SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
ODJEL ZA MATEMATIKU
diplomski studij Financijska matematika i statistika

Lara Juzbašić

Škola

Seminarski rad

Kolegij: Dizajniranje i modeliranje baza podataka
Voditelj kolegija: doc. dr. sc. Slobodan Jelić
Profesorica: dr. sc. Mateja Đumić
Asistentica: Ena Pribisalić

Osijek, 2020.

1 Uvod

Zbog velikog broja učenika, nastavnika i predmeta, svakoj školi važno je imati dobro modeliranu bazu podataka. U ovom seminarskom radu opisano je modeliranje baze podataka jedne škole, točnije izrada MEV modela, relacijskog modela, kreiranje tablica i unos podataka u tablice, dohvaćanje podataka kroz upite te rad s podacima korištenjem procedura i okidača.

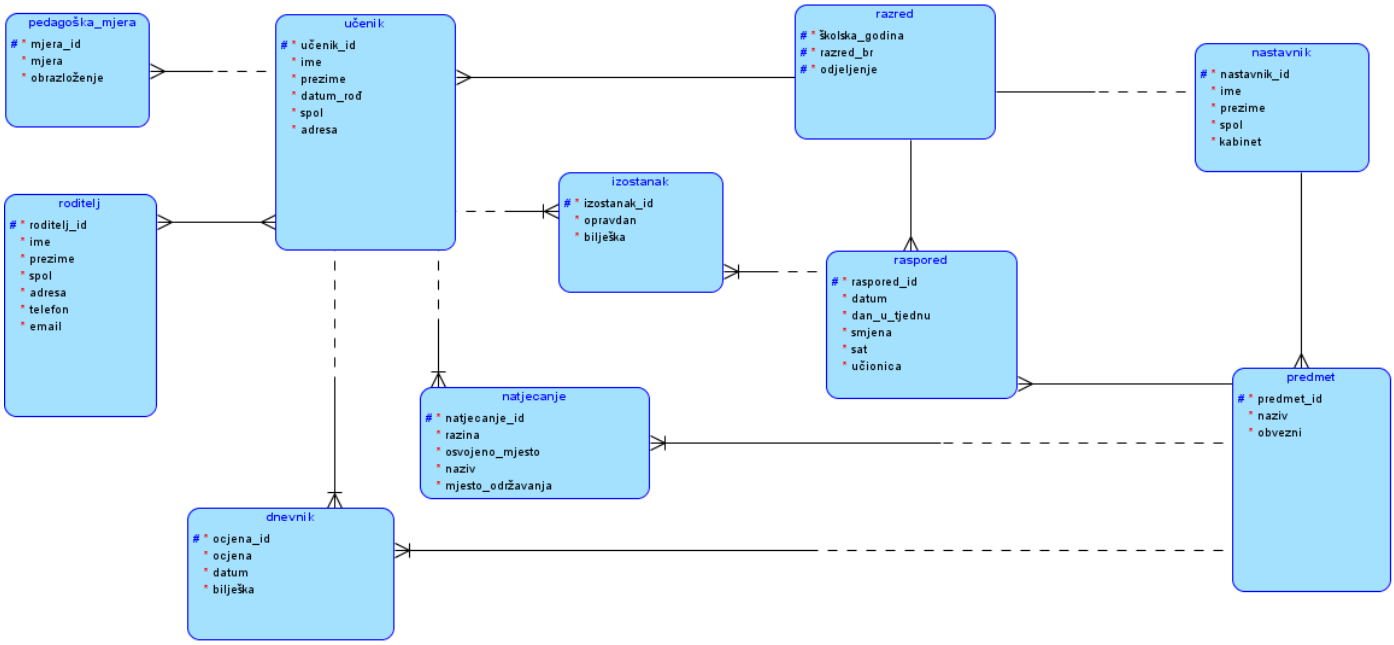
2 MEV model

MEV model se sastoji od deset entiteta s nazivima roditelj, nastavnik, predmet, razred, učenik, pedagoška_mjera, dnevnik, natjecanje, raspored i izostanak. Osim entiteta, u MEV model dodani su i atributi koji opisuju svaki entitet. Atributi sadržani u entitetima su sljedeći:

- Entitet roditelj sadrži atribut roditelj_id tipa integer koji je primarni ključ te atribute ime, prezime, spol, adresa, telefon i email tipa varchar2.
- Entitet nastavnik sadrži atribut nastavnik_id tipa integer koji je njegov primarni ključ i atribute ime, prezime, spol i kabinet tipa varchar2.
- Entitet predmet sadrži primarni ključ predmet_id tipa integer, atribut naziv tipa varchar2 i atribut obvezni tipa integer koji ima vrijednost 1 u slucaju da je predmet obvezan i vrijednost 0 u suprotnom.
- Entitet razred sadrži atribute školska_godina, razred_br i odjeljenje tipa varchar2 koji zajedno čine njegov primarni ključ.
- Entitet učenik sadrži primarni ključ učenik_id tipa integer, atribute ime, prezime, spol i adresa tipa varchar2 te atribut datum_rođ tipa date.
- Entitet pedagoška_mjera sadrži primarni ključ mjera_id tipa integer, atribut mjera tipa varchar2 u koji se upisuju vrijednosti 'opomena', 'ukor' ili 'strogi ukor' te atribut obrazloženje tipa varchar2 u koji se upisuje obrazloženje izricanja pedagoške mjere.
- Entitet dnevnik sadrži primarni ključ ocjena_id tipa integer, atribut ocjena tipa integer u koji se upisuje ocjena koju je učenik dobio, atribut datum tipa date u koji se upisuje datum upisivanja ocjene i atribut bilješka tipa varchar2 u koji se upisuje razlog upisivanja ocjene, to jest vrijednosti 'Pismeni ispit', 'Usmeni ispit', 'Domaća zadaća' i slično.
- Entitet natjecanje sadrži primarni ključ natjecanje_id tipa integer, atribut razina tipa varchar2 u koji se upisuju vrijednosti 'školska', 'županijska' ili 'državna', atribute naziv i mjesto održavanja tipa varchar2 te atribut osvojeno_mjesto tipa integer.

- Entitet raspored sadrži atribut raspored_id tipa varchar2 kao primarni ključ, atribut datum tipa date, atribute dan_u_tjednu tipa varchar2 , atribut smjena tipa varchar2 u koji se upisuje radi li se o prijepodnevnoj ili poslijepodnevnoj smjeni, atribut učionica tipa varchar2 u koji se upisuje u kojoj se učionici sat održao i atribut sat tipa integer u koji se upisuje o kojem se školskom satu radi.

- Entitet izostanak sadrži primarni ključ izostanak_id tipa integer, atribut opravdan tipa integer koji ima vrijednost 1 u slucaju da je izostanak opravdan i vrijednost 0 u suprotnom te atribut bilješka tipa varchar2 u koji se upisuje razlog izostanka sa sata.



Slika 1: MEV model

Kreirani entiteti povezani su određenim vezama. Krenimo od entiteta učenik. Kako svaki učenik mora imati jednog ili više roditelja/skrbnika i svaki roditelj koji je u bazi mora imati barem jednog učenika za dijete, veza između ta dva entiteta je veza više prema više, obvezna s obje strane. Nadalje, učenik može imati više izrečenih pedagoških mjera, a i ne mora ih imati, dok se svaka pedagoška mjera u bazi odnosi na jednog učenika. Prema tome, veza učenik – pedagoška_mjera je veza jedan prema više, opcionalna na strani učenika i obvezna na strani pedagoške mjere. Slično, učenik može imati više upisanih izostanaka, a i ne mora imati nijedan, dok se svaki upisani izostanak odnosi na jednog učenika pa je veza učenik – izostanak veza jedan prema više, opcionalna na strani učenika i obvezna na strani izostanka. Razlika od prethodne veze je to što je veza podržavajuća, što znači da entitet izostanak nasljeđuje primarni ključ entiteta učenik. Kako svaki učenik mora biti upisan u jedan razred, a razred mora imati učenike, veza razred – učenik je veza jedan prema više, obvezna s obje strane. Posljednja dva entiteta s kojima je povezan

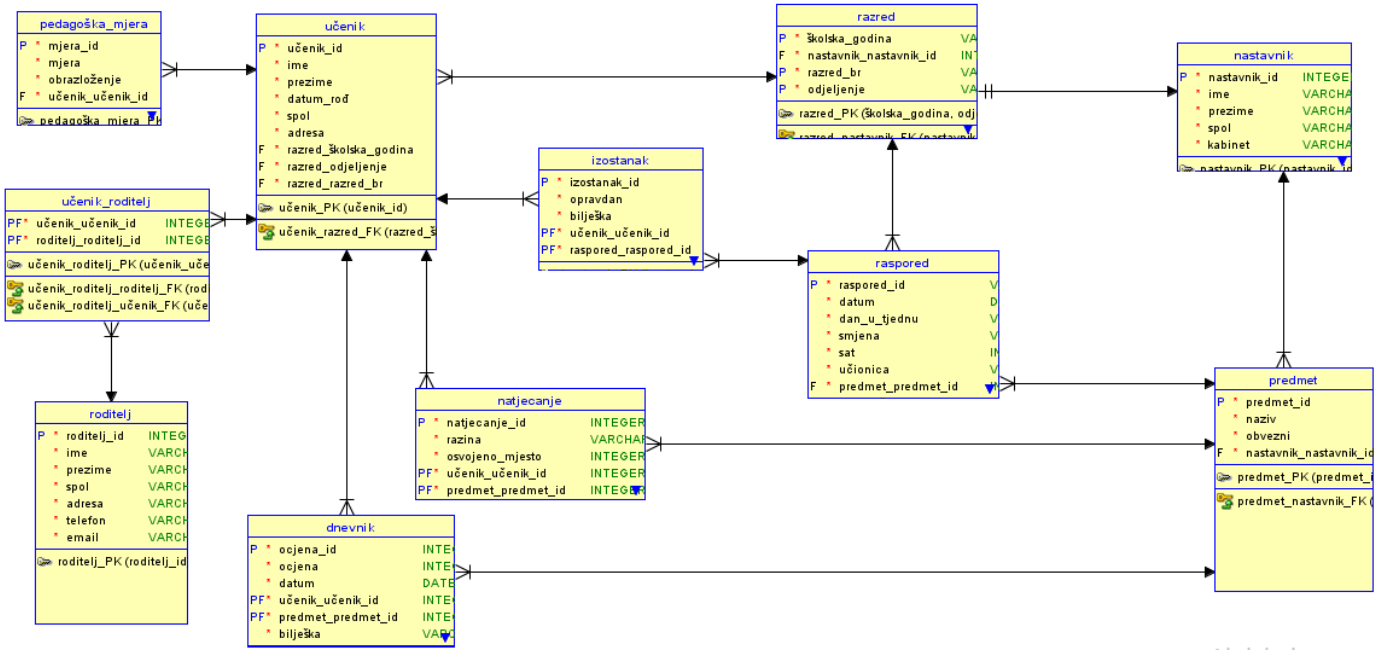
učenik su natjecanje i dnevnik. Instance entiteta dnevnik predstavljaju ocjene učenika, a instance natjecanja sudjelovanje jednog učenika na određenom natjecanju. Učenik može, a i ne mora imati više ocjena u dnevniku i natjecati se na više različitih natjecanja. S druge strane, svaka ocjena dodijeljena je točno jednom učeniku, kao i sudjelovanje na natjecanju. Dakle, veze učenik – dnevnik i učenik – natjecanje su veze jedan prema više, opcionalne na strani učenika, a obvezne na strani dnevnika, odnosno natjecanja. Ove veze su također podržive, to jest entiteti dnevnik i natjecanje nasljeđuju primarni ključ učenik_id.

Prijeđimo sada na entitet raspored. Instance rasporeda su školski sati. Bitno je znati koji se predmet predavao određeni školski sat i koji ga je razred slušao pa raspored vežemo s entitetima predmet i razred. Jednoj instanci rasporeda, školskom satu, dodijeljen je točno jedan razred i jedan predmet, dok razred i predmet moraju imati više školskih sati u rasporedu. Dakle, veze razred – raspored i predmet – raspored su veze jedan prema više, s obje strane obvezne. Raspored je također povezan s entitetom izostanak kako bi se znalo s kojeg sata u rasporedu je izostao učenik. S nekog školskog sata može, a i ne mora, izostati više učenika, dok jedan izostanak pripada točno jednom školskom satu iz rasporeda pa je veza raspored – izostanak veza jedan prema više, opcionalna na strani rasporeda, a obvezna na strani izostanka. Osim toga, veza je podržavajuća, to jest entitet izostanak nasljeđuje primarni ključ raspored_id.

Entitet predmet vežemo s entitetom nastavnik kako bi znali koji nastavnik predaje koji predmet. U ovom modelu pretpostavljeno je da predmet predaje samo jedan nastavnik u školi, ali moguće je da isti nastavnik drži više predmeta. Prema tome, veza nastavnik – predmet je veza jedan prema više, obvezna s obje strane. Predmet je povezan i s entitetima dnevnik i natjecanje i to podržavajućom vezom. Iz nekog predmeta može i ne mora biti organizirano jedno ili više natjecanja, dok natjecanje mora biti iz točno jednog predmeta. Slično, u dnevniku može, a i ne mora biti upisana jedna ili više ocjena iz nekog predmeta, dok je svaka upisana ocjena iz točno jednog predmeta. Dakle, veze predmet – natjecanje i predmet – dnevnik su podržive, jedan prema više veze.

3 Relacijski model

Nakon što stvorimo sve veze, relacijski iz MEV modela dobijemo tako što kliknemo na gumb s dvije plave strelice. U relacijskom je modelu stvoren međuentitet učenik_roditelj, čime je riješena veza više prema više. Novi entitet vezan je za entitete roditelj i učenik podržavajućim vezama. U tablicama relacijskog modela, osim primarnih ključeva, navedeni su i strani ključevi koje čine primarni ključevi tablica s kojima su one vezane.



Slika 2: Relacijski model

4 Kreiranje tablica i unosi u tablice

Tablice iz relacijskog modela u sql skripti kreiramo pomoću naredbe CREATE TABLE, kao što vidimo na sljedećoj slici.

```
CREATE TABLE roditelj (
    roditelj_id INTEGER CONSTRAINT roditelj_PK PRIMARY KEY ,
    ime             VARCHAR2(10) NOT NULL ,
    prezime         VARCHAR2(10) NOT NULL ,
    spol            VARCHAR2(1) NOT NULL ,
    adresa          VARCHAR2(100) NOT NULL ,
    telefon         VARCHAR2(20) NOT NULL ,
    email           VARCHAR2(30) NOT NULL
);
```

Slika 3

U zagradama navodimo atribute te njihove tipove. Uvjetom CONSTRAINT PRIMARY KEY osiguravamo jedinstvene vrijednosti u stupcu roditelj_id koji je primarni ključ. Obveznost atributa se postiže uvjetom NOT NULL. Kako bi se onemogućilo unošenje vrijednosti koje pokazuju na nepostojeće vrijednosti u drugoj tablici te brisanje vrijednosti iz roditeljske tablice prije nego što se obriše u referencirajućoj tablici, pokraj naziva atributa koji su strani ključevi navodi se uvjet CONSTRAINT FOREIGN KEY REFERENCES.

```
CREATE TABLE ucenik (
    ucenik_id INTEGER CONSTRAINT ucenik_PK PRIMARY KEY ,
    ime               VARCHAR2(10) NOT NULL ,
    prezime           VARCHAR2(10) NOT NULL ,
    datum_rodj        DATE NOT NULL ,
    spol              VARCHAR2(1) NOT NULL ,
    adresa            VARCHAR2(100) NOT NULL ,
    skolska_godina    VARCHAR2(20) NOT NULL ,
    razred_br         VARCHAR2(2) NOT NULL ,
    odjeljenje        VARCHAR2(1) NOT NULL ,
    CONSTRAINT ucenik_razred_FK FOREIGN KEY ( skolska_godina, razred_br, odjeljenje )
    REFERENCES razred ( skolska_godina, razred_br, odjeljenje )
);
```

Slika 4

Međutablicu učenik_razred, koja je dodatno stvorena u relacijskom modelu, također kreiramo i to na sljedeći način:

```
CREATE TABLE ucenik_roditelj (
    ucenik_id           INTEGER NOT NULL CONSTRAINT ucenik_roditelj_ucenik_FK
    REFERENCES ucenik ( ucenik_id ),
    roditelj_id         INTEGER NOT NULL CONSTRAINT ucenik_roditelj_roditelj_FK
    REFERENCES roditelj ( roditelj_id ) ,
    CONSTRAINT ucenik_roditelj_PK PRIMARY KEY ( ucenik_id, roditelj_id )
);
```

Slika 5

Kako su tu podržive veze, primarni ključ čini kombinacija stranih ključeva nasljeđenih od tablica ucenik i roditelj.
Želimo li za neki atribut postaviti zadanu vrijednost, učinit ćemo to naredbom DEFAULT nakon koje navodimo željenu vrijednost.

```
CREATE TABLE dnevnik (
    ocjena_id           INTEGER NOT NULL ,
    ocjena              INTEGER NOT NULL ,
    datum               DATE DEFAULT SYSDATE,
    biljeska            VARCHAR2(50) NOT NULL ,
    ucenik_id           INTEGER NOT NULL CONSTRAINT dnevnik_ucenik_FK
    REFERENCES ucenik ( ucenik_id ) ,
    predmet_id          INTEGER NOT NULL CONSTRAINT dnevnik_predmet_FK
    REFERENCES predmet ( predmet_id ) ,
    CONSTRAINT dnevnik_PK PRIMARY KEY ( ocjena_id, ucenik_id, predmet_id )
);
```

Slika 6

Primjerice, u ovoj tablici, zadana vrijednost atributa datum je današnji datum.
Tablice se popunjavaju pomoću naredbe INSERT INTO naziv_tablice VALUES, pokraj koje se u zagradama redom navode vrijednosti atributa koje se želi unijeti.

```
INSERT INTO pedagoska_mjera
VALUES (1, 'opomena', 'Učenik kasni na sat.', 8);
```

Slika 7

Ukoliko se ispusti neki od atributa za koje smo naveli zadanu vrijednost, unijet će se upravo ta vrijednost, a oni koji nemaju zadanu vrijednost bit će postavljeni na NULL. Vrijednosti atributa koji imaju uvjet NOT NULL se moraju navesti. Unos podataka potvrđujemo naredbom COMMIT koja osigurava njihov trajni zapis u tablicama. Želimo li kasnije ažurirati retke u tablicama, možemo to napraviti naredbom UPDATE, a ukoliko želimo obrisati retke, koristimo naredbu DELETE.

5 Upiti

Svaki upit sastoji se od ključnih riječi SELECT, nakon koje se piše ono što želimo da upit vrati i FROM, nakon koje dolaze tablice koje sadrže željene podatke. Želimo li rezultate upita poredati nekim redoslijedom, atribute po kojima ih želimo poredati navodimo nakon ključnih riječi ORDER BY. Uvjete koje rezultati upita trebaju zadovoljavati pišemo nakon ključne riječi WHERE. Jedan primjer jednostavnog upita nad tablicama je upit koji vraća adrese svih učenika 1.a razreda, školske godine 2019./2020.

```
SELECT ime || ' ' || prezime AS "Učenik", adresa
FROM ucenik
WHERE skolska_godina = '2019/2020' AND razred_br = '1.' AND odjeljenje = 'a'
ORDER BY prezime, ime;
```

Slika 8

Pisanjem znakova "||' ' ||" između atributa ime i prezime, rezultati ta dva atributa će se ispisati u jednom stupcu. Ključnom riječi AS dodijeljujemo stupacu željeni naziv.
Kod upita nad više tablica, ako imamo equi-spojeve, možemo koristiti ključne riječi JOIN i USING. Upit će vratiti one relacije kartezijevog produkta tablica, kod kojih nema NULL vrijednosti u stupcima po kojim ih spajamo (stupac u zagradama pokraj USING) te koje zadovoljavaju uvjete navedene pod WHERE. U sljedećem primjeru dohvaćamo predmet koji se predavao 1.12.2019. u učionici U2 tijekom 2. školskog sata.

```
SELECT p.naziv
FROM predmet p JOIN raspored r USING (predmet_id)
WHERE r.datum = TO_DATE('01/12/2019', 'dd/mm/yyyy') AND r.ucionica = 'U2' AND r.sat = 2;
```

Slika 9

U upitima možemo koristiti i agregirajuće funkcije, kao što su, primjerice, funkcija za prosjek AVG(), prebrojavanje COUNT(), maksimalnu vrijednost MAX() i zbroj

SUM(). Na sljedećoj slici prikazan je upit koji vraća broj pedagoških mjera svakog učenika 2019./2020., koristeći agregirajuću funkciju COUNT().

```
SELECT u.ime, u.prezime, COUNT(mjera_id)
FROM pedagoska_mjera p RIGHT OUTER JOIN ucenik u USING (ucenik_id)
JOIN razred USING (skolska_godina, razred_br, odjeljenje)
WHERE skolska_godina = '2019/2020'
GROUP BY u.ime, u.prezime;
```

Slika 10

Ako se u rezultat želi uključiti i one retke kod kojih su stupcu po kojem spajamo tablice NULL vrijednosti, možemo to napraviti tako da napišemo ključne riječi RIGHT OUTER prije riječi JOIN. Tada se takvi reci uključuju iz tablice s desne strane, a analogno se može učiniti i za lijevu tablicu s LEFT OUTER JOIN, odnosno za obje tablice s FULL OUTER JOIN.
Podupiti su pomoćni upiti unutar glavnog upita. Upit koji vraća razred (ili razrede) s najvećom prosječnom ocjenom iz svih predmeta, 2019./2020. školske godine, je primjer jednog upita u kojem možemo koristiti podupit.

```
SELECT razred_br || odjeljenje AS "Razred"
FROM dnevnik d JOIN ucenik u USING (ucenik_id) JOIN predmet p USING (predmet_id)
JOIN razred r USING (skolska_godina, razred_br, odjeljenje)
WHERE skolska_godina = '2019/2020'
GROUP BY razred_br, odjeljenje
HAVING AVG(d.ocjena) >= ALL (SELECT AVG(d.ocjena)
                        FROM dnevnik d JOIN ucenik u USING (ucenik_id) JOIN predmet p USING (predmet_id)
                        JOIN razred USING (skolska_godina, razred_br, odjeljenje)
                        WHERE skolska_godina = '2019/2020'
                        GROUP BY razred_br, odjeljenje);
```

Slika 11

Ključne riječi GROUP BY grupiraju retke prema vrijednosti, u ovom slučaju, atributa ocjena iz tablice dnevnik. Uvjete na grupe redaka pišemo pod HAVING, a operator ALL uspoređuje vrijednost sa svim vrijednostima iz podupita. Točnije, u ovom primjeru provjerava je li vrijednost veća ili jednaka svim vrijednostima iz podupita.
Na rezultirajuće retke dva ili više upita ili podupita možemo djelovati skupovnim opracijama, kao što su UNION, MINUS i INTERSECT. Upit koji vraća predmete koje je 1.12.2019. imao 1.a razred, a nije ih imao 1.b možemo napisati koristeći skupovnu operaciju MINUS, pomoću koje dobijemo one retke iz prvog podupita koji nisu ujedno rezultirajući reci drugog podupita.

```
SELECT p.naziv
FROM predmet p JOIN raspored r USING (predmet_id) JOIN razred r USING (skolska_godina, razred_br, odjeljenje)
WHERE r.datum = TO_DATE('01/12/2019', 'dd/mm/yyyy')
AND p.naziv IN (SELECT p.naziv
                FROM  predmet p JOIN raspored r USING (predmet_id) JOIN razred r
                USING (skolska_godina, razred_br, odjeljenje)
                WHERE r.datum = TO_DATE('01/12/2019', 'dd/mm/yyyy') AND razred_br = '1.'
                      AND odjeljenje = 'a'
                MINUS
                SELECT p.naziv
                FROM  predmet p JOIN raspored r USING (predmet_id) JOIN razred r
                USING (skolska_godina, razred_br, odjeljenje)
                WHERE r.datum = TO_DATE('01/12/2019', 'dd/mm/yyyy') AND razred_br = '1.'
                      AND odjeljenje = 'b');
```

Slika 12

6 Uvjeti

Na tablice možemo dodati različite uvjete. Osim uvjeta primarnog i stranog ključa te NOT NULL uvjeta koje smo dodali pri kreiranju tablica, postoje i CHECK, UNIQUE, CHECK OPTION te READ ONLY uvjet. Ako želimo osigurati da su vrijednosti u stupcu obvezni iz tablice predmet uvijek 0 ili 1, možemo to učiniti tako da dodamo uvjet CHECK:

```
ALTER TABLE predmet
ADD CONSTRAINT predmet_obvezni_ck
CHECK (obvezni IN (0, 1));
```

Slika 13

Ukoliko pokušamo unijeti vrijednost različitu od 0 i 1, baza podataka će vratiti grešku. Uvjet UNIQUE će osigurati jedinstvenost vrijednosti u stupcu kabinet iz tablice nastavnik.

```
ALTER TABLE nastavnik
ADD CONSTRAINT nastavnik_kabinet_uq UNIQUE (kabinet);
```

Slika 14

7 Komentari na tablice

Kako bi i drugi korisnici znali čemu koja tablica služi, možemo dodati komentar na svaku tablicu koristeći ključnu riječ COMMENT, kao što je prikazano na sljedećoj slici.

```
COMMENT ON TABLE natjecanje IS
'tablica natjecanje sadrži razinu, mjesto održavanja i naziv natjecanja na
kojemu se učenik natjecao te mjesto koje je osvojio';
```

Slika 15

8 Indeksi

Indekse u bazama podataka koristimo za brže pretraživanje redaka. Indeks koji kreiramo na stupcu u kojem imamo velik broj razlicitih vrijednosti zovemo "B-tree" indeks. Ako upiti u kojima koristimo određeni stupac u WHERE klauzuli vrate manje od 10 posto redaka, možemo ga koristiti za indeksiranje. Sintaksu kreiranja B-tree indeksa možemo vidjeti na sljedećem primjeru.

```
CREATE INDEX indeks_roditelj_prezime ON roditelj(prezime);
```

Slika 16: B-tree indeks koji će ubrzati pretraživanje roditelja po prezimenu

S druge strane, indeks koji kreiramo na stupcu koji sadrži velik broj redaka, a malen broj različitih vrijednosti zovemo "bitmap" indeks. Ako je broj razlicitih vrijednosti stupca manji od 1 posto broja redaka u tablici, taj stupac je dobar kandidat za bitmap indeks. Također, ažuriranja vrijednosti indeksiranog stupca trebala bi biti rijetka. Sintaksa kreiranja bitmap indeksa je sljedeća:

```
CREATE BITMAP INDEX indeks_pedagoska_mjera_mjera ON pedagoska_mjera(mjera);
```

Slika 17: bitmap indeks koji će ubrzati pretraživanje pedagoških mjera po vrsti mjere

9 Procedure

Procedura je SQL kod koji sadrži skup naredbi. Možemo ju pozvati prosljeđujući joj parametre. Slijedi primjer u kojemu procedura ažurira nastavnika koji predaje određeni predmet.

```
CREATE OR REPLACE PROCEDURE update_predmet_nastavnik(
p_naziv IN predmet.naziv%TYPE,
p_ime IN nastavnik.ime%TYPE,
p_prezime IN nastavnik.prezime%TYPE,
p_spol IN nastavnik.spol%TYPE DEFAULT NULL,
p_kabinet IN nastavnik.kabinet%TYPE DEFAULT NULL
) AS
v_brojac INTEGER;
v_nastavnik_id nastavnik.nastavnik_id%TYPE;
BEGIN
        SELECT COUNT(*)
        INTO v_brojac
        FROM nastavnik
        WHERE ime = p_ime AND prezime = p_prezime;

        IF v_brojac = 1 THEN

        SELECT nastavnik_id
        INTO v_nastavnik_id
        FROM nastavnik
        WHERE ime = p_ime AND prezime = p_prezime;

        UPDATE predmet
        SET nastavnik_id = v_nastavnik_id
        WHERE naziv = p_naziv;

        COMMIT;

        ELSIF v_brojac = 0 THEN

        SELECT MAX(nastavnik_id)+1
        INTO v_nastavnik_id
        FROM nastavnik;

        INSERT INTO nastavnik
        VALUES (v_nastavnik_id, p_ime, p_prezime, p_spol, p_kabinet);

        UPDATE predmet
        SET nastavnik_id = v_nastavnik_id
        WHERE naziv = p_naziv;

        COMMIT;

        END IF;
```

Slika 18

```
    EXCEPTION
    WHEN OTHERS THEN
    ROLLBACK;
END update_predmet_nastavnik;
```

Slika 19

Proceduri update_predmet_nastavnik se proslijede naziv predmeta, ime, prezime, kabinet i spol novog nastavnika kao parametri, pri čemu, ako se nastavnik već nalazi u bazi, ne moramo proslijediti kabinet i spol.

```
SELECT COUNT(*)
INTO v_brojac
FROM nastavnik
WHERE ime = p_ime AND prezime = p_prezime;
```

Slika 20

U ovom dijelu procedure u prethodno deklariranu varijablu v_brojac sprema se broj nastavnika iz tablice s imenom i prezimenom koje smo proslijedili. Ako je broj takvih nastavnika 1, to jest ako nastavnik već postoji u bazi, potrebno je pronaći odgovarajući nastavnik_id, koji spremamo u varijablu v_nastavnik_id. Sada možemo ažurirati tablicu predmet tako da postavimo vrijednost u stupcu nastavnik_id na v_nastavnik_id u onom retku u kojem je naziv predmeta jednak proslijeđenom. Ukoliko je vrijednost varijable v_brojac jednaka 0, to jest nastavnik ne postoji u bazi, prvo ga trebamo dodati. Novu vrijednost u stupcu nastavnik_id, koja će mu biti pridružena, ćemo dobiti tako da odredimo maksimalnu vrijednost stupca uvećanu za jedan.

```
    EXCEPTION
    WHEN OTHERS THEN
    ROLLBACK;
END update_predmet_nastavnik;
```

Slika 21

U ovom dijelu procedure promjene će se opozvati, ukoliko dođe do neke iznimke.

10 Okidači

Okidač je procedura koja se pokreće automatski pokretanjem naredbe INSERT, UPDATE ili DELETE. Sintaksa kreiranja okidača može se vidjeti na sljedećem primjeru.

```
CREATE OR REPLACE TRIGGER before_dnevnik_insert
BEFORE INSERT
ON dnevnik
FOR EACH ROW
DECLARE
v_br_ocjena INTEGER;
BEGIN
    SELECT COUNT(*)
    INTO v_br_ocjena
    FROM dnevnik
    WHERE datum = :NEW.datum  AND ucenik_id = :NEW.ucenik_id;

    IF v_br_ocjena >= 3 THEN
        DBMS_OUTPUT.PUT_LINE('Učenik je danas ocijenjen već tri puta.');
    END IF;
END before_dnevnik_insert;
```

Slika 22

Okidač before_dnevnik_insert upozorava prije nego se upiše četvrta ocjena učeniku na isti datum. U prethodno deklariranu varijablu v_br_ocjena sprema se broj redaka iz tablice dnevnik gdje je su datum i ucenik_id jednaki unesenima. Ako je vrijednost varijable v_br_ocjena veća ili jednaka 3, ispisati će se upozorenje.

11 Zaključak

Korištenje ove baze podataka olakšava organizaciju informacija bitnih za svakodnevni rad škole. Nastavnicima i drugim zaposlenicima škole omogućava pohranjivanje raznih podataka te vrlo brzo dohvaćanje istih kasnije.