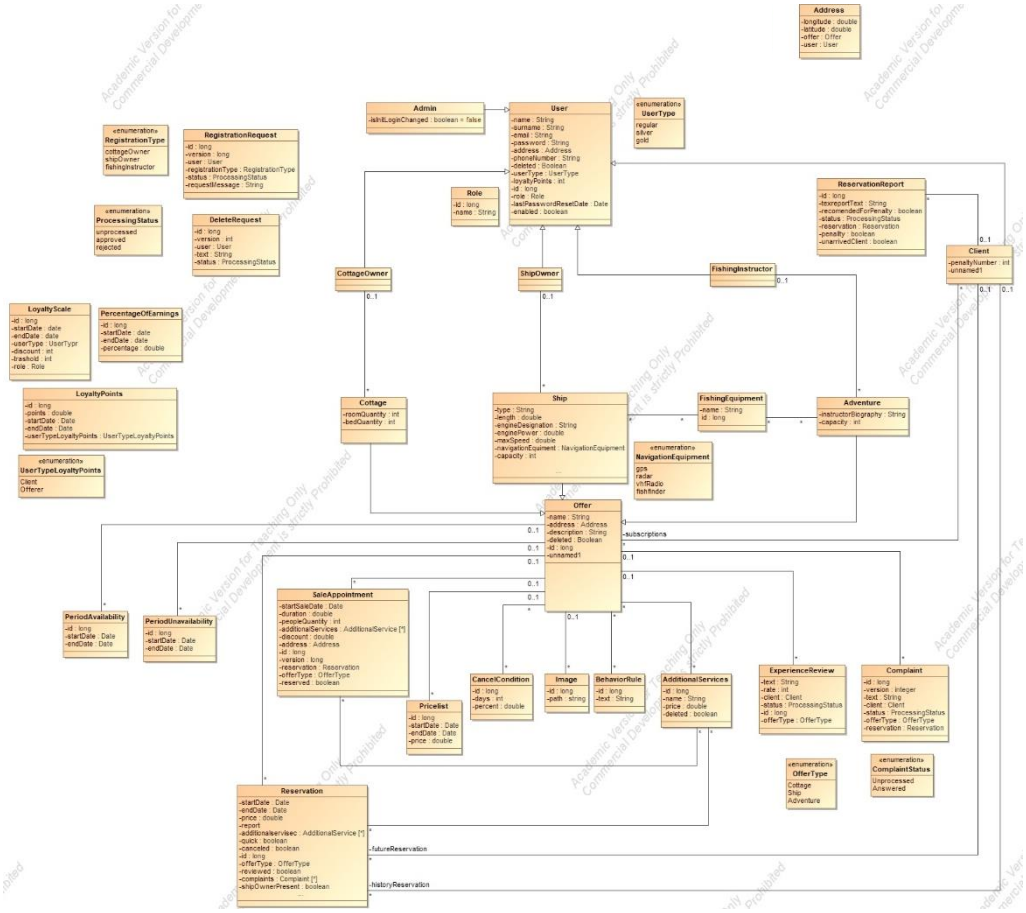
Skalabilnost

1. Dizajn šeme baze podataka



Slika 1 Dizajn šeme baze podataka

2. Particionisanje podataka

Particionisanje se često primenjuje u cilju poboljšanja perfomansi, pristupačnosti, pouzdanosti sistema sa velikom količinom podataka.

Postoje dve vrste particionisanja:

Horizontalno - podela podataka u kojoj se različiti redovi nalaze u različitim tabelama, a razvrstavanje se vrši prema nekom kriterijumu

Vertikalno - struktura tabele se deli vertikalno, jedan deo kolona strukture relacione tabele se čuva u jednoj bazi podataka, a druga baza podataka sadrži drugi deo kolona. Kriterijum za vertikalnu podelu može biti frekvencija pristupa i izmene podataka.

Tabela *users*

Naša aplikacija treba da podrži 100 miliona korisnika. Tabela users sadrži sve korisnike sistema, tako da će ona imati veliku količinu podataka, pa performanse neće biti na prihvatljivom

nivou. Kako bi poboljšali performanse, predlažemo horizontalno particionisanje ove tabele po tipu korisnika(klijent, administrator, vlasnik vikendice, vlasnik broda, instruktor).

Tabele *reservations* i *sale_appointment*

Naša aplikacija treba da podrži milion rezervacija na mesečnom nivou. Sve rezervacije se nalaze u tabeli reservations, tako da će operacije nad rezervacijama biti usporene. Predlažemo horizontalno particionisanje tabele reservations po datumu početka rezervacije, tako da se razdvoje prošle i buduće rezervacije. Isto ovo možemo primeniti i na tabelu sale_appointments, u kojoj se čuvaju brze rezervacije, koje kreiraju vlasnici ponuda.

3. Replikacija baze podataka i otpornost

U našoj aplikaciji je bitno da uvek imamo ažurne podatke jer želimo da nas korisnici naše aplikacije smatraju pouzdanim, pa tako ne bi trebalo da dolazi do situacije da se za isti period naprave dve rezervacije. Da bismo ovo obezbedili *Transactional Replication* (Replikacija zasnovana na transakcije), koji nam omogućava replikaciju podataka blizu realnog vremena. Tehnika *Transactional Replication* isto omogućava i visoku dostupnost.

Sa 100 miliona korisnika, očekivano je da će postojati veći broj serverskih mašina rasprostranjenih po celom svetu. Svaka serverska mašina mora da poseduje kopiju podataka kako bi se održala konzistentnost samog sistema. Strategija koju predlažemo se bazira na propagaciji izmena. Ovim pristupom se smanjuje opterećenje mreže pošto se samo pri promeni propagira izmena. Pošto se ovaj sistem sastoji od većeg broja servera koji su međusobno povezani, pri otkazivanju jednog servera treba obezbediti da sistem i dalje normalno funkcioniše uz lošije performanse.

4. Predlog strategije za keširanje podataka

Keširanje podataka nam omogućava smanjenu potrebu za pozivama baze.

```
<expiry>
    <ttl unit="hours">1</ttl>
</expiry>
```

Slika 2 Maksimalno vreme koje objekti mogu da provedu u kešu

```
<cache alias="adventure" uses-template="default">
    <key-type>java.lang.Long</key-type>
    <value-type>com.project.mrsisa.domain.Adventure</value-type>
    <resources>
        <heap>20</heap>
    </resources>
</cache>
```

Slika 3 Maksimalno 20 adventure objekata mogu da se nalaze u kešu

```
@Cacheable("adventure")
public Adventure findOneById(Long id) {
    return adventureRepository.findOneById(id);
}
```

Slika 4 Dodavanje objekta adventure u keš

```
@CacheEvict(cacheNames = {"adventure"}, allEntries = true)
public void removeFromCache(){

}

@CacheEvict(cacheNames = {"adventure"},key = "#id")
public void removeOneFromCacheById(@Param("id") long id){

}
```

Slika 5 Pražnjenje keša od svih objekata tipa adventure i brisanje objekta adventure sa određenim idnetifikatorom

Implementirano je keširanje entiteta (vikendica, avantura, brod), bez atributa koji su predstavljeni u formi listi (kao što su dodatne usluge i pravila ponašanja). Za poboljšanje performanse predložimo keširanje i ovih entiteta, za brži pristup.

Za odabir kandidata predložili bismo korišćenje pristupa zasnovanog na prediktivnom keširanju unapred, pretpostavljajući šta može zanimati korisnika. I kasnije kašširati na zahteve korisnika.

Ako dolazi do prekoračenja memorija predviđenih keširanih podataka, predlažemo korišćenje pristupa *Least Recently Used*, koji nam omogućava praćenje trendova, koji se pojavljuju kod korisnika.

5. Procena hardverske resurse

Smatramo da neće broj redova svih tabela rasti istim tempom. Tabele kao što su FishingEquipment, AdditionalServices i BehaviorRules smatramo da neće imati više od 100 redova pa njihovo memorijsko zauzeće smatramo zanemarljivim u odnosu na tabele koje će brojati milione kolona. U nastuvku je data analiza zauzeća memorije po tabelama.

Ukupan broj korisnika aplikacije – 100 miliona

Broj rezervacija svih entiteta na mesečnom nivou je – 1 milion

Users – procena broja korisnika je 100 miliona, jedan korisnik približno zauzima 300B pa je ukupno zauzeće ~ 30GB

RegistrationRequest – smatramo da će 10% svih korisnika biti vlanici/instruktori pa je procena broja zahteva za registraciju ~10 miliona. Smatramo da će polovinu svih vlasnika činiti vlasnici vikendica ~5 miliona, zatim instruktori ~ 3 miliona i na kraju vlasnici brodova 2 miliona. Svaki zahtev za rezervaciju za zauzima ~200B pa je ukupna procena zauzeća memorije ~ 2GB

Adventure – smatramo da će svaki instructor prosečno imati 5 svojih avantura, kako ima ~3 miliona instruktora, biće ~ 15 miliona avantura. Jedna avantura zauzima ~600B pa je zauzeće pocenjeno na ~8,5GB

Cottage – smatramo da će svaki vlasnik vikendice imati po 3 vikendice u ponudi što je ~15 miliona vikendica. Jedna vikindica zauzima ~600B, što je ukupno ~8.5GB

Ship – smatramo da će svaki vlasnik broda imati prosečno po 2 broda u ponudi što je ~ 4 miliona brodova. Svaki brod zauzima ~600B pa je ukupno zauzeće ~ 2.3 GB

PeriodAvailability - smatramo da će svaki entitiet imati po deset perioda dostupnosti (2 godišnje) što je ~34 miliona (vikendice, brodovi i avanture zajedno) * 10 ~ 340 miliona redova. Jedan perod dostupnosti zauzima ~56B pa je procena ukupnog zauzeća memorije ~ 18 GB

PeriodUnavailability – smatramo da će svaki entitet imati po dva perioda nedostupnosti godišnje pa je ukupan proj perioda nedostupnosti za 5 godina ~680 miliona čije je zauzeće memorije ~ 36 GB

Reservation – mesečno se očekuje 1 milion rezervacija pa se za 5 godina očekuje ~ 60 miliona rezervacija. Jedna rezervacija zauzima ~96B pa je ukupno zauzeće ~ 5.36 GB

SaleAppointment – smatramo da će 10% svih rezervacija biti brze rezervacije što je 6 miliona. Jedna brza rezervacija zauzima 104B pa je ukupno zauzeće ~0.58 G

Experience review – očekujemo da broj komentara bude približno jednak broju rezervacija ~60 miliona. Jedan komentar zauzima ~300B, procenjeno ukupno zazeće je ~18GB

ReservationReport – očekujemo da broj izveštaja odgovara broju rezervacija ~60 miliona. Jedani izveštaj zauzima ~200B pa je ukupno zauzeće ~ 12GB

Pricelist – očekujemo da će svaki entitet izmeniti bar 5 cena tako očekujemo 34 miliona * 5 ~ 170 miliona cenovnika. Jedan cenovnik zauzima 64B pa je ukupno zauzeće ~ 11GB

Image – očekujemo da će svaki entitet imati 4 slike pa je očekivan broj slika ~ 136 miliona. Jedna slika zauzima 88B pa je ukupno zauzeće ~ 12GB

CancelConditions – očekujemo da svaki entitet imati po 4 uslova za otkaz pa je ukupno uslova za otkaz ~ 136 miliona i očekivano zauzeće je ~7GB

Procenjeno ukupno zauzeće memorije za aplikaciju je ~ 200GB

6. Predlog strategije za postavljanje load balancer-a

Zbog brojnosti korisnika, naša aplikacija će imati više servera. *Load balancer* će raspoređivati zahteve na osnovu predložene strategije, *Weighted Least Connection*. Ova tehnika predstavlja balans između karakteristika servera i broja aktivnih konekcija.

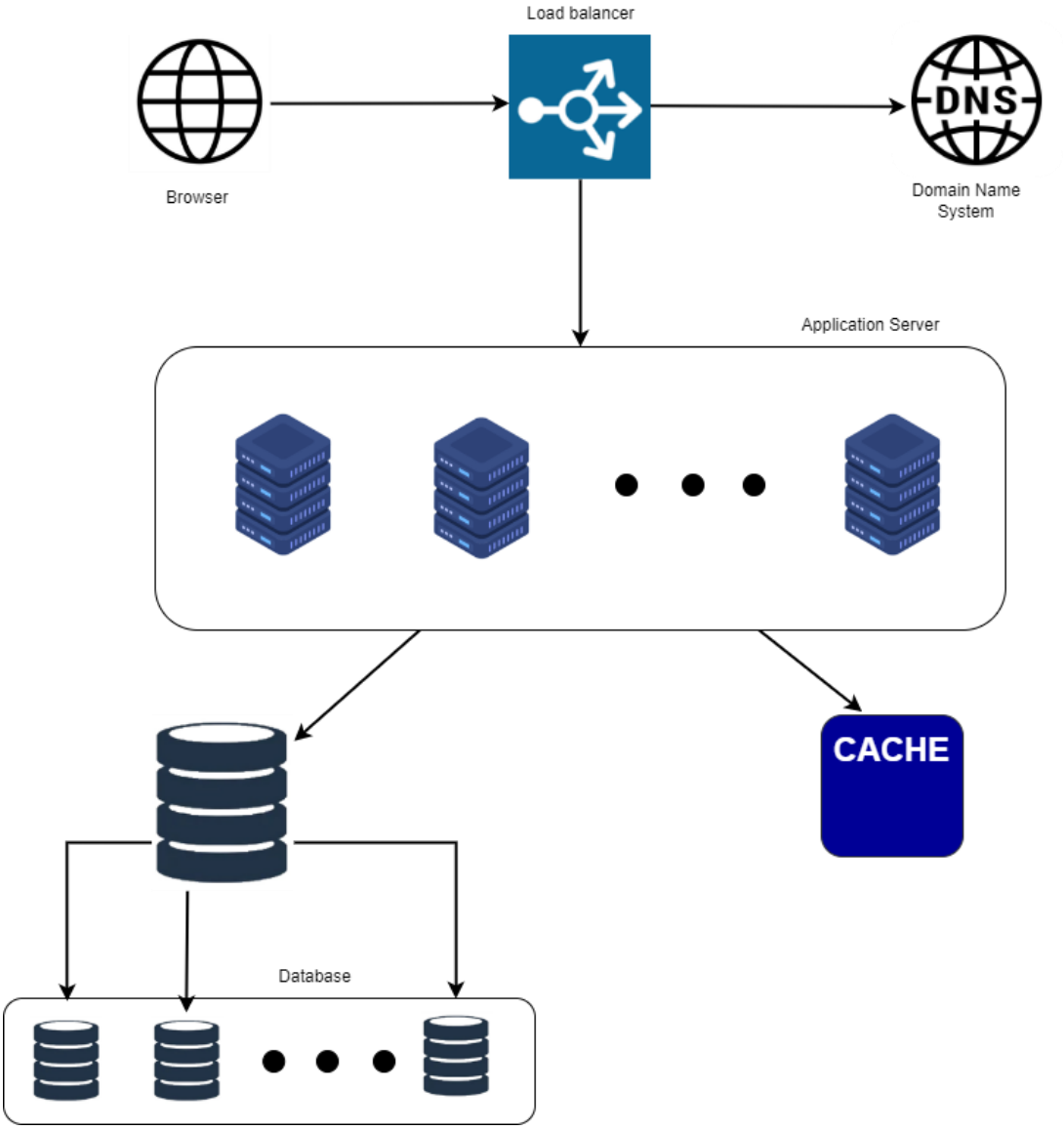
Na osnovu karakteristika, svakom serveru se dodeljuje težina, kako bi se iskazala njegova sposobnost rukovanja saobraćaja. Bitan je i broj konekcija servera. Smatramo da je ova tehnika *load balancing*-a dobra za našu aplikaciju, jer će voditi računa i o dostupnosti servera i o njegovim performansama.

7. Analiza korisničkih akcija

Na osnovu ponašanja i načina korišćenja našeg sistema od strane korisnika, možemo još više poboljšati gore navede strategije. Stvari poput:

- Podaci koji se najviše dobavljaju iz baze (najčešće pretražuju – vikendice, brodovi i avanture) se mogu smestiti na brže server, keširati radi bržeg pristupa
- Podaci kojima se retko pristupa (zahtev za brisanje naloga, uslovi otkaza, procenat zarade sisitema) od strane korisnika se takođe mogu smeštati na sporije servere
- Dodavanje dodatnih servera ili unapređenje postojećih servera na lokacijama gde je aktivnost korisnika naručito velika

8. Dizajn arhitekture



Slika 6 Dizajn predložene arhitekture