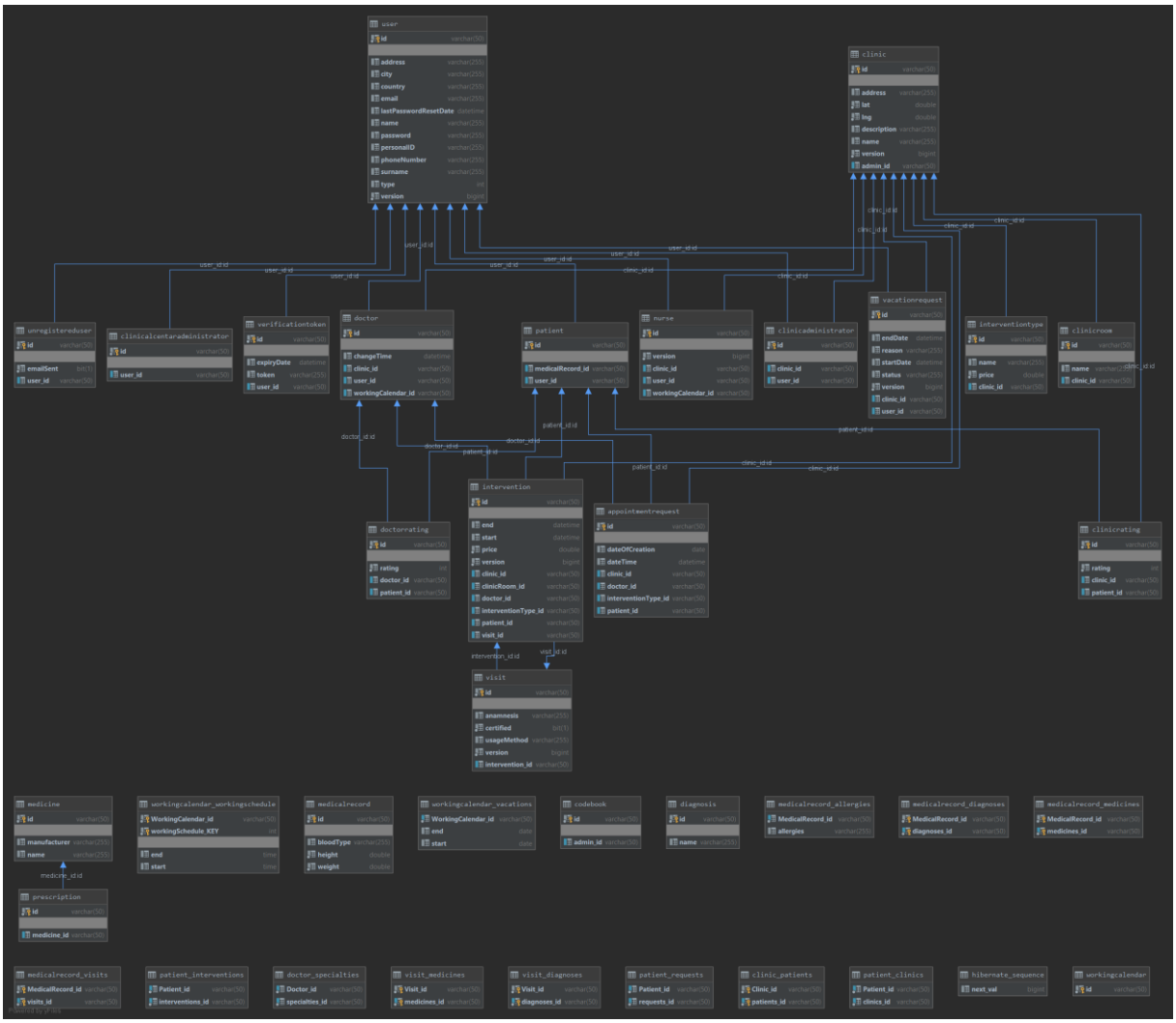
Dizajn šeme baze podataka



Jedina stvar koja bi se, u odnosu da šemu sa te slike razlikovala, bilo bi modelovanje zahteva za pregled i samog pregleda. U tom slučaju, zahtev za pregled bi u stvari bio samo dio tabele *Intervention*, kod kojega bi jedino polje vezano za salu klinike bilo prazno.

Particionisanje podataka

U svrhu optimalnijeg korišćenja resursa, bilo bi potrebno odraditi i tzv. particionisanje baze podataka.

S obzirom na to da je primarna svrha aplikacije u održavanju evidencije o intervencijama i posetama, očekuje se da će skoro svaki od 200 miliona korisnika imati odgovarajući zahtev za pregled, pregled i posjetu ljekaru. Kako su u priloženoj šemi baze podataka sve tri varijante objedinjene jednim jedinim tipom, prije svega bi se podaci koji se odnose na preglede mogli particionisati tako što bi se odvojile tabele na zahteve za pregled, predefinisane preglede, neobrađene i obrađene preglede. U tom slučaju bi se jednostavnije moglo pristupati konkretno određenom skupu elemenata. Takođe, sve četiri navedene isparticionisane grupe bi bilo moguće dalje particionisati horizontalno:

- Dio za zahtjev za pregled po klinici;

- Dio predefinisani pregled, takođe, po klinici;
- Dio koji se odnosi na pregled mogao bi se particionisati na ljekare.

Vrlo bitan dio aplikacije predstavlja i nalog korisnika. Kako bi se prilikom prijave vrlo često pristupalo tom dijelu zarad dobavljanja osnovnih informacija, svakako bi bilo dobro odraditi vertikalno particionisanje po najbitnijim informacijama, odnosno da se dijelovi koji se odnose na lozinku, imejl, ime i prezime nađu u jednom dijelu, a svi ostali u drugom dijelu particionisanih tabela.

Potencijalno, korisnici bi se mogli particionisati i horizontalno po datumu zamene lozinke (ili pak uvesti dio koji označava datum poslednje prijave), jer se pretpostavlja da će korisnici koji su skorije mijenjali lozinku, u naredno vrijeme češće koristiti aplikaciju.

Replikacija baze podataka

Radi jednostavnijeg postavljanja sistema, a pritom njegovog korišćenja sistema i lakšeg očuvanja konzistentnosti sistema smatra se da bi *master-slave* replikacija bila odgovarajuća strategija. Zahtevi za čitanje bi se preusmeravali na *slave*, dok bi *master* upisivao podatke i prosljeđivao ih *slave* dijelovima baze podataka.

Keširanje podataka

Keširanje podataka na prvoj liniji odbrane, nakon slanja zahteva kontroleru, može biti dodato u spring na jednostavan način. Jedan od najpopularnijih načina za keširanje bi bio dodavanjem EhCache podrške. Njegovom upotrebom, ono što se na osnovu upućenih klijentskih zahteva vraćeno, bilo bi keširano i zapamćeno za narednu upotrebu. Time bi se rasteretio sam rad nad bazom podataka. Prije svega, bilo bi najbitnije keširati zahteve koji se odnose na objekte kojima bi se najviše pristupalo. Pretpostavlja se da bi to bilo dobavljanje klinika, dobavljanje ljekara i dobavljanje predefinisanih intervencija.

Procena hardverskih resursa

Aplikacija ima 200 miliona korisnika. Podaci koje nalog korisnika zauzme jesu podaci o nalogu korisnika i njegovom zdravstvenom kartonu. Na osnovu analize izvezenih datoteka, ovakav nalog bi zauzeo u proseku 3KB. Za 200 miliona takvih naloga, potrebno je 600.000.000 kilobajta, odnosno 600GB memorije. Kako se ti nalozi možda neće stvoriti odjednom u prvih godinu dana od nastanka aplikacije, kroz aplikaciju je potrebno da prođe još toliko podataka u vidu zahteva za registraciju. Ukoliko se, u najgorem slučaju, ti zahtevi ne brišu, to je još 400GB zauzeća (zbog nedostatka kartona).

Ako klinički centar ima 200 miliona onlajn korisnika, to će podrazumevati da se radi o nekom ogromnom državnom kliničkom centru koji ima korisnika skoro kao što treća država po broju stanovnika. Stoga se pretpostavlja broj klinika jedne države. Kako u Sjedinjenim Američkim Državama ima oko 5.000 klinika, pretpostavimo da će ovaj klinički centar imati oko 3.000 klinika.

Podaci jedne klinike će, zbog adrese, zauzimati nešto više od 4KB. Svaka klinika ima u proseku 20 sala veličine 1KB, to se prosečna klinika svodi na 24KB podataka koje zauzima, odnosno 72MB za sve klinike. Svaka od tih klinika imaće ocenu (0,5KB i za kliniku i za ljekare), ljekare (oko 50 po 3KB), medicinske sestre (oko 100 po 3KB), te se za sve to može procijeniti zauzeće

na 3000 * (24KB + 3KB * 50 + 3KB * 100 + 0,5KB * 50 * 50.000+ 0,5KB * 50.000), što je ukupno nešto manje od 4TB.

Najbitniji dio klinike je pregled kod ljekara. Ukoliko jedan prosečan čovek ide kod ljekara četiri puta godišnje, sa bilo čije strane zakazan pregled, to bi predstavljalo oko 4*4KB zauzeća za svakog korisnika godišnje. To je 16TB zauzeća za tih 5 godina, za svih 200 miliona korisnika.

Neka postoji još skoro konstantno zauzeće od par hiljada lijekova i dijagnoza, što bi izašlo na par gigabajta.

Stoga, može se procijeniti da bi zauzeće bilo nešto više od 20,5TB skladišta podataka.

Postavljanje load balancera

U slučaju ovakvog sistema, interakcija sa aplikacijom je precizna i vrlo kratka sa strane korisnika. Konekcije sa serverom su stoga prilično brze, pa se bolje u ovom slučaju možda opredijeliti za Least response time metod uravnoteženja. U ovom slučaju bi se server sa najkraćom obradom signala birao za uravnoteženje.

Dizajn arhitekture