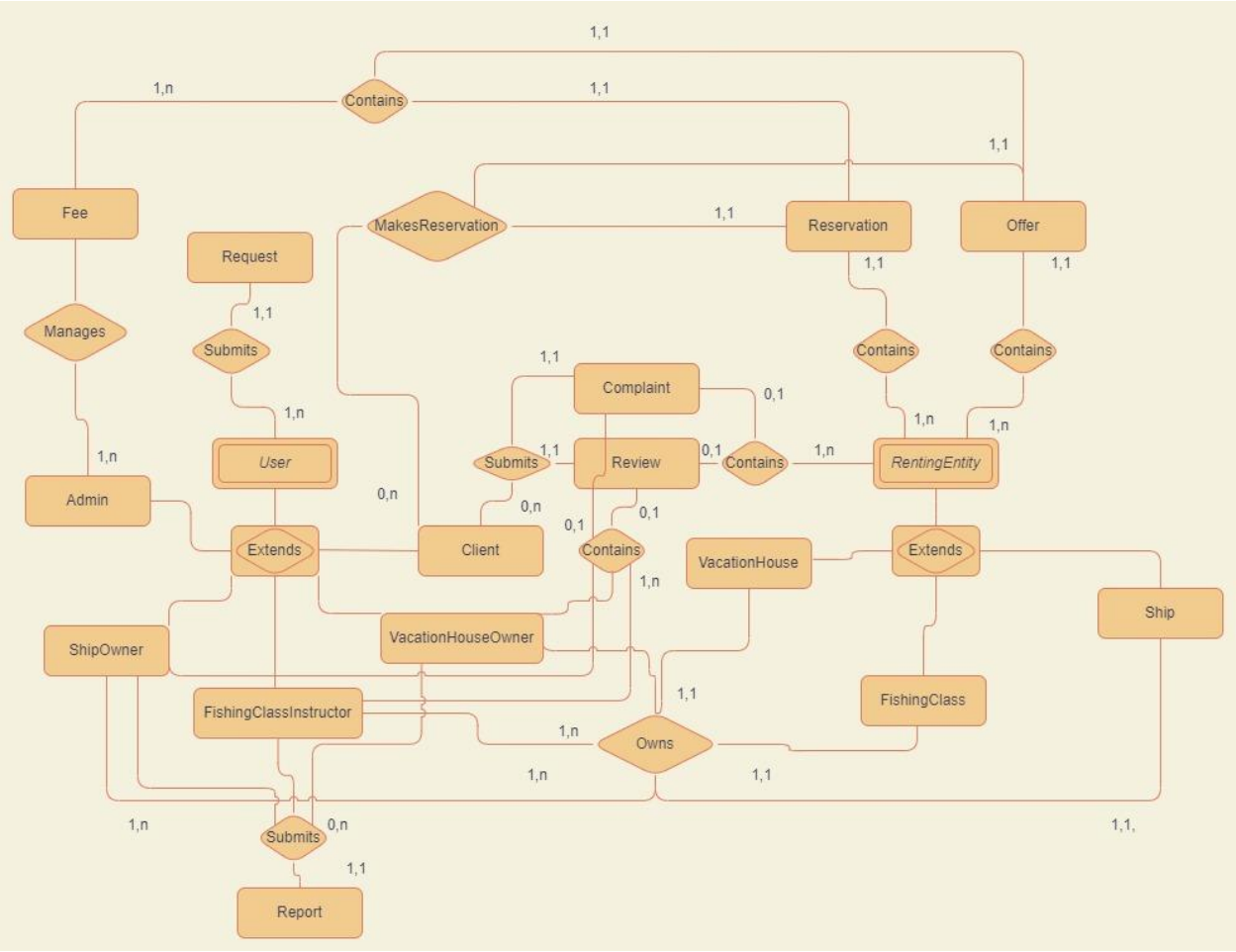
Proof of Concept za visoku skalabilnost aplikacije „Renting Buddy“ predmetnog projekta

Dragan Mirković, SW41/2019
Andrej Čuljak, SW50/2019
Dimitrije Petrov, SW55/2019
Softversko inženjerstvo i informacione tehnologije
Internet softverska arhitektura
Metodologije razvoja softvera
Novi Sad, jun 2022. godina

1. Dizajn šeme baze podataka



2. Predlog strategije za particionisanje podataka

Particionisanje podataka dovodi do smanjenja broja zahteva za pristup resursima i samim tim dovodi do poboljšanja performansi. Glavna akcija aplikacije je rezervisanje entiteta i samim tim najbolje rezultate bi dobili particionisanjem tabela za iznajmljujuće entitete (vikendice, brodove, avanture), rezervacije i akcije. Baza korišćena pri implementaciji projektnog rešenja je „*PostgreSQL*“ koja pruža tri tipa deklarativnog particionisanja: „*Range“* (na osnovu broja ili datuma), „*List“* (grupe koje mi definišemo na osnovu vrijednosti nekog atributa), i „*Hash*“.

Tabele koje se odnose na korisnike sistema bi bilo pogodno horizontalno *hash* particionisati po atributu „*username“*, jer pod pretpostavkom da aplikacija ima

100 miliona korisnika, najviše zahteva bi bilo dobijanjem korisnika na osnovu JSON web tokena i obratno.

Tabele koje se odnose na iznajmljujuće entitete bi bilo pogodno horizontalno *hash* particionisati po vlasniku datog entiteta.

Tabele rezervacija i akcija određenog iznajljujućeg entiteta bi bilo dobro horizontalno *hash* particionisati po entitetu za koji se odnose.

Tabele za žalbe i ocene u sebi sadrže polje za vlasnika entiteta i polje za sami entitet pa bi i bilo pogodno po tome i izvršiti particionisanje, takođe *hash* horizontalno.

Takođe efikasno bi bilo izvršiti i vertikalno particionisanje na delove tabela za koje korisnik ređe traži upite. Time bi došlo do razdvajanja između bitnih i manje bitnih redova i do povećanja performansi pristupa.

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

Glavna uloga naše aplikacije predstavlja rezervisanje entiteta. Pod pretpostavkom da naša aplikacija ima 100 miliona korisnika, možemo očekivati veliki broj *„write“* naredbi. Iz tog razloga predlažemo implementaciju „*multiple master*“ arhitekture, gde bi svaki „*database*“ server obrađivao „*insert“* i „*update*” upite i prosleđivao ih ostalim serverima radi očuvanja konzistentnog stanja. Ovo bi zahtevalo postojanje više od dva „*database*“ servera, iz razloga što ako dođe do otkazivanja mreže ne dođe do skladištenja različitih podataka.

Jedini mogući scenario koji bi doveo replikacije do invalidnog, nekonzistentnog stanja, jeste otkazivanje „*database*“ servera i otkazivanje mreže između druga dva servera. Naša procena glasi da je jako mala verovatnoća za pojavljivanjem ovakve situacije.

U slučaju da se povećanja „*read“* naredbi mogu se dodavati dodatni „*database slave*” server koji bi za posao imali samo čitanje podataka.

Dodatni vid povećanja otpornosti na greške smo dobili uvođenjem transakcija i optimističkog zaključavanja za sve konflikte situacije prilikom rada aplikacije.

4. Predlog strategija za keširanje podataka

Keširanje podataka potencijalno dovodi do ubrzanja pristupa podacima koji se često čitaju, a retko menjaju, kao na primer: lični podaci o klijentu, lični podaci o vlasniku i informacioni podaci o iznajmljujućem entitetu. Za ostale podatke kao što su rezervacije i akcije određenog entiteta koji se često menjaju, pogodno bi bilo podržati *„least frequently used“* keš strategiju. To bi se moglo odraditi upotrebom *„open-source EhCache“* kešom koji rasterećuje bazu i povećava performanse. Uvodi se u većini slučajeva samo navođenjem odgovarajućih notacija.

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

Za potrebe naše aplikacije smo uzeli pretpostavke da će aplikacija imati 100 miliona korisnika, od kojih 80% klijenata a 20% vlasnika entiteta. Jedan korisnik u proseku zauzima 0.5KB što ukupno daje 46.57GB.

Pretpostavili smo da će svaki vlasnik imati jedan do dva entiteta na izdavanju, što nam daje ukupan broj od 35 miliona. Jedan entitet u proseku zauzima 0.5KB, što ukupno daje 16.3GB. Procenili smo da će svaki entitet godišnje imati tri akcije, što nas dovodi do brojke od 8.75 miliona akcija mesečno po 0.15KB potrebnog prostora za skladištenje, odnosno 73.34GB ukupno na pet godina.

Još jedna procena je ta da će svaki klijent godišnje napraviti bar 6 rezervacija, što nas dovodi do 40 miliona rezervacija na mesečnom nivou. Jedna rezervacija u proseku zauzima 0.15KB prostora u bazi, što nas dovodi do ukupnog zauzimajućeg prostora koji iznosi 335.28GB na pet godina.

Pretpostavili smo da će se ocena ili žalba podneti na pola rezervacija, u omjeru tri naprema jedan. Što nas dovodi do 15 miliona ocena i 5 miliona žalbi na mesečnom nivou. I ocena i žalba zauzimaju po 0.15KB što dovodi do ukupnog zauzeća od 125.73GB za ocene i 41.91GB za žalbe na pet godina.

Što se tiče zahteva, oni zauzimaju takođe približno 0.15KB, a naša je procena da će biti oko 100 hiljada zahteva mesečno, što dovodi do ukupnog zauzeća od 0.84GB na pet godina.

Poslednja procena je u vezi sa pretplatama. Procenili smo da će u proseku svaki klijent biti pretplaćen na 5 entiteta. Svaka pretplata zauzima 0.008KB, a ukupno će ih biti 400 miliona. Što dovodi do potrebne memoriju u količini od 2.98GB.

Kada ovo sve saberemo, naša procena je da će nam biti potrebno oko 643GB u narednih pet godina.

6. Predlog strategije za postavljanje load balansera

Upotrebom *„load“* balansera doveli bi do smanjenja opterećenja čvorova. Jednostavniji vid implementacije balansera bi bio *„Round Robin“*, koji bi kružno dodeljivao pristigle zahteve. Kompleksnije, ali bolje rešenje bi bio *„load“* balanser baziran na *„ IP“* adresi klijenta. Ako se geografska distribuiranost korisnika ispostavi povoljna za ovakvu implementaciju, došlo bi do povećanja performansi.

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

Najveći broj operacija koje će se izvršavati nad aplikacijom čine operacije rezervisanja entiteta i rezervisanja akcija entiteta. U skladu sa tim operacije koje bi bilo potrebno nadgledati su:

- Rezervisanje akcija od strane klijenta.

  Ako se utvrdi da se ne koristi toliko često i ne menja u odnosu na potpune rezervacije, moglo bi se izvršiti keširanje.

- Pregled istoriji rezervacija od strane klijenta.

  Isti razlog kao u prethodnoj operaciji.

- Pregled zauzetosti entiteta.

  Ova operacija predstavlja jednu od ključnih za uspešno poslovanje aplikacije, iz tog razloga potrebno je uvek dobaviti tačne rezultate o zauzetosti/dostupnosti određenog entiteta. Potrebno bi bilo ispratiti ponašanje ove operacije i da li postoji prostor za unapređenjem.

8. Kompletan crtež dizajna predložene arhitekture (aplikativni serveri, serveri baza, serveri za keširanje, itd)