# NewSQL: pregledni rad

Saša Sambolek
Srednja škola Tina Ujevića Kutina, Kutina, Hrvatska
sasa.sambolek@gmail.com

**Sažetak - Nepredvidive količine podataka, performanse i skalabilnost zahtjevi su modernih aplikacija stavljeni pred tradicionalne sustave za upravljanje podatcima. Odgovor na te zahtjeve je pojava NoSQL sustava za upravljanje podatcima, no njihovi nedostatci kao što su upravljanje podatcima kroz aplikacije i izostanak ACID svojstava doveli su do pojave alternativnih sustava baza podataka koji se nazivaju NewSQL. NewSQL je nova klasa modernih relacijskih sustava za upravljanje bazama podataka koji podržavaju skalabilne performanse NoSQL sustava za on-line obradu transakcija, istodobno zadržavajući ACID svojstva tradicionalnih sustava baza podataka. U ovom radu je opisan NewSQL sustav upravljanja, prikazana je usporedba s NoSQL i tradicionalnim sustavom baza podataka, opisana je arhitektura i klasifikacija NewSQL sustava, navedene su karakteristike, te je dana usporedba nekih postojećih rješenja.**

**Ključne riječi: NewSQL, NoSQL, baze podataka, ACID.**

## I. UVOD

Današnje poslovanje crpi snagu iz podataka nastalih u sve kraćim i frekventnijim vremenskim intervalima, što neminovno vodi k velikim količinama podataka [4]. Mnogi koncepti nastali su iz potrebe da se na adekvatan način zadovolje zahtjevi obrade velike količine podataka. Prema inicijativi DB-Engines [5], postoji više od 280 sustava za upravljanje podatcima grupiranih u 13 kategorija. Samo 7 klasa (*Relation*, *Key-value*, *Document*, *Column-family*, *Graph Data Model*, *NewSQL* i *NativeXML*) predstavlja gotovo cijelo tržište baza podataka [6]. U većini slučajeva kad se spomene baza podataka pomisli se na relacijsku bazu podataka, budući daje i danas relacijska baza dovoljna za pogon milijuna web stranica. Ipak, relacijske baze podataka ne nose se dobro s današnjim zahtjevima, kao što su performanse, skaliranje, distribuiranje podataka na više čvorova i slično. Kad govorimo o performansama, relacijske baze podataka su sposobne obraditi tisuće transakcija u sekundi, no današnji zahtjevi pred OLTP(engl. *Online Transaction Processing*) sustavom su oglašavanje u realnom vremenu, detekcija prijevara, *multi-player* igre, analiza rizika itd. U tim slučajevima imamo i do milijun transakcija u sekundi, što klasične relacijske baze podataka ne mogu obraditi.

Odgovor na nedostatke relacijskih baza dali su NoSQL sustavi, no oni su uz sve prednosti donijeli i svoje nedostatke, npr. ne podržavaju ACID (engl. *Atomicity, Consistency, Isolation, Durability*) svojstva i jezik SQL (engl. *Structured Query Language*).

U ovom radu dan je pregled baza podataka koje podržavaju ACID svojstva i SQL jezik poput relacijskih baza podataka a istovremeno omogućavaju skaliranje, distribuiranje i visoke performanse.

Sadržaj i struktura ovog rada su sljedeći: poglavlje II daje kratki opis baza podataka i temeljnih pojmova vezanih uz iste, u poglavlju III opisana je NewSQL baza podataka i navedene karakteristike. U IV poglavlju dana je kategorizacija, dok su u V opisane neke od danas najpopularnijih distribucija. U VI je dana usporedba triju postojećih rješenja. Posljednje poglavlje VII daje zaključke i prijedloge za budući rad.

## II. POZADINA

Baza podataka je organizirani skup podataka. Podaci se obično organiziraju za modeliranje relevantnih aspekata stvarnosti na način da podržavaju procese koji zahtijevaju informacije. Sustavi za upravljanje bazom podataka (SUBP) su posebno dizajnirane aplikacije koje su u interakciji s korisnikom, s drugim aplikacijama i sa samom bazom podataka radi prikupljanja i analize podataka. Općenito, SUBP (sustav za upravljanje bazama podataka)je programska podrška koja omogućuje stvaranje, ažuriranje i administraciju baza podataka. Pomoću SUBP korisnici i računalni programeri sustavno dodaju, dohvaćaju ili ažuriraju podatke. Stvaranje podataka omogućeno je kroz DDL (Data Definition Language), dok je upravljanje (Create, Insert, Update, Delete) podatcima kroz DML (Data Manipulation Language). Neki od poznatih SUBP-ova su MySQL, MariaDB, PostgreSQL, SQLite, Microsoft SQL Server, Oracle, SAP HANA, dBASE, FoxPro, IBM DB2, Libre Office Base i File Maker Pro. Baza podataka općenito nije prenosiva na različite SUBP, ali različiti SUBP mogu međusobno djelovati pomoću standarda kao što su SQL, ODBC ili JDBC kako bi se omogućio jedan zahtjev za rad s više baza podataka. Baze podataka su stvorene s ciljem obrade velike količine podataka koje se može unositi, pohranjivati, dohvati i njima upravljati. Baze podataka su postavljene tako da se pomoću skupa programske podrške omogućuje svim korisnicima pristup svim podatcima. [1] Funkcije većine SUBP možemo svrstati u četiri skupine:

- *Definicija podataka* – Definiranje nove strukture podataka, uklanjanje strukture, promjena strukture postojećih podataka.
- *Ažuriranje* – Umetanje, mijenjanje, brisanje podataka.
- *Dohvaćanje* – Dobivanje informacija, bilo za korisnika putem upita i izvješća ili za obradu pomoću aplikacije.
- *Administracija* – Registracija i praćenje korisnika, osiguranje sigurnosti podataka, praćenje performansi, održavanje integriteta

podataka, kontrola konkurentnosti i oporavak podataka u slučaju ispada.

Baze podataka i SUBP mogu se kategorizirati prema modelu kojeg podržavaju (kao što su relacijske ili XML), vrsti računala na kojem su pokrenute (od klastera poslužitelja do mobitela), jeziku upita koji koriste za pristup bazi podataka(kao što su SQL ili XQuery) i njihovom unutarnjem inženjerstvu koje utječe na performanse, skalabilnost, prilagodljivost i sigurnost.

Prema [7] informacijske sustave možemo svrstati u dvije kategorije – sustavi koji pružaju podršku pri izvođenju poslovnih procesa, te sustavi koji pružaju podršku pri analizi poslovnih procesa i pripadajućih podataka. Prvi spomenuti sustavi, koji pružaju podršku pri izvođenju poslovnih procesa, nazivaju se OLTP sustavi ili transakcijski sustavi. Implementiraju se na operativnoj razini poslovanja, a samim time automatiziraju većinu aktivnosti pojedinog poslovnog procesa. Njih karakterizira sposobnost brzog evidentiranja, ažuriranja i brisanja detalja vezanih za izvršenje određene radnje, tj. brza reakcija na zahtjeve korisnika. OLTP sustavi funkcioniraju na operativnoj razini, prate poslovne procese na razini pojedine transakcije te pohranjuju podatke o nastalim promjenama. Realizirani su na relacijskoj tehnologiji, optimizirani za unos i ažuriranje transakcijskih podataka, koji su stabilni i predvidivi. Stoga nova generacija OLTP sustava treba naći odgovor na trend rasta količine podataka uz veliki promet, zadovoljiti visok stupanj dostupnosti i velikog odziva.

SQL je najpopularniji računalni jezik za izradu, traženje, ažuriranje i brisanje podataka u relacijskim bazama podataka. SQL je standardiziran preko standarda ANSI i ISO [18].

U računalnoj znanosti, termin ACID predstavlja skup svojstava koja jamče pouzdanost transakcije baze podataka.

- Atomarnost (engl. *Atomicity*). Transakcija mora pratiti pravilo: „Sve ili ništa“. Svaka transakcija mora biti atomarna. Ukoliko jedan dio transakcije pođe po zlu, sve prošle operacije moraju biti opozvane, sve buduće moraju biti prekinute i baza podataka mora biti u stanju u kojem je bila prije početka transakcije.
- Konzistentnost (engl. *Consistency*). Svojstvo koje govori da samo valjani podaci mogu biti zapisani u bazu podataka. Ukoliko izvršena transakcija može naštetiti konzistentnosti baze podataka, transakcija se mora prekinuti i baza podataka vratiti u konzistentno stanje. Ukoliko je transakcija valjana, ona uvijek mora voditi bazu podataka iz jednog konzistentnog stanja u drugo konzistentno stanje.
- Izolacija (engl. *Isolation*). Svojstvo koje zahtjeva da transakcije koje se izvršavaju u isto vrijeme nemaju utjecaja jedna na drugu. Npr. ukoliko imamo dvije transakcije koje izvršavaju neke operacije nad bazom podataka u isto vrijeme, sustav mora osigurati da transakcije nemaju utjecaja jedna na drugu, jer inače može doći da nepoželjnih nekonzistentnih stanja.
- Trajnost (engl. *Durability*). Svojstvo koje traži da svaka uspješno završena transakcija u bazi podataka mora biti trajna, odnosno da sve promjene nad podacima moraju ostati i u slučaju neočekivanih ispada.

Kako NoSQL sustavi ne podržavaju ACID svojstva uz te sustave često se vežu BASE svojstva:

- *Basically Available*: Većina podataka je dostupna veći dio vremena.
- *Soft state*: Stanje sustava se može promijeniti, čak i u slučaju kada nemamo eksplicitne upite nad bazom podataka. Razlog tomu je što ažuriranje može doći s ažuriranjem čvora kojem pripada baza podataka.
- *Eventually consistent*: Sustav će postati konzistentan s vremenom

Osim BASE svojstva, za NoSQL baze podataka često se veže i CAP teorem (poznat i kao Brewerov teorem) koji govori kako je nemoguće zadovoljiti sva tri poželjna svojstva u distribuiranim sustavima: konzistenciju (engl. *consistency*), dostupnost (engl. *availability*) i toleranciju na ispad djela sustava (engl. *partition tolerance*). Kod razvoja sustava za upravljanje bazama podataka, mogu se odabrati samo dva od navedena tri svojstva distribuiranih sustava. Tradicionalni sustavi za upravljanje bazama podataka koncentriraju se na CA – konzistentnost i dostupnost. Time se povećava složenost kod raspodjele sustava na više čvorova, dok se NoSQL sustavi koncentriraju na PA i PC – žrtvuje se konzistentnost ili dostupnost dok tolerancija na podijeljenost uvijek ostaje kako bi se zadržala jednostavna skalabilnost.

Pod pojmom *Big Data* obično se smatra velika količina strukturiranih i nestrukturiranih podataka kojima se ne može učinkovito upravljati uobičajenim alatima za upravljanje podacima [8]. [9] Mogućnost prikupljanja, pretraživanja i korelacije unutar određenog skupa podataka su karakteristike koje definiraju *Big Data* paradigmu. *Volume*, *velocity*, *variety* odnosno 3V model također se često koristi za opisivanje tog trenda. *Volume* (volumen) označava osnovnu karakteristiku *Big Data* trenda, a to je golema količina podataka, *variety* (raznolikost) je karakteristika *Big Data* podatkovnog seta koja govori da kod tog trenda možemo očekivati heterogene podatke (.pdf, .xlsx, različite videoformate, SMS poruke te razne druge oblike sadržaja) dok *velocity* označava brzinu kojom stižu novi podaci. Jedan od ključnih napredaka u rješavanju „*Big Data*“ problema je pojava alternativne tehnologije baze podataka kao npr. NoSQL (engl. *Not only Structured Query Language*). Takvi sustavi za upravljanje bazama podataka znatno se razlikuju od klasičnog relacijskog sustava za upravljanje bazama podataka. Ne zahtijevaju tipičnu tabličnu shemu, obično izbjegavaju operacije pridruživanja, podaci su pohranjeni na distribuirani način, a podatcima se pristupa i analizira ih pomoću aplikacije.

Relacijske baze podataka su vertikalno skalabilne (engl. *scale-up*), odnosno brzinu same baze podataka možemo povećati npr. ugradnjom boljeg procesora u poslužitelju, ugradnjom više memorije, ugradnjom SSD

diska, što u određenom trenutku može postati skupo i neučinkovito. Horizontalno skaliranje (engl. *scale-out*) odnosi se na povećanje performansi dodavanjem novih poslužitelja (čvorova).

TABLICA I. USPOREDBA TRADICIONALNIH, NOSQL I NEWSQL BAZA PODATAKA [2].

| | Tradicionalni SUBP | NoSQL | NewSQL |
|---|---|---|---|
| SQL | Podržan | Nije podržan | Podržan |
| Ovisnost o serveru | Jedan server | Više servera/ distribuirano | Više servera/ distribuirano |
| Vrsta SUBP | Relacijski | Nerelacijski | Relacijski |
| Shema | Tablica | Key-value, column stor, Document store | Oboje |
| Skladištenje | Na disku + cache | Na disku + cache | Na disku + cache |
| Podržana svojstva | ACID | CAP kroz BASE | ACID |
| Horizontalna Skalabilnost | Nije podržano | Podržano | Podržano |
| Složenost Upita | Niska | Visoka | Vrlo visoka |
| Sigurnost | Vrlo visoka | Niska | Niska |
| Big volume | Slabe performanse | Potpuno podržano | Potpuno podržano |
| OLTP | Nije u potpunosti podržan | Podržan | Potpuno podržan |
| Podrška za oblak | Nije u potpunosti podržan | Podržan | Potpuno podržan |

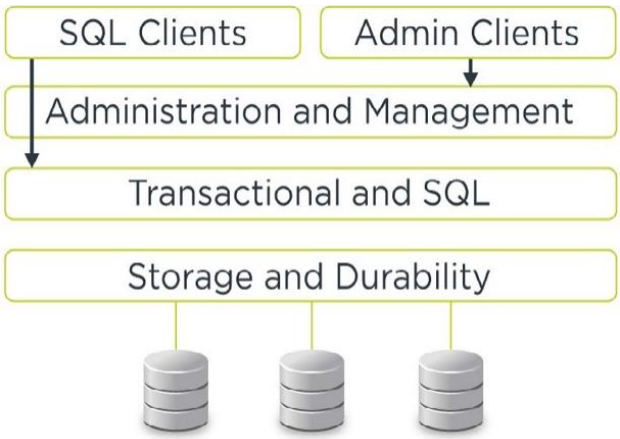
## III. NEWSQL

Izraz NewSQL je prvi put bio upotrijebljen 2011.g. u članku Matthewa Asletta o pojavi novih sustava skupine 451 [1]. Može se reći da je NewSQL nova generacija skalabilnih relacijskih baza podataka za OLTP performansama poput NoSQL-a zadržavajući ACID svojstva za transakcije tradicionalnih baza podataka [2]. Ovi sustavi su poznati i kao *multi-model* sustavi budući podržavaju više od jednog modela podataka [6]. Ne postoji univerzalno rješenje NewSQL arhitekture ovih sustava već se rješenja razlikuju od distribucije do distribucije, pa time i način podrške transakcija, replikacija, *shrading* ili metode pristupa klijenta.

Karakteristike NewSQL-a [3]:

- kao primarni jezik za interakciju sa aplikacijama NewSQL koristi mogućnosti SQL-a,
- NewSQL podržava ACID svojstva,
- NewSQL omogućuje više transakcija nad istim resursom,
- NewSQL arhitektura pruža mnogo veću učinkovitost po čvoru (engl. *per-node*) nego tradicionalni SUBP.
- NewSQL podržava horizontalno skaliranje, *shared-nothing-architecture* [1], izvođenje na velikom broju čvorova bez „uskih grla".
- NewSQL sustavi su otprilike 50 puta brži od tradicionalnih OLTP SUBP
- Distribuirana arhitektura

Iako se arhitektura NewSQL sustava međusobno razlikuje od sustava do sustava, dvije istaknute odlike vrijede za sve, podrška relacijskih modela podatak i SQL kao primarno sučelje. Ciljane aplikacije NewSQL sustava karakterizira velik broj transakcija koje su kratkotrajne, dotiče se mali podskup podataka pomoću indeksiranih upita i ponavljanje (npr. izvršavanje istih upita s različitim ulazima) [10]. Slika 1. prikazuje arhitekturu popularne NewSQL baze podataka NuoDB na kojoj vidimo podjelu u tri sloja; upravljački sloj, transakcijski sloj i sloj za pohranu.




SLIKA 1. PRIKAZ ARHITEKTURE NUODB BAZE PODATAKA [12]

## IV. NEWSQL KATEGORIZACIJA

Kategorizacija se temelji na različitim pristupima u želji za očuvanjem SQL sučelja uz rješavanje problema skalabilnosti, performansi i tradicionalnih OLTP sustava.

### A. *Nove baze podataka*

NewSQL sustavi su projektirani od početka kako bi se postigle što bolje performanse i skalabilnost. Ovi sustavi traže migraciju podataka i izmjene programskog koda. Jedan od ključnih razloga u poboljšanju performansi je pojava *non-disc* memorija kao ili nove vrste diskova (Flash/SSD) kao primarna spremišta podataka. Neki od primjera ovih baza podataka su:VoltDB, NuoDB, Clustrix, TransLattice, Google Spanner, MemSQL.

### B. *Novi MySQL storage engines*

MySQL kao dio LAMP stack-a (Linux, Apache, MySQL, PHP) intenzivno se koristi u OLTP. Kako bi se prevladali problemi skalabilnosti MySQL-a razvijen je niz *storage engines-a*, kao što su: Akiban, MySQL NDB cluster, GenieDB, Tokutek, itd. Dobar dio je korištenje MySQL sučelja ali je loša strana što nije podržana

[1]Shared-nothing-architecture je distribuirana računalna arhitektura u kojoj je svaki čvor neovisan i samodostatan.

migracija podataka iz drugih baza podataka (uključujući MySQL).

### C. Transparent clustering/sharding

Ova rješenja nastoje zadržati OLTP baze podataka u njihovom izvornom obliku, gdje se u prvom slučaju pomoću kreiranja klastera nastoji osigurati skalabilnost, dok se drugim pristupom *shrading* nastoji poboljšati skalabilnost.

Predstavnici prvog pristupa su Schooner MySQL, Tungsten, ScalArc, dok dbShards i ScaleBase, koji omogućuje dobivanje skalabilnosti koristeći postojeću bazu podataka bez prepisivanja koda ili migracije podataka, slijede drugi pristup [2].

### D. Database-as-a-service

Tradicionalni SUBP-a nisu prikladni za izvođenje u oblaku, stoga se pojavila potreba da se SUBP prilagode za izvođenje na javnim i privatnim oblacima. To su baze podataka koje se postavljaju pomoću virtualnih strojeva. NuoDB, MySQL, GaianDB i Oracle baza podataka mogu se primjenjivati u oblaku. Drugi model za bazu podataka u oblaku je *database-as-a-service*. Ta rješenja nude programsku podršku za baze podataka kao i fizičku pohranu. Kod takvih sustava nije potrebna instalacija i konfiguracija već na zahtjev korisnika se pokreće usluga. Davatelj usluge je odgovoran za održavanje, nadogradnju i administriranje baze podataka. Na primjer, Microsoft SQL Azure [20], ClearDB [21] i EnterpriseDB [22] pripadaju kategoriji *database-as-a-service*NewSQL baza podataka [19].

## V. NOVE BAZE PODATAKA

### A. ClustrixDB

ClustrixDB [11] je scale-out SQL baza podataka izgrađena iz temelja s distribuiranom *shared-nothing*arhitekturom, automatskom preraspodjelom podataka, ugrađenom tolerancijom kvarova i sve to dostupno kroz jednostavno SQL sučelje i podršku MySQL mogućnosti.

### B. NuoDB

NuoDB [12] je distribuirani sustav za upravljanje bazama podataka u oblaku, koji podržava ACID svojstva i SQL jezik za upite. NuoDB pruža globalnu platformu za upravljanje podatcima sa zajamčenom visokom dostupnošću, jednostavnim dodavanjem novih poslužitelja. NuoDB koristi peer-to-peer komunikaciju za davanje ruta zadatcima do čvorova, također baza zadatke distribuira preko nekoliko procesora kako bi se izbjegla uska grla.

### C. VoltDB

VoltDB [13] je *tzv. in-memory* relacijska, horizontalno skalabilna, baza podataka koja podržava ACID svojstva, SQL i JSON (engl. *Java Script Object Notation*) [23] fleksibilnost, ima veliku skalabilnost i mogućnost obrade velike količine podataka u stvarnom vremenu. VoltDB računa da je, uz današnje relativno niske cijene procesora i radne memorije koje se nalaze u velikim količinama u poslužiteljima, moguće bazu podataka učitati u memoriju i time postići veliku brzinu rada. Transakcije se obavljaju pomoću pohranjenih procedura napisanih u kombinaciji Java-e i SQL-a. VoltDB je u potpunosti usklađen s ACID svojstvima. Podatci su trajno pohranjeni na disku. Trajnost je osigurana kontinuiranim snimkama (engl. *snapshots*), asinkronim bilježenjem (engl. *asynchronous command logging*), što stvar snimke i zapisnik svih transakcija između snimki i sinkronim bilježenjem (engl. *synchronous command logging*), koje upisuje transakcije u zapisnik nakon izvršenja i prije izvršavanja na bazi podataka. Što omogućava da su sve transakcije izvršene, tj. niti jedna transakcija nije izgubljena.

### D. Google Spanner

Google Spanner temelji se na polu-relacijskom modelu u kojem se tablice vide kao preslikavanje iz stupca primarnih ključeva u ostale stupce. Google Spanner koristi drugačiji model razdvajanja. Spanner implementacija sadrži skup poslužitelja poznatih kao *tzv. spanserveri*, koji su odgovorni čvorovi (engl. *nodes*) za posluživanje podataka klijentima. Spanserver upravlja s više stotina ili tisuća tablica, od kojih svaka sadrži skup direktorija. Direktorij je u osnovi skup redak koji dijeli zajednički prefiks ključa. Direktorij se također smatra osnovnom jedinicom konfiguracije prostora, koji se koristi za definiranje ograničenja za particioniranje podataka i replikacije među dostupnim tablicama. SUBP automatski pomiče direktorije između spanservera poštujući određene kriterije u svrhu poboljšavanja performansi.

### E. MemSQL

MemSQL [14] je *real-time in-memory* baza podataka za transakcije i analizu, što znači da je projektirana za maksimalnu brzinu i učinkovitost upita. MemSQL koristi poznato SQL sučelje, horizontalno skalabilnu distribuiranu arhitekturu. MemSQL kombinira *lock-free* strukture podataka i *just-in-time* (JIT) prevodilac za obradu podataka. Konkretno MemSQL dobivene SQL upite pretvara u C++ koji prevodi pomoću GCC-a[2]. MemSQL distribuira kopije podataka na više čvorova što osigurava sigurnost podataka, dok je trajnost podataka osigurana vođenjem zapisnika transakcija i snimkama stanja baze podataka.

### F. TransLattice Elastic Database

TransLattice Elastic Database (TED) je skalabilan, vrlo dostupan sustav za upravljanje bazom podataka, geografski distribuirani relacijski SQL server napravljen za OLTP. Podržava različita opterećenja aplikacija s visokom dostupnosti i izuzetnim performansama za udaljene korisnike. TED podržava neke važne značajke kao što su elastičnost, skalabilnost, migracija u oblak uz smanjenje troškova.

## VI. KOMPARACIJA ZNAČAJKI CLUSTRIXDB, NUODB I VOLTDB

ClustrixDB, NuoDB i VoltDB baze odabrane su za ovu komparaciju kao NewSQL baze podataka napravljene od početka. Na [11][12][13] dostupna je detaljna dokumentacija, dok na [5] možemo vidjeti da su ove baze jedne od popularnijih u klasi NewSQL baza podataka.

[2]GCC – GNU CompilerCollection

*Data model*

Ova rješenja se po definiciji temelje na relacijskom modelu. VoltDB, Clustrix i NuoDB nude svojim klijentima čisti relacijski pregled podataka. Iako klijenti sa bazom komuniciraju u smislu tablica i relacija, važno je napomenuti da NewSQL može koristiti različita rješenja za interni prikaz podataka. Npr. NuoDB može svoje podatke pohraniti u bilo kojem kompatibilnom *key-value*spremniku podataka.

*Upiti*

SQL jezik za upite jedna je od definiranih svojstava NewSQL bazi podataka, ali razina SQL podrške znatno varira od distribucija. Clustrix i NuoDB su najvećoj mjeri kompatibilni sa standardnim SQL-om. VoltDB ima veći broj ograničenja: tablice se ne mogu spajati same sa sobom, i sve spojene tablice moraju biti udružene preko iste vrijednosti. Način interakcije s VoltDB odvija se putem pohranjenih procedura (engl. *Stored Procedures*). Te procedure su pisane u Javi, gdje se programira logika i SQL izrazi.

*Skaliranje*

Jedna od glavnih karakteristika NewSQL-a je mogućnost horizontalnog skaliranja jednostavnim dodavanjem novih poslužitelja. S obzirom na zahtjev, skaliranje možemo podijeliti na: skaliranje zahtjeva za čitanjem, skaliranje zahtjeva za pisanjem i skaliranje pohrane podataka. Na sposobnost skaliranja podataka značajno utječe particioniranje, replikacija, dosljednost (koja je osigurana ACID svojstvima) i konkurencija strategije upravljanja. Na primjer, particioniranje određuje raspodjelu podataka između više poslužitelja, te je stoga sredstvo za postizanje sve tri dimenzije skaliranja.

Drugi važan čimbenik skaliranja čitanja i pisanja je repliciranje, spremanje istih podataka na više poslužitelja kako bi operacije čitanja i pisanja bile distribuirane preko svih poslužitelja. Repliciranje također ima važnu ulogu u pružanju tolerancije na kvarove, jer i u slučaju kvara na jednom ili više poslužitelja podatci su dostupni.

Konačno, još jedan važan čimbenik pri zahtjevu za skaliranjem čitanja i pisanja je kontrola konkurencije (engl. *concurrency control*). Jednostavne tehnike zaključavanja čitanja/pisanja ne mogu osigurati kontrolu konkurencije koju zahtijeva NewSQL rješenje. Stoga, većina rješenja koristi napredne tehnike, kao što je optimistično zaključavanje i više verzija kontrole konkurentnosti MVCC (engl. *multi-version concurrency control*).

*1) Podjela podataka / particioniranje*

VoltDB koristi tradicionalni pristup kojem je svaka tablica podijeljena pomoću jednog ključa a redovi su distribuirani među poslužitelja pomoću konzistentnog algoritma raspršenja. Pohranjene procedure mogu se izvršavati na jednoj patriciji ili na svima; međutim, nedostatak je da korisnik odlučuje između ove dvije opcije. Podatci i obrada istih je podijeljena kroz sve procesorske jezgre unutar poslužitelja koji čine VoltDB klaster. Proširenje *shared-noting* osnove realizira se na razini jezgre, povećanjem jezgara po procesoru, tj. procesora s više jezgri.

Clustrix podatke razdvaja pomoću dosljednog algoritma raspršivanja preko korisnički definiranog primarnog ključa. Osim toga, Clustrix također razdvaja tablice indeksa koristeći indeksirane stupce kao ključeve. Teorijski, ova strategija omogućava paralelno pretraživanje preko tih indeksa, što dovodi do bržeg rješavanja upita.

NuoDB je rješenje koje koristi potpuno drugačiji pristup za razdvajanje podataka. NuoDB implementacija sastoji se od niza upravitelja pohranom (UP) (engl. *storage managers*) i transakcijskih upravitelja (TU) (engl. *transaction managers*). UP su odgovorni za održavanje podataka, dok su TU-i čvorovi koji obrađuju upite. Svaki UP ima potpunu kopiju cijelog podatka, što u osnovi znači da se raspodjela ne odvija unutar UP. Ipak, temeljni „*key-value*“ spremnik korišten UP može razdvajati podatke, iako to nije niti vidljivo, niti kontrolirano od strane korisnika.

*2) Repliciranje*

Shema repliciranja NewSQL-a može se smatrati *multi-master* ili *masterless* shema jer bilo koji čvor može izvršiti ažuriranje. U VoltDB i Clustrix, upravitelj transakcijama/sesijama prima ažuriranja, koja se prosljeđuju svim replikama i izvršavaju paralelno.

NuoDB, redovi tablice su u memoriji prikazani kao distribuirani objekti koji komuniciraju asinkrono da repliciraju svoje promjene stanja. NuoDB je patentirao elastičan način rada baze podataka koji podrazumijeva fragmentaciju baze podataka u distribuirane objekte nazvane atomi. Svaka promjena na jednoj od kopija atoma se replicira na sve druge lokacije koje sadrže kopiju tog atoma.

*3) Kontrola konkurentnosti*

Kontrola konkurentnosti je od velikog interesa NewSQL baza podataka, budući one rade s velikim brojem korisnika i vrlo velikim brojem zahtijeva nad bazom podataka.

Glavne sheme kontrole konkurentnosti mogu se kategorizirati kao pesimistična i optimistična. Pesimistična kontrola podudarnosti ili pesimistično zaključavanje pretpostavlja da će dva ili više korisnika istodobno pokušati ažurirati isti zapis u isto vrijeme. Kako bi se spriječila ovakva situacija, zaključava se entitet kojem se pristupa, te se daje ekskluzivni pristup jednoj operaciji, ostali korisnici koji pokušavaju pristupiti istom podatku moraju čekati svoj red.

Optimistična kontrola konkurentnosti ili optimistično zaključavanje pretpostavlja da su konflikti mogući, no rijetki. Stoga, umjesto zaključavanja zapisa, SUBP na

kraju operacije provjerava je li bilo istodobnih pokušaja izmjene zapisa. Ako se utvrdi sukob, mogu se koristiti različite strategije za rješavanje sukoba, kao trenutno prekidanje operacije ili ponovnog pokušaja jedne od operacija. Clustrix i NuoDB implementiraju optimističnu kontrolu podudarnosti s *multi-version concurrency control* (skraćeno, MVCC). U MVCC-u kada je potrebno ažurirati zapis, ne piše se preko starog zapisa već se dodaje novi i stari označava kao „zastario“. Time je pohranjeno više verzija istog podatka ali je samo jedan označen kao trenutni. S MVCC pristupom, korisnik vidi podatke kakvi su bili u trenutku kad ih je počeo čitati, čak i ako su podatci u međuvremenu izmijenjeni ili obrisani od strane drugih korisnika.

VoltDB primjenjuje zanimljivu alternativu kontrole konkurentnosti. Ovaj sustav pretpostavlja da je ukupna raspoloživa memorija dovoljno velika za pohranu cijelog sustava. Također, on pretpostavlja da su sve korisničke transakcije kratkog vijeka i da može biti vrlo efikasno izvršavati ih u memoriji (*in-memory*). Na temelju tih pretpostavki sve se transakcije izvršavaju sekvencijalno u jednom nizu (engl. *single-threaded*), bez zaključavanja. Time što su kod VoltDB pohranjene procedure jedinica transakcije, koje se izvršavaju na particiji koja sadrži tražene podatke izbjegava se bespotrebno slanje zahtijeva između SQL naredbi. Pohranjene procedure se izvode serijski do završetka u nizu što omogućuje rad bez zaključivanja (*locking*) ili *latching*. Budući da su podatci u memoriji i lokalnoj particiji, pohranjene procedure se izvršavaju u mikrosekundi.VoltDB-ova shema izvršavanja pohranjenih procedura omogućuje istovremeno pokretanje pohranjenih procedura na svim čvorovima osiguravajući postojanje jednog globalnog poretka za izvođenje.

*Sigurnost*

Sigurnost je važan aspekt sustava za pohranu podataka, koji je zanemaren od većine NoSQL i NewSQL sustava. U ovom poglavlju sustavi za pohranu podataka analizirani su prema sljedećim značajkama:

- Autentifikacija: mehanizmi koji omogućuju provjeru identiteta korisnika koji pristupa podatcima. To se obično postiže putem korisničke lozinke prilikom prijave, ali i pomoću sofisticiranih mehanizama kao što su korisnički certifikati.
- Autorizacija: ovo se odnosi na sposobnost da se osigura kontrola pristupa podatcima. Autorizacija se obično provodi kroz organizaciju svakog korisnika i skupom njegovih ovlasti. Na primjer, neke baze podataka mogu zahtijevati posebne ovlasti za pisanje i čitanje, kreiranje korisnika ili administraciju.
- Enkripcija: se odnosi na mehanizme za šifriranje podataka, čime se osigurava zaštita podataka u slučaju neovlaštenog pristupa. Rješenje šifriranja može se prikazati kroz tri razine:
  - Enkripcija podataka na disku
  - Enkripcija podataka prilikom komunikacije korisnik-poslužitelj
  - Enkripcija podataka prilikom komunikacije poslužitelj-poslužitelj
- Kontrola: obično se odnosi na stvaranje evidencije o prijavi i događajima koji su se dogodili u sustavu. Što je posebno važno za forenzičku analizu sigurnosnih događaja.

Clustrix i NuoDB koriste autorizaciju i autentifikaciju tradicionalnih SUBP podržavajući GRANT/REVOKE ovlasti. VoltDB implementira kontrolu pristupa izvršavanjem pohranjenih procedura.

TABLICA II. KOMPARACIJA ZNAČAJKI VOLTDB, CLUSTRIXDB I NUODB.

| | VoltDB | ClustrixDB | NuoDB |
|---|---|---|---|
| Upiti | SQL | SQL | SQL |
| Druge API | CLI[3] i API[4] u nekoliko jezika. JDBC podržan. | Wire protocol kompatibilan sa MySQL. | CLI i upravljački programi za većinu API-a za pristup podatcima (JDBC, ODBC, ADO.NET). Također pruža C++ API. |
| Pohranjene procedure | Pohranjene procedure su pisane u JAVI. Tablice se ne mogu povezivati same sa sobom, a sve povezane tablice moraju biti podijeljene preko iste vrijednosti. | Podržava MySQL pohranjene procedure uz neke iznimke | Ne podržava pohranjene procedure |
| Licenca | Otvoreni kod AGPL 3.0 licenca | Zatvorenog koda | Zatvorenog koda |
| Particioniranje | Dosljedno raspršivanje. Korisnici određuju hoće li se pohranjene procedure izvoditi na jednom ili više poslužitelja. | Dosljedno raspršenje. | Nema particija. Temeljni key-value spremnik može razdvajati podatke, ali to nije vidljivo korisnicima |
| Repliciranje | Ažuriranje se izvršava na svim replikama u isto vrijeme | Ažuriranje se izvršava na svim replikama u isto vrijeme | Multi-master asinkrono repliciranje |
| Konzistencija | Jaka dosljednost | Jaka dosljednost | Eventualna dosljednost |
| Kontrola konkurentnosti | Izvršavanje u jednom nizu, | MVCC | MVCC |

[3] Comand-line interface

[4] Application programming interface

| | | | |
|---|---|---|---|
| | nije potrebna kontrola konkurentnosti | | |
| Enkripcija podataka na disku | Ne | | Ne podržava. Moguće je koristiti enkripciju poslužitelja na kojem se nalazi. |
| Enkripcija klijent-poslužitelj | Ne | Da | Da |
| Enkripcija poslužitelj-poslužitelj | Ne | | Da |
| Autentifikacija | Da, korisnici su definirani u datoteci koju je potrebno kopirati na sve čvorove. | Da, kao SQL | Da, kao SQL |
| Autorizacija | Da, uloge su definirane na razini sheme, a svaka pohranjena procedura definira koje je uloge mogu izvršavati. | Da, kao SQL | Da, kao SQL |

## VII. ZAKLJUČAK

Općenito govoreći, korištenje NewSQL-a je prikladno u scenarijima u kojima su korišteni tradicionalni SUBP, ali koji imaju dodatne zahtjeve za skalabilnošću i performansama.

NewSQL pohranjuje podatke prikladne za aplikacije koje zahtijevaju upotrebu transakcija koje manipuliraju sa više objekata od jednom, ili imaju jake zahtjeve za dosljednošću, ili čak oboje. Klasični primjer su aplikacije za financijsko tržište, gdje se postupci transfera novca moraju ažurirati na dva računa automatski i sve aplikacije moraju imati isti pogled na bazu podataka.

Relacijski model je prikladan u situacijama u kojim je struktura podataka poznata unaprijed i vjerojatno se neće mijenjati, što omogućuje fleksibilnost upita pomoću SQL-a [15], kao snažan mehanizam koji se može koristiti za provedbu gotovo bilo koje vrste rukovanja podatcima.

Konačno pri odabiru najprikladnijeg rješenja treba uzeti u obzir i prethodna ulaganja u alate te osposobljavanje osoblja. U tom je smislu NewSQL posebno atraktivan jer je kompatibilan sa većinom SUBP alata i koristi SQL kao glavni jezik interakcije [16].

Još je uvijek NewSQL mnogo manje popularan od NoSQL-a, djelom je razlog tome što je novina, ali i djelom što je relacijski pristup i fleksibilnost podataka kompleksno kombinirati. Konkurenciju NewSQL-u čini dobro poznati i uhodani SQL sa mnoštvom zadovoljnih korisnika, što je još jedan razlog težeg probijanja na tržištu [17].

U budućnosti će biti potrebno napraviti alate za migraciju tradicionalnih baza podataka u NewSQL, jer mnogi *startup* projekti teško mogu predvidjeti rast i potrebu za kapacitetima, te se u startu odlučuju za dobro poznata i pristupačna rješenja kao npr. MySQL koji u početku odlično odrađuje posao, no s povećanjem broja korisnika povećavaju se i zahtjevi te je moguće da sustav u jednom trenutku izgubi funkcionalnost. Tada je potrebno osigurati jednostavnu migraciju iz relacijskih u NewSQL sustave, po mogućnosti bez zastoja u radu. Kako se arhitektura NewSQL baza podataka razlikuje od distribucije do distribucije gotovo je nemoguće napraviti univerzalni alat za migraciju. Buduće istraživanje moglo bi ići u smjeru izrade alata za migraciju iz MySQL u VoltDB bazu podataka.

Zbog prioriteta prema performansama NewSQL baze podataka obično imaju sigurnosnih nedostataka koje je potrebno detaljno istražiti, posebno zaštitu podataka prilikom komunikacije korisnik-poslužitelj i poslužitelj-poslužitelj.

## LITERATURA


[1] Aslett, Matthew. „How Will The Database Incumbents Respond To NoSQL And NewSQL?". 451 Group , objavljeno 4.4.2011, Dostupno na: http://www.cs.cmu.edu/~pavlo/courses/fall2013/static/papers/aslett-newsql.pdf (20.9.2015)

[2] A B M Moniruzzaman. "NewSQL: Towards Next-Generation Scalable RDBMS for Online Transaction Processing (OLTP) for Big Data Management", Dostupno na: http://arxiv.org/ftp/arxiv/papers/1411/1411.7343.pdf (20.9.2015)

[3] Kumar R, Gupta N, Maharwal H, Charu S, Yadav K,. "Critical Analysis of Database Management Using NewSQL", International Journal of Computer Science and Mobile Computing Vol. 3 (str. 434-438) 2014

[4] Janković, Olivera."NoSQL dokument baza podataka: prikaz skladištenja podataka sa osvrtom na podatke sa senzora", Infoteh-Jahorina, INFOTEH-JAHORINA Vol. 14 (str. 561-566) 2015

[5] Solid IT, "Knowledge Base of Relationaland NoSQL Database Management Systems", Dostupno na: http://db-engines.com/en/ranking (10.10.2015)

[6] Venkat N Gudivada, Dhana Rao, Vijay Raghavan, „Renaissance in Data Management Systems: SQL, NoSQL and NewSQL" Dostupno na: http://www.researchgate.net/profile/Venkat_Gudivada/publication/268808575_Renaissance_in_Data_Management_Systems_SQL_NoSQL_and_NewSQL/links/55156d460cf2d70ee270272a.pdf(20.9.2015)

[7] "OLTP i OLAP sustavi" (2012) Dostupno na: http://imef.me/oltp-i-olap-sustavi/ (28.9.2015)

[8] Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., & Byers, A. H. "Big data: The next frontier for innovation, competition, and productivity", (2011)Dostupno na: http://www.mckinsey.com/insights/business_technology/big_data_the_next_frontier_for_innovation (28.9.2015)

[9] Strgačić, Mate "Razmišljajte široko Big Data problems" Dostupno na: http://www.infotrend.hr/clanak/2014/10/razmisljajte-siroko-big-data-problems,81,1096.html (1.10.2015)

[10] Wikipedia "NewSQL" url: https://en.wikipedia.org/wiki/NewSQL (20.9.2015)

[11] "ClustrixDB" url: http://www.clustrix.com/ (1.10.2015)

[12] "NuoDB" url: http://www.nuodb.com/ (1.10.2015)

[13] "VoltDB" url: https://docs.voltdb.com/ (1.10.2015)

[14] "MemSQL" url: http://docs.memsql.com/latest/ (1.10.2015)

[15] Redmond E, Wilson JR, "Seven databases in seven weeks: a guide to modern databases and the NoSQL movement". O'Reilly Media.978-1-934356-92-0, (2013)

[16] Grolinger K, Higashino W A, Tiwari A, Capretez M AM, “Data management in cloud environments: NoSQL and NewSQL data stores”, Jurnal of Cloud Computing, (2013)

[17] 16 NoSQL, NewSQL Databases To Watch ”NoSQL, NewSQL Databases” Dostupno na: http://www.informationweek.com/big-data/big-data-analytics/16-nosql-newsqldatabases-to-watch/d/d-id/1269559

[18] Wikipedia “SQL” Dostupno na: https://hr.wikipedia.org/wiki/SQL (7.10.2015)

[19] A Study of NoSQL and NewSQL databases for data aggregation on Big Data ”NoSQL NewSQL Databases” Dostupno na http://www.diva-portal.org/smash/get/diva2:706302/FULLTEXT01.pdf (1.10.2015)

[20] “Azure SQL Database” Dostupno na: https://msdn.microsoft.com/en-us/library/azure/ee336279.aspx (7.10.2015)

[21] “ClearDB” Dostupno na: https://www.cleardb.com/home.view (7.10.2015)

[22] “EnterpriseDB” Dostupno na: http://www.enterprisedb.com/ (7.10.2015)

[23] “Introducing JSON” Dostupno na http://www.json.org/ (10.10.2015)