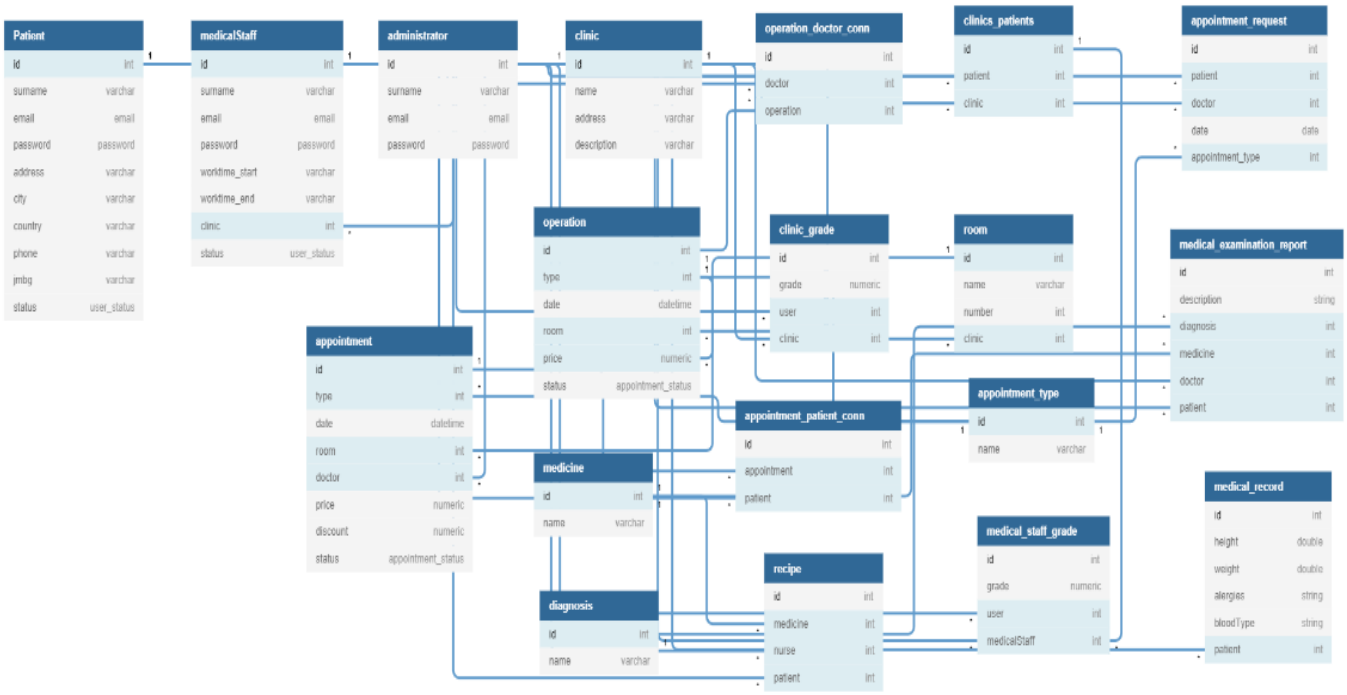
Dizajn seme baze podataka



Predlog za particionisanje baze podataka

U slucaju aplikacije koja sluzi za upravljanje klinickim centrom postoji vise slucajeva u kojima mozemo koristiti particionisanje podataka.

1. Prvi slucaj u kome mozemo koristiti particionisanje podataka bi bila tabela appointments, tokom vremena zbog prirode sistema u ovoj tabeli ce se naci veliki broj informacija o proteklim pregledima, kao i o buducim. Ovu tabelu bi bilo optimalno particionisati po svakom mesecu (*Horizontal Partitioning*), u tom slucaju kada administrator zeli da dobije pregled poslovanja klinike u odredjenom period pristup tim podacima bi bio mnogo efikasniji.

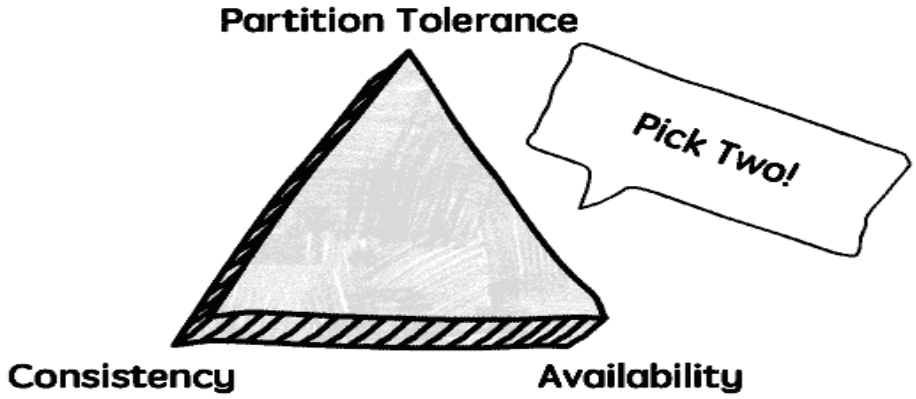
2. Takodje ovim nacinom particionisanja bi svi predstojeci pregledi bili u zasebnoj tabeli, te prilikom prikaza pregleda korisnicima, i doktorima sistem nema potrebu da prolazi kroz celu tabelu

appointments vec moze pristupiti samo tabeli u kojoj se nalaze pregledi. Ovim putem znacajno ubrzavamo pristup potrebnim pregledima.

3. Ovim pristupom se znacajno ubrzavaju sve izmene nad appointment tabelom.
4. Posto su pacijenti, doktori, medicinske sestre i administratori smesteni u istu tabelu po trenutnoj postavci projekta, morala bi se napraviti izmena toga i pored glavne tabele koja sadrzi sve korisnike sistema potrebno je napraviti zasebne tabele sa bitnim informacijama za svaku kategoriju korisnika. Jedini argument za zadrzavanje tabele u kojoj stoje svi korisnici je brze i prakticnije logovanje na sistem. Takodje u realnoj primeni sistema tabele svih korisnika bi najverovatnije imale veliki broj kolona jer bi bile potrebne razne informacije o istim. U ovom slucaju bi bilo dobro za potrebe povezivanja razlicitih tabela prilikom slozenih upita,napraviti particionisanje po kolonama od kojih bi postojala tabela koja bi u sebi sadrzala referencijalne integritete ka drugim tabelama zajedno sa id-ijem korisnika (*Vertical Partitioning*).

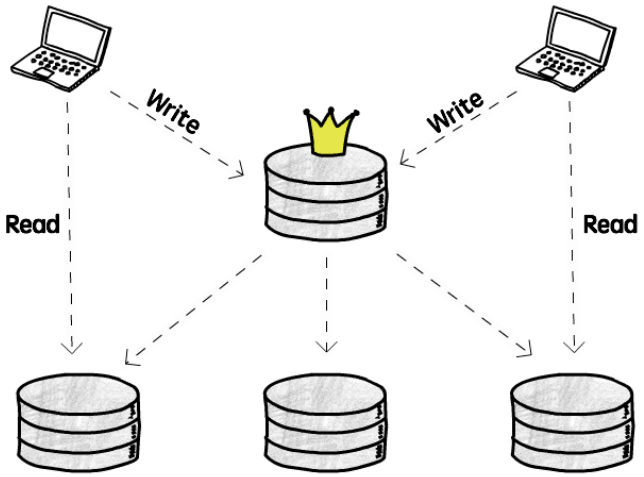
Predlog strategije za replikaciju baze i obezbeđivanje otpornosti na greške

1. Pre svega za sistem ove razmere potrebno je odrediti na koji nacin zelimo da vrsimo replikaciju I distribuciju nase baze podataka. Prema CAP teoremi (*consistency, availability and partition tolerance*) u ovakvom sistemu mozemo se odluciti za jedno od dva: konzistencije ili pristupacnosti. Svaki ima svoju prednost i u razlicitim delovima sistema cemo primenjivati ili jedno ili drugo.



2. Nacin na koji cemo vrsiti replikaciju nase baze podataka i raspored opterecenja je sledeci: unutar jednog klinickog centra vrsicemo SINGLE LEAD REPLICATION**,** odnosno imacemo jedan lead server koji ce primati sve zahteve koji ce vrsiti promenu nad bazom podataka,(INSERT,UPDATE,DELETE), dok ce sa ostalih servera moci da se vrsi samo citanje.

> These followers can, however, answer read queries.



Lead server ce vrsiti asinhrono replikovanje sa ostalim read serverima, u slucaju otkazivanja lead servera imacemo rezervni server koji ce vrsiti sinhrono preuzimanje podataka od lead servera koji ce imati ulogu da preuzme vodjstvo u slucaju otkaza prvog,vodicemo se pretpostavkom da nikada nece otkazati svi lead serveri, te ce se odabir sledbenika lead servera, u slucaju da bude potrebno, vrsiti automatski. Treba napomenuti da ovde pricamo o particionisanju podataka unutar jedne klinike, zato sto najveci deo podataka sa kojima ista radi se odnosi samo na nju, odnosno jednu kliniku ne interesuje kada neka druga klinika ima preglede u odredjenim terminima.

3. Sto se tice particionisanja podataka unutar klinickog centra, rezervni server koji ce imati ulogu da bude lead u slucaju otkaza glavnog lead servera ce imati zadatak da komunicira sa nekim od lead servera klinickog centra, i da salje informacije o svim promenama koje ce se tamo replikovati na sve servere radi zastite od gubitka podataka.
4. Bitno je napomenuti da su korisnici u stanju da vrse pretrage klinika, sto je jedan od izolovanih slucajeva kada je direktnim korisnicima sistema potrebno da imaju pristup “svim” podacima klinickog centra. U tom slucaju zahtevi ce se preusmeravati na servere klinickog centra sa kojih ce dobijati odgovor.
5. Ovakvim nacinom distribuiranja sistema postizemo balansiranje opterecenja i sprecavamo zagusenje sistema u kriticnim peak-ovima.

Predlog strategije za kesiranje podataka

1. Podaci koje bi bilo potrebno kesirati na nivou klinickog centra su sve klinike koje sistem sadrzi, kao i predefinisani pregledi koje korisnik moze da zakaze, na nivou odredjene klinike ukoliko je moguce bilo bi optimalno kesirati i doktore, kao i slobodne preglede za odredjenu kliniku.

Predlog koje operacije korisnika treba nadgledati u cilju poboljsanja sistema

1. Operacije korisnika koje treba nadgledati su vreme koje je potrebno korisniku od vremena logovanja da dodje do kljucnih informacija, odonosno vreme koje je potrebno da pretrazi klinike, doktore, zakaze predefinisan pregled i tako dalje, potrebno je napraviti Keystroke-level model da bi bilo moguce predvideti vreme koje ce korisniku biti potrebno da izvrsi odredjene operacije, ali posto to ne moze biti jasan pokazatelj bitno je pratiti proces kroz koji korisnik prolazi kada zeli da izvrsi odredjenu aktivnost i koliko mu je zaista vremena potrebno. Ovo sve treba raditi u cilju otkrivanja koji konkretni delovi sistema su u globalu mozda loze dizajnirani ili nisu dovoljno jasni korisniku da bi uspeo da se lako snadje.

Load Balancing

1. Za load balancing nase mreze najbolje bi bilo kada bi svaki server slao informacije o tome koliko je opterecen. Nakon toga zahteve bi slali na server koji je u tom trenutku najmanje opterecen. Pored toga trebalo bi primeniti i Performance Traffic Routing koja bi zahteve slala na onaj server koji je po geografskom polozaju najblizi. Ovakva kombinacija metoda razmatra koji server je najblizi, ali i najmanje opterecen.

Procena potrebnog hardvera za period od pet godina

1. Prva tri servera koja su povezana sa load balancer-om kao I sve trojke database servera osim one koja je skroz desno na slici predlozene arhitekture treba da imaju 4GB RAM-a, dok tri Database servera koja su krajnje desno na slici treba da imaju 8GB RAM-a, jer se na njima nalaze podaci celog klinickog centra. Za 200 miliona korisnika bice potrebno oko 200TB za skladistenje svih podataka.

Dizajn predlozene arhitekture