VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

DATABÁZE PRO GENEALOGICKÉ MODELY

DATABASE FOR GENEALOGICAL MODELS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **MICHAL VANKA**
AUTHOR

VEDOUCÍ PRÁCE **Ing. RADEK KOČÍ, Ph.D.**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2020/2021

Zadání bakalářské práce

Student: **Vanka Michal**
Program: Informační technologie
Název: **Databáze pro genealogické modely**
Database for Genealogical Models
Kategorie: Databáze

Zadání:

1. Seznamte se s problematikou práce s genealogickými daty, tvorbou genealogických modelů a existující databází přepisovaných matričních záznamů ČR (DEMoS), která vzniká v rámci projektu na FIT.
2. Proveďte analýzu požadavků na strukturu a ukládání dat pro potřeby genealogických modelů a na základě analýzy vyberte vhodný typ databáze.
3. Navrhněte a realizujte databázi reflektující potřeby ukládání genealogických modelů. Musí být umožněno mapování osob z modelů do výchozí databáze přepsaných záznamů DEMoS.
4. Navrhněte a realizujte uživatelské rozhraní pro správu modelů. Rozhraní integrujte do webového rohraní systému DEMoS.
5. Na datech z databáze DEMoS otestujte systém a diskutujte jeho další vývoj.

Literatura:

- Moravský Zemský Archiv. Digitalizace archivních fondů. http://www.mza.cz/a8web/a8apps1/a8apps1.htm.

Pro udělení zápočtu za první semestr je požadováno:

- První tři body zadání.

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

Vedoucí práce: **Kočí Radek, Ing., Ph.D.**
Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 11. listopadu 2020

Abstrakt

Táto práca, ktorá rozširuje projekt DEMoS, pojednáva o riešení problému efektívneho ukladania genealogických modelov do databázy. Pre tento účel sú porovnané rôzne druhy a konkrétne riešenia databázových systémov za účelom nájdenia toho najvhodnejšieho. Po zvolení vhodného riešenia je v systéme vytvorená štruktúra na ukladanie genealogických záznamov. Riešenie kladie dôraz na zníženie redundancie dát oproti pôvodnému systému na ukladanie osôb projektu DEMoS. Práca poskytuje tento systém zabalený do webového užívateľského rozhrania pre praktickú možnosť začlenenia do projektu DEMoS, ako aj pohodlnú prácu s databázovým systémom.

Abstract

This work, which extends the DEMoS project, deals with solving efficient storage of genealogical models in the database. For this purpose, different types and specific solutions of database systems are compared to find the most suitable one. After selecting the right solution, a structure for storing genealogical records is created, emphasizing the reduction of data redundancy compared to the original system for storing people in the DEMoS project. The work provides this system with a web user interface for the practical possibility of integration into the DEMoS project and convenient work with the database system.

Kľúčové slová

genealogické modely, rodostrom, grafová databáza, dokumentová databáza, multi-modelová databáza, ArangoDB, webové užívateľské rozhranie

Keywords

genealogical models, family tree, graph database, document database, multi-model database, ArangoDB, web user interface

Citácia

VANKA, Michal. *Databáze pro genealogické modely.* Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Radek Kočí, Ph.D.

Databáze pro genealogické modely

Prehlásenie

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

.....
Michal Vanka
7. mája 2021

Poďakovanie

Chcem poďakovať môjmu vedúcemu práce Ing. Radkovi Kočímu, Ph.D. za vecný a nekonečne pokojný prístup, ako aj za odborné rady a snahu pomôcť, keď bolo treba. Ďalej by som rád poďakoval svojim rodičom za vytvorenie skvelého prostredia na písanie tejto práce doma aj v čase globálnej pandémie. V neposlednom rade ďakujem za trpezlivosť svojim priateľom, ktorí museli počúvať moje sťažnosti a predsa pri mne stáli.

Obsah

Kapitola 1

Úvod

Ten, kto sa rozhodol do problematiky genealógie ponoriť hlbšie, určite narazil na nedostatok technologicky pokročilých nástrojov, ktoré by mu v bádaní históriou a organizáciou zistených poznatkov pomohli. S týmto problémom sa nanešťastie stretávajú aj odborníci, ktorí sa venujú prepisovaniu genealogických informácií z archívnych prameňov do digitálnej podoby. Schopnosť efektívne zorganizovať, prehľadávať a filtrovať takto digitalizované záznamy je kľúčová, ak má mať digitalizácia vôbec zmysel.

Práve tomuto problému sa venuje projekt DEMoS. Ide o projekt, ktorý je riešení na Fakulte Informačných Technológii VUT a zaoberá sa digitalizáciou starých matričných záznamov. Postupne sa rozširuje, a súčasťou tohoto rozširovania je aj táto práca.

Cieľom práce je preskúmať rôzne typy databáz za účelom nájdenia toho najvhodnejšieho pre ukladanie genealogických modelov, ktoré budú vychádzať z existujúcich záznamov digitalizovaných v rámci projektu. Následná implementácia má byť schopná tieto dáta prehľadne ukladať a spájať. V neposlednom rade budú musieť byť dáta zobraziteľné, a to tak, aby užívateľ nemusel čítať komplikovaný manuál k databázovému rozhraniu.

Práca sa delí na viacero častí. Kapitola 2 objasňuje termíny a oblasti, od ktorých sa práca odráža, zároveň analyzuje požiadavky na nový systém. Kapitola 3 sa venuje výberu vhodného typu databázy, a ďalej výberu najlepšej technológie implementujúcej tento typ databázy. Následne práca pokračuje implementáciou samotnej štruktúry databázy v kapitole 4 a v kapitole 5 čitateľ nájde integráciu databázy do webového rozhrania projektu DEMoS, pričom tu práca diskutuje aj o ďalšom vývoji.

Kapitola 2

Súčasný stav a analýza požiadaviek

V tejto kapitole sú zhrnuté potrebné poznatky k pochopeniu problematiky genealógie (avšak iba v nutnom merítku) a uvedenie projektu DEMoS. Následne budú zanalyzované požiadavky na systém.

2.1 Genealógia

Genealógia je slovo, ktoré mnohým ľuďom nič nepovie. Avšak vo svojej podstate všetci, ktorí sa aktívnejšie zaujímajú o históriu svojej rodiny sú svojím spôsobom amatérski genealógovia. Táto veda je stará ako ľudstvo samo. Skúma vzťahy jedincov, vyplývajúce z ich spoločného rodového pôvodu.

Pre laika možno bude známejší pojem rodokmeň alebo rodostrom. Tieto pojmy vyplývajú práve z genealógie, a sú to spôsoby usporiadania genealogických informácií do takzvaných **genealogických modelov**.

Formálna definícia genealogického modelu môže byť množina $Mg = Ro,E,RE,ID$, kde Ro je súbor rolí, E je súbor udalostí, RE a ID sú relácie nad Ro. Obe tieto relácie sú reláciami ekvivalencie. RE reprezentuje reláciu medzi všetkými rolami, ktoré sú súčasťou akejkoľvek udalosti. ID je potom relácie, ktorá zoskupuje všetky role, ktoré odkazujú na tú istú osobu.[7]

Genealogické modely majú rôznu podobu, napr. genogram[1], ale tým bezprostredne najznámejším je **rodokmeň**. Menej známe však je, že toto pomenovanie označuje model, v ktorom iba k potomkom mužských členov rodiny. Pokiaľ ide o všetkých potomkov všetkých členov rodiny, ide o **rozrod**. O definícii genealogického modelu pre túto prácu sa pojednáva v sekcii 2.4.1.

[1] https://genopro.com/genogram/

2.2 Existujúce riešenia

Na vytváranie genealogických modelov a ukladanie genealogických dát o osobách existuje niekoľko veľmi prepracovaných riešení. Väčšina týchto riešení je komerčných a dostupných pre použitie širokou verejnosťou za poplatok. Táto sekcia je však iba informatívna, a to z dôvodu, že zadanie tejto práce má na databázové riešenie špecifické požiadavky(2.4.2), ktoré nespĺňa ani jedno z uvedených riešení.

2.2.1 Ancestry

Najväčšia a najznámejšia genealogická stránka na internete[2], založená v roku 1996. Podľa slov jej predstaviteľov majú vo svojej databáze viac než 10 miliárd záznamov, a stránka má viac ako 3 milióny predplatiteľov. Okrem genealogických dát a vytvárania genealogických modelov poskytuje spoločnosť vlastniaca stránku možnosť objednať si genealogický DNA test, ktorý môže užívateľ použiť, zaslať späť, a spoločnosť pomocou DNA zistí historické a etnické korene užívateľa. Ďalej spoločnosť vlastní viacero ďalších služieb, ako napr. *Find a grave*(vyhľadávač hrobov v USA) alebo *Newspapers*(archív skenov historických novín).

2.2.2 FamilySearch

Jediné z "veľkej štvorky"genealogických riešení, ktoré nie je komerčné[3]. Spoločnosť je vlastnená Cirkvou Ježiša Krista Svätých neskorších dní, známou aj ako Mormonská Cirkev, a stránka je úzko prepojená s Oddelením Rodinnej Histórie tejto cirkvi. Po registrácii má užívateľ prístup ku genealogickým záznamom vyše 1,3 miliardy osôb, pričom databáza obsahuje aj okolo 5,7 miliardy obrázkov digitalizovaných mikrofilmov, kníh, a iných záznamov.

2.2.3 Findmypast

Služba vznikla v roku 2003[4], avšak korene výskumu skupiny genealógov, ktorá za ňou stojí, siahajú až do roku 1965. Napodiv nejde o Americkú stránku, ale stránku pochádzajúcu zo Spojeného Kráľovstva, čomu odpovedajú aj jej záznamy, ktoré obsahujú prevažne osoby zo Spojeného Kráľovstva a Írska. Obsahuje viac ako 9 miliárd záznamov, do čoho sa však rátajú aj neindexované dáta, čiže číslo reálne dohľadateľných záznamov je zrejme nižšie.

2.2.4 MyHeritage

Izraelská genealogická služba[5], ktorá obsahuje okolo 7,3 miliardy indexovaných mien a celkovo viac než 13 miliárd historických záznamov. Je špecifická svojou dostupnosťou,

[2] https://www.ancestry.com/
[3] https://www.familysearch.org/
[4] https://www.findmypast.co.uk/
[5] https://www.myheritage.com/

čo potvrdzuje preklad až do 42 svetových jazykov. Založená bola v roku 2003, a má viac než 50 miliónov užívateľov. Jej záznamy sú najviac koncentrované v Európe.

2.3 DEMoS

Genealogická databáza DEMoS (Database of Early Modern Sources) vzniká v rámci projektu TAČR ETA TL01000130 a podieľa sa na nej Fakulta informačních technologií VUT v Brně v spolupráci s Ústavem pomocných věd historických a archivnictví Filozofické fakulty Masarykovy univerzity v Brně. Cieľom projektu je vytvoriť databázu pre prepis informácií z archívnych prameňov, ako sú matriky, sčítacie operáty, Lánové registre, pozemkové knihy a ďalšie.

V rámci tohoto projektu už existuje MySQL databáza, ktorá uchováva prepísané genealogické záznamy. Práve na tie by mali odkazovať záznamy uložené v databáze implementovanej v tejto práci.

2.4 Špecifikácia požiadaviek

Požiadavky na systém môžeme rozdeliť na genealogické, teda na štruktúru databázy, a na integračné, teda začlenenie nového systému do širšieho projektu.

2.4.1 Genealogické požiadavky

Rodokmeň (často nazývaný aj rozrod, ktorý však označuje iný model) je termín, s ktorým sa bude v tejto práci narábať mnohokrát. Ide o základnú genealogickú tabuľku, ktorá väčšinou sleduje iba mužskú líniu predkov určitej osoby, pričom o ženách sú uvedené iba základné údaje.[6]

Pre potreby projektu DEMoS je však v tejto práci táto definícia rozšírená na sledovanie línií otca aj matky určitej osoby, a zároveň aj ďalších rodinných aj mimorodinných vzťahov. V databáze sa teda nemajú nachádzať iba vzťahy medzi rodičmi a deťmi, prípadne manželom a manželkou, ale majú sa tu nachádzať napr. aj pôrodné baby, krstní rodičia, alebo kňazi, ktorí dieťa krstili. Práve toto umožní širšie mapovanie vzťahov a teda aj sofistikovanejšie dotazy na databázu. Pomocou dotazu by napríklad malo byť možné jednoducho zistiť, kto je strýkom danej osoby.

Každý genealogický záznam môže mať mnoho rôznych atribútov, ich počet sa často výrazne líši pri každej osobe (napr. sa stáva, že je pri prepise dát nečitateľný údaj o dátume narodenia, tento atribút sa teda vynechá), preto je pre ukladanie týchto dát vhodné zvoliť takú technológiu, ktorá nebude vyžadovať uniformnú štruktúru pre všetky záznamy. Takáto štruktúra totižto spôsobuje zbytočnú neefektivitu, či už vyhradením miesta na disku pre daný atribút v každom zázname alebo spomaľovaním dotazov kvôli nutnosti prehľadávania takto málo zaplnených tabuliek.
Zhrnutie:

- ukladanie osôb s rôznym počtom atribútov, a to čo najefektívnejšie

- ukladanie rodinných vzťahov medzi osobami
- ukladanie mimorodinných vzťahov

2.4.2 Integračné požiadavky

Ako už bolo spomenuté, pri voľbe a následnej implementácii vhodného riešenia je nutné myslieť na to, že má byť:

- bezplatné, teda nevyužívať platené riešenia, ktoré by boli pre zameranie projektu zbytočné
- prepojené s existujúcou databázou projektu DEMoS, využívať existujúce záznamy
- upraviteľné tak, aby vedelo prijať akékoľvek rozšírenie modelov
- integrované do webového rozhrania projektu DEMoS

Previazanie záznamov s existujúcou databázou by malo jednoznačne spojiť osobu z databázy modelov so záznamom danej osoby v existujúcej databáze záznamov osôb. Databáza bude však umožňovať vkladať aj záznamy bez takéhoto odkazu. Vo webovom rozhraní by tak malo byť možné bez ďalších dodatočných dotazov zobraziť viac detailov o jedincovi. Nebude teda nutné ukladať všetky dáta dvakrát, v prípade takéhoto previazania by bolo nutné uložiť iba kľúčové atribúty.

Ak by v budúcnosti malo dôjsť k rozšíreniu genealogických modelov, a to či už záznamov o osobách, alebo vzťahov medzi nimi, je nutné, aby bolo túto zmenu v riešení jednoducho a efektívne uskutočniť.

Čo sa týka integrácie práce do webového rozhrania, je potrebné myslieť na komunikáciu medzi rozhraním a databázou. Toto by malo byť vyriešené najlepšie pomocou databázového systému, ktorý je schopný komunikovať cez API (Application Programming Interface - rozhranie pre programovanie aplikácií). Pomocou neho by potom bolo jednoduché s databázou pracovať, pričom implementácia nebude vyžadovať žiadne zložité programovanie.

Kapitola 3

Výber databázy

Zo špecifikácie požiadaviek vyplynulo, že pre daný problém nebude klasická relačná databáza správnym riešením, keďže má byť jednoducho rozšíriteľná a uchovávať rôzny počet atribútov. Napokon na technológii MySQL je postavené existujúce databázové riešenie projektu DEMoS. Pri pohľade na existujúce tabuľky je však hneď jasné, že databáza nie je využitá efektívne.

birth_id	marriage_id	burial_id	rel	title	nameFull	snameFull	domicileFull	street	descr_num
1	*NULL*	*NULL*	main	*NULL*	Maria	*NULL*	Adamsthal		10
1	*NULL*	*NULL*	f		Franz	Schlesinger	in Adamsthal		*NULL*
1	*NULL*	*NULL*	m		Anna	*NULL*	*NULL*		*NULL*
1	*NULL*	*NULL*	midwife	*NULL*	Josefa	Blažek	von Babitz		*NULL*

Obr. 3.1: Príklad tabuľky z projektu DEMoS ilustrujúci malé využitie jednotlivých stĺpcov

Je teda jednoznačne na mieste siahnuť po inom riešení, konkrétne NoSQL riešení. Koncept NoSQL nemá presnú definíciu. Istý vedecký článok ho definuje zhruba nasledovne: "Využíva jednoduché a flexibilné dátové modely. Úložiská NoSQL ponúkajú flexibilné schémy alebo schému vôbec neobsahujú, pričom sú navrhnuté tak, aby zvládli rôznorodé dátové štruktúry. Dátové modely aktuálnych riešení sa dajú rozdeliť do štyroch kategórii: úložiská kľúč-hodnota, dokumentové úložiská, úložiská stĺpcových rodín a grafové databázy." [1]

Na odporúčanie garanta boli do porovnania zaradené tieto typy NoSQL databáz:

- objektová databáza
- dokumentová databáza
- grafová databáza

Pre doplnenie bude porovnaná aj relačná databáza.

3.1 Typ databázy

Relačná databáza Jeden z najstarších a najrozšírenejších typov databáz. Tento typ je založený na relačnom modeli dát, čo je model pracujúci s matematickými princípmi *predikátovej logiky* a *relačných množín*. Dáta sú teda usporiadané do n-tíc (v systéme záznamy/riadky) a združené v reláciách (v systéme tabuľky obsahujúce riadky). Každá takáto tabuľka má riadky a stĺpce, ako je známe z bežného života. Systém riadiaci takto štrukturovanú databázu sa nazýva RDBMS - Relational Database Management System (Systém riadenai relačnej databázy). K práci s databázou sa využíva jazyk SQL (Structured Query Language - štruktúrovaný dotazovací jazyk), ktorý je v dnešnej dobe de facto štandardom pre prácu s väčšinou databázových systémov. Pomocou tohoto jazyka sa do systému zadávajú transakcie, ktoré následne manipulujú s dátami. Takéto transakcie musia zároveň spĺňať 4 vlastnosti ACID - Atomicity (atomickosť), Consistency (konzistentnosť), Isolation (izolovanosť) a Durability (trvalosť) aby bol systém efektívny a dáta v ňom validné aj v situáciách kedy môže nastať chyba v databáze, napr. výpadok prúdu. SQL práve toto zaručuje.

Vyššie opísaná štruktúrovanosť je však zároveň slabinou relačných databáz. Pokiaľ dáta ukladané do databázy majú prevažne nejednotnú štruktúru, je nutné pre rozličné typy atribútov tvoriť ďalšie stĺpce, čo nevyhnutne vytvára prázdne alokované miesta na disku. Práve to môžme vidieť na obrázku 3.1.

Príklady systémov: *MySQL*, *PostgreSQL*, *Oracle*

Objektová databáza V objektovej databáze sú informácie reprezentované formou objektov namiesto záznamov a tabuliek. Tento typ systému sa zdal byť zaujímavý pre riešenie zadaného problému, no po krátkom prieskume sa zistilo, že žiadny z ponúkaných konkrétnych systémov nie je bezplatný alebo bol jeho vývoj ukončený. Z porovnania boli teda objektové databázy vylúčené.
Príklady systémov: *Db4o(vývoj ukončený)*, *ObjectDB*

Dokumentová databáza Tento typ databázy uchováva jednotlivé záznamy v dokumentoch, v ktorých sa atribúty uchovávajú vo formáte kľúč-hodnota. Toto umožňuje pridávať ľubovolný počet atribútov do ktoréhokoľvek záznamu bez toho, aby sa alokovalo akékoľvek zbytočné miesto. Keď je takáto databáza zároveň in-memory (dáta sú uložené v pamäti s priamym prístupom, teda RAM, nie klasicky na disku), dokáže dotazy vykonávať niekoľkonásobne efektívnejšie než relačná databáza. Dokumentová databáza by teda určite bola dobrým riešením pre túto prácu.
Príklady systémov: *MongoDB*, *ArangoDB*, *SAP HANA*

Grafová databáza Dá sa povedať, že tento typ databázy je veľmi podobný dokumentovej databáze, no namiesto ukladania informácií do dokumentov ich ukladá do tzv. uzlov (nodes), pričom tieto uzly prepája orientovanými hranami (edges). Takýmto pre-

pojením vzniká grafová štruktúra, ktorou sa dá veľmi jednoducho a rýchlo ”cestovať” (graph traversal). Taktiež bola vyhodnotená ako schopný kandidát pre prácu.
Príklady systémov: *Neo4j*, *ArangoDB*

3.2 Výber konkrétneho systému

Po rozsiahlejšom vyhľadávaní boli vybrané dve konkrétne riešenia pre túto prácu, a to **Neo4j** a **ArangoDB**.

3.2.1 Neo4j

Ide o grafovú databázu, ktorá je vysoko škálovateľná. Dáta sa ukladajú čisto vo forme *uzlov*, ktoré sú spojené *hranami*. Ide o open-source databázu, no toto riešenie je čiastočne spoplatnené (voľná verzia má rôzne obmedzenia), platená verzia je však veľmi efektívna a rýchla. Na trhu sa vyskytuje od roku 2007 a využíva ju mnoho veľkých hráčov na trhu.

Jej čiastočnou nevýhodou je, že v jednej inštancii môže bežať iba jeden graf, čo môže obmedzovať rôznorodosť grafov. Práve vzhľadom na obmedzenosť tohoto systému bolo vykonané iba obmedzené testovanie. V prípade záujmu nájdete viac na stránke tohoto systému.[1]

3.2.2 ArangoDB

Tento databázový systém je zaujímavý svojím netradičným typom, ktorý je kombináciou dokumentovej a grafovej databázy. *Uzly* v grafe sú namiesto samostatných entít odkazmi na *dokumenty* (ktoré sú uložené v tej istej inštancii databázy), pričom *hrany* grafu môžu niesť dodatočné informácie v rovnakom formáte ako dokumenty. Hrany aj dokumenty sú potom združené v *kolekciách*, pričom každá kolekcia môže byť buď hranová, alebo dokumentová. Týmto je dosiahnuté prehľadné a efektívne rozdelenie samotných záznamov a vzťahov medzi nimi. Oba typy kolekcií sa následne môžu združiť no *grafu*.

Ide taktiež o open-source riešenie, avšak na rozdiel od Neo4j je táto databáza úplne bezplatná (vrátane Enterprise edície pokiaľ ide o nekomerčné skúšobné použitie).

Systém využíva dotazovací jazyk vyvinutý jeho autormi nazývaný **AQL**(ArangoDB Query Language), ktorý je v mnohom podobný klasickému SQL, no umožňuje úkony špecifické pre tento databázový model (ako napr. získavanie dokumentov spojených hranou). Veľkou výhodou je jeho veľmi dobrá dokumentácia a jednoduchosť pochopenia pre ľudí, ktorí už majú s klasickým SQL skúsenosti.

Okrem iného sú v ArangoDB možnosti ako vytváranie pohľadov(Views), manažment užívateľov, vytváranie viacerých databáz v rámci jednej inštancie, a ďalšie.

Nesporným plusom je aj užívateľské rozhranie, ktoré je dodávané priamo s databázou. Rozhranie má svieži moderný dizajn, prehľadné členenie, a zároveň veľmi efektívne nakladá so zdrojmi. Vďaka nemu sa dá systém manažovať de facto bez znalosti jeho shell príkazov. Ďalšie informácie na stránke tvorcov. [2]

[1] https://neo4j.com/product/neo4j-graph-database/
[2] https://www.arangodb.com/

3.2.3 Výsledok

Po zhodnotení všetkých pre a proti jednoznačne vysvitlo, že vhodnejším riešením bude práve **ArangoDB**. Multi-modelový typ databázy je pre účely riešenia zadaného problému lepší práve kvôli oddeleniu informácií o osobe a jej vzťahu k iným osobám. Zároveň bude vďaka tomuto typu databázy možné prideľovať osobám ľubovoľný počet atribútov. Dôležitým aspektom je aj to, že ArangoDB je kompletne zdarma, pričom Neo4j je pri voľnej verzii výkonnostne obmedzené.

Treba poznamenať, že obe riešenia majú vstavané REST API, ktoré bude možné využiť pri začlenení do webového rozhrania DEMoS ako aj pri prepojení existujúcej a vznikajúcej databázy, čo boli jedny z hlavných požiadaviek. Zároveň sú obe riešenia vysoko prepracované a ľahko implementovateľné, argumentov pre výber jedného riešenia oproti druhému preto nie je mnoho.

3.3 Pojmy ArangoDB

V zvolenom systéme je nutné poznať niekoľko základných pojmov. Ako už bolo spomenuté, ide o multi-modelový systém, ktorý kombinuje dokumentový a grafový typ databázy, kde sa niektoré vlastnosti prekrývajú. Táto sekcia by mala všetky pojmy objasniť. Väčšina definícií pojmov je prevzatá z manuálu ArangoDB.[2]

3.3.1 Dokument

V tomto systéme sa dokumentom označuje JSON objekt (obsahujúci páry kľúč-hodnota), ktorý môže byť vnorený (do ľubovoľnej hĺbky) a môže obsahovať zoznamy. Každý dokument má unikátny *primárny kľúč*, ktorý ho identifikuje v rámci príslušnej kolekcie. Zároveň je unikátne identifikovaný pomocou *document handle* (madlo alebo rukoväť, ďalej iba handle) naprieč všetkými kolekciami v rámci jednej databázy. Rôzne revízie jedného dokumentu (identifikovaného pomocou jeho handle) môžu byť odlíšené pomocou *dokumentovej revízie*. Okrem týchto atribútov môže dokument obsahovať nula a viac iných atribútov. Každý z týchto atribútov nadobúda hodnotu, ktorá môže byť atomického typu, teda napr. číslo, reťazec, boolean alebo hodnota null, alebo zložený typ, teda napr. pole alebo iný vnorený objekt.
V rámci dokumentu sú povinné tieto atribúty:

- _id - handle, generované systémom, napr. "_id" : "myusers/3456789"
- _key - primárny kľúč, môže byť zadaný užívateľom, inak generované systémom, napr. "_key" : "3456789"
- _rev - revízia, generované systémom pri vytvorení alebo zmene dokumentu, napr. "_rev" : "14253647"

3.3.2 Hrana

V multi-modelovom systéme s ktorým sa pracuje je hrana špeciálny druh dokumentu. Okrem atribútov _id, _key a _rev má ešte ďalšie dva povinné atribúty označujúce smer hrany (ktorá je orientovaná). Tieto atribúty obsahujú *document handle* patriacu dokumentu alebo hrane v rámci rovnakej databázy.
Tie sú označené nasledovne:

- _from - začiatočný bod hrany, napr. "_from" : "docCollection/5823"
- _to - konečný bod hrany, napr. "_to" : "edgeCollection/9774"

Hrana je, zjednodušene povedané, spojením dvoch dokumentov, akousi cestou medzi nimi, ktorá umožňuje medzi týmito dokumentami "cestovať" pomocou dotazu, prípadne ich spojenie ukázať v grafe. Je to veľmi užitočný nástroj na modelovanie vzťahov medzi entitami, a zároveň je kľúčovým prvkom pre rýchle vyhľadávanie informácií v grafovej databáze.

3.3.3 Kolekcia

Dokumenty aj hrany sú združené v kolekciách (ekvivalent tabuľky v relačnej databáze, v tomto prípade by dokumenty alebo hrany boli riadky tabuľky). Tá môže obsahovať nula a viac takýchto položiek, no vždy iba jedného typu, kolekcia je teda buď **dokumentová**(document), alebo **hranová**(edge). Kolekcie sa následne združujú v databáze, pričom databáz môže byť v jednom systéme niekoľko, no sú od seba izolované. Dokument v kolekcii databázy 1 teda nemôže odkazovať na dokument v databázy 2.

3.3.4 Graf

Graf sa skladá z *vrcholov* a *hrán*. Hrany sú uložené ako špeciálne dokumenty v hranovej kolekcii. Vrchol môže byť dokumentom dokumentovej alebo hranovej kolekcie (čiže hrany môžu byť použité ako vrcholy inej hrany). Ktoré kolekcie sú použité v rámci pomenovaného grafu je definované cez *definície hrán*. Pomenovaný graf môže obsahovať viac než jednu definíciu hrán, avšak nutná je aspoň jedna.

3.3.5 Príklad

Aby bolo jednoduchšie si predstaviť, ako databáza funguje, je namieste uviesť príklad.

Majme dva filmy, Film1 a Film2. V týchto filmoch hrajú dvaja herci, Herec1 a Herec2. Pre filmy si vytvorme *kolekciu* Filmy. Pre hercov zase *kolekciu* Herci.
Kolekcia **Filmy** obsahuje:

- Film1 = { _id : "Filmy/125", _key = "125", _rev : "_cP2mxdG—", description : "cool movie" }
- Film2 = { _id : "Filmy/128", _key = "128", _rev : "_cP2mxdG–O", name : "Matrix" }

Kolekcia **Herci** obsahuje:

- Herec1 = { _id : ”Herci/228”, _key = ”228”, _rev : ”_cP2mxdG-_G”, name : ”John”, surname : ”Smith” }
- Herec2 = { _id : ”Herci/234”, _key = ”234”, _rev : ”_cP2mxdG-_Y”, name : ”Lara”, surname : ”Croft” }

Títo herci hrajú v jednom alebo druhom filme, a bolo by vhodné ich k daným filmom priradiť. Tu sa využije hranová kolekcia Účinkujúci, ktorá bude využívať atribúty _id na prepojenie dvoch záznamov.

Hranová kolekcia **Účinkujúci** teda bude obsahovať:

- { _id : ”Účinkujúci/986”, _key = ”986”, _rev : ”_cPz0hPG—”, _from : ”Filmy/125” _to : ”Herci/228” }
- { _id : ”Účinkujúci/988”, _key = ”988”, _rev : ”_cPz0hPG–A”, _from : ”Filmy/125” _to : ”Herci/234’ }
- { _id : ”Účinkujúci/999”, _key = ”999”, _rev : ”_cPz0hPG–D”, _from : ”Filmy/128” _to : ”Herci/234’ }

Tu vidno, že jeden záznam môže mať prepojenie s viacerými inými záznamami v rámci rovnakej kolekcie. Vo Filme1 teda hrali Herec1 aj Herec2, pričom Herec1 zároveň hral vo Filme1 ako aj Filme2.

Pre vytvorenie grafu (príklad grafu na obrázku 4.2) sú zadané parametre:

- Name: Filmový graf
- Edge definitions: **Účinkujúci**
- fromCollections: **Filmy**
- toCollections: **Herci**

Tým sa vytvorí zobrazenie všetkých prepojení medzi kolekciou filmov a kolekciou hercov, pričom vzťahy _from a _to môžu byť aj obrátené, teda by mohla hrana od herca smerovať k filmu a graf ich stále zobrazí.

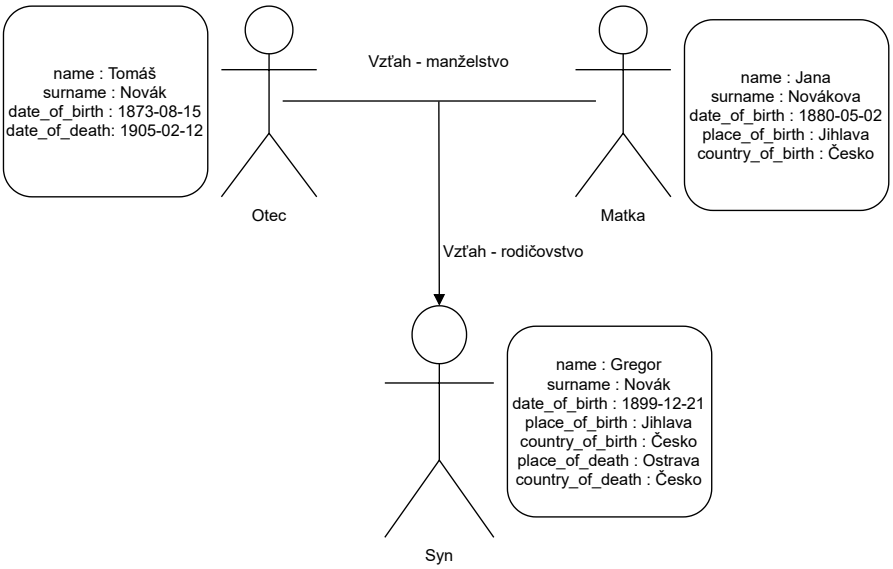
Kapitola 4

Návrh a implementácia štruktúry databázy

Vďaka voľbe multi-modelového NoSQL databázového systému ArangoDB je možné pri návrhu postupovať oveľa flexibilnejšie než pri relačnej databáze. Nie je treba vymýšľať komplikovanú schému pre každú tabuľku, ktorá by sa snažila pokryť všetky prípady užitia. Namiesto toho bude nutné iba definovať entity figurujúce v databáze, a zároveň definovať, ktoré vzťahy medzi nimi bude databáza uchovávať v samostatných kolekciách. Zároveň je však jednou nevýhodou to, že nie je dosť dobre možné oddeliť návrh a implementáciu databázy, pretože v prípade špecifickej technológie ako je ArangoDB je nutné vopred počítať s konkrétnymi vlastnosťami a možnosťami danej technológie. Práve z tohoto dôvodu sú kapitoly spojené a v jednotlivých sekciách sú myšlienky návrhu priamo prepojené s implementáciou.

4.1 Návrh štruktúry

Vo svojej podstate je rodostrom iba štruktúra obsahujúca dve entity, a to **osoby** a **vzťahy** medzi nimi. Keďže má táto databáza obsahovať rôzne detaily o osobách, líšiac sa od záznamu k záznamu, je logické uzavrieť všetky osoby do jedného logického celku, ktorý bude navonok rovnaký, no s rozdielnou štruktúrou vnútri. Vzťah je takisto iba vzťahom, či už ide o vzťah rodiča a dieťaťa, vzťah manželstva, alebo vzťah súrodenecký. Taktiež môžme uzavrieť všetky tieto možnosti do jedného celku. Takéto logické rozdelenie znázorňuje obrázok 4.1.



Obr. 4.1: Náčrt rozdielnych atribútov a vzťahov v databáze

4.2 Implementácia

Pre túto prácu bola vybraná Enterprise edícia databázového systému ArangoDB, ktorá je voľne dostupná pre posudzovanie a skúšanie, čo je aj prípadom tejto práce. Neboli však nakoniec použité žiadne funkcie, ktoré by neobsahovala Community edícia, čo znamená, že v prípade reálneho nasadenia v projekte DEMoS bude možné využiť Community edíciu, ktorá má Apache 2.0 licenciu.

Na získanie Enterprise edície bolo nutné registrovať sa, po čom odkaz na inštaláciu prišiel e-mailom. Pri inštalácii Windows verzie databázy bolo nainštalované klasické shell rozhranie, ale zároveň aj veľmi pekné a pokročilé webové rozhranie(*Arango Management Interface*).

Celá implementácia systému je vytvorená pomocou webového rozhrania. Shell rozhranie bolo využité jedine pri exporte databázy pre účely zálohy, čo však nie je v rámci tejto práce podstatné.

4.2.1 Vytvorenie kolekcie

Pri vytvorení kolekcie vo webovom rozhraní je potrebné:

1. Zadať meno vytváranej kolekcie
2. Zvoliť, či bude táto kolekcia *hranová* alebo *dokumentová*

Okrem týchto možností je ešte pri vytváraní kolekcie zvoliť pokročilú možno *Wait for sync*, ktorú však toto riešenie nevyužíva.

4.2.2 Osoby

Základnou entitou v databáze genealogických modelov je jednoznačne osoba. V databáze je teda vytvorená dokumentová kolekcia nazvaná **Persons**, ktorá bude obsahovať výhradne dokumenty s údajmi o osobách.

Schéma

Napriek tomu, že v ArangoDB nie sú nutné schémy, existuje tu možnosť využiť validáciu každého pridaného dokumentu do kolekcie JSON schémou. [1] ArangoDB využíva mierne upravený formát tejto schémy, kde pridáva definíciu vlastnej chybovej hlášky a úrovne vynucovania schémy. Vytvorením takejto validačnej schémy bude zabezpečené, že do systému nebude možné pridať dokument bez povinných atribútov, ktoré sú v nej uvedené (a samozrejme atribútov _id, _key a _rev). Pre tento prípad boli ako povinné atribúty zvolené *meno, priezvisko, dátum narodenia a pohlavie.*

```
{
  "message": "name, surname and date_of_birth are required attributes",
  "level": "strict",
  "rule": {
    "properties": {
      "name": {
        "type": "string"
      },
      "surname": {
        "type": "string"
      },
      "date_of_birth": {
        "type": "string",
        "format": "date"
      },
      "gender": {
        "type": "string",
        "enum": [
          "male",
          "female"
        ]
      }
    },
    "required": [
      "name",
      "surname",
      "date_of_birth",
      "gender"
    ]
  }
}
```

Výpis 4.1: Definícia pravidiel schémy

Vysvetlivky k definícii pravidiel schémy vo výpise 4.1:

[1] https://json-schema.org/

- ”message” - chybová hláška, ktorú systém vráti v prípade nedodržania schémy pri vkladaní/úprave dokumentu
- ”level” - definuje, kedy je validácia použitá, môže nadobúdať hodnoty[3]:
  - none - pravidlá sú neaktívne a validácia je vypnutá
  - new - validované sú iba novo pridané dokumenty
  - moderate - validované sú nové a upravované dokumenty, avšak s výnimkou modifikovaných dokumentov, kde staré hodnoty (OLD - špecifická hodnota v ArangoDB) neboli tvorené s validáciou, čo v praxi znamená že staré nevalidné dokumenty sa dajú modifikovať na opäť nevalidné dokumenty, no validné dokumenty sa nedajú modifikovať na nevalidné dokumenty
  - strict - validované sú všetky nové a upravované dokumenty bez výnimky
- ”rule” - obsahuje už samotné pravidlá JSON schémy (drží sa JSON schema syntaxi)
  - ”properties” - sem patria jednotlivé atribúty ktoré by mali byť kontrolované, a to vo formáte ”atribute” : restrictions
  - ”required” - zoznam atribútov, ktoré záznam pri validácii musí obsahovať, použité môžu však byť iba atribúty, ktoré sú definované v ”properties”

Pre pridanie iných povinných atribútov stačí pridať položku do pola *”properties”*, kde všetky položky musia byť oddelené čiarkou (ako je vidno na príklade vyššie) a definovať tam jej typ či prípadné ďalšie obmedzenia. Tým sa zaistí, že ak sa bude nachádzať v dokumente daný atribút, bude sa musieť držať definovaného formátu (napr. teda bude musieť byť typu string a bude môcť obsahovať iba hodnoty definované v poli enum, čo je pole vymenovaných hodnôt). Následne treba pridať daný atribút (tentokrát už iba meno) do pola *required*, čím sa stane atribút povinným, a nebude možné pridávať ďalšie dokumenty bez toho, aby tento atribút obsahovali. Pre úplnosť treba povedať, že nie všetky atribúty, pre ktoré sú definované pravidlá, musia byť povinné. Ak je pre atribút definované pravidlo v poli *”properties”*, nemusí sa nachádzať v poli *required*, no naopak v poli *required* sa môžu nachádzať iba atribúty pre ktoré existuje pravidlo v *”properties”*.

Osoby samozrejme môžu mať ľubovoľný počet iných vlastností, ktoré budú všetky definované užívateľom. Ak bude teda po prepise z matriky o osobe známy dátum narodenia, avšak nie miesto narodenia, nie je nutné pre tento de facto neexistujúci údaj vyhradzovať miesto alebo ho akokoľvek značiť. Pri dotaze v ktorom sa bude užívateľ špecificky dotazovať na tento atribút databázový systém nevráti chybu, ale dokument o danej osobe jednoducho preskočí.

4.2.3 Vzťahy

Všetky rodinné aj prípadné mimo-rodinné prepojenia medzi jednotlivými osobami sú budú obsiahnuté v kolekcii **Relationships**, ktorá však bude hranová (Edge collection). Bude teda obsahovať hrany tvoriace graf, ktorý spája existujúce dokumenty, ale nebude obsahovať dokumenty samotné.

Samozrejme nestačí vytvárať iba jednoduché orientované hrany medzi dokumentami, keďže treba uchovávať aj typ vzťahu medzi osobami. Práve tu príde vhod to, že každá hrana je špeciálnym dokumentom, ktorý dokáže uchovávať atribúty rovnako ako normálne dokumenty. Do každej hrany bude pridaný atribút *relationship*, ktorý bude označovať o aký druh vzťahu sa jedná. Vzťahy, pri ktorých je potrebné určite orientáciu, teda napr. kto je rodič komu, alebo kto je dieťa koho, bude využitá orientovanosť hrán. Pri dotazoch na deti osoby sa následne zadá, že je vyhľadávaný rodičovský vzťah, a že má systém vyhľadávať iba odchádzajúce hrany. Pri niektorých vzťahoch nie je potrebné určiť orientáciu hrany (napr. vzťah medzi súrodencami, kedy je jedno, kto je súrodencom komu, vzťah je rovnocenný) sa budú využívať oba smery, a hranu je možné označiť ako neorientovanú, aj keď toto systém nedovoľuje.
Atribút môže nadobúdať nasledujúce hodnoty:

- "parent of" - orientovaná hrana, vzťah od rodiča k dieťaťu
- "siblings" - obojsmerná hrana, vzťah medzi súrodencami
- "married" - obojsmerná hrana, vzťah medzi manželmi
- "godparent of" - orientovaná hrana, vzťah od krstného rodiča k dieťaťu
- "witness to" - orientovaná hrana, vzťah od svedka svadby k manželom
- "baptised" - orientovaná hrana, vzťah od krstiteľa ku krstenej osobe
- "delivered" - orientovaná hrana, vzťah od pôrodnej baby ku narodenej osobe(dieťaťu)
- "wedded" - orientovaná hrana, vzťah od oddávajúceho k oddávaným

Parameter *relationship* je povinný pre každú hranu, a môže nadobúdať jedine tieto hodnoty, čo je definované v JSON schéme kolekcie **Relationships** rovnakým spôsobom ako pri kolekcii **Persons** (4.2.2).

V prípade potreby je možné tieto hodnoty rozšíriť aj na ďalšie vzťahy, systém ani existujúce údaje to nijak neovplyvní. Stačí nový vzťah pridať do atribútu *"enum"* v schéme.

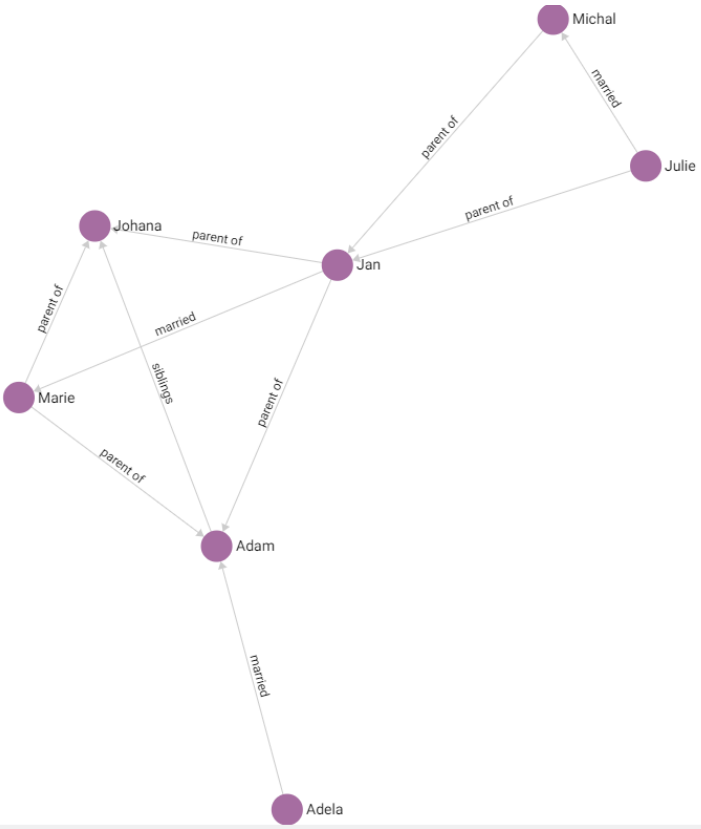
4.2.4 Dáta

Každá databáza vzniká za jediným účelom - ukladať a triediť dáta. ArangoDB má niekoľko spôsobov na vkladanie dát do databázy:

- Importovanie záznamov zo súboru typu CSV/TSV[2] - iba cez shell
- Importovanie záznamov zo súboru typu JSON[3] - iba cez shell
- Vytvorenie záznamov priamo v databázovom systéme - shell aj webové rozhranie

[2] https://www.arangodb.com/docs/stable/programs-arangoimport-examples-csv.html
[3] https://www.arangodb.com/docs/stable/programs-arangoimport-examples-json.html

Pre kontrolu správnej funkčnosti databázového riešenia tejto práce bolo potrebných aspoň niekoľko vzorových záznamov osôb a vzťahov medzi nimi. Bolo teda vytvorených 9 fiktívnych osôb a 10 hrán(vzťahov) medzi nimi (vzniknutý graf je na obrázku 4.2). Na ich vytvorenie sa využilo vytváranie osôb a hrán vo webovom rozhraní. Import dát ostatnými metódami je veľmi dobre popísaný v odkazoch uvedených pod čiarou, a importovanie pomocou *csv* súboru bolo využité pri **Teste systému**(5.5) neskôr v tejto práci.



Obr. 4.2: Vzniknutý graf osôb a vzťahov medzi nimi, zobrazený vo webovom rozhraní ArangoDB

4.2.5 Užívatelia

Pri spúšťaní databázy je nutné vytvoriť tzv. root(kmeňového) užívateľa. Tento užívateľ má následne právo vytvárať ďalších užívateľov, spravovať ich, a prideľovať im práva. Prístup k tomuto užívateľskému účtu by mal mať samozrejme iba hlavný správca databázového systému.

Okrem prístupu do systému a nastaveniu všeobecných práv je možné užívateľom špeciálne práva pre každú kolekciu v systéme zvlášť. Môže teda napríklad existovať užívateľ, ktorý môže iba vytvárať vzťahy medzi osobami, no nie osoby samotné, alebo užívateľ, ktorý môže si môže osoby aj vzťahy medzi nimi iba prezerať.

V tomto systéme bol pre všetky ďalšie činnosti vytvorený užívateľ "test", ktorému boli udelené administrátorské práva pre uľahčenie práce s databázou. Tento užívateľ bol následne použitý pri práci s databázou pomocou API z webového rozhrania vytvoreného vrámci ďalšej časti tejto práce, kde rozhranie na autentifikáciu používa užívateľov databázy ArangoDB namiesto vytvárania ďalšej databázy užívateľov iba pre toto webové rozhranie. Viac v sekcii 5.4.1.

4.3 Zhrnutie

Databáza pre genealogické modely bola implementovaná pomocou NoSQL databázového systému ArangoDB. Tento systém sa v porovnaní s inými NoSQL systémami, ako aj tradičnými relačnými databázami ukázal byť najlepšou voľbou pre tento účel. Databázu sa podarilo implementovať v súlade s požiadavkami na systém, a riešenie je možné začleniť bez väčších problémov alebo prekážok do projektu DEMoS. Na plnohodnotné začlenenie je však potrebné užívateľské rozhranie, ktoré zaobalí prácu s databázou tak, aby užívatelia nemuseli riešiť veci, ktoré s ich genealogickou výskumnou prácou nijako nesúvisia. Tvorbe tohoto rozhrania sa bude práca venovať v ďalšej kapitole.

Kapitola 5

Integrácia databázy do webového rozhrania DEMoS

Hlavným podnetom pre vytvorenie databázy o ktorej pojednáva celá táto práca bolo využitie v projekte DEMoS, predstavenom v kapitole 2. Výsledné produkty tohoto projektu sú združené na stránke projektu.[1] Práve do tejto stránky má byť databáza začlenená, pričom je nutné vytvoriť prepojenie s databázou. Ako bolo spomenuté v kapitole 3, databázový systém obsahuje vstavané rozhranie (API) pre komunikáciu s externými aplikáciami, ktoré sa dá presne pre tento účel využiť.

5.1 ArangoDB API

Preložené z manuálových stránok.[4] ArangoDB sprístupňuje svoje API cez HTTP protokol, čím robí server jednoducho prístupný pomocou rôznych klientov a nástrojov (napr. prehliadačov). Komunikácia medzi serverom a klientom môže byť v prípade potreby šifrovaná. Navyše umožňuje systém využiť vlastný binárny protokol *VelocyStream*, ktorý sa dá použiť pre lepší prietok dát. HTTP požiadavky sa dajú na tento protokol jednoducho namapovať, a neexistuje k nemu ani samostatná dokumentácia, keďže API funguje v podstate rovnako pre oba tieto protokoly.

ArangoDB používa štandardné HTTP metódy (napr. GET, POST, PUT, DELETE), ako aj metódu PATCH popísanú v RFC 5789[2].

Server očakáva, že klienti budú odosielať všetok dátový obsah vo formáte JSON alebo vlastný VelocyPack binárny formát od ArangoDB. Klienti odosielajúci požiadavky na server musia použiť jeden z protokolov HTTP 1.0, HTTP 1.1, HTTP 2 alebo VelocyStream. Iné verzie HTTP protokolu či iné protokoly nie sú podporované.

V tomto riešení je použitý klasický HTTP protokol z dôvodu prehľadnosti riešenia a vyhnutia sa prípadným zbytočným komplikáciám, ktoré by mohli vzniknúť pri použití druhého protokolu.

[1] http://www.genealogickadatabaze.cz/
[2] https://tools.ietf.org/html/rfc5789

5.2 Návrh rozhrania

Vstupným bodom bude určite prihlasovacia obrazovka, ktorá zaistí, že k databáze bude pristupovať skutočne užívateľ tejto databázy. Základom tohoto systému by však mali byť 2 hlavné možnosti. Prvou by mal byť formulár na vyhľadanie určitej osoby, ktorá bude slúžiť ako vstupný bod pre vykreslenie rodostromu. Tou druhou je vloženie novej osoby do databázy.

1. Vyhľadanie osoby - Osobu je možné vyhľadať pomocou mena, priezviska, dátumu narodenia a pohlavia. Užívateľ by si mal byť následne schopný vybrať zo zoznamu existujúcich osôb v databáze. Do jednotlivých polí je vhodné začleniť nejakú nápovedu, či automatické doplnenie, ktoré by možnosti čerpalo z existujúcich údajov v databáze. Z tejto obrazovky sa bude prechádzať na väčšinu ďalších obrazoviek

2. Vloženie novej osoby - Pri vkladaní je dôležité dodržať aplikovanú JSON schému (priblížené v sekcii 4.2.2), ale samozrejme zabaliť celé riešenie do užívateľsky prívetivého užívateľského rozhrania, aby užívateľ nemusel kontrolu schémy riešiť sám. Všetky dotazy na databázu je teda nutné skontrolovať v rozhraní

Detail osoby

Po vyhľadaní a zvolení správnej osoby (na základe kľúčových, garantovane nenulových atribútov) by mal byť užívateľ schopný zobraziť si detaily danej osoby, teda všetky (neinterné) atribúty, ktoré o nej uchováva databáza. Vzhľadom na to, že osoby v databáze už dávno nežijú a ide o databázu na výskumné účely, nie je nutné ktorékoľvek z (neinterných) atribútov skrývať z dôvodu súkromia či ochrany údajov. Internosť atribútov je spomenutá z dôvodu, že užívateľ nepotrebuje poznať interné atribúty ako *_id* alebo *_rev*, ktoré sú pre prácu s databázou nepodstatné.

Na tej istej obrazovke by sa mali nachádzať tlačítka na manipuláciu so záznamom danej osoby v databáze. Nemali by chýbať tlačítka *Edit*(upraviť), *Delete*(odstrániť) a tlačítko na zobrazenie rodostromu osoby.

V neposlednom rade sa tu bude nachádzať práve tlačítko, ktoré zaistí prepojenie s pôvodnou databázou osôb projektu DEMoS, a buď zobrazí informácie z tejto databázy, alebo presmeruje užívateľa priamo na záznam osoby v inom module DEMoSu. Rozhranie bude informácie získavať priamo SQL dotazom na databázu.

Úprava osoby

Pri úprave údajov osoby by malo byť možné upravovať iba neinterné atribúty. Takisto ako nie je dôvod pre úpravu atribútu *_key*, ktorý sa radí medzi neinterné, keďže ho môže užívateľ zmeniť, no v tomto systéme je vždy generovaný automaticky, čiže je logické, aby bol aj spravovaný automaticky. Musí tu byť zároveň opäť kontrola vyplnenia povinných atribútov, keďže by bolo neefektívne zakaždým čakať na odpoveď databázového systému.

Zobrazenie rodostromu

Pravdepodobne najkomplexnejšia časť celého webového rozhrania. Pokiaľ to bude možné, určite by tu bolo vhodnejšie použiť už existujúce vizualizačné riešenie, než sa snažiť vytvoriť nové od nuly. Určite musí zobrazenie obsahovať vizualizáciu jednotlivých osôb spojených s východiskovou osobou. Ďalej treba zobraziť tieto vzťahové prepojenia medzi týmito osobami, ako aj jasne odlíšiť, o aký typ prepojenia ide. Keďže je v databáze rozlíšené, či je osoba niekoho potomkom alebo predkom, zobrazenie by malo byť schopné vyfiltrovať na želanie užívateľa predchádzajúce si následujúce generácie osôb v rodostrome. Samozrejme sa tu môžu nachádzať aj ďalšie filtre, napríklad zobrazovanie iba určitej skupiny vzťahov, či len jedného typu vzťahu. Práve na tejto obrazovke sa bude nachádzať pridanie osôb do rodostromu východiskovej osoby, a to práve tak, že si novú osobu užívateľ vyhľadá, pridá, a zobrazenie sa ihneď aktualizuje. Taktiež bude možné osobu z rodostromu východiskovej osoby odstrániť.

Nemala by chýbať možnosť rozkliknúť si detail osoby kliknutím na ňu, kde sa budú nachádzať s niekoľkými možnosťami interakcie s ňou. V prípade, že by nešlo o východiskovú osobu, bude možné osobu určiť ako východiskovú, pričom sa prekreslí graf tak, aby bol centrovaný na ňu. Z tejto obrazovky však určite nemá byť možné osobu z databáze odstrániť, alebo ju upraviť, to bude možné iba po prechode na obrazovku detailu osoby.

5.3 Použité technológie a vzory

Keďže stránka DEMoSu je postavená na jazyku PHP[3], je jasné, že v ňom bude napísané aj rozhranie pre tento systém. Pred začatím implementácia užívateľského rozhrania bola dilema, či začať vývoj s nejakým z existujúcich PHP frameworkov (aplikačných rámcov, ďalej iba framework) pre zjednodušenie vývoja. Keďže však autor nemal skúsenosti so žiadnym frameworkom, po niekoľkých hodinách skúšania rôznych z nich bolo rozhodnuté, že sa pre začiatok pôjde cestou čistého PHP, HTML a CSS.

Časom bolo webové rozhranie doplnené pre interaktívnosť o JavaScript a hlavne o knižnicu jQuery, vďaka ktorej sa vývoj násobne urýchlil. Taktiež boli použité niektoré ďalšie knižnice opísané nižšie. Všetky tieto knižnice sú vložené ako HTML script, ktorého zdrojom je CDN(Content delivery network - sieť na doručovanie obsahu), čím je zaistené, že potrebné knižnice sú k užívateľovi stiahnuté vždy keď ich je treba.

Využitý bol upravený návrhový vzor Model-View-Controller (ďalej iba MVC)[4], ktorý zabezpečuje oddelenie logiky od zobrazenia.
Rozdelenie od priečinkov je teda nasledovné

- public - obsahuje súbory na ktoré užívateľ pristupuje, spájajú Models a Templates
- Models - obsahuje logiku aplikácie, teda všetky triedy, ktoré nič nevykresľujú
- Templates - súbory vykresľujúce HTML stránku

[3] https://www.php.net/
[4] https://en.wikipedia.org/wiki/Model-view-controller

- index.php - hlavný súbor mimo priečinkov, slúžiaci ako vstupný bod rozhrania

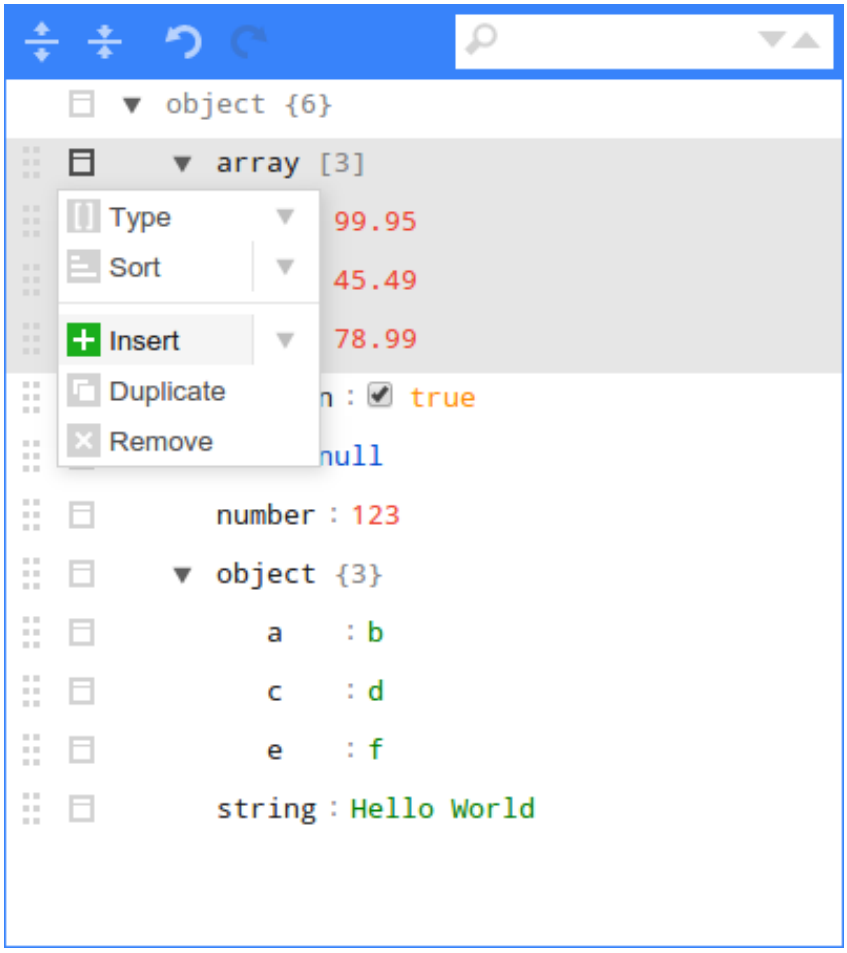
5.3.1 JSON Editor[5]

Po prvotných pokusoch vytvorenia dostatočne funkčného formulára na pridanie a upravovanie osoby bolo jasné, že by takéto riešenia bolo absolútne kontraproduktívne. Padlo teda rozhodnutie, že bude využitá existujúca knižnica na úpravu celých JSON dokumentov, keďže práve toto je formát, v ktorom sú uložené osoby v databáze. Po preskúmaní rôznych možností bola tou najprívetivejšou práve *JSON Editor*, ktorý zapadol do rozhrania práve svojou kompaktnosťou. Z užívateľského hľadiska by mala byť priama úprava taktiež prijateľná, keďže editor je prehľadný, minimalistický, a dokáže v reálnom čase vyhodnocovať dodržanie či porušenie JSON schémy (ukážka rozhrania na obrázku 5.1). Celá knižnica je navyše distribuovaná pod licenciu Apache 2.0.

Do editoru je možné nahrať JSON dokument a zobraziť ho na upravovanie, prípadne v ňom vytvárať úplne nový JSON súbor. Užívateľ však vôbec nemusí vedieť nič o súboroch JSON, stačí, keď to bude brať ako obyčajný editor. Zároveň je veľkou výhodou editoru možnosť pridať JSON schému, ktorá bola použitá pre kontrolu hodnôt v databázovom systéme. Zaistí sa tým teda kontrola záznamu ešte predtým, než je odoslaný do databázy, čím sa eliminuje potreba kontroly správnosti dokumentu v databázovom systéme, a následnej kladnej či zápornej odpovede klientovi.

Okrem iného má editor niektoré užitočné menej výrazné funkcie, ako presúvanie celého riadku s atribútom a hodnotou medzi iné riadky jednoduchým potiahnutím, či pridávanie riadkov pred alebo za ktorýkoľvek riadok samostatným tlačítkom.

[5] https://github.com/josdejong/jsoneditor



Obr. 5.1: Ukážka vzhľadu editora, kde je vidno jeho rôznorodú ponuku dodatočných funkcií (prevzaté z https://github.com/josdejong/jsoneditor)

5.3.2 Sigma js[6]

Pre vykreslenie grafu z databázy bola vybratá práve táto knižnica jazyka Javascript, a to z viacerých dôvodov. Pred výberom tejto knižnice boli preskúmané ešte ďalšie tri riešenia:

1. **dTree**[7] - Knižnica pre vizualizáciu dátových stromov s viacerými rodičovskými vetvami, postavená na knižnici D3[8]. Pôvodne šlo o hlavného kandidáta na využitie v tomto webovom rozhraní, no po ďalšom prieskume bolo zistené, že pomocou tejto knižnice nie je možné spájať 2 vetvy rodín, teda napríklad ak majú rodičia A syna

[6] http://sigmajs.org/
[7] https://github.com/ErikGartner/dTree
[8] https://github.com/d3/d3

A_s, a rodičia B dcéru B_d, nie je možné napriek rovnakej úrovni týchto osôb spojiť syna A_s a dcéru B_d. Je teda jasné, že knižnica je týmto pádom nevyhovujúca.

2. **jHTree**[9] - Tento jQuery plugin(doplnok) pomáha dynamicky vykresľovať rodostrom/organizačnú štruktúru, ktoré sú animované a je možné v nich poťahovať, približovať a rozklikávať položky. Doplnok čerpá svoje dáta zo súboru typu JSON. Opäť však bolo zistené, že tu nie je možné spájať dve osoby z rôznych rodín, rovnako ako je popísané pri **dTree**.

3. **vis.js**[10] - Dynamická prehliadačová vizualizačná knižnica. Knižnica je navrhnutá tak, aby bolo jednoduché ju použiť, aby bola schopná zvládnuť veľké množstvo dynamických dát, a aby aby umožnila manipuláciu a interakciu s týmito dátami. Zo spomenutých knižníc ide o tú najvyspelejšiu a najznámejšiu. S knižnicou **Sigma js** sú na rovnakej úrovni. Sú aktívne vyvíjané, je s nimi oboma jednoduchá práca a majú takmer rovnaké možnosti. V prospech Sigma js však zavážilo to, že na pracuje ako základná knižnica, na ktorú sa podľa potreby dajú nasadzovať doplnky, pričom vis.js je rozdelená na rôzne menšie knižnice, a ak je potreba použiť iný typ grafu, ale funkciu, ktorá nie je v danom balíčku, je nutné začleniť celú ďalšiu knižnicu.

Z porovnania s ostatnými kandidátmi vyšla knižnica Sigma js ako najlepšia voľba. Zároveň je to knižnica, ktorú používa samotné webové rozhranie ArangoDB, takže je jasné, že zvládne prácu s dátami z tohoto systému. Nemenej dôležitým faktom je, že je vytvorená pod licenciou MIT, ktorá umožňuje voľné použitie softwaru v plnom rozsahu, a to aj za komerčným účelom.

Je zameraná na vykresľovanie grafov vo webových aplikáciách a zároveň je jednoduchá na integráciu a používanie. Pracuje, rovnako ako grafové databázy, s uzlami a hranami, ktoré sú uložené vo formáte ukázanom vo výpise 5.1.

[9] https://www.jqueryscript.net/chart-graph/Family-Tree-Organization-Chart-jHTree.html
[10] https://visjs.org/

```
//uzol
{
    // Main attributes:
    id: 'n0',
    label: 'Hello',
    // Display attributes:
    x: 0,
    y: 0,
    size: 1,
    color: '#f00'
}
//hrana
{
    id: 'e0',
    // Reference extremities:
    source: 'n0',
    target: 'n1'
}
```

Výpis 5.1: Príklad definície uzlov a hrán v Sigma js

Pre uzly aj hrany existujú rôzne možnosti nastavenia, napr. je možné nastaviť farbu, nápis, veľkosť nápisu, apod. Niektoré možnosti sú však dostupné iba s pomocou pluginov(doplnkov).

sigma.renderers.edgeLabels[11]

Pomocou tohoto pluginu je možné pridávať nápisy k hranám v grafe. Toto bolo potrebné z dôvodu jasného vypísania vzťahov medzi osobami(uzlami) v grafe. V rámci popisu tvorby webového rozhrania budú vymenované všetky využité nastavenia.

sigma.layout.forceAtlas2[12]

Tento plugin bol využitý kvôli celkovej estetike grafu. Zároveň bolo cieľom urobiť zobrazenie podobné zobrazeniu grafu vo webovom rozhraní ArangoDB, ktorý taktiež využíva Force-layout. Pre úplnosť je dobré spomenúť aj definíciu algoritmu ForceAtlas2.

ForceAtlas2 je silou riadené rozloženie: simuluje fyzikálny systém za účelom umiestnenia siete v priestore. Uzly sa navzájom odpudzujú ako nabité častice, pričom hrany sa navzájom priťahujú ako pružiny. Tieto sily vytvárajú pohyb, ktorý konverguje k vyrovnanému stavu. Táto finálne rozmiestnenie má pomôcť lepšie interpretovať dáta.[5]

[11] https://github.com/jacomyal/sigma.js/tree/master/plugins/sigma.renderers.edgeLabels
[12] https://github.com/jacomyal/sigma.js/tree/master/plugins/sigma.layout.forceAtlas2

5.3.3 ArangoDB-PHP[13]

Pre celú komunikáciu medzi serverom a databázou bol použitý ArangoDB-PHP klient, ktorý je doplnkovou knižnicou pre jazyk PHP. Klient bol vytvorený samotnými tvorcami ArangoDB, teda nejde o žiadnu technológiu tretej strany.

Pre prehľadnosť kódu boli všetky metódy sprostredkujúce túto komunikáciu uzavreté do triedy *DbConnector.php*, ktorej úlohou je vždy sprostredkovať dotazy tak, aby bolo vrátené pole, s ktorým sa dá pracovať bežným spôsobom v PHP (bez znalosti štruktúr ArangoDB). Táto trieda je vytvorená podľa návrhového vzoru *Singleton*, ktorý zaisťuje, aby vždy existovala iba jedna inštancia danej triedy, a to z dôvodu, aby pre jedno sedenie užívateľa bolo vždy vytvorené iba jedno spojenie s databázou, a predišlo sa tak zbytočnej viacnásobnej autentifikácii. Metódy v triede (s výnimkou konštrukčnej metódy *GetInstance*) sa delia na "query" a priame metódy. "Query" metódy komunikujú s databázou pomocou dotazov v jazyku AQL, a využívajú sa pre získanie neurčitého počtu výsledkov z databázy. Naopak priame metódy využívajú metódy a štruktúru triedy *DocumentHandler* z Arango-PHP, ktorá manipuluje vždy s jedným dokumentom. Takéto členenie zaisťuje väčšiu prehľadnosť kódu a väčšinou aj ľahšiu prácu s výsledkami dotazov.

5.4 Implementácia webového rozhrania

Pre rýchlosť vývoja a efektivitu rozhrania neboli využité takmer žiadne komplikované grafické prvky. Z toho dôvodu môže implementácia vyzerať stroho, no je plne funkčná a plní všetky požiadavky, ktoré boli na systém kladené.

5.4.1 Prihlásenie

Vstupnou obrazovkou do systému je prihlásenie. Ide o jednoduchý formulár, ktorý obsahuje vstup pre meno a heslo užívateľa. Tieto údaje sa musia zhodovať s prístupovými údajmi do databázy genealogických modelov ArangoDB. Účet teda musí byť vytvorený administrátorom databázy, alebo iným oprávneným užívateľom, nový užívateľ sa nemôže zaregistrovať sám. Po zadaní údajov a kliknutím na tlačítko prihlásiť server pomocou metódy *DbConnector/checkLogin* kontaktuje databázový server s jednorázovou požiadavkou a to iba na overenie údajov. Pokiaľ sú údaje platné, prihlasovacie meno a heslo sa zašifrujú šifrou AES s náhodne vygenerovaným kľúčom a uložia sa do premennej *$_SESSION* na serveri. Nejde o najbezpečnejšie narábanie s údajmi, no implementácia prihlasovania do ArangoDB cez API v PHP vyžaduje prihlasovacie údaje vo forme čistého textu, teda nie je možné pre tento účel vytvárať jednorázový token. Ďalej je kľúč aj s ďalšími informáciami uložený ako cookie k užívateľovi. Počas celej práce s rozhraním sa následne toto cookie používa na rozšifrovanie potrebných údajov pre kontakt s databázou. V prípade odhlásenia sa z ktorejkoľvek obrazovky je cookie aj údaje v premennej *$_SESSION* ihneď zmažú.

Po prihlásení je užívateľ presmerovaný na obrazovku **Vyhľadávanie osôb**(5.4.2).

[13] https://www.arangodb.com/docs/stable/drivers/php.html

5.4.2 Vyhľadávanie osôb

Ako bolo spomenuté v návrhu, ide o jednu z dvoch hlavných častí rozhrania. Hlavných preto, lebo práve k týmto dvom častiam je možné dostať sa z ktorejkoľvek obrazovky bez toho, aby bola vyžadovaná akákoľvek POST alebo GET hodnota. Zároveň ide o obrazovku, na ktorú je užívateľ presmerovaný po prihlásení.
Pre vyhľadávanie sú použité tieto hodnoty:

- Name - meno osoby, nezáleží na veľkých a malých písmenách, meno na vyhľadanie nemusí byť kompletné (teda pri zadaní písmen "ja" nájde všetky Jany aj Jánov
- Surname - priezvisko osoby, nezáleží na veľkých a malých písmenách, priezvisko na vyhľadanie nemusí byť kompletné (rovnaký princíp ako pri mene)
- Date of birth - musí byť vo formáte YYYY-MM-DD, teda napr. 1820-02-11 označuje 11. február 1820

Ani jedna z týchto položiek však nie je povinná, a pokiaľ užívateľ bez zadania ktorékoľvek z údajov klikne na vyhľadávacie tlačítko, zobrazí sa zoznam všetkých osôb v databáze.

Vo vyhľadávacom formulári bolo vypnuté prednastavené automatické doplňovanie, ktoré bolo nahradené automatickým doplňovaním pomocou jQuery(metóda *.autocomplete*). Účelom bolo poskytnúť užívateľovi doplňovanie priamo z databázy, a to ako pri poli *name*, ako aj pri poli *surname*, aby užívateľ nemusel premýšľať, či dané meno či priezvisko vôbec v databáze je. Na túto komunikáciu je použitá AJAX komunikácia, ktorá pri každom zadanom znaku odošle obsah pola na server, kde sa využije metóda *DbConnector/querySuggestion*, ktorá dotazuje databázu na akékoľvek mená či priezviská začínajúce týmito písmenami.

Vo výberovom boxe sa po vyhľadaní zobrazia všetky osoby zodpovedajúce vyhľadávacím kritériám, identifikované menom, priezviskom a dátumom narodenia. Osoba je vyhľadaná AQL dotazom na databázu (metóda *DbConnector/queryPersonSearch*, pričom sa dotaz líši podľa toho, či obsahuje alebo neobsahuje rok narodenia. Neexistuje totižto spôsob, ako vyhľadať "podobný" rok narodenia, tak ako je to pri mene a priezvisku. Pri tých je možné v dotaze použiť operátor %, ktorý je zhodný s akýmkoľvek počtom ľubovoľných znakov, teda výraz "%an%" je zhodný s menom Jan, ale aj menom Andrea alebo Alexandra.

Užívateľ si zo zobrazených osôb jednu vyberie a pokračuje na stránku **Detail osoby**(5.4.4).

5.4.3 Pridanie osoby

Pre pridanie osoby sa využila knižnica **JSON Editor**(5.3.1) - ďalej iba editor. Ako už bolo spomenuté, toto riešenie bolo zvolené ako schodnejšia cesta oproti ručne vyrobenému formuláru, kde sa narážalo stále na ďalšie a ďalšie problémy pri implementácii.

V editore je nastavená de facto rovnaká JSON schéma ako v databáze genealogických modelov v ArangoDB. De facto preto, že ArangoDB vyžaduje okrem definície JSON

schémy navyše ďalšie atribúty (správu pre užívateľa za porušenie schémy a úroveň vynucovania schémy, viz. 4.2.2). Samotný editor však nevie vynútiť dodržanie tejto schémy pred odoslaním JSON súboru, vie iba upozorniť užívateľa, že schému nedodržal, a to samostatne pre každý chýbajúci či chybne vyplnený atribút (pri každej chybe je vďaka možnostiam nastavenia editoru vytvorené upozornenie špecifické pre daný atribút). Na overenie dodržania schémy musela byť dodatočne pri odosielacom tlačítku *Add* pridaná kontrola na existenciu povinných údajov. Kombináciou kontroly editora a kontroly výstupného JSON súboru pri odoslaní sa zaistí, že sa do databázy nedostanú chybné údaje, a zároveň je chybné údaje dostatočne upozornený užívateľ.

Po odoslaní JSON súboru s údajmi novej osoby AJAX komunikáciu na server sa najprv pomocou metódy *DbConnector/queryPersonExists* dotazom na databázu zistí, či daná osoba už v databáze neexistuje, a to podľa povinných atribútov(meno, priezvisko, dátum narodenia, pohlavie). Pokiaľ áno, je skutočnosť oznámená užívateľovi, a je vrátený späť na stránku pridania osoby s editorom. Ak však osoba neexistuje, metódou *DbConnector/createPerson* je osoba v databáze vytvorená. Databáza následne vráti ID vytvorenej osoby, ktoré sa zo servera vracia ako AJAX odpoveď. Pomocou tohoto ID ako POST parametru je užívateľ presmerovaný na stránku **Detail osoby**(5.4.4), kde sa mu zobrazia údaje tejto novovytvorenej osoby.

5.4.4 Detail osoby

Práve táto obrazovka je vstupným bodom k väčšine interakcií s osobou uloženou v databáze genealogických modelov, vytvorenej v tejto práci. Zobrazujú sa to všetky informácie o osobe z databázy, pričom nie sú nijak formátované, a to z dôvodu, že nie je možné pripraviť formátovanie pre všetky možné atribúty, keďže osoba môže mať ľubovoľný počet ľubovoľných atribútov. Drobným odlíšením je meno a priezvisko, ktoré sú oddelené od zvyšku atribútov písmom, formátovaním aj umiestnením, a to z dôvodu, že sú tieto atribúty povinné, a je teda isté, že ich osoba bude mať.

Užívateľ má možnosť osobu upraviť cez tlačítko **Edit**(5.4.4). Taktiež môže osobu zmazať tlačidlom **Delete**(5.4.4). Kľúčovým prvkom je však prístup k rodostromu tejto osoby, keďže práve to bolo hlavným zameraním tejto práce. Užívateľ sa k nemu dostane tlačítkom **Show family tree**, a ďalej bude o tejto funkcii pojednávať sekcia 5.4.5. V neposlednom rade je tu funkcia odkazu na záznam danej osoby v pôvodnej databáze projektu DEMoS v podobe tlačítka **Show in DEMoS database**. Viac v sekcii 5.4.6.

Treba spomenúť, že celá táto podsekcia užívateľského rozhrania funguje predávaním si ID osoby cez POST správy, čím sa minimalizuje možnosť užívateľa s týmto ID akokoľvek prísť do styku.

Úprava osoby

Z rovnakých dôvodov ako pri **Pridaní osoby**(5.4.3) bola pri úprave osoby použitá knižnica **JSON Editor**(5.3.1). Taktiež bola použitá rovnaká schéma ako v databáze genealogických modelov ArangoDB.

Zmenou však bolo to, že v tomto prípade sú ešte pred zobrazením editoru získané dostupné údaje o osobe z databázy rovnakou metódou ako pri detaile osoby (neprenášajú sa z tejto obrazovky z dôvodu, aby boli na obrazovke úpravy vždy čo najnovšie informácie z databázy). Následne je týmito údajmi editor predvyplnený, a až potom sa zobrazí užívateľovi (samozrejme sa toto deje bez toho, aby to užívateľ mal šancu postrehnúť).

Užívateľ môže následne tieto existujúce dáta upravovať všetkými dostupnými možnosťami editora, pričom pokiaľ poruší pri úpravách schému, editor užívateľa samozrejme upozorní. Pri ukladaní upravených dát je implementovaná rovnaká kontrola, ako pri pridaní osoby, teda užívateľské rozhranie nedovolí užívateľovi do databázy odoslať záznam, ktorý neobsahuje povinné atribúty, alebo pri nich obsahuje chybné údaje.

Po upravení údajov je užívateľ presmerovaný späť na **Detail osoby**(5.4.4). Má však zároveň možnosť sa na túto obrazovku vrátiť aj bez akýchkoľvek úprav tlačítkom *Cancel*.

Zmazanie osoby

Funkcia nie je implementovaná ako samostatná obrazovka (na to zrejme ani nie je dôvod), a preto pre ňu neexistuje ani samostatný súbor. Ide len o krátku funkciu, ktorá má za úlohu jediné, a to zmazať osobu z databázy spolu so všetkými je prepojeniami v grafe vzťahov. Toto sa môže zdať ako jednoduchý úkon, no problém je v tom, že ArangoDB pri zmazaní záznamu z kolekcie ďalej nevyhľadáva, či neexistuje odkaz na tento záznam v iných kolekciách. Teda v prípade, že zmažeme záznam o osobe "Jan Novák" z kolekcie **Persons**, vzťahy, v ktorých figuruje ako atribút *_from* alebo *_to* sa automaticky z kolekcie **Relationships** nezmažú.

Práve z tohoto dôvodu bola vytvorená metóda *DbConnector/deletePersonById*, ktorá okrem vymazania osoby z kolekcie **Persons** naviac vykoná dotaz na databázu, ktorý vyhľadá všetky vzťahy v kolekcii **Relationships**, v ktorých osoba figuruje, a následne ich zmaže. Takto sa zaistí, že v databáze nezostanú nikam nevedúce vzťahy.

Dalo by sa argumentovať, že by nebolo na škodu vzťahy v databáze nechať aspoň ako orientačné, teda aj ak o danom človeku informácie už neexistujú, mohol by o tomto vzťahu ostať aspoň základný záznam. Avšak jediná informácia, ktorá by v kolekcii **Relationships** po osobe zostala, by bolo jej ID v tvare "Persons/_key", čo by nebola ničím užitočná informácia, vzhľadom na to, že aj ak by osoba s rovnakým menom, priezviskom, dátumom narodenia a pohlavím bola opäť vytvorená, jej ID by bolo iné. Údaj o ID by bolo teda úplne zbytočné v databáze ďalej uchovávať.

5.4.5 Zobrazenie rodostromu

Hlavnou časťou druhej polovice práce (ako aj názov napovedá) je vizualizácia genealogických modelov uložených v databáze. Zobrazenie akéhokoľvek rodostromu vyžaduje východiskovú osobu. Z tohoto dôvodu je možné rodostrom v tomto užívateľskom rozhraní zobraziť jedine z obrazovky **Detail osoby**(5.4.4), teda aby sa východiskovou osobou stala osoba, ktorej detaily boli predtým zobrazené.

Zobrazenie rodostromu osoby je realizované pomocou knižnice **Sigma js**(5.3.2). Graf vytváraný knižnicou prijíma dve polia vstupných dát, a to *nodes* a *edges*. Na dodanie

týchto vstupov je použitá metóda *DbConnector/queryGraphOfPerson* obsahujúca zložený dotaz na databázu. Ako argumenty metóda prijíma:

1. $id - ID východiskovej osoby
2. $depth - maximálnu vzdialenosť iných osôb v grafe vo vzťahu k východiskovej osobe (teda napr. ak je vzdialenosť 1, objavia sa iba rodičia, ale nie starí rodičia, iba deti, ale nie vnúčatá)
3. $relationships - množinu vzťahov na zobrazenie (môže obsahovať iba hodnoty vymenované v sekcii 4.2.3)

Prvá časť dotazu vytvorí v databáze dočasnú premennú *edges*, do ktorej uloží všetky hrany, obsahujúce východiskovú osobu a typ vzťahu uvedený v množine vzťahov $relationships. Druhá časť vytvorí dočasnú premennú *nodes*, do ktorej uloží všetky uzly, v tomto prípade osoby, ktoré sú danými hranami prepojené s východiskovou osobou. Nakoniec vráti ako odpoveď tieto dve premenné v jednom poli. Všetky potrebné podoperácie na zozbieranie potrebných dát sa tak vykonávajú na strane databázy, čo je v tomto prípade niekoľkonásobne efektívnejšie, než triedenie dát na strane serveru.

Výsledné pole uzlov a hrán je využité na zostavenie grafu knižnicou Sigma js. Tieto dáta obsahujú (narozdiel od ich využitia v iných moduloch) aj interné atribúty, najmä atribút *_id*, ktorý je v prípade hrán aj uzlov pri vykreslení použitý ako unikátne ID uzlu alebo hrany v grafe. To je užitočné pri spätnej operácií získavania informácii o uzle po kliknutí naň v grafe(čo je taktiež osobitne implementovanou JavaScript funkciou).

Graf, ktorý je zobrazený, sa podobá tomu, ktorý môže užívateľ vidieť vo webovom rozhraní ArangoDB, vzhľadom na to, že v oboch prípadoch ide o knižnicu Sigma js. Zásadným rozdielom je však to, že graf v tomto rozhraní je upravený presne pre tento systém, čo je vidno hneď pri prvom pohľade na akýkoľvek dostatočne rozvinutý zobrazený rodostrom.
Špecifické prvky implementácie grafu:

- východisková osoba je označená zelenou farbou, ako aj hrany smerujúce od nej
- mužské osoby sú označené modrou farbou, ženské osoby ružovou farbou, rovnako ako ich príslušné hrany smerujúce od nich
- obojsmerné hrany sú označené čiernou farbou, a nie sú ukončené šípkou (narozdiel od orientovaných hrán)
- hrany majú zobrazené názvy vzťahov, ktoré reprezentujú, k čomu je využitá doplnková knižnica *sigma.renderers.edgeLabels*
- pri uzloch(osobách) sú zobrazené ako meno, tak ak priezvisko osoby

Náhľad detailu osoby

Súčasťou zobrazenia rodostromu je okno s náhľadom detailu osoby. Osoba v náhľade sa mení kliknutím na osobu(uzol) v grafe. Okno zobrazuje všetky informácie o osobe

rovnako ako obrazovka *Detail osoby*(5.4.4), no ponúka odlišné možnosti práce s osobou. Čo sa týka východiskovej osoby, jedinou možnosťou je **View detail**, ktorá je prítomná aj pri ostatných osobách. Užívateľ sa funkciou dostane na obrazovku *Detail osoby*. Pri iných osobách sa však pridajú tlačítka **Make root person** a **Delete from family tree**.

Make root person zmení zvolenú osobu za východiskovú a prekreslí celý graf. Funkcia je užitočná najmä pre prípad, že chce užívateľ vidieť vzdialenejšie vzťahy danej osoby, a nechce zvyšovať počet zobrazených úrovní grafu.

Delete from family tree vymaže vzťah východiskovej osoby a zvolenej osoby v databáze. Dáta o osobách nie sú nijak inak ovplyvnené. Východisková osoba tak samozrejme stratí aj prepojenie na osoby, ktoré boli spojené so zvolenou osobou. Po odstránení osoby je graf prekreslený bez opätovného načítania stránky, a to AJAX komunikáciou.

Filtre

Pre zlepšenie prehľadnosti grafu je v zobrazení rodostromu možné jednotlivé typy vzťahov medzi osobami zobraziť alebo skryť. Filtrovať sa dajú všetky typy vzťahov vymenované v sekcii 4.2.3. Skrytie alebo zobrazenie vzťahov vyvolá asynchrónne volanie pre prekreslenie grafu, ktoré sa následne udeje bez opätovného načítania stránky. V pozadí sa jednoducho znovu použije funkcia *DbConnector/queryGraphOfPerson*, no s novou hodnotou pre parameter $relationships, konkrétne s polom hodnôt získaných z bloku filtrov.

Rovnako sa tu nachádza aj zmena počtu zobrazených úrovní grafu. Pracuje rovnako interaktívne ako nastavovanie filtrov, a reaguje na zmenu čísla buď po stlačení klávesy Enter, alebo po kliknutí mimo zadávacieho poľa.

Pridanie osoby do rodostromu

Toto tlačítko je implementované ako modálne okno. Využitá je knižnica **jQuery Modal**[14], pomocou ktorej je veľmi jednoduché modálne okno vytvoriť. Obsah tohoto okna je implementovaný v samostatnom súbore(modalSearch.php), pričom ide o modifikovanú verziu okna **Vyhľadávanie osoby**(5.4.2). Postup pridania osoby do rodostromu je teda podobný postupu zobrazenia detailu osoby, avšak naviac je ešte nutné vybrať typ vzťahu s východiskovou osobou.
V modálnom okne existujú dva logické postupy:

1. Vyhľadanie existujúcej osoby:
   (a) Vyhľadanie osoby pomocou mena, priezviska, dátumu narodenia a pohlavie (všetko je voliteľné, užívateľ môže jednoducho stlačiť tlačítko **Search** a zobrazí sa mu 100 náhodných výsledkov)
   (b) Zvolenie jednej osoby zo zobrazených výsledkov a následné zvolenie si jedného zo vzťahov medzi danými osobami (pri orientovaných vzťahoch sú vždy dve možnosti, texty sú napísané tak, aby bolo jednoznačne určiteľné o aký smer vzťahu ide)

[14] https://jquerymodal.com/

(c) Potvrdiť pridanie do rodostromu osoby

2. Pridanie novej osoby pomocou tlačítka **Add new person to DB**, ktorá užívateľa presmeruje na stránku pridania osoby do databázy(5.4.3)

5.4.6 Detail záznamu v databáze DEMoS

Dôležitou požiadavkou bolo prepojenie osôb s pôvodnou databázou projektu DEMoS. V **Detaile osoby**(5.4.4) sa z tohoto dôvodu nachádza tlačítko **Show in DEMoS database**. Ide o odkaz, pomocou ktorého je užívateľ presmerovaný na k tomu určenú stránku v systéme DEMoS, a pomocou GET dotazu v linku je zobrazený príslušný užívateľ v databáze, a všetky k nemu príslušné záznamy. Identifikácia osoby v obidvoch databázach je zabezpečená tak, že osoby v databáze genealogických modelov majú pole *_key* zhodné s primárnym kľúčom v databáze DEMoSu.

V prípade, že sa adresa zobrazenia v DEMoSe zmení, je nutné ju zmeniť v súbore Templates/personDisplayTemplate.php kde sa nachádza prvok typu "a" s id "demos". Tu stačí zmeniť "href" atribút, no je nutné tu ponechať časť <?php echo $person->getKey() ?>, ktorá na toto miesto vytlačí kľúč osoby.

5.5 Test systému

Na testovanie celého systému bola garantom práce poskytnutá vzorka genealogických záznamov z obce Bukovinka prepísaných do formátu *xlsx* (Excel tabuľka). Konkrétne išlo o tri súbory obsahujúce následujúce záznamy(spomenuté sú iba relevantné stĺpce):

- Narodenia:
  1. dátum krstu - dá sa považovať za dátum narodenia - rozdiel je pár dní
  2. ID dieťaťa - použité ako kľúč *_key* a ako identifikátor v pôvodnej databáze DEMoSu
  3. meno dieťaťa
  4. pohlavie dieťaťa
  5. priezvisko otca - následne použité ako meno dieťaťa
  6. ID otca - pre následné spojenie otca a dieťaťa vzťahom
- Svadby (tu stačia iba identifikátory pre spojenie už existujúcich osôb):
  1. ID ženícha
  2. ID nevesty
- Úmrtia:
  1. dátum úmrtia - pri asi štvrtine záznamov nie je
  2. dátum pochovania

3. ID zosnulého - pre spojenie s osobou z narodených osôb

Tieto dáta bolo nutné z pôvodných súborov vyfiltrovať a následne uložiť do súboru formátu *csv*[15] (textový súbor, kde sú hodnoty oddelené čiarkami), ktorý vie ArangoDB spracovať. K tomuto bol vytvorený krátky skript vytvorený v jazyku Python[16], využívajúci knižnicu *openpyxl*[17] na čítanie a navigáciu v *xlsx* súboroch. V skripte sa nachádzajú dve hlavné funkcie, a to *births_and_deaths* a *marriages*.

births_and_deaths načítava najskôr zo súboru úmrtí do asociatívneho pola (ďalej iba slovník z anglického Dictionary) dátum úmrtia a pochovania, a ako kľúč použije ID zosnulého. Ďalej sa ako prvý riadok csv súboru vložia názvy atribútov v poradí ako budú zapísané v nasledujúcich záznamov osôb. V tomto riadku je definované, že ID narodených sa budú ukladať ako kľúčový atribút *_key*, mená ako *name*, atď. Skript potom preiteruje súbor narodení, a po získaní dát z riadku zistí, či sa ID dieťaťa nachádza v kľúčoch slovníka zosnulých (teda ID zosnulých). Pokiaľ áno, k záznamu osoby pridá aj dátum úmrtia a pochovania. Po dokončení zostavovania pola záznamov sa celé pole vezme, a zapíše do csv súboru kódovaním UTF-8(z dôvodu, že ArangoDB číta a ukladá všetky reťazce znakov iba v tomto kódovaní, kódovanie ANSI ktoré je pre csv súbor vytvorený Pythonom implicitné pri prvom pokuse vytvorilo chyby v menách vložených do databázy). Príklad dát vytvorených funkciou sa nachádza v tabuľke 5.1.

marriages jednoducho načíta dáta o číslach ID z dvoch príslušných stĺpcov v xlsx súboroch a spolu s prvým riadkom názvov atribútov ich zapíše do csv súboru. Stĺpce s atribútmi ID ženícha a nevesty sú však pomenované ako *_from* a *_to*, čím sa zaistí, že budú brané ako začiatočný a koncový uzol jednej hrany (manželstva) v grafe.

_key	date_of_birth	name	surname	gender	father_ID	date_of_burial	date_of_death
56	1680-12-21	Thomas	Bezruk	male	779		
800	1714-02-26	Joseph	Beranek	male	802	1782-06-14	
241	1739-11-12	Catharina	Kotulan	female	805	1813-06-21	1813-06-19
861	1759-10-13	Anna	Bezruk	female	860	1763-10-22	
294	1766-01-22	Fabianus	Opletal	male	289	1840-10-15	1840-10-13

Tabuľka 5.1: Ukážka dát vytvoreného *csv* súboru pomocou funkcie *births_and_deaths*(dáta sú pre prehľadnosť v tabuľke miesto pôvodného formátu, taktiež boli odstránené všetky úvodzovky okolo jednotlivých atribútov)

Po uložení dát do príslušných súborov (Persons.csv a Marriage.csv) bol použitý nástroj Arangoimport v prostredí PowerShell[18], a to následujúcim príkazom:

```
.\arangoimport --file "F:\...\Persons.csv" --ignore-missing --type csv
    --collection "Persons"
```

Odôvodnenie jednotlivých parametrov:

[15] https://en.wikipedia.org/wiki/Comma-separated_values
[16] https://www.python.org/
[17] https://openpyxl.readthedocs.io/en/stable/
[18] https://en.wikipedia.org/wiki/PowerShell

- –ignore-missing - niektoré osoby nemajú všetky atribúty (napr. chýba dátum úmrtia), toto zaistí, Arangoimport toto nebude brať ako chybu, keďže v prvom riadku, ktorý definuje počet a názvy atribútov tieto chýbajúce atribúty obsahuje
- –type csv - formát vstupného súboru je *csv*
- –collection "Persons" - osoby sú vkladané do kolekcie "Persons", ktorá je už pripravená v databáze

Po tomto kroku boli dáta úspešne importované do databázy so všetkými zadanými atribútmi. Ďalším krokom bolo nahratie manželstiev. Tu bolo však nutné pridať ďalšie parametre z dôvodu, že ide o import do hranovej kolekcie(3.3.3). Príkaz teda vyzeralo takto:

```
.\arangoimport --file "F:\...\Marriages.csv" --ignore-missing --type csv
--collection "Relationships" --from-collection-prefix "Persons"
--to-collection-prefix "Persons"
```

Hranové kolekcie využívajú v parametroch *_from* a *_to* celé ArangoDB *_id*, nie iba kľúče *_key*. Tvorcovia Arangoimport však rátali s tým, že užívateľ nemusí mať k dispozícii meno kolekcie, v ktorej sú spájané uzly uložené. Preto existujú užitočné parametre *–from-collection-prefix* a *–to-collection-prefix*, ktoré automaticky do stĺpcov pre atribúty _from a _to ku kľúčom doplnia zdrojovú kolekciu podľa zadanej hodnoty(v našom prípade kolekciu "Persons").

So všetkými potrebnými dátami v databáze bolo ďalším logickým krokom vytvorenie rodičovských väzieb. Pre tento test sa pod týmto dá chápať väzba otca a jeho detí, ktorá je zistená pomocou atribútu *father_ID*. Na vytvorenie hrán v kolekcii **Relationships** bol pomocou webového rozhrania vykonaný tento dotaz:

```
for person in Persons
    filter person.father_ID != ''
    for father in Persons
        filter TO_STRING(person.father_ID) == father._key
        insert { _from: father._id, _to: person._id, relationship: "
            parent of" }
            into Relationships
```

Po vykonaní tohoto dotazu sa v kolekcii **Relationships** nachádzajú väzby medzi všetkými osobami s atribútom *father_ID* a osobami s týmto číslom ako atribútom *_key*. Pre úplnosť je nutné podotknúť, že bolo treba vykonať dotaz na zmazanie hrán manželstiev importovaných zo súboru manželstiev, v ktorých sa nachádzali ID čísla osôb, ktoré sa nenachádzali v záznamoch narodení. Odkazovali teda na neexistujúci záznam, čo vytváralo chyby.

Nasledoval test webového užívateľského rozhrania, kde bola vyhľadaná osoba s pomocou prvého mena, konkrétne "Justyna", a bola vybratá osoba Justyny Wopletalovej,

narodenej 27.4.1679. Po zobrazení detailu osoby sa správne zobrazili všetky údaje nachádzajúce sa v databáze, ako aj všetky možnosti. Osobu sa podarilo úspešne upraviť, a to skúškou pridania atribútu ”test” s hodnotou ”test”. Táto zmena sa ihneď preniesla do databázy. Potom nasledoval test zobrazenia rodostromu, ktorý sa zobrazil korektne, čo bolo overené porovnaním so zobrazením grafu danej osoby vo webovom rozhraní ArangoDB. V zobrazení správe fungovalo filtrovanie zobrazenia typov vzťahov, ako aj úroveň zobrazenia spojení v rodostrome.

Justyne bola pridaná sestra, a to náhodne vybraná Anna Wopletal, narodená 12.5.1690. V grafe sa osoba ihneď objavila. Bolo možné ju označiť ako východiskovú osobu, ako aj odstrániť ju z rodostromu Justyny.

Systém reagoval správne aj pri pridaní úplne novej osoby do databázy s parametrami vo výpise 5.2.

```
{
    "name" : "test",
    "surname" : "person",
    "date_of_birth" : "1812-12-12",
    "gender" : "male"
}
```

Výpis 5.2: Parametre testovacieho záznamu vo formáte JSON

Test systému bol teda úspešný, a všetky otestované funkcie fungovali správne. Webové užívateľské rozhranie s databázou komunikuje správe, pričom by v žiadnej z hlavných funkcií tohoto rozhrania nemala byť zásadná chyba.

5.6 Ďalší vývoj

Budúci vývoj systému by sa mohol niesť v zlepšovaní vizuálnej stránky užívateľského rozhrania. Taktiež je takmer isté, že sa v rozhraní nachádzajú neodhalené chyby alebo iné defekty, ktoré by sa mali priebežne opravovať. Existuje zároveň aj veľký priestor pre pridávanie ďalších funkcií do správy rodostromu osoby, kde chýba napríklad funkcia pridania úplne novej osoby do rodostromu inak ako presmerovaním na obrazovku pridania osoby do databázy, prípadne zrušenie celého rodostromu osoby inak ako zmazaním danej osoby.

Určite by taktiež bolo vhodné implementovať v užívateľskom rozhraní funkciu na import súborov typu *csv*. Metóda pre túto funkciu v ArangoDB-PHP existuje, ale je veľmi zle zdokumentovaná, a preto sa ju nepodarilo začleniť do rozhrania.

Ak by na to bola vôľa, určite by bolo možné celé webové užívateľské rozhranie prepísať do ktoréhokoľvek moderného webstránkového frameworku, avšak určite by to vyžadovalo predchádzajúce skúsenosti s takouto implementáciou.

Systém je pripravený byť integrovaný do webového rozhrania projektu DEMoS, kde stačí pripravené PHP súbory vložiť do existujúcej štruktúry súborov na serveri projektu.

Kapitola 6

Záver

Porovnaním rôznych typov databáz a následným porovnaním konkrétnych databázových riešení pre daný typ bolo vybrané najvhodnejšie riešenie pre problém ukladania genealogických modelov. Databáza, ktorá bola v tejto práci vytvorená, ako aj webové užívateľské rozhranie sú výsledkom snahy o vytvorenie efektívneho systému, ktorý zohľadňuje špecifické potreby týchto modelov, kvôli ktorým je ich ukladanie v klasickej relačnej databáze menej efektívne.

Vybraté databázové riešenie ArangoDB je veľmi moderným a pokročilým databázovým systémom typu multi-model, ktorý vie veľmi dobre narábať s rozličnými počtami a typmi atribútov záznamov uložených v jednej štruktúre. ArangoDB dokáže ukladať dáta do dokumentov vo formáte JSON, a následne ich spájať hranami grafu, spája tak vlastnosti dokumentovej a grafovej databázy.

Navrhnutá štruktúra databázy zodpovedá všetkým požiadavkám definovaným v zadaní práce, a je plne uspôsobená na ukladanie genealogických záznamov osôb ako aj všetkých vzťahov medzi nimi. Zároveň ide o veľmi jednoducho rozšíriteľný systém pre budúce potreby genealogických výskumníkov, ktorý by so systémom pracovali. Ďalšími pozitívami je rýchlosť databázy a pokročilá práca s dátami pomocou databázy vlastného jazyka AQL.

Čo sa týka webového užívateľského rozhrania, boli doň zakomponované všetky funkcie potrebné pre správu genealogických modelov jednotlivcov v databáze, spravovanie týchto osôb, ako aj ich prepojenie s pôvodnou databázou. Kľúčová časť tohoto rozhrania, zobrazenie rodostromu osoby, bola navrhnutá a implementovaná tak, aby bola čo najlepším kompromisom medzi výhodami multi-modelovej databázy a klasickým zobrazením rodostromov.

V práci sú popísané postupy pri navrhovaní a implementácii systému, ako aj práca s ním. Pre genealogických výskumníkov teda môže byť vhodným odrazovým mostíkom pre prácu s ním, ako aj jeho prípadné rozširovanie.

Literatúra

[1] Grolinger, K., Higashino, W. A., Tiwari, A. a Capretz, M. A. Data management in cloud environments: NoSQL and NewSQL data stores. *Journal of Cloud Computing: Advances, Systems and Applications* [online]. December 2013, zv. 2, č. 22, s. 4, [cit. 2021-02-28]. Journal of Cloud Computing: Advances, Systems and Applications. DOI: 10.1186/2192-113X-2-22. ISSN 2192-113X. Dostupné z: https://journalofcloudcomputing.springeropen.com/articles/10.1186/2192-113X-2-22.

[2] Inc, A. *ArangoDB v3.7.7 Documentation: Data models & modeling* [online]. 1. vyd. 411 Borel Ave. Suite 650, San Mateo, CA 94402, United States: ArangoDB Inc [cit. 2021-03-14]. Dostupné z: https://www.arangodb.com/docs/stable/data-modeling.html.

[3] Inc, A. *ArangoDB v3.7.7 Documentation: Schema Validation - Levels* [online]. 1. vyd. 411 Borel Ave. Suite 650, San Mateo, CA 94402, United States: ArangoDB Inc [cit. 2021-04-07]. Dostupné z: https://www.arangodb.com/docs/stable/data-modeling-documents-schema-validation.html#levels.

[4] Inc, A. *ArangoDB v3.7.7 Documentation: HTTP Request Handling in ArangoDB* [online]. 1. vyd. 411 Borel Ave. Suite 650, San Mateo, CA 94402, United States: ArangoDB Inc [cit. 2021-03-24]. Dostupné z: https://www.arangodb.com/docs/stable/http/general.html.

[5] Jacomy, M., Venturini, T., Heymann, S. a Bastian, M. ForceAtlas2, a Continuous Graph Layout Algorithm for Handy Network Visualization Designed for the Gephi Software. *PLOS ONE*. Public Library of Science. Jún 2014, zv. 9, č. 6, s. 2. DOI: 0.1371/journal.pone.0098679. Dostupné z: https://doi.org/10.1371/journal.pone.0098679.

[6] Wikipedie. *Rodokmen — Wikipedie: Otevřená encyklopedie* [online]. 2020. [Online; navštíveno 28. 02. 2021]. Dostupné z: https://cs.wikipedia.org/w/index.php?title=Rodokmen&oldid=18662437.

[7] Zboril, F., Rozman, J. a Koci, R. Algorithmic Creation of Genealogical Models. In: Abraham, A., Cherukuri, A. K., Melin, P. a Gandhi, N., ed. *Intelligent*

Systems Design and Applications. Cham: Springer International Publishing, 2020, s. 650–658. ISBN 978-3-030-16660-1.

Príloha A

Návod na inštaláciu

- NÁVOD JE PRE INŠTALÁCIU NA SYSTÉM WINDOWS 10 (64-bit)
- Minimálna verzia ArangoDB 3.7
- Minimálna verzia PHP 7.3

Inštalácia databázy:

1. Stiahnuť a nainštalovať ArangoDB Community Edition (prípadne Enterprise edition - pár krokov naviac) z https://www.arangodb.com/download/
2. Nastaviť databázu do funkčného počiatočného stavu (root užívateľ, žiadna kolekcia, žiadny graf)
3. Pomocou prostredia PowerShell sa dostať do priečinka, kde sa nachádza súbor arangoimport.exe (napr. C:\Program Files\ArangoDB3e 3.7.3\usr\bin)
4. Spustiť príkaz .\arangorestore.exe –input-directory "cesta\ku\priečinku\DBdata"

Inštalácia servera(vykonať až PO inštalácii databázy):

1. Nainštalovať PHP (ideálne vo verzii 8), a to tak aby bolo dostupné cez príkazový riadok
2. Pomocou príkazového riadku sa navigovať do zložky FamilyTree v súboroch práce
3. Spustiť tu príkaz php -S localhost:8000