TARTU ÜLIKOOL
Arvutiteaduste instituut
Informaatika õppekava

Indrek Lõbus

Puuduvate väärtuste probleem relatsioonilises andmemudelis ja selle loogikapõhine lahendus

Bakalaureusetöö (9 EAP)

Juhendaja:
Anne Villems

Tartu 2024

Puuduvate väärtuste probleem relatsioonilises andmemudelis ja selle loogikapõhine lahendus

Lühikokkuvõte:

See töö selgitab relatsioonilise andmemudeli loogilisi aluseid, tutvustab puuduvate väärtuste probleemi ning kaht varasemat katset seda lahendada ja esitab uue lahenduse, mis tugineb ideedele loogikafilosoofiast.

Võtmesõnad:

Relatsiooniline, andmebaas, predikaatarvutus, relatsioonalgebra, andmed, filosoofia

CERCS: P110, Matemaatiline loogika, hulgateooria, kombinatoorika

The Problem of Absent Values in the Relational Data Model and Its Logic-Based Solution

Abstract:

This thesis explains the logical foundations of the relational data model, introduces the problem of absent values and two existing attempts at solving it, and then presents a new solution based on ideas from philosophy of logic.

Keywords:

Relational, database, predicate, logic, data, philosophy

CERCS: P110, Mathematical logic, set theory, combinatories

Sisukord

Sissejuhatus 4

1. Relatsiooniline mudel ja selle loogiline alus 5

1.1. Andmebaas kui tõeste propositsioonide kogum 5

1.2. Päring kui tõestus 8

1.3. Relatsiooniline andmemudel 10

1.4. Puuduvate väärtuste probleem 14

2. Coddi lahendus ja selle probleemid 16

2.1. Coddi lahendus tühiväärtuste abil 16

2.2. Probleem tundmatute väärtustega 19

2.3. Probleem olematute väärtustega 20

3. Date'i lahendus ja selle probleemid 22

3.1. Date'i dekompositsiooniline lahendus 22

3.2. Probleem võtmeta relatsioonidega 26

3.3. Probleem horisontaalse dekompositsiooniga 27

4. Ramsey-Wittgensteini lahendus 30

4.1. Võtmeta relatsiooni dekompositsioon 30

4.2. Ramsey-Wittgensteini baaspropositsioonide käsitlus 33

4.3. Ramsey-Wittgensteini käsitluse põhine lahendus 37

Kokkuvõte 44

Kasutatud kirjandus 45

LISAD 47

I. Litsents 47

Sissejuhatus

Andmebaas sisaldab teavet maailma kohta. Enamikes eluvaldkondades kaasajal tuginetakse teabele viisil, mis eeldab selle salvestatavust ning süstemaatilist kättesaadavust. Seetõttu peavad ka andmebaasid olema sellele vastavalt ehitatud. Enim levinud andmebaaside arhitektuur kaasajal on E. F. Coddi välja pakutud relatsiooniline andmemudel, mis annab formaalloogikale tugineva raamistiku salvestatud andmete organiseerimiseks ja pärimiseks. Kuigi relatsioonilisest mudelist on saanud andmebaaside standard, on sellel ka probleeme. Üheks neist on lünklike andmete käsitlemine, s.t. andmete käsitlemine, millest osad väärtused puuduvad. Nimetan seda puuduvate väärtuste probleemiks. Selles töös pakun probleemile lahenduse, mis tugineb loogikafilosoofiast pärinevatel ideedel, mille on välja töötanud Frank Ramsey ja Ludwig Wittgenstein.

Kuna lahendus eeldab loogikafilosoofiast pärinevate ideede rakendamist andmebaasidele, selgitan esiteks andmebaaside ja loogika seost, mida eeldas relatsioonilist mudelit välja töötades ka Codd. Tutvustan seejuures lähemalt ka relatsioonilist andmemudelit ennast ning selgitan, kuidas selles puuduvate väärtuste probleem esile kerkib. Seejärel tutvustan Coddi enda lahendust probleemile, milles puuduvad väärtused asendatakse eriliste tühiväärtustega. See on ka kaasajal enim levinud viis puuduvate väärtuste käsitlemiseks. Näitan aga, et sellel lahendusel on vastuvõetamatuid tagajärgi, nimelt osutuvad osad päringute tulemused seetõttu valeks. Seejärel tutvustan C. J. Date'i lahendust, mis tühiväärtuseid ei kasuta. Näitan aga, et Date'i lahendus ei võimalda puuduvaid väärtuseid käsitleda kõigil võimalikel juhtudel. Lõpuks esitan omapoolse lahenduse, mis tugineb Ramsey ja Wittgensteini ideedele, ning mis ei ole samal viisil piiratud.

Töös keskendun vaid relatsioonilisele mudelile endale, jättes täielikult kõrvale selle implementeerimisega seotud küsimused. Seetõttu käsitlen ka andmebaasi juhtimissüsteeme vaid niivõrd kui need puudutavad andmete haldamise keelt. Käsitluse alt jäävad välja andmekeele lausete interpreteerimine masinas ning konkreetsed algoritmid tulemuste arvutamiseks. Sellise lähenemise eeliseks on, et esitatud lahendusele on seejärel võimalik välja töötada erinevaid implementatsioone, ilma et lahendus sellele piiranguid seaks. Ainus implementatsiooni puudutav eeldus, mida läbivalt teen, on et andmebaasis on alati lõplik kogus andmeid, kuna see kehtib ükskõik millise implementatsiooni korral.

1. Relatsiooniline mudel ja selle loogiline alus

Relatsioonilise andmemudeli väljatöötaja E. F. Codd kirjutab, et relatsiooniline andmemudel "võimaldab arendada predikaatarvutusel põhineva universaalse andmete alamkeele," mis "toimiks keelise võimsuse mõõdupuuna kõigile teistele väljapakutud andmekeeltele".[1] Predikaatarvutus on lausete vaheliste loogiliste seoste formaalseks käsitamiseks välja töötatud formaalkeel. Niisiis väidab Codd, et kõigi võimalike andmekeelte adekvaatsust peaks hindama selle alusel, kui hästi need keeled suudavad väljendada loogilisi seoseid lausete vahel. Esmapilgul ei ole ilmne, kuidas on seotud loogika, mille peamiseks eesmärgiks on kontrollida argumentide kehtivust, ning andmekeeled, mille peamiseks funktsiooniks on lugeda andmeid andmebaasist. Codd ise seda seost ka lähemalt ei selgita. Järgnevalt näitan aga, et loogika on tõepoolest tihedalt seotud andmetega ning on kohaseks mõõdupuuks andmekeelte hindamisel.

Esiteks selgitan, kuidas andmebaasist on alati võimalik mõelda kui lausete (täpsemalt, propositsioonide) kogumist, millel on võimalik teha loogilisi operatsioone, ning kuidas sellest lähtuvalt on andmete päringud käsitletavad kui tõestused. Seejärel näitan, kuidas niisugusest andmebaaside käsitlusest kasvab loomulikul teel välja Coddi relatsiooniline andmemudel. Lõpuks tutvustan järgnevates peatükkides käsitletavat probleemi relatsioonilises andmemudelis, mida nimetan puuduvate väärtuste probleemiks.

1.1. Andmebaas kui tõeste propositsioonide kogum

Et andmebaas sisaldaks teavet maailma kohta, peab andmetest andmebaasis olema võimalik välja lugeda seda, mis leiab aset reaalsuses. Selleks aga peab andmebaas väljendama asjade seisu reaalsuses. Teisisõnu, andmebaas peab olema *reaalsuse mingi osa tõepärane mudel*.[2] Reaalsusest võime mõelda kui faktide ehk tõsiasjade kogumist. Näiteks on reaalsuse osaks tõsiasjad, et Indrek Lõbus kirjutab informaatika lõputööd Tartus kuupäeval 19.07.2024, et keskmine brutopalk Eestis aastal 2023 oli 1832 eurot,[3] ning et Päikesesüsteemis on teadaolevalt 8 planeeti. Mudeldada mingit reaalsuse osa tähendab seega mudeldada mingit hulka tõsiasju. Tõsiasjast võime omakorda mõelda kui omavahel mingil viisil seotud

[1] Codd 1970: 381; I. L. tõlge
[2] Sellise andmebaasi definitsiooni pakkus ühe võimaliku definitsioonina välja Anne Villems aine "Andmebaasid" (LTAT.03.004) sissejuhatavas loengus. Definitsiooni ei ole välja toodud aga kursuse kirjalikes materjalides.
[3] Andmed pärinevad leheküljelt https://www.stat.ee/et/avasta-statistikat/valdkonnad/tooelu/palk-ja-toojoukulu/keskmine-brutokuupalk (külastatud 12.08.2024).

objektidest. Esimeses näites moodustavad tõsiasja näiteks isik Indrek Lõbus, Tartu linn, kuupäev 19.07.2024, kirjutamise tegevus, informaatika eriala ning lõputöö kui tekstiliik. Andmebaasis kui tõsiasjade mudelis peavad seega leiduma elemendid, mis vastaksid objektidele tõsiasjades, ning need elemendid peavad omavahel olema kokku pandud viisil, mis peegeldaks seda, kuidas neile vastavad objektid on omavahel tõsiasjades seotud. Need elemendid on andmebaasis esinevad andmed.

Selline andmebaaside käsitlus toob välja andmebaasi seose lausetega. Lause koosneb sobival viisil kokku pandud sõnadest, mis samuti vastavad objektidele, ning viis, kuidas sõnad on lauseks kokku pandud väljendab seda, kuidas need objektid peaksid omavahel tõsiasja moodustama, et lause oleks tõene. Andmed andmebaasis täidavad seega sarnast ülesannet nagu lausetes esinevad sõnad. Oluline seejuures on eelkõige see, et nii nagu sõnad, et neist saaks välja lugeda midagi reaalsuse kohta, peavad moodustama tõsiasju väljendavaid lauseid, peavad ka andmed andmebaasis moodustama midagi analoogset lausetele. C. J. Date'i sõnul — kes on olnud üks Coddi ideede peamisi edasiarendajaid — oli just see Coddi üks olulisemaid avastusi, nimelt et "andmebaas (hoolimata nimest) ei ole tegelikult pelgalt andmete kogum; pigem on see *tõsiasjade* kogum või teisisõnu tõeste propositsioonide kogum."[4]

Propositsioonimõiste, mida Date eelnevas tsitaadis kasutab, on pärit loogikafilosoofiast, kus see asendab tavapärast lausemõistet. Erinevalt lausest, mille olemuslike tunnuste juurde kuulub ka hääldus või kirjapilt ning see, kas lause on näiteks eesti-, inglise- või SQL-keelne, on propositsioonile olemuslikud kõik ja ainult need tunnused, mis on tarvilikud, et olla tõene või väär. Nii saame mõelda lausest kui propositsiooni implementatsioonist, kusjuures sama propositsioon võib olla implementeeritud erinevate lausetena, nimelt siis kui laused väljendavad sama tõsiasja. Mõeldes andmebaasist kui *tõeste propositsioonide kogumist* ei eelda me niisiis midagi selle kohta, kuidas on andmebaas implementeeritud ega millises konkreetses keeles andmebaasi vastu päringuid tehakse. Loeb vaid see, et kõik kirjed andmebaasis ütlevad, kuidas asjad reaalsuses on.

Kuna andmebaasist on võimalik mõelda kui tõeste propositsioonide kogumist, on andmebaasis põhimõtteliselt võimalik teha *ükskõik milliseid* loogilisi operatsioone. Seda seetõttu, et loogilised operatsioonid ongi sellised operatsioonid, mida on võimalik teha suvalistel propositsioonidel ehk objektidel, millel on tõeväärtused. Loogiliste operatsioonide tulemused on uued propositsioonid, mille tõeväärtused sõltuvad nende propositsioonide

[4] Date 2012: 100; I. L. tõlge

tõeväärtustest, millel operatsioonid on tehtud. Näiteks, propositsioonide A ja B konjunktsioon on propositsioon, mis on tõene parajasti siis kui A ja B on mõlemad tõesed, ning propositsiooni A negatsioon on propositsioon, mis on tõene parajasti siis kui A on väär. Ainus loogiline operatsioon, mille defineeritavus propositsioonide kaudu ei ole esmapilgul ilmne, on predikaatarvutuses esinev kvantifikatsioon. Enamikes loogikaõpikutes[5] on kvantoriga propositsiooni ⌜∀xA(x)⌝ tõesus defineeritud mitte propositsioonide tõeväärtuste kaudu vaid selle kaudu, millised muutujaväärtustused rahuldavad predikaati ⌜A(x)⌝, millest kvantoriga propositsioon on moodustatud. Nagu selgitab loogikafilosoof Michael Dummett, taandub aga tegelikult ka kvantoriga propositsiooni tõesus lõppkokkuvõttes teiste propositsioonide tõeväärtustele, kuna predikaat omakorda on defineeritav propositsioonide kaudu.[6] Just see, et propositsioonide kaudu on võimalik defineerida ka sellised väljendid nagu predikaadid, on aluseks ka Coddi relatsioonilisele andmemudelile, mistõttu väärib see lähemalt käsitlemist.

Propositsioone on võimalik määratleda nende vormi kaudu. Näiteks on propositsioon ‘Mari loeb’ määratletav kui propositsioon, mis koosneb isiku nimest ‘Mari’ ja verbist ‘loeb’, nii et nimele ‘Mari’ järgneb verb ‘loeb’. Propositsioonil ‘Mari loeb’ on aga ka vähem spetsiifilisi vorme. Näiteks on ‘Mari loeb’ ka propositsioon, mis koosneb *mingist isiku nimest x* ja verbist ‘loeb’, nii et nimele x järgneb ‘loeb’. See vorm on aga lisaks propositsioonile ‘Mari loeb’ ka näiteks propositsioonidel ‘Jüri loeb’ ja ‘Liisi loeb’ ning ei määratle seega enam mitte ühte propositsiooni vaid hulga erinevaid propositsioone, millel kõigil on seesama vorm. Seda vormi võime kirjutada kui ⌜x loeb⌝, kus muutuja ‘x’ tähistab kohta isiku nimele. Predikaatidest võimegi mõelda kui propositsioonide vormidest ning propositsioonidest seega kui predikaatide instantsidest. Predikaat on alati abstraheeritav propositsioonist ning propositsioonide kogum alati määratletav predikaadi abil. Propositsioonist ‘Mari loeb’ on abstraheeritav predikaat ⌜x loeb⌝, mis omakorda võimaldab määratleda kõik sellise vormiga propositsioonid: ‘Mari loeb’, ‘Jüri loeb’, ‘Liisi loeb’, jne. Kvantoriga propositsiooni tõesus on sellise käsitluse abil defineeritav propositsioonide tõeväärtuste kaudu järgmiselt: propositsioon ⌜∀xA(x)⌝ on tõene parajasti siis kui iga propositsioon kujul ⌜A(x)⌝ on tõene, kus x on term. Andmebaas kui tõeste propositsioonide kogum toetab seega ka kvantifikatsiooni.

[5] vt nt Halbach 2010, Sider 2010, Magnus 2017
[6] vt Dummett 1973: 16–17

1.2. Päring kui tõestus

See, et andmebaasis on võimalik teha ükskõik, milliseid loogilisi operatsioone, kindlustab, et andmebaasi jaoks on alati olemas ka viisid sinna andmete kirjutamiseks ja sealt kirjutatud andmete pärimiseks. Keskendun siin vaid andmete pärimisele.

Tavapäraselt peetakse päringukeele all silmas sellist keelt, milles ka ühe lause abil on võimalik andmebaasist kätte saada teoreetiliselt piiramatut hulka andmeid. Sedasorti päringud on näiteks SELECT-laused SQL-keeles, mille tulemuseks olevad tabelid võivad sisaldada ükskõik kui palju ridu.

Lihtsaim võimalik päring — mida Schweikardt *et al.* (2010) nimetavad tõeväärtuslikeks (ingl *boolean*) päringuteks[7] — oleks selline, mille lahendus seisneb kontrollis, kas mingi propositsioon Q esineb andmebaasis. Kui esineb, on päringu tulemus tõene, ning vastasel juhul on tulemus väär. Kuna andmebaas sisaldab vaid lõplikku hulka andmeid, on iga tõeväärtuslik päring lahenduv, s.t. annab vastuseks kas tõe või vääruse, sest alati on võimalik kõik propositsioonid andmebaasis kasvõi ükshaaval läbi kontrollida. Tõeväärtuslikud päringud, nii nagu Schweikardt *et al.* neid mõistavad, ei vaja lahendamiseks ühtki loogilist operatsiooni. Tõeväärtusliku päringu mõiste on aga loomulikul viisil laiendatav ka päringutele, mille lahendamine sisaldab ka loogilisi operatsioone nagu konjunktsioon ja negatsioon, mis on tehtud konkreetsetel propositsioonidel. Näiteks oleks päring propositsiooni ‘A ∧ ¬B’ kohta järgmine. Kuna ‘A ∧ ¬B’ on tõene parajasti siis kui ‘A’ ja ‘¬B’ on tõesed ning ‘¬B’ on tõene parajasti siis kui ‘B’ on väär, on vaja operatsioonide rakendamise järel kontrollida, kas andmebaasis leidub ‘A’ ning ei leidu ‘B’. Kui mõlema kontrolli tulemuseks on tõesus, on ka kogu päringu tulemuseks tõesus, ning vastasel juhul on tulemus väärus. Vastuseks saadud tõeväärtus on tõlgendatav kui propositsiooni ‘A ∧ ¬B’ tegelik tõeväärtus selles reaalsuse osas, mille mudeliks andmebaas on.[8]

Päring andmebaasi vastu, nagu täheldab Date, osutub seega *tõestuseks*: “Kui süsteem väärtustab mõnd relatsioonilist avaldist (eriti mõnele päringule vastates), tuletab ta tegelikult

[7] Schweikardt *et al*. 2010: 6. Schweikardt *et al.* ei mõista andmebaasi propositsioonide kogumina, mistõttu käsitlevad nad ka tõeväärtulikku päringut mitte propositsiooni olemasolu kontrollina andmebaasis, nagu seda siin olen esitanud, vaid korteeži olemasolu kontrollina relatsioonis ning rakendavad tõeväärtusliku päringu mõistet eelkõige seoses relatsiooniliste andmebaasidega. On aga ilmne, et tõeväärtusliku päringu mõiste on ka laiemalt rakendatav.

[8] Seetõttu võib näiteks tegelikult väär eitus ‘¬A’ osutuda andmebaasist lähtuvalt tõeseks, nimelt kui eitatud propositsioon ‘A’ on tegelikult tõene, kuid andmebaasis ei ole vastavat tõsiasja mudeldatud.

uusi tõdesid antud tõdedest; ehk sisuliselt tõestab teoreeme!"[9] Date ise paistab mõistvat päringuid rangelt tuletusteoreetiliste tõestustena, mis seisnevad propositsioonidest tuletusreegleid järgides uute propositsioonide konstrueerimises. Tuletusteoreetilised tõestusmeetodid nagu loomulik tuletus või sekventsarvutus võimaldavad vaid tõeste propositsioonide kohta näidata, et nad on tõesed. Andmebaas aga võimaldab ka semantilisi tõestuseid nagu tõestuseid tõeväärtustabeli meetodil, kus on eksplitsiitselt viidatud väljendite väärtustele, ning mis võimaldavad näidata *kas* propositsioon on andmebaasist lähtuvalt tõene või väär. Kuigi kvantoriga propositsioonide jaoks ei ole see loogikas üldjuhul võimalik, on see siiski võimalik andmebaasis. Kuna andmebaasis on lõplik arv propositsioone, on alati võimalik kvantoriga propositsiooni tõeväärtus tuvastada kõiki propositsioone andmebaasis ükshaaval läbi kontrollides.

Kuna tõestus on eeldustest järelduse suunas tõesust säilitav seos, on selge, kuidas tõeväärtuslikud päringud põhimõtteliselt võiksid olla tõestused, kuid esmapilgul ei pruugi olla ilmne, kuidas võiksid olla tõestused ka niisugused päringud, mis annavad tulemuseks mingi hulga andmeid. Enamik huvitavaid päringuid on just sedasorti, kuna tõeväärtuslikud päringud annavad vaid väga vähe informatsiooni. Seetõttu võimaldavad ka kõik reaalselt kasutusel olevad andmete pärimise keeled, k.a. SQL, just andmehulki tagastavaid päringuid. Kuna andmebaas on propositsioonide kogum, tähendab andmehulga tagastamine mingi hulga propositsioonide tagastamist. Päringus, mis seda teeb, peaks olema seega täpsustatud, millised propositsioonid peaksid tulemuses esinema. Ainus viis seda teha propositsioone päringus ükshaaval loetlemata, on täpsustada soovitud propositsioonid päringus vormi alusel ehk predikaadi abil. Kuna predikaat määratleb kogumi sama vormi jagavaid propositsioone, seisneb predikaadiga päringu lahendamine lihtsalt tõeväärtuslike päringute lahendamises iga propositsiooni kohta, mis predikaadiga määratletud propositsioonide hulka kuulub. Predikaadiga päringu tulemuseks on seega kõigi nende tõeväärtuslike päringute tulemused, üks iga propositsiooni kohta predikaadiga määratletud kogumis. Täpsemalt on tegu funktsiooniga predikaadiga määratletud propositsioonidest nende propositsioonide tõeväärtustesse andmebaasis.

Nii nagu tõeväärtusliku päringu tulemus on päringu propositsiooni tõeväärtus andmebaasis, on ka predikaadiga päringu tulemuseks predikaadi väärtus andmebaasis. See ei ole aga esmapilgul ilmne. Loogikas on tavapärane käsitleda predikaatide väärtuseid

[9] Date 2012: 100; I. L. tõlge

relatsioonidena hulgateooria mõttes.[10] (Järgides loogikas levinud terminoloogiat, nimetan relatsioonina esitatud predikaadi väärtust siin ja edaspidi predikaadi ekstensiooniks.) Täpsemalt, *n*-kohaline relatsioon on *n* hulga otsekorrutise alamhulk ning seega mingi *n*-pikkuseliste korteežide hulk. Üks predikaadi ekstensiooni kuuluv korteež esindab üht predikaadis esinevate muutujate väärtuste komplekti, mille kohta predikaat on tõene. Nagu selgitasin eelnevalt, on predikaadi igale muutujale mingi väärtuste määramine sisuliselt aga ühe propositsiooni valimine predikaadiga määratletud propositsioonide hulgast. Korteežid predikaadi ekstensioonis vastavad seega tõestele propositsioonidele, millel on predikaadiga esitatud vorm. Predikaadiga päringu tulemuseks olev funktsioon propositsioonidest tõeväärtustesse on aga omakorda vaadeldav kui karakteristlik funktsioon selliste propositsioonide hulgale. Kuna hulkade ja nende karakteristlike funktsioonide vahel on üks ühele vastavus, on predikaadiga päringu tulemus seega lõppkokkuvõttes käsitletav predikaadi ekstensioonina ja vastupidi. Et predikaatidega päringute tulemused on käsitletavad relatsioonidena, ongi aluseks Coddi väljatöötatud relatsioonilisele andmemudelile, mida järgnevalt lähemalt tutvustan.

1.3. Relatsiooniline andmemudel

Kuigi käsitlus, mille kohaselt andmebaas võimaldab relatsioonilisi päringuid, on juba ise lähedane relatsioonilisele andmemudelile, on oluline silmas pidada, et sellise tulemuseni jõudmiseks piisas vaid ühest eeldusest: et andmebaasist on võimalik välja lugeda, milline on asjade seis reaalsuses. See eeldus aga kehtib *iga* andmebaasi kohta. Relatsioonilist mudelit järgiv andmebaas on *optimeeritud* relatsiooniliste päringute jaoks. Nõuded sellisele optimeeringule võttis Codd kokku oma kuulsa kaheteistkümne reegliga. Enamik neist puudutavad andmebaasi ja selle juhtimissüsteemi implementeerimisega seotud tehnilisi detaile ning jäävad siinse töö teema alt välja. Esimesed kaks Coddi kaheteistkümnest reeglist, nn. *informatsioonireegel* ja *garanteeritud ligipääsu reegel*, võtavad aga kokku relatsioonilise andmebaasi formaalsed eripärad teiste andmebaasidega võrreldes.[11]

Informatsioonireegel ütleb, et "[k]ogu informatsioon relatsioonilises andmebaasis on eksplitsiitselt esitatud loogilisel tasandil ja täpselt ühel viisil — väärtustena tabelites."[12]

[10] vt nt Halback 2010: 98, Sider 2010: 118

[11] Ka Coddi kolmas reegel ehk süstemaatilise tühiväärtuste käsitluse reegel puudutab relatsioonilise andmebaasi formaalseid eripärasid. Käsitlen seda järgmises peatükis.

[12] Codd 1985; I. L. tõlge

Teisisõnu, relatsioonilises mudelis on andmed alati esitatud relatsioonis ehk mingi predikaadi ekstensioonis. Ühtlasi lubab relatsiooniline mudel seetõttu andmebaasi vastu teha ainult relatsioonilisi päringuid, kuna informatsioonireegel käib ka päringuga leitud andmete kohta. Nagu eelnevalt näitasin, ei ole selline piirang loogika seisukohast nõutud. Loogikast lähtuvalt oleksid võimalikud ka tõeväärtuslikud päringud. Samuti on kõrgemat järku predikaatarvutuses, kus on kasutusel lisaks objektimuutujatele ka predikaadimuutujad, võimalik koostada ka päringuid, mille tulemuseks on relatsioonide hulgad.

Selguse huvides vajab märkimist, et informatsioonireegli sõnastuses ning aegajalt ka mujal räägib Codd relatsioonidest kui tabelitest. Samas märgib Codd ka ise, et lähtuvalt tavamõistuslikust arusaamast tabelite kohta ei ole tabel ja relatsioon samaväärsed.[13] Tabelis on read järjestatud. Näiteks on suvalise tabeli rea kohta võimalik küsida, milline on talle eelnev või järgnev rida. Samuti võib tabelis põhimõtteliselt esineda ka korduvaid ridu. Seevastu "[r]elatsiooniline mudel," nagu Codd rõhutab, "käsitleb korteeže lähtuvalt nende informatsioonilise sisust."[14] Relatsioonilises mudelis ei ole seetõttu lubatud midagi, mida ei ole vaja tõsiasjade väljendamiseks. Kuna propositsioonide järjekord ja kordamine ei mõjuta seda, milliseid tõsiasju nad väljendavad, on relatsioonilise mudeli järgi ka korteežid relatsioonis järjestamata ning unikaalsed. See õigustabki relatsioonilises mudelis andmete esitamist relatsioonidena ehk korteežide hulkadena.

Garanteeritud ligipääsu reegel ütleb, kuidas on andmed andmebaasis kättesaadavad, arvestades, et nad on informatsioonireeglile vastavalt alati organiseeritud relatsioonidesse. Täpsemalt, garanteeritud ligipääsu reegli kohaselt on "[i]ga atomaarne väärtus relatsioonilises andmebaasis … garanteeritult ligipääsetav relatsiooni nime, võtme ning tunnuse nime kombinatsiooni abil."[15] Kuna tunnuse ja võtme mõistet predikaatarvutuses pealtnäha ei esine, vajavad mõlemad aga selgitamist.

Et relatsioon oleks predikaadi ekstensioon ning esindaks seega propositsioonide kogumit, peab iga positsioon relatsiooni korteežides vastama muutujale predikaadis, mille väärtusteks vastavas positsioonis esinevad objektid on. Tunnus (ingl *attribute*) relatsioonis ongi muutujale vastav positsioon relatsiooni korteežides. Predikaatarvutuses tähistavad relatsiooni tunnuseid tavapäraselt predikaadis esinevate muutujate järjekorra indeksid. Esimene objekt igas korteežis vastab esimesele muutujale predikaadis, teine objekt teisele muutujale, jne.

[13] vt Codd 1990: 17–20
[14] Codd 1990: 2; I. L. tõlge
[15] Codd 1985; I. L. tõlge

Loogika seisukohast ei ole aga tunnuste tähistusviis oluline. Codd võtab relatsiooni tunnuste tähistajatena muutujate järjekorra indeksite asemel kasutusele tunnuste nimed. Sellise ülesehituse eeliseks on, et objektide järjekord korteežis ei ole enam oluline tuvastamaks, milline korteež relatsioonis esindab millist propositsiooni. Esitades relatsioone tabelitena, kus read vastavad korteežidele ning veerud tunnustele, tähendab see, et võime ignoreerida mitte üksnes tabeli ridade järjekorda vaid ka veergude järjekorda, kui kirjutame iga veeru päisesse tunnuse nime, mille väärtused vastavas veerus on esitatud. Näiteks on tabelid A ja B mõlemad loogika seisukohast võrdväärsed viisid esitada predikaadi ˹R(x, y)˺ ekstensiooni {(a, b), (c, d)}, kus tunnuste nimedena on kasutatud predikaadi muutujaid:[16]

A

x	**y**
a	b
c	d

B

y	**x**
d	c
b	a

Codd käsitleb tunnuseid sel viisil eelkõige praktilistel kaalutlustel, kuna kasutajatel ei ole sel juhul päringuid koostades vaja teada tunnuste järjekorda.[17]

Relatsiooni (primaarne) võti (ingl *primary key*) on minimaalne hulk relatsiooni tunnuseid, mille väärtused koos on iga võimaliku relatsioonis esineva korteeži jaoks unikaalsed ning võimaldavad seetõttu iga korteeži relatsioonis tuvastada.[18] Kui relatsiooni tunnuste ükski pärisosahulk sellele tingimusele ei vasta, vastab sellele tingimusele relatsiooni kõigi tunnuste hulk, kuna relatsioonides ei esine korduvaid korteeže. Seega on igal relatsioonil olemas vähemalt üks võti.

Garanteeritud ligipääsu reegli järgi on niisiis iga väärtus andmebaasis leitav järgmiselt. Relatsiooni nimi ütleb, millises relatsioonis on otsitav väärtus; võti ütleb, millises korteežis on

[16] Siin ja edaspidi kasutan lihtsuse huvides tunnuste nimedena predikaadi muutujaid. See ei ole aga võimalik iga predikaadi puhul. Näiteks on predikaadil ˹R(x, x)˺ vaid üks muutuja kuid siiski kaks tunnust. Universaalselt kasutatav tähistusviis tunnustele oleks näiteks kirjutada muutujale vastava tunnuse nimi muutuja alaindeksina.

[17] vt Codd 1990: 3

[18] vt Codd 1970: 380. Koos primaarse võtme mõistega on kasutusel välisvõtme (ingl *foreign key*) mõiste. Välisvõti on tunnuste kogum relatsioonis, mille väärtused on mingi relatsiooni primaarse võtme väärtused, võimaldades nii korteeže relatsioonides omavahel kokku viia (vt Codd 1970: 380). Kuna välisvõtme mõiste ei ole edasises oluline, nimetan primaarset võtit selle töö raames lihtsalt võtmeks.

otsitav väärtus relatsioonis; ning tunnuse nimi ütleb, millisel positsioonil on otsitav väärtus korteežis. Kui kõik andmed andmebaasis esinevad relatsioonides, on selline ligipääs iga väärtuse jaoks garanteeritud.

Andmebaas, mis on korraldatud relatsioonilise mudeli põhjal, võimaldab ühtlasi andmete pärimist relatsioonalgebra operatsioonide abil. Relatsioonalgebra operatsioonid on operatsioonid relatsioonidel, mille tulemuseks on uued relatsioonid. Peamisteks relatsioonilises andmebaasis kasutatavateks relatsioonalgebra operatsioonideks on relatsioonide ühend, vahe, otsekorrutis, projektsioon ning piirang.[19] Lisaks on relatsioonalgebras veel ümbernimetuse operatsioon relatsioonide ja nende tunnuste tähistuste muutmiseks.

Relatsioonide ühend ja vahe on moodustatavad vaid relatsioonidel, mis jagavad kõiki tunnuseid. Relatsioonide A ja B ühend, $A \cup B$, on relatsioon, kuhu kuuluvad korteežid, mis kuuluvad kas relatsiooni A või relatsiooni B, ning A ja B vahe, $A - B$, on relatsioon, kuhu kuuluvad korteežid, mis esinevad relatsioonis A aga mitte relatsioonis B. Relatsioonide A ja B otsekorrutis, $A \times B$, on relatsioon, kuhu kuuluvad korteežid $(a_1, \ldots, a_n, b_1, \ldots, b_m)$, kus $(a_1, \ldots, a_n)$ kuulub relatsiooni A ning $(b_1, \ldots, b_m)$ kuulub relatsiooni B. Relatsioonide tunnused peavad otsekorrutise moodustamiseks olema erinevalt nimetatud, et otsekorrutises oleksid eristatavad mõlema esialgse relatsiooni tunnused. Projektsioon ja piirang relatsioonil võimaldavad moodustada uue relatsiooni, millel on vastavalt kas vaid valitud tunnused või valitud korteežid esialgsest relatsioonist. Täpsemalt, relatsiooni A projektsioon tunnustele $T_1, \ldots, T_n$, ehk $\pi_{T_1,\ldots,T_n}(A)$, on relatsioon, millel on vaid tunnused $T_1, \ldots, T_n$ relatsiooni A tunnuste hulgast. Piirang relatsioonil A, ehk $\sigma_{\varphi}(A)$, on aga relatsioon, kuhu kuuluvad relatsiooni A korteežid, mis vastavad tingimusele φ, mis on esitatud predikaatarvutuse valemina.[20]

Kuigi tavapäraselt defineeritakse relatsioonalgebra operatsioone hulgateooria mõistete kaudu, on kõik viis operatsiooni defineeritavad ka predikaatarvutuses. See näitab, et ka relatsioonalgebras esitatud päringud on mõistetavad predikaatarvutuse tõestustena.

[19] Erinevates relatsioonalgebra esitustes võivad baasoperatsioonid erineda. Siin valitud operatsioone kasutab ka näiteks Brown (2024). Codd (1970) kasutab aga veidi erinevaid operatsioone. Siin valitud opertatsioonide kaudu on aga defineeritavad kõik ülejäänud relatsioonalgebra operatsioonid. Märkus ka piiranguoperatsiooni nime kohta: Piiranguks (ingl *restriction*) nimetavad seda operatsiooni näiteks Codd (1970) ning tema järgi ka Date ja Darwen (2014). Enamikes sissejuhatavates veebimaterjalides (nt Brown 2024) nimetatakse seda aga valikuks (ingl *selection*).

[20] Definitsioonid relatsioonalgebra operatsioonidele leiab näiteks allikatest Codd 1969, 1970, 1990; Date, Darwen 2014; Brown 2024. Siin kasutatud tähistused relatsioonalgebra operatsioonidele on standardsed ning kasutusel näiteks allikas Brown 2024.

Definitsioonid predikaatarvutuses on järgmised, kus ˹⟦P⟧˺ tähistab predikaadi P ekstensiooniks olevat relatsiooni:

(∪) $⟦\mathrm{A}(x_1, \ldots, x_n)⟧ \cup ⟦\mathrm{B}(x_1, \ldots, x_n)⟧ = ⟦\mathrm{A}(x_1, \ldots, x_n) \vee \mathrm{B}(x_1, \ldots, x_n)⟧$,

(–) $⟦\mathrm{A}(x_1, \ldots, x_n)⟧ - ⟦\mathrm{B}(x_1, \ldots, x_n)⟧ = ⟦\mathrm{A}(x_1, \ldots, x_n) \wedge \neg\mathrm{B}(x_1, \ldots, x_n)⟧$,

(×) $⟦\mathrm{A}(x_1, \ldots, x_n)⟧ \times ⟦\mathrm{B}(y_1, \ldots, y_m)⟧ = ⟦\mathrm{A}(x_1, \ldots, x_n) \wedge \mathrm{B}(y_1, \ldots, y_m)⟧$,

(π) $\pi_{x_1,\ldots,x_n}(⟦\mathrm{A}(x_1, \ldots, x_n, y_1, \ldots, y_m)⟧) = ⟦\exists y_1 \ldots \exists y_m \mathrm{A}(x_1, \ldots, x_n, y_1, \ldots, y_m)⟧$,

(σ) $\sigma_{\varphi(z_1,\ldots,z_m)}(⟦\mathrm{A}(x_1, \ldots, x_n)⟧) = ⟦\mathrm{A}(x_1, \ldots, x_n) \wedge \varphi(z_1, \ldots, z_m)⟧$,

kus {‘x_1’, …, ‘x_n’} ∩ {‘y_1’, …, ‘y_m’} = ∅ ja {‘z_1’, …, ‘z_m’} ⊆ {‘x_1’, …, ‘x_n’}.

Definitsioonid (∪), (–) ja (×) on lihtsalt hulgateoreetiliste definitsioonide formuleeringud predikaatarvutuse operaatorite abil. Definitsioonid (π) ja (σ) vajavad aga selgitust. Projektsioon säilitab esialgses relatsioonis mingid valitud tunnused. Predikaatarvutuses tähendab see, et predikaadis saavad seotud kõik muutujad peale valitud muutujate. Kuna projektsioonis ei ole tähtsust, millised väärtused esinevad välja jäetud tunnustes, on sobivaks kvantoriks ülejäänud muutujate sidumisel olemasolukvantor. See annab definitsiooni (π). Piirangus esinevad esialgsest relatsioonist vaid piirangu tingimusele vastavad korteežid. Predikaatarvutuses tähendab see, et predikaadi muutujate väärtused täidavad lisaks esialgses predikaadis esitatud tingimusele veel ühe predikaadiga esitatud tingimust. Täiendavas predikaadis ei tohi seejuures olla muutujaid, mida esialgses predikaadis ei esine. Vastasel juhul oleks tulemuseks predikaat, mille ekstensioonil on tunnuseid, mida ei ole esialgse predikaadi ekstensioonil. Nii on tulemuseks definitsioon (σ).

1.4. Puuduvate väärtuste probleem

Olen nüüdseks tutvustanud relatsioonilist andmemudelit ning selle loogilist alust. Pealtnäha teeb relatsiooniline mudel aga eelduseid reaalsuse kohta, mis alati ei kehti, nimelt et tõsiasjade mudelid on alati grupeeritavad tunnuste alusel relatsioonidesse.

Reaalsuse mudeldamiseks on vaja teada, mis leiab reaalsuses aset. Alati on aga võimalik, et mõni detail mõnest tõsiasjast on teadmata. Näiteks on võimalik, et me ei tea kõigi tudengite telefoninumbrit, kuigi teame neist igaühe matriklinumbrit, nime ja sünnikuupäeva. Samas on aga ka osaline tõsiasja mudel tõepärane nende tõsiasja aspektides suhtes, mida mudel siiski kajastab. Andmebaas peaks seega võimaldama ka osalisi tõsiasjade mudeleid. Samuti ei pruugi

muus osas sarnastes tõsiasjades endis esineda ühepalju objekte. Näiteks võib mõnel tudengil ka reaalsuses telefoninumber puududa.

Predikaadi ekstensioon ei saa sisaldada korteeže, millest mõned väärtused puuduvad. Korteež predikaadi ekstensioonis tähistab üht viisi, kuidas igale muutujale predikaadis väärtus määrata nii, et tulemuseks oleks tõene propositsioon. Kui mõni muutuja predikaadis väärtuseta jääks, ei oleks tulemuseks aga mitte propositsioon vaid uus predikaat.[21] Näiteks, kui predikaadis ˹R(x, y, z)˺ määrata väärtused vaid esimesele kahele muutujale, oleks tulemuseks ühe muutujaga predikaat ˹R(a, b, z)˺. Saadud predikaat annaks propositsiooni vaid siis, kui ka muutujale ‘z’ väärtus määrata.

Seda, kuidas relatsioonilises andmebaasis arvestada andmetega, milles on puuduvaid väärtuseid, nimetan *puuduvate väärtuste probleemiks*. Järgnevas kahes peatükis annan ülevaate kahest katsest puuduvate väärtuste probleemi lahendada ning toon välja probleeme mõlemas. Viimases peatükis esitan omapoolse lahenduse probleemile, mis varasemate katsete probleeme väldib.

[21] Analoogse põhjenduse annab ka Date (2012: 308), kuid selle erinevusega, et Date eeldab (ekslikult), et väärtuseta jäänud muutuja tõttu oleks tulemus tähendusetu, mitte uus predikaat.

2. Coddi lahendus ja selle probleemid

Relatsioonilist mudelit välja töötades oli Codd puuduvate väärtuste probleemist teadlik ning pakkus sellele ka oma lahenduse. Coddi lahendus on ühtlasi enim levinud lahendus puuduvate väärtustega andmete haldamiseks kuna on kasutusel enim levinud andmekeeles SQL. Siin peatükis annan ülevaate Coddi lahendusest puuduvate väärtuste probleemile ning toon välja mõned selle puudused. Kuna Coddi lahendus on saanud puuduvate väärtuste probleemi standardseks lahenduseks, annab selles puuduste tuvastamine põhjuse otsida alternatiivseid lahendusi, mida teen peatükkides 3 ja 4.

2.1. Coddi lahendus tühiväärtuste abil

Coddi lahendus puuduvate väärtuste probleemile on lubadata relatsioonilisse mudelisse nn *tühiväärtused* (ingl *null values*) — erilised väärtuseid, mille ülesandeks on tähistada tavapärase väärtuse puudumist andmetes. Selle lahenduse lisas Codd ka oma kaheteistkümne reegli hulka. Coddi kolmas reegel ehk *süstemaatilise tühiväärtuste käsitluse reegel* ütleb:

> Tühiväärtused (erinevalt tühjade sümbolite jadast või tühikute jadast ning erinevalt nullist või mõnest muust arvust) on täielikult relatsioonilistes andmebaasi juhtimissüsteemides toetatud puuduvate ja mitte-asjakohaste andmete süstemaatiliseks esitamiseks, sõltumatult andmetüübist.[22]

Coddi-stiilis lahendus puuduvate väärtuste probleemile on kasutusele võetud ka SQL-keeles, kus tühiväärtuseks on väärtus NULL.

Selleks, et tühiväärtused aitaksid puuduvate väärtuste probleemi lahendada, peab neil olema justkui kahetine loomus. Neid ei saa mõista tavapäraste väärtustena, kuna tühiväärtuste eesmärk on tähistada väärtuste *puudumist*. Samas aga peavad tühiväärtused vähemalt selles osas käituma nagu tavapärased väärtused, et nad peavad olema määratavad väärtusteks predikaadi muutujatele, nii et tulemuseks oleks propositsioon. Teise omaduse kohta ei ole midagi selgitavat lisada, peale selle, et peame leppima, et propositsioonid nagu ‘Mari isikukood on 60101010101 ning tema telefoninumber on NULL’ (kus ‘NULL’ tähistab tühiväärtust) on tähenduslikud ning saavad olla tõesed või väärad. Nii jääb üle selgitada, mille poolest tühiväärtused tavapärastest väärtustest erinevad.

[22] Codd 1985; I. L. tõlge

Nagu ilmneb Coddi reegli sõnastusest on tühiväärtused mõeldud täitma kaht erinevat ülesannet: nad peavad tähistama nii tundmatuid väärtuseid kui ka olematuid väärtuseid, s.t. väärtuseid, mida reaalsuses ei leidu. Et sellest tulenevaid mitmeti mõistetavusi andmebaasis vältida, tuleks rangelt võttes eristada kaht erinevat tühiväärtust. Põhjus, miks Codd sellist eristust ei tee ning miks sellist eristust ka SQL-keeles tehtud ei ole, on arvatavasti selles, et operatsioonid tundmatutel ja olematutel väärtustel käituvad sarnaselt. Keskendun vaid võrdlusoperatsioonidele ning loogilistele operatsioonidele.

Võrdlusoperatsioonid, kus vähemalt üks võrreldavatest on tühiväärtus, annavad tulemuseks tõeväärtuse asemel samuti tühiväärtuse, sõltumata sellest, kas tõlgendame tühiväärtuseid tundmatute või olematute väärtustena. Näiteks, kui *a* on tundmatu, ei ole võimalik otsustada, kas *a* on väiksem kui 5, mistõttu on propositsiooni '$a < 5$' tõeväärtus sel juhul samuti tundmatu. Kui aga *a* on olematu — näiteks igakuise stipendiumi suurus tudengil, kellele stipendiumi ei ole määratud —, siis ei ole propositsiooniga '$a < 5$' väljendatud võrdluses midagi võrreldud, mistõttu ei ole sel propositsioonil ka tõeväärtust ning propositsioon saab tõeväärtuse asemel olematu väärtuse.

Loogilised operatsioonid, kus vähemalt üks operandidest on tühiväärtusega, annavad tulemuseks tühiväärtuse parajasti siis, kui operandide tõeväärtused — tõde või väärus — ei ole kogu propositsioonile tõeväärtuse määramiseks piisavad. Niisuguseid loogilisi operatsioone kirjeldavad tavapärased kolme väärtusega loogika tõeväärtustabelid.[23] Disjunktsiooni, konjunktsiooni ja negatsiooni tõeväärtustabelid kolme väärtusega loogikas on näiteks järgmised (kus paksemas kirjas on esitatud operandide väärtused ja tavapärases kirjas neile vastavad operatsiooni väärtused ning 'T', 'V' ja 'N' tähistavad vastavalt tõde, väärust ja tühiväärtust):

¬	
T	V
V	T
N	N

∧	**T**	**V**	**N**
T	T	V	N
V	V	V	V
N	N	V	N

∨	**T**	**V**	**N**
T	T	T	T
V	T	V	N
N	T	N	N

[23] vt nt Sider 2010: 95

Disjunktsioon tõene parajasti siis kui vähemalt üks tema operandidest on tõene ning väär kui mõlemad operandid on väärad. Seega on disjunktsiooni väärtuseks tühiväärtus — kas tundmatu või olematu — parajasti siis kui kumbki tema operandidest ei ole tõene ning vähemalt üks neist on tühiväärtusega. Konjunktsiooni ja negatsiooni tõeväärtustabelid moodustuvad samal põhimõttel.

Andmebaasi tühiväärtuste lubamisel on ka mõned veidrad tagajärjed. Esiteks on tühiväärtuste tõttu võimalik, et relatsioonis esineb korteeže, mis intuitiivselt on korduvad. Selle selgitus vajab formaalset esitust. Tähistan väärtust, mille korteež k annab tunnusele T, kui $\ulcorner k[\mathrm{T}] \urcorner$. Lähtuvalt hulgateooriast, on korteežid k_1 ja k_2 üks ja sama korteež parajasti siis, kui nad annavad samadele tunnustele samad väärtused, ehk parajasti siis, kui iga tunnuse T korral osutub võrdlus $\ulcorner k_1[\mathrm{T}] = k_2[\mathrm{T}] \urcorner$ tõeseks. Võrdlusoperatsioonid tühiväärtustel annavad aga tulemuseks samuti tühiväärtuse. Seega, kui leidub tunnus N, millele k_1 ja k_2 annavad mõlemad tühiväärtuse, on võrdluse $\ulcorner k_1[\mathrm{N}] = k_2[\mathrm{N}] \urcorner$ väärtuseks samuti tühiväärtus. Järelikult ei ole sel juhul iga tunnuse T korral võrdlus $\ulcorner k_1[\mathrm{T}] = k_2[\mathrm{T}] \urcorner$ tõene ning k_1 ja k_2 osutuvad erinevateks korteežideks ning võivad seega esineda samas relatsioonis.

Samal põhjusel ei ole tühiväärtustega andmebaasis garanteeritud, et igal relatsioonil on olemas võti. Võtme väärtused on iga relatsioonis esineva korteeži jaoks unikaalsed. Võtme väärtuste alusel peaks olema korteež alati teistest korteežidest samas relatsioonis eristatav. Seetõttu ei saa aga võtme väärtuseks olla tühiväärtus. Ka see vajab formaalset selgitust. Tähistan tunnuse T väärtust v kui $\ulcorner v_{\mathrm{T}} \urcorner$. Kui tunnused $\mathrm{T}_1, \ldots, \mathrm{T}_n$ moodustavad relatsiooni võtme, leidub relatsioonis täpselt üks korteež k, mille puhul iga i korral on võrdlus $\ulcorner v_{\mathrm{T}_i} = k[\mathrm{T}_i] \urcorner$ tõene. Olgu k suvaline korteež relatsioonis ning N selle relatsiooni tunnus, mille korral $k[\mathrm{N}]$ on tühiväärtus. Kuna tühiväärtusega võrdlused väärtustuvad tühiväärtuseks on sel juhul võrdluse $\ulcorner v_{\mathrm{N}} = k[\mathrm{N}] \urcorner$ väärtuseks samuti tühiväärtus ning N ei ole osa relatsiooni võtmest. Mitte ükski tunnus, mille võimalike väärtuste seas on tühiväärtuseid, ei saa seega olla osa relatsiooni võtmest. Relatsioonides, kus iga tunnus võib võtta kas tühiväärtuseid või korduvaid väärtuseid, ei ole seega võtit.

See, et tühiväärtuste tõttu võib relatsioonis esineda pealtnäha korduvaid korteeže ning relatsioonidel ei ole tühiväärtuste tõttu võtme garantiid, võib tühiväärtuste kasutamise mõnes olukorras muuta ebasoovitavaks. See aga ei tähenda, et Coddi lahendus põhimõtteliselt ei sobiks puuduvate väärtuste tähistamiseks. Tühiväärtustega kaasnevate eripäradega on vaja andmebaasis lihtsalt arvestada. Coddi pakutud lahendusel on aga ka põhimõttelisi probleeme.

See, millised probleemid tühiväärtustega seoses täpsemalt tekivad, sõltub sellest, kas tõlgendame tühiväärtuseid tundmatute või olematute väärtustena. Järgnevalt selgitan kummagi tõlgendusega seotud probleeme lähemalt.

2.2. Probleem tundmatute väärtustega

Tundmatutena tõlgendatult võimaldavad tühiväärtused olukordi, kus andmebaasi vastu tehtud päringud annavad valesid tulemusi. Tõestus sellele pärineb Date'ilt.[24] Date esitab tõestuse SQL-keele päringu näitel. Siin kasutan aga ühtsuse huvides sellele analoogset päringut eespool tutvustatud relatsioonalgebras. Olgu andmebaasis kaks relatsiooni, mis tabeli kujul esitatult on järgmised (kus 'NULL' tähistab tühiväärtust):[25]

ISIK

ID	**NIMI**
60101010101	NULL

TUDENG

NIMI
'Indrek'

Vaatame järgnevat nende tabelite vastu tehtud päringut, millega küsime nende isikute id väärtuseid, kelle nimi kas ei esine tudengite tabelis või ei ole 'Mari':

(A) $\sigma_{\neg \text{ISIK.NIMI} = \text{TUDENG.NIMI} \vee \neg \text{ISIK.NIMI} = \text{'Mari'}}(\text{ISIK} \times \text{TUDENG})$

Päring on tehtud kahe relatsiooni otsekorrutise vastu, milles on juhtumisi täpselt üks korteež, nimelt (60101010101, NULL, 'Indrek'). Probleem päringuga ilmneb selles, kas päringus esineva piiranguoperatsiooni tingimus on selle korteeži kohta tõene. Asendades tingimuses muutujad väärtustega, mis peegeldavad maailma tegelikku olukorda, saame järgmise tulemuse, kus '*c*' on kohatäitjaks isiku tegelikule nimele, kelle isikukoodiks on 60101010101 (nimetan teda edaspidi isikuks 101):

(A_1) $\neg c = \text{'Indrek'} \vee \neg c = \text{'Mari'}$

Isiku 101 nimi, mis andmebaasis on tundmatu, *reaalsuses* kas on või ei ole 'Mari'. Kui tema nimi on 'Mari', osutuks (A_1) tema kohta tõeseks seetõttu, et vasak disjunkt oleks sel juhul tõene. Kui tema nimi aga ei ole 'Mari', osutuks (A_1) tema kohta tõeseks, kuna tõene oleks

[24] vt Date 2012: 74–76
[25] Siin esitatud näide on kohandatud Date'i (2012: 74–75) näitest.

parem disjunkt. Seega on (A_1) isiku 101 kohta tõene tema tegelikust nimest sõltumata ning päringu (A) oodatav tulemuseks olev relatsioon sisaldaks korteeži (60101010101, NULL, ‘Indrek’).

Tühiväärtuse tõttu annab päring (A) aga oodatust erineva tulemuse. Piirangu tingimuses muutujate asendamine relatsioonides esinevate väärtustega annab tulemuseks propositsiooni, kus isiku 101 tegelikku nime, mis andmebaasi seisukohast on tundmatu, asendab tühiväärtus:

(A_2) ¬NULL = ‘Indrek’ ∨ ¬NULL = ‘Mari’

Nagu selgitasin eespool, annab tühiväärtusel tehtav võrdlusoperatsioon tulemuseks tühiväärtuse ning negatsioon tühiväärtusel väärtustub samuti tühiväärtuseks. Nii osutub (A_2) kahe tühiväärtuse disjunktsiooniks, mis omakorda väärtustub tühiväärtuseks. Kuna päring tagastab vaid korteežid, mille kohta piirangu tingimus tõene on, ei tagasta päring (A) mitte midagi.

2.3. Probleem olematute väärtustega

Kuigi Date’i kriitika on suunatud tühiväärtuste vastu üldiselt, on tema argument otseselt rakendatav ainult tundmatute väärtustena tõlgendatud tühiväärtustele. Ainul tundmatute väärtuste puhul on võimalik küsida, milline võiks olla sellele vastav tegelik väärtus reaalsuses. Kui tühiväärtus tähistab olematut väärtust, ei vastagi reaalsuses sellele midagi. Date’i argumendile analoogne argument on esitatav aga ka olematute väärtuste kasutamise vastu. Ka olematute väärtuste tõttu annavad osad päringud oodatust erinevaid tulemusi.

Täiendame eelnevat relatsiooni TUDENG, lisades sellele tunnuse STIPP, mille väärtusteks on tudengitele makstava igakuise stipendiumi suurus. Kui tudengile ei ole stipendiumit määratud, on tunnuse väärtuseks olematu väärtus. Lisame relatsiooni ka ühe täiendava korteeži, et tunnusel STIPP oleks ka tavapäraseid väärtuseid.

TUDENG

NIMI	**STIPP**
‘Indrek’	NULL
‘Kertu’	200

Vaatame järgnevat päringut:

(B) $\sigma_{\neg\exists y\ y = \text{STIPP}}(\text{TUDENG})$

Oodatav päringu tulemus on relatsioon, mis sisaldab kõiki neid relatsiooni TUDENG kuuluvaid korteeže, millel puudub teine liige, ehk mille teiseks liikmeks on olematu väärtus. Seega peaks tulemuseks olema relatsioon, mille ainsaks korteežiks on (‘Indrek’, NULL). Asendades vaba muutuja piirangu tingimuses vastava väärtustega, saame aga järgmise tulemuse:

(B_1) $\neg\exists y\ y = \text{NULL}$

Et (B_1) oleks tõene, peab ‘$\exists y\ y = \text{NULL}$’ olema väär. Olemasolukvantori reeglist lähtuvalt peavad seega omakorda olema väärad kõik propositsioonid kujul ⌜x = NULL⌝. Kuna tühiväärtusel tehtavad võrdlusoperatsioonid on ka ise tühiväärtusega, saab aga kõigi nende propositsioonide väärtuseks olla vaid tühiväärtus. Seega, kuna ükski neist ei ole väär, ei ole ka ‘$\exists y\ y = \text{NULL}$’ väär, mistõttu ei ole propositsioon (B_1) tõene. Järelikult ei tagasta päring (B) korteeži (‘Indrek’, NULL).

Seega, kui andmete sekka Coddi reeglit järgides lubada tühiväärtuseid — ükskõik, kas need on tõlgendatud tundmatute või olematute väärtustena —, ei anna andmebaasi vastu tehtud päringud enam alati oodatud tulemusi. Date võtab ilmenud probleemi kokku järgnevalt:

> Kui su andmebaasis on tühiväärtuseid, saad sa mõnedele päringutele valesid vastuseid. Veelgi enam, sa ei saa loomulikult teada, millistele päringutele sa valesid vastuseid saad ja millistele mitte; kõik tulemused muutuvad kahtlasteks. Sa ei saa kunagi usaldada vastuseid, mida saad tühiväärtustega andmebaasist.[26]

Teisisõnu, kui andmebaasi vastu on *võimalik* teha päringuid, mis annavad tühiväärtuste tõttu valesid tulemusi, muutuvad *kõigi* päringute tulemused küsitavaks. See on aga andmebaasi puhul lubamatu.

[26] Date 2012: 76; I. L. tõlge

3. Date'i lahendus ja selle probleemid

C. J. Date[27] pakub lahenduse puuduvate väärtuste probleemile, mis ei nõua andmebaasi tühiväärtuste lubamist ning väldib seega eelmises peatükis välja toodud probleeme. Selles peatükis annan ülevaate Date'i lahendusest ning toon selles välja kaks probleemi. Date'i lahenduses ilmnevad probleemid näitavad, millega korrektses puuduvate väärtuste probleemi lahenduses arvestama peab. Lahenduse, millel Date'i lahendusega analoogseid probleeme ei ole, esitan seejärel peatükis 4.

3.1. Date'i dekompositsiooniline lahendus

Date'i lahendus puuduvate väärtuste probleemile on tükeldada tühiväärtuseid sisaldav relatsioon erinevate tunnustega relatsioonideks nii, et esialgse relatsiooni iga korteež kuulub sellisesse tükeldamise teel saadud relatsiooni, millel on vaid need tunnused, millele leiduvad korteežis väärtused. Kuna Date'i lahendus ei tohiks sõltuda tühiväärtuste olemasolul, ei ole tühiväärtused esialgses relatsioonis enam käsitletud kui predikaadi muutujate võimalikud väärtused. Peaksime mõtlema neist lihtsalt kui tühjadest kohtadest. Järgnev relatsioon järgib Date'i enda näidet (k. a. selles, et üks tunnustest järgnevas relatsioonis, nimelt ID, on märgitud relatsiooni võtmeks):[28]

TUDENG*

ID	**NIMI**	**STIPP**
123	'Kertu'	200
456	'Indrek'	NULL
789	'Joosep'	100
098	'Peeter'	NULL

[27] vt Date 2012: lisa C
[28] vt Date 2012: 308. Siin esitatud näide erineb Date'i näitest vaid tunnuste ja tavaväärtuste poolest.

Tühiväärtustega andmebaasis oleks selline relatsioon ekstensiooniks predikaadile ⌜Tudengile NIMI matriklinumbriga ID on määratud stipendiumiks STIPP eurot.⌝ Kuna tühiväärtused ei ole enam käsitletud predikaadi muutujate võimalike väärtustena, ei annaks aga enamik korteeže relatsioonis TUDENG* enam selle predikaadi muutujaid väärtustades propositsioone, mistõttu ei ole relatsioon TUDENG* enam käsitletav selle predikaadi ekstensioonina. Date'i meetod sellise relatsiooni tükeldamisel juhindub põhimõttest leida teised predikaadid, mille ekstensioonid mudeldaksid samu tõsiasju, mida mudeldab esialgne tühiväärtuseid sisaldav relatsioon.

Date'i meetodil on kaks peamist sammu, mida Date nimetab *vertikaalseks* ja *horisontaalseks* dekompositsiooniks.[29] Vertikaalne dekompositsioon seisneb esialgse tühiväärtuseid sisaldava relatsiooni põhjal uute relatsioonide moodustamises, millest vähemalt ühel on ainult tavaväärtuseid sisaldavad tunnused, ning ülejäänutel on kõige enam üks tühiväärtuseid sisaldav tunnus.[30] Date'i eeldus, et vertikaalse dekompositsiooni teel on alati võimalik saada vähemalt üks relatsioon, millel on vaid tavaväärtuseid sisaldavad tunnused, on küsitav, ning naasen selle juurde järgmises alapeatükis. Et Date seda eeldab, ilmneb sellest, et ta märgib esialgses näiterelatsioonis — mille järgi siin kasutatud näide koostatud on — ühe tunnuse võtmeks. Kuna relatsiooni võtmes tühiväärtuseid ei esine, on niisugusel juhul Date'i eeldus lubatud.

Relatsioonile leidub üldjuhul mitu võimalikku vertikaalset dekompositsiooni. Relatsiooni TUDENG* üks vertikaalne dekompositsioon annab järgmised relatsioonid:

TUDENG

ID	**NIMI**
123	'Kertu'
456	'Indrek'
789	'Mari'
098	'Peeter'

TS*

ID	**STIPP**
123	200
456	NULL
789	100
098	NULL

[29] vt Date 2012: 308

[30] vt Date 2012: 308. Lahendust esitades, ei too Date eraldi välja, et vähemalt ühel vertikaalse dekompositsiooni teel saadud relatsioonil peavad olema ainult tavaväärtuseid sisaldavad tunnused. Nagu ilmneb allpool, on see aga vajalik, et lõpptulemus vastaks Date'i sõnastatud nõuetele.

Kuna relatsioon TUDENG ei sisalda tühiväärtuseid, saab see olla predikaadi ekstensiooniks. Täpsemalt on TUDENG ekstensiooniks järgnevale predikaadile:

(T) ˹NIMI on tudeng matriklinumbriga ID˺

Date'i meetodiga jätkates, tuleks vertikaalse dekompositsiooni tulemusel saadud relatsioonidel, mis sisaldavad tühiväärtuseid, omakorda läbi viia horisontaalne dekompositsioon. Siin esitatud näites on ainsaks selliseks relatsiooniks TS*.

Kuna relatsioonis TS* esineb tühiväärtuseid, ei kõlba see veel ühegi predikaadi ekstensiooniks. Date täheldab aga, et käsitledes relatsioonis esinevaid korteeže ükshaaval, ning küsides, milliseid propositsioone need korteežid esindavad, on tuvastatavad kahte sorti propositsioonid, sõltuvalt sellest, kas korteežis esineb tühiväärtus. Näiteks vastab vaid tavaväärtuseid sisaldavale korteežile (123, 200) propositsioon 'tudengile matriklinumbriga 123 makstav stipendium on 200 eurot', millest on võimalik abstraheerida järgmine kahekohaline predikaat:

(S_1) ˹tudengile matriklinumbriga ID makstav stipendium on STIPP eurot˺

Kuna tühiväärtused ei ole enam predikaadi muutujate väärtustena lubatud, moodustavad predikaadi (S_1) ekstensiooni täpselt need relatsioonis TS* esinevad korteežid, milles ei esine tühiväärtuseid.

Date tõlgendab tühiväärtuseid oma dekompositsiooni meetodit tutvustades tundmatute väärtustena, mistõttu leiab Date, et tühiväärtust sisaldavale korteežile (456, NULL) vastab propositsioon: 'tudengile matriklinumbriga 456 makstava stipendiumi suurus ei ole teada'. Sellisest propositsioonist on aga võimalik abstraheerida järgmine *ühekohaline* predikaat:

(S_0) ˹tudengile matriklinumbriga ID makstava stipendium ei ole teada˺

Kuna predikaadil (S_0) ei ole TS* tunnusele STIPP vastavat muutujat, võime korteežides, mis esindavad propositsioone kujul (S_0), eirata selle tunnuse väärtuseid (mis kõik on tühiväärtused). Nii osutub predikaadi (S_0) ekstensiooniks relatsioon, milles esinevad korteežid, mille ainsateks liikmeteks on relatsiooni TS* tühiväärtust sisaldavates korteežides tunnusele ID vastavad väärtused.

Date toob eraldi välja,[31] et horisontaalne dekompositsioon võimaldab samaaegselt eristada ka tundmatuid ja olematuid väärtuseid. Et seda näites kajastada, olgu tudengi 456 stipendiumi suurus tundmatu ning olgu tudeng 098 selline, kellele stipendiumi ei ole määratud. Predikaat, mis viimase kohta tõene on, oleks järgmine:

(S_0') ˹tudengile matriklinumbriga ID ei ole stipendiumit määratud˺

Predikaatide (T), (S_1), (S_0) ja (S_0') ekstensioonid on sel juhul vastavalt järgmised relatsioonid:

TUDENG

ID	**NIMI**
123	'Kertu'
456	'Indrek'
789	'Mari'
098	'Peeter'

TS1

ID	**STIPP**
123	200
789	100

TS0

ID
456

TS0'

ID
098

Sellega on kõik esialgses relatsioonis esinevad tühiväärtused andmebaasist eemaldatud. Selline seis andmebaasis peab aga olema moodustatav ja uute andmete lisamisel sälitatav ka ilma Date'i kahe dekompositsiooni meetodita. Kui andmebaasis ei ole tühiväärtused lubatud, ei esine selles ka esialgset tühiväärtustega relatsiooni, millel Date'i meetodit rakendada. Selleks sõnastab Date nõuded, millele andmebaas tema lahenduse jaoks alati vastama peab.

Esiteks peab andmebaasis leiduma vähemalt üks relatsioon R, kuhu on võimalik andmeid otse lisada, ning mille tunnustel väärtused puududa ei saa.[32] Lisaks võib relatsioonil R olla ka nii-öelda *pseudotunnuseid*, mis ei ole relatsiooni R enda osad, vaid moodustuvad mitmest eraldi relatsioonist ning on relatsiooniga R seotud seeläbi, et jagavad temaga sama võtit.[33]

[31] vt Date 2012: 311–312
[32] Seda piirangut Date (2012: 313) ise ei esita, kuid teised Date'i esitatud piirangud eeldavad seda.
[33] Pseudotunnuse mõistet Date ei kasuta, kuid see lihtsustab tema lahenduse mõistmist ja kirjeldamist.

Korteežid ühes neist tähistavad pseudotunnuse P olemasolevaid väärtuseid ning korteezid ülejäänutes P väärtuste puudumist, kas seetõttu, et vastav väärtus on tundmatu, või seetõttu, et vastavat objekti reaalsuses ei leidu. Võime neid relatsioone tähistada vastavalt kui RP1, RP0 ja RP0'.[34] Kuna relatsiooni R võtme iga väärtuse kohta peab pseudotunnusel P kas väärtus olemas olema või puuduma, peab igale relatsiooni R võtme väärtusele vastama täpselt üks korteež kas relatsioonis RP1, relatsioonis RP0 või relatsioonis RP0'. Siin esitatud näites on relatsiooniks R relatsioon TUDENG, mille ainsaks pseudotunnuseks on STIPP, mille moodustab relatsioonide kolmik TS1, TS0 ja TS0'.

Date'i dekompositsioonilises lahenduses puuduvate väärtuste probleemile on tuvastavad aga vähemalt kaks puudust. Esiteks ei ole Date'i lahendus universaalne. Leidub tühiväärtustega relatsioone, millest Date'i kahe dekompositsiooni meetodil ei ole võimalik tühiväärtuseid eemaldada. Teiseks, horisontaalse dekompositsiooni teel saadavad relatsioonid Date'i lahenduses ei ole omavahel loogiliselt seotud. Järgnevalt selgitan kumbagi probleemi lähemalt.

3.2. Probleem võtmeta relatsioonidega

Peamiseks probleemiks Date'i lahendusel on, et andmebaasis, kus osad väärtused võivad puududa, ei ole garanteeritud, et andmetes on alati võimalik tuvastada tunnuseid, mis moodustavad võtme. Vertikaalse dekompositsiooni samm Date'i lahenduses aga eeldab võtme olemasolu.

Nagu selgitasin peatükis 2.1, ei ole korteežid tühiväärtuste alusel tuvastavad, kuna selleks vajalikud võrdlusoperatsioonid annaksid alati tulemuseks tühiväärtuse. Seetõttu ei saa tühiväärtus olla võtmes esineva tunnuse väärtuseks. Relatsioonil, kus tühiväärtus võib esineda ükskõik millise tunnuse väärtusena, seega võti puudub. Tühiväärtustega relatsiooni vertikaalne dekompositsioon on aga võimalik vaid seetõttu, et sel teel relatsioonist eraldatud tunnused on hiljem võtme alusel omavahel taas kokku viidavad, mistõttu vertikaalne dekompositsioon on võimalik vaid võtmega relatsioonidel. Date'i kahe dekompositsiooni meetod ei ole seega puuduvate väärtuste probleemile universaalne lahendus, mis võimaldaks tühiväärtuseid eemaldada igast võimalikust relatsioonist.

[34] Date (2012: 312) annab ise oma lahenduse üldise sõnastuse viisil, mis jätab lahtiseks, kui palju erinevaid tõlgendusi tühiväärtustele on võimalik anda. Kui tühiväärtuse liike on *n*, annab horisontaalne dekompositstioon, nii nagu Date seda esitab, *n* +1 relatsiooni.

Näide relatsioonist, millel Date'i meetodit ei ole võimalik rakendada, oleks näiteks järgmine relatsioon, mis sisaldab lauateenindaja kogutud andmeid ühes lauas tehtud tellimuste kohta:[35]

TELLIMUS*

KOHT	**EELROOG**	**PÕHIROOG**	**MAGUSTOIT**	**JOOK**
1	NULL	'Beyond smäsh'	'rummipall'	'marja smuuti'
2	'lillkapsatiivad'	'laetud friikad'	NULL	NULL
3	NULL	'Caesari salat'	'kirju koer'	'jääkohv'
2	'lillkapsatiivad'	NULL	NULL	NULL

Iga toidukäigu puhul on võimalik, et klient jätab selle käigu roa tellimata. Relatsiooni ainus tunnus, milles tühiväärtus ilmselt esineda ei saa, on KOHT, mis tähistab tellija istekohta lauas. Kuna aga sama klient — siin näites klient istekohal 2 — võib teha külastuse käigus ka mitu tellimust, ei ole tunnuse KOHT puhul garanteeritud väärtuste unikaalsus. Seega ei ole relatsioonil TELLIMUS* võtit ning Date'i vertikaalne dekompositsioon ei ole relatsioonil võimalik. Relatsioon TELLIMUS* ei ole aga isegi kunstlik näide, mida praktikas võiks lihtsalt eirata. Toidukohas võiksid lauateenindajad ka reaalselt sel viisil tellimuste andmeid andmebaasi kirjutada.

3.3. Probleem horisontaalse dekompositsiooniga

Teine probleem Date'i kahe dekompositsiooniga lahenduses puudutab horisontaalset dekompositsiooni. *Vertikaalse* dekompositsiooni tulemusel saadud relatsioonid jäävad teineteisega horisontaalselt seotuks seeläbi, et jagavad kõik sama võtit. Korteežid, milles on samad väärtused samale võtmele, moodustavad justkui ühe pika korteeži, mille erinevad osad kuuluvad erinevatesse relatsioonidesse. Relatsioonid, milles on olemas võti, on seetõttu

[35] Näide on koostatud kiirtoidukoha Veg Machine menüü põhjal leheküljel https://vegmachine.ee/ (külastatud 06.08.2024).

vertikaalselt tükeldatavad, ilma et seejuures informatsiooni kaduma läheks. Alati on võimalik esialgse relatsiooni korteežid võtme kaudu taastada.

Et relatsioon oleks informatsiooni kaotamata ka horisontaalselt tükeldatav, on vaja, et tulemuseks saadud osad jääksid teineteisega vertikaalselt seotuks. Relatsioonilises mudelis on erinevad relatsioonid vertikaalselt seotud vaid siis, kui nad kõik on täpselt samade tunnustega. Seetõttu on näiteks relatsioonalgebras relatsioonide vahe operatsioon — mis relatsioonalgebra operatsioonidest ainsana tükeldab relatsioone horisontaalselt — võimalik vaid samade tunnustega relatsioonidel. Date'i kirjeldatud horisontaalne dekompositsioon aga annab tulemuseks relatsioonid, mis ei jaga kõiki tunnuseid. Seega on võimalik, et horisontaalse dekompositsiooni käigus läheb informatsiooni kaduma.

Eelnevalt tutvustasin lühidalt Date'i viisi seda probleemi lahendada; nimelt sõnastab Date piirangu, millele horisontaalse dekompositsiooni teel saadud relatsioonid andmebaasis alati vastama peavad. Võtme väärtused, mis esinevad ühes neist, ei tohi Date'i piirangu kohaselt samal ajal esineda võtme väärtustena teistes. Selline täiendav piirang on aga üldse vajalik vaid seetõttu, et relatsioonidest *endist* ei ole võimalik seda tuletada. Seda, et Date'i piirang ei sobi loomulikul viisil relatsioonilisse mudelisse, ilmestab tõsiasi, et relatsioonide *nimed* peavad selleks täitma andmebaasis uut funktsiooni. Date'i piirangud on vaja sõnastada eraldi iga horisontaalse dekompositsiooni teel saadud relatsioonide komplekti jaoks ning iga kord on vaja igale relatsioonile komplektis eraldi viidata. Peatükis 3.1 esitatud näites on vaja öelda näiteks, et relatsioonid TS1, TS0 ja TS0' ei jaga ühtegi võtme väärtust. Seega võimaldab Date'i viis horisontaalse dekompositsiooni teel saadud relatsioone horisontaalselt ühendada vaid nimede alusel. Mitte miski muu relatsioonilises mudelis aga ei sõltu sel viisil relatsioonide nimedest.

Probleem horisontaalse dekompositsiooni teel saadud relatsioonide seotusega on samas ületatav, kui loobume eristusest eri liiki tühiväärtuste vahel. Kui ei ole eristust tundmatute ja olematute väärtuste vahel, võime horisontaalse dekompositsiooni käigus formuleerida vaid ühe predikaadi, nimelt predikaadi, mille ekstensiooni kuuluvad tükeldatavast relatsioonist need korteežid, mis sisaldavad ainult tavaväärtuseid. Nii annab horisontaalne dekompositsioon tulemuseks vaid ühe relatsiooni, mis sisaldab pseudotunnuse olemasolevaid väärtuseid. Kõigil võtme väärtustel, mis sellesse relatsiooni ei kuulu, on pseudotunnusel väärtus puudu. Kuna horisontaalne dekompositsioon annab nüüd tulemuseks vaid ühe relatsiooni, väldime ka vajadust relatsioonide vaheliste seoste järele. Võtme väärtused, millele ühtki pseudotunnuse

väärtust ei vasta, on sellise lahenduse puhul relatsioonalgebras päritavad, mistõttu ei ole ka vajadust neid eraldi relatsioonides hoida (nagu seda teeb Date). Sobivaks relatsioonalgebra päringuks on $\pi_{T_1,\ldots,T_n}(R) - \pi_{T_1,\ldots,T_n}(RP)$, kus R on vertikaalse dekompositsiooni teel saadud tavaväärtustega relatsioon, RP pseudotunnuse P jaoks horisontaalse dekompositsiooni teel saadud relatsioon ning tunnused $T_1, \ldots, T_n$ moodustavad relatsiooni R — ning seega relatsiooni RP — võtme.

Kuigi horisontaalse dekompositsiooni teel saadud relatsioonide seotuse probleem Date'i lahenduses on ületatav, on see nii vaid eeldusel, et tühiväärtustega relatsioonil on olemas võti. Järgnevas peatükis esitan meetodi tühiväärtuste eemaldamiseks, mis vajab vaid horisontaalset dekompositsiooni ning võimaldab tühiväärtuste eemaldamist ka võtmeta relatsioonidest. Horisontaalne dekompositsioon võtmeta relatsioonidel toob aga just ületatud probleemi tagasi uuel kujul, mis nõuab lahenduseks teistsugust lähenemist.

4. Ramsey-Wittgensteini lahendus

Date'i lahendus puuduvate väärtuste probleemile eeldab, et andmete seas on alati tunnuseid, mis moodustavad võtme, ning millel seega väärtus puududa ei saa. Nagu selgitasin peatükis 3.2, ei ole see aga reaalsuses alati nii, mistõttu Date'i lahendus ei võimalda alati tõsiasju tõepäraselt mudeldada. Siin peatükis esitan lahenduse puuduvate väärtuste probleemile, millel sellist piirangut ei ole, tuginedes Frank Ramsey ja Ludwig Wittgensteini ideedele loogikafilosoofias.

4.1. Võtmeta relatsiooni dekompositsioon

Adekvaatne lahendus puuduvate väärtuste probleemile peab suutma eemaldada tühiväärtuseid võtmeta relatsioonidest. Peatükis 3.2 esitatud näide sellisest relatsioonist oli TELLIMUS*:

TELLIMUS*

KOHT	**EELROOG**	**PÕHIROOG**	**MAGUSTOIT**	**JOOK**
1	NULL	'Beyond smäsh'	'rummipall'	'marja smuuti'
2	'lillkapsatiivad'	'laetud friikad'	NULL	NULL
3	NULL	'Caesari salat'	'kirju koer'	'jääkohv'
2	'lillkapsatiivad'	NULL	NULL	NULL

Relatsioonil TELLIMUS* puudub võti, kuna kõik selle tunnused peale tunnuse KOHT võimaldavad tühiväärtuseid ning tunnuse KOHT võimalikud väärtused ei ole unikaalsed. Vertikaalne dekompositsioon, mida Date'i lahendus eeldab, ei ole seega relatsioonil TELLIMUS* võimalik. Kui püüda relatsioonil võtme puudumisest hoolimata vertikaalset dekompositsiooni läbi viia, kasutades võtme asemel tunnust KOHT, oleks ühel sel viisil saadud relatsioonil ainsateks tunnusteks KOHT ja EELROOG, mis esialgset relatsiooni arvestades võtaksid kahel juhul täpselt samu väärtuseid. Kuna relatsioonides ei esine korduvaid korteeže, kaotaksime sellega informatsiooni, et kohal 2 istuv klient tellis lillkapsatiibu kaks korda.

Põhimõtteliselt on aga horisontaalne dekompositsioon võimalik ka ilma eelneva vertikaalse dekompositsioonita. Relatsiooni TELLIMUS* saab horisontaalselt tükeldada kolmeks tühiväärtusteta osaks järgmiste predikaatide abil:

(T_1) ˹koha KOHT tellimuseks on PÕHIROOG, MAGUSTOIT ja JOOK˺

(T_2) ˹koha KOHT tellimuseks on EELROOG ja PÕHIROOG˺

(T_3) ˹koha KOHT tellimuseks on EELROOG˺

Nende predikaatide ekstensioonid on vastavalt järgmised:

T1

KOHT	**PÕHIROOG**	**MAGUSTOIT**	**JOOK**
1	‘Beyond smäsh’	‘rummipall’	‘marja smuuti’
3	‘Caesari salat’	‘kirju koer’	‘jääkohv’

T2

KOHT	**EELROOG**	**PÕHIROOG**
2	‘lillkapsatiivad’	‘laetud friikad’

T3

KOHT	**EELROOG**
2	‘lillkapsatiivad’

Relatsioonides T1, T2 ja T3 ei esine enam tühiväärtuseid, kuid sel viisil relatsiooni tükeldamine toob kaasa probleemi, mida käsitlesin peatükis 3.3. Kuna neil kõigil on erinevad tunnused, ei ole enam midagi, mis neid relatsioone ühendaks. Kuna esialgsel relatsioonil TELLIMUS* ei olnud võimalik eelnevalt teha vertikaalset dekompositsiooni, ei ole nüüd võimalik probleemi ületada ka peatükis 3.3 kirjeldatud viisil. Et informatsiooni kaotamata eemaldada kaht kolmest saadud relatsioonist, peaksid leiduma päringud, mille tulemusteks eemaldatud relatsioonid oleksid. Ilma esialgse relatsiooni võtme väärtuseid sisaldava relatsioonita, mille annab vertikaalne dekompositsioon, ei ole aga selliseid päringuid millelgi teha.

Enne probleemi lähemat käsitlemist võib aga tekkida küsimus seoses predikaatide (T_1), (T_2) ja (T_3) sõnastusega. Nende sõnastus erineb sellest, mida kasutas Date horisontaalses dekompositsioonis. Lühiduse huvides eiran hetkeks relatsiooni T1. Date'i-stiilis sõnastus predikaadile, mille ekstensiooniks peaks olema relatsioon T3, oleks sel juhul järgmine:

(T_3') ˹ei ole nii, et leidub selline x, et koha KOHT tellimuseks oli EELROOG ja x˺

Sellel predikaadil on kaks vaba muutujat, 'KOHT' ja 'EELROOG', mille põhjal võiks esmapilgul arvata, et tema ekstensiooniks, nagu ka predikaadi (T_3) ekstensiooniks, on relatsioon T3. Asendades predikaadis (T_3') aga vabad muutujad relatsioonis T3 esineva korteeži väärtustega, saame tulemuseks *väära* propositsiooni 'ei ole nii, et leidub selline x, et koha 2 tellimuseks oli lillkapsatiivad ja x'. Propositsioon on väär, kuna koha 2 tellimuseks oli ka lillkapsatiivad ja laetud friikad, nagu on näha relatsioonist T2. Seega ei ole relatsioon T3 predikaadi (T_3') ekstensiooniks.

Date sai horisontaalse dekompositsiooni jaoks kasutada eitavat predikaati vaid seetõttu, et võtmega relatsioonis on kõik korteežid teineteisest võtme väärtuste põhjal eristuvad. See välistab olukorra, kus kaks korteeži eristuvad teineteisest vaid sellepoolest, et ühel neist on väärtus positsioonil, kus teisel väärtus puudub. Horisontaalset dekompositsiooni puuduvate väärtuste probleemi *universaalses* lahenduses ei saa seega teha loogiliselt komplekssete predikaatide abil nagu seda tegi Date. See ilmestab veel enam seda, et nende predikaatide ekstensioonid — s.t. horisontaalse dekompositsiooni teel saadud relatsioonid — ei paista olevat kuidagi omavahel seotud.

Miks aga arvata, et horisontaalse dekompositsiooni teel saadud relatsioonid üldse *peavad* olema andmebaasis omavahel seotud? Miks ei võiks süsteem andmete otsimisel relatsioone T1, T2 ja T3 näiteks lihtsalt ükshaaval läbi vaadata, eeldamata, et nad kuidagi erilisel viisil kokku kuuluvad? Kui T1, T2 ja T3 oleksid ainsad relatsioonid andmebaasis, oleks see võimalik. Kuna andmebaasis võib aga olla lisaks veel teisigi relatsioone, on vaja andmebaasi juhtimissüsteemil teada, millistes relatsioonides esinevad andmed on päringu tulemuse jaoks olulised ja millised mitte. Näiteks võiks andmebaasis olla lisaks ka relatsioon A2, millel on samad tunnused relatsiooniga T2, kuid on ekstensiooniks teisele predikaadile:

(A_2) ˹koht KOHT jättis arvustuse roogadele EELROOG ja PÕHIROOG˺

Et sellisest andmebaasist oleks võimalik pärida andmeid, mis käivad tellimuste ja mitte arvustuste kohta, peavad relatsioonid T1 ja T3 olema relatsiooniga T2 seotud viisil, mil nad relatsiooniga A2 seotud ei ole.

Loomulikult võiks relatsioone T1, T2 ja T3 ka lihtsalt igas tellimuste kohta käivate andmete päringus eraldi mainida. Niisuguse lahenduse probleem on aga selles, et relatsioon TELLIMUS* oli *juhtumisi* selline, et horisontaalne dekompositsioon sellel annab relatsioonid T1, T2 ja T3. Relatsiooni TELLIMUS* asemel võib aga olla relatsioon, kus tühiväärtused asuvad teistes kohtades ning kus võib olla ka täiendavaid tunnuseid. Näiteks võib mõni klient tellida ühe joogi asemel korraga kaks, mis tähendaks, et relatsioonil oleks ka tunnus TEINE_JOOK. See aga tähendab, et andmebaasi enne iga päringut läbi vaatamata ei ole võimalik teada, milliseid relatsioone on päringus vaja mainida, et päringu tulemus põhineks kõigi tellimuste kohta käivatel andmetel.

4.2. Ramsey-Wittgensteini baaspropositsioonide käsitlus

Selgitasin eelnevas alapeatükis, kuidas ilma vertikaalse dekompositsioonita võimaldab tühiväärtustega relatsiooni horisontaalne dekompositsioon küll kaotada tühiväärtused, kuid annab tulemuseks relatsioonid, mis ei ole pealtnäha omavahel seotud. Tegemist on lihtsalt sarnaselt nimetatud relatsioonidega, millel on erinevad tunnused. Seos nende relatsioonide vahel on aga oluline, kuna andmed neis relatsioonides kuuluvad reaalsuses kokku. Probleemi lahendamiseks on vaja leida relatsioonilises mudelis vahendid selle kokku kuuluvuse väljendamiseks.

Nagu selgitasin peatükis 1, on relatsiooniline mudel mõistetav predikaatarvutuse kaudu. Probleem horisontaalse dekompositsiooni teel saadud relatsioonide seotusega ei ole seetõttu probleem üksnes relatsioonilisele mudelile vaid ka predikaatarvutusele. Näiteks on propositsioonide ‘Mari ja Jüri kohtusid’ ning ‘Mari, Jüri ja Liisa kohtusid’ loomulikud formaliseeringud predikaatarvutuses vastavalt ‘K(a, b)’ ning ‘K(a, b, c)’, millest on abstraheeritavad vastavalt predikaadid ˹K(x, y)˺ ja ˹K(x, y, z)˺. Ühtlasi on ilmne, et sel viisil on võimalik moodustada ükskõik mitme kohalisi predikaate, mis kõik intuitiivselt väljendavad ühte ja sedasama, nimelt et mingid isikud kohtusid. Loogikafilosoofias nimetatakse sellisel viisil predikaatide moodustamist võimaldavaid seoseid mitmeastmelisteks (ingl *multigrade*)

seosteks[36] ning nende lubamine predikaatarvutusse toob kaasa samasuguse probleemi, mis kaasneb võtmeta relatsioonide horisontaalse dekompositsiooniga. Predikaatidel ˹K(x, y)˺ ja ˹K(x, y, z)˺ on predikaatarvutuses erinevad ekstensioonid. Neid ühendab vaid ühine sümbol 'K', kuid kaasaegses predikaatarvutuses sellele sümbolile — käsitletuna lahus termi kohtadest — väärtust eraldi ei määrata.

Küsimus, kuidas käsitleda relatsioonilises mudelis puuduvaid väärtuseid, on niisiis ühtlasi küsimus, kuidas käsitleda predikaatarvutuses mitmeastmelisi seoseid. See, et puuduvate väärtuste probleemi lahendus tuleb lõppkokkuvõttes loogikast, on oluline, kuna toetab Coddi esialgset nägemust relatsioonilisest mudelist kui puhtal loogikal baseeruvast andmebaaside käsitlusest.

Lahendus sellele, kuidas predikaatarvutuses käsitleda mitmeastmelisi seoseid, pärineb Frank Ramsey'lt,[37] kes omakorda tugineb Ludwig Wittgensteini ideedele.[38] Täpsemalt tuleneb lahendus Ramsey ja Wittgensteini baaspropositsioonide käsitlusest. Baaspropositsioonid loogikas on propositsioonid, mis ei ole moodustatud ühegi loogilise operatsiooni teel, ning mis võimaldavad seega tsirkulaarsuseta defineerida kõik ülejäänud propositsioonid loogiliste operatsioonide kaudu baaspropositsioonidel. Baaspropositsioonide käsitlus peaks ütlema, millistest osadest baaspropositsioonid koosnevad, mis on nende osade väärtusteks, ning kuidas on nende väärtuste kaudu määratletud baaspropositsioonide tõetingimused — s.t. see, millistel tingimustel baaspropositsioonid on tõesed.

Kaasaegsetest loogikaõpikutest leitava baaspropositsioonide käsitluse kohaselt kooseb baaspropositsioon ühest predikaadist ning mingist arvust termidest. Termide väärtusteks on objektid, millele need termid osutavad. Et tutvustada õpikutes kasutusel olevat predikaadimõistet, pean segaduste vältimiseks aga tegema kõigepealt ühe selgitava märkuse. Seda, mida siin töös olen nimetanud predikaadiks — ning mida nimetab nii ka Date — on see, mida loogikaõpikutes nimetatakse vabu muutujaid sisaldavaks valemiks. Valem on väljend, mis on just nagu lause, kuid võib sisaldada vabu muutujaid, mille väärtustamisel saab valemist miski, millel on tõeväärtus, ehk propositsioon. Predikaadid seevastu, nii nagu seda terminit kasutatakse loogikaõpikutes, on erilised baaspropositsioonide osad, mille väärtusteks on aga samuti ekstensioonid või — mis teeb sama välja — funktsioonid objektide korteežidest

[36] Termini võtsid esimesena kasutusele Leonard ja Goodman (1940), kellelt pärineb ka näide kohtumisest kui mitmeastmelisest seosest (vt Leonard, Goodman 1940: 50).
[37] vt Ramsey 1925
[38] Ramsey tugineb peamiselt Wittgensteini *Loogilis-filosoofilisele traktaadile* (1925).

tõeväärtustesse. Nimetan predikaate selles tähenduses edaspidi õpikupredikaatideks. Õpikutes esitatud käsitluse kohaselt on baaspropositsioon ˹R(a_1, …, a_n)˺ tõene parajasti siis, kui õpikupredikaadi R väärtuseks olev funktsioon annab tulemuseks tõe korteeži ([a_1], …, [a_n]) korral, kus [a_i] on objekt, millele osutab term a_i.

Loogikaõpikutes esitatud käsitluse probleemiks on, et õpikupredikaadi väärtuseks olevad funktsioonid on mõistetud osaliselt tõeväärtuste kaudu. Tõeväärtused on alati mingi *propositsiooni* väärtused, kuna kajastavad seda, kas asjad reaalsuses on nii nagu neid öeldakse olevat. Seega, ka õpikupredikaadi väärtuses esinevad tõeväärtused peavad olema mingite propositsioonide tõeväärtused. Nendeks propositsioonideks õpikukäsitlusest lähtuvalt on baaspropositsioonide tõeväärtused, mille osana õpikupredikaat esineb. Propositsiooni tõetingimus peab olema määratud selle kaudu, millised on propositsiooni osade väärtused ning kuidas need osad propositsiooni moodustavad.[39] Seega, et propositsioonil oleks tõeväärtus, peavad tema osadel juba väärtused olemas olema. Et õpikukäsitluse kohaselt saaks baaspropositsioonil olla tõeväärtus, peab seega ka temas esinevale õpikupredikaadile olema väärtus juba määratud. Samas on aga õpikupredikaadi väärtus mõistetud teda sisaldavate baaspropositsioonide tõeväärtuste kaudu. Predikaadi mõiste loogikaõpikutes sisaldab niisiis tsirkulaarsust.[40] Ramsey-Wittgensteini käsitlus väldib seda probleemi kuna ei postuleeri baaspropositsiooni osana õpikupredikaati.

Baaspropositsiooni tõetingimus on Wittgensteini sõnul mõistetav lihtsalt kui "objektide (esemete, asjade) ühendatus".[41] Isiklikes märkmetes lisab Wittgenstein selgituseks, et "[k]a seosed ja omadused jne on *objektid*".[42] Sellest lähtuvalt peab baaspropositsioon ise koosnema ainult objektidele osutavatest termidest.[43] Erinevalt õpikukäsitlusest on Wittgensteini sõnul baaspropositsiooni osad niisiis kõik *sama tüüpi* väljendid. Sellise käsitluse kohaselt on baaspropositsioon tõene parajasti siis, kui objektid, millele propositsioonis esinevad termid osutavad, on reaalsuses seotud samamoodi nagu neile vastavad termid propositsioonis.[44] Näiteks on baaspropositsioon 'Sokrates on tark' selle kohaselt tõene parajasti siis, kui Sokrates

[39] Loogikas ja keeleteaduses nimetatakse seda ka kompositsionaalsuse printsiibiks. Printsiibi kaitsmine jääb siinse töö teema alt välja, kuid ülevaate selle kohta annab näiteks Szabó (2022).
[40] Siin esitatud argument põhineb Dummetti (1973: 30–31) argumendil.
[41] Wittgenstein 1925: §2.01; I. L. tõlge. Baaspropositsioonide tõetingimusi nimetab Wittgenstein asjaoludeks (saks *Sachverhalt*).
[42] Wittgenstein 1961: 61, I. L. tõlge
[43] vt Wittgenstein 1925: §4.22
[44] vt Wittgenstein 1925: §4.0311

on tark. Term ‘Sokrates’ osutab siin tavapärasel viisil Sokratesele, kuid Wittgensteini sõnul on termiks ka ‘tark’, mis osutab *tarkusele*.

Et näha, kuidas Wittgensteini baaslausete käsitlus võimaldab ühendada mitmeastmelisi seoseid väljendavaid predikaate, on vaja ka selgitada, kuidas Ramsey paneb kokku Wittgensteini idee predikaatide käsitlusega, mida tutvustasin peatükis 1.1 ning mida eeldab ka Date. Meenutuseks, predikaat ˹x on tark˺ ei ole sellise käsitluse kohaselt mitte propositsiooni ‘Sokrates on tark’ *osa*, nagu seda eeldatakse loogikaõpikutes baaspropositsioonide puhul, vaid selle *vorm*. Propositsioonide vormina mõistetuna võimaldab predikaat ˹x on tark˺, nagu Ramsey seda sõnastab, “koguda kokku” sama vormi jagavad propositsioonid nagu ‘Sokrates on tark’, ‘Platon on tark’, jne.[45]

Ramsey täheldab, et kui baaspropositsioonid koosnevad vaid objektidele osutavatest termidest, on propositsioonist ‘Sokrates on tark’ võimalik abstraheerida rohkem kui üks predikaat. Lisaks predikaadile ˹x on tark˺, mida ka loogikaõpikutest esitatud käsitlus võimaldab, on võimalik abstraheerida ka predikaat ˹Sokrates on q˺, mis on ühiseks vormiks baaspropositsioonidele nagu ‘Sokrates on tark’, ‘Sokrates on õiglane’, jne.[46] Väärtuseid, mida muutuja ‘q’ saab võtta, nimetab Ramsey kvaliteetideks,[47] et eristada neid loogiliste operatsioonide kaudu defineeritavatest omadustest nagu omadusest olla tark kui Platon on rumal. Loogika seisukohast on kvaliteedid objektid täpselt samamoodi nagu on objekt ka Sokrates. Muutujatega võib propositsioonis asendada ka mõlemad termid *korraga*, mis annab tulemuseks predikaadi ˹x on q˺. Seda vormi jagavate propositsioonide hulk on lihtsalt eelneva kahe propositsioonide hulga ühend. Predikaadi ˹x on q˺ ekstensioon koosneb kaheelemendilistest korteežidest, mille esimene liige on isik ja teine teda iseloomustav kvaliteet.

Ramsey tähelepanek võimaldab näha, kuidas on ühendatud mitmeastmelisi seoseid väljendavate predikaatide ekstensioonid. Propositsioonides ‘Mari ja Jüri kohtusid’ ning ‘Mari, Jüri ja Liisa kohtusid’ on abstraheeritavad vastavalt predikaadid ˹x ja y kohtusid˺ ja ˹x, y ja z kohtusid˺. Ramsey-Wittgensteini käsitlusest lähtuvalt on mõlemas predikaadis aga veel üks term, mida on võimalik muutujaga asendada. See annab tulemuseks vastavalt predikaadid ˹x ja y q-sid˺ ning ˹x, y ja z q-sid˺, kus muutuja ‘q’ väärtusteks on mitmeastmelised seosed. Ekstensioonideks on ühel neist predikaatidest kolme- ja teisel neljaelemendilised korteežid,

[45] Ramsey 1925: 410; I. L. tõlge
[46] vt Ramsey 1925: 411
[47] vt Ramsey 1925: 411

kus üks elementidest on mitmeastmeline seos — tülitsemine, vestlemine, jne. — ning ülejäänud on isikud, kes teineteisega selles seoses on. Nende korteežide seas on loomulikult ka sellised, kus mitmeastmeliseks seoseks on kohtumine. Predikaatide ˹x ja y q-sid˺ ja ˹x, y ja z q-sid˺ ekstensioonide osahulgad, mis sellistest korteežidest koosnevad, ongi käsitletavad predikaatide ˹x ja y kohtusid˺ ja ˹x, y ja z kohtusid˺ ekstensioonidena.

Kuigi predikaat ˹x ja y kohtusid˺ on kahe- ja predikaat ˹x, y ja z kohtusid˺ kolmekohaline, võime nende ekstensioonidena siiski käsitleda vastavalt kolme- ja neljakohaliste korteežide hulkasid. Harilikult konstrueeritakse *n*-kohalise predikaadi ekstensioon küll *n*-kohaliste korteežide hulgana, kuid seda vaid seetõttu, et predikaadi peamine otstarve — nagu selgitasin peatükis 1.1 — on aidata defineerida kvantoritega propositsioonide tõetingimusi, kus on oluline vaid see, milliseid väärtuseid võivad võtta predikaadi muutujad. Täiendav objekt predikaatide ˹x ja y kohtusid˺ ja ˹x, y ja z kohtusid˺ ekstensioonidesse kuuluvates korteežides ei vasta mitte muutujale vaid konstantsele termile ‘kohtusid’. Kuna term ‘kohtusid’ esineb mõlemas predikaadis, kus ta osutab samale objektile, jagavad mõlema predikaadi ekstensioonid konstantse väärtusega tunnust ning on seeläbi omavahel ühendatud, hoolimata sellest, et nad ei jaga kõiki tunnuseid.

4.3. Ramsey-Wittgensteini käsitluse põhine lahendus

Lahendus mitmeastmeliste seoste probleemile loogikas on ülekantav ka relatsioonilisse mudelisse. Näitan, kuidas seeläbi saab lahendatud esiteks probleem horisontaalse dekompositsiooniga, mida selgitasin alapeatükis 4.1, ning lõpuks ka puuduvate väärtuste probleem ise.

Tulles tagasi alapeatükis 4.1 esitatud näite juurde, relatsiooni TELLIMUS* horisontaalse dekompositsiooni jaoks sobivad predikaadid olid järgmised:

(T_1) ˹koha KOHT tellimuseks on PÕHIROOG, MAGUSTOIT ja JOOK˺

(T_2) ˹koha KOHT tellimuseks on EELROOG ja PÕHIROOG˺

(T_3) ˹koha KOHT tellimuseks on EELROOG˺

Lähtudes eelmises alapeatükis tutvustatud Ramsey-Wittgensteini käsitlusest, on kõigis kolmes predikaadis tuvastatav sama konstantne term ‘tellimus’, mille väärtuseks olev objekt — omadus olla tellimus — käitub nagu mitmeastmeline seos. Ka term ‘tellimus’ on asendatav muutujaga, mistõttu on predikaatide (T_1), (T_2) ja (T_3) ekstensioonides võimalik tuvastatada ka

sellele muutujale vastav tunnus. Olgu selleks tunnuseks TEGEVUS. Erinevalt teistest tunnustest on sel aga läbivalt sama väärtus. Tuues välja ka selle täiendava tunnuse, on predikaatide (T_1), (T_2) ja (T_3) ekstensioonid vastavalt järgmised:

T1

KOHT	**TEGEVUS**	**PÕHIROOG**	**MAGUSTOIT**	**JOOK**
1	‘tellimus’	‘Beyond smäsh’	‘rummipall’	‘marja smuuti’
3	‘tellimus’	‘Caesari salat’	‘kirju koer’	‘jääkohv’

T2

KOHT	**TEGEVUS**	**EELROOG**	**PÕHIROOG**
2	‘tellimus’	‘lillkapsatiivad’	‘laetud friikad’

T3

KOHT	**TEGEVUS**	**EELROOG**
2	‘tellimus’	‘lillkapsatiivad’

Kuigi relatsioonid T1, T2 ja T3 ei jaga kõiki tunnuseid, on nad sellegipoolest omavahel kokku viidavad. Nende ühendamiseks piisab sellest, et neis kõigis on olemas tunnus TEGEVUS, mis kõigis kolmes relatsioonis võtab läbivalt sama väärtuse. Andmebaasist kõigi tellimuste kohta käivate andmete pärimiseks on vaja vaid leida relatsioonid, millel on olemas tunnus TEGEVUS, ning neist relatsioonidest on vaja välja valida korteežid, kus selle tunnuse väärtuseks on ‘tellimus’. Vaid peatükis 1.3 tutvustatud relatsioonalgebra operatsioonidega ei ole selline päring koostatav. Kuidas niisugust päringut relatsioonilises mudelis siiski moodustada, selgitan allpool. Kõigepealt selgitan aga, kuidas sellise täiendava tunnuse abil nagu TEGEVUS laheneb puuduvate väärtuste probleem.

Kuigi horisontaalne dekompositsioon tühiväärtustega relatsioonil võimaldab tühiväärtustest vabaneda ning Ramsey ja Wittgensteini idee täiendavatest termidest predikaatides võimaldab sel teel saadud relatsioonid omavahel taas kokku viia, eeldab selline meetod tühiväärtustega relatsiooni olemasolu. Nagu Date’i lahendus peatükis 3.1 peab ka

Ramsey-Wittgensteini lahendus lõpuks näitama, kuidas puuduvate väärtustega andmebaasi hallata, ilma et see vajaks pidevat tühiväärtustega relatsioonide dekompositsiooni. Et säilitada andmebaasis seisu, kus erinevate tunnustega relatsioonid on ühe tunnuse alusel kokku viidavad, on andmete andmebaasi lisamisel vaja lisatavates andmetes alati ära märkida ka siduva tunnuse väärtus. Siin kasutatud näites tähendab see, et tellimuste kohta käivate andmete lisamisel andmebaasi on lisaks tellitud toitudele ja tellija istekohale vaja alati märkida ka tunnuse TEGEVUS väärtuseks ‘tellimus’.

Eelnev toob Ramsey-Wittgensteini lahenduse põhjal organiseeritud andmebaasi puhul välja midagi, mis võib esmapilgul tunduda probleemina. Andmete lisamiseks sellisesse andmebaasi peab kasutaja *ette teadma*, millise väärtuse alusel sedasorti andmeid andmebaasis kokku viiakse. Pealtnäha erineb see tavapärasest relatsioonilisest andmebaasist, kus andmete lisamiseks andmebaasi ei pea kasutaja midagi teadma seal juba leiduvate andmete kohta. See erinevus on aga vaid näiline. Tavapärasesse andmebaasi andmete kirjutamiseks on vaja teada *relatsiooni nime*, kuhu vastavad korteežid paigutada. Et näitena kasutatud esialgsesse tühiväärtustega andmebaasi kirjutada kohal 4 istuva kliendi tellimus, on vaja teada, et tellimuste kohta käivaid andmeid hoitakse relatsioonis TELLIMUS*. Kirjutamise operatsioon on seejärel väljendatav järgmiselt:

(W_1) TELLIMUS*(KOHT = 4, PÕHIROOG = ‘Beyond smäsh’)

Ramsey-Wittgensteini lahendust järgivas andmebaasis ei ole aga relatsiooni nime andmete lisamisel vaja teada. Piisab teadmisest, et otsitaval relatsioonil on tunnus TEGEVUS, mis vajab väärtust ‘tellimus’. Andmebaasi juhtimissüsteem on üles seatav selliselt, et suudab ise selle põhjal andmebaasist leida sobivate tunnustega relatsiooni, kuhu korteežid kirjutada. Seega võib kirjutamise operatsioon, mida väljendab (W_1), olla Ramsey-Wittgensteini lahenduses esitatud järgmiselt:

(W_2) (TEGEVUS = ‘tellimus’, KOHT = 4, PÕHIROOG = ‘Beyond smäsh’)

Täiendavad tunnused ja nende väärtused, mida Ramsey-Wittgensteini lahendusest lähtuvalt on võimalik relatsioonides tuvastada, täidavadki sarnast ülesannet relatsioonide nimedega tavapärases relatsioonilises andmebaasis. See ei ole ka üllatav. Relatsioonid relatsioonilises mudelis on predikaatide ekstensioonid. Relatsiooni nimi vastab konstantsele osale predikaadis, mis annab predikaadile tähenduse ning eristab predikaati teistest samade tunnustega predikaatidest. Ramsey-Wittgensteini käsitluse kohaselt on aga tähendus baaspropositsioonides ning neist abstraheeritud predikaatides alati esitatav termide

väärtustena. Relatsiooni nimi relatsioonilises andmebaasis täidab seega loogika seisukohast termi funktsiooni. Kui baaspropositsioonis *kõik* termid muutujatega vahetada, on tulemuseks predikaat, mis on identifitseeritav täielikult muutujate alusel. Sellise predikaadi ekstensioon on seetõttu identifitseeritav vaid tunnuste alusel ning ei vaja relatsioonilises andmebaasis eraldi nime.

Sellest, et Ramsey-Wittgensteini lahendusel põhinevas andmebaasis ei vaja relatsioonid nimesid vaid on tuvastatavad tunnuste põhjal, ilmneb ühtlasi, et tunnused, mille väärtused erinevaid relatsioone omavahel ühendavad, ei pea võtma alati sama väärtust. Relatsioonide T1, T2 ja T3 omavahel ühendamiseks peavad küll korteežid neis kõigis andma tunnusele TEGEVUS sama väärtuse, kuid see, et nad kõik sisaldavad andmeid just tellimuste kohta, on juhus, mis tulenes sellest, et nad kõik olid sama relatsiooni horisontaalse dekompositsiooni tulemus. Andmebaasis, mis on algusest peale Ramsey-Wittgensteini lahenduse põhjal korraldatud, võib samade tunnustega relatsioonis esineda andmeid erinevate tegevuste kohta. Näiteks, kui klient istekohal 2 jättis ühtlasi tellitud roogadele arvustuse, oleks seda tõsiasja mudeldav korteež hoitav samas relatsioonis sama kliendi tellimuse kohta käiva korteežiga. Selline relatsioon oleks ekstensiooniks predikaadile, kus muutujaga on asendatud ka term, mis andmetesse tegevuse lisab. Sellist predikaati võiksime kirjutada näiteks järgmiselt:

(TA_2) ˊkoht KOHT TEGEVUS EELROOG ja PÕHIROOGˋ

Predikaadi (TA_2) ekstensiooniks oleks näite puhul järgmine relatsioon:

TA2

KOHT	**TEGEVUS**	**EELROOG**	**PÕHIROOG**
2	‘tellimus’	‘lillkapsatiivad’	‘laetud friikad’
2	‘arvustus’	‘lillkapsatiivad’	‘laetud friikad’

Tunnuse TEGEVUS väärtuse ‘tellimus’ abil on võimalik valida relatsioonist TA2 välja tellimuste kohta käivad andmed ning väärtuse ‘arvustus’ abil arvustuste kohta käivad andmed. Relatsioonile TA2 ei ole andmebaasis nime vaja, kuna sellesse kuuluvad kõik ja ainult need korteežid, mis vastavad TA2 tunnustele. Selline andmete korraldamise viis üldistub Ramsey-Wittgensteini lahendusel põhinevas andmebaasis kõigile relatsioonidele. Korteežid ei ole enam

relatsioonidesse kokku grupeeritud tunnuste ja tähenduse põhjal vaid *ainult* tunnuste põhjal. Tähenduse poolest kokku kuuluvad andmed — näiteks ainult tellimuste või ainult arvustuste kohta käivad andmed — on neist relatsioonidest alati päritavad piiranguoperatsiooni abil.

Jääb vaid üle selgitada, kuidas Ramsey-Wittgensteini lahenduse põhjal korraldatud andmebaasist puuduvate väärtustega andmeid pärida. Kuna andmed sellises andmebaasis on relatsioonidesse grupeeritud vaid tunnuste põhjal, on sama tähendusega andmete andmebaasist kättesaamiseks vaja leida kõik relatsioonid, milles on nõutud tunnused nõutud väärtustega. Näiteks kõigi tellimuste kohta käivate andmete kättesaamiseks on vaja leida korteežid, mis annavad tunnusele TEGEVUS väärtuseks ‘tellimus’. Nagu mainisin eespool ei ole relatsioonalgebras sel viisil andmete pärimiseks vajalikke operatsioone. Kõik relatsioonalgebra operatsioonid on tehtavad vaid juba valitud relatsioonidel, et nende põhjal koostada uusi relatsioone. Ramsey-Wittgensteini lahenduse põhjal korraldatud andmebaasis on aga enne selliste operatsioonide tegemist vaja kõigepealt relatsioonid andmebaasist valida. Teisisõnu, relatsioonalgebra operatsioonidega moodustatud päringutes on Ramsey-Wittgensteini lahenduse põhises andmebaasis vaja pärida kõigepealt *baasrelatsioonid*, millel neid operatsioone läbi viia.

Kuna relatsioonid Ramsey-Wittgensteini lahendusel põhinevas andmebaasis on tuvastatavad vaid tunnuste alusel, on üks viis baasrelatsiooni pärimiseks valida mingi kogum tunnuseid. Näiteks on järgmise tunnuste kogumi tulemuseks relatsiooni TA2:

(1) (KOHT, TEGEVUS, EELROOG, PÕHIROOG)

Kui andmebaasi ei ole kirjutatud ühtki korteeži, mis väärtustab kõiki ja ainult valitud tunnuseid, on neile tunnustele vastav relatsioon tühi, kuid relatsioonalgebra operatsioonid on sellel siiski teostatavad. Vaid tunnuste alusel relatsiooni pärimise probleemiks praktikas on, et sel viisil saadud relatsioonis võib esineda korteeže, mis ei ole teineteisega tähenduse poolest seotud. Selle vältimiseks on tunnuste alusel relatsiooni pärimisel aga võimalik anda mõnele valitud tunnusele ka väärtus. Tunnuste kogumile, kus mõni tunnus on väärtustatud, vastab relatsioon, kus igas korteežis on väärtustatud tunnuse kohal etteantud väärtus. Näiteks annab järgnev päring tulemuseks relatsiooni TA2 osarelatsiooni T2, milles on vaid sellised korteežid, mis tunnusele TEGEVUS annavad väärtuseks ‘tellimus’:

(2) (KOHT, TEGEVUS = ‘tellimus’, EELROOG, PÕHIROOG)

Sel viisil päritud baasrelatsioonid sisaldavad tähenduse poolest seotud andmeid ning vastavad relatsioonidele, millele relatsioonilises andmebaasis tavapäraselt viidatakse relatsiooni nime abil.

Puuduvate väärtustega andmete kokku kogumiseks tunnuste valimisest ja osalisest väärtustamisest aga ei piisa, kuna puuduvate väärtustega andmed, mis tähenduse poolest kokku kuuluvad, võivad esineda erinevate tunnustega relatsioonides. Et näha, kuidas ka puuduvate väärtustega andmeid kokku koguda, on vaja vaadata teist poolt Ramsey ideel, mida tutvustasin eelnevas alapeatükis. Olles selgitanud, kuidas propositsioonist 'Sokrates on tark' on termide eemaldamisel võimalik saada predikaate nagu ⌜x on tark⌝, täheldab Ramsey, et on veel teine viis, kuidas propositsioonist predikaati moodustada.[48] Propositsioonis on võimalik muutujaga asendada ka kõik *peale* valitud termide. Propositsioonist 'Sokrates on tark' on sel viisil võimalik moodustada näiteks predikaat ⌜X(tark)⌝, mis on ühiseks vormiks kõigile propositsioonidele, kus esineb term 'tark'. Ka sellised predikaadid on loogikast üle toodavad relatsioonilisse andmemudelisse. Eelneva näitega jätkates on Ramsey idee põhjal võimalik andmebaasi vastu teha päring ka näiteks järgmise predikaadiga:

(3) X(TEGEVUS = 'tellimus')

Andmebaasis on predikaat (3) ühiseks vormiks kõigile korteežidele, mis annavad tunnusele TEGEVUS väärtuseks 'tellimus'. Predikaadi (3) ekstensiooniks on vastavalt kõigi selliste andmebaasis leiduvate korteežide hulk ehk hulk, mis sisaldab kõiki andmebaasis leiduvaid tellimuste kohta käivaid andmeid. Kuna predikaat (3) ei sea mingeid täiendavaid piiranguid sellele, millistele tunnustele need korteežid väärtuseid peavad andma, võib selles hulgas esineda erinevate tunnuste ja erineva pikkusega korteeže. Näiteks esinevad predikaadi (3) ekstensioonis kõik korteežid alapeatükis 4.1 näitena toodud relatsioonides T1, T2 ja T3.

Esmapilgul rikub erinevate tunnuste ja pikkustega korteežide esinemine predikaadi (3) ekstensioonis Coddi informatsioonireeglit, mida tutvustasin peatükis 1.3. Meenutuseks, informatsioonireegli kohaselt peavad andmed relatsioonilises mudelis alati esinema relatsioonides. Relatsioon koosneb aga vaid samu tunnuseid väärtustavatest korteežidest. Et predikaadi (3) ekstensioon on siiski mõistetav viisil, mis on informatsioonireegliga kooskõlas, tuleneb mõningatest eripäradest Ramsey-Wittgensteini põhimõttel korraldatud andmebaasis, mida olen eelnevalt tutvustanud. Esiteks on korteežist alati võimalik saada kogum tunnuseid,

[48] vt Ramsey 1925: 410

mida see korteež väärtustab. See on lihtsalt propositsioonist predikaadi abstraheerimine. Teiseks, kuna relatsioonid on tuvastatavad vaid tunnuste alusel, vastab tunnuste kogumile alati relatsioon. Arvestades neid kaht eripära, moodustavad predikaadi (3) ekstensiooni sees samade tunnustega korteežid omakorda relatsiooni ning predikaadi kogu ekstensioon osutub seeläbi mitte enam erinevate tunnuste ja pikkustega korteežide hulgaks vaid *relatsioonide* hulgaks. Relatsioonide hulgast omakorda võib aga mõelda ka kui ühekohaliste korteežide hulgast, kus iga korteeži ainsaks elemendiks on omakorda relatsioon. Nii on predikaadi (3) ekstensioon käsitletav kui relatsioonide relatsioon ehk *teist järku* relatsioon.

Kuigi teist järku relatsioone harilikult relatsioonilises mudelis ei käsitleta, eeldab Codd ise neid osades oma kirjutistes.[49] Vajadust teist järku relatsioonide järele relatsioonilises mudelis kaitseb ka Date.[50] Relatsioonalgebra operatsioonid on teist järku relatsioonidel tehtavad samamoodi nagu tavapärastel esimest järku relatsioonidel, kuna korteežide sees esinevad väärtused ei oma nende operatsioonide toimimise seisukohast tähtsust. Nii on võimalik relatsioonilises andmebaasis moodustada relatsioonide otsekorrutise abil ka näiteks mitmekohalisi teist järku relatsioone. Põhjalikum teist järku relatsioonide käsitlemine relatsioonilise mudeli kontekstis jääb aga siinse töö teema alt välja. Ramsey-Wittgensteini lahenduses puuduvate väärtuste probleemile on vaja neid vaid tähenduse poolest kokku kuuluvate andmete pärimiseks.

Olen nüüdseks tutvustanud Ramsey-Wittgensteini lahendust puuduvate väärtuste probleemile ning näidanud selle eeliseid Date'i lahenduse ees. Nagu eespool selgitasin, ei ole Date'i lahendus universaalne, kuna eeldab, et andmetes on tunnuseid, millel ei saa väärtus puududa. Ramsey-Wittgensteini lahendusel seda piirangut ei ole. Samas võib aga olla nii, et andmetes juhtumisi on Date'i lahenduseks vajalikud tunnused olemas. Sel juhul on Date'i lahendus kasutatav ka Ramsey-Wittgensteini andmebaasis. Kui anda väärtuseid korteeže horisontaalselt kokku siduvatele tunnustele, moodustuvad relatsioonid, mis sisaldavad samade tunnuste ja tähendusega andmeid, just nagu relatsioonid tavapärases relatsioonilises andmebaasis. Kui sellisel relatsioonil on olemas võti, on selle abil võimalik koostada relatsioonile ka pseudotunnuseid, mis võimaldavad puuduvate väärtuste haldamist Date'i kirjeldatud viisil. Ramsey-Wittgensteini lahendus seega ei välista Date'i lahendust. Mõlemad saavad andmebaasis olla koos kasutusel.

[49] vt nt Codd 1969:3
[50] vt Date 2012: 31–33

Kokkuvõte

Esitasin Ramsey-Wittgensteini loogikakäsitlusel põhineva viisi andmete organiseerimiseks relatsioonilises andmemudelis, mis võimaldab hallata puuduvate väärtustega andmeid, vajamata seejuures tühiväärtuseid. Selgitasin esiteks, kuidas andmebaasid põhinevad loogikal, ning kuidas sellest lähtuvalt on relatsiooniline mudel mõistetav läbi predikaatarvutuse. Tutvustasin seejärel puuduvate väärtuste probleemi ning E. F. Coddi tühiväärtuseid kasutavat lahendust sellele. Näitasin, et Coddi lahenduse probleemiks on, et see võimaldab valede tulemustega päringuid. Seejärel tutvustasin C. J. Date'i lahendust, mis tühiväärtuseid ei eelda, kuid on kasutatav vaid andmetel, milles on tuvastatav võti. Puuduvate väärtuste võimalikkuse korral ei ole aga võtme olemasolu andmetes garanteeritud. Lõpuks esitasin Ramsey-Wittgensteini lahenduse puuduvate väärtuste probleemile, mis erinevalt Date'i lahendusest ei eelda võtme olemasolu andmetes. Selgitasin lahenduse väljatöötamise käigus ka mõningaid olulisi erinevusi, mis on Ramsey-Wittgensteini lahenduse põhjal korraldatud andmebaasil tavapäraste relatsiooniliste andmebaasidega võrreldes, kuid näitasin, et Ramsey-Wittgensteini lahendust järgiv andmebaas vastab siiski relatsioonilisele mudelile.

Kasutatud kirjandus

Brown, Fiona 2024. ‘Relational Algebra in DBMS: Operations with Examples’, URL = <https://www.guru99.com/relational-algebra-dbms.html> (külastatud 11.08.2024)

Codd, Edgar F. 1969. ‘Derivability, redundancy and consistency of relations stored in large data banks’, URL = <https://technology.amis.nl/wp-content/uploads/images/RJ599.pdf> (külastatud 10.08.2024)

Codd, Edgar F. 1970. ‘A Relational Model of Data for Large Shared Data Banks’, *Communications of the ACM*, vol. 13, no. 6, lk. 377–387, URL = <https://dl.acm.org/doi/pdf/10.1145/362384.362685> (külastatud 18.07.2024)

Codd, Edgar F. 1985. ‘Does your DBMS run by the rules?’, *Computerworld*, vol. 19, lk. 49; taasavaldatud 2019 pealkirja all ‘The 12 rules’, URL = <https://reldb.org/c/index.php/twelve-rules/> (külastatud 18.07.2024)

Codd, Edgar F. 1990 *The Relational Model for Database Management*, teine versioon. Addison-Wesley Publishing Company.

Date, C. J. 2012. *SQL and Relational Theory: How to Write Accurate SQL Code*, teine väljaanne. O’Reilly Media.

Date, C. J., Hugh Darwen 2014. *Databases, Types, and the Relational Model: The Third Manifesto*, kolmas väljaanne, URL = <https://www.dcs.warwick.ac.uk/~hugh/TTM/DTATRM.pdf> (külastatud 11.08.24)

Dummett, Michael 1973. *Frege: Philosophy of Language*. Duckworth Publishing.

Halbach, Volker 2010. *The Logic Manual*. Oxford University Press.

Leonard, Henry S., Nelson Goodman 1940. ‘The Calculus of Individuals and Its Uses’ — *The Journal of Symbolic Logic*, vol. 5, no. 2, lk. 45–55.

Magnus, P. D. 2017. *Forallx: An Introduction to Formal Logic*, URL = <https://www.fecundity.com/codex/forallx.pdf> (külastatud 11.08.24)

Ramsey, Frank P. 1925. ‘Universals’ — *Mind*, vol. 34, no. 136, lk. 401–417.

Schweikardt, Nicole, Thomas Schwentick, Luc Segoufin 2010. ‘Database Theory: Query Languages’ — M. J. Atallah, M. Blanton (toim.), *Algorithms and Theory of Computation Handbook: Special Topics and Techniques*, 2. väljaanne, ptk. 19, lk. 1–34. CRC Press.

Sider, Theodore 2010. *Logic for Philosophy*. Oxford University Press.

Szabó, Zoltán Gendler 2022. ‘Compositionality’ — *The Stanford Encyclopedia of Philosophy*, sügis 2022 väljaanne, E. N. Zalta, U. Nodelman (toim.), URL = <https://plato.stanford.edu/archives/fall2022/entries/compositionality/> (külastatud 07.08.2024)

Wittgenstein, Ludwig 1922. *Tractatus Logico-Philosophicus / Logisch-philosophische Abhandlung*. Kegan Paul. URL = <https://people.umass.edu/klement/tlp/> (külastatud 08.08.2024)

Wittgenstein, Ludwig 1961. *Notebooks 1914–1916*, G. H. Von Wright, G. E. M. Anscombe (toim.), G. E. M. Anscombe (tõlk.). Harper & Row, Publishers.

LISAD

I. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Indrek Lõbus,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose

„Puuduvate väärtuste probleem relatsioonilises andmemudelis ja selle loogikapõhine lahendus,“

mille juhendaja on Anne Villems,

reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.
3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.
4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Indrek Lõbus

12.08.2024