VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

SROVNÁNÍ DATABÁZOVÝCH SYSTÉMŮ PRO DIGITALIZACI MATRIČNÍCH ÚDAJŮ

COMPARISON OF DATABES SYSTEMS FOR THE REGISTERS DIGITALIZATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **MARTIN REŠOVSKÝ**
AUTHOR

VEDOUCÍ PRÁCE **Ing. RADEK KOČÍ, PhD.**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2018/2019

Zadání bakalářské práce

Student: **Rešovský Martin**
Program: Informační technologie
Název: **Srovnání databázových systémů pro digitalizaci matričních údajů**
Comparison of Databes Systems for the Registers Digitalization
Kategorie: Databáze

Zadání:

1. Seznamte se s procesem digitalizace matričních údajů a prostudujte strukturu informací o svatbách a úmrtích.
2. Navrhněte strukturu dat pro ukládání informací z matrik.
3. Prostudujte problematiku relačních a objektových databázových systémů. Seznamte se s vybranými volně dostupnými databázovými systémy a po dohodě s vedoucím práce vyberte zástupce z každé skupiny.
4. Navrhněte model dat pro relační a objektové databázové systémy. Vytvořte příslušné databáze ve vybraných databázových systémech.
5. Vytvořte testovací datovou sadu a porovnejte výkonnost různých databázových operací.
6. Vyhodnoťte vhodnost jednotlivých typů databází pro zpracovávaná data.

Literatura:

- Hibatullah Alzahrani. Evolution of Object-Oriented Database Systems. Global Journal of Computer Science and Technology: C, Software & Data Engineering vol. 16, issue 3, 2016.
- E. Olsson, O. Nordkvist. Object Oriented Databases. Dostupné elektronicky http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.462.892&rep=rep1&type=pdf, 2018.

Pro udělení zápočtu za první semestr je požadováno:

- První tři body zadání.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Kočí Radek, Ing., Ph.D.**
Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 1. listopadu 2018

Abstrakt

Táto práca prezentuje porovnávaciu analýzu relačných a objektových databázových systémov nad pevne špecifikovanou sadou dát získaných digitalizáciou matričných údajov o svadbách a úmrtiach. Dáta boli spracovávané pomocou vybraných zástupcov svojich kategórii Db4o a PostgreSQL. Pre obe databázy bol vytvorený a popísaný dátový model. Vyhodnotenie bolo prevedené na základe porovnania teoretických základov, implementácie a výsledkov výkonnostných testov databáz nad vytvorenou testovacou sadou ekvivalentnou s originálnou vzorkovou. Vykonaným výskumom bolo zistené, že má relačný model pevnejšie teoretické základy, omnoho lepšiu podporu a správu a je aj výkonnejší. Objektový prístup však modeluje databázu oveľa prirodzenejšie a predchádza konštrukčným komplikáciám. Hlavným výsledkom je, že relačný databázový model sa ukazuje ako výkonnejší a vhodnejší a tým pádom spolu s nadobudnutými znalosťami môžeme konštatovať, že má i lepšie predpoklady na efektívnejšie spracovanie úmrtných a svadobných dát.

Abstract

This paper presents comparative analysis of relational and object–based database systems over firmly specified data set gathered by registers digitalization of wedding and death records. Data were processed by chosen representants of theirs categories Db4o and PostgreSQL. Data model was created and described for both databases. Evaluation was made on the base of comparison of theoretical foundations, implementation and results of performance tests of the databases on test dataset equivalent to original sample. By carried out research, we found out, that relational model has stronger theoretical foundations, much better support and administration, and is more efficient. On the other hand, object–based approach models database much more naturally and preced construction complications. Main result is, that relational database model appears to be more efficient and preferable, and therefore, with gained knowledge, we can state, that it has better preconditions for effective processing of death and wedding historic records.

Kľúčové slová

Databáza, databázové systémy, relačná databáza, objektová databáza, db4o, PostgreSQL, digitalizácia dát.

Keywords

Database, database systems, relational database, object–based database, dbo4o, PostgreSQL, data digitalization.

Citácia

REŠOVSKÝ, Martin. *Srovnání databázových systémů pro digitalizaci matričních údajů.* Brno, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Radek Kočí, PhD.

Srovnání databázových systémů pro digitalizaci matričních údajů

Prehlásenie

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

.....
Martin Rešovský
15.5.2019

Poďakovanie

Rád by som sa poďakoval doktorovi Radkovi Kočímu za poskytnutú pomoc, čas a trpezlivosť, ktorú mi venoval pri konzultáciách a inžinierovi Vladimírovi Žemlovi za poskytnutie informácii o digitalizácii dát.

Obsah

Kapitola 1

Úvod

Počínajúc rokom 1970, keď Edgar Codd publikoval jeho článok [9], relačné databázy dominovali trhu. V súčasnosti je až sedem pozícii v rebríčku top 10 najobľúbenejších databáz obsadených práve týmto typom [26]. V posledných rokoch však zaznamenávame dynamický rast Internetu a mobilných zariadení, čo spôsobuje enormný nárast generovaných dát v štruktúrovanej i neštrukturovanej podobe. Spracovanie tak rozsiahleho množstva dát vyžaduje rýchlosť, flexibilné schémy a distribuované databázy. A tak sa odborná verejnosť čoraz častejšie začala obracať na alternatívne riešenia k relačným databázam. Výsledkom tohto procesu sú NoSQL databázy,[1] ktoré často ponúkajú lepšie možnosti škálovania výkonu a oveľa flexibilnejšie dátové modely. Čiastočným cieľom tejto práce je porovnať vybranú relačnú databázu PostgreSQL, s vybranou NoSQL (non–relational) databázou Db4o, a vniesť tak exaktné a objektívne výsledky do často subjektívnej diskusie na danú tému.

Na čo však slúžia databázy vo všeobecnosti? Encyklopedický portál ich definuje ako „organizovanú kolekciu dát, vo všeobecnosti uloženú a prístupnú elektronicky“[2], a ďalej tvrdí, že predchodcom databáz boli papierové kartotéky. A práve špeciálnym typom týchto kartoték – matričnými záznamami – a ich digitalizáciou a prevodom do modernej, prístupnej a databázovej podoby sa zaoberá táto práca. Tieto stovky rokov zbierané údaje patria k najčastejším historickým prameňom a často obsahujú predkov a históriu mnohých z nás. Avšak bez digitálnej podoby a databázového spracovania upadajú naveky do zabudnutia. A ako povedal George Santayana, tí, ktorí si nepamätajú svoju históriu, sú odsúdení ju zopakovať. Preto je hlavným cieľom tejto práce vytvoriť vo vyššie spomenutých databázach modely a funkčnú implementáciu pre digitalizované matričné údaje Českej Republiky o úmrtiach a svadbách, a nad touto aplikačnou doménou porovnať spomenuté databázové systémy. V súlade s touto špecifikáciou bude výskum zameraný na výkon a vhodnosť jednotlivých systémov pre dané dátové sady, a nie pre ich všeobecné využitie, čo však nemusí nevyhnutne znamenať rozdiel.

Práca začína popisom procesov digitalizácie, vytvárania databázy a spracovávania dát. Pokračuje teoretickým prehľadom problematiky, základných princípov a všeobecným porovnaním relačných a objektových databáz. Ďalej sa venujeme analýze a návrhu štruktúry a modelov úmrtných a svadobných dát. Ďalšou podstatnou časťou je vytvorenie testovacej sady a testovanie databáz. Na záver hodnotíme databázy zo všetkých rozoberaných pohľadov a uvádzame konečné rozhodnutie.

[1] https://en.wikipedia.org/wiki/NoSQL
[2] https://en.wikipedia.org/wiki/Database

Kapitola 2

Digitalizácia a spracovanie údajov

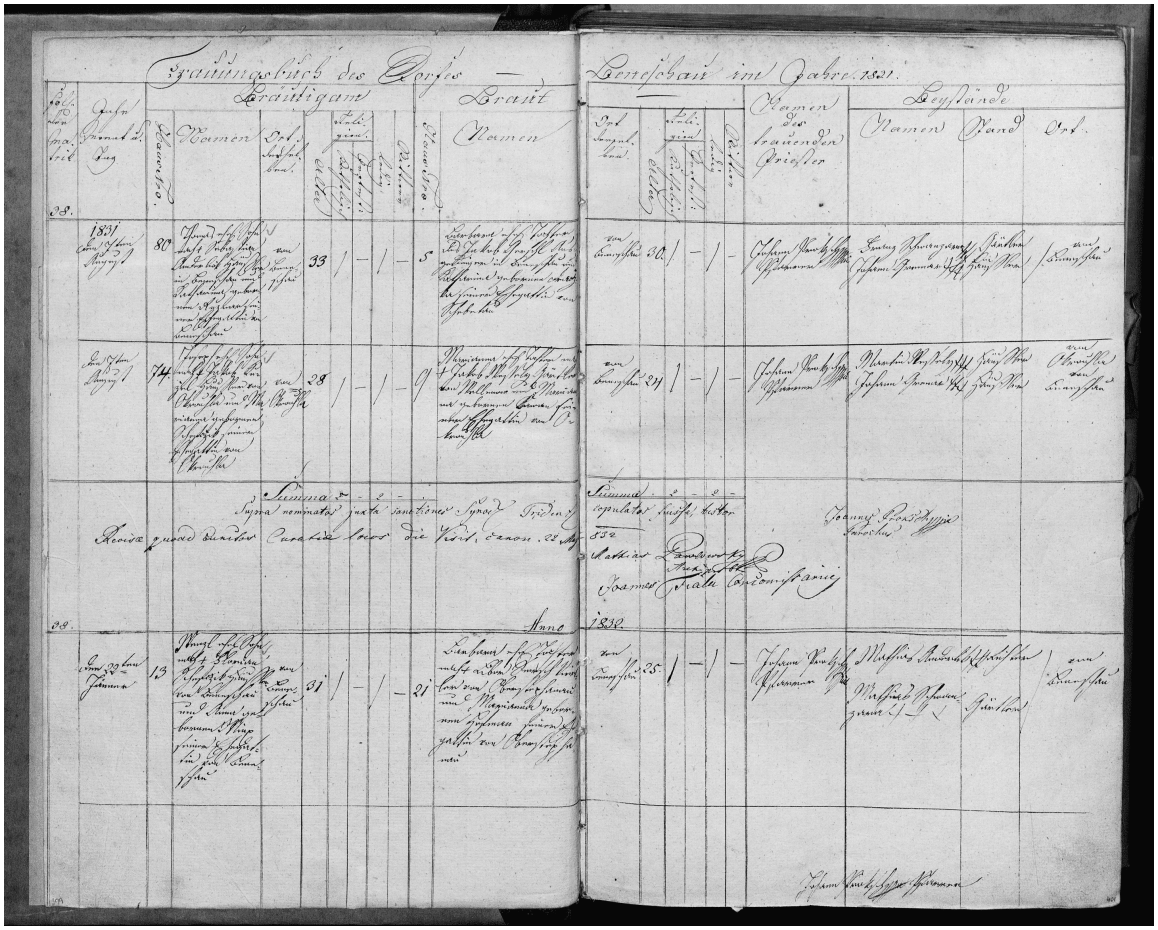
Na tému porovnávania databázových systémov vo všeobecnosti, ale aj špecificky zvolených, vzniklo už množstvo prác a výskumov ktoré sčasti excelentne a sčasti nedostatočne pokryli väčšinu priestoru v tejto oblasti. I keď je nutné poznamenať, že oblasť databázových systémov nie je pre zmenu staticky uzavretá, ale naopak, hardwarovo i softwarovo dynamicky sa meniaca – čo vyžaduje neustály posun a aktualizáciu i vo výskume. Táto práca je však špecifická konkrétnym zadaním vstupných dát a očakávaným cieľom nie je len výstup v podobe porovnania zvolených databázových systémov, ale aj plne funkčné a implementované databázové modely efektívne udržujúce už spomínanú dátovú sadu. Pre detailné porozumenie priebehu výskumu, implementácie a vyvodzovania záverov, ale i pochopenie motivácie za zvolením jednotlivých testovacích dotazov, či niekedy prekvapivých konštrukcii v návrhu modelov, je preto nevyhnutné vstupné dáta, ich štruktúru, proces získavania a ich budúce využitie popísať. Veď už samotný návrh modelov môže byť len ťažko ideálne a hlavne efektívne vyhotovený bez veľmi dobrej znalosti vstupných hodnôt pre ktoré je tvorený.

Ako je už v úvode i v názve tejto kapitoly uvedené, spomínaná dátová sada je teda sada matričných údajov, konkrétne záznamy o **svadbách** a **úmrtiach** ľudí žijúcich prevažne na území dnešnej Českej republiky. Tieto informácie boli po niekoľko sto rokov zapisované v matrikách náboženských spoločenstiev a patria dnes k najčastejšie využívaným historickým prameňom. Ich digitalizácia bola prevedená v rámci projektu *„European Territorial Cooperation – Austria–Czech Republic 2007-2013“*[3] ktorý začal rokom 2005 a jeho cieľom bolo vytvoriť webovú aplikáciu na zverejnenie nascanovaných záznamov. To sa vytvorením aplikácie ActaPublica[4] i podarilo (ukážka v obrázku 2.1), ale slovami vedúceho projektu, inžiniera Vladimíra Žemlu, aj napriek pravidelnej mesačnej údržbe a rozširovaniu sprístupnených dát projekt narazil na technické limity pôvodnej implementácie – hlavne neaktualizovateľná a zastaraná technológia flash player, neekonomická údržba scanneru, náročná oprava chýb a neexistujúca interná centrálna databáza - ale i na svoju dobu životnosti. Z týchto okolností, ako i z túžby po vylepšení možností práce s digitalizovanými dátami vznikol grantový projekt v spolupráci s Masarykovou univerzitou, ktorého súčasťou je i táto práca.

[2] http://actapublica.eu/

[3] https://ec.europa.eu/regional_policy/EN/atlas/programmes/2007-2013/crossborder/operational-programme-austria-czech-republic

[4] http://actapublica.eu/



Obr. 2.1: Ukážka nascanovaného matričného záznamu zhotoveného rímsko–katolíckou cirkvou evidujúceho údaje o svadbách v oblasti Brno–Bohunice v rokoch 1779 – 1784.[2]

2.1 Postup digitalizácie a prvotné spracovanie

Konkrétny postup vytvorenia digitálnych tabuľkových záznamov od obrázku 2.1 cez zverejnenie v online aplikácii ActaPublica až po digitálnu vzorku dát uloženú v aplikácii Microsoft Excel:

1. Spracovávateľ oskenuje dvoj–stránku z matriky.
2. Sken sa vloží do systému a zaregistruje.
3. Sken sa uloží a publikuje na serveri ActaPublica. Žiadna interná databáza neexistuje.
4. Pracovníci zúčastňujúci sa projektu na Masarykovej univerzite ručne prepíšu dáta do Excelu a poskytnú ich na ďalšie spracovanie v tabuľkovej forme.

Výsledok tejto „kompilácie“[5] všetkých matrík spracovávateľmi[6] je ďalej vstupom pre túto prácu. Následne sme začali so spracovávaním zaznamenaných informácii – pomenujme ich atribúty. Tie sú prirodzene rozdelené v skupinách – nazvime ich entity – podľa toho,

[5] vložení všetkých údajov zo všetkých matričných kníh do jedného dokumentu s jednou formou.
[6] http://www.mza.cz/oddeleni-it-digitalizace

na akú entitu sa daný atribút vzťahuje. Je však potrebné si uvedomiť, že tieto atribúty a entity nie sú ešte časťou databázového modelu. Reprezentujú len logickú štruktúru čistého prepisu archívnych záznamov. Atribúty, ich dátové typy a pridelenie k entitám sú uvedené v tabuľkovej forme v nasledujúcich kapitolách. Znova je nutné poznamenať, že dátový typ nie je reprezentáciou existujúcich dátových typov konkrétneho databázového systému, ale len popisom reálneho stavu a formy dát pre účely ich prezentácie, a taktiež slúži ako základ pre návrh databázového modelu a štruktúry perzistentných dát. Ako možnosti pre dátový typ boli zvolené **text**, **číslo**, **dátum** a **bool**.

2.2 Proces návrhu databázy

Proces návrhu databázy, ktorý bol s miernymi úpravami zvolený aj pre túto prácu, môže byť podľa Ramakrishnana a Gehrkeho v ich publikácii *Database Management Systems (2nd Ed.)* [29] rozdelený do šiestich nasledovných krokov.

(1) Analýza požiadavkov: úplne prvý krok pri návrhu databáz je porozumenie aké dáta majú byť uložené v databáze, aká aplikácia bude nad nimi vybudovaná (respektíve aká funkcionalita bude od nich požadovaná), a aké operácie sú najčastejšie a budú predmetom požiadaviek na výkon. Inými slovami, musíme zistiť, čo užívatelia z databázy chcú a očakávajú. To zvyčajne zahrňuje neformálny postup, ktorý zahŕňa diskusie s užívateľskými skupinami, štúdiu súčasného operačného prostredia[7] a aké sú jeho očakávané zmeny, analýzu dostupnej dokumentácie k existujúcej aplikácii, ktorá ide byť nahradená a podobne. Na prezentovanie a organizáciu informácii získaných v tomto kroku bolo navrhnutých niekoľko metodológii a bolo vytvorených pár automatizovaných nástrojov na podporu tohto procesu. Veľmi vhodným výstupom z tejto fázy je dátový slovník (popísaný v sekcii 2.3, príklady v kapitolách 5 a 6).

(2) Konceptuálny návrh databázy: Informácie zhromaždené v prvom kroku sú použité na vypracovanie high–level popisu dát,[8] ktoré majú byť uložené v databáze, spolu s obmedzeniami ktoré sú k týmto dátam známe. Tento krok je často zrealizovaný pomocou ER diagramu (popis v sekcii 3.3, príklady v sekciách 5.5.2 a 6.5.2), alebo podobného high–level dátového modelu.

(3) Logický návrh databázy: Nastáva nutnosť vybrať si DBMS[9] na implementáciu databázového návrhu a konvertovať konceptuálny databázový dizajn do schémy databázy v dátovom modeli zvoleného DBMS. V tejto práci pracujeme len s relačnými a objektovými databázami, a tak je v tomto kroku zadanie pre relačné databázy konvertovať ER schému na relačnú databázovú schému. Pri objektových databázach sa často konceptuálny návrh zhoduje s logickým a vytvára sa priamo objektovo–orientovaný model. Výsledkom tohto kroku je logická schéma. Tá je často veľmi potrebná hlavne pre veľmi rozsiahle a komplexné architektúry, nakoľko jedna logická schéma môže zahŕňať viacero konceptuálnych návrhov.

(4) Zdokonalenie schémy: Štvrtý krok v návrhu databázy je analýza kolekcii entít v našej databáze, identifikovanie potencionálnych problémov a ich vylepšenie. V porovnaní s krokmi analýza požiadavkov a konceptuálny návrh, ktoré sú v podstate subjektívne, sa zdokonalenie schémy dá zakotviť a viesť elegantnou a silnou teóriou.

(5) Fyzický návrh databázy: V tomto kroku je nutné zvážiť očakávané zaťaženie, ktoré naša databáza musí podporovať a ďalej rafinovať databázový návrh, aby sme zaistili, že

[7] https://en.wikipedia.org/wiki/Operating_environment

[8] https://tdwi.org/articles/2010/04/07/high-level-data-model.aspx

[9] Database management system – systém spravovania databázy, definovaný ako softwarový systém ktorý umožňuje používateľom definovať, vytvárať, udržiavať a kontrolovať prístup k databáze

sa nám podarí splniť vyžadované kritériá. Tento krok môže jednoducho zahŕňať vytvorenie indexov nad niektorými tabuľkami a atribútmi, alebo môže zahŕňať „clustering“ niektorých tabuliek – agregácia podobných tabuliek do jednej [20]. Môže taktiež ale zahŕňať podstatné predizajnovanie niektorých častí databázovej schémy získanej v predchádzajúcich častiach.

(6) Bezpečnostný dizajn: V tomto kroku sa identifikujú rôzni užívatelia a užívateľské skupiny a nastavuje sa ich prístup k databáze, alebo jej častiam. Toto však pre našu prácu zatiaľ nie je ani požadované, ani navrhnuté, preto sa tomu viac nebudeme venovať.

2.3 Data dictionary

V nasledujúcich kapitolách 5 a 6 budú ku všetkým entitám a atribútom uvedené veľmi potrebné tabuľky aj s dátovým typom spomínaným v predchádzajúcej sekcii 2.1 a štatistikami z podkapitoly 2.5. Spojením týchto tabuliek vzniká technický základ pre vytvorenie dátových modelov a implementáciu databáz – **Data Dictionary**, pojem definovaný spoločnosťou IBM v knihe *IBM Dictionary of Computing* ako „“centralizovaný repozitár informácii o dátach ako napríklad zmysel, vzťah k iným dátam, pôvod, použitie a formát [18]. O dôležitosti vypracovania data dictionary (ďalej len dátový slovník) a jeho úlohe pri správnej transformácii dát na dátové modely píšu medzi inými aj Lavoie a Garner v ich knihe *Preservation Metadata* [22], ale i Navathe a Kerschberg v ich výskumnom článku *Role of data dictionaries in information resource management* [25]. V súlade s týmito zdrojmi je venovaná zvýšená pozornosť príprave, popisu, štruktúre a konceptuálnemu návrhu dát. To však umožnilo efektívnejšiu prácu neskôr, pri návrhu databázových modelov i pri ich implementácii a veľmi možno i predišlo neskoršej nutnosti početných náročných opráv a optimalizácii. Dátový slovník tak v podstate ovplyvňuje všetkých 6 krokov návrhu databázy, čo len znova utvrdzuje myšlienku o jeho dôležitosti. Oproti klasickej forme dátového slovníka [5] sa však v tejto práci neuvádza informácia o povinnosti výskytu atribútov, nakoľko kvôli problému nekompletnosti spomínaného v sekcii 2.4 nemôžeme očakávať ani permanentnú prítomnosť logicky nevyhnutných údajov, a teda môžeme považovať všetky atribúty u oboch sád za voliteľné.

2.4 Problémové faktory pre databázové spracovanie dát

Ešte pred pokračovaním popisom konkrétnych dát je potrebné spomenúť tri často prítomné faktory, ktoré výrazne ovplyvňujú návrh databázových modelov a ku ktorým sa budeme ešte detailnejšie vracať v neskorších kapitolách.

Prvým je **chybovosť**, kvôli ktorej sa nemôžeme spoľahnúť na integritu dát. Jedna entita či atribút môže vďaka ľudskému zlyhaniu pri vytváraní alebo digitalizácii záznamov, alebo jednoducho kvôli rozdielnym štýlom zápisov v rozdielnych obdobiach, či dokonca kvôli zmene používaného jazyka, vystupovať pod dvoma rôznymi názvami.

Druhým je **nekompletnosť**, kvôli ktorej nemôžeme očakávať rovnaký počet a štruktúru zaznamenaných atribútov pri rovnakých entitách a je nevyhnutné brať do úvahy často i vysokú mieru ich absencie.

Tretím je **rozsiahlosť**, či rôznorodosť atribútov jednotlivých entít, nakoľko boli v priebehu rokov a v rámci rôznych inštitúcii, či dokonca krajín, zaznamenávané rôzne informácie viažuce sa k tej istej entite a nebol pevne stanovený štandard pre formu zápisov.

V tomto projekte je však kladený veľký dôraz na zachovanie čo najväčšieho množstva, ideálne všetkých, hoc i vadných informácii a táto požiadavka má prednosť pred možnými vylepšeniami, či uľahčeniami implementačnej časti za cenu straty dát.

2.5 Štatistiky vstupných dát

Vzhľadom na nedostatočný počet zatiaľ digitalizovaných a spracovaných dát nebola k tejto práci poskytnutá celá sada údajov ani v jednom prípade, ale len jej reprezentatívna vzorka. Navrhnuté a zvolené riešenie bude ale využívané nad celou sadou. Z tohoto dôvodu, ale aj kvôli porovnaniu a optimalizácii dátových modelov je ale potrebné testovanie na rozsiahlom množstve dát, z čoho vychádza potreba vytvoriť generátor poskytujúci tieto dáta a nahradzujúci pôvodnú sadu. Aby sme dosiahli čo najpresnejšie a najlepšie výsledky porovnaní, musí výstup generátora čo najviac konvergovať k skutočným dátam.

Pre návrh takéhoto generátora, ale aj pre vytvorenie ideálneho dátového modelu a efektívne a rýchle využívanie hardvérových zdrojov v následnej implementácii je potrebná detailná analýza reprezentatívnej vzorky [23]. Vzhľadom na to, že spracovávané dáta nie sú relatívne komplikované, analýzu nahradíme uvedením starostlivo nami vybraných a vytvorených štatistík, a to konkrétne *početnosti výskytu atribútu*, uvedenej v percentách, *priemernej rozsiahlosti v znakoch* – v tabuľkách **avg** – a *maximálnej rozsiahlosti* – v tabuľkách **max** – taktiež uvedenej v znakoch. Prvé dva údaje sú zaokrúhlené na dve desatinné miesta. Obe štatistiky sú uvedené v nasledujúcich podkapitolách popisujúcich štruktúru dát a entít.

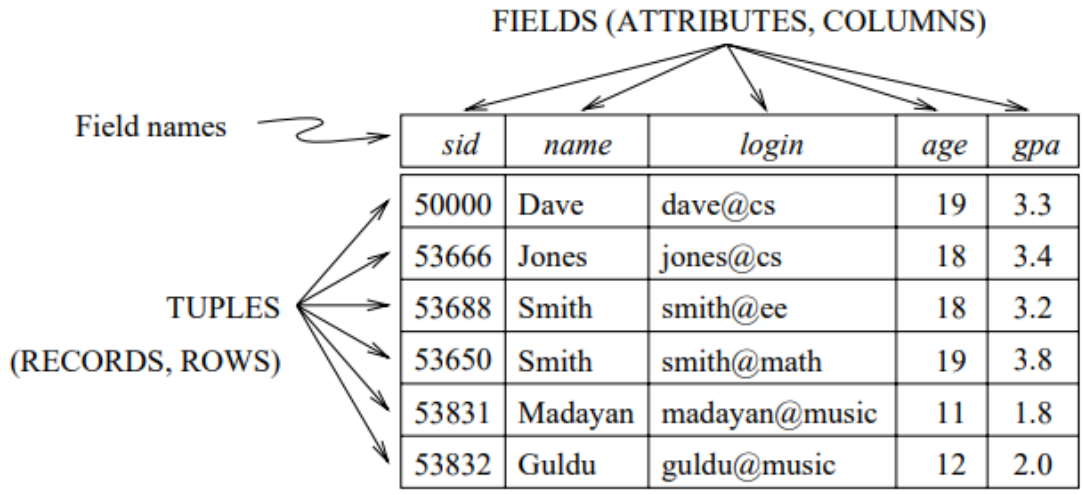
V prípade agregácie viacerých tabuliek do spoločnej je početnosť a priemerná rozsiahlosť spriemerovaná. V maximálnej rozsiahlosti je uvedená maximálna hodnota rovnakého atribútu skrz všetky osoby. V prípade úplnej absencie informácii je ich početnosť odvodená od počtu záznamov a štruktúra, a teda aj štatistiky, od podobných atribútov v iných entitách. Problematické a špecifické prípady sú bližšie popísané pri danej tabuľke.

Kapitola 3

Relačné databázové systémy

3.1 Prehľad relačného modelu

Relačná databáza je digitálna databáza založená na relačnom databázovom modeli. Relačný model reprezentuje databázu ako množina relácii. Základnou konštrukciou na reprezentáciu dát v relačnom modeli je relácia. Relácia pozostáva z relačnej schémy a z relačnej inštancie. Relačná inštancia je tabuľka, relačná schéma popisuje hlavičky (názvy) stĺpcov v tabuľke. Schéma špecifikuje meno relácie, meno každého poľa (resp. stĺpca alebo atribútu) a doménu každého poľa. Inštancia relácie je množina n–tíc, inak nazývaných záznamy, pričom každá n–tica má rovnaký počet polí ako relačná schéma. Relačnú inštanciu si môžeme predstaviť ako tabuľku v ktorej každá n–tica je jeden riadok a všetky riadky majú rovnaký počet polí. Príklad relačnej inštancie môžeme vidieť v obrázku 3.1.



sid	*name*	*login*	*age*	*gpa*
50000	Dave	dave@cs	19	3.3
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

Obr. 3.1: Inštancia S1 študentskej relácie – Students(sid: string, name: string, login: string, age: integer, gpa: real). [29]

3.1.1 Charakteristiky relácií

Čo sa týka charakteristík relácií, existuje množstvo podobných definícii a množín pravidiel, samotný autor relačného modelu Edgar Codd definoval 12 základných princípov [12], avšak

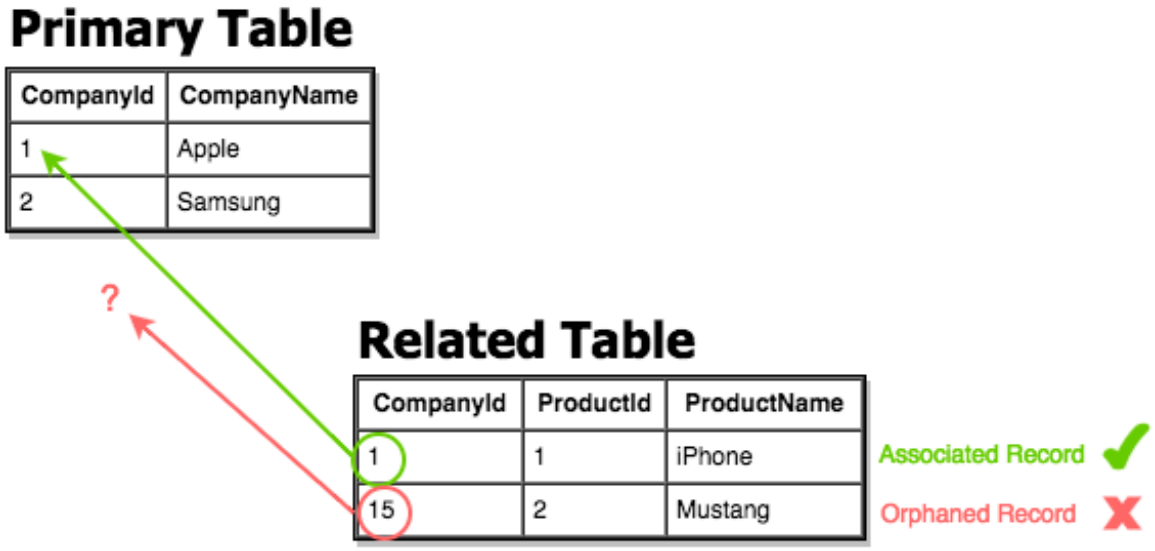
neexistuje jednotná a všeobecne uznávaná forma či sada s daným počtom presne špecifikovaných a definitívnych charakteristík, a preto sme sa rozhodli pre účely tejto práce spísať vlastné. Pri definovaní týchto charakteristík sme vychádzali z akademickej znalosti skúmaného problému, praktických skúseností, pôvodného zámeru autora (už spomínaných 12 pravidiel, ale napríklad aj *Is Your DBMS Really Relational?* [11], alebo *Does Your DBMS Run By the Rules* [10]), ale aj spracovaní a názorov odbornej verejnosti (napr. *Introduction to Relational Databases – Part 1: Theoretical Foundation* [32]). Výsledkom je 10 nasledujúcich pravidiel, ktoré považujeme za jasnú, vhodne štruktúrovanú, všetko zahrňujúcu a definitívnu verziu, ktorej sa budeme držať.

1. Tabuľka (relácia) je dvoj–dimenzionálna štruktúra zložená z riadkov a stĺpcov.
2. Každá tabuľka musí dodržiavať integritné obmedzenia udržiavajúce dátovú konzistenciu naprieč všetkými tabuľkami.
3. Každá tabuľka musí mať stĺpec (atribút), alebo ich kombináciu, ktorá jednoznačne identifikuje každý riadok.
4. Poradie stĺpcov a riadkov je irelevantné.
5. Každý riadok (n–tica) predstavuje jediný výskyt tohto záznamu v sade danej entity.
6. Každý atribút má unikátny názov.
7. Každý stĺpec musí byť dátovo homogénny – všetky hodnoty v stĺpci musia spadať pod rovnaký dátový typ.
8. Každý stĺpec má daný rozsah prijateľných hodnôt známy ako *doména atribútu*.
9. Každý priesek stĺpca a riadku reprezentuje jedinú dátovú hodnotu.
10. Každá hodnota je atomická.

3.1.2 Referenčná integrita

Referenčná integrita, spomínaná v pravidle 2, je základný a nevyhnutný ale netriviálny databázový koncept udržujúci databázu v stabilnom a použiteľnom stave. Je definovaná medzi dvoma reláciami a zabezpečuje konzistenciu medzi n–ticami týchto dvoch relácii. Je implementovaná pomocou *cudzieho kľúča* FK, ktorý je špeciálny atribút odkazujúcej entity slúžiaci ako referencia (alebo ukazovateľ) na primárny kľúč odkazovanej entity. Aby sa referenčná integrita dodržala, cudzí kľúč príbuznej n–tice musí ukazovať na existujúcu n-ticu v primárnej tabuľke a nesmú vznikať takzvané „osirelé záznamy“, ako je demonštrované v obrázku 3.2. Množina atribútov FK v relácii R1 je cudzím kľúčom R1, ak spĺňa nasledujúce dve pravidlá [1]:

- Atribúty FK majú tú istú doménu ako atribúty primárneho kľúča PK v relačnej schéme R2; atribúty FK sú referenciou alebo referujú do relácie R2.
- Hodnota FK v n–tici t1 relačnej schémy R1 sa vyskytuje ako hodnota PK pre nejakú n–ticu t2 v R2, alebo má hodnotu null.



Obr. 3.2: Tabuľka zobrazuje vzťah odkazujúcej (Related) a odkazovanej (Primary) tabuľky, pričom prvý záznam (Associated Record) spĺňa a druhý (Orphaned Record) nespĺňa referenčnú identitu a ostáva teda osirelým záznamom.[2]

3.1.3 Pravá implementácia relačného modelu

Ako bolo v predchádzajúcej sekcii popísané, od jeho teoretického založenia sa relačný model stal základom pre väčšinu databázového softwaru. Avšak je až ironické, že hlavne kvôli historickým okolnostiam jeho pravá implementácia doteraz na komerčnom trhu neuspela. Databázy, ktoré dnes označujeme za relačné, sa síce zakladajú na tejto teórii, takmer žiadna z nich však z rôznych dôvodov nedodržuje teóriu relačného modelu absolútne a väčšina taktiež vedome porušuje spomínaných 12 Coddových pravidiel, ktoré sa dnes už považujú za zastarané. Uznávaný odborník David McGoveran, ktorý sa považuje za priekopníka teórie relačných databáz, o tom v roku 2017 povedal:

> *Relačný dátový model (RDM) odkazuje **len** na dátový model ktorý vzišiel z práce E. F. Codda et. al. Nanešťastie bola relačná terminológia už dávno vykradnutá a je tak často zneužitá a všeobecne zle používaná, že je dnes už nemožné prekonať zmätené myslenie, nesprávne vzdelávanie a pomýlenú komunikáciu, ktorú toto spôsobilo. **„Údajné relačné“** by mal byť pojem na identifikáciu čohokoľvek vedúceho návrh a vývoj databáz predstierajúcich byť relačnými, vrátane SQL a ostatných komerčných DBMS, a relačný „straw man“[3] napadnutý neinformovanými (e.g., zástancovia NoSQL). Veľa z kritiky takzvaných „komerčných relačných systémov“ je validnej, ale nemá nič spoločné s RDM. Vzťahuje sa práve na „údajné relačné“ implementácie, navrhnuté tými, ktorí si požičali niečo z RDM bez toho, aby mu vôbec niekedy rozumeli a tak nie sú schopní využiť jeho výhody na riešenie aplikačných problémov.*

V tejto práci sa tomuto problému nevenujeme do hĺbky, ale tento výrok a aj ďalšie rozvedenie tejto myšlienky je rozsiahlejšie skúmané v článku *What Is a True Relational System* [27],

[2] ZDROJ: https://database.guide/what-is-referential-integrity/
[3] forma chybného argumentu a informačného omylu – https://en.wikipedia.org/wiki/Straw_man

poprípade v jeho zdrojoch. Za zmienku stojí, že vzniklo niekoľko pokusov vyprodukovať pravú implementácie relačného databázového modelu tak, ako bol originálne definovaný a jeden z najčerstvejších je Rel.

3.1.4 Dopytovací jazyk

Dopytovací, alebo vyhľadávací jazyk je počítačový jazyk určený na ovládanie databáz a informačných systémov pomocou príkazov – dopytov. Takmer všetky relačné databázy používajú v súčasnosti najpoužívanejší dopytovací jazyk Structured Query Language (SQL, štruktúrovaný dopytovací jazyk) a tak sa často označujú aj ako SQL databázy. V súvislosti s predchádzajúcou sekciou môžme tvrdiť, že SQL implementuje technickú aproximáciu relačného modelu. Tabuľka v SQL korešponduje s predikátovou premennou, obsah tabuľky s reláciou, obmedzenia kľúčov, iné obmedzenia a SQL dopyty zas s predikátmi. Medzi hlavné odchýlenia od relačného modelu patria:

- **Duplikátne riadky** – rovnaký riadok sa v SQL tabuľke môže objaviť viac než raz.
- **Anonymné stĺpce** – stĺpec v SQL tabuľke môže byť nepomenovaný.
- **Duplikátne názvy stĺpcov** – dva alebo viaceré stĺpce môžu mať v SQL tabuľke rovnaký názov.
- **Dôležitosť poradia stĺpcov** – poradie stĺpcov v SQL tabuľke je definované a podstatné.
- **NULL** – Táto špeciálna značka sa môže objaviť namiesto hodnoty všade tam, kde mohla byť hodnota.

3.2 Porovnanie relačných a nerelačných databáz

Relačné databázy a ich výhody a nedostatky sme porovnávali so zameraním na ne samé, teda všeobecne oproti dnešným NoSQL databázam, ako aj ich predchodcom.

3.2.1 Výhody relačných databáz

Medzi hlavné výhody databázového návrhu podľa relačného modelu patria hlavne podľa článkov *Prehľad a porovnanie relačných a nerelačných databáz* [19] a *Aké sú výhody relačného databázového modelu* [31]:

- Väčšina informácii je uložených v databáze a nie v aplikácii, takže je databáza samo–dokumentujúca.
- Je jednoduché pridať, aktualizovať, zmazať alebo vo všeobecnosti pristúpiť k dátam. Nevyžaduje navigáciu rigidnou cestou skrze strom alebo hierarchiu.
- Poskytuje výhody sumarizácie, získania a nahlasovania údajov.
- Databáza je štruktúrovaná v tabuľkovej forme s vysoko príbuznými tabuľkami. Charakter databázy je predpovedateľný.

- Je možné a pomerne jednoduché urobiť potrebné zmeny v štruktúre databázy. Relačný model je prirodzene škálovateľný a rozšíriteľný a poskytuje flexibilné štruktúry. Zmeny môžu byť implementované bez dopadu na už existujúce dáta, alebo zvyšok databázy. Teoreticky neexistuje limit na počet riadkov, stĺpcov a tabuliek.
- Tabuľová štruktúra sa vyhýba komplexnosti a je intuitívna a prirodzená v organizácii.
- Pevná údržba integrity dát. Silná typovosť a kontroly validity zaručujú, že sa dáta pohybujú v prijateľnom rozsahu a vyžadované dáta sú prístupné. Takto sa udržuje presnosť a konzistentnosť dát.
- Existuje systematická metodológia na odhalenie a elimináciu anomálii, ktoré môžu ovplyvniť integritu a presnosť databázy – *normalizácia.*[4] Poskytuje nám súbor pravidiel, vlastností a cieľov pre dizajn a posudzovanie databázovej štruktúry.

3.2.2 Nedostatky relačných databáz

Relačné databázy sú síce konvenčne akceptované, trpia však aj množstvom nedostatkov, medzi ktoré patria hlavne:

- Relačné databázy nepodporujú vysokú škálovateľnosť. Do určitého bodu môžeme zlepšovať hardware, avšak neskôr už databáza musí byť distribuovaná.
- Jednou z hlavných nevýhod štruktúry tabuľkového systému relačných databáz je, že v prípade, že sa vstupné dáta nedajú jednoducho zapuzdriť v tabuľke táto štruktúra môže vzrásť do vysokej (a často zbytočnej) komplexnosti.
- Mnoho z funkcií poskytovaných relačnými databázami, je často veľmi vzácne, alebo takmer vôbec nevyužívaných, a tak len jednoducho pridávajú k cene a komplexnosti databázy.
- Relačné databázy využívajú exkluzívne SQL, ktorého funkcia je pracovať so štruktúrovanými dátami, ale SQL môže byť vysoko komplexné pri práci s neštruktúrovanými dátami.
- V prípade veľkého počtu dát musí byť databáza rozdelená na niekoľko serverov a to spôsobuje niekoľko problémov, pretože spájať tabuľky na distribuovaných serveroch je náročné a komplexné zadanie.

3.2.3 Porovnanie

Na záver môžeme konštatovať a vyzdvihnúť hlavné rozdiely medzi týmito dvoma kategóriami databáz v tabuľke 3.1

3.3 Entity–relationship model

Entity–relationship dátový model (entitno–vzťahový model, ďalej len ERM) umožňuje popísať dáta z reálneho sveta vo forme objektov a ich vzťahov. Vznikol článkom *The Entity-Relationship Model - Toward a Unified View of Data* [8] ako všeobecný model popisujúci

[4] https://en.wikipedia.org/wiki/Database_normalization

Tabuľka 3.1: Porovnanie hlavných rozdielov medzi relačnými a nerelačnými databázami.

N.	Nerelačné	Relačné
1.	Vysoká dátová priepustnosť	Nízka dátová priepustnosť
2.	Vysoko škálovateľné	Menej škálovateľné
3.	Dáta môžu byť vložené kedykoľvek bez definovania schémy. Dáta môžu byť zmenené kedykoľvek bez väčších problémov – vysoká flexibilita.	V tomto prípade musia dáta napasovať do preddefinovanej tabuľky alebo štruktúry.
4.	Výkon môže byť vylepšený cachovaním dát do systémovej pamäte.	Caching môže byť využitý s pomocou špeciálnej infraštruktúry.
5.	V aplikácii môžu byť implementované viaceré obsluhy transakcii.	Transakčné spracovanie.
6.	Jedno-atribútový index, kľúč–hodnota sklad	Index dostupný na viacerých stĺpcoch.
7.	Poskytuje BASE (Basically Available, Soft state, Eventual consistency) vlastnosti.[5]	Poskytuje ACID (Atomicity, Consistency, Isolation, Durability) vlastnosti.[6]
8.	Nerelačné databázy robia voči konzistencii kompromisy.	Poskytujú lepšiu konzistenciu ako nerelačné databázy.
9.	Je povolená duplikácia dát, čo ohrozuje dátovú integritu. Navyše aktualizácia jedného záznamu znamená aktualizáciu všetkých duplikátov, čo je vysoko neefektívne a znamená veľké zvýšenie réžie.	Návrhový proces relačných databáz eliminuje duplikáciu záznamov, čo slúži ako prevencia pred okupovaním databázy nekonzistentnými dátami.
10.	Vyhľadávanie v nerelačných databázach je neefektívne, špeciálne v prípade vyhľadávania podľa viacerých kritérií, pretože to vyžaduje viac ako jeden priechod.	Organizácia relačných databáz umožňuje dopytovaciemu jazyku, ako napríklad SQL, používať primárny kľúč zdieľaný medzi tabuľkami na rýchle a efektívne zoradenie a návrat vyžiadaných záznamov.

reálne systémy do technickej schémy. Poskytuje užitočné koncepty, ktoré umožňujú transformovať neformálne, alebo polo–formálne popisy databázy do detailného a precízneho popisu, ktorý môže byť implementovaný v DBMS. V poli vytvárania konceptuálnych schém relačných databáz dosiahol tak veľký úspech, že takmer úplne nahradil klasickú reprezentáciu relačného modelu a je taktiež často nesprávne zamieňaný s relačným modelom. Relačný model je prístup k manažovaniu dát používajúci štruktúru a jazyk konzistentný s *predi-*

[5] https://en.wikipedia.org/wiki/Eventual_consistency
[6] https://en.wikipedia.org/wiki/ACID_(computer_science)

kátovou logikou prvého rádu,[7] zatiaľ čo ER model je konceptuálne znázornenie dát, ktoré je v princípoch ekvivalentné relačnému modelu. Vďaka ekvivalencii nie je potrebné v tejto práci uvádzať oba diagramy, vďaka výhodám ERM ako napr. informácie o kardinalite uvádzame práve ten v obrázkoch 5.4 a 6.4. V rámci užšieho kontextu procesu návrhu databázy je ER najrelevantnejší hlavne pre prvé tri body, vytváraný vo fáze konceptuálneho návrhu, a v rámci širšieho kontextu relevantný pre celý proces.

Je nutné poznamenať, že existuje mnoho zaužívaných variácii ERD a neexistuje jedna široko akceptovaná forma považovaná za štandard. Prezentácia v tejto kapitole je reprezentáciou rodiny ER modelov a zahrňuje výber najpopulárnejších funkcii, ako základ je však použitá takzvaná „crow's foot" notácia vyvinutá Gordonom Everestom [15].

ER model je niekedy považovaný za kompletný prístup k projektovaniu logickej databázovej schémy. To je však nesprávne, pretože ER diagram (ďalej len ERD) je len približným popisom dát, skonštruovaný prostredníctvom subjektívneho zhodnotenia informácii, získaných počas analýzy požiadavkov, v lepšom prípade už čiastočne spracovaných v dátovom slovníku. Detailnejšia analýza môže často vylepšiť logickú schému získanú z kroku 3. Po skonštruovaní kvalitnej logickej schémy je stále ešte potrebné zohľadniť výkonnostné kritériá a dizajn fyzickej schémy, musíme adresovať bezpečnostné problémy. To všetko je popísané v posledných troch bodoch.

3.3.1 Transformácia ER modelu na relačný model

Vzhľadom na to, ako sú si tieto modely blízke, je aj transformácia relatívne jednoduchá a priamočiara, často sú prvky takmer rovnaké, mení sa len terminológia.

- Entitné množiny a ich atribúty sú mapované na relácie/tabuľky a ich atribúty.
- Vzťahové množiny s atribútmi sú mapované na relácie.
- Vzťahové množiny so vzťahom Many–to–Many[8] sú taktiež mapované na relácie. Vytvára sa *Asociatívna tabuľka*,[9] v praxi často nazývaná aj „join table". Tá obsahuje N cudzích kľúčov, ktorých kombinácia je unikátna, často fungujú ako kompozitný primárny kľúč a odkazujú na primárne kľúče ostatných entít vo vzťahu. Môže obsahovať svoje atribúty a môže mať aj iný, vlastný primárny kľúč.
- Jednoduché vzťahové množiny so vzťahom Many–to–One (N–k–1) sú mapované ako cudzí kľúč vložený do relácie reprezentujúcej násobnú entitnú množinu vo vzťahu. Často sa nazýva taktiež agregačný vzťah.
- Jednoduché vzťahové množiny so vzťahom One–to–One (1–k–1) sú mapované ako cudzí kľúč bez preferencie relácie.
- Slabé entitné množiny sú špeciálnym prípadom Many–to–One vzťahu, kde je násobná entitná množina podradená a nemôže existovať bez nadradenej. Cudzí kľúč k nadradenej relácii sa stáva primárnym kľúčom. Často sa nazýva taktiež kompozitný vzťah.
- Dedičnosť, alebo generalizácia, je špeciálna konštrukcia, kde viacero entitných množín zdieľa spoločné atribúty. Martin Fowler tento problém rozsiahlo popisuje v jeho knihe *Patterns of Enterprise Application Architecture book* [16] a v súlade s jeho zisteniami môžeme tvrdiť, že máme tri možnosti ako túto štruktúru transformovať:

[7] https://en.wikipedia.org/wiki/First-order_logic
[8] „mnoho–k–mnoho", entite A môže pripadať N entít B, ale aj entite B môže pripadať N entít A.
[9] https://en.wikipedia.org/wiki/Associative_entity

1. **Table-Per-Type (TPT – tabuľka pre typ)**, alebo aj „supertype–subtype“. Každá entitná množina má svoju tabuľku. Základná tabuľka obsahuje všetky zdieľané atribúty a každá z nej odvodená má taktiež svoju tabuľku s primárnym kľúčom, ktorý je zároveň aj cudzím kľúčom k základnej. Odvodená tabuľka obsahuje len rozdielne atribúty.
2. **Table-Per-Hierarchy (TPH – tabuľka pre hierarchiu)** Existuje jediná tabuľka pre celú hierarchiu dedičnosti, čo znamená, že niektoré stĺpce budú pravdepodobne riedke. Je pridaný diskriminačný stĺpec, ktorý v systéme určuje, aký typ entity to je.
3. **Table-Per-Concrete (TPC – tabuľka pre entitu)** Každá entita má vlastnú plne sformovanú tabuľku bez akýchkoľvek referencií na iné tabuľky.

Neexistuje však univerzálne pravidlo pre správnosť využívania týchto možností. Ich uplatnenie je silno špecifické vzhľadom na konkrétnu situáciu a v určitých prípadoch sa dajú dokonca kombinovať.

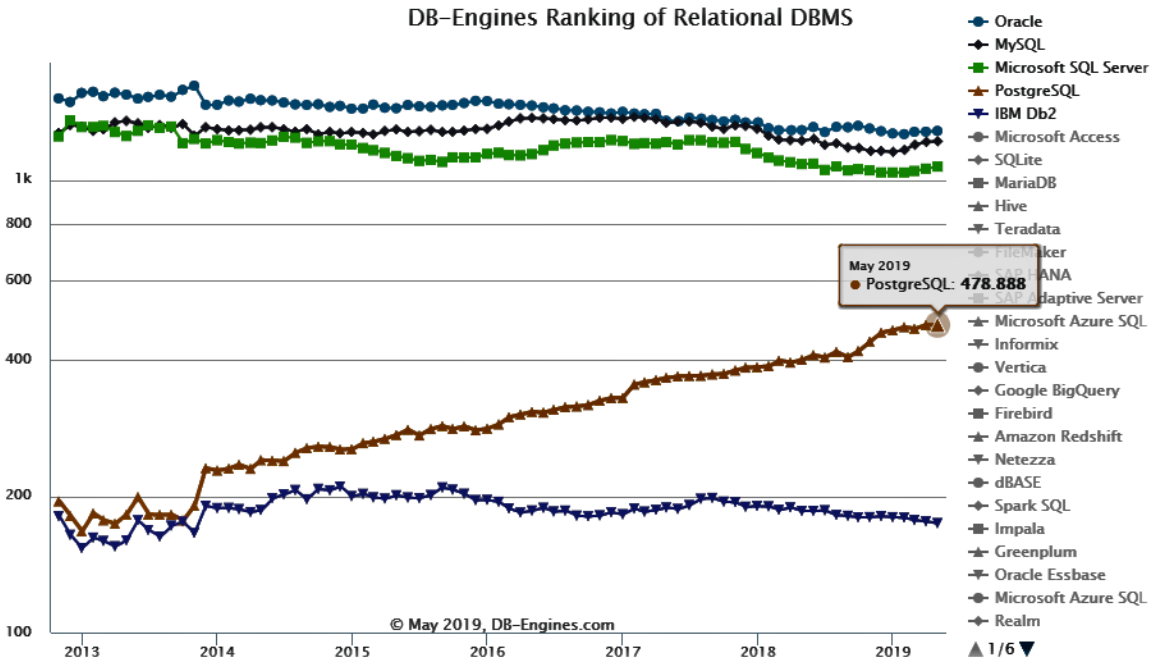
3.4 PostgreSql

PostgreSQL je voľne šíriteľný RDBMS, uvoľnený pod flexibilnou licenciou BSD, zdôrazňujúci rozšíriteľnosť a zhodu s normami. Ponúka alternatívu k ostatným voľne šíriteľným databázovým systémom (napr. MySQL, Firebird atď.), ako aj k proprietárnym (napr. Oracle, Microsoft SQL Server atď.). Zvláda rozsah záťaže od jednoúčelových aplikácii až k webovým službám, alebo dátovým skladom s množstvom súbežných používateľov. Je štandardnou databázou pre macOS Server a je tiež dostupný pre Linux, FreeBSD, OpenBSD a Windows. PostgreSQL spĺňa ACID vlastnosti a využíva transakčné spracovanie. Ponúka podporu RDBMS funkcií ako aktualizovateľné a materializované pohľady, trigger, cudzie kľúče, funkcie a uložené procedúry. Je vyvíjaný *PostgreSQL Global Development Group*, rozmanitou skupinou viacerých spoločností a individuálnych prispievateľov. Podľa mnohých databázových odborníkov je v súčasnosti PostgreSQL najvyspelejší a najsofistikovanejší voľne šíriteľný DBMS. Pre našu implementáciu sme zvolili najnovšiu stabilnú major verziu 11.

3.4.1 Proces výberu

Výber databázového systému nebol zadaním bližšie popísaný a tak boli ako hlavné kritériá zvolené dostupnosť, využívanosť a popularita, reprezentatívnosť, efektívnosť, rýchlosť, bezpečnost a podobne, ale aj osobná skúsenosť. Čo sa týka popularity, 5 najpopulárnejších RDBMS je podľa rankingu databáz iniciatívou DB-Engines [26] v tomto poradí: Oracle, MySQL, Microsoft SQL Server, PostgreSQL a IBM Db2. Z týchto piatich databáz je za posledný rok (obdobie apríl 2018 – apríl 2019) PostgreSQL jedinou rastúcou. Za rovnaké obdobie jej rast je taktiež väčší (+83.25) ako kombinované klesanie ostatných piatich (−69.55). Z grafu 3.3 taktiež môžeme vidieť rovnaký trend, teda silný a stabilný rast PostgreSQL a stagnovanie či regresiu ostatných systémov. Čo sa týka dostupnosti, Oracle, Microsoft SQL Server aj IBM Db2 sú komerčné databázy a to ich pre našu prácu vylučuje. PostgreSQL bol taktiež v roku 2018 už druhýkrát za sebou vyhlásený za DBMS roka. [26] Čo sa týka ďalšieho porovnania PostgreSQl a MySQL, obe reprezentujú myšlienku relačného modelu podobne a dosahujú podobné výsledky, čo sa týka efektívnosti a rýchlosti. I po detailnej analýze dostupných materiálov a výskumov na túto tému, okrem mnohých iných napr. často

spomínaného článku *Showdown: MySQL 8 vs PostgreSQL 10* [14], alebo veľmi precízne vypracovaného *PostgreSQL vs. MySQL: [2019] Everything You Need to Know* [3] a ďalších, nie je možné urobiť presvedčivý a definitívny záver a označiť jeden z týchto systémov za jednoznačne lepší. Na základe štruktúry a charakteru vstupných dát k tejto práci a ich využitia však vieme povedať, že je očakávaný častejší prístup a extrakcia dát (SELECT), ako ich manipulácia (INSERT, UPDATE, DELETE) a to hovorí mierne v prospech PostgreSQL. Na záver sme zhodnotili aj osobnú skúsenosť, ktorá síce nie je najkľúčovejším hodnotiacim prvkom, ale pri veľkej vyrovnanosti má určitú váhu a hovorí taktiež v prospech PostrgeSQL, vďaka čomu sme v kombinácii s predchádzajúcimi informáciami rozhodli práve pre tento RDBSM.



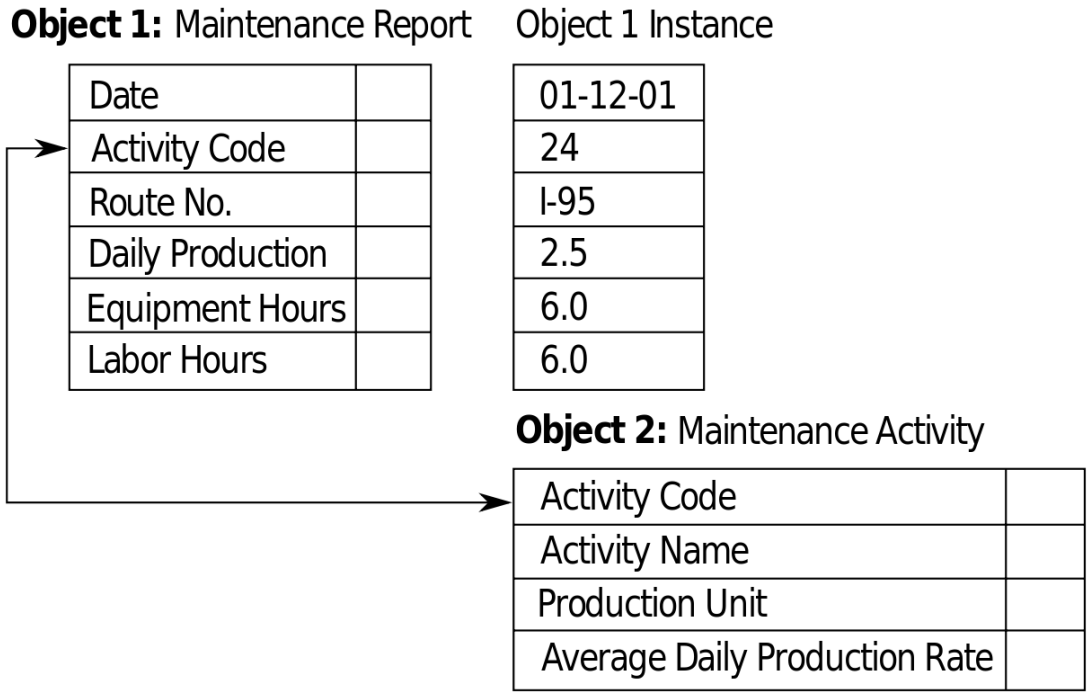
Obr. 3.3: DB-Engines rebríček popularity RDBMS. Horizontálna os reprezentuje skóre (logaritmická stupnica). [26]

Kapitola 4

Objektové databázy

4.1 Prehľad objektovo orientovaného modelu

Objektovo orientovaná databáza je kolekcia objektov, ktorých správanie, stav a vzťahy sú definované podľa objektovo orientovaného dátového modelu. Objektovo orientovaný (objektový) model dát v databázových systémoch vychádza zo známych princípov objektovo orientovaného programovania. Je však ďalej obohatený o techniky perzistencie, reprezentácie vzťahov, dopytovania, transakčného prístupu a pod. Jednoduchý príklad objektového modelu môžeme vidieť na obrázku 4.1.



Obr. 4.1: Príklad objektovo orientovaného modelu. [4]

V objektových databázach sú informácie reprezentované vo forme objektov, rovnako ako v objektovo orientovanom programovaní. Taktiež majú rozhranie pre objektovo orientované jazyky a ukladajú transparentne celé hierarchie objektov. Objektovo orientovaný databá-

zový systém (Object Database Management System, ODBMS) je databázový systém, ktorý umožňuje definíciu a manipuláciu objektovo orientovanej databázy.

Pre objektové databázy neexistuje žiadny oficiálny štandard. Štandardom je de facto kniha *The Object Database Standard: ODMG* [7]. Dôraz ODBMS je na priamu korešpondenciu medzi nasledujúcimi:

- objekty a objektové vzťahy v aplikácii napísanej v OO jazykoch
- ich uchovávanie v databáze

Schéma DB a dopytovací jazyk musí podporovať všetky vlastnosti nutné k práci s objektami, teda plnú podporu objektov, triedy, zapuzdrenie, dedičnosť, polymorfizmus, jednoznačnú identifikáciu objektov a referencie medzi objektami.

Konkrétne koncové implementácie ODBMS sa môžu od seba zásadne líšiť podľa dopytovacieho jazyka, integráciou dopytu a navigačných rozhraní.

4.1.1 Stratégia perzistencie

Je zjavné, že ukladať všetky objekty vyskytujúce sa v ODBMS do perzistentného úložiska by bolo neefektívne. Po dobu behu systému vždy existuje rada tranzientných objektov, ktoré sú potrebné len pre dočasné výpočty a nie je nutné ich ukladať. Vyvstáva teda otázka, ako systém pozná, čo je a čo nie je potrebné ukladať? Existuje niekoľko prístupov, ktoré sú zvyčajne používané v ODBMS [33]:

- *Perzistentné triedy.* V okamžiku, keď sa definuje trieda objektov, je súčasne špecifikované, či má alebo nemá byť perzistentná. Ak je trieda označená ako perzistentná, budú mať všetky objekty od nej inštanciované charakter perzistentných dát.
- *Perzistentné tieňové triedy (shadow classes).* Pri vytváraní triedy objektov sa automaticky vytvorí jedna perzistentná a jedna tranzientná trieda. Ak je potom objekt inštanciovaný od perzistentnej verzie, je perzistentný, ak je inštanciovaný od tranzientnej verzie, je považovaný za tranzientný.
- *Perzistentná koreňová trieda.* V systéme existuje jedna preddefinovaná perzistentná trieda a vlastnosť perzistencie sa prenáša na ďalšie triedy prostredníctvom dedičnosti.
- *Perzistencia špecifikovaná pri vytváraní objektov.* V tomto prípade se nedeklaruje perzistencia na úrovni tried, ale na úrovni jednotlivých objektov. Pri vytváraní objektu je objekt vyhlásený buďto za perzistentný alebo za tranzientný (napr. v operátore new).
- *Explicitné ukladanie.* Tento mechanizmus, blízky tradičným aplikáciám pracujúcim nad súbormi, umožňuje explicitné uloženie objektov jednou z jeho metód. Ukladanie objektov je teda riadené programátorom, ktorý databázovú aplikáciu vytvára.
- *Perzistentné korene (perzistent roots) preddefinované systémom.* Táto veľmi populárna schéma perzistencie je založená na myšlienke perzistencie podľa dosiahnuteľnosti (persistence by reachability). V systéme existujú preddefinované objekty (persistent roots), zvyčajne charakteru kontajneru. Potom všetky objekty,na ktoré existuje odkaz (i nepriamy) z niektorého perzistentného koreňa, sú považované za perzistentné.

- *Pomenované objekty ako perzistentné korene.* Vzniká rozšírením predchádzajúceho modelu o voliteľnosť perzistentných koreňov. Za perzistentné korene sú tentokrát považované všetky takzvané pomenované objekty. Pomenovaním sa rozumie priradenie mena konkrétnemu objektu na úrovni schémy databáze. Rovnako ako v predchádzajúcom prípade sa potom za perzistentné považujú všetky objekty dosiahnuteľné z niektorého perzistentného koreňa.

4.1.2 Odstraňovanie dát

Vzhľadom k limitovanej kapacite perzistentného úložiska je ďalším dôležitým úkolom ODBMS efektívne odstraňovanie nepotrebných objektov. Klasickým prístupom je technika explicitného zrušenia objektu programátorom. Nespornou výhodou tejto metódy je explicitná identifikácia rušených objektov. Nevýhodou je však prenesenie zodpovednosti na stranu programátora a jednoduchá možnosť straty konzistencie. Konkrétne, ak je rušený objekt viazaný určitými vzťahmi na iné objekty, je potrebné rozhodnúť, či ak sa tieto vzťahy zrušia, majú sa odstrániť i ďalšie objekty viazané týmito vzťahmi, alebo je odstránenie daného objektu z hľadiska sémantiky dátového modelu neprípustné.

Druhou variantou odstraňovania nepotrebných objektov je technika garbage collectingu. V tomto prípade ODBMS sám automaticky identifikuje nepotrebné objekty a ruší ich. Vzniká tu však dôležitý problém, ako identifikovať nepotrebné objekty:

- Počítadlo referencií (reference counting). Každý objekt si pamätá počet odkazov sám na seba. Ak klesne na nulu, objekt sa stáva nedostupným a teda ďalej nepotrebným. Tato metóda je síce jednoduchá, ale zlyháva v prípade, že sa vytvoria izolované slučky objektov, ktoré sú síce nedostupné, ale vzájomne na seba odkazujú a teda ich počítadlo referencií nikdy neklesne na nulu. V praxi je teda nutné túto metódu doplniť o ďalšie techniky rozpoznávania izolovaných slučiek.

- Technika mark-and-sweep. Funguje tak, že systém identifikuje objekty dosiahnuteľné z perzistentných koreňov. Všetky ostatné objekty sú potom vyhlásené za nedostupné a označené k zrušeniu.

4.1.3 Dopytovací jazyk

Na prácu s databázou v teórii rozlišujeme tri rôzne základné druhy jazykov, konkrétne **DDL** (Data definition language, slúži k definícii logickej schémy databázy), **DML** (Data manipulation language, slúži na manipuláciu dát v databáze) a **DQL** (data query language, slúži na dopytovanie do databázy). V relačných databázach všetky tieto úlohy spĺňal jazyk SQL a vďaka tomu sa pojem dopytovací jazyk začal používať ako označenie pre databázový jazyk vo všeobecnosti.

Oproti relačnému modelu však implementácia dotazovacieho jazyka v objektových databázach naráža na niekoľko problémov. Prvým problémom je komplikovaná dátová štruktúra objektových databáz, založená na množstve objektov zviazaných radou vzťahov. Druhý typický problém vyplýva z filozofie objektovo orientovaného modelovania – základným princípom objektového návrhu je maximálne zapuzdrenie a ukrývanie dátových položiek v objektoch. Otázkou teda je, akým spôsobom efektívne sprístupniť dáta dopytovaciemu procesoru. Výsledkom týchto úvah je, že v ODBMS je vhodné dopytovací jazyk obmedziť len na vyhľadávanie a výber dát. V rámci štandardu ODMG bol navrhnutý dopytovací jazyk OQL – jedná sa o čisto dopytovací jazyk. Pre definíciu logickej schémy objektovo orientovanej

databázovej aplikácie bol navrhnutý jazyk ODL. Čo sa týka manipulácie dát, vzhľadom na zložitosť objektového modelu a heterogénnosť dát v databáze (dáta rozptýlené v objektoch mnohých tried, množstvo obojsmerných vzťahov a pod.), volia objektové databázy na manipuláciu dát špeciálny prístup, ktorým je integrácia databázovej podpory do existujúcich objektovo orientovaných programovacích jazykov.

4.2 Základy objektovej orientácie

V nasledujúcich podsekciách sa nachádza krátky popis základov objektovej orientácie na ktorej je objektovo–orientovaný model postavený [21]. Zamerali sme sa hlavne na vlastnosti blízko súvisiace s perzistenciou dát a teda nie je možné túto časť považovať za všeobecný návod.

4.2.1 Objekty a triedy

Objektovo orientovaný model je založený na dekompozícii informácií z reálneho sveta na tzv. objekty. Pod pojmom objekt rozumieme každú (i štrukturovanú) entitu, ktorá je jednoznačne a nezávisle identifikovateľná v rámci určitého kontextu okolitého sveta. Objekt tak má jednoznačnú identitu, každé dva i inak dátovo zhodné objekty sú vzájomne odlíšiteľné. Identita objektu je určená identifikátorom (object identifier – oid), ktorý je generovaný systémom, unikátny, nemenný po dobu existencie objektu, skrytý pre programátora i koncového užívateľa. Objekty sú charakterizované pomocou tried. Trieda je abstraktný popis objektu, určuje dátové zložky objektu a operácie (nazývané metódy), ktoré sa dajú nad objektom vykonávať. Každý objekt je inštanciou nejakej triedy, od jednej triedy je možné inštancovať obecne neobmedzený počet štrukturálne zhodných objektov.

4.2.2 Literály

Okrem objektov sa v rámci objektovo orientovaného modelu zavádza aj pojem literálu. Literál je dátová entita určitého dátového typu, ktorá však na rozdiel od objektu nemá vlastnú identitu. Literály sa obvykle vyskytujú ako dátové atribúty objektov. Množina operácií nad dátovým typom literálu je pevne stanovená, nie je možné ju meniť. S objektami a literálmi súvisí pojem premenlivosti (mutability). Premenlivosť je chápaná ako schopnosť meniť dáta pri zachovaní identity – v tomto zmysle sú objekty premenlivé, pretože je možné meniť hodnoty ich dátových zložiek, a pritom si ponechávajú pôvodnú identitu. Naproti tomu literály premenlivé nie sú.

4.2.3 Operácie

Existuje niekoľko základných typov operacii nad objektami. Každý objekt má jeden alebo viacero konšruktorov. Účelom konštruktora je inicializovať objekt v okamžiku jeho vytvorenia. Ďalej je súčasťou každého objektu tzv. deštruktor. Deštruktor je volaný v okamžiku rušenia objektu a jeho cieľom je upratať objekt pred jeho odstránením. Dôležitou operáciou nad objektami je kopírovanie. Rozlišujú se tzv. mäkké (shallow) a hlboké (deep) kópie. V prípade mäkkej kópie dôjde k skopírovaniu atribútov objektu, avšak všetky prípadné odkazy na iné objekty ďalej ukazujú na rovnaké objekty ako v originálnom objekte. Naproti tomu pri hlbokej kópii dôjde nielen k skopírovaniu atribútov kopírovaného objektu, ale sa súčasne vytvorí aj kópia objektov, na ktoré se pôvodný objekt odkazoval.

4.2.4 Zapuzdrenie, skrývanie informácie

Ako už bolo spomenuté, súčasťou triedy je aj definícia operácií, ktoré sa dajú nad objektom vykonávať. Okolie objektov však má prístup len k rozhraniam operácii, vlastná implementácia operácií (niekedy aj časť rozhrania), je vždy pred okolím skrytá. Jedná sa o typickú vlastnosť objektovo orientovaného prístupu, ktorá zvyšuje mieru abstrakcie a nezávislosti objektov. Atribúty a operácie sa dajú označiť ako verejné (public, prístupné z okolitých objektov), súkromné (private, prístupné len v kontexte daného objektu), alebo v niektorých systémoch protected (chránené) – prístupné v objektoch danej triedy a v objektoch tried z tejto triedy zdedených.

4.2.5 Dedičnosť

Dedičnosť je ďalšou typickou vlastnosťou objektového modelu. Vychádza z myšlienky, že niektoré triedy môžu byť špecializovanou verziou inej triedy, resp. určitá trieda je zobecnením jednej či viacero iných špeciálnejších tried. Ak je trieda zdedená z inej triedy, dedí všetky jej atribúty a operácie. Ďalej môže doplniť nové atribúty a operácie, prípadne predefinovať zdedené operácie. Existuje jednoduchá a viacnásobná dedičnosť. Jednoduchá dedičnosť umožňuje triede dediť len z jednej nadradenej triedy, v prípade viacnásobnej dedičnosti môže trieda dediť vlastnosti z väčšieho počtu tried.

4.2.6 Polymorfizmus, neskorá väzba

Polymorfnými operáciami sa označujú operácie, ktoré sa dajú vykonávať nad objektmi rôznych tried. Pritom sa činnosť operácie môže líšiť podľa triedy objektu, nad ktorým je vykonávaná. V objektovo orientovaných programovacích jazykoch sa polymorfizmus metód zvyčajne obmedzuje na triedy, ktoré sú vo vzťahu generalizácie/špecializácie. S polymorfnými operáciami súvisí pojem neskorá väzba. Neskorou väzbou sa označuje spôsob volania polymorfných operácií, keď aplikácia pri volaní operácie dynamicky za chodu programu zvolí kód metódy (vyberie príslušnú implementáciu) na základe triedy objektu, nad ktorým je metóda volaná.

4.3 Porovnanie objektových a relačných databáz

V nasledujúcej sekcii skúmame vo všeobecnosti výhody a nevýhody objektových databáz a objektovo–orientovaného modelu a porovnávame ich s relačnými databázami a relačným modelom. Toto porovnanie sme robili hlavne na základe viacero prác zaoberajúcich sa touto tematikou, ale hlavne *Comparison of Relational Database and Object Oriented Database* [17] a *Performance based Comparison between RDBMS and OODBMS* [30], kde je možnosť detailnejšieho popisu, nakoľko v tejto práci neuvádzame úplný a maximálne detailný zoznam, ale len prehľad niekoľkých z najvýznamnejších faktorov.

4.3.1 Výhody objektových databáz

Medzi hlavné výhody objektovo orientovaných databáz patria hlavne nasledujúce:

- **Prístup k dátam:** Objektovo orientované databázy reprezentujú vzťahy explicitne podporujúce navigačný aj asociatívny prístup k informáciám.

- **Schopnosť zvládnuť rôzne typy dát:** V objektovo orientovanej databáze je možné uložiť akýkoľvek typ dát vrátane textu, čísel, obrázkov, videí alebo hlasu a pod.
- **Modelovanie reálneho sveta:** Objektovo orientovaný systém je schopný modelovať reálny svet prirodzeným spôsobom namiesto tradičných metód, čo je veľmi užitočné pri udržiavaní dát ako napríklad multimediálny obsah.
- **Kombinovanie objektového programovania a databázovej technológie:** Kombinácia týchto dvoch prvkov poskytuje integrovaný systém vývoja aplikácií. Z toho plynú mnohé významné výhody. Operácie, ktorú sú definované, sú aj vykonané a nezávisia na špecifickej databázovej aplikácii bežiacej v tom momente.
- **Zlepšená produktivita:** Dedičnosť umožňuje programátorom vytvoriť riešenia na komplexné problémy definovaním nových objektov z hľadiska objektov definovaných v minulosti. Polymorfizmus a dynamická väzba umožňuje programátorom definovať operácie pre jeden objekt a potom zdieľať špecifikáciu operácie s iným objektom. Tieto objekty môžu ďalej rozšíriť túto operáciu a poskytnúť tak správanie, ktoré je unikátne pre tento objekt.
- **Rozšíriteľnosť:** ODBMS umožňujú vytvoriť nové dátové typy zo starých. Schopnosť vybrať spoločné vlastnosti niekoľkých tried a vytvoriť supertriedu, ktorá môže byť zdieľaná medzi podtriedami a redukuje tak redundantnosť v systéme, je považovaná za jednu z hlavných výhod objektovej orientácie. Znovupoužitie tried navyše propaguje rýchlejší vývoj a ľahkú údržbu databázy a jej aplikácie.
- **Odstránenie Object-relational impedance mismatch:**[1] Jednotnosť v databázovom aj programovacom modely prekonáva tento problém.
- **Podpora evolúcie schémy:** Priliehavá väzba medzi dátami a aplikáciou robí evolúciu a zmeny databázy oveľa prípustnejšími.
- **Bez primárnych kľúčov:** Aby používateľ RDBMS predišiel chybovému stavu, je nútený starať sa o unikátnu identifikáciu n–tíc a ich hodnôt a zabezpečiť, že nemajú žiadne dva n–tice rovnaký primárny kľúč. V ODBMS prebieha unikátna identifikácia objektov na pozadí prostredníctvom OID a je kompletne skrytá pred užívateľom

4.3.2 Nedostatky objektových databáz

Objektové orientované databázy sú zaťažené aj množstvom podstatných nedostatkov, ktoré im zabránili stať sa všeobecne využívaným databázovým systémom. Medzi hlavné patria:

- **Neexistencia všobecného dátového modelu:** Pre ODBMS neexistuje všeobecne uznávaný dátový model a väčšina modelov postrádajú teoretický základ. Táto nevýhoda je považovaná za významný nedostatok v porovnaní s relačnými systémami.
- **Nedostatok skúseností:** V porovnaní s RDBMS je využívanosť ODBMS realtívne limitovaná. To znamená, že nemáme dostatočnú úroveň skúseností ktoré máme s tradičnými systémami. Pokiaľ sú ODBMS stále limitované na malé časti trhu, tento problém bude pretrvávať.

[1] konceptuálne a technické ťažkosti v prípade že je na RDBMS postavená aplikácia napísaná v objektovo orientovanom jazyku alebo štýle – https://en.wikipedia.org/wiki/Object-relational_impedance_mismatch

- **Nedostatok noriem:** Pre ODBMS je vo všeobecnosti nedostatok štandardov. Taktiež neexistuje štandardný objektovo orientovaný dopytovací jazyk.
- **Povedomie ľudí o ODBMS:** Ľudia často vôbec nepoznajú, alebo nemajú hlbšie znalosti o objektových databázach a tým pádom nemôžu ODBMS v obchodnej sfére expandovať.
- **Nedostatočné udržiavanie konzistencie:** ODBMS poskytujú len limitovaný počet čŕt obmedzujúcich dáta na dodržanie konzistencie a plne neimplementujú integritné obmedzenia.
- **Problémy so zabezpečením:** Relačné databázy podporujú autorizáciu, ale väčšina objektových databáz autorizáciu nepodporuje. V objektových databázach užívateľ vyžaduje explicitné nastavenie a uvoľnenie zámkov, zatiaľ čo relačné databázy mechanicky nastavujú a uvoľňujú zámky pri používateľskom dopyte a správach o aktualizácii. [2]

4.3.3 Porovnanie

Na záver môžeme porovnať hlavné charakteristiky medzi medzi objektovými a relačnými DBMS z pohľadu modelovania dát v tabuľke 4.1, a ich cieľov v tabuľke 4.2.

Tabuľka 4.1: Porovnanie ODBMS a RDBMS z hľadiska modelovanie údajov.

Objektovo orientovaný model	**Relačný model**	**Rozdiely**
Objekt	Entita	Objekt má špecifikované aj správanie
Trieda objektov	Typy entít	Trieda objektov zahŕňa spoločné chovanie objektov v tejto triede
Hierarchia tried	Databázová schéma	Hierarchia tried zahŕňa dedičnosť, schéma zahŕňa externé kľúče
Triedna inštancia	Entita, ntica alebo záznam.	Trieda môže mať obmedzujúcejšiu činnosť
Object identifier (OID)	Primary key	V relačnom modeli primárny kľúč nie je identifikovaný systémom.

4.4 Dátový model ODMG

Ako už bolo spomenuté v sekcii 4.1, ODBMS nemá žiadny oficiálny štandard. Skupina OMG[2] však štandardizuje objektovo orientované systémy na obecnejšej úrovni. Z tejto práce

[2]Skupina viac než 300 komerčných firiem, spoločností vyvíjajúcich databázové systémy, a ich užívateľov.

Tabuľka 4.2: Porovnanie ODBMS a RDBMS z ohľadom na ich ciele.

ODBMS	**RDBMS**
Hlavný cieľ: zapuzdrenie dát a nezávislosť	Hlavný cieľ: zaistenie nezávislosti dát od aplikačných programov.
Nezávislosť tried: triedy môžu byť rozpoznané bez ovplyvnenia spôsobu ich používania.	Dátová nezávislosť: dáta môžu byť rozpoznané bez ovplyvnenia spôsobu ich používania.
Ukladá dáta a metódy.	Ukladá dáta.
Zapuzdrenie: dáta môžu byť využívané iba skrz ich triedne metódy.	Rozdeľovanie dát: dáta môžu byť rozdelené na základe požiadaviek užívateľa a na špecifickej užívateľskej aplikácii.
Komplexnosť: štruktúra dát, vrátane dátových typov, môže byť komplexná.	Jednoduchosť: užívateľ vníma dáta ako stĺpce, riadky/ntice a tabuľky.

potom vychádza už spomenutý všeobecne uznávaný štandard ODMG [7]. Jeho časťou je taktiež dátový model ODMG, ktorý sme zvolili ako štandard i pre našu prácu.

Logický objektový model ODMG je postavený na obecných princípoch objektovo orientovaných systémov, ako boli prezentované v sekcii 4.2. Preto je tu uvedený len základný prehľad a vyzdvihnuté prípadné rozdiely. Dátový model ODMG definuje:

- *Objekty a literály.* Ako objekty, tak i literály boli podrobne popísané v sekcii 4.1.
- *Atribúty a vzťahy.* Obe majú rovnaký význam a typy ako v relačnom modeli.
- *Objektové typy.* Pod pojmom objektový typ definuje ODMG to, čo sa obvykle označuje termínom trieda. Okrem základných vlastností triedy doplňuje ODMG definíciu triedy o vyššie popísané vzťahy, kľúčové položky (keys) a tzv. extent.[3]
- *Preddefinované typy.* ODMG definuje radu zabudovaných atomických i štrukturovaných dátových typov. Štandard zároveň zavádza aj bohatú množinu kolekcií polymorfného charakteru.

4.5 Db4O

Db4o (database for objects – databáza pre objekty) je vstavaná, open–source, zero–admin objektová databáza pre Javu a .NET, ktorým poskytuje príslušné API.[4] Bola vyvíjaná, komerčne licencovaná a podporovaná spoločnosťou Actian. Dlhodobo to bola najobľúbenejšia a najpoužívanejšia objektová databáza. V roku 2014 však Actian prestal podporovať a ďalej

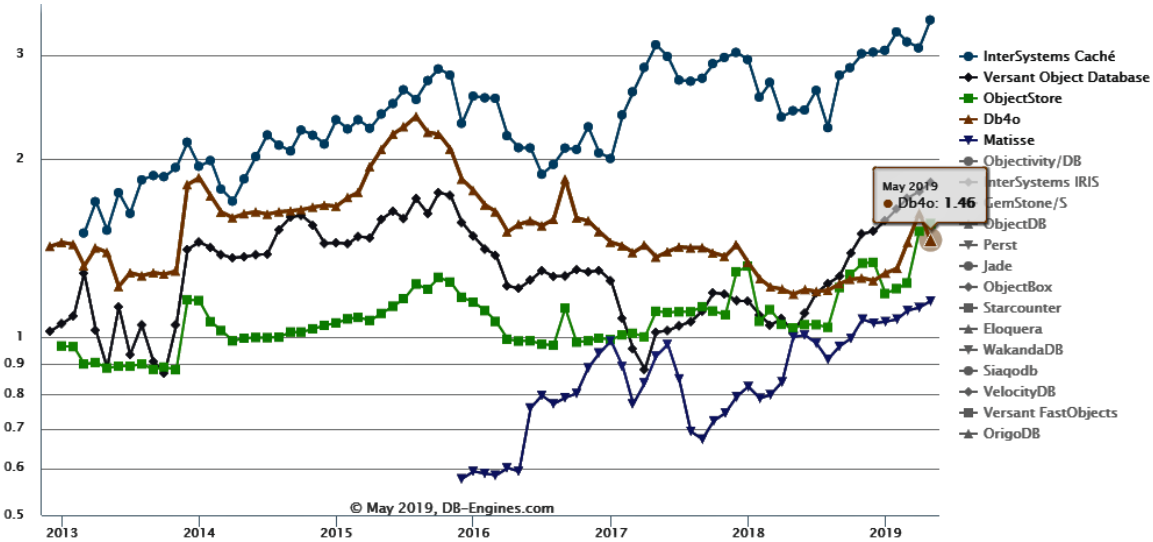
[3] Automaticky spravovaná kolekcia všetkých objektov danej triedy. Logický ekvivalent tabuľky záznamov z relačných databáz. Kľúče sú potom dátové položky, ktoré sú unikátne pre každý objekt danej triedy.

[4] Application programming interface – aplikačné programovacie rozhranie je zbierka procedúr, funkcií, tried či protokolov nejakej knižnice, ktoré môže programátor využívať. API určuje, akým spôsobom sú funkcie knižnice volané zo zdrojového kódu programu. https://cs.wikipedia.org/wiki/API

vyvíjať db4o produkty. Jedným z hlavných cieľov db4o je poskytnúť jednoduché a prirodzené rozhranie na perzistenciu objektov v objektovo orientovanom programovaní. Vývoj s touto databázou nevyžaduje vytvorenie separátneho dátového modelu, triedy aplikácie definujú štruktúru dát. Db4o funguje len na operačných systémoch podporujúcich javu alebo .NET. Ako vstavaná databáza beží v aplikačnom procese. Je distribuovaná ako knižnica (jar/dll). Podporuje taktiež vývoj schémy, indexovanie, transakčné spracovanie a paralelnosť, databázové šifrovanie a replikačnú službu (medzi db4o databázami a určitými relačnými databázami). Pre implementáciu sme zvolili stabilnú verziu 7.4.

4.5.1 Proces výberu

Hlavné kritériá pre výber sme zvolili rovnaké ako pri RDBMS, teda dostupnosť, využívanosť a popularita, reprezentatívnosť, efektívnosť, rýchlosť, bezpečnosť a pod. Výber bol jednoduchý a jednoznačný z hľadiska dostupnosti a využívanosti, nakoľko, ako môžeme vidieť v obrázku 4.2, je z deviatich najobľúbenejších databáz podľa rebríčka popularity [26] 8 pod komerčnou licenciou. Db4o na štvrtom mieste je teda z nich jediná voľne dostupná objektová databáza. Ďalšou voľne dostupnou objektovou databázou je Perst, ktorý však nedosahuje ani tretinu popularity db4o napriek tomu, že je db4o už dlhšiu dobu nepodporovaná. To je zároveň ale aj dôvod, prečo bol výber správnej objektovej databázy zároveň ťažkou voľbou. Objektové databázy ako technológia nikdy nedosiahli výrazného úspechu na trhu a momentálne dramaticky upadajú. Jediný vývoj alebo údržba sa navyše deje pod komerčnými licenciami, a tak je pre verejnosť aktuálna objektová databáza v podstate nedostupná. O úpadku db4o symbolicky svedčí to, že i jej oficiálne stránky,[5] či stránky na podporu už nie sú dostupné a je problém vôbec sa dostať k stabilným verziám tejto databázy. Db4o teda nemôžeme považovať za dostupnú, populárnu a vzhľadom na posun technológii už dnes ani za rýchlu či bezpečnú databázu. Spĺňa však tieto kritériá zo všetkých voľne dostupných ODBMS najlepšie a navyše ju môžeme považovať za vhodného reprezentanta myšlienok a konceptu objektovo orientovaného modelu databáz.



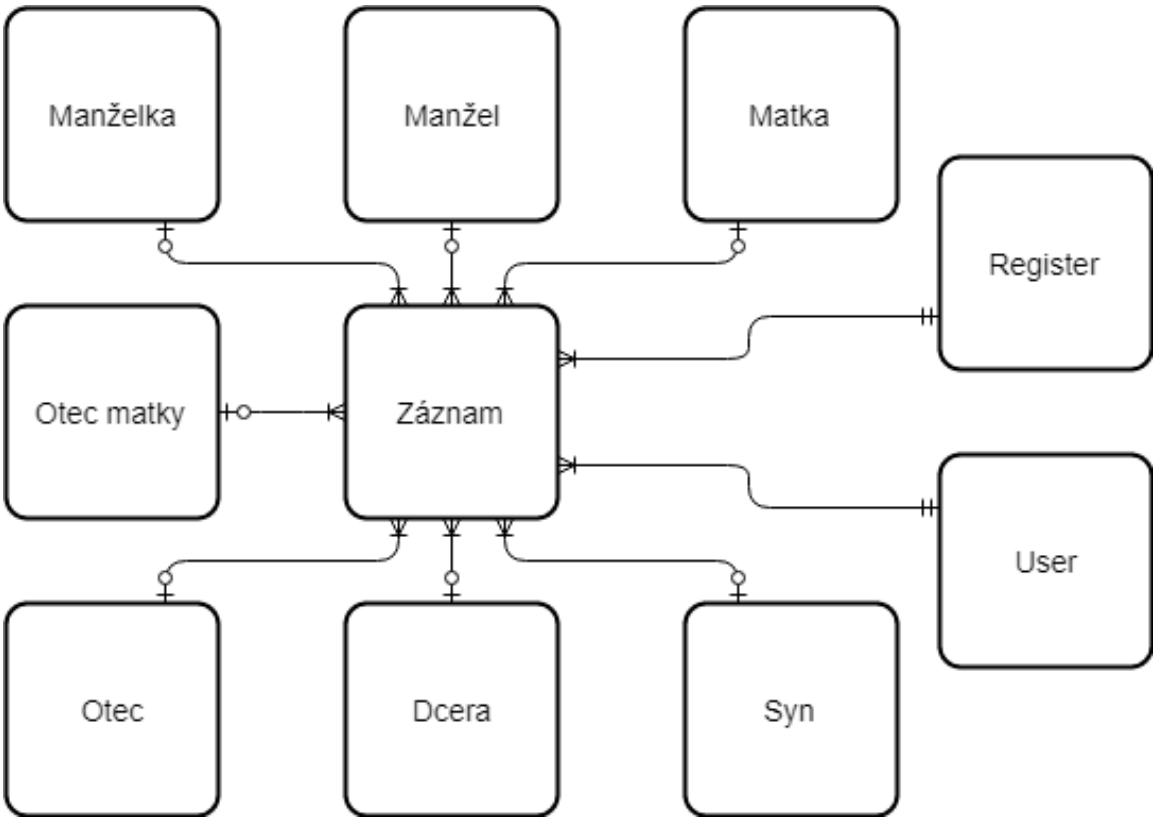
Obr. 4.2: DB-Engines rebríček popularity ODBMS. Horizontálna os reprezentuje skóre (logaritmická stupnica) [26]

[5] http://www.db4o.com/

Kapitola 5

Dáta o úmrtiach

Celkový počet už v sekcii 2.1 definovaných atribútov sa v prípade úmrtných dát vyšplhal na 84. Tie sú prirodzene rozdelené v desiatich entitách. Celkový počet vo vzorke poskytnutých a v štatistike zahrnutých záznamov je 736. Spojením všetkých tabuliek v tejto kapitole vzniká dátový slovník pre úmrtné dáta. V obrázku 5.1 sú bez atribútov zobrazené entity, ktoré záznamy obsahujú, a základné logické vzťahy medzi nimi.



Obr. 5.1: Graf entít, zapísaných v úmrtných dátach, a logických vzťahov medzi nimi.

5.1 Príklad úmrtného záznamu

V obrázku 5.2 je zobrazený vyexportovaný príklad jedného konkrétneho záznamu z referenčnej vzorky. Tento záznam môžeme považovať za klasickú ukážku záznamov v matrike a je z neho zreteľná základná podoba dát. Ako môžeme vidieť, pri porovnaní s celkovým počtom atribútov (84) je veľké (nadpolovičné) množstvo záznamov nevyplnených, a teda sa ani v exporte nenachádzajúcich. Tak isto je tu demonštrovaný i ďalší problém už spomínaný v sekcii 2.4, a to konkrétne chybovosť. Môžeme si všimnúť, že „jméno manžela zemřelého“ nezačína majuskulou, čo je nezvyčajná prax a vzhľadom na to, že sa táto forma v ostatných záznamoch z rovnakého obdobia nenachádza, môžeme to považovať za chybu a meno uložiť ako „Františku“. To je však v rozpore s v už spomínanej sekcii danou premisou o úplnom a neupravenom uložení dát a tak bude osoba uložená ako „františku“, i keď tým vzniká v prípade ďalšieho zápisu rovnakej osoby v správnom formáte šanca duplicity a redundantnosti dát.

zpracovatel/ka Vostrý	**Záznam hotov** 1	**Archiv** MZA
Fond E67	**Signatura** 1014	**Pořadí scanu** 10
rozložení na scanu (C, L, P) L	**Pořadí záznamu** 1	**datum úmrtí** 7.7.1882
datum pohřbení 9.7.1882	**zaopatřovatel jméno** Vilém	**zaopatřovatel příjmení** Sedlák
zaopatřovatel titul farář	**pohřbívající jméno** Vilém	**pohřbívající příjmení** Sedlák
pohřbívající titul/povolání farář	**úmrtí obec** Babice	**úmrtí č. p.** 26
vdovec/vdova 1	**jméno zemřelého** Antonia	**obec zemřelého** v Babicích
manžel zemřelého mrtev 1	**jméno manžela zemřelého** františku	**příjmení manžela zemřelého** Jahodu
povolání manžela zemřelého čtvrtníku	**obec manžela zemřelého** v Babicích	**věk zemřelého roky** 65
příčina úmrtí ochrnutí plic	**poznámky** katolického, ženského	

Obr. 5.2: Ukážka konkrétneho záznamu vyexportovaná z poskytnutých dát o úmrtiach.

5.2 Záznam

Najpodstatnejšie údaje popisujúce udalosť, ktoré sú jadrom historického záznamu, sú zhrnuté v entite pomenovanej *Záznam*. Tá je centrálna pre celý model a schému dát.

5.2.1 Hlavné dáta

Jednotlivé atribúty entity *Záznam*, ktoré majú jednoduchú štruktúru, teda bez potreby alebo náznaku vytvorenie vlastnej entity, sa nachádzajú v tabuľke 5.1.

Tabuľka 5.1: Tabuľka úmrtných atribútov entity Záznam. Z = zemřelý

				rozsiahlosť	
entita	**atribút**	**dát. typ**	**početnosť**	**avg**	**max**
Záznam	datum zaopatření	dátum	5.03%	9.19	11
Záznam	datum úmrtí	dátum	58.02%	8.01	11
Záznam	datum pohřbení	dátum	75.54%	8.70	12
Záznam	úmrtí obec	text	82.20%	12.77	43
Záznam	úmrtí ulice	text	4.35%	14.66	40
Záznam	úmrtí č. p.	číslo	42.66%	2.24	3
Záznam	místo pohřbení	text	40.76%	28.28	43
Záznam	místo úmrtí[1]	text	2.17%	19.31	49
Záznam	vdovec/vdova	boolean	4.35%	1.00	1
Záznam	pohlaví Z	boolean	43.61%	1.00	1
Záznam	jméno Z	text	96.33%	6.43	24
Záznam	příjmení Z	text	56.11%	8.00	28
Záznam	povolání Z	text	35.05%	11.02	63
Záznam	obec Z	text	51.63%	12.42	50
Záznam	ulice Z	text	3.80%	14.89	40
Záznam	č. p.[2] Z	číslo	41.95%	2.10	3
Záznam	vyznání Z	boolean	29.76%	1.00	1
Záznam	věk Z roky	číslo	42.66%	1.70	3
Záznam	věk Z měsíce	číslo	17.80%	1.11	2
Záznam	věk Z dny	číslo	12.91%	1.54	3
Záznam	věk Z hodiny	číslo	1.36%	1.90	4
Záznam	příčina úmrtí	text	52.58%	12.28	71
Záznam	ohledání A/N	boolean	26.36%	1.00	1
Záznam	ohledání kým	text	1.09%	14.38	21
Záznam	narození Z	dátum	8.97%	8.05	12
Záznam	obec narození Z	text	7.07%	14.52	34
Záznam	č. p. narození Z	číslo	–	–	–
Záznam	poznámky	číslo	39.95%	27.31	310

Atribút *vyznání zemřelého* bol označený ako boolean, nakoľko sa v poskytnutej vzorke buď nenachádza, alebo je označený ako „K“, čo je označenie pre člena Latinskej cirkvi.[3] K atribútu *místo úmrtí* je potrebné zaznamenať, že je uvedený len ak sa líši od bydlište zemřelého, a túto informáciu bude potrebné nejakým spôsobom reprezentovať aj v implementácii databázy. Atribút *č. p. narození zemřelého* sa vo vzorke nenachádza ani raz a teda sme vzhľadom na počet záznamov (736) hodnotu početnosti zvolili ako 0.5‰.

[1] pokud se liší od bydliště

[2] číslo popisné

[3] vo verejnosti viac známa pod nesprávnym názvom Rímskokatolícka cirkev – https://sk.wikipedia.org/wiki/Latinsk%C3%A1_cirkev

5.2.2 Potencionálne samostatné dáta

Rýchlym pohľadom na obrázky 5.1 a 5.2 zistíme, že sa všetky pôvodné atribúty (všetky okrem tých popísaných v sekcii 5.4) veľmi prirodzenie delia do dvoch skupín. Prvá obsahuje najmä informácie o zomretom, pohrebe a okolnostiach. Druhou sú osoby v nejakom vzťahu s mŕtvym, ktorým je venovaných niekoľko ďalších podkapitol. Prvá skupina v podstate tvorí entitu *Záznam*. V rámci tejto skupiny však existuje podmnožina atribútov, zobrazených v tabuľke 5.2, ktoré sú blízko prepojené a spolu vytvárajú krátky profil špeciálnej osoby (*zaopatřovatel* a *pohřbívající*), avšak nie sú nevyhnutne vo vzťahu s mŕtvym.

Tabuľka 5.2: Tabuľka potencionálne samostatných úmrtných atribútov, ktoré momentálne obsahuje entita Záznam.

				rozsiahlosť	
entita	**atribút**	**dát. typ**	**početnosť**	**avg**	**max**
Záznam	zaopatřovatel jméno	text	23.64%	6.28	18
Záznam	zaopatřovatel příjmení	text	24.86%	6.44	11
Záznam	zaopatřovatel titul	text	24.05%	10.14	100
Záznam	pohřbívající jméno	text	49.46%	6.36	29
Záznam	pohřbívající příjmení	text	52.17%	6.80	23
Záznam	pohřbívající titul[4]	text	50.82%	9.98	36

Tieto atribúty sú taktiež veľmi vhodnou demonštráciou už spomínaného problému chybovosti a toho, ako významne tento faktor ovplyvňuje celý návrh databáz. Z pohľadu na tabuľku 5.2 nie je možné určiť, ako často a či sa vôbec rovnaké špeciálne osoby nachádzajú opakovane pri rôznych záznamoch. Navyše máme o týchto osobách len informácie o mene, priezvisku a titule (respektíve povolaní), ktoré nemusia stačiť na jednoznačnú identifikáciu osoby. V poskytnutých dátach je však viditeľné, že ako zaopatřovatel tak aj pohřbívající často vystupujú rovnaké osoby rovnakého povolania/titulu („farář“, „vikař“, „kooperator“ atď) i keď pod rôznymi názvami („vikař“, „Vicarius“, „Vikär“, „Wikar“) a často mierne pozmenenom zápise mien.

Na základe tohto vieme odvodiť logickú štruktúru vzťahov medzi entitami tak, ako je uvedená v grafe 5.1. Takisto je však očividné, že by sa i pri jednoduchom zlučovaní identických entít nepredišlo redundancii dát a nedosiahla by sa ich konzistencia. Ak ďalej vezmeme do úvahy početnosť dát (údaje zbierané stovky rokov na relatívne rozsiahlom území) a historické okolnosti, ako napríklad časté dedenie mužských mien v rodine a úzka sada využívaných mien [6], tak môžeme konštatovať, že riziko agregácie dvoch rozdielnych osôb do jednej a tým aj nenávratná strata niektorých údajov nie je zanedbateľné. Z týchto dôvodov sa v tejto práci neberie do úvahy logická kardinalita medzi entitami (napr. možnosť jednej osoby/entity syna byť zúčastneného pri viacerých záznamoch) ale sa považuje každá osoba v každom zázname za unikátnu.

Ďalej však v rámci projektu, ktorého súčasťou je táto práca, prebehne detailnejšia analýza uložených záznamov v nami, i ostatnými pracovníkmi vytvorených databázach, a na jej základe bude prebiehať identifikácia a zlučovanie osôb. Po zvážení všetkých týchto skutočností sme sa rozhodli, že v rámci tohto dátového slovníka je vhodnejšie ponechať tieto atribúty v entite *Záznam*, čo však úplne nezamedzuje vytvoreniu ich vlastnej entity pri implementácii, či pri rozširovaní databázy v budúcnosti.

[4] alebo povolání

5.3 Osoba

Ďalšou skupinou atribútov sú tie, ktoré sa blízko viažu na osoby súvisiace s mŕtvym alebo s pohrebom. Viažu sa síce na rozdielne osoby/entity, všetky tieto entity však majú takmer identickú štruktúru, alebo sú jej derivátom. Konkrétne sú týmito osobami *otec*, *matka*, *otec matky*, *manžel*, *manželka*, *syn* a *dcera*. Atribúty ako také majú rovnakú formu a štatistiky veľmi podobné. Taktiež sa tu, ako aj pri Zázname, stretávame s problémom nekompletnosti. Teraz však nie sú úplne absentujúce len nejaké atribúty, ale úplne chýba príklad čo i len jedného záznamu syna alebo dcéry. Čo sa týka ostatných entít, úplne chýbajú taktiež atribúty *matka zemřelého mrtva*, *ulice matky zemřelého*, *obec manželky zemřelého*, a *č. p. manželky zemřelého*. Z týchto dôvodov (a taktiež kvôli prehľadnosti a nadmernému rozsahu v prípade uvedenia všetkých tabuliek) sme sa rozhodli neuvádzať v tejto kapitole každú entitu v samostatnej tabuľke, ale vytvoriť len jednu vzorovú agregovanú tabuľku 5.3.

Tabuľka 5.3: Tabuľka úmrtných atribútov viažúcich sa na entity reprezentujúce osoby otec (O), matka (M), otec matky (OM), partner (P), syn (S) a dcera (D). ∀ značí prítomnosť vo všetkých entitách.

				rozsiahlosť	
entita	**atribút**	**dát. typ**	**početnosť**	**avg**	**max**
O/M/P/S/D	mrtev	boolean	1.12%	1	1
∀	jméno	text	17.97%	6.39	22
∀	příjmení	text	17.34%	8.32	28
∀	povolání	text	11.51%	9.34	38
∀	obec	text	9.34%	11.67	32
O/M	ulice	text	1.05%	4.28	11
O/M/P/S/D	č. p.	číslo	3.19%	2.85	3

Za zvláštne sa dá považovať oddelenie manžela a manželky, nakoľko z historických a kultúrnych okolností je jasné, že v lokalite, odkiaľ zaznamenané dáta pochádzajú (historická Morava, severné Rakúsko, Slezko), sa nikdy obe osoby pri jednom zázname nachádzať nebudú. Ešte zvláštnejšie sa toto rozhodnutie kronikárov zdá po zistení, že manželka sa v celej vzorke nachádza len jediný krát, samozrejme bez ďalšieho manžela, a navyše záznam vyzerá ako omyl – zomrelá je žena a „manželka“ má meno mužského rodu. Z týchto dôvodov nie je v tomto prípade rešpektovaná pôvodná štruktúra, ale sú spomínané osoby spojené do jednej entity *partner*. Takisto stojí pri implementácii databáz za zváženie spojenie všetkých týchto osôb do jednej entitnej množiny osoba, vytvorenej podľa príkladu v tomto dátovom slovníku, doplnenej jedným atribútom určujúcim typ osoby.

5.4 Pridané dáta

Pri spracovávaní originálnych záznamov vzniklo niekoľko nových informácii, ktoré je potrebné dlhodobo udržiavať pre ľahšiu organizáciu a orientáciu v záznamoch. Tie sa delia na nasledujúce dve skupiny.

5.4.1 Samostatné pridané dáta

Pridané informácie sú však často pre mnoho záznamov identické a sú tak prirodzene oddelené v dvoch samostatných entitách *User*, ktorá udržiava informácie o spracovateľoch a *Register*, ktorá jednoznačne identifikuje knihu obsahujúcu daný záznam. Spolu s ďalšími potrebnými informáciami o atribútoch ich môžeme vidieť v tabuľke 5.4.

Tabuľka 5.4: Tabuľka úmrtných atribútov vzniknutých pri spracovávaní, udržiavaných v samostatných entitách.

				rozsiahlosť	
entita	**atribút**	**dát. typ**	**početnosť**	**avg**	**max**
User	zpracovatel/ka	text	100%	8.67	12
Register	archiv	text	100%	3.00	3
Register	fond	text	100%	3.00	3
Register	signatura	číslo	100%	4.46	5

5.4.2 Nesamostatné pridané dáta

Z predchádzajúcej sekcie 5.4.1 vieme o pridaných dátach. Táto je venovaná rovnakému typu informácií s tou zmenou, že tieto nemôžu existovať samostatne a sú pre každý atribút unikátne a nevyhnutné. Z tohoto dôvodu sa prirodzene viažu na entitu *Record*, ktorá bude bližšie špecifikovaná v nasledujúcej podkapitole 5.2. Podstatná informácia, ktorú treba zaznamenať je, že doména atribútu[5] *rozložení na scanu* je **C**, **L** a **P**. Informácie o týchto atribútoch môžeme vidieť v nasledujúcej tabuľke 5.5.

Tabuľka 5.5: Tabuľka úmrtných atribútov vzniknutých pri spracovávaní ich dátového typu viažúcich sa na entitu Record.

				rozsiahlosť	
entita	**atribút**	**dát. typ**	**početnosť**	**avg**	**max**
Záznam	záznam hotov	číslo	100%	1.00	1
Záznam	pořadí scanu	číslo	100%	2.34	3
Záznam	rozložení na scanu	text	100%	1.00	1
Záznam	pořadí záznamu	číslo	100%	1.57	2

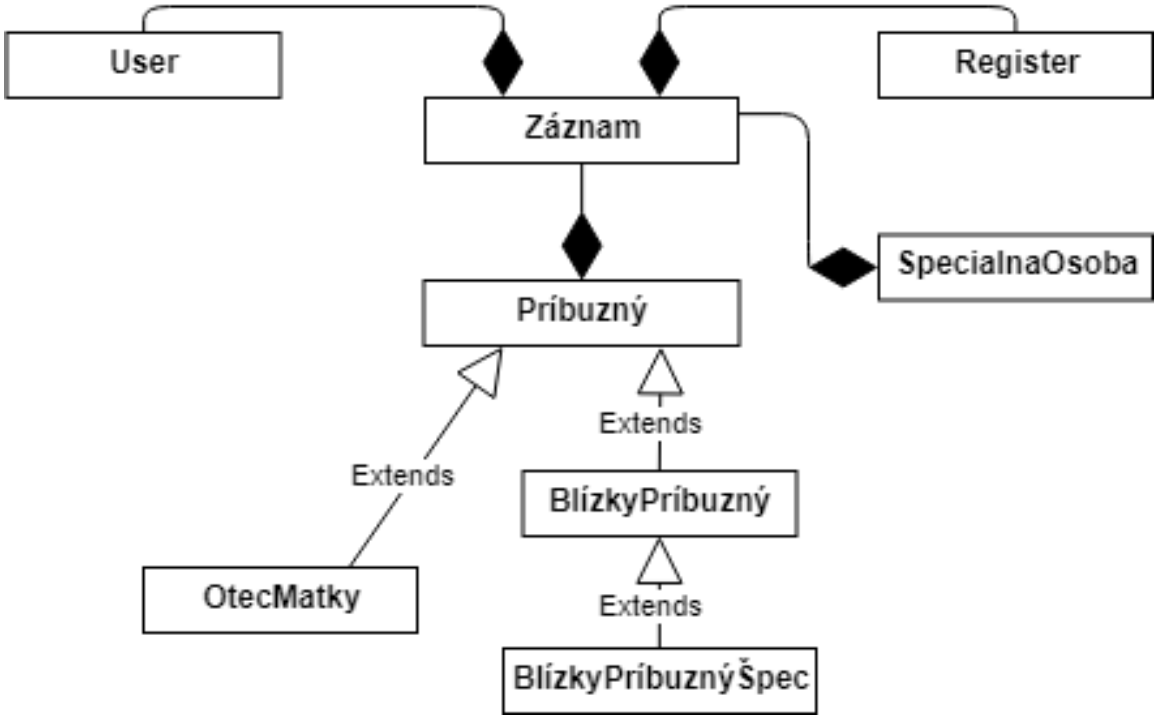
5.5 Konceptuálna schéma

Na základe analýzy dát a následného vytvorenia dátového slovníka a štatistík sme splnili prvý krok v procese návrhu databázy. Na základe toho sme v ďalšom kroku boli schopní vytvoriť konceptuálnu schému databázy a tú ďalej konvertovať na logickú schému, a tak splniť aj druhý a tretí bod návrhu databázy.

[5] Pojem používaný v teórii databáz označujúci množinu všetkých hodnôt, ktoré môže určitý atribút dat nadobudnúť. Doména hodnôt môže byť určená zoznamom (enumeráciou) hodnôt. [24]

5.5.1 Objektovo orientovaný model dát o úmrtí

Pre objektovo orientovaný model je konceptuálny návrh zobrazený v diagrame 5.3. Tento diagram tried neobsahuje atribúty, nakoľko sú zhodné s ER modelom, ktorý môžme považovať za všeobecný high level popis dát, pri ktorom sme taktiež popísali základnú stratégiu modelovania záznamov. Detailnú implementáciu objektovo orientovaného modelu navyše uvádzame v obrázku 7.1.

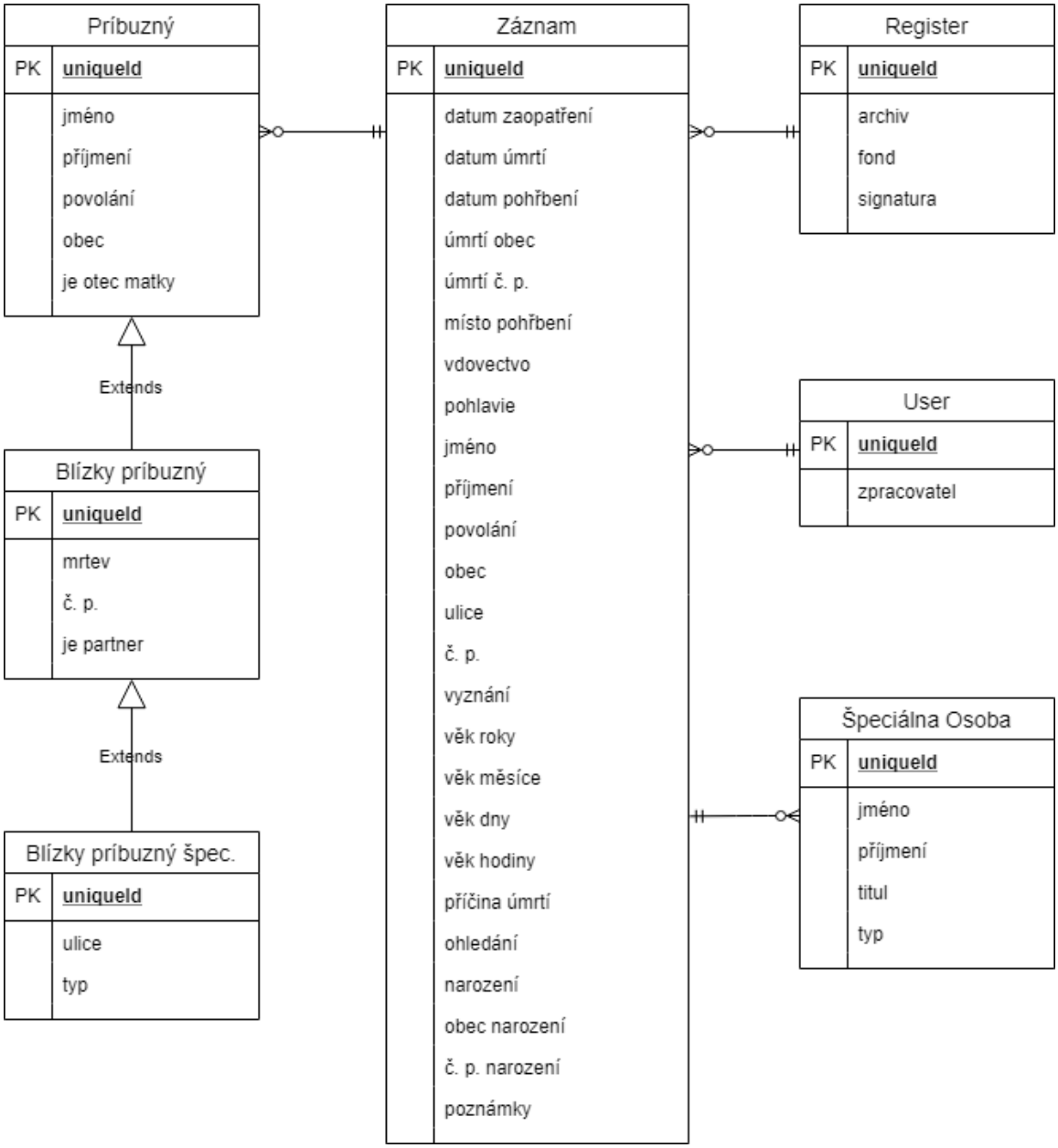


Obr. 5.3: Objektovo orientovaný model úmrtných dát.

5.5.2 ER diagram dát o úmrtí

Na obrázku 5.4 je zobrazený entitno–vzťahový diagram pre sadu matričných dát o úmrtiach. Osoby s veľmi podobnou štruktúrou sme sa rozhodli modelovať ako dedičnosť, respektíve zovšeobecnenie. Nie je to síce ideálnym modelovaním reálneho sveta, napr. otec „nerozširuje“ starého otca, ale je to vhodným popisom štruktúry dát a z určitého pohľadu i ekvivalentnou abstrakciou reálneho sveta – o bližšej osobe (otec) toho vieme viac, ako o vzdialenejšej (starý otec). Na rôznych leveloch dedičnosti sa však stále v jednej entitnej množine nachádza viacero rôznych osôb, čo je reprezentované diskriminačnými atribútmi (je otec matky, typ atď.).

Entitná množina špeciálna osoba je aj napriek svojej čiastočnej podobnosti s ostatnými osobami príliš rozdielna na to, aby sa dalo pri efektívnej implementácii pomýšľať na jej agregáciu do štruktúry ostatných osôb, a tak je modelovaná izolovane. Reprezentuje potenciálne samostatné údaje zo sekcie 5.2.2. Keďže sú to dve rôzne osoby s identickou štruktúrou, nenachádzajúce sa v záznamoch s vysokou početnosťou, je pamäťovo efektívnejšie riešenie modelovať ich týmto spôsobom. To má zas pozitívny vplyv na prácu s tabuľkou *Záznam*



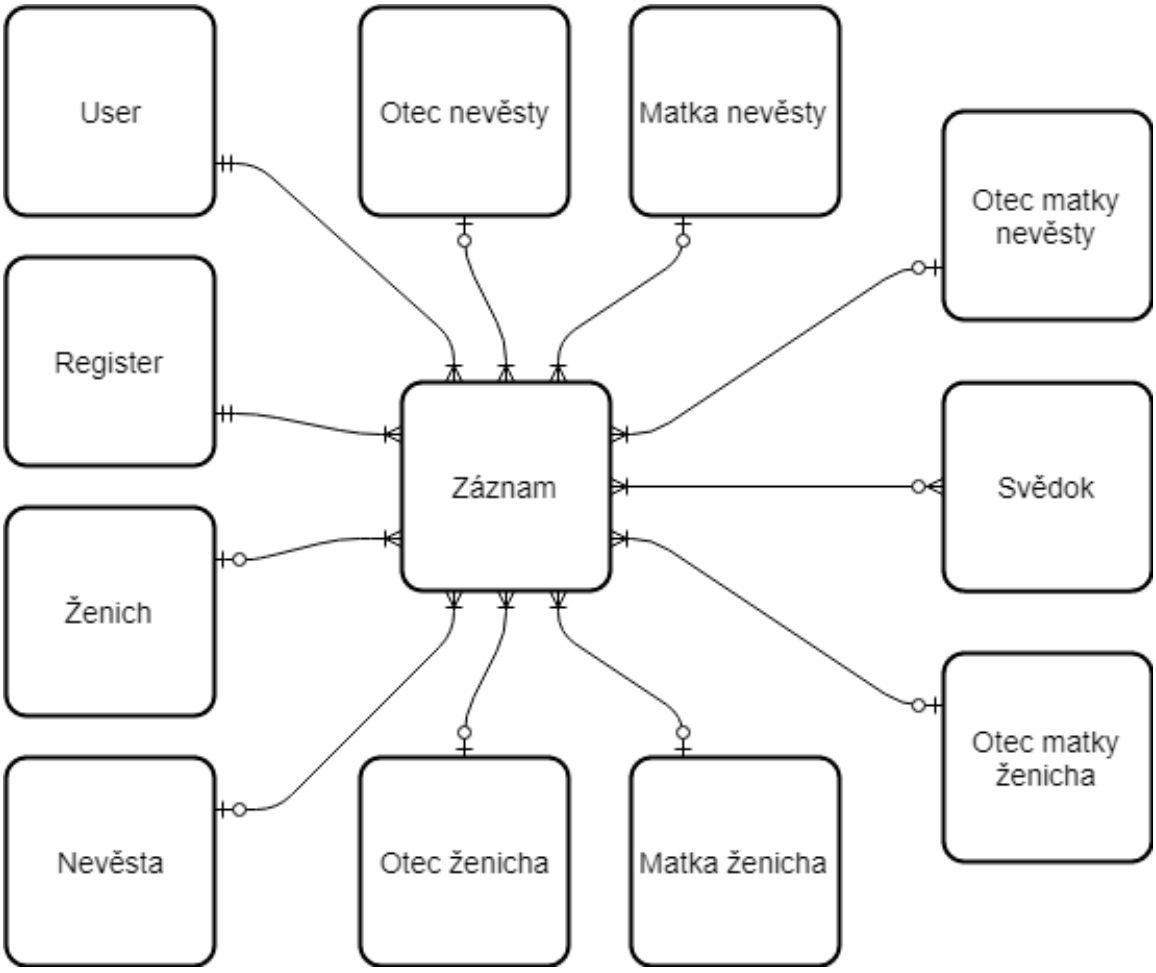
Obr. 5.4: ERD úmrtných dát.

i prácu s týmito osobami. Negatívom je vytvorenie ďalšej tabuľky a teda občasná potreba spájania tabuliek pri vyhľadávaní. Pozitíva však jednoznačne prevládajú nad negatívami, bez ohľadu na zvolený DBMS a teda volíme tento typ modelovania. Tento graf je konceptuálnym návrhom databázy, a zatiaľ čo poskytuje reálnu predstavu o jej podobe, ponecháva i priestor na úpravu a výber najvhodnejších implementačných možností z pohľadu konkrétneho RDBMS pre fyzický návrh databázy.

Kapitola 6

Dáta o svadbách

Celkový počet atribútov sa v tomto prípade vyšplhal na 156, s prirodzeným rozdelením do 12 entít. V nasledujúcich podkapitolách bude rovnako tabuľkovou formou postupne vytváraný dátový slovník. Počet momentálne digitalizovaných záznamov je znova nízky, konkrétne je k dispozícii len 394 z nich. V obrázku 6.1 sú bez atribútov zobrazené entity ktoré záznam obsahuje a základné logické vzťahy medzi nimi.



Obr. 6.1: Graf entít zapísaných v dátach o svadbách a logických vzťahov medzi nimi.

6.1 Príklad sobášneho záznamu

V obrázku 6.2 je zobrazený vyexportovaný príklad jedného konkrétneho záznamu z referenčnej vzorky sobášnych dát. Tento záznam taktiež môžeme považovať za klasickú ukážku záznamov v matrike a je z neho zreteľná základná podoba dát. I tu sú chýbajúce atribúty kvôli rozsahu exportu radšej vynechané a vidíme ešte väčšiu mieru absencie (takmer 80 percent). Príklad taktiež ukazuje pri neveste a 4. svedkovi identický problém s chybovosťou, ako v príklade úmrtných dát (sekcia 5.1). Tak isto je tu demonštrovaný i ďalší už popísaný problém s potencionálne samostatnými dátami. Titul oddávajúceho je z neznámych dôvodov zapísaný skratkou, ale aj celým názvom v zátvorke v rámci jedného atribútu, čo spôsobí pri automatickom spracovaní nezhodu s jednoducho zapísaným povolaním. Podobne sa v celej kapitole budeme stretávať s podobnými problémami ako pri úmrtných dátach, avšak keďže je pri nich už metodika spracovávania dát uvedená, budú dáta o svadbách popísané stručnejšie.

zpracovatel/ka Hausvaterová	**Záznam hotov** 1	**Archiv** MZA
Fond E67	**Signatura** 15949	**Pořadí scanu** 89
rozložení na scanu (C, L, P) P	**Pořadí záznamu** 1	**datum sňatku** 28.10.1635
oddávající jméno Petr	**oddávající příjmení** ham.	**oddávající titul** par.(parochus)
obec ke které svatba náleží fara (Neo Veseli)	**Jméno ženicha** Andreas	**povolání ženicha** mladenecz
otec ženicha mrtev? 1	**Jméno otce ženicha** Bartona	**Příjmení otce ženicha** Chalupj
Obec otce ženicha z Matiegowa	**Jméno nevěsty** kacža	**povolání nevěsty** dieweczka
otec nevěsty mrtev? 1	**Jméno otce nevěsty** Martinowi	**Příjmení otce nevěsty** Myraussowj
Obec otce nevěsty z Brzezj	**Jméno svědka 1** Jan	**Příjmení svědka 1** Sswomu
Jméno svědka 2 Tomass	**Příjmení svědka 2** Osmjczky	**Jméno svědka 3** Manda
Příjmení svědka 3 Rychtarzka	**Jméno svědka 4** Salka	**Příjmení svědka 4** ganausskowa

Obr. 6.2: Ukážka konkrétneho záznamu vyexportovaná z poskytnutých dát o svadbách.

6.2 Záznam

Pri svadobných dátach platí, tak ako aj pri úmrtných, že entita *Záznam* predstavuje jadro historického záznamu s centrálnym postavením v modeli dát a zaobaľuje najkľúčovejšie úzko súvisiace atribúty týkajúce sa udalosti – teda svadby.

6.2.1 Hlavné dáta

Samostatné údaje sa nachádzajú v tabuľke 6.1. Problematické sú viaceré atribúty, ktoré sa vo vzorke vôbec nenachádzajú. Konkrétne to sú *neurčené č. p. 1*, *neurčené č. p. 2*, ďalej *1. ohlášky*, *2. ohlášky* a *3. ohlášky*, ktoré sa v záznamoch nikde inde nevyskytujú a tak nemáme žiadne informácie o ich forme, preto ich budeme považovať za textový atribút. Pri atribúte *stupeň příbuznosti* je zas diskutabilná štruktúra, nakoľko je táto informácia sama o sebe zapísaná v čísle ale v rámci jedného záznamu je ich často viac. Poslednou problémovou informáciou je *zplnoletnění nevěsty* nakoľko je táto informácia v niektorých archívoch udržiavaná ako hodnota prítomná alebo neexistujúca – teda ideálny predpoklad na boolean hodnotu – v iných je však zapísaná vo forme dátumu.

Tabuľka 6.1: Tabuľka svadobných atribútov entity Záznam.

				rozsiahlosť	
entita	**atribút**	**dát. typ**	**početnosť**	**avg**	**max**
Záznam	N. ohlášky	text	–	–	–
Záznam	datum sňatku	dátum	100.00%	9.82	10
Záznam	neurčené č. p. N.	číslo	–	–	–
Záznam	obec[1]	text	58.38%	17.18	44
Záznam	věk ženicha roky	číslo	31.73%	2.00	2
Záznam	věk ženicha měsíce	číslo	9.39%	1.14	2
Záznam	věk ženicha dny	číslo	0.25%	1.00	1
Záznam	věk nevěsty roky	číslo	31.47%	2.00	2
Záznam	věk nevěsty měsíce	číslo	9.39%	1.14	2
Záznam	věk nevěsty dny	číslo	0.51%	2.00	2
Záznam	zplnoletnění ženicha	boolean	4.57%	1.00	1
Záznam	zplnoletnění nevěsty	číslo[2]	13.71%	1.50	10
Záznam	stupeň příbuznosti	číslo[3]	1.02%	2.75	6
Záznam	podpisy	boolean	72.84%	1.00	1
Záznam	poznámky	text	13.45%	69.15	218

6.2.2 Potencionálne samostatné dáta

Osoba, ktorá je kandidátom na samostatnú množinu je *oddávající*, ale i na neho sa vzťahujú už bližšie popísané problémy s chybovosťou a nastáva tu presne rovnaká dilema ako u *zaopatřovatela* a *pohřbívajícího* v kapitole 5.2.2. I tentoraz v tomto dátovom slovníku ponecháme tieto atribúty pridelené entite *Record*, čo však po opakovanom prečítaní analýzy tohto problému nebráni prehodnoteniu implementačnej stratégie.

[1] ku ktorej svadba náleží
[2] alebo dátum
[3] alebo ich zoznam

Tabuľka 6.2: Tabuľka potencionálne samostatných svadobných atribútov entity Záznam.

				rozsiahlosť	
entita	**atribút**	**dát. typ**	**početnosť**	**avg**	**max**
Záznam	oddávající jméno	text	85.53%	12.85	39
Záznam	oddávající příjmení	text	85.79%	7.60	17
Záznam	oddávající titul	text	71.07%	13.55	85

6.3 Osoba

Po atribútoch o udalosti sú ostatné informácie o svadbe v podstate len záznamami o osobách so svadbou úzko spojených. Viacero z týchto postáv je síce možné spojiť do niekoľkých skupín, avšak skupiny, alebo inak povedané postavenia osôb sú rôzne, a tým sú aj zachované informácie iné. Skupiny môžeme prirodzene rozdeliť na mladomanželov, obsahujúcich entity ženícha a nevesty. Rodinu, ktorá obsahuje otca ženícha, matku ženícha, otca matky ženícha, otca nevesty, matku nevesty, otca matky nevesty. Svedkov, ktorý obsahujú 4 osoby s identickou štruktúrou, a nakoniec aj špeciálne osoby kam zaraďujeme rečníka, starú svadby a družbu.

6.3.1 Mladomanželia

Mladomanželia sú najdôležitejšími a najšpecifickejšími osobami a tak sú uvedení samostatne. Oproti sebe však majú obaja takmer rovnaké atribúty a tak za nich uvedieme len jednu reprezentatívnu tabuľku 6.3. To, čo ich oddeľuje, sú 4 „vdovecké“ atribúty ktoré slúžia ako identifikácia nevesty v prípade, že je vdova.

Tabuľka 6.3: Tabuľka svadobných atribútov osôb ženich (Ž) a nevěsta (N).

				rozsiahlosť	
entita	**atribút**	**dát. typ**	**početnosť**	**avg**	**max**
Ž/N	jméno	text	99.75%	6.77	21
Ž/N	příjmení	text	59.01%	6.69	19
Ž/N	vdovec/va	boolean	37.31%	1.00	1
Ž/N	manželský/á	boolean	31.60%	1.00	1
Ž/N	povolání	text	24.75%	11.63	76
Ž/N	obec	text	55.71%	12.34	34
Ž/N	ulice	text	3.05%	17.75	29
Ž/N	č. p.	číslo	31.22%	2.03	3
Ž/N	vyznání	boolean	34.47%	1.00	1
N	vdova po jméno	text	9.39%	7.24	12
N	vdova po příjmení	text	9.90%	8.03	23
N	vdova po povolání	text	1.78%	8.86	16
N	vdova po obec	text	7.87%	11.00	28

6.3.2 Rodina

Ďalšiu skupinu entít atribútmi a štruktúrou vychádzajúcich z jedného vzoru môžeme vzhľadom na postavenie osôb pomenovať rodina. Tieto dáta nie sú ničím špecifické a z hľadiska implementácie databázy sú štruktúrou takmer vzorovým príkladom pre agregáciu do jednej entity. V agregovanej tabuľke sú síce uvedené spriemerované hodnoty, ale je vhodné zaznamenať informáciu, že prítomnosť osoby a vyplnenosť jej atribútov lineárne klesala s jej dôležitosťou od cca. 80% pri menných atribútoch otca nevesty (ako mužského zástupcu jednej rodiny), až k cca 7% pri otcovi matky ženícha. Mnoho atribútov bolo taktiež absolútne chýbajúcich a tak sme hodnotu pre početnosť vzhľadom na počet záznamov zvolili ako 1‰.

Tabuľka 6.4: Tabuľka svadobných atribútov osôb otec ženicha (OZ), matka ženicha (OM), otec matky ženicha (OMZ), otec nevěsty (ON), matka nevěsty (MN) a otec matky nevěsty (OMN). ∀ značí účasť vo všetkých entitách

				rozsiahlosť	
entita	**atribút**	**dát. typ**	**početnosť**	**avg**	**max**
∀	mrtev	boolean	22.46%	1.00	1
∀	jméno	text	34.48%	6.69	13
∀	příjmení	text	32.49%	7.15	12
∀	povolání	text	10.74%	9.02	29
∀	obec[4]	text	20.01%	12.83	65
∀	ulice	text	–	–	–
∀	č. p.	číslo	–	–	–
OZ/OM/ON/MN	vyznání	boolean	29.82%	1.00	1
OZ/OM/ON/MN	datum narození	dátum	–	–	–

6.3.3 Špeciálne osoby

Rečník, stará svadby a družba majú rovnakú dvojicu atribútov a jeden neznámy atribút a tak môžu byť pri implementácii agregované do jednej entity. Kľuč ich tretieho atribútu je ich názov a je to pravdepodobne meno, alebo iná forma identifikácie osoby. Svedkovia si svojou štruktúrou vyžadujú špeciálnu entitu, v našom dátovom slovníku ich však uvádzame v nasledujúcej tabuľke 6.5 s ostatnými špeciálnymi osobami nakoľko taktiež obsahujú spomenutú dvojicu atribútov. Navyše v čase tvorby tejto práce nebol spracovaný ani jeden záznam obsahujúci rečníka, družbu, alebo starú svadby a tak nám práve svedkovia poskytnú náhľad na formu atribútov týchto osôb. V tabuľke o svedkoch sa taktiež nachádzajú 4 atribúty *jméno příbuzného*, *příjmení příbuzného*, *povolání příbuzného* a *obec příbuzného*. Tieto atribúty sú však v poskytnutých záznamoch vzájomne výlučné s podobnými už existujúcimi atribútmi. Reprezentujú len to, že je daný svedok v rodinnom vzťahu s niektorým z mladomanželov a túto informáciu už obsahuje atribút *vztah ke svědek* a tak sme ich zlúčili.

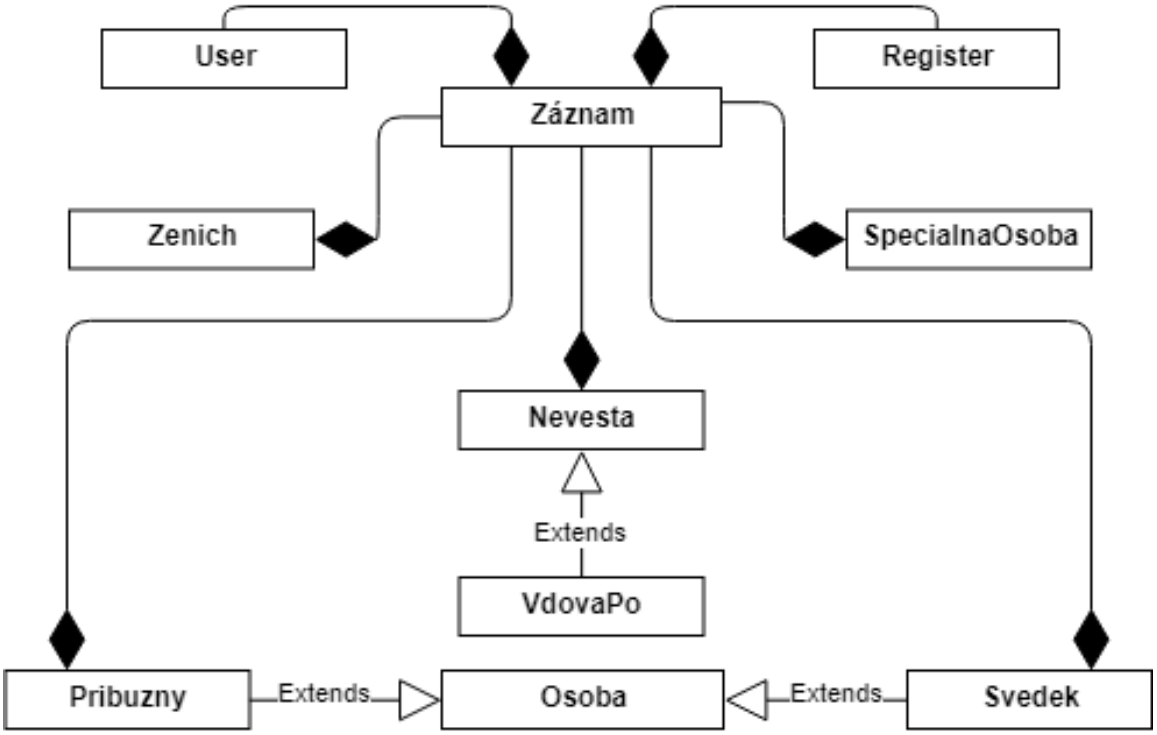
[4] alebo bydliště

Tabuľka 6.5: Tabuľka svadobných atribútov špeciálnych osôb – svědek (S), rečník (R), stará svadby (SS) a družba (D). ∀ značí účasť vo všetkých entitách.

				rozsiahlosť	
entita	**atribút**	**dát. typ**	**početnosť**	**avg**	**max**
svědek	jméno	text	60.03%	6.31	22
svědek	příjmení	text	59.41%	7.66	24
∀	povolání	text	16.75%	7.33	29
∀	obec[5]	text	35.72%	11.80	22
svědek	ulice	text	0.32	9.00	15
svědek	č. p.	číslo	0.32	2.00	9
svědek	vztah ke svědek	text	1.52	4.75	12
D/R/SS	neznámé (jméno)	text	–	–	–

6.4 Pridané dáta

Spracovávanie dát prebieha rovnakým spôsobom a rovnakým tímom a tak sa tu stretávame s identickou štruktúrou, delením a rovnakým počtom pridaných dát, konkrétne *nesamostatnými* a *samostatnými*, pričom štatistiky sú tiež takmer identické. Z tohto dôvodu nie je potrebné vytvárať novú tabuľku, absolútne vyhovujú tabuľky 5.4 a 5.5.



Obr. 6.3: Objektovo orientovaný svadobných dát.

[5] alebo bydliště

6.5 Konceptuálna schéma

Rovnakým postupom, ako pri úmrtných dátach, sme aj pre sobášne záznamy vytvorili dátový slovník, konceptuálnu a logickú schému databázy.

6.5.1 Objektovo orientovaný model sobášnych dát

Pre objektovú databázu platia taktiež rovnaké podmienky a tak popisujeme stratégiu modelovania pri ER diagrame. Plnú implementáciu objektovo orientovaného modelu je možné vidieť v obrázku 7.4.

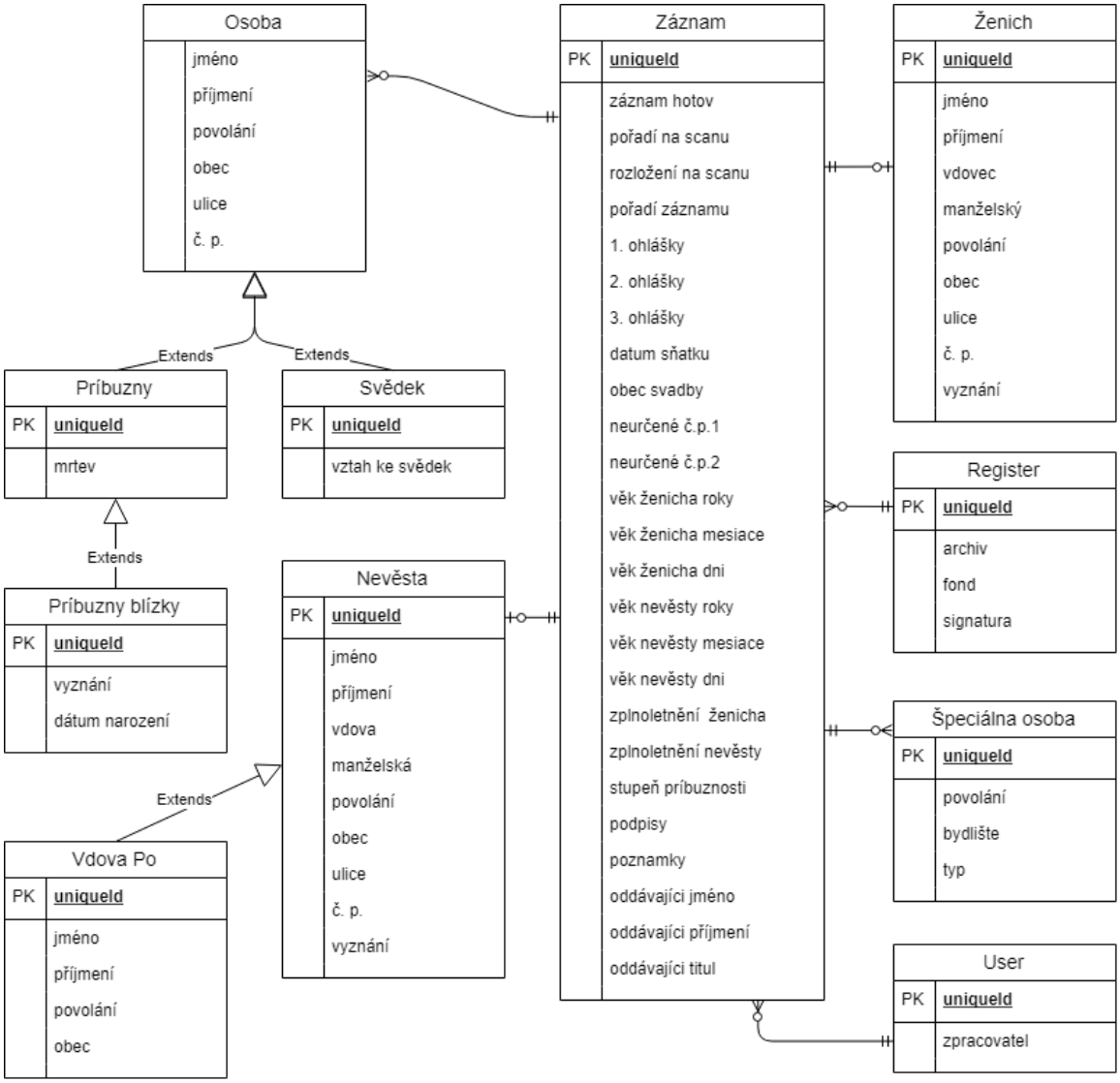
6.5.2 ER diagram sobášnych dát

Na obrázku 6.4 je zobrazený ER diagram pre sadu matričných dát o svadbách. Nakoľko sú sobášne dáta štruktúrou podobné úmrtným, je i proces vytvárania podobný. Taktiež sa tu stretávame s dedičnosťou osôb, teraz dokonca v ešte zložitejšej podobe. Špecifickými osobami, ktoré tentoraz do tejto hierarchie neboli zaradené z rovnakého dôvodu, ako špeciálna osoba pri úmrtných dátach, sú ženích, nevesta a tiež špeciálna osoba.

Ženích a nevesta majú takmer identickú štruktúru a atribúty, a tak sú z tohto pohľadu ideálnymi kandidátmi na agregáciu. Na základe analýzy reprezentatívnej vzorky a vypracovania dátového slovníka so štatistikami sa však naskytá iný pohľad. Z tabuľky 6.3 vieme, že sa atribút *jméno* nachádza v 99.75% záznamoch o mladomanželoch, a teda môžeme s trochou nadhľadu povedať, že sa entity ženícha a nevesty budú nachádzať v každom zázname. Entitná množina *Záznam* reprezentuje danú udalosť a má teda účasť 100%, a zároveň je aj maximom – žiadna neagregovaná entitná množina (pre jednoduchosť a čitateľnosť ďalej v tejto sekcii len tabuľka) nemôže obsahovať viacero entít, a vzhľadom na štruktúru dát nemôže žiadna entita, s výnimkou entít z tabuliek *User* a *Register*, existovať bez priameho alebo nepriameho spojenia so Záznamom, čo je jasné z obrázku entít 6.1. Ak by sme sa teda rozhodli pre agregáciu dvoch tabuliek, ktoré majú takmer rovnakú početnosť ako tá s maximálnou početnosťou, vznikla by tabuľka, ktorá by obsahovala takmer dvojnásobný počet záznamov oproti tej s doteraz najväčším počtom entít. Ostatné tabuľky by mali dokonca len zlomkové počty záznamov. Ženích a nevesta sú navyše prirodzene najdôležitejšími osobami v sobášnych záznamoch a teda sa dá očakávať aj najväčší počet prístupov do ich tabuliek.

Očakávame teda veľmi vysokú (pravdepodobne najvyššiu) frekvenciu dotazov na potenciálnu tabuľku, ktorá udržuje s veľkým odstupom najväčší počet entít, ktorých atribúty sú navyše prevažne textového charakteru (a ich porovnávanie je teda oveľa zložitejšie, ako pri ostatných dátových typoch), a ktorá neobsahuje ani jeden číselný záznam vhodný pre index. Navyše by bolo nevyhnutné pridať k nej ďalší, diskriminačný atribút, určujúci, či sa jedná o nevestu alebo o ženícha. Vyhľadávanie, či už len jednoduchá extrakcia dát alebo v princípe každá operácia v tejto tabuľke, by bola časovo rádovo náročnejšia ako v iných tabuľkách. V mnohých prípadoch by to navyše spôsoboval sčasti aj prechod nepotrebnými dátami (ženícha, ak vyhľadávame nevestu a naopak).

Na základe týchto informácii a zistení sme sa rozhodli nepodľahnúť intuitívnej voľbe agregácie týchto dvoch, i keď atribútmi a štruktúrou takmer identických tabuliek. Toto rozhodnutie ďalej podporuje aj fakt, že sa v tomto koncepte databázy nenachádza veľké množstvo tabuliek a teda nie je žiaden dôvod predpokladať problémy či neúmernú časovú náročnosť pri spájaní tabuliek v dopytoch. Tento stav nijak negatívne neovplyvní ani samostatná implementácia tabuliek ženícha a nevesty.



Obr. 6.4: ERD svadobných dát.

Ďalej sme sa rozhodli modelovať „vdovecké“ atribúty nevesty ako entitu rozširujúcu nevestu, nakoľko je ich početnosť nízka a teda len zbytočne zaťažujú nevestu. Reálna implementácia však záleží na možnostiach konkrétneho DBMS a ako vhodná možnosť sa javí modelovanie týchto atribútov ako slabej entity.

Kapitola 7

Implementácia a testovanie

7.1 Porovnanie implementácií

V nasledujúcich sekciách bude popísaný proces a výsledok implementácie navrhnutých konceptuálnych modelov. Následne porovnáme náročnosť, rozsiahlosť, využité a dostupné prostriedky a konštrukcie, ale aj intuitívnosť procesu implementácie.

Ako bolo popísané a demonštrované v sekcii 5.2.2, alebo v príklade 5.2, na mene „františku“, rovnaká hodnota nemusí byť nevyhnutne rovnako zapísaná. Nakoľko sú mená kľúčovou časťou záznamu, na konečnú implementáciu našich databáz bola vznesená požiadavka na navrhnutie normalizácie (krstných) mien. Normalizácia mala taktiež zahŕňať možnosť viacerých mien pre jednu osobu.

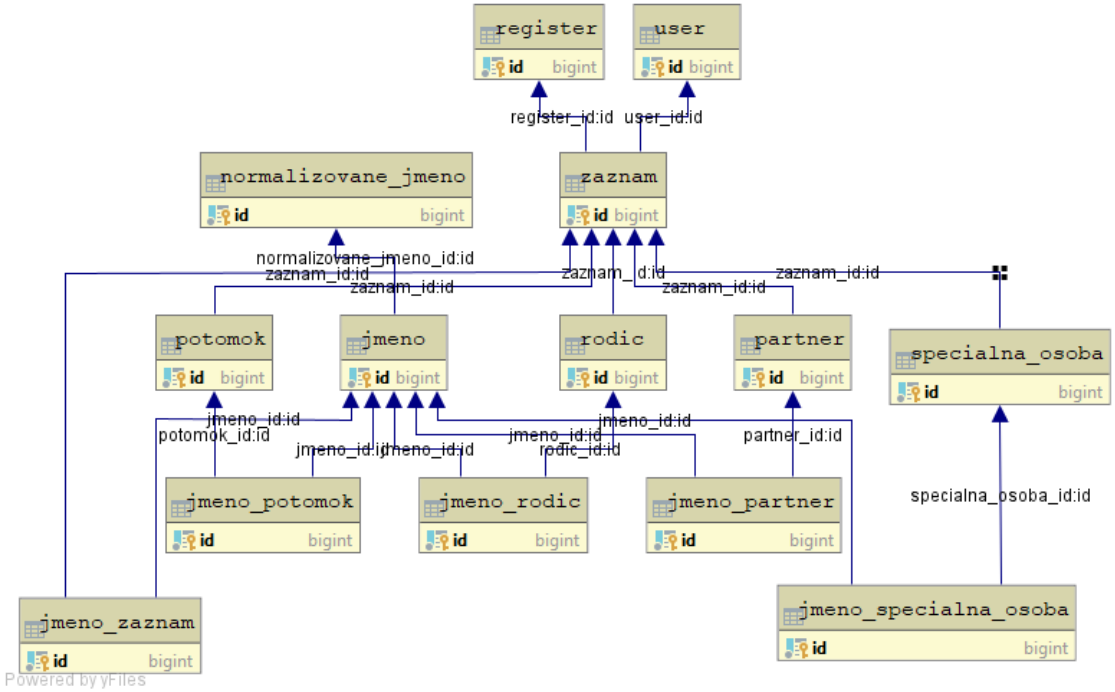
Normalizácia bola následne implementovaná pre oba druhy databáz rovnakým spôsobom. Vytvorili sme tabuľku/triedu (ďalej len tabuľku) *Jmeno* s atribútmi *jmeno* a *poradie*. Z každej tabuľky, kde bol atribút jmeno, sme ho odstránili, nahradili odkazom na tabuľku Jmeno a do nej presunuli hodnotu atribútu. Keďže vzhľadom na potencionálne väčší počet mien takto vznikol M:N vzťah, v relačnej databáze sme boli nútení vytvoriť asociatívne tabuľky pre každú entitu so vzťahom s menom. V prípade objektového návrhu sme atribút jmeno nastavili jednoducho ako zoznam tried Jmeno a nemuseli ďalej riešiť ako je tento vzťah v databáze uložený. Navyše nie je potrebná ani implementácia spätnej referencie z entity Jmeno. Tu vidíme značnú výhodu db4o a objektového prístupu, nakoľko sme v relačnom návrhu museli pridať pre každé meno ďalšiu asociatívnu tabuľku, ktorá musí byť ručne navrhnutá, vytvorená i spravovaná. Priame ukladanie objektov s možnosťou neatomických atribútov a schopnosť db4o vyhľadania vzťahov i bez spätných odkazov úplne eliminuje túto časť a problémovú architektúru asociatívnych tabuliek relačných databáz.

Poslednou časťou normalizácie mien je tabuľka*Normalizovane_jmeno* (trieda *NormalizovaneJmeno*). V tejto tabuľke sú uložené normalizované formy mien a enity Jmeno sa na ňu odkazujú podľa toho, k akému normalizovanému menu patria. V našom generátore testovacej dátovej sady sa NormalizovaneJmeno a Jmeno napĺňajú ako prvé a podľa príkladu vzorky dát sme nastavili pomer entít týchto tabuliek na 1:3. Záznamy sú potom k menám z vygenerovanej množiny priraďované náhodne.

7.1.1 Fyzická schéma relačných databáz

Čo sa týka úmrtných záznamov, pri implementácii hierarchie dedičnosti osôb sme sa rozhodli pre kombinovanú stratégiu, nakoľko ani jedna z definovaných stratégii v sekcii 3.3.1 nie je sama osebe ideálnou možnosťou. Tabuľka pre entitu je síce jediné normalizované

riešenie, ale fakt, že osoba otec matky neobsahuje žiaden atribút, ktorý by ju odlišoval od základnej tabuľky komplikuje celý návrh a kompromituje konzistenciu dát. Stratégia TPH by vzhľadom na to, že samotné atribúty sú pri osobách riedke, vytvorila extrémne riedke záznamy. TPC by zas vytvorila viacero veľmi nízko obsadených tabuliek. Preto sme zvolili kombináciu TPC a TPH a vytvorili tri tabuľky *Rodic*, *Partner* a *potomok* z ktorých každá agreguje logicky si blízke osoby. Je vhodné poznamenať, že výber čo najefektívnejších konštrukcii a posudzovanie „ceny“ jednotlivých databázových operácii, bol do veľkej miery taktiež ovplyvnený brilantným článkom *Cost of a join* [13], ktorý skúma, aký vplyv má v PostgreSQL spájanie tabuliek pri operáciách s dátami v databáze a ako tieto hodnoty ovplyvňujú faktory ako indexovanie, počet stĺpcov, či počet záznamov. Výslednú fyzickú schému môžeme vidieť v diagrame 7.1.



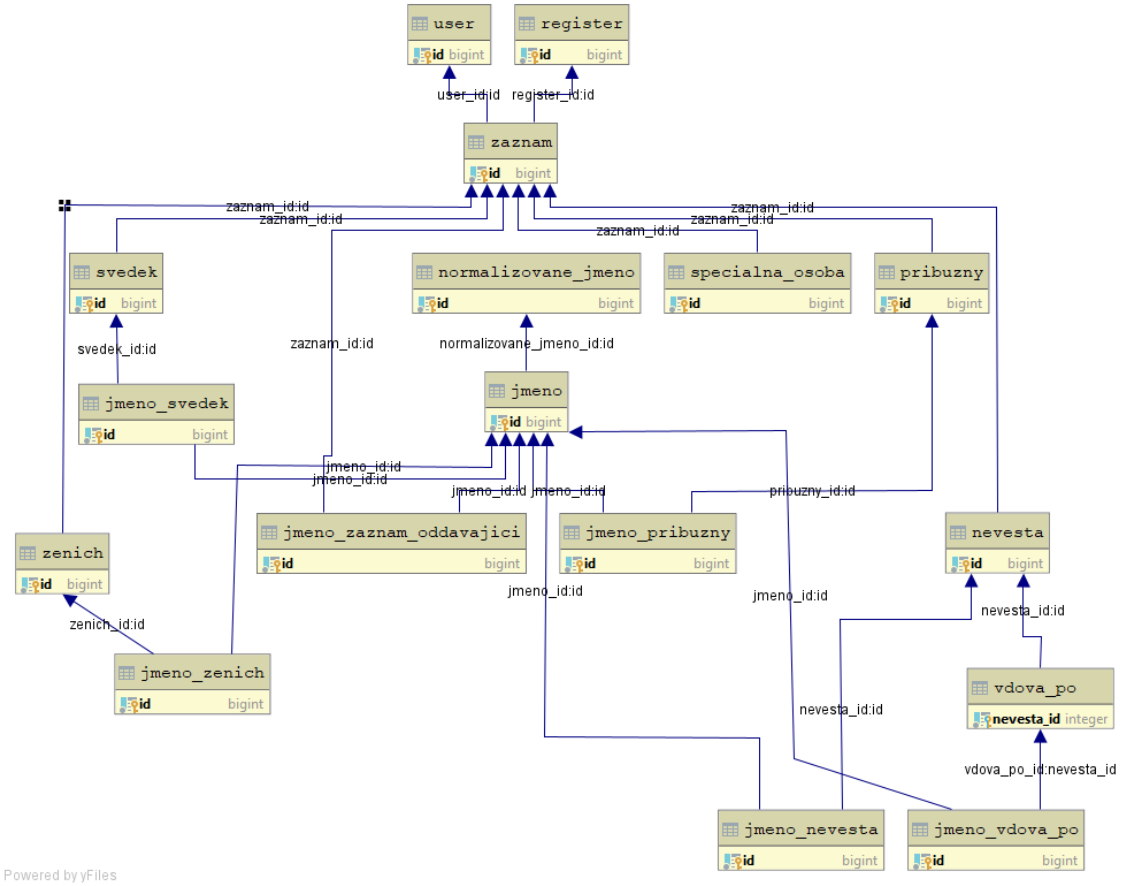
Obr. 7.1: Diagram fyzickej schémy PostgreSQL databázy dát o úmrtiach.

Čo sa týka sobášnych záznamov, vzhľadom na vyšší počet osôb a tabuliek sme sa tentokrát rozhodli rodinných príslušníkov hierarchie dedičnosti agregovať do jednej tabuľky *Príbuzný* podľa stratégie TPC. Tabuľku *Vdova_po* sme modelovali ako slabú entitu entity nevesta. Svedka sme implementovali ako samostatnú tabuľku vzhľadom na to, že sa pri jednom zázname môžu nachádzať až štyria rôzni svedkovia. Agregáciou rodinných príslušníkov taktiež klesol počet tabuliek a podľa už spomínaného článku o cene spájania tabuliek pri dopyte táto konštrukcia nijak výrazne neovplyvní výkon našej databázy. Výslednú implementáciu môžeme vidieť v diagrame 7.2.

Čo sa týka inicializácie databáz a samotného vytvorenia tabuliek, vytvorili sme Laravel[1] projekty UmrtneZaznamy a SvadobneZaznamy, priložené na CD k tejto práci, ktoré sa pripájajú na PostgreSQL databázu a sú schopné pomocou systému nami vytvorených migrácií inicializovať databázovú schému a vytvoriť tabuľky. Pre čo najvyššiu efektivitu

[1]open-source PHP framework pre webové aplikácie

a rýchlosť databázy sme pri definovaní atribútov použili znalosti z dátového slovníka a obmedzili ich maximálny rozsah podľa maximálnej rozsiahlosti. Pri textových atribútoch sme k nej prirátali pre každú stovku jej desatinu a to zaokrúhlili k najbližšej desiatke (teda napr. pre atribút s maximálnou rozsiahlosťou 29 to bude $29 + (100/10) = 39 \approx 40$). Pri číselných atribútoch sme obmedzovali počet bytov a signedness.[2] Takisto sme museli definovať primárne a cudzie kľúče, asociatívne tabuľky, unikátnosť a povinnosť, pričom objektový model tieto vlastnosti spravuje sám a výrazne tak zjednodušuje prácu programátora. Na druhú stranu nám však takáto blízka kontrola umožňuje pevnú kontrolu nad databázou a teda aj jej efektívnejší návrh. Db4o ukladá dátové typy jazyka v ktorom funguje a je pre programátora nemožné do tohto procesu vstúpiť.

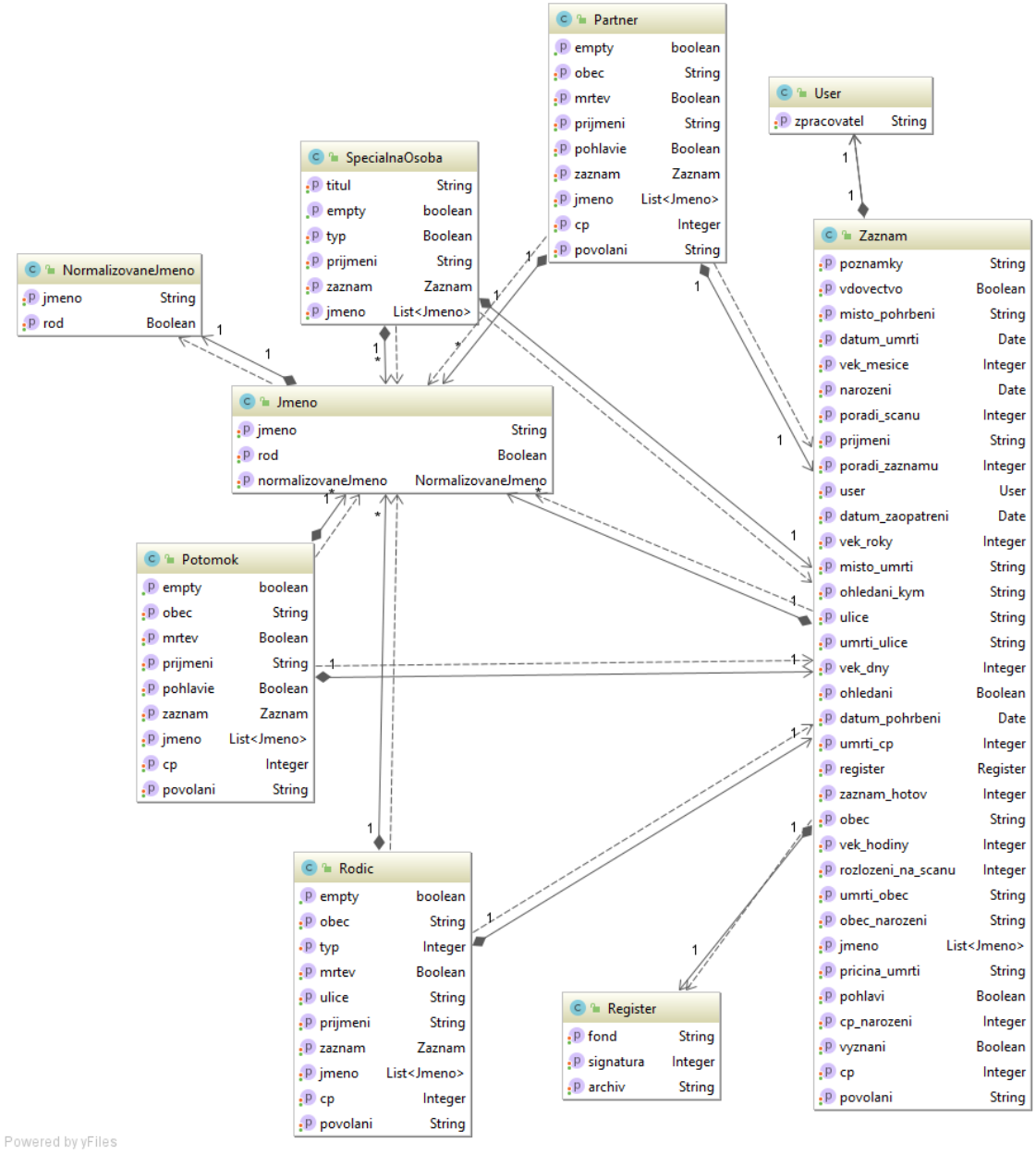


Obr. 7.2: Diagram fyzickej schémy PostgreSQL databázy dát o svadbách.

7.1.2 Fyzická schéma objektových databáz

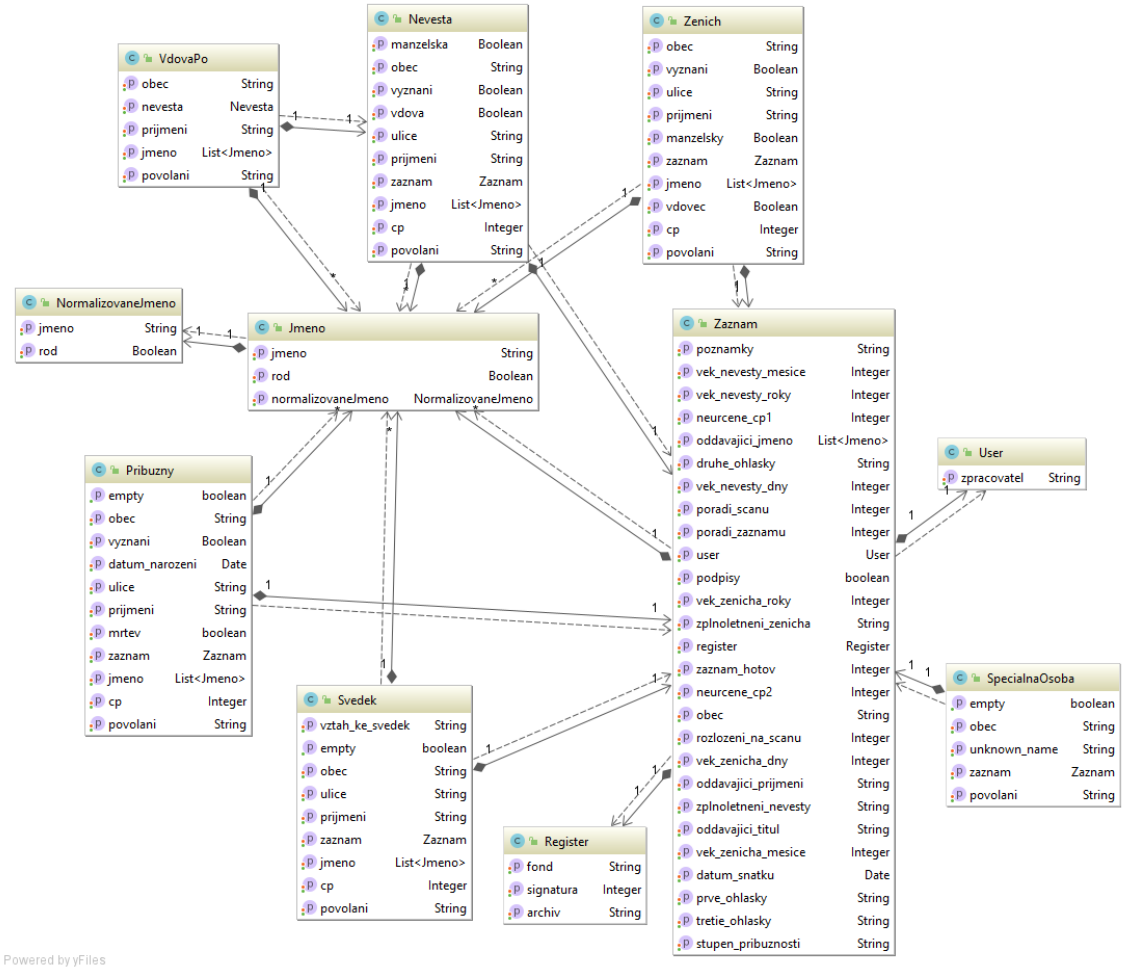
Databáza db4o je blízko prepojená s programovacím jazykom, do ktorého je vložená rovnakým spôsobom ako knižnica a ktorému poskytuje rozhranie na prácu s perzistentnou vrstvou. V našom prípade sme zvolili jazyk Java. Pri objektových databázach je otázka implementácie dedičnosti oveľa jednoduchšia, nakoľko je vďaka objektovému prístupu možné jednoducho ukladať triedy a z programátorského hľadiska sa nezaoberať procesmi databázy

[2] V českom jazyku znaménkovost – vlastnosť čísla udávajúca schopnosť nadobudnúť záporné hodnoty.



Obr. 7.3: Diagram tried záznamov o úmrtiach ukladaných databázou db4o.

na pozadí. Dedičnosť je však skôr tranzientná ako perzistentná konštrukcia a objekty sú definitívne ukladané ako ucelené prvky a nie časti väčšieho celku. Takisto má zmysel zvažovať početnosť výskytov objektov jednotlivých tried rovnako ako pri relačnom modeli. Z týchto dôvodov je štruktúra tried úmrtných (diagram 7.3) a sobášnych (diagram 7.4) záznamov veľmi podobná štruktúre tabuliek v relačnej databáze. Nakoľko je db4o vstavaná databáza nebolo ju potrebné nijak inicializovať ani vytvárať iný počiatočný skript, je však potrebné definovať triedy a ich atribúty a metódy. Ako bolo spomínané, db4o je taktiež zero–admin databáza a tak sa s ňou bez konkrétneho programovacieho jazyka nedá pracovať ani manipulovať, čo je teoreticky pozitívna vlastnosť, prakticky je to však často skôr prekážka ako



Obr. 7.4: Diagram tried záznamov o svadbách ukladaných databázou db4o.

výhoda, na ktorú sme narážali hlavne pri vytváraní štatistík časových náročností pre porovnanie databáz. Ďalšou teoretickou výhodou, ktorá však v praxi môže spôsobovať problémy je nedostupnosť OID. Programátora to síce odbremeňuje od repetitívnej nekreatívnej práce pri vytváraní a udržovaní primárneho kľúča (čo je však v moderných RDBMS už taktiež zvládnuté takmer úplne automaticky) a predchádza i potenciálnym chybám. Berie to však jednoduchú možnosť ako objekt identifikovať a opakovane k nemu pristupovať bez ďalších znalostí o ňom, čo môže mať potenciálne rozsiahle negatívne následky hlavne v prípade, ak medzi sebou komunikuje viacero častí aplikácie a je potrebné komunikačné správy čo najviac zredukovať.

Na základe práce s db4o na návrhu databáz pre tento projekt však môžeme tvrdiť, že z implementačného hľadiska je objektový prístup k databáze pre programátorov veľmi priateľská a intuitívna technológia, ktorá je pre projekt ľahko a stále dostupná, rýchla na vytvorenie a nasadenie, rieši alebo priamo predchádza množstvu spomenutých problémov a ťažkostí, zbavuje sa nesúladu medzi aplikačnou a databázovou vrstvou, jednoduchšie sa modeluje a poskytuje projektu ako celku jednotnejšiu a ľahšie pochopiteľnú štruktúru, čo všetko platí aj pre databázy úmrtných a svadobných dát.

7.2 Vytvorenie testovacej dátovej sady

Vytváranie dátovej sady bolo mierne načrtnuté už v predchádzajúcej sekcii. Pre PostgreSQL sme na to vytvorili dva spomínané projekty, ktoré poskytujú príkazy do príkazového riadku na spustenie aplikácie (*php artisan serve*) a vytvorenie databázovej schémy (*php artisan migrate*). Na naplnenie databázy slúži REST API[3] POST metóda, dostupná na adrese *localhost:cisloPortu/api/db/seed*, kde štandardné nastavenie hodnoty *localhost* je 127.0.0.1 a hodnoty *cisloPortu* je 8000. V tele požiadavky zaslanej na na server je možné uviesť 2 atribúty ovplyvňujúce napĺňanie databázy, *recordCount* ktorý udáva počet vytvorených záznamov, a *nameCount* ktorý udáva počet vytvorených mien. Nevyhnutné je však vytvorenie PostgreSQL databázy, ku ktorej sa bude aplikácia pripájať. Následne je potrebné zo súboru *.env.example* v koreňovej zložke projektu spraviť kópiu, nastaviť DB premenné (obsahujúce ukážkové hodnoty) na hodnoty fungujúcej databázy a súbor uložiť znova v koreňovej zložke projektu ako *.env* súbor (bez mena). Správna hodnota *DB_CONNECTION* je pre PostgreSQL *pgsql*.

Pre db4o sme vytvorili jeden Java projekt ObjDatabases. V tomto projekte nie sú nevyhnutné žiadne ďalšie nastavenie. Napĺňanie databázy sa spúšťa spustením hlavnej triedy *Main*, s troma číselnými parametrami z ktorých prvý je ekvivalentný nameCount, druhý recordCount a tretí určuje, ktorá databáza sa bude napĺňať (0 pre úmrtnú, 1 pre svadobnú).

7.2.1 Vytváranie mien

Napĺňanie databáz prebieha po historických záznamoch. Kvôli vzťahom medzi entitami (v popise budeme využívať relačnú terminológiu) sa naplnia pre jeden záznam všetky tabuľky, následne sa pokračuje cyklicky ďalšími záznamami. Prvým, už popísaným krokom, je však vytvorenie mien. Špeciálnym prípadom sú tabuľky User a Register, nakoľko sa viažu na množstvo záznamov. Tu sme zvolili stratégiu: *ak (x mod 300 = 0) generuj User, Register*, kde x je počet uložených záznamov a novo spracované záznamy sa stále viažu na posledne vygenerované entity.

7.2.2 Vytváranie atribútov

Generovanie a ukladanie jednotlivých entít potom prebieha nasledovne. Pre každý atribút sa vygeneruje náhodné číslo z intervalu 0 – 100. Ak vygenerované číslo spadá do intervalu 0 – početnosťou výskytu atribútu z nášho dátového slovníka, atribút sa vygeneruje podľa dátového typu a priemernej rozsiahlosti taktiež uvedených v dátovom slovníku. V prípade textu je to náhodne generovaný reťazec alfanumerických znakov, v prípade čísla je to náhodné celé číslo, v prípade boolean je to náhodne zvolená boolean hodnota a v prípade dátumu je to aktuálny dátum. V prípade, že sa kvôli nízkej početnosti nevytvoril ani jeden atribút je entita zahodená.

7.2.3 Zhlukovanie atribútov

Atribúty však majú v záznamoch tendenciu zhlukovať sa a aj keď to nie je univerzálne pravidlo, je to faktor výrazne ovplyvňujúci výsledný počet entít a hustotu vyplnenia ich atribútov, preto sme sa rozhodli reprezentovať ho i v našom generátore nasledujúcim spôsobom. V každej entite sme zvolili najčastejšie sa vyskytujúci atribút ako smerodajný. Ak sa

[3]REST – Representational state transfer je cesta, ako jednoducho vytvoriť, čítať alebo zmazať informácie zo serveru pomocou jednoduchých HTTP volaní.

ten v entite bude vyskytovať, entita logicky musí existovať, a tak sme početnosť ostatných atribútov znásobili a teda zvýšili pravdepodobnostnú hodnotu ich výskytu. Ak sa v entite nevyskytne, entita nemusí vzniknúť, a pravdepodobnosť výskytu musí byť o rovnakú pravdepodobnostnú hodnotu znížená.

7.2.4 Počítanie pravdepodobnosti výskytu

Pre výpočet správnych pravdepodobnostných hodnôt vzhľadom na početnosť smerodajného atribútu a udržanie rovnakej celkovej pravdepodobnosti výskytu sme pre každý atribút použili nasledujúci postup. Nazvime početnosť výskytu atribútu **P** a faktor jeho násobného zvýšenia **X**, smerodajný atribút **SA**, početnosť smerodajného atribútu **PSA** a jeho doplnok $\mathbf{PSA^c}$.

- Ak je $PSA \leq 50\%$ a SA bol vytvorený, existencia atribútu bude vygenerovaná na základe početnosti $P \times X$.
- Ak je $PSA \leq 50\%$ a SA nebol vytvorený, existencia atribútu bude vygenerovaná na základe početnosti $P - ((PSA \div PSA^c) \times (P \times (X - 1)))$
- Ak je $PSA > 50\%$ a SA nebol vytvorený, existencia atribútu bude vygenerovaná na základe početnosti $P \div X$.
- Ak je $PSA > 50\%$ a SA bol vytvorený, existencia atribútu bude vygenerovaná na základe početnosti $P + ((PSA^c \div PSA) \times (P \div X))$

Kombináciou popísaných techník a investovaním značného času a úsilia do analýzy vzorky sa nám podarilo vytvoriť generátor testovacích dát poskytujúci sadu konvergujúcu k poskytnutej vzorke.

7.3 Testovanie výkonnosti

Testovanie výkonnosti a rýchlosti databáz je kľúčovým prvkom tejto práce a bolo vykonávané s veľkou pozornosťou a rozsiahlymi testovacími vstupmi a operáciami. Pre testovanie sme zvolili stratégiu s ohľadom na dané dátové sady a ich potenciálne využitie.

7.3.1 Testovací stroj

Všetky výkonnostné testy boli vykonané na PC s týmito špecifikáciami:

- Processor Intel(R) Core(TM) i7-4600U CPU @ 2.10GHz, 2694 Mhz, 2 Core(s), 4 Logical Processor(s)
- Installed Physical Memory (RAM) 8.00 GB DDR3
- OS Microsoft Windows 10 Pro
- TOSHIBA 256GB SSD SATA M.2

Testy používali nasledujúce databázy:

- PostgreSQL 11 for Windows x64
- Db4o 7.4 for Java

7.3.2 Stratégia testovania

V rámci porovnania výkonu DBMS sme testovali zápis, čítanie, aktualizáciu a mazanie dát. Pri čítaní sme navyše rozdelili testy na jednoduché, zahrňujúce len samostatné entity, a štruktúrované, pri ktorom sme sa dopytovali na entity spolu s rôznym počtom súvisiacich entít. Záznamy pre čítanie sme vyhľadávali podľa všetkých druhov využívaných dátových typov. Testy sme vykonávali nad rozdielnym počtom záznamov, každý atomický test sme opakovali viackrát (3+) a získané hodnoty sme spriemerovali. Pre operácie zápisu a mazania dát sme operáciu vykonali nad každou entitou a následne spriemerovali tieto časy a získali tak výsledný čas pre databázu. Tento proces sme opakovali následne s väčším počtom vykonaných operácii. Pre čítanie a aktualizáciu dát sme taktiež spriemerovali hodnoty výsledkov entít pre celé databázy. Operácie sme však nevykonávali vo vyššom počte ale na vyššom počte už existujúcich dát.

Taktiež sme sa rozhodli, že vzhľadom na zadanie témy zamerané na porovnanie ale aj kvôli principiálnej nezmyselnosti absolútnych čísel získaných na špecifickom stroji budeme uvádzať rozdiel medzi databázami vo výkone v percentách a nie v absolútnych číslach.

Čo sa týka samotného priebehu testovanie je nutné poznamenať, že db4o neposkytuje vhodné nástroje a už samotná podstata tohto systému (zero–admin, embeddable) sťažuje túto úlohu. [28] I keď boli vytvorené pokusné aplikácie na manažment databázy ako od jej vydavateľa (plugin Object Management Enterprise do IDE[4] Eclipse a MS Visual Studio 2005/2008), tak aj od verejnej komunity (ObjectManager), tieto nástroje sa kvalitou a možnosťami nikdy neblížili DBMS relačných databáz. Ukončením podpory a vývoja navyše rapídne začala rásť chybovosť a neaktuálnosť týchto aplikácii a urobila z nich len veľmi ťažko použiteľný software. Ako administračnú platformu pre databázu PostgreSQL sme používali pgAdmin, ale tiež integrované nástroje IDE PhpStorm a DataGrip.

7.3.3 Výsledky testovania

Výsledné hodnoty testovania môžeme vidieť pre úmrtné databázy v tabuľke 7.1 a pre svadobné v tabuľke 7.2.

Tabuľka 7.1: Tabuľka výsledkov výkonnostných testov nad úmrtnými dátami. Hlavička udáva počet operácii pre zápis a mazanie a počet záznamov pre ostatné operácie. Hodnoty buniek udávajú rýchlejšiu databázu (R – relačná, O – objektová) a percentuálna hodnota udáva časový rozdiel z pohľadu pomalšej databázy.

	1	**50**	**100**	**500**	**5000**
zápis	O: 4.20%	R: 4.90%	O: 14.30%	O: 43.06%	O: 47.25%
aktualizácia	R: 6.54%	R: 7.14%	R: 8.09%	R: 14.58%	R: 149.08%
mazanie	O: 3.07%	O: 4.85%	O: 9.19%	O: 54.78%	O: 531.14%
jednoduché čítanie	O: 18.96%	O: 6.31%	R: 4.32%	R: 27.40%	R: 221.17%
štruktúrované čítanie	O: 7.58%	R: 12.88%	R: 19.41%	R: 103.39%	R: 517.64%

V oboch dátových sadách víťazí PostgreSQL k db4o v pomere 13:12. To je síce malý rozdiel, dôležité však je, že db4o víťazí hlavne v kategóriách s nižším počtom operácii a záznamov. Taktiež môžme zhodnotiť, že zatiaľ čo je db4o výkonnejšia hlavne v zápise a mazaní dát, tak PostgreSQL víťazí v aktualizácii a dominuje v čítaní dát. Vzhľadom na to, že úlo-

[4]Integrated development environment – integrované vývojové prostredie

hou týchto databáz bude udržiavať historické záznamy a efektívne ich sprístupňovať, je čítanie dát oveľa dôležitejšie a taktiež bude logicky častejšie vykonávané ako ostatné operácie. Db4o dosiahla slabé výsledky hlavne v prípadoch textového vyhľadávania vo väčšom počte záznamov, a to je výrazné negatívum. Na základe týchto faktov môžeme zhodnotiť, že je PostgreSQL efektívnejšou a výkonnejšou databázou pre úmrtné a svadobné záznamy.

Tabuľka 7.2: Tabuľka výsledkov výkonnostných testov nad svadobnými dátami. Hlavička udáva počet operácii pre zápis a mazanie a počet záznamov pre ostatné operácie. Hodnoty buniek udávajú rýchlejšiu databázu (R – relačná, O – objektová) a percentuálna hodnota udáva časový rozdiel z pohľadu pomalšej databázy.

	1	**50**	**100**	**500**	**5000**
zápis	O: 3.77%	O: 3.52%	O: 7.91%	O: 38.11%	O: 49.52%
aktualizácia	R: 0.25%	R: 4.23%	R: 7.84%	R: 19.41%	R: 186.19%
mazanie	R: 4.35%	O: 7.72%	O: 13.04%	O: 59.88%	O: 581.26%
jednoduché čítanie	O: 8.66%	O: 1.32%	R: 3.06%	R: 29.18%	R: 273.97%
štruktúrované čítanie	O: 6.16%	R: 8.44%	R: 21.67%	R: 95.14%	R: 490.36%

Kapitola 8

Záver

V tejto práci sa nám úspešne podarilo zanalyzovať štruktúru historických záznamov o úmrtiach a svadbách, navrhnúť štruktúru pre ich ukladanie, vytvoriť modely dát pre relačnú aj objektovú databázu, na základe nich implementovať obe databázy a následne pomocou vytvorenej dátovej sady aj otestovať ich rýchlosť a porovnať ich implementáciu, výkonnosť a ich teoretický základ. Dôležitými výsledkami sú hlavne konceptuálne návrhy ktoré demonštrujú ako jednotlivé prístupy, tak aj podobu dát v jednej spoločnej forme, a výsledky testovania, ktoré sú najdôležitejším faktorom hodnotenia databáz. Na základe jednotlivých porovnaní môžeme tvrdiť, že i keď má objektový model databáz svoje výhody, relačný model je za momentálnych podmienok vhodnejším typom databáz na spracovanie poskytnutých dát. PostgreSQL poskytuje oproti db4o rýchlejší prístup k dátam, pevnejší teoretický základ a silnejšiu konzistenciu dát, lepšiu perspektívu podpory a vývoja databázového systému do budúcnosti ale aj omnoho lepšiu podporu a dostupnosť informácii v súčasnosti. Db4o, zástupca objektového prístupu, však predstavuje prirodzenejšie, i keď menej špecifikované modelovanie, väčšiu integráciu s programovacími jazykmi a napriek zastaveniu vývoja relatívne rýchle databázové operácie, a preto by bolo vhodné v budúcnosti otestovanie iných, v prípade možnosti i komerčných a hlavne stále podporovaných objektových databázových systémov.

Literatúra

[1] Abiteboul, S.; Hull, R. B.; Vianu, V.: *Foundations of Databases.* Addison-Wesley, 1995, ISBN 0-201-53771-0, 192–199 s.

[2] Bagui, S.: *Achievements and Weaknesses of Object-Oriented Databases.* Journal of Object Technology, *ročník 2, 2003: s. 29–41, doi:10.5381/jot.2003.2.4.c2.*

[3] Bhatia, S.: PostgreSQL vs. MySQL: [2019] Everything You Need to Know. 2019, [Online; navštívené 3.11.2018].
URL https://hackr.io/blog/postgresql-vs-mysql

[4] Bloom, M.: Data Integration Glossary. 2009, [Online; navštívené 15.1.2019].
URL https://web.archive.org/web/20090320001015/http://knowledge.fhwa.dot.gov/tam/aashto.nsf/All+Documents/4825476B2B5C687285256B1F00544258/$FILE/DIGloss.pdf

[5] Brandenburg, L.: What is a Data Dictionary? 2017, [Online; navštívené 10.11.2018].
URL https://www.bridging-the-gap.com/data-dictionary/

[6] Bryner, J.: Parents Choosing More Unusual Baby Names Now. Live Science, *2010.*
URL https://www.livescience.com/9841-parents-choosing-unusual-baby-names.html

[7] Cattell, R.; Barry, D. K.; Berler, M.; aj.: The Object Data Standard. The Morgan Kaufmann Series in Data Management Systems, Morgan Kaufmann, 2000, ISBN 1558606475.

[8] shan Chen, P. P.: The Entity-Relationship Model: Toward a Unified View of Data. ACM Transactions on Database Systems, *ročník 1, 1976: s. 9–36.*

[9] Codd, E. F.: A Relational Model of Data for Large Shared Data Banks. In: Comun. ACM, *ročník 13/6, 1970: str. 377–387.*

[10] Codd, E. F.: Does Your DBMS Run By the Rules. ComputerWorld, *1985.*

[11] Codd, E. F.: Is Your DBMS Really Relational? ComputerWorld, *1985.*

[12] Codd, E. F.: The 12 rules. University of Derby, *1985.*

[13] Davis, B.: Cost of a Join. 2018, [Online; navštívené 21.2.2019].
URL https://www.brianlikespostgres.com/cost-of-a-join.html

[14] *Ejima, K.: Showdown: MySQL 8 vs PostgreSQL 10. 2018, [Online; navštívené 3.11.2018].*
URL
https://hackernoon.com/showdown-mysql-8-vs-postgresql-10-3fe23be5c19e

[15] *Everest, G.: BASIC DATA STRUCTURE MODELS EXPLAINED WITH A COMMON EXAMPLE.* Computing Systems, *ročník 10, 1976: s. 39–46.*

[16] *Fowler, M.: Patterns of Enterprise Application Architecture. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002, ISBN 0321127420.*

[17] *Ghongade, R. S.; Pursani, P. J.: Comparison of Relational Database and Object Oriented Databases.* International Journal of Modern Trends in Engineering and Research, *ročník 01, 2014, ISSN 2349-9745.*

[18] *IBM: IBM Dictionary of Computing. McGraw-Hill, Inc. New York, NY, USA, 1993, ISBN 0070314888.*

[19] *Jatana, N.; Puri, S.; Ahuja, M.; aj.: A Survey and Comparison of Relational and Non-Relational Database.* IJERT, *ročník 1 Issue 6, 2012, ISSN 2278–0181.*

[20] *Kazerooni, J.: What is a cluster table? [Online; navštívené 5.11.2018].*
URL http://www.iselfschooling.com/mc4articles/mc4cluster.htm

[21] *Kim, W.: Introduction to Object-oriented Databases. Cambridge, MA, USA: MIT Press, 1990, ISBN 0-262-11124-1.*

[22] *Lavoie, B.; Gartner, R.: Preservation Metadata (2nd edition). Technická zpráva, Digital Preservation Coalition, 2013, doi:http://dx.doi.org/10.7207/twr13-03, DPC Technology Watch Report.*

[23] *Li, X.: Classification with Large Sparse Datasets: Convergence Analysis and Scalable Algorithms.* Electronic Thesis and Dissertation Repository, *2017.*
URL https://ir.lib.uwo.ca/etd/4682

[24] *Loshin, D.: Enterprise knowledge management: the data quality approach. Morgan Kaufmann, 2001, ISBN 978-0-12-455840-3.*

[25] *Navathe, S. B.; LarryKerschberg: Role of data dictionaries in information resource management.* Information & Management, *ročník 10, Issue 1, 1986: s. 21–46, doi:https://doi.org/10.1016/0378-7206(86)90058-3.*

[26] *nVidia Corporation: DB engines ranking. [Online; navštívené 10.11.2018].*
URL https://db-engines.com/en/ranking

[27] *Pascal, F.: What Is a True Relational System (and What It Is Not). 2017, [Online; navštívené 15.1.2019].*
URL http://www.dbdebunk.com/2017/03/what-is-true-relational-system-and-what.html

[28] *Paterson, J.; Edlich, S.; Hörning, H.; aj.: The Definitive Guide to Db4O. Berkely, CA, USA: Apress, 2006, ISBN 1590596560.*

[29] Ramakrishnan, R.; Gehrke, J.: Database Management Systems (2nd Ed.). McGraw–Hill Higher Education Kaufmann, 2000, ISBN 0-07-246535-2.

[30] Rao, T.; Haq, E.; KHAN, D.: Performance based Comparison between RDBMS and OODBMS. International Journal of Computer Applications*, ročník 180, 02 2018: s. 42–46, doi:10.5120/ijca2018916410.*

[31] Soltesz, D. L.: What are the Advantages of a Relational Database Model? [Online; navštívené 21.12.2018].
URL https://www.techwalla.com/articles/what-is-an-erp-system-administration

[32] Street, Q.: Introduction to Relational Databases - Part 1: Theoretical Foundation. 2008, [Online; navštívené 23.11.2018].
URL https://www.codeguru.com/csharp/.net/net_data/article.php/c19615/Introduction-to-Relational-Databases--Part-1-Theoretical-Foundation.htm

[33] Tauro, C. J. M.; Sahai, R. K.; A., S. R.: Article: Object Persistence Techniques - A Study of Approaches, Benefits, Limits and Challenges. International Journal of Computer Applications*, ročník 85, č. 5, 2014: s. 19–27, full text available.*

Príloha A

Obsah CD

Priložené CD obsahuje nasledujúce súbory:

- BP_ elektronickaForma.pdf – elektronická forma bakalárskej práce
- BP_ listForma.pdf – forma pre tlač bakalárskej práce
- ObjDatabases – Java projekt na vytvorenie a naplnenie objektových db4o databáz
- UmrtneZaznamy – Laravel PHP projekt na inicializáciu a naplnenie relačnej PostgreSQL databázy záznamov o úrmtiach.
- SvadobneZaznamy – Laravel PHP projekt na inicalizáciu a naplnenie relačnej PostgreSQL databázy záznamov o svadbách