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**
   (JMBAG: 0285009061)
3. **Tanja Gattin-Zebić**
   (JMBAG: 0023003769)
4. **Marta Kralj**
   (JMBAG: 2225112019
5. **Marko Valečić**
   (JMBAG: 0135255772)

U Puli, 31.svibnja 2024. godine

Sadržaj

1. 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 korištenjem random data generatora, pomoći ChatGPT-a te ručnim 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.

2. 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:

1. **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 i posebne zahtjeve. Sustav automatski provjerava dostupnost soba i potvrđuje rezervaciju. Svaka potvrđena rezervacija povezuje se s odgovarajućim gostom i sobom.

2. **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.

3. **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.

4. **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.

5. **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.

6. 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.

7. 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.

8. 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.

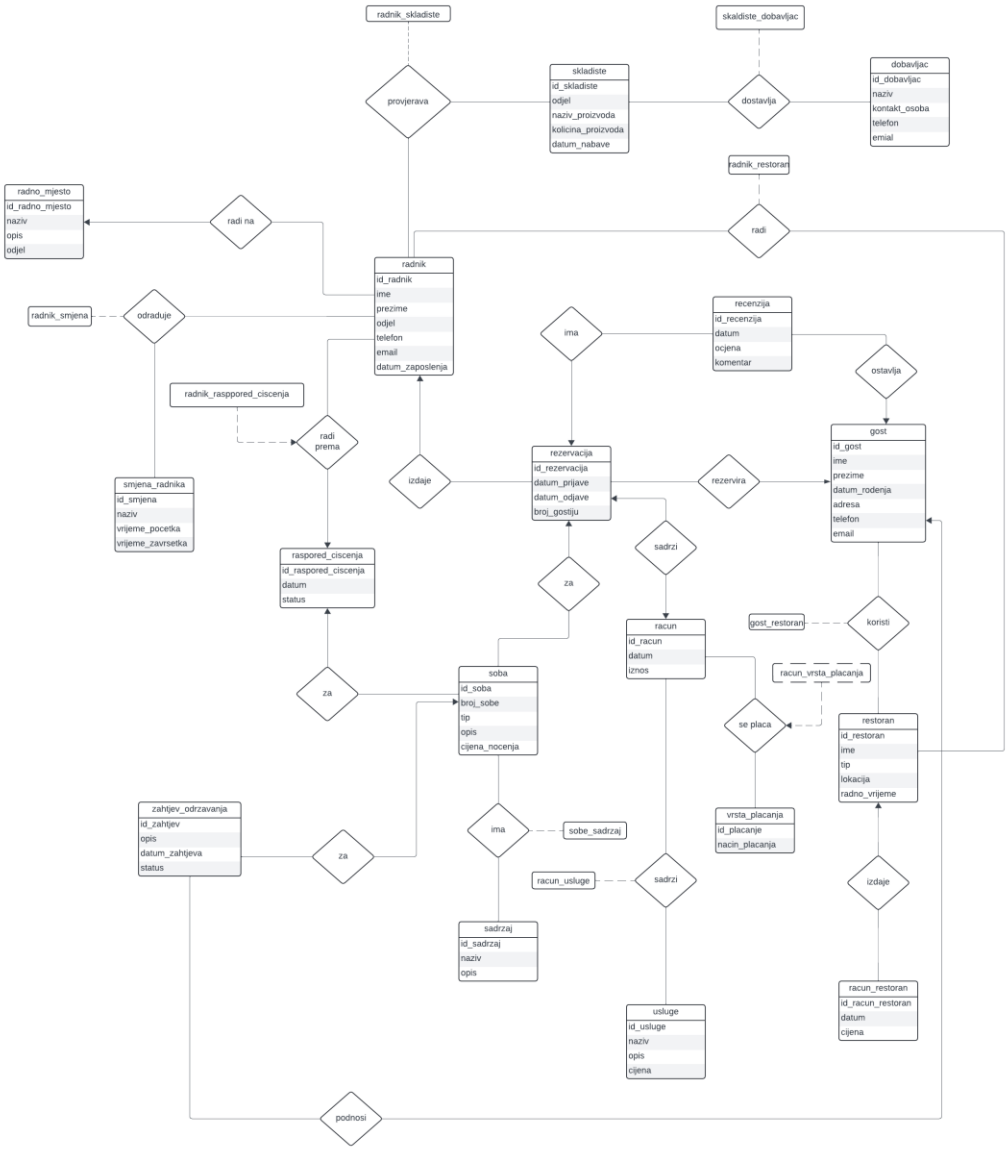
9. Upravljanje rasporedom čišćenja

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.

3. 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.



4. Veze entiteta prema ER dijagramu

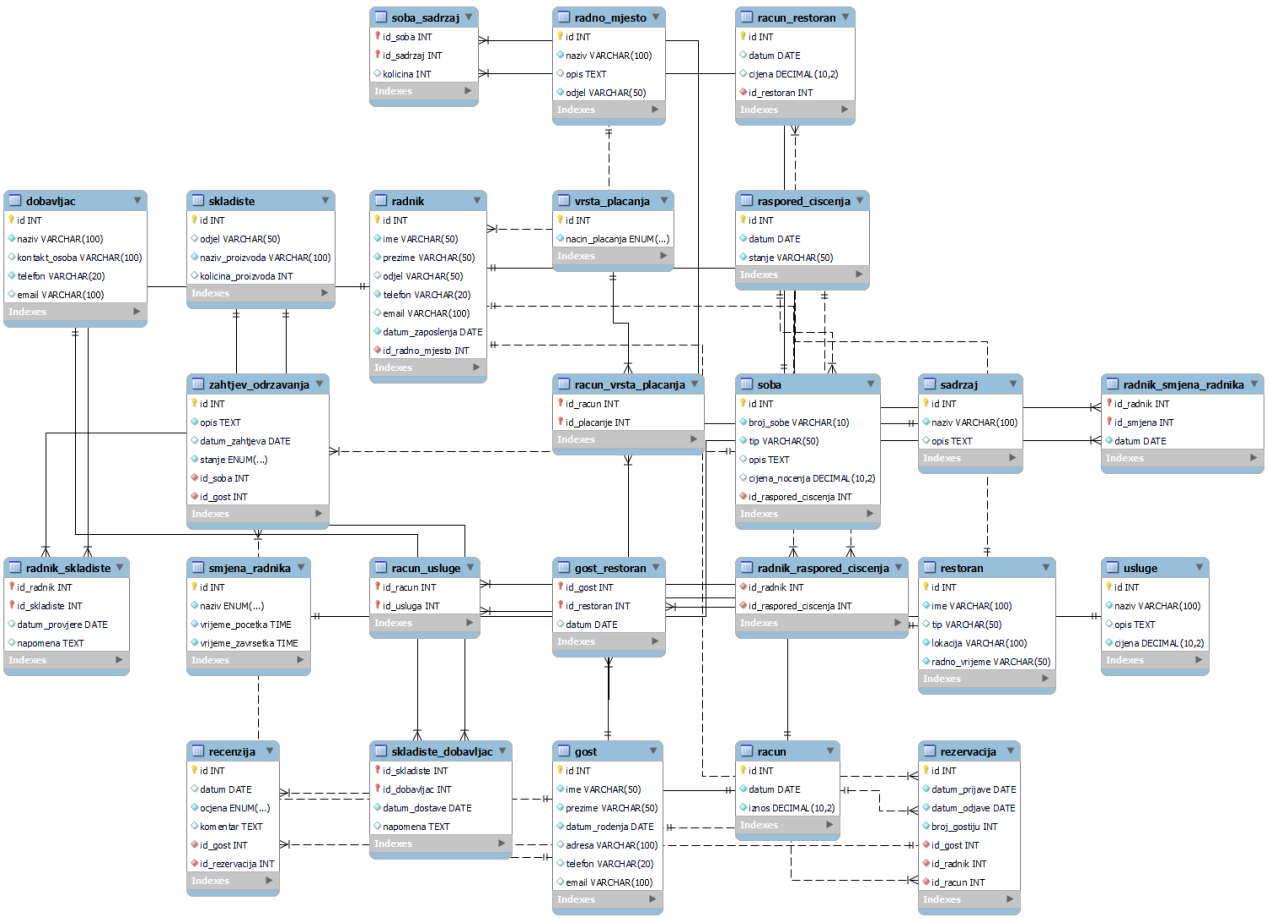
Veze između entiteta	Kardinalnost	Opis
radnik - skladište	Many-to-Many	Jedan radnik provjerava više skladišta, dok jedno skladište može provjeravait više radnika.
skladište - dobavljač	Many-to-Many	U jedno skladište dostavlja više dobavljača, dok jedan dobavljač može dostavljati u više skladišta.
radnik – 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 – restoran	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.

5. Sheme relacijskog modela

radnik(id_radnik, ime, prezime, odjel, telefon, email, datum_zaposlenja, id_radno_mjesto, id_raspored_ciscenja)
skladiste(id_skladiste, odjel, naziv_proizvoda, kolicina_proizvoda)
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(id_smjena, naziv, vrijeme_pocetka, vrijeme_zavrsetka)
radnik_smjena_radnika(id_radnik, id_smjena, datum)
raspored_ciscenja(id_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)
usluge(id_usluga, naziv, opis, cijena)
racun_usluge(id_racun, id_usluga, kolicina)
vrsta_placanja(id_placanje, nacin_placanja)
restoran(id_restoran, ime, tip, lokacija, radno_vrijeme)
racun_restoran(id_ racun_restoran, datum, cijena, id_restoran)
gost_restoran(id_gost, id_restoran, status)
racun_vrsta_placanja(id_racun, id_placanje)

6. EER dijagram (MySQL Workbench)



7. Sadržaj GIT repozitorija

Ovaj projekt sadrži sve potrebne datoteke za kreiranje i manipulaciju bazom podataka. Dokumentacija, ER dijagrami, EER dijagrami, SQL datoteke i dodatne upute nalaze se na GitHub repozitoriju.

Sadržaj Repozitorija:

Dokumentacija:
Detaljna dokumentacija projekta sadrži opis baze podataka, strukturu tablica, te upute za korištenje.
Dokumentacija se nalazi u formatu PDF.

ER Dijagrami:
ER (Entity-Relationship) dijagrami koji prikazuju veze između entiteta u bazi podataka.

EER Dijagrami:
EER (Enhanced Entity-Relationship) dijagrami koji pružaju dodatne detalje poput generalizacije, specijalizacije i drugih složenih odnosa.

SQL Datoteke:
Svaki student je kreirao vlastitu SQL datoteku koja sadrži samo upite.
Datoteke za kreiranje baza podataka.
Datoteke za insert podataka.
Sve SQL datoteke su pohranjene na git repozitorju.

8. SQL tablice

8.1 Tablica radnik

Tablica radnik služi za pohranu informacija o zaposlenicima hotela. Sadrži atribute: id_radnik, ime, prezime, odjel, telefon, email, datum_zaposlenja, id_radno_mjesto i id_raspored_ciscenja. Atribut id je tipa INT i koristi se za jedinstveno identificiranje svakog zaposlenika, čineći ga PRIMARY KEY atributom. Postavljen je na AUTO_INCREMENT, što osigurava automatsko povećanje vrijednosti pri svakom novom unosu i time osigurava jedinstvenost svakog zapisa.

Atributi ime i prezime su tipa VARCHAR, što omogućava pohranu imena i prezimena zaposlenika u tekstualnom formatu. Atribut odjel je također tipa VARCHAR i koristi se za pohranu naziva odjela u kojem zaposlenik radi, omogućavajući fleksibilnost pri unosu različitih odjela. Atribut telefon je tipa VARCHAR, što omogućava unos različitih formata telefonskih brojeva, uključujući međunarodne brojeve.Atribut email je tipa VARCHAR i ima ograničenje UNIQUE, čime se osigurava da svaki zaposlenik ima jedinstvenu email adresu unutar baze podataka. Atribut datum_zaposlenja je tipa DATE, omogućavajući pohranu datuma kada je zaposlenik počeo raditi u hotelu. Atribut id_radno_mjesto je tipa INT i koristi se kao strani ključ koji povezuje zaposlenika s tablicom radno_mjesto, definirajući njegovu radnu poziciju. Atribut id_raspored_ciscenja je također tipa INT i koristi se kao strani ključ koji povezuje zaposlenika s tablicom raspored_ciscenja, što omogućava praćenje odgovornosti za čišćenje i održavanje.

Primarni ključ ove tablice je id, a jedinstveno ograničenje je postavljeno na atribut email. Tablica također sadrži dva strana ključa, id_radno_mjesto i id_raspored_ciscenja, čime se osigurava integritet podataka između tablica. Neki atributi su označeni kao NOT NULL, što znači da moraju biti uneseni za svaki unos.

```
CREATE TABLE radnik (
        id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
        ime VARCHAR(50) NOT NULL,
        prezime VARCHAR(50) NOT NULL,
        odjel VARCHAR(50),
        telefon VARCHAR(20),
        email VARCHAR(100) NOT NULL UNIQUE,
        datum_zaposlenja DATE NOT NULL,
        id_radno_mjesto INT NOT NULL,
        id_raspored_ciscenja INT NOT NULL,
        FOREIGN KEY (id_radno_mjesto) REFERENCES radno_mjesto(id),
```

```
    FOREIGN KEY (id_raspored_ciscenja) REFERENCES
    raspored_ciscenja(id)
);
```

8.2 Tablica skladiste

Tablica skladiste sadrži evidenciju potrošne robe za potrebne hotelu. Tablica radnik sadrži atribute: id_skladiste, odjel, naziv, kolicina i datum_nabave. Atribut "id" je tipa INT jer prima brojčanu vrijednost i koristi se za jedinstveno identificiranje svake skladišne jedinice, postavljen je za PRIMARY KEY kako bi imali jedinstvenu oznaku svakog skladišta. Dodajemo AUTO_INCREMENT, što omogućava automatsko povećanje vrijednosti pri svakom novom unosu, osiguravajući jedinstvenost svakog zapisa.

Atribut "odjel" je tipa VARCHAR jer prima znakovnu vrijednost i koristi se za označavanje odjela kojem skladišna jedinica pripada, što omogućava kategorizaciju i organizaciju skladišnih jedinica po odjelima. Ne možemo unaprijed znati koliko znakova će sadržavati, no možemo postaviti maksimalan mogući broj znakova što smo u ovom slučaju postavili na 50. S druge strane atribut "naziv_proizvoda" je također tipa VARCHAR, no tu smo ograničili znakove na 100, a sadrži naziv pojedine robe. Atribut "kolicina" je tipa INT koji bilježi količinu dostupnih artikala ili proizvoda na skladištu. Ovaj atribut omogućuje praćenje trenutne zalihe stavke. Atribut "datum_nabave" je stupac tipa DATE koji bilježi datum kada je stavka nabavljena ili dodana u skladište. Ovaj atribut pruža informacije o vremenu kada je stavka postala dostupna u skladištu.

Primarni ključ tablice je "id", osiguravajući jedinstvenost svake stavke u skladištu. Osim toga, stupac "naziv" je označen kao NOT NULL kako bi se osiguralo da svaka stavka ima definiran naziv.

```
CREATE TABLE skladiste (
    id  INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    odjel VARCHAR(50),
    naziv_proizvoda VARCHAR(100) NOT NULL,
    kolicina_proizvoda INT
    datum_nabave DATE

);
```

8.3 Tablica radnik_skladiste

Tablica “radnik_skladiste” je tablica napravljena kao nusprodukt kardinalnosti „više na više“ između tablica “radnik” i “skladiste”.

Sadrži atribute kao što su “id_radnik” i “id_skladiste”, koji predstavljaju strane ključeve koji povezuju “id” tablice zaposlenika i “id” tablice skladište.

Atribut “datum_provjere” je tipa DATE a, atribut “napomena” je tipa TEXT kako napomena nebi imala ograničenja u znakovima. U ovaj atribut se unose eventualne opaske ili potvrda provjere skladišta.

```
CREATE TABLE radnik_skladiste (
        id_radnik INT NOT NULL,
        id_skladiste INT NOT NULL,
        datum_provjere DATE,
        napomena TEXT,
        PRIMARY KEY (id_radnik, id_skladiste),
        FOREIGN KEY (id_radnik) REFERENCES radnik(id),
        FOREIGN KEY (id_skladiste) REFERENCES skladiste(id)
);
```

8.4 Tablica dobavljac

Tablica dobavljac služi za pohranu informacija o dobavljačima hotela. Svaki dobavljač ima jedinstveni identifikator koji se automatski povećava. Ostali atributi su “naziv” dobavljača koji je tipa VARCHAR i ima ogranicenje na 100 znakova, što je sasvim dovoljno za potrebe naziva. Ukoliko postoje neke nejasnoće s dobavljačima posjedujemo atribut “kontakt_osoba” odnosno ime osobe te telefon za uspostavu poziva. Kako bi sve potrebne dokumente, i po potrebi posiljke, mogli razmjenjivati, definirali smo atribut “email” te ga odredili kao VARCHAR i postavili UNIQUE za jedinstvenu vrijednost.

```
CREATE TABLE dobavljac (
        id  INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
        naziv VARCHAR(100) NOT NULL,
        kontakt_osoba VARCHAR(100),
        telefon VARCHAR(20) NOT NULL,
        email VARCHAR(50) UNIQUE,

);
```

8.5 Tablica skladiste_dobavljac

Ova tablica omogućuje povezivanje skladišta s dobavljačima, pružajući informacije o datumu dostave robe ili materijala određenog dobavljača u određeno skladište. Svaki zapis u tablici definira vezu između određenog skladišta i određenog dobavljača, čime se olakšava praćenje opskrbe robe ili materijala unutar hotela.

```
CREATE TABLE skladiste_dobavljac (
        id_skladiste INT NOT NULL,
        id_dobavljac INT NOT NULL,
        datum_dostave DATE NOT NULL,
        napomena TEXT,
        PRIMARY KEY (id_skladiste, id_dobavljac),
        FOREIGN KEY (id_skladiste) REFERENCES skladiste(id),
        FOREIGN KEY (id_dobavljac) REFERENCES dobavljac(id)

);
```

8.6 Tablica radno_mjesto

Tablica "radno mjesto" predstavlja skup informacija o različitim radnim mjestima. Svako radno mjesto ima svoj identifikator, automatski dodijeljen. Tablica sadrži atribut "naziv" tipa VARCHAR koji sadrzi informacije radnog mjesta. Atribut "opis" koji je tipa TEXT i u njega navaodimo opis posla koji radnik mora obavljati u svoje radno vrijeme. Odjel u kojem se nalazi radno mjesto omogućuje organizaciju osoblja unutar hotela prema različitim funkcionalnim područjima.

```
CREATE TABLE radno_mjesto (
        id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
        naziv VARCHAR(100) NOT NULL,
        opis TEXT,
        odjel VARCHAR(50) NOT NULL

);
```

8.7 Tablica smjena_radnika

Tablica "smjena_radnika" predstavlja skup informacija o radnim smjenama zaposlenika u hotelu. Svaka smjena ima svoj jedinstveni identifikator, automatski dodijeljen i povećan s dodavanjem novog zapisa. Ostali atributi uključuju: "naziv" tipa ENUM, što znači da se u tablicu pod naziv može upisati samo ono što je ponuđeno, "vrijeme_pocetka" tipa TIME, koje označava vrijeme početka smjene; te "vrijeme_zavrsetka" isto tipa TIME, koje označava vrijeme završetka smjene. Ova tablica omogućuje precizno praćenje rasporeda radnih smjena za osoblje hotela, olakšavajući organizaciju rada i pružajući jasne informacije o vremenima prisutnosti zaposlenika.

```
CREATE TABLE smjena_radnika (
        id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
        naziv ENUM ('Jutarnja smjena', 'Popodnevna smjena', 'Noćna smjena') NOT NULL,
        vrijeme_pocetka TIME NOT NULL,
        vrijeme_zavrsetka TIME NOT NULL

);
```

8.8 Tablica radnik_smjena_radnika

Tablica radnik_smjena_radnika predstavlja rezultat odnosa “više na više” između tablica radnik i smjena_radnika, omogućujući povezivanje zaposlenika i radnih smjena unutar hotela.

Ovaj model omogućuje praćenje povezanosti između zaposlenika i njihovih radnih smjena, što olakšava organizaciju rasporeda rada i upravljanje prisutnošću zaposlenika tijekom smjena. Tablica sadrži atribute koji identificiraju zaposlenika (id_radnik) i smjenu (id_smjena), te dodatne informacije o datumu odrađene smijene za određenog zaposlenika.

```
CREATE TABLE radnik_smjena_radnika (
        id_radnik INT NOT NULL,
        id_smjena INT NOT  NULL,
        datum DATE NOT NULL,
        PRIMARY KEY (id_radnik, id_smjena),
        FOREIGN KEY (id_radnik) REFERENCES radnik(id),
        FOREIGN KEY (id_smjena) REFERENCES smjena_radnika(id)

);
```

8.9 Tablica raspored_ciscenja

Ova tablica omogućuje praćenje rasporeda čišćenja u hotelu. Svaki raspored čišćenja ima svoj jedinstveni identifikator (“id_raspored_ciscenja”), koji se automatski dodjeljuje i povećava s dodavanjem novog zapisa. Atributi uključuju “datum” tipa DATE, kada je raspored čišćenja planiran te atribut “stanje”, tipa VARCAHR, koji opisuje trenutno stanje rasporeda čišćenja (npr. "u tijeku", "završeno", "odgođeno" itd.).

Ova tablica omogućuje efikasno upravljanje rasporedom čišćenja u hotelu, pružajući jasne informacije o rasporedu i statusu čišćenja za svaki dan.

```
CREATE TABLE raspored_ciscenja (
        id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
        datum DATE NOT NULL,
        status VARCHAR(50) NOT NULL

);
```

8.10 Tablica soba

Tablica soba pohranjuje informacije o sobama unutar hotela, a svaki atribut je detaljno opisan u nastavku:

Atribut "id" je jedinstveni identifikator za svkau sobu. Tip atributa je INT, a atribut se automatski povećava sa AUTO_INCREMENT kako bi se osigurala jedinstvenost svake osobe. Atribut "id" je također primarni ključ (PRIMARY KEY) tablice.

Atribut "broj_sobe" je broj sobe unutar hotela. Tip atributa je VARCHAR(10) što omogućuje pohranu brojeva soba u obliku niza znakova, obzirom da brojevi soba mogu sadržavati i slova(npr. "101A"). Ovo polje je obavezno NOT NULL

Atribut "tip_sobe" je tip sobe koji može biti na primjer, jednokrevetna, dvokrevetna, apartman ili slično. Tip atributa je je VARCHAR(50). Ovo polje je također obavezno, odnosno NOT NULL.

"opis" sobe, pruža dodatne informacije o značajkama i pogodnostima sobe. Tip ovog atributa je TEXT, koji omogućuje pohranu dužih tekstualnih opisa.

Atribut "cijena_noćenja" odnosi se na cijenu noćenja u sobi. Tip atributa smo odredili da bude DECIMAL(10, 2), što bi značilo da u tablicu možemo spremiti brojčanu vrijednost s dvije decimalne točke. Ovo polje mora biti obavezn popunjeno, NOT NULL.

Na kraju u tablici imamo atribut "id_raspored_ciscenja" koji nam je strani kljuc. Tip atributa je INT, a atribut je povezan s atributom "id_raspored_ciscenja" iz tablice "raspored_ciscenja" putem FOREIGN KEY ograničenja. Ovo omogućuje da svaka soba bude povezana s određenim rasporedom čišćenja..

```
CREATE TABLE soba (
        id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
        broj_sobe VARCHAR(10) NOT NULL,
        tip VARCHAR(50) NOT NULL,
        opis TEXT,
        cijena_nocenja DECIMAL(10, 2) NOT NULL CHECK (cijena_nocenja >= 0),
        id_raspored_ciscenja INT NOT NULL,
        FOREIGN KEY (id_raspored_ciscenja) REFERENCES
        raspored_ciscenja(id)

);
```

8.11 Tablica zahtjev_odrzavanja

Tablica zahtjev_odrzavanja pohranjuje informacije o zahtjevima za održavanje u hotelu, a svaki atribut je detaljno opisan u nastavku:

Atribut "id" je jedinstveni identifikator za svaki zahtjev za održavanje. Tip atributa je INT, a atribut se automatski povećava (AUTO_INCREMENT) kako bi se osigurala jedinstvenost svakog zahtjeva. Atribut "id" je također primarni ključ (PRIMARY KEY) tablice.

Atribut "opis" detaljno opisuje problem ili radove koje treba obaviti. Tip atributa je TEXT, što omogućuje pohranu dužih tekstualnih opisa. Ovo polje je obavezno (NOT NULL).

Atribut "datum_zahtjeva" označava datum kada je zahtjev za održavanje podnesen. Tip atributa je DATE, što omogućuje pohranu datuma u formatu godine, mjeseca i dana. Ovo polje je obavezno (NOT NULL).

Atribut "stanje" pokazuje trenutni status zahtjeva, koji može biti, na primjer, "u tijeku", "završeno", "odgođeno" i slično. Tip atributa je VARCHAR(50), što omogućuje pohranu različitih statusa kao niz znakova. Ovo polje je obavezno (NOT NULL).

Atribut "id_soba" je strani ključ koji povezuje zahtjev s određenom sobom. Tip atributa je INT, a atribut je povezan s atributom "id_soba" iz tablice soba putem FOREIGN KEY ograničenja. Ovo omogućuje praćenje koji zahtjev za održavanje se odnosi na koju sobu.

Atribut "id_gost" je strani ključ koji povezuje zahtjev s određenim gostom. Tip atributa je INT, a atribut je povezan s atributom "id_gost" iz tablice "gost" putem FOREIGN KEY ograničenja. Ovo omogućuje praćenje koji gost je podnio zahtjev za održavanje.

```
CREATE TABLE zahtjev_odrzavanja (
        id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
        opis TEXT NOT NULL,
        datum_zahtjeva DATE NOT NULL,
        stanje ENUM('U tijeku', 'Završeno', 'Otkazano')NOT NULL,
        id_soba INT NOT NULL,
        id_gost INT NOT NULL,
        FOREIGN KEY (id_soba) REFERENCES soba(id),
        FOREIGN KEY (id_gost) REFERENCES gost(id)

);
```

8.12 **Tablica sadrzaj**

Tablica sadrzaj pohranjuje informacije o sadržajima unutar hotela, a svaki atribut je detaljno opisan u nastavku:

Atribut "id" je jedinstveni identifikator za svaki sadržaj. Tip atributa je INT, a atribut se automatski povećava (AUTO_INCREMENT) kako bi se osigurala jedinstvenost svakog sadržaja. Atribut "id" je također primarni ključ (PRIMARY KEY) tablice.

Atribut "naziv" predstavlja naziv sadržaja. Tip atributa je VARCHAR(100), što omogućuje pohranu naziva sadržaja kao niz znakova. Ovo polje je obavezno (NOT NULL).

Atribut "opis" pruža dodatne informacije o značajkama i pogodnostima sadržaja. Tip atributa je TEXT, što omogućuje pohranu dužih tekstualnih opisa. Ovo polje je također obavezno (NOT NULL).

```
CREATE TABLE sadrzaj (
        id_sadrzaj INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
        naziv VARCHAR(100) NOT NULL,
        opis TEXT NOT NULL

);
```

8.13 Tablica soba_sadrzaj

Tablica soba_sadrzaj povezuje informacije o sadržajima dostupnim u pojedinim sobama unutar hotela, omogućujući odnos "više na više" između tablica "soba" i "sadrzaj". Svaki atribut u tablici je detaljno opisan u nastavku:

Atribut " id_soba" je strani ključ koji povezuje ovu tablicu s tablicom "soba". Tip atributa je INT, a atribut je povezan s atributom "id_soba" iz tablice "soba" putem FOREIGN KEY ograničenja. Ovo omogućuje praćenje koji sadržaji su dostupni u kojoj sobi.

Atribut " id_sadrzaj" je strani ključ koji povezuje ovu tablicu s tablicom "sadrzaj". Tip atributa je INT, a atribut je povezan s atributom " id_sadrzaj" iz tablice "sadrzaj" putem FOREIGN KEY ograničenja. Ovo omogućuje praćenje koji sadržaji su povezani s kojom sobom.

Atribut "kolicina" označava broj jedinica određenog sadržaja prisutnog u sobi. Tip atributa je INT i ovo polje je obavezno (NOT NULL). Ovaj atribut omogućuje precizno praćenje količine svakog sadržaja u pojedinim sobama.

Primarni ključ (PRIMARY KEY) za ovu tablicu je kombinacija atributa "id_soba" i "id_sadrzaj", što osigurava jedinstvenost svakog unosa u smislu kombinacije sobe i sadržaja.

```
CREATE TABLE soba_sadrzaj (
        id_soba INT NOT NULL,
        id_sadrzaj INT NOT NULL,
        kolicina INT,
        PRIMARY KEY (id_soba, id_sadrzaj),
        FOREIGN KEY (id_soba) REFERENCES soba(id),
        FOREIGN KEY (id_sadrzaj) REFERENCES sadrzaj(id)

);
```

8.14 Tablica rezervacija

Tablica rezervacija pohranjuje informacije o rezervacijama unutar hotela, a svaki atribut je detaljno opisan u nastavku:

Atribut "id" je jedinstveni identifikator za svaku rezervaciju. Tip atributa je INT i automatski se povećava (AUTO_INCREMENT) kako bi se osigurala jednistvenost svake rezervacije. Ovaj atribut je također i primarni ključ tablice (PRIMARY KEY)

Atribut "datum_prijave" označava datum kada gost dolazi u hotel. Tip atributa je DATE, što omogućuje pohranu datuma u formatu godine, mjeseca i dana. Ovo polje je obavezno (NOT NULL).

Atribut "datum_odjave" označava datum kada gost odlazi iz hotela. Tip atributa je također DATE. Ovo polje je obavezno (NOT NULL), kako bi kasnije mogli izvrsavati upite za određeni period poslovanja.

Atribut "broj_gostiju" označava broj gostiju koji će boraviti u sobi. Tip atributa je INT i ovo polje je obavezno (NOT NULL).

Atribut "id_gost" je strani ključ koji povezuje ovu tablicu s tablicom "gost". Tip atributa je INT, a atribut je povezan s atributom "id_gost" iz tablice "gost" putem FOREIGN KEY ograničenja. Ovo omogućuje praćenje koji gost je napravio rezervaciju.

Atribut "id_radnik" je strani ključ koji povezuje ovu tablicu s tablicom radnik. Tip atributa je INT, a atribut je povezan s atributom "id_radnik" iz tablice radnik putem FOREIGN KEY ograničenja. Ovo omogućuje praćenje koji radnik je obradio rezervaciju.

Atribut "id_racun" je strani ključ koji povezuje ovu tablicu s tablicom "racun". Tip atributa je INT, a atribut je povezan s atributom "id_racun" iz tablice racun putem FOREIGN KEY ograničenja. Ovo omogućuje praćenje koji račun je povezan s rezervacijom.

```
CREATE TABLE rezervacija (
        id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
        datum_prijave DATE NOT NULL,
        datum_odjave DATE NOT NULL,
        broj_gostiju INT NOT NULL,
        id_gost INT NOT NULL,
        id_radnik INT NOT NULL,
        id_racun INT NOT NULL,
        FOREIGN KEY (id_gost) REFERENCES gost(id),
        FOREIGN KEY (id_radnik) REFERENCES radnik(id),
        FOREIGN KEY (id_racun) REFERENCES racun(id)

);
```

8.15 Tablica recenzija

Tablica recenzija pohranjuje informacije o recenzijama gostiju, a svaki atribut je detaljno opisan u nastavku:

Atribut "id" je jedinstveni identifikator za svaku recenziju. Tip atributa je INT, a atribut se automatski povećava (AUTO_INCREMENT) kako bi se osigurala jedinstvenost svake recenzije. Atribut "id" je također primarni ključ (PRIMARY KEY) tablice.

Atribut "datum" označava datum kada je recenzija napisana. Tip atributa je DATE, što omogućuje pohranu datuma u formatu godine, mjeseca i dana. Ovo polje je obavezno (NOT NULL).

Atribut "ocjena" predstavlja ocjenu koju je gost dao, obično na skali od 1 do 5 ili 1 do 10. Tip atributa je INT i ovo polje je obavezno (NOT NULL).

Atribut "komentar" sadrži tekstualni komentar koji je gost ostavio uz svoju ocjenu. Tip atributa je TEXT, što omogućuje pohranu dužih tekstualnih opisa.

Atribut "id_gost" je strani ključ koji povezuje ovu tablicu s tablicom gost. Tip atributa je INT, a atribut je povezan s atributom "id_gost" iz tablice gost putem FOREIGN KEY ograničenja. Ovo omogućuje praćenje koja recenzija pripada kojem gostu.

Atribut "id_rezervacija" je strani ključ koji povezuje ovu tablicu s tablicom rezervacija. Tip atributa je INT, a atribut je povezan s atributom "id_rezervacija" iz tablice rezervacija putem FOREIGN KEY ograničenja. Ovo omogućuje praćenje koja recenzija je povezana s kojom rezervacijom.

```
CREATE TABLE recenzija (
        id INT NOT NULLL AUTO_INCREMENT PRIMARY KEY,
        datum DATE NOT NULL,
        ocjena ENUM ('1', '2', '3', '4', '5') NOT NULL,
        komentar TEXT,
        id_gost INT NOT NULL,
        id_rezervacija INT NOT NULL,
        FOREIGN KEY (id_gost) REFERENCES gost(id),
        FOREIGN KEY (id_rezervacija) REFERENCES rezervacija(id)

);
```

8.16 Tablica gost

Tablica gost pohranjuje osnovne podatke o gostima hotela. Sadrži sljedeće atribute:

Atribut "id" je jedinstveni identifikator za svakog gosta. Ovaj atribut je tipa INT i automatski se povećava (AUTO_INCREMENT) kako bi osigurao jedinstvenost svakog gosta. Također, služi kao primarni ključ (PRIMARY KEY) tablice.

Atribut "ime" gosta. Ovaj atribut je tipa VARCHAR s maksimalnom duljinom od 50 znakova. Polje je obavezno (NOT NULL).

Atribut "prezime" gosta. Također tipa VARCHAR s maksimalnom duljinom od 50 znakova i obavezno (NOT NULL).

Atribut "datum" rođenja gosta. Ovaj atribut je tipa DATE i pohranjuje datum u formatu godine, mjeseca i dana. Polje je obavezno (NOT NULL).

Atribut "adresa" gosta. Tip atributa je VARCHAR s maksimalnom duljinom od 100 znakova. Ovaj atribut nema obavezu unosa podataka adrese, jer nam nije neophodan za registraciju korisnika.

Atriubt "kontakt_telefon" gosta. Tip atributa je VARCHAR s maksimalnom duljinom od 20 znakova. Ovo polje nije obavezno, omogućujući da bude NULL ako broj telefona nije dostupan. Ali smo postavili ogranicenje UNIQUE, jer ne postoji opcija da dva korisnika imaju isit broj telefona.

Atribut "email" gosta. Tip atributa je VARCHAR s maksimalnom duljinom od 50 znakova i ima jedinstveno ograničenje (UNIQUE) kako bi se spriječilo dupliranje email adresa.

```
CREATE TABLE gost (
        id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
        ime VARCHAR(50) NOT NULL,
        prezime VARCHAR(50) NOT NULL,
        datum_rodenja DATE NOT NULL,
        adresa VARCHAR(100) NOT NULL,
        telefon VARCHAR(20) UNIQUE,
        email VARCHAR(50)  UNIQUE

);
```

8.17 Tablica racun

Tablica racun pohranjuje osnovne podatke o računima izdanima gostima hotela. Svaki atribut je detaljno opisan u nastavku:

Jedinstveni identifikator "id" za svaki račun. Ovaj atribut je tipa INT i automatski se povećava (AUTO_INCREMENT) kako bi se osigurala jedinstvenost svakog računa. Također služi kao primarni ključ (PRIMARY KEY) tablice.

Atribut "datum" izdavanja računa. Tip atributa je DATE, što omogućuje pohranu datuma u formatu godine, mjeseca i dana. Ovo polje je obavezno (NOT NULL).

Atribut "iznos" računa. Tip atributa je DECIMAL(10, 2), koji omogućuje pohranu brojčane vrijednosti s dvije decimalne točke, što je prikladno za financijske podatke. Ovo polje je obavezno (NOT NULL).

Ovi atributi osiguravaju da tablica racun pohranjuje ključne informacije o financijskim transakcijama, omogućujući učinkovito praćenje i upravljanje prihodima hotela.

```
CREATE TABLE racun (
        id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
        datum DATE NOT NULL,
        iznos DECIMAL(10, 2) NOT NULL


);
```

8.18 Tablica usluge

Tablica usluge pohranjuje podatke o različitim uslugama koje hotel nudi gostima. Svaki atribut je detaljno opisan u nastavku:

"id" - nedinstveni identifikator za svaku uslugu. Ovaj atribut je tipa INT i automatski se povećava (AUTO_INCREMENT) kako bi se osigurala jedinstvenost svake usluge. Također služi kao primarni ključ (PRIMARY KEY) tablice.

"naziv" - naziv usluge. Ovaj atribut je tipa VARCHAR s maksimalnom duljinom od 100 znakova. Polje je obavezno (NOT NULL), što znači da svaki unos mora imati naziv.

“opis” - opis usluge. Tip atributa je TEXT, koji omogućuje pohranu dužih tekstualnih opisa. Ovo polje nije obavezno, što znači da može ostati prazno ako nije potreban dodatni opis.

“cijena” - cijena usluge. Tip atributa je DECIMAL(10, 2), koji omogućuje pohranu brojčane vrijednosti s dvije decimalne točke. Ovo polje je obavezno (NOT NULL), što znači da svaki unos mora imati specificiranu cijenu.

Ovi atributi omogućuju da tablica usluge pohranjuje sve potrebne informacije o dostupnim uslugama, što pomaže u upravljanju ponudom usluga i cijenama unutar hotela.

```
CREATE TABLE usluge (
        id_usluga INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
        naziv VARCHAR(100) NOT NULL,
        opis TEXT,
        cijena DECIMAL(10, 2) DEFAULT 0.00 NOT NULL
);
```

8.19 Tablica racun_usluge

Tablica “racun_usluge” služi kao poveznica između tablica racun i usluge, omogućujući praćenje usluga uključenih u svaki račun. Svaki atribut je detaljno opisan u nastavku:

“id_racun” Identifikator računa. Ovaj atribut je tipa INT i služi kao vanjski ključ (FOREIGN KEY) koji se povezuje s atributom “id_racun” u tablici “racun”. Ovaj atribut je također dio primarnog ključa (PRIMARY KEY) tablice racun_usluge.

“id_usluga” Identifikator usluge. Ovaj atribut je tipa INT i služi kao vanjski ključ (FOREIGN KEY) koji se povezuje s atributom “id_usluga” u tablici usluge. Ovaj atribut je također dio primarnog ključa (PRIMARY KEY) tablice “racun_usluge”.

Ovi atributi omogućuju da tablica racun_usluge prati koje su usluge uključene u svaki račun i u kojoj količini, što pomaže u detaljnom praćenju usluga koje hotel pruža svojim gostima i njihovih troškova.

```
CREATE TABLE racun_usluge (
        id_racun INT,
        id_usluga INT,
```

```
    PRIMARY KEY (id_racun, id_usluga),
    FOREIGN KEY (id_racun) REFERENCES racun(id_racun),
    FOREIGN KEY (id_usluga) REFERENCES usluge(id_usluga)
);
```

8.20 Tablica vrsta placanja

Tablica "vrsta_placanja" pohranjuje informacije o načinima plaćanja koji su korišteni u hotelu. Svaki atribut je detaljno opisan u nastavku:

Atribut "id_placanje": Jedinstveni identifikator za svaki način plaćanja. Ovaj atribut je tipa INT i automatski se povećava (AUTO_INCREMENT) kako bi se osigurala jedinstvenost svakog unosa. Također služi kao primarni ključ (PRIMARY KEY) tablice.

"nacin_placanja": Opisuje način plaćanja koji je korišten, kao što su kreditna kartica, gotovina, bankovni prijenos i slično. Ovaj atribut je tipa VARCHAR s maksimalnom duljinom od 50 znakova. Polje je obavezno (NOT NULL), što znači da svaki unos mora imati definiran način plaćanja.

"id_racun": Identifikator računa s kojim je povezan način plaćanja. Ovaj atribut je tipa INT i služi kao vanjski ključ (FOREIGN KEY) koji se povezuje s atributom "id_racun" u tablici "racun". Ovo omogućuje povezivanje svake vrste plaćanja s određenim računom.

Ovi atributi omogućuju da tablica "vrsta_placanja" pohranjuje sve relevantne informacije o različitim načinima plaćanja korištenim u hotelu, što pomaže u praćenju i upravljanju financijskim transakcijama.

```
CREATE TABLE vrsta_placanja (
    id_placanje INT AUTO_INCREMENT PRIMARY KEY,
    nacin_placanja VARCHAR(50) NOT NULL,
    id_racun INT,
    FOREIGN KEY (id_racun) REFERENCES racun(id_racun)

);
```

8.21 Tablica restoran

Tablica “restoran” pohranjuje podatke o različitim restoranima unutar hotela. Svaki atribut je detaljno opisan u nastavku:

“id_restoran”: Jedinstveni identifikator za svaki restoran. Ovaj atribut je tipa INT i automatski se povećava (AUTO_INCREMENT) kako bi se osigurala jedinstvenost svakog unosa. Također služi kao primarni ključ (PRIMARY KEY) tablice.

“ime”: Naziv restorana. Ovaj atribut je tipa VARCHAR s maksimalnom duljinom od 100 znakova. Polje je obavezno (NOT NULL), što znači da svaki unos mora imati definiran naziv restorana.

“tip”: Tip restorana, kao što su talijanski, kineski, roštilj i slično. Ovaj atribut je tipa VARCHAR s maksimalnom duljinom od 50 znakova. Polje je obavezno (NOT NULL), što znači da svaki unos mora imati definiran tip restorana.

“lokacija”: Lokacija restorana unutar hotela ili njegovog kompleksa. Ovaj atribut je tipa VARCHAR s maksimalnom duljinom od 100 znakova. Polje je obavezno (NOT NULL), što znači da svaki unos mora imati definiranu lokaciju.

“radno_vrijeme”: Radno vrijeme restorana. Ovaj atribut je tipa VARCHAR s maksimalnom duljinom od 50 znakova. Polje je obavezno (NOT NULL), što znači da svaki unos mora imati definirano radno vrijeme.

Ovi atributi omogućuju da tablica “restoran” pohranjuje sve relevantne informacije o restoranima unutar hotela, što pomaže u upravljanju njihovim operacijama i ponudom usluga gostima.

```
CREATE TABLE restoran (
        id_restoran INT AUTO_INCREMENT PRIMARY KEY,
        ime VARCHAR(100) NOT NULL,
        tip VARCHAR(50) NOT NULL,
        lokacija VARCHAR(100) NOT NULL,
        radno_vrijeme VARCHAR(50) NOT NULL

);
```

8.22 Tablica racun_restoran

Tablica “racun_restoran” pohranjuje podatke o računima izdanima od strane restorana unutar hotela. Svaki atribut je detaljno opisan u nastavku:

“id_racun_restoran”: Jedinstveni identifikator za svaki račun izdan od restorana. Ovaj atribut je tipa INT i automatski se povećava (AUTO_INCREMENT) kako bi se osigurala jedinstvenost svakog unosa. Također služi kao primarni ključ (PRIMARY KEY) tablice.

“datum”: Datum izdavanja računa. Ovaj atribut je tipa DATE i obavezan je (NOT NULL), što znači da svaki unos mora imati definiran datum.

“cijena”: Ukupna cijena na računu. Ovaj atribut je tipa DECIMAL s preciznošću od 10 znamenki i 2 decimalna mjesta (DECIMAL(10, 2)). Polje je obavezno (NOT NULL), što znači da svaki unos mora imati definiranu cijenu.

“id_restoran”: Identifikator restorana koji je izdao račun. Ovaj atribut je tipa INT i služi kao vanjski ključ (FOREIGN KEY) koji se povezuje s atributom “id_restoran” u tablici restoran. Ovo omogućuje povezivanje svakog računa s odgovarajućim restoranom.

Ovi atributi omogućuju da tablica “racun_restoran” pohranjuje sve relevantne informacije o računima izdanima od strane restorana, što pomaže u praćenju financijskih transakcija i upravljanju restoranima unutar hotela.

```
CREATE TABLE racun_restoran (
        id_racun_restoran INT AUTO_INCREMENT PRIMARY KEY,
        datum DATE NOT NULL,
        cijena DECIMAL(10, 2) NOT NULL,
        id_restoran INT,
        FOREIGN KEY (id_restoran) REFERENCES restoran(id_restoran)
);
```

8.23 Tablica gost_restoran

Tablica “gost_restoran” pohranjuje podatke o gostima koji su koristili usluge restorana unutar hotela. Svaki atribut je detaljno opisan u nastavku:

“id_gost”: Identifikator gosta koji je koristio usluge restorana. Ovaj atribut je tipa INT i služi kao vanjski ključ (FOREIGN KEY) koji se povezuje s atributom “id_gost” u tablici gost. Ovaj atribut je dio primarnog ključa (PRIMARY KEY) tablice “gost_restoran”.

“id_restoran”: Identifikator restorana koji je posjetio gost. Ovaj atribut je tipa INT i služi kao vanjski ključ (FOREIGN KEY) koji se povezuje s atributom “id_restoran” u tablici “restoran”. Ovaj atribut je dio primarnog ključa (PRIMARY KEY) tablice “gost_restoran”.

“status”: Opisuje status posjete gosta restoranu, kao što je "rezervirano", "u tijeku", "završeno", itd. Ovaj atribut je tipa VARCHAR s maksimalnom duljinom od 50 znakova. Polje je obavezno (NOT NULL), što znači da svaki unos mora imati definiran status.

Ovi atributi omogućuju da tablica “gost_restoran” pohranjuje sve relevantne informacije o posjetama gostiju restoranima, što pomaže u praćenju i upravljanju uslugama koje gosti koriste unutar hotela.

```
CREATE TABLE gost_restoran (
        id_gost INT,
        id_restoran INT,
        status VARCHAR(50) NOT NULL,
        PRIMARY KEY (id_gost, id_restoran),
        FOREIGN KEY (id_gost) REFERENCES gost(id_gost),
        FOREIGN KEY (id_restoran) REFERENCES restoran(id_restoran)

);
```

8.24 Tablica_vrsta_placanja

Tablica “racun_vrsta_placanja”služi za povezivanje računa s različitim vrstama plaćanja koje su korištene za taj račun. Svaki redak u ovoj tablici predstavlja vezu između određenog računa i određene vrste plaćanja.

“id_racun”: Ovaj atribut predstavlja identifikator računa koji je izdan. Svaki unos u ovoj koloni referencira se na primarni ključ (PRIMARY KEY) (“id_racun”) tablice “racun”.

“id_vrsta_placanja”: Ovaj atribut predstavlja identifikator određene vrste plaćanja koja je korištena za plaćanje određenog računa. Svaki unos u ovoj koloni referencira se na primarni ključ (PRIMARY KEY) (“id_placanje”) tablice “vrsta_placanja”.

Ova tablica omogućuje fleksibilno upravljanje različitim vrstama plaćanja koje su povezane s određenim računima u hotelu. Korištenjem ovog poveznog mehanizma, sustav može pratiti i analizirati kako su gosti plaćali svoje račune, pružajući bolji uvid u financijske transakcije hotela.

```
CREATE TABLE racun_vrsta_placanja (
        id_racun INT,
        id_vrsta_placanja INT,
```

```
    FOREIGN KEY (id_racun) REFERENCES racun(id_racun),
    FOREIGN KEY (id_vrsta_placanja) REFERENCES vrsta_placanja(id_placanje),
    PRIMARY KEY (id_racun, id_vrsta_placanja)
);
```

9 Upiti

9.1 Upiti – Marta Kralj

9.1.1 Upit 1

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, a tablicu skladiste_dobavljac u sd- preimenovanja smo učinili uz operator AS. Iz tablica smo izvukli podatke samo za redove u kojima atribut naziv u tablici skladiste počinje s riječi Toner- 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, skladiste I skladiste_dobavljac s operatorom JOIN ON na atributima id_dobavljac iz tablice dobavljac i id_skladiste iz tablice skladiste ten a atributima id iz tablice dobavljac i id_dobavljac iz tablice skladiste_dobavljac. 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 i atribut datum_provjere kako bi detektirali koji je zadnji radnik koji je provjeravao stanje i naručivao tonere. 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. Sortirali smo rezultate po atributu datum_provjere I to od najnovijeg datuma pomoću operatora DESC i ORDER BY.

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_proizvoda AS naziv_robe
FROM dobavljac AS d
JOIN skladiste_dobavljac AS sd ON d.id = sd.id_dobavljac
JOIN skladiste AS s ON s.id = sd.id_skladiste
WHERE s.naziv_proizvoda LIKE 'Toner%';
```

```
CREATE VIEW provjera_stanja AS
SELECT r.ime, r.prezime, rs.napomena, rs.datum_provjere
FROM radnik AS r
JOIN radnik_skladiste AS rs ON r.id = rs.id_radnik
WHERE rs.napomena LIKE '%toner%'
ORDER BY rs.datum_provjere DESC;
```

```
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;
```

9.1.2 Upit 2

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 Marta Kralj. Selektirali smo sve podatke iz tablice radnik pomoću SELECT naredbe osim reda koji u atributu ime sadrži Marta i atributu prezime sadrži Kralj pomoću NOT LIKE operatora. Kako bi označili i atribut ime i atribut prezime smo koristili operator AND. Na ovaj način se radnik Marta Kralj ne pojavljuje u tablici preostali_radnici, no i dalje možemo pristupiti podatcima Marte Kralj 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 Marta, a u atributu prezime Kralj te selektirali taj podatak.

U SQL-u to izgleda ovako:

```
CREATE VIEW preostali_radnici AS
SELECT *
FROM radnik
WHERE ime NOT LIKE 'Marta' AND prezime NOT LIKE 'Kralj' AND id!=10;


SELECT id
FROM radnik
WHERE ime LIKE 'Marta' AND prezime LIKE 'Kralj';
```

9.1.3 Upit 3

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 Marta Kralj. Selektirali smo sve podatke iz tablice radnik pomoću SELECT naredbe osim reda koji u atributu ime sadrži Marta i atributu prezime sadrži Kralj pomoću NOT LIKE operatora. Kako bi označili i atribut ime i atribut prezime smo koristili operator AND. Na ovaj način se radnik Marta Kralj ne pojavljuje u tablici preostali_radnici, no i

dalje možemo pristupiti podatcima Marte Kralj 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 Marta, a u atributu prezime Kralj te selektirali taj podatak.

U SQL-u to izgleda ovako:

```
CREATE VIEW preostali_radnici AS
SELECT *
FROM radnik
WHERE ime NOT LIKE 'Marta' AND prezime NOT LIKE 'Kralj' AND id!=10;

SELECT id
FROM radnik
WHERE ime LIKE 'Marta' AND prezime LIKE 'Kralj';
```

9.1.4 Upit 4

Voditelj nabave je zatražio evidenciju količine pojedinog proizvoda te kontakt dobavljača za svaki proizvod. To smo učinili na sljedeći način: kreirali smo pogled kolicina_proizvoda_i_dobavljaci te smo selektirali atribute naziv_proizvoda i kolicina_proizvoda iz tablice skladiste te atribute naziv, kontakt_osoba i telefon iz tablice dobavljac. Spojili smo tablice skladiste i skladiste_dobavljac operatorom LEFT JOIN (kako bi izbjegli duplikate proizvoda) na atributima id iz tablice skladiste i id_skladiste iz tablice skladiste_dobavljac. Operatorom LEFT JOIN smo spojili i tablicu dobavljac na atributima id_dobavljac iz tablice skladiste_dobavljac i atribut id iz tablice dobavljac. Na kraju smo sortirali proizvode po količini proizvoda od najmanje količine do najveće pomoću operatora ORDER BY.

U SQL-u to izgleda ovako:

```
CREATE VIEW kolicina_proizvoda_i_dobavljaci AS
SELECT s.naziv_proizvoda, s.kolicina_proizvoda, d.naziv, d.kontakt_osoba, d.telefon
FROM skladiste AS s
LEFT JOIN skladiste_dobavljac AS sd ON s.id = sd.id_skladiste
```

```
LEFT JOIN dobavljac AS d ON sd.id_dobavljac = d.id
ORDER BY s.kolicina_proizvoda;
```

9.1.5 Upit 5

Voditelj recepcije je zatražio evidenciju koliko soba postoji u hotelu po tipu sobe. To smo učinili na sljedeći način: kreirali smo pogled evidencija_soba. Selektirali smo atribut tip iz tablice soba te smo koristili operator COUNT kako bi prebrojali sve redove. Taj atribut smo nazvali broj_soba pomoću operatora AS. Na kraju smo grupirali podatke pomoću GROUP BY po atributu tip kako bi dobili grupirane tipove soba, a sortirali smo ih pomoću operatora ORDER BY po novom atributu broj_soba, no silazno (od najveće količine soba do najmanje) pomoću operatora DESC.

U SQL-u to izgleda ovako:

```
CREATE VIEW evidencija_soba AS
SELECT tip, COUNT(*) AS broj_soba
FROM soba
GROUP BY tip
ORDER BY broj_soba DESC;
```

9.2 Upiti - Lea Beletić

9.1.1 Upit 1

Menadžer restorana zatražio je evidenciju o ukupnom prihodu restorana odvojenu po tipu restorana. Za izračun ukupnog prihoda korislitli smo funkciju SUM() za zbroj cijena iz tablice racun_restoran, grupirajući ih po tipu restorana. Kroz JOIN operaciju, povezujemo podatke o restoranima iz tablice restoran s odgovarajućim računima iz tablice “racun_restoran”, koristeći njihovu zajedničku vezu preko identifikatora restorana.

Pomoću operatora AS kreirali smo novi atribut “ukupni prihod”.

```
CREATE VIEW PrihodRestoranaPoTipu AS
SELECT restoran.tip, SUM(racun_restoran.cijena) AS ukupni_prihod
FROM restoran
JOIN racun_restoran ON restoran.id = racun_restoran.id_restoran
GROUP BY restoran.tip;
```

9.1.2 Upit 2

Menadžer restorana zatražio je evidenciju koji su gosti posjetili mediteranski tip restorana.

Upit koristi podatke iz više tablica. Prva je tablica gost gdje imamo informacije o gostima, poput imena i prezimena, tablica gost_restoran gdje smo uspostavili vezu između gostiju i restorana, koristeći ID gostiju i ID restorana i tablica restoran (koja nam sadrži informacije o restoranima, uključujući tip restorana.

Kroz fukncju JOIN povezali smo podatke o gostima i restoranima, filtrirajući samo one goste koji su posjetili mediteranski restoran, što smo specificirali u WHERE uvjetu. Na kraju, rezultat sadrži ime i prezime gostiju (g.ime, g.prezime) te tip posjećenog restorana koji smo kreirali pomoću operatera AS (“posjeceni_restoran”).

```
CREATE VIEW posjetitelji_mediteranskog_restorana AS
SELECT g.ime, g.prezime, r.tip AS posjeceni_restoran
FROM gost AS g
JOIN gost_restoran AS gr ON g.id = gr.id_gost
JOIN restoran AS r ON gr.id_restoran = r.id
WHERE r.tip = 'Mediteranski';
```

9.1.3 Upit 3

Menadžer restorana zatražio je evidenciju ukupnog prihoda po tipu sobe.
Ovdje smo također koristili upite iz nekoliko tablica. Prva nam je tablica sobakoja sadrži informacije o sobama, uključujući tip svake sobe. Nadallje imamo tablicu zahtjev_odrzavanja koja povezuje zahtjeve održavanja sa sobama, koristeći ID sobe. Tablica rezervacija sadrži informacije o rezervacijama, uključujući ID gosta, a tablica racun ima informacije o računima, uključujući iznos plaćen za svaku rezervaciju.

Kroz JOIN operacije, upit povezuje podatke o sobama, zahtjevima održavanja, rezervacijama i računima. Pomoću funkcije SUM(izračunali smo ukupni prihod po tipu sobe, grupirajući ih po tipu sobe kroz GROUP BY funkciju. I ovdje smo koristili operater AS kako bi kreirali nove atribute

```
CREATE VIEW PrihodPoTipuSobe AS
SELECT soba.tip AS tip_sobe, SUM(racun.iznos) AS ukupni_prihod
FROM soba
JOIN zahtjev_odrzavanja ON soba.id = zahtjev_odrzavanja.id_soba
JOIN rezervacija ON zahtjev_odrzavanja.id_gost = rezervacija.id_gost
JOIN racun ON rezervacija.id_racun = racun.id
GROUP BY soba.tip;
```

9.1.4 Upit 4

Gost je zatražio samo sobe sa TVom, a recepcioner ih je filtrirao pomoću upita koji funkcionira na sljedeći način. Kroz JOIN operacije, upit povezuje podatke o sobama i sadržaju iz nekoliko tablica, filtrirajući samo one sobe koje sadrže televizor, što je specificirano u WHERE uvjetu.Tablice koje su povezane su:
Tablica soba koja sadrži informacije o sobama, uključujući broj sobe i njen tip, Tablica soba_sadrzaj gdje je uspostavljena veza između soba i njihovih sadržaja, koristeći ID sobe. Tablica sadrzaj sadrži informacije o dostupnom sadržaju u sobama, poput televizora.

```
CREATE VIEW SobeSaTV AS
SELECT soba.broj_sobe, soba.tip, sadrzaj.naziv
FROM soba
JOIN soba_sadrzaj ON soba.id= soba_sadrzaj.id_soba
JOIN sadrzaj ON soba_sadrzaj.id_sadrzaj = sadrzaj.id
WHERE sadrzaj.naziv = 'TV';
```

9.2 Upiti Iva Batur

9.2.1 Upit 1

Najčešći način plaćanja i mjesec u kojem je to plaćanje izvršeno najviše puta

Razumijevanje najčešćih načina plaćanja pomaže u boljem upravljanju financijama hotela. Ako većina gostiju koristi određeni način plaćanja, hotel bi mogao prilagoditi svoje usluge kako bi bolje odgovarale potrebama gostiju.

To smo provjerili na sljedeći način: koristili smo SELECT naredbu za odabir atributa 'nacin_placanja' iz tablice 'vrsta_placanja', koja sadrži informacije o različitim načinima plaćanja. Zatim smo pomoću funkcije 'MONTH' dobili mjesec iz datuma u polju 'r.datum' i to polje smo nazvali mjesec. Pomoću 'COUNT(*)' smo zbrojili broj pojava svakog načina plačanja za određeni mjesec. 'FROM' naredbom odredili smo izvore podataka, a to su : 'racun', 'racun_vrsta_placanja' te 'vrsta_placanja' . Podaci su grupirani po 'najcesci_nacin_placanja' i 'mjesec'. Rezultate smo sortirali pomoću ' ORDER BY' prema broju plaćanja, silaznim redoslijedom (DESC), što nam omogućava da se na vrhu prikaže najveći broj plaćanja.

U SQL-u to izgleda ovako:

```
SELECT
        vp.nacin_placanja AS najcesci_nacin_placanja,
MONTH(r.datum) AS mjesec,
COUNT(*) AS broj_placanja
FROM
   racun AS r
JOIN
   racun_vrsta_placanja AS rvp ON r.id = rvp.id_racun
JOIN
   vrsta_placanja AS vp ON rvp.id_placanje = vp.id
GROUP BY
   najcesci_nacin_placanja, mjesec
ORDER BY
   broj_placanja DESC
LIMIT 1;
```

9.2.2 Upit 2

U kojem mjesecu je bio najveći broj rezervacija u hotelu

Poznavanje mjeseca s najvećim brojem rezervacija omogućuje hotelu da prilagodi svoje marketinške strategije kako bi privukao više gostiju. Isto tako može pomoći hotelu za prilagođavanje troškova i prihoda kako bi se lakše nosili s promjenjivim prihodima tijekom godine. Uvelike može pomoći i za analizu kada je potrebno ponuditi posebne ponude i privući goste.

Najveći broj rezervacija provjerili smo na sljedeći način: u prvom dijelu upita koristili smo funkciju MONTH kako bi se izvukli samo mjesec iz datuma prijave. Korištenjem AS mjesec stvara se privremena oznaka za ovu izračunanu vrijednost kako bi je kasnije bilo lakše referencirati. Zatim smo, uz pomoću funkcije CUONT(*) izračunali broj rezervacija za svaki mjesec. COUNT funkcija broji sve retke u tablici 'rezervacija' za svaki pojedini mjesec. Pomoću FROM klauzule odabiremo tablicu rezervacija iz baze podataka. Ta tablica sadrži informacije o svim rezervacijama u hotelu. U nastavku koristimo GROUP BY kako bismo grupirali rezultate prema mjesecu. To znači da ćemo dobiti broj rezervacija za svaki pojedinačni mjesec. Pomoću ORDER BY sortiramo rezultate prema broju rezervacija u silaznom (DESC) redoslijedu kako bi prvo dobili mjesec s najvećim brojem rezervacija. LIMIT 1 koristili smo kako bi ograničili rezultate na samo jedan redak tj. osigurava nam da dobijemo samo mjesec s najvećim brojem rezervacija, ostali mjeseci nisu nam potrebni.

U SQL-u to izgleda ovako:

```
SELECT MONTH(datum_prijave) AS mjesec, COUNT(*) AS broj_rezervacija
FROM rezervacija
GROUP BY mjesec
ORDER BY broj_rezervacija DESC
LIMIT 1;
```

9.2.3 Upit 3

Prikaz svih gostiju koju su ostavili recenziju s ocjenom većom od 3

Identifikacija gostiju koji su bili zadovoljni svojim boravkom i koji su dali visoku ocjenu omogućava hotelu da prepozna što je dobro funkcioniralo i pruži slično iskustvo drugim gostima. Ovo može pomoći u poboljšanju općeg iskustva gostiju i kvalitete usluge hotela. Isto tako može koristiti gostu koji planira posjetiti hotel, ako gost vidi da su prethodni gosti bili zadovoljni svojim boravkom, to može povećati njihovo povjerenje u hotel i odluku o rezervaciji.

Taj upit smo postavili na sljedeći način: prvo smo pomoću 'SELECT' odabrali podatke za prikaz, te smo pomoću funkcije 'CONCAT' spojili imena i prezimena gostiju, nakon čega smo koristili 'AVG' kako bi izračunali prosječnu ocjenu koji su gosti ostavili. 'FROM' nam služi za navođenje tablica iz kojih se podaci selektiraju, u ovom slučaju to su bile tablica "gost" koju smo označili kao g, te tablica "recenzija" označena kao r. U ovom upitu koristili smo 'JOIN' kako bi spojili podatke iz tablica "gost" i "recenzija" na temelju identifikatora gosta (g.id) koji se podudara s identifikatorom gosta u tablici recenzija (r.id_gost). Podatke smo grupirali pomoću 'GROUP BY' po identifikatoru gosta 'g.id', što će nam izračunati prosječnu ocjenu za svakog gosta zasebno. I na kraju smo pomoću 'HAVING' filtrirali rezultate nakon grupiranja kako bi odabrali samo one rezultate čija je ocjena veća od 3.

U SQL-u to izgleda ovako:

```
SELECT
CONCAT(g.ime, ' ', g.prezime) AS ime_gosta
   AVG(r.ocjena) AS prosjecna_ocjena
FROM
   gost AS g
JOIN
   recenzija AS r ON g.id = r.id_gost
GROUP BY
   g.id
HAVING
   AVG(r.ocjena) > 3;
```

9.2.4 Upit 4

Prikaz svih gostiju koji su potrošili više od 150 eura u restoranu.

Omogućuje vlasnicima hotela praćenje potrošnje gostiju. Analizirajući koji gosti troše više novca, mogu procijeniti uspješnost svojih marketinških strategija, kvalitete hrane i usluge te eventualno identificirati područja za poboljšanje. Jednako tako pomaže prilagodbi usluga i poboljšanju ukupnog iskustva gostiju, što na kraju može dovesti do veće profitabilnosti poslovanja.

Ovaj upit kreirali smo na sljedeći način: pomoću 'SELECT' specificirali smo koje podatke želimo dobiti iz baze podataka. Koristili smo funkciju 'CONCAT' kako bi spojili ime i prezime gosta u jedno polje koje smo nazvali 'ime_gosta'. Nakon čega smo uz pomoću 'SUM' funkcije izračunali ukupan iznos koji je svaki gost potrošio u restoranu te taj izračun nazivamo 'ukupan_iznos_restoran'. Pomoću 'FROM' navodimo da iz tablice “gost“, „gost_restoran“ te „racun_restoran“ želimo povući podatke. Nakon čega smo pomoću 'JOIN' spojili tablice „gost“ i „gost_restoran“ i to prema identifikacijskom broju gosta, što nam služi kako bi dobili podatke o gostima koji su posjetili restoran. Još smo spojili i „gost_restoran“ i „racun_restoran“ tablice kako bismo dobili informacije o računima gostiju iz restorana. Grupirali smo ih pomoću 'GROUP BY' po identifikacijskom broju gosta (g.id) i tako dobili jedan redak za svakog gosta. Filtrirali smo rezultate s 'HAVING' uvjetom tako da dobijemo samo one goste koji su potrošili više od 150 eura.

U SQL-u to izgleda ovako:

```
SELECT
CONCAT(g.ime, ' ', g.prezime) AS ime_gosta,
SUM(rr.cijena) AS ukupni_iznos_restoran
FROM
   gost AS g
JOIN
   gost_restoran AS gr ON g.id = gr.id_gost
JOIN
   racun_restoran AS rr ON gr.id_restoran = rr.id_restoran
GROUP BY
   g.id
HAVING
   ukupni_iznos_restoran > 150;
```

9.2.5 Upit 5

Top 3 najčešće rezervirane sobe,prosječnu ocjenu i recenzije za njih

Recenzije gostiju za najčešće rezervirane sobe pružaju korisne povratne informacije o iskustvima drugih gostiju. Ovo može pomoći gostima da donesu informirane odluke o svojem boravku i izbjegnu sobe koje su možda loše ocijenjene. Isto tako analizirajući recenzije gostiju za najčešće rezervirane sobe, hotel može identificirati nedostatke ili aspekte koji bi se mogli poboljšati. Na temelju ovih informacija, hotel može poduzeti korake za poboljšanje iskustva gostiju i povećanje zadovoljstva gostiju.

Upit smo postavili na sljedeći način: kreirali smo pogled top_najcesce_rezervirane_sobe koji prikazuje najčešće rezervirane sobe. Pomoću naredbe 'SELECT' odabrali smo potrebene podatke. U ovom upitu to su 's.broj_sobe', 'rec.ocjena' i 'rec.komentar'. . Pomoću 'FROM' identificirali smo izvore podataka koji će se koristiti u upitu, a to su redom: 'soba', 'rezervacija', te 'top_najcesce_rezervirane_sobe' te smo ih spojili ('JOIN') kako bi dobili potreban skup podataka. Sobe smo povezali s rezervacijama preko 'id_sobe', rezervacije smo povezali s recenzijama preko 'id_rezervacija', a sobe su povezane s pogledom 'top_najcesce_rezervirane_sobe' . Rezultati se grupiraju pomoći 'GROUP BY' po ID-u sobe i broju sobe kako bi se osiguralo da se brojanje rezervacija obavlja na razini svake sobe. Pomoću 'ORDER BY' broj_rezervacija 'DESC' rezultati se sortiraju u silaznom redoslijedu tako da sobu s najvećim brojem rezervacija dobijemo na vrhu. 'LIMIT 3' ograničava broj rezultata na 3, što znači da će se prikazati samo 3 sobe s najvećim brojem rezervacija.

U SQL-u to izgleda ovako:

```
SELECT soba.broj_sobe, COUNT(*) AS broj_rezervacija, AVG(recenzija.ocjena) AS
prosjecna_ocjena
FROM soba
JOIN rezervacija ON soba.id_rezervacija = rezervacija.id
LEFT JOIN recenzija ON rezervacija.id = recenzija.id_rezervacija
GROUP BY soba.id
ORDER BY broj_rezervacija DESC
LIMIT 3;
```

9.2.6 Upit 6

Radnici koji rade u popodnevnoj smjeni i njihove rezervacije

Praćenje rezervacija koje su obavljene tokom određenih smjena omogućava procjenu učinaka radnika u smislu prodaje ili pružanja usluga gostima tokom tih smjena.

Upit smo postavili na sljedeći način: Linija 'SELECT' definira koje kolone će biti vraćene u rezultatu upita. Pomoću 'FROM' identificirali smo izvore podataka koji će se koristiti u upitu, u ovom slučaju to će biti iz tablice 'radnik'. Pomoću 'JOIN' spojili smo tablicu ' radnik_smjena_radnika' sa tablicom ' radnik.id' , tablicu 'smjena_radnika' sa 'radnik_smjena_radnika_id_smjena', te 'rezervacija' sa 'radnik.id'. 'WHERE' filtrira rezultate tako da se vrate samo redovi u kojima je naziv smjene 'Popodnevna smjena'.

U SQL-u to izgleda ovako:

```
SELECT radnik.ime, radnik.prezime, rezervacija.datum_prijave, rezervacija.datum_odjave
FROM radnik
JOIN radnik_smjena_radnika ON radnik.id = radnik_smjena_radnika.id_radnik
JOIN smjena_radnika ON radnik_smjena_radnika.id_smjena = smjena_radnika.id
JOIN rezervacija ON radnik.id = rezervacija.id_radnik
WHERE smjena_radnika.naziv = 'Popodnevna smjena';
```

9.3 Upiti – Marko

9.3.1 Upit 1

Da bismo bolje razumjeli kako se podaci u ovoj bazi koriste za svakodnevno poslovanje, evo jednog kompleksnijeg upita koji dohvaća detalje o rezervacijama, uključujući informacije o gostima, radnicima koji su obradili rezervaciju, sobama koje su rezervirane, te račune i usluge povezane s tim rezervacijama:

Dohvaća detalje o rezervacijama - Upit prikuplja podatke o svakoj rezervaciji, uključujući datume prijave i odjave te broj gostiju.

Informacije o gostima - Pridružuje podatke o gostima koji su izvršili rezervaciju.

Informacije o radnicima - Pridružuje podatke o radnicima koji su obradili rezervaciju.

Informacije o sobama - Dohvaća podatke o sobama koje su rezervirane, uključujući broj sobe, tip sobe i cijenu noćenja.

Financijski podaci - Prikazuje iznos računa povezanog s rezervacijom.

Dodatne usluge - Koristeći funkciju GROUP_CONCAT, grupira nazive usluga povezanih s računom u jedan zapis.

```
SELECT
    r.id AS rezervacija_id,
    g.ime AS gost_ime,
    g.prezime AS gost_prezime,
    r.datum_prijave, r.datum_odjave, r.broj_gostiju, rd.ime AS radnik_ime, rd.prezime AS
    radnik_prezime,
    s.broj_sobe,
    s.tip AS soba_tip,
    s.cijena_nocenja,
    rc.iznos AS racun_iznos,
    GROUP_CONCAT(u.naziv SEPARATOR ', ') AS usluge
    FROM
    rezervacija r
    JOIN gost g ON r.id_gost = g.id
    JOIN radnik rd ON r.id_radnik = rd.id
    JOIN soba s ON s.id_rezervacija = r.id
    JOIN racun rc ON r.id_racun = rc.id
```

```
    LEFT JOIN racun_usluge ru ON rc.id = ru.id_racun
    LEFT JOIN usluge u ON ru.id_usluga = u.id
GROUP BY
    r.id, g.ime, g.prezime, r.datum_prijave, r.datum_odjave, r.broj_gostiju, rd.ime, rd.prezime,
    s.broj_sobe, s.tip, s.cijena_nocenja, rc.iznos;
```

9.3.2 Upit 2

Ovaj upit koristi CREATE VIEW za stvaranje dva pogleda: detalji_gosta i detalji_gosta_potrosnja. Cilj je olakšati dohvaćanje složenih podataka o gostima, njihovim rezervacijama, računu i korištenim uslugama u hotelu. Pogledi su korisni za analize i izvještavanje, omogućujući jednostavan pristup ključnim informacijama i poboljšavajući upravljanje podacima.

Pogled detalji_gostaPogled detalji_gosta kombinira podatke iz nekoliko tablica (gost, rezervacija, racun, racun_usluge, usluge) kako bi pružio sveobuhvatan pregled gostiju, njihovih rezervacija, računa i korištenih usluga.

Ključne Informacije:

Podaci o gostima: ime, prezime, datum rođenja, adresa, telefon i email.
Informacije o rezervaciji: datum prijave, datum odjave i broj gostiju.
Podaci o računu: datum izdanja računa i ukupni iznos.
Korištene usluge: popis usluga koje je gost koristio tijekom boravka.
Pogled detalji_gosta_potrosnja
Pogled detalji_gosta_potrosnja dodatno filtrira goste koji su potrošili više od 500 jedinica valute, pružajući uvid u goste koji su ostvarili značajne troškove tijekom svog boravka.

Ključne Informacije:

Sve informacije iz pogleda detalji_gosta
Dodatno filtriranje: prikazuje samo goste čiji su računi veći od 200 jedinica valute.
Korist

Ovi pogledi olakšavaju menadžmentu hotela praćenje i analizu podataka o gostima. Posebno su korisni za:

Identifikaciju ključnih gostiju: detalji_gosta_potrosnja omogućuje brzo prepoznavanje gostiju koji su potrošili značajne iznose, što može pomoći u ciljanju marketinških aktivnosti i poboljšanju usluga.

Sveobuhvatan pregled: detalji_gosta omogućuje pregled svih relevantnih informacija o gostima i njihovim interakcijama s hotelom na jednom mjestu.

Poboljšanje usluga: Analiza podataka o korištenim uslugama može pomoći u optimizaciji ponude i prilagođavanju usluga potrebama gostiju.

```
CREATE VIEW detalji_gosta_potrosnja AS
SELECT
    g.id AS gost_id,
    g.ime AS gost_ime,
    g.prezime AS gost_prezime,
    g.datum_rodenja AS datum_rodenja,
    g.adresa AS adresa,
    g.telefon AS telefon,
    g.email AS email,
    r.id AS rezervacija_id,
    r.datum_prijave AS datum_prijave,
    r.datum_odjave AS datum_odjave,
    r.broj_gostiju AS broj_gostiju,
    rc.id AS racun_id,
    rc.datum AS datum_racuna,
    rc.iznos AS iznos_racuna,
    GROUP_CONCAT(DISTINCT u.naziv SEPARATOR ', ') AS usluge
FROM
    gost g
    JOIN rezervacija r ON g.id = r.id_gost
    JOIN racun rc ON r.id_racun = rc.id
    LEFT JOIN racun_usluge ru ON rc.id = ru.id_racun
    LEFT JOIN usluge u ON ru.id_usluga = u.id
WHERE
    rc.iznos > 200
GROUP BY
    g.id, g.ime, g.prezime, g.datum_rodenja, g.adresa, g.telefon, g.email, r.id,
r.datum_prijave, r.datum_odjave, r.broj_gostiju, rc.id, rc.datum, rc.iznos;
```

```
SELECT * FROM detalji_gosta_potrosnja;;
```

9.3.3 Upit 3

Dohvaćamo sve goste koji su ostavili za ocjenu recenzije 5 i id njihove rezervacije

```
SELECT gost.ime, gost.prezime, rezervacija.datum_prijave, rezervacija.datum_odjave,
recenzija.ocjena
FROM gost
JOIN recenzija ON gost.id = recenzija.id_gost
JOIN rezervacija ON recenzija.id_rezervacija = rezervacija.id
WHERE recenzija.ocjena = '5';
```

9.3.4 Upit 4

Dohvaćanje svih radnika koji rade u jutarnjoj smjeni i njihove detalje o radnom mjestu

```
SELECT radnik.ime, radnik.prezime, radno_mjesto.naziv AS radno_mjesto,
smjena_radnika.naziv AS smjena
FROM radnik
JOIN radnik_smjena_radnika ON radnik.id = radnik_smjena_radnika.id_radnik
JOIN smjena_radnika ON radnik_smjena_radnika.id_smjena = smjena_radnika.id
JOIN radno_mjesto ON radnik.id_radno_mjesto = radno_mjesto.id
WHERE smjena_radnika.naziv = 'Jutarnja smjena';
```

9.3.5 Upit 5

Dohvaćanje ukupnog iznosa potrošenog od strane svakog gosta

```
SELECT gost.ime, gost.prezime, SUM(racun.iznos) AS ukupno_potroseno
FROM gost
JOIN rezervacija ON gost.id = rezervacija.id_gost
JOIN racun ON rezervacija.id_racun = racun.id
GROUP BY gost.ime, gost.prezime;
```

10 Zaključak

Naša baza podataka za hotelijerstvo predstavlja temeljnu strukturu za učinkovito upravljanje svim ključnim aspektima poslovanja hotela. Kroz ovaj projekt, osigurali smo da naša baza podržava sve osnovne operacije, uključujući rezervacije, upravljanje gostima, evidenciju radnika i stanje skladišta. Iako smo svjesni da uvijek postoji prostor za unaprjeđenje, vjerujemo da smo uspješno ostvarili ciljeve postavljene u okviru ovog projekta.

Naša baza podržava različite operacije hotela, uključujući evidenciju rezervacija, stanje soba, upravljanje radnicima te praćenje stanja skladišta i suradnju s dobavljačima. Kroz njezinu implementaciju omogućili smo efikasno praćenje podataka o gostima, njihovim rezervacijama te praćenje radnika i stanja u skladištu. Time smo stvorili čvrstu osnovu za uspješno poslovanje hotela.

Unatoč postignućima, svjesni smo da naša baza može biti nadograđena kroz integraciju s drugim sustavima poput sustava za online rezervacije ili CRM sustava. Također, daljnji razvoj može uključiti implementaciju napredne analitike podataka kako bi se bolje razumjeli obrasci rezervacija i potrebe gostiju, te personalizirane preporuke putovanja kako bi se poboljšalo korisničko iskustvo.

Koristili smo različite alate i metode tijekom razvoja baze, uključujući Discord i WhatsApp za komunikaciju, GitHub i git za verzioniranje i dijeljenje koda te MySQL Workbench za izradu i provjeru strukture baze podataka. Također, koristili smo Notepad++, Google Docs i Python za generiranje i provjeru podataka te Lucidchart za izradu ER dijagrama.

U konačnici, smatramo da smo stvorili robustnu bazu podataka koja će podržati operativne potrebe hotela te pružiti temelj za daljnji razvoj i unaprjeđenje poslovanja u budućnosti.