TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Tarkvarateaduse instituut

Eliis Hövel 134292IAPB

MÕNED MITMEKEELSETE SQL-ANDMEBAASIDE DISAINIMUSTRID

Bakalaureusetöö

Juhendaja: Erki Eessaar
Doktor

Tallinn 2017

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Eliis Hövel

22.05.2017

Annotatsioon

Käesoleva töö eesmärgiks on kirjeldada disainimustreid mitmekeelsete andmete SQL-andmebaasis hoidmiseks, kavandada näited, realiseerida disainimustrite põhjal päringute ja andmete muutmise lausete käivitamiseks näiteandmetega SQL-andmebaas, realiseerida mitmekeelne veebirakendus ühe analüüsi põhjal valitud disainimustri põhjal ning saadud kogemust analüüsida.

Mitmekeelsete andmete salvestamiseks mõeldud SQL-andmebaasi realiseerimiseks on mitmeid lahendusi, kuid käesoleva töö autor ei leidnud ühtegi allikat, kus need oleks kõik koos ja struktureeritult kirjeldatud. Käesolevas töös püütakse olemasolevate mustrite ja soovituste põhjal mitmes keeles andmete hoidmiseks mõeldud SQL-andmebaaside disainimustreid struktureeritult kirja panna, kirjeldades nende tugevaid ja nõrku külgi.

Töö tulemusena esitati mustri formaadis kirjeldatuna seitse SQL-andmebaasi disainilahendust. Iga lahenduse kohta toodi välja PostgreSQL näiteandmebaasi tabeleid kirjeldav andmebaasi disaini diagramm. Võrreldavuse huvides lahendasid kõik need näited sama ülesannet. Kõikide mustrite puhul need tabelid realiseeriti, käivitati nende põhjal andmekäitluskeele lauseid ja esitati tulemused. Analüütiliste hierarhiate meetodi abil valiti välja üks disain, mida kasutati PHP veebirakenduse realiseerimisel. Näiterakenduses põhjustab keele muutmine nii kasutajaliidese keele kui ka kasutajaliideses kuvatud andmete muutmise. Mängiti läbi süsteemi uue keele toe lisamine ning see ei olnud keeruline. Valminud näiterakendus on aadressil: http://apex.ttu.ee/t134292/index.php.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 49 leheküljel, 6 peatükki, 46 joonist, 7 tabelit.

Abstract

Some Design Patterns for Keeping Multilingual Data in SQL Databases

The goal of this thesis is to describe design patterns for keeping multilingual data in SQL databases, design examples, create an example SQL database based on the described design patterns, build a web-based application based on one chosen design, and analyse the experience.

There are many approaches to design an SQL database for keeping multilingual data, but the author of the current thesis could not find any sources where they would be all together and described in a structured way. Based on the existing patterns and suggestions, the author will create a catalogue of possible design patterns and analyse their strengths and weaknesses.

The results of this study include seven SQL database design patterns. For each pattern, a design diagram of an example PostgreSQL database was presented. In order to be comparable, all these tables solved the same task. All the tables were implemented in an example database. The database was used for executing some data manipulation statements. One design was picked out by using the analytic hierarchy process and used for creating an example PHP-based web application. In the web application, changing the language changes both the language of user interface as well as the presented data. Adding support of a new language was tested and it was not difficult. The address of the web application is: http://apex.ttu.ee/t134292/index.php.

The author of this thesis found that there is no universally best design pattern for a multilingual database. The choice of a pattern depends on the system that is being built and on the criteria that are considered important. In addition, the design patterns that were presented could be combined and extended in different ways.

An important aspect that was left out of the scope of this thesis, is the comparison of performance of data manipulation operations for each pattern when using the database. This comparison could remarkably change the results of the analytic hierarchy process.

Moreover, since there was no framework used for creating an example web application, it could be investigated which are the best frameworks for creating multilingual web applications.

The thesis is in Estonian and contains 49 pages of text, 6 chapters, 46 figures, 7 tables.

Lühendite ja mõistete sõnastik

AHP	*Analytic Hierarchy Process* Analüütiliste hierarhiate meetod ehk Saaty meetod, mida kasutatakse otsustussegaduse korral subjektiivsete hinnangute alusel valikute e alternatiivide vahel parima valiku tegemisel. Meetod põhineb mõjurite e kriteeriumite ja valikute e alternatiivide paariviisilisel võrdlemisel, et leida nende suhteline olulisus [51].
Andmete liiasus	*Data Redundancy* Andmebaasis on andmete liiasus siis ja ainult siis, kui sama väite saab tuletada sellest kahel või rohkemal erineval viisil.
Andmete terviklikkus	*Data Integration* Andmed on terviklikud, kui andmebaasis ei hoita vastuolulisi fakte, vaid andmed on omavahel kooskõlas.
CASE	*Computer-Aided Software Engineering* „Arenduskeskkond, mis võimaldab projekteerida, disainida ja luua tarkvara ja infosüsteeme.“ [17]
HTML	*HyperText Markup Language* „Enimlevinud kodeerimissüsteem (tekstivorming) veebidokumentide loomiseks. HTML koodid ehk märgendid määravad ära selle, kuidas veebileht arvutiekraanil välja näeb.“ [48]
Kitsendus	*Constraint* Piirang, mida SQL-is saab määrata tabelitele ja veergudele. Kitsendused võimaldavad kontrollida andmebaasi sisestatavaid andmeid enne, kui need salvestatakse ning muudatused, mis viivad andmed kitsendustega vastuollu, tagasi lükata.
MVC	*Model-View-Controller* Rakenduse arhitektuuri disainimuster, kus rakendus jaotatakse komponentideks. See võimaldab koodi efektiivselt taaskasutada ja mitme programmeerija poolt paralleelselt arendada.
NULL	„Marker SQL-is, mis tähistab andmete puudumist.“ [17]
PHP	*PHP: Hypertext Preprocessor* Laialdaselt kasutatav tasuta vabavarana pakutav skriptimiskeel, mida sobib eriti kasutada veebirakenduste loomisel.
PostgreSQL	„Populaarne, tasuta, vabavarana pakutav objekt-relatsiooniline

	andmebaasisüsteem, kus saab kasutada andmebaasikeelt SQL.“ [17]
SQL	*Structured Query Language* Andmebaasikeel andmebaasis olevate andmete, andmestruktuuride ning neid ümbritsevate muude andmebaasiobjektide ja andmebaasi pääsuõiguste haldamiseks. SQL-andmebaas on antud töös andmebaas, mille loomisel on kasutatud andmebaasisüsteemi, milles kasutatakse SQL-i [17].
Tabel	*Table* „SQL-andmebaasi põhiline ehitusplokk, mis ei ole defineeritud teiste tabelite põhjal.“ [17]
Tarkvara mitmekeelsus	*Multilingual software* Andmebaasi mitmekeelsus tähendab, et andmebaasis hoitakse mitmes keeles andmeid. Rakenduse mitmekeelsus tähendab, et kasutajaliidese keelt on võimalik muuta nii, et andmed ning nuppude, vormiväljade ja muude elementide pealkirjad kuvatakse valitud keeles.
Unikood	*Unicode* „Rahvusvaheline standard arvutites kirjasüsteemide kodeerimiseks.“ [17]
URI	*Uniform Resource Identifier* „Internetis leiduvail tarkvaralistel võrguressurssidel (HTML dokumendid, pildid, videoklipid, programmid, muusika jne) on igaühel oma unikaalne URI, mis määrab kindlaks ressursi nime (URN) ja aadressi (URL), mis näita ära, kust ja kuidas võib seda ressurssi kätte saada.“ [48]
UTF-8	*Unicode Transformation Format* Üks võimalik unikoodi kodeering.

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Bakalaureusetöö teemaks on uurida erinevaid SQL-andmebaaside disainilahendusi mitmes keeles andmete hoidmiseks ning valida välja nendest üks, mille põhjal katsetada mitmekeelse rakenduse loomist.

1.1 Taust ja probleem

Mitmekeelsete rakenduste loomine ja mitmes keeles andmete salvestamine on pikemat aega olnud aktuaalsed ja seoses rahvusvahelistumisega, muutuvad aina olulisemaks. Rakenduse mitmekeelsus tähendab nii andmete kui kasutajaliidese mitmekeelsust, kuid mitte andmebaasiskeemis olevate objektide nimede ehk identifikaatorite mitmekeelsust. See tähendab, et muutes rakenduses keelt muutub nii kasutajaliidese keel kui ka rakenduse kaudu kasutajale näidatavate andmete keel. Keelte arvu osas on süsteemidel tihti erinevad nõudmised: ühes piisab paarist keelest ja tulevikus neid juurde ei lisata; teises on sageli vaja lisada uue keele tugi, sest ettevõte laieneb teise riiki jne. Seega on oluline teada, kuidas peaks andmebaasi projekteerima, et seda saaks vajadusel võimalikult lihtsalt uue keele toega laiendada.

Töö kirjutamise ajal (2017. aasta kevad) on SQL-andmebaasisüsteemid endiselt väga populaarsed. 2017. aasta maikuu seisuga on andmebaasisüsteemide populaarsuse esikümnes seitse SQL-andmebaasisüsteemi, sealhulgas neli esimest [3]. Seega keskendutakse käesolevas töös SQL-andmebaaside disainimisele.

Antud töö on mõeldud kõigile neile, kes tegelevad andmebaasidega, kus tuleb hoiustada mitmes keeles andmeid ja loovad nendele andmebaasidele andmebaasirakendusi.

Esitatud füüsilise disaini mudelid on koostatud CASE-vahendis Enterprise Architect ning näiteandmebaasid on realiseeritud PostgreSQL 9.5 andmebaasisüsteemis. PostgreSQL on valitud, kuna see on võimekas, populaarne, tasuta, avatud lähtekoodiga ning autoril on selle kasutamise kogemus. Näiterakenduse realiseerimiseks kasutatakse PHP programmeerimiskeelt. PHP on veebirakenduste loomiseks laialt kasutatud,

populaarne (2017. aasta mai seisuga programmeerimiskeelte populaarsuse pingereas üheksandal kohal [43]) ning jällegi autorile tuttav. PHP rakendusraamistikke ei kasutata.

1.2 Ülesande püstitus

Töö eesmärgiks on süstematiseerida infot mitmekeelsete SQL-andmebaaside disaini kohta ning analüüsida nende disainide kasutatavust päringute ja näiterakenduse loomise abil.

Töö põhitulemused:

- Mitmekeelsete SQL-andmebaaside loomiseks mõeldud disainilahenduste leidmine ja mustrite formaadis kirja panemine. Eesmärgiks ei ole kirja panna täielikku võimalike mustrite nimekirja (näiteks erinevaid lahendusi on võimalik kombineerida ja varieerida).
- Mitmes keeles andmete hoidmise jaoks mõeldud andmebaasi realiseerimine PostgreSQL-is, kasutades erinevaid disainimustreid.
- Loodud andmebaaside põhjal andmete otsimise, lisamise ja muutmise ülesannete lahendamiseks mõeldud andmebaasikeele lausete koostamine.
- Disainilahenduste võrdlemine ja analüüsimine mustrites (kirjanduse põhjal) välja toodud eeliste, probleemide ja koostatud lausete keerukuse põhjal ning analüüsi tulemuste alusel näiterakenduse jaoks ühe disaini välja valimine. Analüüsi tegemisel kasutatakse analüütiliste hierarhiate meetodit ehk Saaty meetodit [36].
- Valitud disainiga tabeleid kasutava mitmekeelse näiterakenduse alamosa disainimine ja realiseerimine. Lisaks hinnatakse, kas valitud disain on nii hea, kui analüütiliste hierarhiate meetodi kasutamise tulemusena loodud otsustusmudel välja pakub.
- Uue keele toe lisamine ja selle mõju hindamine andmebaasile ja rakendusele.

1.3 Metoodika

Töös kasutatakse disainiteaduse metoodikat (*design science research*) [47]. Töö tulemusena valmivad uued disainimustrid (artefaktid ehk tehised). Nende valideerimiseks luuakse disainimustrite põhjal CASE-vahendis esmalt andmebaasi

füüsilise disaini diagrammid. Seejärel realiseeritakse tabelid andmebaasisüsteemis (sama andmebaasi erinevates skeemides) ja katsetatakse andmebaasikeele lausetega. Lisaks realiseeritakse näiterakendus, mis kasutab valikuliselt üht loodud tabelite hulka ja on ka rakenduse tasemel mitmekeelne.

1.4 Ülevaade tööst

Käesolev töö koosneb seitsmest suuremast sisupeatükist. Ülevaade veebilehtede tõlkimisega seotud keerukusest ja masintõlke kasutamisest antakse peatükis „Tõlkimise keerukus“. Sellele järgneb peatükk „Mustrite struktuur ja realiseerimine“, kus kirjeldatakse mustreid üldisemalt, esitatakse antud töös kasutatav mustrite struktuur ja käsitletakse ka mustrite realiseerimise tehnilist poolt. Peatükis „Mustrite kataloog“ on töö põhitulemuseks olevad mustrid. Järgnevas peatükis vaadeldakse erinevate disainide korral andmebaasi loomise ja kasutamise keerukust füüsiliste koodiridade arvu võrdlemise abil. Peatükis „Parima disaini valimine kasutades Saaty meetodit“ võrreldakse disainilahendusi autori subjektiivsete hinnangute alusel ja valitakse välja disain, mida kasutatakse rakenduse realiseerimisel. Viimane peatükk on realiseeritud rakenduse kohta ja selles peatükis on esitatud rakenduse analüüsi ja disainiga seonduv osa. Samuti käsitletakse süsteemi uue keele toe lisamist. Töö kokkuvõttes osutatakse muuhulgas valdkonnaga seotud edasistele uurimissuundadele.

2 Tõlkimise keerukus

Rahvusvahelikustamine (*internationalization*) on protsess, mille käigus toodet ehk rakendust üldistatakse selliselt, et see oleks võimeline toetama mitut keelt ja kultuurikonventsiooni, ilma uuesti disainimise ja realiseerimise vajaduseta [35]. Rahvusvahelikustamisega tegeletakse seega rakenduse disainifaasis. Lokaliseerimine (*localization*) on rakenduse ühe sihtkeele kultuuriruumi jaoks keeleliselt ja kultuuriliselt sobivaks kohandamine. See tähendab, et lokaliseerimine kätkeb endas ka andmete tõlkimist.

Üks variant on tõlkida veebisaidil esitatavaid andmeid jooksvalt ning automaatselt, mida saab teha kasutades tarkvara nagu Google Translate [49], Systran [39], Prompt [29], GTS Website Translator [10] jne. Automaatsel tõlkimisel on eeliseid nagu valmis vahendite olemasolu, aja kokkuhoid, soodsam hind, võimalus valida paljude keelte vahel[1] ja konfidentsiaalsus [42]. Sellel lähenemisel on aga ka puuduseid nagu automaattõlkija poolne ebapiisav arvestamine kultuuriliste ja keeleliste eripäradega[2], ei toetata igasse keelde tõlkimist[3], vajadus esitada andmeid mitte-veebipõhiselt ning ei olda võimeline esitama täpseid andmeid [35].

---

[1] Pistikprogramm Google Translate võimaldab veebilehte tõlkida rohkem kui sajasse erinevasse keelde [49]. Libeerias sündinud Z. Fazah räägib väidetavalt 59 keelt [54].

[2] Masin ei oska hinnata konteksti, mille tõttu ei kasutata tõlkimisel alati sobivaimat sõnavara ning tähendus võib kaduma minna. Näiteks idioomid ja sõnade mitmetähenduslikkus võivad probleeme tekitada: eestikeelse fraasi „kivi kotti“ tõlkimisel tuleb kindlasti konteksti jälgida; eesti keeles sõna „tee“ võib tähendada tegusõna „tegema“ üht vormi, teelehtedest valmistatud jooki, käimiseks kasutatavat pinnaseriba, liikumissuunda, teekonda, meetodit jne [6]; saksa keeles sõna „Schloß“ võib tähendada lukku või lossi [44]. Mõni sõna võib olla nii kultuurispetsiifiline, et sellel puudub vaste teistes keeltes: eesti keelest võib näiteks tuua ühe toidu, milleks on „kama“.

[3] ISO 639-1 standardis on defineeritud 184 põhilise keele kahekohalised koodid ja ISO 639-2 standardis umbes 500 keele kolmekohalised koodid [2], vähemalt osa piiblist oli 2009. aasta seisuga tõlgitud 2058 erinevasse keelde [1] ning maailmas on teadaolevalt kokku 7099 keelt [8], kuid näiteks Google Translate toetab 103 keelde tõlkimist [9].

2017. aasta Eesti keeletehnoloogia konverentsil käsitleti masintõlget ja seal toodi mitmed eelpool nimetatud puudused ka Mark Fišeli[1] poolt välja. Lisaks võrreldi konverentsil neuromasintõlget statistilise masintõlkega ning märgiti ära, et neuromasintõlke korral tõlgitakse tihti harvaesinevad sisendid valesti[2] ning tõlgitakse ka isikute ja firmade nimed, mida üldiselt ei peaks tõlkima. Statistilise masintõlke puhul on aga peamisteks probleemideks sõnade järjekord ja morfoloogiline ühilduvus. Mõlemat tüüpi masintõlke kitsaskohaks on ebastabiilne kvaliteet pikemate lausete tõlkimisel.

Alternatiivne lahendus on salvestada andmebaasis andmed mitmekeelsena. Siis on paindlikkus suurem – algse tõlke võib teha automaatselt ja hiljem lasta inimesel tõlge järeltoimetada või tõlkida andmed kohe inimese poolt. Selle variandi puhul tuleb andmebaasi loomisel mõelda, kuidas seal andmeid hoida ehk kuidas disainida andmebaas, kus mitmes keeles andmeid hoitakse.

Autor uuris mitmeid foorumeid ja blogipostitusi, kus arutleti mitmes keeles andmete salvestamise teemal [4], [14], [15], [19], [20], [26], [38], [45], [52], [55]. Palju leidus seisukohti, et lahendus sõltub konkreetse süsteemi nõuetest, kuid valdavalt soovitati andmebaas disainida selliselt, et seda oleks võimalik laiendada, ilma et tekiks vajadus olemasoleva andmebaasi struktuuri muuta.

[1] Mark Fišel on projekti KaMa (Kasutatav Eesti Masintõlge) projektijuht, mille eesmärk on „… arendada masintõlkesüsteeme tõlkimiseks eesti keelest ja eesti keelde, ning tõsta tõlkekvaliteeti nende kasutatavuseni äritasemel“ [18].

[2] Vikipeediast võetud lause tõlkimisel Tartu Ülikooli masintõlkijaga [41] ilmnes, et lause „A large-scale cyber attack involving ransomware causes severe disruption around the world“ korral tõlgiti *ransomware* hoopis maavalduseks.

3 Mustrite struktuur ja realiseerimine

Antud töö peatükis „Mustrite kataloog“ esitatakse mustrid mitmes keeles andmete hoidmiseks SQL-andmebaasis. Mustrite kirjeldamisel kasutatakse ühist struktuuri, mille aluseks on võetud Vellemaa [50] ja Jõgi [17] bakalaureusetööd. Nendes töödes kasutati struktuuri, mis on sobiv ka käesoleva töö jaoks.

Mustrite struktuur on järgnev:

- *Nimi*, mis peab võimalikult täpselt edasi andma mustri sisu. Kuna iga mustri nimi on esitatud peatüki „Mustrite kataloog“ alampeatüki pealkirjana, siis mustri nime enam mustrites ei korrata. Mustrite nimedes viidatakse mõnikord mõistetele „olemitüüp“ ja „atribuut“. Lähtutakse eeldusest, et kontseptuaalses andmemudelis kirjeldatud olemitüübi või atribuudi kohta, luuakse andmebaasis vastavalt kas tabel või veerg.
- *Probleem*, mida see muster lahendab. Antud töös on probleem kõikidel disainimustritel sarnane: kuidas hoida mitmes keeles andmeid SQL-andmebaasides nii, et oleks lihtne koostada päringuid ja andmemuudatusi ning uue keele toe lisamise mõju andmebaasile ei oleks suur ja nõuaks võimalikult vähest andmete kasutamise koodi muutmist. Seetõttu probleemi peatükis „Mustrite kataloog“ uuesti ei esitata.
- *Taust*, millal nimetatud probleem võib tekkida. Antud töös on taust kõikidel disainimustritel sarnane: uue SQL-andmebaasi disainimisel või olemasoleva edasiarendamisel on tarvis otsustada, missugusel viisil mitmes keeles andmeid andmebaasis hoida. Seetõttu tausta peatükis „Mustrite kataloog“ uuesti ei esitata.
- *Jõud*, mis võivad mõjutada probleemi või selle lahendust.
- *Lahendus probleemile*. Üldine sõnaline kirjeldus.
- *Allikad ja autorid*, mis demonstreerivad mustris pakutava lahenduse kasutamist ja kelle soovituste põhjal on käesoleva töö autor mustreid formuleerinud.
- *Lahenduse mõistlikkus, mustri tugevad ja nõrgad küljed*. Tuginetakse nii kirjanduses esitatud väidetele kui ka autori mõtetele selle disaini kohta.

- *Näide*. Kõikide mustrite puhul esitatakse näide andmebaasi põhjal, kus on registreeritud tooted ja nende hetkeseisundid. Iga toode on käesoleval hetkel täpselt ühes seisundis. Iga seisundiga on seotud null või rohkem toodet. Toote seisundi liik ja keel on klassifikaatorid. Klassifikaatorite koodid ja nimetused peavad olema keele piires unikaalsed. Samuti peavad unikaalsed olema toote kood ja EAN kood. Näitetabelite kirjeldus eeldab, et tabelid luuakse PostgreSQL (9.5) andmebaasis. Andmebaasiobjektide nimed on ingliskeelsed, et näiteks oleks hiljem lihtsam loodud kataloogi laiemale lugejaskonnale jagada. Kõikide mustrite korral esitatakse SQL laused järgmiste ülesannete lahendamiseks.
  - Toote sisestamine.
  - Toodete otsimine ühes keeles ja nimetuse järgi sorteerimine. Kuna kuvatõmmis päringu tulemusest on kõigil sama, siis esitatakse see vaid esimese kirjeldatud mustri ehk mustri „Tõlkeveerud“ korral.
  - Toote muutmise laused muudavad toote hinna, seisundi ja eestikeelse nimetuse. Vastavad näitelaused leiab Lisast 1.

3.1 Andmebaaside PostgreSQL-is realiseerimise tehnilisi küsimusi

Mitmekeelse süsteemi korral tuleks kasutada unikoodi, sest see toetab enamikke maailmas olemasolevaid kirjasüsteeme [32]. PostgreSQL-i korral saab serveri seadistada nii, et igal uuel andmebaasil on vaikimisi unikoodi UTF-8 kodeering. Lisaks saab kodeeringu määrata iga andmebaasi jaoks eraldi andmebaasi loomise hetkel.

PostgreSQL võimaldab kasutada võrdlusreeglistikku (*COLLATION*), mis määrab sortimise järjekorra ja tähestiku klassifikatsiooni [31]. Võrdlusreeglistiku kasutamist saab sätestada kas veerupõhiselt või operatsioonipõhiselt. Kui ühes veerus sisalduvad ainult ühes keeles andmed, siis on mõistlik sellele veerule juba tabeli loomisel või tabelisse veeru lisamisel määrata vastava keele võrdlusreeglistiku kasutamine, sest siis sorditakse andmed vastavalt selle keele reeglitele. Näiteks eesti keele reeglite põhjal sorditakse tähed õ, ä, ö ja ü kasvava järjekorra puhul nimekirja lõppu, kuid inglise keele reeglite puhul sorditakse ä-täht a-tähe järele jne.

„Mustrite kataloogis“ on klassifikaatorite korral näitetabelite diagrammides kasutatud nimetuse korral andmetüüpi *varchar(50)* ja kirjelduse korral *varchar(255)*. Erandiks on mustrid „Sisu tüübiga universaalne tõlketabel olemitüübi kohta“ ja „Korduskasutatavad

tõlked“, kus toote seisundi liigi tõlked salvestatakse *text*[1] tüüpi välja. Erandi tegemise põhjuseks on, et nii nimetused kui kirjeldused salvestatakse samasse veergu, mis tähendab, et mõlemad peavad kasutama sama andmetüüpi ning väljapikkuse valimisel tuleb arvestada suurima võimaliku väärtusega. Põhiobjekti, milleks antud juhul on *Toode*, keelest sõltuvate tekstiliste väärtuste korral on näitetabelite diagrammides alati andmetüüpi *text* kasutatud. See lubab küll piiramatu pikkusega teksti sisestada, kuid vajadusel saab domeeni kasutades teksti pikkust piirata. Lisaks on kõikide tekstiliste väärtuste korral jõustatud kitsendus, mis ei luba salvestada tühja või ainult tühikutest koosnevat stringi.

Kõikide disainide korral on jõustatud unikaalsuse kitsendus toote koodi ja EAN koodi korral. Mustri „Tõlkeread“ korral, peavad unikaalsed olema toote koodi ja keele koodi kombinatsioon ning EAN koodi ja keele koodi kombinatsioon, sest selle disaini puhul korratakse ühe toote koode igal selle toote real. Lisaks peavad igas keeles unikaalsed olema klassifikaatorite nimetused. Disainide korral, mis kasutavad keelepõhiseid tabeleid või veerge (nimes on keeleliide), on selle saavutamiseks vaja nimetuse veerule unikaalsuse kitsendus rakendada. Disainide puhul, kus keel on eraldi klassifikaator, tuleb unikaalsuse kitsendus rakendada keele ja nimetuse kombinatsioonile. Probleem tekib mustri „Sisu tüübiga universaalne tõlketabel korral, kuna selle puhul ei ole võimalik niisama lihtsalt unikaalsuse kitsendust määrata, sest ühes veerus hoitakse erinevat tüüpi tõlgitavaid andmeid (näiteks nimetused ja kirjeldused). Sellisel juhul tuleb otsustada, kas määrata igat tüüpi andmete väärtused unikaalseks või kõik mitteunikaalseks. Kolmas variant on luua unikaalsed osalised indeksid. Osaline indeks luuakse ainult teatud tingimusele vastavatele ridadele [5]. Selliseid indekseid peaks looma palju – üks selline indeks iga unikaalsete väärtustega atribuudi ja keele kombinatsiooni kohta. Autor valis antud juhul näitedisaini loomisel teise variandi (mitteunikaalsus), sest ei soovinud, et kirjeldused oleksid unikaalsed.

„Mustrite kataloogis“ olevates näidetes on EAN-koodi ja kirjelduse väljad ainsad mittekohustuslikud väljad, mis tähendab, et neisse võib väärtuse sisestamata jätta[2].

[1] Varieeruva pikkusega piiranguta tekstiväli [33].

[2] Tühi väärtus on NULL.

Lihtsuse mõttes lubatakse andmebaasi salvestada vaid EAN-13 koodi, mis koosneb 13 numbrist.

Disainide realiseerimisel kasutati kõikide väljade korral, millele on vaja rakendada CHECK-tüüpi kitsendusi, PostgreSQL-i poolt pakutavat domeeni loomise võimalust. Vastasel korral oleks täpselt samad kitsendused iga disaini korral tulnud uuesti deklareerida ehk sama koodi oleks tulnud korrata mitmes kohas. Koodi kordamise vältimiseks loodi andmebaasis eraldi skeem domeenide jaoks, mida näitedisainide skeemid kasutavad.

4 Mustrite kataloog

Järgnevad mustrid on disainilahendused mitmekeelsete andmete hoidmiseks SQL-andmebaasides.

4.1 Tõlkeveerud

Jõud:

- Tulevikus ei ole vaja lisada uutes keeltes andmeid.
- Andmed tõlgitakse vähestesse keeltesse.
- Arvatakse, et üks ühine tabel on lihtsam lahendus.

Lahendus: Luuakse üks tabel, kuhu lisatakse andmeobjekti iga tõlgitava atribuudi korral iga keele kohta üks veerg (Joonis 1).

Allikad ja autorid: Muster on loodud apphp.com veebilehel avaldatud juhendi [26] nimekirjas esimese variandi põhjal, kuid antud mustrit toodi välja ka teistel veebilehtedel ja foorumites [4], [14], [15], [19], [20], [52], [55].

Mustri tugevad küljed:

- Lihtne luua.
- Andmeid mida ei ole vaja tõlkida ei dubleerita.
- Vajaminevat tõlget on lihtne andmebaasist küsida, sest pole vaja koostada tabelite ühendamise päringuid.
- Andmebaasis pole dubleeritud andmeid, vaid iga andmeobjekti kohta on üks rida.
- Igale tõlkeveerule saab määrata seal hoitavale keelele vastava võrdlusreeglistiku, mille alusel toimub sortimine.

Mustri nõrgad küljed:

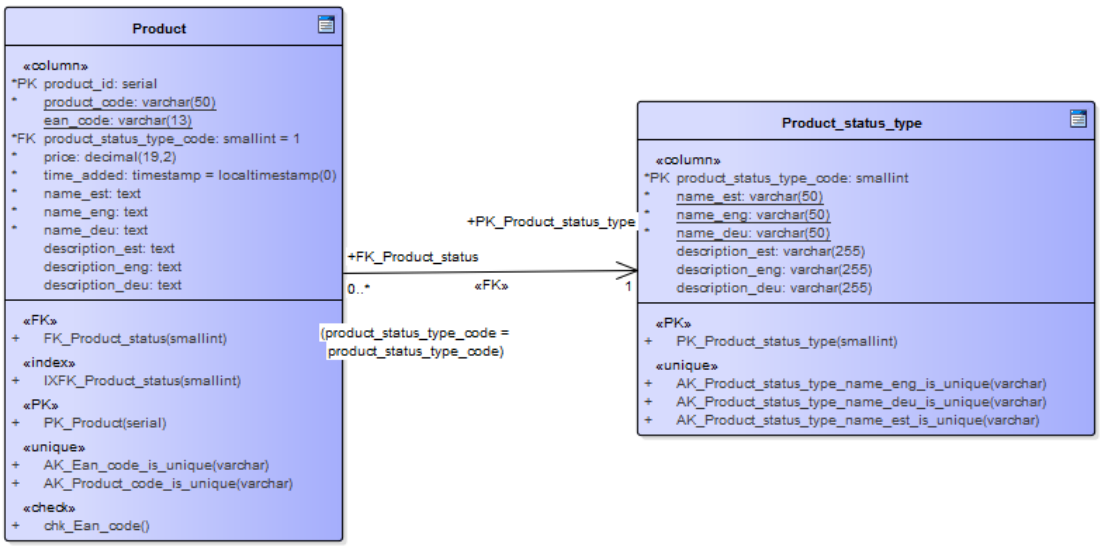
- Raske hallata, kui on palju erinevaid keeli kasutusel ehk palju veerge. Arendaja peab teadma, et andmebaasist tuleb andmeid küsida keeleliitega veeru nime (näiteks kirjeldus_est) järgi.
- Uue keele lisamisel tuleb iga mitmekeelses sisuga tabeli struktuuri muuta.
- Kui kõiki tõlkeid pole vaja, siis hoitakse andmebaasis üleliigset informatsiooni või väljad jäävad tühjaks.
- SQL-andmebaasisüsteemides on tihti rea maht ja veergude arv piiratud (Tabel 1), mis võib antud mustri korral probleemiks osutuda [34].

Tabel 1. Nelja kõige populaarsema SQL-andmebaasisüsteemi [3] maksimaalne rea maht ja veergude arv tabelis.

SQL-andmebaasisüsteem	**Maksimaalne rea maht**	**Maksimaalne veergude arv tabelis**	**Allikas**
Oracle	4 GB	1000	[21]
MySQL	65 535 B	4096	[25]
Microsoft SQL Server 2016	8 060 B	1024 või 30 000[1]	[22]
PostgreSQL	1.6 TB	250–1600	[34]

[1] Maksimaalne veergude arv on 30 000, kui kasutatakse spetsiaalset tüüpi veerge: SPARSE. Sel juhul on väljade, mille väärtus on NULL, andmebaasi mahtu nii optimeeritud, et need ei võtaks andmebaasis ruumi [46]. Laia tabeli (*wide table*) ehk tabeli, mis kasutab SPARSE tüüpi veerge, kasutamine on õigustatud, kui andmebaasis on enamik väljad NULL-id [23].

Näide:



Joonis 1. Andmebaasi disainidiagramm mustrile „Tõlkeveerud

Toote lisamiseks piisab ühest andmete lisamise lausest. Joonis 2 esitab näite ühe toote lisamisest.

```
INSERT INTO language_columns.Product (product_code, ean_code, price,
name_est, name_eng, name_deu, description_est, description_eng) VALUES
	('KPS-KALEV-C', '4740012375470', 1.09, 'Küpsis', 'Cookie',
	'Plätzchen', 'Klassikaline küpsis, mida sobib kasutada küpsisetordi
	valmistamiseks või nautida kohvi kõrvale', 'Classic cookie that is
	suitable for preparing a cookie cake or being enjoyed with a cup of
	coffee.');
```

Joonis 2. Toote lisamise lause.

Kõiki salvestatud tooteid ühes valitud keeles on lihtne andmebaasist küsida, sest kogu info on ühes tabelis. Joonis 4 esitab vastava päringu ja Joonis 4 tulemuse.

```
SELECT product_code, ean_code, price, name_est AS name, description_est AS
description
	FROM language_columns.Product
	ORDER BY name_est;
```

Joonis 3. Päring kõikide toodete eesti keeles leidmiseks.

product_code	ean_code	price	name	description
K-PAULIG-J	6411300158072	4.99	Jahvatatud kohv	Hõrk kauakestva ja rikkaliku järelmaitsega kohv.
K-TERE	4740125110012	0.31	Koor 10%	*NULL*
KT-SOK	4742934010032	4.29	Käsitöö šokolaad	Käsitööna valmistatud luksuslik šokolaad.
KPS-KALEV-C	4740012375470	1.09	Küpsis	Klassikaline küpsis, mida sobib kasutada küpsisetordi valmistamiseks või nautida kohvi kõrvale
SUH-NOR	6414000023138	0.89	Suhkur	Valge suhkur. Sobib küpsetamiseks, hoidiste valmistamiseks, magustoitude tegemiseks ja kohvi sisse.

Joonis 4. Kuvatõmmis kõikide toodete eesti keeles leidmiseks päringu tulemusest.

- Kuna ühes tõlkeveerus on kõik andmed ühes keeles, siis saab juba tabelite loomisel määrata sobiva võrdlusreeglistiku ja tänu sellele ei teki sortimisel probleeme.
- Kuna kirjeldus ei ole kohustuslik väli, siis võivad kõik kirjelduse veerud tühjaks jääda ja seega võib tabelis olla palju NULL-e.

Uue keele lisamisel tuleb mitmekeelse sisuga tabelite struktuuri muuta. Joonis 5 esitab uue keele toe lisamise laused.

```
ALTER TABLE language_columns.Product
       ADD COLUMN name_rus domain.product_name COLLATE "ru_RU",
       ADD COLUMN description_rus domain.product_description COLLATE "ru_RU";

ALTER TABLE language_columns.Product_status_type
       ADD COLUMN name_rus domain.name COLLATE "ru_RU",
       ADD COLUMN description_rus domain.description COLLATE "ru_RU",
       ADD CONSTRAINT AK_Product_status_type_name_rus_is_unique
       UNIQUE(name_rus);
```

Joonis 5. Uue keele toe lisamise laused.

Uutele veergudele ei saa andmebaasi tasemel jõustada kohustuslikkuse (NOT NULL) kitsendust, kui tabelites on juba ridu. Seega kui on soov, et mingisugune tõlge oleks kohustuslik, siis tuleb pärast uue keele tõlgete jaoks mõeldud veeru lisamist see veerg andmetega täita ja alles siis saab veeru kohustuslikuks määrata. Antud näite korral tuleb kõik toote nimetused tõlkida vene keelde ja alles siis on võimalik nimetuse veerule jõustada kohustuslikkuse kitsendus.

4.2 Tõlkeread

Jõud:

- Tulevikus ei ole vaja lisada uutes keeltes andmeid.
- Tõlgitakse vähestesse keeltesse.

Lahendus: Luuakse üks tabel, kuhu lisatakse andmeobjekti iga tõlke kohta üks rida ja luuakse ka keele klassifikaator (Joonis 6).

Allikad ja autorid: Muster on loodud apphp.com veebilehel avaldatud juhendi [26] nimekirjas teise variandi põhjal, kuid antud mustrit toodi välja ka teistel veebilehtedel ja foorumites [19], [55].

Mustri tugevad küljed:

- Lihtne luua.
- Lihtne andmebaasist andmeid küsida.
- Uue keele toe lisamiseks piisab keeleklassifikaatori tabelisse uue rea lisamisest (Joonis 10).

Mustri nõrgad küljed:

- Kui ühes veerus väärtus muutub, siis tuleb ka teistel sama andmeobjekti tõlkeridadel väärtus ära muuta. Näiteks, kui toote hind muutub, siis tuleb hinna väärtust uuendada igal sama toote real.
- Uue keele lisamisel tuleb iga andmeobjekti kohta uus tõlkerida lisada ja kopeerida keelest sõltumatud väärtused (näiteks hind).
- Väga palju duplikaatandmeid keelest sõltumatute andmete näol. Mida rohkem erinevaid keeli on kasutusel, seda rohkem on ka samade andmetega ridu.
- Andmete liiasuse tõttu võivad andmete muutmisel tekkida anomaaliad.

Näide:

Joonis 6. Andmebaasi disainidiagramm mustrile „Tõlkeread“.

Toote lisamiseks kahes keeles tuleb sisestada ühte tabelisse kaks rida. Joonis 7 esitab vastavad andmete sisestamise laused.

```
INSERT INTO language_rows.Product (language_code, product_code, ean_code,
price, name, description) VALUES
	('est', 'KPS-KALEV-C', '4740012375470', 1.09, 'Küpsis', 'Klassikaline
	küpsis, mida sobib kasutada küpsisetordi valmistamiseks või nautida
	kohvi kõrvale');
	('eng', 'KPS-KALEV-C', '4740012375470', 1.09, 'Cookie', 'Classic
	cookie that is suitable for preparing a cookie cake or being enjoyed
	with a cup of coffee.');
```

Joonis 7. Toote lisamise laused.

Toote tõlke lisamisel peab arendaja jälgima, et keelest sõltumatud väärtused (hind, EAN-kood jne) ei erineks sama toote ridadel. Vastasel korral saab andmete terviklikkus rikutud.

Kõiki salvestatud tooteid ühes valitud keeles on lihtne andmebaasist küsida, sest kogu info on ühes tabelis. Joonis 8 esitab vastava päringu ja Joonis 4 tulemuse.

```
SELECT product_code, ean_code, price, name, description
       FROM language_rows.Product
       WHERE language_code = 'est'
       ORDER BY name COLLATE "et_EE";
```

Joonis 8. Päring kõikide toodete eesti keeles leidmiseks

Päringu koostamisel tuleb öelda, millist võrdlusreeglistikku kasutada, sest vastasel korral võidakse andmed järjestada teise keele reeglite järgi. Kui sortida kasvavalt ja kasutades PostgreSQL-i võrdlusreeglit *et_EE.utf8*, on toodete järjekord eesti keele reeglite alusel selline nagu Joonisel 4: koor on enne käsitöö šokolaadi. Kui võrdlusreeglistikuna kasutada *en_US.utf8*, siis oleks inglise keele reeglite alusel käsitöö šokolaad enne koort, kuna inglise keele reeglite järgi on ä-täht üks a-tähe variantidest. Et näide võrdlusreeglistiku olulisusest mitmekeelse andmebaasi korral oleks täielik, esitab Joonis 9 päringu ja kuvatõmmise päringu tulemusest, kus küsitakse eestikeelseid andmeid, kuid sorditakse inglise keele reeglite põhjal.

```
SELECT product_code, ean_code, price, name, description
       FROM language_rows.Product
       WHERE language_code = 'est'
       ORDER BY name COLLATE "en_US";
```

product_code	ean_code	price	name	description
K-PAULIG-J	6411300158072	4.99	Jahvatatud kohv	Hõrk kauakestva ja rikkaliku järelmaitsega kohv.
KT-SOK	4742934010032	4.29	Käsitöö šokolaad	Käsitööna valmistatud luksuslik šokolaad.
K-TERE	4740125110012	0.31	Koor 10%	*NULL*
KPS-KALEV-C	4740012375470	1.09	Küpsis	Klassikaline küpsis, mida sobib kasutada küpsisetordi valmistamiseks või nautida kohvi kõrvale
SUH-NOR	6414000023138	0.89	Suhkur	Valge suhkur. Sobib küpsetamiseks, hoidiste valmistamiseks, magustoitude tegemiseks ja kohvi sisse.

Joonis 9. Päring kõikide toodete eesti keeles küsimiseks ja kuvatõmmis päringu tulemusest. Read on sorditud nimetuse järgi ja inglise keele võrdlusreeglistikku kasutades.

```
INSERT INTO language_rows.Language (language_code, name)
       VALUES ('deu', 'saksa keel');
```

Joonis 10. Uue keele lisamise lause.

Illustreerimaks andmete liiasust selle mustri puhul, küsime tabelist *Product* toote id, toote koodi, nimetuse, keele koodi ja hinna. Kuvatõmmiselt on näha, et hinnad ja toote koodid on tabelis dubleeritud. Joonis 11 esitab vastava päringu ja tulemuse.

```
SELECT product_id, product_code, language_code, name, price
       FROM language_rows.Product
       ORDER BY product_code;
```

product_id	product_code	language_code	name	price
5	K-PAULIG-J	eng	Ground coffee	4.99
1	K-PAULIG-J	est	Jahvatatud kohv	4.99
10	KPS-KALEV-C	eng	Cookie	1.09
9	KPS-KALEV-C	est	Küpsis	1.09
6	K-TERE	eng	Cream 10%	0.31
2	K-TERE	est	Koor 10%	0.31
8	KT-SOK	eng	Handmade chocolate	4.29
4	KT-SOK	est	Käsitöö šokolaad	4.29
3	SUH-NOR	est	Suhkur	0.89
7	SUH-NOR	eng	Sugar	0.89

Joonis 11. Päring kõikide toodete nimetuste ja hinna leidmiseks ning kuvatõmmis tulemusest.

Selleks, et toote hinda, seisundi liiki ja nimetust ühes keeles muuta, tuleb käivitada kaks lauset, millega tabelis *Product* andmeid muuta. Ühe lausega muudetakse toote hind ja seisundi kood kõikidel selle toote ridadel ja teise lausega uuendatakse vastava keeles nimetuse väärtus. Joonis 12 esitab vastavate lausete näite.

```
UPDATE language_rows.Product
       SET price = 0.99,
              product_status_type_code = 2
       WHERE product_code = 'KPS-KALEV-C';
UPDATE language_rows.Product
       SET name = 'Kalevi küpsis'
       WHERE product_code = 'KPS-KALEV-C'
              AND language_code = 'est';
```

Joonis 12. Toote hinna, seisundi liigi ja eestikeelse nimetuse muutmise lause.

Arendaja võib kergesti unustada, et hind ja seisund tuleb muuta kõikidel ridadel ja käivitada vaid sellise lause, mis muudab ära toote nime ühes keeles ning toote seisundi ja hinna vaid selle keele real. Seega antud disaini korral satub ohtu andmete terviklikkus. Kindlam variant oleks proovida andmete terviklikkust tagada luues trigerid ja funktsioonid, mis ühe rea uuendamise korral, uuendab ka teised sama toote read.

4.3 Korduskasutatavad tõlked

Jõud:

- Soovitakse kasutada suurt hulka erinevaid keeli.
- Tulevikus on vaja lisada uusi keeli.

Lahendus: Kõik tõlked on ühes eraldiseisvas tabelis (Joonis 13).

Allikad ja autorid: Muster on loodud apphp.com veebilehel avaldatud juhendi [26] nimekirjas kolmanda variandi põhjal, kuid antud mustrit toodi välja ka teistel veebilehtedel ja foorumites [19], [20], [45].

Mustri tugevad küljed:

- Andmeid mida ei ole vaja tõlkida ei dubleerita.
- Uue keele lisamiseks pole vaja andmebaasi disaini muuta, vaid piisab uue väärtuse lisamisest keeleklassifikaatori tabelisse.
- Kõik tõlked on ühes kohas, mis lihtsustab andmebaasi haldamist.

Mustri nõrgad küljed:

- Andmebaasist andmete küsimiseks tuleb koostada mitu tabelite ühendamise operatsiooni.
- Kõik andmebaasis tehtavad operatsioonid (lisamine, muutmine, kustutamine) on keerulised.
- Kuna kõik tõlked on ühes tabelis, siis tabeli puudumine põhjustab globaalseid probleeme.
- Kuna kõiki tõlkeid hoitakse ühes tabelis ja tuleb teha mitu tabelite ühendamise operatsiooni, siis võivad tekkida jõudlusprobleemid.
- Andmebaasi tasemel on keeruline erinevate väljade tõlgete jaoks kitsendusi jõustada, sest need on kõik salvestatud tõlketabeli ühte veergu.
- Kui mingi tekstilise atribuudi väärtus peab olema mingi olemitüübi puhul unikaalne, siis unikaalsuse kitsenduse deklareerimine tõlketabelile viitavale välisvõtmele seda tegelikult ei taga.

Näide:

Joonis 13. Andmebaasi disainidiagramm mustrile „Korduskasutatavad tõlked“.

Selle mustri puhul on tõlgete kustutamine raskendatud, sest kõikide olemitüüpidele vastavate andmete tõlked salvestatakse ühisesse tabelisse. Seega kui toode kustutada ja on soov kustutada ka tõlked, siis tuleks enne kontrollida, ega ükski teine olem neid samu tõlkeid ei kasuta.

Uue toote lisamiseks tuleb kõigepealt lasta süsteemil genereerida tabelisse *Translation* uute tõlgete identifikaatorid. Seejärel tuleb sisestada tõlked tabelisse *Translation_entry*, kus eelnevalt genereeritud identifikaatori ja keele koodi kombinatsioonid on unikaalsed. See tähendab, et sama identifikaatorit kasutatakse ühe toote ühe atribuudi kõigi tõlgete jaoks. Lisaks peab selles tabelis unikaalne keele ja tõlke kombinatsioon, sest tõlkeid

saab korduvalt kasutada. Viimase sammuna tuleb lisada andmed tabelisse *Product*, milles tõlgitavate andmete veergudesse tuleb lisada vastav identifikaator. Joonis 14 esitab toote lisamise lausete näite.

```
INSERT INTO reusable.Translation (translation_id) VALUES
	(DEFAULT),
	(DEFAULT);
INSERT INTO reusable.Translation_entry (translation_id, language_code, value)
VALUES
	(16, 'est', 'Küpsis'),
	(16, 'eng', 'Cookie'),
	(17, 'est', 'Klassikaline küpsis, mida sobib kasutada küpsisetordi
	valmistamiseks või nautida kohvi kõrvale'),
	(17, 'eng', 'Classic cookie that is suitable for preparing a cookie
	cake or being enjoyed with a cup of coffee.');
INSERT INTO reusable.Product (product_code, ean_code, price, name,
description) VALUES
	('KPS-KALEV-C', '4740012375470', 1.09, 16, 17);
```

Joonis 14. Uue toote lisamise laused.

Joonis 15 näitab, et kõigi toodete eesti keeles küsimiseks tuleb teha mitu tabelite ühendamise operatsiooni.

```
	SELECT product_id, product_code, price, t1.value AS name, t2.value AS
	description
		FROM reusable.Product
		LEFT JOIN reusable.Translation_entry t1
		ON reusable.Product.name = t1.translation_id
			AND t1.language_code = 'est'
		LEFT JOIN reusable.Translation_entry t2
		ON reusable.Product.description = t2.translation_id
			AND t2.language_code = 'est'
		ORDER BY t1.value COLLATE "et_EE";
```

Joonis 15. Päring kõigi toodete eesti keeles küsimiseks.

4.4 Universaalne tõlketabel olemitüübi kohta

Antud muster on üldisem nimetus järgnevatele alam-mustritele. Kõik, mis siin välja tuuakse, kehtivad ka alam-mustrite puhul.

Jõud:

- Soovitakse kasutada suurt hulka erinevaid keeli.
- Tulevikus on vaja lisada uusi keeli.

Lahendus: Iga tabeli kohta, mis tõlgitakse, luuakse lisatabel tõlgetega (Joonis 16, Joonis 19).

Allikad ja autorid: Muster on loodud apphp.com veebilehel avaldatud juhendi [26] nimekirjas neljanda variandi põhjal, kuid antud mustrit toodi välja ka teistel veebilehtedel ja foorumites [4], [15], [19], [20], [52], [55].

Mustri tugevad küljed:

- Andmeid mida ei ole vaja tõlkida ei dubleerita.
- Uue keele lisamiseks pole vaja andmebaasi disaini muuta – piisab kui lisada uus väärtus keeleklassifikaatori tabelisse.
- Veerud säilitavad oma nime (pole vajadust näiteks "_est" järelliite lisamiseks).
- Kõik ühe olemitüübi atribuutide väärtuste tõlked on ühes tabelis, mis lihtsustab andmebaasi haldamist.
- Kuna tõlketabelist põhitabelisse viitavale välisvõtmele saab rakendada *ON DELETE CASCADE*, siis objekti ja tema tõlgete kustutamiseks piisab, kui kustutada rida põhitabelist ja seejärel hoolitseb andmebaasisüsteem selle eest, et ka tõlketabelist saaks vastavad read kustutatud.

Mustri nõrgad küljed:

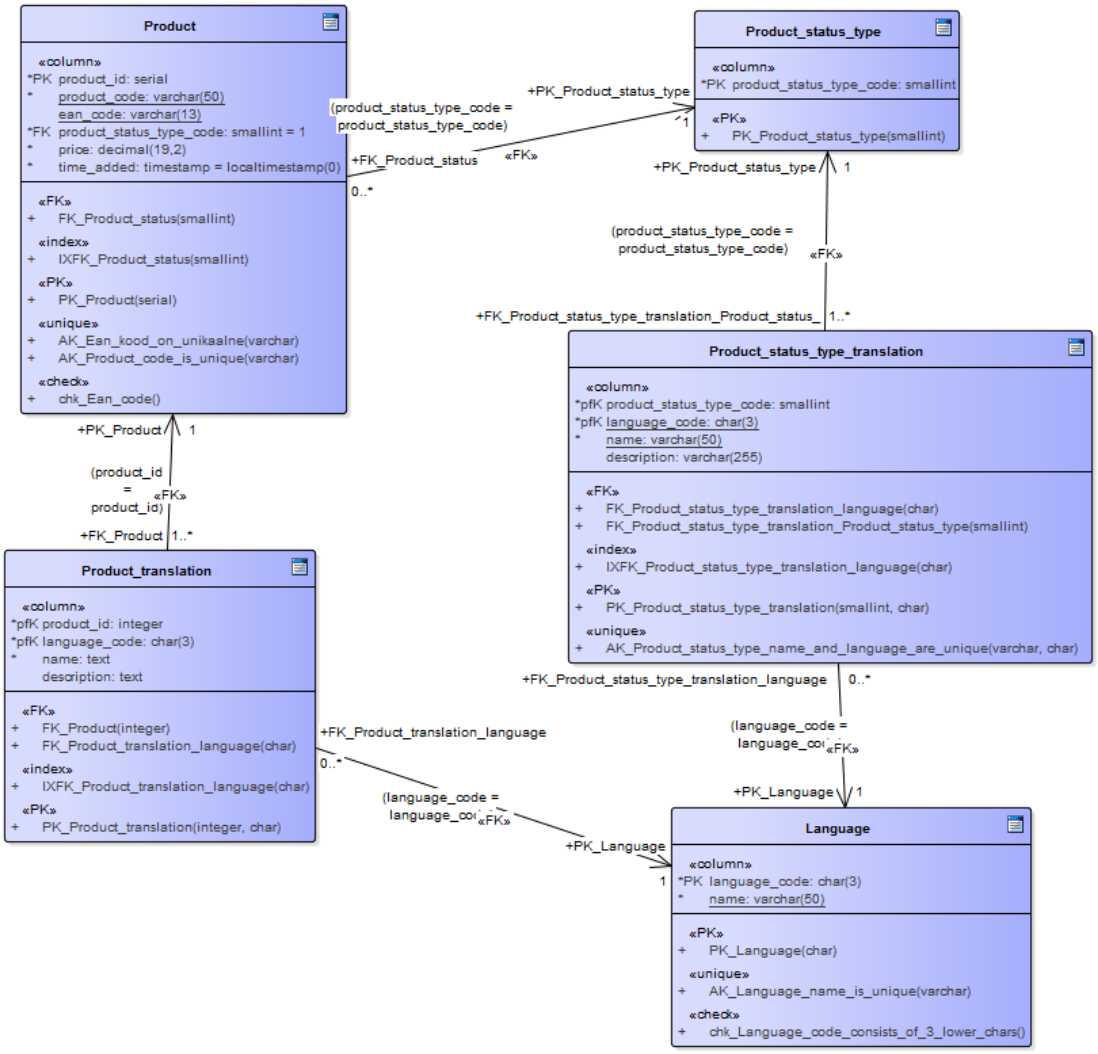
- Andmebaasis hoitavate tabelite arv võib kahekordistuda.
- Andmebaasi tasemel on keeruline jõustada reeglit, et mingi tõlge oleks kohustuslik.

4.4.1 Universaalne tõlketabel olemitüübi kohta

Mustri tugevad küljed:

- Andmebaasipäringute jaoks on vaja kasutada vaid üht tabelite ühendamise operatsiooni.
- Kõrge tasemeni normaliseeritud.

Näide:



Joonis 16. Andmebaasi disainidiagramm mustrile „Universaalne tõlketabel olemitüübi kohta“.

Uue toote lisamiseks tuleb käivitada kaks andmete sisestamise operatsiooni, millest esimesega tuleb lisada keelest sõltumatud andmed tabelisse *Product* ja teisega keelest sõltuvad andmed tabelisse *Product_translation*. Joonis 17 esitab toote lisamise lausete näite.

```
INSERT INTO universal.Product (product_code, ean_code, price) VALUES
      ('KPS-KALEV-C', '4740012375470', 1.09) RETURNING product_id;
INSERT INTO universal.Product_translation (product_id, language_code, name,
description) VALUES
      (5, 'est', 'Küpsis', 'Klassikaline küpsis, mida sobib kasutada
      küpsisetordi valmistamiseks või nautida kohvi kõrvale'),
      (5, 'eng', 'Cookie', 'Classic cookie that is suitable for preparing a
      cookie cake or being enjoyed with a cup of coffee.');
```

Joonis 17. Uue toote lisamise laused.

Joonis 18 näitab, et kõikide toodete ühes keeles küsimiseks tuleb teha üks tabelite ühendamise operatsioon.

```
SELECT t1.product_id, t1.product_code, t1.price, t2.name,
t2.description
      FROM universal.Product AS t1
      LEFT JOIN universal.Product_translation t2
      ON t1.product_id = t2.product_id
            AND t2.language_code = 'est'
      ORDER BY t2.name COLLATE "et_EE";
```

Joonis 18. Päring kõikide salvestatud toodete eesti keeles küsimiseks.

4.4.2 Sisu tüübiga universaalne tõlketabel olemitüübi kohta

Jõud:

- Mitme keele toega andmebaasis ei ole tõlked kohustuslikud ja ei soovita hoida üleliigseid NULL-e.
- Olemitüübil on palju võimalikke mitmekeelseid atribuute ja neid lisatakse pidevalt juurde, kuid enamikke ei väärtustata.

Lahendus: Luuakse klassifikaator sisu tüübi jaoks (Joonis 19).

Mustri tugevad küljed:

- Andmebaasis on vähe NULL-e [7].
- Ühe olemitüübi kohta on võimalik defineerida väga suur hulk erinevat tüüpi atribuute [7].
- Võimaldab olemitüübile uusi atribuute lisada ilma tabeli struktuuri muutmata. See lähenemine on kasulik, kui tabeli loomisel ei ole atribuutide hulk kindlalt teada või kui traditsioonilisel lähenemisel oleks tabelis rohkem NULL-e kui väärtustatud välju [7], [39].

Mustri nõrgad küljed [39]:

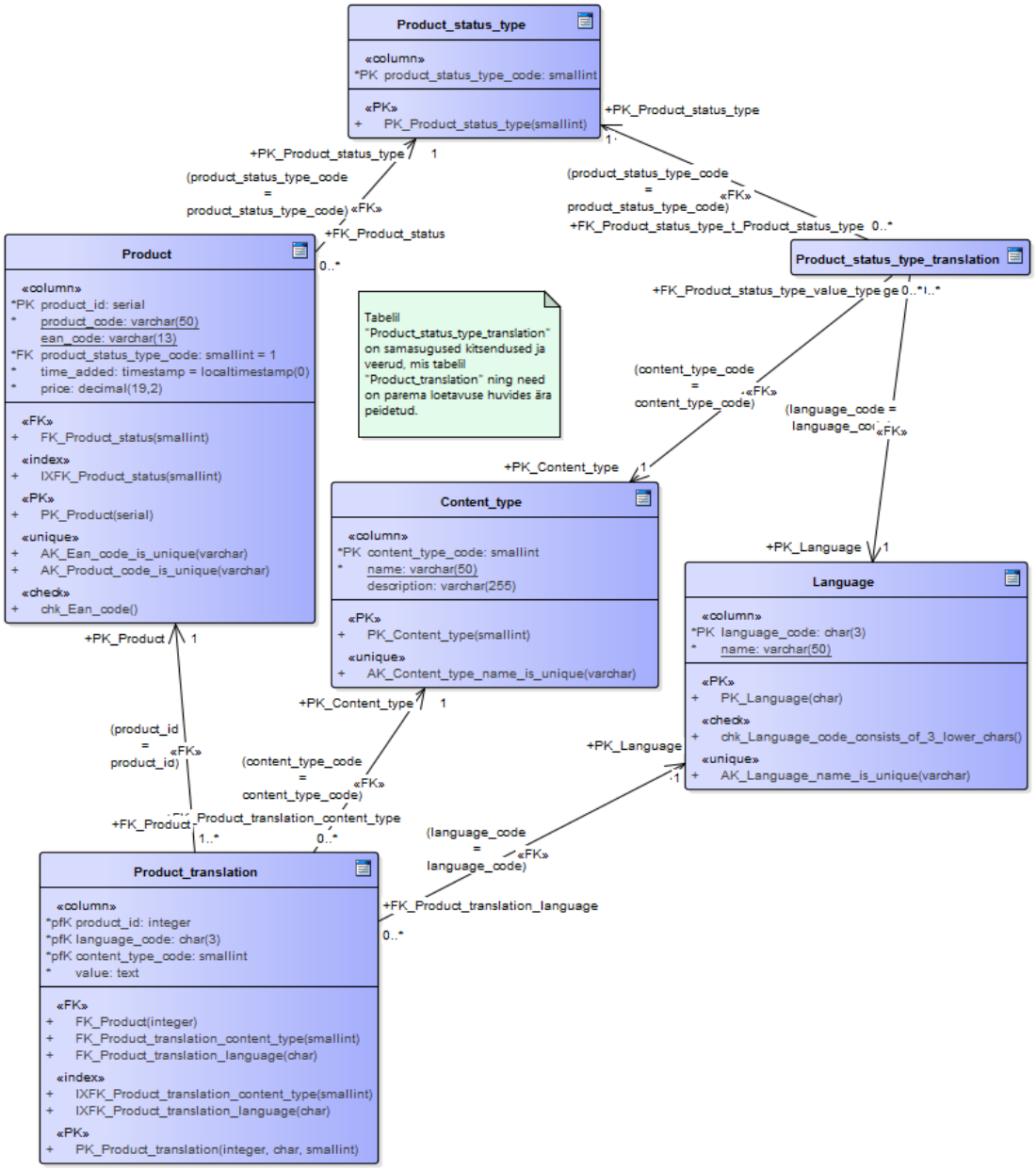
- Erinevat tüüpi väärtuste jaoks kitsenduste loomisel on vaja kasutada implikatsiooni reeglit, mis muudab koodi keerulisemaks. Tabeliga võib olla vaja siduda palju selliseid reegleid.

Kui sisu tüüp on X, siis väärtus peab vastama tingimusele Y.

Binaarset loogilist operatsiooni implikatsioon kasutava avaldise P=>Q võib teisendusreeglite alusel kirjutada ümber samaväärseks avaldiseks: (NOT(P) OR Q). Sellise avaldise saab jõustada tabeliga seotud CHECK kitsenduse abil.

- Erinevate atribuutide väärtused on sama tüüpi (näiteks *text*) veerus.
- Unikaalsuse kitsenduste jõustamine nõuaks suurt hulka osalisi indekseid. Osaline indeks luuakse ainult teatud tingimusele vastavatele ridadele. Selliseid indekseid peaks looma palju – üks selline indeks iga unikaalsete väärtustega atribuudi ja keele kombinatsiooni kohta. Järgneva näite korral, kui eestikeelsed kauba nimetused peaksid olema unikaalsed, siis tuleks luua osaline unikaalne indeks veerule *value* selliste ridade puhul kus keeleks on eesti keel ja sisu tüübiks on kauba nimetus.
- Andmete küsimine on keeruline, kuna ühes veerus hoitakse erinevat tüüpi andmeid ja tuleb teha mitu tabelite ühendamise operatsiooni.

Näide:



Joonis 19. Andmebaasi disainidiagramm mustrile „Sisu tüübiga universaalne tõlketabel olemitüübi kohta“.

Joonis 20 esitab uue toote sisestamise laused.

```
INSERT INTO universal_with_type.Product (product_code, ean_code,
price) VALUES
      ('KPS-KALEV-C', '4740012375470', 1.09) RETURN product_id;
INSERT INTO universal_with_type.Product_translation (product_id,
language_code, content_type_code, value) VALUES
      (5, 'est', 1, 'Küpsis'),
      (5, 'eng', 1, 'Cookie'),
      (5, 'est', 2, 'Klassikaline küpsis, mida sobib kasutada
      küpsisetordi valmistamiseks või nautida kohvi kõrvale.'),
      (5, 'eng', 2, 'Classic cookie that is suitable for preparing a
      cookie cake or being enjoyed with a cup of coffee.');
```

Joonis 20. Toote sisestamise laused.

Joonis 21 esitab päringu kõikide toodete ühes keeles küsimiseks ja Joonis 4 päringu tulemuse.

```
SELECT t1.product_id, t1.product_code, t1.price, t2n.value AS name,
t2d.value AS description
      FROM universal_with_type.Product AS t1
      LEFT JOIN universal_with_type.Product_translation t2n
      ON t2n.product_id = t1.product_id
            AND t2n.language_code='est'
            AND t2n.content_type_code=1
      LEFT JOIN universal_with_type.Product_translation t2d
      ON t2d.product_id = t1.product_id
            AND t2d.language_code='est'
            AND t2d.content_type_code=2
      ORDER BY t2n.value COLLATE "et_EE";
```

Joonis 21. Päring kõikide toodete eesti keeles küsimiseks.

Võrreldes teiste mustritega on selle mustri korral väga keeruline andmeid otsida, sest samast tabelist tuleb korduvalt andmeid küsida ja arendaja peab teadma, millist tüüpi sisu vaja on.

Arvatavasti oleks mõistlik olnud sisu tüübi tabelile lisada veel üks väli, kus hoitakse sisu tüübi omanikku (tabelit). Sellisel juhul oleks paremini ära määratletud, milliseid sisu tüüpe ühe olemitüübi atribuutide väärtuste registreerimisel kasutada võib.

4.5 Keelepõhised tõlketabelid olemitüübi kohta

Jõud: Arvatakse, et erinevas keeles tõlked eraldi tabelitesse salvestada on lihtsam, universaalsem ja kiirem lahendus.

Lahendus: Iga tabeli kohta, mille andmed on vaja tõlkida, luuakse vastava keeleliitega (näiteks Product_est) tabel (Joonis 22, Joonis 23).

Allikad ja autorid: Antud muster on loodud soovituste põhjal, mis on Quora veebilehel välja pakutud vastustena küsimusele andmebaasi korrektse disainimise kohta mitmekeelse süsteemi jaoks [4], kuid antud mustrit pakuti välja ka teistes foorumites [52], [55].

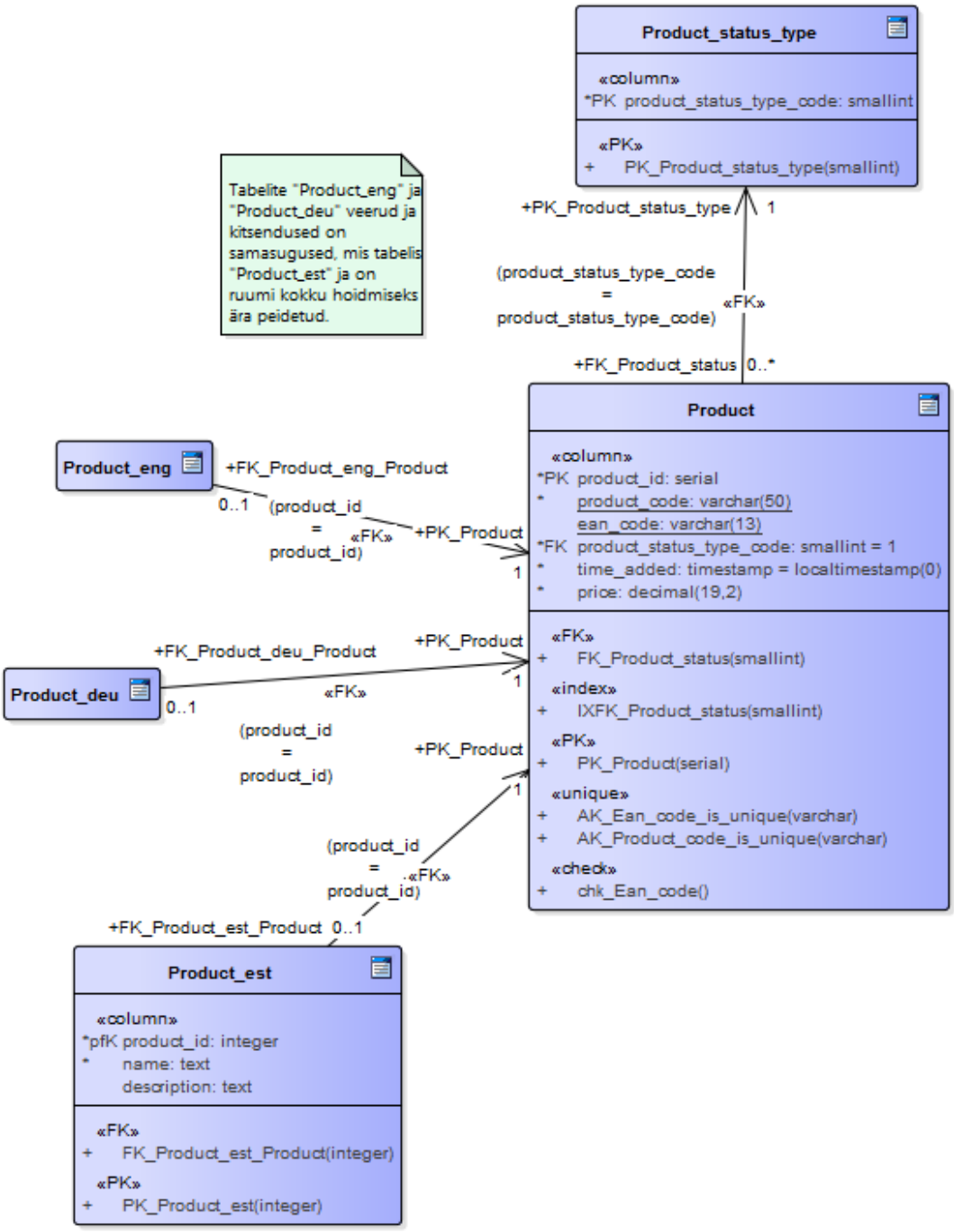
Mustri tugevad küljed:

- Andmeid mida ei ole vaja tõlkida ei dubleerita.

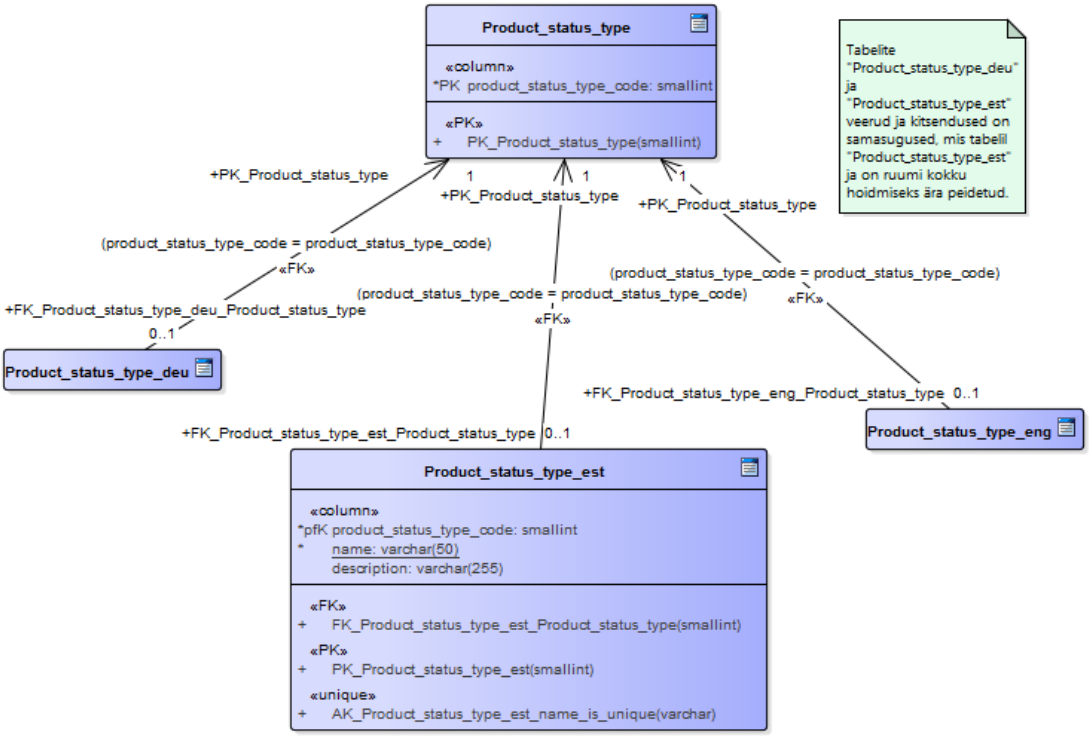
Mustri nõrgad küljed:

- Uue keele lisamiseks tuleb uued tabelid lisada, mille nimedes on vastava keele liide ja mis sisaldavad vastavas keeles andmeid. Andmebaasis hoitavate tabelite arv võib mitmekordistuda. Kui andmebaasis hoitakse kolmes erinevas keeles andmeid, siis tabelite arv võib neljakordistuda, sest ühe objekti jaoks on neli tabelit: üks tabel keelest sõltumatute andmetega ja kolm tabelit vastavates keeltes andmetega.
- Kui tekib vajadus lisada tõlketabelisse uus väli, siis tuleb see lisada kõikidesse sama objekti tõlketabelitesse. Seega arendaja peab jälgima, et kõikide tõlketabelite struktuur saaks uuendatud.
- Andmebaasi tasemel on keeruline jõustada reeglit, et mingis keeles tõlge oleks kohustuslik.

Näide:



Joonis 22. Andmebaasi disainidiagramm mustrile „Keelepõhised tõlketabelid olemitüübi kohta“.



Joonis 23. Andmebaasi disainidiagramm mustrile „Keelepõhised tõlketabelid olemitüübi kohta“.

Uue toote lisamiseks tuleb esmalt lisada keelest sõltumatud väärtused tabelisse *Product* ja seejärel vastavates keeltes väärtused keeletabelitesse. Joonis 24 esitab uue toote lisamise lausete näite.

```
INSERT INTO translation_tables.Product (product_code, ean_code, price) VALUES
      ('KPS-KALEV-C', '4740012375470', 1.09);
INSERT INTO translation_tables.Product_deu (product_id, name) VALUES
      (5, 'Plätzchen');
INSERT INTO translation_tables.Product_eng (product_id, name, description)
VALUES
      (5, 'Cookie', 'Classic cookie that is suitable for preparing a cookie
      cake or being enjoyed with a cup of coffee.');
INSERT INTO translation_tables.Product_est (product_id, name, description)
VALUES
      (5, 'Küpsis', 'Klassikaline küpsis, mida sobib kasutada küpsisetordi
      valmistamiseks või nautida kohvi kõrvale.');
```

Joonis 24. Uue toote lisamise laused.

Joonis 25 esitab päringu kõikide toodete ühes keeles küsimiseks.

```
SELECT t1.product_id, t1.product_code, t1.price, t2.name, t2.description
       FROM translation_tables.Product AS t1
       LEFT JOIN translation_tables.Product_est t2
       ON t1.product_id = t2.product_id
       ORDER BY t2.name;
```

Joonis 25. Päring kõikide toodete eesti keeles küsimiseks.

Kuna tõlketabelitest toote tabelisse viitavale välisvõtmele (näiteks FK_Toode_est_Toode) on rakendatud *ON DELETE CASCADE*, siis toote ja tema tõlgete kustutamiseks piisab, kui kustutada rida tabelist *Product* ja seejärel hoolitseb andmebaasisüsteem selle eest, et ka tõlketabelitest saaks vastavad read kustutatud.

Joonis 26 esitab uue keele toe lisamise näitelaused.

```
CREATE TABLE translation_tables.Product_status_type_rus (
       product_status_type_code smallint NOT NULL,
       name domain.name NOT NULL COLLATE "ru_RU",
       description domain.description COLLATE "ru_RU",
       CONSTRAINT PK_Product_status_type_rus PRIMARY KEY
       (product_status_type_code),
       CONSTRAINT AK_Product_status_type_rus_name_is_unique UNIQUE (name),
       CONSTRAINT FK_Product_status_type_rus_Product_status_type FOREIGN KEY
       (product_status_type_code) REFERENCES
       translation_tables.Product_status_type (product_status_type_code) ON
       DELETE Cascade ON UPDATE Cascade
);

CREATE TABLE translation_tables.Product_rus(
       product_id integer NOT NULL,
       name domain.product_name NOT NULL COLLATE "ru_RU",
       description domain.product_description COLLATE "ru_RU",
       CONSTRAINT PK_Product_rus PRIMARY KEY (product_id),
       CONSTRAINT FK_Product_rus_Product FOREIGN KEY (product_id) REFERENCES
       translation_tables.Product (product_id) ON DELETE Cascade ON UPDATE No
       Action
);
```

Joonis 26. Vene keele toe lisamise laused.

4.6 Keelepõhised tõlketabelid atribuudi kohta

Jõud:

- Andmeobjektidel on vähe mitmekeelseid atribuute
- Soovitakse kasutada palju erinevaid keeli.

Lahendus: Iga atribuudi tõlked salvestatakse eraldi tabelitesse (Joonis 27).

Allikad ja autorid: Antud muster on loodud Microsoft-i foorumis välja pakutud soovituse põhjal [14].

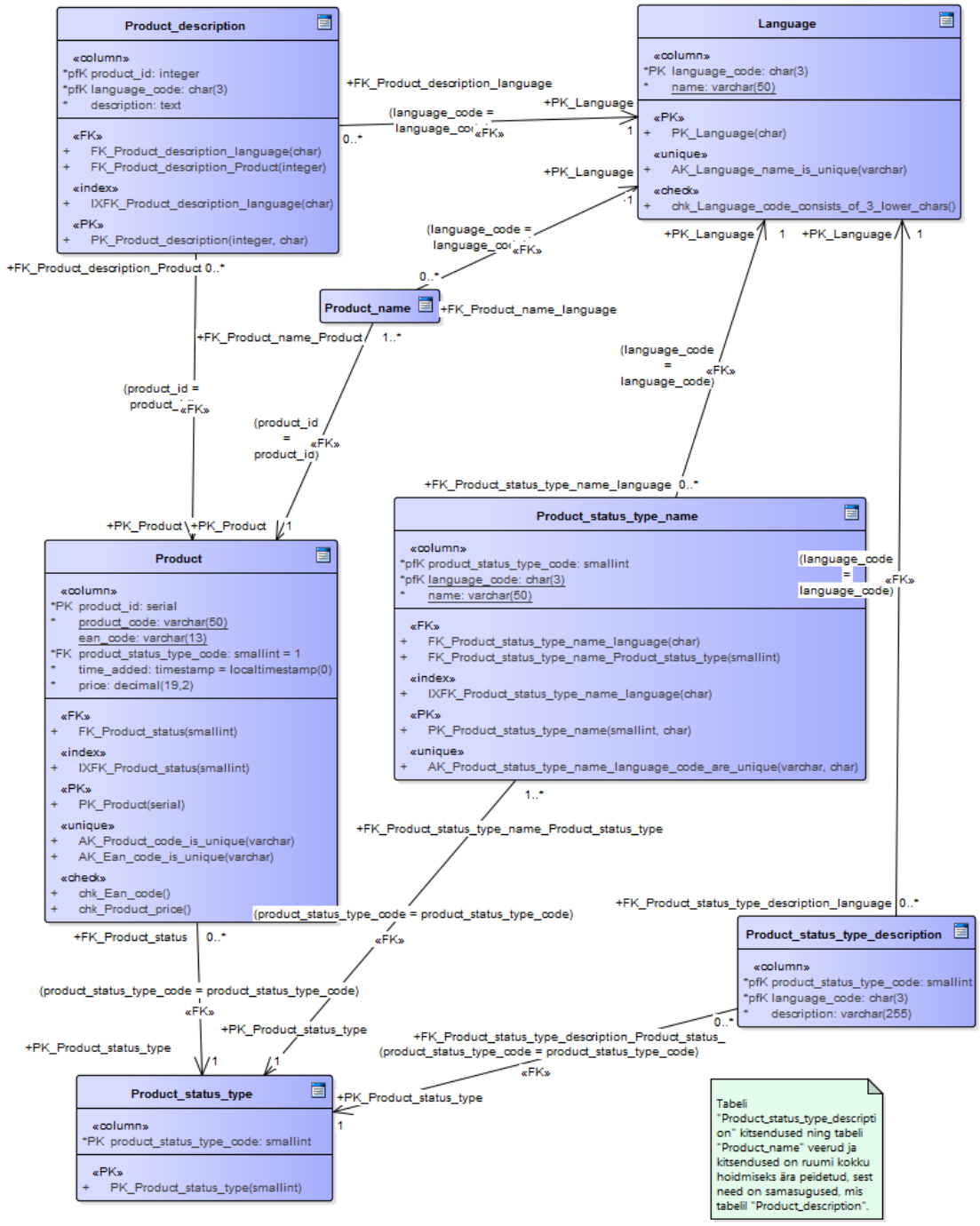
Mustri tugevad küljed:

- Andmeid mida ei ole vaja tõlkida ei dubleerita.
- Uue keele lisamisel pole vaja tabelite struktuuri muuta.
- Andmebaasis ei hoita üleliigseid NULL-e.

Mustri nõrgad küljed:

- Andmebaasi loomine võtab kaua aega, sest tuleb luua suur hulk tabeleid.
- Andmebaasis hoitavate tabelite arv võib mitmekordistuda, sest iga objekti iga mitmekeelse atribuudi jaoks tuleb luua eraldi tabel.
- Andmete küsimiseks tuleb koostada palju tabelite ühendamise operatsioone.

Näide: Kuna tõlketabelitest tabelisse *Product* viitavale välisvõtmele (näiteks FK_Toote_kirjeldus_toode) on rakendatud *ON DELETE CASCADE*, siis toote ja tema tõlgete kustutamiseks piisab, kui kustutada rida tabelist *Product* ja seejärel hoolitseb andmebaasisüsteem selle eest, et ka tõlketabelitest saaks vastavad read kustutatud.



Joonis 27. Andmebaasi disainidiagramm mustrile „Keelepõhised tõlketabelid atribuudi kohta“.

Joonis 28 esitab lausete näited uue toote lisamiseks.

```
INSERT INTO attribute_tables.Product (product_code, ean_code, price) VALUES
	('KPS-KALEV-C', '4740012375470', 1.09);
INSERT INTO attribute_tables.Product_description (product_id, language_code,
description) VALUES
	(5, 'est', 'Klassikaline küpsis, mida sobib kasutada küpsisetordi
	valmistamiseks või nautida kohvi kõrvale.'),
	(5, 'eng', 'Classic cookie that is suitable for preparing a cookie
	cake or being enjoyed with a cup of coffee.');
INSERT INTO attribute_tables.Product_name (product_id, language_code, name)
VALUES
	(5, 'est', 'Küpsis'),
	(5, 'eng', 'Cookie');
```

Joonis 28. Uue toote lisamise laused.

Joonis 29 esitab päringu kõikide toodete ühes keeles küsimiseks.

```
SELECT t.product_id, t.product_code, t.price, n.name, d.description
	FROM attribute_tables.Product AS t
	LEFT JOIN attribute_tables.Product_name n
	ON t.product_id = n.product_id
		AND n.language_code = 'est'
	LEFT JOIN attribute_tables.Product_description d
	ON t.product_id = d.product_id
		AND d.language_code = 'est';
```

Joonis 29. Päring kõikide toodete eesti keeles küsimiseks.

Nagu Jooniselt 29 näha võib, siis toodete küsimiseks tuleb teha mitu tabelite ühendamise operatsiooni. Kui olemitüübil on väga palju atribuute, siis läheb tabelite hulk väga suureks ja sellisel juhul oleks antud mustri kasutamine äärmiselt ebamõistlik, vähemasti juhul kui nende tabelite loomiseks ja haldamiseks ei ole automatiseeritud võimalusi.

5 Lausete keerukuse hindamine

Lausete keerukuse hindamiseks kasutatakse koodiridade arvu metoodikat (*Source Lines of Code*) [27]. Füüsiliste koodiridade kokkuarvestamiseks ei hakatud eritarkvara kasutama, kuna andmete otsimise, lisamise ja andmebaasi loomise keerukust vaadeldi eraldi ning ridu ei olnud märkimisväärselt palju.

SQL-lausete ridadeks jaotamisel võeti eeskuju Holywell'i kirjutatud juhendist [11].

Toodete otsimise laused on iga mustri juures välja toodud. Iga mustri näiteandmebaasi skeemi, tabelite ja indeksite loomise ning andmete lisamise laused leiab Lisast 1. Andmete lisamise puhul lisati kõikide disainide korral tabelitesse samad andmed – kaks keelt, neli toote seisundi liiki ja neli toodet. Uue keele toe lisamise laused on välja toodud mustrite „Tõlkeveerud", „Tõlkeread" ja „Keelepõhised tõlketabelid olemitüübi kohta". Ülejäänud disainide korral on uue keele toe lisamise lause sama, mis mustri „Tõlkeread" korral (ainult skeemi nimi on erinev).

Mustrite „Tõlkeveerud" ja „Keelepõhised tõlketabelid olemitüübi kohta" näitedisainid ja realisatsioonid erinevad teiste mustrite näidetest selle poolest, et neis on kohe kolme keele tugi olemas. Ülejäänud näidetes luuakse vaid kahe keele tugi. Koodiridade kokku lugemisel jäetakse vastavate näidete kolmanda keele tabelid või veerud arvestamata, et keerukuse võrdlemine toimuks sarnastel alustel.

Tabel 2 esitab disainide koodiridade arvud. Mustrid tähistatakse järgnevalt.

A. Tõlkeveerud
B. Tõlkeread
C. Korduskasutatavad tõlked
D. Universaalne tõlketabel olemitüübi kohta
E. Sisu tüübiga universaalne tõlketabel olemitüübi kohta
F. Keelepõhised tõlketabelid olemitüübi kohta
G. Keelepõhised tõlketabelid atribuudi kohta

Iga lausete kategooria (veeru) puhul on **suurim väärtus esitatud rasvaselt** ning väikseim allajoonitult.

Tabel 2. Füüsiliste koodiridade arv näitedisainide korral.

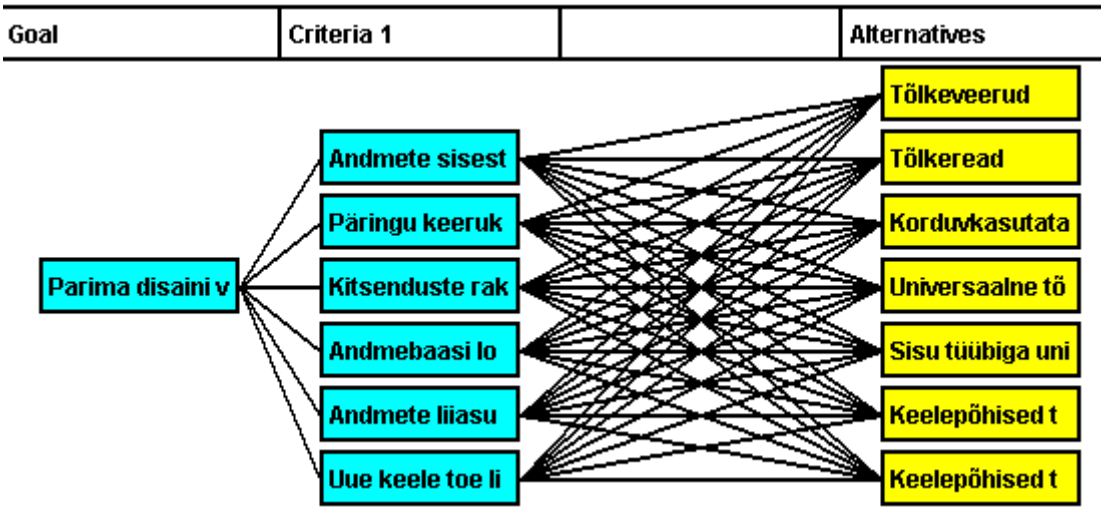
Muster	**Andmebaasi loomine**	**Andmete lisamine**	**Toodete otsimine**	**Keele toe lisamine**	**Kokku**
A	29	10	3	7	49
B	35	21	4	2	62
C	49	**56**	9	2	116
D	45	31	6	2	84
E	55	44	**11**	2	112
F	49	30	5	**15**	99
G	**61**	43	8	2	114

Ära tuleb mainida, et andmebaasi loomise keerukuse hindamisel jäeti domeenide lisamise laused arvestamata. Objektiivsema hinnangu saamiseks oleks tulnud domeenide koodiridade arv arvesse võtta selle järgi, milliseid domeene vastav disain kasutab.

6 Parima disaini valimine kasutades Saaty meetodit

Saaty meetodit ehk analüütiliste hierarhiate meetodit kasutatakse subjektiivsete hinnangute alusel alternatiivide vahel parima valiku tegemisel [51]. Käesolevas töös kasutatakse võrdluse tegemiseks tasuta tarkvara Web-HIPRE [56].

Selleks, et Saaty meetodit kasutada, tuleb kõigepealt leida eesmärk, mõjurid ehk kriteeriumid ja valikud ehk alternatiivid (Joonis 30). Antud töös on eesmärgiks parima disaini valimine mitmekeelsete andmete andmebaasis hoidmiseks toodete halduse korral. Mõjuriteks on andmete sisestamise keerukus, päringute keerukus, uue keele toe lisamise keerukus, andmebaasi loomise keerukus, andmete liiasuse olemasolu ning kitsenduste ja andmetüübi määramise keerukus. Valikuteks on „Mustrite kataloogis“ esitatud näitedisainid. Mõjureid tuleb võrrelda paarikaupa, et leida nende suhteline olulisus. Seejärel tuleb lõppotsuse teadasaamiseks võrrelda iga mõjuri korral kõiki valikuid omavahel paarikaupa.



Joonis 30. Parima disaini valimise analüütilise hierarhiate meetodi otsustusmudel.

6.1 Mõjurite paarikaupa võrdlemine

Saaty otsustusmudel kasutab otsustuse jaoks sõnalisi suhtelisi hinnanguid: võrdtähtis, mõõdukas paremus, oluline paremus, väga tugev paremus ja äärmuslik paremus, mida tähistatakse numbritega vastavalt 1, 3, 5, 7 ja 9. Kõik vahepealsed arvud (2, 4, 6, 8)

tähistavad ka vahepealseid hinnanguid. Kõik käesolevas töös antud hinnangud anti ainult autori enese poolt.

Tabel 4 esitab mõjurite omavahelise võrdluse ning seal on mõjurid tähistatud nagu Tabelis 3.

Tabel 3. Mõjurite ja valikute tähistused tabelis esitamiseks.

Mõjurid	**Valikud**
A. Andmete liiasuse olemasolu B. Päringute keerukus C. Uue keele toe lisamise keerukus D. Andmebaasi loomise keerukus E. Tõlkeid sisaldavatele veergudele kitsenduste rakendamise ja andmetüübi määramise keerukus F. Andmete sisestamise keerukus	G. Tõlkeveerud H. Tõlkeread I. Korduskasutatavad tõlked J. Universaalne tõlketabel olemitüübi kohta K. Sisu tüübiga universaalne tõlketabel olemitüübi kohta L. Keelepõhised tõlketabelid olemitüübi kohta M. Keelepõhised tõlketabelid atribuudi kohta

Tabel 4. Mõjurite omavahelise võrdluse tabel.

Mõjur	**A**	**B**	**C**	**D**	**E**	**F**	**Kaalud**
A	1.00	3.00	0.50	8.00	2.00	4.00	**0.257**
B	0.33	1.00	0.25	4.00	0.50	2.00	**0.100**
C	2.00	4.00	1.00	9.00	3.00	5.00	**0.389**
D	0.13	0.25	0.11	1.00	0.17	0.50	**0.031**
E	0.50	2.00	0.33	6.00	1.00	3.00	**0.162**
F	0.25	0.50	0.20	2.00	0.33	1.00	**0.061**

Uue keele toe lisamise keerukus (C) on kõige olulisem, sest antud mustrite puhul näitab see tegelikult, kas keele toe lisamiseks muudetakse andmebaasi struktuuri. Olulisuselt teisel kohal on andmete liiasuse olemasolu (A), sest andmete liiasuse korral võivad andmete muutmisel tekkida anomaaliad ja seda tuleks vältida. Kolmas autori arvates

olulisim mõjur on tõlkeid sisaldavatele veergudele kitsenduste rakendamise ja andmetüübi määramise keerukus (E).

Päringute keerukust (B) on hinnatud veidi olulisemaks, kui andmete sisestamise keerukust (F), sest suure tõenäosusega tehakse seda operatsiooni rohkem.

Andmebaasi loomise keerukust (D) on hinnatud kõige vähem oluliseks.

6.2 Valikute paarikaupa võrdlemine mõjurite suhtes

Valikute paarikaupa võrdlemise otsustustabelid iga mõjuri suhtes on leitavad Lisast 2. Tabel 5 on esitatud kokkuvõtlikult osakaalud, kus mõjurid (kriteeriumid) ja disainimustrid (alternatiivid) on tähistatud nagu Tabelis 3.

Tabel 5. Kokkuvõte mõjurite osakaaludest iga valiku suhtes.

	G	**H**	**I**	**J**	**K**	**L**	**M**
A	0.161	0.032	0.161	0.161	0.161	0.161	0.161
B	0.302	0.237	0.069	0.132	0.030	0.164	0.065
C	0.032	0.190	0.190	0.190	0.190	0.019	0.190
D	0.388	0.222	0.089	0.120	0.052	0.093	0.036
E	0.189	0.189	0.027	0.189	0.027	0.189	0.189
F	0.403	0.191	0.041	0.115	0.067	0.115	0.067

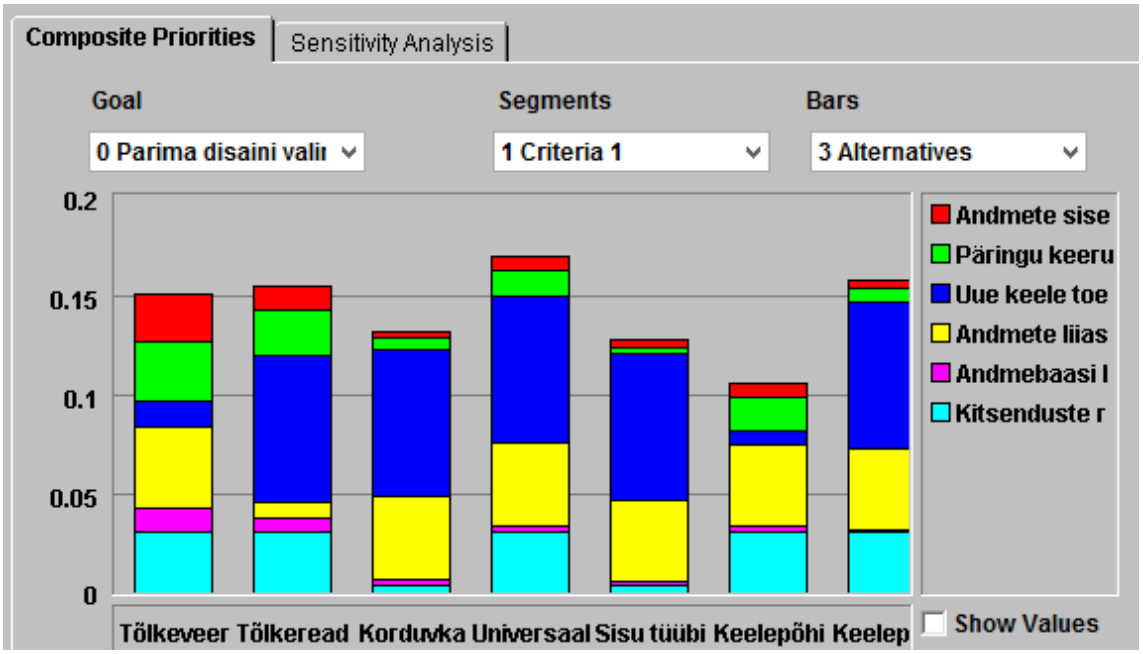
6.3 Tulemuste analüüs

Tabel 6 esitab iga valiku osakaalu ehk suhtelise headuse püstitatud eesmärgi täitmiseks, kus mõjurid ja mustrid on tähistatud vastavalt Tabelis 3 esitatud tähistustele.

Tabel 6. Saaty meetodi rakendamise lõpptulemus.

	G	H	I	J	K	L	M
A	0.041	0.008	0.041	0.041	0.041	0.041	0.041
B	0.030	0.024	0.007	0.013	0.003	0.016	0.007
C	0.012	0.074	0.074	0.074	0.074	0.007	0.074
D	0.012	0.007	0.003	0.004	0.002	0.003	0.001
E	0.031	0.031	0.004	0.031	0.004	0.031	0.031
F	0.024	0.012	0.002	0.007	0.004	0.007	0.004
Kokku	**0.151**	**0.155**	**0.132**	**0.170**	**0.128**	**0.106**	**0.158**

Joonis 31 esitab otsustusanalüüsi tulemuse graafiliselt.

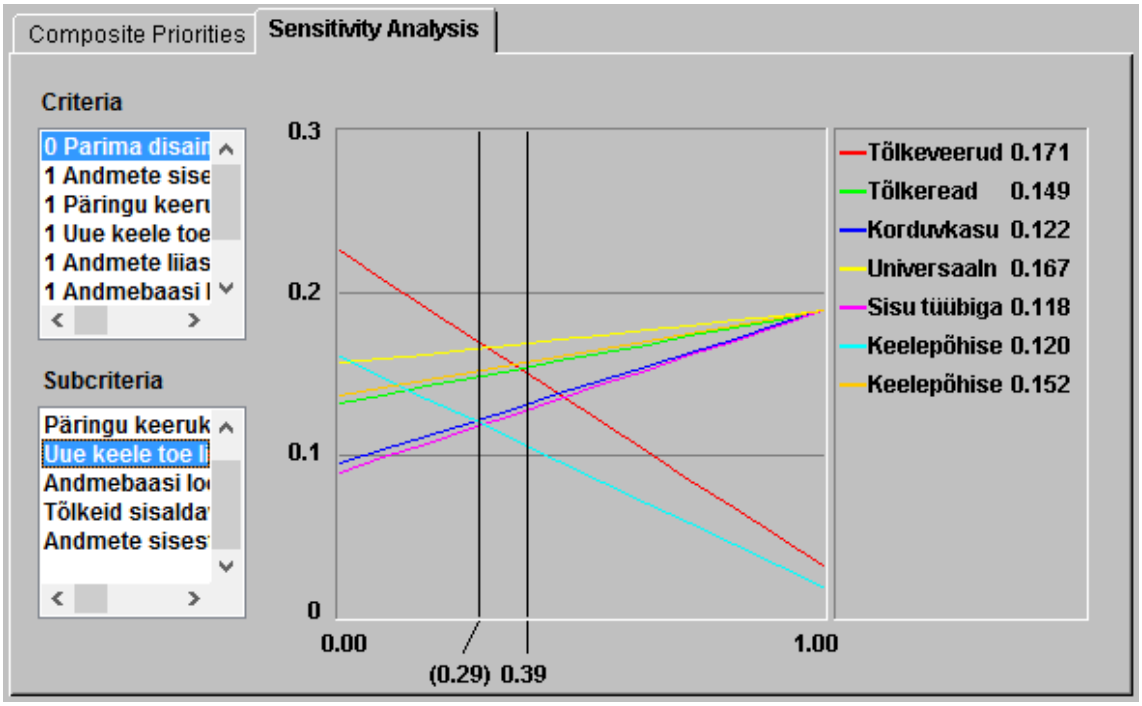


Joonis 31. Saaty meetodi rakendamise tulemus Web-HIPRE tarkvaraga.

Valitud mõjureid ja nende olulisust autori jaoks arvesse võttes selgus, et antud juhul on parimaks disainiks „Universaalne tõlketabel olemitüübi kohta“. Selle disaini esimeseks saamise puhul oli määravaks, et autori poolt kolm kõige tähtsamat mõjurit, said kõrge hinnangu ja tähtsuselt järgmised mõjurid ei olnud kõige kehvema hinnanguga. Jooniselt

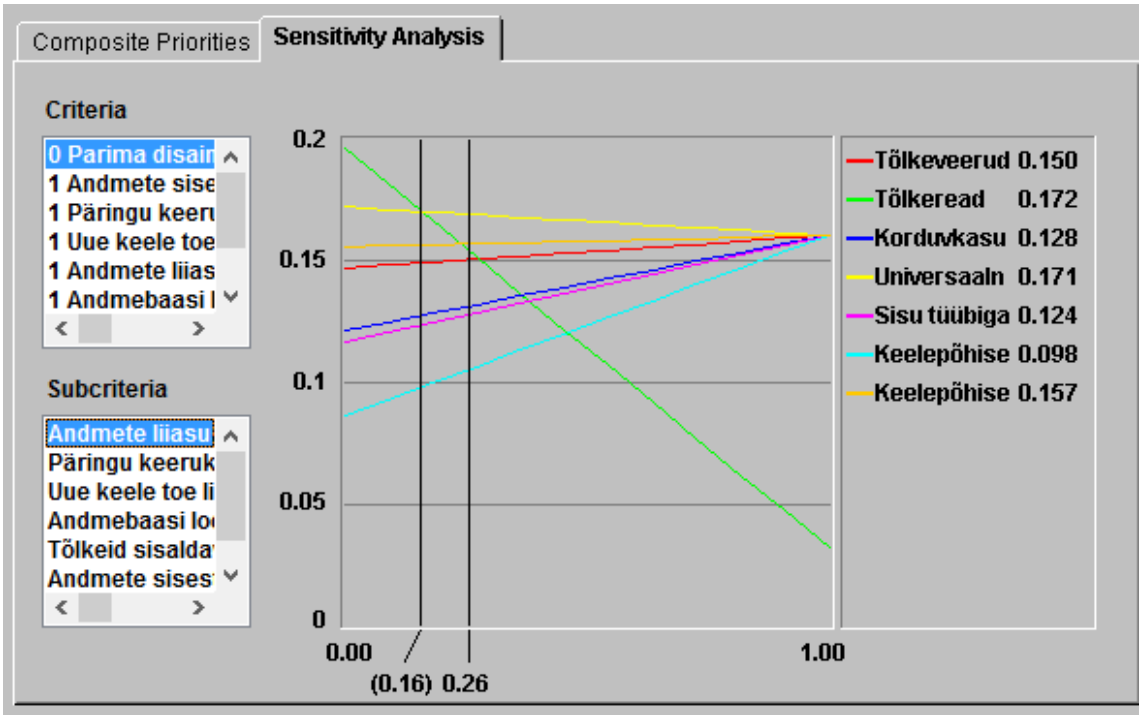
31 on näha, et mustrid „Tõlkeread“, „Tõlkeveerud“ ja „Keelepõhised tõlketabelid atribuudi kohta“ ei jää kokkuvõttes väga palju maha.

Mustri „Tõlkeveerud“ kasuks räägib päringute koostamise, andmete lisamise ja andmebaasi loomise lihtsus. Tundlikkuse analüüs näitab, et kui tõsta päringu keerukuse olulisuse osakaalu 0.21-ni või andmete sisestamise osakaalu 0.13-ni, siis oleks mustri „Tõlkeveerud“ näitedisain parim. „Tõlkeveergude“ mustri põhjal loodud disaini suur puudujääk on uue keele toe lisamise keerukus, kuid kui selle mõjuri osakaalu vähendada 0.29-ni, siis tõuseks see disain pingereas esimeseks (Joonis 32).



Joonis 32. Tundlikkuse analüüs uue keele toe lisamise mõjuri suhtes.

Mustri „Tõlkeread“ suureks puuduseks on andmete liiasuse olemasolu. Tundlikkuse analüüs näitab, et kui andmete liiasuse olemasolu tähtsuse osakaalu vähendada 0.16-ni, siis oleks tõlkeridade muster antud tingimustel parim (Joonis 33).



Joonis 33. Tundlikkuse analüüs andmete liiasuse mõjuri suhtes.

Nagu näha võib, siis lõpptulemus on subjektiivsete hinnangute muutmise suhtes üsna tundlik. Otsustamisel oleks võinud mõjurina arvestada ka mittekohustuslike veergude osakaalu iga disaini korral, sest see mõjutab NULL-ide hulka tabelites ja võib disaini valikul oluliseks parameetriks osutuda.

Nagu peatükis „Lausete keerukuse hindamine“ mainiti, siis andmebaasi loomise keerukuse hindamisel jäeti domeenid arvestamata, kuid kuna andmebaaside loomise keerukuse tähtsus oli kõige madalamaks valitud, siis otsustusmudeli lõpptulemust ei oleks see arvatavasti mõjutanud.

7 Näiterakendus

Piiratud funktsionaalsusega näiterakendus realiseeritakse, et hinnata Saaty meetodi abil peatükis 6 välja valitud disaini (*Universaalne tõlketabel olemitüübi kohta*) headust ja demonstreerida mitmekeelse süsteemi toimimist.

Kuna käesolevas töös keskendutakse lokaliseerimise ühele osale, milleks on andmete ja kasutajaliidese mitmekeelsus, siis ei hakata rakenduses realiseerima teisi lokaliseerimise osi, mille hulka kuuluvad regioonipõhised erisused: numbrite formaat[1], kuupäeva formaat, erinevad ühikud, valuutad jne. Nende kasutuselevõtmise jaoks tuleks defineerida erinevad regioonid ning määrata ära, millised formaadid, valuutad ja muu seda regiooni iseloomustavad ning vastavalt kasutaja regioonile talle õiges formaadis andmeid kuvada. Teine variant on muuta rakendus kasutaja poolt seadistatavaks, et ta saaks ise valida, milliseid formaate, ühikuid ja muud ta kasutada soovib. Regioonipõhiste erinevuste defineerimiseks tuleks luua vastavad konfiguratsioonifailid.

7.1 Kuidas väljendada mitmekeelsuse nõuet analüüsi mudelites?

Funktsionaalne nõue on miski, mida süsteem tegema peab. Näiteks toote lisamise funktsioon.

Mittefunktsionaalne nõue on süsteemi omadus ja defineerib, milline süsteem olema peaks. Näiteks toetab PostgreSQL andmebaasisüsteemi.

Töö autori arvamus on, et mitmekeelsust saab arvestada nii funktsionaalse kui mittefunktsionaalse nõudena.

Mitmekeelsus kui funktsionaalne nõue: kasutaja peab saama vahetada rakenduse keelt valides sobiva keele ja vastavalt sellele rakenduse keel ja rakenduses esitatavate andmete keel muutub.

[1] Punkti, koma ja ülakoma kasutamine arvudes.

Mitmekeelsus kui mittefunktsionaalne nõue: süsteem peab toetama mitut keelt. See tähendab, et vastavad tõlked on defineeritud ja süsteemi saab nendes keeltes kasutada. Valitud keel ei mõjuta rakenduse funktsionaalsust.

„Ainult üks kord“ disainiprintsiibist [28] lähtuvalt tuleks nõuete kirjelduses dubleerimist vältida. Mittefunktsionaalsete nõuete all on laiendatavuse alajaotuses vaja kindlasti välja tuua see, kui süsteemi peab olema võimalik lisada uute keelte tuge. Mis puudutab loodava süsteemi funktsionaalsust, siis kui mitmekeelsus on seotud pärisalamhulgaga süsteemi funktsionaalsusest, siis võiks kirjeldada seda pigem funktsionaalse nõudena. Kui mitmekeelsuse nõue kehtib üldiselt kogu funktsionaalsuse puhul, siis võib seda kirjeldada selle asemel mittefunktsionaalse nõudena („tõsta sulgude ette“). Jaotis 7.2 eksib näite huvides „ainult üks kord“ printsiibi vastu ning pakub välja ühe võimaluse, kuidas esitada nõue keelte kohta nii kasutusjuhtude mudeli (funktsionaalsed nõuded) kui mittefunktsionaalsete nõuete korral. Ka funktsionaalsete nõuete puhul on tõstetud keele valimine eraldi kasutusjuhtudesse, et vähendada dubleerimist ning lihtsustada muudatuste tegemist.

7.2 Analüüs

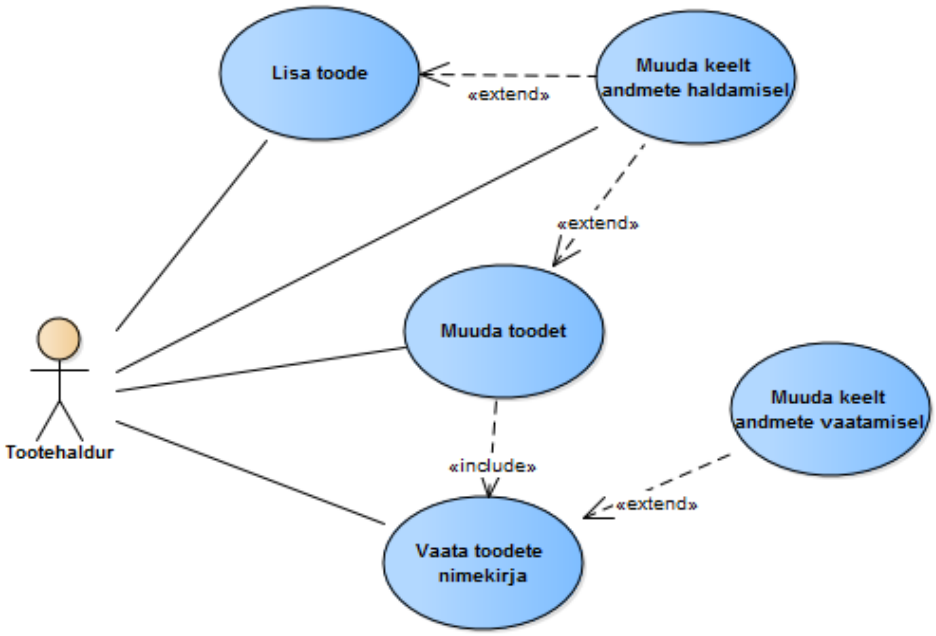
Järgnevalt esitatakse ülevaade toodete halduse süsteemi alamosast, mis peab võimaldama uusi tooteid lisada, olemasolevaid tooteid muuta ja toodete nimekirja vaadata. Kasutajaliidese keelt peab olema võimalik muuta ja tooteid peab saama salvestada erinevates keeltes.

Kasutaja tuvastamist ei realiseerita, sest eesmärgiks on katsetada valitud andmebaasi disaini, mitte realiseerida valmisrakendust.

7.2.1 Kasutusjuhtude mudel

Joonis 34 esitab toodete halduse süsteemi alamosa kasutusjuhtude mudeli, kus tegutsejaks on tootehaldur.

Kuna iga kasutusjuhu korral on primaarseks tegutsejaks tootehaldur, siis ei hakata seda kasutusjuhtude juures eraldi välja tooma.



Joonis 34. Toodete halduse süsteemi alamosa kasutusjuhtude mudel

Järgnevalt esitatakse kasutusjuhtude tekstilised kirjeldused laiendatud formaadis:

Kasutusjuht: Lisa toode

Osapooled ja nende huvid: Tootehaldur soovib lisada uue toote. Kliendid soovivad toote kirjeldusi endale arusaadavas keeles.

Käivitav sündmus: Tootehaldur soovib lisada toote, sest sortimenti tuli uus toode.

Eeltingimused: Teenus on töökorras.

Järeltingimused: Toote andmed salvestati andmebaasi.

Stsenaarium:

1. Tootehaldur avaldab soovi lisada uut toodet.
2. Süsteem kuvab toote sisestamise vormi, kus on muuhulgas sisestusväljad erinevates keeltes olevate tekstide jaoks. [Muuda keelt andmete haldamisel]
3. Tootehaldur täidab vormi ja annab korralduse salvestada.
4. Süsteem salvestab toote andmed.

Laiendused (või alternatiivne sündmuste käik):

4a. Kui sisestatud andmed ei vasta nõuetele, siis toodet ei salvestata.

1. Süsteem kuvab veateate.

Kasutusjuht: Muuda toodet

Osapooled ja nende huvid: Tootehaldur soovib, et andmebaasis oleks kõige värskem toote info. Kliendid soovivad, et info toodete kohta oleks ajakohane.

Käivitav sündmus: Toodete haldur soovib muuta toote infot, sest see on vigaselt sisestatud, aegunud või vaja uude keelde tõlkida.

Eeltingimused: Toode on andmebaasis olemas.

Järeltingimused: Toote andmed uuendati andmebaasis.

Stsenaarium:

1. Käivitub kasutusjuht „Vaata toodete nimekirja“.
2. Tootehaldur valib toodete nimekirjast toote, mida ta soovib muuta.
3. Süsteem kuvab toote muutmise vormi, kus on muuhulgas sisestusväljad erinevates keeltes olevate tekstide jaoks. [Muuda keelt andmete haldamisel]
4. Tootehaldur muudab toote andmeid ja annab korralduse salvestada.
5. Süsteem salvestab toote andmed.

Laiendused (või alternatiivne sündmuste käik):

6a. Kui sisestatud andmed ei vasta nõuetele, siis toote andmeid ei salvestata.

1. Süsteem kuvab veateate.

Kasutusjuht: Vaata toodete nimekirja

Osapooled ja nende huvid: Toodete haldur soovib vaadata toodete nimekirja. Kliendid soovivad toote kirjeldusi endale arusaadavas keeles.

Käivitav sündmus: Toodete haldur soovib saada ülevaadet lisatud toodetest.

Eeltingimused: Toodete andmed on registreeritud.

Järeltingimused: On leitud kõik tooted kasutajaliidesega samas keeles.

Stsenaarium:

1. Tootehaldur avaldab soovi vaadata toodete nimekirja.
2. Süsteem kuvab toodete nimekirja [Muuda keelt andmete vaatamisel].

Laiendused (või alternatiivne sündmuste käik):

2a. Kui süsteemis ei ole ühtegi toodet registreeritud, siis süsteem ei kuva toodete andmeid.

Kasutusjuht: Muuda keelt andmete vaatamisel

Osapooled ja nende huvid: Tootehaldur soovib süsteemi kasutada endale sobivas keeles.

Laienduspunktid: Muuda keelt andmete vaatamisel.

Eeltingimused: Süsteem toetab mitut keelt.

Järeltingimused: Kasutajaliides muutus valitud keelde. Andmed on ainult valitud keeles.

Stsenaarium:

1. Tootehaldur valib endale sobiva keele.
2. Süsteem kuvab kasutajaliidest ja vaadatavaid andmeid ainult kasutaja valitud keeles.

Laiendused (või alternatiivne sündmuste käik):

1a. Kui nimekirjas ei ole soovitud keelt, siis ei saa kasutajaliidese keelt sellesse keelde muuta.

Kasutusjuht: Muuda keelt andmete haldamisel

Osapooled ja nende huvid: Tootehaldur soovib süsteemi kasutada endale sobivas keeles.

Laienduspunktid: Muuda keelt andmete haldamisel.

Eeltingimused: Süsteem toetab mitut keelt.

Järeltingimused: Kasutajaliides muutus valitud keelde.

Stsenaarium:

1. Tootehaldur valib endale sobiva keele.
2. Süsteem kuvab kasutajaliidest kasutaja valitud keeles. Tekstilisi andmeid kuvatakse ja saab muuta kõigi keelte korral. Vastavate väljade juures on valitud keeles kirjas, millised andmed on seal esitatud. Välja kirjelduses on viide keelele.

Laiendused (või alternatiivne sündmuste käik):

1a. Kui nimekirjas ei ole soovitud keelt, siis ei saa kasutajaliidese keelt sellesse keelde muuta.

7.2.2 Mittefunktsionaalsed nõuded

Tabel 7 esitab toodete halduse alamsüsteemi mittefunktsionaalsed nõuded.

Tabel 7. Mittefunktsionaalsed nõuded

Tüüp	Nõude kirjeldus
Andmebaasisüsteem	Süsteem peab andmete hoidmiseks kasutama PostgreSQL andmebaasisüsteemi.
Arendusvahendid	Süsteemi serveripoolne osa peab olema kirjutatud kasutades PHP programmeerimiskeelt.
Kasutajaliides	Süsteemil peab olema veebipõhine kasutajaliides.
Keel	Veebirakendus peab olema eesti ja inglise keeles. Andmebaasis peavad olema andmed eesti ja inglise keeles.
Laiendatavus	Rakendus peab olema ehitatud nii, et see oleks võimalikult lihtsasti laiendatav funktsionaalsuse lisandumisel (kaasa arvatud uue keele toe lisamisel).

7.3 Disain

Järgnevalt tutvustatakse rakenduse struktuuri ja kasutajaliidese disaini ning kirjeldatakse kasutatud disainimustreid. Samuti uuritakse, millised lahendused mitmekeelse rakenduse loomiseks juba olemas on.

7.3.1 Parimad praktikad

Parimatest praktikatest kasutatakse rakenduse loomisel MVC disainimustrit, sõltuvuste sisestuse printsiipi (*Dependency Injection*), pärimist (*Inheritance*), komposiitmusrit (*Composite View*).

MVC disainimustri [12] korral jagatakse rakendus kolmeks osaks, millest igaühel on oma ülesanne:

- Mudeli (*Model*) klass tegeleb andmete haldamisega. Mudel võtab andmeid kontrollerilt vastu, suhtleb andmebaasiga ja edastab andmeid vaatele.
- Vaate (*View*) klass tegeleb andmete kasutajale välja kuvamisega. Andmed küsib vaade otse mudelilt.
- Kontrolleri (*Controller*) klass võtab andmeid vastu ja edastab mudeli klassile.

Sõltuvuste sisestuse printsiip tähendab, et programmikoodist eemaldatakse püsiprogrammeeritud (*hard-coded*) sõltuvused (Joonis 35) ning antakse programmile hoopis töö käigus ette (Joonis 36).

```
class Controller {
	private $model;
	public function __construct() {
		$this->model = new Model();
	}
}
class Model {}
```

Joonis 35. Püsiprogrammeeritud sõltuvus *Model* klassist.

```
class Controller {
	private $model;
	public function __construct(Model $model) {
		$this->model = $model;
	}
}
class Model {}
```

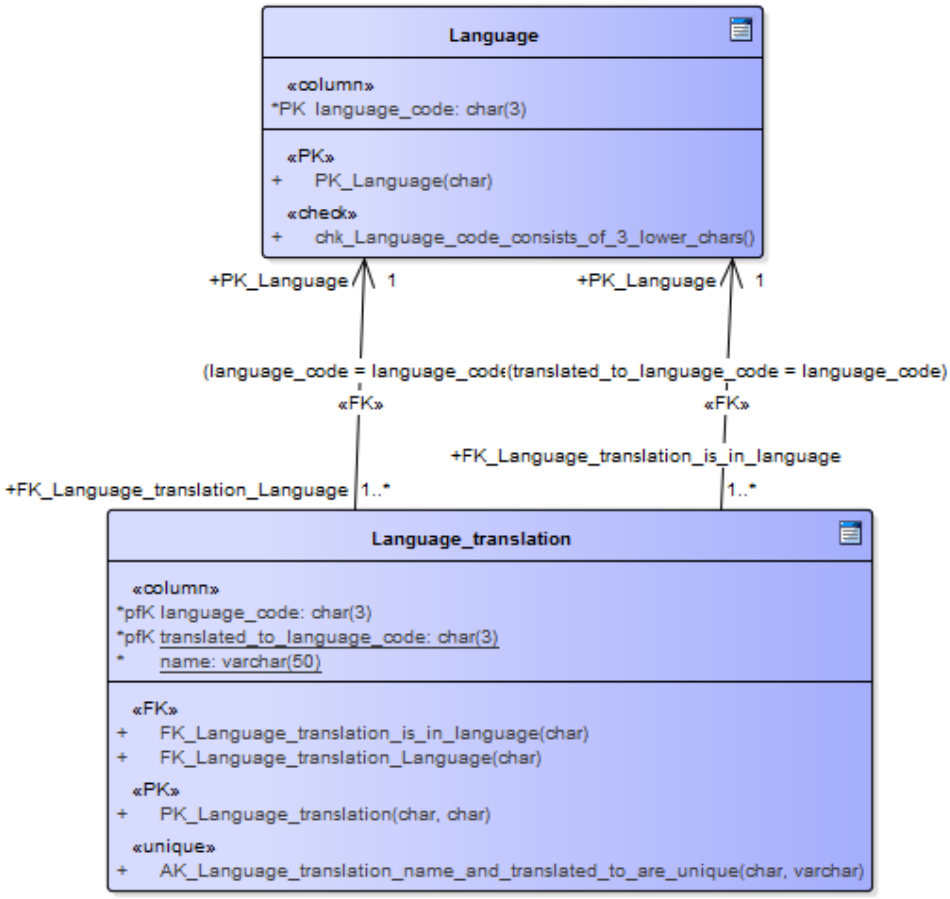
Joonis 36. Sõltuvuste sisestuse printsiibi kasutamine.

Pärimine on objektorienteeritud programmeerimise puhul see, kui mingi teine klass pärib baasklassi ehk vanemklassi omadused ja funktsioonid.

Komposiitmustrit kasutatakse vaate genereerimisel. Nimelt pannakse vaated erinevatest komponentidest kokku. Lihtne näide oleks päise, sisu ja jaluse kombineerimine.

7.3.2 Andmebaasi disain

Andmebaasi disainidiagramm on sisuliselt sama, mis on esitatud jaotises 4.4 (Joonis 16). Ainsaks erinevuseks on tabeli *Language* jagamine kaheks erinevaks tabeliks nii, et ka keele nimetuste tõlkimiseks kasutataks sama mustrit (Joonis 37).



Joonis 37. Rakenduse andmebaasi disainidiagrammi osa.

Keele kood, millesse nimetus tõlgitud on ja nimetus peavad kombineeritult unikaalsed olema. Primaarvõtmeks on kahe keele koodi kombinatsioon.

7.3.3 Olemasolevad juhised mitmekeelsete veebirakenduste loomiseks

Üks variant mitmekeelse veebirakenduse ehitamiseks on luua iga keele jaoks erinevad HTML-i failid [16], kuid sel juhul on hilisem muudatuste tegemine raskendatud, sest koodi tuleb muuta igas erineva keele HTML failis. Tegu on lihtsa lahendusega, mille realiseerimiseks pole eriti teadmisi vaja ja praktikas pole see soovitatav lähenemine, kuna käib DRY (*don't repeat yourself*) printsiibi vastu.

Teine meetod oleks iga keele jaoks eraldi XML-failide (või ka mõnda muut tüüpi fail, mida rakendus lugeda oskab) loomine [16]. Nendes failides oleksid kasutajaliides esitatavate tekstide tõlked.

Kolmandaks lähenemiseks soovitatakse tõlked andmebaasi salvestada [16], [24]. Seda lähenemist soovitatakse juhul, kui on palju erinevaid ja pidevas muutumises olevaid veebilehti [24].

Failipõhise üldlahenduse korral võib näiteks PHP kasutamisel luua failid, mis sisaldavad tõlgete massiive. Erinevad massiivi kasutamise variandid mitmekeelsuse toetamisel on järgmised.

- Erinevate failide loomine erinevate keelte jaoks (näiteks lang.en.php, lang.de.php) [13], kus tõlked salvestatakse massiivi võti-väärtus paaridena. Võtmeks on tõlgitava teksti identifikaator ja väärtuseks tõlge (Joonis 38).
- Mitme dimensiooniga massiivide kasutamine [24], kus ühes massiivis on kõik tõlked (Joonis 39).
- Ühes massiivis on ühe fraasi tõlked [24]. Joonis 40 on võtmena kasutatud keele koodi, kuid võib ka kasutada arve, et saaks kõik tõlked ühe reana lisada ja mujal hoida kaardistust selle kohta, milline keel millisele arvule vastab.

```
$lang['products'] = 'Products';
$lang['product'] = 'Product';
```

Joonis 38. Eraldi failis hoitav ingliskeelsete tõlgetega massiiv.

```
$translation['eng']['products'] = 'Products';
$translation['est']['products'] = 'Tooted';
```

Joonis 39. Mitme dimensiooniga massiivi kasutamine.

```
$products['eng'] = 'Products';
$products['est'] = 'Tooted';
```

Joonis 40. Iga fraasi jaoks eraldi massiiv.

Massiivide kasutamine mitmekeelsuse jaoks on enimlevinud lähenemine, kuid keerulisemate veebilehtede loomisel ei soovitata seda kasutada. Näiteks veebilehel phptherightway [30] soovitatakse kasutada Unix-i põhist süsteemi gettext.

Käesolevas töös otsustati kasutusele võtta erinevates keelefailides olev massiiv.

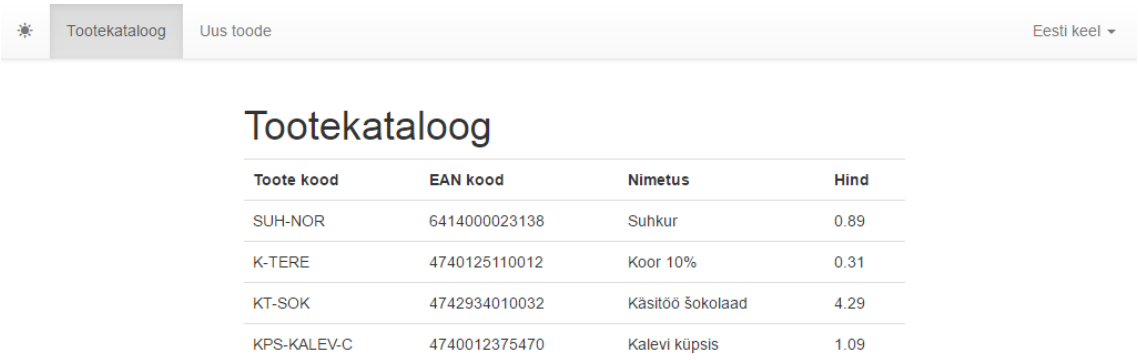
7.3.4 Kasutajaliides

Kasutajaliideses on kaks vaadet: tootekataloogi ja tootevormi vaade. Tootekataloogis kuvatakse kasutajale tabel toodetega (Joonis 41). Tootevormi vaates (Joonis 42) saab uut toodet lisada või olemasoleva toote andmeid muuta, olenevalt sellest, kuidas vormile jõuti. Kui kasutaja vajutab päises olevat nuppu „Uus toode“, siis kuvatakse

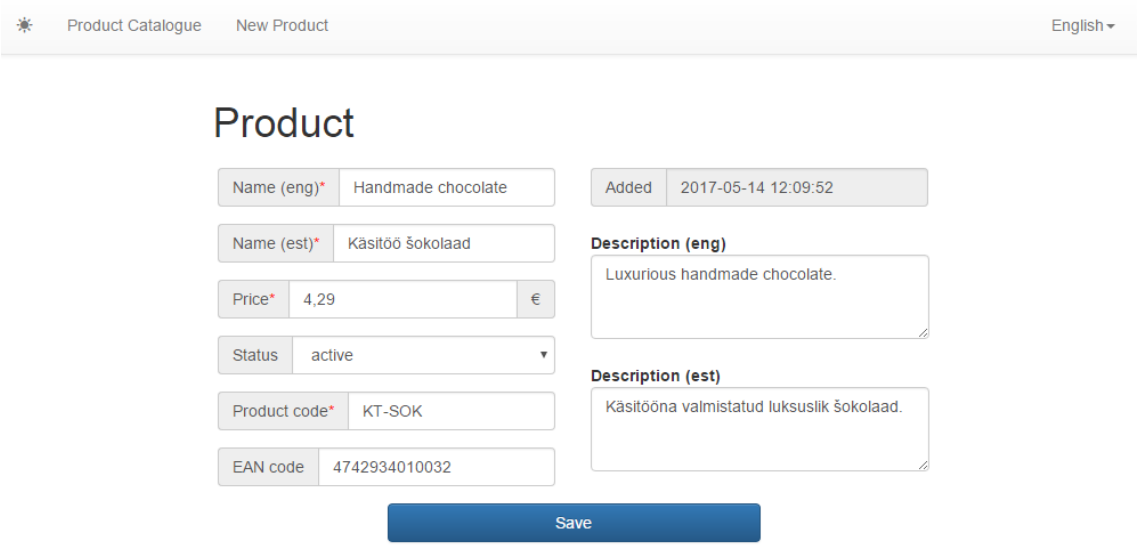
tühja tootevormi. Kui kasutaja vajutab tootekataloogis mõnele toote reale, siis avaneb selle toote andmetega täidetud vorm.

Vaadete kuvamisel kasutatakse malle. Mallid kombineeritakse omavahel üheks terveks leheks komposiitmustri alusel. Loodud rakenduses on viis malli: *header*, *navbar*, *footer*, *product* ja *productList*. Mallid koosnevad staatilisest ja dünaamilisest osast. Staatiline osa on tavaline HTML-is kirjutatud programmikood. Dünaamiline osa tähendab, et mingi osa sisestatakse PHP poolt dünaamiliselt juurde. Loodud rakenduses lisatakse andmed ning nuppude ja vormiväljade pealkirjad dünaamiliselt.

Rakenduse keele valimine käib rippmenüü kaudu, mis asub navigatsioonimenüü paremal servas. Kui rippmenüü kaudu keel vahetada, siis suunatakse kasutaja teisele URI-le, kus parameetri „lang“ väärtus on ära muudetud. Lehe uuesti laadimisel annab vaade mallile tõlgete massiivi uues keeles kaasa.



Joonis 41. Tootekataloogi vaade eesti keeles.



Joonis 42. Tootevormi vaade inglise keeles.

Kasutajaliidese loomisel võeti eeskuju Erply majandustarkvarast.

7.3.5 Rakenduse sisemine ülesehitus

Rakendus on loodud MVC disainiprintsiipe järgides ja koosneb vastavatest komponentidest:

Vaated:

- *View* – baasvaade.
- *ProductView* – toote vormi vaade, mis laiendab baasvaadet *View*.
- *ProductListView* – tootekataloogi vaade, mis laiendab baasvaadet *View*.

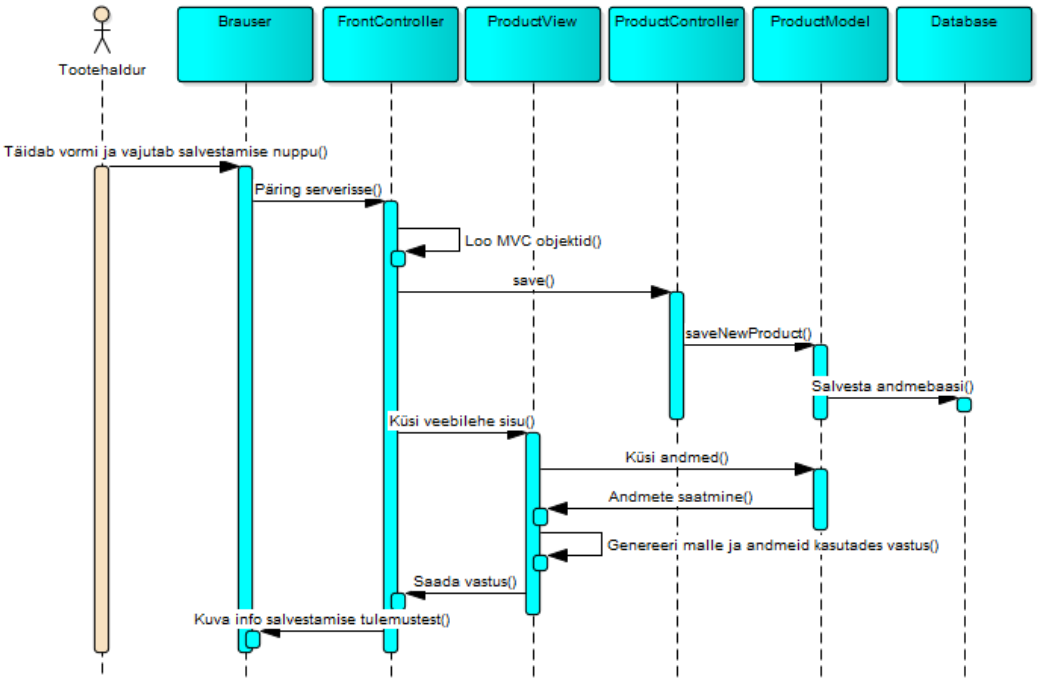
Kontrollerid:

- *ProductController* – edastab täidetud vormi andmed *ProductModel*-ile, kui kasutaja vajutab salvestamise nuppu.
- *ProductListController* – puudub ülesanne, kuna kasutaja ei saa tootekataloogis midagi muud peale toodete nimekirja vaatamise teha.

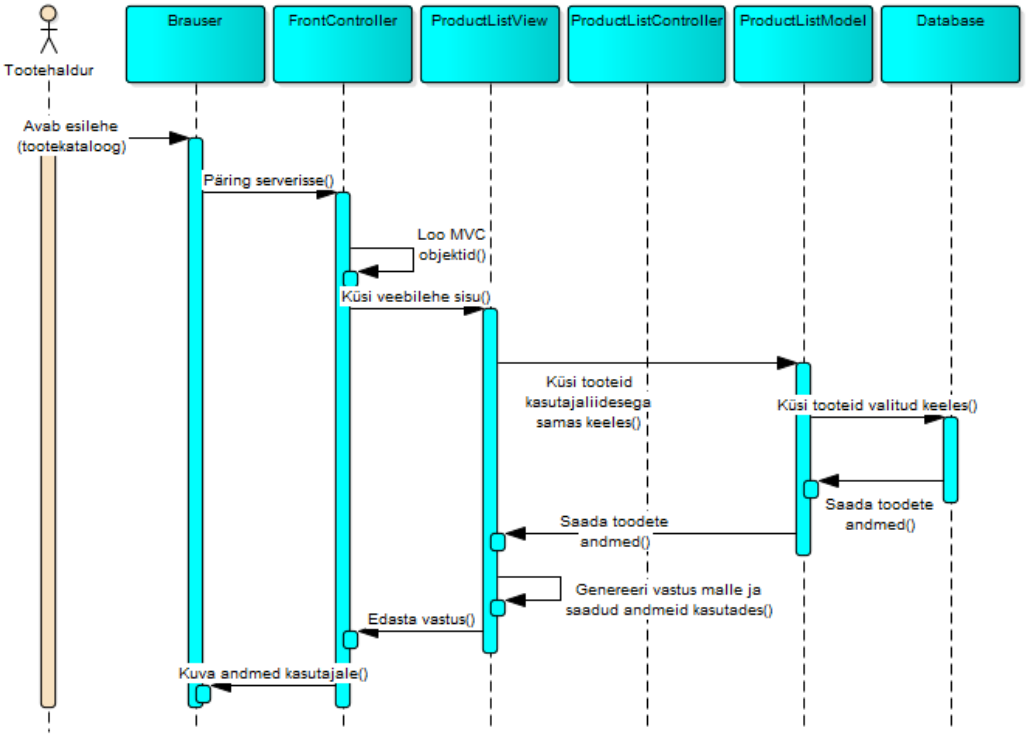
Mudelid:

- *ProductModel* – salvestab toodete andmeid andmebaasi ja uuendab neid ning edastab *ProductView*-le. Lisaks võimaldab vaatel küsida andmebaasis toetatud keeli ja toote seisundi liike, et vaade saaks vormile õiged väljad genereerida.
- *ProductListModel* – küsib andmebaasist toodete nimekirja kasutajaliidesega samas keeles ja edastab *ProductListView*-le.

Joonisel 43 on näha komponentide omavaheline suhtlus toote eduka lisamise korral ja Joonisel 44 tootekataloogi vaatamisel.



Joonis 43. Toote edukalt lisamise jadadiagramm.



Joonis 44. Tootekataloogi vaatamise jadadiagramm.

Router – staatilist marsruutimist kasutav klass. Selles klassis on ära kirjeldatud, millised komponendid (mudel, vaade ja kontroller) tuleb luua vastava URI-s sisalduva parameetri *page* väärtuse korral.

FrontController – vastavalt sisendparameetritele loob vajamineva vaate, mudeli ja kontrolleri. Sisendparameetriteks on URI-s sisalduvad parameetrid ja uus *Router* objekt.

Database – loob andmebaasiühenduse kasutades PDO liidest.

Andmebaasiga suhtlemisel kasutab rakendus PDO-d (*PHP Data Objects*). PDO on liides, mida on seetõttu hea kasutada, et ta lisab andmebaasiga suhtlemiseks abstraktsioonikihi. See tähendab, et erinevate andmebaasisüsteemidega suhtlemiseks saab PHP-s kasutada samasid funktsioone, mitte ei pea kasutama ühe andmebaasisüsteemi spetsiifilisi funktsioone. Lisaks aitab PDO õigesti kasutamine vältida SQL süstimist (*SQL Injection*) [30]. Joonisel 45 on näide, kuidas PDO liidese turvalisuse tagamise funktsionaalsust ei kasutata ära, kuna URI-s sisalduvat parameetrit kasutatakse otse andmebaasipäringus. Pahatahtlik isik saab seda turvaauku hetkega ära kasutada. Seevastu Joonis 46 esitab turvalise viisi andmebaasiga suhtlemiseks kasutades PDO liidest.

```
$pdo = new PDO('pgsql:host=hostname;dbname=db', 'user', 'pass',
       $pdo_options);
$pdo->query('SELECT * FROM products WHERE product_id = ' . $_GET['id']);
```

Joonis 45. Ebaturvaline viis andmebaasist andmete küsimiseks [30].

```
$pdo = new PDO('pgsql:host=hostname;dbname=db', 'user', 'pass',
       $pdo_options);
$pdo->query('SELECT * FROM products WHERE product_id = :id');
$id = filter_input(INPUT_GET, 'id', FILTER_SANITIZE_NUMBER_INT);
$stmt->bindParam(':id', $id, PDO::PARAM_INT);
$stmt->execute();
```

Joonis 46. Turvaline viis andmebaasist andmete küsimiseks kasutades PDO liidest [30].

Valideerimine toimub rakenduses kolmel tasemel: brauseris, serveris PHP-ga ja andmebaasitasemel. Andmebaasi poolt tagastatavad veateated tuleb PHP-ga kinni püüda ning veateadete kasutajale arusaadaval kujul esitamiseks tuleks vaadata, milline sõnum tagastatakse. Sõnumis on kirjas selle välja nimetus, mille salvestamise korral probleem tekkis ja ka kitsenduse nimetus. Keelefailides hoitakse kaardistatult veateadete tõlkeid

ning kui salvestamisel tekib viga, siis mallile antakse tõlgitud veateated kaasa ega salvestata andmeid.

7.3.6 Uue keele toe lisamine

Rakendusele lisati saksa keele tugi nii andmebaasi kui kasutajaliidese poolel, mis osutus üsna lihtsaks. Kuna andmebaasis hoitakse keelte nimetuste tõlkeid, siis oleks soovitatav ka need lisada andmebaasi toetatud keeltes. Ka saksakeelsed toote seisundi liigid tuli andmebaasi lisada. Tänu sellele, et tootevormi vaade genereeritakse andmebaasis toetatud keelte põhjal, siis sai kohe tootevormi kaudu hakata saksakeelseid andmeid sisestama. Kasulik oleks olnud keele tabelisse lisada veel üks väli, mis näitab keele klassifikaatori väärtuse seisundit. See tähendab, et kui mingis keeles andmeid ei soovita enam salvestada, muudetakse keel mitteaktiivseks, mitte ei kustutata ära. Siis jäävad sõltuvatesse tabelitesse selles keeles andmed alles. Seega keele toe lisamise seisukohalt on muster „Universaalne tõlketabel olemitüübi kohta“ väga mugav.

Rakenduse poole pealt tuli luua uus keelefail, mis hoiab saksakeelseid nuppude, vormiväljade ja muude elementide pealkirjade tõlkeid. Lisaks tuli konfiguratsioonist määrata, et saksa keelt on võimalik rippmenüüst kasutajaliidese keeleks valida. Kasulik oleks olnud lisada võimalus määrata üks keel niiöelda põhikeeleks. Sellisel juhul saaks põhikeeles andmed ja pealkirjad välja kuvada, kui valitud keeles need mingil põhjusel puuduma peaks.

7.3.7 Lahenduse puudused

Rakenduse loomist oleks ilmselt märgatavalt lihtsustanud rakendusraamistiku kasutamine, sest siis oleks struktuur juba ees olnud ja mitmed komponendid eelnevalt valmis.

Programmikoodile teste ei kirjutatud. Funktsionaalsust testiti vaid käsitsi.

8 Kokkuvõte

Töö eesmärgiks oli kirjeldada mõningaid disainimustreid mitmekeelsete andmete SQL-andmebaasis hoidmiseks, realiseerida kõikide mustrite põhjal näiteandmetega andmebaas, mille põhjal käivitada päringuid ja andmete muutmise lauseid, realiseerida ühe disainimustri põhjal mitmekeelne näiterakendus ja saadud kogemust analüüsida.

Töö tulemuseks on valminud seitse disainimustrit. Kõigi mustrite puhul on realiseeritud näited, käivitatud päringuid ja andmete muutmise lauseid ning esitatud nende disainide eelised ja puudused. Lisaks kasutati ühe mustri põhjal valminud andmebaasi veebipõhise PHP näiterakenduse loomisel. See rakendus kasutab mitmekeelset andmebaasi ja võimaldab ka muuta kasutajaliidese keelt. Mängiti läbi süsteemi uue keele toe lisamine ning see ei olnud keeruline. Kokkuvõttes võib öelda, et töö eesmärk saavutati.

Analüüsi käigus ilmnes, et üldist parimat disaini ei ole olemas. Parima disaini valik oleneb siiski konkreetse süsteemi nõuetest ja sellest, milliseid mõjureid oluliseks peetakse. Käesolevas töös on esitatud vaid põhilisemad disainimustrid, kuid neid on võimalik varieerida ja ka omavahel kombineerida. Lisaks ilmnes rakenduse loomise käigus mitmeid nüansse, mille peale kohe ei osatud tulla.

Mustrite tugevate ja nõrkade külgede uurimisel jäeti välja andmekäitlusoperatsioonide jõudluse analüüs, mis oleks üks edasi arendamise võimalustest. Arvatavasti muudaks jõudluse arvestamine märgatavalt analüütiliste hierarhiate otsustusmudeli tulemust. Kuna rakenduse loomisel ei kasutatud rakendusraamistikke, siis ka nende uurimine mitmekeelsete rakenduste loomisel, oleks üks edasi arendamise võimalustest. Veel üheks edasiarendamise võimaluseks oleks uurida mitmekeelsete andmete esitamise võimalusi teistel andmemudelitel kui SQL aluseks olev andmemudel põhinevates andmebaasides.

Kasutatud kirjandus

[1] Anderson, S. R. How many languages are there in the world? [WWW] http://www.linguisticsociety.org/content/how-many-languages-are-there-world (21.04.2017)

[2] Codes for the Representation of Names of Languages [WWW] http://www.loc.gov/standards/iso639-2/php/code_list.php (16.04.2017)

[3] DB-Engines Ranking. [WWW] http://db-engines.com/en/ranking (25.03.2017)

[4] Developing a multilingual website, what is the proper way to design the database? [WWW] https://www.quora.com/Developing-a-multilingual-website-what-is-the-proper-way-to-design-the-database (01.03.2017)

[5] Dias, A. Unique partial indexes with PostgreSQL. [WWW] https://medium.com/little-programming-joys/unique-partial-indexes-with-postgresql-86e137905c12 (22.05.2017)

[6] Eesti keele seletav sõnaraamat. Tee. [WWW] http://www.eki.ee/dict/ekss/index.cgi?Q=tee&F=M (21.04.2017)

[7] Entity-attribute-value model. [WWW] https://en.wikipedia.org/wiki/Entity–attribute–value_model (25.04.2017)

[8] Ethologue. [WWW] https://www.ethnologue.com/ (21.04.2017)

[9] Google Translate. [WWW] https://translate.google.com/about/intl/en_ALL/languages/index.html (16.04.2017)

[10] GTS Website Translator. [WWW] https://www.gts-translation.com/tools/website-translator/ (16.04.2017)

[11] Holywell, S. SQL Style Guide.[WWW] http://www.sqlstyle.guide/ (11.05.2017)

[12] Hopkins, C. The MVC Pattern and PHP, Part 1. [WWW] https://www.sitepoint.com/the-mvc-pattern-and-php-1/ (18.05.2017)

[13] How to Add PHP Multilingual Support to a Website. [WWW] http://www.bitrepository.com/php-how-to-add-multi-language-support-to-a-website.html (12.05.2017)

[14] International – Multiple language database design. [WWW] https://social.msdn.microsoft.com/Forums/sqlserver/en-US/0d559276-c669-46f0-b36a-e28fd8fb1bb7/international-multiple-language-database-design?forum=transactsql (20.04.2017)

[15] Is this the correct approach for multilingual data in database? [WWW] http://stackoverflow.com/questions/16002180/is-this-the-correct-aproach-for-multilingual-data-in-a-database (15.04.2017)

[16] Janjic, V. Building a Multilingual PHP Website. [WWW] http://www.phpbuilder.com/columns/MultilingualPHPSite/index.php3 (13.05.2017)

[17] Jõgi, M. Mõned disainimustrid isikunimede hoidmiseks SQL-andmebaasides : bakalaureusetöö. Tallinna Tehnikaülikool, Tallinn, 2016. [*Online*] (28.02.2017)

[18] KAMA: KASUTATAV EESTI MASINTÕLGE. [WWW] https://www.keeletehnoloogia.ee/et/ekt-projektid/kama-kasutatav-eesti-masintolge (21.05.2017)

[19] Kharade, P. Multi Language Database Design Approaches. [WWW] https://www.linkedin.com/pulse/multi-language-database-design-appoaches-pramod-kharade (26.02.2017)

[20] Kher, S. How to Design a Localization-Ready System. [WWW] http://www.vertabelo.com/blog/technical-articles/data-modeling-for-multiple-languages-how-to-design-a-localization-ready-system (28.02.2017)

[21] Kyte, T. Back to Basics [WWW] http://www.oracle.com/technetwork/issue-archive/o40asktom-1870727.html (02.04.2017)

[22] Microsoft. Maximum Capacity Specifications for SQL Server. [WWW] https://docs.microsoft.com/en-us/sql/sql-server/maximum-capacity-specifications-for-sql-server (25.03.2017)

[23] Microsoft. Special Table Types. [WWW] https://msdn.microsoft.com/en-us/library/ms186986.aspx (25.03.2017)

[24] Multilingual Webpages via PHP, Arrays and MySQL. [WWW] https://www.dougv.com/2007/11/multilingual-web-pages-via-php-arrays-and-mysql/ (13.05.2017)

[25] MySQL. C.10.4 Limits on Table Column Count and Row Size [WWW] https://dev.mysql.com/doc/refman/5.7/en/column-count-limit.html (25.03.2017)

[26] Naypoka, L. Multilanguage Database Design in MySQL. [WWW] http://www.apphp.com/tutorials/index.php?page=multilanguage-database-design-in-mysql (26.02.2017)

[27] Nguyen, V., Deeds-Rubin, S., Tan, T., Boehm, B. A SLOC Counting Standard. [WWW] https://pdfs.semanticscholar.org/1110/f97bdbd475ec4b3abdd59e5147583bd50332.pdf?_ga=1.254029832.1057269101.1488309094 (28.02.2017)

[28] Once And Only Once. [WWW] http://wiki.c2.com/?OnceAndOnlyOnce (22.05.2017)

[29] Online-Translator. [WWW] http://www.online-translator.com/webmaster/ (16.04.2017)

[30] PHP The Right Way. [WWW] http://www.phptherightway.com/ (14.05.2017)

[31] PostgreSQL. 22.2. Collation Support. [WWW] https://www.postgresql.org/docs/9.5/static/collation.html (26.02.2017)

[32] PostgreSQL. 22.3. Character Set Support [WWW] https://www.postgresql.org/docs/9.5/static/multibyte.html (09.05.2017)

[33] PostgreSQL. 8.3. Character types. [WWW] https://www.postgresql.org/docs/9.5/static/datatype-character.html (09.05.2017)

[34] PostgreSQL. About. [WWW] https://www.postgresql.org/about/ (24.03.2017)

[35] Pym, A. Localization: On its nature, virtues and dangers. [WWW] https://brage.bibsys.no/xmlui/bitstream/handle/11250/2394909/Pym_2005_17_Localization_%20On%20its%20nature%2c%20virtues%20and%20dangers.pdf?sequence=1&isAllowed=y (28.02.2017)

[36] Saaty, T. L. (2008). Decision making with the analytic hierarchy process. *International journal of services sciences*, *1*(1), 83-98.

[37] Saba, M. Human Translation vs. Machine Translation. [WWW] http://www.anecsys.com/2015/04/human-translation-vs-machine-translation/ (07.03.2017)

[38] Seguin, K. Creating multilingual websites - Part 2. [WWW] https://www.codeproject.com/kb/aspnet/localizedsamplepart2.aspx#databasedesign (26.02.2017)

[39] Sen, A. Five Simple Database Design Errors You Should Avoid. [WWW] https://www.simple-talk.com/sql/database-administration/five-simple--database-design-errors-you-should-avoid/ (24.04.2017)

[40] SYSTRANLinks. [WWW] http://www.systranlinks.com/ (16.04.2017)

[41] Tartu Ülikooli Masintõlge. [WWW] http://neurotolge.ee (21.04.2017)

[42] The Advantages and Disadvantages of Machine Translation. [WWW] http://www.omniglot.com/language/articles/machinetranslation.htm (05.03.2017)

[43] TIOBE Index for May 2017. [WWW] https://www.tiobe.com/tiobe-index/ (21.05.2017)

[44] To Machine Translate Or Not To Machine Translate? [WWW] http://thelanguagefactory.co.uk/to-machine-translate-or-not-to-machine-translate/ (05.03.2017)

[45] Translation – Multilingual/Multilanguage Database Design [WWW] http://cleancodedevelopment-qualityseal.blogspot.com.ee/2013/06/translation-multilingualmultilanguage.html (15.04.2017)

[46] Use Sparse Columns. [WWW] https://docs.microsoft.com/en-us/sql/relational-databases/tables/use-sparse-columns (19.05.2017)

[47] Vaishnavi V., Kuechler B. Design Science Research in Information Systems. [WWW] http://desrist.org/desrist/content/design-science-research-in-information-systems.pdf (16.04.2017)

[48] Vallaste, H. E-teatmik. [WWW] http://www.vallaste.ee/ (19.05.2017)

[49] Veebisaidi tõlkija. [WWW] https://translate.google.com/manager/website/ (05.03.2017)

[50] Vellemaa, A. Mõned disainimustrid klassifikaatorite esitamiseks SQL andmebaasides : bakalaureusetöö. TTÜ Informaatikainstituut, 2015. [*Online*] (09.05.2017)

[51] Võhandu, L. Subjektiivsetest hinnangutest objektiivsete tulemusteni : loengukonspekt. Tallinn : Tallinna Tehnikaülikooli trükikoda, 1998.

[52] What are best practices for multi-language database design? [WWW] http://stackoverflow.com/questions/929410/what-are-best-practices-for-multi-language-database-design (26.02.2017)

[53] What is PHP? [WWW] http://php.net/manual/en/intro-whatis.php (19.05.2017)

[54] What is the record for the most languages spoken by one person? [WWW] http://www.sciencefocus.com/qa/what-record-most-languages-spoken-one-person (08.05.2016)

[55] What's the best database structure to keep multilingual data? [WWW] http://stackoverflow.com/questions/2227985/whats-the-best-database-structure-to-keep-multilingual-data (26.02.2017)

[56] Web-HIPRE. [WWW] http://hipre.aalto.fi/ (10.05.2017)

Lisa 1 – Skriptid

Domeenid

Skeemi loomise lause:

```
CREATE SCHEMA domain;
```

Domeeni loomise laused:

```
CREATE DOMAIN domain.description VARCHAR(255)
CONSTRAINT chk_Description_must_not_consist_of_spaces CHECK
(VALUE!~'^[[:space:]]*$');

CREATE DOMAIN domain.name VARCHAR(50)
CONSTRAINT chk_Name_must_not_consist_of_spaces CHECK
(VALUE!~'^[[:space:]]*$');

CREATE DOMAIN domain.product_name text
CONSTRAINT chk_Product_name_must_not_consist_of_spaces CHECK
(VALUE!~'^[[:space:]]*$');

CREATE DOMAIN domain.product_description text
CONSTRAINT chk_Product_description_must_not_consist_of_spaces CHECK
(VALUE!~'^[[:space:]]*$');

CREATE DOMAIN domain.value text NOT NULL
CONSTRAINT chk_Value_must_not_consist_of_spaces CHECK
(VALUE!~'^[[:space:]]*$');

CREATE DOMAIN domain.product_price decimal(19,2) NOT NULL
CONSTRAINT chk_Product_price CHECK (VALUE>=0);

CREATE DOMAIN domain.product_code varchar(50) NOT NULL
CONSTRAINT chk_Product_code_must_not_consist_of_spaces CHECK
(VALUE!~'^[[:space:]]*$');

CREATE DOMAIN domain.ean_code varchar(13)
       CONSTRAINT chk_Ean_kood CHECK (VALUE~'^[0-9]{13}$');
```

Muster “Keelepõhised tõlketabelid atribuudi kohta”

Skeemi loomise lause:

```
CREATE SCHEMA attribute_tables;
```

Tabelite loomise laused:

```
CREATE TABLE attribute_tables.Language (
       language_code domain.language_code,
       name domain.name NOT NULL,
       CONSTRAINT PK_Language PRIMARY KEY (language_code),
       CONSTRAINT AK_Language_name_is_unique UNIQUE (name)
);

CREATE TABLE attribute_tables.Product_status_type (
       product_status_type_code smallint NOT NULL,
       CONSTRAINT PK_Product_status_type PRIMARY KEY
       (product_status_type_code)
);

CREATE TABLE attribute_tables.Product_status_type_description (
       product_status_type_code smallint NOT NULL,
       language_code char(3)      NOT NULL,
       description domain.description NOT NULL,
       CONSTRAINT PK_Product_status_type_description PRIMARY KEY
       (product_status_type_code,language_code),
       CONSTRAINT FK_Product_status_type_description_language FOREIGN KEY
       (language_code) REFERENCES attribute_tables.Language (language_code)
       ON DELETE No Action ON UPDATE Cascade,
       CONSTRAINT FK_Product_status_type_description_Product_status_type
       FOREIGN KEY (product_status_type_code) REFERENCES
       attribute_tables.Product_status_type (product_status_type_code) ON
       DELETE Cascade ON UPDATE Cascade
);

CREATE TABLE attribute_tables.Product_status_type_name (
       product_status_type_code smallint NOT NULL,
       language_code char(3)      NOT NULL,
       name domain.name NOT NULL,
       CONSTRAINT PK_Product_status_type_name PRIMARY KEY
       (product_status_type_code,language_code),
       CONSTRAINT AK_Product_status_type_name_language_code_on_unikaalne
       UNIQUE (name,language_code),
       CONSTRAINT FK_Product_status_type_name_on_keeles FOREIGN KEY
       (language_code) REFERENCES attribute_tables.Language (language_code)
       ON DELETE No Action ON UPDATE Cascade,
       CONSTRAINT FK_Product_status_type_name_Product_status_type FOREIGN KEY
       (product_status_type_code) REFERENCES
       attribute_tables.Product_status_type (product_status_type_code) ON
       DELETE Cascade ON UPDATE Cascade
);
```

```
CREATE TABLE attribute_tables.Product (
	product_id serial,
	product_code domain.product_code,
	ean_code domain.ean_code,
	product_status_type_code smallint NOT NULL DEFAULT 1,
	time_added timestamp NOT NULL DEFAULT localtimestamp(0),
	price domain.product_price,
	CONSTRAINT PK_Product PRIMARY KEY (product_id),
	CONSTRAINT AK_Product_code_is_unique UNIQUE (product_code),
	CONSTRAINT AK_Ean_code_is_unique UNIQUE (ean_code),
	CONSTRAINT FK_Product_status FOREIGN KEY (product_status_type_code)
	REFERENCES attribute_tables.Product_status_type
	(product_status_type_code) ON DELETE No Action ON UPDATE Cascade
);

CREATE TABLE attribute_tables.Product_description (
	product_id integer NOT NULL,
	language_code char(3)	NOT NULL,
	description domain.product_description NOT NULL,
	CONSTRAINT PK_Product_description PRIMARY KEY
	(product_id,language_code),
	CONSTRAINT FK_Product_description_language FOREIGN KEY (language_code)
	REFERENCES attribute_tables.Language (language_code) ON DELETE No
	Action ON UPDATE Cascade,
	CONSTRAINT FK_Product_description_Product FOREIGN KEY (product_id)
	REFERENCES attribute_tables.Product (product_id) ON DELETE Cascade ON
	UPDATE No Action
);

CREATE TABLE attribute_tables.Product_name (
	product_id integer NOT NULL,
	language_code char(3)	NOT NULL,
	name domain.product_name NOT NULL,
	CONSTRAINT PK_Product_name PRIMARY KEY (product_id,language_code),
	CONSTRAINT FK_Product_name_language FOREIGN KEY (language_code)
	REFERENCES attribute_tables.Language (language_code) ON DELETE No
	Action ON UPDATE Cascade,
	CONSTRAINT FK_Product_name_Product FOREIGN KEY (product_id) REFERENCES
	attribute_tables.Product (product_id) ON DELETE Cascade ON UPDATE No
	Action
);
```

Indeksite loomise laused:

```
CREATE INDEX IXFK_Product_status_type_description_language ON
attribute_tables.Product_status_type_description (language_code ASC);

CREATE INDEX IXFK_Product_status_type_name_language ON
attribute_tables.Product_status_type_name (language_code ASC);
```

```
CREATE INDEX IXFK_Product_status ON attribute_tables.Product
(product_status_type_code ASC);

CREATE INDEX IXFK_Product_description_language ON
attribute_tables.Product_description (language_code ASC);

CREATE INDEX IXFK_Product_name_language ON attribute_tables.Product_name
(language_code ASC);
```

Andmete lisamise laused:

```
INSERT INTO attribute_tables.Language (language_code, name) VALUES
	('est', 'eesti keel'),
	('eng', 'inglise keel');

INSERT INTO attribute_tables.Product_status_type (product_status_type_code)
VALUES
	(0),
	(1),
	(2),
	(3);

INSERT INTO attribute_tables.Product_status_type_description
(product_status_type_code, language_code, description) VALUES
	(0, 'est', 'Toode on arhiveeritud ja seda ei saa osta ega müüa.'),
	(1, 'est', 'Toode on aktiivne ja seda saab nii osta kui müüa.'),
	(2, 'est', 'Toodet ei saa enam lattu tellida.'),
	(3, 'est', 'Toode ei ole müügiks.');

INSERT INTO attribute_tables.Product_status_type_name
(product_status_type_code, language_code, name) VALUES
	(0, 'est', 'arhiveeritud'),
	(0, 'eng', 'archived'),
	(1, 'est', 'aktiivne'),
	(1, 'eng', 'active'),
	(2, 'est', 'ei saa enam tellida'),
	(2, 'eng', 'no longer ordered'),
	(3, 'est', 'ei ole müügiks'),
	(3, 'eng', 'not for sale');

INSERT INTO attribute_tables.Product (product_code, ean_code, price) VALUES
	('K-PAULIG-J', '6411300158072', 4.99),
	('K-TERE', '4740125110012', 0.31),
	('SUH-NOR', '6414000023138', 0.89),
	('KT-SOK', '4742934010032', 4.29);
```

```
INSERT INTO attribute_tables.Product_description (product_id, language_code,
description) VALUES
	(1, 'est', 'Hõrk kauakestva ja rikkaliku järelmaitsega kohv.'),
	(1, 'eng', 'Delicate coffee with long-lasting and rich aftertaste.'),
	(3, 'est', 'Valge suhkur. Sobib küpsetamiseks, hoidiste
	valmistamiseks, magustoitude tegemiseks ja kohvi sisse.'),
	(3, 'eng', 'White sugar. Suitable for baking, making preserves,
	desserts and into coffee.'),
	(4, 'est', 'Käsitööna valmistatud luksuslik šokolaad.'),
	(4, 'eng', 'Luxurious handmade chocolate.');

INSERT INTO attribute_tables.Product_name (product_id, language_code, name)
VALUES
	(1, 'est', 'Jahvatatud kohv'),
	(1, 'eng', 'Ground coffee'),
	(2, 'est', 'Koor 10%'),
	(2, 'eng', 'Cream 10%'),
	(3, 'est', 'Suhkur'),
	(3, 'eng', 'Sugar'),
	(4, 'est', 'Käsitöö šokolaad'),
	(4, 'eng', 'Handmade chocolate');
```

Andmete muutmise laused:

```
UPDATE attribute_tables.Product
	SET price = 0.99,
		product_status_type_code = 2
	WHERE product_code = 'SUH-NOR';

UPDATE attribute_tables.Product_name
	SET name = 'Demerara Suhkur'
	WHERE product_id = 4
		AND language_code = 'est';
```

Muster „Keelepõhised tõlketabelid olemitüübi kohta“:

Skeemi loomise lause:

```
CREATE SCHEMA translation_tables;
```

Tabelite loomise laused:

```
CREATE TABLE translation_tables.Product_status_type (
	product_status_type_code smallint NOT NULL,
	CONSTRAINT PK_Product_status_type PRIMARY KEY
	(product_status_type_code)
);
```

```
CREATE TABLE translation_tables.Product_status_type_deu (
       product_status_type_code smallint NOT NULL,
       name domain.name NOT NULL COLLATE "de_DE",
       description domain.description COLLATE "de_DE",
       CONSTRAINT PK_Product_status_type_deu PRIMARY KEY
       (product_status_type_code),
       CONSTRAINT AK_Product_status_type_deu_name_is_unique UNIQUE (name),
       CONSTRAINT FK_Product_status_type_deu_Product_status_type FOREIGN KEY
       (product_status_type_code) REFERENCES
       translation_tables.Product_status_type (product_status_type_code) ON
       DELETE Cascade ON UPDATE Cascade
);

CREATE TABLE translation_tables.Product_status_type_eng (
       product_status_type_code smallint NOT NULL,
       name domain.name NOT NULL COLLATE "en_US",
       description domain.description   COLLATE "en_US",
       CONSTRAINT PK_Product_status_type_eng PRIMARY KEY
       (product_status_type_code),
       CONSTRAINT AK_Product_status_type_eng_name_is_unique UNIQUE (name),
       CONSTRAINT FK_Product_status_type_eng_Product_status_type FOREIGN KEY
       (product_status_type_code) REFERENCES
       translation_tables.Product_status_type (product_status_type_code) ON
       DELETE Cascade ON UPDATE Cascade
);

CREATE TABLE translation_tables.Product_status_type_est(
       product_status_type_code smallint NOT NULL,
       name domain.name NOT NULL COLLATE "et_EE",
       description domain.description   COLLATE "et_EE",
       CONSTRAINT PK_Product_status_type_est PRIMARY KEY
       (product_status_type_code),
       CONSTRAINT AK_Product_status_type_est_name_is_unique UNIQUE (name),
       CONSTRAINT FK_Product_status_type_est_Product_status_type FOREIGN KEY
       (product_status_type_code) REFERENCES
       translation_tables.Product_status_type (product_status_type_code) ON
       DELETE Cascade ON UPDATE Cascade
);
```

```
CREATE TABLE translation_tables.Product(
	product_id serial,
	product_code domain.product_code,
	ean_code domain.ean_code,
	product_status_type_code smallint NOT NULL DEFAULT 1,
	time_added timestamp NOT NULL DEFAULT localtimestamp(0),
	price domain.product_price,
	CONSTRAINT PK_Product PRIMARY KEY (product_id),
	CONSTRAINT AK_Ean_code_is_unique UNIQUE (ean_code),
	CONSTRAINT AK_Product_code_is_unique UNIQUE (product_code),
	CONSTRAINT FK_Product_status FOREIGN KEY (product_status_type_code)
	REFERENCES translation_tables.Product_status_type
	(product_status_type_code) ON DELETE No Action ON UPDATE Cascade
);

	CREATE TABLE translation_tbles.Product_deu(
	product_id integer NOT NULL,
	name domain.product_name NOT NULL COLLATE "de_DE",
	description domain.product_description COLLATE "de_DE",
	CONSTRAINT PK_Product_deu PRIMARY KEY (product_id),
	CONSTRAINT FK_Product_deu_Product FOREIGN KEY (product_id) REFERENCES
	translation_tables.Product (product_id) ON DELETE Cascade ON UPDATE No
	Action
);

CREATE TABLE translation_tables.Product_eng(
	product_id integer NOT NULL,
	name domain.product_name NOT NULL COLLATE "en_US",
	description domain.product_description COLLATE "en_US",
	CONSTRAINT PK_Product_eng PRIMARY KEY (product_id),
	CONSTRAINT FK_Product_eng_Product FOREIGN KEY (product_id) REFERENCES
	translation_tables.Product (product_id) ON DELETE Cascade ON UPDATE No
	Action
);

CREATE TABLE translation_tables.Product_est(
	product_id integer NOT NULL,
	name domain.product_name NOT NULL COLLATE "et_EE",
	description domain.product_description COLLATE "et_EE",
	CONSTRAINT PK_Product_est PRIMARY KEY (product_id),
	CONSTRAINT FK_Product_est_Product FOREIGN KEY (product_id) REFERENCES
	translation_tables.Product (product_id) ON DELETE Cascade ON UPDATE No
	Action
);
```

Indeksi loomise lause:

```
CREATE INDEX IXFK_Product_status ON translation_tables.Product
(product_status_type_code ASC);
```

Andmete lisamise laused:

```
INSERT INTO translation_tables.Product_status_type (product_status_type_code)
VALUES
	(0),
	(1),
	(2),
	(3);

INSERT INTO translation_tables.Product_status_type_deu
(product_status_type_code, name) VALUES
	(0, 'archiviert'),
	(1, 'aktiv'),
	(2, 'nicht mehr bestellt'),
	(3, 'nicht zu verkaufen');

INSERT INTO translation_tables.Product_status_type_eng
(product_status_type_code, name, description) VALUES
	(0, 'archived', 'Product is archived and it cannot be sold or
	ordered.'),
	(1, 'active', 'Product is active. It is possible to order and sell the
	product.'),
	(2, 'no longer ordered', 'Product is no longer ordered.'),
	(3, 'not for sale', 'Product is not for sale');

INSERT INTO translation_tables.Product_status_type_est
(product_status_type_code, name, description) VALUES
	(0, 'arhiveeritud', 'Toode on arhiveeritud ja seda ei saa osta ega
	müüa.'),
	(1, 'aktiivne', 'Toode on aktiivne ja seda saab nii osta kui müüa.'),
	(2, 'ei saa enam tellida', 'Toodet ei saa enam lattu tellida.'),
	(3, 'ei ole müügiks', 'Toode ei ole müügiks.');

INSERT INTO translation_tables.Product (product_code, ean_code, price) VALUES
	('K-PAULIG-J', '6411300158072', 4.99),
	('K-TERE', '4740125110012', 0.31),
	('SUH-NOR', '6414000023138', 0.89),
	('KT-SOK', '4742934010032', 4.29);

INSERT INTO translation_tables.Product_deu (product_id, name) VALUES
	(1, 'gemahlenen Kaffee'),
	(2, 'Sahne 10%'),
	(3, 'Zucker'),
	(4, 'Handgemachte Schokolade');
```

```
INSERT INTO translation_tables.Product_eng (product_id, name, description)
VALUES
	(1, 'Ground coffee', 'Delicate coffee with long-lasting and rich
	aftertaste.'),
	(2, 'Cream 10%', NULL),
	(3, 'Sugar', 'White sugar. Suitable for baking, making preserves,
	desserts and into coffee.'),
	(4, 'Handmade chocolate', 'Luxurious handmade chocolate.');

INSERT INTO translation_tables.Product_est (product_id, name, description)
VALUES
	(1, 'Jahvatatud kohv', 'Hõrk kauakestva ja rikkaliku järelmaitsega
	kohv.'),
	(2, 'Koor 10%', NULL),
	(3, 'Suhkur', 'Valge suhkur. Sobib küpsetamiseks, hoidiste
	valmistamiseks, magustoitude tegemiseks ja kohvi sisse.'),
	(4, 'Käsitöö šokolaad', 'Käsitööna valmistatud luksuslik šokolaad.');
```

Andmete muutmise laused:

```
UPDATE translation_tables.Product
	SET price = 0.99,
		product_status_type_code = 2
	WHERE product_code = 'SUH-NOR';

UPDATE translation_tables.Product_est
	SET name = 'Demerara suhkur'
	WHERE product_id = 5;
```

Muster „Korduskasutatavad tõlked“:

Skeemi loomise lause:

```
CREATE SCHEMA reusable;
```

Tabelite loomise laused:

```
CREATE TABLE reusable.Language (
	language_code domain.language_code,
	name domain.name NOT NULL,
	CONSTRAINT PK_Language PRIMARY KEY (language_code),
	CONSTRAINT AK_Language_name_is_unique UNIQUE (name)
);

CREATE TABLE reusable.Translation (
	translation_id serial,
	CONSTRAINT PK_Translation PRIMARY KEY (translation_id)
);
```

```
CREATE TABLE reusable.Translation_entry (
	translation_id integer NOT NULL,
	language_code char(3)	NOT NULL,
	value domain.value,
	CONSTRAINT PK_Translation_entry PRIMARY KEY
	(translation_id,language_code),
	CONSTRAINT AK_Translation_entry_language_code_and_value_are_unique
	UNIQUE (language_code, value),
	CONSTRAINT FK_Translation_entry_language FOREIGN KEY (language_code)
	REFERENCES reusable.Language (language_code) ON DELETE No Action ON
	UPDATE Cascade,
	CONSTRAINT FK_Translation_entry_Translation FOREIGN KEY
	(translation_id) REFERENCES reusable.Translation (translation_id) ON
	DELETE Cascade ON UPDATE No Action
);

CREATE TABLE reusable.Product_status_type (
	product_status_type_code smallint NOT NULL,
	name integer NOT NULL,
	description integer,
	CONSTRAINT PK_Product_status_type PRIMARY KEY
	(product_status_type_code),
	CONSTRAINT AK_Product_status_type_name_is_unique UNIQUE (name),
	CONSTRAINT FK_Product_status_type_description_value FOREIGN KEY
	(description) REFERENCES reusable.Translation (translation_id) ON
	DELETE No Action ON UPDATE No Action,
	CONSTRAINT FK_Product_status_type_name_value FOREIGN KEY (name)
	REFERENCES reusable.Translation (translation_id) ON DELETE No Action
	ON UPDATE No Action
);
```

```
CREATE TABLE reusable.Product (
	product_id serial,
	product_code domain.product_code,
	ean_code domain.ean_code,
	name integer NOT NULL,
	product_status_type_code smallint NOT NULL DEFAULT 1,
	description integer,
	price domain.product_price,
	time_added timestamp NOT NULL DEFAULT localtimestamp(0),
	CONSTRAINT PK_Product PRIMARY KEY (product_id),
	CONSTRAINT AK_Ean_code_is_unique UNIQUE (ean_code),
	CONSTRAINT AK_Product_code_is_unique UNIQUE (product_code),
	CONSTRAINT FK_Product_description_value FOREIGN KEY (description)
	REFERENCES reusable.Translation (translation_id) ON DELETE No Action
	ON UPDATE No Action,
	CONSTRAINT FK_Product_name_value FOREIGN KEY (name) REFERENCES
	reusable.Translation (translation_id) ON DELETE No Action ON UPDATE No
	Action,
	CONSTRAINT FK_Product_status FOREIGN KEY (product_status_type_code)
	REFERENCES reusable.Product_status_type (product_status_type_code) ON
	DELETE No Action ON UPDATE Cascade
);
```

Indeksite loomise laused:

```
CREATE INDEX IXFK_Translation_entry_language ON reusable.Translation_entry
(language_code ASC);

CREATE INDEX IXFK_Product_status_type_description_value ON
reusable.Product_status_type (description ASC);

CREATE INDEX IXFK_Product_status_type_name_value ON
reusable.Product_status_type (name ASC);

CREATE INDEX IXFK_Product_description_value ON reusable.Product (description
ASC);

CREATE INDEX IXFK_Product_name_value ON reusable.Product (name ASC);

CREATE INDEX IXFK_Product_status ON reusable.Product
(product_status_type_code ASC);
```

Andmete lisamise laused:

```
INSERT INTO reusable.Language (language_code, name) VALUES
	('est', 'eesti keel'),
	('eng', 'inglise keel');
```

```
INSERT INTO reusable.Translation (translation_id) VALUES
	(DEFAULT),
	(DEFAULT),
	(DEFAULT),
	(DEFAULT),
	(DEFAULT),
	(DEFAULT),
	(DEFAULT),
	(DEFAULT),
	(DEFAULT),
	(DEFAULT),
	(DEFAULT),
	(DEFAULT),
	(DEFAULT),
	(DEFAULT),
	(DEFAULT),
	(DEFAULT);


INSERT INTO reusable.Translation_entry (translation_id, language_code, value)
VALUES
	(1, 'est', 'Jahvatatud kohv'),
	(1, 'eng', 'Ground coffee'),
	(2, 'est', 'Hõrk kauakestva ja rikkaliku järelmaitsega kohv.'),
	(2, 'eng', 'Delicate coffee with long-lasting and rich aftertaste.'),
	(3, 'est', 'Suhkur'),
	(3, 'eng', 'Sugar'),
	(4, 'est', 'Valge suhkur. Sobib küpsetamiseks, hoidiste
	valmistamiseks, magustoitude tegemiseks ja kohvi sisse.'),
	(4, 'eng', 'White sugar. Suitable for baking, making preserves,
	desserts and into coffee.'),
	(5, 'est', 'Koor 10%'),
	(5, 'eng', 'Cream 10%'),
	(6, 'est', 'arhiveeritud'),
	(6, 'eng', 'archived'),
	(7, 'est', 'aktiivne'),
	(7, 'eng', 'active'),
	(8, 'est', 'ei saa enam tellida'),
	(8, 'eng', 'no longer ordered'),
	(9, 'est', 'ei ole müügiks'),
	(9, 'eng', 'not for sale'),
	(10, 'est', 'Toode on arhiveeritud ja seda ei saa osta ega müüa.'),
	(11, 'est', 'Toode on aktiivne ja seda saab nii osta kui müüa.'),
	(12, 'est', 'Toodet ei saa enam lattu tellida.'),
	(13, 'est', 'Toode ei ole müügiks.'),
	(14, 'est', 'Käsitöö šokolaad'),
	(14, 'eng', 'Handmade chocolate'),
	(15, 'est', 'Käsitööna valmistatud luksuslik šokolaad.'),
	(15, 'eng', 'Luxurious handmade chocolate.');
```

```
INSERT INTO reusable.Product_status_type (product_status_type_code, name,
description) VALUES
	(0, 6, 10),
	(1, 7, 11),
	(2, 8, 12),
	(3, 9, 13);

INSERT INTO reusable.Product (product_code, ean_code, price, name,
description) VALUES
	('K-PAULIG-J', '6411300158072', 4.99, 1, 2),
	('K-TERE', '4740125110012', 0.31, 5, NULL),
	('SUH-NOR', '6414000023138', 0.89, 3, 4),
	('KT-SOK', '4742934010032', 4.29, 14, 15);
```

Andmete muutmise laused:

```
UPDATE reusable.Product
	SET price = 0.99,
		product_status_type_code = 2
	WHERE product_code = 'SUH-NOR';

UPDATE reusable.Translation_entry
	SET value = 'Demerara suhkur'
	WHERE translation_id =
		(SELECT name
			FROM reusable.Product
			WHERE product_code = 'SUH-NOR')
		AND language_code = 'est';
```

Muster „Sisu tüübiga universaalne tõlketabel“:

Skeemi loomise lause:

```
CREATE SCHEMA universal_with_type;
```

Tabelite loomise laused:

```
CREATE TABLE universal_with_type.Language (
	language_code domain.language_code
	name domain.name NOT NULL,
	CONSTRAINT PK_Language PRIMARY KEY (language_code),
	CONSTRAINT AK_Language_name_is_unique UNIQUE (name)
);
```

```
CREATE TABLE universal_with_type.Content_type (
	content_type_code smallint NOT NULL,
	name domain.name NOT NULL,
	description domain.description,
	CONSTRAINT PK_Content_type PRIMARY KEY (content_type_code),
	CONSTRAINT AK_Content_type_name_is_unique UNIQUE (name)
);

CREATE TABLE universal_with_type.Product_status_type (
	product_status_type_code smallint NOT NULL,
	CONSTRAINT PK_Product_status_type PRIMARY KEY
	(product_status_type_code)
);

CREATE TABLE universal_with_type.Product_status_type_translation (
	product_status_type_code smallint NOT NULL,
	language_code char(3)	NOT NULL,
	content_type_code smallint NOT NULL,
	value domain.value,
	CONSTRAINT PK_Product_status_type_translation PRIMARY KEY
	(product_status_type_code,language_code,content_type_code),
	CONSTRAINT FK_Product_status_type_t_content_type FOREIGN KEY
	(content_type_code) REFERENCES universal_with_type.Content_type
	(content_type_code) ON DELETE No Action ON UPDATE Cascade,
	CONSTRAINT FK_Product_status_type_language FOREIGN KEY (language_code)
	REFERENCES universal_with_type.Language (language_code) ON DELETE No
	Action ON UPDATE Cascade,
	CONSTRAINT FK_Product_status_type_t_Product_status_type FOREIGN KEY
	(product_status_type_code) REFERENCES
	universal_with_type.Product_status_type (product_status_type_code) ON
	DELETE Cascade ON UPDATE Cascade
);

CREATE TABLE universal_with_type.Product (
	product_id serial,
	product_code domain.product_code,
	ean_code domain.ean_code,
	product_status_type_code smallint NOT NULL DEFAULT 1,
	time_added timestamp NOT NULL DEFAULT localtimestamp(0),
	price domain.product_price,
	CONSTRAINT PK_Product PRIMARY KEY (product_id),
	CONSTRAINT AK_Ean_code_is_unique UNIQUE (ean_code),
	CONSTRAINT AK_Product_code_is_unique UNIQUE (product_code),
	CONSTRAINT FK_Product_status FOREIGN KEY (product_status_type_code)
	REFERENCES universal_with_type.Product_status_type
	(product_status_type_code) ON DELETE No Action ON UPDATE Cascade
);
```

```
CREATE TABLE universal_with_type.Product_translation (
	product_id integer NOT NULL,
	language_code char(3)	NOT NULL,
	content_type_code smallint NOT NULL,
	value domain.value,
	CONSTRAINT PK_Product_translation PRIMARY KEY
	(product_id,language_code,content_type_code),
	CONSTRAINT FK_Product FOREIGN KEY (product_id) REFERENCES
	universal_with_type.Product (product_id) ON DELETE Cascade ON UPDATE
	No Action,
	CONSTRAINT FK_Product_translation_language FOREIGN KEY (language_code)
	REFERENCES universal_with_type.Language (language_code) ON DELETE No
	Action ON UPDATE Cascade,
	CONSTRAINT FK_Product_translation_content_type FOREIGN KEY
	(content_type_code) REFERENCES universal_with_type.Content_type
	(content_type_code) ON DELETE No Action ON UPDATE Cascade
);
```

Indeksite loomise laused:

```
CREATE INDEX IXFK_Product_status_type_content_type ON
universal_with_type.Product_status_type_translation (content_type_code ASC);

CREATE INDEX IXFK_Product_status_type_language ON
universal_with_type.Product_status_type_translation (language_code ASC);

CREATE INDEX IXFK_Product_status ON universal_with_type.Product
(product_status_type_code ASC);

CREATE INDEX IXFK_Product_translation_language ON
universal_with_type.Product_translation (language_code ASC);

CREATE INDEX IXFK_Product_translation_content_type ON
universal_with_type.Product_translation (content_type_code ASC);
```

Andmete lisamise laused:

```
INSERT INTO universal_with_type.Language (language_code, name) VALUES
	('est', 'eesti keel'),
	('eng', 'inglise keel');

INSERT INTO universal_with_type.Content_type (content_type_code, name,
description) VALUES
	('1', 'name', 'Toote või tema seisundi nimetus.'),
	('2', 'description', 'Toote või tema seisundi kirjeldus.');
```

```
INSERT INTO universal_with_type.Product_status_type
(product_status_type_code) VALUES
	(0),
	(1),
	(2),
	(3);


INSERT INTO universal_with_type.Product_status_type_translation
(product_status_type_code, language_code, content_type_code, value) VALUES
	(0, 'est', 2, 'Toode on arhiveeritud ja seda ei saa osta ega müüa.'),
	(1, 'est', 2, 'Toode on aktiivne ja seda saab nii osta kui müüa.'),
	(2, 'est', 2, 'Toodet ei saa enam lattu tellida.'),
	(3, 'est', 2, 'Toode ei ole müügiks.'),
	(0, 'est', 1, 'arhiveeritud'),
	(0, 'eng', 1, 'archived'),
	(1, 'est', 1, 'aktiivne'),
	(1, 'eng', 1, 'active'),
	(2, 'est', 1, 'ei saa enam tellida'),
	(2, 'eng', 1, 'no longer ordered'),
	(3, 'est', 1, 'ei ole müügiks'),
	(3, 'eng', 1, 'not for sale');


INSERT INTO universal_with_type.Product (product_code, ean_code, price)
VALUES
	('K-PAULIG-J', '6411300158072', 4.99),
	('K-TERE', '4740125110012', 0.31),
	('SUH-NOR', '6414000023138', 0.89),
	('KT-SOK', '4742934010032', 4.29);


INSERT INTO universal_with_type.Product_translation (product_id,
language_code, content_type_code, value) VALUES
	(1, 'est', 2, 'Hõrk kauakestva ja rikkaliku järelmaitsega kohv.'),
	(1, 'eng', 2, 'Delicate coffee with long-lasting and rich
	aftertaste.'),
	(3, 'est', 2, 'Valge suhkur. Sobib küpsetamiseks, hoidiste
	valmistamiseks, magustoitude tegemiseks ja kohvi sisse.'),
	(3, 'eng', 2, 'White sugar. Suitable for baking, making preserves,
	desserts and into coffee.'),
	(1, 'est', 1, 'Jahvatatud kohv'),
	(1, 'eng', 1, 'Ground coffee'),
	(2, 'est', 1, 'Koor 10%'),
	(2, 'eng', 1, 'Cream 10%'),
	(3, 'est', 1, 'Suhkur'),
	(3, 'eng', 1, 'Sugar'),
	(4, 'est', 1, 'Käsitöö šokolaad'),
	(4, 'eng', 1, 'Handmade chocolate'),
	(4, 'est', 2, 'Käsitööna valmistatud luksuslik šokolaad.'),
	(4, 'eng', 2, 'Luxurious handmade chocolate.');
```

Andmete muutmise laused:

```
UPDATE universal_with_type.Product
	SET price = 0.99,
		product_status_type_code = 2
	WHERE product_code = 'SUH-NOR';

UPDATE universal_with_type.Product_translation
	SET value = 'Demerara suhkur'
	WHERE product_id = 3
		AND language_code = 'est'
		AND content_type_code = '1';
```

Muster „Tõlkeread“:

Skeemi loomise lause:

```
CREATE SCHEMA language_rows;
```

Tabelite loomise laused:

```
CREATE TABLE language_rows.Language (
	language_code domain.language_code,
	name domain.name NOT NULL,
	CONSTRAINT PK_Language PRIMARY KEY (language_code),
	CONSTRAINT AK_Language_name_is_unique UNIQUE (name)
);

CREATE TABLE language_rows.Product_status_type (
	product_status_type_code smallint NOT NULL,
	language_code char(3)	NOT NULL,
	name domain.name NOT NULL,
	description domain.description,
	CONSTRAINT PK_Product_status_type PRIMARY KEY
	(product_status_type_code,language_code),
	CONSTRAINT AK_Product_status_type_name_language_code_is_unique UNIQUE
	(name,language_code),
	CONSTRAINT FK_Product_status_type_language FOREIGN KEY (language_code)
	REFERENCES language_rows.Language (language_code) ON DELETE No Action
	ON UPDATE Cascade
);
```

```
CREATE TABLE language_rows.Product (
	product_id serial,
	language_code char(3)	NOT NULL,
	product_code domain.product_code,
	ean_code domain.ean_code,
	product_status_type_code smallint NOT NULL DEFAULT 1,
	price domain.product_price,
	time_added timestamp NOT NULL DEFAULT localtimestamp(0),
	name domain.product_name NOT NULL,
	description domain.product_description,
	CONSTRAINT PK_Product PRIMARY KEY (product_id),
	CONSTRAINT AK_Ean_code_language_code_are_unique UNIQUE
	(ean_code,language_code),
	CONSTRAINT AK_Product_code_language_code_are_unique UNIQUE
	(product_code,language_code),
	CONSTRAINT FK_Product_language FOREIGN KEY (language_code) REFERENCES
	language_rows.Language (language_code) ON DELETE No Action ON UPDATE
	Cascade,
	CONSTRAINT FK_Product_status FOREIGN KEY
	(product_status_type_code,language_code) REFERENCES
	language_rows.Product_status_type
	(product_status_type_code,language_code) ON DELETE No Action ON UPDATE
	Cascade
);
```

Indeksite loomise laused:

```
CREATE INDEX IXFK_Product_status_type_language ON
language_rows.Product_status_type (language_code ASC);

CREATE INDEX IXFK_Product_language ON language_rows.Product (language_code
ASC);

CREATE INDEX IXFK_Product_status ON language_rows.Product
(product_status_type_code ASC,language_code ASC);
```

Andmete lisamise laused:

```
INSERT INTO language_rows.Language (language_code, name) VALUES
	('est', 'eesti keel'),
	('eng', 'inglise keel');
```

```
INSERT INTO language_rows.Product_status_type (product_status_type_code,
language_code, name, description) VALUES
	(0, 'est', 'arhiveeritud', 'Toode on arhiveeritud ja seda ei saa osta
	ega müüa.'),
	(1, 'est', 'aktiivne', 'Toode on aktiivne ja seda saab nii osta kui
	müüa.'),
	(2, 'est', 'ei saa enam tellida', 'Toodet ei saa enam lattu
	tellida.'),
	(3, 'est', 'ei ole müügiks', 'Toode ei ole müügiks.'),
	(0, 'eng', 'archived', NULL),
	(1, 'eng', 'active', NULL),
	(2, 'eng', 'no longer ordered', NULL),
	(3, 'eng', 'not for sale', NULL);

INSERT INTO language_rows.Product (language_code, product_code, ean_code,
price, name, description) VALUES
	('est', 'K-PAULIG-J', '6411300158072', 4.99, 'Jahvatatud kohv', 'Hõrk
	kauakestva ja rikkaliku järelmaitsega kohv.'),
	('est', 'K-TERE', '4740125110012', 0.31, 'Koor 10%', NULL),
	('est', 'SUH-NOR', '6414000023138', 0.89, 'Suhkur', 'Valge suhkur.
	Sobib küpsetamiseks, hoidiste valmistamiseks, magustoitude tegemiseks
	ja kohvi sisse.'),
	('est', 'KT-SOK', '4742934010032', 4.29, 'Käsitöö šokolaad',
	'Käsitööna valmistatud luksuslik šokolaad.'),
	('eng', 'K-PAULIG-J', '6411300158072', 4.99, 'Ground coffee',
	'Delicate coffee with long-lasting and rich aftertaste.'),
	('eng', 'K-TERE', '4740125110012', 0.31, 'Cream 10%', NULL),
	('eng', 'SUH-NOR', '6414000023138', 0.89, 'Sugar', 'White sugar.
	Suitable for baking, making preserves, desserts and into coffee.'),
	('eng', 'KT-SOK', '4742934010032', 4.29, 'Handmade chocolate',
	'Luxurious handmade chocolate.');
```

Andmete muutmise laused:

```
UPDATE language_rows.Product
	SET price = 0.99,
		product_status_type_code = 2
	WHERE product_code = 'SUH-NOR';

UPDATE language_rows.Product
	SET name = 'Demerara suhkur'
	WHERE product_code = 'SUH-NOR'
		AND language_code = 'est';
```

Muster „Tõlkeveerud“:

Skeemi loomise lause:

```
CREATE SCHEMA language_columns;
```

Tabelite loomise laused:

```
CREATE TABLE language_columns.Product_status_type (
       product_status_type_code smallint NOT NULL,
       name_est domain.name NOT NULL COLLATE "et_EE",
       name_eng domain.name NOT NULL COLLATE "en_US",
       name_deu domain.name NOT NULL COLLATE "de_DE",
       description_est domain.description COLLATE "et_EE",
       description_eng domain.description COLLATE "en_US",
       description_deu domain.description COLLATE "de_DE",
       CONSTRAINT PK_Product_status_type PRIMARY KEY
       (product_status_type_code),
       CONSTRAINT AK_Product_status_type_name_eng_is_unique UNIQUE
       (name_eng),
       CONSTRAINT AK_Product_status_type_name_deu_is_unique UNIQUE
       (name_deu),
       CONSTRAINT AK_Product_status_type_name_est_is_unique UNIQUE (name_est)
);

CREATE TABLE language_columns.Product (
       product_id serial,
       product_code domain.product_code,
       ean_code domain.ean_code,
       product_status_type_code smallint NOT NULL DEFAULT 1,
       price domain.product_price,
       time_added timestamp NOT NULL DEFAULT localtimestamp(0),
       name_est domain.product_name NOT NULL COLLATE "et_EE",
       name_eng domain.product_name NOT NULL COLLATE "en_US",
       name_deu domain.product_name NOT NULL COLLATE "de_DE",
       description_est domain.product_description COLLATE "et_EE",
       description_eng domain.product_description COLLATE "en_US",
       description_deu domain.product_description COLLATE "de_DE",
       CONSTRAINT PK_Product PRIMARY KEY (product_id),
       CONSTRAINT AK_Ean_code_is_unique UNIQUE (ean_code),
       CONSTRAINT AK_Product_code_is_unique UNIQUE (product_code),
       CONSTRAINT FK_Product_status FOREIGN KEY (product_status_type_code)
       REFERENCES language_columns.Product_status_type
       (product_status_type_code) ON DELETE No Action ON UPDATE Cascade
);
```

Indeksite loomise lause:

```
CREATE INDEX IXFK_Product_status ON language_columns.Product
(product_status_type_code ASC);
```

Andmete lisamise laused:

```
INSERT INTO language_columns.Product_status_type (product_status_type_code,
name_est, name_eng, name_deu, description_est) VALUES
	(0, 'arhiveeritud', 'archived', 'archiviert', 'Toode on arhiveeritud
	ja seda ei saa osta ega müüa.'),
	(1, 'aktiivne', 'active', 'aktiv', 'Toode on aktiivne ja seda saab nii
	osta kui müüa.'),
	(2, 'ei saa enam tellida', 'no longer ordered', 'nicht mehr bestellt',
	'Toodet ei saa enam lattu tellida.'),
	(3, 'ei ole müügiks', 'not for sale', 'nicht zu verkaufen', 'Toode ei
	ole müügiks.');

INSERT INTO language_columns.Product (product_code, ean_code, price,
name_est, name_eng, name_deu, description_est, description_eng) VALUES
	('K-PAULIG-J', '6411300158072', 4.99, 'Jahvatatud kohv', 'Ground
	coffee', 'gemahlenen Kaffee', 'Hõrk kauakestva ja rikkaliku
	järelmaitsega kohv.', 'Delicate coffee with long-lasting and rich
	aftertaste.'),
	('K-TERE', '4740125110012', 0.31, 'Koor 10%', 'Cream 10%', 'Sahne
	10%', NULL, NULL),
	('SUH-NOR', '6414000023138', 0.89, 'Suhkur', 'Sugar', 'Zucker', 'Valge
	suhkur. Sobib küpsetamiseks, hoidiste valmistamiseks, magustoitude
	tegemiseks ja kohvi sisse.', 'White sugar. Suitable for baking, making
	preserves, desserts and into coffee.'),
	('KT-SOK', '4742934010032', 4.29, 'Käsitöö šokolaad', 'Handmade
	chocolate', 'Handgemachte Schokolade', 'Käsitööna valmistatud
	luksuslik šokolaad.', 'Luxurious handmade chocolate.');
```

Andmete muutmise lause:

```
UPDATE language_columns.Product
	SET price = 0.99,
		product_status_type_code = 2,
		name_est = 'Demerara suhkur'
	WHERE product_code = 'SUH-NOR';
```

Muster „Universaalne tõlketabel olemitüübi kohta“:

Skeemi loomise lause:

```
CREATE SCHEMA universal;
```

Tabelite loomise laused:

```
CREATE TABLE universal.Language (
       language_code domain.language_code,
       name domain.name NOT NULL,
       CONSTRAINT PK_Language PRIMARY KEY (language_code),
       CONSTRAINT AK_Language_name_is_unique UNIQUE (name)
);

CREATE TABLE universal.Product_status_type (
       product_status_type_code smallint NOT NULL,
       CONSTRAINT PK_Product_status_type PRIMARY KEY
       (product_status_type_code)
);

CREATE TABLE universal.Product_status_type_translation (
       product_status_type_code smallint NOT NULL,
       language_code char(3)      NOT NULL,
       name domain.name NOT NULL,
       description domain.description,
       CONSTRAINT PK_Product_status_type_translation PRIMARY KEY
       (product_status_type_code,language_code),
       CONSTRAINT AK_Product_status_type_name_language_code_are_unique UNIQUE
       (name,language_code),
       CONSTRAINT FK_Product_status_type_translation_language FOREIGN KEY
       (language_code) REFERENCES universal.Language (language_code) ON
       DELETE No Action ON UPDATE Cascade,
       CONSTRAINT FK_Product_status_type_translation_Product_status_type
       FOREIGN KEY (product_status_type_code) REFERENCES
       universal.Product_status_type (product_status_type_code) ON DELETE
       Cascade ON UPDATE Cascade
);

CREATE TABLE universal.Product (
       product_id serial,
       product_code domain.product_code,
       ean_code domain.ean_code,
       product_status_type_code smallint NOT NULL DEFAULT 1,
       price domain.product_price,
       time_added timestamp NOT NULL DEFAULT localtimestamp(0),
       CONSTRAINT PK_Product PRIMARY KEY (product_id),
       CONSTRAINT AK_Ean_code_is_unique UNIQUE (ean_code),
       CONSTRAINT AK_Product_code_is_unique UNIQUE (product_code),
       CONSTRAINT FK_Product_status FOREIGN KEY (product_status_type_code)
       REFERENCES universal.Product_status_type (product_status_type_code) ON
       DELETE No Action ON UPDATE Cascade
);
```

```
CREATE TABLE universal.Product_translation (
	product_id integer NOT NULL,
	language_code char(3)	NOT NULL,
	name domain.product_name NOT NULL,
	description domain.product_description,
	CONSTRAINT PK_Product_translation PRIMARY KEY
	(product_id,language_code),
	CONSTRAINT FK_Product FOREIGN KEY (product_id) REFERENCES
	universal.Product (product_id) ON DELETE Cascade ON UPDATE No Action,
	CONSTRAINT FK_Product_translation_language FOREIGN KEY (language_code)
	REFERENCES universal.Language (language_code) ON DELETE No Action ON
	UPDATE Cascade
);
```

Indeksite loomise laused:

```
CREATE INDEX IXFK_Product_status_type_translation_language ON
universal.Product_status_type_translation (language_code ASC);

CREATE INDEX IXFK_Product_status ON universal.Product
(product_status_type_code ASC);

CREATE INDEX IXFK_Product_translation_language ON
universal.Product_translation (language_code ASC);
```

Andmete lisamise laused:

```
INSERT INTO universal.Language (language_code, name) VALUES
	('est', 'eesti keel'),
	('eng', 'inglise keel');

INSERT INTO universal.Product_status_type (product_status_type_code) VALUES
	(0),
	(1),
	(2),
	(3);

INSERT INTO universal.Product_status_type_translation
(product_status_type_code, language_code, name, description) VALUES
	(0, 'est', 'arhiveeritud', 'Toode on arhiveeritud ja seda ei saa osta
	ega müüa.'),
	(1, 'est', 'aktiivne', 'Toode on aktiivne ja seda saab nii osta kui
	müüa.'),
	(2, 'est', 'ei saa enam tellida', 'Toodet ei saa enam lattu
	tellida.'),
	(3, 'est', 'ei ole müügiks', 'Toode ei ole müügiks.'),
	(0, 'eng', 'archived', NULL),
	(1, 'eng', 'active', NULL),
	(2, 'eng', 'no longer ordered', NULL),
	(3, 'eng', 'not for sale', NULL);
```

```
INSERT INTO universal.Product (product_code, ean_code, price) VALUES
	('K-PAULIG-J', '6411300158072', 4.99),
	('K-TERE', '4740125110012', 0.31),
	('SUH-NOR', '6414000023138', 0.89),
	('KT-SOK', '4742934010032', 4.29);


INSERT INTO universal.Product_translation (product_id, language_code, name,
description) VALUES
	(1, 'est', 'Jahvatatud kohv', 'Hõrk kauakestva ja rikkaliku
	järelmaitsega kohv.'),
	(1, 'eng', 'Ground coffee', 'Delicate coffee with long-lasting and
	rich aftertaste.'),
	(3, 'est', 'Suhkur', 'Valge suhkur. Sobib küpsetamiseks, hoidiste
	valmistamiseks, magustoitude tegemiseks ja kohvi sisse.'),
	(3, 'eng', 'Sugar', 'White sugar. Suitable for baking, making
	preserves, desserts and into coffee.'),
	(2, 'est', 'Koor 10%', NULL),
	(2, 'eng', 'Cream 10%', NULL),
	(4, 'est', 'Käsitöö šokolaad', 'Käsitööna valmistatud luksuslik
	šokolaad.'),
	(4, 'eng', 'Handmade chocolate', 'Luxurious handmade chocolate.');
```

Andmete muutmise laused:

```
UPDATE universal.Product
	SET price = 0.99,
		product_status_type_code = 2
	WHERE product_code = 'SUH-NOR';

UPDATE universal.Product_translation
	SET name = 'Demerara suhkur'
	WHERE product_id = 3
		AND language_code = 'est';
```

Lisa 2 – Otsustustabelid

Mustrid on allpool esitatud tabelites tähistatud nagu Tabel 3:

Tabel 8. Päringute keerukuse hinnangud.

	G	**H**	**I**	**J**	**K**	**L**	**M**	**Kokku**
G	1.00	1.00	5.00	3.00	7.00	2.00	5.00	0.302
H	1.00	1.00	4.00	2.00	6.00	1.00	4.00	0.237
I	0.20	0.25	1.00	0.50	3.00	0.50	1.00	0.069
J	0.33	0.50	2.00	1.00	5.00	1.00	2.00	0.132
K	0.14	0.17	0.33	0.20	1.00	0.20	0.33	0.030
L	0.50	1.00	2.00	1.00	5.00	1.00	3.00	0.164
M	0.20	0.25	1.00	0.50	3.00	0.33	1.00	0.065

Tabel 9. Andmete sisestamise keerukuse hinnangud.

	G	**H**	**I**	**J**	**K**	**L**	**M**	**Kokku**
G	1.00	3.00	7.00	4.00	5.00	4.00	5.00	0.403
H	0.33	1.00	4.00	2.00	3.00	2.00	3.00	0.191
I	0.14	0.25	1.00	0.33	0.50	0.33	0.50	0.041
J	0.25	0.50	3.00	1.00	2.00	1.00	2.00	0.115
K	0.20	0.33	2.00	0.50	1.00	0.50	1.00	0.067
L	0.25	0.50	3.00	1.00	2.00	1.00	2.00	0.115
M	0.20	0.33	2.00	0.50	1.00	0.50	1.00	0.067

Tabel 10. Andmebaasi loomise keerukuse hinnangud.

	G	**H**	**I**	**J**	**K**	**L**	**M**	**Kokku**
G	1.00	2.00	5.00	4.00	6.00	5.00	7.00	0.388
H	0.50	1.00	3.00	2.00	4.00	3.00	5.00	0.222
I	0.20	0.33	1.00	0.50	2.00	1.00	4.00	0.089
J	0.25	0.50	2.00	1.00	3.00	1.00	3.00	0.120

	G	H	I	J	K	L	M	Kokku
K	0.17	0.25	0.50	0.33	1.00	0.50	2.00	0.052
L	0.20	0.33	1.00	1.00	2.00	1.00	3.00	0.093
M	0.14	0.20	0.25	0.33	0.50	0.33	1.00	0.036

Tabel 11. Andmete liiasuse olemasolu hinnangud.

	G	H	I	J	K	L	M	Kokku
G	1.00	5.00	1.00	1.00	1.00	1.00	1.00	0.161
H	0.20	1.00	0.20	0.20	0.20	0.20	0.20	0.032
I	1.00	5.00	1.00	1.00	1.00	1.00	1.00	0.161
J	1.00	5.00	1.00	1.00	1.00	1.00	1.00	0.161
K	1.00	5.00	1.00	1.00	1.00	1.00	1.00	0.161
L	1.00	5.00	1.00	1.00	1.00	1.00	1.00	0.161
M	1.00	5.00	1.00	1.00	1.00	1.00	1.00	0.161

Tabel 12. Uue keele toe lisamise keerukuse hinnangud.

	G	H	I	J	K	L	M	Kokku
G	1.00	0.14	0.14	0.14	0.14	3.00	0.14	0.032
H	7.00	1.00	1.00	1.00	1.00	9.00	1.00	0.190
I	7.00	1.00	1.00	1.00	1.00	9.00	1.00	0.190
J	7.00	1.00	1.00	1.00	1.00	9.00	1.00	0.190
K	7.00	1.00	1.00	1.00	1.00	9.00	1.00	0.190
L	0.33	0.11	0.11	0.11	0.11	1.00	0.11	0.019
M	7.00	1.00	1.00	1.00	1.00	9.00	1.00	0.190

Tabel 13. Kitsenduste rakendamise ja andmetüübi määramise keerukuse hinnangud

	G	H	I	J	K	L	M	Kokku
G	1.00	1.00	7.00	1.00	7.00	1.00	1.00	0.189
H	1.00	1.00	7.00	1.00	7.00	1.00	1.00	0.189
I	0.14	0.14	1.00	0.14	1.00	0.14	0.14	0.027
J	1.00	1.00	7.00	1.00	7.00	1.00	1.00	0.189

	G	**H**	**I**	**J**	**K**	**L**	**M**	**Kokku**
K	0.14	0.14	1.00	0.14	1.00	0.14	0.14	0.027
L	1.00	1.00	7.00	1.00	7.00	1.00	1.00	0.189
M	1.00	1.00	7.00	1.00	7.00	1.00	1.00	0.189