Sveučilište Jurja Dobrile u Puli  
Zagrebačka 30, 52100 Pula, Hrvatska  
Online studij

„Sustav upravljanja hotelom“

Dokumentacija projekta iz kolegija Baze podataka I

**Nositelj kolegija**: doc.dr.sc. Goran Oreški

**Izvođač**: Romeo Šajina, mag. inf.



Autori projekta su studenti I.godine prijediplomskog sveučilišnog studija informatike (online)

1. **Iva Batur**(JMBAG: 0269125088)
2. **Lea Beletić**
3. **Tanja Gattin-Zebić**
4. **Marta Kralj**(JMBAG: 2225112019)
5. **Marko Valečić**  
   (JMBAG: 0135255772)

U Puli, 31.svibnja 2024. godine

Sadržaj

[1. Uvod 3](#_Toc167739111)

[2. Opis poslovnog procesa 4](#_Toc167739112)

[1. Upravljanje rezervacijama 4](#_Toc167739113)

[2. Upravljanje gostima 4](#_Toc167739114)

[3. Upravljanje zaposlenicima 4](#_Toc167739115)

[4. Upravljanje financijama 4](#_Toc167739116)

[5. Upravljanje uslugama 4](#_Toc167739117)

[6. Upravljanje održavanjem 5](#_Toc167739118)

[7. Upravljanje restoranima 5](#_Toc167739119)

[8. Upravljanje skladištem 5](#_Toc167739120)

[9. Upravljanje rasporedom čišćenja 5](#_Toc167739121)

[3. Entity Relationship (ER) dijagram 6](#_Toc167739122)

[4. Veze entiteta prema ER dijagramu 7](#_Toc167739123)

[5. Sheme relacijskog modela 9](#_Toc167739124)

# Uvod

Tijekom razvoja našeg sustava upravljanja hotelom, projekt je značajno napredovao, postao je daleko opsežniji i detaljniji od prvobitne misli. Ovaj napredak je dokaz našeg rasta i boljeg razumijevanja tehnologija i tema s kojima radimo. Kako smo dublje ulazili u razvoj baze podataka, koncepti su nam postali jasniji.

Za djeljenje koda i podataka koristili smo GitHub. Za komunikaciju smo koristili WhatsApp poruke, dok smo za sastanke i razgovore koristili Zoom. ER dijagrame izrađivali smo u Lucidchartu, dok smo za pisanje shema, upita te generiranje automatskih EER dijagrama putem reverse enginering opcije koristili MySQL Workbench. Generiranje podataka ostvareno je kombinacijom programranja u Pythonu, korištenjem random data generatora, pomoći ChatGPT-a teručnim unosom i pregledavanjem podataka.

Naša baza podataka i poslovni procesi pokrivaju sve aspekte hotelskog poslovanja. Evidentiramo podatke o gostima, zaposlenicima, sobama, rezervacijama, restoranima, računima, zahtjevima održavanja, sadržajima, uslugama, plaćanjima, dobavljačima, skladištu, događajima, recenzijama, rasporedima čišćenja, smjenama radnika i radnim mjestima. Ovaj sustav omogućuje sveobuhvatno upravljanje hotelskim operacijama, od prijema gostiju i rezervacija do financijskog upravljanja i održavanja objekta.

Svi osobni podaci u sustavu su generirani nasumično, te je svaka sličnost s pravim podacima slučajna. Trudili smo se osigurati točnost podataka, kao što su opisi usluga i sadržaja, ali ne možemo jamčiti za njihovu potpunu preciznost. U sljedećim poglavljima ćemo predstaviti naš dijagram, detaljno opisati sve relacije te objasniti proces generiranja podataka.

# Opis poslovnog procesa

Naš sustav upravljanja hotelom obuhvaća sve ključne poslovne procese potrebne za učinkovito i uspješno vođenje hotela. Ti procesi uključuju upravljanje rezervacijama, gostima, zaposlenicima, financijama, uslugama, održavanjem, restoranima i skladištem. U nastavku je detaljan opis svakog od ovih poslovnih procesa:

## Upravljanje rezervacijama

Proces započinje kada gost napravi rezervaciju, odnosno kada radnik sve potrebne podatke unese u sustav. Rezervacija uključuje podatke kao što su datumi prijave i odjave, broj gostiju, tip sobe i posebne zahtjeve. Sustav automatski provjerava dostupnost soba i potvrđuje rezervaciju. Svaka potvrđena rezervacija povezuje se s odgovarajućim gostom i sobom.

## Upravljanje gostima

Podaci o gostima prikupljaju se tijekom procesa rezervacije i prijave. Ovi podaci uključuju osobne informacije, kontakte i povijest boravka. Sustav omogućava praćenje preferencija gostiju, što omogućava personaliziranu uslugu i bolju korisničku podršku. Nakon boravka, gost može ostaviti recenziju koja se također pohranjuje u sustavu.

## Upravljanje zaposlenicima

Podaci o zaposlenicima uključuju osobne informacije, radne pozicije, radne smjene i povijest zapošljavanja. Sustav omogućava vođenje evidencije o radnim smjenama, dodjeljivanje zadataka i praćenje učinkovitosti rada. Radnici su povezani s radnim mjestima, smjenama i odgovornim područjima poput održavanja ili čišćenja.

## Upravljanje financijama

Financijski procesi uključuju izdavanje računa za boravak, usluge i restoranske troškove. Računi su povezani s gostima i rezervacijama, a plaćanja se evidentiraju zajedno s načinom plaćanja. Sustav omogućava generiranje financijskih izvještaja koji pomažu u praćenju prihoda i troškova.

## Upravljanje uslugama

Hotel nudi razne dodatne usluge kao što su spa, fitness, najam vozila i organizacija izleta. Sve usluge su evidentirane u sustavu, zajedno s cijenama i opisima. Gost može rezervirati usluge tijekom boravka, a troškovi se dodaju na njegov račun.

## Upravljanje održavanjem

Sustav omogućava prijavu i praćenje zahtjeva za održavanje. Kada gost ili zaposlenik prijavi kvar ili potrebu za održavanjem, zahtjev se evidentira s opisom problema, datumom prijave i statusom. Zahtjevi se dodjeljuju odgovornim radnicima na temelju njihove radne pozicije i dostupnosti.

## Upravljanje restoranima

Hotel može imati više restorana, a svaki restoran vodi evidenciju o gostima, narudžbama i troškovima. Gost može objedovati u restoranu i troškovi se dodaju na njegov hotelski račun. Restoranski računi su povezani s gostima i centralnim financijskim sustavom hotela.

## Upravljanje skladištem

Sustav omogućava vođenje evidencije o zalihama u skladištu, uključujući informacije o dobavljačima, nabavi i stanju zaliha. Zaposlenici odgovorni za skladište evidentiraju provjere zaliha i narudžbe dobavljačima, što omogućava održavanje optimalne razine zaliha.

## Upravljanje rasporedom čišćenja

Raspored čišćenja soba i zajedničkih prostora upravlja se putem sustava. Zaposlenici zaduženi za čišćenje dobivaju dnevne zadatke, a status čišćenja soba se ažurira u stvarnom vremenu. Ovo osigurava da su sve sobe spremne za nove goste u pravom trenutku.

Ovi poslovni procesi integrirani su u jedinstveni sustav koji omogućava učinkovito upravljanje hotelom, poboljšava korisničko iskustvo i optimizira radne procese. U sljedećim poglavljima detaljno ćemo opisati naš ER dijagram, sve relacije između tablica i način generiranja podataka.

# Entity Relationship (ER) dijagram

ER dijagram našeg sustava za upravljanje hotelom prikazuje glavne entitete kao što su gosti, zaposlenici, sobe, rezervacije, računi, usluge i restorani, te njihove međusobne odnose. Korišteni su različiti tipovi odnosa, uključujući jedan-na-jedan, jedan-na-više i više-na-više. Dijagram jasno definira kako su entiteti povezani, na primjer, jedan gost može imati više rezervacija (1), dok jedna rezervacija može biti povezana s jednim računom (N:1) i jednom sobom (N:1). Ovaj dijagram omogućuje jednostavno razumijevanje i implementaciju strukture baze podataka.

# Veze entiteta prema ER dijagramu

|  |  |  |
| --- | --- | --- |
| Veze između entiteta | Kardinalnost | Opis |
| radnik - skladište | Many-to-Many | Jedan radnik provjerava više skladišta, dok jedno skladište može provjeravait više radnika. |
| skladište - dobavljač | Many-to-Many | U jedno skladište dostavlja više dobavljača, dok jedan dobavljač može dostavljati u više skladišta. |
| radnik – radno\_mjesto | One-to-Many | Jedan radnik radi na jednom radnom mjestu, na jednom radnom mjestu može raditi više radnika. |
| radnik – smjena\_radnika | Many-to-Many | Jedan radnik može raditi u više smjena, dok u jednoj smjeni može raditi više radnika. |
| radnik - raspored\_čišćenja | Many-to-Many | Jedan radnik može raditi prema više rasporeda čišćenja, također prema jednom rasporedu može čisiti više radnika. |
| radnik – rezervacija | One-to-Many | Radnik može napraviti više rezervacija, dok tu istu rezervaciju trenutno može napaviti samo jedan radnik. |
| radnik – restran | One-to-Many | U jendnom restoranu može raditi više radnika, dok jedan radnik može raditi samo u više restoranu. |
| rezrevacija - recenzija | One-to-Many | Jedna rezervacija može dobiti više recenzija, dok jedna recenzija uvijek pripada samo jednoj rezervaciji. |
| rezervacija - racun | One-to-One | Rezervacija može imat samo jedan račun, dok taj isti račun može pripadati samo toj rezervaciji. |
| rezervacija - gost | One-to-Many | Gost može napraviti više rezervacija, ali jedna rezervacija može pripadati samo jednom gostu, tj gostu koji rezervira. |
| soba – raspored\_ciscenja | Many-to-Many | Svaka soba se može čisiti prema više rasporeda, a svaki raspored u sebi može imati više soba |
| soba – zahtjev\_odrzavanja | Many-to-Many | Soba može imati više zahtjeva za održavanje, dok jedan zahjev može pripadati samo jednoj sobi |
| zahtjev\_odrzavanja - gost | One-to-Many | Jedan gost može podnijeti više zahtjeva za održavanje, dok jedan zahtjev se odnosi samo na tog gosta. |
| soba - sadrzaj | Many-to-Many | Soba može imati više sadržaja, isto tako jedan sadržaj se može dodijeliti na više soba |
| racun - usluge | One-to-Many | Jedan racun moze sadržavati više usluga, dok ista usluga može biti na više različitih računa |
| racun - placanje | One-to-Many | Svaki račun se može platiti na više načina |
| gost – restoran | Many-to-Many | Jedan restoran može korisiti više gostiju, dok jedan gost može posječivati više restorana. |

# Sheme relacijskog modela

**radnik**(id\_radnik, ime, prezime, odjel, telefon, email, datum\_zaposlenja, id\_radno\_mjesto, id\_raspored\_ciscenja)  
**skladiste**(id\_skladiste, odjel, naziv, kolicina, datum\_nabave)   
**radnik\_skladiste**(id\_radnik, id\_skladiste, datum\_provjere, napomena)   
**dobavljac**(id\_dobavljac, naziv, kontakt\_osoba, telefon, email)   
**skladiste\_dobavljac**(id\_skladiste, id\_dobavljac, datum\_dostave, napomena)   
**radno\_mjesto**(id\_radno\_mjesto, naziv, opis, odjel)   
**smjena\_radnika**(datum, vrijeme\_pocetka, vrijeme\_zavrsetka)   
**radnik\_smjena\_radnika**(id\_radnik, id\_smjena, vrijeme\_pocetka, vrijeme\_zavrsetka, pozicija)  
**raspored\_ciscenja**(datum, status)   
**soba**(id\_soba, broj\_sobe, tip, opis, cijena\_nocenja, id\_raspored\_ciscenja)   
**zahtjev\_odrzavanja**(id\_zahtjev, opis, datum\_zahtjeva, status, id\_soba, id\_gost)   
**sadrzaj**(id\_sadrzaj, naziv, opis)   
**soba\_sadrzaj**(id\_soba, id\_sadrzaj, kolicina)   
**rezervacija**(id\_rezervacija, datum\_prijave, datum\_odjave, broj\_gostiju, id\_gost, id\_radnik\_id\_racun)  
**recenzija**(id\_recenzija, datum, ocjena, komentar, id\_gost, id\_rezervacija)  
**gost**(id\_gost, ime, prezime, datum\_rodenja, adresa, telefon, email)   
**racun**(id\_racun, datum, iznos, nacin\_placanja)   
**usluge**(id\_usluga, naziv, opis, cijena)   
**racun\_usluge**(id\_racun, id\_usluga, kolicina)   
**vrsta\_placanja**(id\_placanje, nacin\_placanja, id\_racun)   
**restoran**(id\_restoran, ime, tip, lokacija, radno\_vrijeme)  
**racun\_restoran**(id\_ racun\_restoran, datum, cijena, id\_restoran)   
**gost\_restoran**(id\_gost, id\_restoran, status)

# EER dijagram (MySQL Workbench)

# Tablice

1. TABLICA radnik

Tablica radnik sadrži evidenciju svih zaposlenih radnika u hotelu i odjela na kojima rade. Sadrži atribute redom: id\_radnik, ime, prezime, odjel, telefon, email, datum\_zaposlenja, id\_radno\_mjesto i id\_raspored\_ciscenja. Atribut id\_radnik je tipa INTEGER iz razloga što se u njega unosi brojčana vrijednost te je postavljen kao PRIMARY KEY kako bi jednistveno označili svakog radnika.Atribut id\_radnik služi za identifikaciju svakog radnika u hotelu. Atribut ime je tipa string- VARCHAR jer će se unositi slova odnosno riječ s maksimalnom duljinom od 30 znakova te je na ovaj atribut postavljeno ograničenje NOT NULL kako bi osigurali unos tog podatka. Atribut prezime je također tipa VARCHAR, prima 30 znakova i postavljeno je ograničenje NOT NULL. Atributi ime I prezime služe za unos imena I prezimena radnika. Atribut odjel je također tipa VARCHAR, prima 40 znakova- malo više od imena i prezimena te je također postavljeno ograničenje NOT NULL. Atribut odjel sadrži naziv odjela na kojem radi pojedini radnik. Atribut telefon je tipa VARCHAR, a ne INTEGER zbog mogućnosti unosa nule, a atribut email je tipa VARCHAR zbog mogućnosti unosa znakova. Atributi telefon i email sadrže kontakt svakog radnika, a na njih je postavljeno I ograničenje UNIQUE kako bi osigurali jedinstvenu kombinaciju znakova bez ponavljanja. Nadalje, atribut datum\_zaposlenja je tipa DATE te je obavezan pri unosu pa je tako i na njega postavljeno ograničenje NOT NULL, a sadrži podatak kojeg datuma je zaposlen pojedini radnik. Atributi id\_radno\_mjesto i id\_raspored\_ciscenja su tipa INTEGER te su postavljeni za strane ključeve FOREIGN KEY koji se referenciraju iz tablica radno\_mjesto i tablica\_ciscenja. Zadnja dva atributa nam služe kako bismo povezali različite tablice I povukli podatke koji nam trebaju iz njih. Ova tablica je ispunjena podatcima.

U SQL-u to izgleda ovako:

CREATE TABLE radnik (

id\_radnik INTEGER PRIMARY KEY,

ime VARCHAR(30) NOT NULL,

prezime VARCHAR(30) NOT NULL,

odjel VARCHAR(40) NOT NULL,

telefon VARCHAR(20) UNIQUE,

email VARCHAR(50) UNIQUE,

datum\_zaposlenja DATE NOT NULL,

id\_radno\_mjesto INTEGER,

id\_raspored\_ciscenja INTEGER,

FOREIGN KEY (id\_radno\_mjesto) REFERENCES radno\_mjesto(id\_radno\_mjesto),

FOREIGN KEY (id\_raspored\_ciscenja) REFERENCES raspored\_ciscenja(id\_raspored\_ciscenja)

);

1. TABLICA skladiste

Tablica skladiste sadrži evidenciju potrošne robe potrebne hotelu. Sadrži atribute redom: id\_skladiste, odjel, naziv, kolicina i datum\_nabave. Atribut id\_skladiste je tipa INTEGER jer prima brojčanu vrijednost i postavljen je za PRIMARY KEY kako bi imali jedinstvenu oznaku svakog skladišta. Služi za jedinstvenu identifikaciju. Atribut odjel je tipa VARCHAR jer prima znakovnu vrijednost i ne možemo unaprijed znati koliko znakova će sadržavati, no možemo postaviti maksimalan mogući broj znakova što smo u ovom slučaju postavili na 50. Sadrži naziv odjela za koji se koristi potrošna roba. S druge strane atribut naziv je također tipa VARCHAR, no tu smo ograničili znakove na 100, a sadrži naziv pojedine robe. Na oba atributa je postavljeno ograničenje NOT NULL kako bi zagarantirali unos tih podataka. Atribut kolicina je tipa INTEGER jer prima brojčanu vrijednost te je također postavljeno ograničenje NOT NULL kao i na zadnjem atributu ove tablice- datum\_nabave, no ovaj atribut je tipa DATE jer se radi o datumu. Atribut kolicina služi za praćenje stanja robe, a datum\_nabave za praćenje kada se što naručilo. Ova tablica je ispunjena podatcima.

U SQL-u to izgleda ovako:

CREATE TABLE skladiste (

id\_skladiste INTEGER PRIMARY KEY,

odjel VARCHAR(50) NOT NULL,

naziv VARCHAR(100) NOT NULL,

kolicina INTEGER NOT NULL,

datum\_nabave DATE NOT NULL

);

1. TABLICA radnik\_skladiste

Tablica radnik\_skladiste je tablica napravljena kao nusprodukt kardinalnosti „više na više“ između tablica radnik i skladiste. Sadrži sljedeće atribute redom: id\_radnik, id\_skladiste, datum\_provjere i napomena. Atributi id\_radnik i id\_skladiste su tipa INTEGER jer primaju brojčanu vrijednost, označeni su kao FOREING KEY koji referenciraju na tablice radnik i skladiste te su definirani kao atributi od kojeg se sastoji složeni primarni ključ kako bi obje kolone bile jedinstvene. Atribut datum\_provjere je tipa DATE te je postavljeno ograničenje NOT NULL kako bi osigurali unos podataka u taj atribut, a služi za unos datuma provjere stanja skladišta. Na kraju, atribut napomena je tipa TEXT kako napomena nebi imala ograničenja u znakovima. U ovaj atribut se unose eventualne opaske ili potvrda provjere skladišta. Ova tablica je ispunjena podatcima.

U SQL-u to izgleda ovako:

CREATE TABLE radnik\_skladiste (

id\_radnik INTEGER,

id\_skladiste INTEGER,

datum\_provjere DATE NOT NULL,

napomena TEXT,

PRIMARY KEY (id\_radnik, id\_skladiste),

FOREIGN KEY (id\_radnik) REFERENCES radnik(id\_radnik),

FOREIGN KEY (id\_skladiste) REFERENCES skladiste(id\_skladiste)

);

1. TABLICA dobavljac

Tablica dobavljac sadrži evidenciju dobavljača potrošne robe hotela. Sadrži atribute redom: id\_dobavljac, naziv, kontakt\_osoba, telefon i email. Atribut id\_dobavljac je tipa INTEGER jer sadrži brojčanu vrijednost te je postavljen za PRIMARY KEY kako bi podatci u atributu bili jedinstveni te služi za jedinstvenu identifikaciju dobavljača. Atribut naziv je tipa VARCHAR jer sadrži znakovne vrijednosti te je postavljeno ograničenje na maksimalno 100 znakova te ograničenje NOT NULL kako bi zagarantirali unos podataka u taj atribut. U atribut naziv se unosi ime dobavljača. Atribut kontakt\_osoba je također tipa VARCHAR te mu je postavljena maksimalna količina znakova 100. Taj atribut nam je važan kako bi znali koga kontaktirati. Atribut telefon je tipa VARCHAR kako bi omogućili unos broja 0 te je postavljena maksimalna količina znakova na 20. Na kraju, atribut email je također VARCHAR jer sadrži znakove te je maksimalna količina znakova postavljena na 50. Ti atributi su nam važni kako bi znali kako kontaktirati dobavljača. Ova tablica je ispunjena podatcima.

U SQL-u to izgleda ovako:

CREATE TABLE dobavljac (

id\_dobavljac INTEGER PRIMARY KEY,

naziv VARCHAR(100) NOT NULL,

kontakt\_osoba VARCHAR(100),

telefon VARCHAR(20),

email VARCHAR(50)

);

1. TABLICA skladiste\_dobavljac

Tablica radnik\_skladiste je tablica napravljena kao nusprodukt kardinalnosti „više na više“ između tablica skladiste i dobavljac. Sadrži atribute redom: id\_skladiste, id\_dobavljac, datum\_dostave i napomena. Atribut id\_skladiste je tipa INTEGER te je zajedno s atributom id\_dobavljac koji je također tipa INTEGER definiran kao atribut od kojeg se sastoji složeni primarni ključ kako bi obje kolone bile jedinstvene. Također su ta dva atributa označeni kao FOREIGN KEY koji se referenciraju na tablice skladiste i dobavljac- svaki za sebe. Atribut datum\_dostave je tipa DATE jer sadrži datum pojedine dostave te je na njega postavljeno ograničenje NOT NULL kako bi zagarantirali unos tog podatka. Ovaj podatak nam je bitan kako bi mogli pratiti kada je dobavljač dostavio robu. Atribut napomena je tipa TEXT kako bi osigurali mogućnost unošenja znakova bez ograničenja ten am služi a eventualne opaske. Ova tablica je ispunjena podatcima.

U SQL-u to izgleda ovako:

CREATE TABLE skladiste\_dobavljac (

id\_skladiste INTEGER,

id\_dobavljac INTEGER,

datum\_dostave DATE NOT NULL,

napomena TEXT,

PRIMARY KEY (id\_skladiste, id\_dobavljac),

FOREIGN KEY (id\_skladiste) REFERENCES skladiste(id\_skladiste),

FOREIGN KEY (id\_dobavljac) REFERENCES dobavljac(id\_dobavljac)

);

1. TABLICA radno\_mjesto

Tablica radno mjesto sadrži podatke o postojećim pozicijama i odjelima u hotelu. Sadrži atribute redom: id\_radno\_mjesto, naziv, opis i odjel. Atribut id\_radno\_mjesto je tipa INTEGER jer sadrži brojčanu vrijednost te je postavljen za PRIMARY KEY kako bi bio jedinstven. Služi za jedinstvenu identifikaciju radnog mjesta. Atribut naziv je tipa VARCHAR kako bi osigurali mogućnst unošenja više znakova, no s maksimalnom količinom od 100 znakova. Služi za unos naziva pozicije. Na ovaj atribut je postavljeno ograničenje NOT NULL kako bi zagarantirali unos potrebnog podatka. Atribut opis je tipa TEXT kako bi se osigurala mogućnost unosa znakova bez količinskog ograničenja. Služi za detaljan opis pozicije. Na kraju, atribut odjel je tipa VARCHAR s maksimalno određenom količinom od 50 znakova te je postavljeno ograničenje NOT NULL kako bi osigurali unos potrebnih podataka. Služi za unos podatka kojem odjelu pripada pojedino radno mjesto. Ova tablica je ispunjena podatcima.

U SQL-u to izgleda ovako:

CREATE TABLE radno\_mjesto (

id\_radno\_mjesto INTEGER PRIMARY KEY,

naziv VARCHAR(100) NOT NULL,

opis TEXT,

odjel VARCHAR(50) NOT NULL

);

# Upiti

Marta Kralj

-- UPIT 1 (spajanje tablica i izvlačenje potrebnih podataka)

Na odjel recepcija je stigao naručeni paket s krivim tonerima za pisač. Voditelj recepcije se pita tko je provjeravao stanje i pritom naručio tonere za pisač kako bi saznao je li radnik krivo naručio ili je dobavljač krivo poslao. Na kraju ga zanima i tko je dobavljač te njihov kontakt kako bi mogli kontaktirati dobavljača i zamijeniti tonere.

To smo učinili na sljedeći način: kreirali smo pogled kontakt\_dobavljaci te u njemu generaliziranom projekcijom selektirali atribute naziv iz tablice dobavljac, email iz tablice dobavljac i naziv iz tablice skladiste koji smo preimenovali u naziv\_robe kako bi naziv atributa bio jasniji. Tablice dobavljac i skladiste smo preimenovali u d i s radi lakšeg i kraćeg pisanja koda- preimenovanja smo učinili uz operator AS. Iz tablica smo izvukli podatke samo za redove u kojima atribut naziv u tablici skladiste počinje s riječi Toneri- jer nas zanimaju samo toneri za pisač. Za to smo koristili operator LIKE te simbol postotka % kako bi označili da nam treba vrijednost koja ima jedan ili više znakova nakon riječi toneri. No kako bi dobili potrebne podatke, spojili smo tablice dobavljac i skladiste s operatorom JOIN ON na atributima id\_dobavljac iz tablice dobavljac i id\_skladiste iz tablice skladiste. Tako smo dobili informacije o dobavljacu u posebnoj tablici.

Zatim smo kreirali još jedan pogled ovog puta naziva provjera\_stanja gdje smo ponovno generaliziranom projekcijom selektirali ovoga puta atribute ime i prezime iz tablice radnik te atribut napomena iz tablice radnik\_skladiste. Tablice radnik i radnik\_skladiste smo preimenovali u r i rs pomoću operatora AS zbog lakšeg i kraćeg pisanja koda. Ovoga puta smo postavili uvjet da atribut napomena iz tablice radnik\_skladiste sadrzi riječ toner- to smo učinili pomoću operatora LIKE i simbola postotka % ispred i nakon riječi toner. Za kraj smo spojili tablice radnik i radnik\_skladiste pomoću operatora JOIN ON preko atributa id\_radnik iz tablice radnik i atributa id\_radnik iz tablice radnik\_skladiste. Na ovaj način smo dobili ime i prezime radnika koji je provjerio stanje tonera ( te ih nakon toga naručio jer tko provjerava stanje, naručuje) u posebnoj tablici.

Na kraju smo spojili poglede provjera\_stanja i kontakt\_dobavljaci pomoću operatora CROSS JOIN. Projekcijom smo selektirali ime i prezime radnika iz pogleda provjera\_stanja te zbrojili ta dva stringa pomoću operatora CONCAT, selektirali smo i atribut naziv\_dobavljaca, email i naziv\_robe(preimenovani atribut iz pogleda kontakt\_dobavljaci). Preimenovali smo poglede u kd i ps zbog lakšeg i kraćeg pisanja koda te smo za ime atributa zbrojenih stringova stavili narucitelj, a atribut naziv iz tablice kd u naziv\_dobavljaca radi lakšeg čitanja podataka iz tablice. Sva preimenovanja su učinjena operatorom AS. Tako smo dobili podatke tko je naručio tonere za pisač, tko je dobavljač i kontakt dobavljača.

U SQL-u to izgleda ovako:

CREATE VIEW kontakt\_dobavljaci AS

SELECT d.naziv, d.email, s.naziv AS naziv\_robe

FROM dobavljac AS d

JOIN skladiste AS s ON d.id\_dobavljac = s.id\_skladiste

WHERE s.naziv LIKE 'Toneri%';

CREATE VIEW provjera\_stanja AS

SELECT r.ime, r.prezime, rs.napomena

FROM radnik AS r

JOIN radnik\_skladiste AS rs ON r.id\_radnik = rs.id\_radnik

WHERE rs.napomena LIKE '%toner%';

SELECT CONCAT(ps.ime, " ", ps.prezime) AS narucitelj, kd.naziv AS naziv\_dobavljaca, kd.email, kd.naziv\_robe

FROM provjera\_stanja AS ps

CROSS JOIN kontakt\_dobavljaci AS kd;

-- KRAJ UPITA 1

-- UPIT 2 (brisanje radnika nakon otkaza)

Radnik koji je pogriješio u narudžbi tonera je primio par ružnih riječi od voditelja zbog čega se uvrijedio i dao otkaz. Sada je potrebno sakriti tog radnika iz tablice radnika, no bez brisanja iz baze podataka kako nebi izgubili podatke o radu tog radnika.

To smo učinili na sljedeći način: kreirali smo pogled preostali\_radnici koji prikazuje sve podatke iz tablice radnik osim radnika Lidija Perić. Selektirali smo sve podatke iz tablice radnik pomoću SELECT naredbe osim reda koji u atributu ime sadrži Lidija i atributu prezime sadrži Perić pomoću NOT LIKE operatora. Kako bi označili i atribut ime i atribut prezime smo koristili operator AND. Na ovaj način se radnik Lidija Perić ne pojavljuje u tablici preostali\_radnici, no i dalje možemo pristupiti podatcima Lidije Perić iz preostalih tablica. Uz atribute ime i prezime izdvojili smo i atribut id zbog mogućnosti istog imena i prezimena radnika koji su i dalje radnici hotela. Atribut id\_radnik smo saznali pomoću generalizirane projekcije atributa id\_radnik koji u atributu ime sadrži podatak Lidija, a u atributu prezime Perić te selektirali taj podatak.

U SQL-u to izgleda ovako:

CREATE VIEW preostali\_radnici AS

SELECT \*

FROM radnik

WHERE ime NOT LIKE 'Lidija' AND prezime NOT LIKE 'Perić' AND id=28

SELECT id\_radnik

FROM radnik

WHERE ime LIKE 'Lidija' AND prezime LIKE 'Perić';

-- KRAJ UPITA 2

--UPIT 3 ( kreiranje računa za gosta)

Potrebno je kreirati tablicu u kojoj su podatci za račun gosta Ana Anković.

To smo učinili na sljedeći način: kreirali smo pogled racun\_Ana\_Anković u kojem se nalaze ime i prezime platitelja, broj sobe u kojoj je gost odsjeo, cijena noćenja, broj dana koliko je odsjeo u sobi i ukupna cijena. Projekcijom smo selektirali atribute ime i prezime iz tablice gost (te ih spojili operatorom CONCAT u novi atribut platitelj), atribute broj\_sobe i cijena\_nocenja iz tablice soba te smo kreirali novi atribut ukupno koji sadrži podatak ukupne cijene. Atribut ukupno smo dobili množenjem cijena\_nocenja i rezultatom razlike datum\_odjave i datum\_prijave. Razliku smo dobili korištenjem operatora DATEDIFF. Kako bi koristili podatke iz različitih tablica spojili smo tablicu rezervacija s tablicom gost pomoću operatora JOIN ON preko atributa id\_rezervacija iz tablice rezervacija I atributa id\_gost iz tablice gost. Na to smo spojili I tablicu soba pomoću operatora CROSS JOIN, a za uvjet smo postavili da atribut ime u tablici gost mora biti Ana, a atribut prezime iz tablice gost mora biti Anković te smo još za uvjet postavili da atribut id\_soba mora biti 3 jer je to id sobe u kojoj je Ana Anković odsjela. Za uvjet smo koristili operator LIKE, a za postavljanje vuše uvjeta operator AND. Tako smo dobili tablicu u kojoj je nalaze samo podatci za račun Ane Anković.

U SQL-u to izgleda ovako:

CREATE VIEW racun\_Ana\_Anković AS

SELECT CONCAT(g.ime, " ", g.prezime) AS platitelj, s.broj\_sobe, s.cijena\_nocenja, DATEDIFF(rez.datum\_odjave, rez.datum\_prijave) AS broj\_dana, s.cijena\_nocenja\*DATEDIFF(rez.datum\_odjave, rez.datum\_prijave) AS ukupno

FROM rezervacija AS rez

JOIN gost AS g ON rez.id\_rezervacija=g.id\_gost

CROSS JOIN soba AS s

WHERE g.ime LIKE 'Ana' AND g.prezime LIKE 'Anković' AND s.id\_soba=3;

--KRAJ UPITA 3