Konceptualno oblikovanje baze podataka

Bojan, Kavur

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Bojan Kavur

Konceptualno oblikovanje baze podataka

ZAVRŠNI RAD

Varaždin, 2019.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Bojan Kavur

Matični broj: 43256/14–R

Studij: Informacijski sustavi

Konceptualno oblikovanje baze podataka

ZAVRŠNI RAD

Mentor:

Prof. dr. sc. Mirko Maleković

Varaždin, kolovoz 2019.

Bojan Kavur

Izjava o izvornosti

Izjavljujem da je moj završni/diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Tema ovog završnog rada je „Konceptualno oblikovanje baze podataka“. Kao što i sam naslov govori, u ovom završnom radu bit će opisani elementi konceptualnog oblikovanja baze podataka, što je koncept, konceptualna shema, normalne forme te ću kao primjer napraviti transformaciju konceptualnog modela u pripadnu shemu relacijske baze podataka. Tada ću prethodno dobivenu shemu relacijske baze podataka verificirati i poboljšati primjenom normalizacije (BCNF). Navedeni primjer bit će implementiran pomoću alata: SQL Server 2017 te SQL Server Management Studio (SSMS).

Normalizacija je sastavni dio logičkog oblikovanja baze podataka, a s obzirom na to da su za normalizaciju potrebne normalne forme (NF), objasnit ću i logičko oblikovanje baze podataka. Znači, koje zavisnosti postoje i kako se nepoželjne zavisnosti mogu otkloniti te koji se problemi javljaju u relacijskim bazama podataka i kako se ti problemi mogu otkloniti pomoću normalizacije.

Ključne riječi: sql, konceptualni model, ER model, transformacija konceptualnog modela u shemu relacijske baze podataka, implementacija, shema relacijske baze podataka, normalne forme, normalizacija

Sadržaj

1. Uvod

Postoji konceptualno, logičko i fizičko oblikovanje baze podataka. S obzirom na to da je tema ovog rada „Konceptualno oblikovanje baze podataka“ naglasak će biti na konceptualnom oblikovanju baze podataka, dok ću logičko oblikovanje baze podataka samo spomenuti kako bi znali koja je razlika između njih. Konceptualno oblikovanje ili modeliranje baze podataka temelji se na konceptima i vezama između koncepata. Navedene koncepte i veze između njih možemo uočiti u aplikacijskoj domeni, stoga možemo reći da je primarna zadaća konceptualnog oblikovanja baze podataka opis aplikacijske domene. Aplikacijska domena je zapravo dio realnog svijeta, ili stvarnosti, koji želimo detaljnije promotriti. Kao rezultat konceptualnog oblikovanja dobivamo konceptualnu shemu baze koja se sastoji od entiteta i veza (asocijacija) između njih. [1, str. 204], [2, str. 8]

S druge strane logičko oblikovanje baze podataka kao primarnu zadaću ima razmatranje zavisnosti među atributima relacijske sheme, a te zavisnosti upravo predstavljaju uvjete integriteta. Relacijska shema je skup atributa kojim opisujemo obilježja objekata koja nas zanimaju iz aplikacijske domene. Neke od zavisnosti koje se nalaze u relacijskoj shemi mogu biti nepoželjne i stoga je potrebna normalizacija baze podataka. Kada govorimo o zavisnostima mislimo na funkcijske zavisnosti koje su temelj za prvu normalnu formu (1NF), drugu normalnu formu (2NF), treću normalnu formu (3NF) i Boyce – Codd-ovu normalnu formu (BCNF). Postoje još četvrta normalna forma (4NF), peta normalna forma (5NF) i šesta normalna forma (6NF) koje se temelje na višeznačnim zavisnostima i zavisnostima spoja. Rezultat logičkog oblikovanja je logička shema koja se sastoji od relacija ili tablica (entiteta), veza i atributa (stupaca u tablici). [1, str. 167], [2, str. 8]

Na kraju ću napraviti transformaciju konceptualnog modela u pripadnu shemu relacijske baze podataka. Time ću dobiti fizičku shemu što se odnosi na fizičko oblikovanje baze podataka. To je niz SQL naredbi kojima u sustavu za upravljanje bazom podataka kreiramo tablice i odgovarajuća ograničenja iz logičke sheme. [2, str. 8]

U daljem sadržaju ovog rada ću detaljnije opisati sve navedene normalne forme kao i sam postupak normalizacije te naravno konceptualno oblikovanje baze podataka. Ova tema je vrlo zanimljiva jer nam govori kako modelirati dobru bazu podataka ili kako neku postojeću poboljšati. Samo poboljšanje i normalizacija baze podataka znatno utječu na performanse kao što su na primjer brzina izvođenja upita, brži pristup podacima, smanjenje zauzeća prostora i lakše održavanje. Sve navedeno je vrlo bitno kako bi kasnije neki sustav (aplikacija) bio sigurniji te bi mogao brže raditi.

2. Metode i tehnike rada

Pri razradi teme koristit ću slijedeće alate, tj. programe:

- **Microsoft Word 2016** – alat ili program koji ću koristiti kako bi napisao ovaj rad, a samim time i oblikovao tekst, slike i sve ostalo što se može naći u ovom dokumentu.

- **SQL Server 2017** – program čija je glavna funkcija upravljanje podacima koje zahtijevaju drugi programi (u ovom slučaju SQL Server Management Studio 17).

- **SQL Server Management Studio 17 (SSMS)** – program koji služi za administraciju i upravljanje komponentama zajedno sa SQL Server-om. Ovaj alat sadrži sve što je potrebno kako bi se modelirala baza podataka. Sve od izrade nove baze podataka i tablica pa do unosa podataka u bazu, pisanja upita i izrade ERA modela baze podataka.

- **Mendeley desktop** – program koji ću koristiti za pravilno navođenje literature, citiranje i sve što je potrebno vezano uz literaturu rada.

- **Programski jezik SQL** – SQL je kratica od „Structured Query Langugage“ i taj jezik nam služi za izvršavanje SQL naredbi (npr. traženje, ažuriranje, izrada i brisanje podataka) pomoću kojih upravljamo relacijskim bazama podataka.

- **Mozilla Firefox** – web preglednik koji ću koristiti za pristup online literaturi i svim ostalim online dokumentima koje ću koristiti za pomoć prilikom pisanja ovog rada.

- **MySQL Workbench** – program sličan SQL Server Management Studio-u, samo što on koristi drugi sustav za upravljanje bazom podataka. Riječ je o sustavu MySQL, dok SQL Server Management Studio koristi SQL Server kao što sam i ranije naveo.

- **Visual Paradigm Online** – online alat koji služi za izradu raznih dijagrama i grafova. U ovom radu koristit ću ga za izradu konceptualnog modela, tj. ER dijagrama.

3. Konceptualno oblikovanje baze podataka

Konceptualno oblikovanje baze podataka je vodeća komponenta logičkog oblikovanja baze podataka. Naime, kao što sam spomenuo u uvodnom poglavlju, konceptualno oblikovanje baze podataka temelji se na konceptima i vezama između koncepata. Glavna zadaća konceptualnog oblikovanja je opis aplikacijske domene u kojoj uočavamo koncepte i njihove veze. Kao rezultat konceptualnog oblikovanja dobivamo konceptualnu mrežu ili shemu, koja tada predstavlja konceptualni model aplikacijske domene. Upravo se zbog toga može koristiti i naziv konceptualno modeliranje baze podataka koji ste mogli primijetiti u uvodu. Negdje se može naći i naziv semantičko modeliranje baze podataka. [1, str. 204], [3, str. 8 - 9]

Kod konceptualnog oblikovanja najvažniji je naglasak na čitljivosti i jednostavnosti modela. S obzirom na to da je konceptualno oblikovanje prvi korak u modeliranju baze podataka vrlo je važno da taj model bude čitljiv i jednostavan kako bi ga i klijent mogao razumjeti jer nam služi kao sredstvo za komuniciranje s klijentom. Konceptualno oblikovanje baze podataka započinje nakon što dobijemo specifikaciju zahtjeva od klijenta i nastavlja se paralelno sa prikupljanjem daljnjih podataka od klijenta, zbog toga što inicijalni zahtjevi gotovo nikada nisu dovoljno opsežni. Isto tako možemo klijentu dati neke vlastite prijedloge kako bi nešto izradili na bolji način i ukoliko mu to odgovara tada će se promijeniti inicijalni zahtjevi. Vrlo je važno da se napravi dobar model jer kasnije najmanje promjene na modelu mogu rezultirati velikim promjenama na cjelovitom sustavu. Na primjer, možemo imati neku manju promjenu na modelu koju će biti lako napraviti u sustavu za upravljanje bazom podataka, ali u ostatku sustava (aplikaciji) trebat će se promijeniti dizajn, omogućiti unos i prikaz novih podatka, itd. Za što će trebati puno vremena i rezultirat će velikim troškovima. [4, str. 10 - 13, 17 i 273 - 275]

Za konceptualno modeliranje koristimo razne specifikacijske jezike, od kojih je za ovaj rad najzanimljivija grupa grafičkih jezika. Postoje još i grupa formalnih jezika i grupa hibridnih jezika (kombinacije prethodne dvije grupe). U grupu grafičkih jezika spadaju UML (unified modeling language, u prijevodu unificirani jezik za modeliranje), ER (entity-relationship, u prijevodu entiteti-veze). [1, str. 207 - 208]

U idućem ću poglavlju objasniti što je koncept, detaljnije ću objasniti ER model, temeljne i napredne ER konstrukte te ih uz pomoć primjera pojasniti.

3.1. Koncept

„Koncept je ideja (pojam) koji primjenjujemo na stvari ili objekte realnosti.“ [1, str. 204]

Kao što sam rekao, aplikacijska domena je dio realnog svijeta ili stvarnosti koji želimo detaljnije promotriti, a čine ju uočeni objekti realnosti. Dakle, možemo reći da pomoću koncepata strukturiramo aplikacijsku domenu. Koncepti ili tipovi su nam potrebni kako bi mogli percipirati stvarnost jer bez koncepata to nije moguće. Koncepte je na kraju potrebno povezati kako bi dobili potpunu sliku realnosti, za što su nam potrebne veze (asocijacije) koje ću detaljnije objasniti u poglavlju o ER modelu. [1, str. 204 - 205], [5]

Definicija koncepta glasi K = (simbol, intenzija, ekstenzija). Što znači da se koncept sastoji od tri komponente: simbola (oznake ili naziva koncepta), intenzije (definicije koncepta) i ekstenzije (skupa primjera za koncept). [1, str. 205], [5]

Budući da sada znamo definiciju koncepta, možemo ju detaljnije promotriti na primjeru u kojem ćemo opisati koncept vozila. Simbol: vozilo, intenzija: prijevozno sredstvo, a ekstenzija: cestovno vozilo, željezničko vozilo, putničko vozilo, …

3.1.1. Parcijalni koncept

Ukoliko imamo parcijalno znanje o nekoj aplikacijskoj domeni, znači da ne znamo jednu od komponenata, tada koristimo parcijalne koncepte. Parcijalni koncept (Kp) je koncept za koji nisu navedene sve tri komponente. Tako možemo za koncept K = (simbol, intenzija, ekstenzija) imati tri slučaja: [1, str. 206], [5]

1) Koncept bez simbola Kp = (Ø, intenzija, ekstenzija)
2) Koncept bez intenzije Kp = (simbol, Ø, ekstenzija)
3) Koncept bez ekstenzije Kp = (simbol, intenzija, Ø)

3.1.2. Sinonimni koncept

Oznaka za sinonimni koncept je Ks i javlja se u slučaju kada za neki koncept K postoje različiti nazivi. Dakle, definicija za sinonimni koncept glasi Ks = ((s1, s2), intenzija, ekstenzija), gdje je $s1 \neq s2$. Znači da postoje koncepti koji imaju različiti simbol ili naziv i isto značenje. [1, str. 206 - 207], [5]

Kao primjer možemo uzeti koncepte glazba i muzika. Oba pojma imaju isto značenje, ali različit naziv, tj. simbol.

3.1.3. Homonimni koncept

Homonimni koncept Kh javlja se u slučaju kada za neki koncept K postoji više različitih značenja. Definicija homonimnog koncepta glasi Kh = (simbol, (i1, i2), (e1, e2)), gdje je i1 ≠ i2 i e1 ≠ e2. Prema tome postoje koncepti koji imaju isti simbol ili naziv i različito značenje. [1, str. 206 - 207], [5]

Za primjer homonima možemo uzeti list (list papira), list (dio biljke) ili bor (kemijski element), bor (stablo).

U konceptualnom modeliranju potrebno je obratiti pažnju na sinonimne i homonimne koncepte jer ih treba pomno razmotriti i kako bi ostvarili jasniju semantiku koncepata, tj. ukoliko je moguće, izbjeći korištenje sinonimnih i homonimnih koncepata. [1, str. 207]

3.2. ER model

ER (entity-relationship, u prijevodu entiteti-veze) je jedan on grafičkih jezika koji koristimo za konceptualno modeliranje i kojeg ću u ovom poglavlju detaljnije objasniti. Krenimo prvo s temeljnim ER konstruktima.

3.2.1. Temeljni ER konstrukti

Kako bi mogli razumjeti ER model, prvo se moramo upoznati s terminologijom i temeljnim ER konstruktima koji se nalaze u samom ER modelu. Tri su osnovne klase objekata koje se nalaze u svakom ER modelu, a to su: entiteti, atributi i veze. Budući da smo se sada upoznali s osnovnom terminologijom, krenimo na razumijevanje tih pojmova uz detaljan opis svakog i primjere. [3, str. 13], [4, str. 65 - 66 i 75]

3.2.2. Entiteti

Entiteti su glavni elementi o kojima prikupljamo informacije i želimo te informacije sačuvati. Kao primjer entiteta možemo uzeti osobu, mjesto, stvar, događaj ili bilo koju stvar koja nam je od informacijskog interesa i o kojoj želimo čuvati podatke. Određenu pojavu nekog entiteta nazivamo instancom tog entiteta. U ER modelu entitete reprezentiramo kao pravokutnike unutar kojeg navedemo ime ili naziv entiteta, dok se u logičkom i fizičkom modelu koriste tablice za njihov prikaz. [3, str. 13 - 14], [4, str. 75 - 77]

Naziv entiteta mora biti u jednini i referencira se na jednu instancu entiteta (u smislu da predstavlja jedan red ili zapis u tablici). Npr. Student umjesto Studenti ili Proizvod umjesto Katalog Proizvoda. [4, str 78 -79]

Entitete dijelimo na jake i slabe entitete. Svaki entitet ima identifikator koji jednoznačno određuje postojanje pojedine instance entiteta, što vrijedi za jake entitete. Slabi entitet (dijete) nasljeđuje identifikator od pripadnog jakog entiteta kojeg nazivamo roditelj. Kako bi ovo mogli jasnije razumjeti, potrebno je znati što je atribut. Tako da ću u idućem poglavlju o atributima, nakon što shvatimo koncept atributa, detaljnije pojasniti jake i slabe entitete. [3, str. 16, 19 - 22]

Na slijedećim slikama možemo vidjeti primjer entiteta izrađen u alatima MySQL Workbench (Slika 1) i MS SQL Server Management Studio 17 (Slika 2) te primijetiti da se entiteti implementiraju kao tablice.

Slika 1: Entitet u MySQL Workbench-u [vlastita izrada]

Slika 2: Entitet u SSMS-u [vlastita izrada]

3.2.3. Atributi

Nakon što smo shvatili entitete, slijede nam atributi. Atributi predstavljaju karakteristike entiteta koje nam daju njihov detaljniji opis. Zapravo predstavljaju odgovor na pitanje „Koje podatke o entitetu želimo čuvati?“. Podsjetimo se da o entitetima prikupljamo informacije i upravo nam atributi služe kako bi te informacije sačuvali kao karakteristike entiteta. Određenu pojavu nekog atributa unutar entiteta ili veze nazivamo vrijednost atributa. Generalno se implementiraju kao stupci u tablicama i obično se ne prikazuju na konceptualnom modelu, tj. dijagramu. [3, str. 15 - 16], [4, str. 104 - 105]

Razlikujemo dvije vrste atributa: identifikatori i deskriptori. Identifikator ili ključ (ključni atribut) se koristi kako bi jednoznačno identificirali instancu entiteta, a u tablici ga zovemo i primarni ključ. Sve vrijednosti koje upisujemo u taj stupac moraju biti jedinstvene kako bismo mogli jednoznačno identificirati svaki redak ili zapis u tablici. Primarni ključ može biti jednostavan ili složen. Ukoliko se sastoji od jednog atributa (stupaca) tada je jednostavan, a ako se sastoji od više atributa (stupaca) kažemo da je složen. Deskriptori ili neključni atributi koriste se kako bismo odredili ostale karakteristike određene instance entiteta, koje mogu biti zajedničke nekim instancama entiteta s obzirom na to da ne moraju biti jedinstvene vrijednosti.

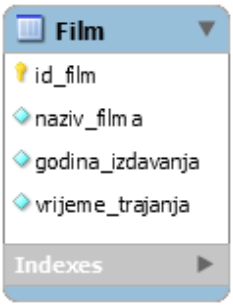
Ako se vratimo na entitet „Film“ tada bi primjer identifikatora ili ključnog atributa bio id_film, a primjer deskriptora ili neključnog atributa naziv_filma, godina_izdavanja ili vrijeme_trajanja. [3, str. 15 - 16], [6, str. 96]

Neki atributi, kao adresa, mogu biti složeni, što znači da se mogu podijeliti na više jednostavnih atributa. Adresu možemo podijeliti na ulicu, kućni broj, grad i poštanski broj. Ponekad nam ovakva podjela može biti korisna i tada svaki atribut zapisujemo zasebno, dok nam s druge strane nije potrebna i u tom slučaju ne radimo podjelu na jednostavne atribute. Drugi primjer može biti slobodna aktivnost neke osobe ili zaposlenika. Ukoliko jedna osoba ima više slobodnih aktivnosti tada bi bilo dobro da te aktivnosti podijelimo, što ćemo vidjeti kasnije da za logičku shemu moramo raditi podjelu zbog normalnih formi. [3, str. 15]

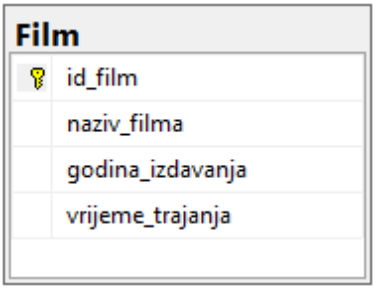
Vrlo je važno da dobro promislimo prilikom određivanja atributa. Ukoliko nam se pojavi atribut koji može imati više vrijednosti tada bismo ga trebali klasificirati kao entitet. Ako više od jedne vrijednosti neključnog atributa odgovara jednoj vrijednosti ključnog atributa, tada se neključni atribut klasificira kao entitet umjesto atributa. Kao primjer možemo uzeti knjige i njima pripadne kategorije. Jednoj knjizi može pripadati više kategorija i tada atribut kategorija može imati više od jedne vrijednosti. U tom slučaju više od jedne vrijednosti atributa kategorija odgovara jednoj knjizi i stoga bi trebali kategoriju knjige klasificirati kao zaseban entitet. [3, str. 57]

S obzirom na to da sada razumijemo atribute, vratimo se na jake i slabe entitete. Dakle, jaki entitet stoji sam za sebe i posjeduje identifikator koji jednoznačno određuje postojanje pojedine instance entiteta. S druge strane slabi entitet (dijete) taj identifikator nasljeđuje od pripadnog jakog entiteta (roditelja). Svaki entitet ili tablica mora imati primarni ključ koji je zapravo taj identifikator. Naime, slabi entitet isto tako posjeduje primarni ključ, ali postojanje slabog entiteta ovisi o njegovom roditelju. Bez roditelja taj slabi entitet ne može postojati, što nije slučaj s jakim entitetom. Jaki entitet može postojati sam za sebe, neovisno o tome posjeduje li dijete ili ne. Ovdje je zapravo riječ o vezi jedan naprema više i vezi više naprema više koje ću objasniti u idućem poglavlju o vezama u ER modelu.

Iako sam spomenuo da se atributi obično ne prikazuju na dijagramu, ja ću ih u ovom radu prikazivati. Najčešće se ne prikazuju zbog opsežnosti modela jer su sustavi za koje se rade veliki i dijagram bi tada zauzimao previše prostora te bi bio nepregledan. Za potrebe ovog rada model neće biti opsežan i iz tog razloga ću prikazati atribute. Atributi se nalaze u pravokutniku ispod naziva entiteta, što možete vidjeti na slikama 3 i 4.



Slika 3: Entitet u MySQL Workbench-u [vlastita izrada]



Slika 4: Entitet u SSMS-u [vlastita izrada]

Na slikama 3 i 4 možemo vidjeti da su entitetu film pridruženi atributi, tj. njegove karakteristike koje ga detaljnije opisuju. Kao što sam i rekao, atributi se implementiraju kao stupci u tablici, a prilikom kreiranja stupca potrebno je specificirati osim naziva i tip podataka. Postoje numerički tipovi kao što su integer, decimal; znakovni tipovi poput char(n), varchar(n); datum i vrijeme; boolean i ostali. Tipove podataka neću detaljnije opisati budući da nisu tema ovog rada.

3.2.4. Veze

Posljednja osnovna klasa objekata koja se nalazi u ER modelu su veze ili asocijacije. Veza je imenovana asocijacija između jednog ili više entiteta. Postojanje veze ovisi o asocijaciji među entitetima. Reprezentiraju se kao linije u dijagramu i generalno se implementiraju preko vanjskih ključeva. Kako bi opisali veze važni su nam slijedeći pojmovi: stupanj veze ili broj tipova entiteta uključenih u vezu, kardinalnost i opcionalnost. Prema broju tipova entiteta uključenih u vezu razlikujemo unarne, binarne i veze višeg stupnja. Nadalje, prema kardinalnosti razlikujemo veze tipa jedan naprema jedan, jedan naprema više i više naprema više. Posljednji pojam je opcionalnost i on označava mora li instanca entiteta sudjelovati u vezi ili ne. Sada ću svaki od ovih koncepata detaljnije objasniti. [3, str. 14 - 15], [4, str. 75 i 82], [6, str. 96 - 99]

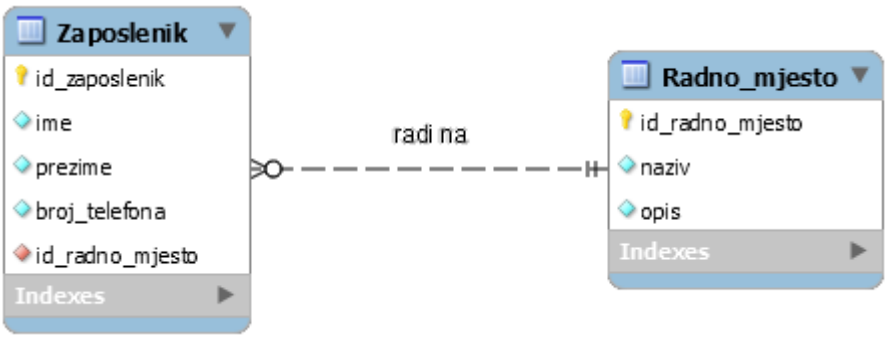
3.2.4.1. Stupanj veze

Stupanj veze ili broj tipova entiteta uključenih u vezu govori nam koliko je tipova entiteta povezano. Kod unarne veze povezuje se samo jedan entitet (veže se sam sa sobom). Unarna veza zapravo povezuje entitete koji su istog tipa, a ponekad se koristi i naziv binarna rekurzivna veza. Kao primjer unarne veze možemo uzeti entitet zaposlenik, svaki zaposlenik ima sebi nadređenog zaposlenika. Promotrimo primjer na slici 5. [6, str. 96 - 97], [5]

Slika 5: Unarna veza [vlastita izrada]

Na slici 5 vidimo da se nalazi samo jedan entitet i sada nam je definicija unarne veze jasnija. Svaki menadžer može biti nadređen nula ili više zaposlenika, dok svaki zaposlenik ima jednog ili nijednog nadređenog menadžera.

Nadalje binarna veza povezuje dva entiteta od kojih je svaki različitog tipa. Ova vrsta veze je ujedno i najčešća u primjeni. Za primjer binarne veze trebamo dva tipa entiteta. Možemo uzeti entitete zaposlenik i radno mjesto. Prvo ću staviti sliku kao primjer, a potom je detaljnije objasniti. [6, str. 97]



Slika 6: Binarna veza [vlastita izrada]

Kao što možemo vidjeti, na slici 6, imamo dva tipa entiteta. Prvi je zaposlenik, a drugi je radno mjesto i povezani su tipom veze „radi na“. Jedan zaposlenik (odnosno instanca entiteta tipa zaposlenik) radi na jednom i samo jednom radnom mjestu (odnosno instanci entiteta tipa radno mjesto), dok na jednom radnom mjestu radi nula ili više zaposlenika. Ovo bi mogli objasniti i na način da veza povezuje jednu instancu entiteta tipa radno mjesto s nula ili više instanci entiteta tipa zaposlenik, odnosno jednu instancu entiteta tipa zaposlenik s točno jednom instancom entiteta tipa radno mjesto. Za sada možemo zanemariti simbole O i | koji označavaju kardinalnost i opcionalnost, budući da će to biti objašnjeno u idućem poglavlju.

Što se tiče stupnja veze, ostale su još samo veze višeg stupnja. Njih ću objasniti nakon što objasnim kardinalnost i opcionalnost jer su nam ti pojmovi bitni kako bi lakše shvatili ternarne veze. Ternarna veza zapravo spada u napredne ER konstrukte.

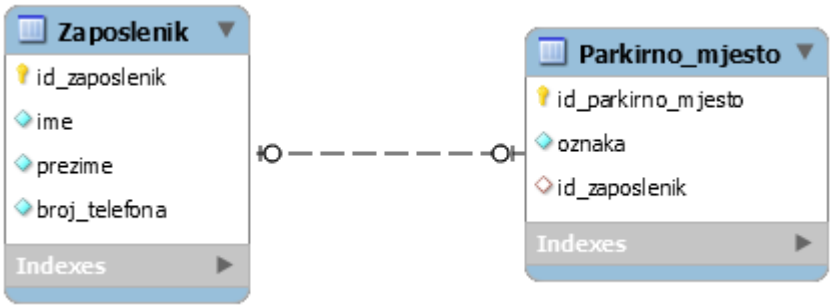
3.2.4.2. Kardinalnost i opcionalnost

Kardinalnost nam govori koliko najviše sudionika (instanci entiteta) smije sudjelovati u vezi. Prema kardinalnosti veze dijelimo na 1:1 (jedan naprema jedan) 1:M (jedan naprema više) i M:N (više naprema više). S druge strane, opcionalnost nam govori koliki je minimalni broj sudionika (instanci entiteta) koji sudjeluje u vezi. Opcionalnost nam označava mora li neka instanca entiteta sudjelovati u vezi (obavezno) ili ne (opcionalno). Kardinalnost je na vezi označena uz tip entiteta (krajnji lijevi i krajnji desni simbol), dok se opcionalnost nalazi odmah pored kardinalnosti. Na slici 7 možemo vidjeti samo vezu (tip veze) bez entiteta. [6, str. 98 - 99]

Slika 7: Veza sa simbolima [vlastita izrada]

Ako se vratimo na sliku 6, možemo vidjeti kako to izgleda s dva tipa entiteta. Postoje tri moguća simbola: O označava nula, | označava 1 (jedan) i treći se naziva „vranino stopalo“ (eng. crow's-foot). Notacija vraninog stopala se koristi na strani više kod veza jedan naprema više i više naprema više. [3, str. 20 - 21], [6, str. 99]

Slijede primjeri na kojima ću detaljnije objasniti veze 1:1, 1:M i M:N kako bi razumjeli sve vrste kardinalnosti te ću odmah pojasniti i opcionalnost na istim primjerima. Krenimo za početak s vezom jedan naprema jedan.



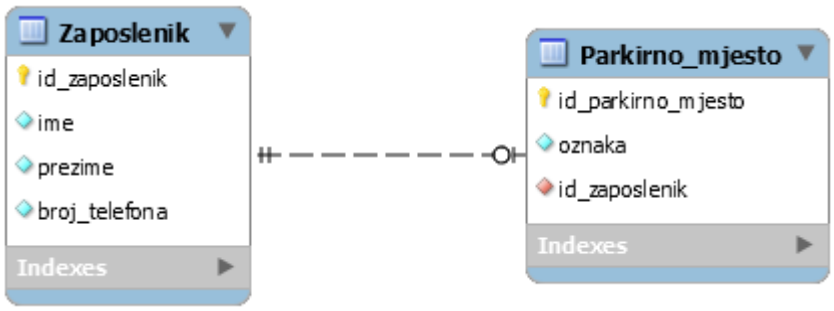
Slika 8: Binarna veza jedan naprema jedan [vlastita izrada]

Slika 8 prikazuje binarnu vezu 1:1 koja povezuje dva tipa entiteta: zaposlenik i parkirno mjesto. Što se tiče kardinalnosti, na obje strane veze je | ili 1 (veza jedan naprema jedan), a što se tiče opcionalnosti s obje strane je O ili nula. Stoga ovu vezu interpretiramo na slijedeći način: parkirno mjesto pripada najviše jednom zaposleniku (s time de ne mora pripadati niti jednom - opcionalnost), dok zaposlenik može imati najviše jedno parkirno mjesto, s time da ne mora imati niti jedno (opcionalnost).

Veza jedan naprema jedan se implementira putem vanjskog ključa koji nam predstavlja poveznicu između entiteta. Tablice povezujemo pomoću primarnih i vanjskih ključeva na način da uključimo primarni ključ iz prvog entiteta u drugi entitet kao vanjski ključ, koji ima isti tip podataka kao i primarni ključ iz prvog entiteta. Kod veze 1:1 nije bitno u kojem se entitetu nalazi vanjski ključ, već je potrebno razmisliti u kojem entitetu ima više smisla. [6, str. 103 - 104]

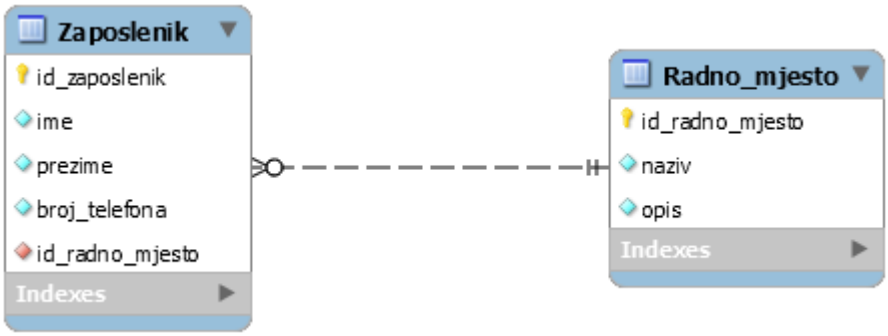
U ovom primjeru vanjski ključ se nalazi u tablici parkirno mjesto. Zapravo je svejedno u kojoj se tablici nalazi jer se s obje strane nalazi simbol O koji nam označava da je sudjelovanje instance entiteta u vezi opcionalno. Stoga će se naći neko parkirno mjesto koje ne pripada niti jednom zaposleniku i vrijednost vanjskog ključa biti će NULL. Isto bi bilo da se vanjski ključ nalazi u tablici zaposlenik. Ukoliko bi situacija bila nešto drugačija i svakom zaposleniku bi pripadalo jedno parkirno mjesto bez obzira treba li ga on ili ne, tada bi opcionalnost na strani tipa entiteta parkirno mjesto bila | ili 1. To bi onda interpretirali na slijedeći način: parkirno mjesto pripada jednom i samo jednom zaposleniku, dok zaposlenik može imati najviše jedno parkirno mjesto, s time da ne mora imati niti jedno (opcionalnost) – vidi sliku 9. U tom slučaju svako parkirno mjesto mora pripadati nekom zaposleniku, dok svaki zaposlenik ne mora imati parkirno mjesto. Tada nema smisla staviti vanjski ključ u tablicu zaposlenik jer bi za svakog zaposlenika koji nema parkirno mjesto vrijednost vanjskog ključa

bila NULL što treba izbjegavati. Sada takav problem nećemo imati jer svako parkirno mjesto mora pripadati nekom zaposleniku pa vrijednost vanjskog ključa nikada neće biti NULL.



Slika 9: Binarna veza 1:1 - vanjski ključ [vlastita izrada]

Slijedi nam veza jedan naprema više (1:M). Ovaj tip veze također spada u binarne veze jer povezuje dva tipa entiteta, primjer smo mogli vidjeti na slici 6 pomoću koje je bila objašnjena binarna veza. Kako bi nam bilo lakše pratiti ponovno ću staviti sliku.



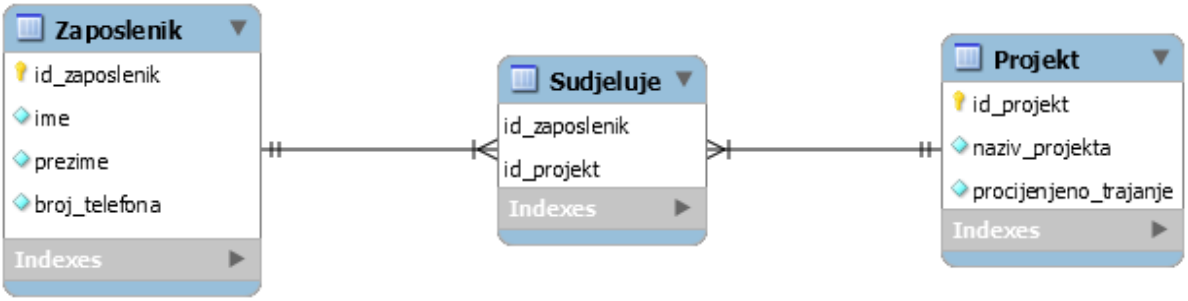
Slika 10: Binarna veza jedan naprema više [vlastita izrada]

Na slici 10 vidimo dva tipa entiteta: zaposlenik i parkirno mjesto povezana binarnom vezom čiji je tip „radi na“. Interpretacija glasi: jedan zaposlenik radi na jednom i samo jednom radnom mjestu, dok na jednom radnom mjestu radi nula ili više zaposlenika (opcionalnost). Što se tiče kardinalnosti, rekli smo da se nalazi uz sam tip entiteta i ovdje je to na strani entiteta radno mjesto jedan, a na strani entiteta zaposlenik je više. Upravo se zbog toga ovaj tip veze naziva jedan naprema više. Stoga možemo reći da veza povezuje jednu instancu entiteta radno mjesto s nula ili više instanci entiteta tipa zaposlenik, odnosno jednu instancu entiteta zaposlenik s točno jednom instancom entiteta tipa radno mjesto. Što se tiče opcionalnosti, koja

se nalazi odmah pored kardinalnosti, vidimo da svaki zaposlenik mora raditi na nekom radnom mjestu te je moguće da postoji radno mjesto na kojem ne radi niti jedan zaposlenik.

Kao što znamo tablice povezujemo pomoću primarnih i vanjskih ključeva. Vezu jedan naprema više implementiramo tako da vanjski ključ dodajemo u tip entiteta koji se nalazi na strani više (tip entiteta kod kojeg se nalazi simbol „vranino stopalo“). Vidimo da se u našem primjeru taj simbol nalazi kod tipa entiteta zaposlenik i tada vanjski ključ dodajemo u tablicu zaposlenik. Kako svaki zaposlenik radi na nekom radnom mjestu, u vanjski ključ upišemo šifru (id) radnog mjesta na kojem zaposlenik rad i s obzirom na to da zaposlenik mora raditi na nekom radnom mjestu vrijednost vanjskog ključa nikada neće biti NULL.

Posljednji tip veze prema kardinalnosti je više naprema više (M:N). Ova veza se može nacrtati na dijagramu, ali je implementacija nešto složenija. Znači, veza M:N ne može se izravno implementirati, već je potrebno dodati novi slabi entitet i tako vezu M:N zapravo dijelimo da dvije veze jedan naprema više (1:M). Pogledajmo sada primjer na slici 11 kako bi nam ovo bilo jasnije, a potom ću taj primjer detaljnije objasniti. [6, str. 104]



Slika 11: Binarna veza više naprema više [vlastita izrada]

Imamo dva tipa entiteta zaposlenik i projekt. Jedan zaposlenik može sudjelovati na nula (jednom) ili više projekata i na jednom projektu može sudjelovati nula (jedan) ili više zaposlenika. Kao što vidimo na slici 11, dodan je slabi entitet (nova tablica) sudjeluje i veza više naprema više je podijeljena na dvije veze jedan naprema više. Što se tiče veze 1:M, znamo kako se ona implementira jer je objašnjena u prošlom primjeru. Primjer (slika 11) je napravljen u alatu MySQL Workbench i ukoliko odaberemo tip veze M:N i odaberemo tablice (entitete) koje želimo povezati, automatski će nam generirati novu tablicu (slabi entitet) i dvije veze 1:M.

Znamo da tablice povezujemo pomoću primarnih i vanjskih ključeva i u tablici Sudjeluje oba atributa (stupca) predstavljaju vanjski ključ koji su ujedno i primarni ključevi. Znači, imamo složen primarni ključ (sastoji se od više stupaca) koji se sastoji od vanjskih ključeva. Vanjski ključevi ukazuju na tablicu roditelj na čiji se primarni ključ odnosi. U tablicu veze (slabi entitet) moguće je staviti i dodatne atribute koji su potrebni vezi. Ukoliko nas zanima kada je neki zaposlenik počeo sudjelovati na projektu, tada možemo dodati atribut „datum_početka“ i u njega zapisivati datum kada je pojedini zaposlenik započeo sudjelovati na pojedinom projektu.

U slučaju da zaposlenik može više puta raditi na istom projektu (npr. potreban je na nekom drugom projektu i tada prekine sudjelovati na prvome i nakon što završi posao na drugom projektu treba se ponovno vratiti na prvi projekt), potrebno je u sustav primarnog ključa dodati još jedan atribut. Za ovaj primjer bio bi dobar atribut „datum_početka“. [6, str. 104]

Kod veza je potrebno obratiti pažnju na redundantne veze. Ukoliko 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 potrebo je jednu vezu maknuti. [3, str. 58 - 60]

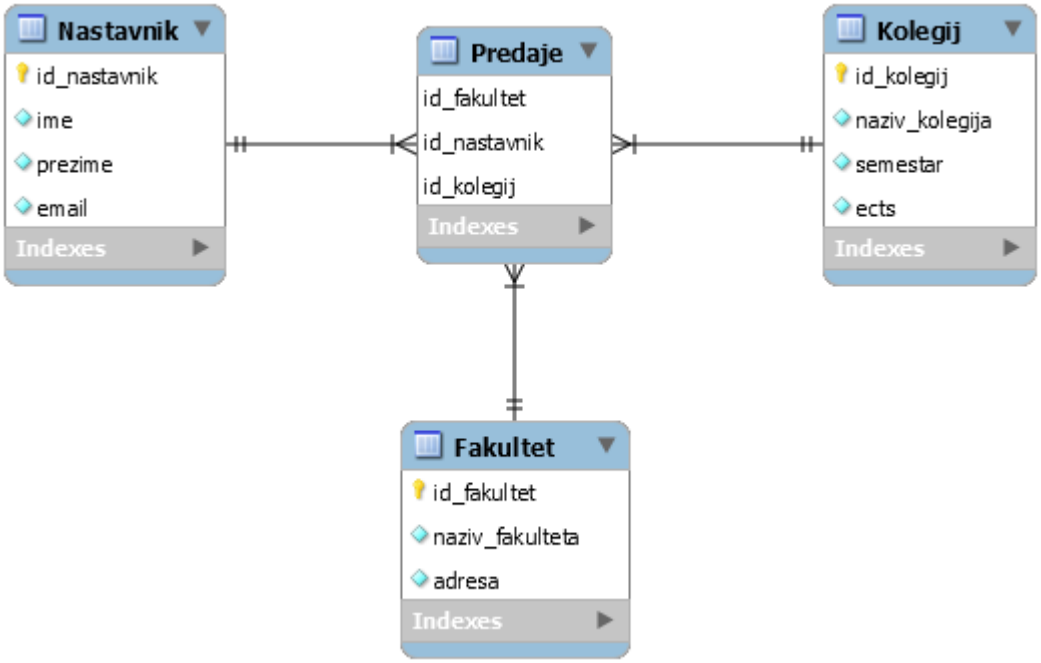
3.2.5. Napredni ER konstrukti

Prema Teoreyu, Lightstoneu i Nadeau [3] u napredne ER konstrukte spadaju generalizacija, agregacija, ternarna veza i veze višeg stupnja.

Njih koristimo za specifične slučajeve za koje nam osnovne tri klase objekata koje se nalaze u svakom ER modelu nisu dovoljne. Krenimo s razumijevanjem naprednih ER konstrukata, za početak s ternarnom vezom.

3.2.5.1. Ternarna veza

Od veza višeg stupnja još nam je zanimljiva jedino ternarna veza u kojoj sudjeluju tri tipa entiteta. Ova vrsta veze potrebna nam je kad pomoću binarne veze ne možemo opisati semantiku asocijacije između tri tipa entiteta. Potreban nam dodatan slabi entitet pomoću kojeg ćemo povezati ostala tri jaka tipa entiteta kako bi implementirali ternarnu vezu, slično kao i kod veze više naprema više. Promotrimo sliku 12 kao primjer ternarne veze te ću je pomoću slike detaljnije razjasniti. [3, str. 25 - 28], [6, str. 97 - 98]



Slika 12: Ternarna veza [vlastita izrada]

Na slici 12 koja prikazuje primjer ternarne veze možemo vidjeti četiri entiteta ili tablice. Imamo tri jaka tipa entiteta: fakultet, nastavnik i kolegij i jedan slabi tip entiteta predaje. Interpretacija bi glasila: na jednom fakultetu jedan kolegij može predavati više nastavnika, jedan nastavnik može jedan kolegij predavati na više fakulteta i jedan nastavnik na jednom fakultetu može predavati više kolegija. S obzirom na to da u vezi sudjeluju tri tipa entiteta, tada se i u kardinalnosti pojavljuje treći, npr. za prethodni primjer vrijedi M:N:P. Tako su moguće slijedeće četiri kombinacije: 1:1:1, 1:1:M, 1:M:N i M:N:P. Za vezu M:N:P znači da za svaki par (dva tipa entiteta) instanci entiteta (Fakultet, Kolegij) postoji više nastavnika, za svaki par instanci entiteta (Nastavnik, Kolegij) postoji više fakulteta i za svaki par instanci entiteta (Nastavnik, Fakultet) postoji više kolegija. Isto kao i kod veze više naprema više, u tablicu veze ili slabi entitet (u ovom slučaju Predaje) možemo staviti dodatne atribute koji su potrebni vezi. Što se tiče povezivanja, isto imamo složeni primarni ključ koji se sastoji od vanjskih ključeva (sva tri atributa u tablici Predaje). U našem primjeru bilo bi dobro dodati atribut „godina“ i staviti ga u sustav primarnih ključeva kako bi omogućili da nastavnik na fakultetu predaje kolegij više puta (zapravo više akademskih godina). [6, str. 110 - 111]

Pretpostavimo da vrijedi slijedeće: na jednom fakultetu jedan kolegij može predavati samo jedan nastavnik, jedan nastavnik može jedan kolegij predavati na samo jednom fakultetu i jedan nastavnik na jednom fakultetu može predavati samo jedan kolegij. Tada se radi o ternarnoj vezi 1:1:1, a što se tiče povezivanja isto imamo složeni primarni ključ koji se sastoji od bilo koja dva atributa (koji predstavljaju vanjske ključeve).

Nadalje možemo pretpostaviti da: na jednom fakultetu jedan kolegij može predavati samo jedan nastavnik, jedan nastavnik može jedan kolegij predavati na samo jednom fakultetu i jedan nastavnik na jednom fakultetu može predavati više kolegija. U ovom slučaju imamo ternarnu vezu 1:1:M, a što se tiče povezivanja isto imamo složeni primarni ključ koji se sastoji od atributa koji predstavlja vanjski ključ na strani više („id_kolegij“) i bilo kojeg od preostala dva vanjska ključa. Znači, u ovom primjeru mogli bi uzeti za primarni ključ („id_fakultet“ i „id_kolegij“) ili („id_nastavnik“ i „id_kolegij“).

Za posljednji slučaj možemo pretpostaviti da: na jednom fakultetu jedan kolegij može predavati samo jedan nastavnik, jedan nastavnik može jedan kolegij predavati na više fakulteta i jedan nastavnik na jednom fakultetu može predavati više kolegija. Tada se radi o ternarnoj vezi 1:M:N, a što se tiče povezivanja također imamo složeni primarni ključ koji se sastoji od atributa koji predstavlja vanjski ključ na strani više, u ovom slučaju („id_kolegij“ i „id_fakultet“).

3.2.5.2. Generalizacija i specijalizacija

Generalizacija predstavlja postupak uočavanja tipova entiteta koji dijele zajedničke karakteristike s drugim tipovima entiteta. Postupak rezultira skupom (entitetom) kojeg nazivamo nadtip ili supertip, koji uključuje druge entitete kao svoje podskupove, a koje nazivamo podtipovima ili subtipovima. Dakle, generalizacija je postupak kojim uočavamo da nekoliko tipova entiteta posjeduje zajedničke karakteristike (atribute) i da ih tada možemo generalizirati u jedan tip entiteta više razine, tj. nadtip. Tipovi entiteta niže razine ili podtipovi u generalizacijskoj hijerarhiji mogu biti disjunktni ili preklapajući podskupovi entiteta više razine. [3, str. 23 - 24], [5]

Za primjer možemo uzeti tipove entiteta osoba, nastavnik i student. Možemo primijetiti da nastavnik i student imaju neke zajedničke atribute kao što su ime, prezime i email. Time smo napravili prvi korak u postupku generalizacije te smo uočili tipove entiteta koji posjeduju neke zajedničke atribute. Sada ih možemo generalizirati u jedan tip entiteta više razine koji će u ovom slučaju biti entitet osoba. Tako će tip entiteta osoba biti nadtip ili supertipm entitetima nastavnik i student, dok će tipovi entiteta nastavnik i student biti podtipovi ili subtipovi entitetu osoba. Jedna od mogućih implementacija generalizacije je nasljeđivanje.

Specijalizacija je suprotna od generalizacije, a možemo reći i inverzni postupak od generalizacije u kojem se entitet više razine može rastaviti na entitete niže razine, tj. podtipove. [3, str. 24]

Znači da je generalizacija postupak u kojem uočavamo da neki tipovi entiteta posjeduju zajedničke atribute i tada ih generaliziramo u jedan tip entiteta više razine (od niže razine prema višoj, tj. od podtipova prema nadtipovima). S druge strane, specijalizacija je obrnuti postupak kod kojeg neki tip entiteta više razine možemo rastaviti na entitete niže razine (od više razine prema nižoj, tj. od nadtipa prema podtipovima).

3.2.5.3. Agregacija i kompozicija

Agregacija je vrsta apstrakcije između nadtipa i podtipa, ali se znatno razlikuje od generalizacije. Generalizacija se opisuje kao tip veze „je“ između podtipa i nadtipa, npr. student je osoba. S druge strane agregacija se opisuje kao tip veze „dio“ između cjeline i njezinih dijelova, npr. procesor i memorija su dijelovi računala. Prisjetimo se da kod generalizacije postoji nasljeđivanje i entiteti dijele neke zajedničke atribute. Kod agregacije nema nasljeđivanja, već svaki tip entiteta ima svoje jedinstvene atribute. [3, str. 25], [4, str. 225]

Možemo reći da je agregacija postupak kojim se pomoću dijelova formira cjelina. Tako su dijelovi računala zasebni entiteti sa svojim jedinstvenim atributima i svi zajedno čine računalo kao cjelinu.

Kompozicija predstavlja agregaciju koja ima barem jednu nezamjenjivu komponentu, tj. dio koji cjelina mora sadržavati i bez tog dijela to više ne bi bila ista cjelina. Tako za računalo možemo reći da procesor i memorija predstavljaju kompoziciju, dok optički uređaj predstavlja agregaciju (nije obavezan dio računala). [4, str. 225], [5]

4. Logičko oblikovanje baze podataka

Kao što sam već rekao, logičko oblikovanje baze podataka se temelji na razmatranju zavisnosti među atributima relacijske sheme koje zapravo predstavljaju uvjete integriteta. Za početak ću malo pojasniti što je to funkcijska zavisnost kako bi mogao detaljnije pojasniti prvu normalnu formu (1NF), drugu normalnu formu (2NF), treću normalnu formu (3NF) i Boyce – Codd-ovu normalnu formu (BCNF). Nakon što se upoznamo s osnovama možemo krenuti na normalizaciju relacijske sheme ili normalizaciju baze podataka koja je potrebna ukoliko postoje nepoželjne zavisnosti u relacijskoj shemi. [1, str. 167]

4.1. Normalne forme (NF)

Postoje slijedeće normalne forme (u daljnjem tekstu NF): 1NF, 2NF, 3NF, BCNF, 4NF, 5NF i 6NF. Od svih navedenih nama su za normalizaciju interesantne 1NF, 2NF, 3NF i BCNF pa ću njih detaljnije objasniti dok ću ostale samo spomenuti. Za 1NF, 2NF, 3NF i BCNF nam je vrlo bitna funkcijska zavisnost.

Dakle, za definiciju funkcijske zavisnosti trebamo neku relacijsku shemu R, relaciju r i neke proizvoljne atribute X i Y takvi da vrijedi $X, Y \subseteq R$. Tada kažemo da je izraz $X \rightarrow Y$ funkcijska zavisnost nad R. Funkcijska zavisnost $X \rightarrow Y$ vrijedi u relaciji r(R) ako je svakoj X vrijednosti pridružena jedna i samo jedna Y vrijednost. Ukoliko su X vrijednosti pridružene dvije ili više različitih Y vrijednosti, tada funkcijska zavisnost $X \rightarrow Y$ ne vrijedi u r. Slijedi jedan vrlo jednostavan primjer kako bi lakše shvatili definiciju funkcijske zavisnosti. [1, str. 169 - 170], [5], [7, str. 63 - 64]

Tablica 1: Funkcijska zavisnost - primjer 1

Student	Kolegij	Nositelj
S1	K1	N1
S2	K2	N2
S3	K2	N2

Ukoliko uzmemo prethodnu tablicu za primjer, vidimo da je relacijska shema od r: r(Student, Kolegij, Nositelj). Slijedi ispitivanje funkcijske zavisnosti $\mathrm{f}\colon \mathrm{Student}, \mathrm{Kolegij} \rightarrow \mathrm{Nositelj}$. Možemo uočiti kako je svakoj Student, Kolegij vrijednosti pridružena jedna i samo jedna Nositelj vrijednost.

Tablica 2: Funkcijska zavisnost - primjer 2

$S1, K1 \rightarrow N1$
$S2, K2 \rightarrow N2$
$S3, K2 \rightarrow N2$

Stoga možemo uočiti kako funkcijska zavisnost f vrijedi u relaciji r. Ako ispitamo neku drugu funkcijsku zavisnost $g: \text{Kolegij}, \text{Nositelj} \rightarrow \text{Student}$, tada ćemo dobiti slijedeće (vidi tablicu 3).

Tablica 3: Funkcijska zavisnost - primjer 3

$K1, N1 \rightarrow S1$
$\boldsymbol{K2, N2 \rightarrow S2}$
$\boldsymbol{K2, N2 \rightarrow S3}$

Promotrimo detaljnije posljednja dva podebljana retka u tablici 3. Vidimo kako su Kolegij, Nositelj K2, N2 vrijednosti pridružene dvije različite Student vrijednosti S2 i S3. Prema tome možemo zaključiti da funkcijska zavisnost g ne vrijedi u relaciji r.

Postoje trivijalne, netrivijalne i nestandardne funkcijske zavisnosti. Trivijalna funkcijska zavisnost ne predstavlja nikakvo ograničenje i vrijedi u svakoj relaciji u kojoj se mogu primijeniti. Naime, ako imamo atribute X i Y takve da vrijedi $Y \subseteq X$, kažemo da je zavisnost $X \rightarrow Y$ trivijalna (kao primjer možemo uzeti $XY \rightarrow X$ ili $Y \rightarrow \emptyset$). S druge strane, ukoliko vrijedi $Y \nsubseteq \text{X}$, tada je riječ o netrivijalnoj funkcijskoj zavisnosti koju koristimo i objašnjena je kroz primjer od tablice 1 do tablice 3 (možemo uzeti i slijedeće primjere $X \rightarrow Y$, $XY \rightarrow AB$). Na kraju nestandardne funkcijske zavisnosti su one zavisnosti čija je lijeva strana prazan skup, a definicija glasi: funkcijsku zavisnost oblika $\emptyset \rightarrow Y$, gdje je $Y \neq \emptyset$, zovemo nestandardna funkcijska zavisnost. Nestandardne funkcijske zavisnosti se rijetko javljaju jer da bi vrijedila nestandardna zavisnost $\emptyset \rightarrow Y$, stupac Y bi morao imati sve vrijednosti jednake, tj. sastojao bi se od samo jednog reda. [1, str. 184]

Sada dok imamo temeljno znanje, možemo krenuti s detaljima o normalnim formama.

4.1.1. Prva normalna forma (1NF)

Prva normalna forma (1NF) je temelj za izgradnju baze podataka. Ukoliko nam relacijska shema (R, F) nije u 1NF, tada se takva baza podataka smatra jako lošom. Najvažnije svojstvo 1NF je da svi atributi u R moraju biti elementarni, odnosno da sadrže samo jednu vrijednost. Drugo vrlo bitno svojstvo je da zapisi spremljeni u jednom stupcu budu istog tipa. Isto tako je bitno da stupci imaju različite nazive. Ukoliko dva ili više stupca imaju isti naziv, tada dolazimo do problema prilikom unosa, ažuriranja ili neke druge operacije nad podacima. Sada ću na jednom jednostavnom primjeru prikazati kako 1NF funkcionira. Recimo da imamo slijedeću tablicu. [1, str. 193 - 195], [3, str. 109], [5], [7, str. 100], [8], [9]

Tablica 4: 1NF - primjer 1

Id_jelo	**Naziv**	**Sastojci**
1	Sendvič	Sir, Kulen
3	Krem juha od mrkve	Mrkva, Sol, Papar
2	Rižoto od liganja	Riža, Lignje, Ulje

Iz tablice 4 možemo vidjeti da prethodno navedeno najvažnije svojstvo za 1NF nije zadovoljeno. Naime, ako pogledamo stupac „Sastojci“, vidimo da neki atributi u tom stupcu sadrže više vrijednosti što narušava valjanost svojstva. S druge strane, možemo vidjeti da su ostala svojstva zadovoljena, svaki stupac ima drugačiji naziv i u svaki su smješteni zapisi istog tipa. Da bi prethodna relacija zadovoljavala 1NF, potrebno je napraviti dekompoziciju, tj. razdvojiti zapise u stupcu „Sastojci“, što možemo vidjeti u tablici 5.

Tablica 5: 1NF - primjer 2

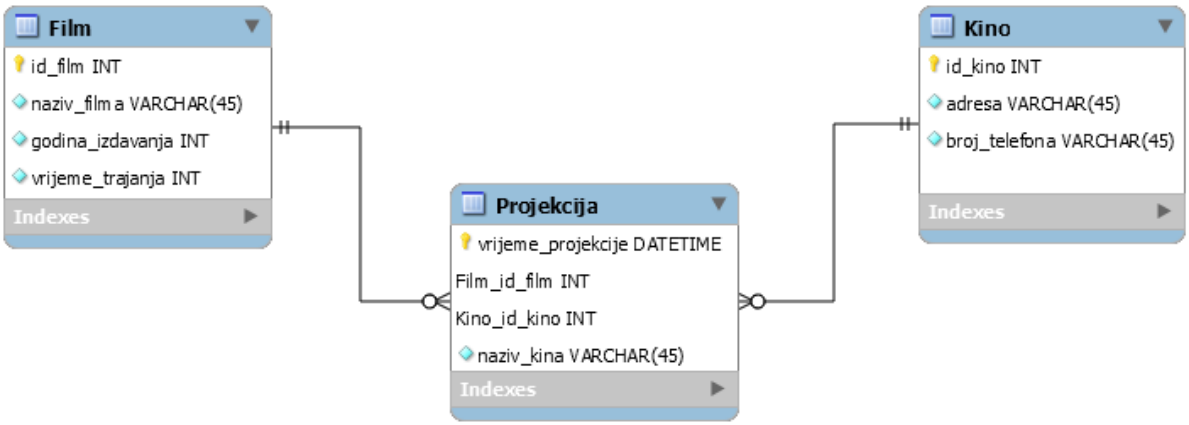
Id_jelo	Naziv	Sastojci
1	Sendvič	Sir
1	Sendvič	Kulen
3	Krem juha od mrkve	Mrkva
3	Krem juha od mrkve	Sol
3	Krem juha od mrkve	Papar
2	Rižoto od liganja	Riža
2	Rižoto od liganja	Lignje
2	Rižoto od liganja	Ulje

Ukoliko se vratimo natrag na svojstva 1NF, možemo vidjeti kako tablica 5 zadovoljava sva navedena svojstva i stoga možemo reći da je ona u prvoj normalnoj formi. Isto tako možemo primijetiti kako se neki podaci ponavljaju, recimo imamo isti naziv u više redova, ali što se tiče 1NF ova tablica je u redu. Problem ponavljanja podataka možemo riješiti tako da napravimo dvije tablice. Prva tablica bi bila Jelo u koju bi spremali nazive jela, a druga bi bila Sastojci u koju bi spremali nazive sastojaka. Kako bi jedno jelo moglo sadržavati više sastojaka i jedan sastojak bi se mogao nalaziti u više jela, riječ je o binarnoj vezi više naprema više za koju znamo kako se implementira.

4.1.2. Druga normalna forma (2NF)

Druga normalna forma (2NF) je zadovoljena ukoliko je relacijska shema (R, F) u 1NF i ukoliko ne postoji parcijalna zavisnost neključnog atributa od ključa. Na početku ovog poglavlja objasnio sam funkcijsku zavisnost i sada nam je potrebna kako bi napravili 2NF. [1, str. 195 - 196], [3, str. 111], [5], [7, str. 100], [8], [9]

Slijedi jedan primjer u kojem ću pojasniti što je parcijalna zavisnost i kako se, ukoliko postoji, može riješiti kako bi zadovoljili 2NF. U primjeru ću koristiti tri tablice od kojih je jedna nastala iz veze više naprema više.



Slika 13: 2NF – primjer 1 [vlastita izrada]

Na slici 13 možemo vidjeti tablice „Film“, „Kino“ i „Projekcija“. Tablica „Projekcija“ nastala je iz veze više naprema više između tablica „Film“ i „Kino“. Jedan film se može prikazivati u više kina, dok se u jednom kinu može prikazivati više filmova. Možemo vidjeti da se u tablici „Projekcija“ nalazi složeni primarni ključ koji se sastoji od vanjskih ključeva i atributa „vrijeme_projekcije“. Taj atribut ulazi u sastav primarnog ključa kako bi se neki film mogao više puta prikazivati u jednom kinu na isti dan. Ukoliko bi atribut „vrijeme_projekcije“ izbacili iz primarnog ključa, tada bi se u jednom kinu jedan film mogao prikazivati samo jednom na dan što nije dobro jer se većina filmova prikazuje najmanje dva puta dnevno, recimo ujutro i navečer. Ovakvim odabirom ključeva smo spriječili da se jedan film prikazuje istovremeno dva ili više puta u jednom kinu, ali se može prikazivati isti dan u drugo vrijeme, što je u redu. Sada ću tablice popuniti podacima kako bih mogao objasniti parcijalnu zavisnost i drugu normalnu formu.

Tablica 6: Film - 2NF

id_film	naziv_filma	godina_izdavanja	vrijeme_trajanja
1	Rambo	1982	93
2	Spašavanje vojnika Rayana	1998	169
3	Kum	1972	175

Tablica 7: Kino - 2NF

id_kino	adresa	broj_telefona
1	Zagreb, Ilica 43	001543271
2	Varaždin, Braće Radića 12	042876543
3	Osijek, Vukovarska 32	031987621

Prethodne dvije tablice nam služe kako bi spremali podatke o filmovima (tablica 6) i kinima (tablica 7) u kojima se filmovi prikazuju. Slijedeća tablica je ključna kako bi objasnio 2NF i parcijalnu zavisnost pa ću sada popuniti i tu tablicu podacima.

Tablica 8: Projekcija - 2NF

vrijeme_projekcije	film_id_film	kino_id_kino	naziv_kina
2018-09-10 14:30	1	1	Cinestar Zagreb
2018-09-10 16:00	2	2	Cinestar Varaždin
2018-09-11 19:00	2	1	Cinestar Zagreb
2018-09-12 20:00	3	1	Cinestar Zagreb
2018-09-13 20:30	3	2	Cinestar Varaždin
2018-09-13 20:00	3	3	Cinestar Osijek

Sve tablice su popunjene i sada možemo temeljitije promotriti tablicu 8 „Projekcija“. Vidimo da se primarni ključ sastoji od atributa „vrijeme_projekcije“, „film_id_film“ i „kino_id_kino". S obzirom na to da je ovo veza više naprema više u tablici „Projekcija“ koja je slabi entitet, funkcijskih zavisnosti ne bi trebalo biti. Ako malo bolje promotrimo, možemo uočiti zavisnost „kino_id_kino“ → „naziv_kina“, za kino 1 naziv je „Cinestar Zagreb“, za kino 2 naziv je „Cinestar Varaždin“ i tako dalje. Imamo složeni primarni ključ i „naziv_kina“ ovisi samo o atributu „kino_id_kino“, dok ostala dva atributa iz primarnog ključa nemaju nikakve veze s nazivom kina. Ovakav slučaj dok je neki atribut ovisan o dijelu primarnog ključa nazivamo parcijalna zavisnost. S obzirom na to da se u drugoj normalnoj formi ne smije nalaziti parcijalna zavisnost potrebno ju je maknuti, tj. napraviti dekompoziciju. U ovom slučaju najbolji način da to napravimo je da „naziv_kina“ prebacimo u tablicu „Kino“, što ima i više smisla nego prethodni primjer.

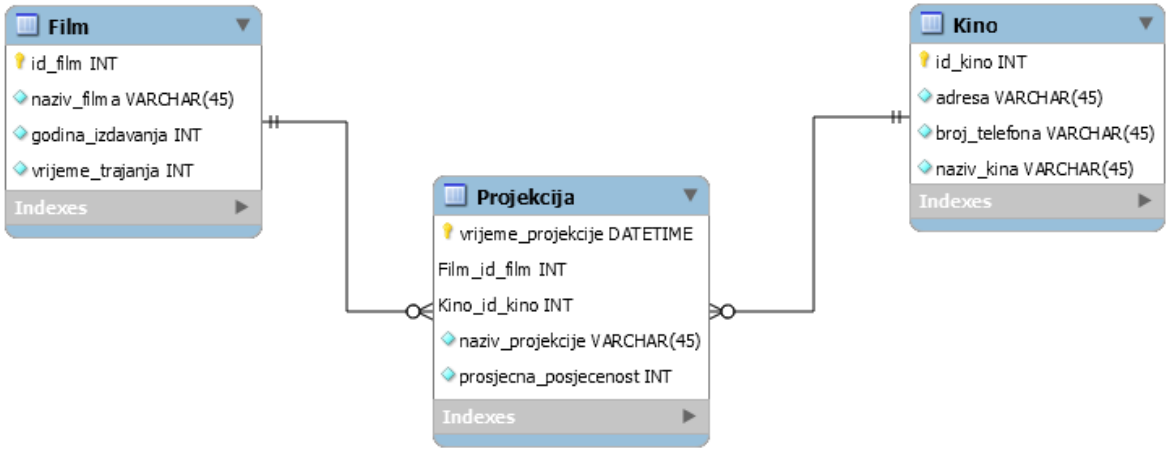
Slika 14: 2NF – primjer 2 [vlastita izrada]

Na slici 14 možemo vidjeti kako izgleda ERA model koji zadovoljava 2NF. Sada neću ponovno prikazivati tablice jer je samo zadnji stupac iz tablice „Projekcija“ premješten u tablicu „Kino“. Još jedan primjer kako se radi dekompozicija za 2NF možete vidjeti u poglavlju o normalizaciji koje slijedi nakon objašnjenja normalnih formi.

4.1.3. Treća normalna forma (3NF)

Već smo kod 2NF mogli uočiti kako tablica prvo mora biti u 1NF da bi je mogli uz dodatno ograničenje pretvoriti u 2NF. Isto tako vrijedi i za 3NF, relacijska shema (R, F) je u 3NF ako je u 1NF i ako ne postoji tranzitivna zavisnost neključnog atributa od ključa.[1, str. 197], [3, str. 113 - 115], [5], [7, str. 101], [8]

Slijedi jedan primjer na kojem ću pojasniti što je tranzitivna zavisnost i kako se ona može maknuti da bi tablica bila u 3NF. Koristit ću prethodni primjer za 2NF, samo ću dodati neke atribute u tablicu „Projekcija“.



Slika 15: 3NF - primer 1 [vlastita izrada]

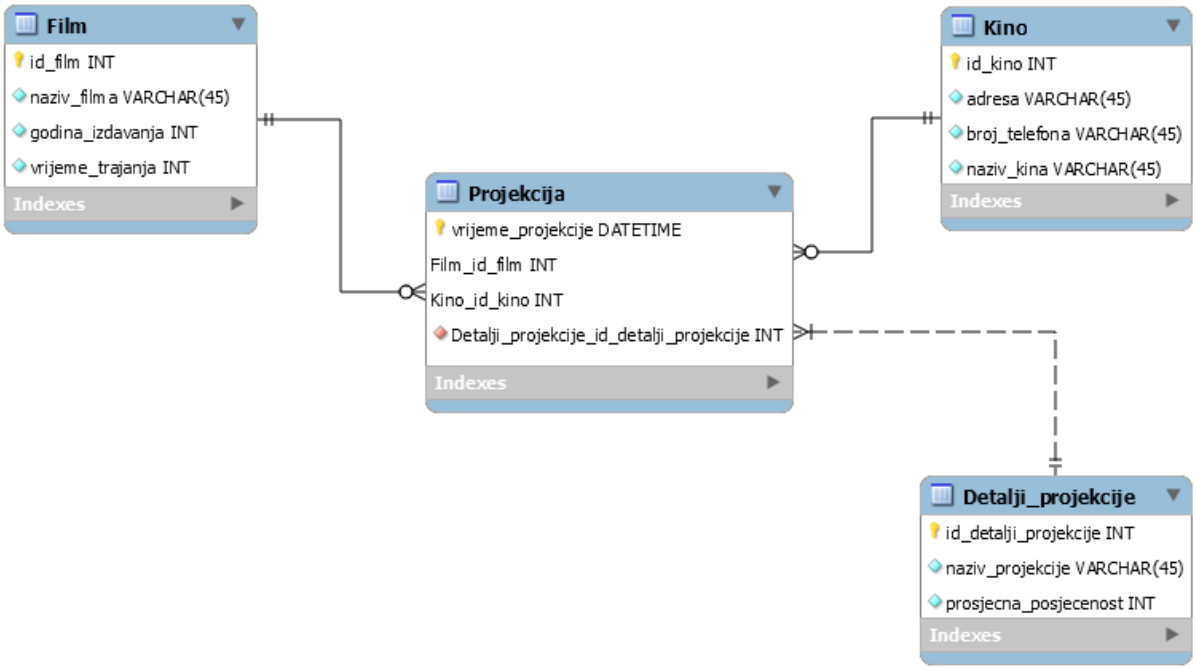
Kao što možemo vidjeti na slici 15, dodao sam atribute „naziv_projekcije“ i „prosjecna_posjecenost. Možemo primijetiti da je primarni ključ u tablici „Projekcija“ složen i sastoji se od stupaca „vrijeme_projekcije“, „Film_id_film“ i „Kino_id_kino“. U tablici vrijedi funkcijska zavisnost „vrijeme_projekcije“ → „naziv_projekcije“ i funkcijska zavisnost „naziv_projekcije“ → „prosjecna_posjecenost“ (neključni atribut ovisi o neključnom atributu). Stoga možemo uočiti tranzitivnu zavisnost neključnog atributa od ključa, tj. neključni atribut „prosjecna_posjecenost“ tranzitivno ovisi o ključu „vrijeme_projekcije“. Isto tako možemo uočiti da je prva zavisnost parcijalna jer „naziv_projekcije“ ovisi samo o dijelu ključa.

Sada ću tablicu „Projekcija“ popuniti podacima kako bi mogli lakše shvatiti prethodno objašnjenje i tranzitivnu zavisnost.

Tablica 9: Projekcija - 3NF

vrijeme_projekcije	Film_id_film	Kino_id_kino	naziv_projekcije	prosjena_posjecenost
2018-09-10 14:30	1	1	Popodnevna	60
2018-09-10 10:00	2	2	Jutarnja	40
2018-09-11 19:00	2	1	Večernja	120
2018-09-12 20:00	3	1	Večernja	120
2018-09-13 20:30	3	2	Večernja	120
2018-09-13 20:00	3	3	Večernja	120

Ukoliko pažljivije promotrimo nove atribute, možemo uočiti kako „prosjecna_posjecenost“ ovisi o atributu „naziv_projekcije“, što znači da između njih postoji zavisnost. Isto tako atribut „naziv_projekcije“ ovisi o atributu „vrijeme_projekcije“, stoga možemo uočiti tranzitivnu zavisnost „vrijeme_projekcije“ → „naziv_projekcije“ → „prosjecna_posjecenost“. Kako bi riješili ovaj problem potrebno je napraviti dekompoziciju kojom dobijemo novu tablicu u koju ćemo spremati detalje projekcije. Na slijedećoj slici možemo vidjeti kako izgledaju tablice nakon dekompozicije.



Slika 16: 3NF - primjer 2 [vlastita izrada]

Na slici 16 vidimo da u novu tablicu spremamo detalje projekcije i time smanjujemo redundanciju, tj. ponavljanje istih podataka za svaku projekciju. Sada ću napraviti prikaz nove tablice i tablice „Projekcija“ kako bi vidjeli da više nema tranzitivne zavisnosti i uočili prednosti u smislu smanjenja dupliciranja podataka ili redundancije.

Tablica 10: Projekcija - 3NF - dekomponirana

vrijeme_projekcije	film_id_film	kino_id_kino	detalji_projekcije_id_detalji_projekcije
2018-09-10 14:30	1	1	2
2018-09-10 10:00	2	2	1
2018-09-11 19:00	2	1	3
2018-09-12 20:00	3	1	3
2018-09-13 20:30	3	2	3
2018-09-13 20:00	3	3	3

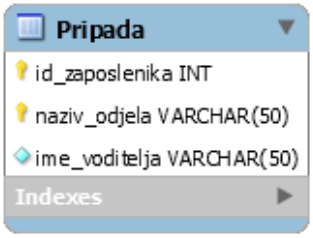
Tablica 11: Detalji_projekcije - 3NF

id_detalji_projekcije	naziv_projekije	prosjecna_posjecenost
1	Jutarnja	40
2	Popodnevna	60
3	Večernja	120

4.1.4. Boyce – Coddova normalna forma (BCNF)

Razlog zašto nosi naziv BCNF je zbog toga što je to nadogradnja 3NF ili specijalni slučaj 3NF. Relacijska shema (R, F) je u BCNF ako je u 3NF i ako svaka netrivijalna funkcijska zavisnost $X \rightarrow Y$, koja vrijedi u F, posjeduje svojstvo da lijeva strana zavisnosti (X) sadrži ključ. [1, str. 198], [3, str. 115], [7, str. 101], [8]

Pogledajmo sliku 17 kao primjer tablice koja nije u BCNF kako bi mogli na temelju primjera lakše shvatiti o čemu je riječ u samoj definiciji Boyce – Coddove normalne forme.



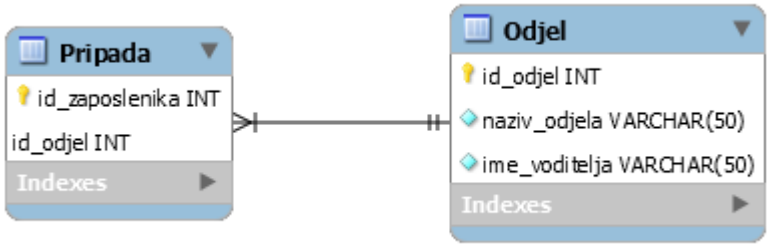
Slika 17: BCNF - primjer 1 [vlastita izrada]

Na slici 17 vidimo tablicu koja sadrži podatke o pripadnosti zaposlenika nekom odjelu unutar tvrtke. Primarni ključ je složen i sastoji se od stupaca „id_zaposlenika“ i „naziv_odjela“. Vrijede slijedeće funkcijske zavisnosti: „id_zaposlenika“, „naziv_odjela“ → „ime_voditelja“ i „ime_voditelja“ → „naziv_odjela“. Kao što možemo primijetiti, druga zavisnost nam narušava svojstvo BCNF da lijeva strana sadrži ključ. Pogledajmo sada tablicu 12 koja predstavlja tablicu „Pripada“ sa slike i u kojoj se nalaze potrebni podaci kako bi lakše razumjeli BCNF.

Tablica 12: Pripada - BCNF

id_zaposlenika	naziv_odjela	ime_voditelja
1	Prvi odjel	Bojan
1	Drugi odjel	Luka
2	Prvi odjel	Bojan
2	Treći odjel	Damjan
3	Treći odjel	Matija

Možemo vidjeti da za dani odjel, svaki zaposlenik ima samo jednog voditelja („id_zaposlenika“, „naziv_odjela“ → „ime_voditelja“) i da svaki voditelj vodi točno jedan odjel („ime_voditelja“ → „naziv_odjela“). Kako bi zadovoljili navedeno svojstvo za BCNF, potrebno je napraviti dekompoziciju na dvije tablice, tablicu „Pripada“ i tablicu „Odjel“, što možemo vidjeti na slici 18.



Slika 18: BCNF - primjer 2 [vlastita izrada]

Dodan je atribut „id_odjel“ kako bi mogli povezati tablice. Sada tablica „Pripada“ ima samo ključne atribute (oba čine složeni primarni ključ) i stoga nema funkcijskih zavisnosti, dok tablica „Odjel“ ima primarni ključ „id_odjel“ i ostalo su neključni atributi te je jedina funkcijska zavisnost u tablici „id_odjel“ → „naziv_odjela“, „ime_voditelja“. S obzirom na to da nema drugih funkcijskih zavisnosti, sva svojstva BCNF su ispunjena i sada je tablica u BCNF. Pogledajmo primjer unesenih podataka.

Tablica 13: Pripada -BCNF - dekomponirana

id_zaposlenika	id_odjel
1	1
1	2
2	1
2	3
3	4

Tablica 14: Odjel - BCNF - dekomponirana

id_odjel	naziv_odjela	ime_voditelja
1	Prvi odjel	Bojan
2	Drugi odjel	Luka
3	Treći odjel	Damjan
4	Treći odjel	Matija

4.2. Više normalne forme

Kao što sam rekao, više normalne forme u ovom radu neću detaljno objašnjavati, već ću ih samo spomenuti uz definicije i vrlo jednostavne primjere. U više normalne forme spadaju 4NF, 5NF i 6NF. Kako bi mogli razumjeti više normalne, potrebno je znati višeznačne zavisnosti i zavisnosti spoja. Krenimo prvo s razumijevanjem višeznačne zavisnosti.

4.2.1. Višeznačne zavisnosti

Višeznačnu zavisnost možemo prepoznati prema slijedećem obliku $X \rightarrow\rightarrow Y$. Funkcijske zavisnosti imaju samo jednu strelicu, dok višeznačne zavisnosti imaju dvije. Kako bi postojala netrivijalna višeznačna zavisnost, tablica mora imati najmanje tri stupca. Ukoliko tablica ima samo dva stupca, tada može postojati jedino trivijalna višeznačna zavisnost. Zavisnost $X \rightarrow\rightarrow Y$ u tablici R vrijedi ako instanca tablice R(X, Y, Z) sadrži par redova koji imaju istu ili dupliciranu vrijednost za X te isto tako sadrži par redova dobivenih izmjenjivanjem Y vrijednosti u originalnom paru. Višeznačna zavisnost je trivijalna ukoliko je Y podskup od X ili ako je unija X i Y jednaka relacijskoj shemi tablice (skupu atributa koje tablica sadrži). U suprotnome, višeznačna zavisnost je netrivijalna ukoliko Y nije podskup od X ili ako X i Y zajedno ne čine relacijsku shemu tablice. Kako bi lakše shvatili ovu definiciju, pogledajmo primjer. [3, str. 127-129], [7, str. 64], [8]

Tablica 15: Višeznačna zavisnost

id_zaposlenik	projekt	slobodna_aktivnost
1	P1	Nogomet
1	P1	Košarka
1	P2	Nogomet
1	P2	Košarka
2	P2	Nogomet
2	P2	Košarka

Tablica sadrži slijedeće višeznačne zavisnosti: „id_zaposlenik“→→„projekt“ i „id_zaposlenik“→→„slobodna_aktivnost“, s time da su „projekt“ i „slobodna_aktivnost“ međusobno nezavisni, tj. ne ovise jedan o drugome. Imamo dva projekta i svaki zaposlenik se bavi dvjema slobodnim aktivnostima. Ukoliko bi željeli dodati zaposlenika 2 na projekt P1,

morali bi dodati dva nova reda ili zapisa u prethodnu tablicu kako bi zadovoljili višeznačne zavisnosti koje u njoj vrijede. Pogledajmo tablicu 16 u kojoj je zaposlenik 2 dodan na projekt P1.

Tablica 16: Višeznačna zavisnost – proširena tablica

id_zaposlenik	projekt	slobodna_aktivnost
1	P1	Nogomet
1	P1	Košarka
1	P2	Nogomet
1	P2	Košarka
2	P2	Nogomet
2	P2	Košarka
2	P1	Nogomet
2	P1	Košarka
3	P3	Tenis

Zaposlenik se bavi dvjema slobodnim aktivnostima i za svaki projekt obje aktivnosti se moraju zapisati jer su stupci međusobno nezavisni. Možemo primijetiti da zadnji red u tablici 16 zadovoljava definiciju jer je trivijalan.

4.2.2. Četvrta normalna forma (4NF)

Relacijska shema (R, F) je u 4NF ako za svaku netrivijalnu višeznačnu zavisnost $X \rightarrow\rightarrow Y$, koja slijedi iz F, vrijedi da njena lijeva strana (X) sadrži ključ od (R, F). Netrivijalna višeznačna zavisnost objašnjena je u prethodnom poglavlju o višeznačnim zavisnostima. [3, str. 130], [5], [7, str. 101]

Ukoliko se vratimo na tablicu 16 koja predstavlja proširen primjer višeznačne zavisnosti, vidimo da nema primarnog ključa i postoji najmanje jedna netrivijalna višeznačna zavisnost što znači da tablica nije u 4NF. Kako bi riješili navedene probleme, potrebno je napraviti dekompoziciju na dvije tablice. Naime, ovdje se radi o dekompoziciji bez gubitaka. Tako ju zovemo jer se svaka nova tablica bazira na istom atributu, onom koji se nalazi s lijeve strane višeznačne zavisnosti. Kod našeg primjera obje tablice će se bazirati na atributu „id_zaposlenik“. Tablica 17 prikazuje kako to izgleda nakon dekompozicije. [3, str. 132]

Tablica 17: 4NF - dekompozicija

id_zaposlenik	projekt
1	P1
1	P2
2	P1
2	P2
3	P3

id_zaposlenik	slobodna_aktivnost
1	Nogomet
1	Košarka
2	Nogomet
2	Košarka
3	Tenis

U obje tablice imamo složeni primarni ključ koji se sastavlja od oba stupca u pojedinoj tablici. Ovakvom dekompozicijom zadovoljeni su uvjeti 4NF i možemo reći da tablice zadovoljavaju 4NF.

4.2.3. Zavisnosti spoja

Za početak ću opisati dekompoziciju skupa atributa R. Skup d(R) = {R1, R2, …, Rn} je dekompozicija skupa atributa R ako vrijede slijedeća tri uvjeta: (1) $Ri \subseteq R$ za svaki $i = 1, 2, \dots, n$ (2) $Ri \neq \emptyset$ za svaki $i = 1, 2, \dots, n$ i (3) $R1 \cup R2 \cup \dots \cup Rn = R$. Neka je d(R) = {R1, R2, …, Rn} dekompozicija skupa atributa R. U tom slučaju izraz ∞(R1, R2, …, Rn) zovemo zavisnost spoja nad R. Zavisnost spoja ∞(R1, R2, …, Rn) vrijedi u r(R) ako je r(R) jednak prirodnom spoju projekcija r(R) na komponente zavisnosti spoja ili ako vrijedi izraz: $r = \pi[R1](r) \infty \pi[R2](r) \infty \dots \infty \pi[Rk](r)$, gdje su R1, R2, …, Rk komponente zavisnosti spoja. [1, str. 185 - 186], [7, str. 65 i 93]

Napravimo sada mali primjer kako bi mogli lakše razumjeti definiciju.

Tablica 18: Zavisnost spoja

A	B	C
1	1	3
1	2	2
2	2	1

Ispitajmo zavisnost spoja ∞(AB, AC) u relaciji r(ABC). Potrebno je prvo napraviti projekciju na dvije tablice $\pi[AB]$(r) (AB) i $\pi[AB]$(r) (AC).

Tablica 19: Zavisnost spoja - projekcija

A	B
1	1
1	2
2	2

A	C
1	3
1	2
2	1

Nakon što smo napravili projekciju, potrebno je dobivene tablice spojiti prirodnim spajanjem pa računamo $\pi[AB]$(r) (AB) ∞ $\pi[AB]$(r) (AC). Nakon izračuna dobijemo slijedeću tablicu s(ABC).

Tablica 20: Zavisnost spoja - prirodno spajanje

A	B	C
1	1	3
1	1	2
1	2	3
1	2	2
2	2	1

Možemo vidjeti da je relacija s(ABC) ≠ r(ABC) i stoga zavisnost spoja ∞(AB, AC) ne vrijedi u relaciji r(ABC). Ukoliko dekompozicija na dvije tablice nije dovoljna, potrebno je raditi dekompoziciju na 3 tablice. Ispitajmo sada iduću zavisnost spoja ∞(AB, AC, BC). Tablica 19 pokazuje prve dvije projekcije pa je potrebna još treća projekcija $\pi[BC]$(r) (BC) koju vidimo na tablici 21.

Tablica 21: Zavisnost spoja - dodatna projekcija

B	C
1	3
2	2
2	1

Nakon toga je potrebno izračunati prirodni spoj tablica. Tablica 20 prikazuje prirodni spoj $\pi[AB]$(r) (AB) ∞ $\pi[AB]$(r) (AC) kojim dobijemo s(ABC) i sada je na to potrebno još dodati s(ABC) ∞ $\pi[BC]$(r) (BC).

Tablica 22: Zavisnost spoja - dodatno prirodno spajanje

A	B	C
1	1	3
1	2	2
2	2	1

Vidimo da smo dobili tablicu koja je jednaka početnoj i time možemo zaključiti da zavisnost spoja ∞(AB, AC, BC) vrijedi u relaciji r(ABC). Zavisnost spoja ∞(R1, R2, …, Rn) je netrivijalna ako je $Ri \neq R$ za svaki $i = 1, 2, \dots, n$. Ovo su neki primjeri netrivijalnih zavisnosti spoja ∞(AB, B, CD) i ∞(AB, AC). Postoje i trivijalne zavisnosti spoja koje vrijede u svakoj relaciji, tj. zavisnost spoja ∞(R1, R2, …, Rn) je trivijalna ako je $Ri = R$ za neki $i = 1, 2, \dots, n$. Za naš primjer relacije r(ABC) trivijalne zavisnosti spoja bile bi ∞(AB, ABC), ∞(AB, AC, BC) ili ∞(AB, ABC, BC). [1, str. 186 - 187]

4.2.4. Peta normalna forma (5NF)

Relacijska shema (R, F) je u 5NF ako za svaku netrivijalna zavisnost spoja ∞(R1, R2, …, Rn) vrijedi da svaka njezina komponenta R1, R2, …, Rn sadrži neki ključ od (R, F). [3, str. 133], [5], [7, str. 101]

Ukoliko se vratimo na naš primjer zavisnosti spoja, točnije tablicu 18, jedini ključ je K = ABC. Imamo netrivijalnu zavisnost spoja ∞(AB, AC, BC) pa možemo uočiti da niti jedna od tri komponente ne sadrži ključ i stoga tablica nije u 5NF. Potrebno je izvršiti dekompoziciju skupa R preko netrivijalne zavisnosti spoja ∞(AB, AC, BC) kojom dobijemo tri tablice vidljive na tablicama 19 i 21. Dobivena dekompozicija je u 5NF.

4.2.5.Šesta normalna forma (6NF)

Relacijska shema (R, F) je u 6NF ako je svaka zavisnost spoja trivijalna, tj. nema netrivijalnih zavisnost spoja. [5], [7, str. 101]

4.3. Normalizacija

Nakon što imamo konceptualni model baze podataka, slijedi logičko oblikovanje čiji je sastavni dio normalizacija. Ukoliko smo utrošili dovoljno vremena i dobro promislili prilikom izgradnje konceptualnog modela, tada on rezultira bazom koja je već normalizirana ili su potrebne neke manje preinake kako bi ju normalizirali. [3, str. 107]

Normalizacija baze podataka je vrlo bitna kako bi izbjegli probleme koji se mogu naći u bazi podataka. Ti problemi su redundancija podataka, tj. ponavljanje istih ili nepotrebnih podataka, što ima za posljedicu zauzimanje više prostora, odnosno memorije i anomalije unosa, ažuriranja i brisanja podataka. Zamislimo da imamo bazu koja se sastoji od samo jedne tablice u kojoj se nalaze svi potrebni podaci. Recimo zaposlenici i projekti na kojima rade. Tada imamo redundantne podatke o projektima jer svaki zaposlenik radi na nekom projektu i ukoliko više zaposlenika radi na istom projektu, za svakog imamo zapisane detalje o projektu koji su isti. Ako želimo pretražiti nešto, tada bi se trebala pretražiti cijela tablica koja sadrži puno podataka koji su redundantni. Anomalija ažuriranja se javlja ako želimo promijeniti broj telefona nekog zaposlenika jer ako zaposlenik radi na više projekata i u svakom zapisu se nalaze redundantni podaci o zaposleniku, tada moramo ažurirati svaki red ili zapis. Nadalje, anomalija unosa se javlja ukoliko želimo dodati novog zaposlenika na određeni projekt i tada moramo paziti da su podaci o projektu konzistentni (da su isti kao i u prethodnim zapisima tog projekta). Na kraju se anomalija brisanja javlja ako želimo maknuti nekog zaposlenika iz projekta, a to je jedini projekt na kojem je radio. Da bi ga maknuli potrebno je obrisati taj zapis, a s obzirom da je to jedini zapis u kojem se taj zaposlenik nalazi, gubimo podatke o njemu (znači ime, prezime, broj telefona). Da bi riješili ovakve probleme potrebno je dekomponirati tu jednu tablicu na više tablica kako bi riješili navedene probleme i učinili bazu bržom i pouzdanijom. Upravo nam postupak normalizacije služi kako bi to postigli. Sam postupak normalizacije se radi pomoću normalnih formi, znači analiziramo zavisnosti među atributima i radimo dekompoziciju (projekciju) na više tablica kako bi se riješili neželjenih zavisnosti. [3, str. 108 - 109]

U prethodnim poglavljima o normalnim formama i višim normalnim formama sam uz definicije dao i primjere pa tamo možemo vidjeti na primjerima kako se provodi postupak normalizacije za pojedinu normalnu formu.

Isto tako bih želio naglasiti da se većinom radi normalizacija do BCNF, a normalizacija na više normalne forme se koristi samo kada je to potrebno.

Slijedi jedan primjer normalizacije do BCNF, s time da će tablica već biti u 1NF jer ju je vrlo lako dovesti u 1NF. Imamo slijedeću tablicu:

Tablica 23: Normalizacija

id_projekt	naziv_projekta	id_zaposlenik	ime	prezime	id_odjel	naziv_odjela	grad
101	P1	1	Luka	Erceg	10	O1	Varaždin
101	P1	2	Damjan	Poljak	10	O1	Varaždin
101	P1	3	Bojan	Kavur	10	O1	Varaždin
202	P2	4	Matija	Horvat	15	O2	Zagreb
202	P2	5	Karlo	Kos	15	O2	Zagreb
202	P2	3	Bojan	Kavur	15	O2	Zagreb

Tablica 23 prikazuje zaposlenike koji rade na projektima u određenim odjelima. Pretpostavimo da imamo slijedeće funkcijske zavisnosti:

1) „id_projekt“ → „naziv_projekta“, „id_odjel“
2) „id_odjel“ → „naziv_odjela“, „grad“
3) „id_zaposlenik“ → „ime“, „prezime“

Kao što sam i rekao i možemo vidjeti, tablica se već nalazi u 1NF jer su atributi elementarni i svaki stupac ima drugačiji naziv. Slijedeće moramo provjeriti zadovoljava li uvjete da bude u 2NF, a to je da ne postoji parcijalna zavisnost. Možemo uočiti tranzitivnu zavisnost između 1) i 2) („id_projekt“ → …, „id_odjel“ → „naziv_odjela“, …), što će nam biti važno za 3NF. Iz navedenih funkcijskih zavisnosti možemo vidjeti da je jedini kandidat za primarni ključ („id_projekt“, „id_zaposlenik“) i zaključujemo da je to primarni ključ. Sada je potrebno provjeriti je li zavisnost „id_projekt“, „id_zaposlenik“ → „naziv_projekta“, „id_odjel“, „naziv_odjela“, „grad“, „ime“, „prezime“ parcijalna. Na temelju funkcijskih zavisnosti koje vrijede možemo zaključiti da je navedena zavisnost parcijalna jer neključni atribut zavisi od ključa. Točnije, postoje dvije zavisnosti: 1) i 3) zbog kojih imamo parcijalnu zavisnost neključnih atributa „naziv_projekta“, „id_odjel“, „naziv_odjela“, „grad“, „ime“, „prezime“ od ključa „id_projekt“, „id_zaposlenik“. Sada je potrebno napraviti dekompoziciju koju možemo raditi prema jednoj od dvije funkcijske zavisnosti zbog koje imamo parcijalnu zavisnost. Ovdje nam je lakše ako uzmemo 3) jer ako uzmemo 1) moramo pripaziti na tranzitivnu zavisnost, ali uzet ćemo 1) da prikažemo teži postupak. Znači, dekompoziciju radimo tako da u prvu tablicu stavimo sve iz odabrane funkcijske zavisnosti koja je u ovom slučaju 1) i moramo pripaziti da uključimo i tranzitivnu vezu čime dobijemo tablicu 24 i primarni ključ nove tablice bit će lijeva strana odabrane zavisnosti („id_projekt“). Dok u drugu tablicu stavimo lijevu stranu odabrane funkcijske zavisnosti, znači lijevu stranu zavisnosti 1) i sve ostale atribute koji nisu u prvoj

tablici iz zavisnosti koju provjeravamo („id_projekt“, „id_zaposlenik“ → „naziv_projekta“, „id_odjel“, „naziv_odjela“, „grad“, „ime“, „prezime“), primarni ključ će biti lijeva strana te zavisnosti („id_projekt“, „id_zaposlenik“), što možemo vidjeti u tablici 25.

Tablica 24: Normalizacija 1

id_projekt	naziv_projekta	id_odjel	naziv_odjela	grad
101	P1	10	O1	Varaždin
202	P2	15	O2	Zagreb

Tablica 25: Normalizacija 2

id_zaposlenik	ime	prezime	id_projekt
1	Luka	Erceg	101
2	Damjan	Poljak	101
3	Bojan	Kavur	101
4	Matija	Horvat	202
5	Karlo	Kos	202
3	Bojan	Kavur	202

Nakon dekompozicije imamo dvije tablice. Za prvu tablicu (Tablica 24 – Normalizacija 1) sada vrijede slijedeće funkcijske zavisnosti:

- „id_projekt“ → „naziv_projekta“, „id_odjel“
- „id_odjel“ → „naziv_odjela“, „grad“

S obzirom na to da je primarni ključ „id_projekt“, ova tablica zadovoljava 2NF. Što se tiče druge tablice (Tablica 25 – Normalizacija 2) vrijedi slijedeća funkcijska zavisnost:

- „id_zaposlenik“ → „ime“, „prezime“

Ako malo bolje promotrimo tablicu 25, možemo uočiti da se pojavljuju redundantne vrijednosti i time već možemo zaključiti da je potrebna normalizacija. Vratimo se na 2NF, znači primarni ključ tablice 25 je („id_projekt“, „id_zaposlenik“) i potrebno je provjeriti ovise li neključni atributi o ključu, tj. sada provjeravamo je li zavisnost „id_projekt“, „id_zaposlenik“ → „ime“, „prezime“ parcijalna. Temeljem funkcijske zavisnosti koja vrijedi u tablici možemo zaključiti da

je zavisnost parcijalna. S obzirom na to da atributi „ime“ i „prezime“ parcijalno ovise o ključu „id_projekt“, „id_zaposlenik“, tablica nije u 2NF i potrebno je napraviti dekompoziciju.

Radimo ju na isti način budući da vrijedi samo jedna zavisnost u tablici, uzmemo ju i u prvu tablicu stavimo sve iz odabrane zavisnosti, tablica 26, i primarni ključ će činiti lijeva strana odabrane zavisnosti („id_zaposlenik“). Dok u drugu tablicu stavimo lijevu stranu funkcijske zavisnosti koju smo uzeli i sve ostale atribute koji nisu u prvoj tablici iz zavisnosti koju provjeravamo („id_projekt“, „id_zaposlenik“ → „ime“, „prezime“), primarni ključ će biti lijeva strana te zavisnosti („id_projekt“, „id_zaposlenik“), što možemo vidjeti u tablici 27.

Tablica 26: Normalizacija 3

id_zaposlenik	ime	prezime
1	Luka	Erceg
2	Damjan	Poljak
3	Bojan	Kavur
4	Matija	Horvat
5	Karlo	Kos

Tablica 27: Normalizacija 4

id_projekt	id_zaposlenik
101	1
101	2
101	3
202	4
202	5
202	3

Nakon dekompozicije početne tablice dobili smo tri tablice koje sada zadovoljavaju 2NF. Prva od tri je tablica 24 – Normalizacija 1 za koju su već navedene funkcijske zavisnosti koje u njoj vrijede. Iduća tablica je tablica 26 – Normalizacija 3 u kojoj je primarni ključ „id_zaposlenik“ i vrijedi slijedeća funkcijska zavisnost:

- „id_zaposlenik“ → „ime“, „prezime“

Posljednja od tri tablice je tablica 27 – Normalizacija 4 u kojoj oba stupca („id_projekt“, „id_zaposlenik“) čine složeni primarni ključ i u toj tablici nema funkcijskih zavisnosti. Nakon 2NF slijedi provjera zadovoljavaju li dobivene tablice 3NF, a to je da ne postoji tranzitivna zavisnost neključnog atributa od ključa. Tranzitivna zavisnost, kao što sam i ranije naveo, postoji u prvoj tablici („id_projekt“ → …, „id_odjel“ → „naziv_odjela“, …). Naime, ključ je „id_projekt“, a neključni atributi „naziv_odjela“ i „grad“ tranzitivno ovise o ključu „id_projekt“. Stoga zaključujemo da tablica nije u 3NF i potrebna je dekompozicija. Zbog funkcijske zavisnosti „id_odjel“ → „naziv_odjela“, „grad“, imamo tranzitivnu zavisnost i potrebno je napraviti dekompoziciju po toj zavisnosti.

Dekompozicija se radi na isti način kao i do sada. Uzmemo zavisnost „id_odjel“ → „naziv_odjela „, „grad“ i sve atribute koji se u njoj nalaze stavimo u prvu tablicu čiji će primarni ključ biti lijeva strana zavisnosti koju smo uzeli („id_odjel“), što možemo vidjeti u tablici 28. U drugu tablicu stavimo lijevu stranu zavisnosti koju smo uzeli i sve ostale atribute koji nisu u prvoj tablici i primarni ključ će biti lijeva strana zavisnosti po kojoj dekomponiramo („id_projekt“ → „naziv_projekta“, „id_odjel“, „naziv_odjela“, „grad“), što vidimo u tablici 29.

Tablica 28: Normalizacija 5

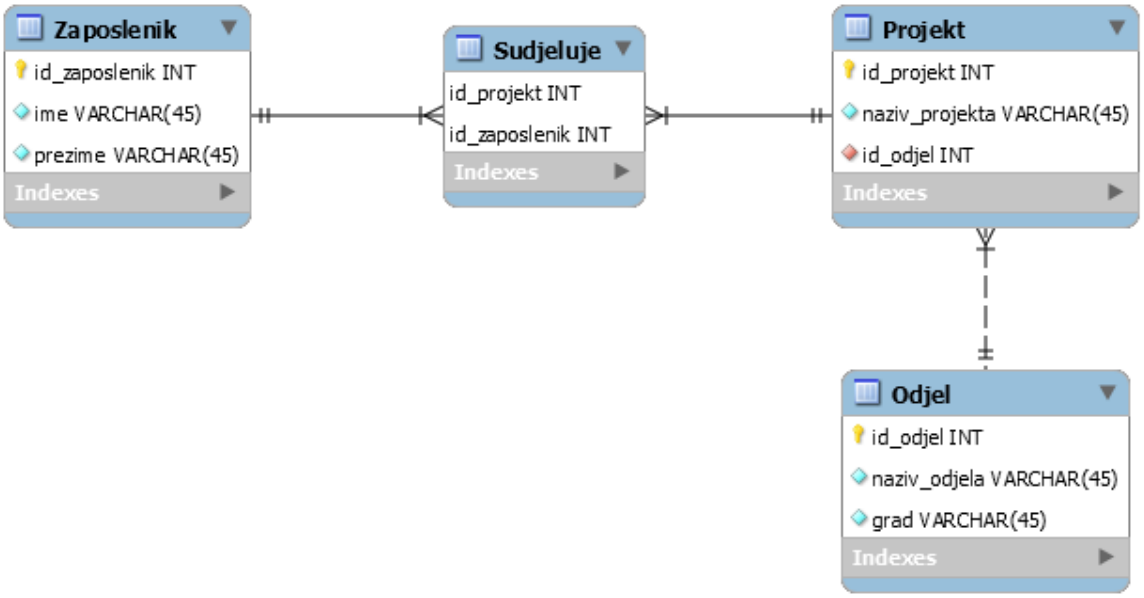
id_odjel	naziv_odjela	grad
10	O1	Varaždin
15	O2	Zagreb

Tablica 29: Normalizacija 6

id_projekt	naziv_projekta	id_odjel
101	P1	10
202	P2	15

Nakon dekompozicije imamo četiri tablice. Prva je vidljiva na tablici 26 i posjeduje samo jednu funkcijsku zavisnost: „id_zaposlenik“ → „ime“, „prezime“. Druga tablica vidljiva na tablici 27 ne posjeduje funkcijske zavisnosti jer oba atributa čine primarni ključ koji je složen. Treća tablica vidljiva na tablici 28 također posjeduje samo jednu funkcijsku zavisnost: „id_odjel“ → „naziv_odjela“, „grad“. Posljednja tablica vidljiva na tablici 29 isto posjeduje samo jednu

funkcijsku zavisnost: „id_projekt“ → „naziv_projekta“, „id_odjel“. Prema tome, sve tablice su u 3NF budući da svaka netrivijalna funkcijska zavisnost posjeduje svojstvo da lijeva strana sadrži ključ te zaključujemo da zadovoljavaju i BCNF. S obzirom na to da ne postoje višeznačne zavisnosti, tada nema potrebe provjeravati više normalne forme, ali možemo vidjeti da i njih zadovoljavaju. Na slici 19 možemo vidjeti kako izgleda ERA model prethodnog primjera nakon normalizacije.



Slika 19: Normalizacija [vlastita izrada]

5. Transformacija konceptualnog modela u pripadnu shemu relacijske baze podataka

U ovom poglavlju prvo ću opisati aplikacijsku domenu koja će biti fast food. Nakon opisa aplikacijske domene slijedi izrada konceptualnog modela koji se sastoji od povezanih koncepata (entiteta). Konceptualni model ću izraditi u alatu Visual Paradigm Online i bit će prikazan kao ER (entity – relationship, u prijevodu entiteti – veze) dijagram ili model. Potom ću izraditi ERA (entity – relationship – attribute, u prijevodu entiteti – veze – atributi) model. Do sada sam na prethodnim primjerima za izradu ERA modela koristio MySQL Workbench pa da vidimo kako to izgleda i u drugim alatima, koristit ću alat MS SQL Server Management Studio. Nakon izrade ERA modela slijedi normalizacija do BCNF i implementacija u SQL Serveru kako bi mogli vidjeti i SQL kod potreban za izradu modela. Krenimo s opisom aplikacijske domene.

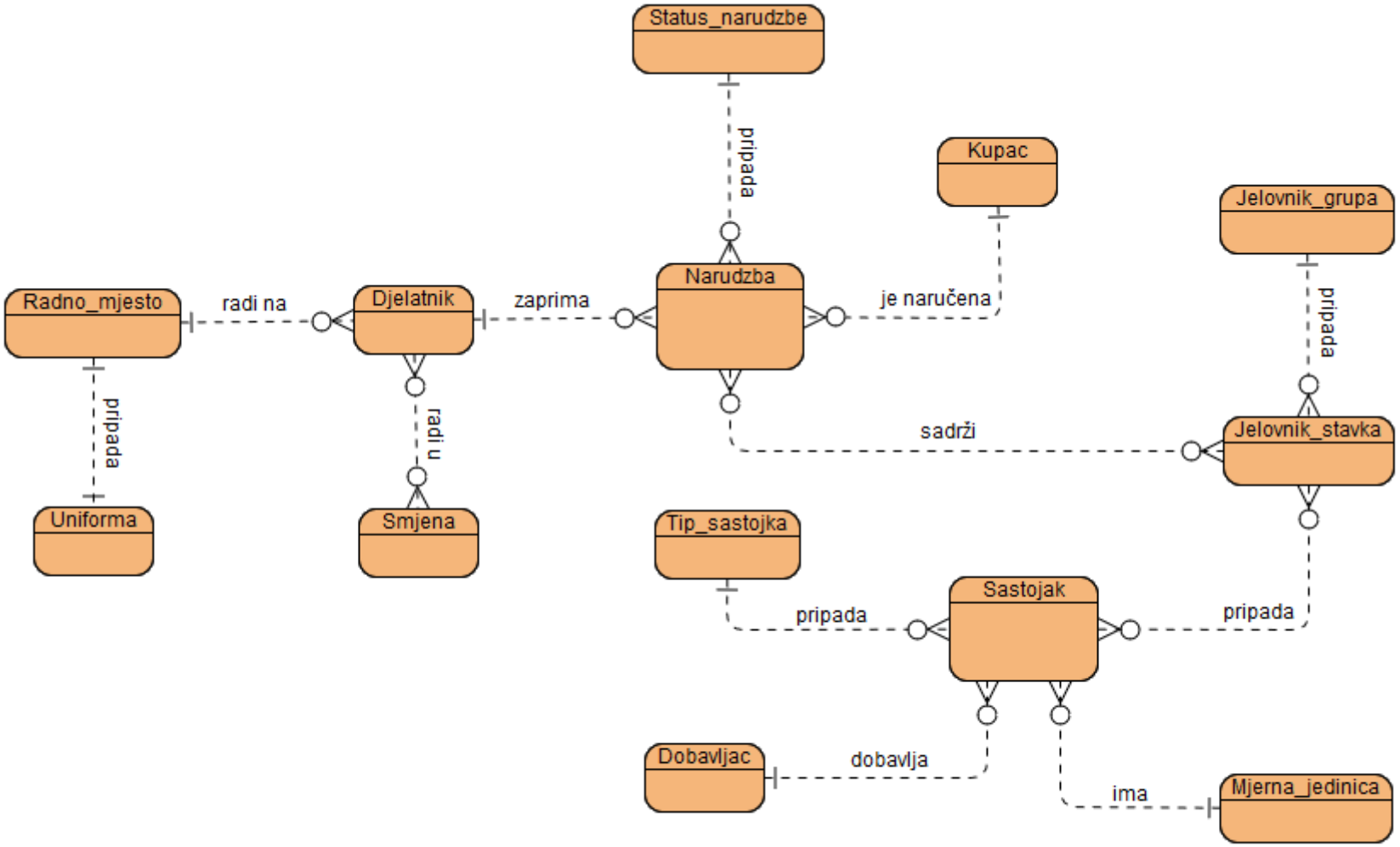
5.1. Opis aplikacijske domene za fast food

Sustav omogućuje pohranu podataka o djelatnicima i njihovu evidenciju. Postoji više uloga, tj. radnih mjesta i stoga vodimo evidenciju o tome koji djelatnik radi na pojedinom radnom mjestu. Također neka radna mjesta zahtijevaju uniformu koju će djelatnik morati nositi. Isto tako postoje smjene u kojima djelatnici mogu raditi i potrebno je voditi evidenciju o tome tko radi u kojoj smjeni pojedini radni dan.

Potrebno je imati jelovnik s popisom hrane koju kupci ili gosti mogu naručiti. Svaka grupa jelovnika, kao što su pizze, sendviči, salate, …, ima svoje stavke. Pojedina stavka jelovnika se priprema od određenih sastojaka koje je također potrebno evidentirati. Sastojke, radi lakšeg snalaženja, svrstavamo u kategorije prema tipu sastojka kao što su meso, mliječni proizvodi, povrće, … Nadalje, svaki sastojak se mjeri na drugi način i stoga je potrebno imati mjerne jedinice kako bi bilo lakše obnoviti zalihe pojedinog sastojka kada je to potrebno. Da bi obnovili zalihe, moramo imati i evidenciju dobavljača za sastojke.

Iduća vrlo bitna stvar je da se u sustavu evidentiraju zaprimljene narudžbe te je potreban status kako bi lakše pratili u kojoj je fazi narudžba, npr. u pripremi, isporučena, otkazana, … Posljednja važna stvar za evidenciju su kupci. Ukoliko se narudžba dostavlja, potrebno je znati neke podatke o kupcu kao što su ime, prezime, adresa. Naravno, adresa nije potrebna ukoliko kupac želi sam preuzetni narudžbu.

Iz ovog kratkog opisa potrebno je za početak prepoznati entitete i veze kako bi mogli izraditi konceptualni model kojeg možemo vidjeti na slici 20.



Slika 20: ER model (konceptualni model) za fast food [vlastita izrada]

ER model sa slike 20 ili konceptualni model prikazuje entitete i veze koji su potrebni za izradu baze podataka za fast food. Što se tiče veza, možemo primijetiti da postoje sve kardinalnosti: 1:1, 1:M i M:N. Slijedi popis veza i opis odnosa između entiteta.

Veza jedan naprema jedan:

1) Radno_mjesto – Uniforma; jednom radnom mjestu pripada najviše jedna uniforma, dok jedna uniforma pripada samo jednom radnom mjestu, 1:1

Veze jedan naprema više:

1) Radno_mjesto – Djelatnik; na jednom radnom mjestu može raditi više djelatnika, dok jedan djelatnik radi na samo jednom radnom mjestu, 1:M
2) Djelatnik – Narudzba; jedan djelatnik može zaprimiti više narudžbi, dok jednu narudžbu zaprima samo jedan djelatnik, 1:M
3) Status_narudzbe – Narudzba; jedan status može pripadati više narudžbi, dok jednoj narudžbi pripada samo jedan status, 1:M
4) Kupac – Narudzba; jedan kupac može naručiti više narudžbi, dok je jedna narudžba naručena od strane samo jednog kupca, 1:M
5) Jelovnik_grupa – Jelovnik_stavka; jednoj grupi jelovnika može pripadati više stavaka jelovnika, dok jedna stavka jelovnika pripada samo jednoj grupi jelovnika, 1:M
6) Tip_sastojka – Sastojak; jednom tipu sastojka može pripadati više sastojaka, dok jedan sastojak pripada samo jednom tipu sastojka, 1:M
7) Mjerna_jedinica – Sastojak; jednu mjernu jedinicu može imati više sastojaka, dok jedan sastojak ima samo jednu mjernu jedinicu, 1:M
8) Dobavljac – Sastojak; jedan dobavljač može dobavljati više sastojaka, dok jedan sastojak dobavlja samo jedan dobavljač, 1:M

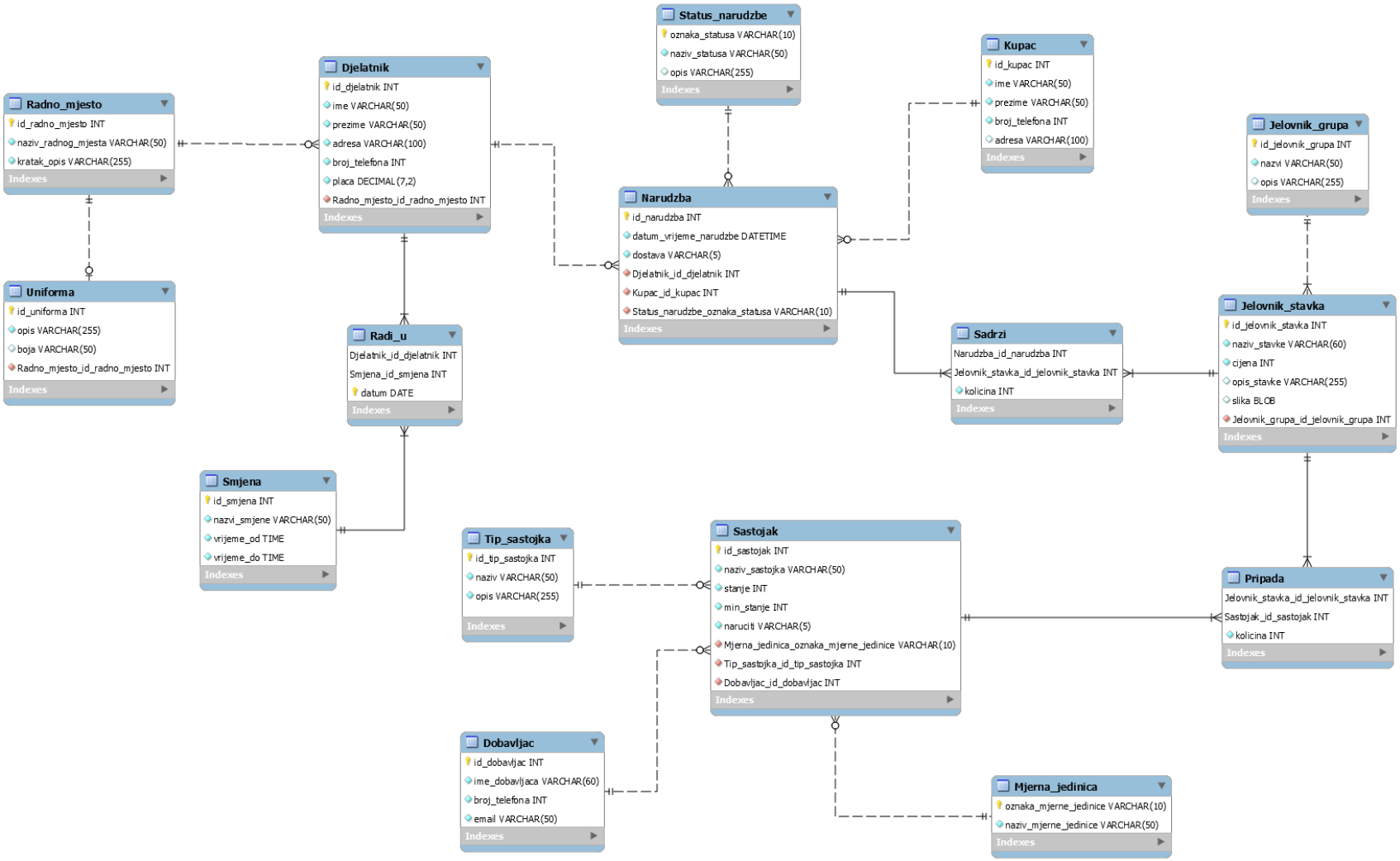
Veze više naprema više:

1) Djelatnik – Smjena; jedan djelatnik može raditi u više smjena i u jednoj smjeni može raditi više djelatnika, M:N
2) Narudzba – Jelovnik_stavka; jedna narudžba može sadržavati više stavaka jelovnika i jedna stavka jelovnika može biti sadržana u više narudžbi, M:N
3) Jelovnik_stavka – Sastojak; jednoj stavci jelovnika može pripadati više sastojaka i jedan sastojak može pripadati više stavaka jelovnika, M:N

Slijedi prikaz ERA modela na kojem možemo vidjeti sve potrebne atribute i kako su implementirane veze više naprema više dodavanjem novog slabog entiteta.

5.2. ERA model za fast food

Slika 21: ERA model za fast food – SQL Server Management Studio [vlastita izrada]



Slika 22: ERA model za fast food - MySQL Workbench [vlastita izrada]

Na slikama 21 i 22 vidimo sve atribute koje pojedini entitet sadrži te isto tako vidimo tri nova slaba entiteta koji su nastali iz veza više naprema više. Kao što sam rekao, za izradu modela koristi sam MS SQL Server Management Studio i taj model je vidljiv na slici 21, ali s obzirom na to da kardinalnost i opcionalnost nisu naznačene, odlučio sam isti primjer napraviti i u MySQL Workbench-u vidljiv na slici 22. Znači, kardinalnost se nalazi odmah uz tip entiteta, dok se opcionalnost nalazi odmah pored kardinalnosti i čine ih simboli: O koji označava nula, | koji označava 1 (jedan) i treći se naziva „vranino stopalo“ (eng. crow's-foot) koji se koristi na strani više kod veza jedan naprema više i više naprema više. Slijedi popis tablica u obliku relacijske sheme koju čini naziv tablice i njeni atributi u obliku Naziv = {atributi, …}. U popisu ću označiti primarne ključeve tako što ću nakon naziva atributa staviti (PK) (Primary Key, u prijevodu primarni ključ) i vanjske ključeve tako što ću staviti (FK) (Foreign Key, u prijevodu vanjski ključ). Ispod svake relacijske sheme napisat ću i funkcijske zavisnosti koje se u njoj nalaze. Višeznačne zavisnosti i zavisnosti spoja nisu potrebne jer se radi normalizacija do BCNF.

1) Radno_mjesto = {id_radno_mjesto (PK), naziv_radnog_mjesta, kratak_opis}
    - id_radno_mjesto → naziv_radnog_mjesta, kratak_opis
2) Uniforma = {id_uniforma (PK), opis, boja, Radno_mjesto_id_radno_mjesto (FK)}
    - id_uniforma → opis, boja, Radno_mjesto_id_radno_mjesto
3) Djelatnik = {id_djelatnik (PK), ime, prezime, adresa, broj_telefona, placa, Radno_mjesto_id_radno_mjesto (FK)}
    - id_djelatnik → ime, prezime, adresa, broj_telefona, placa, Radno_mjesto_id_radno_mjesto
4) Radi_u = {Djelatnik_id_djelatnik (PK) (FK), Smjena_id_smjena (PK) (FK), datum (PK)}
    - nema funkcijskih zavisnosti
5) Smjena = {id_smjena (PK), naziv_smjene, vrijeme_od, vrijeme_do}
    - id_smjena → naziv_smjene, vrijeme_od, vrijeme_do
6) Status_narudzbe = {oznaka_statusa (PK), naziv_statusa, opis}
    - oznaka_statusa → naziv_statusa, opis
7) Kupac = {id_kupac (PK), ime, prezime, broj_telefona, adresa}
    - id_kupac → ime, prezime, broj_telefona, adresa
8) Narudzba = {id_narudzba (PK), datum_vrijeme_narudzbe, dostava, Djelatnik_id_djelatnik (FK), Kupac_id_kupac (FK), Status_narudzbe_oznaka_statusa (FK)}
    - id_narudzba → datum_vrijeme_narudzbe, dostava, Djelatnik_id_djelatnik, Kupac_id_kupac, Status_narudzbe_oznaka_statusa

9) Sadrzi = {Narudzba_id_narudzba (PK) (FK), Jelovnik_stavka_id_jelovnik_stavka (PK) (FK), kolicina}
    - Narudzba_id_narudzba, Jelovnik_stavka_id_jelovnik_stavka → kolicina
10) Jelovnik_stavka = {id_jelovnik_stavka (PK), naziv_stavke, cijena, opis_stavke, slika, Jelovnik_grupa_id_jelovnik_grupa (FK)}
    - id_jelovnik_stavka → naziv_stavke, cijena, opis_stavke, slika, Jelovnik_grupa_id_jelovnik_grupa
11) Jelovnik_grupa = {id_jelovnik_grupa (PK), naziv, opis}
    - id_jelovnik_grupa → naziv, opis
12) Pripada = {Jelovnik_stavka_id_jelovnik_stavka (PK) (FK), Sastojak_id_sastojak (PK) (FK), kolicina}
    - Jelovnik_stavka_id_jelovnik_stavka, Sastojak_id_sastojak → kolicina
13) Sastojak = {id_sastojak (PK), naziv_sastojka, stanje, min_stanje, naruciti, Mjerna_jedinica_oznaka_mjerne_jedinice (FK), Tip_sastojka_id_tip_sastojka (FK), Dobavljac_id_dobavljac (FK)}
    - id_sastojak → naziv_sastojka, stanje, min_stanje, naruciti, Mjerna_jedinica_oznaka_mjerne_jedinice, Tip_sastojka_id_tip_sastojka, Dobavljac_id_dobavljac
14) Mjerna_jedinica = {oznaka_mjerne_jedinice (PK), naziv_mjerne_jedinice}
    - oznaka_mjerne_jedinice → naziv_mjerne_jedinice
15) Tip_sastojka = {id_tip_sastojka (PK), naziv, opis}
    - id_tip_sastojka → naziv, opis
16) Dobavljac = {id_dobavljac (PK), ime_dobavljaca, broj_telefona, email}
    - id_dobavljac → ime_dobavljaca, broj_telefona, email

Ostala je još normalizacija do BCNF i implementacija. Što se tiče normalizacije, možemo vidjeti da su svi atributi elementarni, stoga sve tablice zadovoljavaju 1NF. Nadalje, iz popisa zavisnosti možemo vidjeti da ne postoji parcijalna zavisnost neključnog atributa od ključa, što znači da tablice zadovoljavaju i 2NF. Isto tako iz popisa zavisnosti vidimo da ne postoji tranzitivna zavisnost neključnog atributa od ključa i zaključujemo da tablice zadovoljavaju 3NF. Na kraju vidimo da se s lijeve strane svake netrivijalne funkcijske zavisnosti nalazi primarni ključ, što znači da tablice zadovoljavaju i BCNF. S obzirom na to da tablice zadovoljavaju 1NF, 2NF, 3NF i BCNF, normalizacija nije potrebna. Ovo je primjer u kojem se tokom izgradnje konceptualnog modela dobro promislilo o entitetima, vezama i atributima i kao rezultat imamo bazu koja je već normalizirana.

5.3. Implementacija baze podataka za fast food u SQL Serveru

Sav kod dobiven je korištenjem opcije „Generate Scripts…“ te je prilagođen kako bi bio čitljiviji i pregledniji. Ta opcija je u drugim alatima poznatija pod nazivom „Forward engineering“ i služi kako bi se iz dijagrama generirao SQL kod potreban za kreiranje tablica. Suprotna opcija je „Reverse engineering“ koja služi kako bi se izradio dijagram iz postojeće relacijske sheme baze podataka. Slijedi kod kojim možemo prethodno objašnjenu relacijsku shemu implementirati u sustavu za upravljanje bazom podataka. Odabrani sustav je SQL Server.

```
CREATE TABLE [Radno_mjesto](
	[id_radno_mjesto] [int] NOT NULL IDENTITY(1,1),
	[naziv_radnog_mjesta] [varchar](50) NOT NULL,
	[opis] [varchar](255) NOT NULL,
 CONSTRAINT [PK_Radno_mjesto] PRIMARY KEY
(
	[id_radno_mjesto]
))

CREATE TABLE [Uniforma](
	[id_uniforma] [int] NOT NULL IDENTITY(1,1),
	[opis] [varchar](255) NOT NULL,
	[boja] [varchar](50) NULL,
	[Radno_mjesto_id_radno_mjesto] [int] NOT NULL,
 CONSTRAINT [PK_Uniforma] PRIMARY KEY
(
	[id_uniforma]
),
 CONSTRAINT [Unique_Uniforma] UNIQUE
(
	[Radno_mjesto_id_radno_mjesto]
),
CONSTRAINT [FK_Uniforma_Radno_mjesto] FOREIGN
KEY([Radno_mjesto_id_radno_mjesto])
REFERENCES [Radno_mjesto] ([id_radno_mjesto])
ON UPDATE CASCADE ON DELETE NO ACTION
)
```

```
CREATE TABLE [Djelatnik](
	[id_djelatnik] [int] NOT NULL IDENTITY(1,1),
	[ime] [varchar](50) NOT NULL,
	[prezime] [varchar](50) NOT NULL,
	[adresa] [varchar](100) NOT NULL,
	[broj_telefona] [int] NOT NULL,
	[placa] [decimal](7, 2) NOT NULL,
	[Radno_mjesto_id_radno_mjesto] [int] NOT NULL,
 CONSTRAINT [PK_Djelatnik] PRIMARY KEY
(
	[id_djelatnik]
),
CONSTRAINT [FK_Djelatnik_Radno_mjesto] FOREIGN
KEY([Radno_mjesto_id_radno_mjesto])
REFERENCES [Radno_mjesto] ([id_radno_mjesto])
ON UPDATE CASCADE ON DELETE NO ACTION
)

CREATE TABLE [Smjena](
	[id_smjena] [int] NOT NULL IDENTITY(1,1),
	[naziv_smjene] [varchar](50) NOT NULL,
	[vrijeme_od] [time](0) NOT NULL,
	[vrijeme_do] [time](0) NOT NULL,
 CONSTRAINT [PK_Smjena] PRIMARY KEY
(
	[id_smjena]
))

CREATE TABLE [Radi_u](
	[Djelatnik_id_djelatnik] [int] NOT NULL,
	[Smjena_id_smjena] [int] NOT NULL,
	[datum] [date] NOT NULL,
 CONSTRAINT [PK_Radi_u] PRIMARY KEY
(
	[Djelatnik_id_djelatnik],
	[Smjena_id_smjena],
	[datum]
),
CONSTRAINT [FK_Radi_u_Djelatnik] FOREIGN KEY([Djelatnik_id_djelatnik])
REFERENCES [Djelatnik] ([id_djelatnik])
ON UPDATE CASCADE ON DELETE NO ACTION,
```

```
CONSTRAINT [FK_Radi_u_Smjena] FOREIGN KEY([Smjena_id_smjena])
REFERENCES [Smjena] ([id_smjena])
ON UPDATE CASCADE ON DELETE NO ACTION
)

CREATE TABLE [Kupac](
      [id_kupac] [int] NOT NULL IDENTITY(1,1),
      [ime] [varchar](50) NOT NULL,
      [prezime] [varchar](50) NOT NULL,
      [broj_telefona] [int] NOT NULL,
      [adresa] [varchar](100) NULL,
 CONSTRAINT [PK_Kupac] PRIMARY KEY
(
      [id_kupac]
))

CREATE TABLE [Status_narudzbe](
      [oznaka_statusa] [varchar](10) NOT NULL,
      [naziv_statusa] [varchar](50) NOT NULL,
      [opis] [varchar](255) NULL,
 CONSTRAINT [PK_Status_narudzbe] PRIMARY KEY
(
      [oznaka_statusa]
))

CREATE TABLE [Narudzba](
      [id_narudzba] [int] NOT NULL IDENTITY(1,1),
      [datum_vrijeme_narudzbe] [datetime] NOT NULL,
      [dostava] [varchar](5) NOT NULL,
      [Djelatnik_id_djelatnik] [int] NOT NULL,
      [Kupac_id_kupac] [int] NOT NULL,
      [Status_narudzbe_oznaka_statusa] [varchar](10) NOT NULL,
 CONSTRAINT [PK_Narudzba] PRIMARY KEY
(
      [id_narudzba]
),
CONSTRAINT [FK_Narudzba_Djelatnik] FOREIGN KEY([Djelatnik_id_djelatnik])
REFERENCES [Djelatnik] ([id_djelatnik])
ON UPDATE CASCADE ON DELETE NO ACTION,
CONSTRAINT [FK_Narudzba_Kupac] FOREIGN KEY([Kupac_id_kupac])
```

```
REFERENCES [Kupac] ([id_kupac])
ON UPDATE CASCADE,
CONSTRAINT [FK_Narudzba_Status_narudzbe] FOREIGN
KEY([Status_narudzbe_oznaka_statusa])
REFERENCES [Status_narudzbe] ([oznaka_statusa])
ON UPDATE CASCADE ON DELETE NO ACTION
)

CREATE TABLE [Jelovnik_grupa](
      [id_jelovnik_grupa] [int] NOT NULL IDENTITY(1,1),
      [naziv] [varchar](50) NOT NULL,
      [opis] [varchar](255) NULL,
 CONSTRAINT [PK_Jelovnik_grupa] PRIMARY KEY
(
      [id_jelovnik_grupa]
))

CREATE TABLE [Jelovnik_stavka](
      [id_jelovnik_stavka] [int] NOT NULL IDENTITY(1,1),
      [naziv_stavke] [varchar](60) NOT NULL,
      [cijena] [int] NOT NULL,
      [opis_stavke] [varchar](255) NULL,
      [slika] [image] NULL,
      [Jelovnik_grupa_id_jelovnik_grupa] [int] NOT NULL,
 CONSTRAINT [PK_Jelovnik_stavka] PRIMARY KEY
(
      [id_jelovnik_stavka]
),
CONSTRAINT [FK_Jelovnik_stavka_Jelovnik_grupa] FOREIGN
KEY([Jelovnik_grupa_id_jelovnik_grupa])
REFERENCES [Jelovnik_grupa] ([id_jelovnik_grupa])
ON UPDATE CASCADE ON DELETE NO ACTION
)

CREATE TABLE [Sadrzi](
      [Narudzba_id_narudzba] [int] NOT NULL,
      [Jelovnik_stavka_id_jelovnik_stavka] [int] NOT NULL,
      [kolicina] [int] NOT NULL,
 CONSTRAINT [PK_Sadrzi] PRIMARY KEY
(
      [Narudzba_id_narudzba],
```

```
      [Jelovnik_stavka_id_jelovnik_stavka]
),
 CONSTRAINT [FK_Sadrzi_Jelovnik_stavka] FOREIGN
KEY([Jelovnik_stavka_id_jelovnik_stavka])
REFERENCES [Jelovnik_stavka] ([id_jelovnik_stavka])
ON UPDATE CASCADE ON DELETE NO ACTION,
CONSTRAINT [FK_Sadrzi_Narudzba] FOREIGN KEY([Narudzba_id_narudzba])
REFERENCES [Narudzba] ([id_narudzba])
ON UPDATE CASCADE ON DELETE NO ACTION
)

CREATE TABLE [Tip_sastojka](
      [id_tip_sastojka] [int] NOT NULL IDENTITY(1,1),
      [naziv] [varchar](50) NOT NULL,
      [opis] [varchar](255) NOT NULL,
 CONSTRAINT [PK_Tip_sastojka] PRIMARY KEY
(
      [id_tip_sastojka]
))

CREATE TABLE [Dobavljac](
      [id_dobavljac] [int] NOT NULL IDENTITY(1,1),
      [ime_dobavljaca] [varchar](60) NOT NULL,
      [broj_telefona] [int] NOT NULL,
      [email] [varchar](50) NOT NULL,
 CONSTRAINT [PK_Dobavljac] PRIMARY KEY
(
      [id_dobavljac]
))

CREATE TABLE [Mjerna_jedinica](
      [oznaka_mjerne_jedinice] [varchar](10) NOT NULL,
      [naziv_mjerne_jedinice] [varchar](50) NOT NULL,
 CONSTRAINT [PK_Mjerna_jedinica] PRIMARY KEY
(
      [oznaka_mjerne_jedinice]
))

CREATE TABLE [Sastojak](
      [id_sastojak] [int] NOT NULL IDENTITY(1,1),
      [naziv_sastojka] [varchar](50) NOT NULL,
```

```
	[stanje] [int] NOT NULL,
	[min_stanje] [int] NOT NULL,
	[naruciti] [varchar](5) NOT NULL,
	[Mjerna_jedinica_oznaka_mjerne_jedinice] [varchar](10) NOT NULL,
	[Tip_sastojka_id_tip_sastojka] [int] NOT NULL,
	[Dobavljac_id_dobavljac] [int] NOT NULL,
 CONSTRAINT [PK_Sastojak] PRIMARY KEY
(
	[id_sastojak]
),
CONSTRAINT [FK_Sastojak_Dobavljac] FOREIGN KEY([Dobavljac_id_dobavljac])
REFERENCES [Dobavljac] ([id_dobavljac])
ON UPDATE CASCADE ON DELETE NO ACTION,
CONSTRAINT [FK_Sastojak_Mjerna_jedinica] FOREIGN
KEY([Mjerna_jedinica_oznaka_mjerne_jedinice])
REFERENCES [Mjerna_jedinica] ([oznaka_mjerne_jedinice])
ON UPDATE CASCADE ON DELETE NO ACTION,
CONSTRAINT [FK_Sastojak_Tip_sastojka] FOREIGN
KEY([Tip_sastojka_id_tip_sastojka])
REFERENCES [Tip_sastojka] ([id_tip_sastojka])
ON UPDATE CASCADE ON DELETE NO ACTION
)

CREATE TABLE [Pripada](
	[Jelovnik_stavka_id_jelovnik_stavka] [int] NOT NULL,
	[Sastojak_id_sastojak] [int] NOT NULL,
	[kolicina] [int] NOT NULL,
 CONSTRAINT [PK_Pripada] PRIMARY KEY
(
	[Jelovnik_stavka_id_jelovnik_stavka],
	[Sastojak_id_sastojak]
),
CONSTRAINT [FK_Pripada_Jelovnik_stavka] FOREIGN
KEY([Jelovnik_stavka_id_jelovnik_stavka])
REFERENCES [Jelovnik_stavka] ([id_jelovnik_stavka])
ON UPDATE CASCADE ON DELETE NO ACTION,
CONSTRAINT [FK_Pripada_Sastojak] FOREIGN KEY([Sastojak_id_sastojak])
REFERENCES [Sastojak] ([id_sastojak])
ON UPDATE CASCADE ON DELETE NO ACTION
)
```

6. Zaključak

U ovom radu upoznali smo se s modeliranjem baze podataka. Postoji konceptualno, logičko i fizičko oblikovanje baze podataka. Konceptualno oblikovanje baze podataka je detaljno objašnjeno i temelji se na konceptima i vezama između njih. Logičko oblikovanje se bavi razmatranjem zavisnosti između atributa relacijske sheme. Za logičko oblikovanje su vrlo bitne normalne forme, od koji su za normalizaciju najvažnije 1NF, 2NF, 3NF i BCNF. Normalizacija baze podataka je vrlo važna kako bi maknuli neželjene zavisnosti iz baze te tako bazu učinili bržom, pouzdanijom i sigurnijom. Kroz primjer smo vidjeli kako se provodi postupak normalizacije te smo isto tako kroz primjere lakše shvatili normalne forme, što je entitet, atribut, veza ili asocijacija i razni tipovi veza.

Na kraju sam napravio primjer transformacije konceptualnog modela u pripadnu shemu relacijske baze podataka. Odabrana aplikacijska domena je fast food i iz opisa odabrane aplikacijske domene vidjeli smo kako se formira konceptualni model (ER model) u alatu Visual Paradigm Online. Nakon toga, u alatima SQL Server Management Studio i MySQL Workbench izrađen je ERA model iz konceptualnog modela u kojem možemo vidjeti sve atribute koji su potrebni za opis pojedinog entiteta (tablice). Isto tako na tom primjeru vidimo da, ukoliko uložimo više vremena i dobro promislimo prilikom izrade konceptualnog modela, dobijemo relacijsku shemu koja je već normalizirana (BCNF). Posljednje je napravljena implementacija normalizirane relacijske sheme u sustavu SQL Server. Implementaciju sam napravio na način da sam koristio opciju „Generate Scripts…“ u alatu SQL Server Management Studio i tako iz ERA modela dobio SQL kod koji sam malo prilagodio da bude čitljiviji u samom radu. Time smo dobili fizičku shemu baze podataka koja se sastoji od SQL koda potrebnog za izgradnju baze podataka u nekom sustavu za upravljanje bazom podataka. U ovom slučaju, taj sustav je SQL Server.

Smatram da je ova tema vrlo zanimljiva. Upoznao sam se sa cijelim procesom oblikovanja baze podataka, od proučavanja aplikacijske domene, izrade konceptualnog modela, logičkog modela pa sve do fizičkog modela, tj. SQL koda. Time sam prošao cijeli proces oblikovanja baze podataka kroz izradu primjera baze podataka za fast food. Uz to, postoje još i daljnji koraci bitni kod rada s relacijskim bazama podataka, kao što je za početak unos podataka u samu bazu, pisanje upita, funkcija, procedura, okidača (eng. triggera) te briga o sigurnosti baze i održavanje baze podataka, što u ovom radu nije obuhvaćeno, ali je vrlo bitno za rad s relacijskim bazama podataka.

Popis literature

[1] M. Maleković, K. Rabuzin, *Uvod u baze podataka*. Fakultet organizacije i informatike Varaždin, Sveučilište u Zagrebu, 2016.

[2] R. Manger, *Baze podataka*, Prvo izdan. ELEMENT d.o.o., Zagreb, 2012.

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

[4] Graeme C. Simsion, Graham C. Witt, *Data Modeling Essentials*. Morgan Kaufmann, 2005.

[5] Prof. dr. sc. M. Maleković, "Prezentacije s predavanja na kolegiju Baze podataka 2," 2016.

[6] K. Rabuzin, *SQL - napredne teme*. Fakultet organizacije i informatike, Sveučilište u Zagrebu, 2014.

[7] M. Maleković, M. Schatten, *Teorija i primjena baza podataka*. Sveučilište u Zagrebu Fakultet organizacije i informatike Varaždin, 2017.

[8] "1NF, 2NF, 3NF and BCNF in Database Normalization | Studytonight." [Online]. Available: https://www.studytonight.com/dbms/database-normalization.php. [Accessed: 12-Sep-2018].

[9] "Database Normalization | Normal Forms - GeeksforGeeks." [Online]. Available: https://www.geeksforgeeks.org/database-normalization-normal-forms/. [Accessed: 12-Sep-2018].

Popis slika

Popis tablica