Elektrotehnički fakultet
Univerziteta u Sarajevu

Projekat :

Program za rad nad bazama podataka

Baze Podataka

Studenti : Azinović Dejan
Hasanspahić Aldin
Hasić Haris

Sarajevo,
2.11.2014.

1. Opis teme

Kratko rečeno, baze podataka predstavljaju organizovan skup informacija i/ili podataka u računarski razumljivom obliku. Da bi se efikasno manipuliralo zapisima u bazi podataka koriste se razni programi za rad sa bazama podataka. Programi za rad sa bazama podataka omogućuju korisniku da kreira baze podataka, vrši upis i čitanje zapisa u bazu podataka, izmjene nad tim zapisima, i po potrebi brisanje zapisa iz baze podataka.

Programi za rad sa bazama podataka se koriste kao podrška u radu raznih organizacija. Glavne prednosti ovih programa su brzina pristupa podacima, mogućnost višestrukog pristupa podacima, uniformnost i slično. Primarni korisnici ovih sistema su administratori baza podataka koji upravljaju samom bazom podataka. Osim administratora baza podataka, ove programe koriste i programeri za baze podataka (*database developer-i*), kao i drugi korisnici sa dodjeljenim privilegijama (studenti ETF-a). Primjer upotrebe ovih programa za administratora baze podataka jeste kreiranje user-a, dodjeljivanje prava pristupa istim, dok programeri baze podataka kreiraju razne izvještaje koristeći programe za rad sa bazama podataka. Drugi primjer jeste upotreba u edukativne svrhe kod studenata.

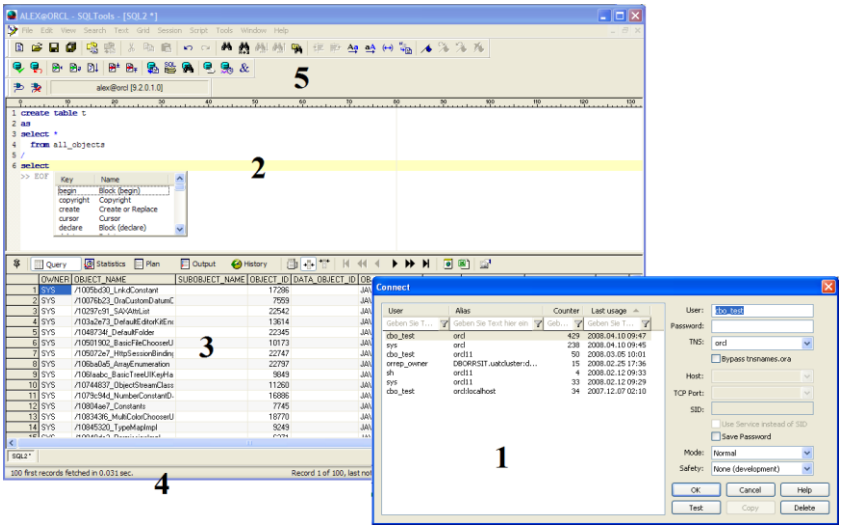
1.1 Opis funkcionalnosti sličnih aplikacija

Jedna od aplikacija sa sličnom funkcijom je SQLTools. Glavne prednosti te aplikacije su:

- Nije potrebna instalacija
- Relativno je mala pa se može lako prenijeti i na druge računare
- Brzina izvršavanja upita

Osnovni dijelovi aplikacije su:

1. Početni prozor za konektovanje na bazu
2. Prostor za pisanje SQL upita
3. Prostor u kojem se prikazuju rezultati upita
4. Linija za informacije
5. Toolbar-ovi sa raznim manjim funkcionalnostima



Slika 1 – Izgled interfejsa sličnog alata (*SQL Tools*)

Aplikacija omogućava pisanje SQL upita za Oracle bazu podataka. Neke od osnovnih funkcionalnosti su:

- Konekcija na Oracle bazu podataka
- Izvršavanje SQL upita
- Spašavanje i učitavanje skripti
- Rad sa više skripti istovremeno

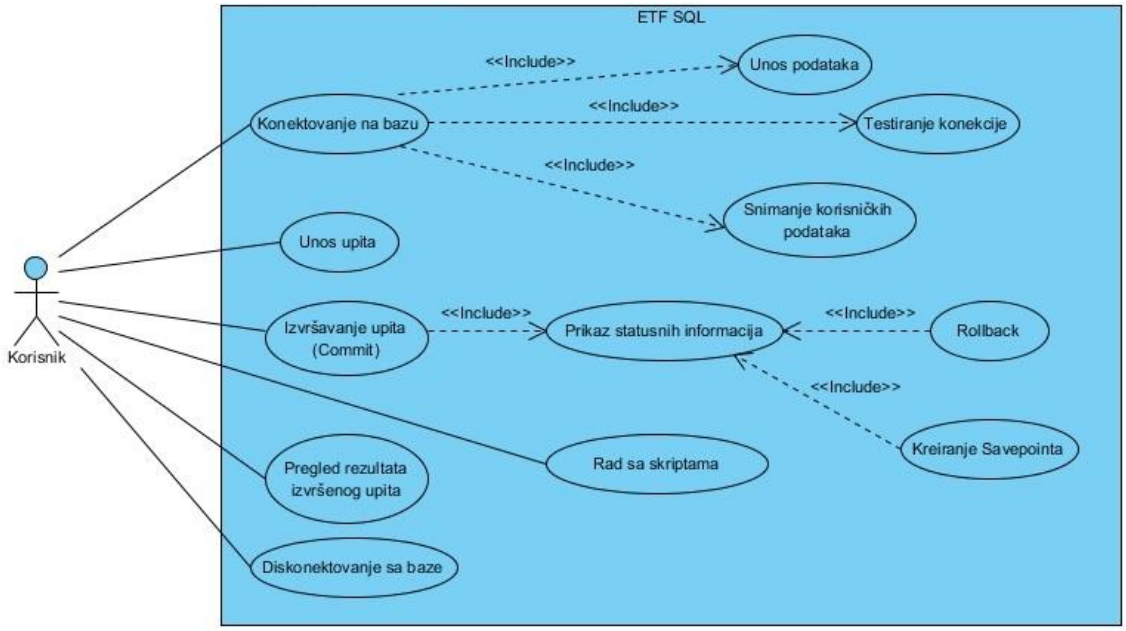
Pored navedenih funkcionalnosti SQLTools posjeduje i još neke naprednije funkcionalnosti. Najveća mana je što se aplikacija izvršava jedino pod Windows operativnim sistemom. Još jedna aplikacija za rad sa bazama podataka je Oracle SQL Developer. Ona nudi sve što i SQLTools, ali pored toga nudi i alate za razvoj i upravljanje bazama podataka, alate za razvoj PL/SQL aplikacija, te niz drugih alata. Najveća mana je veličina aplikacije. Većini korisnika je potreban samo dio funkcionalnosti koje nudi Oracle SQL Developer.

1.2 Funkcionalnosti ETF SQL

Pošto ovakvi programi uglavnom imaju predefinisanu okosnicu funkcija koje trebaju sadržavati, naša aplikacija, pod imenom ETF SQL bi od svih nabrojanih osnovnih funkcionalnosti trebala sadržavati većinu. Više ćemo se fokusirati na kvalitetno funkcionisanje aplikacije prije nego na dodavanje funkcionalnosti koje se koriste rijetko ili skoro nikako. To su sljedeće funkcionalnosti :

- Osnovni prozor za podešavanje konekcije na bazu
- Prostor za unos upita
- Odgovarajući prostor za prikazivanje rezultata izvršenog upita
- Statusne informacije
- Opcije za rad i manipulaciju skriptama

Početni use case dijagram naše aplikacije je dat na slici ispod.



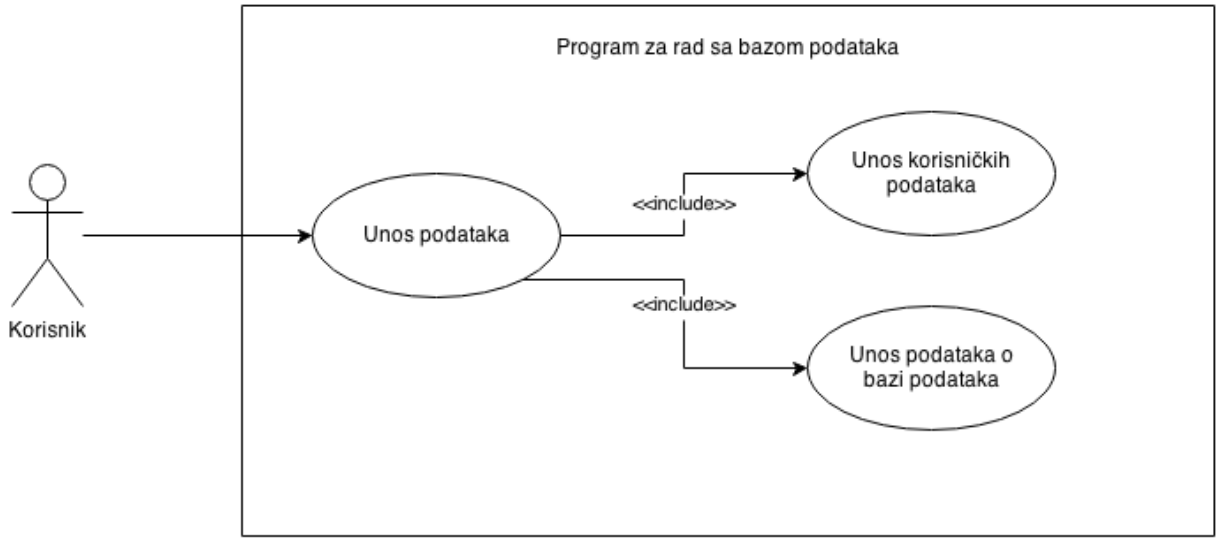
Slika 2 – Use Case dijagram ETF SQL aplikacije

2. Detaljni opis funkcionalnosti

U narednom dijelu će detaljno biti opisane osnovne funkcionalnosti koje će naš program sadržavati, a koje su navedene u prethodnom poglavlju. Biti će opisane samo one funkcionalnosti koje čine program oni što jeste, dakle SQL Tool alatom.

2.1 Konekcija na bazu podataka

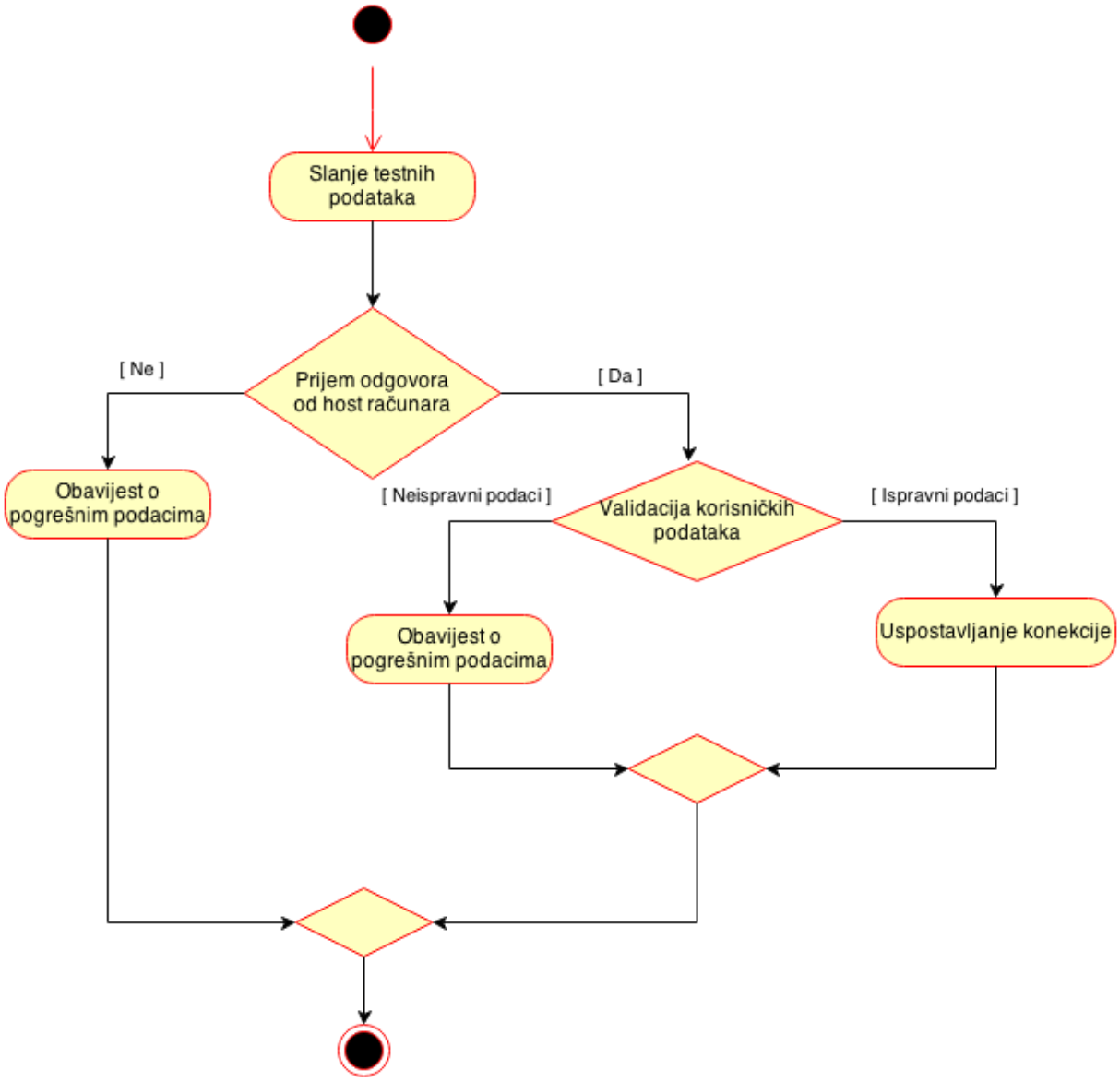
Prvi korak u radu sa bazom podataka jeste konektovanje na bazu podataka. Ono što je potrebno da bi se uspješno konektovali na bazu podataka su korisnički podaci (*username* i *password*), adresa host-a baze podataka, te eventualno port na kojem host računar komunicira sa klijentima.



Slika 3 – Use case dijagram konekcije na bazu podataka

Na slici 3 je prikazan use-case dijagram početka procesa konektovanja na bazu podataka. Unos podataka možemo podijeliti u dvije podgrupe, tj. unos podataka o samoj bazi podataka, te unos korisničkih podataka. Podaci o bazi podataka su adresa host-a baze podataka i port koji su uvijek isti za sve korisnike, dok su druga grupa podataka lični/korisnički podaci koji su jedinstveni za svakog korisnika baze podataka i uključuju korisničko ime i lozinku (*username* i *password*).

Nakon unosa ovih podataka, moguće je izvršiti konektovanje na bazu podataka. Prvo je potrebno testirati konekciju, odnosno provjeriti da li je baza podataka dostupna. Klijentski računar pokušava ostvariti konekciju sa računarom na kojem se nalazi baza podataka, na osnovu unesenih podataka, tj. adresi host-a i port-u na kojem radi host računar. Ukoliko host odgovori, uneseni podaci o adresi i portu hosta su ispravni te se prelazi na sljedeći nivo provjere. Host računaru se šalju korisnički podaci koji se validiraju. Tek nakon što host računar odgovori da su korisnički podaci ispravni, uspostavlja se trajna konekcija sa bazom podataka, i omogućava se korisniku da radi na bazi podataka. Ovaj proces je prikazan na dijagramu aktivnosti na slici 4.



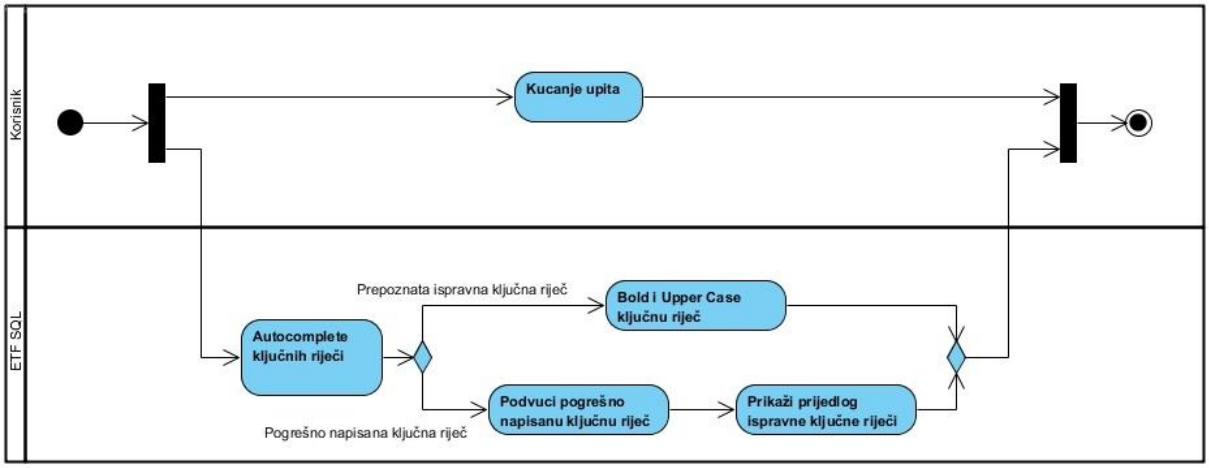
Slika 4 – Dijagram aktivnosti procesa testiranja i uspostavljanja konekcije sa bazom podataka

U prethodnom dijelu su prikazane sve aktivnosti potrebne da bi se korisnik uspješno konektovao sa bazom podataka. Cjelokupan proces kreće od unosa korisničkih i podataka o samoj bazi podataka. Nakon toga se testira postojanost veze sa bazom podataka, i ako je moguće uspostaviti vezu sa bazom podataka, validiraju se korisnički podaci. Ukoliko su validni, uspostavlja se trajna konekcija sa bazom podataka i korisniku se omogućava rad na istoj. Navedeni dijagrami pretpostavljaju da se traženi podaci stvarno unose. U slučaju nedostatka traženih podataka, program izbacuje poruku o greški i dalji tok konektovanja na bazu podataka se obustavlja.

Proces diskonektovanja sa bazom podataka je jednostavan i korisniku se omogućava da se samim gašenjem programa diskonektuje sa baze podataka. S obzirom na jednostavnost ove operacije, nema potrebe za opisivanjem putem UML-a.

2.2 Kucanje upita i prepoznavanje ključnih riječi

Ono što karakteriše svaki SQL alat jeste kucanje upita nad bazom podataka, ali ono što odvaja manje kvalitetne od boljih programa jeste sam način izvedbe i parsiranje unesenog teksta. Prepoznavanje i označavanje ključnih riječi u upitima, nadopunavanje (*autocomplete*) istih, naznačavanje pogrešnih unosa i sl. su sve odlike koje čine ovakve tipove programa intuitivnijim i jednostavnijim za rad ne samo ekspertnim korisnicima, već i korisnicima koji tek ulaze u svijet kucanja upita i rada nad bazom podataka. Napraviti ovu funkcionalnost kvalitetno predstavlja vitalan zadatak, pogotovo jer je ovaj program namijenjen prvenstveno za edukativne svrhe, odnosno za pomaganje studentima prilikom učenja rada sa *Oracle* bazom podataka.



Slika 5 – Activity dijagram operacije kucanja upita

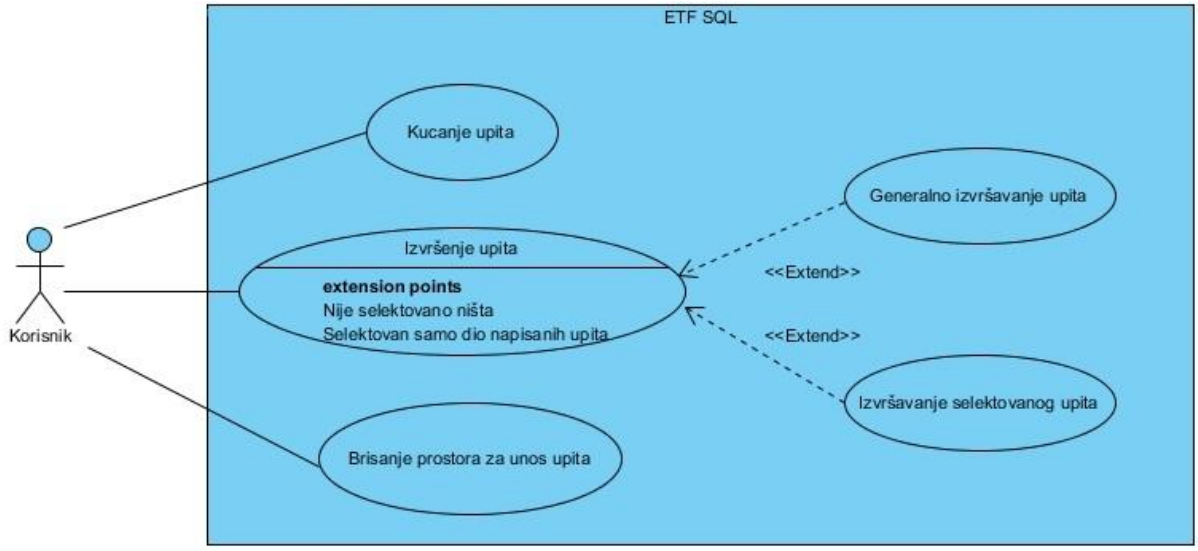
Sa datog dijagrama aktivnosti se jasno vidi tok kojim se odvijaju operacije prilikom unosa teksta u ETF SQL program. Za vrijeme dok korisnik kuca tekst u za to predviđeni prostor u pozadini se odvijaju procesi parsiranja teksta. Ako se prije njenog završetka prepozna neka ključna riječ kao što je SELECT, WHERE, FROM i sl. opcija *autocomplete* će korisniku ponuditi sve riječi koje su prepoznate i on jednostavnim klikom može odabrati neku od ponuđenih opcija. Ako korisnik odluči da ignoriše tu opciju, moguća su dva rezultata : da je ispravno otkucana ključna riječ ili da se desila neka sintaksna greška. U oba slučaja program mora brzo i efikasno reagovati. Ako je ispravno otkucana ključna riječ ona će, radi jasnijeg uvida u kôd biti boldirana i prevedena na velika slova. U suprotnom će biti podvučena i biti će prikazan prijedlog ispravno napisane ključne riječi.

2.2.1 Izvršavanje upita

Kada imamo napisani neki upit ono što logično sljeduje jeste njegovo izvršavanje. Izvršavanje upita će se moći izvršavati na dva načina i to :

- Izvršavanje svega što se nalazi u prostoru za kucanje teksta
- Izvršavanje samo selektovanih upita

Detaljniji prikaz ove funkcionalnosti je dat na use case dijagrmu na slici 6.



Slika 6 – Use case dijagram izvršenja ukucanih upita

2.2.2 Pregled rezultata

Ovo je veoma trivijalna funkcija koja ne zahtijeva dodatno opisivanje sa dijagramima jer rezultati koji se dobiju nekim upitom nad bazom podataka će jednostavno biti izlistani tabelarno. Nad tom tabelom se neće moći vršiti nikakve dodatne modifikacije.

2.3 Commit, Rollback i kreiranje Savepointa

U narednim poglavljima će detaljno biti opisane operacije modifikovanja baze tj. operacija komitanja (*commit*), vraćanja baze podataka na prethodno stanje (*rollback*) i kreiranje sigurnosnih tačaka (*savepoint*).

2.3.1 Commit

Naredba COMMIT se koristi da bi se u bazi spasile promjene nastale tokom transakcije. Promjene izazvane naredbama INSERT, UPDATE i DELETE se ne spašavaju automatski, već ih je potrebno eksplicitno spasiti pomoću naredbe COMMIT. Sintaksa je sljedeća :

```
COMMIT;
```

Primjer :

```
+----+---------+--------+----------+
| ID | IME     | GODINE |  PLATA   |
+----+---------+--------+----------+
|  1 | Aldin   |     21 |  2000.00 |
|  2 | Dejan   |     22 |  1500.00 |
|  3 | Haris   |     22 |  2000.00 |
+----+---------+--------+----------+
```

Razmotrimo sljedeći upit :

```
DELETE FROM zaposlenici WHERE plata = 2000;
COMMIT;
```

Prva linija briše zaposlenike Aldin i Haris, ali se promjene nad bazom spašavaju tek nakon naredbe COMMIT. Nakon commit-a će baza izgledati ovako:

```
+----+---------+--------+----------+
| ID | IME     | GODINE |  PLATA   |
+----+---------+--------+----------+
|  2 | Dejan   |     22 |  1500.00 |
+----+---------+--------+----------+
```

2.3.2 Rollback

Komanda ROLLBACK se koristi da se ponište transakcije koje još uvijek nisu spašene u bazu. Mogu se poništiti jedino one transakcije koje su se desile nakon posljednje COMMIT ili ROLLBACK naredbe. Sintaksa je sljedeća:

```
ROLLBACK;
```

Primjer:

```
+----+---------+--------+----------+
| ID | IME     | GODINE |  PLATA   |
+----+---------+--------+----------+
|  1 | Aldin   |     21 |  2000.00 |
|  2 | Dejan   |     22 |  1500.00 |
|  3 | Haris   |     22 |  2000.00 |
+----+---------+--------+----------+
```

Neka su izvršene sljedeće naredbe :

```
DELETE FROM zaposlenici WHERE plata = 2000;
ROLLBACK;
```

Podaci u bazi će ostati nepromijenjeni :

```
+----+---------+--------+----------+
| ID | IME     | GODINE |  PLATA   |
+----+---------+--------+----------+
|  1 | Aldin   |     21 |  2000.00 |
|  2 | Dejan   |     22 |  1500.00 |
|  3 | Haris   |     22 |  2000.00 |
+----+---------+--------+----------+
```

2.3.3 Savepoint

Pomoću naredbe SAVEPOINT je moguće definisati mjesto na koje se možemo vratiti tokom transakcije u slučaju da dođe do neke greške. Sintaksa je sljedeća:

```
SAVEPOINT ime_savepointa;
```

Naredba koja omogućava vraćanje na mjesto savepoint-a je naredba ROLLBACK. Ona ima sljedeću sintaksu:

```
ROLLBACK TO ime_savepointa;
```

Ovo će poništiti sve promjene nad bazom koje su se desile nakon što je definisan savepoint. Da nismo definisali savepoint jedino mjesto gdje bi se mogli vratiti je početak transakcije. Sljedeći primjer to malo bolje objašnjava:

```
+----+---------+--------+----------+
| ID | IME     | GODINE |  PLATA   |
+----+---------+--------+----------+
|  1 | Aldin   |     21 |  2000.00 |
|  2 | Dejan   |     22 |  1500.00 |
|  3 | Haris   |     22 |  2000.00 |
+----+---------+--------+----------+
```

```
SAVEPOINT s1;
DELETE FROM zaposlenici WHERE ID = 1;
SAVEPOINT s2;
DELETE FROM zaposlenici WHERE ID = 2;
SAVEPOINT s3;
DELETE FROM zaposlenici WHERE ID = 3;
```

U ovom trenutku su izbrisani svi zaposlenici, ali promjene još uvijek nisu spašene u bazi. Pretpostavimo da smo napravili grešku kada smo izbrisali trećeg zaposlenika. Savepoint s3 nam omogućava da ispravimo tu grešku:

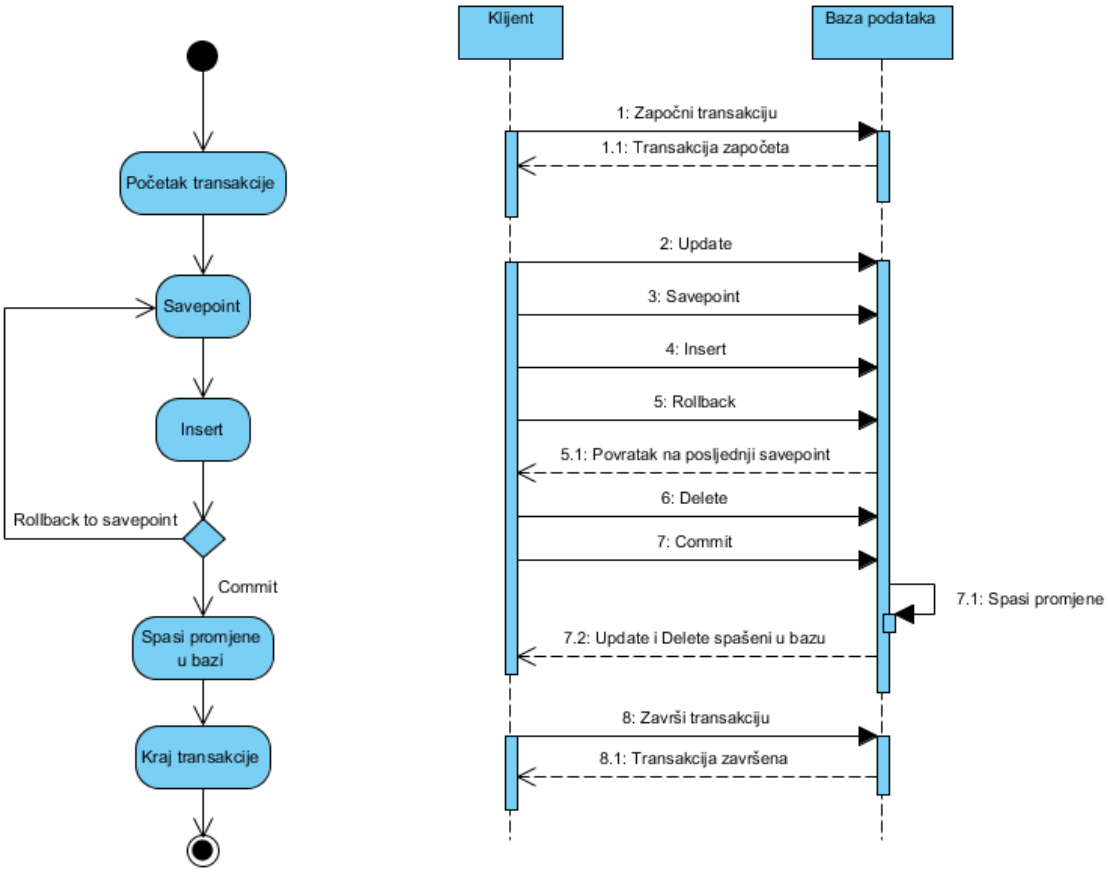
```
ROLLBACK TO s3;
```

Izgled baze nakon commit-a će biti sljedeći:

```
+----+---------+--------+----------+
| ID | IME     | GODINE |  PLATA   |
+----+---------+--------+----------+
|  3 | Haris   |     22 |  2000.00 |
+----+---------+--------+----------+
```

2.3.4 UML dijagrami navedenih operacija

Radi grafičke ilustracije navedenih funkcionalnosti su kreirana dva UML dijagrama. S lijeve strane se nalazi dijagram aktivnosti koji modelira jednu jednostavnu transakciju. Nakon umetanja sloga u bazu moguće je ili vratiti se na zadnji savepoint pomoću naredbe ROLLBACK ili spasiti promjene u bazu pomoću naredbe COMMIT. Sa desne strane je dat dijagram sekvence.



Slika 7 – Dijagrami aktivnosti i sekvence za *commit*, *rollback* i *create savepoint*