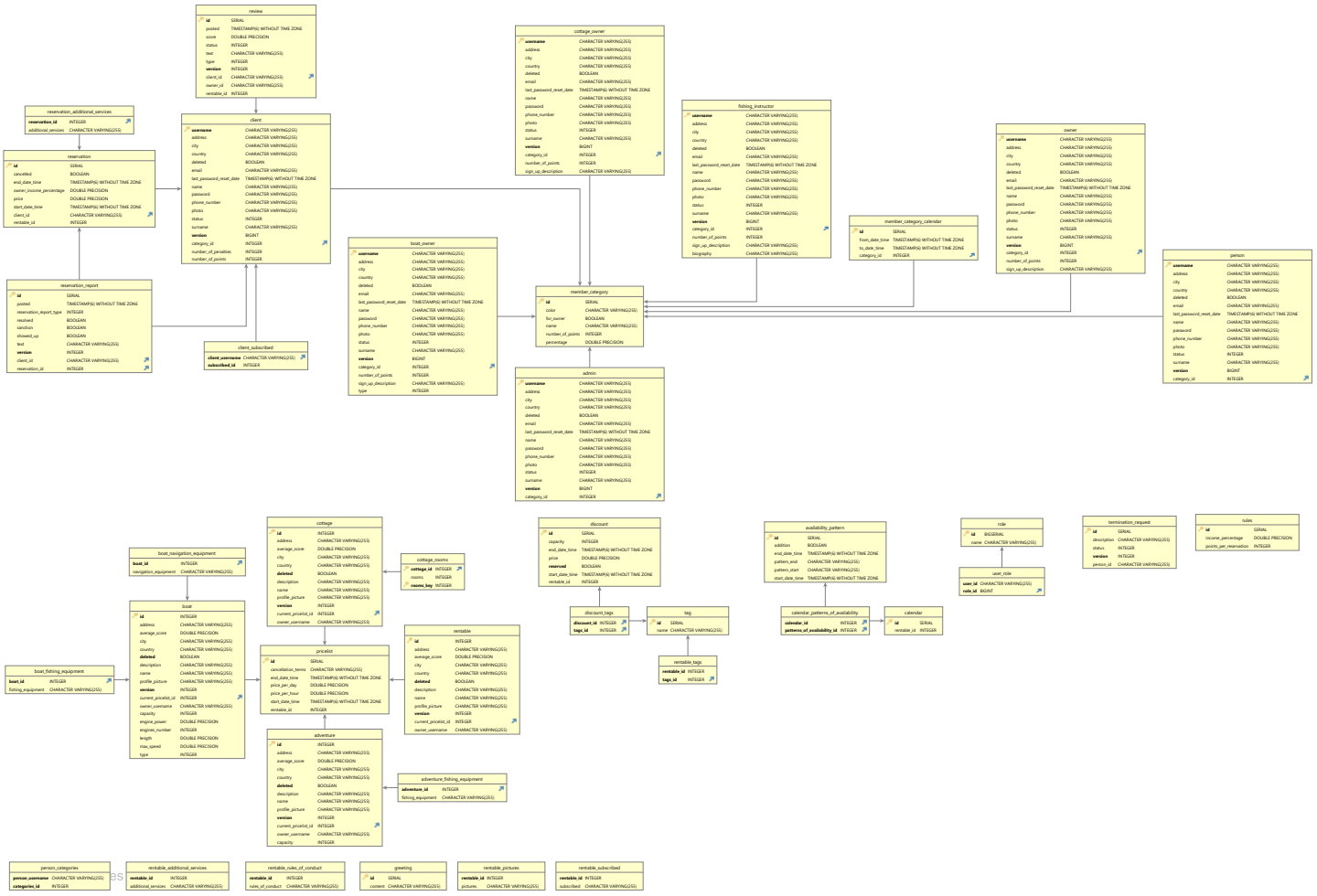
Skalabilnost Rivera aplikacije

ISA/MRS Tim 20

1. Dizajn šeme baze podataka



2. Predlog strategije za particionisanje podataka

Particionisanje podataka možemo odraditi i horizontalno i vertikalno. Predlog je da se podaci vezani za entitete particionišu horizontalno, na osnovu identifikatora. Često čitamo sve podatke vezane za jedan entitet pa je potrebno da se oni nalaze na istom mestu. Postoji velik broj entiteta u sistemu i horizontalno particionisanje ove tabele doprinelo bi brzini sistema.

Predlažemo vertikalno particionisanje podataka tabele korisnika. Najčešća operacija nad korisničkim tabelama je vezana za prijavljivanje na sistem. Korisničke kredencijale možemo izdvojiti u posebnu tabelu, što bi takođe doprinelo brzini sistema.

3. Predlog strategije za replikaciju baze i obezbeđivanje otpornosti na greške

Potrebno je obezbediti veću pouzdanost našeg sistema i to se ćemo postiči replikacijom baze podataka. Ukoliko jedna baza prestane da radi, potrebno je da obezbedimo da gubitak informacija bude što manji što ćemo obezbediti kreiranjem redudantnih baza podataka. Predlažemo upotrebu Master-Slave arhitekturu, gde imamo jednu Master bazu zaduženu za operacije pisanja, dok bismo imali više Slave baza. Slave baze bismo koristili za čitanje podataka. Rivera aplikacija očekuje daleko veći broj čitanja od pisanja, zato smatramo da je ova arhitektura pogodna.

4. Predlog strategije za keširanje podataka

Aplikacija koristi Hibernate i imamo podržano L1 keširanje. Za L2 keširanje predlažemo korišćenje servisa EhCache. Korišćenjem ovog servisa želimo minimizovati upite poslate u bazu kako bismo ubrzali rad aplikacije.

Aplikacija za rezervisanje vikendica, brodova i avantura sadrži ogroman broj statičkih podataka. Svaki od navedenih entiteta može da sadrži do 10 slika koje mogu biti visoke rezolucije što je veliko opterećenje za sistem. Takođe bismo mogli keširati podatke poput lokacije entiteta. Predlog jeste da ove podatke keširamo unapred, pri pokretanju aplikacije. Cilj jeste da klijenti mogu što pre da vide podatke o entitetu kako bismo postigli bolji UX.

Takođe možemo koristiti prediktivno keširanje. Nakon što klijenti izvrše akciju nad nekim entitetom, poput rezervacije, velika je verovatnoća da će ponovo otići na pregled datog entiteta. Zbog toga možemo unapred keširati dati entitet radi bržeg pristupa.

Potrebno je implementirati i mehanizam osveživanja podataka i predlažemo strategiju Least Recently Used.

5. Okvirna procena za hardverske resurse potrebne za skladištenje svih podataka u narednih 5 godina

Procena hardverskih resursa vršena je na osnovu trenutne upotrebe hardverskih resursa. Napomena: Zbog jednostavnosti računa, uzeto je da je 1000kB = 1MB. Ovim ćemo preceniti procenu, što svakako i moramo uraditi.

Entitet	Trenutno zauzeće	Projektovano	Objašnjenje
Klijent	3kB	240GB	90% korisnika su klijenti
Entitet	10kB	300GB	Svaki vlasnik u proseku 3 entiteta
Vlasnik	4kB	40GB	10% korisnika su vlasnici
Rezervacija	1kB	9TB	Mesečno u proseku entitet ima 5 rezervacija, što je 300 rezervacija za 5 godina.
Ocena/Žalba	2kB	9TB	Za svaku drugu rezervaciju postoji žalba/ocena
Slike klijenata/vlasnika	3MB	300TB	Svaka osoba ima profilnu sliku
Slike entiteta	5MB	715TB	Svaki entitet ima 5 slika u proseku

Procene same baze podataka, bez slika, jesu da bi nam bilo potrebno oko 20TB skladišta da bismo sačuvali podatke. Slike su najveći problem ovog sistema, sabrano bi zauzimale preko 1000TB, ili 1PB.

6. Predlog strategije za postavljanje load balansera

Zbog izuzetno velikog broja korisnika, biće nam potreban veći broj servera koji će opsluživati korisnike i horizontalno bismo skalirali servere. Kako bismo ravnomerno rasporedili komputacione zahteve, predlažemo upotrebu load balansera. Uzevši da je naša aplikacija stateless, naši web serveri su potpuno ravnopravni i možemo izabrati bilo koji od njih. Predlažemo da se koristi algoritam baziran na resursima web servera (Resource Based), zahtev se prosleđuje serveru koji trenutno troši najmanje resursa.

7. Predlog koje operacije korisnika treba nadgledati u cilju poboljšanja sistema

U cilju ubrzavanja rada aplikacije, moramo nadgledati specifične operacije kako bismo zaključili koje se to operacije najčešće koriste kao i koje operacije su usko grlo programa.

Prva predložena operacija jeste pretraga. Pretraga se može vršiti po imenu entiteta, tagova, datuma dostupnosti, imenu vlasnika. Takav tip pretrage je izrazito skupa operacija koja bi se možda i najšeće koristila u sistemu pa je neophodno nadgledati njene performanse.

Sledeća operacija jesu rezervacije, obične i akcije. Zbog implementacije zaključavanja baze podataka, ove operacije su po svojoj prirodi skuplje nego ostale. One bi se često dešavale, glavna namena sistema jesu upravo rezervacije. Zbog toga neophodno je analizirati njihove performanse.

Predlog još jedne operacije za nadgledanje jeste pregled entiteta. Entiteti u sebi sadrže najviše podataka u celom sistemu i to je operacija koja bi se dešavala pri svakoj rezervaciji ali i van njih. Slike su jedan od većih problema ove akcije, sadrže veliku količinu memorije i opterećuju komunikaciju.

8. Kompletan crtež dizajna predložene arhitekture