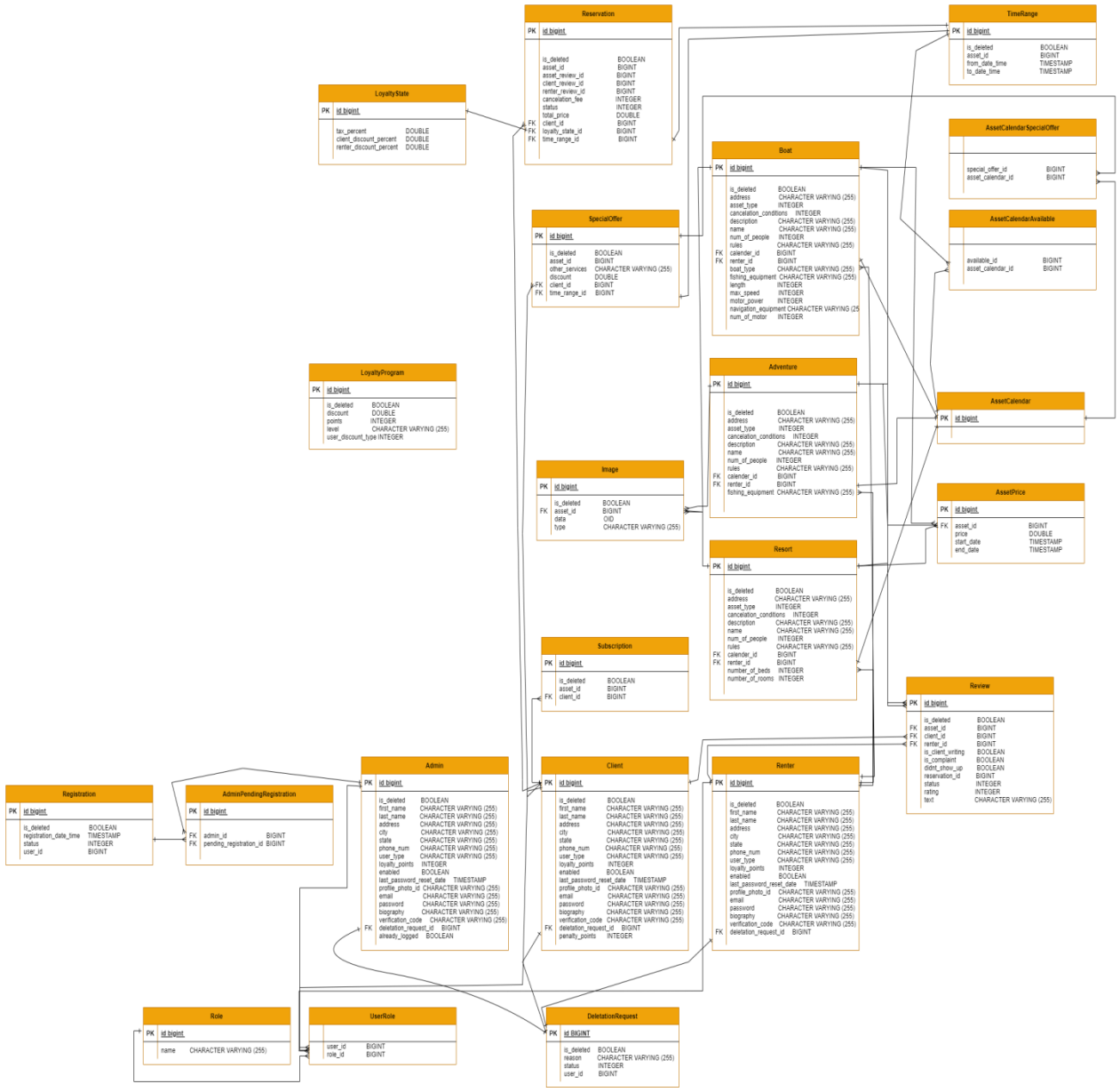
Skalabilnost – Proof Of Concept

1. Dizajn šeme baze podata



2. Strategija za particionisanje podataka

Kako je neophodno da aplikacija podrži 100miliona korisnika I prosečno milion rezervacija na mesečnom nivou, potrebno je izvršiti neku vrstu particionisanja podataka, kako bi performanse I dostupnost ostali na visokom nivou.

Analizom šeme baze podataka I funkcionalnih zahteva, zaključeno je da je najveći broj operacija upućenih ka bazi čitanje podataka; I to pretežno iz tabela *User* (*Client*, *Boat Renter*, *Resort Renter, Fishing instructor*), tabele *Asset* (*Boat, Resort, Adventure*) I tabele *Reservation*.

Stoga, dobro rešenje za našu aplikaciju bi bilo horizontalno particionisanje tabele rezervacije. Pretpostavka je da će svaki korisnik imati veliki broj svojih rezervacija, te bi se u posebnoj particiji čuvale rezervacije jednog korisnika.

3. Strategija za replikaciju baze i obezbeđivanje otpornosti na greške

Da bi se obezbedila još veća optimizacija pri čitanju I pisanju podataka I kako bi se sprečio neželjeni gubitak podaka, smatramo da je Master-Slave arhitektura odlično rešenje.

Primarna(*master*) baza bi podržavala samo *write* operacije, dok bi sve sekundarne(slave) baze podataka predstavljale repliku master baze i podržavale bi samo read operacije. Kako smo već spomenuli da je broj čitanja mnogo veći od broja pisanja, slave instance bi bilo značajno više.

Sekundarne baze registrovaće promene kada se dese u master bazi i tada će biti ažurirane, čime se neće narušiti konzistentnost. Takođe, na ovaj način povećavamo otpornost sistema na greški ali i lakši oporavak ako do grešaka I dođe, jer imamo duplirane necentralizovane podatke.

4. Strategija za keširanje podataka

Trenutno je u okviru naše aplikacije implementirano keširanje ponuda(asset-a) prilikom dobavljanja po id. Smatramo da je ovo bilo najpogodnije mesto za demonstraciju upotrebe keširanja jer je jedna od najčešće upotrebljivanih operacija.

Za dalje unapređivanje treba uzeti u obzir prediktivno keširanje odnosno keširanje nakon kreiranja novog entiteta kako bismo ubrzali vreme odziva

Kako bismo u kešu imali što svežije podatke smatramo da je LRU strategiju (*Least Recently Used*) sasvim korektna s obzirom da se najviše pristupa najpopularnijim podacima.

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

Okvirne procene po entitetima

- Korisnici
  - 7 String polja = 7*(20+10/5*4) = 280B
  - 6 polja osnovnog tipa = oko 100B
  - Oko 380B po korisniku * 100miliona korisnika = 380GB
- Dostupne ponude
  - Oko 15% izdavača u odnosu na svih 100miliona korisnika = 15miliona izdavača
  - Oko 2.5 ponude po izdavaču * 15miliona izdavača = 37.5miliona ponuda
  - Oko 600B po usluzi * 37.5miliona ponuda = 22.5GB
- Rezervacije
  - Milion rezervacija mesečno * 12meseci * 5 godina = 60miliona rezevacija
  - Oko 200B po rezervaciji * 60miliona rezervacija = 12GB
- Ocene
  - Kako je moguće 3vrste ocena (za ponudu, izdavača i klijenta) isključivo za završene rezervacija(oko 65%), ali se očekuje da će korisnici popuniti tek oko 40% svih mogućih ocena
  - 65% završenih rezervacija od 60miliona svih = 39miliona završenih rezervacija
  - 3vrste ocena * 39miliona završenih rezervacija = 117miliona mogućih ocena
  - Oko 40% popunjenih ocena * 117 = 47miliona ocena
  - 10B po oceni * 47miliona ocena = Oko 5GB
- Svi ostali entiteti ukupno
  - Oko 30GB
- Dodatno su i slike
  - Oko 1 profilna slika svakog korisnika * 100miliona korisnika = 100miliona slika
  - Oko 4.5 slike za svaku ponudu * 37.5miliona ponuda = 168.75miliona slika
  - Oko 1MB za sliku * (100miliona + 168miliona) = 268TB

Ukupno: Oko 450GB za entitete + 268TB za slike

6. Strategija za postavljanje load balansera

Zbog obima zahteva koji će se upućivati ka serveru, smatramo da je definitivno neophodno uvesti horizontalno skaliranje a samim tim i *Load Balanser*. Strategija za koju bismo se odlučili bila bi *Least Connections*, jer predstavlja balans između karakteristika servera i broja aktivnih konekcija. Smatramo da je ovo dobar izbor jer je neophodnovoditi računa i o dostupnosti server i o njegovim performansama, kako bi klijenti što pre dobijali odgovo.

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

Neki od osnovnih operacija koje je potrebno nadgledati nakon puštanja u produkciju kako bi mogli da unapredimo naš sistem:

- Čitanje i pisanja u bazu podataka → Ukoliko je broj čitanja veći nego što je pretpostavljeno, možemo poboljšati performanse time što ćemo dodati jos sekundarnih (*slave*) baza podataka.

- Keširanje podataka → Ukoliko vidimo da nema efekta keširanja podataka koji nisu traženi ili uočimo da su često traženi oni podaci koji nisu u kešu, možemo izmeniti inicijalno odabranu strategiju keširanja.

- Atributi entiteta koji se najviše dobavljaju iz baze → Mogli bismo uvesti i vertikalno particionisanje podataka, kako bi omogućili brži rad sa frekventno korišćenim podacima.

8. Crtež dizajna predložene arhitekture