TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Informaatikainstituut
Infosüsteemide õppetool

Animatsioonid andmebaasi normaliseerimise kohta SQL-andmebaaside näitel

Bakalaureusetöö

Üliõpilane: Katre Metsvahi
Üliõpilaskood: 120933IAPB
Juhendaja: dotsent Erki Eessaar

Tallinn
2015

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.

_____ _____

(kuupäev) *(allkiri)*

Annotatsioon

Animatsioonid andmebaasi normaliseerimise kohta SQL-andmebaaside näitel

Antud bakalaureusetöö eesmärk on luua õppematerjalina kasutatavad animatsioonid, mis selgitavad SQL-andmebaaside näitel, kuidas käib normaliseerimine, tutvustavad erinevaid normaalkujusid ning näitavad, kuidas need mõjutavad päringute ja andmemuudatuste tegemist. Animatsioonid on tehtud jQueryga. Töö tulemusena valmib seitse animatsiooni.

Esmalt kirjutatakse töös lühidalt normaliseerimisest ja normaalkujudest. Sellele järgneb kokkuvõte jQueryst ja selle eelistest ning ka ülejäänud kasutatud raamistikest. Seejärel antakse ülevaade üldisest animatsioonide lehekülje ülesehitusest ning kirjeldatakse animatsioonide põhimõtteid UMLiga ja realiseeritakse need animatsioonid. Lõpuks võrreldakse loodud animatsioone teiste samateemaliste animatsioonidega.

Animatsioonid asuvad aadressil: http://viktor.ld.ttu.ee/animation_db_normalization/.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 35 leheküljel, 7 peatükki, 16 joonist.

Abstract

Animations About Database Normalization in the Example of SQL databases

The goal of this bachelor thesis is to create animations that can be used as learning materials. The animations will explain, in the example of SQL-databases, the process of normalization, give an overview of various normal forms, and explain how it affects database queries and data modifications. There will be seven animations and the animations will be created with jQuery.

Firstly, a short overview of normalization and normal forms will be given. After that will come a summary of jQuery library and the benefits that come with it, along with other frameworks used. Thirdly, principles of the animations will be explained using UML after which the animations will be implemented. In the end, there will be a comparison of other animations on the same topic.

The animations can be accessed at: http://viktor.ld.ttu.ee/animation_db_normalization/.

The thesis is in Estonian and contains 35 pages of text, 7 chapters and 16 figures.

Lühendite ja mõistete sõnastik

Dekomponeerimine ***Decomposition***

Mingi suurema objekti jagamine väiksemateks osadeks ehk tükeldamine. Antud töö kontekstis on dekomponeeritavateks objektideks SQL-tabelid.

Funktsionaalne sõltuvus ***Functional dependency***

Veergude hulk B on funktsionaalselt sõltuv veergude hulgast A, kui igale A väärtusele vastab täpselt üks B väärtus (Eessaar 2014).

Multiväärtuslik sõltuvus ***Multivalued dependency***

Olgu meil tabelis T kolm veergude hulka A, B ja C – T{A, B, C}. Öeldakse, et veergude hulkade A ja B vahel on multiväärtuslik sõltuvus, kui igas võimalikus ja legaalses T väärtuses sõltub igale AC väärtuste paarile vastav B väärtuste hulk A väärtusest ning on sõltumatu C väärtusest. (Eessaar 2014)

Plugin ***Plugin***

Tarkvaramoodul, mis lisab suuremale süsteemile teatud omaduse või teenuse (Vallaste 2015).

Rakendusliides ***Application Programming Interface (API)***

„Arvuti operatsioonisüsteemiga või rakendusprogrammiga määratud reeglistik, mille alusel rakendusprogramm kasutab operatsioonisüsteemi või teise rakendusprogrammi teenuseid (Vallaste 2015).

Relatsioon ***Relation***

Relatsiooniline väärtus. Seda saab illustreerida tabelina.

Sisuedastusvõrk ***Content Delivery Network (CDN)***

Internetis või suures intranetis laialipaigutatud sisuga süsteem, kus sisust tekitatakse palju koopiaid ja puhverdatakse neid üle kogu võrgu. Selline meetod võimaldab kasutajatel sisule kiiremini juurde pääseda võrreldes sellega, kui sisu paikneks ainult ühel serveril (ühel veebisaidil).

Ühendamissõltuvus ***Join dependency***

T_1, T_2, ..., T_n on tabeli T atribuutide alamhulgad. Tabelis T on ühendamis-sõltuvus atribuutide alamhulkade (T_1, T_2, …, T_n) vahel siis, kui tabeli T iga legaalse väärtuse saab taastada T_1, T_2, …, T_n vastavate projektsioonide ühendamise tulemusel. (Eessaar 2014)

Jooniste nimekiri

Sisukord

1. Sissejuhatus

Bakalaureusetöö valikul lähtusin soovist luua midagi, millest oleks kasu suuremale hulgale inimestele. Õppematerjalide loomine tundus suurepärane võimalus selle soovi täide viimiseks.

Olen läbinud ained Andmebaasid I ja II, seega olin normaliseerimise teemaga juba tuttav. Leidsin ka, et antud teema kohta ei ole piisavalt ajakohaseid animatsioone, mis pakuks abi antud teema omandamisel. Seetõttu saigi valitud just selline teema, lootuses, et sellest ka tulevastele üliõpilastele või teistele andmebaaside huvilistele kasu on.

1.1 Taust ja probleem

Antud bakalaureusetöö on loodud õppematerjalina üliõpilastele, kes õpivad mõnda andmebaaside ainet, ja ka kõigile teistele huvilistele. See töö on kasulik kõigile neile, kes mingil moel tegelevad andmebaasidega, sest normaliseerimine ja selle mõistmine aitab tunduvalt parandada andmebaasi struktuuri ja muuta päringute tegemist efektiivsemaks.

Töö valmis autori õpingute ajal Tallinna Tehnikaülikoolis.

1.2 Ülesande püstitus

Antud töö käsitleb normaliseerimist SQL-andmebaaside näitel. Töö käigus valmivad animatsioonid esimesest kuuest normaalkujust ja Boyce/Coddi normaalkujust. Animatsioonide juures on selgitused, kuidas tabelit vastavale normaalkujule viia ning ka väikesed ülesanded, et õpitud teadmisi kontrollida. Lisaks on toodud ära ka päringute ja andmemuudatuste laused ning nende erinevused enne ja peale kõrgemale normaalkujule viimist.

1.3 Metoodika

Käesolev töö on näiteks disainiteaduse (*design science*) metoodika rakendamisest, mille tulemuseks on uus ja vajalik artefakt (antud juhul animatsioonide komplekt) teatud probleemide lahendamiseks (antud juhul normaliseerimise mõistmise parandamine) (Wikipedia 2015b). Loodud animatsioone katsetab ja annab tagasisidet andmebaaside õppejõud, kes on ühtlasi töö juhendaja. Samuti võrdleb autor animatsioone teiste sarnaste animatsioonidega.

Animatsioonide loomiseks kasutatakse jQuery ja jQuery UI raamistikke. Neil on sisseehitatud meetodid animatsioonide loomiseks ning samuti on nendega tehtud veebilehed kasutatavad pea kõigis tänapäeva veebilehitsejates.

Kõik animatsioonide lehed on üles ehitatud nii, et neis olevaid andmeid oleks võimalikult lihtne muuta. Tabelites olev info ja võrdluste all toodud päringud võetakse eraldi failist. Nii on võimalik muuta andmeid ilma lehekülje enda koodi muutmata.

1.4 Ülevaade tööst

Kõigepealt antakse ülevaade normaliseerimisest ja normaalkujudest. Sellele järgneb kirjeldus jQueryst ja jQuery UI'st. Kolmandaks tuuakse välja animatsioonide skeem ja ülesehitus, millele järgneb võrdlus juba olemasolevate samateemaliste animatsioonidega.

2. Normaliseerimine

Normaliseerimine üldiselt on mõeldud normaliseeritavate struktuuride arusaadavuse parandamiseks ning sellest tulenevalt disainivigade parandamiseks, huvide eristamise (*separation of concerns*) saavutamiseks ja liiasuse vähendamiseks. Normaliseerimine on hästi tuntud andmebaaside valdkonnas. Käesolevas töös ja loodavates animatsioonides käsitletakse normaliseerimist SQL-andmebaaside näitel ning seetõttu kirjutan edaspidi baastabelite (tabelite) normaliseerimisest.

Normaliseerimine on mitmesammuline protsess, mille käigus organiseeritakse andmeid andmebaasis (Chapple 2015). Tabeli (täiendava) normaliseerimise käigus viiakse tabelid iga sammuga üha kõrgemale normaalkujule, eesmärgiga vähendada andmete liiasust ja sellega seonduvalt andmete muutmisega kaasnevaid probleeme, parandada loogiliselt ebakorrektset disaini, muuta andmebaasi struktuuri kergemini mõistetavaks ning lihtsustada kitsenduste jõustamist (Eessaar 2014).

2.1 Normaalkujud

Tabel on mingil normaalkujul, kui see rahuldab kõiki selle normaalkujuga seotud tingimusi (Eessaar 2014). On olemas palju erinevaid normaalkujusid, kuid üldiselt kehtivad neile samad reeglid:

- Kui tabel on normaalkujul $n + 1$, siis on see ka normaalkujul n
- Tabel, mis on normaalkujul n, ei pruugi olla normaalkujul $n + 1$
- Tabeli mingile normaalkujule viimisel otsitakse tabeli veergude vahel teatavaid seoseid ehk sõltuvusi ning nende leidmisel dekomponeeritakse ehk tükeldatakse tabel nende alusel väiksemateks tabeliteks. Tabel dekomponeeritakse nii, et algset tükki on võimalik tükkidest taastada, kõiki tükke läheb taastamiseks vaja ning kõiki kitsendusi ja piiranguid, mida sai rakendada algsele tabelile, saab rakendada ka selle tükelduse tulemusele.
- Andmebaasi disainimisel tasub eelistada kõrgemaid normaalkujusid (Date 2005). Tükeldamist tagasipöörata ehk tabeleid denormaliseerida tasub vaid juhul, kui andmebaasi realiseerimiseks kasutatud andmebaasisüsteemis on oluliste päringute jõudlus halb, andmebaasis tehakse suhteliselt vähe andmemuudatusi ning muud jõudluse parandamise meetodid ei anna soovitud tulemusi.

Kui räägitakse normaliseerimisest, siis enamasti mõeldakse selle all just esimest kuni viiendat normaalkuju. Oma bakalaureusetöös loon ma animatsioonid neile viiele ja lisaks veel Boyce/Coddi normaalkujule (jääb kolmanda ja neljanda normaalkuju vahele) ja kuuendale normaalkujule.

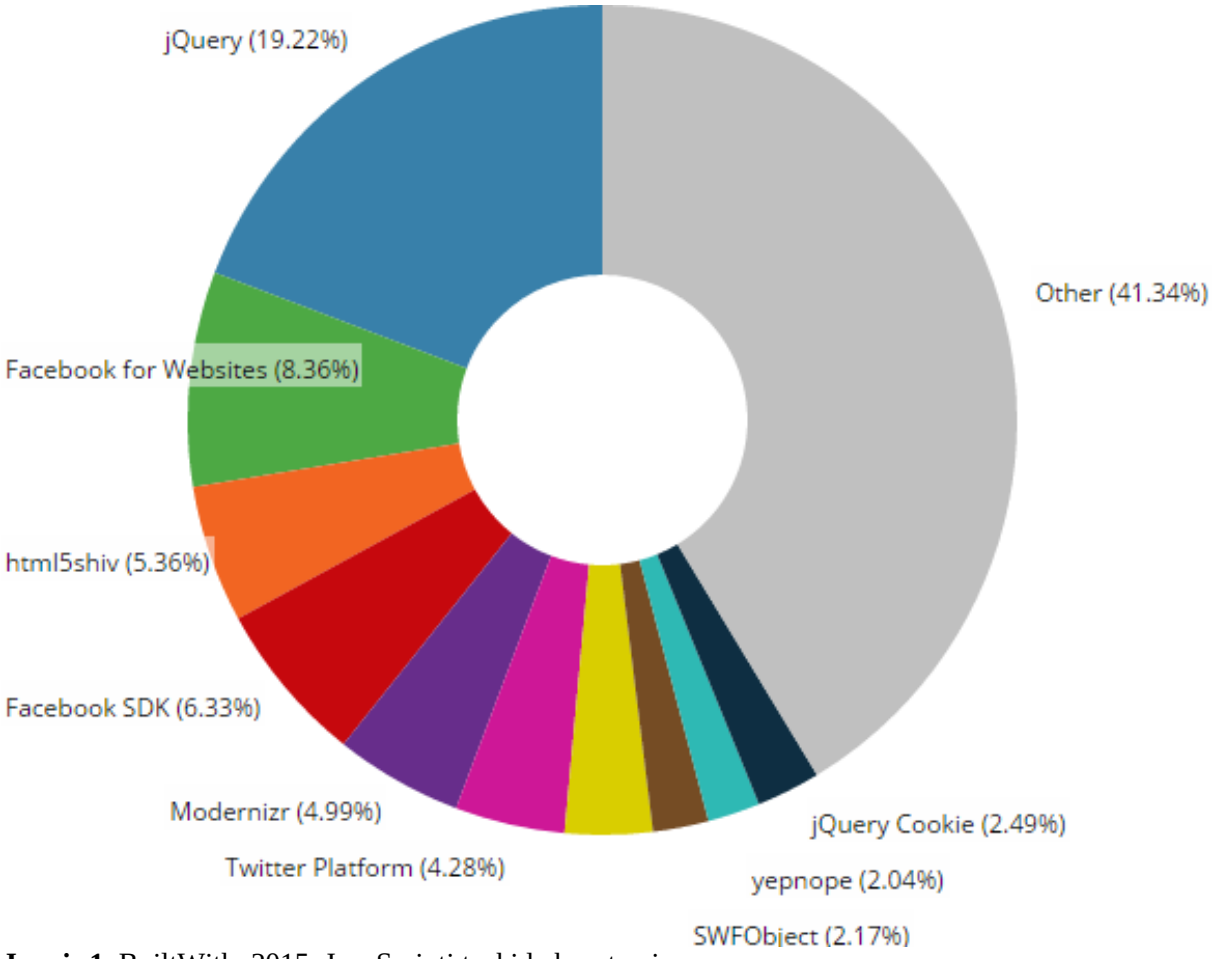
Esimesed kolm normaalkuju pakkus välja E.F. Codd aastal 1972. Need põhinevad relatsioonide kadudeta dekomponeerimisel funktsionaalse sõltuvuse alusel. I. Heath ja peale teda ka R.F. Boyce ja E.F. Codd leidsid kolmanda normaalkuju definitsioonist aga ebatäpsusi ja 1974. aastal avaldasid nad Boyce/Coddi normaalkuju definitsiooni. Neljanda normaalkuju, mis põhineb multiväärtuslikel sõltuvustel, pakkus välja R. Fagin aastal 1977 ning samuti on tema välja pakutud viienda normaalkuju definitsioon (1979), mis põhineb ühendamis-sõltuvusel. (Fotache 2006, 5)

Aastal 2002 pakkusid Date, Darwen ja Lorenzos välja ka kuuenda normaalkuju, eesmärgiga lihtsustada ajaliste ehk temporaalsete andmete haldamist. Sellel normaalkujul tabelite kasutamine on praktikas aina rohkem kasutust leidnud tänu ankurmodelleerimisele, mille kasutamise tulemusena luuakse kuuendal normaalkujul olevad tabelid (Anchor Modeling 2015). Selliste tabelite eeliseks on avanevad head võimalused ajalooliste andmete säilitamiseks, andmebaasi skeemi evolutsiooni lihtsustamine ning puuduvate andmetega toimetulek.

3. Animatsioonide loomiseks kasutatud raamistikud

3.1 jQuery

jQuery on vaieldamatult hetke kõige populaarsem JavaScripti teek (BuiltWith 2015). jQuery rakendusliidest on lihtne kasutada, see töötab kõigis enimtuntud veebilehitsejates ning teeb palju lihtsamaks mitmete JavaScripti funktsioonide rakendamise (jQuery 2015). Antud bakalaureusetöös on kasutatud jQuery mitmeid animeerimisvõimalusi.



Joonis 1. BuiltWith, 2015. JavaScripti teekide kasutamine

jQueryt kasutava veebilehe vaatamiseks ei pea kasutaja oma arvutisse eraldi programmi tõmbama. Kui veebilehitseja toetab JavaScripti, toetab see ka jQueryt. Selliseid veebilehitsejaid, mis JavaScripti ei toeta, on aga väga vähe (Wikipedia 2015a).

Antud bakalaureusetöö tegemisel kasutati jQuery versiooni 1.11.2.

3.1.1 Kasutamine

jQuery lisamine oma veebilehele on väga lihtne. Selleks tuleb HTML päisesse lisada viide jQuery teeki sisaldavale failile. Antud faili võib kas alla laadida oma arvutisse või serverisse või siis kasutada selle asemel hoopis mõnda sisuedastusvõrku.

```
<script src="js/jquery.js"></script>
```

Mõlemal juhul on teegi uuendamine väga lihtne. Uuele versioonile viitamiseks tuleb vaid uuendada *script* märgendi sees oleva *src* parameetri väärtust. Faili allalaadimisel võib vana faili ka uuega üle kirjutada.

3.1.2 Meetodid

jQueryl on palju meetodeid, millega on võimalik luua animatsioone. Järgnevalt toon ära nimekirja kasutatud meetoditest.

3.1.2.1 fadeIn

```
.fadeIn([kestvus_millisekundites])
```

Teeb soovitud elemendi nähtavaks, muutes selle läbipaistvast nähtavaks etteantud aja jooksul (jQuery API Documentation 2015).

3.1.2.2 fadeOut

```
.fadeOut([kestvus_millisekundites])
```

Peidab soovitud elemendi, muutes selle nähtavast läbipaistvaks etteantud aja jooksul (jQuery API Documentation 2015).

3.1.2.3 fadeTo

```
.fadeTo([kestvus_millisekundites], [läbipaistvus])
```

Muudab soovitud elemendi läbipaistvust vastavalt etteantud väärtusele etteantud aja jooksul (jQuery API Documentation 2015).

3.1.2.4 slideUp

```
.slideUp([kestvus_millisekundites])
```

Peidab soovitud elemendid liugleva animatsiooniga (jQuery API Documentation 2015).

3.1.3 Iseloodud meetodid

jQuery võimaldab ka ise oma meetodeid luua. Antud lõputöös sai loodud kaks uut meetodit: *.visible([kestvus_millisekundites)]* ja *.invisible([kestvus_millisekundites)]).*

Näiliselt on need täpselt samad, mis fadeIn ja fadeOut, kuid kui need kaks muudavad elemendi *display* atribuudi väärtust, siis meetodid visible ja invisible muudavad elemendi *visibility* atribuudi väärtust. See on näha ka nende loomise koodist:

```
jQuery.fn.visible = function(speed) {
    return this.css({opacity: 0.0, visibility: 'visible'}).animate({opacity: 1.0},
    speed);
};
```

```
jQuery.fn.invisible = function(speed) {
    return this.animate({opacity: 0.0}, speed);
};
```

Visible ja *display* vahe on selles, et kui elemendile on antud *display: none*, siis seda elementi justkui ei olekski lehel, aga *visibility: hidden* puhul elementi ei ole küll näha, aga see võtab ikka ruumi.

3.1.4 Kestvused

Lisaks numbrilisele väärtusele on jQuerys võimalik kestvus määrata ka sõnadega. Selleks on kolm väärtust: „slow“, „normal“ ja „fast“. Nende väärtused on vastavalt 200, 400 ja 600 millisekundit.

Kestvuse määramiseks on veel ka kolmas variant: jätta see hoopis määramata. Siis võetakse kasutusele vaikimisi väärtus ehk „normal“ (400 millisekundit).

3.2 jQuery UI

jQuery UI on hulk jQuery põhjal loodud kasutajaliidese interaktsioone, efekte, mooduleid ja teemasid (jQuery UI 2015). See lisab jQueryle uusi meetodeid ja täiendab ka mõningaid juba jQuerys olemasolevaid meetodeid.

Antud bakalaureusetöö tegemisel kasutati jQuery UI versiooni 1.11.4.

3.2.1 Kasutamine

jQuery UI lisamine veebilehele käib täpselt samamoodi, nagu jQuery lisamine (vt jaotis 3.1.1). Vajaliku faili võib laadida alla või võtta sisuedastusvõrgust. jQuery UI vajab toimimiseks kindlasti ka jQueryt, seega tuleb lisada HTML päisesse lisada viide jQueryle ja see peab paiknema jQuery UI'st eespool.

```
<script src="js/jquery.js"></script>
<script src="js/jquery-ui.js"></script>
```

3.2.2 Meetodid

jQuerys on olemas meetodid elementidele kujunduse klasside lisamiseks ja eemaldamiseks. Seda on hea kasutada näiteks selleks, et muuta teksti värvi. Vajaliku klassi saab CSSiga valmis teha ja siis selle klassi jQueryga lisada ning vajadusel ka eemaldada. Sedasi toimides jääb kogu kujunduslik osa CSSi alla. Küll on aga probleem selles, et see muudatus toimub hetkelt. Mõnikord on aga vaja, et muutus toimuks sujuvalt. jQuery UI lisab eeltoodud jQuery meetoditele ka kestvuse parameetri, võimaldades sellega animeerida teksti värvi muutumist.

3.2.2.1 addClass

```
.addClass([klassi_nimi], [kestvus_millisekundites])
```

Lisab elemendile klassi ning animeerib kaasnevad muudatused (jQuery UI 2015).

3.2.2.2 removeClass

```
.removeClass([klassi_nimi], [kestvus_millisekundites])
```

Eemaldab elemendilt klassi ning animeerib kaasnevad muudatused (jQuery UI 2015).

4. Muud kasutatud raamistikud ja tehnoloogiad

Lisaks jQueryle ja jQuery UI'le, millega said tehtud kõik animatsioonid, kasutatakse antud bakalaureusetöö tegemisel ka Bootstrapi, LESSi ja AngularJS raamistikke ja tehnoloogiaid.

4.1 Bootstrap

Bootstrap on üks populaarsemaid HTMLi, CSSi ja JavaScripti raamistikke. See pakub võimalust ehitada kiirelt ja mugavalt veebilehti, mis kohanduvad vastavalt kasutaja ekraani suurusele.

Ka Bootstrapil on mõned sisseehitatud animatsioonid, kuid neid antud lõputöös ei kasutatud, sest jQuery pakub animatsioonide tegemiseks palju rohkem võimalusi.

Antud bakalaureusetöö tegemisel kasutati Bootstrapi versiooni 3.3.4.

4.1.1 Kasutamine

Bootstrapi kasutamiseks tuleb HMTL päisesse lisada viited Bootstrapi failidele. Erinevus jQueryga on selles, et Bootstrapi puhul on faile mitu: üks CSSi, teine JavaScripti jaoks. Mõlema kasutamine ei ole kohustuslik. Bootstrapi JavaScripti pluginate ehk pistikprogrammide toimimiseks on vajalik ka jQuery, see tuleb lisada päisesse enne Bootstrapi JavaScripti faili.

```
<link href="css/bootstrap.css" rel="stylesheet">
<script src="js/jquery.js"></script>
<script src="js/bootstrap.js"></script>
```

4.2 AngularJS

AngularJS on raamistik, mis täiendab HTMLi sõnastikku (AngularJS 2015). Vajaliku koodi võib kirjutada otse HTML faili. See teeb koodi märksa loetavamaks ja intuitiivsemaks, sest soovitud muutujad on kohe koodi sees seal, kus neid soovitakse kasutada.

```
<p>1 + 2 = {{ 1 + 2 }}</p>
```

Võrdluseks sama kood JavaScriptiga:

```
<p>1 + 2 = <span id="result"></span></p>
<script>
    document.getElementById('result').innerHTML = (1 + 2);
</script>
```

AngularJS'i on antud bakalaureusetöös kasutatud selliste andmete kuvamiseks, mis võetakse eraldiseisvast andmefailist. See teeb vajadusel andmete muutmise märksa lihtsamaks, sest siis ei pea hakkama näiteks jQuery failist otsima kohta, kus andmed sisse loetakse, vaid saab kohe HTML failis õige koha üles otsida ja vajaliku muudatuse teha.

Antud bakalaureusetöö tegemisel kasutati AngularJS versiooni 1.3.15.

4.2.1 Kasutamine

AngularJS kasutamiseks tuleb lisada viide AngularJS failile HTML päisesse.

```
<script src="js/angular.js"></script>
```

4.3 Less

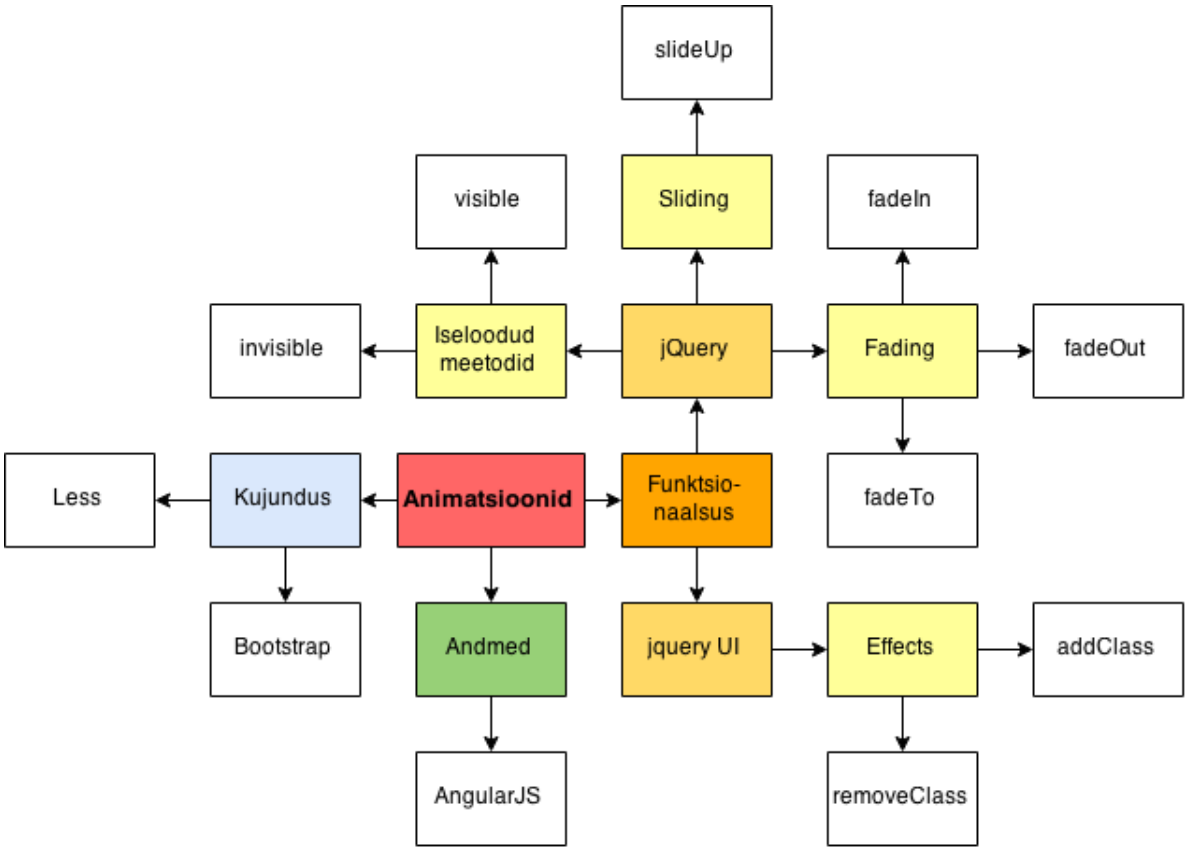
Less on CSSi eeltöötleja. See laiendab CSSi, lisades sellele palju uusi võimalusi, näiteks muutujate loomise, funktsioonid, võimaluse kirjutada stiile üksteise sisse. See teeb CSSi kasutamise palju mugavamaks, võimaldades kiiresti ja mugavalt muuta kogu lehekülje stiili ja organiseerida elemente. (Less 2015)

4.3.1 Kasutamine

Lessi kasutamiseks on vaja kompilaatorit, mis teeks Lessist CSSi. Antud bakalaureusetöö tegemisel on selleks kasutatud Koalat. Koalale tuleb ainult ette öelda, mis kaustas Less failid asuvad ja siis kompileerib Koala antud faile automaatselt kohe, kui neis muudatus toimub. Saadud CSS failid võib lisada HTML faili päisesse just nagu tavaliselt.

4.4 Seosed kasutatud raamistike ja tehnoloogiate ning animatsiooni elementide vahel

Järgnevalt (vt joonis 2) on esitatud mõistekaart, mis toob välja seosed kasutatud raamistike, tehnoloogiate ja animatsiooni elementide vahel. Eri värve on kasutatud erinevate kategooriate grupeerimiseks: sinisega on märgitud kujundus, rohelisega andmed ja kollase tooniga on märgitud funktsionaalsusega seotud mõisted.



Joonis 2. Seosed raamistike, tehnoloogiate ja animatsiooni elementide vahel

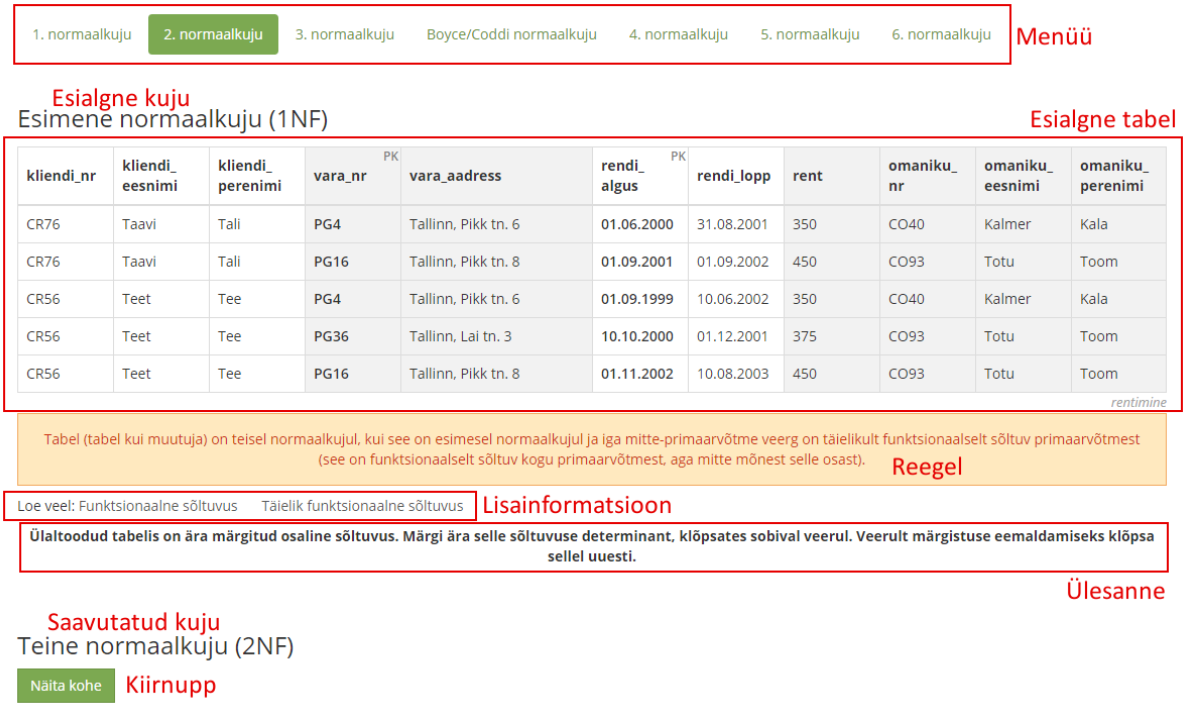
5. Animatsioonide projekteerimine ja realiseerimine

Käesolevas peatükis kirjeldatakse animatsioonide projekteerimist ja realiseerimist. Animatsioonides kasutatavad näitetabelid ja veergude vahelised sõltuvused on saadud õppeaine Andmebaasid I materjalidest. See on taotluslik, et muuta üliõpilastele nende näidete läbimine huvipakkuvamaks ja meeldejäävamaks. Animatsioonides kujutatavad SQL laused ning samuti kogu animatsioonide ülesehitus on käesoleva töö autori looming.

5.1 Animatsioonide seesmine ülesehitus teise normaalkuju animatsiooni näitel

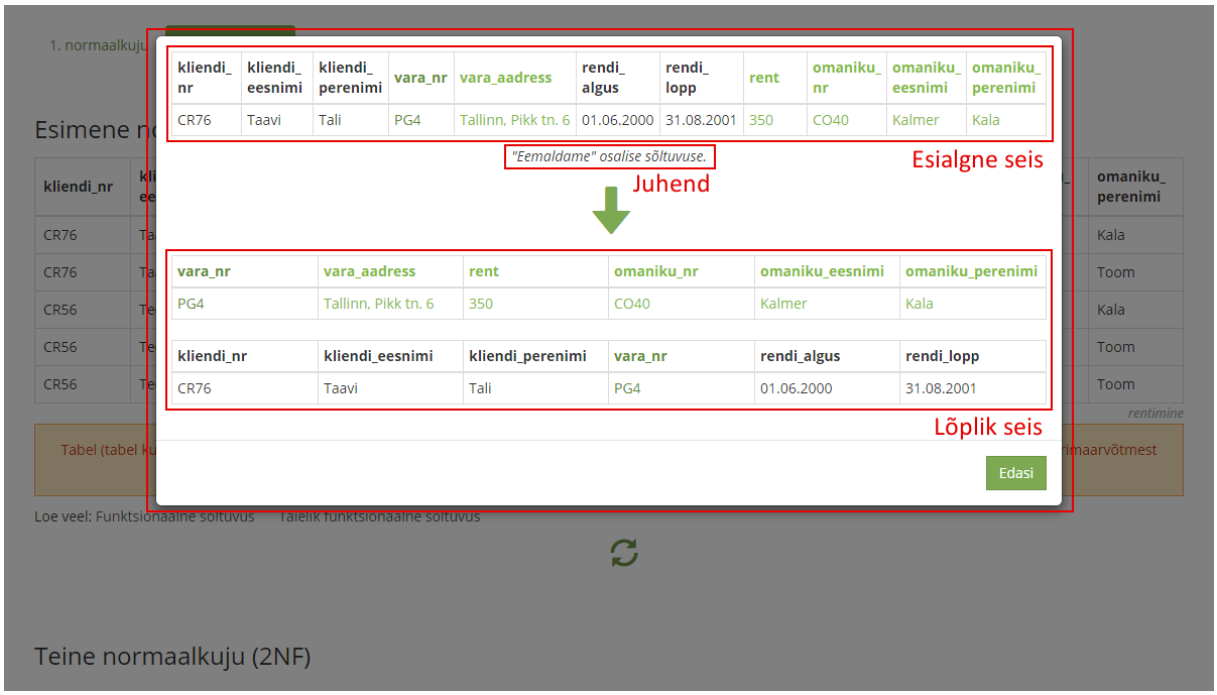
5.1.1 Üldvaade

Kõik animatsioonide leheküljed on üles ehitatud üldiselt sama skeemi järgi. Animatsiooni leheküljele minnes kuvatakse lehekülje algkuju (vt joonis 3).



Joonis 3. Lehekülje algkuju

Peale ülesande edukat lahendamist avatakse modaalaken ja näidatakse kasutajale selgitavat animatsiooni (vt joonis 4). Peale animatsiooni lõppemist võib kasutaja modaalakna sulgeda.



Joonis 4. Modaalaken

Peale „Edasi“ nupu vajutamist suletakse modaalaken ja kuvatakse lehekülje lõppkuju (vt joonis 5), kus on normaliseerimise tulemusena saadud tabelid ning algse ja lõpliku kuju vahelised võrdlused.

Sama tulemuse saab ka siis, kui vajutada nuppu „Näita kohe“. Siis modaalakent ei avata, vaid näidatakse kohe lehekülje lõppkuju.

1. normaalkuju | 2. normaalkuju | 3. normaalkuju | Boyce/Coddi normaalkuju | 4. normaalkuju | 5. normaalkuju | 6. normaalkuju

Esialgne kuju

Esimene normaalkuju (1NF)

kliendi_nr	kliendi_ eesnimi	kliendi_ perenimi	vara_nr PK	vara_aadress	rendi_ algus PK	rendi_lopp	rent	omaniku_ nr	omaniku_ eesnimi	omaniku_ perenimi
CR76	Taavi	Tali	PG4	Tallinn, Pikk tn. 6	01.06.2000	31.08.2001	350	CO40	Kalmer	Kala
CR76	Taavi	Tali	PG16	Tallinn, Pikk tn. 8	01.09.2001	01.09.2002	450	CO93	Totu	Toom
CR56	Teet	Tee	PG4	Tallinn, Pikk tn. 6	01.09.1999	10.06.2002	350	CO40	Kalmer	Kala
CR56	Teet	Tee	PG36	Tallinn, Lai tn. 3	10.10.2000	01.12.2001	375	CO93	Totu	Toom
CR56	Teet	Tee	PG16	Tallinn, Pikk tn. 8	01.11.2002	10.08.2003	450	CO93	Totu	Toom

rentimine

Tabel (tabel kui muutuja) on teisel normaalkujul, kui see on esimesel normaalkujul ja iga mitte-primaarvõtme veerg on täielikult funktsionaalselt sõltuv primaarvõtmest (see on funktsionaalselt sõltuv kogu primaarvõtmest, aga mitte mõnest selle osast).

Loe veel: Funktsionaalne sõltuvus Täielik funktsionaalne sõltuvus

Ülesande taaslaadimise nupp

Lõplik kuju

Teine normaalkuju (2NF)

kliendi_nr	kliendi_ eesnimi	kliendi_ perenimi	vara_nr PK FK	rendi_ algus PK	rendi_lopp
CR76	Taavi	Tali	PG4	01.06.2000	31.08.2001
CR76	Taavi	Tali	PG16	01.09.2001	01.09.2002
CR56	Teet	Tee	PG4	01.09.1999	10.06.2002
CR56	Teet	Tee	PG36	10.10.2000	01.12.2001
CR56	Teet	Tee	PG16	01.11.2002	10.08.2003

vara_rentimine

vara_nr PK	vara_aadress	rent	omaniku_ nr	omaniku_ eesnimi	omaniku_ perenimi
PG4	Tallinn, Pikk tn. 6	350	CO40	Kalmer	Kala
PG16	Tallinn, Pikk tn. 8	450	CO93	Totu	Toom
PG36	Tallinn, Lai tn. 3	375	CO93	Totu	Toom

vara_omanik

Võrdlus

Võrdlus

Esimene normaalkuju

Teine normaalkuju

Andmete otsimine

SELECT kliendi_eesnimi, kliendi_perenimi, vara_aadress, rendi_algus, rendi_lopp, rent, omaniku_eesnimi, omaniku_perenimi
FROM rentimine

Puudub info vara kohta, mida ei ole veel renditud.

SELECT kliendi_eesnimi, kliendi_perenimi, vara_aadress, rendi_algus, rendi_lopp, rent, omaniku_eesnimi, omaniku_perenimi
FROM vara_rentimine
INNER JOIN vara_omanik
ON vara_rentimine.vara_nr = vara_omanik.vara_nr

SELECT DISTINCT vara_nr, vara_aadress, rent
FROM rentimine

Tuleb lisada DISTINCT piirang.

SELECT vara_nr, vara_aadress, rent
FROM vara_omanik

Rea lisamine

INSERT INTO rentimine *(kliendi_nr, kliendi_eesnimi, kliendi_perenimi, vara_nr, vara_aadress, rendi_algus, rendi_lopp, rent, omaniku_nr, omaniku_eesnimi, omaniku_perenimi)*
VALUES *('CR76', 'Taavi', 'Tali', 'PG36', 'Tallinn, Lai tn. 3', '05.12.2003', '05.03.2004', 375, 'CO93', 'Totu', 'Toom')*

Rea lisamisel võib tekkida andmetes vastuolusid.
PG36 → Lai tn. 3 PG36 → Lai tn. **4**

INSERT INTO vara_rentimine *(kliendi_nr, kliendi_eesnimi, kliendi_perenimi, vara_nr, rendi_algus, rendi_lopp)*
VALUES *('CR76', 'Taavi', 'Tali', 'PG36', '05.12.2003', '05.03.2004')*

Varaga seonduvat infot ei pea enam uuesti sisestama.

Rea muutmine

UPDATE rentimine
SET vara_aadress = 'Tallinn, Lai tn. 4'
WHERE vara_nr = 'PG36'

Muudatus tuleb teha mitmes reas.

UPDATE vara_rentimine
SET vara_aadress = 'Tallinn, Lai tn. 4'
WHERE vara_nr = 'PG36'

Muudatus tehakse ainult ühes reas.

Rea kustutamine

DELETE
FROM rentimine
WHERE vara_nr = 'PG36' **AND** rendi_algus = '10.10.2000'

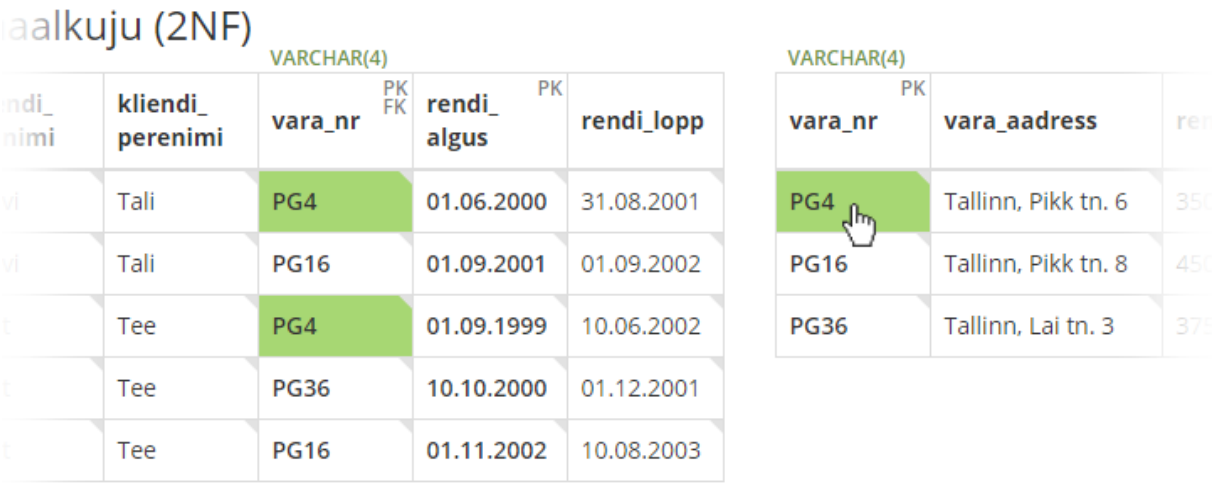
Kaotsi läheb vara nr. 36 aadress ja rent.

DELETE
FROM vara_rentimine
WHERE vara_nr = 'PG36' **AND** rendi_algus = '10.10.2000'

Vara info jääb alles, kuid kliendi info võib kaduma minna.

Joonis 5. Lehekülje lõppkuju

Peale lehekülje lõppkuju laadimist on võimalik viia kursor mõne tabeli lahtri peale ja vaadata, mis andmetüübiga on tegemist (vt joonis 6). Samuti näidatakse ära sellele lahtrile vastavad lahtrid teistes tabelites. Kuna animatsiooni lõppedes on lehel nii tabelid enne kui pärast vaadeldavale normaalkujule viimist, siis on sellisel viisil nähtavad vastavad lahtrid kõigis nendes tabelites. See peaks aitama kasutajal aru saada, kuidas normaliseerimise tulemusena on andmete liiasus tabelites vähenenud.

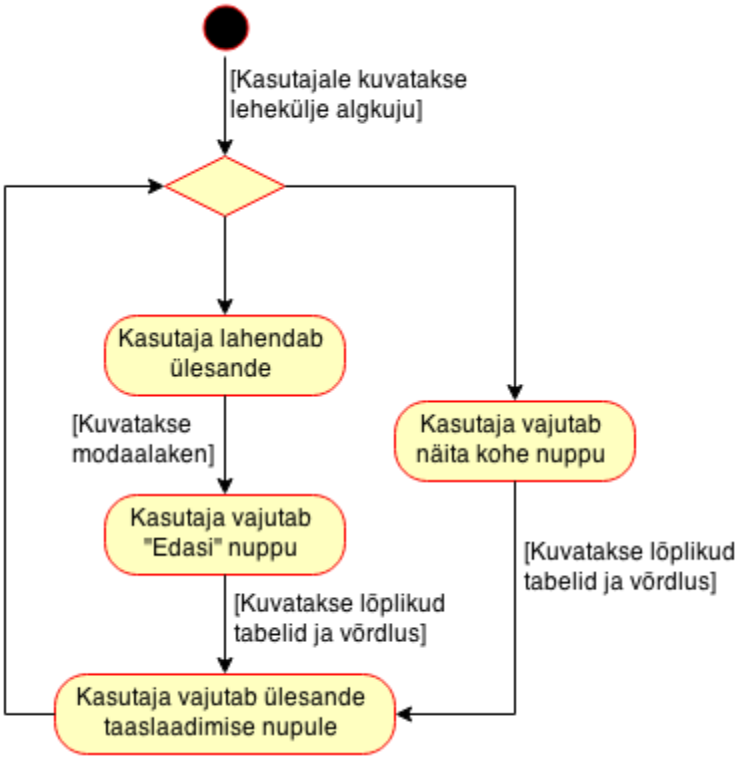


Joonis 6. Kursori viimine lahtri peale

Testisin animatsioone järgnevates brauserites: Google Chrome (versioon 43.0.2357.65 m), Internet Explorer (versioon 11.0.9600.17801), Mozilla Firefox (versioon 37.0.2), Opera (versioon 29.0.1795.60) ja Safari (versioon 5.1.7.7534.57.2). Kõigis nendes töötasid animatsioonid nii nagu oodati.

5.1.2 Tegevusdiagramm

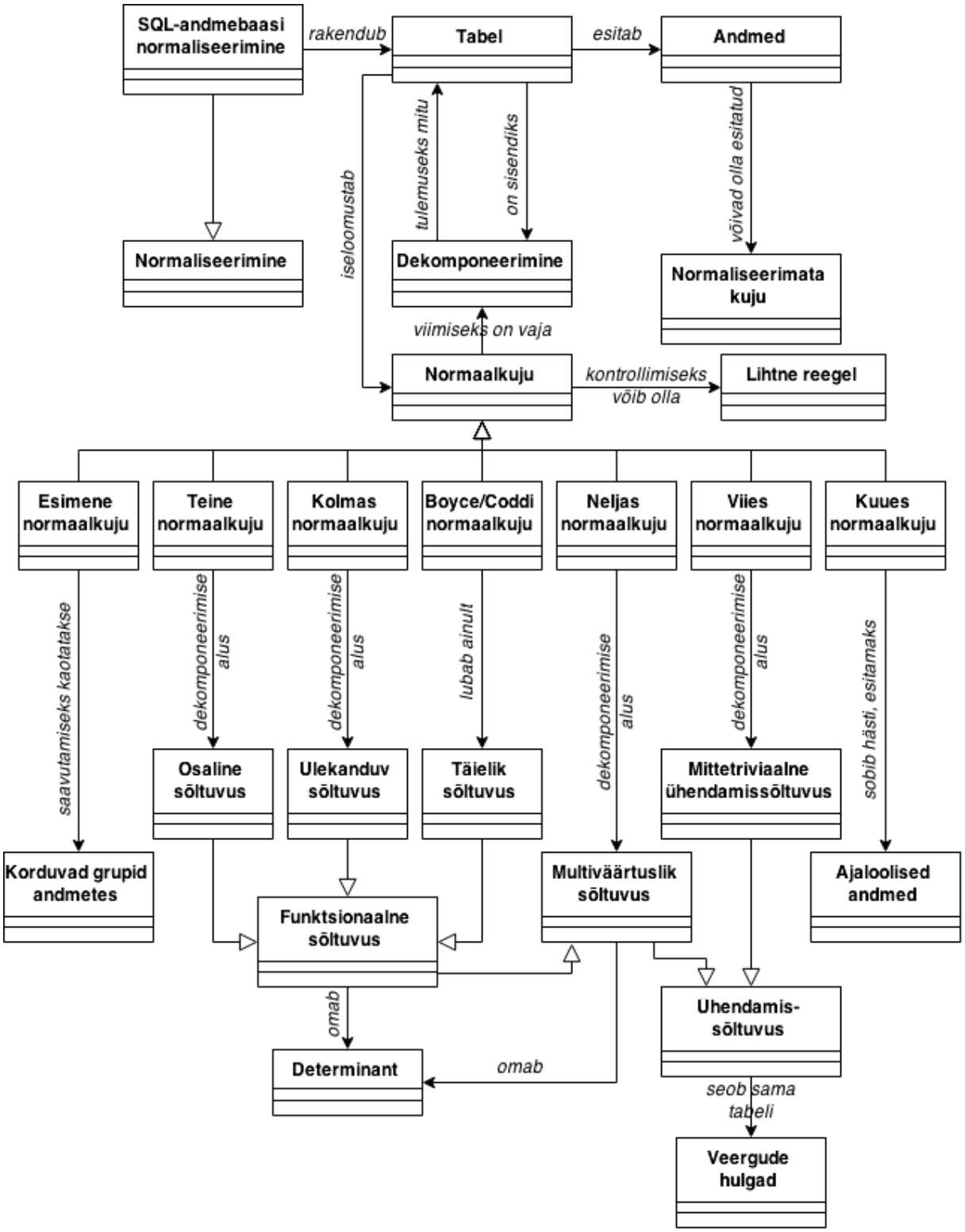
Järgnevalt on toodud teise normaalkuju animatsiooni tegevusdiagramm. Üldiselt järgivad kõik animatsioonid sama skeemi.



Joonis 7. Käesoleva töö animatsioonide üldine tegevusdiagramm

5.1.3 Valdkonnamudel

Järgnevalt esitatakse UML klassidiagramm, mis esitab loodud animatsioonides kirjeldatavad mõisted ja nende mõistete vahelised seosed.



Joonis 8. Käesoleva töö tulemusena loodud animatsioonide mõistete klassidiagramm

5.1.4 Animatsiooni häälestamine

Animatsiooni tabelites olev info ja võrdluste all toodud päringud võetakse eraldi failist. Andmefaili nimi on kujul data*[n]*.js, kus *[n]* on soovitud normaalkuju. Faili nimes võib olla ka vahemik, näiteks data1-3.js, see näitab seda, et samast failist võetakse normaalkujude 1 kuni 3 andmed. Andmefaili kasutades on võimalik muuta andmeid ilma lehekülje enda koodi muutmata. Selles jaotises esitatakse näide selle kohta, kuidas animatsiooniga kaasas käivas failis andmete muutmine põhjustas animatsioonis muudatusi. Joonisel 9 on esitatud animatsiooni fragment enne muudatuse tegemist. Joonisel 11 on esitatud vastavad failis olevad andmed enne muudatuse tegemist. Joonisel 12 on esitatud failis olevad andmed peale muudatuse tegemist. Joonisel 10 on esitatud animatsiooni väljanägemine peale failis andmete muutmist.

PK kliendi_nr	kliendi_eesnimi	kliendi_ perenimi
CR76	Taavi	Tali
CR56	Teet	Tee

klient

Joonis 9. Animatsiooni fragment enne muudatuse tegemist

PK kliendi_nr	kliendi_eesnimi	kliendi_ perenimi
CR76	Kalmer	Kala
CR56	Tiina	Tihane

klient

Joonis 10. Animatsiooni fragment peale muudatuse tegemist

```
var thirdNf1 = {
    "name":"klient",
    "columns":[4, 4, 4],
    "headers":["kliendi_nr", "kliendi_eesnimi", "kliendi_perenimi"],
    "primaryKey":["kliendi_nr"],
    "foreignKeys":[],
    "dataTypes":["VARCHAR(4)", "VARCHAR(35)", "VARCHAR(35)"],
    "dataRows":[
        ["CR76", "Taavi", "Tali"],
        ["CR56", "Teet", "Tee"]
    ]
}
```

Joonis 11. Andmed enne muudatuse tegemist

```
var thirdNf1 = {
    "name":"klient",
    "columns":[4, 4, 4],
    "headers":["kliendi_nr", "kliendi_eesnimi", "kliendi_perenimi"],
    "primaryKey":["kliendi_nr"],
    "foreignKeys":[],
    "dataTypes":["VARCHAR(4)", "VARCHAR(35)", "VARCHAR(35)"],
    "dataRows":[
        ["CR76", "Kalmer", "Kala"],
        ["CR56", "Tiina", "Tihane"]
    ]
}
```

Joonis 12. Andmed peale muudatuse tegemist

Soovi korral on võimalik muuta ka tabelis olevate veergude arvu. Selleks tuleb muuta *„columns“* atribuudi (vt joonis 10 ja 11) väärtust. Nurksulgude vahele peab jääma nii mitu arvu, kui palju on tabelis veerge, ning nende väärtuste summa peab olema 12. See arv tuleb sellest, et antud töös on kujundus üles ehitatud Bootstrapi 12 tulbast koosneva võrgustiku peale. Arvud näitavad veergude laiust, näiteks *„columns“* väärtuse [4, 4, 4] puhul on kõik kolm veergu võrdse laiusega, [8, 4] puhul oleks aga esimene tulp kaks korda laiem kui teine.

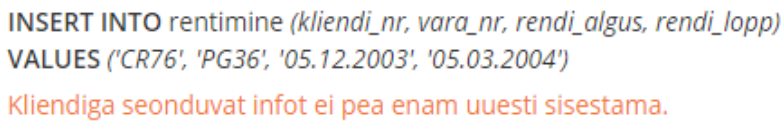
Tabelites olevaid andmeid muutes uuendatakse automaatselt ka vastavad tabelid modaalaknas. Modaalaknas olevate tabelite andmed võetakse täpselt samast kohast, kust tulevad ka põhitabelite andmed, seetõttu ei ole vaja midagi täiendavalt teha selleks, et ka modaalakna tabelites andmed muutuks.

Ka SQL lauseid on võimalik muuta, tehes andmefailis soovitud muudatuse. Sisestada tuleb ainult soovitud SQL lause, kusjuures SQL käsklused tuleb kirjutada trükitähtedes, ning kujundamine toimub automaatselt. Kui lause läheb mitmele reale, tuleks iga (v.a viimase) rea lõppu panna ka längkriips \. Samuti on võimalik muuta vastavate lausete kommentaare.

Joonisel 13 on näidatud SQL lause koos kommentaariga andmefailis ning joonisel 14 on sama lause ja kommentaar animatsioonis.

```
"rea_lisamine":{
    "sentence":"INSERT INTO rentimine (kliendi_nr, vara_nr, rendi_algus, rendi_lopp) \
        VALUES ('CR76', 'PG36', '05.12.2003', '05.03.2004')",
    "comment":"Kliendiga seonduvat infot ei pea enam uuesti sisestama."
},
```

Joonis 13. SQL lause ja selle kommentaar andmefailis



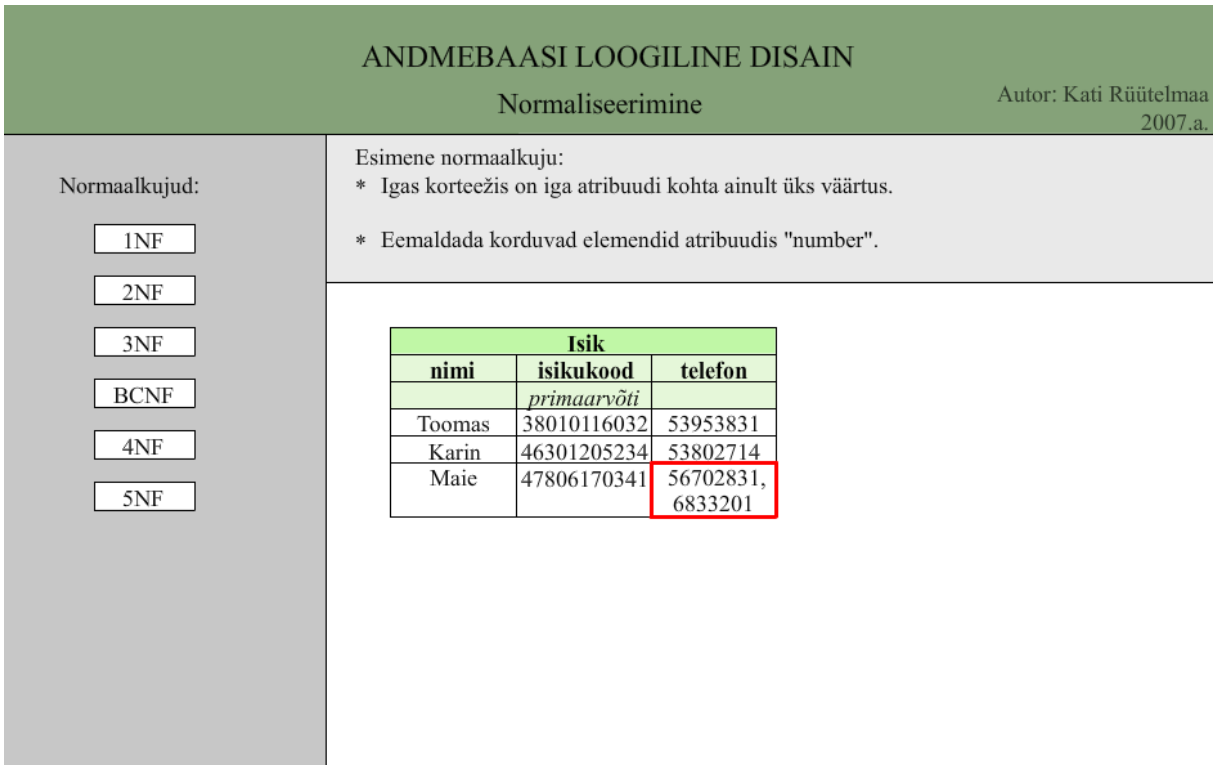
Joonis 14. SQL lause koos kommentaariga animatsioonis

jQueryga on võimalik ka andmebaasist andmeid küsida, kasutades jQuery.ajax päringut. Kui andmebaasist on võimalik pärida andmeid JSON formaadis (nt PostgreSQL, MongoDB või Airtable), võib teha päringu otse andmebaasile, vastasel juhul on vaja aga ka mõnda serveripoolset skriptimiskeelt, nagu näiteks PHP või Python, millele tuleb teha Ajax päring ja mis siis omakorda pärib andmed andmebaasist ja saadab need õiges formaadis jQueryle tagasi.

6. Võrdlus teiste animatsioonidega

6.1 Eestikeelsed animatsioonid

Otsides Google otsingumootorist eesti keeles *„normaliseerimine animatsioon“* leidsin ühe animatsiooni andmebaaside normaliseerimise kohta. See on tehtud Kati Rüütelmaa poolt aastal 2007 samuti lõputööna ning selle leiab aadressilt http://viktor.ld.ttu.ee/db_animatsioonid/4.html (Rüütelmaa 2007). Antud animatsiooni puhul ei saa samaaegselt vaadata alg- ja lõppkuju, seetõttu on raske võrrelda normaliseerimise protsessi sammu sisendiks ja väljundiks olevaid tabeleid. Antud animatsioon on tehtud Flashiga, seega ei saa seda kasutada seadmetega, mis Flashi ei toeta, nagu näiteks enamus mobiilseid seadmeid. Animatsiooni aken on kindlaksmääratud suurusega, seega on seda võimatu kasutada seadmetel, mille ekraan on sellest väiksem. Osa animatsioonist jääb siis ekraani serva taha ja kerimine ka ei toimi. Suurema ekraani puhul jääb aga lehele palju tühja ruumi.

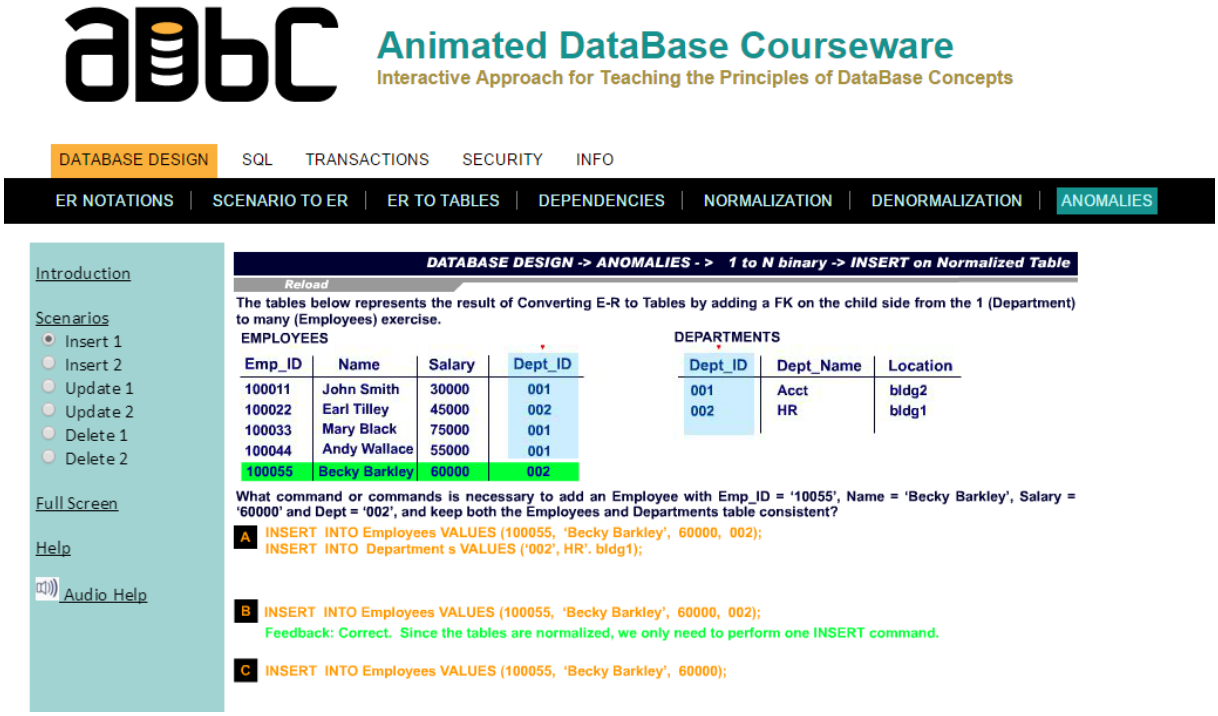


Joonis 15. Rüütelmaa, 2007. Normaliseerimine, esimene normaalkuju

Positiivse külje pealt võib öelda, et animatsioon on väga informatiivne. Iga normaalkuju puhul on öeldud, mis reeglite alusel normaliseerimine käib ning juuresolevad tabelid illustreerivad seda hästi.

6.2 Ingliskeelsed animatsioonid

Otsingu *„normalization animation“* tulemusena leidsin lisaks mitmetele videotele ja esitlustele ühe leheküle, kus on animatsioonid normaliseerimise kohta. See on Animated DataBase Courseware veebileht, mille leiab aadressilt http://adbc.kennesaw.edu/. Jaotuse *database design* alt leiab kategooriad *normalization* ja *anomalies*, mis mõlemad räägivad normaliseerimisest. Kui neid aga lähemalt vaadata, siis selgub, et kategooria *normalization* all ei ole ühtegi animatsiooni. Animatsioon on liikumise simulatsioon, mis saavutatakse hulga piltide või kaadrite näitamisega (Webopedia 2015). Antud lehel toimuvad kõik muudatused aga hetkeliselt. Kategooria *anomalies* all on see eest animeeritud ainult ülesannete sissejuhatused.



Joonis 16. Animated DataBase Courseware, 2015 Anomalies, Insert 1

Antud veebilehel olevad animatsioonid on samuti tehtud Flashiga, seega ei saa ka neid kasutada seadmetel, mis Flashi ei toeta. Küll ei ole seal piiranguks ekraani suurus, sest vajadusel saab ka mõlemas suunas kerida.

Üldiselt on tegu väga õpetliku veebilehega. Seal on palju erinevaid ülesandeid andmebaaside normaliseerimise kohta, kuid veebilehe nimele vaatamata ei ole tegemist animatsioonidega.

7. Kokkuvõte

Antud bakalaureusetöö eesmärgiks oli luua animatsioonid andmebaasi normaliseerimise kohta. Animatsioonid loodi kasutades jQuery teeki. Erinevaid normaalkujusid on mitmeid, kuid oma bakalaureusetöös keskendusin neist esimesele kuni kuuendale normaalkujule ning Boyce/Coddi normaalkujule.

Loodud normaliseerimise animatsioonid põhinevad SQL-andmebaasidel.

Töö tegemise käigus valmisid järgnevad vahetulemused, mis on esitatud antud töö peatükkidena:

- Lühiülevaade normaliseerimisest ja normaalkujudest.
- Ülevaade jQuery ja JQuery UI teekidest ja nende kasutusvõimalustest, samuti muudest kasutatud raamistikest ja tehnoloogiatest.
- Animatsioonide põhimõtete kirjeldamine UML abil ja animatsioonide realiseerimine. Loodud animatsioonide sisu saab failis andmete muutmisega muuta ja seega on need failid häälestuvad.
- Võrdlus teiste samateemaliste animatsioonidega.

SQL-andmebaasi tabelite normaliseerimine on mitmesammuline protsess, mille käigus teisendatakse tabeleid, viies neid üha kõrgematele normaalkujudele. Normaalkuju kujutab endast hulka reegleid. Tabel on mingil normaalkujul, kui see rahuldab selle normaalkuju reegleid.

jQuery on JavaScripti teek, millele on sisse ehitatud mitmeid animatsioonide tegemist lihtsustavaid meetodeid. jQuery UI lisab jQueryle veel lisavõimalusi animeerimiseks.

Teisi samateemalisi animatsioone õnnestus leida kahelt aadressilt: http://adbc.kennesaw.edu/ ja http://viktor.ld.ttu.ee/db_animatsioonid/4.html. Selgus, et need mõlemad on tehtud Flashiga. See aga tähendab, et seadmetest, mis Flashi ei toeta, ei ole võimalik neid animatsioone vaadata. jQuery töötab aga kõigi seadmete peal, kus on JavaScripti toetav brauser. Enamus brausereid seda ka teevad.

Kõik püstitatud eesmärgid said täidetud. Töö kirjutamise käigus sain rakendada oma olemasolevaid teadmisi ning ühtlasi õppisin ka nii mõndagi uut. Valmisid animatsioonid, mida saab kasutada normaliseerimise õppimiseks ja õpetamiseks.

Kindlasti võib antud tööd veel edasi arendada. Antud töös loodi animatsioonid seitsmele normaalkujule, kuid neid on tegelikkuses veel rohkem. Lõputöö sai üles ehitatud nii, et animatsioonides kasutatud andmeid oleks kerge muuta või täiendada. Töö edasiarenduseks võiksid olla ka animatsioonid selliste andmete liiasuste kohta, mida normaliseerimine ei aita kõrvaldada ning samuti ortogonaalse disaini printsiibi kohta, mis kontrollib andmete liiasust üle erinevate tabelite. Lisaks võib luua animatsioone denormaliseerimise kohta baastabelite tasemel ja vaadete tasemel. Samuti oleksid huvitavad animatsioonid, mis üritavad normaliseerimise põhimõtteid rakendada muud tüüpi süsteemide korral, näiteks infosüsteemid ja nende dokumentatsioon. Lisaks oleks huvitav luua töös tutvustatud põhimõtetel animatsioone, mis võtavad lähteandmed otse andmebaasist, kus kasutajad neid omakorda mingi rakenduse vahendusel muuta saaksid.

Valminud animatsioonid leiab aadressilt: http://viktor.ld.ttu.ee/animation_db_normalization/.

Summary

The goal of this bachelor's thesis was to create animations about database normalization. Animations were created using jQuery library. There are many different normal forms, but in my bachelor's thesis, I focused on the first to sixth normal form and Boyce/Codd normal form.

The created animations are based on SQL databases.

The main results, corresponding to the main chapters of the thesis, are the following:

- A short overview on normalization and normal forms.
- An overview of jQuery and jQuery UI along with other libraries and technologies used.
- Description of the principles of animations by using UML and implementation of the animations. One can change the content of animations by changing the content in a file and thus the animations are adaptable.
- Comparison with other animations in the same field.

Normalization in SQL databases is a multistep process during which tables are transformed and taken to higher normal forms. Each normal form is a set of rules and a table is on that normal form when it meets those rules.

jQuery is a JavaScript library that comes with various built-in methods for creating animations. jQuery UI extends jQuery with even more options for making animations.

Similar animations were found from two locations: http://adbc.kennesaw.edu/ and http://viktor.ld.ttu.ee/db_animatsioonid/4.html. Comparison with these animations revealed that the other animations are created with Flash, meaning they are not usable on devices that do not support Flash. jQuery works on any device as long as it has a browser that can display webpages that use JavaScript. Most browsers nowadays do.

All goals that were set were also completed. During the process of making this thesis I got to exercise my existing skills as well as gain new skills and knowledge. The animations created can be used both for teaching and studying individually.

The work definitely cannot be considered done yet. Animations were created for seven normal forms, but in reality there are more normal forms in need of explanative animations. The thesis was built so that it would be easy to update or add new data to the animations.

This thesis could be further extended by adding animations about data redundancy that cannot be removed by normalization and also about principles of orthogonal design, which controls data redundancy across various tables. Additionally, animations could be made about denormalization on both base tables and views. It would also be interesting to have animations that attempt to use the principles of normalization on other types of systems, for example, information systems and their documentation. Moreover, it would be interesting to create animations, based on practices introduced in the work, that take data directly from a database where users could change it through some application.

The animations are available at: http://viktor.ld.ttu.ee/animation_db_normalization/.

Kasutatud kirjandus

1. Anchor Modeling (2015) – [*Online*] http://www.anchormodeling.com/ (21.05.2015)
2. AngularJS (2015) – [*Online*] https://angularjs.org/ (15.05.2015)
3. BuiltWith (2015) *JavaScript Usage Statistics* – [*Online*] https://trends.builtwith.com/javascript (10.05.2015)
4. Chapple, E. *Database Normalization Basics* (2015) – [*Online*] http://databases.about.com/od/specificproducts/a/normalization.htm (09.05.2015)
5. Date, C.J. (2005). *Database in Depth* – [*Online*] ProQuest Tech Books (25.02.2015)
6. Date, C.J., Darwen, H., Lorenzos, N.A. (2002). *Temporal Data and the Relational Model: a detailed investigation into the application of interval and relation theory to the probleem of temporal database management*. Morgan Kaufmann.
7. Eessaar, E. (2014). *Andmebaasi loogilise disaini tulemuse parandamine ja headuse kontrollimine*. – [*Online*] õppekeskkond Maurus (09.05.2015)
8. Fotache, M. (2006) *Why Normalization Failed to Become the Ultimate Guide for Database Designers?* – [*Online*] http://papers.ssrn.com/sol3/papers.cfm?abstract_id=905060 (25.02.2015)
9. jQuery (2015) – [*Online*] https://jquery.com/ (11.05.2015)
10. jQuery API Documentation (2015) – [*Online*] https://api.jquery.com/ (14.05.2015)
11. jQuery UI (2015) – [*Online*] https://jqueryui.com/ (13.05.2015)
12. Less (2015) – [*Online*] http://lesscss.org/ (14.05.2015)
13. Rüütelmaa, K (2007) *Animatsioonid andmebaaside loogilisest disainist*. Bakalaureusetöö. TTÜ Informaatikainstituut.
14. Vallaste, H. (2015) *e-Teatmik Vallaste* – [*Online*] http://www.vallaste.ee/ (14.05.2015)
15. Webopedia (2015) – [*Online*] http://www.webopedia.com/ (13.05.2015)
16. Wikipedia (2015a) *Comparison of web browsers* – [*Online*] https://en.wikipedia.org/wiki/Comparison_of_web_browsers#JavaScript_support (13.05.2015)
17. Wikipedia (2015b) *Design science (methodology)* – [*Online*] https://en.wikipedia.org/wiki/Design_science_%28methodology%29 (21.05.2015)