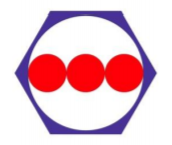
****

**Univerzitet u Novom Sadu**

**Fakultet tehničkih nauka**

**Departman za industrijsko inženjerstvo i inženjerski menadžment**

**Inženjerstvo informacionih sistema**

**INFORMACIONI SISTEM ZA PODRŠKU UPRAVLJANJA KORISNIČKIM SADRŽAJEM TIPA USTANOVA ZA BRIGU O LICIMA SA POSEBNIM POTREBAMA**

**-SISTEMI BAZA PODATAKA**

**Marina Cvijetić IT60g2019**

**Novi Sad, 2022.**

**Sadržaj**

[UVOD 1](#_Toc105149514)

[ANALIZA PROGRAMSKOG DOMENA 1](#_Toc105149515)

[ER MODEL 3](#_Toc105149516)

[ER MODEL PODŠEME 4](#_Toc105149517)

[TABELARNI PRIKAZ OBELEŽJA I OGRANIČENJA 5](#_Toc105149518)

[RELACIONI MODEL 13](#_Toc105149519)

[DDL 14](#_Toc105149520)

[DML 17](#_Toc105149521)

[SQL 20](#_Toc105149522)

[OBJEKTI 21](#_Toc105149523)

[Šema 21](#_Toc105149524)

[Tabele 21](#_Toc105149525)

[Procedure 24](#_Toc105149526)

[Funkcije 24](#_Toc105149527)

[Trigeri 25](#_Toc105149528)

[UPITI 25](#_Toc105149529)

[Prvi upit 25](#_Toc105149530)

[Drugi upit 26](#_Toc105149531)

[Treći upit 27](#_Toc105149532)

[Četvrti upit 27](#_Toc105149533)

[Peti upit 28](#_Toc105149534)

[FUNKCIJE 28](#_Toc105149535)

[PROCEDURE 31](#_Toc105149536)

[TRIGERI 34](#_Toc105149537)

[ZAKLJUČAK 38](#_Toc105149538)

# UVOD

U okviru ove dokumentacije, opisan je projekat razvijen za podršku rada i funkcionisanja jednog segmenta informacionog sistema ustanove za brigu o licima sa posebnim potrebama (UBLPP). Primarno, projekat obuhvata razvoj baze podataka koja ima ulogu da korisnicima, uz odgovarajući interfejs, olakša i ubrza upravljanje podacima neophodnim za poslovanje i funkcionisanje dela sistema koji se odnosi na vršenje zdravstvenih pregleda u okviru ustanove, kao i sve druge aktivnosti koje su u bliskoj vezi sa zdravstvenim pregledom ili nastaju kao posledica/rezultat aktivnosti pregleda.

U nastavku teksta, izvršena je analiza programskog domena kako bi se korisniku približile i konciznije predočile moguće funkcionalnosti sistema, prikazan je ER model podataka celokupne šeme sistema, kao i ER model podataka podšeme koja je tema ovog projekta. U dokumentaciji je takođe priložen i tabelarni prikaz ograničenja i obeležja, ali i opis svih narednih koraka prilikom izrade projekta hronološkim redom.

# ANALIZA PROGRAMSKOG DOMENA

Cilj projekta jeste da obezbedi lakše i brže poslovanje, izvršavanje poslovnih aktivnosti i zadataka, a sve to je moguće samo uz informacije i podatke koje kruže u okviru realnog sistema UBLPP. Kako bi cilj bio postignut, neophodno je razviti bazu podataka čija je uloga da organizovano čuva podatke i omogući prenos i upravljanje istim, ali i komunikaciju sa korisnikom.

Baza podataka mora biti konzistentna i pouzdana, podatke na nivou korisnika je potrebno strukturirati, a na fizičkom nivou organizovati tako da njihovo održavanje bude najlakše, a operisanje njima najefikasnije. Pored važnosti neometane upotrebe, neophodno je obezbediti i operacije dodavanja novih podataka, brisanja, modifikovanja postojećih podataka i čitanje, odnosno uvid u sve ili filtrirane podatke.

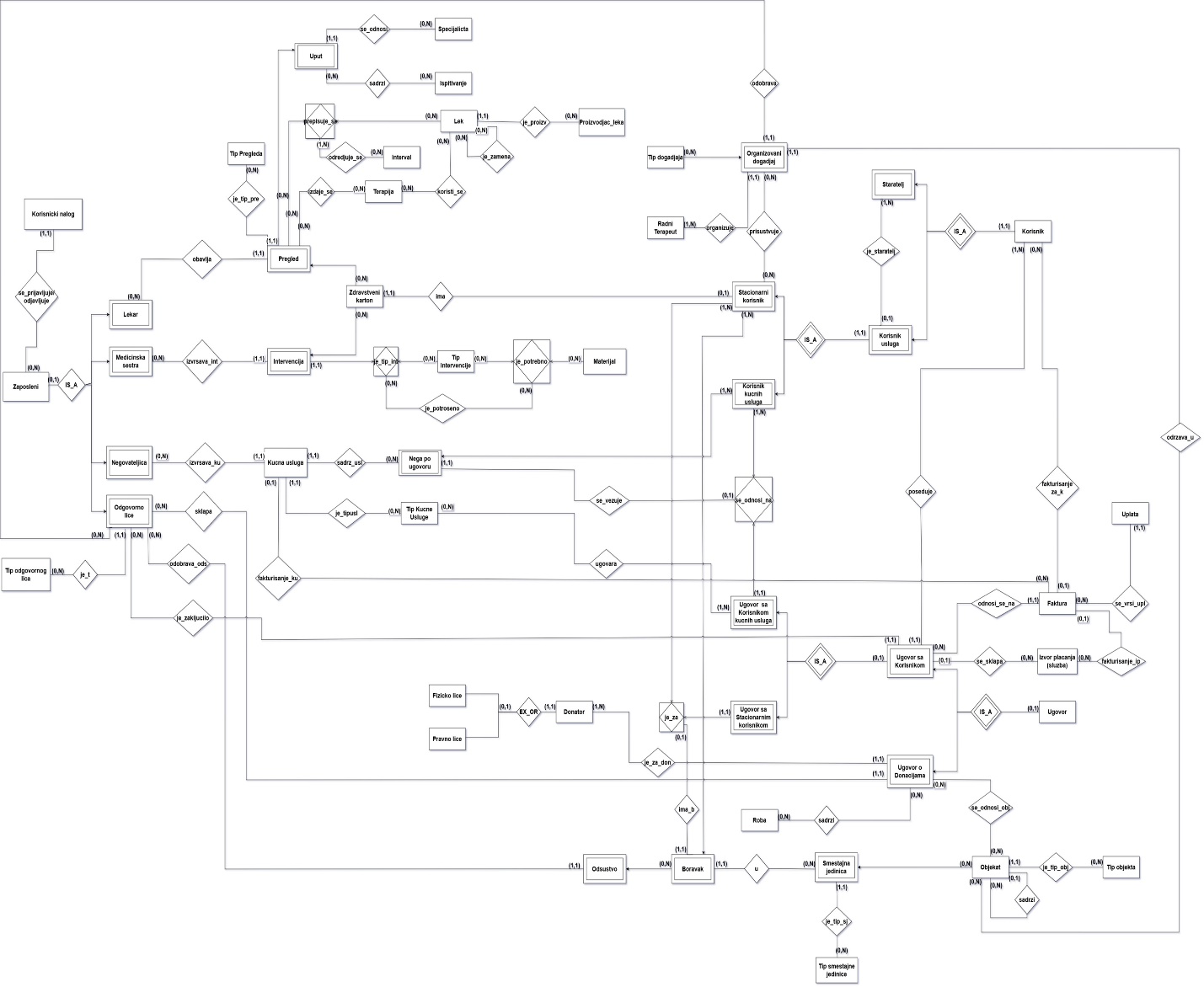
Obezbeđena je fleksibilna struktura baze podataka koja je prilagodljiva i primenjiva u sistemima iste oblasti poput gerontoloških centara ili domova za stare i bolesne. U skladu sa zahtevima teme projekta vodi se evidencija o pregledima koji se vrše u okviru ustanove nad korisnicima usluge boravka u istoj. Pegledi mogu biti različitog tipa, poput redovnog pregleda ili sistematskog pregleda.

Neophodna je evidencija o korisnicima, kao i o njihovim zdravstvenim kartonima, jer svaki korisnik koji boravi u ustanovi mora da ima svoj zdravstveni karton. Vrlo važan podatak koji se beleži jeste identitet lekara (zaposlenog) koji vrši pregled. U zavisnosti od ishoda pregleda, moguće je prepisati terapiju za korisnika tokom koje može da se koristi lek, ali lek može biti propisan i nezavisno od terapije. Kada se lek propisuje nezavisno od terapije, za njega mora biti izdat recept u okviru kog se, između ostalog, beleži uputstvo za upotrebu leka. Pored evidencije lekova, postoji i evidencija za lekove koji su zamena nekim drugim lekovima, kao i evidencija o proizvođaču leka. Korisniku na pregledu može da se izda uput za dalje analize kod lekara specijaliste ili laboratorijsko ispitivanje ukoliko lekar koji vrši zdravstveni pregled opšte prakse u okviru ustanove zaključi da je to neophodno.

Navedene evidencije podataka su od krucijalne važnosti za neometano funkcionisanje dela sistema koji se odnosi na aktivnosti pregleda, a samim tim i celokupni sistem ustanove. U skladu sa potrebama sistema, strukturu ove baze podataka je moguće proširivati i prilagođavati.

# ER MODEL

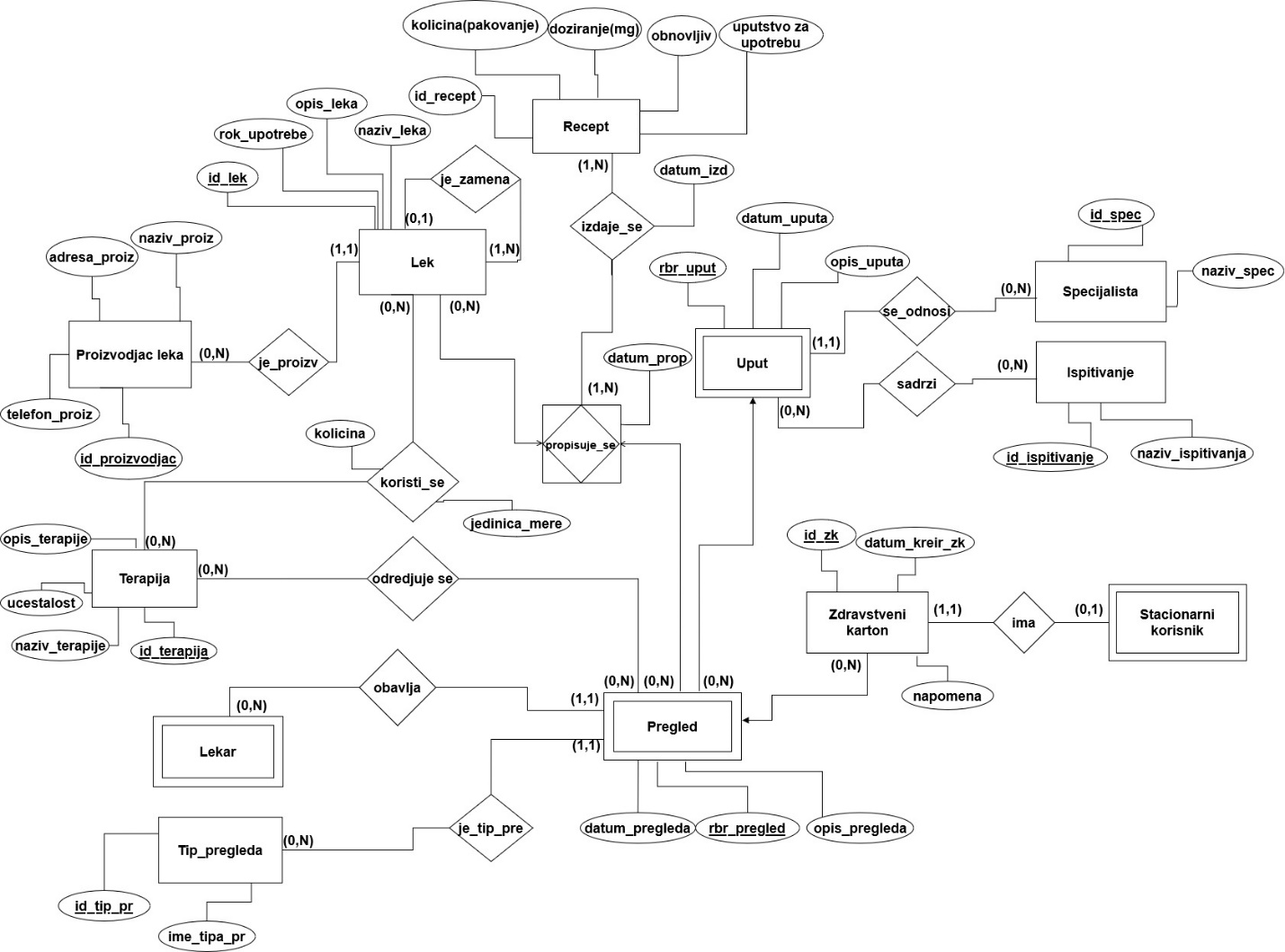
Prikaz ER modela sistema ustanove za brigu o licima sa posebnim potrebama. Tokom detaljnijeg proučavanja celokupne šeme ER modela u svrhu daljeg razvoja ovog projekta, uočeno je da trenutno rešenje (tip entiteta *„Interval“*) za beleženje podataka čija je uloga da opišu način na koji se lek konzumira, nije dovoljno precizno i ne ispunjava kriterijum jasne reprezentacije tih podataka.



Slika 1. ER Dijagram sistema ustanove za brigu o licima sa posebnim potrebama

# ER MODEL PODŠEME

Prikaz ER modela podšeme „*Pregled“* koja predstavlja predmet projekta. Glavni fokus je na strukturi i organizaciji podataka koji se čuvaju, a u vezi su sa aktivnosti vršenja zdravstvenog pregleda. Promena koja je usledila na podšemi, kao rešenje prethodno opisanog problema, je uvođenje novog tipa entiteta *„Recept“.* Recept mora da sadrži podatke o lekaru koji je izdao recept, za koga je recept namenjen, maksimalnu dnevnu dozu izraženu u miligramima, podatak o tome da li je recept obnovljiv i najvažnije – uputstvo za upotrebu propisanog leka koje predstavlja opisno obeležje. Takođe je izmenjen kardinalitet u vezi između tipa entiteta *„Lek“* i tipa poveznika *„je\_zamena“* tako da za lek može da postoji najviše jedna zamena, ali i ne mora. Lek koji je zamena, može da bude zamena više lekova, ali mora biti zamena najmanje jednom leku.



Slika . ER Dijagram podšeme “Pregled”

# TABELARNI PRIKAZ OBELEŽJA I OGRANIČENJA

Tabela 1. STACIONARNIKORISNIK

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Naziv obeležja | Opis obeležja | Tip podatka | Dužina podatka | *NULL* | Uslov |
| **id\_sk** | Identifikaciona šifra korisnika | INTEGER | 10 | 丄 | Δ |
| **ime\_sk** | Ime korisnika | VARCHAR | 20 | 丄 | Δ |
| **prz\_sk** | Prezime korisnika | VARCHAR | 30 | 丄 | Δ |
| **adresa\_sk** | Adresa korisnika | VARCHAR | 100 | 丄 | Δ |
| **grad\_sk** | Grad | VARCHAR | 30 | 丄 | Δ |
| **telefon \_sk** | Broj telefona | VARCHAR | 10 | 丄 | Δ |
| **datum\_rodj** | Datum rodjenja | DATE | Δ | 丄 | Δ |
| **KLJUČ: K = {id\_k}** | | | | | |
|  | | | | | |

Tabela 2.1. ZDRAVSTVENIKARTON

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Naziv obeležja | Opis obeležja | Tip podatka | Dužina podatka | *NULL* | Uslov |
| **id\_zk** | Identifikaciona šifra zdravstvenog kartona | INTEGER | 10 | 丄 | Δ |
| **datum\_kreiranja** | Datum kada je zdravstveni karton kreiran | DATE | Δ | 丄 | Δ |
| **napomena** | Odeljak za posebne napomene | VARCHAR | 100 | T | Δ |
| **KLJUČ: K = {id\_zk}** | | | | | |
|  | | | | | |

Tabela 2.2 PREGLED

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Naziv obeležja | Opis obeležja | Tip podatka | Dužina podatka | *NULL* | Uslov |
| **rbr\_pregled** | Redni broj pregleda | INTEGER | 10 | 丄 | Δ |
| **datum\_pregleda** | Datum kada je izvršen pregled | DATE | Δ | 丄 | Δ |
| **opis** | Opis pregleda | VARCHAR | 100 | T | Δ |
| **KLJUČ: K = {id\_zk + rbr\_pregled}** | | | | | |
|  | | | | | |

Tabela 2.3 UPUT

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Naziv obeležja | Opis obeležja | Tip podatka | Dužina podatka | *NULL* | Uslov |
| **rbr\_uput** | Redni broj uputa | INTEGER | 10 | 丄 | Δ |
| **datum\_izdavanja** | Datum kada je uput izdat | DATE | Δ | 丄 | Δ |
| **opis** | Opis uputa | VARCHAR | 100 | T | Δ |
| **KLJUČ: K = {id\_zk + rbr\_pregled+rbr\_uput}** | | | | | |
|  | | | | | |

Tabela 3. SPECIJALISTA

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Naziv obeležja | Opis obeležja | Tip podatka | Dužina podatka | *NULL* | Uslov |
| **id\_spec** | Identifikaciona šifra specijalizacije | INTEGER | 10 | 丄 | Δ |
| **naziv\_spec** | Naziv specijalizacije | VARCHAR | 30 | 丄 | Δ |
| **KLJUČ: K = {id\_spec }** | | | | | |
|  | | | | | |

Tabela 4. ISPITIVANJE

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Naziv obeležja | Opis obeležja | Tip podatka | Dužina podatka | *NULL* | Uslov |
| **id\_ispitivanje** | Identifikaciona šifra laboratorijskog ispitivanja | INTEGER | 10 | 丄 | Δ |
| **naziv\_isp** | Naziv laboratorijskog ispitivanja | VARCHAR | 30 | 丄 | Δ |
| **opis** | Opis | VARCHAR | 100 | T | Δ |
| **KLJUČ: K = {id\_ispitivanje }** | | | | | |
|  | | | | | |

Tabela 5. TIP PREGLEDA

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Naziv obeležja | Opis obeležja | Tip podatka | Dužina podatka | *NULL* | Uslov |
| **id\_tipa\_pr** | Identifikaciona šifra tipa pregleda | INTEGER | 10 | 丄 | Δ |
| **naziv\_tipa** | Naziv tipa pregleda | VARCHAR | 30 | 丄 | Δ |
| **KLJUČ: K = {id\_tipa\_pr}** | | | | | |
|  | | | | | |

Tabela 6. LEKAR

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Naziv obeležja | Opis obeležja | Tip podatka | Dužina podatka | *NULL* | Uslov |
| **id\_lekar** | Identifikaciona šifra lekara | INTEGER | 10 | 丄 | Δ |
| **ime\_ lekar** | Ime lekara | VARCHAR | 30 | 丄 | Δ |
| **prz\_ lekar** | Prezime lekara | VARCHAR | 30 | 丄 | Δ |
| **adresa\_ lekar** | Adresa lekara | VARCHAR | 50 | 丄 | Δ |
| **br\_tel\_ lekar** | Broj telefona lekara | VARCHAR | 10 | 丄 | Δ |
| **dat\_zavrs\_sk** | Datum završetka školovanja | DATE | Δ | 丄 | Δ |
| **rad\_staz** | Godine radnog staža lekara | INTEGER | 2 | 丄 | d>0 |
| **stepen\_str\_spr** | Stepen stručne spreme | VARCHAR | 20 | 丄 | Δ |
| **specijalizacija** | Naziv oblasti specijalizacije | VARCHAR | 40 | 丄 | Δ |
| **datum\_spec** | Datum specijalizacije | DATE | Δ | 丄 | Δ |
| **KLJUČ: K = {id\_lekar }** | | | | | |
|  | | | | | |

Tabela 7. TERAPIJA

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Naziv obeležja | Opis obeležja | Tip podatka | Dužina podatka | *NULL* | Uslov |
| **id\_terapija** | Identifikaciona šifra terapije | INTEGER | 10 | 丄 | Δ |
| **naziv\_terap** | Naziv terapije | VARCHAR | 30 | 丄 | Δ |
| **ucestalost\_ned** | Učestalost terapije na nedeljnom nivou | INTEGER | 1 | 丄 | 0<d<8 |
| **opis** | Opis terapije | VARCHAR | 100 | T | Δ |
| **KLJUČ: K = {id\_terapija }** | | | | | |
|  | | | | | |

Tabela 8. PROIZVOĐAČ LEKA

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Naziv obeležja | Opis obeležja | Tip podatka | Dužina podatka | *NULL* | Uslov |
| **id\_proizvodjac** | Identifikaciona šifra proizvođača | INTEGER | 10 | 丄 | Δ |
| **naziv\_proiz** | Naziv proizvođača | VARCHAR | 30 | 丄 | Δ |
| **adresa\_proiz** | Adresa proizvođača | VARCHAR | 50 | 丄 | Δ |
| **telefon\_proiz** | Telefon proizvođača | VARCHAR | 10 | 丄 | Δ |
| **KLJUČ: K = {id\_proizvodjac }** | | | | | |
|  | | | | | |

Tabela 9. LEK

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Naziv obeležja | Opis obeležja | Tip podatka | Dužina podatka | *NULL* | Uslov |
| **id\_lek** | Identifikaciona šifra leka | INTEGER | 10 | 丄 | Δ |
| **naziv\_leka** | Naziv leka | VARCHAR | 30 | 丄 | Δ |
| **rok\_upotrebe** | Datum isteka roka | DATE | Δ | 丄 | Δ |
| **opis\_leka** | Opis leka | VARCHAR | 100 | 丄 | Δ |
| **KLJUČ: K = {id\_lek }** | | | | | |
|  | | | | | |

Tabela 10. KORISTI SE – tip poveznika

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Naziv obeležja | Opis obeležja | Tip podatka | Dužina podatka | *NULL* | Uslov |
| **kolicina** | Doza leka koji se koristi u terapiji | NUMERIC | (4,2) | 丄 | Δ |
| **jedinica\_mere** | Jedinica mere u kojoj se izražava doza leka | VARCHAR | 10 | 丄 | Δ |
| **KLJUČ: K = {id\_terapija + id\_lek}** | | | | | |
|  | | | | | |

Tabela 11. PROPISUJE SE– gerund

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Naziv obeležja | Opis obeležja | Tip podatka | Dužina podatka | *NULL* | Uslov |
| **datum\_prop** | Datum kada je lek propisan | DATE | Δ | 丄 | Δ |
| **KLJUČ: K = {id\_zk + rbr\_pregled + id\_lek}** | | | | | |
|  | | | | | |

Tabela 12. RECEPT

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Naziv obeležja | Opis obeležja | Tip podatka | Dužina podatka | *NULL* | Uslov |
| **id\_recept** | Identifikaciona šifra recepta | INTEGER | 10 | 丄 | Δ |
| **kolicina\_pak** | Kolicina leka izražena u pakovanju | INTEGER | 2 | 丄 | Δ |
| **doziranje\_mg** | Maksimalna dnevna doza izražena u miligramima | NUMERIC | (4,2) | 丄 | Δ |
| **obnovljiv** | Recept može ponovo da se iskoristi | BIT | Δ | 丄 | Δ |
| **uputstvo\_za**  **upotrebu** | Uputstvo za upotrebu propisanog leka | VARCHAR | 100 | 丄 | Δ |
| **KLJUČ: K = {id\_recept }** | | | | | |
|  | | | | | |

Tabela 13. IZDAJE SE

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Naziv obeležja | Opis obeležja | Tip podatka | Dužina podatka | *NULL* | Uslov |
| **datum\_izd** | Datum kada je recept izdat | DATE | Δ | 丄 | Δ |
| **KLJUČ: K = {id\_zk + rbr\_pregled + id\_lek + id\_recept}** | | | | | |
|  | | | | | |

# RELACIONI MODEL

Relacione baze podataka su danas izuzetno zastupljene u informacionim sistemima. Zasnovane su na relacionom modelu podataka gde je struktura veoma jednostavna i prihvatljiva korisniku jer se baza predstavlja kao skup tabela. Moguća je formalno-matematička interpretacija modela, odnosno definisanje formalnog „jezika“ baze i optimizacija performansi.

Cela struktura, u kojoj se podaci konceptualno predstavljaju organizovani u redove i kolone, naziva se relacija, a svaki red podataka se zove n-torka.

Ukupan broj n-torki u relaciji određuje kardinalnost relacije, svaka kolona n-torke predstavlja atribut, dok ukupan broj atributa određuje stepen relacije.

Relacija nije uređena i redosled vrsta i kolona nije značajan. Ipak, nije svaka tabela relacija. Da bi tabela bila relacija mora da zadovolji kriterijume koji zahtevaju da ne postoje duplikati redova tabele i da nisu dozvoljeni atributi sa ponavljanjem (*Prva normalna forma*).

Vrlo važan pojam u relacionom modelu podataka je ključ. Ključ predstavlja atribut ili grupu atributa koji jedinstveno identifikuju jednu n-torku u relaciji i mora da zadovoljava osobinu jedinstvenosti i osobinu neredudantnosti. U praktičnoj primeni

bira se jedna kombinacija atributa ili jedan atribut kao primarni ključ. Primarni ključ može biti i strani ključ u nekoj drugoj tabeli. Na taj način, strani ključ reprezentuje one n-torke koje on jedinstveno identifikuje u nekoj drugoj tabeli čiji su podaci međusobno povezani.

Pravila integriteta definišu dozvoljena stanja i dozvoljene prelaze sistema iz stanje u stanje, koje uslove podaci u bazi podataka treba da zadovolje, kada se vrši provera, akcije kada uslovi nisu zadovoljeni. Definišu se za osnovne operacija ažuriranja i treba da budu podržana od strane samog sistema za upravljanje bazom podataka.

Ograničenje vrednosti atributa predstavlja definiciju domena, odnosno dozvoljene vrednosti za taj atribut.

Ograničenje integriteta entiteta podrazumeva da nijedan atribut koji je primarni ključ ili deo primarnog ključa ne može da uzme *NULL* vrednost.

Ograničenje strukturnog (referencijalnog) integriteta je jedan od najvažnijih koncepata relacionog modela i obezbeđuje korektno povezivanje objekata u relacionom modelu. Ono zahteva da strani ključ u nekoj tabeli mora uzimati neke od postojećih vrednosti primarnog ključa one tabele iz koje taj strani ključ dolazi ili *NULL* vrednost ukoliko je dozvoljena.

Transakcija je atomski skup aktivnosti koji se izvršava po principu „sve ili ništa“. Predstavlja jedno izvršenje logičke jedinice rada nad bazom podataka i jedno je od ključnih koncepata relacionih sistema za upravljanje bazom podataka.

Pravila integriteta domena definišu vrednosti pojedinih atributa nezavisno od ostaih atributa u bazi podataka. Mogu se formulisati relativno jednostavnim logičkim izrazima.

Pravila integriteta relacija definišu uslove pod kojima zapis može da ažurira relaciju i zavisnost n-torki jedne relacije od n-torki druge relacije. Primer takvih ograničenja su *TRIGGER* ograničenja ili *CHECK CONSTRAINT* ograničenja koja će biti detaljnije objašnjena kasnije u dokumentaciji.

# DDL

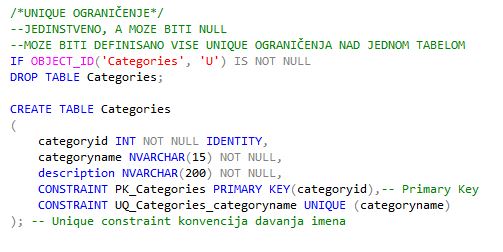
Data Definition Language (DDL) naredbe služe za kreiranje i modifikaciju objekata baze podataka, odnosno koristi se za određivanje strukture baze podataka. DDL takođe definiše dodatna svojstva podataka u bazi i pruža mogućnost određivanja nekih ograničenja.

Naredbe DDL-a su:

1. CREATE
2. ALTER
3. DROP
4. TRUNCATE
5. RENAME

Naredba *CREATE* se koristi za kreiranje novih tabela, definisanje kolona, ograničenja koja se odnose na kolone i kreiranje relacija. Ova naredba se na isti način može koristiti u svim relacionim sistemima za upravljanje bazom podataka, razlike postoje samo u tipovima podataka koji se koriste za opis kolona tabele.

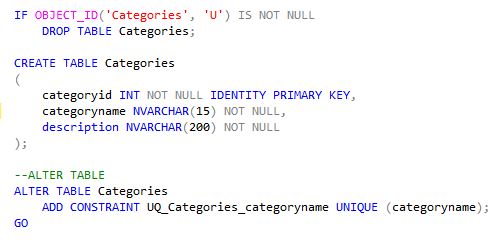
*CREATE* *TABLE* naredba ima pet tipova ograničenja, a to su *PRIMARY KEY, UNIQUE, NULL/NOT NULL, FOREIGN KEY* i *CHECK* ograničenje.



Slika . Naredba CREATE TABLE

U prvom delu naredbe, svaka kolona je definisana svojim imenom, tipom podatka i null statusom odnosno ograničenjem koje nam govori da li vrednost tog atributa može biti izostavljena. U drugom delu naredbe se navode ograničenja – definisan primarni ključ i kandidat, odnosno alternativni ključ. Primarna svrha alternativnog ključa je da obezbedi jedinstvenost vrednosti neke kolone definisanjem ograničenja *UNIQUE*. (*Slika br. 3*)

Naredba *ALTER TABLE* se koristi kada želimo da promenimo strukturu postojeće tabele. Može se koristiti za dodavanje, izbacivanje ili modifikaciju kolona kao i za dodavanje i uklanjanje ograničenja nad tabelama. (*Slika br. 4*)



Slika . Naredba ALTER TABLE

Ukoliko želimo da obrišemo definiciju objekta iz baze, zajedno sa podacima koje sadrži, koristi se naredba *DROP*. Razlika između *DELETE* i *DROP* je što *DELETE* naredbom može da se obriše kompletan sadržaj tabele, ali sama tabela ostaje u bazi podataka i sa njom se može dalje raditi. Naredbom *DROP* *TABLE*, tabela sa sadržajem se izbacuje iz baze podataka i više nije dostupna.

Brisanje nije moguće ukoliko tabela sadrži ograničenje stranog ključa, odnosno *FOREIGN* *KEY* *CONSTRAINT*. Brisanje se može izvršiti samo ako tabela nema decu ili ako je naznačeno kaskadno brisanje kao što je prikazano u primeru. To znači da ukoliko se obriše tabela *Products,* brišu se svi podaci i u drugim tabelama koji dolaze iz tabele ,,*Products”.* (*Slika br. 5*)



Slika . Naredba DROP TABLE

*TRUNCATE* naredba briše sve podatke iz tabele odjednom. (*Slika br. 6*)



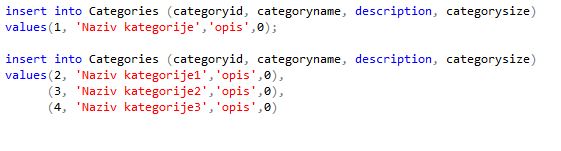
Slika . Naredba TRUNCATE

# DML

Data Manipulation Language (DML) predstavlja jezik za manipulaciju podacima u bazi podataka i DML naredbe omogućuju ažuriranje u širem smislu i izveštavanje o stanju podataka u bazi. Ažuriranje u širem smislu obuhvata dodavanje, izmenu sadržaja i brisanje reda ili redova tabele. Te operacije se realizuju SQL naredbama:

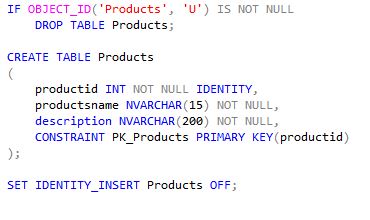
1. INSERT
2. UPDATE
3. DELETE
4. SELECT

*INSERT* naredba se koristi za dodavanje redova tabele. Standardna verzija naredbe *INSERT* je ime tabele, imena kolona u koje se unose podaci i lista podataka kao na primeru. Ukoliko se eksplicitno ne navedu kolone u koje se upisuju podaci, mora se ispoštovati redosled obeležja kao prilikom kreiranja tabele. Obeležja za koja postoji ograničenje *NOT NULL* zahtevaju navedenu vrednost prilikom unosa podataka. (*Slika br.7*)



Slika . Naredba INSERT

Ukoliko u tabeli postoji ograničenje *IDENTITY* za neki atribut, unos vrednosti za to obeležje nije moguće ukoliko se prethodno to ograničenje ne isključi kao što je prikazano u primeru. (*Slika br. 8*)



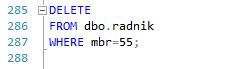
Slika . Isključivanje ograničenja IDENTITY

*UPDATE* naredba služi za izmenu sadržaja tabele. Opšti oblik naredbe je *UPDATE* je naziv tabele koja se ažurira, navođenje kolona (atributa) i novih vrednosti za iste iza klauzule *SET.* U okviru *WHERE* klauzule je moguće postaviti određeni uslov, na osnovu kog filtriramo podatke koje želimo da menjamo. (*Slika br. 9*)



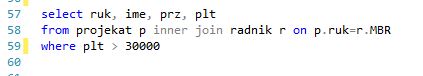
Slika . Naredba UPDATE

*DELETE* naredba se koristi za brisanje sadržaja dela ili cele tabele. Standardna verzija *DELETE* upita je navođenje tabele iz koje se brišu zapisi nakon *FROM* odredbe koji zadovoljavaju odredbu *WHERE* gde se definiše uslov*.* *DELETE* se koristi ako se potencira na brisanju više zapisa. (*Slika br. 10*)

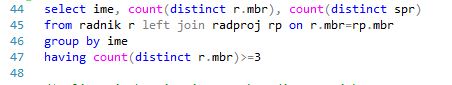


Slika . Naredba DELETE

*SELECT* naredba služi za čitanje i prikaz podataka iz baze. Listom atributa zadaje se operacija projekcije, a kvalifikacionim izrazom se zadaju restrikcije (selekcije) i spajanja, odnsono iskazi slični iskazika u relacionom računu. Klauzule *SELECT* i *FROM* su obavezne, dok klauzula *WHERE* nije. U okviru *SELECT* naredbe je moguća upotreba dodatnih klauzula kao što su *GROUP BY* za grupisanje prema određenom obeležju, *HAVING* za postavljanje uslova grupnih funkcija, *ORDER*  *BY* za sortiranje. (*Slika br. 11 i 12*)



Slika . Naredba SELECT bez dodatnih klauzula



Slika . Naredba SELECT sa dodatnim klauzulama

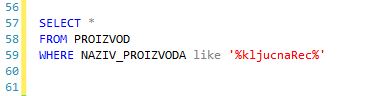
# SQL

Structured Query Language (SQL) je strukturalni upitni jezik koji se koristi za izdavanje komandi za manipulaciju podacima baze podataka, kao i za kompletno njeno kreiranje, izmenu ili brisanje, kako same baze i njenih tabela, tako i podataka koji se nalaze u njoj. SQL ima tri glavne komponente: Jezik za manipulaciju podataka (DML), Jezik za definisanje podataka (DDL) i Jezik za kontrolu podataka (DCL).

Većina ljudi zapravo koristi SQL a da toga nije svesna. Dinamička web stranica koja je bazirana na bazi podataka, uzima korisničke podatke iz formulara i klikova i koristi je za sastavljanje SQL upita koji preuzima informacije iz baze podataka potrebne za generisanje sledeće web stranice.

Primer može biti jednostavan online katalog sa funkcijom pretraživanja. Stranica za pretragu može se sastojati od obrazaca koji sadrže samo polje za tekst u koje unosite pojam za pretragu, a zatim kliknete na pretragu. Tada web server preuzima bilo koji zapis iz baze podataka proizvoda koji sadrži izraz za pretragu i koristi rezultate za kreiranje web stranice specifične za konkretan zahtev.

U pozadini bi taj upit izgledao kao u sledećem primeru gde se u uslovu navodi ključna reč između dva ampersand znaka. Uslov zahteva izlistavanje svih proizvoda u čijem je nazivu sadržana ključna reč. (*Slika br. 13*)

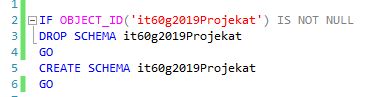


Slika . Primer upita za pretragu

# OBJEKTI

### Šema

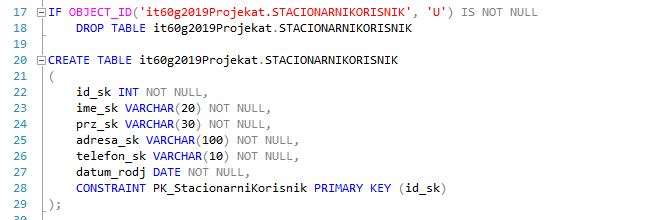
Na samom početku izrade projekta, kreirana je šema it60g2019Projekat u okviru koje projekat nastavlja da se razvija. Šema je lista logičkih struktura podataka. Šema je pojedinačni entitet i predstavlja kontejner objekata. Sintaksa za kreiranje šeme je trivijalna. Pre kreiranja bilo kog objekt, dobra praksa je proveriti da li već postoji kreiran isti objekat u bazi. (*Slika br. 14*)



Slika . Naredba CREATE SCHEMA

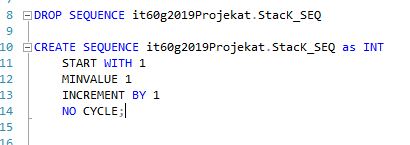
### Tabele

Nakon što je kreirana šema koja će sadržati sve ostale objekte projekta, kreirane su tabele na osnovu prevedenog ER modela podataka podšeme. Rezultat prevođenja ER modela podšeme je sedamnaest tabela. Tabele je poželjno kreirati počevši od onih tabela koje su najjednostavnije i nemaju strane ključeve, kako bi se izbegle greške. Tako je u ovom slučaju prva kreirana tabela bila ,,*StacionarniKorisnik”* koja ne sadrži ni jedno ograničenje stranog ključa. (*Slika br. 15*)



Slika . Tabela STACIONARNIKORISNIK

Za tabelu ,,*StacionarniKorisnik”* je prethodno kreirana sekvenca, kao i za tabele ,,*ZdravstveniKarton”* i ,,*Lek”.* Sekvenca je korisnički definisan i nezavisan objekat koji generiše niz numeričkih vrednosti prema specifikaciji sa kojom je sekvenca kreirana. Niz numeričkih vrednosti se generiše u rastućem ili opadajućem redosledu u definisanom interval i može se konfigurisati da se ponovo pokrene ispočetka. Sekvenca je implementirana da generiše identifikacione vrednosti za navedene tabele. (*Slika br. 16 i 17*)



Slika . Kreiranje sekvence



Slika . Primena sekvence

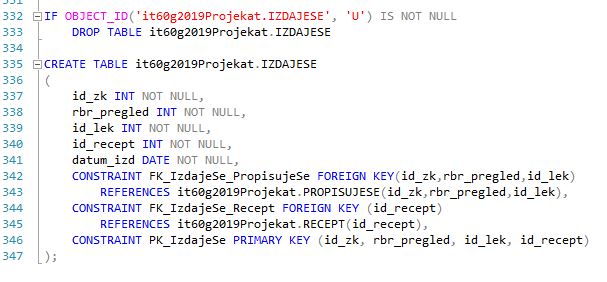
Važno je objasniti strukturu složenih tabela, te će fokus biti na tabelama koje su prilikom prevođenja ER modela podšeme, kao posledica maksimalnih kardinaliteta, rezultirale u vidu novih šema relacija čiji su primarni ključevi kombinacije ključeva iz različitih tabela.

Tabela ,,*Sadrzi”* je nastala od istoimenog tipa poveznika, u vezi ,,*Uput – Ispitivanje”* koja ukazuje na to da uputmože da sadrži više ispitivanja. Tabela *Sadrzi* sadrži podatke o tome koja ispitivanja su sadržana u uputu koji je izdat korisniku.

Tabela ,,*OdredjujeSe”* je nastala kao rezultat potpuno identične veze onoj iz prethodnog primera, ali je u pitanju veza između tipova entiteta ,,*Terapija – Pregled”­*. Tabela ,,*OdredjujeSe”* sadrži podatke kojim korisnicima je određena terapija na pregledu i koja terapija je u pitanju.

Tabela *KoristiSe* takođe nastaje na isti način kao i prethodne dve tabele, gde postoji veza između tipova entiteta ,,*Terapija – Lek”.* U okviru ove tabele, sadržane su informacije o tome koji lek može da se koristi i na kojoj terapiji, al i koja je maksimalna količina leka izražena u miligramima.

Tabela ,,*PropisujeSe”* služi kako bi znali koji su lekovi propisani korisnicima na pregledu, ako su propisani, i od strane kog lekara. Tabela ,,*PropisujeSe”* uslovljava nastanak nove tabele ,,*IzdajeSe”,* jer za svaki propisan lek na pregledu, neophodno je izdati recept za isti. ,,*IzdajeSe”* sadrzi evidenciju recepata za propisane lekove na pregledu. (*Slika br. 18*)



Slika . Primer složene tabele

Kreirane tabele je neophodno popuniti podacima relevantnim za informacioni sistem i aktivnost pregleda. (*Skriptovi sa DDL i DML naredbama su predati uz projekat)*

Nakon kreiranja tabela i njihovog popunjavanja podacima, sledi implementacija jedinica za višekratnu upotrebu koje obuhvataju specifičnu poslovnu logiku aplikacije.

### Procedure

Procedura koja je prva kreirana je ,,*KorisnikUputProc”*, za prosleđeni ID korisnika, vraća listu ispitivanja koje sadrži uput koji je izdat tom korisniku na pregledu. Prilikom kreiranja procedure, primenjen je rad sa parametrima čime je omogućeno lakše čuvanje potrebnih vrednosti, IF-ELSE grananja i kursor koji prolazi kroz listu ispitivanja sadržanih u uputu i prikazuje ih redom.

Druga procedura je malo složenije strukture jer sadrži dva kursora, pri čemu je jedan kursor ugnježden u drugi. Razlog implementacije ovakvog rešenja u proceduru ,,*LekarKorisnik”* je da se za prosleđeni ID lekara najpre izlistaju korisnici koje je taj lekar pregledao, a zatim da se izlistaju i lekovi koje im je taj lekar propisao, ukoliko jeste. Ukoliko lekovi nisu propisani na pregledu, korisniku se prikazuje poruka koja ga o tome obaveštava. (*Skript za kreiranje procedura je predate uz projekat – SQL PROCEDURE.sql*)

### Funkcije

Funkcija ,,*KorisniciNaTerapiji”* je funkcija koja za prosleđeni ID terapije vraća prikaz podataka u vidu tabele, o korisnicima kojima je određena ta terapija kao i o broju terapija na nedeljnom nivou. Funkcija je prilično jednostavne strukture. Nakon navedemo kakav rezultat funkcija vraća i kod tipa, definišemo strukturu tabele koja je rezultat funkcije, a zatim je popunjavamo klasičnim *SELECT* upitom, čiji navedeni atributi moraju da se poklapaju sa atributima definisanim u tabeli u okviru koje čuvamo rezultat. Funkcija se poziva putem *SELECT* upita, gde se u *FROM* odredbi poziva funkcija.

Funkcija ,,*GodineDoSpec”* računa za prosleđeni ID lekara broj godina od završetka škole, do specijalizacije. Funkcija je skalarna, odnosno vraća jednu vrednost koja je tipa integer*.* Broj godina se računa putem funkcije *DATEDIFF,* koja u zavisnosti od prvog parametra, računa razliku u danima/mesecima/godinama između dva prosleđena datuma. (*Skript za kreiranje funkcija je predate uz projekat – SQL FUNKCIJE.sql*)

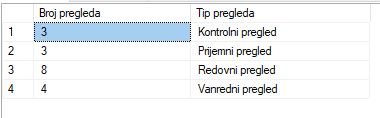
### Trigeri

Triger ,,*RokLeka”* kreiran na tabelom ,,*Lek”* obezbeđuje ograničenje koje ne dozvoljava da se u bazu podataka evidentira lek čiji je rok upotrebe istekao. Triger se pokreće u trenutku pokretanja upita ažuriranja ili dodavanja nove torke, gde se vrednost obeležja za rok upotrebedodaje u sklopu nove n-torke, ili se ažurira. Proverava se nova vrednost koja se unosi, iz tabele *,,inserted”* i poredi se sa trenutnim datumom. Unos je dozvoljen samo ako je datum koji se unosi ili menja veći od trenutnog. U suprotnom se podiže greška.

# UPITI

### Prvi upit

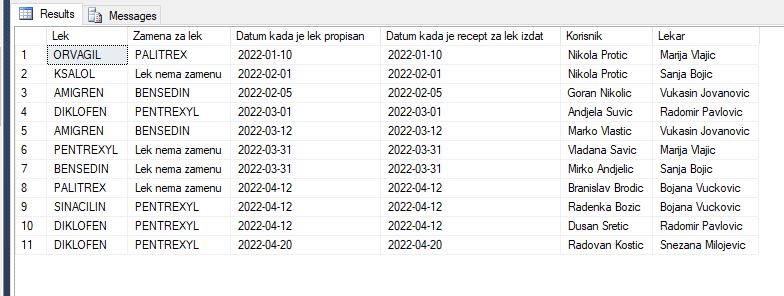
Prvi upit prikazuje broj pregleda koji su izvršeni u period od 01.01.2022. do 01.05.2022. uključujući i te datume. Broj pregleda je grupisan prema tipu pregleda, pri čemu su prikazani samo tipovi pregleda čiji je broj veći od nule. Upotrebljeni su parametri u kojima se čuvaju datumi upotrebljeni za uslov i agregatna funkcija *COUNT* uz pomoć koje se pregledi prebrojavaju. U okviru *FROM* odredbe, spojene su tabele ,,*Pregled”* i *,,TipPregleda”* kako bi se prikazao naziv tipa pregleda. Rezultat upita je tabela sa dve kolone. (*Slika br. 19*)



Slika . Rezultat prvog upita

### Drugi upit

Drugi SQL upit prikazuje lekove koji su propisani korisnicima na pregledu, naziv zamene za propisan lek ako postoji. Ukoliko ne postoji zamena za lek, ispisuje se poruka da lek nema zamenu što se realizuje putem *IIF* funkcije koja sadrži kao parametre uslov, parametar koji se ispisuje u slučaju da je rezultat uslova istinit, i parametar koji se ispisuje u slučaju da je uslov netačan. Prikazuje se datum kada je lek propisan, datum kada je izdat recept za lek, kome je lek propisan i od strane kog lekara. Podaci su sortirani rastuće prema datumu izdavanja recepta. U okviru odredbe *FROM* bilo je neophodno spojiti čak osam tabela kako bi se prikazali svi traženi podaci. (*Slika br. 20*)



Slika . Rezultat drugog upita

### Treći upit

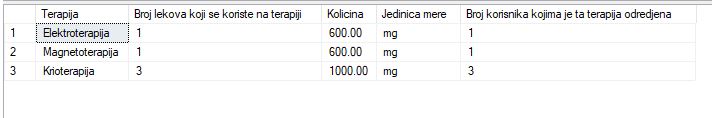
U trećem upitu, rezultat je tabela sa korisnicima kojima je prepisana terapija čije je učestalost na nedeljnom nivou veća ili jednaka od prosečne, naziv terapije i ukupan broj terapija koje su određene tom korisniku. Prosečna učestalost terapija je sačuvana u parametru čija se vrednost podesi pomoću upita uz upotrebu agregatne funkcije *AVG* za računanje prosečne vrednosti. Broj terapija koje su određene korisniku je implementiran pomoću podupita u odredbi *FROM*. U okviru podupita prebrojane su terapije za svaki zdravstveni karton, odnosno korisnika. (*Slika br. 22*)



Slika . Rezultat trećeg upita

### Četvrti upit

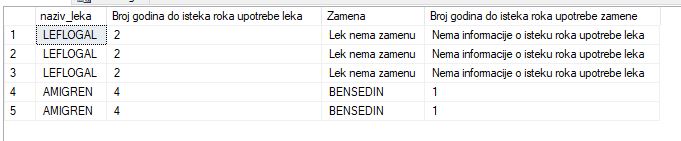
Peti upit prikazuje terapije na kojima se koristi više od jednog leka, koje sadrže slovo O u svom nazivu i gde se koristi veća ili jednaka količina leka od prosečne količine, samo za one lekove čija je jedinica mere izražena u miligramima. Prikazan je i broj korisnika kojima su te terapije određene. Prosečna količina leka je sačuvana u okviru parametra, a zatim upotrebljena u uslovu. Za pretragu po ključnoj reči sadržanoj u nekom stringu, primenjen je uslov sa ključnom reči izmedju dva ampersand znaka. (*Slika br. 22*)



Slika . Rezultat četvrtog upita

### Peti upit

Peti i najsloženiji upit sadrži dosta ugrađenih SQL funkcija. Upit rezultira tabelom sa podacima o nazivu leka, broju godina do isteka roka upotrebe leka, naziv zamene za lek ukoliko postoji, kao i broj godina do isteka roka upotrebe leka koji je zamena. Prilikom računanja broja godina do isteka roka, primenjena je funkcija *DATEDIFF* koja je već objašnjena ranije u dokumentaciji. Zatim se pomoću *IIF* funkcije proverava nulabilnost obeležja zamene leka i ukoliko postoji zamena, rčuna se broj godina do isteka roka, ukoliko ne postoji, ispisuje se poruka da nema informacija o zameni leka. Lekovi su filtrirani na osnovu proizvođača leka. (*Slika br. 23*)

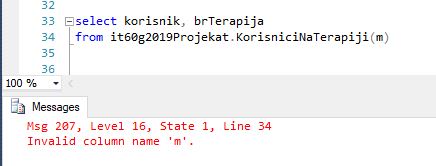


Slika . Rezultat petog upita

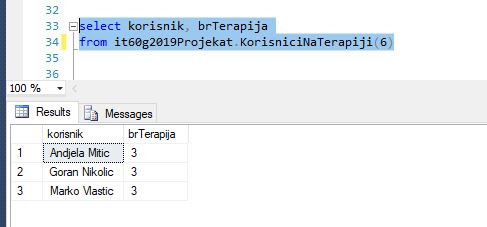
# FUNKCIJE

U funkciji *,,KorisniciNaTerapiji”* ulazni parametar je tipa int i predstavlja ID terapije. Na osnovu prosleđenog ID-ja terapije, funkcija treba da vrati tabelu korisnika kojima je određena ta terapija. Kako bi vratila tabelu, neophodno je eksplicitno definisati strukturu tabele koja je prethodno označena kao povratna vrednost funkcije. Struktura tabele koja se vraća kao rezultat definiše se na isti način kao i sve tabele do sad – navođenjem naziva kolona i njihovih tipova podataka. U okviru tela funkcije, opet se eksplicitno popunjava se tabela navođenjem naredbe *INSERT*, koja se vraća kao rezultat. U upitu kojim se popunjava teba, redosled atributa kao i tipovi podataka, moraju da se poklapaju, odnosno da budu identični. Isto važi i za upit kojim se funkcija poziva. U odredbi *FROM* prva tabela koja je navedena je ,,*OdredjujeSe”.* Kombinacija primarnog ključa ove tabele je redni broj pregleda na kom je određena terapija, ID zdravstvenog kartona korisnika kome je određena terapija i ID same terapije. Tabela ,,*OdredjujeSe”* je tabela iz koje se preko zdravstvenog kartona dolazi do imena i prezimena korisnika, preko pregleda do doktora koji je odredio terapiju, i do naziva terapije preko ID-ja terapije.

Ono što može da izazove grešku prilikom pozivanja funkcije, jeste da se u pozivu funkcije prosledi vrednost koja nije numerička. Ukoliko je prosleđena vrednost koja ne postoji u bazi, rezultat je prazna tabela. (*Slika br. 24 i 25*)



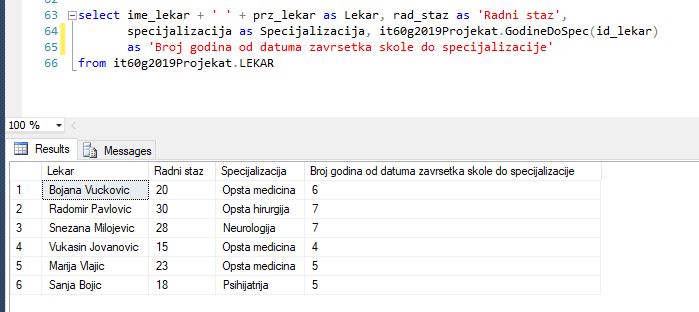
Slika . Rezultat funkcije sa izazvanom greškom



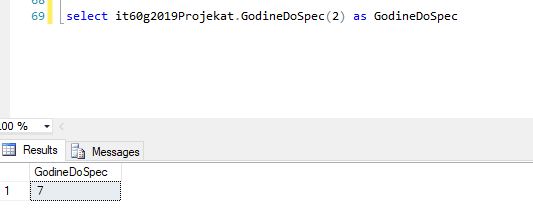
Slika . Rezultat funkcije bez greške

Funkcija *,,GodineDoSpec”* za prosleđeni ID lekara računa koliko godina je prošlo od trenutka završetka škole do specijalizacije. Ulazni parametar koji funkcija prihvata i na osnovu kog vraća rezultat jeste ID lekara koji je tipa integer. U telu funkcije je deklarisan parametar u kom se čuva rezultat, i još dva parametra za datum završetka škole i datum specijalizacije. Uz pomoć funkcije *DATEDIFF* računamo razliku u godinama između dva prosleđena datuma i rezultat se smešta u parametarza rezultat. Rezultat se navodi u *RETURN* naredbikoja prekida izvršenje funkcije i ispisuje rezultat.

Upit je iskorišćen u upitu nad bazom, kako bi se demonstrirala prava svrha funkcije čija je uloga da ubrza proces kodiranja i da se izbegne ponavljanje istog koda, a može da se iskoristi u više različitih slučajeva. Grešku u funkciji se dešava ukoliko se kao ulazni parametar prosledi vrednost koja nije numerička. Rezultat pozivanja funkcije u upitu je prikazan na sledećem primeru. (*Slika br. 25*)



Slika . Rezultat poziva funkcije u upitu

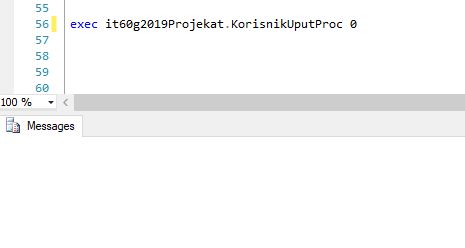


Slika . Poziv funkcije izvan upita

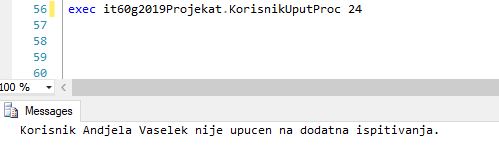
# PROCEDURE

Procedura *,,KorisnikUputProc”* kao ulazni parametar prihvata ID stacionarnog korisnika, a kao rezultat treba da izlista ispitivanja koja sadrži njegov uput. Broj ispitivanja se prvo prebrojava i na osnovu njega se kreira uslov grananja koji proverava da li korisnikov uput sadrži bilo kakva ispitivanja. Ukoliko sadrži, kreira se kursor koji se popunjava filtriranom tabelom definisanom u *SELECT* upitu, koja sadrži listu ispitivanja na koje je upućen taj korisnik. Prolaskom kroz kursor, preuzimaju se vrednosti jedna po jedna i printa se kao poruka korisniku. Ukoliko korisnik nije upućen ni na jedno dodatno ispitivanje, prikazuje se poruka koja o tome obaveštava korisnika. (*Slika br. 29*)

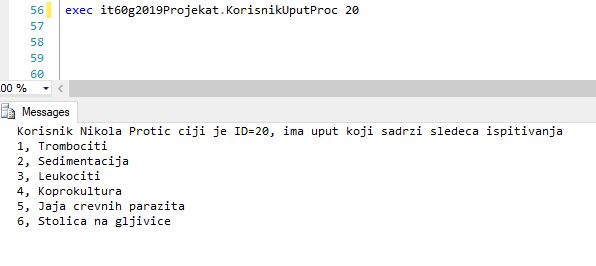
Ukoliko se proceduri prosledi vrednost koja ne postoji u bazi, rezultat izvršavanja procedure je prazan. (*Slika br. 28*)



Slika . Rezultat procedure sa prosleđenom nepostojećom vrednosti



Slika . Rezultat procedure ukoliko korisnik nema dodatna ispitivanja

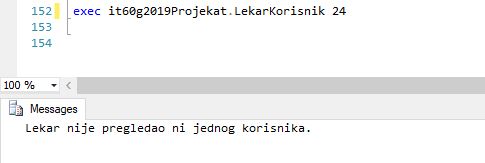


Slika . Rezultat procedure ukoliko uput sadrži listu ispitivanja

Procedura *,,LekarKorisnik”* kao ulazni parametar prihvata ID lekara, a kao rezultat prvo prikazuje listu korisnika koje je taj lekar pregledao, a zatim i za svakog korisnika listu lekova koje im je taj lekar propisao, ukoliko jeste. Ako lekovi nisu propisani, to je naznačeno.

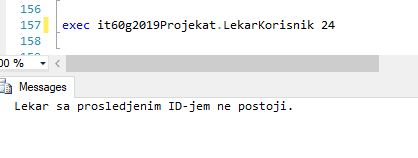
Prva provera se vrši na osnovu prebrojanih korisnika i proverava da li je lekar pregledao bar jednog korisnika, ukoliko jeste kreira se kursor koji se popunjava podacima o korisnicima koji su pregledani od strane lekara čiji je ID prosleđen. U okviru kursora za korisnike, nakon izlistanog imena i prezimena korisnika, proverava se da li za njega postoje propisani lekovi. Ukoliko postoje, u okviru otvorenog kursora *,,KorisnikCur”* otvara se još jedan kursor *,,PropisaniLekoviCur”* koji se popunjava listom lekova koji su propisani korisniku, a zatim se lekovi izlistavaju ispod imena i prezimena korisnika. Ukoliko korisnik nema propisane lekove, ispisuje se poruka koja ukazuje na to.

Prikaz rezultata poziva procedure kojoj je prosleđen ID lekara koji nije pregledao ni jednog pacijenta. (*Slika br. 31*)



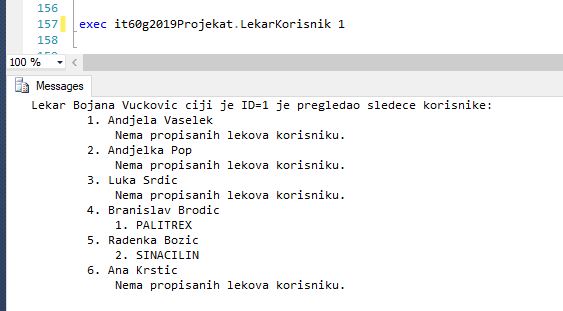
Slika . Rezultat procedure – 1. Slučaj

Rezultat procedure ukoliko je prosleđen nepostojeći ID lekara. (*Slika br. 32*)



Slika . Rezultat procedure – 2. Slučaj

Rezultat procedure ukoliko postoji i lista korisnika i lista propisanih lekova za te korisnike. (*Slika br. 33*)



Slika . Rezultat procedure – 3. Slučaj

# TRIGERI

Trigrer *,,RokLeka”* obezbeđuje ograničenje nad tabelom lek i obeležjem koje definiše datum isteka roka leka. Ograničenje se odnosi na operacije *INSERT* i *UPDATE.* Ukoliko se pokuša unos n-torke u tabelu *,,Lek”*, gde je vrednost obeležja za datum isteka roka datum koji je prošao ili današnji datum, unos nije dozvoljen i podiže se *Error* sa porukom koja obaveštava korisnika o grešci. Za potrebe provere datuma, deklarišu se parametri u okviru kojih se čuva stara vrednost datuma isteka roka iz tabele *,,Deleted”* ukoliko je u pitanju operacija ažuriranja, nova vrednost obeležja preuzeta iz tabele *,,Inserted”* i trenutni datum.

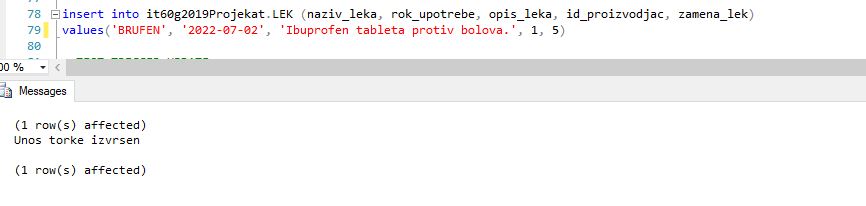
Najpre se proverava da li je nad konkretnim obeležjem registrovana bilo kakva promena ili pokušaj promene vrednosti uz pomoć funkcije *UPDATE().* Ukoliko jeste, sledeća provera služi da se utvrdi da li je u pitanju *INSERT* ili *UPDATE* operacija. Ako i u ,,*Inserted”* tabeli postoji nova vrednost obeležja koja je preuzeta, i u *,,Deleted”* tabeli postoji stara vrednost obeležja, u pitanju je ažuriranje i izmena postojećeg stanja. U tom slučaju sledeća provera utvrđuje da li nova vrednost koja se unosi ispunjava uslove, odnosno veća je od trenutnog datuma. Ako vrednost ne ispunjava uslove, generiše se greška sa porukom da nije moguće uneti n-torku. Ukoliko vrednost ispunjava uslov, a u pitanju je ažuriranje, vrši se *UPDATE* i nova vrednost datuma je postavljena.

Ukoliko u tabeli *,,Deleted”* ne postoji stara vrednost obeležja, a u tabeli *,,Inserted”* postoji nova vrednost, jasno je da je u pitanju *INSERT* operacija i unos nove n-torke.

Takođe se proverava vrednost nove vrednosti, upoređivanjem sa trenutnim datumom, i ukoliko vrednost ispunjava uslove, u trigeru se eksplicitno definiše naredba za unos sa preuzetim novim vrednostima iz tabele *,,Inserted”.* U suprotnom, opet se generiše greška.

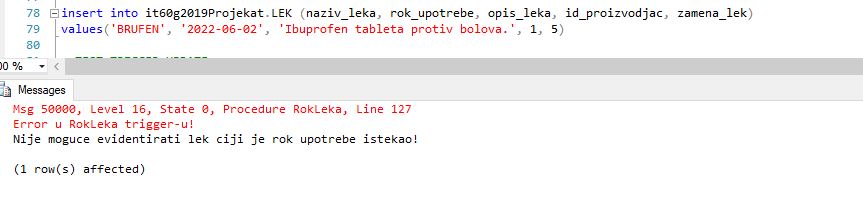
Na sledećim primerima su prikazani rezultati okidanja trigera u različitim slučajevima.

Primer uspešno insertovane n-torke u tabelu *,,Lek”* jer datum zadovoljava uslov. (*Slika br. 34*)



Slika . Primer uspešno insertovane n-toke i okidanje trigera

Primer okidanja trigera prilikom pokušaja upisa nove n-torke čija vrednost datuma isteka roka ne zadovoljava uslov. (*Slika br. 35*)



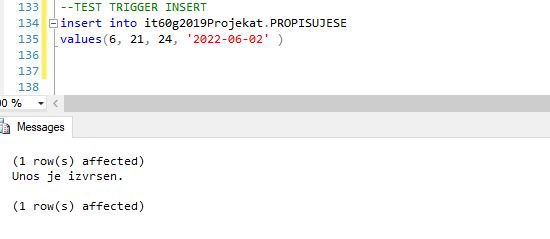
Slika . Primer neuspešnog pokušaja inserto

Primer okidanja trigera prilikom pokušaja ažuriranja postojeće n-torke u tabeli *,,Lek”.* (*Slika br. 36*)



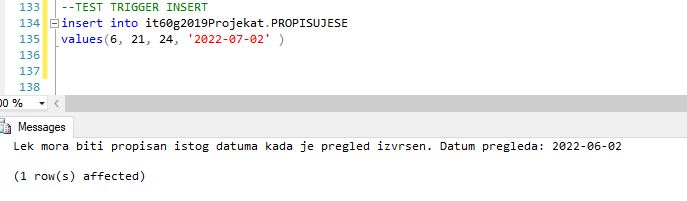
Slika . Primer neuspešnog ažuriranja i okidanje trigera

Trigeri *,,UskladiDatumProp”* i *,,UskladiDatumIzd”* su delimično povezani. Prvo je kreiran triger nad tabelom *,,PropisujeSe”* za usklađivanje datuma propisivanja leka sa datumom pregleda na kom je lek propisan. Na ovaj način je osigurano da lek ne može da se propiše pacijentu mimo pregleda, već samo na konkretnom pregledu jer se triger okida u trenutku unosa nove n-torke i upoređuje datum kada je izvršen pregled na kom je lek propisan i datum koji se unosi kao datum propisivanja leka. Ukoliko su datumi različiti, unos nije dozvoljen i koriniku se prikazuje poruka koja ga o tome obaveštava. Ako je datum isti kao i datum pregleda na kom je lek propisan, unos torke se uspešno izvršava. (*Slika br. 37*)



Slika . Okidanje trigera i uspešan unos n-torke

Primer okidanja trigera prilikom unosa datuma propisivanja leka koji nije u skladu sa datumom pregleda na kom je lek zapravo propisan. (*Slika br. 38*)



Slika . Okidanje trigera i neuspešan unos n-torke

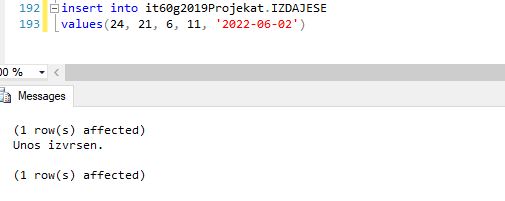
Treći triger je generisan nad tabelom *,,IzdajeSe”* kao posledica generisanog trigera nad tabelom *,,PropisujeSe”.* S obzirom da je zahtev da je datum propisivanja leka isti kao i datum pregleda, isto važi i za izdavanje recepta za propisani lek. U okviru *,,UskladiDatumIzd”* trigera koji je strukturiran kao i triger *,,UskladiDatumProp”*, postoji jedna razlika. Sada je neophodna i provera leka koji se preuzima iz tabele *,,Inserted”* jer lek koji je propisan na pregledu mora biti isti onaj lek koji će se naći na receptu.

Primer okidanja trigera sa unosom nove n-torke gde je pokušan unos pogrešnog leka. (*Slika br. 39*)



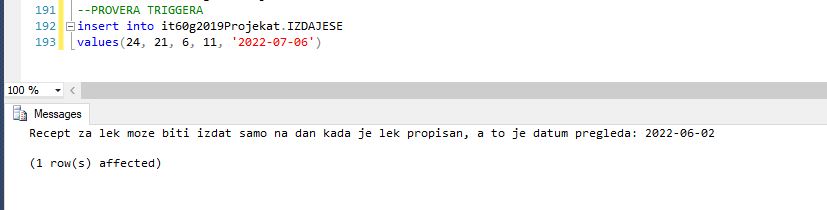
Slika . Okidanje trigera i neuspešan unos

Primer uspešnog unosa n-torke i okidanje trigera. (*Slika br. 40*)



Slika . Uspešan unos n-torke i okidanje trigera

Primer provere trigera sa unetim pogrešnim datumom izdavanja recepta. (*Slika br. 41)*



Slika . Okidanje trigera i neuspešan unos n-torke

# ZAKLJUČAK

Razvoj sistema za podršku rada dela ustanove za brigu o licima sa posebnim potrebama koji se odnosi na preglede, zahteva razumevanje i poznavanje svih aktivnosti i podataka koji su rezultat ili prethode aktivnosti pregleda, kako bi primarni cilj mogao biti ispunjen, a to je konzistentna baza podataka koja će biti pouzdana, sigurna i efikasna.

Podšema je bila prvi korak u razvoju i način da se vizuelno prikažu podaci koje baza podataka treba da čuva i njihova organizacija, dok je njeno prevođenje omogućilo uvid u šemu relacionog modela podataka koja prikazuje strukturu organizacije tih podataka na višem nivou i na osnovu kog je moguće kreirati tabele u bazi. Od samog početka razvoja baze, implementiraju se različita ograničenja i pravila koja imaju svoju ulogu i neizostavna su, kao i objekti koji se kreiraju nad bazom u cilju njene delimične automatizacije i implementacije mehanizama i operacija koje će je održavati konzistentnom.

U skladu sa ciljem projekta, u okviru dokumentacije opisan je svaki korak njegove izrade hronološki. Svaki korak preduzet tokom razvoja projekta je podjednako važan i ima značajnu ulogu za funkcionalnost baze podataka.

Osim samog projekta, vrlo je važno naglasiti koliko je bitan razvoj i projektanta čije veštine i znanje se razvijaju i nadograđuju simultano sa projektom.