Proof of concept

1. Strategija particionisanja podataka

Potrebno je odraditi particionisanje na više načina. Particionisanje je potrebno uraditi prema funkcionalnostima korisnika. Na primer, adminstrator sistema ima pristup funkcionalnostima kao što su žalba, kompanije i loyalty programi. Dva od tri entiteta su mu izrazito svojstvena. Administrator kompanije je usko vezan za poslove u vezi kompanije kojom se bavi i opremom. Stoga je opravdano vršiti particionisanje na osnovu funkcionalnosti korisnika. Ovo se lako može iskoristiti za pravljenje horizontalnih particija prema funkcionalnostima i uz proveru korisnikove uloge vršiti preusmeravanja na određene servere. Takođe se može na jednostavniji način vršiti particionisanje onih grupa funkcionalnosti koje su frekventnije, poput grupe funkcionalnosti vezanih za kupce. Moguće je, dodatno, poboljšati horizontalno particionisanje na način da se konkretne tabele vezane za kompanije, opremu i rezervacije raspoređuju u grupama prema kompanijama i na taj način čuvaju na odvojenim serverima.

2. Strategija za replikaciju baze podataka

Osnovna verzija sistema sa jednim aplikativnim serverom će posedovati četiri *primary* servera i 7 *read* replika, na *slici 7.1* predložena arhitektura je prikazana približnije. Pomoću *PostgreSQL streaming replication* mehanizma će se vršiti propagacija izmena sa servera baza podataka na kojima se vrši ažuriranje ka serverima baza podataka na kojima se vrši isključivo čitanje.

Funkcionalnosti u kojima se isključivo čita, i koje su vrlo intezivne u korišćenju od strane većine korisnike će biti preusmeravane na read replike. Primeri nekih funkcionalnosti koje se koriste za intezivno čitanje su pretrage kompanije i opreme. Funkcionalnosti poput rezervisanja opreme u kojima je osnovni zadatak izmena podataka će se vršiti na read-write serverima ili osnovnim tabelama, kojih će biti manje u odnosu na read replike. Dakle, na *read-write* serverima od kojih će se vršiti dalja propagacija ažuriranja ka read replikama, će se nalaziti tabele koje su horizontalno particionisane. Podsećanja radi, horizontalne particije (engl. *database shards*) će biti formirane u odnosu na dve kategorije, prva je grupa funkcionalnosti konkretnog korisnika i druga je podela entiteta sa svojim povezanim entitetima. Potrebno je precizno isplanirati particionisanje i replikaciju tako da se u što većoj meri smanji zavisnost između servera baza podataka, što naročito može biti problematično tokom transakcione obrade podataka. Sinhrono ili asihrono ažuriranje replika će se obavljati prema vrsti ažuriranja. Kada se u osnovnim tabelama vrši ažuriranje nekih bitnih podataka sa aspekta konzistentnosti, kao što su rezervacije ili oprema tada se vrši sinhrono ažuriranje, u svim ostalim slučajevima se vrši asinhrono ažuriranje. Ukoliko je sistem geografski razuđen tada je moguće odraditi kloniranje postojećeg sistema. Kompletan sistem sa jednim aplikativnim serverom i ostalim serverima baza podataka se mogu klonirati i na nekoj drugoj lokaciji. Tada je potrebno spregnuti odgovarajuće *read-write* servere, radi adekvatne propagacije izmena. Prethodno opisana strategija se može implementirati zahvaljujući *PostgreSQL Replication, Hot Standby* i *Streaming Replication* mehanizmu.

3. Strategija za kesiranje podataka

Kompanije se vrlo retko menjaju, izuzetno često im se pristupa i broj kompanija je zanemarljiv u odnosu na neke druge entitete, stoga ih je potrebno čuvati duže u kešu. Strategija keširanja objekata tipa *Equipment* je *Nonrestricted Read Write.*

Entitete tipa *User* ne treba čuvati u kešu, jer im se najčešće pristupa u vezi podataka oko permisija pristupa koji su sačuvani na nivou sesije od strane *Spring Boot Security* mehanizma.

Entitete tipa *Reservation* bi trebalo keširati, jer ukoliko se uzme u obzir da aplikacija ima 500 000 rezervacija na mesečnom nivou, pretpostavka je da postoji još više termina za preuzimanje (početna rezervacija). Izvršeno je keširanje entiteta tipa *Reservation*, uz strategiju keširanja *Transactional*, pošto se podaci vrlo često menjaju i potrebno je očuvati konzistentnost.

Equipment ne treba čuvati dugo u kešu, jer se očekuje da kompanija može imati veliku količinu opreme. Strategija keširanja objekata tipa *Equipment* je *Transactional.*

Lokacijama se često pristupa zbog kompanija i stoga je pametno čuvati ih u kešu, nikad se ne menjaju.

Količina tipova memorije se određuje preciznije u zavisnosti od metrika i samih hardverskih ograničenja aplikativnog servera. Zbog postojanja više aplikativnih servera, za svaki ponaosob će se dinamički utvrđivati zauzeće različitih tipova memorije. Korekcija strategije će se vršiti u skladu sa metrikama.

Keširanje se vrši samo za one metode u kojima nema parametara koji su u većini slučajeva proizvoljni, poput naziva prilikom pretrage kompanije.

4. Okvirna procena za hardverske resurse potrebne za skladistenje svih podataka u narednih 5 godina

U nastavku je predstavljeno zauzeće stalne memorije svake od tabela.

Tabela Company

Naziv obeležja	Tip podatka	Memorijsko zauzeće
ID	Integer	4 bajta
Name	Varchar(255)	30 bajtova (*)
Created_by_admin	Integer	4 bajta
Description	Varchar(255)	100 bajtova (*)
Location	Integer	4 bajta
OpeningTime	Timestamp	8 bajtova
ClosingTime	Timestamp	8 bajtova
AverageScore	Double precision	8 bajtova
		= 166 bajtova

Tabela User

Naziv obeležja	Tip podatka	Memorijsko zauzeće
ID	Integer	4 bajta
Email	Varchar(255)	30 bajtova (*)
FirstName	Varchar(255)	10 bajtova (*)
LastName	Varchar(255)	10 bajtova (*)
Password	Varchar(255)	10 bajtova (*)
Enabled	Boolean	1 bajt
LastPasswordResetDate	Timestamp	8 bajtova
		= 73 bajta

Tabela CompanyAdmin

Naziv obeležja	Tip podatka	Memorijsko zauzeće
ID	Integer	4 bajta
Company	Integer	4 bajta
Registered_by_admin	Integer	4 bajta
PasswordChanged	Boolean	1 bajt
		= 13 bajtova

Tabela SystemAdmin

Naziv obeležja	Tip podatka	Memorijsko zauzeće
ID	Integer	4 bajta
		= 4 bajta

Tabela RegisteredUser

Naziv obeležja	Tip podatka	Memorijsko zauzeće
ID	Integer	4 bajta
TelephoneNumber	Varchar(255)	10 bajtova (*)
PenaltyPoints	Integer	4 bajta
LoyaltyProgram	Integer	4 bajta
Hospital	Integer	4 bajta
Location	Integer	4 bajta
Points	Integer	4 bajta
Occupation	Varchar(255)	10 bajtova (*)
ActivationCode	Varchar(255)	32 bajta
		= 76 bajtova

Tabela Complaint

Naziv obeležja	Tip podatka	Memorijsko zauzeće
ID	Integer	4 bajta
Comment	Varchar(255)	100 bajtova (*)
Reply	Varchar(255)	100 bajtova (*)
Company	Integer	4 bajta
CompanyAdmin	Integer	4 bajta
RegisterUser	Integer	4 bajta
SystemAdmin	Integer	4 bajta
		= 220 bajtova

Tabela Equipment

Naziv obeležja	Tip podatka	Memorijsko zauzeće
ID	Integer	4 bajta
AvailableQuantity	Integer	4 bajta
Description	Varchar(255)	100 bajtova (*)
Name	Varchar(255)	40 bajtova (*)
Price	Integer	4 bajta
Quantity	Integer	4 bajta

Type	Integer	4 bajta
Company	Integer	4 bajta
Version	Integer	4 bajta
		= 168 bajtova

Tabela Hospital

Naziv obeležja	Tip podatka	Memorijsko zauzeće
ID	Integer	4 bajta
Name	Varchar(255)	40 bajtova (*)
Location	Integer	4 bajta
		= 48 bajtova

Tabela Location

Naziv obeležja	Tip podatka	Memorijsko zauzeće
ID	Integer	4 bajta
City	Varchar(255)	10 bajtova (*)
Country	Varchar(255)	10 bajtova (*)
Latitude	Double precision	8 bajtova
Longitude	Double precision	8 bajtova
StreetName	Varchar(255)	10 bajtova (*)
StreetNumber	Varchar(255)	3 bajta (*)
		= 53 bajta

Tabela LoyaltyProgram

Naziv obeležja	Tip podatka	Memorijsko zauzeće
ID	Integer	4 bajta
DiscountRate	Integer	4 bajta
MaxPoints	Integer	4 bajta
MinPoints	Integer	4 bajta
Type	Integer	4 bajta
Admin	Integer	4 bajta
		= 24 bajta

Tabela Reservation

Naziv obeležja	Tip podatka	Memorijsko zauzeće
ID	Integer	4 bajta
DurationMinutes	Integer	4 bajta
Status	Integer	4 bajta
Version	Integer	4 bajta
Admin	Integer	4 bajta
Hospital	Integer	4 bajta
RegisteredUser	Integer	4 bajta
TotalSum	Double precision	8 bajtova
StartingDate	Timestamp	8 bajtova
		= 44 bajta

Tabela ReservationItem

Naziv obeležja	Tip podatka	Memorijsko zauzeće
ID	Integer	4 bajta
Quantity	Integer	4 bajta
Equipment	Integer	4 bajta
Reservation	Integer	4 bajta
		= 16 bajta

Tabela Role

Naziv obeležja	Tip podatka	Memorijsko zauzeće
ID	Integer	4 bajta
Name	Varchar(255)	19 bajtova (*)
		= 23 bajta

Tabela UserRole

Naziv obeležja	Tip podatka	Memorijsko zauzeće
User	Integer	4 bajta
Role	BigInt	8 bajtova
		= 12 bajtova

Tabela Rating

Naziv obeležja	Tip podatka	Memorijsko zauzeće
ID	Integer	4 bajta
CreatedAt	Timestamp	8 bajtova
Feedback	Varchar(255)	100 bajtova (*)
Score	Integer	4 bajta
UpdatedAt	Timestamp	8 bajtova
Company	Integer	4 bajta
User	Integer	4 bajta
		= 132 bajta

Tabela RatingReasons

Naziv obeležja	Tip podatka	Memorijsko zauzeće
ID	Integer	4 bajta
Reason	Varchar(255)	100 bajtova (*)
		= 104 bajta

Napomena: (*) Za obeležja je izračunato prosečno memorijsko zauzeće.

Osnovna pretpostavka je sledeća:

- 100 000 kompanija koje prodaju opremu.
- 10 admina kompanije u proseku, dakle 1 000 000 admina kompanija.
- Svaka kompanija ima u proseku 300 tipova opreme, dakle postoji 30 000 000 tipova opreme.
- Postoji 1000 admina sistema, dakle preostaje 98 999 000 kupaca.

- Postoji 500 kupaca koji predstavljaju istu bolnicu, dakle postoji 198 000 bolnica.
- Svi podaci se čuvaju, izbegava se brisanje podataka.

Tabela	Početni broj	Godišnji procenat rasta	Broj torki nakon 5 godina	Memorijsko zauzeće
Company	100 000	10	150 000	25 MB
RegisteredUser	98 999 000	20	197 998 000	29 502 MB
Equipment	30 000 000	10	45 000 000	7 560 MB
CompanyAdmin	1 000 000	20	2 000 000	258 MB
SystemAdmin	1 000	20	2 000	0,2 MB
Role	3	0	3	69 B
UserRole	100 000 000	20	200 000 000	2 400 MB
Reservation	500 000	1200	30 500 000	1 342 MB
Hospital	198 000	20	396 000	19 MB
LoyaltyProgram	3	0	3	72 B
ReservationItem	1 750 000	1200	106 750 000	1 708 MB
Location	99 297 000	/	198 544 000	10 523 MB
Complaint	20 000	1200	1 220 000	268 MB
Rating	40 000	1200	2 440 000	322 MB
RatingReasons	20 000	1200	1 220 000	127 MB
				= 54 505 MB

Početni broj je broj na datum pravljenja projekcije za narednih 5 godina. Broj torki nakon 5 godina predstavlja broj u kojem se izračuna iznos procentualnog uvećanja početnog broja, sabere 5 puta i doda početni broj. Godišnji procenat rasta sagledava agregirano sve faktore koji utiču na porast podataka po tabeli nakon jedne godine.

Navedeni su neki od važnijih primera koji su potrebni za razumevanje određenih procenata godišnjeg rasta:

- Broj opreme će tokom godine svakako rasti, kako zbog novih kompanija tako i zbog dodavanja nove opreme.
- Generalno neki entiteti prestaju da budu aktivni i pristižu novi.
- Pretpostavka je da se 20% svih dostupnih termina (rezervacija) neće rezervisati i da se 10% svih rezervacija otkaže, što predstavlja 30% neiskorišćenih rezervacija. Prosečan broj stavki po rezervaciji je 5, što donosi 5x više *ReservationItem*-a u odnosu na iskorišćenu rezervaciju.
- Broj lokacija predstavlja zbir adresa prebivališta registrovanih korisnika, adresa bolnica i kompanija.
- Procenat žalbi u odnosu na sve rezervacije iznosi 4%.
- Procenat ocena u odnosu na sve rezervacije iznosi 8%.
- Procenat korekcije ocena u odnosu na sve ocene iznosi 4%.

Okvirna procena za skladištenje svih podataka u narednih 5 godina iznosi **54,5 GB**.

5. Predlog strategije za postavljanje load balancera

U slučaju otkaza *read-write* servera, jedna od repliciranih baza podataka preuzima ulogu vodeće. Replike će posedovati podjednaku hardversku moć, dok će *read-write* serveri posedovati veću hardversku moć, takođe međusobno jednaku.

HAProxy load balanser uz upotrebu *Geo-IP Routing* algoritma se postavlja između korisničkog zahteva i aplikativnih servera za potrebe preusmeravanja korisničkih zahteva ka sistemu koji će dati najbrži odziv. Kao drugi load balanser koji se postavlja od aplikativnog servera ka bazama podataka koristiće se *HAProxy* sa *Round Robin* algoritmom. U ovom slučaju *Round Robin* algoritam je dobar izbor zbog sličnih hardverskih resursa servera u odnosu na grupe kojima pripadaju.

S obzirom da je radi autorizacije i autentifikacije korišten *JWT* token koji je *stateless* time je situacija umnogome olakšana po pitanju komunikacije servera i praćenja sesije.

6. Predlog koje operacije treba korisnik nadgledati u cilju poboljsanja sistema

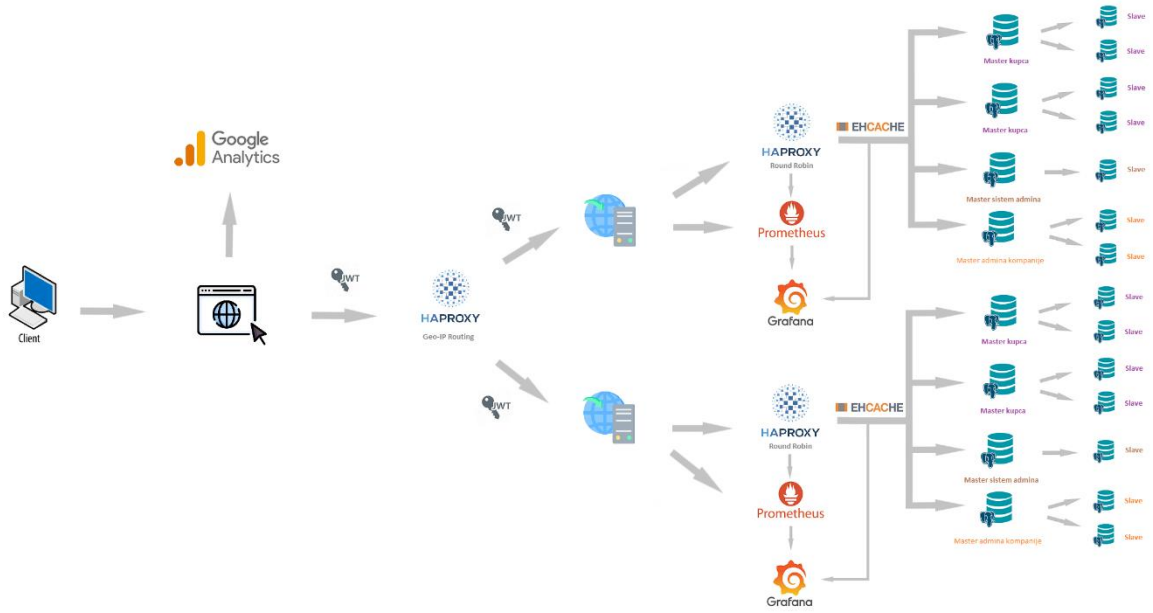
Cilj je da se na osnovu *Google Analytics-a* koji je jedan od najzastupljenijih alata u vezi statistike veb aplikacija uoče obrasci i pravila u ponašanju kupaca. Ideja je da se na osnovu uočenih obrazaca iskoriste velike količine podataka kako bi se na osnovu njih napravila grafovska baza. Grafovska baza podataka će se iskoristiti za sisteme preporuke. Kupcima će biti predlagana oprema koja se najčešće kupuje od kupaca sa sličnim karakteristikama. Još jedan sistem preporuke će biti iskorišten. Kupcima će biti nuđena oprema i drugih kompanija za koju se smatra da je relevantna na osnovu prethodnih kupovina posmatranog kupca, ali i ostalih relevantnih kupaca.

Za potrebe praćenja i adekvatnog reagovanja u vidu upravljanja serverima prilikom povećanja ili smanjanje broja zahteva, kao i otkaza koristiće se *HAProxy Fusion Control Plane*. Razlog za odabir ovog alata leži u činjenici da je u pitanju proizvod *HAProxy* platforme.

Za potrebe napredne distribuirana obrada i vizuelizacije podataka je potrebno koristiti Grafanu. U kombinaciji sa *Google Analytics-om* bi se mogao napraviti moćan spoj u praćenju, prikupljanju, obradi i analizi podataka. Primeri upotrebe, pravljenje adekvatnih reklama koje bi privukle nove kupce, pravljenja izveštaja o uspešnosti prodaje određene kompanije u odnosu na konkurente u vidu besplatnog saveta, kreiranja vizuelnog *dashbord-a* za administratore sistema sa statistikama u vezi žalbi i slično.

Prometheus bi se mogao iskoristiti za sve one delove sistema koji se ne mogu ispratiti upotrebom *HAProxy Fusion Control Plane-a,* poput praćenja broja poruka u redovima poruka.

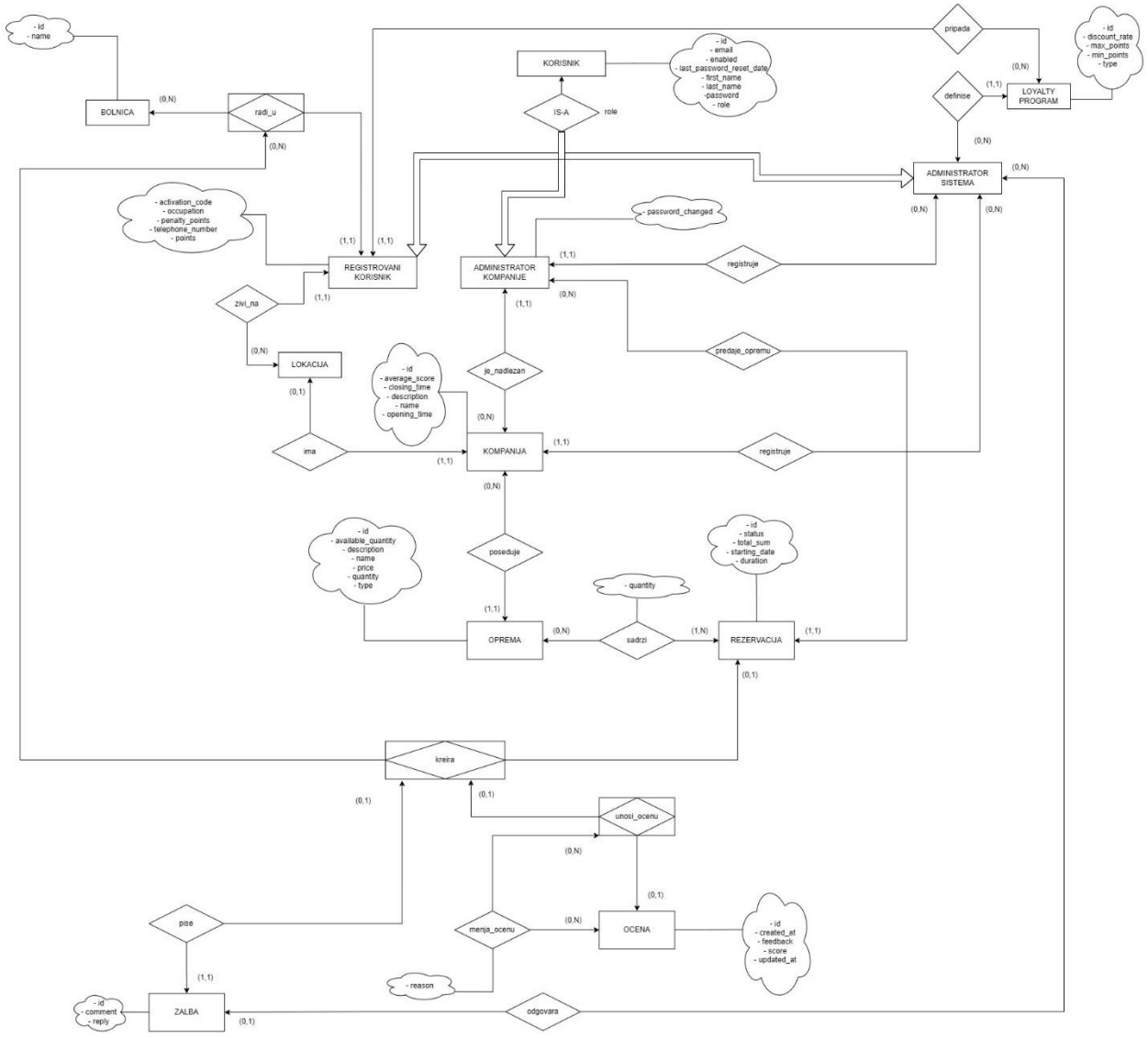
7. Kompletan crtez dizajna predlozene arhitekture



Slika 7.1: Dijagram predložene arhitekture

Na dijagramu je prikazana šema arhitekture za slučaj dva aplikativna servera sa pripadajućim sistemom baza podataka.

8. Dizajn šeme baze podataka



Slika 8.1: EER dijagram realnog sistema prodaja medicinske opreme