

INFORMACIONI SISTEM ZA PODRŠKU UPRAVLJANJA KORISNIČKIM SADRŽAJEM TIPA PEDIJATRIJSKE ORDINACIJE

**-SISTEMI BAZA PODATAKA-**

**Vuk Stojić IT51G2020**

**Novi Sad, 2023.**

**SADRŽAJ**

[UVOD 2](#_TOC_250014)

[ANALIZA PROGRAMSKOG DOMENA 2](#_TOC_250013)

[ER MODEL 3](#_TOC_250012)

[ER MODEL PODŠEME 4](#_TOC_250011)

[TABELARNI PRIKAZ OBELEŽJA I OGRANIČENJA 5](#_TOC_250010)

[RELACIONI MODEL 6](#_TOC_250009)

[DDL 6](#_TOC_250008)

[DML 6](#_TOC_250007)

[SQL 6](#_TOC_250006)

[OBJEKTI 6](#_TOC_250005)

[UPITI 7](#_TOC_250004)

[FUNKCIJE 8](#_TOC_250003)

[PROCEDURE 9](#_TOC_250002)

[TRIGERI 10](#_TOC_250001)

[ZAKLJUČAK 11](#_TOC_250000)

# UVOD

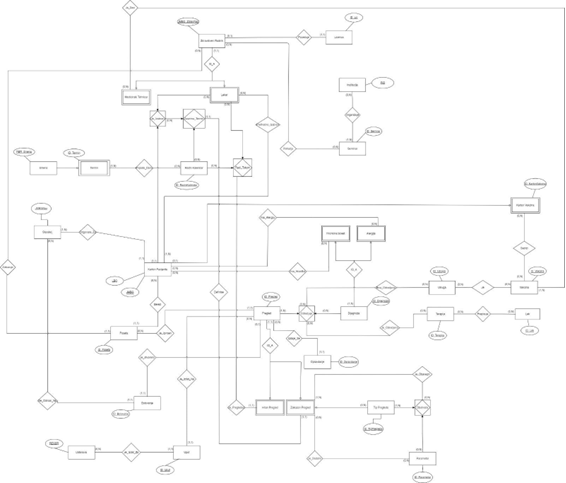
Projekat koji je priložen uz ovu dokumentaciju predstavlja projekat segmenta baze podataka Pedijatrijske Ordinacije koji omogućuje korisnicima kontrolu nad licencama zdravstvenih radnika, kao i manipulaciju nad zakazanim terminima (pregledima). Glavni fokus ovog samostalnog projekta jeste mogućnost obnavljanja odgovarajućih licenci uz priložene uslove. U nastavku priloženi su odgovarajući dijagrami, tabelarne šeme relacija, kao i primeri funkcionalnosti samog projekta.

# ANALIZA PROGRAMSKOG DOMENA

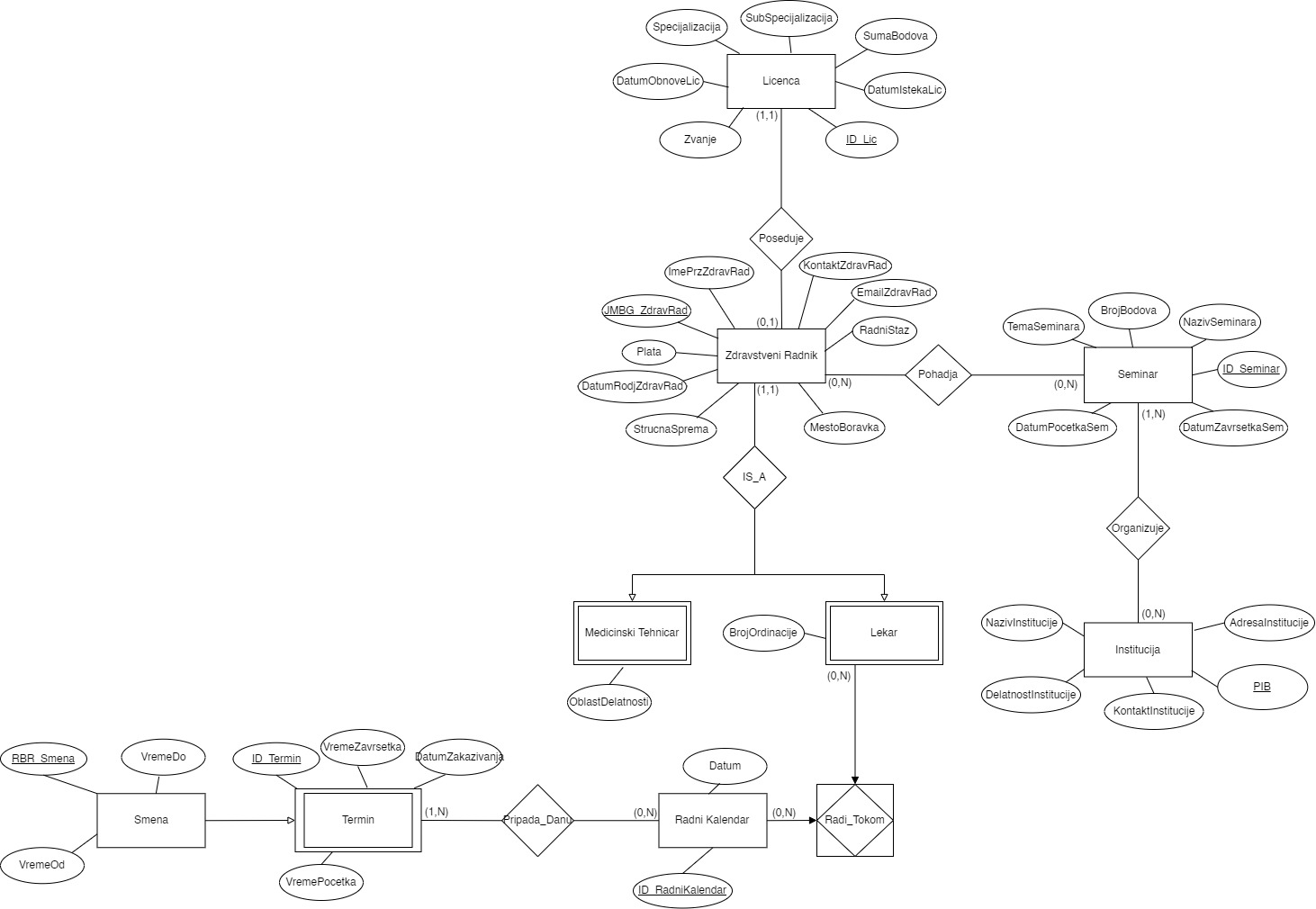
Projekat stvoren je sa motivom da omogući adekvatno upisivanje, brisanje i čitanje podataka kako bi informacioni sistem bio u skladu sa realnim sistemom pedijatrijske ordinacije. Pomoću procesa definisanih u okviru dokumentacije postiže se upravo to. Različiti radnici imaju različite privilegije u korišćenju baze podataka. Određeni zaposleni mogu da vrše kontrolu nad licencama zdravstvenih radnika, kao i aktivnim seminarima. Sa druge strane, sami lekari imaju mogućnost za zakazivanje pregleda u željenom terminu i prijavljivanje za seminare. Još neke mogućnosti jesu upravo dodavanje institucija koje organizuju same seminare, kao i novih zaposlenih tj. zdravstvenih radnika.

Pomenute funkcionalnosti omogućene bi bile putem odgovarajuće aplikacije koja pruža jasno i jednostavno korisničko okruženje. Pritom baza podataka se koristi kako bi čuvala podatke i kontrolisala unos samih podataka u sistem koristeći ograničenja.

# ER MODEL



# ER MODEL PODŠEME



Gore navedena podšema definiše sve elemente baze koji su potrebni radi korektnog funkcionisanja prethodno definisanih procesa.

Tip Entiteta “Zdravstveni Radnik” deo je podšeme kako bi radnici mogli da povežu kome konkretna licenca pripada, kao i koji radnik je bio na kojem seminaru. Sa druge strane, mogu se i povezati pomenuti datumi i termini pregleda konkretnog radnika.

Tip Entiteta “Licenca” je u okviru podšeme kako bismo pristupili podacima o specijalizaciji, subspecijalizaciji, sumi bodova, kao i datuma isteka i obnove licence. Suma bodova predstavlja obeležje koje pruži uvid u to koliko je radnik blizu obanvljanja licence. Radnik stiče uslov za obnavljanjem licence kada skupi sumu bodova koja je karakteristična za njegovu specijalizaciju. Bodovi se skupljaju pohađanjem seminara, što nas dovodi do sledećeg tipa entiteta.

Tip Entiteta “Seminar” i “Institucija” pružaju informacije o trenutno aktivnim seminarima i institucijama koje ih organizuju, tematiku seminara, kao i broj bodova koji ti seminari nose.

Tipovi entiteta “Medicinski Tehničar” i “Lekar” daju uvid u mesto gde bismo pronašli lekara/medicinskog tehničara (obeležjima “OblastDelatnosti” i “BrojOrdinacije”).

Tipovi entiteta “Radni Kalendar”, “Termin” i “Smena” definišu smene u kojima radnici rade, kao i termine u okviru tih smena i određenih datuma u radnom kalendaru.

# TABELARNI PRIKAZ OBELEŽJA I OGRANIČENJA

*1. Tabela 3. 1 – ZDRAVSTVENI RADNIK*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Naziv obeležja** | **Opis obeležja** | **Tip**  **podataka** | **Dužina podataka** | **Null** | **Uslov** |
| *JMBG\_ZdravRad* | Matični broj zdravstvenog radnika | INTEGER | 13 | ⊥ | d>0 |
| *ImePrzZdravRad* | Ime i prezime zdravstvenog radnika | VARCHAR | 20 | ⊥ | Δ |
| *KontaktZdravRad* | Broj telefona zdravstvenog radnika | VARCHAR | 20 | ⊥ | Δ |
| *EmailZdravRad* | Email adresa zdravstvenog radnika | VARCHAR | 30 | ⊥ | Δ |
| *Plata* | Mesečna plata zdravstvenog radnika | VARCHAR | 10 | T | Δ |
| *DatumRodjZdravRad* | Datum rođenja zdravstvenog radnika | DATE |  | ⊥ | Δ |
| *DatumZaposljavanja* | Datum zapošljavanja zdravstvenog radnika | DATE |  | ⊥ | Δ |
| *MestoBoravka* | Pun naziv mesta boravka zdravstvenog radnika | VARCHAR | 30 | T | Δ |
| *RadniStaz* | Radni staž datog lekara | INTEGER | 20 | T | Δ |
| K = {*JMBG\_ZdravRad*} | | | | | |
| *Unique*(*ZdravstveniRadnik****,*** *KontaktZdravRad*) = T  *Unique*(*ZdravstveniRadnik****,*** *EmailZdravRad*) = T | | | | | |

*Tabela 3. 2 - LEKAR*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Naziv obeležja** | **Opis obeležja** | **Tip**  **podataka** | **Dužina podataka** | **Null** | **Uslov** |
| *JMBG\_ZdravRad* | Matični broj zdravstvenog radnika | INTEGER | 13 | ⊥ | d>0 |
| *BrojOrdinacije* | Numericka oznaka ordinacije | INTEGER | 3 | ⊥ | d>0 |
| K = {*JMBG\_ZdravRad*} | | | | | |
|  | | | | | |

*Tabela 3. 3 – MEDICINSKA SESTRA*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Naziv obeležja** | **Opis obeležja** | **Tip**  **podataka** | **Dužina podataka** | **Null** | **Uslov** |
| *JMBG\_ZdravRad* | Matični broj zdravstvenog radnika | INTEGER | 13 | ⊥ | d>0 |
| *OblastDelatnosti* | Oblast delatnosti (npr. sprat) | VARCHAR | 50 | ⊥ | Δ |
| K ={*JMBG\_ZdravRad*} | | | | | |
|  | | | | | |

*Tabela 3. 4 – LICENCA*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Naziv obeležja** | **Opis obeležja** | **Tip**  **podataka** | **Dužina podataka** | **Null** | **Uslov** |
| *ID\_Lic* | Identifikacioni broj parametra | INTEGER | 14 | ⊥ | d>0 |
| *JMBG\_ZdravRad* | Matični broj zdravstvenog radnika  čija je licenca | INTEGER | 13 | ⊥ | d>0 |
| *DatumObnoveLic* | Datum poslednje obnove licence | DATE |  | ⊥ | Δ |
| *DatumIstekaLic* | Datum isteka licence u slučaju ne ispunjavanja dovoljne količine bodova | DATE |  | ⊥ | Δ |
| *Zvanje* | Zvanje | VARCHAR | 50 | ⊥ | Δ |
| *Specijalizacija* | Specijalizacija doktora | VARCHAR | 30 | ⊥ | Δ |
| *SumaBodova* | Suma ostvarenih bodova | INTEGER | 3 | ⊥ | d>100 |
| *SubSpecijalizacija* | Subspecijalizacija doktora | VARCHAR | 30 | T | Δ |
| K = {*ID\_Lic*} | | | | | |
| *DatumObnoveLic <* *DatumIstekaLic* | | | | | |

*Tabela 3. 5 – INSTITUCIJA*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Naziv obeležja** | **Opis obeležja** | **Tip**  **podataka** | **Dužina podataka** | **Null** | **Uslov** |
| *PIB* | Poreski identifikacioni broj institucije | INTEGER | 9 | ⊥ | d>0 |
| *NazivInstitucije* | NazivInstitucije | VARCHAR | 50 | ⊥ | Δ |
| *DelatnostInstitucije* | Delatnost Institucije | VARCHAR | 30 | ⊥ | Δ |
| *KontaktInstitucije* | Kontakt Institucije | INTEGER | 10 | ⊥ | Δ |
| *AdresaInstitucije* | AdresaI nstitucije | VARCHAR | 30 | ⊥ | Δ |
| K = {*PIB*} | | | | | |
| *Unique*(*Institucija****,*** *NazivInstitucije*) = T  *Unique*(*Institucija****,*** *KontaktInstitucije*) = T | | | | | |

*Tabela 3. 6 – SEMINAR*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Naziv obeležja** | **Opis obeležja** | **Tip**  **podataka** | **Dužina podataka** | **Null** | **Uslov** |
| *ID\_Seminar* | Identifikacioni broj parametra | INTEGER | 100 | ⊥ | d>0 |
| *BrojBodova* | Broj bodova koji se dodaju na licencu nakon pohadjanja | INTEGER | 2 | ⊥ | d>0 |
| *TemaSeminara* | Tema seminara | NVARCHAR | 500 | ⊥ | Δ |
| *NazivSeminara* | Naziv seminara | VARCHAR | 50 | ⊥ | Δ |
| *DatumPocetkaSem* | Datum pocetka seminara | DATE |  | ⊥ | Δ |
| *DatumZavrsetkaSem* | Datum zavrsetka sem | DATE |  | ⊥ | Δ |
| K = {*ID\_Seminar*} | | | | | |
| *DatumPocetkaSem* < *DatumZavrsetkaSem* | | | | | |

*Tabela 3. 7 – TERMIN*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Naziv obeležja** | **Opis obeležja** | **Tip**  **podataka** | **Dužina podataka** | **Null** | **Uslov** |
| *ID\_Termin* | Identifikaciono obeležje termina | INTEGER | 10 | ⊥ | d>0 |
| *DatumZakazivanja* | Datum kad je termin zakazan | DATE |  | ⊥ | Δ |
| *VremePocetka* | Vreme kad je termin zakzan | TIME |  | ⊥ | Δ |
| *VremeZavrsetka* | Kad je očekivano da se termin završi | TIME |  | ⊥ | Δ |
| K = {*ID\_Termin* + *RBR\_Smena*} | | | | | |
| *VremePocetka* < *VremeZavrsetka* | | | | | |

*Tabela 3. 8 – SMENA*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Naziv obeležja** | **Opis obeležja** | **Tip**  **podataka** | **Dužina podataka** | **Null** | **Uslov** |
| *RBR\_Smena* | Redni broj smene | INTEGER | 10 | ⊥ | d>0 |
| *VremeOd* | Vreme početka smene | TIME |  | ⊥ | Δ |
| *VremeDo* | Vreme završetka smene | TIME |  | ⊥ | Δ |
| K = {*RBR\_Smena*} | | | | | |
| *VremeOd* < *VremeDo* | | | | | |

*Tabela 3. 9 – RADNI KALENDAR*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Naziv obeležja** | **Opis obeležja** | **Tip**  **podataka** | **Dužina podataka** | **Null** | **Uslov** |
| *ID\_RadniKalendar* | Identifikaciono obeležje radnog kalendara | INTEGER | 10 | ⊥ | d>0 |
| *Datum* | Odabrani datum u radnom kalendaru | DATE |  | ⊥ | Δ |
| K = {*ID\_RadniKalendar*} | | | | | |
|  | | | | | |

*Tabela 3. 10 – POHAĐA*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Naziv obeležja** | **Opis obeležja** | **Tip**  **podataka** | **Dužina podataka** | **Null** | **Uslov** |
| *JMBG\_ZdravRad* | Matični broj zdravstvenog radnika | INTEGER | 13 | ⊥ | d>0 |
| *ID\_Seminar* | Identifikacioni broj parametra | INTEGER | 100 | ⊥ | d>0 |
| K = {*ID\_Seminar* + *JMBG\_ZdravRad*} | | | | | |
|  | | | | | |

*Tabela 3. 11 – ORGANIZUJE*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Naziv obeležja** | **Opis obeležja** | **Tip**  **podataka** | **Dužina podataka** | **Null** | **Uslov** |
| *PIB* | Poreski identifikacioni broj institucije | INTEGER | 9 | ⊥ | d>0 |
| *ID\_Seminar* | Identifikacioni broj parametra | INTEGER | 100 | ⊥ | d>0 |
| K = {*PIB* + *ID\_Seminar*} | | | | | |
|  | | | | | |

*Tabela 3. 12 – PRIPADA\_DANU*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Naziv obeležja** | **Opis obeležja** | **Tip**  **podataka** | **Dužina podataka** | **Null** | **Uslov** |
| *RBR\_Smena* | Redni broj smene | INTEGER | 10 | ⊥ | d>0 |
| *ID\_Termin* | Identifikaciono obeležje termina | INTEGER | 10 | ⊥ | d>0 |
| *ID\_RadniKalendar* | Identifikaciono obeležje radnog kalendara | INTEGER | 10 | ⊥ | d>0 |
| K = {*RBR\_Smena* + *ID\_Termin* + *ID\_RadniKalendar*} | | | | | |
|  | | | | | |

# RELACIONI MODEL

Relacioni model podataka je matematički model za organizovanje i manipulaciju podacima. Osnovna ideja relacionog modela je da podaci budu organizovani u tabelama koje se nazivaju relacije. Svaka relacija se sastoji od skupa redova, gde svaki red predstavlja jedan zapis ili entitet, dok se svaka kolona u tabeli odnosi na određenu karakteristiku ili atribut tog entiteta. Relacije se međusobno povezuju preko ključeva kako bi se uspostavili odnosi između njih.

Relaciona baza podataka je kolekcija povezanih tabela koje predstavljaju entitete i njihove međusobne odnose. Osnovna jedinica podataka u relacionoj bazi je tabela koja se sastoji od redova (zapisa) i kolona (atributa). Relaciona baza podataka koristi relacioni model podataka kao osnovu za definisanje strukture podataka i uspostavljanje veza između tabela.

Relacioni model podataka i relacione baze podataka su široko korišćeni u industriji i akademskim krugovima zbog svoje fleksibilnosti, skalabilnosti i efikasnosti u organizaciji i manipulaciji podacima.

# DDL

DDL (Data Definition Language) je skup jezičkih instrukcija koje se koriste za definisanje i manipulaciju strukture baze podataka. DDL se koristi za kreiranje, izmenu i brisanje objekata u bazi podataka, kao što su tabele, indeksi, pogledi, procedure i druge komponente. DDL se najčešće koristi prilikom projektovanja i kreiranja baze podataka, jer omogućava definisanje strukture i karakteristika objekata. Sa druge strane, DDL se koristi kako bismo održali bazu u stanju koje je u skladu sa realnim sistemom DDL komande koje su korištene u okviru ovog projekta jesu:

CREATE - Radi kreiranja tabela, procedura, funkcija, i drugih objekata u bazi podataka.

*Create Table Seminar {*

*ID\_Seminar numeric(10),*

*TemaSeminara nvarchar(100)*

*}*

Iznad navedeni kod kreira tabelu pod nazivom “Seminar” sa dva obeležja. ID\_Seminar koji je numeric tipa, što znači da može biti isključivo broj, i to dužine do 10. TemaSeminara je obeležje nvarchar tipa, što znači da je tekstualnog tipa, maksimalne dužine 100.

ALTER - Korišteno kako bi se izmenila struktura prethodno pomenutih objekata, u ovom konkretnom projektu je korišteno samo radi implementacije sekvence unutar tabela.

*Alter Table Seminar Add Constraint SEQ\_seminar default(next value for seq\_seminar) for ID\_Seminar;*

Gore navedeni kod je korišten kako bi kreirane sekvence implementiralo u okviru prethodno definisane tabele Seminar. Ovom sekvencom mi omogućujemo da se prilikom pravljenja instance tabele ne mora navoditi vrednosti obeležja ID\_Seminar, nego se automatski inkrementira za jedan (u ovom slučaju).

DROP – Funkcionalnost DROP komande leži u brisanju postojećih objekata. Implementirano je brisanjem svih objekata skripte pre samog izvršavanja iste skripte (query-a).

*IF Object\_Id ('Seminar', 'U') is not null*

*Drop Table Seminar;*

*Go*

Ispisani kod proverava da li postoji tabela Seminar i briše je iz sistema ukoliko postoji.

# DML

DML (Data Manipulation Language) je skup jezičkih instrukcija koje se koriste za manipulaciju podacima u bazi podataka. DML omogućava dodavanje, izmenu, brisanje i preuzimanje **podataka** iz tabela. Dakle za razliku od DDL-a, DML direktno manipuliše podacima u objektima definisanim kroz DDL. Primeri komandi:

SELECT – Komanda kojom preuzimamo podatke koristeći neke uslove (ili bez uslova). Ova komanda omogućava i sortiranje podataka, kao i izvršavanje određenih funkcija nad njima.

*Select min(Plata), avg(Plata), max(Plata)*

*from ZdravstveniRadnik*

*where MestoBoravka = ‘Novi Sad’;*

Navedeni primer će pronaći minimalnu, srednju i maksimalnu vrednost plate iz tabele zdravstvenih radnika iz Novog Sada.

INSERT - Pomoću INSERT-a upisuju se podaci u tabele, specificiranjem željene tabele u koju ih upisujemo, kao i same vrednosti podataka.

*INSERT INTO Seminar (ID\_Seminar, TemaSeminara)*

*VALUES (1, N'Nove tehnologije u medicini');*

Iznad definisani kod unosi vrednosti za obeležja ID\_Seminar i TemaSeminara kao “1” i “Nove tehnologije u medicini”, respektivno, u tabelu Seminar.

UPDATE – Koristi se kako bi se menjali već postojeći podaci u tabelama.

*Update Licenca*

*Set SumaBodova = SumaBodova + 5*

*where JMBG\_ZdravRad = 1234567890012;*

Ovaj kod će povećati vrednost obeležja “SumaBodova” za pet radniku kojem je JMBG baš jednak vrednosti “1234567890012”.

DELETE - Pomoću ove komande se brišu vrednosti u tabelama. Brisanje se može vršiti pod određenim uslovima.

*Delete from Seminar*

*where ID\_Seminar = 1;*

Ispisani kod briše red u tabeli “Seminar” gde je ID\_Seminar baš vrednost 1.

# SQL

SQL (Structured Query Language) je programski jezik koji se koristi za upravljanje i manipulaciju relacionim bazama podataka. SQL podržava DML, i DDL i time omogućava definisanje strukture baze podataka, kreiranje tabela i ograničenja, kao i izvršavanje upita za preuzimanje, dodavanje, izmenu i brisanje podataka. (sql se u projektu prikazuje kroz upite)

# OBJEKTI

Kroz kreiranje projekta korišćen je velik broj različitih objekata koji se mogu kreirati i koristiti kroz SQL. Ovim objektima definišemo strukturu baze podataka, kao i njena ograničenja i dodatne funkcionalnosti koje omogućavaju automatizaciju pojedinih procesa. U okviru projekta korišćeni su:

SCHEMA - Schema (šema) je logička grupa objekata, kao što su tabele, pogledi, funkcije i procedura, koja se koristi za organizaciju i razdvajanje podataka. U ovom projektu kreirana je schema “Projekat” kojom smo izdvojili sve objekte projekta od ostatka baze podataka.

TABLE - Tabela je osnovni objekat u SQL-u koji se koristi za organizaciju podataka u strukturi tabele sa redovima (unosima) i kolonama (atributima). Svaki red u tabeli predstavlja pojedinačnu instancu podataka, dok svaka kolona predstavlja određenu karakteristiku tih podataka. Tabele su u projektu korištene kako bi se realizovali svi Tipovi Entiteta sa dijagrama podšeme baze podataka.

SEQUENCE - Sekvenca se koristi za generisanje jedinstvenih brojeva u sekvencijalnom redosledu. Sekvence se često koriste za automatsko generisanje primarnih ključeva. Sekvence omogućavaju generisanje nizova brojeva koje je moguće koristiti u INSERT naredbama ili prilikom dodele vrednosti kolonama. Upravo na takav način su sekvence iskorištene u projektu, gde je za svaki primarni ključ u tabelama vezana adekvatna sekvenca.

PROCEDURE - Procedura je objekat koji sadrži skup SQL naredbi koje se izvršavaju kao logička jedinica. Procedura se može koristiti za izvršavanje određenih kompleksnih operacija nad podacima. Takođe, procedura omogućava programeru da definiše tačan proces izvršavanja kako bi stekao željeni efekat. Procedura se može kreirati pomoću DDL naredbe CREATE PROCEDURE, a zatim se može pozivati i izvršavati koristeći odgovarajuće DML naredbe. Procedura je korištena u okviru projekta kako bi se omogućila automatizacija procesa obnavljanja licenci radnika i visestruke istovremene promene bodova na seminarima.

FUNCTION - Funkcija je objekat u SQL-u koji vraća vrednost na osnovu unetih parametara. Funkcije se koriste za izračunavanje vrednosti, obradu podataka ili obavljanje određenih operacija. Funkcije mogu biti unapred definisane (ugrađene) u SQL jeziku ili mogu biti korisnički definisane. Za razliku od procedura, funkcije moraju primiti ulazne parametre i vratiti određene parametre. Funkcije se mogu koristiti u SELECT naredbama, kao deo izraza ili u uslovima upita. U projektu je funkcija iskorištena kako bi se na osnovu unetog datuma, smene i broja dana prikazali svi zauzeti termini u sistemu.

TRIGGER - Triger se koristi za automatsko izvršavanje određenih naredbi kada se dogodi određeni događaj, kao što je INSERT, UPDATE ili DELETE nad tabelom. Trigeri se često koriste za održavanje doslednosti podataka ili za obavljanje određenih akcija pri promeni podataka. Trigeri se mogu dodati na tabele korišćenjem DDL naredbe CREATE TRIGGER. Prvi trigger koji je integrisan u projekat jeste AFTER INSERT, UPDATE trigger koji se izvršava nad tabelom “Zdravstveni Radnik”. Pomenuti trigger nakon unosa ili ažuriranja radnika računa njegov radni staž. Drugi implementirani trigger je INSTEAD OF DELETE trigger koji se izvršava nad tabelom “Seminar”, on omogućava dodavanje odgovarajućeg broja poena radnicima koji su prisustvovali na seminaru nakon brisanja seminara iz baze podataka.

CURSOR - Kursor se koristi za prolazak kroz skup rezultata upita korak po korak, što omogućava iteraciju kroz redove i izvršavanje operacija nad tim redovima. Kursori se često koriste u procedurama ili funkcijama za obradu rezultata upita ili za iterativno izvršavanje naredbi. Kursori se mogu deklarisati, otvoriti, zatvoriti i pomerati pomoću odgovarajućih SQL naredbi. Kursori su u projektu realizovali prolazak kroz tabelu “Licenca” pod određenim uslovima i time omogućili rad jedne procedure i jednog trigger-a.

# UPITI

## Prvi upit

Prvobitni upit prikazuje ime, prezime, kao i radni staž svih radnika koji imaju radni staž manji od srednje vrednosti radnog staža na nivou baze podataka. Rezultat je sortiran u opadajućem redosledu. Kako bi prikaz pomenutih podataka bio moguć, pristupa se tabeli “Zdravstveni Radnik” i unutar nje poredimo vrednost radnog staža svakog radnika sa srednjom vrednošću radnog staža. Srednja vrednost radnog staža dobijena je kroz ugrađenu funkciju avg() koja vraća srednju vrednost određenog numeričkog obeležja. Rezultat prvog upita je sledeći:

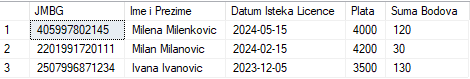


Kao što je pomenuto, svi navedeni radnici imaju radni staž kraci od proseka (prosečan radni staž je 6).

## Drugi upit

Sledeći upit izlistava JMBG, ime i prezime, platu i sumu bodova svih radnika sa platom manjom od srednje vrednosti plate, važećom licencom i sumom bodova većom od 0.

Podatke pronalazimo spajanjaem (join-ovanjem) tabela “Zdravstveni Radnik” i “Licenca” zajedničkim ključem tj. Obeležjem “JMBG\_ZdravRad”. Pritom u WHERE klauzulu navodimo da obeležje “DatumIstekaLic” (podatak iz tabele Licenca) mora biti veće od današnjeg datuma, kao i da “SumaBodova” (takođe iz tabele Licenca) mora sadržati vrednost veću od 0. Nadalje, u HAVING klauzuli postavljamo uslov da obeležje “Plata” mora biti manja od srednje vrednosti plata radnika iz tabele Zdravstveni Radnik. Rezultat upita je:



Bitna napomena jeste da rezultat nije mogao da uključuje medicinske sestre jer one ne poseduju licencu. Takođe, treba imati na umu da je srednja vrednost plate približno 4589.

## Treći upit

Naredni upit prikazuje sve lekare koji su pohađali seminar koji nosi najviše bodova od institucije “Bolnica Novi Sad”, a ujedno tim lekarima ističe licenca ove godine.

Podatke koje ovaj upit prikazuje su JMBG, ime, prezime, kontakt, datum isteka licence, kao i sumu bodova licence zdravstvenog radnika. Uz podatke o radniku navodi se i naziv institucije koja organizuje seminar i broj bodova koji taj seminar nosi. Kako bismo došli do odgovarajućih podataka moramo izvršiti spoj između tabela “ZdravstveniRadnik”, “Licenca”, “Pohađa”, “Seminar”, “Organizuje” i “Institucija”. Iz tabele Zdravstveni Radnik uzimamo podatke o JMBG-u, imenu, prezimenu i kontakt zdravstvenog radnika. Tabela Licenca nam pruža podatke o datumu isteka licence i sumi bodova. Iz tabele Institucija preuzimamo podatke o nazivu institucije, dok broj bodova preuzmemo iz tabele Seminar. Tabele Pohađa (Lekar pohađa Seminar) i Organizuje (Institucija organizuje Seminar) predstavljaju tabele poveznika odgovarajućih tipova entiteta.

Nakon spajanja tabela definišemo WHERE klauzulu koja pomoću ugrađene funkcije Year() uzima godinu iz datuma isteka licence i poredi je sa trenutnom godinom, kako bismo proverili da li radniku istice licenca ove godine.

Takođe u okviru WHERE klauzule proveravamo da li je baš taj seminar onaj seminar koji nosi najviše bodova. To proveravamo putem select upita unutar same klauzule kojim uzimamo najveci broj bodova (funkcijom max()) iz tabele seminar, gde je naziv institucije baš “Bolnica Novi Sad” (ovo proveravamo još jednom WHERE klauzulom).

Rezultat upita:



## Četvrti upit

Četvrti upit izlistava sve lekare sa specijalizacijom “Ortopedija” koji imaju slobodan termin u prvoj smeni datuma “2023-09-01”.

Upit prikazuje ime, prezime, kontakt, broj ordinacije, specijalizaciju i poslednji zakazani termin. Poslednji zakazani termin dobijamo korišćenjem ugrađene funkcije max() i obeležjima “VremePocetka” i “VremeZavršetka”. Koristimo oba kako bismo prikazali kompletno trajanje poslednjeg termina tj. Kada počinje i kada se završava. Pored funkcije max() moramo koristiti i cast() kako bismo podatak koji je tipa TIME pretvorili u podatak time varchar i izvršili konkatenaciju (spajanje string-ova). Sve pomenute podatke dobija iz tabela “Zdravstveni Radnik”, “Licenca”, “Lekar”, “RadniKalendar” i “Termin”. Takođe koristimo tabele poveznika “Radi\_Tokom” i “Pripada\_Danu”.

Nakon navođenja korektnih podataka i njihovih izvornih tabela, definišemo WHERE klauzulu kojom uslovljavamo da vrednost obeležja “Specijalizacija” (iz tabele Licenca) bude “Kardiologija”. Nadalje u istoj klauzuli postavljamo uslov da obeležje “RBR\_Smena” bude 1 (jer je to prva tj. prepodnevna smena). Konačno, kroz istu klauzulu uslovljavamo da datum mora biti “2023-09-01”.

Pomoću having klauzule, radi sigurnosti, definišemo da vreme završetka poslednjeg termina mora biti manja vrednost od završetka prve smene.

Rezultat upita:



## Peti upit

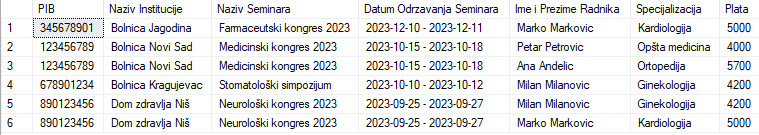
Poslednji upit prikazuje sve institucije koje organizuju seminare, kao i sve radnike sa najvećom platom u svojoj specijalizaciji koji pohađaju taj seminar. Pritom podaci su sortirani po datumu održavanja seminara.

Za svaku pojedinačnu instituciju prikazujemo odgovarajući PIB, naziv institucije, kao i naziv seminara koju ta institucija organizuje. Uz naziv seminara prikazujemo i datum početka i završetka seminara. Konačno upit prikazuje ime i prezime radnika sa najvećom platom u svojoj specijalizaciji i ime njegove specijalizacije.

Kako bismo mogli da pronađemo radnike koji imaju najveću platu u okviru svoje specijalizacije moramo da definišemo SELECT u okviru JOIN klauzule. Praktično definišemo tabelu pod nazivom “speci” koja sadrži JMBG radnika i njegovu platu koju smo imenovali kao promenljivu “maksi”. U okviru tabele speci pronalazimo najvece plate u okviru specijalizacija. Ovo je moguće time što kroz funkciju max() uzimamo isključivo najveću vrednost plate, a pomoću klauzule group by grupišemo po specijalizacijama. Pritom ovu tabelu onda spajamo sa prethodnim tabelama da bismo izdvojili isključivo radnike sa najvećom platom u svojoj specijalizaciji.  
  
Naime, pomoću WHERE klauzule uslovljavamo da je plata radnika mora biti ista kao plata radnika sa najvećom platom u njegovoj specijalizaciji.

Konačno, pomoću ORDER BY klauzule sortiramo dobijene rezultate po datumima.

Rezultat upita:



# FUNKCIJE

## Funkcija FN\_PlatePoSpecijalizaciji

Prva funkcija u okviru projekta jeste funkcija koja za određenu specijalizaciju vraća minimalnu, srednju i maksimalnu platu radnika sa tom specijalizacijom. Ova funkcija bi se koristila radi prikupljanja konkretne statistike o platama u okviru sistema.

Kao ulazni parametar u funkciju definišemo @specijalizacija, što predstavlja naziv specijalizacije koji će korisnik uneti. Nakon toga definišemo tabelu koja funkcija vraća. Tabela će sadržati adekvatne podatke koje želimo da vratimo tj. minimalnu platu, srednju vrednost plate, maksimalnu vrednost plate. To navedemo kroz promenjive “minPlata”, “avgPlata” i “maxPlata”.

Nadalje deklarisemo promenljivu @error tipa int kako bismo kasnije mogli da prikažemo grešku ukoliko korisnik navede nepostojeću specijalizaciju. Upravo nakon toga definišemo IF koji proverava da li je parametar @specijalizacija zapravo postojeća specijalizacija. Ukoliko nije, SUBP pokuša da prikaže promenljivu @error kao varchar, pošto je @error tipa int, SUBP vraća grešku:



Ukoliko je korisnik uneo postojeću funkciju, u tabelu koju će funkcija vratiti kao povratnu vrednost unosimo upravo minimalni, srednju i maksimalnu platu unesene specijalizacije. Te podatke dobijamo spajanjem tabela “Zdravstveni Radnik” i “Licenca”, uz WHERE klauzulu koja uslovljava da je Specijalizacija jednaka sa korisnički navedenim parametrom @specijalizacija.

Nakon preuzimanja adekvatnih podataka, funkcija vraća tabelu:



Gore naveden rezultat sledi iz pozivanja funkcije sa parametrom “Ortopedija”. Razlog zbog kojeg su dobijene gore navedene vrednosti jeste jer se u bazi trenutno nalaze dva radnika sa specijalizacijom Ortopedija:



Kada bismo uneli parametar “Opšta Medicina”, dobili bismo:



Ponovo, rezultat sledi iz toga što imamo dva radnika sa specijalizacijom Opšta Medicina:



## Funkcija FN\_TerminiUSmeni

Za prosledjenu smenu i datum vratiti sve zakazane termine za navedeni broj dana i lekare koji su zakazali te termine

Funkcija FN\_TerminiUSmeni vraća sve zakazane termine u prosleđenoj smeni, sa prosleđenim datumu. Takođe se prosleđuje i broj dana za koje će funkcija prikazati termine.

Dakle, kako bi to bilo moguće, definišu se tri parametra: @rbr\_Smena, @datum i @brojDana. Nakon toga moramo definisati tabelu koju će funkcija vratiti, slično kao u prethodnoj funkciji. Kao povratne parametre u tabeli ispisujemo “ID\_Termin”, “RBR\_Smena”, “VremePocetka”, “VremeZavrsetka”, “Datum” i “ImePrez”. Bitna napomena je da VremePocetka i VremeZavrsetka predstavljaju vreme početka i vreme završetka pronađenog termina. Ostali parametri su poprilično jasnog imenovanja.

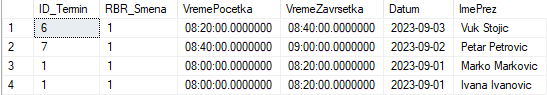
Ulaskom u funkciju deklarišemo promenljivu @error tipa INT da bismo kasnije mogli da vratimo grešku ukoliko korisnik prosledi nepostojeću smenu. Dakle, kao pomenuto nakon toga funkcija proverava postojanje prosleđenog parametra @rbr\_Smena. Ukoliko parametar ne postoji u bazi, vraća se greška:



Nadalje, pomoću WHILE petlje se prolazi kro odgovovarajući broj dana od datuma. WHILE vrši iteracije dokle god važi uslov: “@brojDana > 0”, jer ćemo u samoj petlji nakon svake iteracije smanjivati promenljivu @brojDana za jedan. Unutar WHILE petlje nalazi se INSERT u prethodno definisanu tabelu @returnTable koja je zapravo povratna vrednost funkcije. Pomoću INSERT-a unose se vrednosti “ID\_Termin”, “RBR\_Smena”, “VremePocetka”, “VremeZavrsetka”, “Datum” i “ImePrez” iz tabela “Termin”, “RadniKalenda”, “Lekar” i “Zdravstveni Radnik”. Pritom, definisana je WHILE klauzula kojom uslovljavamo da RBR\_Smena koju izlistavamo mora biti jednaka sa korisnički prosleđenim parametrom @rbr\_Smena. Pored toga, u WHERE klauzuli postavljamo uslov da Datum Termina mora biti jednak datumu dana kojeg trenutno proveravamo.

Pritom, kako bismo dobili efekat prolaženja kroz broj prosleđenih dana, moramo koristiti funkcjiu DATEADD(), kojom na promenljivu @datum kojom označavamo datum koji trenutno proveravamo dodajemo @brojDana – 1. Smanjujemo trenutnu vrednost promenljive @brojDana za jedan jer će korisnik uneti npr. 3 kao broj dana za koji želi da mu funkcija prikaže termine, a mi za prvi dan ne moramo dodavati @brojDana jer je to samo inicijalno prosleđeni datum, inače bismo preskočili prosleđeni datum i ne bismo prikazali termine za taj datum. Time, umanjivanjem vrednosti promenljive @brojDana za jedan zapravo prikazujemo termine za prosleđeni datum i sledeća dva dana (što je ukupno tri).

Za prosleđenu prvu smenu, datum ‘2023-09-01’ i 3 kao broj dana, dobijamo rezultat:



Kao što možemo videti, funkcija je za prosleđeni datum, i sledeća dva dana navela 4 postojeća termina u prvoj smeni, kao i lekare koji su zakazali pomenute termine.

Ukoliko funkciji prosledimo iste parametre, ali 1 kao broj dana, ispisaće nam samo termine u prosleđenom datumu:



# PROCEDURE

## Procedura PR\_ObnovaLicenci

Procedura PR\_ObnovaLicenci proverava svakog radnika i obnavlja im licence ukoliko ispunjavaju određene uslove. Kako bi se radniku obnovila licenca potrebno je da skupi određen broj bodova pohađanjem seminara. U zavisnosti od specijalizacije radnika potreban je različit broj ukupnih bodova kako bi se licenca obnovila.

Dakle, obnavljanje licence realizujemo kroz proceduru na sledeći način. Prvo kreiramo kursor licence\_cursor kojim prolazimo kroz tabele “Licenca” i “Zdravstveni Radnik” i skupljamo podatke o specijalizaciji, datumu obnove i isteka licence, sumi bodova, imenu, prezimenu i JMBG-u radnika.

Nakon toga deklarišemo sve potrebne promenljive u koje ćemo smeštati podatke iz kursora. Ujedno deklarišemo i dodeljujemo vrednost 1 promenljivi @RedniBroj kojom ćemo brojati ukupan broj radnika kojim smo obnovili licence.

Pre nego što uđemo u WHILE petlju kojom ćemo prolaziti kroz tabele koristeći kursor, moramo otvoriti kursor kroz komandu OPEN licence\_cursor. Takođe, uzimamo prve vrednosti iz kursora komandom FETCH NEXT FROM licence\_cursor i smeštamo ih u odgovarajuće promenljive pomoću komande INTO.

Nakon toga zalazimo u WHILE petlju koja će se izvršavati dokle god cursor pronalazi nove podatke u odgovarajućim tabelama, što je omogućeno uslovom WHILE @@FETCH\_STATUS = 0. Kursor će vraćati FETCH\_STATUS sa vrednošću 0 dokle god pronalazi vrednosti u tabeli.

Unutar petlje prvo proveravamo da li je ukupna suma bodova radnika majna od 100 (kroz IF uslov), ukoliko jeste, momentalno prelazimo na sledećeg radnika jer ne postoji specijalizacija koja zahteva manje od 100 bodova kako bi se licenca obnovila. Prelazimo na sledećeg radnika tako što ponovo izvršimo komandu FETCH NEXT FROM licence\_cursor into...Nakon toga se izvršava sledeća iteraciju u petlji zbog navođenja instrukcije CONTINUE, kojom forsiramo petlju da izvrši sledeću iteraciju.

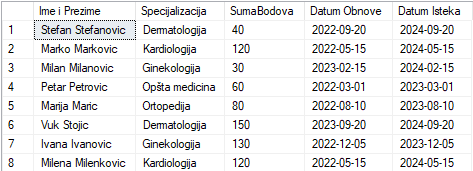
Ukoliko radnik ima više ili tačno 100 bodova, proverava se specijalizacija radnika kroz ELSE IF uslove. Koristimo ELSE IF kako petlja ne bi i nakon pronalaska specijalizacije radnika proveravala ostale IF uslove. Kada se pronađe odgovarajuća specijalizacija radnika, proverava se da li taj radnik poseduje odgovarajuću sumu bodova za tu specijalizaciju (jer svaka specijalizacija sulovljava različit broj bodova). Ukoliko radnik nema dovoljan broj bodova, funkcija izlazi iz uslova IF, pronalazi nove vrednosti u kursoru i prelazi na sledeću iteraciju.

U suprotnom, funkcija pomoću UPDATE-a ažurira odgovarajući red u tabeli Licenca, tako što vraća sumu bodova radnika na 0, postavlja datum obnove na današnji datum i datum isteka licence na datum 2 godine od današnjeg. Momentalno nakon ažuriranja tabele, ispisuje se “Radniku \*Ime i prezime\* je obnovljena licenca” kako bismo znali kojim radnicima je obnovljena licenca. Konačno, inkrementira se redni broj za jedan i prelazi na sledeću iteraciju.

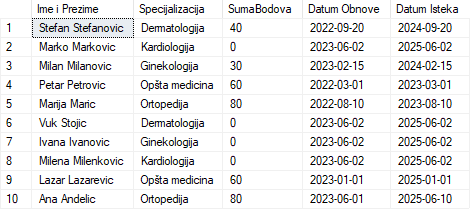
Kada funkcija izvrši svaku iteraciju WHILE petlje, ispisuje se ukupan broj radnika kojima je licenca obnovljenja pomoću promenljive @redniBroj. Bitno je napomenuti da moramo smanjiti vrednost promenjive za jedan jer smo u poslednjoj iteraciji inkrementirali njenu vrednost na jednu više nego što je potrebno (jer se vrednost promenljive inkrementira nekon ispisa radnika kojem smo obnovili licencu).

Nakon ispisivanja pomenute napomene, funkcija zatvara kursor pomoću komande CLOSE licence\_cursor i briše kursor iz memorije pomoću komande DEALLOCATE licence\_cursor.

Dole navedene su vrednosti spojenih tabela “Zdravstveni Radnik” i “Licenca” pre izvršavanja procedure:



Nakon izvršavanja funkcije, možemo videti da se pojedinim radnicima obnovila licenca, postavila suma bodova ponovo na 0 i promenili datumi obnove i isteka licence:



Razlog zbog kojeg je navedenim radnicima obnovljena jeste jer su imali odgovarajući broj bodova, a zahtevi su bili sledeći:

Opšta medicina – 140 bodova

Kardiologija – 110 bodova

Ortopedija – 100 bodova

Dermatologija – 120 bodova

Ginekologija – 100 bodova

## Procedura PR\_MenjanjeBodovaSeminara

Procedura PR\_MenjanjeBodovaSeminara oduzima ili dodaje određen broj bodova svim seminarima određene institucije, pod uslovom da seminar nije započet. Dakle, kao ulazne parametre definisani su nam broj bodova koji dodajemo ili oduzimamo od seminara (unosi se negativan broj ukoliko se oduzima) i naziv institucije čije seminare želimo da ažuriramo.

Ulaskom u proceduru nailazimo na IF uslov koji proverava da li je prosleđeni naziv institucije postojeći, ukoliko nije, procedura vraća grešku:



Grešku vraćamo pomoću funkcije RAISEERROR(), kojoj prosleđujemo ispis greške, nivo greške (level) i stanje (state). Za korisničke greške je standard postavljati nivo na 16.

Nadalje, ukoliko je procedura prošla prvi IF uslov, pomoću ELSE-a počinjemo sa izvršavanjem odgovarajućeg procesa. Prvo su deklarisane promenljive @id\_Seminar, @datumPocetkaSem i @brojBodovaSeminara kako bismo iz kursora kojeg ćemo koristiti mogli da smestimo povratne vrednosti. Iz toga razloga definiše se kursor kojim prikupljamo id seminara i datum početka seminara prolaskom kroz sve seminare koje je navedena institucija organizovala. To je moguće pristupanjem tabela “Seminar” i “Institucija”, a kako bismo utvrdili koja institucija je organizovala koje seminare, moramo zaći i u tabelu “Organizuje” koja predstavlja tabelu poveznika prethodno dva pomenuta tipa entiteta. Na kraju deklarisanja kursora pomoću WHERE klauzule uslovljavamo da naziv institucije mora biti baš jednak parametru koji je prosleđen od strane korisnika.

Nakon deklaracija, kursor se otvara komandom OPEN seminariInstitucije\_cursor, i preuzimaju se i smeštaju prve vrednosti iz odgovarajućih tabela kursorom. To radimo komandom FETCH NEXT FROM seminariInstitucije\_cursor INTO...

Nakon toga, slično kao u prethodnoj proceduri pomoću WHILE petlje preuzimamo i smeštamo vrednosti iz kursora dokle god kursor pronalazi nove vrednosti u odgovarajućim tabelama. Uslovljavamo to proveravanjem da li je @@FETCH\_STATUS = 0. Kao što je ranije napomenuto, FETCH\_STATUS će nam vratiti nula samo ukoliko je kursor pronašao novu torku u tabeli.

Unutar WHILE petlje pomoću IF uslova proveravamo da li je seminar na kojem je kursor trenutno već počeo. To je moguće poređenjem vrenosti datuma pocetka seminara sa današnjim datumom. Današnji datum dobijamo ugrađenom funkcijom GETDATE() koja nam vraća trenutni datum u momentu izvršavanja funkcije.

Ukoliko seminar nije već započet, proveravamo da li ćemo nakon oduzimanja broja bodova dobiti negativnu vrednost bodova seminara. Ukoliko bismo dobili negativnu vrednost, postavljamo broj bodova na 0 u odgovarajućem seminaru. U suprotnom, oduzmemo/dodatmo odgovarajući broj bodova.

Dakle, pomoću UPDATE komande ažuriramo vrednosti torke u tabeli “Seminar” tako što sabiramo trenutni broj bodova odgovarajućeg seminara sa brojem bodova prosleđenim od strane korisnika. Ponovo, ukoliko je prosleđena negativna vrednost, sabiranje će postati oduzimanje i oduzeće se odgovarajući broj bodova. Pronalazimo odgovarajući seminar WHERE klauzulom koja uslovljava da je ID\_Seminara baš jednak vrednosti promenljive @id\_seminar (promenljiva u koju smeštamo vrednost pomoću kursora).

Nakon ažuriranje torke u tabeli, putem FETCH NEXT FROM seminariInstitucije\_cursor se pronalazi sledeća odgovarajuća torka i smeštaju sledeće vrednosti u promenljive @brojBodova i @nazivInstitucije.

Napokon, nakon svih iteracija WHILE petlje, procedura zatvara kursor komandom CLOSE seminariInstitucije\_cursor, i briše isti kursor iz memorije putem DEALLOCATE seminariInstitucije\_cursor.

Ukoliko izvršimo proceduru sa parametrima –3 i “Apoteka Beograd”, dobićemo sledeće rezultate:

Vrednosti seminara pre izvršavanja procedure:



Vrednosti seminara nakon izvršavanja procedure:



Ukoliko ponovo izvršimo proceduru sa istim parametrima (-3 i “Apoteka Beograd”), videćemo da će procedura postaviti broj bodova na 0 umesto –1:



# TRIGERI

## Triger TR\_RadniStaz

Triger TR\_RadniStaz jer triger koji se okida nakon INSERT i UPDATE naredbe. Ovaj triger nakon okidanja izračuna radni staž radnika na osnovu unetih podataka. Oblast izvršavanja trigera jeste tabela “ZdravstveniRadnik” u okviru koje se nalaze apsolutno svi radnici, bili to lekari ili medicinske sestre.

Za početak, unutar trigera se deklariše promenljiva @radnikJMBG, čiju vrednost momentalno određujemo preuzimanjem vrednosti JMBG-a radnika iz tabele INSERTED. Znajući da će u se u slučaju INSERT-a ova vrednost zasigurno nalaziti u INSERTED tabeli, a i da će ažurirana vrednos nakon izvršavanja UPDATE komande takođe pronaći u ovoj tabeli, ne moramo proveravati o kojoj komandi reč pre određivanja vrednosti tog obeležja. Na identičan način deklarišemo i definišemo vrednost promenljive @stariDatumZaposljavanja.

Nakon toga do duše, moramo proveriti o kojoj komandi je tačno reč. To radimo pomoću IF uslova koji proverava da li ćemo dobiti vrednosti (IF EXISTS) iz SELECT upita koji preuzima sve vrednosti iz deleted tabele. Ukoliko postoje torke u deleted tabeli, znači da je reč o UPDATE-u, i onda zalazimo u IF.

Nakon prvog IF uslova, deklarišemo promenljivu @stariDatumZaposljavanja kojoj pridružujemo vrednost datum zapošljavanja iz tabele DELETED. Ovo radimo kako bismo mogli sledećim IF uslovom uporediti vrednosti promenljivih @stariDatumZaposljavanja i @noviDatumZaposljavanja radi potvrde da je datum zapošljavanja promenjen prilikom ažuriranja torke. Ukoliko su vrednosti promenljivih iste, pomoću komande RETURN izlazimo iz procedure.

U suprotnom, tj. ukoliko je datum zaposljavanja radnika izmenjen, onda proveravamo da li je mesec trenutnog datuma skoriji od novog meseca datuma zapošljavanja radnika. To proveravamo poređenjem rezultata dva izvršavanja DATEPART() funkcije. U prvom izvršavanju prosleđujemo funkciji month kao prvi parametar, kako bi uzimala samo mesec datuma. Kao drugi parametar navodima GETDATE() funkciju koja će preuzeti datum momenta izvršavanja trigera. Time izdvajamo isključivo mesec iz današnjeg datuma. Pritom dobiju vrednost prve DATEPART() funkcije poredimo sa vrednošću druge DATEPART() funkcije kojoj takođe kao prvi parametar prosleđujemo month. Međutim, kao drugi parametar joj prosleđujemo promenljivu @noviDatumZaposljavanja.

Ukoliko je današnji mesec skoriji od novog datuma zapošljavanja, tada ćemo radni staz odgovarajućeg radnika (dobijeno kroz poređenje vrednosti deklarisane promenljive JMBG-a i JMBG-a radnika u WHERE klauzuli) promeniti. Ažuriraćemo radni staž pomenutog radnika korišćenjem UPDATE komande, gde ćemo definisati da je nova vrednost obeležja “RadniStaz” u odgovarajućoj torci jednaka razlici u godinama između današnjeg datuma i novog datuma zapošljavanja radnika – 1. Oduzimamo jednu godinu upravo jer to što je današnji mesec skoriji od meseca zapošljavanja znači da “godišnjica” zapošljavanja radnika i dalje nije prošla, pa se ova godina ne može uračunati u radni staž radnika. Razliku između naših datuma dobijamo korišćenjem ugrađene funkcije DATEDIFF() kojoj prosleđujemo parametre year, @noviDatumZaposljavanja i @GETDATE(). Prvim parametrom razjašnjavamo da želimo da nam funkcija vrati razliku u godinama, dok druga dva parametra označavaju između kojih vrednosti računamo razliku.

Nadalje, ako današnji mesec nije skoriji od novog meseca zapošljavanja radnika, proveravamo da li su isti. Vršimo potrebnu proveru koristeći ELSE IF uslov (time zalazimo u ovaj uslov isključivo ukoliko prošli uslov nije ispunjen) kojim na sličan način (koristeći DATEPART() funkciju) utvrđujemo da li su meseci isti. U okviru istog ELSE IF uslova proveravamo (korišćenjem logičkog AND uslova) da li takođe važi da je današnji dan skoriji od novog dana zapošljavanja radnika. Proveru, ponovo vršimo pomoću funkcije GETDATE(), ali joj kao prvi parametar navodimo “day” kako bismo izdvojili isključivo dane naših datuma.

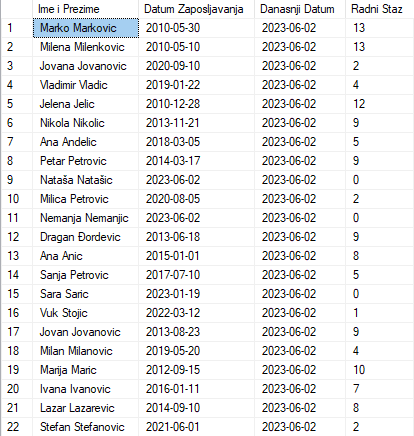
Važenjem pomenutog ELSE IF uslova ponovo smo potvrdili da se mora oduzeti jedna godina prilikom ažuriranja radnog staža radnika, pa to i činimo na isti način kao i prethodnog puta.

Međutim, ukoliko prethodni uslov nije zadovoljen, zalazimo u finalnu ELSE granu, gde ažuriramo radnikov radni staž BEZ oduzimanja jedne godine prilikom korišćenja DATEDIFF() funkcije.

Ukoliko se triger ne izvršava nakon UPDATE-a, već nakon INSERT-a nove torke u tabelu “ZdravstveniRadnik”, preskačemo sve prethodno i zalazimo u ELSE grananje prvog IF uslova (gde smo proverili da li postoji torka u tabeli DELETED). Naime, ukoliko prvobitni IF uslov nije ispunjen, vršimo apsolutno iste provere i ažuriranje radnog staza kao i prilikom potvrde da je promenjena vrednost datuma zaposljavanja prilikom UPDATE-a.

NAPOMENA: Radi provere trigera, bitno je kreirati sam triger pre vršenja INSERT naredbi u tabele, kako bi se triger zaista okidao prilikom unošenja novih vrednosti u tabelu “Zdravstveni Radnik”.

Radi provere nakon unošenja velikog broja torki, dobijamo sledeće rezultate:



Funkcionalnost trigera moguće je proveriti u okviru gore navedenih rezultata, jer su navedeni i datum zapošljavanja radnika, kao i današnji datum.

## Triger TR\_SumaBodova

Triger TR\_SumaBodova kreiran je sa namerom da se nakon brisanja nekog seminara (npr. nakon njegovog održavanja) svim radnicima koji su prisustvovali seminaru dodele odgovarajući bodovi. Ujedno se i u tabelu “IstorijatSeminara” unosi obrisani seminar, kako bi se čuvale informacije o prethodno održanim seminarima. Navedeni triger je INSTEAD OF DELETE triger, što znači da se izvršava umesto komande DELETE. Takođe, oblast izvršavanja trigera jeste tabela “Seminar”.

Sa zalaskom u funkcionalnost trigera, prvo vidimo da se deklarišu promenljive u koje smeštamo sve potrebne podatke: @ID\_Seminar, @ID\_IstorijatSeminara, @BrojBodova, @TemaSeminara, @NazivSeminara, @DatumPocetkaSem, @DatumZavrsetkaSem, kao i @JMBG\_ZdravRad. Nakon deklaracija promenljivih, postavljamo njihove vrednosti uzimajući podatke iz tabele DELETED. Možemo uvek pronaći potrebne vrednosti u ovoj tabeli jer će se ovaj triger izvršavati samo umesto DELETE komande. Jedinu vrednost koju eksplicitno definišemo jeste vrednost promenljive @ID\_IstorijatSeminara.

Kako bismo postavili vrednost pomenute promenljive, prvo proveravamo da li uopšte postoji bilo kakva torka u tabeli IstorijatSeminara, korišćenjem SELECT upita unutar IF EXISTS uslova. Kada potvrdimo da postoji torka u ovoj tabeli, postavljamo vrednost promenljive @ID\_IstorijatSeminara na trenutnu najveću vrednost identifikacione vrednosti istorijata seminara, inkrementirano za 1. Ukoliko ne postoji ni jedna torka u tabeli IstorijatSeminara, postovljamo vrednost promenljive na 1.

Nakon uspešnog definisanja svih promenljivih, deklarišemo novi kursor pod imenom “radniciSaSeminara\_cursor” kojim prolazimo kroz sve radnike koji su pohađali upravo taj seminar. To postižemo SELECT upitom koji preuzima vrednost JMBG-a radnika iz pridruženih (join-ovanih) tabela “ZdravstveniRadnik” i “Pohađa” uz WHERE klauzulom koja uslovljava da ID pohađanog seminara mora biti baš jednaka prethodno definisanoj promenljivi @ID\_Seminar.

Naredni korak jeste otvaranje kursor kroz komandu OPEN radniciSaSeminara\_cursor, kao i preuzimanja vrednosti sledeće torke kursorom koristeći FETCH NEXT FROM radniciSaSeminara\_cursor i unošenja tih vrednosti u našu promenljivu pomoću INTO @JMBG\_ZdravRad.

Sledi kreiranje WHILE petlje koju izvršavamo dokle god kursor pronalazi nove torke, pomoću uslova WHILE FETCH\_STATUS = 0. FETCH\_STATUS će biti 0 dokle god kursor pronalazi nove torke.

Unutar WHILE petlje ispisujemo kojem radniku dodajemo bodove i koliko bodova mu dodajemo koristeći komandu PRINT. Nakon toga ažuriramo odgovarajuću torku u tabeli “Licenca” koristeći UPDATE komandu u okviru kojeg staru sumu bodova inkrementiramo sa brojem bodova dobijenih iz seminara (što je skladišteno u promenljivoj @BrojBodova). Pomoću WHERE klauzule sa uslovom da je JMBG radnika baš vrednost promenljive @JMBG\_ZdravRad koju dobijamo kursorom pronalazimo odgovarajućeg radnika. Posle ažuriranja odgovarajućih vrednosti preuzimamo sledeću torku iz kursora pomoću FETCH NEXT FROM radniciSaSeminara\_cursor i prelazimo u sledeću iteraciju petlje.

Nakon izvršavanja svih iteracija petlje tj. prolaska kroz sve radnike koji su pohađali seminar koji se briše, zatvaramo kursor kroz komandu CLOSE radniciSaSeminara\_cursor. Zatvaranju kursora sledi i brisanje kursora iz memorije, pa se time izvršava i komanda DEALLOCATE radniciSaSeminara\_cursor. Posle brisanja kursora unosimo odgovarajuće vrednosti u tabelu “IstorijatSeminara” pomoću INSERT komande sa svim prethodno preuzetim vrednostima (iz tabele DELETED). Sledeći korak jeste da obrišemo sve podatke vezane za pomenuti seminar, pa time brišemo torku iz tabele “Pohađa” gde je ID seminara bas vrednost promenljive @ID\_Seminar (takođe iz tabele DELETED). Na isti način brišemo torke iz tabela “Organizuje” i “Seminar”.

Nakon brisanja seminara sa ID-jem 3 iz tabele “Seminar”, dobijamo sledeće rezultate:

Suma bodova radnika koji pohađaju pomenuti seminar pre brisanja seminara:



Suma bodova radnika koji pohađaju pomenuti seminar nakon brisanja seminara:



Vrednosti u tabeli “IstorijatSeminara” nakon brisanja seminara:



U tabeli iznad možemo i potvrditi da je broj bodova koji je dodat radniku korektan, jer je broj bodova koji seminar nosi baš 7.

# ZAKLJUČAK

Radom na ovom projektu shvatio sam važnost kreiranja adekvatnih podšema relacija za veće sisteme i stekao uvid u proces kreiranja baze podataka koristeći samo prethodno kreiranu podšemu. Ovaj projekat mi je ukazao na to koliko je kompleksno kreirati potpuno funkcionalnu i smislenu bazu podataka, po definisanim korisničkim uslovima. Ipak, nakon dosta rada i truda, projekat je potpun, i ja sam zadovoljan konačnim proizvodom.