TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Informaatikainstituut

IDK40LT

Julia Poljuhovitš 123725IABB

INVESTLY FAKTOORINGUTARKVARA ANDMEBAASISKEEMI LOOMINE JA ANDMETE MIGREERIMINE PÄRANDSÜSTEEMIST

Bakalaureusetöö

Juhendaja: Martin Rebane
MSc
Lektor

Tallinn 2017

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Julia Poljuhovitš

17.11.2016

Annotatsioon

Käesolevas bakalaureusetöös analüüsitakse Investly Holding OÜ andmemudelit ning luuakse uus andmebaasiskeem, mille põhjal leitakse parim lahendus, kuidas andmeid migreerida olevast andmebaasist ja mitte kajastada andmete sisu.

Töö eesmärgiks on analüüsida andmebaasiskeemi ja kirjutada päringuid, kasutades SQL programmeerimiskeel, mis võimaldavad andmeid migreerida vanast andmebaasist. Uus andmebaasiskeem peab olema normaliseeritud ning andmed peavad vastama andmemudeli struktuurile.

Tulemuseks on esitatud valmis andmebaasiskeem, SQL kood ning andmed, mis on võetud vanast andmebaasist ja vastavad uuele andmemudeli struktuurile.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 25 leheküljel, 6 peatükki, 30 joonist.

Abstract

Implementation of database schema for Investly factoring software, and data migration from legacy system

In this thesis is analyzed the OÜ Investly Holding data model and created a new database schema, which is considered for finding the best solution for how to migrate data from the database and not to reflect the content of the data.

The main goal is to analyze database schema and write queries using the SQL programming language, allowing data to migrate from the old database. The new database schema must be normalized must correspond the data model structure.

The result is presented as a ready database schema, SQL code and data, which are taken from the old database and are corresponded to the new structure.

The thesis is in estonian and contains 25 pages of text, 6 chapters, 30 figures.

Lühendite ja mõistete sõnastik

SQL	Structrured Query Language
IT	Information Technology
IBM	International Business Machines
PRIMARY KEY	Primaarvõti, mis identifitseerib ühe kirje
FOREIGN KEY	Välisvõti seose loomiseks

Sisukord

Jooniste loetelu

1 Sissejuhatus

Lõputöö teemaks on valitud „Investly faktooringutarkvara andmebaasiskeemi loomine ja andmete migreerimine pärandsüsteemist“. Töö on tehtud Investly Holding OÜ firma jaoks, mis tegeleb faktooringuga. Investly andmebaasiskeem vajab uuendamist ja normaliseerimist, et töötada suure andmete mahuga ning suurema kiirusega päringuid käivitada.

Praegune andmemudel ei ole normaliseeritud. See tähendab, et tabelites on korduvaid väärtusi, palju tühjaid kirjeid ning käesoleva töö eesmärgiks on vabaneda nendest vigadest ning luua uus andmemudel vanast andmebaasist andmeid migreerides.

Esimeseks lõputöö eesmärgiks on luua uue andmebaasiskeemi prototüüpi, kus on tõsised vead välistatud. Teine eesmärk, mille seab autor lõputöös, on andmete migreerimine vanast andmebaasist. Lahendusena kirjutatakse SQL päringukeeles koodi, kasutades tekstiredaktori Notepad++ ja andmebaasi uurimiseks MariaDB programmi.

Käesolev lõputöö koosneb neljast osast. Esimeses osas kirjutatakse väike ülevaade firmast, faktooringust. Samuti mainitakse põhilised probleemid ja eesmärgid, mida peab saavutama ning seletatakse, mis on normaliseerimine ja andmete migreerimine. Teises osas analüüsitakse vana andmebaasiskeemi ja tuuakse välja vead, mis ei anna head töövõimsust.

Kolmandas osas toimub uue andmebaasiskeemi prototüübi loomine IBM Rational Rose programmi abil, mida on lihtne ja mugav kasutada. Selleks, et alustada, tuleb analüüsida vana andmebaasi skeemi ning leida ainult olulised tabelid, väärtused ja andmed. Selles osas autor parandab vanas andmebaasiskeemis leitud vigu uue skeemi prototüübis.

Neljandas osas, kui uus andmebaasiskeem saab valmis, analüüsitakse skeemi ning tuuakse välja andmed, mis tuleb migreerida vanast andmebaasist. See osa on kõige tähtsam, kuna selles toimub põhilise probleemi lahendamine.

Oodatava tulemusena on valmis SQL kood, mille abil luuakse tabelid, seosed uue skeemi põhjal, kus andmed on võetud vanast andmebaasist ning tõsised vead on parandatud. Selline tehniline muudatus aitab Investlyl teenindada suuremat hulka kliente ja vähendavad IT infrastruktuuri ülalpidamise kulusid tänu paremale tehnilisele realisatsioonile märgatavalt.

2 Taust ja probleem

2.1 Investly Holding OÜ

Investly on faktooringu firma, mis aitab ettevõtetel pika maksetähtajaga arvete eest raha kohe kätte saada ning oma äri edasi arendada. Faktooring ehk arvete müük sobib väga hästi investoritele, kuna nendel on võimalus lühiajaliseks perioodiks raha investeerida, sest arvete maksetähtaeg on kuni 120 päeva. Investly ühisrahastusplatvorm on väga lihtne ja turvaline nii arvete müüjatele kui ka investoritele. See annab väga hea võimaluse ettevõtjatele kiiresti raha kätte saada ning samuti võib raha investeerida väikeseks perioodiks. [1]

2.1.1 Faktooring

Faktooring on tegevus, mille käigus faktooringu firma võtab endale ettevõtete müügiarved ja maksab teenustasu eest arve täissumma ning jälgib ise makse laekumist tähtajaks. [2]

Investly faktooringuteenuse eripäraks on asjaolu, et erinevalt pankadest ei nõuta kõigi ostja ja müüja vaheliste arvete puhul faktooringu kasutamist, vaid müüja saab ise valida, kas ja milliste arvete jaoks ta teenust kasutab.

2.2 Probleem ja eesmärk

Praeguse andmebaasi probleemiks on see, et tabelid ei ole normaliseeritud, mis tähendab seda, et tabelites on korduvaid andmeelemente. Seetõttu päringute käivitamine töötab aeglasemini ning andmemaht on suurem. Andmebaasis on väga palju dubleeritud andmeid, mis raskendab igapäevast tööd ning selle tõttu suureneb andmevigade arv.

Põhiline töö eesmärk on loobuda korduvatest ridadest ja veergudest ning koostada uus skeem, mis on täpne ja korrektne ning igas tabelis on kindlasti üks *primary key*, mille andmetüüp on täisarv. Andmed on võetud ainult need, mida on vaja firmale ja ei kordu. Järgmine samm on siis migreerida andmeid vanast andmebaasist uude vastavalt andmemudeli struktuurile.

2.3 Andmete migreerimine

Andmete migreerimine on selline protsess, mille käigus toimub andmete edastamine andmesalvestuse tüüpide vahel või arvutisüsteemide vahel. Migreerimise põhjuseks võib olla serveri või varustuse vahetus, tehniline hooldus või uuendamine, rakenduse migratsioon ning tarkvara ümberpaigutamine. [3]

Migreerimine võib olla erinevat tüüpi ning tavaliselt kasutatakse selle jaoks spetsiaalse programmi, kus migreerimine toimub automaatselt. [4]

Autori lõputöö jaoks on kasutatud skeemidevahelist andmete migreerimist. Skeemi migreerimine andmebaasis toimub juhul, kui on vaja andmebaasi skeemi uuendada mingi uuema versioonini või taastada vanemat. [4]

Migreerimine tavaliselt toimub skeemi migtratsiooni vahendi abil. Enamus vahendeid on suunatud, et vähendada andmebaasis olevatele andmetele muudatuste mõju. Kahjuks see ei anna garantiid, et kõik andmed säilivad, näiteks kustutades tabelis mingit veergu, selle veeru andmed samuti kaovad ära. [4]

Käesolevas töös autor ei kasuta migreerimise jaoks spetsiaalset vahendit, vaid kirjutab SQL päringukeeles koodi, mis võtab andmed vanast skeemist ning loob uued tabelid ja vajalikud seosed.

Kõige keerulisem on analüüsida vana skeemi, et leida olulised andmed ja tabelid, mis on vaja võtta uue skeemi loomisel.

2.4 Normaliseerimine

Normaliseerimine on andmete organisatsiooni protsess andmebaasis, mille käigus toimub tabelite loomine ja nende vahele seoste paigutamine vastavalt reeglitele. Reeglid aitavad andmeid kaitsta ja liiasusi vältida. [5]

Edukaks andmebaasi normaliseerimiseks piisab kolme esimese reegli täitmisest, teiste sõnadega öeldes peab andmebaasi esitama kuni 3.normaalkujuni. Allpool on toodud kolm põhilist normaliseerimise reeglit. [5] [6]

Esimene normaalkuju:

- Eemaaldada korduvad grupid tabelites
- Luua eraldi tabel igale grupile seotud andmetega
- Luua igale tabelile identifikaator primaarvõtme (*Primary Key*) abil

Teine normaalkuju:

- Luua eraldi tabelid väärtuste kogumi jaoks, mis seovad mitu kirjet
- Seostada tabelid välisvõtmega (*Foreign Key*)

Kolmas normaalkuju:

- Eemaldada väljad, mis ei sõltu võtmest

[5] [6]

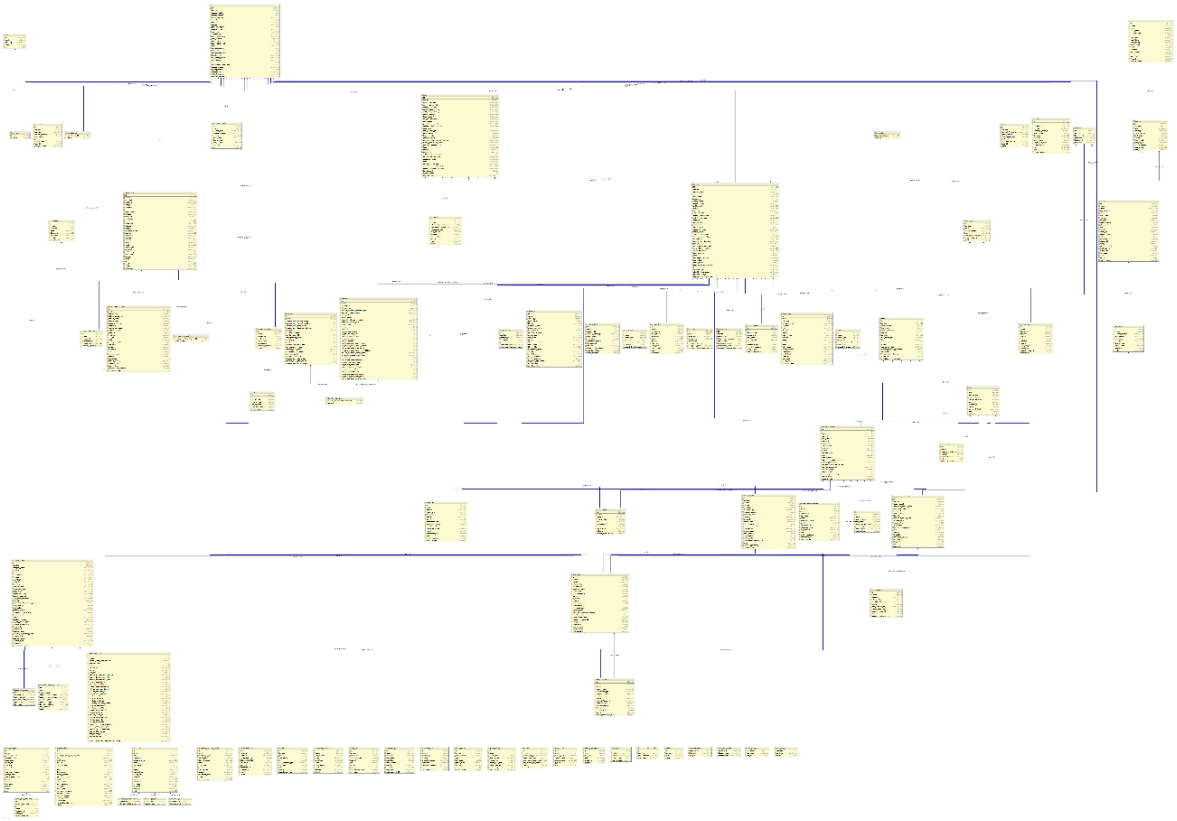
3 Vana andmebaasiskeemi analüüs

Selles peatükis autor kirjutab vana andmebaasiskeemi analüüsi, leiab tõsiseid vigu, mida tuleb parandada.

3.1 Esialgne andmebaasiskeem

Selles alampeatükis on esitatud esialgne andmebaasiskeem, mis on Investlyl kasutamisel.

Joonisel 1 on praegune Investly andmebaasiskeem. Siin ei ole mitte midagi näha, kuna skeem on päris suur, aga on näha, et tabeleid on palju. Loetavat skeemi saab vaadata Lisas – 1. Käesolevas töös on eesmärgiks teha skeem rohkem kompaktseks ja arusaadavamaks.



Joonis 1. Investly andmebaasiskeem

3.2 Tõsised vead vanas andmebaasiskeemis

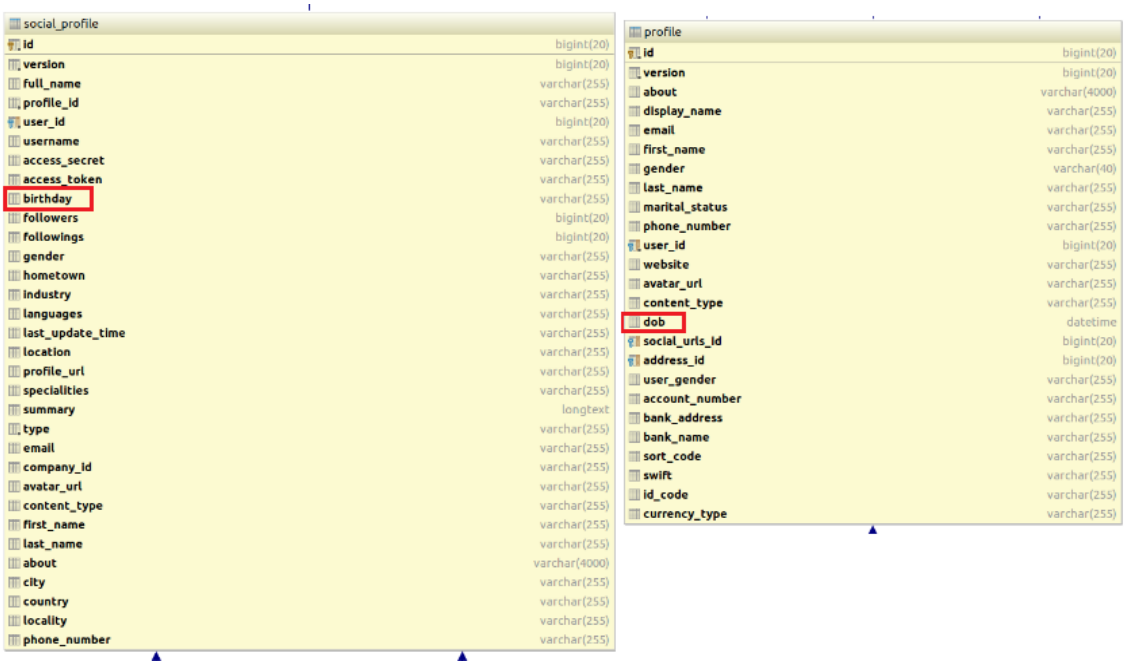
Käesolevas alampeatükis toob autor välja põhilised vead, mille tõttu andmebaasi töö on aeglasem.

Skeem sisaldab väljade hulka, millel ei ole parameeter NOT NULL pandud. See tähendab, et väli võib olla tühi. Seetõttu tabelites on päris palju tühje kirjeid ja veergusid, mis üldse ei sisalda andmeid. Joonisel 2 on näiteks tabel *profile*, kus enamus väljadest lubab NULL väärtuse.

```
MariaDB [investly_old]> explain profile;
+----------------+---------------+------+-----+---------+----------------+
| Field          | Type          | Null | Key | Default | Extra          |
+----------------+---------------+------+-----+---------+----------------+
| id             | bigint(20)    | NO   | PRI | NULL    | auto_increment |
| version        | bigint(20)    | NO   |     | NULL    |                |
| about          | varchar(4000) | YES  |     | NULL    |                |
| display_name   | varchar(255)  | YES  |     | NULL    |                |
| email          | varchar(255)  | YES  |     | NULL    |                |
| first_name     | varchar(255)  | YES  |     | NULL    |                |
| gender         | varchar(40)   | YES  |     | NULL    |                |
| last_name      | varchar(255)  | YES  |     | NULL    |                |
| marital_status | varchar(255)  | YES  |     | NULL    |                |
| phone_number   | varchar(255)  | YES  |     | NULL    |                |
| user_id        | bigint(20)    | NO   | UNI | NULL    |                |
| website        | varchar(255)  | YES  |     | NULL    |                |
| avatar_url     | varchar(255)  | YES  |     | NULL    |                |
| content_type   | varchar(255)  | YES  |     | NULL    |                |
| dob            | datetime      | YES  |     | NULL    |                |
| social_urls_id | bigint(20)    | YES  | MUL | NULL    |                |
| address_id     | bigint(20)    | YES  | MUL | NULL    |                |
| user_gender    | varchar(255)  | YES  |     | NULL    |                |
| account_number | varchar(255)  | YES  |     | NULL    |                |
| bank_address   | varchar(255)  | YES  |     | NULL    |                |
| bank_name      | varchar(255)  | YES  |     | NULL    |                |
| sort_code      | varchar(255)  | YES  |     | NULL    |                |
| swift          | varchar(255)  | YES  |     | NULL    |                |
| id_code        | varchar(255)  | YES  |     | NULL    |                |
| currency_type  | varchar(255)  | YES  |     | NULL    |                |
+----------------+---------------+------+-----+---------+----------------+
```

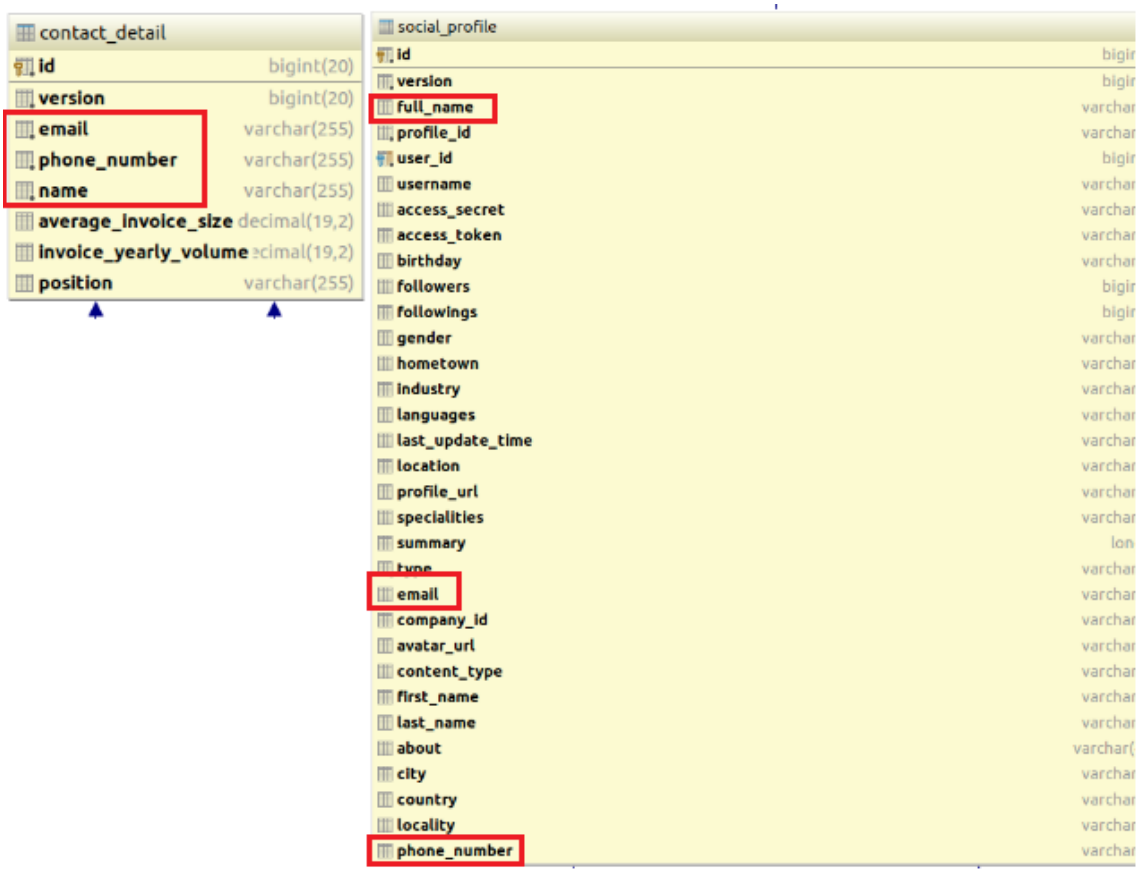
Joonis 2. Tabel *profile*, kus on paljudel veergudel NULL väärtus lubatud

Skeemis on palju korduvaid ridu. Näiteks kasutaja sünnikuupäev kordub tabelites *profile* ja *social_profile*. Mõlemad tabelid on seotud tabeliga *user* välisvõtme kaudu. Uue skeemi loomisel peab vältima selliseid juhtumeid, kus kirjete või väljade andmed korduvad, kuna sellisel juhul anmdebaas töötab aeglasemini. Näide on toodud Joonisel 3.



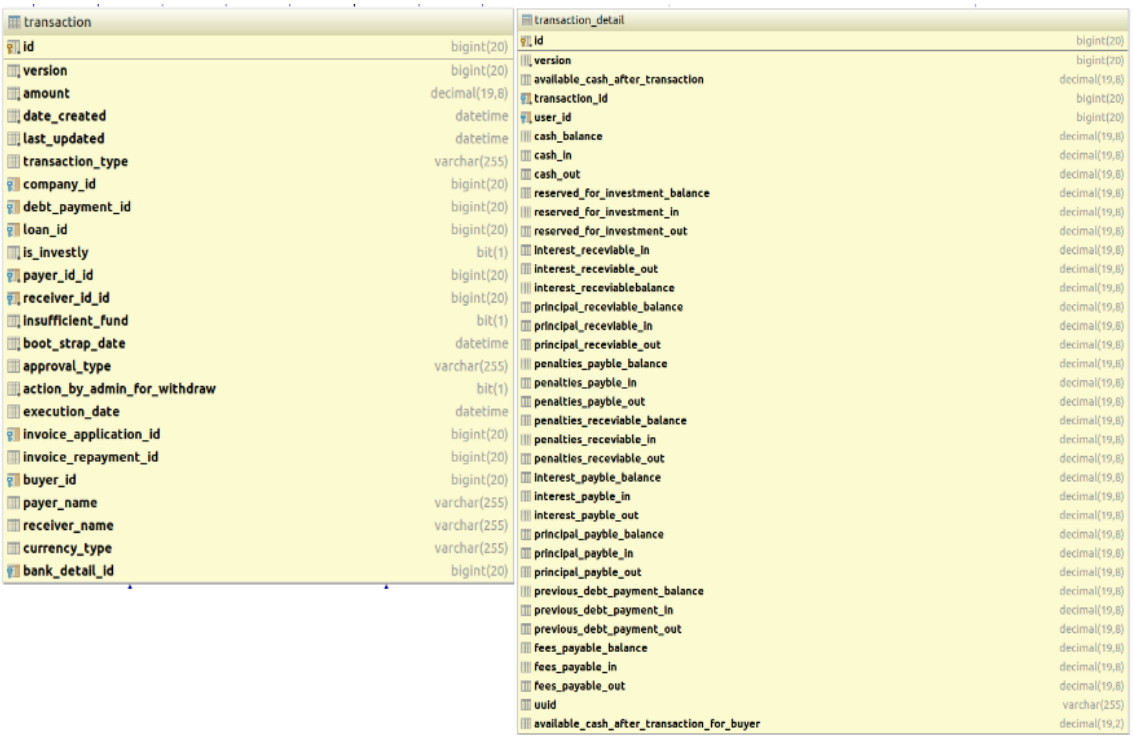
Joonis 3. Tabelid *social_profile* ja *profile* ning mõlemas on sünnikuupäeva väärtus

Samuti tabelis *contact_detail* on sellised veerud nagu nimi, telefoninumber, email ning tabel on seotud tabelitega *seller* ja *buyer*, mis on seotud tabeliga *user*. Skeemis on tabel *social_profile*, kus on seos tabeliga *user* ning kõik need andmed on olemas. Järelikult peab looma eraldi tabeli kõikide kasutajate kontaktidega, et samad andmed ei kordu erinevates tabelites. Joonisel 4 on korduvad kontaktandmed.



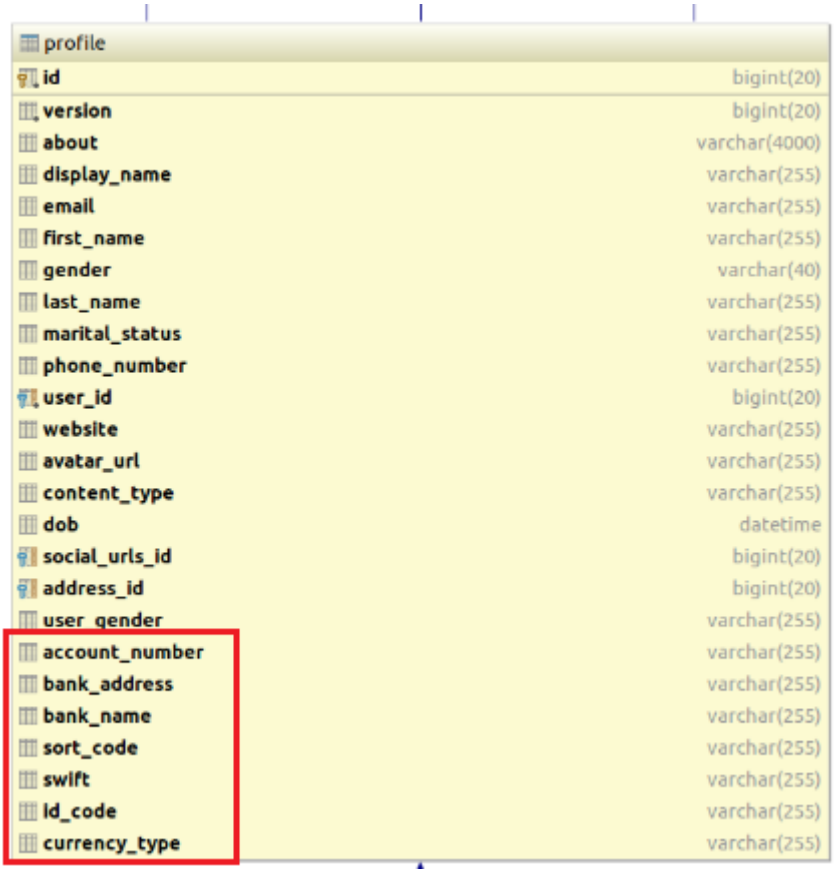
Joonis 4. Kontaktandmed tabelites *contact_detail* ja *social_profile*

Kõige olulisem tabel on skeemis *transaction*. Seal toimuvad igasugused rahalised tehingud. Alguses on päris raske aru saada, mis tehing on tehtud, kas „deebet“ või „kreedit“. Selle jaoks on skeemis tabel *transaction_detail*, kus on väga palju andmeid ning samuti on näha, milline tehing on tehtud. See ei ole väga mugav kasutajale, sellepärast tuleb luua üks tabel kõikide tehingutega, kus on korrektselt näha, kas on see „deebet“ või „kreedit“, mis on staatus, näiteks põhimaks või intress. Praeguses tabelis *transaction* üks kirje näitab mõlemat suunda, mis samuti ei ole kasutajale mugav.



Joonis 5. Tabelid *transaction* ja *transaction_detail*

Tabel *profile* samuti sisaldab panga andmed, nagu *bank_name*, *bank_address*, *swift* jne. Mõistlikum oleks panga andmeid säilitada eraldi tabelis, näiteks nimega *bank_account*, sest kui andmebaasis on näiteks 10 panka ja 1000 kasutajat, siis tabelis *profile* panga andmed korduvad 100 korda, mis jälle raskendab süsteemi tööd.

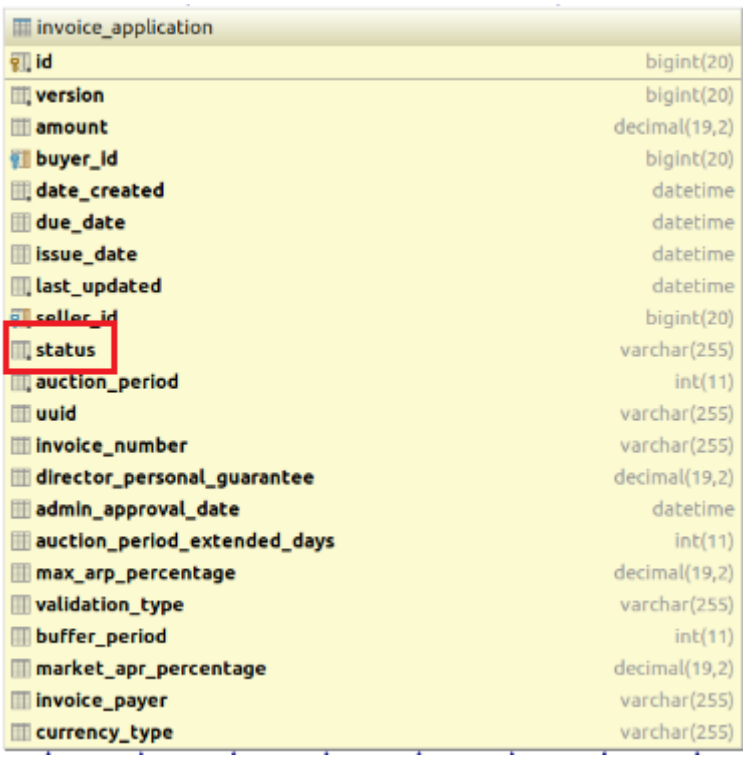


Joonis 6. Tabel *profile* panga andmetega

Veel üks viga andmebaasiskeemis on see, et skeemis puuduvad klassifikaatorite tabelid. Näiteks, tabelis *invoice_application* staatuse väärtus peab olema klassifitseeritud eraldi ja igal staatusel peab olema unikaalne kood. Sellisel juhul tabelis *invoice_application* oleks antud ainult staatuse *id*.

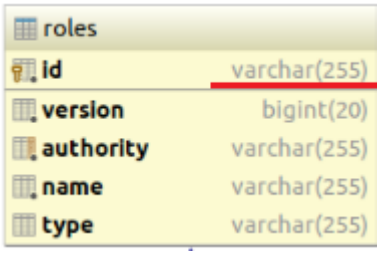
```
MariaDB [investly_old]> select distinct status from invoice_application;
+---------------+
| status        |
+---------------+
| ON_BOARDING   |
| REPAID        |
| SUBMITTED     |
| INCOMPLETE    |
| REJECTED      |
| IN_REPAYMENT  |
| AUCTION       |
+---------------+
```

Joonis 7. Tabelis *invoice_application* olevad staatused



Joonis 8. Tabel *invoice_application*

Tabelis *roles* väljal *id*, mis on Primary Key, on väärtus VARCHAR, mis ei ole õige, sest igas tabelis unikaalne identifikaator Primary key peab olema numbri ja väärtusega BIGINT.

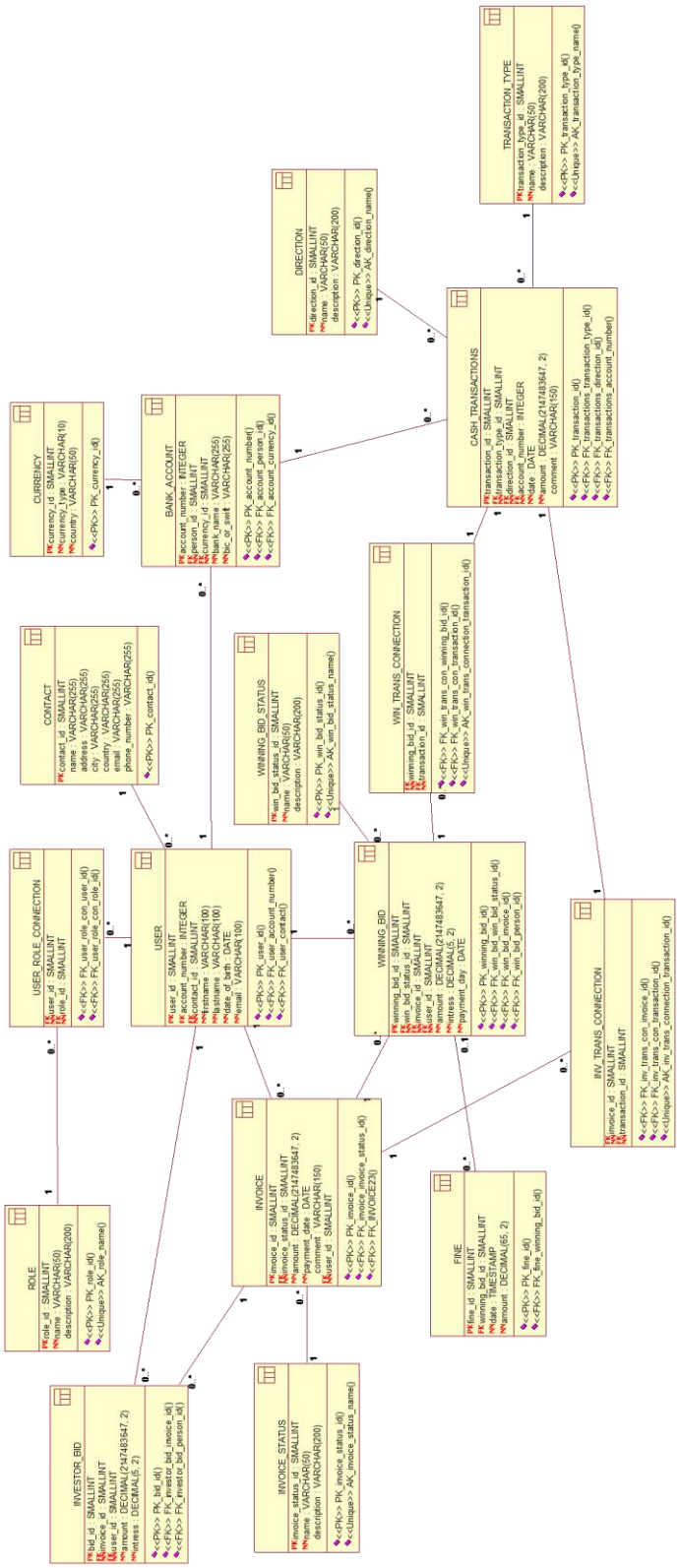


Joonis 9. Tabelis *roles* ebaõige identifikaatori väärtus

4 Uue skeemi loomine

4.1 Uue andmebaasiskeemi prototüüp

Pärast täielikku vana andmebaasiskeemi analüüsi autor projekteerib uue andmebaasiskeemi prototüüpi Rational Rose programmi abil. Autor parandab tõsised vead, millest oli kirjutatud eelmises peatükis. Antud skeem on esitatud järgmisel joonisel.

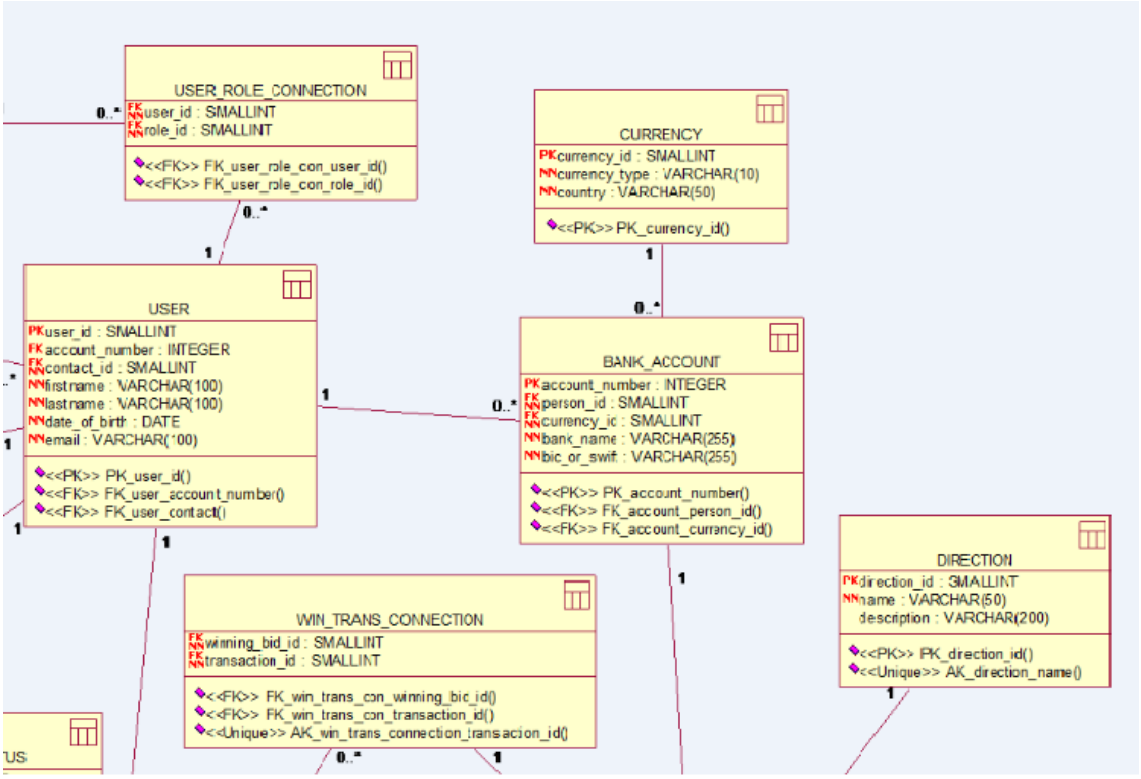


Joonis 10. Andmebaasiskeem Rational Rose programmi kaudu

4.2 Vigade parandused

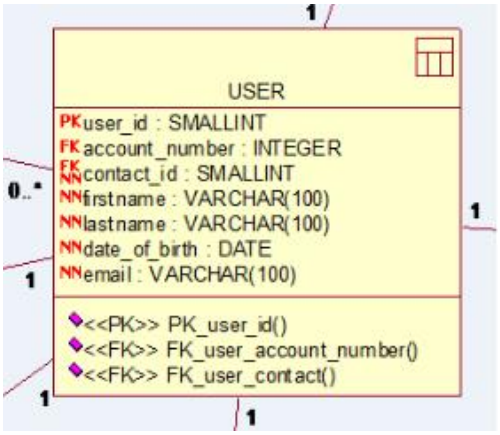
Allpool on toodud põhiliste parandatud vigade näited:

- Skeemis on näha, et enamustel väljadel on atribuut NOT NULL ning tänu sellele tabelid ei sisalda tühje kirjeid.



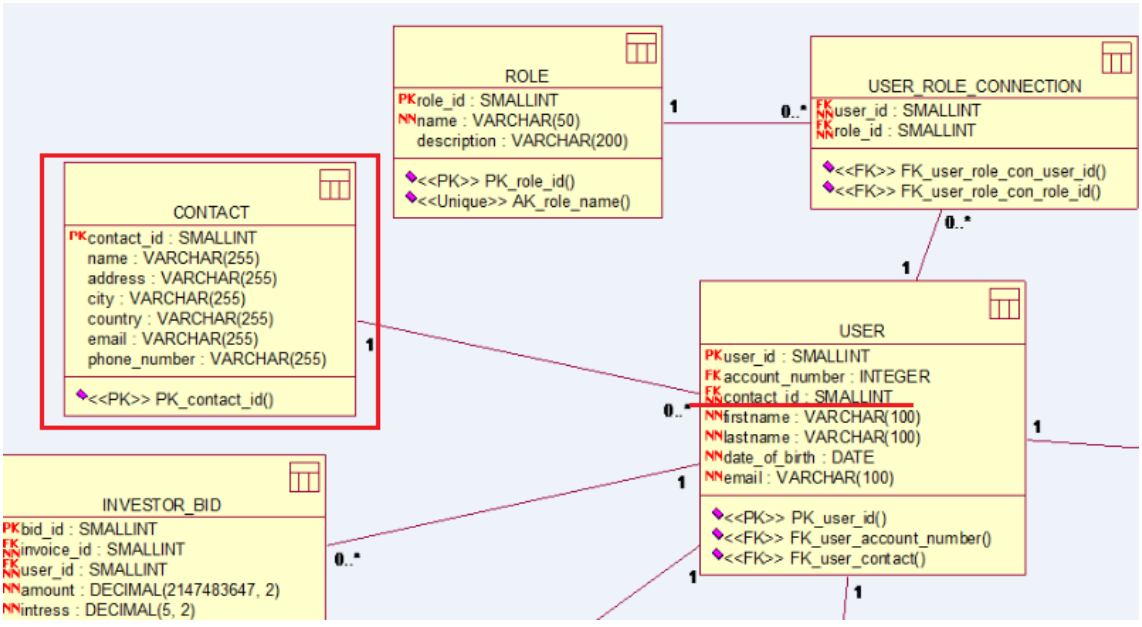
Joonis 11. Osa skeemist, kus on näha atribuutidel NN – NOT NULL väärtus

- Tabelites puuduvad dubleeritud andmed. Näiteks kasutaja andmed nagu nimi, perenimi, sünnikuupäev jne on esitatud ainult tabelis *user* ja teistes tabelites on kasutatud ainult *user_id* välisvõtme abil.



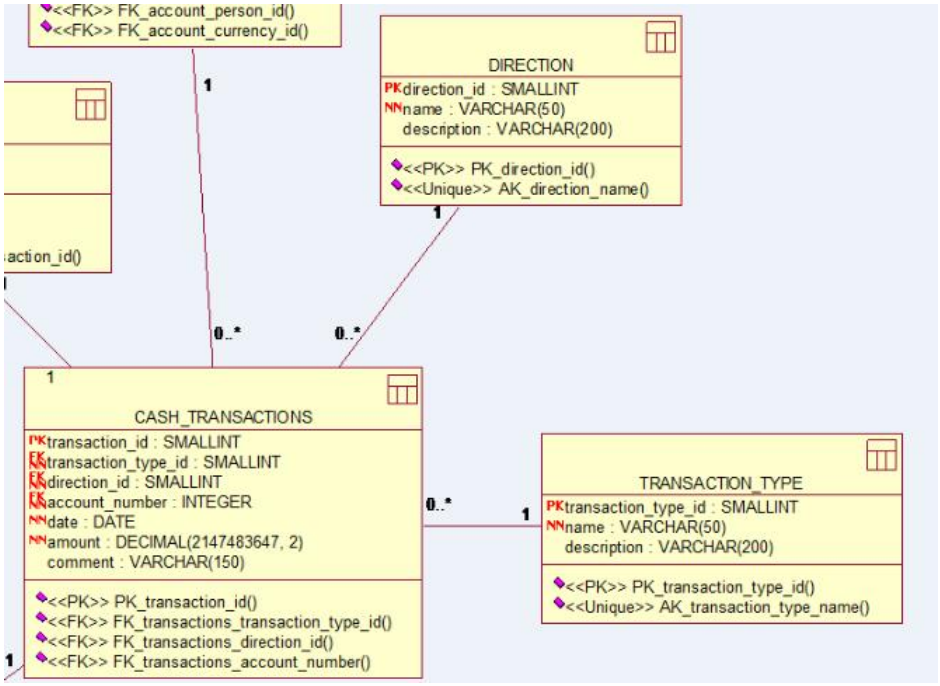
Joonis 12. Tabel *user*

- Samuti nagu eelmises punktis dubleeritud andmete vältimiseks on loodud tabel *contact*, kus on kõik vajalikud andmed nagu aadress, telefoni number jne. Tabel *contact* on seotud tabeliga *user* välisvõtme kaudu.



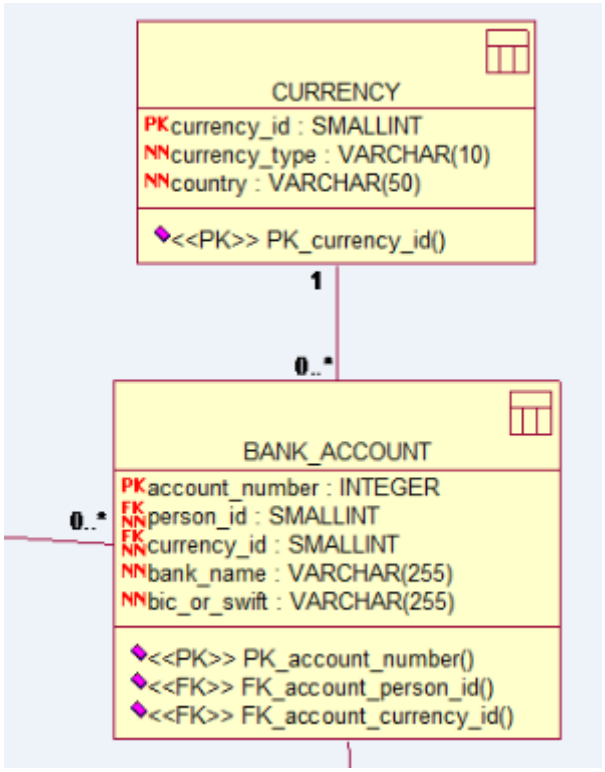
Joonis 13. Tabel *contact* ja seos tabeliga *user*

- Põhiline tabel *cash_transaction* on loodud niimodi, et on hästi näha, milline transaktsioon tehti, kas näiteks debet või credit, see on esitatud skeemis eraldi tabelis *direction*. Samuti on näha transaktsiooni tüüpi, näiteks intress või põhimaks, mis on ka loodud eraldi tabelis *transaction_type*. Nüüd debet ja credit on tabelis kaks erinevat transaktsiooni, mitte nagu oli vanas skeemis ühe kirjega.



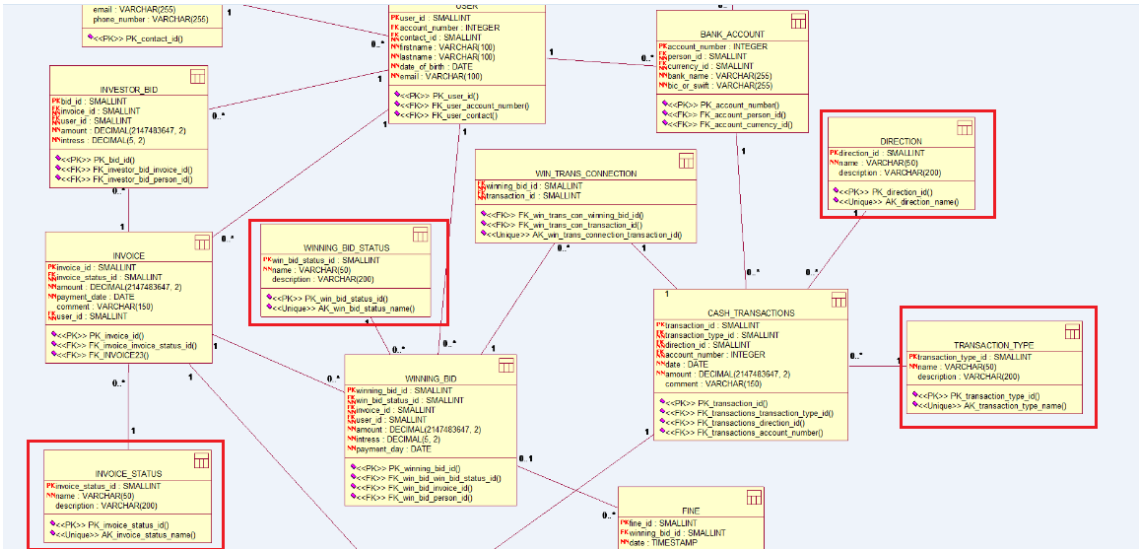
Joonis 14. Tabel *cash_transaction* ja tema klassifikaatorite tabelid

- Skeem on loodud eraldi tabel *bank_account* selleks, et panga andmed ei korduks tuhat korda, nagu see oli vanas skeemis tabelis *profile*.



Joonis 15. Tabel *bank_account* ja eraldi tabel *currency*

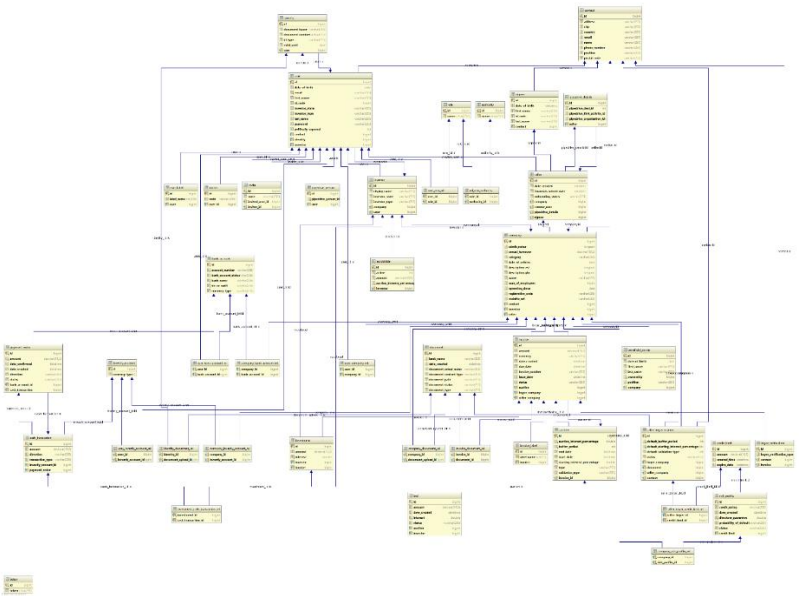
- Skeemi on lisatud mitu klassifikaatorite tabelit, näiteks *direction*, *winning_bid_status*, *invoice_status*, *transaction_type*.



Joonis 16. Klassifikaatorite tabelite näited

4.3 Uus andmebaasiskeem

Antud skeem sai prototüübiks Investly arendustiimile, et Java keeles genereerida täielik skeem koos vajalike muutustega. Kui uus skeem sai valmis ja oli edastanud autorile, siis autor võis alustada antud lõputöö põhilise ülesandega tegeleda – migreerida andmed. Uus skeem on näidatud allpool. Loetava skeemi saab vaadata Lisas – 2.



Joonis 17. Uus Investly andmebaasiskeem

5 Andmete migreerimine

Käesolevas peatükis kirjeldatakse kuidas migreerida andmeid vanast andmebaasist kirjutades SQL päringukeeles koodi.

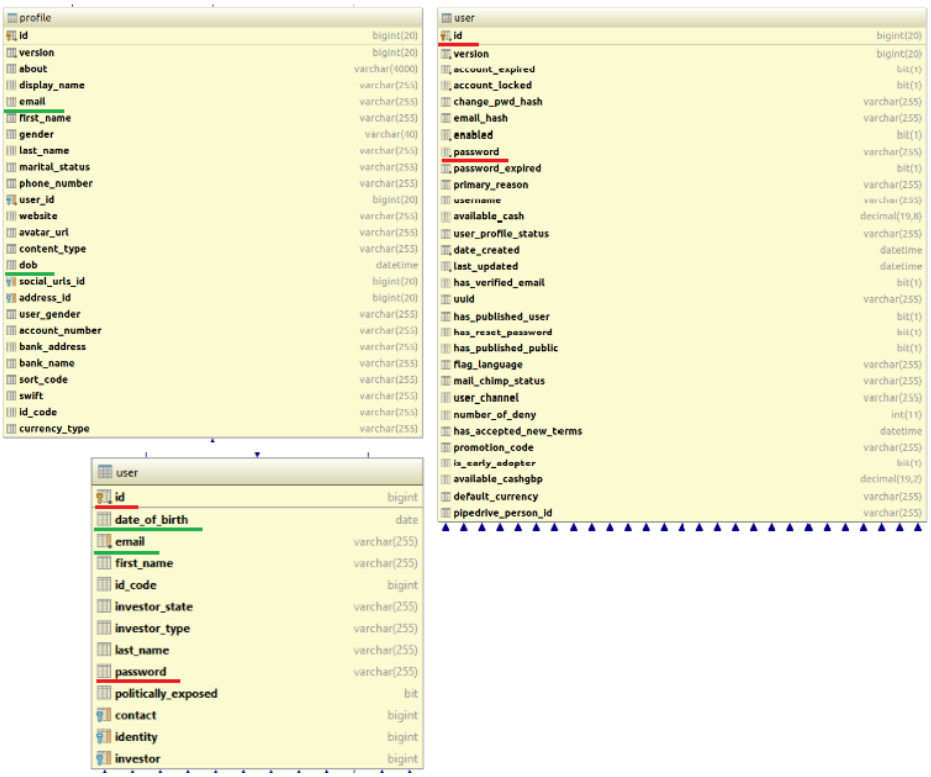
Selleks et üle tuua andmed ühest andmebaasist teise, autor kasutab programmi, mille nimi on MariaDB ja see toetab MySQL. Programmis Notepad++ on kirjutatud SQL kood, mis loob kõik tabelid, kõik seosed ja vajalikud võtmed ning vastab täiesti uuele skeemile.

Selle koodi abil on tehtud andmete migreerimine, mis on selle lõputöö tulemus.

5.1 Tabelite analüüs ning SELECT lausete kirjutamine

Alustamiseks on vaja:

- Leida sobivaimad veergude väärtused uue skeemi igas tabelis, millele vastavad väärtused vanast skeemist. Näiteks, tabelis *user* uues skeemis on andmed, mis on võetud tabelitest *user* ja *profile* vanast skeemist. Järgmisel pildil on näha, mis andmed on võetud tabelitest.



Joonis 18. Andmed uues tabelis *user*, mis vastavad erinevatele tabelitele vanast andmebaasist

- Kirjutada SELECT lause, mis on seotud vana andmebaasiga ja võtab sealt vajalikud andmed. Näide on järgmine:

```
SELECT U.id AS id, DATE(P.dob) AS date_of_birth, P.email AS email,
       P.first_name AS first_name,CAST(P.id_code AS SIGNED) AS id_code,
       P.last_name AS last_name, U.password AS password, 0 AS
       politically_exposed
FROM investly_old.user AS U
INNER JOIN investly_old.profile AS P ON U.id = P.user_id
```

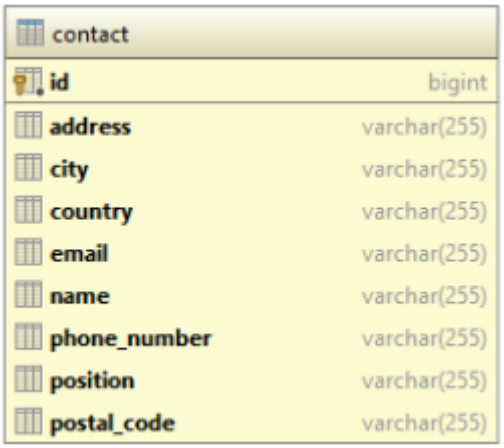
Joonis 19. Näide SQL koodist, kuidas andmed võetakse vanast andmebaasist

Kuna tabeleid on palju, siis sobivate andmete otsingu jaoks kasutab autor MariaDB käsurida, kus on väga mugav vaadata tabelite kirjeldusi, nende väärtuseid ning samuti leida sarnased andmed spetsiaalse lause abil:

```
SELECT DISTINCT table_name
FROM INFORMATION_SCHEMA.COLUMNS
WHERE COLUMN_NAME LIKE "%company%"
```

Joonis 20. Kergem variant, et otsida vajalikud tabelid

Veel üks näide on tabel *contact* ja see tabel on illustreeritud allpool.



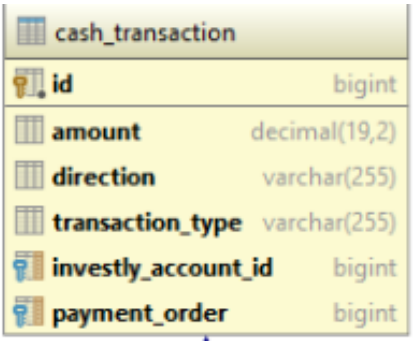
Joonis 21. Tabel *contact* uues andmebaasis

Selleks, et võtta andmed vanast andmebaasist, on vaja seostada mitu tabelit kokku ning kuidas seda teha on näidatud allpool toodud väikeses osas koodist.

```
SELECT A.id AS id, A.locality AS address, A.city AS city, A.country AS
       country, A.public_email AS email, A.phone_number AS phone_number,
       COALESCE(CONCAT(P.first_name, " ", P.last_name) , C.business_name) AS
       name, CD.position AS position, A.postal_code AS postal_code
FROM investly_old.address AS A
LEFT JOIN investly_old.company AS C ON A.id = C.address_id
LEFT JOIN investly_old.profile AS P ON A.id=P.address_id
JOIN investly_old.user AS U ON U.id = P.user_id
LEFT JOIN investly_old.user AS U2 ON U2.id=C.creator_id
LEFT JOIN investly_old.seller AS S ON U2.id = S.user_id
LEFT JOIN investly_old.contact_detail AS CD ON S.contact_detail_id = CD.id
```

Joonis 22. Vanast andmebaasist võetud andmed tabelisse *contact*

Kõige keerulisem tabel on *cash_transaction*, kuna see on kõige olulisem tabel ja seal on kõige rohkem kirjeid. Vanas andmebaasis on kaks tabelit: *transaction* ja *transaction_detail*, teises on põhiline informatsioon sellest, et mis tüüpi ja suunda on tehtud transaktsioon, kas debet või credit, kas intress või põhimaks. Uues skeemis on eraldi klassifikaator *transaction_type* ja *direction*, kus on määratud, kas tehtud transaktsioon on debet või credit. Selleks, et ühendada kõik debeti transaktsioonid, tuleb võtta neid kirjeid, millel tabelis *transaction_detail* ja veergudel, millel on *_in* osa, mitte null väärtus. Vastasel juhul, crediti transaktsioonides on mitte null väärtused veergudel lõpuga *_out*. Pilt, mis näitab tabeli *cash_transaction* ning vastav SQL koodi näide on toodud allpool.



Joonis 23. Tabel *cash_transaction*

```
SELECT TD.id AS id, CAST(T.amount AS DECIMAL(19,2)) AS amount, T.date_created
       AS date_created, "DEBIT" as direction,
       CASE
              WHEN (TD.principal_receviable_in > 0) THEN 'PRINCIPAL'
              WHEN (TD.interest_payble_in > 0) THEN 'INTEREST'
              WHEN (TD.penalties_payble_in > 0) THEN 'PENALTIES'
              WHEN (TD.fees_payable_in > 0) THEN 'AUCTION_FEE'
       ELSE "OTHER"
       END AS transaction_type
FROM investly_old.transaction AS T
JOIN investly_old.transaction_detail AS TD ON T.id = TD.transaction_id
UNION
SELECT (TD.id + 200000) AS id, CAST(T.amount AS DECIMAL(19,2)) AS amount,
       T.date_created AS date_created, "CREDIT" as direction,
       CASE
              WHEN (TD.principal_receviable_out > 0) THEN 'PRINCIPAL'
              WHEN (TD.interest_payble_out > 0) THEN 'INTEREST'
              WHEN (TD.penalties_payble_out > 0) THEN 'PENALTIES'
              WHEN (TD.fees_payable_out > 0) THEN 'AUCTION_FEE'
       ELSE "OTHER"
       END AS transaction_type
FROM investly_old.transaction AS T
JOIN investly_old.transaction_detail AS TD ON T.id = TD.transaction_id
```

Joonis 24. Vastavalt tabelile *cash_transaction* võetud andmed vanast andmebaasist

Selleks, et tabelis ei tuleks topelt rida, teises SELECTis on pandud *id + 200000*.

5.2 CREATE TABLE lausete koostamine

Kui kõik vajalikud andmed on võetud vanast andmebaasist SELECT lause abil, loob autor uusi tabeleid, kasutades CREATE TABLE lauset, siis selle sees on SELECT lause, mis on juba kirjutatud ning andmed on võetud vanast andmebaasist. Näidises on tabel *company*.

```
DROP TABLE IF EXISTS company;
CREATE TABLE company (
	SELECT C.id AS id, CAST(C.stated_turnover AS DECIMAL(19,2)) AS
		annual_turnover, BD.company_category AS category,
		DATE(C.date_of_article) AS date_of_articles, BD.description AS
		description_est, C.business_name AS name,
		BD.employees_number AS num_of_employees, DATE(C.date_created) AS
		operating_since, C.registration_code AS registration_code,
		C.companyurl AS website_url
	FROM investly_old.company AS C
	INNER JOIN investly_old.business_detail AS BD ON C.id = BD.company_id
);
```

Joonis 25. Tabel *company* koos migreeritud andmetega

5.2.1 Primaarvõtmete lisamine

Järgmisena autor lisab kõikidesse tabelitesse PRIMARY KEY, kuna see on tähtis normaliseerimise reegel. Igas tabelis peab olema PRIMARY KEY. Kuidas lisada PRIMARY KEY tabelisse on näidatud järgmisel joonisel ja näidisena on valitud tabel *user*.

```
ALTER TABLE user ADD PRIMARY KEY (id);
ALTER TABLE user MODIFY id BIGINT AUTO_INCREMENT;
```

Joonis 26. PRIMARY KEY tabelisse lisamine

Kui kõik vajalikud andmed on võetud vanast andmebaasist ning loodud uued tabelid SELECT lause abil, uues skeemis on jäänud veel tabelid, mis puuduvad vanas skeemis või vanas olevad andmed uues andmebaasis ei ole vaja migreerida. Need tabelid luuakse tavalise CREATE TABLE lausega. Järgmisel joonisel on toodud näide tabeliga *payment_order*.

```
DROP TABLE IF EXISTS payment_order;
CREATE TABLE payment_order (
	id BIGINT NOT NULL AUTO_INCREMENT,
	amount DECIMAL(19,2) NOT NULL,
	date_confirmed DATETIME,
	date_created DATETIME,
	direction VARCHAR(255),
	status VARCHAR(255),
	bank_account_id BIGINT,
	cash_transaction BIGINT,
	PRIMARY KEY(id)
);
```

Joonis 27. Tabeli loomine ilma andmete migreerimiseta

5.2.2 Välisvõtmete lisamine

Pärast tabelite loomist ja PRIMARY KEY lisamist on vaja seostada tabelid ning lisada FOREIGN KEY.

```
ALTER TABLE user ADD CONSTRAINT FK_contact_user FOREIGN KEY (contact)
       REFERENCES contact (id) ON DELETE NO ACTION ON UPDATE CASCADE;
```

Joonis 28. FOREIGN KEY tabelisse lisamine

Tabel *roles* vanas andmebaasiskeemis oli loodud nii, et PRIMARY KEY – *id* väljal oli tüüp VARCHAR ja väli sisaldas tekstiandmeid, samad nagu *name* väli. Andmete migreerimiseks tuleb *id* väljale anda 0 väärtus, et pärast oleks võimalik anda igale reale õige *id* number ja teha PRIMARY KEY-ks. Tabel *role* on esitatud järgmisel joonisel.

```
DROP TABLE IF EXISTS role;
CREATE TABLE role (
       SELECT 0 AS id, R.name AS name
       FROM investly_old.roles AS R
);
UPDATE role SET id = '1' WHERE name = 'ROLE_ADMIN';
UPDATE role SET id = '2' WHERE name = 'ROLE_BORROWER';
UPDATE role SET id = '3' WHERE name = 'ROLE_BUYER';
UPDATE role SET id = '4' WHERE name = 'ROLE_COMPANY';
UPDATE role SET id = '5' WHERE name = 'ROLE_INVESTOR';
UPDATE role SET id = '6' WHERE name = 'ROLE_LENDER';
UPDATE role SET id = '7' WHERE name = 'ROLE_SALES';
UPDATE role SET id = '8' WHERE name = 'ROLE_SELLER';
UPDATE role SET id = '9' WHERE name = 'ROLE_USER';
UPDATE role SET id = '10' WHERE name = 'ROLE_USER_FULLY_REG';
UPDATE role SET id = '11' WHERE name = 'ROLE_USER_NOT_ACTIVE';
UPDATE role SET id = '12' WHERE name = 'ROLE_USER_NOT_FULLY';
ALTER TABLE role ADD PRIMARY KEY (id);
ALTER TABLE role MODIFY id BIGINT AUTO_INCREMENT;
```

Joonis 29. Tabel *role* uues skeemis

5.3 Seosetabelite loomine

Kui kõik vajalikud tabelid on loodud ning andmed on võetud vanast andmebaasist, on jäänud looma seosetabeleid, mis seostavad kaks tabelit omavahel ainult välisvõtme abil. Need tabelid on vaja luua, et seostada omavahel domeenid, näiteks kasutajat pangakonto või firmaga ning seos on nende vahel kas *one-to-many* või *many-to-one*, mis tähendab

seda, et ühel kasutajal võib olla mitu pangakontot, aga ühel pangakontol on ainult üks kasutaja. Samuti on neid tabeleid vaja, kui on *many-to-many* seos, näiteks üks kasutaja on seotud mitme firmaga ja üks firma mitme kasutajaga.

Näidises on tabelid *company* ja *bank_account* ja nad on seotud omavahel *one-to-many* seosega ning seosetabel on esitatud allpool.

```
DROP TABLE IF EXISTS company_bank_account_rel;
CREATE TABLE company_bank_account_rel (
	company_id BIGINT NOT NULL,
	bank_account_id BIGINT NOT NULL,
	FOREIGN KEY (company_id) REFERENCES company(id),
	FOREIGN KEY (bank_account_id) REFERENCES bank_account(id)
);
```

Joonis 30. Seosetabeli näide *company* ja *bank_account* vahel

Kirjutatud SQL kood on kasutamiseks valmis, kui kõik vajalikud tabelid on loodud, kõik andmed on võetud vanast andmebaasist, kõik primaar- ja välisvõtmed on loodud ning samuti seosetabelid on koostatud. Kontrollimiseks autor käivitas valmistatud SQL fail MariaDB programmis, kus saab üle vaadata ja parandada vigu, kui neid leidub. Pärast edukat kontrolli, antud SQL kood on saadetud Investly arendustiimile edasi kasutamiseks.

6 Kokkuvõte

Käesoleva töö eesmärgiks oli luua uue andmebaasiskeemi prototüüp ning uue skeemi põhjal migreerida andmeid vanast andmebaasist. Selleks oli tehtud vana andmebaasiskeemi analüüs, leitud tõsised vead ning loodud uus andmebaasiskeemi prototüüp IBM Rational Rose programmi abil.

Uue andmebaasiskeemi prototüüp sai normaliseeritud ja põhilised vead parandatud, näiteks tabelites ei ole enam korduvaid veerge, on loodud eraldi klassifikaatorite tabelid, eemaldatud NULL väärtused, kuna neid oli liiga palju, mille tõttu andmebaasi töö oli aeglasem. Paljud andmed olid teisendanud, näiteks mõned tekstiväljad tuli konverteerida täisarvuks, et teha seda primaarvõtmeks.

Töö käigus oli tehtud palju koostööd Investly arendustiimiga, kuna nemad genereerisid prototüübi põhjal tegelikku andmebaasiskeemi. Selle skeemi põhjal on teinud autor andmete migreerimist, kirjutades SQL koodi ja võttes andmeid vanast andmebaasist SELECT ja CREATE TABLE lausete abil.

Töö eesmärgid on täidetud ning tulemusena on esitatud valmis SQL kood arendustiimile. Uus skeem on normaliseeritud, ei sisalda korduvaid andmeid ja ridu ning kõik seosed on tehtud välisvõtmete abil.

Kasutatud kirjandus

[1] „Investly web site,“ [Võrgumaterjal]. Available: http://investly.co [Kasutatud 29 November 2016].

[2] „Faktooringust,“ [Võrgumaterjal]. Available: http://arileht.delfi.ee/news/uudised/faktooring-mis-see-on-ja-kuidas-aitab-ettevotte-tegevust-rahastada?id=70109011

[3] „Data migration,“ [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/Data_migration

[4] „Schema migration,“ [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/Schema_migration

[5] „Normaliseerimine,“ [Võrgumaterjal]. Available: https://support.microsoft.com/ru-ru/kb/283878

[6] „Normaliseerimisest,“ [Võrgumaterjal]. Available: http://corpgov.crew.ee/Materjalid/Normaliseerimine_ee.pdf

Lisa 1 – Loetav vana andmebaasiskeem

Allpool toodud lingil saab näha vana andmebaasiskeemi pilti ning vajadusel suurendada:

https://drive.google.com/open?id=0BxdxC0FLNmY4TEJXMXY3UGY3amc

Lisa 2 – Loetav uus andmebaasiskeem

Allpool toodud lingil saab vaadata uue andmebaasiskeemi pilti ning vajadusel suurendada:

https://drive.google.com/open?id=0BxdxC0FLNmY4cWR1dEltMG5lYkE