FAKULTET ELEKTROTEHNIKE
TEHNIČKA INFORMATIKA
ŠK.GOD. 2012/2013.

PROJEKTNI ZADATAK IZ PREDMETA BAZE PODATAKA

Predmetni asistent:
Dr.sci. Emir Mešković, viši asistent

Studenti:
Edna Džebić
Hasanhodžić Haris

Tuzla, 2013.

Sadržaj:

1. PROJEKTNI ZADATAK

I dio

- Oblikovati ER model baze podataka za odabrani sistem (segment realnog svijeta):
- Napisati detaljan opis odabranog sistema (segmenta realnog svijeta) prema kojem će biti oblikovan ER model baze podataka
- Nacrtati ER model, opisati entitete i veze (njihove atribute i ključeve). Sve sheme moraju zadovoljavati 3NF.
- Dobijeni ER model transformisati u relacijski model
- Napisati SQL skriptu za kreiranje baze podataka s ugrađenim opštim pravilima integriteta.
- Kreirati datoteke sa probnim podacima i napisati naredbe za punjenje baze podataka.

II dio

- Za bazu podataka iz prvog dijela projektnog zadatka kreirati dvije procedure:
- Napisati detaljan opis zadatka kojeg će implementirati procedure koje je potrebno kreirati
- Procedure treba da se obavljaju kao samostalne transakcije: transakcija započinje i završava unutar procedure
- Tipove ulaznih argumenata i varijabli koje se koriste za pohranu vrijednosti iz baze podataka treba deklarisati implicitno.
- U procedurama treba obrađivati pogreške (iznimke) i treba brinuti o tome šta će se desiti ukoliko procedura završi u pogrešci i ostavi transakciju otvorenom
- Relacijama u bazi podataka sigurno će istovremeno pristupati više korisnika te je zato u procedurama potrebno voditi računa o kontroli paralelnog pristupa određivanjem potrebnih nivoa izolacije
- Na zaključavanje prilikom čitanja iz relacija u bazi podataka čeka se najviše 5 sekundi, a na sva ostala zaključavanja ne smije se čekati

III dio

Kreirati okidače za kritične operacije u bazi podataka (INSERT, UPDATE, DELETE) u kojima će se voditi računa o ispravnosti obavljenih operacija. Okidači mogu koristiti procedure kreirane u drugom dijelu projektnog zadatka ili po potrebi kreirati nove procedure.

2. I DIO

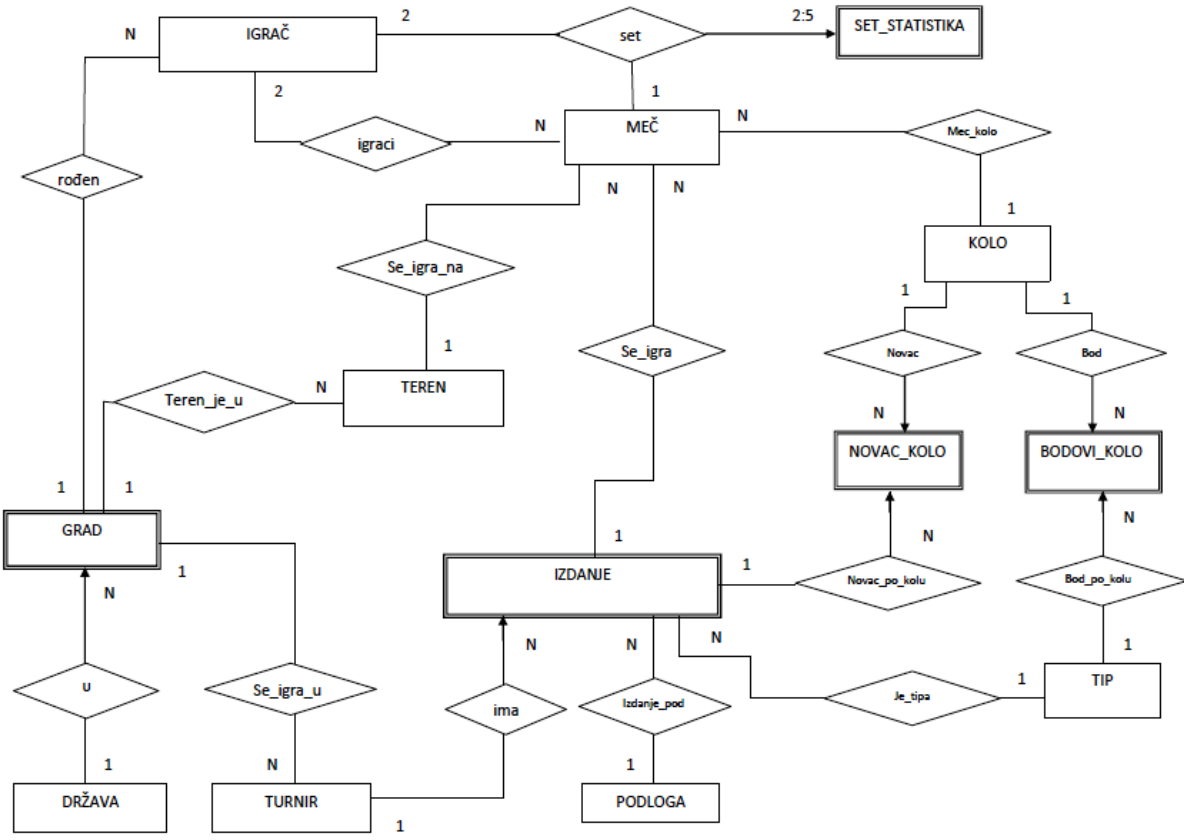
2.1. Opis sistema

U ovom projektu se vode podaci o ATP/WTA teniskim turnirima, igračima, terenima, gradovima u kojim se održavaju turniri, o odigranim mečevima i setovima na turnirima.

U bazi podataka se vode podaci o turnirima. Svaki turnir ima svoj naziv po kojem se on razlikuje od drugih (na primjer Roland Garros, ili Wimbledon, i slično) a osim imena, svaki turnir koji se igra je određenog tipa (na primjer ATP Masters 250, Grandslam...). Za turnir se dalje može znati u kojem gradu i u kojoj državi se igra taj turnir (Pariz, London...) a osim toga, potrebno je znati i neke osnovne informacije kao sto su kada taj turnir počinje, odnosno od kojeg do kojeg datuma traje, pa dalje koliko učesnika učestvuje na tom turniru, koliki je nagradni fond na turniru, na kojoj se podlozi igra, te da li je muški ili ženski turnir. Međutim, svi ovi podaci nisu određeni samo turnirom, nego su određeni i godinom u kojoj se turnir igra, jer na primjer nagradni fond na nekom turniru ne mora biti isti svake godine. Zbog toga postoje određeni podaci koji su vezani za turnir, ali i za godinu kada se taj turnir igra. Obzirom da se turniri igraju svake godine, onda zapravo za svaku godinu imamo jedno izdanje određenog turnira za tu konkretnu godinu, i podaci kao što je broj učesnika, nagradni fond, podloga na kojoj se igra, kategorija (da li je ATP ili WTA) i tip se vezuju za to izdanje turnira. Moguće je dakle da turnir također neke godine promijeni tip, pa je zbog toga i podatak kojeg je tipa turnir potrebno vezati za određeno izdanje turnira, a ne sam turnir.
Određeno izdanje turnira se igra na određenoj podlozi. Podloga može biti tvrda, travnata, šljaka... Dakle, moramo voditi i podatke o tome na kojoj se podlozi turnir igra, a osim toga, možemo uvesti i oznaku, tj skraćenicu za naziv podloge osim naziva te podloge, te nju koristiti u ispisima podataka o turniru.
Kao što vidimo, potrebno je znati u kojem gradu se igra turnir. To znači da je potrebno voditi i evidenciju o gradovima. Za grad se standardno vezuju podaci kao što je poštanski broj grada, ime grada i država u kojoj se nalazi taj grad. Ovo znači da je dalje potrebno voditi evidenciju i o državama u kojima se nalaze gradovi u kojima se igraju turniri. Za državu možemo voditi na primjer ime države, ali i skraćenicu, ili oznaku, kako se ne bi uvijek morao pisati puni naziv države ondje gdje on nije potreban.
Na turnirima igraju teniseri. Neke od njihovih karakteristika su ime i prezime, datum rođenja, visina i težina. Kako imamo podatke o gradovima i državama, onda za igrače možemo imati podatke i o mjestu odnosno gradu rođenja. Osim toga, potrebno je znati koji igrač je kojeg spola, tj da li je muško ili žensko, jer na taj način možemo odrediti u kojoj kategoriji, ATP ili WTA, se igrač takmiči. Pored toga, svaki teniser ima i trenera, pa se moze poznavati također i ime i prezime trenera, ali je također poželjno znati i kojom rukom igra taj igrač, da li je ljevak ili dešnjak te od kad je počeo igrati kao profesionalac.

Na ATP i WTA listi igrači i igračice se rangiraju prema osvojenom broju bodova na turnirima, što znači da se i taj podatak treba poznavati, odnosno moći odrediti, a osim toga, takmičenjem na turnirima igrači zarađuju određeni novac, pa je dakle potrebno poznavati i taj podatak.
Broj bodova koje igrač može osvajiti na nekom turniru se mora poznavati. Broj bodova na nekom turniru zavisi od toga kojeg je tipa turnir. Dakle, potrebno je poznavati, odnosno voditi podatke i o tipovima turnira koji se igraju. Za tip se poznaje naziv tipa, da li je Grandslam, ATP Masters 1000, neki od ITF Futures serije i slično.
Osim sto broj bodova ovisi od tipa turnira, broj bodova zavisi i od kola turnira, odnosno, broj bodova koje igrač može dobiti nije isti ako igrač igra prvu ili drugu rundu, ili finale. Očito će biti potrebno na neki način evidentirati i kolo koje se igra, a za kolo možemo imati skraćenicu i puni naziv. Broj kola zavisi od turnira, odnosno, nije isti za svaki turnir.
Bodovanje igrača, odnosno rangiranje igrača na ATP ili WTA listi, zavisi dakle od bodova koje igrač osvaja na turnirima koje igra. Međutim, ne uzimaju se u obzir svi turniri, nego samo najboljih osamnaest rezultata jednog igrača iz protekle godine. To se treba uzeti u obzir prilikom određivanja bodova za svakog igrača.
Novac kojeg igrač može zaraditi na nekom turniru ne zavisi samo od turnira, nego i godine kada se taj turnir igra, odnosno, zavisi od izdanja turnira. Osim toga, novac kojeg igrač može zaraditi zavisi, kao i broj bodova, od kola turnira, jer ne donose isto para polufinale i finale, kao ni bilo koje drugo kolo. Dakle, potrebno je na neki način evidentirati koliko novca igrač, na određenom izdanju turnira, i to za određeno kolo, može dobiti.
Na nekom turniru, odnosno na određenom izdanju tog turnira, igrači igraju mečeve. Dakle, potrebno je voditi i podatke o tome. Međutim, neki meč se mora moći evidentirati bez obzira što će se tek odigrati, a svakako se mora moći evidentirati nakon što je odigran. Znači, potrebno je znati datum kad će se meč odigrati, ili kad je odigran, u koliko je sati odigran (ili će se tek odigrati). Osim toga, interesantan podatak je često i trajanje meča pa se i taj podatak može koristiti.
Svaki meč ima najmanje dva seta, a najviše pet setova. U toku seta za igrače se vode podaci koji određuju statistiku seta, a na osnovu toga se dalje može odrediti statistika za meč. Neki od podataka koji se često vode su broj asova i duplih servis grešaka, brzine prvog i drugog servisa, te precenat osvojenih bodova na te servise...
Najbitniji podatak je svakako rezultat seta, koji se odeđuje na osnovu broja osvojenih game-ova u tom setu za svakog igrača. Na osnovu rezultata seta može se odrediti rezultat meča.
Na svakom meču učestvuju igrači, odnosno, moramo znati koji su igrači odigrali određeni meč. Ako će se meč tek odigrati, možda se neće znati koji igrači igraju, kao na primjer kada se na nekom turniru igra prvo kolo, pa se ne zna koji igrači će igrati drugo kolo, a mi bi smo željeli napraviti raspored mečeva za to drugo kolo. Tada nećemo moći sve podatke potrebne za meč unijeti, ali ti se podaci naravno mogu kasnije dodati.
Svaki meč se igra na određenom terenu, pa je potrebno znati i gdje se meč odigrao, ili će se tek odigrati. Kod opisa terena moramo zbati naziv terena, na primjer RodLaver Arena, ili Court Philippe Chatrier. Svaki teren ima neke određene karakteristike, kao što su broj mjesta na terenu, ili to da li teren ima krov, pa dalje da li ima Hawkeye sistem i slicno. Naravno, i za teren se može voditi podatak u kojem gradu, odnosno državi, se nalazi.

2.2. Shema er modela



2.3. Entiteti i veze

- *ENTITETI*

Na slici je prikazana shema ER modela koja sadrži slijedeće entitete:
grad, država, igrač, teren, turnir, izdanje, tip, bodovi_kolo, novac_kolo, meč, set_statistika, podloga, kolo.
Jaki entiteti su država, igrač, meč, teren, turnir, tip, podloga i kolo, dok su slabi bodovi_kolo, novac_kolo, set_statistika, izdanje i grad.

U nastavku su prikazani sheme svih entiteta, a podvučeni su ključevi svakog entiteta:

DRŽAVA: oznDrzava, nazivDrzava
GRAD: pbrGrad, oznDrzava, grad
IGRAČ: sifIgrac, imeIgrac, prezIgrac, datRod, spol ,visina, tezina, plays, turned-pro, trener, brojBodova, zarada
TEREN: sifTeren, nazivTeren, brojMjesta, krov, Hawkeye,
TURNIR: sifTurnir, Turnir
IZDANJE: sifTurnir, datPoc, datKraj, kategorija, broj_ucesnika, nagradniFond,

TIP: sifTip, nazivTip, brojBodovaUkupno
BODOVI_KOLO: sifTip, ozn_kolo, bodoviKolo
NOVAC_KOLO: sifTurnir, datPoc, ozn_kolo, novacKolo
MEČ: sifMec, datMec, vrijemeMec, trajanje
SET_STATISTIKA: sifMec, brojSet, sifIgrac, aces, doubleFaults, FstServiceIn, FstServicePts, SndServicePts, netPtsWon,breakPtsWon, winners, unForcedErrors, gamesWon, fastestServe, FstServeAverage, SndServeAverage
PODLOGA: ozn_podloga, podloga
KOLO: ozn_kolo, kolo

U ovom ER modelu imamo 13 entiteta, 8 jakih i 5 slabih entiteta. U nastavku su data značenja atributa svih entiteta.

Entitet *država* ima atribute *oznDrzava* i *nazivDrzava*. Atribut *oznDrzava* označava skraćenicu za puni naziv države, dok atribut *nazivDrzava* predstavlja puni naziv države.

Entitet *grad* ima atribute *pbrGrad* i *oznDrzava* i *grad.* Atribut *pbrGrad* je zapravo poštanski broj grada, atribut *grad* predstavlja ime grada, a atribut *oznDrzava* predstavlja skraćenicu za ime države, odnosno oznaku države (npr USA).

Entitet *igrač* ima slijedeće atribute: *sifIgrac, imeIgrac, prezIgrac, datRod, spol, visina, tezina, plays, turned-pro,trener, brojBodova* i *zarada*. Atribut *sifIgrac* predstavlja šifru igrača, a atributi *imeIgrac* i *prezIgrac* predstavljaju ime i prezime igrača. Atribut *datRod* i *spol* su datum rođenja i spol igrača, dok atributi *visina* i *težina* predstavljaju visinu igrača izraženu u centimetrima i težinu igrača izraženu u kilogramima. Atribut *plays* nam govori kojom rukom igrač igra, atribut *turned-pro* nam govori godinu od koje je igrač počeo igrati kao profesionalni teniser. Atributi *brojBodova* i *zarada* nam govore koliko je bodova, odnosno novca, igrač zaradio. Atribut *trener* je zapravo ime i prezime trenera koji trenutno trenira igrača.

Entitet *teren* ima slijedeće atribute: *sifTeren, nazivTeren, brojMjesta, krov, Hawkeye.* Atribut *sifTeren* je sifra terena, a atribut *nazivTeren* je naziv terena. Atributi *brojMjesta, krov* i *Hawkeye* nam govore koliko mjesta na terenu ima, te da li teren ima krov i Hawkeye sistem.

Entitet *turnir* ima atribute *sifTurnir* i *Turnir.* Atribut *Turnir* predstavlja naziv turnira a atribut *sifTurnir* predstavlja cjelobrojnu šifru turnira koja je jedinstvena za svaki turnir.

Entitet *izdanje* ima slijedeće atribute: *sifTurnir, datPoc, datKraj, kategorija, broj_ucesnika, nagradniFond.* Atribut *sifTurnir* predstavlja šifru turnira, a atributi *datPoc* i *datKraj* nam predstavljaju datum početka, odnosno datum kraja tog izdanja turnira. Atribut *broj_ucesnika* govori koliko tenisera učestvuje na tom konkretnom izdanju turnira, a atribut *nagradniFond* koliko je ukupno novca (u dolarima) odvojeno za nagrade teniserima. Atribut *kategorija* nam govori da li je to muški ili ženski turnir.

Enitet *tip* ima atribute *sifTip, nazivTip, brojBodovaUkupno*. Atribut *sifTip* je sifra pojedinog tipa, dok je atribut *nazivTip* naziv tipa turnira. Atribut *brojBodovaUkupno* nam govori koliko se bodova najviše može dobiti na određenom tipu turnira.

Enitet *kolo* ima atribute *ozn_kolo* i *kolo*. Atribut *ozn_kolo* je zapravo skraćenica za puni naziv kola (na primjer finale ima oznaku 'F', a polufinale oznaku 'SF'), a atribut *kolo* je puni naziv kola (na primjer polufinale ima naziv kola 'SemiFinal').

Entitet *podloga* ima atribute *ozn_podloga* i *podloga*. Atribut *ozn_podloga* je skraćenica za puni naziv podloge, a atribut *podloga* je puni naziv podloge (na primjer tvrda podloga ima oznaku 'H' a puni naziv 'Hard').

Entitet *bodovi_kolo* ima atribute *sifTip, ozn_kolo, bodoviKolo* i to je slabi entitet. Jedini neobjašnjeni atribut je *bodoviKolo* i taj atribut nam govori koliko za svako kolo na turniru određenog tipa jedan igrač može osvojiti bodova.

Entitet *novac_kolo* ima atribute *Turnir, ozn_kolo, novacKolo*. Jedini neobjašnjeni atribut je *novacKolo* i on nam govori koliko novca (u dolarima) na nekom turniru, u svakom kolu tog turnira, igrač može zaraditi.

Entitet *meč* sadrži slijedeće atribute*: sifMec*, *datMec, vrijemeMec, trajanje*. Atribut *sifMec* predstavlja cjelobrojnu šifru meča koja je jedinstvena za svaki meč. Atribut *datMec* označava datum na koji je meč odigran, ili će tek biti odigran, a atribut *vrijemeMec* označava u koliko sati je određeni meč počeo, ili će tek početi. Atribut *trajanje* je trajanje meča izraženo u minutama.

Entitet *set_statistika* ima slijedeće atribute: *sifMec, brojSet, sifIgrac, aces, doubleFaults, FstServiceIn, FstServicePts, SndServicePts, netPtsWon, breakPtsWon, winners, unforcedErrors, fastestServe, FstServiceAverage, SndServiceAverage.* Atribut *sifMec* je šifra meča na kojem je taj set odigran, atribut *brojSet* je broj koji nam govori koji je po redu taj set bio, a atribut *sifIgrac* nam govori za kojeg igrača su vezani podaci koji se vode kao statistika za taj set. Atribut *aces* govori nam broj as servisa, a atribut *doubleFaults* govori nam broj duplih servis grešaka. Atribut *FstServiceIn* govori nam koliko je igrač imao ispravnih prvih servisa , a atributi *FstServicePts* i *SndServicePts* nam govore koliko je poena igrač osvojio nakon prvog, odnosno drugog servisa. Svi ovi podaci se izražavaju u procentima. Atribut *netPtsWon* nam govori koliko je igrač osvojio bodova na mreži, dok atribut *breakPtsWon* govori koliko je break šansi igrači osvojio. Svi ovi poeni su izraženi u procentima. Atributi *winners* i *unforcedErrors* nam govore koliko su igrači imali winnera i neiznuđenih grešaka u toku meča. Atributi *fastestServe, FstServiceAverage,SndServiceAverage* nam govore koliki je bio najbrži prvi servis, te kolike su prosječne brzine prvog i drugog servisa u toku seta za nekog igrača. Sve ove brzine su izražene u kilometrima na sat. Atribut *gamesWon* nam govori koliko ja game-ova u setu određeni igrač osvojio.

- *VEZE*

Na shemi ER modela su osim entiteta prikazane i veze koje povezuje entitete. Niti jedna veza nema vlastite atribute. Shema veze sadrži dakle samo ključeve entiteta koje dotična veza povezuje, a ključevi veza definisani su pomoću ključeva entiteta koje povezuju i njihovih spojnosti. Sve spojnosti su prikazane na shemi ER modela.

U nastavku su prikazane sheme svih veza iz ER modela, a podvučeni su ključevi svake veze:

U: pbrGrad, oznDrzava
Rodjen: sifigrac, pbrGrad, oznDrzava
Teren_je_u: sifTeren, pbrGrad, oznDrzava
Se_igra_u: sifTurnir, pbrGrad, oznDrzava
Izdanje_podloga: sifTurnir, datPoc, ozn_podloga
Ima: sifTurnir, datPoc
Je_tipa: sifTurnir, datPoc, sifTip
Se_Igra: sifMec, sifTeren
Se_Igra_na: sifMec, sifTurnir, datPoc
Meč_kolo: sifMec, ozn_kolo
Igraci: sifMec, sifIgrac
Set: sifMec, brojSet, sifIgrac
Bod_po_kolu : sifTip, ozn_kolo
Bod: sifTip, oznkolo
Novac_po_kolu: sifTurnir, datPoc, ozn_kolo
Novac: sifTurnir, datPoc, ozn_kolo

Kako niti jedna veza nema vlastite atribute, svi atributi su već objašnjeni kod objašnjavanja značenja atributa kod pojedinih entiteta.

2.4. Relacijski model

Gore prikazani ER model smo transformisali u relacijski model baze podataka tako što smo ujedinili relacijske sheme sa istim ključevima, te smo na taj način dobili slijedeće relacijske sheme:

DRŽAVA: oznDrzava, nazivDrzava
GRAD: pbrGrad,oznDrzava, grad
PODLOGA: ozn_podloga, podloga
KOLO: ozn_kolo, kolo
IGRAC: sifIgrac, imeIgrac, prezIgrac, datRod, spol, visina, tezina, plays, turned-pro, brojBodova, zarada, trener, pbrGradRod, oznDrzavaRod
TEREN: sifTeren, nazivTeren, brojMjesta, krov, Hawkeye, pbrGrad, oznDrzava,
TURNIR: sifTurnir, Turnir, pbrGrad, oznDrzava
IZDANJE: sifTurnir, datPoc, datKraj, sifTip, kategorija, broj_ucesnika, nagradniFond, ozn_podloga,
TIP: sifTip, nazivTip, brojBodovaUkupno
BODOVI_KOLO: sifTip,ozn_kolo, bodoviKolo
NOVAC_KOLO: sifTurnir, datPoc, ozn_kolo, novacKolo
MEC: sifMec, datMec, vrijemeMec, ozn_kolo , sifTurnir, datPoc, sifTeren, trajanje
SET_STATISTIKA: sifMec, brojSet, sifIgrac, aces, doubleFaults, FstServiceIn, FstServicePts, SndServicePts, netPtsWon, breakPtsWon, winners, unforcedErrors, fastestServe, FstServiceAverage, SndServiceAverage, gamesWon
IGRACI: sifMec, sifIgrac

U bazi podataka smo postupkom transformacije iz ER modela u relacijski dobili 14 relacija.

Relacija *DRZAVA* sadrži naziv države i oznaku za taj naziv, odnosno skraćenicu za naziv države.
Relacija *GRAD* sadrži ime poštanski broj grada u kojem se igra turnir, te oznaku za naziv države u kojoj se taj grad nalazi.
Relacija *IGRAC* sadrži podatke o igračima. U relaciji vodimo podatke o imenu i prezimenu igrača, datumu rođenja, visini, težini, broju bodova, zaradi, kojom rukom igra, imenu i prezimenu trenera, te mjestu i državi rođenja. Atribut spol nam omogućava da odredimo da li se igrač nalazi na ATP ili WTA listi. Ako je oznaka spola 'M' onda igrač pripada ATP listi, a ako je oznaka 'F' onda pripada WTA listi.
Relacija *TEREN* sadrži podatke o terenima na kojima se igraju turniri, a to su podaci o nazivu terena, podlozi, broju mjesta na terenu, da li teren ima krov i Hawkeye sistem, te u kojem se gradu i državi se nalazi.
Relacija *TURNIR* sadrži podatke o nazivu turnira, gradu i državi u kojoj se igra turnir.
Relacija IZDANJE predstavlja jedno izdanje turnira. Ovdje vodimo podatke o datumu početka i datumu kraja tog izdanja turnira, broju učesnika, tipu turnira, nagradnom fondu turnira. Atribut kategorija govori da li su učesnici muškarci ili žene.
Relacija *TIP* sadrži podatke o tipovima turnira, tj. da li je riječ o Grandslam turniru, ATP MASTERS 1000, ITF Futures i slično.
Relacija *KOLO* sadrži oznake kola koje se ograju, kao što je na primjer prvo kolo, odnosno First Round i slično.
Relacija *PODLOGA* daje podatke koje trenutno podloge postoje, tj na kojim sve podlogama se igraju turniri.
Relacija *MEC* sadrži podatke o mečevima koji su se odigrali, ili će se tek odigrati. Ovdje evidentiramo datum meča na koje je meč odigran, ili će tek biti odigran, vrijeme, tj. u koliko sati je odigran, ili će biti odigran, i trajanje meča u minutama.
Relacija IGRACI sadrži samo šifru meča i šifru igrača, a služi nam da bi smo znali koji igrači igraju na kojem meču.
Relacija SET_STATISTIKA nam služi zapravo da vodimo statistiku za setove na meču, za svakog igrača, a to su atributi broj asova, duplih greški, ubačenih prvih servisa, osvojenih bodova nakon prvog i drugog servisa, poena dobijenih na mreži, dobijenih break šansi, broj winnera i neiznuđenih grešaka, maksimalna brzina servisa te prosječne brzine prvog i drugog servisa. Osim toga, vodimo kao podatak i broj dobijenih game-ova svakog igrača u setu, a taj nam podataka omogućava da odredimo rezultat seta, a sami tim i rezultat meča.
Relacije *BODOVI_KOLO* i *NOVAC_KOLO* nam služe kako bi smo imali podatke koliko bodova, odnosno novca igrač može dobiti u svakom kolu određenog tipa turnira, ili određenog izdanja turnira.

2.5. Init skripta i učitavanje podataka

- *INIT SKRIPTA*

```
CREATE TABLE drzava(
oznDrzava              NCHAR(20) NOT NULL,
nazivDrzava            NCHAR(100) NOT NULL,
PRIMARY KEY (oznDrzava))
LOCK MODE ROW;

CREATE TABLE grad (
pbrGrad        NCHAR(20)  NOT NULL,
grad           NCHAR(50)  NOT NULL,
oznDrzava      NCHAR(20) NOT NULL,
PRIMARY KEY(pbrGrad, oznDrzava),
FOREIGN KEY (oznDrzava) REFERENCES drzava (oznDrzava))
LOCK MODE ROW;

CREATE TABLE podloga(
ozn_podloga            NCHAR(2) NOT NULL,
podloga                NCHAR(20) NOT NULL,
PRIMARY KEY (ozn_podloga))
LOCK MODE ROW;

CREATE TABLE kolo(
ozn_kolo       CHAR(5) NOT NULL,
kolo           CHAR(20) NOT NULL,
PRIMARY KEY(ozn_kolo))
LOCK MODE ROW;

CREATE TABLE tip(
sifTip                 INTEGER NOT NULL,
nazivTip               NCHAR(30) NOT NULL,
brojBodovaUkupno       INTEGER,
PRIMARY KEY (sifTip))
LOCK MODE ROW;

CREATE TABLE bodovi_kolo(
sifTip         INTEGER NOT NULL,
ozn_kolo       CHAR(5) NOT NULL,
bodoviKolo     INTEGER NOT NULL,
PRIMARY KEY (sifTip, ozn_kolo),
FOREIGN KEY(sifTip) REFERENCES tip(sifTip) ON DELETE CASCADE,
FOREIGN KEY(ozn_kolo) REFERENCES kolo(ozn_kolo))
LOCK MODE ROW;
```

```
CREATE TABLE igrac(
sifIgrac          INTEGER NOT NULL,
imeIgrac          NCHAR(50) NOT NULL,
prezIgrac         NCHAR(50) NOT NULL,
datRod                    DATE,
spol              CHAR(1) NOT NULL CHECK (spol IN ("M","F")),
visina            INTEGER,
tezina            INTEGER,
plays             NCHAR(2) CHECK (plays IN ("RH", "LH")),
turned_pro        NCHAR(4),
trener            NCHAR(100),
brojBodova        INTEGER DEFAULT 0,
zarada            INTEGER DEFAULT 0,
pbrGradRod        NCHAR(20) DEFAULT NULL,
oznDrzavaRod              NCHAR(20) DEFAULT NULL,
PRIMARY KEY (sifIgrac),
FOREIGN KEY (pbrGradRod, oznDrzavaRod) REFERENCES grad(pbrGrad,oznDrzava))
LOCK MODE ROW;

CREATE TABLE teren(
sifTeren                  INTEGER NOT NULL,
nazivTeren                NCHAR(50),
brojMjesta                INTEGER,
krov                      CHAR(1)  CHECK (krov IN ("Y", "N")),
Hawkeye                   CHAR(1)  CHECK  (Hawkeye IN ("Y", "N")),
pbrGrad                   NCHAR(20) DEFAULT NULL,
oznDrzava                 NCHAR(20)  DEFAULT NULL,
PRIMARY KEY(sifTeren),
FOREIGN KEY(pbrGrad,oznDrzava) REFERENCES grad(pbrGrad,oznDrzava))
LOCK MODE ROW;

CREATE TABLE turnir(
sifTurnir         INTEGER NOT NULL,
Turnir            NCHAR(50) NOT NULL,
pbrGrad           NCHAR(20) DEFAULT NULL,
oznDrzava         NCHAR(20) DEFAULT NULL,
PRIMARY KEY(sifTurnir),
FOREIGN KEY(pbrGrad,oznDrzava) REFERENCES grad(pbrGrad, oznDrzava))
LOCK MODE ROW;

CREATE TABLE izdanje(
sifTurnir         INTEGER NOT NULL,
sifIzdanje        INTEGER NOT NULL,
datPoc            DATE NOT NULL,
datKraj           DATE NOT NULL,
sifTip            INTEGER DEFAULT NULL,
broj_ucesnika INTEGER,
```

```
nagradniFond INTEGER,
ozn_podloga    NCHAR(2) DEFAULT NULL,
PRIMARY KEY (sifTurnir, sifIzdanje),
FOREIGN KEY (sifTurnir) REFERENCES turnir(sifTurnir) ON DELETE CASCADE,
FOREIGN KEY(sifTip) REFERENCES tip(sifTip),
FOREIGN KEY(ozn_podloga) REFERENCES podloga(ozn_podloga))
LOCK MODE ROW;

CREATE TABLE novac_kolo(
sifTurnir          INTEGER NOT NULL,
sifIzdanje         INTEGER NOT NULL,
ozn_kolo           CHAR(5) NOT NULL,
novacKolo          INTEGER,
PRIMARY KEY (sifTurnir, sifIzdanje, ozn_kolo),
FOREIGN KEY(sifTurnir, sifIzdanje) REFERENCES izdanje(sifTurnir, sifIzdanje) ON DELETE
CASCADE,
FOREIGN KEY(ozn_kolo) REFERENCES kolo(ozn_kolo))
LOCK MODE ROW;

CREATE TABLE mec(
sifMec             INTEGER NOT NULL,
datMec             DATE NOT NULL,
vrijemeMec         CHAR(5) DEFAULT  NULL CHECK (vrijemeMec MATCHES '[0-2][0-9]:[0-5][0-9]'),
sifTurnir          INTEGER DEFAULT NULL,
sifIzdanje         INTEGER DEFAULT NULL,
ozn_kolo           CHAR(5) DEFAULT NULL,
sifTeren           INTEGER DEFAULT NULL,
trajanje           INTEGER,
PRIMARY KEY (sifMec),
FOREIGN KEY (sifTurnir,sifIzdanje) REFERENCES izdanje(sifTurnir,sifIzdanje),
FOREIGN KEY (sifTeren) REFERENCES teren(sifTeren),
FOREIGN KEY (ozn_kolo) REFERENCES kolo(ozn_kolo))
LOCK MODE ROW;

CREATE TABLE igraci(
sifMec             INTEGER NOT NULL,
sifIgrac                   INTEGER NOT NULL,
PRIMARY KEY (sifMec, sifIgrac),
FOREIGN KEY (sifMec) REFERENCES mec(sifMec),
FOREIGN KEY (sifIgrac) REFERENCES igrac(sifIgrac))
LOCK MODE ROW;

CREATE TABLE set_statistika(
sifMec                     INTEGER NOT NULL,
brojSet                    INTEGER NOT NULL,
sifIgrac                   INTEGER NOT NULL,
aces                       INTEGER,
```

```
doubleFaults          INTEGER,
FstServiceIn          INTEGER,
FstServicePts         INTEGER,
SndServicePts         INTEGER,
netPtsWon             INTEGER,
breakPtsWon           INTEGER,
winners               INTEGER,
unForcedErrors        INTEGER,
fastestServe          INTEGER,
FstServiceAverage     INTEGER,
SndServiceAverage     INTEGER,
gamesWon              INTEGER,
PRIMARY KEY(sifMec, brojSet, sifIgrac),
FOREIGN KEY(sifMec) REFERENCES mec(sifMec) ON DELETE CASCADE,
FOREIGN KEY(sifIgrac) REFERENCES igrac(sifIgrac) ON DELETE CASCADE)
LOCK MODE ROW;
```

- *UČITAVANJE PODATAKA*

```
LOAD FROM "C:\Users\pc\Desktop\PROJEKAT_HARIS_EDNA\datoteke\drzava.unl"
DELIMITER "#" INSERT INTO drzava;
LOAD FROM "C:\Users\pc\Desktop\PROJEKAT_HARIS_EDNA\datoteke\grad.unl"
DELIMITER "#" INSERT INTO grad;
LOAD FROM "C:\Users\pc\Desktop\PROJEKAT_HARIS_EDNA\datoteke\podloga.unl"
DELIMITER "#" INSERT INTO podloga;
LOAD FROM "C:\Users\pc\Desktop\PROJEKAT_HARIS_EDNA\datoteke\kolo.unl"
DELIMITER "#" INSERT INTO kolo;
LOAD FROM "C:\Users\pc\Desktop\PROJEKAT_HARIS_EDNA\datoteke\tip.unl"
DELIMITER "#" INSERT INTO tip;
LOAD FROM "C:\Users\pc\Desktop\PROJEKAT_HARIS_EDNA\datoteke\bodovi_kolo.unl"
DELIMITER "#" INSERT INTO bodovi_kolo;
LOAD FROM "C:\Users\pc\Desktop\PROJEKAT_HARIS_EDNA\datoteke\igrac.unl"
DELIMITER "#" INSERT INTO igrac;
LOAD FROM "C:\Users\pc\Desktop\PROJEKAT_HARIS_EDNA\datoteke\teren.unl"
DELIMITER "#" INSERT INTO teren;
LOAD FROM "C:\Users\pc\Desktop\PROJEKAT_HARIS_EDNA\datoteke\turnir.unl"
DELIMITER "#" INSERT INTO turnir;
LOAD FROM "C:\Users\pc\Desktop\PROJEKAT_HARIS_EDNA\datoteke\izdanje.unl"
DELIMITER "#" INSERT INTO izdanje;
LOAD FROM "C:\Users\pc\Desktop\PROJEKAT_HARIS_EDNA\datoteke\novac_kolo.unl"
DELIMITER "#" INSERT INTO novac_kolo;
LOAD FROM "C:\Users\pc\Desktop\PROJEKAT_HARIS_EDNA\datoteke\mec.unl"
DELIMITER "#" INSERT INTO mec;
LOAD FROM "C:\Users\pc\Desktop\PROJEKAT_HARIS_EDNA\datoteke\igrac.unl"
DELIMITER "#" INSERT INTO igraci;
LOAD FROM "C:\Users\pc\Desktop\PROJEKAT_HARIS_EDNA\datoteke\set.unl"
DELIMITER "#" INSERT INTO set;
```

3. II DIO

U ovom projektu smo napravili dvije procedure, jednu proceduru koja računa broj osvojenih bodova za svakog igrača, a drugu proceduru koja računa prosjek određenih atributa za igrače i rezultat svih odigranih mečeva.

3.1. Prva procedura

Prva procedura je procedura za računanje bodova za svakog igrača u toku jedne godine.

Princip rada je slijedeći:

Najprije moramo odrediti na kojim je sve turnirima u toku jedne godine neki igrač učestvovao jer je na njima i dobio određeni broj bodova, zavisno od toga do kojeg kola je stigao. To možemo tako što najprije iz relacije *igrači* nađemo igrače koji su igrali na mečevima u toku prethodne godine, a nakon toga, preko relacije *meč* odredimo na kojem izdanju turnira su ti mečevi odigrani. Moramo znati kojeg je tipa to izdanje turnira jer je broj bodova određen tipom turnira koji se igra, te kolom u kojem se meč igra. Broj bodova koje je maksimalno taj igrač mogao dobiti se može pronaći u relaciji *bodovi_kolo*, jer se u njoj nalazi broj bodova po kolu za svaki tip turnira.
Preko šifre meča iz tabele *igrači* i šifre meča iz relacije *meč* vršimo spajanje te dvije relacije, a nakon toga, preko šifre turnira i šifre izdanja iz relacije *meč* vršimo spajanje te relacije sa relacijom *izdanje*. To moramo jer moramo da znamo tip tog izdanja, a tip nam treba kako bi smo odredili koje podatke ćemo koristiti iz relacije *bodovi_kolo*. Preko šifre tipa iz relacije *izdanje* vršimo spajanje sa relacijom *bodovi_kolo*, koja se na osnovu oznake kola iz relacije *meč* spaja sa tom relacijom, i onda nađemo koliko je maksimalno bodova mogao igrač dobiti.
Na taj način za svakog igrača odredimo koliko je maksimalno mogao osvojiti bodova na određenom izdanju turnira te taj rezultat spremimo u privremenu relaciju *trez* kako bi smo mogli to kasnije koristiti.

Slijedeće što je potrebno je da se odredi ko je pobijedio na kojem meču, jer smo u prethodnom koraku odredili samo koliko je maksimalno bodova igrač mogao osvojiti u tom kolu na tom meču, samo ako je došao do toga kola, a ne i ako je pobijedio u tom meču i prošao u slijedeće kolo. Potrebno je znati ko je pobijedio kako bi se odredilo koliko će taj pobjednik bodova dobiti za to kolo, odnosno za taj meč. To možemo preko relacije *set_statistika* jer u njoj imamo atribut *gamesWon* koji nam govori koliko je game-ova koji igrač u kojem setu dobio, tako da na osnovu toga možemo odrediti rezultat svakog seta, a samim tim i svakog meča. Relaciju *set_statistika* je potrebno imati dva puta jer se u njoj nalaze igrači koji su igrali jedan protiv drugog u tom meču, pa će se ona jednom pod jednim imenom imati za jednog igrača, a pod drugim za njegovom protivnika.
Najprije dakle spajamo relaciju *igraci* sa relacijom *set_statistika* preko atributa *sifIgrac* i *sifMec*, a onda to isto radimo ponovo samo sada za drugog igrača ta taj isti meč.

Dakle, potrebno je preimenovati relaciju *set_statistika* u na primjer *set2* te tu relaciju spojiti sa relacijom *igrači* preko šifre meča, i to tako da se za tu šifru meča, šifra igrača u tom spajanju razlikuje od šifre igrača u prvobitnom spajanju.
Onda iz te dvije relacije *set_statistika* i *set2* nađemo setove iz tog meča na kojima su igrači igrali jedan protiv drugog, na osnovu atributa *gamesWon* vidimo koji ima više osvojenih game-ova i on je pobjednik tog seta. Ovdje smo također morali još odrediti i ukupan broj odigranih setova na meču, jer onaj koji je osvojio taj zadnji set, on i jeste pobjednik meča. Zbog toga nam je trebala još jednom relacija *set_statistika*, ali opet preimenovana sada u *set3*.
Tako sada dakle odredimo pobjednike svih mečeva, a rezultat izvođenja ove SELECT naredbe spremimo u privremenu relaciju *pobjednik*.

Kada smo odredili pobjednike svih mečeva, onda trebamo odrediti koji igrači su pobijedili u finalnom meču nekog izdanja turnira, jer oni moraju dobiti onoliko bodova koliko nosi oznaka 'WF' iz relacije *bodovi_kolo*, a to nije moguće uraditi u prethodnim koracima jer nigdje u *relaciji meč* nemamo tu oznaku. To radimo slijedećom SELECT naredbom u kodu. Koristimo privremenu relaciju *pobjednik* u kojoj imamo sve pobjednike svih odigranih mečeva, te na osnovu oznake kola iz meča 'F' i pobjednika iz relacije *pobjednik* dođemo do onih igrača koji su osvojili turnir, a onda na osnovu oznake kola 'WF' iz relacije *bodovi_kolo* nađemo koliko bodova donosi osvajanje tog izdanja određenog turnira.
Rezultat ove operacije spremimo u privremenu relaciju *tpobjednik*.

Kada smo sve ovo uradili onda je potrebno promijeniti broj bodova za svakog igrača. Međutim, bodovanje se radi tako što se igraču uzima u obzir samo najboljih 18 rezultata u toku jedne godine. Zato najprije radimo UPDATE broja bodova nekog igrača u privremenoj relaciji *trez*, a onda iz te relacije prebacimo rezultat u drugu privremenu relaciju *tt*, i to tako da ih poredamo silazno po broju bodova.

Nakon toga radimo kroz dvije FOREACH petlje proceduru određivanja broja bodova koje svaki igrač treba da ima, i UPDATE broja bodova u relaciji igrač.
Vanjski FOREACH služi za kretanje po relaciji *igrač*, a unutrašnji po privremenoj relaciji *tt* jer se u njoj i nalazi broj bodova koje zapravo treba sabrati za svakog igrača. Također nam on služi da uzmemo u obzir samo 18 najboljih rezultata za svakog igrača i samo njih saberemo.
Kada smo odredili taj broj bodova za igrača, onda u vanjskom FOREACHU uradimo UPDATE atributa *brojBodova* za igraca u relaciji *igrac* na stanje određenom varijablom *bodovi.*

Kod ove procedure je dat u nastavku:

```
CREATE PROCEDURE izr_bodove()
DEFINE sIgrac,sif LIKE igrac.sifIgrac;
DEFINE bodovi LIKE igrac.brojbodova;
DEFINE tur LIKE izdanje.sifTurnir;
DEFINE izd LIKE izdanje.sifIzdanje;
DEFINE n,i,j SMALLINT;
DEFINE k CHAR(80);

ON EXCEPTION SET i, j, k
        ROLLBACK WORK;
```

```
        IF i =-701 THEN
                RAISE EXCEPTION -746, 0, k;
        ELSE
                RAISE EXCEPTION i, j, k;
        END IF
END EXCEPTION

ON EXCEPTION IN (-107, -113, -134, -143, -144, -154)
        RAISE EXCEPTION -701, 0, k;
END EXCEPTION

BEGIN WORK;
SET ISOLATION TO REPEATABLE READ;
SET LOCK MODE TO WAIT 5;

LET k='Ne moze se postaviti READ LOCK na neku od relacija
mec,izdanje,igraci,bodovi_kolo';

SELECT igraci.sifIgrac, izdanje.sifTurnir, izdanje.sifIzdanje, MAX(bodoviKolo) bodovi FROM
mec,izdanje,igraci,bodovi_kolo
        WHERE   igraci.sifMec=mec.sifMec
            AND izdanje.sifTurnir=mec.sifTurnir
             AND izdanje.sifIzdanje=mec.sifIzdanje
            AND mec.ozn_kolo=bodovi_kolo.ozn_kolo
             AND bodovi_kolo.sifTip=izdanje.sifTip
            AND izdanje.datKraj<TODAY
            AND
               izdanje.datKraj>MDY(MONTH(TODAY),DAY(TODAY),YEAR(TODAY)-1)
GROUP BY 1,2,3
INTO TEMP trez;

LET k='Ne moze se postaviti READ LOCK na neku od relacija igraci,set_statistika ';

SELECT igraci.* FROM igraci,set_statistika, set_statistika set2
WHERE   igraci.sifMec=set_statistika.sifMec
        AND igraci.sifIgrac=set_statistika.sifIgrac
        AND igraci.sifMec=set2.sifMec
        AND igraci.sifIgrac<>set2.sifIgrac
        AND set_statistika.brojSet=set2.brojSet
        AND set_statistika.brojset=(SELECT MAX(brojset) FROM set_statistika set3
                                        WHERE   set3.sifMec=igraci.sifMec
                                                AND set3.sifIgrac=igraci.sifIgrac)
        AND set_statistika.gameswon>set2.gameswon
INTO TEMP pobjednik;

LET k='Ne moze se postaviti READ LOCK na neku od relacija
mec,bodovi_kolo,izdanje,pobjednik';

SELECT pobjednik.sifIgrac,izdanje.sifTurnir, izdanje.sifIzdanje,bodoviKolo FROM
mec,bodovi_kolo,izdanje,pobjednik
        WHERE   mec.sifturnir=izdanje.sifTurnir
                AND mec.sifIzdanje=izdanje.sifIzdanje
                AND izdanje.sifTip=bodovi_kolo.sifTip
                AND mec.sifMec=pobjednik.sifMec
                AND mec.ozn_kolo='F'
```

```
                AND bodovi_kolo.ozn_kolo='WF'
                AND izdanje.datKraj<TODAY
                AND   izdanje.datKraj>MDY(MONTH(TODAY),DAY(TODAY),YEAR(TODAY)-
        1)
INTO TEMP tpobjednik;

SET LOCK MODE TO NOT WAIT;
LET k = 'Privremena relacija trez: WRITE LOCK nije odobren';
UPDATE trez SET bodovi=(SELECT bodoviKolo FROM tpobjednik
                        WHERE   trez.sifIgrac=tpobjednik.sifIgrac
                                AND trez.sifTurnir=tpobjednik.sifTurnir
                                AND trez.sifIzdanje=tpobjednik.sifIzdanje)
                WHERE EXISTS (SELECT * FROM tpobjednik
                                        WHERE trez.sifIgrac=tpobjednik.sifIgrac
                                        AND trez.sifTurnir=tpobjednik.sifTurnir
                                        AND trez.sifIzdanje=tpobjednik.sifIzdanje);
SET LOCK MODE TO WAIT 5;
SELECT * FROM trez ORDER BY bodovi DESC  INTO TEMP tt;

SET LOCK MODE TO NOT WAIT;
FOREACH kursor FOR
        SELECT igrac.sifIgrac  INTO sIgrac FROM igrac

        ON EXCEPTION IN (-107, -113, -134, -143, -144, -154)
        RAISE EXCEPTION -701, 0, k;
        END EXCEPTION

        LET n=0;
        LET bodovi=0;
                FOREACH kursor2 FOR SELECT sifIgrac,sifturnir,sifIzdanje INTO sif,tur,izd
FROM tt WHERE sifIgrac=sIgrac
                        LET n=n+1;
                        SET LOCK MODE TO WAIT 5;
                        LET bodovi=bodovi+(SELECT tt.bodovi FROM tt WHERE sif=tt.sifIgrac
AND tur=tt.sifturnir AND izd=tt.sifIzdanje);
                        SET LOCK MODE TO NOT WAIT;
                        IF n=18 THEN
                        EXIT FOREACH;
                        END IF;
                END FOREACH;
        SET LOCK MODE TO NOT WAIT;
        LET k = 'Relacija igrac: WRITE LOCK nije odobren';
        UPDATE igrac SET brojBodova = bodovi WHERE CURRENT OF kursor;
END FOREACH;

DROP TABLE trez;
DROP TABLE tpobjednik;
DROP TABLE tt;
DROP TABLE pobjednik;
COMMIT WORK;
END PROCEDURE;
```

3.2. Druga procedura

Druga procedura je procedura koja nam određuje prosjek atributa po setu za svakog igrača, i to onih atributa o kojima vodimo statistiku za svaki set.

Princip rada je slijedeći:

Najprije prvom SELECT naredbom odredimo za svakog igrača iz relacije *igrači* odredimo koliko je on u svakom setu određenog meča osvojio game-ova. To možemo tako što relaciju *igrači* spojimo sa relacijom *set_statistika* preko šifre meča i šifre igrača, a onda u relaciji *set_statistika* nađemo atribut *gamesWon.*
Ovaj rezultat spremimo u privremenu relaciju *t* kako bi smo ga mogli poslije koristiti.

Slijedeći korak je da SELECT naredbom odredimo koliko je svaki meč iz relacije iz ove privremene relacije *t* imao ukupno setova jer nam je to potrebno za daljnje određivanje pobjednika meča.
Rezultat ove SELECT naredbe smjestimo u privremenu relaciju *tt*.

Slijedeća SELECT naredba nam služi da odredimo sifru igrača koji je pobijedio, ali i rezultat meča. Za to koristimo privremene relacije *tt, t* ali i jos jednom relaciju *t* preimenovanu u relaciju *n* jer nam se u njoj za jednu šifru meča i jedan broj seta javljaju dva igrača, pa nam zbog spajanja treba da možemo razlikovati koji je igrač prvi, a koji drugi. Preko atributa *sifMec* i *brojSet* spajamo te dvije relacije *t* i *n* ali tako da nam se atribut *sifIgrac* razlikuje. Tu poredimo atribut *gamesWon* iz te dvije relacije, i koji igrač ima više dobijenig game-ova onda je on pobjednik, i to ako je u pitanju posljednji set u meče, odnosno, onaj koji pobijedi u zadnjem setu on i jeste pobjednik tog meča.
Rezultat ove SELECT naredbe smjestimo u privremenu relaciju *rez_mec*. U ovoj dakle relaciji imamo šifru meča, šifru pobjednika i rezultat meča.

Slijedećom SELECT naredbom određujemo koliko je svaki igrač iz relacije *igrac* pobijedio mečeva, i to pomoću vanjskog spajanja relacija *igrac* i *rez_mec.* Ovaj rezultat smještamo u privremenu relaciju *tpob*.

Nakon ovoga slijedi SELECT naredba kojom određujemo prosjek atributa za koje vodimo evidenciju u relaciji *set_statistika* ali i nekih drugih.
U ovom koraku određujemo slijedeće: ukupan broj as servisa igrača, prosjek broja as servisa, prosjek duplih servis greški, prosjek ubačenog prvog servisa, prosjek osvojenih bodova nakon prvog i drugog servisa, prosjek poena osvojenih na mreži, prosjek osvojenih break šansi, prosječan broj winnera i neiznuđenih grešaka, maksimalnu brzinu servisa, te prosječne brzine prvog i drugog servisa. Osim ovoga, određujemo i ukupan broj odigranih mečeva i broj dobijenih mečeva. Prosjeke određujemo pomoću agregatne funkcije *AVG*, ukupan broj as servisa pomoću agregatne funkcije *SUM*, ukupan broj odigranih mečeva pomoću agregatne funkcije *COUNT* a maksimalnu brzinu servisa pomoću agregatne funkcije *MAX.*
Kako bi smo odredili ukupan broj odigranih mečeva i ispisali broj dobijenih mečeva, onda moramo spojiti relacije *igrac* i *tpob* preko atributa *sifIgrac.*
Kako bi smo odredili sve ove prosjeke, a kako bi se za svakog igrača našao rezultat, onda vršimo vanjsko spajanje relacija *igrac* i *set_statistika,* i to preko atributa *sifIgrac.*

Rezultat ove SELECT naredbe smještamo u privremenu relaciju *prosjek*, a osim gore navedenih atributa, u njoj se nalaze još i atributi *sifIgrac, imeIgrac, prezIgrac, brojBodova* i *zarada.*

Poziv procedure vršimo na slijedeći način:

EXECUTE PROCEDURE pros();

Nakon toga, da bi smo vidjeli rezultate, SELECT naredbom iz relacije *prosjek*, pregledamo rezultat izvršenja procedure:

SELECT * FROM prosjek;

U ovoj proceduri se također određuje i rezultat meča, a taj rezultat možemo vidjeti u privremenoj relaciji *rez_mec* tako što ćemo izvršiti slijedeću SELECT naredbu:

SELECT * FROM rez_mec;

Kod ove procedure je dat u nastavku:

```
CREATE PROCEDURE pros( )
DEFINE i,j INTEGER;
DEFINE k CHAR(100);
DEFINE GLOBAL var INTEGER DEFAULT 0;

ON EXCEPTION SET i, j, k
        ROLLBACK WORK;
        IF i =-701 THEN
                RAISE EXCEPTION -746, 0, k;
        ELSE
                RAISE EXCEPTION i, j, k;
        END IF
END EXCEPTION

ON EXCEPTION IN (-107, -113, -134, -143, -144, -154)
        RAISE EXCEPTION -701, 0, k;
END EXCEPTION

BEGIN WORK;
SET ISOLATION TO REPEATABLE READ;
SET LOCK MODE TO WAIT 5;

IF var>0 THEN
DROP TABLE prosjek;
DROP TABLE rez_mec;
END IF

LET var=var+1;

LET k='Ne moze se postaviti READ LOCK na neku od relacija igraci ili set_statistika!';
SELECT  igraci.*, set_statistika .brojSet, set_statistika .gamesWon FROM
igraci,set_statistika
```

```
        WHERE  igraci.sifMec=set_statistika.sifMec AND igraci.sifIgrac=set_statistika.sifIgrac
        INTO TEMP t;  {ovdje imam sifMec, sifru igraca, broj seta i koliko je on gameova
dobio u tom setu}

LET k='NE moze se postaviti READ LOCK na privremenu relaciju t!';
SELECT  t.sifMec,MAX(brojSet) setova FROM t
        GROUP BY 1
        ORDER BY 1
        INTO TEMP tt;

LET k='NE moze se postaviti READ LOCK na privremenu relaciju tt!';
SELECT  t.sifMec,t.sifIgrac AS pobjednik, COUNT(*) || ':' || MAX(t.brojSet)-COUNT(*) rezultat
FROM t, t n
        WHERE t.sifMec=n.sifMec AND t.sifIgrac<>n.sifIgrac AND t.brojSet=n.brojSet AND
t.gameswon>n.gameswon
        GROUP BY 1,2
        HAVING MAX(t.brojSet)=(SELECT setova FROM tt WHERE t.sifMec=tt.sifMec)
        ORDER BY 1,2
        INTO TEMP  rez_mec;

LET k='Ne moze se postaviti READ LOCK na relaciju igrac, ili na privremenu relaciju
rez_mec!';
SELECT igrac.sifIgrac, COUNT(sifMec) AS WON
        FROM igrac, OUTER rez_mec
        WHERE igrac.sifIgrac = rez_mec.pobjednik
        GROUP BY 1
        ORDER BY 1
        INTO TEMP tpob;

LET k='Ne moze se postaviti READ LOCK na privremenu relaciju tpob, ili na relaciju igrac, ili
na relaciju set_statistika!';
SELECT igrac.sifIgrac,imeIgrac,prezIgrac,brojBodova,zarada, SUM(aces) ACES_NUM,
AVG(aces) ACES_AVG,  AVG(doubleFaults) doubleFaults_AVG,  AVG(FstServiceIn)
        FirstServiceIn_AVG, AVG(FstServicePts) FirstServicePts_AVG, AVG(SndServicePts)
SecondServicePts_AVG, AVG(netPtsWon) netPtsWon_AVG, AVG(breakPtsWon)
breakPtsWon_AVG, AVG(winners) winners_AVG, AVG(unforcedErrors)
unforcedErrors_AVG, AVG(fastestServe) fastestServe_AVG, AVG(FstServiceAverage)
FirstService_AVG, AVG(SndServiceAverage) SecondService_AVG, COUNT(DISTINCT
set_statistika.sifMec) AS MATCH_PLAYED, WON AS MATCH_WON  FROM tpob, igrac,
OUTER set_statistika
        WHERE igrac.sifIgrac=set_statistika.sifIgrac AND igrac.sifIgrac=tpob.sifIgrac
        GROUP BY 1,2,3,4,5,WON
        ORDER BY 1
        INTO TEMP prosjek;
COMMIT WORK;
DROP TABLE t;
DROP TABLE tt;
DROP TABLE tpob;
END PROCEDURE;
```

4. III DIO

4.1. Triger INSERT

Triger kreiran za INSERT naredbu nad relacijom *igraci* je triger koji se aktivira kada se insertuje svaki red u tu relaciju. Definisan je tako da se nakon inserta novog reda u relaciju *igraci* pozove procedura *povecaj_zaradu()* , a kreiran je na slijedeći način:

```
CREATE TRIGGER up_novac INSERT ON igraci REFERENCING NEW as new FOR EACH
    ROW    (EXECUTE PROCEDURE povecaj_zaradu(new.sifMec,new.sifIgrac));
```

Testiranje se može izvršiti slijedećim naredbama:

```
INSERT INTO igraci VALUES(7,8);

INSERT INTO igraci VALUES(7,7);

INSERT INTO igraci VALUES(9,4);

INSERT INTO igraci VALUES(9,13);

INSERT INTO igraci VALUES(10,4);

INSERT INTO igraci VALUES(10,8);
```

Stanje atributa *zarada* prije ovih INSERT naredbi za ove igrače, kao i stanje nakon INSERT naredbi, je prikazano u slijedećoj tabeli:

sifIgrac	zarada prije	zarada poslije
4	24855621	26070621
7	17049089	17549089
8	45686497	48116497
13	76014777	76514777

Ovo se može testirati slijedećom SELECT naredbom:

```
SELECT sifIgrac, zarada FROM igrac WHERE sifIgrac=4 OR sifIgrac=7 OR sifIgrac=8 OR
sifIgrac=13;
```

Ovu naredbu je potrebno izvršiti prije i poslije obavljanja INSERT naredbi.

4.1.1. Procedura

Kod trigera za INSERT naredbu poziva se procedura *povecaj_zaradu(),* a princip rada te procedure je slijedeći:

Najprije određujemo pobjednika na svakom meču. Potrebno je znati ko je pobijedio kako bi se odredilo koliko za to kolo, odnosno za taj meč, igrač dobija novca. To možemo preko relacije *set_statistika* jer u njoj imamo atribut *gamesWon* koji nam govori koliko je game-ova koji igrač u kojem setu dobio, tako da na osnovu toga možemo odrediti rezultat svakog seta, a samim tim i svakog meča. Relaciju *set_statistika* je potrebno imati dva puta jer se u njoj nalaze igrači koji su igrali jedan protiv drugog u tom meču, pa će se ona jednom pod jednim imenom imati za jednog igrača, a pod drugim za njegovom protivnika.
Najprije dakle spajamo relaciju *igraci* sa relacijom *set_statistika* preko atributa *sifIgrac* i *sifMec*, a onda to isto radimo ponovo samo sada za drugog igrača ta taj isti meč. Dakle, potrebno je preimenovati relaciju *set_statistika* u na primjer *set2* te tu relaciju spojiti sa relacijom *igrači* preko šifre meča, i to tako da se za tu šifru meča, šifra igrača u tom spajanju razlikuje od šifre igrača u prvobitnom spajanju.
Onda iz te dvije relacije *set_statistika* i *set2* nađemo setove iz tog meča na kojima su igrači igrali jedan protiv drugog, na osnovu atributa *gamesWon* vidimo koji ima više osvojenih game-ova i on je pobjednik tog seta. Ovdje smo također morali još odrediti i ukupan broj odigranih setova na meču, jer onaj koji je osvojio taj zadnji set, on i jeste pobjednik meča. Zbog toga nam je trebala još jednom relacija *set_statistika*, ali opet preimenovana sada u *set3*.
Tako dakle odredimo pobjednike svih mečeva, a rezultat izvođenja ove SELECT naredbe spremimo u privremenu relaciju *pobjednik*.

Procedura određivanja zarađenog novca se računa za svako kolo, međutim, ukoliko igrač prođe u naredno kolo, u njemu će dobiti novac predviđen samo za to određeno kolo, a ne taj novac dodat na dosadašnju zarađenu svotu. U narednom SELECT-u računamo koliko je novca igrač zaradio na tom turniru prije odigranog meča jer se taj iznos treba dakle oduzeti prije nego što se doda nova vrijednost.
Prvo spojimo relaciju *meč* preko atributa *sifMec* sa varijablom *novi_sifMec* . To radimo da bi odredili izdanje turnira na kojem je odigran taj meč. Nakon toga spojimo relaciju *M* sa *mec* preko atributa *sifTurnir* i *siIizdanje* da bi uzeli sve mečeve sa tog izdanja za igraca *novi_sifIgrac* osim meča koji je trenutno ubačen. Zatim relaciju *igraci* spajamo sa *M* i uzimamo sve mečeve na kojima je igrao igrač *novi_sifIgrac*. Spojimo privremenu relaciju *pobjednik* sa relacijom *M* preko *sifMec* kako bi smo znali pobjednike tih mečeva. Spojimo relaciju *meč* sa relacijom *novac_kolo* preko atributa *sifturnir* i *sifizdanje* i ako je *novi_sifIgrac* pobjednik i u finalu je, tada se atribut *ozn_kolo* iz relacije *novac_kolo* spaja sa 'WF', ili ako nije pobjednik ili nije u finalu tada se *ozn_kolo* spaja sa *m.ozn_kolo*. Na kraju se uzima maksimalna zarada na tim mečevima za tog igrača na tom turniru, te tu vrijednost smjestimo u varijablu *novac*.

IF uslov nam provjerava da li postoji pobjednik u tom novom meču jer može se desiti da zbog toga što nismo unijeli u relaciju *igrači* drugog igrača sa tog meča, ne bude tog meča u relaciji *pobjednik*.

Slijedeći SELECT nam služi da bi smo izračunali koliko novca igrač dobija zbog toga sto je odigrao meč koji unosimo i taj podatak smještamo u varijablu *osvojeno*.

Iz relacije *meč* izdvojimo samo meč koji ima sifru *novi_sifMec*. To radimo kako bi smo dobili atribute *sifTurnir* i *sifIzdanje*, i *ozn_kolo* za taj meč. Zatim spajamo relaciju *mec* sa relacijom *pobjednik* preko atributa *sifMec*. Spojimo relaciju *mec* sa relacijom *novac_kolo* preko atributa *sifTurnir*, *sifIzdanje* i ovisno od toga ako je *novi_sifIgrac* pobijedio u finalu *ozn_kolo* spojimo sa 'WF', a ako nije pobijedio, ili pak nije finale, tada spojimo *novac_kolo.ozn_kolo* sa *mec.ozn_kolo*.

Ako se igrač sa šifrom *novi_sifigrac* ne nalazi u relaciji *pobjednik,* onda on zapravo nije pobijedio niti na jednom meču. Onda samo iz relacije *mec* uzmemo meč sa šifrom *novi_sifMec*, nakon toga preko atributa *sifTurnir i sifIzdanje* i *ozn_kolo* spojimo relacije *mec* i *novac_kolo*, i saznamo vrijednost atributa *novacKolo*, te taj podatak ubacimo u varijablu *osvojeno.*

Procedura UPDATE atributa *zarada* u relaciji *sifIgrac* za igrača čija je šifra *novi_sifIgrac* je da se na zaradu doda *osvojeno* ako igrač nije prošao u naredno kolo, a ako jeste, onda mu se najprije treba oduzeti koliko je do tad *novca* zaradio, do meča kojeg smo upravo ubacili, pa tek onda dodati onoliko novca koliko mu meč donosi meč kojem pripada n-torka koju ubacujemo u relaciju *igraci*.

Kod ove procedure je dat u nastavku:

```
CREATE PROCEDURE povecaj_zaradu (novi_sifMec LIKE igraci.sifMec, novi_sifIgrac LIKE
igrac.sifIgrac)
DEFINE novac,osvojeno LIKE igrac.zarada;
DEFINE i,j  SMALLINT;
DEFINE k CHAR(100);

ON EXCEPTION SET i, j, k

                RAISE EXCEPTION i, j, k;

END EXCEPTION

SELECT igraci.* FROM igraci,set_statistika, set_statistika set2
WHERE   igraci.sifMec=set_statistika.sifMec
        AND igraci.sifIgrac=set_statistika.sifIgrac
        AND igraci.sifMec=set2.sifMec
        AND igraci.sifIgrac<>set2.sifIgrac
        AND set_statistika.brojSet=set2.brojSet
        AND set_statistika.brojset=(SELECT MAX(brojset) FROM set_statistika set3
                                    WHERE   set3.sifMec=igraci.sifMec
                                        AND set3.sifIgrac=igraci.sifIgrac)
        AND set_statistika.gameswon>set2.gameswon
INTO TEMP pobjednik;

SELECT MAX(novacKolo) INTO novac FROM novac_kolo,mec,igraci,mec m, pobjednik
WHERE   mec.sifMec=novi_sifMec
        AND mec.sifTurnir=m.sifTurnir
        AND mec.sifIzdanje=m.sifIzdanje
        AND igraci.sifMec=m.sifMec
        AND m.sifMec<>novi_sifMec
        AND igraci.sifIgrac=novi_sifIgrac
        AND pobjednik.sifMec=m.sifMec
```

```
        AND mec.sifTurnir=novac_kolo.sifTurnir
        AND mec.sifIzdanje=novac_kolo.sifIzdanje
        AND (
                (
                m.ozn_kolo=novac_kolo.ozn_kolo AND
                        (
                                (
                                        pobjednik.sifIgrac=novi_sifIgrac
                                        AND m.ozn_kolo <> "F"
                                )
                                OR pobjednik.sifIgrac<>novi_sifIgrac
                        )
                )OR
                (
                pobjednik.sifIgrac=novi_sifIgrac
                AND m.ozn_kolo = "F"
                AND novac_kolo.ozn_kolo="WF"
                )
          );

IF EXISTS (SELECT * FROM pobjednik WHERE pobjednik.sifMec=novi_sifMec) THEN

        SELECT novacKolo INTO osvojeno FROM novac_kolo,mec, pobjednik
                WHERE   mec.sifMec=novi_sifMec
                AND pobjednik.sifMec=mec.sifMec
                AND novac_kolo.sifturnir=mec.sifturnir
                AND novac_kolo.sifIzdanje=mec.sifIzdanje
AND(
                (
                        mec.ozn_kolo="F"
                        AND novi_sifIgrac=pobjednik.sifIgrac
                        AND novac_kolo.ozn_kolo="WF"
                )OR
                (
                        novac_kolo.ozn_kolo=mec.ozn_kolo
                        AND mec.ozn_kolo="F"
                        AND(
                                novi_sifIgrac<>pobjednik.sifIgrac
                                OR pobjednik.sifIgrac IS NULL
                        )
                )OR
                (
                        novac_kolo.ozn_kolo=mec.ozn_kolo
                        AND mec.ozn_kolo<>"F"
                )

);

ELSE

        SELECT novacKolo INTO osvojeno FROM novac_kolo,mec
                WHERE   mec.sifMec=novi_sifMec
                AND novac_kolo.sifturnir=mec.sifturnir
                AND novac_kolo.sifIzdanje=mec.sifIzdanje
                AND novac_kolo.ozn_kolo=mec.ozn_kolo ;
```

```
END IF;


IF novac IS NULL THEN
        UPDATE igrac SET zarada=zarada+osvojeno WHERE igrac.sifIgrac=novi_sifIgrac;
ELIF novac < osvojeno THEN
        UPDATE igrac SET zarada=zarada-novac+osvojeno WHERE
igrac.sifIgrac=novi_sifIgrac;
END IF;

DROP TABLE pobjednik;
END PROCEDURE;
```

4.2. Triger DELETE

Triger kreiran za DELETE naredbu nad relacijom *igraci* je triger koji se aktivira kada se briše neki red iz te relacije. Definisan je tako da se nakon brisanja reda u relaciji *igraci* pozove procedura *smanji_zaradu()* , a kreiran je na slijedeći način:

```
CREATE TRIGGER down_novac DELETE ON igraci REFERENCING OLD as stari FOR
EACH ROW  (EXECUTE PROCEDURE smanji_zaradu(stari.sifMec,stari.sifIgrac));
```

Testiranje ovog trigera se može izvršiti slijedećim naredbama:

```
DELETE FROM igraci WHERE sifMec=10 AND sifIgrac=4;

DELETE FROM igraci WHERE sifMec=10 AND sifIgrac=8;

DELETE FROM igraci WHERE sifMec=9 AND sifIgrac=4;

DELETE FROM igraci WHERE sifMec=9 AND sifIgrac=4;
```

Stanje atributa *zarada* prije ovih DELETE naredbi za ove igrače, kao i stanje nakon tih naredbi, je prikazano u slijedećoj tabeli:

sifIgrac	zarada prije	zarada poslije
4	26070621	24855621
7	17549089	17549089
8	48116497	46186497
13	76514777	76014777

Ovo se može testirati slijedećom SELECT naredbom:

```
SELECT sifIgrac, zarada FROM igrac WHERE sifIgrac=4 OR sifIgrac=8 OR sifIgrac=7 OR
sifIgrac=13;
```

Ovu naredbu je potrebno izvršiti prije i poslije obavljanja DELETE naredbi.

4.2.1. Procedura

Za triger *down_novac* koristimo proceduru *smanji_zaradu().* Ova procedura nam služi kako bi, ako je neko pogrešno unio neki podatak u relaciju *igraci*, naredbom INSERT, oduzeli novac, odnosno umanjili zaradu tog igrača, jer bi se nakon unosa te n-torke aktivirao triger za INSERT naredbu, te bi se tom igraču zarada povećala. Zbog toga ovom procedurom mi vratimo zaradu tog igrača na ispravnu vrijednost. Princip rada je slijedeći:

Najprije određujemo pobjednika na svakom meču. Potrebno je znati ko je pobijedio kako bi se odredilo koliko je za to kolo, odnosno za taj meč, igrač dobio novca. To možemo preko relacije *set_statistika* jer u njoj imamo atribut *gamesWon* koji nam govori koliko je game-ova koji igrač u kojem setu dobio, tako da na osnovu toga možemo odrediti rezultat svakog seta, a samim tim i svakog meča. Relaciju *set_statistika* je potrebno imati dva puta jer se u njoj nalaze igrači koji su igrali jedan protiv drugog u tom meču, pa će se ona jednom pod jednim imenom imati za jednog igrača, a pod drugim za njegovom protivnika.
Najprije dakle spajamo relaciju *igraci* sa relacijom *set_statistika* preko atributa *sifIgrac* i *sifMec*, a onda to isto radimo ponovo samo sada za drugog igrača ta taj isti meč. Dakle, potrebno je preimenovati relaciju *set_statistika* u na primjer *set2* te tu relaciju spojiti sa relacijom *igrači* preko šifre meča, i to tako da se za tu šifru meča, šifra igrača u tom spajanju razlikuje od šifre igrača u prvobitnom spajanju.
Onda iz te dvije relacije *set_statistika* i *set2* nađemo setove iz tog meča na kojima su igrači igrali jedan protiv drugog, na osnovu atributa *gamesWon* vidimo koji ima više osvojenih game-ova i on je pobjednik tog seta. Ovdje smo također morali još odrediti i ukupan broj odigranih setova na meču, jer onaj koji je osvojio taj zadnji set, on i jeste pobjednik meča. Zbog toga nam je trebala još jednom relacija *set_statistika*, ali opet preimenovana sada u *set3*.
Tako dakle odredimo pobjednike svih mečeva, a rezultat izvođenja ove SELECT naredbe spremimo u privremenu relaciju *pobjednici*.

Slijedeće sto određujemo je koliko je na meču kojeg želimo obrisati igrač maksimalno novca zaradio, tj koliko novca mu je to kolo donijelo. Odnosno, n-torka koju brišemo iz relacije *igraci* pripada nekom meču, a na tom meču je igrač zaradio određeni iznos, a taj novac se igraču dodao na zaradu jer se nakon INSERT naredbe pokrenuo triger *up_novac*, odnosno, izvršila procedura *povecaj_zaradu*, pa sada prilikom brisanja igraču treba od zarade oduzeti taj iznos koji mu se dodao.

Prvo spojimo relaciju ***meč*** preko atributa *sifMec* sa varijablom *stari_sifMec* . To radimo da bi odredili izdanje turnira na kojem je odigran taj meč. Nakon toga spojimo relaciju *M* sa *mec* preko atributa *sifTurnir* i *siIizdanje* da bi uzeli sve mečeve sa tog izdanja za igraca *stari_sifIgrac* osim meča koji se trenutno briše. Zatim relaciju *igraci* spajamo sa relacijom *M* i uzimamo sve mečeve na kojima je igrao igrač *stari_sifIgrac*. Spojimo privremenu relaciju *pobjednici* sa relacijom *M* preko *sifMec* kako bi smo znali pobjednike tih mečeva. Spojimo relaciju ***meč*** sa relacijom *novac_kolo* preko atributa *sifturnir* i *sifizdanje* i ako je *novi_sifIgrac* pobjednik i u finalu je, tada se atribut *ozn_kolo* iz relacije *novac_kolo* spaja sa 'WF', ili ako nije pobjednik ili nije u finalu tada se *ozn_kolo* spaja sa *m.ozn_kolo*. Na kraju se uzima maksimalna zarada na tim mečevima za tog igrača na tom turniru, te tu vrijednost smjestimo u varijablu *novac*.

Nakon toga ponovo moramo odrediti pobjednike mečeva, jer se može desiti da pobjednik meča kojem pripada n-torka iz relacija *igraci* ne postoji u relaciji privremenoj *pobjednici.* To se može desiti na primjer kada smo najprije izbrisala n-torku koja je vezana za jednog igrača sa tog meča, a onda nakon toga obrišemo i drugu n-torku vezanu za taj meč. Tada u relaciji *pobjednici* tog meča neće biti, a nama treba i podatak koji je igrač pobijedio na tom meču, pa zbog toga ponovo određujemo pobjednike svih mečeva i taj rezultat spremimo i privremenu relaciju *pobjednik.*

Nakon toga, zarada igrača bi trebala da bude onolika kolika je bila prije nego što se uopšte naredbom INSERT ta n-torka unijela u relaciju *igraci*. Zbog toga slijedećom SELECT naredbom odredimo koliko je novca taj igrač osvjio na tom izdanju turnira, bez meča kojem pripada n-torka kuju brišemo. Iz relacije ***meč*** izdvojimo samo meč koji ima sifru *stari_sifMec*. To radimo kako bi smo dobili atribute *sifTurnir* i *sifIzdanje*, i *ozn_kolo* za taj meč. Zatim spajamo relaciju *mec* sa relacijom *pobjednik* preko atributa *sifMec*. Spojimo relaciju *mec* sa relacijom *novac_kolo* preko atributa *sifTurnir*, *sifIzdanje* i ovisno od toga ako je *stari_sifIgrac* pobijedio u finalu *ozn_kolo* spojimo sa 'WF', a ako nije pobijedio, ili pak nije finale, tada spojimo *novac_kolo.ozn_kolo* sa *mec.ozn_kolo*. Nama je zapravo potrebno koliko novca je do tad osvojio, pa zato uzimamo podatak *novacKolo* iz relacije *novac_kolo* i spremimo ga u varijablu *osvojeno*.

Ako se igrač sa šifrom *stari_sifigrac* ne nalazi u relaciji *pobjednik,* onda on zapravo nije pobijedio niti na jednom meču. Onda samo iz relacije *mec* uzmemo meč sa šifrom *stari_sifMec*, nakon toga preko atributa *sifTurnir i sifIzdanje* i *ozn_kolo* spojimo relacije *mec* i *novac_kolo*, i saznamo vrijednost atributa *novacKolo*, te taj podatak ubacimo u varijablu *osvojeno.*

Nakon toga određujemo kako će se vršiti UPDATE atributa *zarada* u relaciji *igrac* za igrača sa šifrom *stari_sifIgrac.* Treba od zarade oduzeti onoliko novca koliko je dobio za meč kome pripada n-torka koju brišemo, a ako postoje mečevi sa tog izdanja turnira koje je taj igrač igrao, onda mu je potrebno još dodati svotu novca koju je zaradio do meča kojeg brišemo.

Kod procedure *smanji_zaradu()* je dat u nastavku:

```
CREATE PROCEDURE smanji_zaradu (stari_sifMec LIKE igraci.sifMec, stari_sifIgrac LIKE
igrac.sifIgrac)
DEFINE novac,osvojeno LIKE igrac.zarada;
DEFINE i,j INTEGER;
DEFINE k CHAR(80);

ON EXCEPTION SET i, j, k

        RAISE EXCEPTION i, j, k;

END EXCEPTION


SELECT igraci.* FROM igraci,set_statistika, set_statistika set2
```

```
WHERE   igraci.sifMec=set_statistika.sifMec
        AND igraci.sifIgrac=set_statistika.sifIgrac
        AND igraci.sifMec=set2.sifMec
        AND igraci.sifIgrac<>set2.sifIgrac
        AND set_statistika.brojSet=set2.brojSet
        AND set_statistika.brojset=(SELECT MAX(brojset) FROM set_statistika set3
                                    WHERE   set3.sifMec=igraci.sifMec
                                            AND set3.sifIgrac=igraci.sifIgrac)
        AND set_statistika.gameswon>set2.gameswon
INTO TEMP pobjednici;

SELECT MAX(novacKolo) INTO novac FROM novac_kolo,mec,igraci,mec m, pobjednici
WHERE   mec.sifMec=stari_sifMec
        AND mec.sifTurnir=m.sifTurnir
        AND mec.sifIzdanje=m.sifIzdanje
        AND igraci.sifMec=m.sifMec
        AND m.sifMec<>stari_sifMec
        AND igraci.sifIgrac=stari_sifIgrac
        AND pobjednici.sifMec=m.sifMec
        AND mec.sifTurnir=novac_kolo.sifTurnir
        AND mec.sifIzdanje=novac_kolo.sifIzdanje
        AND (
                (
                m.ozn_kolo=novac_kolo.ozn_kolo AND
                        (
                                (
                                        pobjednici.sifIgrac=stari_sifIgrac
                                        AND m.ozn_kolo <> "F"
                                )
                                OR pobjednici.sifIgrac<>stari_sifIgrac
                        )
                )OR
                (
                pobjednici.sifIgrac=stari_sifIgrac
                AND m.ozn_kolo = "F"
                AND novac_kolo.ozn_kolo="WF"
                )
           );

SELECT  set_statistika.sifMec,set_statistika.sifIgrac FROM set_statistika, set_statistika set2
WHERE   stari_sifMec=set_statistika.sifMec
        AND stari_sifIgrac=set_statistika.sifIgrac
        AND stari_sifMec=set2.sifMec
        AND stari_sifIgrac<>set2.sifIgrac
        AND set_statistika.brojSet=set2.brojSet
        AND set_statistika.brojset=(SELECT MAX(brojset) FROM set_statistika set3
                                    WHERE   set3.sifMec=stari_sifMec
                                            AND set3.sifIgrac=stari_sifIgrac)
        AND set_statistika.gameswon>set2.gameswon
INTO TEMP pobjednik;



IF EXISTS (SELECT * FROM pobjednik WHERE pobjednik.sifMec=stari_sifMec) THEN
```

```
SELECT novacKolo INTO osvojeno FROM novac_kolo,mec, pobjednik
WHERE   mec.sifMec=stari_sifMec
        AND pobjednik.sifMec=mec.sifMec
        AND novac_kolo.sifturnir=mec.sifturnir
        AND novac_kolo.sifIzdanje=mec.sifIzdanje
AND(
        (
                (
                        mec.ozn_kolo="F"
                        AND stari_sifIgrac=pobjednik.sifIgrac
                        AND novac_kolo.ozn_kolo="WF"
                )OR
                (
                        novac_kolo.ozn_kolo=mec.ozn_kolo
                        AND mec.ozn_kolo="F"
                        AND(
                                stari_sifIgrac<>pobjednik.sifIgrac
                                OR pobjednik.sifIgrac IS NULL
                        )
                )OR
                (
                        novac_kolo.ozn_kolo=mec.ozn_kolo
                        AND mec.ozn_kolo<>"F"
        )       )

);
ELSE

SELECT novacKolo INTO osvojeno FROM novac_kolo,mec
WHERE   mec.sifMec=stari_sifMec
        AND novac_kolo.sifturnir=mec.sifturnir
        AND novac_kolo.sifIzdanje=mec.sifIzdanje
        AND novac_kolo.ozn_kolo=mec.ozn_kolo ;


END IF;


IF novac IS NULL THEN
        UPDATE igrac SET zarada=zarada-osvojeno WHERE igrac.sifIgrac=stari_sifIgrac;
ELIF novac < osvojeno THEN
        UPDATE igrac SET zarada=zarada+novac-osvojeno WHERE
igrac.sifIgrac=stari_sifIgrac;
END IF;

DROP TABLE pobjednik;
DROP TABLE pobjednici;
END PROCEDURE;
```

4.3. Triger UPDATE

Triger kreiran za UPDATE naredbu je triger koji se aktivira kada neko pokuša izvršiti tu naredbu nad relacijom *igraci.* U ovom slučaju smo mi zabranili izvršenje UPDATE naredbe, odnosno atributa *sifMec* i *sifIgrac*, a ukoliko dođe do pogreške prilikom INSERT naredbe, onda se umjesto UPDATE naredbe, izvrši najprije DELETE naredba, a onda ponovi INSERT sa ispravnim podacima.

Triger je kreiran na slijedeći način:

```
CREATE TRIGGER update_igraci UPDATE OF sifMec, sifIgrac ON igraci  BEFORE
(EXECUTE PROCEDURE ispisi_poruku ('Nedozvoljeno azuriranje sifre meca i sifre igraca u
relaciji igraci. Obrisite n-torku ukoliko ste pogrijesili kod unosa iste!'));
```

Naredbe kojima se može aktivirati su:

```
UPDATE igraci SET sifMec=2 WHERE sifMec=8 AND sifIgrac=4;

UPDATE igraci SET sifMec=2 WHERE sifMec=8 AND sifIgrac=8;
```

4.3.1. Procedura

Kod trigera za UPDATE, obzirom da je UPDATE onemogućen, procedura koja se izvršava zapravo samo služi da korisniku ispišemo poruku da se ta naredba ne može izvršiti nad relacijom *igraci*, te da umjesto naredbe UPDATE, izvrši naredbu DELETE.

Kod procedure *ispisi_poruku()* dat je u nastavku:

```
CREATE PROCEDURE ispisi_poruku (ispis NCHAR(150))
RAISE EXCEPTION -746, 0, ispis;
END PROCEDURE;
```