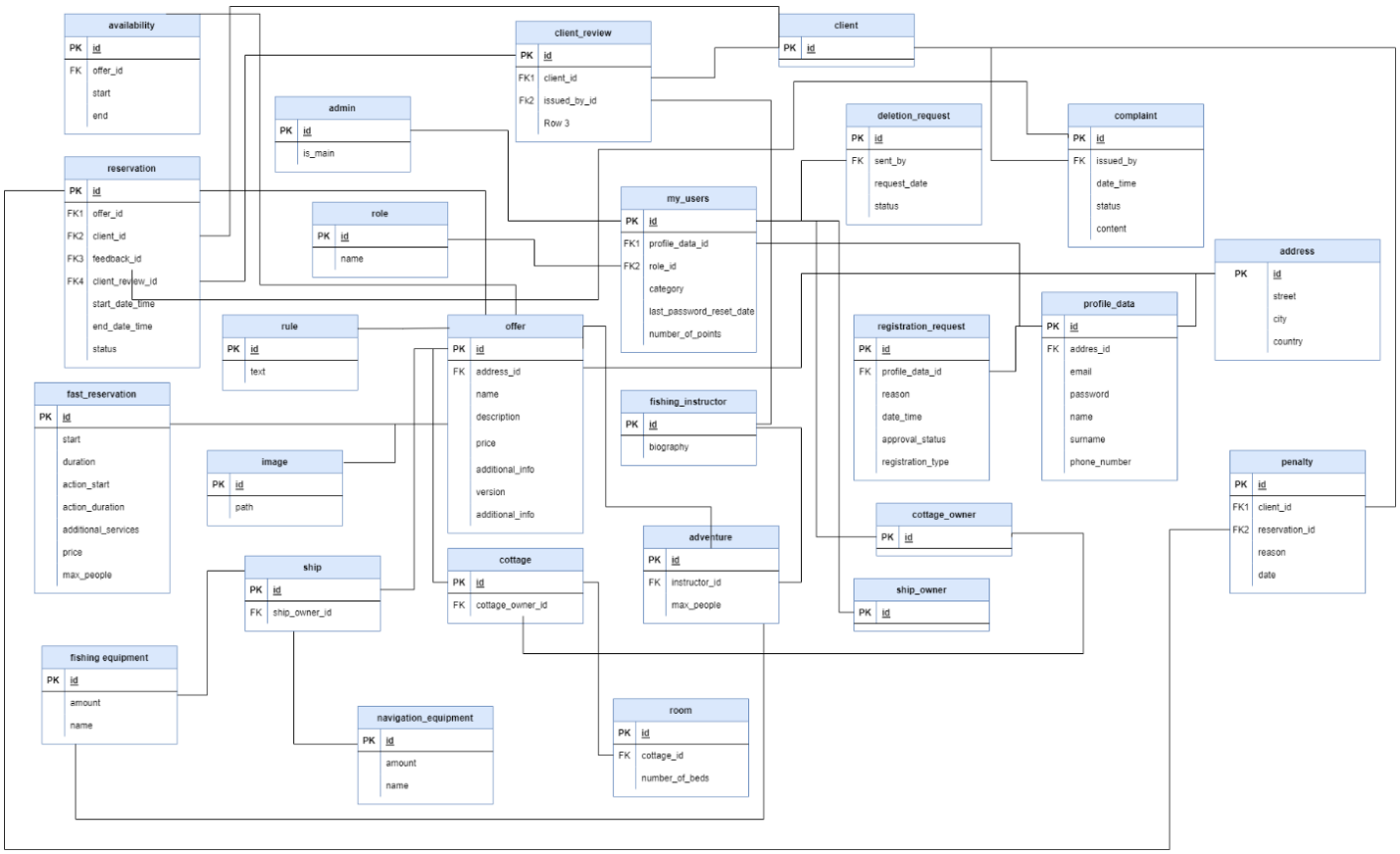
Skalabilnost

Dizajn šeme baze podataka



Strategija za particionisanje podataka

Predlažemo da se podaci o ponudama i rezervacijama particionišu horizontalno. Pošto u sistemu postoji veći broj ponuda i rezervacija, horizontalnim particionisanjem možemo podatke o njima rasporediti na više servera i time povećati efikasnost sistema. Na ovaj način se smanjuje vreme potrebno za izvršavanje upita.

Prilikom horizontalnog particionisanja značajno je voditi računa o tome da podaci budu ravnomerno raspoređeni u particije. Zbog toga preporučujemo da se

particionisanje vrši na osnovu ključa, odnosno da se heširanjem vrednosti ključa entiteta dobije broj particije u koju entitet treba da se upiše. Ukoliko se monitoringom ustanovi da se nekim podacima češće pristupa nego drugim, onda se ova strategija može promeniti tako da se dobiju veće particije sa ređe korišćenim podacima i manje particije sa češće korišćenim podacima.

Za čuvanje podataka o korisnicima predlažemo vertikalno particionisanje, jer na taj način možemo odvojiti kredencijale od ličnih podataka. Kredencijale ima smisla čuvati odvojeno od ličnih podataka jer su to osjetljivi podaci za koje je potrebno obezbediti veći nivo sigurnosti. Takođe, njima se često pristupa, te se na ovaj način povećava brzina sistema.

Replikacija baze podataka

Pošto je neophodno da sistem bude pouzdan i dostupan, podatke treba čuvati u više baza podataka. Mi predlažemo rešenje u kojem će postojati više primarnih i više sekundarnih baza podataka.

Primarne baze primaju podatke od servera i upisuju ih, dok sekudarne baze dobijaju kopije tih podataka. Pri tome se u okviru primarnih baza podataka obavlja samo operacija pisanja, a u okviru sekundarnih samo operacija čitanja, što dovodi do poboljšanja performansi.

Prednost upotrebe više primarnih baza podataka u odnosu na to da postoji samo jedna, je to što se podaci u sekundarne baze upisuju iz najbliže primarne baze. Ovim se smanjuje kašnjenje i povećava se pouzdanost sistema, što je posebno izraženo u slušaju geografski distribuiranih baza podataka. Takođe, ukoliko dođe do otkaza neke od primarnih baza, ostale mogu preuzeti njenu ulogu.

Keširanje podataka

Potrebno je obezbediti L2 nivo keširanja, jer je L1 nivo automatski podržan od strane Hibernate-a. Za podržavanje i L2 nivoa, odlučili smo se za eksterni provajder *Ehcache*, jer on podržava sve strategije keširanja .

Pošto su ponude objekti kojima se često pristupa, keširanjem njihovih podataka bi se povećala brzina odgovora i smanjilo bi se opterećenje na bazu podataka. Keširanje ponuda se radi na zahtev. Pri tome pretpostavljamo da se osnovni podaci o ponudi (naziv, opisi, adresa, slike...) ne menjaju često.

Kao strategiju za invalidaciju keša predlažemo *time-to-live* (TTL) strategiju. Ovaj parametar se može podešavati prema tome koliko retko očekujemo da će se objekat menjati. On garantuje da se podaci ne zadrže predugo u kešu. Takođe predlažemo da se primeni i *time-to-idle* strategija, da se ne bi u kešu zadržavali podaci kojima se retko pristupa. Kao strategiju za brisanje podataka iz keša predlažemo *Least Recently Used* strategiju.

Hardverski resursi

Odradićemo procenu količine memorije koja je neophodna za smeštanje svih entiteta u vremenskom periodu od 5 godina. To ćemo uraditi tako što ćemo pomnožiti trenutnu veličinu objekata sa očekivanim brojem objekata nakon 5 godina.

- Korisnici

  U proseku, jedan korisnik zauzima 5kB memorije. Pretpostavljamo da će sistem imati 100 miliona korisnika. Stoga je za čuvanje podataka o korisnicima potrebno obezbediti oko 500GB memorije.

- Ponude

  Jedna ponuda u proseku zauzima oko 7MB memorije. Ako pretpostavimo da vlasnici čine 15% korisnika i da svaki vlasnik ima 5 ponuda, dobijamo da se za skladištenje podataka o ponudama potrebno 525TB memorije.

- Rezervacije

  Posmatraćemo veličinu rezervacije koja sadrži ocene (klijenta i vlasnika), jer one zauzimaju najviše memorije. Jedna takva rezervacija u proseku zauzima 5kB memorije. Pošto je broj rezervacija na mesečnom nivou 1 milion, to znači da je za 5 godina ostvareno 60 miliona rezervacija. Zbog toga je za skladištenje rezervacija potrebno 300GB memorije.

Sumiranjem ovih procena, dobijamo da je za skladištenje celokupnog sistema potrebno oko 530TB memorije.

Load balancer

Pošto je neophodno da aplikacija opsluži veliki broj korisnika, opterećenje je neophodno raspodeliti na više servera. Zbog toga je neophodno da postoji *load*

balancer koji će na odgovarajući način raspodeliti zahteve koji pristižu od klijenata na servere.

Za *load balancing* tehniku predlažemo *Weighted Response Time*. U okviru ove tehnike se za server kome će se proslediti idući zahtev uzima onaj server koji ima najkraće vreme odgovora. Iako ova tehnika uglavnom ne raspoređuje zahteve ravnomerno, kao npr. *Least Connection* tehnika, njena prednost je u tome što zahteva malo memorije i malu procesorsku moć.

Nadgledanje korisnika

Nadgledanjem aktivnosti korisnika tokom upotrebe aplikacije stičemo uvid u to koje akcije korisnici najčešće izvršavaju. To znanje nam je korisno jer će optimizacija najčešće izvršavanih akcija najviše doprineti optimizaciji aplikacije. Mi za nadgledanje predlažemo neke od operacija koje su najskuplje i za koje pretpostavljamo da će biti najčešće korišćene.

Jedna od najčešćih operacija je rezervacija ponuda, jer je to primarni cilj klijenata aplikacije. Takođe, kreiranje rezervacije je skupa operacija, jer je tokom nje može doći do konflikata, te je neohodno zaključati bazu podataka. Zbog toga smatramo da je ova operacija najznačajnija za nadgledanje. Isto važi i za rezervisanje ponude preko akcije za brzu rezervaciju.

Još jedna od ključnih operacija je pregledanje svih ponuda, što je takođe često izvršavana operacija. Ponuda u sistemu ima mnogo i one mogu da sadrže veću količinu podataka, tako da je njihovo dobavljanje skupa operacije, te je treba optimizovati radi povećanja efikasnosti sistema.

Predložena arhitektura