SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINAR

Transakcije u sustavu zasnovanom na uslugama

Maja Legac

Voditelj: *Doc.dr.sc. Domagoj Jakobović*

Zagreb, travanj, 2010.

Sadržaj

1. Uvod

U seminarskom radu razvijena je mrežna usluga u programskom jeziku Python uz promoć radnog okvira Django za rezervaciju kino ulaznica. Usluga omogućava korisniku prijavu na stranicu te rezervaciju jednog ili više slobodnih mjesta. Svaka rezervacija je jedna transakcija. U istom trenutku više korisnika pristupa stranici i može se odlučiti na rezervaciju. Javljaju se problemi održavanja konzistentnosti podataka.

U drugom poglavlju uvodi se pojam transakcije te se navode osnovna svojstva i objašnajvaju problemi međudjelovanja transakcija. U trećem poglavlju opisan je "*two phase commit*" protokol kojim se rješavaju neki od problema međudjelovanju transakcija, te se navode neki njegovi nedostatci. U četvrtom poglavlju dan je zaključak nakon kojeg slijedi popis literature i sažetak.

2. Transakcije

U računarskoj znanosti, obrada transakcija je ekvivalentna obradi informacija koje su podijeljene u pojedinačne, nedjeljive operacije nazvane transakcije. Transakcija se definira kao jedinica rada nad sustavom koja se sastoji od niza logički povezanih operacija koje se obavljaju u potpunosti ili se ne obavljaju. Njihova obrada je oblikovana za održavanje računalnih sustava (tipično baze podataka) u poznatom i konzistentnom stanju jer osiguravaju da međuzavisne operacije na sustavu se izvedu uspješno ili se uspješno otkažu. Omogućuje da više pojedinačnih operacija budu povezane kao jedinstvena, nedjeljiva cjelina, te da se sve operacije izvršavaju bez greške ili se ne izvrše. Ako neke od operacija ne završe uspješno, cijeli sustav se vraća u prethodno, dosljedno stanje. Nakon što sve transakcije uspješno završe sustav se nalazi u novom dosljednom stanju iz kojeg se nije moguće vratiti u prethodno stanje. Štiti od sklopovskih i programskih pogrešaka koje mogu ostaviti transakciju djelomično (ne)dovršenu i sustav u nedosljenom stanju. Obrađuju se u strogom kronološkom redosljedu. Ako *n+1* transakcija namjerava mijenjati isti opseg baze podataka kao i *n - ta* transakcija, *n + 1* transakcija ne započinje sve dok *n - ta* transakcija nije završila. Prije no što bilo koja transakcija završi, sve ostale transakcije koje djeluju na isti dio sustava moraju biti obavljene.

2.1. Svojstva transakcija

Četiri osnovna svojstva transkacija su : nedjeljivost, dosljednost, odvojenost i trajnost tzv. ACID svojstva.

2.1.1. Nedjeljivost

Nedjeljivost (eng. *atomicity*) zahtjeva da sve promjene u sustavu slijede "sve ili ništa" pravilo.

Svaka transkacija je nedjeljiva. Ako jedan dio transakcije ne uspije, cijela transakcija ne uspjeva i sustav ostaje nepromijenjen, tj. obavi se u cjelosti ili se ne obavi. Od izuzetne je važnosti da se zadrži nedjeljiva struktura transakcije bez obzira na pad baze podataka, operacijskog sustava ili neke aplikacije. Ne može se podijeltit na podzadatke, a mora biti obrađena u cjelosti ili uopće ne. Nedjeljivost znači da korisnik ne mora brinuti o posljedicama nepotpune transakcije.

Transakcija može ne uspijeti zbog više razloga :

1. Kvara na sklopovlju: kvara na disku koji spriječava da neke od promjena u sustavu stupe na snagu.
2. Pada sustava : Korisnik gubi vezu s aplikacijom prije no što je dao sve potrebne informacije.
3. Pada baze podataka : npr. bazi podataka nestane prostora za čuvanje podataka
4. Greške u radu aplikacije : aplikacija pokušava poslati podatak koji narušava pravila baze podataka, npr. pokušaj stvaranja novog filma bez unosa imena filma.

2.1.2. Dosljednost

Svojstvo dosljednosti (eng. *consistency*) osigurava da sustav ostane u dosljednom stanju, preciznije kaže da bilo kojom transakcijom sustav će prijeći iz jednog dosljednog stanja u drugo. Svojstvo dosljednosti ne daje informaciju o tome kako bi sustav trebao rukovati s nedosljednostima nego kako se osigurati da sustav bude u dosljednom stanju na kraju transakcije. Ako iz nekog razloga transakcija koja se izvršava krši pravilo dosljednosti,

cjeloukupna transakcija se poništava (eng. *rollback*) i sustav se vraća u prethodno stanje (prije transakcije). Dakle, ako je shema sustava takva da određeno polje je za čuvanje cjelobrojnih vrijednosti, sustav može odlučiti odbaciti pokušaje spremanja realnih vrijednosti u to polje ili može odlučiti odbaciti realni dio. Obje opcije zadržavaju konzistentnost. Pravilo dosljednosti se odnosi samo na pravila integriteta koja su unutar njegovog dosega (eng. *scope*). Dosljednost je određena cjelovitošću (eng. *integrity*) ili svrhovitošću podataka u sustavu. Sljedeći mehanizmi zadržavaju dosljednost:

1. otkazivanje transakcije (eng. *rollback*) i povratak u prijašenje konzistentno stanje.
2. brisanje svih struktura podataka koji pokazuju na obrisanu strukturu (eng. *cascade delete*)
3. nuliranje (eng. *nullify*) relevanthih pokazivača u svim strukturama koja pokazuju na obrisanu strukturu.

Razvijatelji aplikacija su odgovorni za osiguravanje dosljednosti aplikacijske razine.

2.1.3. Odvojenost

Odvojenost (eng. *isolation*) se odnosi na zahtjev da različite transakcije nisu svjesne jedna druge. Kada se više paralelnih transakcija odvija istovremeno, njihov učinak mora biti kao da se odvijaju jedna iza druge. Ovo svojstvo služi za prikrivanje djelomičnih rezultata.

2.1.4. Trajnost

Kada je transakcija završila i korisnik je obavješten o uspješnosti transakcije, njeni učinici neće biti izgubljeni. Ako je uspješno završila, učinci koje je načinila moraju biti postojani i u slučaju kvara sustava.

Učinci moraju biti postajani i u slučaju kvara sustava netom prije završetka transakcije. To je svojstvo stalnosti (eng. *durability*) konačnih rezultata. Mnoštvo sustava implementiraju trajnost zapisivanjem transakcije u zapisnik transakcija (eng. *transaction log*) koji se može iskoristiti za povratak u prethodno stanje u slučaju pada. Transakcija se smatra završenom tek nakon što je ušla u zapisnik. Trajnost ne podrazumjeva trajno stanje sustava. Transakcije mogu mijenjati podatke promijenje prijašnjim transakcijama i svejedno ne prekrštiti svojstvo trajnosti.

2.2. Međudjelovanje transakcija

Osiguravanje ACID svojstava u raspodijeljenim transakcijama preko Interneta gdje ni jedan posebni čvor nije odgovoran za sve podatke koji sudjeluju u transakciji donosi dodatne poteškoće. Transakcija se izvodi nad više primjenskih podsustava na Internetu. Jedna transakcija može uspješno završiti svoj dio, no zbog pada veze (eng. *network connection*) ili ispada nekog čvora učinjene promjene ne mogu stupiti na snagu, nego se cijeli sustav mora vratiti u prethodno stanje (eng. *rollback*). Ostvarivanje kontrole izvođenja transakcija u sustavima zasnovanim na uslugama je veoma složeno zbog međudjelovanja transakcija.

2.2.1. Prljavo čitanje

Nepotvrđena promjena vrijednosti podataka koje koristi paralelena transakcija. Npr. jedna od transakcija može koristiti vrijednost koja nikada nije postojala.

2.2.2. Neponovljivo čitanje

Potvrđena promjena vrijednosti podataka koje koristi paralelna transakcija. Npr. ponovnim čitanjem paralelna transakcija dobiva različitu vrijednost.

2.2.3. Nepostojano čitanje

Potvrđena promjena strukture (i vrijednosti) podataka koje koristi paralelna transakcija. Npr. ponovnim čitanjem paralelna transakcija dobiva novostvorenu vrijednost.

Prilikom razriješavanja međusobnog utjecaja transakcija postoje četiri razine odvojenosti : čitanje nepotvrđenih transakcija, čitanje potvrđenih transakcija, ponovljivo čitanje, slijedno čitanje. Slijedno čitanje nije pogodno za sustave zasnovane na uslugama zbog stroge kontrole izvođenja (podtransakcije se ne mogu obaviti bez suglasnosti koordinatora) čime se gubi otvorenost tj. neovisnost izgradnje usluge. 2PC (eng. *two phase commit*) protokol pruža nedjeljivost u raspodijeljenim transakcijama kako bi se osigurala da svaki sudionik slaže treba li se transakcije prihvatiti ili ne. Ukratko, u prvoj fazi jedan čvor (koordinator) ispituje ostale čvorove (kohorte) i kada se svi dogovore da su spremni, onda u drugoj fazi koordinator formalizira transakciju.

3. "Two phase commit" protokol

U obradi transakcija, baze podataka i računalne mreže, 2PC je vrsta protokola za nedjeljive akcije. To je raspodijeljeni algoritam za kordinaciju svih procesa koji sudjeluju u raspodijeljenim nedjeljivim transakcijama s obzirom na to da li će se transakcija izvršiti ili otkazati, pri čemu se radi povratak sustava u prijašnje stanje (eng. *rollback*). To je specijalizirani tip protokola usuglašavanja. Dosiže svoj cilj čak i u mnogo slučajeva privremenih grešaka u sustavu (ispadanje čvorova, gašenje procesa, komunikacijeske greške i sl.) i zbog toga se mnogo koristi. Ipak nije u potpunosti otporan na sve tipove pogrešaka, te je ponekad ljudska intervencija potrebna. Dvije faze algoritma su pošalji - zahtjevaj i izvrši.

U prvoj fazi proces koordinator pokušava pripremiti sve procese sudionike da poduzmu potreben korake za izvršavanje ili odustajenje od transakcije. Zatim se vrši glasovanje ("Da" - izvrši ili "Ne" - otkaži). U drugoj fazi na temelju rezulata glasovanja koordinator odlučuje ili da izvrši ili otkaže transakciju. O rezultatu obavještava ostale sudionike. Sudionici zatim slijede potrebne korake sa svojim resursima i svojim dijelovima u transakciji.

3.1. Algoritam

Protokol radi na sljedeći način : jedan čvor je koordinator, koji je glavna stranica, a ostali čvorovi u mreži su kohorta (eng. *cohorts*). Protokol pretpostavlja da postoji stabilno spremište na svakom čvoru s unaprijednim zapisnikom (eng. *write - ahead log*), te da se čvor ne ruši uvijek tako da podaci zapisani u unaprijenom zapisniku nisu izgubljeni ili pogrešni i da svaka dva čvora mogu međusobno komunicirati. Posljednja pretpostavka nije toliko ograničavajuća, pošto obična komunikacijska mreža može biti preusmjerena. Prve dvije pretpostavke su mnogo jače. Ako je čvor u cjelosti uništen onda podaci mogu biti izgubljeni. Protokol inicira koordinator nakon što je posljednji korak transakcije dosegnut. Kohorte onda odgovaraju sa "Da" ili "Ne" ovisno o uspješnosti završetka transakcije.

3.1.1. Faza pošalji - zahtjevaj

Faza glasanja.

1. Koordinator šalje upite kohortama da pošalju svoje odgovore i čeka dok ne primi poruku od svih.

2. Kohorte izvršavaju transakciju do točke gdje se od njih traži da pošalju poruku. Pišu svoj odgovor u svoj poništi zapisnik (eng. *undo log*) i ponovi zapisnik (eng. *redo log*).
3. Svaka kohorta odgovara s "Da" ako je transakcija uspjela ili "Ne" ako nije uspjela.

3.1.2. Faza izvršavanja

Faza završetka. Postoje dva slučaja : uspjeh ili neuspjeh.

Uspjeh :
Ako je koordinator primio poruku o suglasnosti od svih kohorti tijekom pošalji - zahtjevaj faze :

1. Koordinator šalje poruku o uspjehu završetka transakcije svim kohortama.
2. Svaka kohorta završava operaciju i oslobađa sve resurse zaključane resurse tijekom transakcije.
3. Svaka kohorta šalje potvrdu koordinatoru.
4. Koordinator završava transakciju kada primi sve potvrde.

Neuspjeh :
Ako je barem jedna kohorta poslala poruku o odustajanju tijekom prve prve faze, onda :

1. Koordinator šalje poruku svim kohortama o odustajanju od transakcije (eng. *rollback message*).
2. Svaka kohorta poništava učinke transakcije i vraća sustav u prethodno stanje koristeći poništi zapisnik i oslobađa resurse koje je koristila tijekom transakcije.
3. Svaka kohota šalje potvrdu koordinatoru.

4. Koordinator poništava transakciju kada primi potvrde od svih kohorti.

3.2. Nedostaci

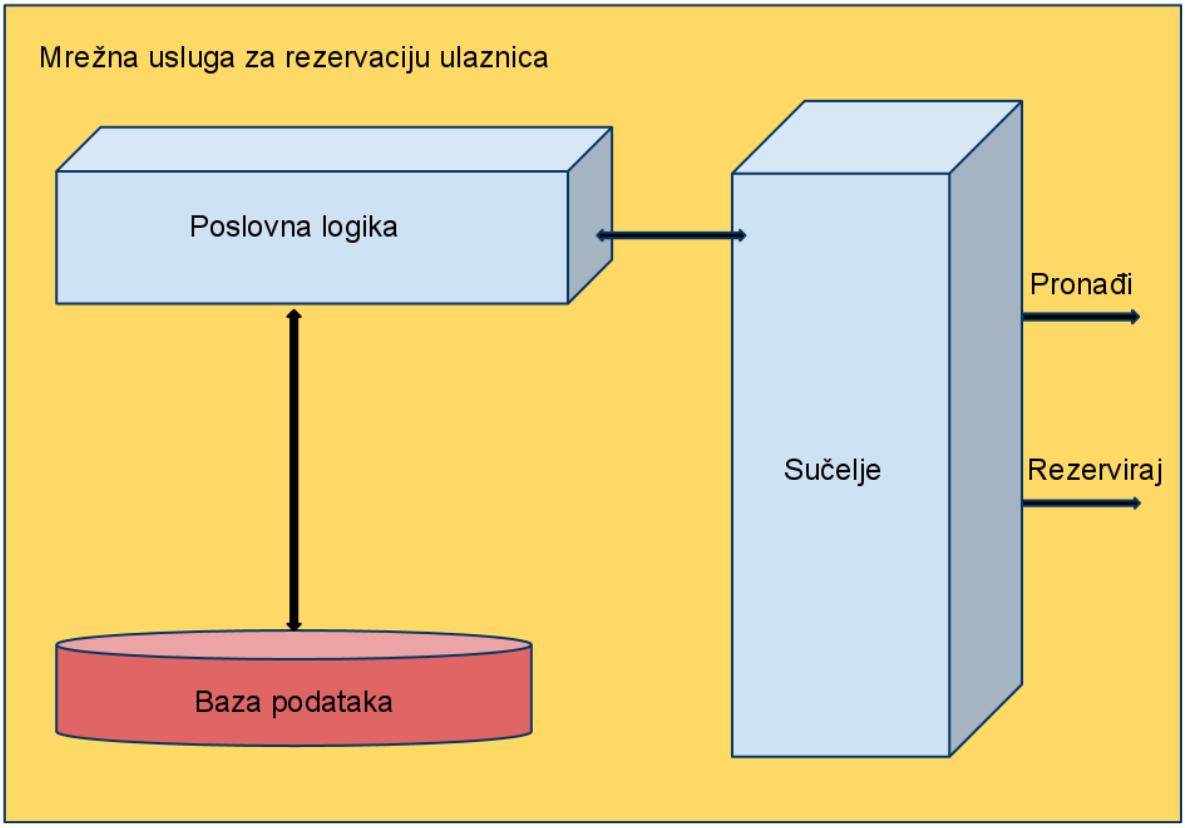
Najveći nedostatak 2PC protokola je to da je blokoirajući protokol. Čvor će blokoirati dok čeka poruke. Drugi procesi koji se natječu za resurse moraju čekati da proces koji trenutno koristi resusurse prvo ih oslobodi. Pojedniačni čvor će nastaviti čekati čak i ako su neki od čvorova prestali raditi. Ako čvor koordinator trajno prestane raditi, neke kohorte nikada neće dovršiti transakciju (neće postojati potvrda o uspješnosti završetka), što će uzrokovati da se neki od resursa nikada neće oslobodit. Algoritam može beskonačno blokoirati na sljedeći način : Ako je kohorta poslala poruku o suglasnosti koordinatoru, blokirati će sve dok ne primi poruku o izvšavanju ili odustajanju od transakcije. Ako je koordinator trajno prestao raditi, resursi koje kohorta koristi biti će trajno blokirani. Kada je koordinator poslao upit za izvršavanjem (eng. *query - to - commit*) kohortama, blokoirati će sve dok od svih kohorti ne primi odgovor. Ako je neka kohorta trajno prestala raditi, koordinator neće beskonačno blokirati ako postoji neko vrijeme čekanja odgovora. Koordinator nakon isteka vremena šalje poruku o odustajanju od transakcije. Ovakvo konzervativno ponašanje protokola je još jedan nedostatak : pristranije je odustajanju od transakcije.

4. Arhitektura usluge

U ovom poglavlju izložen je pregled arhitekture usluge visoke razine i detaljan opis REST sučelja. Potpoglavlje 4.1. izlaže arhitekturu, a potpoglavlje 4.2. izgled web klijenta.

4.1. Visoka razina arhitekture

Arhitektura usluge temelji se na osnovnim načelima izgradnje Web usluga koje koriste REST model. Na slici 1 prikazana je arhitektura usluge na najvišoj razini.



Slika 1 : Arhitektura sustava na visokoj razini

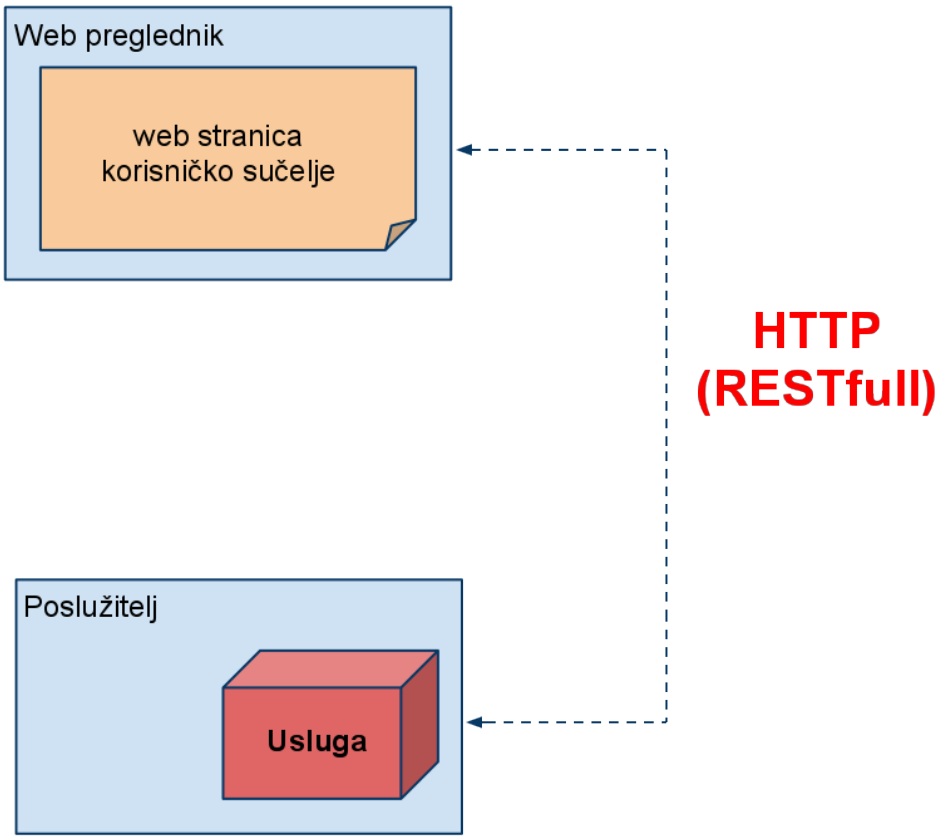
Sučelje arhitekture vanjskom svijetu izlaže tri metode: *prijava, pronađi i rezerviraj*.

Metoda *prijava* koristi se za prijavljivanje novog korisnika u bazu korisnika usluge. Samo prijavljeni korisnici mogu rezervirati mjesta. Metoda rezerviraj omogućuju rezervaciju jednog ili više mjesta u dvorani. Poziv metode *pronađi* obrađuje se u poslovnoj logici i uzrokuje jedan ili više upita prema bazi podataka kojima se izvlači popis. Prilikom pronađenog filma mogu se vidjeti podatci o tom filmu (vremena igranja i dvorane).

4.2.Web klijent

Za jednostavnu uporabu od strane krajnjih korisnika, bez potrebe za poznavanjem REST arhitekturalnog stila, potrebno je izgraditi jednostavno sučelje za pozivanje usluge. U tu svrhu iskoristit će se web aplikacija pokrenuta iz web preglednika, kako je prikazano na slici 2.

Korisnik iz web preglednika odabire način rada usluge. Aplikacija se brine za stvaranje valjanog zahtjeva i šalje ga preko Interneta do poslužitelja s uslugom. Nakon obrade zahtjeva usluga istim putem vraća odgovor, koji web aplikacija pretvara u zapis pogodan za čitanje krajnjem korisniku koristeći web preglednik. Klijent nudi mehanizam za prijavu korsnika.



Slika 2 : Primjer korištenja usluge

Kako se za kompletnu poslovnu logiku, bazu filmova i nerezerviranih sjedala brine sama usluga, ovaj klijent bit će izrazito jednostavan i lagan. Cilj je na što jednostavniji i intuitivniji način korisniku omogućiti pretragu baze filmova i rezervaciju. Činjenica kako se u pozadini poziva i koristi udaljena usluga potpuno je sakrivena.

5. Zaključak

Za razvoj web usluge odabran je programski jezik Python i radni okvir Django. Iako prvotno osmišljen za razvoj drugačijeg tipa aplikacija, onih temeljenih na objavljivanju i uređivanju sadržaja, okvir se pokazao izuzetno prikladnim i za izradu mrežnih usluga. Jednostavan i pragmatičan dizajn omogućuje brz razvoj na višoj razini apstrakcije, bez pretjeranog zamaranja s detaljima oko konkretnih tehnologija poput HTML-a, SQL-a i slično.

Glavna značajka okvira je ušteda vremena. Naime, Django nije nastao kao komercijalni proizvod niti kao rezultat akademskog istraživanja na sveučilištu. Razvijen je isključivo za rješavanje konkretnih problema iz industrije. Temeljno načelo okvira je skraćivanje vremena potrebnog od ideje do konačnog upotrebljivog proizvoda. Korištenjem proširenja *Django-RESTify-API* dodatno je ubrzan razvoj usluge, iako je samo proširenje i dalje u fazi razvoja.

6. Literatura

1. Newcomer, E. Understanding web services : XML, WSDL, SOAP and UDDI
2. Transaction processing, 18.3. 2010. *Transaction procesing*, http://en.wikipedia.org/wiki/Transaction_processing, 20.4.2010.
3. ACID, 8.4.2010. *ACID,* http://en.wikipedia.org/wiki/ACID, 20.4.2010.
4. Two - phase commit protocol, 19.4.2010. *Two - phase commit protocol*, http://en.wikipedia.org/wiki/Two-phase_commit_protocol, 21.4.2010.
5. Srbljić, S., Benc, I., Skuliber, I., 7. predavanje, 13.11.2009. *Računarstvo zasnovano na uslugama : Transakcije,* http://www.fer.hr/predmet/rznu, 19.4.2010.
6. The Django Book: Version 2.0, 8.1.2009. *The Django Book*, http://www.djangobook.com/en/2.0/, 7.3.2010.

7. Sažetak

U sklopu ovog seminarskog rada razvijena je usluga koja korisnicima pruža mogućnost rezervacije slobodnih sjedala. Svaka rezervacija je jedna transakcija. Opisuju se svojstva transakcija, problemi koji se javljaju njihovim međudjelovanjem a karakteristični su za sustave zasnovane na uslugama i nudi se rješenje u obliku 2PC protokola.

Ključne riječi : mrežna usluga, transakcija, ACID svojstva, 2PC