## 1. Партиционисање података

### 1.1. Табела ZAHTEV\_REGISTRACIJA

Ако је претпостављен број **регистрованих** корисника 200 милиона, можемо очекивати да ће број пристиглих захтева за регистрацију у секунди бити значајно велики. Зато се може десити да у једном третку број редова ова табеле буде јако велики. Из тог разлога предлажемо хоризонтално партиционисање ове табеле по распону вредности колоне BROJ\_OSIGURANIKA.

### 1.2. Табела ZAHTEV\_POSETA

Пошто је претпостављен број регистрованих корисника 200 милиона, можемо сматрати да ће број пристиглих захтева за посету бити значајно велики, поготово мислимо на оне захтеве које пацијент шаље, јер он то може да уради у сваком тренутку. Зато предлажемо хоризонтално партиционисање ове табеле према колони KLINIKA, које би затим по потреби биле хоризонтално партиционисане према распону вредности колоне DATUM.

### 1.3. Табела ZAHTEV\_ODMOR

За ову табелу предлажемо хоризонтално партиционисање према колони KLINIKA, а након тога по потреби хоризонтално партиционисање према колони ODOBREN.

### 1.4. Табела TIP\_POSETE и табела SALA

За ову табеле предлажемо хоризонтално партиционисање према колони KLINIKA, а након тога по потреби хоризонтално партиционисање према колони AKTIVAN.

### 1.5. Табела POSETA

Партиционисање које предлажемо за ову табелу је хоризонтално партиционисање према колони DATUM.

### 1.6. Табела LEK, DIJAGNOZA

Вршили бисмо хоризонтално партиционисање ових табела према распону колоне SIFRA.

### 1.7. Табела KLINIKA

Претпостављамо да број редова у овој табели не би био значајно велики (односно да наш клинички центар не би имао на пример 100 регистрованих клиника), а како табела има само четири колоне, не бисмо вршили њено партиционисање.

### 1.8. Табела KARTON

Урадили бисмо хоризонтално партиционисање према колони KRVNA\_GRUPA, а након тога према распону вредности колоне BROJ\_OSIGURANIKA.

### 1.9. Табела IZVESTAJ

Урадили бисмо хоризонтално партиционисање партиционисање према распону вредности колоне TERAPIJA.

### 1.10. Табела KORISNIK

Пошто ова табела има значајно већи број колона од осталих, урадили бисмо вертикално партиционисање на две табеле – прва би садржала колоне ID, EMAIL, LOZINKA, IME, PREZIME, DRZAVA, GRAD, ADRESA, TELEFON, AKTIVAN, а друга табела би садржала остале колоне. Разлог за то је што се наведене колоне често мењају, па је оваквим партиционисањем оптимизована брзина read-write операција. Додатно, урадили бисмо хоризонтално партиционисање прве табеле према распону колоне DRZAVA, и хоризонтално партиционисање друге табеле према колони TIP – пацијент, лекар, администратор клинике, медицинска сестра или администратор клиничког центра.

### 1.11. Табела OCENA

Како је максималан број редова ове табеле два пута број регистрованих пацијената целог клиничког центра, неопходно ју је партиционисати и предлажемо хоризонтално партиционисање према распону вредности колоне PACIJENT.

**Напомена:** Користили бисмо више машина код партиционисања.

## 2. Процена хардверских ресурса у наредних 5 година

Нека у просеку један корисник на један дан унесе у базу до 100KB података (зато што у апликацији нема чувања меморијски захтевних садржаја као што су слике и видео). Помножимо тај број са 30 (број дана у месецу), њега са 12 (број месеци у години), њега са 5 (тражени број година) и њега са 200 милиона (број регистрованих корисника). Укупно, то представља 32.74PB.

## 3. Нагледање операција корисника

Увешћемо активно посматрање операција корисника, како бисмо могли унапредити предложене стратегије за постизање скалабилности система. Можемо мерити следеће параметре:

* Број нових захтева за регистрацију по дану
* Број нових захтева за регистрацију из одређених подручја
* Број нових захтева за преглед/операцију по секунди
* Број нових захтева за преглед/операцију по сату
* Број нових захтева за преглед/операцију по дану

На основу резултата можемо да видимо да ли је потребна нова стратегија за кеширање, пертиционисање или репликацију...

## 4. Предлог стратегије за кеширање

У сврху кеширања користићемо готова решења попут Memcached и Redis да чувамо целе објекте у кешу. Тако ће се пре упита у базу прво проверавати да ли тражени објкеат постоји у кешу, ако га нема ту, тек се онда приступа бази.

За корисника који је пацијент кеширали бисмо историју болести (посете које је обавио) зато што претпостављамо да ће пацијент често приступати тој страници, како би имао бољи увид у своје здравствено стање. Кеширали бисмо и клинике јер претпостављамо да би пацијент често приступао страницама које садрже листу свих клиника, да би их претражио или нашао неки повољан термин.

За сестру кеширали бисмо неоверене рецепте. За лекара кеширали бисмо листу пацијената који су га некад посетили. За админа кеширали бисмо оцене из базе које се односе на његову клинику, јер претпостављамо да би он често желео да види извештај пословања своје клинике. Кеширали бисмо захтеве за посете, типове посета и сале. Типове посета бисмо кеширали јер се оне користе кад код се креира нови лекар (унос специјализације), када се креира предефинисани преглед, а сале због креирања предефинисаног прегледа и обраде захтева за преглед/операцију.

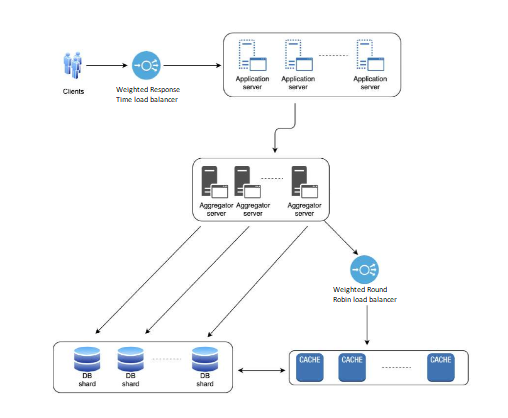
## 5. Репликација базе и отпорност на грешке

Постојао би један главни сервер базе који би садржао све податке клиничког центра и један backup те базе, која би била read-only и служила би за опоравак система ако би главна база отказала . Постојао би још један backup који би садржао податке који нису везани за неку клинику – то су регистровани пацијенти и супер админи и захтеви за регистрацију. Пошто претпостављамо да укупан број клиника центра не би био значајно велики, предлажемо постојање backup сервера и главног сервера базе за сваку клинику, који би садржао податке везане само за њу. Backup база би била read-only и служила би за опоравак система у случају отказа.

## 6. Стратегија за постављање load balanserа

Користили бисмо „Weighted Response Time“ технику за постављања load balansera између корисника и апликативних сервера. Овај метод ослања се на информације које нам даје периодични „server health check“, а који служи да нас обавести који сервер за најкрађе време обрађује пристигле захтеве. Онај који најбрже одговори, њему се прослеђује новопристигли захтев. Код load balansera између агрегационих сервера и сервера за кеширање користићемо стратегију „Weighted Round Robin“ која се од обичне round robin стратегије разликује по томе што је сваком серверу додељен тежински број, сервери за већим тежинским бројем имају предност приликом усмеравања захтева ка њима.

## 7. Дизајн предложене архитектуре



Слика 1. Дизајн предложене архитектуре