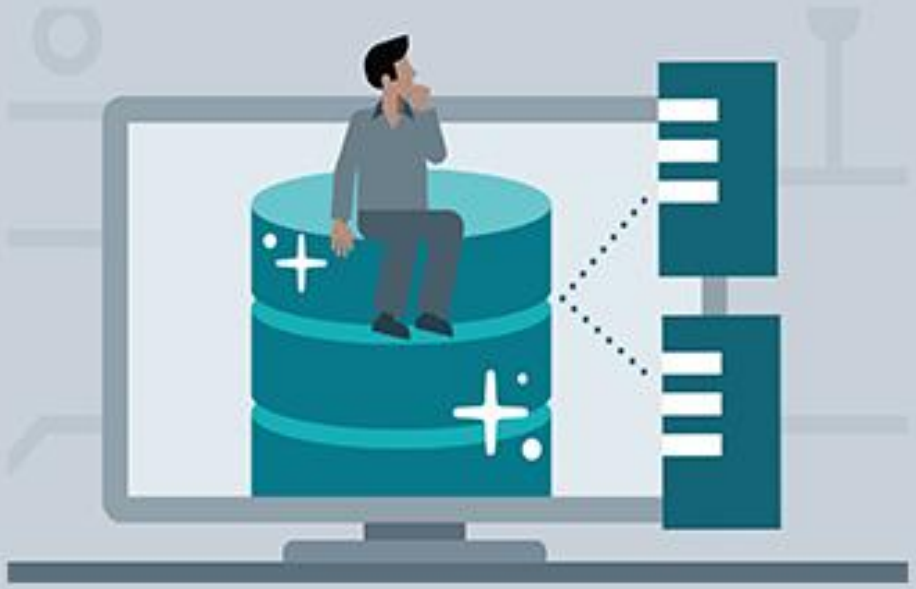
Универзитет у Приштини
са седиштем у Косовској Митровици
Природно-математички факултет

Милан Савић
Милан Дејановић

ПРАКТИКУМ ИЗ БАЗА ПОДАТАКА



Косовска Митровица, 2019.

Универзитет у Приштини
са седиштем у Косовској Митровици
Природно-математички факултет

Милан Савић
Милан Дејановић

ПРАКТИКУМ ИЗ БАЗА ПОДАТАКА

Косовска Митровица, 2019.

Милан Савић
Милан Дејановић

ПРАКТИКУМ ИЗ БАЗА ПОДАТАКА

Издавач: Природно-математички факултет у Косовској Митровици

За издавача: Проф. др Небојша В. Живић

Рецензенти: Проф. др Марко Петковић,
Природно-математички факултет, Ниш

др Негован Стаменковић, ванр. проф.,
Природно-математички факултет, Косовска Митровица

Штампа: Unigraf-X-Copy, Ниш

ISBN: 978-86-80795-42-3

Тираж: 100 примерака

Одлуком Наставно-научног већа Природно-математичког факултета Универзитета у Приштини са привременим седиштем у Косовској Митровици, број 22/2 од 26.02.2019. године, одобрено је штампање помоћног уџбеника.

ПРЕДГОВОР

Овај практикум је намењен студентима који слушају предмет „Базе података“, на Одсеку за информатику Природно-математичког факултета у Косовској Митровици. Такође, може послужити и свима онима који се у свакодневном животу сусрећу са различитим базама података да на што једноставнији начин разумеју како систем база података фукционише и да из тог система извуку максимум.

Практикум садржи материјал довољан за разумевање пројектовања и коришћења релационих база података. Потрудили смо се да изложени материјал конципирамо на начин који ће омогућити лако и брзо савладавање материје.

Захваљујемо се рецензентима др Марку Петковићу и др Неговану Стаменковићу на драгоценим сугестијама, примедбама и запажањима.

Аутори

САДРЖАЈ

1. БАЗЕ ПОДАТАКА

База података је организован и уређен скуп међусобно повезаних података. Базе података омогућавају једноставно складиштење података исте врсте, једноставно претраживање тих података те једноставно манипулисање подацима. Најједноставније речено, база података је скуп организованих информација које се односе на одређену тему а које се једноставно могу прегледати, претражити, мењати, сортирати, упоређивати и сл.

Део реалног света на који се односе подаци који се чувају у бази података назива се „мини свет“. У процесу развоја базе података најпре се формира модел реалног система, а затим се на основу тог модела гради формални систем. Формирањем модела истичу се значајне карактеристике система и привремено се стављају у други план све оне карактеристике које нису битне за систем који се развија. Постоји много могућности да се моделира систем. У фази моделирања, задатак систем аналитичара, односно инжењера захтева, је да открије функције које систем мора извршавати, податке које мора памтити и обрађивати, информације које мора обезбеђивати за потребе корисника, секвенце у којима се функције морају извршавати и у којима се може приступити подацима. Део модела система који се односи на податке назива се модел података.

1.1. Систем за управљање базама података

Софтверски систем који омогућава корисницима дефинисање, ажурирање и контролу приступа бази података назива се систем за управљање базама података (енг. *Database Management System - DBMS*). *DBMS* омогућава крајњим корисницима и програмерима да деле податке, тј. омогућава да се подаци користе од стране више апликација, а не да свака апликација има своју копију податка сачувану у посебним датотекама. *DBMS* такође пружа могућност контроле приступа подацима, осигурава интегритет података, успоставља контролу конкурентности и врши опоравак базе података. Када поменемо систем за управљање базом података (*DBMS*), углавном мислимо на релациони *DBMS* тј. *RDBMS* (*Relational Data Base Management System*).

DBMS обично нуди:

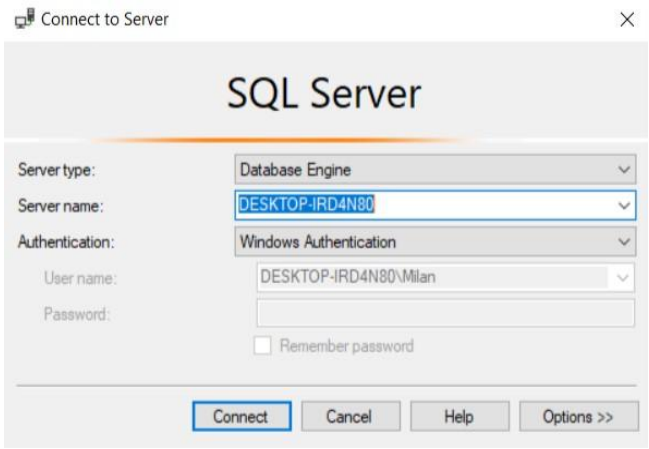
- Језик за опис података (енг. *Data Definition Language - DDL*), који омогућава корисницима дефинисање типа и структуре података, као и ограничења над подацима меморисаним у бази података.
- Језик за манипулацију подацима (енг. *Data Manipulation Language - DML*), који омогућава корисницима уметање, ажурирање, брисање и претраживање података из базе података.
- Језик за дефинисање начина меморисања података (енг. *Storage Definition Language - SDL*), који се користи за специфицирање интерне шеме базе података.
- Контролисани приступ бази података, што укључује:
    - *сигурносни систем*, који онемогућава приступ бази података неауторизованим корисницима,
    - *интегритетни систем*, који одржава конзистентност меморисаних података,
    - *систем за контролу конкуренције*, који допушта дељиви приступ подацима из базе података,
    - *систем за контролу опоравка базе података*, који омогућава реконструкцију претходног конзистентног стања у случају неке хардверске или софтверске неисправности, и
    - *каталог*, који садржи опис података који су меморисани у бази података и коме корисници могу приступати.

1.1.1. SQL Server

Microsoft SQL Server је систем за управљање релационим базама података или *RDBMS*. Као и други *RDBMS* софтвери, *Microsoft SQL Server* је изграђен на основама *SQL*-а. *SQL Server* је везан за *Transact-SQL*, једну имплементацију *SQL*-а из *Microsoft*-а који додаје скуп програмских додатака на стандардни језик. *RDBMS* као што је *SQL Server*, може имати више база података на само једном серверу а може имати и само једну. Број база података које се налазе на појединачном *SQL Server-у* зависи од фактора, као што су капацитет (снага процесора, меморија, И/О ограничења диска, итд.), аутономија (једна особа има управљачка права на серверу на коме је покренут систем, а да неко други има администраторска права на неком другом серверу), као и то колико база података одређена компанија или клијент поседује.

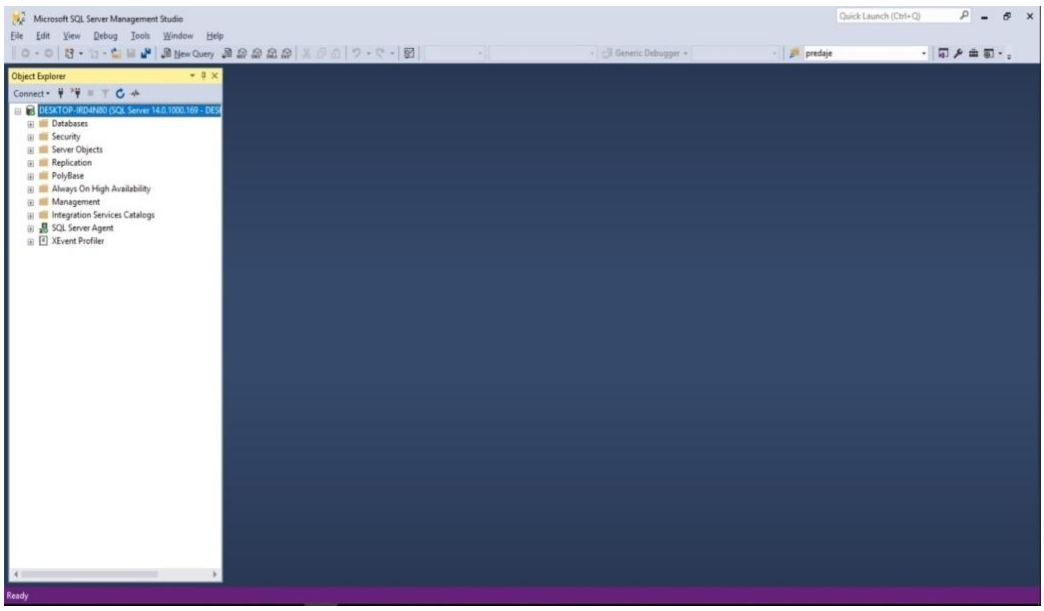
1.1.2. Развојно окружење SQL Server Management Studio

Microsoft SQL Server Management Studio је алат који омогућава управљање *SQL Server*-ом из графичког корисничког интерфејса односно без коришћења командне линије. На *слици 1.* је приказан први прозор који се приказује након покретања програма. У овом прозору се уносе подаци неопходни за конекцију са сервером. Од корисника се захтева унос одговарајућег типа сервера, имена сервера и типа аутентификације.



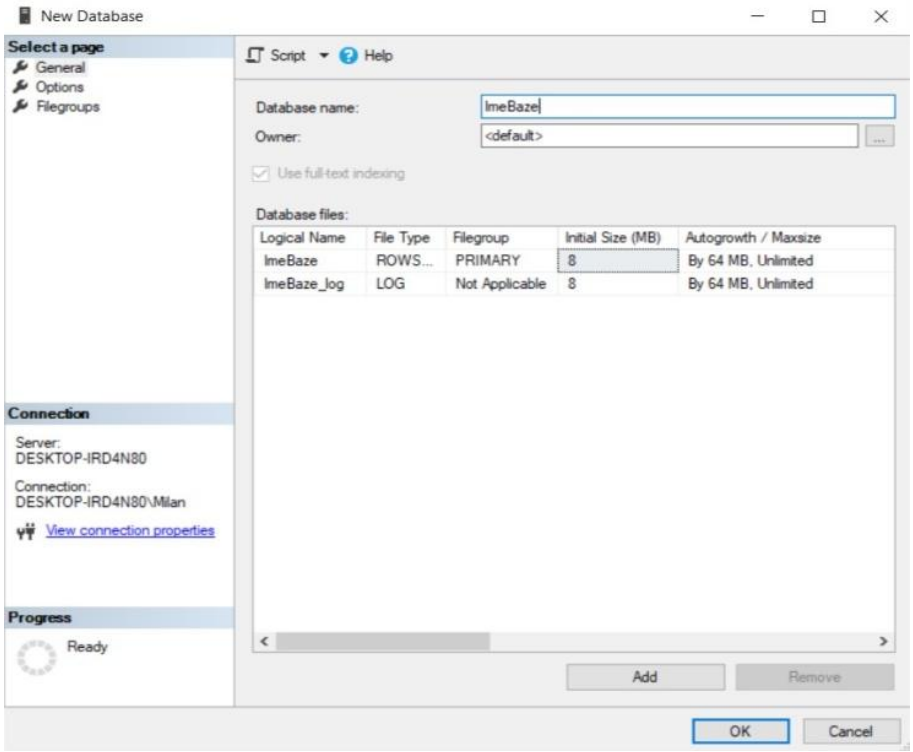
Слика 1. *Конекција са сервером*

Након успешне конекције са сервером отвара се развојно окружење приказано на *слици 2.*



Слика 2. *Развојно окружење*

На левој страни налази се *Object Explorer*. *Object Explorer* служи за приказ свих објеката базе података на серверу. Из *Object Explorer*-а се може креирати база података тако што се кликне десним кликом на *Databases* а потом одабере опција *New Database.* Након клика на *New database*, отвара се нови прозор у коме се се уноси назив базе података, власник базе података и додатни параметри (*слика 3*).

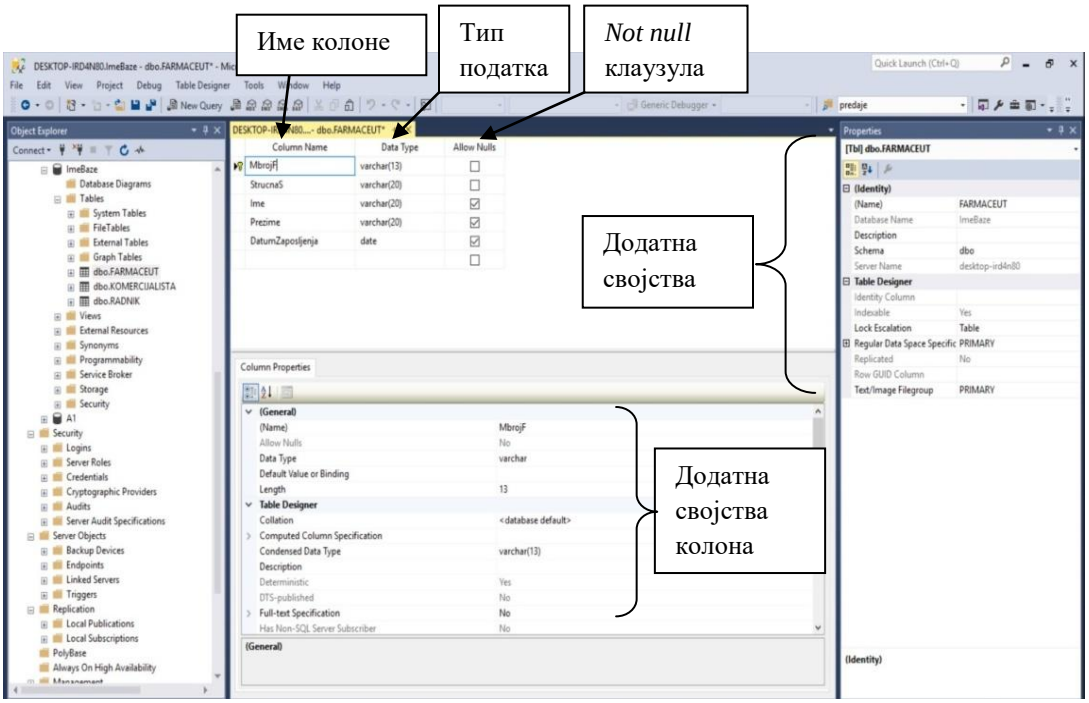


Слика 3. *Дефинисање назива базе, власника и додатних параметара базе*

Након клика на дугме *OK* у *Object Explorer*-у ће се појавити име новокреиране базе података. Двоструким кликом на име базе приказаће се додатне опције (*Database Diagrams, Tables, Views, External Resources, Synonyms, Programmability, Service Broker, Storage и Security*).

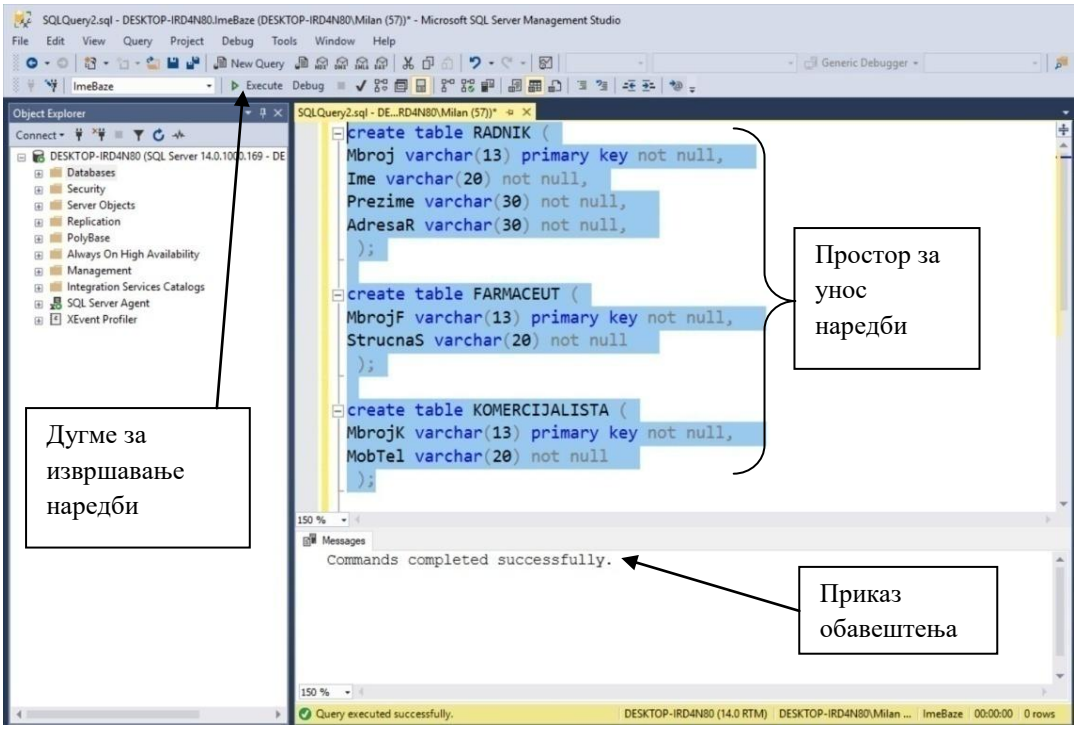
Десним кликом на опцију *Database Diagrams* могуће је креирати дијаграм базе података који може садржати неколико табела или све табеле базе података. Опција за креирање дијаграма базе података је веома корисна приликом дефинисања упита за претраживање базе података а највише приликом спајања више табела базе података јер се могу лакше уочити везе између табела и колона у бази података.

Двоструким кликом на опцију *Tables* приказаће се све табеле које постоје у одређеној бази података. Том приликом се постојеће табеле могу изменити тако што се кликне десним кликом миша на име табеле а затим на опцију *Design*. Тако је могуће изменити име колона табеле, променити тип података колона, додати колоне, брисати колоне, додати примарне и стране кључеве и још много тога. Десним кликом на име табеле а затим на опцију *Select Top 1000 Rows* могу се једноставно и брзо приказати 1000 записа који постоје у тој табели. Десним кликом на опцију *Tables* а потом на *New* ⇨ *Table*, могуће је креирати нову табелу у бази података (*слика 4*).



Слика 4. *Креирање нове табеле*

Десним кликом на име базе у *Object Explorer*-у појављује се падајући мени. Из падајућег менија одаберемо опцију *New Query*. Након тога, појавиће се нови прозор у коме се могу уносити све *SQL* наредбе (*за креирање табела у бази, измену табела, унос података, претраживање базе, ажурирање базе, итд.).* Када се наредбе унесу потребно их је означити мишем а затим кликнути на дугме *Execute*. На *слици 5*. је приказан поступак којим се наредбе уносе и извршавају као и приказ обавештења о томе да ли су наредбе успешно извршене.



Слика 5. *Унос и извршавање наредби и приказ обавештења*

Уколико су наредбе извршене успешно приказаће се порука „Commands completed successfully“. Уколико се извршава упит за претраживање базе, у овом делу за приказ обавештења биће приказана табела са записима који одговарају том упиту. Ако се приликом извршавања наредби јави грешка (нпр. грешка у синтакси или погрешно написана резервисана реч) исписаће се текстуална порука о типу грешке и број линије у којој је дошло до грешке.

2. ПРОЈЕКТОВАЊЕ БАЗЕ ПОДАТАКА

2.1. Основни концепти ЕР модела података

Један од најпопуларнијих семантичких модела података који се користи на концептуалном нивоу је модел *ентитет-веза* (енг. *Entity Relationship Model – ER model*). Основна идеја овог модела података је у томе да се реални свет, или неки његов део, може описати помоћу два примитивна концепта: ентитет и веза. Ентитет је било који објекат који се може једнозначно идентификовати, док је веза однос између два или више ентитета. Творац основног ЕР модела је *Chen*. Касније је овај модел проширен новим концептима и тако је добијен *проширени модел ентитет-веза* (енг. *Extended ER Model – EER model*).

Основни концепти ЕР модела података су:

- ентитет и тип ентитета; јаки и слаби тип ентитета
- атрибут и вредност атрибута; кључни атрибут; домен атрибута
- веза и тип везе; специјални типови веза.

2.1.1. Ентитети

Ентитет је субјекат, објекат, догађај, појам или стање о којима се прикупљају, меморишу, обрађују и презентују подаци у аутоматизованим информационим системима, а који се може једнозначно идентификовати. Ентитети из реалног света се могу класификовати у **скупове ентитета** са истим својствима. Исти ентитет може припадати различитим скуповима ентитета. **Тип ентитета** је модел тог скупа ентитета. Опис типа ентитета се назива **шема типа ентитета**. Појава типа ентитета је скуп података којима је описан један конкретни ентитет задатог типа.

Сваки тип ентитета се може идентификовати именом и листом својстава. База података садржи много различитих типова ентитета.

2.1.2. Атрибути

Ентитети се описују помоћу атрибута. Атрибути су заједничке особине које поседују сви ентитети једног скупа ентитета. Из скупа атрибута одређеног ентитета, за потребе конкретног информационог система бира се само одређени подскуп тих атрибута. Атрибут за сваки конкретни ентитет из скупа ентитета поседује одређену вредност. Скуп вредности које неки атрибут може узимати зовемо **домен атрибута.** Сваки домен атрибута се дефинише типом података, дужином података и опсегом вредности.

Атрибути могу бити прости, сложени и изведени. Атрибут је прост ако се даље не може декомпоновати или ако у конкретној ситуацији не долази до раздвојене примене појединих компонената атрибута. Атрибут је сложен ако је састављен из низа елементарних атрибута. Сложени атрибути се могу декомпоновати на просте атрибуте. Вредност простог атрибута је прост податак, а вредност сложеног атрибута је структурни податак. Атрибут је изведен ако се његова вредност може добити из других атрибута, применом неког алгоритма.

2.1.3. Веза и тип везе

Објекти у реалном систему нису међусобно изоловани већ се налазе у одређеној међусобној интеракцији. Дакле, међу ентитетима једног скупа, као и међу ентитетима више скупова, могу постојати различите релације, што се у ЕР моделу података моделира као **тип везе.** Број типова ентитета који учествују у вези представља **степен типа везе.** Ако у одређеном типу везе учествују два ентитета веза је бинарна, за три ентитета веза је тернарна, док је за *n* типова ентитета веза *n*-арна.

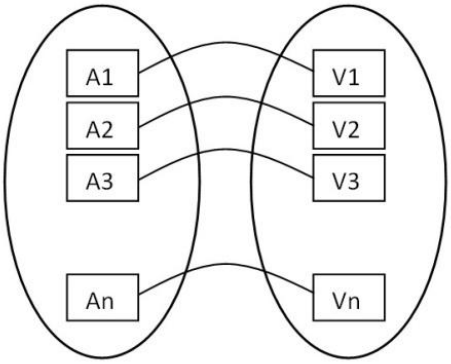
2.1.3.1. Ограничења типа везе

За типове веза дефинишу се две врсте ограничења: кардиналност и партиципација. У зависности од броја ентитета одређене врсте који су у одређеној вези са ентитетима друге врсте тј. с обзиром на кардиналност, разликујемо следеће врсте веза:

- један према један (1:1)

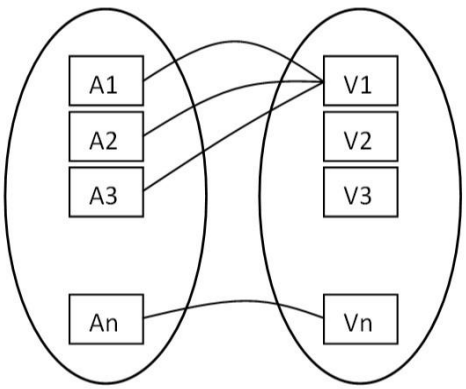
- више према један (N:1)
- један према више (1:N)
- више према више (N:M)

Веза 1:1. Сваки елемент првог скупа може бити повезан са највише једним елементом другог скупа (*слика 6*). Истовремено сваки елемент другог скупа може бити повезан са највише једним елементом првог скупа.



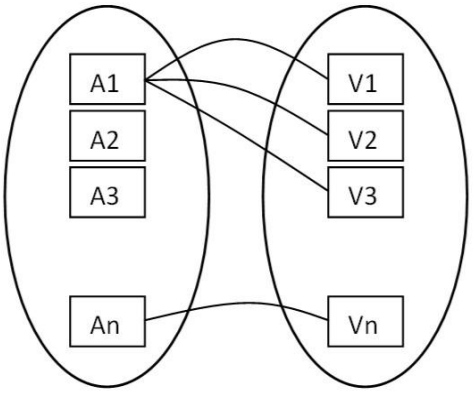
Слика 6. *Веза 1:1*

Веза N:1. Сваки елемент првог скупа може бити повезан са највише једним елементом другог скупа (*слика 7*). Истовремено сваки елемент другог скупа може бити повезан са више елемената првог скупа.



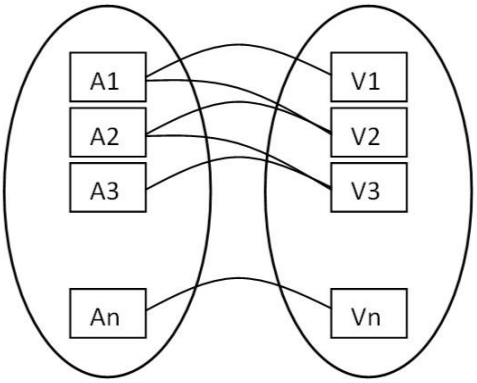
Слика 7. *Веза N:1*

Веза 1:N. Сваки елемент првог скупа може бити повезан са више елемената другог скупа (*слика 8*). Истовремено сваки елемент другог скупа може бити повезан само са једним елементом првог скупа.



Слика 8. *Веза 1:N*

Веза N:M. Сваки елемент првог скупа може бити повезан са више елемената другог скупа и обрнуто, сваки елемент другог скупа може бити повезан са више елемената првог скупа (*слика 9*).



Слика 9. *Веза N:M*

Постоје два типа ограничења партиципације ентитета у вези и то:

1. тотална партиципација и
2. парцијална партиципација.

Тотална партиципација ентитета E_1 у типу везе $R(E_1, E_2)$ значи да сваки ентитет из скупа ентитета E_1 мора бити повезан са неким ентитетом из скупа ентитета E_2 преко везе R.

Парцијална партиципација ентитета E_1 у типу везе $R(E_1, E_2)$ значи да сваки ентитет из скупа E_1 не мора бити повезан са неким ентитетом из скупа ентитета E_2 преко везе R.

2.1.4. Кључеви

Да бисмо један ентитет једнозначно идентификовали у посматраном скупу ентитета он мора поседовати неко својство, или комбинацију од неколико својстава, такву да вредност тог или тих својстава једнозначно одређују сваку појаву тог типа ентитета. Таква својства називамо карактеристичним, а њихове вредности користимо као идентификатор ентитета унутар скупа.

Размотримо сада концепт кључева. **Примарни кључ** је колона или скуп колона који једнозначно одређују остатак података у сваком реду. На пример, колона *IDKupca* у табели *Kupac* једнозначно одређује одређени ред (*табела 1*). Ово значи две ствари: ниједна два реда не смеју имати исту вредност у колони *IDKupca*, као и да чак уколико постоје два купца с истим именом и презименом, колона *IDKupca* обезбеђује да они не буду помешани јер се за манипулисање њима у целој бази користи колона *IDKupca* а не њихова имена или презимена.

Табела 1. *Табела "Kupac"*

IDKupca	Ime	Prezime	Broj telefona
1	Marko	Markovic	064123123
2	Nikola	Nikolic	064556889
3	Pera	Peric	065002003
4	Stefan	Stefanovic	063224224
5	Marko	Markovic	065555444

Ако неки тип ентитета нема кључ, тада се назива **слаби тип ентитета**. Да би се идентификовао слаби тип ентитета мора бити везан са неким типом ентитета који има способност идентификовања. То је власник идентификације. Веза између слабог типа и његовог власника назива се идентификациона веза.

Страни кључ је колона који се користи за повезивање табела у бази података. Страни кључ је колона у табели која је примарни кључ у другој табели, што значи да све вредности у колони страног кључа морају имати одговарајуће податке у другој табели у којој је та колона примарни кључ. У терминологији релационих база података, ова веза се назива референцијални интегритет. На пример, у табели *Narudzbina*, колона *IDKupca* представља страни кључ за примарни кључ табеле *Kupac (табела 2)*. Колона *IDKupca* у

табели *Narudzbina* има уписане вредности 5, 3 и 5. Ови бројеви се користе за указивање на купце који су извршили наруџбину, без потребе за коришћењем њихових стварних имена.

Табела 2. *Табела „Narudzbina“*

IDNarudzbine	BrojNarudzbine	IDKupca
1	11255	5
2	11256	3
3	11257	5

2.2. ЕР дијаграм

Шема концептуалног ЕР модела података представља се дијаграмом. Постоје различите графичке нотације које се користе за обележавање ентитета, атрибута, веза и кључева. Овде је дата нотација према [Elmasri & Navathe, 2011]:

- Типове ентитета ћемо обележавати великим словима (нпр. KUPAC, PRODAVAC).
- Атрибуте и везе се могу обележавати на два начина и то преко низа речи које се пишу великим словима са знаком за подвлачење или повлаком између речи (нпр. ID_KUPCA, ADRESA_RADNJE) и преко низа речи у којима се прво слово сваке речи пише велико (нпр. IdKupca, AdresaRadnje).
- Домен атрибута обележаваћемо са dom(A).
- Тип ентитета E са атрибутима A1, A2,...,An обележаваћемо као E(A1, A2,...,An).
- Атрибути који чине кључ типа ентитета биће подвучени.

2.2.1. Графичка нотација ЕР дијаграма

Концепти основног ЕР модела се графички представљају на следећи начин:

- Тип ентитета

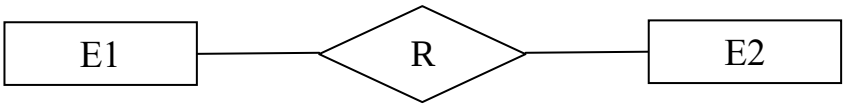
NAZIV

- Слаби тип ентитета

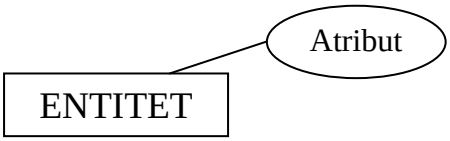
- Тип везе

- Идентификациони тип везе

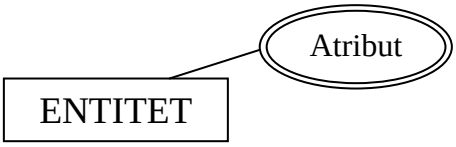
- Тип везе R између типова ентитета



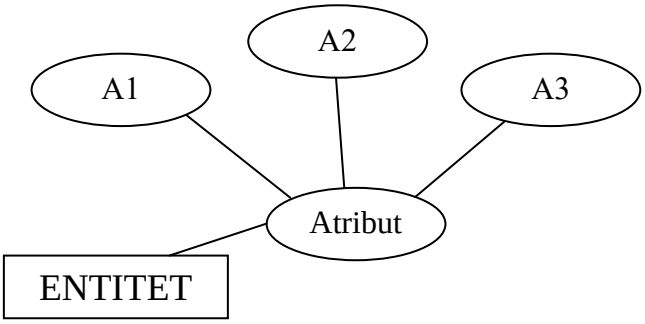
- Прости атрибути



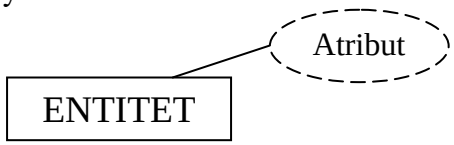
- Вишевредносни атрибути



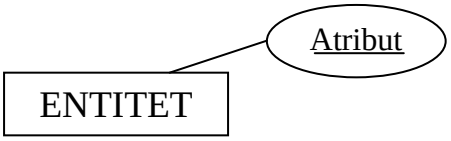
- Сложени атрибути



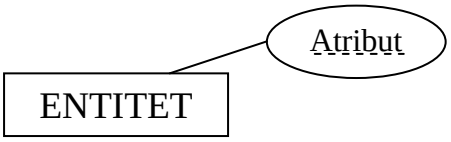
- Изведени атрибути



- Кључни атрибути

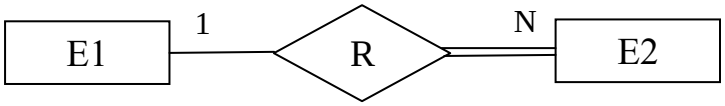


- Парцијални кључеви



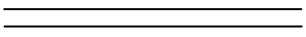
- Герунд

- Ограничење типа везе, кардиналност и партиципација, приказују се поред потега од ознаке типа ентитета до ознаке типа везе

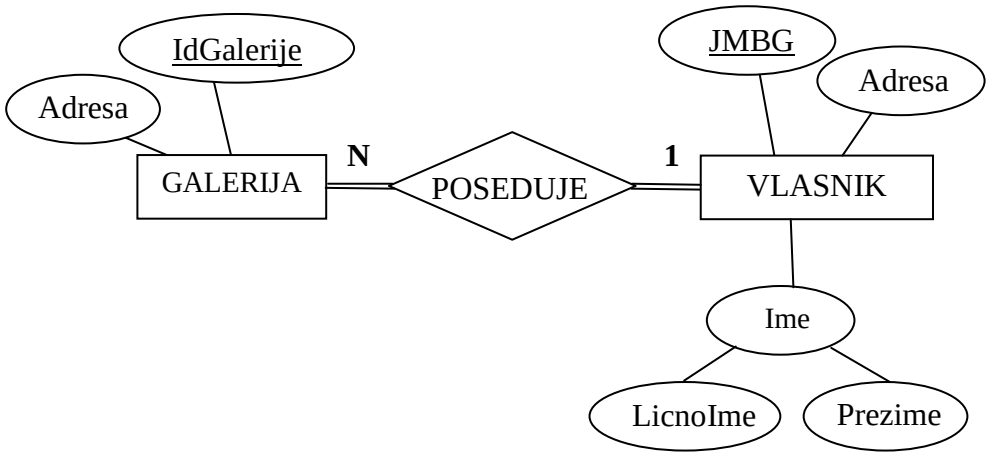


Ознака за парцијалну партиципацију:

Ознака за тоталну партиципацију:



Пример 1.



Слика 10. *Пример ЕР дијаграма*

На *слици 10*. приказана су два типа ентитета:

1. *GALERIJA* – са атрибутима *IdGalerije* (примарни кључ) и *Adresa* и
2. *VLASNIK* – са атрибутима *JMBG* (примарни кључ), *Adresa* и *Ime* (сложени атрибут који се састоји из две компоненте: *LicnoIme* и *Prezime*).

Између ова два типа ентитета постоји веза *POSEDUJE*, која дефинише однос власника и галерије. Веза је типа 1:N, зато што власник може да у власништву поседује више галерија, а једна галерија има само једног власника. Партиципација на страни власника и галерије је тотална, уз претпоставку да се у бази података чувају подаци само о власницима који поседују неку галерију, као и подаци о галеријама које имају власника.

2.3. Релациони модел

Релациони модел се заснива на групи математичких принципа изведених из теорије скупова и предикатне логике. Те принципе је касних шездесетих година први применио на област моделовања података др E. F. Codd. Правила релационог модела дефинишу облик у којем се подаци представљају (структура података), начин на који се подаци штите (интегритет података) и операције које се могу извршавати над подацима

(манипулисање подацима). Релациони модел ослобађа корисника фрустрација око руковања подацима на ниском нивоу, залажења у детаље смештања података и методама приступа из корисничког интерфејса.

У релационом моделу података, релације се користе за чување информација о објектима које треба представити у бази података. Релација се представља табелом, где врсте одговарају појединим слоговима а колоне атрибутима. Елементи релације су врсте у табели.

2.4. Пресликавање ЕР модела у релациони модел

Пресликавање ЕР модела у релациони модел одвија се у седам сукцесивних корака.

Корак 1. Пресликавање регуларног типа ентитета

За сваки регуларни тип ентитета Е у ER шеми креира се релација R која садржи све просте атрибуте и све просте компоненте свих сложених атрибута из Е. Један од кључних атрибута из Е се узима за примарни кључ релације R. Ако је кључни атрибут у Е сложен, тада се његов скуп простих атрибута узима заједно као примарни кључ у R.

Корак 2. Пресликавање слабог типа ентитета

За сваки слаби тип ентитета S у ER шеми чији је власник тип ентитета Е, креира се релација R која садржи све просте атрибуте из S и све просте компоненте свих сложених атрибута из S. Релација R као спољашњи кључ садржи све атрибуте примарног кључа релације типа ентитета Е. За примарни кључ релације R узима се комбинација примарног кључа власника Е и парцијалног кључа слабог типа ентитета S.

Корак 3. Пресликавање веза 1:1

За сваки бинарни тип веза R(1:1) у ER шеми идентификују се релације S и T које одговарају типовима ентитета који партиципирају у R. Бира се једна од ове две релације, рецимо S, и у њу се као спољашњи кључ укључује примарни кључ релације T. За релацију S треба бирати тип ентитета који тотално партиципира у R. Као атрибуте релације S треба укључити све просте атрибуте и све просте компоненте сложених атрибута типа везе R. Може се изабрати и алтернативно решење по коме се формира заједничка релација за

типове ентитета S и T у коју се укључују сви атрибути из обе релације. Ово решење је добро када оба типа ентитета тотално партиципирају у R и када не партиципирају ни у једном другом типу везе.

Корак 4. Пресликавање веза 1:N

За сваки бинарни тип везе R(1:N) у ER шеми идентификује се релација S која партиципира на N страни. У S се као спољашњи кључ укључује примарни кључ релације T која представља тип ентитета који партиципира на 1 страни. Као атрибуте релације S треба укључити све просте атрибуте и све просте компоненте сложених атрибута типа везе R.

Корак 5. Пресликавање веза M:N

За сваки бинарни тип везе R(M:N) у ER шеми креира се нова релација S. У S се као спољашњи кључеви укључују примарни кључеви релација које представљају типове ентитета који партиципирају у R. Комбинација ових спољашњих кључева формира примарни кључ релације S. У релацији S се такође укључују сви прости атрибути и све просте компоненте сложених атрибута типа везе R.

Корак 6. Пресликавање вишевредносног атрибута

За вишевредносни атрибут A типа ентитета E или типа повезника P, креира се нова релација R која садржи атрибут A и примарни кључ K релације којом је представљен тип ентитета E или тип везе P. За примарни кључ релације R бира се комбинација A и K. Ако је вишевредносни атрибут сложен укључују се све његове просте компоненте.

Корак 7. Пресликавање n-арног типа везе

За сваки n-арни тип везе R, n>2, креира се нова релација S. У S се као спољашњи кључеви укључују примарни кључеви релација које представљају типове ентитета који партиципирају у R. Комбинација ових спољашњих кључева формира примарни кључ релације S. У релацији S се такође укључују сви прости атрибути и све просте компоненте свих сложених атрибута n-арног типа везе. Ако неки тип ентитета E партиципира у R са ограничењем (min, max) и ако је max=1, тада се као примарни кључ релације S може узети онај спољашњи кључни атрибут којим се релација S референцира на релацију којом је представљен тип ентитета E.

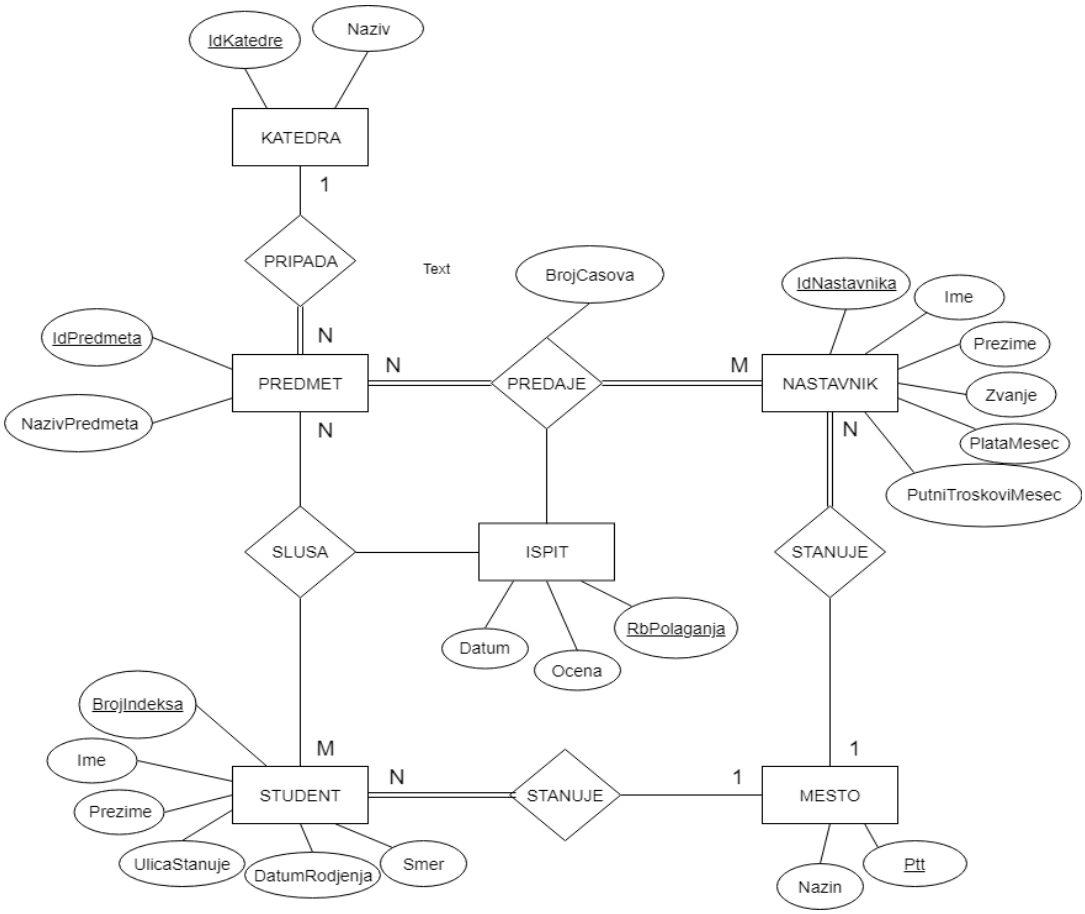
2.4.1. Превођење герунда

Када треба повезати два типа везе користи се герунд. Такође, када тип везе има своје атрибуте, може постати герунд. Потреба таквог селективног повезивања ентитета три или више скупова, код којег у вези могу учествовати само ентитети који су већ у некој другој вези са ентитетима једног (или више) других скупова, указује на неопходност коришћења герунда, као модела тих веза.

Приликом превођења герунда, користе се следећа правила:

- Преводи се у посебну шему релације. Примењују се правила превођења регуларних повезника типа N:M.
- При повезивању герунда са другим елементима, герунд се посматра као тип ентитета и тако се третира при превођењу те везе.

Пример 2. ER дијаграм базе „Fakultet“



***Слика 11**. ER дијаграм базе података “Fakultet”*

Релациони модел базе „Fakultet“

KATEDRA	
IdKatedre	Naziv

PREDMET		
IdPredmeta	NazivPredmeta	*IdKatedre*

SLUSA	
BrojIndeksa	IdPredmeta

STUDENT						
BrojIndeksa	Ime	Prezime	UlicaStanuje	DatumRodjenja	Smer	*PttStanuje*

ISPIT					
RbPolaganja	*BrojIndeksa*	*IdPredmeta*	*IdNastavnika*	Ocena	Datum

PREDAJE		
IdNastavnika	IdPredmeta	BrojCasova

NASTAVNIK						
IdNastavnika	Ime	Prezime	Zvanje	PlataMesec	PutniTrosakMesec	*PttStanuje*

MESTO	
Ptt	Naziv

3. КРЕИРАЊЕ И ПРЕТРАЖИВАЊЕ БАЗЕ ПОДАТАКА

За приступ подацима који се налазе у бази података потребан је језик упита. Ово се односи на било коју врсту базе података. Језик упита не представља нужно програмски језик у класичном смислу већ може бити уграђен у програмски језик и коришћен од стране програмера база података, или се пак може користити за једноставну сврху приступа подацима. Добар програмер који ради унутар базе података треба да има солидно познавање неког језика упита.

SQL (енгл. *Structured Query Language*) је стандардни релациони упитни језик који служи за креирање, организацију и манипулацију подацима у релационим базама података. Сам настанак језика се везује за *IBM*-ову истраживачку лабораторију у Сан Хозеу у Калифорнији, где је *SQL* развијен касних 70-их година, у склопу пројекта „*System R*“.

SQL је језик за сваку категорију људи који из неког разлога имају потребу за приступом подацима. Као језик упита, *SQL* представља сет инструкција са јасном сврхом: дозволити приступ подацима и метаподацима.

3.1. Основне SQL наредбе

SQL наредбе се могу писати било великим било малим словима (*case insensitive*). На пример, наредба за селектовање се може писати и као *SELECT* и као *select*. Празни редови у коду и додатни размак између наредби се игноришу приликом извршавања кода.

Наредба 1. *CREATE DATABASE*

Ова наредба користи се за креирање нове SQL базе података.

Синтакса је следећа:

```
CREATE DATABASE database_name;
```

Пример:

```
CREATE DATABASE FirstDB;
```

Наредба 2. *DROP DATABASE*

Ова наредба користи се за брисање одређене SQL базе података.

Синтакса је следећа:

```
DROP DATABASE database_name;
```

Пример:

```
DROP DATABASE FirstDB;
```

Наредба 3. *BACKUP DATABASE*

Ова наредба користи се за креирање копије одређене SQL базе података.

Синтакса је следећа:

```
BACKUP DATABASE database_name
TO DISK = 'filepath';
```

Пример:

```
BACKUP DATABASE database_name
TO DISK = 'C:\database\backups\FirstDB.bak';
```

Наредба 4. *CREATE TABLE*

Ова наредба користи се за креирање табеле у бази података.

Синтакса је следећа:

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    column3 datatype,
    ....
);
```

Први параметри (*column1, column2, column3, ...)* означавају имена колона у табели која се креира. Параметар *datatype* означава тип података који колона може да чува. Неки од најчешћих типова података који се користе у базама

података су *char*, *varchar*, *integer*, *double*, *date*, *time* итд. Опис ових типова података дат је у *табели 3.*

Табела 3. *Основни типови података*

Тип података	Опис
char	Користи се за чување стрингова фиксне дужине (слова, бројеве и специјалне карактере). Може да прими до 255 карактера.
varchar	Користи се за чување стрингова променљиве дужине (слова, бројеве и специјалне карактере). Може да прими до 255 карактера.
integer	Користи се за чување целобројних вредности.
double	Користи се за чување реалних бројева.
date	Користи се за чување датума у формату YYYY-MM-DD.
time	Користи се за чување времена у формату HH:MI:SS.

Пример 1:

```
CREATE TABLE Automobil (
    RegistarskiBroj varchar(30),
    Marka varchar(20),
    DatumProizvodnje date
);
```

Извршавањем ових наредби, добили смо *табелу 4.*

Табела 4. *Креирана табела „Automobil“*

RegistarskiBroj	Marka	DatumProizvodnje
025-KM-123	Audi	2019-02-16
115-UR-101	Fiat	2019-01-02
225-BG-689	Fiat	2018-12-25

У овом примеру приказан је начин креирања табеле *Automobil* са колонама *RegistarskiBroj*, *Marka* и *DatumProizvodnje*. Колона *RegistarskiBroj* и

колона *Marka* чувају податке типа *varchar*, што значи да могу да садрже бројеве, слова и специјалне карактере. Максимални број карактера који колона може да прими дефинисан је у загради, што у нашем примеру износи максималних 30 карактера у колони *RegistarskiBroj* односно 20 у колони *Marka*. Колона *DatumProizvodnje* чува податке типа *date*, односно податке о одређеном датуму. Подразумевани (*default*) формат датума који се чува у бази података је YYYY-MM-DD (нпр. 2019-01-12). Формат датума је могуће прилагодити.

Приликом креирања табеле могуће је поставити одређена ограничења (енг. *constraints*). То значи да можемо да дефинишемо примарни кључ табеле, стране кључеве, јединствене кључеве, подразумеване вредности колона итд. У наредном примеру, креирамо табелу у бази података која има примарни кључ и *NOT NULL* ограничења. Ограничења ће опширније бити обрађена у наредном поглављу.

Пример 1.

```
CREATE TABLE Automobil (
    RegistarskiBroj varchar(30) not null primary key,
    Marka varchar(20) not null,
    DatumProizvodnje date not null
);
```

Овом наредбом смо креирали табелу у бази података која има три колоне. Примарни кључ табеле је колона *RegistarskiBroj*. Записи у овој колони морају бити јединствени. Све ћелије у свим колонама морају имати одређену вредност (*NOT NULL* ограничење).

Пример 2.

```
CREATE TABLE Vlasnik (
   Id int not null primary key,
   JMBG varchar(13) not null,
   Ime varchar(30) not null,
   Prezime varchar(30) not null,
   RegistarskiBroj varchar(30) foreign key references Automobil
   (RegistarskiBroj),
   Adresa varchar(30) not null
);
```

Добијена је *табела 5*.

Табела 5. *Креирана табела „Vlasnik“*

Id	JMBG	Ime	Prezime	RegistarskiBroj	Adresa
1	1255566688744	Marko	Markovic	025-KM-123	Kosovska Mitrovica
2	2587412369854	Pera	Peric	115-UR-101	Beograd
3	3248753698745	Uros	Urosevic	225-BG-689	Beograd

У овом примеру креирана је табела *Vlasnik*. Осим примарног кључа, ова табела садржи и страни кључ. Он се користи за повезивање табела. Страни кључ ове табеле је колона *RegistarskiBroj*. Вредности ове колоне морају да се подударају са вредностима колоне *RegistarskiBroj* табеле *Automobil*.

Наредба 5. *DROP TABLE*

Ова наредба користи се за брисање табеле из базе података.

Синтакса је следећа:

```
DROP TABLE table_name;
```

Пример:

```
DROP TABLE Automobil;
```

Наредба 6. *ALTER TABLE*

Ова наредба користи се за брисање, додавање и измену колона у постојећој табели. Користи се и за додавање и брисање различитих ограничења (енг. *constraints*) у постојећој табели.

Синтакса за додавање колоне:

```
ALTER TABLE  table_name
ADD  column_name datatype;
```

Пример:

```
ALTER TABLE  Automobil
ADD Boja varchar(15);
```

Сада у табели имамо још једну колону са називом *Boja (табела 6)*.

Табела 6. *Додавање нове колоне „Боја“*

RegistarskiBroj	Marka	DatumProizvodnje	Boja
025-KM-123	Audi	2019-02-16	crna
115-UR-101	Fiat	2019-01-02	crna
225-BG-689	Fiat	2018-12-25	plava

Синтакса за измену типа податка колоне (*datatype* овде означава нови тип података):

```
ALTER TABLE table_name
ALTER COLUMN column_name datatype;
```

Пример:

```
ALTER TABLE Automobil
ALTER COLUMN Boja nvarchar(15);
```

Синтакса за брисање колоне:

```
ALTER TABLE table_name
DROP COLUMN column_name;
```

Пример:

```
ALTER TABLE Automobil
DROP COLUMN Boja;
```

Из табеле *Automobil* бришемо колону *Boja (табела 7)*.

Табела 7. *Брисање колоне „Боја“*

RegistarskiBroj	Marka	DatumProizvodnje
025-KM-123	Audi	2019-02-16
115-UR-101	Fiat	2019-01-02
225-BG-689	Fiat	2018-12-25

Синтакса за додавање примарног кључа у постојећу табелу:

```
ALTER TABLE table_name
ADD CONSTRAINT PK_ table_name PRIMARY KEY(column_name);
```

Пример:

```
ALTER TABLE Automobil
ADD CONSTRAINT PK_Automobil PRIMARY KEY(RegistarskiBroj);
```

Синтакса за брисање примарног кључа из постојеће табеле:

```
ALTER TABLE table_name
DROP CONSTRAINT PK_table_name;
```

Пример:

```
ALTER TABLE Automobil
DROP CONSTRAINT PK_Automobil;
```

Синтакса за додавање страног кључа у постојећу табелу:

```
ALTER TABLE table1
ADD CONSTRAINT FK_table2table1
FOREIGN KEY (column_name)
REFERENCES table2(column_name);
```

Пример:

```
ALTER TABLE Vlasnik
ADD CONSTRAINT FK_AutomobilVlasnik
FOREIGN KEY (RegistarskiBroj)
REFERENCES Automobil(RegistarskiBroj);
```

Синтакса за брисање страног кључа из постојеће табеле:

```
ALTER TABLE table1
DROP CONSTRAINT FK_table2table1;
```

Пример:

```
ALTER TABLE Vlasnik
DROP CONSTRAINT FK_AutomobilVlasnik;
```

Наредба 7. *SELECT*

У релационим базама података, подаци се смештају у табелама. Табела *Vlasnik* садржи колоне *Id*, *JMBG*, *Ime*, *Prezime*, *RegistarskiBroj* и *Adresa (табела 5)*. Претпоставимо да је потребно пронаћи све податке о власницима аутомобила. То је могуће одрадити помоћу наредбе *SELECT*.

Синтакса:

```
SELECT column1, column2, ...
FROM table_name;
```

Пример:

```
select Id, Ime, Prezime, RegistarskiBroj, Adresa
from Vlasnik;
```

У овом примеру смо селектовали све колоне из табеле тако што смо користили имена колона. То је могуће одрадити и на једноставнији начин, коришћењем * (*звезда*). Приказ добијених резултата дат је у *табели 8*.

Табела 8. *Резултат SELECT наредбе (сви записи)*

Id	JMBG	Ime	Prezime	RegistarskiBroj	Adresa
1	1255566688744	Marko	Markovic	025-KM-123	Kosovska Mitrovica
2	2587412369854	Pera	Peric	115-UR-101	Beograd
3	3248753698745	Uros	Urosevic	225-BG-689	Beograd
4	2258874422589	Stefan	Nikolic	225-BG-225	Beograd

Коришћењем SELECT наредбе могуће је креирати и веома сложене упите тј. могуће је спајати табеле и издвајати вредности у зависности од задатих услова. Овде је приказан још један пример у коме из табеле *Vlasnik* издвајамо све власнике аутомобила чија је адреса Београд (*табела 9*). То постижемо коришћењем *WHERE* оператора и то на следећи начин:

```
select * from Vlasnik where Adresa = 'Beograd';
```

Табела 9. *Приказ власника аутомобила из Београда*

Id	JMBG	Ime	Prezime	RegistarskiBroj	Adresa
2	2587412369854	Pera	Peric	115-UR-101	Beograd
3	3248753698745	Uros	Urosevic	225-BG-689	Beograd
4	2258874422589	Stefan	Nikolic	225-BG-225	Beograd

Наредба 8. *INSERT INTO*

Ова наредба служи за уношење података у табелу базе података.

Синтакса је следећа:

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

Пример:

```
INSERT INTO Vlasnik VALUES (3, '3248753698745', 'Uros', 'Urosevic',
'225-BG-689', 'Beograd');
```

Наредба 9. *DELETE*

Ова наредба служи за брисање података из одређене табеле у бази података.

Синтакса је следећа:

```
DELETE FROM table_name WHERE condition;
```

Пример:

```
DELETE FROM Vlasnik WHERE Ime = 'Stefan' AND Prezime = 'Nikolic';
```

Наредба 10. *UPDATE*

Ова наредба служи за ажурирање података у одређеној табели базе података.

Синтакса је следећа:

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

Пример:

```
UPDATE Vlasnik SET Adresa = 'Nis' WHERE JMBG = '3248753698745';
```

3.2. Оператори

У *SQL*-у постоји неколико врста оператора који су приказани у табелама.

Табела 10. *Аритметички оператори*

Оператор	Опис
+	сабирање
-	одузимање
*	множење
/	дељење
%	остатак целобројног дељења

Пример: Селектовање власника аутомобила чији је *Id* једнак 2+1.

```
SELECT Id, Ime, Prezime, Adresa FROM Vlasnik
WHERE Id = 1+2
```

Табела 11. *Оператори поређења*

Оператор	Опис
=	једнако
>	веће
<	мање
>=	веће или једнако
<=	мање или једнако
!=	различито

Пример: Селектовање власника аутомобила чији је *Id* већи или једнак 2.

```
SELECT Id, Ime, Prezime, Adresa FROM Vlasnik
WHERE Id >= 2
```

Табела 12. *Логички оператори*

Оператор	Опис
AND	логичко „и“
OR	логичко „или“
NOT	негација
ALL	тачно – ако све вредности испуњавају услов
ANY	тачно – ако нека вредност испуњава услов
BETWEEN	тачно – ако је операнд унутар задатих граница
IN	тачно – ако је операнд једнак једној вредности из листе
EXIST	тачно – ако подупит врати неку вредност
LIKE	тачно – ако операнд задовољава одређени образац

Пример: Селектовање власника аутомобила чије је име Марко или Урош.

```
SELECT Id, Ime, Prezime, Adresa FROM Vlasnik
WHERE Ime = 'Marko' OR Ime = 'Uros';
```

Пример: Селектовање власника аутомобила чији је *Id* у опсегу од 3 до 10 (укључујући 3 и 10).

```
SELECT * FROM Vlasnik
```

```
WHERE Id BETWEEN 3 AND 10;
```

Пример: Селектовање власника аутомобила чије име почиње на слово М.

```
SELECT * FROM Vlasnik
WHERE Ime LIKE 'M%';
```

Пример: Селектовање власника аутомобила чија је адреса Београд или Косовска Митровица.

```
SELECT * FROM Vlasnik
WHERE Adresa IN ('Beograd', 'Kosovska Mitrovica');
```

3.3. Агрегатне функције

У овом делу ћемо размотрити пет важних агрегатних функција: *SUM, AVG, MIN, MAX* и *COUNT*. Оне се називају агрегатним функцијама јер дају сумиране резултате неког упита а не списак свих редова.

- SUM() - даје збир вредности дате колоне у свим редовима који задовољавају неки услов, при чему је дата колона нумеричка.
- AVG() – даје просечну вредност дате колоне.
- MIN() – даје најмању вредност у датој колони.
- MAX()– даје највећу вредност у датој колони.
- COUNT() – даје број редова који задовољавају одређени услов.

Сада ћемо на табели *Automobil (табела 13),* приказати примере коришћења агрегатних функција.

Табела 13. *Табела „Automobil“*

RegistarskiBroj	Marka	DatumProizvodnje	Vrednost
025-KM-123	Audi	2019-02-16	1220000
115-UR-101	Fiat	2019-01-02	380000
225-BG-225	BMW	2018-12-30	2360000
225-BG-689	Fiat	2018-12-25	440000

Синтакса – *SUM():*

```
SELECT SUM(column_name)
FROM table_name
WHERE condition;
```

Пример : Збир вредности свих аутомобила марке Fiat.

```
SELECT SUM(Vrednost)
FROM Automobil
WHERE Marka = 'Fiat';
```

Добија се резултат приказан у *табели 14.*

Табела 14. *Резултат упита*

(no column name)
820000

Да бисмо добијеној колони дали назив, користимо AS наредбу која резултате одређеног упита смешта у привремену колону (*табела 15*).

```
SELECT SUM(Vrednost) AS VrednostFiat
FROM Automobil
WHERE Marka = 'Fiat';
```

Табела 15. *Резултат упита с називом привремене колоне*

VrednostFiat
820000

Синтакса – *AVG():*

```
SELECT AVG(column_name)
FROM table_name
WHERE condition;
```

Пример: Просечна вредност свих аутомобила.

```
SELECT AVG(Vrednost) AS ProsecnaVrednost
FROM Automobil;
```

Добија се резултат приказан у *табели 16.*

Табела 16. *Резултат упита*

ProsecnaVrednost
1100000

Синтакса – *MIN()*:

```
SELECT MIN(column_name)
FROM table_name
WHERE condition;
```

Пример: Најмања вредност уписаних аутомобила.

```
SELECT MIN(Vrednost) AS MinimalnaVrednost
FROM Automobil;
```

Добија се резултат приказан у *табели 17.*

Табела 17. *Резултат упита*

MinimalnaVrednost
380000

Синтакса – *MAX()*:

```
SELECT MAX(column_name)
FROM table_name
WHERE condition;
```

Пример: Највећа вредност уписаних аутомобила.

```
SELECT MAX(Vrednost) AS MaksimalnaVrednost
FROM Automobil;
```

Добија се резултат приказан у *табели 18.*

Табела 18. *Резултат упита*

MaksimalnaVrednost
2360000

Синтакса – *COUNT():*

```
SELECT COUNT(column_name)
FROM table_name
WHERE condition;
```

Пример: Број аутомобила марке „Fiat“.

```
SELECT COUNT(Marka) AS BrojFiatAutomobila
FROM Automobil
WHERE Marka = 'Fiat';
```

Добија се резултат приказан у *табели 19.*

Табела 19. *Резултат упита*

BrojFiatAutomobila
2

3.4. Додатне математичке функције

Осим агрегатних функција, постоји још математичких функција које се могу користити у *SELECT* блоку. На тај начин се као резултат претраживања могу приказати резултати израчунавања неког математичког израза. Најважније математичке функције које се могу користити су:

- ROUND(x,d) – заокруживање броја n на d децимала, ако се d не наведе, заокружује се на најближи цео број.
- POWER(x,n) – број x на n-ти степен.
- ABS(n) – апсолутна вредност броја n.
- SIGN(n) – предзнак броја n.
- SQRT(n) – корен броја n.

Примери:

```
SELECT POWER(8, 2) AS Stepen; =>  добија се 64

SELECT ROUND(123.9994, 3) AS ZaokruzeniBroj; => добија се 123.999

SELECT ABS(-243.5) AS ApsolutnaVrednost; => добија се 243.5

SELECT SIGN(-243.5) AS Predznak; => добија се -1.0

SELECT SQRT(64) AS Koren; => добија се 8
```

3.5. Клаузуле

Клаузуле су саставни делови SQL упита. Најчешће се користе ради примене различитих ограничења и дефинисања услова селекције.

SQL клаузуле су:

- CONSTRAINT

- WHERE
- GROUP BY
- HAVING
- ORDER BY

3.5.1. CONSTRAINT клаузуле

Клаузуле за дефинисање и одређивање примарних и страних кључева спадају у *constraint* клаузуле. *Constraint* клаузуле у ствари представљају део *create table* наредбе или *alter table* наредбе. *Constraint* у ствари представља услов који податак мора да потврди, односно испуни како би постао део табеле или колоне.

Constraint клаузуле које се најчешће користе у *SQL*-у:

- NOT NULL – све ћелије у колони морају имати одређену вредност
- UNIQUE – колона не може имати две исте вредности
- PRIMARY KEY - комбинација NOT NULL и UNIQUE.
- FOREIGN KEY – јединствен идентификатор из друге табеле
- CHECK – осигурава да унета вредност испуњава одређени услов
- DEFAULT – постављање подразумеване вредности
- INDEX – користе се да би се убрзала претрага и приступ подацима

У следећем примеру приказан је начин коришћења *constraint* клаузула. Примарни кључ је колона *Id*, јединствени кључ је колона *JMBG*, колона *RegistarskiBroj* је страни кључ и последња колона *Starost* има постављено ограничење *CHECK* којим се проверава да ли је број година власника већи или једнак 18 (приликом уписа података о власнику, подаци ће бити уписани у табелу ако је испуњен услов, у супротном неће бити уписани).

```
CREATE TABLE Vlasnik (
   Id int not null primary key,
   JMBG varchar(13) not null unique,
   Ime varchar(30) not null,
   Prezime varchar(30) not null,
   RegistarskiBroj varchar(30) foreign key references
   Automobil (RegistarskiBroj),
   Adresa varchar(30) not null,
   Starost int not null check(Starost>=18)
);
```

3.5.2. WHERE клаузула

WHERE клаузула се користи за проналажење записа у табели који задовољавају одређени услов.

Синтакса је следећа:

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Пример 1.

```
SELECT * FROM Automobil
WHERE Marka = 'Fiat';
```

Овај упит извлачи податке из табеле *Automobil* и приказује регистрациони број, марку и датум производње свих аутомобила марке *Fiat*. Резултат упита приказан је у *табели 20*.

Табела 20. *Аутомобили марке „Fiat“*

RegistarskiBroj	Marka	DatumProizvodnje
115-UR-101	Fiat	2019-01-02
225-BG-689	Fiat	2018-12-25

Пример 2.

```
SELECT * FROM Vlasnik
WHERE Adresa = 'Beograd' AND Ime = 'Uros';
```

Из табеле *Vlasnik* селектујемо све власнике аутомобила чија је адреса Београд и чије је име Урош. Резултат упита приказан је у *табели 21*.

Табела 21. *Подаци о власницима аутомобила чије је име Урош и који су из Београда*

Id	JMBG	Ime	Prezime	RegistarskiBroj	Adresa
3	3248753698745	Uros	Urosevic	225-BG-689	Beograd

3.5.3. GROUP BY клаузула

Ова клаузула се користи након *WHERE* клаузуле и има за циљ груписање добијених записа. Најчешће се употребљава заједно са агрегатним

функцијама (*count, min, max, sum, avg*). Један специјалан начин употребе ове клаузуле је повезивање неке агрегатне функције са групом редова (ово се нарочито односи на функцију *COUNT* која броји редове у свакој групи). Важно је напоменути да колоне које се налазе у *SELECT* делу морају наћи и у *GROUP BY* делу (осим колона које су обухваћене неком агрегатном функцијом).

Синтакса је следећа:

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```

Пример: Приказ броја власника који су груписани по адреси на којој живе.

```
SELECT COUNT(Ime) AS BrojLjudi, Adresa
FROM Vlasnik
GROUP BY Adresa;
```

Резултат је приказан у *табели 22.*

Табела 22. *Резултат упита*

BrojLjudi	Adresa
3	Beograd
1	Kosovska Mitrovica

3.5.4. HAVING клаузула

Коришћењем *GROUP BY* клаузуле формирамо групе а критеријуме за селекцију група одређујемо помоћу *HAVING* клаузуле.

Синтакса је следећа:

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition
ORDER BY column_name(s);
```

Пример: Број власника који су груписани по адреси на којој живе с тим да је потребно да тачан број људи буде једнак или већи од три.

```
SELECT COUNT(Ime) AS BrojLjudi,  Adresa
FROM Vlasnik
GROUP BY Adresa
HAVING COUNT(Ime) >= 3;
```

Резултат је приказан у *табели 23.*

Табела 23. *Резултат упита*

BrojLjudi	Adresa
3	Beograd

3.5.5. ORDER BY клаузула

Коришћењем ове клаузуле сортира се резултат одређеног упита. Резултат може бити сортиран у растућем редоследу (нпр. од 1 па навише или од A до Z) и опадајућем редоследу. За растући редослед користи се наредба *ASC*, а за опадајући наредба *DESC*. Ова клаузула се увек налази на крају *SELECT* блока.

Синтакса је следећа:

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2, ... ASC|DESC;
```

Пример: Сви власници аутомобила који су из Београда. Резултати сортирани по имену власника, растући распоред.

```
SELECT * FROM Vlasnik
WHERE Adresa = 'Beograd'
ORDER BY Ime ASC;
```

Резултат је приказан у *табели 24.*

Табела 24. *Резултат упита*

Id	JMBG	Ime	Prezime	RegistarskiBroj	Adresa
2	2587412369854	Pera	Peric	115-UR-101	Beograd
4	2258874422589	Stefan	Nikolic	225-BG-225	Beograd
3	3248753698745	Uros	Urosevic	225-BG-689	Beograd

Пример: Сви власници аутомобила који су из Београда. Резултати сортирани по имену власника, опадајући распоред.

```
SELECT * FROM Vlasnik
WHERE Adresa = 'Beograd'
ORDER BY Ime DESC;
```

Резултат је приказан у *табели 25.*

Табела 25. *Резултат упита*

Id	JMBG	Ime	Prezime	RegistarskiBroj	Adresa
3	3248753698745	Uros	Urosevic	225-BG-689	Beograd
4	2258874422589	Stefan	Nikolic	225-BG-225	Beograd
2	2587412369854	Pera	Peric	115-UR-101	Beograd

3.6. Спајање табела - JOIN

Наредба *JOIN* служи за спајање две или више табела. Постоји више типова спајања у SQL-у и то:

- INNER JOIN
- LEFT JOIN
- RIGHT JOIN
- FULL OUTER JOIN

Начин рада *JOIN-ова* приказан је преко следећих табела.

Табела 26.

Predmet	IdKatedre
A	1
B	2
C	

Табела 27.

IdKatedre	NazivKatedre
1	Informatika
2	Fizika
3	Biologija

Табела 28. *INNER JOIN*

Predmet	NazivKatedre
A	Informatika
B	Fizika

Табела 29. *LEFT JOIN*

Predmet	NazivKatedre
A	Informatika
B	Fizika
C	NULL

Табела 30. *RIGHT JOIN*

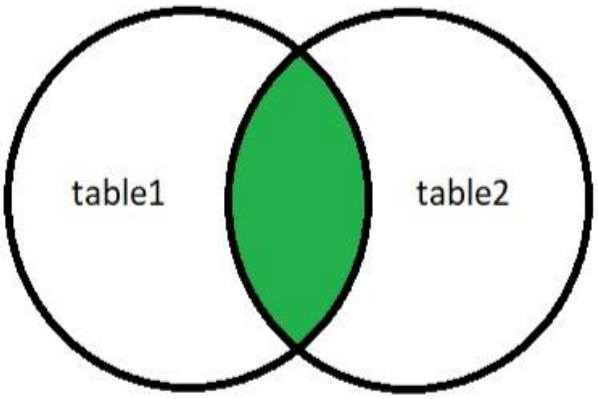
Predmet	NazivKatedre
A	Informatika
B	Fizika
NULL	Biologija

Табела 31. *FULL OUTER JOIN*

Product	Name
A	Informatika
B	Fizika
C	NULL
NULL	Biologija

3.6.1. INNER JOIN

***Inner join*:** Враћа записе који имају одговарајуће вредности у обе табеле (*слика 12*).



Слика 12. *INNER JOIN*

Синтакса:

```
SELECT column_name(s)
FROM table1
INNER JOIN table2 ON table1.column_name = table2.column_name;
```

Пример:

```
SELECT Ime, Prezime, Marka, Automobil.RegistarskiBroj
FROM Automobil
INNER JOIN Vlasnik
ON Automobil.RegistarskiBroj = Vlasnik.RegistarskiBroj
ORDER BY Prezime ASC;
```

Табела 32. *Табела "Automobil"*

RegistarskiBroj	Marka	DatumProizvodnje	Vrednost
025-KM-123	Audi	2019-02-16	1220000
114-KG-878	BMW	2018-11-28	960000
115-UR-101	Fiat	2019-01-02	380000
225-BG-225	BMW	2018-12-30	2360000
225-BG-689	Fiat	2018-12-25	440000
686-NI-878	Mercedes	2018-10-30	1960000

Табела 33. *Табела "Vlasnik"*

Id	JMBG	Ime	Prezime	RegistarskiBroj	Adresa
1	1255566688744	Marko	Markovic	025-KM-123	Kosovska Mitrovica
2	2587412369854	Pera	Peric	115-UR-101	Beograd
3	3248753698745	Uros	Urosevic	225-BG-689	Beograd
4	2258874422589	Stefan	Nikolic	225-BG-225	Beograd
5	2548889998745	Milan	Dejanovic	NULL	Kosovska Mitrovica

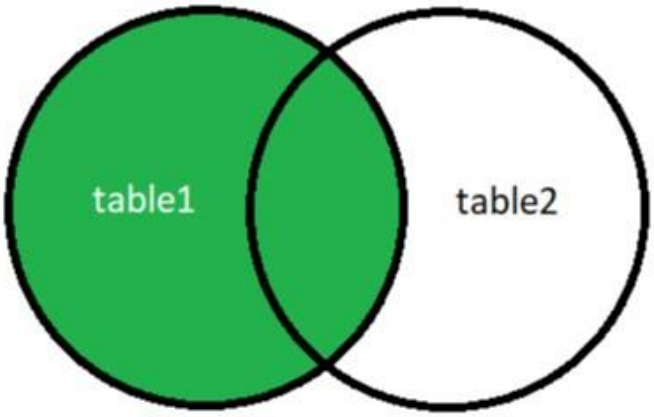
Резултат упита приказан је у *табели 34.*

Табела 34. *INNER JOIN табеле "Automobil" и табеле "Vlasnik"*

Ime	Prezime	Marka	RegistarskiBroj
Marko	Markovic	Audi	025-KM-123
Stefan	Nikolic	BMW	225-BG-225
Pera	Peric	Fiat	115-UR-101
Uros	Urosevic	Fiat	225-BG-689

3.6.2. LEFT JOIN

Left join: Враћа све записе из леве табеле и одговарајуће записе из десне табеле (*слика 13*).



Слика 13. *LEFT JOIN*

Синтакса:

```
SELECT column_name(s)
FROM table1
LEFT JOIN table2 ON table1.column_name = table2.column_name;
```

Пример:

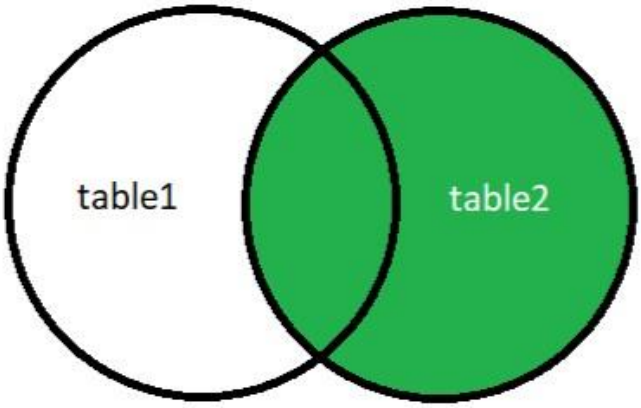
```
SELECT Ime, Prezime, Marka, Automobil.RegistarskiBroj
FROM Automobil
LEFT JOIN Vlasnik
ON Automobil.RegistarskiBroj = Vlasnik.RegistarskiBroj
ORDER BY Prezime ASC;
```

Табела 35. *LEFT JOIN*

Ime	Prezime	Marka	RegistarskiBroj
NULL	NULL	BMW	114-KG-878
NULL	NULL	Mercedes	686-NI-878
Marko	Markovic	Audi	025-KM-123
Stefan	Nikolic	BMW	225-BG-225
Pera	Peric	Fiat	115-UR-101
Uros	Urosevic	Fiat	225-BG-689

3.6.3. RIGHT JOIN

Right join: Враћа све записе из десне табеле и одговарајуће записе из леве табеле (*слика 14*).



Слика 14. *RIGHT JOIN*

Синтакса:

```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2 ON table1.column_name = table2.column_name;
```

Пример:

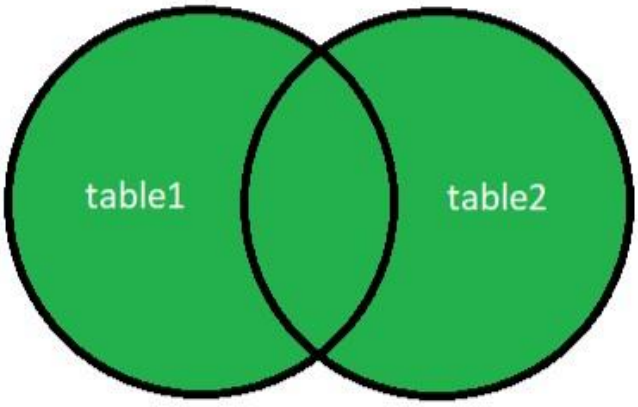
```
SELECT Ime, Prezime, Marka, Automobil.RegistarskiBroj
FROM Automobil
RIGHT JOIN Vlasnik
ON Automobil.RegistarskiBroj = Vlasnik.RegistarskiBroj
ORDER BY Prezime ASC;
```

Табела 36. *RIGHT JOIN*

Ime	Prezime	Marka	RegistarskiBroj
Milan	Dejanovic	NULL	NULL
Marko	Markovic	Audi	025-KM-123
Stefan	Nikolic	BMW	225-BG-225
Pera	Peric	Fiat	115-UR-101
Uros	Urosevic	Fiat	225-BG-689

3.6.4. FULL OUTER JOIN

Outer join: Враћа све записе и из леве и из десне табеле (*слика 15).*



Слика 15. *FULL OUTER JOIN*

Синтакса:

```
SELECT column_name(s)
FROM table1
FULL OUTER JOIN table2 ON table1.column_name = table2.column_name;
```

Пример:

```
SELECT Ime, Prezime, Marka, Automobil.RegistarskiBroj
FROM Automobil
FULL OUTER JOIN Vlasnik
ON Automobil.RegistarskiBroj = Vlasnik.RegistarskiBroj
ORDER BY Prezime ASC
```

Табела 37. *FULL OUTER JOIN*

Ime	Prezime	Marka	RegistarskiBroj
NULL	NULL	BMW	114-KG-878
NULL	NULL	Mercedes	686-NI-878
Milan	Dejanovic	NULL	NULL
Marko	Markovic	Audi	025-KM-123
Stefan	Nikolic	BMW	225-BG-225
Pera	Peric	Fiat	115-UR-101
Uros	Urosevic	Fiat	225-BG-689

3.7. Креирање погледа - VIEWS

Погледи (енг. *views*) су као и табеле састављени од поља, назива колона, редова, али за разлику од табела они не садрже сопствене податке већ податке добијају из табела на основу којих су и креирани. За креирање погледа користимо наредбу *CREATE VIEW*, док се за уништавање погледа користи наредба *DROP VIEW*.

Садржај погледа освежава се у тренутку извођења упита на основу садржаја табела над којим је поглед дефинисан, па се на основу тога свака промена одмах види у погледу који је везан са табелом где се промена догодила.

Погледи се углавном користе из следећих разлога:

1. Да би заштитили податке од прегледа сваког корисника. Нпр. можемо у погледу уклонити информације о матичном броју и адреси тако да осигурамо приватност тим особама, а онда свим запосленима можемо омогућити приступ дефинисаном погледу док само овлашћени корисници могу да виде табелу.

Пример:

```
CREATE VIEW PrikaziVlasnikeAutomobila(Ime, Prezime, Marka,
DatumProizvodnje) AS
SELECT Ime, Prezime, Marka, DatumProizvodnje
FROM Vlasnik
INNER JOIN Automobil
ON Vlasnik.RegistarskiBroj = Automobil.RegistarskiBroj;
```

Након креирања погледа, потребно је селектовати податке које садржи. То постижемо преко наредбе *SELECT* (као назив табеле уносимо назив креираног погледа):

```
SELECT * FROM PrikaziVlasnikeAutomobila;
```

Добијамо резултат приказан у *табели 38.*

Табела 38. *Креирани поглед*

Ime	Prezime	Marka	DatumProizvodnje
Marko	Markovic	Audi	2019-02-16
Pera	Peric	Fiat	2019-01-02
Uros	Urosevic	Fiat	2018-12-25
Stefan	Nikolic	BMW	2018-12-30

2. Други разлог јесте да би нам подаци били приказани у нама одговарајућем облику.

Због своје специфичности погледи имају нека ограничења и то:

- Не можемо користити *DELETE* наредбу над погледима који су креирани из више табела;
- Не можемо користити *INSERT* наредбу уколико сви *NOT NULL* атрибути нису у погледу;
- Уколико уносимо или мењамо податке коришћењем погледа који спаја више табела, сви атрибути које мењамо морају да припадају истој табели;
- Уколико користимо *DISTINCT* наредбу приликом креирања погледа не можемо мењати или уносити редове у поглед.

3.8. Индексирање

У индексу је сачувана локација записа на неком пољу или пољима. Индекси у табели се користе према принципу речника појмова у књизи - приликом тражења податка гледа се његова локација у индексу. Користе се да би се убрзала претрага и приступ подацима. Корисници не могу видети индексе, они се користе само за убрзавање претраге и упита. Код табела са малим бројем слогова није потребно користити индексе јер се неће приметити убрзање приликом претраживања или сортирања. Али ако постоји много података, индексирањем одређених поља може се очекивати значајно убрзање приликом претраге и сортирања.

Индекс може бити креиран одмах, док је табела на коју се односи празна, или може бити креиран накнадно. Ако се креира накнадно, индекс добија садржај који одговара садржају своје табеле. Од тада се садржај табеле

и индекса синхронизује. Било која промена података у колонама које су саставни део индексног израза рефлектује се и на сам садржај индекса.

Синтакса за креирање индекса:

```
CREATE INDEX index_name
ON table_name (column1, column2, ...);
```

Синтакса за креирање јединственог индекса:

```
CREATE UNIQUE INDEX index_name
ON table_name (column1, column2, ...);
```

Синтакса за брисање индекса:

```
DROP INDEX table_name.index_name;
```

Пример 1. Креирање индекса под називом “ind_imeprezime” над колонама *Ime* и *Prezime* табеле *Vlasnik*.

```
CREATE INDEX ind_imeprezime
ON Vlasnik (Ime, Prezime);
```

Пример 2. Брисање индекса.

```
DROP INDEX Vlasnik.ind_imeprezime
```

3.9. Трансакције

Трансакције су наредбе које служе за побољшање перформанси у оквиру неке базе. У ситуацијама када један или више корисника шаљу захтев за приступ деловима табела унутар базе, и све у истом тренутку, долази до преплитања упита и покренутих акција. У циљу решавања оваквих проблема, постоје наредбе које обезбеђују конкурентно извођење акција, а чак служе и за опоравак база података.

Трансакција се дефинише као јединица посла, коју чине једна или више *SQL* наредби које извршавају одређени скуп акција.

3.9.1. SET TRANSACTION наредба

Ова наредба контролише многе карактеристике промене података, најпре *read/write* карактеристике и ниво изолације трансакција. Када се трансакција извршава, све трансакције ће бити укључене, и оне које су наведене у наредби и оне које нису. Карактеристике које нису наведене ће преузети подразумеване вредности. Све карактеристике трансакције нису расположиве ни за једну трансакцију осим прве.

Синтакса наредбе *SET TRANSACTION:*

```
SET TRANSACTION ISOLATION LEVEL {READ COMMITTED | READ UNCOMMITTED
| REPEATABLE READ | SERIALIZABLE | SNAPSHOT}
```

3.9.2. BEGIN наредба

Ова наредба означава полазну тачку експлицитне, локалне трансакције. Експлицитне трансакције почињу са изразом *BEGIN TRANSACTION* и завршавају се са *COMMIT* или *ROLLBACK* изразом.

Синтакса:

```
BEGIN {TRAN | TRANSACTION}
[{transaction_name | @tran_name_variable}
[WITH MARK ['description']]]
```

3.9.3. COMMIT наредба

Ова наредба експлицитно завршава отворену трансакцију и чини промене сталним у бази података. Трансакције могу да буду отворене имплицитно као део наредбе *INSERT, UPDATE* или *DELETE*, или отворене експлицитно са наредбом *START*. У било ком случају, експлицитно издвајање наредбе *COMMIT* ће да заврши отворену трансакцију.

Синтакса:

```
COMMIT [{TRAN | TRANSACTION}
[transaction_name | @tran_name_variable]]
[WITH (DELAYED_DURABILITY = {OFF | ON})]
```

3.9.4. ROLLBACK наредба

Синтакса:

```
ROLLBACK {TRAN | TRANSACTION}
[transaction_name | @tran_name_variable
| savepoint_name | @savepoint_variable]
```

Наредба *ROLLBACK* поништава трансакције до њиховог почетка или до претходно дефинисане тачке - *SAVEPOINT*. Ова наредба такође затвара све отворене курсоре. У додатку поништавања једне операције за манипулацију подацима као што су наредбе *INSERT*, *UPDATE*, *DELETE*, наредба *ROLLBACK* поништава трансакције до последње издате наредбе *START TRANSACTION*, *SET TRANSACTION* или *SAVEPOINT*.

3.9.5. SAVEPOINT – SAVE наредба

Синтакса:

```
SAVE {TRAN | TRANSACTION}
{savepoint_name | @savepoint_variable}
```

Ова наредба поставља тачку чувања која се назива *savepoint_name* у оквиру текуће трансакције. Трансакције могу бити парцијално повраћене у ознаци тачке прекида коришћењем наредбе *ROLLBACK*. Тачке прекида су постављене у оквиру целе трансакције у којој су дефинисане. Имена тачака би требало да буду јединствена у оквиру њихових делокруга.

Пример 1: На овом примеру је демонстриран унос података у табелу *Automobil* употребом трансакције. Ова трансакција обухвата унос нових података о аутомобилу марке *Audi*. Вредност аутомобила једнака је највећој вредности аутомобила која се налази у табели *Automobil*.

```
BEGIN TRANSACTION
DECLARE @najvredniji int
SELECT @najvredniji = MAX(Vrednost) FROM Automobil
INSERT INTO Automobil(RegistarskiBroj, Marka, DatumProizvodnje,
Vrednost)
VALUES ('202-KG-335', 'Audi', '2019-02-02', @najvredniji)
IF @@ERROR <>0
      rollback transaction
```

```
ELSE
      commit transaction
```

Пример 2: На овом примеру је демонстриран унос података у табелу *Prodavnica* употребом трансакције. У трансакцији је коришћена системска функција *@@ERROR*, помоћу које се проверава исправност уписа. Ако је све исправно, односно вредности су унешене без грешке, исте се уписују у табелу. Уколико на неком делу трансакције постоји грешка, трансакција се поништава до следеће тачке чувања података.

```
BEGIN TRANSACTION prviPrimer
INSERT INTO Prodavnica VALUES(1, 'proizvod1', 100.50, 50)
      IF @@ERROR <> 0 ROLLBACK
            save transaction transakcija1
INSERT INTO Prodavnica VALUES(2, 'proizvod2', 110.50, 50);
      IF @@ERROR <> 0 ROLLBACK TRANSACTION transakcija1
            save transaction transakcija2
INSERT INTO Prodavnica VALUES(3, 'proizvod3', 120.50, 50)
      IF @@ERROR <> 0 ROLLBACK TRANSACTION transakcija2
            save transaction transakcija3
INSERT INTO Prodavnica VALUES(4, 'proizvod4', 130.50, 50)
      IF @@ERROR <> 0 ROLLBACK TRANSACTION transakcija3
            save transaction transakcija4
INSERT INTO Prodavnica VALUES(5, 'proizvod5', 140.50, 50)
      IF @@ERROR <> 0 ROLLBACK TRANSACTION transakcija4
            save transaction transakcija5
INSERT INTO Prodavnica VALUES(6, 'proizvod5', 150.50, 50)
      IF @@ERROR <> 0 ROLLBACK TRANSACTION transakcija5
            save transaction transakcija6
INSERT INTO Prodavnica VALUES(7, 'proizvod6', 160.50, 50)
      IF @@ERROR <> 0 ROLLBACK TRANSACTION transakcija6
            save transaction transakcija7
INSERT INTO Prodavnica VALUES(8, 'proizvod7', 170.50, 50)
      IF @@ERROR <> 0 ROLLBACK TRANSACTION transakcija7
            COMMIT TRANSACTION prviPrimer;
```

Пример 3. У овом примеру је приказан упис података у табелама *racun* и *stavka_racuna*. У случају грешке, трансакције се поништавају.

```
BEGIN TRANSACTION TransakcijaTri
INSERT INTO racun VALUES (1,' ',111)
IF @@ERROR != 0
      BEGIN
             ROLLBACK TRANSACTION drugiPrimer
            PRINT 'Nije dozvoljeno upisivanje u RACUN! Transakcija
```

```
                    je poništena'
              RETURN
       END
UPDATE prodavnica SET stanje = stanje - 10 WHERE proizvod = 5
IF @@ERROR != 0
       BEGIN
              ROLLBACK TRANSACTION drugiPrimer
               PRINT 'Nije dozvoljeno upisivanje u MAGACIN!
                      Transakcija je poništena'
              RETURN
       END
INSERT INTO stavka_racuna VALUES (1, 6, 10)
IF @@ERROR != 0
       BEGIN
               ROLLBACK TRANSACTION drugiPrimer
              PRINT 'Nije dozvoljeno upisivanje u STAVKA_RACUNA!
                      Transakcija je poništena'
        RETURN
END
COMMIT TRANSACTION drugiPrimer
       IF @@ERROR != 0
              PRINT 'Transakcija ne može da se potvrdi'
       ELSE
              PRINT 'Transakcija potvrđena';
```

3.10. SQL INJECTION техника

Напад уметањем *SQL* кода (енг. *SQL Injection*) сматра се једним од 10 најчешћих и најопаснијих напада на *web* странице. Овај облик напада искоришћава рањивост на слоју базе података. Напад *SQL INJECTION* техником могућ је због недовољних провера уноса корисника тј. када корисник унесе одређене податке, ти подаци се могу преобличити у *SQL* наредбе које могу непожељно деловати на базу података. Овај напад утиче на:

- Поверљивост – у базама података се најчешће налазе осетљиви подаци попут личних података о кориснику. Нападачи могу открити информације о адресама електронске поште корисника, што се касније може искористити за додатне нападе.
- Аутентикацију – коришћењем овог напада могуће је заобићи нормалну пријаву на систем. Нападач се, захваљујући овом облику напада, може

пријавити као неки други корисник без познавања одговарајуће лозинке.

- Ауторизацију – нападач може искористити податке у бази како би се пријавио као администратор и тако добио додатне дозволе над базом или web страницом.
- Интегритет података – једнако као што нападач може читати податке из базе, може их и мењати. У крајњем случају, нападач може уништити целу базу података.

Најпознатији облици напада уметањем *SQL* кода искоришћавају поља за унос података на *web* страницама попут поља за претрагу или поља за унос корисничког имена и лозинке. Напад је могуће извршити ако се подаци које корисник уноси не проверавају на одговарајући начин. Овај облик напада уметањем *SQL* кода се често зове уметање првог реда (енг. *first-order injection*) зато што злонамерно обликован *SQL* код који је нападач унео у поље за унос директно утиче на базу података и изводи се моментално.

Код уметањем другог реда (енг. *second-order injection*), нападачи прво постављају посебно обликоване податке у базу. Приликом каснијег коришћења тих података, они ће узроковати уметање *SQL* кода и остварiће напад. Приликом извођења ове технике нападач мора добро познавати како и када програм узима податке како би могао правилно обликовати свој *SQL* код.

У наставку ће бити представљени најчешћи облици напада уметањем *SQL* кода.

3.10.1. Примери напада уметањем SQL кода

Узроковање делимичног или потпуног уништења базе података или онемогућавање приступа бази легитимним корисницима врши се нападом одбијања услуге (енг. *Denial of Service - DoS*).

Једна од најопаснијих наредби, ове врсте напада јесте следећа наредба:

```
drop table table_name
```

Ова наредба брише целу табелу у бази, укључујући све податке који су се у том тренутку налазили у табели. Ако нападач успе да пронађе начин како уметнути *SQL* код са наредбом *drop table* могуће је уништење целе базе.

Уместо наредбе *drop table* може се користити наредба *shutdown* која гаси базу података, али је не уништава.

Уколико нападач жели да заобиђе легитимну пријаву на систем, најједноставнији пример овог облика напада представља поступак у којем приликом пријаве, корисник унесе у поља за корисничко име и шифру следећи код: ' OR '1'='1.

Након уноса овог кода, ако не постоји неки вид заштите од овог напада, као што су ограничења симбола који могу да се користе за шифру налога, створиће се следећи упит:

```
select KorisnickoIme
from Korisnici
where KorisnickoIme = '' OR '1'='1' and Lozinka = '' OR '1'='1'
```

Оно што се догађа у бази података јесте да се не извршава упоређивање података у табели са корисничким именом, него се проверава истинитост тврдње '1'='1'. Будући да је та тврдња увек истинита, цео упит ће бити истинит. Ово резултира враћањем првог реда у табели *Korisnici* чиме се нападач успешно пријавио у систем као корисник који је први наведен у табели.

3.10.2. Заштита од SQL Injection напада

Уз мале измене у програмском коду и увођења додатних провера могуће је одбранити базу података од већег броја напада овом техником. Препоручује се да се не креирају табеле са типичним именима. Уместо назива табеле *Korisnici* препоручљиво би било користити другачији назив попут *ListaKorisnika*. Такође, увек се препоручује ограничити приступ бази. Обичним корисницима би требало бити омогућено само извршење наредбе *select*, а забранити извођење наредби попут *insert*, *delete* или *drop table* које мењају податке у бази података.

Најважнији савет за заштиту од ове врсте напада јесте да се сви подаци које корисник уноси морају проверавати. Приликом логовања треба проверавати број знакова, провера наводника, јер у већини случајева корисници немају потребу за уношењем једноструких или двоструких наводника. Такође, треба проверавати да ли је унети знак тачка-зарез (;) или да ли су унете неке од *SQL* кључних речи попут *select*, *union*, *insert* и слично.

4. ПРОЈЕКТИ

У наредном делу биће приказани пројекти одређених база података. За први пројекат, осим ЕР дијаграма и упита, биће приказан и релациони модел као и *DDL* и *DML* наредбе. За остале пројекте биће приказани само ЕР дијаграми (због обимности *DDL* и *DML* наредби) и одговарајући упити за претраживање и ажурирање база.

4.1. Апотекарска установа

1. Пројектовати ЕЕР дијаграм базе података према приложеним захтевима. Пресликати добијени ЕЕР дијаграм у шему релационе базе података.

База података АПОТЕКАРСКА УСТАНОВА садржи следеће информације. За сваку апотеку се памти шифра, назив, адреса и количина сваког од лекова којима располаже. За сваког фармацеута памти се презиме, име, матични број, адреса, стручна спрема и све апотеке дате апотекарске установе у којима је био запослен, као и датум запослења и време проведено у датој апотеци (број месеци и број дана). За сваки лек памти се шифра и назив лека, као и просечне месечне потребе датог лека за сваку од апотека. За сваку веледрогерију (добављач лекова) памти се шифра, назив, адреса и телефон. Од разних веледрогерија приликом пријема лекова апотекарска установа добија различите документе, али сви они садрже податке о датуму испоруке, укупном износу, року и начину плаћања, као и податке о количини сваког од лекова који се испоручују. Приликом пријема ових докумената они се шифрирају, јединствено, на нивоу апотекарске установе. За сваки документ памти се комерцијалиста који га је примио. За сваког комерцијалисту памти се презиме, име, матични број, адреса и број мобилног телефона. Лекови добијени у некој испоруци распоређују се по апотекама, па је за сваку испоруку датог лека потребно евидентирати све апотеке којима је испоручен и количина која је испоручена. Апотеци се може испоручити и лек којим она не располаже. За сваки лек потребно је пратити којим лековима може бити замењен. Дати лек може бити замена за више лекова.

2. На основу ЕЕР модела и одговарајућег релационог модела из претходног примера написати *SQL* наредбе за креирање базе података АПОТЕКАРСКА

УСТАНОВА (направити одговарајући *DDL* фајл). Код свих табела навести неопходна ограничења: дефинисати примарне, стране кључеве и дефинисати *CHECK* ограничења за вредности неких атрибута ако је то потребно.

3. У *Microsoft SQL Server Managment Studio*-у покренути *query* са претходно урађеним *DDL* фајлом. У поменутом алату нацртати одговарајући дијаграм на основу формираних табела и упоредити добијени дијаграм са пројектованим ЕЕР дијаграмом.

4. Претраживање базе података

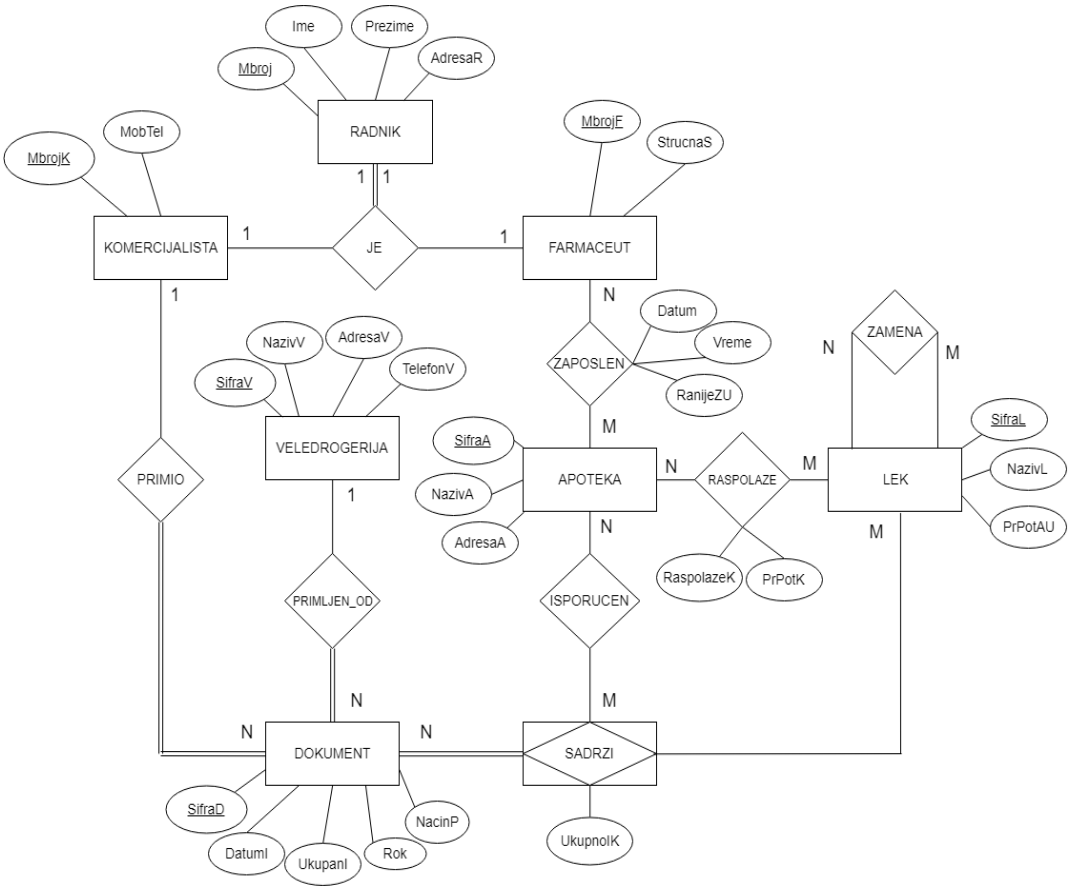
Дефинисати наредбе *SQL*-а за добијање одговора на следеће упите:

1. Приказати податке о веледрогеријама које су испоручиле лекове 15. децембра 2017. год.
2. Пронаћи све апотеке којима је дана 15. децембра 2017. год. испоручен лек "*Febricet*".
3. Приказати податке о фармацеутима који раде у апотекама којима је веледрогерија "*Hemofarm*" у задњих 10 дана испоручивала неки лек.

5. Ажурирање базе података

Дефинисати наредбе *SQL*-а за ажурирање базе података у складу са следећим захтевима:
1. Додати у бази да замена за лек "Фебрицет" може бити лек "Парацетамол".
2. За веледрогерију "Галеника" повећати рок плаћања на 90 дана.
3. Обрисати податке о фармацеутима који су били запошљени у апотеци Дона у периоду од 15. октобра 2016. год. до 15. децембра 2016. год.

EEP дијаграм



Слика 16. *EEP дијаграм базе „Апотекарска установа“*

Релациона шема

RADNIK			
<u>Mbroj</u>	Ime	Prezime	AdresaR

FARMACEUT	
<u>*MbrojF*</u>	StrucnaS

KOMERCIJALISTA	
<u>*MbrojK*</u>	MobTel

VELEDROGERIJA			
<u>SifraV</u>	NazivV	AdresaV	TelefonV

LEK		
SifraL	NazivL	PrPotAU

DOKUMENT						
SifraD	DatumI	UkupanI	Rok	NacinP	*MbrojK*	*SifraV*

RASPOLAZE			
SifraA	*SifraL*	RaspolazeK	PrPotK

APOTEKA		
SifraA	NazivA	AdresaA

SADRZI		
SifraD	*SifraL*	UkupnoIK

ISPORUCEN			
SifraD	*SifraL*	*SifraA*	UkupnoIKA

ZAMENA	
SifraLO	*SifraLZ*

ZAPOSLEN				
MbrojF	*SifraA*	RanijeZU	Datum	Vreme

DDL наредбе

```
create database ApotekarskaUstanova;

go

use ApotekarskaUstanova;

go

create table RADNIK (
Mbroj varchar(13) primary key not null,
Ime varchar(20) not null,
Prezime varchar(30) not null,
AdresaR varchar(30) not null,
 );
```

```
create table FARMACEUT (
MbrojF varchar(13) primary key not null,
StrucnaS varchar(20) not null
 );

create table KOMERCIJALISTA (
MbrojK varchar(13) primary key not null,
MobTel varchar(20) not null
 );

create table LEK (
SifraL varchar(10) primary key not null,
NazivL varchar(30),
PrPotAU int not null
 );

create table VELEDROGERIJA (
SifraV varchar(10) primary key not null,
NazivV varchar(20) not null,
AdresaV varchar(30) not null,
TelefonV varchar(20) not null
);

create table DOKUMENT (
SifraD varchar(10) primary key not null,
DatumI date not null,
UkupanI int not null,
Rok int not null,
NacinP varchar(20) not null,
SifraV varchar(10),
MbrojK varchar(13)
);

create table APOTEKA (
SifraA varchar(10) primary key not null,
NazivA varchar(20) not null,
AdresaA varchar(30) not null
);

create table RASPOLAZE (
SifraA varchar(10) not null,
SifraL varchar(10) not null,
RaspolazeK int not null,
PrPotK int not null
);
```

```
create table SADRZI (
SifraD varchar(10) not null,
SifraL varchar(10) not null,
UkupnoIK int not null
);

create table ISPORUCENO (
SifraD varchar(10) not null,
SifraL varchar(10) not null,
SifraA varchar(10) not null,
UkupnoIKA int not null
);

create table ZAMENA (
SifraLO varchar(10) not null,
SifraLZ varchar(10) not null,
);

create table ZAPOSLEN (
MbrojF varchar(13) not null,
SifraA varchar(10) not null,
RanijeZU varchar(30) not null,
Datum date not null,
Vreme time not null
);

alter table RASPOLAZE
add constraint PK_RASPOLAZE primary key (
SifraA,
SifraL
);

alter table SADRZI
add constraint PK_SADRZI primary key (
SifraD,
SifraL
);

alter table ISPORUCENO
add constraint OK_ISPORUCENO primary key (
SifraD,
SifraL,
SifraA
);

alter table ZAMENA
```

```
add constraint PK_ZAMENA primary key (
SifraLO,
SifraLZ
);

alter table ZAPOSLEN
add constraint PK_ZAPOSLEN primary key (
MbrojF,
SifraA,
RanijeZU
);

alter table FARMACEUT
add constraint KF_FARMACEUT
foreign key (MbrojF) references RADNIK(Mbroj)
on delete cascade
on update cascade;

alter table KOMERCIJALISTA
add constraint FK_KOMERCIJALISTA
foreign key (MbrojK) references RADNIK(Mbroj)
on delete cascade
on update cascade;

alter table DOKUMENT
add constraint FK_DOKUMENT
foreign key (SifraV) references VELEDROGERIJA(SifraV),
foreign key (MbrojK) references KOMERCIJALISTA(MbrojK)
on delete cascade
on update cascade;

alter table RASPOLAZE
add constraint FK_RASPOLAZE
foreign key (SifraA) references APOTEKA(SifraA),
foreign key (SifraL) references LEK(SifraL)
on delete cascade
on update cascade;

alter table SADRZI
add constraint FK_SADRZI
foreign key (SifraD) references DOKUMENT(SifraD),
foreign key (SifraL) references LEK(SifraL)
on delete cascade
on update cascade;

alter table ISPORUCENO
```

```
add constraint FK_ISPORUCENO
foreign key (SifraD,SifraL) references SADRZI(SifraD,SifraL),
foreign key (SifraA) references APOTEKA(SifraA)
on delete cascade
on update cascade;
alter table ZAMENA
add constraint FK_ZAMENA
foreign key (SifraLO) references LEK(SifraL),
foreign key (SifraLZ) references LEK(SifraL)
on delete cascade
on update cascade;

alter table ZAPOSLEN
add constraint FK_ZAPOSLEN
foreign key (MbrojF) references FARMACEUT(MbrojF),
foreign key (SifraA) references APOTEKA(SifraA)
on delete cascade
on update cascade;
```

DML наредбе

```
use ApotekarskaUstanova;

go

Insert into RADNIK
Values (123, 'Ivan', 'Ivanovic', 'Beograd')
Insert into RADNIK
Values (124, 'Branko', 'Ivanovic', 'Beograd1')
Insert into RADNIK
Values (125, 'Ranko', 'Ivanovic', 'Beograd2')
Insert into RADNIK
Values (126, 'Marko', 'Ivanovic', 'Beograd3')
Insert into RADNIK
Values (127, 'Janko', 'Ivanovic', 'Beograd4')
Insert into RADNIK
Values (128, 'Sinisa', 'Ivanovic', 'Beograd5')
Insert into RADNIK
Values (129, 'Bora', 'Ivanovic', 'Beograd6')
Insert into RADNIK
Values (130, 'Milan', 'Maksimovic', 'Novi Sad')
Insert into RADNIK
Values (131, 'Uros', 'Urosevic', 'Vrsac')
Insert into RADNIK
Values (132, 'Nikola', 'Nikolic', 'Vranje')
```

```
go

Insert into APOTEKA
Values (11111, 'Galenika', 'Ustanicka 22, Beograd')
Insert into APOTEKA
Values (22222, 'Hemofarm', 'Ustanicka 26, Beograd')
Insert into APOTEKA
Values (33333, 'Drzavna', 'Ratnih profitera 23, Zrenjanin')
Insert into APOTEKA
Values (44444, 'Moc prirode', 'Ratnih profitera 24, Nis')
Insert into APOTEKA
Values (55555, 'Vels', 'Bosko Buha, Kosovska Mitrovica')
Insert into APOTEKA
Values (66666, 'Dona', 'Banovo Brdo')
Insert into APOTEKA
Values (77777, 'Benu', 'Zemun')

go

Insert into VELEDROGERIJA
Values ('1', 'Prva', 'Nis', '06456555')
Insert into VELEDROGERIJA
Values ('2', 'Druga', 'Beograd', '065456454')
Insert into VELEDROGERIJA
Values ('3', 'Treca', 'Novi Sad', '060456456')
Insert into VELEDROGERIJA
Values ('4', 'Cetvrta', 'Gnjilane', '061456456')
Insert into VELEDROGERIJA
Values ('5', 'Peta', 'Pristina', '069786786')
Insert into VELEDROGERIJA
Values ('6', 'Galenika', 'Zemun', '0690024345')
Insert into VELEDROGERIJA
Values ('7', 'Hemofarm', 'Vrsac', '0645005114')

go

Insert into KOMERCIJALISTA
Values (123, '0642255877')
Insert into KOMERCIJALISTA
Values (124, '065666575')
Insert into KOMERCIJALISTA
Values (125, '066002547')

go

Insert into FARMACEUT
```

```
Values (126, 'Gimnazija')
Insert into FARMACEUT
Values (127, 'Medicina')
Insert into FARMACEUT
Values (128, 'Farmacija')
Insert into FARMACEUT
Values (129, 'Medicina')
Insert into FARMACEUT
Values (130, 'Farmacija')
Insert into FARMACEUT
Values (131, 'Medicina')
Insert into FARMACEUT
Values (132, 'Medicina')

go

Insert into DOKUMENT
Values (11111, '2017-12-15', 55, 2, 'kes', 1, 125)
Insert into DOKUMENT
Values (22222, '2017-12-15', 66, 7, 'kartica', 1, 123)
Insert into DOKUMENT
Values (33333, '2017-12-15', 11, 3, 'cekovi', 2, 123)
Insert into DOKUMENT
Values (44444, '2017-02-11', 55, 5, 'kes', 3, 124)
Insert into DOKUMENT
Values (55555, '2017-06-01', 44, 6, 'kartica', 1, 124)
Insert into DOKUMENT
Values (66666, '2017-09-12', 33, 3, 'cekovi', 4, 125)
Insert into DOKUMENT
Values (77777, '2017-12-12', 22, 7, 'cekovi', 4, 125)
Insert into DOKUMENT
Values (88888, '2018-12-12', 44, 2, 'kes', 6, 125)
Insert into DOKUMENT
Values (99999, '2018-12-11', 44, 2, 'kartica', 6, 125)
Insert into DOKUMENT
Values (11112, '2017-12-15', 30, 10, 'kartica', 6, 125)
Insert into DOKUMENT
Values (11113, '2019-01-20', 15, 6, 'kes', 7, 124)
Insert into DOKUMENT
Values (11114, '2019-01-20', 15, 2, 'kartica', 7, 125)
Insert into DOKUMENT
Values (11115, '2019-01-19', 19, 7, 'kartica', 7, 125)

go

Insert into LEK
```

```
Values (234, 'Fabricet', 42)
Insert into LEK
Values (235, 'Paracetamol', 15)
Insert into LEK
Values (236, 'Paracetamol Duo', 39)
Insert into LEK
Values (237, 'Espumisan', 45)
Insert into LEK
Values (238, 'Ranisan', 21)
Insert into LEK
Values (239, 'Ranitadin', 3)
Insert into LEK
Values (240, 'Beli slez', 67)
Insert into LEK
Values (241, 'Beli slez Deciji', 90)
Insert into LEK
Values (242, 'Sinacilin', 47)

go

Insert into RASPOLAZE
Values (11111, 241, 900, 45)
Insert into RASPOLAZE
Values (22222, 238, 800, 60)
Insert into RASPOLAZE
Values (33333, 237, 330, 30)
Insert into RASPOLAZE
Values (11111, 234, 600, 24)
Insert into RASPOLAZE
Values (44444, 241, 700, 62)
Insert into RASPOLAZE
Values (11111, 235, 100, 51)
Insert into RASPOLAZE
Values (55555, 239, 300, 57)
Insert into RASPOLAZE
Values (33333, 242, 100, 35)

go

Insert into sadrzi
Values (77777, 234, 10)
Insert into sadrzi
Values (33333, 235, 15)
Insert into sadrzi
Values (44444, 236, 20)
Insert into sadrzi
```

```
Values (55555, 240, 30)
Insert into sadrzi
Values (11111, 239, 9)
Insert into sadrzi
Values (22222, 238, 14)
Insert into sadrzi
Values (55555, 236, 6)
Insert into sadrzi
Values (66666, 235, 24)
Insert into sadrzi
Values (11112, 234, 10)
Insert into sadrzi
Values (11113, 239, 10)
Insert into sadrzi
Values (11114, 242, 10)
Insert into sadrzi
Values (11115, 240, 5)

go

Insert into ISPORUCENO
Values (11111, 239, 11111, 50)
Insert into ISPORUCENO
Values (22222, 238, 11111, 80)
Insert into ISPORUCENO
Values (11111, 239, 22222, 25)
Insert into ISPORUCENO
Values (44444, 236, 33333, 42)
Insert into ISPORUCENO
Values (55555, 236, 44444, 32)
Insert into ISPORUCENO
Values (66666, 235, 55555, 15)
Insert into ISPORUCENO
Values (11111, 239, 44444, 60)
Insert into ISPORUCENO
Values (11112, 234, 11111, 12)
Insert into ISPORUCENO
Values (11113, 239, 22222, 9)
Insert into ISPORUCENO
Values (11114, 242, 22222, 10)
Insert into ISPORUCENO
Values (11115, 240, 33333, 10)

go

Insert into ZAMENA
```

```
Values (234, 237)
Insert into ZAMENA
Values (234, 239)
Insert into ZAMENA
Values (235, 238)
Insert into ZAMENA
Values (236, 239)

go

Insert into ZAPOSLEN
Values (126, 11111, 'Hemofarm', '2017-12-12', '23:44:55')
Insert into ZAPOSLEN
Values (126, 22222, 'Dona', '2016-11-16', '2016-12-14')
Insert into ZAPOSLEN
Values (127, 33333, 'Moc prirode', '2017-12-12', '2016-12-16')
Insert into ZAPOSLEN
Values (128, 44444, 'Dona', '2016-10-16', '2016-12-14')
Insert into ZAPOSLEN
Values (129, 55555, 'Dona', '2017-12-12', '2016-12-19')
Insert into ZAPOSLEN
Values (128, 66666, 'Hemofarm', '2017-12-12', '2016-11-14')
Insert into ZAPOSLEN
Values (127, 33333, 'Dona', '2017-12-12', '2016-11-14')
Insert into ZAPOSLEN
Values (128, 66666, 'Galenika', '2017-12-12', '2016-10-14')
Insert into ZAPOSLEN
Values (127, 33333, 'Galenika', '2017-12-12', '2016-10-11')
Insert into ZAPOSLEN
Values (126, 11111, 'Dona', '2017-12-12', '23:44:55')
Insert into ZAPOSLEN
Values (127, 11111, 'Dona', '2017-12-13', '19:23:11')
Insert into ZAPOSLEN
Values (129, 11111, 'Dona', '2017-12-15', '14:51:01')
Insert into ZAPOSLEN
Values (130, 33333, 'Dona', '2017-12-15', '17:55:01')
Insert into ZAPOSLEN
Values (131, 33333, 'Moc prirode', '2016-10-15', '13:55:01')
Insert into ZAPOSLEN
Values (132, 33333, 'Galenika', '2016-05-05', '13:00:01')
Insert into ZAPOSLEN
Values (132, 44444, 'Hemofarm', '2016-02-02', '19:20:01')
```

Претраживање базе

Задатак 1: Приказати податке о веледрогеријама које су испоручиле лекове 15. децембра 2017. год.

```
select distinct VELEDROGERIJA.SifraV, VELEDROGERIJA.NazivV,
VELEDROGERIJA.AdresaV, VELEDROGERIJA.TelefonV, DOKUMENT.DatumI
from  DOKUMENT inner join VELEDROGERIJA
on  DOKUMENT.SifraV = VELEDROGERIJA.SifraV
where  DOKUMENT.DatumI = '2017-12-15';
```

Задатак 2: Пронаћи све апотеке којима је дана 15. децембра 2017. год. испоручен лек „Фабрицет“.

```
select ISPORUCENO.SifraA, APOTEKA.NazivA, LEK.NazivL,
DOKUMENT.DatumI
from DOKUMENT
inner join ISPORUCENO
on DOKUMENT.SifraD = ISPORUCENO.SifraD
inner join APOTEKA
on APOTEKA.SifraA = ISPORUCENO.SifraA
inner join LEK
on LEK.SifraL = ISPORUCENO.SifraL
where LEK.NazivL = 'Fabricet' and DOKUMENT.DatumI = '2017-12-15';
```

Задатак 3: Приказати податке о фармацеутима који раде у апотекама којима је веледрогерија „Хемофарм“ у задњих 10 дана испоручивала неки лек.

```
select distinct RADNIK.Ime, RADNIK.Prezime, RADNIK.AdresaR,
FARMACEUT.StrucnaS
from VELEDROGERIJA
inner join DOKUMENT
on VELEDROGERIJA.SifraV = DOKUMENT.SifraV
inner join ISPORUCENO
on DOKUMENT.SifraD = ISPORUCENO.SifraD
inner join APOTEKA
on ISPORUCENO.SifraA = APOTEKA.SifraA
inner join ZAPOSLEN
on APOTEKA.SifraA = ZAPOSLEN.SifraA
inner join FARMACEUT
on ZAPOSLEN.MbrojF = FARMACEUT.MbrojF
inner join RADNIK
```

```
on FARMACEUT.MbrojF = RADNIK.Mbroj
where VELEDROGERIJA.NazivV = 'Hemofarm' and DOKUMENT.DatumI >=
DATEADD(day,-10, GETDATE());
```

Ажурирање базе

Задатак 1: Додати у бази да замена за лек „Фабрицет“ може бити лек „Парацетамол“.

```
/* Najpre pretrazimo sifre lekova a zatim azuriramo na osnovu tih
sifri */
select SifraL, NazivL, SifraLZ from LEK
inner join ZAMENA
on LEK.SifraL = ZAMENA.SifraLO
where NazivL = 'Fabricet' OR NazivL = 'Paracetamol'

/*Zbog toga sto lek Fabricet vec ima dva leka upisana u bazi
podataka kojima se moze zameniti, potrebno je odabrati sifru
postojeceg leka za zamenu koju zelimo da azuriramo. Sifra leka
Fabricet je 234, sifra leka Paracetamol je 235.*/

update zamena
set SifraLZ = 235
where SifraLO = 234 and SifraLZ = 239;
```

Задатак 2: За веледрогерију „Галеника“ повећати рок плаћања на 90 дана.

```
update DOKUMENT
set DOKUMENT.Rok = 90
from DOKUMENT
inner join VELEDROGERIJA
on DOKUMENT.SifraV = VELEDROGERIJA.SifraV
where VELEDROGERIJA.NazivV = 'Galenika';
```

Задатак 3: Обрисати податке о фармацеутима који су били запослени у апотеци „Дона“ у периоду од 15. октобра 2016. год. до 15. децембра 2016. год.

```
delete from radnik where Mbroj
in (select farmaceut.MbrojF from farmaceut where MbrojF
in (select ZAPOSLEN.MbrojF from ZAPOSLEN where RanijeZU = 'Dona'
and zaposen.Datum between '2016-10-15' and '2016-12-15'))
```

4.2. Грађевинско предузеће

1. Пројектовати EER дијаграм базе података према приложеним захтевима. Пресликати добијени EER дијаграм у шему релационе базе података.

База података ГРАЂЕВИНСКО ПРЕДУЗЕЋЕ садржи податке потребне за вођење евиденције о ресурсима грађевинског предузећа (ГП). ГП изводи радове на више градилишта. За извођење тих радова потребне су разне машине и радници. За свако градилиште прати се шифра, назив и врста објекта, шеф градилишта (који је радник предузећа) и пројектант објекта (који не мора бити радник предузећа), датум почетка радова и рок завршетка. За сваки тип машине прати се назив, време потребно за монтажу и време потребно за демонтажу датог типа машине, као и сви радници који су обучени за рад на датом типу машине. ГП поседује више машина датог типа. За сваку конкретну машину прати се датум набавке, цена и амортизована вредност, као и сва градилишта на којима је била (јесте или ће бити) ангажована, датум почетка и трајање ангажмана (у данима), као и статус ангажмана (посао завршен, посао у току или резервисан за термин). За сваког радника евидентирају се сва градилишта на којима је био (јесте или ће бити) ангажован, датум ангажовања и трајања ангажовања (у данима). За свако градилиште прати се број машина датог типа које је потребно ангажовати као и датум почетка ангажовања и трајање ангажовања у данима. За сваког радника ангажованог на датом градилишту евидентира се број часова рада на машини ангажованој на датом градилишту уколико је радник обучен за рад на датом типу машине.

2. На основу EER модела и одговарајућег релационог модела из претходног примера написати *SQL* наредбе за креирање базе података ГРАЂЕВИНСКО ПРЕДУЗЕЋЕ (направити одговарајући *DDL* фајл). Код свих табела навести неопходна ограничења: дефинисати примарне, стране кључеве и дефинисати *CHECK* ограничења за вредности неких атрибута ако је то потребно.

3. У *Microsoft SQL Server Management Studio*-у покренути qуеру са претходно урађеним *DDL* фајлом. У поменутом алату нацртати одговарајући дијаграм на основу формираних табела и упоредити добијени дијаграм са пројектованим EER дијаграмом.

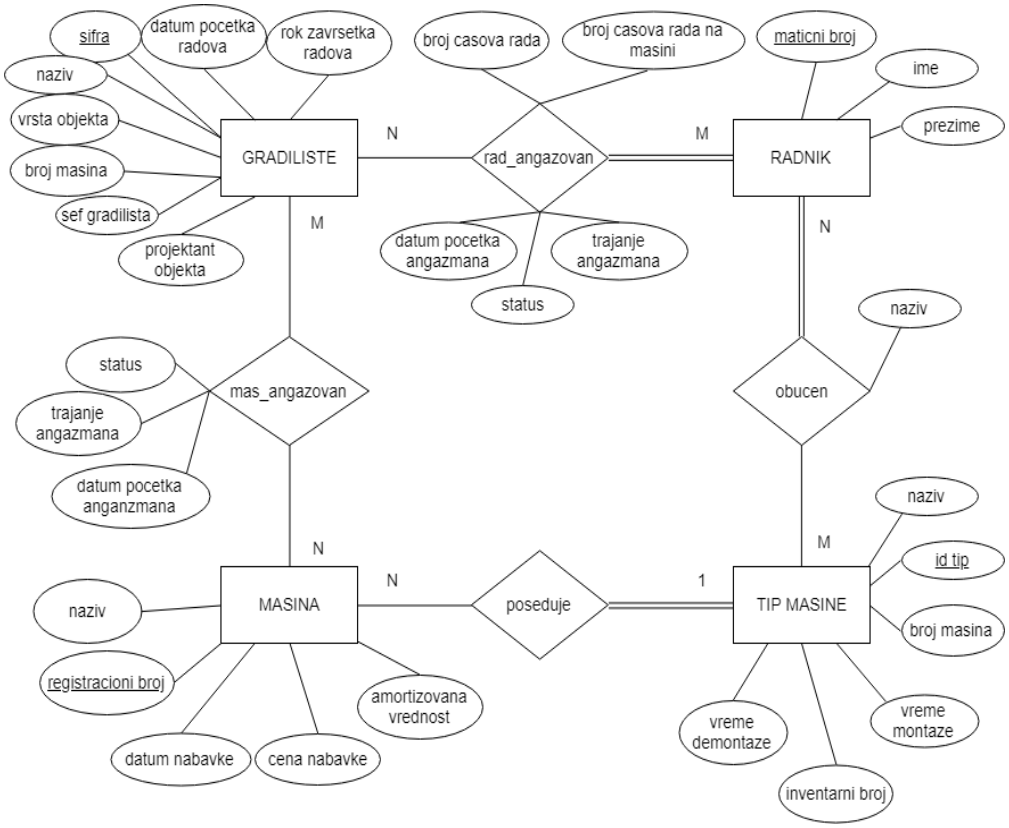
4. Претраживање базе података

Дефинисати наредбе *SQL*-а за добијање одговора на следеће упите:
1. Пронаћи све раднике који су обучени за рад на машини “Багер”.
2. Приказати све шефове градилишта на којима је била ангажована машина “Грејдер”.
3. Приказати све грађевинске машине које треба ангажовати на свим градилиштима дана 15. марта 2018. год.

5. Ажурирање базе података

Дефинисати наредбе *SQL*-а за ажурирање базе података у складу са следећим захтевима:
1. Обрисати сва забележена ангажовања грађевинских машина на градилишту под шифром 435.
2. За тип машине “Грејдер”, време монтаже поставити на 3 дана.
3. За сва градилишта где је шеф Јовановић Петар, смањити време планираног ангажовања грађевинских машина за 1 дан.

EER дијаграм



Слика 17. *EER дијаграм базе „Грађевинско предузеће“*

Претраживање базе

Задатак 1: Пронаћи све раднике који су обучени за рад на машини “Багер”.

```
select ime, prezime, tip_masine.naziv
from radnik inner join obucen
on radnik.[maticni broj] = obucen.[maticni broj]
inner join tip_masine
on obucen.[id tip] = tip_masine.[id tip]
where tip_masine.naziv = 'Bager';
```

Задатак 2: Приказати све шефове градилишта на којима је била ангажована машина “Грејдер”.

```
select gradiliste.[sef gradilista], gradiliste.naziv
from gradiliste inner join mas_angazovan
on gradiliste.sifra = mas_angazovan.sifra
inner join masina
on mas_angazovan.[registracioni broj] =
masina.[registracioni broj]
where masina.naziv = 'Grejder';
```

Задатак 3: Приказати све грађевинске машине које треба ангажовати на свим градилиштима дана 15. марта 2018. год.

```
select masina.naziv, masina.[registracioni broj]
from gradiliste inner join mas_angazovan
on gradiliste.sifra = mas_angazovan.sifra
inner join masina
on mas_angazovan.[registracioni broj]=
masina.[registracioni broj]
where '2018-03-15' between gradiliste.[datum pocetka radova] and
gradiliste.[rok zavrsetka radova]
```

Ажурирање базе

Задатак 1: Обрисати сва забележена ангажовања грађевинских машина на градилишту под шифром 435.

```
delete from mas_angazovan where mas_angazovan.sifra = 435;
```

Задатак 2: За тип машине “Грејдер”, време монтаже поставити на 3 на дана.

```
update tip_masine set tip_masine.[vreme montaze] = 3
where tip_masine.naziv = 'Grejder';
```

Задатак 3: За сва градилишта где је шеф Јовановић Петар, смањити време планираног ангажовања грађевинских машина за 1 дан.

```
update mas_angazovan
set mas_angazovan.[trajanje angazmana] =
mas_angazovan.[trajanje angazmana] - 1
from  gradiliste
inner join mas_angazovan
on gradiliste.sifra = mas_angazovan.sifra
where gradiliste.[sef gradilista] = 'Jovanovic Petar';
```

4.3. Градско саобраћајно предузеће

1. Пројектовати EER дијаграм базе података према приложеним захтевима. Пресликати добијени EER дијаграм у шему релационе базе података.

База података ГРАДСКО САОБРАЋАЈНО ПРЕДУЗЕЋЕ треба да садржи следеће податке. За аутомеханичара се памти: презиме, име, адреса, телефон и стручна спрема. За возаче се памти: презиме, име, адреса, телефон, категорија возачке дозволе и здравствено осигурање (описује се шта све има осигурано). За сваки аутобус се памти: регистарски број, тип, година набавке и да ли је активан или не. Аутобуске линије су шифриране и за сваку се памти полазна станица, крајња станица и трајање вожње. Евидентирају се и подаци о резервним деловима и то: назив резервног дела, јединица мере, тренутне залихе и минималне дозвољене залихе. Резервни део може, у случају недостатка, бити замењен неким другим резервним делом. За сваки резервни део евидентира се којим све резервним деловима може бити замењен, као и текстуално наведен услов замене. Један резервни део може бити замена за више других резервних делова. Сервисирање аутобуса се врши у самом предузећу. Сервисне услуге су ширфиране и за сваку од њих се памти: назив, колико пута се извршава у току године, као и количина сваког од резервних делова који су потребни за њено извршење. Једна сервисна

услуга може се извршавати више пута у току године на истом аутобусу. За сваки аутобус се прате све сервисне услуге које су над њим извршене, датум извршења, као и аутомеханичар које је обавио дато извршење. Поред тога за свако извршење сервисне услуге над датим аутобусом, потребно је памтити и потрошене количине сваког од резервних делова. За сваки аутобус се евидентирају возачи који су на њега распоређени, као и датум када је то распоређивање извршено. Памте се само актуелна распоређивања. За сваки аутобус се евидентира и линија на којој саобраћа и датум и време када је распоређен на дату линију. Еведентирају се само актуелна распоређивања аутобуса.

2. На основу EER модела и одговарајућег релационог модела из претходног примера написати *SQL* наредбе за креирање базе података ГРАДСКО САОБРАЋАЈНО ПРЕДУЗЕЋЕ (направити одговарајући *DDL* фајл). Код свих табела навести неопходна ограничења: дефинисати примарне, стране кључеве и дефинисати *CHECK* ограничења за вредности неких атрибута ако је то потребно.

3. У *Microsoft SQL Server Management Studio* покренути *query* са претходно урађеним *DDL* фајлом. У поменутом алату нацртати одговарајући дијаграм на основу формираних табела и упоредити добијени дијаграм са пројектованим EER дијаграмом.

4. Претраживање базе података

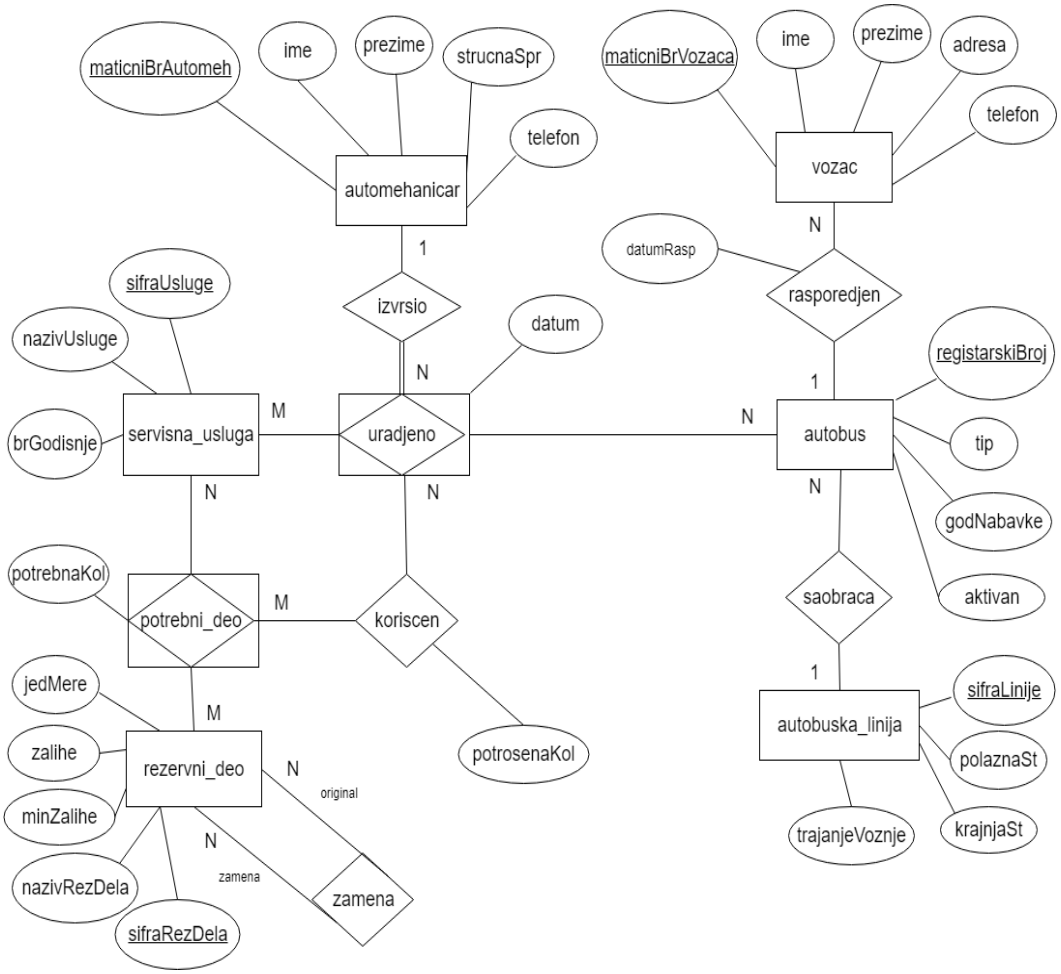
Дефинисати наредбе *SQL*-а за добијање одговора на следеће упите:
1. Приказати податке о резервним деловима који су потребни за сервисну услугу “замена зупчастог каиша”.
2. Приказати податке о свим сервисима који су извшени над аутобусом број 405 у 2015. години. Такође приказати податке о аутомеханичарима који су урадили сервисе.
3. Приказати податке о возачима и аутобусима који сутра треба да изађу да врше превоз путника.

5. Ажурирање базе података

Дефинисати наредбе *SQL*-а за ажурирање базе података у складу са следећим захтевима:
1. Обрисати податке о аутомеханичару Драгану Јанковићу.

2. У свим подацима о сервисима које је радио Драган Јанковић заменити име аутомеханичара у Зоран Крстић
3. Пребацити аутобус 405 са линије 65Л на линију 3.

EER дијаграм



Слика 18. *EER дијаграм базе „Градско саобраћајно предузеће“*

Претраживање базе

Задатак 1: Приказати податке о резервним деловима који су потребни за сервисну услугу “замена зупчастог каиша”.

```
select rezervni_deo.sifraRezDela, rezervni_deo.nazivRezDela,
potrebnaaKol, servisna_usluga.nazivUsluge
from rezervni_deo inner join potrebni_deo
on rezervni_deo.sifraRezDela = potrebni_deo.sifraRezDela
inner join servisna_usluga
on potrebni_deo.sifraUsluge = servisna_usluga.sifraUsluge
where servisna_usluga.nazivUsluge = 'zam zupcastog kaisa';
```

Задатак 2: Приказати податке о свим сервисима који су извшени над аутобусом број 405 у 2015. години. Такође приказати податке о аутомеханичарима који су урадили сервисе.

```
select servisna_usluga.sifraUsluge, servisna_usluga.nazivUsluge,
uradjeno.datum,
uradjeno.registarskiBroj, automehanicar.ime, automehanicar.prezime,
automehanicar.strucnaSpr
from servisna_usluga
inner join uradjeno
on servisna_usluga.sifraUsluge = uradjeno.sifraUsluge
inner join automehanicar
on uradjeno.maticniBrAutomeh = automehanicar.maticniBrAutomeh
where registarskiBroj = 405 and year(datum) = 2015;
```

Задатак 3: Приказати податке о возачима и аутобусима који сутра треба да изађу да врше превоз путника.

```
select  datumRasp, ime, prezime, vozac.kategorija, adresa,
autobus.registarskiBroj, tip, ativan from vozac
inner join autobus
on vozac.registarskiBroj = autobus.registarskiBroj
where year(datumRasp) = year(getdate()) and month(datumRasp) >=
month(getdate()) and day(datumRasp) = day(getdate())+1;
```

Ажурирање базе

Задатак 1: Пребацити аутобус 405 са линије 65Л на линију 3.

```
update autobus set sifraLinije = '3' where sifraLinije = '65L' and
registarskiBroj = 405;
```

Задатак 2: У свим подацима о сервисима које је радио Драган Јанковић заменити име аутомеханичара у Зоран Крстић.

```
update automehanicar set ime = 'Zoran', prezime = 'Krstic'
where ime = 'Dragan' and prezime = 'Jankovic';
```

Задатак 3: Обрисати податке о аутомеханичару Драгану Јанковићу.

```
delete automehanicar where ime = 'Dragan' and prezime = 'Jankovic';
```

4.4. Галерија слика

1. Пројектовати EER дијаграм базе података према приложеним захтевима. Пресликати добијени EER дијаграм у шему релационе базе података.

База података ГАЛЕРИЈА СЛИКА садржи следеће информације о свакој слици: ИД слике, назив, ширина и висина слике, година када је настала и мотив слике. За сваку слику се памти техника којом је нацртана. Свака техника има назив и свој ИД. Техника припада одговарајућој подтехници за коју се такође памти назив и ИД. Слика може да има више делова (мозаик слика од нпр. 4 засебна дела). За део се памти висина, ширина и техника сликања. За сваку слику се памти податак о сликару који ју је насликао. За сликара се памти његов ИД, презиме, име и поље за унос одређене напомене о самом сликару. Памте се подаци о галерији где је слика изложена. Зна се цена слике као и датуми од када до када је слика била изложена у галерији. За галерију се зна регистарски број, назив, адреса као и број телефона. Галерија има свог власника о коме се зна име, презиме и његов ИД. Када се слика прода неком купцу памти се датум продаје. За купца се зна његов ИД као и број телефона. Купац може бити правно или физичко лице. Ако је купац физичко лице о њему се памти име, презиме и број личне карте. Ако је у питању правно лице о њему се чувају следећи подаци: назив, факс и ПИБ. За сваког купца се памти место где станује. Такође се чувају подаци и о месту где се налазе галерије. Место је једнозначно одређено својим називом и ПТТ бројем. Треба предвидети могућност да се памте подаци о државама где се налазе места. За државу је довољно да се зна назив и међународна ознака.

2. На основу EER модела и одговарајућег релационог модела из претходног примера написати *SQL* наредбе за креирање базе података ГАЛЕРИЈА СЛИКА (направити одговарајући *DDL* фајл). Код свих табела навести неопходна ограничења: дефинисати примарне, стране кључеве и дефинисати *CHECK* ограничења за вредности неких атрибута ако је то потребно.

3.У *Microsoft SQL Server Management studio*-у покренути *query* са претходно урађеним *DDL* фајлом. У поменутом алату нацртати одговарајући дијаграм на основу формираних табела и упоредити добијени дијаграм са пројектованим EER дијаграмом.

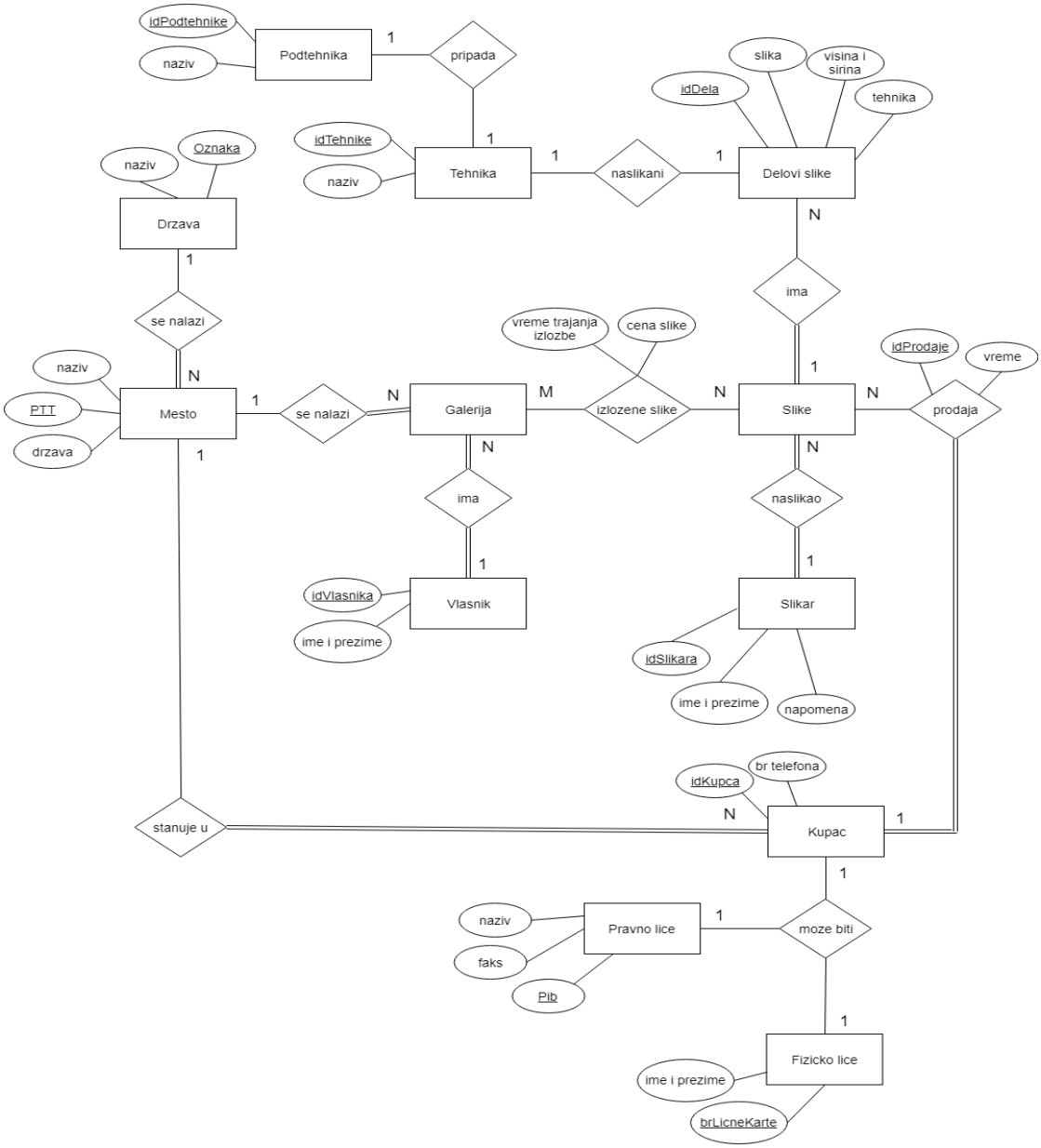
4. Претраживање базе података

Дефинисати наредбе *SQL*-а за добијање одговора на следеће упите:
1. Наћи све слике које су насликане у периоду од 1845. год. до 1905. год. које су насликане у пастелној техници.
2. Приказати све слике сликара Леонарда Да Винчија које су биле изложене у галерији Москва од 12.03.2012. год. до 15.06.2015. год.
3. Приказати податке о техникама којима су сликане слике које су биле изложене у галеријама у Новом Саду.

5. Ажурирање базе података

Дефинисати наредбе *SQL*-а за ажурирање базе података у складу са следећим захтевима:
1. Изменити податке о месту галерије Москва. Галерија Москва је била у Београду а сада је измештена у Панчеву.
2. Измените податке о власнику Галерије Москва. Нови власник је Александар Јанковић.

EER дијаграм



Слика 19. *EER дијаграм базе „Галерија слика“*

Претраживање базе

Задатак 1. Наћи све слике које су насликане у периоду од 1845. год. до 1905. год. које су насликане у пастелној техници.

```
select Slika.nazivS, Slika.godinaS, Tehnika.naziv
from Slika
inner join DeoSlike on Slika.idSlike = DeoSlike.slika
inner join Tehnika on Tehnika.IDTehnike = DeoSlike.tehnika
where godinaS > 1844 and godinaS < 1905 and
Tehnika.naziv = 'pastel';
```

Задатак 2. Приказати све слике сликара Леонарда Да Винчија које су биле изложене у галерији Москва од 12.03.2012. год. до 15.06.2015. год.

```
select Slika.nazivS, Slika.GodinaS, Slikar.Ime, Slikar.Prezime,
Galerija.naziv, Izlozene_Slike.izlozenaOd,
Izlozene_Slike.izlozenaDo
from Galerija
inner join Izlozene_Slike on Galerija.RedniBr =
Izlozene_Slike.Galerija
inner join Slika on Izlozene_Slike.Slika = Slika.idSlike
inner join Slikar on Slikar.idSlikara = Slika.idSlikara
where Izlozene_Slike.izlozenaOd > '2012/03/12' and
Izlozene_Slike.izlozenaDo < '2015/06/15' and Slikar.ime =
'Leonardo' and Slikar.prezime = 'Da Vinci';
```

Задатак 3. Приказати податке о техникама којима су сликане слике које су биле изложене у галеријама у Новом Саду.

```
select Slika.nazivS, DeoSlike.tehnika, Tehnika.naziv, Mesto.naziv
from Tehnika
inner join DeoSlike on Tehnika.idTehnike = DeoSlike.tehnika
inner join Slika on DeoSlike.slika = Slika.idSlike
inner join Izlozene_Slike on Slika.idSlike = Izlozene_Slike.Slika
inner join Galerija on Galerija.RedniBr = Izlozene_Slike.Galerija
inner join Mesto on Mesto.PTT = Galerija.PTT
where Mesto.naziv = 'Novi Sad';
```

Ажурирање базе

Задатак 1. Изменити податке о месту галерије Москва. Галерија Москва је била у Београду а сада је измештена у Панчеву.

```
update Galerija
set Mesto = (select Ptt from Mesto where Naziv = 'Pancevo')
```

```
where Mesto = (select Ptt from Mesto where Naziv = 'Beograd') and
Galerija.naziv = 'Moskva';
```

Задатак 2. Измените податке о власнику Галерије Москва. Нови власник је Александар Јанковић.

```
update Galerija
set Vlasnik = (select IdVlasnika from Vlasnik where ImeiPrezime =
'Aleksandar Jankovic')
from Vlasnik inner join Galerija
on Vlasnik.idVlasnika = Galerija.Vlasnik
where Galerija.naziv = 'Moskva';
```

4.5. Зубарска ординација

1. Пројектовати EER дијаграм базе података према приложеним захтевима. Пресликати добијени EER дијаграм у шему релационе базе података.

База података ЗУБАРСКА ОРДИНАЦИЈА треба да садржи податке о запосленима, клијентима, добављачима, наручивању материјала, заказаним и извршеним прегледима. Сваки зубни материјал доставља један добављач. За зубни материјал се памти шифра материјала и назив материјала. За добављача се памти његова шифра и назив. Када се зубни материјал наручује за њега се прави посебна наруџбеница за коју се евидентира број наруџбенице, датум, укупна цена и количина наруученог материјала. Таква наруџбеница се шаље добављачу који је регистрован у бази као добављач за дати материјал. При достављању наруученог материјала, у систему се уноси пријемница којом се ажурира стање зубног материјала. За пријемницу се памти број пријемнице и стање материјала. За сваког запосленог се памти име, презиме, матични број, година рођења, адреса становања, телефон, датум заснивања радног односа и степен стручне спреме. Запослени може бити администратор, доктор или медицински техничар. Медицински техничар отвара картон пацијенту. За пацијента се памти име, презиме, пол, јмбг, адреса и телефон. За пацијента се води евиденција о заказаном прегледу при чему се памти датум и време прегледа, доктор који преглед треба да изврши као и тражена услуга. За услугу се памти назив и цена. Када пацијент дође у ординацију, заказани преглед се изврши при чему се памти датум и време прегледа као и доктор који је преглед извршио. Уколико је по завршеном прегледу потреба

интервенција, памте се подаци о извршеној интервенцији тј. број интервенција као и подаци о зубу на коме је интервенција извршена. За зуб се памти број зуба и стање зуба. На крају се формира рачун о извршеним услугама и доставља се пацијенту.

2. На основу EER модела и одговарајућег релационог модела из претходног примера написати *SQL* наредбе за креирање базе података ЗУБАРСКА ОРДИНАЦИЈА (направити одговарајући *DDL* фајл). Код свих табела навести неопходна ограничења: дефинисати примарне, стране кључеве и дефинисати *CHECK* ограничења за вредности неких атрибута ако је то потребно.

3. У *Microsoft SQL Server Management Studio*-у покренути query са претходно урађеним *DDL* фајлом. У поменутом алату нацртати одговарајући дијаграм на основу формираних табела и упоредити добијени дијаграм са пројектованим EER дијаграмом.

4. Претраживање базе података

Дефинисати наредбе *SQL*-а за добијање одговора на следеће упите:

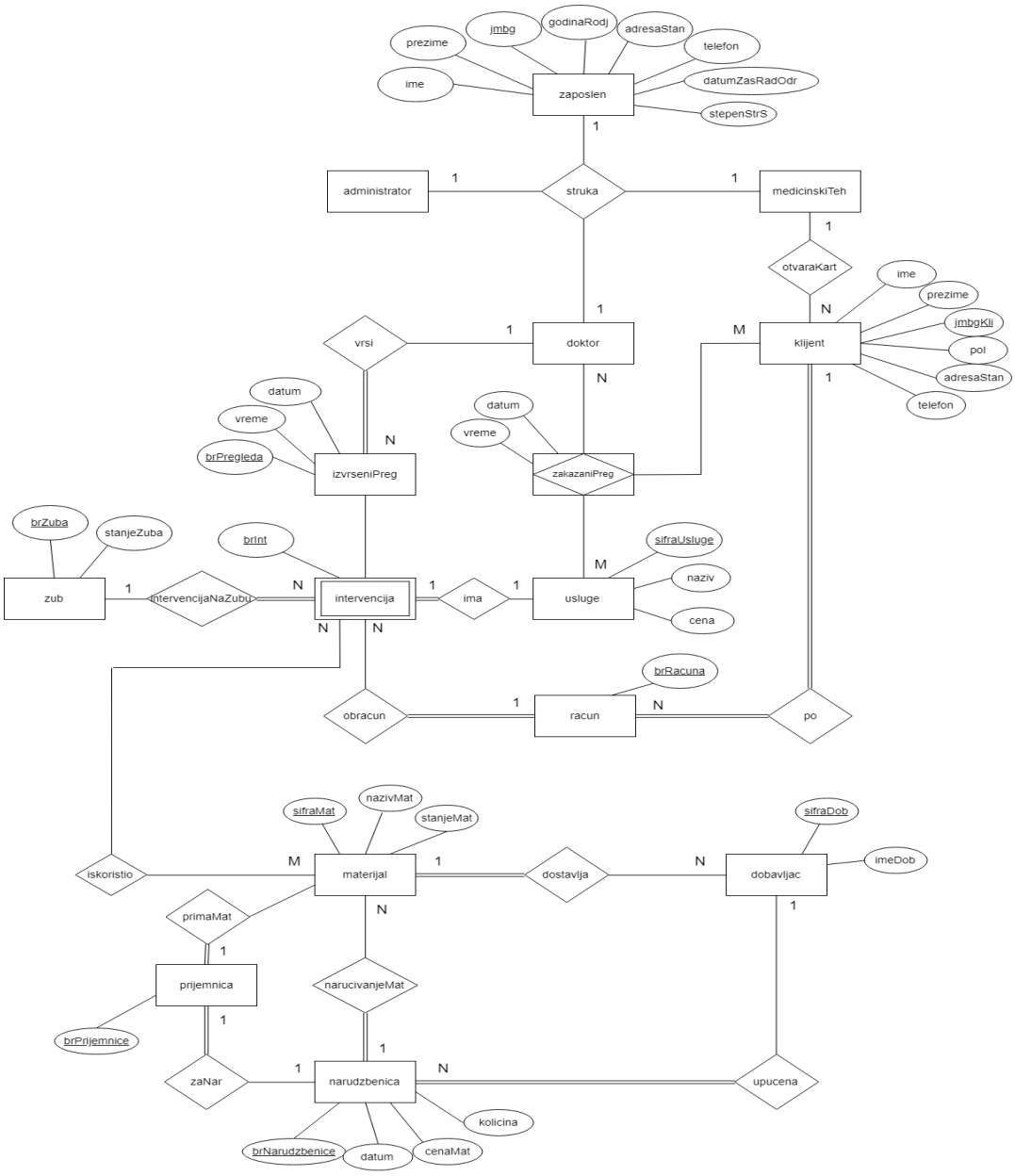
1. Наћи све материјале које испоручују добављач Galenika d.o.o а чија је вредност мања од 10.000 дин.
2. Приказати податке свих пацијената који су имали рачун већи од 5.500 дин.
3. Приказати податке пацијената који су прегледани 25.12.2017. год. од стране доктора „Јована Петровића”.

5. Ажурирање базе података

Дефинисати наредбе *SQL*-а за ажурирање базе података у складу са следећим захтевима:

1. Обрисати податке о свим клијентима чији телефонски број почиње са 069.
2. Изменити податке о добављачу *Galenika*. Уместо *Galenika* променити добављача у *Novartis*.
3. Обрисати податке о свим интервенцијама заказаним за 18.02.2018. год.

EER дијаграм



Слика 20. *EER дијаграм базе „Зубарска ординација“*

Претраживање базе

Задатак 1. Наћи све материјале које испоручује добављач Галеника а чија је вредност мања од 10.000 дин.

```
select narudzbenica.sifraDob, imeDob,cenaMat, nazivMat
```

```
from dobavljac inner join narudzbenica
on dobavljac.sifraDob = narudzbenica.sifraDob
inner join materijal on materijal.sifraMat = narudzbenica.sifraMat
where imeDob = 'Galenika' AND cenaMat < 10000;
```

Задатак 2. Приказати податке свих пацијената који су имали рачун већи од 5500 дин.

```
select ime, prezime, telefon, racun.jmbgKli, usluge.cena
from klijent inner join racun on klijent.jmbgKli = racun.jmbgKli
inner join intervencija on racun.brInt = intervencija.brInt
inner join usluge on intervencija.sifraUsluge = usluge.sifraUsluge
where usluge.cena > 5500;
```

Задатак 3. Приказати податке пацијената који су прегледани 25.12.2017. год. од стране доктора „Јована Петровића”.

```
select klijent.ime, klijent.prezime, klijent.telefon,
klijent.adresaStan, klijent.jmbgKli, CONCAT(zaposlen.ime,' ',
zaposlen.prezime) as Doktor
from zaposlen inner join doktor
on doktor.jmbg = zaposlen.jmbg inner join izvrseniPreg
on izvrseniPreg.jmbg = doktor.jmbg inner join intervencija
on intervencija.brPregleda = izvrseniPreg.brPregleda inner join
racun on racun.brInt = intervencija.brInt inner join klijent
on racun.jmbgKli = klijent.jmbgKli
where izvrseniPreg.datum = '25.12.2017. god.';
```

Ажурирање базе

Задатак 1. Обрисати податке о свим клијентима чији телефонски број почиње са 069.

```
DELETE FROM klijent WHERE telefon LIKE '069%';
```

Задатак 2. Изменити податке о добављачу Галеника. Уместо Галенике променити добављача у Новартис.

```
UPDATE dobavljac SET imeDob = 'Novartis' WHERE imeDob = 'Galenika';
```

Задатак 3. Обрисати податке о свим интервенцијама заказаним за 18.02.2018. год.

```
DELETE FROM zakazaniPreg WHERE datum = '18.02.2018. god.';
```

ЛИТЕРАТУРА

1. R. Elmasri, S. B. Navathe, *Fundamentals of Database Systems*, 2011.

2. С. Ђорђевић-Кајан, Л. Стоименов, *Практикум за вежбе на рачунару из предмета Структуре и базе података. Део 2, Базе података,* 2004.

3. М. Веиновић, Г. Шимић, *Увод у базе података,* 2010.

4. Г. Павловић-Лазетић, *Основе релационих база података,* 1999.