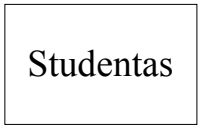
1 Esybių - Ryšių (E-R) modelis

1.1 E-R Modelio komponentai

E-R modelis sudaro bazę **E-R diagramoms,** kurios atitinka konceptualią duomenų bazę, kaip kad ją įsivaizduoja vartotojas. Pagrindiniai E-R diagramų komponentai yra **esybės** *(entities)*, **ryšiai** *(relationships)* ir **atributai** *(attributes)*.

1.2 Esybės

Esybės - tai gerai skiriami fiziškai ar mintyse egzistuojanyts modeliuojamo pasaulio konceptai. Panašios esybės sudaro esybių aibę (esybių tipą), kuris dažniausiai vadinamas tiesiog esybe. Esybės žymimos stačiakampiais, viduje užrašant esybės vardą:



1 pav. Esybė *Studentas*

Taigi esybė E-R modelyje atitnka reliacinės aplinkos lentelę, o ne eilutę. (Eilutę lentelėje atitiktų vienas konkretus esybės egzempliorius - *entity instance, entity occurrence*).

Pavyzdžiai: *Studentas*, *Dėstytojas*, *Paskaita*, *Detalė*, *Tiekėjas*, ...

Ieškant esybių, verta atkreipti dėmesį į daiktavardžius (veiksnius sakinyje), pavyzdžiui, skaitant dalykinės srities aprašymą ar užduoties specifikaciją.

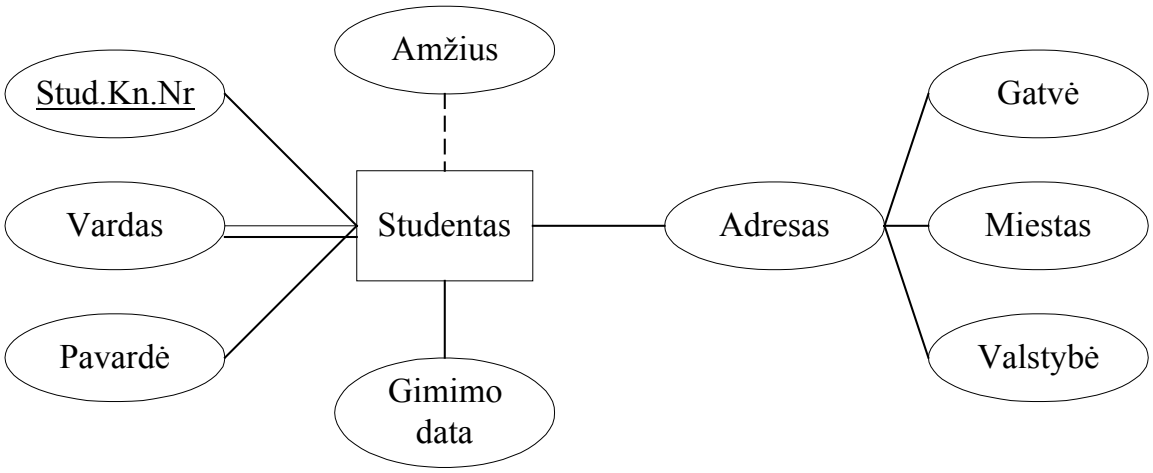
1.3 Atributai

Atributai charakterizuoja esybes (arba ryšius). Vaizduojami ovalais, kurių viduje rašomas atributo vardas. Ovalai sujungiami su atitinkama esybe.

Pavyzdžiai (Studentui): *Vardas*, *Pavardė*, *Stud.Kn.Numeris*, *Gimimo metai*

Ieškant atributų, verta atkreipti dėmesį į būdvardžius (pažyminius).

Atributai turi **domenus** (sritis). Domeną sudaro visos galimos atributo reikšmės. Pvz, atributo *Studento Įvertinimas* domenas yra skaičių aibė {1,2,3,4,5,6,7,8,9,10}. Atributai gali turėti vienodus domenus (pvz, *Studento Adresas* ir *Dėstytojo Adresas*).



2 pav. Esybės *Studentas* atributai (eskizas)

Raktiniai atributai *(key attributes)*, t.y. atributai, įeinantys į pirminį raktą, yra pabraukiami.

Atributai gali būti skirstomi į **paprastus** ir **sudėtinius**. Sudėtiniai atributai (nemaišyti su sudėtiniais raktais!) - atributai, kurie gali būti toliau skaidomi, gaunant papildomus atributus. Pavyzdžiui, atributas *Adresas* gali būti išskaidytas į *Gatvę, Miestą, Valstybę, Indeksą*. Paprasti atributai yra neskaidomi (pvz, *Amžius*, *Lytis*).

Pastaba: Reliaciniame modelyje, atributai negali būti sudėtiniai (išskyrus, galbūt, laiką ir datą), todėl, atvaizduojant E-R diagramą į lenteles, sudėtinius atributus gali tekti išskaidyti į sudedamąsias dalis.

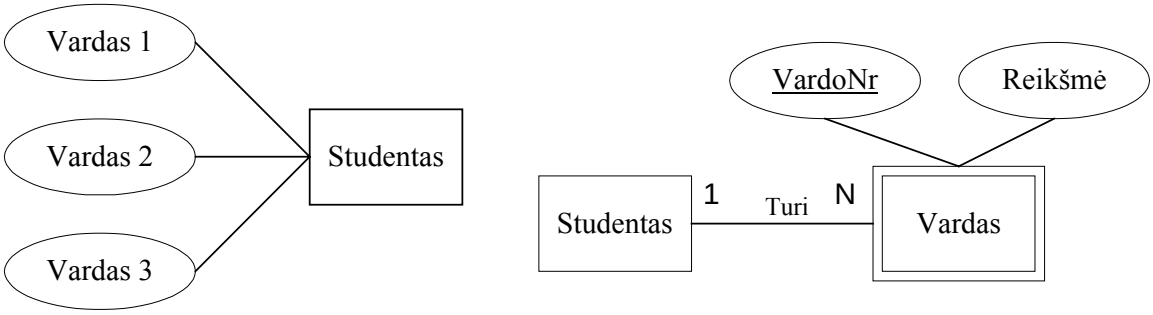
Vienos reikšmės atributai turi tik vieną reikšmę. Pvz, asmuo gali turėti tik vieną asmes kodą, gaminama dalis - tik vieną serijinį numerį. Vienos reikšmės atributai nebūtinai turi būti paprasti - jie gali būti ir sudėtiniai (pvz, asmens kode užkoduota lytis, gimimo data, ...)

Daugelio reikšmių atributai turi keletą reikšmių. Pvz, asmuo gali turėti kelis vardus, būti baigęs keletą universitetų, kambaryje gali būti keli telefonai su skirtingais numeriais. Tokie atributai su atitinkama esybe jungiami dviguba linija.

Pastaba: Reliaciniame modelyje, atributų reikšmės negali būti aibės ar lentelės, todėl, atvaizduojant E-R diagramą į lenteles, gali tekti pasirinkti vieną iš kelių sprendimų:

1. Originalios esybės viduje sukurti naujus atributus - po vieną kiekvienam originalaus daugelio reikšmių atributo komponentui.
2. Sukurti naują esybę, susidedančią iš originalaus daugelio reikšmių atributo komponentų.

3 pav. parodyta, kaip galima pakeisti daugelio reikšmių atributą *Vardas*. Beje, antrasis variantas leidžia turėti daugiau nei tris vardus.



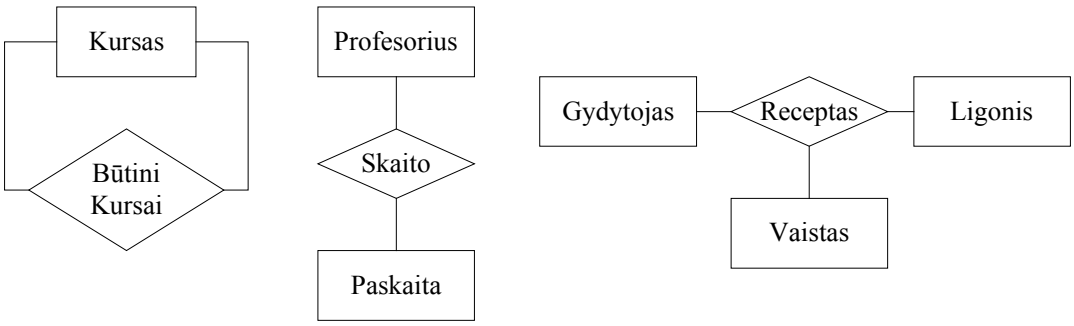
3 pav. Du daugelio reikšmių atributo *Vardas* išskaidymo variantai

Yra dar ir **išvestiniai** atributai. Tokie atributai duomenų bazėje gali fiziškai neegzistuoti, o būti išskaičiuojami (išvedami) iš kitų atributų. Pvz, *Asmens Amžius* gali būti išskaičiuotas, žinant asmens gimimo bei einamąją datas. Išvestinis atributas skiriamas pagal punktyrinę liniją, jungiančią atributą ir esybę.

1.4 Ryšiai

Ryšys - tai asociacija tarp dviejų esybių. Kiekvienas ryšys pavadinamas taip, kad vardas atspindėtų ryšio esmę. Pvz, studentas lanko užsiėmimus, profesorius veda užsiėmimus, etc. Kaip ir su esybėmis, galima būtų skriti konkrečius ryšius (tarp konkrečių esybių) ir ryšių tipus (tarp esybių tipų). Paprastai ryšių tipai vadinami tiesiog ryšiais. Ryšiai žymimi rombais, kurių viduje rašomas ryšio vardas. Rombas jungiamas su esybėmis, tarp kurių ir yra aprašomas ryšys.

Ryšio **laipsnis** parodo ryšyje dalyvaujančių esybių skaičių. **Unariniai** ryšiai - tai ryšiai vienos esybės viduje (pvz, *Būtini Kursai* - nurodo, kokius kursus reikia išklausyti, norint klausyti konkretų kursą). Tokie ryšiai dar vadinami **rekursyviais**. **Binarinis** ryšys bus tada, kai jungiamos dvi esybės. Šis ryšys yra dažniausiai pasitaikantis. **Ternarinis** ryšys jungia tris esybes. Ryšiai tarp keturių ir daugiau esybių yra itin reti.



4 pav. Unarinio, binarinio ir ternarinio ryšio pavyzdžiai

Binariniai ryšiai yra dažniausi. Kad supaprastinti konceptualųjį projektavimą, daugelį aukštesnės eilės ryšių galima konvertuoti į keletą binarinių ryšių. Tačiau kartais taip galima kiek pakeisti ryšio semantiką (prarasti informacijos). Pvz, ternarinis ryšys *Ligonis - Gydytojas - Vaistas* negali būti išskaidytas į tris binarinius ryšius, neprarandant informacijos.

Ryšį R tarp esybių E_1, E_2, ..., E_n galima suprasti kaip sąryšį matematine prasme. Tuomet ryšys R yra esybių E_1, E_2, ..., E_n dekarto sandaugų poaibis. n - ryšio laipsnis:

$$R \subseteq E_1 \times E_2 \times ... \times E_n$$

Elementas $(e_1, e_2, ..., e_n) \in R$ vadinamas ryšio egzemplioriumi (*instance*), kur $e_i \in E_i$ kiekvienam $1 \leq i \leq n$.

Galima apibrėžti ir **rolės** sąvoką. Panagrinėjus ryšį *Būtini Kursai* matosi, kad reikia tiksliau nurodyti, ką kuri esybė ryšyje reiškia:

$$Būtini\ Kursai \subseteq Kursas \times Kursas$$

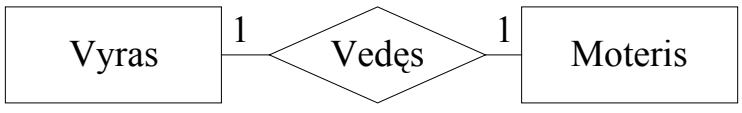
Kad charakterizuoti egzempliorių $(k_1, k_2) \in Būtini\ Kursai$, reikalingos rolės: $(Prieš{:}k_1, Po\ to{:}k_2) \in Būtini\ Kursai$. Iš čia visiškai aišku, kad kursas k_1 yra būtina sąlyga kursui k_2.

1.5 Jungiamumas (Funkcionalumas)

Ryšiai gali būti klasifikuojami į vienas-su-vienu (1:1), vienas-su-daug (1:N) ir daug-su-daug (N:M). Jungiamumas (*connectivity*) arba funkcionalumas (*functionality*) naudojamas klasifikacijai nusakyti.

- 1:1 ryšys bus tada, kai kiekvienai esybei e_1 iš E_1 gali būti priskiriama daugiausiai viena esybė e_2 iš E_2 ir atvirkščiai, kiekvienai esybei e_2 iš E_2 gali būti priskiriama daugiausiai viena esybė e_1 iš E_1. Beje, gali būti esybių iš E_1 (E_2), kurios neturi priskirto partnerio iš E_2 (E_1).

  Pavyzdys: Tarp esybių *Vyras* ir *Moteris* gali būti nusakytas ryšys *Vedęs*. Tai bus 1:1 ryšys (bent jau pagal Europoje priimtas normas).



5 pav. 1:1 ryšys

- 1:N ryšys bus tada, kai kiekvienai esybei e_1 iš E_1 gali būti priskiriama kiek norima daug (arba iš viso nei vienos) esybių iš E_2, bet kiekvienai esybei e_2 iš E_2 gali būti priskiriama daugiausiai viena esybė e_1 iš E_1.

  Pavyzdys: Tarp esybių *Firma* ir *Asmuo* gali būti nusakytas ryšys *Dirba*. Tai bus

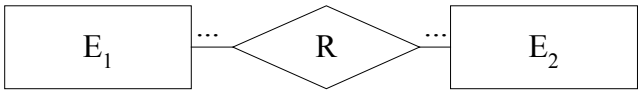
1:N ryšys, jei su sąlyga, kad daug asmenų darbuojasi vienos firmos labui, bet vienas asmuo dirba tik vienai firmai.

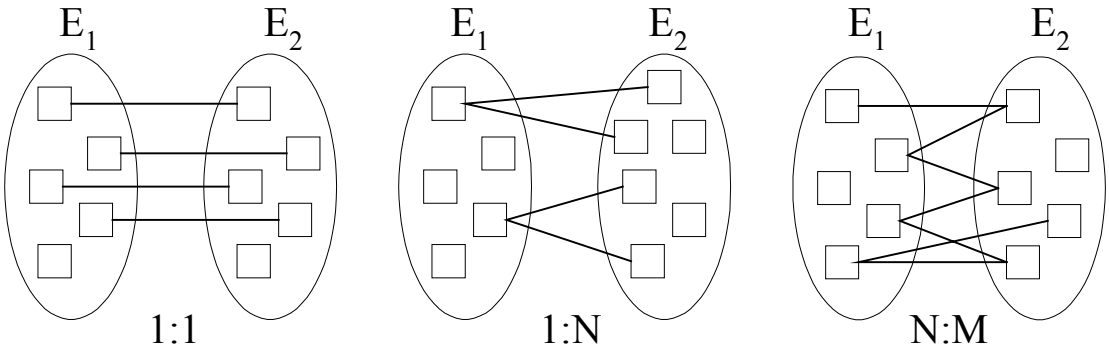
6 pav. 1:N ryšys

- N:M ryšiui nėra jokių apribojimų, t.y. kiekviena esybė e_1 iš E_1 gali būti susijusi su kiek norima daug esybių iš E_2 ir atvirkščiai, kiekviena esybė iš E_2 gali būti asocijuota su kiek norima daug esybių iš E_1.

7 pav. N:M ryšys

Beje, jungiamumas - tai integralumo sąlygos, kurios modeliuojamame pasaulyje turi būti teisingos visada, ne tik esant vienai konkrečiai modeliuojamo pasaulio būsenai.





8 pav. Ryšių 1:1, 1:N, N:M schemos

Į binarinius 1:1, 1:N ir N:1 sąryšius galima žiūrėti ir kaip į **dalines funkcijas**.

Į 1:1 sąryšį R tarp esybių E_1 ir E_2 galima žiūrėti kaip į dalinę funciją R: $E_1 \rightarrow E_2$. Be to, apibrėžta ir dalinė funkcija R^{-1}: $E_2 \rightarrow E_1$. Iš pavyzdžio turėtume:

Žmona: Vyras → Moteris
Vyras(sutuoktinis): Moteris → Vyras

Ryšyje 1:N kryptis bus griežtai apibrėžta: iš “N”-esybės į “1”-esybę. Buvusiame pavyzdyje būtų:

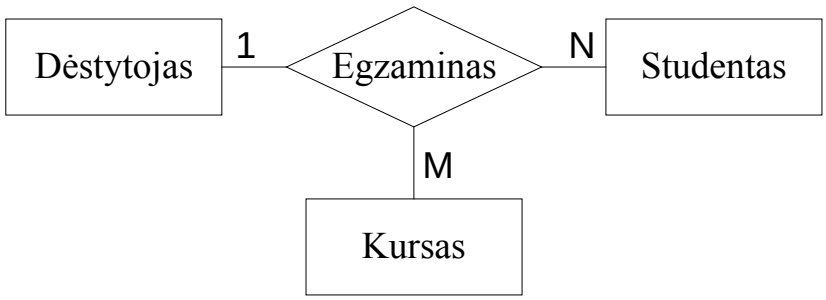
Dirba: Asmuo $\rightarrow$ *Firma*

Funkcijos yra dalinės, nes gali būti esybių iš apibrėžimo srities, kurios visiškai nedalyvauja ryšyje (žr. 8 pav.).

Jungiamumo nurodymą galima praplėsti bet kokiems n-tojo laipsnio ryšiams. Tegu R - ryšys tarp esybių E_1, E_2, ..., E_n. Tuomet jungiamumas ties esybe E_k ($1 \le k \le n$) turėtų būti žymimas “1” (ties kitomis esybėmis - “1” arba kokia nors unikalia raide), jei R apibrėžia tokią dalinę funkciją:

$$R: E_1 \times ... \times E_{k-1} \times E_{k+1} \times ... \times E_n \rightarrow E_k$$

Tokie funkciniai sąryšiai, suprantama, turi galioti visoms esybėms, kurios pažymimos “1”. Aukščiau apibrėžti jungiamumai yra šio apibrėžimo atskiri atvejai.



9 pav. Ternarinis ryšys su nurodytais funkcionalumais

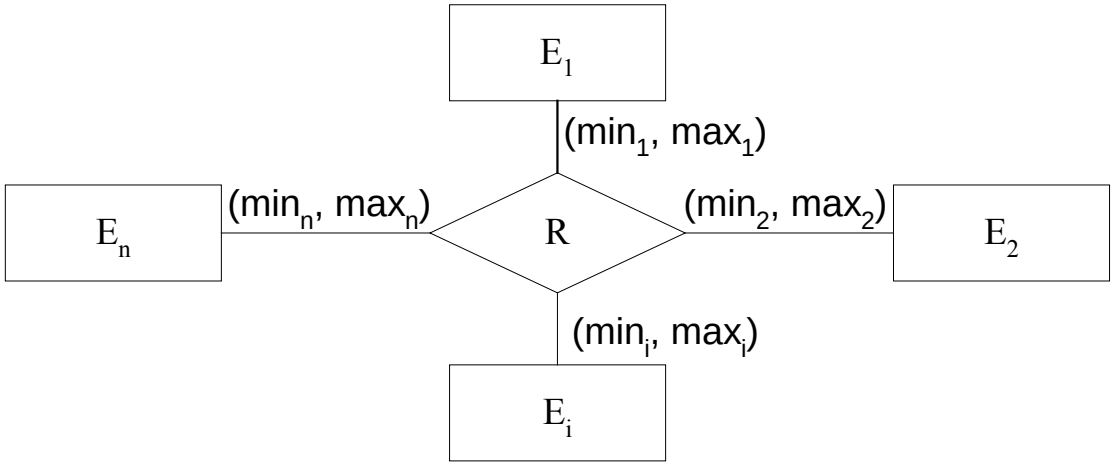
9 pav. užfiksuota sąlyga, kad konkretų studentą iš konkretaus kurso egzaminuoja tik vienas dėstytojas. Jei ši sąlyga negalioja, tai *Dėstytojo* pusėje taip pat turi būti simbolis “daug”.

1.6 Kardinalumas

Šiame skyrelyje aprašomas formalizmas leidžia dar tiksliau charakterizuoti ryšį, nei tai daroma jungiamumo pagalba. Jei tik viena esybė yra asocijuojama su daugiau nei viena esybe, jungiamumo atveju prie pastarosios žymima “N”. *(min, max)* žymėjimai (kardinalumas) nurodo kiek tikslesnes - viršutinę ir apatinę - ribas.

Kiekvienai ryšyje dalyvaujančiai esybei priskiriama skaičių pora *(min, max)*. Ši pora nurodo, kad kiekviena šio tipo esybė mažiausiai *min* kartų ir daugiausiai *max* kartų dalyvauja minėtame ryšyje.

Kiek formaliau tai atrodytų taip:



10 pav. Abstraktus n-tojo laipsnio ryšys su kardinalumais

Jei R - abstraktus n-tojo laipsnio ryšys tarp esybių E_1, E_2, ..., E_n, tai teisingas sąryšis

$$R \subseteq E_1 \times E_2 \times ... \times E_n$$

Žymėjimas (min_i, max_i) reiškia, kad kiekvienam $e_i \in E_i$ aibėje R egzistuoja mažiausiai min_i ir daugiausiai max_i eilučių, turinčiu pavidalą (..., e_i, ...) (kiekvienam $1 \le i \le n$).

Beje, kaip ir jungiamumo (funkcionalumo) atveju, kardinalumas atspindi dėsnius, galiojančius modeliuojamame pasaulyje, o ne aprašinėja kažkokią (atsitiktinę) tuo momentu esančią duomenų būseną.

Atskiri atvejai aprašomi taip:

- Jei gali būti esybių, kurios visiškai nedalyvauja ryšyje, tai *min* lygus 0.
- Jei esybė gali kiek norima daug kartų dalyvauti ryšyje, *max* lygus *.

Pavyzdžiai:

11 pav. 1:N ryšys su nurodytais kardinalumais

11 pav. pavaizduotas ryšys tarp *Dėstytojo* ir *Kurso* - *Skaito*. Nurodytus kardinalumus reikėtų suprasti taip:

- Dėstytojas per semestrą gali skaityti iki 3 kursų. Taip pat jis gali ir visai neskaityti jokio kurso.
- Vieną kursą skaito lygiai vienas dėstytojas. Jei būtų galima, kad kursą skaitytų keli dėstytojai, tai kardinalumas būtu, pvz, *(1, *)*.

12 pav. N:M ryšys su nurodytais kardinalumais

12 pav. pavaizduotas ryšys tarp *Studento* ir *Kurso* - *Klauso*. Nurodytus kardinalumus reikėtų suprasti taip:

- Studentas privalo klausyti bent 1 kursą per semestrą, tačiau jis neturi klausyti daugiau kaip 6 kursų.
- Kursą gali klausyti kiek norima daug studentų arba kursas tą semestrą gali būti visai nedėstomas. Taigi užrašas (0, *) reiškia, kad esybė gali kiek norima kartų dalyvauti ryšyje arba visai jame nedalyvauti.

Palyginus jungiamumo (1:N, N:M) ir kardinalumo *(min, max)* užrašus, matosi, jog tam tikra prasme jie yra "priešingi". Jei kuri nors esybė žymima simboliu "daug" (N), tai, nurodant kardinalumą, skaičius *max* (arba *) "keliauja" prie priešingos esybės.

1.7 Egzistavimo priklausomybė ir silpnosios esybės

Iki šiol buvo laikoma, kad esybės egzistuoja nepriklausomai viena nuo kitos (autonomiškai) ir tarp to paties tipo esybių yra vienareikšmiškai nusakomos raktu (raktinių atributų kombinacija). Realiame pasaulyje yra dar ir **silpnųjų esybių**, kurioms tai negalioja.

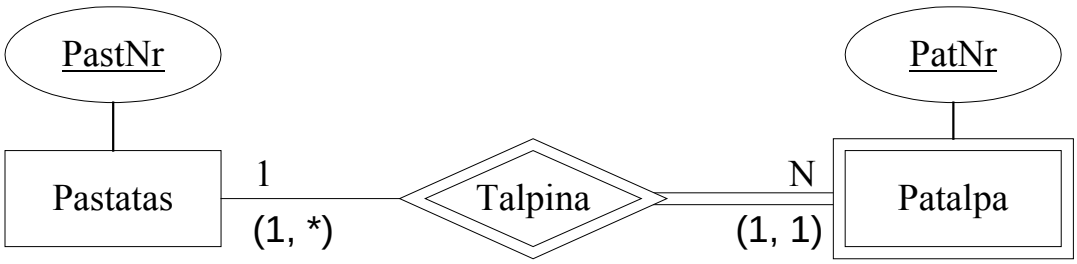
Silpnoji esybė pasižymi savybėmis:

- Egzistavimas priklauso nuo kitos (pagrindinės, aukštesnės) esybės
- Pati esybė vienareikšmiškai identifikuojama tik prie rakto prijungus kitos (pagrindinės) esybės raktą.

Silpnosios esybės žymimos dvigubu stačiakampiu. Ryšys su pagrindine esybe taip pat žymimas dvigubu rombu, rombą ir silpnąją esybę jungia dviguba linija.

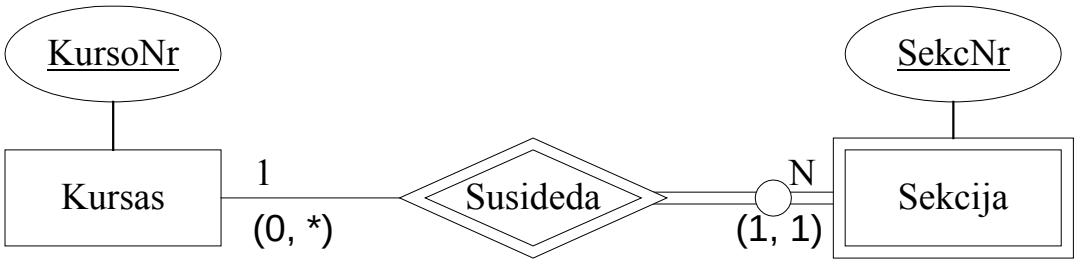
Ryšio tarp silpnosios esybės ir pagrindinės esybės jungiamumas dažniausiai būna 1:N, kartais 1:1. N:M jungiamumo būti negali. (Kodėl?)

Pavyzdžiai:



13 pav. Egzistavimo priklausomybė ir silpnoji esybė

13 pav. pavaizduota silpnoji esybė *Patalpa.* Egzistavimo priklausomybė akivaizdi - nugriovus pastatą, dingsta visos patalpos. Beje, patalpą vienareikšmiškai identifikuoti galima tik žinant jos ir pastato, kuris ją talpina, numerius.



14 pav. Egzistavimo priklausomybė ir silpnoji esybė

Jei universitetas didelis, vieną kursą gali skaityti keli dėstytojai skirtingiems studentų srautams (pvz, Matematinė Analizė dėstoma matematikams ir informatikams atskirai). Taigi kursas susideda iš sekcijų (14 pav.). Tačiau kursas nebūtinai turi būti skaidomas į sekcijas. Tai pažymima skrituliuku *Sekcijos* pusėje. (Jei esybės dalyvavimas ryšyje būtinas, skrituliuko nėra. Plačiau - skyrelyje “Dalyvavimas Ryšyje”).

1.8 Dalyvavimas ryšyje

Nagrinėjant ryšio tipą tarp dviejų esybių tipų, galima pastebėti, kad esybės dalyvavimas ryšyje gali būti **privalomas** arba **neprivalomas (nebūtinas)** (*mandatory* - *optional*). Esybės dalyvavimas ryšyje bus neprivalomas, kai gali būti konkrečių kito tipo esybių, kurios nėra susiję su jokia nagrinėjamo tipo esybe. Pvz, 11 pav. *Dėstytojas* gali neskaityti jokio *Kurso* arba 14 pav. *Kursas* gali neturėti *Sekcijų*. Tuomet 11 pav. esybės *Kursas* dalyvavimas ryšyje *Skaito* yra neprivalomas *Dėstytojo* atžvilgiu; esybės *Sekcija* (14 pav.) dalyvavimas ryšyje *Susideda* yra neprivalomas *Kurso* atžvilgiu.

Neprivalomas dalyvavimas ryšyje žymimas mažu skrituliuku prie neprivalomos esybės, kaip kad parodyta 14 pav.

Yra svarbu suprasti skirtumą tarp privalomo ir neprivalomo dalyvavimo ryšyje. Nuo to, ar esybių dalyvavimas yra privalomas ar ne, dažnai priklauso, kaip bus realizuojamas ryšys, o nuo ryšio realizacijos - ar galima sukurti naują esybės egzempliorių.

14 pav. parodyta, kad *Sekcijos* dalyvavimas ryšyje *Susideda* yra neprivalomas *Kurso* atžvilgiu. Tai reiškia, kad galima pradžiai sukurti konkretų *kursą*, o vėliau išskaidyti jį į *sekcijas*. (T.y. pradžiai *kursas* neturi *sekcijų*). Jeigu Sekcijos dalyvavimas būtų privalomas, tai konkretų *kursą* galima būtų sukurti tik sukūrus bent vieną *sekciją*.[1]

1.9 Generalizavimas ir specializavimas

Kad būtų galima struktūrizuoti esybės, įvedama generalizavimo/specializavimo sąvoka.

Generalizavimo metu išskiriamos bendros panašių esybių savybės (atributai ir ryšiai, kuriuose dalyvauja esybės) ir priskiriamos atskirai (bendrai) esybei - **supertipui**. Panašios esybės, iš kurių buvo išskirtos bendros savybės, vadinamos supertipo **potipiais** (subtipais, supertipo kategorijomis). Tos savybės, kurios nėra bendros visiemos potipiams, lieka prie to potipio, kuriam iš pradžių ir priklausė.

Galimas ir atvirkščias procesas: jei konkrečiame esybių tipe išsiskiria esybių poaibiai, kurių viduje esybės pasižymi tomis pačiomis savybėmis, tačiau poaibiai vienas nuo kito skiriasi, tai specializavimo metu pradinė esybė išskaidoma į bendrąja dalį (supertipą) ir specializacijas (potipius).

Esminė supertipų ir potipių sąryšio savybė - **paveldimumas**. Potipis paveldi visas savo supertipo charakteristikas (t.y. atributus ir ryšius). Taigi konkrečios potipio esybės tuo pačiu yra ir supertipo esybės. Paveldimumas - tai specialus ryšio tipas ("is-a" ryšys).

Potipio esybių aibė yra supertipo esybių aibės poaibis. Tai yra natūralu, kadangi visos potipių esybės yra konkretaus supertipo esybės (bet ne atvirkščiai!).

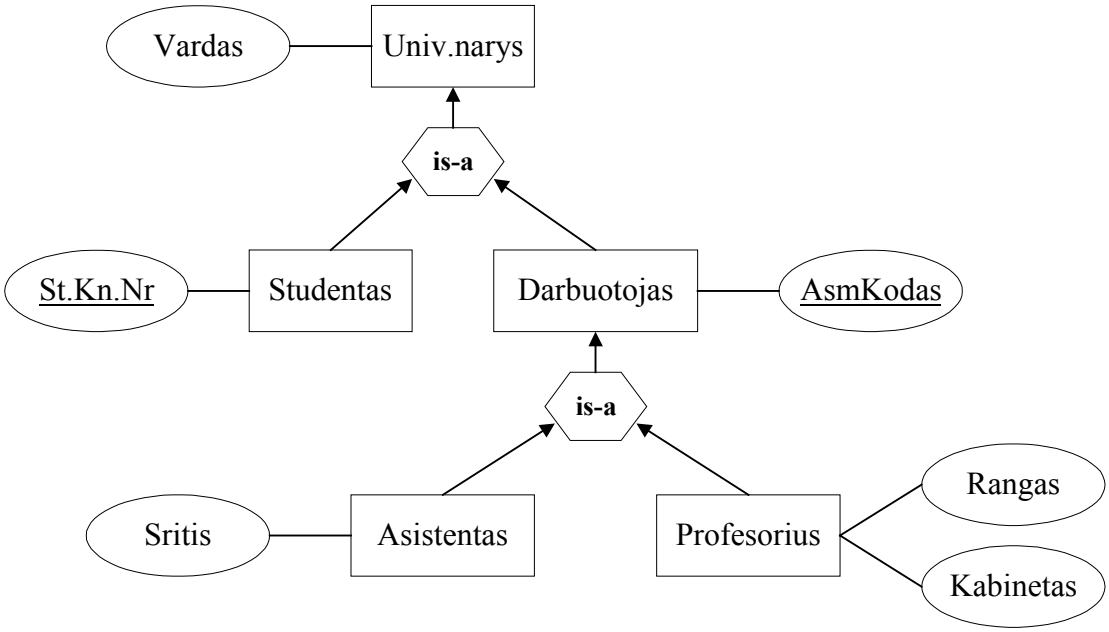
Galima būtų išskirti du generalizavimo/specializavimo atvejus.

- Disjunktyvus (nepersidengiantis) specializavimas būna tada, kai potipių aibės yra poromis skirtingos.
- Pilnas specializavimas būna tada, kai supertipo esybių aibė neturi tiesioginių egzempliorių, t.y. susideda tik iš potipių egzempliorių.

15 pav. pavaizduotas *Universiteto Narių* specializavimas į *Studentus* ir *Darbuotojus* yra pilnas (nes visi nariai yra griežtai arba studentai, arba darbuotojai) ir disjunktyvus (nes darbuotojas negali tuo pačiu būti ir studentas). *Darbuotojų* specializavimas į *Asistentus* ir *Profesorius* yra disjunktyvus, bet ne pilnas, nes gali būti ir kitokių darbuotojų tipų - pvz., *Sekretorė* arba *Elektrikas*. Jeigu būtų įvestas dar vienas potipis

[1] Bendru atveju, tai priklauso nuo to, kaip ši vieta realizuojama konkrečioje sistemoje.

supertipui *Darbuotojas* - *Administratorius* , tai tarp *Profesoriaus* ir *Administratoriaus* būtų persidengimas (pvz, katedros vedėju gali būti tik profesorius; katedros vedėjas - tai administratorius!)



15 pav. *Universiteto Narių* generalizavimas/specializavimas

2 E-R diagramos atvaizdavimas į duomenų bazę

2.1 Realizavimo lygis ir jo konceptai

Konceptualiame lygyje projektavimas vyksta (kiek galima) nepriklausomai nuo duomenų bazių valdymo sistemos, kurioje bus kuriama duomenų bazė, modelio. T.y. stengiamasi negalvoti apie lenteles, laukus (reliacinis modelis) ar objektus, jų atributus (objektinis modelis).

Realizavimo lygyje turimas konceptualus modelis atvaizduojamas į duomenų bazės struktūrą, remiantis konkretaus duomenų modelio konceptais.

Duomenų modeliai ir jų pagrindinės sąvokos gali būti:

Duomenų modelis	Konceptai (sąvokos)
Failinė sistema	Failas (file), laukas (field), įrašas (record), duomenys (data)
Hierarchinis	Segmentas (segment, node), tėvas (parent), sūnus (child), hierarchinė seka (hierarchic sequence)
Tinklinis	Aibė (set), įrašas-savininkas (owner record) įrašas-narys (member record), ryšys (relationship)
Reliacinis	Lentelė (table, relation), stulpelis (column), eilutė (row), pirminiai ir išoriniai raktai (primary/foreign keys)
Objektinis	Objektas (object), atributas (attribute), metodas (method), nuoroda (reference), ryšys (relationship)

Taigi, atvaizduojant konceptualią schemą į reliacinę duomenų bazę, operuojama tokiomis sąvokomis kaip lentelė, stulpelis, eilutė, pirminis/išorinis raktas, etc.

2.2 E-R modelio atvaizdavimo į reliacinės DB struktūrą taisyklės

Nors projektavimas (tiek konceptualiame lygyje, tiek ir realizavimo lygyje) yra pakankamai subjektyvus procesas (t.y. priklauso nuo projektuotojo supratimo bei patirties), galima nusakyti bendras taisykles, kaip vienokie ar kitokie konceptualaus modelio elementai gali būti atvaizduojami į realizavimo (DBVS) modelio elementus.

2.2.1 Esybės ir atributai

E-R modelio esybė (esybės tipas) atitinka **lentelę** reliaciniame modelyje. Lentelės **stulpeliai** - tai esybės atributai. Tuomet konkreti esybė atvaizduojama į **įrašą** (eilutę) lentelėje. Eilutės (atitinkančios konkrečią esybę) ir stulpelio (atitinkančio esybės atributą) sankirta bus konkreti vienos esybės kažkokio atributo reikšmė.

Reliaciniame modelyje naudojama **pirminio rakto** sąvoka. Panašiai kaip E-R modelio raktiniai atributai, pirminis raktas skirtas vienareikšmiškam konkrečių įrašų identifikavimui. Tai išnaudojama nusakant ryšius tarp lentelių. Taigi E-R modelio raktiniai atributai tampa lentelės pirminio rakto atributais.

Gali būti, kad konceptualaus projektavimo metu raktiniai atributai nebuvo išskirti. Tuomet, realizavimo lygyje projektuojant lentelės struktūrą, reikia išskriti vieną stulpelį (ar kelių stulpelių kombinaciją), kuris taptų pirminiu raktu. Galima pridėti naują stulpelį, kurio paskirtis būtų užtikrinti eilučių unikalumą.

2.2.2 Ryšiai

Reliaciniame modelyje galimi ryšiai tarp lentelių. Ryšį sudaro **bazinė** (*parent*) ir **priklausoma** (*dependent*) lentelės. Jos gali ir sutapti - tuomet bus rekursyvus ryšys. Iš tiesų, ryšys - tai pirminio ir išorinio raktų pora, t.y. išorinis raktas priklausomoje lentelėje nurodo (*references*) bazinės lentelės pirminį raktą.

E-R modelio ryšiai tarp esybių (o taip pat ryšiai tarp specializuotų ir bendrų esybių, egzistavimo ryšiai bei keletas kitokių konceptų) į reliacinį modelį yra atvaizduojami būtent ryšių tarp lentelių pagalba.

Atvaizduojant E-R modelio ryšius tarp esybių į reliacinio modelio ryšius tarp lentelių reikia atkreipti dėmesį į kelis dalykus:

- E-R ryšio jungiamumą (funkcionalumą)
- E-R esybių dalyvavimo ryšyje būtinumą

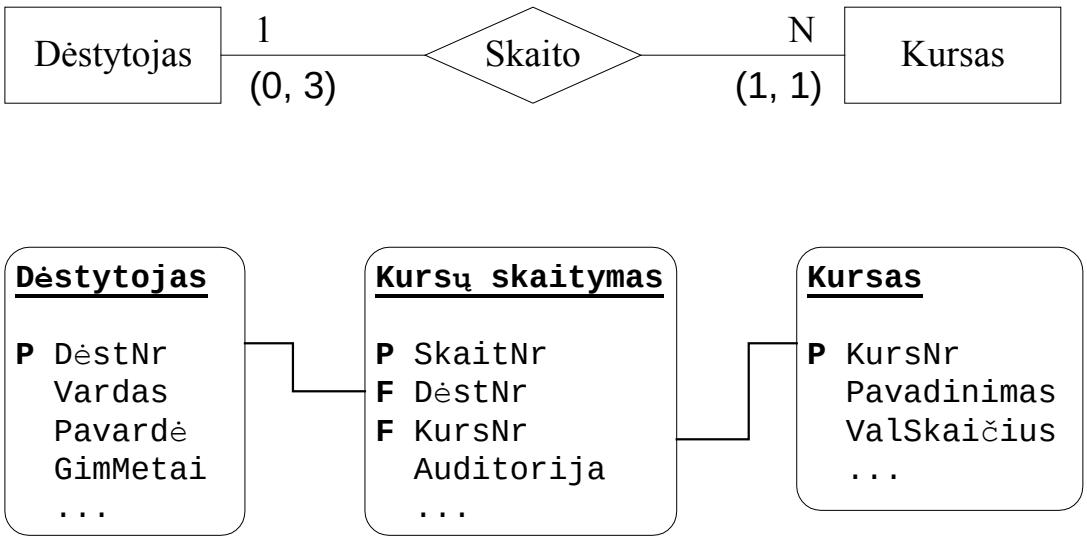
Nuo to priklausys, kurioje lentelėje turės būti išorinis raktas, kokiomis sąlygomis turės būti suvaržyti ryšiai.

"Tiesmukiškas" realizavimas

Pačiu paprasčiausiu būdu, kiekvieną ryšį galima būtų realizuoti atskira lentele, į kurią įtraukiami visų lentelių, atitinkančių ryšyje dalyvaujančias esybes, pirminiai raktai. Tokie pirminiai raktai tampa naujosios lentelės išoriniais raktais. Kadangi kiekviena lentelė turi turėti pirminį raktą[2], naujojoje lentelėje tokiu raktu gali tapti visų išorinių raktų kombinacija arba specialiai tam sukurtas stulpelis.

Tačiau daugeliu atvejų toks realizavimas yra per brangus. Užtenka panagrinėti ryšius 1:1 ar 1:N. Galima pastebėti, kad norint susieti dvi esybes, realiai tenka apjunginėti 3 lenteles. Iš kitos pusės, visiškai akivaizdu, kad vienos lentelės būtų galima išvengti.

[2] Šioje vietoje gali būti ir išimčių - dažna DBVS leidžia kurti lentelę, nenurodant pirminio rakto. Tačiau vėliau tai gali sąlygoti problemų atsiradimą.

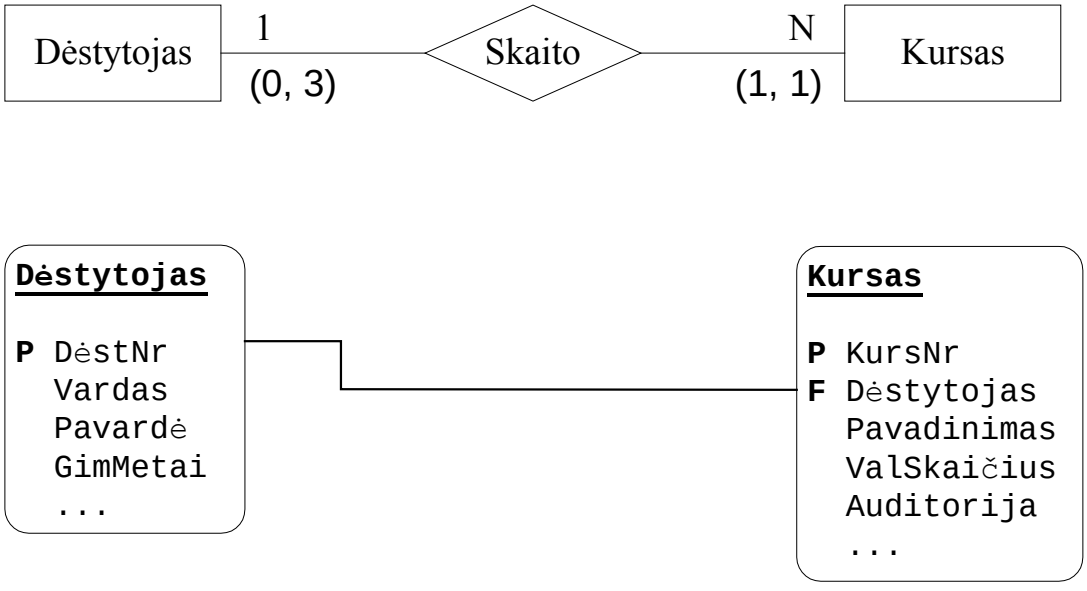


16 pav. "Tiesmukiškas" 1:N ryšio realizavimas

1:N Ryšiai

1:N ryšį galima realizuoti ir taip:

- Į lentelę, atitinkančią esybę "daug", įtraukiamas lentelės, atitinkančios esybę "vienas", pirminis raktas ir padaromas išoriniu raktu.
- Į lentelę "daug" įtraukiami visi ryšio atributai.



17 pav. 1:N ryšio realizavimas

Galima būtų apsvarstyti ir ryšių suvaržymus. Klausimai, į kuriuos reikia atsakyti:

- Ar gali išorinis raktas turėti NULL reikšmių?
- Jei išmetamas bazinės lentelės įrašas, ką daryti su priklausomos lentelės įrašais, kurie nurodo į išmetamąjį įrašą?

Šioje vietoje svarbu, ar į ryšį įeinančių esybių dalyvavimas yra būtinas. Jei esybės "daug" dalyvavimas ryšyje yra nebūtinas (*Kursui* tam tikru momentu gali būti nepriskirtas *Dėstytojas*), tai suvaržymai galėtų būti:

- Išoriniam raktui NULL reikšmės leidžiamos
- Priklausomoje lentelėje šiam ryšiui reikėtų nurodyti ON DELETE SET NULL ir ON UPDATE CASCADE

Jei esybės "daug" dalyvavimas ryšyje yra būtinas (negali būti *Kurso*, kuriam nepriskirtas *Dėstytojas*), tai suvaržymai galėtų būti:

- Išoriniam raktui NULL reikšmės neleidžiamos (t.y. sąlyga NOT NULL)
- Priklausomoje lentelėje šiam ryšiui reikėtų nurodyti ON DELETE RESTRICT (ar net ON DELETE CASCADE) ir ON UPDATE CASCADE

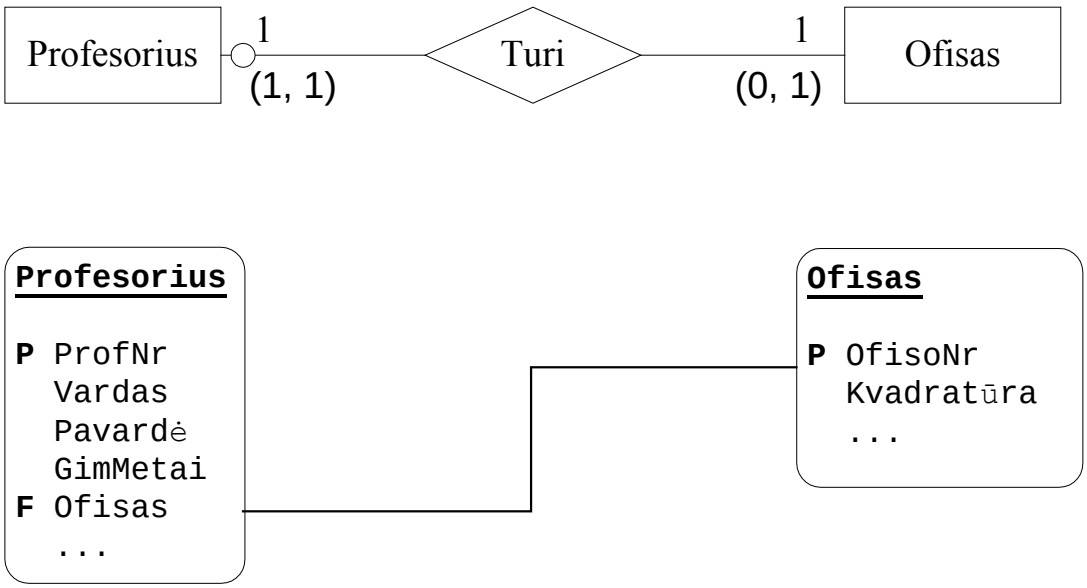
Pastaba: Esybės "vienas" dalyvavimas ryšyje šiuo atveju neatsispindi. Tai, kad kiekvienas dėstytojas turi skaityti bent vieną kursą turėtų užtikrinti pati programa arba tam turi būti rašomi specialūs trigeriai.

1:1 Ryšiai

Jei abiejų ryšyje dalyvaujančių esybių dalyvavimas būtinas ir jos daugiau nesiejamos jokiais ryšiais, tai panašu, kad šios esybės turėtų būti vienos esybės dalimis. Tačiau tai nėra tvirta taisyklė, o greičiau perspėjimas, kadangi daugiau ryšių tarp šių esybių gali nebūti tik dėl to, kad taip buvo apsiribota projektuojant dalykinę sritį (t.y. realiai ryšiai gali egzistuoti, tačiau projektuojant jie buvo neįdomūs).

Beje, kadangi ryšys 1:1 yra simetriškas, galima laisvai pasirinkti, kuri lentelė taps "bazine", o kuri - "priklausoma". Jei vienos esybės dalyvavimas ryšyje būtinas, o kitos - ne, tai rekomenduotina "būtinosios" dalies pirminį raktą padaryti kitoje lentelėje išoriniu raktu ir uždėti sąlygas:

- Išorinis raktas negali tureti NULL reikšmių (NOT NULL)
- Priklausomoje lentelėje ryšiui galioja taisyklės ON DELETE RESTRICT (ar net ON DELETE CASCADE) ir ON UPDATE CASCADE



18 pav. 1:1 ryšio realizavimas

Jei abu ryšio galai yra privalomi (arba nebūtini), tai bazinę ir priklausomą lenteles galima pasirinkti visiškai laisvai.

N:M Ryšiai

N:M ryšiai realizuojami atskira lentele. Ryšyje dalyvaujančias esybes atitinkančių lentelių pirminiai raktai įtraukiami į ryšio lentelę kaip išoriniai raktai; ryšio lentelės pirminiu raktu parenkama išorinių raktų kombinacija (arba sukuriamas specialus stulpelis).

Taigi N:M ryšio atveju realizacija yra “tiesmukiška”.

Ryšių ypatumai

Tarp E-R ryšio ir jo realizavimo reliaciniame modelyje gali atsirasti skirtumų. E-R modelyje visi ryšiai yra dvikrypčiai. Reliaciniame modelyje ryšį sudaranti išorinio ir pirminio raktų pora apibrėžia konkrečią ryšio tarp lentelių kryptį. Tiesa, kryptis šiuo atveju nereiškia informacijos praradimo. Greičiau tik tai, kad “į vieną pusę keliauti” yra paprasčiau (greičiau).

17 pav. pavaizduotoje ryšio realizacijoje galima pastebėti “antisimetriją”. Žinant *Kursą*, visiškai nesunku rasti *Dėstytoją* (nes žinomas pirminis raktas!). Tačiau norint surasti, kokius *Kursus* skaito konkretus *Dėstytojas*, tenka “peržiūrėti” visą Kursų lentelę. Su 1:N ryšiais tai gali atrodyti natūralu. Tačiau akivaizdu, kad su 1:1 ryšiais yra visiškai tokia pati situacija: viena pusė yra privilegijuota kitos atžvilgiu (18 pav.).

Jei ryšys realizuojamas atskira lentele, “antisimetrijos” nebus. Be to, gali būti taip, kad peržiūrėti reikia palyginti nedaug įrašų, jei tik konkrečių ryšių yra nedaug (o konkrečių esybių - į valias).

Taigi, svarstant, į kurią lentelę verta įtraukti išorinį raktą (1:1 atveju) ar renkantis 1:N ryšio realizavimo variantą (atskira lentelė ar ne), verta atkreipti dėmesį į ryšio kryptį.

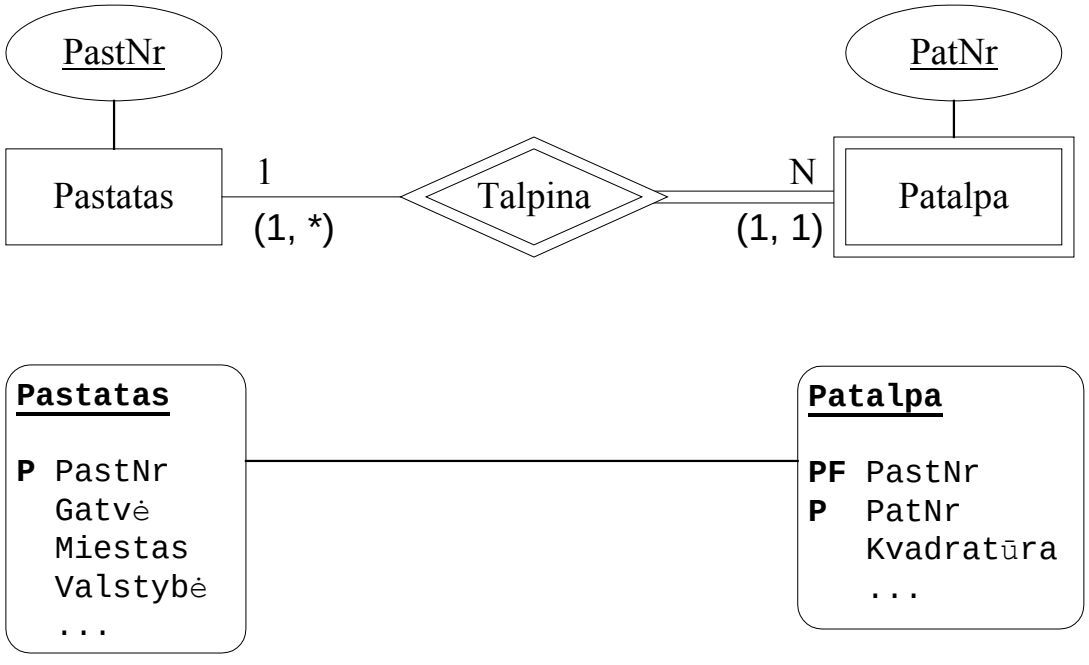
2.2.3 Silpnosios esybės

E-R modelio silpnąją esybę reliaciniame modelyje galima būtų atvaizduoti taip:

- Stipriosios esybės, nuo kurios silpnoji priklauso, lentelės pirminis raktas įtraukiamas į silpnąją esybę atitinkančią lentelę ir padaromas išoriniu raktu.
- Silpnąją esybę atitinkančios lentelės pirminiu raktu paskelbiama stipriosios esybės lentelės pirminio rakto (šie stulpeliai į lentelę įtraukiami pirmu žingsniu) ir silpnosios esybės kandidatinių raktų kombinacija. (Arba sukuriamas atskiras stulpelis).

Ryšys tarp stipriosios ir silpnosios esybių gali būti suvaržytas taip:

- Išorinis raktas negali turėti NULL reikšmių (NOT NULL)
- Ryšiui galioja taisyklės ON DELETE CASCADE ir ON UPDATE CASCADE



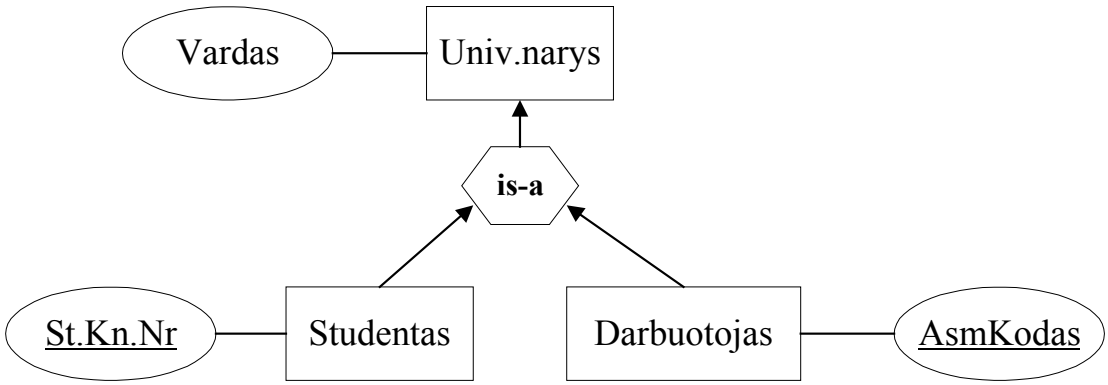
19 pav. Silpnosios esybės realizacija

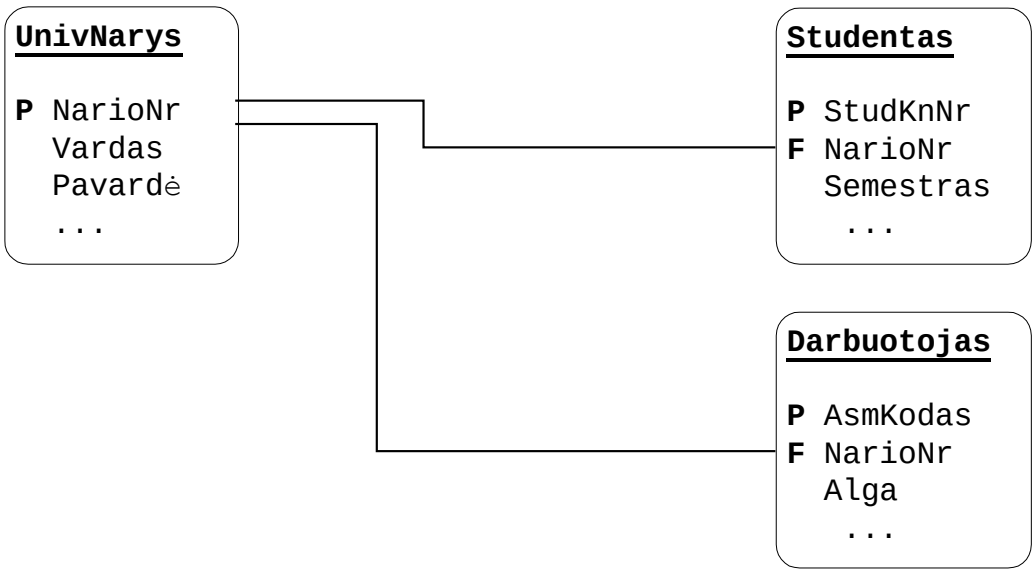
2.2.4 Generalizavimas ir specializavimas

Generalizavimo sąryšį reliaciniame modelyje galima būtų atvaizduoti taip: visos esybės, kurios susiję "is-a" sąryšiais, atvaizduojamos viena lentele. Tokia lentelė turi tiek stulpelių, kiek yra skirtingų atributų. Suprantama, kad bus atributų (priklausančių potipiui), kurie daliai esybių bus neaktualūs. Pvz, 15 pav. pavaizduotoje situacijoje, *Darbuotojai* neturi *Studijų Knygelės Numerio*. Tokiais atvejais, kai esybei atributo reikšmė "neaktuali", ją galima pažymėti NULL.

Tačiau šis sprendimas yra ydingas. Rezultate gausis "išpampusi" lentelė, kurioje NULL reikšmių bus daugiau, nei prasminių. O juk kaip tik dėl tokių situacijų ir buvo įvesta generalizavimo sąvoka!

Todėl yra daroma kitaip. Kiekviena esybė atitinka lentelę. Tačiau į visas lenteles, kurios atitinka kurį nors potipį, įtraukiamas artimiausio supertipo pirminis raktas. Jis tampa potipio lentelės išoriniu raktu.





20 pav. Generalizavimo/specialiazavimo ryšio realizavimas

Disjunktyvaus ar pilno specializavimo atvejai turi būti nagrinėjami atskirai. Šioms sąlygoms užtikrinti reikia pasitelkti specialias sistemos savybes (pvz, trigerius) arba kontrolę realizuoti taikomojoje programoje.

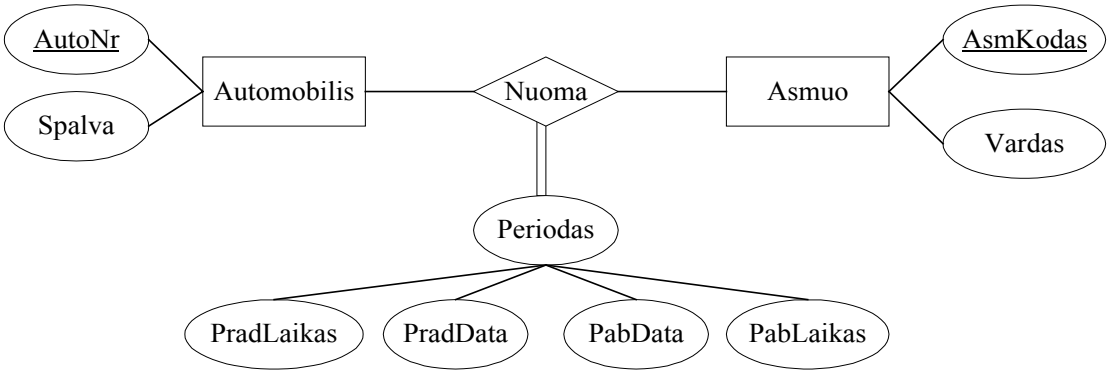
2.2.5 Iteruojančios duomenų grupės

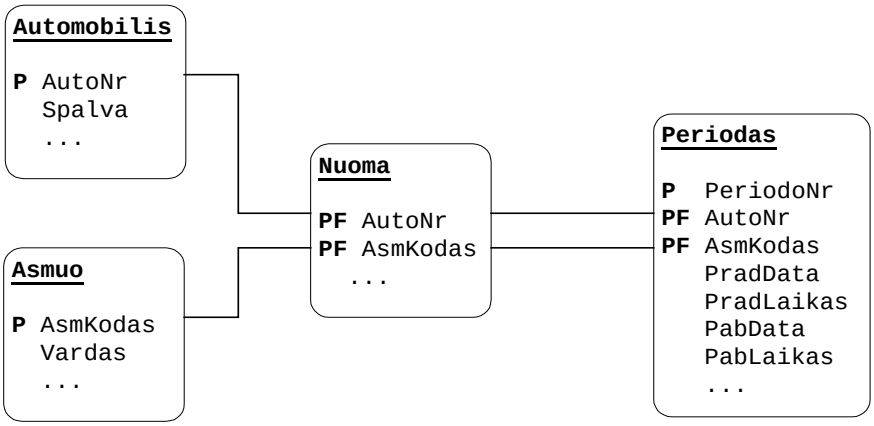
Skyrelyje apie atributus buvo minėti daugelio reikšmių atributai. Tokie atributai literatūroje kartais vadinami **iteruojančiais duomenų elementais**. Sudėtiniai atributai savo ruožtu vadinami **duomenų grupėmis**. Tuomet sudėtiniai daugelio reikšmių atributai vadinami **iteruojančiomis duomenų grupėmis**.

Iteruojančios duomenų grupės dažnai naudojamos istorinių (t.y. periodiškai pasikartojančių) duomenų modeliavimui.

Galima įsivaizduoti ryšį *Nuoma* tarp esybių *Automobilis* ir *Asmuo*. *Periodas* - ryšio *Nuoma* atributas. Kadangi tas pats asmuo gali kelis kartus išsinuomuoti tą patį automobilį, tik skirtingais periodais, *Periodas* - daugelio reikšmių atributas. Be to, *Periodas* - sudėtinis atributas. Taigi *Periodas* - iteruojanti duomenų grupė.

Kalbant apie daugelio reikšmių atributus, buvo pateikta, kaip juos atvaizduoti į lentelinę struktūrą (reliaciniame modelyje). Antrasis variantas (naujos esybės, susidedančios iš originalaus daugelio reikšmių atributo komponentų kūrimas) yra dažniausiai naudojamas iteruojančių duomenų grupių realizavime.





21 pav. Iteruojančios grupės realizacija

Tokioje realizacijoje iteruojanti grupė (daugelio reikšmių sudėtinis atributas) tampa atskira esybe. Kadangi jos egzistavimas priklauso nuo esybės (ar ryšio), kuriam priklauso iteruojanti grupė, esybė yra silpnoji. Tuomet jos realizacija niekuo nesiskiria nuo paprastų silpnųjų esybių realizacijos. Įdomu tik tai, kad bendru atveju, iteruojanti grupė gali priklausyti tiek esybei, tiek ryšiui (t.y. tiek esybė, tiek ryšys gali turėti sudėtinių daugelio reikšmių atributų).

2.2.6 Efektyvumas ir kiti aspektai

Aptarinėjant E-R modelio ryšių realizavimą, buvo paminėtas "tiesmukiškas" diagramų atvaizdavimo būdas. Galima būtų sakyti, kad E-R diagramos realizavimas, remiantis poskyriuose 2.2.1 - 2.2.5 pateiktais nurodymais irgi yra tam tikra prasme tiesmukiškas. Tokį diagramos atvaizdavimą į realicinės duomenų bazės struktūrą galima atlikti automatiškai. Iš tiesų, tai net yra programinės priemonės, kurios tai daro, t.y. automatiškai generuoja duomenų bazių struktūrą (DDL sakinius) iš turimos E-R (ar kito koncepcinio modelio) diagramos. Tokios programinės priemonės, padedančios projektuoti sistemą, vadinamos CASE[3] įrankiais (CASE Tools). Be kita ko, jos leidžia gražiai ir lengvai piešti diagramas, gali palaikyti projekto neprieštaringumą (kad atskiros dalys viena kitai neprieštarautų), versijų kontrolę, skirtingus modelio vaizdus ir t.t.

Tačiau automatinis duomenų bazės struktūros generavimas (arba paprasčiau - "tiesmukiškas" E-R diagramos atvaizdavimas į DB struktūrą) turi minusų. Didžiausias jų - efektyvumas. Gali būti situacijų, kai informacijos išgavimui tenka "pasinaudoti" daugeliu ryšių tarp esybių, keliaujant nuo vienos esybės prie kitos. Prisiminus, kad kiekviena esybė, o galbūt net ir ryšys, tampa lentele, akivaizdu, jog SQL rašyta užklausa gali apimti daug lentelių. O kuo daugiau lentelių apjungiama - tuo ilgesnis užklausos vykdymo laikas.

Dėl šios priežasties po to, kai konceptualiame lygyje sumodeliuojama dalykinė sritis, modelis gali būti papildomas - pridedama naujų ryšių, išskaičiuojamų atributų ar net naujų esybių. Suprantama, toks papildymas informacijos prasme nieko naujo neatneša. Dar blogiau - įvedami duomenų pasikartojimai, kas gali apsunkinti duomenų keitimą, padidinti reikalingos atminties kiekį ir tapti papildomu klaidų šaltiniu. Tačiau laimėtas efektyvumo kiekis gali būti vertas tokios aukos.

Beje, sprendžiant efektyvumo klausimą, vien saugojamos informacijos poreikių žinojimo (t.y. žinių apie tai, kokią informaciją reikią išsaugoti) nebepakanka. Reikia žinoti ir informacijos apdorojimo poreikius (t.y. žinių apie tai, kas bus daroma su informacija).

Efektyvumą padidinti galima ir **fiziniame lygyje**. Tuomet reikia žinoti naudojamos DBVS charakteristikas (o, galbūt, ir naudojamos operacinės sistemos ar technikos galimybes). Paprasčiausios fizinio lygio priemonės efektyvumui padidinti yra indeksai, klasterizavimas (clustering), užklausų optimizavimai. Tačiau šie klausimai jau išeina iš E-R modeliavimo temos ribų.

[3] CASE - Computer Aided Software Engineering

Literatūra

[Kemp 96]
Alfons Kemper, André Eickler. *"Datenbanksysteme. Eine Einführung"*. R.Oldenbourg Verlag, München, Vien, 1996

[Rob 93]
Peter Rob, Carlos Coronel. *"Database Systems. Design, Implementation and Management".* Wadsworth Publishing Company, Belmont, California, 1993

[Huber 92]
Peter Huber, Carsten Christiansen, Karen Kitgaard *"Informationsanalyse"* 2.udgave. Teknisk Forlag, København, 1992.