SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
ELEKTROTEHNIČKI FAKULTET

Stručni studij

CALL & RECHARGE HISTORY

Projektni zadatak iz kolegija
Baze podataka

Ime Prezime, Axxxx

Osijek, 2011.

Sadržaj

I. Opis projektnog zadatka

Call&Recharge History

Opis:

Potrebno je osmisliti i realizirati bazu podataka koja može učinkovito poslužiti za potrebe praćenja povijesti događaja unutar mobilne mreže, vezano za svakog pojedinog "Pre-paid" korisnika. U fazi analize, utvrđeno je da poslužitelj (provider) dobiva veliku količinu podataka u obliku tekstualnih datoteka koje dolaze s raznih mrežnih elemenata, te se isti podaci trebaju efikasno učitati u radnu bazu podataka. Poslužitelj razlikuje podatke o povijesti poziva i uplata na račun svojih PPS (Pre-Paid Service) korisnika, te podatke o trenutnom stanju korisničkih računa. Osnovni zahtjevi korisnika su slijedeći:

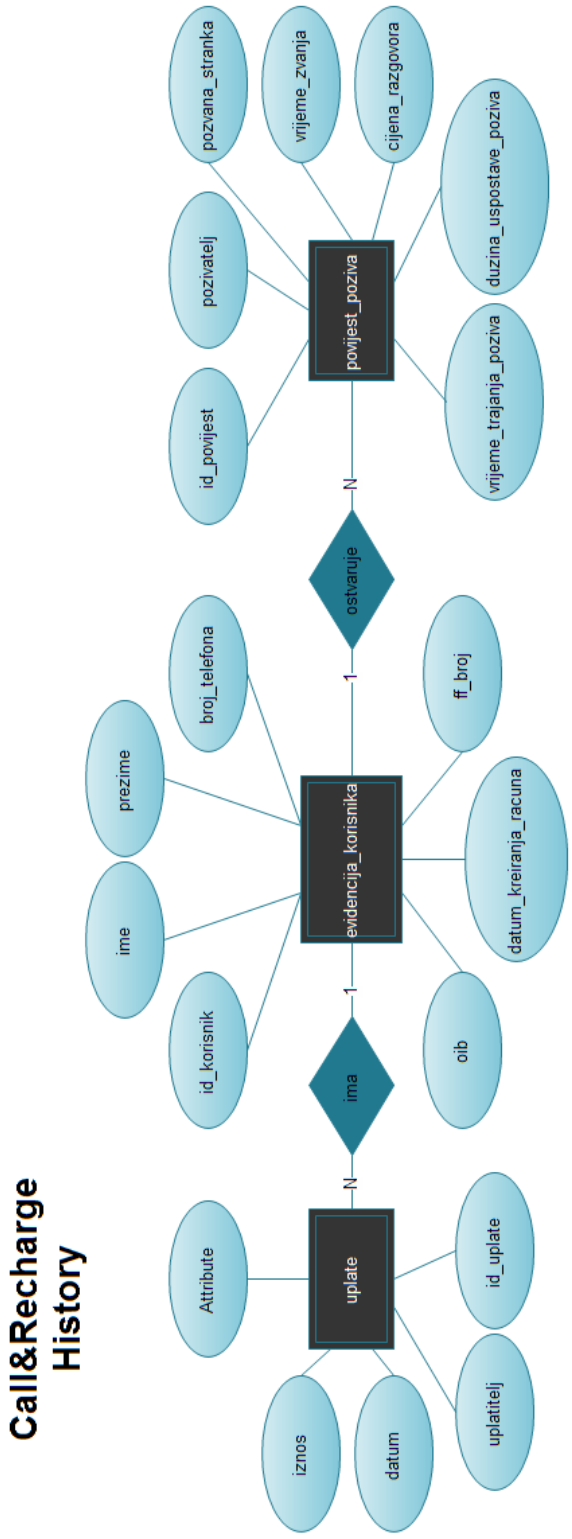
- imati uvid u povijest poziva (tko (callingparty) je koga (called party) zvao, kada, koliko dugo je trajao razgovor (dialog duration), koliko dugo je trebalo da se veza uspostavi (call setup time), cijena razgovora (call charge)) i uplata (tko je kada i koliko novaca uplatio (recharge) na svoj račun)
- temeljem prijave poslužitelju i povijesti poziva i uplata treba ažurirati evidenciju korisnika (naziv, broj telefona, datum otvaranja računa, datum zadnje uplate, stanje računa, trajanje računa (mjeri se od dana zadnje uplate ili otvaranja računa), "friends" broj, ...)

Baza podataka treba biti u što većoj mjeri normalizirana.

Zadaci:

1. Napraviti ER dijagram (entity-relationship diagram) dijagram sa svim bitnim elementima.
2. Napraviti relacijski model
3. Dati kraći tekstualni opis odabranog rješenja uz osvrt na evetualne specifičnosti, pretpostavke ili ograničenja u modelu.
4. Napraviti SQL naredbe za kreiranje baze podataka koja odgovara relacijskom modelu.
5. Napraviti SQL naredbe kojima se baza puni podacima za potrebe testiranja.
6. Napraviti primjer SQL upita za koje se očekuje da će biti najčešće upotrebljavani od strane korisnika baze podataka (prema zahtijevima u opisu zadatka), opisati ih riječima (koja je svrha upita?) i dati konkretni primjer rezultata kakav se dobije takvim upitom.

II. ER dijagram



III. Relacijski model

evidencija_korisnika				
polje	vrsta	null	zadano	poveznica
id_korisnik	int(11)	No		
ime	varchar(50)	No		
prezime	varchar(50)	No		
oib	char(11)	No		
broj_telefona	char(12)	No		
datum_otvaranja_racuna	date	No		
ff_broj	char(12)	No		

timovi				
polje	vrsta	null	zadano	poveznica
id_povijest	int(11)	No		
pozivatelj	int(11)	No		evidencija_korisnika -> id_korisnik
pozvana_stranka	int(11)	No		evidencija_korisnika -> id_korisnik
vrijeme_zvanja	timestamp	No	CURRENT_TIMESTAMP	
vrijeme_trajanja_poziva	time	No		
duzina_uspostave_poziva	time	No		
cijena_razgovora	float(10,2)	No		

utrka				
polje	vrsta	null	zadano	poveznica
id_uplata	int(11)	No		
uplatitelj	int(11)	No		evidencija_korisnika -> id_korisnik
datum	date	No		
iznos	float(10,2)	No		

IV. Opis odabranog rješenja

Baza je izrađena u MySQL-u pod imenom „cr_history“. Sastoji se od tri tablice koje udovoljavaju uvjetima postavljenim u zadatku.

Tablica „evidencija_korisnika“

Po opsegu najveća tablica. Služi za unos korisnika prema imenu, prezimenu, njegovom broju, datumu otvaranja računa, njegovom favorit broju. Kao primarni ključ stavit je id_korisnik.

Tablica „povijest_poziva“

Ova tablica služi za unos podataka o pozivima. Tako možemo unijeti podatke o pozivatelju i pozvanoj osobi, koliko je taj poziv trajao te cijeni poziva. Ona sadrži dva strana ključa: *povijest_poziva.pozivatelj* i *povijest_poziva.pozvana_stranka*. Stupac vrijeme_zvanja postavljen je timestamp tip podataka. To znači da se vrijeme samo ažurira prilikom unosa novih poziva.

Tablica „uplate“

Tablica za unos novog iznosa na račun. Sastoji se od tri stupca: uplatitelj, iznos i datum, te primarnog ključa id_uplata. Tablica ima strani ključ *uplate.uplatitelj*.

V. SQL naredbe za kreiranje baze

Kreiranje baze podataka:

```
CREATE DATABASE `cr_history` DEFAULT CHARACTER SET latin1
COLLATE latin1_swedish_ci;
USE `c&r_history`;
```

Kreiranje tablice „evidencija_korisnika“:

```
CREATE TABLE IF NOT EXISTS `evidencija_korisnika` (
  `id_korisnik` int(11) NOT NULL AUTO_INCREMENT,
  `ime` varchar(50) NOT NULL,
  `prezime` varchar(50) NOT NULL,
  `oib` char(11) NOT NULL,
  `broj_telefona` char(12) NOT NULL,
  `datum_otvaranja_racuna` date NOT NULL,
  `ff_broj` char(12) NOT NULL,
  PRIMARY KEY (`id_korisnik`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=4 ;
```

Kreiranje tablice „povijest_poziva“:

```
CREATE TABLE IF NOT EXISTS `povijest_poziva` (
  `id_povijest` int(11) NOT NULL AUTO_INCREMENT,
  `pozivatelj` int(11) NOT NULL,
  `pozvana_stranka` int(11) NOT NULL,
  `vrijeme_zvanja` timestamp NOT NULL DEFAULT
CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  `vrijeme_trajanja_poziva` time NOT NULL,
  `duzina_uspostave_poziva` time NOT NULL,
  `cijena_razgovora` float(10,2) NOT NULL,
  PRIMARY KEY (`id_povijest`),
  KEY `pozivatelj` (`pozivatelj`),
  KEY `pozvana_stranka` (`pozvana_stranka`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=4 ;
```

Kreiranje tablice „uplate“:

```
CREATE TABLE IF NOT EXISTS `uplate` (
  `id_uplata` int(11) NOT NULL AUTO_INCREMENT,
  `uplatitelj` int(11) NOT NULL,
  `datum` date NOT NULL,
  `iznos` float(10,2) NOT NULL,
  PRIMARY KEY (`id_uplata`),
  KEY `uplatitelj` (`uplatitelj`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=4 ;
```

Postavljanje ograničenja:

```
ALTER TABLE `povijest_poziva`

    ADD CONSTRAINT `povijest_poziva_ibfk_2` FOREIGN KEY
(`pozvana_stranka`) REFERENCES `evidencija_korisnika`
(`id_korisnik`) ON DELETE CASCADE ON UPDATE CASCADE,

    ADD CONSTRAINT `povijest_poziva_ibfk_1` FOREIGN KEY
(`pozivatelj`) REFERENCES `evidencija_korisnika`
(`id_korisnik`) ON DELETE CASCADE ON UPDATE CASCADE;
```

```
ALTER TABLE `uplate`

    ADD CONSTRAINT `uplate_ibfk_1` FOREIGN KEY (`uplatitelj`)
REFERENCES `evidencija_korisnika` (`id_korisnik`) ON DELETE
CASCADE ON UPDATE CASCADE;
```

VI. Punjenje baze podacima

Unos podataka u tablicu „avioni“:

```
INSERT INTO `EVIDENCIJA_KORISNIKA` (`ID_KORISNIK`, `IME`,
`PREZIME`, `OIB`, `BROJ_TELEFONA`, `DATUM_OTVARANJA_RACUNA`,
`FF_BROJ`) VALUES
(1, 'IVAN', 'GRBIC', '21474836453', '0989965211', '2011-02-
07', '0987482563'),
(2, 'TOMISLAV', 'HORVAT', '21474836453', '0995656231', '2011-
02-06', '0997825635'),
(3, 'JOSIP', 'MARIC', '25633891665', '0927785156', '2011-02-
02', '0927715635');
```

Unos podataka u tablicu „kompanije“:

```
INSERT INTO `povijest_poziva` (`id_povijest`, `pozivatelj`,
`pozvana_stranka`, `vrijeme_zvanja`,
`vrijeme_trajanja_poziva`, `duzina_uspostave_poziva`,
`cijena_razgovora`) VALUES
(1, 1, 2, '2011-02-09 20:14:18', '00:01:25', '00:00:21',
3.16),
(2, 2, 3, '2011-02-09 20:15:21', '00:06:56', '00:00:08',
8.11),
(3, 1, 3, '2011-02-09 20:16:08', '00:00:26', '00:00:17',
0.25);
```

Unos podataka u tablicu „letovi“:

```
INSERT INTO `uplate` (`id_uplata`, `uplatitelj`, `datum`,
`iznos`) VALUES
(1, 1, '2011-02-08', 100.00),
(2, 2, '2011-02-08', 25.00),
(3, 3, '2011-02-08', 200.00),
(4, 1, '2011-02-10', 500.00);
```

VII. Upiti koji se najčešće koriste

Ispis podataka iz tablice „evidencija_korisnika“:

```
SELECT * FROM evidencija_korisnika;
```

Ovom naredbom dohvaćamo sve podatke koji se nalaze u tablici „evidencija_korisnika“. Tablica ne sadrži strani ključ pa je ovo najbolji i najbrži ispis svih korisnika koji se nalaze u tablici.

Ispis podataka iz tablice „uplate“:

```
SELECT `evidencija_korisnika`.`ime`,
`evidencija_korisnika`.`prezime`,
`evidencija_korisnika`.`broj_telefona`, `uplate`.`datum`,
`uplate`.`iznos`
FROM `uplate`, `evidencija_korisnika`
WHERE evidencija_korisnika.id_korisnik=uplate.uplatitelj
```

Ovim upitom ispisat ćemo korisnika, odnosno uplatitelja i njegov iznos. Pomoću WHERE definirali smo da se prikažu stvarni nazivi uplatitelja koji je inače u tablici uplate krije iz id-a.

Ispis podatka iz tablice „povijest_poziva“:

```
SELECT `evidencija_korisnika`.`broj_telefona` AS `pozivatelj`,
(SELECT broj_telefona FROM evidencija_korisnika WHERE
povijest_poziva.pozvana_stranka=evidencija_korisnika.id_korisn
ik) AS `pozvana stranka`, `povijest_poziva`.`vrijeme_zvanja`,
`povijest_poziva`.`vrijeme_trajanja_poziva` AS `trajanje
poziva`, `povijest_poziva`.`duzina_uspostave_poziva` AS
`uspostava poziva (h:m:s)`,
`povijest_poziva`.`cijena_razgovora` AS `cijena (kn)`
FROM `povijest_poziva`
JOIN evidencija_korisnika
ON povijest_poziva.pozivatelj=evidencija_korisnika.id_korisnik
```

Ovim upitom ispisat ćemo podatke iz tablice povijest_poziva. Sličan je ispis kao i gore gdje smo se poslužili sa WHERE, ali ovdje smo dodatno morali koristiti naredbu SELCET. To je bilo nužno jer se isti podatak dohvaća samo što je pod drugim id-om pospremljeno u ovoj tablici. To je zato što imamo pozivatelja i primatelja poziva. Bilo je također nužno koristiti JOIN jer se podaci iz druge tablice ne bi ispravno prikazali. Sa AS smo prilikom ispisa malo prilagodili imena stupaca. To je privremeno i pojavljuje se samo kod ispisa.

Ispis iznosa na računu za korisnika:

```
SELECT DISTINCT `evidencija_korisnika`.`ime`,
`evidencija_korisnika`.`prezime`,
`evidencija_korisnika`.`broj_telefona`, ((SELECT DISTINCT
SUM(iznos) FROM uplate WHERE uplate.uplatitelj=1)-(SELECT
DISTINCT SUM(povijest_poziva.cijena_razgovora) FROM
povijest_poziva WHERE povijest_poziva.pozivatelj=1)) AS `iznos
na racunu`
FROM `povijest_poziva`
JOIN `evidencija_korisnika`, `uplate`
WHERE
evidencija_korisnika.id_korisnik=povijest_poziva.pozivatelj
AND evidencija_korisnika.id_korisnik=uplate.uplatitelj
AND povijest_poziva.pozivatelj=1
```

Ovaj uvjet ispisat će nam preostali iznos na računu korisnika „Ivan Grbic“ nakon svih obavljenih razgovora. Pomoću SUM naredbe, zbrojeni su svi iznosi u stupcu cijena_razgovora, a da bi pravilan ispis bio morali smo postaviti id pozivatelja. Na taj način pravilno je zbrojen stupac za jednog korisnika. Onda smo pomoću oduzimanja oduzeli sumu od stupca uplate.iznos i dobili iznos koji je preostao na računu. Dodavanjem novih stavki u povijest_poziva automatski se pomoću ovog upita može vidjeti stanje na računu. Ako hoćemo vidjeti stanje kod drugog korisnika onda moramo promijeniti vrijednost kod uplate.uplatitelj i povijest_poziva.pozivatelj.

Izmjena podatka korisnika:

```
UPDATE `cr_history`.`evidencija_korisnika` SET `ff_broj` = '09
97832452' WHERE `evidencija_korisnika`.`id_korisnik` =2
LIMIT 1
```

Korisnik je promijenio broj friends broj pa je potrebno promijeniti i podatke u bazi. To ćemo napraviti pomoću naredbe UPDATE gdje je id korisnika 2.

Brisanje korisnika :

```
DELETE FROM `cr_history`.`evidencija_korisnika` WHERE
`evidencija_korisnika`.`id_korisnik` = 3 LIMIT 1;
```

Nakon što je korisnik odlučio da više ne bude u bazi, potrebno je maknuti njegove podatke. To je napravljeno pomoću naredbe DELETE prema id-u korisnika. LIMIT znači da se samo jedan red u tablici briše.

VIII. Zaključak

Bazu smo napravili u MySQL-u (Xampp). Udovoljava većini uvjeta postavljenih zadatkom. Nastojano je da ispis bude što razumljiviji pa je svaki dio koda stavit u novi red. Tipa je InnoDB što nam omogućava postavljanje ograničenja.

Upoznali smo se kako baza radi i osnovne naredbe: insert, select, join, where, as..., te načine na koje se mogu računati matematičke funkcije u bazi. Za provjeru uvjeta koristili smo MySQL Query Browser, a ER dijagram je napravljen pomoću programa Edraw UML Diagram 5.1.