Baza podataka za rezervacije u lokalima

Gotal, Kristina

Sveučilište Josipa Jurja Strossmayera u Osijeku

Fakultet primijenjene matematike i informatike

Sveučilišni prijediplomski studij matematike

BAZA PODATAKA ZA REZERVACIJE U LOKALIMA

Završni rad

Kandidat:

Kristina Gotal

Osijek, 2024

Sveučilište Josipa Jurja Strossmayera u Osijeku

Fakultet primijenjene matematike i informatike

Sveučilišni prijediplomski studij matematike

BAZA PODATAKA ZA REZERVACIJE U LOKALIMA

Završni rad

Mentor:

doc. dr. sc. Mateja Đumić

Kandidat:

Kristina Gotal

Osijek, 2024

Sadržaj

1 | Uvod

Baze podataka su neizostavni dio naših života iako na prvu možda nismo svjesni toga, ali zapravo što god koristimo od aplikacija i raznih sustava sve se u pozadini temelji na bazama podataka. Analitičar podataka po imenu Radi iz CONTEGENE-a je možda najbolje sažeo važnost baza podataka u svakodnevnom životu: „Analiza podataka je budućnost, a budućnost je SADA! Svaki klik mišem, pritisak tipke na tipkovnici, prelazak prstom ili dodir koristi se za oblikovanje poslovnih odluka. Danas se sve vrti oko podataka. Podaci su informacije, a informacije su moć .“ [5]

U ovom završnom radu će se na primjeru pravljenja rezervacija za lokale prikazati koliko zapravo baze podataka, uz pravilno održavanje, olakšavaju život i pružaju dugoročno rješenje za probleme.

Ideja za izradu baze podataka za rezervaciju stolova u prostorima u određenim gradovima je nastala proučavanjem općenitog postupka rezervacije stolova u određenim lokalima. Trenutno se rezervacije najčešće zapisuju ručno i potrebno je svaki put provjeravati koliko ima rezervacija i ima li slobodnih mjesta, koji stolovi su slobodni i slično. Navedeni postupak rezervacije oduzima dosta vremena i lako može doći do pogrešne procjene slobodnih stolova što rezultira nezadovoljstvom gostiju. Ova baza podataka bi riješila navedene probleme jer bi se na jednostavan način odvijao cijeli proces rezervacije i omogućio dodatne stvari kao što su lak način otkazivanja rezervacija čime bi se automatski oslobodili stolovi, uvid u popunjenost prostora i mogućnost analize po datumima i slično. U drugom poglavlju dan je kratki pregled baza podataka, dok je u trećem poglavlju prikazano je oblikovanje baze podataka koje se može podijeliti u tri faze: konceptualna faza, logička faza i fizička faza. U četvrtom poglavlju objašnjene su komponente baza podataka kao što su: tablice, upiti i procedure, zatim je u zadnjem poglavlju pisano testiranje baze podataka pomoću napravljene desktop aplikacije.

2 | Baza podataka

Baza podataka je organizirani skup podataka spremljen na nekom računalnom sustavu koji je lako dostupan korisnicima i aplikacijama. Sastoji se od skupa koji sadrži međusobno povezane podatke. Omogućava dohvat, dodavanje, mijenjanje i brisanje podataka. Sustav za upravljanje bazom podataka (SUBP, engl. DBMS – Database Management System) je programska podrška koja služi za izvođenje svih operacija nad bazom podataka. On oblikuje fizički prikaz baze obzirom na logičku strukturu te je zadužen za sve operacije s podacima. Operacije nad bazom mogu biti kreiranje strukture, mijenjanje i dohvaćanje podataka, administracija, brisanje i slično. Također, sustav za upravljanje bazom podataka određuje fizičku lokaciju podataka, brine se o administraciji i obnovi podataka nakon rušenja baze podataka. [1]

2.1 Kratka povijest baza podataka

Potreba za praćenjem i upravljanjem ograničenih sredstava navela je prva ljudska bića na pohranjivanje informacija kako bi znali donijeti ispravne i informirane odluke, ne znajući da su zapravo oni rani usvojitelji upravljanja bazama podataka. Međutim, drevne civilizacije kao što su Egipćani i Sumerani znale su što rade kada su uvele računovodstvene tehnike za praćenje podataka i razumijevanje njihovih svakodnevnih života. Povijest računala i baza podataka međusobno su povezane. Automatizacija baza podataka je započela 1960-ih kada je uporaba računala postala isplativija opcija za privatne organizacije. Edgar Frank Codd je 1970-ih objavio važan rad pod nazivom „Relacijski model podataka za velike zajedničke banke podataka”[1]. U Coddovu članku koji je služio kao uvod u relacijsku bazu podataka, promijenjen je način ljudskog razmišljanja o bazama podataka. D. D. Chamberlin i R. F. Boyce koji su bili zaposleni u IBM-u, objavili su 1974. godine rad pod nazivom “SEQUEL” (engl. *Structured English Query Language*). Cilj članka bio je približavanje relacijskih baza podataka korisnicima.

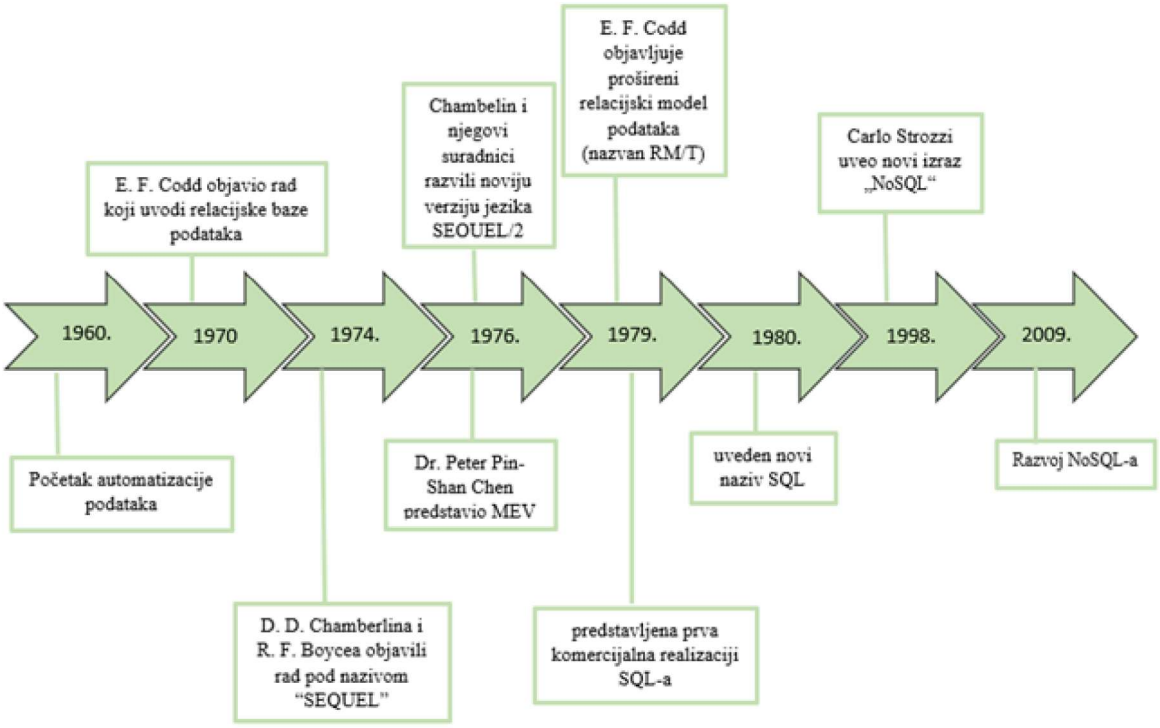
Chambelin i njegovi suradnici su 1976. godine razvili noviju verziju jezika pod nazivom SEQUEL/2. Nedugo nakon toga izrađen je prototip nazvan Sistem R u koji je Oracle kompanija implementirala SEQUEL/2. Nakon što je razvijena teorija modela podataka uočeno je da navedeni relacijski model nije u potpunosti točan te da ima niz nedostataka. Na osnovu toga definiran je prošireni relacijski

[1] engl. *"A Relational Model of Data for Large Shared Data Banks“*

model. Prema tome postoje dvije vrste modela: relacijski model (RM) i prošireni relacijski model (RM/T). Codd je svoj prošireni relacijski model podataka (nazvan RM/T) objavio nakon razvoja teorije modela podataka (Codd, 1979.) u okviru koje je razvijeno niz semantički bogatih koncepata. Ovi koncepti omogućavaju na formalan način ugradnju interpretacije podataka u model na formalan način. Osnovni koncepti za izgradnju strukture relacijskoga modela podataka su relacije. Relacija u relacijskome modelu podataka je slična relaciji u matematici s jednom veoma vrlo bitnom razlikom, a to je da su relacije u relacijskome modelu podataka vremenski promjenljive. Osnovna karakteristika relacijskog modela je jednostavnost i jaka teorijska osnova koja omogućava precizno definiranje struktura podataka i njihovih međusobnih odnosa.

Peter Pin-Shan Chen, američki znanstvenik rođen na Tajvanu, je 1976. godine predložio novi model baza podataka nazvan model entitet – veza (MEV, engl. Entityrelationship model, ER). MEV se nalazi u većini knjiga u kojima se spominje razvoj informacijskih sustava i baza podataka. Njegov rad „1st International Conference on Very Large Databases" smatra se jednim od najutjecajnijih radova u informatičkoj povijesti. MEV detaljno i precizno opisuje potrebe organizacije za informacijama ili poslovnog procesa za funkcijama. Shema modela podataka jednostavna je za razumijevanje i komunikaciju korisnika i projektanta. Postoje četiri osnovna cilja modeliranja entiteta i veza, a to su: obuhvatiti sve potrebne informacije, pobrinuti se da se svaka informacija pojavljuje točno jednom, osigurati da se nijedna informacija ne može izvesti iz već modeliranih te stavljanje informacije na logičko i predvidivo mjesto. [3]

Tvrtka „SDL" (danas Oracle korporacija) radi na razvoju Oracle baze podataka te su 1979. godine predstavili prvu komercijalnu realizaciji SQL-a kupcima. Kratica SEQUEL bila je zaštitni znak zrakoplovne tvrtke Hawker Siddeley iz Ujedinjenog Kraljevstva koja je zabranila korištenje tog imena te je oko 1980. godine uveden novi naziv SQL koji je ostao i do dan danas. SQL je standardiziran zahvaljujući organizacijama ANSI (engl. American National Standards Institute) i ISO (engl. The International Organization for Standardization). Carlo Strozzi je 1998. godine stvorio svoju bazu podataka koju je nazvao Strozzi NoSQL pritom je uveo novi izraz „NoSQL". No taj izraz nije bio ponovno upotrebljen sve do 2009. godine kada je programer Johan Oskarsson organizirao skup „Distribuiranih, nerelacijskih baza podataka otvorenog koda". NoSQL baze podataka sve se više koriste u slučaju velike količine podataka i web aplikacijama koje rade u stvarnom vremenu. Jednostavnog su dizajna, pružaju fleksibilnost i laku promjenu podataka. U NoSQL bazama podataka razlikujemo četiri glavna tipa: ključ-vrijednost bazu podataka, dokument bazu podataka, bazu podataka temeljenu na stupcima i graf bazu podataka. Slika 2.1 prikazuje vremensku liniju povijesti baza podataka. [3]



Slika 2.1: Povijest baza podataka.

3 | Razvoj baze podataka

Razvoj baze podataka je kompleksan proces jer je potrebno dobro poznavanje poslovnih procesa same organizacije za koju se kreira baza podataka. Kompleksnost razvoja baze podataka raste povećanjem individualnih komponenata poslovanja organizacije jer raste broj eniteta i veza koji moraju zadržati traženu funkcionalnost. Oblikovanje baze podataka se može podijeliti na tri faze:

1. konceptualna faza,
2. logička faza,
3. fizička faza.

U nastavku će detaljno biti objašnjena svaka faza.

3.1 Konceptualno oblikovanje modela

Glavni cilj prve faze oblikovanja baze podataka jest stvoriti konceptualnu shemu cijele baze sastavljenu od entiteta, atributa i veza. Konceptualna shema opisuje sadržaj baze i način povezivanja podataka u njoj. Ona je jezgrovita i lako razumljiva korisnicama. Konceptualni model je neovisan o implementaciji te daje cjelovit i razumljiv opis podataka informacijskog sustava. Konceptualna shema obuhvaća potrebe poslovnog procesa, zasniva se na trenutnim potrebama, ali može uključivati i buduće potrebe. Njezinim korištenjem smanjuju se pogreške i daje se osnova za fizički dizajn baze podataka. Za izradu i obradu podataka unutar baze podataka odgovoran je sustav za upravljanje bazom podataka koji se kategorizira prema modelu koji podržava:

- Relacijski model

  Zasnovan je na matematičkom pojmu relacije. I podaci i veze među njima prikazuju se tablicama koje se sastoje od redaka i stupaca.

- Mrežni model

  Baza je prikazana mrežom koja se sastoji od čvorova i usmjerenih lukova. Čvorovi predstavljaju tipove zapisa, tj. slogove podataka, a lukovi definiraju veze među tipovima zapisa.

- Hijerarhijski model

  Baza je prikazana jednim stablom (hijerarhijom) ili skupom stabala. Svako stablo sastoji se od čvorova i veza „nadređeni-podređeni“ između čvorova.

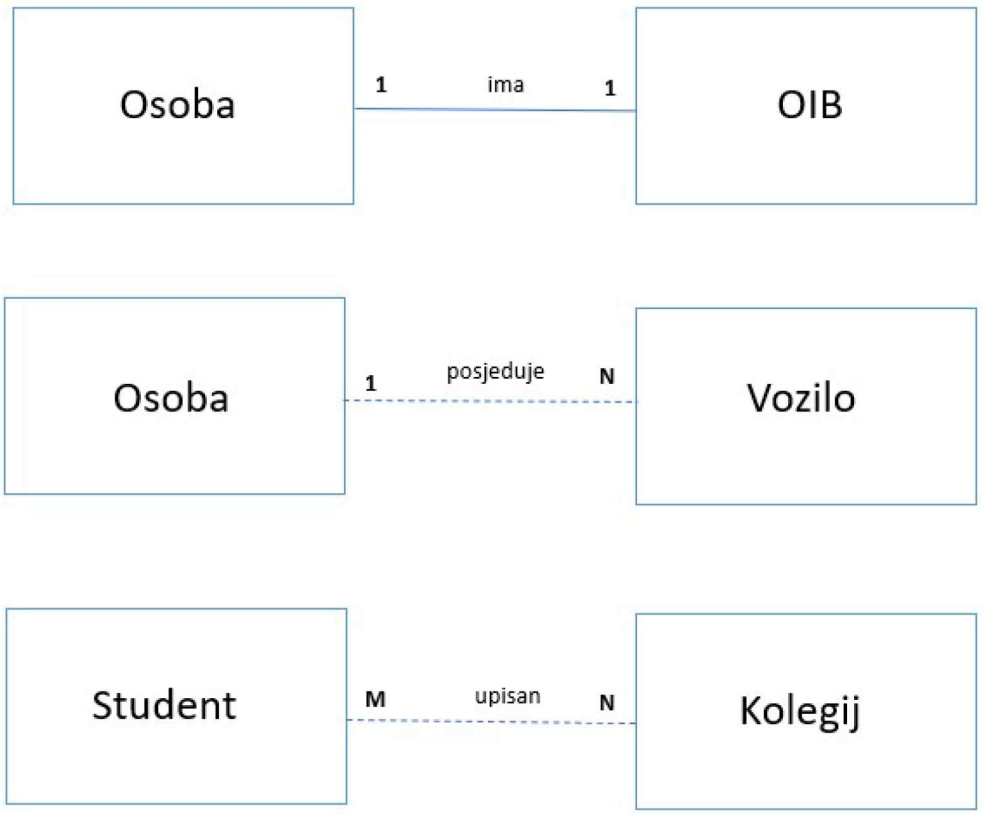
- Objektni model

  Inspiriran je objektno-orijentiranim programskim jezicima. Baza je prikazana kao skup trajno pohranjenih objekata koji se sastoje od svojih internih „atributa“ (podataka) i „metoda“ (operacija) za rukovanje tim podacima. Svaki objekt pripada nekoj klasi.

3.1.1 Model entitet-veza

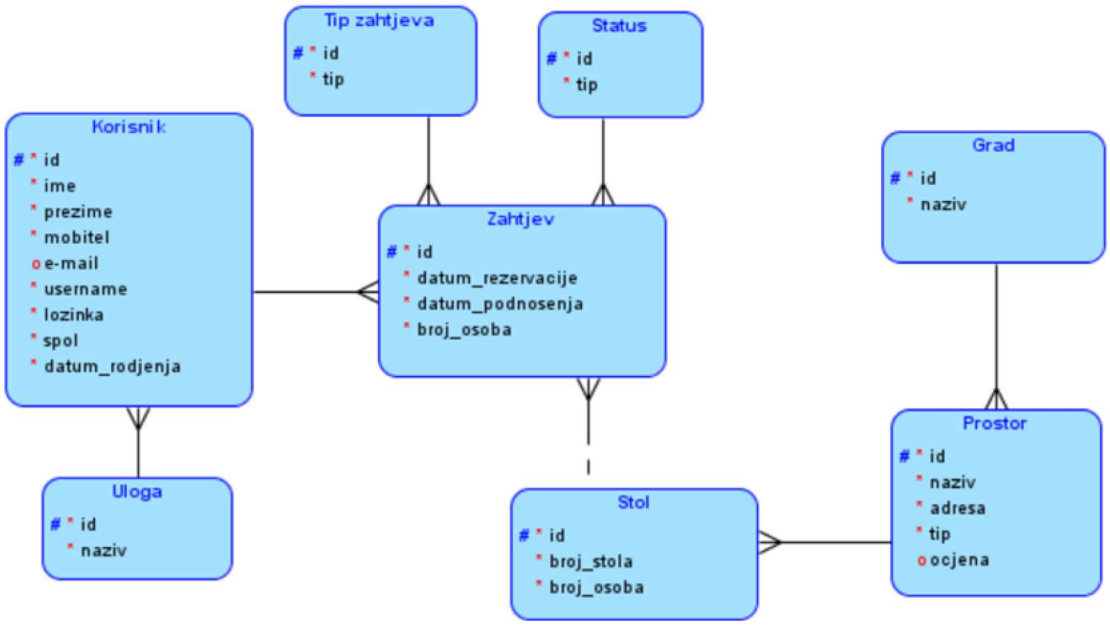
Svaki model entitet veza (MEV) sastoji se od entiteta, atributa i veza. Entitet može biti bilo koji objekt u stvarnom svijetu koji se razlikuje od drugih objekata i jedinstven je. Svaku pojavu nekog entiteta nazivamo instanca entiteta. Atributi predstavljaju karakteristike entiteta odnosno daju nam njihov detaljniji opis. Entiteti su međusobno povezani vezama koje prikazuju njihov odnos. Veze mogu biti mandatorne ili opcionalne te nam prikazuju radi li se o jednini ili množini odnosno koliko je pojava jednog entiteta povezano s pojavom drugog entiteta. Ako se radi o mandatornoj vezi koristit ćemo riječ "*mora*", ako se radi o opcionalnoj vezi koristit ćemo riječ "*može*". Slika 3.1. prikazuje kardinalnost veza koja može biti: 1:1 (jedan naprema jedan), 1:M (jedan naprema više) i M:N (više naprema više). Kod veza je potrebno obratiti pažnju na redundantne veze. Ako imamo dvije ili više veza koje povezuju dva tipa entiteta, moramo dobro proučiti značenje pojedine veze. Naime, ako veze imaju različito značenje tada nisu redundantne, a u slučaju da je značenje isto smatramo ih redundantnima i potrebno je jednu vezu maknuti.[4]

Prva veza 1:1 je mandatorna i prikazuje kako svaka osoba mora imati jedinstveni OIB (puna linija). Druga veza 1:M je opcionalna te prikazuje kako svaka osoba može posjedovati više automobila (isprekidana linija). Treća veza je također opcionalna i prikazuje kako svaki student može upisati nula ili više kolegija. S druge strane svaki kolegij može slušati nula ili više studenata.



Slika 3.1: Kardinalnost veza u MEV modelu

Slika 3.2 prikazuje izrađen model entiteta i veza za potrebe ovog završnog. MEV se sastoji od 8 entiteta. Svaki entitet u sebi sadrži svoj jedinstveni identifikator. Entitet grad sadrži obvezan atribut naziv. Entitet prostor sadrži naziv, adresu te tip prostora koji može biti: fast food, palačinkarnica i sl. Entitet stol sadrži broj stola te maksimalan broj ljudi koji može zauzeti određeni stol. Entitet zahtjev sastoji se od datuma rezervacije te datuma kada je zahtjev podnesen kako bi se prvo pregledavali stariji zahtjevi. Entitet tip zahtjeva sastoji se od atributa tip koji govori je li zahtjev podnesen za rezervaciju ili za otkazivanje. Slično tome, atribut tip unutar entiteta status prikazuje je li trenutni zahtjev odobren, otkazan ili je na čekanju. Entitet korisnik sadrži sve potrebne osobne podatke o korisniku te njegove podatke za prijavu. U entitetu korisnik pojavljuje se opcionalan atribut email dok su ostali atributi mandatorni u svim entitetima. Entitet uloga pomaže pri dodjeli točne uloge korisniku koji se prijavljuje ovisno o tome je li administrator ili korisnik. Slika 3.2 također prikazuje odnose među entitetima. Sve veze su jedan naprema više. Na primjer veza prostor-grad je mandatorna s obje strane i prikazuje kako u jednom gradu može biti više prostora, a prostor se može nalaziti u točno jednom gradu. Veza stol-zahtjev je opcionalna sa strane entiteta stol te prikazuje kako stol ne mora, ali može biti rezerviran.



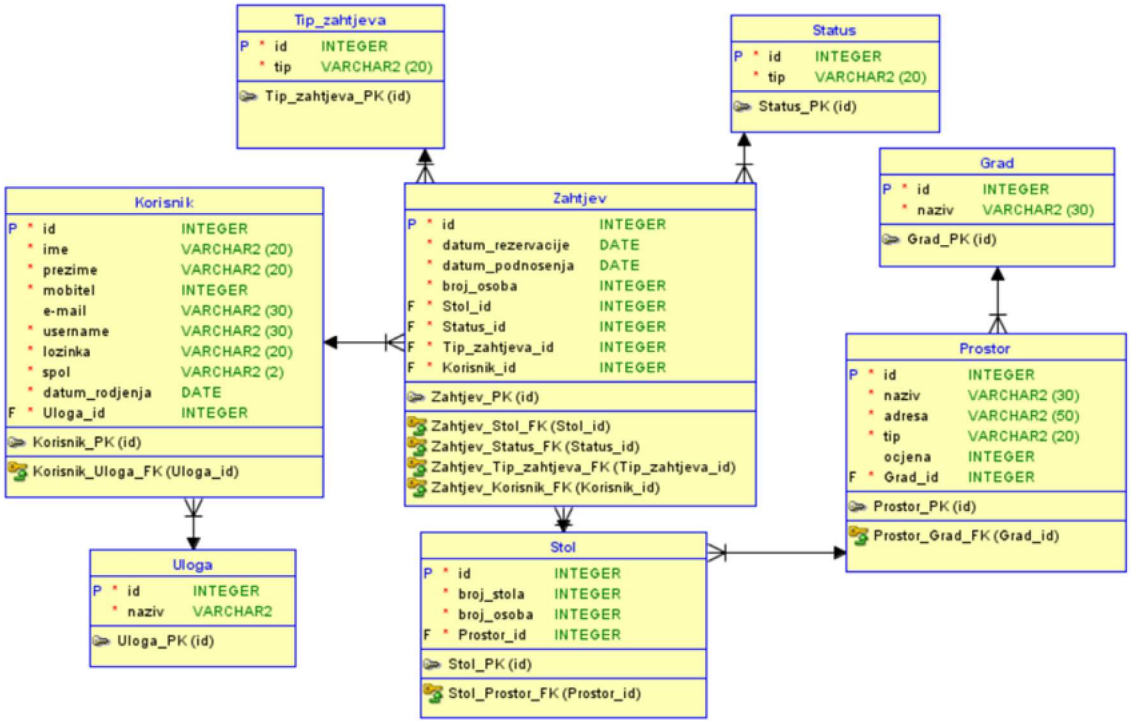
Slika 3.2: MEV model za bazu podataka za rezervaciju mjesta u lokalima.

3.2 Logičko oblikovanje modela

Glavni cilj druge faze oblikovanja baze podataka je stvoriti relacijsku shemu koja opisuje logičku strukturu baze u skladu s pravilima relacijskog modela. Ona je većini korisnika manje razumljiva od konceptualne. Entiteti i veze iz MEV-a pretvaraju se u relacije. Jedan stupac relacije sadrži vrijednost jednog atributa (za entitet ili vezu). Vrijednost atributa mora biti jednostruka i jednostavna odnosno ne ponavlja se i ne da se rastaviti na dijelove . Jedan redak relacije opisuje instancu entiteta ili bilježi vezu između dva ili više entiteta.

3.2.1 Relacijski model

Na temelju pojašnjenog MEV-a (Slika 3.2), napravljen je relacijski model koji prikazuje slika 3.3 Također, na njemu je moguće vidjeti detaljnije informacije o svakoj tablici i tipovima atributa i međusobnim vezama. Kod pretvorbe MEV modela u relacijski model svaki entitet postat će jedna tablica, svaka instanca entiteta jedan redak, a svaki atribut jedan stupac tablice. Primarni jedinstveni identifikator postaje primarni ključ te se veze ostvaruju koristeći strani ključ.



Slika 3.3: Relacijski model za bazu podataka za rezervaciju mjesta u lokalima.

Kod pretvorbe veze jedan naprema više (1:M), atribut koji predstavlja primarni ključ na strani entiteta koji ima vezu "1" se stavlja u tablicu entiteta na strani veze "M" kao strani ključ. Kod pretvorbe veze više naprema više (N:M) stvara se nova tablica i dvije veze jedan naprema više. U novoj tablici oba atributa predstavljaju strane ključeve koji su ujedno i primarni ključevi. Odnosno, nalazi se složeni primarni ključ (sastoji se od više stupaca) koji se sastoji od stranih ključeva. Svaki stupac baze ima određeni tip podataka (engl. *data type*)[2]. Neki od osnovnih tipova podataka su:

- *char*: sprema znakovni niz fiksne duljine,
- *varchar*: sprema znakovni niz promijenjive duljinje,
- *integer*: sprema cijele brojeve,
- *date*: sprema datum i vrijeme,
- *number*: sprema brojeve s pomičnim zarezom i cijele brojeve.

3.3 Fizičko oblikovanje modela

Fizičko oblikovanje modela je fizička shema cijele baze. Za njezino ostvarenje koristi se DBMS sustav koji je zasnovan na SQL-u te se brine o fizičkoj pohrani podataka na tvrdom disku kako bi ostali sačuvani i mogli se održavati i razvijati. U suštini fizička shema je niz SQL naredbi kojima se model iz faze dva - logički

model pretvaraju u tablice. Također, pomoću SQL naredbi se dodaju pomoćne strukture i mehanizmi za postizanje traženih performansi.

Danas postoji nekoliko važnih i široko zastupljenih DBMS-a:

- DB2 - skup je proizvoda relacijske baze podataka koje je izgradio IBM, sadrži jak sigurnosni režim te je iznimno popularan proizvod baze podataka u suvremenoj softverskoj industriji.
- Oracle - proizvod istoimene tvrtke, pokriva gotovo sve računalne platforme, kao što su: UNIX, Linux i MS Windows. Oracle baza podataka ima vlastitu mrežnu komponentu koja omogućuje komunikaciju preko mreža.
- MS SQL Server - Microsoftov proizvod, namijenjen poslužiteljskim računalima s operacijskim sustavima MS Windows.
- MySQL - besplatni proizvod tvrtke Oracle, popularan na raznim platformama, prvenstveno kao podrška web-aplikacijama.

Svi ti proizvodi uz DBMS u sebi sadrže i dodatne alate za razvoj aplikacija, administriranje baze i slično.

Pri izradi ovog rada korišten je MS SQL Server jer je besplatan i pruža alat Microsoft SQL Server Managment Studio koji nudi razne mogućnosti između kojih su i kreiranje nove baze podataka te mijenjanje postojećih shema baza podataka. Također, jedan od razloga zašto se koristi Microsoft SQL Server Managment Studio je taj što on omogućava lako upravljanje i povezivanje s MS SQL Serverom putem svog grafičkog sučelja što olakšava kreiranje i sam rad na bazi podataka jer nema potrebe za korištenjem naredbenih linija (engl. *command lines*).

3.3.1 SQL

SQL (engl. Structured Query Language) je jezik relacijskih baza podataka koji služi za pristupanje podacima. Nastao je kao potreba da se teorijska verzija relacijskog modela prenese u praksu. SQL je izuzetno lagan jezik koji je usredotočen na poslovnu logiku i model same baze podataka.
Naredbe SQL-a možemo podijeliti u 5 skupina.

1. naredbe za upite: *select*,
2. jezik za manipuliranje podacima (JMP naredbe): *insert*, *update*, *delete*,
3. jezik za definiranje podataka (JDP naredbe): *create*, *alter*, *drop*, *rename*, *truncate*,
4. naredbe za kontrolu transakcija: *commit*, *rollback*, *savepoint*,
5. naredbe za kontrolu podataka: *grant*, *revoke*, za dodjelu dozvola nad podacima.

Pomoću SQL naredbi kreirane su komponente baze podataka navedene u sljedećem poglavlju.

4 | Komponente baza podataka

U nastavku će biti objašnjeni elementi baze podataka.

4.1 Tablice

Tablice su napravljene na temelju relacijskog modela (Slika 3.3). Na slici 4.1 moguće je vidjeti kreiranje tablice *Prostor* pomoću grafičkog sučelja *Microsoft SQL Server Managment Studija*.

	Column Name	Data Type	Allow Nulls
🔑	prostor_id	int	☐
	grad_id	int	☑
	tip	varchar(20)	☐
	naziv	varchar(30)	☐
	adresa	varchar(50)	☐
	ocjena	int	☑
			☐

Slika 4.1: Tablica *Prostor* prikazana pomoću grafičkog sučelja.

Moguće je primijetiti kako se na vrlo jednostavan način može kreirati tablica i dodijeliti joj primarni ključ. Primarni ključ omogućava integritet svakog retka jer onemogućuje dupliciranje podataka unutar baze podataka za određenu tablicu. Referencijski integritet je značajka baze podataka u sustavima za upravljanje relacijskim bazama podataka. Time se osigurava da veze između tablica u bazi podataka budu točne primjenom ograničenja kako bi spriječili korisnike ili aplikacije da unesu netočne podatke ili pokazuju na podatke koji ne postoje. Kako bi se stvorila relacija i održao referencijski integritet, uz primarni ključ potrebno je koristiti strane ključeve. Referencijalni integritet je svojstvo koje osigurava da su veze između podataka ispravne odnosno postoji konzistentnost među podacima.[7] Strani ključ neće dozvoliti da se unese podatak s vrijednosti koja ne postoji u referenciranoj tablici s primarnim ključem. Pokuša li se u tablicu *Prostor* unijeti vrijednost *grad_id* koji ne postoji u tablici grad, pojavit će se greška. Na slici 4.3 prikazana je SQL naredba za kreiranje tablice. Usporedbom te slike sa slikom 4.1

jasno se vidi već ranije objašnjena prednost korištenja alata *Microsoft SQL Server Managment Studija*.

```
CREATE TABLE [dbo].[PROSTOR](
    [prostor_id] [int] NOT NULL,
    [grad_id] [int] NULL,
    [tip] [varchar](20) NOT NULL,
    [naziv] [varchar](30) NOT NULL,
    [adresa] [varchar](50) NOT NULL,
    [ocjena] [int] NULL,
 CONSTRAINT [prostor_pk] PRIMARY KEY CLUSTERED
(
    [prostor_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[PROSTOR]  WITH CHECK ADD  CONSTRAINT [prostor_fk_grad] FOREIGN KEY([grad_id])
REFERENCES [dbo].[GRAD] ([grad_id])
GO

ALTER TABLE [dbo].[PROSTOR] CHECK CONSTRAINT [prostor_fk_grad]
GO
```

Slika 4.2: Tablica *Prostor* prikazana pomoću SQL naredbe.

Kako bi se struktura tablice promijenila koriste se operacije dodavanja, brisanja ili modifikacije stupca na postojeću tablicu naredbom *alter*. Za dodavanje podataka u postojećoj tablicu koristi se ključna riječ *insert*. Kod dodavanja podataka u tablicu bitno je obratiti pažnju na poredak.

4.2 Upiti

Upite koristimo za pregledavanje, mijenjanje, brisanje ili dodavanje podataka. Praktični su za sažimanje i filtriranje podataka. Upite započinjemo ključnom riječi *select* koja je ujedno i najkorištenija naredba na svakoj relacijskoj bazi podataka. Omogućava projekcije podataka po želji, filtriranje podataka, spajanje više tablica, grupiranje podataka i slično. Nakon naredbe *select* potrebano je navesti nazive stupaca, zatim staviti ključnu riječ *from* i nazive tablica iz kojih se žele dohvatiti podaci, te u *where* dijelu popis uvjeta koje podaci trebaju zadovoljavati. Upit koji možemo dobiti pomoću jedne tablice naziva se jednostavan upit. Slika 4.3 prikazuje par primjera jednostavnih upita.
Prvi upit dohvaća sve podatke iz tablice *Korisnik* u kojima je vrijednost atributa *uloga_id*=1. Sljedeći upit iz tablice *Prostor* dohvaća naziv i adresu svih palačinkarnica. Treći primjer je upit koji dohvaća nazive svih gradova i *grad_id* iz tablice *Grad*. Zadnji primjer jednostavnog upita dohvaća sve podatke iz tablice *Prostor* kojima naziv adrese ima proizvoljno slovo nakon kojeg slijede slova r i a, zatim ide proizvoljni broj slova, te točno jedno slovo ispred slova h, nakon kojeg dolazi proizvoljan broj slova.

```
SELECT * FROM KORISNIK WHERE uloga_id = 1;
SELECT naziv, adresa FROM PROSTOR WHERE tip= 'Palačinkarnica';
SELECT naziv, grad_id from GRAD;
SELECT * FROM PROSTOR WHERE adresa LIKE '_ra%_h%';
```

Slika 4.3: Primjeri jednostavnih upita.

Ako je potrebno dohvatiti informacije iz više tablica, tablice se spajaju pomoću equi i non-equi spojeva. Za equi spoj koristi se operator =, dok se za non-equi spojeve koristi neki od sljedećih operatora: <, >, LIKE, IN, BETWEEN, <=, >=. Postoje tri vrste spojeva: unutarnji, vanjski i povratni. Upiti koji koriste neke od navedenih operatora nazivaju se složeni upiti i primjer je moguće vidjeti na slici 4.4. Za *status_id*=3 dohvaćamo *zahtjev_id*, tip zahtjeva, ime i prezime korisnika te datum rezervacije i naziv prostora na koji se odnosi rezervacija.

```
SELECT zahtjev_id AS ZahtjevID, tip_zahtjeva.tip AS TipZahtjeva, ime AS Ime,
prezime AS Prezime, datum_rezervacije AS DatumRezervacije, prostor.naziv AS Prostor
FROM ZAHTJEV
INNER JOIN TIP_ZAHTJEVA ON ZAHTJEV.tip_zahtjeva_id=TIP_ZAHTJEVA.tip_zahtjeva_id
INNER JOIN KORISNIK ON ZAHTJEV.korisnik_id=KORISNIK.korisnik_id
INNER JOIN STOL ON ZAHTJEV.stol_id=STOL.stol_id
INNER JOIN PROSTOR ON STOL.prostor_id=PROSTOR.prostor_id
WHERE status_id=3
```

Slika 4.4: Primjer složenog upita.

4.3 Podupiti

Podupite dijelimo u dvije osnovne skupine: jednoretčani koji vraćaju nula ili jedan redak u vanjski upit (Slika 4.5) i višeretčane koji vraćaju jedan ili više redaka u vanjski upit (Slika 4.6). Također razlikujemo tri vrste jednoretčanih i višeretčanih upita:

1. višestupičasti podupit: vraća više od jednog stupca,
2. korelirani podupit: referencira jedan ili više stupaca iz vanjskog upita,
3. ugniježđeni podupiti: podupiti unutar drugih upita.

```
SELECT naziv, ocjena FROM PROSTOR
WHERE ocjena = (SELECT MAX(ocjena) FROM PROSTOR) AND tip='Noćni klub';
```

Slika 4.5: Primjer podupita koji vraća jedan redak.

```
SELECT *
FROM STOL
WHERE STOL.stol_id not in (SELECT ZAHTJEV.stol_id from ZAHTJEV
where datum_rezervacije='2022-09-18' and status_id in (1,3));
```

Slika 4.6: Primjer podupita koji vraća više redaka.

4.4 Okidači

Okidači su aktivni elementi SQL-a. Postoje tri osnovne kategorije okidača, a to su oni koji imaju kategorizaciju:

- na razini okidača:
  - ROW Level okidač izvršava se za svaki zapis koji je ažuriran JMP izrazom,
  - STATEMENT okidač izvedba događaja poziva okidač samo jednom za cijeli skup naredbi koje se izvršavaju odjednom.
- na vremenu aktiviranja:
  - prije okidača - izvršava se prije određenog događaja koji se treba dogoditi,
  - nakon okidača - izvršava se nakon određenog događaja koji se dogodio,
  - umjesto okidača - izvršava se za svaki zapis koji je ažuriran JMP izrazom.
- događaja okidača:
  - JMP okidač izvršava se ako se izvede JMP naredba,
  - JDP okidač izvršava se ako se izvede JDP naredba,
  - Datebase okidač izvršava se ako se dogodio događaj baze podataka kao što su: shutdown, startup, logoff, logon.

4.5 Indeksi

Indeksi služe za ubrzanje pretrage po bazi podataka nad podacima koji su vrlo različiti ili vrlo slični. B-tree indeksi služe za ubrzavanje upita nad stupcima koji imaju puno različitih vrijednosti. Svaki lokal ima svoju adresu te zbog velikog broja različitih vrijednosti stavljan je indeks na adresu, slično i za prezime što je prikazano na slici 4.7.

Bitmap indeksi služe za ubrzavanje upita nad stupcima koji imaju malen broj različitih vrijednosti. Kako spol u tablici korisnik ima samo dvije moguće vrijednosti stavljan je na njega indeks. Slično i za tip lokala što je prikazano na slici 4.8.

```
create index prezime_index on korisnik(prezime);
```

Slika 4.7: Primjer indeksa.

```
create bitmap index spol_index on korisnik(spol);
```

Slika 4.8: Primjer bitmap indeksa.

4.6 Komentari

Dodavanje komentara na tablicu pobliže objašnjava svrhu tablice. U nastavku je naveden primjer na slici 4.9 i prikazana sintaksa pozivanja radi provjere komentara koji su dodani na tablicu.

```
comment on table ZAHTJEV is 'Sadrži sve zahtjeve koji su podneseni';
select * from user_tab_comments where table_name='ZAHTJEV';
```

Slika 4.9: Primjer komentara na tablicu.

Uz komentare mogu se dodati i uvjeti na tablicu: CHECK, NOT NULL, PRIMARY KEY, FOREIGN KEY, UNIQUE, CHECK OPTION i READ ONLY.

4.7 Procedure

Procedure obavljaju niz SQL naredbi te mogu vratiti skup rezultata. Rezultat ne mora vratiti direktno već rezultat vraćaju preko varijabli ili koristeći SELECT naredbu. Procedura se kreira pomoću sljedećih koraka:

1. izrada procedure pomoću ključne riječi *Create*,
2. definira se ime procedure,
3. definiraju se ulazni i izlazni parametri,
4. definira se tijelo procedure.

Naredbom CREATE procedura je kreirana i pohranjena u sklopu baze. Prilikom poziva procedure, potrebno joj je predati odgovarajuće parametre. Procedura se poziva ključnom naredbom CALL nakon čega se navodi naziv procedure te se navode vrijednosti parametara koje se proceduri predaju i popis varijabli u koje se rezultat pohranjuje. Parametar je varijabla pomoću koje procedura komunicira. Postoji više vrsti parametara:

1. ulazni parametar IN koristi se kao ulaz u proceduru. Ako procedura unutar koda mijenja vrijednost ove varijable, njegova izvorna vrijednost ostaje nepromijenjena nakon završetka pohranjene procedure. Drugim riječima, pohranjena procedura radi samo na kopiji parametra IN,

2. izlazni parametar OUT koristi se za vraćanje vrijednosti iz procedure, njegova vrijednost može se promijeniti unutar pohranjene procedure i njegova se nova vrijednost prosljeđuje natrag u program koji poziva proceduru,

3. ulazno – izlazni parametar IN OUT koristi se kao ulaz i izlaz iz procedure. Preko istog parametra se proceduri daje podatak te procedura u njega zapisuje i vraća rezultat. Ova vrsta parametra koristi se kada se radi o istom tipu ulaznog i izlaznog podatka. [6]

Na slikama 4.10 i 4.11 prikazani su primjeri korištenih procedura. Procedura Prijava radi na principu da vraća vrijednost 1 ako je uneseno ispravno korisničko ime i lozinka i vrijednost 0 ako nije, dok procedura za dohvaćanje stolova radi na principu da dohvati sve slobodne stolova na željeni datum i u željenom mjestu te prostoru.

```
ALTER PROCEDURE [dbo].[Prijava]
@korisnicko_ime varchar(30),
@lozinka varchar(20)
AS

BEGIN
declare @provjera INT
declare @povratnaVrijednost int =0
    SET NOCOUNT ON;

    set @provjera= (SELECT count(*) from  Korisnik where username=@korisnicko_ime and lozinka=@lozinka);
    set @povratnaVrijednost=1;
    if @provjera > 0
    begin
    return @povratnaVrijednost
    end
    else
    return 0
END
```

Slika 4.10: Procedura Prijava.

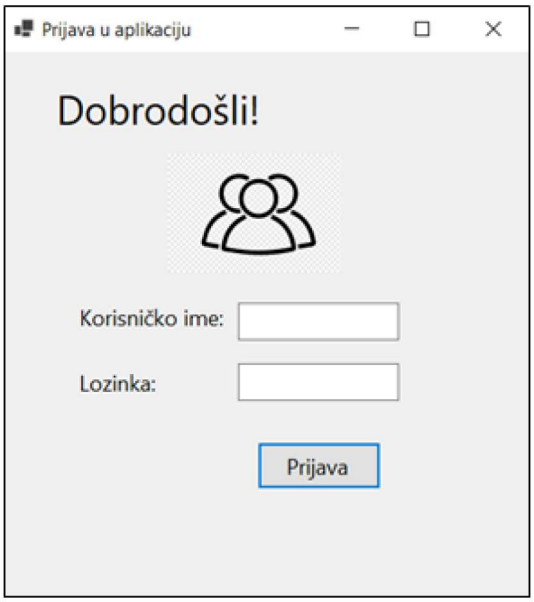
```
CREATE PROCEDURE [dbo].[Dohvati_slobodne_stolove]
@prostor VARCHAR(30),
@br_osoba INT,
@datum DATE,
@CurrencyCursor CURSOR VARYING OUTPUT
AS
BEGIN
SET NOCOUNT ON;
SET @CurrencyCursor = CURSOR FORWARD_ONLY STATIC FOR
SELECT broj_stola FROM STOL
LEFT JOIN PROSTOR on PROSTOR.prostor_id = STOL.PROSTOR_id
LEFT JOIN ZAHTJEV on Stol.stol_id = ZAHTJEV.STOL_id
WHERE PROSTOR.prostor_id = (select prostor_id from PROSTOR where naziv = @prostor) and STOL.broj_osoba >= @br_osoba
AND  STOL.stol_id not in (select ZAHTJEV.stol_id from ZAHTJEV where datum_rezervacije=@datum and status_id in (1,3));
OPEN @CurrencyCursor;
END
GO
```

Slika 4.11: Procedura za dohvaćanje slobodnih stolova.

5 | Desktop aplikacija za testiranje baze podataka

5.1 Forma za prijavu korisnika

Pri pokretanju desktop aplikacije otvara se forma za prijavu (Slika 5.1), gdje korisnik upisuje svoje podatke. Klikom na dugme *Prijava*, ovisno o ulozi korisnika, otvara se pripadajući zaslon.



Slika 5.1: Forma za prijavu.

Na klik gumba *Prijava* pokreće se procedura prikazana na slici 4.10. Poziv procedure kroz *Visual Studio* izgleda kao na slici 5.2 gdje je prikazana provjera unesenih polja u formi. Ako su polja dobro popunjena, pozvat će se procedura *Prijava* iz *Microsoft SQL Server Managment Studija*. U suprotnom pojavit će se pripadajuća poruka.

```
public bool Provjera(string korisnickoIme, string lozinka) {

    if (korisnickoIme == "" || lozinka == "")
    {
        MessageBox.Show("Molim Vas unesite korisničko ime i lozinku! ");
        return false;
    }
    else {
        SqlCommand cmd = new SqlCommand(@"dbo.[Prijava]", connection);
        SqlParameter unesenoKorisnickoIme = new SqlParameter();
        unesenoKorisnickoIme.ParameterName = "@korisnicko_ime";
        unesenoKorisnickoIme.SqlDbType = SqlDbType.VarChar;
        unesenoKorisnickoIme.Value = korisnickoIme;

        SqlParameter unesenaLozinka = new SqlParameter();
        unesenaLozinka.ParameterName = "@lozinka";
        unesenaLozinka.SqlDbType = SqlDbType.VarChar;
        unesenaLozinka.Value = lozinka;

        cmd.Parameters.Add(unesenoKorisnickoIme);
        cmd.Parameters.Add(unesenaLozinka);
        cmd.CommandType = CommandType.StoredProcedure;

        var returnParameter = cmd.Parameters.Add("@ReturnValue", SqlDbType.VarChar);
        returnParameter.Direction = ParameterDirection.ReturnValue;

        connection.Open();
        cmd.ExecuteNonQuery();
        var result = returnParameter.Value;
        connection.Close();
        if (result.Equals(1))
            return true;
        else return false;
    }

}
```

Slika 5.2: Poziv procedure *Prijava*.

U nastavku će ukratko biti objašnjen postupak poziva procedure iz Visual Studija:

1. Prije poziva procedure, potrebno je osigurati konekciju na željenu bazu podataka što je postignuto pomoću klase *SqlConnection* kao što je prikazano na slici 5.4.

```
private SqlConnection connection =
    new SqlConnection("Data Source=DESKTOP-JK7O8IG;Initial Catalog=rezeracijeStolova;Integrated Security=True");
```

Slika 5.3: Konekcija na bazu podataka.

2. SqlCommand je klasa koja omogućava izvršavanje SQL naredbe ili procedure preko SQL Servera pomoću unaprijed napravljene konekcije na bazu podataka (korak 1.),

3. dodjela parametara se izvršava pomoću klase SqlParameter gdje se pridaju atributi kao što su:

   (a) ParameterName – ime parametra,

(b) SqlDbType – tip parametara,

(c) Value – vrijednost parametra.

4. Nakon definiranja parametara, parametri se dodjeljuju SqlCommandi i određuje se tip komande.

5. Za proceduru je potrebno odrediti i povratni parametar u slučaju da ga ima i kojeg tipa će biti. Pomoću SqlCommand atributa Direction, određuje se je li parametar ulazni ili povratni. Ako se ne odredi, onda je unaprijed definiran kao ulazni parametar i zato nije bilo potrebe da se dodatno postavlja njegova vrijednost kod ulaznih parametara.

6. Konekcija na bazu podataka se otvara pomoću naredbe *Open*(), a izvršava se pomoću naredbe *ExecuteNonQuery*().

7. Na kraju se konekcija na bazu zatvara pomoću naredbe*Close*().

Forma za prijavu je testirana na način prikazan na slici 5.4 gdje je prikazan ishod svakog testiranja. Testiranje je provedeno uspješno i njime je potvrđen ispravan rad procedure.

KOMBINACIJE TESTIRANJA	ISHOD TESTIRANJA
Ispravno korisničko ime i lozinka	Uspješna prijava u aplikaciju.
Neispravno korisničko ime i ispravna lozinka	Neuspješna prijava u aplikaciju.
Ispravno korisničko ime i neispravna lozinka	Neuspješna prijava u aplikaciju.
Neispravno korisničko ime i lozinka	Neuspješna prijava u aplikaciju.

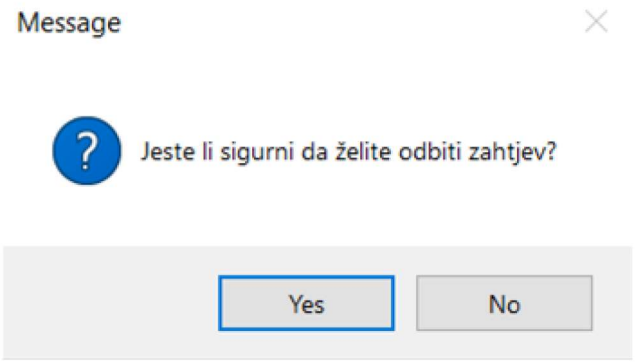
Slika 5.4: Testiranje procedure *Prijava*.

5.2 Forma za administratora

Ako se korisnik ulogira kao administrator otvorit će mu se zaslon prikazan na slici 5.5. Na zaslonu su podnesene rezervacije koje čekaju da budu odobrene od strane administratora. Prikaz je postignut pomoću upita prikazanog na slici 4.3 koji je dohvatio sve potrebne podatke i popunio tablicu.

Slika 5.5: Prikaz forme za administratora.

Nakon što administrator odobri ili odbije zahtjev, vrši se *update* tablice gdje se mijenja tip zahtjeva u rezervirano ili odbijeno te taj redak nestaje iz prikazane tablice na formi. Također, kako bi se izbjeglo slučajno odobravanje ili odbijanje zahtjeva, napravljena je iskočna poruka prikazana na slici 5.6 gdje korisnik mora potvrditi da je siguran u svoj odabir i tek onda se izvršava SQL naredba za ažuriranje zapisa u tablici *Zahtjev*.



Slika 5.6: Prikaz poruke koja se prikazuje nakon klika na gumb *Odbij*

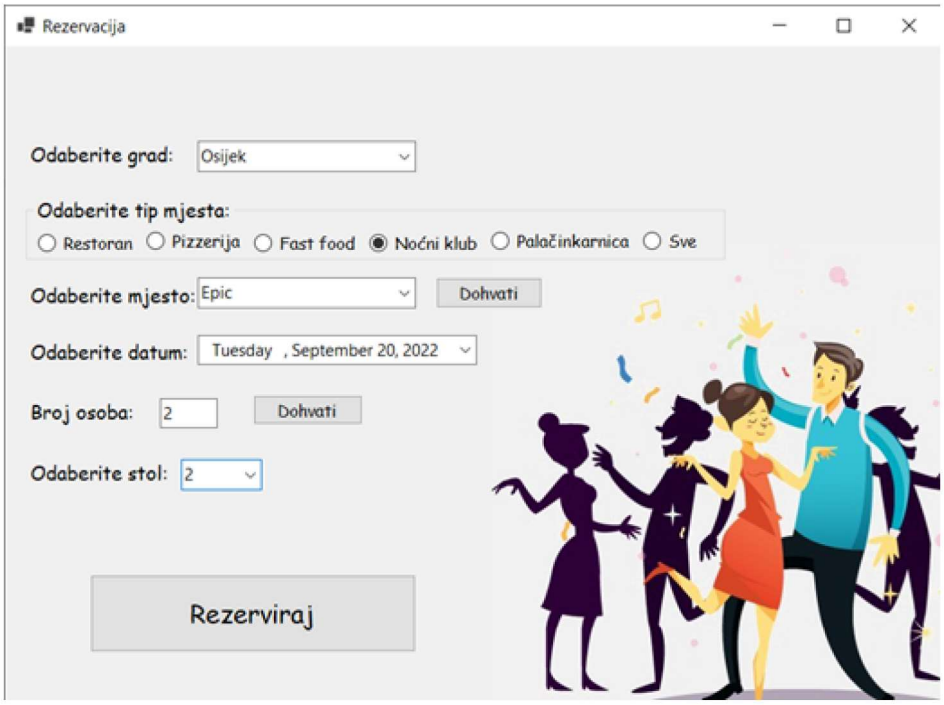
Testiranje ovog dijela je provedeno na način prikazan na slici 5.7 gdje su vidljivi i ishodi testiranja.

KOMBINACIJA TESTIRANJA	ISHOD TESTIRANJA
Jesu li dohvaćeni ispravni zahtjevi?	DA
Nakon odobravanja/odbijanja zahtjeva prikazana je pripadajuća poruka?	DA
Nakon odobravanja/odbijanja zahtjeva, promijenjen je tip zahtjeva?	DA

Slika 5.7: Testiranje forme za administratora

5.3 Forma za podnošenje rezervacija

Forma za podnošenje rezervacije otvara se ako je korisnik klijent (Slika 5.8).



Slika 5.8: Prikaz forme za ispunjavanje rezervacije.

Postupak podnošenja rezervacije je sljedeći:

1. Prvo se odabire grad i tip mjesta te se klikom na gumb *Dohvati* kod odabira mjesta, dohvaćaju sva mjesta koja odgovaraju odabranim parametrima. U slučaju da se odaberu svi tipovi mjesta izvršit će se upit koji nema zadan tip, a u suprotnom onaj u kojem je zadan tip (Slika 5.9).

```
if (Sve.Checked)
{
    sc = new SqlCommand("select naziv, prostor_id from prostor where " +
   "grad_id=(select grad_id from grad where naziv=@grad); ", cnn);
}
else
{
    sc = new SqlCommand("select prostor.naziv, prostor_id from prostor where tip=@tip and " +
        "grad_id=(select grad_id from grad where naziv=@grad); ", cnn);
}
```

Slika 5.9: Prikaz odabira.

2. Nakon što se dohvate sva mjesta, odabire se mjesto, datum željene rezervacije i broj osoba. Zatim se klikom na *Dohvati* kod odabira stola, dohvaćaju svi slobodni stolovi pomoću procedure sa slike 4.11.

3. Zadnji korak je odabir stola i pritisak na gumb *Rezerviraj* gdje se poziva metoda sa slike 5.10 u kojoj se provjerava može li se izvršiti rezervacija te ovisno o tome se dobije pripadajuća poruka.

```
1 reference
private void podnesi_rez_Click(object sender, EventArgs e)
{
    string odabrani_grad = gradovi.Text;
    string odabrani_tip = provjeraRB();
    string odabrano_mjesto = mjesto.Text;
    int br_osoba = Int32.Parse(brojosoba.Text);
    int br_stola = Int32.Parse(stolovi.Text);
    DateTime datum = DateTime.Now ;
    string rez_datum = datum_r.Value.ToString("yyyy-MM-dd");
    cnn.Open();

    SqlCommand sc;
    if (odabrani_grad != null && br_osoba > 0 && odabrani_tip != null && br_stola !=0)
    {
          sc = new SqlCommand("insert into zahtjev (tip_zahtjeva_id, korisnik_id, stol_id, status_id, datum_rezervacije, datum_podnosenja, broj_osoba) " +
        " values(1, select korisnik_id from korisnik where username=@korisnicko_ime, select stol_id from stol where broj_stola=@br_stol and" +
        " prostor_id=(select prostor_id from prostor where naziv=@mjesto), " +
        "3, @datum_rez , @datum, @br_osoba); ", cnn);

       sc.Parameters.Add(new SqlParameter("@tip", odabrani_tip));
       sc.Parameters.Add(new SqlParameter("@grad", odabrani_grad));
       sc.Parameters.Add(new SqlParameter("@mjesto", odabrano_mjesto));
       sc.Parameters.Add(new SqlParameter("@datum_rez", rez_datum));
       sc.Parameters.Add(new SqlParameter("@datum", datum));

       cnn.Close();
       MessageBox.Show("Uspjeh!");
    }
    else
    {
       MessageBox.Show("Provjerite jeste li odabrali grad, tip mjesta, mjesto i broj osoba prije nego odaberete dohvat stola!");
    }
}
```

Slika 5.10: Prikaz metode za podnošenje rezervacije.

Testiranje procedura je provedeno unosom raznih opcija kako bi se obuhvatilo što je moguće više kombinacija. Testiranje je provedeno uspješno.

6 | Zaključak

U današnje vrijeme zahvaljujući brzom napretka tehnologije napravljene su brojne aplikacije koje nam pomažu pri svakodnevnim obavezama kako osobnim tako i poslovnim. Baze podataka su iznimno korisne i omogućavaju nam mnogo više od jednostavnih matematičkih operacija. Danas se one koriste u školama, tvrkama, trgovinama i slično. Trenutno se rezervacije najčešće zapisuju ručno i potrebno je svaki put provjeravati koliko ima rezervacija i ima li slobodnih mjesta, koji stolovi su slobodni i slično. Navedeni postupak rezervacije oduzima dosta vremena i lako može doći do pogrešne procjene slobodnih stolova što rezultira nezadovoljstvom gostiju. U tu svrhu modelirana je baza podataka, najprije koristeći model entiteta-veza. Napravljeni model zatim je pretvoren u relacijski model, te je nakon toga baza implementirana koristeći SQL. Kako bi se provjerila uspješnost napravljene baze podataka, napravljena je jednostavna desktop aplikacija koja omogućava jednostavno testiranje putem grafičkog sučelja. Na ovaj način bazu podataka može testirati i osoba koja nije upoznata sa SQL-om i samim bazama podataka. Također, izradom aplikacije dobio se uvid kako je moguće dodatno razviti aplikaciju kako bi ona obuhvatila još veći spektar opcija koje bi olakšale korisnicima korištenje takve aplikacije.

Literatura

[1] Tonči Carić i Mario Buntić, *Uvod u relacijske baze podataka, Zagreb, 2015.*

[2] Robert Manger, *Baze podataka, 2011.*

[3] Mile Pavlić, *Oblikovanje baza podataka, Rijeka, 2011.*

[4] T. Teorey, S. Lightstone, T. Nadeau, *Database modeling and design: logical design, Četvrto iz. Morgan Kaufmann, 2006.*

[5] Web izvor dostupan na `https://careerfoundry.com/en/blog/data-analytics`. (*Careerfoundry*).

[6] Web izvor dostupan na `https://www.mysqltutorial.org/stored-procedures-parameters`. (*mysqltutorial*).

[7] Web izvor dostupan na `https://sr.wikipedia.org/sr-el/Referencijalni_integritet`. (*Wikipedia*).

Sažetak

Baza podataka je organizirani skup informacija na nekom računalnom sustavu koji je lako dostupan korisnicima i aplikacijama. U ovom završnom radu je opisana njezina kratka povijest od 1960-ih godina kada su započele automatizacije baza podataka pa sve do pojave NoSQL-a. Zatim je prikazan razvoj baza podataka koji se sastoji od tri faze: konceptualne, logičke i fizičke baze. Svaka od faza je zasebno objašnjena i dani su primjeri. Detaljno i uz primjere objašnjeni su i elementi baze podataka kao što su tablice, upiti i procedure. U zadnjem dijelu je opisano testiranje baze podataka pomoću napravljene desktop aplikacije koja je osmišljena tako da se korisnik ulogira u aplikaciju i ovisno o njegovoj ulozi otvara se pripadna forma.

Ključne riječi

baze podataka, MEV, SQL, entitet, atribut, relacijski model, MS SQL Server, tablice, upiti, podupiti, okidači, indeksi, procedure

DATABASE FOR RESERVATIONS IN LOCALS

Summary

A database is an organized set of information on a computer system that is easily accessible to users and applications. This final paper describes its short history from the 1960s, when database automation began, until the emergence of NoSQL. Then the development of databases is presented, which consists of three phases: conceptual, logical and physical database. Each of the phases is explained separately and examples are given. Database elements such as tables, queries and procedures are explained in detail and with examples. The last part describes the testing of the database using the created desktop application, which is designed so that the user logs into the application and, depending on his role, the corresponding form is opened.

Keywords

databases, MEV, SQL, entity, attribute, relational model, MS SQL Server, tables, queries, subqueries, triggers, indexes, procedures

Životopis

Rođena sam u Osijeku, 08.04.1999. godine. Prva četiri razreda osnovne škole pohađala sam u osnovnoj školi Mladost. Nakon toga selim u Bilje gdje završavam svoje osnovnoškolsko obrazovanje u OŠ Bilje te upisujem III. Gimnaziju u Osijeku. Nakon završetka srednje škole 2018. godine upisujem prijediplomski studij matematike, smjer *Matematika*, na Sveučilištu J. J. Strossmayera. Završila sam program Dragovoljnog vojnog osposobljavanja u Požegi kao 41. naraštaj u 2024. godini.