VIŠJA STROKOVNA ŠOLA ACADEMIA

MARIBOR

Primerjava zmogljivosti podatkovnih baz PostgreSQL in Redis za analizo podatkov s pomočjo umetne inteligence

Kandidat: Michel Velkov

Vrsta študija: študent izrednega študija

Študijski program: Sistemsko inženirstvo

Mentor predavatelj: mag. Ervin Schaff

Mentor v podjetju: Peter Metljak, dipl. inž. el.

Lektorica: Mija Čuk, univ. dipl. spl. jez.

Maribor, 2024

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Podpisani Michel Velkov sem avtor diplomskega dela z naslovom Primerjava zmogljivosti podatkovnih baz PostgreSQL in Redis za analizo podatkov s pomočjo umetne inteligence, 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 predloženem delu, 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.

Kranj, oktober 2024 Podpis študenta:

ZAHVALA

Najprej se iskreno zahvaljujem svojemu mentorju iz podjetja gospodu Petru Metljaku za dragoceno podporo in strokovno vodenje skozi celoten potek raziskave. Njegovi nasveti in praktična znanja so bili ključni za uspešno izvedbo tega projekta.

Prav tako se zahvaljujem mentorju na šoli mag. Ervinu Schaffu za vse usmeritve, potrpežljivost in dragocene povratne informacije, ki so mi pomagale pri izpopolnjevanju vsebine in strukture diplomskega dela.

Navsezadnje pa izrekam globoko zahvalo svoji partnerki za neomajno podporo, razumevanje in spodbudo v času študija in pisanja diplomskega dela. Brez njene pomoči in motivacije tega ne bi zmogel.

POVZETEK

V diplomskem delu smo analizirali zmogljivosti dveh priljubljenih podatkovnih baz, PostgreSQL in Redis, v kontekstu umetne inteligence in obdelave podatkov. Cilj raziskave je bil ugotoviti, katera baza podatkov je bolj primerna za določene naloge, kot so hitro filtriranje podatkov, učenje modelov umetne inteligence ter stabilnost in zanesljivost rezultatov. Postavili smo štiri hipoteze, ki smo jih preverjali z eksperimentalno primerjavo hitrosti poizvedb, časov treniranja modelov in stabilnosti rezultatov.

Rezultati so potrdili, da je Redis zelo hiter pri enostavnih poizvedbah, kot so lookup operacije, zahvaljujoč arhitekturi, ki temelji na pomnilniškem shranjevanju podatkov. Vendar pa ni bil hitrejši pri kompleksnih filtrirnih poizvedbah, kot je trdila hipoteza 2. PostgreSQL je pokazal boljšo zmogljivost pri obdelavi zapletenih poizvedb, kjer je bil zaradi napredne optimizacije in uporabe indeksov čas izvedbe bistveno krajši. To kaže, da je PostgreSQL bolj primeren za scenarije, kjer je potrebno filtriranje večjega števila podatkov s kompleksnimi pogoji.

Pri učenju modelov umetne inteligence je Redis pokazal prednost pri enostavnejših modelih, kot je Ridge regresija, saj je hitrost treniranja nekoliko presegla PostgreSQL. Pri bolj zapletenih modelih, kot sta naključni gozd in gradientno ojačenje, pa sta se PostgreSQL in Redis izkazala za enakovredna, kar nasprotuje hipotezi 4 o večji stabilnosti PostgreSQL.

Raziskava je prav tako pokazala, da PostgreSQL uživa večjo podporo skupnosti in večje povpraševanje na trgu dela, medtem ko Redis vodi pri odprtokodnih projektih na GitHubu. Na podlagi teh ugotovitev priporočamo PostgreSQL za aplikacije, ki zahtevajo obsežno obdelavo podatkov, in Redis za aplikacije, kjer je hitrost ključnega pomena.

Poleg teh ključnih ugotovitev smo proučili tudi vpliv izbire podatkovne baze na celotno arhitekturo aplikacij in njeno skalabilnost. Ugotovili smo, da PostgreSQL nudi boljše možnosti za skaliranje v okolju, kjer se zahteve po podatkih dinamično spreminjajo, medtem ko Redis omogoča enostavnejšo integracijo v realnočasovne sisteme zaradi svoje izjemne hitrosti. Ti dodatni vpogledi omogočajo bolj informirano odločitev pri izbiri podatkovne baze glede na specifične potrebe projekta ter bodo prispevali k večji učinkovitosti in zanesljivosti končnih rešitev.

Ključne besede: *PostgreSQL, Redis, umetna inteligenca, podatkovna baza, strojno učenje.*

ABSTRACT

Comparison of the Performance of PostgreSQL and Redis Databases for Data Analysis Using Artificial Intelligence

In this thesis, we analyzed the performance of two popular databases, PostgreSQL and Redis, in the context of artificial intelligence and data processing. The aim of the research was to determine which database is better suited for specific tasks such as fast data filtering, training AI models, and ensuring stability and reliability of results. We formulated four hypotheses, which were tested through experimental comparisons of query speed, model training times, and result stability.

The results confirmed that Redis excels at simple queries, such as lookup operations, thanks to its in-memory data storage architecture. However, it was not faster for complex data filtering queries, contrary to the H2 hypothesis. PostgreSQL demonstrated better performance in handling complex queries, where its advanced query optimization and indexing capabilities resulted in significantly shorter execution times. This indicates that PostgreSQL is better suited for scenarios where large amounts of data need to be filtered with complex conditions.

When it comes to training AI models, Redis showed an advantage with simpler models like Ridge regression, as its training speed slightly outpaced PostgreSQL. For more complex models such as random forest and gradient boosting, both PostgreSQL and Redis proved to be equally effective, contradicting the H4 hypothesis, which claimed PostgreSQL's superior stability.

The research also demonstrated that PostgreSQL enjoys greater community support and higher demand in the job market, while Redis leads in open-source projects on GitHub. Based on these findings, we recommend PostgreSQL for applications that require extensive data processing and Redis for scenarios where speed is of critical importance.

In addition to these key findings, we examined the impact of the database choice on overall application architecture and scalability. We found that PostgreSQL offers better scalability options in environments where data requirements dynamically change, while Redis facilitates easier integration into real-time systems due to its exceptional speed. These additional insights allow for a more informed decision when selecting a database based on specific project needs and will contribute to greater efficiency and reliability of the final solutions.

Keywords: *PostgreSQL, Redis, artificial intelligence, database, machine learning*

KAZALO VSEBINE

1 UVOD 9

1.1 Opis področja in opredelitev problema 9
1.2 Namen, cilji in osnovne trditve 9
1.3 Predpostavke in omejitve 10
1.4 Uporabljene raziskovalne metode 10

2 PREGLED LITERATURE 11

2.1 Umetna inteligenca in njena uporaba 11
2.1.1 Zgodovinski razvoj umetne inteligence 11
2.1.2 Temeljni koncepti umetne inteligence 13
2.1.3 Praktične aplikacije umetne inteligence 16
2.1.4 Vloga podatkovnih baz pri umetni inteligenci 17
2.2 Pregled podatkovnih baz SQL in NoSQL 21
2.2.1 Uvod v podatkovne baze 21
2.2.2 Karakteristike SQL podatkovnih baz 25
2.2.3 Karakteristike NoSQL podatkovnih baz 29
2.2.4 Primeri uporabe SQL in NoSQL baz 34
2.3 Primerjalna analiza PostgreSQL in Redis 38
2.3.1 Uvod v PostgreSQL 38
2.3.2 Uvod v Redis 42
2.3.3 Primerjava arhitektur PostgreSQL in Redis 46
2.3.4 Primerjava zmogljivosti 51
2.3.5 Podpora in skupnost uporabnikov 56
2.3.6 Uporaba PostgreSQL in Redis v umetni inteligenci 65

3 METODOLOGIJA RAZISKOVANJA 71

3.1 Priprava virtualnih okolij in baze podatkov 71
3.2 Zbiranje podatkov o nogometnih igralcih 77
3.2.1 Priprava CSV datotek 77
3.2.2 Priprava podatkov za uvoz 79
3.2.3 Redis Uvoz 81
3.2.4 Uvoz podatkov v PostgreSQL in Redis 82
3.3 Implementacija modelov umetne inteligence 82
3.3.1 Priprava podatkov za modeliranje 82
3.3.2 Izbor modelov umetne inteligence 83

3.3.3 Implementacija učenja modelov ... 85
3.3.4 Učinkovitost in skaliranje ... 86
3.4 METODE PRIMERJAVE IN OCENJEVANJA ... 86
3.4.1 Razdelitev podatkov na učne in testne množice ... 86
3.4.2 Uporaba meril zmogljivosti ... 87
3.4.3 Časovna analiza ... 88
3.4.4 Optimizacija modelov in iskanje hiperparametrov ... 89
3.4.5 Primerjava metod med PostgreSQL in Redis ... 90
4 EKSPERIMENTALNA PRIMERJAVA POSTGRESQL IN REDIS ... 91
4.1 HITROST POIZVEDB ... 91
4.2 ZMOGLJIVOST MODELOV UMETNE INTELIGENCE ... 92
4.3 STABILNOST IN ZANESLJIVOST REZULTATOV ... 94
4.4 PORABA VIROV IN STROŠKI UPORABE ... 95
4.4.1 Poraba CPU-ja, RAM-a in časovna učinkovitost ... 96
4.4.2 Stroški uporabe ... 97
5 REZULTATI IN ANALIZA ... 99
5.1 ANALIZA HITROSTI POIZVEDB ... 99
5.2 ANALIZA ZMOGLJIVOSTI MODELOV UMETNE INTELIGENCE ... 100
5.3 PRIMERJAVA STABILNOSTI REZULTATOV ... 101
6 RAZPRAVA ... 104
6.1 POMEN REZULTATOV ... 104
6.2 OMEJITVE RAZISKAVE ... 105
6.3 MOŽNOSTI ZA NADALJNJE RAZISKAVE ... 106
7 SKLEP ... 108
8 VIRI, LITERATURA ... 111
9 PRILOGE ... 113

KAZALO SLIK

SLIKA 1: VIRTUALNI NAPRAVI V ORACLE VM ... 71
SLIKA 2: KONFIGURACIJA STROJNE OPREME DEBIAN_POSTGRESQL ... 72
SLIKA 3: KONFIGURACIJA STROJNE OPREME DEBIAN_REDIS ... 73
SLIKA 4: POSTGRESQL PORT FORWARDING PRAVILA ... 74
SLIKA 5: REDIS POST FORWARDING PRAVILA ... 74
SLIKA 6: KONFIGURACIJA POVEZAVE POSTGRESQL V DATAGRIP ... 76

Slika 7: Konfiguracija povezave Redis v Datagrip 76
Slika 8: Diagram PostgreSQL baze v Datagrip 80
Slika 9: Prikaz strukture hash ključev v Redis bazi 81

KAZALO TABEL

Tabela 1: Lestvica priljubljenosti baz podatkov na DB-Engines 63
Tabela 2: Povprečni čas izvajanja "lookup" poizvedb v PostgreSQL in Redis 91
Tabela 3: Povprečni čas izvajanja filtrirnih poizvedb v PostgreSQL in Redis 92
Tabela 4: Povprečni čas treniranja modelov umetne inteligence v PostgreSQL in Redis 93
Tabela 5: Primerjalna tabela stroškov 97

KAZALO GRAFOV

Graf 1: Primerjava števila Stack Overflow vprašanj za PostgreSQL in Redis 61
Graf 2: Primerjava priljubljenosti na GitHubu 62
Graf 3: Število objavljenih delovnih mest za obe bazi na LinkedIn glede na regijo 64
Graf 4: Zmogljivost modelov umetne inteligence v PostgreSQL 93
Graf 5: Zmogljivost modelov umetne inteligence v Redis 94
Graf 6: Prikaz variance v rezultatih za naključni gozd 95

1 UVOD

1.1 Opis področja in opredelitev problema

V zadnjem desetletju je umetna inteligenca (AI) doživela izjemen napredek in je postala ključna komponenta številnih industrijskih in akademskih področij. Učinkovitost algoritmov umetne inteligence je tesno povezana s podatkovnimi bazami, iz katerih pridobivajo podatke za učenje in napovedovanje. Podatkovne baze omogočajo shranjevanje, urejanje in hitro pridobivanje velikih količin podatkov, kar je bistvenega pomena za uspešno delovanje AI sistemov. Dve pogosto uporabljeni podatkovni bazi za te namene sta PostgreSQL, ki je zmogljiva SQL baza podatkov z bogato funkcionalnostjo, in Redis, hitra NoSQL baza, zasnovana za delovanje v pomnilniku. Kljub temu da imata obe bazi podatkov svoje prednosti, ostaja odprto vprašanje, katera izmed njiju je bolj primerna za uporabo v kontekstu umetne inteligence, še posebej, ko gre za specifične naloge, kot so hitro filtriranje podatkov, učenje modelov in stabilnost rezultatov. Ta raziskava bo prispevala k boljšemu razumevanju zmogljivosti obeh podatkovnih baz in njihovi uporabi v realnih scenarijih umetne inteligence.

1.2 Namen, cilji in osnovne trditve

Namen diplomskega dela je primerjati zmogljivosti PostgreSQL in Redisa v kontekstu uporabe umetne inteligence. Cilj je ugotoviti, kako različne arhitekture podatkovnih baz vplivajo na učinkovitost in natančnost modelov umetne inteligence ter kateri sistem zagotavlja boljše rezultate glede na specifične kriterije, kot so hitrost poizvedb, čas učenja modelov, stabilnost in zanesljivost rezultatov.

Na podlagi tega so bile oblikovane naslednje hipoteze:

- **H1:** PostgreSQL ima večjo podporo in širšo skupnost uporabnikov v primerjavi z Redisom.
- **H2:** Poizvedbe za filtriranje podatkov so hitrejše v podatkovni bazi Redis kot v PostgreSQL.
- **H3:** Učenje modelov umetne inteligence je hitrejše pri uporabi Redisa kot pri PostgreSQL.
- **H4:** Random Forest model zagotavlja stabilnejše in bolj zanesljive rezultate pri uporabi PostgreSQL kot Redisa.

1.3 Predpostavke in omejitve

Pri izvedbi te raziskave se predpostavlja, da so vsi uporabljeni podatki pravilni in popolni, da se virtualna okolja, kjer delujejo podatkovne baze, ne razlikujejo v strojni opremi ali konfiguracijah, ki bi lahko vplivale na rezultate. Prav tako se predpostavlja, da so uporabljeni algoritmi za učenje umetne inteligence pravilno implementirani in optimizirani na obeh podatkovnih bazah.

Omejitve raziskave vključujejo dejstvo, da bodo testi izvedeni na omejenem številu modelov umetne inteligence in z uporabo specifičnega nabora podatkov, kar pomeni, da rezultati morda ne bodo neposredno prenosljivi na vse možne scenarije. Prav tako so raziskave omejene na določene različice PostgreSQL in Redisa, kar pomeni, da bi lahko več različic ali konfiguracij privedlo do različnih rezultatov.

1.4 Uporabljene raziskovalne metode

Raziskava temelji na primerjalni eksperimentalni metodi, kjer bosta podatkovni bazi PostgreSQL in Redis preizkušeni glede na različne parametre, pomembne za uporabo v umetni inteligenci. Eksperimenti bodo izvedeni na dveh ločenih virtualnih strojih, kjer bodo podatki o nogometnih igralcih uvoženi v obe bazi. Nato bodo implementirani različni modeli umetne inteligence, kot so Ridge Regression, Decision Tree, Random Forest in Gradient Boosting, na obeh podatkovnih bazah, rezultati pa bodo primerjani glede na čas učenja, natančnost (MSE, MAE, R^2), hitrost poizvedb in stabilnost rezultatov. Pridobljeni rezultati bodo nato analizirani in interpretirani, da se ugotovi, katera baza je bolj primerna za specifične naloge v okviru umetne inteligence.

Poleg eksperimentalne metode so bile uporabljene tudi druge raziskovalne metode, kot sta pregled in študij literature, lastnih zapiskov in drugih virov. Znanje, pridobljeno iz literature, smo uporabili v nadaljnji raziskavi in izdelali povzetke v poglavju 2, kar nam je omogočilo temeljito razumevanje obstoječih rešitev in njihovega vpliva na učinkovitost podatkovnih baz pri uporabi umetne inteligence.

2 PREGLED LITERATURE

2.1 Umetna inteligenca in njena uporaba

2.1.1 Zgodovinski razvoj umetne inteligence

Umetna inteligenca (AI) se je skozi desetletja razvijala iz filozofskega pojma v eno izmed ključnih področij računalništva in tehnologije. Kljub temu da so sodobni dosežki AI nedvomno impresivni, je pomembno razumeti zgodovinski kontekst, iz katerega so se razvili današnji algoritmi in tehnike umetne inteligence.

Začetki umetne inteligence (1940–1950)

Prvi koraki k razvoju umetne inteligence segajo v sredino 20. stoletja, ko so se pojavili prvi računalniki in so raziskovalci začeli proučevati, kako bi lahko stroji simulirali človeško razmišljanje. Leta 1950 je angleški matematik in logik Alan Turing v svojem članku »Computing Machinery and Intelligence« postavil temeljno vprašanje, ki je zaznamovalo področje AI: »Ali lahko stroj razmišlja?« (Goodfellow, Bengio, & Courville, 2016). Turing je predlagal tudi slavni Turingov test, s katerim bi lahko ocenili, ali stroj uspešno simulira človeško inteligenco. Turingov test še danes ostaja pomemben koncept v razpravah o zmožnostih umetne inteligence.

V tem obdobju so raziskovalci začeli razvijati prve algoritemske tehnike, ki bi omogočile strojem izvajanje nalog, ki jih običajno povezujejo s človeško inteligenco, kot so reševanje problemov, igranje iger in dokazovanje teoremov. Leta 1956 je potekala konferenca na Dartmouth College, ki jo mnogi štejejo za formalni začetek raziskovalnega področja umetne inteligence. Na tej konferenci je bil skovan tudi izraz »umetna inteligenca«, ki ga je predlagal John McCarthy.

Zgodnje raziskave in optimizem (1950–1970)

V desetletjih po Dartmouthski konferenci so raziskovalci hitro napredovali pri razvoju AI, vključno z ustvarjanjem programov za igre in univerzalnih orodij za reševanje problemov (Ng, 2018). Prvi uspehi so vključevali ustvarjanje programov za igre, kot je bil program za igranje

dame (angl. draughts) iz leta 1951, ki ga je razvil Christopher Strachey, in General Problem Solver (GPS), ki sta ga leta 1957 razvila Allen Newell in Herbert A. Simon. GPS je bil zasnovan kot univerzalno orodje za reševanje problemov in čeprav so bili njegovi rezultati omejeni, je odprl nova vrata za razmišljanje o tem, kako lahko algoritmi modelirajo človeško reševanje problemov.

V tem obdobju je prevladoval velik optimizem glede zmožnosti umetne inteligence. Mnogi znanstveniki so verjeli, da bodo računalniki v nekaj desetletjih sposobni izvajati skoraj vse dejavnosti, ki jih izvajajo ljudje, vključno s splošno inteligenco. To obdobje je prineslo tudi prve nevronske mreže, pri čemer je pionirsko delo opravil Frank Rosenblatt z razvojem perceptrona leta 1957. Perceptron je bil zasnovan kot model za prepoznavanje vzorcev in je predstavljal zgodnji poskus simulacije delovanja človeških možganov.

Pojav skepticizma in obdobje »zim« umetne inteligence (1970–1990)

Kljub začetnemu optimizmu so se raziskovalci kmalu soočili z omejitvami obstoječe tehnologije in metod. Zgodnji AI sistemi so se izkazali za precej omejene pri reševanju kompleksnejših problemov iz resničnega sveta. Leta 1973 je britanska vlada objavila poročilo »Lighthill Report«, ki je ostro kritiziralo dosežke umetne inteligence, in s tem pripeljala do zmanjšanja financiranja AI raziskav v Združenem kraljestvu. Podobno se je zgodilo tudi v Združenih državah, kjer so bili raziskovalci prisiljeni zmanjšati svoja pričakovanja.

To obdobje, znano kot »zima umetne inteligence«, je zaznamovano s pomanjkanjem napredka in zmanjšanim zanimanjem za raziskave AI. Kljub temu so v tem času potekale pomembne raziskave na področjih, kot so ekspertni sistemi, ki so omogočili simulacijo človeškega strokovnega znanja na ozkih področjih, kot je medicinska diagnostika. Ekspertni sistemi, kot je bil MYCIN, ki je bil razvit za diagnosticiranje bakterijskih okužb, so kljub omejenim zmožnostim pokazali potencial AI za specializirane naloge.

Ponovno prebujenje in moderni razvoj (1990–danes)

Z napredkom v računalniški strojni opremi in razvojem novih algoritmov se je zanimanje za umetno inteligenco v devetdesetih letih ponovno povečalo. Pojav strojnega učenja, še posebej nevronskih mrež, je dal nov zagon raziskavam AI. Uspehi, kot je zmaga računalniškega

programa IBM Deep Blue proti svetovnemu šahovskemu prvaku Garryju Kasparovu leta 1997, so pokazali, da AI postaja vedno bolj zmogljiv in sposoben reševanja zapletenih problemov.

V zadnjih dveh desetletjih je umetna inteligenca doživela izjemen napredek, še posebej z razvojem globokega učenja, ki temelji na večplastnih nevronskih mrežah. Današnji AI sistemi so sposobni prepoznati vzorce v velikih količinah podatkov, kar je omogočilo napredek na področjih, kot so prepoznavanje govora, računalniški vid, avtonomna vozila in personalizirana medicina. Poleg tega so odprtokodne knjižnice, kot sta TensorFlow in PyTorch, omogočile širši dostop do naprednih AI orodij, kar je spodbudilo nadaljnje raziskave in inovacije.

Razvoj umetne inteligence še naprej sledi hitremu tempu, saj raziskovalci iščejo načine za izboljšanje zmogljivosti AI sistemov, zmanjšanje njihovih pomanjkljivosti in širšo uporabo v novih področjih. Kljub velikim uspehom pa ostajajo odprta vprašanja, kot so etika, varnost in vpliv AI na družbo, kar zagotavlja, da bo razvoj umetne inteligence še naprej ena izmed najbolj dinamičnih in pomembnih raziskovalnih tem v prihodnosti.

2.1.2 Temeljni koncepti umetne inteligence

Umetna inteligenca (AI) je široko področje raziskav in aplikacij, ki se ukvarja z ustvarjanjem sistemov, ki lahko izvajajo naloge, ki zahtevajo človeško inteligenco. Da bi bolje razumeli delovanje umetne inteligence in njeno široko uporabnost, je ključno razumevanje temeljnih konceptov, na katerih temelji. Ti koncepti vključujejo strojno učenje, globoko učenje, nadzorovano in nenadzorovano učenje ter nekatere specializirane metode, kot so nevronske mreže, odločitvena drevesa in okrepljeno učenje. Vsak izmed teh konceptov ima specifično vlogo pri razvoju in implementaciji umetne inteligence.

Strojno učenje (ML) je osrednji koncept umetne inteligence, ki omogoča računalniškim sistemom, da se učijo iz podatkov in izboljšujejo svojo učinkovitost brez eksplicitnega programiranja. Strojno učenje temelji na algoritmih, ki identificirajo vzorce v podatkih in jih uporabijo za napovedovanje ali odločanje. Strojno učenje se deli na tri glavne kategorije: nadzorovano učenje, nenadzorovano učenje in okrepljeno učenje.

Nadzorovano učenje je najpogostejši pristop, kjer model uči iz označenih podatkov, kar pomeni, da je za vsak vhodni podatek podana tudi pravilna izhodna vrednost. Algoritem se uči tako, da poskuša minimizirati napake med napovedmi in dejanskimi vrednostmi. Klasični primeri nadzorovanega učenja so klasifikacija (npr. prepoznavanje slik) in regresija (npr. napovedovanje cen nepremičnin).

Nenadzorovano učenje je druga oblika strojnega učenja, kjer model nima dostopa do označenih podatkov. Namesto tega algoritem poskuša najti skrite strukture ali vzorce v vhodnih podatkih. Tipični primeri nenadzorovanega učenja vključujejo gručenje (angl. clustering), kjer se podatki razdelijo v skupine glede na podobnost, in dimenzionalno zmanjševanje, ki se uporablja za poenostavitev podatkov med ohranjanjem pomembnih značilnosti.

Okrepljeno učenje (angl. Reinforcement Learning, RL) je tretji pomemben pristop, kjer se agent uči prek interakcij z okoljem, tako da maksimizira nagrado. Namesto statičnega nabora podatkov agent sprejema odločitve na podlagi povratnih informacij iz okolja, pri čemer poskuša najti optimalno strategijo za doseganje ciljev. Okrepljeno učenje se pogosto uporablja v igrah, robotiki in nadzornih sistemih.

Globoko učenje (DL) je posebna vrsta strojnega učenja, ki temelji na nevronskih mrežah z več plastmi (globokih nevronskih mrež). Glavna prednost globokega učenja je sposobnost samodejnega učenja kompleksnih predstavitev podatkov, kar omogoča obdelavo zelo velikega obsega surovih podatkov, kot so slike, zvok ali naravni jezik.

Nevronske mreže so osnovni gradnik globokega učenja. Sestavljene so iz več plasti nevronov, pri čemer vsaka plast izvaja linearne ali nelinearne transformacije podatkov. Z vsako dodatno plastjo postaja model bolj zmožen zajeti kompleksne vzorce in značilnosti podatkov. Klasična uporaba globokega učenja vključuje prepoznavanje obrazov, prevajanje besedila, samovozeča vozila in še veliko več.

Med ključnimi koncepti globokega učenja so konvolucijske nevronske mreže (angl. Convolutional Neural Networks, CNN), ki so posebej učinkovite za obdelavo slikovnih podatkov, in rekurentne nevronske mreže (angl. Recurrent Neural Networks, RNN), ki so prilagojene za delo s sekvenčnimi podatki, kot so besedila in časovne vrste. Posebej RNN,

skupaj z naprednimi različicami, kot je Long Short-Term Memory (LSTM), so se izkazali za izjemno uporabne pri nalogah, kot je napovedovanje zaporednih dogodkov.

Algoritmi in metode v umetni inteligenci

Različni algoritmi in metode so ključni za razvoj učinkovitih AI sistemov. Nekateri izmed najpogostejših so:

- **Odločitvena drevesa (angl. Decision Trees)** so strukture, ki modelirajo odločitve in njihove možne posledice, vključno z rezultati, stroški in koristmi. Gre za priljubljen pristop v nadzorovanem učenju, saj je enostaven za razlago in vizualizacijo.
- **Random Forest** je izboljšana različica odločitvenih dreves, kjer se zgradi več dreves na različnih vzorcih podatkov, končni rezultat pa je povprečje napovedi vseh dreves. To zmanjšuje možnost prenaučenja in povečuje natančnost napovedi.
- **Podporno vektorski stroji (angl. Support Vector Machines, SVM)** so zmogljivo orodje za klasifikacijo, ki se osredotoča na iskanje optimalne meje med različnimi razredi podatkov. SVM je zelo priljubljen pri obdelavi kompleksnih in večdimenzionalnih podatkov.
- **K-means** je priljubljen algoritem za gručenje, kjer se podatki razdelijo v K skupin, tako da je vsaka skupina čim bolj homogena znotraj sebe in čim bolj heterogena glede na druge skupine. Uporablja se predvsem v nenadzorovanem učenju.
- **Okrepljeno učenje (angl. Reinforcement Learning, RL)** je metoda učenja prek nagrad in kazni. Gre za obliko učenja, kjer agent poskuša maksimizirati svojo nagrado z optimizacijo svoje politike delovanja. Ta metoda se je izkazala za zelo uspešno v aplikacijah, kot je igranje iger, kjer je agent usposobljen za iskanje optimalne strategije.

Vloga podatkov v umetni inteligenci

Podatki igrajo osrednjo vlogo v vseh metodah umetne inteligence. Za vsako vrsto učenja so potrebni različni tipi podatkov, od označenih primerov za nadzorovano učenje do neoznačenih podatkov za nenadzorovano učenje. Kakovost in količina podatkov močno vplivata na uspešnost AI modelov. Pri globokem učenju, na primer, modeli potrebujejo ogromne količine podatkov, da se lahko naučijo kompleksnih vzorcev, medtem ko pri strojno podprtem učenju kakovostne oznake podatkov vplivajo na natančnost klasifikatorjev.

Etični in praktični vidiki

Z rastjo umetne inteligence postajajo vse pomembnejša tudi etična vprašanja, povezana z uporabo teh tehnologij. To vključuje vprašanja zasebnosti, pristranskosti v podatkih, avtomatizacije delovnih mest in odločitev, ki jih sprejemajo AI sistemi. Prav tako je pomembno, da se AI razvija na način, ki je koristen in pravičen za vse družbene skupine.

Razumevanje temeljnih konceptov umetne inteligence omogoča lažje razumevanje njenih zmožnosti, omejitev in potenciala. Ti koncepti ne le tvorijo osnovo za raziskave in razvoj na področju AI, temveč tudi služijo kot izhodišče za aplikacije, ki spreminjajo svet. S pravilno implementacijo in uporabo umetne inteligence lahko te tehnologije bistveno izboljšajo različne aspekte življenja in dela, hkrati pa se moramo zavedati odgovornosti, ki jo prinaša njihova uporaba.

2.1.3 Praktične aplikacije umetne inteligence

Umetna inteligenca (AI) je postala nepogrešljiv del številnih industrij in vsakodnevnega življenja, saj omogoča reševanje kompleksnih problemov, optimizacijo procesov ter ustvarjanje novih storitev in izdelkov. Nekatera glavna področja uporabe umetne inteligence vključujejo:

- umetno inteligenco v medicini,
- umetno inteligenco v financah,
- umetno inteligenco v proizvodnji in industriji,
- umetno inteligenco v transportu in logistiki,
- umetno inteligenco v e-trgovini,
- umetno inteligenco v zabavni industriji.

Pri vseh teh aplikacijah je treba shranjevati in obdelovati različne podatke. Ker cilj diplomskega dela ni prikaz vseh možnosti uporabe umetne inteligence, bomo podali podrobnejši pregled le za umetno inteligenco v e-trgovini in medicini.

Umetna inteligenca v e-trgovini

E-trgovina je eno izmed področij, kjer je umetna inteligenca prinesla številne inovacije, ki izboljšujejo nakupovalno izkušnjo, personalizacijo in upravljanje zalog.

- **Personalizacija ponudb:** AI omogoča prilagajanje ponudb in priporočil, ki temeljijo na analiziranju vedenja uporabnikov, zgodovine nakupov in demografskih podatkov. To omogoča personalizirano izkušnjo za vsakega uporabnika.
- **Chatboti in virtualni asistenti:** E-trgovinska podjetja uporabljajo chatbote, podprte z AI, za samodejno odgovarjanje na vprašanja strank, pomoč pri nakupu in reševanje težav.
- **Optimizacija zalog:** Algoritmi umetne inteligence analizirajo povpraševanje in zaloge, kar omogoča boljše načrtovanje, zmanjšanje stroškov in skrajšanje časa dostave.

Umetna inteligenca v medicini

V medicini umetna inteligenca ponuja vrsto aplikacij, ki izboljšujejo diagnostiko, zdravljenje in oskrbo pacientov.

- **Diagnostika z uporabo AI:** AI se uporablja za avtomatizacijo diagnostike z identifikacijo vzorcev in nepravilnosti v medicinskih slikah, kar pomaga radiologom in drugim zdravstvenim delavcem pri zgodnjem odkrivanju bolezni.
- **Napovedovanje izidov zdravljenja:** Algoritmi umetne inteligence lahko analizirajo zgodovinske podatke pacientov in ugotovijo dejavnike, ki vplivajo na uspešnost določenih terapevtskih pristopov, s čimer omogočajo personalizirano zdravljenje.
- **Telemedicina in robotska kirurgija:** Umetna inteligenca omogoča napredne oblike telemedicine in robotsko asistirano kirurgijo, kjer sistemi, podprti z AI, omogočajo natančnejše in manj invazivne kirurške posege.

2.1.4 Vloga podatkovnih baz pri umetni inteligenci

Podatkovne baze igrajo ključno vlogo v razvoju in delovanju umetne inteligence (AI), saj zagotavljajo osnovo za shranjevanje, upravljanje in dostop do podatkov, ki so potrebni za učenje in izvajanje AI modelov. V dobi, ko se generira in obdeluje več podatkov kot kdaj koli

prej, je učinkovita uporaba podatkovnih baz postala bistvenega pomena za uspeh AI sistemov. To podpoglavje bo podrobno raziskalo, kako podatkovne baze podpirajo umetno inteligenco, vključno z njihovimi različnimi funkcijami, izzivi in primeri uporabe v realnih scenarijih.

Shranjevanje in upravljanje velikih podatkov

Eden glavnih izzivov pri umetni inteligenci je obdelava velikih količin podatkov, ki jih je treba shranjevati in organizirati na način, ki omogoča učinkovito pridobivanje informacij. Podatkovne baze so orodje, ki omogoča to organizacijo in dostopnost, ne glede na vrsto podatkov, s katerimi se ukvarjamo – bodisi strukturirani, nestrukturirani, polstrukturirani ali časovno odvisni podatki.

Relacijske podatkovne baze (SQL), kot je PostgreSQL, so tradicionalno orodje za shranjevanje strukturiranih podatkov. Te baze omogočajo zapletene poizvedbe in manipulacijo podatkov z uporabo jezika SQL (angl. Structured Query Language), kar je še posebej koristno, ko je treba podatke natančno razvrstiti, filtrirati ali združiti iz več tabel. Relacijske podatkovne baze so idealne za primere, kjer so podatki jasno definirani in imajo stalno strukturo, kot so npr. finančni podatki, transakcije ali zapisi o strankah.

NoSQL podatkovne baze, kot je Redis, pa ponujajo večjo fleksibilnost pri delu z nestrukturiranimi ali polstrukturiranimi podatki. Te baze so zasnovane tako, da se lažje prilagajajo spremembam v strukturi podatkov in omogočajo hitrejše poizvedbe ter višjo zmogljivost pri obdelavi velikih količin podatkov v realnem času. To je še posebej pomembno v aplikacijah, kjer so podatki heterogeni ali hitro spreminjajoči, kot so podatki iz družbenih omrežij, dnevniške datoteke ali podatki o uporabniških dejavnostih.

Podatkovne baze kot vir za učenje modelov umetne inteligence

Podatkovne baze so ključnega pomena za pripravo podatkovnih zbirk, ki se uporabljajo pri učenju modelov umetne inteligence. Učenje AI modelov zahteva veliko količino podatkov, ki morajo biti pravilno označeni, očiščeni in organizirani. Podatkovne baze omogočajo zbiranje, filtriranje in agregacijo podatkov iz različnih virov, kar pripomore k ustvarjanju kakovostnih podatkovnih nizov za učenje.

V praksi se podatkovne baze uporabljajo za ekstrakcijo ustreznih atributov, ki jih model potrebuje za učenje. Na primer, v primeru napovedovanja cen nepremičnin bi podatkovna baza

vsebovala različne atribute, kot so velikost, lokacija, starost nepremičnine, bližina javnih storitev ipd. Ti atributi so nato pripravljeni in pretvorjeni v obliko, primerno za vnos v AI model. Poleg tega omogočajo tudi upravljanje metapodatkov, kot so oznake podatkov (npr. ali so podatki veljavni, kateri podatki manjkajo), kar je pomembno za zagotavljanje natančnosti modela.

Učinkovitost in zmogljivost podatkovnih baz pri poizvedbah in obdelavi podatkov

Za umetno inteligenco, zlasti pri obdelavi velikih količin podatkov ali pri delu v realnem času, je ključnega pomena učinkovitost podatkovnih baz pri izvedbi poizvedb. To vključuje čas, potreben za pridobivanje podatkov, ko AI model zahteva določene informacije, in hitrost, s katero lahko baza obdeluje te poizvedbe, ne da bi upočasnila delovanje sistema.

Indeksiranje in optimizacija poizvedb sta dve ključni funkcionalnosti, ki ju podatkovne baze ponujajo za izboljšanje učinkovitosti. Z učinkovitim indeksiranjem lahko baze podatkov bistveno zmanjšajo čas iskanja podatkov, kar je kritično za aplikacije, kjer so potrebni hitri odgovori, kot so sistemi za priporočanje ali realnočasovna analitika. Poleg tega podatkovne baze omogočajo optimizacijo poizvedb, s čimer se zmanjša obremenitev sistema in poveča zmogljivost, kar omogoča, da AI modeli delujejo na največji možni zmogljivosti.

Pri NoSQL bazah, kot je Redis, je ena ključnih prednosti hitro dostopanje do podatkov v pomnilniku, kar omogoča skoraj takojšnjo obdelavo podatkov (Redis, Docs, 2024). To je posebej pomembno pri aplikacijah, kot so sistemi za spremljanje in opozarjanje, kjer hitrost dostopa do podatkov neposredno vpliva na učinkovitost celotnega sistema.

Integracija podatkovnih baz z umetno inteligenco

Za učinkovito delovanje umetne inteligence je potrebna tesna integracija med podatkovnimi bazami in AI sistemi. Podatkovne baze morajo podpirati ne le shranjevanje podatkov, ampak tudi učinkovito pridobivanje, transformacijo in pošiljanje podatkov v AI modele ter prejemanje in shranjevanje rezultatov iz teh modelov.

Procesi ETL (angl. Extract, Transform, Load) so kritični v tej integraciji. ETL omogoča pridobivanje podatkov iz različnih virov, njihovo čiščenje, pretvorbo in nalaganje v podatkovne baze, kjer jih lahko uporabljajo AI sistemi. Učinkoviti procesi ETL so bistveni za zagotavljanje, da so podatki, ki jih uporabljajo AI modeli, pravilni, konsistentni in pripravljeni za uporabo.

Poleg tega se vedno več uporablja pristop baze podatkov v živo (angl. real-time databases), kjer se podatki obdelujejo sproti, kar je bistveno za aplikacije, kot so avtonomna vozila, borzno trgovanje ali pametna mesta. V teh primerih morajo biti podatkovne baze sposobne obdelovati velike količine podatkov v realnem času ter zagotavljati natančne in pravočasne informacije AI sistemom.

Podpora za obsežno analiziranje in vizualizacijo podatkov

Poleg shranjevanja in upravljanja podatkov morajo podatkovne baze podpirati tudi obsežno analiziranje podatkov in vizualizacijo, kar omogoča globlje razumevanje podatkov in njihovega vpliva na AI modele. Analitična orodja, ki so pogosto integrirana z bazami podatkov, omogočajo raziskovalcem in inženirjem, da pridobijo vpogled v podatkovne vzorce, ugotovijo korelacije in odkrijejo pomembne trende, ki jih nato lahko uporabijo za optimizacijo AI modelov.

SQL analitične funkcije omogočajo izvajanje kompleksnih poizvedb, kot so združevanje podatkov, izračuni povprečij, standardnih odklonov in drugih statističnih metrik, ki so pomembne za razumevanje distribucije podatkov. NoSQL baze pa pogosto podpirajo MapReduce modele, ki omogočajo paralelno obdelavo velikih podatkovnih nizov, kar je še posebej uporabno za analizo obsežnih podatkovnih zbirk.

Vizualizacija podatkov je prav tako ključni del analitičnega procesa. Orodja, kot so Grafana (Grafana, 2024), Tableau ali Power BI, ki se lahko povezujejo s podatkovnimi bazami, omogočajo ustvarjanje vizualnih prikazov podatkov, ki pomagajo razumeti kompleksne odnose in rezultate AI modelov. S pomočjo vizualizacije je mogoče lažje razložiti rezultate, ugotoviti anomalije in se odločiti za nadaljnje korake v raziskavah ali optimizaciji modelov.

Izzivi in prihodnji razvoj

Čeprav podatkovne baze ponujajo številne prednosti za umetno inteligenco, se soočajo tudi z izzivi, ki so povezani z obsegom in kompleksnostjo podatkov, varnostjo, zasebnostjo in integracijo različnih tehnologij. Z naraščajočimi količinami podatkov se povečuje potreba po hitrih in skalabilnih rešitvah, ki bodo lahko obvladovale vedno bolj zahtevne naloge.

Prihodnji razvoj podatkovnih baz bo verjetno usmerjen v še tesnejšo integracijo z AI sistemi, večjo avtomatizacijo upravljanja podatkov in izboljšanje zmogljivosti za obdelavo podatkov v

realnem času. Razvoj hibridnih sistemov, ki združujejo prednosti SQL in NoSQL baz, ter napredek v tehnologijah, kot so in-memory computing in distribuirani sistemi, bo igral ključno vlogo pri podpori prihodnjih potreb umetne inteligence.

2.2 Pregled podatkovnih baz SQL in NoSQL

2.2.1 Uvod v podatkovne baze

Podatkovne baze so temelj sodobne informatike in informacijskih sistemov (Wikipedia, 2024), saj omogočajo shranjevanje, upravljanje in dostop do velikih količin podatkov. Razvoj podatkovnih baz je bil ključen za digitalno revolucijo in nadaljnji napredek na številnih področjih, vključno z znanostjo, poslovanjem, zdravjem, izobraževanjem in zabavo. V tem poglavju bomo raziskali osnovne koncepte, zgodovinski razvoj in vrste podatkovnih baz, ki so oblikovale trenutno stanje na tem področju.

Definicija in osnovni koncepti

Podatkovna baza je organizirana zbirka podatkov, ki omogoča shranjevanje, urejanje in pridobivanje informacij na učinkovit način. Ključni cilj podatkovnih baz je omogočiti uporabnikom in aplikacijam hiter in zanesljiv dostop do informacij, ki jih potrebujejo za sprejemanje odločitev ali izvajanje poslovnih procesov.

Podatkovne baze so sestavljene iz tabel (v primeru relacijskih podatkovnih baz) ali drugih struktur, kot so dokumenti, grafi ali ključ-vrednostni pari (v primeru NoSQL podatkovnih baz), ki organizirajo podatke v logične enote. Tabela je sestavljena iz vrstic in stolpcev, kjer vsaka vrstica predstavlja zapis ali entiteto, vsak stolpec pa atribut te entitete. Ta struktura omogoča jasno in dosledno organizacijo podatkov, kar olajša poizvedovanje, posodabljanje in vzdrževanje podatkov.

Pomemben koncept pri podatkovnih bazah je normalizacija, ki se nanaša na proces organiziranja podatkov v bazo tako, da se zmanjšajo podvajanja podatkov in zagotovi njihova integriteta. Normalizacija vključuje razdelitev velikih tabel na manjše, povezanih s tujimi ključi, kar pomaga zmanjšati anomalije, kot so podvajanje podatkov, posodabljanje in brisanje.

Zgodovinski razvoj podatkovnih baz

Razvoj podatkovnih baz se je začel v zgodnjih šestdesetih, ko je digitalizacija podatkov postala nujna zaradi naraščajoče količine informacij, ki jih je bilo treba obdelati in shraniti. Prve podatkovne baze so bile hierarhične in mrežne baze podatkov, ki so bile osnovane na strukturi dreves oziroma grafov. Te baze podatkov so omogočale preprosto shranjevanje in pridobivanje podatkov, vendar so imele omejitve glede prilagodljivosti in skalabilnosti.

Relacijski model, ki ga je v zgodnjih sedemdesetih uvedel Edgar F. Codd, je bil prelomnica v zgodovini podatkovnih baz. Relacijski model je temeljil na matematičnih konceptih relacij in množic, kar je omogočilo bolj fleksibilno in učinkovito shranjevanje podatkov. Podatki v relacijskih podatkovnih bazah so organizirani v tabelah, kjer so odnosi med podatki jasno definirani prek ključev. Ta model je omogočil razvoj poizvedovalnega jezika SQL (angl. Structured Query Language), ki je postal standard za delo z relacijskimi podatkovnimi bazami. SQL je omogočil uporabnikom, da izvajajo kompleksne poizvedbe, manipulirajo podatke in upravljajo shemo baze podatkov.

V osemdesetih in devetdesetih letih so relacijske baze podatkov postale prevladujoč standard v industriji. IBM, Oracle in Microsoft so razvili komercialne sisteme za upravljanje relacijskih baz podatkov (RDBMS), kot so DB2, Oracle Database in Microsoft SQL Server, ki so omogočili podjetjem učinkovito upravljanje velikih količin podatkov.

S prehodom v digitalno dobo in pojavom interneta se je povečala potreba po bolj fleksibilnih in skalabilnih rešitvah za upravljanje podatkov. To je pripeljalo do razvoja NoSQL podatkovnih baz v letih 2000. NoSQL baze podatkov, kot so MongoDB, Cassandra in Redis, so bile zasnovane za obdelavo velikih količin nestrukturiranih podatkov, ki so značilni za aplikacije, kot so spletne platforme, družbena omrežja in analitika velikih podatkov.

Vrste podatkovnih baz

Podatkovne baze lahko razvrstimo v več kategorij glede na njihove značilnosti in namen uporabe. Glavne vrste podatkovnih baz so:

- **Relacijske podatkovne baze (RDBMS):** To so najpogostejše vrste podatkovnih baz, kjer so podatki organizirani v tabelah, ki so medsebojno povezane z relacijami. Relacijske baze podatkov podpirajo transakcije, zagotavljajo integriteto podatkov in omogočajo kompleksne

poizvedbe z uporabo SQL. Primeri RDBMS vključujejo PostgreSQL, MySQL, Oracle Database in Microsoft SQL Server.

- **NoSQL podatkovne baze:** NoSQL baze podatkov so zasnovane za delo z velikimi količinami nestrukturiranih ali polstrukturiranih podatkov. Obstaja več vrst NoSQL baz, vključno z:
  - **Key-Value baze podatkov:** Ti sistemi shranjujejo podatke kot pare ključ-vrednost. Primeri vključujejo Redis in Amazon DynamoDB.
  - **Dokumentne baze podatkov:** Podatki so shranjeni v obliki dokumentov (npr. JSON, BSON). Primeri vključujejo MongoDB in CouchDB.
  - **Grafovne baze podatkov:** Specializirane za shranjevanje in poizvedovanje po podatkih, ki so medsebojno povezani kot vozlišča in povezave v grafu. Primeri vključujejo Neo4j in ArangoDB.
- **Column-Family baze podatkov:** Ti sistemi shranjujejo podatke v stolpčnih družinah, kar omogoča optimizirano obdelavo velikih količin podatkov. Primeri vključujejo Apache Cassandra in HBase.
- **Objektno usmerjene podatkovne baze:** Te baze podatkov kombinirajo koncepte objektno usmerjenega programiranja z bazami podatkov, kjer so podatki shranjeni kot objekti. To omogoča bolj naravno mapiranje podatkovnih struktur znotraj programskih jezikov, ki podpirajo objektno usmerjeno programiranje. Primeri vključujejo db4o in ObjectDB.
- **Distribuirane podatkovne baze:** Distribuirane baze podatkov so razdeljene med več fizičnih ali virtualnih strežnikov. Takšen sistem omogoča večjo odpornost, skalabilnost in hitrost obdelave podatkov, saj lahko več strežnikov hkrati obdeluje poizvedbe. Primeri vključujejo Google Spanner in CockroachDB.
- **Hibridne podatkovne baze:** Hibridne baze podatkov združujejo lastnosti relacijskih in NoSQL baz podatkov, kar omogoča izkoriščanje prednosti obeh pristopov. Te baze so zasnovane za reševanje kompleksnih problemov, kjer je potrebna visoka stopnja prilagodljivosti in zanesljivosti. Primeri vključujejo Oracle NoSQL Database in Azure Cosmos DB.

Ključne funkcije in lastnosti podatkovnih baz

Podatkovne baze zagotavljajo več ključnih funkcij, ki omogočajo učinkovito upravljanje podatkov:

- **ACID lastnosti (angl. Atomicity, Consistency, Isolation, Durability):** To so ključni atributi transakcijskih sistemov, ki zagotavljajo, da so vse operacije znotraj transakcije izvedene pravilno in da se podatki ohranijo tudi ob napakah sistema.
- **Indeksiranje:** Indeksiranje je tehnika, ki omogoča hitrejše iskanje podatkov z ustvarjanjem kazalnikov na lokacijo podatkov v bazi. Učinkovito indeksiranje bistveno izboljša zmogljivost poizvedb.
- **Replikacija:** Replikacija omogoča, da se podatki kopirajo na več strežnikov, kar povečuje zanesljivost sistema in omogoča, da so podatki dostopni tudi v primeru izpada enega od strežnikov.
- **Varnost:** Podatkovne baze vključujejo različne mehanizme za zaščito podatkov, vključno z nadzorom dostopa, šifriranjem podatkov in revizijskimi sledmi, ki beležijo vse spremembe v bazi.
- **Skalabilnost:** Podatkovne baze morajo biti zasnovane tako, da omogočajo enostavno prilagajanje povečanju obsega podatkov ali obdelave. To vključuje tako vertikalno (dodajanje več virov znotraj strežnika) kot horizontalno skaliranje (dodajanje več strežnikov).

Zaključek

Podatkovne baze so temeljni gradnik informacijskih sistemov, saj omogočajo učinkovito shranjevanje, dostopanje in upravljanje podatkov. Razvoj podatkovnih baz je bil ključen za napredek na številnih področjih, in njihove različne vrste in funkcije omogočajo prilagajanje specifičnim potrebam sodobnih aplikacij, vključno z umetno inteligenco, obdelavo velikih podatkov, spletnimi storitvami in še mnogo več. Razumevanje osnovnih konceptov in razvoja podatkovnih baz je ključno za nadaljnjo razpravo o tem, kako te tehnologije podpirajo in omogočajo napredne aplikacije, kot je umetna inteligenca, kar bo podrobneje obravnavano v naslednjih poglavjih.

2.2.2 Karakteristike SQL podatkovnih baz

SQL (angl. Structured Query Language) podatkovne baze so temelj sodobnih informacijskih sistemov in so že desetletja prevladujoči standard za upravljanje strukturiranih podatkov. SQL baze podatkov se uporabljajo v različnih aplikacijah, od poslovnih informacijskih sistemov do spletnih platform, kjer so potrebne zanesljivost, doslednost in zmogljivost pri obdelavi podatkov. V tem podpoglavju bomo podrobno raziskali ključne značilnosti SQL podatkovnih baz, vključno z njihovo arhitekturo, funkcionalnostmi, prednostmi in slabostmi ter njihovim vplivom na razvoj in upravljanje podatkovnih sistemov.

Arhitektura SQL podatkovnih baz

SQL podatkovne baze temeljijo na relacijskem modelu, ki ga je v sedemdesetih letih predstavil Edgar F. Codd. V relacijskih podatkovnih bazah so podatki organizirani v tabelah, kjer vsaka tabela predstavlja množico podatkov, strukturiranih v vrstice (zapisi) in stolpce (atributi). Relacijski model omogoča jasno in dosledno organizacijo podatkov, ki so medsebojno povezani s pomočjo ključev, kot so primarni ključi in tujki.

Primarni ključ je edinstveni identifikator vsakega zapisa v tabeli (Juba, 2019), ki zagotavlja, da ni dveh enakih zapisov. Tuji ključi so atributi, ki ustvarjajo povezave med tabelami, kar omogoča definiranje odnosov med različnimi entitetami v podatkovni bazi. Ta struktura je izjemno učinkovita za organizacijo kompleksnih podatkovnih struktur in zagotavljanje celovitosti podatkov.

Ena od ključnih lastnosti arhitekture SQL podatkovnih baz je podpora za normalizacijo, proces, pri katerem so podatki organizirani v tabele tako, da se minimizirajo podvajanja in zagotovi integriteta podatkov. Normalizacija poteka v več korakih, imenovanih normalne forme, kjer se podatki razdelijo na manjše tabele, povezane s tujimi ključi. Čeprav normalizacija izboljšuje konsistenco podatkov, lahko vodi do bolj zapletenih poizvedb, zato se v praksi pogosto uporablja denormalizacija, kjer so nekatere tabele združene, da se izboljša zmogljivost poizvedb.

Funkcionalnosti SQL podatkovnih baz

SQL podatkovne baze so opremljene z različnimi funkcionalnostmi, ki omogočajo učinkovito upravljanje podatkov. Te funkcionalnosti vključujejo transakcijsko obdelavo, poizvedovanje, manipulacijo podatkov, upravljanje uporabnikov in varnost.

Transakcije so ključna funkcionalnost SQL podatkovnih baz, ki zagotavlja, da se skupina operacij izvaja kot ena enota, ki je bodisi v celoti uspešna bodisi v celoti neuspešna. To zagotavlja, da podatki ostanejo dosledni in celoviti, tudi v primeru napak ali motenj v sistemu. SQL podatkovne baze podpirajo ACID lastnosti (angl. Atomicity, Consistency, Isolation, Durability), ki zagotavljajo zanesljivo in varno izvedbo transakcij (Juba, 2019).

- **Atomarnost (angl. Atomicity):** Vsaka transakcija je »vse ali nič«, kar pomeni, da se vse operacije znotraj transakcije bodisi izvedejo bodisi zavrnejo, če katera od operacij ne uspe.
- **Doslednost (angl. Consistency):** Transakcija pripelje bazo podatkov iz enega konsistentnega stanja v drugo, kar pomeni, da so po zaključku transakcije vsi podatki veljavni glede na določena pravila in omejitve.
- **Izolacija (angl. Isolation):** Izolacija zagotavlja, da so transakcije izolirane druga od druge, kar pomeni, da delne spremembe ene transakcije niso vidne drugim, dokler transakcija ni dokončana.
- **Trpežnost (angl. Durability):** Po uspešnem zaključku transakcije so spremembe trajno zapisane v bazo podatkov, tudi v primeru izpada sistema.

SQL podatkovne baze podpirajo tudi obsežno poizvedovanje in manipulacijo podatkov. SQL je izjemno zmogljiv poizvedovalni jezik, ki omogoča uporabnikom, da izvajajo zapletene poizvedbe, ki vključujejo filtriranje, združevanje, razvrščanje in združevanje podatkov iz več tabel. To omogoča uporabnikom, da pridobijo specifične informacije iz velikega obsega podatkov z visoko stopnjo natančnosti.

Poleg poizvedovanja SQL omogoča tudi manipulacijo podatkov, kot so vstavljanje (angl. INSERT), posodabljanje (angl. UPDATE), brisanje (angl. DELETE) in spreminjanje strukture tabel (angl. ALTER). Te operacije omogočajo fleksibilno upravljanje podatkov in prilagajanje sheme baze podatkov potrebam uporabnikov in aplikacij.

Upravljanje uporabnikov in varnost sta prav tako ključni funkcionalnosti SQL podatkovnih baz. SQL baze podatkov omogočajo natančen nadzor dostopa do podatkov, kjer se lahko pravice dostopa določijo na ravni tabele, stolpca ali celo posamezne poizvedbe. To zagotavlja, da so podatki dostopni samo pooblaščenim uporabnikom in da so zaupni podatki zaščiteni pred nepooblaščenim dostopom. Poleg tega SQL baze podatkov podpirajo šifriranje podatkov, ki zagotavlja dodatno raven varnosti, zlasti pri občutljivih podatkih.

Prednosti SQL podatkovnih baz

SQL podatkovne baze ponujajo številne prednosti, zaradi katerih so še vedno ena izmed najbolj priljubljenih tehnologij za upravljanje podatkov v poslovnem in akademskem okolju.

- **Zanesljivost in doslednost:** Ena od glavnih prednosti SQL podatkovnih baz je njihova zanesljivost in doslednost pri obdelavi podatkov. ACID lastnosti zagotavljajo, da so podatki vedno pravilni in dosledni, tudi v primeru napak ali nepredvidenih dogodkov. To je še posebej pomembno v kritičnih aplikacijah, kot so bančništvo, zdravstvena oskrba in logistika, kjer je napaka v podatkih lahko katastrofalna.
- **Standardizacija:** SQL je standardiziran jezik, kar pomeni, da je podprt na širokem spektru platform in aplikacij. To omogoča razvijalcem, da svoje rešitve prenašajo med različnimi sistemi brez potrebe po obsežnih prilagoditvah. Poleg tega standardizacija poenostavlja učenje in uporabo SQL, saj so osnovni koncepti in sintaksa enaki ne glede na implementacijo baze podatkov.
- **Obsežno poizvedovanje:** SQL omogoča uporabnikom, da izvajajo zelo zapletene poizvedbe, ki vključujejo več tabel in podpoizvedb. Ta zmogljivost je ključna za poslovne analitike in poročanje, kjer so potrebne podrobne informacije, izbrane iz velikih podatkovnih zbirk.
- **Integriteta podatkov:** SQL baze podatkov podpirajo stroga pravila integritete podatkov, kot so tuji ključi, preveritvena pravila (angl. CHECK constraints) in edinstvene omejitve (angl. UNIQUE constraints). Ta pravila zagotavljajo, da so podatki v bazi pravilni in konsistentni, kar je ključnega pomena za ohranjanje kakovosti podatkov.
- **Podpora za transakcije:** Podpora za transakcije omogoča, da se skupine operacij obravnavajo kot enovite enote, kar povečuje zanesljivost sistema in preprečuje, da bi napake

povzročile nepopravljivo škodo na podatkih. To je ključna funkcionalnost v aplikacijah, kjer so podatki kritični, kot so sistemi za obdelavo plačil ali upravljanje zalog.

Slabosti SQL podatkovnih baz

Kljub številnim prednostim imajo SQL podatkovne baze tudi nekatere omejitve, ki lahko vplivajo na njihovo učinkovitost in primernost v določenih aplikacijah.

- **Skalabilnost:** Tradicionalne SQL podatkovne baze so zasnovane za delovanje na enem strežniku, kar lahko omeji njihovo skalabilnost, zlasti ko gre za obdelavo zelo velikih količin podatkov. Čeprav so bile razvite rešitve, kot je sharding, ki omogoča porazdelitev podatkov na več strežnikov, je ta pristop zapleten in lahko povzroči dodatne težave z upravljanjem.
- **Rigidnost sheme:** SQL baze podatkov temeljijo na strogi shemi, kar pomeni, da morajo biti podatki organizirani v vnaprej določeno strukturo tabel in stolpcev. Čeprav to zagotavlja doslednost, je lahko problematično, ko je potrebno hitro prilagajanje ali spreminjanje strukture podatkov, kar je pogost primer v dinamičnih aplikacijah, kot so spletne platforme ali mobilne aplikacije.
- **Učinkovitost pri nestrukturiranih podatkih:** SQL baze podatkov so manj primerne za obdelavo nestrukturiranih podatkov, kot so besedilni dokumenti, slike ali podatki iz družbenih omrežij. NoSQL baze podatkov ponujajo večjo prilagodljivost in zmogljivost pri obdelavi takšnih podatkov, zaradi česar se pogosto uporabljajo v aplikacijah, kjer so podatki manj strukturirani ali se njihova struktura pogosto spreminja.
- **Stroški upravljanja:** Upravljanje SQL podatkovnih baz zahteva visoko stopnjo strokovnega znanja, zlasti pri kompleksnih aplikacijah, kjer so potrebni optimizacija poizvedb, upravljanje indeksov in vzdrževanje integritete podatkov. To lahko poveča stroške razvoja in vzdrževanja aplikacij, zlasti v primerjavi z enostavnejšimi NoSQL rešitvami.

Zaključek

SQL podatkovne baze so temeljni gradnik sodobnih informacijskih sistemov, ki omogočajo učinkovito in zanesljivo upravljanje strukturiranih podatkov. Njihova zanesljivost, standardizacija in zmogljivost pri obdelavi kompleksnih poizvedb jih delajo nepogrešljive v številnih industrijah. Kljub nekaterim omejitvam, kot so rigidnost sheme in izzivi pri

skalabilnosti, ostajajo SQL baze podatkov prva izbira za aplikacije, kjer so ključni dejavniki integriteta, doslednost in zanesljivost podatkov. Razumevanje karakteristik SQL podatkovnih baz je ključno za izbiro prave podatkovne tehnologije glede na specifične potrebe aplikacij, kar je še posebej pomembno v kontekstu integracije umetne inteligence in obdelave velikih podatkov.

2.2.3 Karakteristike NoSQL podatkovnih baz

NoSQL podatkovne baze so se pojavile kot odgovor na naraščajoče zahteve po fleksibilnosti, skalabilnosti in zmogljivosti, ki jih tradicionalne relacijske (SQL) podatkovne baze pogosto ne morejo zadovoljiti. NoSQL (angl. Not Only SQL) baze podatkov so zasnovane tako, da omogočajo učinkovito obdelavo velikih količin nestrukturiranih ali polstrukturiranih podatkov, ki jih generirajo sodobne aplikacije, kot so spletne platforme, družbena omrežja, analitični sistemi in IoT naprave. V tem podpoglavju bomo podrobno raziskali ključne značilnosti NoSQL podatkovnih baz, vključno z njihovimi vrstami, funkcionalnostmi, prednostmi in slabostmi ter njihovo vlogo v sodobnih informacijskih sistemih.

Arhitektura in osnovne vrste NoSQL podatkovnih baz

NoSQL podatkovne baze se bistveno razlikujejo od tradicionalnih relacijskih podatkovnih baz po svoji arhitekturi in načinu shranjevanja podatkov. Medtem ko SQL baze podatkov temeljijo na tabelarni strukturi in strogi shemi, so NoSQL baze podatkov zasnovane tako, da omogočajo večjo fleksibilnost pri shranjevanju podatkov, saj ne zahtevajo vnaprejšnje definicije sheme. Ta pristop omogoča enostavno prilagajanje in skaliranje glede na potrebe aplikacij, ki pogosto delujejo z dinamičnimi in hitro spreminjajočimi se podatki.

NoSQL podatkovne baze se običajno razvrščajo v štiri glavne kategorije glede na način shranjevanja in organiziranja podatkov:

- **Key-Value baze podatkov:** Key-Value (ključ-vrednost) podatkovne baze so najpreprostejša oblika NoSQL baz, kjer se podatki shranjujejo kot pari ključ-vrednost (Da Silva & Tavares, 2015). Ključ je edinstveni identifikator, medtem ko vrednost predstavlja podatke, ki so lahko poljubnega tipa – od preprostih nizi znakov do kompleksnih binarnih objektov. Ta enostavna

struktura omogoča hitro shranjevanje in pridobivanje podatkov, kar je še posebej koristno v aplikacijah, kjer je ključna hitrost odziva, kot so predpomnilniki, seje uporabnikov ali shranjevanje konfiguracijskih nastavitev. Primeri Key-Value podatkovnih baz vključujejo Redis, Amazon DynamoDB in Riak.

- **Dokumentne baze podatkov:** Dokumentne baze podatkov so zasnovane za shranjevanje podatkov v obliki dokumentov, običajno v formatu JSON, BSON ali XML. Vsak dokument je samostojen zapis, ki lahko vsebuje kompleksne podatkovne strukture, kot so vgrajeni dokumenti ali seznami. Dokumentne baze ponujajo večjo fleksibilnost kot relacijske baze, saj omogočajo shranjevanje različno strukturiranih zapisov v isti zbirki (ekvivalent tabeli v SQL bazi). Ta lastnost je še posebej uporabna v aplikacijah, kjer se podatkovna shema pogosto spreminja, kot so spletne trgovine, kjer različni izdelki zahtevajo različne atribute. Primeri dokumentnih baz vključujejo MongoDB, CouchDB in RavenDB.
- **Column-Family baze podatkov:** Column-Family (stolpčne družine) podatkovne baze temeljijo na shranjevanju podatkov v obliki stolpcev, ki so združeni v družine stolpcev. Ta model je podoben relacijskim bazam, vendar ponuja večjo fleksibilnost pri shranjevanju različnih vrst podatkov v isti zbirki. Stolpčne družine omogočajo shranjevanje velikih količin redkih podatkov (podatkov, kjer večina atributov ni izpolnjena) brez nepotrebnega zapravljanja prostora. Column-Family baze podatkov so pogosto uporabljene v analitičnih aplikacijah in aplikacijah za obdelavo velikih podatkov. Primeri vključujejo Apache Cassandra in HBase.
- **Grafovne baze podatkov:** Grafovne baze podatkov so zasnovane za shranjevanje in poizvedovanje po podatkih, ki so medsebojno povezani v obliki grafov. V grafovnih bazah so podatki organizirani kot vozlišča (angl. entities) in povezave (angl. relationships), kar omogoča učinkovito modeliranje in obdelavo kompleksnih mrež podatkov, kot so družbena omrežja, priporočilni sistemi ali omrežja cest. Grafovne baze podatkov omogočajo hitro iskanje in analizo povezav med podatki, kar je ključna prednost pri aplikacijah, kjer so odnosi med podatki bolj pomembni kot sami atributi. Primeri grafovnih baz vključujejo Neo4j, ArangoDB in Amazon Neptune.

Funkcionalnosti NoSQL podatkovnih baz

NoSQL podatkovne baze ponujajo številne funkcionalnosti, ki so prilagojene potrebam sodobnih aplikacij, zlasti tistih, ki zahtevajo visoko zmogljivost, prilagodljivost in skalabilnost.

Medtem ko se funkcionalnosti lahko razlikujejo med različnimi vrstami NoSQL baz, obstajajo nekatere skupne značilnosti, ki so značilne za večino NoSQL sistemov:

- **Prilagodljivost sheme:** Ena izmed ključnih funkcionalnosti NoSQL baz je prilagodljivost sheme, kar pomeni, da podatki ne potrebujejo vnaprej določene strukture. To omogoča aplikacijam, da shranjujejo različne tipe podatkov v isti zbirki, brez potrebe po spreminjanju sheme baze, ko se zahteve po shranjevanju podatkov spreminjajo. Ta lastnost je še posebej koristna v dinamičnih okoljih, kjer se struktura podatkov lahko pogosto spreminja.
- **Horizontalna skalabilnost:** NoSQL baze podatkov so zasnovane za enostavno razširitev na več strežnikov, kar omogoča obdelavo zelo velikih količin podatkov brez izgube zmogljivosti. Ta lastnost, znana kot horizontalna skalabilnost, omogoča NoSQL bazam, da podpirajo aplikacije z milijoni uporabnikov in petabajti podatkov. Horizontalna skalabilnost se doseže s porazdelitvijo podatkov med več vozlišč, kar omogoča obdelavo več poizvedb hkrati.
- **Visoka zmogljivost:** NoSQL baze podatkov so zasnovane za visoko zmogljivost, kar pomeni, da lahko obdelujejo veliko število poizvedb in operacij na sekundo, tudi pri zelo velikih količinah podatkov. To se doseže z uporabo lahkih podatkovnih modelov in optimiziranih algoritmov za shranjevanje in pridobivanje podatkov. Visoka zmogljivost je ključna v aplikacijah, kjer je odzivni čas kritičen, kot so spletne platforme, igralni sistemi in realnočasovne analitične rešitve.
- **Distribuirana obdelava podatkov:** Številne NoSQL baze podatkov podpirajo distribuirano obdelavo podatkov, kar omogoča, da se obdelava poizvedb in operacij razdeli med več vozlišč. To povečuje zmogljivost in odpornost sistema, saj omogoča, da so podatki vedno dostopni, tudi v primeru izpada posameznega vozlišča. Distribuirani sistemi zagotavljajo visoko razpoložljivost in zmogljivost, kar je ključno za aplikacije, ki zahtevajo neprekinjeno delovanje.
- **CAP teorema:** NoSQL baze podatkov pogosto sledijo CAP teoremi, ki pravi, da podatkovni sistem ne more hkrati zagotoviti doslednosti (angl. Consistency), razpoložljivosti (angl. Availability) in odpornosti na particioniranje (angl. Partition Tolerance). Namesto tega morajo razvijalci izbrati dva izmed treh dejavnikov glede na potrebe aplikacije. Na primer, v sistemih, kjer je ključna razpoložljivost, lahko razvijalci sprejmejo nekaj kompromisov glede doslednosti, da zagotovijo, da so podatki vedno dostopni.

Prednosti NoSQL podatkovnih baz

NoSQL podatkovne baze prinašajo številne prednosti, ki jih tradicionalne relacijske baze podatkov ne morejo ponuditi. Te prednosti so še posebej izrazite v sodobnih aplikacijah, ki morajo obdelovati velike količine nestrukturiranih podatkov ali hitro prilagajati svojo strukturo glede na potrebe.

- **Prilagodljivost:** NoSQL baze podatkov ponujajo izjemno prilagodljivost pri shranjevanju in upravljanju podatkov. Ker ni potrebe po vnaprejšnji definiciji sheme, lahko razvijalci hitro prilagodijo podatkovne strukture glede na spremembe v aplikaciji ali podatkih. To omogoča hitrejše razvijanje in prilagajanje aplikacij, kar je ključnega pomena v dinamičnih okoljih, kot so zagonska podjetja ali spletne storitve.
- **Skalabilnost:** Horizontalna skalabilnost, ki jo ponujajo NoSQL baze, omogoča enostavno razširitev zmogljivosti sistema z dodajanjem novih strežnikov. To je še posebej pomembno za aplikacije, ki morajo obdelovati zelo velike količine podatkov ali upravljati s številnimi uporabniki. Skalabilnost NoSQL baz omogoča podjetjem, da rastejo brez skrbi, da bi obseg podatkov postal prevelik za obstoječo infrastrukturo.
- **Hitrost:** NoSQL baze podatkov so zasnovane za hitrost. Lahki podatkovni modeli in optimizirane strukture za iskanje omogočajo hitro shranjevanje in pridobivanje podatkov. To je še posebej pomembno v aplikacijah, kjer so nizki časi odziva ključni za uporabniško izkušnjo, kot so spletne trgovine, igre ali aplikacije za sporočanje.
- **Prilagoditev velikim podatkom:** NoSQL baze so idealne za aplikacije, ki delajo z velikimi podatki (angl. Big Data). Ker te baze podatkov lahko obvladajo nestrukturirane in polstrukturirane podatke, so primerne za aplikacije, ki zbirajo podatke iz številnih virov, kot so senzorji IoT, družbena omrežja ali spletni dnevniki. To omogoča organizacijam, da zbirajo in analizirajo podatke v realnem času ter tako pridobijo dragocene vpoglede za poslovne odločitve.

Slabosti NoSQL podatkovnih baz

Kljub številnim prednostim imajo NoSQL podatkovne baze tudi nekatere slabosti, ki jih je treba upoštevati pri izbiri prave podatkovne tehnologije za specifične aplikacije.

- **Pomanjkanje standardizacije:** Medtem ko je SQL standardiziran jezik, ki ga podpirajo skoraj vse relacijske baze podatkov, NoSQL baze pogosto uporabljajo lastne poizvedovalne jezike in API. To pomeni, da so aplikacije bolj vezane na specifično implementacijo NoSQL baze, kar lahko otežuje migracijo podatkov ali integracijo z drugimi sistemi.
- **Omejena podpora za ACID transakcije:** Večina NoSQL baz podatkov ne podpira popolnih ACID lastnosti, kot jih poznamo iz SQL baz, kar lahko predstavlja težavo v aplikacijah, kjer je potrebna visoka stopnja doslednosti (Carlson, 2013). Namesto tega se pogosto zanašajo na eventualno doslednost, kjer so podatki dosledni šele po določenem času. To lahko predstavlja težavo v aplikacijah, kjer je potrebna visoka stopnja doslednosti, kot so bančni sistemi ali kritične poslovne aplikacije.
- **Kompleksnost:** NoSQL baze so lahko bolj zapletene za upravljanje, še posebej za razvijalce, ki so vajeni dela z relacijskimi bazami. Konfiguracija, optimizacija in zagotavljanje doslednosti podatkov lahko zahtevajo večje tehnično znanje in izkušnje, kar lahko poveča stroške razvoja in vzdrževanja.
- **Omejene možnosti za analitične poizvedbe:** Medtem ko so NoSQL baze zelo učinkovite pri hitrem shranjevanju in pridobivanju podatkov, pogosto ne ponujajo enake stopnje podpore za zapletene analitične poizvedbe, kot jih poznamo iz SQL baz. To lahko omeji njihovo uporabnost v aplikacijah, kjer so potrebne obsežne analize podatkov.

Zaključek

NoSQL podatkovne baze predstavljajo pomemben napredek v upravljanju podatkov, saj ponujajo visoko prilagodljivost, skalabilnost in zmogljivost, ki so potrebne za sodobne aplikacije, ki delajo z velikimi količinami nestrukturiranih podatkov. Njihova arhitektura, funkcionalnosti in prednosti jih delajo še posebej primerno izbiro za aplikacije, kot so spletne platforme, družbena omrežja, IoT in realnočasovna analitika. Kljub temu pa NoSQL baze podatkov prinašajo tudi nekatere izzive, kot so pomanjkanje standardizacije, omejena podpora za ACID transakcije in kompleksnost upravljanja. Razumevanje teh značilnosti je ključno za pravilno izbiro podatkovne tehnologije glede na specifične potrebe aplikacij in okolja, v katerem delujejo.

2.2.4 Primeri uporabe SQL in NoSQL baz

SQL in NoSQL podatkovne baze predstavljajo dve različni paradigmi za shranjevanje in upravljanje podatkov, vsaka s svojimi prednostmi in slabostmi. Katera tehnologija je primernejša, je odvisno od specifičnih zahtev aplikacije, obsega podatkov, potrebe po doslednosti, zmogljivosti in skalabilnosti. V tem podpoglavju bomo raziskali različne primere uporabe SQL in NoSQL baz, s poudarkom na tem, kako vsaka izmed teh tehnologij izpolnjuje specifične zahteve v različnih industrijah in aplikacijah.

SQL baze podatkov: primeri uporabe

SQL podatkovne baze so zaradi svoje zanesljivosti, doslednosti in podpore za kompleksne poizvedbe postale temelj številnih tradicionalnih in sodobnih informacijskih sistemov. Nekateri najbolj pogosti primeri uporabe SQL baz vključujejo:

- **Finančni sistemi:** Finančne institucije, kot so banke, borze in zavarovalnice, se močno zanašajo na SQL baze podatkov zaradi njihove zmožnosti zagotavljanja visoke stopnje doslednosti, integritete in varnosti podatkov. Transakcijski sistemi v bankah, kot so sistemi za obdelavo plačil, upravljanje računov in kreditna tveganja, zahtevajo strogo spoštovanje ACID lastnosti, ki jih SQL baze podatkov zagotavljajo. Primer je uporaba Microsoft SQL Server v bančništvu, kjer mora sistem zagotoviti, da so vse transakcije izvedene pravilno in brez napak, tudi v primeru izpadov ali napak v sistemu.
- **Sistemi za upravljanje odnosov s strankami (CRM):** CRM sistemi, kot so Salesforce, Microsoft Dynamics in Oracle CRM, uporabljajo SQL baze podatkov za shranjevanje in upravljanje velike količine podatkov o strankah. Ti podatki vključujejo informacije o stikih, zgodovini komunikacije, prodajnih priložnostih, pogodbah in drugih interakcijah s strankami. Zaradi narave teh podatkov je ključnega pomena, da so informacije natančne, konsistentne in zlahka dostopne za analizo. SQL baze podatkov omogočajo izvedbo zapletenih poizvedb za segmentacijo strank, analizo prodajnih trendov in pripravo poročil, kar pomaga podjetjem izboljšati svoje odnose s strankami in optimizirati poslovne procese.
- **Sistemi ERP (sistemi za upravljanje virov podjetja):** Podjetja, ki uporabljajo sisteme ERP, kot so SAP, Oracle ERP in Microsoft Dynamics NAV, se zanašajo na SQL baze podatkov za shranjevanje podatkov o različnih poslovnih funkcijah, vključno s financami, zalogami, proizvodnjo, nabavo in človeškimi viri. SQL baze podatkov omogočajo

centralizirano upravljanje teh podatkov, kar olajša usklajevanje različnih poslovnih procesov in zagotavlja, da so vse informacije konsistentne in ažurne. Na primer, podatki o zalogah, ki se uporabljajo v proizvodnji, morajo biti točni, da se preprečijo zamude pri proizvodnji ali prekomerne zaloge, kar lahko negativno vpliva na poslovanje podjetja.

- **Spletne trgovine:** Spletne trgovine, kot so Amazon, eBay in manjše platforme, uporabljajo SQL baze podatkov za shranjevanje podatkov o izdelkih, strankah, naročilih in plačilih. SQL baze podatkov omogočajo natančno sledenje zalogam, obdelavo naročil in upravljanje podatkov o strankah, kar je ključnega pomena za uspešno delovanje spletne trgovine. Poleg tega omogočajo izvedbo zapletenih poizvedb, ki jih uporabljajo sistemi za priporočanje izdelkov na podlagi preteklih nakupov in preferenc strank. To omogoča spletni trgovini, da izboljša uporabniško izkušnjo in poveča prodajo.
- **Zdravstveni informacijski sistemi:** Zdravstveni informacijski sistemi, kot so sistemi za elektronske zdravstvene zapise (EHR) in sistemi za upravljanje bolnišnic, uporabljajo SQL baze podatkov za shranjevanje pacientovih zdravstvenih podatkov, diagnoz, zdravil, laboratorijskih rezultatov in drugih informacij. Ker so ti podatki občutljivi in življenjsko pomembni, je ključnega pomena, da so pravilno shranjeni, zaščiteni in vedno dostopni. SQL baze podatkov zagotavljajo visoko raven varnosti in integritete podatkov, kar je bistveno za zagotavljanje kakovostne zdravstvene oskrbe in skladnosti z zakonskimi predpisi.

NoSQL baze podatkov: primeri uporabe

NoSQL baze podatkov so postale priljubljene v aplikacijah, kjer so podatki nestrukturirani, kjer je potrebna visoka stopnja skalabilnosti ali kjer relacijski model ni primeren za specifične zahteve aplikacije. Nekateri najbolj pogosti primeri uporabe NoSQL baz vključujejo:

- **Družbena omrežja:** Družbena omrežja, kot so Facebook, Twitter in LinkedIn, uporabljajo NoSQL baze podatkov za shranjevanje in obdelavo velike količine nestrukturiranih podatkov, vključno s statusi, fotografijami, videoposnetki, komentarji in povezavami med uporabniki. Ker se podatkovna struktura na teh platformah lahko hitro spreminja in ker je potrebna visoka stopnja prilagodljivosti in zmogljivosti, NoSQL baze, kot so Cassandra, HBase in MongoDB, omogočajo enostavno skaliranje ter hitro shranjevanje in pridobivanje podatkov. Na primer, Facebook uporablja Cassandra za upravljanje velike količine podatkov, povezanih z uporabniškimi objavami in interakcijami, kar omogoča hitro iskanje in prikaz informacij na platformi.

- **Spletne platforme in vsebinske storitve:** Spletne platforme, kot so Netflix, YouTube in Spotify, uporabljajo NoSQL baze podatkov za shranjevanje podatkov o uporabnikih, priporočilih, predvajanjih in vsebinah. NoSQL baze podatkov, kot je Cassandra, omogočajo hitro iskanje in priporočanje vsebin na podlagi uporabniških preferenc in preteklih dejanj. Poleg tega NoSQL baze zagotavljajo, da so te platforme sposobne obvladati velike količine prometa in uporabnikov, zlasti v času visoke obremenitve, ko na tisoče ali milijone uporabnikov hkrati dostopa do vsebin. Na primer, Netflix uporablja NoSQL baze za shranjevanje podatkov o gledanju vsebin, kar omogoča personalizacijo priporočil in zagotavlja gladko izkušnjo pretakanja videoposnetkov.
- **IoT (internet stvari):** IoT aplikacije, kot so pametna mesta, pametni domovi in industrijski nadzorni sistemi, uporabljajo NoSQL baze podatkov za shranjevanje in obdelavo velikih količin podatkov, ki jih zbirajo senzorji in naprave. Ti podatki so pogosto nestrukturirani in se lahko hitro spreminjajo, kar zahteva visoko prilagodljivost in zmogljivost NoSQL baz. Na primer, pametni domači sistemi, ki vključujejo naprave, kot so termostati, pametne žarnice in varnostne kamere, uporabljajo NoSQL baze, kot je MongoDB, za shranjevanje podatkov o statusu naprav, zgodovini dejavnosti in uporabniških preferencah. To omogoča uporabnikom, da v realnem času spremljajo in upravljajo svoje naprave ter optimizirajo delovanje sistema.
- **Veliki podatki in analitika:** NoSQL baze podatkov so prav tako ključne za aplikacije, ki se ukvarjajo z velikimi podatki (angl. Big Data) in analitiko. Podjetja, kot so Google, Amazon in LinkedIn, uporabljajo NoSQL baze podatkov za zbiranje, shranjevanje in analizo velikih količin podatkov iz različnih virov, vključno z dnevniki, transakcijami, interakcijami strank in drugimi dogodki. Ti podatki so pogosto nestrukturirani ali polstrukturirani in zahtevajo hitro obdelavo in analizo, kar omogoča pridobivanje vpogledov v realnem času. Na primer, Google uporablja Bigtable, ki je NoSQL baza podatkov, za shranjevanje in obdelavo velikih količin podatkov, povezanih z iskalnimi poizvedbami, oglaševalskimi kampanjami in analitiko spletnih mest.
- **Spletne igre:** Spletne igre, kot so MMORPG (angl. Massively Multiplayer Online Role-Playing Games), uporabljajo NoSQL baze podatkov za upravljanje velikih količin podatkov, povezanih z igralci, njihovimi dosežki, virtualnimi predmeti in interakcijami z drugimi igralci. NoSQL baze podatkov, kot je Redis, omogočajo hitro shranjevanje in pridobivanje podatkov v realnem času, kar je ključnega pomena za zagotavljanje tekoče igralne izkušnje.

Poleg tega NoSQL baze omogočajo enostavno skaliranje, kar je pomembno, ko število igralcev hitro narašča ali ko so na voljo novi vsebinski paketi in nadgradnje. Na primer, igre, kot je League of Legends, uporabljajo NoSQL baze za shranjevanje podatkov o napredovanju igralcev in za sinhronizacijo teh podatkov med različnimi strežniki.

Kombinirana uporaba SQL in NoSQL baz (poliglotska obstojnost)

V praksi se pogosto zgodi, da nobena izmed baz podatkov ne more popolnoma izpolniti vseh zahtev kompleksne aplikacije. Zato se v sodobnih informacijskih sistemih pogosto uporablja kombinacija SQL in NoSQL baz podatkov, kar imenujemo poliglotska obstojnost. V tem pristopu se različne baze podatkov uporabljajo za različne dele aplikacije glede na njihove specifične zahteve in prednosti.

- **E-trgovina:** Velike e-trgovinske platforme pogosto uporabljajo SQL baze podatkov za shranjevanje podatkov o naročilih, strankah in plačilih, kjer so ključni dejavniki doslednost, integriteta in varnost podatkov. Hkrati pa uporabljajo NoSQL baze podatkov za shranjevanje nestrukturiranih podatkov, kot so uporabniške ocene, zgodovina iskanj in priporočila izdelkov. Ta kombinacija omogoča platformi, da izkoristi prednosti obeh tehnologij – zanesljivost SQL baz za ključne poslovne podatke in fleksibilnost NoSQL baz za personalizacijo in skalabilnost.
- **Digitalni marketing:** Podjetja, ki se ukvarjajo z digitalnim marketingom, pogosto uporabljajo SQL baze podatkov za shranjevanje podatkov o kampanjah, strankah in transakcijah, hkrati pa uporabljajo NoSQL baze podatkov za shranjevanje podatkov o vedenju uporabnikov, interakcijah na spletnih mestih in analitiki družbenih omrežij. Na primer, SQL baze se uporabljajo za ustvarjanje in upravljanje kampanj, medtem ko NoSQL baze omogočajo analizo uporabniških interakcij v realnem času, kar omogoča prilagajanje kampanj na podlagi trenutnih podatkov.
- **IoT in pametna mesta:** V pametnih mestih se kombinacija SQL in NoSQL baz uporablja za upravljanje različnih vrst podatkov, zbranih s senzorji in napravami IoT. SQL baze podatkov se uporabljajo za shranjevanje dolgoročnih podatkov o infrastrukturi in upravljanje projektov, medtem ko NoSQL baze podatkov obdelujejo in shranjujejo realnočasovne podatke, kot so prometni podatki, okoljski senzorji in podatki o energiji. Ta kombinacija omogoča mestom, da učinkovito upravljajo svoje vire in zagotavljajo storitve v realnem času.

Zaključek

SQL in NoSQL baze podatkov ponujajo različne prednosti, ki so ključne za specifične primere uporabe. SQL baze podatkov izstopajo pri zagotavljanju doslednosti, integritete in zmogljivosti za zapletene poizvedbe v aplikacijah, kjer so podatki strogo strukturirani. Po drugi strani pa NoSQL baze podatkov omogočajo visoko prilagodljivost, skalabilnost in zmogljivost pri obdelavi nestrukturiranih ali polstrukturiranih podatkov, kar jih naredi nepogrešljive v sodobnih aplikacijah, kot so družbena omrežja, IoT in Big Data analitika. Kombinirana uporaba obeh vrst baz podatkov omogoča organizacijam, da izkoristijo prednosti obeh pristopov in tako zagotovijo optimalno delovanje svojih informacijskih sistemov.

2.3 Primerjalna analiza PostgreSQL in Redisa

2.3.1 Uvod v PostgreSQL

PostgreSQL je ena izmed najstarejših in najbolj razširjenih odprtokodnih relacijskih baz podatkov (RDBMS), ki slovi po svoji zanesljivosti, razširljivosti in bogati funkcionalnosti. PostgreSQL, ki se pogosto imenuje preprosto »Postgres«, se je razvil iz projekta POSTGRES, ki ga je v osemdesetih letih začel Michael Stonebraker na Univerzi v Kaliforniji, Berkeley. Od takrat se je PostgreSQL razvil v robusten in zmogljiv sistem za upravljanje podatkovnih baz, ki je primeren za širok spekter aplikacij, od majhnih podjetij do velikih korporacijskih sistemov.

Zgodovinski razvoj PostgreSQL

PostgreSQL ima dolgo zgodovino, ki sega v leto 1986, ko se je začel projekt POSTGRES (Post Ingres). Namen projekta je bil odpraviti omejitve, ki so jih imele takratne relacijske podatkovne baze, zlasti glede obvladovanja kompleksnih podatkovnih tipov in razširljivosti. Projekt POSTGRES je uvedel številne inovacije, kot so podpora za dedovanje tabel, sistem za obdelavo transakcij z zamenljivimi pravilniki in izboljšano ravnanje z objektno usmerjenimi podatkovnimi tipi.

Leta 1995 je POSTGRES postal PostgreSQL, ko je bila dodana podpora za SQL kot privzeti poizvedovalni jezik. Ta prehod je označil začetek nove ere za PostgreSQL, saj je kombiniral

inovativne funkcije POSTGRES z močjo in priljubljenostjo jezika SQL. Od takrat se je PostgreSQL nenehno razvijal, pri čemer so bili uvedeni številni pomembni dodatki, kot so podpora za tuje ključe, triggere, pogledi in shranjene procedure. Sčasoma je PostgreSQL postal eden najmočnejših sistemov za upravljanje relacijskih baz podatkov, ki je povsem primerljiv s komercialnimi rešitvami, kot so Oracle, Microsoft SQL Server in IBM Db2.

Arhitektura PostgreSQL

PostgreSQL ima modularno in razširljivo arhitekturo, ki temelji na procesnem modelu, kar omogoča učinkovito upravljanje z viri in zanesljivo obdelavo podatkov. Arhitektura PostgreSQL sestoji iz več ključnih komponent, ki skupaj omogočajo delovanje celotnega sistema.

- **Postmaster in procesi:** Postmaster je glavni proces v PostgreSQL, ki nadzira delovanje vseh ostalih procesov v sistemu. Ko se uporabnik poveže z bazo podatkov, Postmaster ustvari nov proces (imenovan backend proces), ki obdeluje poizvedbe in druge zahteve tega uporabnika. Ta procesna arhitektura zagotavlja visoko stopnjo izolacije med uporabniki, kar povečuje stabilnost in varnost sistema.
- **Delitev dela med procesi:** PostgreSQL uporablja procesno arhitekturo, kjer vsak uporabnik dobi svoj lasten proces, ki obdeluje njegove poizvedbe. Ta pristop omogoča učinkovito izkoriščanje večprocesorskih sistemov in zmanjšuje možnosti za konflikte med uporabniki. Vsak proces uporablja svoj lastni del pomnilnika in ima dostop do skupnega skladišča podatkov, kar omogoča hitro obdelavo transakcij.
- **Sistem za transakcije in zaklepanje:** PostgreSQL podpira popolne ACID lastnosti (angl. Atomicity, Consistency, Isolation, Durability), kar zagotavlja zanesljivo izvajanje transakcij. Transakcijski sistem uporablja mehanizem večverzijske hrambe (angl. Multi-Version Concurrency Control – MVCC), ki omogoča več uporabnikom hkratno izvajanje poizvedb brez potrebe po zaklepanju tabel, s čimer se zmanjšajo zastoji in izboljša zmogljivost sistema.
- **Razširljivost:** Ena izmed najbolj izstopajočih značilnosti PostgreSQL je njegova razširljivost. Uporabniki in razvijalci lahko zlahka dodajajo nove tipe podatkov, operaterje, indekse, funkcije, jezike za shranjene procedure in celo celotne metode za dostop do

podatkov. Ta modularna zasnova omogoča, da se PostgreSQL prilagodi različnim potrebam in se uporablja v zelo specifičnih aplikacijah, kjer so potrebne posebne funkcionalnosti.

Podprte funkcionalnosti in lastnosti

PostgreSQL se ponaša z bogatim naborom funkcionalnosti, ki omogočajo razvijalcem in administratorjem učinkovito delo s podatki in prilagoditev sistema specifičnim potrebam aplikacij. Nekatere ključne funkcionalnosti PostgreSQL vključujejo:

- **Podpora za kompleksne tipe podatkov:** PostgreSQL podpira širok nabor podatkovnih tipov, vključno s primitivnimi tipi (npr. INTEGER, VARCHAR, BOOLEAN), naprednimi tipi (npr. JSON, XML, hstore) in uporabniško definiranimi tipi podatkov. To omogoča shranjevanje in obdelavo kompleksnih podatkov, kot so strukture, geografski podatki (PostGIS) in druge specializirane aplikacije.
- **Polnitev in iskanje v polnih besedilih:** PostgreSQL vključuje močne funkcionalnosti za iskanje po polnih besedilih, kar omogoča hitro in učinkovito iskanje po velikih besedilnih podatkovnih zbirkah. Funkcija tsvector in operaterji za iskanje po besedilu omogočajo napredno indeksiranje in poizvedovanje, kar je še posebej uporabno v aplikacijah, kot so spletni iskalniki in informacijski sistemi.
- **Napredne možnosti indeksiranja:** PostgreSQL podpira več vrst indeksov, vključno z B-tree, Hash, GiST (angl. Generalized Search Tree), GIN (angl. Generalized Inverted Index) in BRIN (angl. Block Range INdex). Te različne vrste indeksov omogočajo optimizacijo iskanja in poizvedovanja glede na specifične potrebe aplikacije, kar izboljša zmogljivost in učinkovitost baze podatkov.
- **Replikacija in visoka razpoložljivost:** PostgreSQL ponuja več možnosti za replikacijo podatkov, vključno z asinhrono in sinhrono replikacijo, kar zagotavlja visoko razpoložljivost sistema in varnost podatkov. Replikacija omogoča ustvarjanje kopij baze podatkov na oddaljenih strežnikih, kar poveča odpornost na napake in omogoča skaliranje z razdelitvijo obremenitve med več strežnikov.
- **Podpora za particioniranje tabel:** Particioniranje tabel omogoča razdelitev velikih tabel na manjše, bolj obvladljive dele (particije), kar lahko izboljša zmogljivost poizvedb in olajša upravljanje velikih podatkovnih zbirk. PostgreSQL omogoča več načinov particioniranja,

vključno z obsegovnim in seznamnim particioniranjem, kar administratorjem omogoča prilagajanje particioniranja glede na potrebe aplikacije.

- **Podpora za shranjene procedure in trigerje:** PostgreSQL podpira shranjene procedure in trigerje, ki omogočajo izvajanje kompleksne poslovne logike znotraj baze podatkov. Shranjene procedure so napisane v več različnih programskih jezikih, vključno s PL/pgSQL, PL/Python, PL/Perl in drugimi. Trigerji omogočajo avtomatsko izvajanje določenih operacij, kot so preverjanja integritete ali posodobitve drugih tabel, ob določenih dogodkih v podatkovni bazi.

Uporaba PostgreSQL v praksi

PostgreSQL je široko uporabljen v različnih industrijah in aplikacijah zaradi svoje zanesljivosti, zmogljivosti in prilagodljivosti. Nekateri pogosti primeri uporabe vključujejo:

- **Poslovne aplikacije:** PostgreSQL se pogosto uporablja v poslovnih aplikacijah, kot so sistemi ERP, sistemi CRM in aplikacije za upravljanje dobavne verige, kjer so pomembne zanesljivost, doslednost in zmogljivost baze podatkov.
- **Spletne aplikacije:** Mnoge spletne aplikacije uporabljajo PostgreSQL zaradi njegove sposobnosti obvladovanja velikega števila uporabnikov in zahtevnih poizvedb. Priljubljene platforme, kot so Reddit, Instagram in Disqus, uporabljajo PostgreSQL za shranjevanje in obdelavo podatkov, ki so povezani z uporabniškimi interakcijami, vsebinami in drugimi podatki.
- **Geografski informacijski sistemi (GIS):** PostgreSQL je v kombinaciji s podaljškom PostGIS pogosto uporabljena rešitev za shranjevanje in obdelavo geolokacijskih podatkov. Aplikacije GIS uporabljajo PostgreSQL za obdelavo podatkov o zemljevidih, koordinatah, infrastrukturah ter drugih geografskih informacijah, kar omogoča kompleksne prostorske poizvedbe in analize.
- **Znanstvene in raziskovalne aplikacije:** Znanstveni in raziskovalni projekti pogosto uporabljajo PostgreSQL za shranjevanje in obdelavo velikih količin podatkov. Njegova podpora za napredne tipe podatkov, razširljivost in zanesljivost omogočajo učinkovito upravljanje podatkov, ki so ključni za raziskave na področjih, kot so genomika, astronomija in okoljske znanosti.

Sklep

PostgreSQL je izjemno zmogljiv in prilagodljiv sistem za upravljanje relacijskih baz podatkov, ki je postal izbira številnih organizacij po svetu. Njegova bogata funkcionalnost, podpora za napredne tipe podatkov, razširljivost in zanesljivost ga postavljajo med vodilne odprtokodne rešitve na področju baz podatkov. Razumevanje osnovne arhitekture in ključnih lastnosti PostgreSQL je ključnega pomena za izkoriščanje njegovih prednosti v sodobnih aplikacijah in informacijskih sistemih, kar omogoča zanesljivo in učinkovito upravljanje podatkov v različnih industrijah.

2.3.2 Uvod v Redis

Redis (angl. Remote Dictionary Server) je odprtokodni sistem za shranjevanje podatkov v pomnilniku, znan po svoji hitrosti, enostavnosti in zmogljivosti. Redis je bil zasnovan kot podatkovna baza ključ-vrednost, kjer so podatki shranjeni kot pari ključ-vrednost, vendar sčasoma se je razvil v napreden sistem, ki podpira številne strukture podatkov, kot so seznami, množice in hashtable. Zaradi svojih zmogljivosti in enostavnosti uporabe se Redis pogosto uporablja kot predpomnilnik, posrednik sporočil in celo kot primarna podatkovna baza za določene aplikacije, ki zahtevajo visoko hitrost in nizko zakasnitev.

Zgodovinski razvoj Redisa

Redis je leta 2009 ustvaril Salvatore Sanfilippo, italijanski programer, ki je želel rešiti specifičen problem pri aplikaciji, ki jo je razvijal. Sanfilippo je potreboval sistem, ki bi omogočal hitro dostopanje do podatkov in obdelavo velikih količin informacij v realnem času. Na podlagi tega je zasnoval Redis, ki je bil sprva mišljen kot predpomnilnik podatkov, a se je hitro razvil v večnamenski sistem za shranjevanje podatkov v pomnilniku.

Od svojega nastanka je Redis doživel številne izboljšave in razširitve. Postopoma so bile dodane nove strukture podatkov, kot so nizi, seznami, množice in urejene množice, kar je omogočilo uporabo Redisa v vedno bolj kompleksnih scenarijih. Poleg tega so bile uvedene funkcionalnosti, kot so trajnost (persistence), replikacija, skriptiranje s pomočjo jezika Lua in podpora za porazdeljene operacije (angl. Cluster mode), kar je Redisu omogočilo, da se je razvil v robusten sistem za upravljanje podatkov.

Danes je Redis ena izmed najbolj priljubljenih baz podatkov za shranjevanje podatkov v pomnilniku, ki jo uporabljajo podjetja po vsem svetu, vključno z velikani, kot so Twitter, GitHub, Stack Overflow in Instagram. Njegova hitrost, prilagodljivost in enostavnost uporabe so ključni razlogi za njegovo široko uporabo.

Arhitektura Redisa

Redis ima enostavno, a zelo učinkovito arhitekturo, ki temelji na shranjevanju podatkov v pomnilniku, kar omogoča izjemno hitrost dostopa in obdelave podatkov. Ključni elementi arhitekture Redisa vključujejo naslednje komponente:

- **Enotna instanca in proces:** Redis je zasnovan kot enojni proces, ki deluje na enem CPU-ju in uporablja enotno nit (angl. single-threaded) za obdelavo vseh operacij. Ta enostavna zasnova omogoča, da Redis doseže visoke hitrosti in nizke zakasnitve, saj se izogne zapletenostim, povezanih s sinhronizacijo več niti ali procesov.
- **Shranjevanje podatkov v pomnilniku:** Vsi podatki v Redisu so shranjeni v pomnilniku, kar omogoča zelo hitro branje in pisanje podatkov. Ta arhitektura je ključna za zmogljivost Redisa, saj odpravi zakasnitve, ki bi jih povzročil dostop do diska. Redis uporablja različne metode za optimizacijo uporabe pomnilnika, kot so shranjevanje podatkov v kompaktnem formatu in izogibanje nepotrebnim kopiranjem podatkov.
- **Podpora za trajnost:** Čeprav je Redis primarno zasnovan za shranjevanje podatkov v pomnilniku, podpira tudi trajnost podatkov (persistence), kar omogoča shranjevanje podatkov na disk za zaščito pred izgubo podatkov v primeru izpada sistema. Redis omogoča dve glavni metodi za dosego trajnosti: shranjevanje podatkovnih posnetkov (angl. snapshotting) in zapisovanje operacij v dnevnik (angl. Append-Only File – AOF). Te funkcionalnosti omogočajo, da se Redis uporablja tudi v scenarijih, kjer je trajnost podatkov kritična.
- **Replikacija in visoka razpoložljivost:** Redis podpira asinhrono replikacijo, kjer lahko en strežnik (angl. master) replicira svoje podatke na enega ali več podrejenih strežnikov (angl. slaves). Ta funkcionalnost omogoča, da Redis doseže visoko razpoložljivost in odpornost na napake, saj lahko v primeru izpada glavnega strežnika podrejeni strežniki takoj prevzamejo naloge glavnega strežnika. Redis prav tako podpira avtomatsko preklapljanje na nov glavni strežnik v primeru izpada, kar povečuje zanesljivost sistema.

- **Porazdeljene strukture podatkov:** Redis omogoča uporabo porazdeljenih struktur podatkov s pomočjo načina Redis Cluster. Redis Cluster razdeli podatke med več strežnikov (angl. shards), kar omogoča vodoravno skaliranje in povečuje zmogljivost sistema. Redis Cluster omogoča tudi avtomatsko preklapljanje v primeru izpada vozlišča, kar zagotavlja, da so podatki vedno dostopni.

Podprte funkcionalnosti in lastnosti

Redis se ponaša z bogatim naborom funkcionalnosti, ki omogočajo, da se uporablja v širokem spektru aplikacij. Nekatere ključne funkcionalnosti Redisa vključujejo:

- Podpora za različne strukture podatkov: Redis podpira več različnih struktur podatkov, ki omogočajo fleksibilno shranjevanje in obdelavo podatkov. Med podprte strukture podatkov spadajo:
  - **Nizi (angl. Strings):** Osnovna enota shranjevanja v Redisu je niz, ki lahko vsebuje katero koli binarno vrednost, kot so besedila, števila ali binarni podatki.
  - **Seznami (angl. Lists):** Seznami so urejene zbirke nizov, ki omogočajo vstavljanje in odstranjevanje elementov na začetku ali koncu seznama. Uporabljajo se za vrste opravil (angl. queues), shranjevanje dnevniških datotek (angl. logs) in druge aplikacije.
  - **Množice (angl. Sets):** Množice so neurejene zbirke edinstvenih nizov, ki omogočajo hitro preverjanje članstva, združevanje, presek in razliko med množicami.
  - **Urejene množice (angl. Sorted Sets):** Urejene množice so podobne množicam, vendar vsak element vsebuje tudi vrednost točke (angl. score), na podlagi katere so elementi urejeni. Uporabljajo se za shranjevanje rezultatov, razvrstitev in podobne naloge.
  - **Hash tabele (angl. Hashes):** Hash tabele omogočajo shranjevanje parov polje-vrednost znotraj enega ključa, kar omogoča organizacijo povezanih podatkov, kot so atributi uporabnika ali konfiguracijske nastavitve.
  - **Bitmap in HyperLogLog**: Redis podpira tudi specializirane strukture podatkov, kot so bitmap za obdelavo bitnih operacij in HyperLogLog za ocenjevanje kardinalnosti množic.
  - **Transakcije:** Redis omogoča izvajanje transakcij s pomočjo ukazov MULTI, EXEC, DISCARD in WATCH. Transakcije v Redisu omogočajo izvajanje več operacij kot

atomarno enoto, kar pomeni, da se vse operacije izvedejo ali nobena. To je pomembno za ohranjanje doslednosti podatkov v scenarijih, kjer je treba izvesti več povezanih operacij.

- **Skriptiranje z Lua:** Redis podpira skriptiranje s pomočjo jezika Lua, kar omogoča izvajanje kompleksnih operacij neposredno v strežniku. Skripte v jeziku Lua lahko združujejo več ukazov v eno transakcijo, kar izboljša zmogljivost in zmanjša omrežni promet.
- **Pub/Sub (angl. Publish/Subscribe):** Redis omogoča implementacijo sistema za posredovanje sporočil z uporabo modela Publish/Subscribe. Ta funkcionalnost omogoča, da se sporočila pošiljajo več prejemnikom (subscriberjem) hkrati, kar je uporabno za aplikacije, kot so klepeti, obvestilni sistemi in sistemi za distribucijo dogodkov.
- **Geolokacijske poizvedbe:** Redis podpira geolokacijske poizvedbe, kar omogoča shranjevanje in obdelavo geolokacijskih podatkov, kot so koordinate krajev. S to funkcionalnostjo je mogoče izvajati poizvedbe, kot so iskanje najbližjih točk ali razdalje med točkami, kar je uporabno v aplikacijah za sledenje, navigacijo in geografsko iskanje.

Uporaba Redisa v praksi

Redis se uporablja v številnih industrijah in aplikacijah, kjer so ključne visoka hitrost, nizka zakasnitev in prilagodljivost. Nekateri pogosti primeri uporabe Redisa vključujejo:

- **Predpomnilniki (angl. Caching):** Eden izmed najbolj priljubljenih primerov uporabe Redisa je kot predpomnilnik za shranjevanje pogosto dostopnih podatkov, kar zmanjšuje obremenitev glavne baze podatkov in izboljša odzivni čas aplikacij. Spletne strani, kot so Twitter, GitHub in Stack Overflow, uporabljajo Redis za predpomnjenje rezultatov poizvedb, sejnih podatkov in drugih informacij, ki jih uporabniki pogosto potrebujejo.
- **Sistemi za sledenje in analitiko:** Redis se pogosto uporablja v sistemih za sledenje in analitiko, kjer je pomembno, da se podatki obdelujejo v realnem času. Na primer, platforme za oglaševanje uporabljajo Redis za sledenje prikazom oglasov, klikom in konverzijam, kar omogoča natančno in hitro analizo učinkovitosti oglaševalskih kampanj.
- **Sistemi za upravljanje seje:** Redis se pogosto uporablja za shranjevanje sejnih podatkov v spletnih aplikacijah, saj omogoča hiter dostop in posodobitev podatkov o uporabnikih, kot so prijavni podatki, košarice in nastavitve. Redisov model ključ-vrednost je idealen za

shranjevanje podatkov o sejah, saj omogoča enostavno iskanje in posodabljanje podatkov z uporabo edinstvenih ID-jev sej.

- **Sistemi za posredovanje sporočil:** Redisov model Pub/Sub omogoča hitro in učinkovito distribucijo sporočil med različnimi komponentami sistema. To se pogosto uporablja v aplikacijah, kot so klepeti v realnem času, obvestilni sistemi in sistemi za distribucijo dogodkov, kjer je pomembno, da se sporočila hitro prenesejo med različnimi odjemalci.
- **Razvrstitev in ocenjevanje:** Urejene množice v Redisu so idealne za implementacijo sistemov za razvrstitev in ocenjevanje, kot so lestvice najboljših igralcev v igrah, seznami priljubljenih vsebin ali rangiranje uporabnikov. Redis omogoča hitro vstavljanje, posodabljanje in iskanje elementov glede na njihove točke, kar zagotavlja nemoteno delovanje aplikacij, ki temeljijo na razvrstitvi podatkov.

Sklep

Redis je izjemno hiter in prilagodljiv sistem za shranjevanje podatkov v pomnilniku, ki se uporablja v številnih kritičnih aplikacijah, kjer so hitrost, zanesljivost in enostavnost uporabe ključnega pomena. Njegova podpora za različne strukture podatkov, možnosti za trajnost, replikacija, porazdeljene operacije in zmogljivosti skriptiranja omogočajo široko uporabo Redisa v različnih industrijah. Razumevanje osnovne arhitekture in funkcionalnosti Redisa je ključno za učinkovito izkoriščanje njegovih zmogljivosti in prilagoditev specifičnim potrebam sodobnih informacijskih sistemov.

2.3.3 Primerjava arhitektur PostgreSQL in Redisa

PostgreSQL in Redis sta dve priljubljeni, a po naravi zelo različni podatkovni bazi, ki služita različnim namenom in se uporabljata v različnih vrstah aplikacij. Medtem ko je PostgreSQL zmogljiv relacijski sistem za upravljanje baz podatkov (RDBMS), znan po svoji doslednosti, kompleksnih funkcionalnostih in podpori za strukturirane podatke, je Redis sistem za shranjevanje podatkov v pomnilniku, optimiziran za hitrost in nizko zakasnitev pri obdelavi podatkov, pogosto uporabljen kot predpomnilnik ali za specifične naloge, kjer je ključna hitrost dostopa. V tem podpoglavju bomo podrobno primerjali arhitekture obeh sistemov, s poudarkom na njihovih ključnih značilnostih, prednostih in slabostih.

Arhitekturni model PostgreSQL

PostgreSQL je zasnovan kot relacijska baza podatkov, ki sledi strogi skladnosti ACID (angl. Atomicity, Consistency, Isolation, Durability), kar zagotavlja zanesljivost in celovitost podatkov. Arhitektura PostgreSQL je kompleksna in robustna, kar omogoča obdelavo zapletenih poizvedb, transakcij in poslovne logike. Glavne značilnosti arhitekture PostgreSQL vključujejo:

- **Večnitenost in večprocesnost:** PostgreSQL deluje na principu večnitenosti in večprocesnosti. Ko se uporabnik poveže z bazo podatkov, PostgreSQL ustvari ločen proces (imenovan »backend process«), ki upravlja z zahtevami tega uporabnika. Ta ločitev omogoča izolacijo transakcij in povečuje stabilnost sistema, saj se morebitne težave v enem procesu ne prenesejo na druge procese.
- **Večverzijska hramba (MVCC):** PostgreSQL uporablja sistem večverzijske hrambe (MVCC), ki omogoča hkratno izvajanje poizvedb in posodobitev brez zaklepanja tabel (Obe & Hsu, 2017). MVCC omogoča, da lahko uporabniki dostopajo do več različic podatkov hkrati, kar izboljša zmogljivost sistema in zmanjšuje konflikte med transakcijami.
- **Strukturiranost in podpora za kompleksne tipe podatkov:** PostgreSQL je zasnovan za delo s strukturiranimi podatki, shranjenimi v tabelah. Podpira širok nabor podatkovnih tipov, vključno s preprostimi tipi (kot sta INTEGER, TEXT), kompleksnimi tipi (kot sta JSON, ARRAY) in uporabniško definiranimi tipi. PostgreSQL omogoča izvajanje zapletenih poizvedb z uporabo jezika SQL, kar je bistveno za aplikacije, ki zahtevajo analizo in manipulacijo velikih količin podatkov.
- **Podpora za shranjene procedure in trigerje:** PostgreSQL omogoča izvajanje kompleksne poslovne logike znotraj baze podatkov s pomočjo shranjenih procedur in trigerjev. To omogoča, da se določeni postopki izvajajo avtomatsko ob določenih dogodkih, kot so vstavljanje, posodabljanje ali brisanje podatkov.
- **Replikacija in razširljivost:** PostgreSQL podpira več načinov replikacije (npr. asinhrona, sinhrona replikacija), kar omogoča ustvarjanje kopij podatkov na več strežnikih za zagotavljanje visoke razpoložljivosti in odpornosti na napake. Poleg tega PostgreSQL omogoča razširitev funkcionalnosti prek dodatkov (angl. extensions), kar omogoča prilagajanje sistema specifičnim potrebam aplikacije.

Arhitekturni model Redisa

Redis je zasnovan kot podatkovna baza ključ-vrednost v pomnilniku, ki je optimizirana za hitrost dostopa in nizko zakasnitev (Carlson, 2013). Arhitektura Redisa je enostavnejša v primerjavi s PostgreSQL, vendar omogoča izjemno hitro obdelavo podatkov, zaradi česar je idealen za aplikacije, ki zahtevajo visoko zmogljivost v realnem času. Ključne značilnosti arhitekture Redisa vključujejo:

- **Enotni proces in enotna nit:** Redis deluje kot enotni proces, ki uporablja enotno nit za obdelavo vseh zahtev. Ta zasnova, čeprav na prvi pogled omejujoča, omogoča Redisu, da doseže zelo visoke hitrosti, saj se izogne zapletenostim, ki so povezane s sinhronizacijo več niti ali procesov. Enotni proces pomeni, da vse operacije potekajo zaporedno, kar zagotavlja atomarnost in doslednost podatkov.
- **Shranjevanje podatkov v pomnilniku:** Vsi podatki v Redisu so shranjeni v pomnilniku, kar omogoča izjemno hitrost pri branju in pisanju podatkov. Ta arhitektura je ključna za zmogljivost Redisa, saj se odpravi potreba po dostopu do diska, kar bi povečalo zakasnitev. Vendar pa to pomeni tudi, da je zmogljivost sistema omejena z velikostjo pomnilnika.
- **Podpora za različne strukture podatkov:** Čeprav je Redis primarno zasnovan kot sistem ključ-vrednost, podpira več različnih struktur podatkov, vključno z nizi, seznami, množicami, urejenimi množicami in hashtable. Ta raznolikost omogoča, da se Redis uporablja za številne različne naloge, kot so predpomnilniki, vrste opravil in sistemi za sledenje.
- **Replikacija in porazdeljenost:** Redis podpira asinhrono replikacijo, ki omogoča, da en strežnik replicira svoje podatke na več podrejenih strežnikov, kar povečuje razpoložljivost sistema in odpornost na napake. Redis prav tako omogoča uporabo Redis Clustera, ki razdeli podatke med več strežnikov in omogoča horizontalno skaliranje. To zagotavlja, da lahko Redis obvladuje zelo velike obremenitve in velike količine podatkov.
- **Trajnost podatkov:** Redis omogoča shranjevanje podatkov na disk prek funkcionalnosti za trajnost (persistence), ki vključuje shranjevanje posnetkov stanja (angl. snapshotting) in zapisovanje operacij v dnevnik (AOF). Čeprav je Redis zasnovan za delovanje v pomnilniku, te funkcionalnosti omogočajo zaščito podatkov v primeru izpada sistema.

Primerjava ključnih arhitekturnih značilnosti

Pri primerjavi arhitektur PostgreSQL in Redis je pomembno razumeti, kako se ti sistemi razlikujejo glede na zasnovo, cilje in namen uporabe.

- **Namen in uporaba:** PostgreSQL je zasnovan kot splošno uporabna relacijska baza podatkov, ki je primerna za širok spekter aplikacij, ki zahtevajo strogo doslednost podatkov, podporo za kompleksne poizvedbe in obsežno shranjevanje strukturiranih podatkov. PostgreSQL je idealen za poslovne aplikacije, analitiko, upravljanje poslovnih procesov in aplikacije, kjer je ključna natančnost podatkov.

  Redis je zasnovan kot sistem za shranjevanje podatkov v pomnilniku, optimiziran za hitrost in nizko zakasnitev. Idealen je za aplikacije, kjer je ključna hitrost dostopa do podatkov, kot so predpomnilniki, sistemi za obdelavo podatkov v realnem času, klepeti in sistemi za sledenje dogodkom. Redis se pogosto uporablja v kombinaciji z drugimi bazami podatkov, kjer služi kot predpomnilnik ali kot hitro dostopna baza za specifične naloge.

- **Podatkovni model in strukture podatkov:** PostgreSQL uporablja relacijski podatkovni model, kjer so podatki organizirani v tabelah, ki so med seboj povezane z relacijami. Ta struktura omogoča shranjevanje in obdelavo kompleksnih podatkovnih tipov in podporo za zapletene poizvedbe z uporabo jezika SQL.

  Redis uporablja podatkovni model ključ-vrednost, kjer so podatki shranjeni kot pari ključ-vrednost. Čeprav Redis podpira več vrst struktur podatkov, je njegova zasnova veliko enostavnejša in bolj omejena v primerjavi z relacijskim modelom PostgreSQL. Vendar pa ta enostavnost omogoča izjemno hitrost pri obdelavi podatkov.

- **Skalabilnost in zmogljivost:** PostgreSQL je zasnovan za obvladovanje velikih količin podatkov in omogoča vertikalno skaliranje (povečanje zmogljivosti enega strežnika) ter horizontalno skaliranje prek replikacije in delitve podatkov. Vendar je zmogljivost PostgreSQL odvisna od zapletenosti poizvedb in strukture podatkov.

  Redis je zasnovan za horizontalno skaliranje prek Redis Clustera, kar omogoča enostavno razširitev zmogljivosti z dodajanjem novih strežnikov. Redisova zmogljivost je optimizirana za hitrost, saj se vsi podatki shranjujejo v pomnilniku, kar omogoča zelo hitro obdelavo podatkov.

- **Transakcije in doslednost:** PostgreSQL sledi strogi skladnosti ACID, kar zagotavlja doslednost podatkov tudi v kompleksnih transakcijah. To pomeni, da so vse operacije v transakciji izvedene bodisi v celoti bodisi sploh ne, kar zagotavlja visoko raven zanesljivosti.

  Redis omogoča izvajanje atomarnih operacij, vendar nima enake ravni podpore za transakcije kot PostgreSQL. Redis podpira osnovne transakcije z uporabo ukazov MULTI in EXEC, vendar je večina operacij izvedenih brez zaklepanja, kar omogoča hitrejše delovanje, a manjšo stopnjo doslednosti v primeru napak.

- **Trajnost podatkov:** PostgreSQL zagotavlja trajnost podatkov z uporabo shranjevanja na disk, kjer so vsi podatki trajno shranjeni v datotečnem sistemu. To zagotavlja, da so podatki vedno na voljo tudi po izpadu sistema ali ponovnem zagonu.

  Redis je primarno zasnovan za shranjevanje podatkov v pomnilniku, kar omogoča izjemno hitrost, vendar to pomeni, da so podatki v primeru izpada sistema lahko izgubljeni. Redis omogoča trajnost prek posnetkov in dnevnikov operacij, vendar ta funkcionalnost ni tako robustna kot pri PostgreSQL.

Uporabniški primeri za PostgreSQL in Redis

PostgreSQL:

- **Poslovne aplikacije:** PostgreSQL je idealen za aplikacije, kjer so ključni podatki strogo strukturirani in kjer je potrebna podpora za kompleksne poizvedbe, transakcije in integriteto podatkov. Primeri vključujejo sisteme ERP, sisteme CRM in finančne aplikacije.
- **Analitične aplikacije:** PostgreSQL je pogosto uporabljen v analitičnih aplikacijah, kjer je treba obdelovati velike količine podatkov z uporabo zapletenih poizvedb in analitičnih funkcij.
- **Upravljanje velikih podatkovnih zbirk:** PostgreSQL se uporablja za shranjevanje velikih količin strukturiranih podatkov, kjer sta ključni zanesljivost in celovitost podatkov.

Redis:

- **Predpomnilniki:** Redis je pogosto uporabljen kot predpomnilnik za shranjevanje pogosto dostopnih podatkov, kar zmanjšuje obremenitev glavne baze podatkov in izboljša odzivni čas aplikacij.
- **Sistemi za sledenje in analitiko v realnem času:** Redis se uporablja v sistemih, kjer je pomembno, da se podatki obdelujejo v realnem času, kot so sistemi za sledenje dogodkom, obdelavo dnevniških datotek in sledenje uporabnikom.
- **Sistemi za razvrstitev in ocenjevanje:** Redisove urejene množice so idealne za implementacijo lestvic in sistemov za ocenjevanje, kjer je pomembna hitrost dostopa in razvrščanja podatkov.

Sklep

PostgreSQL in Redis sta močni orodji, ki služita različnim namenom v svetu podatkovnih baz. PostgreSQL izstopa kot robusten sistem za upravljanje relacijskih baz podatkov, ki je primeren za aplikacije, kjer so ključne zanesljivost, celovitost in podpora za kompleksne poizvedbe. Redis, po drugi strani, blesti pri hitri obdelavi podatkov v realnem času, kjer sta ključni nizka zakasnitev in enostavnost uporabe. Razumevanje razlik med arhitekturami PostgreSQL in Redis je ključno za izbiro prave tehnologije glede na specifične potrebe aplikacij, kar omogoča optimalno delovanje informacijskih sistemov v različnih industrijah.

2.3.4 Primerjava zmogljivosti

Primerjava zmogljivosti med PostgreSQL in Redisom je ključna za razumevanje, kako vsak od teh dveh sistemov za upravljanje podatkovnih baz deluje v različnih scenarijih. Zmogljivost se nanaša na sposobnost sistema za obdelavo določene količine dela v določenem času, pri čemer so pomembni dejavniki, kot so hitrost poizvedb, odzivni časi, obvladovanje sočasnosti (angl. concurrency) in skalabilnost. Medtem ko je PostgreSQL zasnovan za obvladovanje kompleksnih transakcij in zagotavljanje doslednosti pri obdelavi velikih količin podatkov, je Redis optimiziran za hitrost in nizko zakasnitev pri obdelavi podatkov v realnem času. V tem podpoglavju bomo podrobno analizirali zmogljivost obeh sistemov v različnih kontekstih in uporabili rezultate za razumevanje, kdaj in kje je kateri od sistemov bolj primeren.

Hitrost poizvedb in obdelava podatkov

PostgreSQL je optimiziran za obdelavo kompleksnih poizvedb, ki vključujejo združevanje tabel, filtriranje podatkov, agregacije in druge zahtevne operacije. Zaradi svoje relacijske narave PostgreSQL zagotavlja visoko stopnjo natančnosti in doslednosti pri obdelavi podatkov, vendar to lahko vpliva na hitrost poizvedb, zlasti ko gre za obdelavo velikih količin podatkov ali zapletenih transakcij.

- **Poizvedbe in transakcije:** PostgreSQL je zelo učinkovit pri obdelavi kompleksnih SQL poizvedb, ki vključujejo več tabel, podpoizvedbe in agregacije. Zmogljivost PostgreSQL se močno opira na optimizacijo poizvedb, uporabo indeksov in pravilno načrtovanje baze podatkov. Pri obdelavi velike količine podatkov PostgreSQL uporablja večverzijsko hrambo (MVCC), ki omogoča hkratno izvajanje poizvedb in posodobitev brez zaklepanja tabel, kar povečuje zmogljivost in zmanjšuje konflikte med transakcijami.
- **Hitrost pisanja:** V primerih, ko so poizvedbe preproste ali ko gre za obdelavo manjših količin podatkov, lahko PostgreSQL zagotovi visoko zmogljivost. Vendar pa se hitrost pisanja lahko zmanjša, ko gre za velike transakcije ali veliko število sočasnih uporabnikov, zlasti če so potrebna obsežna zaklepanja ali sinhronizacija podatkov.
- **Uporaba indeksov:** Indeksiranje je ključno za zmogljivost PostgreSQL. Z uporabo primernih indeksov, kot so B-tree, GiST ali GIN, je mogoče znatno izboljšati hitrost poizvedb, zlasti pri poizvedbah, ki zahtevajo filtriranje ali združevanje velike količine podatkov.

Redis je zasnovan za izjemno hitrost, saj se vsi podatki obdelujejo neposredno v pomnilniku. To pomeni, da so poizvedbe in operacije pisanja v Redisu skoraj trenutne, kar je ključna prednost tega sistema.

- **Poizvedbe in operacije:** Redis je izjemno hiter pri preprostih operacijah ključ-vrednost, kot so shranjevanje, pridobivanje, brisanje in posodabljanje podatkov. Ker vse operacije potekajo v pomnilniku, so odzivni časi izjemno kratki, običajno v milisekundah ali mikrosekundah. To omogoča Redisu, da obvladuje visoko število poizvedb na sekundo, kar je idealno za aplikacije, ki zahtevajo hitro obdelavo podatkov v realnem času, kot so predpomnilniki, sistemi za sledenje dogodkom in sistemi za obdelavo podatkov v realnem času.

- **Hitrost pisanja:** Redis je optimiziran za hitro pisanje, kar omogoča, da sistem zlahka obvladuje milijone operacij na sekundo. Ta hitrost je še posebej pomembna pri aplikacijah, kjer je treba shranjevati podatke, kot so dnevniki dogodkov, v realnem času ali kjer je potrebno pogosto posodabljanje podatkov.
- **Enostavne poizvedbe:** Redis ni zasnovan za zapletene poizvedbe, kot so združevanje tabel ali kompleksne agregacije. Njegova zmogljivost temelji na enostavnosti podatkovnega modela in hitrem dostopu do podatkov, kar pomeni, da je najbolj učinkovit pri enostavnih poizvedbah in operacijah.

Obvladovanje sočasnosti (angl. Concurrency)

PostgreSQL je zasnovan za obvladovanje velikega števila sočasnih uporabnikov in transakcij. MVCC (angl. Multi-Version Concurrency Control) omogoča, da se poizvedbe in posodobitve izvajajo hkrati, ne da bi pri tem prišlo do zaklepanja tabel. To povečuje zmogljivost v okolju z več uporabniki in zmanjšuje verjetnost konfliktov.

- **Sočasne transakcije:** PostgreSQL lahko obvladuje veliko število sočasnih transakcij, saj MVCC omogoča, da vsaka transakcija vidi svoj »snapshot« baze podatkov, ne da bi vplivala na druge transakcije. To povečuje učinkovitost v okoljih, kjer veliko uporabnikov hkrati dostopa do podatkov in izvaja različne operacije.
- **Zaklepanja:** Čeprav PostgreSQL poskuša čim bolj zmanjšati zaklepanja, so ta včasih potrebna, zlasti pri kompleksnih transakcijah, ki vključujejo več tabel ali kjer je potrebna stroga doslednost podatkov. Zaklepanja lahko zmanjšajo zmogljivost v primerih, ko je veliko sočasnih posodobitev ali če pride do konfliktov med transakcijami.

Redis uporablja enotni proces in enotno nit za obdelavo vseh operacij, kar pomeni, da se operacije izvajajo zaporedno in ne pride do sočasnih operacij. Ta zasnova omogoča enostavno obvladovanje sočasnosti, saj ni potrebna nobena sinhronizacija ali zaklepanje med operacijami.

- **Zaporedno izvajanje:** V Redisu se vse operacije izvajajo zaporedno, kar pomeni, da ni konkurence med operacijami. To povečuje predvidljivost in stabilnost sistema, saj vsaka operacija poteka v izolaciji, brez vpliva na druge operacije.

- **Enostavnost obvladovanja:** Ker Redis ne podpira zapletenih transakcij ali sočasnih operacij, je obvladovanje sočasnosti preprosto. Vendar pa to pomeni, da Redis ni primeren za aplikacije, kjer je potrebna zapletena obdelava transakcij ali kjer več uporabnikov hkrati dostopa do istih podatkov na način, ki zahteva visoko stopnjo doslednosti.

Skalabilnost

PostgreSQL omogoča skaliranje na več načinov, vključno z vertikalnim in horizontalnim skaliranjem. Skalabilnost PostgreSQL temelji na replikaciji, porazdelitvi podatkov in optimizaciji poizvedb.

- **Vertikalno skaliranje:** PostgreSQL omogoča vertikalno skaliranje z dodajanjem več virov (procesorjev, pomnilnika, diskovnega prostora) na en strežnik. To omogoča, da sistem obvladuje večje količine podatkov in večje število sočasnih uporabnikov.
- **Horizontalno skaliranje:** PostgreSQL podpira horizontalno skaliranje z uporabo replikacije in particioniranja. Replikacija omogoča, da se podatki kopirajo na več strežnikov, kar povečuje zmogljivost in odpornost sistema. Particioniranje omogoča, da se velike tabele razdelijo na manjše dele, kar izboljša zmogljivost poizvedb in olajša upravljanje velikih podatkovnih zbirk.

Redis je zasnovan za enostavno horizontalno skaliranje, kar pomeni, da lahko sistem razdelite na več strežnikov, da povečate zmogljivost in razpoložljivost.

- **Redis Cluster:** Redis podpira horizontalno skaliranje prek Redis Clustera, ki omogoča razdelitev podatkov med več strežnikov. Vsak strežnik (ali vozlišče) v grozdu hrani del celotne zbirke podatkov, kar omogoča obvladovanje zelo velikih količin podatkov in visoko število operacij na sekundo.
- **Replikacija:** Redis podpira asinhrono replikacijo, kjer lahko en glavni strežnik replicira svoje podatke na več podrejenih strežnikov. Ta funkcionalnost omogoča povečanje razpoložljivosti sistema in zagotavlja, da so podatki vedno na voljo, tudi v primeru izpada glavnega strežnika.

Poraba virov in učinkovitost

PostgreSQL je zasnovan za optimalno izkoriščanje sistemskih virov, vendar njegova kompleksna narava pomeni, da lahko poraba virov variira glede na vrsto operacij.

- **Pomnilnik in procesor:** PostgreSQL porablja pomnilnik in procesorsko moč za obdelavo kompleksnih poizvedb, transakcij in zagotavljanje doslednosti podatkov. Učinkovitost sistema je močno odvisna od optimizacije poizvedb, uporabe indeksov in konfiguracije baze podatkov.
- **I/O operacije:** Ker PostgreSQL shranjuje podatke na disk, je učinkovitost sistema odvisna tudi od hitrosti diskovnih operacij. Optimizacija poizvedb, kot je zmanjšanje števila dostopov do diska, lahko znatno izboljša zmogljivost.

Redis je zasnovan za izjemno učinkovitost pri porabi virov, saj se vsi podatki shranjujejo in obdelujejo v pomnilniku.

- **Pomnilnik:** Redisova največja prednost in hkrati omejitev je odvisnost od pomnilnika. Čeprav to omogoča izjemno hitrost obdelave podatkov, omejuje količino podatkov, ki jih sistem lahko obvladuje, glede na razpoložljiv pomnilnik.
- **Procesor:** Ker Redis deluje v enotni niti, je poraba procesorske moči linearna glede na število operacij. Redis je zelo učinkovit pri obdelavi velikega števila enostavnih operacij, vendar se lahko zmogljivost zmanjša, če se sistem preobremeni z zahtevnimi operacijami ali če podatki presežejo razpoložljiv pomnilnik.

Sklep

Primerjava zmogljivosti med PostgreSQL in Redisom jasno kaže na različne prednosti in slabosti vsakega sistema, ki so močno odvisne od specifičnih zahtev aplikacij. PostgreSQL ponuja visoko zmogljivost pri obdelavi kompleksnih poizvedb in transakcij, kjer je ključna doslednost podatkov, vendar njegova kompleksnost in odvisnost od diskovnih operacij lahko vplivata na hitrost pri velikih obremenitvah. Po drugi strani je Redis izjemno hiter in učinkovit pri obdelavi enostavnih operacij ključ-vrednost v realnem času, kar je idealno za aplikacije, ki zahtevajo hitro obdelavo podatkov, vendar ima omejitve pri obvladovanju kompleksnih poizvedb in večjih količin podatkov, ki presegajo zmogljivost pomnilnika.

Razumevanje teh razlik je ključno za izbiro pravega sistema za določene aplikacije. PostgreSQL je najbolj primeren za scenarije, kjer so ključne natančnost, celovitost in obdelava kompleksnih podatkov, medtem ko Redis izstopa v aplikacijah, kjer je hitrost bistvena in kjer so poizvedbe preproste, a pogoste. V mnogih primerih se ta dva sistema lahko uporabljata komplementarno, kjer Redis služi kot predpomnilnik ali sistem za hitro obdelavo podatkov, medtem ko PostgreSQL skrbi za dolgoročno shranjevanje in obdelavo podatkov, ki zahtevajo visoko stopnjo doslednosti in analitične zmožnosti.

2.3.5 Podpora in skupnost uporabnikov

Podpora in skupnost uporabnikov sta ključna dejavnika pri izbiri katere koli tehnologije, saj vplivata na dostopnost virov, izkušenj in pomoči, ki jo lahko uporabniki dobijo pri reševanju težav, učenju in optimizaciji svojega dela. Tako PostgreSQL kot Redis imata močne skupnosti in obsežne vire podpore, ki so se razvijali skozi leta. V tem podpoglavju bomo podrobno analizirali naravo in obseg podpore ter skupnosti uporabnikov za oba sistema, pri čemer bomo raziskali različne vire, orodja, dokumentacijo in načine, kako se uporabniki lahko vključijo in pridobijo pomoč.

Podpora in skupnost PostgreSQL

PostgreSQL, kot ena najstarejših in najbolj uveljavljenih odprtokodnih relacijskih baz podatkov, ima zelo obsežno in dobro organizirano skupnost, ki zagotavlja širok spekter podpore za uporabnike na vseh ravneh znanja in izkušenj.

- **Dokumentacija:** PostgreSQL se ponaša z obsežno in podrobno dokumentacijo, ki pokriva vse vidike sistema, od osnovne namestitve in konfiguracije do napredne optimizacije poizvedb, varnosti in prilagajanja baze podatkov. Dokumentacija je na voljo brezplačno na uradni spletni strani PostgreSQL in je redno posodobljena z novimi izdajami.
  - **Uradna dokumentacija:** Uradna dokumentacija PostgreSQL je izjemno temeljita in vključuje podrobne opise vseh funkcionalnosti, primerov uporabe in najboljše prakse. Uporabniki lahko najdejo informacije o vseh vidikih sistema, vključno z administracijo, programiranjem, optimizacijo zmogljivosti in varnostjo.

- **Knjižnice in viri:** Poleg uradne dokumentacije so na voljo številne knjige, vodiči, tečaji in članki, ki jih pripravljajo strokovnjaki in člani skupnosti. Med priljubljene vire sodijo knjige, kot sta »PostgreSQL: Up and Running« in »Mastering PostgreSQL«, ki ponujajo poglobljeno znanje in praktične primere.

- **Forumi in diskusijske skupine:** PostgreSQL ima zelo aktivno skupnost na različnih forumih in diskusijskih skupinah, kjer lahko uporabniki postavljajo vprašanja, delijo svoje izkušnje in se učijo od drugih.
  - **PostgreSQL mailing liste:** PostgreSQL mailing liste so že dolgo glavno sredstvo komunikacije znotraj skupnosti. Obstajajo različne liste, osredotočene na različne teme, kot so uporabniki, razvijalci, zmogljivost, varnost in lokalizacija. Uporabniki lahko te liste uporabljajo za iskanje odgovorov na specifična vprašanja ali za sodelovanje v diskusijah o novih funkcionalnostih in izboljšavah sistema.
  - **Forumi in spletne skupnosti:** Poleg uradnih mailing list so priljubljene tudi spletne platforme, kot so Stack Overflow, Reddit (r/PostgreSQL) in PostgreSQL forum, kjer lahko uporabniki dobijo hitro pomoč in nasvete od drugih članov skupnosti. Te platforme omogočajo takojšnjo komunikacijo in so odličen vir za iskanje odgovorov na pogosta vprašanja ali za učenje na podlagi izkušenj drugih.

- **Komercialna podpora:** Poleg brezplačne podpore, ki jo zagotavlja skupnost, je za PostgreSQL na voljo tudi komercialna podpora, ki jo nudijo različna podjetja po vsem svetu. Ta podjetja ponujajo strokovne storitve, kot so namestitev, konfiguracija, optimizacija, migracija, varnostno svetovanje in razvoj specifičnih rešitev.
  - **Podjetja, ki nudijo podporo:** Podjetja, kot so EnterpriseDB, 2ndQuadrant (zdaj del EDB), Crunchy Data in druga, ponujajo plačljivo podporo za PostgreSQL, ki vključuje 24/7 pomoč, prilagojene rešitve in dolgoročno vzdrževanje sistema. Ta podpora je ključna za podjetja, ki uporabljajo PostgreSQL v kritičnih poslovnih aplikacijah in potrebujejo zanesljiv dostop do strokovnjakov.
  - **Certifikati in usposabljanje:** Številni ponudniki komercialne podpore ponujajo tudi certificirane tečaje in usposabljanja za PostgreSQL, kar omogoča strokovnjakom, da

pridobijo poglobljeno znanje in uradno priznanje za svoje spretnosti pri delu s PostgreSQL.

- **Konference in srečanja:** PostgreSQL skupnost organizira številne konference, delavnice in srečanja po vsem svetu, kjer se lahko uporabniki srečajo, izmenjajo ideje in se učijo od strokovnjakov.
  - **PGConf in PGDay:** PostgreSQL konferenca (PGConf) in manjša lokalna srečanja, imenovana PGDay, so priljubljeni dogodki, kjer se zbirajo razvijalci, administratorji in uporabniki PostgreSQL. Ti dogodki ponujajo predavanja, delavnice in priložnosti za mreženje, kar omogoča udeležencem, da ostanejo na tekočem z najnovejšimi trendi in praksami v skupnosti.
  - **Meetupi:** Poleg velikih konferenc so po vsem svetu organizirani tudi manjši meetupi, ki jih vodi lokalna skupnost PostgreSQL. Ti dogodki so odlična priložnost za učenje, reševanje specifičnih težav in povezovanje z drugimi uporabniki.

Podpora in skupnost Redisa

Redis, kot izjemno priljubljena odprtokodna baza podatkov za shranjevanje podatkov v pomnilniku, ima prav tako močno podporno skupnost in številne vire, ki so na voljo uporabnikom. Skupnost Redis je znana po svoji aktivni vlogi pri razvoju, izmenjavi znanja in pomoči pri reševanju težav.

- **Dokumentacija:** Redis ima dobro organizirano in lahko dostopno dokumentacijo, ki pokriva širok spekter tem, od osnovne uporabe do naprednih konfiguracij in optimizacij.
  - **Uradna dokumentacija:** Redisova uradna dokumentacija je obsežna in uporabnikom nudi podrobna navodila za namestitev, konfiguracijo, uporabo in optimizacijo sistema. Dokumentacija je jasno strukturirana in redno posodobljena, kar uporabnikom omogoča, da hitro najdejo informacije, ki jih potrebujejo.
  - **Dodatni viri:** Poleg uradne dokumentacije so na voljo številne knjige, vodiči in spletni tečaji, ki jih pripravljajo strokovnjaki za Redis. Knjige, kot sta »Redis in Action« in

»Mastering Redis«, ponujajo poglobljeno razumevanje Redisa in praktične primere uporabe v različnih scenarijih.

- **Forumi in diskusijske skupine:** Redis ima zelo aktivno skupnost na različnih spletnih platformah, kjer uporabniki lahko izmenjujejo izkušnje, postavljajo vprašanja in pridobijo pomoč.
  - **Redis mailing liste:** Redisova mailing lista je ključen kanal za komunikacijo znotraj skupnosti. Uporabniki lahko prek mailing liste zastavijo vprašanja, delijo izkušnje in spremljajo razvoj sistema. To je odlična platforma za pridobivanje neposredne pomoči od razvijalcev in drugih izkušenih uporabnikov.
  - **Forumi in spletne skupnosti:** Platforme, kot so Stack Overflow, Reddit (r/redis) in uradni Redis forum, omogočajo hitro komunikacijo in izmenjavo znanja med uporabniki. Te platforme so odlične za iskanje odgovorov na pogosta vprašanja, deljenje skriptov in primerov uporabe ter reševanje specifičnih težav, s katerimi se srečujejo uporabniki v različnih aplikacijah.

- **Komercialna podpora:** Redis, podobno kot PostgreSQL, ponuja možnosti za komercialno podporo, kar je še posebej pomembno za podjetja, ki uporabljajo Redis v produkcijskih okoljih, kjer je potrebna stalna in zanesljiva podpora.
  - **Redis Labs:** Redis Labs, ki je glavni skrbnik Redisa, ponuja Redis Enterprise, komercialno različico Redisa z dodatnimi funkcionalnostmi, kot so visoka razpoložljivost, avtomatsko preklapljanje (angl. failover), varnostne funkcije in podpora za trajnost. Redis Labs ponuja tudi 24/7 podporo, upravljanje oblačnih storitev in prilagojene rešitve za stranke.
  - **Drugi ponudniki:** Poleg Redis Labs obstajajo tudi drugi ponudniki komercialne podpore za Redis, kot so Aiven, AWS in DigitalOcean, ki nudijo Redis kot storitev (Redis-as-a-Service) z vključeno podporo in vzdrževanjem. Ti ponudniki omogočajo podjetjem, da uporabljajo Redis brez skrbi glede upravljanja infrastrukture.

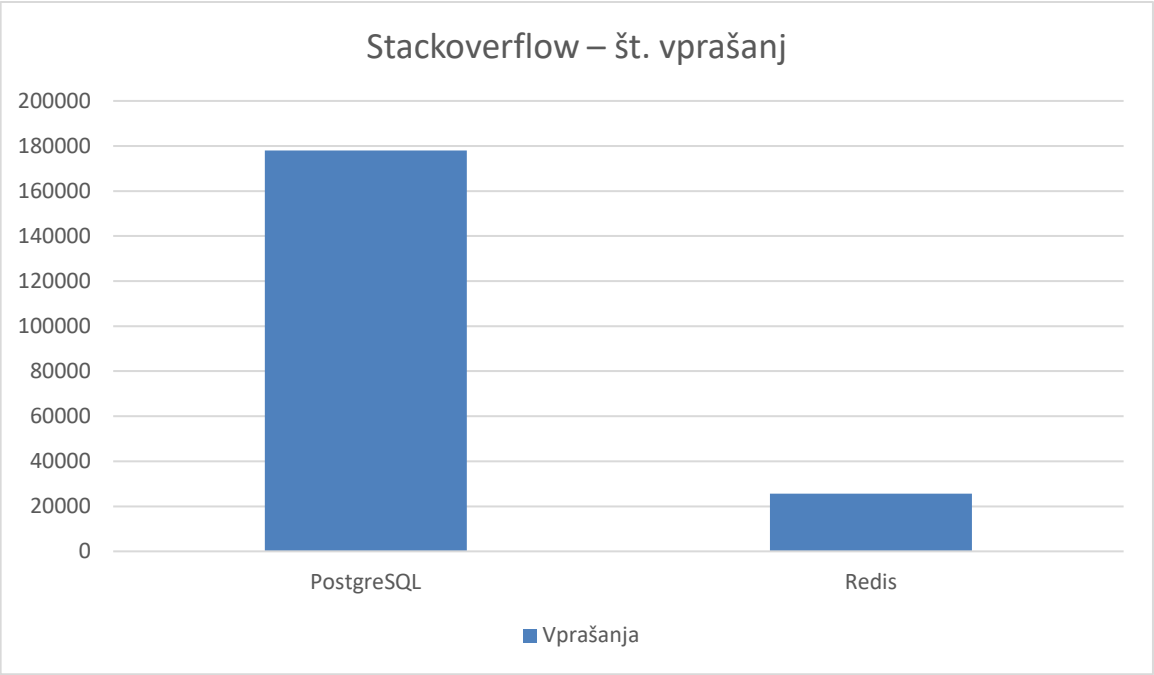
- **Konference in srečanja:** Redis ima svojo skupnost, ki organizira konference, delavnice in srečanja, kjer se lahko uporabniki učijo od strokovnjakov in izmenjujejo ideje.
    - **RedisConf:** RedisConf je glavna konferenca, ki jo vsako leto organizira Redis Labs. Dogodek privablja razvijalce, administratorje in uporabnike Redisa z vsega sveta, ki se udeležujejo predavanj, delavnic in predstavitev najnovejših funkcionalnosti in primerov uporabe Redisa.
    - **Regionalni dogodki in meetupi:** Redis meetupi so organizirani po vsem svetu, kjer se uporabniki srečujejo, izmenjujejo znanje in razpravljajo o najboljših praksah pri uporabi Redisa v različnih scenarijih. Ti dogodki so odličen način za pridobivanje praktičnih nasvetov in navezovanje stikov z drugimi strokovnjaki v skupnosti.

Primerjava podpore in skupnosti PostgreSQL in Redisa

Pri primerjavi podpore in skupnosti za PostgreSQL in Redis je jasno, da imata oba sistema močne in aktivne skupnosti, vendar se razlikujeta glede na naravo svojih uporabnikov in fokus podpore.

Stack Overflow primerjava

Na platformi Stack Overflow, ki je eden največjih virov za tehnično podporo razvijalcem, je PostgreSQL precej bolj zastopan kot Redis. Do danes je bilo zastavljenih več kot 178.000 vprašanj, povezanih s PostgreSQL, kar kaže na široko bazo uporabnikov in pogoste izzive, s katerimi se srečujejo pri uporabi te baze. Redis, čeprav zelo priljubljen v določenih scenarijih, ima bistveno manj vprašanj – nekaj več kot 25.600. To nakazuje na manjšo skupnost aktivnih uporabnikov v primerjavi s PostgreSQL.

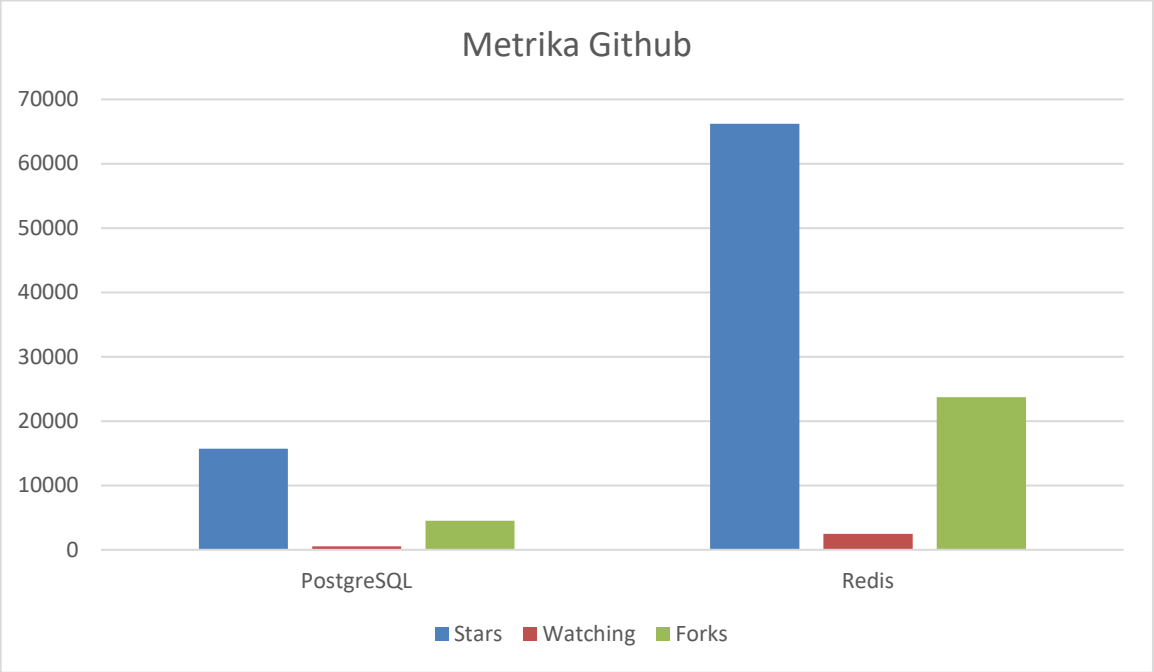


Graf 1: Primerjava števila Stack Overflow vprašanj za PostgreSQL in Redis

Vir: (https://stackoverflow.com, 2024)

Metrika GitHub

Kljub široki uporabi PostgreSQL na forumih in v industriji Redis vodi v priljubljenosti na platformi GitHub, kjer je projekt Redis prejel več kot 66.000 zvezdic, medtem ko je PostgreSQL zbral 15.700 zvezdic. Redis ima tudi več kot 23.700 forkov, kar kaže na pogostejše prilagajanje in eksperimentiranje z izvorno kodo v primerjavi s PostgreSQL, ki ima okoli 4500 fork-ov.



Graf 2: Primerjava priljubljenosti na GitHubu

Vir: (https://github.com, 2024)

DB-Engines Ranking

Na lestvici DB-Engines (DB-Engines, 2024), ki ocenjuje priljubljenost baz podatkov na podlagi več kriterijev, kot so spletne poizvedbe, uporaba v industriji in prispevki, je PostgreSQL dosledno visoko uvrščen. Trenutno je PostgreSQL na četrtem mestu z rezultatom 644,36, medtem ko je Redis na šestem mestu z rezultatom 149,43. Ta razlika kaže, da je PostgreSQL bolj priljubljen in širše uporabljen v industriji.

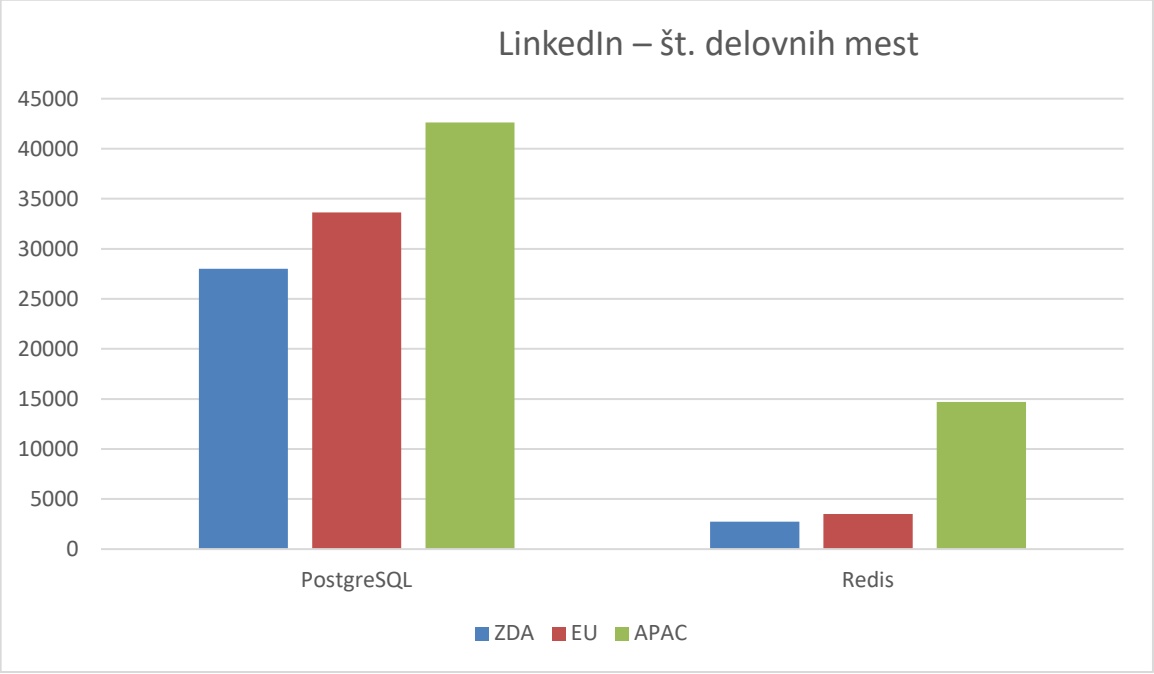
Tabela 1: Lestvica priljubljenosti baz podatkov na DB-Engines

Rank	DBMS	Database Model	Score
1	Oracle	Relational	1286,59
2	MySQL	Relational	1029,49
3	Microsoft SQL Server	Relational	807,76
4	PostgreSQL	Relational	644,36
5	MongoDB	Document	410,24
6	Redis	Key-value	149,43
7	Snowflake	Relational	133,72
8	Elasticsearch	Search engine	128,79
9	IBM Db2	Relational	123,05
10	SQLite	Relational	103,35

Vir: (https://db-engines.com/en/ranking, 2024)

Delovna mesta na LinkedInu

Analiza delovnih mest na platformi LinkedIn kaže na znatno večje povpraševanje po strokovnjakih za PostgreSQL v primerjavi z Redisom, kar odraža širšo uporabo PostgreSQL v poslovnih okoljih in industriji. V Združenih državah Amerike je za PostgreSQL objavljenih več kot 28.000 delovnih mest, medtem ko je za Redis objavljenih nekaj več kot 2700 delovnih mest. Podobno je v Evropski uniji, kjer rezultati iskanja kažejo, da je za PostgreSQL na voljo več kot 33.600 delovnih mest, medtem ko je za Redis objavljenih več kot 3500 delovnih mest. V regiji Azija-Pacifik (APAC) pa je za PostgreSQL razpisanih več kot 42.600 delovnih mest, medtem ko je za Redis na voljo več kot 14.600 delovnih mest. Ti podatki jasno nakazujejo, da je PostgreSQL splošno bolj uveljavljen in iskan v različnih industrijskih panogah in geografskih območjih, medtem ko se Redis uporablja v bolj specifičnih primerih.



Graf 3: Število objavljenih delovnih mest za obe bazi na LinkedIn glede na regijo

Vir: (https://www.linkedin.com, 2024)

Sklep

Na podlagi analize različnih virov je očitno, da ima PostgreSQL širšo skupnost in večjo podporo v primerjavi z Redisom. PostgreSQL je močno zastopan na forumih, kot je Stack Overflow, in ima večje povpraševanje na trgu dela, medtem ko Redis izstopa na platformi GitHub, kjer beleži višjo priljubljenost v odprtokodnih projektih. Kljub temu PostgreSQL ohranja prednost v večini meritev, kar potrjuje hipotezo, da ima večjo in bolj raznoliko skupnost uporabnikov, ki jo podpirajo številni komercialni ponudniki in strokovnjaki.

2.3.6 Uporaba PostgreSQL in Redisa v umetni inteligenci

Umetna inteligenca (UI) je ena izmed najhitreje rastočih področij v tehnologiji, kjer podatkovne baze igrajo ključno vlogo pri shranjevanju, obdelavi in dostopu do podatkov, ki so potrebni za učenje, treniranje in izvajanje modelov umetne inteligence. PostgreSQL in Redis, kot dve zelo različni podatkovni rešitvi, ponujata različne pristope in orodja za podporo aplikacijam, ki temeljijo na umetni inteligenci. V tem podpoglavju bomo podrobno raziskali, kako se PostgreSQL in Redis uporabljata v kontekstu umetne inteligence, pri čemer bomo poudarili njihove prednosti, slabosti in primere uporabe v različnih scenarijih.

Uporaba PostgreSQL v umetni inteligenci

PostgreSQL, kot robusten sistem za upravljanje relacijskih baz podatkov, igra ključno vlogo v številnih aplikacijah umetne inteligence, kjer je treba upravljati velike količine strukturiranih podatkov in izvajati kompleksne analitične poizvedbe. PostgreSQL ponuja več funkcionalnosti, ki so še posebej uporabne v kontekstu umetne inteligence.

Shranjevanje in upravljanje strukturiranih podatkov

Umetna inteligenca se pogosto zanaša na velike količine strukturiranih podatkov, ki se uporabljajo za treniranje modelov, preverjanje hipotez in pridobivanje vpogledov iz podatkov. PostgreSQL omogoča učinkovito shranjevanje teh podatkov v relacijske tabele, kjer so lahko organizirani in indeksirani na način, ki omogoča hitro iskanje in obdelavo.

- **Podpora za napredne tipe podatkov:** PostgreSQL podpira napredne tipe podatkov, kot so JSON, XML in ARRAY, kar omogoča shranjevanje kompleksnih struktur podatkov, ki so pogosto prisotne v aplikacijah umetne inteligence. Na primer, podatki o uporabniških interakcijah, zabeleženih v formatu JSON, se lahko enostavno shranjujejo in poizvedujejo v PostgreSQL, kar omogoča hitre analize in pripravo podatkov za treniranje modelov UI.
- **Indeksiranje:** PostgreSQL omogoča napredno indeksiranje, vključno z B-tree, GIN in GiST indeksi, kar povečuje zmogljivost poizvedb in omogoča hitrejše iskanje podatkov, ki so potrebni za umetno inteligenco. To je še posebej uporabno pri poizvedbah, kjer je treba iz velikih podatkovnih zbirk hitro pridobiti relevantne podatke za analizo ali treniranje modelov.

Integracija z orodji za analizo podatkov in strojno učenje

PostgreSQL se enostavno integrira z različnimi orodji za analizo podatkov in strojno učenje, kot sta Python, R, in različne knjižnice za strojno učenje, kot so scikit-learn, TensorFlow in PyTorch.

- **Funkcije PostgreSQL in shranjene procedure:** Z uporabo shranjenih procedur in funkcij, napisanih v jeziku PL/pgSQL ali drugih jezikih, kot sta PL/Python ali PL/R, lahko uporabniki izvajajo kompleksne izračune neposredno v bazi podatkov. To omogoča, da se del analize ali predobdelave podatkov izvede že na ravni baze podatkov, kar zmanjšuje potrebo po prenosu velikih količin podatkov med bazo in aplikacijo.
- **Povezljivost s sistemi za analizo:** PostgreSQL omogoča enostavno povezovanje z orodij za analizo podatkov prek ODBC, JDBC ali neposredne integracije z aplikacijami, napisanimi v Pythonu ali drugih programskih jezikih. To omogoča raziskovalcem in razvijalcem, da enostavno dostopajo do podatkov, shranjenih v PostgreSQL in jih uporabljajo za treniranje modelov umetne inteligence.

Analitika in podatkovno skladiščenje

PostgreSQL se pogosto uporablja kot podatkovno skladišče, kjer se zbirajo in analizirajo podatki iz različnih virov. To je še posebej uporabno v umetni inteligenci, kjer se podatki iz različnih virov (npr. senzorji, spletni dnevniki, transakcijski podatki) združujejo in analizirajo za pridobivanje vpogledov.

- **Podatkovno skladiščenje:** PostgreSQL omogoča shranjevanje velikih količin podatkov v podatkovna skladišča, kjer se podatki zbirajo, transformirajo in pripravljajo za analizo. To je še posebej pomembno za umetno inteligenco, kjer so podatki iz različnih virov združeni in analizirani za treniranje modelov.
- **Analitične poizvedbe:** PostgreSQL podpira napredne analitične funkcije, kot so okenske funkcije, agregacijske poizvedbe in CTE (angl. Common Table Expressions), ki omogočajo kompleksne analize podatkov. To omogoča uporabnikom, da pridobijo vpoglede iz podatkov, ki so ključni za razvoj in optimizacijo modelov umetne inteligence.

Uporaba Redisa v umetni inteligenci

Redis, kot sistem za shranjevanje podatkov v pomnilniku, se pogosto uporablja v aplikacijah umetne inteligence, kjer je ključna hitrost dostopa do podatkov in nizka zakasnitev. Redis ponuja več funkcionalnosti, ki so še posebej uporabne v aplikacijah umetne inteligence, zlasti v realnočasovnih aplikacijah in sistemih, kjer sta potrebna hitro pridobivanje in obdelava podatkov.

Predpomnjenje modelov in podatkov

Ena izmed ključnih uporab Redisa v umetni inteligenci je predpomnjenje modelov in podatkov, kar omogoča hitrejši dostop do pogosto uporabljenih informacij in zmanjšuje obremenitev glavne baze podatkov ali strežnikov za strojno učenje.

- **Predpomnjenje podatkov:** Redis se pogosto uporablja za predpomnjenje rezultatov poizvedb, predobdelanih podatkov ali celotnih modelov umetne inteligence, kar omogoča hitrejši dostop do teh virov. Na primer, v aplikacijah, kjer je potrebno pogosto napovedovanje ali klasifikacija na podlagi določenega modela, se lahko model in podatki, potrebni za napovedovanje, shranijo v Redis za hitro pridobivanje in obdelavo.
- **Izvajanje modelov v realnem času:** Redis se lahko uporablja kot predpomnilnik za modele strojnega učenja, kjer so podatki o modelih in njihovi parametri shranjeni v pomnilniku, kar omogoča hitro izvajanje napovedi v realnem času. To je še posebej uporabno v aplikacijah, kot so personalizacija vsebine, sistemi za priporočanje in obdelava signalov v realnem času.

Sledenje in beleženje podatkov v realnem času

Redis se pogosto uporablja za sledenje dogodkov in beleženje podatkov v realnem času, kar omogoča aplikacijam umetne inteligence, da obdelujejo in analizirajo podatke takoj, ko so ti na voljo.

- **Sistemi za sledenje:** Redis se uporablja v sistemih za sledenje, kjer sta potrebna beleženje in obdelava dogodkov v realnem času. Na primer, v aplikacijah za spremljanje uporabniškega vedenja na spletu se lahko podatki o interakcijah uporabnikov s spletno stranjo shranjujejo v Redis, kar omogoča takojšnjo analizo in prilagoditev vsebine na podlagi teh podatkov.
- **Realnočasovna analitika:** Redis omogoča shranjevanje in obdelavo podatkov v realnem času, kar je ključnega pomena za aplikacije, ki temeljijo na umetni inteligenci in potrebujejo sprotne podatke za analizo in ukrepanje. Na primer, v finančnih aplikacijah se lahko Redis uporablja za sledenje tržnim podatkom v realnem času in izvajanje algoritmičnega trgovanja na podlagi teh podatkov.

Razširljivost in skalabilnost za aplikacije umetne inteligence

Redis omogoča enostavno razširitev in skaliranje, kar je ključno za aplikacije umetne inteligence, ki morajo obdelovati velike količine podatkov ali delovati v distribucijskih okoljih.

- **Redis Cluster:** Redis Cluster omogoča razdelitev podatkov med več strežnikov, kar povečuje zmogljivost sistema in omogoča obdelavo večjih količin podatkov. To je še posebej pomembno za aplikacije umetne inteligence, kjer se podatki pridobivajo in obdelujejo v realnem času na globalni ravni.
- **Podpora za porazdeljene operacije:** Redis podpira porazdeljene operacije, kot so razvrščanje podatkov in obdelava množic, kar omogoča enostavno obdelavo velikih količin podatkov v porazdeljenih okoljih. To omogoča aplikacijam umetne inteligence, da se razširijo na več strežnikov in obvladujejo večje obremenitve brez izgube zmogljivosti.

Primeri uporabe PostgreSQL in Redisa v umetni inteligenci

- **Priporočilni sistemi:** Priporočilni sistemi so ključna področja uporabe umetne inteligence, kjer se PostgreSQL in Redis pogosto uporabljata skupaj za izboljšanje učinkovitosti in natančnosti priporočil.
  - **PostgreSQL:** PostgreSQL se uporablja za shranjevanje in analizo velikih količin uporabniških podatkov, kot so zgodovina nakupov, interakcije s spletno stranjo in demografski podatki. Ti podatki se nato uporabijo za treniranje priporočilnih modelov, ki ustvarjajo prilagojene predloge za uporabnike.
  - **Redis:** Redis se uporablja za hitro pridobivanje in predpomnjenje priporočil, ki so bila izračunana s pomočjo modelov, shranjenih v PostgreSQL. Na primer, priporočila za izdelke na spletnem mestu se lahko predpomnijo v Redis, kar omogoča, da so ta priporočila takoj na voljo, ko se uporabnik prijavi ali obišče določeno stran.

- **Obdelava naravnega jezika (NLP):** Obdelava naravnega jezika je še eno pomembno področje umetne inteligence, kjer se uporabljata PostgreSQL in Redis.
  - **PostgreSQL:** PostgreSQL se uporablja za shranjevanje velikih korpusov besedilnih podatkov, ki se uporabljajo za treniranje modelov NLP. PostgreSQL podpira tudi iskanje po polnem besedilu, kar omogoča hitro iskanje in analizo besedilnih podatkov, ki so ključni za razvoj modelov za obdelavo naravnega jezika.
  - **Redis:** Redis se uporablja za predpomnjenje rezultatov modelov NLP in hitro obdelavo vhodnih podatkov v realnem času. Na primer, v aplikacijah za analizo sentimenta se lahko rezultati modelov shranjujejo v Redis, kar omogoča hitro obdelavo in prikaz rezultatov uporabnikom v realnem času.

- **Finančna analitika in algoritemsko trgovanje:** V finančni industriji se umetna inteligenca pogosto uporablja za algoritemsko trgovanje in finančno analitiko, kjer PostgreSQL in Redis igrata ključno vlogo.

- **PostgreSQL:** PostgreSQL se uporablja za shranjevanje zgodovinskih tržnih podatkov, transakcij in finančnih kazalnikov, ki se uporabljajo za treniranje modelov za napovedovanje cen, oceno tveganja in optimizacijo portfeljev.
- **Redis:** Redis se uporablja za hitro obdelavo in sledenje tržnim podatkom v realnem času, kar omogoča izvajanje trgovalnih strategij na podlagi trenutnih tržnih razmer. Redis lahko shranjuje trenutne tržne podatke in rezultate trgovalnih algoritmov, kar omogoča hitro prilagoditev strategij in zmanjšanje zakasnitve pri trgovanju.

Sklep

PostgreSQL in Redis igrata ključni vlogi v različnih aplikacijah umetne inteligence, pri čemer vsak sistem ponuja specifične prednosti glede na potrebe aplikacije. PostgreSQL je idealen za aplikacije, ki zahtevajo kompleksno upravljanje strukturiranih podatkov, analitične poizvedbe in integracijo z orodji za strojno učenje. Redis pa izstopa v aplikacijah, kjer sta ključni hitrost dostopa do podatkov in nizka zakasnitev, kar omogoča hitro izvajanje napovedi in analitiko v realnem času.

Kombinacija obeh sistemov omogoča izkoriščanje prednosti vsakega sistema, kar je ključnega pomena za uspešno implementacijo umetne inteligence v sodobnih aplikacijah. Medtem ko PostgreSQL zagotavlja zanesljivo in obsežno shranjevanje podatkov, Redis omogoča hitro in učinkovito obdelavo teh podatkov, kar omogoča optimizacijo zmogljivosti in natančnosti modelov umetne inteligence.

3 METODOLOGIJA RAZISKOVANJA

3.1 Priprava virtualnih okolij in baze podatkov

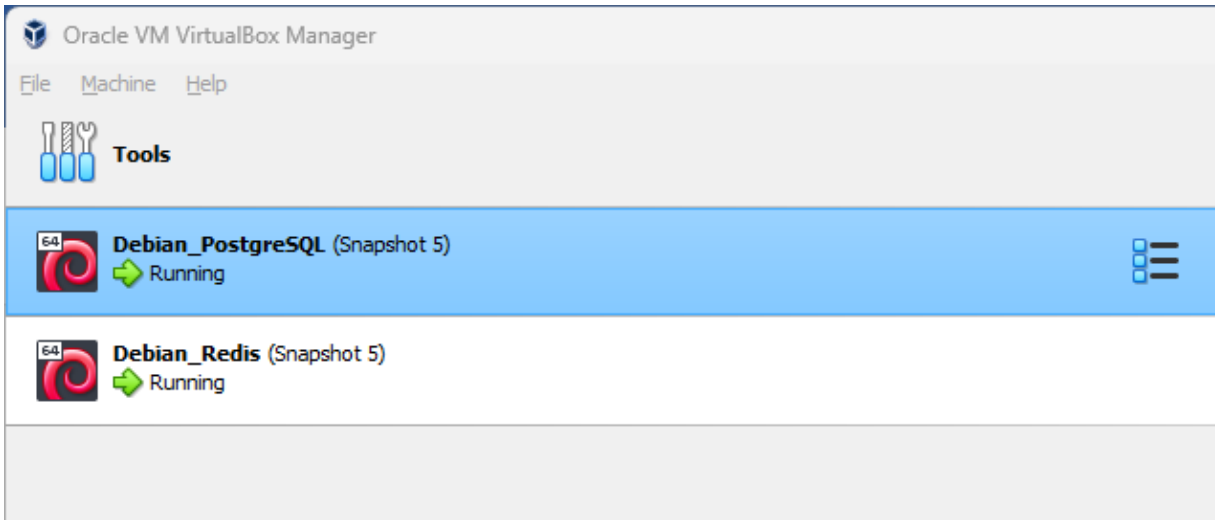
Za izvedbo eksperimentalne primerjave zmogljivosti podatkovnih baz PostgreSQL in Redis smo uporabili dve ločeni virtualni okolji, vsako namenjeno eni izmed podatkovnih baz. Virtualni strežniki omogočajo izolirano in nadzorovano okolje, kar je bistvenega pomena za natančno analizo delovanja obeh podatkovnih baz.

Ustvarjanje virtualnih strojev

Virtualizacijo smo izvedli s pomočjo Oracle VM VirtualBox Managerja, kjer smo ustvarili dve ločeni virtualni napravi:

- **Debian_PostgreSQL:** namenjena namestitvi in konfiguraciji podatkovne baze PostgreSQL.
- **Debian_Redis:** namenjena namestitvi in konfiguraciji podatkovne baze Redis.

Obe virtualni napravi sta zasnovani na operacijskem sistemu Debian 12.6.0, kar zagotavlja stabilno in zanesljivo okolje za izvajanje podatkovnih poizvedb in testiranje.



Slika 1: Virtualni napravi v Oracle VM

Vir: (Lastni vir)

Konfiguracija strojne opreme

Virtualni napravi imata identične strojne konfiguracije, da bi zagotovili enakovredne pogoje za primerjavo obeh podatkovnih baz:

- RAM: 4096 MB (4 GB),
- CPU: 2 jedri procesorja,
- Diskovni prostor: 50 GB,
- Mrežni adapter: NAT (angl. Network Address Translation), ki omogoča povezljivost virtualk s strežnikom za dostop do interneta.

Enaka konfiguracija obeh sistemov je ključna za zagotavljanje objektivnih rezultatov primerjave zmogljivosti, saj noben izmed sistemov ne uživa prednosti glede strojnih virov.



Slika 2: Konfiguracija strojne opreme Debian_PostgreSQL

Vir: (Lastni vir)

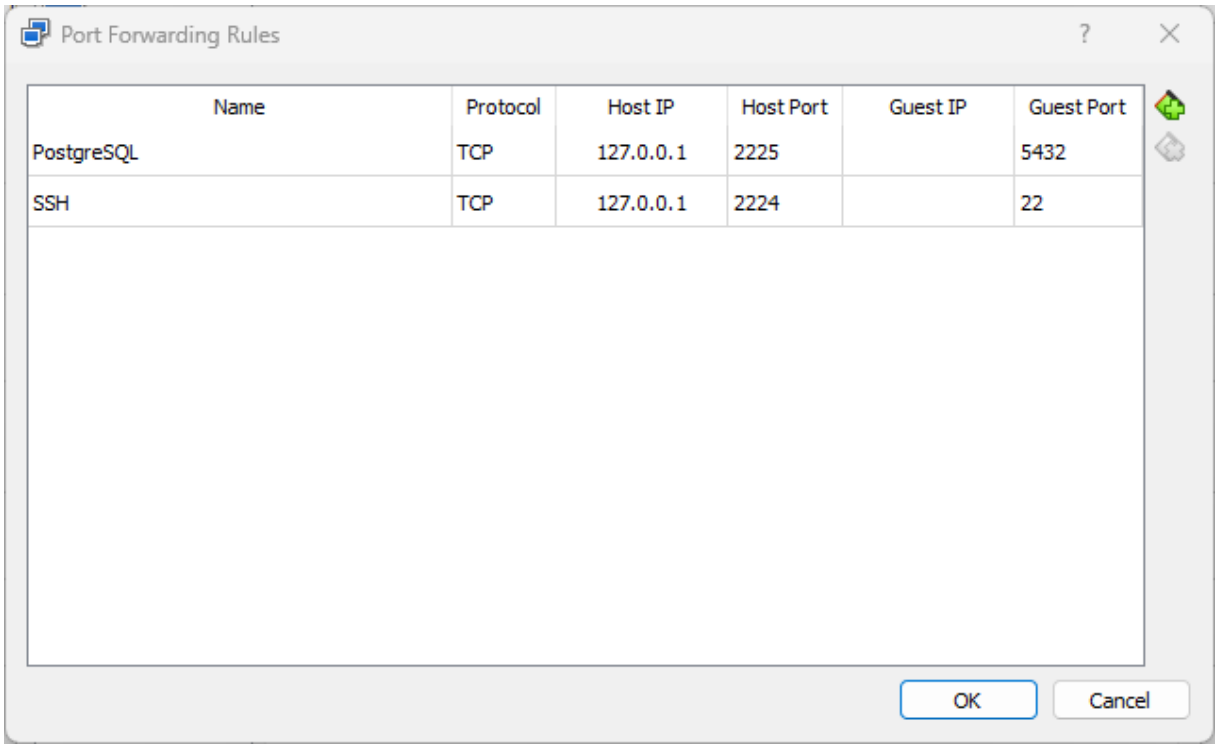


Slika 3: Konfiguracija strojne opreme Debian_Redis

Vir: (Lastni vir)

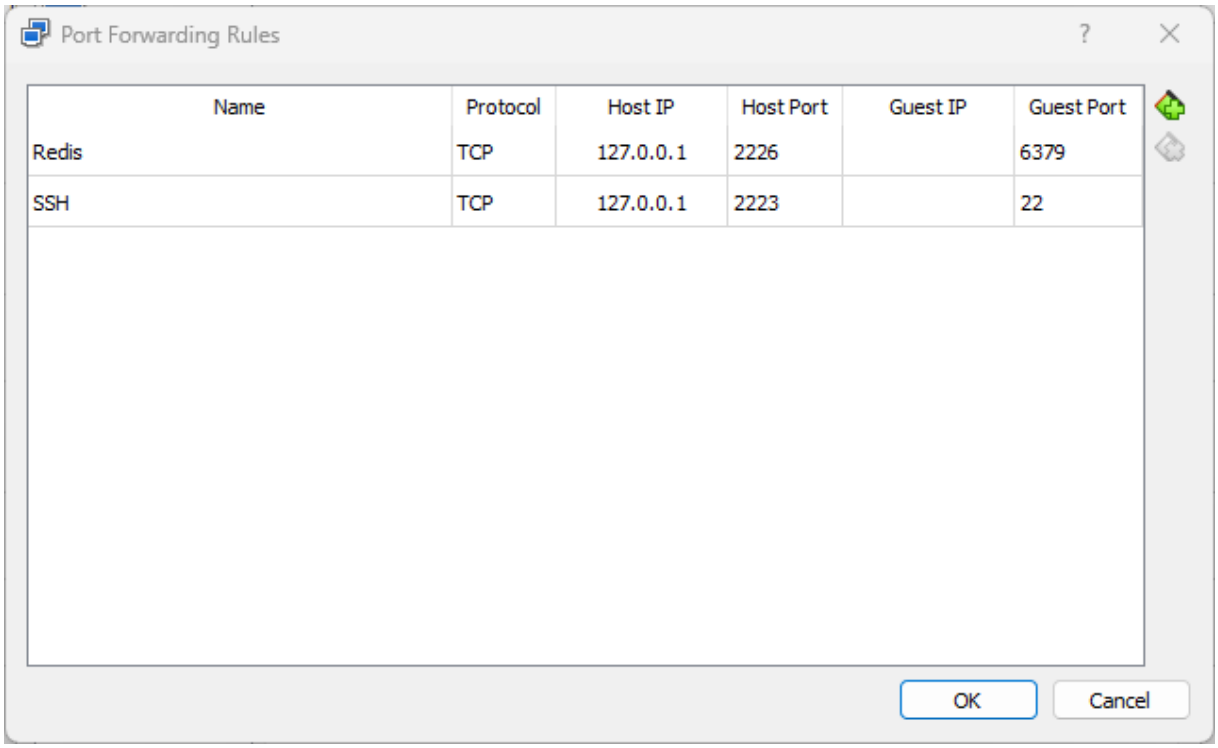
Omrežna konfiguracija

Da bi omogočili pravilno povezljivost z zunanjimi orodji, kot je DataGrip, smo uporabili pravila port-forwarding, ki omogočajo dostop do podatkovnih baz in SSH na lokalnem računalniku. Za PostgreSQL smo nastavili port-forwarding na Host Port 2225 za PostgreSQL (Guest Port 5432) in Host Port 2224 za SSH (Guest Port 22). Za Redis smo nastavili Host Port 2226 za Redis (Guest Port 6379) in Host Port 2223 za SSH (Guest Port 22).



Slika 4: PostgreSQL port-forwarding pravila

Vir: (Lastni vir)



Slika 5: Redis post-forwarding pravila

Vir: (Lastni vir)

Namestitev PostgreSQL in Redisa

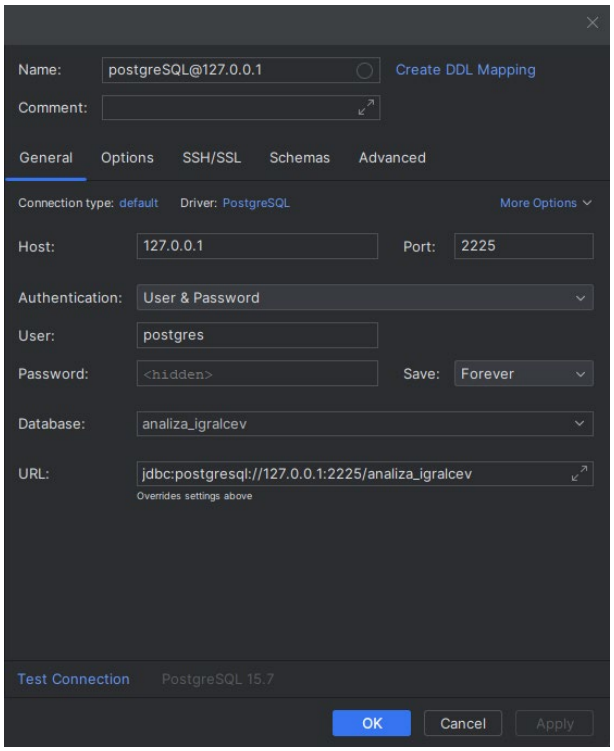
Na vsako virtualno napravo smo namestili ustrezno podatkovno bazo:

- Na Debian_PostgreSQL smo namestili podatkovno bazo PostgreSQL. Uporabljena različica je PostgreSQL 15.7 (Debian 15.7-0+deb12u1), ki je znana po svoji stabilnosti in prilagodljivosti pri obdelavi velikih količin podatkov.
- Na Debian_Redis smo namestili podatkovno bazo Redis. Uporabljena različica je Redis 7.0.15, ki je optimizirana za hiter dostop do podatkov v pomnilniku, kar omogoča visoko zmogljivost pri specifičnih vrstah poizvedb.

Dodatne nastavitve za lažji dostop

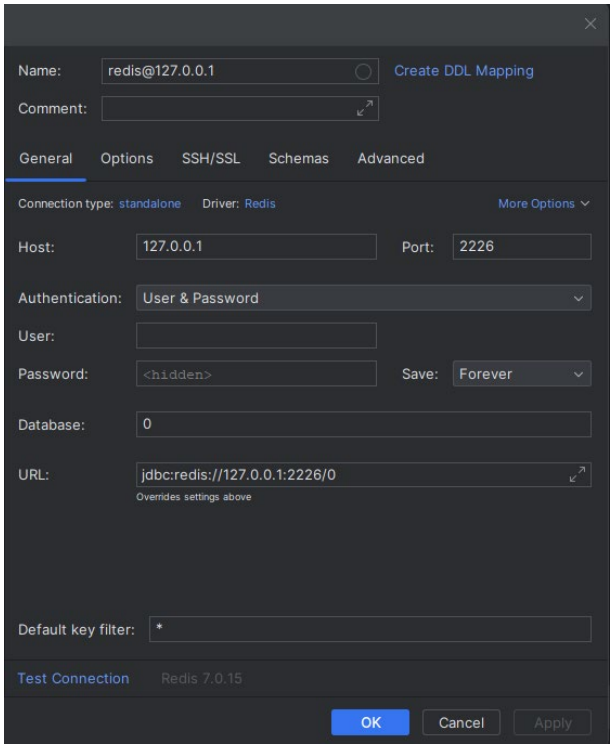
Da bi olajšali dostop do virtualnih naprav in omogočili učinkovito upravljanje baz podatkov, smo na obe napravi namestili module OpenSSH. To omogoča uporabo oddaljenega dostopa do terminala prek Visual Studio Code, kar znatno poenostavi delo z obema bazama.

Prav tako smo uporabili DataGrip, napredno orodje za vizualizacijo in upravljanje podatkovnih baz. Povezali smo se na baze PostgreSQL in Redis, kar nam omogoča jasen pregled nad strukturami tabel ter izvedbo poizvedb in optimizacij v grafičnem okolju.



Slika 6: Konfiguracija povezave PostgreSQL v Datagrip

Vir: (Lastni vir)



Slika 7: Konfiguracija povezave Redis v Datagrip

Vir: (Lastni vir)

Zaključek

Z uporabo dveh enakovredno konfiguriranih virtualnih okolij smo ustvarili nadzorovane pogoje za eksperimentalno primerjavo zmogljivosti podatkovnih baz PostgreSQL in Redis. Specifične različice podatkovnih baz in omrežna konfiguracija z uporabo port-forwardinga zagotavljajo optimalne pogoje za testiranje, medtem ko dostop SSH in DataGrip omogočata učinkovito upravljanje in vizualizacijo podatkov.

3.2 Zbiranje podatkov o nogometnih igralcih

Za analizo zmogljivosti podatkovnih baz PostgreSQL in Redis smo izbrali podatke o nogometnih igralcih iz petih najboljših evropskih nogometnih lig. Te podatke smo zbrali ročno s spletne platforme **Transfermarkt**, ki je znana po obsežnem naboru podatkov o klubih, igralcih in njihovih statističnih dosežkih. Proces zbiranja in priprave podatkov je bil ključnega pomena za analizo, saj podatki omogočajo realno primerjavo zmogljivosti obeh podatkovnih baz pri obdelavi kompleksnih poizvedb in analitičnih operacij.

3.2.1 Priprava datotek CSV

Podatki o nogometnih igralcih so bili razdeljeni v štiri ločene datoteke CSV, vsaka z različnimi atributi, potrebnimi za analizo. Vsaka datoteka je bila posebej prilagojena določeni vrsti podatkov, kar omogoča enostaven uvoz in obdelavo v obeh podatkovnih bazah.

Klubi (clubs.csv): vsebuje podatke o klubih in ključne atribute, kot so:

- club_id: enolični identifikator kluba,
- club_name: ime kluba,
- competition_id: identifikator tekmovanja,
- club_value: tržna vrednost kluba,
- national_team_players: število igralcev, ki igrajo za nacionalne ekipe,

- squad_size: velikost moštva,
- average_age: povprečna starost igralcev,
- foreigners: število tujih igralcev.

Tekmovanja (competitions.csv): vsebuje podatke o tekmovanjih, kot so:

- competition_id: enolični identifikator tekmovanja,
- competition_name: ime tekmovanja,
- number_of_teams: število ekip v tekmovanju,
- total_market_value: skupna tržna vrednost klubov,
- average_club_market_value: povprečna vrednost klubov,
- player_average_age: povprečna starost igralcev,
- foreigners: število tujih igralcev v tekmovanju.

Igralci (players.csv): vsebuje podatke o posameznih igralcih:

- player_id: enolični identifikator igralca,
- player_name: ime igralca,
- club_id: identifikator kluba, kjer igralec igra,
- age: starost igralca,
- citizenship: državljanstvo,
- position: igralni položaj,
- contract_expires: datum, ko igralcu poteče pogodba,
- current_market_value: trenutna tržna vrednost igralca.

Statistika igralcev (player_stats.csv): vsebuje podatke o dosežkih igralcev:

- stat_id: enolični identifikator statistike,
- player_id: identifikator igralca,
- goals: število doseženih golov,
- assists: število asistenc,
- appearances: število nastopov,
- goals_conceded: število prejetih golov (za vratarje),
- clean_sheets: število tekem brez prejetega zadetka,
- minutes_played: število odigranih minut.

Ti podatki so bili ključni za analizo zmogljivosti podatkovnih baz pri obdelavi poizvedb in učenju modelov umetne inteligence.

3.2.2 Priprava podatkov za uvoz

Za vsako podatkovno bazo (PostgreSQL in Redis) smo pripravili ločene skripte, ki omogočajo avtomatiziran uvoz podatkov iz datotek CSV. Glede na razliko v arhitekturi obeh baz je bil postopek prilagojen posamezni bazi.

PostgreSQL Uvoz

Za PostgreSQL smo uporabili skripto Python, ki uporablja knjižnici Pandas in SQLAlchemy za vzpostavitev povezave z bazo, ustvarjanje tabel in uvoz podatkov. PostgreSQL uporablja relacijski model, zato smo morali definirati tabele z ustreznimi relacijami med njimi. Skripta najprej ustvari potrebne tabele, nato pa iz vsake datoteke CSV prebere podatke in jih vstavi v ustrezno tabelo.

Ključni del skripte za uvoz podatkov v PostgreSQL

```
# Ustvari tabele in uvozi podatke v PostgreSQL

tables = ['competitions', 'clubs', 'players', 'player_stats']

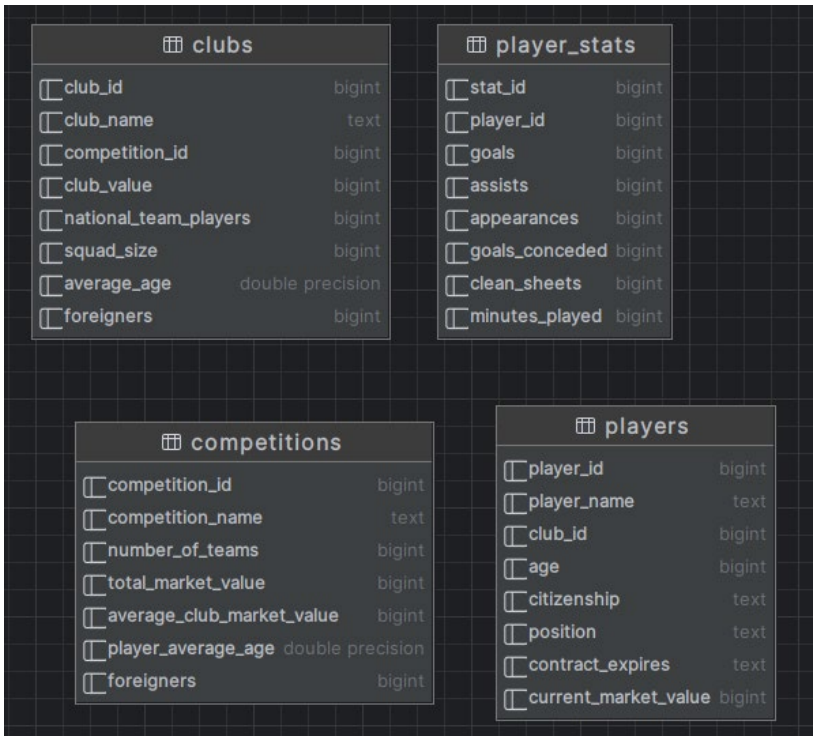

for table in tables:

    data = pd.read_csv(f'{table}.csv')  # Preberi CSV datoteke

    data.to_sql(table, engine, if_exists='append', index=False)


print("Podatki uspešno uvoženi v PostgreSQL.")
```

Tabele so definirane z ustreznimi primarnimi in tujimi ključi, kar omogoča referenčno celovitost podatkov. Ta pristop omogoča relacijske poizvedbe, kjer lahko podatki iz več tabel medsebojno komunicirajo.



Slika 8: Diagram baze PostgreSQL v Datagrip

Vir: (Lastni vir)

3.2.3 Redis Uvoz

Za Redis, ki deluje po modelu ključ-vrednost, smo morali uporabiti drugačen pristop. Namesto relacijskih tabel smo uporabili strukture Hash v Redis, kjer se vsaka vrstica iz datoteke CSV shrani kot hash. Uvoz smo izvedli s pomočjo knjižnice redis-py.

Ključni del skripte za uvoz podatkov v Redis

```
# Naloži podatke iz CSV v Redis
def load_csv_to_redis(table_name, key_column):
    with open(f'{table_name}.csv', mode='r') as file:
        reader = csv.DictReader(file)
        for row in reader:
            key = f"{table_name}:{row[key_column]}"  # Uporabi primarni ključ kot Redis ključ
            r.hset(key, mapping=row)
print("Podatki uspešno uvoženi v Redis.")
```

V tem primeru je ključ sestavljen iz imena tabele in vrednosti primarnega ključa, kar omogoča hitro iskanje po ključu. Ta struktura je optimizirana za hitre poizvedbe, kjer je treba dostopati do specifičnih podatkov brez potrebe po relacijah med več entitetami.

clubs:1 [redis@127.0.0.1]

8 rows | DDL

HMGET clubs:1

	field	value
1	club_value	1140000000
2	squad_size	25
3	club_name	Arsenal
4	foreigners	18
5	competition_id	1
6	national_team_players	18
7	average_age	25.3
8	club_id	1

Slika 9: Prikaz strukture ključev hash v bazi Redis

Vir: (Lastni vir)

3.2.4 Uvoz podatkov v PostgreSQL in Redis

Po pripravi podatkovnih skript smo vse štiri datoteke CSV uvozili v obe podatkovni bazi. V PostgreSQL so bili podatki razdeljeni v relacijske tabele, ki omogočajo kompleksne poizvedbe in analizo prek več tabel hkrati. V Redis pa so bili podatki shranjeni v obliki hash, kjer vsaka tabela predstavlja zbirko ključev, ki omogočajo hiter dostop do posameznih entitet.

Tako PostgreSQL kot Redis sta omogočila učinkovito shranjevanje in obdelavo podatkov, a vsaka na svoj način, kar bo omogočilo primerjavo zmogljivosti v nadaljnjih poglavjih.

3.3 Implementacija modelov umetne inteligence

Implementacija modelov umetne inteligence (AI) je ključni del raziskave, ki omogoča analizo zmogljivosti podatkovnih baz PostgreSQL in Redis pri učenju in napovedovanju tržnih vrednosti nogometnih igralcev. V tem podpoglavju bomo predstavili proces implementacije, izbiro modelov, pripravo podatkov in tehnike, ki smo jih uporabili za zagotavljanje uspešnega delovanja modelov v obeh bazah podatkov.

3.3.1 Priprava podatkov za modeliranje

Preden smo lahko implementirali modele umetne inteligence, smo morali pripraviti podatke iz obeh baz, PostgreSQL in Redis, tako da so bili združljivi z algoritmi za strojno učenje.

- **PostgreSQL:** Podatke iz PostgreSQL smo pridobili s pomočjo SQL poizvedbe, ki povezuje tabele players, clubs, competitions in player_stats. Uporabili smo knjižnico SQLAlchemy, ki omogoča preprosto povezavo med skripto Python in bazo PostgreSQL ter omogoča izvrševanje SQL poizvedb znotraj Pythona.
- **Redis:** Redis uporablja model ključ-vrednost, kar pomeni, da smo morali za pridobivanje podatkov iz več ključev uporabiti ustrezne funkcije, kot je hgetall. Ustvarili smo funkcijo, ki pridobi vse podatke o igralcih, klubih in tekmovanjih ter te podatke združi v enoten nabor, primeren za uporabo pri modelih umetne inteligence.

Po pridobitvi podatkov iz obeh baz smo za vsako podatkovno zbirko izvedli obsežno predprocesiranje, ki vključuje feature engineering, odstranjevanje odvečnih podatkov in kodiranje spremenljivk, ki so bile potrebne za nadaljnje modeliranje.

Feature Engineering in predprocesiranje

Feature engineering je postopek ustvarjanja novih značilnosti iz obstoječih podatkov, ki lahko izboljšajo zmogljivost modelov. Pri tej raziskavi smo uporabili naslednje metode feature engineeringa:

- **Goli in podaje na 90 minut:** Za igralce smo ustvarili značilnosti, ki prikazujejo njihovo učinkovitost v obliki golov in podaj na 90 minut igranja.
- **Pomembnost igralca:** Izračunali smo pomembnost igralca na podlagi odigranih minut glede na skupno število tekem. S tem smo ustvarili merilo, kako pomemben je igralec za svojo ekipo.
- **Leta do konca pogodbe:** Iz datuma poteka pogodbe igralca smo ustvarili značilnost, ki prikazuje, koliko let pogodbe igralcu še ostane, kar je ključen dejavnik pri napovedovanju njegove tržne vrednosti.
- **Polinomske značilnosti:** Starost igralca smo pretvorili v polinomske značilnosti tretje stopnje, kar omogoča modelu boljšo prilagodljivost pri zajemanju nelinearnih odnosov med starostjo in tržno vrednostjo.

Vse značilnosti, ki so nastale v tem procesu, so bile nato standardizirane z uporabo StandardScaler iz knjižnice scikit-learn, da smo zagotovili enako obravnavo vseh značilnosti znotraj modelov.

3.3.2 Izbor modelov umetne inteligence

Za analizo podatkov smo izbrali več različnih regresijskih modelov, ki temeljijo na različnih algoritmih. Namen uporabe več modelov je bil primerjati učinkovitost posameznih modelov pri učenju iz podatkov iz baz PostgreSQL in Redis. Izbrali smo modele, opisane v nadaljevanju.

Ridge Regression

Ridge regresija je linearni model, ki uporablja L2 regularizacijo za zmanjšanje možnosti prenaučenja. Gre za preprost model, ki je bil uporabljen kot osnovna linija za primerjavo z bolj kompleksnimi modeli. Ridge regresija je bila še posebej primerna za obdelavo podatkov z več značilnostmi, kjer obstaja tveganje za prenaučenje.

Ridge(alpha=1.0)

V tem primeru je alpha parameter regularizacije, ki smo ga prilagajali s pomočjo metode naključnega iskanja hiperparametrov (RandomizedSearchCV).

Odločitveno drevo (angl. Decision Tree)

Odločitveno drevo je model, ki razdeli podatke na podlagi značilnosti in jih uvršča v kategorije ali vrednosti. Prednost odločitvenega drevesa je v njegovi interpretabilnosti, saj omogoča vizualno predstavitev poti odločitev, ki jih sprejema model. Uporabljali smo ga za napovedovanje tržne vrednosti na podlagi več ključnih dejavnikov igralcev.

DecisionTreeRegressor(max_depth=10, min_samples_split=5)

Glavni parametri, kot sta max_depth in min_samples_split, so bili prilagojeni na podlagi RandomizedSearchCV, da bi poiskali optimalno strukturo drevesa.

Random Forest

Random Forest je ensemble metoda, ki združuje več odločitvenih dreves za izboljšanje natančnosti modela. Vsako drevo v gozdu napoveduje vrednosti neodvisno, končna napoved pa je povprečje napovedi vseh dreves. Random Forest je zelo robusten model, saj zmanjšuje varianco posameznih dreves in tako zagotavlja bolj stabilne napovedi.

RandomForestRegressor(n_estimators=100, max_depth=10, random_state=42)

Uporabili smo različna števila dreves (n_estimators) in globine dreves (max_depth) za doseganje optimalnih rezultatov.

Gradient Boosting

Gradient Boosting je napreden algoritem, ki gradi modele zaporedno, kjer vsak model poskuša izboljšati napake prejšnjega. Gradient Boosting je bil izbran zaradi svoje sposobnosti prilagajanja podatkom in visoke natančnosti pri napovedih.

GradientBoostingRegressor(n_estimators=200, learning_rate=0.1, max_depth=3, random_state=42)

Ključni parametri, ki smo jih prilagajali, so learning_rate (hitrost učenja) in n_estimators (število dreves), da smo dosegli čim boljšo prilagoditev modela podatkom.

3.3.3 Implementacija učenja modelov

Vsakega izmed naštetih modelov smo implementirali z uporabo Python knjižnice scikit-learn, ki ponuja celovito podporo za različne regresijske algoritme in orodja za vrednotenje modelov. Modeli so bili trenirani z uporabo metode fit, nato pa smo napovedi izvedli na testnem naboru podatkov z uporabo metode predict.

Proces učenja je vključeval naslednje korake:

- **Treniranje modela:** Vsak model smo trenirali na učni množici (angl. train dataset), kjer je model poskušal najti najboljše parametre za napovedovanje na podlagi značilnosti igralcev.
- **Napovedi na testni množici:** Po učenju modela smo napovedi izvajali na testni množici, kjer smo ocenili natančnost modelov.

- **Optimiranje hiperparametrov:** Za iskanje najboljših hiperparametrov za posamezne modele smo uporabili metodo RandomizedSearchCV, ki omogoča učinkovito iskanje po prostoru parametrov in izbiro najboljših nastavitev modela.

3.3.4 Učinkovitost in skaliranje

Za oba nabora podatkov, PostgreSQL in Redis, smo izvedli enake postopke učenja modelov. Pri tem smo primerjali čas učenja modelov na podatkih iz obeh podatkovnih baz, kar bo podrobneje obravnavano v naslednjem poglavju, kjer bomo primerjali metode vrednotenja modelov in zmogljivosti baz.

V tej fazi je bil poudarek na implementaciji modelov in zagotoviti, da so modeli enako optimizirani za podatke iz obeh baz, kar bo omogočilo natančno primerjavo v nadaljevanju analize.

3.4 Metode primerjave in ocenjevanja

V tem podpoglavju so opisane metode, ki smo jih uporabili za primerjavo zmogljivosti in ocenjevanje modelov umetne inteligence v kontekstu podatkovnih baz PostgreSQL in Redis. Cilj teh metod je oceniti, kako dobro podatki iz obeh baz podpirajo modele umetne inteligence, ter določiti učinkovitost, natančnost in časovno zahtevnost različnih modelov v obeh sistemih.

3.4.1 Razdelitev podatkov na učne in testne množice

Za začetek smo za oba nabora podatkov, pridobljena iz PostgreSQL in Redisa, uporabili metodo train-test split. Ta metoda omogoča razdelitev podatkov na dve skupini:

- učni podatki (train data): uporabljeni za treniranje modela;
- testni podatki (test data): uporabljeni za ocenjevanje učinkovitosti modela na nevidenih podatkih.

Vsak nabor smo razdelili v razmerju 80 : 20 (80 % učnih podatkov in 20 % testnih podatkov). S tem smo zagotovili, da se modeli učijo na večini podatkov, vendar jih ocenjujemo na ločenih testnih podatkih, kar pomaga preprečiti prenaučenje.

X_train_pg, X_test_pg, y_train_pg, y_test_pg = train_test_split(X_pg, y_pg, test_size=0.2, random_state=42)

X_train_redis, X_test_redis, y_train_redis, y_test_redis = train_test_split(X_redis, y_redis, test_size=0.2, random_state=42)

Ta pristop je enak za podatke iz obeh baz, kar omogoča neposredno primerjavo med PostgreSQL in Redisom glede na njihovo sposobnost, da podpira modele umetne inteligence.

3.4.2 Uporaba meril zmogljivosti

Za ocenjevanje uspešnosti modelov smo uporabili standardna merila zmogljivosti, ki so uveljavljena v strojno-učnih metodah regresije. Ta merila zagotavljajo kvantitativno oceno natančnosti modelov in kakovosti napovedi.

Srednja kvadratna napaka (angl. Mean Squared Error – MSE)

MSE je osnovno merilo za oceno, kako natančne so napovedi modela. Računa se kot povprečje kvadratov razlik med napovedanimi vrednostmi in dejanskimi vrednostmi. Nižja vrednost MSE pomeni, da so napovedi modela bolj natančne.

Formula za MSE:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y_i})^2$$

Kjer sta y_i dejanska vrednost in $\widehat{y_i}$ napovedana vrednost.

V Pythonu je MSE izračunan s funkcijo mean_squared_error iz knjižnice scikit-learn:

mse = mean_squared_error(y_test, y_pred)

Srednja absolutna napaka (angl. Mean Absolute Error – MAE)

MAE je podobno merilo kot MSE, vendar upošteva absolutne razlike med napovedmi in dejanskimi vrednostmi. MAE je bolj robustno merilo, saj ni tako občutljivo na izjemno napačne napovedi kot MSE.

Formula za MAE:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

V Pythonu je MAE izračunan s funkcijo mean_absolute_error:

mae = mean_absolute_error(y_test, y_pred)

Koeficient determinacije (R^2)

R^2 meri, kolikšen delež variance v podatkih lahko pojasni model. R^2 je razpon med 0 in 1, pri čemer je 1 popolno prileganje, 0 pa pomeni, da model ne pojasni nobene variance v podatkih.

Formula za R^2:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \bar{y})^2}{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Kjer je $\bar{y}$ povprečje dejanskih vrednosti.

V Pythonu je R^2 izračunan s funkcijo *r2_score*:

r2 = r2_score(y_test, y_pred)

3.4.3 Časovna analiza

Poleg natančnosti modelov smo analizirali tudi časovno učinkovitost posameznih postopkov učenja in napovedovanja.

Čas treniranja modela

Za vsak model smo merili čas, potreben za učenje modela z uporabo funkcije fit. To nam je omogočilo primerjavo, kateri model se hitreje nauči iz podatkov in kateri podatkovni sistem omogoča hitrejše izvajanje teh postopkov.

```
start_time = time.time()
model.fit(X_train, y_train)
training_time = time.time() - start_time
```

Čas napovedovanja

Po učenju modela smo merili tudi čas napovedovanja, tj. koliko časa potrebuje model, da izvede napovedi na testnih podatkih z uporabo funkcije predict. S tem smo analizirali, kako hitro model generira rezultate na podlagi novih podatkov.

```
start_time = time.time()
y_pred = model.predict(X_test)
prediction_time = time.time() - start_time
```

3.4.4 Optimizacija modelov in iskanje hiperparametrov

Za izboljšanje zmogljivosti modelov smo uporabili metodo naključnega iskanja hiperparametrov (RandomizedSearchCV). Ta tehnika omogoča samodejno iskanje optimalnih vrednosti hiperparametrov, kot je alpha pri Ridge regresiji ali max_depth pri odločitvenem drevesu.

```
random_search = RandomizedSearchCV(model, param_grid, n_iter=100, scoring='neg_mean_squared_error', random_state=42, n_jobs=-1)
random_search.fit(X_train, y_train)
```

Ta postopek je bil enako uporabljen za vse modele in je omogočil izboljšanje učinkovitosti modelov pri delu z različnimi podatkovnimi bazami.

3.4.5 Primerjava metod med PostgreSQL in Redisom

Za oba sistema, PostgreSQL in Redis, smo izvedli enake postopke treniranja in ocenjevanja modelov, da bi zagotovili pravično primerjavo zmogljivosti obeh podatkovnih baz pri delu z modeli umetne inteligence. Ključne primerjalne točke vključujejo:

- **časovno učinkovitost:** primerjava časa pridobivanja podatkov, treniranja modelov in izvajanja napovedi;
- **natančnost napovedi:** primerjava natančnosti napovedi na podlagi MSE, MAE in R^2;
- **optimizacijo modelov:** analiza, kako učinkovito so se modeli prilagajali na podatke iz obeh baz s pomočjo optimizacije hiperparametrov.

Te metode ocenjevanja in primerjave so bile uporabljene za analizo, ali Redis zaradi svoje strukture ključ-vrednost omogoča hitrejše izvajanje poizvedb in modeliranja v primerjavi s PostgreSQL, ter kako to vpliva na natančnost in stabilnost modelov.

S temi metodami primerjave in ocenjevanja smo postavili temelje za natančno in celovito analizo zmogljivosti obeh podatkovnih baz, kar bo podrobneje predstavljeno v naslednjih poglavjih, kjer bodo obravnavani rezultati in analiza dobljenih podatkov.

4 EKSPERIMENTALNA PRIMERJAVA PODATKOVNIH BAZ POSTGRESQL IN REDIS

To poglavje se osredotoča na natančno tehnično primerjavo dveh podatkovnih baz – **PostgreSQL** in **Redis** – v kontekstu različnih operacij, vključno s hitrostjo poizvedb, zmogljivostjo modelov umetne inteligence, stabilnostjo rezultatov in stroški. V vsakem podpoglavju se bomo poglobili v določene aspekte, ki so bili predmet testiranja, in podprli ugotovitve s konkretnimi rezultati iz skripte Python, ki je bila uporabljena za izvajanje teh testov.

4.1 Hitrost poizvedb

Hitrost poizvedb je ključni dejavnik pri izbiri podatkovne baze, zlasti kadar so odzivni časi bistvenega pomena. V tem delu primerjamo hitrost izvajanja osnovnih poizvedb med PostgreSQL in Redisom.

- **Operacija »lookup«:** Pri preprostem iskanju specifičnega igralca po ID-ju je Redis pokazal izjemno hitrost, saj je povprečni čas izvajanja poizvedbe znašal **0,000889 sekunde**. Ta čas je praktično ničeln, saj Redis deluje na principu shranjevanja podatkov v pomnilnik, kar zagotavlja skoraj takojšen dostop do podatkov. Na drugi strani je PostgreSQL za isto operacijo v povprečju porabil **0,004500 sekunde**, kar je prav tako hitro, vendar počasneje v primerjavi z Redisom. Takšne rezultate pričakujemo glede na arhitekturo obeh sistemov, kjer PostgreSQL deluje z diskom, Redis pa s pomnilnikom.

Tabela 2: Povprečni čas izvajanja »lookup« poizvedb v PostgreSQL in Redisu

	Test #1	**Test #2**	**Test #3**	**Test #4**	**Test #5**	**Povprečje**
PostgreSQL	0,003911 s	0,003904 s	0,004882 s	0,005895 s	0,003906 s	0,004500 s
Redis	0,000981 s	0,000000 s	0,000982 s	0,001504 s	0,000975 s	0,000889 s

Vir: (Lastni vir)

- **Filtrirne poizvedbe:** Pri bolj kompleksnih poizvedbah, kot je filtriranje igralcev po starosti (nad 25 let) in številu doseženih golov (več kot 10), se je PostgreSQL izkazal z zelo dobrim povprečnim časom izvedbe **0,0065 sekunde**, kar kaže na učinkovitost sistema pri procesiranju večjih količin podatkov in bolj kompleksnih poizvedb. Redis pa je za isto operacijo potreboval v povprečju **3,5361 sekunde**, kar je znatno več. Razlog za to razliko je v tem, da Redis ni optimiziran za kompleksne poizvedbe, kot je PostgreSQL.

Tabela 3: Povprečni čas izvajanja filtrirnih poizvedb v PostgreSQL in Redisu

	Test #1	**Test #2**	**Test #3**	**Test #4**	**Test #5**	**Povprečje**
PostgreSQL	0,0054 s	0,0078 s	0,0065 s	0,0064 s	0,0064 s	0,0065 s
Redis	3,6075 s	3,4183 s	3,6196 s	3,2969 s	3,7381 s	3,5361 s

Vir: (Lastni vir)

4.2 Zmogljivost modelov umetne inteligence

Pri analizi zmogljivosti modelov umetne inteligence smo uporabili več različnih algoritmov, med njimi Ridge regresijo, odločitveno drevo, naključni gozd in gradientno ojačenje. Za primerjavo zmogljivosti teh modelov smo merili čas treniranja modelov, MSE (srednja kvadratna napaka), MAE (srednja absolutna napaka) in R^2 (koeficient determinacije).

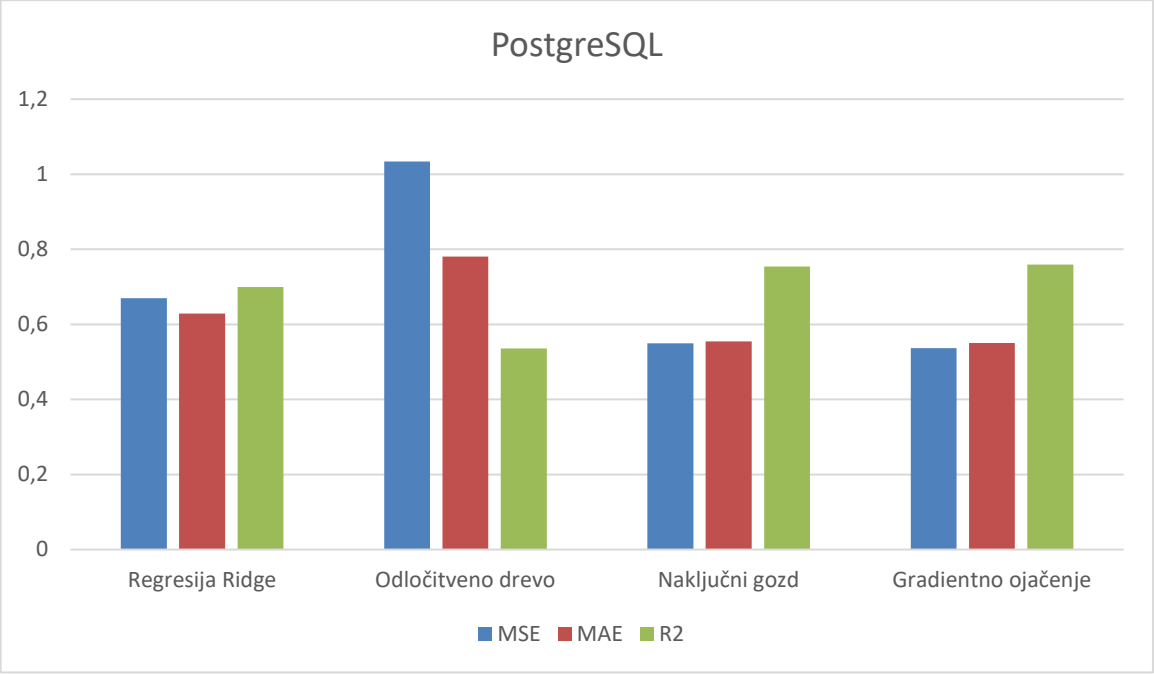
- **Regresija Ridge:** Trening tega modela je bil v obeh sistemih zelo hiter, pri čemer je PostgreSQL potreboval 0,0009 sekunde, Redis pa 0,0008 sekunde. V smislu natančnosti sta sistema pokazala različne rezultate. PostgreSQL je imel MSE 0,6699 in R^2 0,6998, medtem ko je Redis izboljšal rezultate na MSE 0,6423 in R^2 0,712. To kaže na nekoliko boljšo zmogljivost Redisa pri regresiji Ridge.
- **Odločitveno drevo**: Čas treniranja je bil za oba sistema sorazmerno kratek; PostgreSQL je potreboval 0,0091 sekunde, Redis pa 0,0089 sekunde. Vendar so rezultati natančnosti pokazali nekaj razlik. PostgreSQL je imel MSE 1,0336 in R^2 0,5361, Redis pa nekoliko boljše rezultate z MSE 1,0195 in R^2 0,5424. Redis je tako dosegel nekoliko boljše napovedi.

- **Naključni gozd:** Pri tem modelu so rezultati pokazali večjo natančnost pri obeh sistemih. PostgreSQL je dosegel MSE 0,549 in R^2 0,7539, medtem ko je Redis izboljšal natančnost na MSE 0,5286 in R^2 0,7627. To pomeni, da je Redis nekoliko bolj natančen pri tem modelu, vendar so bili rezultati med obema sistemoma zelo podobni.
- **Gradientno ojačenje**: Gradientno ojačenje je bilo najučinkovitejše pri obeh sistemih, pri čemer je PostgreSQL dosegel MSE 0,5368 in R^2 0,7591, Redis pa je še dodatno izboljšal rezultate na MSE 0,5154 in R^2 0,769. To kaže na odlično zmogljivost Redisa pri izvajanju tega modela.

Tabela 4: Povprečni čas treniranja modelov umetne inteligence v PostgreSQL in Redisu

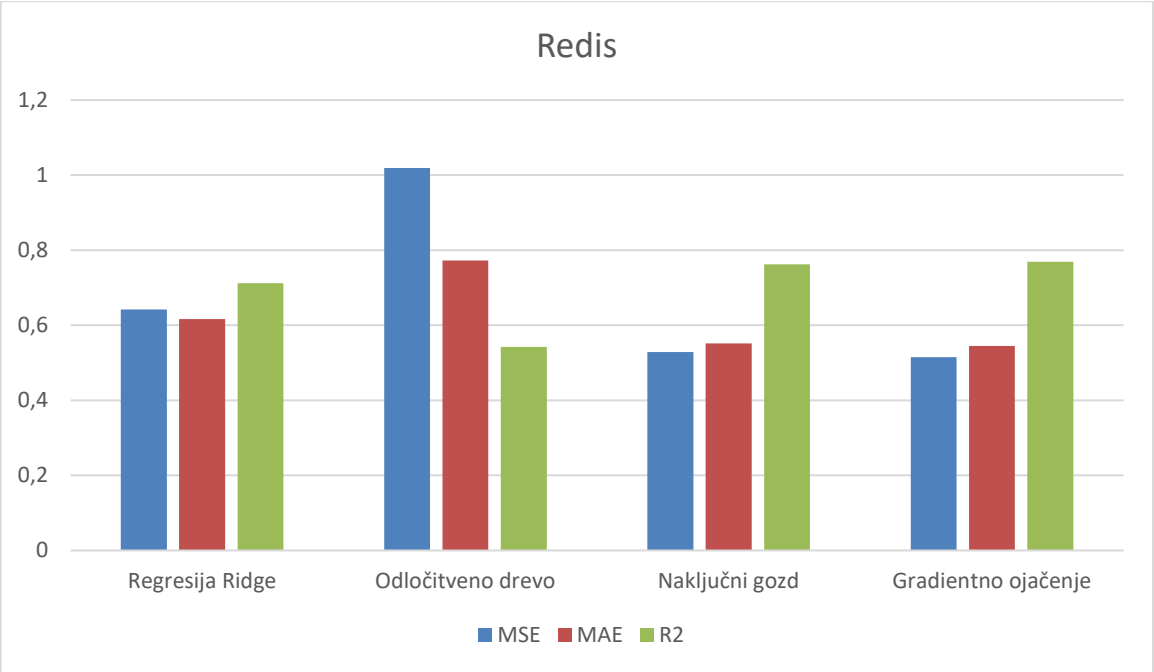
	Regresija Ridge	**Odločitveno drevo**	**Naključni gozd**	**Gradientno ojačenje**
PostgreSQL	0,0009 s	0,0091 s	0,6053 s	0,2250 s
Redis	0,0008 s	0,0089 s	0,5970 s	0,2192 s

Vir: (Lastni vir)



Graf 4: Zmogljivost modelov umetne inteligence v PostgreSQL

Vir: (Lastni vir)



Graf 5: Zmogljivost modelov umetne inteligence v Redisu

Vir: (Lastni vir)

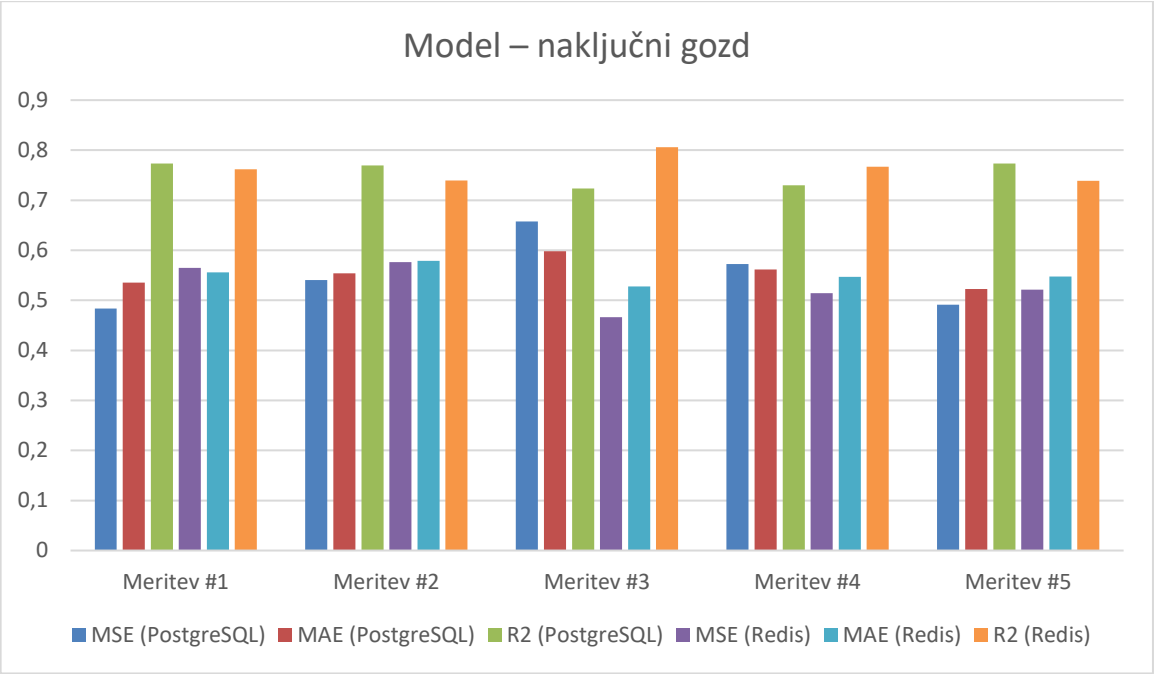
4.3 Stabilnost in zanesljivost rezultatov

Stabilnost rezultatov modelov umetne inteligence je ključna pri ocenjevanju zanesljivosti baz podatkov. Stabilnost smo preverjali z večkratnim izvajanjem modelov in ocenjevanjem njihove variabilnosti. Poleg tega smo analizirali, kako se modeli obnašajo ob prisotnosti odstopajočih podatkov (outlierjev).

- **Stabilnost pri večkratnem treniranju:** Na podlagi več ponovitev treniranja modelov se zdi, da Redis v določenih primerih dosega bolj stabilne in natančne rezultate kot PostgreSQL. To je še posebej očitno pri meritvah #3 in #4, kjer je Redis dosegel nižji MSE (0,4666 in 0,5143) in višji R^2 (0,8062 in 0,767) v primerjavi s PostgreSQL (MSE 0,6574 in 0,5725, R^2 0,7232 in 0,7301). Tudi pri drugih meritvah je Redis pogosto pokazal konkurenčne rezultate, kar kaže, da je pri določenih modelih stabilnejši.
- **Zanesljivost ob prisotnosti odstopajočih podatkov (outlierjev):** Redis je pokazal dobro odpornost na prisotnost odstopajočih podatkov. Na primer, pri meritvi #3, kjer je Redis dosegel bistveno boljši MSE in višji R^2 od PostgreSQL, to kaže na večjo zanesljivost Redisa pri obravnavi bolj zapletenih podatkovnih nizov. Kljub temu je PostgreSQL dosegel zelo

podobne rezultate pri nekaterih drugih meritvah, kot sta meritvi #1 in #5, kjer so bile vrednosti R^2 skoraj identične, kar kaže na konkurenčnost obeh sistemov v večini primerov.

- **Primerjava splošne stabilnosti:** Čeprav je PostgreSQL pokazal dobre rezultate pri nekaterih meritvah, so razlike med obema sistemoma majhne. Redis se je izkazal za nekoliko bolj stabilnega in natančnega pri večjem številu meritvenih primerov, zlasti ko gre za nižji MSE in višji R^2. To pomeni, da je Redis pri določenih modelih bolj učinkovit pri doseganju natančnih napovedi in stabilnih rezultatov.



Graf 6: Prikaz variance v rezultatih za naključni gozd

Vir: (Lastni vir)

4.4 Poraba virov in stroški uporabe

V tem poglavju analiziramo porabo strojnih virov, kot sta CPU in RAM, ter časovne stroške, ki jih povzročajo različne operacije v podatkovnih bazah PostgreSQL in Redis. Poleg tega podajamo pregled stroškov uporabe obeh tehnologij, vključno s stroški licenciranja in podpore. Poraba virov in časovna učinkovitost sta ključna dejavnika pri ocenjevanju stroškovne učinkovitosti teh dveh podatkovnih baz.

4.4.1 Poraba CPU-ja, RAM-a in časovna učinkovitost

Pridobivanje podatkov (angl. Data Retrieval)

Pri nalogi pridobivanja podatkov je PostgreSQL pokazal nižjo porabo CPU-ja in RAM-a v primerjavi z Redisom. V prvih petih meritvah se je poraba CPU-ja pri PostgreSQL gibala med 0 % in 2,3 %, medtem ko je bila poraba CPU-ja pri Redisu povprečno višja, z največjo vrednostjo 1,4 %. Kar se tiče časovne učinkovitosti, je PostgreSQL konstantno pokazal boljše rezultate, saj je pridobil podatke v povprečju dveh sekund, medtem ko je Redis za isto operacijo potreboval znatno več časa, od sedem do več kot 10 sekund. To kaže, da je PostgreSQL bolj učinkovit pri operacijah, ki vključujejo pridobivanje velikih količin kompleksnih podatkov.

Treniranje modelov (angl. Model Training)

Pri treniranju modelov, kot so Ridge Regression, Decision Tree, Random Forest in Gradient Boosting, so bile ugotovljene zanimive razlike. Na splošno je Redis pokazal nekoliko višjo porabo RAM-a, vendar so bile razlike minimalne. Pri porabi CPU-ja je pri enostavnejših modelih, kot je Ridge Regression, PostgreSQL izkazal z nekoliko večjo porabo CPU-ja (do 1,6 %), medtem ko je bila poraba pri Redisu nižja (v večini primerov med 0 % in 1,3 %). Časovna učinkovitost pri tem modelu med obema bazama ni pokazala večjih razlik, kar kaže na enako zmogljivost za enostavnejše modele umetne inteligence.

Pri bolj kompleksnih modelih, kot je Decision Tree, je PostgreSQL v nekaterih meritvah pokazal višjo porabo CPU-ja (do 5,3 %), medtem ko je Redis ohranjal bolj konstantno, nižjo porabo (do 1,8 %). Vendar pa je čas treniranja pri obeh bazah ostal primerljiv, kar kaže na to, da Redis za enako časovno učinkovitost porabi manj virov pri enostavnejših modelih.

Pri modelu Random Forest je bila poraba CPU-ja pri obeh bazah enakovredna, pri čemer je PostgreSQL včasih presegel Redis z višjo porabo (do 2,2 %), medtem ko je bil Redis bolj stalen z nižjo porabo virov. Časovna razlika med njima je bila minimalna, kar ponovno kaže, da sta oba sistema učinkovita pri obdelavi bolj zapletenih modelov.

Gradient Boosting model je pokazal zanimivo dinamiko – PostgreSQL je v večini primerov porabil manj virov, zlasti pri CPU-ju (pogosto le 0,1 %), medtem ko je Redis v nekaterih meritvah kazal večjo porabo CPU-ja (do 5,5 %). Kljub temu pa je bila časovna učinkovitost

obeh baz skoraj enaka, kar kaže na to, da Redis doseže primerljive rezultate, vendar pri večji porabi virov.

4.4.2 Stroški uporabe

PostgreSQL in Redis imata različne modele licenciranja in stroškov uporabe, kar je pomemben dejavnik pri izbiri tehnologije za specifične potrebe.

PostgreSQL: PostgreSQL je odprtokoden in brezplačen za osnovno uporabo. Stroški nastanejo predvsem pri uporabi upravljanih storitev in tehnične podpore. EnterpriseDB ponuja upravljano storitev PostgreSQL v oblaku, kjer se stroški gibajo od približno 62,5 EUR mesečno na vCPU (kar ustreza 0,0856 USD na vCPU-uro) za osnovno raven podpore. Za naprednejše storitve, kot je EDB Postgres Advanced Server, lahko stroški narastejo do 250 EUR mesečno na vCPU, kar vključuje visoko razpoložljivost, avtomatske varnostne kopije in napredno tehnično podporo (EDB, 2024).

Redis: Redis je prešel na licenco source-available, kar pomeni, da njegova izvorna koda ni več povsem odprtokodna, ampak zahteva licenco za komercialno uporabo. Redis Cloud ponuja brezplačno različico, Redis Cloud Essentials, ki podpira do 30 MB pomnilnika in 30 sočasnih povezav. Plačljive različice Redis Cloud Essentials se začnejo pri 5 EUR na mesec. Naprednejši paket, Redis Cloud Pro, stane približno 640 EUR mesečno, pri čemer vključuje neomejeno število baz, operacij in povezav ter dodatne funkcije, kot sta avtomatsko prilagajanje virov in visoka razpoložljivost (Redis, Pricing, 2024)

Tabela 5: Primerjalna tabela stroškov

	PostgreSQL	**Redis**
Stroški licence	0 (open-source)	0 (source-available)
Osnovni stroški podpore	~62,5 €/mesec	~5 €/mesec
Napredni stroški podpore	~250 €/mesec	~640 €/mesec

Vir: (Lastni vir)

Sklep

Na podlagi analiziranih podatkov se izkaže, da je PostgreSQL na splošno bolj učinkovit pri nalogah, ki vključujejo pridobivanje velikega števila kompleksnih podatkov, saj porabi manj CPU-ja in RAM-a kot Redis. Kar zadeva treniranje modelov, je Redis nekoliko bolj učinkovit pri enostavnejših modelih, kjer porabi manj CPU-ja in RAM-a. Pri bolj kompleksnih modelih pa Redis pogosto porabi več virov, čeprav se časovna učinkovitost med obema bazama ne razlikuje bistveno.

Glede stroškov je PostgreSQL bolj dostopna rešitev za podjetja, ki potrebujejo osnovno upravljanje in podporo, medtem ko so napredne funkcije Redisa, kot so neomejene baze, operacije in povezave, precej dražje.

5 REZULTATI IN ANALIZA

V tem poglavju bomo analizirali rezultate eksperimentalne primerjave med PostgreSQL in Redisom, ki smo jih obravnavali v prejšnjem poglavju. Namen tega poglavja je podati globljo interpretacijo teh rezultatov, ugotoviti, kako vplivajo na postavljene hipoteze, in raziskati, zakaj se podatkovni sistemi obnašajo na določen način v specifičnih scenarijih.

5.1 Analiza hitrosti poizvedb

V eksperimentalnem delu so bile izvedene meritve hitrosti poizvedb za oba sistema, in sicer tako za preproste operacije lookup kot tudi za bolj kompleksne filtrirne poizvedbe.

Redis proti PostgreSQL pri preprostih poizvedbah: Rezultati so pokazali, da je Redis izjemno hiter pri izvajanju preprostih poizvedb, kot so operacije lookup. To je v veliki meri posledica njegove arhitekture, ki temelji na pomnilniškem hranjenju podatkov, kar omogoča skoraj takojšen dostop do ključev in povezanih vrednosti. Ta zasnova je optimalna za scenarije, kjer je hitrost dostopa do podatkov ključnega pomena. PostgreSQL, ki temelji na relacijskem modelu in diskovnem shranjevanju podatkov, je bil pričakovano nekoliko počasnejši, čeprav še vedno izjemno hiter glede na naravo operacije. V realnih aplikacijah, kjer gre za pogoste preproste poizvedbe, na primer pri predpomnjenju podatkov ali shranjevanju seje, bi Redis predstavljal pomembno prednost.

Kompleksnejše poizvedbe in filtriranje: Pri bolj kompleksnih poizvedbah je PostgreSQL dosegel boljše rezultate. Filtriranje igralcev na podlagi več atributov (npr. starost in doseženi goli) je bilo izvedeno precej hitreje kot pri Redisu, ki pri takšnih operacijah ni optimiziran. Relacijska struktura PostgreSQL omogoča optimizirane kompleksne poizvedbe z uporabo SQL sintakse, kjer lahko sistem učinkovito obravnava in filtrira velike količine podatkov. Redis, na drugi strani, za to potrebuje več časa, saj ni zasnovan za obvladovanje kompleksnih poizvedb z več pogoji, kar je potrdil tudi več kot štirikrat daljši čas izvedbe filtrirnih operacij.

Vpliv na realne aplikacije: Razlike v hitrosti poizvedb med PostgreSQL in Redisom kažejo na pomembne izbire pri arhitekturi sistemov, odvisno od vrste aplikacije. Redis bi bil bolj primeren za aplikacije, kjer so potrebni hitri dostopi do vnaprej znanih podatkov, kot so predpomnilniki ali hitri brskalniški zahtevki. PostgreSQL pa bo imel prednost pri bolj kompleksnih operacijah, kjer sta potrebna učinkovito filtriranje in združevanje podatkov, na primer v aplikacijah, kjer se obdelujejo veliki podatkovni nabori z več pogoji.

5.2 Analiza zmogljivosti modelov umetne inteligence

Pri primerjavi zmogljivosti modelov umetne inteligence na obeh sistemih smo analizirali čas treniranja, natančnost (MSE, MAE) in koeficient determinacije (R^2). Štirje modeli –regresija Ridge, odločitveno drevo, naključni gozd in gradientno ojačenje – so bili trenirani tako na PostgreSQL kot na Redisu, pri čemer so se pojavile določene pomembne razlike.

Regresija Ridge: Pri regresiji Ridge sta oba sistema, PostgreSQL in Redis, pokazala primerljive rezultate, čeprav je Redis dosegel nekoliko boljši MSE (0,6423 v primerjavi s PostgreSQL 0,6699) in nekoliko višji koeficient determinacije R^2 (0,712 v primerjavi z 0,6998). To nakazuje, da Redis nekoliko bolje optimizira napovedi pri tem modelu, vendar so razlike majhne. V praksi to pomeni, da sta oba sistema sposobna zagotoviti solidne rezultate pri preprostih modelih, kot je regresija Ridge. V aplikacijah, kjer je hitrost ključnega pomena, bi Redis s hitrejšim časom treniranja lahko imel prednost, vendar razlika ni tako velika, da bi to pomembno vplivalo na končne odločitve o sistemu.

Odločitveno drevo: Rezultati odločitvenega drevesa so pokazali podobno sliko v obeh sistemih, pri čemer je Redis imel nekoliko boljši MSE (1,0195) in R^2 (0,5424) v primerjavi s PostgreSQL (MSE 1,0336, R^2 0,5361). Čeprav sta oba sistema dosegla sorazmerno podobno natančnost, to kaže, da je Redis morda nekoliko bolj učinkovit pri tem modelu. Odločitveno drevo je zaradi svoje preproste narave manj odvisno od kompleksnosti sistema in se bolj naslanja na značilnosti podatkov, zato se rezultati med obema sistemoma bistveno ne razlikujejo. V realnih aplikacijah bi oba sistema lahko uspešno uporabljali za hitre in enostavne napovedi, saj so razlike v zmogljivosti minimalne.

Naključni gozd: Pri naključnem gozdu se je pokazala zanimiva dinamika. PostgreSQL je dosegel MSE 0,549 in R^2 0,7539, medtem ko je Redis izboljšal MSE na 0,5286 in R^2 na 0,7627. To kaže, da je Redis nekoliko boljši pri natančnem prilagajanju napovedi za ta model. Redisova višja natančnost in hiter čas treniranja bi lahko bila prednost v aplikacijah, kjer sta pomembni tako hitrost kot natančnost. PostgreSQL pa kljub temu kaže stabilno zmogljivost, kar pomeni, da je prav tako zanesljiv za izvajanje kompleksnih modelov, kot je naključni gozd, vendar bi Redis morda nudil boljše rezultate v krajši časovni dinamiki.

Gradientno ojačenje: Gradientno ojačenje je pokazalo najboljše rezultate med vsemi modeli. PostgreSQL je dosegel MSE 0,5368 in R^2 0,7591, Redis pa je ponovno nekoliko izboljšal rezultate z MSE 0,5154 in R^2 0,769. To kaže, da Redis bolje obvlada kompleksnejše modele z večjo natančnostjo in manjšo napako. Redis bi bil v tem primeru primernejši za aplikacije, kjer je potrebna visoka natančnost napovedi z minimalnim MSE in hitrim časom treniranja, kot na primer pri finančnih napovedih ali optimizaciji poslovnih procesov.

Vpliv na realne aplikacije: Rezultati analize potrjujejo, da oba sistema, PostgreSQL in Redis, nudita dobre zmogljivosti za različne modele umetne inteligence. Redis izstopa z nekoliko višjo natančnostjo in hitrejšim časom treniranja, kar ga naredi primernega za aplikacije, kjer sta hitrost in natančnost ključnega pomena. PostgreSQL pa izkazuje svojo vrednost s stabilnostjo in zanesljivostjo, zaradi česar je primeren za scenarije, kjer so napovedi manj občutljive na hitro treniranje, vendar je ključna dolgoročna stabilnost rezultatov. Na podlagi teh rezultatov bi bilo pametno izbrati sistem glede na specifične zahteve aplikacije in naravo uporabljenih modelov.

5.3 Primerjava stabilnosti rezultatov

Stabilnost modelov umetne inteligence je ključnega pomena za dolgoročno uporabo sistema, še posebej pri ponavljajočem se treniranju na različnih podatkovnih naborih. Analizirali smo stabilnost rezultatov pri večkratnih ponovitvah treniranj na PostgreSQL in Redis sistemih ter proučili, kako stabilni so rezultati glede na spremembe v podatkih in morebitne odstopajoče vrednosti.

Stabilnost v PostgreSQL: PostgreSQL je v večini primerov pokazal zelo stabilne rezultate, zlasti pri enostavnejših modelih in modelih z manjšo kompleksnostjo. Na primer, pri meritvah #1 in #5 so bile vrednosti MSE (0,4836 in 0,4913) in R^2 (0,7732 v obeh primerih) zelo podobne med ponovitvami, kar kaže na visoko stopnjo stabilnosti tega sistema. Čeprav je pri bolj kompleksnih modelih, kot je naključni gozd, PostgreSQL dosegel nekoliko višje vrednosti MSE v primerjavi z Redisom (0,6574 proti 0,4666 pri meritvi #3), je še vedno pokazal solidno konsistentnost rezultatov. Stabilnost PostgreSQL se tako izkaže kot prednost pri scenarijih, kjer je potrebno večkratno treniranje na podobnih podatkovnih naborih in kjer je ponovljivost ključnega pomena.

Stabilnost v Redisu: Redis je prav tako pokazal zavidljivo stabilnost, vendar je bil v določenih primerih nekoliko bolj občutljiv na spremembe v podatkih. To je postalo najbolj očitno pri bolj kompleksnih modelih, kot je gradientno ojačenje, kjer so se vrednosti MSE in R^2 med ponovitvami treniranj nekoliko bolj razlikovale. Na primer, pri meritvi #3 je Redis dosegel zelo nizko vrednost MSE (0,4666) in visoko vrednost R2 (0,8062), kar kaže na večjo natančnost v primerjavi s PostgreSQL, vendar so bili razponi med ponovitvami širši, kar lahko nakazuje rahlo večjo variabilnost rezultatov. Kljub temu Redis ni pokazal bistvenih odstopanj, kar pomeni, da je še vedno primeren za aplikacije, kjer popolna konsistentnost rezultatov ni ključna, a je visoka natančnost še vedno zahtevana.

Primerjava stabilnosti med sistemi: Čeprav je PostgreSQL pokazal nekoliko bolj stabilne rezultate, zlasti pri enostavnejših modelih in manjših naborih podatkov, se Redis v določenih primerih izkaže z večjo natančnostjo, zlasti pri kompleksnejših modelih, kot sta naključni gozd in gradientno ojačenje. Na primer, pri meritvah #3 in #4 je Redis dosegel nižje MSE in višji R^2 kot PostgreSQL, kar kaže na njegov potencial za natančnejše napovedi, vendar z rahlo večjo variabilnostjo pri večkratnem treniranju.

Vpliv na realne aplikacije: Rezultati analize stabilnosti kažejo, da bi PostgreSQL lahko bil boljša izbira za aplikacije, kjer sta ključnega pomena ponovljivost rezultatov in zanesljivost napovedi. Na primer, pri dolgoročnih projektih, kjer so potrebni stabilni rezultati čez več ponovitev treniranj na istih podatkovnih naborih, bo PostgreSQL zagotovil višjo stopnjo

stabilnosti. To je še posebej pomembno pri modelih, kjer se pričakuje konstantna kakovost rezultatov, kot so napovedi prihodkov ali dolgoročne analize tržnih trendov.

Po drugi strani pa se Redis izkaže kot odlična izbira za aplikacije, kjer sta hitrost in natančnost treniranja pomembnejši kot stabilnost rezultatov med več ponovitvami. Redis je pri meritvah, kot sta meritvi #3 in #4, pokazal boljše rezultate z nižjim MSE in višjim R^2, kar pomeni, da bi bil primernejši za aplikacije, kjer so podatkovni nabori bolj kompleksni in kjer so pomembne natančne napovedi v krajšem časovnem okviru, tudi če se stabilnost med večkratnim treniranjem nekoliko zmanjša.

6 RAZPRAVA

V poglavju razprave bomo povezali rezultate eksperimentov z raziskovalnimi vprašanji in hipotezami, ki so bile postavljene v začetku tega dela. Osredotočili se bomo na širši pomen ugotovitev ter ovrednotili omejitve raziskave in možnosti za nadaljnje delo na tem področju. Namen razprave je poglobljeno razmišljanje o tem, kako pridobljeni rezultati prispevajo k razumevanju zmogljivosti PostgreSQL in Redisa pri uporabi umetne inteligence ter kaj to pomeni za prihodnjo uporabo teh podatkovnih baz v resničnih aplikacijah.

6.1 Pomen rezultatov

V sklopu primerjave podpore in skupnosti je bilo ugotovljeno, da PostgreSQL uživa širšo podporo in bolj raznoliko uporabniško bazo, kar potrjuje hipotezo 1, ki trdi, da ima PostgreSQL večjo podporo in širšo skupnost uporabnikov v primerjavi z Redisom. Analiza podatkov z različnih virov, kot so Stack Overflow, GitHub in DB-Engines Ranking, je pokazala, da je PostgreSQL močno zastopan v širši industriji, z več kot 178.000 vprašanji na Stack Overflow in višjim rangom na lestvici DB-Engines, kjer je na četrtem mestu. Kljub temu da Redis vodi na GitHub z večjim številom zvezdic in forkov, je PostgreSQL bolj uveljavljen in iskan na trgu dela, kar potrjujejo podatki z LinkedIna, kjer je za PostgreSQL objavljenih znatno več delovnih mest v primerjavi z Redisom, kar nakazuje na večjo industrijsko prisotnost in širšo podporo.

Pridobljeni rezultati eksperimentov so prinesli pomembne vpoglede v obnašanje in zmogljivosti obeh podatkovnih baz, PostgreSQL in Redisa, ter njihovo uporabnost v različnih kontekstih. Prvi del rezultatov, ki se nanaša na hitrost poizvedb, je pokazal, da je PostgreSQL hitrejši pri kompleksnejših poizvedbah, kar ovrže hipotezo 2, ki trdi, da so poizvedbe za filtriranje podatkov hitrejše v Redisu. Dejstvo, da je PostgreSQL bistveno hitrejši pri filtrirnih poizvedbah, še posebej pri obdelavi več atributov in zapletenih poizvedbah, kaže, da je bolj primeren za aplikacije, kjer je potreben učinkovit proces filtriranja večjih količin podatkov. Redis je v tem pogledu manj primeren za obvladovanje kompleksnih poizvedb, kar je bilo v eksperimentih potrjeno z daljšimi časi izvajanja.

Pri analizi zmogljivosti modelov umetne inteligence so rezultati potrdili hipotezo 3, da je učenje modelov umetne inteligence rahlo hitrejše v Redisu, čeprav je bila razlika med obema sistemoma zanemarljiva. Redisova rahla prednost pri hitrosti treniranja modelov, kot je regresija Ridge, je verjetno posledica hitrega dostopa do podatkov, vendar razlika ni dovolj velika, da bi imela bistven vpliv na izbiro podatkovne baze. Obe bazi, Redis in PostgreSQL, sta bili skoraj enakovredni v času treniranja modelov, kar kaže na njihovo primerljivo učinkovitost pri preprostejših modelih umetne inteligence.

Pri bolj zapletenih modelih, kot sta naključni gozd in gradientno ojačenje, se hipoteza 4, da PostgreSQL zagotavlja stabilnejše in bolj zanesljive rezultate, ni potrdila. V eksperimentih je Redis pokazal povsem enakovredne rezultate v primerjavi s PostgreSQL, kar pomeni, da sta oba sistema enako primerna za učenje kompleksnih modelov. Stabilnost rezultatov je bila v obeh sistemih dosledna, kar pomeni, da je Redis enako zanesljiv kot PostgreSQL pri ponovitvah eksperimentov. To kaže, da je izbira med obema podatkovnima bazama pri teh modelih lahko odvisna od drugih dejavnikov, kot so infrastrukturne zahteve ali specifike aplikacije, saj sta glede stabilnosti in zanesljivosti izenačena.

6.2 Omejitve raziskave

Kljub pridobljenim vpogledom je raziskava imela nekaj omejitev, ki bi lahko vplivale na splošnost ugotovitev. Prva omejitev je povezana s podatkovnim naborom, ki smo ga uporabili za eksperimente. Čeprav so bili podatki o nogometnih igralcih zbrani iz večjih lig in so vključili različne atribute, je velikost podatkovnega nabora morda premajhna, da bi v celoti izkoristili zmogljivosti obeh podatkovnih baz. Pri uporabi večjih podatkovnih množic, ki vključujejo milijone zapisov, bi se lahko obnašanje PostgreSQL in Redisa še bolj izrazilo, predvsem pri hitrostnih meritvah in obvladovanju velikih obsegov podatkov.

Druga omejitev je strojna oprema, na kateri so bili izvedeni testi. Eksperimenti so bili izvedeni v virtualnem okolju z omejenimi strojni viri (RAM in CPU), kar bi lahko vplivalo na rezultate, še posebej pri meritvah porabe virov in časa izvajanja operacij. V resničnem okolju, kjer bi bile na voljo večje procesorske zmogljivosti in spomin, bi morda Redis dosegel še večjo hitrost pri poizvedbah, PostgreSQL pa bi lahko bolje obvladoval obsežnejše operacije.

Poleg tega raziskava ni zajela vseh možnih primerov uporabe obeh podatkovnih baz. Fokus je bil na primerjavi hitrosti poizvedb in zmogljivosti umetne inteligence, vendar niso bile testirane druge pomembne funkcionalnosti, kot so varnost, skalabilnost in zmožnost razširljivosti obeh sistemov v večuporabniških okoljih. Te lastnosti bi lahko vplivale na izbiro ene ali druge baze v specifičnih primerih uporabe.

6.3 Možnosti za nadaljnje raziskave

Raziskava je odprla več poti za nadaljnje raziskave, ki bi lahko razširile in poglobile razumevanje zmogljivosti PostgreSQL in Redisa v različnih okoljih. Prva možnost za nadaljnje raziskave bi bila izvedba eksperimentov na večjih podatkovnih naborih. Vključno z milijoni zapisov bi lahko zagotovili vpogled v to, kako se oba sistema obnašata pri obsežnejših operacijah in kakšne so omejitve njune skalabilnosti. Poleg tega bi takšni eksperimenti pokazali, ali se rezultati, pridobljeni na manjših podatkovnih naborih, prenesejo na večje podatkovne baze.

Druga možnost bi bila uporaba različnih modelov umetne inteligence. Čeprav so bili v raziskavi uporabljeni štirje modeli (regresija Ridge, odločitveno drevo, naključni gozd in gradientno ojačenje), bi bilo zanimivo proučiti, kako se PostgreSQL in Redis obneseta pri drugih modelih, kot so nevronske mreže ali globoko učenje. Ti modeli imajo pogosto večje zahteve po procesorski moči in spominu, zato bi se zmogljivosti obeh sistemov lahko bolj izrazile.

Še ena pomembna smer za nadaljnje raziskave bi bila analiza porabe virov in učinkovitosti v različnih okoljih. Raziskava je pokazala, da se oba sistema obnašata drugače glede na porabo CPU in RAM. Z nadaljnjimi raziskavami bi lahko natančneje analizirali vpliv različnih vrst strojne opreme in okolij (npr. oblačna infrastruktura) na učinkovitost obeh sistemov. Prav tako bi bilo smiselno proučiti, kako se oba sistema obneseta v večuporabniških okoljih, kjer so pomembni varnost, zaklepanje transakcij in dostop do podatkov v realnem času.

Na koncu bi bilo mogoče raziskati tudi različne optimizacijske tehnike, ki so specifične za PostgreSQL in Redis, da bi izboljšali njuno zmogljivost. Na primer, optimizacija indeksiranja v PostgreSQL bi lahko še povečala učinkovitost filtriranja, medtem ko bi optimizacija shranjevanja ključev in vrednosti v Redis prinesla izboljšave pri zahtevnejših poizvedbah.

7 SKLEP

Zaključek tega diplomskega dela povzema ključne ugotovitve raziskave, ki je obravnavala primerjavo zmogljivosti dveh priljubljenih podatkovnih baz, PostgreSQL in Redisa, v kontekstu uporabe z modeli umetne inteligence in obdelave podatkov. Cilj raziskave je bil oceniti, katera podatkovna baza je bolj primerna za specifične naloge, kot so filtriranje podatkov, učenje modelov umetne inteligence ter stabilnost in zanesljivost rezultatov. Raziskava je vključevala natančno analizo zmogljivosti obeh sistemov v različnih scenarijih, vključno z njihovo industrijsko priljubljenostjo, hitrostjo poizvedb, učinkovitostjo pri učenju modelov in stabilnostjo rezultatov. Ključni cilj raziskave je bil proučiti, kako se PostgreSQL in Redis razlikujeta v teh pomembnih vidikih in kako ti rezultati vplivajo na specifične aplikacije, ki uporabljajo umetno inteligenco.

Eden izmed glavnih ciljev tega dela je bil preveriti hipotezo 1, ki je trdila, da ima PostgreSQL širšo podporo in večjo skupnost uporabnikov v primerjavi z Redisom. Ta hipoteza je bila potrjena na več ravneh. Analiza na platformah, kot sta Stack Overflow in LinkedIn, je pokazala, da ima PostgreSQL večjo skupnost razvijalcev in je bolj priljubljen v širši industriji. PostgreSQL ima več kot 178.000 vprašanj na Stack Overflow, kar je znak večjega števila uporabnikov, ki uporabljajo ta sistem za zapletene naloge. Podatki z LinkedIna so prav tako potrdili večjo prisotnost PostgreSQL na trgu dela, saj je bilo razpisanih bistveno več delovnih mest za strokovnjake s poznavanjem PostgreSQL v primerjavi z Redisom. Na drugi strani je Redis bolj priljubljen na odprtokodnih platformah, kot je GitHub, kjer ima več zvezdic in forkov. Vendar to ne spremeni dejstva, da PostgreSQL ostaja bolj zastopan v širšem industrijskem okolju, kjer je podpora skupnosti in komercialnih podjetij ključnega pomena.

Naslednji pomemben sklop rezultatov raziskave je povezan z zmogljivostjo poizvedb. Hipoteza 2, ki je trdila, da so poizvedbe za filtriranje podatkov hitrejše v Redisu kot v PostgreSQL, se ni potrdila. Dejanski rezultati so pokazali, da je PostgreSQL znatno hitrejši pri kompleksnih filtrirnih poizvedbah, ki vključujejo več atributov in pogojev. Redis je pokazal svojo prednost pri preprostih poizvedbah tipa lookup, kjer je njegova pomnilniška arhitektura omogočila skoraj takojšen dostop do podatkov. Vendar pa se je pri kompleksnejših operacijah, kot je filtriranje na podlagi več parametrov, izkazalo, da PostgreSQL zaradi svoje optimizirane SQL arhitekture doseže bistveno boljše rezultate. To pomeni, da je PostgreSQL bolj primeren za aplikacije, ki

zahtevajo napredne poizvedbe in združevanje podatkov, medtem ko je Redis primernejši za aplikacije, kjer so poizvedbe preproste in je ključnega pomena hitrost dostopa do podatkov.

Tretja ključna točka raziskave je bila povezana z zmogljivostjo pri učenju modelov umetne inteligence. Hipoteza 3, ki je trdila, da je učenje modelov hitrejše pri Redisu kot pri PostgreSQL, se je večinoma potrdila, čeprav je bila razlika majhna. Redis je nekoliko hitrejši pri enostavnejših modelih, kot je regresija Ridge, kar je posledica njegovega hitrega dostopa do podatkov iz pomnilnika. Ta prednost je še posebej očitna pri manjših in manj kompleksnih modelih, kjer hitrost dostopa do podatkov igra pomembno vlogo. Kljub temu razlika med PostgreSQL in Redisom ni bila dovolj velika, da bi v praksi pomembno vplivala na odločitev o izbiri sistema. Obe bazi sta pokazali primerljive rezultate pri enostavnejših modelih umetne inteligence, kar pomeni, da sta obe primerni za uporabo v teh kontekstih.

Pri bolj zapletenih modelih umetne inteligence, kot sta naključni gozd in gradientno ojačenje, hipoteza 4, ki je trdila, da PostgreSQL zagotavlja stabilnejše in zanesljivejše rezultate, ni bila potrjena. Rezultati so pokazali, da sta PostgreSQL in Redis v teh primerih enakovredna glede stabilnosti rezultatov in zanesljivosti napovedi. Redis se je izkazal za stabilnega in zanesljivega pri učenju teh kompleksnih modelov, prav tako kot PostgreSQL. To pomeni, da je izbira med obema sistemoma v kontekstu teh modelov odvisna od drugih dejavnikov, kot so infrastrukturne zahteve ali posebnosti aplikacije, saj sta oba sistema enako sposobna pri obdelavi zapletenih modelov umetne inteligence.

Na podlagi teh ugotovitev je mogoče podati ključne predloge za uporabo PostgreSQL in Redisa v praksi. PostgreSQL je priporočljiv za aplikacije, ki zahtevajo kompleksno obdelavo podatkov, kjer so napredne poizvedbe in obsežne analize podatkov ključnega pomena. Takšne aplikacije vključujejo analitične sisteme, obsežna podatkovna skladišča in znanstvene raziskave, kjer sta stabilnost rezultatov in zanesljivost napovedi ključni za sprejemanje odločitev. PostgreSQL se je izkazal za zmogljivejšega pri zapletenih poizvedbah in bolj stabilnega pri večkratnih ponovitvah eksperimentov, kar ga postavlja v prednost v primerih, kjer je ponovljivost rezultatov ključnega pomena.

Redis pa je idealen za aplikacije, kjer sta ključni takojšnja odzivnost in hitrost dostopa do podatkov. To vključuje realnočasovne sisteme, hitre iskalne aplikacije, predpomnilnike in aplikacije, kjer je dostop do podatkov zelo hiter, vendar poizvedbe niso kompleksne. Redis je pokazal izjemno hitrost pri preprostih poizvedbah in nekoliko boljšo zmogljivost pri enostavnih modelih umetne inteligence, kar ga naredi primernega za aplikacije, kjer je hitro izvajanje poizvedb in treniranje modelov ključnega pomena.

Na koncu lahko ugotovimo, da sta oba sistema močna ter imata svoje prednosti in slabosti. Izbira med PostgreSQL in Redisom je odvisna od specifičnih zahtev aplikacije in konteksta uporabe. PostgreSQL je bolj primeren za aplikacije, ki zahtevajo kompleksno obdelavo podatkov, stabilnost in zanesljivost, medtem ko je Redis optimalna izbira za aplikacije, kjer je ključna hitrost dostopa do podatkov in enostavne operacije. Raziskava je pokazala, da noben sistem ni univerzalno najboljši, temveč je treba izbiro podatkovne baze prilagoditi glede na specifične zahteve in potrebe aplikacije.

8 VIRI, LITERATURA

Ameisen, E. (2020). Building Machine Learning Powered Applications: Going from Idea to Product. O'Reilly Media.

Briney, K. (2015). Data Management for Researchers: Organize, Maintain and Share Your Data for Research Success. Pelagic Publishing.

Burkov, A. (2020). Machine Learning Engineering. True Positive Inc.

Carlson, J. L. (2013). Redis in Action. Manning.

Chauhan, C., & Kumar, D. (2017). PostgreSQL High Performance Cookbook. Packt Publishing.

Da Silva, M. D., & Tavares, H. L. (2015). Redis Essentials. Packt Pub Ltd.

Das, V. (2015). Learning Redis. Packt Pub Ltd.

DB-Engines. (2024). Plans Comparison. Pridobljeno iz DB-Engines: https://db-engines.com/en/ranking

Drake, J. D., & Worsley, J. C. (2002). Practical PostgreSQL. O'Reilly Media.

EDB. (2024). Plans Comparison. Pridobljeno iz EDB: https://www.enterprisedb.com /products/plans-comparison#edbbiganimalcommunity360plan

Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. O'Reilly Media.

Github. (2024). Pridobljeno iz Github: https://github.com

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning (Adaptive Computation and Machine Learning series). The MIT Press.

Grafana. (2023). Docs. Pridobljeno iz Grafana Labs: https://grafana.com

Juba, S., & Volkov, A. (2019). Learning PostgreSQL 11 - Third Edition: A beginner's guide to building high-performance PostgreSQL database solutions (3rd ed.). Packt Publishing.

Kimball, R., & Ross, M. (2013). The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling (3rd ed.). Wiley.

LinkedIn. (2024). Pridobljeno iz LinkedIn: https://www.linkedin.com

Liu, Y. (2017). Python Machine Learning By Example: The easiest way to get into machine learning. Packt Publishing.

Macedo, T., & Oliveira, F. (2011). Redis Cookbook: Practical Techniques for Fast Data Manipulation (1st ed.). O'Reilly Media.

Müller, A. C., & Guido, S. (2016). Introduction to Machine Learning with Python: A Guide for Data Scientists. O'Reilly Media.

Ng, A. (2018). Machine Learning Yearning. Self-published.

Obe, R. O., & Hsu, L. S. (2017). PostgreSQL: A Practical Guide to the Advanced Open Source Database (3rd ed.). O'Reilly Media.

Petrov, A. (2019). Database Internals: A Deep Dive into How Distributed Data Systems Work. O'Reilly Media.

Redis. (2024). Docs. Pridobljeno iz Redis: https://redis.io

Redis. (2024). Pricing. Pridobljeno iz Redis: https://redis.io/pricing/#monthly

Riggs, S., & Ciolli, G. (2022). PostgreSQL 14 Administration Cookbook: Over 175 proven recipes for database administrators to manage enterprise databases effectively. Packt Publishing.

Schönig, H. (2023). Mastering PostgreSQL 15: Advanced techniques to build and manage scalable, reliable, and fault-tolerant database applications (5th ed.). Packt Publishing.

Stackoverflow. (2024). Questions. Pridobljeno iz Stackoverflow: https://stackoverflow.com/questions/

Wikipedia. (2024). Database. Pridobljeno iz Wikipedia: https://en.wikipedia.org/wiki/Database

9 PRILOGE

Priloga 1: Python skripte

Na priloženem USB ključku se nahajajo naslednje Python skripte, ki so ključne za delovanje in analizo v diplomskem delu:

- Skripte za uvoz podatkov v Redis in PostgreSQL baze.
- Skripta za izvajanje poizvedb v PostgreSQL in Redis bazah, ki primerja hitrost pridobivanja podatkov in filtriranja po določenih kriterijih.
- Skripta za merjenje porabe sistemskih virov (CPU in RAM) med celotnim procesom.
- Osrednja skripta za treniranje modelov, meritve njihove učinkovitosti in izvajanje poizvedb.

Priloga 2: CSV datoteke

Priložene CSV datoteke vsebujejo podatke, ki so bili uporabljeni za treniranje modelov in izvajanje analiz:

- *competitions.csv* – Podatki o tekmovanjih.
- *clubs.csv* – Podatki o klubih.
- *player_stats.csv* – Statistični podatki o igralcih.
- *players.csv* – Podatki o igralcih.