

Ülesanne 2

Eesmärk: CASE vahendis tuleb andmebaasi füüsilise disaini mudel saada sellisesse korda, et sellest koodi genereerimisel on saadud kood võimalik kas parandusteta või minimaalsete paranduste ja täiendustega käivitada ning tulemuseks on hästi disainitud andmebaas. Samuti peavad olema andmebaasi diagrammid nii ilmekad ja visuaalselt hästi esitatud, et need saab kopeerida portree-orientatsioonis A4 formaadis dokumenti ning tulemus on hästi loetav. **Sisuliselt tegelete Te programmeerimise ja programmi silumisega, ainult et tekstilise lähtekoodi asemel töötate selle visuaalse esitusega UMLis. Selleks, et programmeerida, on vaja nõudeid. Nõuded on antud juhul "Andmebaasid I" projektis (kontseptuaalses andmemudelis ja mujal). Nii selle kui järgnevate ülesannete täitmisel tuleb selle dokumendiga konsulteerida ja see peaks kõrval lahti olema.**

Väga paljud ülesandes ettenähtavad tegevused on mõeldud andmebaasi disainist või selle mudelitena esitusest *halvasti lõhnavate* kohtade eemaldamiseks ja selle kaudu *tehnilise võla* vähendamiseks. Väga paljudele nendele halbadele lõhnadele ja parandustele leidub vaste Robert C. Martini puhta koodi raamatus.

Martin, R. C., 2008. *Clean Code: A Handbook of Agile Software Craftsmanship*. 1st ed. New York – Prentice Hall.

Seda raamatut saate lugeda TTÜ raamatukogu kaudu e-raamatuna, mis on Safari andmebaasis.

<http://proquestcombo.safaribooksonline.com/book/software-engineering-and-development/agile-development/9780136083238>

Siin on info, kuidas kasutada seda andmebaasi väljastpoolt TTÜ võrku:

<https://www.ttu.ee/asutused/raamatukogu/teenused/kaugtoo/>

Tööd tuleb teha siin ja edaspidi oma projekti kaaslastega koos rühmatööna (kui käite samas harjutustunnis) või kui käite erinevates harjutustundides, siis tuleb jätkata tööd, mida samal nädalal varem harjutustunnis käinud kaaslased alustasid.

Väljaspool harjutustundi korrastage dokumenti. Eesmärk on, et Teie kirjatöö vormistus järgiks võimalikult täpselt lõputööde kirjutamise juhendit, mille leiate aadressil: <https://www.ttu.ee/teaduskond/infotehnoloogia-teaduskond/it-tudengile/loputoo-ja-lopetamine-9/loputoo-vormistamine-4/> Kontrollige ja vajadusel parandage oma dokumenti järgmiste reeglite alusel.

- ⤴ Kõik esimese taseme pealkirjad peavad algama uuel leheküljelt.
- ⤴ "Kõigis peatükkides ja alapeatükkides peab pealkirjale järgnema tekst. Uue alapealkirja, joonise või tabeli esitamine koheselt pärast pealkirja ei ole lubatud – pealkirjale järgneval real on alati tekst."
- ⤴ Kõikidel joonistel on allkirjad ning tabelitel on pealkirjad.
- ⤴ Kõikidele joonistele ja tabelitele viidatakse töö tekstis.
- ⤴ Kõikidele kasutatud materjalidele viidatakse töö tekstis.
- ⤴ Hoidutakse neljanda taseme pealkirjadest.

Olen kohendanud projekti töövihikut ja näiteprojekti, et olla nende nõuetega kooskõlas.

Kui kasutate CASE vahendit Enterprise Architect, siis on Teil tegelikult erinevad võimalused, kuidas saavutada ülesannete 2–4 oodatav tulemus.

1. Teha koopia eeldusaine projektis loodud andmebaasi tabelite disaini mudelist ja jätkata tööd koopiaga. Täiustatud mudelist tuleb genereerida andmekirjelduskeele lausete kood, mida tuleb siluda ja siis käivitada.
Sellist lahendust kirjeldavad ülesanded 2–4.
2. Luua pöördprojekteerimist (*reverse engineering*) kasutades eeldusaines realiseeritud andmebaasi põhjal andmebaasi disaini mudel. Jätkata tööd selle mudeliga vastavalt ülesannetes 2–4 esitatud juhendile.
3. Luua otse andmebaasi tabelid. Tabelite disainimisel peaks arvestama kõigi ülesannetes 2–4 väljatoodud põhimõtetega. Lõpuks tuleb pöördprojekteerimist (*reverse engineering*) kasutades luua loodud tabelite põhjal andmebaasi disaini mudel ja esitada see diagrammidel vastavalt ülesandes 2 toodud juhendile.
 - PostgreSQL andmebaasi kitsenduste nimede ühtlustamiseks saate kasutada PostgreSQL andmebaasi laiendust:
https://github.com/katrinaibast/constr_name_unif

Pöördprojekteerimine EA vahendis:

http://www.sparxsystems.com/enterprise_architect_user_guide/12.0/database_engineering/importdatabaseschemafromod.html

Selle ülesande täitmise lõpuks tuleb saada korda oma iseseisva töö projekti andmebaasi füüsilist disaini kirjeldavad diagrammid (välja arvatud CHECK kitsendused ja indeksid, mida pole vaja selle ülesande lahendamise käigus kirjeldada). Vajadusel tuleb kooskõla säilitamiseks täiendada kontseptuaalset andmemudelit. Andmebaasi diagrammid on näiteprojekti jaotises 3.1.

Otsustuspuu, kuidas alustada, kui **"Andmebaasid I" projekt on tehtud**.

- 1 Soovin kasutada **Enterprise Architect (EA) CASE** vahendit
 - 1.1 **"Andmebaasid I" projekt on tehtud EA abil**
 - 1.1.1 Soovin kasutada **Oracle** andmebaasisüsteemi
 - 1.1.1.1 Juhul, kui ma kevadel ei teinud projekti Oracles, siis teen kevadel genereeritud tabelite kirjeldust sisaldavast paketist/pakettidest koopia ning määran koopias kõikide tabelite puhul, et need realiseeritakse andmebaasisüsteemis Oracle
 - 1.1.1.2 Juhul kui ma kevadel tegin projekti Oracles, siis alustan kevadel genereeritud andmebaasi füüsilise disaini mudeli parandamist
 - 1.1.2 Soovin kasutada **PostgreSQL** andmebaasisüsteemi
 - 1.1.2.1 Juhul, kui ma kevadel ei teinud projekti PostgreSQLis, siis teen kevadel genereeritud tabelite kirjeldust sisaldavast paketist/pakettidest koopia ning määran koopias kõikide tabelite puhul, et need realiseeritakse andmebaasisüsteemis PostgreSQL
 - 1.1.2.2 Juhul kui ma kevadel tegin projekti PostgreSQLis, siis alustan kevadel genereeritud andmebaasi füüsilise disaini mudeli parandamist
 - 1.2 **"Andmebaasid I" projekt on tehtud RR abil**
 - 1.2.1 Loen "Kuidas saada Enterprise Architect" dokumendist (kataloog *Tarkvara saamine ja kasutamine*), kuidas minna RR kasutamiselt üle EA kasutamisele. Kui olen ülemineku lõpetanud, siis jätkan punktist 1.1
- 2 Soovin kasutada **Rational Rose (RR) CASE** vahendit
 - 2.1 **"Andmebaasid I" projekt on tehtud RR abil**
 - 2.1.1 Soovin kasutada **Oracle** andmebaasisüsteemi
 - 2.1.1.1 Juhul, kui ma kevadel ei teinud projekti Oracles, siis genereerin kontseptuaalsest andmemudelist uue andmebaasi füüsilise disaini mudeli, määrates andmebaasisüsteemiks Oracle
 - 2.1.1.2 Juhul kui ma kevadel tegin projekti Oracles, siis alustan kevadel genereeritud andmebaasi füüsilise disaini mudeli parandamist
 - 2.1.2 Soovin kasutada **PostgreSQL** andmebaasisüsteemi
 - 2.1.2.1 Juhul kui ma kevadel tegin projekti määrates andmebaasi füüsilise disaini mudelis andmebaasisüsteemiks ANSI SQL 92, siis alustan kevadel genereeritud andmebaasi füüsilise disaini mudeli parandamist
 - 2.1.2.2 Juhul, kui ma kevadel ei teinud projekti määrates andmebaasi füüsilise disaini mudelis andmebaasisüsteemiks ANSI SQL 92, siis genereerin kontseptuaalsest andmemudelist uue andmebaasi füüsilise disaini mudeli, määrates andmebaasisüsteemiks ANSI SQL 92

Kui Te teete projekti **mõne muu CASE vahendiga kui EA või RR**, siis tuleb valida selline CASE vahend, milles tehtud andmebaasi diagrammides saate esitada samasugust informatsiooni kui EA või RR vahendis tehtud diagrammides.

Kui Teil on **olemas PostgreSQL või Oracle andmebaas**, kuid **puudub andmebaasi füüsilise disaini mudel**, siis võite selle esimese versiooni genereerimiseks EA vahendis kasutada pöördprojekteerimist (*reverse engineering*).

http://sparxsystems.com/enterprise_architect_user_guide/12.1/database_engineering/importdat abaseschemafromod.html

Teadmiseks! Kahjuks on nii RR kui EA CASE vahendis loodud andmebaasi disaini mudelitega seoses õhus riknemise oht.

RR korral avaldub riknemine selliselt, et tehes tabeli kirjeldusel topeltklõpsu, ei avane tabeli kirjelduse aken, kui soovite andmebaasi kirjeldusest genereerida SQL laused, siis see ei õnnestu. Te ei saa disaini kirjeldust enam muuta. Mudel rikneb, sest mõni selle element rikneb. Kui suudate (*katse-eksituse meetodil*) selle elemendi üles leida ja ära kustutada, siis saate mudeliga normaalselt edasi töötada. EA korral kaovad ära näiteks välisvõtme kitsendused. Uuesti loomise, salvestamise ning faili uuesti avamise järel on need jälle kadunud. Ei jäägi muud üle, kui disaini mudel uuesti luua.

Ühel üliõpilaste rühmal RR korral probleem (vähemalt ajutiselt) kadus, kui mudelis eemaldati skeemi (*schema*) nimest ##. Seega, palun ärge igaks juhuks kasutage RR mudelis skeemi nimedes # märke. Skeemi nime C##TUD4 asemel kasutage nime TUD4.

Sellega seoses järgmine soovitus – enne kui hakkate mudelifailiga tööle, tehke sellest koopia. Kui fail peaks riknema, siis saate koopiat kasutades faili suhteliselt värske versiooni taastada. Soovitan sellise koopia teha iga harjutustunni alguses ja ka iga kord kui väljapool koolitunde selle failiga tööle hakkate.

RR pole (võrreldes EAga) üldse paha ja mõni asi on seal ka parem/mugavam kui EAs. Seega nii RR kui EA sobivad antud aines kasutamiseks.

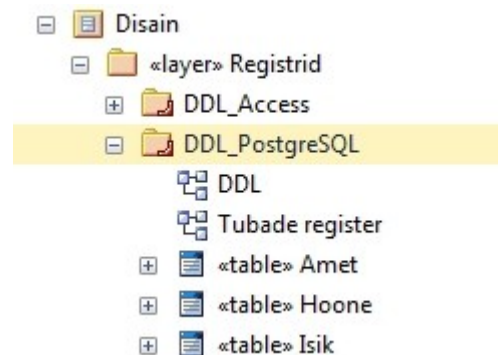
Hoiatus! EAs on välisvõtmete muutmine väga ebamugav ning välisvõtme kitsenduste kirjeldused võivad väga kergesti rikneda.

Mõelge UML paketist kui failisüsteemi kataloogist. Te loote katalooge, et:

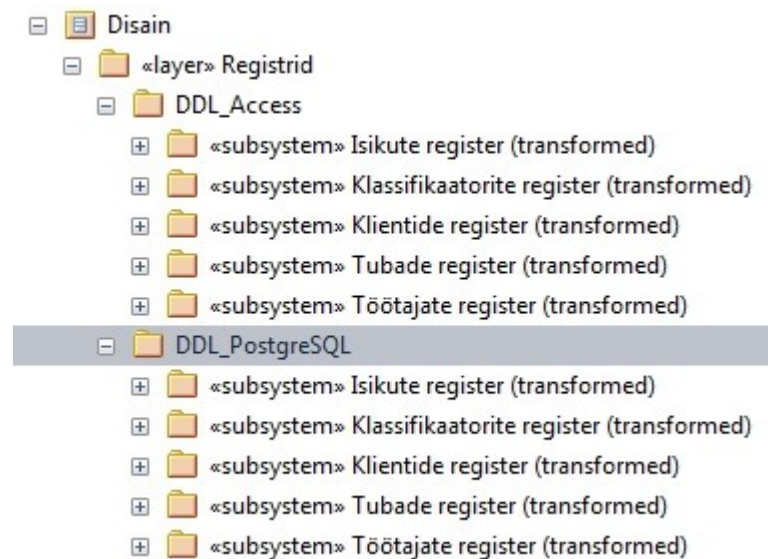
- faile organiseerida ja lihtsamini üles leida,
- vältida failide nimekonflikte.

Järgnevalt illustreeritakse võimalikku andmebaasi füüsilise disaini mudeli pakettide struktuuri EAs.

Juhul, kui kasutasite tabelite kirjelduse saamiseks EA vaikimisi mudeliteisendust.



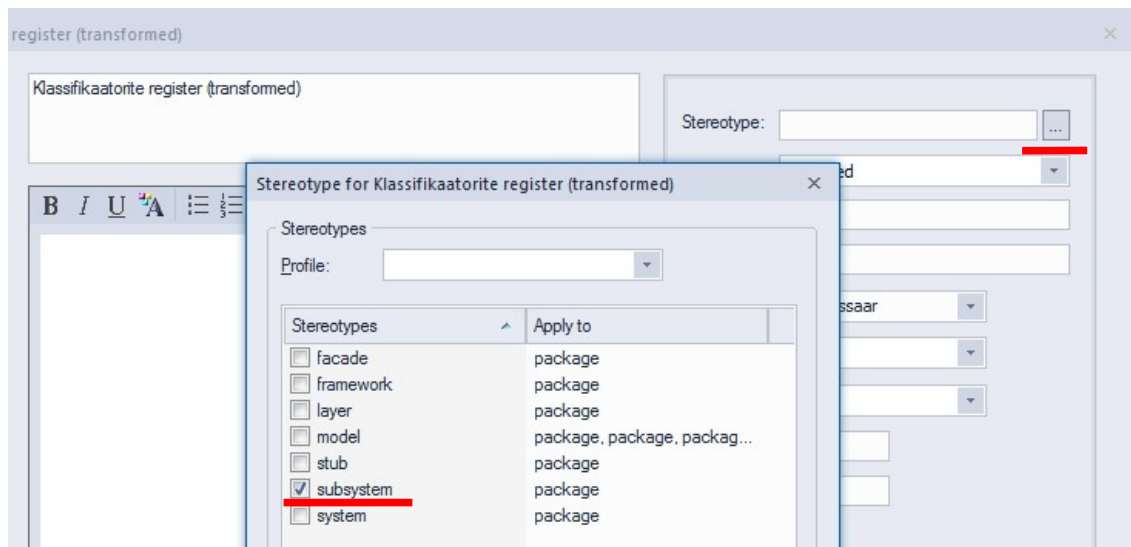
Juhul, kui kasutasite tabelite kirjelduse saamiseks "Andmebaasid I" aines ja ka selles dokumendis eelnevalt kirjeldatud täiustatud mudeliteisendust.



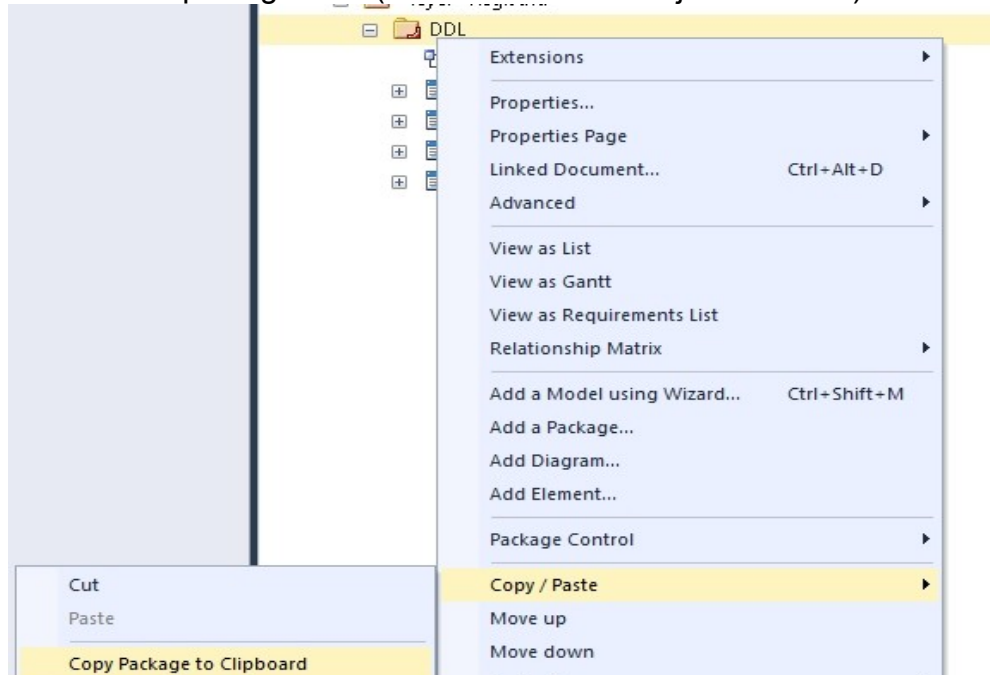
Pange tähele, et registreid (andmekesksed allsüsteemid) tähistavatele pakettidele on stereotüübiks määratud <<subsystem>>. See on tehtud ilmekuse huvides, et mudeli vaatlejal oleks kohe selge, mida see pakett näitab. Samal põhjusel on paketil *Registrid*, stereotüüp <<layer>>, sest registreite näol on tegemist äriarhitektuuri ühe kihiga.

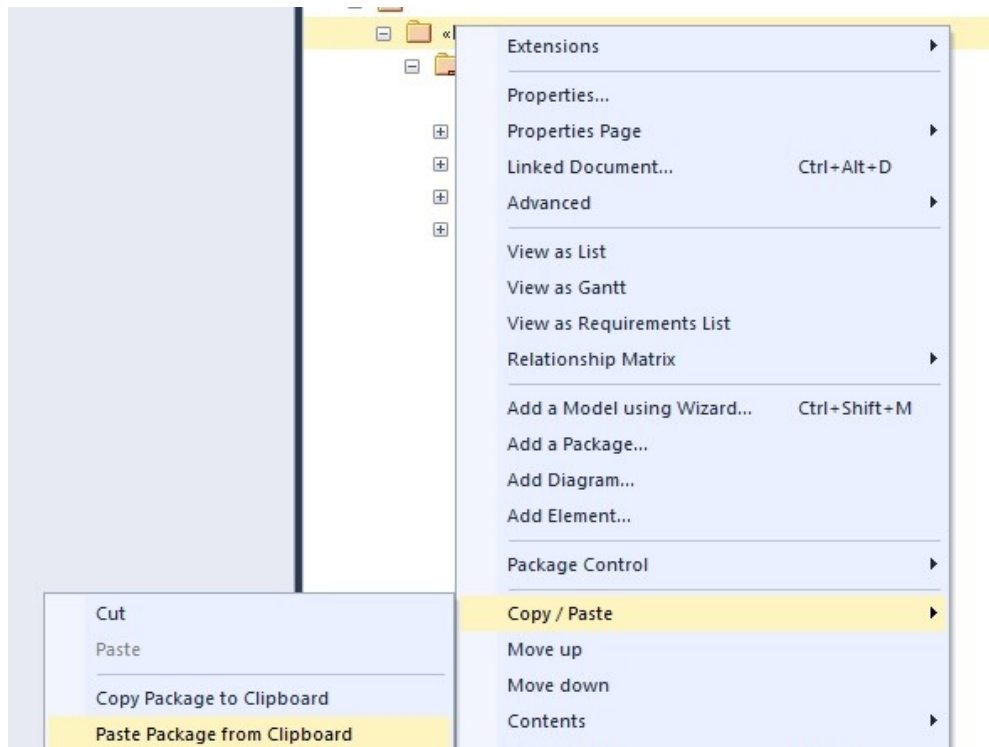
Kuidas sellise struktuurini jõuda? Järgnevad ekraanitõmmised illustreerivad EAs tabelite kirjeldust sisaldavast paketist koopia tegemist, pakettide ümbernimetamist ning tabeliga seotud andmebaasisüsteemi määramist.

Paketi stereotüübi määramine

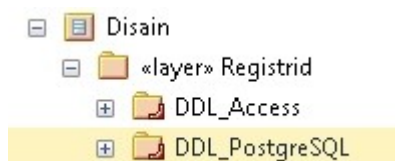
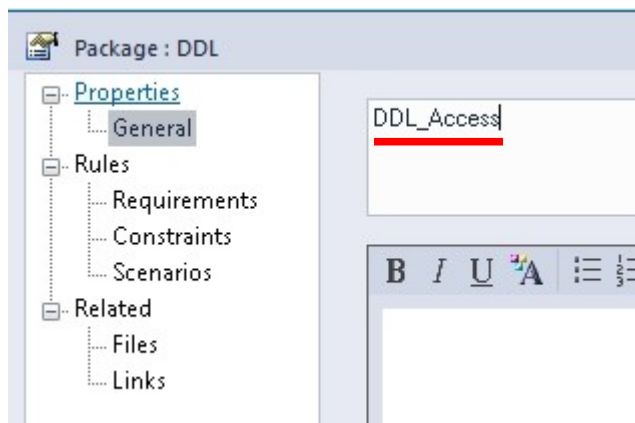


Paketist koopia tegemine (töötab ka CTRL + C ja CTRL + V)



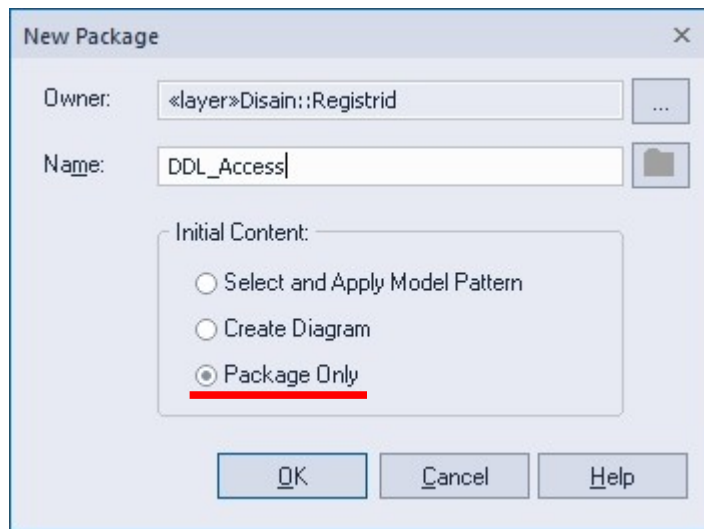


Pakettide ümbernimetamine, et oleks selge, millises paketis on millise andmebaasisüsteemi jaoks mõeldud andmebaasi füüsilise disaini kirjeldus.

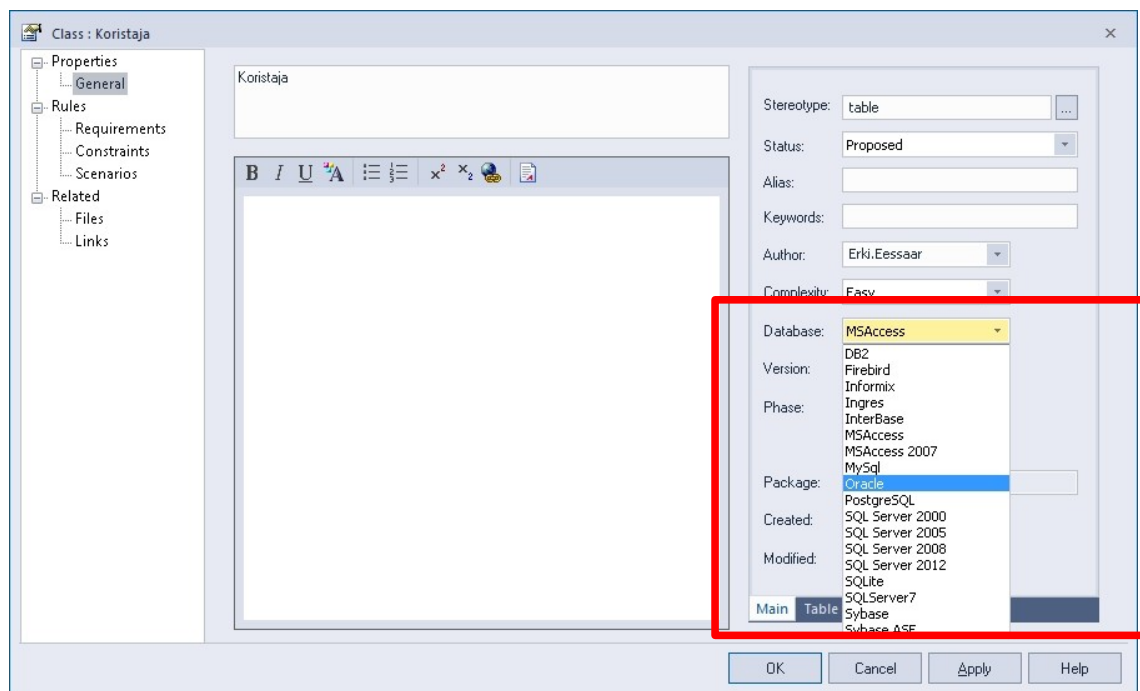


Veenduge alati, et avate vaatamiseks õiges paketis olevad diagrammid.

Kui on vaja lisada uus tühi pakett:



Tabeliga seotud andmebaasisüsteemi muutmine. Selle tulemusena muudab CASE vahend eeldefineeritud vastavustabeli alusel veergude tüüpe. Kahjuks on tulemus ebakvaliteetne ja seal on vaja teha palju muudatusi. Andmebaasisüsteemi muudatus tuleb teha kõikides kopeeritud pakettide tabelite kirjeldustes. Muudatuse tulemusena saab tabeli veergude kirjeldamisel valida andmetüübi tabeli kirjelduses määratud uue andmebaasisüsteemi andmetüüpide hulgast.



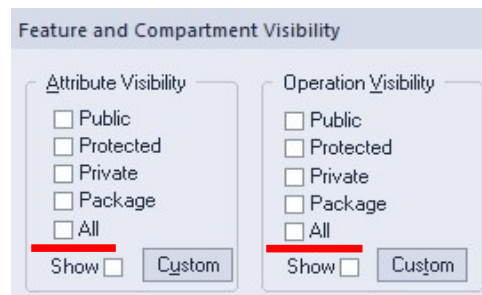
Millele andmebaasi füüsilise disaini mudeli täpsustamisel tähelepanu pöörata?

- ⤴ Kontseptuaalne andmemudel ja andmebaasi füüsilise disaini mudel peavad olema kooskõlas. Kõik, mis on disainitaseme andmemodelis (v.a disainitaseme abitabelid), peab olema tuletatav teisendusreeglite rakendamise tulemusena kontseptuaalsest andmemodelist. Kui tahate muuta tabelite struktuuri, et näiteks alustada mingite uute andmete registreerimist, siis tuleb see muudatus viia sisse ka kontseptuaalsesse andmemodelisse ning kui nende andmete registreerimine toimub projekti teemaks olevas funktsionaalses allsüsteemis, siis ka teistesse mudelitesse.

PostgreSQL või Oracle andmebaasi disaini kirjeldavad diagrammid tuleb Teil lõpuks panna projekti dokumendi kolmandasse peatükki – kevadel tehtud andmebaasi disaini diagrammide asemele.

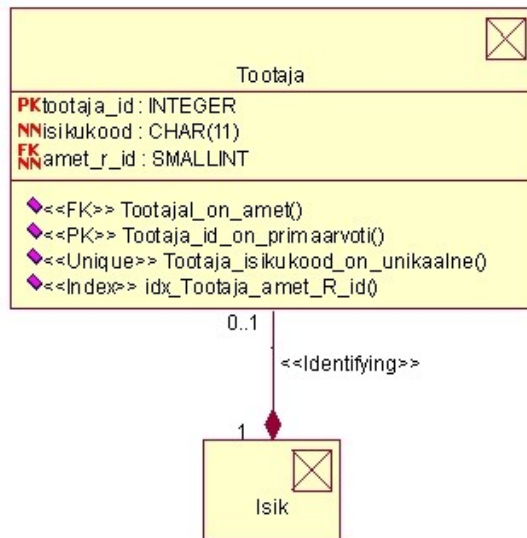
- ⤴ Eesmärk: Kõik diagrammid peavad olema dokumendis (A4 formaadis portree-orientatsiooniga leht) 100% suurendusega loetavad. Järgnevalt kirjeldatakse, kuidas seda andmebaasi füüsilise disaini diagrammide korral saavutada.
 - **Kui Te pole seda veel teinud**, siis looge iga funktsionaalse allsüsteemi poolt kasutatava või teenindatava registri kohta eraldi andmebaasi diagrammid (vt näiteprojekti). Teiste sõnadega, looge vähemalt üks diagramm iga alajaotuses **"1.2.3Allsüsteemi poolt vajatavad registrid"** nimetatud registri kohta.
 - Iga registri kohta tuleb luua üks või mitu diagrammi (mitu siis, kui ühe diagrammi jaoks on elemente liiga palju või on neid ühelt diagrammilt halb lugeda). Näiteks ühel diagrammil on kõik klassifikaatorite tabelid (kui äriarhitektuuris on *klassifikaatorite register*). Teisel diagrammil on tabelid töötajate kohta (kui äriarhitektuuris on *töötajate register*) jne. Kui klassifikaatorite tabelleid on ühel diagrammil esitamiseks liiga palju, siis tehke mitu diagrammi – näiteks ühel diagrammil on ainult seisundiklassifikaatorid.
 - Igal registri R kohta loodud diagrammil on registrisse R kuuluvad tabelid + nende tabelitega **otseselt** välisvõtmete kaudu seotud tabelid, **kui välisvõti on mõnes registrisse R kuuluvas tabelis**.
 - ⤴ Registrisse R *kuuluvate* tabelite korral on näha veerud, kitsendused ja indeksid.
 - ⤴ Registrisse R *mittekuuluvate* tabelite korral on veerud, kitsendused ja indeksid **peidetud**.
 - Tabeli veergude ja kitsenduste peitmiseks RR andmebaasi diagrammil aktiveerige diagrammil olev tabeli kirjeldus ning valige parema hiireklahvi alt avanevast menüüst:
 - *Options => Suppress Columns*
 - *Options => Suppress Triggers*
 - Tabeli veergude ja kitsenduste peitmiseks EA andmebaasi diagrammil aktiveerige diagrammil olev tabeli kirjeldus ning valige parema hiireklahvi alt avanevast menüüst:
 - *Features & Properties => Feature and Compartment Visibility...*

- ✧ Avanenud akna vasakus ülemises nurgas eemaldage märgendid valikute "All" eest:

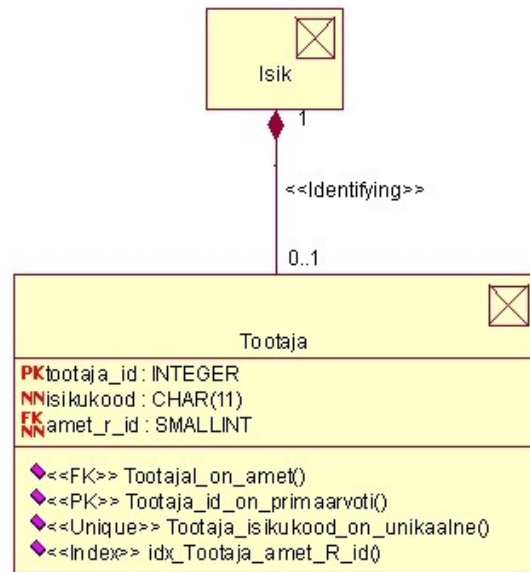


- Kui toimite eelnevalt toodud juhiste järgi, siis on täidetud ka järgmised üldised nõuded.
 - Rusikareegli kohaselt ei peaks diagrammil olema üle 5+/-2 elemendi (antud juhul tabeli). Pidage meeles eesmärki, milleks on hästi loetav diagramm. See rusikareegel on lihtsalt abivahend selle eesmärgini jõudmiseks. Vabalt võib olla, et viis suure veergude/kitsenduste arvuga tabelit on liiga palju ja kümme väikese veergude/kitsenduste arvuga tabelit on täiesti piisav.
 - Diagrammid peavad olema osalise ülekattega, st neil on mõned ühised elemendid. See aitab erinevad diagrammid mõttes tervikuks kokku siduda.
 - Kui esitate ühe ja sama tabeli kirjelduse mitmel diagrammil, siis tuleks selle tabeli veerge, kitsendusi ja indekseid näidata vaid ühel diagrammil ([*Ainult Üks Kord*](#) disainipõhimõte).
 - Iga tabeli detailid (veerud, kitsendused, indeksid) on näha täpselt ühel diagrammil. Tabeli detailideta esitus koos seostega võib olla nähtav rohkem kui ühel diagrammil.
- ✧ Paigutage loetavuse huvides diagrammil tabelid nii, et ülatüübile vastav tabel ei oleks madalamal kui selle alamtüüpidele vastavad tabelid. Seega, kui *Isik* on ülatüüp ning *Klient* ja *Töötaja* on selle alamtüübid, siis on tabel *Isik* diagrammil kõrgemal (või vähemalt samal tasemel) kui tabelid *Töötaja* ja *Klient*.

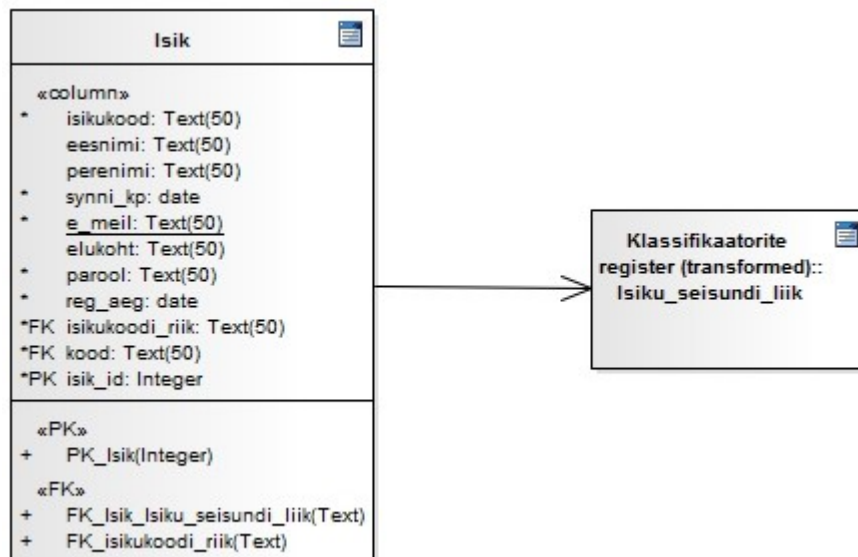
Halvem paigutus diagrammil



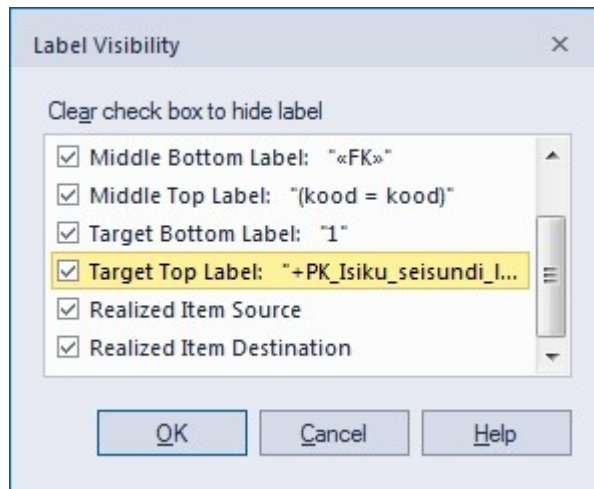
Parem paigutus diagrammil



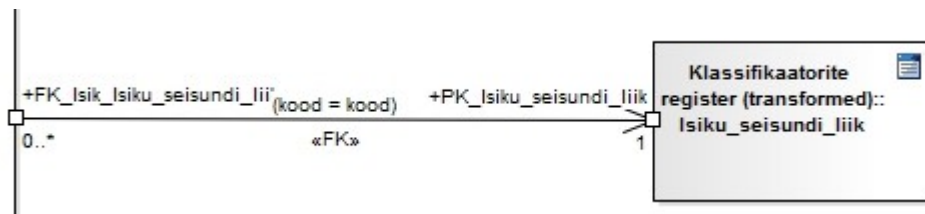
- ⤴ Nii nagu Te annate sisukad nimed registritele, tabelitele või veergudele, tuleb sisukad nimed anda ka diagrammidele.
- ⤴ Tagage EA vahendis, et välisvõtmete kohta väljastataks diagrammil võimalikult palju informatsiooni. Kui tabelite vaheline seos näeb diagrammil välja nii



, siis EA12 korral klõpsake seosel ning valige parempoolse hiireklahvi alt hüpikmenüüst *Visibility => Set Label Visibility* Märgistage kõik valikud:



Veenduge, et oleks nähtav valik *Visibility => Hide All Labels* (st *Show All Labels* on sisse lülitatud). Seose visuaalne esitus muutub diagrammil selliseks:



- ▲ Skeemiobjektide, veergude ja kitsenduste identifikaatorid e nimed.
 - Lugege seda dokumenti, järgige selle soovitusi ja vältige selles kirjeldatud vigu:
<http://bonesmoses.org/2016/07/08/pg-phriday-all-in-a-name/> Kuigi see artikkel on kirjutatud PostgreSQL jaoks, kehtib see ka Oracle ning kõigi teiste SQL-andmebaasisüsteemide korral.
 - Nimed peavad lähtuma mingist (Teie valitud) reeglistikust, mida tuleb kõigi nimede puhul **järjekindlalt** järgida. Soovitan need reeglid kuhugi kirja panna ja projekti liikmete vahel laiali jagada, et kõik neid järgiksid.
 - Nimed peavad vastama valitud andmebaasisüsteemi reeglitele (seega näiteks kõik tühikud tuleb eemaldada – piiritletud identifikaatorid, kus need on lubatud, on ebasoovitavad).

PostgreSQL: <https://www.postgresql.org/docs/10/static/sql-syntax-lexical.html#SQL-SYNTAX-IDENTIFIERS>

Oracle: https://docs.oracle.com/database/121/SQLRF/sql_elements008.htm

- Ärge kasutage läbisegi erinevaid kirjastiile.
 - [S/s]nake_case
 - camelCase
 - PascalCase

Soovitan kasutada **[S/s]nake_case**, sest andmebaasiobjektide regulaarsed identifikaatorid salvestatakse andmebaasi süsteemikataloogis kas suur- või väiketähtedega. Genereerides andmebaasi põhjal tagasi andmebaasikeele lauseid, on tulemuseks (sõltuvalt andmebaasisüsteemist) identifikaatorid nagu:

CAMELCASE, camelcase, PASCALCASE, pascalcase, SNAKE_CASE, snake_case.

- Kuigi see pole andmebaasisüsteemi nõue, siis soovitan asendada ka täpitähed (õÕäÄöÖüÜ). Sõltuvalt kasutatavast tehnoloogiast pole välistatud probleemid programmides, mis peavad kasutama selliste nimedega andmebaasiobjekte. Samuti võib tekkida probleem andmebaasisüsteemi vahetamisel, kui üks süsteem lubab identifikaatorites täpitähti ja teine mitte. Jällegi – asendustes tuleb olla järjekindel, teha seda igal pool ning kasutades igal pool ühesugust asendust.
- Suur- ja väiketähtedest. Juhul kui andmebaasiobjekti loomise lauses pole nimi jutumärkides (st nime näol on tegemist regulaarse identifikaatoriga), siis andmebaasis on nimi tõstutundetu (*case insensitive*). See tähendab, et andmebaasi kasutamise mõttes pole vahet, kas nimes kasutati suurtähti, väiketähti või nende kombinatsiooni. Lause `CREATE TABLE Isik ...` loob tabeli, mille põhjal saab teha päringut nii `SELECT * FROM Isik;` kui `SELECT * FROM ISIK;` kui ka `SELECT * FROM isik;`. Andmebaasi kasutatavuse seisukohalt tuleks eelistada tõstutundetuid nimesid tõstutundlikele (st ülesandes 4 on vaja hoolitseda, et SQL koodis ei oleks nimed jutumärkides – jutumärkides nimi on tõstutundlik). Mudelite ja koodi loetavuse seisukohalt oleks siiski hea, kui olete nimeses suur ja väiketähtede kasutamisel järjekindel. Järgnevalt esitan mõned soovitusel (pole reeglid; peaasi – olge järjekindel):
 - Ärge kasutage nimes läbivalt suurtähti (ISIK). Sotsiaalmeedias seostatakse seda karjumisega: https://en.wikipedia.org/wiki/All_caps
NB! Kui muudate EA12 Oracle andmebaasi disaini mudelis objekti (nt tabel, veerg) nime, siis muudab CASE vahend automaatselt seal kõik tähed suurtähtedeks. Hea oleks, kui siis oleks tabelite disaini mudelis KÕIK nimed suurtähtedega.
 - Tabeli nimes võiks olla esitäh suurtäht ja ülejäänud tähed väiketähed (nt *Isiku_seisundi_liik*).
 - Veergude nimeses võiks kasutada läbivalt väiketähti (nt *isiku_seisundi_liik_kood*).
 - Kui kitsenduse nimes on tabeli(te) nimed, siis ka see (need) võiksid alata suurtähaga ja veergude nimed vastavalt väiketähaga (nt *chk_Isik_eesnimi_perenimi*).
 - Tüübi tähise puhul võiks läbivalt kasutada ühesugust suur- ja väiketähtede kombinatsiooni (nt *PK_Isiku_seisundi_liik* – PK kõikides nimeses suurtähtedega).
- SQL-andmebaasis esitatakse andmed tabelites, kus on read ja veerud. Mõttetu on kasutada tabelite (sh tuletatud tabelite) nimeses üldmõistetele viitavaid termineid nagu "tabel", "andmed", "info" ning veergude nimeses üldmõistetele viitavaid termineid nagu "veerg", "andmed", "info".
 - **Halvem:**
 - ▲ Ruumi_andmed
 - **Parem:**
 - ▲ Ruum
- Ärge kasutage liiga üldiseid veergude nimesid.

- **Halb:**
 - ⤴ id
 - ⤴ kood
 - ⤴ tüüp
 - ⤴ aeg
 - ⤴ kp
- **Parem:**
 - ⤴ isik_id
 - ⤴ riik_kood
 - ⤴ kaup_tüüp
 - ⤴ reg_aeg
 - ⤴ synni_kp
- Lihtsustamaks tabelite ühendamise lausete kirjutamist:
 - andke välisvõtme veergudele ja nende poolt viidatavatele võtmeveergudele ühesugused nimed. Selle soovitus vastu *peab* eksima, kui välisvõti viitab sama tabeli kandidaatvõtmele (rekursiivne seos), sest nimega tabelis ei tohi olla mitu samanimelist veergu. Selle soovitus vastu *võib* eksida, kui välisvõtme veeru nimi kirjeldab primaarse tabeli *rolli* välisvõtmele vastava seosetüübi kontekstis – näiteks tabelis *Riik* on primaarvõtme veerg *riik_kood* ja tabelis *Isik* on välisvõtme veerg *isikukoodi_riik*.
 - ⤴ **Halvem:**
 - Välisvõtme veeru nimi: riik
 - Välisvõtme poolt viidatava veeru nimi: riik_kood
 - ⤴ **Parem:**
 - Välisvõtme veeru nimi: riik_kood
 - Välisvõtme poolt viidatava veeru nimi: riik_kood
 - ⤴ **Kontranäide**
 - **Halvem:**
 - Tabelis *Treening* välisvõtme veeru nimi *isik_id*, mis on seotud tabeli *Tootaja* veeruga *isik_id*.
 - ⤴ *Treening* ja *isik* võivad olla seotud väga erinevatel põhjustel – *isik* võib näiteks olla treeningu registreerija, viimane andmete muutja, treeningu kinnitaja, treener, treeningu eest vastutaja jne. Veeru nimest ei tule see põhjus välja ja see halvendab inimkasutaja jaoks tabeli struktuurist arusaamist.
 - **Parem:**
 - Tabelis *Treening* välisvõtme veeru nimi *registreerija_id*
 - ärge kasutage erinevate tabelite võtme ja välisvõtme veergude nimedes segamini erinevaid käändeid.
 - ⤴ **Halb:**
 - isiku_id
 - kaup_id
 - klient_kood
 - kauba_kood
 - ⤴ **Parem:**
 - isik_id
 - kaup_id

- kliendi_kood
- kauba_kood
- Oracle puhul tuleb **kõigis** skeemiobjektide nimedes ja kitsenduste nimedes kasutada nimekonfliktide vältimiseks oma matrikli numbrit (nt eesliide **t990999_**). Kui teete projekti rühmas, siis valige ühe rühma liikme matrikli number ja kasutage seda järjekindlalt kõigi loodavate andmebaasiobjektide puhul. **Ärge kasutage Oracles matriklinumbreid veergude nimedes. Selleks pole mingit vajadust ja see muudab hilisema andmebaasis päringute ja muudatuste tegemise oluliselt tülikamaks.**
- Kitsendustele tuleb anda sisukad nimed. Kui CASE vahend lisab kitsenduste nimedesse numbreid, siis tuleb need eemaldada.
- Kitsenduste nimed peaksid olema skeemi piires unikaalsed. Lisage kitsenduse nimesse tabeli nimi, millega see on seotud. Miks on see veel kasulik?
 - Kui saate andmeid muutes kitsenduse vastu eksides veateate, milles sisaldub kitsenduse nimi, siis saate vea põhjuse kergemini tuvastada.
 - Kuna primaarvõtme ja unikaalsuse kitsenduste alusel loob süsteem automaatselt indeksid, mille nimi tuletatakse kitsenduse nimest, siis on SQL lausete täitmisplaane lugedes lihtsam aru saada, millise tabeli millist indeksit süsteem loeb.
- Oracle puhul jälgige, et nimed (sh kitsenduste nimed) poleks pikemad kui **30 baiti** (märki). Kontrollimiseks võite näiteks kasutada: <https://mothereff.in/byte-counter> (PostgreSQLis on *apex.ttu.ee* serveris kasutusel vaikumisi maksimaalne pikkus – **63 baiti**; Oracle lubab *enamike* nimede puhul (teatud tüüpi andmebaasiobjektide korral peab nimi olema siiski lühem) maksimaalse pikkusena 128 baiti alates versioonist 12.2, kuid serveris on versioon 12.1.0.1).
 - Kui PostgreSQLis käivitavas SQL lauses (nt CREATE TABLE, SELECT) on identifikaator lubatust pikem, siis lühendab PostgreSQL ise nime. Siiski on vajalik see nimi ise lühemaks teha – nii andmebaasis kui seda kirjeldavates mudelites.
 - ⤴ Lühendamise tulemusena ei pruugi nimi enam olla unikaalne.
 - ⤴ Lühendamine ei toimu psqli käskudes.
 - ⤴ Liiga pikka nime on tülikas lugeda, meelde jätta ja kirjutada – tõmmake paralleele:
 - <http://www.historyrundown.com/top-5-people-with-the-longest-names/>
- Jälgige, et veeru nimi ja selles registreeritavate andmete sisu oleks kooskõlas.
 - **Halb:**
 - ⤴ registreerimise_kp TIMESTAMP
 - **Parem:**
 - ⤴ registreerimise_kp DATE
- Temporaalsete (ajaandmete) veergudele soovitan järgmist nimeskeemi.
 - Kui veerus registreeritakse kuupäev + kellaaeg, siis veeru nimel järelliide *aeg* (nt alguse_aeg, registreerimise_aeg)
 - Kui veerus registreeritakse ainult kuupäev, siis veeru nimel järelliide *kp* (nt alguse_kp, registreerimise_kp)
 - Ärge kasutage kontseptuaalses andmemudelis atribuutide nimedena ja tabelites veergude nimedena nimesid *aeg* ja *kp*, sest need on liiga üldised ja ei anna täpsemat infot vastavate andmete tähenduse kohta.

- ✧ Üldine klassifikaatorite tabel ei sobi. Andmebaasi füüsilise disaini mudel tuleb genereerida nii, et seda ei tekiks. Kui see tabel on andmebaasi füüsilise disaini mudelis varasemast ajast olemas, siis tuleb see tabel kustutada ning muuta andmebaasi füüsilise disaini mudelit nii, et igale klassifikaatori tüübile vastab eraldi tabel, kus on kõik kontseptuaalsest andmemudelist tulenevad veerud.

Huvi (pole kohustuslik!) korral lugege selle kohta, kuidas esitada klassifikaatoreid SQL-andmebaasides lugege järgnevast bakalaureusetööst.

Vellemaa, A.S., 2015. *Mõned disainimustrid klassifikaatorite esitamiseks SQL andmebaasides*. Bakalaureusetöö. TTÜ Informaatikainstituut. [WWW] <https://digi.lib.ttu.ee/i/?3484> (04.09.2018)

- ✧ Isikunimede registreerimisel tuleb arvestada võimalusega (nii kontseptuaalses andmemudelis kui andmebaasi füüsilise disaini mudelis), et kultuurilistest eripäradest tulenevalt pole isikul näiteks perenime, nimed on ootamatult pikad või sisaldavad ootamatuid sõnu/sümboleid. Nimi *Null* ei tohiks rakendusi rivist välja lüüa.
 - <http://www.bbc.com/future/story/20160325-the-names-that-break-computer-systems>
 - <https://www.wired.com/2015/11/null/>

Soovitan väga lugeda isikunimede ja nende SQL-andmebaasides hoidmise kohta käivat bakalaureusetööd.

Jõgi, M., 2016. *Mõned disainimustrid isikunimede hoidmiseks SQL-andmebaasides*. Bakalaureusetöö. TTÜ Informaatikainstituut. [WWW] <https://digi.lib.ttu.ee/i/?7072> (04.09.2018)

Kui registreerite eesnime ja perenime, siis peaks olema kasutatud ühte järgnevatest lahendustest. Veenduge, et kontseptuaalses andmemudelis olev atribuutide kirjeldus on tehtud valikuga kooskõlas (nii atribuutide kirjelduse tabel, kui ka olemi-suhte diagrammil olevad atribuutide võimsustikud).

- ✧ Nii veerg *eesnimi* kui *perenimi* on mittekohustuslikud. Lisaks on jõustatud andmebaasis kitsendus, et vähemalt üks nendest peab olema registreeritud.
- ✧ Veerg *eesnimi* on kohustuslik ja *perenimi* mittekohustuslik.

Eesti Vabariigi nimeseadusega (<https://www.riigiteataja.ee/akt/106032015032?leiaKehtiv>) "sätestatakse füüsilise isiku nime (edaspidi isikunime) andmise ja kohaldamise põhimõtted ja kord ning Eesti kodaniku ja Eestis viibiva välismaalase (edaspidi isiku) isikunime ühtse kasutamise alused." Seadus ütleb, et "Isikunimi koosneb eesnimest ja perekonnanimest." On tõlgendamise küsimus, kas see tähendab, et eesnimi ja perekonnanimi (perenimi) on mõlemad kohustuslikud või nimetab see kõigest isikunime võimalikke komponente. Seaduse kohaselt: "(3) Välisriigis antud või kohaldatud isikunime kasutusele võtmisel järgitakse käesoleva seaduse kohaseid isikunime andmisel isikunime suhtes kehtivaid nõudeid, kusjuures: 1) isiku nõusolekul säilitatakse talle välisriigis antud või kohaldatud isikunimi, kui isikunimi on kirjutatud eesti-ladina tähtedega; (4) Välisriigi dokumendis olev nimi, mis ei ole eesnime ega perekonnanime osa, loetakse eesnime osaks ja lisatakse eesnime järele." Järeldan, et isikud kellel on näiteks vaid üks nimi (https://en.wikipedia.org/wiki/Mononymous_person), võivad säilitada oma nime ning eesnime välja on üldjuhul vaja midagi kirjutada.

- ✧ Andmebaasi tuleb kohandada tulenevalt vajadusest tuvastada süsteemi kasutajad ning lubada süsteemi kasutada ainult selleks volitatud kasutajatel ja ainult neile antud volituste piires. Järgnev lähtub eeldusest, et kasutate levinud lahendust, mille korral süsteemi kasutajad registreeritakse andmebaasis ning lõppkasutajate ja andmebaasis loodud kasutajate vahel ei ole üks-ühele vastavust. Igal kasutajal peab olema unikaalne kasutajanimi. *kasutajanimi* võib olla eraldi atribuut/veerg. Samas, kui Te registreerite isiku e-meili aadressi ja see on kohustuslik ning unikaalne (peaks nii olema), siis võite kasutada e-meili kasutajanimena ning eraldi veergu *kasutajanimi* pole vaja. Kasutajanime/e-meiliga samas tabelis peaks olema veerg *parool*.

Kui Te ei kasuta lahendust, mis esitab soola parooli räsiväärtuse osana, siis peab leiduma ka veerg *sool*, milles olevad väärtused on nt 32 märgi pikkused juhuslikult valitud märkidest koosnevad stringid. Miks see vajalik on, lugege palun siit:

<https://www.martinstoeckli.ch/hash/en/>

Vt ka <http://php.net/manual/en/function.password-hash.php>

Veerus *parool* lubatud märkide arv sõltub kasutatavast räsifunktsioonist. Kui tahate kasutada näiteprojekti lahendust, siis vaadake millised on seal parooli veeru väljapikkus. Kui Teie projektis on üldine tabel *Isik*, siis võib need lisada sinna. Vastavad atribuudid peavad olema ka kontseptuaalses andmemudelil. Kui jätkate olemasoleva andmebaasi füüsilise disaini mudeli täpsustamisega, siis võite lisada atribuudid/veerud vastavatesse diagrammidesse ning ei ole mõtet hakata kontseptuaalsest andmemudelist uuesti disaini mudelit genereerima.

Kui soovite kasutada õppejõu väljapakutavaid tehnilisi lahendusi (sellega tegeleb ülesanne 10), mille korral arvutatakse parooli räsi andmebaasisüsteemi poolt, siis looge veerud järgmiste omadustega.

PostgreSQL:

parool: VARCHAR(60) NOT NULL

Veergu *sool* ei ole vaja, sest sool integreeritakse algoritmi poolt parooli räsisse.

Oracle:

sool: VARCHAR2(32) NOT NULL

parool: VARCHAR2(128) NOT NULL

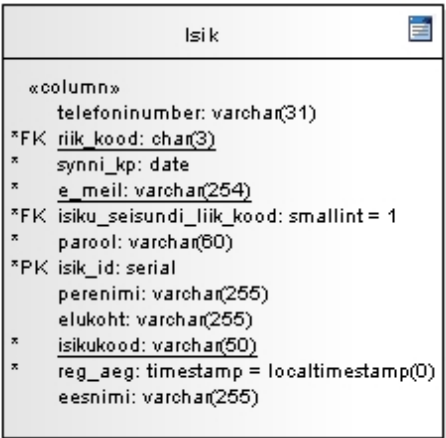
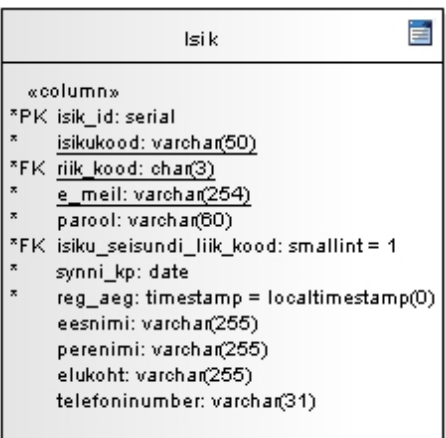
- ✧ Muutke tabelites veergude järjekorda nii,
- et kõigepealt tuleb *primaarvõtme veerg/veerud*,
 - siis *alternatiivvõtmete veerud*,
 - siis esimesse kahte kategooriasse mitte kuuluvad *välisvõtmete veerud*,
 - siis ülejäänud *kohustuslikud veerud*
 - ning kõige lõpuks ülejäänud *mittekohustuslikud veerud*.

Sama kategooria loogiliselt kokkukuuluvad veerud (nt *eesnimi* ja *perenimi*; *kasutajanimi* või *e_meil*, *parool* ja *sool*) võiks paigutada kõrvuti, isegi kui see läheb vastuollu eelnevalt nimetatud üldreeglitega.

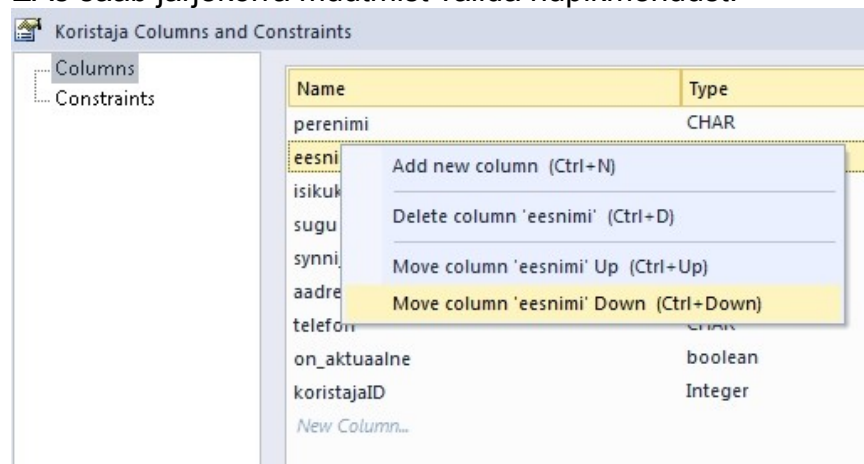
Veergude järjekord *ei mõjuta* andmete tähendust! Samas selline järjekord kiirendab mudeli ülevaatamist ning võimaldab lihtsamalt avastada puuduvaid veerge/kitsendusi. Samuti lihtsustab see andmebaasis tabelist ülevaate saamist ning selle kaudu ka päringute ja andmemuudatuse lausete kirjutamist. Veergude järjekorra kohta leidub erinevaid soovitusi: <https://stackoverflow.com/questions/2113553/ordering-columns-in-database-tables>. Võite kasutada minu pakutust erinevat järjekorda – oluline on olla järjekindel. Siiski, üldise tavana peaksid primaarvõtme veerud olema tabelis esimesed.

Veergude järjekorraga seoses tuleb arvestada, et tabeli struktuur ei ole kivisse raiutud ning tabeli eluea jooksul võib tekkida vajadus lisada tabelisse uusi veerge. Selleks on ALTER TABLE lause. Uued veerud lisatakse tabeli lõppu. Head ja kõigis süsteemides ühtemoodi toimivat veergude järjekorra muutmise lahendust ei ole. ALTER TABLE lausega seda teha ei saa. Üks variant on muuta veergude järjekorda tabelile loodud vaadetes, kuid jätta tabelites see järjekord muutmata. Kui soovida ikkagi muuta järjekorda ka tabelites, siis siin on soovitusi:

- ▲ PostgreSQL jaoks: https://wiki.postgresql.org/wiki/Alter_column_position
- ▲ Oracle 12c jaoks: <https://blogs.oracle.com/oraclemagazine/on-oracle-database-12c-%2c-part-5> (põhineb peidetud veergude funktsionaalsusel, mis lisandus Oracle 12c)

Halvem veergude järjekord	Parem veergude järjekord
 <pre> «column» telefoninumber: varchar(31) *FK riik_kood: char(3) * synni_kp: date * e_meil: varchar(254) *FK isiku_seisundi_liik_kood: smallint = 1 * parool: varchar(60) *PK isik_id: serial perenimi: varchar(255) elukoht: varchar(255) * isikukood: varchar(50) * reg_aeg: timestamp = localtime(0) eesnimi: varchar(255) </pre>	 <pre> «column» *PK isik_id: serial * isikukood: varchar(50) *FK riik_kood: char(3) * e_meil: varchar(254) * parool: varchar(60) *FK isiku_seisundi_liik_kood: smallint = 1 * synni_kp: date reg_aeg: timestamp = localtime(0) eesnimi: varchar(255) perenimi: varchar(255) elukoht: varchar(255) telefoninumber: varchar(31) </pre>

EAs saab järjekorra muutmist valida hüpikmenüüst.

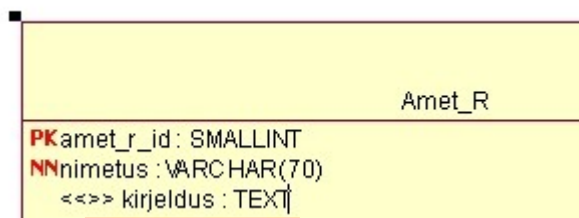


- ✎ Iga veeru jaoks tuleb valida kõige sobivam andmetüüp andmebaasi realiseerimiseks kasutatava andmebaasisüsteemi poolt pakutavate tüüpide hulgast. Kui CASE vahend ei võimalda kõige sobivat tüüpi valida, siis tuleb meeles pidada, et muudatus tuleb teha genereeritud SQL koodis (pole selle ülesande teema).

PostgreSQL: <https://www.postgresql.org/docs/10/static/datatype.html>

Oracle: https://docs.oracle.com/database/121/SQLRF/sql_elements001.htm

RR vahendis saab sobiva tüübi valikus puudumisel selle sisestada otse diagrammile:



Taustainfo! Kui muudate EA tabelite disaini mudelis tabeliga seoses andmebaasisüsteemi, siis teeb EA esialgse andmetüüpide asenduse, kuid see asendus on ebapiisav ja ebakvaliteetne. Asendus tehakse EAs kirjeldatud andmetüüpide vastavuse alusel. Kuna see on ebapiisav, siis on ka tulemus mitterahuldav. EA12 tuleb kasutatud vastavuse vaatamiseks valida: *Project => Settings => Database Datatypes => Datatype Map*

Database Datypes Mapping

From Product Name: **MSAccess** To Product Name: **PostgreSQL**

Datatype: **Text** Datatype:

Common Type: **char** Common Type:

Size

☐ None ☒ Length ☐ Precision & Scale

Defined Datypes for Databases

Product	Datatype	Size Unit	Default	Max
MSAccess	Byte			
MSAccess	Counter			
MSAccess	Currency	Precision and Scale	(15,4)	15
MSAccess	DateTime			
MSAccess	Double			
MSAccess	Integer			
MSAccess	Long			
MSAccess	Long Integer			
MSAccess	Memo			
MSAccess	OLEObject			
MSAccess	Single			
MSAccess	Text	Length	50	255
MSAccess	YesNo			

Save Close Help

- ✧ Andmetüüpide ja väljapikkuste valimisel lähtuge põhimõttest – **nii suur kui vajalik (et kõik andmed saaks registreeritud) ja nii väike kui võimalik (et andmebaasiskeemi uurijal ei tekiks andmetest eksiarvamusi).** VARCHAR(n), VARCHAR2(n), CHAR(n) tüüpide korral tuleb valida sobiv maksimaalne väljapikkus. NUMBER(p), NUMBER(p,s), DECIMAL(p,s) tüüpide korral tuleb valida täpsus (*precision*, p) ja skaala (*scale*, s).
- Lähtemudelil võib olla VARCHAR(255) – see on maksimaalne tekstivälja pikkus MS Accessis. PostgreSQLis ja Oracles on maksimaalne väljapikkus suurem ja selle valimisel tuleb arvestada reaalse maailma eripäradega. Mõelge sellele, et 255 märki on vähem kui kaks korda suurem Twitteri säutsu algsest maksimumpikkusest (140 märki) ning umbes sama pikk kui Twitteri säutsus on praegu lubatud (<https://www.theverge.com/2017/11/7/16616076/twitter-280-characters-global-rollout>). Kas tõesti saab 255 märgiga kõik vajaliku kirja panna?
 - n, p, s väärtuste valimisel arvestage piirjuhtudega (maksimaalne võimalik väärtus, mida soovite registreerida). Vajadusel googeldage (näiteks otsingustring "what is the longest email address").
 - n, p, s tuleb määrata nii suured kui vajalik ja nii väikesed kui võimalik. Lähimõeldud väärtused annavad skeemi uurijale (inimkasutaja või programm) taustainfot selles veerus oodatavate väärtuste kohta ning ühtlasi vähendavad võimalust registreerida ebakorrektsed andmeid. Samas on töötavas andmebaasis n, p ja s muutmine mõnikord tehniliselt tülikas ja suure andmete hulga korral aeganõudev ning tabelite teiste kasutajate poolt kasutamist takistav tegevus.
 - Kuna tüübi CHAR korral lisatakse väärtuse lõppu tühikuid, et saavutada maksimaalne lubatud väljapikkus, siis seda tüüpi tuleks kasutada vaid juhul, kui on teada, et väärtused on alati ühesuguse pikkusega (nt riikide koodid).
 - Andmebaas peaks võimaldama registreerida ka väga pikki isikunimesid (vaadake palun üle väljapikkused):
<http://www.historyrundown.com/top-5-people-with-the-longest-names/>
 - Liiga väikesed lubatud väärtused võivad põhjustada kannatusi, segadust, lisatööd ja mainekadu (süsteemi omanikule):
 - <https://reisile.postimees.ee/4484884/reisija-pidi-lisatasu-maksma-sest-tema-nimi-ei-mahtunud-lennupiletile>
 - Aadressi välja väljapikkuse valimisel arvestage, et ka kohanimed võivad olla pikad ning et aadress koosneb enamasti mitmest komponendist (nt linn, tänav, korter; maakond; vald; küla; talu), millest igaühel on oma nimi.
 - https://en.wikipedia.org/wiki/List_of_long_place_names
 - Rahasummade korral on levinud soovitus kasutada tüübi deklaratsiooni DECIMAL(19,4) (PostgreSQL; *decimal* ja *numeric* on sünonüümid ja viitavad samale tüübile) või NUMBER(19,4) (Oracle). Rahasummade puhul arvestage, et kaupade ja teenuste hinnad tuleb esitada *vähemalt* kahe komakoha täpsusega. Pidage silmas, et erinevatel valuutadel on ettenähtud erinev vaikimisi komakohtade arv (euro korral vaikimisi kaks kohta peale koma: <http://www.thefinancials.com/Default.aspx?SubSectionID=curformat>). Näiteks väikesed tariifid, ühikuhinnad kaupadel, mida saab osta ainult suurtes kogustes ja mõned finantstehingud nõuavad rohkem kui kaks (kolm-neli) kohta peale koma.

Kui hinnad on väga väikesed ja müüdavad kogused väga suured, siis lubage ümardamise probleemide vältimiseks kuni viis kohta peale koma. PostgreSQL pakub tüüpi *money*, kuid sellesse on sisse integreeritud valuuta tähis (sõltub serveri seadetest – apex.ttu.ee serveris \$) ning enne aritmeetikatehteid on vaja väärtus teisendada NUMERIC tüüpi. Seega soovitan *money* mitte kasutada ja tarvitada selle asemel püsikomatüüpi. Lugege lisaks:

- <https://sakala.postimees.ee/386430/euroraha-harjutab-ostjaid-tuhandikega>
- <https://stackoverflow.com/questions/27325620/decimal19-4-or-decimal19-2-which-should-i-use>
- <https://www.riigiteataja.ee/akt/111022016017>
- <http://kasulik.delfi.ee/news/kasulikselgitab/4-99-voi-5-00-kuidas-napilt-madalam-hind-meie-ostuotsuseid-mojutab?id=80146622>
 - ⤴ Kuidas hinna esimene number ja komakohad mõjutavad ostuotsuseid?
- Jälgige rahasummade ja teiste arvude korral, mille puhul arvutustes ja ümardamistes ei tohi olla vigu, et need *ei oleks* salvestatud ujukomatüüpi veerus, mille korral väärtused salvestatakse kahendsüsteemis
 - Peatükk 10 raamatus:
 - ⤴ http://www.ester.ee/record=b2888667*est
 - ⤴ <http://it-ebooks.info/book/70/>
 - <http://floating-point-gui.de/basic/>
 - Katsetage väärtuse kahendsüsteemis esitamist: <https://www.h-schmidt.net/FloatConverter/IEEE754.html>
 - ⤴ Näiteks väärtus **0.3** salvestatakse tegelikult väärtusena **0.300000011920928955078125**
- Kui Teil on tabelis veerud nagu *xxx_kood*, *nimetus*, *kirjeldus*, mis on kõik tekstitüüpi, siis pole loogiline et neil kõigil on ühesugune väljapikkus. Ilmselt on kood lühike, nimetus veidi pikem ja kirjeldus kõige pikem.
- *Täisarvu tüüpi* võtmeveergude tüübi määramisel tuleb mõelda võimalikule ridade arvule tabelis. Tüüp tuleks valida nii, et vähemalt lähema viie aasta perspektiivis poleks vaja seda tüüpi muuta. Erinevat tüüpi andmeid sisaldavates tabelites on erinev hulk andmeid. Järgnev on väga "jäme" hinnang.
 - Klassifikaatorid (nt keeled, riigid, ametikohad, kauba seisundi liigid): üksikud read kuni tuhanded read. Sobiv tüüp SMALLINT või NUMBER(1)–NUMBER(5)
 - Põhiandmed (nt isikud, kaubad, teenused, teavikud, sõiduvahendid): tuhanded kuni miljonid read. Sobiv tüüp INTEGER/SERIAL või NUMBER(5)–NUMBER(10)
 - Transaktsioonilised andmed (nt tellimused, teadmiste kontrollid, taotlemised, paiknemised): kümned tuhanded kuni sajad miljonid read. Sobiv tüüp INTEGER/SERIAL või BIGINT/BIGSERIAL või NUMBER(n), kus $n \geq 10$.
- Tekstivälja puhul, mille pikkus on üks märk, on loogilisem CHAR(1) kui VARCHAR(1). Lisaks peab olema CHECK kitsendus (vt ülesanne 3), et veerus ei tohi olla tühikutest koosnev string. Tühi string asendatakse CHAR tüübi korral tühiku(te)ga.

- ⤴ Kui Teie andmebaasis on vaja hoida pilte, siis mõelge veelkord läbi kuidas seda teha (ning vajadusel parandage ka kotseptuaalset andmemudelit). Võimalused on hoida andmebaasis järgnevat.
 - *Pildifaili* (andmebaas suurem, kuid pilte käsitletakse kui ülejäänud andmeid – saab kasutada andmebaasisüsteemi pakutavaid turvalisuse, transaktsioonide ning varundamise/taastamise mehhanisme).
 - PostgreSQL korral veeru andmetüüp BYTEA või OID; Oracle korral tüüp BLOB.
 - *Pildifaili aadressi* (andmebaas väiksem, võibolla andmete lisamise ja kuvamise mõttes esmapilgul lihtsam realiseerida, kuid turvalisuse, transaktsioonide ning varundamise/taastamise teemad nõuavad eritähelepanu, sest pildid ei ole andmebaasisüsteemi hoole all ja seega nende teemadega tegelemiseks ei saa andmebaasisüsteemi võimalusi kasutada).
 - Andmebaasis selleks tekstitüüpi veerg. Väljapikkuse valimisel tuleb arvestada failisüsteemi lubatava kataloogitee ja failinime maksimaalse pikkusega:
<https://blog.codinghorror.com/filesystem-paths-how-long-is-too-long/> või kui soovite hoida URLi, siis selle maksimaalse pikkusega
<https://stackoverflow.com/questions/417142/what-is-the-maximum-length-of-a-url-in-different-browsers>
 - Bill Karwini antimustrite raamat käsitleb failide hoidmist väljaspool andmebaasi antimustrina. Peatükk 12 raamatus:
 - ⤴ http://www.ester.ee/record=b2888667*est
 - ⤴ <http://it-ebooks.info/book/70/>
- ⤴ Igas tabelis peab olema primaarvõti, kõik alternatiivvõtmed tuleb üles leida ja jõustada UNIQUE+NOT NULL kitsendustega, kõik välisvõtmed tuleb jõustada FOREIGN KEY kitsendustega.
 - Tüüpviga on, et UNIQUE kitsendus dubleerib PRIMARY KEY kitsendust. Primaarvõti tähendab unikaalsust.
 - Kasutajanime UNIQUE kitsenduse kustutamine ja funktsioonil põhineva UNIQUE indeksi loomine toimub ülesandes 10. Seal loodav funktsioonil põhinev UNIQUE indeks aitab tagada, et kasutajanimi (nt e-meili aadress) on *tõstutundetult* unikaalne. Näiteks kui ühel isikul on e-meili aadress erki.eessaar@ttu.ee, siis teisel ei saa see olla Erki.Eessaar@ttu.ee
 - Kui kontseptuaalne andmemudel sõnastab kitsenduse, mille kohaselt peab atribuudi väärtuse unikaalsus kehtima ainult teatud tingimustele vastavate olemite korral (näiteks, et aktuaalses seisundis kaupade nimetused peavad olema unikaalsed), siis selle realiseerimiseks tuleb PostgreSQL ja Oracle andmebaasides luua *osaline unikaalne indeks*. Sellist indeksit ei saa EA ning RR vahendis kirjeldada. Seega tuleb sellise indeksi loomise kood ise kirjutada.
 - PostgreSQL: <https://www.postgresql.org/docs/10/static/indexes-partial.html>
 - Oracle:
<http://dba-presents.com/index.php/databases/oracle/41-filtered-index-equivalent-in-oracle>
- ⤴ Mõelge läbi, millised veerud on kohustuslikud (NOT NULL) ja millised mitte. Kõik sellised valikud peavad olema kooskõlas kontseptuaalse andmemudeliga. Vajadusel tuleb teha muudatusi kontseptuaalses andmemodelis. EA vaikumisi mudeliteisendus jätab tabelite kirjelduses kõik

mitte-primaarvõtme veerud mittekohustuslikuks (lubavad NULLe). Andmebaas tuleks disainida nii, et enamik veerge on kohustuslikud – seega paljudele veergudele tuleb kohustuslikkus käsitsi määrata.

- ⤴ Tuleb läbi mõelda, millistes tabelites on primaarvõtmed surrogaatvõtmed, millistes sisulised võtmed. Näiteks, kui CASE vahend pakub tabelisse *Isik* välja primaarvõtme veeru **isik_id** **INTEGER**, siis on Teie võimuses näiteks määrata, et primaarvõti on hoopis kombinatsioon **isikukood** **VARCHAR(50)** ja **isikukoodi_riik** **CHAR(3)** ning *isik_id* veerg kustutada. See omakorda viib kaskaadselt vajaduseni muuta selliste tabelite struktuuri, kus olev välisvõti peab otseselt või kaudselt viitama tabelile *Isik*. Paraku EA CASE vahendis on selliseid muudatusi väga tülikas teha ja seega Te ei pea oma projektis seda muutust tegema. **Klassifikaatorite koodid on sisulise tähendusega võtmed – neid ei genereeri süsteem, vaid sisestab teatud rollis lõppkasutaja!**
 - Kui kasutate EA, siis saate andmebaasi füüsilise disaini mudelis määrata, et PostgreSQL andmebaasis on surrogaatvõtme veeru korral "tüüp" SERIAL (see on tegelikult notatsioon, mis tingib andmebaasis INTEGER tüüpi veeru loomise, arvujada generaatori loomise ning veeru ja generaatori sidumise vaikselt väärtuse mehhanismi kaudu).
 - Töövihiku projektis on tabeli *Isik* veerg *isik_id* tüüpi SERIAL.
- ⤴ Millal kasutada primaarvõtme veeru nimes järelliidet **_id**? CASE vahend võib sellise nimega veeru lisada kõikidesse tabelitesse, kuid Teie võimuses on see veerg ümber nimetada või veerg kustutada. Seleks peaks primaarvõti rahuldama **kõiki** järgnevaid tingimusi.
 - Primaarvõti on lihtvõti.
 - Primaarvõti on surrogaatvõti, mitte sisulise tähendusega võti. Selle võtme väärtuseid hakkab genereerima süsteem. Võtmeväärtused moodustavad monotoonselt kasvava/kahaneva täisarvude jada.

Näide:

- **kaup_id** – tabeli *Kaup* primaarvõtme veerg, millesse hakkab väärtuseid genereerima süsteem.
 - ⤴ Miks ei sobi lihtsalt nimi ID? Vastuse annab <http://it-ebooks.info/book/70/> Peatükk 4 – "ID Required".
 - Siis on Teil endal ka hiljem lihtsam – arvujada generaator on vaja siduda vaid nende primaarvõtme veergudega, mis ei kattu välisvõtmega ja mille nimi lõpeb järelliitega **_id**.
- ⤴ Millal kasutada primaarvõtme veeru nimes järelliidet **_kood**? Primaarvõti peaks rahuldama **kõiki** järgnevaid tingimusi.
 - Primaarvõti on lihtvõti.
 - Primaarvõti on sisulise tähendusega võti. Sellesse veergu hakkab väärtuseid sisestama inimkasutaja.

Näide:

- **kauba_seisundi_liik_kood** – kauba seisundiklassifikaatori tabeli primaarvõtme veerg, millesse hakkab andmeid lisama piisavate õigustega inimkasutaja.
- Alternatiivne nimetamise strateegia on kasutada klassifikaatorite tabelite nimedes järelliidet **_R** (R nagu *reference data*) ning sellise tabeli primaarvõtme veeru nimes järelliidet **_r_id**.

- ⌘ Kui nimetate ümber võtmeveerge, siis tagage, et vastavalt nimetatakse ka ümber seotud välisvõtme veerud. Kui need nimed on ühesugused, siis on lihtsam kirjutada tabelite ühendamiseks mõeldud lauseid. Muudatus tuleb teha kõigepealt primaarses (välisvõtme poolt viidatavas) tabelis. RR suudab selle nime muudatuse automaatselt üle kanda sõltuvasse (välisvõtit sisaldavasse) tabelisse (eeldusel, et mudelis on välisvõti korrektselt kirjeldatud). EAs tuleb see muudatus teha ka sõltuvas tabelis käsitsi.

Näide:

[Kauba_seisundi_liik]-1-----0..*-[Kaup]

Kõigepealt tuleb teha muudatus tabelis *Kauba_seisundi_liik* ning siis tabelis *Kaup*.

- ⌘ Jälgige välisvõtme puhul, et selles osalevatel primaarse ja sõltuva tabeli veergudel oleksid ühesugused või ühilduvad andmetüübid. Andmetüüpide muutmisel muutke *primaarse tabeli* võtmeveeru tüüpi. RR ja EA suudavad selle muudatuse automaatselt sõltuva tabeli kirjeldusse üle kanda (eeldusel, et mudelis on välisvõti korrektselt kirjeldatud). Kui muudate välisvõtme veeru tüüpi sõltuvas tabelis, siis EA ei kanna seda muudatust automaatselt primaarsesse tabelisse üle. RR ei lase sellist muudatust teha.

Näide:

[Kauba_seisundi_liik]-1-----0..*-[Kaup]

Kui soovin määrata, et *kauba_seisundi_liik_kood* on tüüpi SMALLINT, mitte INTEGER, siis teen muudatuse tabeli *Kauba_seisundi_liik* kirjelduses.

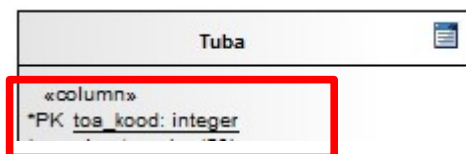
Tähelepanu: Kui kasutate PostgreSQL'i ning:

- ⌘ primaarses tabelis on võtmeveeru tüüp SMALLSERIAL, siis peab sõltuvas tabelis vastav välisvõtme veerg olema tüüpi SMALLINT,
- ⌘ primaarses tabelis on võtmeveeru tüüp SERIAL, siis peab sõltuvas tabelis vastav välisvõtme veerg olema tüüpi INTEGER,
- ⌘ primaarses tabelis on võtmeveeru tüüp BIGSERIAL, siis peab sõltuvas tabelis vastav välisvõtme veerg olema tüüpi BIGINT.

Kui määrate EAs veeru "tüübiks" SERIAL / SMALLSERIAL / BIGSERIAL, siis määrab CASE vahend selle "tüübi" automaatselt ka sellele veerule viitavatele välisvõtme veergudele. Välisvõtme veergude tüübid tuleb käsitsi asendada vastavalt tüüpidega INTEGER / SMALLINT / BIGINT.

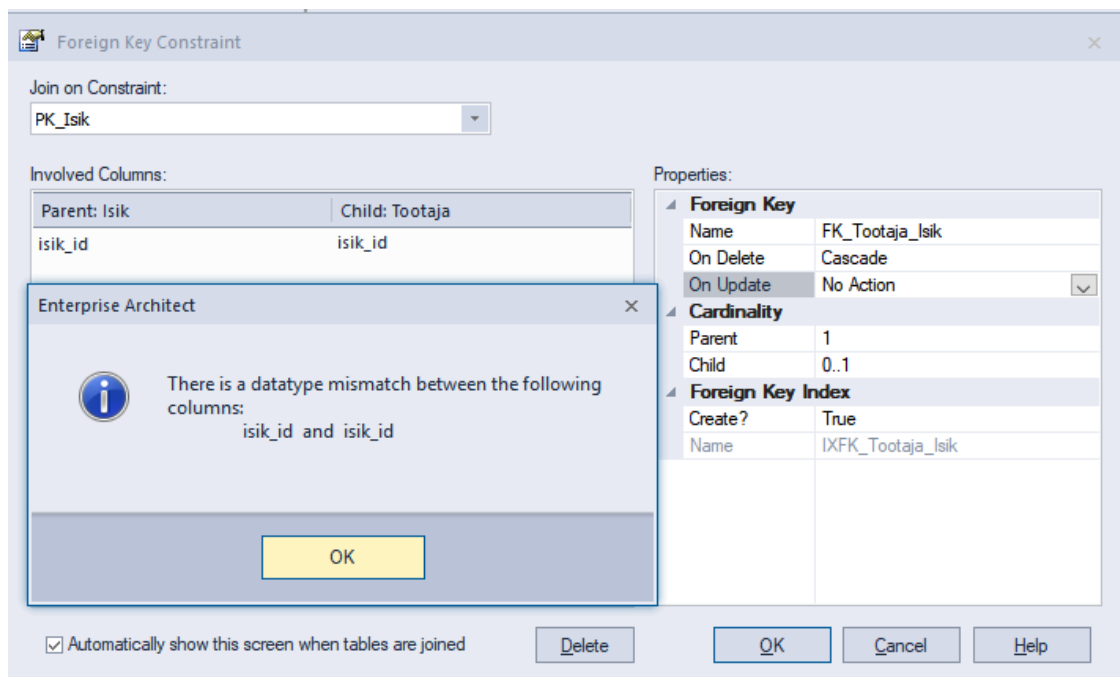
- ⌘ Ärge unustage, et võti võib olla liitvõti. Liitvõtme kirjeldamise kohta vaadake videot "Mitut veergu hõlmava UNIQUE kitsenduse kirjeldamine andmebaasi diagrammis".
- ⌘ Ärge kirjeldage üksteist dubleerivaid võtmeid. Primaarvõti tähendab juba unikaalsust. Primaarvõtme veergudele lisaks UNIQUE kitsenduse deklareerimine oleks viga. Järgnevalt on illustreeritud vea esinemist EA andmebaasi disaini mudelis. PK veeru nime ees näitab, et veerg kuulub

primaarvõtmesse. Veeru allajoonimine näitab, et see veerg kuulub UNIQUE kitsendusse.



- ▲ Välisvõtmete puhul tuleb määrata sobivad kompenseerivad tegevused.
 - Kui **sama olemi andmed on laiali mitmes tabelis**, siis neid omavahel siduvate välisvõtmete korral võiks kasutada ON DELETE CASCADE.
 - Näiteks oletame et, kaubaartiklite kohta on andmed tabelites *Kaup*, *Kauba_kategooria_omamine*, *Kauba_omadus*. Kui tahan kauba *k* andmed andmebaasist kustutada, siis pean kustutama *k* kohta käivad read kõigist nendest tabelitest. Oleks ju lihtsam, kui peaksin kustutama kauba andmed ainult tabelist *Kaup* ja süsteem hoolitseb automaatselt ülejäänud andmete kustutamise eest!
 - Kontseptuaalses andmemudelil olnud **üldistuste** alusel tekkinud välisvõtmetele ON DELETE CASCADE.
 - Kontseptuaalses andmemudelil **kompositsiooniseoste** alusel tekkinud välisvõtmetele ON DELETE CASCADE.
 - Kui välisvõti **viitab sisulise tähendusega võtmele** (klassifikaatori kood, isikukood, auto registrikood, dokumendi number, üliõpilaskood, ...), siis ON UPDATE CASCADE.
 - seda saab määrata PostgreSQLis, kuid ei saa määrata Oracles.
 - Kui välisvõtme poolt realiseeritava seosetüübi korral on püstitatud reegel, et **vanema andmete kustutamisel peavad selle laste andmed säilima**, siis on sobiv määrang ON DELETE SET NULL.
 - Kõigil ülejäänud juhtudel sobib vaikimisi määrang ON UPDATE/DELETE NO ACTION.
 - Näiteks oletame, et *isik_id* väärtused tabelisse *Isik* genereerib süsteem. Esiteks, *isik_id* väärtuseid pole kellelgi põhjust muuta. Teiseks, kui *isik_id* väärtust ikkagi muuta, siis võib tekkida olukord, et süsteem genereerib uue rea lisamisel veeru jaoks väärtuse, mille keegi on sinna juba lisanud. Sellised read jäävad tabelisse lisamata. Kokkuvõtteks, antud juhul puhul pole andmete muutmise kompenseerivat tegevust vaja (st ON UPDATE NO ACTION).

Küsimus EA kasutamise kohta: "Nimelt, juhendi järgi peaks olema primaartabelis(nt isik) veerg isik_id(serial) ning tabelis(nt Töötaja), mille välisvõti viitab sellele veerule peaks olema Integer. Viisin muudatused sisse, aga kui hakkan lisama kitsendust sellele seosele ON DELETE CASCADE, ON UPDATE NO ACTION, siis EA ei lase->ütleb, et datatype mismatch. Kuidas käituda?"



Vastus: Peate panema tabelisse *Tootaja* veerule *isik_id* tüübi SERIAL.

Peate tegema soovitud muudatuse.

Peate peate panema tabelisse *Tootaja* veerule *isik_id* tüübi INTEGER.

- ⤴ Seal kus vaja, tuleb veerule määrata vaikimis väärtus (nt välisvõtme veerule, mis viitab seisundiklassifikaatorile või veerule, kus registreeritakse iga rea korral selle loomise aeg). Samas, kui tahate näiteks registreerida, milline töötaja registreeris süsteemis millise olemi, siis välisvõtmel (töötaja_id) vaikimisi väärtust ei ole, sest välisvõtme väärtus tuleb määrata rakenduse poolt vastavalt sellele, kes sisse logis ja registreerimise tegevust läbi viis. RR vahendis loodud mudelis kirjutatakse vaikimisi väärtus *Default Value* välja ning EA vahendis loodud mudelis *Initial Value* välja.
 - Kui veeru tüüp on `TIMESTAMP`, siis see on sama kui `TIMESTAMP WITHOUT TIME ZONE` ning sellisele veerule vaikimisi väärtuse leidmiseks on funktsioon `LOCALTIMESTAMP`, mitte `CURRENT_TIMESTAMP` või `Now()`.
 - Kui Te *ei* soovi registreerida sekundi murdosi, siis kasutage `LOCALTIMESTAMP(0)` või `CURRENT_TIMESTAMP(0)`.
 - Kui Te soovite vaikimisi väärtusena registreerida ajatempli minuti täpsusega, siis kasutage järgmisi avaldisi:
 - PostgreSQLis: `date_trunc('minute', LOCALTIMESTAMP)`
 - Oracles: `trunc(LOCALTIMESTAMP, 'mi')`
 - Kui vaikimisi väärtus peaks olema hetke kuupäev, siis selle leiab funktsiooniga `CURRENT_DATE`.
 - Jälgige, et Te ei määraks staatilist vaikimis väärtust veerule, mis moodustab primaarvõtme või millel on unikaalsuse kitsendus. Vaikimisi väärtuse mõte on, et see võib paljudes ridades korduda. Kuid eelnimetatud veergudes peavad igas reas olema unikaalne väärtus. Kui soovin määrata, et uued kaubad tuleb luua seisundis koodiga 10, siis määran vaikimisi väärtuse 10 tabeli *Kaup* veerule *kauba_seisundi_liik_kood*.

- Ärge unustage, et tekstilised vaikumisi väärtused tuleb panna ülakomade vahele (nagu nt 'AKT') ning ülakomad tuleb määrata juba mudelis.
- Tõeväärtustüüpi veerule *on_nous_tylitamisega* sobib vaikumisi väärtus FALSE. 2018. aasta mais jõustus andmekaitse üldmäärus, mis karmistab oluliselt isikuandmete töötlemiseks vajaliku nõusoleku tingimusi (<http://arileht.delfi.ee/news/triniti/andmekaitsemaarus-muutub-tootlemiseks-vajaliku-andmete-omaniku-nousoleku-tingimused-karmistuvad?id=75871155>). "Määruse kohaselt kehtib nõusolek isikuandmete töötlemiseks ainult juhul kui selleks on antud selge avaldus, kinnitus (olgu siis suuliselt, elektrooniliselt või kirjalikult) või nõusolekut väljendav tegevus. Nõusolek ei saa tuleneda isiku vaikumisest või tegevusetusest. Seega on määruse mõtte kohaselt nõusolek antud nt veebisaidil kastis linnukese või risti tegemisega, kuid vabatahtliku nõusolekuga ei ole tegemist, kui linnuke või rist kastis on juba eelnevalt olemas (ja isik peaks keeldumiseks selle ise eemaldama)."

Väike andmetüüpide vastavustabel.

PostgreSQL	Oracle
SMALLINT	NUMBER(5)
INTEGER	NUMBER(10)
DECIMAL(19,4) NUMERIC(19,4)	NUMBER(19,4)
VARCHAR(100)	VARCHAR2(100 CHAR)
CHAR(3)	CHAR(3)
TEXT	CLOB
DATE	DATE
TIME	DATE
TIMESTAMP	DATE TIMESTAMP
BOOLEAN	NUMBER(1) (lisaks CHECK kitsendus, mis lubab veerus ainult väärtuseid 0 ja 1)

- ▲ Kui kontseptuaalses andmemudelil on olemitüübil rekursiivne üks mitmele seosetüüp iseendaga (näiteks igal töötajal on null või üks mentor, kes on samuti töötaja), siis see viitab vajadusele hoida andmebaasis hierarhilisi andmeid. Levinud (kui mitte levinuim) lahendus SQL-andmebaasides on panna tabelisse (nt *Tootaja*) välisvõti, mis viitab sama tabeli primaarvõtmele. Sellist disainilahendust nimetatakse külgnevusnimistuks (*adjacency list*). Kuigi sellel lahendusel on ka puuduseid, kompenseerib seda lahenduse populaarsus ja seega võrgus leiduvate juhendite ning näidete rohkus.

Huvi (pole kohustuslik!) korral lugege järgnevatest lõputöödest selle kohta, kuidas esitada hierarhilisi andmeid SQL-andmebaasides. Krönström (2015) magistritöös esitatakse struktureeritult kümme disainilahendust hierarhiliste andmete esitamiseks SQL-andmebaasides.

Muide, need lahendused ei sisalda võimalust hoida SQL-andmebaasis hierarhilisi andmeid JSON või XML tüüpi veerus, st võimalusi on veelgi rohkem. Hierarhiliste andmete esitamisest PostgreSQL andmebaasis JSON tüüpi veerus kirjutab oma lõputöös Ossipova (2016). Mõlemas lõputöös on lisaks disainilahenduste kirjeldustele ka eksperimendid nendega PostgreSQL (Krönström, Ossipova) ja Oracle (Krönström) andmebaasides.

Krönström, K., 2015. *Hierarhiliste andmete esitamine SQL-andmebaasides kolme disainilahenduse näitel*. Magistritöö. TTÜ Informaatikainstituut.
[WWW] <https://digi.lib.ttu.ee/i/?3678> (08.09.2018)

Ossipova, M., 2016. *Mitme väite ühe andmeväärtusena esitamise eelised ja puudused SQL-andmebaasides*. Bakalaureusetöö. TTÜ Informaatikainstituut.
[WWW] <https://digi.lib.ttu.ee/i/?6997> (08.09.2018)

Selle ülesande lahendamiseks saab lisainfot

- ✧ "Andmebaaside lisamaterjalid" kodulehel (<http://193.40.244.90/346>) kataloogis *Andmebaasid 1/Loengud/Lisamaterjalid* olevast dokumendist "Andmebaasiobjektide nimetamine SQL-andmebaasides",
- ✧ slaidikomplektist: **Baastabelid_systeemikataloog_IDU0230_2018.ppt**, mille leiab samuti (<http://193.40.244.90/346>)