Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Matic Bernik

Zlivanje podatkov v relacijskih podatkovnih bazah

diplomsko delo

univerzitetni študijski program
prve stopnje
računalništvo in informatika

Mentor: doc. dr. Tomaž Curk

Ljubljana, 2018

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Relacijske baze navadno opisujejo prepleteno mrežo relacij med entitetami. Razumevanje in črpanje podatkov iz tovrstnih zbirk zahteva od povprečnega uporabnika, ki prvič dostopa do baze, sorazmerno velik časovni vložek in napor. Preučite možnosti samodejnega razvrščanja relacij po pomembnosti in v ta namen preverite ustreznost pristopa zlivanja podatkov, ki temelji na razcepu matrik. Poročajte o uspešnosti razvitega pristopa na nekaj umetnih in realnih primerih.

Zahvaljujem se doc. dr. Tomažu Curku za usmerjanje in strokovno vodenje, ki mi ga je kot mentor nudil tekom izdelave diplomskega dela. Zahvaljujem se tudi svoji družini in prijateljem za vso podporo v času študija. Hvala!

Kazalo

Povzetek

Abstract

1 Uvod **1**

1.1 Motivacija 1

1.2 Struktura diplomskega dela 1

2 Podatki **3**

2.1 Relacijske podatkovne baze 3

2.2 PostgreSQL 4

2.3 Poizvedovalni jezik SQL 6

3 Pregled obstoječih orodij **9**

3.1 QueRIE 10

3.2 YMALDB 10

3.3 Wrangler 11

4 Predlagana metoda za rangiranje relacij **13**

4.1 Zlivanje podatkov z matričnim razcepom 14

4.2 Gradnja relacijskih matrik 17

4.3 Obvladovanje prostorske in časovne zahtevnosti problema . . . 18

5 Orodje **21**

5.1 Predstavitev delovanja orodja 21

5.2 Primer uporabe orodja 31
5.3 Ponovljivost rangiranja 40

6 Sklepne ugotovitve **43**

Literatura **47**

Seznam uporabljenih kratic

kratica	angleško	slovensko
CA	classification accuracy	klasifikacijska točnost
SUPB	database management system	sistem za upravljanje podatkovnih baz
DFMF	data fusion by matrix factorization	zlivanje podatkov z matričnim razcepom
RMSE	root mean square error	koren povprečja kvadratov napak
PK	primary key	primarni ključ
FK	foreign key	tuji ključ
SQL	structured query language	strukturiran poizvedovalni jezik
DQL	data query language	jezik za poizvedovanje nad podatki
DDL	data definition language	jezik za definicijo podatkov
DCL	data control language	jezik za nadzor nad podatki
DML	data manipulation language	jezik za rokovanje s podatki

Povzetek

Naslov: Zlivanje podatkov v relacijskih podatkovnih bazah

Avtor: Matic Bernik

Analiza podatkov in odkrivanje povezav med entitetnimi tipi znotraj podatkovnih baz sta lahko zelo zahtevni in zamudni opravili. Vsaka baza ima specifično strukturo, raziskovanje katere od uporabnika terja veliko časa. V diplomski nalogi ovrednotimo uporabo rekonstrukcijske napake pri modeliranju z matričnim razcepom kot mero za ocenjevanje stopnje povezanosti med tabelami oz. entitetnimi tipi v podatkovnih bazah. V ta namen smo razvili orodje, ki se poveže na podatkovno bazo in izpiše rangiran seznam relacij med hranjenimi entitetnimi tipi. Uporabnik lahko tako odkrije informativne povezave in usmerjeno raziskuje podatkovno bazo.

Ključne besede: zlivanje podatkov, podatkovna baza, rangiranje relacij, preiskovalna analiza.

Abstract

Title: Data fusion of relational databases

Author: Matic Bernik

Data analysis and discovery of relations between connected entity types within databases can be very labour and time intensive. The reason being that every database has its specific structure, which needs to be examined. In this thesis, we evaluated if the reconstruction error of the matrix factorization model can be used to relate tables or entity types within a database. To test this concept, we developed a Python module that connects to a database and returns a ranked list of relations (pairs of entity types). This enables the user to identify the most informative relations and explore them further.

Keywords: data fusion, database, relation ranking, exploratory analysis.

Poglavje 1

Uvod

1.1 Motivacija

Velik del strukturiranih podatkov se hrani v podatkovnih bazah ali podatkovnih skladiščih. Če želimo analizirati takšno bazo, se moramo najprej seznaniti z njeno strukturo. Prepoznati moramo tabele, ki predstavljajo tiste izmed entitetnih tipov, ki so relevantni za našo analizo. Velikokrat nam povezave med parom ali skupino entitetnih tipov, katere želimo preučiti, določa že problem, ki ga rešujemo. V nekaterih primerih pa nam naloga določa zgolj vrsto entitete, o kateri bi želeli vedeti več. V takem primeru moramo sami ugotoviti, katere povezave je vredno preiskovati, pri čemer problemskega prostora včasih ne moremo dovolj omejiti niti z domenskim znanjem. Kolikor nam je znano trenutno ni nobenega uveljavljenega pristopa za razvrščanje povezav med pari entitetnih tipov v podatkovnih bazah.

1.2 Struktura diplomskega dela

V drugem poglavju Podatki opišemo relacijske podatkovne baze in način interakcije z njimi preko poizvedovalnega jezika SQL. Nekoliko podrobneje obdelamo tudi PostgreSQL SUPB, katerega podpira naša implementacija.

Kljub temu, da metod in orodij, ki bi rangirale relacij med tabelami oz.

entitetnimi tipi v podatkovni bazi, nismo našli, pa v tretjem poglavju Pregled obstoječih orodij predstavimo nekaj orodij ki podpirajo uporabnika v fazi spoznavanja z zakonitostmi podatkov hranjenih v relacijskih podatkovnih bazah.

V četrtem poglavju predstavimo predlagano metodo in njene ključne komponente. Od teh posvetimo večjo pozornost metodi Zlivanja podatkov z matričnim razcepom (in njeni implementaciji - programskemu modulu *scikit-fusion*), ki je pri naši metodi ključnega pomena za samo ocenjevanje stopnje povezanosti parov entitetnih tipov.

Peto poglavje je namenjeno predstaviti programskega modula Python, ki smo ga tekom dela razvili in predstavlja implementacijo naše metode. Uporabo in vmesne rezultate predstavimo nad testno podatkovno bazo, ki vsebuje majhno število tabel in vnosov.

V sklepnem poglavju predstavimo nekaj možnih izboljšav in predlogov za nadaljnje delo.

Poglavje 2

Podatki

Vir podatkov, nad katerim deluje naša rešitev, so relacijske podatkovne baze. Poglavje se začne z opisom relacijske podatkovne baze. Nekoliko podrobneje obdelamo še PostgreSQL SUPB, specifike katerega podpira naše orodje. Na koncu predstavimo še poizvedovalni jezik SQL in nekaj poizvedb, ki jih naše programsko orodje izvede nad tarčno podatkovno bazo, z namenom pridobitve podatkov potrebnih za pripravo rangiranega seznama relacij.

2.1 Relacijske podatkovne baze

Relacijskim podatkovnim bazam je skupno, da so podatki v njih predstavljeni s tabelami. Vsaka vrstica tabele predstavlja entiteto, ki je opisana z množico atributov oz. stolpcev. Tabele so nato medsebojno povezane v razmerja:

- ena-proti-mnogo,
- mnogo-proti-mnogo,
- ena-proti-ena.

Vsak zapis v tabeli mora imeti oznako, ki ga enolično določa. V ta namen se uporablja enega ali več stolpcev, ki skupaj tvorijo primarni ključ. Vsak objekt lahko svoj atributni opis tudi razširi s sklicevanjem na drug objekt, ki

se lahko nahaja v isti, lahko pa tudi v drugi tabeli. To dosežemo z enim ali večimi dodatnimi atributi, ki se preslikujejo v stolpce, ki v tabeli referenciranega objekta sestavljajo primarni ključ. Takšno skupino stolpcev imenujemo tuji ključ.

Sistem za upravljanje s podatkovnimi bazami (SUPB) skrbi za:

- kreiranje podatkovnih struktur,
- vzdrževanje podatkovne zbirke,
- zaščito podatkov,
- zagotavljanje integritete podatkov,
- izvajanje transakcij.

Sistem SUPB se razlikuje med ponudniki (MySQL, Oracle SQL, PostgreSQL, itd). Razlike se med drugim odražajo v organizaciji sistemskih tabel, katere hranijo razne strukturne, varnostne in druge meta podatke o sami podatkovni bazi. Za povezovanje na resnično poljubno relacijsko podatkovno bazo, bi morali razvito programsko orodje prilagoditi posebnostim vsakega ponudnika, na bazo katerega bi se želeli povezovati. Med razvojem smo se osredotočili na podprtost podatkovnih baz, s katerimi upravlja PostgreSQL.

2.2 PostgreSQL

PostgreSQL je odprtokodni sistem za upravljanje objektno-relacijskih podatkovnih baz, ki podpira tako Windows kot tudi operacijske sisteme, ki temeljijo na Linux in UNIX osnovi. PostgreSQL podpira širok nabor podatkovnih tipov, od katerih se naše orodje v prvi vrsti osredotoča na številske ter na tekstovne tipe:

- bigint (predznačena 8-bitna celoštevilska vrednost),
- bigserial (samo-naraščajoča 8-bitna celoštevilska vrednost),

- bit (niz bitov),
- boolean (logična vrednost 'True' ali 'False'),
- character (niz znakov),
- date (koledarski datum označen z letom, mesecem in dnevom),
- double precision (decimalno število zapisano v dvojni natančnosti),
- integer (predznačena 4-bitna celoštevilska vrednost),
- numeric (decimalna vrednost poljubne natančnosti),
- real (4-bitno decimalno število),
- smallint (predznačeno 2-bytno celo število),
- serial (samo-naraščajoče 4-bytno celo število),
- text (niz znakov spremenljive dolžine),
- timestamp (časovni žig - datum in čas).

Sistemski katalog je ime za množico sistemskih tabel, ki hranijo informacije o zgradbi baze. Od teh nas zanimajo:

- seznam vseh podatkovnih tabel,
- seznam stolpcev znotraj posamezne tabele in informacije o podatkovnih tipih, ki jih ti hranijo,
- informacije o omejitvah, ki se navezujejo na posamezne tabele (primarni in tuji ključi, podatkovni tipi stolpcev itd.).

Pri postgres podatkovnih bazah lahko tovrstne informacije pridobimo s poizvedbami sistemskih tabel:

- pg_tables,

- pg_index,
- pg_attribute,
- columns,
- referential_constraints,
- key_column_usage.

2.3 Poizvedovalni jezik SQL

SQL je programski jezik razvit posebej za potrebe upravljanja strukturiranih podatkov, ki se hranijo v relacijskih podatkovnih bazah ali pa se pretakajo v podatkovnih tokovih. Stavke jezika SQL lahko glede na namembnost razdelimo v štiri kategorije [6]:

- jezik za poizvedovanje nad podatki (DQL),
- jezik za definicijo podatkov (DDL),
- jezik za nadzor nad podatki (DCL),
- jezik za rokovanje s podatki (DML).

Tekom dela smo uporabljali pretežno stavke DQL za priklic opisa same strukture in povezav znotraj podatkovne baze ter za pridobitev podatkov potrebnih za gradnjo relacijskih matrik. Za namene stvaritve manjše, vzorčene različice podatkovne baze (v primeru, da ima obravnavana podatkovna baza veliko število zapisov), uporabljamo tudi stavke DDL, z uporabo katerih preselimo shemo podatkov.

V orodju skušamo večji del obdelave podatkov preložiti na SUPB podatkovne baze z gradnjo ustreznih SQL poizvedb. Nekaj primerov poizvedb SQL, ki jih naš program uporablja:

- poizvedba, s katero pridobimo seznam podatkovnih tabel:

```
SELECT *
FROM pg\_catalog.pg\_tables
WHERE schemaname = 'public';
```

- pozvedba, s katero pridobimo seznam tujih ključev:

```
SELECT c.constraint_name
                , x.table_name
                , x.column_name
                , y.table_name as foreign_table_name
                , y.column_name as foreign_column_name
            FROM information_schema.referential_constraints c
            JOIN information_schema.key_column_usage x
                ON x.constraint_name = c.constraint_name
            JOIN information_schema.key_column_usage y
                ON y.ordinal_position = x.position_in_unique_constraint
                AND y.constraint_name = c.unique_constraint_name
            ORDER BY c.constraint_name
             , x.ordinal_position;
```

- poizvedba, s katero pridobimo seznam primarnih ključev za tabelo, katere ime hrani spremenljivka table_name:

```
SELECT '"+table_name+"'
, a.attname
, format_type(a.atttypid, a.atttypmod) AS data_type
FROM pg_index i
JOIN pg_attribute a
ON a.attrelid = i.indrelid
AND a.attnum = ANY(i.indkey)
WHERE i.indrelid = '"+table_name+"'::regclass
AND i.indisprimary;
```

Poglavje 3

Pregled obstoječih orodij

V poglavju predstavimo nekatere metode in orodja, ki uporabnika podpirajo v fazi preiskovalne analize nad podatkovno bazo. Tradicionalni sistemi za upravljanje podatkov predpostavljajo da uporabnik, ki sestavlja poizvedbo, dobro pozna podatkovno shemo in vsebino baze ter je prepričan, da je oblikovana poizvedba res tista, ki bi jo rad zastavil. Te predpostavke niso vedno pravilne. Naraščajoča količina zbranih podatkov zahteva spremembo pristopa k njihovemu upravljanju. Pogosto bi uporabnik želel biti najprej opozorjen na zanimive vzorce, ki se kažejo v podatkih. Na podlagi priporočil bi uporabnik oblikoval naslednji korak analize. Sisteme, ki uporabniku pomagajo pri preiskovanju podatkovne baze, delimo na tri kategorije [7]:

- sistemi, ki uporabniku pomagajo pri oblikovanju poizvedb SQL,
- sistemi, ki avtomatizirajo proces preiskovanja podatkov s samodejno prepoznavo in predstavitvijo relevantnih podatkov - naša metoda pripada tej kategoriji,
- vmesniki, ki uvajajo nove načine poizvedovanja.

3.1 QueRIE

QueRIE [4] je sistem namenjen podpori pri preiskovanju podatkovnih baz s pomočjo priporočanja s sodelovanjem več uporabnikov (ang. *collaborative*). Sistem neprestano spremlja uporabnikovo poizvedovanje nad bazo in pregleduje sistemski dnevnik poizvedb. Medtem poskuša prepoznati vzorce podobne tistim, ki so jih ustvarili predhodni uporabniki. Glavna predpostavka je, da imata uporabnika, ki nad bazo izvajata podobne poizvedbe, tudi podobne informacijske potrebe. QueRIE torej uporabi informacijo o podobnih uporabnikih in njihovih poizvedbah za priporočanje novih poizvedb trenutnemu uporabniku. Na te poizvedbe lahko uporabnik gleda kot na predloge, ki jih bistveno lažje in hitreje prilagodi svojim potrebam, kot da bi sam oblikoval popolnoma nove. QueRIE vsako poizvedbo razstavi na osnovne elemente, ki zajemajo njeno logiko. Te nato uporabi za opis prepoznavnih lastnosti uporabnikovega poizvedovanja in računanje podobnosti med različnimi uporabniki.

3.2 YMALDB

Uprabnikova interakcija s podatkovno bazo običajno poteka preko poizvedb SQL. Orodje YMALDB [5] ('You May Also Like Data Base') uporabnika pri preiskovanju podatkov usmerja v dele podatkovne baze, ki sicer niso del odgovora na prvotno poizvedbo. Pri tem se, za razliko od tradicionalnih priporočilnih sistemov in OLAP navigacijskih sistemov, ne zanaša na obstoj podatkovnih dnevnikov s poizvedbami ostalih uporabnikov. Uporablja zgolj informacijo o vsebini podatkovne baze in zgradbi prvotne poizvedbe. YMALDB uporabniku ob poizvedovanju kot priporočila dodatno vrne seznam parov (atribut, vrednost), ki so zelo povezani s tistimi iz rezultata prvotne poizvedbe.

Osnova za izračun podobnosti med vrsticami so pojavitve posebej zanimivih množic vrednosti atributov (poimenovanih tudi *faset*) znotraj rezultatov poizvedbe. Zanimivost posameznega *faset* temelji na njegovi pogostosti

znotraj rezultata poizvedbe v primerjavi s pogostostjo znotraj podatkovne baze. Intuitivno sledi, da več pojavitev *faset* znotraj rezultatov poizvedbe viša njegovo relevantnost, za katero hkrati velja, da s frekvenco znotraj baze pada. Nabor redkih *faset*, ki so reprezentativni ter torej najprimernejši za primerjave med vrsticami, se stalno vzdržuje in prilagaja.

3.3 Wrangler

Wrangler [8] je interaktiven sistem za upravljanje podatkovnih transformacij. Na vsakem koraku preiskovanja ima uporabnik pred seboj vizualizacijo podatkov. Wrangler avtomatsko izbere nabor relevantnih transformacij nad podatki in tako uporabniku ponudi naslednji korak. Orodje tako omogoča iterativno preiskovanje prostora veljavnih operacij in predogled njihovih učinkov na podatke.

Poglavje 4

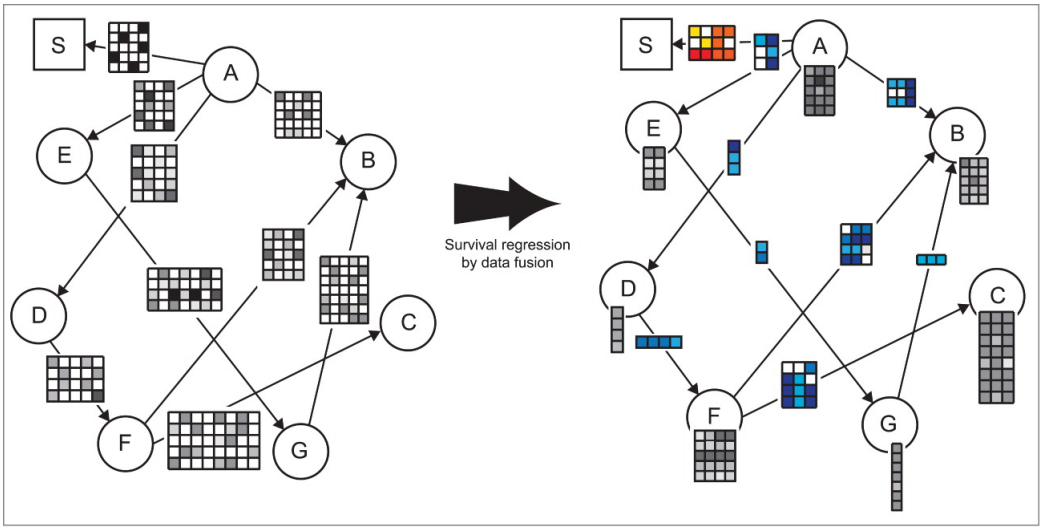
Predlagana metoda za rangiranje relacij

Osnovna mera, ki jo naš pristop uporablja za ocenjevanje stopnje povezanosti med pari entitetnih tipov, je ocena kvalitete prileganja modela DFMF matričnemu opisu podatkov v tarčni bazi. Začetni del poglavja zato namenimo predstavitvi metode Zlivanja podatkov z matričnim razcepom (DFMF) in njeni implementaciji v knjižnici *scikit-fusion*. Nadaljujemo z razlago, kako relacije znotraj baze opišemo z relacijskimi matrikami, kar je predpogoj za uporabo DFMF. Ker se je med razvojem pokazalo tudi nekaj problemov vezanih na prostorsko in računsko zahtevnost postopka, na koncu poglavja predstavimo, kako smo le-te reševali.

4.1 Zlivanje podatkov z matričnim razcepom

Namen orodja je ocenjevanje stopnje povezanosti med pari entitetnih tipov znotraj podatkovnih baz. Pri tem ima ključno vlogo uporaba postopka zlivanja podatkov z uporabo matričnega razcepa (DFMF), ki sta ga na Ljubljanski Fakulteti za računalništvo in informatiko razvila Žitnik in Zupan [10].

DFMF [10] spada med algoritme zlivanja podatkov, ki temeljijo na delni integraciji podatkov. Gre za najmlajšo vejo izmed tovrstnih algoritmov, za pripadnike katere je značilno, da ohranjajo strukturo vhodnih podatkov (torej jih ne združujejo kot postopki zgodnje oz. polne integracije), in razvijejo en skupen model. Prav v tem se razlikujejo od postopkov s pozno integracijo, ki pa razvijejo en model za vsak uporabljen podatkovni vir. Metode delne integracije tipično dosegajo višje napovedne točnosti kot pripadniki preostalih dveh pristopov.



Slika 4.1: Sestava podatkovnega grafa pred in po zlivanju s postopkom DFMF. Vozlišča v grafu predstavljajo entitetne tipe, matrike pa opis relacij. [11, Povzeto po sliki 1.]

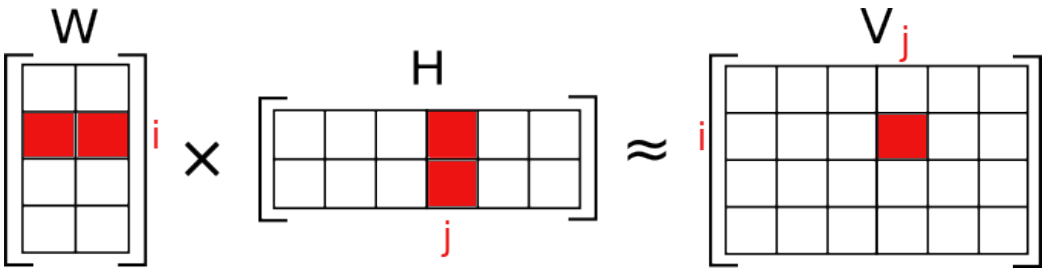
Algoritem DFMF na vhod sprejme zbirko objektnih tipov in relacijske matrike, ki opisujejo povezave med objektnimi tipi. Za izvedbo ni potrebno, da

podamo relacijo za vsak par objektnih tipov, pač pa je ključnega pomena zgolj to, da je graf, ki ga oblikujejo, povezan. Primer grafa je prikaza na sliki 4.1. Poseben tip relacijskih matrik so omejitvene matrike, ki opisujejo povezavo med objekti istega tipa.

Algoritem nato nad relacijskimi matrikami izvede postopek simultanega matričnega razcepa, ki pri regularizaciji upošteva matrike omejitev. Kot rezultat dobimo sistem, v katerem so tako vsi objektni tipi, kot tudi povezave med njimi, predstavljeni s svojimi faktorskimi matrikami. Napovedi relacije med poljubnim parom objektnih tipov dobimo z množenjem faktorskih matrik obeh objektnih tipov in povezovalne faktorske matrike.

4.1.1 Matrični razcep

Osnovna zamisel postopka matričnega razcepa je čim bolj zvesta predstavitev vhodne matrike s produktom dveh ali večih matrik nižjega ranga. Primer dvo-faktorskega razcepa matrike prikazuje slika 4.2.



Slika 4.2: Predstavitev vhodne podatkovne matrike V kot produkt dveh manjših matrik W in H, kot to poteka pri najosnovnejši različici matričnega razcepa (NMF). Vidimo lahko tudi, da napoved vrednosti celice prvotne matrike V v vrstici i in stolpcu j izračunamo kot produkt i-te vrstice matrike W in j-tega stolpca matrike H. [1, Povzeto po sliki.]

Postopek DFMF je bil razvit na osnovi postopka tri-faktorskega matričnega razcepa, ki sočasno razstavi vse relacijske matrike vhodnega grafa in vsako

predstavi kot produkt treh matrik: dve faktorski matriki predstavljata vsaka svoj objektni tip, ena pa predstavlja povezovalno preslikavo.

Napoved vrednosti, ki se v prvotni relacijski matriki nahaja v i-ti vrstici in j-tem stolpcu dobimo tako, da najprej izračunamo skalarni produkt i-te vrstice faktorske matrike prvega izmed objektnih tipov z linearno kombinacijo stolpcev matrike, ki predstavlja preslikovalno funkcijo med tipoma, obtežene z j-to vrstico faktorske matrike drugega objektnega tipa.

4.1.2 Programska knjižnica scikit-fusion

Uporabljamo implementacijo algoritma DFMF v knjižnici *scikit-fusion*, ki je objavljena na github repozitoriju dr. Marinke Žitnik [9].

Pred zlivanjem podatkov moramo najprej zgraditi graf zlivanja (ang. *fusion graph*). Najprej definiramo vse objektne tipe, ki so predstavljeni znotraj vira podatkov, ki ga bomo uporabili. V našem primeru je to tarčna podatkovna baza. S tem določimo vozlišča grafa. V nadaljevanju za vsako relacijsko matriko, ki smo jo zgradili nad podatki, označimo katerega izmed parov objektnih tipov povezuje. S tem določimo povezave v grafu zlivanja. Nad zgrajenim grafom izvedemo zlivanje z uporabo metode DFMF, ki objekte in relacije preslika v latentni prostor.

Omejitev, s katero smo se srečali pri uporabi knjižnice *scikit-fusion*, je da nam ta omogoča le dodajanje matrik za pare objektnih tipov, ne pa tudi dodajanje več-dimenzionalnih tenzorjev, s katerimi bi lahko opisovali tudi lastnosti skupne večim objektnim tipom. Relacijske podatkovne baze namreč vsebujejo tudi tabele z več kot dvema tujima ključema. Take tabele lahko torej med seboj povezujejo večje število objektnih tipov, relacijo med katerimi bi morali predstaviti s tenzorjem.

Tekom dela smo vpliv te omejitve na natančnost rekonstrukcije matrik po zlivanju poskusili vsaj delno omejiti tako, da v graf vključimo po eno matriko za vsakega od parov tujih ključev, preko katerih tabela povezuje objektne tipe.

4.2 Gradnja relacijskih matrik

Pred uporabo postopka DFMF nad podatkovno bazo, je potrebno zajeti relacije med tabelami v njej in jih predstaviti v obliki matrik. Relacijska matrika opisuje povezavo med parom objektnih tipov. Je torej matrika, vrstice katere predstavljajo objekte prvega, stolpci pa objekte drugega tipa, številska vrednost v posamezni celici pa vrednoti objektoma skupno lastnost. Kot primer bi lahko taka matrika povezovala paciente s predpisanimi zdravili, pri čemer bi vrednosti v celicah predstavljale količino nekega zdravila, ki je bila pacientu predpisana. Poseben tip relacijskih matrik so omejitvene matrike, ki opisujejo povezavo med objekti istega tipa. Taka omejitvena matrika bi na primer lahko opisovala interakcijo med zdravili.

Relacijske matrike, ki jih gradi naše orodje, v delimo v dve skupini:

- matrike zgrajene na podlagi neposredne povezave para tabel preko tujega ključa,
- matrike zgrajene na podlagi povezave para tabel preko vmesne tabele.

Matrike prvega tipa so indikatorske matrike. Vrednost 1 v njih označuje, da obstaja odvisnost med parom objektov različnih tipov iz pripadajočega stolpca in vrstice. S tem, ko zgradimo po eno indikatorsko matriko za vsako omejitev tipa *tuji ključ* hkrati zagotovimo povezan podatkovni graf.

V relacijski podatkovni bazi lahko entitetni tipi med seboj vstopajo v relacije preko tabel z večimi tujimi ključi (asociativnih entitet). Stolpce takih tabel lahko obravnavamo kot atributne opise relacij, ki jih tvorijo referencirane tabele. Tako bi lahko za vsako tovrstno relacijo zgradili po eno matriko za vsak tak stolpec. V resnici so za gradnjo v matrikah primerni zgolj stolpci z numeričnim podatkovnim tipom. Naš program samodejno poskrbi, da se tudi kategorične spremenljivke, ki sicer ne hranijo numeričnih tipov podatkov, pretvorijo v ustrezno obliko s postopkom tvorbe indikatorskih spremenljivk. Tak stolpec torej nadomestimo z množico novih, od katerih vsak predstavlja po eno izmed unikatnih vrednosti prvotnega stolpca.

Vsak zapis tabele bo imel vrednost 1 pri enem izmed novo-ustvarjenih atributov, pri ostalih pa 0. Ker lahko s tem postopkom število stolpcev in posledično matrik hitro zelo naraste, ima uporabnik možnost preko vhodnega parametra določiti najvišje število različnih vrednosti, ki jih lahko kategorični atribut še obsega, da se nad njim izvede postopek tvorbe indikatorskih spremenljivk.

Tak način gradnje relacijskih matrik lahko za katero izmed relacij pripravi tudi zelo veliko število matrik. Pri vsakem modelu DFMF, ki ga zgradimo, lahko v graf kot opis posamezne relacije med parom objektnih tipov vključimo zgolj po eno matriko. Če bi zgradili samo en model, bi torej zavrgli veliko informacije, ki jo nosijo različni opisi relacij oz. različne matrike.

Problem poskušamo omejiti tako, da za relacije tvorimo vse možne kombinacije izborov matrik. Na podlagi vsake take kombinacije nato zgradimo graf in po zlitju ocenimo vsako izmed relacij. Privzeto kot končni rezultat povprečimo ocene vsake izmed relacij čez vse zgrajene grafe, uporabnik pa lahko sicer tudi določi, da se za vsako od relacij upošteva zgolj najboljša izmed ocen oz. najnižja napaka RMSE.

Veliko število matrik torej pomeni še toliko večje število modelov DFMF, ki jih zgradimo. Gradnja vsakega modela je lahko časovno precej zahtevna. Uporabniku zato dopuščamo možnost, da preko vhodnega parametra določi omejitev števila matrik, ki naj se upoštevajo pri obravnavi posamezne relacije. Ob morebitni prekoračitvi števila matrik obdržimo zgolj najgostejše, torej, tiste z najvišjim razmerjem polnih celic proti praznim.

4.3 Obvladovanje prostorske in časovne zahtevnosti problema

Analiza velikih baz lahko zahteva veliko procesorskega časa in pomnilnika. S tem razlogom pri predlaganem postopku uporabniku ponujamo vsaj delen nadzor nad velikostjo obravnavanega problema. Poleg omenjenega nadzora nad številom obravnavanih relacijskih matrik za vsako izmed relacij lahko

uporabnik preko vhodnih parametrov poda tudi omejitev velikosti matrik in število obravnavanih objektnih tipov. Namen tega razdelka je, da posebej predstavimo še koraka vzorčenja zapisov v tabelah in obravnavanih objektnih tipov, ki sta ključna pri omejitvi velikosti problema.

4.3.1 Vzorčenje zapisov v tabelah

V podatkovnih bazah so tabele z velikim številom vnosov nekaj povsem običajnega. To lahko privede do problema, ko želimo ustvariti relacijsko matriko za objektna tipa (v bazi predstavljena s tabelama), za katera hranimo veliko število objektov (vrstice v tabeli), saj lahko velikost take matrike presega kapaciteto pomnilnika, s katero razpolagamo. Problem rešujemo tako, da pri obravnavi zelo velikih podatkovnih baz uporabniku ponudimo možnost stvaritve in obravnave manjše verzije podatkovne baze. Uporabnik ob tem določi tudi delež zapisov v tabelah, ki naj se prenesejo vanjo. Vrstice izbiramo naključno in spoštujoč referenčne odvisnosti med objekti različnih tabel. V ta namen uporabljamo Pythonov modul rdbms-subsetter [2].

4.3.2 Vzorčenje objektnih tipov

Število tabel se lahko od baze do baze zelo razlikuje. Veliko število teh za naš postopek pomeni gradnjo večjega števila relacijskih matrik in večji graf zlivanja. Oboje se pri izvajanju odraža na porabi pomnilnika in procesorskega časa. Uporabniku zato nudimo možnost določanja omejitve števila obravnavanih tabel. Uporabnik lahko tudi določi ime entitete, ki predstavlja središče njegovega preiskovanja in za katero zahteva, da se nahaja tudi v obravnavani podmnožici. Taka tabela predstavlja prvega izmed izborov obravnavane podmnožice. V primeru, da uporabnik entitete ne določi, kot prvi element izberemo tabelo z največjim številom povezav na ostale tabele. V vsakem naslednjem koraku v izbor dodamo novo tabelo, pri čemer se odločamo zgolj med tistimi, ki so povezane na katero od že izbranih tabel. S tem zagotovimo povezanost novega podgrafa. Med kandidati pri vsakem izboru izberemo ti-

stega, ki tvori najvišje število povezav z ostalimi tabelami.

Poglavje 5

Orodje

Končni rezultat tega diplomskega dela je Python programski modul [3], s katerim se je mogoče povezati na poljubno podatkovno bazo PostgreSQL in zgraditi urejen seznam relacij, ki se v njej nahajajo. Namen tega poglavja je predstavitev implementacije metode rangiranja relacij med objektnimi tipi v podatkovni bazi skozi uporabo orodja nad testno podatkovno bazo.

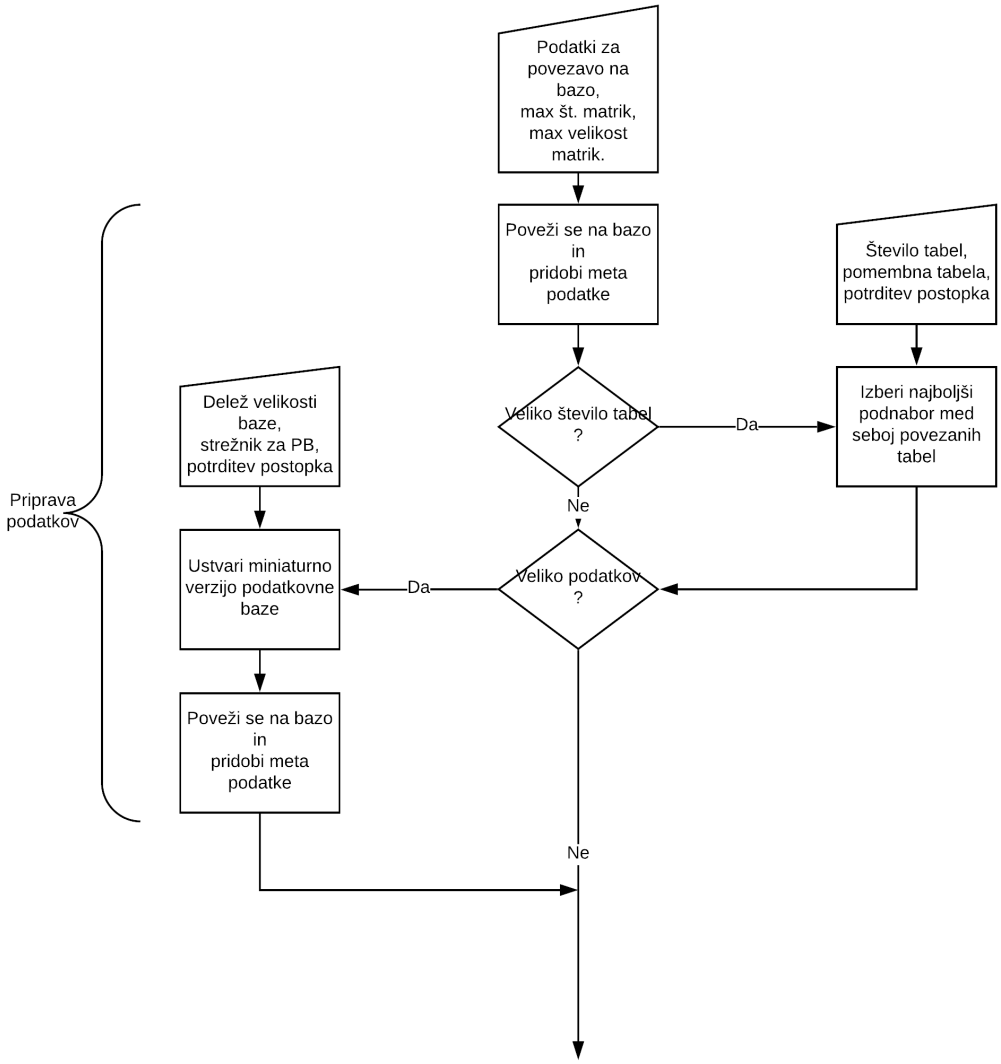
5.1 Predstavitev delovanja orodja

Uporabnik najprej ustvari nov objekt razreda FuseRDB in ob tem poda:

- podatke za dostop do podatkovne baze (naslov strežnika, uporabniško ime, geslo in ime podatkovne baze),
- število različnih vrednosti za kategorične spremenljivke, pri katerih naj se še izvaja delitev na množico indikatorskih spremenljivk (privzeto število vrednosti ni omejeno),
- velikost latentnega prostora matričnega razcepa (privzeta vrednost je 5 dimenzij),
- naj se rezultati različnih modelov za vsako relacijo povprečijo, ali naj se obdrži zgolj najboljše (privzeto se jih povpreči).

Sledijo trije glavni koraki:

1. **Priprava podatkov.**



Slika 5.1: Programski potek korakov priprave podatkov.

Programski potek koraka priprave podatkov prikazuje slika 5.1. V tem koraku se vzpostavi povezava s podatkovno bazo od koder se iz sistemskih tabel pridobi:

- seznam vseh podatkovnih tabel,

- seznam vseh omejitev tipov primarni in tuji ključ,
- seznam podatkovnih tipov stolpcev tabel,
- seznam identifikatorjev vrstic v tabelah.

```
Uporabnik@PC:~$> python3.5 FuseRDB.py
***Povezovanje na podatkovno bazo...
V bazi je veliko stevilo obravnavanih tabel (63), kar lahko mocno poveca
porabo pomnilnika in izvajalni cas.
zelite omejiti stevilo obravnavanih tabel? [DA/ne]:da
Koliko od trenutnih 63 tabel zelite obdrzati? [1-63]:12
Se mora katera od tabel nujno pojaviti v vzorcu?
    0     NE, vi izberite tabele
    1     stock
    2     phenotype_comparison
    3     cvterm_dbxref
    4     assay_biomaterial
    5     feature_cvterm
    6     feature_dbxref
    7     element
          ...
    57    project
    58    organism
    59    feature_genotype
    60    study_assay
    61    stock_dbxref
    62    synonym
    63    dbxref

[Vnesite stevilko pred izbiro]:0
Zaradi velikosti podatkovne baze lahko pride do prekoracitve porabe razpolozljivega
pomnilnika.
Zelite zmanjsati stevilo vrstic v tabelah s postopkom vzorcenja? [DA/ne]:da
Vnesite niz za povezavo na novo bazo ('connection string').
[postgresql://[user[:password]@[netloc][:port]/[database]]:
postgresql://postgres:geslo123@127.0.0.1/mini_parameciumdb
Vnesite delez podatkov, ki jih zelite obdrzati. [predlagano 3.228e-07]:0.00001
```

Slika 5.2: Začeten programski izpis, kjer je zaradi velikosti baze parameciumDB uporabniku ponujeno omejevanje števila obravnavanih tabel in pa vzorčenje podatkovne baze. Viden je tudi izpis ob začetku migracije podatkov v novo-ustvarjeno podatkovno bazo. Zaradi omejitve dolžine seznama so na sliki nekateri zapisi izpuščeni.

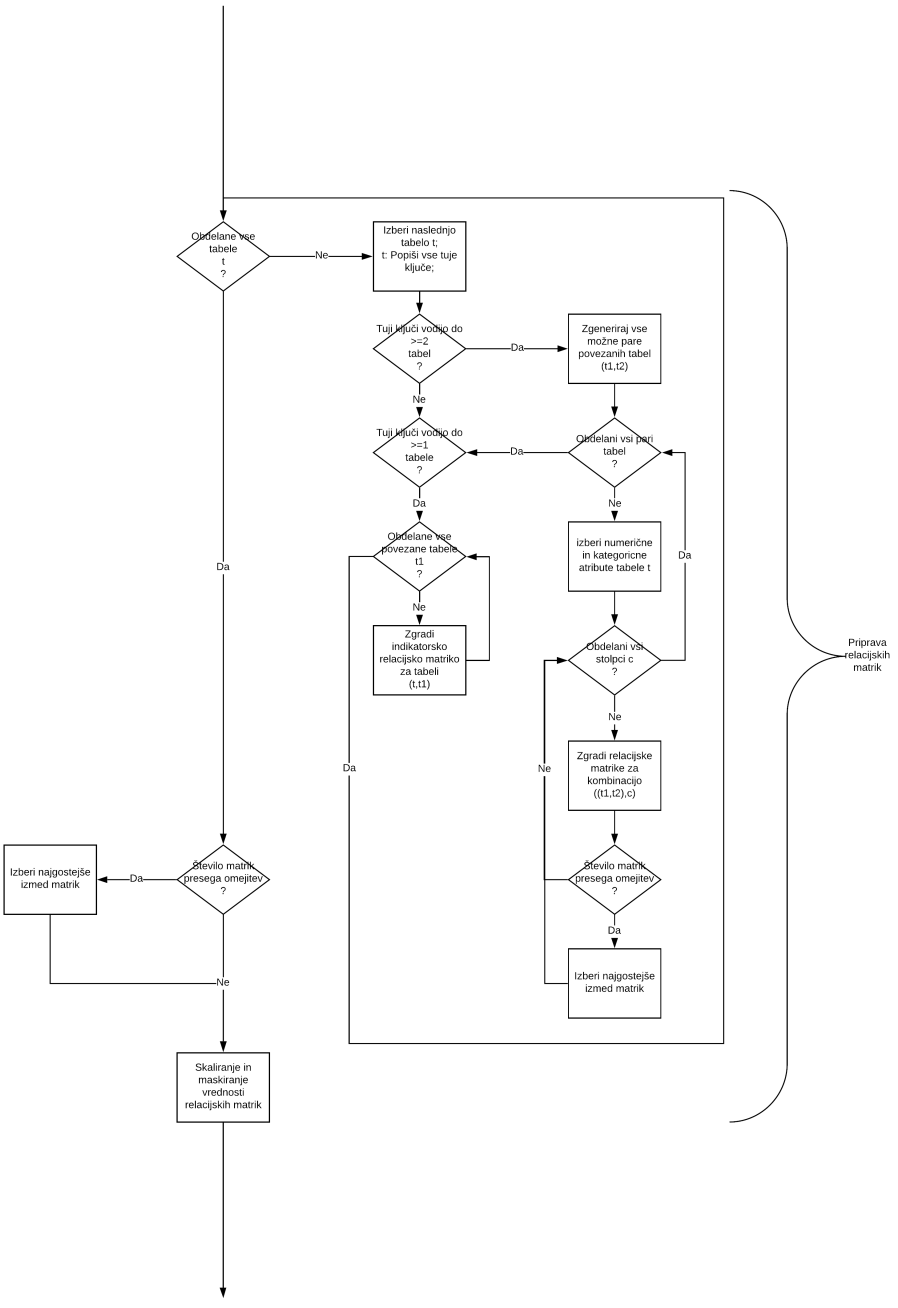
Če je v bazi veliko tabel, se uporabnika o tem obvesti in se mu predlaga zmanjšanje števila obravnavanih tabel. Uporabnik ob potrditvi

postopka poda tudi število tabel, ki naj jih obsega podnabor. Poda lahko tudi ime tabele, ki je ključna za uporabnikovo preiskovanje baze in za katero bi želel, da se nahaja tudi v novem izboru. Primer programskega izpisa prikazuje slika 5.2.

Izbor tabel se začne s tabelo, katere ime je podal uporabnik. Če te tabele ni določil, se kot prvi element novega nabora izbira tista izmed tabel, ki jo na ostale veže največje število povezav oz. tujih ključev. V vsaki iteraciji nato končni izbor razširimo z eno izmed tabel, ki so neposredno povezane s katero od že izbranih. Od kandidatov izberemo tabelo z največjim številom tujih ključev. Postopek ponavljamo, dokler izbor ne obsega števila tabel, ki ga je podal uporabnik.

Preverimo tudi, če je število zapisov v tabelah podatkovne baze veliko. V tem primeru uporabniku ponudimo možnost stvaritve nove podatkovne baze, ki predstavlja pomanjšano oz. vzorčeno verzijo prvotne. Če uporabnik postopek potrdi, mora hkrati podati tudi podatke potrebne za povezavo na podatkovni strežnik PostgreSQL, kjer naj se nova baza ustvari, in pa delež podatkov oz. zapisov v tabelah, ki naj se v njej hrani. Uporabniku program predlaga kakšen delež naj izbere, da velikost nobene matrike ne bo presegla želene velikosti (privzeto 1000000 celic). Po tem, ko na strežniku ustvarimo novo bazo, vanjo izvozimo podatkovno shemo iz prvotne baze. Nato se z uporabo modula rdbms-subsetter v novo bazo uvozi tudi zapise tabel. Vrstice se izbira naključno, spoštujoč referenčne omejitve, ki veljajo med objekti različnih tabel.

2. **Gradnja relacijskih matrik.**



Slika 5.3: Programski potek korakov gradnje relacijskih matrik.

```
***Povezovanje na podatkovno bazo...
***Gradnja relacijskih in omejitvenih matrik..
  ( 1 / 12 )    Gradim relacijske matrike za tabelo:  protocol
      ( 1 / 496 )       Gradim relacijske matrike za povezavi:
        ('$2', 'protocol', 'dbxref') in ('$2', 'protocol', 'protocol')
          ( 1 / 9 )     Gradim relacijske matrike stolpec: protocol_id
              Stolpec predstavlja kljuc (primarni ali tuji) in zato
              ni primeren za gradnjo matrik.
          ( 2 / 9 )     Gradim relacijske matrike stolpec: pub_id
              Stolpec predstavlja kljuc (primarni ali tuji) in zato
              ni primeren za gradnjo matrik.
          ( 3 / 9 )     Gradim relacijske matrike stolpec: protocoldescription
              Poskusam ustvariti prazno matriko velikosti: 765  X  8
          ( 4 / 9 )     Gradim relacijske matrike stolpec: uri
              Poskusam ustvariti prazno matriko velikosti: 765  X  8
          ( 5 / 9 )     Gradim relacijske matrike stolpec: name
              Poskusam ustvariti prazno matriko velikosti: 765  X  8
          ( 6 / 9 )     Gradim relacijske matrike stolpec: hardwaredescription
              Poskusam ustvariti prazno matriko velikosti: 765  X  8
          ( 7 / 9 )     Gradim relacijske matrike stolpec: type_id
              Stolpec predstavlja kljuc (primarni ali tuji) in zato
              ni primeren za gradnjo matrik.
          ( 8 / 9 )     Gradim relacijske matrike stolpec: softwaredescription
              Poskusam ustvariti prazno matriko velikosti: 765  X  8
          ( 9 / 9 )     Gradim relacijske matrike stolpec: dbxref_id
              Stolpec predstavlja kljuc (primarni ali tuji) in zato
              ni primeren za gradnjo matrik.
      ( 2 / 496 )       Gradim relacijske matrike za povezavi:
        ('$2', 'protocol', 'dbxref') in ('$2', 'protocol', 'quantification')
          ( 1 / 9 )     Gradim relacijske matrike stolpec: protocol_id
              Stolpec predstavlja kljuc (primarni ali tuji) in zato
              ni primeren za gradnjo matrik.
          ( 2 / 9 )     Gradim relacijske matrike stolpec: pub_id
              Stolpec predstavlja kljuc (primarni ali tuji) in zato
              ni primeren za gradnjo matrik.
          ( 3 / 9 )     Gradim relacijske matrike stolpec: protocoldescription
              Poskusam ustvariti prazno matriko velikosti: 765  X  80
          ...
    ...
```

Slika 5.4: Programski izpis ob začetku gradnje relacijskih matrik na primeru izvajanja programa nad podatkovno bazo parameciumDB.

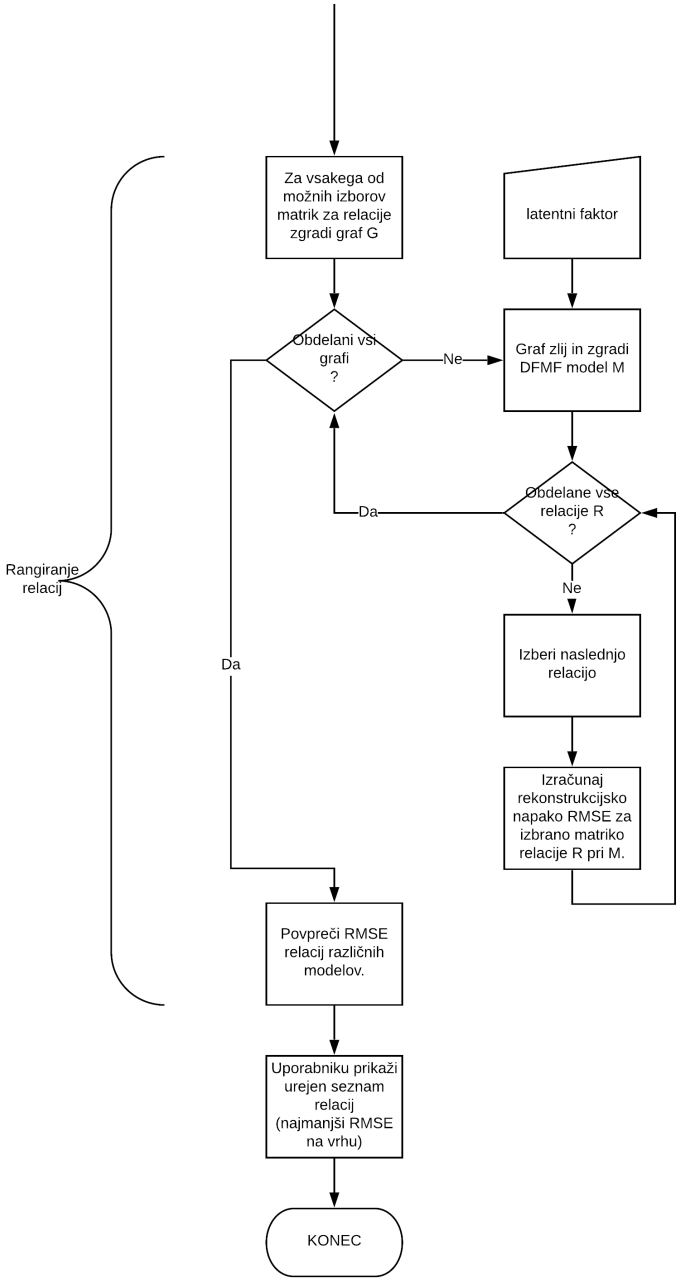
Najprej se sprehodimo čez vse tabele z dvema ali večimi tujimi ključi. Za vsak par entitetnih tipov, ki jih taka tabela povezuje, lahko zgradimo po eno relacijsko matriko za vsakega izmed njenih stolpcev, ki hrani numerični podatkovni tip. Primer programskega izpisa, ki nastane v tem koraku prikazuje slika 5.4. V primeru, da stolpec hrani kategorične vrednosti, ga najprej prevedemo na množico indikatorskih

spremenljivk. Ker imajo določeni atributi lahko zelo veliko število kategorij, ima uporabnik možnost določiti največje število kategorij, ki jih še lahko ima nek stolpec, če naj se pretvori na množico indikatorskih spremenljivk. Programski potek koraka gradnje matrik prikazuje slika 5.3.

Zahteva algoritma DFMF, da je graf povezan - obstaja povezava med poljubnim parom entitetnih tipov - s prej omenjenim načinom gradnje matrik ni zagotovljena. Da pogoju gotovo zadostimo, dodatno zgradimo po eno indikatorsko matriko za vsak par neposredno povezanih tabel. Indikatorska matrika ima vrednost 1 na mestu, kjer se objekta iz pripadajoče vrstice in stolpca matrike sklicujeta drug na drugega.

Vrednosti vsake od relacijskih matrik pred uporabo lestvičimo na interval med 0 in 1. S tem preprečimo, da bi se model prilagodil kateri od relacij v večji meri kot ostalim zgolj zaradi višjih vrednosti elementov matrike. Hkrati tudi maskiramo manjkajoče vrednosti, saj od modela ne želimo, da se jim poskuša prilagoditi.

3. **Gradnja modelov DFMF in ocenjevanje relacij na podlagi rekonstrukcijske napake matrik.**



Slika 5.5: Programski potek korakov rangiranja relacij.

```
***Zlivanje podatkov..
  ( 1 / 1 ) Pripravljam graf..
  ..zlivam graf..
      ( 1 / 78 ) ocenjujem rekonstrukcijo relacije: ('quantification', 'stock')
      ( 2 / 78 ) ocenjujem rekonstrukcijo relacije: ('pub', 'acquisition')
      ( 3 / 78 ) ocenjujem rekonstrukcijo relacije: ('acquisition', 'contact')
      ( 4 / 78 ) ocenjujem rekonstrukcijo relacije: ('protocol', 'protocol')
          Preskocil relacijo med enakima objektnima tipoma
      ( 5 / 78 ) ocenjujem rekonstrukcijo relacije: ('acquisition', 'acquisition')
          Preskocil relacijo med enakima objektnima tipoma
      ( 6 / 78 ) ocenjujem rekonstrukcijo relacije: ('arraydesign', 'quantification')
      ( 7 / 78 ) ocenjujem rekonstrukcijo relacije: ('feature', 'feature')
      ...
```

Slika 5.6: Programski izpis ob pripravi in zlitju grafa ter merjenju rekonstrukcijske napake relacij na primeru izvajanja programa nad podatkovno bazo parameciumDB.

Programski potek korakov gradnje modelov DFMF in rangiranja relacij prikazuje slika 5.5. Na tej točki imamo povezan graf, kjer imamo vsako povezavo predstavljeno z najmanj eno relacijsko matriko. Vsaka od pripravljenih matrik opisuje enega izmed načinov kako vrstice povezanih entitetnih tipov vstopajo v medsebojno relacijo. Model DFMF za opis posamezne povezave sprejme zgolj po eno relacijsko matriko. Posamezni model upošteva zelo majhen del razpoložljive informacije o relacijah. Vpliv te omejitve poskušamo zmanjšati tako, da zgradimo več modelov - po enega za vsakega izmed različnih možnih izborov relacijskih matrik za povezave v grafu. Programski izpis, ki nastane ob zlitju takega grafa in ob ocenjevanju rekonstrukcije vsebovanih relacij, prikazuje slika 5.6. Rezultate različnih modelov na koncu združimo s povprečenjem ali izbiro najboljših.

Ker je pri nekaterih povezavah število alternativnih matrik zelo veliko, je število modelov, ki jih je potrebno zgraditi, tako še mnogo večje. To seveda ponovno podaljša izvajalni čas in poveča porabo pomnilnika. Uporabnik ima zato možnost preko vhodnega parametra podati

omejitev števila alternativnih relacijskih matrik, s katerimi je posamezna povezava lahko predstavljena. Glede na podano številsko omejitev odberemo zgolj najgostejše izmed matrik.

```
RANGIRAN SEZNAM RELACIJ:
   1. ('pub', 'acquisition') RMSE: 0.0
   2. ('acquisition', 'dbxref') RMSE: 0.00017089995956527307
   3. ('contact', 'dbxref') RMSE: 0.00025837887362275
   4. ('pub', 'arraydesign') RMSE: 0.0002622941465080095
   5. ('acquisition', 'feature') RMSE: 0.0003255948216986513
   6. ('feature', 'quantification') RMSE: 0.0004692597911641465
   7. ('protocol', 'dbxref') RMSE: 0.000926053113242821
   8. ('assay', 'contact') RMSE: 0.0011347559349784363
   9. ('quantification', 'stock') RMSE: 0.0012185844331132618
  10. ('pub', 'contact') RMSE: 0.0012360582468341407
  11. ('biomaterial', 'pub') RMSE: 0.0014187123149998556
  12. ('biomaterial', 'dbxref') RMSE: 0.0019563405481843143
  13. ('pub', 'dbxref') RMSE: 0.0020979810541968736
  14. ('protocol', 'contact') RMSE: 0.0021761127971508315
      ...
  53. ('arraydesign', 'assay') RMSE: 0.030620513455066778
  54. ('protocol', 'pub') RMSE: 0.03497108859244497
  55. ('pub', 'quantification') RMSE: 0.0352884020501903
  56. ('contact', 'stock') RMSE: 0.036726434621152666
  57. ('acquisition', 'stock') RMSE: 0.05862938100683658
  58. ('feature', 'contact') RMSE: 0.07478047280723861
  59. ('biomaterial', 'protocol') RMSE: 0.07650793740831364
  60. ('protocol', 'cvterm') RMSE: 0.07731645464267718
  61. ('feature', 'stock') RMSE: 0.08262754980334999
  62. ('assay', 'stock') RMSE: 0.08363575335030676
  63. ('protocol', 'arraydesign') RMSE: 0.11448524947929081
  64. ('arraydesign', 'quantification') RMSE: 0.11622659224524415
  65. ('biomaterial', 'cvterm')  RMSE: 0.11885106191764255
  66. ('arraydesign', 'cvterm')  RMSE: 0.1516894138930169
===== Postopek je trajal:        0:15:21.933945
```

Slika 5.7: Programski izpis končnega seznama relacij na primeru izvajanja programa nad podatkovno bazo parameciumDB. Zaradi omejitve dolžine seznama so izpuščeni nekateri zapisi.

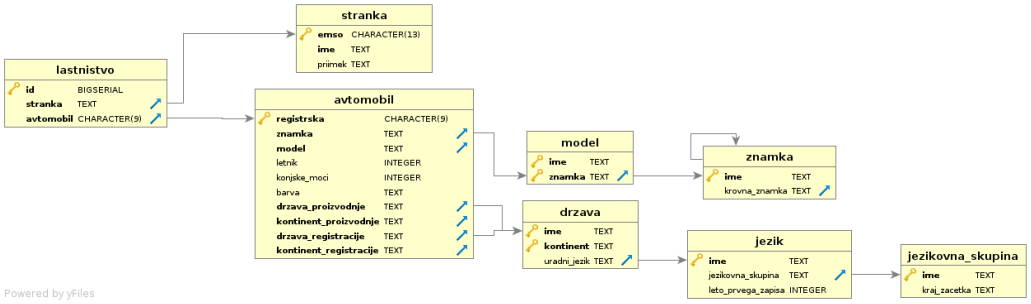
Z vsakim od zgrajenih modelov DFMF generiramo še napovedne matrike za vsako izmed relacij med entitetnimi tipi in jih primerjamo s pripadajočimi vhodnimi matrikami. Uporabniku se nato izpiše urejen seznam imen relacij entitetnih tipov, kjer se na vrhu nahajajo relacije z najmanjšimi rekonstrukcijskimi napakami (RMSE), proti dnu pa tiste, katerim se modeli v splošnem niso uspeli dobro prilagoditi.

5.2 Primer uporabe orodja

Namen tega odseka naloge je predstavitev korakov izvajanja programa. V ta namen se z orodjem povežemo na zelo majhno podatkovno bazo, kjer imamo lažji pregled nad posameznimi koraki izvajanja.

5.2.1 Testna baza

V namen prikaza delovanja orodja smo oblikovali preprosto podatkovno bazo z majhnim številom zapisov v tabelah in dovolj raznoliko strukturo, da lahko na njej prikažemo nekaj ključnih scenarijev delovanja.



Slika 5.8: Diagram zgradbe testne podatkovne baze 'avtomobilizem'.

Podatkovna baza obsega sledeče tabele:

- '**avtomobil**' s stolpci: 'registrska' (primarni ključ), 'znamka' (skupaj z atributom 'model' tvori tuji ključ), 'model' (skupaj z atributom 'znamka' tvori tuji ključ), 'letnik', 'konjske_moci', 'barva', 'drzava_proizvodnje' (skupaj z atributom 'kontinent_proizvodnje' tvori tuji ključ), 'kontinent_proizvodnje' (skupaj z atributom 'drzava_proizvodnje' tvori tuji ključ), 'drzava_registracije' (skupaj z atributom 'kontinent_registracije' tvori tuji ključ), 'kontinent_registracije' (skupaj z atributom 'drzava_proizvodnje' tvori tuji ključ),
- '**znamka**' s stolpci: 'ime' (primarni ključ), 'krovna_znamka' (tuji ključ),

- '**model**' s stolpci: 'ime' (primarni ključ), 'znamka' (tuji ključ),
- '**stranka**' s stolpci: 'emso' (primarni ključ), 'ime', 'priimek',
- '**lastnistvo**' s stolpci: 'id' (primarni ključ), 'stranka' (tuji ključ), 'avtomobil' (tuji ključ),
- '**drzava**' s stolpci: 'ime' (skupaj z atributom 'kontinent' tvori primarni ključ), 'kontinent' (skupaj z atributom 'ime' tvori primarni ključ), 'uradni_jezik' (tuji ključ),
- '**jezik**' s stolpci: 'ime' (primarni ključ), 'jezikovna_skupina' (tuji ključ), 'leto_prvega_zapisa'.
- '**jezikovna_skupina**' s stolpci: 'ime' (primarni ključ), 'kraj_zacetka.'

5.2.2 Vhodni parametri

V primeru, skozi katerega bi želeli prikazati uporabo orodja, smo programu poleg podatkov potrebnih za povezavo na podatkovni strežnik nastavili tudi naslednje vrednosti parametrov:

- število različnih vrednosti za kategorične spremenljivke, pri katerih naj se še izvaja delitev na več indikatorskih spremenljivk, nastavljeno na 4,
- postopka vzorčenja objektov nismo uporabili,
- števila obravnavanih entitetnih tipov nismo omejevali,
- nismo podali imen entitetnih tipov, ki bi nas posebej zanimali in bi želeli glede na njih filtrirati končni seznam relacij,
- števila alternativnih matrik, ki jih program za posamezno relacijo še upošteva pri gradnji modelov, nismo omejevali,
- velikosti matrik nismo omejevali,
- rezultate različnih modelov smo združevali s povprečenjem.

5.2.3 Primer izvajanja

Orodje v podatkovni bazi prepozna naslednje entitetne tipe:

- 'drzava',
- 'jezik',
- 'znamka',
- 'model',
- 'jezikovna_skupina',
- 'stranka',
- 'avtomobil',
- 'lastnistvo'.

Prešteje tudi število tujih ključev, ki jih posamezna tabela vsebuje. Tako prepozna 'lastnistvo' in 'avtomobil' kot tabeli, ki povezujeta več entitetnih tipov, saj imata 2, oz. v primeru tabele 'avtomobil' 3, tuje ključe. Ker je podatkovna baza majhna, nam program ne predlaga omejevanja števila obravnavanih tabel, niti omejevanja števila zapisov v tabelah. Iterira čez vse tabele, ki vsebujejo vsaj en tuji ključ: 'lastnistvo', 'avtomobil', 'model', 'znamka', 'drzava'. Za vsako od njih popiše tabele, na katere jih tuji ključ veže. Za vsakega od tujih ključev program zgradi po eno indikatorsko relacijsko matriko za predstavitev relacije med objektnima tipoma, katera povezuje. V našem primeru se tako zgradijo indikatorske matrike za relacije:

- 'lastnistvo - stranka',
- 'lastnistvo - avtomobil',
- 'avtomobil - model',
- 'avtomobil - drzava' (2 indikatorski matriki),
- 'model - znamka',
- 'znamka - znamka',
- 'drzava - jezik',
- 'jezik - jezikovna_skupina'.

Za tabele, ki vsebujejo dva ali več tujih ključev (v našem primeru sta to tabeli 'avtomobil' in 'lastništvo'), program dodatno zgradi tudi relacijske

matrike za vsakega izmed parov tujih ključev znotraj tabele. Matrike se ustvarijo za vsakega izmed stolpcev trenutno obravnavane tabele in sicer:

- Če gre za kategorično spremenljivko, se stolpec prevede na množico indikatorskih spremenljivk, katere moč je enaka številu unikatnih vrednosti.

- Če gre za numerično vrednost, se za vsak par objektov iz povezanih tabel popiše množica vrednosti obravnavanega atributa preko katerega sta povezana. Ker lahko pri gradnji posamezne matrike za par objektov izberemo zgolj eno tako vrednost, se v nadaljevanju zgradi po eno matriko za vsakega od možnih izborov vrednosti različnih parov objektov.

V našem primeru se tako zgradijo matrike zgolj za relacijo 'drzava - model'. V rezultatu ni matrik za relacijo 'avtomobil - stranka,' ker v tabeli 'lastnistvo', ki ju povezuje, ni primernih stolpcev. Vsak izmed stolpcev namreč predstavlja bodisi primarni bodisi tuji ključ.

	registrska [PK] character(9)	znamka text	model text	letnik integer	konjske_moci integer	barva text	drzava_proizvodnje text	kontinent_proizvodnje text	drzava_registracije text	kontinent_registracije text
1	LJ-111-11	VW	GOLF	2009	122	rumena	Nemcija	Evropa	Slovenija	Evropa
2	LJ-222-22	VW	GOLF	2006	140	rdeca	Nemcija	Evropa	Slovenija	Evropa
3	LJ-333-33	FORD	MONDEO	2006	170	modra	USA	Amerika	Slovenija	Evropa
4	LJ-368-43	VW	GOLF	2010	210	rdeca	Nemcija	Evropa	Slovenija	Evropa
5	LJ-444-44	FORD	FIESTA	2011	120	rumena	USA	Amerika	Slovenija	Evropa
6	LJ-555-55	AUDI	3	2012	105	bela	Nemcija	Evropa	Slovenija	Evropa
*										

Slika 5.9: Zapisi v tabeli 'avtomobil' testne baze.

Poglejmo, katere relacijske matrike se zgradijo med obravnavo tabele 'avtomobil', prikazana na sliki 5.9. Ker poleg tega, da tabela vsebuje tuji ključ, velja tudi, da predstavlja entitetni tip, najprej zgradimo indikatorske matrike. Tako zgradimo eno relacijsko matriko za relacijo 'avtomobil - model' in dve matriki za relacijo 'avtomobil - drzava', saj par tabel, ki nastopata v slednji, povezujeta dva tuja ključa ('drzava_proizvodnje' in 'drzava_registracije'). Zgradimo torej naslednje indikatorske relacijske matrike:

- Relacija 'avtomobil - model':

$$\begin{bmatrix} \_ & \_ & \_ & \_ & 1 & \_ & \_ & \_ & \_ \\ 1 & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ \\ \_ & \_ & \_ & \_ & 1 & \_ & \_ & \_ & \_ \\ \_ & \_ & \_ & \_ & \_ & 1 & \_ & \_ & \_ \\ \_ & \_ & 1 & \_ & \_ & \_ & \_ & \_ & \_ \\ \_ & \_ & \_ & \_ & 1 & \_ & \_ & \_ & \_ \end{bmatrix}$$

- Relacija 'avtomobil - drzava':
  - tuji ključ 'drzava_proizvodnje':

$$\begin{bmatrix} \_ & \_ & \_ & \_ & 1 \\ \_ & \_ & \_ & \_ & 1 \\ \_ & \_ & \_ & \_ & 1 \\ \_ & \_ & \_ & \_ & 1 \\ \_ & \_ & \_ & \_ & 1 \\ \_ & \_ & \_ & \_ & 1 \end{bmatrix}$$

  - tuji ključ 'drzava_registracije':

$$\begin{bmatrix} \_ & 1 & \_ & \_ & \_ \\ \_ & 1 & \_ & \_ & \_ \\ \_ & 1 & \_ & \_ & \_ \\ \_ & \_ & 1 & \_ & \_ \\ \_ & \_ & 1 & \_ & \_ \\ \_ & 1 & \_ & \_ & \_ \end{bmatrix}$$

Ker velja tudi, da ima tabela 'avtomobil' tujih ključev več (tri tuje ključe), zgradimo tudi po eno relacijsko matriko za vsakega izmed parov entitetnih tipov, na katere se ti ključi navezujejo. V primeru tabele 'avtomobil' se tako tvori par povezav 'drzava - model'. Za to relacijo torej tvorimo po eno relacijsko matriko za vsak stolpec znotraj obravnavane tabele 'avtomobil', ki predstavlja številsko vrednost ali kategorijo. Stolpec tudi ne sme nastopati v vlogi katerega izmed ključev (niti tujega niti primarnega). Taki stolpci so:

- letnik,
- konjske_moci,
- barva_jeRdeca,
- barva_jeRumena,
- barva_jeModra,
- barva_jeBela.

Tu velja opozoriti, da je zadnje štiri stolpce ustvaril program sam na podlagi stolpca 'barva'. Pretvorbo se izvede, saj je 'barva' kategorična spremenljivka, število unikatnih vrednosti katere pa ne presega podane omejitve štirih različnih vrednosti.

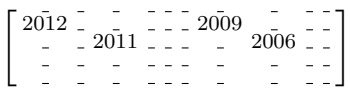
Tabela 'model' je z 'avtomobil' povezana preko enega, tabela 'drzava' pa preko dveh tujih ključev ('drzava_proizvodnje' in 'drzava_registracije'). Ker

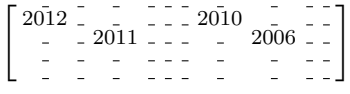
lahko vsak par tujih ključev objekte različnih tipov povezuje na drugačen način, moramo za vsakega zgraditi ločene matrike. V primeru indikatorskih spremenljivk izpeljanih iz atributa 'barva' to pomeni gradnjo 8 matrik (za vsakega od 2 tujih ključev po 4 indikatorske spremenljivke).

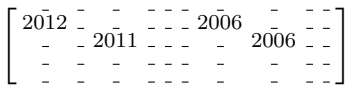
Pri obravnavi numeričnih spremenljivk 'letnik' in 'konjske_moci' moramo upoštevati, da med podatki Volkswagnov model 'Golf' z vsako od držav 'Slovenija' (preko tujega ključa 'drzava_registracije') in 'Nemčija' (preko tujega ključa 'drzava_proizvodnje') povezujejo trije avtomobili različnih letnikov in različnih konjskih moči. Spremenljivki med obravnavo zato zahtevata gradnjo dodatnih 12 relacijskih matrik (za vsako od spremenljivk in vsakega od tujih ključev po 3 matrike - po ena za vsak možen izbor vrednosti).

Skupaj pri obravnavi tabele 'avtomobil' torej nastane sledečih 20 relacijskih matrik:

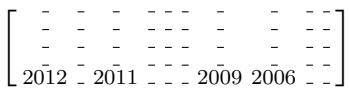
- relacija 'drzava - model' za stolpec 'letnik':
  - tuji ključ 'drzava_proizvodnje':

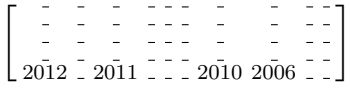


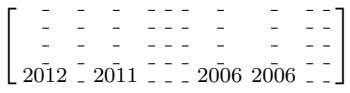




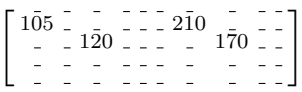
  - tuji ključ 'drzava_registracije':

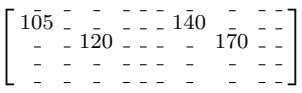


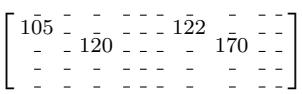




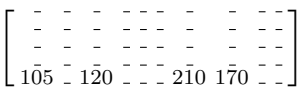
- relacija 'drzava - model' za stolpec 'konjske_moci':
  - tuji ključ 'drzava_proizvodnje':

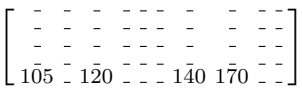


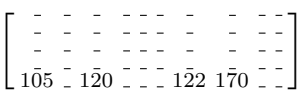




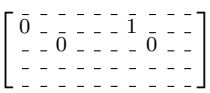
  - tuji ključ 'drzava_registracije':



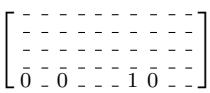




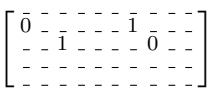
- relacija 'drzava - model' za stolpec 'barva_jeRdeca':
  - tuji ključ 'drzava_proizvodnje':



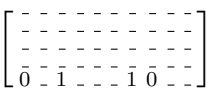
  - tuji ključ 'drzava_registracije':



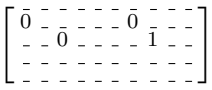
- relacija 'drzava - model' za stolpec 'barva_jeRumena':
  - tuji ključ 'drzava_proizvodnje':



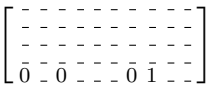
  - tuji ključ 'drzava_registracije':



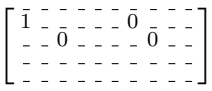
- relacija 'drzava - model' za stolpec 'barva_jeModra':
  - tuji ključ 'drzava_proizvodnje':



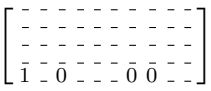
  - tuji ključ 'drzava_registracije':



- relacija 'drzava - model' za stolpec 'barva_jeBela':
  - tuji ključ 'drzava_proizvodnje':



  - tuji ključ 'drzava_registracije':



Ko obdelamo še ostale tabele, skupaj zgradimo:

- 2 matriki za relacijo 'avtomobil - drzava',
- 1 matriko za relacijo 'avtomobil - model',
- 20 matrik za relacijo 'drzava - drzava',
- 20 matrik za relacijo 'drzava - model',
- 1 matrika za relacijo 'drzava - jezik',
- 1 matrika za relacijo 'lastnistvo - avtomobil',
- 1 matrika za relacijo 'lastnistvo - stranka',
- 1 matrika za relacijo 'model - znamka',
- 1 matrika za relacijo 'znamka - znamka',
- 1 matrika za relacijo 'jezik - jezikovna_skupina'.

Vrednosti znotraj vsake od matrik lestvičimo na interval med 0 in 1. Tako preprečimo, da bi se model bolj prilagodil kateri izmed relacij zgolj zaradi višjih vrednosti. Ravno tako se maskirajo manjkajoče vrednosti.

V nadaljevanju se za vsako od možnih izborov relacijskih matrik za relacije zgradi po en graf in se ga zlije s postopkom DFMF. Skupaj se tako tvori 800 modelov DFMF (20 * 20 * 2). Z vsakim od modelov še napovemo vsako od relacij in tako dobimo matrike, ki jih lahko primerjamo s tistimi, ki smo jih za posamezno relacijo vključili v graf pred postopkom zlivanja. Za vrednotenje odstopanja med matriko napovedi in vhodno matriko uporabimo mero napake RMSE.

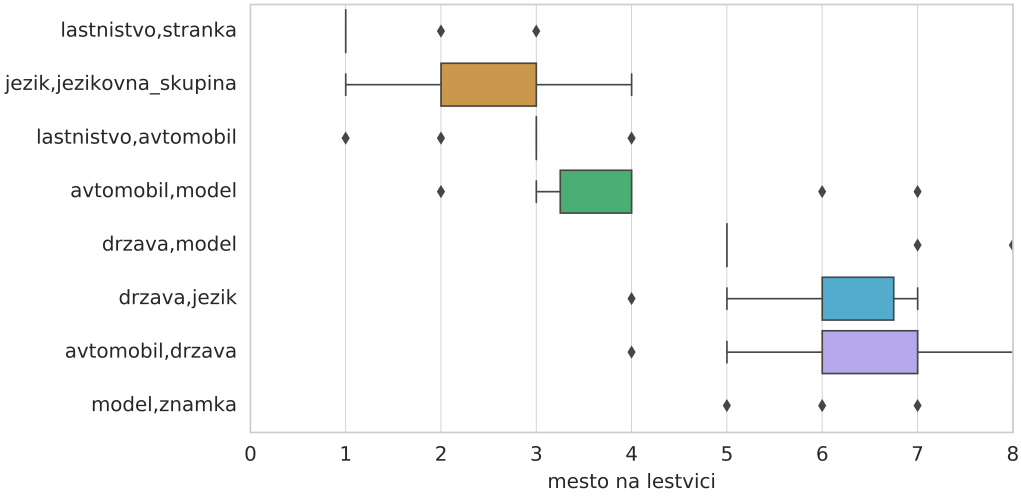
Kot zadnji korak se napako za vsako relacijo povpreči preko vseh zgrajenih modelov. Nato se uporabniku izpiše rangiran seznam relacij, na katerem so relacije z najnižjo rekonstrukcijsko napako prikazane na vrhu. V našem primeru je končni seznam, ki se uporabniku prikaže, sledeč:

1. 'lastnistvo - stranka' (RMSE: 6.67972253145e-17),
2. 'jezik - jezikovna_skupina' (RMSE: 1.77203815873e-05),
3. 'lastnistvo - avtomobil' (RMSE: 5.77513910573e-05),
4. 'avtomobil - model' (RMSE: 0.00140092935334),
5. 'drzava - jezik' (RMSE: 0.00181387759263),
6. 'avtomobil - drzava' (RMSE: 0.00211329479469),
7. 'model - znamka' (RMSE: 0.00295505230647),
8. 'drzava - model' (RMSE: 0.0777218610922).

Celoten postopek se je na sistemu z Intelovim i7 4510u procesorjem in 16 GB RAM pomnilnika izvajal 1 uro in 21 minut.

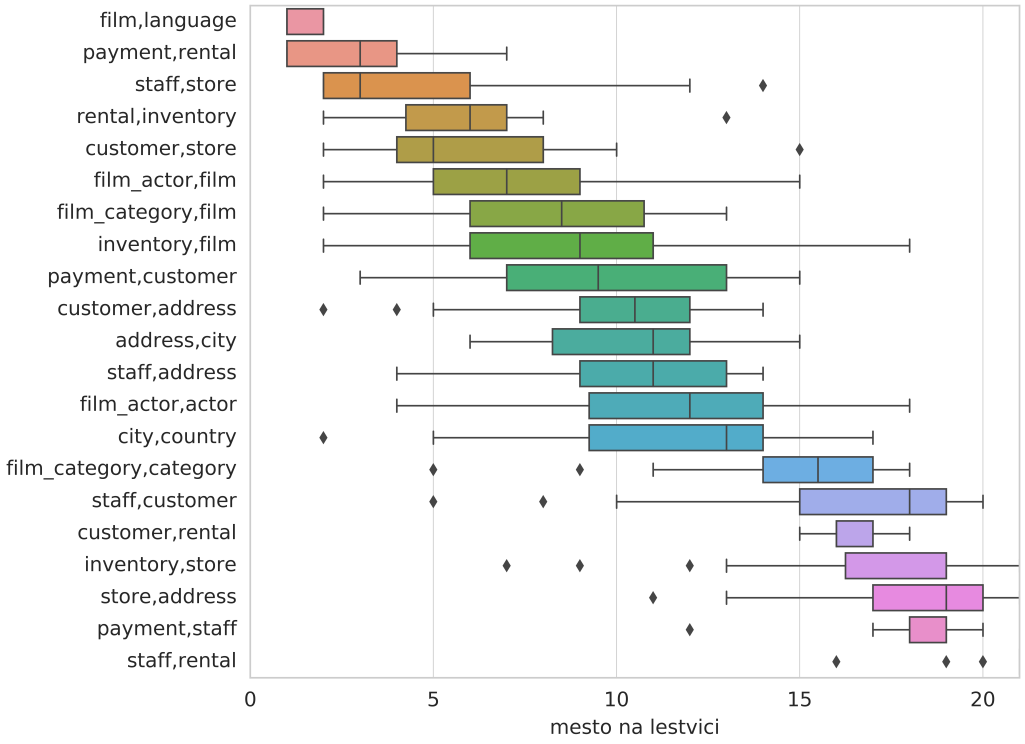
5.3 Ponovljivost rangiranja

Ponovljivost uvrstitve posamezne relacije podatkovne baze v končni seznam smo ovrednotili z izvajanjem 30 ponovitev nad vsako izmed baz avtomobilizem, pagila in nad pomanjšano različico baze parameciumDB. Velikost latentnega prostora smo pustili na privzeti vrednosti, kot jo za postopek DFMF določi knjižnica *scikit-fusion*. Ker je ta vrednost privzeto nastavljena na konstanto 5 za vse objektne tipe, menimo, da bi bolj dinamičen pristop izbire velikosti latentnega prostora, glede na število primerkov vsakega izmed objektnih tipov, bistveno pripomogel k ponovljivosti rezultatov. Predvsem pri večjih bazah.



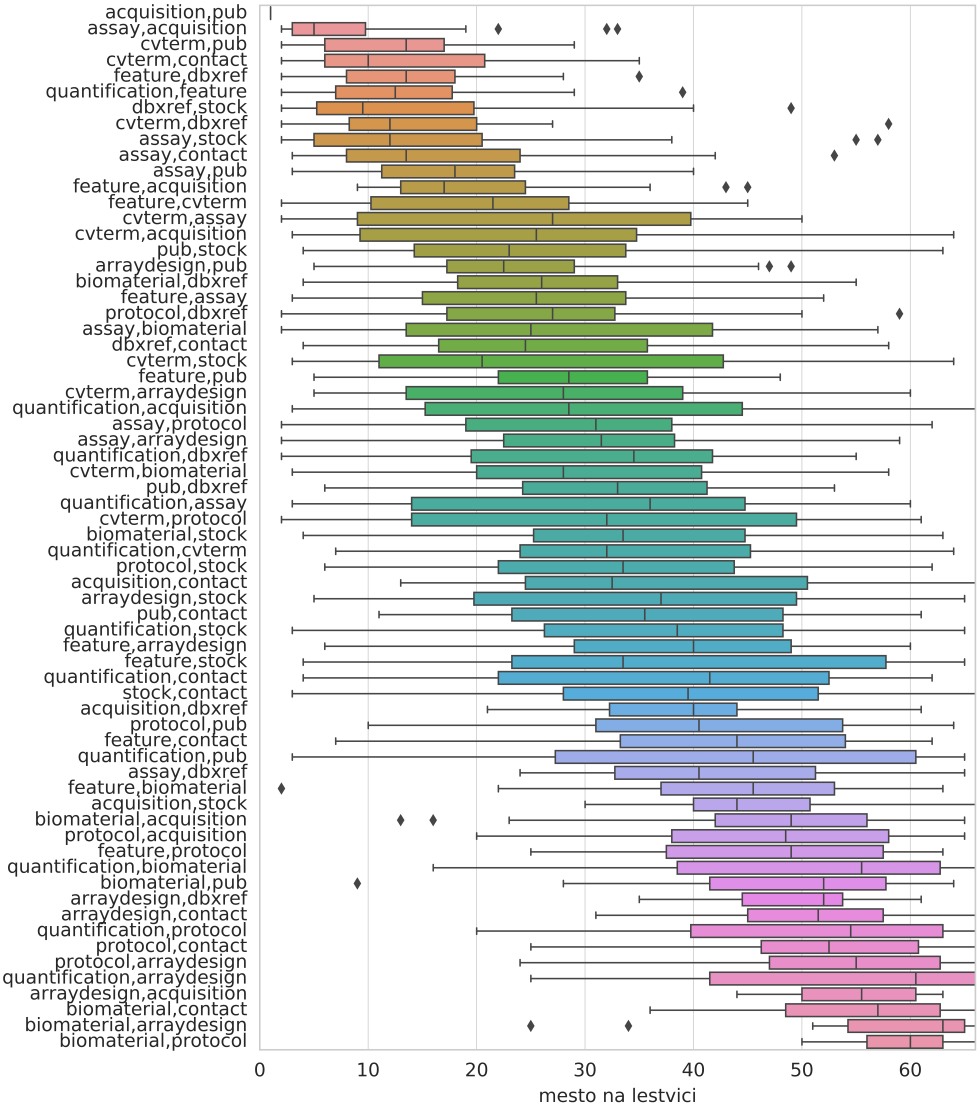
Slika 5.10: Konsistentnost uvrstitve posamezne relacije na določeno mesto v seznamu, v 30 izvedbah programa nad testno bazo avtomobilizem.

Avtomobilizem z 8 tabelami in ravno toliko relacijami predstavlja najmanjšo izmed treh baz. Ker imajo tabele v njej tudi majhno število zapisov, se jim model DFMF lažje prilagodi tudi z majhno velikostjo latentnega prostora. Rezultate prikazuje slika 5.10.



Slika 5.11: Konsistentnost uvrstitve posamezne relacije na določeno mesto v seznamu, v 30 izvedbah programa nad bazo pagila.

Baza Pagila z 21 tabelami in ravno toliko relacijami je primer baze srednje velikosti. Tu lahko vidimo, da ima uvrstitev relacij, ki zasedajo glavo in rep seznamov, nižjo varianco, kot pa relacije, ki se pojavljajo v srednjem delu. Rezultate prikazuje slika 5.11.



Slika 5.12: Konsistentnost uvrstitve posamezne relacije na določeno mesto v seznamu, v 30 izvedbah programa nad pomanjšano različico baze parameciumDB.

Baza parameciumDB je sicer največja izmed baz (141 tabel), vendar smo njeno velikost bistveno zmanjšali v koraku predobdelave. Pomanjšana različica baze je ob analizi tako vsebovala 0.01% zapisov prvotne baze, 12 obravnavanih tabel in 66 relacij. Porazdelitev rangiranja prikazuje slika 5.12.

Poglavje 6

Sklepne ugotovitve

Tekom dela smo si ogledali nekaj obstoječih metod namenjenih preiskovalni analizi podatkovnih baz in predlagali svojo metodo, ki razširja nabor sistemov za avtomatizacijo procesa preiskovanja podatkov s samodejno prepoznavo in predstavitvijo relevantnih podatkov. Koncept metode smo implementirali v obliki modula za programski jezik Python. Pri tem smo tudi naslovili nekaj zmogljivostnih problemov, ki so se v praksi pokazali pri obdelavi večjih baz. Programska koda je dostopna na avtorjevem github repozitoriju [3].

Med delom smo oblikovali tudi nekaj zamisli o izboljšavah, implementacije katerih pa so izven obsega tega diplomskega dela, bi jih pa želeli nasloviti v prihodnosti. Ena od teh je rekurzivno prepisovanje atributov iz tabel, ki ne vsebujejo tujih ključev, in na katere se ne sklicuje nobena od vmesnih tabel. Take tabele tipično niso močno vpete v graf oz. ne tvorijo veliko povezav s sosedi. Posledica je, da take tabele v koraku vzorčenja izpadejo iz nadaljnje obravnave, z njimi pa tudi nekaj informacije. Atribute takih tabel bi lahko pred njihovo izločitvijo rekurzivno pridružili k opisu tabel, ki se na njih sklicujejo.

Vredno bi bilo tudi preizkusiti vpliv izbora velikosti latentnega prostora objektnih tipov pri gradnji modela DFMF na kvaliteto končnega seznama relacij in izvajalni čas programa pri različnih velikostih in strukturah podatkovnih baz. Tako bi lahko vrednost parametra tudi predlagali uporabniku

ob času izvajanja.

Glede na razpoložljive sistemske vire bi program svetoval uporabniku izvedbo postopkov vzorčenja zapisov v tabelah, postopkov omejevanja števila obravnavanih tabel ter vrednosti parametrov, vezanih na pomnilniško in računsko zahtevnost, kot so:

- omejitev velikosti posamezne matrike,
- omejitev števila alternativnih relacijskih matrik, ki se jih obravnava za vsako od relacij in
- omejitev števila različnih kategorij za tvorbo indikatorskih atributov.

Od uporabnika ne bi smeli pričakovati, da poda parametre za ustvarjanje pomanjšane verzije podatkovne baze, temveč bi se morala ta operacija izvajati v ozadju in s kar najmanjšim vložkom uporabnikovega dela. Orodje bi potrebovalo tudi prijaznejši uporabniški vmesnik.

Literatura

[1] Illustration of approximate non-negative matrix factorization (nmf). `https://commons.wikimedia.org/wiki/File:NMF.png`. Dostopano: 6. 9. 2018.

[2] 18F. rdbms-subsetter. `https://github.com/18F/rdbms-subsetter`, 2018.

[3] Matic Bernik. Rdb relation ranking tool. `https://github.com/MaticBernik/Relational-database-fusion-tool`, 2018.

[4] Gloria Chatzopoulou, Magdalini Eirinaki, Suju Koshy, Sarika Mittal, Neoklis Polyzotis, and Jothi Swarubini Vindhiya Varman. The querie system for personalized query recommendations. *IEEE Data Eng. Bull.*, 34(2):55–60, 2011.

[5] Marina Drosou and Evaggelia Pitoura. Ymaldb: Exploring relational databases via result-driven recommendations. *The VLDB Journal*, 22(6):849–874, December 2013.

[6] Susan Sales Harkins and Martin WP Reid. Structured query language. In *SQL: Access to SQL Server*, pages 1–5. Springer, 2002.

[7] Stratos Idreos, Olga Papaemmanouil, and Surajit Chaudhuri. Overview of data exploration techniques. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, pages 277–281, New York, NY, USA, 2015.

[8] Sean Kandel, Andreas Paepcke, Joseph Hellerstein, and Jeffrey Heer. Wrangler: Interactive visual specification of data transformation scripts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 3363–3372. ACM, 2011.

[9] Marinka Žitnik. scikit-fusion. `https://github.com/marinkaz/scikit-fusion`, 2018.

[10] Marinka Žitnik and Blaž Zupan. Data fusion by matrix factorization. *CoRR*, abs/1307.0803, 2013.

[11] Marinka Žitnik and Blaž Zupan. Survival regression by data fusion. *Systems Biomedicine*, 2(3):47–53, 2014.