Servisno orijentisane arhitekture

Predavanje 9: Mikroservisi i paterni, Obrasci za upravljanje podacima, CQRS, Domain Event, Event Sourcing, Obrasci za transakcionu razmenu poruka

Univerzitet u Novom Sadu
Fakultet Tehničkih Nauka

Obrasci za upravljanje podacima

- Jedna baza po servisu
- Deljene baze
- Saga
- Kompozicija API-ja
- CQRS (Command Query Responsibility Segregation)
- Domain Event (domenski dogadjaji)
- Event Sourcing (izvori dogadjaja)

CQRS - Command Query Responsibility Segregation

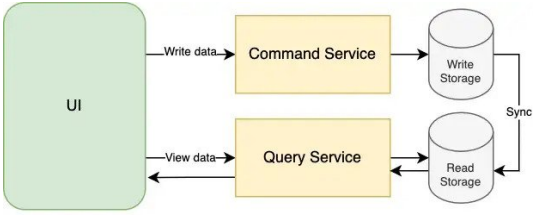
- Implementirana je mirkoservisna arhitektura, za čuvanje podataka obrazac jedne baze po servisu.
- Kao rezultat tog pristupa nije više jednostavno prikupiti podatke koji pripadaju različitim servisima.
- Kada je implementiran i Event Sourcing Pattern to čini upite nad podacima komplikovanim.

Problem

- Kako implementirati upite koji povlače podatke sa različitih mikroservisa?

Rešenje

- Definisati bazu podataka samo za čitanje, koja predstavlja repliku podataka potrebnih da se podrži odredjeni upit.
- Aplikacija obezbedjuje da ova replika bude uvek ažurna putem pretplate na odgovarajuće dogadjaje koje emituje onaj mikroservis koji je vlasnik podataka.



(CQRS Software Architecture Pattern: The Good, the Bad, and the Ugly)

Osobine

- Dobre osobine:
  - Obezbedjuje podršku za više denormalizovanih upita pri čemu se obezbedjuju dobre performanse i skalabilnost sistema.
  - Pojačava se princip podele nadležnosti - što rezultuje jednostavnijim komandama i upitima.
  - Neophodan je u arhitekturama koje su zasnovane na event sourcing-u.
- Loše osobine
  - Povećava se kompleksnost sistema
  - Moguće je dupliranje koda u različitim servisima.
  - Kašnjenje pri replikaciji podataka.

Domain Event

- Servis često treba da objavi dogadjaj da je ažurirao svoje podatke
- Primeri za to su u prethodnom CQRS obrascu, kao i u koreografski zasnovanoj SAGA-i.

Problem

- Kako mikroservis objavljuje dogadjaj nakon što je ažurirao svoje podatke?

Rešenje

- Organizovati poslovnu logiku servisa kao kolekciju DDD agregata (aggregates - graf/hijerarhija objekata koja se može posmatrati kao jedinstvena celina).
- Oni emituju domenske dogadjaje svaki put kada se kreiraju ili ažuriraju
- Servis objavljuje ove dogadjaje tako da ih mogu obraditi drugi servisni moduli.

Event Sourcing

- Servisna komanda treba da ažurira bazu podataka i da emituje poruku/dogadjaj.
- Na primer servis koji učestvuje u SAGA-i obrascu treba atomično da ažurira jednu bazu i nakon toga emituje poruku/dogadjaj o tome.
- Servis koji emituje domenski dogadjaj treba da atomično ažurira agregirani objekat i o tome emituje dogadjaj.
- Obe ove operacije - ažuriranje baze i samo slanje poruke/dogadjaja - moraju se obaviti nedeljivo, kako bi se izbegla nekonzistencija.
- Istovremeno, nije moguće koristiti distribuirane transakcije preko više servisa i koristiti message broker da se objave ovakve izmene.

Problem i fatkori

- Kako pouzdano/nedeljivo obaviti ažuriranje baze i objavu tog dogadjaja?
- Nije moguće koristiti 2PC

Rešenje

- Event sourcing podatke o stanju poslovnog entiteta čuva u stanju sekvence dogadjaja koji menjaju stanje (serija promena koja poslovni objekat dovode u tekuće stanje).
- Kad god se stanje objekta promeni, novi dogadjaj, koji opisuje tu promenu se doda na listu dogadjaja.
- Kako je snimanje jednog dogadjaja atomična samo jedna operacija, ona je inherentno atomična.
- Aplikacija rekonstruiše stanje objekata tako što izvršava sve dogadjaje iz liste.
- Aplikacija pamti ove dogadjaje/promene stanja u bazi izmena.

- Ova baza ima API za dodavanje ili čitanje dogadjaja za neki objekat.
- Ova baza se koristi i kao message broker jer obezbedjuje API kojim se servisi mogu pretplatiti da slušaju izmene nad odredjenim objektima.
- Neki poslovni entiteti mogu da imaju veliki broj registrovanih dogadjaja - izmena.
- Kako bi se optimizovalo opterećenje aplikacija može periodično da napravi zamrznuti snimak (snapshot) stanja objekta.
- U tom slučaju se za rekonstukciju stanja objekta koristi najnoviji snapshot i lista izmena nakon njega.

Osobine

- Dobre osobine:
  - Rešava ključne problem pri implementaciji event-driven arhitektura i čini pouzdanim objavljivanje dogadjaja kad dodje do promene stanja.
  - Kako se čuvaju izmene, a ne sama stanja objekata, nema ni uobičajenih problema koji se mogu javiti pri ORM.
  - Obezbedjuje 100% pouzdane provere izmena koje su nastale.
  - Omogućava izvršavanje vremenskih upita kojima se može utvrditi stanje objekata u bilo kom momentu u vremenu.
  - Poslovna logika bazirana na Event sourcingu sastoji se od slabo medjusobno zavisnih poslovnih objekata koji razmenjuju samo dogadjaje informacije o promenama svog stanja.
  - Ovo omogućava lakše migracije na novu arhitekturu.

- Loše osobine
  - Programiranje sistema koji koriste ovakav pristup zahteva dodatno navikavanje i dosta je teško za učenje na početku.
  - Upiti nad bazom dogadjaja su dosta komplikovani, jer su neophodni precizni upiti kako bi se izvuklo stanje objekata u nekom momentu u vremenu.
  - Upiti su stoga često kompleksni i poprilično neefikasni.
  - Kao rezultat ovih nedostataka, aplikacija skoro po pravilu koristi i CQRS kako bi obezbedila jednostavniji način za izvršavanje složenih upita.
  - Što opet zahteva održavanje ažurnih replika.

- Transakcioni outbox (aplikativni dogadjaji)
- Praćenje kraja transakcionih logova (Transaction logs tailing)
- Pooling publisher

Transakcioni outbox

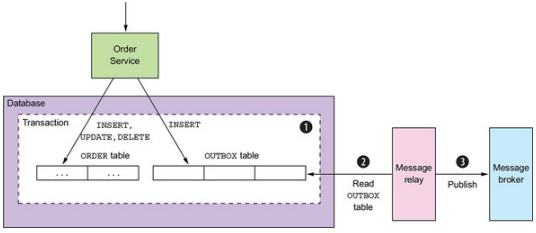
- Isti kao kod event sourcinga.
- Servisna komanda treba da ažurira bazu podataka i da emituje poruku/dogadjaj.
- Na primer servis koji učestvuje u SAGA-i obrascu treba atomično da ažurira jednu bazu i nakon toga emituje poruku/dopgadjaj o tome.
- Servis koji emituje domenski dogadjaj treba da atomično ažurira agregirani objekat i o tome emituje dogadjaj.
- Obe ove operacije - ažuriranje baze i samo slanje poruke/dogadjaja - moraju se obaviti nedeljivo, kako bi se izbegla nekonzistencija.
- Istovremeno, nije moguće koristiti distriburiane transakcije preko više servisa i koristiti message broker da se objave ovakve izmene.

Problem i faktor

- Kako pouzdano/nedeljivo obaviti ažuriranje baze i objavu tog dogadjaja?
- Nije moguće koristiti 2PC

Rešenje

- Servis koji koristi relacionu bazu poruke dodaje u outbox tabelu, kao sastavni deo lokalne transakcije.
- Servis koji koristi NoSQL bazu podataka dodaje poruke/dogadjaje kao atribute zapisa.
- Poseban, proces za slanje poruka Message Relay objavljuje ove poruke message brokeru.



(https://pradeepl.com/blog/transactional-outbox-pattern/)

Osobine

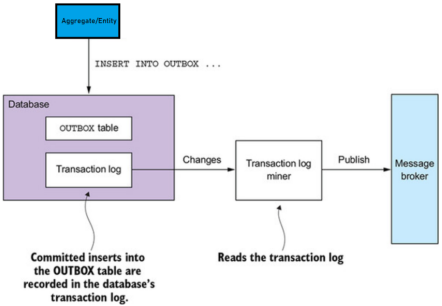
- Dobre osobine:
  - Servis objavljuje domenske dogadjaje sikogo nivoa.
  - Nije neophodan 2PC protokol.
- Loše osobine
  - Potencijalno sklon greškama jer je moguće da se pri razvoju zaboravi odraditi objavljivanje odredjenih poruka posle ažuriranja baza
- Dodatni problemi
  - Message relay može objaviti neku poruku više od jedan put.
  - Stoga ona komponenta koja prima poruke mora biti dizajnirana kao idempotentna, ali srećom većina sistema za konzumaciju poruka je takva.

Transaction logs tailing

- Implementiran je transaction outbox pattern
- Kako poruku koja je u outboxu emitovati/objaviti?

Rešenje

- Konstantno pratiti promene na kraju transakcionog log fajla i objaviti message brokeru svaku poruku insertovanu u outbox (tabelu).



(https://pradeepl.com/blog/transactional-outbox-pattern/)

Osobine

- Dobre osobine:
  - Nije neophodan 2PC protokol.
  - Garantovano je tačno detektovanje izmena
- Loše osobine
  - Poprilično slabo prihvaćeno rešenje, mada postaje sve poznatije i šire korišćeno.
  - Zahteva rešenja koja su specifična za pojedine baze.
  - Zahteva pažnju da se izbegnu višestruka objavljivanja.

Pooling publisher

- Implementiran je transaction outbox pattern
- Kako poruku koja je u outboxu emitovati/objaviti?

Rešenje

- Poruke objavljivati tako što se periodično čita (polling) iz outbox tabela.

Osobine

- Dobre osobine
  - Jednostavno rešenje, primenljivo na sve relacione baze.
- Loše osobine
  - Treba posebna pažnja da se obezbedi da se poruke preuzimaju i objavljuju po redosledu.
  - NoSQL baze možda ne podržavaju na jednostavan način ovako nešto.

Dodatni materijali

- Building Microservices, Sam Newman
- Microservices • Martin Fowler • GOTO 2014
- What are microservices?
- Microservices patterns

Kraj predavanja

Pitanja? :)