TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Informaatikainstituut

IDU40LT

Susanna Peek 135207 IAPB

DENORMALISEERIMISE PRAKTIKA UURIMINE ÜHE SQL-ANDMEBAASI NÄITEL

bakalaureusetöö

Juhendaja: Erki Eessaar
Doktor
Dotsent

Tallinn 2016

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: Susanna Peek

23.05.2016

Annotatsioon

Andmebaaside denormaliseerimise peamiseks eesmärgiks on päringute töökiiruse parandamine kuna denormaliseerimise tulemusena ühendatakse tabeleid või dubleeritakse veerge, et andmete otsimisel ei peaks läbi viima palju tabelite ühendamise operatsioone. Päringute töökiirus (jõudlus) on üks paljudest andmebaasi kvaliteedikriteeriumitest. Antud lõputöö eesmärgiks on praktiliste katsete najal saada teada, millist laiemat mõju omab tabelite denormaliseerimine andmebaasi kasutamisele SQL-andmebaasisüsteemides. Arvestades töö mahtu tehakse katsetusi ühes andmebaasisüsteemis, milleks on valitud tasuta, avatud lähtekoodiga ja populaarne andmebaasisüsteem PostgreSQL (9.4).

Töö oluliseks tulemuseks on kaks erinevat andmebaasi disaini, nende põhjal loodud andmebaas ning nende skeemid, skeemide põhjal tehtud päringute ja kitsenduste näited ning nende disainide võrdlemise tulemusena tehtud järeldused.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 46 leheküljel, 6 peatükki, 21 joonist, 7 tabelit.

Abstract

Investigating the Denormalization Practice in the Example of a SQL Database

The main goal of database denormalization is to improve the speed in which the database management system carries out queries. In order to avoid joining many tables when searching for the data, the denormalization creates combined tables and duplicated columns. It turns back the further normalization process that designers perform to to reduce data redundancy and increase the understandability of the schema. The query speed (performance) is one of many data quality criterions. The aim of this thesis is to find out by using practical tests the broader impact that denormalization of tables has on usage of databases. Considering the expected amount of work, we conduct the experiments based on one database management system (DBMS). We have selected free, open source, and popular DBMS PostgreSQL (9.4) for the experiments.

We compare two database designs that have base tables (tables in short) normalized to different levels to achieve the aims. The designs are compared in order to see how the denormalization of tables affects database tables in terms of the data size, queries and data changes processing speed, and the complexity of queries and constraints in the database. One of the compared databases is denormalized based on the algorithm described in Steve Hobermann's book „Data Modeler's Workbench" and the other database is fully normalized, meaning that all its tables are in fifth normal form.

Important outcomes of the thesis are two different database designs, database and schemas that we created based on these designs, examples of queries and constraints, and the resulting conclusions based on these two designs.

The thesis is in Estonian language and contains 46 pages of text, 6 chapters, 21 figures, 7 tables.

Lühendite ja mõistete sõnastik

5NK	Viies normaalkuju Normaalkuju, mille korral on andmete liiasus normaliseerimise teooria seisukohalt viidud miinimumini. SQL-andmebaasi loogiline disain võiks ideaalis saavutada tabelite kirjeldused, kus kõik tabelid on viiendal normaalkujul.
5NK andmebaas	Täielikult normaliseeritud andmebaas. Antud töös tähistab see SQL-andmebaasi, milles kõik baastabelid on vähemalt viiendal normaalkujul.
CASE	*Computer-Aided Software Engineering* Tarkvara, mida kasutatakse infosüsteemide ja andmebaaside kavandamiseks ning võimalik, et ka kavandist lähtekoodi genereerimiseks.
SLOC-P	Füüsiliste käivitatavate koodiridade arv Arvutatakse lahutades koodiridade koguarvust kommenaaride ja tühjade ridade arvu. Seda mõõdikut kasutatakse lähtekoodi mahukuse hindamiseks.
SQL	*Structured SQL Query Language* Andmebaasikeel, mis põhineb relatsioonilisel andmemudelil ning mida kasutatakse andmete, õiguste ning andmestruktuuride haldamiseks. Antud töös peetakse SQL-andmebaasi all silmas andmebaasi, mis on loodud kasutades andmebaasisüsteemi, milles kasutatakse SQL andmebaasikeelt.
UML	*Unified Modeling Language* Üldotstarbeline (st paljudes eri valdkondades kasutatav), visuaalne modelleerimiskeel, mida kasutatakse tarkvara struktuuri ning käitumise visualiseerimiseks.
Tabel	Käesoleva töö tähenduses SQL-andmebaasi baastabel. Selline tabel ei ole defineeritud teiste tabelite põhjal.

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Tänapäeval on erinevate organisatsioonide infotehnoloogia toega infosüsteemides saanud andmebaasisüsteemide abil loodud andmebaasid üheks olulisemaks komponendiks. Andmebaase kasutatakse infosüsteemides andmete talletamiseks ja haldamiseks. Endiselt on lõpuni vastamata küsimus kuidas tuleks disainida andmebaas nii, et selle kasutamine oleks võimalikult efektiivne. Efektiivsuse kriteeriumid (nt, et andmebaasioperatsioonide tegemine oleks kõige kasutajamugavam, kiirem ja optimaalsem) võivad eri osapooltele ja eri olukordades olla erinevad. Antud töös vaadeldakse kahte erineva tasemeni normaliseeritud andmebaasi disaini SQL-andmebaasis. Seda tehakse, et saada teada kuidas mõjutab tabelite denormaliseerimine ja normaliseerimine PostgreSQL andmebaasides andmemahtusid, päringute töökiirust, andmemuudatuste kiirust ning päringute ja kitsenduste keerukust.

Andmebaaside denormaliseerimise peamiseks eesmärgiks on päringute töökiiruse parandamine kuna denormaliseerimise tulemusena ühendatakse tabeleid või dubleeritakse veerge, et andmete otsimisel ei peaks läbi viima palju tabelite ühendamise operatsioone. Päringute töökiirus on üks paljudest andmebaasi kvaliteedikriteeriumitest. Antud lõputöö eesmärgiks on praktiliste katsete najal saada teada, millist laiemat mõju omab tabelite denormaliseerimine andmebaasi kasutamisele.

Töös katseteks kasutatavaks andmebaasiks on autori poolt ainetes „Andmebaasid I“ ja „Andmebaasid II“ tehtud projekti „Ülikooli infosüsteemi õppeainete arvestuse allsüsteem“ andmebaas, mille kaasautoriteks on Maria Ossipova ja Helen Pikkaro [1]. Antud infosüsteemi konseptuaalse andmemudeli põhjal modelleeritakse kaks infosüsteemi nõuetele vastavat andmebaasi disaini kasutades selleks CASE vahendit Rational Rose [2]. Esimene andmebaas on normaliseeritud viienda normaalkujuni ning teine andmebaas on denormaliseeritud algoritmi järgi, mida kirjeldab Steve Hobermann oma raamatus „Data Modeler's Workbench: Tools and Techniques for Analysis and Design 1st Edition“ [3]. Andmebaasid realiseeritakse PostgreSQL (9.4)

andmebaasisüsteemis. Järgnevalt täidetakse mõlemad andmebaasid ühesuguste testandmetega. Testandmed genereeritakse programmiga ApexSQL Generate abil. Seejärel viiakse läbi erinevaid teste objektiivse hinnangu saamiseks. Esiteks mõõdetakse mõlema andmebaasi andmemahtusid. Teiseks on koostatud nii koondandmete otsimise päring kui ka konkreetse olemi otsimise päringuid, mida realiseeritakse eraldi mõlema andmebaasi jaoks. Päringute puhul jälgitakse nende täitmiskiirust ning keerukust. Kolmandaks mõõdetakse konkreetse olemi andmete muudatuse kiirust ning keerukust. Neljandaks mõõdetakse kitsenduste jõustamise keerukust mõlemas andmebaasis. Saadud tulemusi võrreldakse, et saada teada, millist mõju omab tabelite denormaliseerimine andmebaasi kasutamisele.

Antud töö koosneb kuuest suuremast sisupeatükist, nendest esimeses peatükis „Teoreetiline taust“ tutvustatakse lühidalt normaliseerimise ja denormaliseerimise mõistet. Järgmine peatükk „Eksperimendi andmebaasi projekteerimine“ sisaldab eksperimendiks kasutatavate andmebaaside disaini, mis on võrreldavuse tagamiseks esitatud ühesugust struktuuri kasutades. Kolmas peatükk „Eksperimendi kirjeldus“ tutvustab antud töös tehtavat eksperimenti ja selle eesmärke. Antud peatükis antakse ka ülevaate kasutavatest andmebaasisüsteemidest, testandmete genereerimisest ja tehtavatest päringutest. Peatükk „Eksperimendi tulemused ja analüüs“ sisaldab tehtud eksperimendi tulemusi (andmebaaside andmemahte, päringute ning andmemuudatutste täitmiseks kulunud aega, päringute keerukust, kitsenduste jõustamise keerukust) ning nende analüüsi, mille põhjal esitatakse järeldused.

Kui süsteemide arendamise juures on üldse midagi kindlat siis see, et süsteemides tuleb teha muudatusi. Üldiselt on süsteemide muutmine keeruline. Eriti keeruline on muuta andmebaase, sest need on süsteemi kesksеteks osadeks, millest sõltuvad kõik teised osad (nt lähtekood, mudelid, testid). Muudatused, mille puhul tuleb hakata kohe tegema ringi disainiotsuseid, sest algsed otsused olid halvad, on ressursi raiskamine. Heade disainiotsuste langetamiseks on vaja objektiivseid andmeid. Konkreetsed katsetused koos arvuliste mõõtmistulemustega pakuvad selliseid andmeid. Edasi saab iga disainer juba ise otsustada, milline on antud juhul erinevate andmebaasi kvaliteedikriteeriumite olulisus ning teha sellest lähtuvaid disainivalikuid. Käesolev töö on kasulik PostgreSQLil põhinevate SQL-andmebaaside disainiotsuste tegijatele. See pakub erinevatele süsteemi kvaliteedi aspektidele vastavaid mõõtmistulemusi, mida disainerid saavad otsuste tegemisel põhjendusena kasutada.

2 Teoreetiline taust

Eesti õigekeelsus sõnaraamat defineerib normaliseerimist kui normile vastavaks tegemist [4]. Andmebaaside vallas peetakse selle all silmas kasutajatele andmete arusaadavamaks muutmist, liiasuse vähendamist ning huvide eristamise (*separation of concerns*) saavutamist. Huvide eristamine tähendab siinkohal, et andmestruktuuridel on normaliseerimise tulemusena üha kitsam oma „teema“. Denormaliseerimine on normaliseerimise vastasprotsess. Käesolevas peatükis antakse lühiülevaade andmebaaside normaliseerimisest ja denormaliseerimisest. Normaliseerimiseerimine ja denormaliseerimine on võimalik erinevate andmemudelite alusel loodud andmebaasides, kuid käesoleva töö teemaks on SQL-andmebaasid ja edaspidi mõeldakse normaliseerimise ning denormaliseerimise all just vastavat SQL-andmebaasis toimuvat protsessi.

2.1 Normaliseerimine

Normaliseerimine on protsess, mille igal sammul kontrollitakse normaliseeritava andmebaasi vastavust mingile reeglite hulgale. Normaliseerimise eesmärgiks on vähendada andmete liiasust ja selle tulemusena vähendada andmete muutmise anomaaliaid ning muuta andmebaasis olevad andmed kasutajale arusaadavamaks. Selleks, et kindlustada paremini andmete õigsus ja terviklikkus ning vähendada andmete liiasust on andmebaasi jaoks välja töötatud normaliseerimise protsess, mille käigus viiakse andmebaasi tabelid üha kõrgematele normaalkujudele. Siin ja edaspidi mõistetakse tabelite all baastabeleid.

Normaliseerimise protsessi pakkus esmakordselt välja E.F.Codd. Kolme normaalkuju on algselt kavandatud nimetama esimeseks normaalkujuks, teiseks normaalkujuks ja kolmandaks normaalkujuks. Järgnevalt tutvustasid R.Boyce ja E.F.Cood tugevamat definitsioon kolmandale normaalkujule nimetades selle Boyce-Codd normaalkujuks. Kõik need normaalkujud põhinevad tabeli veergude vahelistel funktsionaalsetel sõltuvustel. Kõrgema järgu normaalkujud, mis järgnevad Boyce-Codd normaalkujule

nagu neljas normaalkuju, mis põhineb multiväärtuslikel sõltuvustel ja viies normaalkuju, mis põhineb ühendamissõltuvustel tutvustati maailmale 1970-ndate lõpus R.Fagin poolt. [5] Aastal 2002 pakkusid H.Darwin, N.Lorenzo ja C.J.Date välja ka kuuenda normaalkuju. Tabelite sellele viimine ei aita vähendada andmete liiasust, kuid lihtsustab andmemuudatuste ajaloo säilitamist, puuduvate andmetega toimetulekut ja skeemimuudatuste tegemist. Kuuendal normaalkujul tabeleid luuakse näiteks andmebaaside ankurmodelleerimise tulemusena. [6]

SQL-andmebaas on nime omav tabelite kogum. SQL-andmebaasides peaksid tabelid olema viidud vähemalt esimesele normaalkujule. Sellisel juhul saab juba öelda, et tabel on normaliseeritud. Kui andmed pole organiseeritud vähemalt esimese normaalkuju reeglite järgi pole tegemist SQL-andmebaasi jaoks sobiva tabeliga. Iga andmebaasi tabeli saab viia kuuendale normaalkujule. Üldjuhul on soovitatav viia tabelid viienda normaalkujuni. [7] Tabeli viimist esimesest normaalkujust kõrgematele normaalkujudele nimetatakse täiendavaks normaliseerimiseks (sageli ka lihtsalt normaliseerimiseks).

Millised on normaalkujude omavahelised seosed?

- Kui tabel on normaalkujul N, siis on ta ka normaalkujul N-1, N-2 jne.
- Kui tabel on normaalkujul N, siis võib ta olla, aga ei pruugi ka normaalkujul N+1, N+2 jne.
- Tabel on mingil normaalkujul kujul kui ta rahuldab kõiki selle normaalkujuga seotud tingimusi. Kui tabelit soovitakse viia ühelt normaalkujult teisele, siis rakendatakse talle teisendusreegleid.
- Kõiki kitsendusi ja piiranguid, mida saab rakendada algsele tabelile, saab rakendada ka järgnevale normaalkujule viidud tabelile. [8] Praktikas muutub osade kitsenduste jõustamine lihtsamaks (neid saab jõustada tabelitele võtmeid deklareerides) ja osade jõustamine keerulisemaks (vaja on kirjutada mitut tabelit hõlmavaid avaldisi).

Kui räägitakse normaliseerimisest, siis enamasti mõeldakse selle all tabelite viimist esimesest kuni viienda normaalkujuni. Antud bakalaureusetöös on normaliseeritud üks eksperimendiks kasutatava andmebaas viienda normaalkujuni.

Andmebaas on täielikult normaliseeritud kui iga selles olev tabel on viiendal normaalkuju. Tabel on viiendal normaalkujul kui iga selles oleva mittetriviaalse ühendamissõltuvuse *{T1, T2, ..., Tn} korral on iga T1, T2, ..., Tn tabeli T supervõtmeks. Eelnevalt sõnastatud definitsioon tähendab, et kõik ülejäänud mittetriviaalsed ühendamissõltuvused peavad olema kõrvaldatud. Viiendat normaalkuju kutsutakse teisisõnu ka projektsiooni-ühendamise normaalkujuks. [7] Teiste sõnadega öeldes, viiendal normaalkujul olevat tabelit ei saa enam liiasuse vähendamise eesmärgil väiksemateks osadeks tükeldada.

2.2 Denormaliseerimine

Denormaliseerimine on normaliseerimise pöördprotsess. See tähendab, et vähendatakse ühe või mitme andmebaasis oleva tabeli normaliseerituse astet. Denormaliseerimise protsessi käigus dubleeritakse veerge või ühendatakse tabeleid, et tabelist andmeid otsides tuleks viia läbi vähem ühendamisoperatsioone. Andmebaasisüsteemilt nõuab ühendamise operatsiooni läbi viimine (sõltuvalt sellest, kuidas andmebaasisüsteem on realiseeritud) palju vaeva, sest ta peab läbi viima palju sisend/väljund operatsioone ja selle kasutamine võib mõjuda andmebaasist andmete otsimiseks mõeldud lausete töökiirusele halvasti. Üheks võimaluseks parandamaks päringute töökiirust on andmebaasi denormaliseerida. [9]

Probleemid, mis võivad tuleneda andmebaasi denormaliseerimisest.

- Normaliseerimise üheks eesmärgiks on andmete liisasuse vähendamine. Denormaliseerimine seevastu tekitab andmete liiasust, mis omakorda suurendab andmebaasi andmemahtu.
- Ilmnevad andme muudatustest tulenevad anomaaliad.
- Andmete liiasuse tõttu tuleb andmeid muuta mitmes erinevas tabelis või mitmes erinevas tabeli reas. Seetõttu denormaliseerimine võib suurendada andmete muutmiseks ning sisestamiseks kuluvat aega.

- Andmebaasis salvestatakse andmed sisemiselt lehekülgedel ehk plokkides, millel on fikseeritud suurus (Postgresis 8KB). Denormaliseerimine võib suurendada päringule vastuse saamiseks kuluvat aega juhul kui tabeli veergude arv läheb nii suureks, et üks rida ei mahu ära ühte plokki. Sellisel juhul jaotub rida mitme erineva ploki vahel ning andmete lugemiseks ja muutmiseks kulub rohkem aega. Samuti võib tekkida ka olukordi kus päringule vastuse leidmiseks tuleb vaadata läbi rohkem plokke.

- Denormaliseerimine võib põhjustada probleeme andmete kvaliteedis. Kui dubleeritud ridade puhul tehakse andmemuudatus ainult ühes reas, siis see tekitab vastuolu andmetes ja põhjustab andmete kvaliteedi vähenemise.

- Andmebaasi konseptuaalne skeem muutub kasutaja jaoks peale denormaliseerimist ebaselgemaks.

- Mõnede päringute ja andmebaasi kitsenduste kirjutamine muutub keerulisemaks.

- Denormaliseerimine vähendab süsteemi paindlikust ja laiendamise võimalusi.

Kunagi ei tuleks denormaliseerida ilma eelneva tabelite kõrge tasemeni (soovitavalt viienda normaalkujuni) normaliseerimiseta. Vastasel juhul jääb arusaamine andmetest ja nende vahelistest seostest poolikuks. [3]

Andmebaasi denormaliseerimist tuleks kaaluda vaid tabelite puhul, mida kasutakse sageli päringutes kuid andmete muudatused toimuvad harva nagu näiteks andmeaida ja andmevaka andmebaasides. Sellistes andmebaasides hoitakse nii ajaloolisi kui aktuaalseid andmeid, mille põhjal tehakse tihti keerukaid koondandmete päringuid. Seetõttu rakendatakse sellist tüüpi andmebaaside tabelite struktuuri leidmisel sageli denormaliseerimist. [9]

3 Eksperimendi andmebaasi projekteerimine

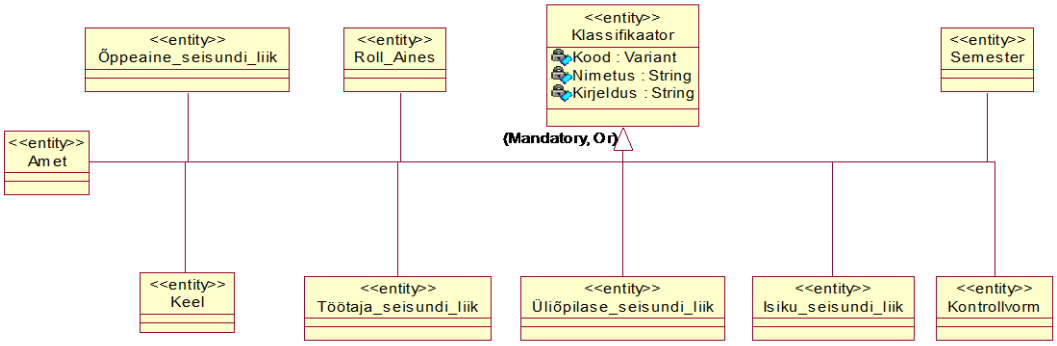
Eksperimendis kasutatavaks andmebaasiks on autori poolt TTÜ õppeainetes „Andmebaasid I“ ja „Andmebaasid II“ tehtud projekti „Ülikooli infosüsteemi õppeainete arvestuse allsüsteem“ andmebaas, mille kaasautoriteks on Maria Ossipova ja Helen Pikkaro [1].

Ülikooli infosüsteemi õppeainete arvestuse allsüsteemi eesmärgid

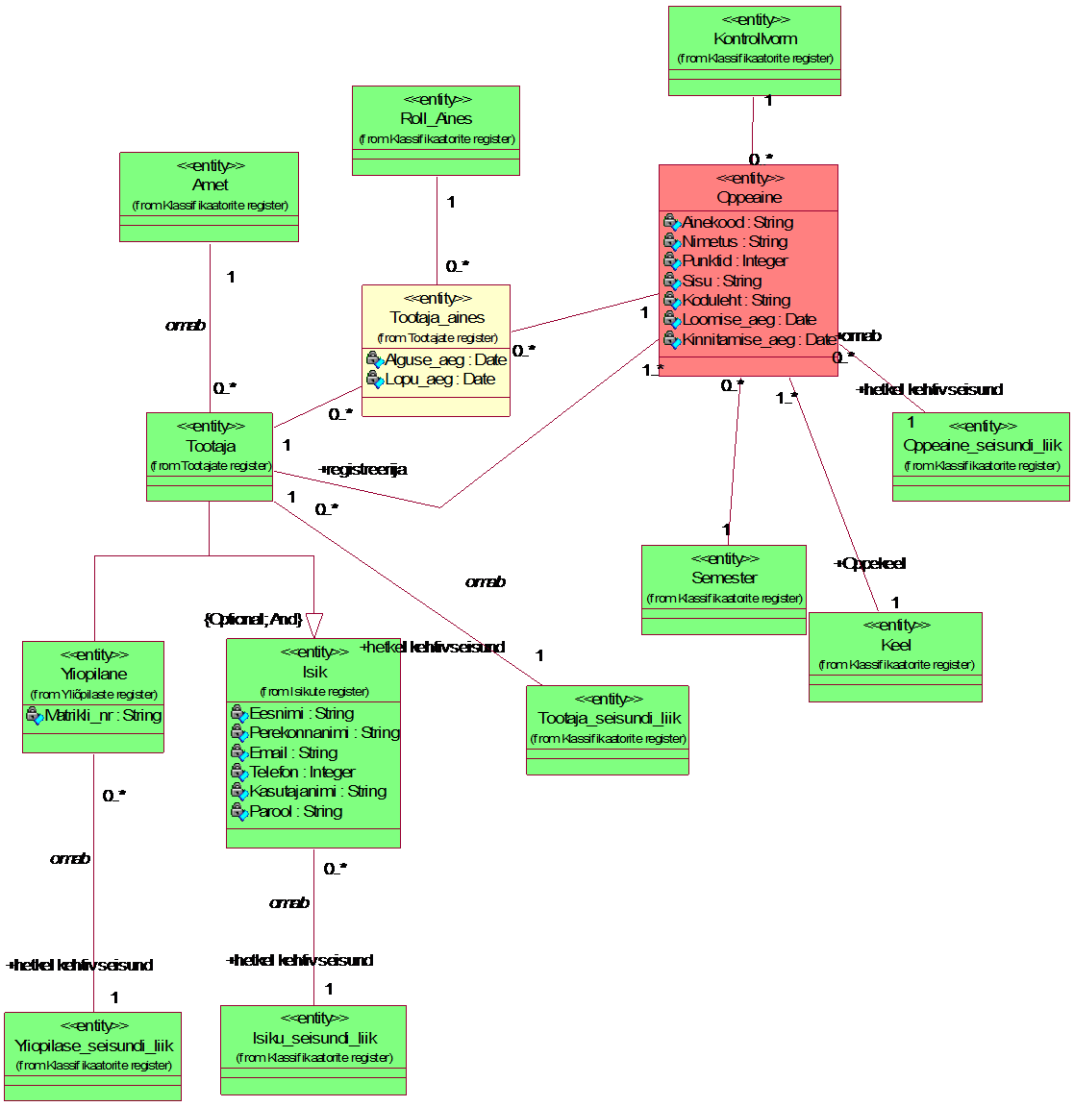
1. Võimaldada ainekaartide andmete registreerimist
2. Võimaldada õppejõududel tühistada õppeaine lisamist
3. Võimaldada õppejõududel elektrooniliselt registreerida õppeaineid
4. Võimaldada üliõpilastel, õppejõududel ja dekanaadi juhatajal teha päringuid õpetavate ainete kohta

3.1 Konseptuaalne andmemudel

Kontseptuaalne andmemudeli esitamiseks on kasutatud olemi-suhte diagrammi (vt joonis 1-2). Jooniste loomiseks on kasutatud CASE vahendit Rational Rose Enterprise Edition [2]. *Lisa 1* sisaldab diagrammil esitatud olemitüüpide ja olemitüüpide atribuutide kirjeldusi.



Joonis 1. Konseptuaalne andmemudel klassifikaatorite olemitüüpide olemi-suhte diagrammina.



Joonis 2. Konseptuaalne andmemudel olemitüübid olemi-suhte diagrammina.

Antud bakalaureusetöös on lisatud suuremate andmemahtude saavutamiseks andmebaasi disaini juurde olemitüüp nimega „Oppimine“, mis esitab soovi hoida andmeid üliõpilaste deklaratsioonide kohta. Sellele vastavad transaktsioonilased andmed, mille hulk on andmebaasis tavaliselt kõige suurem. Töö eesmärgiks ei ole ülikooli infosüsteemi detailne projekteerimine, vaid denormaliseerimise praktika katsetamine mille jaoks see andmebaas on lihtsalt vahend.

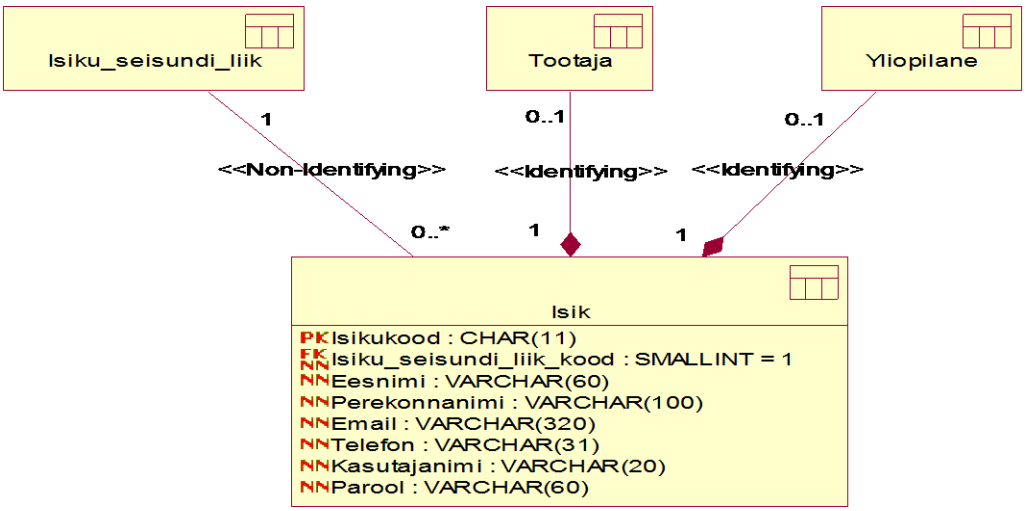
3.2 Viienda normaalkujuni normaliseeritud andmebaasi disain

Edaspidises töös kutsutakse sellist andmebaasi 5NK andmebaasiks. SQL-andmebaasi füüsilise disaini mudel luuakse kontseptuaalse andmemudeli alusel kasutades teisendusreegleid. Disaini mudelid on loodud eeldusel, et andmebaaside realiseerimiseks kasutatakse PostgreSQL andmebaasisüsteemi. Viis kuidas antud töös SQL-objekte nimetatakse on ära toodud *Lisas 2* olevas tabelis.

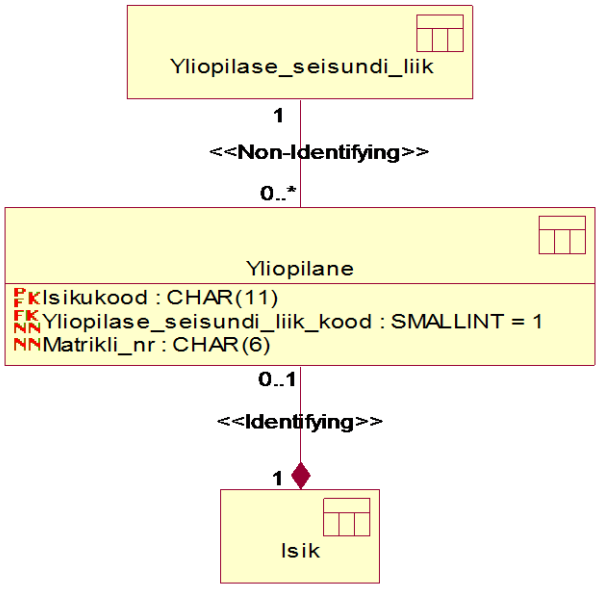
PostgreSQL loob primaarvõtme ja unikaalsuse kitsenduse alusel automaatselt indeksi. Tabelite täiendaval indekseerimisel peeti silmas, et ei loodaks üksteist dubleerivaid indekseid. Indeksid on andmebaasi sisemise taseme objektid, mis kiirendavad andmete otsimist. [8]

Välisvõtme määrangut ON UPDATE CASCADE ON DELETE CASCADE on kasutatud tabelites *Tootaja* ja *Yliopilane* välisvõtme (isikukood) puhul. Ülejäänud kõikide teiste välisvõtmete puhul on kasutatud määrangut ON UPDATE CASCADE ON DELETE NO ACTION.

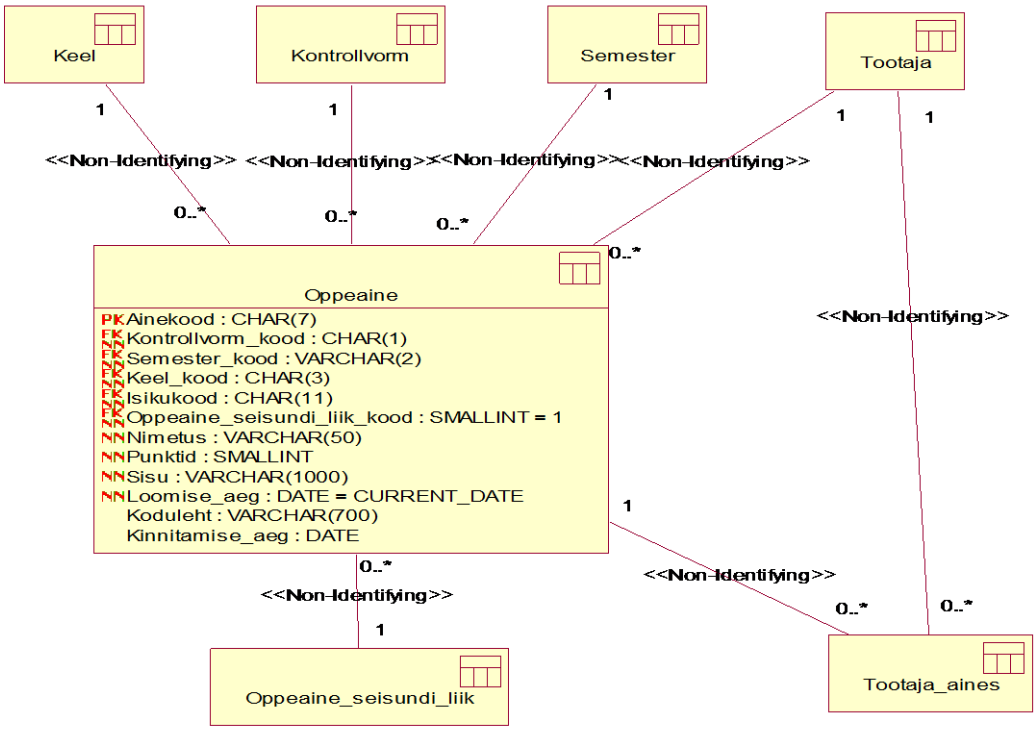
Järgnevatel joonistel(vt joonis 3-8) on esitatud 5NK andmebaasi disain, mis on koostatud kasutades teisendusreegleid. *Lisa 3* sisaldab endast tõestust, et andmebaas on normaliseeritud viienda normaalkujuni.



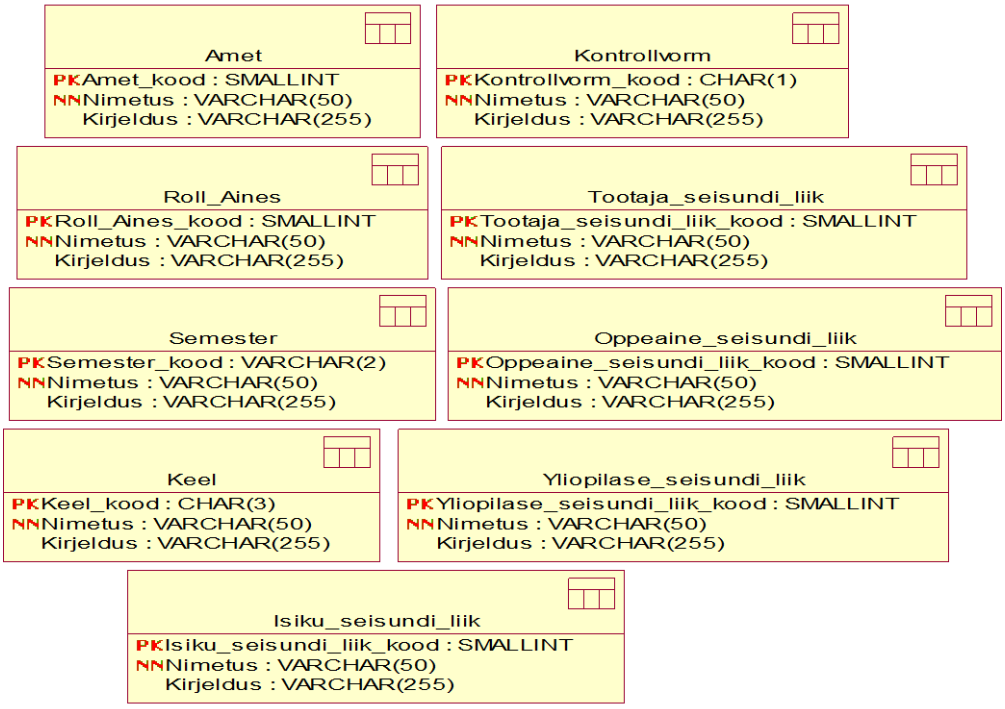
Joonis 3. 5NK andmebaasi isikute registri füüsilise disaini andmebaasi diagramm (UML notatsioonis).



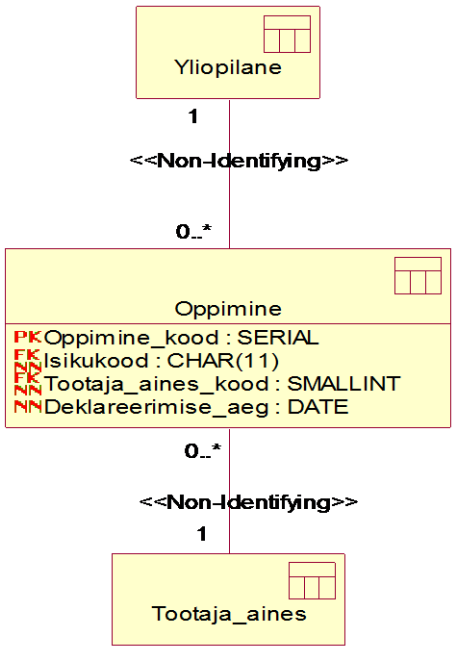
Joonis 4. 5NK andmebaasi üliõpilaste registri füüsilise disaini andmebaasi diagramm (UML notatsioonis).



Joonis 5. 5NK andmebaasi õppeainete registri füüsilise disaini andmebaasi diagramm (UML notatsioonis).



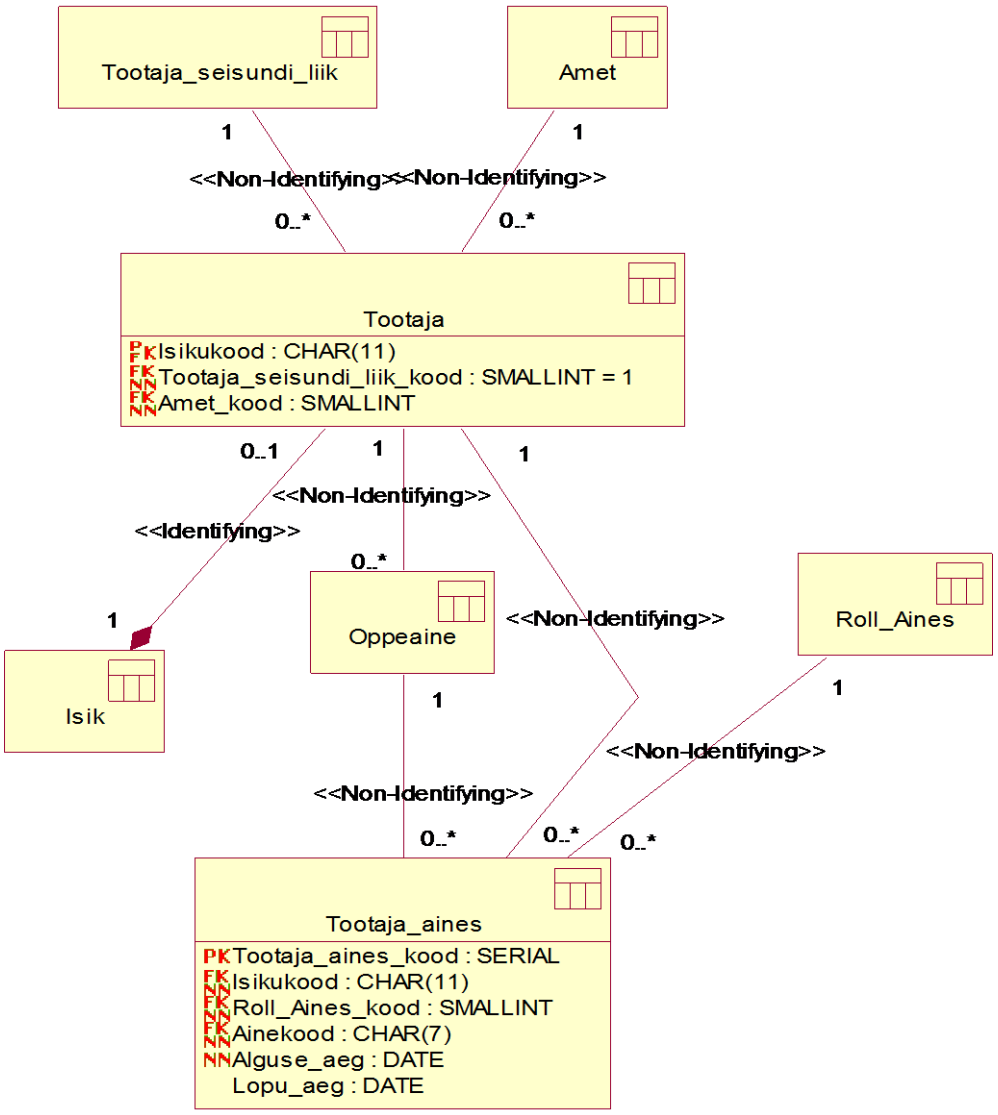
Joonis 6. 5NK andmebaasi klassifikaatorite registri füüsilise disaini andmebaasi diagramm (UML notatsioonis).



Joonis 7. 5NK andmebaasi õppimiste registri füüsilise disaini andmebaasi diagramm (UML notatsioonis).

Oppimine tabelis on määratud tabeli unikaalseks identifikaatoriks (mida jõustab UNIQUE kitsendus): *isikukood, tootaja_aines_kood, deklareerimise_aeg*. *Oppimine*

tabelis olev välisvõtme veerg *isikukood* on *Yliopilane* tabeli primaarvõtmeks. Antud unikaalne kombinatsioon tagab, et üliõpilane saab deklareerida ainet samal ajahetkel samale õppejõule ainult ühe korra.



Joonis 8. 5 NK andmebaasi töötajate registri füüsilise disaini andmebaasi diagramm (UML notatsioonis).

Unikaalseks identifikaatoriks *Tootaja_aines* tabelis (mida jõustab UNIQUE kitsendus) on *isikukood, roll_aines_kood, ainekood, alguse_aeg*. Antud kombinatsioon tagab, et ei saa lisada sama töötajat samasse ainesse sama töötamise alguse ajaga ja rolliga aines.

3.3 Denormaliseeritud andmebaasi disain

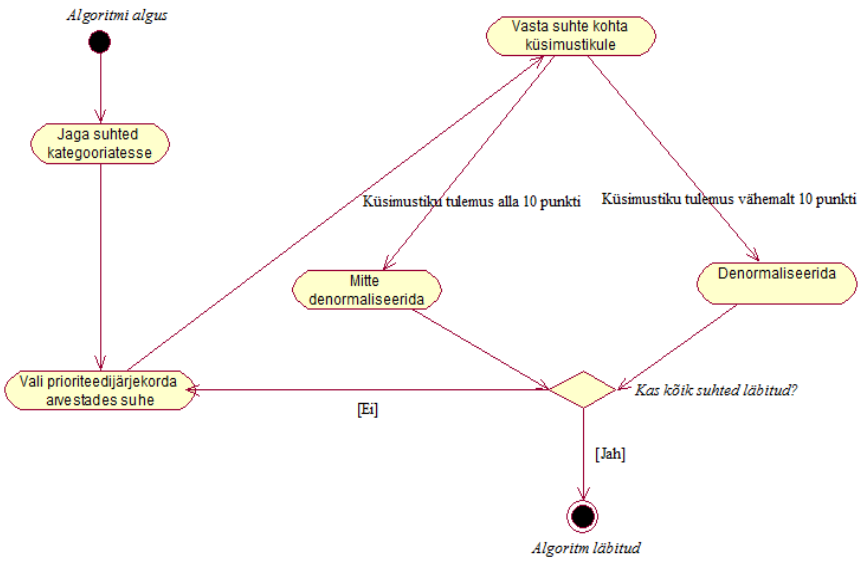
Tabelite denormaliseerimiseks kasutatakse algoritmi, mida kirjeldab Steve Hoberman oma raamatus [3] lehekülgedel 342-361. See algoritm aitab otsustada, millised tabelid tuleks denormaliseerida ja millised mitte. Iga andmemudelis esitatud tabelite vahelise suhte kohta vastatakse küsimustikule. Igal küsimustikus sisalduval küsimusel on vastusevariandid. Iga variant annab teatud arv punkte.

Denormaliseerimise algoritm

Denormaliseerimise algoritmi üldine idee on välja toodud joonisel 9. Nagu näha jooniselt tuleb esimese tegevusena jagada tabelite vahelised seosed e suhted kategooriatesse. Suhted jaotatakse kategooriatesse järgnevalt:

a) Üks-üks suhted
b) Klassifikaatorite/põhiandmete tabelite omavahelised suhted
c) Transaktsiooniliste andmetega omavahelised suhted
d) Klassifikaatorite/põhiandmete tabelite ning transaktsiooniliste andmetega tabelite vahelised suhted

Kategooria määrab ära suhete analüüsimise järjekorra. Alustatakse suhetest, mille puhul on denormaliseerimise tõenäosus kõige suurem. Alustatakse a kategooriasse kuuluvatest suhetest liikudes edasi kuni d kategooria suheteni. Kui ühte kategooriasse kuulub mitu suhet, siis nende vahel valiku järjekord on vaba. Iga uue suhte analüüsimise korral tuleb arvestada eelneva suhte analüüsimise ja denormaliseerimise tulemusena tekkinud tabeliga. [3]



Joonis 9. Denormaliseerimise algoritm.

Suhte kontekstis sisaldab sõltuv tabel välisvõtit. Sõltuv tabel on välisvõtme kaudu seotud primaarse tabeliga ehk antud töös peremeestabeliga. [9]

Järgnevalt on esitatud selle algoritmi kasutatav küsimustik koos vastustega tabelis 1 suhtele Isik-Töötaja. *Lisa 4* sisaldab endast denormaliseerimise algoritmi küsimustiku koos võimalike vastusevariantide ning neile vastavate punktidega. Denormaliseerimise algoritm antud töös kasutatava konseptuaalse mudeli põhjal on väga mahukas, seetõttu on see lisatud lisadesse. *Lisa 4* sisaldab endast veel denormaliseerimise algoritmi põhjal suhete kategooriate jaotamist ning kõikide suhete põhjal vastatud küsimustike.

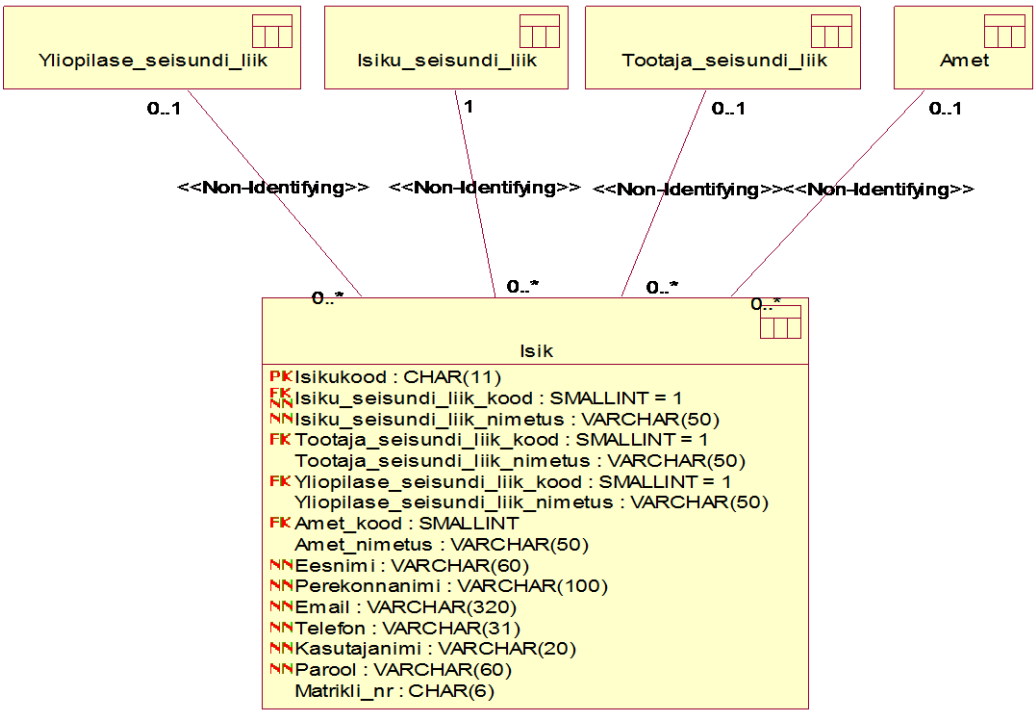
Tabel 1. Denormaliseerimise algoritmi küsimustiku näide suhte Isik-Töötaja alusel.

Küsimus	**Valik**
Mis tüüpi suhtega on tegemist peremees tabeli ja sõltuva tabeli vahel?	Osa-terviku (agregatsiooni või kompositsiooni) seos, hierarhia. Nt. tellimus sisaldab tellimuste ridu, kauba kategooria sisaldab alamkategooriaid. **(20)**
Milline on ühe peremeestabeli reaga seotud sõltuva tabeli ridade arv?	1 kuni 5 **(20)**
Kui palju veerge on peremeestabelis, jättes arvestamata primaarvõtme veerud?	Vähem kui 10 **(20)**
Kui tehakse päring sõltuva tabeli kohta, siis kui	Päringus soovitakse sageli andmeid

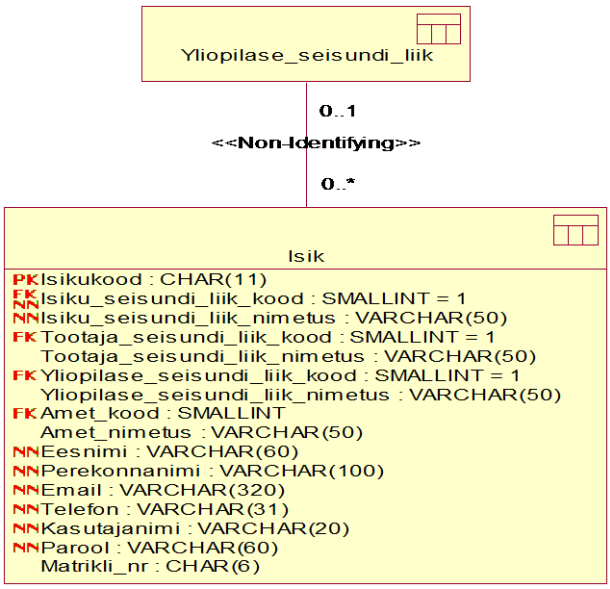
Küsimus	**Valik**
sageli soovitakse andmeid ka peremeestabelist?	peremeestabelist **(30)**
Kas peremeestabelisse on lähiajal plaanis lisada uusi veerge või siduda uusi tabeleid?	Ei **(20)**
Kas peremeestabelis ja sõltuvas tabelis toimub ajaperioodis sarnane arv lisamisi ja muutmisi?	Ei (ei toimu sarnane arv lisamisi ja muutmisi) **(-20)**
KOKKU PUNKTE	90 Otsus: **Kaaluda denormaliseerimist**

Denormaliseerimise algoritm soovitas kaaluda denormaliseerimist kõikide suhete korral peale *Oppimine–Yliopilane*. Kuna töös on denormaliseerimise aluseks võetud Steve Hobermanni kirjeldatud denormaliseerimise algoritm, siis denormaliseerisitakse kõik suhted peale *Oppimine–Yliopilane* suhte. Tabelite denormaliseerimisel on võetud aluseks Erki Eessaare konspekti teemal „Andmebaaside projekteerimine (füüsilisest disainist töötava andmebaasi hooldamiseni)“ [9]. Üks-üks suhete korral on tehtud tabel nimega *Isik*, mis koosneb 5NK andmebaasi disaini tabelitest *Isik*, *Yliopilane* ja *Tootaja*. Klassifikaatorite/põhiandmete tabelite omavaheliste seoste korral on lisatud peremeestabelisse juurde veerud klassifikaatoritele vastavate nimetuste jaoks. Kui eelnevalt 5NK andmebaasi disainis oli peremeestabelis välisvõtmeks vaid klassifikaatori tabelile vastav primaarvõtme kood, siis denormaliseerimise tulemusel on nüüd võimalik näha klassifikaatori nimetust ühendamisoperatsioone läbiviimata. Transaktsiooniliste andmetega tabelite vaheliste suhete korral dubleeriti sõltuvasse tabelisse enim kasutajate poolt otsitavaid või vaja minevaid veerge peremeestabelist. *Tootaja_aines-Oppimine* suhte korral on dubleeritud sõltuvasse tabelissse ehk *Oppimine* kõik veerud, mis kuuluvad *Tootaja_aines* tabeli unikaalsesse identifikaatorisse. Klassifikaatorite/põhiandmete tabelite ning transakatsiooniliste andmetega tabelite vaheliste seoste puhul on denormaliseeritud sama loogika alusel nagu transaktsiooniliste andmetega tabelite omavaheliste suhete korral ehk dubleeritud sõltuvasse tabelisse enim otsitavad veerge peremeestabelist. Veergude dubleerimise tagajärjel on dubleeritud ka veergudele vastavad kontrollkitsendused.

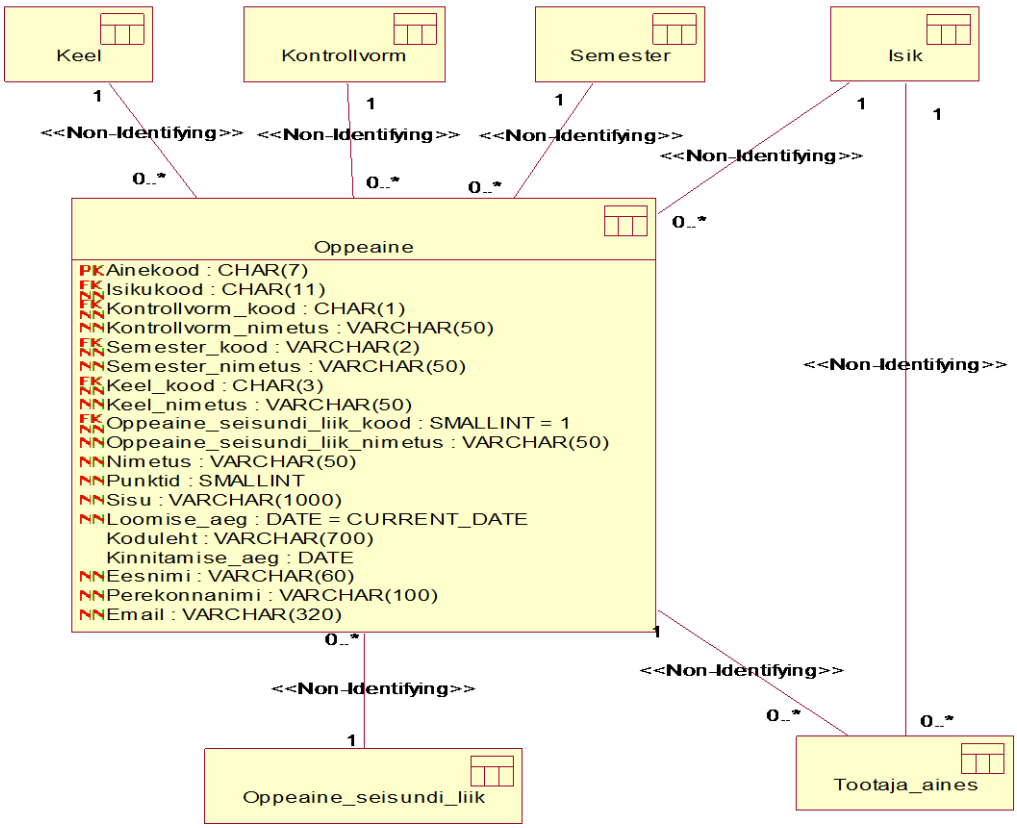
Kõikide välisvõtmete puhul on kasutatud määrangut ON UPDATE CASCADE ON DELETE NO ACTION. Järgnevatel joonistel(vt joonis 10-15) on esitatud denormaliseeritud andmebaasi disain.



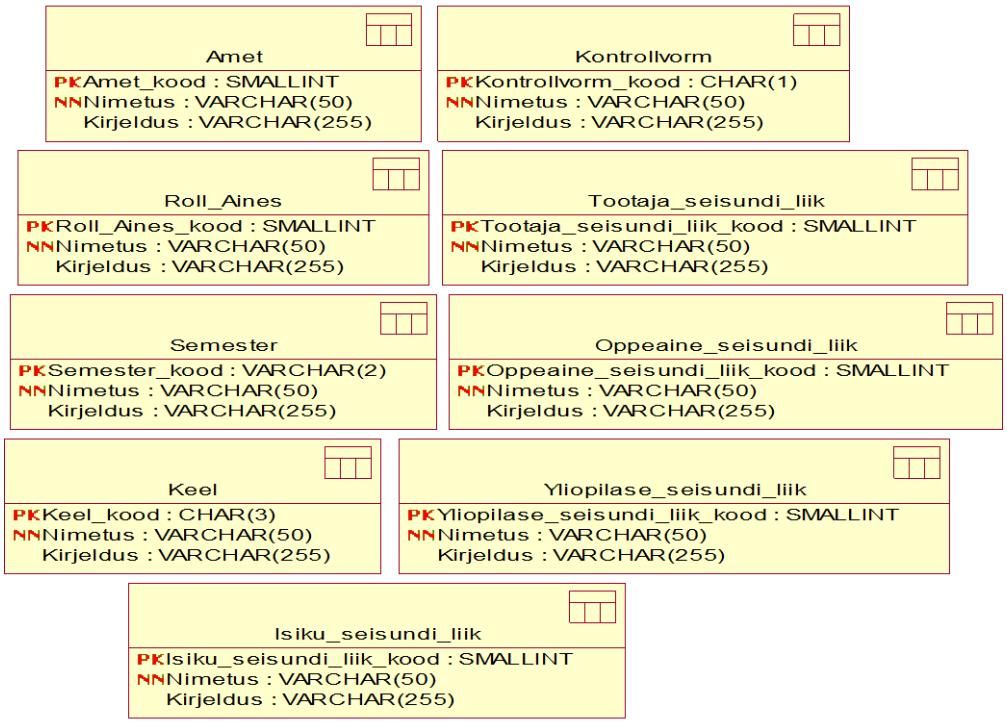
Joonis 10. Denormaliseeritud andmebaasi isikute registri füüsilise disaini andmebaasi diagramm (UML notatsioonis).



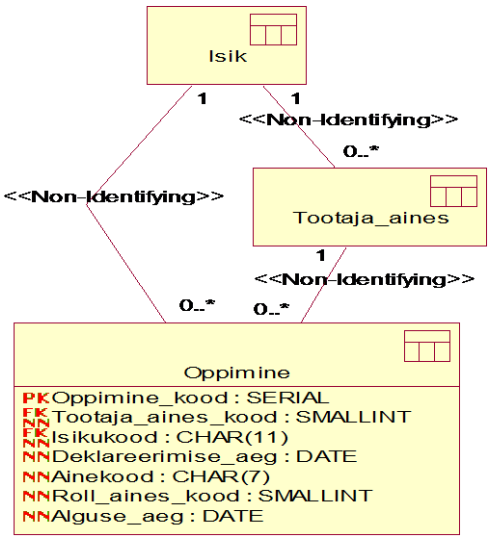
Joonis 11. Denormaliseeritud andmebaasi üliõpilaste registri füüsilise disaini andmebaasi diagramm (UML notatsioonis).



Joonis 12. Denormaliseeritud andmebaasi õppeainete registri füüsilise andmebaasi diagramm (UML notatsioonis).

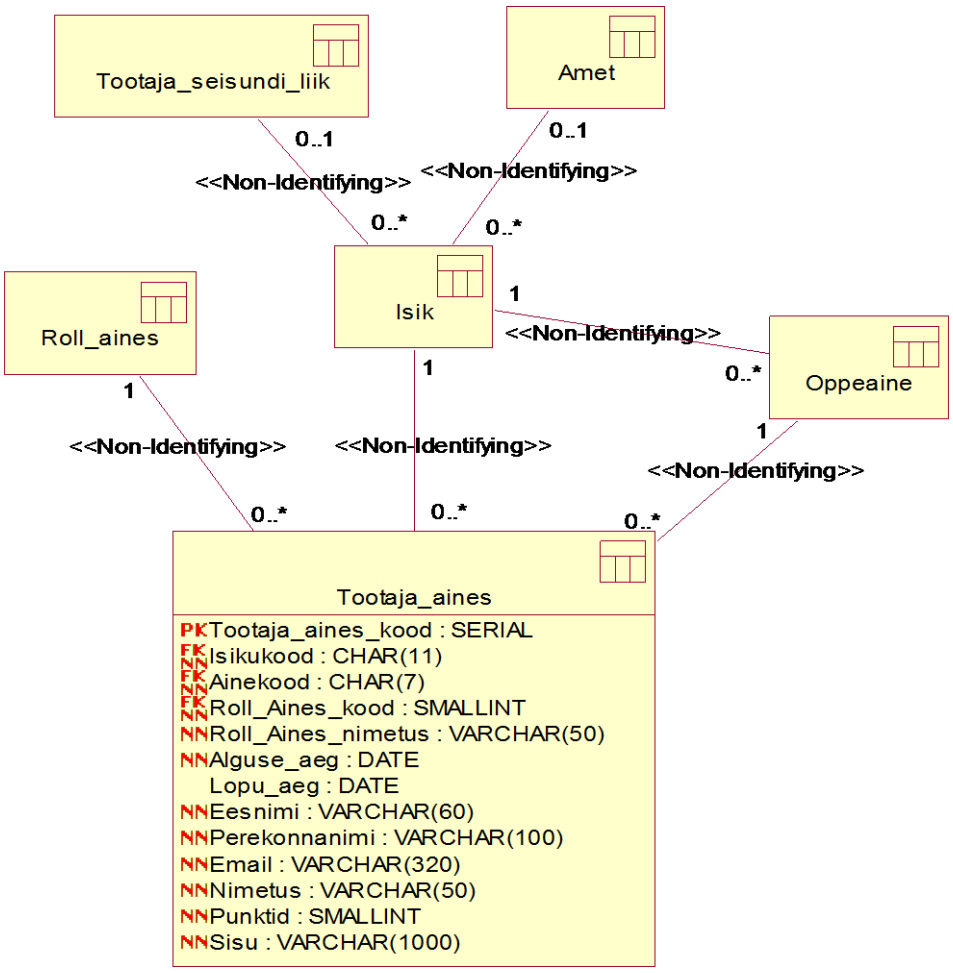


Joonis 13. Denormaliseeritud andmebaasi klassifikaatorite registri füüsilise disani andmebaasi diagramm (UML notatsioonis).



Joonis 14. Denormaliseeritud andmebaasi õppimiste registri füüsilise disaini andmebaasi diagramm (UML notatsioonis).

Denormaliseeritud andmebaasi *Oppimine* tabelis on määratud tabeli unikaalseks identifikaatoriks (mida jõustab UNIQUE kitsendus): isikukood, tootaja_ainesi_kood, deklareerimise_aeg ja ainekood. *Oppimine* tabelis olev välisvõtme veerg *isikukood* on *Yliopilane* tabeli primaarvõtmeks. Antud unikaalne kombinatsioon tagab, et üliõpilane saab samal ajahetkel deklareerida õppeainet õppejõule ainult ühe korra.



Joonis 15. Denormaliseeritud andmebaasi töötajate registri füüsilise disaini andmebaasi diagramm (UML notatsioonis).

Unikaalseks identifikaatoriks *Tootaja_aines* tabelis (mida jõustab UNIQUE kitsendus) on *isikukood, roll_aines_kood, ainekood, alguse_aeg*. Antud kombinatsioon tagab, et ei saa lisada sama töötajat samasse ainesse sama töötamise alguse ajaga ja rolliga aines.

4 Eksperimendi kirjeldus

Eksperimendi käigus luuakse ühe konseptuaalse andmemudeli alusel kaks erineva disainiga andmebaasi, millest üks on täielikult normaliseeritud ning teine denormaliseeritud algoritmi järgi, mida kirjeldab Sterve Hobermann oma oma raamatus [3] lehekülgedel 342-361. Andmebaasisüsteemiks on valitud PostgreSQL 9.4. Valiku tegemisel lähtuti süsteemi kättesaadavusest ning sellest, et autor tunneb seda. Valikut toetab see, et tegemist on tasuta, avatud lähtekoodiga ja populaarse andmebaasisüsteemiga. 2016. aasta mais on see andmebaasissüsteemide populaarsuse indeksis viiendal kohal [10]. Testid tehti serveris *apex.ttu.ee*. Serveri tehnilised andmed: virtuaalmasin QEMU Virtual CPU version, 811 GB HDD, 40 GB RAM, 15 virtuaalset CPU'd, CentOS 6.4.

Füüsiliselt on loodud üks andmebaas, mille nimeks on *denormaliseerimise_uurimine* ja kaks erinevat skeemi *normalized* ja *denormalized* [11]. Esimene skeem (*normalized*) on viiendal normaalkujul olevate tabelite jaoks ja teine skeem on algoritmi järgi denormaliseeritud (*denormalized*).

Mõlema skeemi loomise kood on küllaltki mahukas, seega on nad paigutatud lisadesse:

- *Lisa 5* sisaldab SQL koodi 5NK andmebaasi loomiseks
- *Lisa 6* sisaldab SQL koodi algoritmi järgi denormaliseeritud andmebaasi loomiseks

Denormaliseerimise tulemusena on denormaliseeritud andmebaasis paljud veerud dubleeritud teistese tabelitesse. Seetõttu on dubleeritud ka kontrollkitsendused. Kitsenduste dubleerimine raskendaks koodi hallatavust. Selle vältimiseks kasutatakse andmebaasis domeene, mis võimaldavad veeru omaduste (sh kitsenduste) kirjelduse üks kord kirja panna ja siis seda taaskasutada. [8]

Nende kahe erineva disaini võrdlemiseks tehakse järgnevad testid:

- Mõõdan tabelite ja neile loodud indeksite andmemahte
- Mõõdan konkreetse olemi otsimise päringute täitmise kiirust ning keerukust
- Mõõdan koondandmete päringu täitmise kiirust ning keerukust
- Mõõdan konkreetse olemi andmete muudatuse kiirust ning keerukust
- Mõõdan kitsenduse jõustamise keerukust

Tabelitele ja neile loodud indeksite andmemahu teada saamiseks kasutan PostgreSQL andmebaasisüsteemis olemasolevat süsteemi-defineeritud funktsiooni *pg_total_relation_size(regclass)*, millega saab teada tabeli andmemahu baitides. Antud funktsioon tagastab tabeli, tabelitega seotud indeksite ning TOAST andmete andmemahu baitides. [12] Töös kasutatakse mõlema skeemi iga tabeli jaoks päringut, mis on esitatud joonisel 16.

```
SELECT  pg_total_relation_size('schema.table_name');
```

Joonis 16. Andmebaasi andmemahu mõõtmise päring.

Antud päringus tuleb asendada *schema* vastava skeemi nimega, milles asuva tabeli andmemahtu soovite leida ning *table_name* asendada vastava tabeli nimega.

Eksperimendi käigus koostatakse kokku viis erinevat päringut. Osad neist on mõeldud koondandmete otsimiseks teised aga konkreetse olemi otsimiseks andmebaasist. Töös analüüsitakse nende päringute töökiirust ja keerukust. Tabelis 2 on välja toodud päringud 5NK andmebaasi jaoks koos kirjeldustega ning Tabelis 3 on samad päringud denormaliseeritud andmebaasi jaoks.

Tabel 2. 5NK andmebaasi eksperimendi päringud.

Päringu kirjeldus	**Päring**
Leia kõikide üliõpilaste arv	`SELECT count(*) FROM yliopilane;`

Leia isikute arv, kes on samal ajal nii aktiivsed töötajad kui ka üliõpilased	`SELECT count(*) FROM yliopilane`
	`INNER JOIN Tootaja ON yliopilane.isikukood = tootaja.isikukood`
	`WHERE yliopilane.yliopilase_seisundi_liik_kood = 1;`

Leia 2015.aastal alanud	`SELECT`

Päringu kirjeldus	Päring
õppimised ning väljasta ka aine nimetus, aine hetkeseisundi nimetus, töötaja perenimi, üliõpilase perenimi, üliõpilase hetkeseisundi nimetus ning järjesta need aine nimetuse järgi	oppeaine.nimetus AS Oppeaine_nimetus, oppeaine_seisundi_liik.nimetus AS Oppeaine_hetkeseisundi_nimetus, isik_2.perekonnanimi AS Tootaja_perenimi, isik.perekonnanimi AS Opilase_perenimi, yliopilase_seisundi_liik.nimetus AS Yliopilase_hetkeseisundi_nimetus FROM oppeaine INNER JOIN oppeaine_seisundi_liik ON oppeaine.oppeaine_seisundi_liik_kood = oppeaine_seisundi_liik.oppeaine_seisundi_liik_kood INNER JOIN tootaja_aines ON tootaja_aines.ainekood = oppeaine.ainekood INNER JOIN tootaja ON tootaja_aines.isikukood = tootaja.isikukood INNER JOIN isik as isik_2 ON tootaja.isikukood = isik_2.isikukood INNER JOIN oppimine ON oppimine.tootaja_aines_kood = tootaja_aines.tootaja_aines_kood INNER JOIN yliopilane ON oppimine.isikukood = yliopilane.isikukood INNER JOIN yliopilase_seisundi_liik ON yliopilane.yliopilase_seisundi_liik_kood = yliopilase_seisundi_liik.yliopilase_seisundi_liik_kood INNER JOIN isik ON yliopilane.isikukood = isik.isikukood WHERE oppimine.deklareerimise_aeg >= '2015-01-01' AND oppimine.deklareerimise_aeg <= '2015-12-31' ORDER BY oppeaine.nimetus;
Leia kõik read tabelist töötajad aines, kus alguse aeg on 2015. aastal	SELECT * FROM tootaja_aines WHERE alguse_aeg >= '2015-01-01' AND alguse_aeg <= '2015-12-31';
Leia kõik read tabelist töötajad aines, kus alguse aeg on 2015.aastal ja ainekood on IDD8132	SELECT * FROM tootaja_aines WHERE alguse_aeg >= '2015-01-01' AND alguse_aeg <= '2015-12-31' AND ainekood = 'IDD8132';

Tabel 3. Denormaliseeritud andmebaasi eksperimendi päringud.

Päringu kirjeldus	Päring
Leia kõikide üliõpilaste arv	SELECT count(*) FROM isik WHERE yliopilase_seisundi_liik_kood IS NOT NULL and matrikli_nr IS NOT NULL;
Leia isikute arv, kes on samal ajal nii töötajad	SELECT count(*) FROM Isik WHERE tootaja_seisundi_liik_kood IS NOT NULL AND

Päringu kirjeldus	Päring
kui ka üliõpilased	`yliopilase_seisundi_liik_kood = 1;`

Leia 2015.aastal alanud õppimised ning väljasta ka aine nimetus, aine hetkeseisundi nimetus, töötaja perenimi, üliõpilase perenimi, üliõpilase hetkeseisundi nimetus ning sorteeri need aine nimetuse järgi	`SELECT`
	`oppeaine.nimetus AS Oppeaine_nimetus,`
	`oppeaine.oppeaine_seisundi_liik_nimetus AS Oppeaine_hetkeseisundi_nimetus,`
	`oppeaine.perekonnanimi AS Tootaja_perenimi,`
	`isik.perekonnanimi AS Opilase_perenimi,`
	`isik.yliopilase_seisundi_liik_nimetus AS Yliopilase_hetkeseisundi_nimetus`
	`FROM oppeaine`
	`INNER JOIN oppimine ON oppimine.ainekood = oppeaine.ainekood`
	`INNER JOIN isik ON oppimine.isikukood = isik.isikukood`
	`WHERE oppimine.deklareerimise_aeg >= '2015-01-01' AND oppimine.deklareerimise_aeg <= '2015-12-31'`
	`ORDER BY oppeaine.nimetus;`

Leia kõik read tabelist töötajad aines, kus alguse aeg on 2015. aastal	`SELECT * FROM tootaja_aines` `WHERE alguse_aeg >= '2015-01-01' AND alguse_aeg <= '2015-12-31';`
Leia kõik read tabelist töötajad aines, kus alguse aeg on 2015. aastal ja ainekood on IDD8132	`SELECT * FROM tootaja_aines` `WHERE alguse_aeg >= '2015-01-01' AND alguse_aeg <= '2015-12-31' AND ainekood ='IDD8132';`

Konkreetse olemi andmete muudatuse kiirust ning keerukust testitakse antud lõputöös tehes muudatuse aine nimetuses. 5NK andmebaasi puhul peab tegema muudatuse ainult ühes tabelis, denormaliseeritud andmebaasis seevastu mitmes tabelis seoses veergude dubleerimisega. Joonisel 17 on esitatud andmemuudatuse lause 5NK andmebaasi jaoks. Kuna denormaliseeritud andmebaasis peab muutma aine nimetust mitmes tabelis, siis koostatakse selle jaoks funktsioon, mida saab andmemuudatuse korral väljakutsuda. [13] Joonisel 18 on esitatud andmemuudatuse lause denormaliseeritud andmebaasi jaoks.

```
UPDATE oppeaine SET nimetus = 'Andmebaasid' WHERE ainekood = 'IEC5535';
```

Joonis 17. Andmemuudatuse lause 5NK andmebaasi jaoks.

```
CREATE OR REPLACE FUNCTION
denormalized.f_test_andmemuudatus(denormalized.oppeaine.nimetus%TYPE,
denormalized.oppeaine.ainekood%TYPE)
RETURNS VOID
AS $$
UPDATE denormalized.oppeaine SET nimetus = $1 WHERE ainekood = $2;
UPDATE denormalized.tootaja_aines SET nimetus =  $1 WHERE ainekood = $2;
$$
LANGUAGE SQL SECURITY DEFINER;

SELECT denormalized.f_test_andmemuudatus('Andmebaasid', 'IEC5535');
```

Joonis 18. Andmemuudatuse lause denormaliseeritud andmebaasi jaoks.

Päringute keerukust mõõdetakse LocMetrics nimelise programmiga, mis on tasuta veebist kättesaadav aadressil http://www.locmetrics.com/ [14]. Antud programm väljastab füüsiliselt käivitatavate koodiridade arvu (SLOC-P), mille tulemusel saab analüüsida päringute keerukust [15]. Eelnevalt kasutatakse SQL päringute formaatimiseks internetipõhist programmi Instant SQL Formater, mis on tasuta kättesaadav veebis aadressil http://www.dpriver.com/pp/sqlformat.htm [16].

Lausete täitmise kiiruse mõõtmiseks kasutatakse PostgreSQL lauset EXPLAIN ANALYZE. Antud käsk tagastab päringu täitmisplaani, selle planeerimiseks kulunud aja ning päringu täitmiseks kulunud aeg. Päring küll kävitatakse, aga ilma vastuseid tagastamata. Igat päringut käivitatakse viis korda ning nende põhjal arvutatakse keskmine tulemus. [17] Päringu täitmise aja leidmiseks liideti kokku planeerimise ja plaani elluviimise aeg.

Kitsenduste jõustamise eesmärgiks on tagada, et andmebaasis olevad andmed vastavad mingitele reeglitele. Kitsendus võib näiteks nõuda, et mõni tekstivälja väärtus ei koosneks ainult tühikutest või õppeaine loomise aeg on reaalne (st ei ole enne ülikooli algust või kaugel tulevikus). Kitsendusi saab jõustada nii protseduurselt, koostades trigereid, kui ka deklaratiivselt luues tabelitega seotud kitsendusi nagu näiteks PRIMARY KEY, CHECK, UNIQUE jne. [8] Nii deklaratiivsete kitsenduste kui ka trigerite puhul on veergude sisu kontrolliks võimalik kasutada regulaaravaldisi. Kitsenduse jõustamise keerukuse analüüsimiseks luuakse üks uus kitsendus tabeli *Oppimine* veerule *deklareerimise_aeg*. Kitsenduse sisuks on kontrollida, et *Tootaja_aines* tabelis olevas veerus *alguse_aeg* ei oleks hilisem kui *deklareerimise_aeg.* Teiste sõnadega, töötajat ei tohi ainesse õpetajaks määrata hiljem kui üliõpilane on aine deklareerinud. [18]

Antud töö raames on võetud testandmete genereerimisel aluseks Tallinna Tehnikaülikooli töötajate, õppeainete ja üliõpilaste kohta internetist kättesaadavad andmed ning tehtud nende põhjal lihtsustusi. TTÜ kodulehel on on välja toodud 2011. aastal olev TTÜ ametikohtade arv, millest 842 on mitte-akadeemilised ametikohad ja 1251 akadeemilised ametikohad [19]. Antud töös on võetud keskmiseks töötajate arvuks 2000, kellest 1200 töötavad akadeemilistel töökohtadel ning 800 mitte-akadeemilistel töökohtadel. TTÜ õppeinfosüsteemist õppeainete otsingul väljastatakse 5250 õppeainet, oma lõputöös võtsin arvesse väljastatud õppeainete arvu ning genereerisin kokku 10 000 õppeainet, millest 6000 on seisundis „Aktiivne“ [20]. Testandmete genereerimisel *Yliopilase* tabelisse võeti aluseks TTÜ veebilehel olev info 2011-nda aasta üliõpilaste arvu kohta, milleks oli 14 378 üliõpilast. Andtud lõputöös on lihtsustatud eelnevalt leitud arvu ning seega lisatud Yliopilase tabelisse andmeid 10 000 üliõpilase kohta, kellest vähemalt 7500 on aktiivsed üliõpilased [19]. Kuna on võimalus, et töötaja võib olla ka samal ajal üliõpilane, siis on lisatud *Isik* tabelisse kokku 11 750 isikut. *Töötaja_aines* tabelisse andmete genereerimisel arvestati, et keskmiselt on iga õppeaine kohta 2 töötajat, seega on genereeritud antud tabeli jaoks 20 000 rida. Unikaalsuse kitsenduse tõttu tuleb kustutada osad read. Seega lõpptulemusena on lisatud *Tootaja_aines* tabelisse 19 996 rida. *Oppimine* tabelisse on genereeritud 120 000 õppimise rida võttes aluseks terve aasta jooksul keskmiselt deklareeritud ainete hulga, mis antud lõputöö koostaja ülikooli aastate jooksul on olnud keskmiselt 6 semestri kohta ning korrutatud seda üliõpilaste arvuga. Unikaalsuse kitsenduse tõttu pidi samuti eemaldama korduvad read. Seega lisati *Oppimine* tabelisse 119 787 rida. Testandmed on genereeritud nii, et mõnel õppejõul on rohkem õpetatavaid õppeaineid kui teisel õppejõul ning mõnel üliõpilasel on deklareeritud rohkem õppeaineid, teisel üliõpilasel vähem. Klassfikaatorite väärtustamise laused on välja toodud *Lisas 7*.

Testandmete genereerimiseks on kasutatud programmi ApexSQL Generate, mis võimaldab importida SQL koodi ning eksportida csv formaadis genereeritud testandmeid iga tabeli jaoks eraldi [21]. 5NK andmebaasi kopeeritakse testandmed csv failist, lause on esitatud joonisel 19 [22].

```
COPY tabelinimi FROM '/tmp/normalized/tabelinimi.csv' DELIMITER ',' CSV;
```

Joonis 19. Andmebaasi testandmete lisamise lause.

5NK andmebaasist kantakse andmed edasi denormaliseeritud andmebaasi kasutades selleks INSERT INTO lauseid [23]. Antud lähenemine tagab, et mõlemas andmebaasis on ühesugused testandmed. Pärast andmete ülekandmist teise tabelisse käivitatakse mõlema andmebaasi jaoks statistika värskendamiseks käsk ANALYZE [24].

5 Eksperimendi tulemused ja analüüs

Käesolevas peatükis on välja toodud eksperimendi tulmused ning nende analüüs. Lisaks on tehtud üldistus tulemuste põhjal.

5.1 Eksperimendi tulemuste analüüs

Esiteks tuuakse välja viienda normaalkujuni normaliseeritud ja denormaliseeritud andmebaasi andmemahud (vt tabel 4). Eksperimendi alguses eeldati, et denormaliseeritud andmebaasi andmemaht on suurem kui 5NK andmebaasi andmemaht, sest denormaliseeritud andmebaasis on palju dubleeritud veerge teistest tabelitest. Veergude dubleerimise tagajärjel tekib andmete liiasus. Antud eeldus vastab tõele, sest denormaliseeritud andmebaasi mahuks on ligikaudu 47 megabaiti võrreldes 5NK andmebaasi mahuga 38 megabaiti. Andmebaaside andmemahtude erinevus on suur, ligi 10 megabaiti. Antud andmemahtude võrdluse juures on huvitavaks hoopis see, et klassifikaatorite tabelid, mis on peaaegu identsed kuna veerge ei dubleeritud neisse tabelitesse ning mille testandmed vastavad üksühele on siiski suuruselt sõltuvalt andmebaasist erinevad. Ainuke erinevus klassifikaatorite tabelite puhul on domeenide kasutuses. 5NK andmebaasis on deklareeritud igale veerule kontrollkitsendus eraldi, denormaliseeritud andmebaasis kasutati domeene, et vähendada suure hulga kitsenduste haldamist. Kontrollimaks domeenide mõju on tehtud denormaliseeritud andmebaasi üks uus klassfikaatori tabeli ilma domeenita, mis on identne 5NK andmebaasi tabeliga. Järgnevalt mõõdeti uue klassifikaatori tabeli andmemahtu ning tulemus oli identne 5NK andmebaasi klassfikaatori tabeliga. Kindluse mõttes eemaldati ka testandmeid ning mõõdeti mõlema skeemi iga tabeli andmemahtu eraldi. *Lisa 8* sisaldab endast tabelit kus on välja toodud mõlema andmebaasi andmemaht enne testandmete sisestust. Tulemustest selgus, et denormaliseeritud andmebaasi andmemaht on ligikaudu 425 kilobaiti, mis on endiselt suurem 5NK andmebaasi andmahust, milleks on 360 kilobaiti. Samuti püsis ka klassfikaatorite tabelite andmemahtude erinevus. Seega antud võrdluse põhjal saab öelda, et domeenide kasutamine mõjutab kas andmemahtude mõõtmise programmi tööd või siis tegelikke andmemahte (mis on täpne põhjus vajaks eraldi

uurimist), kuid denormaliseeritud andmebaasi põhiandmete tabelid on siiski andmemahult suuremad kui 5NK andmebaasi vastavad tabelid.

Tabel 4. Andmebaaside andmemahud.

Tabeli nimi	**5NK andmebaas**	**Denormaliseeritud andmebaas**
Amet	40960	49152
Isik	4759552	9994240
Isiku_seisundi_liik	40960	49152
Keel	40960	49152
Kontrollvorm	40960	49152
Oppeaine	3080192	3760128
Oppeaine_seisundi_liik	40960	49152
Oppimine	23715840	2701726
Roll_aines	40960	49152
Semester	40960	49152
Tootaja	360448	-
Tootaja_aines	4587520	5898240
Tootaja_seisundi_liik	40960	49152
Yliopilane	1556480	-
Yliopilase_seisundi_liik	40960	49152
KOKKU	38428672 (36.6 MB)	47112192 (44.9 MB)

Järgnevalt, päringute kiiruse analüüsiks mõõdetakse koondandmete otsimise päringu ja erinevate konkreetse olemi otsimise päringute töökiirust. Tulemused on esitatud tabelis 5. Esimese päringu, milleks oli leida kõikide üliõpilaste arv, täitmiseks läks kauem aega denormaliseeritud andmebaasi põhjal (15,322ms), sest denormaliseerimise tulemusena kustutati tabelid *Yliopilane* ja *Tootaja* ning liideti kõik isikute andmed kokku tabelisse *Isik*. *Isik* tabelis on denormaliseerimise tulemusena 16 veergu, et leida üliõpilaste arv tuleb läbi vaadata rohkem veerge, mis annavad infot selle kohta kas isik on üliõpilane

või mitte. 5NK andmebaasi puhul seevastu tuleb pärida kõik andmed ainult tabelist *Yliopilane* kus on kolm veergu. 5NK andmebaasis läks päringu täitmiseks aega 10,354ms, mis on 5ms vähem kui denormaliseeritud andmebaasil. Antud päringu võrdluse tulemusena saab öelda, et konkreetse olemi otsing 5NK andmebaasi puhul on kiirem ja optimaalsem.

Teise päringu, milleks on leida isikute arv, kes on samal ajal nii töötajad kui üliõpilased, täitmiseks läks denormaliseeritud andmebaasis aega 3,210ms, 5NK andmebaasil 15,660ms. Päringute täitmiskiiruse vahe on ligikaudu 12ms, mis on suur erinevus. 5NK andmebaasist otsimisel tuleb läbi viia ühendamisoperatsioon, mis teebki antud päringu täitmisekiiruse aeglasemaks kui denormaliseeritud andmebaasis, kus tuleb otsida ainult ühest tabelist. Kiiremaks eelnevast päringust teeb denormaliseeritud andmebaasis vastuste otsimist see, et algselt välistatakse juba need read, millel üliõpilase seisundi liik on võrdne ühega ehk aktiivne. Antud päringu tulemusena võib öelda, et denormaliseeritud andmebaasi eeliseks on ühendamisoperatsioonide puudumine, mis teeb päringu täitmiskiiruse ligikaudu viis korda kiiremaks.

Kolmandaks päringuks on koondandmete päring, mille tulemusena leitakse kõik 2015.aastal alanud õppimised ning väljastatakse lisaks andmeid teistest tabelitest. Denormaliseeritud andmebaasis päringu täitmine võttis aega 61,982ms ja 5NK andmebaasis 101,453ms. Nagu eelneva päringu puhul tuleneb väga suur täitmiskiiruste erinevus ühendamisoperatsioonidest. Denormaliseeritud andmebaasis tuleb päringule vastuse saamiseks ühendada kolm tabelit. Seevastu 5NK andmebaasis kaheksa tabelit, mis teebki 5NK andmebaasi päringu täitmise aeglasemaks kui denormaliseeritud andmebaasis. Antud päringu tulemusena võib öelda, et koondandmete päringu täitmiskiirus on denormaliseeritud andmebaasi puhul kiirem, sest puuduvad kulukad ühedamisoperatsioonid.

Neljandaks ja viiendaks päringuks on leida kõik read tabelist tootaja_aines kus töötaja alguse aeg õppeaines on 2015.aastal, erinevuseks on lisatingimus viienda päringu puhul kus on vaja kontrollida ka ainekoodi. Neljanda päringu täitmiskiiruse vahe on ligikaudu 3,50 ms, antud päringu puhul tuleb erinevus sellest, et denormaliseeritud andmebaasis on veerge mida läbi vaadata ja tagastada rohkem kui 5NK andmebaasis. Viienda päringu täitmiskiirus on 5NK andmebaasis 0,496ms ja denormaliseeritud andmebaasis

0,595ms. Erinevuseks on jällegist veergude arv ning tagastavate andmete hulk, päringu töökiirus on mõlema andmebaasi puhul väga kiire. Täitmisplaan, mille leidmiseks kasutan PostgreSQL funktsiooni EXPLAIN ANALYZE, ütleb, et otsimiseks kasutatakse *tootaja_aines* tabeli unikaalset identifikaatorit ning nendel välisvõtme veergudel põhinevaid indekseid ehk otsimisel viiakse läbi *index uniqe scan* operatsioon. Antud päringu tulemusena selgub jälle nagu esimese päringu puhul, et konkreetse olemi otsimise päringu täitmisekiirus on suurem 5NK andmebaasis.

Tabel 5. Eksperimendi päringute täitmiskiirus.

Päring	**5NK andmebaas**	**Denormaliseeritud andmebaas**
Leia kõikide üliõpilaste arv	10,354 ms	15,322 ms
Leia isikute arv, kes on samal ajal nii aktiivsed töötajad kui ka üliõpilased	15,660 ms	3,210 ms
Leia 2015.aastal alanud õppimised ning väljasta ka aine nimi, aine hetkeseisundi nimi, töötaja perenimi, üliõpilase perenimi, üliõpilase hetkeseisundi nimi ning järjesta need aine nime järgi	101,453 ms	61,982 ms
Leia kõik read tabelist töötajad_aines, kus alguse aeg on 2015. aastal.	5,428 ms	8,73 ms
Leia kõik read tabelist töötajad_aines, kus alguse aeg on 2015.aastal ja ainekood on IDD8132.	0,496 ms	0,595 ms

Konkreetse olemi andmete muudatuse kiiruse ehk õppeaine nimetuse muutmise tulemused on välja toodud tabelis 6. 5NK andmebaasis kulus andmemuudatuste läbiviimiseks 0,531ms, denormaliseeritud andmebaasis kulus 1,722ms. Tulemustest selgub, et 5NK andmebaasis andmete muudatuse täitmiskiirus on kiirem kui denormaliseeritud andmebaasis, sest denormaliseeritud andmebaasis peab denormaliseerimise tulemusena muutma nimetust mitmes tabelis. Antud tulemuste põhjal saab öelda, et andmemuudatused on denormaliseeritud andmebaasis kulukad ja aeganõudvad. Negatiivse küljena võib veel välja tuua denormaliseeritud andmebaasis andmemuudatus operatsioonide läbi viimisel tekkiva ohu andmete kvaliteedis ja usaldusväärsuses. Seda juhul kui kasutaja andmeid muutes ei muuda fakti selle kõigis esinemise kohtades.

Tabel 6. Andmemuudatuse täitmiskiirus millisekundites.

5NK andmebaas	**Denormaliseeritud andmebaas**
0,531ms	1,722ms

Konkreetse objekti otsimise ja koondandmete päringute füüsiliste ridade arvud on toodud välja tabelis 7. Tabelist on näha, et suuri erinevusi konkreetse olemi otsimise päringute koodiridade arvus ei esinenud. Seevastu koondandmete päringu koodiridade arvu erinevus on suur. Esimese päringu erinevus tuleneb denormaliseeritud andmebaasi päringus olevatest lisatingimustest, mille järgi leitakse *Isik* tabelist kõik üliõpilased. Teise päringu erinevus tuleneb sellest, et 5NK andmebaasis viiakse läbi ühendamisoperatsioon, mis teeb päringu keerukamaks ja pikemaks. Neljas ja viies on päring on mõlema andmebaasi puhul identsed. Suurim erinevus tuleneb koondandmete päringust kus denormaliseeritud andmebaasi koodiridade arv on 14, 5NK andmebaasis seevastu 27 rida. Nagu eelnevalt juba öeldud, siis 5NK andmebaasis tuleb päringule tulemuste leidmiseks ühendada kaheksa tabelit, denormaliseeritud andmebaasis samale päringule tulemuste leidmiseks ainult kolm tabelit. Antud tulemuste põhjal võib öelda, et koondandmete päringud, mille korral on vaja läbi viia palju ühendamisoperatsioone, teevad päringud keerulisemaks. Denormaliseeritud andmebaasis on kasutajatel koondandmete päringute kirjutamine lihtsam. Konkreetse objekti otsimise päringuid võrreldes märgatavat vahet päringute keerukuses ei esinenud. Samas lõputöö autor usub, et konkreetse olemi otsimise päringu puhul on 5NK andmebaasi disain kasutajatele arusaadavam ning seetõttu on kasutajatel on lihtsam andmeid otsida ning päringuid koostada 5NK andmebaasi põhjal.

Tabel 7. Eksperimendi päringute koodiridade arv.

Päring	**5NK andmebaas**	**Denormaliseeritud andmebaas**
Leia kõikide üliõpilaste arv	2	4
Leia isikute arv, kes on samal ajal nii aktiivsed töötajad kui ka üliõpilased	5	4
Leia 2015.aastal alanud õppimised ning väljasta ka aine nimi, aine hetkeseisundi nimi, töötaja perenimi,	27	14

Päring	5NK andmebaas	Denormaliseeritud andmebaas
üliõpilase perenimi, üliõpilase hetkeseisundi nimi ning järjesta need aine nime järgi		
Leia kõik read tabelist töötajad aines, kus alguse aeg on 2015. aastal.	4	4
Leia kõik read tabelist töötajad aines, kus alguse aeg on 2015.aastal ja ainekood on FAI5528.	5	5

Selleks, et võrrelda kitsenduste jõustamise keerukust tehakse läbi kitsenduse jõustamise *Oppimine* tabeli veerule *deklareerimise_aeg* mõlemas andmebaasis. Kitsenduse sisuks on kontrollida, et *Tootaja_aines* tabelis olevas veerus *alguse_aeg* olev kuupäev pole hilisem kui veerus *deklareerimise_aeg* ole kuupäev, st et töötaja on ainesse õpetatavaks määratud enne või vähemalt samal ajal kui üliõpilane saab ainet deklareerida.

5NK andmebaasis asuvad veerud *alguse_aeg* ja *deklareerimise_aeg* erinevates tabelites. Seega tuleb kitsenduse jõustamiseks koostada kitsenduse triger, mis kutsub välja funktsiooni [25]. 5NK andmebaasi kitsenduse loomise näide on toodud välja joonisel 21.

Denormaliseeritud andmebaasis on antud kitsenduse jõustamine keerukam, sest veergude dubleerimise tulemusena on *Tootaja_aines* tabelis kõigepealt olemas nii veerud *deklareerimise_aeg* kui ka *alguse_aeg*. Kitsenduse loomise lause denormaliseeritud andmebaasis on esitatud joonisel 20.

```
ALTER TABLE oppimine
  ADD CONSTRAINT chk_oppimine_deklareerimise_aeg_hilisem_kui_alguse_aeg
      CHECK  (deklareerimise_aeg>=alguse_aeg);
```

Joonis 20. Denormaliseeritud andmebaasi kitsenduse realisatsioon.

Samas ei tohi unustada, et need andmed on dubleeritud ka erinevates tabelites. Sellises olukorras kitsenduse täielikuks jõustamiseks tuleks teha ka mitut tabelit hõlmav kontroll, mida PostgreSQL abil on vaja realiseerida kitsenduste trigeri abil, mis sarnaneb 5NK andmebaasi trigerile.

```
CREATE OR REPLACE FUNCTION normalized.f_deklareerimise_aeg() RETURNS trigger
AS $$
DECLARE
 m_alguse_aeg DATE;
       BEGIN
       SELECT alguse_aeg INTO m_alguse_aeg FROM normalized.tootaja_aines
       WHERE tootaja_aines_kood = NEW.tootaja_aines_kood FOR UPDATE;
       IF (m_alguse_aeg > NEW.deklareerimise_aeg)  THEN
              RAISE EXCEPTION 'Ei saa lisada deklareerimist kui deklareerimise
              aeg ei ole hilisem töötaja aine alguse ajast';
       ELSE
              RETURN NEW;
END IF;
END;
$$ LANGUAGE plpgsql SECURITY DEFINER;

CREATE CONSTRAINT TRIGGER trig_deklareerimise_aeg AFTER INSERT OR UPDATE of
tootaja_aines_kood, deklareerimise_aeg ON normalized.Oppimine
DEFERRABLE INITIALLY DEFERRED
FOR EACH ROW EXECUTE PROCEDURE normalized.f_deklareerimise_aeg();

COMMENT ON FUNCTION normalized.f_deklareerimise_aeg() IS 'See trigeri
funktsioon aitab jõustada reeglit: Ei saa lisada deklareerimiset kui
deklareerimise aeg ei ole hilisem töötaja aine alguse ajast';
```

Joonis 21. 5NK andmebaasi kitsenduse realisatsioon.

Antud kitsenduste loomise põhjal saab öelda, et 5NK andmebaasis on väiksema andmete liiasuse tõttu kontrollkitsendusi lihtsam lisada ja kontrollida, sest ühte ja sama asja ei pea mitu korda erinevate tabelite põhjal kontrollima. Denormaliseeritud disaini puhul oleks võinud võtta lähenemise, et jõustada kõigest CHECK kitsendus tabeli *Oppimine* põhjal. Sellisel juhul peaks andmete muutmine tabelites *Oppimine* ja *Tootaja_aines* toimuma alati garanteeritult ühe transaktsioonina. Kui andmemuudatus ühes tabelis (*Oppimine*) ebaõnnestub, siis ebaõnnestub terve transaktsioon. Kuna muudatuste transaktsiooni koondamine on andmebaasi kasutajate vaba tahte avaldus ei ole samas garantiid, et seda alati tehakse ja siis võivad ilma täiendava kontrollita tekkida andmetesse vastuolud.

5.2 Eksperimendi üldistus

Käesolevas alapeatükkis antakse eksperimendi tulemuste põhjal ülevaade denormaliseeritud ja 5NK e täielikult normaliseeritud andmebaaside headest ja

halvadest külgedest ning lõputöö autor annab arvamuse kumba andmebaasi disaini kasutamine on analüüsi tulemuste põhjal optimaalsem.

Võrreldes omavahel denormaliseeritud ja 5NK andmebaasi saab eksperimendi tulemuste ja nende analüüsi põhjal välja tuua mõlema andmebaasi disainis nii positiivseid kui ka negatiivseid külgi. Denormaliseeritud andmebaasi positiivseteks külgedeks on koondandmete päringute täitmiskiirus ja keerukus. Negatiivsena võib välja tuua suured andmemahud, andmete muudatuse töökiiruse ning kitsenduste jõustamise keerukuse. Seevastu 5NK andmebaasi positiivseteks külgedeks on konkreetse olemi otsimise päringute täitmiskiirus, väiksemad andmemahud ning andmemuudatuste operatsioonide läbiviimine.

Disainilahenduse valik sõltub kontekstist ja selle määratud kriteeriumite olulisusest. Antud võrdluse põhjal võib öelda, et kuigi andmebaasi denormaliseerimisel leidub ka positiivseid külgi, on operatiivandmete andmebaasides, kus toimub palju väikeseid andmemuudatusi ja üksikute olemite andmete otsimisi, siiski andmebaasi kõrge tasemeni normaliseerimine otstarbekam tegevus kui denormaliseerimine. Autor eelistab seda eelkõige selle tõttu, et 5NK andmebaasid on kasutajatele mugavamad kasutamiseks ja arusaadavamad andmete otsimiseks ning andmemuudatuste operatsioonide läbiviimiseks, mis kindlasti tegelikus elus andmebaasidega töötavatel inimestel tihti ette tuleb. Andmeaida ja andmevaka tüüpi andmebaasides, kus toimuvad keerukad koondandmete päringud ja andmete massiline lisamine, oleks denormaliseerimisest rohkem kasu ja seda tasub seal kaaluda.

6 Kokkuvõte

Töö eesmärgiks oli praktiliste katsete najal saada teada, millist laiemat mõju omab tabelite denormaliseerimine andmebaasi kasutamisele SQL-andmebaasisüsteemides. Eesmärgi saavutamiseks võrreldi kahte SQL-andmebaasi disaini, millest üks oli normaliseeritud kuni viienda normaalkujuni ning teine denormaliseeritud algoritmi järgi, mille kirjeldas Steve Hobermann oma teoses „Data modeler's workbench: Tools and Techniques for Anlaysis and Design" lehekülgedel 342-361. Andmebaasid realiseeriti PostgreSQL 9.4 andmebaasisüsteemis ning täideti samasguseid fakte esitavate testandmetega. Testandmete loomiseks kasutati programmi ApexSQL Generate ja testide läbiviimiseks kasutati PostgreSQL 9.4 enda sisemisi abifunktsioone. Andmebaasi disaine võrreldi erinevate kriteeriumite alusel.

Töö oluliseks tulemuseks on kaks erinevat andmebaasi disaini, nende põhjal loodud andmebaas ja skeemid, skeemide põhjal tehtud päringute ja andmemuudatuste kontrollide näited ning nende disaini võrdlemise tulemusena tehtud järeldused. Tulemuste analüüsimisel selgus, et andmebaaside denormaliseerimine mõjub hästi koondandmete päringute täitmiskiirusele ja keerukusele. Halvasti seevastu andmebaasi andmemahtudele, konkreetse olemi otsimise päringute täitmiskiirusele, andmemuudatuse operatsioonidele ning keerulisemate kitsenduste jõustamisele. Tulemuste põhjal on tehtud üldistus, et kuigi andmebaasi denormaliseerimise praktika kasutamises leidub positiivseid külgi, on andmebaasi kõrge tasemeni normaliseerimine operatiivandmete andmebaasides siiski otstarbekam tegevus kui denormaliseerimine.

Püstitatud eesmärgid saavutati ning mõlemad näiteandmebaasid on realiseeritud. Lisaks selgus huvitava tähelepanekuna, et domeenide kasutamine denormaliseeritud andmebaasis mõjutas andmebaaside andmemahtusid või andmemahtude mõõtmise programmi tööd tabelites kus denormaliseerimist läbi ei viidud. Eelnimetatud tähelepanek on näide edasistest uurimissuundadest.

Lõpetuseks soovin tänada töö juhendajat Erki Eessaar't osutatud abi eest.

Kasutatud kirjandus

[1] M. Ossipova, H. Pikkaro ja S. Peek, „Ülikooli infosüsteemi õppeainete arvestuse allsüsteem: Andmebaasid II projekt," Tallinna Tehnikaülikool, Tallinn, 2015.

[2] IBM Corp, *Rational Rose,* 2010.

[3] S. Hoberman, Data Modeler's workbench: Tools and Techniques for Analysis and Design, New York: John Wiley & Sons, Inc, 2002.

[4] Eesti Keele Instituut, Eesti õigekeelsussõnaraamat ÕS 2013, Tallinn: Eesti Keele Sihtastus, 2013.

[5] T. Connoly ja C. Begg, Database systems: A Practical Approach to Design, Implementation and Management, 3rd Edition., Universtiy of Paisley : Pearson, 2002.

[6] C. Knowles, „6NF Conceptual Models and Data Warehousing 2.0," 2012. [Võrgumaterjal]. Available: http://sais.aisnet.org/2012/Knowles.pdf. [Kasutatud 2 Aprill 2016].

[7] E. Eessaar, „Teema 9. Andmebaasi loogilise disani tulemuse parandamine ja headuse kontrollimine," Tallinna Tehnikaülikool, 2016.

[8] E. Eessaar, Andmebaaside projekteerimine, Tallinn: TTÜ kirjastus, 2008.

[9] E. Eessaar, „Teema 14. Andmebaaside projekteerimine (füüsilisest disainist töötava andmebaasi hooldamiseni)," Tallinna Tehnikaülikool, 2016.

[10] „http://db-engines.com/en/ranking," 2016. [Võrgumaterjal]. Available: http://db-engines.com/en/ranking. [Kasutatud 22 05 2016].

[11] The PostgreSQL Global Development Group, „Create Schema," 1996-2016. [Võrgumaterjal]. Available: http://www.postgresql.org/docs/9.4/static/sql-createschema.html. [Kasutatud 14 Märts 2016].

[12] The PostgreSQL Global Development Group, „System Administration Functions," 1996-2016. [Võrgumaterjal]. Available: http://www.postgresql.org/docs/9.0/static/functions-admin.html . [Kasutatud 28 Märts 2016].

[13] The PostgreSQL Global Development Group, „Create Function," 1996-2016. [Võrgumaterjal]. Available: http://www.postgresql.org/docs/9.4/static/sql-

createfunction.html. [Kasutatud 31 Märts 2016].

[14] „LocMetrics - C#, C++, Java, and SQL,“ 2006-2010. [Võrgumaterjal]. Available: http://www.locmetrics.com/index.html . [Kasutatud 4 Mai 2016].

[15] „Source lines of code,“ 2016. [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/Source_lines_of_code. [Kasutatud 20 Aprill 2016].

[16] Gudu Software, „Instant SQL Formatter,“ 2001-2016. [Võrgumaterjal]. Available: http://www.dpriver.com/pp/sqlformat.htm . [Kasutatud 5 Mai 2016].

[17] The PostgreSQL Development Group, „Explain,“ 1996-2016. [Võrgumaterjal]. Available: http://www.postgresql.org/docs/9.4/static/sql-explain.html . [Kasutatud 1 Mai 2016].

[18] The PostgreSQL Global Development Group, „Create trigger,“ 1996-2016. [Võrgumaterjal]. Available: http://www.postgresql.org/docs/9.4/static/sql-createtrigger.html. [Kasutatud 13 Mai 2016].

[19] „Tallinna Tehnikaülikooli koduleht,“ [Võrgumaterjal]. Available: http://www.ttu.ee/. [Kasutatud 2016 Aprill 29].

[20] „Tallinna Tehnikaülikooli õppeinfosüsteem,“ [Võrgumaterjal]. Available: https://ois.ttu.ee/pls/portal/ois2.ois_public.main. [Kasutatud 29 Aprill 2016].

[21] ApexSQL LLC, „ApexSQL Generate,“ 2016. [Võrgumaterjal]. Available: http://www.apexsql.com/sql_tools_generate.aspx . [Kasutatud 30 Aprill 2016].

[22] The PostgreSQL Global Development Group, „Copy,“ 1996-2016. [Võrgumaterjal]. Available: http://www.postgresql.org/docs/9.4/static/sql-copy.html . [Kasutatud 1 Mai 2016].

[23] The PostgreSQL Global Development Group, „Insert,“ 1996-2016. [Võrgumaterjal]. Available: http://www.postgresql.org/docs/9.4/static/sql-insert.html. [Kasutatud 1 Mai 2016].

[24] The PostgreSQL Global Development Group, „Analyze,“ 1996-2016. [Võrgumaterjal]. Available: http://www.postgresql.org/docs/9.4/static/sql-analyze.html. [Kasutatud 2 Mai 2016].

[25] A. Marin, „Serializable SnapShot Isolation ja SnapShot Isolation protokollide mõju PostgreSQL andmebaasides kitsenduste jõustamisele: bakalaureusetöö,“ Tallinna Tehnikaülikool, Tallinn, 2014.

Lisa 1 – Olemitüüpide ja atribuutide kirjeldused

Tabelis 8 on esitatud olemitüüpide kirjeldused.

Tabel 8. Olemitüüpide kirjeldused.

Olemitüübi nimi (aliased)	**Kuuluvus registrisse**	**Definitsioon**
Õppeaine	Õppeainete register	Õppeaine on ülikoolis õpetatav teadus-, tehnika-, kunsti- või ametiala sisu. Õppekavade sisu on jagatud õppeaineteks, mida annab õppejõud.
Õppeaine seisundi liik	Klassifikaatorite register	Õppeaine ja ülikooli vahelise suhte hetkeolukorra iseloomustus. Näiteks: aktiivne, mitteaktiivne, kavast maha võetud, kustutatud.
Keel	Klassifikaatorite register	Keel on inimeste poolt kasutatav märgisüsteem, kommunikatsiooni või arutluse vahend, mis kasutab sümboleid ja teisi märke ja nende kombineerimise reegleid.
Semester	Klassifikaatorite register	Semester on kaheks pooleks jaotatava õppeaasta üks pool.
Töötaja_aines	Töötajate register	Töötaja aines iseloomustab, kui kaua on töötaja ametis töötanud.
Roll_aines	Klassifikaatorite register	Töötaja rolli õppeaines iseloomustus. Näiteks: Õppejõud annab harjutustunde, Õppejõud annab loenguid, Õppejõud annab praktikumitunde.
Töötaja	Töötajate register	Ülikoolis kas töölepinguga või tunnitasulisena töötav isik (nt õppejõud, dekanaadi juhataja).
Amet	Klassifikaatorite register	Amet on ülikooli töötaja töölepingus sätestatud ametikohustuse üldnimetus. Ameti näiteks on assistent, lektor, dotsent, professor.
Töötaja_seisundi_liik	Klassifikaatorite register	Töötaja ja ülikooli vahelise suhte hetkeolukorra iseloomustus. Näiteks: töötav, puhkusel.

Olemitüübi nimi (aliased)	Kuuluvus registrisse	Definitsioon
Isik	Isikute register	Mistahes ülikooliga seotud füüsiline isik. Isik võib olla seotud ülikooliga läbi töösuhte, õppimise või lepingu.
Isiku_seisundi_liik	Klassifikaatorite register	Isiku ja ülikooli vahelise suhte hetkeolukorra iseloomustus. Näiteks: on õigus muuta õppeaine andmeid või mitte.
Üliõpilane	Üliõpilaste register	"Üliõpilane on isik, kes on vastu võetud (immatrikuleeritud) ülikooli rakenduskõrgharidus-, bakalaureuse-, magistri-, doktoriõppe või bakalaureuse- ja magistriõppe integreeritud õppekavadel põhineva õppe täiskoormusega või osakoormusega õppesse." ("Ülikooliseadus")
Klassifikaator	Klassifikaatorite register	Klassifikaatorid on "mistahes andmed, mida kasutatakse andmebaasis teiste andmete liigitamiseks või andmebaasis olevate andmete seostamiseks väljaspool organisatsiooni vastutusala oleva informatsiooniga." (Chisholm, 2000)
Üliõpilase_seisundi_liik	Klassifikaatorite register	Üliõpilase ja ülikooli vahelise suhte hetkeolukorra iseloomustus. Näiteks: immatrikuleeritud, eksmatrikuleeritud.
Kontrollvorm	Klassifikaatorite register	Kontrollvorm on viis õppeaine läbimise jooksul saadud teadmisi kontrollimiseks. Näiteks: Eksam, hinne, arvestus.

Tabelis 9 on esitatud olemitüüpide atribuudid ja nende kirjeldused.

Tabel 9. Olemitüüpide atribuudid ja nende kirjeldused.

Olemitüübi nimi	Atribuudi Nimi	Atribuudi definitsioon	Näiteväärtus
Isik	Eesnimi	“Lapsele tema sünni registreerimisel antav nimi.” [4] {Registreerimine on kohustuslik. Eesnimi ei tohi sisaldada numbreid	Kadri

Olemitüübi nimi	Atribuudi Nimi	Atribuudi definitsioon	Näiteväärtus
		või olla tühi string või olla ainult tühikutest koosnev string. Eesnimi võib sisaldada ainult tähti, tühikuid või kriipse. Kui isikul on mitu eesnime, siis need on eraldatud ühe tühiku või kriipsuga.}	
Isik	Perekonnanimi	"Vanemailt lapsele kanduv nimi(meil eesnime järel)." [4] {Registreerimine on kohustuslik. Perenimi ei tohi olla tühi string või ainult tühikutest koosnev string. Perenimi ei tohi sisaldada numbreid. Perenimi võib sisaldada ainult tähti, tühikuid või kriipse. Kui isikul on mitu perenime, siis need on eraldatud ühe tühiku või kriipsuga.}	Mets
Isik	Email	Aadress, millele saab üle võrgu(ühest arvutist või tööjaamast teise) saata isikule mõeldud kirjalikke sõnumeid. {Isiku unikaalne identifikaator. Registreerimine on kohustuslik. Email peab vastama mustrile string@string.string.}	kadri@hot.ee
Isik	Kasutajanimi	Isikule antud unikaalne nimi, mida kasutatakse tema õppeinfosüsteemi poolseks autentimiseks. {Isiku unikaalne identifikaator. Registreerimine on kohustuslik. Kasutajanimi ei tohi olla lühem kui 5 tähemärki ja pikem kui 20 tähemärki. Kasutajanimi ei tohi sisaldada tühikuid ja numbreid. Kasutajanimi ei tohi koosneda ainult tühikutest.}	KadriM
Isik	Telefon	Isiku telefoninumber. {Registreerimine on kohustuslik. Telefoninumber peab koosnema ainult numbritest, ei tohi sisaldada	+372 1234567

Olemitüübi nimi	Atribuudi Nimi	Atribuudi definitsioon	Näiteväärtus
		tähemärke. Telefoninumber peab algama ’’+’’ märgiga ja riigi suunakoodiga, millele järgneb tühik.}	
Isik	Parool	Isiku identsust tõendav teadmuslik(miski, mida isik teab) volitustõend. {Registreerimine on kohustuslik. Ei saa olla tühi ning ei saa sisaldada tühikuid.}	K31mkn
Klassifikaator	Kood	Klassifikaatori väärtust esitav kood, mida saab kasutada selle väärtuse lühidalt esitamiseks. Kood võib olla tekstiline või numbriline väärtus. {Klassifikaatori unikaalne identifikaator. Registreerimine on kohustuslik.}	1
Klassifikaator	Nimetus	Klassifikaatori väärtuse ametlik nimetus. {Klassifikaatori unikaalne identifikaator. Registreerimine on kohustuslik. Nimetus ei tohi olla tühi string või ainult tühikutest koosnev string.}	Loodud
Klassifikaator	Kirjeldus	Klassifikaatori väärtuse vabatekstiline kirjeldus. {Kirjeldus võib olla tühi string või anult tühikutest koosnev string.}	Õppeaine algseisund
Üliõpilane	Martikli_nr	Kliendile(antud juhul üliõpilasele, kes soovib vaadata õpetavaid aineid) kindla algoritmi alusel koostatav ülikooli teiste üliõpilaste seas unikaalne kood. {Kliendi unikaalne identifikaator. Registreerimine on kohustuslik. Martiklinumber koosneb kuuest	990999

Olemitüübi nimi	Atribuudi Nimi	Atribuudi definitsioon	Näiteväärtus
		numbrimärgist.}	
Töötaja_ained	Alguse_aeg	Töötaja tööle asumisaeg. {Registreerimine on kohustuslik. Alguse aeg ei tohi olla väiksem kui ülikooli asutamise aeg. Alguse aeg ei tohi olla suurem kui ülikooli asutamise aeg pluss 200 aastat }	01.09.2000
Töötaja_ained	Lõpu_aeg	Töötaja töölt lahkumisaeg. {Lõpu aeg ei tohi olla väiksem kui alguse aeg. Lõpu aeg ei tohi olla suurem, kui ülikooli asutamise aeg pluss 200 aastat}	09.10.2009
Õppeaine	Ainekood	Õppeaine tunnuseks on ainekood, mis luuakse ÕISi poolt automaatselt ainet õpetatava struktuuriüksuse koodi alusel. Kasutuses olnud ainekoode ei ainta uutele ainetele. Ainekood koosneb komponentidest, mis väljendavad aine kuuluvust mingile üksusele. Ainekoodi järgi on võimalik ainet andmebaasist otsida. {Õppeaine unikaalne identifikaator. Registreerimine on kohustuslik. Ainekood ei tohi olla tühi string ega tühikutest koosnev string. Ainekood koosneb kolmest tähemärgist ja neljast numbrimärgist.}	idu0220
Õppeaine	Nimetus	Õppeaine nimi, mis on omistatud õpetatavale õppeainele tema sisu järgi. {Registreerimine on kohustuslik. Nimetus ei tohi olla tühi string ega tühikutest koosnev string.}	Andmebaasid I
Õppeaine	Punktid	Õppeaine punktid, mis saadakse antud õppeaine läbimisel. Arvestust Euroopa ainepunktisüsteemi alusel ning EAP tähistab ainepunkti.	5

Olemitüübi nimi	Atribuudi Nimi	Atribuudi definitsioon	Näiteväärtus
		{Registreerimine on kohustuslik. Täisarv. Peab olema suurem kui 0 ja väiksem kui 200}	
Õppeaine	Sisu	Õppeaine kirjeldus, mida antud õppeaines õpetatakse või millega kokkupuututakse. {Registreerimine on kohustuslik. Sisu ei tohi olla tühi string ega tühikutest koosnev string.}	SQL andmebaasikeel
Õppeaine	Koduleht	Kirjeldus, mis sisaldab juhiseid õppeaine kodulehele pääsemiseks, kus on üleval antud õppeaine materjalid. {Registreerimine ei ole kohustuslik.}	maurus.ttu.ee/aine_index.php?aine=342
Õppeaine	Loomise_aeg	Õppeaine registreerimise kuupäev. {Registeerimine kohustuslik. Loomise aeg ei tohi olla suurem kui kinnitamise aeg. Loomise aeg ei tohi olla väiksem kui ülikooli loomise aeg ja suurem kui ülikooli loomise kuupäev pluss 200 aastat.}	05.05.2005
Õppeaine	Kinnitamise_aeg	Õppeaine kinnitamise kuupäev. {Registreerimine ei ole kohustuslik. Kinnitamise aeg ei tohi olla väiksem kui loomise aeg. Kinnitamise aeg ei tohi olla suurem kui ülikooli loomise kuupäev pluss 200 aastat.}	31.10.2012

Lisa 2 – SQL andmebaasiobjektide nimetamine

Viis, kuidas SQL-objekte töös nimetatakse on ära toodud tabelis 10.

Tabel 10. SQL-andmebaasiobjektide nimetamine.

SQL-objekti tüüp	**Nimi**	**Näide**
Tabel	Olemitüübi nimi, kus tühik on asendatud alakriipsuga ja täpitähed on asendatud vastavalt: ü=>y, õ => o, ö => o.	Tootaja_aines
Veerg	Atribuudi nimi, kus tühik on asendatud alakriipsuga ning täpitähed on asendatud vastavalt: ü => y, õ => o, ö => o.	Oppeaine_seisundi_liik_kood
Primaarvõti	Tabelinimi_kood või ilma tühikuta ehk alakriipsuta	Oppimine_kood, Isikukood
Primaarvõtme kitsendus	PK_tabelinimi	PK_Isik
Välisvõtme kitsendus	FK_tabelinimi_veerunimi	FK_Oppeaine_isikukood
Unikaalsuse kitsendus	AK_tabelinimi_veerunimi või unikaalsete kombinatsioonide korral AK_tabelinimi_kombinatsioon	AK_Isik_email või AK_Tootaja_aines_kombinatsioon
Indeks	idx_tabelinimi_veerunimi	idx_oppeaine_isikukood
Kontroll kitsendus	chk_tabelinimi_kirjeldus	chk_Isik_isikukood

Lisa 3 – 5NK andmebaasi tõestus

Järgneva tõestuse koostamiseks on kasutatud Erki Eessaare teema 9 konspekti „Andmebaasi loogilise disaini tulemuse ja headuse kontrollimine“ [7]. Tõestus lähtub siinkohal tõestamata eeldusest, et tabelid juba on Boyce/Codd normaalkujul. Lihtsasse võtmesse kuulub üks veerg.

On sõnastatud ja tõestatud lihtsaid reegleid tabeli viiendal normaalkujul oleku kontrollimiseks.

- **R1:** Kui tabel on kolmandal normaalkujul või ka Boyce/Coddi normaalkujul ja iga selle kandidaatvõti on lihtne, siis on see tabel ka viiendal normaalkujul.
- **R2:** Kui tabel on Boyce/Coddi normaalkujul ja sel on vähemalt üks veerg, mis ei kuulu ühtegi kandidaatvõtmesse, siis see tabel on ka viiendal normaalkujul.
- **R3:** Kui tabel on Boyce/Coddi normaalkujul ja mõni selle kandidaatvõti on lihtne, siis on see relvar ka neljandal normaalkujul (aga ei pruugi olla viiendal normaalkujul).

Järgnevad tabelid(vt tabel 11-13) toovad välja R1, R2 ja R3 tulenevad põhjendused kõigi tabelite kohta.

Tabel 11. R1 põhjendus.

Tabeli nimi	**R1: Kõik kandidaatvõtmed on lihtvõtmed**
Amet	(amet_kood), (nimetus)
Isik	(isikukood), (kasutajanimi), (e_mail)
Isiku_seisundi_liik	(isiku_seisundi_liik_kood), (nimetus)
Keel	(keel_kood), (nimetus)
Kontrollvorm	(kontrollvorm_kood), (nimetus)
Oppeaine	(ainekood)
Oppeaine_seisundi_liik	(oppeaine_seisundi_liik_kood), (nimetus)
Roll_aines	(roll_aines_kood), (nimetus)
Semester	(semester_kood), (nimetus)
Tootaja	(isikukood)

Tabeli nimi	**R1: Kõik kandidaatvõtmed on lihtvõtmed**
Tootaja_seisundi_liik	(tootaja_seisundi_liik_kood), (nimetus)
Yliopilane	(isikukood)
Yliopilase_seisundi_liik	(yliopilase_seisundi_liik_kood), (nimetus)

Tabel 12. R2 põhjendus.

Tabeli nimi	**R2: Leidubvähemalt üks veerg, mis ei kuulu ühtegi kandidaatvõtmesse**
Tootaja_aines	lopu_aeg

Tabel 13. R3 põhjendus.

Tabeli nimi	**R3: Mõni kandidaatvõti on lihtne**
Oppimine	(oppimine_kood)

Lisa 4 – Denormaliseerimise algoritm

Tabelis 14 on esitatud denormaliseerimise algoritmi kasutatav küsimustik koos võimalike valikvariantidega. Vastuse lõpus sulgudes olev arv näitab kui palju punkte antud valikuvariandi annab.

Tabel 14. Denormaliseerimise algoritmi küsimustik.

Küsimus	**Valik**
Mis tüüpi suhtega on tegemist peremees tabeli ja sõltuva tabeli vahel?	Osa-terviku (agregatsiooni või kompositsiooni) seos, hierarhia. Nt. tellimus sisaldab tellimuste ridu, kauba kategooria sisaldab alamkategooriaid. (20) (Annab kõige rohkem punkte, sest osa ja terviku andmeid kasutatakse ilmselt sageli üheskoos) Suhe võrdsete vahel. Kummalegi tabeli rida võib eksisteerida teise tabeli reast sõltumatult. Nt. õppejõud koostab dokumendi. (-10) Definitsioon. Peremeestabelis olevad andmed määravad sõltuvas tabelis olevate andmete tähenduse. Nt. õppeaine spetsifikatsioon ja õppeaine tegelik õpetamine või klassifikaator ja selle poolt klassifitseeritavad andmed. Kui olemitüüpide vahel on M:N seosetüüp, siis luuakse selle esitamiseks vahetabel. Suhe selle vahetabeli ja peremeestabelite vahel on defineeriv suhe. (-20)
Milline on ühe peremeestabeli reaga seotud sõltuva tabeli ridade arv?	1 kuni 5 (20) (Annab kõige rohkem punkte, sest denormaliseerimise korral on tulemuseks olevas tabelis peremeestabelile vastavad read esitatud korduvalt – korduste arvu määrab sõltuva tabeli ridade arv. Mida väiksem on korduvate andmete hulk, seda parem). 5 kuni 100 (-10) 100 ja rohkem (-20)
Kui palju veerge on peremeestabelis, jättes arvestamata primaarvõtme veerud?	Vähem kui 10 (20) (Annab kõige rohkem punkte, sest denormaliseerimise korral on tulemuseks olevas tabelis peremeestabelile vastavad read esitatud korduvalt – korduste arvu määrab sõltuva tabeli ridade arv. Mida väiksem on korduvate andmete hulk, seda parem).

Küsimus	Valik
	10 kuni 20 (-10) Rohkem kui 20 (-20)
Kui tehakse päring sõltuva tabeli kohta, siis kui sageli soovitakse andmeid ka peremeestabelist?	Päringus soovitakse sageli andmeid peremeestabelist(30) (Annab kõige rohkem punkte, sest denormaliseerimise eesmärk on töökiiruse huvides vähendada ühendamisoperatsiooni kasutavate päringute hulka) Päringus ei soovita sageli andmeid peremeestabelist(-30)
Kas peremeestabelisse on lähiajal plaanis lisada uusi veerge või siduda uusi tabeleid?	Jah(-20) Ei(20) (Annab kõige rohkem punkte, sest denormaliseerimise korral on tulemuseks olevas tabelis peremeestabelile vastavad read esitatud korduvalt – korduste arvu määrab sõltuva tabeli ridade arv. Mida väiksem on korduvate andmete hulk, seda parem).
Kas peremeestabelis ja sõltuvas tabelis toimub ajaperioodis sarnane arv lisamisi ja muutmisi?	Jah (toimub sarnane arv lisamisi ja muutmisi) (20) (Annab kõige rohkem punkte, sest denormaliseerimine mõjub andmete muutmise operatsioonide töökiirusele pigem halvasti). Ei (ei toimu sarnane arv lisamisi ja muutmisi) (-20)
KOKKU PUNKTE	

Järgenevalt on esitatud denormaliseerimise algoritm antud lõputöös kasutatava konseptuaalse andmemudeli põhjal.

Konseptuaalse mudeli põhjal jaotatakse tabelite vahelised seosed e suhted kategooriatesse:

a) Üks-üks seosed
   - Isik-Töötaja
   - Isik-Yliopilane

b) Klassifikaatorite/põhiandmete tabelite omavahelised seosed
   - Isik-Isiku_seisundi_liik
   - Tootaja-Tootaja_seisundi_liik

- Yliopilane-Yliopilase_seisundi_liik
- Tootaja-Amet
- Oppeaine-Keel
- Oppeaine-Kontrollvorm
- Oppeaine-Semester
- Oppeaine-Oppeaine_seisundi_liik
- Oppeaine-Tootaja

c) Transaktsiooniliste andmetega omavahelised seosed

- Tootaja_aines-Oppimine

d) Klassifikaatorite/põhiandmete tabelite ning transaktsiooniliste andmetega tabelite vahelised seosed

- Tootaja_aines-Roll_aines
- Tootaja-Tootaja_aines
- Oppeaine-Tootaja_aines
- Yliopilane-Oppimine

Teiseks ülesandeks vastavalt alogritmile on vastata suhete kohta küsimustikule. Alustatakse suhetest mis kuuluvad kategooria a alla ning liigutakse edasi järgnevate kategooriate juurde. Tabelites 15–29 on esitatud denormaliseerimis algoritmi küsimustikud kõikide suhete põhjal.

Suhe: Isik-Yliopilane (*Isik* on peremeestabel ja *Yliopilane* on sõltuv tabel)

Tabel 15. Denormaliseerimise algoritmi küsimustik suhte Isik-Yliopilane alusel.

Küsimus	**Valik**
Mis tüüpi suhtega on tegemist peremees tabeli ja sõltuva tabeli vahel?	Osa-terviku (agregatsiooni või kompositsiooni) seos, hierarhia. Nt. tellimus sisaldab tellimuste ridu, kauba kategooria sisaldab alamkategooriaid. (20) (Annab kõige rohkem punkte, sest osa ja terviku andmeid kasutatakse ilmselt sageli üheskoos)
Milline on ühe peremeestabeli reaga seotud sõltuva tabeli ridade	1 kuni 5 **(20)**

Küsimus	Valik
arv?	
Kui palju veerge on peremeestabelis, jättes arvestamata primaarvõtme veerud?	Vähem kui 10 **(20)**
Kui tehakse päring sõltuva tabeli kohta, siis kui sageli soovitakse andmeid ka peremeestabelist?	Päringus soovitakse sageli andmeid peremeestabelist**(30)**
Kas peremeestabelisse on lähiajal plaanis lisada uusi veerge või siduda uusi tabeleid?	Ei **(20)**
Kas peremeestabelis ja sõltuvas tabelis toimub ajaperioodis sarnane arv lisamisi ja muutmisi?	Ei (ei toimu sarnane arv lisamisi ja muutmisi) **(-20)**
KOKKU PUNKTE	**90** Otsus: **Kaaluda denormaliseerimist**

Suhe: Isik-Isiku_seisundi_liik(*Isiku_seisundi_liik* on peremees tabel ja *Isik* on sõltuv tabel)

Tabel 16. Denormaliseerimise algoritmi küsimustik suhte Isik-Isiku_seisundi_liik alusel.

Küsimus	Valik
Mis tüüpi suhtega on tegemist peremees tabeli ja sõltuva tabeli vahel?	Definitsioon. Peremeestabelis olevad andmed määravad sõltuvas tabelis olevate andmete tähenduse. Nt. õppeaine spetsifikatsioon ja õppeaine tegelik õpetamine või klassifikaator ja selle poolt klassifitseeritavad andmed. Kui olemitüüpide vahel on M:N seosetüüp, siis luuakse selle esitamiseks vahetabel. Suhe selle vahetabeli ja peremeestabelite vahel on defineeriv suhe. **(-20)**
Milline on ühe peremeestabeli reaga seotud sõltuva tabeli ridade arv?	100 ja rohkem **(-20)**
Kui palju veerge on peremeestabelis, jättes arvestamata primaarvõtme veerud?	Vähem kui 10 **(20)**
Kui tehakse päring sõltuva tabeli kohta, siis kui sageli soovitakse andmeid ka peremeestabelist?	Päringus soovitakse sageli andmeid peremeestabelist **(30)**

Küsimus	Valik
Kas peremeestabelisse on lähiajal plaanis lisada uusi veerge või siduda uusi tabeleid?	Ei **(20)**
Kas peremeestabelis ja sõltuvas tabelis toimub ajaperioodis sarnane arv lisamisi ja muutmisi?	Ei (ei toimu sarnane arv lisamisi ja muutmisi) **(-20)**
KOKKU PUNKTE	**10** Otsus: **Kaaluda denormaliseerimist**

Suhe: Tootaja-Tootaja_seisundi_liik (*Tootaja_seisundi_liik* on peremeestabel ja *Tootaja* on sõltuv tabel)

Tabel 17. Denormaliseerimise algoritm suhte Tootaja-Tootaja_seisundi_liik alusel.

Küsimus	Valik
Mis tüüpi suhtega on tegemist peremees tabeli ja sõltuva tabeli vahel?	Definitsioon. Peremeestabelis olevad andmed määravad sõltuvas tabelis olevate andmete tähenduse. Nt. õppeaine spetsifikatsioon ja õppeaine tegelik õpetamine või klassifikaator ja selle poolt klassifitseeritavad andmed. Kui olemitüüpide vahel on M:N seosetüüp, siis luuakse selle esitamiseks vahetabel. Suhe selle vahetabeli ja peremeestabelite vahel on defineeriv suhe. **(-20)**
Milline on ühe peremeestabeli reaga seotud sõltuva tabeli ridade arv?	100 ja rohkem **(-20)**
Kui palju veerge on peremeestabelis, jättes arvestamata primaarvõtme veerud?	Vähem kui 10 **(20)**
Kui tehakse päring sõltuva tabeli kohta, siis kui sageli soovitakse andmeid ka peremeestabelist?	Päringus soovitakse sageli andmeid peremeestabelist **(30)**
Kas peremeestabelisse on lähiajal plaanis lisada uusi veerge või siduda uusi tabeleid?	Ei **(20)**
Kas peremeestabelis ja sõltuvas tabelis toimub ajaperioodis sarnane arv lisamisi ja muutmisi?	Ei (ei toimu sarnane arv lisamisi ja muutmisi) **(-20)**

Küsimus	Valik
KOKKU PUNKTE	**10** Otsus: **Kaaluda denormaliseerimist**

Suhe: Yliopilane-Yliopilase_seisundi_liik (*Yliopilase_seisundi_liik* on peremeestabel ja *Yliopilane* on sõltuv tabel)

Tabel 18. Denormaliseerimise algoritmi küsimustik suhte Yliopilane-Yliopilase_seisundi_liik alusel.

Küsimus	Valik
Mis tüüpi suhtega on tegemist peremees tabeli ja sõltuva tabeli vahel?	Definitsioon. Peremeestabelis olevad andmed määravad sõltuvas tabelis olevate andmete tähenduse. Nt. õppeaine spetsifikatsioon ja õppeaine tegelik õpetamine või klassifikaator ja selle poolt klassifitseeritavad andmed. Kui olemitüüpide vahel on M:N seosetüüp, siis luuakse selle esitamiseks vahetabel. Suhe selle vahetabeli ja peremeestabelite vahel on defineeriv suhe. **(-20)**
Milline on ühe peremeestabeli reaga seotud sõltuva tabeli ridade arv?	100 ja rohkem **(-20)**
Kui palju veerge on peremeestabelis, jättes arvestamata primaarvõtme veerud?	Vähem kui 10 **(20)**
Kui tehakse päring sõltuva tabeli kohta, siis kui sageli soovitakse andmeid ka peremeestabelist?	Päringus soovitakse sageli andmeid peremeestabelist **(30)**
Kas peremeestabelisse on lähiajal plaanis lisada uusi veerge või siduda uusi tabeleid?	Ei **(20)**
Kas peremeestabelis ja sõltuvas tabelis toimub ajaperioodis sarnane arv lisamisi ja muutmisi?	Ei (ei toimu sarnane arv lisamisi ja muutmisi) **(-20)**
KOKKU PUNKTE	**10** Otsus: **Kaaluda denormaliseerimist**

Suhe: Tootaja-Amet (*Amet* on peremees tabel ja *Tootaja* on sõltuv tabel)

Tabel 19. Denormaliseerimise algoritmi küsimustik suhte Tootaja-Amet alusel.

Küsimus	**Valik**
Mis tüüpi suhtega on tegemist peremees tabeli ja sõltuva tabeli vahel?	Definitsioon. Peremeestabelis olevad andmed määravad sõltuvas tabelis olevate andmete tähenduse. Nt. õppeaine spetsifikatsioon ja õppeaine tegelik õpetamine või klassifikaator ja selle poolt klassifitseeritavad andmed. Kui olemitüüpide vahel on M:N seosetüüp, siis luuakse selle esitamiseks vahetabel. Suhe selle vahetabeli ja peremeestabelite vahel on defineeriv suhe. **(-20)**
Milline on ühe peremeestabeli reaga seotud sõltuva tabeli ridade arv?	100 ja rohkem **(-20)**
Kui palju veerge on peremeestabelis, jättes arvestamata primaarvõtme veerud?	Vähem kui 10 **(20)**
Kui tehakse päring sõltuva tabeli kohta, siis kui sageli soovitakse andmeid ka peremeestabelist?	Päringus soovitakse sageli andmeid peremeestabelist **(30)**
Kas peremeestabelisse on lähiajal plaanis lisada uusi veerge või siduda uusi tabeleid?	Ei **(20)**
Kas peremeestabelis ja sõltuvas tabelis toimub ajaperioodis sarnane arv lisamisi ja muutmisi?	Ei (ei toimu sarnane arv lisamisi ja muutmisi) **(-20)**
KOKKU PUNKTE	**10** Otsus: **Kaaluda denormaliseerimist**

Suhe: Oppeaine-Keel (*Keel* on peremeestabel ja *Oppeaine* on sõltuv tabel)

Tabel 20. Denormaliseerimise algoritmi küsimustik suhte Oppeaine-Keel alusel.

Küsimus	**Valik**
Mis tüüpi suhtega on tegemist peremees tabeli ja sõltuva tabeli vahel?	Definitsioon. Peremeestabelis olevad andmed määravad sõltuvas tabelis olevate andmete tähenduse. Nt. õppeaine spetsifikatsioon ja õppeaine tegelik õpetamine või klassifikaator ja selle poolt klassifitseeritavad andmed. Kui

Küsimus	Valik
	olemitüüpide vahel on M:N seosetüüp, siis luuakse selle esitamiseks vahetabel. Suhe selle vahetabeli ja peremeestabelite vahel on defineeriv suhe. **(-20)**
Milline on ühe peremeestabeli reaga seotud sõltuva tabeli ridade arv?	100 ja rohkem **(-20)**
Kui palju veerge on peremeestabelis, jättes arvestamata primaarvõtme veerud?	Vähem kui 10 **(20)**
Kui tehakse päring sõltuva tabeli kohta, siis kui sageli soovitakse andmeid ka peremeestabelist?	Päringus soovitakse sageli andmeid peremeestabelist **(30)** (Õppeainete nimekirjas soovitakse näha ka keele nimetust, milles antud õppeaine läbi viiakse)
Kas peremeestabelisse on lähiajal plaanis lisada uusi veerge või siduda uusi tabeleid?	Ei **(20)**
Kas peremeestabelis ja sõltuvas tabelis toimub ajaperioodis sarnane arv lisamisi ja muutmisi?	Ei (ei toimu sarnane arv lisamisi ja muutmisi; klassifikaatorid registreeritakse juba enne andmete ülekandmist andmebaasi ning klassifikaatorite hulk suureneb väga aeglaselt, õppeainete andmeid aga lisatakse regulaarselt) **(-20)**
KOKKU PUNKTE	**10** Otsus: **Kaaluda denormaliseerimist**

Suhe: Oppeaine-Kontrollvorm (*Kontrollvorm* on peremeestabel ja *Oppeaine* on sõltuv tabel)

Tabel 21. Denormaliseerimise algoritmi küsimustik suhte Oppeaine-Kontrollvorm alusel.

Küsimus	Valik
Mis tüüpi suhtega on tegemist peremees tabeli ja sõltuva tabeli vahel?	Definitsioon. Peremeestabelis olevad andmed määravad sõltuvas tabelis olevate andmete tähenduse. Nt. õppeaine spetsifikatsioon ja õppeaine tegelik õpetamine või klassifikaator ja selle poolt klassifitseeritavad andmed. Kui olemitüüpide vahel on M:N seosetüüp, siis luuakse selle esitamiseks vahetabel. Suhe selle vahetabeli ja peremeestabelite vahel on defineeriv suhe. **(-20)**

Küsimus	Valik
Milline on ühe peremeestabeli reaga seotud sõltuva tabeli ridade arv?	100 ja rohkem **(-20)**
Kui palju veerge on peremeestabelis, jättes arvestamata primaarvõtme veerud?	Vähem kui 10 **(20)**
Kui tehakse päring sõltuva tabeli kohta, siis kui sageli soovitakse andmeid ka peremeestabelist?	Päringus soovitakse sageli andmeid peremeestabelist **(30)** (Õppeainete nimekirjas soovitakse näha ka kontrollvormi nimetust)
Kas peremeestabelisse on lähiajal plaanis lisada uusi veerge või siduda uusi tabeleid?	Ei **(20)**
Kas peremeestabelis ja sõltuvas tabelis toimub ajaperioodis sarnane arv lisamisi ja muutmisi?	Ei (ei toimu sarnane arv lisamisi ja muutmisi; klassifikaatorid registreeritakse juba enne andmete ülekandmist andmebaasi ning klassifikaatorite hulk suureneb väga aeglaselt, õppeainete andmeid aga lisatakse regulaarselt) **(-20)**
KOKKU PUNKTE	**10** Otsus: **Kaaluda denormaliseerimist**

Suhe: Oppeaine-Semester (*Semester* on peremeestabel ja *Õppeaine* on sõltuv tabel)

Tabel 22. Denormaliseerimise algoritm küsimustik suhte Oppeaine-Semester alusel.

Küsimus	Valik
Mis tüüpi suhtega on tegemist peremees tabeli ja sõltuva tabeli vahel?	Definitsioon. Peremeestabelis olevad andmed määravad sõltuvas tabelis olevate andmete tähenduse. Nt. õppeaine spetsifikatsioon ja õppeaine tegelik õpetamine või klassifikaator ja selle poolt klassifitseeritavad andmed. Kui olemitüüpide vahel on M:N seosetüüp, siis luuakse selle esitamiseks vahetabel. Suhe selle vahetabeli ja peremeestabelite vahel on defineeriv suhe. **(-20)**
Milline on ühe peremeestabeli reaga seotud sõltuva tabeli ridade arv?	100 ja rohkem **(-20)**

Küsimus	Valik
Kui palju veerge on peremeestabelis, jättes arvestamata primaarvõtme veerud?	Vähem kui 10 **(20**)
Kui tehakse päring sõltuva tabeli kohta, siis kui sageli soovitakse andmeid ka peremeestabelist?	Päringus soovitakse sageli andmeid peremeestabelist **(30)** (Õppeainete nimekirja soovitakse näha ka semestri nimetust)
Kas peremeestabelisse on lähiajal plaanis lisada uusi veerge või siduda uusi tabeleid?	Ei **(20)**
Kas peremeestabelis ja sõltuvas tabelis toimub ajaperioodis sarnane arv lisamisi ja muutmisi?	Ei (ei toimu sarnane arv lisamisi ja muutmisi; klassifikaatorid registreeritakse juba enne andmete ülekandmist andmebaasi ning klassifikaatorite hulk suureneb väga aeglaselt, õppeainete andmeid aga lisatakse regulaarselt) **(-20)**
KOKKU PUNKTE	**10** Otsus: **Kaaluda denormaliseerimist**

Suhe: Oppeaine-Oppeaine_seisundi_liik (*Oppeaine_seisundi_liik* on peremeestabel ja *Oppeaine* on sõltuv tabel)

Tabel 23. Denormaliseerimise algoritmi küsimustik suhte Oppeaine-Oppeaine_seisundi_liik alusel.

Küsimus	Valik
Mis tüüpi suhtega on tegemist peremees tabeli ja sõltuva tabeli vahel?	Definitsioon. Peremeestabelis olevad andmed määravad sõltuvas tabelis olevate andmete tähenduse. Nt. õppeaine spetsifikatsioon ja õppeaine tegelik õpetamine või klassifikaator ja selle poolt klassifitseeritavad andmed. Kui olemitüüpide vahel on M:N seosetüüp, siis luuakse selle esitamiseks vahetabel. Suhe selle vahetabeli ja peremeestabelite vahel on defineeriv suhe. **(-20)**
Milline on ühe peremeestabeli reaga seotud sõltuva tabeli ridade arv?	100 ja rohkem **(-20)**
Kui palju veerge on peremeestabelis, jättes arvestamata primaarvõtme veerud?	Vähem kui 10 **(20)**

Küsimus	Valik
Kui tehakse päring sõltuva tabeli kohta, siis kui sageli soovitakse andmeid ka peremeestabelist?	Päringus soovitakse sageli andmeid peremeestabelist **(30)** (Õppeaine nimekirjas soovitakse näha ka Õppeaine seisundi nimetust)
Kas peremeestabelisse on lähiajal plaanis lisada uusi veerge või siduda uusi tabeleid?	Ei **(20)**
Kas peremeestabelis ja sõltuvas tabelis toimub ajaperioodis sarnane arv lisamisi ja muutmisi?	Ei (ei toimu sarnane arv lisamisi ja muutmisi; klassifikaatorid registreeritakse juba enne andmete ülekandmist andmebaasi ning klassifikaatorite hulk suureneb väga aeglaselt, õppeainete andmeid aga lisatakse regulaarselt) **(-20)**
KOKKU PUNKTE	**10** Otsus: **Kaaluda denormaliseerimist**

Suhe: Oppeaine-Tootaja (*Tootaja* on peremeestabel ja *Oppeaine* on sõltuv tabel)

Tabel 24. Denormaliseerimise algoritmi küsimustik suhte Oppeaine-Tootaja alusel.

Küsimus	Valik
Mis tüüpi suhtega on tegemist peremees tabeli ja sõltuva tabeli vahel?	Definitsioon. Peremeestabelis olevad andmed määravad sõltuvas tabelis olevate andmete tähenduse. Nt. õppeaine spetsifikatsioon ja õppeaine tegelik õpetamine või klassifikaator ja selle poolt klassifitseeritavad andmed. Kui olemitüüpide vahel on M:N seosetüüp, siis luuakse selle esitamiseks vahetabel. Suhe selle vahetabeli ja peremeestabelite vahel on defineeriv suhe. **(-20)**
Milline on ühe peremeestabeli reaga seotud sõltuva tabeli ridade arv?	1 kuni 5 **(20)**
Kui palju veerge on peremeestabelis, jättes arvestamata primaarvõtme veerud?	10 kuni 20 **(-10)**
Kui tehakse päring sõltuva tabeli kohta, siis kui sageli soovitakse andmeid ka peremeestabelist?	Päringus soovitakse sageli andmeid peremeestabelist **(30)**
Kas peremeestabelisse on lähiajal plaanis lisada uusi veerge või	Ei **(20)**

Küsimus	Valik
siduda uusi tabeleid?	
Kas peremeestabelis ja sõltuvas tabelis toimub ajaperioodis sarnane arv lisamisi ja muutmisi?	Ei (ei toimu sarnane arv lisamisi ja muutmisi) **(-20)**
KOKKU PUNKTE	**40** Otsus: **Kaaluda denormaliseerimist**

Suhe: Tootaja_aines-Oppimine (*Tootaja_aines* on peremees tabel ja *Oppimine* on sõltuv tabel)

Tabel 25. Denormaliseerimise algoritmi küsimustik suhte Tootaja_aines-Oppimine alusel.

Küsimus	Valik
Mis tüüpi suhtega on tegemist peremees tabeli ja sõltuva tabeli vahel?	Definitsioon. Peremeestabelis olevad andmed määravad sõltuvas tabelis olevate andmete tähenduse. Nt. õppeaine spetsifikatsioon ja õppeaine tegelik õpetamine või klassifikaator ja selle poolt klassifitseeritavad andmed. Kui olemitüüpide vahel on M:N seosetüüp, siis luuakse selle esitamiseks vahetabel. Suhe selle vahetabeli ja peremeestabelite vahel on defineeriv suhe. **(-20)**
Milline on ühe peremeestabeli reaga seotud sõltuva tabeli ridade arv?	100 ja rohkem **(-20) (**Võetud arvesse informaatika teaduskonnas õpetatavaid ained, andmebaasid I ja II deklareerimistearv)
Kui palju veerge on peremeestabelis, jättes arvestamata primaarvõtme veerud?	Vähem kui 10 **(20)**
Kui tehakse päring sõltuva tabeli kohta, siis kui sageli soovitakse andmeid ka peremeestabelist?	Päringus soovitakse sageli andmeid peremeestabelist **(30)** (Õppeaine nimekirjas soovitakse näha Töötajat kes on seotud antud ainega)
Kas peremeestabelisse on lähiajal plaanis lisada uusi veerge või siduda uusi tabeleid?	Ei **(20)**
Kas peremeestabelis ja sõltuvas tabelis toimub ajaperioodis sarnane arv lisamisi ja muutmisi?	Ei (ei toimu sarnane arv lisamisi ja muutmisi) **(-20)**
KOKKU PUNKTE	**10**

Küsimus	Valik
	Otsus: **Kaaluda denormaliseerimist**

Suhe: Oppeaine-Tootaja_aines (*Oppeaine* on peremeestabel ja *Tootaja_aines* on sõltuv tabel)

Tabel 26. Denormaliseerimise algoritmi küsimustik suhte Oppeaine-Tootaja_aines alusel.

Küsimus	Valik
Mis tüüpi suhtega on tegemist peremees tabeli ja sõltuva tabeli vahel?	Definitsioon. Peremeestabelis olevad andmed määravad sõltuvas tabelis olevate andmete tähenduse. Nt. õppeaine spetsifikatsioon ja õppeaine tegelik õpetamine või klassifikaator ja selle poolt klassifitseeritavad andmed. Kui olemitüüpide vahel on M:N seosetüüp, siis luuakse selle esitamiseks vahetabel. Suhe selle vahetabeli ja peremeestabelite vahel on defineeriv suhe. **(-20)**
Milline on ühe peremeestabeli reaga seotud sõltuva tabeli ridade arv?	1 kuni 5 **(20)**
Kui palju veerge on peremeestabelis, jättes arvestamata primaarvõtme veerud?	10 kuni 20 **(-10)**
Kui tehakse päring sõltuva tabeli kohta, siis kui sageli soovitakse andmeid ka peremeestabelist?	Päringus soovitakse sageli andmeid peremeestabelist **(30)**
Kas peremeestabelisse on lähiajal plaanis lisada uusi veerge või siduda uusi tabeleid?	Ei **(20)**
Kas peremeestabelis ja sõltuvas tabelis toimub ajaperioodis sarnane arv lisamisi ja muutmisi?	Ei (ei toimu sarnane arv lisamisi ja muutmisi) **(-20)**
KOKKU PUNKTE	**20** Otsus: **Kaaluda denormaliseerimist**

Suhe: Tootaja_aines-Tootaja (*Tootaja* on peremeestabel ja *Tootaja_aines* on sõltuv tabel)

Tabel 27. Denormaliseerimise algoritmi küsimustik suhte Tootaja_aines-Tootaja alusel.

Küsimus	**Valik**
Mis tüüpi suhtega on tegemist peremees tabeli ja sõltuva tabeli vahel?	Osa-terviku (agregatsiooni või kompositsiooni) seos, hierarhia. Nt. tellimus sisaldab tellimuste ridu, kauba kategooria sisaldab alamkategooriaid. **(20)**
Milline on ühe peremeestabeli reaga seotud sõltuva tabeli ridade arv?	5 kuni 100 **(-10)**
Kui palju veerge on peremeestabelis, jättes arvestamata primaarvõtme veerud?	10 kuni 20 **(-10)**
Kui tehakse päring sõltuva tabeli kohta, siis kui sageli soovitakse andmeid ka peremeestabelist?	Päringus soovitakse sageli andmeid peremeestabelist **(30)**
Kas peremeestabelisse on lähiajal plaanis lisada uusi veerge või siduda uusi tabeleid?	Ei **(20)**
Kas peremeestabelis ja sõltuvas tabelis toimub ajaperioodis sarnane arv lisamisi ja muutmisi?	Ei (ei toimu sarnane arv lisamisi ja muutmisi) **(-20)**
KOKKU PUNKTE	**30** Otsus: **Kaaluda denormaliseerimist**

Suhe: Tootaja_aines-Roll_aines (*Roll_aines* on peremees tabel ja *Tootaja_aines* on sõltuv tabel)

Tabel 28. Denormaliseerimise algoritmi küsimustik suhte Tootaja_aines-Roll_aines alusel.

Küsimus	**Valik**
Mis tüüpi suhtega on tegemist peremees tabeli ja sõltuva tabeli vahel?	Definitsioon. Peremeestabelis olevad andmed määravad sõltuvas tabelis olevate andmete tähenduse. Nt. õppeaine spetsifikatsioon ja õppeaine tegelik õpetamine või klassifikaator ja selle poolt klassifitseeritavad andmed. Kui olemitüüpide vahel on M:N seosetüüp, siis luuakse selle

Küsimus	Valik
	esitamiseks vahetabel. Suhe selle vahetabeli ja peremeestabelite vahel on defineeriv suhe. **(-20)**
Milline on ühe peremeestabeli reaga seotud sõltuva tabeli ridade arv?	100 ja rohkem **(-20)**
Kui palju veerge on peremeestabelis, jättes arvestamata primaarvõtme veerud?	Vähem kui 10 **(20)**
Kui tehakse päring sõltuva tabeli kohta, siis kui sageli soovitakse andmeid ka peremeestabelist?	Päringus soovitakse sageli andmeid peremeestabelist**(30)**
Kas peremeestabelisse on lähiajal plaanis lisada uusi veerge või siduda uusi tabeleid?	Ei **(20)**
Kas peremeestabelis ja sõltuvas tabelis toimub ajaperioodis sarnane arv lisamisi ja muutmisi?	Ei (ei toimu sarnane arv lisamisi ja muutmisi) **(-20)**
KOKKU PUNKTE	**10** Otsus: **Kaaluda denormaliseerimist**

Suhe: Yliopilane-Oppimine (*Yliopilane* on peremees tabel ja *Oppimine* on sõltuv tabel)

Tabel 29. Denormaliseerimise algoritmi küsimustik suhte Yliopilane-Oppimine alusel.

Küsimus	Valik
Mis tüüpi suhtega on tegemist peremees tabeli ja sõltuva tabeli vahel?	Definitsioon. Peremeestabelis olevad andmed määravad sõltuvas tabelis olevate andmete tähenduse. Nt. õppeaine spetsifikatsioon ja õppeaine tegelik õpetamine või klassifikaator ja selle poolt klassifitseeritavad andmed. Kui olemitüüpide vahel on M:N seosetüüp, siis luuakse selle esitamiseks vahetabel. Suhe selle vahetabeli ja peremeestabelite vahel on defineeriv suhe. **(-20)**
Milline on ühe peremeestabeli reaga seotud sõltuva tabeli ridade arv?	5 kuni 100 **(-10)**
Kui palju veerge on peremeestabelis, jättes	Vähem kui 10 **(20)**

Küsimus	Valik
arvestamata primaarvõtme veerud?	
Kui tehakse päring sõltuva tabeli kohta, siis kui sageli soovitakse andmeid ka peremeestabelist?	Päringus soovitakse sageli andmeid peremeestabelist **(30)**
Kas peremeestabelisse on lähiajal plaanis lisada uusi veerge või siduda uusi tabeleid?	Jah **(-20)** (ülikoolis on plaanis jätkata uute registrite väljatöötamist ning olemasolevate täiustamist)
Kas peremeestabelis ja sõltuvas tabelis toimub ajaperioodis sarnane arv lisamisi ja muutmisi?	Ei (ei toimu sarnane arv lisamisi ja muutmisi) **(-20)**
KOKKU PUNKTE	**-20** Otsus: **Mitte denormaliseerida**

Lisa 5 – 5NK andmebaasi loomise skript

```
CREATE TABLE Amet (
    Amet_kood SMALLINT NOT NULL,
    Nimetus VARCHAR ( 50 ) NOT NULL,
    Kirjeldus VARCHAR ( 255 ),
    CONSTRAINT AK_Amet_nimetus UNIQUE (Nimetus),
    CONSTRAINT PK_Amet PRIMARY KEY (Amet_kood),
    CONSTRAINT chk_Amet_nimetus_ei_koosne_tyhikutest CHECK
(Nimetus!~'^[[:space:]]*$'),
    CONSTRAINT chk_Amet_kirjeldus_ei_koosne_tyhikutest CHECK
(Kirjeldus!~'^[[:space:]]*$')
);
CREATE TABLE Roll_Aines (
    Roll_Aines_kood SMALLINT NOT NULL,
    Nimetus VARCHAR ( 50 ) NOT NULL,
    Kirjeldus VARCHAR ( 255 ),
    CONSTRAINT AK_Roll_Aines_nimetus UNIQUE (Nimetus),
    CONSTRAINT PK_Roll_Aines PRIMARY KEY (Roll_Aines_kood),
    CONSTRAINT chk_Roll_aines_nimetus_ei_koosne_tyhikutest CHECK (
Nimetus!~'^[[:space:]]*$'),
    CONSTRAINT chk_Roll_aines_kirjeldus_ei_koosne_tyhikutest CHECK
(Kirjeldus!~'^[[:space:]]*$')
    );
CREATE TABLE Kontrollvorm (
    Kontrollvorm_kood CHAR ( 1 ) NOT NULL,
    Nimetus VARCHAR ( 50 ) NOT NULL,
    Kirjeldus VARCHAR ( 255 ),
    CONSTRAINT PK_Kontrollvorm PRIMARY KEY (Kontrollvorm_kood),
    CONSTRAINT AK_Kontrollvorm_nimetus UNIQUE (Nimetus),
    CONSTRAINT chk_Kontrollvorm_nimetus_ei_koosne_tyhikutest CHECK
(Nimetus!~'^[[:space:]]*$'),
    CONSTRAINT chk_Kontrollvorm_kood_koosneb_yhest_tahest CHECK
(Kontrollvorm_kood~'^[[:alpha:]]{1}$'),
    CONSTRAINT chk_Kontrollvorm_kirjeldus_ei_koosne_tyhikutest CHECK
(Kirjeldus!~'^[[:space:]]*$')
    );
CREATE TABLE Oppeaine_seisundi_liik (
    Oppeaine_seisundi_liik_kood SMALLINT NOT NULL,
    Nimetus VARCHAR ( 50 ) NOT NULL,
    Kirjeldus VARCHAR ( 255 ),
    CONSTRAINT AK_Oppeaine_seisundi_liik_nimetus UNIQUE (Nimetus),
    CONSTRAINT PK_Oppeaine_seisundi_liik PRIMARY KEY
(Oppeaine_seisundi_liik_kood),
    CONSTRAINT chk_Oppeaine_seisundii_liik_nimetus_ei_koosne_tyhikutest CHECK
(Nimetus!~'^[[:space:]]*$'),
    CONSTRAINT chk_Oppeaine_seisundi_liik_kirjeldus_ei_koosne_tyhikutest
CHECK (Kirjeldus!~'^[[:space:]]*$')
    );
CREATE TABLE Semester (
    Semester_kood VARCHAR ( 2 ) NOT NULL,
```

```
    Nimetus VARCHAR ( 50 ) NOT NULL,
    Kirjeldus VARCHAR ( 255 ),
    CONSTRAINT PK_Semester PRIMARY KEY (Semester_kood),
    CONSTRAINT AK_Semester_nimetus UNIQUE (Nimetus),
    CONSTRAINT chk_Semester_kirjeldus_ei_koosne_tyhikutest CHECK
(Kirjeldus!~'^[[:space:]]*$'),
    CONSTRAINT chk_Semester_nimetus_ei_koosne_tyhikutest CHECK
(Nimetus!~'^[[:space:]]*$'),
    CONSTRAINT chk_Semester_kood_koosneb_kuni_kahest_tahest CHECK
(Semester_kood~'^[[:alpha:]]{1,2}$')
    );
CREATE TABLE Tootaja (
    Isikukood CHAR ( 11 ) NOT NULL,
    Tootaja_seisundi_liik_kood SMALLINT DEFAULT 1 NOT NULL,
    Amet_kood SMALLINT NOT NULL,
    CONSTRAINT PK_Tootaja PRIMARY KEY (Isikukood)
    );
CREATE INDEX idx_tootaja_tootaja_seisundi_liik_kood ON Tootaja
(Tootaja_seisundi_liik_kood);
CREATE INDEX idx_tootaja_amet_kood ON Tootaja (Amet_kood);
CREATE TABLE Tootaja_aines (
    Tootaja_aines_kood SERIAL NOT NULL,
    Isikukood CHAR ( 11 ) NOT NULL,
    Roll_Aines_kood SMALLINT NOT NULL,
    Ainekood CHAR ( 7 ) NOT NULL,
    Alguse_aeg DATE NOT NULL,
    Lopu_aeg DATE,
    CONSTRAINT PK_Tootaja_aines PRIMARY KEY (Tootaja_aines_kood),
    CONSTRAINT AK_Tootaja_aines_kombinatsioon UNIQUE (Ainekood, Alguse_aeg,
Roll_Aines_kood, Isikukood),
    CONSTRAINT chk_Tootaja_aines_alguse_aeg_ylempiir CHECK
(Alguse_aeg<='2118-09-17'),
    CONSTRAINT chk_Tootaja_aines_lopu_aeg_ylempiir CHECK (Lopu_aeg<='2118-09-
17'),
    CONSTRAINT chk_Tootaja_aines_ainekood_kolm_tahti_neli_numbrit CHECK
(Ainekood~'^([[:alpha:]]{3}[[:digit:]]{4})$'),
        CONSTRAINT
chk_Tootaja_aines_alguse_aeg_ei_saa_olla_varem_kui_ttu_loomine CHECK
(Alguse_aeg>='1918-09-17')
    );
CREATE INDEX idx_tootaja_aines_roll_aines_kood ON Tootaja_aines
(Roll_Aines_kood);
CREATE INDEX idx_tootaja_aines_isikukood ON Tootaja_aines (Isikukood);
CREATE TABLE Yliopilase_seisundi_liik (
    Yliopilase_seisundi_liik_kood SMALLINT NOT NULL,
    Nimetus VARCHAR ( 50 ) NOT NULL,
    Kirjeldus VARCHAR ( 255 ),
    CONSTRAINT PK_Yliopilase_seisundi_liik PRIMARY KEY
(Yliopilase_seisundi_liik_kood),
    CONSTRAINT AK_Yliopilase_seisundi_liik_nimetus UNIQUE (Nimetus),
    CONSTRAINT chk_Yliopilase_seisundi_liik_nimetus_ei_koosne_tyhikutest
CHECK ( Nimetus!~'^[[:space:]]*$'),
```

```
    CONSTRAINT chk_Yliopilase_seisundii_liik_kirjeldus_ei_koosne_tyhikutest
CHECK (Kirjeldus!~'^[[:space:]]*$')
    );
CREATE TABLE Isiku_seisundi_liik (
    Isiku_seisundi_liik_kood SMALLINT NOT NULL,
    Nimetus VARCHAR ( 50 ) NOT NULL,
    Kirjeldus VARCHAR ( 255 ),
    CONSTRAINT PK_Isiku_seisundi_liik PRIMARY KEY (Isiku_seisundi_liik_kood),
    CONSTRAINT AK_Isiku_seisundi_liik_nimetus UNIQUE (Nimetus),
    CONSTRAINT chk_Isiku_seisundi_liik_kirjeldus_ei_koosne_tyhikutest CHECK
(Kirjeldus!~'^[[:space:]]*$'),
    CONSTRAINT chk_Isiku_seisundi_liik_nimetus_ei_koosne_tyhikutest CHECK
(Nimetus!~'^[[:space:]]*$')
    );
CREATE TABLE Keel (
    Keel_kood CHAR ( 3 ) NOT NULL,
    Nimetus VARCHAR ( 50 ) NOT NULL,
    Kirjeldus VARCHAR ( 255 ),
    CONSTRAINT AK_Keel_nimetus UNIQUE (Nimetus),
    CONSTRAINT PK_Keel PRIMARY KEY (Keel_kood),
    CONSTRAINT chk_Keel_kood_vaikesed_tahed CHECK (Keel_kood =
lower(Keel_kood)),
    CONSTRAINT chk_Keel_nimetus_ei_koosne_tyhikutest CHECK
(Nimetus!~'^[[:space:]]*$'),
    CONSTRAINT chk_Keel_kood_koosneb_kolmest_tahest CHECK
(Keel_kood~'^[[:alpha:]]{3}$'),
    CONSTRAINT chk_Keel_kirjeldus_ei_koosne_tyhikutest CHECK
(Kirjeldus!~'^[[:space:]]*$')
    );
CREATE TABLE Tootaja_seisundi_liik (
    Tootaja_seisundi_liik_kood SMALLINT NOT NULL,
    Nimetus VARCHAR ( 50 ) NOT NULL,
    Kirjeldus VARCHAR ( 255 ),
    CONSTRAINT PK_Tootaja_seisundi_liik PRIMARY KEY
(Tootaja_seisundi_liik_kood),
    CONSTRAINT AK_Tootaja_seisundi_liik_nimetus UNIQUE (Nimetus),
    CONSTRAINT chk_Tootaja_seisundi_liik_kirjeldus_ei_koosne_tyhikutest CHECK
(Kirjeldus!~'^[[:space:]]*$'),
    CONSTRAINT chk_Tootaja_seisundi_liik_nimetus_ei_koosne_tyhikutest CHECK
(Nimetus!~'^[[:space:]]*$')
    );
CREATE TABLE Oppeaine (
    Ainekood CHAR ( 7 ) NOT NULL,
    Kontrollvorm_kood CHAR ( 1 ) NOT NULL,
    Semester_kood VARCHAR ( 2 ) NOT NULL,
    Keel_kood CHAR ( 3 ) NOT NULL,
    Isikukood CHAR ( 11 ) NOT NULL,
    Oppeaine_seisundi_liik_kood SMALLINT DEFAULT 1 NOT NULL,
    Nimetus VARCHAR ( 50 ) NOT NULL,
    Punktid SMALLINT NOT NULL,
    Sisu VARCHAR ( 1000 ) NOT NULL,
    Loomise_aeg DATE DEFAULT CURRENT_DATE NOT NULL,
```

```
    Koduleht VARCHAR ( 700 ),
    Kinnitamise_aeg DATE,
    CONSTRAINT PK_Oppeaine PRIMARY KEY (Ainekood),
    CONSTRAINT chk_Oppeaine_punktid_vaiksem_200_ CHECK (Punktid<=200),
    CONSTRAINT chk_Oppeaine_nimetus_ei_koosne_tyhikutest CHECK
(Nimetus!~'^[[:space:]]*$'),
    CONSTRAINT chk_Oppeaine_loomise_aeg_alampiir CHECK (Loomise_aeg>='1918-
09-17'),
    CONSTRAINT chk_Oppeaine_kinnitamise_aeg_ylempiir CHECK
(Kinnitamise_aeg<='2118-09-17'),
    CONSTRAINT chk_Oppeaine_sisu_ei_koosne_tyhikutest CHECK
(Sisu!~'^[[:space:]]*$'),
    CONSTRAINT chk_Oppeaine_loomise_aeg_ylempiir CHECK (Loomise_aeg<='2118-
09-17'),
    CONSTRAINT chk_Oppeaine_ainekood_kolm_tahti_neli_numbrit CHECK
(Ainekood~'^([[:alpha:]]{3}[[:digit:]]{4})$'),
    CONSTRAINT chk_Oppeaine_punktid_suurem_0 CHECK (Punktid>=0)
    );
CREATE INDEX idx_oppeaine_semester_kood ON Oppeaine (Semester_kood );
CREATE INDEX idx_oppeaine_seisundi_liik_kood ON Oppeaine
(Oppeaine_seisundi_liik_kood );
CREATE INDEX idx_oppeaine_isikukood ON Oppeaine (Isikukood );
CREATE INDEX idx_oppeaine_kontrollvorm_kood ON Oppeaine (Kontrollvorm_kood );
CREATE INDEX idx_oppeaine_keel_kood ON Oppeaine (Keel_kood );
CREATE TABLE Yliopilane (
    Isikukood CHAR ( 11 ) NOT NULL,
    Yliopilase_seisundi_liik_kood SMALLINT DEFAULT 1 NOT NULL,
    Matrikli_nr CHAR ( 6 ) NOT NULL,
    CONSTRAINT AK_Yliopilane_matrikli_nr UNIQUE (Matrikli_nr),
    CONSTRAINT PK_Yliopilane PRIMARY KEY (Isikukood),
    CONSTRAINT chk_Yliopilane_matrikli_nr_kuus_numbrit CHECK
(Matrikli_nr~'^([[:digit:]]{6})$')
    );
CREATE INDEX idx_yliopilane_yliopilase_seisundi_liik_kood ON Yliopilane
(Yliopilase_seisundi_liik_kood );
CREATE TABLE Isik (
    Isikukood CHAR ( 11 ) NOT NULL,
    Isiku_seisundi_liik_kood SMALLINT DEFAULT 1 NOT NULL,
    Eesnimi VARCHAR ( 60 ) NOT NULL,
    Perekonnanimi VARCHAR ( 100 ) NOT NULL,
    Email VARCHAR ( 320 ) NOT NULL,
    Telefon VARCHAR ( 31 ) NOT NULL,
    Kasutajanimi VARCHAR ( 20 ) NOT NULL,
    Parool VARCHAR ( 60 ) NOT NULL,
    CONSTRAINT PK_Isik PRIMARY KEY (Isikukood),
    CONSTRAINT AK_Isik_kasutajanimi UNIQUE (Kasutajanimi),
    CONSTRAINT Ak_Isik_email UNIQUE (Email),
    CONSTRAINT chk_Isik_telefoni_nr_ei_sisalda_tahti CHECK
(Telefon!~'^.*[[:alpha:]].*$'),
    CONSTRAINT chk_Isik_isikukood CHECK (Isikukood~'^([3-
6]{1}[[:digit:]]{2}[0-1]{1}[[:digit:]]{1}[0-3]{1}[[:digit:]]{5})$'),
```

```
    CONSTRAINT chk_Isik_kasutajanimi_5_kuni_20_marki CHECK
(Kasutajanimi~'^([[:alpha:]]{5,20})$'),
    CONSTRAINT chk_Isik_email_vastab_oigele_kujule CHECK (Email~* '^[A-Za-z0-
9._%-]+@[A-Za-z0-9.-]+[.][A-Za-z]+$'),
    CONSTRAINT chk_Isik_perekonnanimi_ei_sisalda_numbreid CHECK
(Perekonnanimi!~'^.*[[:digit:]].*$'),
    CONSTRAINT chk_Isik_telefoni_nr_ei_koosne_tyhikutest CHECK
(Telefon!~'^[[:space:]]*$'),
    CONSTRAINT chk_Isik_perekonnanimi_ei_koosne_tyhikutest CHECK
(Perekonnanimi!~'^[[:space:]]*$'),
    CONSTRAINT chk_Isik_telefoni_nr_sisaldab_nr_pluss_thk CHECK
(Telefon~'^[+]{1}[[:digit:]]{3}[[:space:]]{1}[[:digit:]]{6,8}$'),
    CONSTRAINT chk_Isik_eesnimi_ei_koosne_tyhikutest CHECK
(Eesnimi!~'^[[:space:]]*$'),
    CONSTRAINT chk_Isik_eesnimi_ei_sisalda_numbreid CHECK
(Eesnimi!~'^.*[[:digit:]].*$')
    );
CREATE INDEX idx_Isik_seisundi_liik_kood ON Isik (Isiku_seisundi_liik_kood );
CREATE TABLE Oppimine (
    Oppimine_kood SERIAL NOT NULL,
    Isikukood CHAR ( 11 ) NOT NULL,
    Tootaja_aines_kood SMALLINT NOT NULL,
    Deklareerimise_aeg DATE NOT NULL,
    CONSTRAINT AK_Oppimine_kombinatsioon UNIQUE (Deklareerimise_aeg,
Isikukood, Tootaja_aines_kood),
    CONSTRAINT PK_Oppimine PRIMARY KEY (Oppimine_kood),
    CONSTRAINT
chk_Oppimine_deklareerimise_aeg_ei_saa_olla_varem_kui_ttu_loomine CHECK
(Deklareerimise_aeg>='1918-09-17')
    );
CREATE INDEX idx_oppimine_isikukood ON Oppimine (Isikukood);
CREATE INDEX idx_oppimine_tootaja_aines_kood ON Oppimine
(Tootaja_aines_kood);
ALTER TABLE Isik ADD CONSTRAINT FK_Isik_seisundi_liik_kood FOREIGN KEY
(Isiku_seisundi_liik_kood) REFERENCES Isiku_seisundi_liik
(Isiku_seisundi_liik_kood)  ON DELETE NO ACTION ON UPDATE CASCADE;
ALTER TABLE Tootaja_aines ADD CONSTRAINT FK_Tootaja_aines_ainekood FOREIGN
KEY (Ainekood) REFERENCES Oppeaine (Ainekood)  ON DELETE NO ACTION ON UPDATE
NO ACTION;
ALTER TABLE Tootaja_aines ADD CONSTRAINT FK_Tootaja_aines_isikukood FOREIGN
KEY (Isikukood) REFERENCES Tootaja (Isikukood)  ON DELETE NO ACTION ON UPDATE
CASCADE;
ALTER TABLE Tootaja_aines ADD CONSTRAINT FK_Tootaja_aines_roll_aines_kood
FOREIGN KEY (Roll_Aines_kood) REFERENCES Roll_Aines (Roll_Aines_kood)  ON
DELETE NO ACTION ON UPDATE CASCADE;
ALTER TABLE Tootaja ADD CONSTRAINT FK_Tootaja_isikukood FOREIGN KEY
(Isikukood) REFERENCES Isik (Isikukood)  ON DELETE CASCADE ON UPDATE CASCADE;
ALTER TABLE Tootaja ADD CONSTRAINT FK_Tootaja_seisundi_liik_kood FOREIGN KEY
(Tootaja_seisundi_liik_kood) REFERENCES Tootaja_seisundi_liik
(Tootaja_seisundi_liik_kood)  ON DELETE NO ACTION ON UPDATE CASCADE;
ALTER TABLE Tootaja ADD CONSTRAINT FK_Tootaja_amet_kood FOREIGN KEY
(Amet_kood) REFERENCES Amet (Amet_kood)  ON DELETE NO ACTION ON UPDATE
CASCADE;
ALTER TABLE Yliopilane ADD CONSTRAINT FK_Yliopilane_isikukood FOREIGN KEY
(Isikukood) REFERENCES Isik (Isikukood)  ON DELETE CASCADE ON UPDATE CASCADE;
```

```
ALTER TABLE Yliopilane ADD CONSTRAINT FK_Yliopilane_seisundi_liik_kood
FOREIGN KEY (Yliopilase_seisundi_liik_kood) REFERENCES
Yliopilase_seisundi_liik (Yliopilase_seisundi_liik_kood)  ON DELETE NO ACTION
ON UPDATE CASCADE;
ALTER TABLE Oppimine ADD CONSTRAINT FK_Oppimine_tootaja_aines_kood FOREIGN
KEY (Tootaja_aines_kood) REFERENCES Tootaja_aines (Tootaja_aines_kood)  ON
DELETE NO ACTION ON UPDATE CASCADE;
ALTER TABLE Oppimine ADD CONSTRAINT FK_Oppimine_isikukood FOREIGN KEY
(Isikukood) REFERENCES Yliopilane (Isikukood)  ON DELETE NO ACTION ON UPDATE
CASCADE;
ALTER TABLE Oppeaine ADD CONSTRAINT FK_Oppeaine_kontrollvorm_kood FOREIGN KEY
(Kontrollvorm_kood) REFERENCES Kontrollvorm (Kontrollvorm_kood)  ON DELETE NO
ACTION ON UPDATE CASCADE;
ALTER TABLE Oppeaine ADD CONSTRAINT FK_Oppeaine_seisundi_liik_kood FOREIGN
KEY (Oppeaine_seisundi_liik_kood) REFERENCES Oppeaine_seisundi_liik
(Oppeaine_seisundi_liik_kood)  ON DELETE NO ACTION ON UPDATE NO ACTION;
ALTER TABLE Oppeaine ADD CONSTRAINT FK_Oppeaine_isikukood FOREIGN KEY
(Isikukood) REFERENCES Tootaja (Isikukood)  ON DELETE NO ACTION ON UPDATE
CASCADE;
ALTER TABLE Oppeaine ADD CONSTRAINT FK_Oppeaine_semester_kood FOREIGN KEY
(Semester_kood) REFERENCES Semester (Semester_kood)  ON DELETE NO ACTION ON
UPDATE CASCADE;
ALTER TABLE Oppeaine ADD CONSTRAINT FK_Oppeaine_keel_kood FOREIGN KEY
(Keel_kood) REFERENCES Keel (Keel_kood)  ON DELETE NO ACTION ON UPDATE
CASCADE;
```

Lisa 6 – Denormaliseeritud andmebaasi loomise skript

```
CREATE DOMAIN d_eesnimi VARCHAR(60) NOT NULL CONSTRAINT
chk_eesnimi_tyhistring_ja_ei_koosne_numbritest CHECK (VALUE!~'^[[:space:]]*$'
AND VALUE!~'^.*[[:digit:]].*$');
CREATE DOMAIN d_perekonnanimi VARCHAR(100) NOT NULL CONSTRAINT
chk_perekonnanimi_tyhistring_ja_ei_koosne_numbritest CHECK
(VALUE!~'^[[:space:]]*$' AND VALUE!~'^.*[[:digit:]].*$');
CREATE DOMAIN d_nimetus VARCHAR(50) NOT NULL CONSTRAINT
chk_nimetus_tyhistring CHECK (VALUE!~'^[[:space:]]*$');
CREATE DOMAIN d_sisu VARCHAR(1000) NOT NULL CONSTRAINT chk_sisu_tyhistring
CHECK (VALUE!~'^[[:space:]]*$');
CREATE DOMAIN d_kirjeldus VARCHAR(255) CONSTRAINT chk_kirjeldus_tyhistring
CHECK (VALUE!~'^[[:space:]]*$');
CREATE TABLE Tootaja_aines (
    Tootaja_aines_kood SERIAL NOT NULL,
    Isikukood CHAR ( 11 ) NOT NULL,
    Ainekood CHAR ( 7 ) NOT NULL,
    Roll_Aines_kood SMALLINT NOT NULL,
    Roll_Aines_nimetus d_nimetus,
    Alguse_aeg DATE NOT NULL,
    Lopu_aeg DATE,
    Eesnimi d_eesnimi,
    Perekonnanimi d_perekonnanimi,
    Email VARCHAR ( 320 ) NOT NULL,
    Nimetus d_nimetus,
    Punktid SMALLINT NOT NULL,
    Sisu d_sisu,
    CONSTRAINT PK_Tootaja_aines PRIMARY KEY (Tootaja_aines_kood),
    CONSTRAINT AK_Tootaja_aines_kombinatsioon UNIQUE (Ainekood, Alguse_aeg,
Roll_Aines_kood, Isikukood),
    CONSTRAINT chk_Tootaja_aines_email_vastab_oigele_kujule CHECK (Email~*
'^[A-Za-z0-9._%-]+@[A-Za-z0-9.-]+[.][A-Za-z]+$'),
    CONSTRAINT chk_Tootaja_aines_lopu_aeg_ylempiir CHECK (Lopu_aeg<='2118-09-
17'),
    CONSTRAINT chk_Tootaja_aines_oppeaine_punktid_suurem_0 CHECK
(Punktid>=0),
    CONSTRAINT chk_Tootaja_aines_oppeaine_punktid_vaiksem_200 CHECK
(Punktid<=200),
    CONSTRAINT chk_Tootaja_aines_alguse_aeg_ei_saa_olla_varem_kui_ttu_loomine
CHECK (Alguse_aeg>='1918-09-17'),
    CONSTRAINT chk_Tootaja_aines_alguse_aeg_ylempiir CHECK
(Alguse_aeg<='2118-09-17')
    );
CREATE INDEX idx_tootaja_aines_roll_aines_kood ON Tootaja_aines
(Roll_Aines_kood );
CREATE INDEX idx_tootaja_aines_isikukood ON Tootaja_aines (Isikukood );
CREATE TABLE Amet (
    Amet_kood SMALLINT NOT NULL,
    Nimetus d_nimetus,
    Kirjeldus d_kirjeldus,
    CONSTRAINT AK_Amet_nimetus UNIQUE (Nimetus),
```

```
    CONSTRAINT PK_Amet PRIMARY KEY (Amet_kood)
    );
CREATE TABLE Yliopilase_seisundi_liik (
    Yliopilase_seisundi_liik_kood SMALLINT NOT NULL,
    Nimetus d_nimetus,
    Kirjeldus d_kirjeldus,
    CONSTRAINT PK_Yliopilase_seisundi_liik PRIMARY KEY
(Yliopilase_seisundi_liik_kood),
    CONSTRAINT AK_Yliopilase_seisundi_liik_nimetus UNIQUE (Nimetus)
    );
CREATE TABLE Isik (
    Isikukood CHAR ( 11 ) NOT NULL,
    Isiku_seisundi_liik_kood SMALLINT DEFAULT 1 NOT NULL,
    Isiku_seisundi_liik_nimetus d_nimetus,
    Tootaja_seisundi_liik_kood SMALLINT DEFAULT 1,
    Tootaja_seisundi_liik_nimetus VARCHAR ( 50 ),
    Yliopilase_seisundi_liik_kood SMALLINT DEFAULT 1,
    Yliopilase_seisundi_liik_nimetus VARCHAR ( 50 ),
    Amet_kood SMALLINT,
    Amet_nimetus VARCHAR ( 50 ),
    Eesnimi d_eesnimi,
    Perekonnanimi d_perekonnanimi,
    Email VARCHAR ( 320 ) NOT NULL,
    Telefon VARCHAR ( 31 ) NOT NULL,
    Kasutajanimi VARCHAR ( 20 ) NOT NULL,
    Parool VARCHAR ( 60 ) NOT NULL,
    Matrikli_nr CHAR ( 6 ),
    CONSTRAINT PK_Isik PRIMARY KEY (Isikukood),
    CONSTRAINT AK_Isik_kasutajanimi UNIQUE (Kasutajanimi),
    CONSTRAINT AK_Isik_email UNIQUE (Email),
    CONSTRAINT AK_Isik_matrikli_nr UNIQUE (Matrikli_nr),
    CONSTRAINT chk_Isik_telefoni_nr_ei_sisalda_tahti CHECK
(Telefon!~'^.*[[:alpha:]].*$'),
    CONSTRAINT chk_Isik_isikukood CHECK (Isikukood~'^([3-
6]{1}[[:digit:]]{2}[0-1]{1}[[:digit:]]{1}[0-3]{1}[[:digit:]]{5})$'),
    CONSTRAINT chk_Isik_kasutajanimi_5_kuni_20_marki CHECK
(Kasutajanimi~'^([[:alpha:]]{5,20})$'),
    CONSTRAINT chk_Isik_email_vastab_oigele_kujule CHECK (Email~* '^[A-Za-z0-
9._%-]+@[A-Za-z0-9.-]+[.][A-Za-z]+$'),
    CONSTRAINT chk_Isik_telefoni_nr_ei_koosne_tyhikutest CHECK
(Telefon!~'^[[:space:]]*$'),
    CONSTRAINT chk_Isik_telefoni_nr_sisaldab_nr_pluss_thk CHECK
(Telefon~'^[+]{1}[[:digit:]]{3}[[:space:]]{1}[[:digit:]]{6,8}$'),
    CONSTRAINT chk_Isik_yliopilane_matrikli_nr_kuus_numbrit CHECK
(Matrikli_nr~'^([[:digit:]]{6})$'),
    CONSTRAINT chk_Isik_tootaja_seisundi_liik_nimetus_ei_koosne_tyhikutest
CHECK (Tootaja_seisundi_liik_nimetus!~'^[[:space:]]*$'),
    CONSTRAINT chk_Isik_yliopilase_seisundi_liik_nimetus_ei_koosne_tyhikutest
CHECK (Yliopilase_seisundi_liik_nimetus!~'^[[:space:]]*$'),
    CONSTRAINT chk_Isik_amet_nimetus_ei_koosne_tyhikutest CHECK
(Amet_nimetus!~'^[[:space:]]*$')
    );
```

```
CREATE INDEX idx_isik_yliopilase_seisundi_liik_kood ON Isik
(Yliopilase_seisundi_liik_kood );
CREATE INDEX idx_isik_tootaja_seisundi_liik_kood ON Isik
(Tootaja_seisundi_liik_kood );
CREATE INDEX idx_isiku_seisudi_liik_kood ON Isik (Isiku_seisundi_liik_kood );
CREATE INDEX idx_isik_amet_kood ON Isik (Amet_kood );
CREATE TABLE Keel (
    Keel_kood CHAR ( 3 ) NOT NULL,
    Nimetus d_nimetus,
    Kirjeldus d_kirjeldus,
    CONSTRAINT AK_Keel_nimetus UNIQUE (Nimetus),
    CONSTRAINT PK_Keel PRIMARY KEY (Keel_kood),
    CONSTRAINT chk_Keel_kood_vaikesed_tahed CHECK (Keel_kood =
lower(Keel_kood)),
    CONSTRAINT chk_Keel_kood_koosneb_kolmest_tahest CHECK
(Keel_kood~'^[[:alpha:]]{3}$')
    );
CREATE TABLE Tootaja_seisundi_liik (
    Tootaja_seisundi_liik_kood SMALLINT NOT NULL,
    Nimetus d_nimetus,
    Kirjeldus d_kirjeldus,
    CONSTRAINT PK_Tootaja_seisundi_liik PRIMARY KEY
(Tootaja_seisundi_liik_kood),
    CONSTRAINT AK_Tootaja_seisundi_liik_nimetus UNIQUE (Nimetus)
    );
CREATE TABLE Semester (
    Semester_kood VARCHAR ( 2 ) NOT NULL,
    Nimetus d_nimetus,
    Kirjeldus d_kirjeldus,
    CONSTRAINT PK_Semester PRIMARY KEY (Semester_kood),
    CONSTRAINT AK_Semester_nimetus UNIQUE (Nimetus),
    CONSTRAINT chk_Semester_kood_koosneb_kuni_kahest_tahest CHECK
(Semester_kood~'^[[:alpha:]]{1,2}$')
    );
CREATE TABLE Kontrollvorm (
    Kontrollvorm_kood CHAR ( 1 ) NOT NULL,
    Nimetus d_nimetus,
    Kirjeldus d_kirjeldus,
    CONSTRAINT PK_Kontrollvorm PRIMARY KEY (Kontrollvorm_kood),
    CONSTRAINT AK_Kontollvorm_nimetus UNIQUE (Nimetus),
    CONSTRAINT chk_Kontrollvorm_kood_koosneb_yhest_tahest CHECK
(Kontrollvorm_kood~'^[[:alpha:]]{1}$')
    );
CREATE TABLE Oppeaine_seisundi_liik (
    Oppeaine_seisundi_liik_kood SMALLINT NOT NULL,
    Nimetus d_nimetus,
    Kirjeldus d_kirjeldus,
    CONSTRAINT AK_Oppeaine_seisundi_liik_nimetus UNIQUE (Nimetus),
    CONSTRAINT PK_Oppeaine_seisundi_liik PRIMARY KEY
(Oppeaine_seisundi_liik_kood)
    );
CREATE TABLE Roll_aines (
```

```
    Roll_Aines_kood SMALLINT NOT NULL,
    Nimetus d_nimetus,
    Kirjeldus d_kirjeldus,
    CONSTRAINT AK_Roll_Aines_nimetus UNIQUE (Nimetus),
    CONSTRAINT PK_Roll_Aines PRIMARY KEY (Roll_Aines_kood)
    );
CREATE TABLE Isiku_seisundi_liik (
    Isiku_seisundi_liik_kood SMALLINT NOT NULL,
    Nimetus d_nimetus,
    Kirjeldus d_kirjeldus,
    CONSTRAINT PK_Isiku_seisundi_liik PRIMARY KEY (Isiku_seisundi_liik_kood),
    CONSTRAINT AK_Isiku_seisundi_liik_nimetus UNIQUE (Nimetus)
    );
CREATE TABLE Oppeaine (
    Ainekood CHAR ( 7 ) NOT NULL,
    Isikukood CHAR ( 11 ) NOT NULL,
    Kontrollvorm_kood CHAR ( 1 ) NOT NULL,
    Kontrollvorm_nimetus d_nimetus,
    Semester_kood VARCHAR ( 2 ) NOT NULL,
    Semester_nimetus d_nimetus,
    Keel_kood CHAR ( 3 ) NOT NULL,
    Keel_nimetus d_nimetus,
    Oppeaine_seisundi_liik_kood SMALLINT DEFAULT 1 NOT NULL,
    Oppeaine_seisundi_liik_nimetus d_nimetus,
    Nimetus d_nimetus,
    Punktid SMALLINT NOT NULL,
    Sisu d_sisu,
    Loomise_aeg DATE DEFAULT CURRENT_DATE NOT NULL,
    Koduleht VARCHAR ( 700 ),
    Kinnitamise_aeg DATE,
    Eesnimi d_eesnimi,
    Perekonnanimi d_perekonnanimi,
    Email VARCHAR ( 320 ) NOT NULL,
    CONSTRAINT PK_Oppeaine PRIMARY KEY (Ainekood),
    CONSTRAINT chk_Oppeaine_punktid_vaiksem_200 CHECK (Punktid<=200),
    CONSTRAINT chk_Oppeaine_loomise_aeg_alampiir CHECK (Loomise_aeg>='1918-
09-17'),
    CONSTRAINT chk_Oppeaine_email_vastab_oigele_kujule CHECK (Email~* '^[A-
Za-z0-9._%-]+@[A-Za-z0-9.-]+[.][A-Za-z]+$'),
    CONSTRAINT chk_Oppeaine_kinnitamise_aeg_ylempiir CHECK
(Kinnitamise_aeg<='2118-09-17'),
    CONSTRAINT chk_Oppeaine_loomise_aeg_ylempiir CHECK (Loomise_aeg<='2118-
09-17'),
    CONSTRAINT chk_Oppeaine_ainekood_kolm_tahti_neli_numbrit CHECK
(Ainekood~'^([[:alpha:]]{3}[[:digit:]]{4})$'),
    CONSTRAINT chk_Oppeaine_punktid_suurem_0 CHECK (Punktid>=0)
    );
CREATE INDEX idx_oppeaine_semester_kood ON Oppeaine (Semester_kood );
CREATE INDEX idx_oppeaine_seisundi_liik_kood ON Oppeaine
(Oppeaine_seisundi_liik_kood );
CREATE INDEX idx_oppeaine_isikukood ON Oppeaine (Isikukood );
CREATE INDEX idx_oppeaine_kontrollvorm_kood ON Oppeaine (Kontrollvorm_kood );
```

```
CREATE INDEX idx_oppeaine_keel_kood ON Oppeaine (Keel_kood );
CREATE TABLE Oppimine (
    Oppimine_kood SERIAL NOT NULL,
    Isikukood CHAR ( 11 ) NOT NULL,
    Tootaja_aines_kood SMALLINT NOT NULL,
    Deklareerimise_aeg DATE NOT NULL,
    Ainekood CHAR ( 7 ) NOT NULL,
    Roll_aines_kood SMALLINT NOT NULL,
    Alguse_aeg DATE NOT NULL,
    CONSTRAINT AK_Oppimine_kombinatsioon UNIQUE (Deklareerimise_aeg,
Isikukood, Tootaja_aines_kood, Ainekood),
    CONSTRAINT PK_Oppimine PRIMARY KEY (Oppimine_kood),
    CONSTRAINT chk_Oppimine_alguse_aeg_ei_saa_olla_varem_kui_ttu_loomine
CHECK (Alguse_aeg>='1918-09-17'),
    CONSTRAINT
chk_Oppimine_deklareerimise_aeg_ei_saa_olla_varem_kui_ttu_loomine CHECK
(Deklareerimise_aeg>='1918-09-17'),
    CONSTRAINT chk_Oppimine_alguse_aeg_ylempiir CHECK (Alguse_aeg<='2118-09-
17'),
    CONSTRAINT chk_Oppimine_ainekood_kolm_tahti_neli_numbrit CHECK
(Ainekood~'^([[:alpha:]]{3}[[:digit:]]{4})$')
    );
CREATE INDEX idx_oppimine_tootaja_aines_kood ON Oppimine (Tootaja_aines_kood
);
CREATE INDEX idx_oppimine_isikukood ON Oppimine (Isikukood );
ALTER TABLE Isik ADD CONSTRAINT FK_Isik_yliopilase_seisundi_liik_kood FOREIGN
KEY (Yliopilase_seisundi_liik_kood) REFERENCES Yliopilase_seisundi_liik
(Yliopilase_seisundi_liik_kood)  ON DELETE NO ACTION ON UPDATE CASCADE;
ALTER TABLE Isik ADD CONSTRAINT FK_Isik_tootaja_seisundi_liik_kood FOREIGN
KEY (Tootaja_seisundi_liik_kood) REFERENCES Tootaja_seisundi_liik
(Tootaja_seisundi_liik_kood)  ON DELETE NO ACTION ON UPDATE CASCADE;
ALTER TABLE Isik ADD CONSTRAINT FK_Isik_amet_kood FOREIGN KEY (Amet_kood)
REFERENCES Amet (Amet_kood)  ON DELETE NO ACTION ON UPDATE CASCADE;
ALTER TABLE Isik ADD CONSTRAINT FK_Isiku_seisundi_liik_kood FOREIGN KEY
(Isiku_seisundi_liik_kood) REFERENCES Isiku_seisundi_liik
(Isiku_seisundi_liik_kood)  ON DELETE NO ACTION ON UPDATE CASCADE;
ALTER TABLE Tootaja_aines ADD CONSTRAINT FK_Tootaja_aines_ainekood FOREIGN
KEY (Ainekood) REFERENCES Oppeaine (Ainekood)  ON DELETE NO ACTION ON UPDATE
CASCADE;
ALTER TABLE Tootaja_aines ADD CONSTRAINT FK_Tootaja_aines_isikukood FOREIGN
KEY (Isikukood) REFERENCES Isik (Isikukood)  ON DELETE NO ACTION ON UPDATE
CASCADE;
ALTER TABLE Tootaja_aines ADD CONSTRAINT FK_Tootaja_aines_roll_aines_kood
FOREIGN KEY (Roll_Aines_kood) REFERENCES Roll_aines (Roll_Aines_kood)  ON
DELETE NO ACTION ON UPDATE CASCADE;
ALTER TABLE Oppimine ADD CONSTRAINT FK_Oppimine_tootaja_aines_kood FOREIGN
KEY (Tootaja_aines_kood) REFERENCES Tootaja_aines (Tootaja_aines_kood)  ON
DELETE NO ACTION ON UPDATE CASCADE;
ALTER TABLE Oppimine ADD CONSTRAINT FK_Oppimine_isikukood FOREIGN KEY
(Isikukood) REFERENCES Isik (Isikukood)  ON DELETE NO ACTION ON UPDATE
CASCADE;
ALTER TABLE Oppeaine ADD CONSTRAINT FK_Oppeaine_kontrollvorm_kood FOREIGN KEY
(Kontrollvorm_kood) REFERENCES Kontrollvorm (Kontrollvorm_kood)  ON DELETE NO
ACTION ON UPDATE CASCADE;
```

```
ALTER TABLE Oppeaine ADD CONSTRAINT FK_Oppeaine_seisundi_liik_kood FOREIGN
KEY (Oppeaine_seisundi_liik_kood) REFERENCES Oppeaine_seisundi_liik
(Oppeaine_seisundi_liik_kood)  ON DELETE NO ACTION ON UPDATE CASCADE;
ALTER TABLE Oppeaine ADD CONSTRAINT FK_Oppeaine_isikukood FOREIGN KEY
(Isikukood) REFERENCES Isik (Isikukood)  ON DELETE NO ACTION ON UPDATE
CASCADE;
ALTER TABLE Oppeaine ADD CONSTRAINT FK_Oppeaine_semester_kood FOREIGN KEY
(Semester_kood) REFERENCES Semester (Semester_kood)  ON DELETE NO ACTION ON
UPDATE CASCADE;
ALTER TABLE Oppeaine ADD CONSTRAINT FK_Oppeaine_keel_kood FOREIGN KEY
(Keel_kood) REFERENCES Keel (Keel_kood)  ON DELETE NO ACTION ON UPDATE
CASCADE;
```

Lisa 7 – Klassfikaatorite väärtustamise laused

```
INSERT INTO Semester (Semester_kood, Nimetus, Kirjeldus) VALUES ('S',
'Sügis', 'Sügissemester'), ('K', 'Kevad', 'Kevadsemester'), ('SK',
'SügisKevad', 'SügisKevadsemester');

INSERT INTO Kontrollvorm (Kontrollvorm_kood, Nimetus, Kirjeldus) VALUES ('E',
'Eksam', 'Eksam semestri lõpus'), ('H', 'Hinne', 'Hindeline arvestus'), ('A',
'Arvestus', 'Arvestus');

INSERT INTO Keel (Keel_kood, Nimetus, Kirjeldus) VALUES ('est', 'Eesti',
'Eesti keel'), ('eng', 'Inglise', 'Inglise keel'), ('rus', 'Vene', 'Vene
keel');

INSERT INTO Roll_aines (Roll_Aines_kood, Nimetus, Kirjeldus) VALUES (1,
'Harjutustunni õppejõud', 'Harjutustunni õppejõud'), (2, 'Praktikumi
õppejõud', 'Praktikumi õppejõud'), (3, 'Loengu õppejõud', 'Loengu õppejõud');

INSERT INTO Amet(Amet_kood, Nimetus, Kirjeldus) VALUES (1,'professor',
'Professor on rahvusvahelisel tasemel teadus-, arendus- või muus
loometegevuses aktiivselt osalev oma eriala juhtiv õppejõud.'),
(2,'dotsent', 'Dotsent on õppejõud, kes viib ülikoolis läbi oma ainevaldkonna
õpet, osaleb aktiivselt teadus-, arendus- või muus loometegevuses ning
juhendab tulemuslikult nendesse tegevustesse kaasatud üliõpilasi ja
õppejõude.'),
(3,'lektor', 'Lektor on õppejõud, kes viib ülikoolis läbi õpet kõrghariduse
esimesel kahel astmel või juhul, kui lektoril on doktorikraadkraad, kõigil
kõrgharidustaseme astmetel. '),
(4,'assistent', 'Assistent on õppejõud, kes viib ülikoolis läbi õpet
kõrghariduse esimesel kahel astmel. Assistent võib juhendada kõrghariduse
esimese ja teise astme õppe üliõpilasi ning olla kaasatud teadus- ja
arendustegevusse.'),
(5,'õpetaja', 'Õpetaja on õppejõud, kes viib ülikoolis läbi õpet kõrghariduse
esimesel kahel astmel. Õpetajal on magistrikraad või sellele vastav
kvalifikatsioon. Rakenduskõrgharidusõppe õpetajal on vähemalt kõrgharidus.'),
(6,'juhtivteadur', 'Juhtivteadur on oma eriala rahvusvaheliselt tunnustatud
teadlane. Juhtivteaduril on doktorikraad või sellele vastav
kvalifikatsioon.'),
(7,'vanemteadur', 'Vanemteadur on oma eriala tunnustatud teadlane.
Vanemteaduril on doktorikraad või sellele vastav kvalifikatsioon.'),
(8,'teadur', 'Teadur on teadustöötaja, kes osaleb teadusteema või
uurimisprojekti täitmises. Teaduril on teaduskraad või sellele vastav
kvalifikatsioon.'),
(9,'nooremteadur', 'Nooremteadur on teadustöötaja, kes osaleb teadusteema või
uurimisprojekti täitmises. Nooremteaduril on vähemalt magistrikraad või
sellele vastav kvalifikatsioon.'),
(10,'administraator', 'Administraator tagab ülikooli infosüsteemide tõrgeteta
töö');

INSERT INTO Yliopilase_seisundi_liik(Yliopilase_seisundi_liik_kood, Nimetus,
Kirjeldus) VALUES
(1,'immatrikuleeritud', 'Isik on arvatud üliõpilaste nimekirja'),
```

```
(2,'lõpetamisetapis', 'Üliõpilane on kogunud õppekavaga ette nähtud
ainepunktide summa ning ei pea seetõttu täitma õppetegevuse eeskirjas toodud
õpingukava kohustusliku esitamise ja sooritusnõudeid.'),
(3,'eksmatrikuleeritud', 'Isik on üliõpilaste nimekirjast välja arvatud'),
(4,'akadeemilisel puhkusel', 'Üliõpilane on vabastatud õppe- ja teadustöö
kohustustest.');

INSERT INTO Isiku_seisundi_liik (Isiku_seisundi_liik_kood, Nimetus) VALUES
(1, 'Töötab'), (2, 'Õpib'), (3, 'Ei tööta') ;

INSERT INTO Tootaja_seisundi_liik (Tootaja_seisundi_liik_kood, Nimetus)
VALUES (1, 'Töötab'), (2, 'Puhkusel'), (3, 'Haige') ;

INSERT INTO Oppeaine_seisundi_liik (Oppeaine_seisundi_liik_kood, Nimetus)
VALUES (1, 'Aktiivne'),(2, 'Mitteaktiivne'), (3, 'Kinnitatud'), (4,
'Koostamisel');
```

Lisa 8 – Andmebaasi andmemaht enne testandmete sisestust

Tabelis 30 on esitatud andmebaasi tabelite andmemahud enne testandmete sisestamist.

Tabel 30. Andmebaasi andmemaht enne testandmete sisestust.

Tabeli nimi	**5NK andmebaas, maht baitides**	**Denormaliseeritud andmebaas, maht baitides**
Amet	16384	24576
Isik	40960	73728
Isiku_seisundi_liik	16384	24576
Keel	16384	24576
Kontrollvorm	16384	24576
Oppeaine	57344	57344
Oppeaine_seisundi_liik	16384	24576
Oppimine	32768	32768
Roll_aines	16384	24576
Semester	16384	24576
Tootaja	24576	-
Tootaja_aines	32768	40960
Tootaja_seisundi_liik	16384	24576
Yliopilane	24576	-
Yliopilase_seisundi_liik	16384	24576
KOKKU	360448	425984