

|  |  |  |
| --- | --- | --- |
| VILNIAUS KOLEGIJA  ELEKTRONIKOS IR INFORMATIKOS FAKULTETAS  PROGRAMINĖS ĮRANGOS KATEDRA | | |
| **UML ir duomenų bazės** | | |
|  | | |
| REFERATAS  SD 6531BX028 PI17A | | |
| STUDENTAS |  | Gediminas Šaulys |
|  |  |  |
|  |  |  |
| DĖSTYTOJAS |  | Dainius Savulionis |
|  |  |  |
|  |  |  |
| 2018 | | |

**TURINYS**

[ĮVADAS 3](#_Toc532968696)

[1. DUOMENŲ MODELIAVIMO METODOLOGIJOS IR JŲ SAMPRATA 4](#_Toc532968697)

[2. UML 5](#_Toc532968698)

[3. VEIKSMO DIAGRAMA (UML) 6](#_Toc532968699)

[4. DUOMENŲ MODELIAI 8](#_Toc532968703)

[5. PAGRINDINĖS SAVOKOS 9](#_Toc532968704)

[6. SANTYKIAI 10](#_Toc532968705)

[7. DUOMENŲ VIENTISUMO SĄLYGOS 11](#_Toc532968706)

[8. DUOMENŲ ANOMALIJOS 11](#_Toc532968707)

[9. PIRMOJI NORMINĖ FORMA 12](#_Toc532968708)

[10. FUNKCINĖS PRIKLAUSOMYBĖS 12](#_Toc532968709)

[11. ANTROJI NORMINĖ FORMA 12](#_Toc532968710)

[12. TREČIA NORMALINĖ FORMA (TNF) 12](#_Toc532968711)

[13. KETVIRTOJI NORMINĖ FORMA 13](#_Toc532968712)

[IŠVADOS 14](#_Toc532968713)

[LITERATŪROS SĄRAŠAS 15](#_Toc532968714)

# ĮVADAS

Duomenų bazė – tai yra rinkinys tarpusavyje susijusių duomenų, kuriems apdoroti naudojamos kompiuterinės programos. Šių duomenų susisteminimas ir apdorojimas yra laikomas duomenų bazės projektavimu. Toks projektavimas yra veikla, kurios pagrindinis tikslas yra palengvinti bei pagreitinti susistemintų duomenų naudojimą praktikoje.

Loginį duomenų projektavimą dažniausiai sudaryto dvi dalys:

1) veiksmo diagramos sudarymas;

2) duomenų modelio kūrimas.

Veiksmo diagramoms kūrimui skiriama tik UML metodologija, tačiau duomenų modeliavimui yra sukurta pakankamai daug skirtingų metodologijų. Išnagrinėti visas duomenų modeliavimo metodologijas yra praktiškai neįmanoma, nes jos visos turi tą patį tikslą ir objektus, atvaizduotus nedaug besiskiriančiais būdais. Visose metodologijose naudojamos net gi tos pačios priemonės: kuriamos esybės, išskiriami jų atributai bei domenai ir nustatomi santykiai tarp esybių. Esminis šių santykių skirtumas, leidžiantis kiekvieną metodologiją laikyti atskira metodologija, yra santykių pavaizdavimas.

# DUOMENŲ MODELIAVIMO METODOLOGIJOS IR JŲ SAMPRATA

Terminas „duomenų modeliavimas“ plačiąja prasme yra dažnai vartojamas viso duomenų bazių (toliau - DB) projektavimo proceso aprašymui.

. Reikalingas ne toks sudėtingas būdas pateikti informaciją, kuri buvo pateikta dokumente

lentelės pavidalu. Tam tikslui pavaizduojami duomenys grafiniais modeliais (paveiksliukais), kad

dokumentas būtų paverstas paveiksliuku.

Modelių ar diagramų tipų egzistuoja daug. Pvz.: proceso modeliai, duomenų srauto diagramos,

duomenų modeliai, sekų diagramos – sąrašas galėtų būti pakankamai ilgas, tačiau pagrindinis dėmesys yra į du specifinius modelius:

1) veiksmo diagrama (Use Case) – modelis, kuris įeina į bendrą aprašymą, dar yra žinomas kaip

Unified Modeling Language – unifikuota modeliavimo kalba (toliau – UML), naudojamas vartotojų poreikių bei reikalavimų modeliavimui. Šis modelis yra naudojamas informacijos apie projektą įvedimui taip, kad ją naudoti būtų patogu ne tik vartotojams, bet ir projektuotojams;

2) duomenų modeliai (Data Models) – modeliai, kurie yra susiję su duomenų struktūrų pavaizdavimu duomenų bazėje. Prieš apžvelgiant šiuos modelius, vertėtų trumpai aptarti naudojamas modeliavimo metodologijas.

Veiksmų modelis faktiškai yra dalis bendro aprašymo, kuris vadinamas UML. UML

aprašymas neturi reliacinių DB duomenų modeliavimo priemonių, taigi reikia naudoti kitą duomenų

modeliavimo metodologiją.

Duomenų modeliavimo metodologija yra lengviausia skaitymui. Be to ji taip pat leidžia pavaizduoti ir išdėstyti visą reikiamą informaciją. Vienas populiariausių modeliavimo formatų reliacinėms DB – Integration Definition For Information Modeling – skirtas integralinio nustatymo informaciniam modeliavimui, vėliau trumpai nagrinėjamas kitas formatas – Information Engineering – informacinis nagrinėjimas. Duomenų modeliavimo metodologiją galima rinktis vadovaujantis asmeniniais prioritetais, todėl pasirinkimą daug lemia ekonomika, standartai ar kompanijos galimybės parenkant priemones. Analizę galima pradėti nuo įvairių priemonių ypatumų, kurie reikalingi arba pageidaujami, kai reikia praplėsti grafinio modeliavimo pagrindus. Visa informacija, esanti dokumente, turi būti pateikta ir išdėstyta duomenų modelyje, todėl reikia turėti galimybę gauti tuos duomenis tam, kad būtų galima juos pateikti klientams bei programuotojams.

# UML

UML – de facto standartinė nustatymo ir dokumentavimo metodologija yra komponentas objektyviai orientuotų programinių sistemų, kurios yra sukurtos trijų žmonių darbo dėka: Gredi Bučo, Ivaro Jakobsono ir Džimo Rambo. Kiekvienas iš šių žmonių turėjo savo (arba dalinai savo) metodologiją, bet jos buvo sujungtos, nes visi šie žmonės dirbo vienoje kompanijoje Rational Software su produktu Rational Rose.

Ši metodologija nėra uždara. Šiuo metu ji plačiai naudojama ir yra standartizuota programinėse priemonėse, kurias siūlo daugelis korporacijų. Taip pat svarbu paminėti, kad UML yra ne tik modeliavimo metodologija, bet ir galutinis orientuoto į objektą projektavimo metodas.

Tuo metu, kai modeliavimo metodologija yra tik metodas, grafinio atvaizdavimo komponentų, kompiuterinės sistemos, projektavimo metodas suteikia galimybę projektuoti sistemą nuo pradžios iki pabaigos. UML kelia nedidelius reikalavimus modeliavimo procesams. Kitas tikslas - paaiškinti, kodėl duomenų modeliavime reikia naudoti UML, nors jis tiesiogiai nėra naudojamas reliacinių duomenų kūrimui ir modeliavimui, nes jis nėra pritaikytas reliacinių technologijų DB.

UML yra sudarytas iš kelių skirtingų modelių. Kiekvienas jų yra naudojamas sistemos komponentams pristatyti, tačiau čia naudojamas tik vienas iš jų. Duomenų architektui, užsiimančiam tik reliacinių bazių kūrimu, daugelis šių modelių nėra aktualūs, išskyrus veiksmo diagramų modelį (Use Care Diagram).

# VEIKSMO DIAGRAMA (UML)

Kaip aprašyta UML instrukcijoje, 1.1 versija pristato funkcionalias sistemos arba klasės galimybes sistemos viduje taip, kaip tai mato vartotojas. Veiksmo diagrama aprašo:

1) sistemos komponentus, sumodeliuotus kaip veiksmas (Use Cases);

2) sistemos vartotojus, sumodeliuotus kaip veikėjas (Actors);

3) santykius tarp veiksmų ir veikėjų.

Supaprastintos veiksmo diagramos yra abstraktus dialogas tarp veikėjo ir sistemos. Nekreipiant dėmesio į smulkmenas, jos aprašo potencialias sąveikas. Šis informacijos tipas yra svarbus DB architektui, nes reikia identifikuoti vartotojus, kurie naudosis DB, ir tai, ką jie nori padaryti. Tam yra keletas svarbių priežasčių:

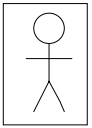
1) saugumas;

2) duomenų, reikalingų esamo proceso palaikymui, užtikrinimas;

3) kūrimas saugomų procedūrų ir trigerių, reikalingų proceso palaikymui.

Galima teigti, kad duomenų diagramos apskritai neturi nieko bendra su procesais ar vartotojais, nes veiksmo diagramos yra sudedamoji UML dalis – jos dažnai naudojamos ne tik griežtai objektyviai orientuotų projektuotojų, bet ir visų tipų sistemų architektų, kad būtų gautas veiksmų, kurie duos norimą rezultatą, brėžinys. Skirtingiems projektuotojam bei architektams naudojant skirtingas modeliavimo technologijas reikia naudoti ir skirtingas veiksmo diagramų dalis.

Veiksmo diagramos yra nesudėtingos ir turi du pagrindinius simbolius, kurie naudojami diagramose. Pirmas simbolis – veikėjas (Actor), kurį vaizduoja žmogeliuko figūrėlė (2.1 pav.). Veikėjas vaizduoja vartotoją, sistemą arba dalį priemonių, kurios atlieka veiksmus sistemoje. Antras simbolis – veiksmas (Use Case), vaizduojamas elipse su pavadinimu viduje.(2.2 pav)



2.1 pav. Veikėjo simbolis

# 

# 

2.2 pav. Veiksmo simbolis

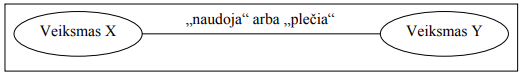
# 

2.3 pav. Santykių komunikacija

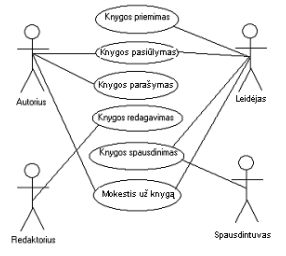
Norint sumodeliuoti faktą, kad veikėjas naudoja veiksmą, šie du simboliai yra sujungiami linija, kuri vadinama santykių komunikacija (Communicates Relationship).(1.3 pav)

Veiksmo elementai gali būti sujungti vienas su kitu dar dviem būdais:

1) tikslo pasiekimui kito veiksmo pagalba – santykis „naudoja“ (uses);

2) plečiant kitą veiksmą tam, kad būtų patikslintas jo veikimas – santykis „plečia“ (extends). Jie modeliuojami paprastu būdu: 

2.4 pav. Veiksmo plėtimas

Tarkime, kad reikia sumodeliuoti knygos kūrimo procesą. Pirmiausia, tam reikalingi veikėjai: leidėjas, spausdintuvas, redaktorius ir autorius. Be to dar reikalingi ir veiksmai: knygos pasiūlymas, knygos gavimas, knygos parašymas, knygos redagavimas, knygos spausdinimas ir atlygis už knygą. Aišku, šis pavyzdys neparodo visų veiksmų ir veikėjų, reikalingų knygos publikavimui, tačiau šiam pavyzdžiui to pakanka. Gaunama tokia diagrama:

2.5 pav. Knygos kūrimo proceso veiksmo diagrama

Šioje diagramoje vienas momentas gali pasirodyti neįprastas, nes knygos priėmimas numatytas anksčiau negu pats knygos pasiūlymas. Tačiau veiksmų diagramoje dar nėra tvarkos, nes dėsningumas atsiranda tik vėlesniame proceso nagrinėjime. Faktiškai labai didelėje diagramoje gali būti net šimtai veiksmų, jungiamų daugybės veikėjų. Kiekvienas elementas turi informaciją, paaiškinančią jo egzistavimą. Ši lentelė rodo vieno iš naudojamų veiksmų galimą aprašymą:

|  |  |
| --- | --- |
| Pavadinimas | Knygos parašymas |
| Aprašymas | Idėjos kūrimo procesas bei jos vertimas keliais šimtais puslapių |
| Prielaidos | Knyga turi būti pateikta ir patvirtinta  Schema turi būti pilnai išdėstyta ir paruošta knygos parašymui |
| Pagrindinės sąlygos | Medžiaga bus pabaigta ir paruošta leidimui |
| Darbo etapai | Plano sudarymas  Įvado parašymas  Visų skyrių parašymas  Reziumė parašymas |

2.1 lentelė Veiksmo aprašymas

# DUOMENŲ MODELIAI

Pagrindiniai duomenų modelių tipai – loginis ir fizinis. Proceso pradžioje sukuriamas loginis modelis, o paskui jo pagrindu sukuriamas vienas ar keletas fizinių modelių. Jei projektavimas vyksta sėkmingai, pilnas loginis modelis turi būti ekvivalentus fiziniam modeliui, kai tik bus visiškai realizuotas loginis modelis. Kol duomenų nebus loginiame modelyje, jokie duomenys nepasirodys ir fiziniame.

Loginis modelis sudaro pagrindinę informacijos dalį, kurią vartotojas turi palaikyti. Loginis modelis, priklausomai nuo fizinės realizacijos reikalavimų, gali būti realizuojamas įvairiais būdais. Nepriklausomai nuo to, ar naudojamas SQL Server 2000, Oracle, MS Access ar net MS Excel duomenų patalpinimui, loginis modelis turi būti nekintantis.

Loginio modeliavimo procese pirmiausia privalu įsitikinti, kad kiekviena informacijos dalis yra užregistruota, tuomet ją galima kur nors patalpinti. Būtent todėl, toliau vykdant normalizacijos procesą, ankstesnis modelis taps organizuotu rinkiniu. Iki šio proceso pabaigos reikia identifikuoti kiekvieną atskirą informacijos dalį, kurią norima fiziškai kur nors patalpinti.

Pagrindinis skirtumas tarp loginio ir fizinio modelio yra realizacija. Dėl to, kad naudojama ta pati programavimo kalba, loginis modelis tampa panašus į fizinį modelį, tačiau loginio modeliavimo stadijoje reikia atkreipti dėmesį į tai, kas įeis į DB, o ne į tai, kaip tai įeis į DB.

Fizinis modelis užtikrina detalų veiksmų planą DB realizacijai. Tuo metu, kai loginis modelis turi būti pakankamai pilnas, kad galima būtų aprašyti komercinę veiklą, fizinis modelis, kuris sukuriamas vėliau, gali būti priklausomas nuo norimo efektyvumo ir charakteristikų. Dėl šių modelių skirtumo kyla klausimas – koks yra loginio modeliavimo tikslas?

Fiksuojant tai, koks turėtų būti optimaliausias duomenų išdėstymas, remiantis loginiu mąstymu, sudaromas pagrindas geresnės fizinės DB sukūrimui, net ir tuo atveju, jei fizinis modelis truputį skirsis nuo loginio modelio. Loginis modelis lieka labai svarbus dokumentas vėlesniems pakeitimams sistemoje ir nebus smarkiai pakeistas duomenų išdėstymo detalėmis.

# PAGRINDINĖS SAVOKOS

**Esybė** (anglų k. entity) – tai grindžiama vieno tipo objektų klasė, fiziškai egzistuojantys ir mintyse suvokiami bei lengvai skiriami modeliuojamo pasaulio bendrieji vaizdai. Kiekvienai esybei modeliuojamoje sistemoje yra suteikiamas unikalus tapatumo vardas.

Esybių tipai :

1. **Esybė vadinama nepriklausoma**, kai ji neturi jokių priklausomybių, pirminio rakto nuo kitu esybių, kitaip tariant, nėra jokių išorinių raktų arba migruojančių raktų pirminiame rakte. Jis faktiškai nejuda, jis priklauso pirminiam raktui vienos esybės, migruoja (kopijuojasi) kaip atributas kitos esybės, tokiu būdu nustatant santykį tarp šių esybių. Darytina išvada, kad esybė nepriklausoma nuo kitos esybės. Visi atributai, kurie nemigruoja, yra nuosavi, nes jie turi savo kilmę esamoje esybėje.
2. **Priklausoma** **esybė** priešinga nepriklausomai esybei, nes ji turi pirminį raktą kitos esybės, kuris migravo į jos pirminį raktą.

Visi objektai turi tam tikrus požymius, vadinamus **atributais.** Visi vienos esybės objektai turi tuos pačius atributus. Kiekvienam atributui priskiriama galimų reikšmių aibė. Paprastai objekto atributai yra **vienareikšmiai**, pavyzdžiui, kiekvienas skaitytojas turi vieną asmens kodą. Tačiau kai kurie atributai gali būti **nevienareikšmiai**, pavyzdžiui, knyga gali būti parašyta keleto autorių. Taigi esybės Knyga atributas Autorius yra **daugiareikšmis**. Atributas gali būti paprastas arba sudėtinis. **Sudėtinį** atributą sudaro keletas paprastų atributų. Pavyzdžiui, esybės Skaitytojas objektai gali turėti sudėtinį atributą Adresas, kurį, tarkime, kai visi skaitytojai gyvena tame pačiame mieste, sudaro trys paprasti atributai: gatvė, namas ir butas. Atributai, vienareikšmiškai atitinkantys esybės objektą, vadinami **raktu**. Pavyzdžiui.,esybės Knyga objektų atributas ISBN yra raktas, kadangi kiekviena knyga turi unikalų ISBN numerį. Kai kurie atributai gali būti išvestiniai, jie apskaičiuojami pagal kitų atributų reikšmes, pavyzdžiui, skaitytojo amžių galima apskaičiuoti žinant jo gimimo ir dabartinę datą. **Esybės atributai** turi turėti unikalius vardus jos viduje.

**Raktai :**

**Pirminis raktas** yra stulpelis (ar kelių stulpelių derinys), kurį žinodami galime vienareikšmiškai pasakyti, iš kurios eilutės jis yra paimtas.

**Antriniai raktai** – tai rinkinys vieno ar daugiau laukų, kurių individualumą siekiama užtikrinti visoje esybėje. Antriniai raktai neturi specialių žymėjimų, ir jie negali neemigruoti.

**Išoriniai raktai** kitaip vadinami migruojančiais atributais. Jie atlieka pirminio esybės rakto

vaidmenį, kai reikia nurodyti informacijos egzempliorių iš kitos esybės. Išoriniai raktai žymimi kaip antriniai raktai tik su simboliu „FK“ (Foreign Key) po išorinio rakto.

**Domenai -** tai yra galimų atributo reikšmių rinkinys, mechanizmas, kuris ne tik leidžia nustatyti galimas reikšmes, kurios gali būti atribute, bet ir užtikrina duomenų tipų paveldėjimą apibrėžimuose.

**Atributų žymėjimai -** Esybės pavadinimas niekada neturi būti vartojamas daugiskaitos forma. Ši taisyklė formaliai tinka ir atributų pavadinimams. Tačiau modeliavimo procese vis dėlto daugiskaita bus naudojama. Daugiskaitos naudojimas pavadinime gali būti bruožas, parodantis, kad laukiama keleto reikšmių. Pavyzdžiui, būtų galima turėti esybę Person (žmogus) su identifikuojančiu atributu Children (vaikai). Esybė Person identifikuotų vieną žmogų, o atributas Children identifikuotų jo sūnus ir dukteris.

# SANTYKIAI

**Santykis** – tai grafiškai vaizduojamas junginys, kuris nusako esybių tarpusavio ryšį. Santykiai gali būti klasifikuojami pagal:

1) dalyvavimo laipsnį;

2) galią;

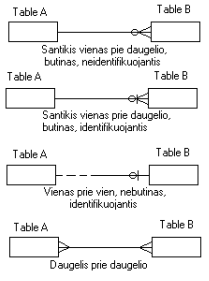
3) kardinalumo požymį;

4) kryptį;

5) tipą;

6) būtinumą.

IE (Information Engineering) metodologija yra gerai žinoma bei plačiai naudojama. Be to dar ji yra populiari ir labai patogi atvaizduojant reikalingą informaciją. Ši metodologija metaforiškai yra vadinama „paukščio letenėlės“ metodu.



3.1 pav. Santykių pavaizdavimo pavyzdžiai

|  |  |
| --- | --- |
| **Simbolis** | **Aprašymas** |
|  | Daugelis – esybė iš „paukščio letenėlės“ pusės reiškia, kad gali būti daugiau negu viena reikšmė, kuri priklauso egzemplioriui, turinčiam kitokias savybes. |
|  | Nebūtina esybė – parodo, kad nebūtinai egzistuos egzempliorius iš šios santykių pusės. Tai gali būti aprašoma „nei vieno arba daugiau“, priešingybė jai būtų „vienas ar daugiau“. |
|  | Identifikuojantis santykis – esybės raktas iš kitos santykio pusės migruoja į šia esybę. |
|  | Nebūtinas santykis – punktyrinė linija iš vienos santykio pusės rodo, kad migruojantis raktas gali būti NULL. |

3.2 lentelė Simbolių, naudojamų santykių pavaizdavime, aprašymas

# DUOMENŲ VIENTISUMO SĄLYGOS

Reliacinėje teorijoje yra suformuluoti keletas reikalavimų, kuriuos turi atitikti DB duomenys. Viena reikalavimų rūšis vadinama duomenų vientisumo sąlygomis. Čia aptariamos trys iš jų:

1) kategorijų vientisumas;

2) nuorodų vientisumas;

3) funkcinės priklausomybės.

Reliacinės lentelės eilutės yra konkrečių realaus pasaulio objektų atitikmenys duomenų bazėje. Pavyzdžiui, kiekviena lentelės Darbuotojai eilutė atitinka konkretų įmonės darbuotoją.

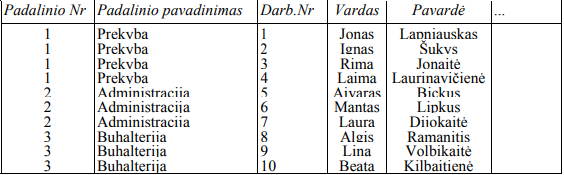
Realaus pasaulio objektai, atitinkantys lentelės eilutes, reliacinėje teorijoje vadinami kategorijomis. Lentelės raktas vienareikšmiškai nusako lentelės eilutę ir kategoriją. Kad rastų duomenis apie konkrečią kategoriją (objektą), vartotojas privalo žinoti rakto reikšmes. Tai reiškia, kad kategorija neturi prasmės duomenų bazėje, jeigu bent vieno rakto atributo reikšmė yra nežinoma. Todėl, pagal kategorijų vientisumo reikalavimą, joks lentelės rakto atributas nė vienoje eilutėje negali turėti NULL reikšmės.

Reliacinėje duomenų bazėje vienos lentelės eilutėms susieti su kitos lentelės eilutėmis vartojami išoriniai raktai. Duomenų bazė, kurioje visi netušti išoriniai raktai nurodo į egzistuojančią pirminio rakto reikšmę, tenkina nuorodų vientisumo reikalavimą. Matyti, kad lentelėje išorinis raktas gali būti tuščias (jo atributų reikšmės yra NULL), jei jo atributai neįeina į pirminį raktą. Nuorodų vientisumo reikalavimas gali būti suformuluotas taip: kiekvieno išorinio rakto reikšmė duomenų bazėje turi būti tuščia arba turi sutapti su viena pirminio rakto reikšme lentelėje, į kurią kreipia išorinis raktas.

Trečiasis reikalavimas duomenims – funkcinės priklausomybės – detaliau aptariamos kituose skyreliuose.

# DUOMENŲ ANOMALIJOS

Tarkime, kad norima aprašyti įmonės padalinius ir jiems priklausančius darbuotojus. Vietoje dviejų lentelių Darbuotojai ir Padaliniai, kurios yra naudojamos praktinėje užduotyje, bandoma sukurti vieną lentelę Padal\_Darbuotojai. Užpildžius lentelę Padal\_Darbuotojai duomenimis, kurie buvo dviejose lentelėse Darbuotojai ir Padaliniai, gaunama nauja lentelė:



4.1 lentelė Duomenų lentelės pavyzdys

Nesunku pastebėti, kad lentelė Padal\_Darbuotojai yra sudaryta nesėkmingai. Pavyzdžiui, keliose eilutėse kartojasi tas pats padalinio pavadinimas bei jo numeris. Tokie pertekliniai duomenys ne tik užima vietos kompiuterio atmintyje – dėl jų gali būti prarastas duomenų vientisumas arba, kitaip tariant, atsirasti duomenų prieštaringumas. Mat tada vieną projektą gali vykdyti keli vykdytojai. Tarkime, pasikeitė padalinio pavadinimas ir naujas pavadinimas buvo pakeistas tik vienoje eilutėje. Tuomet tarp eilučių, kuriose yra informacija apie tą patį padalinį, atsiranda neatitikimas, kuris vadinamas atnaujinimo anomalija – tai duomenų prieštaringumas, atsirandęs dėl duomenų pertekliaus, atnaujinus tik dalį jų.

# PIRMOJI NORMINĖ FORMA

Normine forma (toliau – NF) vadinamos sąlygos, kurias turi tenkinti duomenų bazės reliacinė schema, kad DB išvengtų tam tikrų nepageidaujamų savybių. Lentelė yra pirmos norminės formos (1NF), jei visų jos atributų reikšmės yra atomai. Atomu laikoma reikšmė, kuri nėra nei aibė, nei sąrašas. Paprastai apibrėžiant reliacinę lentelę reikalaujama, kad visos visų atributų reikšmės būtų atomai. Todėl visos nagrinėjamos lentelės laikomos atitinkančiomis 1NF.

# FUNKCINĖS PRIKLAUSOMYBĖS

Aptariant kategorijų ir nuorodų vientisumo sąlygas, būtina akcentuoti, kad dar griežtesnius reikalavimus reliacinei schemai apibrėžia funkcinės priklausomybės. Tai bene svarbiausia sąlyga. Sakoma, kad atributas B funkciškai priklauso nuo A, jei kiekvienai A reikšmei egzistuoja lygiai viena jį atitinkanti reikšmė B. Funkcinė priklausomybė žymima A —> B.

Jos esmė tokia, kad vienų atributų reikšmės eilutėje gali vienareikšmiškai apibrėžti kitų atributų reikšmes. Pavyzdžiui, kiekviena lentelės Padal\_darbas eilutė Padalinys vienareikšmiškai apibrėžia atributų Padal\_nr ir Padal\_pavadinimas reikšmes.

Užrašą A—>B galima skaityti dvejopai: B funkciškai priklauso nuo A, taip pat A funkciškai apibrėžia B. Funkcinės priklausomybės kairiosios dalies atributai vadinami determinantu – A yra B determinantas. Determinantai – tai atributai, kurių reikšmės apibrėžia kitų atributų reikšmes.

Funkcinės priklausomybės sąvoka yra sąvokos „lentelės viršraktis“ apibendrinimas. Lentelės viršraktis – tai toks jos atributų aibės R poaibis S (S ⊆ R), kad S —> R. Faktas, kad lentelės atributų aibė yra R, žymimas L(R); tai atitinka L schemos užrašą be lentelės pirminio rakto.

# ANTROJI NORMINĖ FORMA

Reikia užtikrinti, kad kiekvienas neraktinis atributas būtų funkcionaliai priklausomas nuo visų raktų, nes jeigu taip nėra, reikia atskirti į skirtingus sąryšius tuos atributus, kurie priklauso tik nuo rakto dalies.

# TREČIA NORMALINĖ FORMA (TNF)

Reikia užtikrinti, kad visi neraktiniai atributai būtų vienas nuo kito nepriklausomi, nes jeigu taip nėra, turi būti sukurtas naują sąryšis, kuris neturėtų jokios neraktinių atributų tarpusavio priklausomybės.

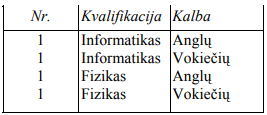
# KETVIRTOJI NORMINĖ FORMA

Ketvirtajai norminei formai apibrėžti nepakanka funkcinės priklausomybės sąvokos. Tam įvedama daugiareikšmės priklausomybės sąvoka, apibendrinanti funkcinę priklausomybę.

Lentelėje L(A, B, C) yra daugiareikšmė (anglų k. MV – multivalued) priklausomybė A —>—> B tada ir tik tada, kai atributų B reikšmių aibė, atitinkanti bet kurias atributų aibių A ir C reikšmes, priklauso tik nuo A ir nepriklauso nuo atributų C reikšmių.

Esant funkcinei priklausomybei, kiekvieną determinanto reikšmę atitinka tik viena nuo jo priklausomo atributo reikšmė. Daugiareikšmės priklausomybės atveju atributų reikšmę gali atitikti kelios kito atributo reikšmės. Daugiareikšmės priklausomybės sąvoka yra funkcinės priklausomybės apibendrinimas ta prasme, kad kiekviena funkcinė priklausomybė kartu yra ir daugiareikšmė priklausomybė. Tiksliau, funkcinė priklausomybė – tai daugiareikšmė priklausomybė, kurioje determinantą atitinkanti reikšmių aibė visada yra iš vieno elemento. Kaip ir funkcinių priklausomybių atveju, aktualios yra tik tokios daugiareikšmės priklausomybės, kurios lentelėje galioja visada, o ne kuriuo nors konkrečiu momentu. Daugiareikšmė priklausomybė A —>—> B yra netriviali lentelėje L(R), jei nė vienas atributas iš B neįeina į A ir, be atributų A ir B, lentelėje yra dar ir kitų atributų.

Tarkime, kiekvienas įstaigos darbuotojas gali turėti kelias kvalifikacijas ir mokėti kelias užsienio kalbas, o informacijai apie tai skirta lentelė Išsilavinimas(Nr, Kvalifikacija, Kalba), kurios stulpeliai atitinkamai reiškia darbuotojo numerį, įgytos kvalifikacijos pavadinimą ir užsienio kalbą, kurią jis moka. Ši lentelė turi vieną raktą, kurį sudaro visi trys lentelės atributai. Tarkime, darbuotojas Nr. 1 turi informatiko ir fiziko kvalifikaciją, taip pat moka dvi užsienio kalbas: anglų ir vokiečių. Lentelės fragmentas su duomenimis apie šį darbuotoją gali būti toks:



5.2 pav. Duomenų lentelė Išsilavinimas 1

Lentelėje Išsilavinimas yra dvi daugiareikšmės priklausomybės:

Nr. —>—> Kvalifikacija; Nr. —>—> Kalba. 45

Iš pirmo žvilgsnio gali atrodyti, kad turimai informacijai išreikšti pakanka dviejų eilučių lentelėje, pavyzdžiui, pirmos ir paskutinės. Tačiau tada gali būti suprasta, kad žmogus moka tik anglų kalbą, kai dirba informatiko darbą, ir tik vokiečių kalbą, kai fiziko. Taip, žinoma, neturėtų būti. Vienintelis būdas išreikšti, kad kvalifikacijos niekaip nesusiję su užsienio kalbomis, yra kiekvieną kalbą pakartoti su kiekviena kvalifikacija. Tai, be abejo, sukelia duomenų perteklių ir su tuo susijusias duomenų anomalijas. Pastebima, kad lentelėje nėra nepirminių atributų, todėl ji yra BKNF.

# IŠVADOS

Išnagrinėjus duomenų vaizdavimo modelius, galima daryti išvadą, kad tiek veiksmo diagrama, tiek duomenų modelis atlieka savo specifines funkcijas ir nėra vienas nuo kito priklausomi. Veiksmo diagrama naudojama apibrėžiant tikslus, iškeltus konkrečiai duomenų bazei – kokius veiksmus galima su ja atlikti, kokius reikalavimus ji turi atitikti. Duomenų modelis – tai duomenų susisteminimas, atitinkantis visus projektavimo reikalavimus. Jis vaizduojamas trijų metodologijų pagrindu.

Pagrindinės duomenų modelio sąvokos yra šios: esybė, atributas, pirminis raktas, domenas ir santykis tarp esybių. Visos šios sąvokos yra svarbios, nes papildo viena kitą, t.y. negali egzistuoti viena be kitos, bei palaiko duomenų modelio vientisumą. Kiekviena sąvoka, atskirai paėmus, yra specifinė kitų atžvilgiu ir kiekvienos iš jų specifiniai bruožai daro įtaką duomenų modeliui.

Esybės ir santykiai turi įvairių klasifikavimo būdų ir, priklausomai nuo duomenų bazės, šie būdai kiekvienu atveju gali būti skirtingi. Raktų klasifikavimas yra labiau koncentruotas, t.y. beveik kiekviena duomenų bazė turi pirminių, antrinių ir išorinių raktų, o tuo tarpu esybių santykiai negali būti vienu metu klasifikuojami pagal visus klasifikavimo požymius – konkrečiais atvejais santykiai yra klasifikuojami pagal vieną arba kelis klasifikavimo požymius. Esybių ir santykių klasifikavimui būdinga tai, kad kartais autorius, nagrinėdamas metodologijas, pasiūlo savo klasifikavimo požymius, todėl apžvelgti visus klasifikavimo požymius yra fiziškai neįmanoma.

Kad lentelė atitiktų visas normines formas, būtina, kad ji atitiktų visas normines formas paeiliui, t.y., neatitikdama žemesnės norminės formos, lentelė neatitiks ir aukštesnės. Norminės formos užtikrina duomenų įvedimo, pašalinimo ir papildymo anomalijų nebuvimą. Su kiekviena aukštesne normine forma reikalavimai lentelei vis didėja.

# LITERATŪROS SĄRAŠAS

*1) Baronas R. Duomenų bazių sistemos. Vilnius. TEV. 2002. [žiurėta 2018.12.2]*

*2) Minkevičius S. Microsoft Access XP pradžiamokslis.Vilnius. Piketas. 2003. [žiurėta 2018.12.2]*

*3) Misevičius A., Riškus A. Duomenų bazių valdymo sistema Visual Foxpro. Kaunas. Technologija. 2003. [žiurėta 2018.12.2]*

*4) Stanaitis Č. Access 2.0 ir 7.0 Kaunas. Smaltija. 1997. [žiurėta 2018.12.2]*

*5) Batini, C., S. Ceri, S. Kant, and B. Navathe. Conceptual Database Design: An Entity Relational Approach. The Benjamin/Cummings Publishing Company, 1991. [žiurėta 2018.12.2]*