VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

GENEROVÁNÍ RODOKMENŮ Z ARCHIVNÍCH ZÁZNAMŮ

FAMILY TREES MAKING FROM ARCHIVE RECORDS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **DENIS POJEZDÁL**
AUTHOR

VEDOUCÍ PRÁCE **Ing. JAROSLAV ROZMAN, Ph.D.**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

144926

Ústav: Ústav inteligentních systémů (UITS)
Student: **Pojezdál Denis**
Program: Informační technologie
Specializace: Informační technologie
Název: **Generování rodokmenů z archivních záznamů**
Kategorie: Umělá inteligence
Akademický rok: 2022/23

Zadání:

1. Nastudujte obor genealogie a materiály v ní používané (matriky, lánové rejstříky, urbáře, atd.). Nastudujte práce, zabývající se automatickým propojováním osob v matričních záznamech (křty, svatby, úmrtí) do větších rodokmenů.
2. Na základě nastudované literatury navrhněte způsob, jak osoby uvedené v archivních pramenech propojovat do větších celků. Počítejte s tím, že pro danou oblast nebudou v danou chvíli k dispozici všechny záznamy, ty budou teprve průběžně přibývat. Dále systém navrhněte jako pravděpodobnostní, tzn. dítě může mít více rodičů, pro které se budou počítat pravděpodobnosti, které se budou s postupně přidávanými záznamy měnit. Data načítejte z databáze a opět je do databáze ukládejte.
3. Navržený systém implementujte.
4. Testování proveďte na dodané testovací sadě.

Literatura:

- MARHEFKA, Adam. Generování rodokmenů z matričních záznamů. Brno, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jaroslav Rozman, Ph.D.
- TUŠIMOVÁ, Lucia. Generování rodokmenů z matričních záznamů. Brno, 2020. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jaroslav Rozman, Ph.D.

Při obhajobě semestrální části projektu je požadováno:

- První dva body zadání

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Rozman Jaroslav, Ing., Ph.D.**
Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.
Datum zadání: 1.11.2022
Termín pro odevzdání: 10.5.2023
Datum schválení: 3.11.2022

Abstrakt

Práca sa venuje oblasti genealógie, rôznym historickým záznamom, ich spracovaniu, porovnaniu a následnému prepojeniu a uloženiu vo forme grafu. Nadväzuje na práce Bc. Marhefku a Ing. Tušimovej, do ktorých pridáva podporu pre nové typy vstupných záznamov a optimalizuje rôzne aspekty pôvodného programu. Nové záznamy a úpravy sú na záver testované a porovnané na pôvodnej aj rozšírenej dátovej sade.

Abstract

This work studies the field of genealogy, various types of historical records, their processing, comparing and subsequent linking and storing in the form of a graph. The work expands the works of Bc. Marhefka and Ing. Tušimová, adds support for new types of input records and optimizes various aspects of the original program. New types of records and modifications are afterwards tested and compared on the original and extended datasets.

Kľúčové slová

genealógia, spracovávanie záznamov, tvorba rodokmeňov, grafová databáza

Keywords

genealogy, record processing, pedigree creation, graph database

Citácia

POJEZDÁL, Denis. *Generování rodokmenů z archivních záznamů.* Brno, 2023. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jaroslav Rozman, Ph.D.

Generování rodokmenů z archivních záznamů

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Jaroslava Rozmana Ph.D.. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Denis Pojezdál
9. mája 2023

Poďakovanie

Rád by som poďakoval vedúcemu svojej bakalárskej práce Ing. Jaroslavovi Rozmanovi Ph.D. za jeho pomoc a konzultácie pri vypracovávaní tejto práce.

Obsah

Kapitola 1

Úvod

Vďaka vyvíjajúcimi sa technológiám je genealógia v dnešnej dobe podstatne prístupnejšia než v minulosti. Digitalizácia historických záznamov prebieha už niekoľko rokov, čo umožňuje prístup ku kategorizovaným dátam z pohodlia domova. Napriek tomu je systematické vyhľadávanie a prepájanie osôb do väčších celkov veľmi pracné a časovo náročné. Automatizácii tohto procesu sa na škole venuje hneď niekoľko projektov. Práce Bc. Marhefku a Ing. Tušimovej sa zameriavajú práve na prepájanie osôb z jednotlivých záznamov do väčších rodokmeňov na základe podobnosti obsiahnutých informácii. Úspešne spracovávajú záznamy z matrík narodených, zosnulých a zosobášených. Moja práca sa zameriava na rozšírenie týchto prác o nové typy vstupných záznamov a riešenie problémov, ktoré pri prepájaní vznikajú.

Na začiatku bakalárskej práce sa budeme venovať teórii ohľadom genealógie a existujúcich záznamov a oboznámime sa s návrhom predchádzajúcich prác. Krok po kroku si prejdeme aj implementáciu ich systému.

Ďalej sa pozrieme, ako sú nove záznamy uložené v relačnej databáze a ako ich budeme spracovávať. Tiež si ukážeme rôzne typy anomálií, ktoré môžu pri prepájaní vzniknúť.

Nakoniec sa budeme venovať drobným úpravám zvyšujúcim úspešnosť a efektivitu programu, ako aj samotnej implementácii nových návrhov a testovaniu na poskytnutých dátach z rôznych typov záznamov.

Kapitola 2

Naštudovaná teória

Táto kapitola slúži na popis základných pojmov použitých v tejto práci a pojmov týkajúcich sa genealógie ako takej. Ďalej sú tu spomenuté rôzne typy záznamov, s ktorými sa v genealógii môžeme stretnúť. Nakoniec sú tu popísané predchádzajúce práce, na ktoré moja práca naväzuje. Informácie v tejto kapitole sú prevzaté zo zdrojov [11, 14, 10, 8, 12, 13, 4, 1].

2.1 Genealógia

Genealógia patrí medzi pomocné vedy historické. Slovo pochádza z gréckeho génos (rod) a logos (rozum, veda) = veda o rode, rodine. Zaoberá sa vzťahmi medzi jednotlivými osobami, rodinnou históriou a prevažne tvorbou rodokmeňov. Z historického hľadiska môžeme genealógiu rozdeliť do troch časových období.

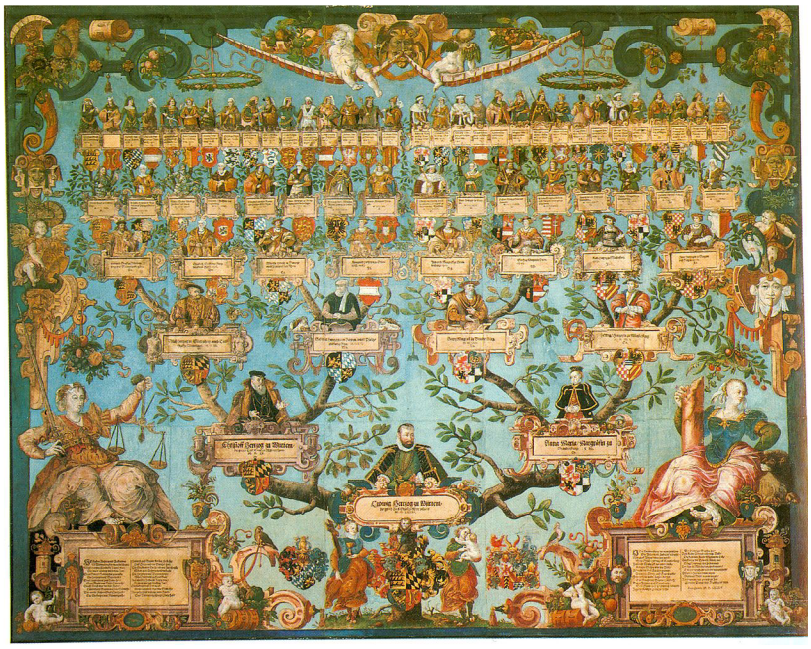
Ústna tradícia a biblické zdroje

Na počiatku civilizácie, než bolo vynájdené písmo, sa všetky záznamy museli uchovávať v pamäti ľudí alebo za pomoci jednoduchých mnemotických systémov (uzlíky na lankách, koráliky). Neskôr môžeme v Biblii nájsť rôzne zmienky odkazujúce na pôvod, vedúce až k Noemovi alebo Adamovi.

Ranné písomne zdroje

Toto obdobie začalo v starovekom Grécku a Ríme, kde vznikali básne a príbehy. Zatiaľ sa však nedá hovoriť o vede, ale skôr o vedľajšom produkte umelcov. Genealógia sa doteraz prevažne venovala významným osobnostiam. Panovníci a hrdinovia boli často považovaní za potomkov niektorého z bohov. V stredovekej Európe môžeme nájsť rodokmene kráľov siahajúce až do 6. storočia.

V 16. storočí sa prvý krát začínajú bežne objavovať záznamy týkajúce sa bežných ľudí napríklad prevody pozemkov alebo súdne záznamy. Na zvýšenom počte záznamov ma zásluhu aj väčší záujem renesančných panovníkov o stav ich pozemkov a poddaných.



Obr. 2.1: Rodinný strom Ludvíka II. Württemberského (asi 1568–1593) [17]

Moderná genealógia

V dnešnej dobe je genealógia dostupná pre všetkých. Nie len preto, že sa vedie podstatne viac podrobnejších záznamov o každej osobe, ale aj vďaka internetu, kde môžeme nájsť množstvo digitalizovaných záznamov z rôznych archívov.

2.2 Záznamy

Pri zisťovaní vzťahov medzi jednotlivými osobami a tvorbe rodokmeňov nám môžu pomôcť rôzne typy historických záznamov. V podstate sa dá použiť takmer akýkoľvek záznam obsahujúci pre nás relevantné informácie. Tie najdôležitejšie a najobsiahlejšie si preberieme v nasledujúcej sekcii.

2.2.1 Matričné záznamy

Matričné záznamy začali vznikať už v stredoveku a spravovala ich cirkev prevažne ako záznamy o svojich veriacich. Povinnosť viesť matriky bola zavedená v roku 1563 Tridentským koncilom, ale toto nariadenie sa zavádzalo do praxe veľmi pomaly. Nasledovalo zjednotenie ich formy v rokoch 1781 a 1784 a od polovice 20. storočia sa presunuli pod štátnu správu a používajú sa dodnes.

Matriky narodených

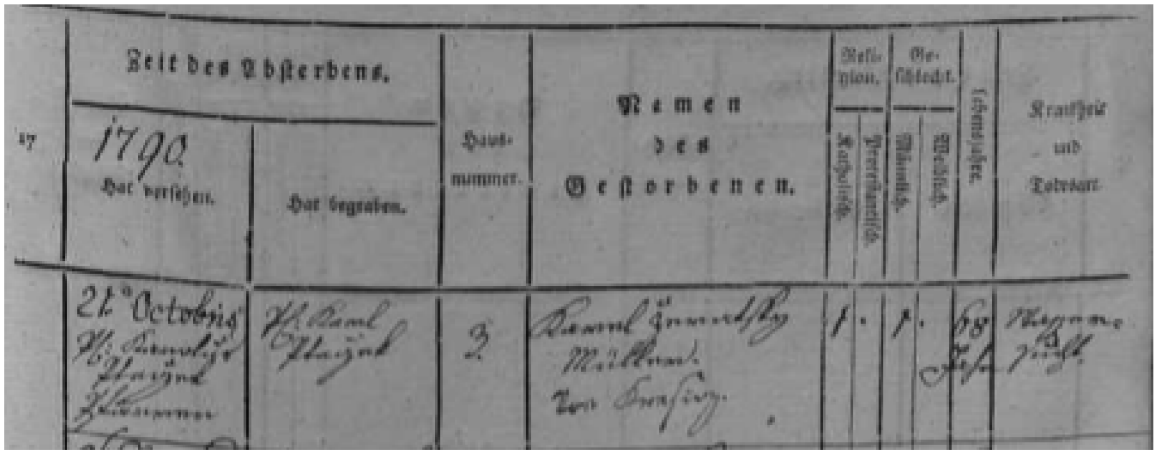
Obsahujú záznamy o narodení alebo krste dieťaťa. Obsahujú meno a pohlavie dieťaťa, dátum a miesto narodenia (alebo krstu), poprípade môžu obsahovať ďalšie informácie o dieťati, ako napríklad či sa jedná o manželské alebo nemanželské dieťa.

Ďalej obsahuje údaje o rodičoch dieťaťa, ich meno, bydlisko, vierovyznanie a povolanie, ale aj informácie o ďalších zúčastnených osobách ako je kmotor, starí rodičia, farár alebo pôrodná baba.

Obr. 2.2: Záznam o narodení z Brnenského biskupstva (1784–1845) [3]

2.2.2 Matriky úmrtí

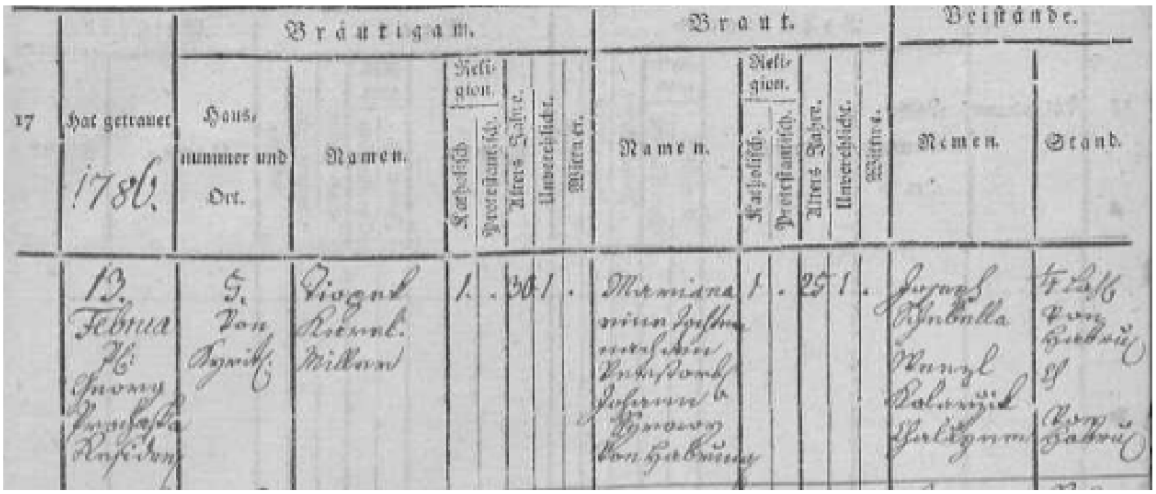
Obsahujú základné údaje o zosnulom ako meno, pohlavie, bydlisko a dátum narodenia. Väčšina ďalších informácií sa týka samotného úmrtia, dátum, miesto, príčina a meno kňaza zodpovedného za zaopatrenie a pohreb. Ak bola osoba zosobášená, mohli sa vyskytovať aj informácie o manželovi/manželke.



Obr. 2.3: Záznam o úmrtí z Brnenského biskupstva (1784–1894) [3]

2.2.3 Matriky sobášov

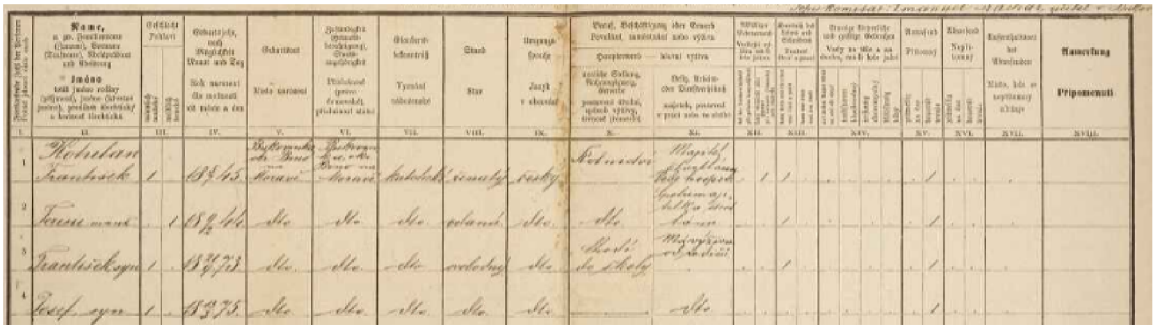
Najčastejšie údaje sú dátum sobáša a mená ženícha, nevesty, kňaza a svedkov. Rozsiahlejšie záznamy môžu obsahovať aj mená rodičov oboch oddávaných a ďalšie informácie ako bydlisko, povolanie a vek alebo dátum narodenia jednotlivých osôb.



Obr. 2.4: Záznam o sobáši z Brnenského biskupstva (1784–1845) [3]

2.2.4 Sčítacie operáty

Sčítania obyvateľstva sa uskutočňovali už v starovekom Babylone pred takmer 6 000 rokmi. Pôvodne sa nesčítavali len obyvatelia, ale aj stav dobytka alebo hospodárskeho vybavenia a hlavným účelom bolo stanovenie výšky daní. Prvé celo-územné sčítanie v Česku bolo takzvané 'soľné sčítanie' v roku 1702. Ale prvé novodobé sčítanie (také, ktoré obsahuje aj doplňujúce údaje o zaznamenaných osobách ako vek, pohlavie, zamestnanie, rodinný stav) sa uskutočnilo až v roku 1857 a opakuje sa približne každých 10 rokov.



Obr. 2.5: Záznam o sčítaní ľudu, Brno 1880 [3]

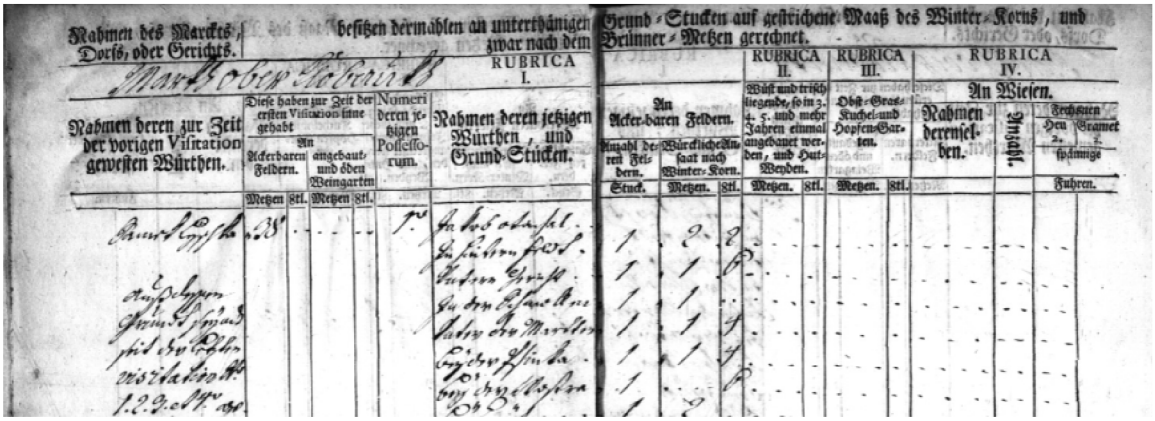
2.2.5 Lánové rejstríky

Patria medzi najstaršie dochované katastre v Česku a sú zostavené na základe dvoch lánových vizitácií z druhej polovice 17. storočia. Pozostávajú z 365 zväzkov pre všetky moravské panstvá. Vznikli za účelom výpočtu presných daňových príjmov z jednotlivých panstiev. Obsahujú súpis držiteľov usadlostí a pozemkov spolu s informáciami o veľkostiach ich polí.

Obr. 2.6: Záznam zo 145. knihy [3]

2.2.6 Poddanská priznávacia fase

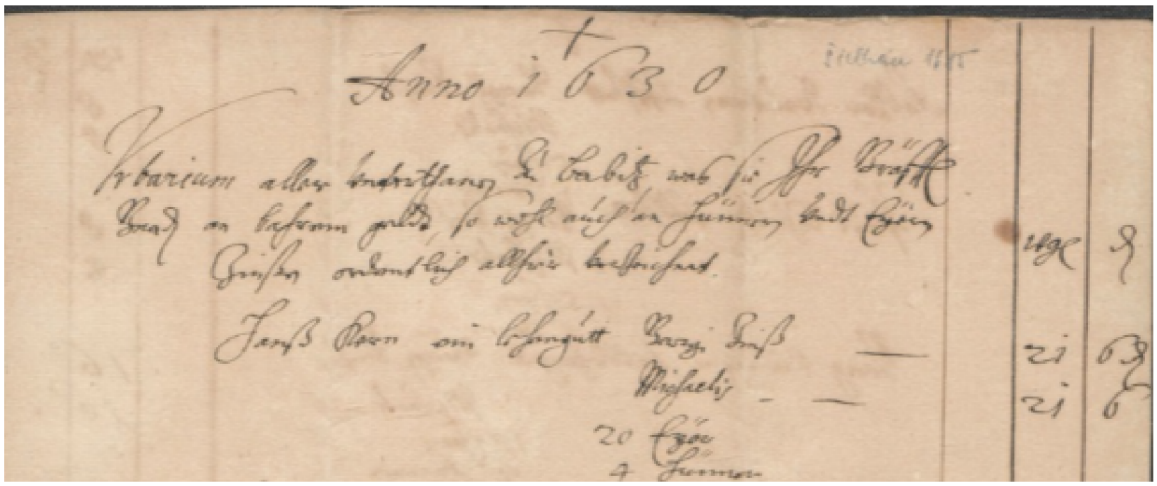
Poddanská priznávacia fase je najrozsiahlejšou súčasťou takzvaných rektifikačných akt z rokov 1667–1768. Samotná fase pochádza z roku 1749. Obsahuje informácie o majiteľoch pozemkov a veľkosti ich polí a viníc z čias druhej lánovej vizitácie (pred rokom 1749) ako aj informácie o majiteľoch v čase súpisu (v roku 1749), a údaje o ich pozemkoch vrátane hospodárskych budov ako napríklad mlyny a píly. Môžeme tu nájsť aj popis obce a informácie o úrodnosti pôdy.



Obr. 2.7: Záznam z rektifikačných akt [3]

2.2.7 Urbáre

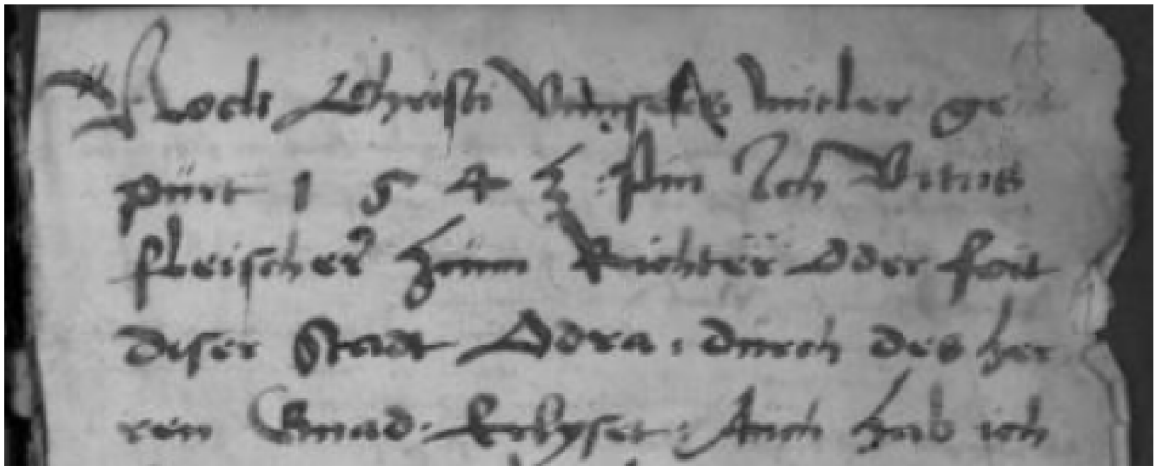
Slovo urbár pochádza zo staro-nemeckého ur-beran alebo erbern, čo znamená približne odvodený výnos. Urbáre dávalo vyhotovovať už panstvo v stredoveku za účelom evidencie poplatkov a povinností svojich poddaných. Najstarší dochovaný urbár v Česku, ktorý je súčasťou tzv. Manuálníku Jana Staicze pochádza z druhej polovice 13. storočia.



Obr. 2.8: Urbár z roku 1630 [3]

2.2.8 Pozemkové knihy

Pozemkové (gruntové) knihy slúžili na zaznamenávanie prevodov a predajov pozemkov. Mohli sa v nich nachádzať aj práva a povinnosti vyžadované od majiteľa (nájomníka) daného pozemku. Pred rokom 1848, než došlo k zrušenie poddanstva, vlastnila väčšinu pôdy šľachta a obyčajní ľudia si ju len prenajímali. Najčastejšie sa pozemok prevádzal z otca na syna alebo z vdovy na nového manžela. Šľachta si však mohla vynútiť zmenu ak nebola s hospodárom spokojná.



Obr. 2.9: Pozemková kniha z rokov 1543-1558 [2]

2.3 Podkladové práce

Táto práca vychádza z práce Bc. Marhefku z 2021/2022 [10], ktorý nadväzoval na prácu Ing. Tušimovej z roku 2019/2020 [14]. Hlavným cieľom je pridanie nových typov vstupných záznamov a optimalizácia niektorých častí algoritmu. V tejto podkapitole si popíšeme existujúci návrh, technológie použité pri implementácii a samotnú implementáciu riešenia s výsledkami, ktoré produkuje.

2.3.1 Návrh

Informácie o osobách musíme najprv získať z poskytnutých záznamov. Jeden záznam môže obsahovať informácie o vyše 20 osobách, niektoré explicitne a iné sa dajú vyvodiť z typu záznamu a vzťahov medzi osobami. Pre čo najpresnejšie porovnávanie a vyhodnocovanie vzťahov potrebujeme čo najviac informácií, ktoré nám môžu pomôcť pri identifikácii a klasifikácii porovnávaných osôb. Medzi tie najpodstatnejšie patria:

- Meno
- Priezvisko
- Pohlavie
- Titul
- Povolanie
- Bydlisko
- Dátum narodenia
- Odhad dátumu narodenia
- Miesto narodenia
- Dátum krstu
- Dátum svadby
- Miesto svadby
- Dátum úmrtia
- Odhad dátumu úmrtia
- Miesto úmrtia
- Dátum pohrebu
- Členovia rodiny

Odhad dátumu narodenia a úmrtia sú dva najdôležitejšie údaje, ktoré vieme väčšinou iba vyvodiť z iných dát. Zabezpečujú zníženie počtu porovnaní, ktoré musíme spraviť, keďže osoby, ktoré nežili v rovnakom časovom období neporovnávame.

Ako môžeme vidieť, máme rôzne typy dát. Slová, napríklad meno a priezvisko, dátumy a geografické dáta. Pre každý typ sa používa špecifický druh porovnávania a v nasledujúcich podkapitolách si ich rozoberieme.

Porovnávanie slov

Slová môžeme mať buď v nenormalizovanej forme (tak, ako sa nachádzali v originálnych písomných záznamoch) alebo v normalizovanej forme. Vzhľadom k tomu, že záznamy môžu pochádzať z rôznych časových období a môžu byť písané rôznym jazykom, niektoré výrazy pre tú istú vec sa môžu líšiť. Môže ísť o drobné rozdiely v menách (Peter a Petr, Josef a Jozef) alebo rôzne pomenovanie pre to iste povolanie (farmár a bauer). Aby sa predišlo zníženiu presnosti, môžu mať tieto slová normalizovanú formu, ktorá ich zhlukuje do jedného.

Z toho vyplýva, že ak máme normalizovanú formu pre obe porovnávané slová, zisťujeme, či sú úplne zhodné alebo nie. Pri nenormalizovaných by takéto porovnávanie často nefungovalo a preto sa využíva Levenshteinova vzdialenosť. Testované boli aj iné metriky (Jarova, Jaro-Winklerova) ale Levenshteinova sa ukázala ako najúspešnejšia. Tá nám udáva koľko jedno-znakových operácií treba spraviť, aby sme previedli jedno slovo na druhé. Medzi tieto operácie patrí vloženie, zmazanie a substitúcia znaku.

		0	1	2	3	4	5
			P	**E**	**T**	**E**	**R**
0		**0**	1	2	3	4	5
1	**P**	1	**0**	1	2	3	4
2	**E**	2	1	**0**	1	2	3
3	**T**	3	2	1	**0**	1	2
4	**R**	4	3	2	1	**1**	2
5	**A**	5	4	3	2	2	**2**

Tabuľka 2.1: Ukážka určenia Levenshteinovej vzdialenosti

V tabuľke 2.1 je vidieť postup výpočtu Levenshteinovej vzdialenosti medzi slovami Peter a Petra. Finálna vzdialenosť je 2. Túto vzdialenosť použijeme pri výpočte podobnosti slov.

$$sim_{levenshtein}(s_1, s_2) = 1 - \frac{dist_{levenhstein}(s_1, s_2)}{max(|s_1|, |s_2|)}$$

$sim_{levenshtein}$ je finálna podobnosť medzi dvoma slovami. Ako môžeme zo vzorca vidieť, čím menej úprav musíme spraviť, tým viac sa podobnosť blíži 1 a naopak, ak sa slová vôbec nepodobajú, klesá až na 0.

Porovnávanie dátumov

Pri porovnávaní dátumov sa využíva niekoľko spôsobov. Všetky dátumy sú prevedené na reťazce a porovnané metódou spomenutou vyššie. Toto porovnávanie berie ohľad na možné chyby spôsobené prepisom, napríklad dátumy 21.8.1852 a 21.8.1862 majú vysokú podobnosť napriek tomu, že sú od seba vzdialené 10 rokov.

Ďalšou chybou, ktorá sa môže vyskytovať, je prehodenie dňa a mesiaca v zápise, napríklad 11.2.1834, a 2.11.1834. Preto ak majú dátumy rovnaký rok, porovnávame navzájom aj ich dni a mesiace. Ak nám vyjde zhoda, predpokladáme, že majú podobnosť 0,5.

Pri porovnávaní dátumu, z ktorého vieme vypočítať vek (dátum narodenia, dátum krstu), používame ešte metódu porovnania na základe percentuálneho rozdielu ich veku (age percentage difference). Ten sa vypočíta ako pomer rozdielu vekov k ich maximu.

$$apc = \frac{|d_1 - d_2|}{max(|d_1|, |d_2|)} * 100$$

Ďalej si musíme určiť maximálnu toleranciu apc_{max}, ktorá nám bude udávať najväčší možný akceptovateľný rozdiel, momentálne nastavená na 10 %. Výslednú podobnosť dátumov potom určíme na základe

$$sim_{age} = \begin{cases} 1 - \frac{apc}{apc_{max}} & \text{ak } apc < apc_{max} \\ 0 & \text{inak} \end{cases}$$

Tento spôsob rieši nepresnosti vzniknuté tým, že v minulosti si ľudia dosť často presne nepamätali svoj vek. Ako finálnu podobnosť dátumov zvolíme najvyššiu nájdenú z vyššie spomenutých možností.

Porovnávanie geografických dát

Mestá a obce taktiež porovnávame pomocou viacerých spôsobov. Normalizované a nenormalizované názvy porovnávame klasicky ako slová. Ak však máme aj informácie o polohe mesta, vypočítame ich vzdialenosť v kilometroch. Keďže sťahovanie sa na veľké vzdialenosti nebolo v minulosti bežné, môžeme predpokladať, že čím bližšie sa k sebe mestá nachádzajú tým je väčšia pravdepodobnosť, že sa jedná o tú istú osobu. Opäť si za finálnu podobnosť vyberieme tu najvyššiu.

Prevod zo vzdialenosti v kilometroch na podobnosť je realizovaný za pomoci exponenciálnej funkcie, pričom je definovaná maximálna akceptovateľná vzdialenosť na 100 km.

$$sim_{dist} = \begin{cases} 1 - \frac{e^{dist}}{e^{dist_{max}}} & \text{ak } dist < dist_{max} \\ 0 & \text{inak} \end{cases}$$

Porovnávanie čísel

Jediný číselný údaj, ktorý porovnávame je číslo ulice. Opäť využívame Levenstainovu vzdialenosť, aby sme brali ohľad na chyby opisu napríklad 89 a 99. Okrem toho si vypočítame aj numerickú podobnosť porovnávaných čísel. Zvolená maximálna vzdialenosť d_{max} je 10. Podobnosť vypočítame podobne ako pri veku.

$$sim_{num} = \begin{cases} 1 - \frac{|n_1 - n_2|}{d_{max}} & \text{ak } |n_1 - n_2| < d_{max} \\ 0 & \text{inak} \end{cases}$$

Porovnávanie záznamov

Cele záznamy o osobách sú rozdelené na jednotlivé atribúty, na každý je uplatnené jedno z vyššie spomenutých porovnaní a výsledky sa ukladajú do porovnávacieho vektoru. Tento vektor využijeme pri klasifikácii záznamu. V nasledujúcej tabuľke 2.2 môžeme vidieť časť porovnávacieho vektoru dvoch osôb s hodnotami atribútov a ich podobnosťou.

ID	Meno	Priezvisko	Pohlavie	Dátum nar.	Č. domu	Názov ulice	Mesto
a1	alica	mlynárová	žena	18.10.1956	15	božetechova	brno
a2	alica	mikuláková	žena	18.10.1958	68	dožetekova	břeclav
	1,0	0,0	1,0	0,87	0,0	0,73	0,9

Tabuľka 2.2: Porovnávací vektor

2.3.2 Klasifikácia

Porovnávanie spomenuté v predchádzajúcej sekcii využijeme pri rozhodovaní, či sa záznamy týkajú tej istej osoby. Čím vyššia nám vyjde podobnosť medzi dvoma záznamami, tým je väčšia pravdepodobnosť, že sa jedná o tú istú osobu.

Po načítaní nového záznamu a získaní všetkých informácií o osobe, ju porovnáme so všetkými už existujúcimi osobami a pre každú dvojicu r_i a r_j nám vznikne porovnávací vektor γ.

Vzhľadom k tomu, že každá hodnota porovnávacieho vektoru je normalizovaná do intervalu $\langle 0, 1 \rangle$, treba každému atribútu priradiť váhu, ktorá bude aplikovaná pred samotným

sčítaním vektoru. Týmto zabezpečíme možnosť nastaviť každému atribútu individuálnu dôležitosť na základe toho, koľko informácií nám poskytuje. Napríklad priezvisko by malo mať väčšiu váhu než vierovyznanie, ktoré mala väčšina ľudí rovnaké.

Po prenásobení tohoto vektoru s váhami jednotlivých atribútov a sčítaní nám vznikne finálna podobnosť sim_{sum}, ktorú využijeme pri určení vzťahu medzi porovnávanými záznamami. Tá je následne tiež normalizovaná do intervalu $\langle 0, 1 \rangle$.

Ak si zoberieme hodnoty γ z tabuľky vyššie a váhy:

$$w_{Meno} = 2,\ w_{Priezvisko} = 3,\ w_{Pohlavie} = 0,5,\ w_{Narodenie} = 3,$$
$$w_{cDomu} = 1,\ w_{Ulica} = 3,\ w_{Mesto} = 2$$

tak môžeme finálnu podobnosť spočítať ako:

$$sim_{sum}[r_i, r_j] = 2*1,0+3*0,0+0,5*1+1*1,0+3*0,87+1*0,0+3*0,73+2*0,0 = 8,3$$

Alebo skrátene ako skalárny súčin, kde w je vektor váh:

$$sim_{sum}[r_i, r_j] = \gamma \cdot w$$

Normalizácia prebieha vydelením maximálnou možnou podobnosťou (súčtom váh), v našom prípade $14,5$.

$$sim_{sum}[r_i, r_j] = 8,3 \div 14,5 \approx 0,57$$

Na základe tejto podobnosti klasifikujeme porovnávaný pár záznamov o osobe do jednej z troch kategórií.

$$\begin{aligned} sim_{sum}[r_i, r_j] \geq t_u &\implies r \rightarrow \text{Zhoda} \\ t_l < sim_{sum}[r_i, r_j] < t_u &\implies r \rightarrow \text{Potenciálna zhoda} \\ sim_{sum}[r_i, r_j] \leq t_l &\implies r \rightarrow \text{Nezhoda} \end{aligned}$$

Prah zhody t_u a prah potenciálnej zhody t_l majú veľmi veľký dopad na výslednú úspešnosť systému. Z testovania predchádzajúcich autorov vyplynulo, že hodnoty 0,9 a 0,7 by mali poskytovať najpresnejšie výsledky.

Pravdepodobnostná klasifikácia

Pravdepodobnostná klasifikácia bola predstavená Ivanom Fellegi a Alanom Sunterom v roku 1969 v ich práci *A Theory for Record Linkage* [6]. Namiesto ručne definovaných váh jednotlivých atribútov využíva pravdepodobnosti pre výpočet dvoch váh (matched a unmatched) pre každý atribút.

Máme dve populácie A a B pričom si jednotlivé záznamy označíme ako a a b. Predpokladáme, že niektorí jednotlivci sa nachádzajú v oboch populáciách. Množinu usporiadaných dvojíc z týchto populácií si označíme ako:

$$A \times B = \{(a, b); a \in A, b \in B\}$$

Túto množinu môžeme ďalej rozdeliť na dve disjunktné množiny, množinu M (matched) obsahujúcu dvojice záznamov, kde sa jedná o tu istú osobu, to avšak nemusí znamenať, že sa záznamy musia úplne zhodovať,

$$M = \{(a, b); a = b, a \in A, b \in B\}$$

a U (unmatched), ktorá obsahuje dvojice, kde záznamy a a b hovoria o rôznych osobách.

$$U = \{(a, b); a \neq b, a \in A, b \in B\}$$

Pre každý pár porovnaných záznamov nám vzniká porovnávací vektor γ zložený z výsledkov porovnania jednotlivých atribútov. Pravdepodobnosť, že záznamy hovoria o zhodnej osobe, sa potom vypočíta ako podmienená pravdepodobnosť zhodných a nezhodných údajov v zázname.

$$R = \frac{P(\gamma \in \Gamma | (a, b) \in M)}{P(\gamma \in \Gamma | (a, b) \in U)}$$

Na základe tejto pravdepodobnosti a prahov spomenutých vyššie sa bude rozhodovať o klasifikácii záznamov.

$$\begin{aligned} R \geq t_u &\implies r \to \text{Zhoda} \\ t_l < R < t_u &\implies r \to \text{Potenciálna zhoda} \\ R \leq t_l &\implies r \to \text{Nezhoda} \end{aligned}$$

Za predpokladu, že sú tieto pravdepodobnosti nezávislé pre rôzne atribúty, môžeme pre každý atribút i vypočítať individuálnu váhu w_i. Pre výpočet potrebujeme dve pravdepodobnosti, m_i udávajúcu pravdepodobnosť, že dva záznamy majú rovnakú hodnotu v atribúte i ak hovoria o tej istej osobe

$$m_i = P([a_i = b_i, a \in A, b \in B] \mid (a, b) \in M)$$

a pravdepodobnosť u_i, ktorá udáva pravdepodobnosť, že dva záznamy majú rovnakú hodnotu v atribúte i ak nehovoria o tej istej osobe.

$$u_i = P([a_i = b_u, a \in A, b \in B] \mid (a, b) \in U)$$

Z týchto dvoch hodnôt môžeme vypočítať váhu w_i pre každý atribút i podľa toho či je hodnota zhodná alebo nie.

$$w_i = \begin{cases} \log_2 \frac{m_i}{u_i} & \text{ak } a_i = b_i \\ \log_2 \frac{1-m_i}{1-u_i} & \text{ak } a_i \neq b_i \end{cases}$$

2.3.3 Výsledky

Úspešnosť systému je testovaná na špeciálnej testovacej sade dát, v ktorej je manuálne určené, ktoré záznamy reálne odkazujú na tu istú osobu. Vďaka tomu môžeme po každej porovnanej dvojici skontrolovať, či výsledná klasifikácia súhlasí s realitou. Výsledok tejto kontroly vždy patrí do jednej zo štyroch kategórií.

- Pravdivá zhoda (True positive - **TP**) - záznamy sme klasifikovali ako zhodu a v skutočnosti sa jedná o tú istú osobu
- Nepravdivá zhoda (False positive - **FP**) - záznamy sme klasifikovali ako zhodu, ale v skutočnosti sa nejedná o tú istú osobu

- Pravdivá nezhoda (True negative - **TN**) - záznamy sme klasifikovali ako nezhodu a v skutočnosti sa nejedná o tú istú osobu
- Nepravdivá nezhoda (False negative - **FN**) - záznamy sme klasifikovali ako nezhodu, ale v skutočnosti sa jedná o tú istú osobu

Tieto hodnoty môžeme zapísať do takzvanej chybovej matice (Confusion matrix).

		Predpovedaný výsledok	
		Zhoda	Nezhoda
Skutočný výsledok	Zhoda	Pravdivá zhoda - **TP**	Nepravdivá nezhoda - **FN**
	Nezhoda	Nepravdivá zhoda - **FP**	Pravdivá nezhoda - **TN**

Tabuľka 2.3: Chybová matica

Ideálne chceme docieliť, aby bolo čo najviac záznamov klasifikovaných správne, to znamená v kategóriách **TP** alebo **TN**. Z tejto matice vieme vypočítať rôzne štatistiky, ktoré nám môžu pomôcť pochopiť ako úspešný bol algoritmus pri klasifikácii. Medzi tie najzaujímavejšie patria:

Presnosť (precision)

Udáva pomer správne klasifikovaných zhôd ku všetkým záznamom klasifikovaným ako zhoda.

$$precision = \frac{TP}{TP + FP}$$

Citlivosť (recall)

Udáva pomer správne klasifikovaných zhôd ku všetkým záznamom, ktoré mali byť klasifikované ako zhoda.

$$recall = \frac{TP}{TP + FN}$$

Špecifickosť (specificity)

Udáva pomer správne klasifikovaných nezhôd ku všetkým záznamom, ktoré mali byť klasifikované ako nezhoda.

$$specificity = \frac{TN}{TN + FP}$$

F-miera (F-measure)

Kombinuje presnosť a citlivosť do jednej hodnoty, počíta sa ako ich harmonický priemer.

$$Fmeasure = 2 * \frac{precision * recall}{precision + recall}$$

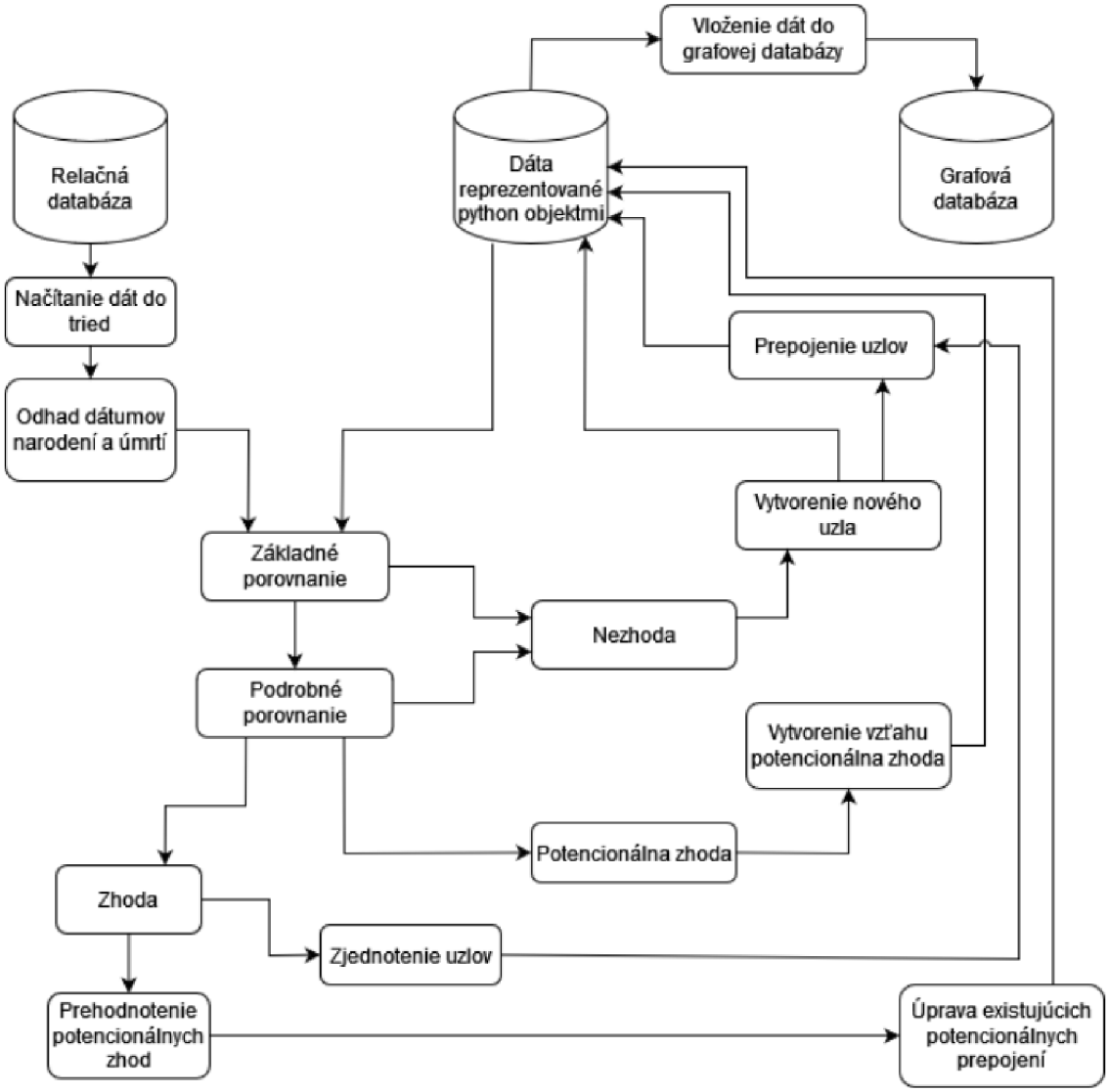
Balancovaná presnosť (balanced-accuracy)

Zohľadňuje nepomer vo veľkosti klasifikovaných množín (v databáze je podstatne viac záznamov ľudí, ktoré nehovoria o tej istej osobe než tých, ktoré hovoria), počíta sa ako aritmetický priemer citlivosti a špecifickosti.

$$BA = \frac{recall + specificity}{2}$$

2.3.4 Implementácia

V tejto podkapitole si najprv spomenieme využité technológie a potom krok po kroku prejdeme implementáciu celého systému podľa schémy nižšie 2.10.



Obr. 2.10: Schéma systému [10]

Python 3

Implementácia práce je v jazyku Python 3, čo je interpretovaný, dynamicky typovaný programovací jazyk, ktorý poskytuje veľké množstvo balíčkov ponúkajúcich ďalšiu funkcionalitu či už na prácu s databázami alebo na matematické výpočty.

Využité balíčky

- Levenshtein [1] - knižnica ponúkajúca výpočet Levenshteinovej vzdialenosti
- geopy [2] - výpočet vzdialenosti medzi geografickými súradnicami
- numpy [3] - matematická knižnica s množstvom funkcií
- datetime - práca a výpočty s dátumami
- json - práca s json súbormi
- pandas [4] - načítavanie a spracovávanie csv súborov
- mysql.connector [5] - ovládač k MySQL databáze
- neo4j [6] - ovládač k Neo4j grafovej databáze

MySQL databáza

Ako hlavný vstup dát obsahujúcich spracovávané záznamy slúži relačná MySQL databáza, ktorá obsahuje tabuľky pre každý typ záznamov, osoby, matriky a množstvo doprovodných tabuliek s ďalšími informáciami. K pripojeniu a komunikácii s ňou sa využíva MySQL Connector/Python balíček umožňujúci jednoduché pripojenie na základe prihlasovacích údajov a rýchle spúšťanie požadovaných MySQL dotazov.

Ďalšie vstupy dát

Okrem relačnej databázy môžu byť záznamy načítané aj z csv súborov, ktoré na prvom riadku obsahujú pomenovanie stĺpca a na ďalších riadkoch následne obsahujú čiarkami oddelené relevantné informácie.

Geografické dáta (zemepisná dĺžka a šírka) pre jednotlivé mestá a obce sú načítané z json súborov a následne uložené v pamäti po zbytok behu programu.

Neo4j databáza

Výsledné dáta s poprepájanými osobami sa vo forme grafu ukladajú do *Neo4j* grafovej databázy. Tá patrí k najpoužívanejším grafovým databázam vôbec a využíva dotazovací jazyk *Cypher*, ktorý mieri na prehľadnosť a jednoduchosť vďaka tomu, že dotazy vizuálne reprezentujú to, čo majú vykonať.

```
(nodes)-[:ARE_CONNECTED_TO]->(otherNodes)
```

[1] https://pypi.org/project/python-Levenshtein/
[2] https://numpy.org/
[3] https://numpy.org/
[4] https://pandas.pydata.org/
[5] https://dev.mysql.com/doc/connector-python/en/
[6] https://neo4j.com/developer/python/

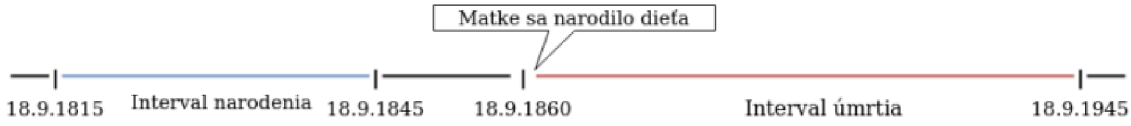
Krok 1: Načítanie nových záznamov

Ako už bolo spomenuté, primárne sa záznamy načítavajú z relačnej MySQL databázy. Práce, z ktorých vychádzam, implementovali spracovávanie záznamov z matrík narodených/pokrstených, matrík zosnulých a matrík oddaných. Pre každý z týchto záznamov sa v databáze nachádza samostatná tabuľka zhromažďujúca informácie, ktoré môže daný záznam poskytovať. Na tieto tabuľky môžu byť väzbami napojené ďalšie tabuľky, v ktorých sa nachádzajú informácie so všetkými zúčastnenými osobami, matrikami alebo normalizovanými dátami. Tie sú vytvorené za pomoci práce Bc. Davida Hříbeka s názvom *Poloautomatická normalizace slov z matričních záznamů* [7].

Záznamy každého typu sa najprv pomocou dotazov načítajú z databázy. S použitím cudzích kľúčov v hlavnej tabuľke záznamu nájdeme aj všetky dodatočné dáta v ďalších tabuľkách, ktoré následne uložíme do vytvorených python objektov (`Record, Register, Person, Domicile`). Ďalej už pracujeme len s týmito objektami. Keďže osoby môžu zmeniť meno, povolanie a množstvo ďalších atribútov, všetky atribúty osoby sú uložené ako listy.

Aby sme mohli porovnávať aj geografickú polohu miest, pre každé načítané mesto skontrolujeme, či sa nachádza v zozname miest so súradnicami, ktoré boli na začiatku programu načítané z json súborov. Ak sa dané mesto v zozname z nejakého dôvodu nenachádza, necháme súradnice nastavené na (-1, 1), aby sme vedeli, že jeho polohu nemáme porovnávať.

Aby sa zlepšil výkon, pre každú načítanú osobu zároveň vypočítame obdobie, v ktorom žila. Respektíve interval, v ktorom sa narodila a v ktorom zomrela. Tie určujeme podľa typu záznamu a role každej osoby. Takto nám pre každú osobu vzniknú dva intervaly, ktoré môžu mať rozpätie až desiatky rokov, ale zároveň môžu byť na deň presné. Napríklad pri zázname z matriky narodených vieme presne určiť narodenie dieťaťa, ale narodenie matky môžeme iba odhadnúť na základe predom určených pravidiel. Pri matke je pravidlo, že sa musela narodiť aspoň pred 15 rokmi a zároveň maximálne pred 45 rokmi, z toho vieme, za predpokladu, že sa ľudia dožívajú maximálne 100 rokov, vypočítať aj interval úmrtia 2.11.



Obr. 2.11: Určenie intervalov matky zo záznamu o narodení [14]

Krok 2: Vloženie záznamu a matriky

Ako prvé sa do dátovej reprezentácie, ďalej len DR, vloží samotný záznam. Následne sa podľa poskytnutého id matriky, z ktorej záznam pochádza skontroluje, či sa matrika už nachádza v DR, keďže z jednej matriky môže pochádzať viac záznamov. Podľa toho sa buď vytvorí nový uzol matriky, alebo sa záznam napojí na existujúci uzol.

Krok 3: Porovnávanie

Načítané osoby sa postupne porovnávajú s osobami už uloženými v DR. Táto štruktúra, ktorá nám počas celého behu programu nahrádza grafovú databázu, bola implementovaná Bc. Adamom Marhefkom [10]. Jej účelom je urýchliť porovnávanie, keďže táto dočasná „reprezentácia databázy“ nám odstraňuje nutnosť pri každom porovnaní načítavať všetky osoby z grafovej databázy nanovo do pamäti.

Osoby sú v nej rozdelené podľa pohlavia, aby sa predišlo zbytočným porovnávaniam, iba ak pohlavie danej osoby nieje známe, tak sa porovná so všetkými osobami.

Ako prvé prebehne len základné porovnanie medzi osobami, v tom sa podľa intervalov spomenutých vyššie skontroluje, či osoby žili v tom istom období a zároveň sa skontroluje, či majú podobný aspoň jeden zo základných atribútov (meno, priezvisko, povolanie alebo bydlisko).

Ak osoby splnili toto porovnávanie prechádza sa k viac detailnému porovnávaniu, kde sa využijú všetky dostupné informácie. Ak to záznam umožňuje a vieme z neho určiť aj predkov porovnávanej osoby a zároveň, aj osoba v DR ma pripojených predkov, detailne porovnáme aj ich. Výsledok tohoto porovnávania je už vyššie spomenutý porovnávací vektor (typ slovník pre prehľadnosť) najlepších zhôd, poprípade niekoľko vektorov, ak boli porovnávaní aj predkovia. Podľa neho prebehne následná klasifikácia.

Krok 4: Klasifikácia

Jednotlivé atribúty tohto vektoru vynásobíme s ich váhami a sčítame, aby sme dostali finálnu podobnosť. Podľa prahov podobnosti t_l a t_u sa rozhodne, či sa osoby zhodujú, potenciálne zhodujú, alebo ide o nezhodu. Na základe tejto klasifikácie prebehne vkladanie alebo aktualizácia DR.

Krok 5.1: Nezhoda

Ak nevyšla zhoda ani potenciálna zhoda medzi práve porovnávanou osobou a žiadnou z osôb uložených v DR, predpokladáme, že sa jedna o úplne novú osobu. Objekt `Person`, do ktorého boli načítané všetky údaje sa vloží do DR a stane sa z neho „uzol“. Podľa pohlavia sa priradí do jedného z troch listov (muž, žena, nedefinované) pre rýchlejší prístup.

Krok 5.2: Potenciálna zhoda

Potenciálna zhoda znamená, že osoby majú niektoré informácie podobné, ale nie je ich dosť na to, aby sme mohli s určitosťou povedať, že sa jedná o tú istú osobu. Ich podobnosť vyšla v intervale (t_l, t_u). Takúto osobu vložíme do DR rovnako ako v prípade nezhody a naviac ju prepojíme vzťahom POTENCIALNI_ZHODA so všetkými osobami, ktoré boli klasifikované ako potenciálne zhody. Vzťah je reprezentovaný slovníkom, ktorý obsahuje typ vzťahu (v tomto prípade POTENCIALNI_ZHODA), ale môže obsahovať aj ďalšie informácie, ako napríklad skóre (využité pri potenciálnych zhodách) alebo dátum.

Krok 5.3: Zhoda

Ak sme osoby klasifikovali ako zhodu, čiže ich podobnosť bola väčšia alebo rovná ako t_u, predpokladáme, že sa jedná o tú istú osobu. Keďže nová osoba môže obsahovať informácie, ktoré sa v pôvodnej osobe v DR nenachádzali, musíme ju aktualizovať (zlúčiť listy s údajmi). Pri zlučovaní prebehne aj úprava intervalov života, v oboch prípadoch sa zachová iba prienik týchto intervalov.

Vzhľadom k tomu, že spojením osôb nám mohli pribudnúť informácie, treba prekontrolovať všetky uzly, ktoré sú spojené s práve aktualizovanou osobou vzťahom POTENCIALNI_ZHODA. To prebehne ako klasické porovnávanie a klasifikácia s tým rozdielom, že ak nám klasifikácia vyjde ako zhoda, musíme po aktualizácii jeden z uzlov zmazať a všetky vzťahy, ktoré doňho alebo z neho viedli napojiť do toho druhého. Ak nám opäť

vyjde potenciálna zhoda nedeje sa v podstate nič, iba aktualizujeme skóre vzťahu POTENCIALNI_ZHODA. Môže sa ale stať, že po aktualizácii už osoby nebudú dosť podobné aby sa klasifikovali ako potenciálna zhoda (intervaly života sa už nemusia prekrývať alebo pridané informácie nesedia), v tom prípade sa iba odstráni tento vzťah z oboch uzlov.

Krok 6: Vkladanie miest do DR

Bydliská sú v grafovej databáze reprezentované tromi typmi uzlov, mesto, ulica a popisné číslo. V DR sú uložené ako list slovníkov reprezentujúcich mestá. Tie obsahujú informácie o meste, ďalej obsahujú list pre ulice v danom meste, tie sú tiež reprezentované ako slovníky, a list pre popisné čísla, ku ktorým nevieme ulicu. Pri vkladaní nového mesta do DR postupne po vrstvách kontrolujeme či už dané mesto, ulica v meste alebo číslo v ulici, poprípade v meste existuje. Vložíme vždy len nové dáta.

```
{
  "id": uuid.uuid4(),
  "name": "V Brně-Husovicích",
  "normalized_name": None,
  "gps": [0.0,0.0]
  "streets": [
              {
                "street_name": ("Palackého", uuid.uuid4()),
                "numbers": [
                            ("105", uuid.uuid4())
                          ]
              },
              {
                "street_name": ("Na rovině", uuid.uuid4()),
                "numbers": []
              },
              {
                "street_name": ("Brandlova", uuid.uuid4()),
                "numbers": []
              },
              {
                "street_name": ("Třebízského", uuid.uuid4()),
                "numbers": []
              },
              {
                "street_name": ("Vrchlického", uuid.uuid4()),
                "numbers": []
              }
            ],
  "numbers": []
}
```

Obr. 2.12: Slovník obsahujúci informácie o meste

Objekt `Domicile`, do ktorého sú zo záznamu načítané údaje, nie je použitý v DR. Slúži pri porovnávaní bydlísk osôb a pri prepájaní osôb a jednotlivých uzlov bydliska v grafovej databáze.

Krok 7: Prepojenie osôb v DR

V DR sú osoby prepájané so záznamami, v ktorých sa vyskytujú a s ďalšími osobami. Vzťahy medzi osobou a záznamom reprezentujú rolu osoby v zázname (napríklad MATKA, OTEC, KMOTOR), a obsahujú aj dátum záznamu. Každá osoba môže byť prepojená s viacerými záznamami vďaka spájaniu uzlov. Tá istá osoba môže v jednom zázname figurovať ako pôrodná baba a v ďalšom napríklad ako matka.

Vzťahy medzi osobami väčšinou reprezentujú rodinný stav (JE_MATKA, JE_OTEC, JE_KMOTOR), ale sú aj iné napríklad ODRODILA alebo KRSTIL. Tieto vzťahy vznikajú len medzi osobami z jedného záznamu kde ich vieme presne určiť z roly danej osoby, tá je väčšinou uložená ako `enum` aj v relačnej databáze aj v DR.

Krok 8: Vkladanie do grafovej databázy

Keď sme už porovnali všetky nové záznamy, treba dočasnú DR vložiť do grafovej databázy. Do nej sú najprv vložené matriky nasledované záznamami, ktoré sa aj hneď napoja vzťahom JE_V. Následne sa vložia a poprepájajú mestá, ulice a popisné čísla tak, ako boli uložené v DR (3 vrstvy). Pri osobách sa z dôvodu, že osoba môže byť napojená na osobu, ktorá sa v databáze ešte nenachádza najprv vložia všetky osoby bez vzťahov a až potom sa prepájajú s ďalšími osobami, záznamami a bydliskom. Bydliská sú napojené za pomoci štruktúr `domicile`, ktoré obsahujú potrebné id jednotlivých častí bydliska.

Kapitola 3

Návrh

Hlavnou úlohou tejto práce bolo pridanie podpory pre ďalšie typy vstupných záznamov. Ďalej sa venuje úpravám pôvodného systému za účelom zlepšenia presnosti a efektivity jednotlivých častí programu.

3.1 Nové záznamy

V tejto sekcii si preberieme spracovávanie nových typov záznamov. Ukážeme si ako sú uložené v relačnej databáze, aké informácie nám poskytnú a ako ich vieme využiť.

Sčítacie operáty

Jedným z najpodrobnejších a zároveň najužitočnejších záznamov sú sčítacie operáty. Nájdeme v nich podrobné informácie o celej rodine. V čase písania tejto bakalárskej práce sa v databáze na školskom servery `radegast.fir.vutbr.cz` ešte nenachádzali tabuľky popisujúce tento druh záznamu.

Sčítacie operáty boli dostupné iba vo forme csv súborov, ktoré budú v tejto sekcii opísané. Každý záznam sa v nich skladá iba z jednej osoby, my potrebujeme celú rodinu pridať do jedného záznamu aby sme mohli vytvárať vzťahy. To docielime agregovaním podľa bydliska, ktoré je veľmi podrobné, často až po čísla jednotlivých bytov.

O každej osobe nájdeme informácie ako meno, priezvisko, titul, pohlavie, väčšinou aj rodinný stav, dátum narodenia (nie veľmi presný) a miesto narodenia. Občas sa môže vyskytnúť aj dátum počiatku trvalého pobytu v zaznamenávanej obci.

Záznam obsahuje aj povolanie a môže sa vyskytnúť aj názov a miesto podniku, v ktorom daná osoba pracuje. Ďalšími informáciami môžu byť vierovyznanie, hovorené jazyky alebo telesné a duševné postihnutia osoby.

Vzťahy jednotlivých osôb v domácnosti sú definované voči majiteľovi domácnosti. To nám poskytuje množstvo vzťahov, ktoré takmer nikdy nenájdeme na jednom mieste. Zároveň tu však vzniká niekoľko problémov.

Prvým je, že tento vzťah môže byť od bežných ako manželka, brat, sestra, syn, dcéra, vnúča, otec, matka, svokra, svokor cez strýko, teta, bratranec, neter, až po podivnejšie ako sirota, nocľažník, stravník alebo čeledín. Tieto vzťahy sa zároveň môžu kombinovať, takže sa občas nájde aj manželkin brat, bratova manželka a tak ďalej. V csv súboroch tieto vzťahy zároveň nie sú normalizované, takže sa vyskytujú aj v rôznych formách a jazykoch, čo spôsobuje nemožnosť správne zarátať všetky.

Druhým problémom sú nevlastné deti. Môžeme mať dieťa z prechádzajúceho manželstva a manželkino dieťa. Niekedy je to špecifikované v názve vzťahu niekedy je to len spomenuté v poznámke. Obdobný problém vzniká aj pri vnúčatách, občas je meno rodiča pridané do názvu, občas je to v poznámke.

Vzhľadom na fakt že nemám zoznam normalizovaných vzťahov, ktoré sa môžu vyskytnúť, je prepájanie sčítacích operátov založené len na údajoch, ktoré sa mi podarilo vytiahnuť z poskytnutých dát. Pri pridaní do relačnej databázy bude potrebné prepájanie upraviť.

Lánové rejstríky

Vzhľadom k tomu, že lánové rejstríky boli vytvárané prevažne za účelom presnejšieho výpočtu daní, obsahujú veľké množstvo informácií týkajúcich sa pozemkov a hospodárskych zvierat. Nájdeme tu však aj zopár informácií o majiteľovi pozemku ako sú meno, priezvisko, pohlavie, či sa jedna o vdovu alebo dieťa. Všetky tieto údaje sa nachádzajú v tabuľke `taxPerson`. Keďže človek môže mať viac mien, medzi menom a osobou je vzťah „many-to-many“, aby sme sa dostali ku všetkým menám, musíme využiť spojovaciu tabuľku `taxPerson_name`, ktorá má odkazy na tabuľku `name`. Ak sa jedná o vdovu alebo dieťa záznam možno obsahuje aj ďalších ľudí, manžela alebo rodiča, tí budú označení v atribúte `rel`.

Collumn	Type	Null
id	mediumint	No
record	mediumint	Yes
surname	mediumint	Yes
alt_surname	mediumint	Yes
rel	enum('tax_main', 'tax_widow', 'tax_child')	Yes
mlst	enum('ml.', 'st.', 'U')	Yes
sex	enum('M', 'F', 'U', '[M]', '[F]')	Yes
widow	tinyint(1)	Yes
son_daughter	tinyint(1)	Yes
other	varchar(255)	Yes
roles	int	Yes
saddledType	enum('p', 'c', 'U')	

Tabuľka 3.1: Tabuľka `taxPerson`

Zo záznamu nachádzajúceho sa v tabuľke `taxRecord` vieme vyčítať všetky všeobecné informácie o zázname a taktiež za pomoci tabuľky `domicile` zistiť názov obce, v ktorej sa popisovaný pozemok nachádza.

Collumn	Type	Null
id	mediumint	No
register	mediumint	Yes
scan	smallint	Yes
pos	tinyint(1)	Yes
lang	enum('CZ', 'GE', 'LAT', 'SC', 'PL', 'SK', 'NONE')	Yes
domicile	mediumint	Yes
manor	mediumint	Yes
comment	varchar(255)	Yes

Tabuľka 3.2: Časť tabuľky `taxRecord`

Poddanská priznávacia fase

Najdôležitejším údajom, ktorý v nej nájdeme sú informácie o majiteľovi daného pozemku. Tie vieme získať z tabuľky `phasePerson` a tabuliek `phasePerson_names` a `surname`, na ktoré sa odkazuje. Informácie samotného záznamu ako je jazyk, poradie skenu, pozícia na skene a id matriky, z ktorej záznam pochádza, nájdeme v tabuľke `phaseRecord`, kde nájdeme aj niektoré dáta týkajúce sa pozemku. Z tabuľky `domicile` ešte vieme vyčítať, kde človek žil v čase vytvorenia záznamu.

Collumn	Type	Null
id	mediumint	No
record	mediumint	Yes
rel	enum('phase_orig', 'phase_orig_rel', 'phase_cur', phase_cur_rel)	Yes
surname	mediumint	Yes
sex	enum('M', 'F', 'U', '[M]', '[F]')	Yes
personRelation	mediumint	Yes

Tabuľka 3.3: Tabuľka `phasePerson`

Collumn	Type	Null
id	mediumint	No
register	mediumint	Yes
scan	smallint	Yes
pos	tinyint(1)	Yes
lang	enum('CZ', 'GE', 'LAT', 'SC', 'PL', 'SK', 'NONE')	Yes
manor	mediumint	Yes
domicile	mediumint	Yes
field	mediumint	Yes
comment	varchar(255)	Yes

Tabuľka 3.4: Časť tabuľky `phaseRecord`

Ako môžeme vidieť z atribútu `rel` v tabuľke `phasePerson`, záznam môže obsahovať až štyroch ľudí, dvoch z čias 2. lánovej vizitácie (1669-1679) a dvoch z čias súpisu fasy (1749).

Skutočnosť, že vieme, kedy sa obe tieto udalosti konali, využijeme na odhad narodenia a úmrtia týchto osôb, keďže môžeme predpokladať, že počas súpisu boli nažive. Zároveň môžeme využiť atribút `personRelation` odkazujúci na tabuľku `personRelation` pre zistenie vzťahu, ktorý medzi sebou osoby mali.

Urbáre

Informácie o osobách z urbárnych záznamov sa nachádzajú v tabuľkách `urbariumPerson`, `urbariumPerson_name` a `urbariumPerson_occupation`, ktorá slúži ako spojovacia tabuľka s tabuľkou povolaní `occupation`. Z týchto tabuliek vieme vyčítať meno, priezvisko, povolanie, vierovyznanie (údaj o židovskej viere) a z tabuľky `personRelation` vieme zistiť vzťah majiteľa pozemku k jeho príbuznému ako vyplýva z atribútu `rel`. Môžeme vidieť, že záznam môže obsahovať až štyroch majiteľov a príbuzného každého z nich. Atribút `ownFromYear` nám aspoň čiastočne udáva od kedy pozemok patril ktorej osobe, čo môžeme využiť pri odhadoch dátumov narodenia a úmrtia daných osôb.

Collumn	Type	Null
id	mediumint	No
record	mediumint	Yes
rel	enum('urb_orig', 'urb_orig_rel', 'urb_next_1', 'urb_next_1_rel', 'urb_next_2', 'urb_next_2_rel', 'urb_next_3', 'urb_next_3_rel')	Yes
desolate	tinyint(1)	Yes
surname	mediumint	Yes
descr_num	varchar(255)	Yes
jew	tinyint(1)	Yes
personRelation	mediumint)	Yes
ownFromYear	varchar(255)	Yes

Tabuľka 3.5: Časť tabuľky `urbariumPerson`

V tabuľke `urbariumRecord` nájdeme už vyššie spomenuté základné údaje o zázname a matrike, z ktorej pochádza. Okrem toho tu je názov a poradie gruntu a keďže na rozdiel od predchádzajúcich záznamov vznikali urbáre v širokom časovom rozpätí, je tu aj rok súpisu daného záznamu. Z tabuľky `domicile` opäť vyplýva, kde sa popisovaný grunt nachádzal.

Collumn	Type	Null
id	mediumint	No
register	mediumint	Yes
scan	smallint	Yes
pos	tinyint(1)	Yes
lang	enum('CZ', 'GE', 'LAT', 'SC', 'PL', 'SK', 'NONE')	Yes
grunt_order	int	Yes
domicile	mediumint	Yes
created_year	varchar(255)	Yes
name	varchar(255)	Yes
comment	varchar(255)	Yes

Tabuľka 3.6: Časť tabuľky `urbariumRecord`

Pozemkové knihy

Pozemkové knihy sú obsiahnutými informáciami dosť podobné lánovým rejstríkom. V tabuľkách `landIndexPerson` a `landIndexPerson_name` klasicky nájdeme meno a priezvisko majiteľa, jeho titul, pohlavie, vierovyznanie a vzťah k ostatným osobám v zázname.

Collumn	Type	Null
id	mediumint	No
rel	enum('li_main', 'li_orphan', 'li_widow', 'li_child', 'li_vo', 'li_fo', 'li_of')	Yes
record	mediumint	Yes
title	varchar(255)	Yes
surname	mediumint	Yes
alt_surname	mediumint	Yes
orphan	tinyint(1)	Yes
widow	tinyint(1)	Yes
son_daughter	tinyint(1)	Yes
jew	tinyint(1)	Yes
sex	enum('M', 'F', 'U', '[M]', '[F]')	Yes

Tabuľka 3.7: Časť tabuľky `landIndexPerson`

Tabuľka záznamu `landIndexRecord` obsahuje matriku, z ktorej pochádza, jazyk, v ktorom bol záznam písaný, meno gruntu a kde sa nachádzal. Taktiež tu nájdeme rok výstavby (založenia) a rok prevodu pozemku, ktorý môžeme využiť pri odhadoch dátumov.

Collumn	Type	Null
id	mediumint	No
register	mediumint	Yes
scan	smallint	Yes
pos	tinyint(1)	Yes
lang	enum('CZ', 'GE', 'LAT', 'SC', 'PL', 'SK', 'NONE')	Yes
domicile	mediumint	Yes
manor	mediumint	Yes
landName	mediumint	Yes
construction_year	varchar(255)	Yes
take_year	varchar(255)	Yes
comment	varchar(255)	Yes

Tabuľka 3.8: Časť tabuľky `landIndexRecord`

3.2 Anomálie

Podstatným problémom, ktorý som odhalil behom úprav, konkrétne pri porovnávaní predkov osoby, sú rôzne anomálie, ktoré vznikajú pri nesprávnom zlúčení osôb, ktoré nemali byť zlúčené. Keďže osoba, ktorá vznikla zlúčením dvoch osôb, obsahuje všetky informácie z oboch pôvodných, vrátane všetkých vzťahov, ktoré tieto osoby mali, občas sa stane, že výsledná osoba obsahuje vzťahy, ktoré nedávajú zmysel.

V ďalších sekciách si preberieme aké druhy anomálií môžu vznikať, akým spôsobom vznikajú a ukážeme si návrh, ktorý by mohol ich vzniku predísť, alebo ich vznik aspoň čiastočne obmedziť.

3.2.1 Cykly dĺžky 1

Najpodivnejšiou anomáliou, na ktorú som narazil, sú cykly dĺžky 1. To znamená ze človek bol sám sebe otcom alebo matkou. Keďže vzťahy typu JE_OTEC a JE_MATKA sa vytvárajú iba vrámci osôb načítaných z jedného záznamu, kde vieme presne určiť požadované vzťahy medzi osobami, pri bežnom zlúčení dvoch osôb takýto výsledok nemôže nastať vzhľadom na to, že osoby z toho istého záznamu nie sú navzájom porovnávané.

Na druhú stranu, vzťahy POTENCIALNI_ZHODA môžu vzniknúť iba medzi porovnávanými osobami, to znamená osobami, ktoré nepochádzajú z jedného záznamu. Inými slovami, vzťahy JE_OTEC alebo JE_MATKA (aj každý iný vzťah) a POTENCIALNI_ZHODA by sa mali vzájomne vylučovať.

Tento fakt zaručuje, že pri bežnej kontrole potenciálnych zhôd a prípadnom následnom zlúčení nám cyklus dĺžky 1 nemôže vzniknúť.

Ukázalo sa, že tento predpoklad v jednom prípade nebol dodržaný. Problém nastával pri viacnásobnom zlučovaní jednej osoby cez vzťah POTENCIALNI_ZHODA. Pre jednoduchšie pochopenie si tento prípad ukážeme na názornej príklade.

Máme dve osoby, napríklad otca so synom, ktorých sme načítali zo záznamu o narodení. Tieto osoby by mali byť podobné, takže si ich nazveme **X** a **x**.3.1a

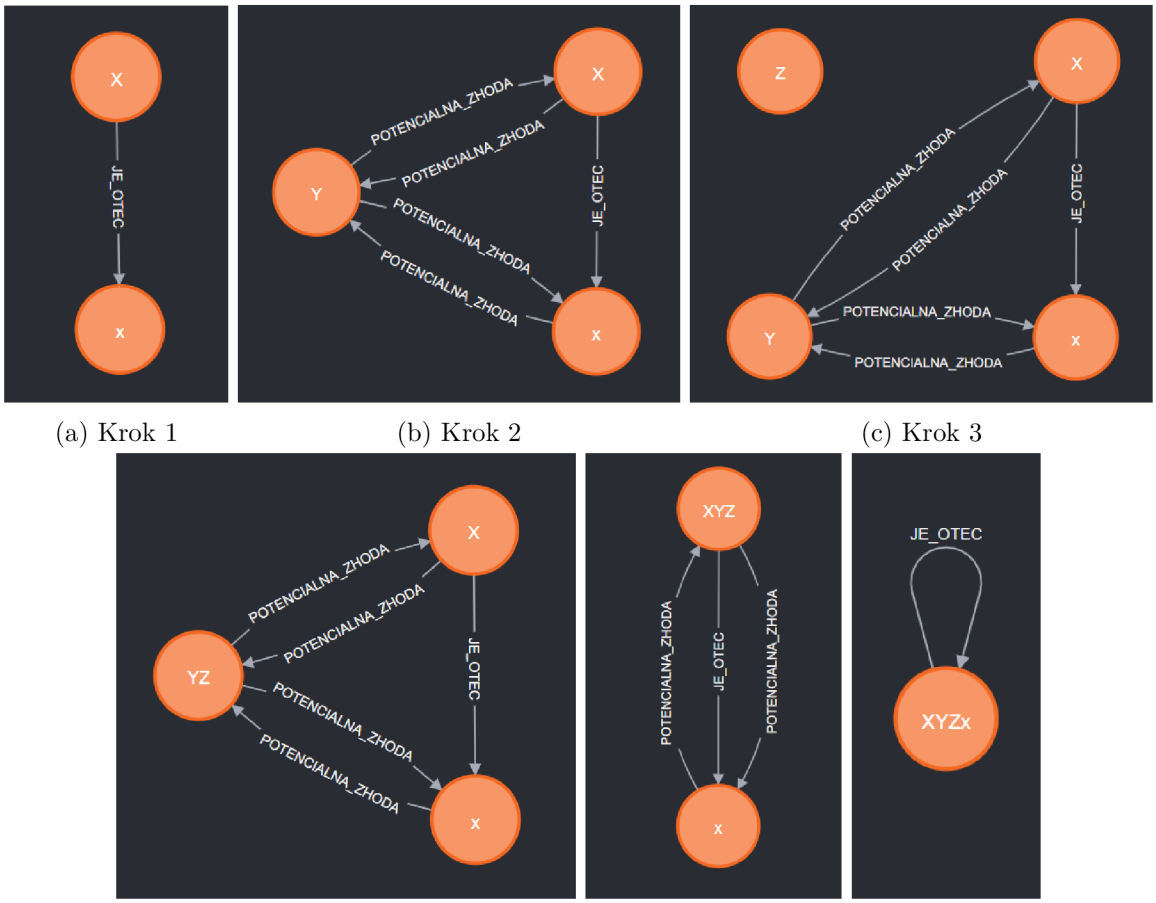
Z iného záznamu sme načítali ďalšiu osobu, nazvyme ju **Y**, ktorú sme porovnali s otcom aj synom a vyšlo, že **Y** nie je dostatočne podobná ani s jedným z nich, ale je dosť podobná, aby s oboma osobami vznikol vzťah POTENCIALNI_ZHODA.3.1b

Potrebujeme ešte jednu osobu z tretieho záznamu, nazveme ju **Z**.3.1c Táto osoba je taktiež porovnaná s predchádzajúcimi osobami. Vyjde zhoda s osobou **Y** a prebehne zlúčenie na osobu **YZ**.3.1d

Po zlúčení prebehne kontrola potenciálnych zhôd osoby **YZ**. S informáciami získanými z osoby **Z** je už osoba **YZ** dostatočne podobná na prepojenie s osobou **X**. Prebehne zlúčenie na osobu **XYZ**. Tu nastáva problém, keď sa všetky vzťahy z osoby **YZ** presunú na osobu **XYZ**.3.1e

Keďže prebehlo zlúčenie na osobu **XYZ**, opäť sa skontrolujú potenciálne zhody, vrátane jeho syna **x**. Ako som spomenul na začiatku, osoby boli podobné, takže prebehne zreťazené zlúčenie do jednej osoby **XYZx**. Vzťah JE_OTEC je prepojený na túto osobu a vzniká cyklus dĺžky 1.3.1f

Zreťazené zlúčenie sa nemuselo uskutočniť v jednom kroku, mohlo prebehnúť až pri pridaní ďalšej osoby niekedy v budúcnosti.



(a) Krok 1 (b) Krok 2 (c) Krok 3

(d) Krok 4 (e) Krok 5 (f) Krok 6

Obr. 3.1: Postup vzniku cyklu dĺžky 1

3.2.2 Cykly dĺžky 2 a viac

Okrem cyklov dĺžky 1 môžu vznikať aj dlhšie cykly, teoreticky akejkoľvek dĺžky, avšak odhady dátumov to obmedzujú maximálne na niekoľko generácii. Tieto cykly môžu na

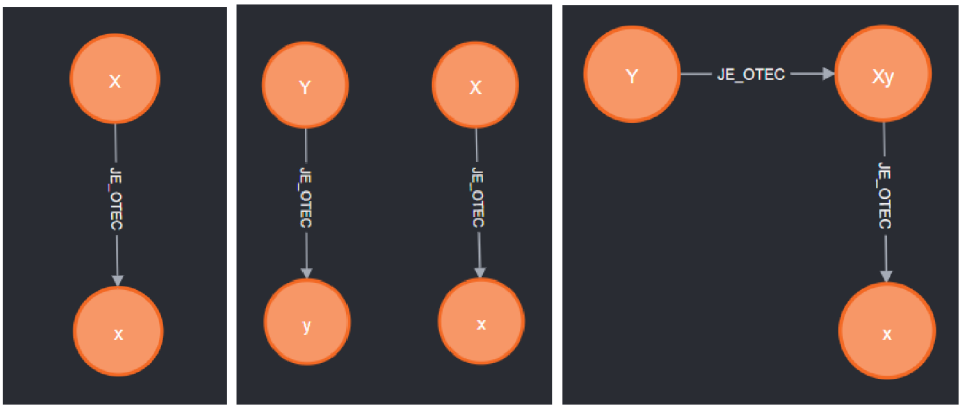
rozdiel od tých dĺžky 1 vzniknúť aj počas klasického zlučovania osôb pri spracovávaní osôb zo záznamu. Najjednoduchší je cyklus dĺžky 2, ktorý si následne aj ukážeme.

Opäť budeme potrebovať dve osoby načítané z jedného záznamu, pri ktorých vieme jednoznačne určiť príbuzenský vzťah **X** a **x**.3.2a

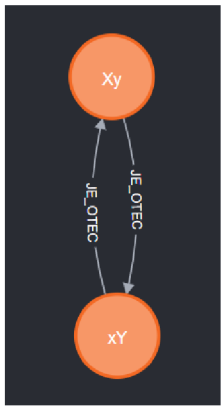
Z druhého záznamu načítame ďalšiu dvojicu osob **Y** a **y** 3.2b, s tým istým vzťahom a prejde sa k porovnávaniu. V podstate je jedno ktorá z načítaných osôb bude porovnávaná ako prvá, v našom prípade to bude **y** a bude porovnávaná s osobami **X** aj **x**. Po klasifikácii vyjde zhoda s osobou **X** a ich následnému prepojeniu do osoby **Xy**.3.2c

Ďalej sa porovná osoba **Y**, tentokrát len s osobou **x**, keďže osoba **Xy** patrí do záznamu, z ktorého pochádza aj osoba **Y**. Po zhode s osobou **x** budú aj tieto osoby zlúčené a ich vzťah previazaný. Novo-vzniknuté osoby **Xy** a **xY** sú si navzájom otcami a tvoria cyklus o dĺžke 2.

Spôsobov ako môžu podobné cykly vzniknúť je viac, vrátane zlučovania pomocou vzťahov POTENCIALNI_ZHODA, podobne ako cykly dĺžky 1.



(a) Krok 1 (b) Krok 2 (c) Krok 3



(d) Krok 4

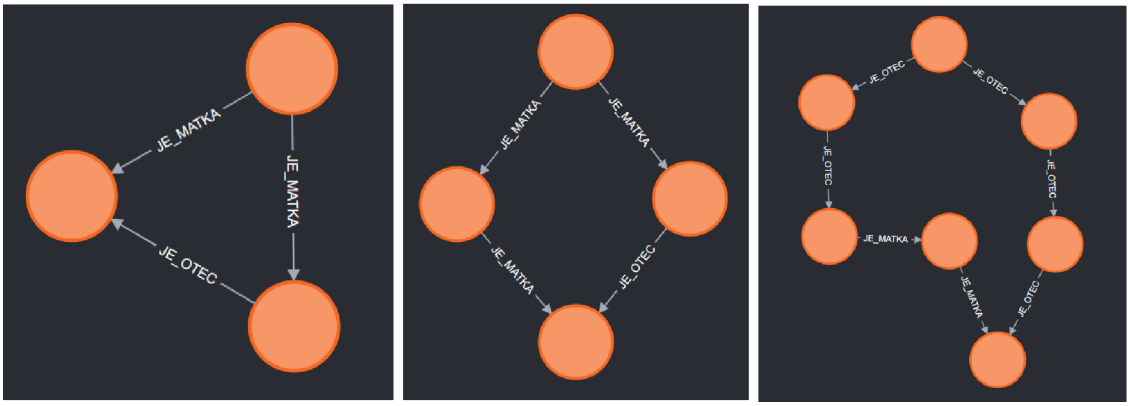
Obr. 3.2: Postup vzniku cyklu dĺžky 2

3.2.3 Ďalšie anomálie

Okrem spomenutých cyklov vzniká aj veľké množstvo iných anomálií, ktoré už nie sú v realite nemožné, ale aj tak sú väčšinou nepravdepodobné a nežiadúce. V podstate sa tiež jedná o cyklus v rodine, ak by sme vzťahy brali ako neorientované hrany grafu. Tieto ano-

málie môžu byť akokoľvek dlhé, počínajúc dĺžkou 3, keďže ich neobmedzujú ani odhady dátumov.3.3

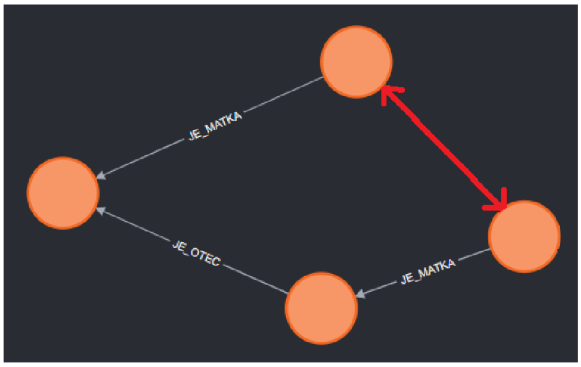
Tiež sa dajú opísať spôsobom, že človek má spoločného predka aj po matkinej aj po otcovej línii.



(a) Dĺžka 3 (b) Dĺžka 4 (c) Dľžka 7

Obr. 3.3: Rôzne dĺžky anomálií

Vznikajú rovnako ako predchádzajúce cykly až na to, že zlúčené osoby nie sú predok a potomok, ale stále patria do širšej rodiny. Na obrázku 3.4 môžeme vidieť stav tesne pred zlúčením a vytvorením anomálie na obrázku 3.3a.



Obr. 3.4: Stav tesne pred vznikom anomálie

3.2.4 Riešenie

Budeme riešiť len vzťahy typu JE_OTEC a JE_MATKA, nakoľko iné vzťahy sa nemusia navzájom vylučovať (niekoho strýko môže byť zároveň jeho kmotor a podobne).

Ako môžeme vidieť, anomálie vznikajú pri zlúčení osôb, ktoré v skutočnosti nehovoria o tej istej osobe, to je štatistika **FP** (false positive). Riešením týchto anomálii je teda čo najviac eliminovať práve tieto prípady. Nakoľko program nikdy nebude 100% presný, nikdy sa nepodari tento problém úplne odstrániť.

Riešenie anomálií sa bude skladať z dvoch častí, treba upraviť klasické porovnávanie a klasifikáciu, a taktiež mierne pozmeniť zlučovanie pri kontrole potenciálnych zhôd.

Ako už bolo spomenuté, pri načítaní nového záznamu sa postupne každá osoba z tohto záznamu porovná s každou ďalšou osobou v DR, ktorá nepatrí práve do tohto záznamu,

pretože sa predpokladá, že tieto osoby majú medzi sebou nejaký vzťah a nemôže sa jednať o tú istú osobu.

Tento spôsob však neberie do úvahy fakt, že po zlúčení prvej osoby z práve spracovávaného záznamu s nejakou už existujúcou v DR, vznikajú nepriame vzťahy medzi osobami z aktuálneho záznamu a skupinou osôb, ktorá bola v DR napojená na práve zlúčenú osobu 3.2.

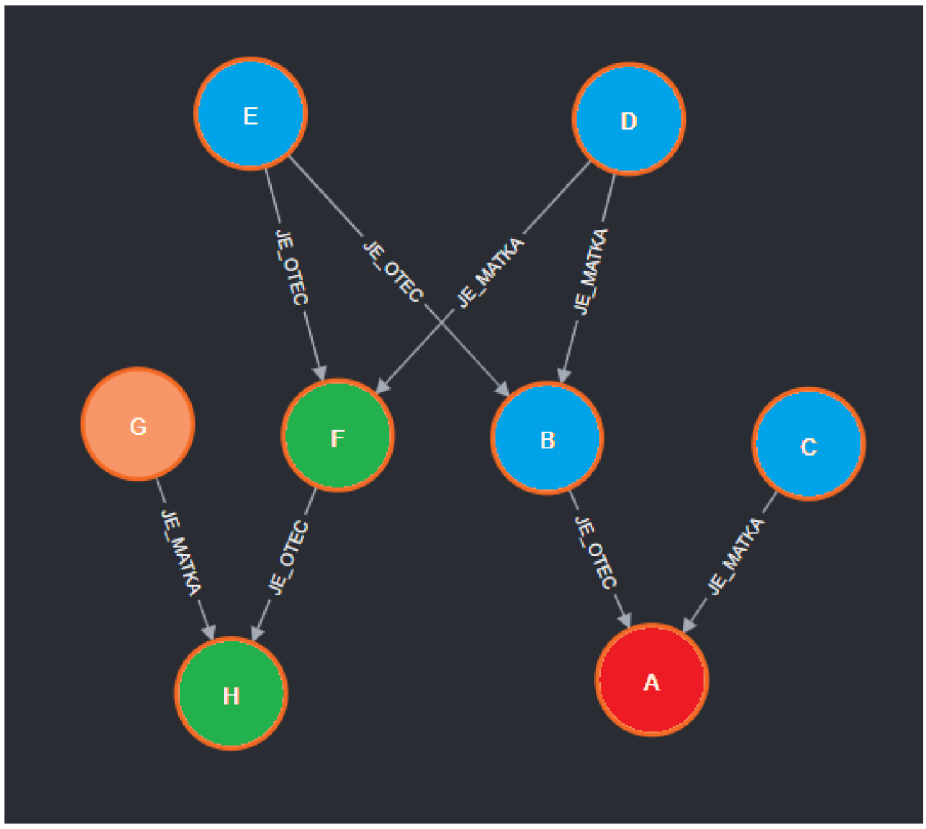
Z tohoto dôvodu zúžime množinu osôb, ktoré sú s osobou porovnávané okrem osôb z toho istého záznamu aj o osoby, ktoré sú na práve porovnávanú osobu napojené nejakými vzťahmi. Tieto osoby budeme hľadať aj rekurzívne.

Dôležitým rozdielom je typ vzťahu. Cykly sú nežiadúce nezávisle na type vzťahu, tak ako si dvaja ľudia nemôžu byť navzájom otcami, nemôžu si byť ani kmotrami. Necyklické anomálie robia problém, len ak sa jedná o vzťahy JE_OTEC a JE_MATKA. Anomálie ako na obrázkoch 3.3 nie sú problém, ak by sa jednalo o vzťah typu JE_KMOTOR.

Aby sme sa zbavili cyklov, stačí ísť po vzťahoch iba jedným smerom. To znamená, že pri osobách, do ktorých sme vošli po smere vzťahu budeme kontrolovať len ďalšie osoby po smere vzťahu. To isté, ale naopak, platí pre osoby, do ktorých sme vošli proti smeru vzťahu. Takéto prehľadávanie bude uplatnené na vzťahy typu JE_KMOTOR, KRSTIL, ODRODILA, POCHOVAL a ZAOPATRIL.

S necyklickými anomáliami je to zložitejšie, pretože môžu vzniknúť z podstatne širšieho okruhu ľudí. Ak by sme sa však pre vzťahy JE_OTEC a JE_MATKA rozhodli ísť vždy oboma smermi, mohli by sme sa dopracovať až k ľuďom, ktorí už s pôvodným človekom nie sú rodina.

Preto treba definovať nejaké obmedzenie, ktoré preruší zahrnutie príliš vzdialenej rodiny. Ja som použil iba ľudí, s ktorými má pôvodný človek buď spoločného predka alebo potomka. To znamená, že strýko (syn starej mamy) pôvodnej osoby je zahrnutý, aj jeho deti (bratranci a sesternice) sú zahrnuté, ale ich matka nie, lebo nemá s pôvodnou osobou spoločného predka.



Obr. 3.5: Vylúčené osoby

Na grafe 3.5 vidíme pôvodnú osobu červenou farbou, modrou farbou sú vyznačení predkovia a zelenou osoby stále zahrnuté do vylúčenia z prehľadávania, lebo majú s červenou osobou spoločného predka. Oranžová osoba už zahrnutá nieje.

To isté platí aj z druhej strany, zahrnutý je švagor aj s jeho rodičmi, lebo zdieľajú vnuka, ale švagrov brat už zahrnutý nie je. Jediné výnimky sú osoby na rovnakej úrovni ako aktuálna osoba (bratia a sestry) ktoré môžu v skutočnosti hovoriť o porovnávanej osobe.

Táto úprava zaručí kompletné zabránenie vzniku cyklov a zníženie počtu necyklických anomálii pri klasickom zlučovaní osoby. Tieto osoby bude potrebné nájsť pre každú porovnávanú osobu zvlášť, keďže sa rodina môže v každom kroku zmeniť.

Aby sme úplne zabránili vzniku cyklov, treba ešte zaistiť, že pri zlučovaní potenciálnych zhôd nedôjde k stavu ako na obrázku 3.1e, t.j. že sa vzťah POTENCIALNI_ZHODA prenesie na príbuznú osobu.

K zaisteniu tejto skutočnosti využijeme ten istý prístup ako pri klasickom zlučovaní. Nájdenú rodinu akurát nebudeme vylučovať z porovnávania, ale pri preväzovaní vzťahov skontrolujeme, či osoba, na ktorú sa má vzťah napojiť, nepatrí do získanej množiny s príbuznými.

Menší problém, ktorý vzniká je, že dve osoby, ktoré majú spoločného predka po oboch rodičovských líniách, nemusia byť chyba. Budeme však predpokladať že takáto situácia nastáva veľmi zriedkavo. Riešením by mohlo byť určenie maximálnej hĺbky predkov, dáta k určeniu tejto hodnoty nemám k dispozícii.

Ďalším problémom (ktorý stále zostáva) je chybovosť programu. Pri každom nesprávnom prepojení sa dve rodiny môžu spojiť a vytvoriť väčšiu. Keďže je táto rodina väčšia, má väčšiu pravdepodobnosť, že sa na aspoň jedného z jej členov napojí ďalšia nesprávna rodina.

Ukázalo sa, že tento jav spôsobí, že na konci behu programu je vyše polovica všetkých uzlov osôb prepojená do jednej obrovskej rodiny. To môže spôsobiť, že aj osoby, ktoré by mali byť prepojené, nebudú vo výsledku ani porovnané. Obmedzenie na ľudí so spoločným predkom alebo potomkom to čiastočne zmierňuje, ale nerieši kompletne.

Nutnosť prehľadávania grafov tiež spôsobí spomalenie výsledného programu.

3.3 Wildcards

Zo sčítacích operátov vyplynulo, že vydaté ženy, ktorým sa zmenilo priezvisko, tvoria nemalú časť **FN**. Je to zapríčinené tým, že priezvisko má vysokú váhu a jeho nezhoda spôsobí značný pokles podobnosti. Nedostatok informácií o rodičoch zároveň väčšinou zabraňuje možnosti porovnania predkov a navýšenia podobnosti týmto spôsobom.

Návrhom, ako tento problém čiastočne vyriešiť, bolo neporovnávať priezvisko vydatých žien. Zmena programu, aby ignoroval určité atribúty, by bola problematická. Preto som sa rozhodol pridať znak '*' ako „wildcard“ niektorým atribútom. Ak funkcie v module `comparator` narazia na '*' ako jednu z hodnôt, automaticky vrátia podobnosť porovnávaného atribútu rovnú 1.

Keď pri načítaní osoby ženského pohlavia vieme, že je vydatá, do atribútu `surname` sa okrem aktuálneho priezviska pridá aj '*'. Pri vkladaní do DR sa funkciou `remove_wildcards` tieto hodnoty odstránia, aby neboli natrvalo uložené v databáze.

Ako sa však ukázalo pri testovaní, okrem zníženia **FN** sa približne rovnakým pomerom zdvihol aj počet **FP**. Keďže **FP** pôsobia v databázy väčšie problémy ako **FN**, táto úprava nie je využitá napriek tomu, že je implementovaná.

Kapitola 4

Implementácia

Vzhľadom k tomu, že na práci už predo mnou robili viacerí ľudia, výsledná čitateľnosť a konzistencia kódu neboli v najlepšom stave. Nepomáhalo tomu ani absolútne minimum komentárov a fakt, že python je dynamicky typovaný jazyk, takže často ani nebolo jasné, v akom formáte funkcie očakávajú svoje argumenty. Narazil som aj na množstvo malých chýb a nedostatkov, či už po logickej alebo technickej stránke.

Preto sa bude prvá časť implementácie zameriavať na tie najpodstatnejšie úpravy, ktoré som spravil. Taktiež sa pozrieme na tvorbu indexov do grafovej databázy, aby sme podstatne zrýchlili vkladanie údajov z DR.

Ďalej si ukážeme zmeny, potrebné pre pridanie ďalších typov záznamov, vrátane úpravy reprezentácie vzťahov v DR.

Nakoniec sa pozrieme na implementáciu výpočtu váh podľa modelu Fellegi-Sunter, ktorá v programe chýbala a čiastočné riešenie problému s anomáliami, ktoré sme si ukázali v sekcii 3.2.

4.1 Oprava dátumov

V relačnej databáze sú všetky dátumy ukladané ako `VARCHAR(255)` z dôvodu, že nie všetky záznamy majú rovnaký formát dátumu. Niekde je dátum celý, ale často sa stretneme s tým, že v zázname je spomenutý iba rok. Zároveň nie všetky dátumy musia byt korektné, či už z dôvodu chyby prepisu, alebo záznam naozaj obsahuje neplatný dátum. V niektorých prípadoch môžu obsahovať aj znak '**?**' ak bol dátum úplne nečitateľný.

V programe sa pracuje s dátumami typu `datetime.date` z dôvodu jednoduchších operácií a podpory vstavaných funkcií. Kvôli vyššie spomenutým problémom nie je možné implicitne prevádzať tieto reťazce na dátumy.

Na tento účel bola v existujúcom programe implementovaná oprava dátumov. Ak sa v dátume nachádzal '**?**', dátum bol neplatný alebo nekompletný, prebehla drobná úprava. Táto kontrola bola roztrieštená do niekoľkých funkcií s množstvom duplicitného kódu a pre dáta z relačnej databázy sa používala iná funkcia ako pre dáta z csv súborov. Tieto funkcie zároveň neposkytovali tie isté výsledky a funkcia použitá v csv nevedela spracovať niektoré neplatné hodnoty a vyvolávala výnimku.

Riešením bolo vytvoriť jednu funkciu `fix_date`, ktorú je možné použiť pri všetkých zdrojoch, zohľadňujúcu všetky eventuality, vrátane tých neplatných. Manuálnu kontrolu počtu dní v mesiaci som nahradil pomocou balíčku `calendar` a funkcie `monthrange`, ktorá pre vybraný mesiac a rok vráti počet dní v tomto mesiaci. Táto možnosť zároveň korektne

zohľadňuje prestupné roky na rozdiel od predchádzajúcej verzie, kde sa pre február vždy očakávalo maximálne 28 dní.

4.2 Určovanie geografickej polohy

Ďalším problémom bolo zisťovanie súradníc mesta. Názov mesta sa klasicky načíta z databázy z tabuľky `domicile`, ale súradnice sa nachádzajú v separátnych json súboroch. Prvým problémom je, že json súborov môže byť viac (pre rôzne kraje, ...) a formát týchto súborov nie je jednotný. Z tohto dôvodu sa pre každý súbor musela používať iná funkcia. V týchto súboroch sa taktiež nachádzali milióny zbytočných riadkov s informáciami, ktoré sú pre program úplne zbytočné.

Problém som odstránil normalizáciou súborov a odstránením všetkých zbytočných informácií. Vďaka tomu sa môžu pridávať nové súbory bez potreby úpravy samotného programu za predpokladu, že sú v normalizovanom formáte. Súbory sa pritom zmenšili zo stoviek MB na stovky KB, ktoré musia byť počas celého behu programu načítané v pamäti.

Druhým problémom bol výkon. Nakoľko json súbory obsahovali list slovníkov a názov a súradnice mesta sa nachádzali v tomto slovníku, pri načítaní každého mesta sa musel celý tento list prezrieť, aby sa skontrolovalo, či obsahuje súradnice požadovaného mesta. Kvôli množstvu prístupov je tento spôsob absolútne nevhodný. Jednoduchou úpravou na slovník slovníkov, kde názvy miest použijeme ako kľúče, sa dostávame z lineárnej zložitosti na konštantnú. Vďaka týmto zmenám sa celková doba behu programu na testovacích dátach znížila približne o 25 %.

4.3 Neplatné odhady dátumov

Pri analýze finálnych dát uložených v grafovej databáze som narazil na osoby, ktoré mali neplatný interval odhadu dátumu narodenia alebo úmrtia a to $\langle a, b \rangle$ kde $a > b$. Napríklad mohli vzniknúť osoby, ktoré mali odhad narodenia od 1.1.1890 do 1.1.1880, čo nedáva zmysel.

Ukázalo sa, že problém nenastával pri zlučovaní osôb, ale pri samotnej kontrole, či by sa osoby mali porovnávať. Nekontrolovalo sa, či sa intervaly narodenia a úmrtia prekrývajú, ale iba či aktuálny záznam pochádza z doby života porovnávanej osoby.

Kvôli tomu sa mohli porovnať, a vo výsledku aj zlúčiť, osoby, o ktorých sme vedeli, že sa nenarodili, alebo neumreli v tom istom období. Z toho mohli vzniknúť spomenuté neplatné intervaly. Napríklad ak sa spojili osoby, ktoré sa narodili v intervaloch $\langle 1.1.1800, 1.1.1810 \rangle$ a $\langle 1.1.1850, 1.1.1870 \rangle$, interval pre zlúčenú osobu bol $\langle 1.1.1850, 1.1.1810 \rangle$.

Úprava funkcie rozhodujúcej či sa budú osoby porovnávať tak, aby sa intervaly narodenia aj úmrtia museli aspoň čiastočne prekrývať vyriešila tento problém, spolu s miernym zlepšením výkonu, vďaka nižšiemu počtu porovnávaných osôb a miernym zlepšením presnosti vďaka zníženiu počtu **FP**.

4.4 Zbytočné deep kópie

Pri načítaní osoby z relačnej databázy sa dáta ukladajú do rovnakých štruktúr, aké sú použité pre ukladanie uzlov v DR. Pri vkladaní uzlov do DR sa vždy vytvárala deep kópia načítaného objektu.

S menšími úpravami toto vôbec nie je nutné a do DR sa môžu rovno ukladať objekty, ktoré vzniknú pri načítaní z databázy. Keďže sa vo finále každý objekt vytvára iba raz namiesto 2x, táto zmena mierne zlepšila výkon.

4.5 Porovnávanie bydliska

Pri odstraňovaní zbytočných kópií som tiež našiel problém s objektom `domicle`, ktorý obsahuje informácie o bydlisku osoby (mesto, ulica, popisné číslo). Atribúty obsahujúce názov ulice a číslo sa pri vkladaní do DR nahradili párom `(ulica, id)` a `(číslo, id)`.

Deje sa to z dôvodu, že DR ukladá mestá, ulice a čísla v slovníku, ako je ukázané v Kroku 6 2.3.4, nie v objekte `domicile`. Ulica ani číslo pri vytvorení `domicile` neobsahujú unikátne id, ktoré je potrebné pre finálne prepojenie v grafovej databáze.

Ak pri vkladaní do DR ulica alebo číslo ešte neexistovali, vygenerovalo sa nové id a to sa následne uložilo aj v slovníku aj v štruktúre `domicile` ako vyššie spomenutá dvojica. Ak už DR ulicu alebo číslo obsahovala, iba sa uložili už existujúce id.

Pri porovnávaní sa na túto skutočnosť bral ohľad a používal sa pristúp `street[0]`, ktorý v tomto prípade správne vybral názov ulice a `street_number[0]`, čo tiež správne vracalo číslo. Problém bol, že sa rovnaké pravidlo uplatňovalo aj na bydlisko pravé načítané z databázy, ktoré ešte nemalo prepísané atribúty ulice a čísla, pretože ešte neprebehlo vloženie do DR.

V tom prípade sa z ulice zobralo iba prvé písmeno a z čísla iba prvá číslica. Toto spôsobovalo čiastočné skreslenie výsledkov, nakoľko sa nikdy neporovnávala skutočná ulica ani číslo.

Prepisovať niečo, čo pôvodne obsahovalo len názov, na dvojicu je veľmi mätúce. Rozhodol som sa do `domicile` a do slovníku v DR pridať ďalšie dva atribúty `street_id` a `number_id`, do ktorých budú id uložené namiesto atribútu pre meno ulice a číslo. Vyriešila sa tým aj spomenutá chyba a ulice a čísla sa porovnávajú celé, vďaka čomu sa zlepšila úspešnosť programu.

4.6 Úprava vyhodnotenia porovnávania

Správa porovnávania prebieha vo funkcii `get_result_of_comparison` a `handle_result_of_comparison` modulu `create_database`, kde sa rozhoduje, ktoré osoby budú porovnávané a kontroluje sa výsledok tohto porovnania. Funkcia bola implementovaná tak, že osoby boli postupne porovnávané a všetky, ktoré dosiahli podobnosť aspoň `precision_potential`, boli ukladané do listu. Ak sa však našla osoba, ktorá dosahovala hodnoty `precision_match`, bola ihneď vrátená ako osoba, ktorá má byť zlúčená.

Tento postup by bol správny za predpokladu, že program pracuje 100 % korektne a každá osoba sa v DR nachádza maximálne raz a dve rozdielne osoby nikdy nedosiahnu požadovanú hranicu pre zlúčenie.

Keďže tento predpoklad neplatí, viac porovnaných osôb môže dosiahnuť požadovanú hodnotu `precision_match`. Vybratím prvej takejto osoby zamietame možnosť, že by sa v DR mohla nachádzať ďalšia dostatočne podobná osoba, možno aj s väčšou podobnosťou než prvá vybratá.

Preto som sa rozhodol túto funkciu pozmeniť, aby nevyberala prvú osobu, ale osobu s najvyššou úrovňou podobnosti nad požadovanú hranicu.

Tak ako sa doteraz osoby s podobnosťou nad `precision_potential` ukladali do listu, po novom to bude platiť aj pre osoby s podobnosťou nad `precision_match`. V liste budú uložené ako dvojica (`tuple`), kde prvá hodnota obsahuje podobnosť a druhá je porovnávaná osoba. Funkcia `get_result_of_comparison` vždy vráti celé pole podobných osôb a vo funkcii `handle_result_of_comparison` sa nájde tá s najvyššou podobnosťou. Ak je táto hodnota vyššia než `precision_match` prebehne zlúčenie, inak sa všetky osoby z listu pripoja ako potenciálne zhody.

Pracoval som aj s možnosťou zlúčiť všetky osoby s podobnosťou nad `precision_match`, ukázalo sa však, že to nie je možné z dôvodu, že po zlúčení s prvou osobou, novo-zlúčená osoba nemusí dosahovať požadovanú podobnosť. Aby kvôli tomu nevznikali nesprávne výsledky, bolo by nutné všetky porovnania prepočítať, čo by zbytočne spomalilo cely program a pravdepodobne neprinieslo zlepšené výsledky.

Ďalšou variantou bolo prepojiť všetky zostávajúce osoby ako potenciálna zhoda, aj keď sme našli úplnú zhodu. Pri testovaní to spôsobilo zvýšenie počtu **TP**, ale celkové výsledky boli znateľne horšie. Taktiež to zapríčinilo podstatne vyšší výskyt anomálií ako pri pôvodnom spôsobe.

Implementácia listom je dostatočná vzhľadom na celkom nízky počet prvkov a na fakt, že najväčší prvok potrebujeme nájsť iba raz a taktiež, že na usporiadaní zbytku listu nezáleží. Použitie `PriorityQueue` alebo podobného kontajneru je preto zbytočné a vo výsledku by fungovala podobne rýchlo alebo ešte pomalšie.

4.7 Pridanie indexov do grafovej databázy

Pri podrobnejšom skúmaní času jednotlivých častí programu vyšlo, že približne 75 % behu zaberá len vkladanie vzťahov do grafovej databázy 4.1.

Fáza	Čas	Percentuálny podiel
Porovnávanie	231.78 s	19 %
Vkladanie uzlov	74.86 s	6 %
Vkladanie vzťahov	917.91 s	75 %

Tabuľka 4.1: Časové rozloženie programu

Toto je spôsobené faktom, že pri vkladaní každého vzťahu sa musia v grafovej databáze vyhľadať už uložené uzly, ktoré má daný vzťah prepájať. Tak ako relačná databáza aj grafová databáza podporuje indexy nad požadovanými atribútmi. Bolo treba vytvoriť index nad každým atribútom, ktorý bol použitý počas vkladania vzťahov.

Tieto atribúty sú:

- `Záznam_o_křte.id_rel_databáza`
- `Záznam_o_úmrti.id_rel_databáza`
- `Záznam_o_svatbe.id_rel_databáza`
- `Sčítací_operát.id_rel_databáza`
- `Lánový_rejstřík.id_rel_databáza`
- `Poddanská_přiznávací_fase.id_rel_databáza`
- `Urbar.id_rel_databáza`
- `Pozemková_kniha.id_rel_databáza`
- `Osoba.id`
- `Matrika.ID_matrika`

- `Mesto.id`
- `Popisné_číslo.id`
- `Ulica.id`

Treba si uvedomiť, že tieto id nie sú identifikačné čísla samotných uzlov generované grafovou databázou, nad ktorými sú indexy automaticky vytvorené, ale iba jedna z hodnôt uložených v každom uzle poskytnutá našim programom.

Pri napojení na grafovú databázu vždy prebehne kontrola, či v nej požadované indexy existujú a ak nie, tak sa automaticky vytvoria. Syntax takého dotazu vyzerá napríklad takto:

```
CREATE INDEX osobaIndex IF NOT EXISTS FOR (o:Osoba) ON (o.id)
```

Vytvorenie indexov v grafovej databáze urýchlilo fázu vkladania vzťahov o takmer 80 % a celkový beh programu o vyše 60 %.

4.8 Ručná úprava intervalov

Nové pridané záznamy (aj niektoré už existujúce) obsahujú, alebo z ich dát vyvodzujeme dátumy, ktoré nie sú úplne presné. Napríklad pri sčítacích operátoch máme pri osobách aj dátum narodenia. Tento dátum je často určený iba rokom, a aj tak tento rok nemusí byť zhodný pre všetky záznamy obsahujúce tu istú osobu.

V záznamoch o sobášoch sú informácie o veku ženícha a nevesty, tieto údaje sú taktiež často určené len rokom, z ktorého počítame dátum narodenia, ktorý je považovaný za presný napriek tomu, že možné rozpätie je minimálne rok.

Funkcia `update_date_guess` prijíma dátum a vzťah osoby k tomuto dátumu. Na základe preddefinovaných pravidiel určí, alebo zúži, intervaly narodenia a úmrtia. Považovanie dátumu za presný môže spôsobovať, že niekedy záznamy o zhodnej osobe nemusia byť porovnané na základe toho, že sa ich intervaly nemuseli prekrývať.

Napríklad ak máme v DR záznam o narodení osoby z 15.8.1870 a prišiel nám záznam o sčítaní, kde mala tá istá osoba uvedený dátum narodenia len ako 1870, tento dátum by bol upravený na validný dátum 1.1.1870. Intervaly narodenia vzniknuté z týchto dátumov sa neprekrývajú a tieto osoby nebudú porovnané a tým pádom ani zlúčené.

Aby bol možný zásah do tvorby intervalov, rozšíril som funkciu `update_date_guess` o ďalší voliteľný parameter `inaccuracy` typu `relativedelta`. Bude sa využívať pri dátumoch, pri ktorých vieme, že sú nepresné a vieme odhadnúť o koľko. Vypočítané intervaly budú zľava aj sprava rozšírené o túto hodnotu.

Pre dátumy narodenia pri sčítacích operátoch som použil hodnotu dva roky, čo vo výsledku znamená rozšírenie intervalu o štyri roky. Pri záznamoch sobášov a veku oddaných, používam buď jeden rok alebo jeden mesiac podľa poskytnutých informácií. Rovnakým spôsobom som upravil aj niekoľko ďalších dátumov v rôznych záznamoch.

4.9 Nové typy záznamov

Algoritmus som upravil tak, že pre pridanie nového záznamu je potrebné, okrem drobných úprav v niektorých funkciách, ktoré rozhodujú o behu programu, pridať štyri až päť funkcií. Ako prvé funkciu `get_record_x`, ktorá načíta záznam z databázy a funkciu `get_persons_x` pre načítanie údajov z tabuliek o osobe. Ak sa v zázname nachádzajú nejaké údaje špecifické

pre nejakú osobu, treba aj funkciu `extract_person_info_from_record_x`. Všetky tieto funkcie v podstate len extrahujú dáta zo slovníkov získaných z relačnej databázy.

Ďalej je potrebná funkcia `guess_date_according_to_role_x`, ktorá pre všetky osoby určí intervaly života na základe roly osoby v zázname. Poslednou funkciou, ktorá je volaná až po načítaní všetkých osôb je `create_connection_with_person_in_x_record`, kde sa všetky osoby a samotný záznam spolu poprepájajú.

Okrem týchto funkcií treba rozšíriť `RelationType enum` o všetky nové vzťahy, ktoré sa v zázname môžu vyskytnúť.

4.10 Vzťahy v DR

Vzťahy v DR boli reprezentované slovníkom, ktorý obsahoval odkaz na pripojený uzol a ďalšie informácie ako napríklad úroveň podobnosti alebo dátum. Tieto slovníky boli okrem vzťahov POTENCIALNI_ZHODA uložené v štruktúre `Person` v atribútoch `relationships_from` (vzťahy vychádzajúce z uzla) a `relationships_to` (vzťahy vedúce do uzla). To znamená, že vzťah je uložený aj v uzle, z ktorého vychádza aj v uzle, do ktorého vchádza.

To znamenalo, že pre každý vzťah existovali dva slovníky, jeden uložený v uzle, z ktorého vzťah vychádzal a druhý v uzle, do ktorého vchádzal. Tieto dva slovníky o sebe nevedeli, čo znamená, že ak sme potrebovali vzťah odstrániť alebo ho prepojiť na iné uzly, museli sme nájsť oba napojené uzly a v cykle v nich vyhľadať požadované slovníky vzťahu.

Pre uľahčenie manipulácie som sa rozhodol vytvoriť novú štruktúru `Relation`, ktorá bude reprezentovať všetky vzťahy, ktoré sa v DR môžu nachádzať. Narozdiel od predchádzajúceho slovníka obsahuje uzol, z ktorého vychádza a aj uzol, do ktorého vchádza, spolu so všetkými potrebnými informáciami.

Ďalej som upravil atribúty `relationships_from` a `relationships_to` z listov na slovníky, kde je ako kľuč použité id opačného uzla (`relationships_from` používa ako kľúče id uzla do ktorého vchádza, `relationships_to` používa ako kľúče id uzla z ktorého vychádza), čo umožňuje jednoduché vyhľadávanie.

V štruktúre `Relation` som vytvoril funkciu `rebind`, ktorá je schopná zmeniť odkiaľ a kam vzťah vedie vďaka pomocným funkciám `add_relationship` a `remove_relationship` z `Person`.

Teoreticky môže medzi konkrétnymi dvoma osobami naraz existovať viac vzťahov, takže je atribút `Relation.type` reprezentovaný ako množina (`set`), ktorá môže obsahovať hodnoty z `RelationType`.

Vzťahy potenciálna zhoda sú vždy obojstranné, takže ich nie je nutné ukladať do dvoch atribútov. Atribút `potential_matches` som však tiež zmenil na slovník, aby bola ich forma konzistentná s inými vzťahmi.

4.11 Výpočty váh – Fellegi-Sunter

Váhy ako sú ukázané v sekcii 2.3.2 neboli implementované. Namiesto nich sa používali ručne definovane váhy pre jednotlivé atribúty.

Preto som sa rozhodol implementovať funkciu, ktorá by z klasifikovaných vstupných dát vypočítala hodnoty váh, ktoré bude neskôr možné použiť. Využije pri tom koncepty spomenuté pri pravdepodobnostnej klasifikácii.

Keďže náš komparátor nedáva binárne hodnoty (zhoda/nezhoda) ale podobnosť na škále $\langle 0, 1 \rangle$, využívam rozšírenú verziu, ktorá počíta váhy pre špecifické úrovne podobnosti [5].

Hodnoty m_i a u_i si musíme definovať pre každú úroveň podobnosti δ (δ používam ako podobnosť, nie rozdiel).

$$m_i(\delta) = \frac{\text{počet zhodných dvojíc osôb s hodnotou atribútu } i \text{ podobnou na } \delta}{\text{počet zhodných dvojíc osôb}}$$
$$u_i(\delta) = \frac{\text{počet nezhodných dvojíc osôb s hodnotou atribútu } i \text{ podobnou na } \delta}{\text{počet nezhodných dvojíc osôb}}$$

Váha atribútu i pri podobnosti δ sa potom z týchto hodnôt počíta rovnako ako v pôvodnej verzii.

$$w_i(\delta) = \log\left(\frac{m_i(\delta)}{u_i(\delta)}\right)$$

Vzhľadom k tomu, že δ patrí do množiny reálnych čísel na intervale $\langle 0, 1\rangle$, musíme niektoré hodnoty zlúčiť. Implementoval som to tak, že je interval rozdelený na predom určený počet rovnako veľkých dielov, pričom sa všetky hodnoty z týchto sub-intervalov agregujú do jednej.

Funkcia `calculate_attribute_weights_fellegi_sunter` musí najprv porovnať všetky osoby zo všetkých záznamov medzi sebou a v slovníku si pre každý atribút počíta výskyty podľa vypočítanej podobnosti.

V slovníku je pre každý atribút dvojica listov o veľkosti počtu dielikov, prvé pole obsahuje počty výskytov pre záznamy, ktoré hovoria o tej istej osobe a druhé o rozdielnych osobách. Taktiež musíme spočítať celkový počet zhodných a nezhodných dvojíc.

Z týchto hodnôt sa vypočíta *prior*, čo je počiatočná pravdepodobnosť, že sa dva náhodne vybraté záznamy zhodujú. To sa vypočíta ako pomer zhodných dvojíc osôb ku všetkým dvojiciam, na ktorý ešte aplikujeme logaritmus ako pri výpočte váh [9].

$$prior = \log\left(\frac{\text{počet zhodných dvojíc osôb}}{\text{počet dvojíc osôb}}\right)$$

Všetky vypočítané hodnoty sú nakoniec uložené do json súboru, z ktorého sa môžu následne načítať a aplikovať pri porovnávaní, bez nutnosti opakovaného výpočtu vzhľadom na jeho časovú náročnosť, ktorá je takmer dve hodiny.

Pri využití týchto váh, ktoré nie sú normalizované do intervalu $\langle 0, 1\rangle$ a zároveň môžu byť aj záporné, nám po sčítaní porovnávacieho vektoru γ a hodnoty *prior* vyjde výsledok v „log-odds“ (logit) formáte[9, 16].

Aby sme dostali pravdepodobnosť, že sú záznamy zhodné, musíme „log-odds“ previesť pomocou štandardnej logistickej funkcie [15].

$$prob = \frac{1}{1 + 2^{-odds}}$$

To nám vráti pravdepodobnosť v intervale $\langle 0, 1\rangle$, ktorú môžeme porovnať prahmi t_u a t_l ako pri ručne definovaných váhach. Porovnanie všetkých vypočítaných váh, je zhrnuté v kapitole 5.

4.12 Prevencia cyklov

Na vyhladanie osôb, ako bolo popísané v sekcii 3.2.4, som do štruktúry `Person` pridal funkciu `get_family_relatives`, ktorá do šírky prehľadáva rodinu osoby, nad ktorou bola zavolaná. Všetky nájdené osoby ukladá do množiny (`set`) a tie na rovnakej úrovni ukladá aj do separátnej množiny, aby mohli byť nakoniec odstránené.

Funkcia je volaná pred začiatkom porovnávania novej osoby, aby mohli byť nežiadúce osoby odstránené z množiny všetkých osôb z DR pripravených na porovnanie. Funkcia `unite_potential_matches_nodes` po previazaní vzťahov POTENCIALNI_ZHODA využíva získanú množinu pre odstránenie tých vzťahov, ktoré vedú na člena rodiny.

4.13 Úprava skriptov

Skripty `get_person.py`, `get_all_records.py` a `get_family_tree.py` boli rozšírené o nové typy záznamov. Bolo potrebné opraviť aj zopár chýb ako napríklad ukladanie všetkých dátumov ako reťazce namiesto `datetime`, pretože tieto dátumy boli pri načítaní uložené ako typ `neo4j.time.Date`, ktorý má iný formát než bolo očakávané.

Upravené je aj spracovávanie vstupných argumentov za pomoci modulu `argparse`, ktorý umožňuje podstatne prehľadnejšie spracovanie a ponúka ďalšie možnosti ako automatická kontrola povinných argumentov a pridanie prepínaču `-h` pre vypísanie definovanej pomôcky k spusteniu.

4.14 Konfigurácia

Vzhľadom na celkom veľký počet možných nastavení programu, ktoré boli rozdelené do rôznych súborov a funkcií, som pre jednoduchšiu manipuláciu a možnosť ovplyvniť fungovanie programu bez zásahu do kódu, pridal `config.json` súbor. Do neho som zoskupil väčšinu nastaviteľných atribútov programu. Dajú sa v ňom upraviť prahy pre zhody a potenciálne zhody, vstupný zdroj, ktoré typy záznamov sa majú spracovať, či sa majú použiť vypočítané váhy zo separátneho súboru a zopár ďalších menších nastavení.

Samotný program sa spúšťa buď bez argumentov, vtedy sa použije predvolený súbor `json/config.json`, alebo s prepínačom `-c, --config` nasledovaný cestou k vlastnému json súboru.

Kapitola 5

Testovanie

Táto kapitola sa venuje testovaniu výsledného programu na poskytnutej dátovej sade. Porovnáme úspešnosť a rýchlosť po väčšine úprav a pozrieme sa ako výsledky ovplyvňuje prevencia anomálií a použitie nových vypočítaných váh pomocou modelu Fellegi-Sunter.

5.1 Štatistiky odrážajúce stav grafovej databázy

Štatistiky ukázané v sekcii 2.3.3 slúžia celkom dobre a hovoria nám o tom, s akým výsledkom prebehli jednotlivé porovnávania. Nezistíme z nich však stav, v akom sa výsledná databáza nachádza. Nevieme, koľko uzlov osôb sa v nej nachádza, ani v akom stave sú jednotlivé uzly a osoby.

Preto som pridal štatistiku, ktorá na konci skontroluje DR a uzly osôb v nej. To prebieha na základe predom klasifikovaných id, ktoré hovoria, či sú osoby zhodné alebo nie. Tieto id sú využité aj pri výpočtoch predchádzajúcich štatistík. Keďže sa po zlúčení uzlov uchovávajú všetky tieto id, vieme pri pohľade naň zistiť, či bol zlúčený s nesprávnou osobou.

Zaviedol som štyri stavy, v akých sa osoba môže nachádzať.

- perfect (**P**) - id osoby sa v DR nachádza len raz a tento uzol obsahuje iba toto id
- split (**S**) - id osoby sa v DR nachádza viac krát, ale všetky uzly obsahujú len toto id
- mixed (**M**) - id osoby sa v DR nachádza len raz, ale obsahuje aj id inej osoby
- split_mixed (**SM**) - id osoby sa v DR nachádza viac krát a aspoň jeden z jej uzlov obsahuje aj id inej osoby

Pri každej kategórií budem zároveň zisťovať aj počet uzlov v databáze, ktoré patria do týchto skupín. Počítam ich hlavne kvôli faktu, že všetky osoby z mixed kategórie by sa vo výsledku mohli zlúčiť do jedného uzla.

5.2 Dáta

Dáta využité pri porovnávaní pochádzajú z csv súborov. Okrem pôvodných datasetov obsahujúcich matriky narodených, oddaných a zosnulých som mal k dispozícii aj klasifikované záznamy o sčítacích operátoch, poddanskej priznávacej fase a pozemkových knihách.

Záznam	Počet
Matriky narodených	2 251
Matriky zosnulých	1 206
Matriky oddaných	471
Sčítacie operáty	551
Fase	43
Pozemková kniha	125
Σ	4 647

Tabuľka 5.1: Počty jednotlivých typov záznamov

Ako je vidieť v tabuľke 5.1, spolu je testovaných vyše 4 600 záznamov, ktoré dávajú viac ako 30 000 záznamov individuálnych osôb.

5.3 Porovnanie na pôvodnej sade

Aby sme zistili, ako úspešný je program po implementovaných úpravách v porovnaní s pôvodnou verziou, využil som pôvodnú testovaciu sadu a prahy 0,9 pre zhodu a 0,7 pre potenciálnu zhodu a otestoval som obe verzie.

	TP	FP	TN	FN	Prec.	Rec.	F1	Bal. acc	Čas
Pred	11 805	2 713	6 686 689	13 050	0,813	0,475	0,600	0,737	3 213,90 s
Po	14 113	1 659	791 535	4 739	0,895	0,749	0,815	0,873	764,58 s

Tabuľka 5.2: Porovnanie programu

Ako môžeme vidieť v tabuľke 5.2, všetky aspekty programu sa aspoň čiastočne zlepšili. Hodnota **TN** klesla takmer na desatinu vďaka opravenej kontrole intervalov života porovnávaných osôb. Hlavne vďaka tomu a indexom v grafovej databáze sa skrátil aj čas behu programu viac než 4 krát.

5.4 Porovnanie rôznych verzií prevencie anomálií

Ako bolo spomenuté v sekcii 3.2 vo výsledných dátach môžu vznikať rôzne anomálie. V tejto sekcii sa pozrieme na to, ako sa prejaví využitie rôznych metód snažiacich sa im zabrániť. Konkrétne sa pozrieme na výsledky pri vypnutej prevencii (none), prevencii za použitia iba predkov a potomkov (anc_desc) a prevencii pri využití ľudí, s ktorými má pôvodná osoba spoločných predkov alebo potomkov (full). Testovanie prebiehalo na celej dátovej sade ako je spomenutá v sekcii 5.2 s váhami 0,9 a 0,7.

	Prec.	Rec.	F1	Bal. acc	Čas	Anom.
none	0,889	0,749	0,813	0,873	1 161,45 s	597
anc_desc	0,888	0,749	0,813	0,873	1 165,56 s	390
full	0,883	0,748	0,810	0,873	1 230,16 s	116

Tabuľka 5.3: Porovnanie pri rôznych prevenciách proti anomáliám

Je vidieť, že využitie prevencie nemá takmer žiadny vplyv na úspešnosť finálneho porovnávania, avšak s každou ďalšou úrovňou prevencie klesá množstvo anomálií až na približne

pätinu pôvodného počtu. Kvôli väčšiemu počtu osôb potrebných pri prevencii postupne rastie aj čas, ale veľmi zanedbateľným tempom. Drobné zníženie presnosti je spôsobené občasnou prevenciou porovnania aj správnych osôb.

5.5 Porovnanie rôznych váh

Pomocou modelu zo sekcie 4.11 a predom klasifikovaných dát som vypočítal váhy (FSV), ktoré budú v tejto sekcii porovnané s ručne definovanými váhami (DV) na celej testovacej sade. Váhy boli vypočítane pri použití desiatich intervalov. Počiatočná pravdepodobnosť, že sú dva záznamy zhodné vyšla 0,0027 čo znamená, že približne 1 z 500 porovnaných párov osôb reálne hovorí o zhode.

	TP	FP	TN	FN	Prec.	Rec.	F1	Bal. acc	Čas
DV	15 736	1 991	906 266	5 270	0,888	0,749	0,813	0,873	1 165,56 s
FSV	14 113	1 211	929 326	15 429	0,921	0,478	0,629	0,738	1 372,51 s

Tabuľka 5.4: Porovnanie využitia vypočítaných váh

V tabuľke 5.4 je jasne vidieť, že vypočítané váhy produkujú podstatne horšie výsledky. Najpodstatnejší problém je nárast **FN** na trojnásobok pôvodnej hodnoty. O takmer tretinu klesli **FP** čo by samo o sebe bolo v poriadku. Celkovo znížený počet pozitívnych klasifikácií a nárast tých negatívnych indikuje, že hranica zhody 0,9 je pre nové váhy príliš vysoká. Preto som sa rozhodol nové váhy otestovať aj na nižších prahoch.

t_u	t_l	TP	FP	TN	FN	Prec.	Rec.	F1	Bal. acc
0,9	0,7	14 113	1 211	929 326	15 429	0,921	0,478	0,629	0,738
0,8	0.65	15 812	1 416	706 536	9 691	0,918	0,620	0,740	0,809
0,7	0,6	15 991	1 509	665 878	9 367	0,914	0,631	0,746	0,814
0,6	0,5	16 118	1 615	643 163	8 917	0,909	0,644	0,754	0,821
0,5	0,4	16 273	1 711	595 910	8 390	0,905	0,660	0,763	0,828

Tabuľka 5.5: Porovnanie vypočítaných váh pri rôznych prahoch

Prirodzene, čím nižší prah pre zhodu nastavíme, tým viac záznamov bude klasifikovaných ako zhoda. Najväčší skok môžeme vidieť pri zmene z 0,9 na 0,8. Pri prahu 0,8 sa podarilo dosiahnuť vyšší počet **TP** než pri ručne definovaných váhach s o štvrtinu nižším počtom **FP**, ale hodnota **FN**, ktorá je stále takmer dvojnásobná podstatne zhoršuje štatistiky. Ďalšie znižovanie váh už nemá veľký význam vzhľadom na rýchlo narastajúce **FP** a zanedbateľné zmeny v **FN**.

Tieto výsledky ale môžu byť skreslené nenormalizovanými hodnotami a faktom, že o niektorých osobách máme veľmi málo informácií. Z počiatočnej pravdepodobnosti sa vypočítalo „log-odds“ -8,53. Táto hodnota je využitá ako počiatočné skóre každej porovnávanej dvojice. Vypočítané váhy pre úplnú zhodu mena a priezviska sú 5,57 a 5,97 a prudko klesajú. Pri podobnosti z intervalu $\langle 0,8, 0,9\rangle$ sú už iba 3,63 a 4,76. To znamená, že ak máme o nejakej osobe iba tieto dve informácie, musia byť takmer totožné, aby sa skóre dostalo nad 0 (pravdepodobnosť zhody 0,5). Nenormalizované hodnoty takúto podobnosť často nedosahujú.

Ak máme iba meno alebo iba priezvisko osoby, taktiež sa môže stať, že sa tieto osoby stanú nezlučiteľnými, keďže s jedným atribútom nikdy nedosiahnu potrebné skóre, a tým pádom nikdy nezískajú ďalšie atribúty pomocou zlúčenia.

Napriek horším výsledkom si však myslím, že použitie týchto váh netreba zavrhnúť. Ideálne by bolo ich otestovať na normalizovaných dátach, ktoré zatiaľ nie sú k dispozícii, aby sa overila ich funkcionalita.

5.6 Nové štatistiky

Na záver sa pozrieme na nové štatistiky definované v sekcii 5.1. Testované budú zvlášť na pôvodnej sade, iba na sčítacích operátoch a na celej sade. Otestujú sa aj obe nastavenia váh.

	Pôvodná sada				
Váhy	Id/Uzly	**P** (id/uzly)	**S** (id/uzly)	**M** (id/uzly)	**SM** (id/uzly)
DV	2 886 / 5 159	1 137 / 1 137	972 / 2 920	241 / 114	536 / 988
FSV	2 886 / 5 982	1 251 / 1 251	1 050 / 3 747	227 / 92	358 / 892

Tabuľka 5.6: Nové štatistiky na pôvodnej sade

Na pôvodnej sade 5.6 môžeme vidieť, že sa vo výslednej databáze nachádza približne dvojnásobok uzlov než bolo v pôvodnej databáze unikátnych osôb. Najväčšiu časť týchto uzlov tvoria nezlúčené uzly z kategórie **S**. Táto hodnota by mohla pri využití normalizovaných hodnôt klesnúť, je ale otázne, či by to nespôsobilo aj nárast zlúčených uzlov s rôznymi osobami v kategóriach **M** a **SM**.

	Sčítacie operáty				
Váhy	Id/Uzly	**P** (id/uzly)	**S** (id/uzly)	**M** (id/uzly)	**SM** (id/uzly)
DV	1 019 / 1 051	462 / 462	178 / 382	289 / 137	90 / 70
FSV	1 019 / 590	220 / 220	51 / 106	525 / 154	223 / 110
FSV (SO)	1 019 / 1 070	467 / 467	175 / 382	259 / 114	118 / 107

Tabuľka 5.7: Nové štatistiky na sčítacích operátoch

Pri sčítacích operátoch 5.7 výsledky podstatne záležia na využitých váhach. Ručne definované váhy produkujú takmer rovnaké množstvo osôb v pôvodnej aj výslednej databáze. Takmer polovica z nich sú dokonca ideálne osoby, ktoré sa v nej nachádzajú iba raz a nie sú s nikým nesprávne zlúčené. Takto dobré výsledky sú pravdepodobne spôsobené veľkým počtom informácii, ktoré sčítacie operáty poskytujú o jednotlivých osobách. Využitie vypočítaných váh a pravdepodobnostnej klasifikácie ponúka podstatne horšie výsledky. Vidíme, že množstvo osôb, ktoré nemali byť zlúčené sa vo výslednej databáze zlúčili do jednej, čo spôsobilo veľký nárast v kategórii **M**. Predpokladám, že je to spôsobené tým, že váhy boli počítané na celej sade, ktorá obsahuje podstatne menej informácii. Preto som ich otestoval aj na váhach vypočítaných čisto z dát sčítacích operátov, a ako je vidieť na poslednom riadku tabuľky 5.7, výsledky sú takmer totožné s ručne definovanými váhami.

	Celá sada				
Váhy	Id/Uzly	**P** (id/uzly)	**S** (id/uzly)	**M** (id/uzly)	**SM** (id/uzly)
DV	3 058 / 5 764	954 / 954	1 141 / 3 497	342 / 140	621 / 1 173
FSV	3 058 / 6 245	988 / 988	1 096 / 3 859	430 / 164	544 / 1 234

Tabuľka 5.8: Nové štatistiky na celej sade

Testy na celej sade poskytujú podobné výsledky ako pôvodná sada s matrikami. Približne dvojnásobok uzlov vo finálnej databáze oproti pôvodnej. Rozdiel medzi ručne definovanými váhami a tými vypočítanými však nie je až tak veľký ako pri pôvodnej sade, čo môže byť opäť spôsobené tým, že váhy boli počítané práve na celej sade.

Kapitola 6

Záver

Hlavným cieľom práce bolo rozšíriť pôvodnú prácu o nové typy záznamov a opraviť niektoré nedostatky pôvodného systému. Tomu predchádzalo naštudovanie oboru genealógie a rôznych historických prameňov, ktoré vieme pri prepájaní osôb využiť. Taktiež bolo potrebné sa oboznámiť s fungovaním klasifikačných systémov a podrobne pochopiť návrh a implementáciu predchádzajúcich prác.

Tento cieľ bol splnený a podpora pre všetky typy záznamov bola implementovaná (sčítacie operáty iba z csv súborov, keďže ešte neexistovali tabuľky v relačnej databáze). Na samotnom programe prebehlo niekoľko zmien a rozšírení, ktoré zlepšili efektivitu niektorých častí programu, odstránili chyby pri porovnávaní a zjednotili celkovú štruktúru kódu. Pridané boli aj ďalšie štatistiky, ktoré popisujú finálnu podobu grafovej databázy. Vo výsledku je program viac než štyri krát rýchlejší a robí podstatne menej nesprávnych klasifikácií než pôvodná verzia.

V práci na projekte bude potrebné ďalej pokračovať, minimálne doplnením chýbajúcej podpory pre sčítacie operáty z relačnej databázy a rozšírením existujúcich vzťahov osôb o všetky možné normalizované hodnoty.

Literatúra

[1] *Badatelna*. Moravský zemský archív v Brne. Dostupné z: http://www.mza.cz/a8web/a8apps1/a8apps1.htm.

[2] *Knihy pozemkové, pojišťující soukromá práva měšťanů Kniha kupů a prodejů domů*. Státní okresní archiv Nový Jičín. Dostupné z: https://digi.archives.cz/da/permalink?xid=CFFA50E4886A11E1A634D0DF9A2C4EFF.

[3] *Matriky*. Moravský zemský archív v Brne, 2020. Dostupné z: https://www.mza.cz/actapublica/matrika/hledani?typ=obec&obec_id=172.

[4] Baženov, V. *Za bohatstvím našich předků*. 2019. Dostupné z: https://www.hledanipredku.cz/gruntovni-pozemkove-knihy/.

[5] DuVall, S. L., Kerber, R. A. a Thomas, A. Extending the Fellegi–Sunter probabilistic record linkage method for approximate field comparators. *Journal of Biomedical Informatics*. 2010, zv. 43, č. 1, s. 24–30. DOI: https://doi.org/10.1016/j.jbi.2009.08.004. ISSN 1532-0464. Dostupné z: https://www.sciencedirect.com/science/article/pii/S1532046409001051.

[6] Fellegi, I. P. a Sunter, A. B. A theory for record linkage. *Journal of the American Statistical Association*. 1969, zv. 64, č. 328, s. 1183–1210.

[7] Hříbek, D. *Poloautomatická normalizace slov z matričních záznamů*. Brno, CZ, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/thesis/21640/.

[8] Lednická, B. *Rodopisné stránky*. Dostupné z: http://rodokmen.nase-koreny.cz/matriky/obsah_matrik.htm.

[9] Linacre, R. Understanding match weights. 2021. Dostupné z: https://www.robinlinacre.com/understanding_match_weights/.

[10] Marhefka, A. *Generování rodokmenů z matričních záznamů*. Brno, CZ, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/thesis/24660/.

[11] Pine, L. G. *Genealogy*. Encyclopedia Britannica, 5. september 2021. [Online; accessed 18.12.2022]. Dostupné z: https://www.britannica.com/topic/genealogy.

[12] Radimský, J. Rektifikační akta. Moravský zemský archiv v Brně. 1958. Dostupné z: http://www.mza.cz/a8web/a8apps1/D2/D2-Inventar.pdf.

[13] Radimský, J. a Štaud, D. Lánové rejstříky. Moravský zemský archiv v Brně. 2018. Dostupné z: http://www.mza.cz/a8web/a8apps1/d1/D1-Inventar.pdf.

[14] Tušimová, L. *Generování rodokmenů z matričních záznamů*. Brno, CZ, 2020. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/thesis/22818/.

[15] Wikipedia contributors. *Logistic function — Wikipedia, The Free Encyclopedia*. 2023. [Online; accessed 5-May-2023]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Logistic_function&oldid=1148617753.

[16] Wikipedia contributors. *Logit — Wikipedia, The Free Encyclopedia*. 2023. [Online; accessed 5-May-2023]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Logit&oldid=1149432870.

[17] Wikipedie. *Genealogie — Wikipedie: Otevřená encyklopedie*. 2022. [Online; navštíveno 18. 12. 2022]. Dostupné z: https://cs.wikipedia.org/w/index.php?title=Genealogie&oldid=22716487.