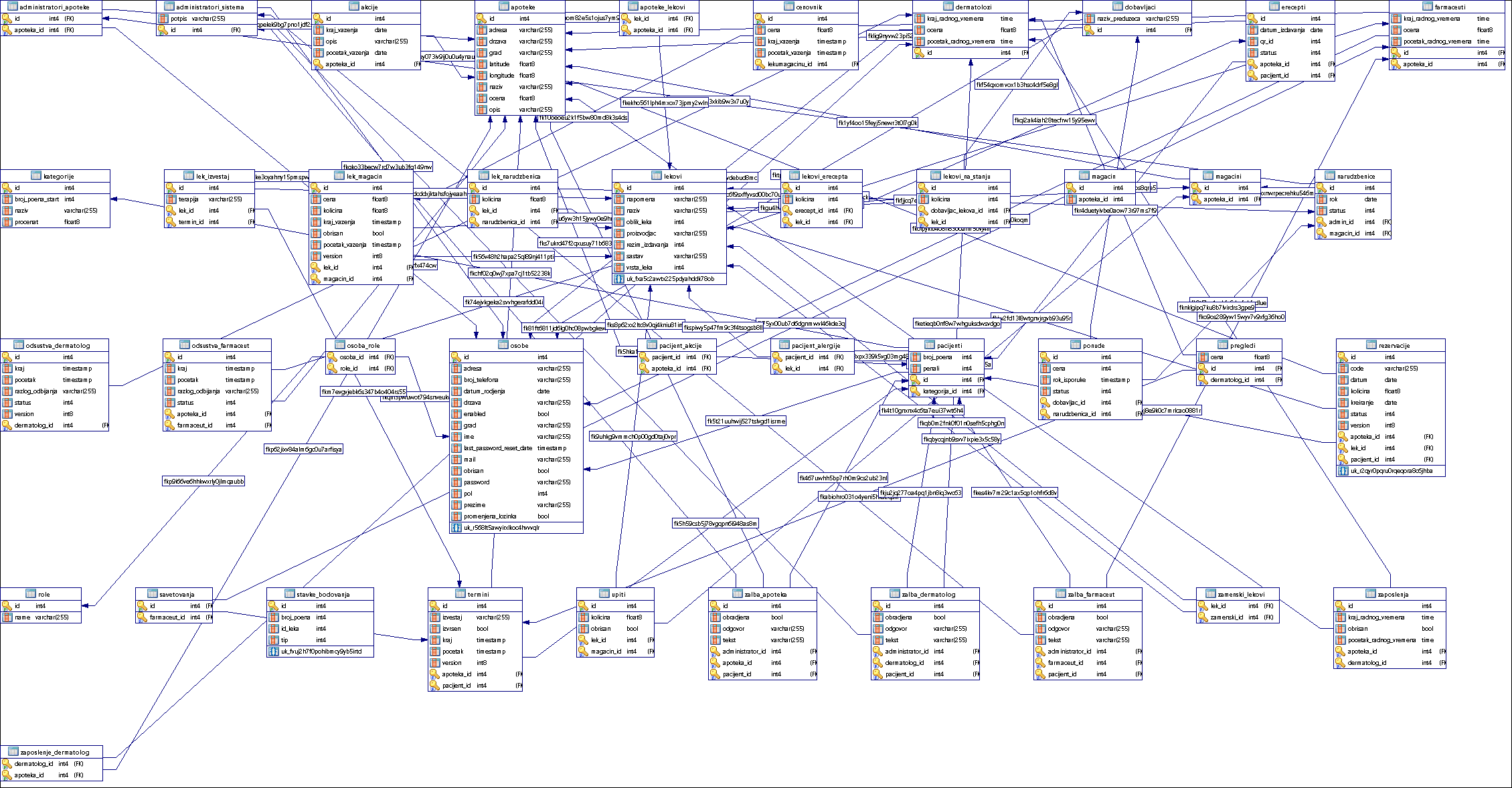
Proof of Concept

Tim 14

# Dizajn šeme baze podataka



# Particionisanje podataka

Particionisanje podatka predstavlja razdvajanje postojećih grupisanih podataka na zasebne particije, radi poboljšanja skalabilnosti, performansi, sigurnosti, fleksibilnosti i dostupnosti podataka u aplikaciji. Načini kojima se može pristupiti particionisanju podataka su horizontalni, verikalni i funkcionalni.

Načini na koje se particionisnaje podataka može primeniti na našu aplikaciju su sledeći.

Ogroman broj korisnika dovodi do ogromnog broja održanih termina( svaetovanja i pregleda) samim tim potrebno je optimizovati pristup tabeli baze koja sadrži ove podatke. Pre svega, moguće je kreirati zasebne particije na osnovu vremena izvršavanja termina. Češće će se vršiti pristup terminima koji su bliži trenutnom trenutku i u slučaju provera koje vrši aplikacija, ali i samim farmaceutima i dermatolozima su svakako bitnije svežije informacije. Pored ovoga, informacije o starijim terminima se mogu skladištiti na jeftinijim skladištima podataka, jer potencijalan gubitak ovih informacija nije katastrofalan. Na ovaj način smanjujemo trošak održavanja podataka o terminima. Takođe, poželjno bi bilo vršiti i particionisanje na osnovu sličnosti podataka koje termini nose. Npr. termini održani u određenom gradu, državi ili na manjem nivou apoteci ili skupini apoteka mogu biti smešteni u zasebne particije. Pored toga, ukoliko određene mreže apoteka žele da svoje podatke dodatno zaštite može se napraviti particija koja sadrži samo njihove termine radi povećanja sigurnosti tih podatka.

Osim termina, naša aplikacija radi sa velikim brojem korisnika i apoteka. Moguće su određene particije tabele apoteka na osnovu lokacije( veća šansa da će korisnici imati interakciju sa apotekama koje su im u neposrednoj blizini). Što se tiče kornisnika, određeno particionisanje podataka već postoji, na osnovu njihove uloge u sistemu. Dodatne particije koje se mogu uvesti su: particija koja sadrži samo username, password i role korisnika i particije na osnovu aktivnosti korisnika. Username, password i role su atributi korisnika kojima će se najčešće pristupati zbog prijave na sistem, međutim moguće je napraviti dodatnu particiju koja sadrži ime, prezime i lokaciju, jer su ovo podaci koji bi trebalo prvo prikazati, a onda po potrebi dobaviti ostale podatke. Što se tiče particionisanja podatka na osnovu aktivnosti korisnika, na ovaj način informacije o slabo aktivnim korisnicima možemo čuvati u jeftinijim skladištima podataka i smanjiti trošak održavanja podataka. Takođe, korisnici koji su slabo aktivni će mnogo ređe dolaziti u interakciju sa drugim korisnicima i samim tim njihovi podaci neće biti često potrebni.

Na kraju, treba i spomenuti rad sa lekovima. Velika popularnost aplikacije sa sobom donsi i više zainteresovanih farmaceutskih kuća koje žele da prodaju svoje lekove u apotekama na našem sistemu. Usled ovoga raznovrsnot lekova može eksponencijano rasti. Neke od particija koje možemo uvesti usled ovoga particije na osnovu potražnje za lekom i particija koja sadrži najbitnije informacije o leku. Svakako neće svi lekovi biti podjenako preporučivani i kupovani( što se može posmatrati na osnu rada aplikacije i izveštaja) samim tim lekove za koja je velika šansa da će njihovim informacija biti pristupano tokom termina i kupovine leka, možemo staviti u zasebnu particiju od onih za kojima je potražnja manja. Pored ovoga, često nam nisu potrebne sve informacije o leku. Farmaceuti i Dermatolozi iskustvom često napamet znaju informacije o većini lekova tako da im je potreban samo naziv, eventualno proizvođač leka. Takođe, bitan podatak je i količina leka na stanju jer će ovo biti proveravano u skoro svim akcijama sa lekovima.

# Replikacija baze i obezbeđivanje otpornosti na greške

Replikacija baze koja bi mogla da se primeni na našu aplikaciju je tipa transakcione replikacije. Pored ovoga prilikom dodavanja novog pomoćnog skladišta(subscriber) potrebno je kirstiti snapshot primarne baze.

Pošto je struktura naše baze statična i neće biti ponovnog dodavanja i brisanja kolona moguće je korišćenje replikacije bazirane na logovima. Ovaj pristup je malo komplikovaniji od ostalih jer zahteva postojanje log file, ali je najefikasniji pristup. Potrebno je posedovati log file u primarnoj bazi koji će sadržati ili sve izmenjenje redove ili same akcije izmene. Kada se započne proces replikacije, na svim pomoćnim bazama se prolazi kroz log file primarne baze i izvršavaju se izmene redom kojim su dadate u log file. Prednost ovog pristupa je to što se nikad ne šalju redovi koji nisu menjani, samim tim komunikacija između baza i proces replikacije su znatno efikasniji.

Sama frekvencija replikacije bi trebala da bude što veća, ali isto tako potrebno je naći balans sa performansama aplikacije. Čestom replikacijom povećavamo otpornost sistema na greške, jer u slučaju da primarni server padne, ostali serveri imaju veoma svežu kopiju podataka. Ovo nam je bitno jer naša aplikacija radi sa medicinskim podacima, koje se mogu smatrati veoma bitnim. Zbog ovog, preporučuje se i korišćenje pune replikacije.

# 4. Load Balancing

Veliki broj korisnika aplikacije podrazumeva potrebu za većim brojem servera, kao i njihov raspored na veoma širokom geografskom području. U cilju upravljanja saobraćajem, odnosno komunikacijom sa serverom, preporučuje se uvođenje load balancer-a. Load balancer preusmerava zahtev klijenta na server koji će obraditi taj zahtev. Cilj preusmeravanja zahteva je jednako raspoređivanje posla među serverima kako nijedan server ne bi bio preopterećen, a samim tim i nepouzdan.

Za load balancing postoje različiti algoritmi. Round Robin pristup je najmanje kompleksan za implementaciju. Glavna ideja ovog algoritma je da se zahtevi prosleđuju servirima jednom po jednom, do svakog servera. Mana algoritma je to što nema provere opterećenosti servera. Zahtevi koje klijent naše aplikacije može da uputi serveru se među sobom veoma razlikuju po pitanju potrebnog procesorskog vremena za obradu. Iz tog razloga, koristeći Round Robin pristup može doći do preopterećenja nekih servera. Takođe, uz veliki broj servera ne možemo očekivati da svi serveri imaju iste performanse, te je to još jedan razlog za odbacivanje Round Robin strategije.

Sa druge strane, metod najkaćeg vremena odziva je dobro rešenje za našu aplikaciju. Glavni razlog upotrebe ovog metoda su zahtevi različite kompleksnosti. Pomenuti metod se zasniva na proveri brzine odgovora na zahtev provere servera. Brzina odgovora servera je indikator koji pokazuje koliko je server opterećen. Uzimajući u obzir opterećenost servera, rešava se problem Round Robin algoritma nastao usled zahteva različite kompleksnosti. Dakle, serveri će biti opterećeni proporcionalno svojim performansama, te će opterećenje servera biti približno jednako raspoređeno. Ono što informacija o brzini odgovora servera sa sobom nosi je i informacija o očekivanom korisničkom iskustvu. S obzirom da aplikaciju koristi 200 miliona korisnika, interakcija sa korisnikom i njegovo iskustvo prilikom upotrebe aplikacije su važni, te je ovo još jedan razlog za upotrebu metoda najkaćeg vremena odziva.