Implementacija koncepata nerelacijskih baza podataka u sustavu MS SQL Server

Brkić, Barbara

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Barbara Brkić

IMPLEMENTACIJA KONCEPATA NERELACIJSKIH BAZA PODATAKA U SUSTAVU MS SQL SERVER

ZAVRŠNI RAD

Varaždin, 2020.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Barbara Brkić

Matični broj: 43993/15-R

Studij: Informacijski sustavi

IMPLEMENTACIJA KONCEPATA NERELACIJSKIH BAZA PODATAKA U SUSTAVU MS SQL SERVER

ZAVRŠNI RAD

Mentor:

Prof. dr. sc. Kornelije Rabuzin

Varaždin, rujan 2020

Barbara Brkić

Izjava o izvornosti

Izjavljujem da je moj završni/diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

U ovome radu cilj je prikazati implementaciju ne relacijskih koncepata (eng. *NoSQL*) u sustavu za upravljanje relacijskim bazama podataka (MS SQL Server). Na početku samog rada upoznajemo se s pojmom ne relacijskih baza podataka, početcima, vrstama istih i njihovim karakteristikama, a kasnije se uviđa potreba za usporedbom ne relacijskih i relacijskih baza podataka te se stavlja naglasak na modele podataka ne relacijskih baza. Nakon detaljnijeg opisa ne relacijskih koncepata upoznajemo se s MS SQL Server-om i načinom na koji su ne relacijski koncepti podržani u istom. Prikazom načina implementacije odabranih modela dolazimo do kraja rada.

Ključne riječi:

Baza podataka (BP) – organizirani skup podataka

Structured Query Language (SQL) – jezik za rad s relacijskim bazama podataka

Ne relacijske baze podataka (NoSQL) – baze podataka koje ne počivaju na relacijskom model

Sadržaj

1. Uvod

Gdje si sada, a gdje ćeš biti za 5 godina? Pitanje na koje nitko ne zna odgovor, ali jedino što zna je da će biti stariji, iskusniji i pametniji. Tako nekako možemo opisati i razvoj tehnologije. Nitko nije znao kakva će biti, ali svi smo znali da će uznapredovati. Ključno je da nakon ostvarenja jednog cilja postavimo novi - onaj koji će poboljšati dosadašnji i tako rastemo iz dana u dan, od cilja do cilja, od jednostavnog do kompleksnog.

Vjerujem da su se i stručnjaci iz Microsofta[1] vodili ovim načelom kada su odlučili implementirati rad s nerelacijskim konceptima baza podataka u jedom potpuno relacijskom SQL Server-u. Napredovati, implementirati nešto potpuno novo i drugačije, a s druge strane opstati na tržištu i širiti područja korištenja. Kako se broj korisnika i zahtjeva povećavao tako su se s vremenom počeli javljati i nedostaci relacijskih baza podataka. Novonastale ne relacijske BP nametnule su se kao bolja solucija sustava za pohranu podataka, a modeli podataka koji variraju od grafova, dokumenata pa sve do tablica čine ih pogodnim za veći broj različitih vrsta korisnika.

Tako je nastala tema mojeg završnog rada. Kako su stručnjaci iz Microsofta implementirali ne relacijske koncepte u MS SQL Serveru i koje su prednosti, a koji su nedostatci korištenja takvih koncepata? U ovom radu opisat ću ne relacijske baze podataka i njene koncepte, MS SQL Server i njegovu povijest te na kraju prikazati kako su ne relacijski koncepti implementirani u relacijskom obliku tj. kako se s njima radi. Na kraju rada bit će iznesen zaključak u kojem ću prokomentirati zapažanja koja sam uočila za vrijeme istraživanja ove teme.

[1] Microsoft Corporation – tehnološka tvrtka koja proizvodi, razvija i prodaje računalni softver, računala etc.

2. Metode i tehnike rada

Svrha ovog rada je prikazati modele ne relacijskih baza podataka i način implementacije istih u MS SQL Server-u. Za uređivanje teksta ovog rada korišten je program Microsoft Office Word dok su za obradu i izradu slika korišteni programi Paint i Adobe XD. Kako do sada nisam bila upoznata s NoSQL bazama podataka prvi korak u pisanju ovog rada bilo je istraživanje istih. Za početak sam analizirala članke o NoSQL bazama koje su objavile kompanije kao što su IBM i Microsoft, a za bolji uvid proučila sam i različite portale kao što su: GeeksforGeeks, Datavesity, MSSQLtips i slični. Bilo je potrebno uvidjeti ono što je bitno i ono što je relevantno te su zbog toga glavni izvori bile baš web stranice tih kompanija.

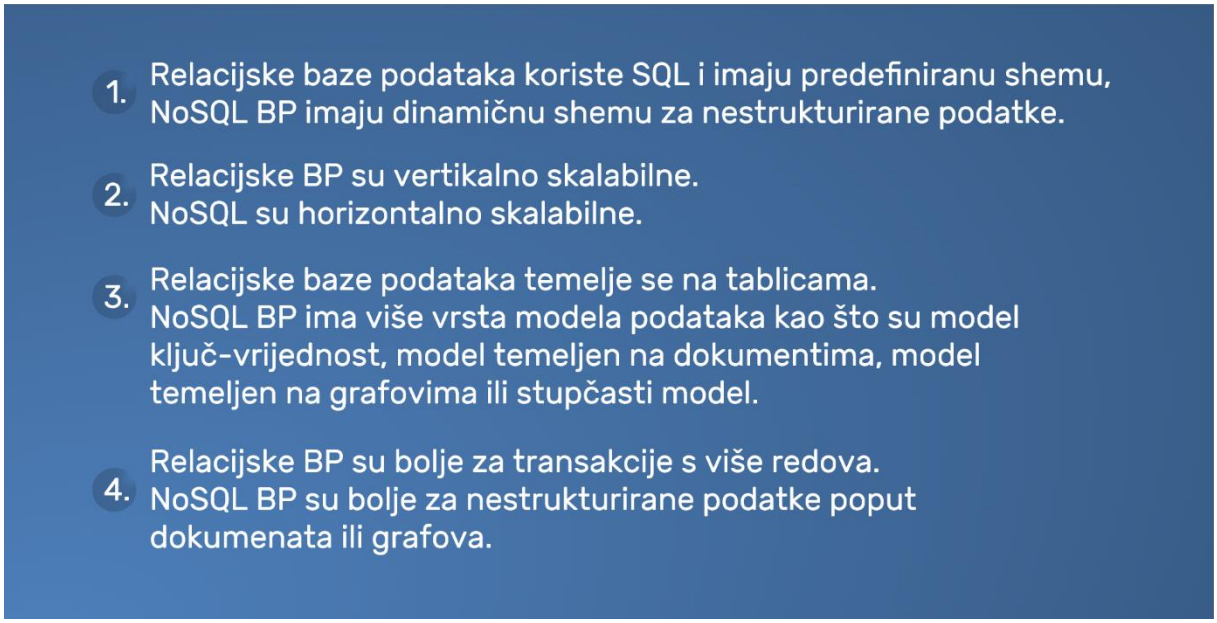
Važne informacije i svojstva relacijskih baza podataka proučila sam iz knjiga prof. Rabuzina koje spadaju pod obaveznu literaturu na fakultetskim predmetima, a ključne razlike između relacijskih i ne relacijskih baza podataka saznala sam sa stranice MongoDB-a. Za vrijeme istraživanja sustava za upravljanje ne relacijskim bazama podataka naišla sam na mnoge fotografije koje nisu točno prezentirale vrste SUBP te sam zbog toga odlučila sama izraditi slike s popisom sustava koji odgovaraju određenom modelu. S obzirom na to da je u radu bilo potrebno implementirati određene nerelacijske koncepte u sustav MS SQL Server za izradu praktičnog dijela rada korišteni su programi Microsoft SQL Server 2019 i Microsoft SQL Server Management Studio. Za izradu ERA modela korišten je program MySQLWorkbench dok je za izradu skice grafa korišten program Visual Paradigm.

Jedan od izvora informacija, Microsoft-ova online emisija Data Exposed koja se u više nastavaka bazira na rad s JSON datotekama u Server-u pomogla mi je da uvidim način korištenja i osnovne naredbe za rad s JSON formatom, a način izrade BP u skladu sa grafovskim modelom proučila sam na službenim stranicama kompanije Microsoft.

Ono što me je iznenadilo u samom procesu istraživanja bila je lakoća kretanja kroz Microsoft-ovu dokumentaciju. Teme su dobro organizirane i jednostavno sam mogla pristupiti temi koja me je zanimala, a s obzirom na to da se na kraju svake teme nude i relevantni članci/video uradci uvijek bih naišla na neki dodatni sadržaj.

3. Ne relacijske baze podataka (*NoSQL*)

Pojam „NoSQL“ prvi put se pojavio 1998. godine, a koristio se za opisivanje relacijske BP koja nije pružala nikakav oblik SQL jezika za postavljanje upita (zbog toga dolazi do naziva No SQL). Takav model BP razvio je Carlo Strozzi i nazvana je Strozzi NoSQL.[1] Ipak NoSQL koji danas poznajemo podosta se razlikuje od Strozzijevog, a sada iza pojma NoSQL stoji novo značenje: Not only SQL što je u prijevodu; nije samo SQL. Ne relacijske baze podataka služe za pohranu i rad s nestrukturiranim i polu strukturiranim podacima i koriste UnQL[2] jezik sličan SQL-u. Postoji nekoliko ključnih razlika između relacijskih i ne relacijskih BP koje dobro opisuju same baze podataka i njihove karakteristike. [2]



Slika 1. Razlike između SQL-a i NoSQL-a (autorski rad) [2]

3.1. Karakteristike

Iako su relacijske baze podataka raširenije u upotrebi, NoSQL baze podataka imaju specifične karakteristike zbog kojih su preduvjet za izgradnju moćnih aplikacija. Tako su neki od najpoznatijih softvera koji koriste NoSQL baze podataka navedeni u tablici 1 u nastavku. Karakteristike koje daju prednost NoSQL bazama podataka nad relacijskim bazama podataka bit će opisane u idućim poglavljima.

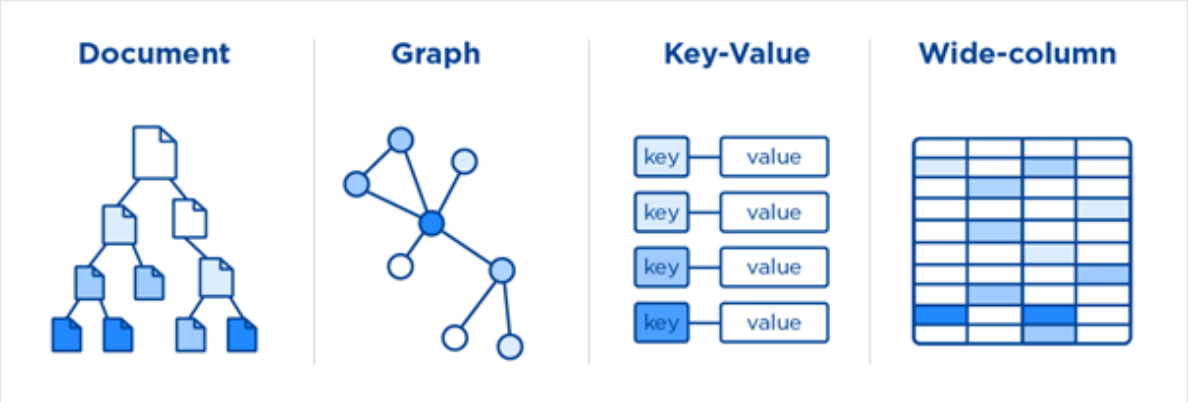
Tablica 1. Aplikacije koje koriste NoSQL baze podataka (autorski rad) [3] [4] [5]

[2] UnQL – standardizirani jezik za ne relacijske baze podataka

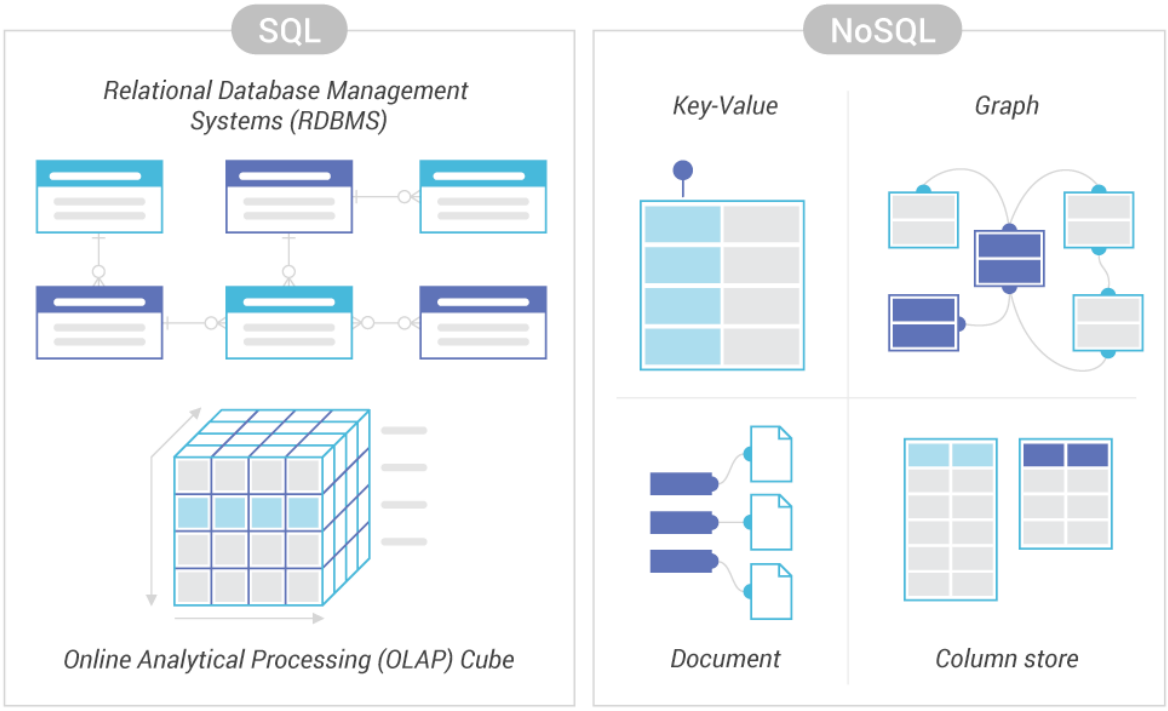
Naziv aplikacije	Sustav za upravljanje NoSQL BP	Model BP koji sustav koristi
Facebook razmjena poruka (inbox)	Cassandra	Stupčasti model
Amazon	Amazon DynamoDB	Ključ-vrijednost i dokument
Google Mail	Bigtable	Ključ-vrijednost
LinkedIn	Espresso	Dokument
Twitter	FlockDB	Model temeljen na grafovima

3.1.1. Modeli podataka

Svi znamo da su relacijske baze podataka kolekcije „podataka kod kojih se podaci nalaze u relacijama, odnosno tablicama“. [6, str. 4]. Suprotnost relacijskih su ne relacijske BP kod kojih postoji više vrsta modela. Neki od modela su: model ključ-vrijednost, model temeljen na dokumentima, model temeljen na stupcima i model grafova.[7] Kako za ne relacijske BP ne postoji samo jedna vrsta SUBP (kod relacijskih ima više različitih proizvođača dok je sami softver u suštini isti) stoga za svaku vrstu modela podataka postoji više različitih SUBP. Tako ću u poglavlju 3.2. detaljno objasniti svaki model i spomenuti najpoznatije SUBP za pojedini model.



Slika 2. Prikaz modela ne relacijskih BP [8]



Slika 3. Usporedba modela relacijskih i ne relacijskih BP [9]

3.1.2. Skalabilnost

Kako bi lakše shvatili što je uopće skalabilnost i koja je razlika u skalabilnosti relacijskih i ne relacijskih BP, prvo ćemo iskazati njenu definiciju. Skalabilnost je svojstvo sustava da podnosi sve veću količinu posla dodavanjem resursa u sustav. Tako možemo reći da skalabilnost baze podataka označava „spremnost“ BP na izvršavanje dodatnog posla s obzirom na rast broja čimbenika kao što su korisnici, podaci i sl. [10]

S obzirom na svakodnevni porast broja korisnika i popratnih podataka koje je potrebno pohraniti, nužno je planirati način same pohrane tih istih podataka na serverima. U tom slučaju razlikujemo dva pristupa: horizontalno i vertikalno skaliranje. Horizontalno skaliranje odnosi se na dodavanje novih servera i pohranjivanje podataka na više različitih servera, dok se vertikalno skaliranje odnosi na dodavanje novih resursa na jednom serveru. Uz pomoć slike 4 prikazat ću jednostavnije objašnjenje skalabilnosti. Bitno je naglasiti da relacijske baze podataka koriste vertikalnu skalabilnost koja je ranjiviji oblik skalabilnosti u slučaju pojave neočekivanih grešaka, dok ne relacijske BP koriste horizontalnu skalabilnost. U tom slučaju pri pojavi većeg broja novih korisnika ili pojavi velikih količina podataka i slično valja samo dodati novi server u trenutnu skupinu servera.

Prednost horizontalne skalabilnosti je ta što u slučaju „pada“ servera / pojave neke greške još uvijek možemo pristupiti podacima pomoću preostalih servera iz klastera[3].

Slika 4. Prikaz horizontalne i vertikalne skalabilnosti [11]

3.1.3. UnQL

Svi smo upoznati sa SQL-om (eng. Structured Query Language) koji je standardizirani jezik za relacijske baze podataka, no bilo bi vrijeme da upoznamo glavnog aktera ne relacijskih baza podataka. To je UnQL (eng. Unstructured Data Query Language) nestrukturirani upitni jezik koji koriste ne relacijske baze podataka. Razvijen je u suradnji Couchbasea[4] i tvoraca SQL Litea[5] sa zajedničkim ciljem da se za ne relacijske baze podataka uredi standardizirani jezik za definiranje i manipulaciju podatcima.[12] Iako je namijenjen ne relacijskim bazama podataka UnQL sadrži i neke konstrukte samog SQL-a. U konačnici UnQL se može smatrati inačicom SQL-a no s fokusom na kolekcije i dokumente za razliku od tablica i redaka. Još nešto po čemu se UnQL razlikuje od SQL-a je to što ne nudi tipične DDL funkcionalnosti kao što su CREATE, DROP TABLE i DROP INDEX. U NoSQL bazama podataka nije potrebno koristiti CREATE naredbu kako bi se kreirale kolekcije već je potrebno samo navesti naredbu INSERT INTO ispred naziva kolekcije u koju spremamo podatke [13]. U nastavku će biti prikazan primjer UnQL jezika gdje se sprema objekt u kolekciju naziva *nosql_collection*.

[3] Klaster servera – grupa servera koji zajednički omogućuju rad sustava i pružaju korisnicima veću dostupnost
[4] Couchbase Inc. – privatna tvrtka koja razvija i nudi podršku za Couchbase Server i Couchbase Lite (open-source NoSQL multi-model softveri)
[5] SQL Lite - sustav za upravljanje relacijskim bazama podataka

```
INSERT INTO nosql_collection VALUE
   {
        type: „nested“,
        content:
                {
                        Content: „document object“,
                        docNumber: 1,
                        articleContent: {str: „Dataversity“,str2:“Article“},
                        newArticle: true
                }

   };
```

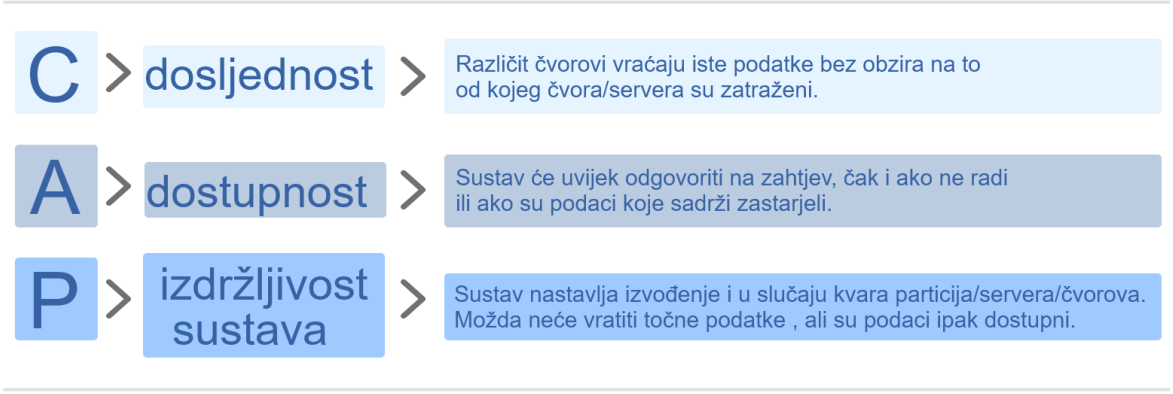
Kod izrađen pomoću izvora [13].

3.1.4. ACID, CAP & BASE

Kod relacijskih BP naglasak je stavljen na ACID svojstva transakcija dok se NoSQL BP baziraju na CAP teoremu, a same transakcije su u skladu s BASE principom . U ovom poglavlju ću objasniti ACID, CAP i BASE koncepte te napraviti kratki osvrt. ACID (Atomicity = atomnost, Consistency = dosljednost, Isolation = izolacija, Durability = postojanost) se odnosi na standardizirani skup svojstva koja opisuju BP i jamče ispravnu obradu transakcija. Znači ako je transakcija izvršena, onda su zadovoljena ACID svojstva. Pomoću slike u nastavku svojstva će biti ukratko objašnjena.

Slika 5. ACID svojstva (autorski rad) [14]

CAP teorem (Consistency = dosljednost, Availability = dostupnost, Partition tolerance= izdržljivost sustava) je koncept koji se odnosi na svojstva distribuiranih sustava[6]. Teorem napominje da je nemoguće da distribuirana pohrana podataka pruži sva tri svojstva istovremeno (nužan odabir dva od tri svojstva). Ovisno o odabiru svojstva nastaje baza podataka koja odgovara traženim zahtjevima. Cap teorem navodi sljedeća 3 svojstva: dosljednost, dostupnost i izdržljivost sustava (ukratko objašnjeno na slici).



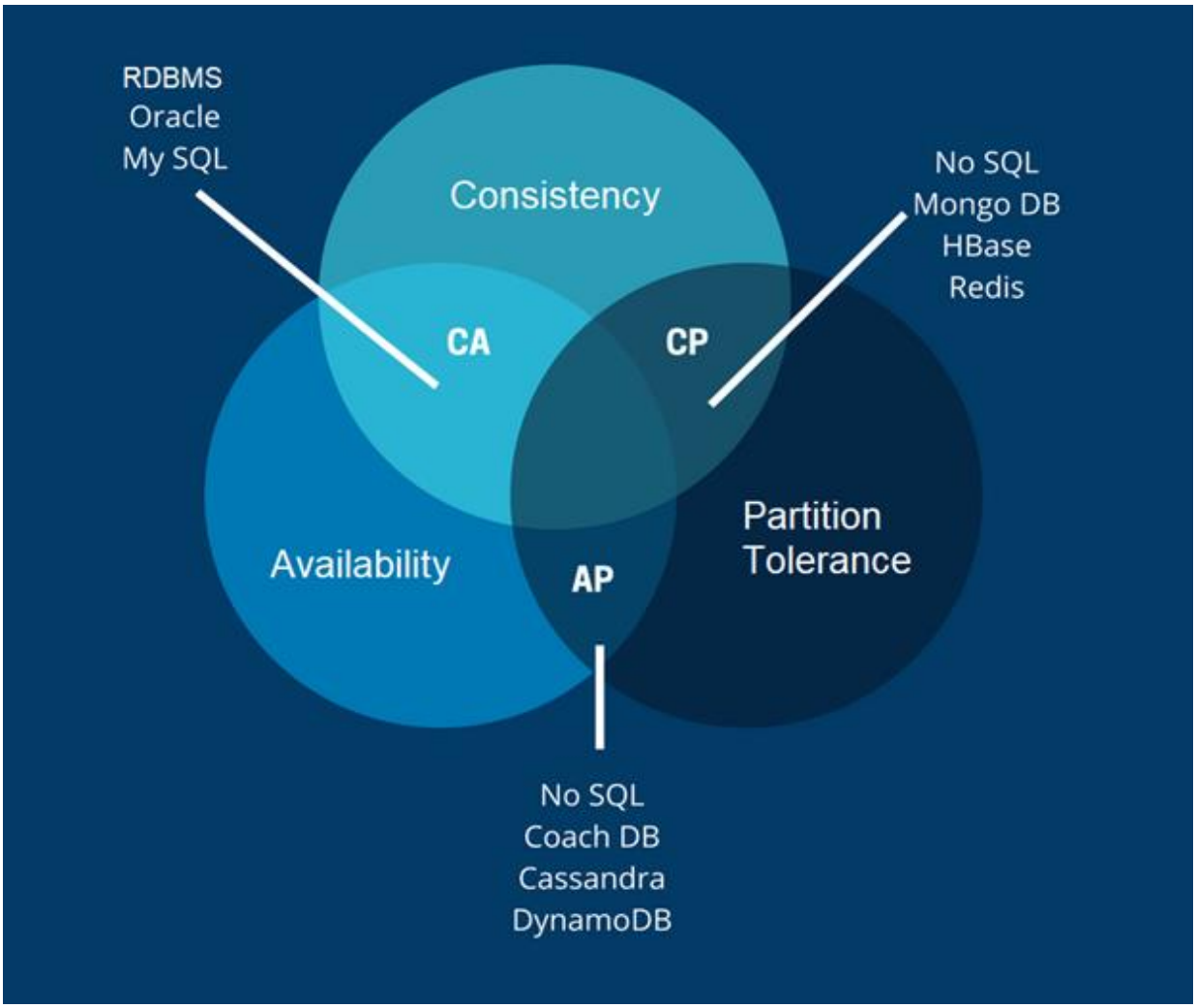
Slika 6. CAP svojstva (autorski rad) [15]

S obzirom na dosad izrečeno, shvaćamo da je kreiranje baze podataka važan posao, a arhitektima je dana mogućnost kreiranja AP, CP ili CA vrste sustava. Tako AP (availability and partition tolerance) predstavlja *dostupnost* i izdržljivost baze podataka, CP (consistency and partition tolerance) *dosljednost* i izdržljivost baze podataka, a CA (consistency and availability) *dostupnost* i *dosljednost* baze podataka.[16]

- **AP** (dostupnost i izdržljivost sustava) – sustav će uvijek obraditi zahtjev klijenta i pokušati vratiti najnoviju dostupnu verziju podataka, čak i ako ne može jamčiti da je aktualna
- **CP** (dosljednost i izdržljivost sustava) – sustav će vratiti grešku ako se određeni podaci ne mogu ažurirati na druge čvorove zbog kvara (nekonzistentni čvor se „gasi“ tj. sustav ga čini nedostupnim)
- **CA** (dosljednost i dostupnost) -takva baza podataka pruža dosljednost i dostupnost u svim čvorovima, no ako postoji particija između bilo koja dva čvora u sustavu onda se to ne može jamčiti i greške se ne toleriraju [17]

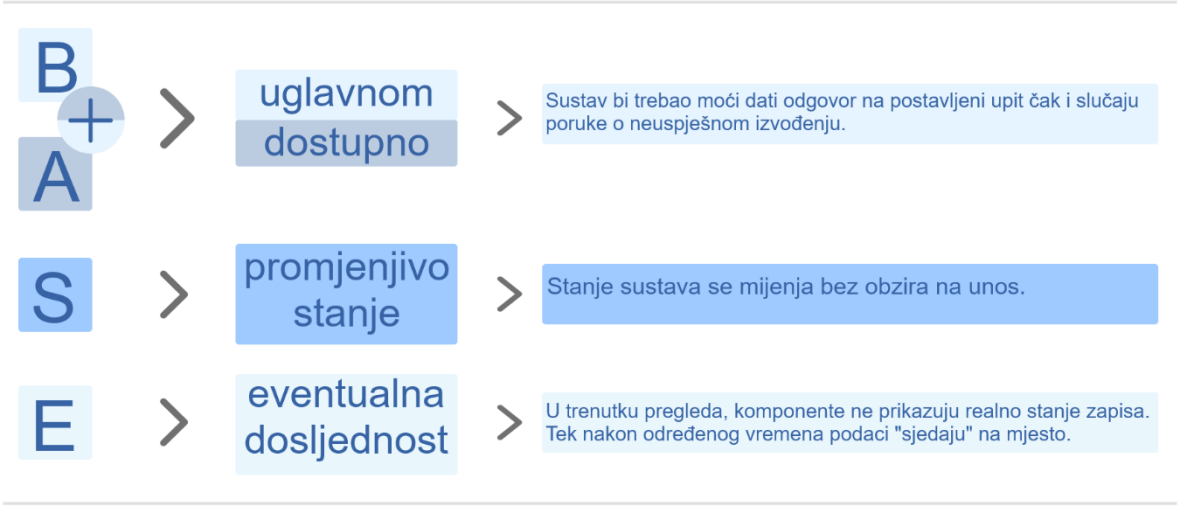
[6] Distribuirani sustav – skup komponenata smještenih na različitim računalima koji međusobno komuniciraju i koordiniraju akcije s ciljem da se krajnjem korisniku prikaže jedinstveni koherentni sustav.

Na slici 7 prikazan je CAP teorem i sustavi za upravljanje bazama podataka (relacijskim i ne relacijskim). Također prikazana je podjela sustava prema vrsti CA/CP/AP što daje bolju predodžbu o kojoj se vrsti sustava radi i koje su njegove prednosti.



Slika 7. CAP teorem i SUBP [18]

Nasuprot relacijskom ACID-u, BASE (Basically-Available = uglavnom dostupno, Soft-State = promjenjivo stanje, Eventual consistency = eventualna dosljednost) se koristi kako bi lakše opisali svojstva ne relacijskih baza podatka. Ono što BASE omogućuje je brzina i dostupnost na uštrb snažne dosljednosti koju pruža ACID. Na slici u nastavku možemo vidjeti da su BASE svojstva suprotna ACID svojstvima.



Slika 8. BASE (autorski rad) [19] [20]

Možemo reći da ACID svojstva zahtijevaju dosljednost nakon izvršetka svake operacije dok BASE svojstva omogućuju i kašnjenje promjena (dovodi do prikaza zastarjelih podataka). „Na primjer, distribuirani sustav održava kopije zajedničkih podataka na više računala u klasteru kako bi osigurao visoku dostupnost. Kada se podaci ažuriraju u klasteru, može postojati određeni vremenski interval tijekom kojeg će se neke kopije ažurirati, ali druge neće.“[17] Cijeli proces se svodi na to da će se promjene s vremenom proširiti na preostale strojeve, ali s naglaskom na vremensko odstupanje. Zbog takvog načina ponašanja sustava dolazi i do naziva eventualna dosljednost.

Kada smo saželi sve modele, možemo reći da su u relacijskim bazama podataka ACID svojstva više usmjerena na sigurnost podataka dok BASE ograničuje dosljednost kako bi omogućio obradu zahtjeva čak i u nedosljednom stanju. NoSQL BP minimizirale su zahtjeve za trenutnom dosljednošću, realnim i preciznim prikazom podataka kako bi se dobile druge prednosti poput skalabilnosti (poglavlje 2.1.2.) i otpornosti (mogućnosti oporavka BP).

3.2. Modeli podataka

Naspram relacijskih baza podataka koje se temelje na relacijskom modelu takozvanom modelu tablica, ne relacijske baze podataka nisu orijentirane na samo jedan model već postoji više vrsta modela podataka. Za svaku „vrstu“ NoSQL BP tj. za svaki model podataka postoji i poseban sustav za upravljanje istima. Zbog toga na tržištu postoji puno više sustava za upravljanje ne relacijskim bazama podataka od SURBP[7].

[7] SURBP- Sustav za upravljanje relacijskim bazama podataka

Glavna su četiri modela: model ključ-vrijednost, model temeljen na dokumentima, model temeljen na stupcima i model grafova.

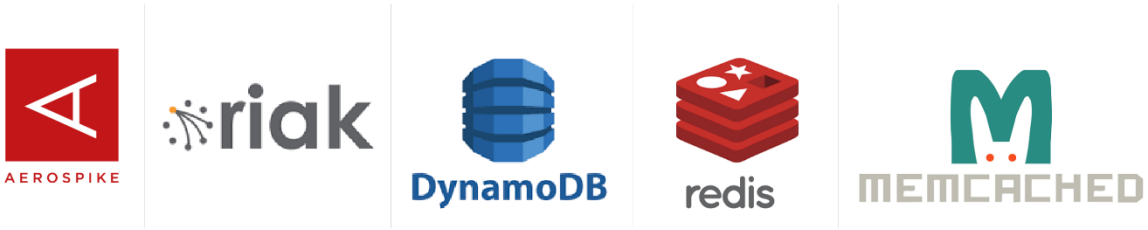
Slika 9. Broj SUBP u svijetu (autorski rad) [21][22]

3.2.1. Model ključ-vrijednost (eng. *Key-value store*)

Model ključ-vrijednost najjednostavniji je od četiri modela te zbog toga prednjači u broju korisnika. Neke od bitnijih karakteristika su [7][23][24]:

- Podaci su prikazani kao kolekcija parova jedinstvenog ključa i pripadajuće vrijednosti
- Svaki se ključ pojavljuje samo jednom u kolekciji (*unique key*)
- Vrijednosti se mogu spremiti u obliku slova, brojeva, niza ili u JSON formatu
- Podaci se pretražuju prema vrijednosti ključa
- Ne koristi UnQL već naredbe GET, PUT, DELETE
- Koristi BASE arhitekturu
- Često se koriste se za potrebe pohrane velike količine podataka nad kojima nije potrebno izvršavati složene upite

Baze podataka temeljene na modelu ključ-vrijednost svoju prednost očituju kroz mogućnost skladištenja velike količine podataka u jednostavnoj strukturi (oblik ključ-vrijednost) te zbog toga pohrana složenijih struktura kao što su grafovi nije moguća. SUBP temeljeni na modelu ključ-vrijednost su:



Slika 10.SUBP bazirani na modelu ključ-vrijednost (autorski rad)

3.2.2.Model temeljen na dokumentima (eng. *Document based*)

Baze podataka temeljene na modelu dokumenata nazivaju se i skladišta dokumenata, a neke od najbitnijih karakteristika su [7][23][24]:

- Model podataka je složenija kolekcija parova ključ-vrijednost (na kraju čine dokument)
- Podaci su pohranjeni u obliku dokumenta (formati XML[8], JSON[9], BSON[10])
- Vrijednosti se mogu spremiti u obliku slova, brojeva, nizova, objekata …
- Koriste UnQL
- Mogućnost pohrane velikog broja atributa u kombinaciji s velikom količinom podataka

SUBP temeljeni na modelu dokumenta su:

Slika 11. SUBP bazirani na modelu dokumenta (autorski rad)

[8] XML – eng. Extensible Markup Language – koristi se za pohranu i organizaciju podataka u obliku teksta
[9] JSON – eng. JavaScript Object Notation - standardni tekstualni format za predstavljanje strukturiranih podataka na temelju JavaScript objektne sintakse
[10] BSON – eng. Binary JSON- binarni oblik JSON formata

3.2.3.Model temeljen na stupcima (eng. *Wide-column based*)

Baze podataka temeljene na modelu stupaca pohranjuju podatke u tablice, redove i dinamične stupce, a u odnosu na relacijske BP pružaju veću fleksibilnost jer nije nužno koristiti iste stupce u svakom redu. U nastavku slijede neke od karakteristika BP temeljenih na modelu stupaca [23][24]:

- Podaci su organizirani po stupcima
- Koristi višedimenzionalno mapiranje (*row-value, column-value, timestamp*)
- Upiti dohvaćaju podatke brže nego u klasičnim relacijskim BP
- Nisu optimizirane za spajanje (u SURBP - JOIN)
- Najčešće se koriste u slučaju potrebe za pohranom velikih količina podataka kao što su podaci korisničkih računa i slično

SUBP temeljeni na modelu stupaca su:

Slika 12. SUBP bazirani na modelu širokih stupaca (autorski rad)

3.2.4.Model temeljen na grafovima (eng. *Graph based*)

Baze podataka temeljene na modelu grafova fokusirane su na odnos dva različita entiteta. U nastavku slijede neke od karakteristika :

- Model se temelji na teoriji grafova
- Podaci se pohranjuju pomoću čvorova, bridova i svojstva
- Zbog jednostavne vizualne predodžbe najčešće se koriste za izradu modela u slučaju prijevara i određivanju udaljenosti između više lokacija, a koriste ih i popularne društvene mreže kao što su Facebook, Twitter pa čak i Google

SUBP temeljeni na modelu grafova su:

Slika 13. SUBP bazirani na modelu grafova (autorski rad)

3.3. Usporedba relacijskih i ne relacijskih BP

U ovom poglavlju ukratko ću usporediti glavne značajke relacijskih i ne relacijskih baza podataka kako bi istaknuli ključne razlike u samim sustavima. Neke od značajka koje ću detaljnije pojasniti su:

1. Model pohrane podataka
2. Shema BP
3. Skalabilnost
4. Razvojni model
5. Jezik BP
6. Transakcije
7. Dosljednost podataka

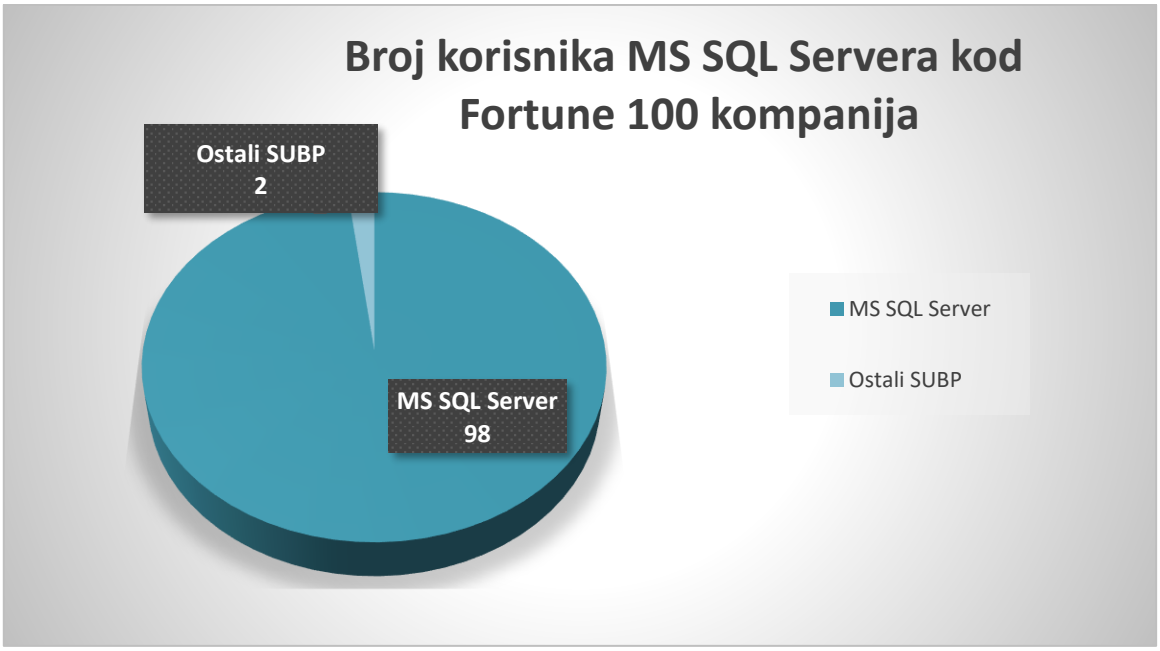
Tablica 2. Usporedba relacijskih i ne relacijskih BP [23][25]

	Relacijske BP	**Ne relacijske BP**
Model pohrane podataka	Jedan model	Više modela
	Podaci koji se unose pohranjuju se u obliku tablica. Svaka tablica sadrži atribute i njihove vrijednosti te se pohranjuje i tip podataka. Kod izvođenja upita može se spajati više tablica povezanih istim atributom.	Postoji četiri modela podataka: model ključ-vrijednost, model temeljen na dokumentima, model temeljen na stupcima i model temeljen na grafovima, a same podatke možemo spremati kao skup parova ključ-vrijednost, u XML ili JSON formatu ili u čvorove i veze od kojih nastaje graf.
Shema	Statična	Dinamična
	Shemu baze podataka i same tipove podataka koji će se koristiti potrebno je unaprijed odrediti. Ukoliko dodajemo novu vezu ili novi atribut shema se u cijelosti mijenja. Za vrijeme uređenja sheme, baza podataka je nedostupna za korisnike.	S obzirom na to da se podaci ne pohranjuju u tablice struktura podataka je dinamična što omogućuje dostupnost korisnicima i za vrijeme mijenjanja sheme.

	Relacijske BP	Ne relacijske BP
Skalabilnost	Vertikalna	Horizontalna
	Vertikalno skaliranje odnosi se na dodavanje novih resursa na jednom serveru - to znači da je potreban server koji može pohraniti velike količine podataka. Niska tolerancija sustava na moguće greške.	Horizontalno skaliranje odnosi se na dodavanje novih servera i pohranjivanje podataka na više različitih servera – to znači da su podaci pohranjeni na većem broju različitih servera. Veća tolerancija sustava na moguće greške.
Razvojni model	Postoje i komercijalne i *open source* baze podataka.	Sve ne relacijske baze podataka su *open source.*
Jezik BP	Relacijske baze podataka koriste SQL naredbe pri kreiranju upita.	Neke ne relacijske baze podataka koriste UnQL pri kreiranju upita dok druge koriste set svojih specifičnih naredbi.
Transakcije	ACID	CAP pomoću BASE-a
	Kod relacijskih baza podataka temeljni model izvođenja transakcija je ACID. Naglasak je na dosljednost podataka naspram dostupnosti.	BASE model karakterističan je za ne relacijske baze podataka i predstavljen je kao optimističan oblik ACID-a. Naglasak je stavljen na dostupnost naspram dosljednosti.
Dosljednost podataka	Kako po CAP teoremu relacijske baze podataka zadovoljavaju CA svojstva (dosljednost i dostupnost) tako su dosljedne u bilo kojem trenutku.	Kako ne relacijske BP po CAP teoremu mogu biti ili CP (dosljednost i izdržljivost sustava) ili AP (dostupnost i izdržljivost sustava) tako možemo reći da neki sustavi (CP) pružaju dosljednost podataka dok drugi(AP) pružaju eventualnu dosljednost.

4. SQL Server

Microsoft SQL Server najčešće zvan samo SQL server proizvod je tvrtke Microsoft i nastao je davne 1989. godine. SQL Server je sustav za upravljanje relacijskim bazama podataka (SURBP), a primarna zadaća mu je dohvaćanje i pohranjivanje podataka. Da bi bolje shvatili sami SQL Server ukratko ću opisati što je to uopće SUBP. Do sada smo naučili kako je baza podataka skup međusobno povezanih podataka koji čine neku cjelinu, a sami podaci su dostupni korisnicima te aplikacijama koje ih koriste. Kako korisnici ne bi morali znati detalje baze podataka da bi izveli neku jednostavnu radnju poput upisivanja, brisanja ili čitanja podataka nastali su SUBP koji im to olakšavaju. „Sustav za upravljanje bazom podataka je poslužitelj (server) baze podataka. On oblikuje fizički prikaz baze u skladu s traženom logičkom strukturom. Također, on obavlja u ime klijenta sve operacije s podacima“.[26, str 2.] Zaključno tome možemo reći da je SUBP poslužitelj baze tj. program koji omogućuje bazi podataka da pruža svoje usluge drugim programima/računalima, a definiran je modelom klijent-poslužitelj. U sadašnjosti MS SQL Server raste u svim pogledima, od novih verzija pa sve do povećanja profita, a samo prestanak pružanja tehničke podrške za SQL server 2008 utjecao je čak i na 10% -tni porast prihoda novijih verzija kao što su SQL Server 2017 i SQL Server 2019. [27] Na sljedećoj slici možemo vidjeti graf koji predstavlja podatke preuzete sa službene stranice Microsofta, a odnosi se na broj kompanija koje koriste MS SQL Server, a da se nalaze na Fortune 100 ljestvici kompanija.



Slika 14. Broj kompanija koje koriste MS SQL Server (autorski rad) [28]

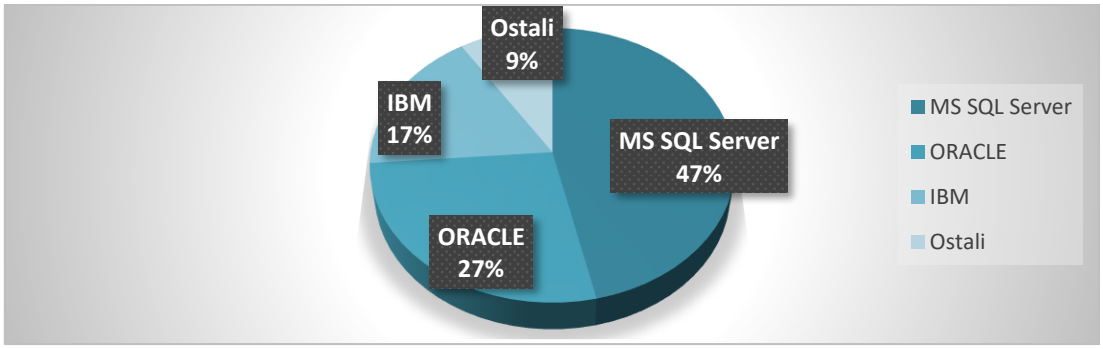
4.1. Upotreba MS SQL Server-a tijekom godina

U ovome poglavlju opisat ću početke SUBP-a i prikazati razinu popularnosti MS SQL Server-a u zadnjih 10 godina. Na početku ću iznijeti povijest nastanka sustava za upravljanje bazama podataka, a nakon toga ću se orijentirati na položaj Server-a naspram konkurencijskih SUBP-a.

Sve je počelo u ranim 1960-ima kada je Charles W. Bachman pod okriljem tvrtke General Electric dizajnirao prvi SUBP tada zvan IDS (Integrated Data Store). Ne želeći zaostajati za konkurencijom, neposredno nakon toga (krajem '60-ih) IBM predstavlja svoj sustav za upravljanje bazama podataka poznat i kao IMS (Information Management System).

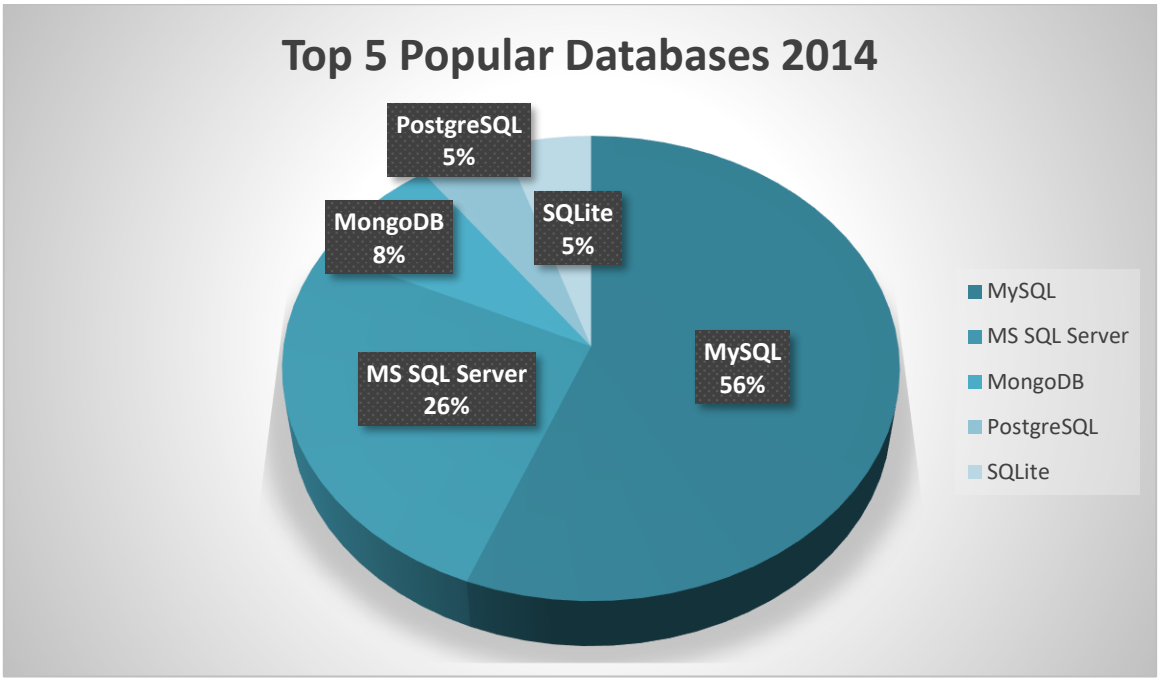
1970-ih Edgar F. Codd (djelatnik IBM-a) razvija novi model baza podataka kao suprotnost tadašnjem sistemu. Novi model takozvani relacijski model, zagovara pretraživanje podataka prema sadržaju nasuprot tradicijskog koji se odnosio na pretraživanje sadržaja sljedeći poveznice.[29] S obzirom na to da je IBM uložio podosta u IMS, Codd-ove ideje nisu bile dobro prihvaćene. Srećom 1973. godine Codd-ove ideje o relacijskom modelu opazili su stručnjaci Michael Stonebraker i Eugene Wong te donose odluku o provođenju istraživanja relacijskih baza podataka. Naziv projekta bio je INGRES (Interactive Graphics and Retrieval System) i uspješno je demonstrirao praktičnost i učinkovitost relacijskog modela. INGRES kao tadašnji SUBP koristio je upitni jezik QUEL, a zbog mogućnosti za većim ulaganjem u relacijske baze tvorci INGRES-a uputili su apel IBM-u da razvije standardni upitni jezik (SQL) koji već '80-ih godina postaje ANSI i OSI standard i time zamjenjuje dotadašnji QUEL. [30] Komercijalizacija INGRES-a dovodi do nastajanja novih SUBP kao što su Postgres, MS SQL Server i slični. Pola stoljeća kasnije relacijske BP čine više od 50% tržišta BP.[31] Nova konkurencija su NoSQL BP, ali i ostali SURBP koji čine veliku tržišnu konkurenciju.

Godine 2012-te MS SQL Server je vodeći SUBP na tržištu, a slijede ga Oracle i IBM. U to vrijeme na tržište je plasirana nova verzija MS SQL Server 2012 koja omogućuje poboljšanje dostupnosti baze podataka i nove modificirane prikaze dinamičkog upravljanja. „Microsoft postaje lider u udjelu upotrebe poslužitelja(Server-a) i smanjuje cijene“.[32]



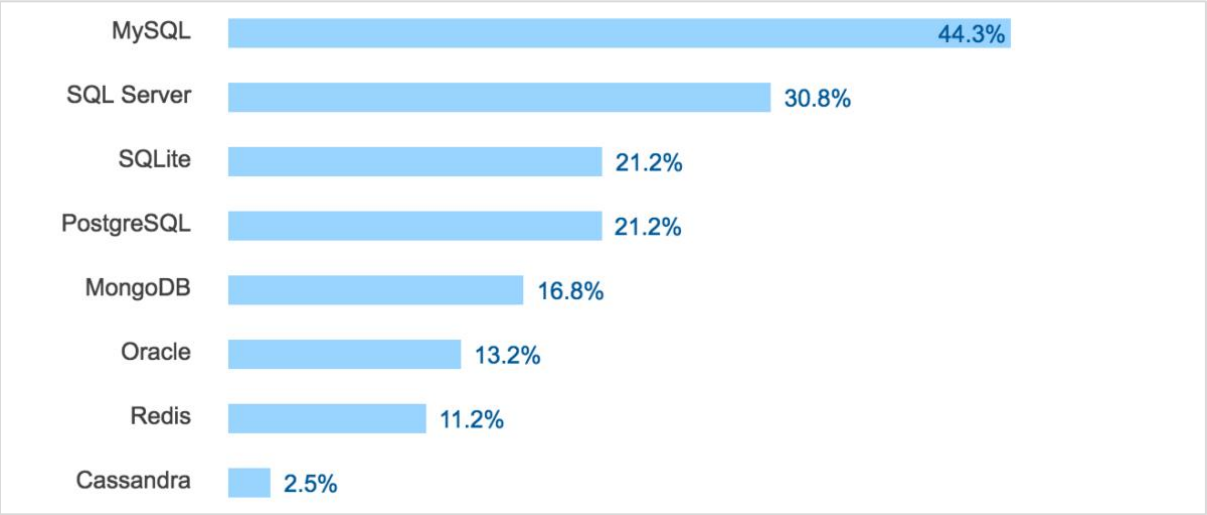
Slika 15. Pozicioniranje MS SQL Server-a na tržištu 2012. godine [32]

S obzirom na ubrzani razvoj i kompetitivnost konkurenata već samo dvije godine nakon možemo vidjeti da se smanjio broj korisnika MS SQL Server-a.



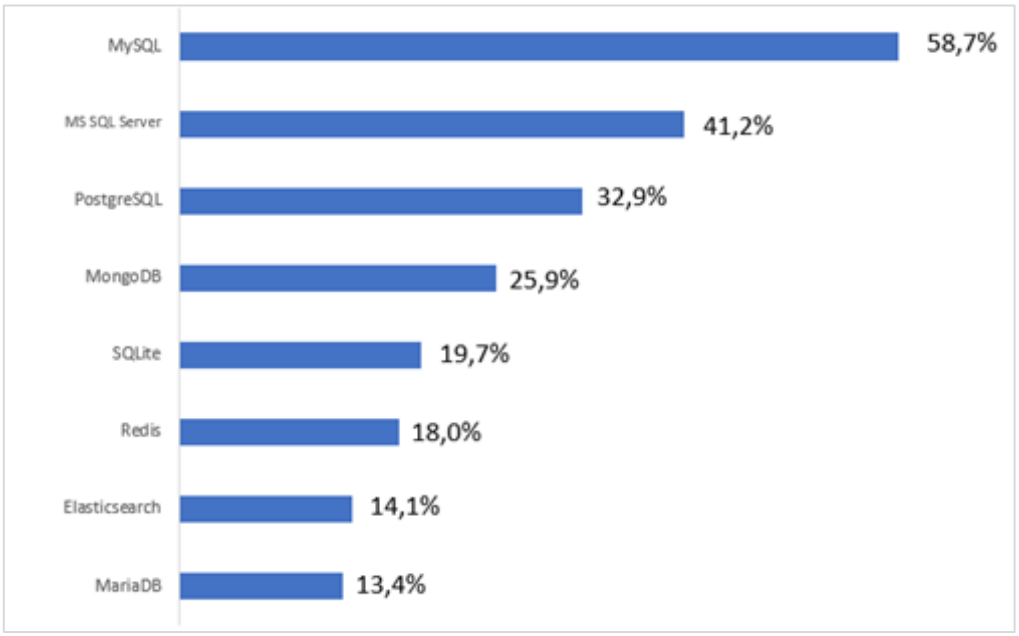
Slika 16. Pozicioniranje MS SQL Server-a na tržištu 2014. godine [33]

Možemo vidjeti da je tada prednjačio MySQL, ali i da je MS SQL Server bez obzira na postotni broj ipak drugi najpopularniji SUBP. Nekoliko godina kasnije možemo vidjeti da je MS SQL Server još uvijek na drugom mjestu po uporabi, ali da se postotni broj povećao. Također uviđamo da je postotak korisnika MySQL SUBP-a pao s obzirom na rast postotka korisnika nekih drugih SUBP-a kao što su: SQLite, PostgreSQL i MongoDB. Na sljedećoj slici prikazan je rezultat upita najpopularnijih SUBP-a 2017. godine proveden na stručnoj stranici StackOverflow.

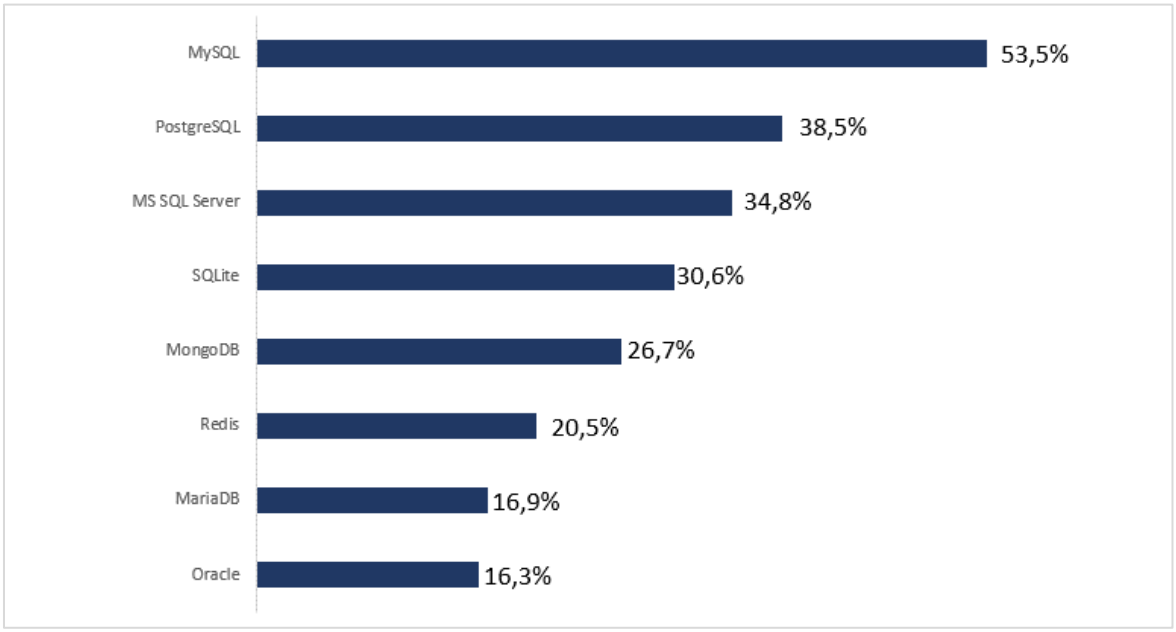


Slika 17. Pozicioniranje MS SQL Server-a na tržištu 2017. godine[34]

Na slikama koje slijede moći ćemo vidjeti kako je MS SQL Server konstantno na vrhu ljestvice popularnih SUBP-a. Na drugom mjestu ovisno o godini možemo naći ili MS SQL Server ili PostgreSQL dok prvi na listi MySQL prednjači za približno 15% naspram oba konkurenta. Time možemo zaključiti da je SQL Server dostojan „protivnika“, a podatak o tome da je konstantno među prvih pet najpopularnijih SUBP-a ipak pokazuje njegovu vrijednost, kvalitetu i moć na tržištu.



Slika 18. Pozicioniranje MS SQL Server-a na tržištu 2018. godine - autorski rad [35]



Slika 19. Pozicioniranje MS SQL Server-a na tržištu 2020. godine (autorski rad)[36]

4.2. MS SQL Server 2016-2019

U ovome poglavlju istaknuti ću najrelevantnije informacije o razvoju MS SQL Server-a u zadnjih 5 godina. Neke od značajka koje su ključne za ovaj rad su implementacija podrške za rad s JSON formatom te korištenje grafovskih modela u MS SQL Server-u.

Kako je konstantno unaprjeđivanje funkcionalnosti ključno za opstanak na tržištu, Microsoft se pobrinuo da ne zaostaje za konkurencijom. MS SQL Server 2016 tržištu je predstavljen kao budućnost korištenja SUBP-a. Implementirana su nova svojstva koja konkurenti nemaju, a naglasak je stavljen na sigurnost i korištenje nove standardizirane notacije JSON.

Always encrypted novo je svojstvo koje omogućuje čuvanje osjetljivih podataka u bilo kojem okruženju, lokalno ili udaljeno (*Cloud based*[11]), a kako je sve više kompanija počelo koristiti rješenja kao što su Azure i slične, stari način enkripcije više nije pružao potrebnu razinu sigurnosti. Baze podataka koje su prije bile zaštićene samo na razini BP (enkripcija cijele BP) ili na razini stupca (pojedinačni stupac – definirano šifriranje) sada su bile sigurnije, a to se postiglo na način da je cijeli postupak enkripcije sada bio premješten izvan SQL Server-a i implementiran na strani klijenta. [37] Drugo značajno svojstvo bilo je rad s *JSON* formatom. Mogućnost čitanja podataka u JSON formatu, učitavanje istih u tablice te primjena svojstva indexa bile su samo neke od mogućnosti koje su se nudile široj javnosti, da bi se SQL Server 2017 bazirao se na poboljšanje novouvedenih svojstava. Implementirane su mogućnosti brže obrade podataka i značajnije uštede u pogledu troškova.

Ono što je SQL Server 2019 dodatno poboljšao naspram verzije iz 2017. godine je rad s *modelom temeljenim na grafovima*. Server sadrži nove ugrađene funkcije a podržano je i korištenje TSQL-a.

[11] Cloud based – izraz na engleskom jeziku koji označava da je nešto (u ovom slučaju spremljeni podaci) pohranjeno u oblaku tj. ne nalazi se fizički na računalu

5. Način implementacije u SQL serveru

Kako sam u prethodnom poglavlju već naglasila da se SQL Server s godinama razvijao, tako ću u ovom poglavlju opisati način implementacije ne relacijskih modela podataka u samome serveru. Imamo četiri modela, a svaki će pojedinačno biti opisan u posebnom pod poglavlju.

5.1. Ključ-vrijednost model podataka

Model ključ-vrijednost kako mu i sam naziv kaže sastoji se od tablica „ključ“ i tablica „vrijednost“. U ovakvom modelu, atribut ključ uvijek mora biti jedinstven stoga se pri definiranju naglasak stavlja na *unique* svojstvo. Ono što ovakav model čini jednostavnim za korištenje je svojstvo modela da podržava samo jednostavne upite tj. operacije kao što su umetanje i brisanje. Možemo reći da sustav radi na principu „sve ili ništa“. To se odnosi na način izmjene podataka. „U slučaju da želimo izmijeniti neku vrijednost (djelomično ili u potpunosti) sustav mora prebrisati već postojeće podatke za cijelu vrijednost. [38] U samome modelu podaci se mogu pohraniti kao proizvoljni skup vrijednosti (može se koristiti JSON format), a pretraživanje podataka se izvodi pomoću vrijednosti ključa.

Key	Value
AAAAA	1101001111010100110101111...
AABAB	1001100001011001101011110...
DFA766	0000000000101010110101010...
FABCC4	1110110110101010100101101...

Slika 20. Ključ vrijednost arhitektura [38]

Način na koji se to implementira u MS SQL Server-u je jednostavan. Napravi se jednostavna tablica *Ključ-vrijednost.*

	Column Name	Data Type	Allow Nulls
🔑	kljuc	varchar(10)	☐
	boja_vrijednost	varchar(10)	☑
			☐

tablica_kljuc_vrijednost

- 🔑 kljuc
- boja_vrijednost

Slika 21. Tablica ključ – vrijednost (autorski rad)

Results | Messages

	kljuc	boja_vrijednost
1	boja_1	Crvena
2	boja_2	Cma
3	boja_3	Plava
4	boja_4	Zelena
5	boja_5	Siva

Slika 22. Podaci tablice ključ-vrijednost (autorski rad) [39]

Model ključ-vrijednost na tržištu se ističe svojom visokom optimizacijom za aplikacije koje izvode jednostavna pretraživanja, ali s druge strane nije najprikladniji u slučaju kad treba pretraživati podatke. Pretraživanje podataka po vrijednosti nije dobro optimizirano stoga se češće provodi pretraživanje prema ključu.

5.2. Dokument model podataka

Baza podataka koja se temelji na modelu dokumenata pohranjuje kolekciju dokumenata gdje se svaki dokument mora sastojati od podataka i imenovanih polja. Podaci koji se unose mogu biti jednostavni ili složeni elementi kao što su liste, a dokumenti se dohvaćaju uz pomoć jedinstvenih ključeva (eng. *unique keys*). Tipično je da dokument sadrži podatke za jedan entitet npr. student, kupac ili narudžba.[38] Neki od formata dokumenata koji se često koriste su XML, JSON i BSON dok je na slici u nastavku prikazan JSON format.

XML

```
<siblings>
<sibling>
<firstName>Anna</firstName>
<lastName>Clayton</lastName>
</sibling>
<sibling>
<firstName>Alex</firstName>
<lastName>Clayton</lastName>
</sibling>
</siblings>
```

JSON

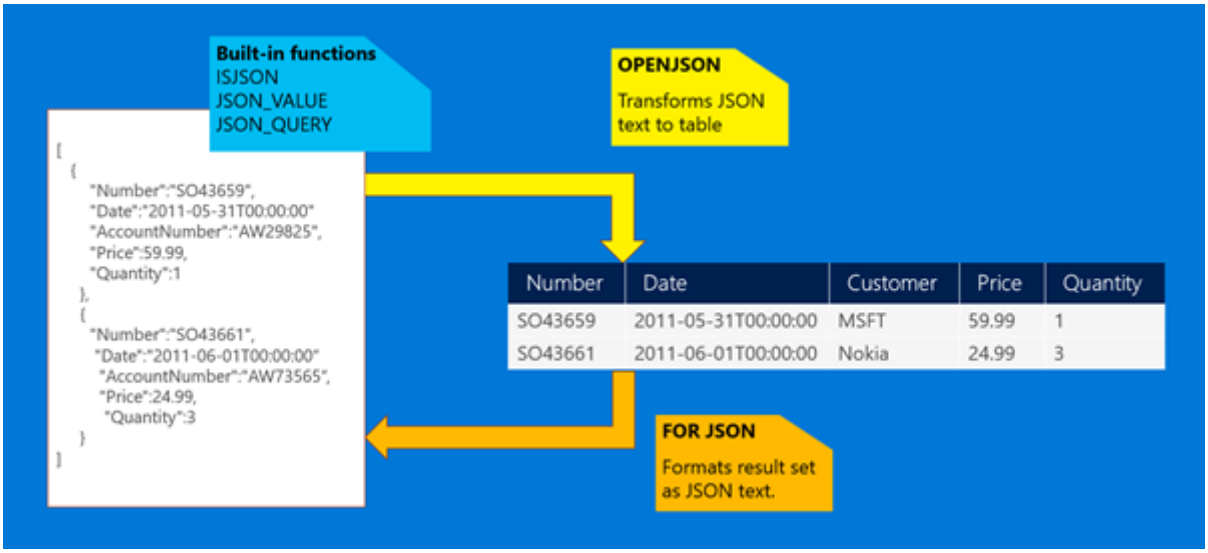
```
{
"siblings": [
{"firstName":"Anna","lastName":"Clayton"},
{"firstName":"Alex","lastName":"Clayton"}
]
}
```

Slika 23. XML i JSON format - izgled [40]

Key	Document
1001	{ "CustomerID": 99, "OrderItems": [{ "ProductID": 2010, "Quantity": 2, "Cost": 520 }, { "ProductID": 4365, "Quantity": 1, "Cost": 18 }], "OrderDate": "04/01/2017" }
1002	{ "CustomerID": 220, "OrderItems": [{ "ProductID": 1285, "Quantity": 1, "Cost": 120 }], "OrderDate": "05/08/2017" }

Slika 24 Dijagram dokument modela podataka [38]

Ono što je specifično za ovakvu vrstu modela je to što dokument sadrži skup podataka koji bi se u relacijskim bazama podataka morali prikazati u više tablica, stoga su stručnjaci u SQL Server implementirali i naredbe koje bi olakšale korištenje JSON formata. Time su omogućili pretvorbu JSON dokumenata uz korištenje osnovnih SQL naredbi. [38][41]



Slika 25. Transformacija JSON formata u relacijski [42]

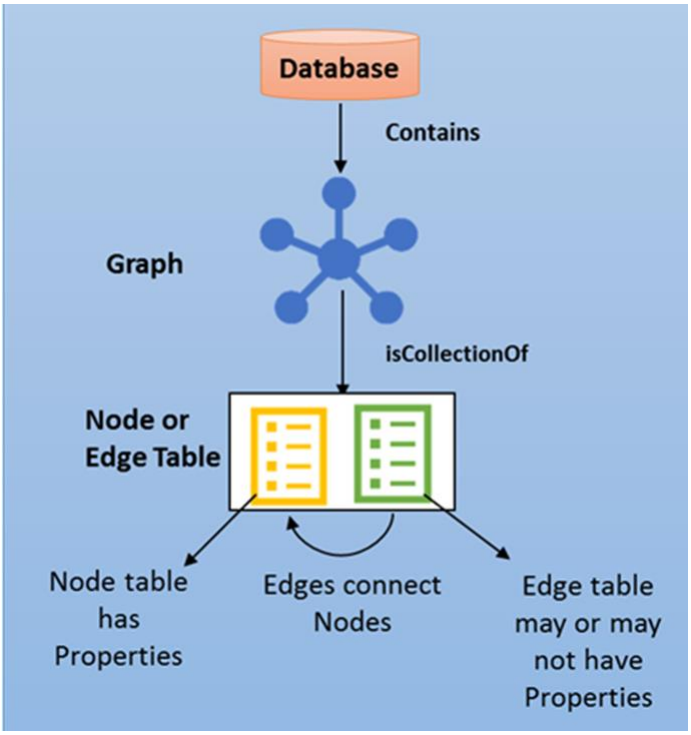
Uz pomoć ugrađenih funkcija i operatora u SQL Serveru s JSON tekstom se može:

- Pretvarati JSON tekst, čitati ili uređivati vrijednosti
- Pretvarati nizove JSON objekata u tablični format
- Izvoditi bilo koji T-SQL upit nad pretvorenim JSON objektima
- Formatirati rezultate T-SQL upita u JSON format

Svakako možemo reći da u novijoj verziji SQL Servera rad s JSON formatom više nije nikakav problem. Dokumenti se s lakoćom pretvaraju u relacijske modele, a samim time „nepoznati“ NoSQL se pretvara u lako čitljivi SQL.

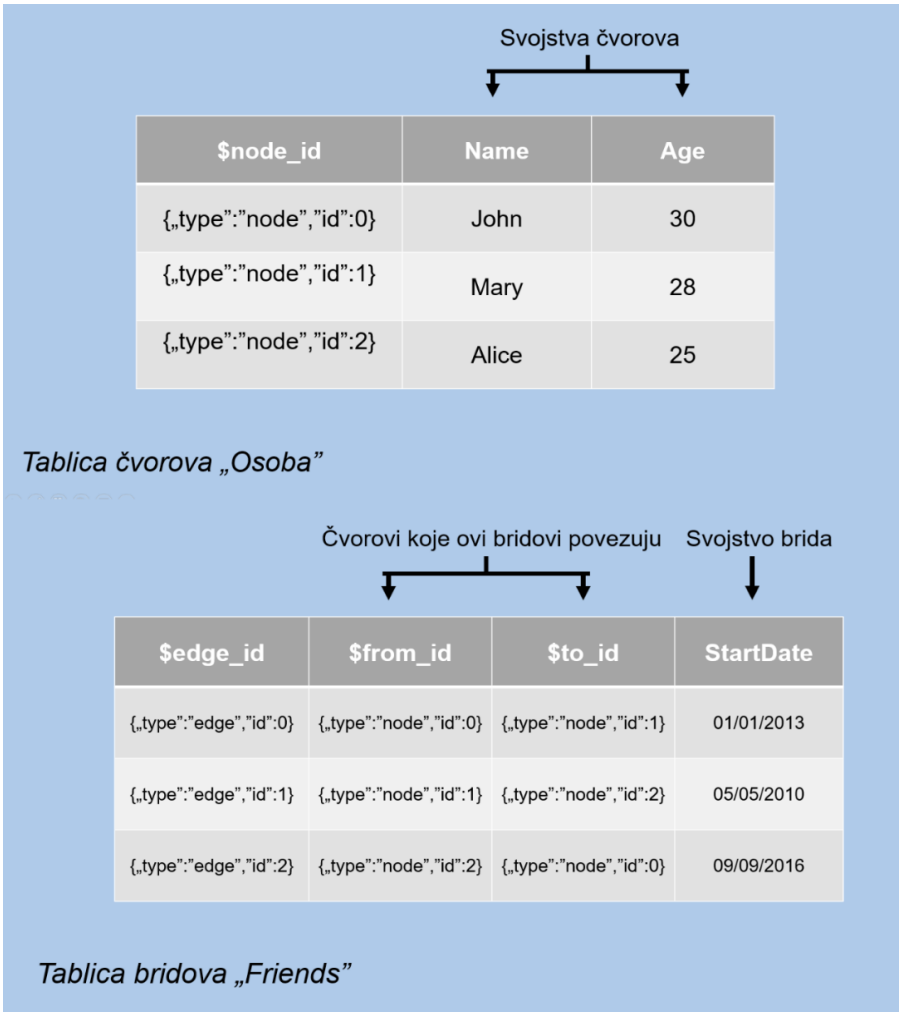
5.3. Graf model podataka

Graf model podataka bazira se na čvorovima, bridovima i svojstvima. Bridovi određuju odnose između čvorova, a svojstva pružaju informacije o čvorovima i bridovima. U MS SQL Server-u model grafova je implementiran tako da postoji tablica čvorova i tablica bridova. Tablice se mogu kreirati pod bilo kojom shemom iz BP no bez obzira na to, sve pripadaju jednom logičkom grafu. Bitno je naglasiti da korisnik može kreirati samo jedan graf po bazi podataka.



Slika 26. Arhitektura modela grafa [43]

Tablica čvorova predstavlja entitet u shemi grafa, a svaki put kada se kreira tablica čvora, zajedno sa korisnički definiranim stupcima, stvara se implicitni stupac $node_id koji jedinstveno identificira zadani čvor u BP. Vrijednosti u $node_id generiraju se automatski i predstavljaju kombinaciju tablice čvora i object_id-a i interno generirane *BigInt* vrijednosti. [43] Bitno je naglasiti da se korisnicima preporučuje da u trenutku kreiranja tablice čvora postave jedinstveno ograničenje ili indeks u stupcu $node_id, a u slučaju i da korisnik zaboravi to učiniti sustav će se sam pobrinuti za to i stvorit će jedinstveni indeks. Tablica bridova s druge strane predstavlja odnos u grafu. Bridovi služe da povezuju čvorove i omogućuju korisnicima da oblikuju odnose na grafu, a sama tablica može ili ne mora sadržavati atribute koje je definirao korisnik. „Svaki put kada se stvori tablica bridova, zajedno s atributima definiranim od strane korisnika, u rubnoj se tablici stvaraju tri implicitna stupca: $edge_id, $from_id i $to_id“.[43] Kod prvog objavljivanja opcija stavljanja ograničenja (eng. *constraint*) na tablici bridova nije omogućena tj. ograničeno je spajanje bilo koje vrste čvorova. Čvorove koje bridovi mogu povezivati regulirani su vrijednostima koje sadrže stupci $from_id i $to_id, a bez obzira na vrstu, brid može povezati bilo koja dva čvora.



Slika 27. Prikaz tablice čvorova i tablice bridova (autorski rad) [43]

Također je bitno naglasiti i funkciju SHORTEST_PATH koja omogućuje pronalazak najkraćeg puta između bilo koja dva čvora. Između ostalog može omogućiti i pronalazak najkraćeg puta počevši od odabranog čvora do svih ostalih.

Neka od ograničenja i poznatih problema na koje možemo naići su:

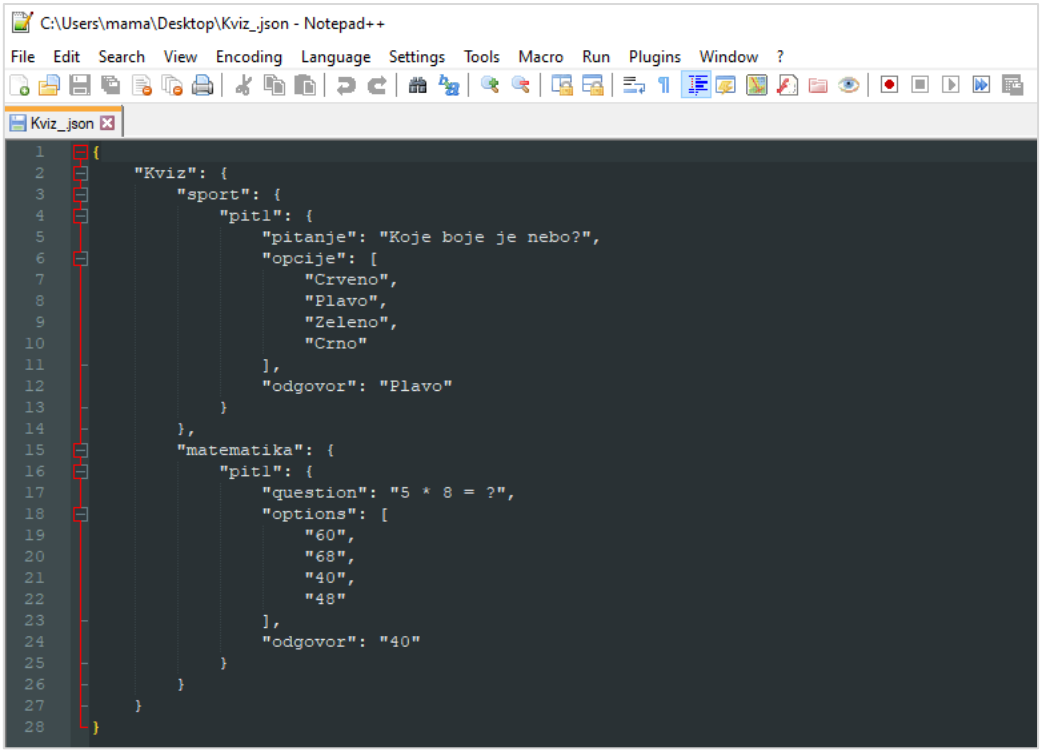
- Lokalne ili globalne privremene tablice ne mogu biti tablice čvorova ili tablice bridova
- Tipovi tablica i tipovi varijabla ne mogu biti deklarirani kao tablica čvorova ili tablica bridova
- Korisnicima je zabranjeno ažuriranje $from_id i $to_id vrijednosti kroz korištenje UPDATE naredbe. Kako bi se izvršilo uspješno ažuriranje korisnici moraju kreirati novi brid koji pokazuje na novi čvor i izbrisati stari.
- Upiti između više baza podataka na objektima grafa nisu podržani

6. Implementacija odabranih modela podataka u sustavu MS SQL Server 2019

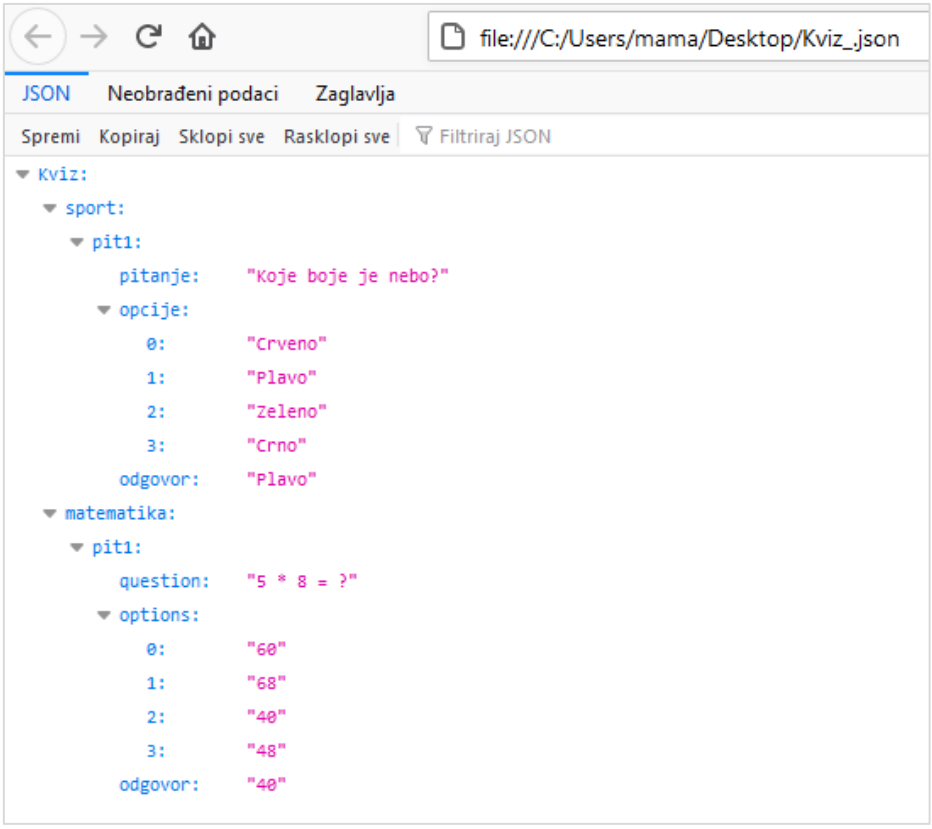
U ovome poglavlju opisat ću kako je dokument model implementiran u sustavu MS SQL Server te sve korake do uspješno formatiranog modela. Na samome početku potrebno je instalirati MS SQL Server 2019 i MS SQL Server Management Studio (SSMS). Pitate se zbog čega je potrebno instalirati i SSMS. Odgovor je lak. SSMS je program koji se koristi uz sami SQL Server, a služi za konfiguriranje, administraciju i upravljanje komponentama Server-a. To je besplatan alat koji pruža grafičko sučelje i omogućuje lakše snalaženje za rad s MS SQL Server-om. U sljedećim poglavljima bit će opisana implementacija modela temeljenog na dokumentima i modela temeljenog na grafovima u sustavu MS SQL Server 2019.

6.1. Implementacija modela temeljenog na dokumentu

Za implementaciju prvog modela odabrala sam ne relacijski model dokumenata koji je baziran na JSON formatu. Ukratko da ponovimo JSON (JavaScript Object Notation) je standardni tekstualni format za predstavljanje strukturiranih podataka na temelju JavaScript objektne sintakse. Primarno služi za prijenos podataka između poslužitelja i same web aplikacije, a koristi se i kao alternativa popularnome XML formatu. JSON format podataka moguće je kreirati i u jednostavnim aplikacijama (npr. Notepad), a programerska verzija Notepad-a, Notepad++ omogućuje i spremanje dokumenta u JSON obliku (JSON file (*.json)). Bitno je samo formatirati tekst uz pomoću JSON nadogradnje i spremiti ga sa .json ekstenzijom. Nešto više o tome možemo vidjeti na ovome linku dok kreiranje JSON dokumenta, a prikaz u web pregledniku možemo vidjeti u nastavku.



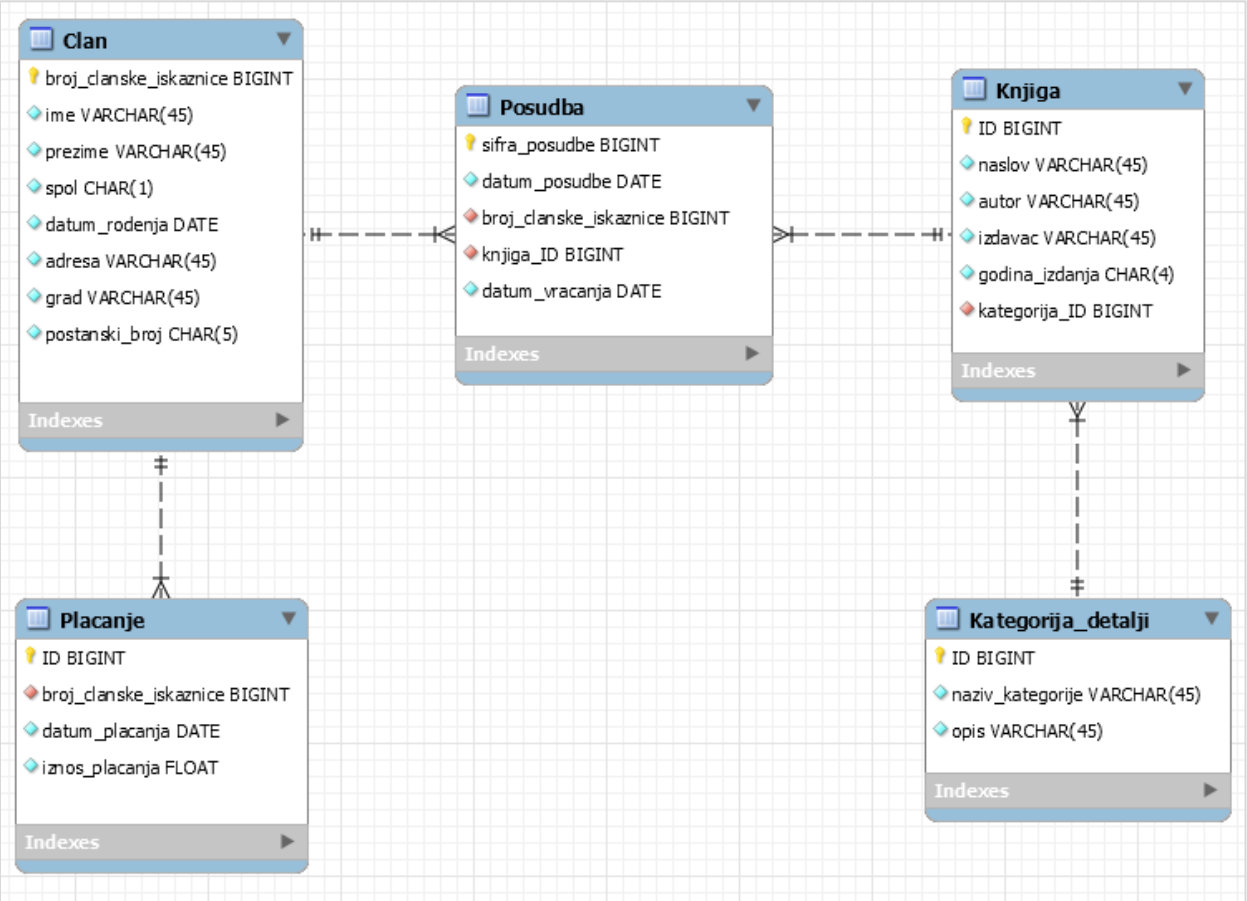
Slika 28. Izrada JSON dokumenta u aplikaciji Notepad++ (autorski rad)



Slika 29. Prikaz JSON dokumenta u pregledniku Mozilla Firefox (autorski rad)

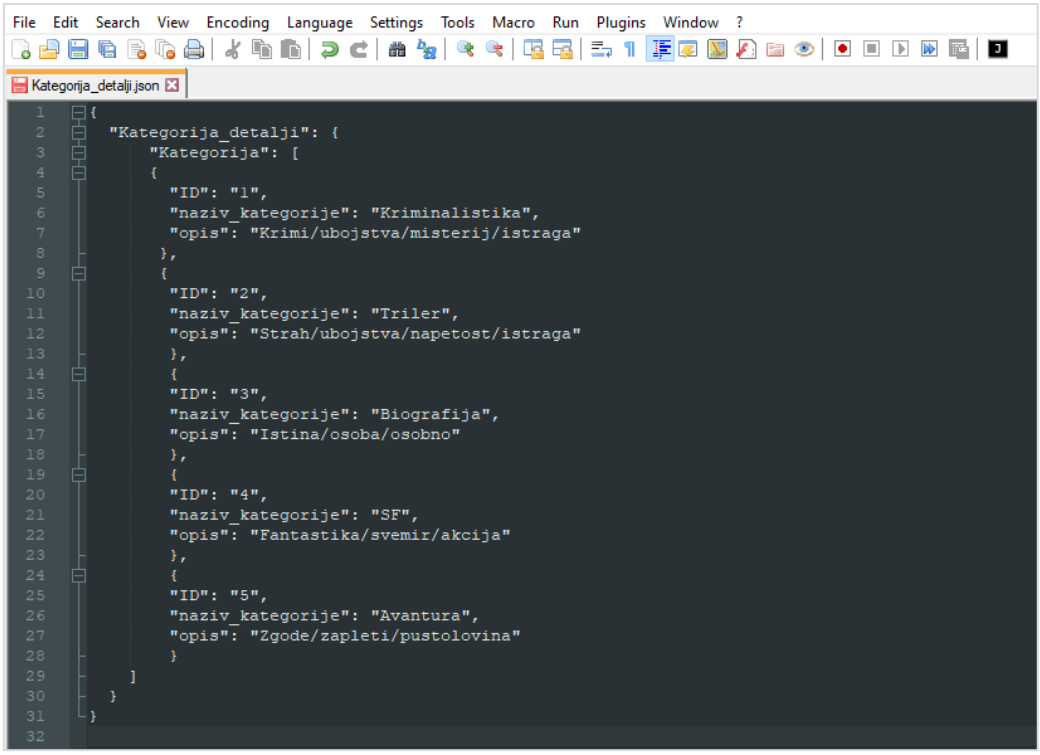
Sad kada smo definirali JSON, pažnju možemo obratiti na sami model. Za početak ću prikazati željeni izgled modela baze podataka koristeći se slikom ERA dijagrama.

Odlučila sam se za izradu jednostavne baze podataka koja opisuje način posudbe knjiga u Knjižnici. Baza sadrži ukupno 5 tablica, a tablica *Kategorija_detalji* će biti implementirana kao JSON dokument. Na toj tablici bit će prikazane osnovne operacije za rad s JSON formatom u SQL Server-u.



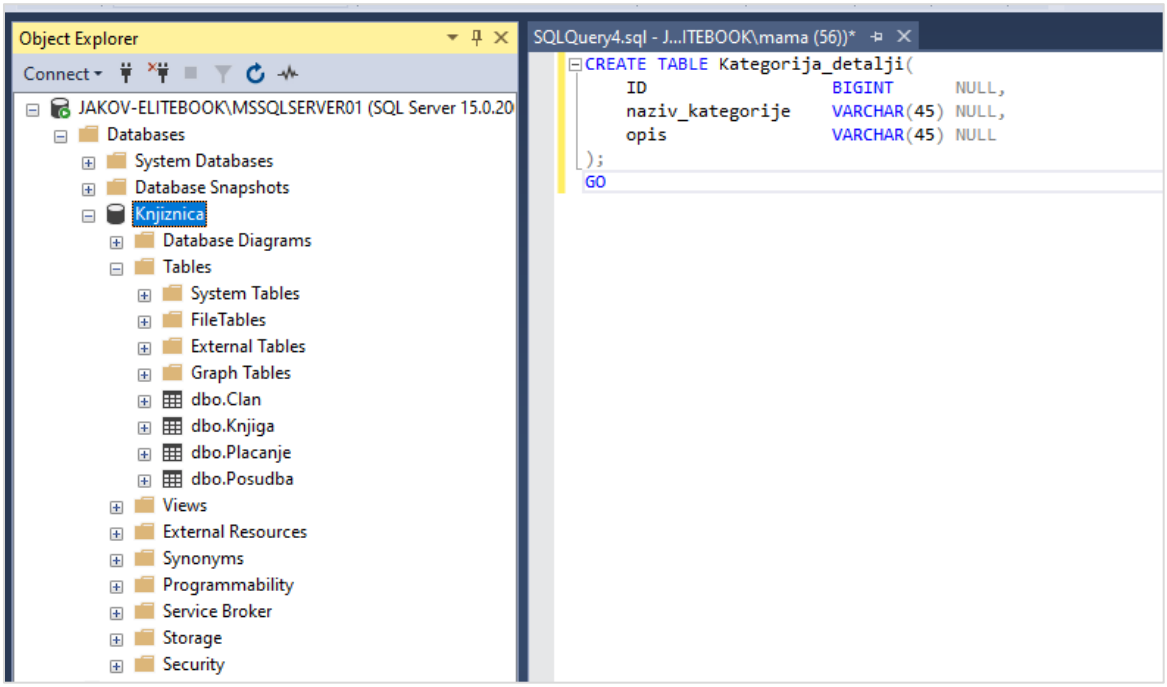
Slika 30. ERA model BP Knjižnica (autorski rad)

Prvi korak u izradi ERA modela bilo je popunjavanje tablica. Podacima sam popunila sve tablice osim tablice *Kategorija_detalji* koju ću ispuniti uz pomoć JSON dokumenta. Izrada JSON dokumenta prikazana je na sljedećoj slici.

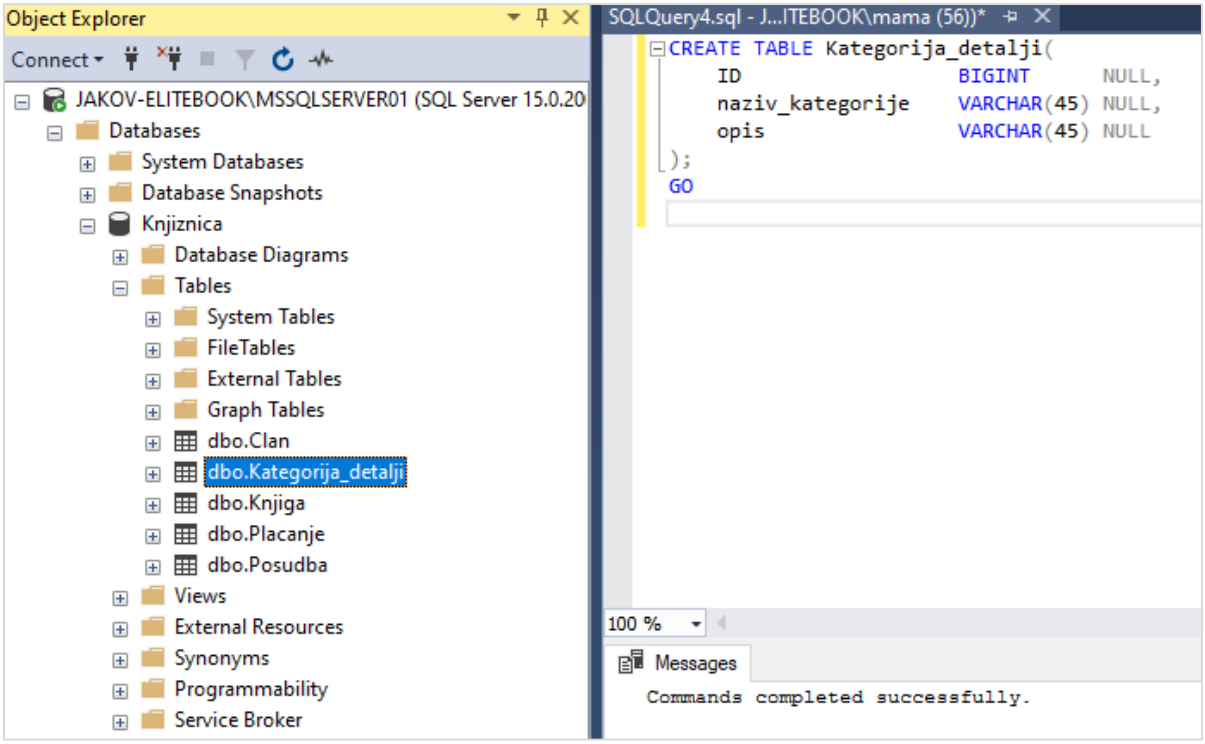


Slika 31. Kreiranje tablice *Kategorija_detalji* u JSON formatu (autorski rad)

Prvi korak je kreiranje tablice *Kategorija_detalji*.



Slika 32. Kreiranje tablice *Kategorija_detalji* – SQL (autorski rad)

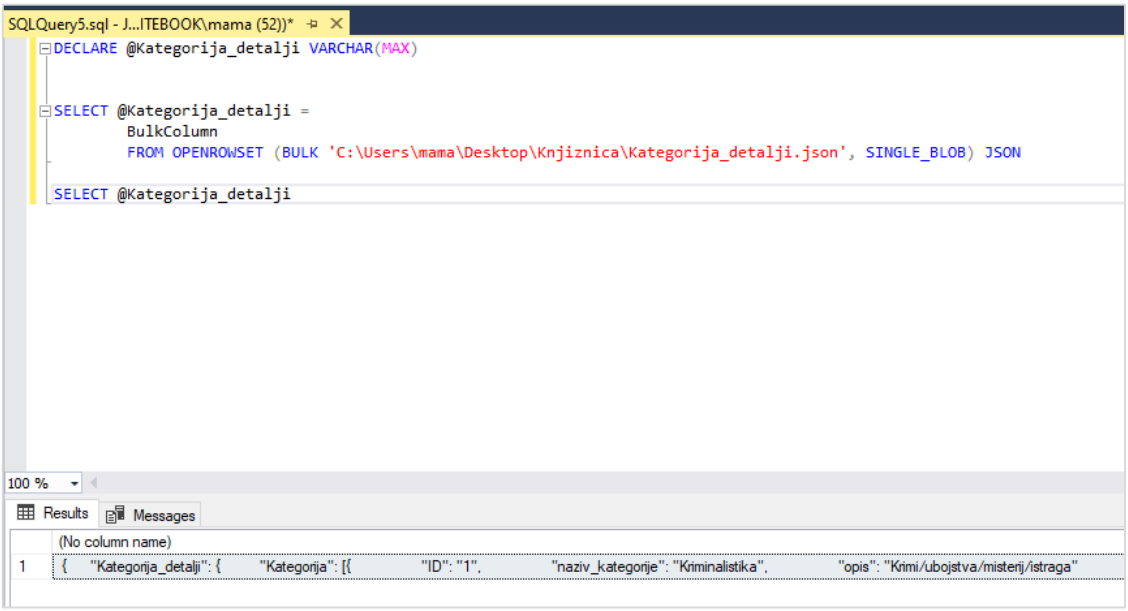


Slika 33. Kreirana tablica Kategorija_detalji (autorski rad)

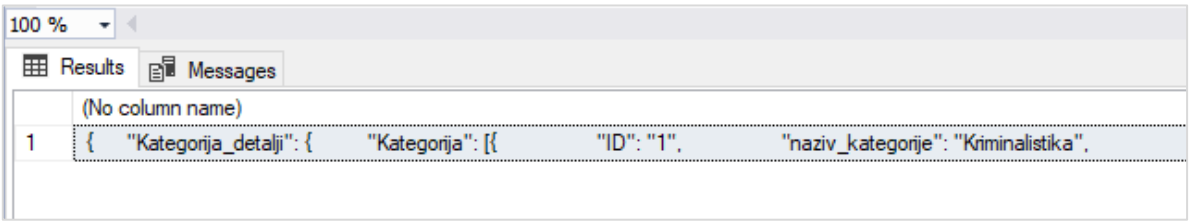
Nakon uspješno kreirane tablice Kategorija_detalji potrebno je u tablicu uvesti podatke iz JSON datoteke. To ćemo učiniti tako da za početak deklariramo varijablu *Kategorija_detalji* kao *varchar* tip podatka. Nakon toga odabiremo tu istu varijablu i definiramo je tako da nastane glavni stupac(*BulkColumn*) s podacima iz JSON datoteke. Funkcija OPENROWSET je T-SQL funkcija koja omogućuje čitanje podataka iz više izvora uključujući i mogućnost uvoza podataka s pomoću BULK ključne riječi. BULK omogućuje čitanje individualnih datoteka iz računala direktno u SQL Server, npr. čitanje podataka iz tekstualnih datoteka ili Word dokumenta u tablicu baze podataka. [44] BULK opcija dodana je još u SQL Serveru iz 2005. godine, a koristeći se njome može se primijeniti čitanje podatkovne datoteke kao jednog od tri tipa podataka.

- SINGLE_BLOB – čita datoteku kao varbinary(MAX)
- SINGLE_CLOB – čita datoteku kao varchar(MAX) i u ASCII formatu
- SINGLE_NCLOB – čita datoteku kao nvarchar(MAX) i u UNICODE formatu

OPENROWSET u našem primjeru uz odabir varijable Kategorija_detalji vraća jedan stupac nazvan BulkColumn kao rezultat dohvata JSON datoteke.

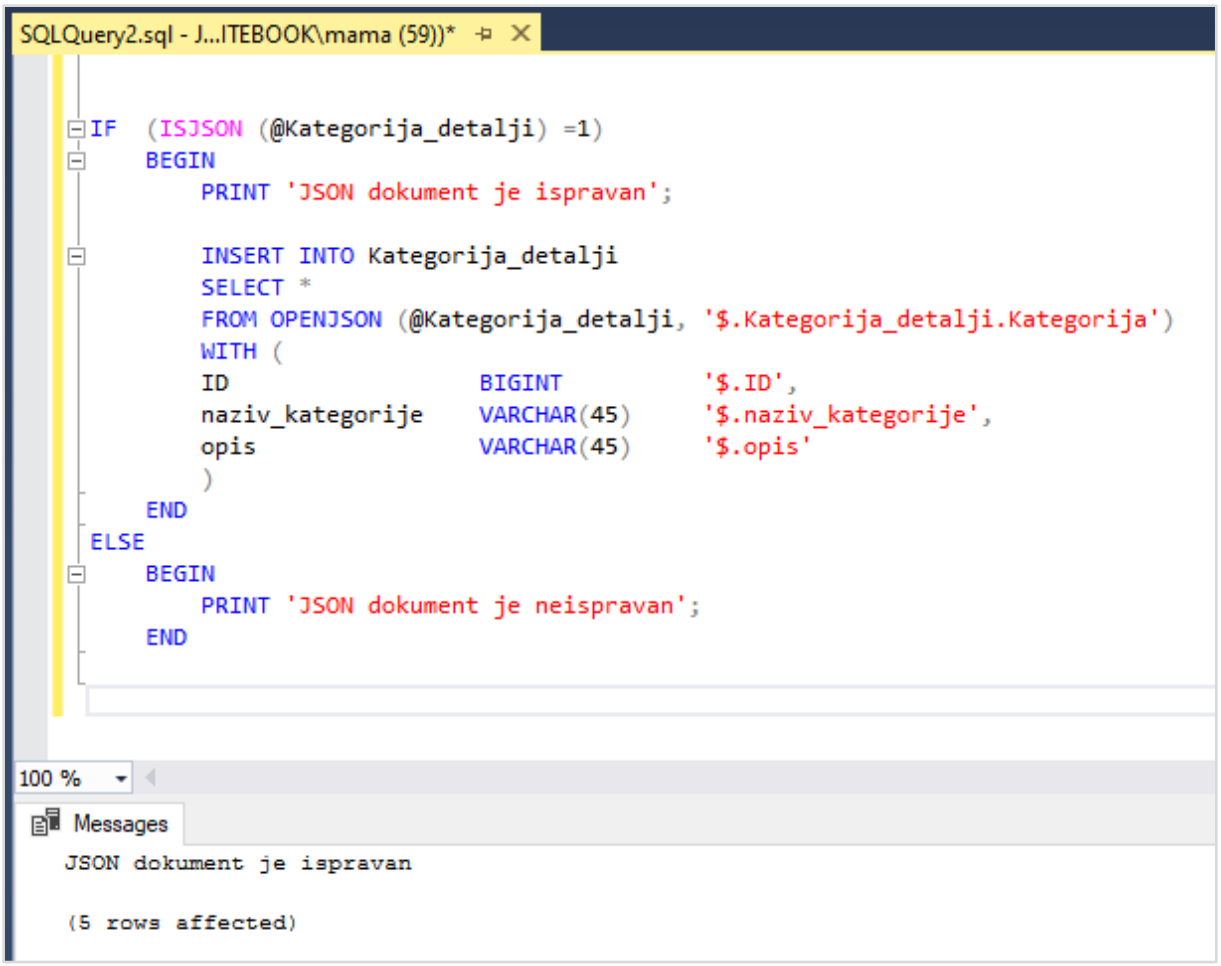


Slika 34. Dohvat JSON podataka uz OPENROWSET funkciju (autorski rad)

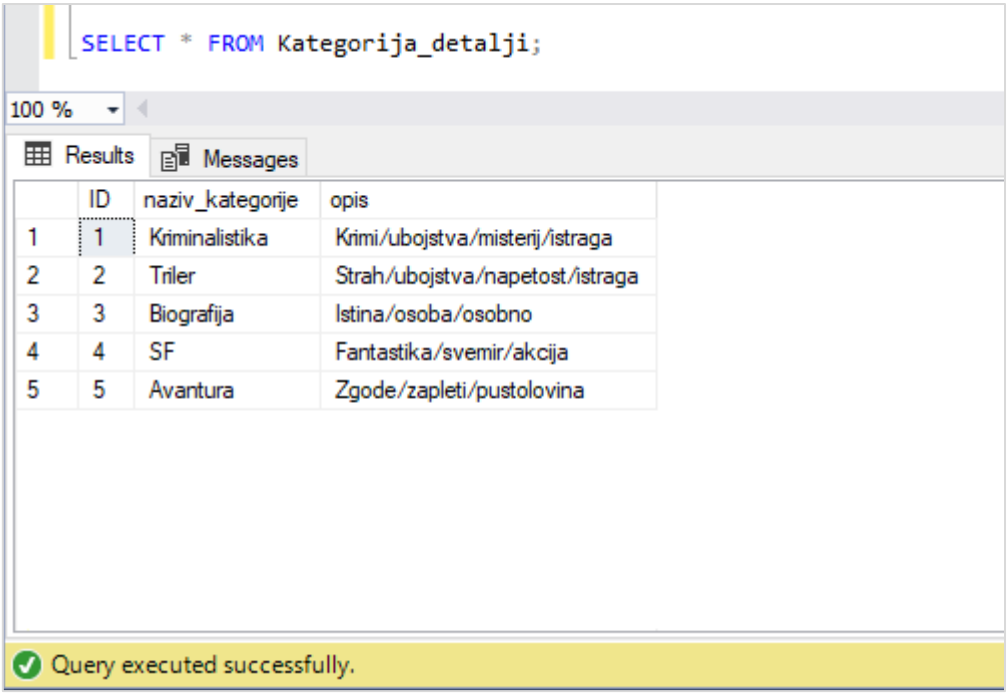


Slika 35. Detaljniji prikaz JSON podataka (autorski rad)

Nakon što smo prenijeli podatke iz JSON dokumenta u tablicu možemo provjeriti je li naš JSON dokument ispravan. U slučaju da je smjestit ćemo JSON dokument u tablicu Kategorija_detalji i uz pomoć naredbe OPENJSON pretvoriti JSON dokument u relacijski na način da rasporedimo određene atribute po stupcima i smjestimo vrijednosti u iste. Na slici 39 možemo vidjeti upit koji izvršava pretvorbu u relacijski oblik dok ćemo na slici 40 vidjeti izvršeni upit koji prikazuje novonastalo stanje i izgled tablice *Kategorija_detalji.*

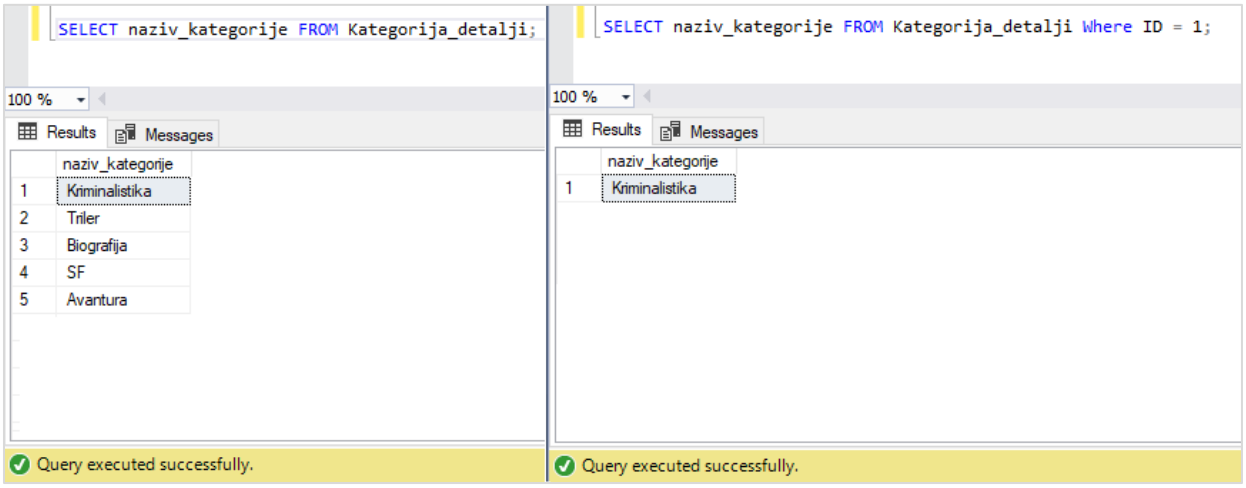


Slika 36. Upit -pretvorba u relacijski oblik (autorski rad)



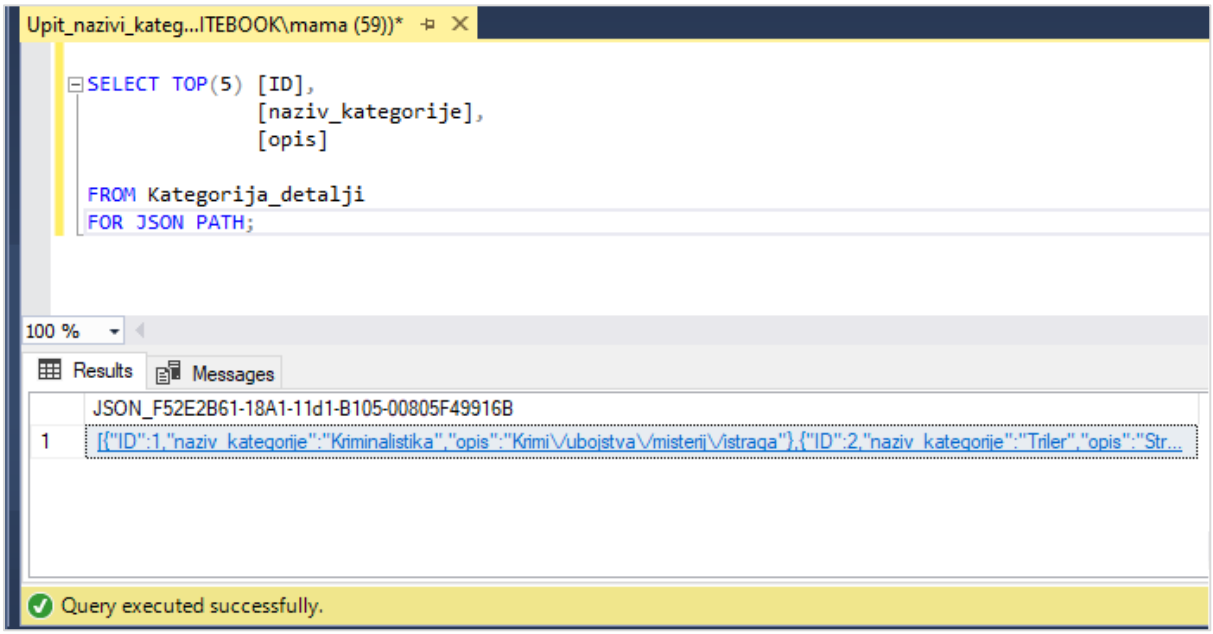
Slika 37. Izvršeni upit – izgled tablice Kategorija_detalji (autorski rad)

Kako su podaci sada spremljeni u relacijskom obliku možemo probati napisati par standardnih SQL upita i vidjeti kako se ponašaju novi podaci.



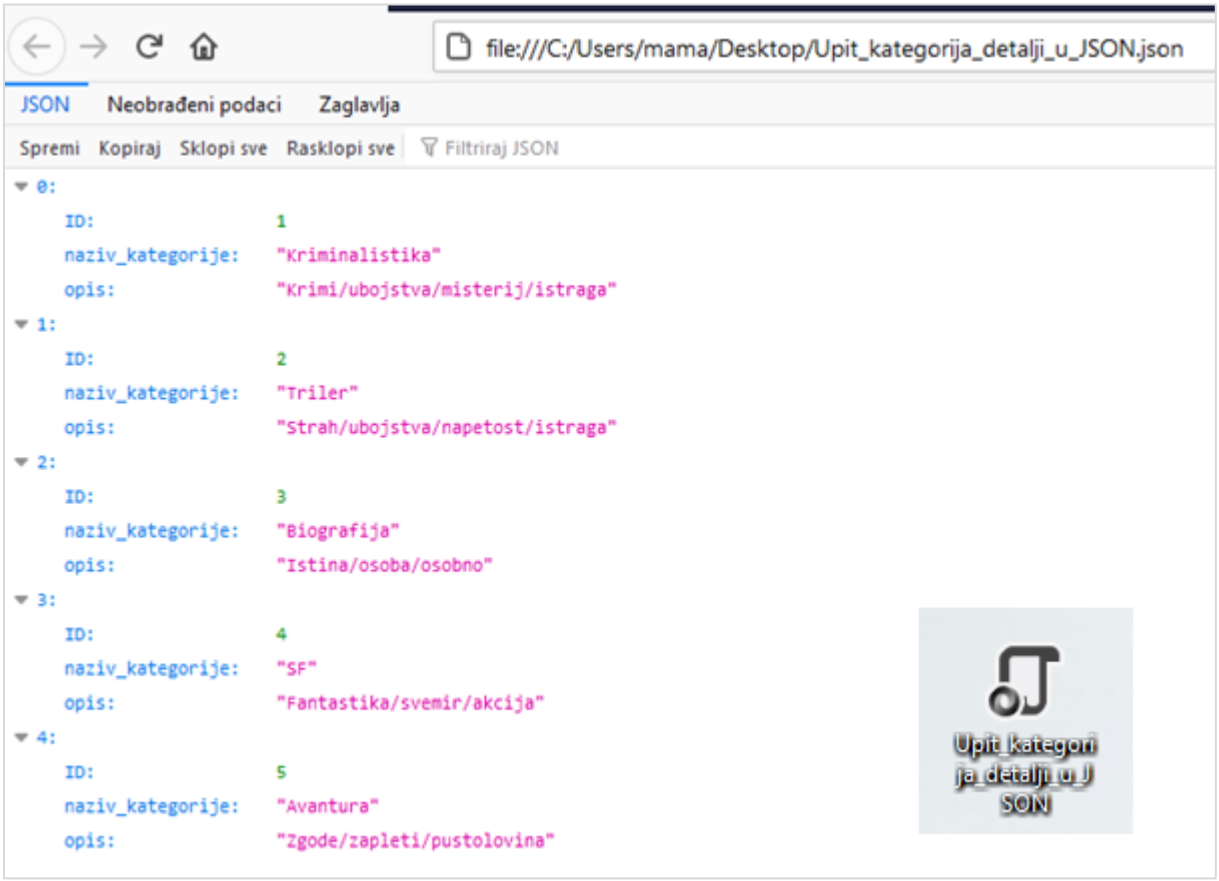
Slika 38. Korištenje jednostavnih SQL upita (autorski rad)

Ono što je još važno naglasiti je da stručnjaci nisu učinili samo uslugu onima koji se koriste ne relacijskim bazama podataka, već su omogućili i korisnicima relacijskih da lako pretvaraju svoje modele u neke druge formate pa tako rezultate sljedećeg upita možemo pohraniti i u JSON formatu. Novi SQL Server omogućuje i pretvorbu iz JSON-a u relacijski model i pretvorbu iz relacijskog modela u JSON.



Slika 39. Pretvorba iz relacijskog u JSON format (autorski rad)

Na sljedećoj slici možemo vidjeti izgled JSON spremljenog dokumenta u pregledniku Mozilla Firefox, dok je u donjem desnom kutu iste slike postavljen snimak radne površine kako bi se prikazao izgled JSON dokumenta (ikona i naziv).

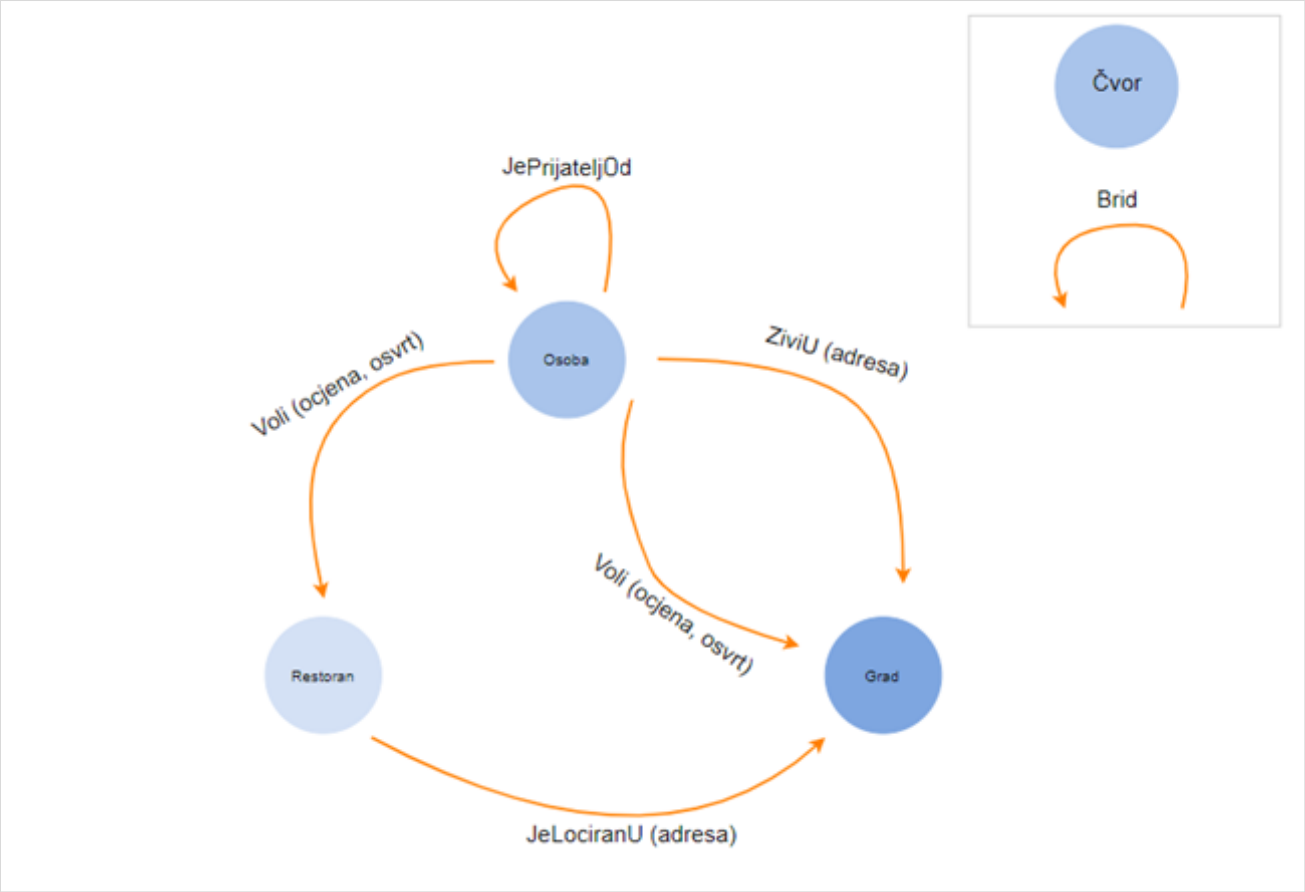


Slika 40. JSON datoteka - radna površina i preglednik (autorski rad)

U načelu je vidljivo da rad je s ne relacijskim konceptima na jednostavan način implementiran u MS SQL Server-u i da se težilo ka jednostavnosti korištenja JSON formata.

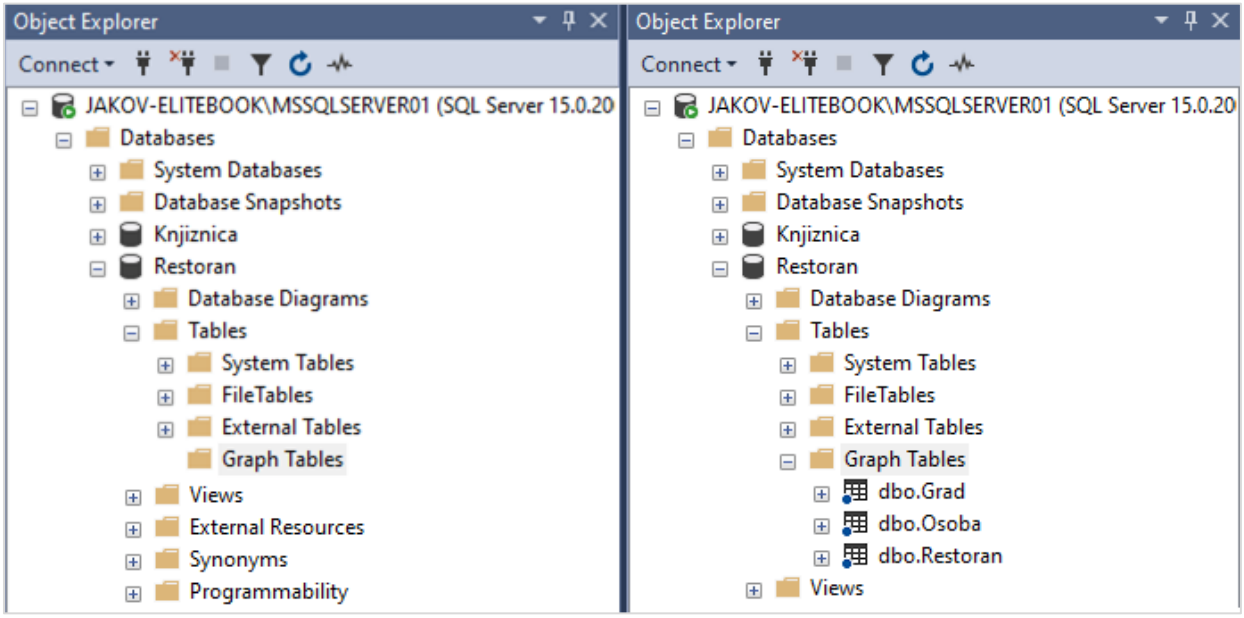
6.2. Implementacija modela temeljenog na grafovima

Za implementaciju drugog modela odabrala sam implementaciju modela temeljenog na grafovima. U nastavku ću pojasniti na koji način se radi s takvim modelom u MS SQL Server-u 2019. Kako sam u poglavlju 5.4 detaljnije objasnila princip rada s grafovskim modelom u Server-u ovdje ću prikazati implementaciju jednostavnog koncepta. Grafovske baze podataka orijentirane su na odnos između više entiteta, a kao primjer na kojem ću prikazati rad s takvim modelom odabrala sam sustav povezanosti restorana, ljudi i gradova. U osnovi, moći ćemo vidjeti korelaciju između ljudi, grada u kojem žive te različitih restorana koje posjećuju. Radi lakšeg pregleda korelacija, na sljedećoj slici prikazat ću grafovski model izrađen u alatu Visual Paradigm, a kasnije kako nalaže kreiranje grafovskog modela u Server-u kreirat ću tablice čvorova i tablice bridova. Ukoliko je potrebno same podatke prikazati i u obliku grafa u Server-u možemo koristiti Microsoft Power BI ili napisati malo duži upit.

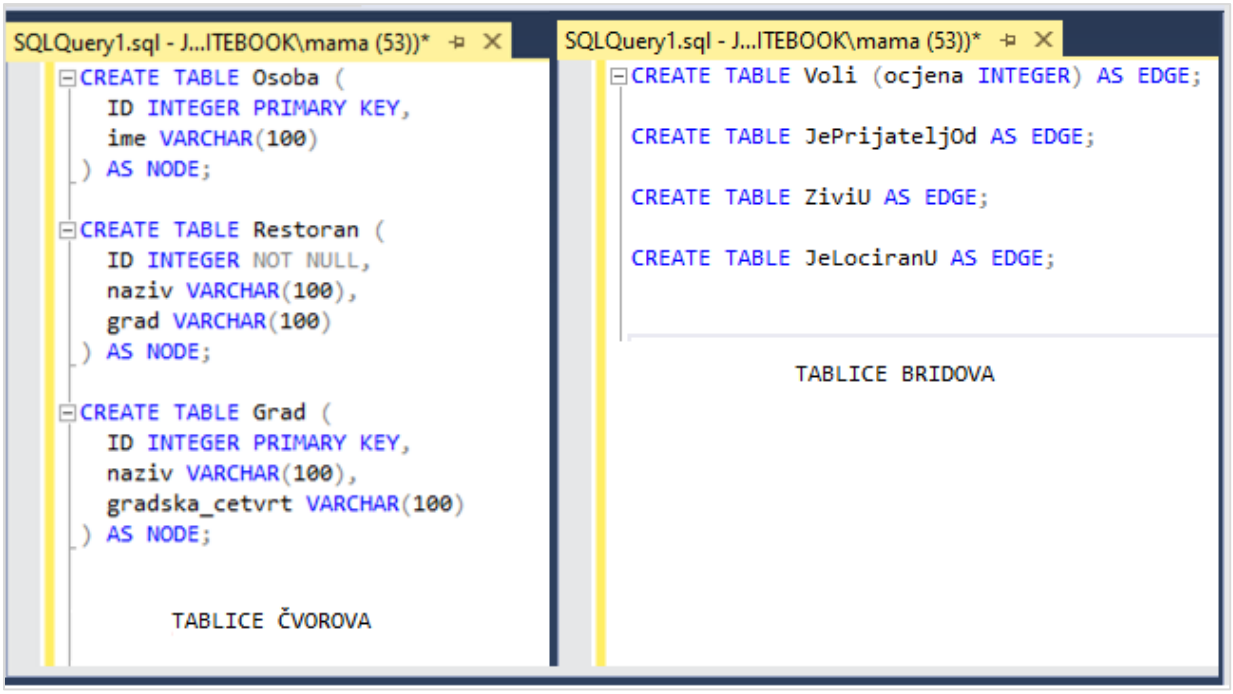


Slika 41. Prikaz grafovskog modela (autorski rad)

Sve tablice koje ćemo kreirati u grafovskom modelu spremaju se na posebnu lokaciju tj. u mapu *Graph Tables*, stoga kada kreiramo tablice čvorova i bridova, one će biti spremljene u toj mapi. Na slici 40 možemo vidjeti izgled mape prije i nakon kreiranja nekoliko tablica.



Slika 42. Spremanje tablica u mapu Graph Tables (autorski rad)



Slika 43. Kreiranje tablica čvorova i bridova (autorski rad)

Nakon što smo kreirali tablice čvorova i tablice bridova potrebno je popuniti tablice podacima. Prikaz unosa podataka i rezultate unosa prikazat ću na slikama koje slijede.

SQLQuery2.sql - J...ITEBOOK\mama (52))* | SQLQuery6.sql - J...ITEBO

```
INSERT INTO Osoba (Id, ime)
    VALUES (1, 'Jakov')
         , (2, 'Marija')
         , (3, 'Barbara')
         , (4, 'Vinko')
         , (5, 'Zvonimir');

INSERT INTO Restoran (Id, naziv, grad)
    VALUES (1, 'Vinodol','Zagreb')
         , (2, 'Duje','Split')
         , (3, 'Pet bunara', 'Zadar');

INSERT INTO Grad (Id, naziv, gradska_cetvrt)
    VALUES (1,'Zagreb','Centar')
         , (2,'Split','Manuš')
         , (3,'Zadar','Poluotok');
```

Slika 44. Unos podataka u tablice Čvorova (autorski rad)

Izgled tablica s unesenim podacima možemo vidjeti na sljedećoj slici.

100 % | Results | Messages

	$node_id_14D127DF426943779B234C7B07EDB34D	ID	naziv	gradska_cetvrt
1	{"type":"node","schema":"dbo","table":"Grad","id":0}	1	Zagreb	Centar
2	{"type":"node","schema":"dbo","table":"Grad","id":1}	2	Split	Manuš
3	{"type":"node","schema":"dbo","table":"Grad","id":2}	3	Zadar	Poluotok

Slika 45. Podaci u tablici ČVOR - Grad (autorski rad)

100 % | Results | Messages

	$node_id_1818F6777C5D41A1B26127096CF01337	ID	ime
1	{"type":"node","schema":"dbo","table":"Osoba","id":0}	1	Jakov
2	{"type":"node","schema":"dbo","table":"Osoba","id":1}	2	Marija
3	{"type":"node","schema":"dbo","table":"Osoba","id":2}	3	Barbara
4	{"type":"node","schema":"dbo","table":"Osoba","id":3}	4	Vinko
5	{"type":"node","schema":"dbo","table":"Osoba","id":4}	5	Zvonimir

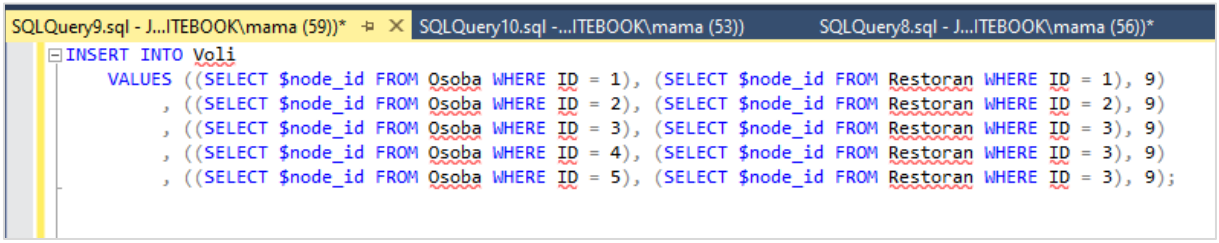
Slika 46. Podaci u tablici ČVOR - Osoba (autorski rad)

100 % | Results | Messages

	$node_id_6A74ABAD7F2E46CC948ABCFC1255C6A5	ID	naziv	grad
1	{"type":"node","schema":"dbo","table":"Restoran","id":0}	1	Vinodol	Zagreb
2	{"type":"node","schema":"dbo","table":"Restoran","id":1}	2	Duje	Split
3	{"type":"node","schema":"dbo","table":"Restoran","id":2}	3	Pet bunara	Zadar

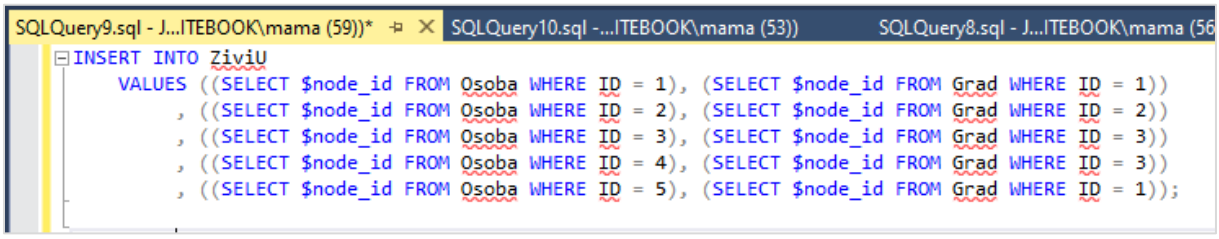
Slika 47. Podaci u tablici ČVOR - Restoran (autorski rad)

Nakon što smo kreirali tablice čvorova i unijeli podatke u njih, vrijeme je da povežemo podatke iz tablica čvorova sa podacima iz tablica bridova. Na slici u nastavku možemo vidjeti da smo povezali bridnu tablicu Voli na način da smo odredili čvorove tablica *Osoba* i *Restoran*. Tablice se spajaju tako što se čvor odabere na temelju selektiranog ID-a iz odabrane prve tablice (u ovom slučaju tablice Osoba), a drugi čvor se odabere na temelju selektiranog ID-a iz druge tablice (u ovom slučaju tablice Restoran). Tako smo zapravo napravili vezu između čvorova Osoba i Restoran i unijeli podatke o tome koja osoba voli koji restoran.

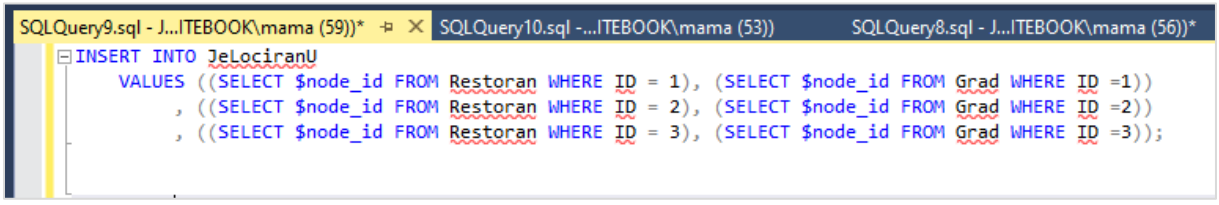


Slika 48. Povezivanje tablice Osoba i tablice Restoran (autorski rad)

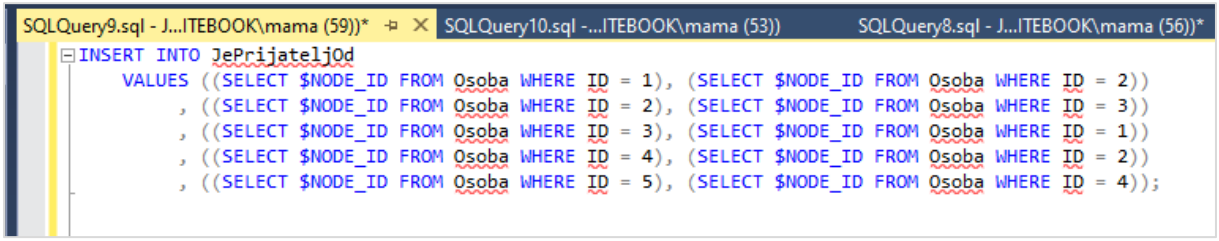
Na isti način povezat ćemo i preostale tablice čvorova s tablicama bridova.



Slika 49. Povezivanje tablice Osoba i tablice Grad (autorski rad)

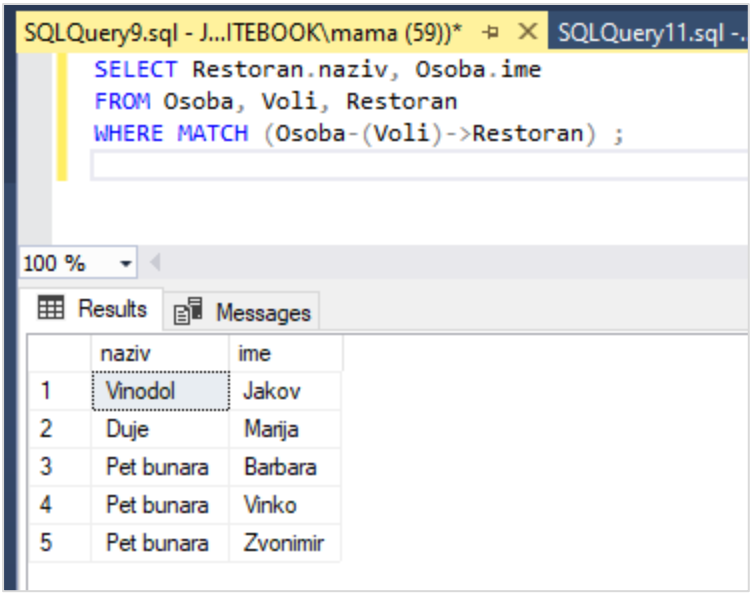


Slika 50. Povezivanje tablice Restoran i tablice Grad (autorski rad)



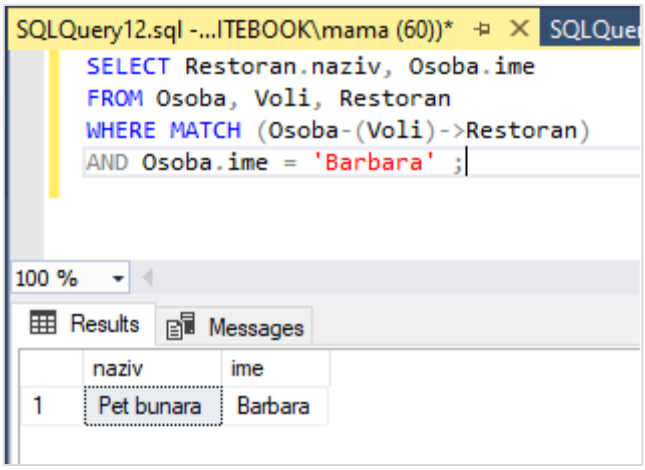
Slika 51. Kreiranje veze *JePrijateljOd* na tablici Osoba (autorski rad)

Putem sljedećeg upita pokazat ću kako su čvorovi povezani bridovima konstruirali sljedeće veze između entiteta. Cilj je prikazati podatke o spajanju osoba i restorana kojeg osoba voli pa tako možemo vidjeti popis osoba i odabrane restorane. Prikazani su nazivi restorana i imena Osoba.



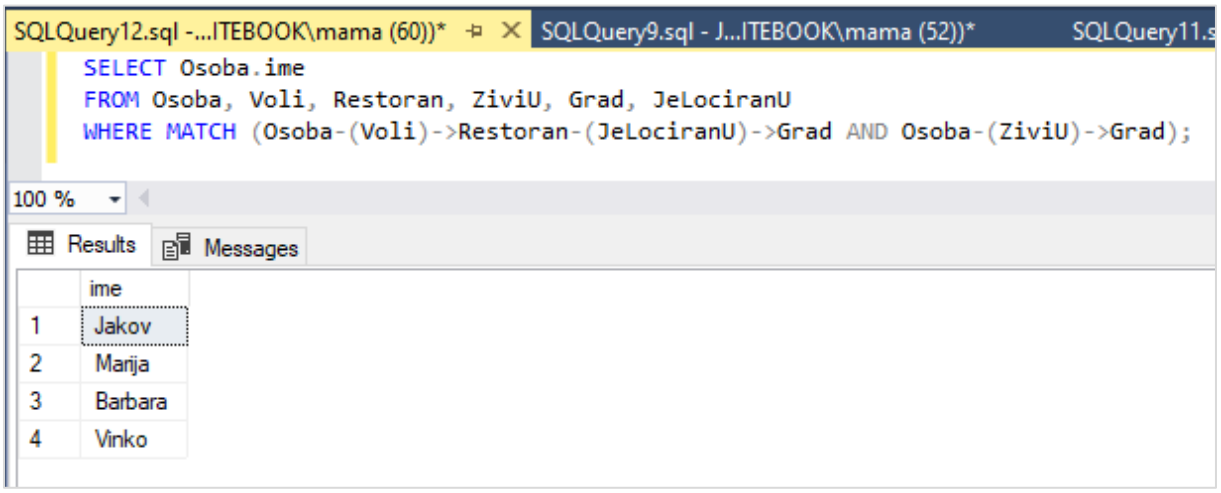
Slika 52. Upit relacije Restoran – Osoba (autorski rad)

Na slici 55 možemo vidjeti koji restoran voli određena osoba. Tako ako prethodni upit nadopunimo s naredbom *AND* i odaberemo atribut *ime* iz tablice *Osoba* te ga izjednačimo sa željenim imenom dobivamo željeni rezultat.



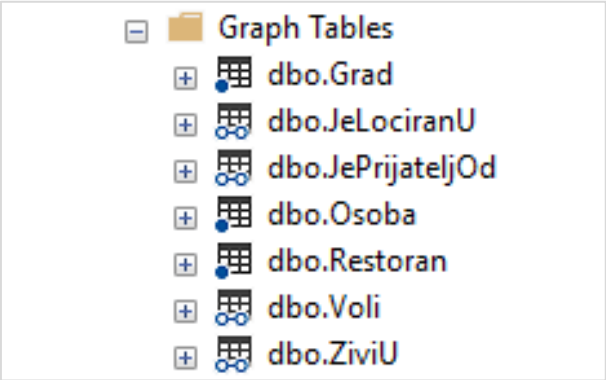
Slika 53. Pretraživanje restorana kojeg voli osoba Barbara (autorski rad)

U slučaju da želimo vidjeti popis ljudi koji vole restoran lociran u gradu u kojem žive, možemo napisati ovakav upit. Po ovome zaključujemo da Jakov, Marija, Barbara i Vinko žive u istome gradu, dok Zvonimir živi u nekom drugom.



Slika 54. Upit - Osobe - Gradovi (autorski rad)

Ono što bih još nadodala je to da se kod izrade ovakvih modela podataka u SQL Server-u nikada nećete moći zbuniti koja je tablica čvor a koja brid jer je naznačeno plavim kružićem kao što možemo vidjeti i na sljedećoj slici.



Slika 55. Razlika između tablice Čvor i tablice Brid (autorski rad)

7. Zaključak

U današnje doba bitno je pratiti trendove i opstati na tržištu, a ostati konkurentan jedan je od najtežih izazova s kojim se kompanije moraju nositi. Pojavom modernih NoSQL baza podataka, sustavi za upravljanje relacijskim bazama podataka našli su se u nezahvalnoj poziciji. Nova mogućnost korištenja nestrukturiranih podataka koju su nudile samo ne relacijske baze podataka odjednom je postala velika stvar u svijetu baza podataka. Tako je Microsoft godinama radio na unaprjeđenju svojih aplikacija kako bi na kraju postao pravi mogul. Od prve inačice SQL Server-a 2016 konstantno se radilo na unaprjeđenju i mogućnostima implementacije konkurentnih ne relacijskih modela, a na pitanje je li bolje koristiti relacijske ili ne relacijske baze podataka došla sam do zaključka da je nemoguće dati točan odgovor. Svaka od tih dviju vrsta baza podataka ima svojih prednosti i mana, a na dizajneru baze podataka je da odluči koja će se vrsta koristiti ovisno o potrebama.

Na temelju opsežne analize ne relacijskih baza podataka i načinu implementacije koncepta istih u sustavu MS SQL Server mogla sam zaključiti kako SQL Server uopće ne zaostaje za NoSQL bazama podataka jer nudi podršku za rad sa svim vrstama nestrukturiranih modela podataka. Na implementiranom primjeru dokument baze podataka zaključila sam da s ovako pretvorenim podacima iz JSON dokumenata možemo raditi kao i sa standardno unesenim podacima u tablicama, a ono što je bitno naglasiti je da je ispravno formatiran JSON dokument ključan u daljnjoj pretvorbi u relacijski model. Također tablični prikaz grafovskih baza podataka dizajniran je da korisniku olakša snalaženje (istaknute vrste tablica Čvor/Brid), a za postavljanje upita možemo koristiti i TSQL.

Na samome kraju možemo uvidjeti da proces pretvorbe ne relacijskog modela u MS SQL Server-u ne zahtijeva puno vremena niti iskustva i zbog toga je vrlo pristupačan i predvodnicima SQL-a i NoSQL-a. Takva funkcionalnost omogućuje unos većih količina podataka u samo par koraka dok se standardnim načinom čak i za manje baze podataka može izgubiti dosta vremena.

Jako me interesirala ova tema, a za vrijeme istraživanja NoSQL koncepata, zainteresirala sam se i za korištenje modela grafova zbog njihove popularne upotrebe u aplikacijama društvenih mreža te mislim nastaviti proučavati i preostale modele koje nisam implementirala u ovome radu.

Popis literature

[1] Shannon Kempe, „The NoSQL Movement – What is it“, 11.10.2012. [Na internetu] Dostupno: https://www.dataversity.net/the-nosql-movement-what-is-it/# [pristupano: 10.07.2020.]

[2] Mark Smallcombe, „SQL vs NoSQL: 5 Critical Differences“, 19.05.2020. [Na internetu], Dostupno: https://www.xplenty.com/blog/the-sql-vs-nosql-difference/#bigpicture [pristupano: 12.07.2020.]

[3] Adam Fowler, „10 Killer NoSQL Applications“, 26.03.2016 [Na internetu] Dostupno: https://www.dummies.com/programming/big-data/nosql/10-killer-nosql-applications/ [pristupano: 12.07.2020.]

[4] Klint Finly, „How Twitter uses NoSQL“, 02.01.2011. [Na internetu] Dostupno: https://readwrite.com/2011/01/02/how-twitter-uses-nosql/ [pristupano: 12.07.2020.]

[5] A. Auradkar, T. Quiggle, „Introducing Espresso – Linkedin's hot new distributed document store“, 21.01.2015. [Na internetu]

Dostupno: https://engineering.linkedin.com/espresso/introducing-espresso-linkedins-hot-new-distributed-document-store [pristupano: 12.07.2020.]

[6] K. Rabuzin, „Uvod u SQL“ , Fakultet organizacije i informatike, izd. 1, str.4 , 2011.

[7] IBM Cloud Education, „NoSQL Databases“, 06.08.2019. [Na internetu] Dostupno: https://www.ibm.com/cloud/learn/nosql-databases [pristupano: 12.07.2020.]

[8] DBBest, „NoSQL Technologies“, 14.08.2019. [Na internetu] Dostupno: https://www.dbbest.com/technologies/nosql-databases/ [pristupano: 12.07.2020.]

[9] Scylla, „NoSQL vs SQL“, 10.07.2019. [Na internetu] Dostupno: https://www.scylladb.com/resources/nosql-vs-sql/ [pristupano: 12.07.2020.]

[10] „Scalability“, na Wikipedia, the Free Encyclopedia, Dostupno: https://en.wikipedia.org/wiki/Scalability [pristupano: 12.07.2020.]

[11] Kamal Chouhbi, „It's time to familiarize yourself with NoSQL databases more than ever“, 24.03.2020. [Na internetu] Dostupno: https://towardsdatascience.com/its-time-to-familiarize-yourself-with-nosql-databases-more-than-ever-5fb1f65c22b1 [pristupano: 12.07.2020.]

[12] Rishikesh Palvee, „What is UnQL?“, 06.05.2015. [Na internetu]
Dostupno: https://www.quora.com/What-is-UnQL [pristupano: 13.07.2020.]

[13] Shannon Kempe, „UnQL: A standardized Query Language for NoSQL Databases“, 29.03.2012. [Na internetu]
Dostupno: https://www.dataversity.net/unql-a-standardized-query-language-for-nosql-databases/ [pristupano: 15.07.2020.]

[14] Architi Nengalia, „ACID properties in DBMS“, 21.11.2019. [Na internetu]
Dostupno: https://www.geeksforgeeks.org/acid-properties-in-dbms/ [pristupano: 21.07.2020.]

[15] Srividhya Umashanker, „ACID vs. CAP“, 16.02.2015. [Na internetu]
Dostupno: http://techie-experience.blogspot.com/2015/02/acid-properties-in-transactions.html [pristupano: 21.07.2020.]

[16] Vivek K. Singh, „CAP Theorem“ , 04.03.2019. [Na internetu]
Dostupno: https://medium.com/system-design-blog/cap-theorem-1455ce5fc0a0 [pristupano: 21.07.2020.]

[17] IBM Cloud Education , „CAP Theorem“, 14.11.2019. [Na internetu]
Dostupno: https://www.ibm.com/cloud/learn/cap-theorem [pristupano: 21.07.2020.]

[18] Hollis Chuang , „CAP theory of distributed systems “, 12.03.2015. [Na internetu]

Dostupno: https://www.hollischuang.com/archives/666 [pristupano: 21.07.2020.]

[19] Pranabjyoti Bordoloi, „ACID, CAP, and BASE“, 24.02.2020. [Na internetu]

Dostupno:https://medium.com/@pranabj.aec/acid-cap-and-base-cc73dee43f8c
[pristupano: 22.07.2020.]

[20] M. U. Bokhari, A. Khan, „The NoSQL Movement Prof. (Dr) “, 2016., [Na internetu]

Dostupno: https://www.semanticscholar.org/paper/The-NoSQL-Movement-Prof-.-(-Dr-)-Bokhari-Khan/a6f01c9103d3bafb8ce92641c9f2a4deaccd12f9 [pristupano: 22.07.2020.]

[21] „List of relational database management systems“, (bez dat.) u Wikipedia, the free Encyclopedia [Na internetu]

Dostupno: https://en.wikipedia.org/wiki/List_of_relational_database_management_systems
[pristupano: 25.07.2020.]

[22] HostingData, „List of NoSQL Database Management Systems“, 17.04.2020. [Na internetu]

Dostupno: https://hostingdata.co.uk/nosql-database/ [pristupano: 25.07.2020.]

[23] Lauren Schaefer, „What is NoSQL ?“, MongoDB Resources(2020) . [Na internetu]

Dostupno: https://www.mongodb.com/nosql-explained [pristupano: 25.07.2020.]

[24] M. Rouse, „ NoSQL (Not Only SQL database)“, 13.03.2017. [Na internetu]

Dostupno: https://searchdatamanagement.techtarget.com/definition/NoSQL-Not-Only-SQL [pristupano: 25.07.2020.]

[25] Prashanth Jayaram, „SQL vs NoSQL“, 15.07.2016. [Na internetu]
Dostupno: https://sqlpowershell.blog/2016/07/15/sql-and-nosql/ [pristupano: 30.07.2020.]

[26] Robert Manger, „Uvod u baze podataka“, Element, 2. izd. 2. str, 2014.

[27] Sanja Ledinek, „Microsoft Q4/20219: profitabilnost u svim segmentima“, 07.02.2020. [Na internetu]
Dostupno: https://ezadar.net.hr/sci-tech/3642945/microsoft-q4-2019-profitabilnost-u-svim-segmentima/ [pristupano: 30.07.2020.]

[28] Microsoft.inc , „SQL Server“, bez datuma, [Na internetu]
Dostupno: https://www.microsoft.com/en-us/sql-server [pristupano: 05.08.2020.]

[29] T. J. Seymour, K. Berg, „History of databases“, 31.12.2012. [Na internetu]
Dostupno: https://www.researchgate.net/publication/298332910_History_Of_Databases [pristupano: 10.09.2020.]

[30] Keith D. Foote, „A brief history of database management“, 23.3.2017. [Na internetu]

Dostupno: https://www.dataversity.net/brief-history-database-management/#:~:text=In%201960%2C%20Charles%20W.,the%20forerunners%20of%20navigational%20databases. [pristupano: 11.09.2020.]

[31] ScaleGrid, „2019 Database trends – SQL vs. NoSQL, Top Databases, Single vs. Multiple Database Use“, 04.03.2019. [Na internetu]
Dostupno: https://scalegrid.io/blog/2019-database-trends-sql-vs-nosql-top-databases-single-vs-multiple-database-use/ [pristupano: 11.09.2020.]

[32] Insiders.com, „SQL Server Pricing: Oracle vs. IBM vs. Microsoft“, 19.07.2012. [Na internetu]

Dostupno: http://www.insideris.com/sql-server-pricing-oracle-vs-ibm-vs-microsoft/ [pristupano: 11.09.2020.]

[33] CoolMinE, „Top 5 Most Popular Databases 2014“, 19.12.2014 [Na internetu]

Dostupno: https://www.fluxbytes.com/tech-news/top-5-most-popular-databases-2014/ [pristupano: 13.09.2020.]

[34] Tomer Shay, „Most popular databases in 2017 according to StackOwerflow survey“, 26.06.2017. [Na internetu]

Dostupno: https://www.eversql.com/most-popular-databases-in-2017-according-to-stackoverflow-survey/ [pristupano: 15.09.2020.]

[35] Tomer Shay, „Most popular databases in 2018 according to StackOwerflow survey“, 13.03.2018. [Na internetu]
Dostupno: https://www.eversql.com/most-popular-databases-in-2018-according-to-stackoverflow-survey/ [pristupano: 15.09.2020.]

[36] Oded Valin, „Most popular databases in 2020 and new trends“, 02.06.2020. [Na internetu]
Dostupno: https://www.eversql.com/most-popular-databases-in-2020/ [pristupano: 16.09.2020.]

[37] D. Sarka, M. Radivojević, W. Durkin, „Introduction to SQL Server 2016 - SQL Server 2016 Developer’s Guide“, 2017. [Na internetu]

Dostupno: http://projanco.com/Library/SQL%20server%202016%20developer%20guide.pdf [pristupano: 16.09.2020.]

[38] Microsoft.inc, „Understanding data store models“, 08.08.2020. [Na internetu]
Dostupno: https://docs.microsoft.com/en-us/azure/architecture/guide/technology-choices/data-store-overview [pristupano: 05.08.2020.]

[39 dodano] StackOwerflow , „Key-Value pairs in a database table“, 2009. [Na internetu]
Dostupno: https://stackoverflow.com/questions/514603/key-value-pairs-in-a-database-table [pristupano: 05.08.2020.]

[40] YourDictionary.com, „JSON – Computer Definition“, bez datuma [Na internetu]

Dostupno: https://www.yourdictionary.com/json [pristupano: 15.09.2020.]

[41] Microsoft.inc, „Store JSON documents in SQL Server or SQL Database“, 06.03.2020. [Na internetu]
Dostupno: https://docs.microsoft.com/en-us/sql/relational-databases/json/store-json-documents-in-sql-tables?view=sql-server-ver15 [pristupano: 19.08.2020.]

[42] Microsoft.inc, „JSON data in SQL Server“, 06.03.2020. [Na internetu]
Dostupno: https://docs.microsoft.com/en-us/sql/relational-databases/json/json-data-sql-server?view=sql-server-ver15 [pristupano: 19.08.2020.]

[43] Microsoft.inc, „SQL Graph Architecture“, 24.09.2018. [Na internetu]
Dostupno: https://docs.microsoft.com/en-us/sql/relational-databases/graphs/sql-graph-architecture?view=sql-server-ver15 [pristupano: 20.08.2020.]

[44] Andy Novick, „Using OPENROWSET to read large files into SQL Server“, 21.04.2020. [Na internetu]
Dostupno: https://www.mssqltips.com/sqlservertip/1643/using-openrowset-to-read-large-files-into-sql-server/ [pristupano: 26.08.2020.]

Popis slika

Popis tablica

Popis tablica treba biti izrađen po uzoru na indeksirani sadržaj, te upućivati na broj stranice na kojoj se tablica može pronaći.

Prilozi

[1] Slika 1 – Pretraga_MS_SQL_Servera_na_internetu_(autorski rad).png

[2] Slika 2 – Odabir_željene_verzije_Server-a.png

[3] Slika 3 - Odabir_tipa_instalacije_SQL_Server-a_(autorski rad).png

[4] Slika 4 - Odabir_opcije_Install_SSMS_(autorski rad).png

[5] Slika 5 - Službena_stranica_SSMS-a.png

[6] Slika 6 - SSMS_setup_skinut_u_mapi_Downloads_(autorski rad).png

[7] Slika 7 - Provjera_instalacije_SSMS_(autorski rad).png

[8] Slika 8 - Konekcija_SSMS-a_(autorski rad).png

[9] Slika 9 - Kreiranje_nove_BP_(autorski rad).png