Univerza v Ljubljani
Fakulteta za matematiko in fiziko

Matematika – uporabna smer (UNI)

Urša Remžgar

Načrtovanje relacijskih podatkovnih baz z entitetno relacijskimi diagrami

Diplomsko delo

Ljubljana, 2012

Kazalo

Slike

Program diplomskega dela

V diplomskem delu obravnavajte metode za načrtovanje relacijskih podatkovnih zbirk. Predstavite ER diagrame, algoritem za preslikavo ER diagrama v relacijsko shemo ter normalne oblike shem in normalizacijo.

Osnovna literatura:

- R. Ramakrishnan, J. Gehrke, Database management systems-third edition. New York, McGraw-Hill 2003
- T. Mohorič, Podatkovne baze. Ljubljana, BI-TIM 2002

Ljubljana, december 2011

izred. prof. dr. Andrej Bauer

Povzetek

V nalogi sem predstavila postopek načrtovanja relacijske podatkovne baze z entitetno relacijskimi diagrami. Najprej sem opredelila podatkovne baze in pojasnila pomen dobrega načrtovanja. Predstavila sem faze izdelave podatkovne baze in v nalogi podrobneje pregledala fazi konceptualnega in logičnega načrtovanja ter fazo pregleda sheme. Ponazorila sem jih na primeru izdelave relacijske sheme fakultete. V fazi konceptualnega načrtovanja sem opredelila entitetno relacijski model in njegov pomen. Predstavila sem osnovne konstrukte entitetno relacijskega diagrama in jih ponazorila na primerih iz sheme "Fakulteta". Navedla sem postopek izdelave entitetno relacijske sheme in predstavila možne dileme pri izdelavi. V fazi logičnega načrtovanja sem opredelila relacijski podatkovni model in predstavila algoritem za preslikavo entitetno relacijske sheme v relacijsko shemo podatkovne baze. V fazi pregleda sheme sem predstavila metode za preverjanje ustreznosti relacijske sheme in težave, ki nastanejo zaradi neustrezne izdelave sheme. Opredelila sem funkcionalne odvisnosti in predstavila normalne oblike.

Math. Subj. Class. (MSC 2010): 68P15

Ključne besede:
relacijske podatkovne baze, entitetno relacijski diagram, načrtovanje podatkovne baze, normalizacija

Keywords:
relational database, entity relational diagram, database design, normalization

Načrtovanje podatkovne baze

Podatkovne baze

Podatkovne baze so postale nepogrešljiv del informacijskih sistemov v mnogih organizacijah. Skoraj vsak od nas vsakodnevno pride v stik z aplikacijami, ki temeljijo na podatkovnih bazah, na primer, ko v banki položi depozit, rezervira letalsko vozovnico, preveri dano knjigo v knjižničnem kataloškem sistemu in tako dalje.

Podatkovna baza je definirana kot urejena zbirka medsebojno povezanih podatkov. Podatki predstavljajo znana dejstva, ki jih je moč zabeležiti in imajo neposreden pomen. Takšna definicija pa je precej splošna in navadno pojem podatkovne baze razumemo ožje. Običajno mora podatkovna baza zadoščati dodatnim lastnostim:

- Podatkovna baza predstavlja nek vidik realnega sveta, ki ga imenujemo tudi minisvet, oziroma problemska domena. Spremembe tega minisveta se nato odražajo v podatkovni bazi.
- Podatkovna baza je logično povezana zbirka podatkov z nekim naravnim pomenom. Zgolj naključno izbran nabor podatkov še ne predstavlja podatkovne baze.
- Podatkovna baza je načrtovana, zgrajena in vsebuje podatke z določenim namenom. Prav tako obstaja skupina ljudi, imenovana uporabniki, ki jo uporabljajo.

Podatkovno bazo lahko zgradimo in vzdržujemo ročno, ali pa s pomočjo sistemov za upravljanje s podatkovnimi bazami, oziroma okrajšano SUPB. **SUPB** je zbirka programov, ki omogočajo uporabnikom izgradnjo in vzdrževanje podatkovne baze. Obstaja več različnih vrst SUPB, ki se ločijo glede na podatkovni model, na katerem temeljijo. Najpogostejše uporabljena podatkovna modela sta relacijski podatkovni model in objektni podatkovni model. V nalogi se bomo omejili na relacijski podatkovni model, ker se najpogosteje uporablja.

Relacijske podatkovne baze

Relacijske podatkovne baze temeljijo na relacijskem podatkovnem modelu, ki ga je leta 1969 iznašel E. F. Codd, raziskovalec pri IBM. Relacijski podatkovni model temelji na matematični teoriji množic in predikatni logiki. Osnovni gradnik relacijskega podatkovnega modela je matematična relacija, predstavljena s tabelo, ki je človeku dobro razumljiva. Glavna ideja pri iznajdbi relacijskega podatkovnega modela je bila, da naj bo podatkovna baza sestavljena iz zaporedja neurejenih relacij, s katerimi lahko manipuliramo z različnimi neproceduralnimi operacijami, ki vračajo relacije. Relacijski podatkovni model

je imel velike prednosti v primerjavi s tedanjimi tradicionalnimi podatkovnimi modeli, ker je bil enostavnejši, fleksibilnejši in neodvisen od metod za fizično shranjevanje podatkov.

Načrtovanje podatkovne baze

Načrtovanje podatkovne baze je postopek opredelitve in razvoja strukture podatkovne baze. Cilj tega procesa je izdelava logične in fizične strukture podatkovne baze, ki bo zadoščala danim potrebam uporabnikov v organizaciji za vnaprej določeno množico uporabnikov. V času načrtovanja podatkovne baze naredimo formalni model nekaterih vidikov realnega sveta. Pri tem je merilo za pravilnost načrtovane sheme podatkovne baze realni svet. To pomeni, da mora vsebina podatkovne baze pravilno in natančno odražati podatke, pravila in izjeme iz realnega sveta (povzeto po poglavju Načrtovanje podatkovne baze v [8]). Poleg tega pa mora podatkovna baza zadoščati tudi nekaterim zmogljivostnim kriterijem, kot so: odzivni čas, čas potreben za izvajanje operacij in količina prostora, ki ga potrebuje za shranjevanje. Ti kriteriji pa so včasih tudi izključujoči, zato je potrebno premišljeno in natančno načrtovanje podatkovne baze. Prav tako velja, da je spreminjanje že izdelane podatkovne baze, ki je bila slabo ali površno načrtovana, vse prej kot enostavno. Zato je sam proces načrtovanja ključnega pomena za učinkovitost podatkovne baze.

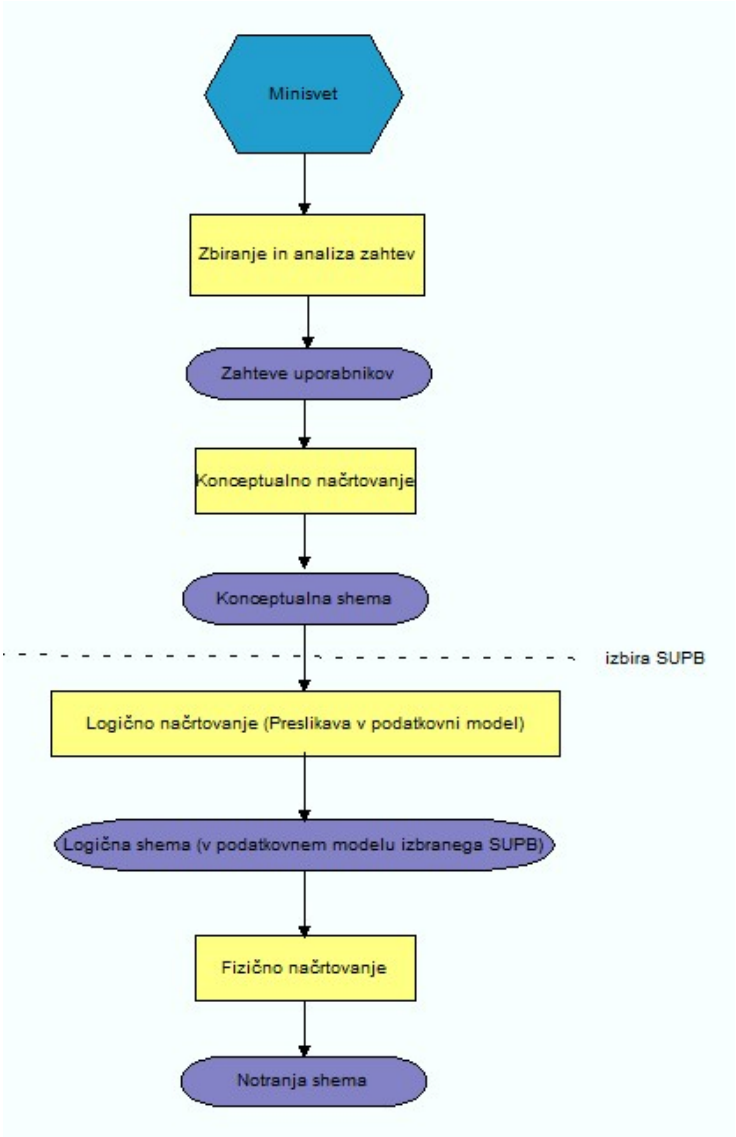
Pri načrtovanju podatkovne baze se razvijalci soočajo z vrsto izzivov, kot so: nepoznavanje področja, upoštevanje vseh pravil in izjem ter kompleksnost načrta podatkovne baze. Izbira primerne metode za načrtovanje podatkovne baze razvijalcu prinese veliko prednosti, kot so:

- Zmanjšanje možnosti za pojavitev napak pri načrtovanju (minimiziranje števila popravkov).
- Lažja izgradnja aplikacije, ki temelji na dani podatkovni bazi.
- Podatkovna baza se obnaša predvidljivo, ker je izdelana na osnovi dobro formuliranega modela.
- Dopolnitve podatkovne baze so enostavno izvedljive.
- Pridobivanje podatkov je učinkovito.
- Operacije vstavljanja, posodabljanja in brisanja podatkov so učinkovite.

Pri majhnih podatkovnih bazah, ki imajo nekje do dvajset uporabnikov, načrtovanje ni nujno posebno zapleteno. Toda pri srednje velikih in velikih podatkovnih bazah, ki jih uporablja več skupin z desetinami ali celo stotinami uporabnikov, pa je sistematičen pristop k načrtovanju nujno potreben. Obstaja več različnih metod za načrtovanje podatkovne baze. Ker smo se omejili na relacijske podatkovne baze, bomo predstavili eno od metod za načrtovanje teh baz, in sicer metodo za načrtovanje z entitetno relacijskim diagramom.

Faze izdelave podatkovne baze

Postopek načrtovanja podatkovne baze poteka po fazah, kot je prikazano na sliki 1. Prvi dve fazi sta neodvisni od izbire danega SUPB, medtem ko so preostale specifične za izbrani SUPB. Faze načrtovanja podatkovne baze so:



Slika 1: Faze načrtovanja

1. **Zbiranje in analiza zahtev uporabnikov**
   V prvi fazi zberemo informacije o podatkih, ki jih želimo shraniti v podatkovni bazi. Razumeti moramo, katere podatke je potrebno hraniti v podatkovni bazi in kako so ti podatki med seboj povezani. Premisliti moramo, katere operacije se bodo najpogosteje izvajale nad podatki in kako bodo uporabniki uporabljali podatkovno bazo. Ugotoviti moramo torej, kaj uporabniki želijo od podatkovne baze. V tej začetni fazi lahko pride do razlik med željami uporabnikov in razumevanjem načrtovalcev, zato je potrebno njihovo tesno sodelovanje. Aktivno sodelovanje uporabnikov v fazi zbiranja in analiziranja zahtev navadno zagotavlja večje zadovoljstvo uporabnikov s končnim izdelkom. Običajno je potrebno več delavnic ali sestankov, da se pričakovanja uporabnikov uskladijo z razumevanjem načrtovalcev. Načrtovalci nato po uskladitvi na podlagi zahtev uporabnikov pripravijo dokument, imenovan analiza zahtev, v katerem čim bolj natančno in celovito opišejo in razdelajo zahteve uporabnikov. Dokument tako predstavlja načrtovalcem temelj za nadaljnji postopek načrtovanja podatkovne baze. Po uskladitvi in potrditvi ustreznosti dokumenta s strani uporabnikov se začne faza konceptualnega načrtovanja.

2. **Konceptualno načrtovanje**
   Informacije, zbrane v prejšnji fazi, uporabimo za razvoj konceptualne sheme. Konceptualna shema je strnjena predstavitev zahtev uporabnikov, ki vključuje podroben opis entitetnih tipov, relacij in omejitev ter temelji na konceptualnem modelu. Konceptualni model je pregleden opis podatkov in njihovih omejitev, ki predstavlja izhodišče za razvoj podatkovnega sistema. Obstaja več vrst konceptualnih modelov. Pomembno je, da izberemo konceptualni model, ki je enostaven, vendar natančen, razumljiv in ima pregledno grafično predstavitev. V nalogi se bomo omejili na entitetno relacijski model, oziroma okrajšano ER model, ker je enostaven, pregleden in pogosto uporabljen. Cilj izgradnje konceptualnega modela je zagotoviti enostavno in strnjeno predstavitev podatkov, ki bo razumljiva uporabnikom in bo hkrati primerna za nadaljnjo pretvorbo v podatkovni model. Izgradnja konceptualne sheme poteka skladno z določenimi pravili in značilnostmi, ki jih določa dani konceptualni model. Ti koncepti pa ne vključujejo implementacijskih podrobnosti, zato so primerni tudi za komunikacijo z uporabniki, ki nimajo tehničnega znanja o podatkovnih bazah. Konceptualna shena je uporabnikom pregledna in razumljiva, zato omogoča in olajša komunikacijo načrtovalcev z uporabniki. Služi za usklajevanje in preveritev, da so upoštevane vse zahteve uporabnikov in da si te zahteve med seboj ne nasprotujejo. Konceptualna shema predstavlja tudi dokumentacijo projekta, saj natančno in razumljivo opisuje vsebino podatkovne baze.

3. **Logično načrtovanje**
   Naslednji korak v načrtovanju podatkovne baze je izbira ustreznega sistema za upravljanje s podatkovnimi bazami (SUPB) in pretvorba dobljene konceptualne sheme v logično shemo. Ta proces se imenuje logično načrtovanje oziroma preslikava v podatkovni model. Rezultat procesa je shema podatkovne baze (logična shema) zgrajena na podlagi podatkovnega modela izbranega SUPB. Ker smo se v nalogi omejili zgolj na relacijske podatkovne modele in pri konceptualnem načrtovanju predstavili zgolj entitetno relacijski model, smo v fazi logičnega načrtovanja specifično obravnavali pretvorbo entitetno relacijske sheme v relacijsko

podatkovno shemo. Pri tem smo si pomagali z orodjem za načrtovanje podatkovne baze PowerDesigner, ki smo ga podrobneje predstavili v poglavju Orodja za izdelavo ER modelov.

4. **Pregled sheme**
   Entitetno relacijska shema predstavlja približen opis podatkov, saj je konstruirana subjektivno iz informacij pridobljenih v fazi zbiranja zahtev uporabnikov. Običajno moramo v postopku načrtovanja podatkovne baze dobljeno logično shemo še dopolniti. Preučiti moramo relacije dane relacijske podatkovne sheme, da odkrijemo morebitne težave in jih popravimo. V nasprotju s fazo zbiranja zahtev in fazo konceptualnega načrtovanja, ki sta izrazito subjektivni, si lahko v fazi pregleda sheme pomagamo s teorijo. V nalogi bomo podrobneje predstavili teorijo funkcionalnih odvisnosti in normalnih oblik.

5. **Fizično načrtovanje**
   V fazi fizičnega načrtovanja je potrebno pregledati zmogljivostne cilje podatkovne baze. Poznati moramo delovno obremenitev, ki jo mora podatkovna baza podpirati. Delovno obremenitev sestavljajo večinoma poizvedbe in posodobitve tabel. Prav tako imajo navadno uporabniki zahteve, kako hitro se morajo izvajati določene poizvedbe oziroma posodobitve tabel, ali pa zahtevajo, koliko transakcij naj se izvede v sekundi. Pri fizičnem načrtovanju moramo torej upoštevati oceno delovne obremenitve in zmogljivostne zahteve uporabnikov. Za uspešno izvedbo moramo dobro poznati delovanje uporabljenega SUPB, še posebej indeksiranje in tehnike izvajanja poizvedb. Odločiti se moramo, katere in koliko indeksov bomo uporabili, katere poizvedbe bo morda potrebno optimizirati in ali bomo morda za izboljšanje učinkovitosti uporabili denormalizacijo (obraten postopek kot normalizacija v prejšnji fazi).

6. **Implementacija**
   V tej fazi ustvarimo shemo podatkovne baze in naložimo vanjo prazne objekte, ki jih nato lahko napolnimo s podatki. Prav tako definiramo uporabniške role in za vsako uporabniško rolo določimo, do katerih delov podatkovne baze lahko dostopa in katere operacije nad posamezno tabelo lahko izvaja, katerih pa ne.

Entitetno relacijski model

Pomen ER modela v konceptualnem načrtovanju

Entitetno relacijski model, oziroma okrajšano ER model, je iznašel dr. Peter P. Chen leta 1976. ER model je predstavljen v obliki diagrama, sestavljenega iz entitet (objektov), relacij med njimi in atributov (lastnosti) entitet in relacij. Prednosti ER modela so:

- je enostaven,
- možno ga je pretvoriti v različne podatkovne modele,
- je neodvisen od konkretnih komercialnih izvedb baz podatkov in njihovih SUPB.

Osnovni konstrukti ER modela so: entiteta in entitetni tip, atribut, entitetni identifikator (ključ entitetnega tipa), razmerje med entitetnimi tipi in tip (številnost) razmerja ter šibki entitetni tip.

ER diagram

Entitetno relacijski diagram je grafična predstavitev entitetno relacijskega modela. Grafična predstavitev omogoča boljši pregled nad strukturo podatkov in dobro komunikacijo z uporabniki. Uporabnike lahko zato s pomočjo ER diagrama lažje vključimo v proces izdelave in pregled ustreznosti načrta podatkovne baze.

V nalogi smo načrtovanje podatkovne baze predstavili na primeru načrtovanja podatkovne baze fakultete. Upoštevali smo zahteve, ki jim mora zadoščati podatkovna baza fakultete: izbrana fakulteta ima več oddelkov, vsak oddelek zaposluje nekaj profesorjev. Vsak oddelek je nato razdeljen na več različnih smeri študija. Vsaka smer študija ima več različnih letnikov: 1. letnik, 2. letnik in 3. letnik. V vsakega od letnikov posamezne smeri so vpisani študenti. Vsakemu študentu je ob prvem vpisu dodeljena vpisna številka, ki ga enolično določa. Prav tako ima vsak profesor določeno oznako, ki ga enolično določa. Profesor poučuje enega ali več predmetov, ki ga obiskujejo študenti. Vsak profesor ima svoj kabinet. Nekateri profesorji so tudi mentorji mladim raziskovalcem. Na podlagi navedenih zahtev smo z entitetno relacijskim diagramom izdelali konceptualno shemo fakultete, ki je predstavljena na sliki 20.

Orodja za izdelavo ER modelov

V začetku iznajdbe tehnologije podatkovnih baz so proces načrtovanja in izdelave ročno izvajali izkušeni načrtovalci, ki so se zanašali predvsem na svoje izkušnje in znanje. Vendar

pa obstajata vsaj dva razloga za uvedbo avtomatizacije v proces načrtovanja podatkovnih baz. Prvi razlog je zapletenost modela, drugi pa velikost modela. Bolj kot je model zapleten (zapletenejša razmerja in omejitve), večje je število možnosti, kako model izdelati. Težko je namreč ročno upoštevati vse različne možnosti pri njegovi izdelavi. Prav tako je ročno težko obvladovati modele, ki zajemajo stotine entitetnih tipov in razmerij. Vsi ti razlogi so spodbudili nastanek orodij za načrtovnje podatkovnih baz, imenovanih CASE (Computer-Aided Software Engineering). Orodja CASE običajno združujejo izdelavo diagramov, preslikavo v podatkovni model in normalizacijo. Načrtovalcu omogočajo izris konceptualnega diagrama sheme v neki specifični notaciji. Dobljeno konceptualno shemo nato znajo z algoritmom preslikati v podatkovni model nekaterih specifičnih SUPB. Nekatera orodja CASE znajo s pomočjo določenih funkcionalnih odvisnosti izvesti dekompozicijo relacij, tako da ustrezajo določeni normalni obliki (povzeto po poglavju 16.1 v [3]).

V nalogi sem uporabljala orodje PowerDesigner, verzijo 15.1. PowerDesigner je vodilna programska oprema na področju poslovnih procesov in modeliranja podatkov. Izbrala sem ga, ker je enostaven in pregleden ter z bogatim naborom vgrajenih funkcij načrtovalcu olajša izdelavo sheme podatkovne baze. Edina pomanjkljivost programa je, da je plačljiv, vendar sem za potrebe diplomske naloge uporabljala poskusno različico.

Notacije ER diagramov

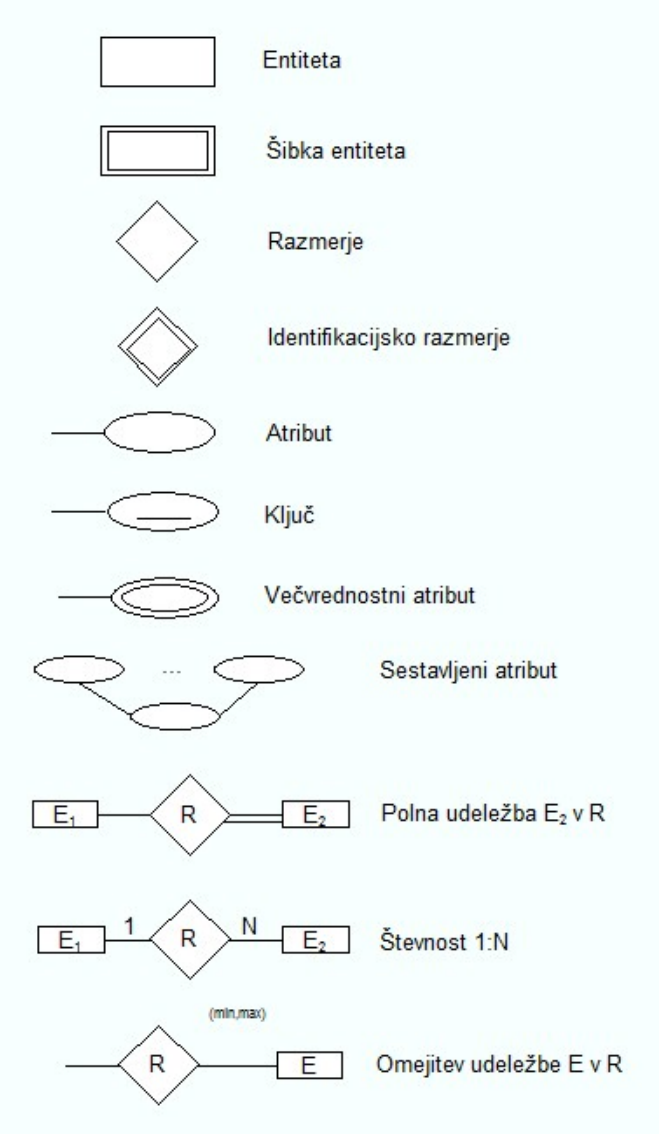
Obstaja več različnih načinov zapisovanja ER diagramov, vendar nobena notacija ni standardna. V teoriji se najpogosteje uporablja izvorna Chenova notacija, ki je predstavljena na sliki 2. Ta notacija se običajno pojavlja v knjigah, medtem ko orodja CASE uporabljajo več različnih vrst notacij. Izbrano orodje PowerDesigner uporablja tako imenovano notacijo vranjih nog, ki jo bomo podrobneje predstavili v poglavju Razmerja med entitetnimi tipi.

Entitete, entitetni tipi in atributi

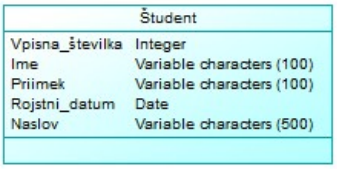
Osnovni objekt, ki ga predstavlja entitetno relacijski model je entiteta. **Entiteta** je objekt realnega sveta, ki ga je možno razlikovati od drugih objektov. Vsaka entiteta mora torej imeti neko lastnost, po kateri jo lahko enolično opredelimo. Entiteta je lahko fizičen objekt, na primer: študent, avto, hiša, zaposleni, ali pa abstrakten objekt, na primer: podjetje, delovno mesto, predmet na urniku.

Entitetni tip je množica podobnih entitet, ki imajo enake atribute (lastnosti). Izbira imen za entitetne tipe ni vedno enostavna. Izbrati moramo ime, ki kar se da možno povzema vsebino danega konstrukta. Običajno izberemo ime entitetega tipa v ednini, saj se to ime navezuje na vsako posamezno entiteto znotraj danega entitetnega tipa. Primer entitetnega tipa je denimo entitetni tip "Študent" na sliki 3, ki predstavlja vse študente na fakulteti.

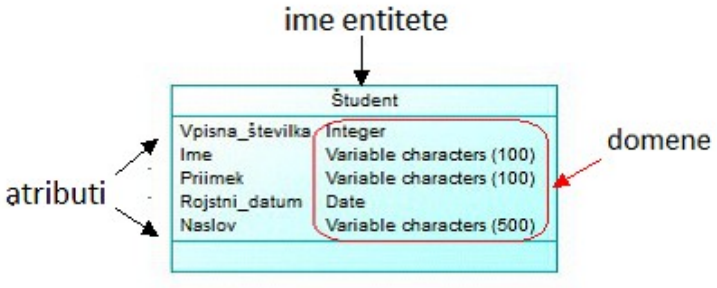
Entiteto opisujejo lastnosti, imenovane **atributi**. Na primer entiteta "Študent", predstavljena na sliki 4, ima atribute: "Ime", "Priimek", "Vpisna_številka", "Rojstni_datum" in "Naslov". Entiteta ima za vsak atribut predpisano vrednost. Vrednost atributa je podatkovni element nekega podatkovnega tipa, na primer niz (Character), število (Number), datum (Date) ipd. Vrednosti atributov, ki opisujejo vsako entiteto, so glavni del podatkov, ki jih shranjujemo v podatkovni bazi. Za vsak atribut, ki pripada



Slika 2: Chenova notacija



Slika 3: Entitetni tip "Študent"



Slika 4: Sestavni deli entitete

entiteti, moramo določiti domeno vseh možnih vrednosti. Denimo domena atributa "Vpisna_številka" je množica celih števil (Integer), domena atributa "Rojstni_datum" pa množica datumov (Date).

Poznamo več vrst atributov: enostavne in sestavljene, enovrednostne in večvrednostne ter opcijske atribute. **Sestavljeni atributi** so takšni, ki jih je mogoče razdeliti v več manjših delov, vsak od njih pa je enostaven atribut z neodvisnim pomenom. Na primer atribut "Naslov" je mogoče razdeliti v več manjših atributov: "Ime_ulice", "Hišna_številka", "Poštna_številka" in "Pošta". Atributi, ki jih ni mogoče razdeliti, se imenujejo **enostavni atributi**. Sestavljeni atributi so uporabni takrat, kadar se enkrat sklicujemo na celotno vrednost atributa, drugič pa zgolj na eno komponento atributa. V kolikor pri sklicevanju na atribut vedno uporabljamo celotno vrednost atributa, ni nikakršne potrebe, da atribut obravnavamo kot sestavljen, ampak ga obravnavamo kot enostaven atribut.

Večina atributov ima predvideno eno vrednost za posamezno entiteto. Takšne atribute imenujemo **enovrednostni**. Na primer "Starost" je tipičen primer enovrednostnega atributa. V nekaterih primerih pa ima lahko posamezen atribut več vrednosti za dano entiteto. Takšne atribute imenujemo **večvrednostni**. Primer takšnega atributa je "Diploma", saj oseba lahko ima ali nima diplome, obstajajo pa tudi ljudje, ki imajo več diplom.

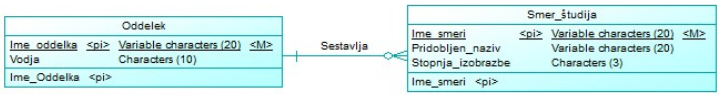
V nekaterih primerih entiteta za posamezen atribut nima zabeležene vrednosti. Takrat atributu pripišemo posebno vrednost, imenovano null, ki označuje, da vrednost atributa ni znana. Primer takšnega atributa je "Telefonska_številka", ki jo pri vseh osebah ne poznamo. Atribut, ki ima dovoljene tudi null vrednosti, imenujemo opcijski atribut.

Razmerja med entitetnimi tipi

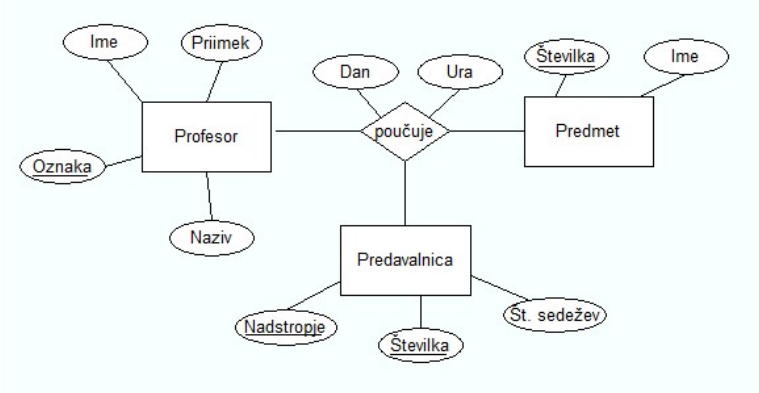
Razmerje je povezava med dvema ali več entitetnimi tipi. Množica razmerij R, ki povezuje entitetne tipe $E_1, E_2, \ldots E_n$, je množica n-teric $(e_1, e_2, \ldots e_n)$, kjer je vsaka entiteta e_i vsebovana v entitetnem tipu E_j. Vsaka n-terica predstavlja povezavo med entitetami e_1 do e_n. Veljati pa mora še, da je iz vsakega entitetnega tipa vključena natanko ena entiteta.

Stopnja razmerja je število entitetnih tipov, ki sodelujejo v razmerju. Na sliki 5 je prikazano razmerje "Sestavlja", ki je stopnje dve. Razmerja stopnje dve imenujemo tudi binarna razmerja.

Razmerja višjih stopenj so zapletenejša kot binarna in se redkeje uporabljajo. Izbrano



Slika 5: Razmerje "Sestavlja"

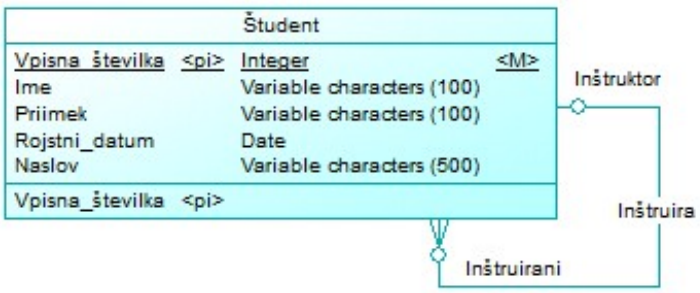


Slika 6: Razmerje "Poučuje" stopnje 3

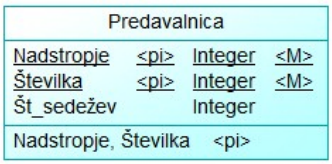
orodje Powerdesigner in večina preostalih CASE orodij ne dopušča razmerij višjih stopenj. Vendar pa, ker takšna razmerja vendarle obstajajo, smo jih izrisali v originalni Chenovi notaciji v orodju za izris splošnih diagramov, imenovanem Diagram Designer. Na sliki 6 je predstavljeno razmerje "Poučuje", ki je stopnje tri. Razmerje povezuje med seboj tri entitetne tipe: "Profesor", "Predmet" in "Predavalnica".

Vsak entitetni tip, ki nastopa v razmerju, ima v njem določeno vlogo. Ime vloge označuje, kakšno vlogo ima entiteta, ki nastopa v posameznem razmerju. Na primer v razmerju "Inštruira", predstavljenem na sliki 7, ima en študent vlogo poučevalca-inštruktorja, drugi študent pa vlogo poučevanega-inštruiranega.

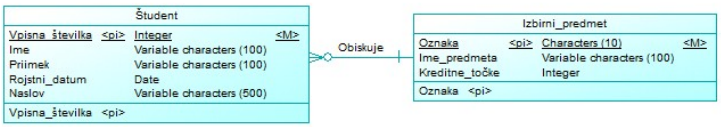
Imena vlog niso potrebna v razmerjih, kjer so vsi udeleženi entitetni tipi različni, saj lahko ime vloge določimo kar z imenom entitetnega tipa. Toda v razmerjih, kjer kakšen entitetni tip nastopa več kot enkrat, je ime vloge nujno potrebno za razlikovanje med



Slika 7: Razmerje "Inštruira"



Slika 8: Entitetni tip "Predavalnica"



Slika 9: Razmerje "Obiskuje"

posameznimi udeleženimi entitetnimi tipi. Takšna razmerja imenujemo tudi rekurzivna razmerja. Razmerje "Inštruira" na sliki 7 je primer rekurzivnega razmerja, saj povezuje entitetni tip "Študent" s samim seboj.

Razmerje ima prav tako lahko opisne atribute, ki beležijo informacije o razmerju. Denimo razmerje "Poučuje" na sliki 6 ima dva opisna atributa: "Dan" in "Ura", ki določata, na kateri dan in ob kateri uri profesor poučuje predmet v predavalnici.

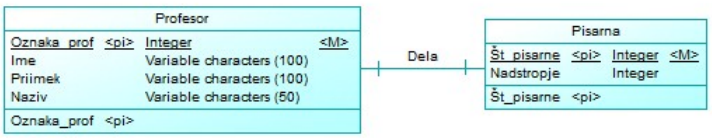
Ključi

Vsako entiteto moramo znati ločiti od preostalih entitet, zato za vsak entitetni tip izberemo ključ. **Ključ** je najmanjša možna množica atributov, katerih vrednosti lahko enolično določijo posamezno entiteto. Na primer pri entitetnem tipu "Študent" na sliki 3 za ključ izberemo atribut "Vpisna_številka", saj enolično določa posameznega študenta. Nekateri entitetni tipi imajo lahko več možnih ključev, zato vse takšne možne ključe poimenujemo kandidati za ključ. Prav tako obstajajo entitetni tipi, ki nimajo ključev. Takšni entitetni tipi se imenujejo šibki entitetni tipi. Izmed vseh kandidatov za ključ izberemo enega in ga označimo kot primarni ključ. V ER diagramu so atributi, ki sestavljajo primarni ključ, predstavljeni podčrtano. Denimo v entitetnem tipu "Predavalnica" na sliki 8 primarni ključ sestavljata atributa "Nadstropje" in "Številka", ki enolično določata predavalnico, zato sta predstavljena podčrtano.

Številnost (kardinalnost) razmerja

Razmerja imajo običajno omejitve, ki določajo možne kombinacije entitet, ki lahko nastopajo v tem razmerju. Te omejitve določimo glede na situacijo v minisvetu, ki ga predstavlja razmerje. Na primer v razmerju "Obiskuje', predstavljenem na sliki 9, imamo lahko pravilo, ki določa, da mora vsak študent obiskovati natanko en izbirni predmet.

Ločujemo med dvema skupinama omejitev: številnostjo in udeležbo. Številnost bomo definirali zgolj za binarna razmerja, to je razmerja med dvema entitetnima tipoma. **Številnost** oziroma kardinalnost razmerja označuje število razmerij, v katerih lahko



Slika 10: Razmerje "Dela"

posamezen entitetni tip sodeluje. Številost razmerja predstavimo v obliki (min, max), kjer je min najmanjše število pojavitev entitete v razmerju, max pa največje število pojavitev entitete v razmerju. Številost opredelimo za vsak entitetni tip posebej. Denimo v razmerju "Obiskuje" na sliki 9 ima entitetni tip "Izbirni_predmet" številost $(0, N)$. Minimalno število študentov, ki obiskuje dani izbirni predmet je namreč 0, maksimalno pa je N. Zaradi poenostavitve je splošna praksa, da se za številost razmerja beleži le največja številost (max) za oba entitetna tipa, ki sodelujeta v razmerju. Minimalne številosti namreč ne vplivajo na razvrstitev razmerij med osnovne zvrsti, zato jih lahko pri nekaterih notacijah izpustimo. Tako številost opredelimo zgolj z obema maksimalnima številoma pojavitve. Značilne številosti razmerij so:

- $1 : 1$ (ena proti ena) 

- $1 : N$ (ena proti mnogo) 

- $N : 1$ (mnogo proti ena) 

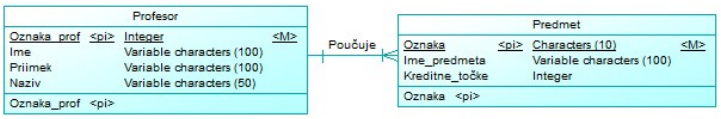
- $M : N$ (mnogo proti mnogo) 

Številost $1 : 1$ pomeni, da en primerek entitetega tipa A sodeluje v povezavi z enim primerkom entitetnega tipa B. Primer številosti $1 : 1$ je razmerje "Dela" na sliki 10. Vsak profesor namreč lahko dela samo v eni pisarni, prav tako lahko v vsaki pisarni dela samo en profesor.

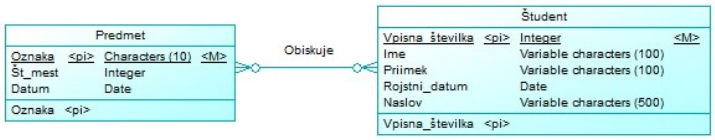
Številost $1 : N$ pomeni, da en primerek entitetega tipa A sodeluje v povezavi z enim ali več primerov entitetnega tipa B. Primer številosti $1 : N$ je razmerje "Poučuje", predstavljeno na sliki 11. Vsak profesor lahko poučuje več različnih predmetov, vendar pa vsak predmet poučuje samo en profesor.

Številost $M : N$ pomeni, da več primerkov entitetega tipa A sodeluje v povezavi z več primerki entitetnega tipa B. Primer številosti $M : N$ je razmerje "Obiskuje" na sliki 12. Vsak študent lahko namreč obiskuje več različnih predmetov, prav tako pa lahko vsak predmet obiskuje več različnih študentov.

Obstaja več različnih oblik predstavitve številosti v ER diagramu. Nekaj primerov oblik predstavitve številosti je predstavljeno na sliki 13. Najpogosteje uporabljeni obliki

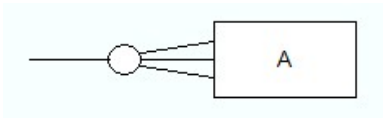


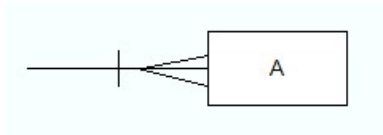
Slika 11: Razmerje “Poučuje”

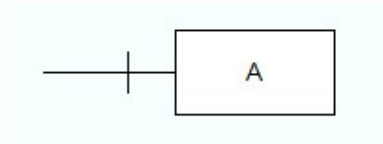


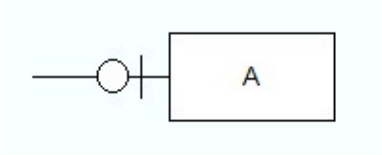
Slika 12: Razmerje “Obiskuje”

sta: označevanje z oznakami 1, M in N poleg razmerja (kot v originalni Chenovi notaciji) in notacija vranjih nog (Craw-Feet Notation). Številnost je pri notaciji vranjih nog predstavljena na naslednji način:

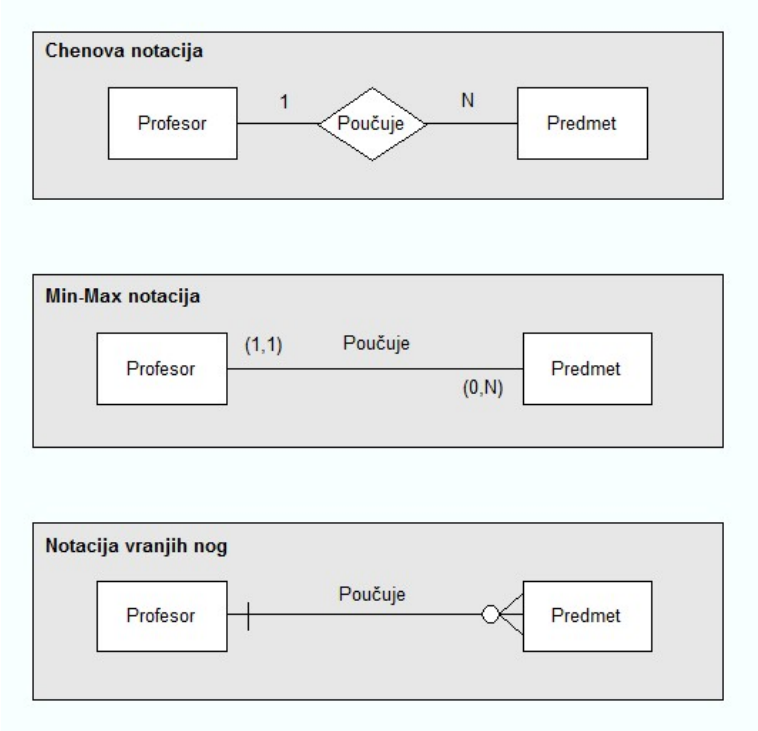
- 0 ali več: 

- 1 ali več: 

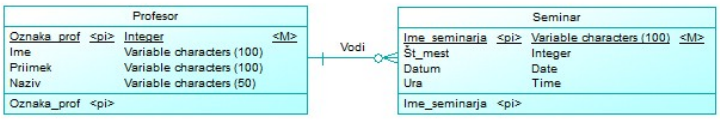
- natanko 1: 

- 0 ali 1: 

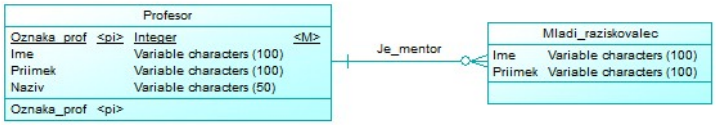
Udeležba (participacija) razmerja določa, ali je obstoj posamezne entitete odvisen od njenega razmerja z drugo entiteto. Poznamo dve vrsti udeleženosti razmerja: polno (totalno) udeležbo in delno (parcialno) udeležbo. Pri polni udeležbi velja, da se vsaka entiteta mora udeležiti v razmerju. Za parcialno udeležbo pa velja, da se entitete lahko udeležijo, ali pa ne udeležijo v razmerju. Polno udeležbo bomo ponazorili na primeru razmerja “Poučuje”, predstavljenem na sliki 11. Če na primer določimo pravilo, da mora vsak profesor poučevati vsaj en predmet, potem entiteta “Profesor” obstaja zgolj, če je udeležena v relaciji “Poučuje”. Entiteta “Profesor” ima tedaj polno udeležbo. V primeru razmerja “Vodi” na sliki 14 pa velja, da ne vodijo seminarja nujno vsi profesorji, zato ima entiteta “Profesor” delno udeležbo v relaciji.



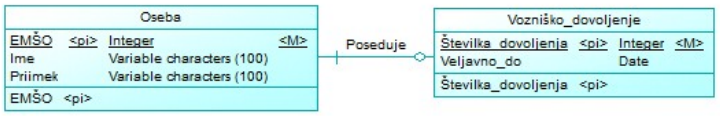
Slika 13: Različne notacije razmerij



Slika 14: Razmerje “Vodi”



Slika 15: Razmerje "Je_mentor"



Slika 16: Entitetni tip "Vozniško_dovoljenje"

Močni, šibki entitetni tipi

Entitetni tipi, ki sami nimajo atributov, ki bi tvorili ključ, se imenujejo **šibki entitetni tipi**. Entitete šibkega entitetnega tipa lahko enolično določimo zgolj v povezavi z določenimi entitetami drugega entitetnega tipa in z vrednostmi svojih atributov. Ta drug entitetni tip imenujemo starševski entitetni tip. Veljati mora, da sta starševski entitetni tip in šibki entitetni tip v razmerju številnosti $1 : N$ (ena proti mnogo). Na sliki 15 je predstavljen primer šibkega entitnega tipa, to je entitetni tip "Mladi_raziskovalec", ki je z razmerjem "Je_mentor" povezan s starševskim entitetnim tipom "Profesor". Velja, da je profesor lahko mentor enemu ali več mladim raziskovalcem, vsak mladi raziskovalec pa ima natanko enega mentorja. Mladi raziskovalec ni nujno študent te fakultete, zato nima vsak določene vpisne številke. Tako torej entitetni tip "Mladi_raziskovalec" nima določenega primarnega ključa. Posameznega mladega raziskovalca lahko enolično določimo zgolj v povezavi z njegovim mentorjem. Zgodi se namreč lahko, da sta na fakulteti dva mlada raziskovalca z istim imenom in priimkom. Tedaj ju lahko ločimo le v povezavi z mentorjem-vsak od njiju ima namreč drugega mentorja.

Šibki entitetni tip ima vedno polno udeležbo, torej obstaja zgolj v povezavi s starševskim entitetnim tipom. V našem primeru entitetnega tipa "Mladi_raziskovalec" velja, da mora vsak mladi raziskovalec na fakulteti obvezno imeti mentorja-profesorja. To pomeni, da posamezen mladi raziskovalec obstaja zgolj, če ima določenega mentorja. Vendar pa zgolj polna udeležba še ni pogoj, da je nek entitetni tip šibek. Protiprimer je denimo entitetni tip "Vozniško_dovoljenje" na sliki 16, ki ne more obstajati, če ni povezan z entitetnim tipom "Oseba". Vendar pa ima vsako vozniško dovoljenje svoj ključ, to je "Številka_dovoljenja", zato entitetni tip "Vozniško_dovoljenje" ni šibki entitetni tip.

Vsak šibki entitetni tip ima običajno **delni ključ**, to je množica atributov, ki enolično določijo šibko entiteto, ki pripada izbranemu lastniku. V primeru entitetnega tipa "Mladi_raziskovalec" na sliki 15 delni ključ sestavljata atributa "Ime" in "Priimek", ki enolično določata mladega raziskovalca za izbranega profesorja.

Postopek izdelave ER sheme

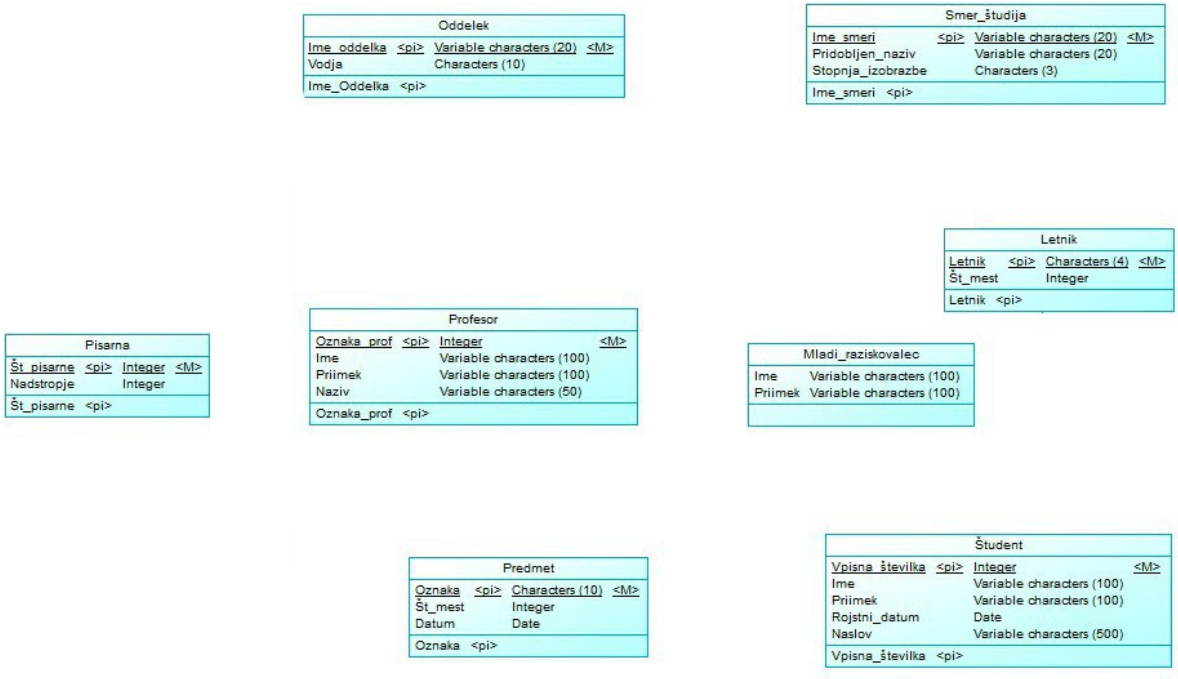
Pri izdelavi ER sheme v orodjih CASE običajno sledimo naslednjemu zaporedju korakov.

Slika 17: Dodani entitetni tipi

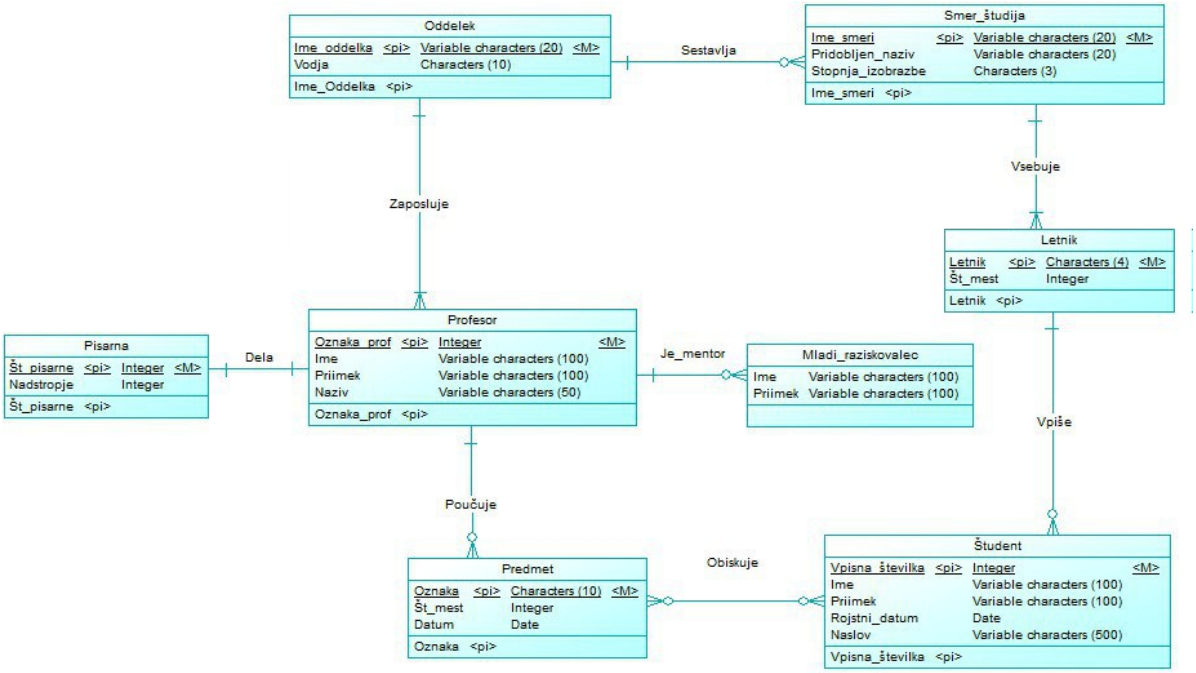
- Dodamo entitetne tipe.
- Dodamo atribute entitetnim tipom.
- Določimo entitetni identifikator (ključ).
- Določimo razmerja (povežemo entitetne tipe).
- Dodamo atribute razmerjim.
- Preverimo ustreznost modela.

Izdelavo ER sheme ponazorimo na primeru sheme “Fakulteta”. Najprej dodamo entitetne tipe: “Oddelek”, “Smer_študija”, “Letnik”, “Študent”, “Profesor”, “Pisarna”, “Predmet” in “Mladi_raziskovalec”, kar je prikazano na sliki 17. Entitetnim tipom nato dodamo atribute. Shema z dodanimi atributi je prikazana na sliki 18. Entitetnim tipom dodamo primarne ključe, kar je prikazano na sliki 19. Naslednji korak v izdelavi naše sheme “Fakulteta” je določitev razmerij med entitetnimi tipi. Dodamo razmerja, jih poimenujemo in določimo njihovo številnost. Razmerjim sicer ne dodamo nobenih atributov, vendar pa na koncu preverimo ustreznost sheme, da pravilno odraža zgradbo fakultete, ki je podana v zahtevah. Dobljena konceptualna shema je predstavljena na sliki 20.

Slika 18: Dodani atributi



Slika 19: Dodani ključi



Slika 20: Shema “Fakulteta”

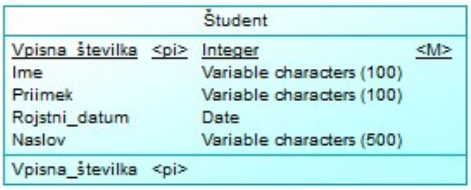
Dileme pri izdelavi ER modela

Pri izdelavi entitetno relacijskih diagramov se moramo navadno odločiti med več možnostmi:

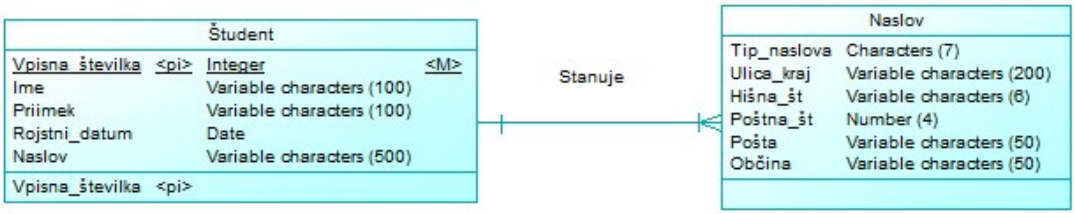
- Ali naj bo značilnost predstavljena kot entiteta ali kot atribut?
- Ali naj bo značilnost predstavljena kot entiteta ali kot razmerje?

Entiteta ali atribut

Ko določamo atribute za dan entitetni tip, včasih ne vemo zagotovo, ali naj bo neka značilnost predstavljena kot atribut tega entitetnega tipa, ali pa kot samostojni entitetni tip, ki je z razmerjem povezan s prvim entitetnim tipom. Takšen primer značilnosti je naslov za entitetni tip “Študent”. Prva možnost je, da naslov dodamo kot atribut, kar je prikazano na sliki 21. Tako ravnamo v primeru, da ima vsak študent zgolj en naslov in da zadošča, da imamo naslov predstavljen v obliki strnjenega niza. Druga možnost, predstavljena na sliki 22, pa je, da ustvarimo nov entitetni tip “Naslov” in ga z razmerjem povežemo s prvotnim entitetnim tipom “Študent”. Ta možnost je sicer zapletenejša, saj potrebujemo nov entitetni tip in razmerje, vendar pa je neizogibna v dveh primerih: kadar moramo za vsakega študenta zabeležiti več kot le en naslov (denimo stalni in začasni naslov) in kadar želimo naslov predstaviti razdeljeno na posamezne komponente: ime ulice/kraja, hišna številka, poštna številka, pošta in država.



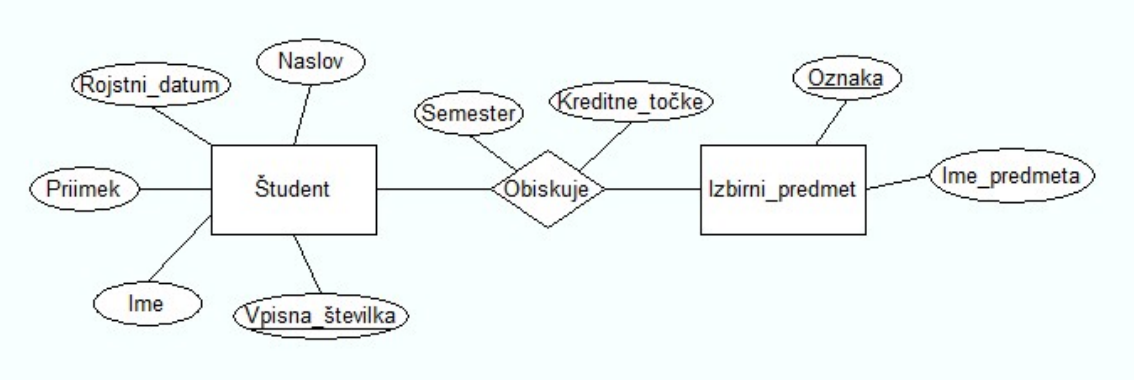
Slika 21: Entitetni tip "Študent"



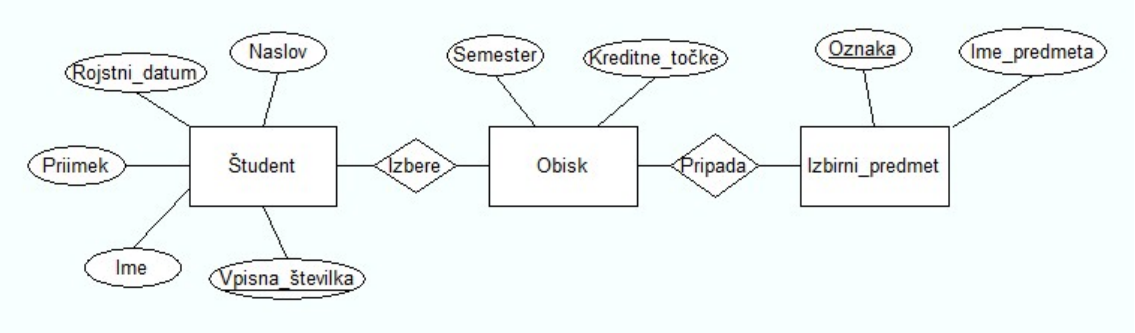
Slika 22: Ločena entitetna tipa "Študent" in "Naslov"

Entiteta ali razmerje

Včasih se lahko zgodi, da določeno značilnost označimo za razmerje, ko pa bi bilo bolj primerno, da jo obravnavamo kot samostojno entiteto. Načrtovanje entitetno relacijskega diagrama je namreč dokaj subjektiven proces, ki dopušča precej nenatančnosti. Tako lahko posamezen atribut kaj hitro pripišemo razmerju, namesto ustreznemu entitetnemu tipu. Takšne napake pa lahko vodijo do nepotrebnega podvajanja podatkov in povzročijo mnogo težav. Problem ponazorimo na primeru razmerja "Obiskuje", ki povezuje entititna tipa "Študent" in "Izbirni_predmet". Za takšno obiskovanje izbirnega predmeta študent prejme ustrezno število kreditnih točk. Zabeležimo pa tudi, v katerem semestru študent obiskuje posamezen izbirni predmet. Naraven izbor struktur je takšen, kot ga prikazuje slika 23. Razmerju pripadata dva atributa: "Semester" in "Kreditne_točke". Vendar pa se v primeru, če število kreditnih točk označuje skupno število vseh kreditnih točk, ki jih prejme študent, pojavi nepotrebno podvajanje podatkov. Tedaj se namreč število kreditnih točk ponovi pri razmerju "Obiskuje" za vsak izbirni predmet, ki ga obiskuje študent. Prav tako je skupno število kreditnih točk značilnost, ki je povezana s posameznim študentom in ne z obiskovanjem izbirnega predmeta. Tem težavam se izognemo z vpeljavo novega entitetnega tipa "Obisk", kjer se atributi razmerja "Obiskuje" prenesejo na nov entitetni tip "Obisk", kar je prikazano na sliki 24.



Slika 23: Obisk predstavljen kot razmerje



Slika 24: Obisk predstavljen kot entitetni tip

Relacijski podatkovni model

Značilnosti relacijskega podatkovnega modela

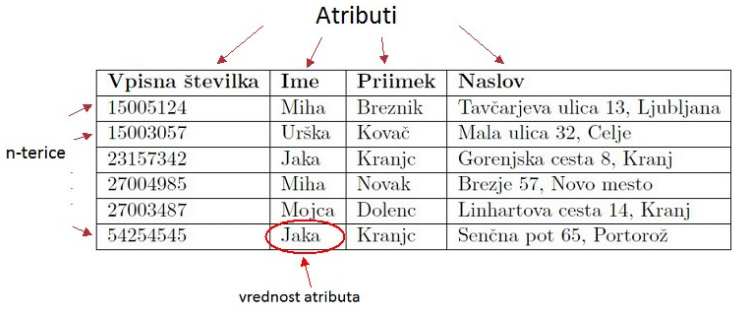
Relacijski podatkovni model predstavlja podatkovno bazo kot zbirko relacij. V praksi nam relacija predstavlja tabelo vrednosti, atributi pa so predstavljeni s stolpci tabele, kot je prikazano na sliki 25. V tabeli so zapisane vrstice, ki jih imenujemo n-terice relacije. Vrednosti atributov za določeno n-terico so zapisane v celicah vrstice. Posamezna relacija je torej množica n-teric. Iz tega dejstva sledi, da vrstni red vrstic znotraj tabele načeloma ni definiran in da v relaciji ni podvajanj n-teric (vrstic). Večina današnjih SUPB-jev kljub temu dovoljuje podvajanje vrstic tabele, dokler ta nima definiranega ključa.

Relacija je sestavljena iz relacijske sheme in stanja relacije. Relacijsko shemo si lahko predstavljamo kot glavo tabele, posamezen zapis pa kot vrstico v tabeli. Relacijska shema je opis relacije, torej vsebuje ime relacije, imena atributov in njihove domene. Stanje relacije pa je primer takšne tabele. Stanje relacije je torej množica zapisov, ki so zapisane v relaciji. Vsak zapis ima enako število polj kot relacijska shema. Stanje relacije je lahko:

- legalno (dovoljeno), kjer so vrednosti vseh atributov skladne z opredeljenimi integritetnimi omejitvami ali
- nelegalno (nedovoljeno).

Za legalnost stanja relacije v praksi skrbi SUPB.

Relacijska shema določa domeno za vsak atribut v relaciji. Domena D je množica nedeljivih (atomarnih) vrednosti atributa. Običajna metoda za določitev domene atributa je določitev tipa podatkov, iz katerega lahko za dani atribut zajemamo vrednosti. Vrednosti, ki se pojavijo v posameznem stolpcu, morajo namreč biti izbrane iz domene,



Vpisna številka	Ime	Priimek	Naslov
15005124	Miha	Breznik	Tavčarjeva ulica 13, Ljubljana
15003057	Urška	Kovač	Mala ulica 32, Celje
23157342	Jaka	Kranjc	Gorenjska cesta 8, Kranj
27004985	Miha	Novak	Brezje 57, Novo mesto
27003487	Mojca	Dolenc	Linhartova cesta 14, Kranj
54254545	Jaka	Kranjc	Sončna pot 65, Portorož

Slika 25: Relacija "Študent"

ki je določena za ta atribut. Izbor možnih zalog vrednosti določa SUPB in različni SUPB-ji ponujajo različne tipe podatkov. Osnovni tipi podatkov so nizi znakov, števila (cela, realna), datum, čas, valuta in binarni podatki. Določimo domene za atribute "Vpisna_številka", "Ime" in "Rojstni_datum" za relacijo "Študent":
Vpisna_številka: množica osemmestnih števil, ki predstavljajo veljavno vpisno številko
Ime: množica imen študentov
Rojstni_datum: množica veljavnih datumov

Relacijska podatkovna shema S je množica posameznih relacijskih shem $S = \{R_1, R_2, \ldots, R_m\}$ in integritetnih omejitev. Posamezno **relacijsko shemo** označimo z $R(A_1, A_2, \ldots, A_n)$. Določimo jo torej z imenom relacije R in seznamom atributov $A_1, A_2, \ldots, A_n$. **Stanje relacije** označimo z $r(R)$ in predstavlja množico n-teric $r = \{t_1, t_2, \ldots, t_n\}$. Vsaka n-terica je urejen seznam vrednosti $t = \langle v_1, v_2 \ldots, v_n \rangle$, kjer je vsaka vrednost v_i, $1 \leq i \leq n$ element iz domene atributa ali pa posebna vrednost null. Vrednost atributa A_i v n-terici t sicer označimo z oznako $t[A_i]$ oziroma $t.A_1$. Na atribut se lahko sklicujemo tudi z imenom atributa in relacije. Na primer na atribut "Ime" relacije "Študent" se lahko sklicujemo takole: ŠTUDENT.Ime.

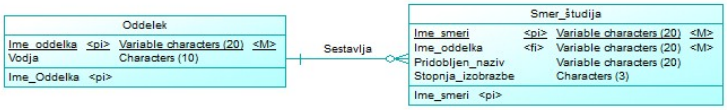
Vsaka relacija R je definirana kot množica enoličnih zapisov. Vsi elementi relacije morajo biti različni, to pomeni, da ne moreta imeti dva zapisa hkrati iste kombinacije vrednosti za vse podmnožice atributov. Označimo s SK neko poljubno neprazno množico atributov. Za dva različna zapisa t_1 in t_2 v stanju relacije r mora veljati: $t_1[SK] \neq t_2[SK]$. Takšna množica atributov SK se imenuje **superključ** relacijske sheme R. Superključ določa omejitev enoličnosti. Vsaka relacija ima vsaj en privzet superključ, in sicer množico vseh atributov.

Ključ relacije K je superključ v R, za katerega dodatno zahtevamo, da z odstranitvijo katerega koli atributa A iz ključa K dobimo množico atributov K', ki ni superključ v R. Ključ lahko definiramo tudi kot minimalno podmnožico atributov relacije, katerih vrednosti so za posamezen zapis enolične.

Relacijska shema ima lahko več kot en ključ, zato vse takšne ključe imenujemo **kandidati za ključ**. Izmed vseh kandidatov za ključ izberemo enega izmed njih za primarni ključ relacije. Običajo izberemo primarni ključ, ki ga sestavlja en atribut oziroma manjše število atributov. Entitetna omejitev določa, da vrednosti atributov primarnega ključa ne smejo biti null, saj mora primarni ključ enolično določati posamezen zapis v relaciji.

Informacije, shranjene v relaciji, so včasih povezane z informacijami, shranjenimi v drugi relaciji. Če spremenimo eno relacijo, potem moramo preveriti tudi drugo relacijo in jo po potrebi spremeniti, da ohranimo skladnost podatkov. V ta namen vpeljemo **referenčne omejitve**, ki povezujejo dve relaciji in zagotavljajo skladnost med zapisi teh dveh relacij. Referenčna omejitev določa, da se mora zapis v eni relaciji sklicevati na obstoječ zapis v drugi relaciji. Najpogostejša oblika referenčne omejitve je tuji ključ, ki povezuje dve relacijski shemi R_1 in R_2. Množica atributov FK v relacijski shemi R_1 je **tuji ključ** v R_1, ki se sklicuje na relacijo R_2, kadar zadošča naslednjim dvem lastnostim:

- Atributi v FK imajo iste domene kot atributi primarnega ključa PK v R_2. Za atribute v FK pravimo, da se nanašajo oziroma sklicujejo na relacijo R_2.
- Vrednosti atributov iz FK v zapisu t_1 iz $r(R_1)$ se bodisi pojavijo kot vrednosti v primarnem ključu PK za nek zapis t_2 v $r(R_2)$ ali pa so enake null. Seveda pa se



Slika 26: Tuji ključ, ki povezuje relaciji "Oddelek" in "Smer_študija"

lahko tuji ključ sklicuje tudi na svojo relacijo.

Primer tujega ključa, ki povezuje relaciji "Oddelek" in "Smer_študija" je predstavljen na sliki 26. Atribut "Ime_oddelka" v relaciji "Smer_študija" se sklicuje na atribut "Ime_oddelka" v relaciji "Oddelek", domeni obeh atributov pa sta enaki.

Algoritem za preslikavo v podatkovni model

V poglavju bomo obravnavali pretvorbo entitetno relacijske sheme v relacijsko shemo podatkovne baze. Obstajata dva razloga za preslikavo entitetno relacijskega modela v relacijski podatkovni model:

- SUPB neposredno ne podpira entitetno relacijskega modela in
- SUPB podpira relacijski ali objektno-relacijski podatkovni model.

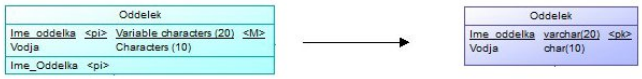
Sam postopek preslikave lahko opravimo na 2 načina:

- avtomatizirano – orodja CASE načeloma podpirajo avtomatizacijo procesa kreiranja podatkovne baze in s tem avtomatično preslikajo entitetno relacijski model v relacijski ali
- ročno – uporabimo pravila za preslikavo in naredimo sheme relacij.

Predstavili bomo osnutek algoritma, ki dano entitetno relacijsko shemo pretvori v relacijsko podatkovno shemo. Algoritem bomo razdelili na sedem korakov, povzetih po poglavju 9.1 v [3], za ponazoritev pa bomo uporabili entitetno relacijsko shemo "Fakulteta" na sliki 20.

Korak 1: Za vsak entitetni tip E v entitetno relacijski shemi ustvarimo relacijo R. Vsak enostaven atribut entitetnega tipa E postane atribut relacije R. V primeru sestavljenih atributov vključimo zgolj njegove enostavne komponente. Izberemo enega od ključev entitetnega tipa E kot primarni ključ relacije R. Če je izbrani ključ sestavljen, potem za ključ izberemo množico enostavnih atributov, ki ga sestavlja. Pri preslikavi ER sheme "Fakulteta" denimo entitetni tip "Oddelek" preslikamo v relacijo "Oddelek", ki je predstavljena na sliki 27.

Korak 2: Za vsak šibki entitetni tip W v entitetno relacijski shemi ustvarimo relacijo R in vanjo vključimo vse enostavne atribute oziroma enostavne komponente sestavljenih atributov. Atribute razmerja, ki povezuje šibki entitetni tip s starševskim entitetnim tipom, vključimo kot tuji ključ relacije R. Primarni ključ relacije R določimo kot kombinacijo primarnih ključev starševskega entitetnega tipa in morebitnih delnih ključev



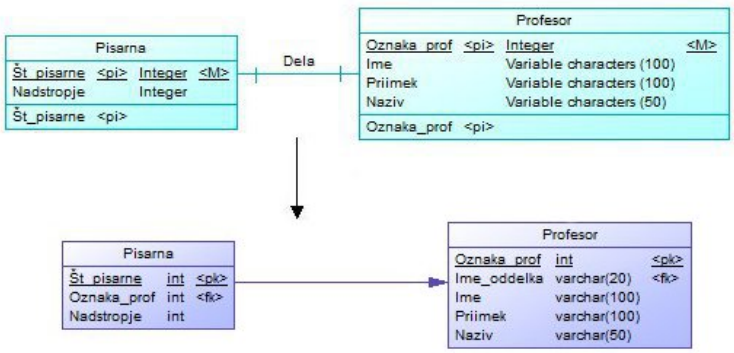
Slika 27: Preslikava entitetnega tipa "Oddelek"

Slika 28: Preslikava šibkega entitetnega tipa "Mladi_raziskovalec"

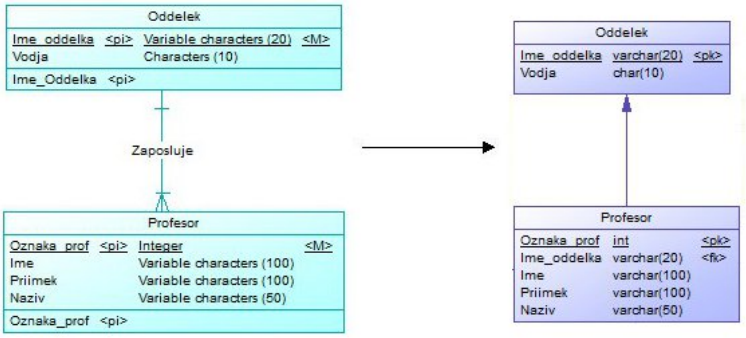
šibkega entitetnega tipa W. Denimo pri preslikavi ER sheme "Fakulteta" šibki entitetni tip "Mladi_raziskovalec" preslikamo v relacijo "Mladi_raziskovalec", ki je predstavljena na sliki 28.

Korak 3: Za vsako binarno $1:1$ razmerje R v entitetno relacijski shemi poiščemo relaciji S in T, ki predstavljata entitetna tipa, ki ju razmerje povezuje. Izberemo eno izmed relacij, denimo S in vanjo kot tuji ključ vključimo primarni ključ relacije T. Vse enostavne atribute in enostavne komponente sestavljenih atributov $1:1$ razmerja R vključimo kot atribute relacije S. Pri preslikavi ER sheme "Fakulteta" denimo razmerje "Dela" preslikamo tako, da v relacijo "Pisarna" vključimo tuji ključ, ki se navezuje na relacijo "Profesor", kar je prikazano na sliki 29.

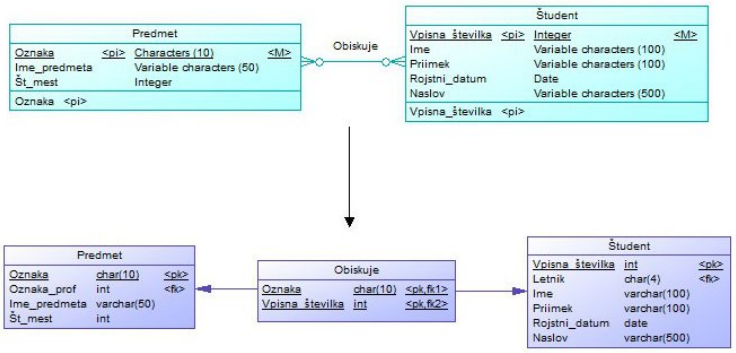
Korak 4: Za vsako binarno $1:N$ razmerje R v entitetno relacijski shemi poiščemo relaciji S in T, ki predstavljata entitetna tipa, ki ju razmerje povezuje. Za S izberemo relacijo na N-strani razmerja, za T pa relacijo na drugi strani razmerja. V relacijo S kot tuji ključ vključimo primarni ključ relacije T. Takšen postopek je ustrezen, saj je vsaka entiteta na N-strani povezana največ z eno entiteto na 1-strani razmerja. Vse enostavne atribute in enostavne komponente sestavljenih atributov $1:N$ razmerja R vključimo kot atribute relacije S. Pri preslikavi ER sheme "Fakulteta" na primer razmerje "Zaposluje"



Slika 29: Preslikava $1:1$ razmerja "Dela"



Slika 30: Preslikava $1 : N$ razmerja "Zaposluje"

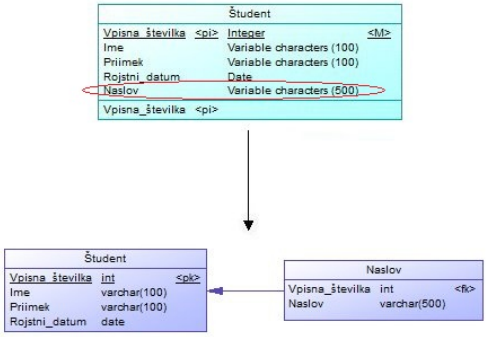


Slika 31: Preslikava $M : N$ razmerja "Obiskuje"

preslikamo tako, da v relacijo "Predmet" vključimo tuji ključ, ki se navezuje na relacijo "Oddelek", kar prikazuje slika 30.

Korak 5: Za vsako binarno $M : N$ razmerje R v entitetno relacijski shemi ustvarimo novo relacijo S, ki predstavlja razmerje R. Poiščemo relaciji, ki predstavljata sodelujoča entitetna tipa v razmerju R in njuna primarna ključa vključimo kot tuja ključa relacije S. Kombinacija obeh tujih ključev nam določa primarni ključ relacije S. Prav tako v relacijo S vključimo morebitne enostavne atribute oziroma enostavne komponente sestavljenih atributov $M : N$ razmerja R. Pri preslikavi ER sheme "Fakulteta" denimo razmerje "Obiskuje" preslikamo tako, da ustvarimo novo relacijo "Obiskuje". Vanjo vključimo tuja ključa, ki se navezujeta na relaciji "Predmet" in "Študent", kar prikazuje slika 31.

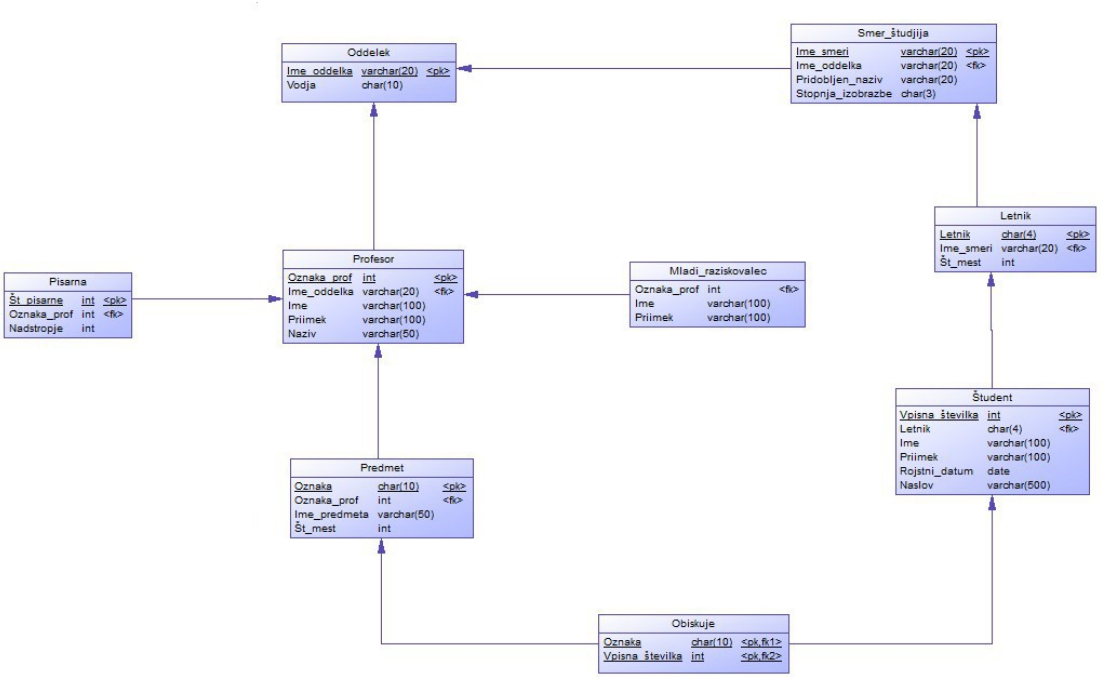
Korak 6: Za vsak večvrednostni atribut A v entitetno relacijski shemi ustvarimo novo relacijo R. V to relacijo vključimo atribut, ki predstavlja večvrednostni atribut A in dodatni tuji ključ K. Ta tuji ključ K je primarni ključ relacije, ki predstavlja entitetni tip, ki mu pripada atribut A. Primarni ključ nove relacije R je kombinacij atributa A in tujega ključa K. Če je atribut A sestavljen, vključimo zgolj njegove enostavne komponente. Takšen primer je denimo atribut "Naslov" entitetnega tipa "Študent". Pri preslikavi ustvarimo novo relacijo "Naslov", prikazano na sliki 32, ki je povezana s prvotno relacijo "Študent".



Slika 32: Preslikava atributa "Naslov"

Korak 7: Za vsako razmerje R, ki povezuje več kot dva entitetna tipa, ustvarimo novo relacijo S. Poiščemo relacije, ki predstavljajo sodelujoče entitetne tipe v razmerju R. Njihove primarne ključe vključimo kot tuje ključe relacije S. Prav tako vključimo morebitne enostavne atribute oziroma enostavne komponente sestavljenih atributov razmerja R. Primarni ključ relacije S je običajno kombinacija vseh tujih ključev, ki se navezujejo na relacije, ki predstavljajo sodelujoče entitetne tipe v razmerju R. Kadar je številnost kakšnega entitetnega tipa E, ki sodeluje v razmerju R, enaka 1, potem primarni ključ relacije S ne sme vsebovati tujega ključa, ki se navezuje na relacijo, ki predstavlja entitetni tip E.

V orodju PowerDesigner sem izvedla tudi avtomatsko pretvorbo entitetno relacijske sheme "Fakulteta" v relacijsko podatkovno shemo. Uporabila sem ukaz "Generate Physical shema" in izbrala sistem za upravljanje s podatkovnimi bazami MySQL 5.0. Ime ukaza je nekoliko zavajujoče, saj dejansko dobimo logično shemo. Gre le za različno poimenovanje orodja PowerDesigner, saj orodje pod pojmom logična shema (Logical shema) označuje shemo, ki je po lastnostih nekje med konceptualno in logično shemo. Končna dobljena relacijska podatkovna shema "Fakulteta2" je prikazana na sliki 33.



Slika 33: Relacijska podatkovna shema “Fakulteta2”

Pregled sheme in normalizacija

Pregled sheme

V fazi konceptualnega in logičnega načrtovanja je ustreznost oziroma kvaliteta načrtovanja odvisna predvsem od intuicije načrtovalca, saj ni določenega nobenega formalnega kriterija za ustreznost dobljene sheme. V fazi pregleda sheme pa lahko uporabimo teorijo, ki zagotavlja razvoj dobre relacijske podatkovne sheme. Predstavili bomo teorijo funkcionalnih odvisnosti in normalne oblike.

Ustreznost in kvaliteto dobljene sheme dosežemo, če:

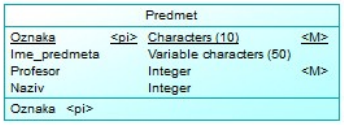
- preverimo poimenovanje (semantiko) atributov,
- zmanjšamo podvajanje podatkov in
- zmanjšamo število opcijskih atributov (vrednosti null).

Pri tem nam je predvsem pri zmanjševanju podvajanja podatkov v pomoč **normalizacija**. To je tehnika načrtovanja relacijskih podatkovnih baz, ki minimizira podvajanje podatkov in podatkovno bazo zavaruje pred anomalijami.

Poimenovanje atributov

Atributi v relacijski shemi imajo določen pomen. Poimenovanje oziroma semantika atributov nam določa, kako interpretiramo vrednosti atributov v zapisu. Če pazljivo izberemo poimenovanje atributov že v fazi konceptualnega načrtovanja, potem imajo atributi v shemi jasen pomen. V nasprotnem primeru pa moramo pregledati poimenovanje in razvrščanje atributov. Pri tem sta nam lahko v pomoč naslednja nasveta (povzeta po poglavju 14.1 v [3]):

- Relacije načrtujmo tako, da bo enostavno pojasniti njen pomen. Na primer v relaciji "Študent" na sliki 25 je jasno razvidno, kaj označujejo atributi "Vpisna_številka", "Ime", "Priimek" in "Naslov". Ustrezno poimenovanje atributov torej izboljša preglednost in razumljivost sheme.
- Ne mešajmo v eni relaciji atributov, ki sicer pripadajo različnim entitetnim tipom. Denimo v relacijski shemi "Predmet" na sliki 34 imamo atribute "Ime", "Oznaka", "Profesor" in "Naziv". Takšna zgradba sicer usteza zahtevam relacijske sheme, vendar jo vseeno obravnavamo kot primer slabega načrtovanja. V njej se namreč mešajo atributi profesorjev in predmetov.



Slika 34: Relacija “Predmet”

Vpisna_številka	**Ime_Priimek**	**Predmet**	**Kreditne_točke**
15005124	Mojca Kosec	Topologija I	20
15003057	Alenka Rožnik	Algebra II	30
23157342	Aleš Gočnik	Diferencialna geometrija	20
27004985	Vesna Krpan	Algebra II	30
16003487	Tomaž Polanec	Diferencialna geometrija	20
24003878	Miha Trpin	Diferencialna geometrija	20

Tabela 1: Primer relacije “ObiskPredmeta” s podvajanjem podatkov

Vrednosti null

Uporaba opcijskih atributov, ki imajo dovoljene nedoločene oziroma null vrednosti, lahko povzroči več težav. Vrednost null ima lahko v danem zapisu več možnih pomenov:

- atribut v zapisu ne nastopa,
- vrednost atributa v zapisu ni znana,
- vrednost atributa v zapisu še ni določena.

Vrednosti null lahko zasedejo nepotreben prostor in povzročijo težave pri izvajanju nekaterih operacij (na primer pri združevanju relacij). Prav tako atributi, ki imajo dovoljene vrednosti null, ne morejo biti del primarnega ključa. Kolikor je le možno se izognemo uporabi atributov, ki imajo pogosto vrednost null. Če se temu ne moremo izogniti, poskrbimo, da se vrednosti null pojavijo čim manjkrat, sploh pa ne v večini zapisov.

Podvajanje podatkov

Eden od ciljev načrtovanja je tudi zmanjšanje potrebnega pomnilniškega prostora za hranjenje podatkov v relaciji. Način razporejanja atributov v relacije ima velik vpliv na porabo pomnilnika. Napačno razvrščanje atributov ima lahko za posledico tudi anomalije, do katerih pride pri posodabljanju podatkov. Posledice anomalij so neskladnosti in dvoumnosti med podatki ali celo njihova izguba. Podatki, ki se podvajajo, torej se pojavijo na več mestih v podatkovni bazi, lahko povzročijo mnogo težav. Primer relacije s podvajanjem podatkov je prikazan v tabeli 1.

Težave, ki nastanejo zaradi nepotrebnega podvajanja podatkov, so:

- **Podvajanje prostora**: Informacije, ki so podvojene, potrebujejo več prostora za shranjevanje.

Vpisna_številka	Ime_Priimek	Predmet	Kreditne_točke
15005124	Mojca Kosec	Topologija I	20
15003057	Alenka Rožnik	Algebra II	25
23157342	Aleš Gočnik	Diferencialna geometrija	20
27004985	Vesna Krpan	Algebra II	30
16003487	Tomaž Polanec	Diferencialna geometrija	20
24003878	Miha Trpin	Diferencialna geometrija	20

Slika 35: Primer anomalije pri posodabljanju podatkov

Vpisna_številka	Ime_Priimek	Predmet	Kreditne_točke
15005124	Mojca Kosec	Topologija I	20
15003057	Alenka Rožnik	Algebra II	30
23157342	Aleš Gočnik	Diferencialna geometrija	20
27004985	Vesna Krpan	Algbra II	30
16003487	Tomaž Polanec	Diferencialna geometrija	20
24003878	Miha Trpin	Diferencialna geometrija	20

Slika 36: Primer anomalije pri vstavljanju podatkov

- Anomalije pri **posodabljanju podatkov**: Če spremenimo eno kopijo podvojenega podatka, moramo spremeniti tudi vse preostale kopije podatka, sicer pride do neskladja med podatki. Če se denimo na sliki 35 število kreditnih točk pri predmetu Algebra II spremeni iz 30 na 25, moramo to popraviti pri vseh študentih, ki obiskujejo predmet Algebra II. Kajti, če popravimo število kreditnih točk zgolj pri enem študentu, pride do neskladnosti podatkov.

- Anomalije pri **vstavljanju podatkov**: Nekaterih podatkov ne moremo shraniti, če ne shranimo tudi nekaj drugih podatkov, ki sicer niso povezani. Ne moremo denimo shraniti novega predmeta Programiranje I, če nima nobenega študenta, ki bi ga obiskoval. Vnos vseh podatkov o predmetu mora biti povsem točen, sicer pride do dvoumnih zapisov, kar je prikazano na sliki 36.

- Anomalije pri **brisanju podatkov**: Ni možno izbrisati nekaterih podatkov brez da ne bi zraven izgubili tudi druge podatke, ki sicer niso povezani. Če na primer izbrišemo vse študente, ki obiskujejo predmet Algebra II, izgubimo podatek o številu kreditnih točk pri predmetu, kar je vidno na sliki 37.

Večino problemov, ki nastanejo zaradi nepotrebnega podvajanja podatkov, lahko rešimo z dekompozicijo. **Dekompozicija** relacijske sheme R je nadomestitev relacijske sheme z dvema ali več relacijskimi shemami, kjer vsaka vsebuje podmnožico atributov

Vpisna_številka	Ime_Priimek	Predmet	Kreditne_točke
15005124	Mojca Kosec	Topologija I	20
15003057	Alenka Rožnik	Algebra II	30
23157342	Aleš Gočnik	Diferencialna geometrija	20
27004985	Vesna Krpan	Algebra II	30
16003487	Tomaž Polanec	Diferencialna geometrija	20
24003878	Miha Trpin	Diferencialna geometrija	20

Slika 37: Primer anomalije pri brisanju podatkov

Vpisna_številka	**Ime Priimek**	**Predmet**
15005124	Mojca Kosec	Topologija I
15003057	Alenka Rožnik	Algebra II
23157342	Aleš Gočnik	Diferencialna geometrija
27004985	Vesna Krpan	Algebra II
16003487	Tomaž Polanec	Diferencialna geometrija
24003878	Miha Trpin	Diferencialna geometrija

Tabela 2: Relacija "Obisk"

Predmet	**Kreditne točke**
Topologija I	20
Algebra II	30
Diferencialna geometrija	20

Tabela 3: Relacija "Predmet"

relacijske sheme R, vse relacijske sheme skupaj pa vsebujejo vse atribute relacijske sheme R.

Denimo, da relacijo "ObiskPredmeta", ki je prikazana v tabeli 1, z dekompozicijo razbijemo na relaciji "Obisk" in "Predmet", ki sta prikazani v tabelah 2 in 3. Sedaj lahko dodamo nov predmet in mu določimo število kreditnih točk, ne da bi potrebovali podatek o študentu, ki ta predmet obiskuje. Prav tako je posodabljanje podatka o številu kreditnih točk bolj učinkovito. Vrednost števila kreditnih točk spremenimo zgolj v relaciji "Predmet". Tudi pri brisanju vseh študentov, ki obiskujejo predmet Algebra II, v relaciji "Predmet" še vedno ohranimo podatke o predmetu Algebra II.

Vendar pa lahko z dekompozicijo, v kolikor nismo pazljivi, povzročimo več problemov kot koristi. Pomembno je, da si postavimo naslednji dve vprašanji: ali zares potrebujemo dekompozicijo določene relacije? Kakšne probleme lahko določena dekompozicija povzroči? V pomoč pri odgovoru na prvo vprašanje so nam normalne oblike (predstavljene v naslednjem poglavju), ki nam pomagajo pri odločitvi, ali je potrebno izvesti dekompozicijo. Pri drugem vprašanju moramo preveriti predvsem dve lastnosti: ali lahko iz dekompozicije sestavimo nazaj originalno relacijo in ali lahko iz dekompozicije učinkovito pregledamo integritetne omejitve. Potrebno pa je razmisliti tudi o učinkovitosti podatkovne baze. Poizvedbe po originalni relaciji namreč zahtevajo od nas, da povežemo razbite relacije. Če so takšne poizvedbe pogoste, potem je posledica dekompozicije zmanjšanje zmogljivosti podatkovne baze. V kolikor je takšno zmanjšanje za uporabnike nesprejemljivo, se raje odločimo za osnovno tabelo in ne izvedemo dekompozicije. Pomembno pa je, da se ob tem zavedamo problemov, ki jih lahko povzroči podvajanje podatkov. Tem problemom se lahko izognemo tako, da v aplikacijski kodi dodamo dodatne pogoje. V nekaterih primerih pa lahko dekompozicija celo izboljša zmogljivost podatkovne baze. To je denimo takrat, kadar se večina poizvedb in posodobitev nanaša zgolj na eno od tabel iz razbitja, ki je seveda manjša od originalne tabele.

Vpisna_številka	**Ime**	**Priimek**	**Naslov**
15005124	Miha	Breznik	Tavčarjeva ulica 13, Ljubljana
15003057	Urška	Kovač	Mala ulica 32, Celje
23157342	Jaka	Kranjc	Gorenjska cesta 8, Kranj
27004985	Miha	Novak	Brezje 57, Novo mesto
27003487	Mojca	Dolenc	Linhartova cesta 14, Kranj
54254545	Jaka	Kranjc	Senčna pot 65, Portorož

Tabela 4: Primer stanja relacije "Študent"

Funkcionalne odvisnosti

Funkcionalne odvisnosti so glavni koncept, povezan s postopkom normalizacije podatkovne baze. Koraki normalizacije namreč temeljijo na določenih pravilih, ki se nanašajo na funkcionalne odvisnosti med atributi tabele. Pravilno opredeljene funkcionalne odvisnosti na začetku postopka normalizacije so tako pogoj za uspešno izvedbo normalizacije.

Funkcionalna odvisnost je odvisnost med dvema množicama atributov podatkovne baze. Naj bo R relacijska podatkovna shema, X in Y pa naj bosta dve neprazni množici atributov iz R. Pravimo, da stanje r relacije R zadošča funkcionalni odvisnosti $X \rightarrow Y$, če za vsak par zapisov t_1 in t_2 v r velja: če je $t_1.X = t_2.X$, potem je tudi $t_1.Y = t_2.Y$. Oznaka $t_1.X$ pomeni projekcijo zapisa t_1 na množico atributov X. Ta omejitev pomeni, da so vrednosti Y komponent zapisa v r odvisne od vrednosti X komponent. Pri funkcionalni odvisnosti $X \rightarrow Y$ pravimo tudi, da množica atributov X funkcionalno določa množico atributov Y, oziroma, da je množica atributov Y funkcionalno odvisna od množice atributov X. Funkcionalne odvisnosti se poglavitno uporabljajo za dodatni opis relacijske sheme. Z njimi lahko določimo omejitve atributov, ki vedno držijo. Funkcionalno odvisnost ponazorimo na primeru relacije "Študent". Oglejmo si stanje relacije "Študent" v tabeli 4. V njem lahko določimo funkcionalno odvisnost {*Vpisna_številka*} $\rightarrow$ {*Ime,Priimek*}. Atribut "Vpisna_številka" je v relaciji "Študent" primarni ključ, torej enolično določa zapis. To pomeni, da so vrednosti vseh ostalih atributov določene z vrednostjo atributa "Vpisna_številka". Tako atribut "Vpisna_številka" določa vrednosti atributov "Ime" in "Priimek" in zato funkcionalna odvisnost {*Vpisna_številka*} $\rightarrow$ {*Ime,Priimek*} drži.

Z vpogledom v zgolj eno stanje relacije lahko ugotovimo, da določena funkcionalna odvisnost ne drži. Denimo iz stanja relacije "Študent" v tabeli 4 lahko ugotovimo, da funkcionalna odvisnost {*Ime,Priimek*} $\rightarrow$ {*Naslov*} ne drži. Vrednosti atributov "Ime" in "Priimek" ne določata enolično vrednost atributa "Naslov". Obstajata namreč dva zapisa z enakim imenom in priimkom (Jaka Kranjc), vendar pa imata različen naslov. Ne moremo pa na podlagi zgolj enega stanja relacije sklepati, da določena funkcionalna odvisnosti drži, saj je funkcionalna odvisnost značilnost vseh možnih stanj relacije. Denimo iz primera stanja relacije "Študent" lahko napačno sklepamo, da drži funkcionalna odvisnost {*Naslov*} $\rightarrow$ {*Ime,Priimek*}. Vendar pa je povsem možno, da v nekem stanju relacije obstajata študenta z enakim naslovom, vendar različnim imenom in priimkov. Poseben primer funkcionalne odvisnosti je primarni ključ. Atributi primarnega ključa predstavljajo množico X, množica vseh ostalih atributov relacije pa predstavlja množico Y. V danem primeru relacije "Študent" je primarni ključ atribut "Vpisna_številka", ki predstavlja

Profesor	**Predmet**	**Zap_št**	**Datum**	**Ura**	**Predavalnica**	**Št_mest**
Moder J.	Verjetnostni račun	1	12.2.2012	10:00	A7	30
Moder J.	Verjetnostni račun	2	14.6.2012	12:00	A4	50
Moder J.	Verjetnostni račun	3	29.6.2012	11:00	A4	50
Novak B.	Analiza II	2	14.6.2012	10:00	A1	80
Novak B.	Analiza II	3	2.7.2012	16:00	A7	30
Novak B.	Analiza I	2	7.6.2012	10:00	A1	80
Novak B.	Analiza I	3	28.6.2012	9:00	A2	60
Novak B.	Analiza I	4	15.9.2012	11:00	A2	60

Tabela 5: Relacija "Izpitni_rok"

množico X. Atribut "Vpisna_številka" funkcionalno določa vse preostale atribute, to je množico $Y = \{Ime, Priimek, Naslov\}$.

Obstajajo tri vrste funkcionalnih odvisnosti: delna funkcionalna odvisnost, polna funkcionalna odvisnost in tranzitivna funkcionalna odvisnost. Delno funkcionalno odvisnost bomo ponazorili na primeru relacije "Ogled_izpita", prikazane v tabeli 6, polno in tranizitivno pa na primeru relacije "Izpitni_rok", prikazane v tabeli 5.

Delna funkcionalna odvisnost $X \to Y$ je funkcionalna odvisnost, pri kateri lahko kakšnega od atributov iz množice X odstranimo in je funkcionalna odvisnost še vedno izpolnjena. Takšen primer je funkcionalna odvisnost $\{Profesor,\ Predmet,\ Zap\_št\} \to \{Kabinet\}$ iz tabele 6. Velja namreč, da je vrednost atributa "Kabinet" odvisna zgolj od vrednosti atributa "Profesor". Torej tudi če iz množice $X = \{Profesor,\ Predmet,\ Zap\_št\}$ odstranimo denimo atribut "Zap_št", bo funkcionalna odvisnost $\{Profesor,\ Predmet\} \to \{Kabinet\}$ še vedno izpolnjena.

Polna funkcionalna odvisnost $X \to Y$ je funkcionalna odvisnost, pri kateri odstranitev katerega koli atributa iz množice X pomeni, da funkcionalna odvisnosti ni več izpolnjena. Primer polne funkcionalne odvisnosti je $\{Profesor,\ Predmet,\ Zap\_št\} \to \{Datum\}$ iz tabele 5. Če iz množice $X = \{Profesor,\ Predmet,\ Zap\_št\}$ odstranimo en atribut, denimo "Zap_št", dobimo množico $X_1 = \{Profesor,\ Predmet\}$. Množica X_1 pa funkcionalno ne določa atributa "Datum". Obstajata namreč zapisa (prvi in drugi zapis), ki imata enako vrednost atributov "Profesor" in "Predmet", vendar različno vrednost atributa "Datum".

Tranzitivna funkcionalna odvisnost $X \to Y$ je funkcionalna odvisnost, za katero obstaja množica atributov Z, ki ni možen ključ relacije, prav tako ni podmnožica kakšnega ključa relacije, in velja $X \to Z$ in $Z \to Y$. Primer tranzitivne funkcionalne odvisnosti je odvisnost $\{Profesor,\ Predmet,\ Zap\_št\} \to \{Št\_mest\}$ iz tabele 5. Obstaja namreč množica atributov $Z = \{Predavalnica\}$, ki ni del nobenega ključa relacije, vendar pa velja $\{Profesor,\ Predmet,\ Zap\_št\} \to \{Predavalnica\}$ in $\{Predavalnica\} \to \{Št\_mest\}$ (število mest na izpitnem roku je določeno z vrednostjo atributa "Predavalnica").

Normalne oblike

Proces normalizacije, ki ga je vpeljal Codd leta 1972, za dano relacijsko shemo izvede niz preizkusov. Ti preizkusi potrdijo, ali relacijska shema zadošča določenim normalnim oblikam. Namreč, kadar je relacijska shema v kateri od normalnih oblik, se določeni tipi

Profesor	**Predmet**	**Zap_št**	**Datum**	**Ura**	**Kabinet**
Moder J.	Verjetnostni račun	1	15.2.2012	8:00	32
Moder J.	Verjetnostni račun	2	18.6.2012	15:00	32
Moder J.	Verjetnostni račun	3	2.7.2012	13:00	32
Novak B.	Analiza II	2	18.6.2012	15:00	16
Novak B.	Analiza II	3	5.7.2012	10:00	16
Novak B.	Analiza I	2	10.6.2012	14:00	16
Novak B.	Analiza I	3	3.7.2012	9:00	16
Novak B.	Analiza I	4	19.9.2012	11:00	16

Tabela 6: Relacija "Ogled_izpita"

Profesor	**Predmet**	**Smer**
Moder J.	Seminar I	Pedagoška smer
Novak B.	Analiza III	Teoretična smer, Uporabna smer
Perko K.	Računalništvo II	Uporabna smer
Jazbec M.	Fizika	Pedagoška smer, Uporabna smer, Teoretična smer

Tabela 7: Primer relacije "Predmetnik", ki ni v 1NF

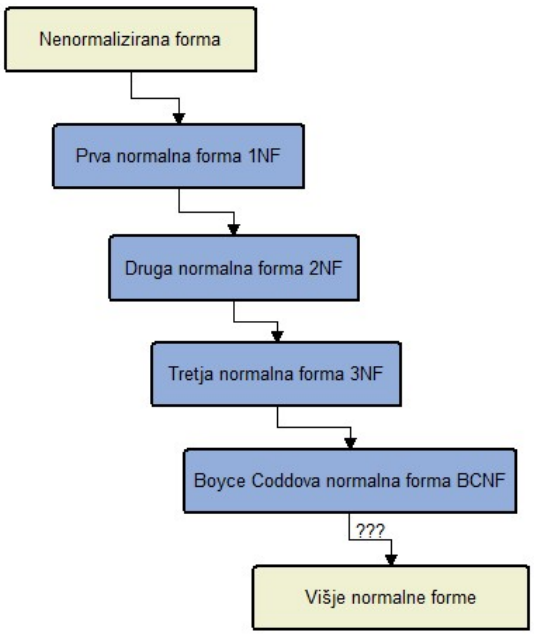
problemov v shemi ne morejo pojaviti. Sheme relacij, ki ne zadoščajo določeni normalni obliki, razdelimo na več manjših relacij, tako da dobljene relacije ustrezajo normalni obliki. Codd je vpeljal tri normalne oblike, imenovane prva (1NF), druga (2NF) in tretja normalna oblika (3NF). Kasneje sta Codd in Boyce na podlagi tretje oblike definirala močnejšo različico, Boyce-Coddovo normalno obliko (BCNF). Vse te normalne oblike temeljijo na funkcionalnih odvisnostih med atributi relacije. Zanje velja, da je vsaka relacija, ki je v BCNF, vsebovana tudi v 3NF. Prav tako je vsaka relacija, ki je v 3NF, vsebovana tudi v 2NF in vsaka relacija, ki je v 2NF, je vsebovana tudi v 1NF. Kasneje so vpeljali še višje stopnje normalizacije, ki pa jih v nalogi ne obravnavamo. Formalni postopek normalizacije je pregledno predstavljen na sliki 38. Pri postopku normalizacije je izhodišče nenormalizirana podatkovna baza, ki jo nato pretvorimo v prvo normalno obliko, nato v drugo normalno obliko, tretjo normalno obliko in po potrebi v Boyce-Coddovo in višje normalne oblike. Višje stopnje normalizacije imajo običajno za posledico več relacij in lahko zmanjšajo učinkovitost podatkovne baze, zato moramo pri uporabi normalizacije ravnati racionalno glede na potrebe uporabnikov.

Prva normalna oblika (1NF)

Relacija R je v **prvi normalni obliki** (1NF), če in samo če vse domene atributov relacije vsebujejo le atomarne vrednosti. Prva normalna oblika torej prepoveduje, da bi kakšen zapis imel za vrednost posameznega atributa množico več vrednosti.

Primer relacije, ki ni v 1NF, je relacija "Predmetnik", prikazana v tabeli 7. V njej imajo nekateri zapisi za vrednost atributa "Smer" množico več vrednosti. Denimo za drugi zapis v relaciji ima atribut "Smer" vrednost {Teoretična smer, Uporabna smer}.

Kadar relacija R ni v prvi normalni obliki, obstajajo tri glavne metode, kako doseči



Slika 38: Formalni postopek normalizacije

Profesor	**Predmet**
Moder J.	Seminar I
Novak B.	Analiza III
Perko K.	Računalništvo II
Jazbec M.	Fizika

Tabela 8: Relacija "Predmetnik" po dekompoziciji

prvo normalno obliko (povzeto po poglavju 14.3.2 v [3]).

1. **Dekompozicija relacije**, ki ni v 1NF, v dve relaciji, ki sta v 1NF
   Odstranimo atribut, ki krši 1NF in ga prenesemo v novo relacijo skupaj s primarnim ključem prvotne relacije. V primeru relacije "Predmetnik" odstranimo atribut "Smer" in ga prestavimo v novo relacijo skupaj z atributom "Predmet", ki je primarni ključ prvotne relacije. Dobimo torej dve ločeni relaciji: prvotno relacijo "Predmetnik", prikazano v tabeli 8, in novo relacijo "Predmetnik_smer", prikazano v tabeli 9.

2. Če je znano maksimalno število vrednosti atributa, **atribut razdelimo** v več ločenih atributov. Ta možnost lahko proizvede nezaželene vrednosti null.

   Pri relaciji "Predmetnik" v tabeli 7 privzamemo, da so možne največ tri različne smeri. Atribut "Smer" zato razdelimo v tri ločene atribute: "Smer1", "Smer2" in "Smer3". Pri vseh zapisih, ki imajo manj kot tri smeri, preostalim atributom pripišemo vrednosti null, kar je prikazano v tabeli 10. Opazimo, da pri tem dobimo veliko vrednosti null.

Predmet	**Smer**
Seminar I	Pedagoška smer
Analiza III	Teoretična smer
Analiza III	Uporabna smer
Računalništvo II	Uporabna smer
Fizika	Pedagoška smer
Fizika	Uporabna smer
Fizika	Teoretična smer

Tabela 9: Nova relacija "Predmetnik_smer" po dekompoziciji

Profesor	**Predmet**	**Smer1**	**Smer2**	**Smer3**
Moder J.	Seminar I	Pedagoška smer	null	null
Novak B.	Analiza III	Teoretična smer	Uporabna smer	null
Perko K.	Računalništvo II	Uporabna smer	null	null
Jazbec M.	Fizika	Pedagoška smer	Uporabna smer	Teoretična smer

Tabela 10: Primer relacije "Predmetnik" po razdelitvi atributa "Smer"

3. **Razširimo ključ** in za vsako od vrednosti atributa ustvarimo svoj zapis. Ta možnost lahko proizvede nehoteno podvajanje podatkov. Denimo v relaciji "Predmetnik" v tabeli 11 razširimo primarni ključ na atributa "Predmet" in "Smer" in relacijo dopolnimo z novimi zapisi.

   Izmed vseh možnosti je v splošnem najbolj primerna izbira prva možnost, saj ne proizvaja nepotrebnega podvajanja, prav tako ne potrebuje določitve maksimalnega števila vrednosti atributa. Seveda pa je v določenih primerih narava podatkov takšna, da je morda ustreznejša druga ali tretja možnost.

Druga normalna oblika (2NF)

Druga normalna oblika temelji na konceptu polne funkcionalne odvisnosti. Drugo normalno obliko bomo najprej definirali le za primarne ključe, nato pa bomo definicijo posplošili na vse kandidate za ključ. Relacija R je v **drugi normalni obliki** (2NF), če

Profesor	**Predmet**	**Smer**
Moder J.	Seminar I	Pedagoška smer
Novak B.	Analiza III	Teoretična smer
Novak B.	Analiza III	Uporabna smer
Perko K.	Računalništvo II	Uporabna smer
Jazbec M.	Fizika	Pedagoška smer
Jazbec M.	Fizika	Uporabna smer
Jazbec M.	Fizika	Teoretična smer

Tabela 11: Primer relacije "Predmetnik" po razširitvi ključa

Profesor	**Predmet**	**Dan**	**Ura**	**Kabinet**
Moder J.	Seminar I	Sreda	10:00	23
Novak B.	Analiza III	Četrtek	12:00	34
Perko K.	Računalništvo II	Sreda	9:00	12
Perko K.	Uvod v programiranje	Ponedeljek	11:00	12
Jazbec M.	Fizika	Torek	14:00	17

Tabela 12: Primer relacije "Govorilna_ura", ki ni v 2NF

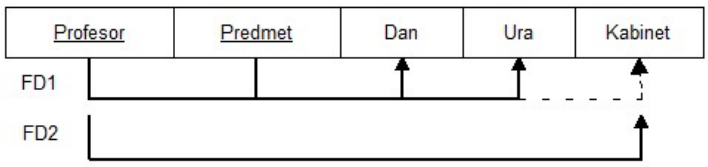
Profesor	**Kabinet**
Moder J.	23
Novak B.	34
Perko K.	12
Jazbec M.	17

Tabela 13: Nova relacija "Kabinet"

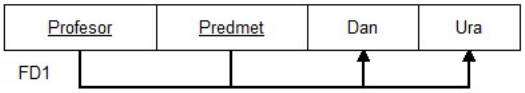
in samo če je v prvi normalni obliki in je vsak atribut, ki ni del nobenega kandidata za ključ, polno funkcionalno odvisen od primarnega ključa v R. Relacija R torej ni v 2NF, kadar obstaja atribut, ki ni del nobenega ključa in je le delno funkcionalno odvisen od primarnega ključa. Kadar relacija R ni v drugi normalni obliki poiščemo takšne atribute in jih skupaj z delom primarnega ključa, od katerega so ti atributi polno funkcionalno odvisni, prestavimo v novo relacijo. Oglejmo si primer relacije "Govorilna_ura" v tabeli 12, ki predstavlja tedenski razpored govorilnih ur za profesorje, ločeno glede na posamezne predmete, ki jih poučujejo. Primarni ključ sestavljata atributa "Profesor" in "Predmet". Atribut "Kabinet" ni del nobenega ključa, odvisen pa je zgolj od atributa "Profesor". Kabinet, kjer poteka govorilna ura, je odvisen zgolj od profesorja, ne pa tudi od predmeta, ki ga profesor poučuje. Atribut "Kabinet" je tako zgolj delno funkcionalno odvisen od primarnega ključa, zato krši 2NF. Skupaj z atributom "Profesor" ga prestavimo v novo relacijo "Kabinet", kar je prikazano v tabeli 13.

Shematično lahko shemo relacije "Govorilna_ura" z označenimi funkcionalnimi odvisnostmi prikažemo tudi tako, kot je podano na sliki 39. Relacijski shemi po razdelitvi pa sta prikazani na slikah 40 in 41.

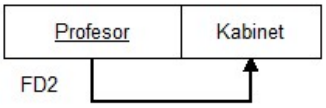
V splošnem želimo izdelati takšno relacijsko shemo, ki nima niti delnih niti tranzitivnih odvisnosti (obravnavnani so v 3NF), saj ti tipi odvisnosti povzročijo anomalije pri



Slika 39: Relacijska shema, ki ni v 2NF



Slika 40: Relacijska shema relacije "Govorilna_ura", ki je v 2NF



Slika 41: Relacijska shema nove relacije "Kabinet"

posodabljanju podatkov. V definiciji druge normalne oblike smo delne odvisnosti izločili zgolj za primarne ključe, ne pa tudi za preostale kandidate za ključ. S tem namenom posplošimo definicijo 2NF na vse kandidate za ključ.

Posplošitev definicije druge normalne oblike: Relacija R je v drugi normalni obliki (2NF), če in samo če je v prvi normalni obliki in je vsak atribut, ki ni del nobenega kandidata za ključ, polno funkcionalno odvisen od vsakega ključa v R.

Tretja normalna oblika (3NF)

Tretja normalna oblika temelji na konceptu tranzitivne odvisnosti. Tretjo normalno obliko bomo podobno kot drugo normalno obliko najprej definirali le za primarne ključe, nato pa definicijo posplošili na vse kandidate za ključ. Relacija R je v **tretji normalni obliki** (3NF), če in samo če je v drugi normalni obliki in za vsak atribut, ki ni del nobenega kandidata za ključ velja, da ni tranzitivno odvisen od primarnega ključa. Relacija R torej ni v tretji normalni obliki, kadar obstaja atribut, ki ni del nobenega kandidata za ključ in je tranzitivno odvisen od primarnega ključa. Za takšen atribut velja, da ga funkcionalno določa drug atribut, ki tudi ni del nobenega kandidata za ključ. Kadar relacija R ni v 3NF, poiščemo takšne atributi, ki kršijo 3NF in jih prestavimo v novo relacijo. Oglejmo si primer relacije "Izbira_predavanja", predstavljene v tabeli 14. Relacija predstavlja izbiro predavanja na izobraževalnem dnevu na fakulteti, kjer vsak študent izbere eno predavanje, ki se izvaja v določeni predavalnici. Primarni ključ predstavlja atribut "Vpisna_številka". Atribut "Predavalnica" ni del nobenega kandidata za ključ in je funkcionalno določen z atributom "Predavanje", ki tudi ni del nobenega kandidata za ključ. Torej je atribut "Predavalnica" tranzitivno odvisen od primarnega ključa in zato krši 3NF. Skupaj z atributom "Predavanje" ga prestavimo v novo relacijo, kar je prikazano v tabeli 15.

Shematično lahko shemo relacije "Izbira_predavanja" z označenimi funkcionalnimi odvisnostmi prikažemo tudi tako, kot na sliki 42. Relacijski shemi po razdelitvi pa sta prikazani na slikah 43 in 44.

Iz enakih razlogov kot pri drugi normalni obliki tudi pri tretji normalni obliki posplošimo definicijo na vse kandidate za ključ.

Vpisna_številka	**Ime**	**Priimek**	**Predavanje**	**Predavalnica**
15005124	Miha	Breznik	Objektno programiranje	A2
15003057	Urška	Kovač	Elementarna geometrija	B3
23157342	Jaka	Kranjc	Numerične metode	A1
27004985	Miha	Novak	Podatkovne baze	A2
27003487	Mojca	Dolenc	Topološki prostori	B2
54254545	Jaka	Kranjc	Podatkovne baze	A2

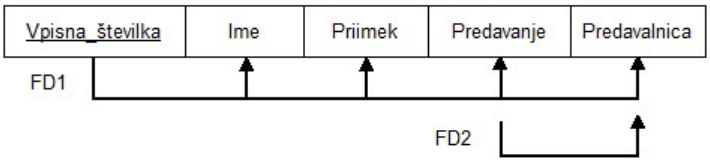
Tabela 14: Primer relacije "Izbira_predavanja", ki ni v 3NF

Vpisna_številka	**Ime**	**Priimek**	**Predavanje**
15005124	Miha	Breznik	Objektno programiranje
15003057	Urška	Kovač	Elementarna geometrija
23157342	Jaka	Kranjc	Numerične metode
27004985	Miha	Novak	Podatkovne baze
27003487	Mojca	Dolenc	Topološki prostori
54254545	Jaka	Kranjc	Podatkovne baze

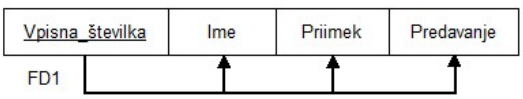
Tabela 15: Relacija "Izbira_predavanja", ki je v 3NF

Predavanje	**Predavalnica**
Objektno programiranje	A2
Elementarna geometrija	B3
Numerične metode	A1
Podatkovne baze	A2
Topološki prostori	B2

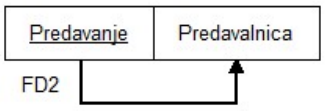
Tabela 16: Nova relacija "Predavanje"



Slika 42: Relacijska shema, ki ni v 3NF



Slika 43: Relacijska shema relacije "Izbira_predavanja", ki je v 3NF



Slika 44: Relacijska shema nove relacije "Predavanje"

Profesor	**Predmet**	**Asistent**
Moder J.	Teorija množic	Pleško M.
Novak B.	Analiza III	Maršič B.
Perko K.	Računalništvo II	Matko K.
Perko K.	Uvod v programiranje	Drešar M.
Perko K.	Računalništvo II	Matko K.
Jazbec M.	Fizika	Jagodnik R.

Tabela 17: Primer relacije "Asistent", ki je v 3NF, ni pa v BCNF

Posplošitev definicije tretje normalne forme: označimo z F množico vseh funkcionalnih odvisnosti v relacijski shemi R. Naj bo X neka podmnožica atributov v R in naj bo A atribut v R. Relacijska shema R je v tretji normalni obliki, če in samo če je v drugi normalni obliki in za vsako funkcionalno odvisnost $FD : X \to Y$ iz F velja ena od navedenih trditev:

- $A \in X$; FD je torej trivialna funkcionalna odvisnost.
- X je superključ.
- A je del kakšnega ključa iz R.

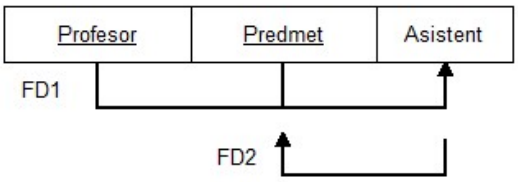
Boyce-Coddova normalna oblika (BCNF)

Relacija R je v **Boyce-Coddovi normalni obliki** (BCNF), če in samo če je v tretji normalni obliki in velja: če je $X \to A$ netrivialna funkcionalna odvisnost v R, potem je X superključ v R. V praksi večina relacij, ki zadoščajo 3NF, zadoščajo tudi BCNF. Relacija R, ki je v 3NF, ne bo v BCNF le takrat, kadar v R velja $X \to A$ za nek X, ki ni superključ, A pa je del kandidata za ključ.

Primer relacije, ki je v 3NF, ni pa v BCNF je relacija "Asistent", prikazana v tabeli 17, ki predstavlja asistente pri določenem profesorju za nek predmet. Pri tem predpostavimo, da je vsak asistent lahko dodeljen le k enem predmetu. Shematično je relacijska shema relacije "Asistent" prikazana na sliki 45. Iz nje lahko razberemo dve funkcionalni odvisnosti: $FD1 : \{Profesor,\ Predmet\} \to \{Asistent\}$ in $FD2 : \{Asistent\} \to \{Predmet\}$. Množica atributov $\{Profesor,\ Predmet\}$ predstavlja primarni ključ relacije. Opazimo, da funkcionalna odvisnost $FD2$ krši BCNF, ker ni trivialna in atribut "Asistent" ni superključ, saj določa zgolj vrednost atributa "Predmet".

Dekompozicija relacijske sheme v dve relacijski shemi pa ni takoj očitna, saj je možno relacijo "Asistent" razbiti na tri načine:

1. {Profesor, Asistent} in {Profesor, Predmet}



Slika 45: Relacijska shema, ki ni v BCNF

2. {Predmet, Asistent} in {Predmet, Profesor}
3. {Asistent, Predmet} in {Asistent, Profesor}

Izmed vseh treh možnosti je najprimernejša tretja možnost, saj ob združevanju obeh relacij ne bo proizvedla sumljivih zapisov.

Literatura

[1] R. Ramakrishnan, J. Gehrke, Database management systems-third edition. New York, McGraw-Hill, 2003.

[2] T. Mohorič, Podatkovne baze. Ljubljana, BI-TIM, 2002.

[3] R. Elmasri, S.B. Navathe, Fundamentals of database systems-third edition. Reading, Addison-Wesley, 2000.

[4] M. Reingruber, W.M. Gregory, The Data Modelling Handbook. New York, John Wiley& Sons, 1994.

[5] T. Mohorič, Načrtovanje relacijskih podatkovnih baz. Ljubljana, BI-TIM, 1997.

[6] S. Bagui, R. Earp, Database Design Using Entity-Relationship Diagrams. New York, Auerbach Publications, 2003.

[7] Sybase, PowerDesigner Conceptual Data Model Getting started, Version 9.5, dostopno na spletnem naslovu: [`http://download.sybase.com/pdfdocs/pdd0950e/cdgs.pdf`](dostop januar 2012).

[8] e-gradiva za predmet Računalništvo, dostopno na spletnem naslovu: [`http://colos1.fri.uni-lj.si/ERI/RACUNALNISTVO`] (dostop december 2011).