TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Informaatikainstituut
Infosüsteemide õppetool

PostgreSQL´is kaskokindlustuse infosüsteemi loomine

Bakalaureusetöö

Üliõpilane: Harald Kosk
Üliõpilaskood: 094078IABB
Juhendaja: Raul Liivrand

Tallinn
2014

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

Käesolev töö uurib PostgreSQL´is kaskokindlustuse infosüsteemi loomise võimalusi. Töö tulemusena valmib funktsioneeriv andmebaas, mis laseb sõlmida kaskokindlustuse lepinguid ja sooritada lepinguga seotud protseduure nagu näiteks lepingu pikendamine või kuumakse arvutamine tulevikus kehtivate soodustustega.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 39 leheküljel, 5 peatükki, 5 joonist, 14 tabelit.

Abstract

The aim of this Project ist to create a casco insurance infosystem. By the end of this Project a functional database will be created, that will allow the creation of casco insurance contracts, perform tasks related to the contract. For example prolonging the contract or calculating the monthly payment amount in the futuure with all the discounts active then.

The thesis is in estonian and contains 39 pages of text, 5 chapters, 5 figures, 14 tables.

Lühendite ja mõistete sõnastik

PostgreSQL	***PostgreSQL*** Objekt-relatsiooniline andmebaasisüsteem
MySQL	***MySQL*** Objekt-relatsiooniline andmebaasisüsteem
Oracle	***Oracle*** Objekt-relatsiooniline andmebaasisüsteem
Sequence	***Sequence*** Funktsioon, mille eesmärk on tabelile unikaalsete identifikaatorite genereerimine
Serial	***Serial*** PostgreSQL genereerib sellisele andmetüübile ise tabelipiires unikaalse väärtuse
Character Varying	***Character Varying*** Varieeruva pikkusega tähemärkidest koosnev andmetüüp
Double Precision	***Double Precision*** Komakohaga arvuline andmetüüp
Real	***Real*** Komakohaga arvuline andmetüüp
Smallint	***Small Integer*** Lühike arvuline andmetüüp
Date	***Date*** Andmetüüp, mis mõeldud kuupäevade käsitlemiseks

Character	***Character*** Kindla pikkusega tähemärkidest koosneb andmetüüp
Schema	***Schema*** Andmebaasis teatud haldusstruktuur. Kasutatakse õiguste piiramise lihtsustamiseks ja ka muudel eesmärkidel
Boolean	***Boolean*** Binaarne andmetüüp
Best Practice	***Best Practice*** Parimad tavad. Arendusel jälgitavad soovituslikud juhised
Select	***Select*** Käsklus andmete pärimiseks tabelist või mõnest muust objektist.
FK	***Foreign Key*** Välisvõti. Veerg tabelis, mis viitab mõne teise tabeli primaarvõtmele.
Floor	***Floor*** Funktsioon, mis ümardab arvu alla, lähima täisarvuni.
Extract	***Extract*** Funktsioon, mis eraldab kuupäevast, kindla ajaühiku (aasta, kuu, päev).
From	***From*** Tähistab, kust tabelist või objektist andmete päring tehakse.
PK	***Primary Key*** Primaarvõti. Tabeli piires unikaalne identifikaator, mille järgi saab lihtsalt tabelist otsida konkreetse rea.

Jooniste nimekiri

Tabelite nimekiri

Sisukord

1. Sissejuhatus

Andmebaasidega puutuvad inimesed kokku igapäevaselt rohkem kui algul paista võib. Näiteks poes kaardiga makstes või internetis arveid tasudes. Andmebaasid pakuvad mugava ja efektiivse viisi andmete salvestamiseks, töötlemiseks ja analüüsimiseks. Enamus suuremaid organisatsioone kasutavad andmebaase oma klientide andmete haldamiseks. Lisaks pakuvad osad andmebaasid ka võimalust neid andmeid töödelda, analüüsida ja kasutada endale vajalikul kujul. Kõige levinuim andmebaasiliik on relatsiooniline andmebaas ja sellise ka valisin selle projekti jaoks.

Kaskokindlustuse andmebaasi valisin teemaks üsna juhuslikult. Otsisin teemat, mis oleks keerukam kui tavaline veebipood ja millel oleks tulevikus ruumi edasiarenduseks. Selleks sobiks kindlustuse teema hästi ja et hoidmaks teemat liiga laiaks valgumast valisin ühe kindla kindlustuse alamliigi.

1.1 Ülesande püstitus

Töötava kaskokindlustuse infosüsteemi andmebaasi osa loomine PostgreSQL´is.

1.2 Metoodika

PostgreSQLi kasutades luuakse infosüsteemi andmebaas, mis koosneb andmebaasist ja seal olevatest funktsioonidest ja tabelitest. Samuti on vaja lühikest ülevaadet antud infosüsteemi edasiarendamise võimalustest.

2. Ülevaade kindlustuse süsteemi funktsionaalsusest

Antud infosüsteem vajab töötamiseks mitmeid funktsioone. Osade eesmärk on lihtsalt andmete sisestamine, mõned arvutavad lisaks ka kindlustusmakse koefitsienti ja mõned keerulisemad kutsuvad omakorda välja teisi protseduure.

Süsteemi funktsioonid:

1. Pankade andmete haldus (lisamine, muutmine).
2. Kasutajate andmete haldus (lisamine, muutmine).
3. Auto koefitsientide arvutus.
4. Kliendi koefitsientide arvutus.
5. Lepingu koefitsientide arvutus.
6. Lepingu pikendamine ja sõlmimine.
7. Tulevikus kliendi kuumakse ja koefitsiendi arvutus.

2.1 Funktsioonid

Lisa_pank – Funktsioon võtab sisenditeks panganime ja kontonumbri ja pank_pank_id_seq sequenceist võtab endale unikaalse identifikaatori ja lisab need tabelisse pank.

Lisa_aadress – Funktsioon võtab sisenditeks riiginime ja maakonna, linna, aadressi, postiindeksi ja lisab need tabelisse aadress. Seejärel aadress_aadress_id_seq sequenceist võtab funktsioon sisestatud aadressi identifikaatori aadress_id ja tagastab selle väärtuse.

Lisa_kasutaja – Funktsioon võtab sisenditeks kasutajanime ja parooli ja lisab need tabelisse kasutaja_info. Seejärel kasutaja_info_kasutaja_id_seq sequenceist võtab endale sisestatud kasutaja identifikaatori kasutaja_id ja tagastab selle.

Arvuta_autokoef – Funktsioon tegeleb auto koefitsiendi arvutamisega. Ta võtab sisenditeks mudeli_id, liigi_id, turva_id ja auto väärtuse i_vaartus. Mudel_id järgi otsib ta auto_mudel tabelist vastava mudeli koefitsiendi mudel_koefitsient. Liik_id järgi otsib ta auto_liik tabelist vastava autoliigi koefitsiendi liik_koefitsient. Turva_id järgi otsib ta turva_seaded tabelist vastava turvasüsteemi koefitsiendi turva_koefitsient, turva seadete puudumisel on algväärtuseks võetud turva_id 1, mis tagastab soodustuse koefitsiendi 0.0. Seejärel arvutatakse auto koefitsient kasutades valemit:

auto_koefitsient = mudel_koefitsient * liik_koefitsient – turva_koefitsient + i_vaartus / 100000

Seejärel tagastab ta selle arvutatud väärtuse funktsiooni väljakutsujale.

Lisa_auto – Funktsiooni ülesanne on auto lisamine ja tema koefitsiendi arvutamine. Sisenditeks on tal mudeli id, autoliigi id, turva id, auto numbrimärk, auto väärtus ja auto registreerimis aasta.

Kõigepealt kutsub funktsioon välja teise funktsiooni , arvuta_autokoef, et leida vastava auto koefitsient. Seejärel sisestab ta kõik andmed tabelisse auto. Seejärel võtab ta auto_auto_id_seq sequenceist sisestatud auto identifikaatori ja tagastab selle funktsiooni kutsujale.

Arvuta_kliendikoef – Funktsioon tegeleb kliendi koefitsiendi arvutusega. Sisenditeks võtab ta sünnikuupäeva synniaeg ja viimase õnnetuse kuupäeva „õnnetus“. Kliendi koefitsiendi algväärtuseks on 1. Kõigepealt arvutab funktsioon kliendi vanuse leides sünniaja ja praeguse kuupäeva vahe aastates. Seejärel arvutab funktsioon aja, mis on möödunud kliendil viimasest õnnetusest. Selleks leiab ta praeguse kuupäeva ja viimase õnnetuse vahe aastates. Seejärel arvutatakse kliendi koefitsient kasutades valemit:

kliendi_koefitsient = kliendi_koefitsient - aeg_õnnetusest / 10

Need soodustused käivad kõik hetkel staaži soodustuste kohta. Kui kliendil pole ühtegi õnnetust toimunud 5 aastat, siis saab ta kätte soodustuse 50%, mis annab kliendi koefitsiendiks 0.5, mis on ka maksimaalne võimalik kliendi soodustus. Suurema soodustuse puhul määratakse kliendi koefitsiendiks 0.5. Samuti on piiratud ka kliendi maksimaalne õnnetuse põhine koefitsient , mille suurim väärtus on 1.0, seega sellest ei saa kuidagi kliendi koefitsient suuremaks minna, küll aga saab see koefitsient kasvada sõltuvalt kliendi vanusest.

Seejärel arvutatakse kliendi koefitsient sõltuvalt kliendi vanusele. Kui klient on vanuses 30 – 50 aastat, siis ta arvestatakse kõige väiksema riskiga gruppi ja tema koefitsient ei muutu. Kui kliendi vanus on alla 20, siis ta arvestatakse kõige suurema riskiga gruppi ja koefitsient korrutatakse kolmega. Kui klient on vanuses 20 – 29 või üle 50 aasta, siis arvestatakse ta keskmisesse riskigruppi ja koefitsient korrutatakse kahega.

Seejärel tagastab funktsioon arvutatud koefitsiendi funktsiooni väljakutsujale.

Lisa_klient – Funktsiooni ülesanne on kliendi andmete sisestamine. Sisendid on i_kasutajanimi, i_parool, i_riik, i_maakond, i_linn, i_aadress, i_postiindeks, i_panganimi, i_pangakonto, i_synniaeg, i_nimi, i_isikukood. Kõikidele uutele klientidele määratakse alguseks roll id-ga 1, ehk tavaklient. Hetkel teisi rolle süsteemis pole, aga süsteemi kasvades on neid kindlasti vaja. Funktsioon kutsub välja funktsiooni arvuta_kliendikoef, et saada kliendi koefitsient. Seejärel kutsub ta välja lisa_kasutaja, lisa-aadressi ja lisa panga ja nendest kõigist saab tagasi lisatud andmete vastavad id-d. Seejärel lisatakse andmed ja uued id-d tabelisse klient. Peale seda tagastab funktsioon klient_klient_id_seq sequenceist saadud kliendi id.

Lepingu_autokoef – Funktsiooni eesmärk on kogu lepingu koefitsiendi arvutamine. Sisenditeks on auto id, kliendi id ja kindlustuskaitse id. Esmalt kogub funktsioon kokku sisenditeks olevate id-de järgi auto, kliendi ja kaitse koefitsiendid vastavalt siis auto, kliendi ja kindlustuskaitse_liigid tabelitest. Siis kontrollitakse kas hetkel lepingut sõlmides on mõni kehtiv soodustus ja kui on siis leiab selle. Soodustuse puududes määratakse soodustuse koefitsiendiks lihtsalt 0.0.

Seejärel arvutatakse kogu lepingu koefitsient kasutades valemit:

lepingu_koefitsient = m_auto_koefitsient * klient_koefitsient * kaitse_koefitsient - m_soodustus

Seejärel tagastab funktsioon saadud koefitsiendi.

Lisa_leping – Selle funktsiooni ülesanne on lepingu sõlmimisel selle andmete salvestamine ja kuumakse arvutamine. Sisenditeks on tal kliendi id, auto id, kaitse id, alguskuupäev ja lõppkuupäev. Funktsioonis on lepingu kuumakse algväärtuseks 25 eurot. Sõlmitava lepingu algolekuks on ’A’, mis märgib et leping on kehtiv. Süsteemi arenedes oleks vaja kindlasti olekuid lisada.

Esmalt arvutab funktsioon lepingu koefitsiendi, selleks kutsub ta välja lepingu_autokoefi. Seejärel arvutatakse lepingu kuumakse kasutades valemit;

m_kuumakse := m_baaskuumakse * m_koefitsient

Seejärel lisatakse lepingut andmed tabelisse leping.

Funktsioon tagastab sisestatud lepingu id, mille ta saab leping_leping_id_seq sequenceist.

Pikenda_leping – See funktsioon lubab lepingut pikendada. Sisenditeks on tal kliendi id, auto id, kaitse id ja lõppkuupäev. Ta kasutab samu algväärtusi, mida lisa_leping: baas kuumakse 25 eurot ja lepingu olek 'A'. Kõigepealt kutsub ta välja lepingu_autokoefi, et leida uus lepingu koefitsient. Ja selle järel arvutab ta uue kuumakse kasutades sama valemit, mida lepingu lisamisel kasutati

```
m_kuumakse := m_baaskuumakse * m_koefitsient
```

Seejärel uuendatakse tabelit kindlustus, kus uuendatakse tulpasi kaitse_id, lopp, kuumakse ja olek, sest need võivad olla uuenduse käigus muutunud.

Kui kõik töötas tagastab funktsioon väärtuse 1.

Tuleviku_koef – See funktsioon lubab olemasolevatel klientidel vaadata oma kindlustusmakse suurust eri tingimustel. Tema abil saad vaadata kuidas kuumakse muutub kui kliendi vanus muutub ja ta liigub ühest riskigrupist teise, veel lubab ta arvutada kuumakse, kui klient pole pikka aega üheski õnnetuses või siis just kui on õnnetus ja kuumakse kasvab.

Sisenditeks on tal auto_id, kliendi_id, kaitse_id ja „i_õnnetus".

Baaskuumakseks on endiselt 25 eurot. Auto tabelist leitakse id järgi auto koefitsient. Kliendi id järgi otsitakse klient tabelist kliendi sünniaeg ja seejärel arvutatakse kliendikoefitsient kasutades arvuta_kliendikoef funktsiooni.

Kindlustuskaitse_liigid tabelist leitakse kaitse_id järgi talle vastav koefitsient ja seejärel kontrollitakse kas on olemas mõni kehtiv soodustus tabelis soodustus. Soodustuse puudumisel on soodustuse koefitsient 0.0.

Seejärel leitakse koefitsient valemiga:

```
m_lepingu_koefitsient = m_auto_koefitsient * m_klient_koefitsient * m_kaitse_koefitsient - m_soodustus
```

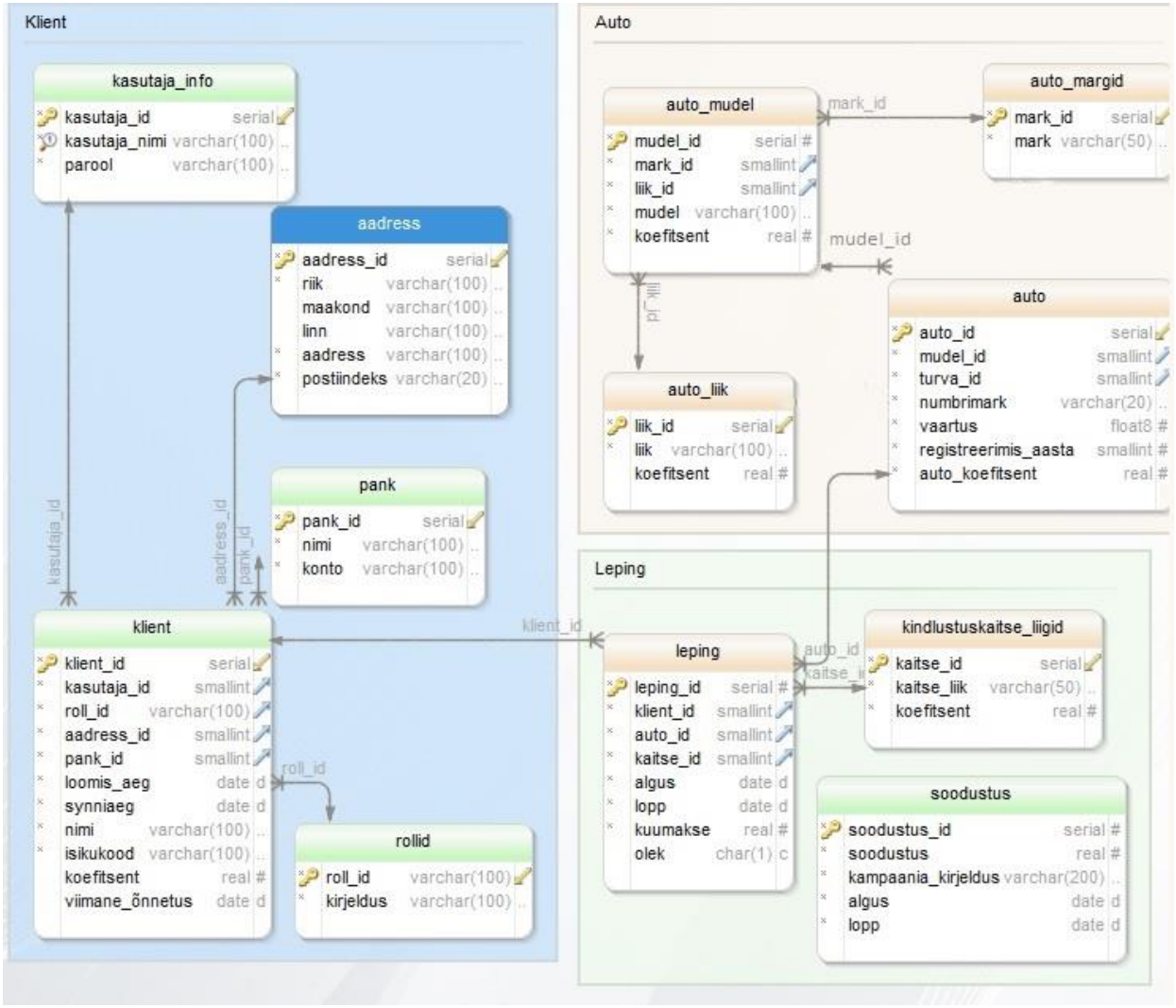
Ja arvutatakse kuumakse valemiga:

```
m_kuumakse := m_baaskuumakse * m_lepingu_koefitsient
```

Seejärel tagastab funktsioon saadud uue kuumakse väärtuse.

3. Andmevaade

3.1 Andmebaasi disain



Joonis 1. Andmebaasiskeem

- Kasutaja_info – Tabeli eesmärk on registreeritud kasutaja kasutajanime ja parooli hoidmine.
- Aadress – Tabel hoiab registreeritud kasutaja elukoha aadressi.
- Pank – Tabel hoiab registreeritud kasutaja panga andmeid.
- Rollid – Hoiab kliendi rolli ja selle kirjeldust.

- Klient – Võtab kokku teistest tabelitest kliendi andmed läbi foreign keyde ja hoiab infot kliendi nime, sünniaja ja viimase õnnetusaja ja kliendi koefitsiendi kohta.
- Auto_margid – Hoiab automargi kohta infot.
- Auto_liik – Hoiab infot eri autoliikide ja nende koefitsientide kohta.
- Auto_mudel – Võtab kokku auto loogi ja margi info ja koefitsiendi.
- Auto – Võtab kokku auto kohta olemasoleva informatsiooni, koos auto numbrimärgi, väärtuse ja registreerimis aastaga ja auto kohta arvutatud koefitsiendiga.
- Kindlustuskaitse_liigid – Hoiab erinevaid kaitseliike, mis auto kasutusala kohta käivad (Sõiduauto, takse, jne) ja nende koefitsienti.
- Soodustus – Hoiab informatsiooni soodustuste ja nende kehtivusaegade kohta.
- Leping – Võtab kokku kõik lepingu sõlmimiseks vajaliku info eri tabelite välisvõtmetega. Samuti lepingu kehtivusaeg, kuumakse ja olek.

Protseduuride loomisel üritasin neid luua võimalikult modulaarseteks, et neid oleks hiljem vajaduse korral lihtne uuesti kasutada. Osa nendest funktsioonidest saigi juba kasutaud mitmes teises protseduuris.

Tabelite disainides katsusin hoida andmeid võimalikult lahus, et vältida seda et tekivad üksikud kümnete parameetritega tabelid, mida on hiljem raske hoomata ja kasutada. Samuti teeb see õiguste piiramise lihtsamaks, et funktsioonid ja kasutajad ei omaks õigusi liiga suure andmehulga üle.

Protseduure ja tabeleid disainides püüdsin koguaeg arvestada sellega, et süsteemi oleks tulevikus võimalikult lihtne laiendada. Tabelite ja funktsioonide moduleerimine aitab sellele kindlasti kaasa

Kõikides protseduurides, mis lisavad tabelitesse andmeid viitasin tabelites olevatele väljadele nimeliselt. Selle eesmärk on see, et kui tulevikus tabelitesse tuleb tulpasi juurde, siis see ei tee katki olemasolevaid protseduure ja neid on võimalik kerge vaevaga täiendada.

3.2 Tabelite kirjeldused

Tabel 1. Aadress

Veeru nimi	Tüübinimi	Kommentaar
Aadress_id**(PK)**	serial	Automaatselt genereeritav identifikaator
Riik	character varying	Riik, kust klient on pärit
Maakond	character varying	Kliendi maakond
Linn	character varying	Kliendi linn
Aadress	character varying	Kliendi elukoha aadress(tänavanimi koos maja ja korterinumbriga)
Postiindeks	character varying	Kliendi postiindeks

Tabel 2. Auto

Veeru nimi	Tüübinimi	Kommentaar
Auto_id**(PK)**	serial	Automaatselt genereeritav identifikaator
Mudel_id**(FK)**	smallint	Viide vastava auto mudelile **(kindlustus.auto_mudel.mudel_id)**
Turva_id**(FK)**	smallint	Viide vastava auto turvaseadetele **(kindlustus.turva_seaded.turva_id)**
Numbrimark	character varying	Auto numbrimärk
Vaartus	double precision	Auto väärtus kindlustuse sõlmimise hetkel
Registreerimis_aasta	smallint	Auto esmase registreerimise aasta

Auto_koefitsient	real	Auto koefitsient, mida kasutatakse lepingu kuumakse arvutamisel

Tabel 3. Auto_liik

Veeru nimi	Tüübinimi	Kommentaar
Liik_id**(PK)**	serial	Automaatselt genereeritav identifikaator
Liik	character varying	Auto liik (sõiduauto, mootorratas, veoauto)
Koefitsient	real	Auto liigi koefitsient, mida kasutatakse auto mudeli koefitsiendi arvutamiseks

Tabel 4. Auto_margid

Veeru nimi	Tüübinimi	Kommentaar
Mark_id**(PK)**	serial	Automaatselt genereeritav identifikaator
Mark	character varying	Auto margi nimetus

Tabel 5. Auto_mudel

Veeru nimi	Tüübinimi	Kommentaar
Mudel_id**(PK)**	serial	Automaatselt genereeritav identifikaator
Mark_id**(FK)**	smallint	Viide vastava automudeli margile **(kindlustus.auto_margid.mark_id)**
Liik_id**(FK)**	smallint	Viide vastava automudeli liigile **(kindlustus.auto_liik.liik_id)**
Mudel	character varying	Automudeli nimetus
Koefitsient	real	Auto mudeli koefitsient, mida kasutatakse

		auto koefitsiendi arvutamiseks

Tabel 6. Kasutaja_info

Veeru nimi	Tüübinimi	Kommentaar
Kasutaja_id**(PK)**	serial	Automaatselt genereeritav identifikaator
Kasutaja_nimi	character varying	Kasutajanimi
Parool	character varying	Kasutaja parool

Tabel 7. Kindlustuskaitse_liigid

Veeru nimi	Tüübinimi	Kommentaar
Kaitse_id**(PK)**	serial	Automaatselt genereeritav identifikaator
Kaitse_liik	character varying	Vastava lepingu kaitse liik(Sõiduauto, takso, õppesõiduk)
Koefitsient	real	Kaitse koefitsient, mida kasutatakse lepingu koefitsiendi arvutamisel

Tabel 8. Klient

Veeru nimi	Tüübinimi	Kommentaar
Klient_id**(PK)**	serial	Automaatselt genereeritav identifikaator
Kasutaja_id**(FK)**	smallint	Viide kasutaja infole **(kindlustus.kasutaja_info.kasutaja_id)**
Roll_id**(FK)**	character varying	Viide kasutaja rollile **(kindlustus.rollid.roll_id)**

Aadress_id**(FK)**	smallint	Viide kasutaja aadressile **(kindlustus.aadress.aadress_id)**
Pank_id**(FK)**	smallint	Viide kasutaja panga andmetele **(kindlustus.pank.pank_id)**
Loomis_aeg	date	Kliendi registreerimise kuupäev
Synniaeg	date	Kliendi sünnikuupäev
Nimi	character varying	Kliendi nimi
Isikukood	character varying	Kliendi isikukood
Koefitsient	real	Kliendi koefitsient, vastavalt tema vanusele ja viimasest õnnetusest möödunud ajast
„Viimane_õnnetus“	date	Viimase õnnetuse kuupäev, milles jäi süüdlaseks klient. Algväärtuseks on lepingu sõlmimise kuupäev

Tabel 9. Leping

Veeru nimi	**Tüübinimi**	**Kommentaar**
Leping_id**(PK)**	serial	Automaatselt genereeritav identifikaator
Klient_id**(FK)**	smallint	Viide lepingu sõlminud kliendile **(kindlustus.klient.klient_id)**
Auto_id**(FK)**	smallint	Viide kindlustatavale autole **(kindlustus.auto.auto_id)**
Kaitse_id**(FK)**	smallint	Viide kindlustuskaitse liigile **(kindlustus.kindlustuskaitse_liigid.kaitse_id)**

Algus	date	Lepingu kehtivuse alguskuupäev
Lopp	date	Lepingu kehtivuse loppkuupaev
Kuumakse	real	Lepingu kuumakse, arvutatakse saadud lepingu koefitsiendi järgi
Olek	character	Lepingu olek, näitab kehtivust, algväärtuseks on ’A’

Tabel 10. Pank

Veeru nimi	Tüübinimi	Kommentaar
Pank_id**(PK)**	serial	Automaatselt genereeritav identifikaator
Nimi	character varying	Panga nimi, millega klient on seotud
Konto	character varying	Kliendi kontonumber pangas

Tabel 11. Rollid

Veeru nimi	Tüübinimi	Kommentaar
Roll_id**(PK)**	character varying	Kliendi rolli kirjeldus, hetkel olemas vaid tavaklient
Kirjeldus	character varying	Rolli ja tema õiguste ja eripära kirjeldus

Tabel 12. Soodustus

Veeru nimi	Tüübinimi	Kommentaar
Soodustus_id**(PK)**	serial	Automaatselt genereeritav identifikaator
Soodustus	real	Soodustuse suurus. Üldiselt jääb vahemikku

		0.0- 1.0
Kampaania_kirjeldus	character varying	Sooduskampaania lühikirjeldus
Algus	date	Kampaania alguskuupäev
Lopp	date	Kampaania alguskuupäev

Tabel 13. Turva_seaded

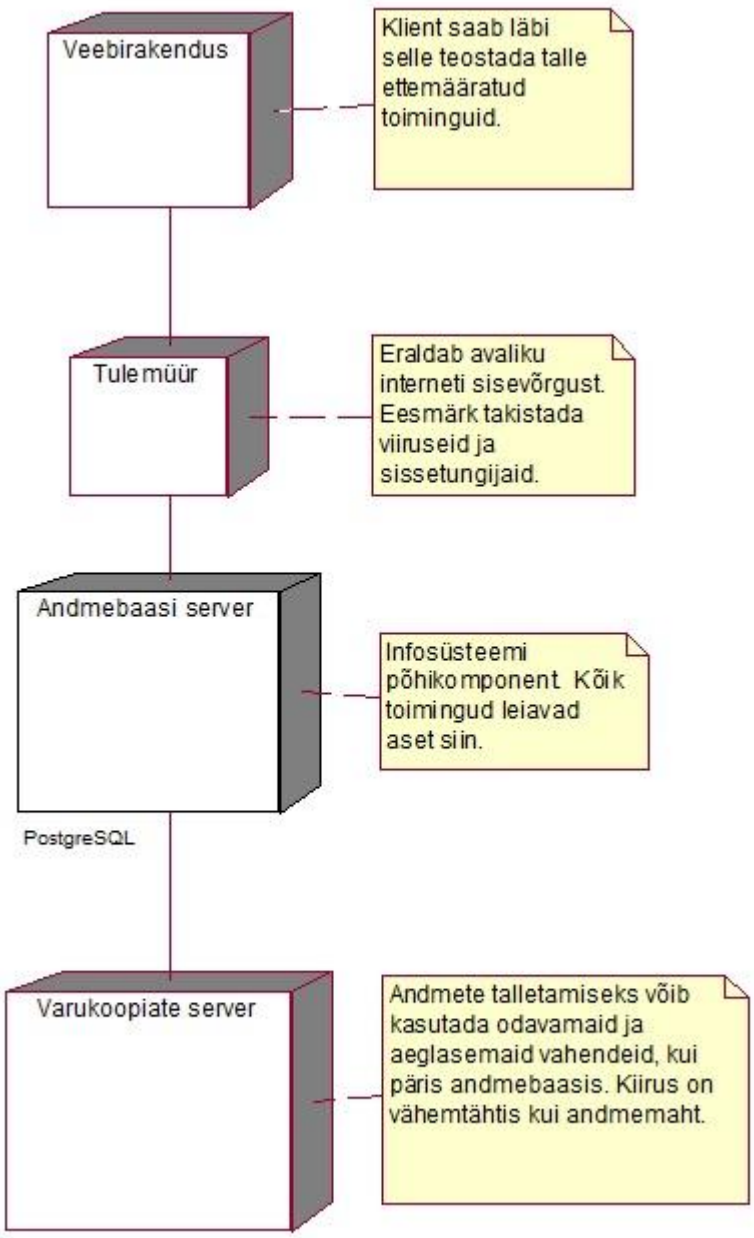
Veeru nimi	**Tüübinimi**	**Kommentaar**
Turva_id**(PK)**	serial	Automaatselt genereeritav identifikaator
Mudel	character varying	Turvaseadme mudel
Koefitsient	real	Turvaseadme koefitsient

4. Realisatsioon

4.1 Lahenduse arhitektuur

Põhilised funktsionaalsused on kõik seotud PostgreSQLi andmebaasi serveriga, mis on ka siin töös loodud. Süsteemi tööle rakendamisel võetakse lisaks kasutusele varukoopiate server, kuhu tuleks andmeid salvestada iga öö. Samuti peaks olemas olema arhiivi andmebaas, kuhu talletatakse andmed, mida pole enam aktiivselt vaja, nagu näiteks suletud lepingud, aga mida peab alles hoidma teatud perioodi seaduste tõttu, juhuks kui näiteks tekib vaidlusi teise poolega.

Klient peaks saama andmebaasile ligi ainult läbi vastava veebirakenduse, kus on tema tegevus täpselt piiratud. Sealsed võimalused piirduks põhiliselt registreerimisega, uue kaskolepingu sõlmimise ja vana pikendamise ja enda ja olemasolevate poliiside andmete vaatamisega.



Joonis 2. Arhitektuuri skeem

4.2 PostgreSQL ja tema omadused

PostgreSQL sai valitud selle projekti jaoks selle tõttu, et tal on olemas kõik vajalikud omadused, mida ühelt andmebaasisüsteemilt eeldada võib. Andmetüüpide valik on mitmekesine, ja kasutajal on väga lihtne luua just endale vajalikke protseduure. Osades teistes on funktsioonide loomine üsna piiratud ja raske täita vajalikku eesmärki. Ta on vabavarana jagatav ja avatud lähtekoodiga. Selle tõttu on ta üks laiemalt kasutatav andmebaasisüsteem ja ta uueneb pidevalt. Avatud lähtekood tagab selle, et vigade ilmnemisel saavad need üldiselt üsna kiirelt parandatud. Tema üheks heaks omaduseks on et ta ei nõua eriti palju ressursse.

Võrreldes MySQLiga on tal palju suurem funktsionaalsus ja üldiselt kipub ta ka olema stabiilsem. Kui MySQL lubab kasutajal luua protseduure, mis on sisuliselt tavalised päringud, millele on antud nimi ja mida saab hiljem välja kutsuda, siis on PostgreSQLi võimalused tunduvalt suuremad.

Oraclega võrreldes on tal palju sarnaseid omadusi ja üldiselt kipuvad nad jääma üsna võrdseteks. Oracle eeliseks võib olla veidi parem jõudlus suurte andmehulkade puhul, see on ka ilmselt põhjuseks miks väga suured ettevõtted kasutavad pigem Oraclet. Oracle miinuseks võib muidugi tuua selle, et ta on tasuline ja kinnise lähtekoodiga.

Andmebaasi kataloogide ja schemade olemasolu teeb suuremate andmebaaside ja nende õiguste halduse tunduvalt mugavamaks. MySql´is ja Oracles on neist kas üks või teine, aga PostgreSQL´is on olemas mõlemad võimalused.

Veel üheks valiku tegemisel otsustavaks punktiks sai see, et kuigi ka näiteks Oracle andmebaasil on olemas tasuta arendustööriist, siis isikliku kogemuse järgi on Oracle tööriist kehvavõitu ja parema tulemuse saamiseks oleks vaja mõni tasuline variant. PostgreSQL´iga kaasa tulev pgAdmin III on see eest jätnud üsna hea mulje ja tema kasutamine on üsna lihtne ja mugav, kuigi kindlasti mõningaid puudusi leidub.

Boolean väärtuse olemasolu andmetüübina, võib tunduda väikse asjana, kuid ta teeb osad protseduurid palju mugavamaks. Ta on olemas nii MySQL’is kui PostgreSQL’is, aga puudub Oraclest.

Lisaks põhjus miks valisin just PostgreSQLi on see, et kuna tööalaselt puutun kokku Oracle andmebaasiga, siis oli endal soov proovida midagi, millega kokkupuude on väiksem olnud ja

selleks sobis PostgreSQL väga hästi. Tema kasutamine kindlasti laiendas silmaringi ja andis uusi kogemusi.

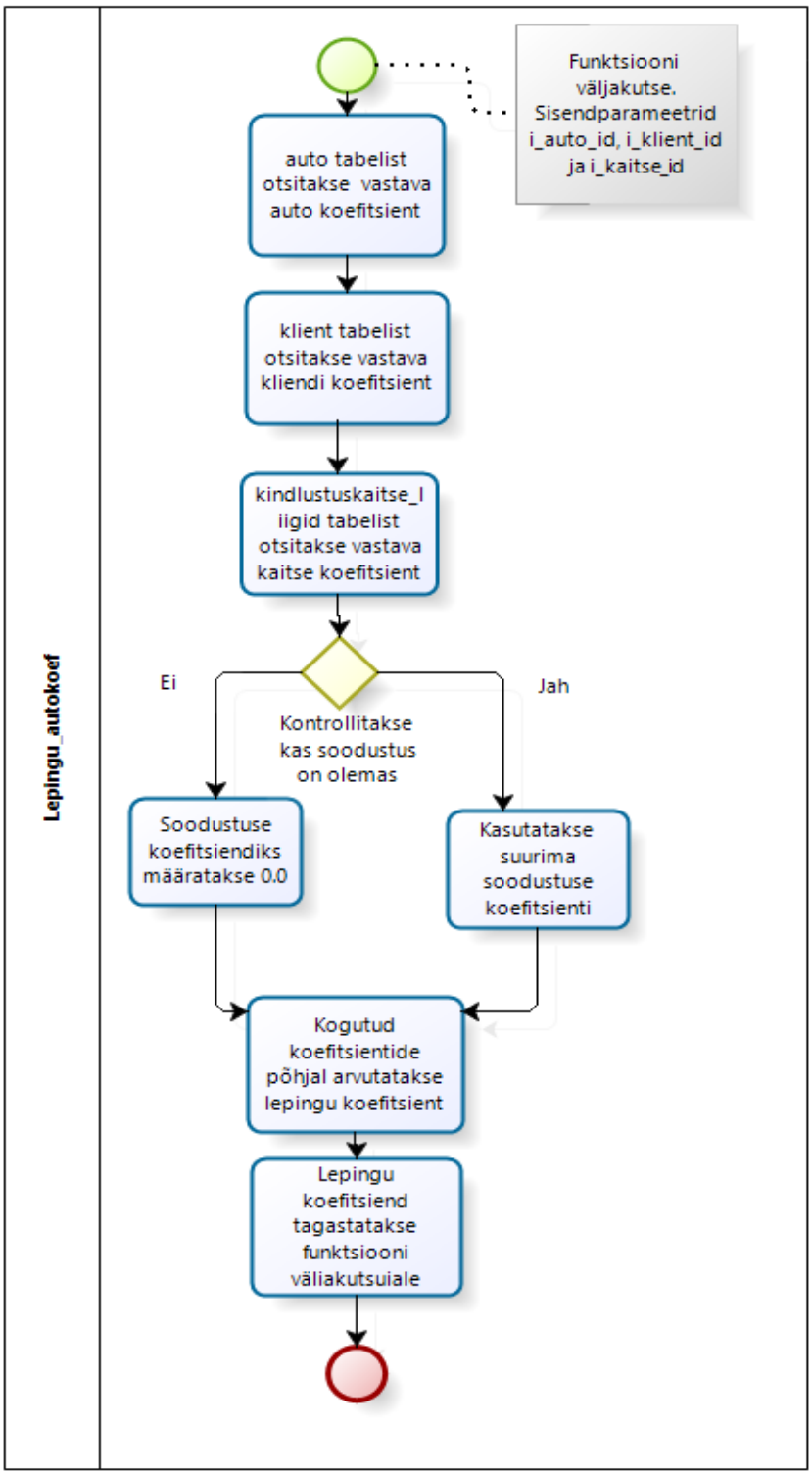
Toon tabelina välja osad erinevused kolme levinuma andmebaasisüsteemi vahel. Võrdlen selliseid punkte, mida läks vaja käesolevas projektis.

Tabel 14. Andmebaaside võrdlus

	PostgreSQL	Mysql	Oracle
Andmebaasi liik	Relatsiooniline	Relatsiooniline	Relatsiooniline
Avatud lähtekood	Jah	Jah	Ei
Kasutaja loodud funktsioonid	Suur funktsionaalsus, võimalik lisada eri keelte tuge	Piiratud funktsionaalsus	Suur funktsionaalsus, võimalik lisada eri keelte tuge
Boolean väärtus	Jah	Jah	Ainult PL/SQL keeles, mitte tabeli andmetüübina
Schemad	Jah	Ei	Jah
Andmebaasi kataloogid	Jah	Jah	Ei

4.3 Füüsiline disain

Toon välja mõne põhifunktsiooni kohta detailsema kirjelduse.



Joonis 3. Lepingu_autokoef tegevus diagramm

Lepingu_autokoef (i_auto_id integer, i_klient_id integer, i_kaitse_id integer)

Funktsiooni eesmärgiks on kindla lepingu jaoks kuumakse koefitsiendi arvutus. Olen püüdnud funktsioone luues jälgida PostgreSQLi best practice. Sellest tulenevalt on sellel funktsioonil piiratud õigused. Kuna tema eesmärk pole andmeid muuta, siis on tal ainult select õigused teatud tabelitele. Andmeid sisestada ega muuta ta ei saa.

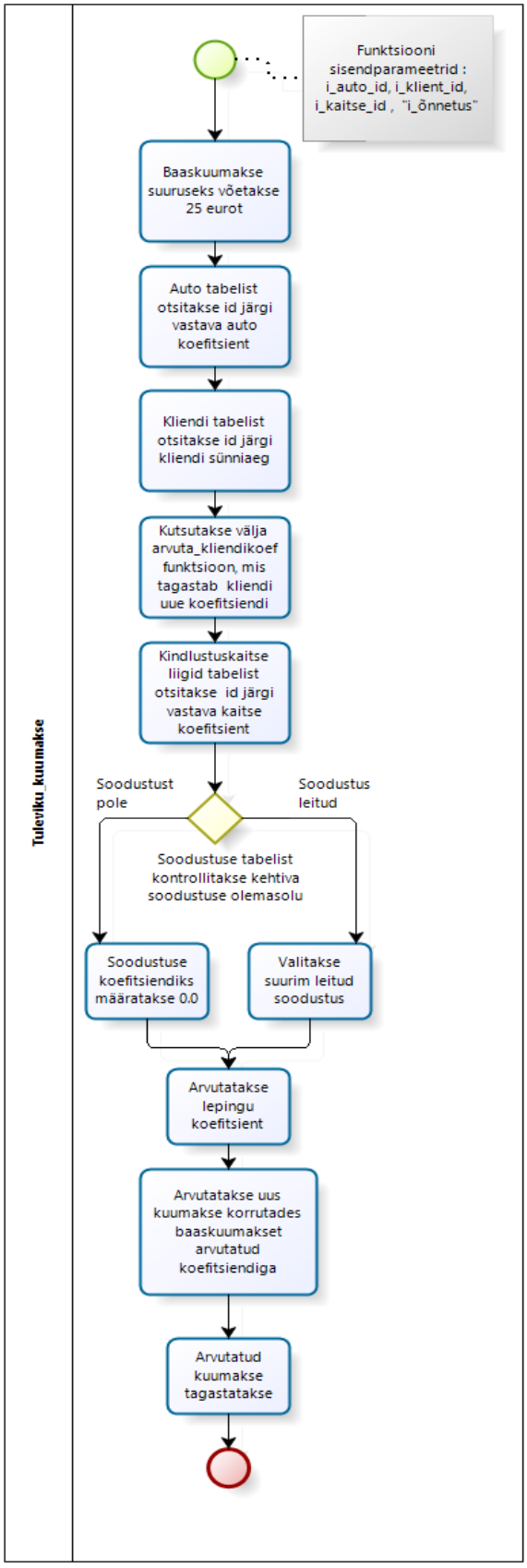
Funktsiooni töötab nii, et alguses kogub kokku eri tabelitest vastavad koefitsiendid, mis ta leiab üles sisendparameetriteks olevate väärtuste järgi.

Seejärel vaatab funktsioon, kas soodustuse tabelis on olemas hetkel kehtiv soodustus. Kui soodustust pole, määratakse soodustuse koefitsiendiks 0.0. Kui soodustusi on aga mitu, siis valitakse neist kõige suurem soodustus.

Seejärel arvutatakse lepingu koefitsiendi suurus vastavalt valemile :

```
lepingu_koefitsient = m_auto_koefitsient * klient_koefitsient * kaitse_koefitsient - m_soodustus;
```

Seejärel tagastab funktsioon arvutatud lepingu koefitsiendi suuruse (**lepingu_koefitsient**) enda väljakutsujale ja lõpetab tegevuse.

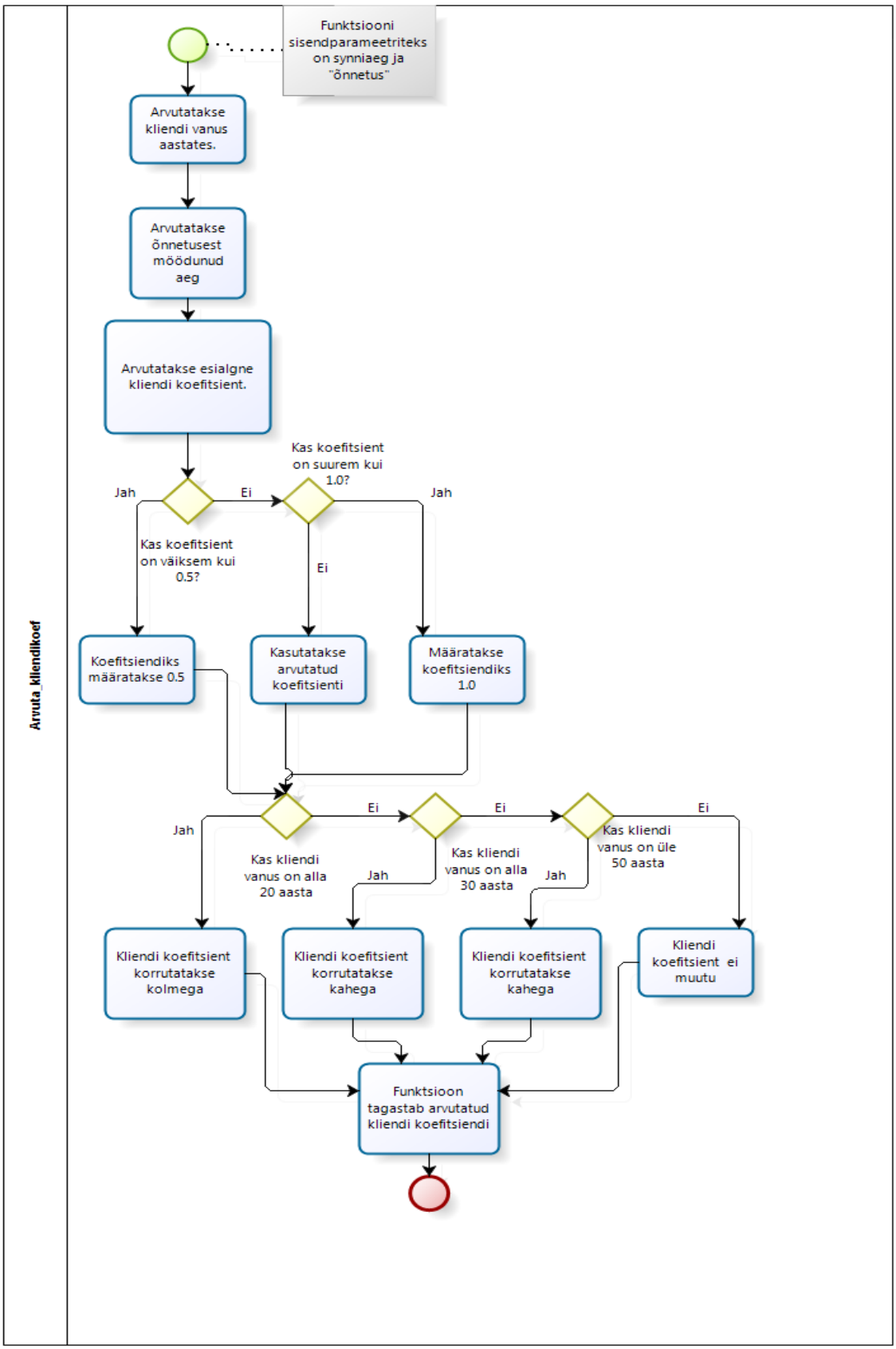


Joonis 4. Tuleviku_kuumakse tegevus diagramm

kindlustus.tuleviku_kuumakse(i_auto_id integer, i_klient_id integer, i_kaitse_id integer, "i_õnnetus" date)

Funktsioon lubab kasutajal arvutada, kuidas muutuks tulevikus tema kuumakse suurus, kui ta põhjustaks mingi õnnetuse, see kasvatab kuumakset. Samas lubab ta näha kuumakse muutust tulevikus, kui klient pole pikemat aega ühtegi õnnetust põhjustanud, sellega näitab klient et on usaldusväärne ja pole ettevõttele väga suureks riskiks ja seega tema kuumakse suurus väheneb.

Ka siin funktsioonis olen püüdnud jälgida best practice reegleid. Alates õiguste piiramisest funktsiooni ligipääsul andmetele ja nende muutmisele. Lisaks üritan hoida funktsioone võimalikult lihtsate ja modulaarsetena, et neid saaks lihtsalt vajaduse korral uuesti kasutada. See funktsioon omakorda kutsubki välja teist, varem loodud funktsiooni, et arvutada kliendi koefitsient uuesti.



Joonis 5. Arvuta_kliendikoef tegevus diagramm

kindlustus.arvuta_kliendikoef(synniaeg date, "õnnetus" date)

Funktsiooni eesmärk on kliendi põhise koefitsiendi arvutamine, kasutades kliendi sünniaega ja viimasest kliendi põhjustatud õnnetusest möödunud aega. Kliendi koefitsiendi arvutamisel võetakse baaskoefitsiendiks algväärtusena 1.

Seejärel arvutatakse kliendi vanus aastates. Kuna praegune kuupäev ja sisendparameeter synniaeg on mõlemad date formaadis, on see arvutamine lihtsustatud ja saab kasutada PostgreSQLi sisse ehitatud funktsioone. Kasutades Funktsiooni EXTRACT saab mõlemast kuupäevast eraldada vajalikud väärtused ja arvutada kliendi vanuse aastates.

Seejärel arvutatakse õnnetuses möödunud aeg kasutades valemit:

```
aeg_õnnetusest = FLOOR(EXTRACT(year FROM age(current_date,õnnetus)) + EXTRACT(month FROM age(current_date,õnnetus))/12);
```

Kui see on arvutatud, arvutatakse esialgne kliendikoefitsient. Selleks kasutatakse valemit:

```
kliendi_koefitsient = kliendi_koefitsient - aeg_õnnetusest / 10;
```

Seejärel toimub vahekontroll. Kontrollitakse, et koefitsient muutunud liiga suureks ega liiga väikseks. Kui koefitsient on alla 0.5 siis pannakse ta väärtuseks 0.5. Kui koefitsient on läinud suuremaks kui 1.0, siis määratakse ta väärtuseks 1.0.

Edasi toimub koefitsiendi arvutus sõltuvalt kliendi vanusest. Kui kliendi vanus on alla kahekümne aasta siis korrutatakse koefitsient kolmega, sest klient kuulub suurima riskiga gruppi.

Kui kliendi vanus on vahemikus 21 kuni 30, siis korrutatakse kliendi koefitsient kahega. Samuti korrutatakse koefitsient kahega, kui kliendi vanus on üle 50 aasta. Need vanusegrupid kuuluvad keskmise riskitasemega vanusegruppi.

Kui kliendi vanus on vahemikus 31-50, siis jääb koefitsient muutumatuks, sest need kliendid kuuluvad väikseima riskitasemega gruppi.

Seejärel väljastab funktsioon arvutatud koefitsiendi **kliendi_koefitsient**;

4.4 Edasiarenduse võimalused

Tulevikus on võimalik juurde lisada teisi kindlustuse liike. Esimene oleks ilmselt liikluskindlustus, mis on sarnane kaskoga. Kuna nüüd varsti on tulemas seadusemuudatused, mis muudavad tavalise liikluskindlustuse sarnasemaks kaskoga, siis ilmselt pole selle jaoks väga suuri muutusi vaja.

Veel oleks võimalik lisada nii reisikindlustus kui ka kodukindlustus. Need mõlemad vajaksid natukene analüüsimist, et kuidas kõige paremini olemasoleva süsteemiga sobitada. Ilmselt oleks lepingu tabelit vaja muuta, et ta poleks otseselt enam seotud auto_id kaudu auto tabeliga, vaid mingi üldisema objektiga, kust ta siis põhilise asjana saaks kätte koefitsiendi, mis rakendub, et oleks võimalik leida kuumakse. Kuumakse arvutuste jaoks oleks vaja ka uusi valemeid, sest nende puhul on ikkagi tegemist suhteliselt erinevate kindlustusliikidega kui võrrelda liikluskindlustusega.

Siis järgmiseks sammuks oleks lubada ka juriidilistel isikutel lepinguid sõlmida. See vajaks samuti natuke lisa analüüsi, sest kuumakse arvutus võiks sisalda lisa soodustusi vastavalt kindlustatavate objektide arvule ühe ettevõtte kohta. Veel peaks äriklientide jaoks ümber kujundama praeguse soodustuse süsteemi, mille soodustuse suurus sõltub ajast mis on möödunud viimasest õnnetusest. Ärikliendi puhul ilmselt ei peaks pärast ühte õnnetusjuhtumit kohe kuumakse suurus kasvama, vähemalt mitte liikluskindlustuse puhul. Kui on mingil ettevõttel tihedalt vaja kahju korvata, siis kindlasti peaks seda arvesse võtma ja vastavalt kuumakse suurust tõstma.

5. Kokkuvõte

Töö põhieesmärgiks oli funktsioneeriva kaskokindlustuse infosüsteemi andmebaasi osa loomine PostgreSQL´is. Loodud andmebaas lubab teha kõiki põhilise operatsioone, mida selliselt süsteemilt eeldada võib. Saab luua kasutajaid, lepinguid ja kõiki muid poliisi sõlmimiseks vajaminevaid objekte ja ka samuti kaskokindlustuse poliisi ennast. Loodud on ka funktsioone, mis tegelevad poliisi loomise käigus koefitsiendi arvutusega, mis lõpuks on vajalik, et arvutada loodava kaskokindlustus poliisi kuumaksumus eurodes.

Samuti on olemas ülevaade sellest, kuidas oleks tulevikus võimalik süsteemi edasi arendada.

Seega on töö raames seatud eesmärk saavutatud.

Summary

The main goal of this project was to create a functional casco insurance database in PostgreSQL. The created database allows the user to perform all the tasks, that you can excpect from such a system. You can create users, contracts and everything else needed for the creation of a insurance policy. Also you can create the casco insurance policy. Also there are functions that calculate the coefficient, that is finally needed when calculating the monthly payment in euros for the casco insurance policy.

There is also a synopsis of how it would be possible to forther develop and improve the current system.

So the goals set for this project have been met.

Kasutatud kirjandus

1. Five reasons why i choose postgresql [WWW] http://www.alandmoore.com/blog/2013/02/28/five-reasons-why-i-choose-postgresql/ (28.03.2014)

2. 20 Database Design Best Practices [WWW] http://www.javacodegeeks.com/2012/02/20-database-design-best-practices.html (28.03.2014)

3. 4.1. Lexical Structure [WWW] http://www.postgresql.org/docs/9.3/static/sql-syntax-lexical.html (19.04.2014)

4. B.4 MySQL 5.6 FAQ: Stored Procedures and Functions [WWW] docs.oracle.com/cd/E17952_01/refman-5.6-en/faqs-stored-procs.html#qandaitem-B-4-1-1 (16.05.2014)

5. 3 Getting Started with Database Administration [WWW] http://docs.oracle.com/cd/E11882_01/server.112/e10897/em_manage.htm#ADMQS12135 (16.04.2014)

6. Artifacts,_swimlanes_and_Connecting_Objects [WWW] http://wiki.bizagi.com/en/index.php?title=Artifacts,_swimlanes_and_Connecting_Objects (18.05.2014)

7. 9.9. Date/Time Functions and Operators [WWW] http://www.postgresql.org/docs/9.3/static/functions-datetime.html (05.04.2014)

8. 2.5 Six Principles of Database Design [WWW] https://www.inkling.com/read/access-2010-missing-manual-matthew-macdonald-1st/chapter-2/six-principles-of-database (01.04.2014)

9. 5.3. Constraints [WWW] http://www.postgresql.org/docs/9.3/static/ddl-constraints.html (24.03.2014)

10. 40.6. Control Structures [WWW] http://www.postgresql.org/docs/9.3/static/plpgsql-control-structures.html (29.03.2014)

Lisa 1

Lihtlitsents lõputöö üldsusele kättesaadavaks tegemiseks ja reprodutseerimiseks

Mina Harald Kosk (sünnikuupäev: 04.12.0989)

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose PostgreSQL´is kaskokindlustuse infosüsteemi loomine mille juhendaja on Raul Liivrand

1.1. reprodutseerimiseks säilitamise ja elektroonilise avaldamise eesmärgil, sealhulgas TTÜ raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas TTÜ raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2. Olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.

3. Kinnitan, et lihtlitsentsi andmisega ei rikuta kolmandate isikute intellektuaalomandi ega isikuandmete kaitse seadusest ja teistest õigusaktidest tulenevaid õigusi.

_____ (allkiri)

_____ (kuupäev)

Lisa 2

METAANDMED

Töö pealkiri (eesti keeles): PostgreSQL´is kaskokindlustuse infosüsteemi loomine
Töö pealkiri (inglise keeles): Creating a casco insurance infosystem in PostgreSQL
Autor: Harald Kosk
Juhendaja(d): Raul Liivrand
Kaitsmise kuupäev:
Töö keel: est
Asutus (eesti keeles): TTÜ / TTÜ õppeasutus (nimi): Tallinna Tehnikaülikool
Asutus (inglise keeles): TTÜ / TTÜ õppeasutus (nimi): Tallinn University of Technology
Teaduskond (eesti keeles): Infotehnoloogia teaduskond
Teaduskond (inglise keeles): Faculty of Information Technology
Instituut (eesti keeles): Informaatikainstituut
Instituut (inglise keeles): Department of Informatics
Õppetool (eesti keeles): Infosüsteemide õppetool
Õppetool (inglise keeles): Chair of Informations Systems
Märksõnad /kui on/ (eesti keeles):
Märksõnad /kui on/ (inglise keeles):
Õigused: juhul kui ligipääs on piiratud, siis sellekohane märkus