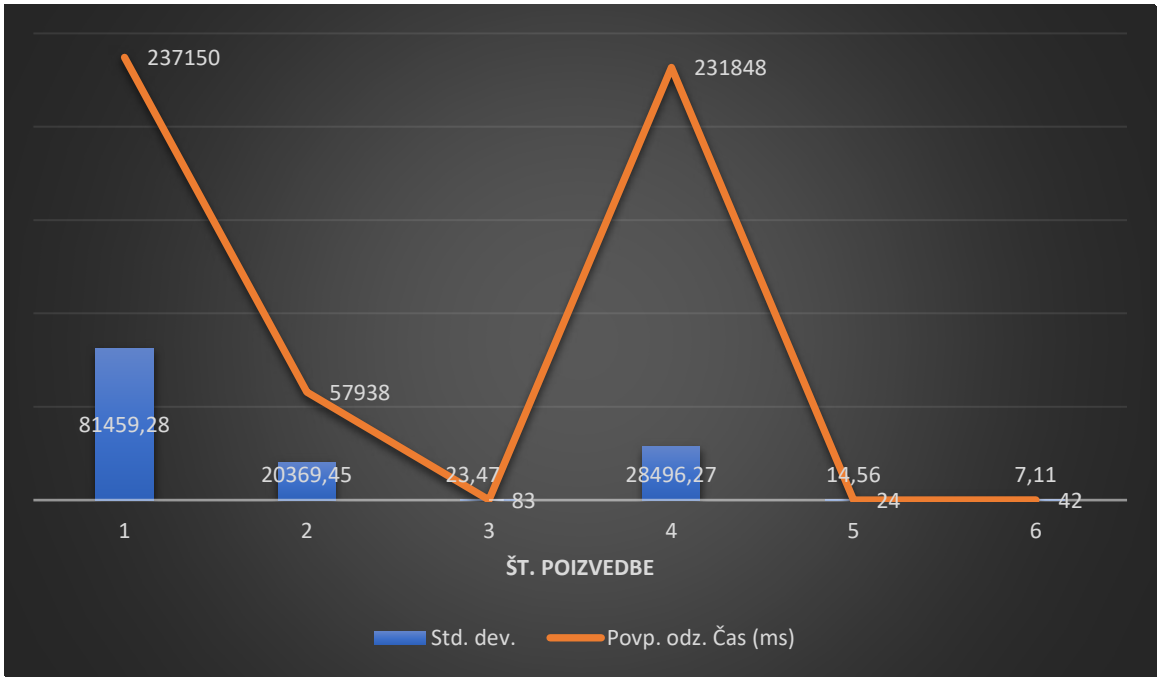
Graf 4: Deviacija in odzivni čas ob branju podatkov TimescaleDB

Vir: (lasten)



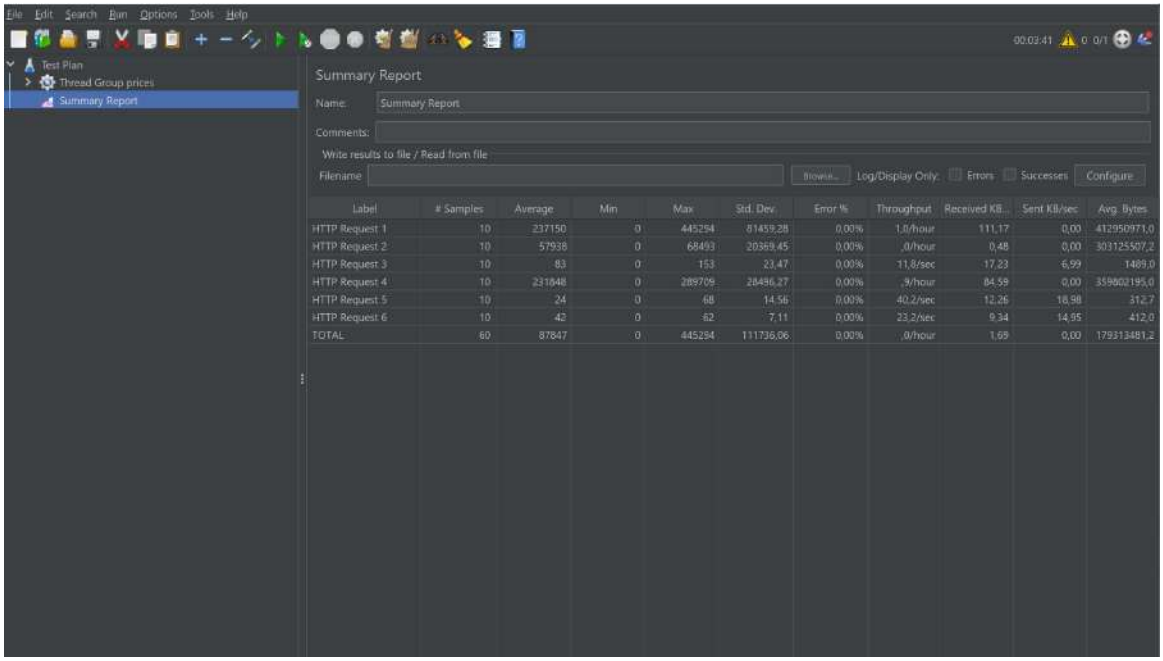
Slika 20: JMeter test branje podatkov TimescaleDB

Vir: (lasten)



Graf 3: Deviacija in odzivni čas ob branju podatkov InfluxDB

Vir: (lasten)



Slika 21: JMeter test branje podatkov InfluxDB

Vir: (lasten)

4.4.3. Ugotovitve

Pri testiranju zapisovanja podatkov v bazo je TimescaleDB beležila veliko hitrejši povprečni odzivni čas 13ms, v primerjavi z InfluxDB, ki je zabeležila odzivni čas 130ms. To pomeni, da je TimescaleDB desetkrat hitrejša v odzivnosti v tem konkretnem testu. Glede napak je

TimescaleDB zabeležila 0,72% napak, medtem ko je InfluxDB med testiranjem delovala brezhibno, torej brez napak. Ko pogledamo standardno deviacijo, opazimo, da je TimescaleDB imela višjo vrednost (196,98) v primerjavi z InfluxDB (80,47). To kaže, da je bila variabilnost odzivnih časov v TimescaleDB večja, kar lahko pomeni manj predvidljive rezultate ali večje nihanje v časih odziva. Glede na porabo sistemskih virov sta bili obe bazi dokaj primerljivi, čeprav je TimescaleDB porabila malo več pomnilnika (74% v primerjavi z 68% za InfluxDB). CPU obremenitev je bila tudi v istem obsegu, kjer je TimescaleDB porabila 18%, InfluxDB pa 21%. Nazadnje, če pogledamo čas testiranja, je InfluxDB potrebovala nekoliko več časa (38:07) v primerjavi s TimescaleDB (34:17). To lahko razložimo z daljšim povprečnim odzivnim časom InfluxDB.

Skupno gledano se TimescaleDB v tem testu izkaže za hitrejšo v odzivnosti, vendar z nekoliko večjo variabilnostjo in manjšo zanesljivostjo v primerjavi z InfluxDB. InfluxDB se izkaže kot zelo zanesljiva, čeprav z večjim odzivnim časom.

Pri analizi in branju podatkov iz baz TimescaleDB in InfluxDB smo zapisali rezultate, ki so ključni za razumevanje njihove zmogljivosti. V TimescaleDB so se povprečni odzivni časi gibali med 314 ms in 3172 ms, medtem ko je InfluxDB dosegla čase od 27 ms do kar 237150 ms. Ta velik razpon časov v InfluxDB nam razkriva, da je ta baza podatkov izredno hitra pri določenih vrstah poizvedb, vendar lahko pri obsežnejših poizvedbah traja veliko dlje kot TimescaleDB. To pokaže, da InfluxDB morda ni najbolj optimalna za določene vrste poizvedb, predvsem za poizvedbe, ki vračajo večjo količino podatkov. Po drugi strani pa se InfluxDB izredno izkaže ob poizvedbah, ki vračajo manjšo količino podatkov. Oba sistema je odlikovala odlična stabilnost, saj je bila stopnja napak v obeh primerih 0%.

Ob pregledu obremenitve sistema opazimo, da je TimescaleDB med izvajanjem poizvedb porabila več spomina in procesorske moči v primerjavi z InfluxDB. Ta podatek je ključnega pomena za tiste, ki želijo optimizirati svoje vire.

Zaključimo lahko, da se TimescaleDB izkaže kot bolj dosledna baza podatkov, ko gre za zagotavljanje hitrih odzivnih časov pri številnih poizvedbah. Na drugi strani pa InfluxDB ponuja izjemno hitrost pri specifičnih poizvedbah, vendar lahko pri obsežnejših poizvedbah traja precej dlje. Izbira med obema bazama podatkov bo torej odvisna od specifičnih potreb in pričakovanj uporabnika. Če je glavna prednostna naloga doslednost odzivnih časov, je morda TimescaleDB boljša izbira, medtem ko bi lahko InfluxDB bolj ustrezala tistim, ki iščejo hitrost

pri določenih vrstah poizvedb in imajo na voljo dodatne vire za obvladovanje občasnih dolgih odzivnih časov.

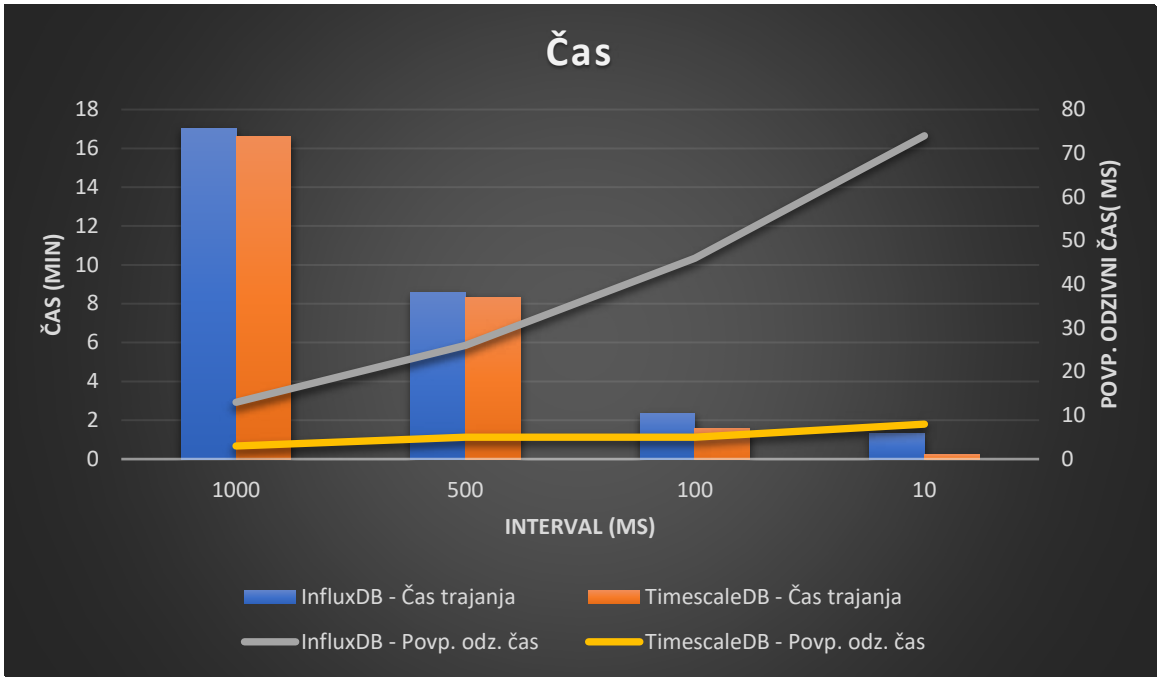
4.5. Primerjava obnašanja pri obdelavi podatkov z različnimi časovnimi intervali

Z uporabo JMeter orodja bomo izvedli teste, s katerimi bomo lahko primerjali obnašanje podatkovnih baz ob vnosu podatkov v različnih časovnih intervalih. Za vsak časovni interval bomo imeli pripravljeno konfiguracijo za 50 uporabnikov, ki bodo poslali skupno 50.000 zahtevkov na strežnik, s katerimi bomo zapisovali v tabelo prices v TimescaleDB in meritev prices v InfluxDB.

Tabela 4: Testiranje različnih časovnih intervalov

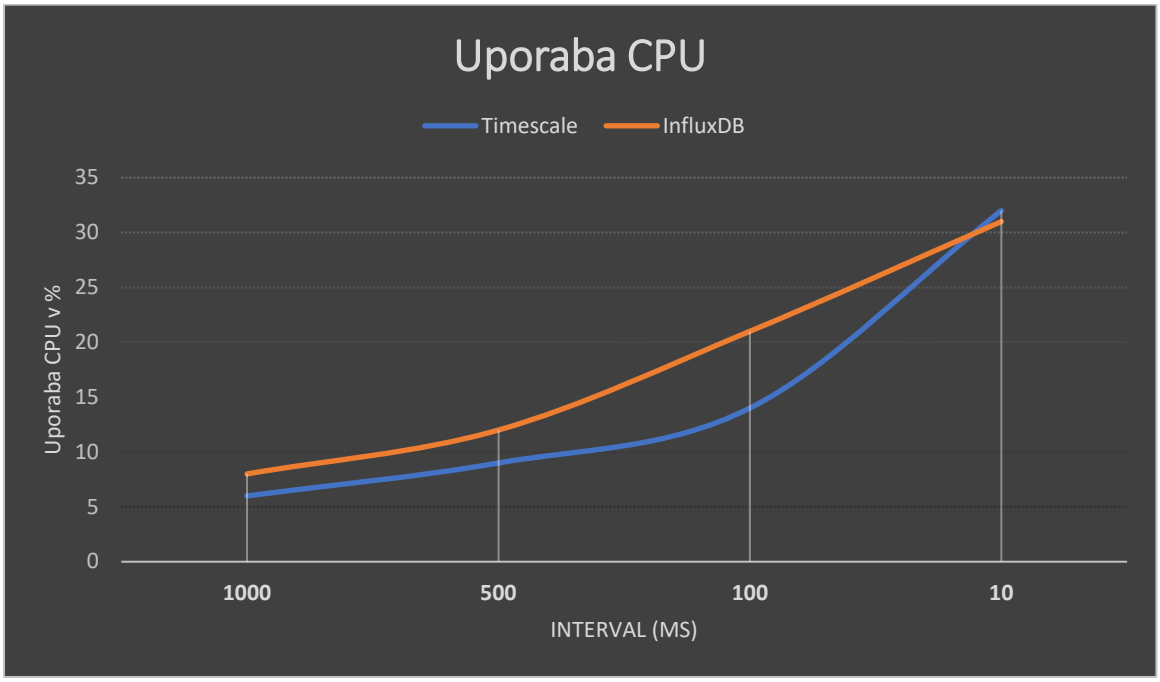
Podatkovna baza	Interval	Povp. odzivni čas	% napak	Std. dev.	CPU	Memory	Čas
InfluxDB	1s	13ms	0,00%	8,25	8%	49%	00:17:04
TimescaleDB	1s	3ms	0,00%	49,24	6%	48%	00:16:59
InfluxDB	0,5s	26ms	0,00%	15,94	12%	61%	00:08:56
TimescaleDB	0,5s	5ms	0,00%	51,03	9%	51%	00:08:34
InfluxDB	0,1s	46ms	0,00%	25,33	21%	65%	00:02:37
TimescaleDB	0,1s	5ms	0,00%	61,92	14%	56%	00:01:55
InfluxDB	0,01s	74ms	0,00%	51,53	31%	67%	00:01:34
TimescaleDB	0,01s	8ms	0,00%	58,97	32%	63%	00:00:22

Vir: (lasten)



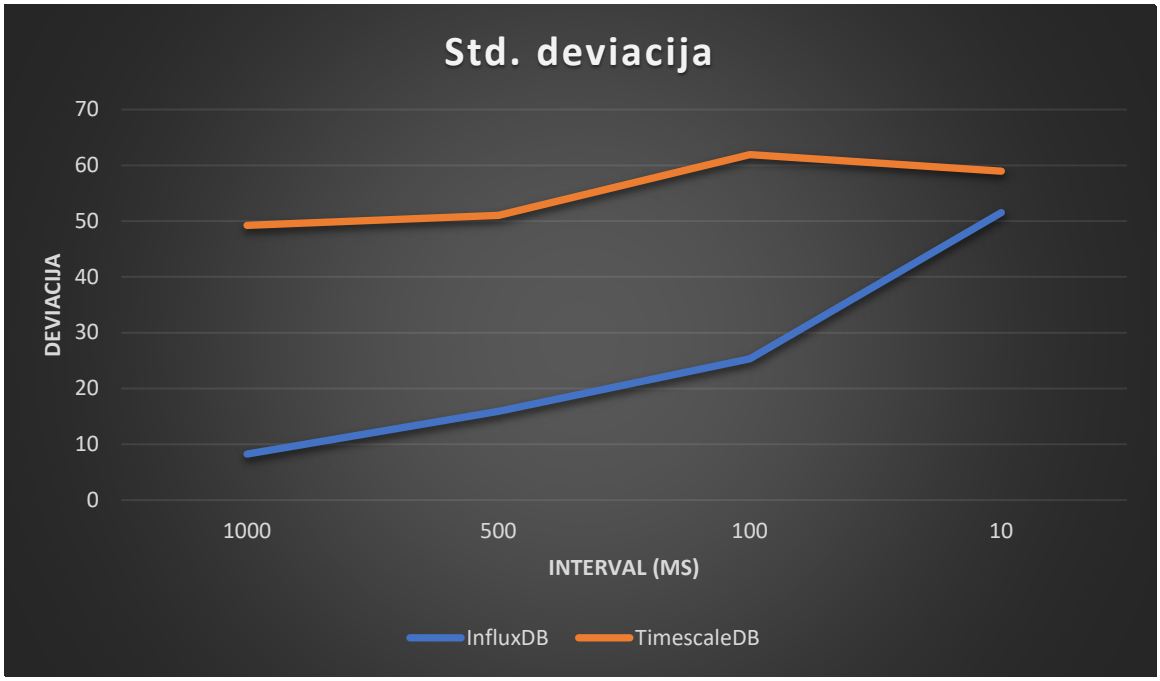
Graf 4: Časi ob testiranju različnih časovnih intervalov

Vir: (lasten)



Graf 5: Poraba CPU ob testiranju različnih časovnih intervalov

Vir: (lasten)



Graf 6: Std. deviacija ob testiranju različnih časovnih intervalov

Vir: (lasten)

V vsakem časovnem intervalu je TimescaleDB dosledno izkazala hitrejši odzivni čas v primerjavi z InfluxDB. To je še posebej izstopalo v intervalu 1s, kjer se je TimescaleDB odzvala v zgolj 3ms, medtem ko je InfluxDB potrebovala 13ms. Kljub tej dosledni hitrosti pa je TimescaleDB pokazala večjo variabilnost v odzivnih časih, kot kaže njena višja standardna deviacija v primerjavi z InfluxDB. Kar zadeva zanesljivost, sta obe bazi pokazali izjemno trdnost z 0% napakami skozi vse teste. To poudarja, da sta obe bazi zanesljivi, ne glede na frekvenco zahtev. S krajšanjem časovnega intervala se je povečala tako obremenitev CPU kot poraba spomina. TimescaleDB je, še posebej pri krajših intervalih, zahtevala nekoliko več CPU, medtem ko je bila razlika v porabi pomnilnika med obema bazama minimalna. V večini intervalov je TimescaleDB končala test hitreje kot InfluxDB, pri čemer je bila najopaznejša razlika v intervalu 0,01s. V tem intervalu je TimescaleDB potrebovala samo 22 sekund, medtem ko je InfluxDB zaključila v 1 minuti in 34 sekundah.

Sklenemo lahko, da je TimescaleDB v tem nizu testov pokazala boljše zmogljivosti v smislu hitrosti, čeprav z nekoliko večjo variabilnostjo. Kljub temu sta obe bazi demonstrirali visoko zanesljivost.

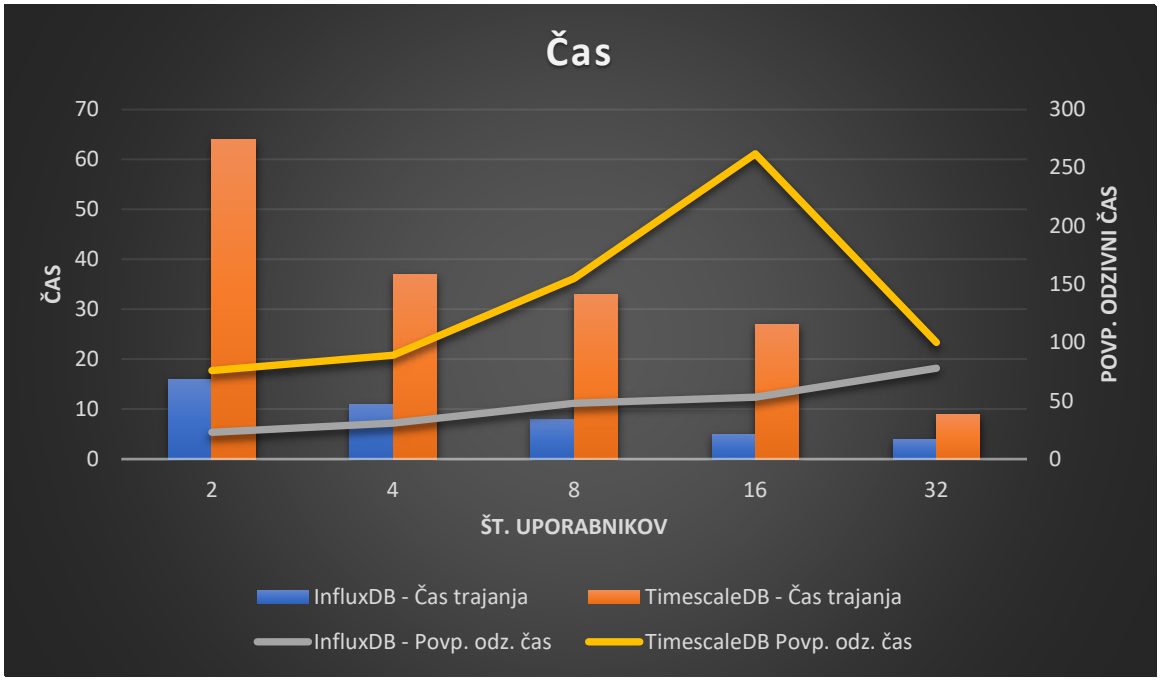
4.6. Primerjava možnosti za skaliranje

Z uporabo JMeter orodja bomo izvedli teste, s katerimi bomo lahko primerjali možnosti skaliranja obeh podatkovnih baz. Za vsako podatkovno bazo bomo pripravili test, pri katerem bomo zviševali število uporabnikov. V samem testu bo polovica uporabnikov izvajala zapisovanje v podatkovno bazo, druga polovica pa bo izvajala branje iz podatkovne baze. Vsaka tabela ima ob začetku izvajanja testov 900.000 zapisov. Začeli bomo s konfiguracijo dveh uporabnikov in postopoma dvigovali število uporabnikov. V vsakem testu bomo na strežnik poslali skupno 50.000 zahtevkov.

Tabela 5: Skaliranje uporabnikov

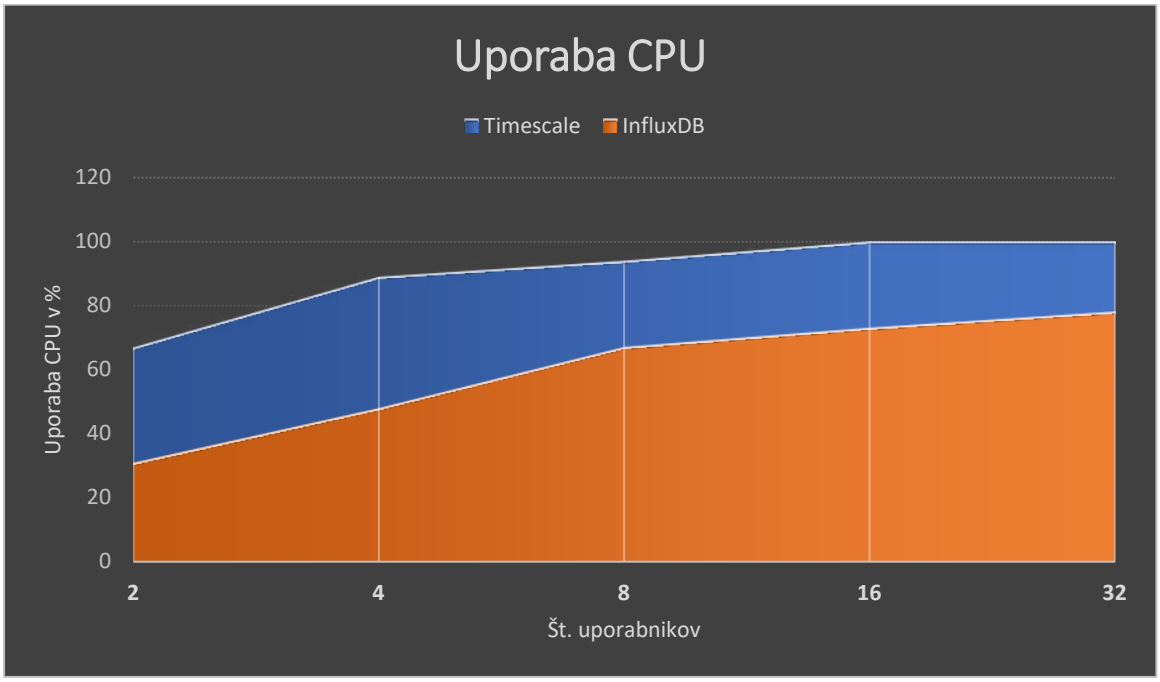
Podatkovna baza	Št. Up.	Povp. odzivni čas (ms)	% napak	Std. dev.	CPU	Memory	Čas(v urah)
InfluxDB	2	23	0,00%	17,07	31%	65%	00:16
TimescaleDB	2	76	0,00%	77,41	67%	51%	01:04
InfluxDB	4	31	0,00%	20,60	48%	68%	00:11
TimescaleDB	4	89	0,00%	89,80	89%	56%	00:37
InfluxDB	8	48	0,00%	34,43	64%	58%	00:08
TimescaleDB	8	155	0,00%	160,98	94%	58%	00:33
InfluxDB	16	53	0,00%	38,63	73%	59%	00:05
TimescaleDB	16	262	12,53%	473,43	100%	61%	00:27
InfluxDB	32	78	0,00%	57,94	78%	59%	00:04
TimescaleDB	32	100	66,06%	475,77	100%	63%	00:09

Vir: (lasten)



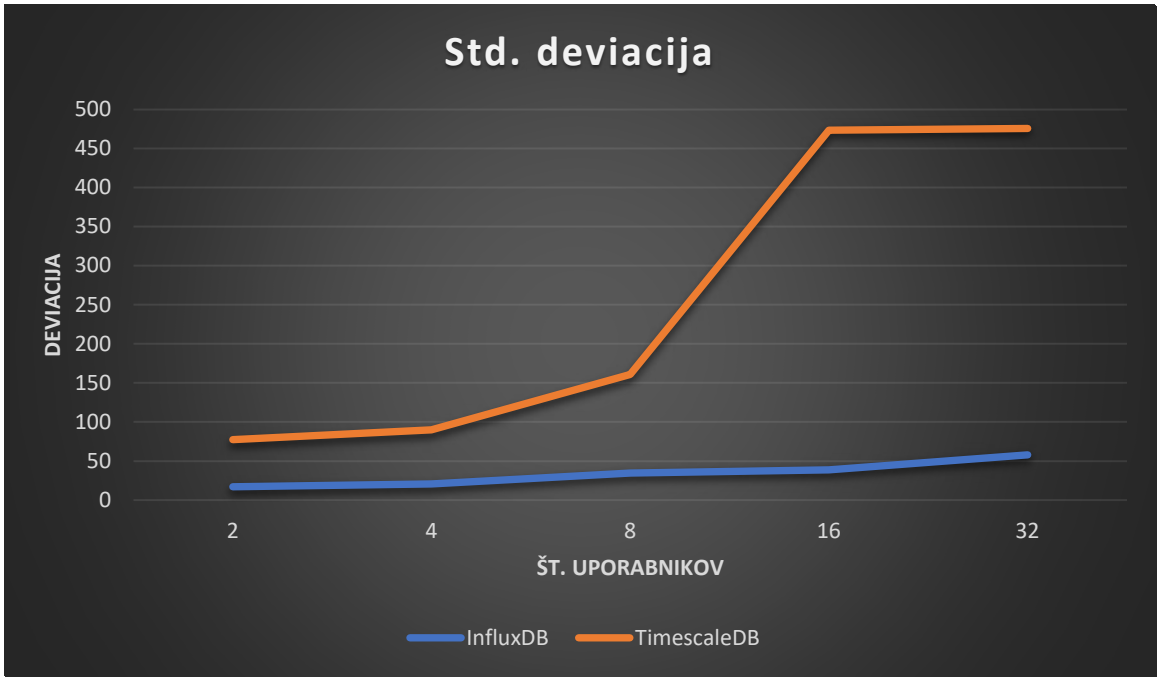
Graf 7: Časi ob testiranju možnosti skaliranja

Vir: (lasten)



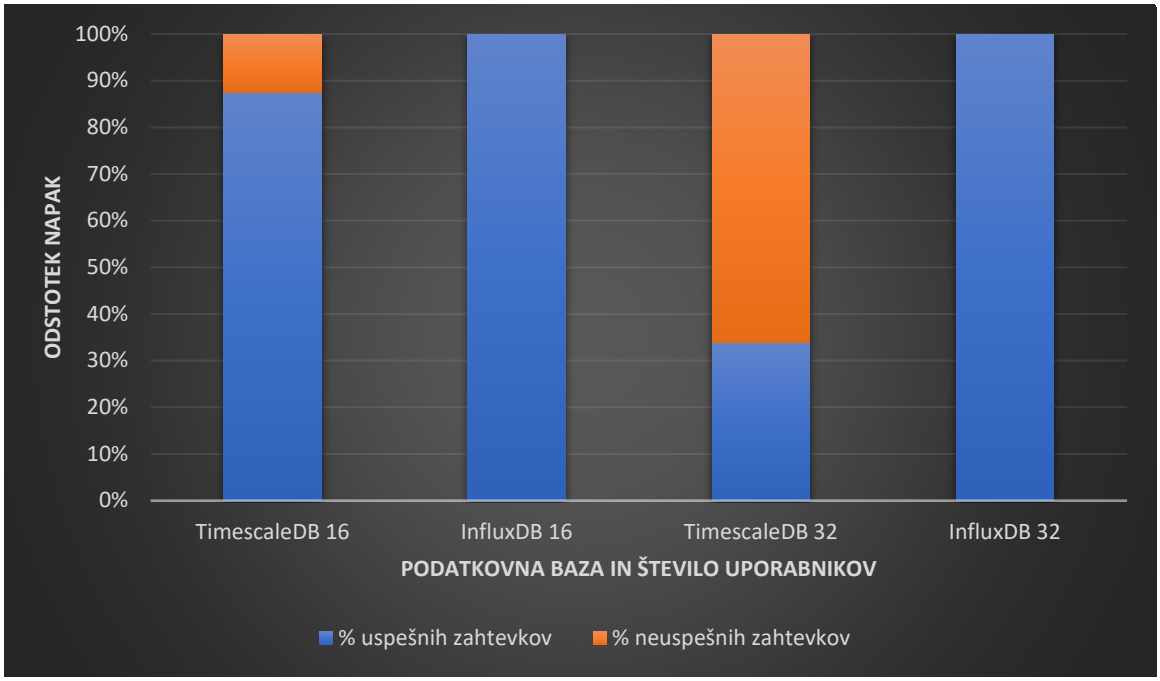
Graf 8: Uporaba CPU ob testiranju možnosti skaliranja

Vir: (lasten)



Graf 9: Deviacija ob testiranju možnosti skaliranja

Vir: (lasten)



Graf 10: Odstotek napak ob testiranju možnosti skaliranja

Vir: (lasten)

Prvi ključni ugotovitvi sta, da je InfluxDB v vseh testih dosledno dosegala hitrejše odzivne čase kot TimescaleDB, ter da je InfluxDB ohranila 0% napak skozi vse teste. V nasprotju s tem je TimescaleDB pri 32 uporabnikih pokazala skrb vzbujajoč visok odstotek napak, kar 66,06%.

Glede na standardno deviacijo, ki predstavlja stabilnost odziva, je InfluxDB tudi tukaj prednjačila z bolj doslednimi rezultati.

Vprašanje zmogljivosti nas je pripeljalo do opažanja, da je TimescaleDB ob večji obremenitvi hitro dosegla 100% obremenitev CPU, medtem ko je InfluxDB ohranjala obremenitev pod 80%. Prav tako je v večini primerov InfluxDB končala obdelavo zahtevkov v krajšem času.

Sklepamo lahko, da v obravnavanih testnih okoliščinah InfluxDB izkazuje prednost pred TimescaleDB v hitrosti, zanesljivosti in zmogljivosti in je boljša izbira za aplikacije, ki zahtevajo dobre možnosti za skaliranje, zanesljivost in učinkovitost.

5. POTRDITEV ALI ZAVRNITEV HIPOTEZ

H1: InfluxDB podatkovna baza je hitrejša od TimescaleDB podatkovne baze

V poglavju 4.3.1. smo izvajali teste zapisovanja podatkov, kjer smo lahko videli, da je pri zapisovanju podatkov TimescaleDB vidno prevladovala, saj je imela desetkrat hitrejši povprečni odzivni čas v primerjavi z InfluxDB. To je pomemben pokazatelj učinkovitosti, saj lahko hitro zapisovanje podatkov v veliki meri vpliva na delovanje aplikacij v realnem času. Ko pa preidemo na branje podatkov, postane slika nekoliko bolj zapletena. Čeprav je TimescaleDB v nekaterih primerih pokazala hitrejše odzivne čase, je InfluxDB v drugih primerih dosegla zelo visoke odzivne čase, predvsem kadar poizvedbe vračajo veliko količino rezultatov, medtem ko je bila pri poizvedbah, ki vračajo manjšo količino podatkov izredno učinkovita, kar je povzročilo velika odstopanja v njenih rezultatih, kar smo lahko videli v testih, ki smo jih izvajali v poglavju 4.3.2. Zanimivo je tudi opazovati, kako se obe podatkovni bazi obnašata ob različnih časovnih intervalih, kar smo opazovali v poglavju 4.4. TimescaleDB je bila dosledno hitrejša pri vseh testiranih intervalih, kar kaže na njeno robustnost in učinkovitost pri obdelavi podatkov v realnem času, vendar je treba poudariti, da se je pri obnašanju ob različnih časovnih intervalih testiralo samo zapisovanje podatkov, kjer se je že v prejšnjem testu zapisovanja podatkov TimescaleDB izkazala za bolj učinkovito. Vendar pa se je InfluxDB izkazala za hitrejšo pri testiranju ob različnih količinah uporabnikov, zlasti pri manjšem številu uporabnikov, kjer se je testirala kombinacija branja in zapisovanja podatkov. To kaže na njeno sposobnost, da se učinkovito prilagodi obremenitvam in zagotovi hitre odzivne čase, ko je to najbolj potrebno.

Če povzamemo vse rezultate, lahko pridemo do ugotovitve, da je naša hipoteza delno potrjena. Medtem ko je TimescaleDB v večini scenarijev pokazala boljše rezultate, je InfluxDB v določenih okoliščinah prevladovala. Tako je odgovor na našo hipotezo odvisen od specifičnega konteksta in načina uporabe obeh podatkovnih baz.

H2: InfluxDB podatkovna baza ima večjo podporo

Ko razmišljamo o podpori podatkovnih baz, je potrebno razumeti, da ne gre le za tehnično podporo, ampak tudi za skupnost, dokumentacijo in ekosistem, ki obdaja določeno tehnologijo. V primeru InfluxDB in TimescaleDB imamo dve močni rešitvi, ki sta v vrhu priljubljenost v

svetu podatkovnih baz časovnih vrst. InfluxDB, ki je namensko zgrajena za časovne vrste, je pridobila veliko priljubljenost v skupnosti, ki se ukvarja z meritvami in metrikami. Njen integriran ekosistem, ki vključuje orodja kot so Telegraf, Chronograf in Kapacitor, omogoča uporabnikom, da zgradijo celovite rešitve za časovne vrste. Poleg tega obsežna dokumentacija in aktivna skupnost zagotavljata, da imajo uporabniki podporo vedno na voljo.

Po drugi strani pa TimescaleDB prinaša najboljše iz obeh svetov. Kot razširitev za eno izmed najbolj priljubljenih relacijskih podatkovnih baz, PostgreSQL, TimescaleDB združuje moč relacijskega sistema za upravljanje z bazami podatkov s funkcionalnostjo za časovne vrste. To pomeni, da lahko uporabniki izkoristijo vse napredne funkcionalnosti PostgreSQL-a, hkrati pa imajo koristi od specializirane podpore za časovne vrste. Tudi TimescaleDB se ponaša z obsežno dokumentacijo in rastočo skupnostjo.

Zato, čeprav se morda zdi, da ima InfluxDB nekoliko večjo vidnost v industriji časovnih vrst, TimescaleDB zagotovo ni daleč za njo, zlasti ko upoštevamo njeno integracijo s PostgreSQL. Obe podatkovni bazi ponujata močne skupnosti in obsežne vire za uporabnike.

Obe sta močni rešitvi z obsežno podporo, ki ju lahko uporabniki izkoristijo glede na svoje specifične potrebe. Glede na to, da imata obe podatkovni bazi obsežno dokumentacijo, nudita tehnično podporo, imata brezplačne tečaje, ki so na volju tudi v video obliki, hkrati pa imata obe zelo aktivne skupnosti na GitHub-u, Slack-u in forumih je zelo težko trditi, da ima ena večjo podporo kot druga, zato te hipoteze ne moremo ne potrditi in ne zavreči.

H3: TimescaleDB podatkovna baza je lažja za namestitev

Čeprav je InfluxDB zasnovana kot "plug-and-play" rešitev, kar pomeni minimalno konfiguracijo in hitro namestitev, TimescaleDB prav tako ponuja preprost postopek namestitve, ki vključuje dodajanje razširitve k že obstoječemu PostgreSQL okolju. Vendar pa je treba upoštevati, da namestitev TimescaleDB zahteva dodaten korak namestitve PostgreSQL. Lahko bi rekli, da je InfluxDB nekoliko lažja za namestitev, saj je zasnovana kot "plug-and-play" rešitev. Vendar je razlika v kompleksnosti namestitve med obema podatkovnima bazama minimalna, saj obe ponujata preproste postopke namestitve. Hipotezo, da je TimescaleDB lažja za namestitev v primerjavi z InfluxDB, lahko brez dvoma zavržemo.

H4: InfluxDB podatkovna baza je primernejša za uporabo z Big Data

Ko govorimo o uporabi podatkovnih baz v kontekstu Big Data, je ključnega pomena, da izberemo pravo orodje, ki lahko obvladuje velike količine podatkov hitro in učinkovito. Pri zapisovanju podatkov je TimescaleDB jasno izstopala s svojo hitrostjo, saj je bila desetkrat hitrejša kot InfluxDB. To je pomemben pokazatelj, saj v okolju Big Data v večini primerov potrebujemo hitro zapisovanje velikih količin podatkov. Branje podatkov je prav tako zelo pomembno, še posebej, ko obdelujemo velike količine informacij. InfluxDB je v nekaterih testih pokazala zelo visoke odzivne čase, kar lahko kaže na težave pri obdelavi velikih naborov podatkov. Ko smo preučevali odzivne čase pri različnih časovnih intervalih, je TimescaleDB ponovno prevladovala, kar kaže na njeno doslednost in učinkovitost pri obdelavi podatkov v realnem času. Zanimivo pa je, da je InfluxDB bolje obvladovala obremenitve ob testiranju skaliranja, predvsem na račun nižjih odzivnih časov ob branju podatkov, kjer je imela tudi konkretno nižji odstotek napak, saj je podatkovna baza TimescaleDB imela ob 32 uporabnikih odstotek napak kar 66%. Vendar je treba poudariti, da imamo pri podatkovni bazi TimescaleDB večjo možnost konfiguracije baze in optimizacije tabel, kar pomeni, da bi lahko te rezultate ob bolj optimalni konfiguraciji tudi izboljšali. Čeprav je InfluxDB pokazala nekatere prednosti, kot so nižji odstotek napak in lažje skaliranje uporabnikov, so njeni daljši odzivni časi in nekonsistentnost rezultatov pri večjih obremenitvah razlog za zaskrbljenost. Tako lahko sklenemo, da naša hipoteza o primernosti InfluxDB za uporabo z Big Data glede na rezultate testov, ki smo jih mi opravili, ne drži. TimescaleDB se zdi bolj robustna in zanesljiva izbira za obdelavo in analizo velikih količin podatkov, je pa odgovor na to hipotezo odvisen tudi od glavnega cilja zbiranja podatkov, recimo zbiranje podatkov za kasnejšo analizo (npr. zbiranje log datotek ali senzorskih podatkov), kjer je hitro in zanesljivo zapisovanje podatkov ključnega pomena, ali pa, če je glavni cilj analiza podatkov v realnem času ali obdelava kompleksnih poizvedb za pridobivanje vpogledov, kjer je hitro branje podatkov bolj pomembno.

6. NAPOTKI ZA NADALJNJE DELO

V okviru diplomskega dela sem se poglobil v primerjavo podatkovnih baz časovnih vrst InfluxDB in TimescaleDB, s poudarkom na njihovi zmogljivosti, obnašanju in skalabilnosti predvsem v kontekstu Big Data. Obstaja še veliko področij, s katerimi lahko razširimo temo, ki smo jo obravnavali. V kontekstu tega, kar smo obravnavali v tem diplomskem delu, bi lahko pripravili dodatne teste predvsem na področju branja podatkov, s katerimi bi predvsem preverjali zmogljivost in skalabilnost podatkovnih baz. Eno izmed ključnih področij, ki bi ga bilo vredno dodatno raziskati, je notranja arhitektura obeh podatkovnih baz. Poglobljeno razumevanje, kako sta zasnovani in kako obdelujeta podatke, bi lahko ponudilo dodatne vpoglede v njuno zmogljivost in učinkovitost. Poleg tega bi bilo koristno raziskati varnostne funkcije, protokole in prakse obeh baz, da bi bolje razumeli, kako se obnašajo v kritičnih scenarijih, kot so izgube podatkov ali varnostne kršitve. Prav tako bi bilo zanimivo preučiti, kako se InfluxDB in TimescaleDB integrirata z drugimi orodji in platformami. To bi vključevalo orodja za vizualizacijo, skladiščenje podatkov in obdelavo podatkov v realnem času. Na koncu bi lahko pripravili tudi stroškovno analizo, primerjavo stroškov vzdrževanja, skaliranja in upravljanja obeh podatkovnih baz, ki bi prav tako ponudila informacije, ki bi lahko bile ključnega pomena predvsem za organizacije, ki razmišljajo o njihovi uporabi. Namen diplomskega dela je bila primerjava ene SQL in ene NoSQL podatkovne baze časovnih vrst. Za nadaljnje raziskave bi bilo koristno v primerjavo vključiti tudi druge podatkovne baze časovnih vrst. To bi omogočilo bolj celovit pogled na krajino podatkovnih baz časovnih vrst in na to, kako se različne baze obnašajo v primerjavi druga z drugo. S tem diplomskim delom smo dobili vpogled v vzpostavitev dveh podatkovnih baz časovnih vrst in njuno obnašanje, zmogljivost ter skalabilnost, vendar obstaja še veliko potenciala za nadaljnje raziskave na tem področju, če želimo pridobiti še bolj poglobljeno razumevanje obeh podatkovnih baz.

7. SKLEP

V okviru tega diplomskega dela smo se podrobno posvetili primerjavi dveh vodilnih podatkovnih baz za obdelavo časovnih serij: InfluxDB in TimescaleDB. Naš cilj ni bil le površinsko ocenjevanje, temveč poglobljena analiza ključnih značilnosti, ki vplivajo na izbiro ustreznega sistema za upravljanje baz podatko. Analiza je zajela širok spekter parametrov, ki so bistvenega pomena za odločanje pri implementaciji podatkovnih rešitev v realnih projektih. Osredotočili smo se na različne vidike delovanja teh baz, vključno z zmogljivostjo, obnašanjem, skalabilnostjo in podporo, s posebnim poudarkom na kontekstu Big Data.

Podatkovni bazi InfluxDB in TimescaleDB smo preizkusili v različnih testnih scenarijih, ki so simulirali realno delovno breme in pogoje, kot se pojavljajo v praksi. Obe podatkovni bazi sta pokazali visoko mero robustnosti in zmogljivosti, vsaka s svojimi prednostmi, ki so lahko odločilne v določenih aplikacijskih kontekstih.

TimescaleDB je v večini testnih scenarijev odlikovala visoka zmogljivost zlasti pri zapisovanju podatkov, kjer je pokazala izjemno doslednost in hitrost. Takšne značilnosti so nujne za aplikacije, kjer je ključnega pomena hitro in zanesljivo zapisovanje velikih količin podatkov, kot na primer pri finančnih transakcijah, kjer vsak milisekundni zamik lahko pomeni razliko v vrednosti transakcije. Dodatno prednost TimescaleDB predstavlja njena tesna integracija z PostgreSQL, kar pomeni, da se lahko razvijalci zanašajo na dobro uveljavljene mehanizme varnosti, transakcijskega vodenja in podporo standardnim SQL operacijam.

Na drugi strani pa je InfluxDB izstopala v scenarijih, ki so zahtevali hitro in učinkovito branje podatkov, zlasti pri visokih obremenitvah. Njena arhitektura je bila posebej optimizirana za hitro izvajanje kompleksnih poizvedb, kar je ključno v aplikacijah, kot so spremljanje omrežij, kjer lahko hitrost analize podatkov neposredno vpliva na odzivne čase pri odpravljanju incidentov.

Pri podpori, ki je kritična za dolgoročno vzdrževanje in nadgradnjo sistemov, obe podatkovni bazi ponujata široko paleto virov. InfluxDB ima izjemno aktivno skupnost, ki je razvila bogat ekosistem orodij in integracij, ki olajšajo razvoj in razširitev funkcionalnosti. TimescaleDB pa se ponaša z močno podporo, ki izhaja iz svoje osnove na PostgreSQL, kar pomeni dostop do obsežne dokumentacije in zanesljive skupnosti, ki je skozi leta razvila zanesljive prakse za upravljanje podatkov. Obe podatkovni bazi pa nista omejeni le na uradna spletna mesta.

Dejavnosti okoli teh podatkovnih baz je mogoče najti na različnih platformah po spletu, kot sta priljubljen tehnični forum, Stack Overflow in platforma za predvajanje video vsebin Youtube.

V smislu namestitve je InfluxDB morda na prvi pogled lažja za namestitev, saj je bila zasnovana kot samostojna podatkovna baza, ki je od začetka prilagojena delu s časovnimi vrstami. Vendar pa razlike v kompleksnosti postopka namestitve med TimescaleDB in InfluxDB niso tako izrazite, da bi lahko bile odločilni faktor pri izbiri saj obe ponujata preproste postopke namestitve, ki so zasnovani tako, da uporabnikom omogočajo hitro in učinkovito vzpostavitev podatkovne baze.

Ko govorimo o uporabi v kontekstu Big Data, je TimescaleDB verjetno bolj primerna izbira. Njena robustnost in zanesljivost sta ključni za njeno prednost v tem kontekstu. Vendar pa je odgovor na to vprašanje odvisen tudi od specifičnega konteksta in načina uporabe obeh baz. Sposobnost obvladovanja obsežnih podatkovnih nizov in zagotavljanje hitrih poizvedb je neprecenljiva, še posebej v industrijskih aplikacijah, kjer so časovne serije ključni del operativnih meritev.

Zaključno lahko potrdimo, da je izbira med InfluxDB in TimescaleDB odvisna od številnih faktorjev, vključno s specifičnimi tehničnimi zahtevami, predvidenim načinom uporabe in cilji aplikacije ali organizacije. Pomembno je temeljito pretehtati vse ugotovitve, upoštevati posamezne prednosti in slabosti ter se zavedati, da ni univerzalne rešitve, saj obe podatkovni bazi ponujata močne in zanesljive rešitve za obdelavo časovnih vrst. Priporočljivo je tudi nadaljnje raziskovanje in testiranje v specifičnih okoljih, da se zagotovi najboljša možna rešitev za določen projekt ali aplikacijo.

8. VIRI

Apache JMeter. (brez datuma). Pridobljeno iz Apache Software Foundation: https://jmeter.apache.org/

Arif, A. (23. 5 2023). *Timescale*. Pridobljeno 2023 iz https://www.timescale.com/blog/what-is-a-time-series-plot-and-how-can-you-create-one/

Atelšek, S., Tanja, F., Jemec Tomazin, M., Trojar, M., & Žagar Karer, M. (11. 12 2017). Koordinirani univerzalni čas. Ljubljana, Slovenija. Pridobljeno iz https://isjfr.zrc-sazu.si/sl/terminologisce/svetovanje/koordinirani-univerzalni-cas

Characteristics of Big Data: Types & Examples. (01. 06 2020). Pridobljeno iz https://bau.edu/blog/characteristics-of-big-data/

Coursera. (16. 06 2023). SQL vs. NoSQL: The Differences Explained + When to Use Each. Pridobljeno iz https://www.coursera.org/articles/nosql-vs-sql

FAQ. (brez datuma). Pridobljeno iz Graphite: https://graphite.readthedocs.io/en/latest/faq.html

Freedman, M. (14. 08 2018). TimescaleDB vs. InfluxDB: purpose built differently for time-series data. Pridobljeno 09 2023 iz https://medium.com/timescale/timescaledb-vs-influxdb-for-time-series-data-timescale-influx-sql-nosql-36489299877

Getting started with Timescale. (brez datuma). Pridobljeno iz Timescale: https://docs.timescale.com/getting-started/latest/

Introduction to InfluxDB. (brez datuma). Pridobljeno iz Influxdata: https://awesome.influxdata.com/docs/part-1/introduction-to-influxdb/#what-is-influxdb

Namiot, D. (2015). Time Series Databases. *Data Analytics and Management in Data Intensive Domains.* Pridobljeno iz https://www.researchgate.net/publication/286732446_Time_Series_Databases

Noor Zehra, S., & Yfantidou, S. (17. 12 2017). Time Series Databases and InfluxDB. Bruselj, Belgija. Pridobljeno iz https://www.devopsschool.com/blog/wp-content/uploads/2022/09/influxdb_2017.pdf

Overview. (brez datuma). Pridobljeno iz OpenTSDB: https://opentsdb.net/overview.html

Overview. (brez datuma). Pridobljeno iz Prometheus: https://prometheus.io/docs/introduction/overview/

Silberschatz, A., Korth, H., & Sudarshan, S. (2019). *Database System Concepts* (Sedma izd.). New York: McGraw-Hill.

PRILOGA

Koda

```
-- Podatkovne strukture za podatkovno bazo postgres
CREATE TABLE exchanges (
    exchange_id SERIAL,
    name VARCHAR(100) NOT NULL,
    location VARCHAR(100),
    currency_code CHAR(3),
    PRIMARY KEY (exchange_id)
);

CREATE TABLE stocks (
    stock_id SERIAL,
    exchange_id INTEGER,
    symbol VARCHAR(20) NOT NULL,
    company_name VARCHAR(200),
    PRIMARY KEY (stock_id),
    FOREIGN KEY (exchange_id) REFERENCES exchanges(exchange_id)
);

CREATE TABLE prices (
  timestamp TIMESTAMPTZ NOT NULL,
  stock_id INTEGER,
```

```
  price DOUBLE PRECISION,
  PRIMARY KEY (timestamp, stock_id),
  FOREIGN KEY (stock_id) REFERENCES stocks(stock_id)
);
SELECT create_hypertable('prices', 'timestamp');


--Poizvedbe
-- TimescaleDB
-- 1. Preprosta poizvedba
SELECT *
  FROM prices;
-- 2. Join
SELECT p.timestamp, p.price, s.symbol, s.company_name
  FROM prices p
  JOIN stocks s ON p.stock_id = s.stock_id
  WHERE p.stock_id < 1000;


-- 3. Z indeksom
SELECT timestamp, price
  FROM prices
  WHERE stock_id in (1, 800, 180518, 44000);
-- 4. Brez indeksa
SELECT timestamp, price
  FROM prices
```

```
  WHERE price > 100;


-- 5.Poizvedbe, ki uporabljajo agregatne funkcije
SELECT AVG(price)
  FROM prices;
-- 6. Poizvedba, ki uporablja podpoizvedbo
SELECT s.company_name
  FROM stocks s
  WHERE s.stock_id = (SELECT stock_id
                                FROM prices
                        ORDER BY price DESC
                        LIMIT 1);


-- influxDB
--1. Preproste poizvedbe
from(bucket: "academia_log")
  |> range(start: 0)
  |> filter(fn: (r) => r._measurement == "prices")
--2. Poizvedbe z razponom casa:
from(bucket: "academia_log")
  |> range(start: -7d)
  |> filter(fn: (r) => r._measurement == "prices")
--3. Poizvedba z indeksom
from(bucket: "academia_log")
```

```
  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
  |> filter(fn: (r) => r["_measurement"] == "prices")
  |> filter(fn: (r) => r["_field"] == "price")
  |> filter(fn: (r) => r["symbol"] == "AAAAA" or r["symbol"] == "AAAAE" or r["symbol"] == "AAABK" or r["symbol"] == "AAABN" or r["symbol"] == "AAACM" or r["symbol"] == "AAACR")
--4. Poizvedba brez indeksa
from(bucket: "academia_log")
    |> range(start: 0)
    |> filter(fn: (r) => r._measurement == "prices" and
        (r._value == 251.0 or r._value == 485.0 or r._value == 601.0 or r._value == 542.0)
  )
--5. Agregatne  funkcije
from(bucket: "academia_log")
  |> range(start: 0)
  |> filter(fn: (r) => r._measurement == "prices" and r.exchange_name == "AAAAH")
  |> group(columns: [])
  |> mean()
--6. Podpoizvedba
highPrices = from(bucket: "academia_log")
    |> range(start: 0)
    |> filter(fn: (r) =>
        r._measurement == "prices" and
        r._field == "price" and
```

```
    (r.symbol == "SNNOB" or r.symbol == "ITEOL" or r.symbol == "PDJDK" or r.symbol
== "DZTMF" or r.symbol == "FNCKV")

    )

highPrices

  |> group(columns: [])

  |> max()


--Priprava podatkov

CREATE OR REPLACE PROCEDURE new_price()

LANGUAGE plpgsql

AS $$

DECLARE

w_stock_id INTEGER;

w_price DOUBLE PRECISION;

BEGIN

  w_stock_id := TRUNC(RANDOM() * 200000)::numeric;


  w_price := (TRUNC((RANDOM() * 1000)::numeric, 2));

  INSERT INTO prices (timestamp, stock_id, price)

  VALUES (now(), w_stock_id, w_price);

END;

$$;

/


CREATE OR REPLACE PROCEDURE exchanges_prepare_data(p_row_count
INTEGER)

LANGUAGE plpgsql

AS $$

DECLARE
```

```
    i INTEGER;
    random_name VARCHAR(100);
    random_location VARCHAR(100);
    selected_currency VARCHAR2(4);
    rand_num NUMBER;
BEGIN
    FOR i IN 1..p_row_count LOOP
        SELECT STRING_AGG(SUBSTRING('ABCDEFGHIJKLMNOPQRSTUVWXYZ'
FROM (floor(random() * 26) + 1)::integer FOR 1), '')
        FROM generate_series(1, 10) INTO random_name;


        SELECT STRING_AGG(SUBSTRING('ABCDEFGHIJKLMNOPQRSTUVWXYZ'
FROM (floor(random() * 26) + 1)::integer FOR 1), '')
        FROM generate_series(1, 10) INTO random_location;


        rand_num := RANDOM();
        IF rand_num < 0.5 THEN
          selected_currency := 'EUR';
        ELSE
          selected_value := 'USD';
        END IF;


        INSERT INTO exchanges (name, location, currency_code) VALUES
(random_name, random_location, selected_currency);
    END LOOP;
END;
$$;
/


CREATE OR REPLACE PROCEDURE stocks_prepare_data(p_row_count
INTEGER)
LANGUAGE plpgsql
```

```
AS $$
DECLARE
    i INTEGER;
    w_symbol VARCHAR(20);
    w_company_name VARCHAR(200);
    w_exchange_id INTEGER;
BEGIN
    FOR i IN 1..p_row_count LOOP
        SELECT STRING_AGG(SUBSTRING('ABCDEFGHIJKLMNOPQRSTUVWXYZ'
FROM (floor(random() * 26) + 1)::integer FOR 1), '')
        FROM generate_series(1, 5) INTO w_symbol;


        SELECT STRING_AGG(SUBSTRING('ABCDEFGHIJKLMNOPQRSTUVWXYZ'
FROM (floor(random() * 26) + 1)::integer FOR 1), '')
        FROM generate_series(1, 50) INTO w_company_name;


        SELECT exchange_id INTO w_exchange_id
        FROM exchanges
        OFFSET floor(random() * (SELECT count(*) FROM exchanges))
        LIMIT 1;


        INSERT INTO stocks (exchange_id, symbol, company_name) VALUES
(w_exchange_id, w_symbol, w_company_name);
    END LOOP;
END;
$$;
/


CREATE OR REPLACE PROCEDURE delete_all_data()
LANGUAGE plpgsql
AS $$
```

```
BEGIN
    DELETE FROM prices        cascade;
    DELETE FROM stocks        cascade;
    DELETE FROM exchanges     cascade;
END;
$$;
/
```