TARTU ÜLIKOOL
Arvutiteaduste instituut
Infotehnoloogia mitteinformaatikutele õppekava

Sirje Lind

Andmete migratsiooni testimine

Magistritöö (15 EAP)

Juhendaja: Anne Villems

Tartu 2019

Andmete migratsiooni testimine

Lühikokkuvõte: Infosüsteemide uuendamise ajajärgul puutuvad infosüsteemide arendajad üha enam kokku vajadusega tõsta (migreerida) vana infosüsteemi andmed uue infosüsteemi andmebaasi. Magistritöös selgitatakse migratsiooni testimist ning leitakse võimalusi testide koostamise ja testimise lihtsustamiseks (automatiseerimiseks).

Testimise automatiseerimise vajadusest lähtuvalt käsitletakse migratsiooni **testi** päringute komplektina, kus igasse komplekti kuulub üks isevalideeruv test (päring) ja vähemalt üks vigaseid kirjeid tagastav abipäring. Sellise lähenemisviisiga saab testide käivitamisel kohese ülevaate migreeritud andmete seisust ja vigaste andmete (kirjete) detailvaate saamiseks ei ole vaja kirjutada SQL lauseid. Seega muutub vigade analüüsimine efektiivsemaks, sest väheneb täiendavate päringute koostamise vajadus.

Testide koostamise automatiseerimise eelduseks testide sarnasuse alusel testide tüüpidesse jagamine. Töös defineeriti 16 **testi tüüpi** ja iga testi tüübi jaoks koostati testi päringu mall (*template*) ja abipäringute mallid (näidispäringud) ning selgitati testi metaandmete vajadust. Lisas toodud testide tüüpide kirjeldused on näidiseks migratsiooni testijatele. Testide kirjelduste alusel saab arendada ka testide päringute koostamise generaatori.

Võtmesõnad: andmete migratsioon, testimine, andmete kvaliteet

CERCS: P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine

Data Migration Testing

Abstract: In the era of information system upgrades, information system developers are increasingly confronted with the need to upgrade (migrate) the old information system data into the database of the new information system. Master's thesis explains migration testing and finds ways to simplify (automate) test design and testing. Based on the need for automation testing, the migration test is considered as a set of queries where every element include one self-validating test (query) and at least one auxiliary query that returns invalid entries. With this approach, one can instantly view the status of migratory data when one runs the tests, and no need to write new SQL statements to get a detailed view of the incorrect data (records). Thus, the analysis of errors becomes more efficient as the need for new additional queries is reduced.

The prerequisite for automation of test preparation is the division of tests into test types based on similarity of tests. In this work, 16 test types were defined, and a template for the test query and auxiliary query templates (pattern queries) were prepared for each type of test and the need for test metadata was explained. The descriptions of the types of tests in the Appendix are an example of migration testing. A test generator can also be developed based on test descriptions.

Keywords: data migration, testing, data quality

CERCS: P170 Computer science, numerical analysis, systems, control

Sisukord

1 Sissejuhatus **4**

2 Migratsiooniprojekt **6**

2.1 Migratsiooni vajadus infosüsteemide uuendamisel 6

2.2 Migratsiooni projekti arhitektuur 6

2.3 Migratsiooniprojekti etapid 8

2.4 Migratsiooniprojekti sarnasus andmelao projektiga 10

2.4.1 Andmeladu 10

2.4.2 Andmelao ja migratsiooniprojekti sarnasus 11

3 Migratsiooni testimine **12**

3.1 Andmelao testimine migratsiooni testimise eeskujuna 12

3.2 Migratsiooni testimine komponentide kaupa 14

3.2.1 Lähtebaasi andmete testimine (1-A) 14

3.2.2 Sihtbaasi andmete testimine (2-A) 15

3.2.3 Lähtebaasi (1-A) ja sihtbaasi (2-A) andmete testid 16

3.2.4 Sihtbaasi andmemudeli valideerimine (2-B) 17

3.2.5 Andmete laadimise testimine (3) 18

3.2.6 Rakenduste testimine (4) 19

3.3 Migratsiooni testimine projekti etappide lõikes 19

4 Migratsiooni testimiseks sobivate lähenemisviiside näited **22**

4.1 Väljavõtete võrdlemine 22

4.2 Objektiga seotud tunnuste kontrollimine 22

4.3 Objektiga seotud tunnuste kaudne kontrollimine 23

4.4 Objektiga seotud tunnuste kontrollimine räside abil 24

4.5 Vastavuse testimise automatiseerimine 25

4.6 Ühiktestide lähenemisviisi kasutamine 26

4.7 Ühe migratsiooniprojekti testimise kogemus 27

5 Migratsiooni testide päringud **28**

5.1 Testide päringute koostamine 28

5.2 Testide päringute analüüsimine ja testide tüübid 29

5.2.1 Testide päringute analüüs 29

5.2.2 Vigaste testide tulemuste analüüsimiseks abipäringud 30

5.2.3 Testide tüübid 31

5.2.4 Testide metaandmed 32
5.2.5 Testide tüüpide kasutamine 34
6 Testimise korraldus 36
6.1 Ühe objekti andmete testimine 36
6.2 Ettevalmistatud testide kasutamine 36
6.3 Testide haldus 38
6.3.1 Näide: testid on PL/SQL protseduurides, abipäringud failides 38
6.3.2 Näide: testide päringud on andmebaasis 39
Kokkuvõte 40
Viidatud kirjandus 42
LISAD 44
I Testi metaandmed 44
II Testide tüüpide päringute mallid 45
1 Testi tüüp NOT_NULL 45
2 Testi tüüp IS_NULL 47
3 Testi tüüp STATIC_VALUE 49
4 Testi tüüp IN_VALUES 51
5 Testi tüüp UNIQUE_VALUE 53
6 Testi tüüp CODE 55
7 Testi tüüp VALUE_BETWEEN 57
8 Testi tüüp METRIC_COUNT 59
9 Testi tüüp EQUAL_ROWS 60
10 Testi tüüp MATCH_KEY 61
11 Testi tüüp MATCH_DISTINCT_KEY 63
12 Testi tüüp MATCH_HASH 65
13 Testi tüüp MATCH_COLUMN 67
14 Testi tüüp MATCH COUNT 69
15 Testi tüüp MATCH_SUM 72
16 Testi tüüp INFORMAL 74
III Testi tüüpide kaupa metaandmete kasutamise kokkuvõte 75
IV Litsents 76

1 Sissejuhatus

Eestis on infosüsteeme loodud üle 15 aasta. Ettevõtetes on infosüsteeme, mis ei vasta enam kaasaegse tehnoloogia nõuetele ega ajas muutunud ärilistele vajadustele. Kui leitakse, et olemasoleva infosüsteemi täiendamine/parandamine ei ole mõistlik ja otsustatakse luua uus infosüsteem vana infosüsteemi asemele, siis tekib vajadus ka vanade andmete ülekandmiseks uude infosüsteemi.

Seega puutuvad infosüsteemide arendajad ja testijad üha enam kokku pärandsüsteemist[1] andmete uude infosüsteemi laadimise (migreerimise) vajadusega ning tehtud töö testimisega. Andmete migratsiooni testimine erineb infosüsteemi testimisest: migratsiooni testimisel on vaja anda hinnang kogu andmebaasi sisu korrektsusele. Infosüsteemide testijatel võib puududa kogemus kogu andmebaasi andmete täielikkuse ja nõuetele vastavuse testimisel.

Seni on andmete migratsiooni teemat ka vähe käsitletud, kuid see muutub praegusel ajajärgul üha olulisemaks.

Andmete migreerimisel on palju sarnasusi andmelao projektides andmete laadimisega, samas andmelaonduse projektid oma regulaarse andmete laadimise vajadusega on oluliselt keerukamad. Kuna andmelao toimimine on ettevõtete jaoks ärikriitise tähtsusega, siis on ka andmelaonduse valdkonda rohkem uuritud ning arendatud andmete laadimise tarkvarasid.

Andmelaonduse tarkvarad sobivad ka migratsiooniprojektis andmete laadimiseks ja tulemuste testimiseks. Nende võimsate programmide kasutamine ei ole majanduslikult põhjendatud, sest andmete migratsioon vanast andmebaasist uude andmebaasi on **ühekordne tegevus.** Samuti on ka keerukate programmide kasutusele võtmisel õppimise aeg ühekordse tegevuse jaoks liiga suur.

Eeltoodu tõttu vaadeldakse andmete migreerimist piiratud **ressursside tingimustes**, kus soovitakse testida **SQL[2] vahenditega**.

Töö eesmärk on selgitada migratsiooni testimise vajadust ning pakkuda lähenemisviise testimise lihtsustamiseks. Tööle seatakse kolm eesmärki:

1. **täpsustada migratsiooni testimise vajaduse ulatust;**
2. leida testimise lähenemisviis, mis lihtsustaks **testimist** ja vigade analüüsimist;
3. leida võimalusi **testide koostamise lihtsustamiseks (automatiseerimiseks).**

Töös kasutatakse põhilise uurimismeetodina andmelaonduse testimise kogemuse võrdlemist migratsiooni testimise vajadustega ning tehakse järeldusi migratsiooni testimise jaoks.

Töö koosneb kuuest peatükist, millest esimene on sissejuhatus. **Teises peatükis** tutvustatakse migratsiooniprojekti ja selle etappe ning selgitatakse andmelao ja migratsiooniprojekti sarnasusi ning erinevusi. **Kolmandas peatükis** võetakse aluseks andmelao projekti testimise vajadus ja kirjeldatakse migratsiooni testimise vajadust ning luuakse testimisvajaduse seos migratsiooni projekti etappidega. **Neljandas peatükis** tutvustatakse erinevate allikate põhjal migratsiooni ja andmelaonduse testimise näiteid ning tuuakse välja soovitused migratsiooni testimiseks. **Viiendas peatükis** analüüsitakse migratsiooni testide päringute struktuuri ning

[1] Pärandsüsteem (Legacy System) on kasutusel olev vana süsteem, mida ei täiustata ega uuendata, sest tulevane süsteem on juba saadaval või väljatöötamisel ja muul edasiarendamisel võib tekitada ühilduvuse probleeme.

[2] SQL (*Structured Query Language*) on struktureeritud päringukeel andmebaasiga suhtlemiseks (Cybernetica, 2019; „Struktuurpäringukeel“, 2016).

defineeritakse testide päringute koostamise lihtsustamiseks **testide tüübid.** Selgitatakse, kuidas testide tüüpidesse jagamine lihtsustab testide päringute koostamist. **Kuuendas peatükis** selgitatakse testimise lähenemisviise. **Lisades** on ära toodud kõikide testi tüüpide kirjeldused koos näidispäringutega.

2 Migratsiooniprojekt

Migratsioon on protsess, mille käigus teisaldatakse andmed ühelt platvormilt/formaadilt teisele platvormile / formaadile.

2.1 Migratsiooni vajadus infosüsteemide uuendamisel

Paljude infosüsteemide arendamisel on kätte jõudnud faas, kus olemasolev infosüsteem ei rahulda enam kaasaja vajadusi. Kasvavate tehnoloogiliste nõuete rahuldamiseks ja süsteemi paindlikkuse parandamiseks on vaja infosüsteeme ajakohastada (Althani & Khaddaj, 2017, lk 154).

Olemasolevate infosüsteemide (pärandsüsteemide) uuendamine on väga ressursimahukas ja enne infosüsteemi uuendamist kaalutakse erinevaid moderniseerimise viise.

- Ärikriitiline ja väga halva kvaliteediga pärandsüsteemi asemele soovitatakse uue infosüsteemi loomist (Althani & Khaddaj, 2017, lk 156).
- IT-ressursside haldamise paindlikkuse tõstmiseks kaalutakse ka pilveteenustele üleminekut („What Is ...“, i.a).
- Kaasaegsemale platvormile/tehnoloogiale üleminek võimaldab parandada jõudlust, kulutasuvust, paramaid liidestusi. („What Is ...“, i.a).

Sõltuvalt valitud moderniseerimise viisist kasutatakse erinevaid migreerimise tüüpe („Types of Migration ...“, 2019).

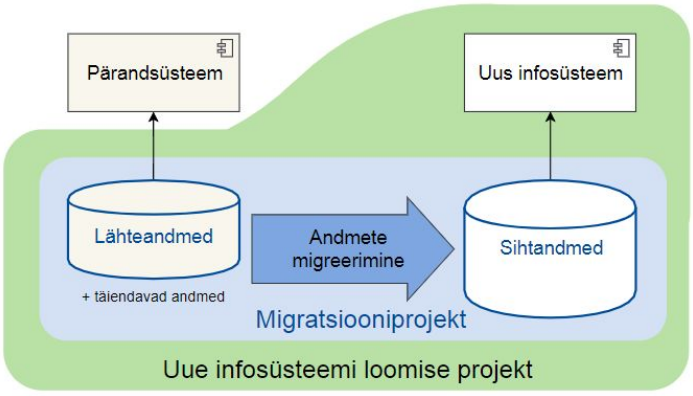
- **Rakenduste uuendamine**, kus rakendus tõstetakse teise keskkonda (nt pilve) või uuendatakse rakenduse tehnoloogiat.
- **Andmebaasi migratsioon**, kus olemasoleva rakenduse või uue rakenduse jaoks kantakse vanast andmebaasist andmed uude andmebaasi. Andmed kas kopeeritakse või teisendatakse vastavalt uuele andmestruktuurile.
- **Serveri migratsioon**, kus serveris asuvad andmed tõstetakse ühelt serverilt teisele.
- **Operatsioonisüsteemi migratsioon**, kus rakendus tõstetakse ühelt operatsiooni-süsteemilt teisele (nt . WINDOWS → LINUX).

Selles töös käsitletakse uue infosüsteemi loomise projekti, kus koos infosüsteemiga muudetakse ka andmestruktuuri ning **andmed migreeritakse pärandsüsteemi andmetest.**

2.2 Migratsiooni projekti arhitektuur

Migratsiooniprojekti saab käsitleda uue infosüsteemi loomise projekti alamprojektina (vt joonis 2.1), sest migratsiooniga seonduvad tööd on suhteliselt sõltumatud uue infosüsteemi arendusest ning esindatud on kõik tarkvaraarenduse etapid: analüüs, arendus, testimine. Lähtuvalt migratsiooniprojekti sisust kasutatakse selles töös mõisteid järgnevalt.

- **Pärandsüsteem** (*legacy system*) on vana infosüsteem (ainult pealisehitus ilma andmeteta).
- **Lähteandmed**, **lähtebaas**, **lähtetabelid** viitavad vana infosüsteemiga seotud andmetele ja andmeobjektidele.
- **Sihtandmed**, **sihtbaas**, **sihttabel** viitavad uue infosüsteemiga seotud andmetele ja andmeobjektidele.



Joonis 2.1 Migratsiooniprojekt on uue infosüsteemi loomise alamprojekt ja migratsiooniprojekti sisuks on andmete migreerimine lähtebaasist sihtbaasi.

Uue infosüsteemi arendamise projektis on soovitav migratsiooni alamprojekt planeerida **võimalikult varajases faasis**, mis võimaldab uut infosüsteemi testida juba eluliste (migreeritud) andmetega (Sadalage, 2015).

Migratsiooniprojekti eesmärk on tagada, et uue infosüsteemi juurutamisel toimub lähtebaasist vanade andmete ülekandmine uue infosüsteemi jaoks korrektselt. See tähendab, et migratsiooni käivitakse toodangukeskkonnas ainult üks kord ja migratsiooni õnnestumise tagamiseks testitakse migratsiooni toimumist arendus- ja testkeskkondades.

Tavaliselt on migratsiooniprojektis vähemalt kolm keskkonda.

- **Toodangukeskkond** on reaalne infosüsteemi toimimise keskkond. Migratsiooni arenduse eesmärk on tagada, et projekti lõpus toodangukeskkonnas migratsiooni käivitamine annab kohe korrektse tulemuse.
- **Testkeskkonnas** testitakse migratsiooni toimimist. Testkeskkonnas on soovitav kasutada lähteandmetena võimalikult elulähedasi andmeid (nt toodangukeskkonna lähteandmete koopiat või selle alamhulka). Suuremates projektides ja/või konfidentsiaalseid andmeid sisaldavates projektides võib olla vajadus mitme testkeskkonda järele: eraldi keskkonnad arendusmeeskonnale ja tellija testijatele.
- **Arenduskeskkonnas** toimub migratsiooni arendamine ja esmane testimine. Arenduskeskkonnas võib olla väiksem lähtebaas, kuid andmed peaksid katma toodangubaasis esinevad erijuhtumid.

Migratsiooniprojekti edukuse aluseks on äripoole kaasamine, planeerimine ja õigesti valitud lähenemisviis (Horward, 2011).

Migratsiooniprojektis osalevad nii arendaja, kui ka tellija (Haller, Matthes, & Schulz, 2012, lk 169–170):

- **Projekti sponsor** määrab projekti ulatuse ja tagab rahastuse.
- **Tellija ekspertide meeskond** (edaspidi äripool) tunneb äriprotsesse ja pärandsüsteemi toimimist. Eksperte kaastakse konsultantidena ja testijatena.
- **Uue rakenduse tulevased kasutajad** (edaspidi lõppkasutajad)**,** kes tunnevad uue süsteemi vajadusi ja nõudeid. Neid kaasatakse testijatena.
- **Arendusmeeskond** vastutab andmete ja nõuete **analüüsimise**, migratsiooni nõuetekohase **arendamise** eest ning arenduskeskkonnas **migratsiooni käivitamise** ja **testimise eest.**

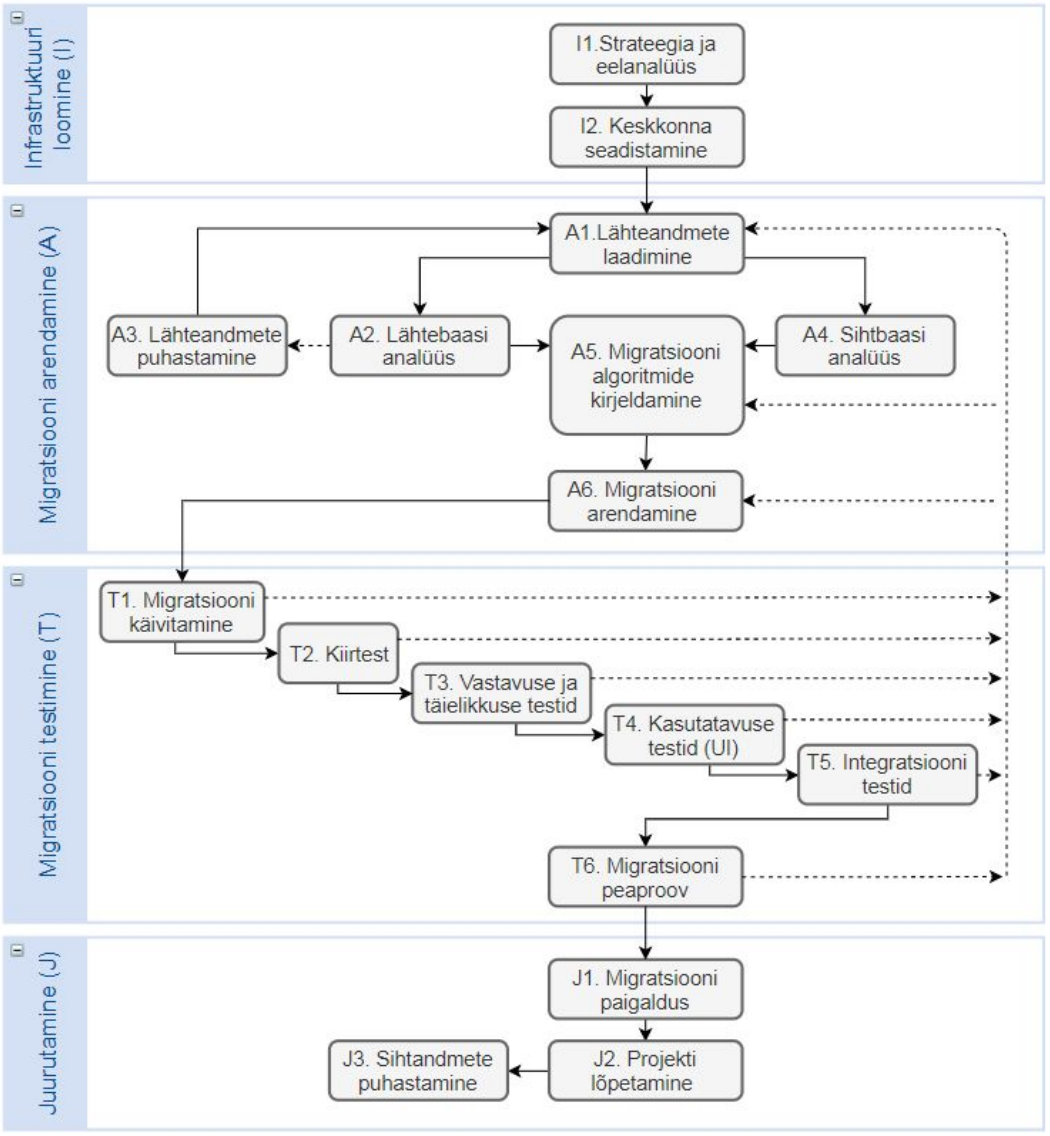
- **Infrastruktuuri meeskond** (edaspidi administraatorid) tagab keskkondade (võrgud, serverid, andmebaasid) seadistamise ja migratsiooni käivitamise erinevates keskkondades.
- **Väline audiitor** kaasatakse projekti sponsori soovil või projekti nõuetest tulenevalt.

Selles töös käsitletakse arendusmeeskonna testijate tegevusi migratsiooniprojektis.

2.3 Migratsiooniprojekti etapid

Migratsiooniprojekti tööd saab jagada traditsioonilistesse tarkvara arenduse etappidesse.

Selles peatükis tutvustatakse Halleri (Haller jt, 2012, lk 170–174) kirjeldatud migratsiooni protsessi mudelit (vt joonis 2.2).



Joonis 2.2 Andmete migratsiooniprojekti etapid ja etappide tegevused. Katkendliku joonega on kujutatud testimise tulemusena vigade arendusse tagasi suunamised. Allikas: (Haller jt, 2012, lk 170) joonise alusel.

Haller eristab migratsiooniprojektis nelja etappi.

- Esimene etapp (joonisel 2.2 I) on projektorganisatsiooni ja tehnilise **infrastruktuuri loomiseks** (*Initialization*).
- Teine etapp (A) on **migratsiooni arendamiseks** (*Migration Development*). Nad vaatlevad analüüsi arenduse etapi osana.
- Kolmas etapp (T) on testimiseks
- Neljandal etapil (J) toimub migratsiooni paigaldus ja migratsiooni projekti lõpetavad tegevused.

Projekti põhilised tööd toimuvad arenduse ja testimise etappides ning vastavalt testimise tulemustele võib olla vajadus korduvalt arenduse etapi juurde tagasi pöörduda.

Järgnevas kirjeldatakse migratsiooniprotsessi tegevused etappide kaupa. (Haller jt, 2012, lk 170–174).

Infrastruktuuri loomise etapi (I) tegevused

I1. **Migreerimise strateegia ja eelanalüüsiga** täpsustatakse migratsiooni läbiviimise strateegia: keskkondade vajadus, andmemahud, kasutatavad ressursid, projekti ajakava, vaadatakse üle piirangud ning hinnatakse riske. Täpsustakse töökorraldust (nt vigade haldus, koodi hoidla).

I2. **Keskkonna seadistamisega** pannakse üles vajalikud keskkonnad, luuakse arendajatele-testijatel juurdepääsud.

Migratsiooni arenduse etapi (A) tegevused

A1. **Lähteandmete laadimisel** luuakse arendamiseks ja testimiseks vajalik lähteandmestik. Otsustatakse milliseid andmeid kasutada arendamisel/testimisel, (nt vana infosüsteemi arenduse andmeid või toodanguandmete koopiat või selle osa).

A2. **Lähtebaasi analüüsil** õpitakse tundma lähteandmete struktuuri ja sisu ning testitakse lähteandmete andmekvaliteeti lähtudes uue infosüsteemi nõuetest.

A3. **Lähteandmete puhastamisel** parandatakse vajadusel lähteandme andmekvaliteedi vead.

A4. **Sihtbaasi analüüsil** tutvutakse uue infosüsteemi nõuetega ja veendutakse, et sihtbaasi andmemudel sobib lähtebaasi andmete ülekandmiseks.

A5. **Migratsiooni algoritmid kirjeldatakse** eelnevalt analüüsitud lähte- ja sihtandmete analüüside (A2, A4) põhjal ning vormistatakse migratsiooni skriptide arendamise alusdokument (spetsifikatsioon).

A6. **Migratsiooni arendamisel** koostatakse migratsiooni skriptid.

Testimise etapi (T) tegevused

T1. **Migratsiooni käivitamine**.

T2. **Kiirtestidega** tuvastatakse migratsiooni õnnestumine.

T3. **Täielikkuse ja vastavuse testidega** (*Completeness and Type Correspondent Test*) hinnatakse sihttabelite täidetuse ja üle kantud andmete korrektsust;

T4. **Kasutatavuse testidega** (*Processability tests*) kontrollitakse testlugude alusel kasutajaliidese kaudu andmete kasutatavust.

T5. **Integratsiooni testidega** (*Integration tests*) kontrollitakse teiste rakendustega koos toimimist; kas erinevaid rakendusi/liideseid hõlmavad kasutusjuhtumid toimivad algusest lõpuni ootuspäraselt (*end-to-end testing*).

T6. **Migratsiooni peaprooviga** veendutakse migratsiooni täielikus korrektsuses. Migratsiooni peaprooviks tekitatakse toodangukeskkonnaga võimalikult sarnane keskkond (sh riistvara, toodanguandmete koopia). Vajadusel pöördutakse arenduse juurde tagasi.

Juurutamise etapp (J) tegevused

J1. **Migratsiooni paigaldus** toodangukeskkonda.

J2. **Projekti lõpetamine**.

J3. **Sihtandmete korrastamise** etapp on vajalik juhul, kui migratsiooni või migratsiooni testimise tarbeks on sihtbaasi lisatud ajutisi tabeleid, tunnuseid või PL/SQL[3] skripte.

Järgnevas uuritakse mida ja millistes migratsiooni etappides on vaja testida.

2.4 Migratsiooniprojekti sarnasus andmelao projektiga

Migratsiooniprojektis on palju sarnaseid etappe andmelaonduse (*Data Warehouse*) projektide laadimise etappidega. Seetõttu saab migratsiooniprojektides kasutada andmelaonduse projektide kogemust, kuid infosüsteemide arendustiimide analüütikud ei ole tihti kursis nende paremini uuritud valdkondade heade tavadega. Selles peatükis uuritakse, mida andmelaonduse projektide testimise kogemustest saaks kasutada migratsiooniprojektide testimisel.

2.4.1 Andmeladu

Andmeladu on andmete hoidla (andmebaas), mis on disainitud operatiivandmetest[4] ja muudest välistest allikatest andmete koondamiseks kokkuvõtlikul kujul ning andmemudel sobib ettevõtte andmete analüüsiks ja aruannete koostamiseks („Data Warehouse“, i.a).

Andmelao põhikomponendid on (vt joonis 2.3):

- erinevate infosüsteemide lähteandmed,
- andmete laadimise protsess,
- andmelao andmebaas
- aruandluskeskkond.

Lähtesüsteemide andmebaasid on loodud lähtudes operatiivtegevuste vajadustest (kiire lugemine, kiire kirjutamine, lühiajalised aruanded, võib puududa ajaloo säilimine) ning tavaliselt on andmemudel kolmandal normaalkujul (st sisaldab palju võõrvõtmetega seotud tabeleid).

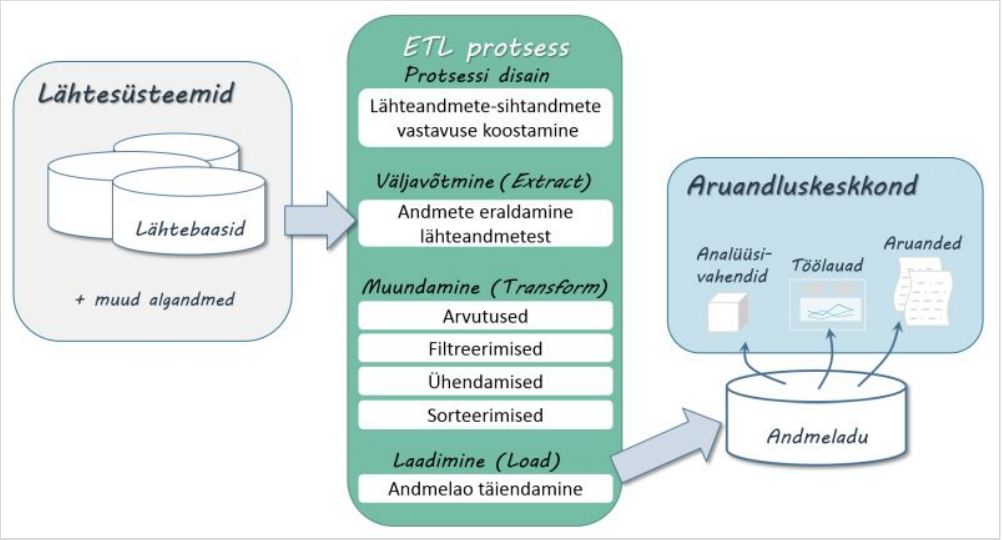
Andmelao andmemudel on denormaliseeritud ja see lähtub vajadusest optimeerida andmemudelit suurtelt andmetelt paljude päringute kiireks tegemiseks (Rouse, 2017).

Erineva arhitektuurilise lähenemisviisi tõttu on lähtebaaside ja andmelao andmemudelid väga erinevad ning vajalikud teisendused tehakse andmete laadimisel lähtebaasidest andmelattu.

[3] **PL/SQL (***Procedural Language extension to Structured Query Language*) on laiendus SQL keelele, mis võimaldab protseduurilist lähenemisviisi kasutada relatsioonilistes andmebaasides.

[4] **operatiivandmed on** igapäevaseks operatiivseks tegevusteks disainitud infosüsteemide andmed.

Andmelao projekti kõige olulisem osa on andmete laadimine (ETL - *Extract Transform Load*), millega tagatakse aruandluse jaoks vajalike andmete teisendamine ja **regulaarne uuendamine**. Regulaarselt kantakse andmelattu juurde operatiivsüsteemidest andmete muutusi. Andmete laadimisel (vt joonis 2.3) eraldatakse (*Extract*) uued ja täiendamist vajavad andmed, tehakse teisendused (*Transform*) ning lisatakse andmed andmelattu (*Load*).



Joonis 2.3 Andmelao põhikomponendid on erinevate allikate lähteandmed, andmete laadimise protsess ETL, andmelao andmebaas ja sellele ehitatud aruandluskeskkond. Allikas: („ETL Concepts ...“, 2015) alusel.

Andmelaoprojekti andmete laadimise (ETL) eesmärk on aruandluskeskkonna rakendustele kvaliteetsete andmete kogumine lähtebaasidest.

2.4.2 Andmelao ja migratsiooniprojekti sarnasus

Kui migratsiooniprojektis toimub andmete **laadimine üks kord**, siis andmelao projektis toimub andmete **laadimine regulaarselt**. See seab andmelao laadimise skriptidele kõrgemad nõudmised jõudluse, tõrkekindluse, skaleeruvuse, seadistatavuse ja laadmiste muudatuste haldamise osas.

Andmete laadimise protsessi põhikomponentide (andmetest arusaamise, andmete laadimise, andmete testimise) osas on migratsiooni ja ETL projektid väga sarnased. Andmelao projektides on regulaarne testimine ja vigade kiire avastamine väga olulised aruandluse usaldusväärsuse tagamisel.

Kuna andmelaonduse projektides on **andmete laadimise jõudluse** ja **andmekvaliteedi teemad ärikriitilised**, siis on andmelaonduse valdkonna meetodite ja tööriistade arendamine olnud ka teadlaste ja praktikute huviobjektiks. Nii laadimise kui ka testimise jaoks on arendatud tööriistu ja spetsiaalsed tarkvarasid (QuerySurge[5], Informatica[6], Talend[7] jt).

Ühekordselt käivitatava migratsiooniprojekti jaoks ei ole kallite tarkvarade ostmine alati põhjendatud. Seetõttu vaadeldakse selles töös andmelaonduses kasutavaid testimise tehnikaid ning kohandatkse neid andmete migratsiooniprojekti jaoks.

[5] QuerySurge https://www.querysurge.com/
[6] Informatica https://www.informatica.com/services-and-training.html
[7] Talend https://www.talend.com/

3 Migratsiooni testimine

Tarkvara testimine on tegevus, millega kontrollitakse, kas tegelikud tulemused vastavad oodatavatele tulemustele ja veendutakse, et tarkvara on veavaba. Tarkvara testimisega tuvastatakse („What is Software ...“, i.a):

- arenduse ja andmete vead,
- puuduvaid nõuded (nõuete spetsifitseerimisel käsitlemata jäänud vajadused) ,
- tegelike vajadustega vastuolus olevad nõuded.

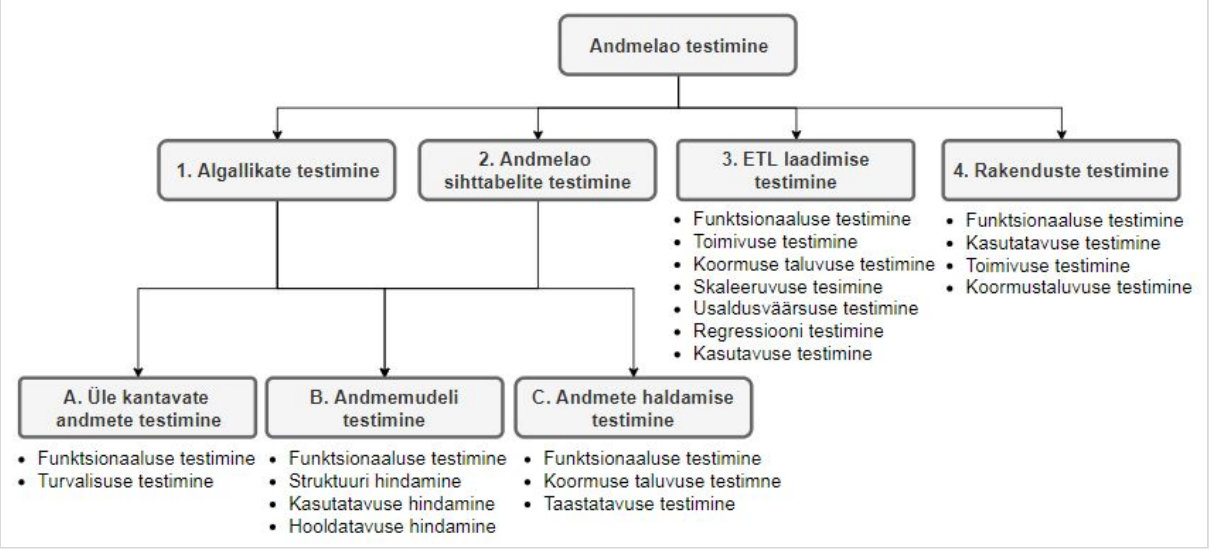
Andmete migratsiooni testimise eesmärk on tagada, et andmete ülekandmine on toimunud korrektselt ja vastab uue infosüsteemi nõuetele.

Selles peatükis võrreldakse migratsiooniprojekti testimise vajadust andmelao projekti testimisega (ptk 3.1) ja selle järeldusena kirjeldatakse migratsiooni testimise mudel ning kirjeldatakse testimise eesmärke komponentide kaupa (ptk 3.2). Selgitatakse ka etappide kaupa migratsiooniprojekti testimist (ptk 3.3).

3.1 Andmelao testimine migratsiooni testimise eeskujuna

Andmelaol ja migratsioonil on sarnased põhikomponendid: lähtebaas, laadimine, sihtbaas ja sihtbaasi poolt kasutatav rakendus (infosüsteem või analüüsikeskkond). Kuna mõlemal juhul on eesmärgiks tagada rakenduse toimimiseks korrektsed andmed, siis võetakse migratsiooni testimise mudeli koostamisel eeskujuks andmelao testimise mudel.

Homayouni (Homayouni, 2018, lk 14–44) kirjeldab andmelao komponentide testimise mudelis (vt joonis 3.1) iga komponendi (lähtebaasi, laadimise, sihtbaasi ja rakenduse) jaoks vajalikud testimise liigid. Lähteandmete (1) ja andmelao (2) testimise sarnasuse tõttu on joonisel nende komponentide testimise vajadust kujutatud koos, sest mõlema baasi korral on vaja testida andmete korrektsust (A), andmemudeli sobivust (B) ja andmete haldamise toimivust (C).



Joonis 3.1 Andmelo komponentide testimine. Allikas: (Homayouni, 2018, lk 15).

Järgnevas võrreldakse Homayouni kirjeldatud (Homayouni, 2018, lk 14–44) andmelao testimise vajadusi migratsiooni testimise vajadustega ning tuuakse välja erinevused.

Järgnevates lõikudes on kasutatud viiteid joonisel olevatele numbritele (nt 1-A, 2-C, 3).

1. Lähtebaasi andmete testimine (1-A)
   - Andmelao projektides on **lähtebaasi testimise** eesmärk tuvastada andmekvaliteedi vead ja leida meetodid kvaliteedivigade tekke vältimiseks.
   - Migratsiooni korral on vaja tagada andmete sobivus ainult migratsiooni hetkel. Seega testimise eesmärk on veenduda ülekantavate andmete kvaliteedis ja vigade korral otsustada, kuidas käituda vigaste andmetega migratsiooni hetkel.
2. Sihtbaasi andmete testimine (2-A) on andmelao ja migratsiooniprojektis sarnased, mõlemal juhul veendutakse sihtbaasi andmekvaliteedi nõuete täidetust.
3. Andmemudeli testimine
   - Andmelao projektis on **sihtbaasi andmemudel** (2-B) aruandluse loomise alustala. Andmemudel peab sobima valitud andmelao lähenemisviisiga[8] ja valitud aruandlustarkvaraga[9]. Aruandlusvajaduse tõttu võidakse teha muudatusi ka **lähtebaaside andmestruktuurides** (1-B).
   - Migratsiooniprojektis lähtebaasi andmemudelit (1-B) ei muudeta ja seega puudub vajadus lähtebaasi andmemudelit valideerida[10]. Sihtbaasi andmemudel (2-B) on loodud uue infosüsteemi vajaduste järgi ja migratsiooniprojektis valideeritakse, kas andmemudel sobib lähteandmete ülekandmiseks sihtbaasi.
4. Andmete haldamine
   - Andmelao projektis on vaja veenduda, et lähteandmete haldus (1-C) välistab (ka tulevikus) kvaliteedivigade tekke ja vigade kandumise regulaarsete laadimistega andmelattu.
   - Migratsiooniprojektis kantakse üle andmete hetkeseis ja andmete haldamise verfitseerimine[11] ei ole oluline (2-C).
5. Andmete laadimise testimine (3)
   - Nii andmelao, kui ka migratsiooniprojekti korral on andmete ülekandmise korrektsuse kontrollimine kõige olulisem ja kõige mahukam etapp. Kuna migratsiooni käivitatakse üks kord ja käsitsi, siis ei ole migratsiooni testimisel vaja andmelao testimisele omaseid tõrkekindluse, laadimise jõudluse ja laadimise hallatavuse teste.
6. Rakenduste testimine (4)
   - Andmelao projektides kasutatakse enamasti pakendatud aruandlustarkvara ja seetõttu ei ole testimise põhirõhk rakenduse toimivuse testimisel vaid andmelao andmete õigsuse kontrollimisel ja valitud tarkvara juurutamise testimisel.
   - Migratsiooniprojektil eraldi rakendust ei ole. Uue infosüsteemi rakenduste testimise põhiline eesmärk uue infosüsteemi arenduse korrektsuse testimine. Testimisel migreeritud andmete kasutamisega saab tuvastada ka migreerimise vigu.

Siin kirjeldatud erinevuste ja andmelao komponentide testimise skeemi (vt joonis 3.1) põhjal koostati migratsiooni komponentide testimise skeem (vt joonis 3.2 järgmises peatükis).

[8] Andmelaonduses on kasutusel kaks erinevat lähenemisviisi andmemudeli ja aruandluse üles ehitamisele: Inmoni ja Kimballi oma (Rangarajan, 2016) .

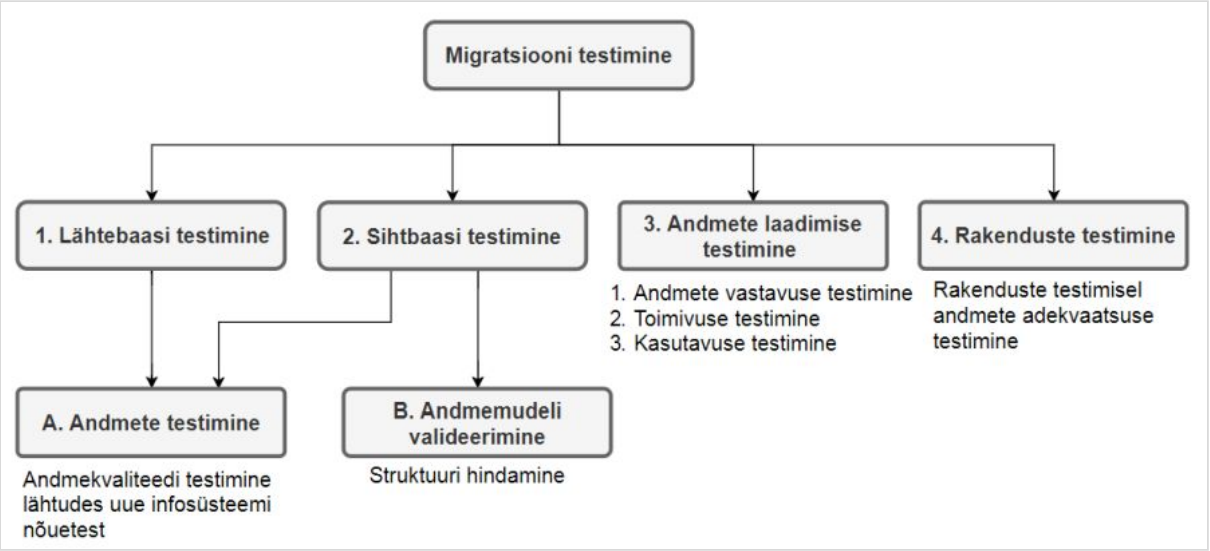
[9] **pakendatud analüüsivahendid** - Andmelao baasil aruannete loomise tarkvarad, nt Tableau, WebFocus, Qlikview jt.

[10] **valideerimine** - tegevus, millega kontrollitakse lõppkasutajate vajaduste ja ootuste täidetust („Verification vs Validation“, 2019).

[11] **verifitseerimine -** tegevus, millega kontrollitakse arendatu vastavust spetsifitseeritud nõuetele („Verification vs Validation“, 2019) .

3.2 Migratsiooni testimine komponentide kaupa

Selles peatükis on kirjeldatakse põhjalikumalt migratsiooni **komponentide testimise mudeli** (vt joonis 3.2) põhjal migratsiooni testimist.



Joonis 3.2 Migratsiooni komponentide testimine. Joonise eeskujuks on Homayouni andmlao komponetide testimise skeem (vt joonis 3.1) (Homayouni, 2018, lk 15)

Migratsiooni komponentide testimise eesmärgid on järgmised:

- **Lähtebaasi andmete testimisel** (1-A[12]) kontrollitakse vajalike andmete olemasolu, nende kvaliteedi vastavust uue infosüsteemi nõuetele.
- **Sihtbaasi andmemudeli valideerimise** (2-B) eesmärk on kontrollida, et lähtebaasi andmestruktuurid on sobivad andmete üle kandmiseks.
- **Sihtbaasi andmete testimisel** (2-A) kontrollitakse andmete vastavust uue infosüsteemi nõuetele.
- **Andmete laadimise testimisel** (3) kontrollitakse andmete ülekandmise korrektsust, laadimise toimivust.
- Migreeritud andmetega **rakenduste testimisel** (4) veendutakse migreeritud andmete korrektsuses ja uue infosüsteemi nõuetele vastavuses.

Järgnevalt kirjeldatakse nende komponentide testimist põhjalikumalt.

3.2.1 Lähtebaasi andmete testimine (1-A)

Lähtebaasi andmekvaliteedi kontrollide eesmärk on tagada pärandsüsteemi andmete ülekandmine uude andmebaasi nii, et oleks täidetud uue infosüsteemi nõuded.

Analüüsi etapis (A2) tuvastatakse üle kandmist vajavad andmed. **Andmekvaliteedi kontrolle on vaja teha ainult üle kantavatele andmetele**. Näiteks kui on otsustatud üle kanda ainult kehtivad kirjed, siis ei ole vaja teha kontrolle minevikus kehtinud kirjetele.

[12] Selles peatükis viidatakse sulgudes migratsiooni projekti etappidele joonisel 2.2 (A1-A6; T1-T6) ja migratsiooni komponentide testimisele joonisel 3.2 (1-A, 2-A, 2-B, 3-1,3-2,3-3,4).

Andmetega tutvumisel hinnatakse andmete kvaliteeti uue infosüsteemi nõuetest lähtuvalt. Näiteks kui uues infosüsteemis on ISIKUKOOD kohustuslik, siis tuleb kontrollida, et lähtebaasis on ISIKUKOOD alati olemas.

Lähtebaasi andmekvaliteedi **testide koostamisel kasutada allikatena**:

- andmete analüüsil tuvastatud mittekorrektseid väärtuseid ja/või seoseid;
- uue infosüsteemi seosed või nõuded, mille tagamiseks on mõistlik teha andmete parandused pärandsüsteemis.
- pärandsüsteemi dokumentatsioonist või äripoolega vestlusest tuvastatud seosed, mis olid lähtebaasis tehniliste kontrollidega katmata.

Testimise eesmärk on testida **lähteandmete kvaliteeti toodangukeskkonnas.** Erinevate keskkondade andmete kvaliteet võib olla erinev ja seetõttu tuleb toodangukeskkonnas testimiseks koostada testid teemade kohta, mis tunduvad olulised. Arendus- ja testkeskkondades andmekavliteedivead segavad testimist, kuid nende parandamine ei ole tingimata vajalik.

Toodangukeskkonnas leiduvate andmekvaliteedi vigade korral on vaja hinnata vea **mõju uuele infosüsteemile**:

- kas **viga takistab** andmete laadimist (nt unikaalsuse viga);
- **viga** takistab uue infosüsteemi toimimist st viga **on kriitiline**;
- **viga on häiriv** (nt täpitähtede vale kuvamine).

Lähtudes vea mõjust ja ulatusest on lähtebaasi andmekvaliteedi vigade korrigeerimiseks mitu võimalust: (Haller jt, 2012, lk 172).

- vead **parandatakse pärandsüsteemi kaudu** lähtebaasis;
- vea mõju **korrigeeritakse migratsiooni skriptidega** (nt laialt levinud sobimatute väärtuste massiline asendamine);
- **vead parandatakse** hiljem **uues infosüsteemis** rakenduse kaudu (nt üksikud erandjuhtumid, mille töötlemine migratsiooni skriptidega oleks liiga kulukas).

Tuvastatud vead vaadatakse üle koos äripoolega ning täpsustatakse iga vea prioriteeti, määratakse parandajad ja parandamise ajagraafik.

Lähteandmete kvaliteedi varajane testimine toodangukeskkonnas on vajalik, et vigade korral oleks aega lähteandmetes parandusi teha või migratsiooni algoritme muuta.

Lähteandmetes võib leiduda ka vigu, mida on testidega raske avastada. Näiteks tekstide **õigekirja vead**, väärtuste esinemised **valedes veergudes** või **ebaühtlane andmevorming** (‘USA’ *vs.* ‘Ameerika Ühendriigid’). Selliste vigade märkamisel on soovitav lasta äripoolel vead üle vaadata ja vajadusel lähtebaasis parandada. Samuti võiks uuele infosüsteemile üleminekul süsteemselt üle vaadata olulisemad tekstid (näiteks klassifikaatorite väärtused või tekstid erinevates keeltes).

3.2.2 Sihtbaasi andmete testimine (2-A)

Sihtbaasi andmete testimise eesmärk on kontrollida migreeritud andmete vastavust uue infosüsteemi **andmekvaliteedi nõuetele** ning **hinnata andmemahtude vastavust ootustele**.

Andmekvaliteedi testimine

Testid koostatakse uue infosüsteemi nõuete ja ärireeglite alusel sihtbaasi tabelite väärtuste kontrollimiseks.

Sihtbaasi andmete testimisel tuleb tähelepanu pöörata järgmistele teemadele.

- Testida tunnuseid, mille väärtused ei pärine lähtetabelitest vaid täidetakse migreerimise ajal nõuetes kirjeldatud väärtusega. Näiteks täidetakse STAATUS kõigi kirjete korral väärtusega 'KEHTIV', sest üle kantakse ainult kehtivad read.
- Testida tunnuseid, mis leitakse/arvutatakse teiste sihttabeli tunnus(t)e põhjal.
- Testida tunnuste vaheliste väärtuste sobivusi. Näiteks maakonna ja riigi väärtuste sobivus.
- Testida tabeleid, kus andmed koondatakse mitmest tabelist või mitme migratsiooni sammuga.

Kui migratsiooni skriptid on valminud uue infosüsteemi arendamise alguses, siis võivad andmestruktuure mõjutavad nõuded muutuda/täieneda veel infosüsteemi arendamise käigus. Seetõttu võib tekkida vajadus kuni tarkvara tootestamiseni[13] ka migratsiooni skriptide ja/või testide muutmiseks/lisamiseks.

Migratsiooniprojektis kasutatud sihtbaasi andmekvaliteeditestide baasil saab luua uue infosüsteemi andmekvaliteedi testide komplektid, et hilisema infosüsteemi hoolduse käigus regulaarselt andmekvaliteeti monitoorida (Horward, 2011).

Prognoositavate mahtude hindamise mõõdikud

Andmekvaliteedi testidega kontrollitakse sihtbaasis olevaid andmeid, aga nii ei tuvastata lähteandmete puudumisest või valede lähteandmete kasutamisest tingitud vigu. Seetõttu on soovitav defineerita mõned kontrollmõõdikud.

- Kirjeldada mõõdikutena olulisemate tabelite **oodatavad kirjete arvud**. Näiteks kui on teada, et hetkel on toodangukeskkonnas töötajate arv 510, siis võiks kontrollida töötajate arvu kuulumist vahemikku 500 - 520.
- Koostada olulisemate faktide kohta **agregeeritud kontrolltunnused.** Näiteks laos olevate kaupade hindade summa prognoositav vahemik või osakondade arv.

Mõõdikuid on soovitav kasutada migratsiooni järgsetes kiirtestides (vt joonis 2.2 T2), et saada kiire hinnang laadimisetulemusele.

3.2.3 Lähtebaasi (1-A) ja sihtbaasi (2-A) andmete testid

Lähtebaasi ja sihtbaasi andmete testid on sarnased, sest mõlemate testide korral koostatakse test ühe andmebaasi andmetel.

Uue infosüsteemi nõuete põhjal testide koostamisel lähtutakse võimalikest andmekvaliteedi probleemidest. Woodalli (Woodall, Oberhofer, & Borek, 2014, lk lk4) koostanud kvaliteedi probleemide nimekirja eeskujul on koostatud andmekvaliteedi nõuete nimekiri.

- Kohustuslik tunnus peab olema täidetud.
- Tunnuse väärtused peavad olema unikaalsed.
- Väärtus peab olema täitmata (NULL). Näiteks vahetult peale migreerimist peavad sihtbaasis olema tühjad migratsiooniskriptidega mittetäidetud tunnused.
- Tunnuse väärtus peab olema etteantud loetelus.

[13] **tootestamine** - tarkvara paigaldamine toodangukeskkonda (*LIVE* keskkonda) .

- Tunnuse väärtus peab jääma etteantud vahemikku. Näiteks ALGUS peab olema väiksem migreerimise ajast.
- Tunnuste vaheliste väärtuste sobivusi. Näited:
    - tunnus A ja tunnus B ei tohi olla korraga täidetud, aga üks neist peab olema täidetud.
    - Kuupäevade võrdlused ALGUS < LÕPP.
- Võõrvõtmete kontrollid. Näited:
    - töötajate tabelis kasutatud ISIK_ID on isikute tabelis olemas.
    - kasutatud klassifikaatori kood on klassifikaatorite tabelis olemas.

Kui lähteandmed on testitud, siis võib jääda mulje, et sihtbaasi nõuetele vastavuse teste pole vaja. Lähteandmete testid on olulised lähtebaasi andmekvaliteedi vigade **varaseks avastamiseks.** Testid koostatakse pigem vea hüpoteesi korral ja vea ilmnemisel saab varakult otsustada vea käitlemise viisi üle. Sihtbaasi testid peavad olema põhjalikumad ning nendega tulevad lõpuks välja ka lähtebaasi kvaliteedivead, kuid siis on vähem aega vigadele reageerimiseks.

3.2.4 Sihtbaasi andmemudeli valideerimine (2-B)

Andmelaonduse projektis on andmemudeli valideerimise eesmärk kontrollida, et andmemudel järgib valitud andmelao metoodika andmemudelit, vastab aruandlustarkvara vajadustele ning aruandluse nõuete spetsifikatsioonidele.

Migratsiooniprojektis on sihtbaasi andmemudeli valideerimise eesmärk kontrollida, et uue infosüsteemi nõuete alusel loodud andmestruktuur arvestab ka pärandsüsteemi andmete ülekandmise vajadusega. Seega migratsiooniprojekti raames tuleb uues andmemudelis valideerida neid tunnuseid, mis täidetakse lähteandmete väärtuste põhjal.

Sihtbaasi andmemudeli valideerimisel tuleb tähelepanu pöörata järgmistele teemadele.

1. Sihtbaasis on olemas kõikide üle kantavate andmete jaoks tunnused.
2. Sihtbaasi **tunnused on süntaktiliselt korrektsed** (*syntactic validity*).
    2.1. Lähteandmed on sama andmetüübiga kui sihtandmed või on lähteandmed on teisendatavad sihttabeli tunnuste tüüpidele sobivaks (*Attribute data type match*).
    2.2. Sihttabelites agregeeritud tunnuste tüübid on sobilikud. Näiteks täisarvude keskmine võib olla täisarv või komakohaga arv.
    2.3. Sihttabeli **andmetüüpide pikkused on sobivad** (*Attribute length match*), ka liidetavate/teisendatavate tunnuste korral. Näiteks sihttabelis peab lubatud komakohtade arv olema suurem või võrdne lähtebaasi komakohtade arvuga.(Wei & Chen, 2014, lk 2–3)
    2.4. Sihttabeli **tunnuste piirangud on sobivad** (*Attribute boundary match)*.
3. Sihtbaasis on tabelite vahelised **seosed sobivad.** Näiteks kui lähtebaasis on üks-mitmele seos, siis peab ka olema võimalus sihtbaasi mitu väärtust salvestada.
4. Sihtbaasis täitmiskohustusega tunnuste (NOT NULL) jaoks on lähtebaasis andmed olemas.
5. Sihtbaasi ja lähtebaasi aegadega seotud tunnuste täpsused (kuupäeva või kellaaja täpsus) on sobivad.
6. Sihtbaasi võtmetunnusteks (*primary key*) planeeritud tunnuste väärtused on lähtetabelis unikaalsed.
7. Sihtbaasi tunnuste nimed vastavad andmete sisule. Näiteks kas antud kontekstis on SEOSE_ALGUS ja KEHTIVUSE_ALGUS sama tähendusega.

8. Sihtbaasi tunnuste nimed ja tunnuste tüübid peaksid olema kooskõlas. Näiteks kui tunnuse nimi on ISIK_ID, siis peaks andmetüüp olema numbriline, ning lähtetabelis peaks ka olema vastav tunnus numbriline, mitte tekstiline.

Andmestruktuuride valideerimisel avastatud vastuolude lahendamisel võib olla vajadus suhelda äripoolega. Vastuolude lahenduseks on ettepanekud arhitektile **uue infosüsteemi andmemudeli muutmiseks** või uued nõuded **migratsiooni skriptide täiendamiseks.**

Valideerimise tulemuseks on sihtbaasi **andmemudeli kinnitamine.**

3.2.5 Andmete laadimise testimine (3)

Migratsiooniprojektides on andmete laadimise (migratsiooni) testimise eesmärk on veenduda, et andmed on üle kantud õigesti ja ei toimu andmete kadumist ega moondumist. See on migratsiooni testimise kõige olulisem osa.

Kuna andmelaonduse projektides toimub laadimine ajatatud protsessina regulaarselt, siis andmelao projektides on lisaks andmete vastavuse kontrollimisele suurem rõhuasetus laadimise tõrkekindluse, skaleeruvuse, usalduse, kasutatavuse testimisel.

Andmete vastavuse testimine

Vastavuse testidega (*balancing tests*) kontrollitakse, et vajalikud lähteandmed jõuavad sihtbaasi, ei toimuks andmete kadumist ja muutumist laadimise protsessis. Testimisel võrreldakse lähte- ja sihttabelites väärtuseid ning leitakse erinevused. (Homayouni, 2018, lk 34).

See on migratsiooni testimise kõige olulisem osa, millega kontrollitakse andmete laadimise täielikkust (*completeness*), järjepidevust (*consistency*).

Täielikkuse testimisel kontrollitakse: (Homayouni, 2018, lk 50–52):

- lähte- ja sihtbaasis **kirjete arvu võrdsust** (*Record count match*), kui andmed on migreeritud üks ühele.
- lähte- ja sihtbaasis **unikaalsete väärtuste** arvu võrdsust (*Distinct record count match*), kui migreerimisel on kasutatud agregeerimist või üks mitmele teisendamist.

Järjepidevuse testimiseks kontrollitakse objektiga seotud tunnuste (*attribute*) korrektsust. Järjepidevuse testidega kontrollitakse (Homayouni, 2018, lk 52–54):

- lähte- ja sihttabeli **tunnuste väärtuste vastavust** (*Attribute value match*);
- lähteallika kitsenduste põhjal kirjeldatud sihttabeli **tunnuste kitsenduste sobivust** (*Attribute constraint match*): unikaalsust, primaarvõtit, võõrvõtmed, seosed;
- lähteallika piiride põhjal sihttabelis väärtuse **min-max piirides püsimist** (*Outliers match)*
- lähte ja sihttabeli **tunnuste väärtusest arvutatud kontrolltunnuste võrdsust** (sh *Average match*).

Tähelepanu peaks pöörama olukordadele, kus sihtbaasi andmete kandmisel toimub väärtuste jaotamine mitmetesse tabelitesse või kasutatakse agregeerimist. Näiteks ‘isikuga’ seotud andmete korral kontrollida aadresse, lastega seoste arvu/õigsust.

Vastavustestid koostatakse (etapis A5 koostatud) migratsiooni spetsifikatsiooni põhjal. Vastavustestide päringud ei tohi kattuda **migratsiooni skriptide päringutega**, sest sellisel juhul töödeldakse testimisel andmeid samal viisil, kui andmete laadimisel ja testimisega ei suuda tuvastada vigu (Haller, 2009, lk 76).

Toimivuse testimine

Migratsiooni toimivuse testimisega (*Performance testing*) verifitseeritakse migratsiooni käivitamise protsessi.

Fikseeritakse migreerimise **kestus,** hinnatakse kestuse vastavust ootustele ning vajadusel optimeeritakse migratsiooni skripte. Migratsiooni kestuse jälgimine annab ka esmase hinnangu laadimise toimimisele. Näiteks kui laadimise aeg on oluliselt erinev samas keskkonnas eelnevatest käivitamise aegadest, siis on alust arvata, et midagi on algandmetes muutunud.

Migratsiooni laadimine peab olema **jälgitav** ja laadimise ajal peab väljastama infot protsessi edenemisest. See võimaldab aru saada, kas laadimine on hangunud või toimib tavapärasest aeglasemalt.

Kasutatavuse testimine

Migratsiooniprojekti korral on kasutusel mitu keskkonda: näiteks arenduskeskkond, tooteandmetega sarnasete andmetega testkeskkond ja toodangukeskkond. Erinevates keskkondades võivad olla erinevad seadistused (näiteks andmeskeemide nimed). Kasutatavuse testimine tähendab migreerimise käivitamist testkeskkondades juhendite põhjal. Testimise tulemiks on vajadusel soovitused migratsiooni konfigureeritavuse suurendamiseks või juhendite täiendamiseks.

3.2.6 Rakenduste testimine (4)

Uues infosüsteemis on rakenduste testimise põhiline eesmärk uue funktsionaalsuse testimine.

Lõppkasutajate poolt rakenduse funktsionaalsuse testimisel migreeritud (eluliste) andmetega on mitu eesmärki:

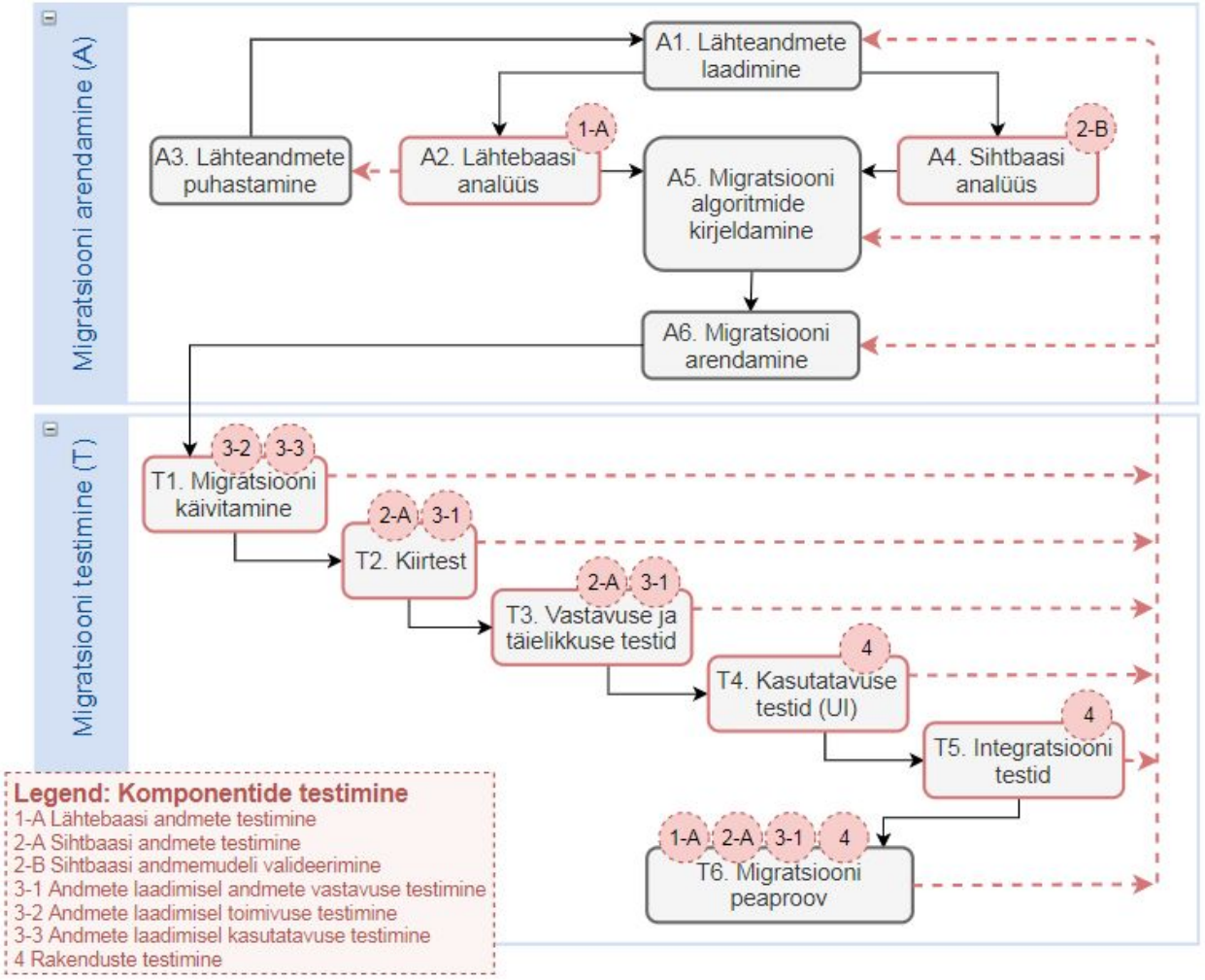
1. Migreeritud andmetega testimisel on kergem märgata **infosüsteemi arendamise vigu**.
2. Uue infosüsteemi kaudu migreeritud andmete nägemisel võidakse märgata **migreeritud andmete puudujääke,** mis on analüüsi etapil olid tähelepanuta jäänud.
3. Integratsiooni testide abil veendutakse, et teiste andmeallikate/liidestega koostöö toimib. Seega veendutakse, et **võtmetunnused on adekvaatselt migreeritud**.

Migratsiooni testimine puudutab neist kahte viimast punkti. Osalt tulevad migratsioonivead välja juhuslikult muude testimiste käigus (punkt 2), kuid selles etapis on migratsiooni testimise põhiliseks viisiks on üksikobjektide võrdlemine. Testimiseks valitakse erinevaid objekte, võrreldakse infosüsteemides nendega seotud andmete kuvamist. Näiteks valitakse üks isik, võrreldakse selle isiku andmeid ja seoseid mõlemas infosüsteemis. Erinevuste korral peab tuvastama, kas viga oli uue infosüsteemi arenduses või migreeritud andmetes.

Testimine toimub lõppkasutajate poolt ning **testimise tulemusena** võivad tekkida **uued nõuded** migratsiooni skriptidele või lähtebaasi andmekvaliteedile.

3.3 Migratsiooni testimine projekti etappide lõikes

Eelpool kirjeldatust on näha, et migratsiooni testimine ei piirdu ainult migratsiooni käivitamise järgse sihtbaasi väärtuste testimisega. Selles peatükis täpsustakse komponentide testimise vajadust (vt joonis 3.3) eelpool kirjeldatud migratsiooniprojekti etappides (joonis 2.2). Analüüsi tulemust kirjeldatakse migratsiooni testimisvajaduse joonisel 3.3.



Joonis 3.3 Migratsiooni komponentide testimine projekti etappide kaupa. Selle joonisega luuakse seosed migratsiooni etappide (joonis 2.2) ja komponentide testimise (joonis 3.2) vahel.

Kui testimise tulemusena leitakse viga, siis vea parandamine suunatakse arenduse etappi (joonisel punased nooled). Vea põhjuse analüüsi tulemusena võivad parandamist vajada migratsiooni skriptid (A6), migratsiooni spetsifikatsioon (A5) või lähteandmed (A3 → A1).

Lähteandmete tutvumise ja analüüsifaasis (A2) koostatakse ka lähtebaasi andme-kvaliteeditestid. **Lähteandmete testimine** (1-A) on teistest migratsiooniprojekti etappidest sõltumatu. Kuna lähteandmete puhastamine võib võtta aega, siis on soovitav varakult testida ka toodangukeskkonnas. Kui infosüsteemi arendamise käigus lisanduvad täiendavaid nõuded lähtetabelite kvaliteedile, siis lisatakse ka teste. Testimist korratakse ka hiljem, kui toodangukeskkonnas on kvaliteedivigu parandatud.

Sihtbaasi andmemudeli valideerimine (2-B) on ühekordne tegevus ja see toimub enne migratsiooni algoritmide kirjeldamist.

Esmakordsetel **migratsiooni käivitamisel** (T1) veendutakse **migratsiooni toimivuses** (3-2), et migratsiooni edenemine on jälgitav ja võimalike hilisemate käivitamise hangumise või katkemise korral tagastab piisavalt infot. Hilisematel käivitamistel jälgitakse migratsiooni kestust.

Esmakordsetel migratsiooni käivitamisel (T1) erinevates keskkondades on vaja veenduda **migratsiooni kasutatavuses (3-3)** erinevates keskkondades.

Vahetult pärast migreerimist on vaja veenduda, et migratsioon toimis. Kuna täielik testimine võib võtta tunde, siis **kiirtestiga** (T2) ülevaate saamiseks koostatakse väike testide komplekt, mis sisaldab olulisemad vastavusteste (3-1) ja mahtude teste (2-A).

Testimise etapi kõige põhjalikum ja ajakulukam osa on lähte- ja sihtbaasi **vastavuse testimine ja täielikkuse testimine** (T3). Selles etapis tuleb testida ülekantud andmeid (3-1) kui ka kontrollida, et uue süsteemi kvaliteedi nõuded on täidetud (2-A).

Kasutatavuse testimisel (T4) testitakse lõppkasutaja poolt andmeid rakenduse (4) kaudu.

Integratsioonitestimisel (T5) testitakse rakenduse kaudu (4) infosüsteemi funktsionaalsusi, mis kasutavad teisi andmeallikaid või mis kasutavad liideseid teiste süsteemidega.

Kui infosüsteem on tootestamiseks valmis ja migratsiooni korrektsuses ollakse veendunud, siis vahetult enne tootestamist tuleb teha toodanguandmete koopialt **migratsiooni peaproov** (T6). Enne migratsiooni käivitamist veendutakse lähtebaasi kvaliteeditestidega, et toodangu-andmete kvaliteet vastab nõuetele (1-A). Veendutakse sihttabeli andmete olemasolus ja täielikkuses (3-1, 2-A). Teostatakse ka integratsioonitestid (4).

Migratsiooni testimise kõige ajamahukamad tegevused on lähte- ja sihttabelite vastavuste testimised (3-1) ja andmekvaliteedi testimised (1-A, 2-A). Nende sammude testimine tähendab väga paljude päringute tegemist ning nende päringute tulemuste analüüsi. Järgnevas peatükis (ptk 4) tutvustatakse erinevate artiklite põhjal testimise efektiivsust tõstvaid lähenemisviise vastavustestide koostamiseks, testide päringute genereerimiseks, testimise protsessi korraldamiseks.

4 Migratsiooni testimiseks sobivate lähenemisviiside näited

Selle peatüki eesmärk on tutvuda erinevate andmeladude ja migratsiooni testimise lähenemisviisidega. Iga alapeatüki lõppu on lisatud kirjeldatud lähenemisviisist otseselt või kaudselt tulenevad **soovitused testide koostamiseks** või **testimise korraldamiseks**.

4.1 Väljavõtete võrdlemine

Vastavustestide enamlevinud viisid on **väljavõtete võrdlemine** ja **välistavad päringud.**

Väljavõtete võrdlemisel (*Sampling*) („Sampling ...“, i.a) tehakse lähte- ja sihtbaasis päringud ning tulemusi võrreldakse manuaalselt. See meetod on äärmiselt ajamahukas ja sobib ainult siis, kui laadimise käigus pole väärtusi teisendatud.

Välistavad päringute (*Minus Queries*) („Minus ...“, i.a) korral tehakse kaks päringut, mis tagastavad tabelite erinevused: ainult sihttabelis või ainult lähtetabelis olevad kirjed. See meetod on ressursimahukas ja sihttabelis lubatud dublikaatide korral annab mittekorrektseid tulemusi.

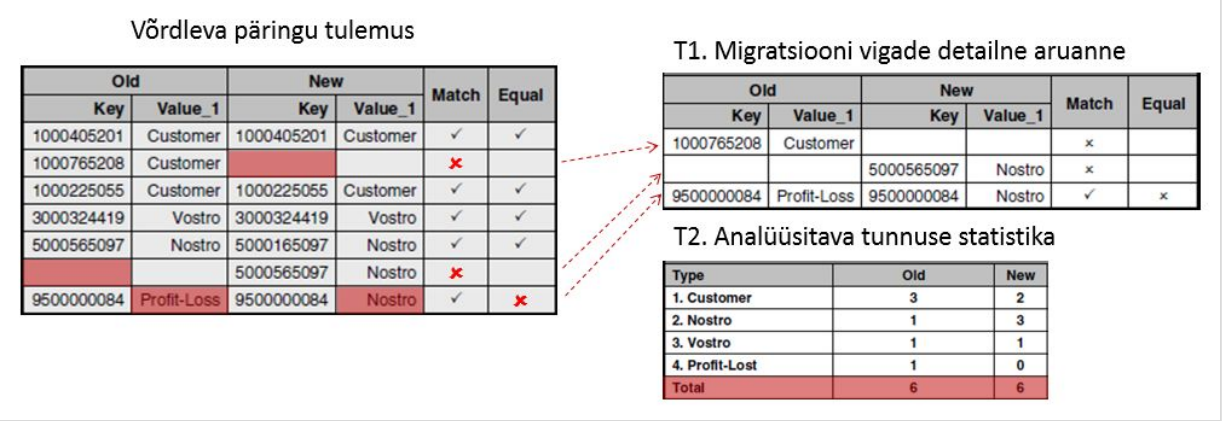
Need meetodid ei sobi testimisel kiire tulemuse saamiseks, aga need meetodid on asjakohased vigase tulemusega testide vea põhjuste analüüsimisel (Homayouni, 2018, lk 3)**.**

Soovitus 4.1.1. Testimise kiire tulemuse saamiseks vältida ajamahukaid võrdlevaid meetodeid. Pigem kasutada isevalideeruvaid teste (vt soovitus 4.6.1) mis annavad kohe hinnangu kontrollitavate nõuete täidetuse kohta.

Soovitus 4.1.2. Vigase tulemusega testide vea põhjuste analüüsimiseks kasutada võrdlevaid päringuid.

4.2 Objektiga seotud tunnuste kontrollimine

Haller (Haller, 2009, lk 74–77) soovitab migratsiooni kontrollimisel lisaks tavapäraste tunnuste olemasolu kontrollidele kontrollida ka kõigi **oluliste objektide asjakohaselt valitud tunnuste adekvaatsust**. Vaatame joonise 4.1 abil pangakonto ja konto tüübi migratsiooni kontrollimist. Joonisel vasakul on tehtud päring, kus kontonumber (veerg KEY) on olnud ühendavaks tunnuseks ja kontrollitavaks atribuudiks on konto tüüp (VALUE1).



Joonis 4.1 Pangakonto tüübi kontroll. Vasakul joonisel on detailpäringu tulemus. Paremal testimise aruanne, mis koosneb kahest osast: migreerimise vigadest ja statistikast. Allikas: (Haller, 2009, lk 75–76) jooniste alusel.

Kui teha nendel andmetel ainult väärtuste olemasolu summaarne kontroll, siis võib jääda mulje, et migratsioon on olnud korrektne, sest vanas ja uues tabelis on kontonumbrite koguarvud ühesugused (count(old.key) = count(new.key) = 6) ja ka konto tüüpide koguarvud on ühesugused (count(old.value_1) = count(new.value_1) = 6).

Kuid joonisel vasakul olevalt päringu tulemusest on näha:

- et leidub kontonumbreid, mida teises tabelis ei ole (read 2 ja 6). Nende korral antakse sobivuse kontrolltunnusele MATCH väärtuseks FALSE;
- et leidub kontonumber, mille korral kirje on mõlemas tabelis olemas, kuid kontotüüp on tabelites erinev (7.rida). Sellisel juhul antakse kontrolltunnusele EQUAL väärtuseks FALSE.

Selle päringuga õnnestub tuvastada migratsiooni viga ning Haller soovitab aruande esitada kahes osas (vt joonisel 4.1 paremal):

- **migratsiooni vigade raportina** (T1)**,** kus on välja toodud kõik päringu vigased read;
- analüüsitava tunnuse (konto tüüp) **võrdleva sagedustabelina** (T2).

Soovitus 4.2.1. Vastavuse testimisel eelistada **väärtuste võrdlemisi**, sest ainult summaarsel võrdlemisel jäävad tähelepanuta olukorrad, kus lähte- ja sihttabelite vigade arvud on võrdsed. (nt. lähtebaasi on 3 üle kandmata kirjet ja sihtbaasi on tekkinud 3 sobimatut kirjet).

Soovitus 4.2.2. Negatiivse testi tulemuse analüüsimiseks lisada testile abipäring, mis annab väärtuste jaotustabeli (vt joonisel 4.1 näide T2).

Soovitus 4.2.3. Negatiivse testi tulemuse analüüsimiseks lisada testile abipäring, mis annab väärtuste erinevuste loendi (vt joonisel 4.1 näide T1).

4.3 Objektiga seotud tunnuste kaudne kontrollimine

Kui andmestruktuurid on siht- ja lähtebaasis väga erinevad, siis võib testimise takistuseks olla sobivate võrreldavate tunnuste puudumine. Sellisel juhul soovitab Haller (Haller, 2009, lk 76–77) konstrueerida asjakohastest tunnustest **tehniline agregeeritud** tunnus (FINGERPRINT). Näiteks joonisel 4.2 on agregeeritud tunnuseks pangakontoga seotud kaartide intresside summa, mis äriloogiliselt on mõttetu, kuid aitab leida vead, kus pangakontoga on mõni kaart sidumata.

```
SELECT o.ACCOUNT_ID, n.ACCOUNT_ID,
CASE WHEN o.ACCOUNT_ID is not null AND n.ACCOUNT_ID is not null THEN 'OK'
     ELSE 'FAILED'
END as match,
CASE WHEN o.FINGERPRINT= n. FINGERPRINT THEN 'OK'
     ELSE 'ERROR'
END as equal
FROM            (SELECT ACCOUNT_ID, SUM(RATE) as fingerprint
                      FROM T_INTR_OLD GROUP BY ACCOUNT_ID) o
FULL OUTER JOIN (SELECT ACCOUNT_ID, SUM(RATE) as fingerprint
                      FROM T_INTR_NEW GROUP BY ACCOUNT_ID) n
ON n.account_id=o.account_id
```

Võrdleva päringu tulemus

Old		New		Match	Equal
Key	Fingerprint_1	Key	Fingerprint_1		
1000405201	10.25%	1000405201	10.25%	✓	✓
1000765208	3.00%			×	
1000225055	6.60%	1000225055	6.50%	✓	×
3000324419	0.50%	3000324419	0.50%	✓	✓

Joonis 4.2 Agregeeritud kontrolltunnusega testi päring ja päringu tulemus. Allikas: (Haller, 2009, lk 76–77) jooniste alusel.

Soovitus 4.3.1. Migratsiooni testimise eesmärk on veenduda, et andmed on üle kantud korrektselt. Seega võib andmeid vaadata **äriloogikast sõltumatuna** ja testide koostamisel lähtuda eelkõige andmete kontrollimise vajadusest.

Soovitus 4.3.2. Seoste testimisel kasutada vajadusel agregeerimist:näiteks loendamist (COUNT), summeerimist (SUM).

4.4 Objektiga seotud tunnuste kontrollimine räside abil

Wei (Wei & Chen, 2014, lk 5–7) soovitab kasutada kirjete vastavuse testimiseks mitmetest tunnustest koostatud kontrolltunnust.

Laadimise ajal **tehniliste probleemide tõttu** või **andmetüüpide konfliktist** (nt väiksem komakohtade arv, sihttunnuse lühem pikkus) tingituna võib esineda vigu, mis võivad võrdlustestidega tähelepanuta jääda.

Selliste vigade leidmiseks soovitab Wei lisada lähtetabelisse kontrolltunnuse, mis genereeritakse mitme kontrollitava tunnuse väärtuste põhjal **räsifunktsiooni**[14] abil. Pärast andmete migreerimist koostatakse sihttabeli samadest tunnustest sama räsifunktsiooniga kontrolltunnus. Testimisel võrreldakse lähte- ja sihttabeli kontrolltunnuste räsisid (vt joonis 4.3).

Employees source table to be migrated using the checksum verification method

ID	Name	Address	...	Checksum Hash Code
4	Mark	1324 Galaxy Lane	...	a88798f5a9025645020026f11c35c93f
6	Skip	1326 Humuhumunukunukuapuaa Avenue	...	17c126a5e3c06f20a8f36a3f1703778c

Employees target table received with error

ID	Name	Address	...	Checksum Hash Code Received
4	Mark	1324 Galaxy Lane	...	a88798f5a9025645020026f11c35c93f
6	Skip	1326 Humuhumunukunuk	...	3f2ce06d09d7bf5835041c1eece0a0b3

Comparison of received and calculated hash codes

ID	Checksum Hash Code Received		Hash Code Calculated After Receipt
4	a88798f5a9025645020026f11c35c93f	=	a88798f5a9025645020026f11c35c93f
6	17c126a5e3c06f20a8f36a3f1703778c	! =	3f2ce06d09d7bf5835041c1eece0a0b3

Joonis 4.3 Kontrollräsi alusel vigaste kirjete leidmise näide. Migreerimise käigus moondunud väärtusega kirjete kontroll-räsid on lähte ja sihttabelis erinevad. Allikas: (Wei & Chen, 2014, lk 5–6) jooniste alusel.

Tabeli kõikide tunnuste ükshaaval testimine on ajamahukas, testimise aega hoiab kokku kirjete kaupa testitavate tunnuste räside võrdlemine. Igast tunnusest räsifunktsiooniga moodustatud fikseeritud pikkusega räside liitmisel saadakse kontrolltunnus. Näide mitme tunnuse põhjal räsi moodustamisest: `md5(nimi::text) || md5(tel_nr::text) || md5(kood::text) as hash`

Sellise liiträsi moodustamisel saab korraga testida väga erinevate andmetüüpidega (*date, numeric, text* jt) väärtusi ning tagatakse ka iga tunnuse eraldi kontrollimise korrektsus. Näiteks lihtne kahe teksti liitmine võib anda erijuhtudel sama tulemuse, kuid räside liitmine annab täiesti erineva tulemuse:

ANN + EELMAA ANNEELMAA
`76233b2878b3da472c66c8075ca74930b555d0de547865d384b7f4b75391e501`

[14] **Räsifunktsioon** (*hash function*) on algoritm, mis teisendab sisendi fikseeritud pikkusega bitijadaks. vt ka (Veldre, 2015)

Soovitus 4.4.1. Testida kirjete kaupa kasutades testitavatest tunnustest genereeritud räside võrdlemist. Kui kirjete räsides leidub erinevusi, siis kontrollida igat tunnust eraldi.

4.5 Vastavuse testimise automatiseerimine

Homayouni tutvustab andmelaonduses laadimise kontrollimisel vastavuse testimise (*balancing tests*) täieliku automatiseerimise ideed (Homayouni, 2018, lk 57–65).

Andmelaonduses kasutatakse laadimisprotsesside kirjeldamiseks vastavustabeleid (*mapping table*) (vt. joonis 4.4), kus on kirjeldatud tunnuste kaupa lähtebaasist andmete sihtbaasi kandmise tingimused.

Vastavus	Lähtetabel	Tingimus	Lähtetabeli tunnus	Sihttabel	Sihttabeli tunnus
1:1 table	Address	Year>200		Location	
1:1 attribute	Address	Year>200	Address_key	Location	Location_id
n:1 table	[0]:Patient [1]:Concept Operation:LEFT JOIN	IsCurrent=1		Persons	

Joonis 4.4 Andmete laadimise vastavustabeli näide.

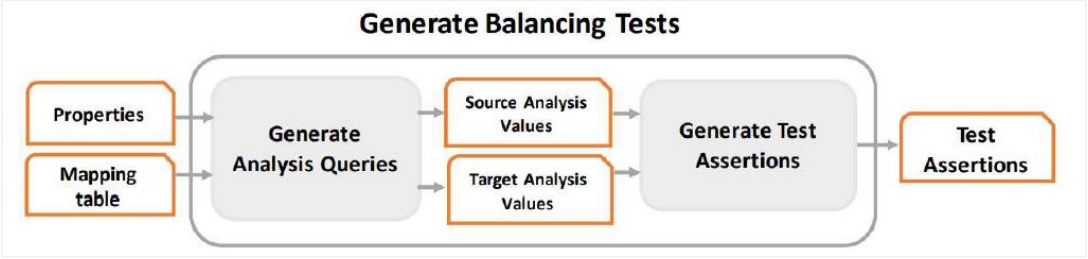
Homayouni on analüüsinud erinevaid testimise liike ning seoste tüüpe (1:1, 1:n) ning kirjeldanud vastavate testide genereerimise algoritmid (vt. joonis 4.5).

Algorithm 1 Analysis Query Generator Function

```
1: function ANALYSISQUERYGENERATOR(mapping, property)
2:     switch property do
3:         case RecordCountMatch
4:             if mapping is one-to-one then
5:                 target_analysis_query = "SELECT Record_count(mapping.target_table)"
6:                 source_analysis_query= "SELECT Record_count(mapping.source_table)
7:                 WHERE mapping.selection_condition = TRUE"
8:         case DistinctRecordCountMatch
9:             if mapping is many-to-one and mapping.table_operation="LEFT JOIN" then
10:                target_analysis_query = "SELECT Distinct_record_count(mapping.target_table)"
11:                source_analysis_query= "SELECT Record_count(mapping.source_table[0])
12:                WHERE mapping.selection_condition[0] = TRUE"
13:        case AttributeValueMatch
14:            target_analysis_query = "SELECT ALL(mapping.target_attribute)"
15:            if mapping is one-to-one then
16:                source_analysis_query= "SELECT ALL(mapping.source_attribute)"
```

Joonis 4.5. Testide päringute generaatori näide. Allikas (Homayouni, 2018, lk 62).

Homayouni soovitab vastavustabelid vormistada masinloetavalt, et testimise generaator (vt. joonis 4.6) suudaks vastavustabeli andmete põhjal genereerida testimiseks vajalikud päringud, need käivitada ning tagastada testi tulemuse. Seega testimise sisendiks on vastavustabel (*Mapping Table*) koos viitega testi testi tüübile, selle põhjal genereeritakse lähte- ja sihtandmete päringud (SQL laused), leitakse päringu tulemused (*Source Analysis Value*, *Target Analysis Value*) ning arvestades korrektsuse tingimusi leitakse testi tulemus (*Test Assertations*).



Joonis 4.6. Vastavuse testimise generaator. Allikas (Homayouni, 2018, lk 61).

Sellise lähenemisviisiga jääb ära ajamahukas testide kirjutamine ning väheneb oluliselt testimisele kuluv aeg.

Kokkuvõtteks see lähenemisviis baseerub testide tüüpide määramisel ning tüüpidele sobivalt masinloetavalt vormistatud vastavustabelite olemasolul. Kuid sarnast struktureeritud mõtteviisi võib kasutada ka testide käsitsi kirjutamisel. Järgnevates soovitustes ja peatükkides lähtutakse selle kogemuse kasutamisel kahest perspektiivist:

- teha eeltööd testide genereerimise ja haldamise lahenduse jaoks;
- luua tüüptestide päringute mallid[15], mida saavad eeskujuna kasutada migratsiooni projektide testijad.

Soovitus 4.5.1. Koostada sagedamini esinevate testide päringute põhjal erinevate tüüptestide päringute mallid (näidistestid).

Soovitus 4.5.2. Vaadata teste päringute struktuuri sarnasuse kaupa. Testide mallide (näidistestide) kasutamise abil muuta testide koostamine efektiivsemaks.

Soovitus 4.5.3 (Arendusidee) Testi tüüpide ja testide päringute mallide olemasolu korral saab koostada programmi testide genereerimiseks.

Soovitus 4.5.4 (Arendusidee) Kui kõigi testide kohta on hallataval kujul (nt andmebaasis) olemas testide päringud, siis saab testimist saab automatiseerida: süsteem käivitab testid, salvestab tulemuse, koostab testimise raporti.

4.6 Ühiktestide lähenemisviisi kasutamine

Andmete migratsiooni testimisel on vaja teostada palju sõltumatuid kontrolle, kontrollitakse väga paljude üksikute tunnuste väärtuseid. Tarkvaraarenduses kasutatakse konkreetsete koodi osade (tavaliselt funktsioonide) toimivuse pidevaks automaatseks kontrollimiseks ühikteste (*unit test*). Ühiktestide koostamisel lähtutakse **printsiibist FIRST** (*Fast - Isolated - Repeatable - Self-validating - Timely*), mille järgi testid peavad vastama järgmistele omadustele (Gupta, i.a).

- Testid peavad olema **kiired** (*Fast*), sest testide hulk on suur ja kõigi testide käivitamisel kuluv summeeritud aeg võib muutuda väga suureks.
- Testid peavad olema **isoleeritud** (*Isolated*), see tähendab, et iga test on omaette üksus ja teste saab käivitada ükshaaval sõltumata teiste testide tulemustest. Ühe testiga testida ainult ühte teemat, siis on vea põhjuste leidmine kiirem.
- Testid peavad olema **korratavad** (*Repeatable*), see tähendab et test peab andma igal käivitamisel sama tulemuse.
- Testid peavad olema **isevalideeruvad** (*Self-validating*). Testi tulemus peab selgelt väljendama, kas tulemus on ootuspärane või mitte.

[15] mall määrab ära päringu struktuuri (vt ptk 5.2)

- Testid peavad olema **õigeaegselt** koostatud (*Timely*). Kui testid koostada varakult, siis testid on arendamisel abivahendiks korrektse skripti(koodi) kirjutamisel.

Soovitus 4.6.1. Kasutada migratsiooni järgsel testimisel ühiktestidega sarnaselt **isevalideeruvaid teste,** mis tagastavad tulemuse tõeväärtuse: korras (OK) või vigane (NOK).

Soovitus 4.6.2. Arvestada testide **korratavuse nõudega**, kui migreerimise skriptides on kasutatud võrdlust hetke kuupäevaga. Sellisel juhul peab testides võrdlemisel kasutama migratsiooni kuupäevaga. Näiteks kui isikute migreerimisel täidetakse tunnus ON_ALAEALINE, siis 10 päeva hiljem selle tunnuse testimisel käesoleva hetke kuupäevaga kajastatakse vigadena viimase 10 päeva jooksul täisealiseks saanud isikuid.

4.7 Ühe migratsiooniprojekti testimise kogemus

Enne magistritöö kirjutamist osales autor ühes migratsiooni projektis, kus realiseeriti PL/SQL protseduuride abil paljude testide korraga käivitamine, mis tagastas tulemuseks testide tulemuste tõeväärtusena (korras/vigane). Vigade analüüsiks olid ette valmistatud abipäringud.

Soovitus 4.7.1 Testimise korraldamisel on soovitav testide päringute käivitamine koondada **üheks/paariks protseduuriks**, mis salvestab testimise **tulemuse tabelisse.** Niisuguse töökorralduse korral saavad teistes keskkondades teste käivitada administraatorid ja testimise tulemuse saab säilitada ning edastada testijale/analüütikule.

Soovitus 4.7.2 Koostada mõned testid, mis võrdlevad **migreeritud andmete mahtusid oodatavate mahtudega**. Selliste testidega saab tuvastada olukorda, kus valede või puuduvate algandmete tõttu on sihtbaasis liiga vähe või liiga palju kirjeid. Vastavustestidega saab avastada erinevusi lähte ja -sihtbaasi vahel, kuid ei tuvastata lähteandmete puudumist.

Selles migratsiooniprojektis koostati ka üksikobjekti (nt isiku) põhjalikumaks analüüsimiseks päringute komplektid. Loodi sql.fail, kus faili alguses oli objekti muutuja (nt isiku_id) väärtustamine ning edasi järgnesid muutujat kasutavad objektiga seotud päringud, mis tagastasid lähte- ja sihtbaasist väljavõtteid selle objekti ja tema seoste kohta. Sellised päringud olid abivahendiks teiste testide vigade põhjalikumal analüüsimisel ning infosüsteemi testimisel leitud vigade põhjuste otsimisel.

Soovitus 4.7.3 Koostada süvaanalüüsi lihtsustamiseks abipäringute komplektid, kus on koondatud **ühe objektiga** seotud andmete ja seoste päringud mõlemast andmebaasist (lähtebaasist ja sihtbaasist).

Järgnevates peatükkides lähtutakse selles peatükis välja toodud soovitustest ning kirjeldatakse testide tüüpidest lähtuvat migratsiooni testimise lähenemisviisi.

5 Migratsiooni testide päringud

Migratsiooni testimisel on põhirõhk andmete korrektsuse testimisel ja põhiline testimise meetod on **päringute abil** andmebaasi sisu kontrollimine (joonisel 3.3 tööd 1-A, 2-A, 2-B, 3-1). Iga testiga kontrollitakse ühe nõude (väärtuste vastavuse nõude, kvaliteedi nõude jt) täidetust.

Selles töös kasutatakse päringute abil testimisel järgmist lähenemisviisi:

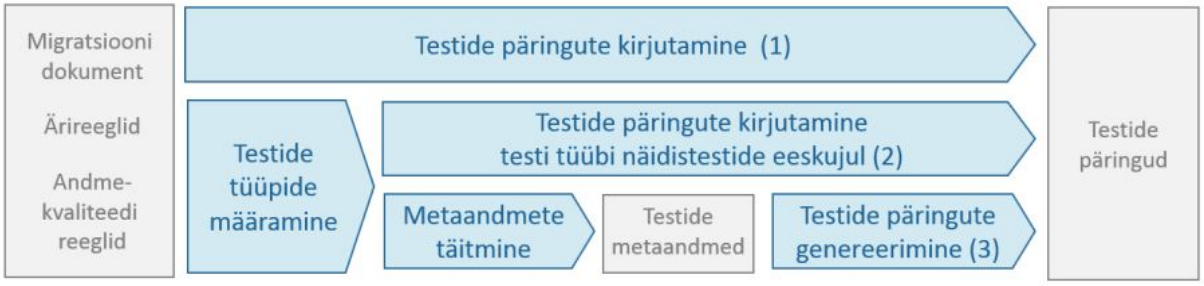
- Nõude kontrollimiseks koostatakse päring (edaspidi **testi päring**), mis loendab nõudele mittevastavusi (edaspidi vigu).
- **Testi tulemus** loetakse **positiivseks (OK)**, kui päring tagastab väärtuse 0 (vigade arvu) või null rida (vigaste kirjete päring) või st nõuetele mittevastavuste arv on 0. Muudel juhtudel testi tulemus on **vigane (NOK)**.

5.1 Testide päringute koostamine

Testide koostamise aluseks on **migratsiooni dokumendid** (sh vastavustabelid), uue keskkonna **ärireeglid** ja mõlema keskkonna **andmekvaliteedi reeglid**.

Kuna testimisel tuleb veenduda sihtbaasi kõikide tabelite/veergude korrektses täidetuses ja lähtebaasi andmete ülekandmise korrektsuses, siis testimiseks vajalike päringute hulk ja nende koostamisele kuluv aeg on väga suur.

Testide ettevalmistamisel (vt joonis 5.1) annab aja kokkuhoidu päringute käsitsi kirjutamise (1) asemel **päringute genereerimine** (3). Nii päringute genereerimine, kui ka näidistestide eeskujul testide kirjutamine (2) eeldab sarnasuse alusel **testide grupeerimist testide tüüpideks**. Lähenemisviis tuleneb soovitustest 4.5.1-4.5.3.



Joonis 5.1 Testide päringute koostamise viisid: käsitsi kirjutades(1), näidiste eeskuju kasutades (2) või süsteemi poolt teste genereerides(3).

Selles töös tehakse ära eeltöö testide päringute genereerimise programmidele (soovitus 4.5.3) defineerides migratsiooni projektile sobivad **testide tüübid** ning koostades iga testi tüübi kohta päringute mallid. Selliselt ettevalmistatud testide mallid lihtsustavad ka käsitsi testide kirjutamist (2).

Alati leidub ka keerulisi päringuid, mis ühegi tüübi alla ei kuulu. Selliste keerukamate testide korral jääb alles testide päringute käsitsi kirjutamise vajadus. Autori kogemusel peaks saama tüüptestidega ära katta 70%-90% migratsiooni testimise vajadusest[16].

[16] Antud katvuse hinnang baseerub autori migratsiooniprojekti kogemusel.

5.2 Testide päringute analüüsimine ja testide tüübid

Testide päringuid saab genereerida, kui päringute sarnasuse alusel on defineeritud testide tüübid. Selle peatüki eesmärk selgitada **testide tüüpide** lähenemisviisini jõudmist (ptk 5.2.1, 5.2.2), testide tüüpide defineerimist (ptk 5.2.3) ning selgitada testide mallide ja metaandmete olemust(5.2.4).

5.2.1 Testide päringute analüüs

Migratsiooni testimisel tuleb teha väga palju päringuid, kuid paljud neist päringutest on sarnase struktuuriga, erinevus on tihti ainult tabelite/tunnuste nimedes. Sarnase struktuuriga päringuid saab vaadelda ühe **testi tüübina**.

Vaatame lähemalt ühte päringut ja selle päringu üldistamise võimalusi. Joonisel 5.2 on isikukoodi unikaalsuse kontrollimise päring. Vasakpoolsel joonisel on testi päring, mis tagastab mitte-unikaalsete isikukoodide arvu ja testi tulemus loetakse korrektseks, kui päringu tulemus on 0.

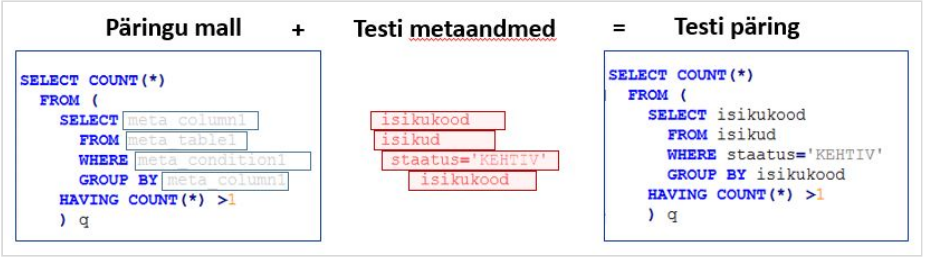
```
1  -- Tabelis ISIKUD kehtivate kirjete hulgas
2  -- isikukoodi unikaalsuse kontroll
3
4  SELECT COUNT(*)
5    FROM (
6      SELECT isikukood
7        FROM isikud
8        WHERE staatus='KEHTIV'
9        GROUP BY isikukood
10     HAVING COUNT(*) >1
11     ) q
```

```
1  -- Mall: Unikaalsuse kontroll
2  -- Metaandmed: meta_column1, table1, meta_condition1
3
4  SELECT COUNT(*)
5    FROM (
6      SELECT meta_column1
7        FROM meta_table1
8        WHERE meta_condition1
9        GROUP BY meta_column1
10     HAVING COUNT(*) >1
11     ) q
```

Joonis 5.2 Väärtuse unikaalsuse kontrolli päringud. **Vasakpoolsel joonisel on isikukoodi unikaalsuse testimise päring**, mis tagastab mitteunikaalsete isikukoodide arvu. Päring tagastab alampäringu (read 6-8) kirjete arvu (rida 4). Alampäringuga leitakse isikute tabelist (rida 7) isikukoodid (read 6 ja 9), mida leidub antud tingimustel (rida 8) rohkem, kui üks kord (rida 10). **Parempoolsel joonisel on sama päringu üldistus**, kus konkreetsete andmete asemel on viide metatunnustele.

Kui vasakpoolsel joonisel asendada ISIKUKOOD mõne teise tunnusega (näiteks ISIK_ID-ga), siis saaksime sama struktuuriga testi päringu teise tunnuse (ISIK_ID) unikaalsuse kontrollimiseks. Antud näite alusel saab defineerida väärtuse **unikaalsuse kontrollimise malli** (vt. joonisel 5.2 parempoolne päring), kus konkreetsete vääruste asemel on viited testi üldistatud tunnustele (edaspidi **metatunnustele**). Selle teadmise põhjal defineeritakse testi tüüp 'UNIQUE_VALUE', mis vajab päringu genereerimiseks tunnuse nime (META_COLUMN1), tabeli nime (META_TABLE1) ja filtritingimust (META_CONDITION1).

Kõik testi tüüpide kirjeldused sisaldavad **päringu malli**, mis annab ette SQL lause struktuuri ja **testi metaandmete** (metatunnuste komplekti) **vajadust**. Testi päring saadakse päringu malli ja metaandmete kokku panemisel (vt joonis 5.3).



Joonis 5.3 Päringu malli ja metaandmete kasutamine testi päringu koostamisel.

Sarnase lähenemisviisiga saab defineerida võrdlevaid teste kahelt tabelilt (vt joonis 5.3).

```
7  SELECT COUNT(*)
8  FROM
9    (SELECT KOOD, NIMI
10       FROM ISIKUD WHERE sugu='N') t1
11   FULL JOIN
12   (SELECT ID, NAME
13       FROM PERSONS WHERE gender=1) t2
14   ON t1.KOOD =  t2.ID
15 WHERE t1.NIMI!= t2.NAME
16   OR  t1.NIMI IS NULL
17   OR t2.NAME IS NULL
```

```
7  SELECT COUNT(*)
8  FROM
9    (SELECT meta_key1, meta_column1
10       FROM meta_table1 WHERE meta_condition1 ) t1
11   FULL JOIN
12   (SELECT meta_key2, meta_column2
13       FROM meta_table2 WHERE meta_condition2 ) t2
14   ON t1.meta_key1 =  t2.meta_key2
15 WHERE t1.meta_column1!= t2.meta_column2
16   OR  t1.meta_column1 IS NULL
17   OR t2.meta_column2 IS NULL
```

Joonis 5.4 Vasakul on kahes tabelis isiku nime võrdlemise testi päring, paremal on sama tüüpi testi mall koos metatunnuste vajadusega.

Joonisel 5.4 oleva testiga kontrollitakse, kas naiste nimed on üle kantud korrektselt. Isikute sidumine toimub seose ISIKUD.KOOD = PERSONS.ID abil (rida 14). Test loendab vigaseid kirjeid, kus nimed ei ole ühesugused (rida 15) või ühes tabelist on väärtused puudu (read 16, 17).

Need testide päringud annavad vigade arvu, järgnevas selgitatakse vigaste kirjete saamise päringuid.

5.2.2 Vigaste testide tulemuste analüüsimiseks abipäringud

Kui test on osutunud vigaseks, siis kasutaja peab vea põhjuste tuvastamiseks tegema täiendavaid päringuid. Sama päringu või selle alampäringu käivitamine ei anna kasutajale piisavalt infot vea võimalike põhjuste kohta. Kui testi päringut veidi muuta, on võimalik genereerida kasutajale ka täiendavaks vigade otsimiseks **esmane abipäring**.

Näitena vaatame olukorda, kus eelpool vaadeldud unikaalsuse testi (vt joonis 5.2) tulemusena saab kasutaja teada, et isikukood ei ole unikaalne. Kui kasutaja käivitab sama päringu alampäringu (joonis 5.2 read 6-10), siis saab ta isikukoodide loendi. Kuid vea tuvastamiseks vajab ta nende isikute kohta rohkem andmeid, sest ainult isikukoodi nägemine ei pruugi anda piisavalt infot vea tekke põhjuste kohta. Kasutaja vajab testi päringuga sarnast päringut (vt. joonis 5.5), kus lisaks isikukoodile on näha ka muude tunnuste väärtusi (rida 18), kuid päringu andmete leidmise osa (read 19-23) jääb samaks. Abipäringute genereerimiseks on vaja lisaks varem defineeritud metatunnustele ainult täiendavate tunnuste loetelu META_EXAMINE_COLUMNS.

```
14 -- Isikute tabeli unikaalsus kontrolli
15 -- vigaste kirjete päring
16
17
18 SELECT t1.isikukood, t1.nimi, t1.synniaeg
19 FROM isikud t1,
20 (SELECT isikukood FROM isikud
21         WHERE staatus='KEHTIV'
22         GROUP BY isikukood  HAVING COUNT(*) >1 ) q
23 WHERE t1.isikukood = q.isikukood
24 order by isikukood
```

```
14 -- Unikaalsuse kontrolli abitabel
15 -- Metaandmed: meta_examine_columns, meta_column1,
16 --             meta_table1, meta_condition1
17
18 SELECT meta_examine_columns
19 FROM meta_table1 t1,
20 (SELECT meta_column1 FROM meta_table1
21         WHERE meta_condition1
22         GROUP BY meta_column1  HAVING COUNT(*) >1 ) q
23 WHERE t1.column1 = q.column1
24 order by meta_column1
```

Joonis 5.5 Vasakul on unikaalsuse testi päring konkreetsete andmetega, paremal on unikaalsuse testi mall koos metatunnuste vajadusega.

Päringute sarnasuse tõttu on soovitav päring ja abipäringud koostada samaaegselt. Hiljem vigade põhjuste analüüsimise käigus päringu struktuuri läbimõtlemine võtab rohkem aega.

Järgnevas vaadeldakse testi päringute komplektina: isevalideeruv test ja temaga seotud abipäringud.

5.2.3 Testide tüübid

Eelpool kirjeldatud lähenemisviisi kasutades defineeriti testide tüübid järgmiste tegevustega.

1. Migratsiooni testimise vajaduse (vt ptk 3) ning andmete testimise kogemuste (vt ptk 4 soovituste) põhjal koostati esmane testide tüüpide loetelu.
2. Iga testi tüübi jaoks koostati eelnevalt kirjeldatud viisil (vt. ptk. 5.2.1, 5.2.2) päringute mallid ning tuvastati selle testi tüübi jaoks vajalikud metaandmed.
3. Võrreldi kõikide testide tüüpide metaandmeid ning muudeti metaandmete käsitlust paindlikumaks ning meta-tunnuste loetelu lühemaks. (vt. ptk. 5.2.4 ja lisa I).
4. Koostati testi tüüpide kirjeldused (vt lisa II), kus on iga testi tüübi jaoks on kirjeldati **metaandmete vajadus**, **päringute mallid** ja täpsustati metaandmete kasutamist (sh kohustuslikkust).
5. Tehtud tööd valideeriti ühe migratsiooniprojekti testide peal ning korrigeeriti koostatud testide tüüpide kirjeldusi.
6. Iga testi tüübi jaoks koostati vigase testi tulemuste näidised (vt lisa II).

Selle töö tulemusena defineeritud 16 testi tüüpi, mille loend koos näite ja kasutusalaga on toodud tabelis 5.1

Tabel 5.1 Testide tüübid

Nr	Testi tüüp	Testi kirjeldus	Näide	Kasutusala
1	**NOT_NULL**	Kohustuslikkuse kontroll. Tunnus peab olema täidetud. Tunnuse sisu ei kontrollita.	ISIKUKOOD on alati täidetud.	Lähtetabeli testimine Sihttabeli testimine
2	**IS_NULL**	Tunnuse väärtused peavad olema täitmata.	KEHTIV_KUNI on alati täitmata.	Lähtetabeli testimine Sihttabeli testimine
3	**STATIC_VALUE**	Tunnus on täidetud etteantud väärtusega.	STAATUS on alati 'KEHTIV'.	Lähtetabeli testimine Sihttabeli testimine
4	**IN_VALUES**	Tunnuse väärtus sisaldub lubatud väärtuste loetelus.	SUGU sisaldab ainult lubatud väärtusi 'NAINE', 'MEES'.	Lähtetabeli testimine Sihttabeli testimine
5	**UNIQUE_VALUE**	Tunnuse väärtus on etteantud tingimuste korral on unikaalne.	ISIKUKOOD on unikaalne.	Lähtetabeli testimine Sihttabeli testimine
6	**CODE**	Tunnuse väärtused (koodid) on esindatud teises(nn klassifikaatorite) tabelis.	Tunnuse EMAKEEL väärtus sisaldub tabelis KEELED.	Lähtetabeli testimine Sihttabeli testimine
7	**VALUE_BETWEEN**	Tunnuse väärtused peavad jääma etteantud vahemikku. Erijuhtumid: 'suurem kui' ja 'väiksem kui'	Isiku VANUS on vahemikus 0-120	Lähtetabeli testimine Sihttabeli testimine
8	**METRIC_COUNT**	Kirjete arv peab jääma etteantud vahemikku. Erijuhtumid: 'suurem kui' ja 'väiksem kui'. (Soovitus 4.7.2)	Töötajate tabeli oodatav kirjete arv on vahemikus 100-120.	Mõõdikud sihttabeli testimisel

Nr	Testi tüüp	Testi kirjeldus	Näide	Kasutusala
9	**EQUAL_ROWS**	Kahe tabeli kirjete arv on võrdne.	Tabelites ISIKUD ja PERSONS on kirjete arv võrdne.	Vastavuse testimine
10	**MATCH_KEY**	Kahe tabeli tunnuste väärtuse olemasolu võrdlus. Samad tunnused on ka seose loomise aluseks.	Veergudes ISIK.KOOD ja PERSON.ID on samad väärtused, st kõik isikud on üle kantud ja sihttabelis ei ole ühtegi isikut üle.	Vastavuse testimine
11	**MATCH_DISTINCT_KEY**	Kahest tabelist tunnuse unikaalsete väärtuste arvu võrdlus.	Unikaalsete isikute arv tabelis DOKUMENDID võrdub tabeli ISIKUD isikute arvuga. Kui andmemudeli muudatuse tõttu on sihttabelis ühe kirje asemel mitu kirjet.	Sihttabeli testimine Vastavuse testimine
12	**MATCH_HASH**	Kahe tabeli mitme tunnuse põhjal genereeritud räside võrdlus. (Soovitus 4.4.1)	Isikuga seotud tunnused on korrektselt üle kantud. Võrreldakse mitme tunnuse põhjal genereeritud räsisid. r(NIMI)+r(SUGU)+r(TELEFON) *vs.* r(NAME)+r(GENDER)+r(PHONE). Seose tingimus on ISIK.KOOD=PERSON.ID.	Vastavuse testimine
13	**MATCH_COLUMN**	Kahe tabeli tunnuste väärtuste võrdlus. Seos luuakse teiste tunnuste abil (Soovitus 4.2.1).	Isikuga seotud tunnus on korrektselt üle kantud. Võrreldakse tunnuseid ISIK.NIMI ja PERSON.NAME. Seose tingimus on ISIK.KOOD=PERSON.ID.	Vastavuse testimine
14	**MATCH_COUNT**	Kahe tabelis grupeeritud kirjete arvu võrdlus (Soovitus 4.3.2).	Kahes tabelis isikuga seotud laste kirjeid on sama palju.	Vastavuse testimine
15	**MATCH_SUM**	Kahe tabelis grupeeritud alamsummade võrdlus (Soovitused 4.3.1-4.3.2.).	Kahes tabelis on arvetega seotud sama palju arvete ridu.	Vastavuse testimine
16	**INFORMAL**	Kasutaja poolt vabalt loodud test, kus päringuid ei genereerita, vaid testide päringud on kasutaja poolt sisestatud.	Keerukas test, kus lähte- või sihtbaasis on mitmeid tabeleid hõlmav päring.	Lähtetabeli testimine Sihttabeli testimine Vastavuse testimine

Kogu testimisvajaduse katvuse mõttes kirjeldati ka testi tüüp INFORMAL (tabelis 5.1 rida 16), mis on mõeldud teistesse testi tüüpidesse mitte sobivatele testidele. Kuna tegemist on käsitsi kirjutatud testide tüübiga, siis on see tüüp vajalik ainult testimise automatiseerimisel.

5.2.4 Testide metaandmed

Eelnevalt kirjeldatud päringute analüüsil (ptk. 5.2.1, 5.2.2) leiti, et tabelis 5.1 toodud kõigi testide vajaduste haldamiseks on vaja kümmet metatunnust. Mõned näited metaandmete koondamise analüüsil tehtud otsustest:

- Meta-tunnuste nimetamisel loobuti siht-ja lähtetabeli mõistest ning võeti kasutusele nummerdatud tabelid (table1, table2). Selline sõnakasutus võimaldab defineerida ka ühe baasi kahe tabeli vahelisi seoste kontrolle. Näiteks klassifikaatori väärtuse kontrolli (testi tüüp CODE).
- Mitme päringuga testide korral kasutatakse päringute kokkupanemisel tabelitel lühinimesid (*alias*) t1 ja t2. Neid lühinimesid tuleb kasutada ka vastavustestides abipäringute tunnuste loeteludes (META_EXAMINE_COLUMNS).
- Kahe tabeli vastavuse testimisel kasutatakse metatunnuseid META_KEY1, META_KEY2 tabelite vaheliste seose loomiseks (t1.META_KEY1= t2.META_KEY2).
- Väärtuste meta-tunnused META_KEY1, META_KEY2 kasutatakse väärtuse sisu kontrollimiseks (testi tüübid STATIC_VALUE, IN_VALUES) või väärtuse vahemikku kuulumise kontrollimiseks (testi tüübid VALUES_BETWEEN, METRIC_COUNT).
- Kui testide haldus automatiseerida, siis on vaja ka testi tüüpi käsitsi kirjutatud päringute jaoks (testi tüüp INFORMAL). Selle testi tüübi kasutajasõbraliku veateade malli saab panna metatunnusesse META_EXAMINE_COLUMNS. Testide päringute kirjutamiseks selle tüübi juures metaandmeid vaja ei ole.

Metatunnuste täitmine ja kasutus sõltub otseselt testi tüübist (vt lisa II, lisa III), kuid enamasti on metatatunnuse tähendus järgmine:

- Ühe päringu jaoks vajatakse tavaliselt kolme metatunnust: META_TABLE1, META_COLUMN1, META_CONDITION1.
- Kui päringuid on kaks, siis teise tabeli jaoks vajatakse veel kuni kolme metatunnust: META_TABLE2, META_COLUMN2, META_CONDITION2.
- Kui päringus kasutatakse väärtusega võrdlemist, siis on vaja päringule ette anda ka oodatav väärtus või väärtuste loend: META_KEY1, META_KEY2.
- Abipäringute jaoks vajatakse tagastatavate tunnuste loetelu jaoks tunnuseid: META_EXAMINE_COLUMNS1, META_EXAMINE_COLUMNS2.
- Kui filtritingimused META_CONDITION1, META_CONDITION2 on täitmata, siis tehakse päringud kogu tabeli pealt.

Kokkuvõte metatunnuste kasutusest küsimuste tüüpide kaupa on lisas III.

Metaandmete ja testide mallide struktuuri põhjal koostatakse tekstide liitmise meetodil testide päringud. (vt. näidet joonisel 5.6).

```
_query1 :=  'SELECT COUNT(*) FROM ( SELECT '|| _column1 ||' FROM '|| _table1 ||' t1 '||
   _condition1 ||' GROUP BY '|| _column1 ||' HAVING COUNT(*) >1 ) q ';

_query_examine1 := 'SELECT '||_examine_columns||' FROM '|| _table1 ||' t1, ( SELECT '||
   _column1 ||' FROM '|| _table1 ||' t1 '|| _condition1 ||' GROUP BY '|| _column1 ||'
   HAVING COUNT(*) >1 ) q WHERE t1.'|| _column1 ||' = q.'|| _column1 ;
```

Joonis 5.6 Unikaalsuse testi päringute koostamine PL/SQL koodis.

Tekstide liitmisel pannakse võtmesõnade vahele metatunnuste väärtused, mille oskusliku täitmisega saab kirjeldada ka keerukamaid teste. Näiteks võib tabeli (META_TABLE1) väärtuses võib kasutada keerukamat alampäringut või ühe tunnuse META_COLUMN1 asemele panna tunnuste loetelu ja kasutada tunnuste lühinimesid (nt META_COLUMN1:= 't1.eesnimi, t1.nimi', t2. nimi as uus_nimi).

5.2.5 Testide tüüpide kasutamine

Kõigi koostatud testi tüüpide **metaandmete** ja **päringute genereerimise mallide** kirjeldused on lisas II. Tabel 5.2 on näidisena esitatud ühe testitüübi kirjeldus, kus

- QUERY1, QUERY2 on testi päringute mallid,
- RESULT_COMMENT on mittekorrektse tulemuse (NOK) veateate mall,
- QUERY_EXAMINE1, QUERY_EXAMINE2 on abipäringute mallid.

Tabel 5.2 Unikaalsuse kontrollimise testi tüübi UNIQUE_VALUE kirjelduse näide

METAANDMED			
meta_table1	**Tabel**	meta_table2	-
meta_column1	**Tunnus**	meta_column2	-
meta_condition1	Filtritingimus	meta_condition2	-
meta_key1		meta_key2	-
meta_examine_columns1	Tunnuste loetelu	meta_examine_columns2	-

PÄRINGUTE JA TULEMUSTE GENEREERIMISE MALLID	
query1	SELECT COUNT(*) FROM (SELECT {meta_column1} FROM {meta_table1} WHERE {meta_condition1} GROUP BY {meta_column1} HAVING COUNT(*) >1) q
query2	-
	Testitud andmed on korrektsed, kui esimese päringu tulemus on 0
result_comment	NOK: Leidub {query1_result} väärtust, mida esineb tunnuses {meta_column1} rohkem kui 1 kord.
query_examine1	SELECT {meta_column1} , COUNT(*) as amount FROM {meta_table1} WHERE {meta_condition1} GROUP BY {meta_column1} HAVING COUNT(*) >1 ORDER BY COUNT(*) DESC
query_examine2	SELECT {meta_examine_columns1} FROM {meta_table1} WHERE {meta_column1} IN (SELECT {meta_column1} FROM {meta_table1} WHERE {meta_condition1} GROUP BY {meta_column1} HAVING COUNT(*) >1) ORDER BY {meta_column1}

Testide kirjeldustele (vt lisa 2) on lisatud iga testi tüübi jaoks ka metaandmete ja negatiivse testi tulemuse näidised. Tabelis 5.2 on toodud unikaalsuse testi metaandmete täitmise ja vigase tulemuse näide.

Tabel 5.2 Unikaalsuse testi metaandmete täitmise vajadus ja vigase testi tulemuse näited.

METAANDMED			
meta_table1	**ISIKUD**	meta_table2	-
meta_column1	**ISIK_ID**	meta_column2	-
meta_condition1		meta_condition2	-
meta_key1	-	meta_key2	-
meta_examine_columns1	*	meta_examine_columns2	-

VIGASE TESTI TULEMUSED	
query_result1	2
result_comment	Leidub 2 väärtust, mida esineb tunnuses ISIK_ID rohkem 1 kord.
Esimese abipäringu tulemus	**Esimene abipäring** tagastab loetelu tunnuse ISIK_ID väärtustest, mida esineb tabelis PROFILE rohkem, kui üks kord, kuvatakse ka väärtuse esinemise sagedus. isik_id integer / amount bigint 1: 1 / 2 2: 35 / 2
Teise abipäringu tulemus	**Teine abipäring** tagastab kirjed, kus tunnus ISIK_ID väärtus on mitte-unikaalsete väärtuste hulgas. isik_id integer / eesnimi text / perenimi text / sugu text / riik text / alates date / kuni date / staatus text / vanus integer 1: 1 / Foss / Yatman / Male / SE / 2018-07-29 / 2018-07-29 / VALID / 92 2: 1 / Foss / Yatman / Male / SE / 2018-07-29 / [null] / VALID / 92 3: 35 / Rhys / Landon / Male / SE / 2018-09-24 / [null] / VALID / 68 4: 35 / Foss / Yatman / Male / SE / 2018-07-29 / 2019-04-29 / CLOSED / 92

Neid testi tüüpide kirjeldusi saavad kasutada migratsiooni testide koostajad näidistestidena ja testide generaatori arendajad sisendina, et realiseerida päringute genereerimine metaandmete alusel.

6 Testimise korraldus

Migratsiooniprojekti planeerimise etapis (joonisel 2.2 etapp I1) täpsustakse testimise korraldus (sh testimise lähenemisviis ja keskkonnad) ja koostatakse projekti testiplaan. Testiplaan sisaldab testimise eesmärki, eeldusi, lähenemisviisi, keskkondi, ajakava ja tulemeid.

Migratsiooni testimise vajadus tekib keskkonna muutmise/muutumise järel, näiteks on vaja migratsiooni testida pärast migratsiooni käivitamist või lähteandmete muutumist. Selles peatükis kirjeldatakse mõningaid testide haldamise võimalusi ja testimise korraldamise lähenemisviise.

6.1 Ühe objekti andmete testimine

Vigade põhjuste uurimisel põhjalikuma analüüsi käigus on sagedamini esinev vajadus põhiobjektide (nt isikute) ja nende seoste analüüsimine.

Rakenduse kaudu testimisel (vt ptk 3.2.6) leitakse just objektiga (nt isiku) seotud vigu. Kui arvatav viga on migreeritud andmetes, siis on vaja tuvastada vea asukoht: kas viga on migreerimise skriptis või lähteandmetes. Sellisel juhul on vaja teha päringud mõlemasse baasi ning võrrelda objekti andmeid lähte- ja sihtbaasis.

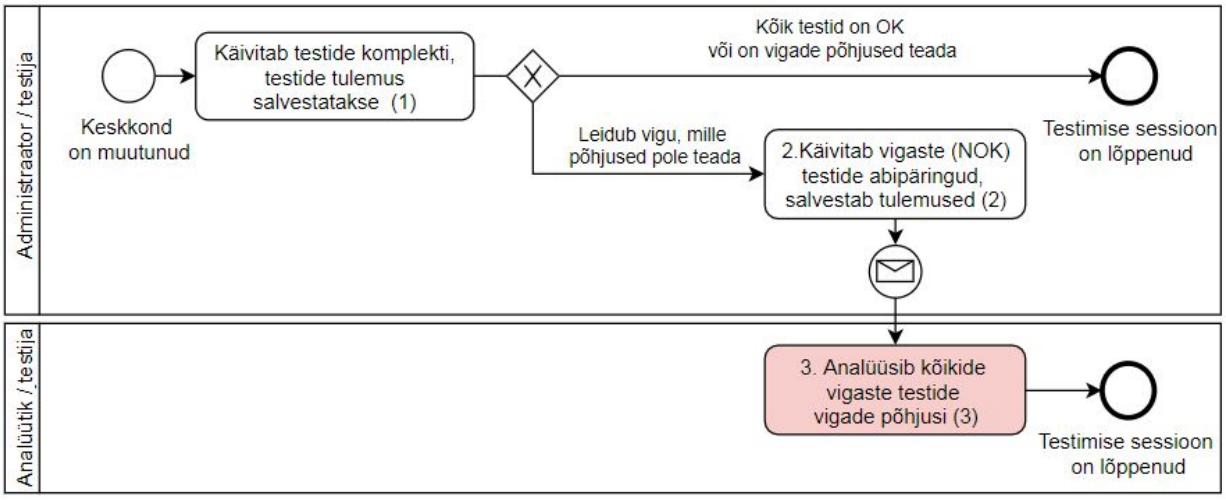
Samasuguse põhjalikuma objekti uurimise järele võib olla vajadus ka siis, kui testimisel osutub, et sihtbaasis puuduvad objektiga seotud kirjed või sihtbaasis on objektiga seotud kirjeid liiga palju.

Soovitame koostada vea põhjuste analüüsi lihtsustamiseks **olulisemate objektide kohta** abipäringute komplektid (nt sql-failid), kus muutuja (nt isiku_id) täpsustamisega saab kiiresti käivitada lähte- ja sihtbaasis erinevaid objektiga seotud päringuid.

6.2 Ettevalmistatud testide kasutamine

Migratsiooni testimisel on põhiline meetod andmebaasi sisu testimine (joonisel 3.3 tööd 1-A, 2-A, 2-B, 3-1) ettevalmistatud testidega.

Eelmistes peatükkides kirjeldatud abipäringute kasutamine võimaldab testimise korralduses lahutada testide käivitamine ja vigade analüüs (vt joonis 6.1) erinevateks etappideks.

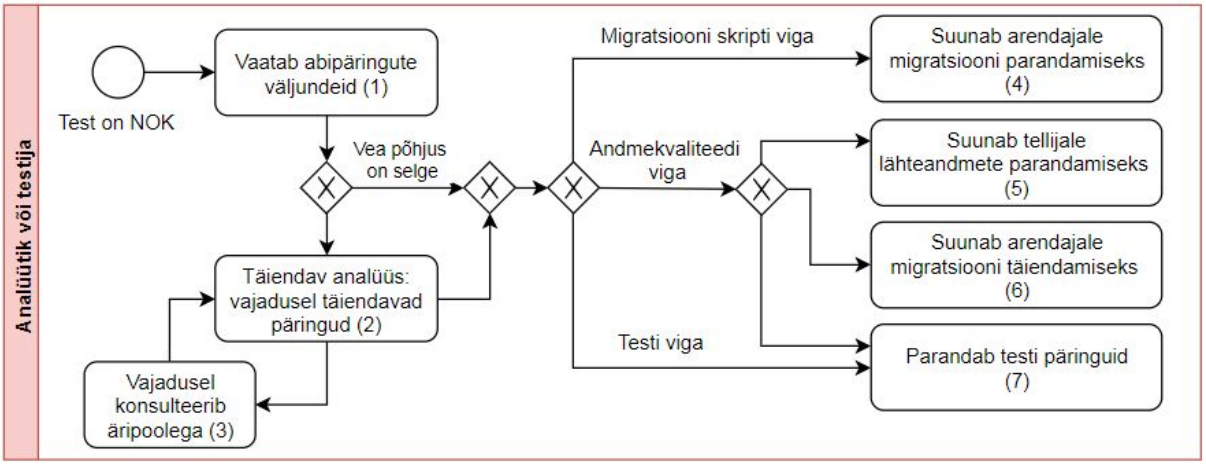


Joonis 6.1 Testimist saab jagada kahe rolli vahel. Teste käivitab administraator (1)(2), vigade põhjusi analüüsib migratsiooni tundev testija või analüütik (3).

Migratsiooni arenduskeskkonna testija tunneb andmebaasi: ta käivitab testi(d) ja analüüsib vigade põhjusi kohe. Kui tellija testkeskkonnas administraator on käivitanud migratsiooni, siis peab tal olema võimalus migratsiooni testidega veenduda migratsiooni korrektsuses. Selleks käivitab ta migratsiooni testid (joonis 6.1 (1)). Kui leidub vigase tulemusega (NOK) teste, siis käivitab ta ka vigaste tulemustega testide abipäringud (2) ning edastab need migratsiooni tundvale testijale. Paljudel juhtudel[17] on abipäringutega saadud info piisav vea põhjuste tuvastamiseks.

Vea põhjuste analüüs

Migratsiooni vea põhjuste analüüsi kirjeldab joonis 6.2. Vea põhjustest aru saamiseks võib piisata ainult vigade (abipäringute tulemuste) vaatamisest (1), kuid võib olla vaja ka sügavamat andmete analüüsi ning täiendavate päringute tegemist andmebaasidesse (2). Täiendavate päringute puhul on abiks ettevalmistud päringute mallid või päringute komplektid (nt objektiga seotud päringute komplektid (vt ptk 6.1)). Vahel võib tekkida vajadus vigaste andmete olemust ja ärinõuetele vastavust täpsustada äripoolega.



Joonis 6.2 Analüütiku tegevused testi vea põhjuse tuvastamisel.

Tuvastatud vea põhjusest sõltub, kellele vea parandamise töö suunatakse.

- Kui viga on migratsiooni skriptides, siis suunatakse vea parandamine arendajale (4).
- Kui osutub, et viga on lähteandmete kvaliteedis siis on vaja koos äripoolega otsustada, kuidas vigane olukord lahendada (vt ka ptk 3.2.1).
  - Kui vead otsustatakse parandada lähtebaasis, siis suunatakse töö tellijale (5).
  - Kui lähteandmetes on vea parandamine keeruline, võidakse otsustada vea mõju korrigeerida migratsiooni skriptidega. Sellisel juhul suunatakse töö arendajale migratsiooniskriptide täiendamiseks (6).
  - Kui vea analüüsil otsustatakse viga parandada hiljem uues infosüsteemis, siis on vaja muuta või kustutada ka tarbetuks muutunud testid (7).
  - Kui testimisel otsustatakse muuta/tühistada aluseks olevat nõuet, siis on vaja ka testi päringuid muuta/tühistada (7). Testid peavad olema kooskõlas muutunud nõuetega.
- Kui osutub, et vea põhjuseks olid vigased testi päringud, siis tuleb muuta testi päringuid (7).

[17] Autori kogemuse põhjal on 50%-60% juhtudest võimalik migratsiooni tundval isikul vea põhjus tuvastada abipäringute väljundi abil ilma täiendavaid päringuid tegemata.

6.3 Testide haldus

Testimise automatiseerimiseks on vaja valida testide haldamiseks niisugune viis, et testide käivitamist oleks võimalik automatiseerida.

Testide haldamine võiks vastata järgmistele tingimustele:

1. Korraga saab käivitada **kõiki teste või osa testidest**. Osade testide käivitamise võimaldamiseks võiks olla testid grupeeritud (nt migreerimise sammude või tabelite lõikes).
2. Igal testil peaks olema unikaalne **kood**, millele on saab viidata vigade parandamise töökäskudes. Soovitav on kasutada tekstilist koodi, mis sisaldab informatiivset eesliidet. Näiteks 'SAMM01', 'ISIKUD_023'.
3. Testide abipäringud on testi juures või on kergesti leitavad testi koodi alusel.
4. Testil võiks olla ka käivitamise staatus. Migratsiooni arendamise ajal ei ole vaja kõiki teste käivitada, sest osa migratsiooni funktsionaalsust on veel arendamata ('NOT_YET') või vea põhjus juba tuvastatud, kuid viga on veel parandamata ('BUG').

Testide haldamise ja testide automaatse käivitamise funktsionaalsus ei ole liiga keeruline. Määravaks saab testija/ arendaja valitud automatiseerimisel kasutav keel (PL/SQL, Python, *Visual Basic* jt) ja sellega sobiv testide päringute hoidmise viis (andmetabelites, PL/SQL protseduurides, tekstifailides, MS Exceli lahtrites).

MS Excelis testide haldamise ja testide kasutamine lahendust ühes eesti finantsasutuses kirjeldab Vaupere (Vaupere, 2015, lk 34–35). Lahenduses kasutatakse *Visual Basic*u makrosid testide päringute genereerimiseks, testide käivitamiseks, testimise tulemuse Exelisse salvestamiseks, tulemuste aruande genereerimiseks.

Järgnevas tuuakse kaks näidet testide halduse korraldamiseks SQLi vahenditega.

6.3.1 Näide: testid on PL/SQL protseduurides, abipäringud failides

Järgnev näide[18] tutvustab testide grupeerimist PL/SQL protseduuridesse ja testimisel protseduuri käivitamist.

Testide käivitamise protseduur (vt joonis 6.3) koosneb paljudest testidest, kus iga test moodustab omaette struktureeritud ploki.

1. Testi **päises** (1) väärtustatakse raportisse salvestamisel kasutatavad muutujad.
2. Testi sisuks on **testi päring**(2), mis korrektsete andmete põhjal peab andma tulemuseks null rida (vt joonisel esimene näide) või nulli (vt joonisel teine näide).
3. Viimases plokis on **tulemuse salvestamine** (3) tulemuste tabelisse. See plokk on kõikide testide korral ühesugune, sest muutujad on väärtustatud testi päises.

Niisugune protseduuri struktureerimine muudab testide vormistamise lihtsamaks ja skripti loetavaks.

[18] Autori osalusel ühes migratsiooniprojektis realiseeritud lähenemisviis.

```
DECLARE
   count numeric := 0;
   test_no varchar(50) := '';
   test_name text := '';
   nok_comment text := '';
   comment text := NULL;
   result varchar(5) := '';
   is_nok int := 0;
BEGIN
```

Testi päis (1)	`test_no := 'S1_1';`
	`test_name :='Isikukoodi unikaalsus';`
	`nok_comment :='Käivitada abipäring S1_1_abi1 ja analüüsida väljundit';`

Testi päring (2)	`IF (SELECT COUNT(*) FROM (SELECT ISIKUKOOD FROM ISIKUD GROUP BY ISIKUKOOD HAVING COUNT(*) >1 ) q1) = 0`
Salvestamine (3)	`THEN result:='OK'; comment := NULL; ELSE result:='NOK'; comment:=nok_comment; END IF;` `PERFORM test_log_action(test_no, test_name, result, comment, localtimestamp);`

Testi päis (1)	`test_no := 'S1_2';`
	`test_name :='Riigi koodi sobivus';`
	`nok_comment :='Käivitada abipäring S1_2_abi1 ja analüüsida väljundit';`

Testi päring (2)	`PERFORM SELECT RIIK_KOOD FROM ISIKUD WHERE RIIK_KOOD not in (SELECT KOOD FROM RIIGID);`
	`GET DIAGNOSTICS count = ROW_COUNT;`
	`IF count = 0`

Salvestamine (3)	`THEN result:='OK'; comment := NULL; ELSE result:='NOK'; comment:=nok_comment; END IF;` `PERFORM test_log_action(test_no, test_name, result, comment, localtimestamp);`

Joonis 6.3 Ühe protseduurina paljude testide käivitamise näide.

Protseduuri käivitamise tulemuseks on tulemuste tabel (vt joonis 6.4), kus iga testi kohta on tulemuse rida ja vigaste tulemusega testidel on viited abipäringutele, mis asuvad sql failides.

name charac	description character varying (500)	status character varying (50)	comment character varying (500)	test_time timestamp wit
S1_1	Isikukoodi unikaalsus	OK	[null]	2019-05-14 ...
S1_2	Riigi koodi sobivus	NOK	Käivitada abipäring S1_2_abi1 ja analüüsid...	2019-05-14 ...

Joonis 6.4 Testi tulemuste tabeli näide.

6.3.2 Näide: testide päringud on andmebaasis

Järgnev näide[19] tutvustab testide hoidmist ja haldamist SQL vahenditega, kahe andmetabeli ja paari PL/SQL protseduuriga.

Kõiki teste hoitakse andmebaasis **testide tabelis** (nt tabelis QUERIES). Iga testi kohta on tabelis üks kirje, mis sisaldab **testi üldandmeid** (testi kood, testi kirjeldus, testi staatus, testi grupp) ja **testiga seotud päringuid** (testi päring ja abipäringud).

Testi käivitamisel lisatakse iga käivitatud testi kohta rida **tulemuste tabelisse** (nt tabel RESULTS), kuhu

- salvestatakse testimise aeg,
- kopeeritakse testide tabelist testi üldandmed,
- kopeeritakse testide tabelist testi päringud ja abipäringud,
- salvestatakse testimise tulemus (OK/NOK ja vigaste testide korral veateade).

Selline lähenemisviis võimaldab SQL lausete abil hallata teste ning käivitada ainult soovitud osa testidest. Testide tulemuste tabelis on vigaste testide juures ka abipäringud, mida on lihtne käivitada. Säilib ka testimise ajalugu koos testimise hetkel kehtinud päringutega.

[19] Autor testis selle idee tomivust töö kirjutamise ajal Postgres.

Kokkuvõte

Magistritöös käsitleti infosüsteemide uuendamise projektides sisalduva migratsiooni testimist.

Magistritöö eesmärgiks seati migratsiooni testimise selgitamist ning otsiti lähenemisviise testimise lihtsustamiseks. Defineeriti kolm eesmärki:

1. **täpsustada migratsiooni testimise vajaduse ulatust;**
2. leida testimise lähenemisviis, mis lihtsustaks **testimist** ja vigade analüüsimist;
3. leida võimalusi **testide koostamise lihtsustamiseks (automatiseerimiseks).**

Esimene eesmärk: Täpsustada migratsiooni testimise vajaduse ulatus

Migratsiooni testimise vajaduse täpsustamisel võeti eeskujuks andmelaoprojekti testimise vajadus. Võrreldi andmelao projekti testimise sisu ja eesmärke migratsiooniprojekti nõuetega ning koostati **migratsiooni komponentide testimise mudel** (vt ptk 3.2). Seejärel koostati ülevaatlik skeem migratsiooni testimise vajadusest migratsiooniprojekti etappide lõikes (vt ptk 3.3).

Teine eesmärk: leida testimise lähenemisviis, mis lihtsustaks testimist ja vigade analüüsimist

Testimise korraldamise kirjeldamisel võeti aluseks ühe migratsiooniprojekti kogemus, kus autor oli migratsiooni analüüsi ja testimise eestvedaja rollis. Kirjeldatud lähenemis baseerub testi päringute kompletil: iga test koosneb isevalideeruva testi päringust ja vigu tagastatavatest äbipäringutest. Niisugune lähenemisviis võimaldab muuta analüüsimise efektiivsemaks ja jagada testimise kaheks osaks: testide käivitamiseks ja vigaste andmete analüüsiks.

Kolmas eesmärk: leida võimalusi testide koostamise lihtsustamiseks

Andmebaasi sisu testimisel päringute (SQL struktuuri) sarnasusega arvestamine võimaldab teste grupeerida ning testide jaoks päringute koostamist automatiseerida. Töö tulemusena defineeriti **16 testi tüüpi** (Vt ptk 5.2.3, 5.2.5), mis sisaldab testi päringut ja abipäringute malle (näidispäringuid) ning metaandmete vajadust (Vt lisa II).

Koostatud testide tüüpide malle saavad kasutada migratsiooni testijad **näidispäringutena**.

Testide tüüpide kirjelduste olemasolu võimaldab arendada ka päringute generaatori. Konkreetse testi genereerimise sisendparameetriteks oleks testi tüüp ja metaandmed. Homayouni näite (vt ptk 4.5) eeskujul koostas autor paari testitüübi jaoks PL/SQL protseduuri, kus metaandmed võeti csv failist ja koostatud testide päringud salvestati andmetabelisse, et hiljem neid saaks andmebaasis käivitada. Seega idee on teostatav. Kuna mallide ja metaandmete sidumine on lihtne tekstitöötluse operatsioon, siis võibolla on mõistlikum valida mõni tekstitöötlusele paremini sobiv programmeerimise keel (nt Python).

Ja autori soovitus on alati koos isevalideeruvate testidega koostada kohe ka abipäringud (vt ptk 5.2.2), sest selliste päringute koos koostamine võtab vähem aega ja säästetakse oluliselt aega hiljem vigade analüüsimisel.

Lõpetuseks

Töö eesmärgid täideti ja selle tööga tekkis eestikeelne materjal, kust migratsiooni testijad saavad koguda ideid oma projektis migratsiooni testimise korraldamiseks.

Kuigi töös räägitakse migratsiooni testimisest, siis väga palju selle töö sisust on kasutatav ka **andmekvaliteedi monitooringu** korraldamisel. Pooled defineeritud testi tüüpidest sobivad kvaliteeditestide koostamiseks ja testimise metoodika sobib ka regulaarselt toimiva monitooringu korraldamiseks.

Tänan oma kolleege Karl Sobakut (Industry 62) ja Ragnar Ziugandit (Cybernetica), kellega koostöös ühes migratsiooniprojektis väljatöötatud ideed ka sellesse magistritöösse jõudsid.

Viidatud kirjandus

Althani, B., & Khaddaj, S. (2017). Systematic Review of Legacy System Migration. *2017 16th International Symposium on Distributed Computing and Applications to Business, Engineering and Science (DCABES)*, 154–157. https://doi.org/10.1109/DCABES.2017.41

Cybernetica. (2019). AKIT - Andmekaitse ja infoturbe leksikon. Kasutatud 5. märts 2019, https://akit.cyber.ee/

Data Warehouse. (i.a). Kasutatud 14. mai 2019, Gartner kodulehekülg: https://www.gartner.com/it-glossary/data-warehouse

ETL Concepts. (2015). Kasutatud 14. mai 2019, LearnDataModeling.com kodulehekülg: https://learndatamodeling.com/blog/etl-concepts/

Gupta, L. (i.a). FIRST Principles for Writing Good Unit Tests. Kasutatud 14. aprill 2019, https://howtodoinjava.com/best-practices/first-principles-for-good-tests/

Haller, K. (2009). Towards the Industrialization of Data Migration: Concepts and Patterns for Standard Software Implementation Projects. *Advanced Information Systems Engineering*, 63–78. Springer Berlin Heidelberg.

Haller, K., Matthes, F., & Schulz, C. (2012). A Detailed Process Model for Large Scale Data Migration Projects. *Business Information Systems*, 165–176. Springer Berlin Heidelberg.

Homayouni, H. (2018). *An Approach For Testing The Extract-Transform-Load Process In Data Warehouse Systems* (Magistritöö, Colorado State University). Kasutatud http://doi.acm.org/10.1145/3216122.3216149

Horward, P. (2011). White Paper. Data Migration. Kasutatud 14. aprill 2019, https://docplayer.net/13730054-White-paper-data-migration.html

Minus Queries and Testing. (i.a). Kasutatud 13. aprill 2019, QuerySurge kodulehekülg: http://www.querysurge.com/solutions/minus-queries

Rangarajan, S. (2016). Data Warehouse Design – Inmon versus Kimball. Kasutatud 14. mai 2019, TDAN.com kodulehekülg: http://tdan.com/data-warehouse-design-inmon-versus-kimball/20300

Rouse, M. (2017). What is denormalization? Kasutatud 14. mai 2019, https://searchdatamanagement.techtarget.com/definition/denormalization

Sadalage, P. (2015). 8 Techniques for testing migration of data from legacy systems · Passionate about data. Kasutatud 13. mai 2019, https://sadalage.com/blog/2015/01/23/8-techniques-for-testing-migration-of-data-from-legacy-systems/

Sampling Method of Data Validation. (i.a). Kasutatud 13. aprill 2019, QuerySurge kodulehekülg: http://www.querysurge.com/solutions/sampling

Struktuurpäringukeel. (2016). Kasutatud 13. mai 2019, Vikipeedia kodulehekülg: https://et.wikipedia.org/w/index.php?title=Struktuurp%C3%A4ringukeel&oldid=4514969

Types of Migration Testing: With Test Scenarios for Each Type. (2019). Kasutatud 13. mai 2019, https://www.softwaretestinghelp.com/migration-testing-types/

Vaupere, H. (2015). *Testimise parendamise meetodid andmeaidas.* (Bakalaureuse töö). Tallinna Tehnikaülikool.

Veldre, A. (2015). Räsist ja räsimisest. Kasutatud 14. mai 2019, RIA blogi koduleheküIg: https://blog.ria.ee/rasist-ja-rasimisest/

Verification vs Validation: Do you know the difference? (2019). Kasutatud 14. mai 2019, Plutora koduleheküIg: https://www.plutora.com/blog/verification-vs-validation

Wei, B., & Chen, T. X. (2014). Verifying Data Migration Correctness: The Checksum Principle. *RTI Press publication OP-0019-1403*. Kasutatud https://docplayer.net/17251232-Verifying-data-migration-correctness-the-checksum-principle.html

What Is Data Migration? (i.a). Kasutatud 13. mai 2019, https://www.netapp.com/us/info/what-is-data-migration.aspx

What is Software Testing? Introduction, Definition, Basics & Types. (i.a). Kasutatud 14. mai 2019, https://www.guru99.com/software-testing-introduction-importance.html

Woodall, P., Oberhofer, M., & Borek, A. (2014). A Classification of Data Quality Assessment and Improvement Methods. *International Journal of Information Quality (IJIQ)*, (Vol. 3, No. 4, 2014).

LISAD

I Testi metaandmed

Testide tüüpide kasutamisel konkreetsete testide genereerimiseks on vaja testi kirjeldamiseks täita metatunnused. Iga testi jaoks tuleb ette valmistada üks komplekt metaaandmeid.

- Testi haldamisega seotud tunnused on prefiksiga 'test_'. Neid tunnuseid saab kasutada testide haldamisel, näiteks töötades ainult ühe grupi või ühe tüübi testidega.
- Testi päringutega seotud metatunnused algavad prefiksiga 'meta_' ja nende tunnuste abil saab generaatorile ette anda testi päringute genereerimiseks vajalikud sisend-andmed.

Tunnus	Täitmise selgitus
test_code	**Testi unikaalne kood.** Koodides soovitame kasutada prefikseid, mis viitavad näiteks testi grupile või migratsiooni skripti sammule.
test_group	**Testi grupp.** Abitunnus halduse paindlikumaks muutmiseks. Näiteks 'KVALITEET' või 'SAMM1'.
test_description	**Testi kirjeldus** ja/või kontrollitav nõue.
test_type	**Testi tüüp** määrab ära testpäringute struktuuri ehk testide genereerimisel kasutatavad mallid.
meta_table1	**Tabeli nimi**. Võrdlevate testide puhul kasutatakse seda lähtetabelina. Kahte tabeliga päringute korral kasutatakse on lühinime t1.
meta_column1	**Tunnuse nimi.** Võrdlevate testide puhul kasutatakse seda lähtetabeli tunnusena.
meta_condition1	**Filtritingimus.** Esimese tabeli filtritingimused. Tingimuse kirjeldus peab vastama SQL süntaksile. Mitme tingimuse korral peavad olema tingimused seotud operaatoritega (AND, OR).
meta_table2	**Teine tabel.** Võrdlevate testide puhul kasutatakse seda sihttabelina. Kahte tabelit kasutavate päringute korral on lühinimi t2.
meta_column2	**Tunnuse nimi.** Võrdlevate testide puhul kasutatakse seda sihttabeli tunnusena.
meta_condition2	**Filtritingimus.** Teise tabeli filtritingimused. Tingimuse kirjeldus peab vastama SQL süntaksile. Mitme tingimuse korral peavad olema tingimused seotud operaatoritega(AND, OR).
meta_key1	1. **Väärtus**. Ühe päringuga testides kasutatakse väärtuse võrdlemiseks. Vahemikke kontrollivate testide korral **miinimum väärtus**. Kui vahemike testis on täitmata teine väärtus, siis *'suurem kui {meta_key1}'* Mitme väärtuse korral antakse ette loetelu. Näiteks ('KEHTETU','TÜHISTATUD'). 2. **Seose tunnus.** Kahe päringuga testide korral kasutatakse seda tunnust tabelite vaheliste seose loomisel esimese tabeli võtmetunnusena (t1.meta_key1= t2.meta_key2).
meta_key2	1. **Väärtus**. Ühe päringuga testides väärtuste võrdlemisel **maksimum väärtus**. Kui vahemike testis on täitmata esimene väärtus, siis *'väiksem kui {meta_key2}'*. 2. **Seose tunnus.** Kahe päringuga testide korral kasutatakse seda tunnust tabelite vaheliste seose loomisel teise tabeli võtmetunnusena (t1.meta_key1= t2.meta_key2).
meta_examine_columns1	**Loetelu abipäringu tunnustest**. Kahe tabeliga päringute korral esimese tabeli tunnused. Võib sisaldada viiteid tabelite lühinimedele t1 ja t2. ja kasutada ümbernimetamisi. Kui see tunnus on täitmata, siis kasutatakse tärni (*) ehk antakse kõik valitud tabelite tunnused.
meta_examine_columns2	**Loetelu abipäringu** teise tabeli **tunnustest** (SELECT) kahe tabeliga päringutes. Võib sisaldada viiteid mõlemale tabeli lühinimedele t1 ja t2. ja kasutada ümbernimetamisi. Kui see tunnus on täitmata, siis kasutatakse tärni (*) ehk antakse kõik valitud tabelite tunnused.

II Testide tüüpide päringute mallid

Siin on 16 testi tüübi **metaandmed** ja **päringute genereerimise mallid.**

Metaandmete paneelis on toodud metaandmete täitmise vajadus. Antud testi tüübi jaoks kohustuslikud metatunnused on kuvatud rasvase (*bold*) kirjaviisiga.

Päringute mallide paneelis on päringute ja abipäringute SQL lausete konstruktsioonid.

- QUERY1, QUERY2 on testi päringute mallid,
- RESULT_COMMENT on mittekorrektse tulemuse (NOK) veateate mall,
- QUERY_EXAMINE1, QUERY_EXAMINE2 on abipäringute mallid.

Iga testi kohta on lisatud ka vigase testi tulemuse ja abipäingute tulemuste näidis[20].

1 Testi tüüp NOT_NULL

Testi tüübi lühikirjeldus:
Kohustuslikkuse kontroll. Kontrollitakse, et etteantud tingimuste korral on tunnus {meta_column1} täidetud.
Esimene abipäring tagastab kirjed, kus tunnus {meta_column1} on täitmata.

METAANDMED			
meta_table1	**Tabel**	meta_table2	-
meta_column1	**Tunnus**	meta_column2	-
meta_condition1	Filtritingimus	meta_condition2	-
meta_key1	-	meta_key2	-
meta_examine_columns1	Tunnuste loetelu	meta_examine_columns2	-

PÄRINGUTE JA TULEMUSTE GENEREERIMISE MALLID	
query1	SELECT COUNT(*) FROM {meta_table1} WHERE {meta_condition1} AND {meta_column1} IS NULL
query2	-
	Testitud andmed on korrektsed, kui esimese päringu tulemus on 0
result_comment	NOK: Leidub {query1_result} kirjet, kus tunnus {meta_column1} on täitmata.
query_examine1	SELECT {meta_examine_columns1} FROM {meta_table1} WHERE {meta_condition1} AND {meta_column1} IS NULL
query_examine2	-

[20] Näidisandmete genereerimiseks kasutati generaatorit MOCAROO https://mockaroo.com/.

Negatiivse tulemusega testi väljundite näidis - tüüp NOT_NULL

Näidistesti kirjeldus: kontrollitakse PERSONS tabelis kehtivuse alguse täidetust.

METAANDMED			
meta_table1	**PERSON**	meta_table2	-
meta_column1	**VALID_FROM**	meta_column2	-
meta_condition1		meta_condition2	-
meta_key1	-	meta_key2	-
meta_examine_columns1	id, status, first_name, last_name, valid_from, valid_to	meta_examine_columns2	-

VIGASE TESTI TULEMUSED	
query_result1	5
result_comment	Leidub 5 kirjet, kus tunnus VALID_FROM on täitmata.
Esimese abipäringu tulemus	**Esimene abipäring** tagastab kirjed, kus tunnus VALID_FROM on täitmata.

	id integer	status text	first_name text	last_name text	valid_from date	valid_to date
1	20	[null]	Ford	Yewdale	[null]	[null]
2	143	[null]	Kamillah	Bilton	[null]	[null]
3	148	[null]	Willem	Mosedill	[null]	[null]
4	184	[null]	Erik	Senior	[null]	[null]
5	230	[null]	Muhammad	McNirlin	[null]	[null]

2 Testi tüüp IS_NULL

Testi tüübi lühikirjeldus:

Kontrollitakse, et etteantud tingimuste korral oleks tunnus {meta_column1} täitmata (NULL).

Esimene abipäring tagastab kirjed, kus tunnus {meta_column1} on täidetud.

Teine abipäring tagastab tunnuse {meta_column1} väärtuste jaotustabeli.

METAANDMED			
meta_table1	**Tabel**	meta_table2	-
meta_column1	**Tunnus**	meta_column2	-
meta_condition1	Filtritingimus	meta_condition2	-
meta_key1	-	meta_key2	-
meta_examine_columns1	Tunnuste loetelu	meta_examine_columns2	-

PÄRINGUTE JA TULEMUSTE GENEREERIMISE MALLID	
query1	SELECT COUNT(*) FROM {meta_table1} WHERE {meta_condition1} AND {meta_column1} IS NOT NULL
query2	-
	Testitud andmed on korrektsed, kui esimese päringu tulemus on 0
result_comment	NOK: Leidub {query1_result} kirjet, kus tunnus {meta_column1} ei ole tühi.
query_examine1	SELECT {meta_examine_columns1} FROM {meta_table1} WHERE {meta_condition1} AND {meta_column1} IS NOT NULL
query_examine2	SELECT {meta_column1}, COUNT(*) FROM {meta_table1} WHERE {meta_condition1} AND {meta_column1} IS NOT NULL GROUP BY {meta_column1} ORDER BY COUNT(*) DESC

Negatiivse tulemusega testi väljundite näidis - tüüp IS_NULL

Näidistesti kirjeldus: Kontrollitakse, et tunnus VALID_TO oleks täitmata (NULL).

METAANDMED			
meta_table1	**PERSON**	meta_table2	-
meta_column1	**VALID_TO**	meta_column2	-
meta_condition1		meta_condition2	-
meta_key1	-	meta_key2	-
meta_examine_columns1	id, status, first_name, last_name, valid_from, valid_to	meta_examine_columns2	-

VIGASE TESTI TULEMUSED	
query_result1	7
result_comment	Leidub 7 kirjet, kus tunnus valid_to ei ole tühi.
Esimese abipäringu tulemus	**Esimene abipäring** tagastab kirjed, kus tunnus VALID_TO on täidetud.
Teise abipäringu tulemus	**Teine abitabel** tagastab tunnuse VALID_TO väärtuste jaotustabeli.

	id integer	status text	first_name text	last_name text	valid_from date	valid_to date
1	1	VALID	Foss	Yatman	2018-07-29	2018-07-29
2	29	CLOSED	Guillaume	Bellingham	2018-11-14	2018-06-26
3	102	INVALID	Horatius	Donaldson	2018-02-17	2018-04-12
4	118	CLOSED	Heinrick	Spurman	2018-06-24	2018-11-22
5	199	VALID	Thatcher	Ead	2018-10-27	2018-06-26
6	217	CLOSED	Jed	Saggs	2018-06-12	2018-05-10
7	35	CLOSED	Foss	Yatman	2018-07-29	2019-04-29

	valid_to date	count bigint
1	2018-06-26	2
2	2018-04-12	1
3	2018-05-10	1
4	2018-07-29	1
5	2018-11-22	1
6	2019-04-29	1

3 Testi tüüp STATIC_VALUE

Testi tüübi lühikirjeldus:
Kontrollitakse, et etteantud tingimuste korral oleks tunnuse {meta_column1} väärtus võrdne etteantud väärtusega *{meta_key1}*.
Esimene abipäring tagastab kirjed, kus tunnuse {meta_column1} väärtus ei ole {meta_key1}.
Teine abipäring tagastab tunnuse {meta_column1} lubamatute väärtuste jaotustabeli.

METAANDMED			
meta_table1	**Tabel**	meta_table2	-
meta_column1	**Tunnus**	meta_column2	-
meta_condition1	Filtritingimus	meta_condition2	-
meta_key1	**Etteantud väärtus**	meta_key2	-
meta_examine_columns1	Tunnuste loetelu	meta_examine_columns2	-

PÄRINGUTE JA TULEMUSTE GENEREERIMISE MALLID	
query1	SELECT COUNT(*) FROM {meta_table1} WHERE {meta_condition1} and {meta_column1}!={meta_key1}
query2	-
	Testitud andmed on korrektsed, kui esimese päringu tulemus on 0
result_comment	NOK: Leidub {query1_result} kirjet, kus tunnuse {meta_column1} väärtus ei ole {meta_key1}.
query_examine1	SELECT {meta_examine_columns1} FROM {meta_table1} WHERE {meta_condition1} and {meta_column1}!={meta_key1}
query_examine2	SELECT {meta_column1}, COUNT(*) FROM {meta_table1} WHERE {meta_column1}!={meta_key1} GROUP BY {meta_column1} ORDER BY COUNT(*) DESC

Negatiivse tulemusega testi väljundite näidis - tüüp STATIC_VALUE

Näidistesti kirjeldus: Kontrollitakse et, pangakaartide tabelis oleks staatus ‘VALID’.

METAANDMED			
meta_table1	**CARD**	meta_table2	-
meta_column1	**STATUS**	meta_column2	-
meta_condition1		meta_condition2	-
meta_key1	‘VALID’	meta_key2	-
meta_examine_columns	id,person_id,card_type,card_nr	meta_examine_columns2	-

VIGASE TESTI TULEMUSED	
query_result1	3
result_comment	leidub 3 kirjet, kus tunnuse STATUS väärtus ei ole ‘VALID’.
Esimese abipäringu tulemus	**Esimene abipäring** tagastab kirjed, kus tunnuse STATUS väärtus ei ole ’VALID’.
Teise abipäringu tulemus	**Teine abipäring** tagastab tunnuse STATUS lubamatute väärtuste jaotustabeli.

	status text	id integer	person_id integer	card_type text	card_nr text
1	ACTUAL	265	38	jcb	3589891609295972
2	VALIDD	456	17	maestro	5018234761973264387
3	ACTUAL	15	204	mastercard	5002359841302042

	status text	count bigint
1	ACTUAL	2
2	VALIDD	1

4 Testi tüüp IN_VALUES

Testi tüübi lühikirjeldus: Kontrollitakse, et etteantud tingimuste korral kuulub tunnuse {meta_column1} täidetud väärtus lubatud väärtuste loetellu {meta_key1}.

Esimene abipäring tagastab kirjed, kus tunnuse {meta_column1} väärtus on täidetud, aga ei kuulu lubatud väärtuste loetellu {meta_key1}.

Teine abipäring tagastab tunnuse {meta_column1} lubamatute väärtuste jaotustabeli.

METAANDMED			
meta_table1	**Tabel**	meta_table2	-
meta_column1	**Tunnus**	meta_column2	-
meta_condition1	Filtritingimus	meta_condition2	-
meta_key1	**Lubatud väärtuste loetelu**	meta_key2	-
meta_examine_columns1	Tunnuste loetelu	meta_examine_columns2	-

PÄRINGUTE JA TULEMUSTE GENEREERIMISE MALLID	
query1	SELECT COUNT(*) FROM {meta_table1} WHERE {meta_condition1} and {meta_column1} NOT IN ({meta_key1})
query2	-
	Testitud andmed on korrektsed, kui esimese päringu tulemus on 0
result_comment	NOK: Leidub {query1_result} kirjet, kus tunnuse {meta_column1} väärtus ei ole lubatud väärtuste loetelus: {meta_key1}.
query_examine1	SELECT {meta_examine_columns1} FROM {meta_table1} WHERE {meta_condition1} and {meta_column1} NOT IN ({meta_key1})
query_examine2	SELECT {meta_column1}, COUNT(*) FROM {meta_table1} WHERE {meta_condition1} and {meta_column1} NOT IN ({meta_key1}) GROUP BY {meta_column1} ORDER BY COUNT(*) DESC

Negatiivse tulemusega testi väljundite näidis - tüüp IN_VALUES

Näidistesti kirjeldus: Isiku sugu peab olema loetelus (‘Female’ või ‘Male’)

METAANDMED			
meta_table1	**person**	meta_table2	-
meta_column1	**gender**	meta_column2	-
meta_condition1		meta_condition2	-
meta_key1	‘Female’ või ‘Male’	meta_key2	-
meta_examine_columns1	gender, id, status, first_name, last_name	meta_examine_columns2	-

VIGASE TESTI TULEMUSED	
query_result1	5
result_comment	Leidub 5 kirjet, kus tunnuse GENDER väärtus ei ole lubatud väärtuste loetelus: ‘Female’ või ‘Male’.
Esimese abipäringu tulemus	**Esimene abipäring** tagastab kirjed, kus tunnuse {meta_column1} väärtus on täidetud, aga ei kuulu lubatud väärtuste loetellu ‘Female’ või ‘Male’.
Teise abipäringu tulemus	**Teine abipäring** tagastab tunnuse GENDER lubamatute väärtuste jaotustabeli.

	gender text	id integer	status text	first_name text	last_name text
1	M	305	INVALID	Winn	Meldrum
2	M	341	INVALID	Godart	Carlens
3	F	49	INVALID	Madalena	Gargett
4	F	272	CLOSED	Golda	Kiely
5	F	304	VALID	Shoshanna	Ville

	gender text	count bigint
1	F	3
2	M	2

5 Testi tüüp UNIQUE_VALUE

Testi tüübi lühikirjeldus: Kontrollitakse, et etteantud tingimuste korral oleks tunnuse väärtused unikaalsed.

Esimene abipäring tagastab loetelu tunnuse {meta_column1} väärtustest, mida esineb tabelis {meta_table1} rohkem, kui üks kord. Kuvatakse ka väärtuse esinemise sagedus.

Teine abipäring tagastab kirjed, kus tunnus {meta_column1} väärtus on mitteunikaalsete väärtuste hulgas.

METAANDMED			
meta_table1	**Tabel**	meta_table2	-
meta_column1	**Tunnus**	meta_column2	-
meta_condition1	Filtritingimus	meta_condition2	-
meta_key1		meta_key2	-
meta_examine_columns1	Tunnuste loetelu	meta_examine_columns2	-

PÄRINGUTE JA TULEMUSTE GENEREERIMISE MALLID	
query1	SELECT COUNT(*) FROM (SELECT {meta_column1} FROM {meta_table1} WHERE {meta_condition1} GROUP BY {meta_column1} HAVING COUNT(*) >1) q
query2	-
	Testitud andmed on korrektsed, kui esimese päringu tulemus on 0
result_comment	NOK: Leidub {query1_result} väärtust, mida esineb tunnuses {meta_column1} rohkem kui 1 kord.
query_examine1	SELECT {meta_column1} , COUNT(*) as amount FROM {meta_table1} WHERE {meta_condition1} GROUP BY {meta_column1} HAVING COUNT(*) >1 ORDER BY COUNT(*) DESC
query_examine2	SELECT {meta_examine_columns1} FROM {meta_table1} WHERE {meta_column1} IN (SELECT {meta_column1} FROM {meta_table1} WHERE {meta_condition1} GROUP BY {meta_column1} HAVING COUNT(*) >1) ORDER BY {meta_column1}

Negatiivse tulemusega testi väljundite näidis - tüüp UNIQUE_VALUE

Näidistesti kirjeldus: Kontrollitakse tabelis ISIKUD tunnuse ISIK_ID unikaalsust.

METAANDMED			
meta_table1	**ISIKUD**	meta_table2	-
meta_column1	**ISIK_ID**	meta_column2	-
meta_condition1		meta_condition2	-
meta_key1	-	meta_key2	-
meta_examine_columns1	*	meta_examine_columns2	-

VIGASE TESTI TULEMUSED	
query_result1	2
result_comment	Leidub 2 väärtust, mida esineb tunnuses ISIK_ID rohkem 1 kord.
Esimese abipäringu tulemus	**Esimene abipäring** tagastab loetelu tunnuse ISIK_ID väärtustest, mida esineb tabelis PROFILE rohkem, kui üks kord, kuvatakse ka väärtuse esinemise sagedus. isik_id integer / amount bigint 1: 1, 2 2: 35, 2
Teise abipäringu tulemus	**Teine abipäring** tagastab kirjed, kus tunnus ISIK_ID väärtus on mitte-unikaalsete väärtuste hulgas. isik_id integer / eesnimi text / perenimi text / sugu text / riik text / alates date / kuni date / staatus text / vanus integer 1: 1, Foss, Yatman, Male, SE, 2018-07-29, 2018-07-29, VALID, 92 2: 1, Foss, Yatman, Male, SE, 2018-07-29, [null], VALID, 92 3: 35, Rhys, Landon, Male, SE, 2018-09-24, [null], VALID, 68 4: 35, Foss, Yatman, Male, SE, 2018-07-29, 2019-04-29, CLOSED, 92

6 Testi tüüp CODE

Testi tüübi lühikirjeldus: Kontrollitakse, kas tunnuse {meta_column1} kõik väärtused on esindatud teises (nn klassifikaatori) tabelis {meta_table2}. Sobib välisvõtmete (*foreign key*) kontrollimiseks.

Esimene abipäring tagastab tabeli {meta_table1} kirjed, kus tunnuse {meta_column1} väärtus puudub tabelist {meta_table2} .

Teine abipäring tagastab tunnuse {meta_column1} lubamatute väärtuste jaotustabeli.

METAANDMED			
meta_table1	**Tabel**	meta_table2	**Klassfikaatori tabel**
meta_column1	**Tunnus**	meta_column2	**Klassifikaatori kood**
meta_condition1	Filtritingimus	meta_condition2	Filtritingimus
meta_key1		meta_key2	-
meta_examine_columns1	Tunnuste loetelu	meta_examine_columns2	-

PÄRINGUTE JA TULEMUSTE GENEREERIMISE MALLID	
query1	SELECT COUNT(*) FROM {meta_table1} WHERE {meta_condition1} AND {meta_column1} NOT IN (SELECT {meta_column2} FROM {meta_table2} WHERE {meta_condition2})
query2	-
	Testitud andmed on korrektsed, kui esimese päringu tulemus on 0
result_comment	NOK: Leidub {query1_result} kirjet, kus tunnuses {meta_table1} .{meta_column1} on väärtus, mida ei leidu tabelis {meta_table2}
query_examine1	SELECT {meta_examine_columns1} FROM {meta_table1} WHERE {meta_condition1} AND {meta_column1} NOT IN (SELECT {meta_column2} FROM {meta_table2} WHERE {meta_condition2})
query_examine2	SELECT {meta_column1}, COUNT(*) FROM {meta_table1} WHERE {meta_condition1} AND {meta_column1} NOT IN (SELECT {meta_column2} FROM {meta_table2} WHERE {meta_condition2}) GROUP BY {meta_column1} ORDER BY COUNT(*) DESC

Negatiivse tulemusega testi väljundite näidis - tüüp CODE

Näidistesti kirjeldus: Kontrollitakse, kas isikute tabelis riigi kood leidub riikide tabelis.

METAANDMED			
meta_table1	**person**	**meta_table2**	**country**
meta_column1	**country_code**	**meta_column2**	**code**
meta_condition1		meta_condition2	
meta_key1	-	meta_key2	-
meta_examine_columns1	country_code, id, last_name, status	meta_examine_columns2	-

VIGASE TESTI TULEMUSED	
query_result1	76
result_comment	Leidub 76 kirjet, kus tunnuses person.country_code on väärtus, mida ei leidu tabelis country.
Esimese abipäringu tulemus	**Esimene abipäring** tagastab tabeli PERSON kirjed, kus tunnuse COUNTRY_CODE väärtus puudub tabelist COUNTRY . country_code text / id integer / last_name text / status text 1 [null] 2 Drackford CLOSED 2 [null] 9 Menear INVALID 3 LV 13 Ingre VALID 4 [null] 26 Turbard INVALID jne
Teise abipäringu tulemus	**Teine abipäring** tagastab tunnuse COUNTRY_CODE lubamatute väärtuste jaotustabeli. country_code text / count bigint [null] 72 LV 4

7 Testi tüüp VALUE_BETWEEN

Testi tüübi lühikirjeldus:

Kontrollitakse, et etteantud tingimuste korral jäävad tunnuse {meta_column1} väärtused etteantud vahemikku {meta_key1} < {meta_column1} < {meta_key2}.

- Kui jätta miinimum määramata ({meta_key1}=NULL), siis kontrollitakse, et väärtus on väiksem, kui {meta_key2}.
- Kui jätta maksimum määramata ({meta_key2} =NULL), siis kontrollitakse, et väärtus on suurem, kui {meta_key1}.

Esimene abipäring tagastab tabeli {meta_table1} kirjed, kus tunnuse {meta_column1} väärtus ei kuulu etteantud vahemikku {meta_key1} < {meta_column1} < {meta_key2}.

Teine abipäring tagastab tunnuse {meta_column1} lubatud vahemikku mittekuuluvate väärtuste jaotustabeli.

METAANDMED			
meta_table1	**Tabel**	meta_table2	**-**
meta_column1	**Tunnus**	meta_column2	**-**
meta_condition1	Filtritingimus	meta_condition2	-
meta_key1	lubatud miinimum või NULL	meta_key2	Lubatud maksimum või NULL
meta_examine_columns1	Tunnuste loetelu	meta_examine_columns2	-

PÄRINGUTE JA TULEMUSTE GENEREERIMISE MALLID	
query1	SELECT COUNT(*) FROM {meta_table1} WHERE {meta_condition1} AND ({meta_column1} <= {meta_key1} OR {meta_column1}>={meta_key2})
query2	-
	Testitud andmed on korrektsed, kui esimese päringu tulemus on 0
result_comment	NOK: Leidub {query1_result} rida, millele vastav väärtus ei ole vahemikus ({meta_key1}-{meta_key2}).
query_examine1	SELECT {meta_examine_columns1} FROM {meta_table1} WHERE {meta_condition1} AND ({meta_column1} <= {meta_key1} OR {meta_column1}>={meta_key2})
query_examine2	*SELECT {meta_column1}, COUNT(*)* FROM {meta_table1} WHERE {meta_condition1} AND ({meta_column1} <= {meta_key1} OR {meta_column1}>={meta_key2}) *GROUP BY {meta_column1}* *ORDER BY COUNT(*) DESC*

Negatiivse tulemusega testi väljundite näidis - tüüp VALUE_BETWEEN

Näidistesti kirjeldus: Isiku vanus peab olema väiksem kui 115.

METAANDMED			
meta_table1	**person**	**meta_table2**	
meta_column1	**country_code**	**meta_column2**	
meta_condition1		meta_condition2	
meta_key1		meta_key2	115
meta_examine_columns1	country_code, id, last_name, status	meta_examine_columns2	-

VIGASE TESTI TULEMUSED	
query_result1	8
query_result2	-
result_comment	Leidub 8 kirjet, millele vastav väärtus ei ole väiksem, kui 115
Esimese abipäringu tulemus	**Esimene abipäring** tagastab tabeli person kirjed, kus vanus on suurem ,kui 115
Teise abipäringu tulemus	**Teine abipäring** tagastab vanuste jaotustabeli, kus vanus >115..

	age integer	id integer	gender text	valid_from date	valid_to date
1	120	30	Male	2018-05-11	[null]
2	999	133	Female	2018-03-08	[null]
3	119	263	Female	2018-08-13	[null]
4	999	256	Male	2018-06-08	[null]
5	119	381	Female	2019-03-22	[null]
6	120	394	Male	2018-12-16	[null]
7	999	88	Female	2019-03-13	[null]
8	999	155	Female	2018-11-26	[null]

	age integer	count bigint
1	999	4
2	120	2
3	119	2

8 Testi tüüp METRIC_COUNT

Testi tüübi lühikirjeldus: Mõõdikuga kontrollitakse, kas etteantud tingimustega kirjete arv jääb oodatavasse vahemikku {meta_key1} <= kirjete arv <= {meta_key2}.

METAANDMED			
meta_table1	**Tabel**	meta_table2	**-**
meta_column1	**-**	meta_column2	**-**
meta_condition1	Filtritingimus	meta_condition2	-
meta_key1	**Oodatav miinimum**	**meta_key2**	**Oodatav maksimum**
meta_examine_columns1	-	meta_examine_columns2	-
Metaandmete täitmise kommentaar: Metaandmetes peab olema täidetud vähemalt üks piiridest: {meta_key1} või {meta_key2} või mõlemad korraga.			

PÄRINGUTE JA TULEMUSTE GENEREERIMISE MALLID	
query1	SELECT COUNT (meta_column1) FROM {meta_table1} WHERE {meta_condition1}
query2	-
	Testitud andmed on korrektsed, kui esimese päringu tulemus jääb etteantud vahemikku, st. **{meta_key2} <= esimese päringu tulemus <= {meta_key2}**
result_comment	ERROR: Väärtused puuduvad NOK+: liiga palju kirjeid. Maksimumist {query1 _result} - {meta_key2} võrra suurem. Oodatav ({meta_key1}- {meta_key2}) NOK-: liiga vähe kirjeid. Miinimumist {meta_key1} - {query1 _result} võrra väiksem. Oodatav ({meta_key1}- {meta_key2})
query_examine1	-
query_examine2	-

Negatiivse tulemusega testi väljundite näidis - tüüp METRIC_COUNT

Näidistesti kirjeldus: iskute tabeli on oodatav ridade arv vahemikus 200-220

METAANDMED			
meta_table1	**person**	**meta_table2**	
meta_column1		**meta_column2**	
meta_condition1		meta_condition2	
meta_key1	200	meta_key2	220
meta_examine_columns1	-	meta_examine_columns2	-

VIGASE TESTI TULEMUSED	
query_result1	172
result_comment	NOK-: liiga vähe kirjeid. Miinimumist 28 võrra väiksem.Oodatav (200-220)

9 Testi tüüp EQUAL_ROWS

Testi tüübi lühikirjeldus: Kahest tabelist kirjete arvu võrdlus. Veateates kuvatakse mõlema tabeli kirjete arv ja ning erinevus. Sellel testil ei ole alampäringuid. Täpsemaks testimiseks kirjete kaupa valida MATCH_HASH või võtmetunnuse testimiseks MATCH_KEY.

METAANDMED			
meta_table1	**Lähtetabel**	**meta_table2**	**Sihttabel**
meta_column1	-	meta_column2	-
meta_condition1	Lähtetabeli filtritingimus	meta_condition2	Sihttabeli filtritingimus
meta_key1		meta_key2	
meta_examine_columns1	-	meta_examine_columns2	-

PÄRINGUTE JA TULEMUSTE GENEREERIMISE MALLID	
query1	SELECT COUNT (*) FROM {meta_table1} WHERE {meta_condition1}
query2	SELECT COUNT (*) FROM {meta_table2} WHERE {meta_condition2}
	Testitud andmed on korrektsed, kui päringute tulemused on võrdsed ja on nullist erinevad
result_comment	ERROR: Päringu tulemused on nullid. NOK: Päring 1: {query1_result} kirjet; Päring 2: {query2_result} kirjet. Erinevus: {query2_result} - {query1_result}.
query_examine1	-
query_examine2	-

Negatiivse tulemusega testi väljundite näidis - tüüp EQUAL_ROWS

Näidistesti kirjeldus: kirjete arvu võrdlus tabelites ISIKUD ja PERSON

METAANDMED			
meta_table1	**isikud**	**meta_table2**	**person**
meta_column1	-	meta_column2	-
meta_condition1		meta_condition2	-
meta_key1		meta_key2	
meta_examine_columns1	-	meta_examine_columns2	-

VIGASE TESTI TULEMUSED	
query_result1	455
query_result2	480
result_comment	Päring 1: 455 kirjet; Päring 2: 480 kirjet. Erinevus:25.

10 Testi tüüp MATCH_KEY

Testi tüübi lühikirjeldus: Kahest tabelist võtmetunnuste võrdlus. Kontrollitakse, millised väärtused on tabelis {meta_table1} rohkem ja millised väärtused on {meta_table2} rohkem.

Näide: Võrreldakse isikute identifikaatorit kahes tabelis ISIK ja PERSON.
Esimese päring tagastab tunnuse ISIK.KOOD väärtuste arvu, mida ei ole tabelis PERSON.
Teine päring tagastab tunnuse PERSON.ID väärtuste arvu, mida ei ole tabelis ISIK.

Esimene abipäring tagastab kirjed, mis on esimeses tabelis, aga teise tabelisse ei jõudnud.

Teine abipäring tagastab kirjed, mis on teises tabelis, aga esimeses tabelis ei olnud.

METAANDMED			
meta_table1	**Lähtetabel**	**meta_table2**	**Sihttabel**
meta_column1	**Lähtetabeli tunnus**	**meta_column2**	**Sihttabeli tunnus**
meta_condition1	Lähtetabeli filtritingimus	meta_condition2	Sihttabeli filtritingimus
meta_key1		meta_key2	
meta_examine_columns1	t1 tunnused	meta_examine_columns2	t2 tunnused
Metaandmete täitmise kommentaar: Abipäringu metatunnustes {meta_examine_columns} peab kasutama tunnuste ees tabelite lühinimesid t1 või t2.			

PÄRINGUTE JA TULEMUSTE GENEREERIMISE MALLID	
query1	SELECT count(*) FROM (select {meta_column1} FROM {meta_table1} WHERE {meta_condition1}) t1 LEFT JOIN (select {meta_column2} FROM {meta_table2} WHERE {meta_condition2}) t2 ON (t1.{meta_column1}= t2.{meta_column2}) WHERE t2.{meta_column2} IS NULL
query2	SELECT count(*) FROM (select {meta_column2} FROM {meta_table2} WHERE {meta_condition2}) t2 LEFT JOIN (select {meta_column1} FROM {meta_table1} WHERE {meta_condition1}) t1 ON (t1.{meta_column1}= t2.{meta_column2}) WHERE t1.{meta_column1} IS NULL
	Testitud andmed on korrektsed, kui mõlema päringu tulemused on nullid.
result_comment	NOK: Esimeses tabelis on {query1_result} tunnuse {meta_column1} väärtust, mida ei ole teises tabelis; Teises tabelis on {query2_result} tunnuse {meta_column2} väärtust, mida ei ole esimeses tabelis.
query_examine1	SELECT {meta_examine_columns1 } FROM (select * FROM {meta_table1} WHERE {meta_condition1}) t1 LEFT JOIN (select {meta_column2} FROM {meta_table2} WHERE {meta_condition2}) t2 ON (t1.{meta_column1}= t2.{meta_column2}) WHERE t2.{meta_column2} IS NULL
query_examine2	SELECT {meta_examine_columns2} FROM (select * FROM {meta_table2} WHERE {meta_condition2}) t2 LEFT JOIN (select {meta_column1} FROM {meta_table1} WHERE {meta_condition1}) t1 ON (t1.{meta_column1}= t2.{meta_column2}) WHERE t1.{meta_column1} IS NULL

Negatiivse tulemusega testi väljundite näidis - tüüp MATCH_KEY

Näidistesti kirjeldus: kontrollitakse võtmetunnuse olemasolu mõlemas tabelis ISIKUD.ISIK_ID ja PERSON.ID

METAANDMED			
meta_table1	**PERSON**	**meta_table2**	ISKUD
meta_column1	**ID**	**meta_column2**	ISIK_ID
meta_condition1		meta_condition2	-
meta_key1		meta_key2	
meta_examine_columns1	t1.id,t1.first_name,t1.last_name, t1.status	meta_examine_columns2	t2.isik_id,t2.perenimi, t2.staatus

VIGASE TESTI TULEMUSED	
query_result1	2
query_result2	1
result_comment	Esimeses tabelis on 2 tunnuse ID väärtust, mida ei ole teises tabelis; Teises tabelis on 1 tunnuse ISIK_ID väärtust, mida ei ole esimeses tabelis.
Esimese abipäringu tulemus	**Esimene abipäring** tagastab kirjed, mis on tabeis PERSON, aga teise tabelisse ei jõudnud.
Teise abipäringu tulemus	**Teine abipäring** tagastab kirjed, mis on teises tabelis ISIKUD, aga esimeses tabelis ei olnud.

	id integer	first_name text	last_name text	status text
1	250	Merle	Sarfat	VALID
2	400	Jessalyn	Arrowsmith	INVALID

	isik_id integer	perenimi text	staatus text
1	506	Rivers	VALID

11 Testi tüüp MATCH_DISTINCT_KEY

Testi tüübi lühikirjeldus: Kahest tabelist tunnuste unikaalsete väärtuste võrdlus.

Esimene abipäring tagastab väärtused, mis esinevad ainult ühe tabelis, aga teises ei esine.

Näited:

- Igal isikul peab olema tabelis DOKUMENDID vähemalt üks dokument.
  Võrreldakse, et (distinct ISIKUD.KOOD) = (distinct DOKUMENDID.ISIK_KOOD)
- Iga tabeli ISIKUD rea kohta peab olema vähemalt üks rida isikute logi tabelis.
  Võrreldakse, et (distinct ISIKUD.ID) = (distinct ISIKUD_LOG.ISIKUD_ID)

METAANDMED			
meta_table1	**Lähtetabel**	**meta_table2**	**Sihttabel**
meta_column1	**Lähtetabeli tunnus**	**meta_column2**	**Sihttabeli tunnus**
meta_condition1	Lähtetabeli filtritingimus	meta_condition2	Sihttabeli filtritingimus
meta_key1		meta_key2	
meta_examine_columns1		meta_examine_columns2	
Metaandmete täitmise kommentaar: ● Vaikimisi on abipäringu väljundis tunnused t1.{meta_key1}, t1.{meta_column1} , t2.{meta_key2}, t2.{meta_column2}. Metaandmetega saab abipäringu väljundisse tunnuseid lisada.			

PÄRINGUTE JA TULEMUSTE GENEREERIMISE MALLID	
query1	SELECT COUNT(*) FROM (SELECT DISTINCT {meta_column1} FROM {meta_table1} WHERE {meta_condition1}) t1 FULL JOIN (SELECT DISTINCT {meta_column2} FROM {meta_table2} WHERE {meta_condition2}) t2 ON t1.{meta_column1}= t2.{meta_column2} WHERE t1.{meta_column1} IS NULL OR t2.{meta_column2} IS NULL
query2	
	Testitud andmed on korrektsed, kui esimese päringu tulemus on 0
result_comment	NOK: Leidub {query1_result} kirjet, mille korral tunnuste t1.{meta_COLUMN1} ja t2.{meta_column2} väärtused ei ole võrdsed.
query_examine1	SELECT t1.{meta_column1}, t2.{meta_column2} FROM (SELECT DISTINCT {meta_column1} FROM {meta_table1} WHERE {meta_condition1}) t1 FULL JOIN (SELECT DISTINCT {meta_column2} FROM {meta_table2} WHERE {meta_condition2}) t2 ON t1.{meta_column1}= t2.{meta_column2} WHERE t1.{meta_column1} IS NULL OR t2.{meta_column2} IS NULL
query_examine2	

Negatiivse tulemusega testi väljundite näidis - tüüp MATCH_DISTINCT_KEY

Näidistesti kirjeldus: kontrollitakse, kas kõikidel PERSON tabeli isikutel on vähemalt üks pangakaart.

METAANDMED			
meta_table1	**PERSON**	**meta_table2**	CARD
meta_column1	**ID**	**meta_column2**	PERSON_ID
meta_condition1		meta_condition2	-
meta_key1		meta_key2	
meta_examine_columns1		meta_examine_columns2	

VIGASE TESTI TULEMUSED	
query_result1	10
result_comment	Leidub 10 kirjet, mille korral tunnuste t1.ID ja t2.PERSON_ID kasutses on erisusi.
Esimese abipäringu tulemus	**Esimene abipäring** tagastab väärtused, mis esinevad ainult ühe tabelis, aga teises ei esine.

	id integer	person_id integer
1	[null]	550
2	497	[null]
3	486	[null]
4	478	[null]
5	458	[null]
6	474	[null]
7	498	[null]
8	456	[null]
9	464	[null]
10	476	[null]

12 Testi tüüp MATCH_HASH

Testi tüübi lühikirjeldus: Kahest tabelist kirjete võrdlus räsi abil.

Esimene abipäring tagastab kirjed, kus võrreldavad tunnuste väärtustes esineb erinevusi.

METAANDMED			
meta_table1	**Lähtetabel**	**meta_table2**	**Sihttabel**
meta_column1	**Lähtetabeli tunnus**	**meta_column2**	**Sihttabeli tunnus**
meta_condition1	Lähtetabeli filtritingimus	meta_condition2	Sihttabeli filtritingimus
meta_key1	**Lähtetabeli seosetunnus**	**meta_key2**	**Sihttabeli seosetunnus**
meta_examine_columns1	t1 tunnused	meta_examine_columns2	t2 tunnused
Metaandmete täitmise kommentaar: Abipäringute metatunnuses {meta_examine_columns} peab kasutama tunnuste ees tabelite lühinimesid t1 või t2.			

PÄRINGUTE JA TULEMUSTE GENEREERIMISE MALLID	
query1	SELECT count(*) FROM (select {meta_key1}, MD5(ROW({meta_column1})::text) AS hash FROM {meta_table1} WHERE {meta_condition1}) t1 FULL JOIN (select {meta_key2}, RMD5(ROW({meta_column2})::text) AS hash FROM {meta_table2} WHERE {meta_condition2}) t2 ON t1.{meta_key1}= t2.{meta_key2} WHERE t1.hash!= t2.hash OR t1.hash IS NULL OR t2.hash IS NULL
query2	-
	Testitud andmed on korrektsed, kui esimese päringu tulemus on 0.
result_comment	NOK: Leidub {query1_result} kirjet, mille korral kirjete kontrolltunnused (räsid) ei ole ühesugused või ühes tabelis kirjed puuduvad. Võrreldud tunnused: tabel {meta_table1} tunnused {meta_column1} ja tabel {meta_table2} tunnused {meta_column1}.
query_examine1	SELECT t1.{meta_key1}, t1.{meta_column1}, {meta_examine_columns1} t2.{meta_key2}, t2.{meta_column2}, {meta_examine_columns2} FROM (select * , MD5(ROW({meta_column1})::text) AS hash FROM {meta_table1} WHERE {meta_condition1}) t1 FULL JOIN (select * , MD5(ROW({meta_column2})::text) AS hash FROM {meta_table2} WHERE {meta_condition2}) t2 ON t1.{meta_key1}= t2.{meta_key2} WHERE t1.hash!= t2.hash
query_examine2	-

Negatiivse tulemusega testi väljundite näidis - tüüp MATCH_HASH

Näidistesti kirjeldus: Isikute kirjete võrdlus räsi abil. Võrdlus tehakse korraga kolmele tunnusele. Võrreldakse järgmisi tunnuseid:

- tabelis PERSON tunnused FIRST_NAME, AGE, VALID_FROM,
- tabelis ISIKUD tunnused EESNIMI, VANUS, ALATES,

METAANDMED			
meta_table1	**PERSON**	**meta_table2**	**ISIKUD**
meta_column1	**first_name, age, valid_from**	**meta_column2**	**eesnimi, vanus, alates**
meta_condition1		meta_condition2	-
meta_key1	**ID**	**meta_key2**	**ISIKUD_ID**
meta_examine_columns1	t1.status as v_staatus	meta_examine_columns2	t2.staatus as u_saatus, t2.perenimi

VIGASE TESTI TULEMUSED	
query_result1	5
result_comment	Leidub 5 kirjet, mille korral kirjete kontrolltunnused (räsid) ei ole ühesugused või ühes tabelis kirjed puuduvad. Võrreldud tunnused: tabel PERSON tunnused 'first_name, age, valid_from' ja tabel ISIKUD tunnused 'eesnimi, vanus, alates'.
Esimese abipäringu tulemus	**Esimene abipäring** tagastab kirjed, kus võrreldavad tunnuste väärtused on erinevad.

	id integer	first_name text	age integer	valid_from date	v_saatus text	isik_id integer	eesnimi text	vanus integer	alates date	u_saatus text	perenimi text
1	72	Britni	21	[null]	VALID	72	Britni	21	2018-02-09	VALID	Bulfit
2	203	Keen	[null]	2019-04-03	INVALID	203	Keen	45	2019-04-03	INVALID	Dunthorn
3	450	Anjanette	21	[null]	CLOSED	450	Anjanette	21	2018-09-13	CLOSED	O'Criane
4	19	Gloria	21	[null]	VALID	19	Gloria	21	2018-02-19	VALID	O'Sirin
5	71	Lynnett	59	2018-03-15	VALID	71	Lynnetti	59	2018-03-15	VALID	Rivers

Joonisele on erinevused märgitud x-ga.

13 Testi tüüp MATCH_COLUMN

Testi tüübi lühikirjeldus: Kahest tabelist tunnuste väärtuste võrdlus. Seos luuakse teiste tunnuste abil.

Esimene abipäring tagastab kirjed, kus võrreldavad tunnuste väärtused erinevad.

Teine abipäring tagastab kirjed, millel väärtused esinevad ainult ühe tabelis, aga teises ei ole.

Näide: Võrreldakse tunnuseid ISIK.NIMI ja PERSON.NAME ja seos luuakse ID abil.

METAANDMED			
meta_table1	**Lähtetabel**	**meta_table2**	**Sihttabel**
meta_column1	**Lähtetabeli tunnus**	**meta_column2**	**Sihttabeli tunnus**
meta_condition1	Lähtetabeli filtritingimus	meta_condition2	Sihttabeli filtritingimus
meta_key1	**Lähtetabeli seosetunnus**	**meta_key2**	**Sihttabeli seosetunnus**
meta_examine_columns1	t1 tunnused	meta_examine_columns2	t2 tunnused

Metaandmete täitmise kommentaar:

- Vaikimisi on abipäringu väljundis tunnused t1.{meta_key1}, t1.{meta_column1} , t2.{meta_key2}, t2.{meta_column2}. Metaandmetega saab väljundisse tunnuseid lisada.
- Abipäringute metatunnustes {meta_examine_columns1},{meta_examine_columns2}, peab kasutama tabelite lühinimesid t1 ja t2.

PÄRINGUTE JA TULEMUSTE GENEREERIMISE MALLID	
query1	SELECT COUNT(*) FROM (SELECT {meta_key1}, {meta_column1} FROM {meta_table1} WHERE {meta_condition1}) t1 FULL JOIN (SELECT {meta_key2}, {meta_column2} FROM {meta_table2} WHERE {meta_condition2}) t2 ON t1.{meta_key1} = t2.{meta_key2} WHERE t1.{meta_column1}!= t2.{meta_column2} OR t1.{meta_column1} IS NULL OR t2.{meta_column2} IS NULL
query2	
	Testitud andmed on korrektsed, kui esimese päringu tulemus on 0
result_comment	NOK: Leidub {query1_result} kirjet, mille korral tunnuste t1.{meta_COLUMN1} ja t2.{meta_column2} väärtused ei ole võrdsed.
query_examine1	SELECT t1.{meta_key1}, t1.{meta_column1} , {meta_examine_columns1}, t2.{meta_key2}, t2.{meta_column2}, {meta_examine_columns2} FROM (SELECT * FROM {meta_table1} WHERE {meta_condition1}) t1 FULL JOIN (SELECT * FROM {meta_table2} WHERE {meta_condition2}) t2 ON t1.{meta_key1} = t2.{meta_key2} WHERE t1.{meta_column1}!= t2.{meta_column2}
query_examine2	SELECT t1.{meta_key1}, t1.{meta_column1} , {meta_examine_columns1}, t2.{meta_key2}, t2.{meta_column2}, {meta_examine_columns2} FROM (SELECT * FROM {meta_table1} WHERE {meta_condition1}) t1 FULL JOIN (SELECT * FROM {meta_table2} WHERE {meta_condition2}) t2 ON t1.{meta_key1}= t2.{meta_key2} WHERE t1.{meta_column1} IS NULL OR t2.{meta_column2} IS NULL

Negatiivse tulemusega testi väljundite näidis - tüüp MATCH_COLUMN

Näidistesti kirjeldus: Kontrollitakse lähte- ja sihttabelis isiku eesnime korrektsust, seos luuakse isiku identifikaatori alusel.

METAANDMED			
meta_table1	**PERSON**	**meta_table2**	**ISIKUD**
meta_column1	**first_name**	**meta_column2**	**eesnimi**
meta_condition1		meta_condition2	-
meta_key1	**ID**	**meta_key2**	**ISIK_ID**
meta_examine_columns1	t1.status as v_staatus	meta_examine_columns2	t2.staatus as u_saatus, t2.perenimi

VIGASE TESTI TULEMUSED	
query_result1	6
result_comment	Leidub 6 kirjet, mille korral tunnuste t1.FIRST_NAME ja t2.EESNIMI väärtused ei ole võrdsed.
Esimese abipäringu tulemus	**Esimene abipäring** tagastab kirjed, kus võrreldavad tunnuste väärtused erinevad.
Teise abipäringu tulemus	**Teine abipäring tagastab kirjed,** millel väärtused esinevad ainult ühe tabelis, aga teises ei ole.

	id integer	first_name text	status text	last_name text	isik_id integer	eesnimi text	staatus text	perenimi text
1	71	Lynnett	VALID	Rivers	71	Lynnetti	VALID	Rivers

	id integer	first_name text	status text	last_name text	isik_id integer	eesnimi text	staatus text	perenimi text
1	250	Merle	VALID	Sarfat	[null]	[null]	[null]	[null]
2	400	Jessalyn	INVALID	Arrowsmith	[null]	[null]	[null]	[null]
3	[null]	[null]	[null]	[null]	506	Lynnetti	VALID	Rivers
4	[null]	[null]	[null]	[null]	35	Foss	CLOSED	Yatman
5	[null]	[null]	[null]	[null]	35	Rhys	VALID	Landon

14 Testi tüüp MATCH COUNT

Testi tüübi lühikirjeldus: Objektiga seotud kirjete arvu võrdlus.

Esimene abipäring tagastab summeeritud väljavõtte kus on siduvad tunnused {meta_key} ja mõlema tabeli pealt loendamise tulemus.

Teine abipäring tagastab kirjed, kõik siduva tunnuse {meta_key} kirjed. Väljundist on näha, millised read on puudu.

Näide: Kui isikuga seotud dokumendid on tabelis DOKUMENDID, siis võetakse grupeerivaks tunnuseks ISIK_ID ja võrreldakse isikuga seotud dokumentide kirjete arvu.

METAANDMED			
meta_table1	**Lähtetabel**	**meta_table2**	**Sihttabel**
meta_column1	Lähtetabeli tunnus	meta_column2	Sihttabeli tunnus
meta_condition1	Lähtetabeli filtritingimus	meta_condition2	Sihttabeli filtritingimus
meta_key1	**Lähtetabeli seosetunnus**	**meta_key2**	**Sihttabeli seosetunnus**
meta_examine_columns1	t1 tunnused	meta_examine_columns2	t2 tunnused
Metaandmete täitmise kommentaar: ● Kui tunnused {meta_column1} ja {meta_column2} on täitmata, siis võrreldakse kirjete arvu. ● Vaikimisi on teise abipäringu väljundis tunnused t1.{meta_key1}, t1.{meta_column1} , t2.{meta_key2}, t2.{meta_column2}. Metaandmetega saab väljundisse tunnuseid lisada. ● Abipäringute metatunnustes {meta_examine_columns1},{meta_examine_columns2}, peab kasutama tabelite lühinimesid t1 ja t2.			

PÄRINGUTE JA TULEMUSTE GENEREERIMISE MALLID	
query1	SELECT COUNT(*) FROM (SELECT {meta_key1}, COUNT({meta_column1}) AS count1 FROM {meta_table1} WHERE {meta_condition1} GROUP BY {meta_key1}) t1 FULL JOIN (SELECT {meta_key2}, COUNT({meta_column2}) AS count2 FROM {meta_table2} WHERE {meta_condition2} GROUP BY {meta_key2}) t2 ON t1.{meta_key1}= t2.{meta_key2} WHERE t1.count1!= t2.count2 OR t1.count1 IS NULL OR t2.count2 IS NULL
query2	
	Testitud andmed on korrektsed, kui esimese päringu tulemus on 0
result_comment	NOK: Leidub {query1_result} tunnuse {meta_key1} väärtust, millega seotud kirjete arvud ei ole tabelites võrdsed st. COUNT (t1.{meta_column1}) != COUNT(t2.{meta_column2}).
query_examine1	SELECT t1.{meta_key1}, t1.count1, t2.{meta_key2}, t2.count2 FROM (SELECT {meta_key1}, COUNT({meta_column1}) AS count1 FROM {meta_table1} WHERE {meta_condition1} GROUP BY {meta_key1}) t1 FULL JOIN (SELECT {meta_key2}, COUNT({meta_column2}) AS count2 FROM {meta_table2} WHERE {meta_condition2} GROUP BY {meta_key2}) t2

	ON t1.{meta_key1}= t2.{meta_key2} WHERE t1.count1!= t2.count2 OR t1.count1 IS NULL OR t2.count2 IS NULL
query_examine2	**NB! Seda päringut saab koostada ainult siis, kui on täidetud ka võrdlevad tunnused {meta_column1} ja {meta_column2} .** WITH errors AS (SELECT COALESCE(t1.{meta_key1}, t2.{meta_key2}) FROM (SELECT {meta_key1}, COUNT({meta_column1}) AS count1 FROM {meta_table1} WHERE {meta_condition1} GROUP BY {meta_key1}) t1 FULL JOIN (SELECT {meta_key2}, COUNT({meta_column2}) AS count2 FROM {meta_table2} WHERE {meta_condition2} GROUP BY {meta_key2}) t2 ON t1.{meta_key1}= t2.{meta_key2} WHERE t1.count1!= t2.count2 OR t1.count1 IS NULL OR t2.count2 IS NULL) SELECT t1.{meta_key1}, t1.{meta_column1}, {meta_examine_columns1} t2.{meta_key2}, t2.{meta_column2}, {meta_examine_columns2} FROM (SELECT * FROM {meta_table1} WHERE {meta_condition1}) t1 FULL JOIN (SELECT * FROM {meta_table2} WHERE {meta_condition2}) t2 ON (t1.{meta_key1} = t2.{meta_key2} AND t1.{meta_column1} = t2.{meta_column2}) WHERE t1.{meta_key1} IN (SELECT * FROM errors) OR t2.{meta_key2} IN (SELECT * FROM errors) ORDER BY t1.{meta_key1}, t1.{meta_column1}, t2.{meta_key2}, t2.{meta_column2}

Negatiivse tulemusega testi väljundite näidis - MATCH_COUNT

Näidistesti kirjeldus: Isikuga seotud kaartide arvu võrdlus.

METAANDMED			
meta_table1	**CARD**	**meta_table2**	**KAART**
meta_column1		**meta_column2**	
meta_condition1		meta_condition2	-
meta_key1	**PERSON_ID**	**meta_key2**	**ISIK_ID**
meta_examine_columns1	t1.card_id, t1.card_nr ;	meta_examine_columns2	t2.kaart_id, t2.kaart_nr

VIGASE TESTI TULEMUSED	
query_result1	6
result_comment	Leidub 6 tunnuse person_id väärtust, millega seotud kirjete arvud ei ole tabelites võrdsed.
Esimese abipäringu tulemus	**Esimene abipäring** tagastab isikud, kellel krediitkaartide arv oli lähte- ja sihtbaasis erinev.

	person_id integer	count1 bigint	isik_id integer	count2 bigint
1	550	1	550	2
2	114	1	[null]	[null]
3	171	1	171	2
4	22	2	22	1
5	124	2	124	1
6	492	2	492	1

Teise abipäringu tulemus

Teine abipäring tagastab kõik esimese abipäringu isikute krediitkaartide read.

	person_id integer	card_limit integer	id integer	card_nr text	isik_id integer	limiit integer	kaart_id integer	kaart_nr text
1	22	800	143	3562086924366340	22	800	143	3562086924366340
2	22	600	178	3552927637058462	[null]	[null]	[null]	[null]
3	114	800	44	4508659455462749	[null]	[null]	[null]	[null]
4	124	800	478	6759871051224041818	124	800	478	6759871051224041818
5	124	700	457	502028818437424664	[null]	[null]	[null]	[null]
6	171	900	30	3531231082156970	171	900	30	3531231082156970

jne

15 Testi tüüp MATCH_SUM

Testi tüübi lühikirjeldus: Objektiga seotud kirjete arvu võrdlus.

Esimene abipäring tagastab summeeritud väljavõtte kus on siduvad tunnused {meta_key} ja mõlema tabelis leitud summad : SUM({meta_column1}) ja SUM({meta_column2}).

METAANDMED			
meta_table1	**Lähtetabel**	**meta_table2**	**Sihttabel**
meta_column1	**Lähtetabeli tunnus**	**meta_column2**	**Sihttabeli tunnus**
meta_condition1	Lähtetabeli filtritingimus	meta_condition2	Sihttabeli filtritingimus
meta_key1	**Lähtetabeli seosetunnus**	**meta_key2**	**Sihttabeli seosetunnus**
meta_examine_columns1	-	meta_examine_columns2	-
Metaandmete täitmise kommentaar: ● Summeeritavad tunnused {meta_column1} ja {meta_column2} need peavad olema arvuliste väärtustega.			

PÄRINGUTE JA TULEMUSTE GENEREERIMISE MALLID	
query1	SELECT COUNT(*) FROM (SELECT {meta_key1}, SUM({meta_column1}) AS sum1 FROM {meta_table1} WHERE {meta_condition1} GROUP BY {meta_key1}) t1 FULL JOIN (SELECT {meta_key2}, SUM({meta_column2}) AS sum2 FROM {meta_table2} WHERE {meta_condition2} GROUP BY {meta_key2}) t2 ON t1.{meta_key1} = t2.{meta_key2} WHERE t1.sum1!= t2.sum2 OR t1.sum1 IS NULL OR t2.sum2 IS NULL
query2	
	Testitud andmed on korrektsed, kui esimese päringu tulemus on 0
result_comment	NOK: Leidub {query1_result} tunnuse {meta_key1} väärtust, millega seotud kirjete tunnusete summad ei ole võrdsed st. SUM(t1.{meta_column1}) != SUM(t2.{meta_column2}).
query_examine1	SELECT t1.{meta_key1}, t1.sum1, t2.{meta_key2}, t2.sum2 FROM (SELECT {meta_key1}, SUM({meta_column1}) AS sum1 FROM {meta_table1} WHERE {meta_condition1} GROUP BY {meta_key1}) t1 FULL JOIN (SELECT {meta_key2}, SUM({meta_column2}) AS sum2 FROM {meta_table2} WHERE {meta_condition2} GROUP BY {meta_key2}) t2 ON t1.{meta_key1}= t2.{meta_key2} WHERE t1.sum1!= t2.sum2 OR t1.sum1 IS NULL OR t2.sum2 IS NULL
query_examine2	

Negatiivse tulemusega testi väljundite näidis - tüüp MATCH_SUM

Näidistesti kirjeldus: Võrreldakse lähte ja sihttabelites isikuga seotud krediitkaartide limiitide summat isikute lõikes.

METAANDMED			
meta_table1	**CARD**	**meta_table2**	**KAART**
meta_column1	**card_limit**	**meta_column2**	**limiit**
meta_condition1		meta_condition2	-
meta_key1	**PERSON_ID**	**meta_key2**	**ISIK_ID**
meta_examine_columns1	-	meta_examine_columns2	-

VIGASE TESTI TULEMUSED	
query_result1	2
result_comment	Leidub 2 tunnuse PERSON_ID väärtust, millega seotud kirjete tunnusete summad ei ole võrdsed st. SUM(t1.card_limit) != SUM(t2.limiit).
Esimese abipäringu tulemus	**Esimene abipäring** tagastab summeeritud väljavõtte isikute kohta, kellel lähte- ja sihtbaasis pangakaartide limiitide summad erinesid.

	person_id integer	sum1 bigint	isik_id integer	sum2 bigint
1	550	900	550	1800
2	114	800	[null]	[null]

16 Testi tüüp INFORMAL

Kui testide genereerimine ja testide kasutamine on ühtses süsteemis, siis on vaja eraldi testi tüüpi ka käsitsi kirjutatud päringute jaoks. Eelkõige on seda vaja kasutaja poolt eeldefineeritud **veateate** genereerimiseks.

METAANDMED			
meta_table1	-	meta_table2	-
meta_column1	-	meta_column2	-
meta_condition1	-	meta_condition2	-
meta_key1	-	meta_key2	-
meta_examine_columns1	veateate mall	meta_examine_columns2	

Metaandmete täitmise kommentaar:

- Kuna testide päringuid ei genereerita, siis päringute jaoks ei ole metaandmeid vaja.
- Kasutaja defineerib veateatemalli tunnuse {meta_examine} abil. Mallis võib kasutada muutujatena testi tulemusi {result_query1}, {result_query2}.

PÄRINGUTE JA TULEMUSTE GENEREERIMISE MALLID	
query1	**Kasutaja sisestatud päring**
query2	Kasutaja sisestatud päring
	• Kui testil on ainult esimene päring ja selle tulemus on 0, siis on testitud andmed korrektsed. • Kui testil on kaks päringut ja nende tulemused on võrdsed, siis on testitud andmed korrektsed.
result_comment	NOK: {meta_examine_columns1}
query_examine1	**Kasutaja sisestatud abipäring**
query_examine2	Kasutaja sisestatud teine abipäring

III Testi tüüpide kaupa metaandmete kasutamise kokkuvõte

Testi tüübist sõltub metaandmete täitmise vajadus, kohustuslikkus ja kontekst. Järgnevas tabelis on kokkuvõte testi tüüpide kaupa metatunnuste kasutamisest.

- Oranži taustaga metatunnused on kohustuslikud,
- kollase taustaga metatunnuseid saab kasutada kasutada vajadusel.

	Testi tüüp	table1	column1	condition1	key1	examine_ columns1	table2	column2	condition 2	key2	examine_ columns2
1	NOT_NULL	tabel	tunnus	tingimus		tunnused					
2	IS_NULL	tabel	tunnus	tingimus		tunnused					
3	STATIC_VALUE	tabel	tunnus	tingimus	väärtus	tunnused					
4	IN_VALUES	tabel	tunnus	tingimus	väärtused	tunnused					
5	UNIQUE_VALUE	tabel	tunnus	tingimus		tunnused					
6	CODE	tabel	tunnus	tingimus		tunnused	KL tabel	KL tunnus	KL tingimus		
7	VALUES_ BETWEEN	tabel	tunnus	tingimus	min	tunnused				max	
8	METRIC_COUNT	tabel		tingimus	min	tunnused				max	
9	EQUAL_ROWS	lähtetabel		tingimus			sihttabel		tingimus		
10	MATCH_KEY	lähtetabel	tunnus	tingimus		t1 tunnused	sihttabel	tunnus	tingimus		t2 tunnused
11	MATCH_DISTINCT_ KEY	lähtetabel	tunnus	tingimus			sihttabel	tunnus	tingimus		
12	MATCH_HASH	lähtetabel	tunnuse	tingimus	seose 1.tunnus	t1 tunnused	sihttabel	tunnused	tingimus	seose 2. tunnus	t2 tunnused
13	MATCH_COLUMN	lähtetabel	tunnus	tingimus	seose 1.tunnus	t1 tunnused	sihttabel	tunnus	tingimus	seose 2. tunnus	t2 tunnused
14	MATCH_COUNT	lähtetabel	tunnus	tingimus	seose 1.tunnus	t1 tunnused	sihttabel	tunnus	tingimus	seose 2. tunnus	t2 tunnused
15	MATCH_SUM	lähtetabel	tunnus	tingimus	seose 1.tunnus		sihttabel	tunnus	tingimus	seose 2. tunnus	
16	INFORMAL					veateate mall					veateate mall

IV Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, **Sirje Lind**,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose

'Andmete migratsiooni testimine',

mille juhendaja on **Anne Villems**

reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni;

2. annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni;

3. olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4. kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Sirje Lind

16.05.2019