

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Vodič za studente na kolegiju Razvoj primijenjene programske podrške

v1.1

Zagreb, 2022.

Sadržaj

1.	Uvod.....	1
2.	Dijagrami	2
2.1	Konceptualni dijagram	2
2.1.1	Notacija Crow's Foot	3
2.1.2	Chenova notacija.....	4
2.1.3	Kako najlakše sastaviti konceptualni dijagram?.....	5
2.2	Fizički dijagram.....	6
2.3	Dodatna razrada konceptualnog modela u fizički model	8
3.	Složeni prikazi podataka	11
3.1	Master-detail forma.....	11
3.2	Složeni tablični prikaz.....	12
4.	Raspodjela domene unutar tima	14
5.	Ostale važne napomene.....	16
5.1	Unos i stvaranje vrijednosti primarnog ključa	16

1. Uvod

Na RPPP-u je većina studentskih obaveza koncentrirana na projekt. Nepisano pravilo je da se obavljanje kasnijih obaveza može olakšati dobrim obavljanjem posla na početnim obavezama. Ovaj dokument je sastavljen kako biste lakše i brže mogli uroniti u kolotečinu tih obaveza.

Bitno nam je pojasniti određene pojmove koji su na RPPP-u mučili studente prijašnjih generacija. Problem su često bile ideje vezane uz dijagrame, raspodjelu domene i oblike programskih zaslona. Neke od tih ideja bi vas trebale pratiti u nastavku studiranja, pa i u vašim poslovnim karijerama.

Koncentrirajući se na sadašnjost, ovo je dokument koji bi vam trebao popuniti moguće praznine u znanju, uštediti vrijeme na nekim nedoumicama i olakšati prolazak RPPP-a. Nadamo se da će ovaj dokument u tome biti uspješan, kao i vi u polaganju RPPP-a.

RPPP Tim

2. Dijagrami

Dijagrami su praktičan medij prenošenja sadržajno opsežnih ideja, poput modela podataka. Vrlo su važan alat u razvoju softvera, do te mjere da su i određene vrste standardizirane; kao UML (eng. *Unified Modelling Language*), kojega je moguće i strojno obrađivati. Iako se dijagrami na nekim kolegijima koriste samo kao dio dokumentacije, oni u stvarnoj upotrebi mogu poslužiti za razmjenu ideja i stvaranje dogovora. To ćete imati prilike iskusiti na samom početku odrađivanja obaveza za RPPP.

Na RPPP-u se zahtijeva izrada barem dvije vrste dijagrama za opis modela podataka: konceptualnog i fizičkog. Bez ta dva dijagrama sigurno nećete moći uspješno implementirati svoje timske aplikacije. Stoga je bitno dobro odraditi taj dio posla.

2.1 Konceptualni dijagram

Konceptualnim dijagramom se pokušavaju razraditi odnosi između pojedinih elemenata u domeni. Konceptualni dijagram u pravilu može biti razumljiv i laiku. Stoga se ovakva vrsta dijagrama može koristiti i za komunikaciju s klijentima¹. Konkretno, pokušavaju se komunicirati koncepti i njihovi odnosi u domeni – to je i izvor naziva *konceptualni dijagram*.

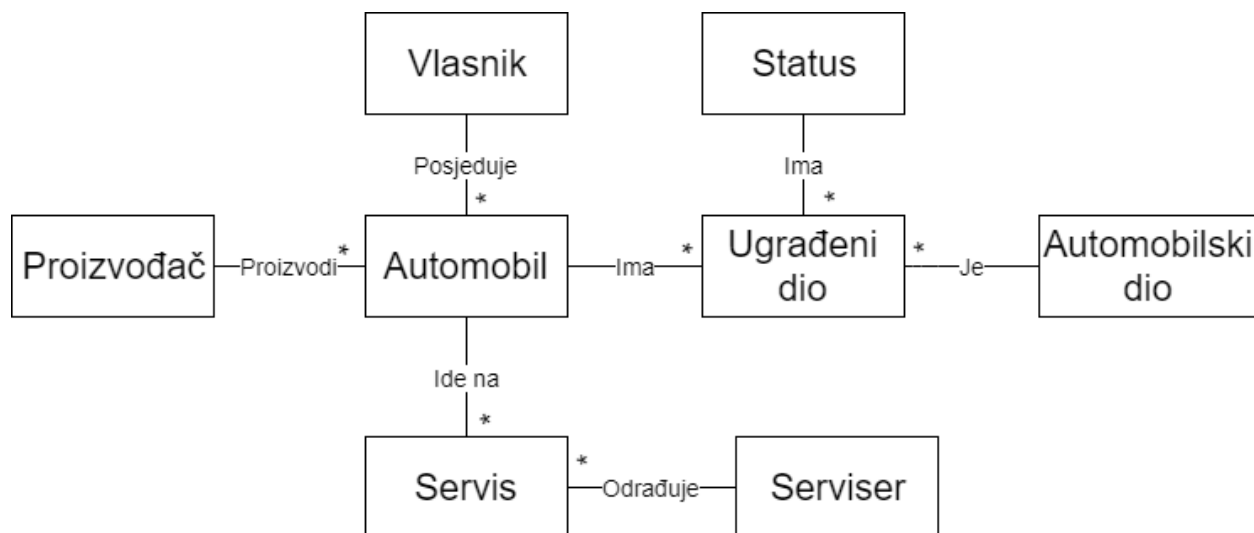
Konceptualni dijagram se može razraditi korištenjem ERD-a (*Entity Relationship Diagram*), koji bi vam u nekoj mjeri trebao biti poznat iz kolegija Baze podataka. Iako ste na tom kolegiju imali mogućnost primijeniti ERD izravno na dizajn baze podataka, važno je uočiti da konceptualnim dijagramom **ne pokušavamo odrediti dizajn sheme niti implementaciju baze podataka**; to se ostvaruje kasnije. Stoga konceptualni dijagram neće sadržavati oznake stranih niti primarnih ključeva (iako je moguće označiti važne identifikacijske attribute). U konceptualnom dijagramu nije potrebno prikazati sve attribute; mogu se prikazati samo oni važniji za tumačenje. Također, u konceptualnom dijagramu ne prikazujemo tipove podataka. Model opisan konceptualnim dijagramom trebao bi se moći primijeniti na modele različitih baza podataka².

Na RPPP-u se konceptualni dijagram sastavlja kako bi se dogovorio izgled domene nad kojom ćete graditi vaše timske aplikacije. Također, preko konceptualnog dijagrama se dogovara raspodjela domene na članove timova. O raspodjeli domene i utjecaju tog zahtjeva na vaš dizajn možete pročitati u poglavlju 4.

Jednostavni konceptualni model bi mogao izgledati kao što je prikazano na Slika 2.1. Očito je riječ o nekoj vrsti evidencije servisa automobila i automobilskih dijelova. Vlasnik može imati više automobila. Automobil pripada nekom proizvođaču. Automobil posjeduje neke standardizirane dijelove koji su mu ugrađeni, a dijelovima se može dodijeliti status ispravnosti. Također, automobil je moguće više puta poslati na servis, a servis odrađuje serviser.

¹ Dobar primjer komunikacije putem konceptualnog dijagrama možete pronaći u knjizi Erica Evansa: *Domain Driven Design*. Autor predstavlja priču iz svog iskustva kada mu je dodijeljen posao izgradnje softvera za simulaciju elektroničkog sklopovlja (o čemu autor do tada nije ništa znao). Autor je posjeo klijente pred konceptualni dijagram i razradio domenu.

² Što uključuje i nerelacijske (NoSQL) baze podataka.



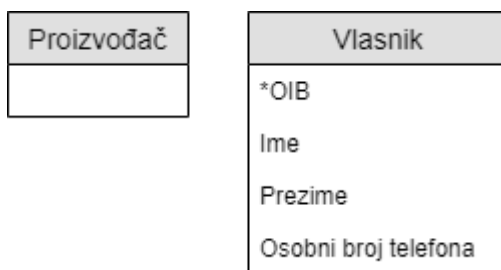
Slika 2.1 Primjer jednostavnog konceptualnog modela (nije striktno ERD)

Priloženi dijagram (Slika 2.1) nam dosta govori o postojanju određenih entiteta i njihovim odnosima, ali izostaju određeni detalji. Navedimo primjer – taj dijagram nam ne govori može li postojati takav serviser, a da nije odradio niti jedan servis. Također, dijagram nam ne govori kako identificirati i kontaktirati vlasnika ili koji su nam ključni podaci za automobil. Srećom smo odlučili koristiti ERD, te zbog potrebe prikazivanja određenih detalja možemo koristiti neku općeprihvaćenu notaciju. Najpopularnije su Crow's Foot i Chenova notacija.

2.1.1 Notacija Crow's Foot

Notacija Crow's Foot je vrlo jednostavna i intuitivna po pitanju prikazivanja entiteta i veza.

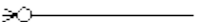



Entitet (Slika 2.2) se prikazuje kao tablica kojom je prikazan naziv entiteta i njegovi ključni atributi koje želimo trenutno istaknuti. Atribut nije obavezno prikazati, pa je moguće ostaviti samo prazno polje atributa. Također, ako je jedan atribut identifikator ili treba biti istaknut, može biti označen asteriskom (znak *).



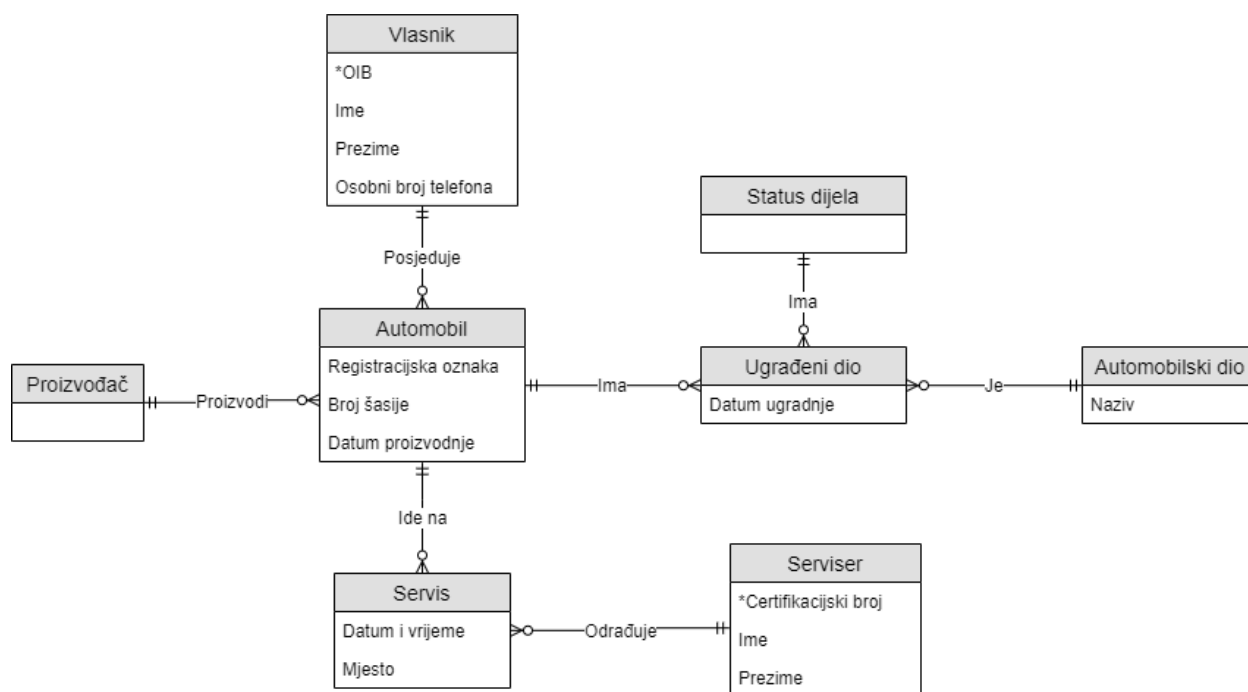
Slika 2.2 Prikaz entiteta u notaciji Crow's Foot

Prikaz brojnosti veza je ono što je dalo naziv ovoj notaciji, jer podsjeća na ptičje stopalo. Oznake brojnosti prikazane su u Tablica 2.1.

Tablica 2.1 Notacija brojnosti u notaciji Crow's Foot

Notacija	Brojnost	Opis
	0..N	Nula ili više
	1..N	Striktno jedan ili više
	1	Striktno jedan
	0..1	Nula ili jedan

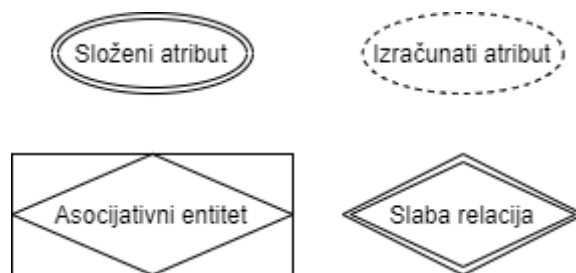
Ranije prikazani jednostavni konceptualni model (Slika 2.1) može biti detaljiziran Crow's Foot notacijom prema Slika 2.3.



Slika 2.3 Konceptualni ERD u notaciji Crow's Foot

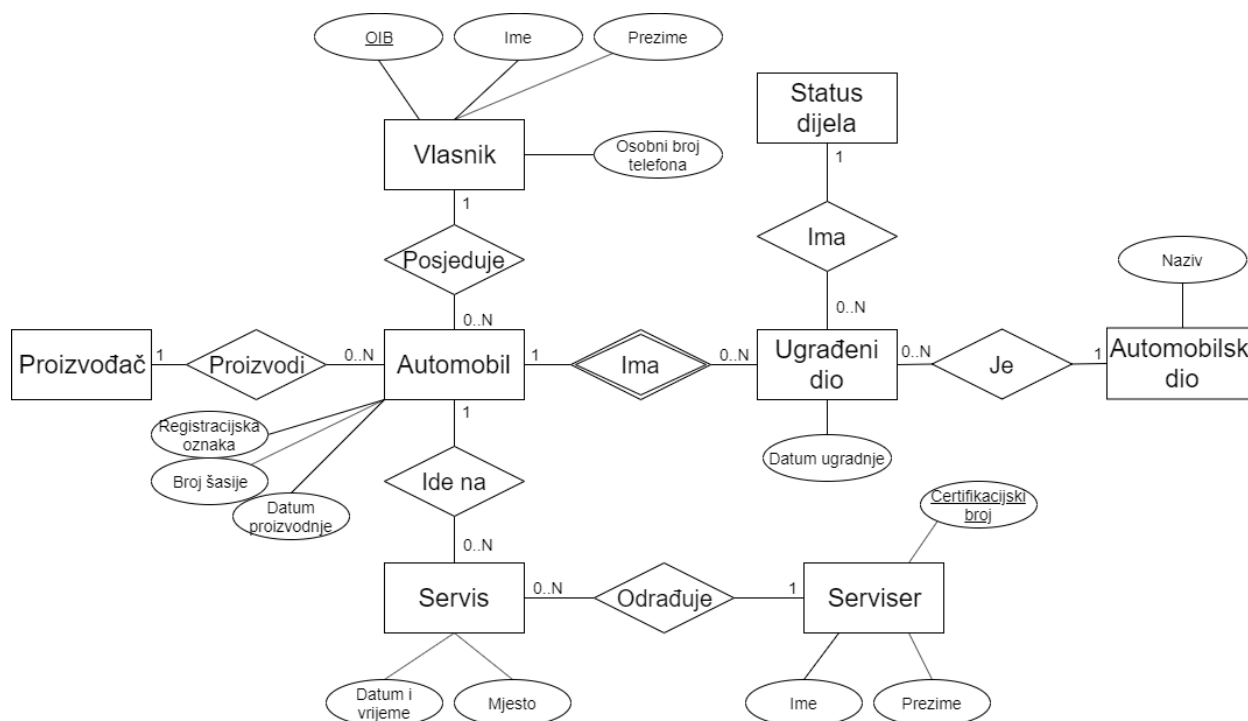
2.1.2 Chenova notacija

Chenova notacija bi vam trebala biti poznata s kolegija Baze podataka. Entiteti se prikazuju pravokutnicima, njihovi atributi se prikazuju u elipsama; gdje važnije attribute označavamo podvlakom naziva. Relacije se prikazuju rombovima, a brojnosti se prikazuju uz poveznice entiteta i relacija. Također, ništa ne sprječava korištenje specijaliziranih elemenata koji služe za bolje raščišćivanje značenja modela (Slika 2.4).



Slika 2.4 Specijalizirani elementi Chenove notacije

Na Slika 2.5 je prikazana već predstavljena domena u Chenovoj notaciji.



Slika 2.5 Konceptualni ERD u Chenovoj notaciji

2.1.3 Kako najlakše sastaviti konceptualni dijagram?

Prvo je potrebno dobro usporediti svoje zapisnike i bilješke s intervju³, te navesti sve entitete koje prepoznajete u tim tekstovima. Kako bi domena bila prihvatljive veličine za raspodjelu, pretpostavlja se da tim od petoro studenata mora imati barem 20 entiteta (koji imaju potencijal postati samostalne tablice u bazi podataka).

Identificirane entitete je zatim moguće staviti na papir ili u program za crtanje⁴ i početi određivati veze između entiteta. Možete običi svaki pojedinačni entitet i priupitati se koji su entiteti vezani uz taj odabrani

³ Primjer dobrog zapisnika s intervjua dan vam je na razmatranje u predmetnom git repozitoriju pod /Materijali/Prilozi/Zapisnik-Primjer.docx. Na istom mjestu slobodno potražite primjere i uzorke za ostale oblike dokumentacije čija se izrada zahtijeva od vas.

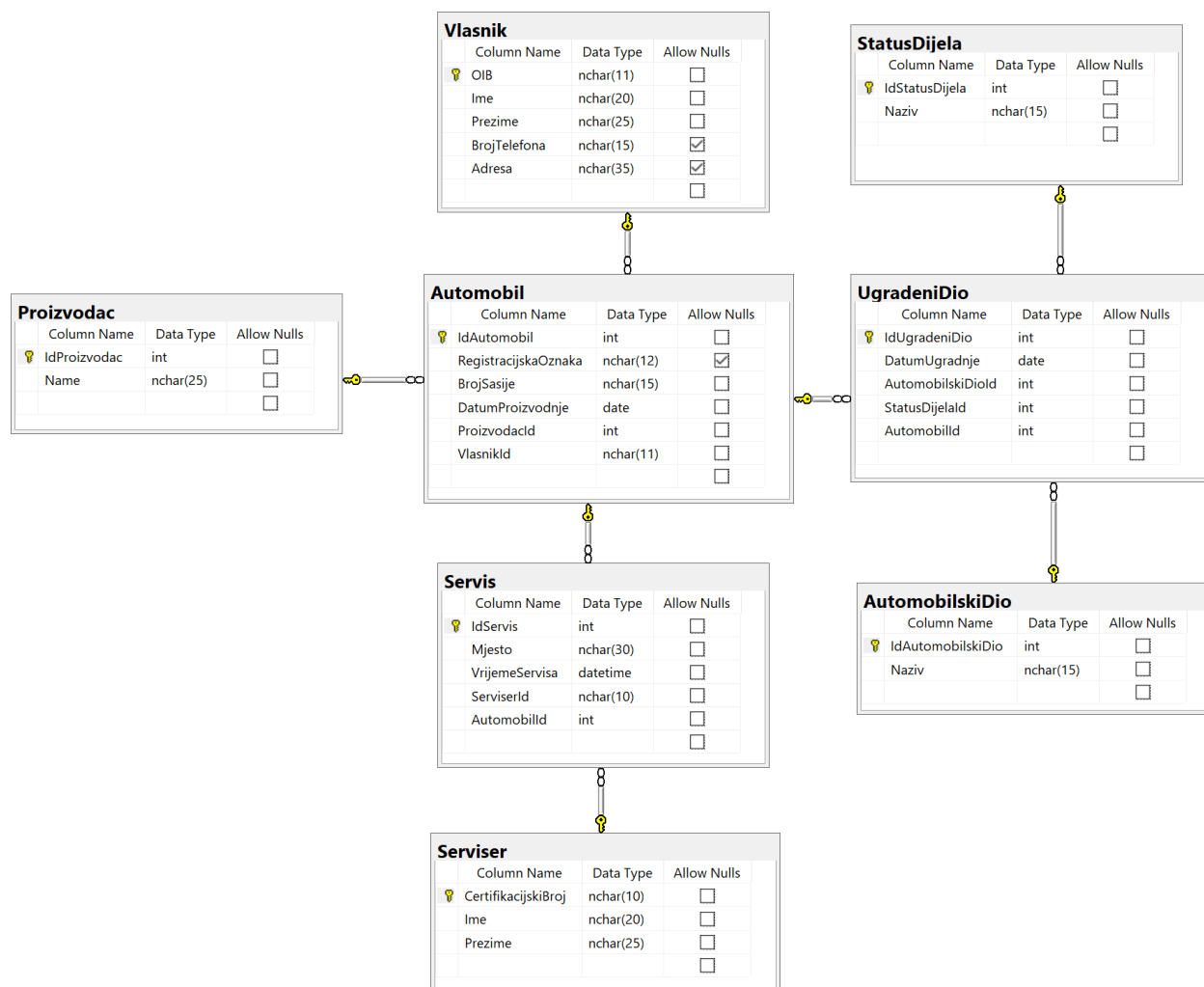
⁴ Dijagrami i crteži u ovom dokumentu nacrtani su korištenjem aplikacije draw.io, no moguće je koristiti i neki drugi program po vlastitoj volji. Bitno je da program omogućuje izvoz dijagrama u obliku slike.

entitet. Također, možete proći kroz svaki pojedinačni zapisnik ili bilješku, te upisivati veze kako se pojavljuju u tekstu. Nakon što ste prošli sav materijal, možete dodatno razraditi neke veze ili entitete. Za početak možete razraditi skicu u obliku jednostavnog konceptualnog dijagrama iz Slika 2.1.

Za formalniju razradu, trebali biste odabrati jednu od prikazanih notacija. To vam je prepušteno na izbor. Chenova notacija daje više mogućnosti semantičkog izražavanja, dok je Crow's Foot notacija sažetija i preglednija.

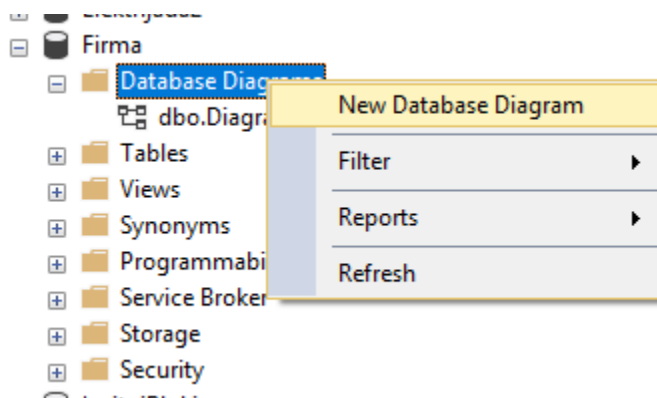
2.2 Fizički dijagram

Fizički dijagram se izrađuje tijekom i nakon izrade baze podataka. Fizički dijagram treba objasniti kako je domena konkretno implementirana u nekoj bazi podataka. Fizičkim dijagramom je bitno prikazati sve tablice, atribute, tipove podataka, primarne i strane ključeve, te određena ograničenja (*constraints*).



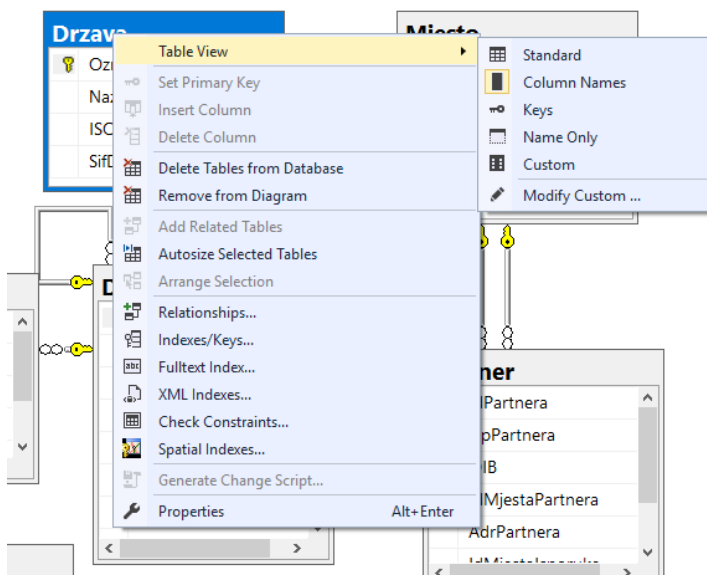
Slika 2.6 Fizički dijagram primjera

Fizički dijagram u obliku prikazanom na Slika 2.6 moguće je generirati u SQL Server Management Studio⁵ (SSMS). U SSMS-u proširite stablo svoje baze podataka, napravite desni klik na direktorij *Database Diagrams* i iz izbornika odaberite *New Database Diagram* (Slika 2.7). Potom je potrebno prevući sve željene tablice iz stabla (direktorij *Tables*) na radnu podlogu za dijagram.



Slika 2.7 Stvaranje novog fizičkog dijagrama

Desnim klikom na pojedinu tablicu dijagrama, pod stavkom *Table View*, možete pronaći mogućnosti za prikaz dodatnih podataka o tablici (tipovi podataka, *constraints* itd.).



Slika 2.8 Odabir opcije prikaza detalja tablice

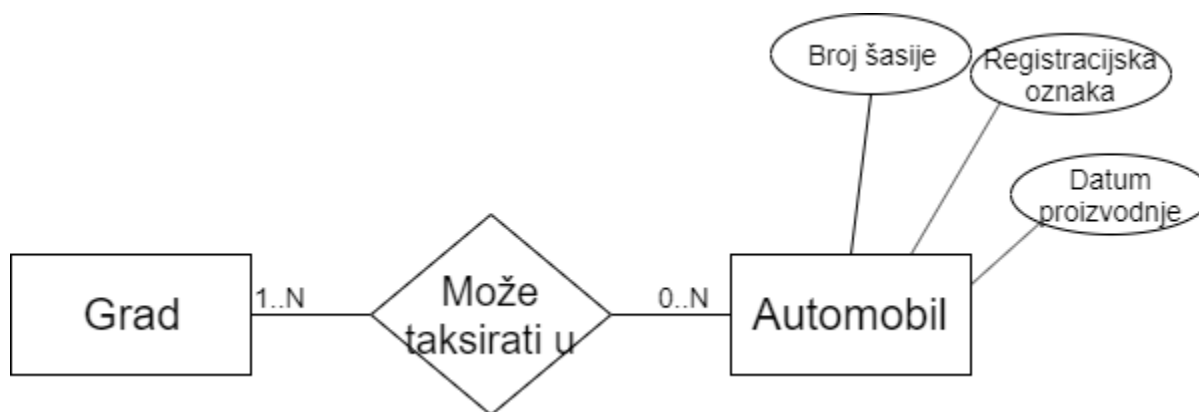
Za generiranje fizičkog modela nije potrebno koristiti SSMS, dozvoljeno je korištenje drugih alata za upravljanje bazom podataka (npr. DBeaver). Samo je bitno pripaziti da su svi relevantni podaci prisutni u dijagramu.

⁵ Napomena: U novijoj verziji SSMS-a iz 19, dolazi do pogreške pri spremanju dijagrama, pa dijagram više nije moguće otvoriti nakon što se jednom zatvori pregled. Zasad je najsigurnije koristiti SSMS 17.

2.3 Dodatna razrada konceptualnog modela u fizički model

Do sada je prikazani primjer vrlo jednostavno omogućio preslikavanje svakog entiteta u tablicu fizičkog modela. To ne mora uvijek biti slučaj! Vrlo je izvjesna mogućnost postojanja više tablica fizičkog modela nego entiteta konceptualnog modela. U tom slučaju izvor tablica mogu biti relacijski odnosi ili čak neka svojstva entiteta.

Razmotrimo proširenje modela kojim želimo nekim automobilima odrediti specijalizaciju taxija, te evidentirati u kojim gradovima ti taxiji imaju dozvolu poslovanja (Slika 2.9).



Slika 2.9 Proširenje domene evidencijom prava taksiranja

Ovo se neće moći trivijalno preslikati u fizički model. Prođimo kroz dvije mogućnosti.

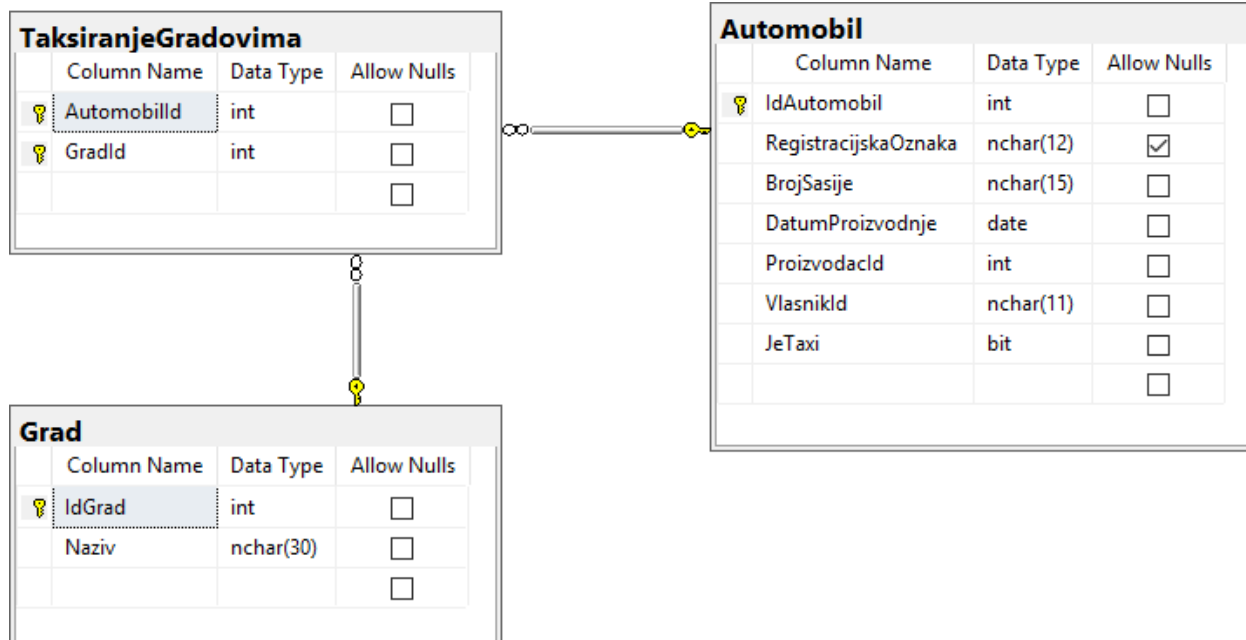
Kao i u originalnom primjeru, stvorimo tablicu automobila i proširimo je dodatnim atributom *JeTaxi* tipa *bit* (indikator istinitosti). Time ćemo određivati je li neki automobil ujedno i taxi. Tablicu *Grad* je trivijalno modelirati, jer se ona izravno preslikava u fizički model. Potom, ostvarimo N-na-N vezu stvaranjem tablice *TaksiranjeGradovima*. Rezultat tih radnji je prikazan Slika 2.10.

Ovim modeliranjem se nazire problem. Trenutno ne postoji nikakav mehanizam koji bi uveo restrikciju stvaranja N-na-N veze *automobila* prema *gradovima* prema vrijednosti atributa *JeTaxi*. Konkretno – moguće je pohraniti evidenciju u kojoj automobil koji nije označen kao taxi ima pravo taksiranja nekim gradom ili gradovima. Na ovom dijelu modela potrebno je uvesti zaštitu nekim oblikom okidača (eng. *trigger*). U fizičkom modelu ne postoji formalan način označavanja takvog ponašanja, stoga je na ovakvim mjestima potrebno ostaviti neki oblik fusnote ili bilješke.

Navedeno rješenje je ispravno, ali je dobro samo ako smo sigurni da se evidencija svojstava taksija neće proširivati⁶. Što ako bismo htjeli proširiti svojstva taksija (npr. datum početka taksi službe ili identifikacijsku oznaku taxi licence)? Dodavanje atributa očito nije prikladno proširivo rješenje, jer će se naknadnim proširenjem uzrokovati stvaranje rijetko popunjene tablice (eng. *sparse table*)⁷.

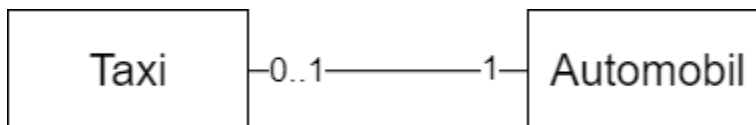
⁶ U stvarnom svijetu ovakve odluke dizajna mogu predstavljati ozbiljan rizik (u malo manjem razmjeru je to istina na RPPP-u, gdje su uvjeti kako-tako kontrolirani).

⁷ Prisjetimo se da je NULL isto podatak koji zauzima memorijski prostor. Stoga bi nam više memorijskog prostora zauzimale NULL vrijednosti nego konkretni podaci.



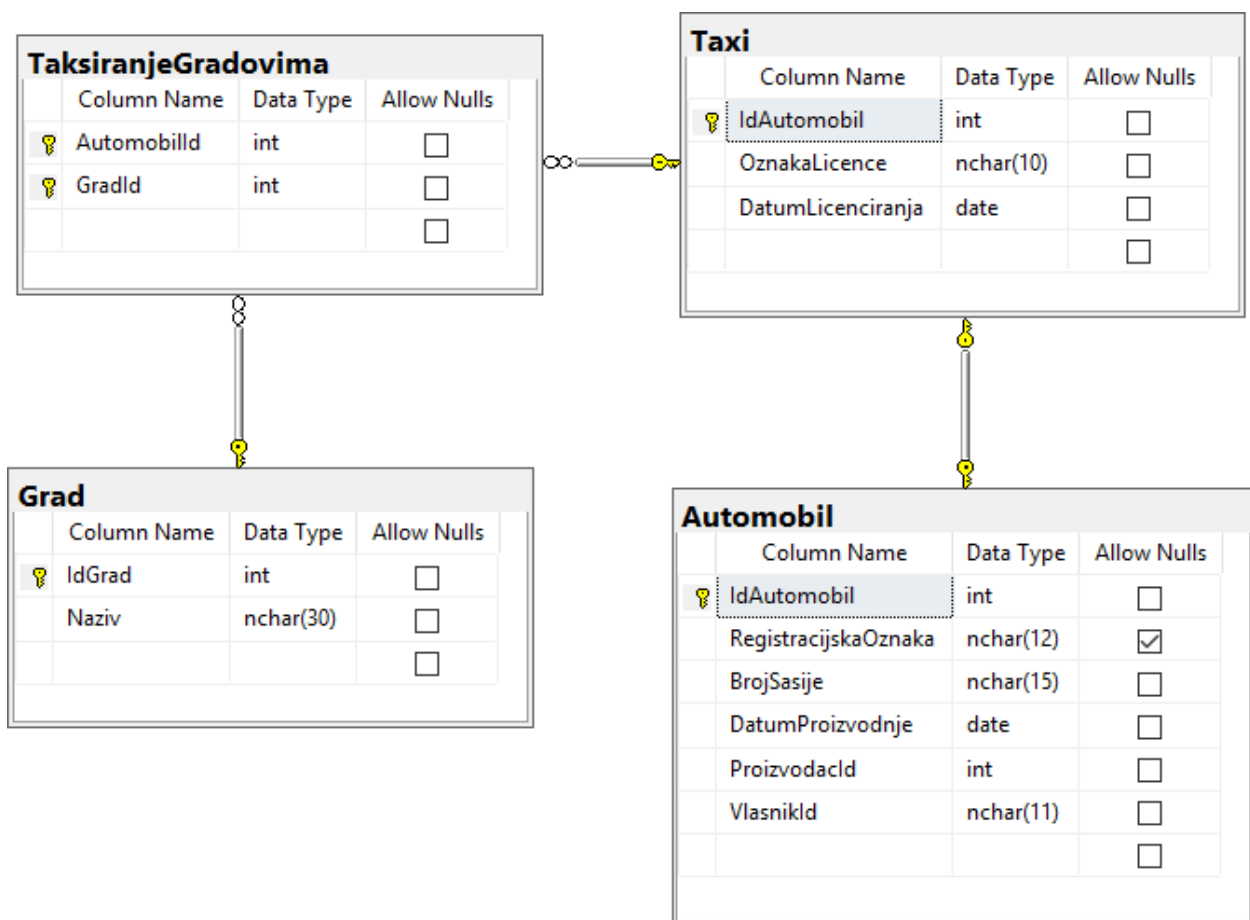
Slika 2.10 Proširenje prikazano fizičkim dijagramom

Bolje razrađeno rješenje bilo bi uvesti tablicu specijalizacije automobila – *Taxi*. Traženi logički odnos tablica *Automobil* i *Taxi* prikazan je Slika 2.11. Ovakav odnos se ostvaruje specificiranjem da je primarni ključ tablice *Taxi* ujedno i strani ključ na tablicu *Automobil*. Sprega primarnog ključa će osigurati unikatnost n-torke, dok će sprega stranog ključa osigurati da se navode samo vrijednosti postojećih automobila.



Slika 2.11 Logički odnos specijalizacijske tablice *Taxi* i osnovne tablice *Automobil*

Tablicu *Taxi* je sada moguće iskoristiti za ostvarivanje N-na-N veze prema gradovima za koje je određeni taxi licenciran. Konačan izgled tog dijela modela na fizičkom dijagramu prikazan je Slika 2.12. Ovaj oblik modeliranja je smanjio kompleksnost modela, jer je uklonio potrebu za korištenjem okidača, te je omogućeno proširivanje svojstva taxija.



Slika 2.12 Proširenje specijalizacijskom tablicom prikazano fizičkim dijagramom

3. Složeni prikazi podataka

Pozamašan dio bodova iz domaćih zadaća odnosi se na prikaze i forme za složene podatke. Složeni podatak je spoj podataka odabrane nadstavke i podstavki koje se referenciraju na tu nadstavku. Očito je da će za to biti potrebno imati 1-na-N vezu između tablica u bazi podataka.

Također, bitno je uočiti razliku prikaza i forme. Prikaz samo prikazuje podatke, forma omogućuje i izravno uređivanje podataka.

3.1 Master-detail forma

Master-detail (MD) forma se sastoji od dva okvira: **mastera** i **detaila**. Konkretni primjer izgleda MD forme na domenskom primjeru prikazan je Slika 3.1. *Master* omogućuje uređivanje podataka nadstavke, te kretanje po redcima tablice nadstavke. *Detail* tablično prikazuje sve podstavke i njihove relevantne atribute. Svaku stavku detaila je moguće urediti ili izbrisati, a moguće je i stvoriti novu podstavku ili stvoriti asocijaciju prema već postojećoj stavci. U detailu se omogućuje uređivanje samo onih atributa koji su relevantni za tematiku nadstavke.

Također, uzmite u obzir da i nadstavka i podstavka imaju 1-na-N vezu prema šifarničkim tablicama, čime se ostvaruju padajući izbornici (*dropdown list*, *combobox*...). Konkretnije iz domenskog primjera možemo vidjeti da automobili imaju proizvođača, a dijelovi imaju svoj status ispravnosti. Ovakvu funkcionalnost ćete trebati implementirati i u svojim formama.

MD prikaz podatkovno izgleda jednako MD formi (pa i po pitanju navigacije po stavkama), uz razliku da MD prikaz ne omogućava uređivanje prikazanih podataka. Tijekom izrade MD prikaza promislite o implementaciji koja bi kasnije omogućila što jednostavniju pretvorbu (proširenje) prikaza u formu. Drugim riječima, prikazom si jednostavno možete postaviti temelje za izradu forme.

MojaApp

Automobili Proizvođači Statusi dijelova

Prethodni Sljedeći

Registracija: ZG-12134-HN

Proizvođač: Opel

Broj šasije: 22233344556234

Godina proizvodnje: 2015

Spremi Obriši

Dijelovi

#	Naziv dijela	Datum ugradnje	Status		
1	Kotac LP	5.5.2020	Ispravan	Uredi	Obriši
2	Volan	27.6.2015	Servisiran	Uredi	Obriši
3	Filtar čestica	31.1.2018	Neispravan	Uredi	Obriši

Stvori novi dio

MASTER

DETAIL

Slika 3.1 Master-detail forma⁸

3.2 Složeni tablični prikaz

Implementacija složenog tabličnog prikaza se očekuje nad tablicama koje ste odabrali za izradu MD forme. Složeni tablični prikaz tablično prikazuje podatke mastera, ali i sažete podatke o detaljima u jednom stupcu (Slika 3.2). U slučaju automobilskih dijelova, moguće je pobrojati samo nazive dijelova koje neki automobil ima. U nekom kompleksnijem slučaju moguće je stvoriti i tekstualnu reprezentaciju podstavke koja se može sastojati od vrijednosti više njenih atributa. U sažetom popisu je bitno prikazati razumljive podatke.

⁸ **NAPOMENA:** Ovaj zaslon je samo primjer izgleda MD forme i ne sadrži sve elemente koji se ocjenjuju na kontrolnim točkama projekta. Za detaljnije informacije proučite dokument *RPPP-vrednovanje.pdf*.

Konkretna način prikaza je prepušten slobodnoj volji; moguće je implementirati gumb koji za svaki redak otkriva i sakriva sažete podatke.

U svakom retku složenog tabličnog prikaza trebali bi postojati gumbi za uređivanje i brisanje. Gumb za uređivanje može služiti kao navigacija na MD formu za uređivanje te stavke. Gumb za detalje može služiti kao navigacija na MD prikaz.

#	Registracija	Broj šasije	Godina proizv.	Proizvođač	Dijelovi	
1	ZG 134-HH	11122233344	2019	Opel	LP kotač, Volan, DP Kotač	Detalji Uredi Obriši
2	KC 777-PR	55556666777	2004	VW	Retrovizor, Vjetrobran	Detalji Uredi Obriši
3	RI 442-NJ	89876435724	2015	Renault	Filtar čestica, Volan, Filtar ulja, Auspuh	Detalji Uredi Obriši

<< 1 2 3 4 5 6 7 8 9 >>

Slika 3.2 Složeni tablični prikaz⁹

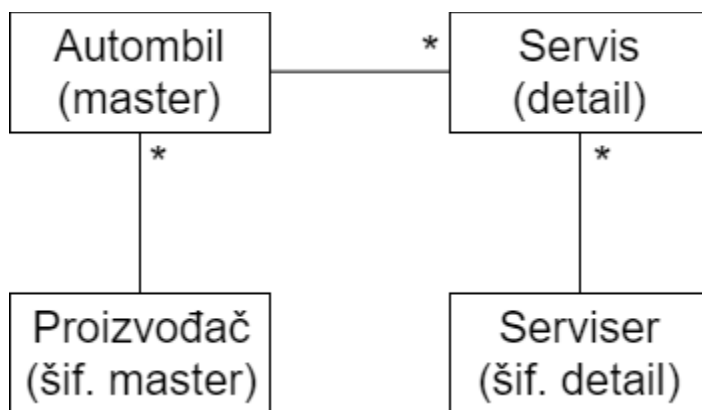
⁹ **NAPOMENA:** Ovaj zaslon je samo primjer izgleda složenog tabličnog prikaza i ne sadrži sve elemente koji se ocjenjuju na kontrolnim točkama projekta. Za detaljnije informacije proučite dokument *RPPP-vrednovanje.pdf*.

4. Raspodjela domene unutar tima

Prolaskom kroz teme o modeliranju i oblicima složenih prikaza sada možemo diskutirati o raspodjeli domene unutar tima. Svaki član tima treba dobiti svoju poddomenu nad kojom će graditi vlastiti dio aplikacije (forme i prikaze) i sastavljati dokumentaciju. Pri tome se treba obratiti pozornost da svi u timu imaju podjednak broj tablica, te da imaju sve tablice koje su potrebne za izradu MD forme, MD prikaza i složenog tabličnog prikaza.

Intuitivno je prvo razmotriti tematsku raspodjelu domene. To je dobar početak, ali to nekada nije sasvim izvedivo zbog nedostatka adekvatnih „tabličnih formacija“. Stoga je potrebno ili preurediti poddomenu ili pomaknuti poddomenu kako bi obuhvaćala druge tablice.

U konačnici, svaki član tima trebao bi **barem** posjedovati tablice za: master stavke, detail stavke, šifarnik mastera i šifarnik detalja¹⁰. Primjer jednog dobrog odabira je prikazan na Slika 4.1. Tablice mastera i detalja, uz attribute stranih ključeva, bi trebale imati i nekoliko podatkovnih atributa. Za svaki slučaj, svatko može imati još i rezervne tablice u slučaju da se prvobitni odabir pokaže nedostatnim.

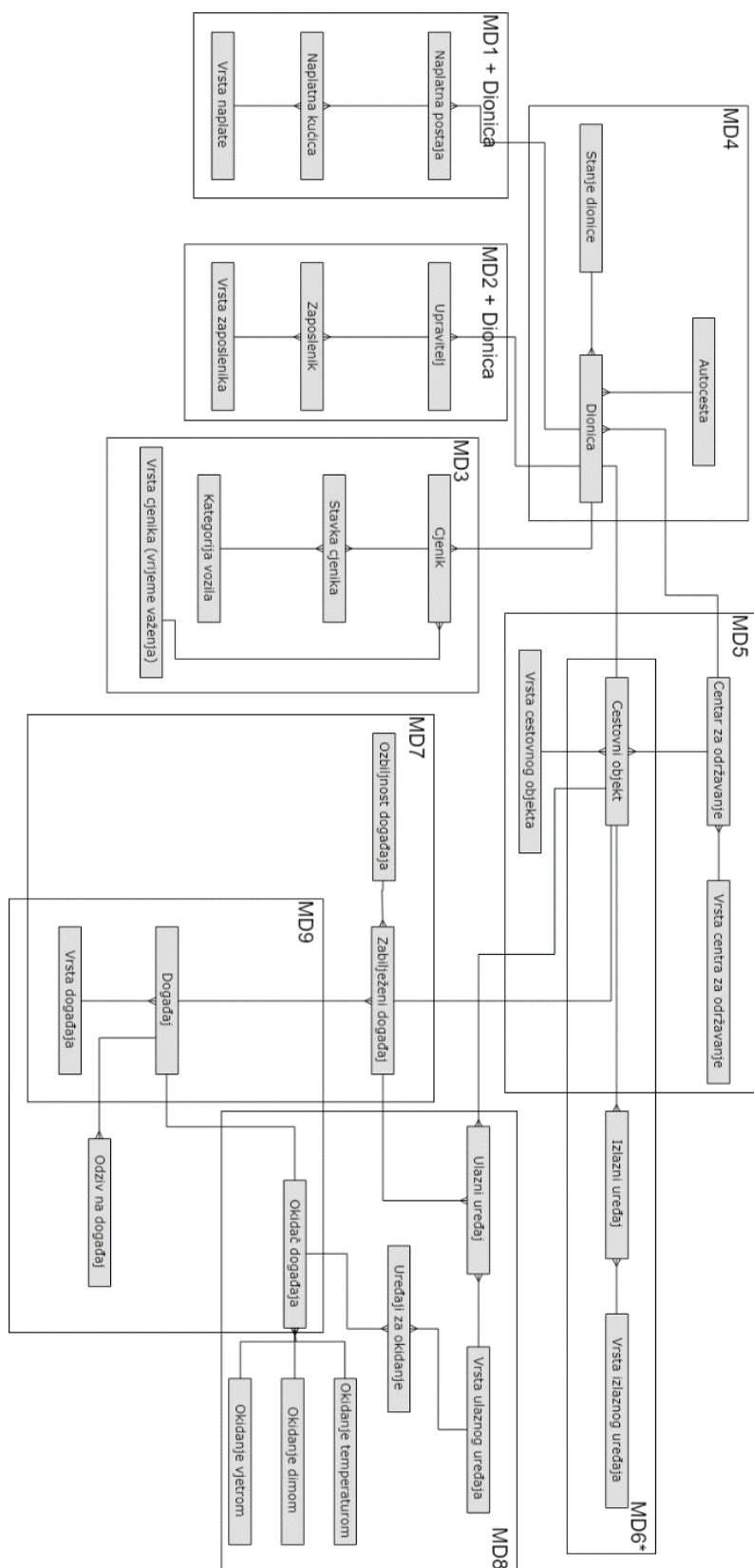


Slika 4.1 Primjer jedne poddomene iz domenskog primjera koja podržava izradu MD forme

Ovaj „uzorak“ tablica (Slika 4.1) može se improvizirati i na mjestima gdje postoji N-na-N veza tablica, gdje se 1-na-N veza može razmatrati samo u jednom smjeru. Također, nečija master tablica, može biti nečija šifrarska tablica. U lošijem slučaju (koji bi se trebao izbjegavati) nečija master tablica može biti tuđa details tablica. U tim slučajevima svaki član ekipe mora imati implementirane vlastite prikaze i forme za te tablice – **forme i prikazi ne smiju se dijeliti**.

Intervju uvijek ima više nego dovoljno sadržaja za modeliranje domene dostatne veličine za sve članove tima. Za temu Autocesta iz akademske godine 2018./2019., uz dobru razradu bilješki s intervjuja, mogla se razraditi domena prikazana **Error! Reference source not found.** Iz te domene moglo se odrediti čak 9 različitih poddomena. Kvaliteta domene i raspodjele ovisi o uloženom trudu za izradu. Uzimajući u obzir da je razrada domene ključna točka razvoja, u nju se svakako isplati uložiti maksimalan trud.

¹⁰ Ovdje se jasno može vidjeti zašto je predloženo barem 20 domenskih tablica na peteročlani tim (4 tablice × 5 članova).



Slika 4.2 Razrada domene za temu autocesta

5. Ostale važne napomene

5.1 Unos i stvaranje vrijednosti primarnog ključa

Česta pogreška studenata na RPPP-u tijekom implementiranja formi je odabir načina na koji se stvaraju primarni ključevi (PK) u bazi podataka. Je li stvaranje i unos vrijednosti PK-a odgovornost korisnika ili sustava?

Ako PK nema konkretne veze s podacima koji se pohranjuju u tablicu, izuzev kao unikatni identifikator n-torke, onda taj PK mora biti nekako generiran. Najjednostavniji oblici automatskog generiranja PK-a su stvaranje *globally unique identifier* (GUID) ili postavljanje PK-a na atribut cjelobrojne vrijednosti s auto-inkrementom (AI). GUID je unikatna (u slučaju brisanja n-torke neće se više pojaviti n-torka s tim istim ID-em), ali memorijski je velik i nepregledan tijekom debugiranja. AI je pregledniji tijekom debugiranja i zauzima manje memorijskog prostora, ali je nesiguran i nije unikatna (što može biti problem kod raspodijeljenih baza podataka)¹¹. Za potrebe RPPP-a u većini slučajeva je dovoljan AI.

Također, vrijednosti PK-a moguće je stvarati programski korištenjem vrijednosti ostalih atributa n-torke. Primjer programskog generiranja PK-a može se vidjeti u primjeru JMBG-a, koji je sadržavao vidljive segmente o datumu, regiji i rednom broju rođenja osobe.

Postoji i mogućnost ručnog upisivanja vrijednosti PK-a u nekim specifičnim tablicama. Često je tada riječ o kataloškim vrijednostima koje nemaju neku programski ostvarivu logiku PK-a ili o entitetima koji već posjeduju neku unikatnu značajku. Primjeri toga su CPV u javnoj nabavi (<https://narodne-novine.nn.hr/clanci/sluzbeni/dodatni/419684.pdf>) i OIB kod pravnih i fizičkih osoba u Republici Hrvatskoj.

¹¹ Više o tome može se pronaći na sljedećim mjestima:

<https://blog.codinghorror.com/primary-keys-ids-versus-guids/>

<https://www.clever-cloud.com/blog/engineering/2015/05/20/why-auto-increment-is-a-terrible-idea/>

<https://blog.pythian.com/case-auto-increment-mysql/>