Tamara Govorčin RA 63/2017

1. Na jednu žalbu može da odgovori samo jedan admnistrator sistema.

- Opis situacije

  Amidnistrator sistema pregleda sve žalbe koji su pacijenti napisali. Unošenjem odgovora na žalbu I klikom na dugme za slanje, šalje se zahtev ka serveru za update ove žalbe. Server zatim pronađe žalbu na koju je administrator sistema odgovarao, promeni answered Boolean vrednost na true I čuva u bazu.
- U čemu je problem?

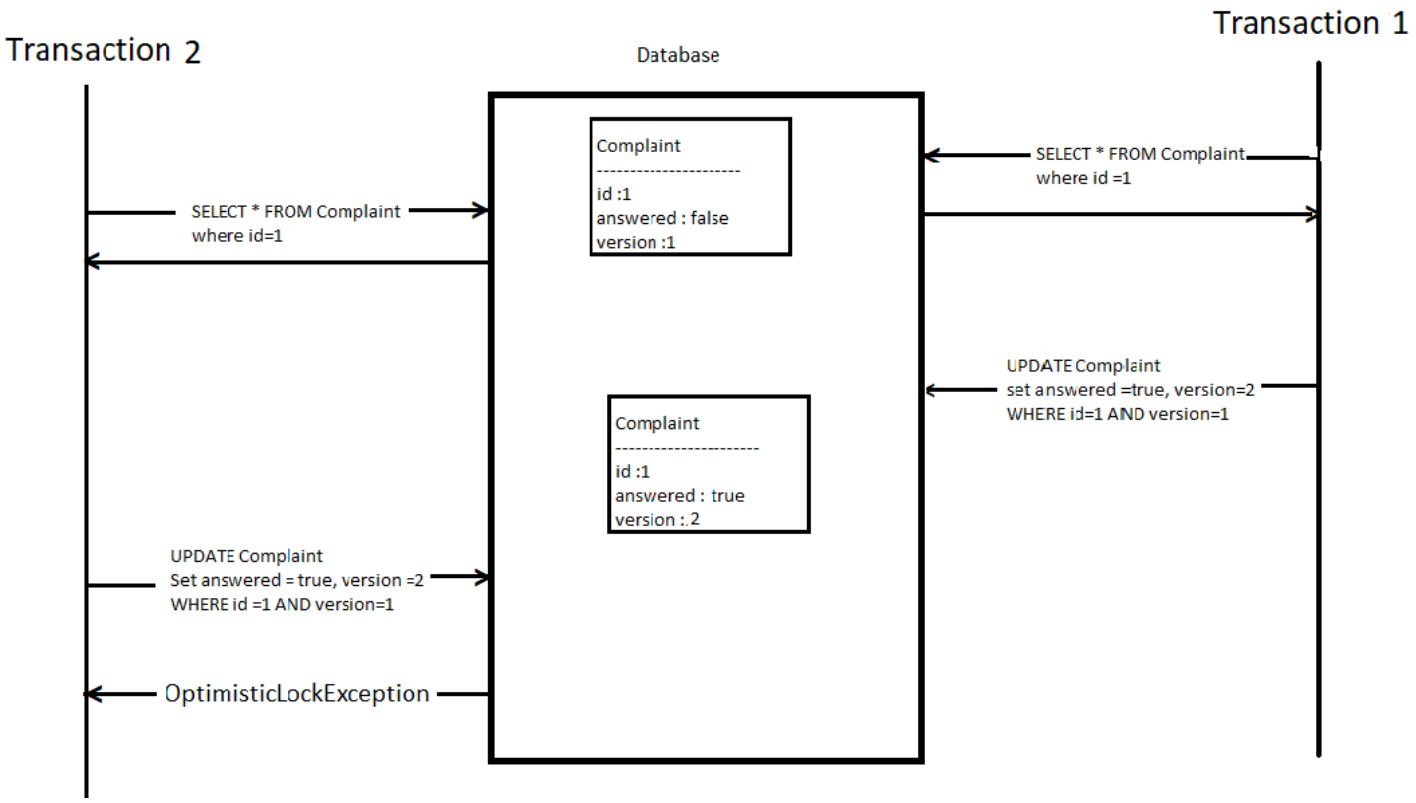
  Dva administratora sistema – admin1 I admin2 kliknu na istu žalbu I skoro u isto vreme kliknu na dugme za slanje odgovora. Međutim, jedan od njih je prvi to uradio, pa će njegov odgovor biti prosleđen, dok će drugom administratoru to biti onemogućeno.

- Rešenje problema

  Problem je rešen pomoću optimistic locking-a. U Complaint je dodato novo polje, private Long version I dodeljena mu je anotacija @Version.

  Pomoću @Version anotacije postiže se to, da se pre čuvanja izmene da polje boolean answered dobija vrednost true, proveri verzija podatka koji se čuva. Ako je verzija podatka ista, kao u trenutku kada je učitana iz baze, podatak se ažurira a polje version se inkrementira. U slučaju da admin1 malo brže klikne na zahtev za slanje odgovora, njegov zahtev će pre stići do servera, pri čemu će se njegova izmena sačuvati i njegov će odgovor biti poslat. A za admina2 će se proveravati verzija podatka, i ona neće biti ista kao u trenutku preuzimanja informacija iz baze, jer je admin1 u međuvremenu uradio izmenu. Iz tog razloga dolazi do ObjectOptimisticLockingFailureException-a kada se u okviru ComplaintService klase, u metodi update pozove complaintRepository.save(complaint).

- Tok


Transaction 2
Database
Transaction 1
Complaint
id :1
answered : false
version :1
SELECT * FROM Complaint
where id=1
SELECT * FROM Complaint
where id =1
UPDATE Complaint
set answered =true, version=2
WHERE id=1 AND version=1
Complaint
id :1
answered : true
version : 2
UPDATE Complaint
Set answered = true, version =2
WHERE id =1 AND version=1
OptimisticLockException

2. Prilikom izdavanja eRecepta se izdaju ili svi ili ni jedan lek i stanje leka u apoteci se azurira.

- Opis situacije

  Pacijent radi upload QR koda I prikazuju mu se sve apoteke koje imaju na stanju sve lekove I količine istih koji se nalaze na eReceptu. Pacijent bira apoteku iz koje želi da preuzme lekove klikom na dugme. Šalje se zahtev na serves da se ažuriraju koliične lekova u izabranoj apoteci.

- U čemu je problem?

  U toku odabira apoteke iz koje će se preuzeti lekovi, može se desiti da se dostupna količina određenog leka izmeni pri čemu onda neće svi lekovi sa eRecepta biti dostupni u izabranoj apoteci.
  Odnosno, ako transakcija 1 iščita trenutno stanje lekova u apotekama, a u međuvremenu druga kokurentna transakcija uradi update količina, znači da je transkacija 1 ištitala podatke koji nisu više validni.
  Ovo je problem Non-repeatable reads.

- Rešenje problema

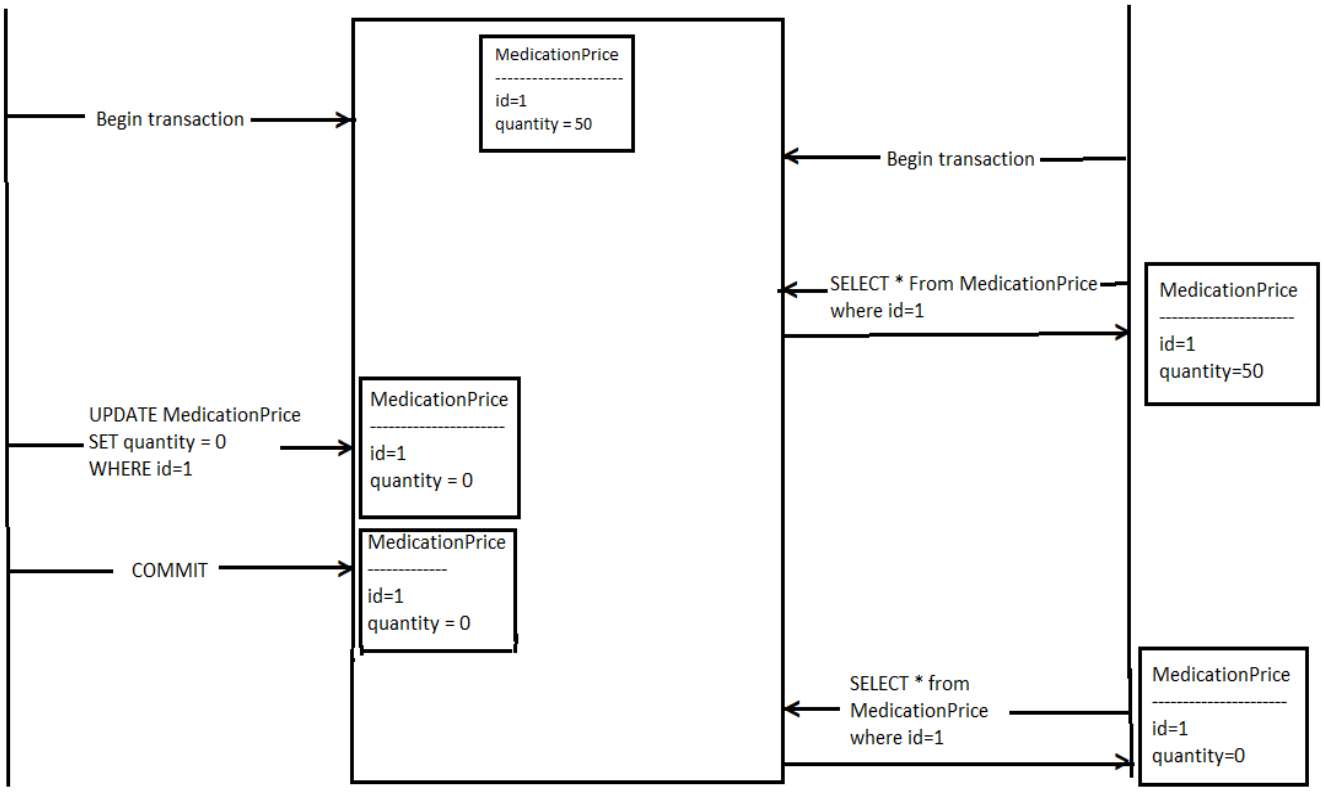
  Problem je rešen pomoću @Transactional anotacije I ISOLATION_READ_COMMITTED.
  U EPrescriptionServisu metoda proccedEreipt(..) je anotirana sa anotacijom @Transactional.

  Metoda save(..) u EPrescriptionServisu je anotirana sa @Transactional(readOnly = false, propagation = Propagation.REQUIRES_NEW), pri čemu je sa readOnly omogućena izmena podataka a sa propagacijom REQUIRES_NEW je omogućeno kreiranje nove transakcije I ako neka već postoji.

  Metoda updateMedicineQuantityEreceipt(..) u MedicationPriceServisu je takođe anotirana sa @Transactional(readOnly = false, propagation = Propagation.REQUIRES_NEW).

  Na ovaj način je omogućeno da ili sve akcije budu izvršene, ili sve budu poništene.

- Tok


MedicationPrice
id=1
quantity = 50
Begin transaction
Begin transaction
SELECT * From MedicationPrice
where id=1
MedicationPrice
id=1
quantity=50
UPDATE MedicationPrice
SET quantity = 0
WHERE id=1
MedicationPrice
id=1
quantity = 0
COMMIT
MedicationPrice
id=1
quantity = 0
SELECT * from
MedicationPrice
where id=1
MedicationPrice
id=1
quantity=0

3. U jednom trenutku, samo jedan system admin može da izmeni loyalty program

- Opis situacije

  Amidnistrator sistema pregleda trenutni loyalty program. Unošenjem novih vrednosti, šalje se zahtev ka serveru za izmenu programu. Server zatim pokuša da sačuva izmenjene vrednosti u bazu.
- U čemu je problem?

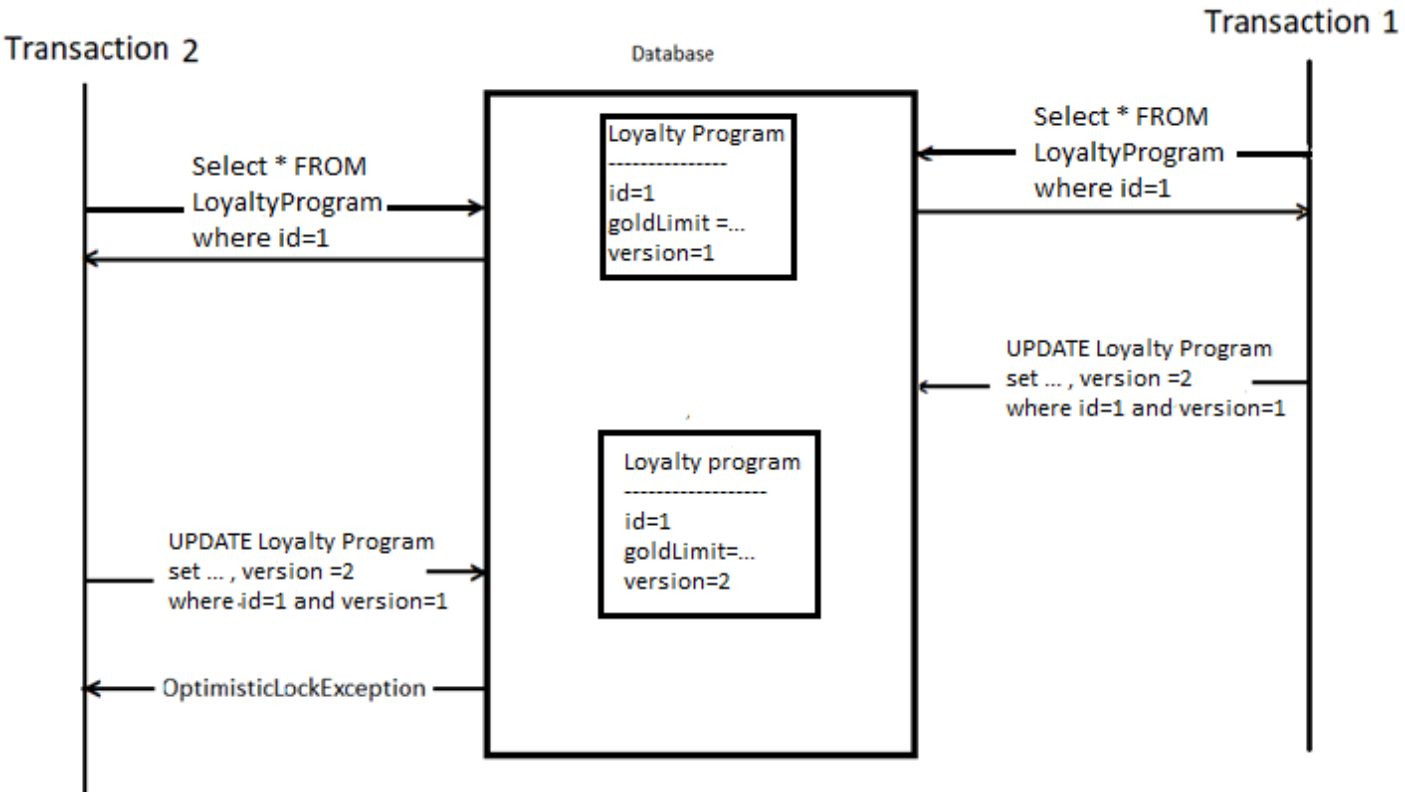
  Dva administratora sistema – admin1 I admin2 pregledaju trenutni loyalty program I skoro u isto vreme kliknu na dugme za izmenu. Međutim, jedan od njih je prvi to uradio, pa će njegova izmena biti sačuvana, dok će drugom administratoru to biti onemogućeno.

- Rešenje problema

  Problem je rešen pomoću optimistic locking-a. U Loyalty program je dodato novo polje, private Long version I dodeljena mu je anotacija @Version.

  Pomoću @Version anotacije postiže se to, da se pre čuvanja izmenjnih vrednosti programa, proveri verzija podatka koji se čuva. Ako je verzija podatka ista, kao u trenutku kada je učitana iz baze, podatak se ažurira a polje version se inkrementira. U slučaju da admin1 malo brže klikne na zahtev za slanje odgovora, njegov zahtev će pre stići do servera, pri čemu će se njegova izmena sačuvati i njegov će odgovor biti poslat. A za admina2 će se proveravati verzija podatka, i ona neće biti ista kao u trenutku preuzimanja informacija iz baze, jer je admin1 u međuvremenu uradio izmenu. Iz tog razloga dolazi do ObjectOptimisticLockingFailureException-a kada se u okviru LoyaltyProgramService klase, u metodi save pozove loyaltyProgram.save(loyaltyProgram).

- Tok


Transaction 2
Database
Transaction 1
Loyalty Program
---
id=1
goldLimit =...
version=1
Select * FROM
LoyaltyProgram
where id=1
Select * FROM
LoyaltyProgram
where id=1
UPDATE Loyalty Program
set ... , version =2
where id=1 and version=1
Loyalty program
---
id=1
goldLimit=...
version=2
UPDATE Loyalty Program
set ... , version =2
where id=1 and version=1
OptimisticLockException