Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Marko Mikuletič

Primerjava relacijskih, NoSQL in NewSQL podatkovnih baz

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: doc. dr. Luka Šajn

Ljubljana, 2015

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

V zadnjem času se je pojavilo veliko novih različic podatkovnih baz, ki pa niso vse zgrajene na relacijskem modelu. V okviru diplomskega dela naj kandidat opiše nove podatkovne baze NoSQL in NewSQL ter naredi primerjavo med relacijsko, NoSQL in NewSQL podatkovno bazo, v praktičnem delu pa naj naredi primerjavo v hitrosti izvajanja med izbrano relacijsko, NoSQL in NewSQL podatkovno bazo.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Marko Mikuletič, z vpisno številko **63080312**, sem avtor diplomskega dela z naslovom: Primerjava relacijskih, NoSQL in NewSQL podatkovnih baz.

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Luke Šajna,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 28. januarja 2015 Podpis avtorja:

Zahvala

Iskreno se zahvaljujem mentorju doc. dr Luki Šajnu za koristne nasvete in napotke pri izdelavi diplomskega dela ter za strokoven pregled. Hvala za ustrežljivost in prijaznost.

Posebna zahvala gre staršem, mami Sabrini in očetu Joškotu za spodbudo, podporo, razumevanje in potrpežljivost med mojim študijem.

Staršem

Kazalo

Seznam uporabljenih kratic

Kratica	Angleško	Slovensko
ACID	Atomicity, Consisteny	atomarnost, konsistentnost,
	Isolation, Durability	izolacija, trajnost
API	application programming interface	aplikacijski programski vmesnik
BASE	Basically Available, Soft State	večinoma razpoložljiv, mehko stanje,
	Eventually Consistent	sčasoma dosledno
BLOB	binary large object	veliki binarni objekt
CAP	Consistency, Availability,	konsistentnost, razpoložljivost,
	Partition tolerance	particijska toleranca
CPU	central processing unit	centralna procesna enota
DBMS	Database Management System	sistem za upravljane podatkovnih baz
DDL	Data Definition Language	jezik za definiranje podatkovnih
		struktur in tipov
DML	Data Manipulation Language	jezik za manipulacijo s podatki
DSS	Decision Support System	sistem za podporo odločanja
JDBC	Java Database Connectivity	Java API za dostop do podatkovne baze
JSON	java script object notation	format zapisa za izmenjavo podatkov
LDBC	Librery DataBase Conectivity)	JDBC driver, ki omogoča
		dostop do podatkovne baze
OLAP	online analytical processing	analitična obdelava podatkov
OLTP	Online Transaction Processing	spletna obdelava transakcij
DB/PB	database	podatkovna baza
RAM	random acess memory	pisalni pomnilnik
RDBMS	Relational Database	sistem za upravljanje
	Managament System	relacijskih podatkovnih baz
SQL	Structured Query Language	poizvedovalni jezik v relacijskih
		podatkovnih bazah
SSD	solide state drive	
SUPB		sistem za upravljanje s
		podatkovnimi bazami
XML	Extensible Markup Language	format za izmenjavo
		strukturiranih podatkov v spletu

Povzetek

Diplomsko delo obravnava najnovejše področje podatkovnih baz, to so NewSQL podatkovne baze. Cilj diplomskega dela je opisati No SQL in NewSQL podatkovne baze in jih primerjati z relacijsko podatkovno bazo. NewSQL podatkovne baze uporabljajo isti jezik SQL kot relacijske podatkovne baze, imajo isto ACID podporo transakcij kot relacijske podatkovne baze, medtem ko NoSQL podatkovne baze sledijo teoremu CAP v skladu z načelom BASE. NewSQL podatkovne baze so prav tako kot NoSQL podatkovne baze horizontalno skalabilne. V prvem poglavju je predstavljen uvod v diplomsko delo, sledi uvod v podatkovne baze in opis relacijskih podatkovnih baz. V tretjem in četrtem poglavju so predstavljene NoSQL in NewSQL podatkovne baze. Predstavili smo, kaj so te baze, njihove značilnosti, kategorizacijo in opisali primere NewSQL podatkovnih baz. Naredili smo primerjavo med relacijskimi, NoSQL in NewSQL podatkovnimi bazami. Zadnji dve poglavji opisujeta pripravo na testiranje podatkovnih baz in testiranje oz. merjenje časa izvajanja podatkovnih baz. Zaključno poglavje predstavi sklepne ugotovitve o obravnavanih podatkovnih bazah in obete za njihovo prihodnost.

Ključne besede: relacijska podatkovna baza, SQL, NoSQL, NewSQL, ACID, CAP, BASE, računalnik.

Abstract

The thesis deals with the newest area of databases that are NewSQL database. The aim of the thesis is to describe NoSQL and NewSQL databases and compare them with relational database. NewSQL database is using the same language SQL as relational databases, with ACID support for transactions same as a relational database, while NoSQL database follow the CAP theorem, in accordance with the principle of BASE. NewSQL database also as NoSQL database are horizontally scalable. The first chapter is an introduction to the study, followed by an introduction to databases and description of relational databases. The third and the fourth chapter presents NoSQL and NewSQL database. We presented what these bases, their characteristics, categorize and describe examples of NewSQL databases. We made a comparison between the relational, NoSQL and NewSQL databases. The last two chapters describe the preparation of test databases and testing respectively measurement runtime of databases. The final section presents conclusions about the studied databases and makes some predictions about their future.

Keywords: relational database, SQL, NoSQL, NewSQL, ACID, CAP, BASE, computer.

Poglavje 1

Uvod

Podatkovna baza je avtomatizirana, večuporabniška, formalno definirana in (centralno) nadzorovana zbirka logično povezanih podatkov. Lahko si jo predstavljamo kot računalniški sistem za shranjevanje podatkov ali kot klasično knjižnico. S podatkovnimi bazami se srečujemo tudi v vsakdanjem življenju. Do pojava podatkovnih baz je prišlo zaradi potrebe po hitrem dostopu do informacij.

Pojem podatkovna baza se je prvič pojavil v zgodnjih 60. letih, relacijska podatkovna baza oz. relacijski podatkovni model pa se je pojavil v 70. letih, ko je Edgar Codd predlagal relacijski podatkovni model. V 80. letih se je razvil SQL-poizvedovalni jezik, ki je postal standardni jezik za izvajanje poizvedb v relacijski podatkovni bazi. Pojavilo se je veliko proizvajalcev SUPBJ-jev (sistem za upravljanje s podatkovno bazo). Najnovejša področja podatkovnih baz so nerelacijske podatkovne baze oz. NoSQL in NewSQL podatkovne baze. NoSQL podatkovne baze ne vsebujejo relacijskega podatkovnega modela, podatki so tipično nerelacijski in kot poizvedovalni jezik po poizvedbah ne uporabljamo SQL-ja. Rešiti poskušajo problematiko razširljivosti, prilagodljivosti in dostopnosti. NewSQL je nov poizvedovalni jezik in ni nadgradnja SQL-ja. Je lažji, konsistentnejši, elegantnejši in bolje definiran kot SQL.

V teoretičnem delu bomo predstavili relacijske, NoSQL in NewSQL podatkovne baze. V praktičnem delu bomo naredili primerjavo v hitrosti izvajanja med MySQL, Cassandro in NuoDB. Cilj diplomskega dela je opis novih NoSQL in NewSQL podatkovnih baz in primerjava med relacijsko, NoSQL in NewSQL podatkovno bazo. Zaradi narejene primerjave se bodo kupci podatkovnih baz lažje odločali med njimi.

Diplomsko delo je razdeljeno na šest poglavij. V prvem poglavju z naslovom Relacijska podatkovna baza bomo predstavili pojem podatkovne baze, značilnosti relacijskih podatkovnih baz, lastnosti transakcij ACID in SQL poizvedovalnega jezika.

Drugo poglavje NoSQL podatkovne baze predstavi definicijo NoSQL, lastnosti in tehnologijo porazdeljenih baz NoSQL. Spoznali bomo pojme replikacija, fragmentacija, konsistentnost, podatkovni model MapReduce, načelo BASE in teorem CAP. Spoznali bomo tudi vrste NoSQL podatkovnih baz, kot so ključ-vrednost shramba, široko stolpčne shrambe, dokumentne shrambe in grafične podatkovne baze.

Tretje poglavje NewSQL-podatkovne baze predstavi definicijo NewSQL podatkovnih baz. Predstavili bomo značilnosti teh baz, in sicer sisteme, arhitekturo in priljubljenost. Ogledali si bomo kategorizacijo NewSQL podatkovnih baz in spoznali nove podatkovne baze, nove skladiščne motorje MySQL, transparentno grozdenje in podatkovno bazo kot storitev. Sledila bo predstavitev nekaterih NewSQL podatkovnih baz.

Četrto poglavje z naslovom Primerjava relacijskih, NoSQL in NewSQL podatkovnih baz predstavi njihovo primerjavo, peto poglavje Priprava na testiranje pa predstavi testno okolje, pripravo testnih podatkov, programske jezike za testiranje in pripravo testnih skript.

Šesto poglavje Testiranje na podatkih predstavi uporabniški scenarij, kjer so prikazani različni scenariji pri podatkovnih bazah (branje, vstavljanje, brisanje). V tem poglavju smo merili čas izvajanja med relacijskimi, NoSQL in NewSQL podatkovnimi bazami.

V zadnjem poglavju z naslovom Sklep bomo povzeli ugotovitve in si ogledali napovedi za prihodnost NewSQL podatkovnih baz.

Poglavje 2

Relacijska podatkovna baza

2.1 Podatkovna baza

Podatkovna baza je sklop zbirke dokumentov, medsebojnih sklicevanj na dokumente in sistema za razvrščanje, iskanje in urejanje podatkov v zbirki [1]. Podatkovna baza je zbirka medsebojno logično povezanih podatkov (in opisov podatkov), ki zadovoljujejo informacijske potrebe organizacije in njenih poslovnih procesov [2].

Poznamo tudi sisteme za upravljanje podatkovnih baz, kot je npr. SUPB. To je skupek programske opreme, ki omogoča kreiranje, vzdrževanje in nadzor nad dostopom do podatkov v podatkovni bazi. Kreiranje podatkovnih struktur je omogočeno prek DDL (Data Definition Language), upravljanje s podatki (Create, Insert, Update, Delete) pa prek DML (Data Manipulation Language) [2].

Model, s katerim opišemo, kaj bi želeli hraniti in kakšne povezave obstajajo med elementi, ki jih želimo hraniti, se imenuje podatkovni model [3]. Poznamo več vrst podatkovnih baz oz podatkovnih modelov, vsa pa služijo svojemu namenu. Podatkovne baze razdelimo na hierarhične, mrežne, relacijske in na objektne. Mrežne in hierarhične danes niso več v uporabi, objektne

pa se uporabljajo v posebne namene. Večina današnjih podatkovnih baz spada pod relacijski tip [1].

2.2 Relacijska podatkovna baza

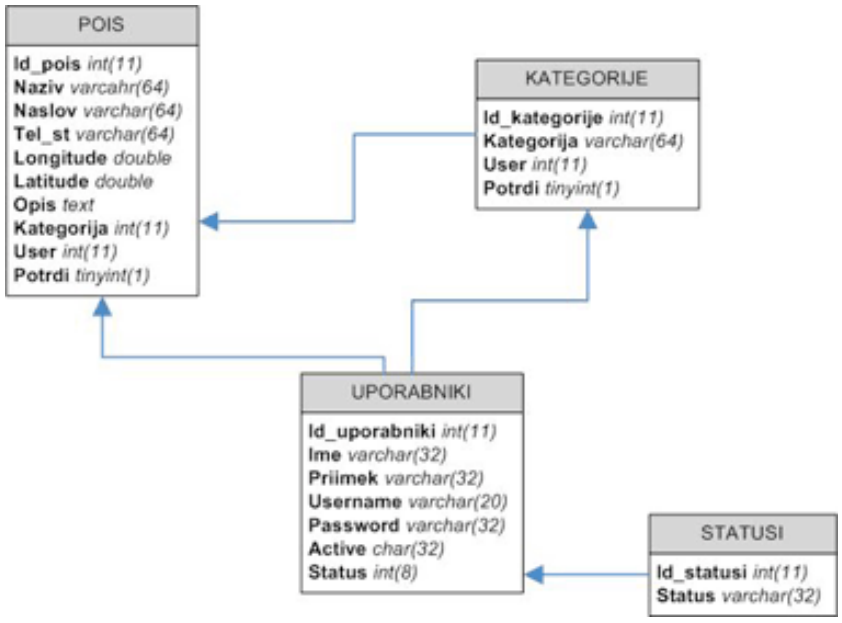
Relacijska podatkovna baza oz relacijski podatkovni model se pojavi leta 1970, ko je EF Codd, raziskovalec pri IBM-u, napisal knjigo, ki opisuje proces. Relacijska podatkovna baza je zbirka podatkov, ki shranjuje informacije o podatkih in kako so ti povezani. Programska oprema za relacijsko podatkovno bazo se imenuje sistem za upravljanje relacijskih podatkovnih baz (RDBMS- Relational Database Management System) in nadzoruje branje, pisanje, spreminjanje ter obdelavo informacij, shranjenih v podatkovni bazi [4].

Podatkovna baza, ki temelji na relacijskem modelu je predstavljena z množico relacij oz odnosov. Vsaka relacija je predstavljena kot tabela z vrsticami in stolpci (atributi) [3]. Sistem za upravljanje relacijskih podatkovnih baz (RDBMS) je programska oprema, ki omogoča ustvarjanje in uporabo relacijskih podatkovnih baz [5].

Na voljo so enostavni, vendar močni jeziki za poizvedovanje po podatkovni bazi [3]. Relacijska podatkovna baza uporablja strukturiran jezik poizvedb (SQL), ki je standardna uporabniška aplikacija in omogoča enostaven programski vmesnik za interakcijo s podatkovnimi bazami [6].
Pri relacijskih podatkovnih bazah nastanejo problemi zaradi redundance podatkov v osnovnih relacijah. Normalizacija je postopek, s katerem pridemo do množice primernih (primerno strukturiranih) relacij, ki ustrezajo potrebam uporabe [3]. S postopkom normalizacije odstranimo ažurne anomalije(relacije, ki vsebujejo odvečne podatke) pri operacijah nad podatki [3, 7]. Obratni postopek se imenuje denormalizacija (znižanje stopnje normalne oblike), ki je primeren v določenih primerih [7].

Uporabniki						
Id uporabnika	Ime	Priimek	Username	Password	Active	Status
1	Janez	Novak	janeznovak1	gozdnaPot	YES	0
2	Mitja	Gorišek	mitja-gorišek	blečaajs	YES	0
3	Tamara	Novakovič	tamara14	norakov	YES	0
4	Rok	Kuret	rok-kuret8	košarkar	NO	1
5	Maja	Kraševec	maja10	vincent	YES	0

Slika 2.1: Primer tabele relacijske podatkovne baze



Slika 2.2: Primer relacijskega podatkovnega modela

2.2.1 Lastnosti transakcij ACID

ACID je skupek lastnosti, ki zagotavlja, da se transakcije v podatkovni bazi zanesljivo obdelujejo [8]. ACID je kratica za atomarnost (angl. *atomicity*), konsistentnost (angl. *consistency*), izolacijo (angl. *isolation*) in trajnost (angl. *durability*). Transakcija je operacija ali niz operacij, ki berejo ali pišejo v podatkovno bazo ter so izvedene s strani enega uporabnika oziroma uporabniškega programa [7].

Vsaka transakcija mora zadoščati štirim osnovnim lastnostim.

Atomarnost: transakcija predstavlja atomaren sklop operacij. Ali se izvede vse ali nič. Atomarnost mora zagotavljati SUPB [3].
Konsistentnost: transakcija je sklop operacij, ki podatkovno bazo privede iz enega konsistentnega stanja v drugo. Zagotavljanje konsistentnosti je naloga SUPB (zagotavlja, da omejitve nad podatki niso kršene) in programerjev (preprečuje vsebina neskladnosti) [3].
Izolacija: transakcije se izvajajo neodvisno ena od druge → delni rezultati transakcije ne smejo biti vidni drugim transakcijam. Za izolacijo skrbi SUPB [3].
Trajnost: učinek potrjene transakcije je trajen. Če želimo njen učinek razveljaviti, moramo to narediti z novo transakcijo, ki z obratnimi operacijami podatkovno bazo privede v prvotno stanje. Zagotavljanje trajnosti je naloga SUPB [3].

2.2.2 Poizvedovalni jezik SQL

SQL je do sedaj edini široko sprejeti standardni podatkovni jezik [3]. SQL ali strukturirani povpraševalni jezik za delo s podatkovnimi bazami (angl. Structured Query Language) je najbolj razširjen in standardiziran povpraševalni jezik za delo s podatkovnimi zbirkami, s programskimi stavki, ki posnemajo ukaze v naravnem jeziku. SQL jezik je določen z ANSI/ISO SQL standardom [9].

SQL je transformacijsko usmerjen jezik, ki ga sestavljata dve skupini ukazov:

- skupina ukazov DDL (Data Definition Language) za opredelitev strukture podatkovne baze,
- skupina ukazov DML (Data Manipulation Language) za poizvedovanje in ažuriranje podatkov (vključuje tudi DQL) [3].

Poglavje 3

Podatkovne baze NoSQL

Relacijske podatkovne baze so bile včasih uporabne za vsak problem. Danes je problem v porazdeljenosti (transakcije, stične operacije) [10]. Relacijske podatkovne baze imajo omejeno razširljivost, problem pa je tudi v tem, da je razpoložljivost bolj pomembna kot doslednost. Drugi problemi relacijskih podatkovnih baz so veliko število podatkov, veliko istočasnih uporabnikov in veliko zahtevanih obdelav. Težave se pojavljajo tudi pri prekompleksnem neobvladljivem modelu in če je veliko začasnih podatkov, ki niso povezani z glavnimi [10]. NoSQL je nova generacija v razvoju podatkovnih baz. NoSQL podatkovne baze uporabljajo npr. Twitter, Facebook, Google, Digg, eBay in drugi. Po podatkih s strani NoSQL-database.org je trenutno 150 različnih NoSQL podatkovnih baz [12]. Primeri teh baz so Voldermort, Cassandra, MongoDB, CouchDB in BigData.

3.1 Definicija

NoSQL podatkovna baza (Not Only SQL) zagotavlja mehanizem za shranjevanje in dostop do podatkov, ki je modelirana po vzoru drugih, kot v tabeli odnosov, ki se uporabljajo v relacijskih podatkovnih bazah [16]. NoSQL podatkovne baze za razliko od klasičnih podatkovnih baz ne potrebujejo tabelarične sheme, po navadi ne uporabljajo «join» operacij in so po navadi

horizontalne [13]. NoSQL podatkovne baze ne podpirajo poizvedovalnega jezika SQL in niso relacijske. NoSQL sistemi ne zagotavljajo ACID transakcijiskih lastnosti [14]. V primerjavi z relacijskimi podatkovnimi bazami so te baze bolj prilagodljive in zagotavljajo boljšo zmogljivost [15]. NoSQL ima naslednje lastnosti: pogosto je odprtokoden, ni fiksne sheme tabel, združitvi (JOIN) se izognemo, redundanca je v redu, horizontalno luščenje [12].

NoSQL sistemi imajo običajno šest ključnih značilnosti
1. zmožnost horizontalne skalabilnosti, «enostavna operacija», pretok preko številnih strežnikov
2. sposobnost, da ponovi in porazdeli (particije) podatkov prek številnih strežnikov
3. preprost nivo klica vmesnika ali protokola (v nasprotju s SQL vezavo)
4. šibkejši model sočasnosti, in sicer od ACID transakcij do večine relacijskih(SQL) podatkovnih baz
5. učinkovita uporaba porazdeljenih indeksov in RAM-a za shranjevanje podatkov, in
6. zmožnost za dinamično dodajanje novih atributov v podatkovne zapise [14].

3.2 Lastnosti in tehnologija porazdeljenih baz NoSQL

Poznamo tri dimenzije razširljivosti podatkovnih baz, in sicer po številu pisalnih operacij, velikosti podatkovne baze in po številu bralnih operacij. Zagotavljamo jo s pomočjo replikacije in fragmentacije (angl. *sharding*), navadno s kombinacijo [10].
Replikacija (angl. *replication*)
Replikacija je zapis podatkovnih enot na več kot eno vozlišče, kar zagotavlja učinkovitejše branje (agl. *load balancer*) in večjo zanesljivost (odpornost proti odpovedi posameznega vozlišča) [10].

Fragmentacija (delitev, angl *sharding*)
Fragmentacija je razbitje podatkov na fragmente. Drugo ime za fragmentacijo je delitev. Fragmenti so zapisani na posameznih vozliščih [10].
Konsistentnost...vrste
Stroga konsistentnost pomeni, da so vse verzije posamezne podatkovne enote enake, pri postopni konsistentnosti pa bodo vse verzije posamezne podatkovne enote sčasoma enake [10].
Podatkovni model MapReduce
MapReduce je programski model/ogrodje za porazdeljeno računanje, ki ga je razvil Google. Ogrodje MapReduce poskrbi za distibucijo po n strežnikih, za sinhronizacijo in paralelizacijo. Primeren je za obdelavo velikih količin podatkov, ki jih ne moremo obdelati na enem strežniku [10].

Načelo BASE in teorem CAP
ACID model zagotavlja zanesljivost podatkovne baze, vendar obstajajo številni primeri, kjer ni mogoče uporabiti teh modelov. To je povzročilo, da so pred kratkim naredili NoSQL DBMS, ki sledi CAP teoremu in je v skladu z načelom BASE [17].
CAP teorem pravi, da pri deljenju podatkov lahko optimiziramo **največ dve** od teh treh lastnosti [10]. CAP izrek se imenuje tudi Brewerjev izrek (po Eric Brewer) [17].
Konsistentnost (angl. *Consistency*): pomeni, da podatki v podatkovni bazi ostanejo konsistentni po izvedbi operacije [18].
Razpoložljivost (angl *availability*): pomeni, da lahko pričakujemo, da se vsaka operacija konča v pričakovanem odzivu [17].
Odpornost proti odpovedim posameznih particij (angl. *Partition tolerance*): V podatkovno bazo je še vedno mogoče pisati in brati, tudi ko deli baze niso dostopni. Ko so nedostopna vozlišča spet dostopna, so ustrezno posodobljeni [17].
POMEMBNO: teorem CAP navaja, da je za vsako sistemsko delitev podatkov nemogoče zagotoviti vse tri lastnosti, zato so možne kombinacije CP, AP

in CA (težko združljive). Glede na teorem CAP, razpoložljivost ne more biti vedno zagotovljena, ne da bi sprostili doslednost [17].
Porazdeljeni sistemi morajo biti particijsko tolerantni (P), zato moramo zbrati med usklajenostjo in razpoložljivostjo. Trenutne podatkovne baze NoSQL sledijo različnim kombinacijam med C in A iz izreka CAP [18].
Žrtvovati je potrebno enega od kriterijev, in sicer C, A ali P

- **konsistentni in razpoložljivi (CA)** sistemi imajo težavo z deljenjem podatkov, ki jo običajno obvladujejo z replikacijo,
- **konsistentni, porazdeljeni sistemi (CP)** imajo težavo z razpoložljivostjo podatkov ob naporu zagotavljanja njihove konsistentnosti,
- **razpoložljivi, porazdeljeni sistemi (AP)** dosegajo sočasno konsistentnost (angl. eventual consistentcy) z replikacijo podatkov in občasnim preverjanjem konsistentnosti v podatkovnih vozliščih [10].

Načelo BASE

BASE je skupek lastnosti, ki so večinoma razpoložljiv (angl. *basically available*), mehko stanje (angl. *soft state*) in eventuelna konsistentnost (angl. *eventually consistent*) [19].
Večinoma razpoložljiv: gre za sistem podatkovne baze, kjer se vedno zdi, da deluje [18]. Večinoma razpoložljiv kaže, da sistem zagotavlja razpoložljivost v skladu s CAP teoremom [22].
Mehko stanje: ni nujno, da so ves čas skladni oz konsistentni [18]. Mehko stanje kaže, da se lahko stanje sistema v času spreminja tudi brez vnosa. To je zaradi morebitne doslednosti modela [22].
Eventuelna konsistentnost: kaže, da bo sistem postal sčasoma skladen ob upoštevanju, da sistem v tem času ne prejme vnosa [22].
Lastnosti načela BASE bi lahko povzeli: aplikacija deluje v bistvu ves čas (večinoma razpoložljiv), ni treba, da je nenehno skladna oz konsistentna (mehko stanje), vendar bo sčasoma v nekem znanem stanju (eventuelna konsistentnost) [20].

3.3 Vrste NoSQL podatkovnih baz

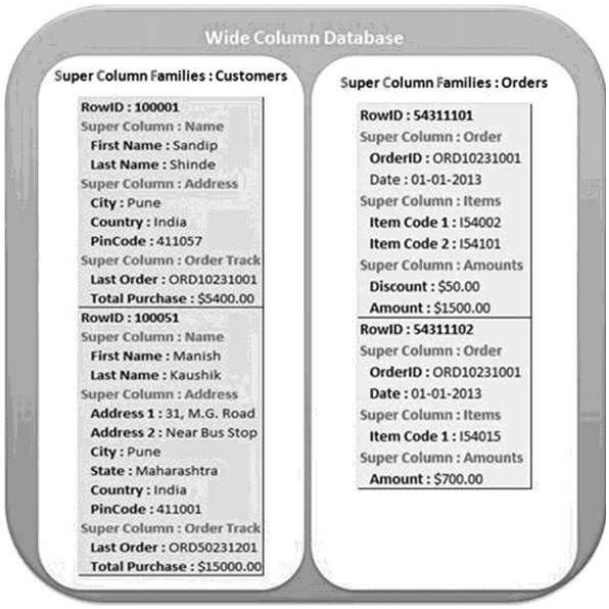
Ključ-vrednost shrambe (angl. *Key-Value Store*)
Ključ-vrednost shrambe uporabljajo asociativno polje (znano tudi kot zemljevid ali slovar) kot svoj temeljni podatkovni model. V tem modelu so podatki predstavljeni kot zbirka parov ključ – vrednost, tako da se vsak možni ključ pojavi največ enkrat v zbirki [16].
Problemi pri shrambah ključ – vrednost so nestrukturirani in nekonsistentni podatki, kar pomeni, da ni sheme, ni praznih polj in uporaba SQL ni možna. Fragmentacija se naredi s pomočjo razpršitve (angl. *hash*), replikacija pa na n-predhodnih vozliščih. Pri konsistentnosti je podprta stroga in postopna konsistentnost [10]. Primeri shramb ključ-vrednost so: Azure Table Storage, MEMBASE, GenieDB, Tokyo Cabinet /mTyrant, MemcacheDB, Amazon Dynamo, Voldemort, BerkleyDB, Keyspace in Dynomite.

Car	
Key	Attributes
1	Make: Nissan Model: Pathfinder Color: Green Year: 2003
2	Make: Nissan Model: Pathfinder Color: Blue Color: Green Year:2005 Transmission: Auto

Slika 3.1: Primer shrambe ključ-vrednost

Široko stolpčne shrambe (angl. *Wide Column Store*)
Široko stolpčne shrambe, imenovane tudi shrambe razširljivih zapisov, imajo shranjevanje podatkov v evidencah, ki imajo sposobnost zelo velikega števila dinamičnih stolpcev. Ker imena stolpcev kot tudi zapisi ključev niso fiksni in ker ima lahko zapis milijarde stolpcev, lahko vidimo širokostolpčne shrambe kot dvodimenzionalne ključ – vrednost shrambe. Širokostolpčne shrambe si

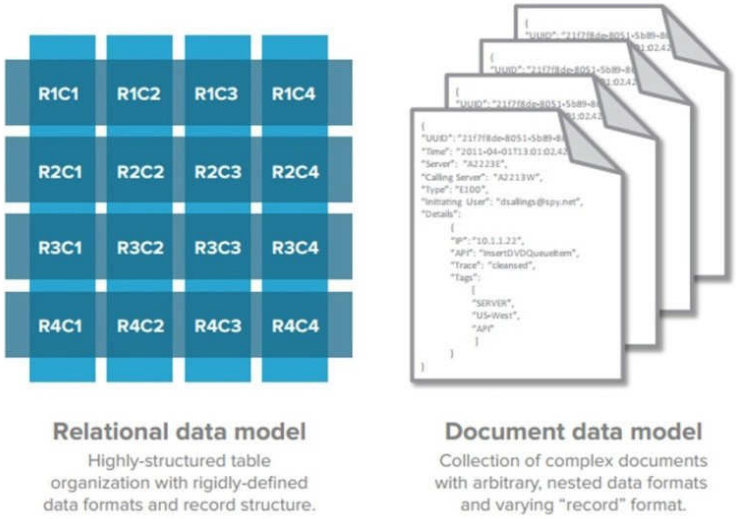
delijo značilnosti z dokumentnimi shrambami, so brez sheme, vendar pa je izvajanje zelo različno [21].
Za to vrsto baz so značilne velike količine podatkov. Stolpce združujemo v družine stolpcev, ki postanejo super stolpci. Stolpce združujemo v družine super stolpcev, ki postanejo tabele [10].
Primeri široko stolpčnih shramb so: Hadoop, Cassandra(Digg, Facebook, Twitter, Rackspace), Cloudata, Cloudera, Amazon SimpleDB in Hypertable.



Slika 3.2: Primer široko stolpčne shrambe

Dokumentna shramba (angl. *document store*)
V dokumentni shrambi je osnovna podatkovna enota dokument. Dokument je delno strukturiran. Standardni formati so XML, YAML, JSON, binarni formati pa so PDF, MS Word itd. Dokumenti so le podobni, neenaki in nimajo predpisane sheme. Organizacija dokumentov je odvisna od implementacije, koncepti pa so Bucket, Tag, Collection. Vsak dokument ima svoj ključ (npr. strin, URI), dokumenti pa so indeksirani. Poizvedovanje v do-

kumentnih shrambah je po ključu (za cel dokument). Navadno dokumentne shrambe uporabljajo tudi API ali poizvedovalni jezik za poizvedovanje po vsebini dokumentov. Med implementacijami obstajajo velike razlike. Zelo popularni dokumentni bazi sta MongoDB in CouchDB [10].
Primeri dokumentnih shramb so MongoDB (Foursquare, SourceForge, Fotopedia, Joomla Ads), CouchDB(CERN, BBC) , Citrusleaf in Redis.



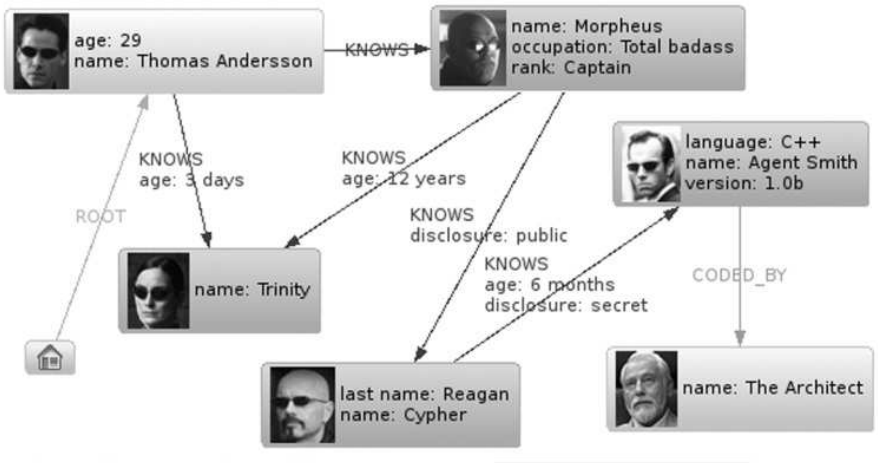
Slika 3.3: Primer dokumentne shrambe

Grafične podatkovne baze (angl. *graph databases*)
Grafične podatkovne baze so baze, ki so specializirane za shranjevanje podatkov, predstavljenih v obliki grafov (elementi so med seboj povezani z nedoločenim številom odnosov med njimi). Vozlišča in povezave imajo lahko lastnosti [10]. Vrsta podatkov bi lahko bili družbeni odnosi, povezave javnega prevoza, avtokarte ali omrežne topologije. Primeri grafičnih podatkovnih baz so: Neo4J, Infinite Graph, Bigdata, AllegroGraph, InfoGrid, Sones, graphDB in FlockDB.

Lastnosti:

- shramba specializirana za hranjenje podatkovnih struktur, s katerimi opisujemo grafe,

- shramba optimizirana za sprehajanje po grafu brez uporabe indeksov...,
- namesto globalnih poizvedb, iskanje iz izbranih vozišč navzven,
- fleksibilen podatkovni model – podatkovni tipi vozlišč in povezav nedefinirani,
- močno teoretično ozadje – teorija grafov... številni algoritmi (shortest path, Dijsktra, A*, betweenness, . . .) [10].



Slika 3.4: Primer grafične podatkovne baze

Poglavje 4

Podatkovne baze NewSQL

NewSQL je spodbudilo gibanje novih vrst relacijskih podatkovnih baz (ali razširitev za obstoječe relacijske podatkovne baze). Njen namen je zagotoviti enako prilagodljivo zmogljivost NoSQL sistemov, vendar še vedno temelji na relacijski paradigmi in ohranja stari dobri SQL kot poizvedovalni jezik [23]. NewSQL podatkovne baze so primarno namenjene predvsem za podjetja, ki se ukvarjajo s podatki visokega profila in zahtevajo prilagodljivost, potrebujejo pa tudi več doslednosti, kot jih lahko zagotovijo NoSQL podatkovne baze [24]. Pričakovanje je da so NewSQL podatkovne baze približno 50 krat hitrejše od relacijskih podatkovnih baz [25]. Primeri NewSQL podatkovnih baz so VoltDB, ClustrixDB, NuoDB, GenieDB, ScaleArc, ScaleDB, Tokutek/TokuDB, JustoneDB, MemSQL, Splice Machine in Translattice.

4.1 Definicija

Izraz NewSQL je bil prvič uporabljen leta 2011, v raziskovalnem članku analitika Matthew-a Aslett-a o pojavu novih sistemov iz skupine 451 [27]. NewSQL je razred sodobnih sistemov za upravljanje relacijskih podatkovnih baz, ki si prizadevajo zagotoviti enako razširljive zmogljivosti NoSQL sistemov za spletno obdelavo transakcij(OLTP), branje in pisanje delovnih obremenitev, medtem ko še vedno ohranja ACID jamstva od tradicionalnega

sistema podatkovnih baz [26]. NewSQL z združevanjem zanesljivosti SQL in s hitrostjo ter uspešnostjo NoSQL zagotavlja izboljšano funkcionalnost in storitve [28]. Načrtovan je bil z namenom omogočanja zahtev po masivnem horizontalnem skaliranju, a je bil performančni napredek tako velik, da horizontalno skaliranje ni več potrebno [10].

Značilnosti NewSQL podatkovnih baz.

- SQL kot osnovni mehanizem za interakcijo aplikacij,
- ACID podpora za transakcije,
- nezaklepni mehanizem nadzora sočasnosti, tako da realno, časovno branje ne bo v konfliktu z branjem in s tem ne bo povzročilo, da se zavleče,
- arhitektura zagotavlja veliko večjo zmogljivost na vozlišču kot tiste, ki so na voljo v tradicionalnih RDBMS rešitvah,
- Horizontalno skaliranje, nič skupna arhitektura, ki lahko teče na velikem številu vozlišč brez trpljenja ozkih grl [25].

NewSQL je lažji, konsistentnejši, elegantnejši in bolje definiran kot SQL. Poizvedovalni jezik SQL je zastarel, izredno kompleksen in ima veliko pomankljivosti. NewSQL temelji na «cross database library LDBC». Prvotna ideja je bila, da bi bil NewSQL standardiziran [27].

NewSQL naj bi podpiral že obstoječe podatkovne baze. Narejena bosta dva konverterja, in sicer SQL → NewSQL in NewSQL → SQL. Pri tem mislimo na vse obstoječe večje dialekte SQL-a. To ne bo samo pomagalo pri migraciji iz enega sistema na drugega, ampak bo tudi dovoljevalo poganjanje že obstoječih aplikacij z različnimi SUPB-ji [27].

SQL	NewSQL 'Jdb'	NewSQL 'S2'
CREATE TABLE TEST(ID INT PRIMARY KEY, NAME VARCHAR(255))	test=new table(int id, string name, key(id))	create table test(id int, name string, primary key(id))
INSERT INTO TEST VALUES(1,'Hello')	test.add(1,"Hello")	insert test (1,'Hello')
SELECT * FROM TEST	test.get()	select test
SELECT T1.ID,T2.NAME FROM TEST T1, TEST T2 WHERE T.ID=T2.ID	t1=test; t2=test; t1.join(t2[t1.id==t2.id]).get(t1.id,t2.name)	select t1:test join t2:test on t1.id==t2.id get t1.id, t2.name
UPDATE TEST SET NAME='Hi' WHERE ID=1	test[id==1].set(name="Hi")	update test set id=1 where name=='Hi'
DELETE FROM TEST WHERE ID=1	test[id==1].delete()	delete test where id==1
DROP TABLE TEST	test.drop()	drop test

Slika 4.1: Prva ideja, kako bi bila videti standardizirana verzija

4.2 Značilnosti NewSQL podatkovnih baz

Sistemi

Čeprav se NewSQL sistemi zelo razlikujejo v svojih notranjih arhitekturah, sta med njimi dve razpoznavni značilnosti, da vsi podpirajo relacijski podatkovni model in uporabljajo SQL kot primarni vmesnik. Aplikacije, ki jih obravnavajo ti NewSQL sistemi, so označene, kot da imajo veliko število transakcij (1), ki so kratkotrajne (2), se dotaknejo majhne podmnožice podatkov z uporabo indeksa poizvedbe in (3) se ponavljajo (tj. izvršitve enake poizvedbe z različnimi vhodi). Ti NewSQL sistemi so dosegli visoko zmogljivost in razširljivost z izogibanjem velike zapuščine arhitekture prvotnega IBM System R-zasnove kot težjega okrevanja ali algoritma za nadzor sočasnosti. Od prvih znanih NewSQL sistemov je vzporedni sistem podatkovne baze H-Store [26].

Arhitektura

Prvi tipi NewSQL-sistemov so popolnoma nove platforme podatkovnih baz. Te so namenjene za delovanje v porazdeljenih grozdih z deljenimi nič vozlišči, v katerih ima vsako vozlišče v lasti podmnožico podatkov. Te podatkovne baze so pogosto napisane iz nič s porazdeljeno arhitekturo v mislih in vključujejo komponente, kot so porazdeljen nadzor sočasnosti, nadzor pretoka in porazdeljene povpraševalne poizvedbe. Primeri sistema v tej kategoriji so Google Spanner, Clustrix, VoltDB, MemSQL, Pivotal SQLFirein Gemfire

XD, SAP HANA, FoundationDB, NuoDB, TransLattice, ActorDB in Trafodion [26].

Priljubljenost

Do sedaj so bile NewSQL možnosti precej manj priljubljene kot NoSQL, deloma zato, ker so zelo nove, a tudi zato, ker relacijski pristop in fleksibilnost podatkov ni enostavno kombinirati. NewSQL prodajalci se soočajo z večjo konkurenco starih prodajalcev SQL in samozadovoljnih strank. Prodajalci NewSQL-ja trdijo, da je to draga napaka [29].

4.3 NewSQL kategorizacija

Razvrstitev temelji na različnih pristopih, ki so jih prodajalci sprejeli za ohranitev vmesnika SQL in za obravnavanje razširljivosti ter pomislekov zmogljivosti tradicionalnih OLTP rešitev [25]. Obstajajo štiri kategorije NewSQL podatkovnih baz, in sicer nove podatkovne baze (angl. *new data bases*), podatkovna baza kot storitev(angl. *database-as-a-service*), shranjevanje motorjev(angl. *storage engines*) in grozdenje ali fragmentacija (angl. *clustering or sharding*) [30].

1.Nove podatkovne baze: to so vrste relacijskih podatkovnih baz, ki so bile v zadnjem času razvite za horizontalno razširljivost, visoko zmogljivost in razpoložljivost [30]. Ti NewSQL sistemi so na novo oblikovani iz nič z namenom, da se dosežeta prilagodljivost in učinkovitost. Seveda pa bodo potrebne nekatere (upajmo majhne) manjše spremembe kode, še vedno pa je potrebna migracija podatkov. Eden od ključnih premislekov v izboljšanju učinkovitosti je kar ne-disk (pomnilnik) ali nove vrste diskov (Flash/SSD) primarnih podatkovnih shramb. Rešitve so lahko edino programska oprema (VoltDB, NuoDB in Drizzle) ali podprta naprava (Clustrix, Translattice). Primeri ponudbe so Clustrix, NuoDB in Translattice (komercialni) ter VoltDB, Drizzle itd. (odprtokodni) [25]. Drugi primeri so MemSQL, JustOneDB, SQLFire in Akiban Translattice [30].

2. Novi skladiščni motorji MySQL: MySQL je OLTP podatkovna baza,

ki je v celoti skladna z ACID in je učinkovita s transakcijami, ima pa omejitve pri razširljivosti in zmogljivosti. Ena od rešitev tega problema je fragmentacija podatkov, kar lahko privede do problemov s konsistentnostjo in konsistenco, ki je na ravni aplikacije zapletena za obravnavo. Ko so podatki porazdeljeni med strežniki, je treba vse kopije ohraniti konsistentno, kjer lahko pride do zapletov pri ravnanju s transakcijami. Zato so bili razviti nadomestni shranjevalni motorji za reševanje problemov pri razširljivosti in zmogljivosti [30]. MySQL je del svežnja LAMP in se v veliki meri uporablja v OLTP. Za premagovanje težav razširljivosti MySQL se razvija set za shranjevanje motorjev. Dober del je uporaba vmesnika MySQL, slaba stran pa je, da migracija podatkov iz drugih podatkovnih baz (vključno s starim MySQL) ni podprta. Primeri ponudbe so Xeround, GenieDB in TokuTek (komercialni) ter Akiban, MySQL NDB Cluster in drugi v odprti kodi [25]. MySQL NDB Cluster je priljubljen shranjevalni motor, ki se pogosto uporablja še danes [30].

3.Transparentno grozdenje: te rešitve ohranjajo OLTP podatkovne baze v svoji prvotni obliki, zagotavljajo pa tudi plug funkcijo k pregledu grozda, da se zagotovi razširljivost [25]. Nekatere rešitve podatkovnih fragmentov so pregledno obravnavanje vprašanja nadgradljivosti [30]. Drugi pristop je, da zagotovi pregledne drobce za izboljšanje razširljivosti. Schooner MySQL, Continuent Tungsten in ScalArc sledijo prejšnjemu pristopu, medtem ko ScaleBase in dbShards sledijo slednjemu. Oba pristopa lahko ponovno uporabljata obstoječe «skillsets» in ekosistem ter se izogneta potrebi, da se ponovno napiše kodo ali storitve kakršne koli migracije podatkov. Primeri ponudbe so ScalArc, Schooner MySQL, dbShards in ScaleBase (komercialni) ter Continuent Tungsten (odprtokodni) [25].

4. Podatkovna baza kot storitev

Tradicionalni DBMS niso primerni za zagon v okolju oblaka. Torej se podatkovne baze kot storitve pojavijo zaradi reševanja tega vprašanja in da se prilagodi DBMS tako, da se lahko izvaja na javnem ali zasebnem oblaku. To so podatkovne baze kot storitve v oblaku, ki so razporejene z uporabo

virtualnih strojev. Baze NuoDB, MySQL, GaianDB in Oracle so lahko razporejene v oblaku. Drugi podatkovni model za podatkovne baze v oblaku so podatkovne baze kot storitev. Te rešitve zagotavljajo programsko opremo za podatkovne baze in fizično shranjevanje s smiselno škatlo nastavitev. Za zagon takšne storitve ni potrebna namestitev ali nastavitev, ki je na zahtevo, plačilo po uporabi storitev. Ponudnik storitev je v celoti odgovoren za ohranjanje storitev, nadgradnjo in upravljanje podatkovne baz. Na primer, Microsoft SQL Azure, StormDB, ClearDB in EnterpriseDB pripadajo kategoriji podatkovnih baz kot storitev [30].

4.4 New SQL podatkovne baze

4.4.1 NuoDB

Podjetje je bilo ustanovljeno leta 2008 kot NimbusDB in je spremenilo svoje ime v NuoDB leta 2011. Med soustanovitelji družbe so Barry S. Morris, CEO in Jim Starkey. NuoDB ima sedež v Cambridgeu, Massachusetts [32].

NuoDB je porazdeljena podatkovna baza, ki ponuja bogato SQL implementacijo in prave ACID transakcije. Zasnovana je za sodobni podatkovni center in kot horizontalna skalabilnost podatkovne baze v oblaku. NuoDB je NewSQL rešitev, ki poenostavi uvajanje aplikacij [31] in prodaja NewSQL podatkovno bazo, ki deluje v oblaku. Deluje lahko tako za enega samega prodajalca v oblaku nastavitev kakor tudi za več prodajalcev v oblaku nastavitev [32]. Ponuja pričakovano visoko razpoložljivost, vroče nadgradnje, redundanco podatkov in zmogljivost na nesreče in okrevanje. Eden izmed zasukov NuoDB je poudarek na uvajanje oblakov z vgrajeno podporo za več najemnih pogodb. To je obetaven izdelek, sicer eden izmed mlajših NewSQL prodajalcev, ki se mora še izkazati z veliko kupci [29].

4.4.2 ClustrixDB

ClustrixDB je porazdeljena SQL podatkovna baza, zgrajena za obsežne in hitro rastoče aplikacije. To je horizontalno skalabilna NewSQL podatkovna baza za oblak. ClustrixDB nedvomno omogoča realno časovne analitike na žive operativne podatke z masivnim paralelnim procesiranjem [35]. Clustrix obljublja spletno skalabilnost s SQL. Clustrix je porazdeljeni relacijski DBMS, ki podpira samodejno fragmentacijo in replikacijo. Clustrix preprodaja zelo distribuirano skalabilnost, ki je odporna na napake, ne da bi opustili SQL ali ACID uspešnost transakcij. Računi podjetij kupca Twoo.com uvaja 21-vozlišče kot «največjo svetovno namestitev horizontalne skalabilnosti SQL». Vsi NewSQL prodajalci obljubljajo skladnost SQL z lažjo operacijo (od prvotnih podatkovnih baz) na visoki ravni. Izziv za Clustrix in konkurente, kot so MemSQL, NuoDB in VoltDB, bo razlikovanje in hitro naraščanje uspešne visoke ravni zmag strank [29].

4.4.3 MemSQL

MemSQL je realno časovna analitična platforma, ki pomaga podjetjem, da hitro poizvedujejo po veliki količini podatkov in prilagajanju na spreminjajoče se poslovne razmere [35]. MemSQL združuje v pomnilniku zmogljivosti s SQL sklicevanjem. Ima visoko raven in v pomnilniku vrstično shrambo z nedavno dodano nosilno analitično zmogljivostjo. Kot pove že ime, se MemSQL loči od visoke ravni NoSQL možnosti s svojo kombinacijo v pomnilniku, ACID skladnimi transakcijskimi zmogljivostmi in združljivostmi SQL. Kot dodajanje svojega relacijskega vmesnika v pomnilniku podatkov vrste ima MemSQL pred kratkim dodano stisnjeno nosilno shrambo, ki podpira utrip in možnost diska za shranjevanje in globoko zgodovinsko analizo. Tekmovalec je starejši VoltDB (in druge vrste NewSQL podatkovnih baz), tako da je na dirki za NewSQL prevlado in sprejetje strank [29].

4.4.4 SpliceMachine

SpliceMachine postavlja transakcije SQL na Hadoop. SQL na Hadoop RDBMS podpira transakcije in analitike. Obstaja veliko možnosti za SQL na Hadoop, a edinstvena trditev zagona SpliceMachine je, da lahko vodi tako transakcijske aplikacije kot tudi podporne analitike na vrhu Hadoopa. Stranka Harte Hanks nam pove, da je tekmovanje v teku kup aplikacij, ki so bile zasnovane, da delujejo na običajnih podatkovnih bazah, vključno z integracijo podatkov programske opreme IBM Unica, Cognos BI in Ab initio. To je zelo mlado podjetje z zelo kratkim seznamom omembe vrednih kupcev, a je zaganjanje transakcijskih aplikacij na Hadoop edinstven predlog, ki izstopa iz množice NewSQL [29].

4.4.5 TransLattice

TransLattice obsega PostgreSQL. Ima porazdeljeni relacijski DBMS z oblakom in možnostjo uvajanja naprav. TransLattice začenja z visokim obsegom, porazdeljene variacije na PostgreSQL se imenujejo podatkovna baza TransLattice Elastic, ki se razvije v prostorih, na aparatu ali pa na več oblakih. Leta 2013 je družba pridobila StormDB, ki je bil tudi razširitev PostgreSQL in uporablja svojo intelektualno lastnino, da začne Postgres-XL prilagodljivo, masovno, vzporedno analitično podatkovno bazo. Ideja je ponuditi številnim uporabnikom Postgree poznan način k horizontalni skalabilnosti za veliko podatkov OLTP in analiz. Zelo kratek seznam javnih referenčnih kupcev kaže, da TransLattice, tako kot mnogi NewSQL prodajalci, potrebuje več tržnih dokazov točk [29].

4.4.6 VoltDB

VoltDB je v pomnilniku podatkovne baze zelo hitra in z neverjetno visoko bralno ter pisalno hitrostjo. Ta NewSQL podatkovna baza podpira JSON na ravni dogodka transakcije [35]. VoltDB se ukvarja s hitrim pretakanjem podatkov, uporablja ACID transakcije in SQL skladnost v spominu rela-

cijske podatkovne baze. VoltDB ima velik obseg in obdela visoke hitrosti transakcij, zahvaljujoč zelo porazdeljeni in pomnilniški arhitekturi. Odprtokodni GNU-licensed DBMS, ki si ga je delno izmislil soustanovitelj DBMS imenitnež dr. Michael Stonebraker, zmore hitro pretakanje podatkov med mobilnimi oglaševalskimi mrežami in igralniškimi podjetji. Analitične nadgradnje, napovedane v začetku leta 2014, ki jih uvaja VoltDB 4.0, so prinesle povečanje pretovora poizvedbe in podporo tako sočasnim uporabnikom kakor tudi globlji SQL zmogljivosti na področjih, kot so časovne serije analiz. Večina neposrednih konkurentov VoltDB so MemSQL, NuoDB in Clustrix, vendar je z Oraclovim in Microsoftovim prebijanjem v pomnilniku za obdelavo transakcij v letošnjem letu največja konkurenca lahko poznavanje teh uveljavljenih sistemov za upravljanje podatkovnih baz in upanje strank za njihov vpliv [29].

4.4.7 Google Spanner

Spanner je Googlova globalno porazdeljena NewSQL podatkovna baza, ki je naslednik BigTable. Google opisuje Spanner kot nečist relacijski sistem, ker mora imeti vsaka tabela primarni ključ stolpca. Ker je pomanjkanje transakcij v BigTable privedlo do pogostih pritožb uporabnikov, je Google naredil porazdeljene transakcije osrednjega pomena za Spannerjev načrt. Na podlagi izkušenj z BigTable Google trdi, da je bolje, da se programerji aplikacij ukvarjajo s problemi uspešnosti, ker se zaradi prekomerne rabe transakcij pojavijo ozka grla, namesto da bi vedno kodirali okoli pomanjkanja transakcij. Sistem Google F1 SQL za upravljanje podatkovnih baz (DBMS) je zgrajen na njem in nadomešča Googlov običajni MySQL. Opisan kot NewSQL platforma se uporablja interno znotraj njihove infrastrukture, in sicer kot del platforme Google. Uporablja algoritem Paxos kot del svojega delovanja za izmenjavo podatkov med več sto podatkovnim centri. To naredi težko uporabo strojne opreme podpore časovne sinhronizacije z uporabo GPS-a in atomske ure, in sicer zato, da se zagotovi globalna konsistentnost [33].

Poglavje 5

Primerjava relacijskih, NoSQL in NewSQL podatkovnih baz

5.1 Splošna primerjava

Relacijske podatkovne baze so bile vedno odlikovane z načelom ACID (atomarnost, konsistentnost, izolacija in trajnost), ki zagotavlja, da je integriteta(celovitost) podatkov ohranjena za vsako ceno. SQL je postal de facto standard za obdelavo podatkov, saj združuje elemente, kot so opredelitev podatkov, manipulacija s podatki, in poizvedovanje po podatkih, vse pod eno streho [34]. RDBMS zagotavlja uspešnost vrstnega reda tisoč transakcij na sekundo. Med tem, ko je primerna za tradicionalne poslovne aplikacije, zaostaja za čedalje večjim številom scenarijev v uporabi spletne obdelave transakcij (OLTP), kot so oglaševanje v realnem času, odkrivanje prevar in analiz tveganj, med drugim tudi, da se lahko ustvari skoraj milijon transakcij na sekundo [40].

NoSQL sistemi za upravljanje podatkovnih baz shranjujejo podatke v različnih formatih. Glavni med njimi so dokumentne shrambe, grafične podatkovne baze in ključ-vrednost shrambe. Večina NoSQL izdelkov opusti ACID učinkovitost, da se doseže fleksibilnosti za shranjevanje podatkov s sposob-

nostjo pomanjšanja in podpore pretoku visoke zmogljivosti. Odstranjujejo trde omejitve, kot so tabele vrstic shramb in strogih definicij podatkov, in jih predvidijo za obseg s porazdeljeno arhitekturo, ki podpirajo prepustnost visoke zmogljivosti [34]. NoSQL podatkovne baze niso relacijske, ne uporabljajo poizvedovalnega jezika SQL, imajo horizontalno skalabilnost in so brez sheme. NoSQL podatkovne baze sledijo CAP teoremu in so v skladu z načelom BASE.

Najnovejši udeleženci v areni podatkovnih baz NewSQL, ohranijo tako SQL in ACID, vendar pri premagovanju omejitev RDBMS z uporabo porazdeljene računalniške arhitekture [40]. NewSQL podatkovne baze so relacijske, imajo horizontalno skalabilnost in shemo.

Navsezadnje bi morala biti narava podatkov merilo, ki narekuje izbiro tehnologije podatkovne baze. To pomeni, da bi morali transakcijski podatki, v katerih se zahteva strogo upoštevanje celovitosti in doslednost podatkov, uporabiti RDBMS in NewSQL namesto NoSQL, medtem ko sta lahko NoSQL in NewSQL boljši možnosti, ko gre za horizontalno skalabilnost [40].

Lastnosti	StariSQL	NoSQL	NewSQL
Relacijska	Da	Ne	Da
SQL	Da	Ne	Da
Transakcije ACID	Da	Ne	Da
Horizontalna skalabilnost	Ne	Da	Da
Učinkovitost/velik obseg	Ne	Da	Da
Brez sheme	Ne	Da	Ne

Tabela 5.1: Splošna primerjava podatkovnih baz [23]

Lastnosti	RDBMS	NoSQL	NewSQL
SQL	Podpiran	Ni podpiran	Podpiran
Odvisnost naprave	Ena naprava	Več naprav\ razdeljen	Več naprav\ razdeljen
DBMS tip	Relacijski	Nerelacijski	Relacijski
Shema	Tabela	Ključ-vrednost stolpčna shramba dokumentna shramba	Oboje
Skladišče	Na disk + cache(shramba)	Na disk + cache(shramba)	Na disk + cache(shramba)
Podpora lastnosti	ACID	CAP skozi BASE	ACID
Horizontalna skalabilnost	Ni podpirana	Podpirana	Podpirana
Kompleksnost poizvedb	Nizka	Visoka	Zelo visoka
Skrb varnosti	Zelo visoka	Nizka	Nizka
Velik obseg	Manj učinkovit	V celoti podpira	V celoti podpira
OLTP	Ni v celoti podpiran	Podpiran	V celoti podpira
Podpora za oblak	Ni v celoti podpiran	Podpiran	V celoti podpira

Tabela 5.2: Primerjave med podatkovnimi bazami [35]

5.2 Izbira prave podatkovne baze

Kdaj bi izbrali NoSQL ali NewSQL podatkovno bazo namesto relacijske podatkovne baze? Navedli smo deset lastnosti, ki vam bodo pomagale pri izbiri prave podatkovne baze.

Kako izbrati?

Za reševanje izbire tipov podatkovnih baz začnimo z naslednjimi vprašanji:

V kolikšni meri se zanašati na podatke v smislu shranjevanja, obdelave in analize? Stopnja odvisnosti v vsakem primeru lahko zelo vpliva na izbor podatkovne baze. Razvoj aplikacij, na primer, ni močno usmerjen v podatke, medtem ko analiza podatkov je. Nekatera podjetja se vrtijo okoli podatkov, druga pa uporabljajo podatke, da dopolnijo svoja ključna območja usmeritve [34].

Kako pomembni so skalabilnost, fleksibilnost in vidiki uspešnosti SUPB?

Kakšna je raven vlaganj v prvotne tehnologije? Če ste že investirali v DBMS, ali ste pripravljeni prevzeti stroške prehoda na novejšo tehnologijo (in morda soočenje s funkcijami nezdružljivosti ali upravljanje in vrzeli znanja programiranja med osebjem) [34]?

Transakcijski podatki(strogo upoštevanje celovitosti in doslednosti podatkov)

Narava naših podatkov navsezadnje narekuje izbiro tehnologije podatkovnih baz. Na primer, transakcijski podatki, ki zahtevajo strogo upoštevanje celovitosti in doslednosti podatkov podpirajo uporabo RDBMS in NewSQL in ne NoSQL [34].

Nestanovitni(spremenljivi) podatki

Za nestanovitne podatke na drugi strani pa je značilno spreminjanje objektnih modelov in podatkovne strukture formatov, ki zahtevajo fleksibilnost, omogočajo pa tudi, da je NoSQL najboljša izbira, NewSQL pa sledi v manjšem obsegu. RDBMS se lahko s svojo togostjo oblikovanja sheme izkaže za zelo drago, ko se ukvarjamo s takšnimi podatki [34].

Skalabilnost

Ko gre za skalabilnost, imajo podjetja rajši horizontalno skalabilnost, arhitekturni pristop, ki je stroškovno učinkovit in ki zagotavlja boljšo toleranco napak. Skaliranje pri RDBMS podatkovnih bazah vključuje distribucijo podatkov v več vozlišč, kar lahko naredi vzdrževanje podatkov kaotično. NoSQL in NewSQL izdelki niso omejeni s takimi omejitvami, zato jih je veliko lažje vzdrževati, ko gre za horizontalno skalabilnost [34].

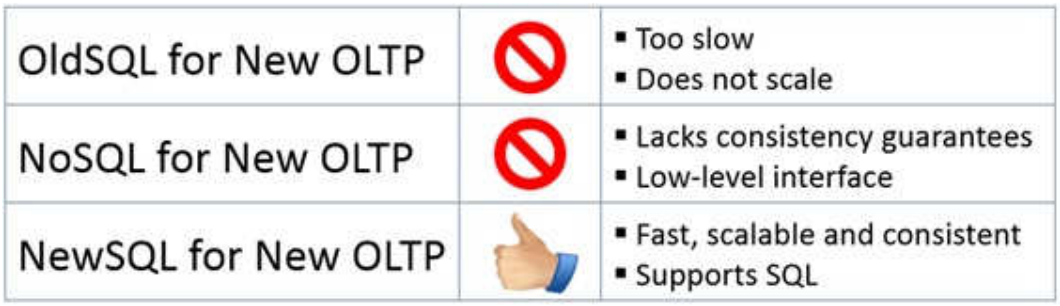
Uspešnost

Pri odločanju o uspešnosti so odločilni dejavniki pri izbiri podatkovne baze osnovne oblike podatkov in število operacij, ki se izvaja. To je nemogoče reči ali bo en tip podatkovne baze hitrejši kot drugi brez konteksta, a literatura in merila kažejo, da so NewSQL izdelki prekosili NoSQL in SQL na

področjih, kot so elastične razširljivosti in obdelava transakcij na sekundo. To je pomembno za e-poslovanje podjetij, ki ravnajo spremljanje naročil ali upravljanje zalog, in za spletne igre na srečo ter za podjetja, ki ravnajo z večplastnimi milijoni transakcij na sekundo [34].

Tehnologija

Tehnologije podatkovnih baz se hitro prilagaja dohajanju eksplozije količine podatkov, naraščajoči raznolikosti podatkov in povečanju njihove hitrosti. Vrst RDBMS, NoSQL in NewSQL možnosti je veliko, zato je pomembno, da se zberejo podrobne zahteve za praktično vsak vidik porabe podatkov, preden se odločimo za eno. Bodo NoSQL ali NewSQL pregnali RDBMS kot industrijski standard? Predelava te poizvedbe je v teku [34].



Slika 5.1: Podatkovne baze na novem OLTP

Novi OLTP: stari SQL je prepočasen in ne omogoča skaliranja. NoSQL ne zagotavlja konsistentnosti in ima nizek nivo vmesnika. NewSQL je hiter, skalabilen in konsistenten, podpira tudi SQL [36].

Lastnosti	RDBMS	NoSQL	NewSQL
ACID skladnost (podatki, celovitosti transakcij)	Da	Ne	Da
OLAP/OLTP	Da	Ne	Da
Analiza podatkov (seštevek, transformacija, itd)	Da	Ne	Da
Togost sheme(stroga preslikava modela)	Da	Ne	Mogoče
Fleksibilnost formata podatkov	Ne	Da	Mogoče
Porazdeljeno računalništvo	Da	Da	Da
Vertikalna/horizontalna skalabilnost	Da	Da	Da
Uspešnost z naraščanjem podatkov	Hiter	Hiter	Zelo hiter
Obremenitev zmogljivosti	Velika	Zmerna	Minimalen
Priljubljenost/podpora skupnosti	Velika	Narašča	Počasi narašča

Tabela 5.3: Deset lastnosti primerjav med podatkovnimi bazami [34]

Poglavje 6

Priprava na testiranje

Za primerjavo oz performančno analizo med MySQL, AppacheCassandra in NuoDB je treba podatkovne baze primerjati s podatki.

6.1 Testno okolje

Pred samim začetkom testiranja in primerjavo rezultatov med relacijsko, NoSQL in NewSQL podatkovno bazo je treba namestiti vsako podatkovno bazo posebej in pripraviti testne skripte. Delali smo v okolju Windows 7 v virtualki VMware Workstation.

6.1.1 MySQL

Začnemo z namestitvijo MySQL strežnika 5.529(32 biten). Instalacija MySQL strežnika je enostavna, edino kar je treba paziti je, da pri namestitvi izberemo typical. Sledi konfiguracija MySQL strežnika, izberemo detailed configuration, developer machine, Multifuctional Database, Decision Support (DSS)/OLAP in Best Support For Multilingualism. Pri konfiguraciji kot skrbniki imamo uporabniško ime «root» in izberemo skrbniško(root) geslo. Namestimo tudi odjemalca MySQL Workbench 5.2.45. Instalacija je enostavna, sledimo samo navodilom. Naredimo novo povezavo in notri vnesemo uporabniško ime ter geslo. Shemo naredimo s sledečimi ukazi.

CREATE SCHEMA travian;
CREATE USER 'root' IDENTIFIED BY 'skrivnogeslo';
GRANT ALL PRIVILEGES ON travian.* TO root WITH GRANT OPTION;
COMMIT;

Za testiranje MySQL podatkovne baze namestimo
mysql-connector-java-5.1.34-bin.jar
Java JDK.
Podatkovno bazo MySQL povežemo tako:
String mySqlUrl = "jdbc:mysql://127.0.0.1:3306/travian";
Class.forName("com.mysql.jdbc.Driver");
con = DriverManager.getConnection(mySqlUrl, "root", "skrivnogeslo");

6.1.2 Cassandra

Za delovanje Cassandre je treba najprej inštalirati Javo. Nastaviti je treba spremenljivke okolja. Ustvarimo novo uporabniško spremenljivko JAVA_HOME in notri damo pot do jre knjižnice C:\Program Files\Java\jre7. Pri tem je treba paziti, da damo jre in ne jdk. Nadaljujemo z instalacijo Cassandre. S spletne strani http://cassandra.apache.org/download/ povlečemo Cassandro in jo razširimo v direktoriju. Sledi konfiguracija datoteke cassandra.yaml, ki se nahaja v C:\Cassandra\conf. Najti je treba data_file_directories in /var/lib/cassandra/data spremenimo v C:/Cassandra/data. Nato je treba najti commitlog_directory in /var/lib/cassandra/commitlog spremenimo v C:/Cassandra/commitlog. Najti je treba saved_caches_directory in /var/lib/-cassandra/saved_caches spremenimo v C:/Cassandra/saved_caches.

Za povezavo s Cassandro smo uporabili program Datastax DevCenter. Za njegovo delovanje je treba predhodno zagnati Cassandro v Command Prompt na Windowsih. Najprej ustvarimo novo povezavo s File→New Connection. Shemo oz keyspace naredimo s:

CREATE KEYSPACE mykeyspace
WITH REPLICATION = $\{'class' : 'SimpleStrategy','replication\_factor' : 1\}$;
USE mykeyspace;

Uporabljamo USE mykeyspace ali pa nastavimo keyspace v programu Datastax DevCenter, kjer izberemo tudi povezavo.

Za testiranje Cassandrine podatkovne baze smo namestili naslednje gonilnike, pri čemer je treba paziti, da so naslednji jari iste verzije, to so cassandra jar(jdbc, thrift, clienutil), slf4j(api, jdk, log4j12, simple).
cassandra-jdbc-1.2.5.jar
cassandra-thrift-1.2.5.jar
cassandra-clientutil-1.2.5.jar
slf4j-api-1.7.2.jar
slf4j-jdk14-1.7.2.jar
slf4j-log4j12-1.7.2.jar
slf4j-simple-1.7.2.jar
libthrift-0.9.1.jar
cassandra-all-2.0.7.jar
cassandra-driver-core-2.0.2.jar
cassandra-driver-dse-2.0.2.jar
hector-core-2.0-0.jar
selenium-server-standalone-2.21.0.jar

Podatkovno bazo Cassandra povežemo tako:
Class.forName(”org.apache.cassandra.cql.jdbc.CassandraDriver”);
con = DriverManager.getConnection(”jdbc:cassandra://localhost:9160/travian”);

6.1.3 NuoDB

Najprej gremo na uradno stran NuoDB, kjer se registriramo, da lahko povlečemo nuodb_2.0.4.20. Zaženemo namestitveno datoteko nuodb_2.0.4.20 in

odpre se čarovnik za namestitev. Namestitev je enostavna, slediti je treba samo navodilom. Podatkovno bazo smo ustvarili tako, da smo dali CREATE database in določil ime. Localhost je določen kot 48004. Nastavili smo pot, kjer naj bi se shranjevala podatkovna baza, recimo C:\Program Files\NuoDB\samples\travian Database. Določil smo si uporabniško ime(use) in password.

Gonilnike dobimo ob namestitvi NuoDB podatkovne baze C:\Program Files\-NuoDB\jar. Za testiranje te baze namestimo naslednje gonilnike.
nuodbjdbc.jar
embedded-api.jar
nuoagent.jar
nuodb-hibernate.jar
nuodbmanager.jar
nuodb-rest-api.jar
nuodbtraymonitor.jar
nuodbwebconsole.jar

Podatkovno bazo NuoDB povežemo tako
Class.forName(”com.nuodb.jdbc.Driver”); // Create the connection
Connection conn
=DriverManager.getConnection(”jdbc:com.nuodb://localhost:48004/travianDatab-ase? schema=travian”, ”marko_m”, ”mmgeslo”);
//Create a Statement class to execute the SQL statement
Statement stmt = conn.createStatement();

6.2 Priprava testnih podatkov

Ob predpostavki, da se Cassandra in NuoDB dobro izkažeta pri veliki količini podatkov, smo na spletu poiskali podatkovno bazo, ki ima zapisane zapise o spletni igri travian. Podatkovna baza travian.sql vsebuje podatke o

- plemenih
- aliansah
- igralcih
- naseljih

Tabela pleme vsebuje 5 zapisov, tabela aliansa 1.368 zapisov , tabela igralec 12.419 zapisov in tabela naselje vsebuje 20.967 zapisov. Vseh zapisov je 34.759, kar znaša obsežno množico podatkov. Ustvarili smo tabele pleme, aliansa, igralec in naselje ter se potem lotili vstavljanja podatkov. Za podatkovni bazi Cassandra in NuoDB smo morali spremeniti podatke pri ustvarjanju tabel. Spodaj je naveden primer tabele igralec pri MySQL.

```
CREATE TABLE igralec(
  pid integer default 0 NOT NULL,
  player varchar(100) default '' NOT NULL,
  tid integer default 0 NOT NULL,
  aid integer default 0 NOT NULL,
  PRIMARY KEY (pid),
) DEFAULT CHARSET=utf8 COLLATE=utf8_slovenian_ci;
```

Spodaj navajamo primer tabele igralec pri Cassandri.

```
CREATE TABLE igralec(
  pid int,
  player varchar,
  tid int,
  aid int,
  primary key(pid)
);
```

Spodaj sledi primer tabele igralec pri NuoDB.

```
CREATE TABLE igralec(
  pid integer primary key,
  player varchar(100),
  tid integer,
  aid integer
)
```

Pri vseh podatkovnih bazah ime spremenljivke ostane isto. Sintakse so si sicer podobne, le pri My SQL in NuoDB so podatkovni tipi integer, varchar, pri Cassandra pa int, varchar.

6.3 Programski jeziki za testiranje

Preden smo začeli s testiranjem učinkovitosti, smo se morali odločiti za programski jezik, ki ga bomo uporabljali pri testiranju. Odločali smo se med več programskimi jeziki za testiranje, in sicer med C++, Java, PHP. Odločili smo se za programski jezik Java, ker smo se z njim že ukvarjali in sicer pri povezavi podatkovnih baz. To je tudi programski jezik, ki ga najbolje obvladamo.

6.4 Priprava testnih skript

Napisali smo skripto v programskem jeziku Java, ki skrbi za dostop do podatkovne baze. Poleg dostopa do podatkovne baze, smo v skripto dodali kodo za merjenje časa izvajanja, in to pri vsaki podatkovni bazi posebej. Napisali smo skripto, ki pridobi vse podatke iz tabele igralec ali pa določeno število podatkov iz tabele igralec.

Poglavje 7

Testiranje na podatkih

7.1 Uporabniški scenarij

Testiranje podatkov v podatkovni bazi temelji na realnih dogodkih. Testiranje smo opravili na različnih scenarijih. Najprej je bilo treba ustvariti tabele. Ker se meritve med testiranjem spreminjajo, smo vsaj trikrat pognali teste za posamezni uporabniški scenarij in dobljene meritve delili s srednjo vrednostjo meritev. Uporabniški scenariji, ki smo jih testirali, so naslednji:

- kreiranje tabele,
- vstavljanje podatkov,
- branje podatkov,
- brisanje podatkov.

Pri scenariju kreiranja tabele smo merili čas izvajanja za vse tabele. Vsi časi so podani v sekundah. Testirali smo v okolju Windows 7 v virtualki VMware Workstation in zaradi tega testi niso popolnoma natančni.

V tabeli so prikazani časi izvajanja pri kreiranju tabele, podani v sekundah. Na podlagi tabele smo narisali graf, ki nazorno prikaže čas izvajanja pri posamezni podatkovni bazi.

	MySQL	Cassandra	NuoDB
Create table pleme	0,062	1,427	0,125
Create table aliansa	0,47	1,850	0,312
Create table igralec	0,015	1,671	0,140
Create table naselje	0,453	1,615	0,296

Tabela 7.1: Merjenje časa pri kreiranju tabele

Slika 7.1: Graf merjenja časa pri kreiranju tabele

Kot vidimo, je pri kreiranju tabele najhitrejša MySQL podatkovna baza, sledi NuoDB, najslabše pa se izkaže Cassandra.

7.1.1 Scenarij branja podatkov

Pri tem scenariju preberemo določeno število podatkov iz določene tabele. Pridobiti želimo podatke o n-igralcih. Pri tem primeru je treba, recimo, vstaviti 10.000 podatkov in nato spreminjati ukaz SELECT ter določiti omejitev z ukazom LIMIT za n-podatkov.

V tabeli so prikazani časi izvajanja operacije branja podatkov, glede na število testnih primerov. Vsi časi so podani v sekundah. Na podlagi tabele

smo narisali graf, ki nazorno prikaže čas izvajanja pri posamezni podatkovni bazi.

Primer ukaza je SELECT * FROM igralec LIMIT 100;

Testni primeri	MySQL	Cassandra	NuoDB
1	0,015	0,296	0,072
5	0,015	0,110	0,039
10	0,016	0,125	0,0385
50	0,0235	0,1565	0,0235
100	0,0385	0,2805	0,0235
500	0,422	0,952	2,938
1.000	0,5155	1,281	2,033
5.000	1,5785	2,159	2,165
10.000	2,394	4,1935	4,7257

Tabela 7.2: Merjenje časa pri branja podatkov

Slika 7.2: Graf merjenja časa pri branja podatkov

Pri tem vidimo, da je najhitrejša podatkovna baza MySQL, sledita Cassandra in NuoDb. Časi pri Cassandri in NuoDB so si malce podobni, pri veliki količini podatkov pa je Cassandra hitrejša pri branju podatkov kot NuoDB.

7.1.2 Scenarij vstavljanja podatkov

Pri tem scenariju smo meril čas izvajanja pri vstavljanju podatkov v tabelo. Za vsak vnos smo merili čas izvajanja in sicer za 1, 5, 10, 50, 100, 500, 1000, 5000 in 10000 vnosov v tabelo.

V tabeli so prikazani časi izvajanja operacije pri vstavljanju podatkov, glede na število testnih primerov. Vsi časi so podani v sekundah. Na podlagi tabele smo narisali graf, ki nazorno prikaže čas izvajanja pri posamezni podatkovni bazi.

Primer ukaza je
INSERT INTO ‘igralec‘ VALUES (1,’Multihunter’,1,1); pri MySQL.
INSERT INTO igralec(pid, player, tid, aid) VALUES (1,’Multihunter’,1,1); pri Cassandra
INSERT INTO igralec VALUES (1,’Multihunter’,1,1); pri NuoDB

Testni primeri	MySQL	Cassandra	NuoDB
1	0,16	0,783	0,203
5	0,47	0,724	0,093
10	0,109	0,8585	0,016
50	0,094	2,075	0,187
100	0,140	1,904	0,031
500	0,249	9,7595	0,6945
1.000	0,374	17,818	0,842
5.000	0,6165	79,5685	1,716
10.000	0,686	162,366	4,227

Tabela 7.3: Merjenje časa pri vstavljanja podatkov

Slika 7.3: Graf merjenja časa pr vstavljanja podatkov

Pri tem scenariju se najboljše izkaže MySQL podatkovna baza, sledi ji NuoDB, ki je malce počasnejši. Najpočasnejša podatkovna baza pri vstavljanju podatkov je Cassandra.

7.1.3 Scenarij brisanja podatkov

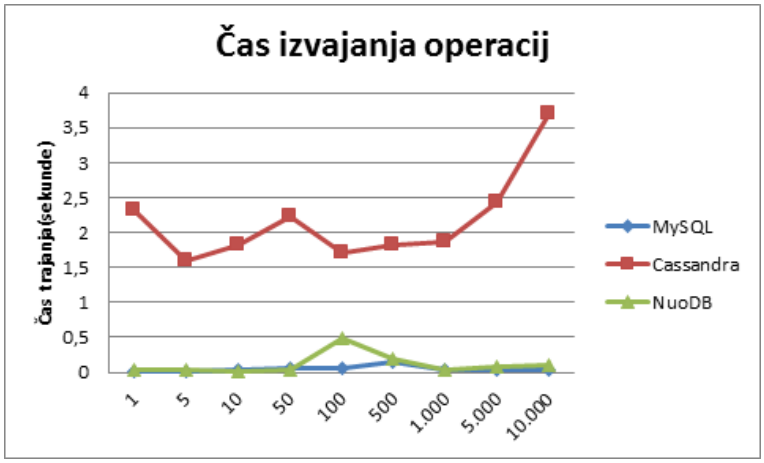
Pri scenariju brisanja podatkov smo merili čas izvajanja, ki je potreben pri brisanju tabele. Pri vsakem določenem vnosu podatkov v tabelo smo izbrisali določeno število vnosov v tabeli.

Primer ukaza je DROP TABLE igralec, ki je za vse podatkovne baze enak.

V tabeli so prikazani časi izvajanja operacije pri brisanju podatkov, glede na število testnih primerov. Vsi časi so podani v sekundah. Na podlagi tabele smo narisali graf, ki nazorno prikaže čas izvajanja pri posamezni podatkovni bazi.

Testni primeri	MySQL	Cassandra	NuoDB
1	0,015	2,3155	0,031
5	0,016	1,605	0,032
10	0,032	1,819	0,016
50	0,062	2,229	0,032
100	0,063	1,710	0,483
500	0,156	1,828	0,203
1.000	0,047	1,857	0,032
5.000	0,032	2,428	0,078
10.000	0,047	3,691	0,109

Tabela 7.4: Merjenje časa pri brisanju podatkov



Slika 7.4: Graf merjenja časa pri brisanja podatkov

Tudi pri brisanju podatkov se najbolj izkaže MySQL podatkovna baza, sledi NuoDB, ki je malce počasnejši. Cassandra pa se je izkazala za najpočasnejšo podatkovno bazo pri brisanju podatkov.

Poglavje 8

Sklepne ugotovitve

8.1 Sklepne ugotovitve

Primerjanje relacijskih, NoSQL in NewSQL podatkovnih baz je zahtevno opravilo. Pri testiranju smo se odločili testirati MySQL, Appache Cassandro in NuoDB, ker je vsaka podatkovna baza predstavnik ene izmed podatkovnih baz in ker bi bilo zelo težko testirati vse predstavnike teh baz. Videli smo, da je običajno Cassandra veliko počasnejša, kot sta MySQL in NuoDB, razen pri branju podatkov s podatkovne baze, kjer je Cassandra hitrejša kot NuoDB. Testirali smo v operacijskem sistemu Windows 7 v virtualki VMware Workstation, zaradi česar testiranje ni popolnoma realno.

Relacijske podatkovne baze zagotavljajo zanesljiv način izvajanja transakcij (ACID), imajo standardiziran poizvedovalni jezik in podatkovno shemo. Relacijske podatkovne baze niso najbolj primerne, ko imamo veliko količino podatkov.

Zakaj je dober NoSQL?
NoSQL je primeren za netransakcijske sisteme in za enozapisne, komutativne operacije. Omogoča upravljanje podatkov brez definirane sheme. Ne potrebujemo enovitega povpraševalnega jezika (programske kode, CQL, UnQL (po

zgledu SQL, nestandardno, nekompatibilno)). NoSQL ima zagotovljeno varnost pred odpovedjo in je optimiziran za spletne storitve. NoSQL v nasprotju z NewSQL ni primeren za sodobne OLTP sisteme. NoSQL podatkovna baza je bolj primerna za kompleksne poizvedbe in tudi hitrejša kot relacijske podatkovne baze.

Kdaj se je dobro odločiti za NewSQL podatkovno bazo?

- uspešnost z naraščanjem podatkov je zelo hitra
- minimalna obremenitev zmogljivosti
- visoka kompleksnost poizvedb
- dober je za nov OLTP
- dobra podpora za oblak

NewSQL je lažji, konsistentnejši, elegantnejši in bolje definiran kot SQL. Podpiral naj bi tudi že obstoječe relacijske podatkovne baze. NewSQL je dober tudi za finančne sisteme in sisteme za procesiranje naročil. Taki sistemi torej ne morejo uporabljati NoSQL rešitev, saj jim NoSQL ne zagotavlja prepotrebnih transakcijskih in konsistenčnih zahtev.

8.2 Zaključek

V diplomskem delu smo predstavili relacijske, NoSQL in NewSQL podatkovne baze. Osredotočili smo se na novo generacijo podatkovnih baz NewSQL. Naredili smo primerjavo med podatkovnimi bazami in merili hitrost izvajanja teh baz.

Diplomsko delo bo v pomoč vsem inženirjem in administratorjem podatkovnih baz pri izbiri pravilne podatkovne baze. Tako se bodo ti lažje odločili, kateri tip podatkovne baze je bolj primeren za zeleni tip aplikacije oziroma storitve. V pomoč bo tudi kratek vodič pri instalaciji vsake podatkovne baze, in sicer za MySQL, Cassandro in NuoDB.

Diplomsko delo bo v pomoč tudi pri spoznavanju z novimi NewSQL podatkovnimi bazami. Spoznali smo definicijo NewSQL in njene glavne značilnosti, spoznali smo tudi kategorizacijo NewSQL. Predstavili smo nekaj NewSQL podatkovnih baz, kot so NuoDB, ClustrixDB, MemSQL, SpliceMachine, TransLattice, VoltDB, Google Spanner.

Menimo, da bo v prihodnosti proizvodnja NoSQL in NewSQL podatkovnih baz še bolj naraščala, ker so bile do sedaj NewSQL možnosti precej manj priljubljene kot NoSQL možnosti, deloma zato, ker so zelo nove, a tudi zato, ker relacijski pristop in fleksibilnost podatkov ni enostavno kombinirati. Menimo, da bodo NewSQL podatkovne baze pridobile priljubljenost med prodajalci podatkovnih baz.

Glavna ovira pri izdelavi diplomskega dela je bilo pomanjkanje literature na spletu. Ker je NewSQL nova generacija podatkovnih baz, smo bili omejeni na iskanje člankov na spletu, po forumih in dokumentacij posameznih podatkovnih baz. Težko smo našli kakršno koli gradivo, saj je tehnologija precej nova in se spreminja. Prav tako še ni sprejetih standardov.

Dodatek A

Testne Skripte

A.1 MySQL

```
import java.io.*;
import java.sql.*;

public class TestIgralec {
public static void main(String[] args){
Connection con = null;
  try {
   String mySqlUrl = "jdbc:mysql://127.0.0.1:3306/travian";
   Class.forName("com.mysql.jdbc.Driver");
   con = DriverManager.getConnection(mySqlUrl, "root", "skrivnogeslo");
   long start = System.currentTimeMillis();
   String query = "SELECT * FROM igralec LIMIT 10000";

   Statement stmt = con.createStatement();
   ResultSet result = stmt.executeQuery(query);
   int stVrstice=0;
   while (result.next()) {
    System.out.print(result.getString("pid"));
    System.out.print(result.getString("player"));
    System.out.print(result.getString("tid"));
    System.out.print(result.getString("aid"));
    stVrstice++;
```

```
    System.out.println("vrstica"+stVrstice);
   }
   long end = System.currentTimeMillis();
   System.out.println("cas izvajanja "+(end-start)+" ms."); //end-start
  } catch (ClassNotFoundException e) {
   e.printStackTrace();
  } catch (SQLException e) {
   e.printStackTrace();
  } finally {
   if (con != null) {
    try {
     con.close();
    } catch (SQLException e) {
     e.printStackTrace();
    }
    con = null;
  }
 }
}
}
```

A.2 Cassandra

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
public class TestIgralec {

public static void main(String[] args) throws Exception{

  Connection con = null;
  try {
   Class.forName("org.apache.cassandra.cql.jdbc.CassandraDriver");
   con = DriverManager.getConnection("jdbc:cassandra://localhost:9160/travi
   an");
```

```
   long start = System.currentTimeMillis();
   String query = "SELECT * FROM igralec LIMIT 10000";

   Statement stmt = con.createStatement();
   ResultSet result = stmt.executeQuery(query);
   int stVrstice=0;
   while (result.next()) {
    System.out.print(result.getString("pid"));
    System.out.print(result.getString("player"));
    System.out.print(result.getString("tid"));
    System.out.print(result.getString("aid"));
    stVrstice++;
    System.out.println("vrstica"+stVrstice);
   }
   long end = System.currentTimeMillis();
   System.out.println("cas izvajanja "+(end-start)+" ms."); //end-start
  } catch (ClassNotFoundException e) {
   e.printStackTrace();
  } catch (SQLException e) {
   e.printStackTrace();
  } finally {
   if (con != null) {
    try {
     con.close();
    } catch (SQLException e) {
     e.printStackTrace();
    }
    con = null;
  }
 }
}
}
```

A.3 NuoDB

```
import java.sql.Connection;
```

```
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class TestIgralec {
public static void main(String[] args) throws SQLException {
try {
    // Load the driver
    Class.forName("com.nuodb.jdbc.Driver");
    // Create the connection
    Connection conn = DriverManager.getConnection("jdbc:com.nuodb://local
    host:48004/travianDatabase?schema=travian",  "marko_m", "mmgeslo");
    //Create a Statement class to execute the SQL statement
    Statement stmt = conn.createStatement();
    //Execute the SQL statement and get the results in a ResultSet
   long start = System.currentTimeMillis();
   ResultSet result = stmt.executeQuery("SELECT * FROM igralec LIMIT 10000
    ");
     int stVrstice=0;
     while (result.next()) {
    System.out.print(result.getString("pid"));
    System.out.print(result.getString("player"));
    System.out.print(result.getString("tid"));
    System.out.print(result.getString("aid"));
    stVrstice++;
    System.out.println("vrstica"+stVrstice);
     }
     long end = System.currentTimeMillis();
System.out.println("cas izvajanja "+(end-start)+" ms."); //end-start
}
catch (ClassNotFoundException e) {
    System.out.println("Caught Class Not found Exception");
    e.printStackTrace();
}
catch (SQLException e) {
    e.printStackTrace();
}
```

```
catch (Exception e) {
    e.printStackTrace();
}
}
}
}
```

Literatura

[1] (2014)Wikipedia "Podatkovna zbirka" Dostopno na http://sl.wikipedia.org/wiki/Podatkovna_zbirka

[2] (2014) Spletna učilnica FRI "PB_sl_2009_2010_v02_CB_2_na_stran" Dostopno na https://ucilnica.fri.uni-lj.si/

[3] (2014) Spletna učilnica FRI "PB2013" Dostopno na https://ucilnica.fri.uni-lj.si/

[4] (2014) Wikipedia "Relacijska podatkovna baza" Dostopno na http://en.wikipedia.org/wiki/Relational_database

[5] (2014) Relational Databases for Dummies Dostopno na http://code.tutsplus.com/tutorials/relational-databases-for-dummies–net-30244

[6] (2014) technopedia "What is relational database" Dostopno na http://www.techopedia.com/definition/1234/relational-database-rdb

[7] (2014) Spletna učilnica FRI "PB2b - prevadanja 2011-2012_v1.7" Dostopno na https://ucilnica.fri.uni-lj.si/

[8] (2014) Wikipedia "ACID" Dostopno na http://en.wikipedia.org/wiki/ACID

[9] (2014) Wikipedia "SQL" Dostopno na http://sl.wikipedia.org/wiki/SQL

[10] (2014) Spletna učilnica FRI "NoSQL podatkovne baze Kratek pregled" Dostopno na https://ucilnica.fri.uni-lj.si/

[11] (2014) Spletna učilnica FRI "SODOBNE NERELACIJSKE PODATKOVNE BAZE (NOSQL)" Dostopno na https://ucilnica.fri.uni-lj.si/

[12] (2014) informatik.uni-konstanz.de "NoSQL databases" Dostopno na http://www.informatik.uni-konstanz.de/en/berthold/teaching/ss2013/analyzing-bigdata/

[13] (2014) Pomagalnik "NoSQL" Dostopno na http://www.pomagalnik.com/slovar/nosql-not-only-sql/

[14] (2014) catthell.net "Scalable SQL and NoSQL Data Stores" Dostopno na http://www.cattell.net/datastores/Datastores.pdf

[15] (2014) mongoDB "NoSQL databases explained" Dostopno na http://www.mongodb.com/nosql-explained#implement-nosql

[16] (2014) Wikipedia "NoSQL" Dostopno na http://en.wikipedia.org/wiki/NoSQL

[17] (2014) SQL vs NoSQL "NoSQLDatabases.pdf" Dostopno na http://www.cse.yorku.ca/ jarek/courses/6421/F12/presentations/NoSQLDatabases.pdf

[18] (2014) NoSQL Data Stores "NoSQL.pptx" Dostopno na http://home.aubg.bg/students/ENL100/Cloud%20Computing/

[19] (2014) SQL and NoSQL Databases - IJARCSSE "V2I800154.pdf " Dostopno na http://www.ijarcsse.com/docs/papers/8_August2012/Volume_2_issue_8/V2I800154.pdf

[20] (2014) NoSQL Databases – nosqldbs.pdf "NoSQL Databases" Dostopno na http://coitweb.uncc.edu/ xwu/5160/nosqldbs.pdf

[21] (2014) Wide Column Stores - DB-Engines Encyclopedia "Wide Column Stores"Dostopno na http://db-engines.com/en/article/Wide+Column+Stores

[22] (2014) stackoverflow "Explanation of BASE terminology" Dostopno na http://stackoverflow.com/questions/3342497/explanation-of-base-terminology

[23] (2014) NewSQL: what's this? "NewSQL" Dostopno na http://labs.sogeti.com/newsql-whats/

[24] (2014) What is NewSQL? - Definition from WhatIs.com "NewSQL definition" Dostopno na http://searchdatamanagement.techtarget.com/definition/NewSQL

[25] (2014) NewSQL — The New Way to Handle Big Data "NewSQL "Dostopno na http://www.opensourceforu.com/2012/01/newsql-handle-big-data/

[26] (2014) Wikipedia "NewSQL" Dostopno na http://en.wikipedia.org/wiki/NewSQL

[27] (2014) Spletna učilnica FRI "NoSQL (Sanja Božič)" Dostopno na https://ucilnica.fri.uni-lj.si/

[28] (2014) technopedia "NewSQL" Dostopno na http://www.techopedia.com/definition/29093/newsql

[29] (2014) 16 NoSQL, NewSQL Databases To Watch "NoSQL, NewSQL Databases" Dostopno na http://www.informationweek.com/big-data/big-data-analytics/16-nosql-newsql-databases-to-watch/d/d-id/1269559

[30] (2014) A Study of NoSQL and NewSQL databases for data aggregation on Big Data "NoSQL NewSQL Databases"Dostopno na http://www.diva-portal.org/smash/get/diva2:706302/FULLTEXT01.pdf

[31] (2014) NuoDB "NuoDB" Dostopno na http://www.nuodb.com/

[32] (2014) Wikipedia "NuoDB" Dostopno na http://en.wikipedia.org/wiki/NuoDB

[33] (2014) Wikipedia "Google Spanner" Dostopno na http://en.wikipedia.org/wiki/Spanner_(database)

[34] (2014) InformationWeek "NoSQL, NewSQL, or RDBMS: How To Choose " Dostopno na http://www.informationweek.com/big-data/big-data-analytics/nosql-newsql-or-rdbms-how-to-choose/a/d-id/1297861

[35] (2014) arxiv.org "NewSQL: Towards Next-Generation Scalable RDBMS for Online Transaction Processing (OLTP) for Big Data Management" Dostopno na http://arxiv.org/ftp/arxiv/papers/1411/1411.7343.pdf

[36] (2014) Usenix "OldSQL vs. NoSQL vs.NewSQL on New OLTP" Dostopno na www.usenix.org/events/lisa11/tech/slides/stonebraker.pdf

[37] (2014) coiteweb "NoSQL Databases" Dostopno na http://webpages.uncc.edu/xwu/5160/nosqldbs.pdf

[38] (2014) Datastax "CQL for Cassandra 2.x Documentation" Dostopno na http://www.datastax.com/documentation/cql/3.1/pdf/cql31.pdf

[39] (2014) (2014) US Ltr - NuoDB "NuoDB® User Guide and Reference" Dostopno na http://doc.nuodb.com/.../NuoDB_Documentation_v1.0.2%20(1).pdf?...1...

[40] (2014) Fierce Cio "How to choose between RDBMS, NoSQL and NewSQL" Dostopno na http://www.fiercecio.com/story/how-choose-between-rdbms-nosql-and-newsql/2014-08-13