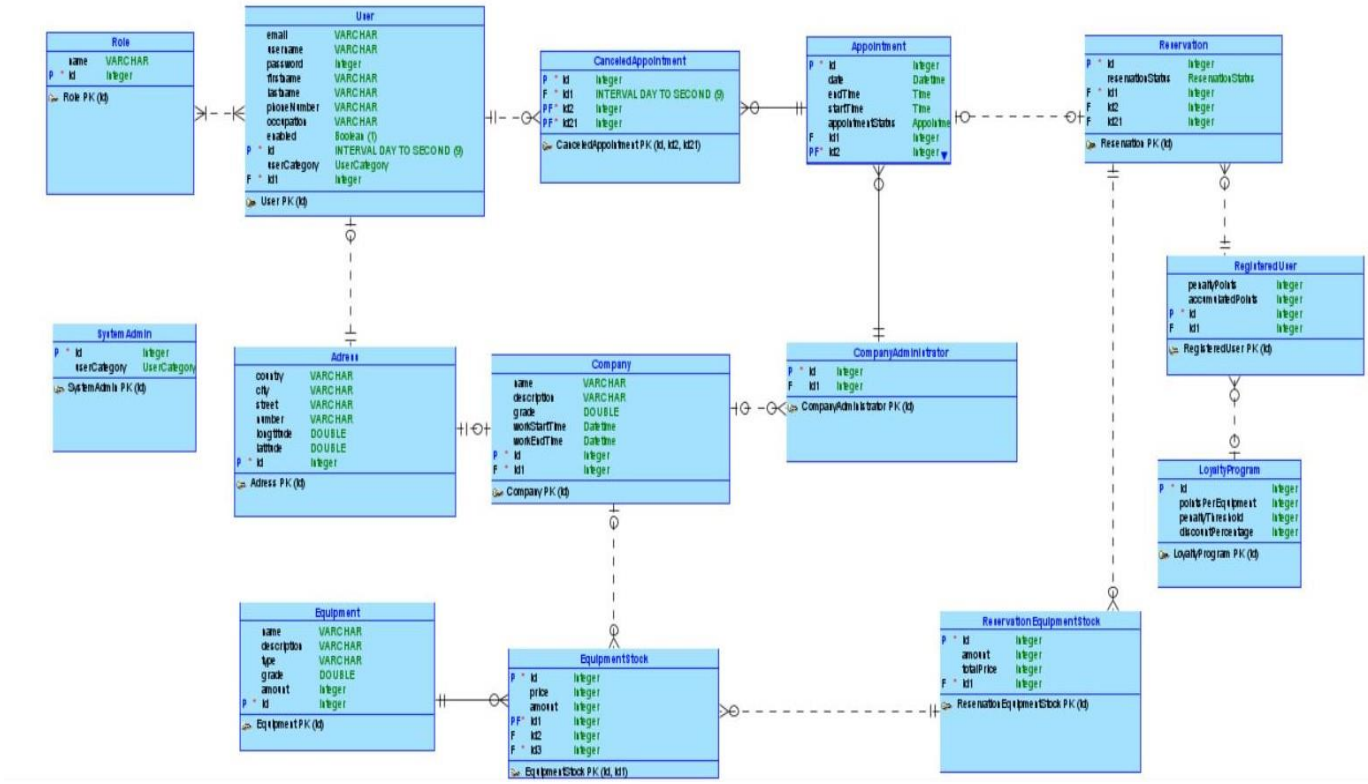
PREDLOG SKALABILNOSTI APLIKACIJE

SKALABILNOST BAZEPODATAKA

Što se tiče predloga skalabilnosti same baze podataka opredelile smo se za vertikalno skaliranje tj. davanje više resursa samoj bazi za funkcionisanje. Dakle, dodela više RAM memorije, procesorske moći, prostora itd.Za ovakvo skaliranje baze opredelile smo se zato što je sa aspekta vremena ovo najefikasnije rešenje. Ono što bismo takođe uradile, jeste dodavanje više servera u klaster tj. podigle bismo instancu



bekend server na više različitih portova,što bi nam kasnije omogućilo da dodamo LoadBalancer I uradimo replikaciju podataka.

PARTICIONISANJEPODATAKA

Što se tiče particionisanja podataka ovde bismo se opredelile za range partitioning obzirom da na mesečnom nivou imamo 500000 rezervacija. Napravile bismo particije tabele koja skladišti rezervacije, gde bi kljiuč bio datum termina za koji je rezervacija napravljena , pa bismo shodno tome imale dve tabele. U jednoj bismo skladištile podatke o rezervacijama koje su prošle a u drugoj buduće rezervacije. Na taj način bismo smanjili opseg u kojem tražimo podatke iz tabele.

Ovo ćemo pokazati na primeru tabele Appointment,gde bismo particionisanjem ove tabele na dve manje, termini koji su prošli i termini u budućnosti, smanjile bismo vreme obrade, ubacivanja novih podataka i izvlačenja podataka iz ovih tabela značajno.

REPLIKACIJA BAZE I OTPORNOST NA GREŠKE

Za ovaj slučaj predlažemo master-master replikaciju. Koristićemo replikaciju baze podataka gde bi korisnik u slučaju pada glavnog sistema I dalje mogao da koristi čitanje-pisanje u bazu. U slučaju master-slave korisnik prilikom pada neće moći da upisuje podatke u bazu već samo da čita iz nje. To je loše sa aspekta cilja naše aplikacije gde je akcenat na kreiranju rezervacija, dakle upisa u bazu.

Otpornost na greške bi bila realizovana putem Transakcione sigurnosti: Koristila bi se transakciona sigurnosti ACID transakcije koje pomažu da se osigura integritet podataka I da se spreče greške u slučaju problema sa hardverom ili mrežom.

Validacija unosa:Validiranje unosa podataka koji se unosi u bazu podataka,koristeći metode kao što su provere formata, kontrole duplikata i provere integriteta referenci, pomoći će da se spreče greške u unosu podataka.

Korišćenje replikacije:Korišćenjem već gore pomenute replikacije osiguraćemo dostupnost podataka i sprečiti gubitak podataka u slučaju gubitka nekog od servera.

KEŠIRANJE PODATAKA

Za keširanje bismo iskoristile distribuirano keširanje podataka,tj.na više računara bi se istovremeno čuvali keširani podaci što bi u mnogome povećalo performanse I skalabilnost jer bismo ovim keširanjem smanjile protok saobraćaja samim tim I troškove održavanja,takođe sa distribuiranjem keša na mnogo računara bismo mogli da primimo ogroman broj korisnika. Na našem primeru odlučile smo se da keširamo opremu zbog toga što ona sadrži samo osnovne podatke koji se jako retko menjaju a često dobavljaju.

PROCENA HARDVERSKIH RESURSA

Ako podelimo broj mesečnih rezervacija sa 30,dolazimo do cifre od oko 17500 rezervacija dnevno,što bi značilo da dnevno aplikaciju koristi oko 15 hiljada korisnika. Uzećemo po 5MB rama za svakog korisnika gde dolazimo do cifre od oko 75GB RAM memorije potrebne, zbog sigurnosti povećaćemo ovo na 100GB. Za skladištenje se treba opredeliti za SSD od oko 5TB (zbog sigurnosti) I brzinu konekcije od 10Gbit/s.

LOAD BALANCER

S obzirom da na sat vremena stigne minimum 700 rezervacija , jedan server koji bi dobijao toliko zahteva bi bio preopterećen. Morali bismo prvo podeliti naš backend server na više backend servera.Kada smo to uradili,iskoristićemo LeastConnections tehniku za load balancing.Dakle, ako podelimo naš server na 3 preko replikacije koju smo gore opisale, putem loadbalancer-a ćemo novi zahtev uvek slati onom serveru koji trenutno obrađuje najmanje zahteva. Ovime ćemo drastično povećati performanse i brzinu obrade samih zahteva.

SISTEMI NADGLEDANJA

Zakazivanje termina:Potrebno je nadgledati kako se korisnici kreću kroz process zakazivanja termina i koliko je korisnika završilo proces zakazivanja uspešno.

Otkazivanje termina:Potrebno je nadgledati koliko korisnika otkazuje svoje termine.

Rezervisanje opreme : Pratiti koja oprema se najčešće rezerviše, tj. kupuje.
Nadgledanjem ovih operacija korisnika,moguće je identifikovati problem sa korisničkim iskustvom i poboljšati sistem da bi se poboljšalo korisničko iskustvo. Takođe, korisničko ponašanje možemo pratiti uz alat kao što je Grafana koji nam daje vizualizaciju o najposećenijim stranicama. Dodatno, neke stavke koje bismo želeli da nadgledamo u cilju poboljšanja performansi su metrike poput CPU , memorija, broj HTTP zahteva.Sve ovo bi nam omogućilo open-source sistema za nadgledanje kao što je Prometheus.