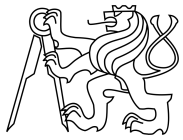
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Vyhledávání osobních údajů v relačních databázích
Student:	David Skalský
Vedoucí:	Ing. Jiří Mlejnek
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2018/19

Pokyny pro vypracování

Seznamte se s problematikou ochrany osobních údajů (Zákon č. 101/2000 Sb., Nařízení (EU) 2016/679 (GDPR)). Zaměřte se především na definici osobních a citlivých údajů.
Na základě seznámení identifikujte informace, které lze považovat za osobní údaje. Po dohodě s vedoucím práce, vyberte podmnožinu těchto informací a proveďte jejich detailní analýzu.
Popište jejich typické vlastnosti a různé způsoby jejich reprezentace v relačních databázových systémech (použité formáty, porušení vs. dodržení 1NF, apod.).
Navrhněte a implementujte řešení, které dokáže tyto osobní údaje proaktivně v databázi vyhledávat a upozorňovat na případné nálezy. Pro implementaci použijte jazyk Groovy.
Připravte testovací data a implementované řešení na nich otestujte.
Otestuje použitelnost navrženého řešení také z pohledu rychlosti, pro tabulky obsahující velké množství záznamů.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 15. listopadu 2017

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Vyhledávání osobních údajů v relačních databázích

David Skalský

Katedra Softwarového inženýrství
Vedoucí práce: Ing. Jiří Mlejnek

14. května 2018

Poděkování

Zde bych rád poděkoval vedoucímu mé práce Ing. Jiřímu Mlejnkovi za stálou podporu a pomoc v průběhu psaní této práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Praze dne 14. května 2018 .....

Odkaz na tuto práci

Skalský, David. *Vyhledávání osobních údajů v relačních databázích.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

Abstrakt

V této bakalářské práci jsou analyzované osobní a citlivé údaje podle GDPR (nařízení EU 2016/679 upravující náležitosti práce a nakládání s osobními daty) z pohledu formálního a zároveň z pohledu eventuálního uložení těchto údajů v reálných databázích. Dále je součástí návrh a implementace systému, který je schopen tyto údaje, podle popisu v analytické části, aktivně v databázích vyhledávat. Implementační část je vytvořena jako rozšíření již existujícího nástroje na řezy dat a anonymizaci Winch. Celá aplikační část využívá technologie Java, konkrétně se jedná o jazyk Groovy. Co se týče databázových strojů tak podporuje Oracle DB a Microsoft SQL Server. Kromě holé funkčnosti se práce zabývá i optimalizacemi a výkonem výsledného nástroje pro různorodá vstupní data.

Klíčová slova GDPR, relační databáze, osobní údaje, Groovy, Oracle SQL, MSSQL, Winch

Abstract

This Bachelor's thesis analyses personal data according to the GDPR regulation in both their formal appearance and their appearance in the actual

databases. It also contains both the design and the implementation of the system that is able to actively search for the analysed personal data in relational databases. The implementation part is an extension to an already existing utility used for personal data anonymization called Winch. The backend of this utility is coded in the Groovy programming language and supports Oracle DB and Microsoft SQL server. Apart from securing the functionality it also aims to optimize and test the performance of the final result.

Keywords GDPR, relation database, personal data, Groovy, Oracle SQL, MSSQL, Winch

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Dne 27. 4. 2016 bylo schváleno obecné nařízení Evropské Unie o ochraně osobních údajů (GDPR), které upravuje způsob, jakým musí správci osobních údajů s těmito nakládat, především zpřísňuje povinnosti pro zpracovatele, za účelem ochrany údajů koncových uživatelů. Nařízení se týká všech členských zemí EU a vejde v účinnost 25. 5. 2018 v každé z nich (včetně České Republiky). I když se datum, kdy se nařízení stane účinným, rychle blíží, většina firem na toto nařízení pořád není připravená a riskuje tak obrovské pokuty, které mohou být pro firmu likvidační. Právě kvůli tomu je v dnešní době velmi aktuální tento problém řešit. Jednou z hlavních změn, které nové nařízení přinese je povinnost zpracovatele nakládat s osobními daty občana, podle toho jak uzná sám občan za vhodné, má tedy právo správce požádat o vymazání a zapomenutí svých údajů z databází, má právo vědět přesně jaké údaje o něm zpracovatel drží a ten má zase povinnost mu na přání tyto údaje vydat.

Úkolem této bakalářské práce je navrhnout způsoby jakými lze identifikovat osobní a citlivé údaje v databázích a dokumentech, což je hlavní krok k jejich následné anonymizaci. Konkrétně se jedná o analýzu problému zjištění přítomnosti osobních údajů a implementaci řešení na základě této analýzy, jakožto rozšíření již existujícího nástroje na řezy dat – Winch, který vznikal v rámci jiné bakalářské práce přímo na FIT ČVUT v Praze [1]. Toto rozšíření by mělo zvládat identifikaci osobních údajů jak ve strukturovaných datech v databázích, tak v nestrukturovaných binárních datech (např. v dokumentech uložených v databázi, což mohou být různé smlouvy atd.). Kromě základních způsobů zjištění pravděpodobnosti přítomnosti osobních dat ve sloupci databáze (porovnání jména sloupce, komentáře, porovnání s validační funkcí – v případě rodného čísla) bylo úkolem práce vytvořit složitější vazby mezi jednotlivými osobními údaji, které pomohou co možná nejpřesněji poznat, který údaj se ve sloupci nachází.

Kapitola 1

Cíl práce

Cílem této bakalářské práce je seznámit se s problematikou ochrany osobních údajů (Zákonem č. 101/2000 Sb., Nařízení (EU) 2016/679 (GDPR)) a zaměřit se na definici osobních a citlivých údajů. Na základě tohoto seznámení potom dále identifikovat informace, které lze považovat za osobní údaje, a z těchto provést detailní analýzu. Součástí této analýzy jsou typické vlastnosti jednotlivých údajů a různé způsoby jejich reprezentace v databázových systémech. Na základě této analýzy je potom cílem navrhnout a implementovat řešení, které dokáže tyto osobní údaje proaktivně v databázích vyhledávat a upozorňovat na případné nálezy. Implementace je v jazyce Groovy a kromě implementace se počítá i s vytvořením sady testovacích dat, na kterých bude toto řešení testováno.

Kapitola 2

Nařízení GDPR a základní pojmy

V této kapitole jsou shrnuty základní pojmy týkající se ochrany osobních údajů a popis nařízení GDPR.

2.1 GDPR

GDPR neboli Obecné nařízení o ochraně osobních údajů (z anglického General Data Protection Regulation), plným názvem Nařízení Evropského parlamentu a Rady (EU) č. 2016/679 ze dne 27. dubna 2016 o ochraně fyzických osob v souvislosti se zpracováním osobních údajů a o volném pohybu těchto údajů a o zrušení směrnice 95/46/ES je nařízení Evropské Unie, které bylo přijato dne 27. dubna si dává za cíl výrazně zvýšit úroveň ochrany osobních údajů občanů zemí EU.[2]

Hlavní přínos tohoto nařízení je v předefinování a rozšíření okruhu osobních údajů, lepší definování dříve sporných údajů a přidání nových údajů, které jsou zákonem chráněny jako jsou například biometrické údaje nebo údaje o rasové či etnické příslušnosti. Nedodržování tohoto nařízení bude mít v praxi za následek obrovskou pokutu pro toho, kdo nařízení porušil.[2]

2.2 Základní pojmy

Osobní údaj: „informace týkající se určeného nebo určitelného subjektu údajů. Subjekt údajů se považuje za určený nebo určitelný, jestliže lze subjekt údajů přímo či nepřímo identifikovat zejména na základě čísla, kódu nebo jednoho či více prvků, specifických pro jeho fyzickou, fyziologickou, psychickou, ekonomickou, kulturní nebo sociální identitu“[3]

Citlivý údaj: „údaj vypovídající o národnostním, rasovém nebo etnickém původu, politických postojích, členství v odborových organizacích, nábo-

ženství a filozofickém přesvědčení, odsouzení za trestný čin, zdravotním stavu a sexuálním životě subjektu údajů a genetický údaj subjektu údajů; citlivým údajem je také biometrický údaj, který umožňuje přímou identifikaci nebo autentizaci subjektu údajů“ [3]

Správce: „každý subjekt, který určuje účel a prostředky zpracování osobních údajů, provádí zpracování a odpovídá za něj. Zpracováním osobních údajů může správce zmocnit nebo pověřit zpracovatele, pokud zvláštní zákon nestanoví jinak“ [3]

Zpracovatel: „každý subjekt, který na základě zvláštního zákona nebo pověření správcem zpracovává osobní údaje podle tohoto zákona“ [3]

Anonymní údaj: „údaj, který buď v původním tvaru nebo po provedeném zpracování nelze vztáhnout k určenému nebo určitelnému subjektu údajů, subjektem údajů fyzická osoba, k níž se osobní údaje vztahují“ [3]

Pseudoanonymizace: „proces skrytí identity, jehož účelem je mít možnost sbírat další údaje týkající se stejného jednotlivce, aniž by bylo nutné znát jeho totožnost. Principem je nahrazení co největšího počtu údajů, podle kterých lze osobu identifikovat, uměle vytvořenými identifikátory. Takovéto údaje nicméně nejsou anonymizované a stále se na ně vztahuje GDPR!“ [3]

Kapitola 3

Analýza osobních údajů

V této kapitole jsou popsané veškeré zkoumané osobní a citlivé údaje. Zkoumání bylo provedeno jak z formálního hlediska, tak z pohledu jejich výskytu v databázi, čili jak se jednotlivé údaje v databázi jmenují, jaká klíčová slova obsahují komentáře těchto sloupců, jestli existuje mezi jednotlivými údaji nějaká vazba, jestli lze údaj validovat nějakou funkcí či regulárním výrazem, nebo jestli je potřeba údaj porovnávat proti slovníku nejčastějších hodnot. Veškeré tyto principy jsou popsané vzhledem k česky nazývané databázi, popřípadě databázi nazývané velmi jednoduchou angličtinou. Počítá se s porovnáváním řetězců textu s ignorací velikosti písmen a diakritiky.

3.1 Osobní údaje

V této sekci jsou popsané osobní údaje a jejich vlastnosti. Definici pojmu osobní údaj lze nalézt v sekci 2.2.

3.1.1 Jméno a příjmení

Za předpokladu, že se v databázi vyskytují další sloupce doplňující jméno (příjmení, tituly...) očekává se, že jméno bude minimálně jedno slovo, s možností dalších slov (druhé jméno u nás a dalších několik slov pro jméno u zahraničních jmen). Pokud se další sloupce zabývající se jménem v databázi nevyskytují, bude ve sloupci jméno nejspíš kompletní jméno s příjmením a tituly.

Z toho plyne že ideální způsob identifikace bude porovnání všech slov v tomto jméně na shodu se slovníkem jmen a příjmení, popřípadě i titulů. Slovník nejčastějších jmen v ČR lze sehnat např. na oficiálních stránkách ministerstva vnitra.[4]

Názvy sloupce Nejčastější budou v česky navržené databázi názvy „JMENO“, „KRESTNI_JMENO“ nebo „PRIJMENI“. V anglicky navržené databázi

to pak budou „NAME“ „FIRST_NAME“ nebo „SURNAME“.

Komentáře sloupce V komentáři se budou hledat klíčová slova jako „jméno“, „křestní“, „příjmení“

3.1.2 Pohlaví

Uložení tohoto údaje je popsáno standardem ISO/IEC 5218[5]. Ten určuje, že sloupec s takovým údajem by se měl jmenovat právě „SEX“, význam hodnot je popsaný v tabulce 3.1.

Hodnota v DB	Význam
0	neznámé pohlaví
1	muž
2	žena
9	neurčitelné pohlaví

Tabulka 3.1: Výčet a význam hodnot v ISO/IEC 5218 standardu

Kromě takovéto standardizované podoby se musí očekávat i nestandardní výčet hodnot, i ten je ale velmi omezený, v praxi to budou dvojice hodnot „muž“/„žena“, „male“/„female“, „m“/„ž“ nebo „m“/„f“.

Vzhledem k povaze dat a velmi omezenému oboru hodnot je vhodné kompletní údaj validovat regulárními výrazy:

1. ”^(\s*(0|1|2|9)\s*)\$”
2. ”^(\s*(m(už|uz|ale)?|(z|ž)(ena)?|f(emale)?)\s*)\$”

První regulární výraz odpovídá ISO/IEC 5218 standardu, druhý potom odpovídá ostatním hodnotám. Důležitá část těchto regulárních výrazů je začátek a konec, který jasně určuje že hodnota v databázi musí být právě taková jaké možnosti nabízí regulární výraz, jakékoliv znaky zpředu či zezadu záznamu už tento výraz nebere za validní, čili „muž“ je validní hodnota, „amuž“ už ne. Tato pojistka je použitá hlavně kvůli zkratkovitým zápisům („m“ / „z“), pokud by tam nebyla, tak by se jako validní hodnoty zdály všechny obsahující písmena „m“ a „z“ bez ohledu na zbytek řetězce, což je očividně špatné a nežádoucí chování.

Při použití této metody validace se sluší dodat, že hledat pouze poměr záznamů odpovídajících výrazu ku všem záznamům nemusí stačit. Lepší variantou je projít záznamy, které neodpovídají regulárnímu výrazu, a pokud jich bude větší množství, pak se neočekává, že by skutečně sloupec obsahoval údaje o pohlaví. Například se může stát, že se sice objeví záznamy s hodnotami podle tabulky 3.1, ale zároveň se objeví i jiné hodnoty. Tato skutečnost by měla pravděpodobnost nálezu razantně snížit, ne-li úplně vyloučit.

Názvy sloupce Kromě standardního „SEX“ se dá očekávat i anglické „GENDER“ nebo české „POHLAVI“.

Komentáře sloupce V komentáři se budou hledat klíčová slova „pohlaví“, „gender“ nebo „sex“.

3.1.3 Rodné číslo

Jednoznačný číselný identifikátor přidělený obyvatelům ČR. Jeho podobu určuje předpis č. 302/2004 Sb. §13. Skládá se většinou z 10 číslic, vzácně pak z číslic 9. Celé číslo je dělitelné 11 beze zbytku. První dvě číslice vyjadřují poslední dvě číslice roku narození, druhé dvojčíslí vyjadřuje měsíc narození (u žen je k tomuto přičteno 50) a třetí dvojčíslí vyjadřuje den narození. Koncovka tohoto čísla je rozlišujícím znakem obyvatel narozených v tomtéž dni. Mezi první část a koncovku se často vkládá znak lomítka „/“

RČ[1] přidělená před 1. lednem 1954 mají stejnou strukturu, ale koncovka je u nich pouze třímístná a celé číslo nesplňuje podmínku dělitelnosti 11.

Pokud v jeden den dojdou rodná čísla, potom se u mužů ke druhému dvojčíslí přičte 20 a u žen 70.

Tato pravidla mají několik výjimek, do roku 1985 bylo přiděleno cca 1000 RČ, které nejsou dělitelná 11 beze zbytku, zbytek u nich vychází 10. Dále mohou být i nějaká RČ před rokem 1954 desetimístná a to pokud byla přidělena dodatečně po roce 1954.[6]

Celkově je tedy rodné číslo velmi jednoduché na identifikaci, stačí projít všechny záznamy a použít validační funkci, která zkontroluje délku, dělitelnost a validitu dne, měsíce a roku. Jediný zádrhel jsou možnosti formátování, program musí počítat s variantami zápisu s lomítkem: „980412/2339“, s mezerou místo lomítka: „980412 2339“ a s variantou bez oddělovače „9804122339“.

Názvy sloupce V češtině jsou názvy sloupce jasné, tam se dá téměř jistě počítat s „RODNE_CISLO“ nebo „RC“. Pokud bude sloupeček nazván anglicky, tak se nemůže brát za validní název „ID“, jelikož by to mělo za následek velké množství falešně pozitivních nálezů. Místo toho se používá „BIRTH_NUMBER“ a „PERSONAL_ID“.

Komentáře sloupce V komentářích se bude vyhledávat celá fráze „rodné číslo“, „birth number“ a „personal id“. Použití samotných frází „číslo“ nebo „id“ by mělo opět za následek falešně pozitivní nálezy.

3.1.4 Adresa

Teoreticky jednoznačný údaj podle, kterého by mělo jít naleznout fyzické místo na světě, kde buď daná osoba bydlí (v případě trvalé adresy), nebo kam dané

[1] Rodné číslo

osobě chodí korespondence (korespondenční adresa). Typy adresy, ale na její formát nemají žádný vliv.

Existuje velmi mnoho způsobů zápisu, důležité je identifikovat složky adresy, čili popisné / evidenční / orientační číslo budovy, město, kraj, okres, PSČ, ulici, čtvrť, stát. Z toho jsou běžně povinné jenom číslo popisné, město, ulice, popřípadě PSČ[2].

Všechny tyto složky se identifikují podle slovníku nejčastějších hodnot (kromě popisného čísla). Po úspěšné identifikaci složek adresy je možné zjistit, zda je jejich kombinace validní v Registru územní identifikace, adres a nemovitostí[7].

Názvy sloupce Názvy budou „MESTO“, „OBEC“, „CITY“, „TOWN“ pro případ města, „CP“, „CISLO_ORIENTACNI“, „ORIENTACNI_CISLO“, „CISLO_POPISNE“, „POPISNE_CISLO“ v případě ČP, kraj, okres a stát budou mít názvy sloupců přesně podle svých vlastních názvů, čili „KRAJ“, popřípadě „REGION“, „OKRES“ a „STAT“, „STATE“ nebo „COUNTRY“. Dále je také možné, že všechny složky budou v jednom sloupci, potom se očekává „ADRESA“ jako název, popřípadě rozdělené po skupinách. V tom případě je „ADRESA“ myšlená kombinace ulice a ČP, někdy i města, PSČ bývá zvlášť.

Komentáře sloupce V komentářích se budou vyskytovat fráze „adresa“, „město“, „obec“, „vesnice“, „číslo popisné“, „stát“, „kraj“, „okres“, „address“, „region“, „city“, „town“ nebo „country“.

3.1.4.1 PSČ

PSČ je jediný údaj, který jde identifikovat i jinak než porovnáním se slovníkem, vždycky totiž bude obsahovat právě pět číslic. Při použití této metody je potřeba dát si pozor na formátování a ignorovat v záznamech bílé znaky, často se totiž stává, že někdo PSČ zapisuje ve formátu s mezerou „323 01“ nebo bez mezery „32301“.

Stejně je ale pořád porovnání se slovníkem elegantnější a mnohem přesnější varianta. Slovník měst a PSČ je k dispozici na webu České pošty.[8]

Názvy sloupce V české databázi se sloupec bude jmenovat „PSC“ nebo „POSTOVNI_SMEROVACI_CISLO“. V angličtině se používá „ZIP“, popřípadě „ZIP_CODE“ nebo „POSTCODE“ a „POSTAL_CODE“.

Komentáře sloupce V komentářích u sloupce obsahujícího PSČ se objeví fráze jako „poštovní“, „směrovací“, „psč“ nebo anglicky „postal code“, „zip“.

[2] Poštovní směrovací číslo

3.1.5 Datum narození

Tento údaj se dá v databázi uložit dvojím způsobem. Pokud je uložen ve formátu „DATETIME“, potom není s jeho validací problém. Může se ale uložit i jako řetězec znaků, což už je na validaci složitější.

Podle standardu ISO 8601[9] se datum zapisuje ve formátu „YYYY-MM-DD“, čili se jedná o čtyřčíslí (které určuje rok) a dvě dvojčíslí, z nichž to první určuje měsíc v roce a druhé den v měsíci s tím, že všechny části tohoto data jsou odděleny pomlčkami. Konkrétní příklad zápisu data 19. června 1996 by v tomto formátu byl 1996-06-19.

V praxi je vhodné nespoléhat se na dodržování standardu, a pokud už bude datum v textové podobě vyzkoušet ostatní možnosti.

- DD.MM.YYYY, např: 19.06.1996, 06.06.1996
- D. M. YYYY, např: 19. 6. 1996, 6. 6. 1996
- DD/MM/YYYY, např: 19/06/1996, 06/06/1996
- D/M/YYYY, např: 19/6/1996, 6/6/1996
- D. MMMM. YYYY, např: 19. červen 1996, 6. červen 1996
- DD-MM-YYYY, např: 19-06-1996, 06-06-1996
- YYYY/MM/DD, např: 1996/06/19, 1996/06/06

Názvy sloupce V češtině to bude jednoduše „DATUM_NAROZENI“ nebo „NAROZEN“. V angličtině potom „BIRTHDATE“.

Komentáře sloupce V češtině mohou komentáře obsahovat fráze „datum narození“ a „narozen“. V angličtině to bude „birth“ nebo „birthdate“.

3.1.6 Identifikační číslo osoby

Jednoznačný identifikátor ekonomického subjektu, který musí mít každá podnikající fyzická osoba, každá právnická osoba i organizační složky státu.

Jedná se o osmimístné číslo, pokud je číslo starší a nemá 8 cifer, pak je doplněno zepředu nulami. Prvních 7 číslic IČO je generováno náhodně, poslední číslice je kontrolní. Tato kontrola funguje na principu dělitelnosti váženého součtu číslic jedenácti.[10]

Kontrolní číslice x pro IČO $n_8n_7n_6n_5n_4n_3n_2x$ se vypočítá jako:

$$x = (11 - (8n_8 + 7n_7 + 6n_6 + 5n_5 + 4n_4 + 3n_3 + 2n_2) mod 11) mod 10$$

Vyhledání takového údaje tedy funguje velmi podobně jako v případě rodného čísla, je potřeba projít každý záznam a zkontrolovat jestli splňuje podmínky na délku a jestli vyhovuje kontrolnímu vzorci. Některé osoby ale

používají jako IČO svoje rodné číslo, což může zásadně ovlivnit hledání. Je potřeba buď validovat oba dva údaje současně v jednom sloupci, nebo snížit nutný poměr správně validovaných IČO k prohlášení nálezu.

Názvy sloupce Nejčastější budou jednoduché názvy jako „ICO“ nebo zastarale „IC“, zřídka se pak může objevit celý název „IDENTIFIKACNI_CISLO_OSOBY“. V angličtině se tomuto údaji říká „CRN“ neboli „COMPANY_REGISTRATION_NUMBER“

Komentáře sloupce V komentářích se budou objevovat očekávané fráze „ičo“, „ič“ nebo „identifikační číslo osoby“. V angličtině se bude jednat o termíny „crn“, nebo celé „company registration number“.

3.1.7 Daňové identifikační číslo

Jednoznačný identifikátor plátce daně nebo plátce DPH, často se používá zkratka „DIČ“. Tento údaj se používá v rámci celé EU[3], kde se tento údaj nazývá anglicky „VAT ID“ neboli „Value added tax identification number“[11].

První dva znaky tohoto čísla jsou „CZ“, potom následuje buď RČ (pokud se jedná o fyzickou osobu) nebo IČO (pokud se jedná o právnickou osobu).

Jediný rozumný způsob nalezení je podle již nalezených a validovaných údajů RČ nebo IČO. Více o těchto vazbách mezi údaji se dočtete v sekci 3.3.

Názvy sloupce Nejčastější budou zkratkové názvy „DIC“, výjimečně se může objevit celý název čísla „DANOVE_IDENTIFIKACNI_CISLO“. V angličtině se očekává sloupec s názvem „VAT_ ID“ nebo „VAT_IDENTIFICATION_NUMBER“.

Komentáře sloupce V komentářích se budou vyhledávat klíčová slova „daň“, „daňové“, „dič“, „vat“ a „vat id“.

3.1.8 Telefonní číslo

Jedinečná identifikace SIM karty, pevné linky nebo faxu. Pro ČR platí bez výjimky, že tel. číslo skutečných osob je devítičíslí, pro kompletní zápis je vyžadována jak předvolba tak hlavní část telefonního čísla, v případě českých vnitrostátních hovorů, ale tato není vyžadovaná - volání na číslo bez zadané předvolby se snaží spojit pouze s aparáty s předvolbou „00420“ nebo „+420“. Kromě této předvolby mají všichni operátoři fungující v ČR přidělené rozsahy dalších předvoleb, ze kterých mohou čerpat. I čísla pevných linek mají pevně dané předvolby, jak podle operátorů, ale i podle kraje. Seznam všech těchto dodatečných předvoleb lze získat například na internetu.[12]

Pro telefonní čísla z ostatních zemí platí, že se liší zaprvé v tzv. mezinárodní státní předvolbě, ale i v celkové délce běžně používaných tel. čísel (od

[3] Evropská unie

3 číslic až po třeba 12). Seznam mezinárodních státních předvoleb ostatních států lze získat na internetu[13].

Ideální řešení je tedy použít kombinaci vyjmenovaných slovníků k eliminování co možná největšího počtu falešně pozitivních výsledků.

Názvy sloupce Jako název sloupce se očekává zkratkové „TEL“, nebo celé názvy „TELEFON“, „TELEFONNI_CISLO“, „FAX“, „CISLO_FAXU“, „MOBILNI_CISLO“ a „MOBIL“. Angličtina rozšíří výčet názvů o „TELEPHONE“, „PHONE“, „PHONE_NUMBER“, „TELEPHONE_ NUMBER“ a „FAX NUMBER“.

Komentáře sloupce V komentářích se budou objevovat fráze jako „telefon“, „mobil“, „mobilní“, „fax“ a „tel“. V angličtině potom „phone“, „telephone“ a „cellphone“.

3.1.9 E-mailová adresa

Unikátní údaj, který identifikuje elektronickou poštovní schránku uživatele e-mailu. Dělí se na dvě části, místní a doménovou a oddělovačem mezi těmito dvěma částmi je znak „@“.[14]

1. Místní část:

   „V této části adresy se mohou vyskytovat následující znaky:

   - Znaky velké a malé abecedy (a–z, A–Z) (ASCII: 65–90, 97–122).
   - Číslice 0 - 9 (ASCII: 48–57).
   - Znaky ASCII: 33, 35–39, 42, 43, 45, 47, 61, 63, 94–96, 123–126.
   - Znak tečka (ASCII: 46) nesmí být na začátku adresy a také nesmí být použit vícekrát po sobě.
   - Speciální znaky (ASCII: 32, 34, 40, 41, 44, 58, 59, 60, 62, 64, 91–93) jsou povoleny s restrikcemi.

   Speciální znaky jsou povoleny za předpokladu, že jsou uzavřeny v uvozovkách (””).“[1]

2. Doménová část:

   „Druhá část emailové adresy je klasické doménové jméno. Někdy může být nahrazeno i IP adresou uzavřenou do hranatých závorek „address@[192.168.0.1]“.“ [1] Doménové jméno se vyhledá v DNS a přeloží, aby se zjistil skutečný server na který má pošta přijít.

Názvy sloupce Očekávají se názvy „EMAIL“, „MAIL“, „E-MAIL“, „EMAILOVA_ADRESA“, „E-MAILOVA_ADRESA“, „EMAIL_ADDRESS“ a „E-MAIL_ADDRESS“

Komentáře sloupce V komentáři sloupce se objeví fráze „email“, „e-mail“ a „mail“.

3.1.10 IP Adresa

Adresa počítače v nějaké síti. Existují dva hlavní typy adres, popisují je verze protokolů IP – IPv4[15] a IPv6[16].

1. IPv4:

   Dnes stále více používaná verze. Adresa IPv4 se zapisuje jako 4 byty zapsané v desítkové soustavy oddělené tečkami (např. 192.0.77.225).

2. IPv6:

   Novější verze protokolu, má mnohem větší rozsah a tudíž je i delší. Jedná se o 8 skupin čtyř hexadecimálních číslic oddělených dvojtečkou (např. 2001:0db8:0000:0000:0000:ff00:0042:8329). Sekvence nul (alespoň dvě skupiny s nulovou hodnotou) v adrese se může zapsat jako „::“ (což se preferuje např. 2001:0db8::ff00:0042:8329), ale pouze jednou v zápise. Dále se také nezapisují předcházející nuly před podstatnou číslicí (např. 2001:db8::ff00:42:8329). Písmena používaná v šestnáctkovém zápise jsou v drtivé většině malá.

Identifikace bude probíhat buď validační funkcí na formát adresy, nebo případně regulárním výrazem. Pro formát IPv4 se ještě dá regulární výraz napsat relativně jednoduše a čitelně.

- Regulární výraz generující pouze validní IPv4 adresy:

  ”ˆ(1\d2|2[0-4]\d|25[0-5]|[1-9]\d|[1-9])\.(1\d{2}|2[0-4]\d|25[0-5]|[1-9]\d|\d)\.
  (1\d{2}|2[0-4]\d|25[0-5]|[1-9]\d|\d)\.(1\d{2}|2[0-4]\d|25[0-5]|[1-9]\d|\d)$”

Pro IPv6 adresu se počítá s použitím funkce, protože regulární výraz by byl zbytečně složitý, nečitelný a neopravitelný.

V databázi se mohou vyskytnout oba typy IP adres zároveň! Je tedy nežádoucí vyřadit detekci jednoho typu jenom kvůli nalezení několika záznamů s adresou druhého typu.

Názvy sloupce Očekávají se názvy „IP_ADRESA“, „IP“ nebo anglicky „IP_ADDRESS“. Také se může stát, že názvem bude konkrétní typ adresy čili názvy „IPV4“ a „IPV6“.

Komentáře sloupce V komentářích se budou hledat fráze „ip“, „ipv4“ nebo „ipv6“.

3.1.11 Číslo občanského průkazu

Devítičíselný údaj, který jednoznačně určuje Občanský průkaz a jeho držitele. Tento doklad se vydává pouze občanům ČR, je tedy možné, že v databázi nebudou všechny záznamy vyplněné právě číslem OP, pokud bude databáze určená jak občanům ČR tak cizincům.

Formát čísla je tedy devět náhodných číslic, bez speciálního generování a bez kontrolních znaků. Je tedy nemožné takový údaj s jistotou identifikovat. Jediným znakem je délka čísla, čili se dají nalézt všechny záznamy ze sloupce, které splňují, že jsou tvořeny 9 číslicemi, na což lze použít např. regulární výraz:

- "^(\d{9})$"

Je potřeba se podívat především na ostatní příznaky, než na počet validních záznamů. Jelikož je číslo OP údaj, který může mít pouze občan ČR, potom musí mít cizinci nějakou alternativu (většinou jde o číslo pasu), kvůli které není možné razantně snižovat pravděpodobnost nálezu jenom kvůli několika hodnotám, které neodpovídají předchozímu regulárnímu výrazu.

Identifikaci může rovněž pomoct kontrola, jestli se nejedná autoinkrementační sloupec, takový nebude nikdy osobním údajem žádného typu, tudíž ani číslo OP.

Názvy sloupce Očekávají se názvy „CISLO_OP“ a nebo nezkratkové „CISLO_OBCANSKEHO_PRUKAZU“. V angličtině potom „ID_NUMBER“, „IDENTIFICATION_NUMBER“ nebo „ID_CARD_NUMBER“.

Komentáře sloupce V komentářích se objeví fráze „občanský průkaz“ nebo „id number“

3.1.12 GPS souřadnice

Data z satelitního GPS[4] systému, která jednoznačně určují místo na planetě Zemi. Souřadnice používané v ČR mají dva základní formáty:

1. Souřadnicový systém WGS 84

   World Geodetic System[17], mezinárodní systém používaný pro vojenské i civilní účely. V principu se jedná o 3 údaje, zeměpisnou délku, šířku a výšku. V praxi se zapisuje lokace podle délky (která nabývá hodnot 0°-180° na západ od nultého poledníku a 0°-180° na východ od nultého poledníku) a šířky (která nabývá hodnot 0°-90° na sever od rovníku a 0°-90° na jih od rovníku). Běžné formáty jsou:

   - stupně, minuty a vteřiny. Např. 40° 26’ 46” N 79° 58’ 56” W

[4]Global Positioning system (Globální polohový systém)

- stupně a minuty. Např. 40° 26.767' N 79° 58.933' W
- stupně. Např. 40.446° N 79.982° W

2. Systém jednotné trigonometrické sítě katastrální (S-JTSK)[18]

   Systém používaný v geodézii na území České a Slovenské republiky. Zapisuje se jako dvě kladná čísla (jedno pro šířku jedno pro délku, podobně jako u WGS 84) oddělená buď mezerou nebo jiným oddělovačem.

   - Např. 712216.8 1119069.1

V databázi se může stát, že jednotlivé složky budou v separátních sloupcích, čili že jeden sloupec budou záznamy o šířce a druhý o délce. Je tedy potřeba s tímto počítat a vyzkoušet i tuto kombinaci.

Názvy sloupce Očekávané názvy budou „GPS“ „GPS_SOURADNICE“ nebo anglicky „LOCATION“, „GPS_COORDINATES“.

Komentáře sloupce Komentáře se prohledávají na shodu s frázemi „gps“, „souřadnice“ a „coordinates“.

3.2 Citlivé údaje

V této sekci jsou popsané citlivé údaje a jejich vlastnosti. Definici pojmu citlivý údaj lze nalézt v sekci 2.2.

3.2.1 Biometrické údaje

„Osobní údaje vyplývající z konkrétního technického zpracování týkající se fyzických či fyziologických znaků nebo znaků chování fyzické osoby, které umožňuje nebo potvrzuje jedinečnou identifikaci, například zobrazení obličeje nebo daktyloskopické údaje.“[2]

3.2.1.1 Otisky prstů

V databázi se ukládají jako binární data, většinou ve formě obrázku, který pořídil scanner prstu a který se potom zpracovává. Tento přístup má tu výhodu, že jde zpracovat jakkoliv, čili je jedno jaký algoritmus čtení a porovnání se použije.

Další možnost je ukládat jako strukturované informace o jednotlivých vychýleních (přerušení, větvení, mosty mezi jednotlivými vzory) na specifikovaných místech prstu. Forma uložených dat se pak liší podle použitého standardu a algoritmu uložení.

Nejpoužívanější mezinárodní standard na toto porovnávání pochází ze Spojených států a jmenuje se Integrated Automated Fingerprint Identification System (IAFIS)[19][20].

Názvy sloupce Názvem může být třeba „OTISK_PRSTU“ nebo anglické „FINGERPRINT“. U tohoto údaje by ale název sloupce neměl dostávat velkou pozornost, protože slovo fingerprint má v informatice a v počítačové bezpečnosti úplně jiný smysl.

Komentáře sloupce V komentářích se budou objevovat fráze „otisk“ nebo „fingerprint“, platí ale stejné podmínky jako v případě názvu sloupce, může se stát, že se nalezne fráze, která bude ukazovat na fingerprint ve smyslu nikoliv daktyloskopie, ale ve smyslu počítačové bezpečnosti.

3.2.1.2 Rozpoznání obličeje

Data obsahující informace o strojově zpracovaném záznamu obličeje se v databázi ukládají jako binární data a to buď ve formě obrázku, nebo ve specifickém formátu podle použitého algoritmu na zpracování.

Rozpoznání obličeje se provádí dvěma technikami, první z nich je tzv. PCA[5][21] algoritmus, který pracuje na základě ukládání a porovnávání vlastních vektorů hlavních vlastností kontur obličeje.

Druhý algoritmus se jmenuje LDA[6][22]. Ten pracuje na principu lineárních kombinací jevů na obličeji popsaných u předchozího algoritmu.

Uložení pomocí obrázku dává výhodu nezávislosti na použitém algoritmu. Využití jednoho z popsaných algoritmů zase snižuje objem dat a tím i čas nutný k porovnání dvou obličejových dat.

Názvy sloupce Názvem může být třeba „OBLICEJ“, „FACE“.

Komentáře sloupce V komentářích se budou hledat fráze „obličej“, „pca“, „lda“ nebo „face“.

3.2.2 Rasové a etnické údaje

Za etnikum se považuje skupina lidí, kteří sdílí společnou etnicitu, kulturu a zvyky. Často jsou to příslušníci stejného národa (někdy se etnikum považuje za synonymum slova národ).

Rasa je zase skupina lidí, kteří sdílí stejné fyziologické vlastnosti, čili spíše než kulturní a náboženské vlivy se považuje za důležitou země původu těchto osob.

Vyhledání těchto údajů spoléhá na slovníky se seznamem těchto údajů. [23] [24]

Názvy sloupce Očekává se například název „ETNIKUM“, „ETHNICS“ nebo „RASA“ a „RACE“.

[5] Principal Component Analysis
[6] Linear Discriminant Analysis

Komentáře sloupce V komentářích se budou hledat fráze „etnika“, „etnikum“, „etnický“, „rasa“, „ethnic“, „ethnics“ a „race“.

3.2.3 Státní příslušnost

Národnost se liší od probraných osobních údajů v tom, že je jasně určená zemí narození či zemí kde má tato osoba státní občanství. Tento údaj se rovněž identifikuje použitím slovníku.

Vytvoření slovníku pro podporu tohoto údaje ale není jednoduché, jelikož se nikde nevyskytují data, která lze přímo použít. K jeho vytvoření bylo potřeba využít seznam států světa (k dostání například na stránkách Ministerstva vnitra ČR[25]) a k nim ručně dosadit název příslušníka – čili doplnit další sloupec, který vytvoří dvojici např. Česká republika – Čech. Pro hledání správné dvojice ke každému státu jsem využil Pravidla českého pravopisu[26], které pro hledání každého státu vrací i jak se nazývá jeho obyvatel.

Názvy sloupce Očekává se například název „OBCANSTVI“, „STATNI_PRISLUSNOST“ nebo „NATIONALITY“, „RESIDENT“ a „CITIZENSHIP“.

Komentáře sloupce V komentářích se budou hledat fráze „občan“, „státní“, „národnost“, „občanství“, „nationality“, „resident“, „citizenship“ a „citizen“.

3.2.4 Náboženská víra

Identifikování informace o náboženské víře je o něco málo složitější než u předchozích údajů, protože tento údaj se dá zapsat v mnoha různých formách (čili je potřeba sehnat více slovníků).

Jedna z možností je identifikovat podle církevní příslušnosti. Seznam církví schválených v České republice upřesňuje příloha zákona č.3/2002 Sb. [27]. V tomto rejstříku jsou vyjmenovány pouze samotné názvy církví, je tedy potřeba ke každé z nich připsat i název příslušníka – to totiž bude ten údaj, který se s největší pravděpodobností v praxi objeví, nikoliv název instituce.

Církve nepopisují samotnou náboženskou příslušnost, pouze určují, ke které instituci zabývající se náboženstvím se člověk hlásí. Je tedy potřeba další slovník, tentokrát náboženství a jejich odnoží. Seznam hlavních náboženských věr lze najít například v knize „Světová Náboženství“.[28]

Názvy sloupce Očekávají se názvy „NABOZENSTVI“, „CIRKEV“, „NABOZENSKA_VIRA“, „VIRA“ nebo anglicky „RELIGION“, „CHURCH“ nebo „FAITH“.

Komentáře sloupce V komentářích se budou hledat fráze „církev“, „církve“, „náboženství“, „náboženská“, „náboženský“, „náboženské“, „víra“, „víry“ nebo anglicky „church“, „faith“ a „religion“.

3.2.5 Sexuální orientace

Informace o sexuální orientaci člověka se může v databázi vyskytnout v několika málo hodnotách, základní rozdělení variací sexuální orientace u lidí[29]:

- „Heterosexualita“, neboli orientace pouze na lidi opačného pohlaví. Takto orientovaný člověk se nazývá „heterosexuál“.
- „Homosexualita“, neboli orientace pouze na lidi stejného pohlaví. Takto orientovaný člověk se nazývá „homosexuál“. Navíc se rozlišuje takto orientovaný muž – „gay“ a žena – „lesba“.
- „Bisexualita“, neboli orientace na osoby obou pohlaví. Takto orientovaný člověk se nazývá „bisexuál“.
- „Asexualita“, neboli absence sexuální touhy, neexistuje orientace ani na jedno z pohlaví. Takový člověk se nazývá „asexuál“.

Názvy sloupce Očekávají se názvy „ORIENTACE“, „SEXUALNI_ ORIENTACE“ nebo anglicky „SEXUAL_ORIENTATION“.

Komentáře sloupce V komentářích se objeví fráze „sexuální“, „orientace“ nebo „sexual orientation“.

3.3 Vazby mezi údaji

Výše uvedené metody sice fungují, ale jsou údaje, které pouze nimi buď nejdou identifikovat vůbec, nebo jdou, ale nepřesně. K tomu, aby se těmto nepřesnostem předešlo se musí osobní údaje identifikovat a potvrzovat navzájem ve vazbách.

3.3.1 Vazby s rodným číslem

Rodné číslo (viz sekce 3.1.3) se jednoduše identifikuje výše vypsanými metodami, a přitom obsahuje velké množství informací.

- Podle první části rodného čísla lze zjistit datum narození (viz sekce 3.1.5) osoby, čímž lze potvrdit či vyvrátit validitu a existenci sloupce s datem narození. Datum narození se stane potvrzeným osobním údajem až ve chvíli, kdy se potvrdí právě rodným číslem, do té doby se nedá rozhodnout, jestli se jedná o datum narození, nebo jakékoliv jiné neidentifikující a neosobní datum.
- Podle přičtení či nepřičtení 50 k druhému dvojčíslí v rodném číslem se dá zjistit pohlaví (viz sekce 3.1.2) osoby. Samotné pohlaví se identifikuje snadno, takže tato kontrola slouží jako zpřesnění výsledků pokud pohlaví souhlasí, popřípadě snížení pravděpodobnosti ve sloupci pohlaví, to v případě, že pohlaví s rodným číslem nesouhlasí.

3.3.2 Vazby s adresou

Adresa (viz sekce 3.1.4) ve své textové podobě se identifikuje složitě přes shodu s řadou objemných slovníků. Lze ale využít ostatní údaje ke zrychlení, či zpřesnění výpočtu, případně lze použít adresu i na validaci těch ostatních údajů.

- Podle souřadnic GPS (viz sekce 3.1.12) se dá (s použitím správných odchylek a vyhledávacího systému) zjistit adresa místa, tato adresa může sloužit ke zrychlení identifikačního procesu adresy uložené v databázi nebo zpřesnění označení adresy za osobní údaj. Stejně tak se dá adresa, která je již zvalidovaná převést do GPS souřadnic, které pak zase s nějakou odchylkou (řádově desítek metrů) mohou potvrdit jiné GPS souřadnice v databázi za osobní údaj.

- PSČ (viz sekce 3.1.4.1) obsahuje informaci o městu, kraji i městské části. Všechny tyto informace mohou potvrdit a urychlit identifikaci adresy a stejně jako v minulém případě opačným směrem může identifikovaná adresa validovat nalezené PSČ.

3.3.3 Vazby s DIČ

DIČ (viz sekce 3.1.7) je jeden z údajů, který pro svojí identifikaci téměř nutně potřebuje ostatní údaje (jediná další možnost je spoléhat na předponu a předpokládanou délku).

- Pro fyzickou osobu se (pokud nestanoví zákon jinak) pro kmenovou část[7] daňového identifikačního čísla používá rodné číslo (viz sekce 3.1.3), je tedy potřeba nejprve vyhledat rodné číslo a až potom začít validovat sloupec obsahující DIČ.

- Pro právnickou osobu se (pokud nestanoví zákon jinak) pro kmenovou část daňové identifikačního čísla používá IČO (viz sekce 3.1.6), je teda potřeba nejprve vyhledat IČO a až potom začít validovat sloupec obsahující DIČ.

Vzhledem k tomu, že se může stát, že v databázi budou v jedné tabulce jak právnické tak fyzické osoby, nejlepší řešení by bylo najít jak IČO tak RČ a až teprve potom začít validovat DIČ, čili nezkoušet validovat pouze s jedním pomocným údajem – to může mít za následek nepřesné výsledky.

[7] číselná část, po předponě „CZ“

3.3.4 Vazby se jménem

Jméno se identifikuje pomocí slovníků, ale i z toho se dají získat důležité informace.

- Slovníky jsou rozdělené na „ženská jména“ a „mužská jména“. Podle slovníku který se použije na úspěšné validování jména se tudíž dá rovnou získat i pohlaví tohoto člověka, popřípadě podle pohlaví hledat jenom ve slovníku jmen toho pohlaví, které bylo nalezeno.
- Pohlaví jde rovněž získat z oslovení („Pan“/„Paní“/„Slečna“), pokud je přítomno v databázi, to se dá potom dále využít, což je popsáno výše.

3.4 Shrnutí

V této sekci se nachází přehledná tabulka obsahující všechny analyzované osobní a citlivé údaje. Pro každý údaj shrnuje způsob jeho validace a pokud pro ten údaj existuje nějaký vazebný vztah (podle sekce 3.3).

Údaj	**Způsob validace**	**Vazba**
Jméno a příjmení	porovnání se slovníkem	ano
Pohlaví	regulární výraz	ano
Rodné číslo	validační funkce	ano
Adresa	porovnání se slovníkem	ano
PSČ	porovnání se slovníkem	ano
Datum narození	vazba	ano
IČO	validační funkce	ano
DIČ	vazba	ano
Telefonní číslo	porovnání se slovníkem	ne
E-mailová adresa	validační funkce	ne
IP Adresa	validační funkce	ne
Číslo OP	regulární výraz	ne
GPS Souřadnice	validační funkce	ano
Otisky prstů	validační funkce	ne
Rozpoznání obličeje	validační funkce	ne
Rasové a etnické údaje	porovnání se slovníkem	ne
Náboženská víra	porovnání se slovníkem	ne
Sexuální orientace	porovnání se slovníkem	ne

Tabulka 3.2: Shrnutí analýzy osobních a citlivých údajů

Kapitola 4

Návrh řešení

Tato kapitola popisuje implementačně nezávislé řešení použité ve finální programu. Rovněž obsahuje i platformně nezávislé požadavky na výslednou funkčnost.

4.1 Požadavky

Napřed je třeba vytyčit si, co se vlastně bude od aplikace vyžadovat. Cílem je rozšířit existující program, který se zabývá anonymizací a vyhledáváním osobních údajů v databázích tak, aby ve své vyhledávací (discovery) části podporoval co nejvíce možných osobních údajů, které je schopen vyhledávat a upozorňovat na jejich nálezy. Počítá se s využitím jednoduchých i složitějších způsobů identifikace.

Výsledný program by měl nejenom umět vyhledávat data, která jsou strukturovaně uložená ve sloupcích v databázi, ale také projít nestrukturovaná data (např. dokumenty), která mohou být v databázi přímo formou souboru, nebo jako odkaz buď na lokální filesystem nebo jako URL odkaz na vzdálený server.

Při procházení velkého množství dat (což se v komerčních databázích očekává) může dělat problém vysoká časová náročnost těchto operací, proto se očekává možnost uživatelsky volitelných řezů[8] zkoumaných dat a možnost uživatelsky volitelných optimalizací.

4.2 Způsoby identifikace

V kapitole 3 jsou popsány všechny charakteristické vlastnosti jednotlivých údajů, které mohou pomoci k výsledné identifikaci hledaného osobního údaje. Existuje několik způsobů jak takovou identifikaci provádět:

[8] Datový řez je optimalizační technika používaná při práci s databázemi s velkým počtem záznamů, která umožní pouštět příkazy na menším objemu dat, které ale vlastnostmi odpovídají (s menšími odchylkami) většině původních dat.[30]

1. Jednoduché způsoby, které nevyžadují čtení dat

   Tyto metody je možné implementovat pouze jako čtení jednoduchých metadat tabulky a sloupce. Není potřeba přistupovat k datům a tudíž se jedná o nejrychlejší (výpočetně nejméně náročné) způsoby identifikace.

   - Porovnání očekávaných názvů sloupců se skutečným názvem.
   - Porovnání očekávaných klíčových slov v komentáři sloupce na shodu ve skutečném komentáři.
   - Zjištění datového typu sloupce a porovnání s očekávanými hodnotami. Například ve sloupci, který má nastavený datový typ na čtyřznakový řetězec nemůže obsahovat PSČ, nebo rodné číslo, protože do 4 znaků se tyto údaje nevejdou, není tudíž potřeba řešit nic jiného a pro takový sloupec lze rovnou prohlásit, že tyto (a další, viz kapitola 3) osobní údaje nemůže nikdy obsahovat.
   - Zjištění minimální či maximální délky datového typu uložených záznamů a porovnání s očekávanými hodnotami.
   - Zjištění vlastností (auto–increment[9], primární klíč. . .) sloupce, které mohou vyvrátit přítomnost osobního údaje. Např. pokud je sloupec nastaven jako auto–increment, potom je jisté že tento sloupec nebude obsahovat osobní data.

2. Jednoduché způsoby, které vyžadují přístup k datům

   Tyto algoritmy pracují na principu získání počtu všech záznamů, které neobsahují null hodnotu a počtu validních záznamů splňujících zadané formáty a podmínky. Ve finále se z poměru validních ku všem záznamům vypočte pravděpodobnost přítomnosti osobního údaje.

   - Použití validační funkce na kontrolu očekávaných známých vlastností údaje.
   - Použití regulárního výrazu na kontrolu očekávaných známých vlastností údaje.
   - Porovnávat záznamy ve sloupci se slovníky osobních dat s omezeným výčtem hodnot (jméno, město, citlivé údaje. . .).

3. Složitější způsoby, které vyžadují přístup k datům

   Tyto metody rovněž získají stejný poměr, na kterém pracuje předchozí skupina algoritmů. Jediný rozdíl je v tom, že podmínky, kterými metody v této skupině validují buď nejsou na první pohled evidentní, nebo je potřeba využít i víc než jenom jeden zadaný sloupec a podívat se v hierarchii databáze výš.

[9] sloupec jehož hodnoty se automaticky generují

- Použití záznamů z již identifikovaného sloupce k potvrzení nálezu, či zpřesnění výsledků. Všechny vazby, které se v tomto způsobu identifikace používají jsou popsané v sekci 3.3.
- Zjištění skutečných délek jednotlivých záznamů a použití statistické analýzy k vyvrácení nálezu. Pod pojmem skutečné délky se rozumí jednotlivé délky všech záznamů typu řetězce znaků, nikoliv maximální / minimální možné délky, které jsou nastavené jako součást datového typu. Např. snížení pravděpodobnosti nálezu údaje PSČ pokud má 90 % záznamů délku větší než 7 znaků (víme totiž, že PSČ má délku 5 nebo 6 znaků, podle formátu).

Na použití těchto způsobů v praxi by nemělo mít vliv jestli se jedná o binární nestrukturovaná data, nebo jestli se jedná pouze o data ve strukturované databázi. Princip je stejný v obou případech a měl by vracet stejné výsledky.

Všechny tyto metody se mohou kombinovat pro získání nejpřesnějšího možného výsledku.

Kapitola 5

Implementace

V této kapitole se řeší problematika samotné implementace rozšíření současného nástroje Winch. Z implementačně specifických požadavků je potřeba zmínit hlavně programovací jazyk Groovy, ve kterém se píše veškerá logika nástroje a podpora minimálně databázových strojů Oracle DB a Microsoft SQL Server.

5.1 Anonymizační třídy

Základním stavebním kamenem implementace discovery procesu nástroje Winch jsou tzv. anonymizační třídy, neboli konkrétní implementace rozhraní `AnonymizationClassDiscovery`, které se starají o samotnou logiku prohledávání databáze.

Každý osobní údaj, který se hledá v databázi má vlastní anonymizační třídu nazvanou stejně, jako se jmenuje údaj v anglickém jazyce, pokud lze název jednoznačně přeložit, jinak se používá česká alternativa (např. `EmailDiscoverer` vyhledává emaily a `RodneCisloDiscoverer` vyhledává rodná čísla).

V rámci rozšíření bylo potřeba přidat anonymizační třídy pro většinu osobních údajů (předchozí implementace podporovala pouze 5 základních údajů – email, jméno, rodné číslo, IČO a adresa). Seznam údajů, ze kterých vznikly přidané anonymizační třídy, jejich specifika a způsoby identifikace jsou popsané v části 3.

Každá anonymizační třída implementuje rozhraní `AnonymizationClassDiscovery`. Toto rozhraní shrnuje společné chování všech anonymizačních funkcí, čili nabízí tyto funkcionality:

- Porovnává jména sloupců s očekávanými názvy osobního údaje
- Hledá v komentáři sloupce očekávaná klíčová slova, která se tam mohou vyskytovat

- Kontroluje jestli má sloupec relevantní datový typ pro hledaný osobní údaj
- Nabízí základní funkce výpočtu pravděpodobnosti podle počtu validních záznamů

Anonymizační třídy, které implementují toto rozhraní musí doplnit pouze svůj název, který se používá ve vázaném hledání (viz sekce 5.2) a ve výpisu výsledků, a metodu `discover`, která řeší všechny úkoly spojené s identifikací specifické dané anonymizační třídě. V této metodě se provádí všechny dotazy nad jednotlivými záznamy v databázi a všechny specifické výpočty pravděpodobností. Tento návrh a příklady implementace v původních anonymizačních tříd již byly součástí nástroje a autor práce se na nich nepodílel.

5.2 Vázané hledání osobních údajů

Jedno z hlavních rozšíření současného řešení je přidání podpory hledání osobních údajů, podle už identifikovaných sloupců. Kterých osobních údajů se toto týká a jak to funguje po formální stránce je vysvětleno v sekci 3.3 respektive v sekci 3.

Implementačně je potřeba zasáhnout do současného procesu vyhledávání, čili do třídy `AbstractDiscoverTableGenerator` a metody `discoverTable`, která se stará o prohledání jednotlivých tabulek databáze. Ve zkratce funguje metoda tak, že dostane na vstupu tabulku, projde všechny sloupce a pustí na každý z nich hledání pro všechny anonymizační třídy. Metoda doběhne s poslední anonymizační třídou v posledním sloupci. Pro hledání závislých osobních údajů se pak nabízí tyto přístupy:

1. Opačný návrh

   Moje původní myšlenka byla otočit současnou implementaci „naruby“, čili místo spouštění všech anonymizačních tříd v každém sloupci, projít postupně pro každou anonymizační třídu každý ze sloupců tabulky. Tahle myšlenka by ke své správné funkci potřebovala jasně dané pořadí v jakém se spouští anonymizační třídy tak, aby analýza závislých údajů přišla až po analýze údajů, na jejichž základě funguje tato vazba (např. rodné číslo a ičo by se mělo prohledávat dříve než dič, které z rodného čísla a ičo vychází).

2. Druhé kolo

   Další myšlenka je zachovat současnou implementaci a přidat pouze jakési druhé kolo, které při nálezu osobního údaje zanalyzuje, jaká z anonymizačních tříd nalezla správný výsledek. Na základě této analýzy určí, které osobní údaje jsou s tím nalezeným v potenciálním vztahu

a okamžitě spustí průchod všemi sloupci tabulky a hledá v nich právě tyto osobní údaje ve vztahu.

Nakonec jsem zvolil druhou metodu, která sice musí potenciálně provést více přístupů do databáze, jelikož při nálezu údaje spouští nové průchody všemi sloupci, tudíž může být výpočetně náročnější, na druhou stranu zase vždycky projde všechny vázané sloupce a nemůže se stát, že by nějaký vynechala. Původní myšlenka opačného návrhu by padla na nutnosti stanovit přesně dané pořadí osobních údajů, ve kterém se tabulka prochází. Takové pořadí nelze jednoznačně určit, protože vazby mezi údaji nejsou jednosměrné. Pohlaví může upřesnit rodné číslo, ale stejně tak naopak. Kromě toho metoda druhého kola nevyžaduje větší zásah do současné architektury, protože ho akorát rozšiřuje.

5.3 Porovnání se záznamy ve slovníku

Další část, která je potřeba do aplikace přidat, je používání slovníků hodnot osobních údajů rovnou v aplikaci. Dosud spoléhala na externí tabulky ve všech podporovaných databázových strojích, které používala jako slovníky. Toto řešení ale může být neefektivní a navíc se nejedná o úplně čistý návrh.

Upravená implementace vyžaduje všechny současně používané slovníky v `.csv` formátu souboru ve zdrojích projektu Winch. Rozhraní `AbstractDictionary` se potom postará o parsování těchto souborů a vložení dat z nich přímo do paměti, tak aby se docílilo co možná nejvyšší efektivity při vyhledávání ve slovnících. Kromě toho rozhraní nabízí ještě základní funkce pro vyhledávání ve slovníku – jednu pro vyhledávání podle fráze, která projde celý slovník a vrátí `true` v případě úspěšného nalezení a druhou která dokáže vrátit hodnotu na zadaném indexu.

Rozhraní `AbstractDictionary` potom implementují další třídy podle jednotlivých slovníků, přidají chybějící parametry jako je název slovníku, název souboru a výčet sloupců ze souboru, které se využívají přímo v této třídě / slovníku.

Kromě toho tahle implementace podporuje i okamžitý převod do SQL vytvářecích a vkládacích skriptů, o to se stará třída `SqlDictionaryBuilder`. Pro každý podporovaný databázový stroj se vytvoří odvozená třída od `SqlDictionaryBuilder`, do které se dopíšou platformně specifické informace (jako je syntax insert a create skriptů nebo používané datové typy).

Autorem tohoto návrhu i implementace je vedoucí práce, autor pouze přidal vlastní slovníky hodnot pro analyzované osobní údaje.

5.4 Specifika jednotlivých databází

Pro jednotlivé databáze je potřeba vymezit jak vypadají (jaké mají datové typy) a jak fungují binární data, jelikož extrakce těchto dat a následné hledání osobních údajů je jedna ze stěžejních částí této práce.

Další vyžadovaná funkcionalita je datový řez, nutný pro použití v jakýchkoliv větších databázích z optimalizačních důvodů. Datové řezy byly navrhnuty autorem práce, ale implementaci a testování provedl někdo jiný.

V současné době podporuje nástroj dva databázové stroje – Oracle DB a SQL Server od firmy Microsoft. Požadavky na práci s databázemi jsou:

5.4.1 Oracle DB

Uložení binárních dat: Binární data jsou v Oracle databázi uložena v těchto datových typech[31]:

- `RAW`, který má variabilní, nastavitelnou délku v bytech, maximálně však do 2000 bytů.
- `LONG RAW` je už zastaralý datový typ, který se udržuje především z důvodů zpětné kompatibility. Maximální délka dat uložených v tomto formátu je 2 GB. Není možné v jedné tabulce mít více sloupců typu `LONG RAW` a databáze nad tímto typem podporuje pouze sekvenční přístup čtení.
- `BFILE` je forma symbolického linku na soubor ve vnějším prostoru (vně samotné databáze). Maximální délka takového souboru je 4 GB, maximální délka jeho názvu 255 znaků. Maximální možný počet těchto souborů v databázi se určuje inicializačním parametrem `SESSION_MAX_OPEN_FILES`, který je sám limitován maximálním počtem současně otevřených souborů, co dovoluje operační systém.
- `BLOB` (binary large object) je řetězec binárních dat s proměnnou velikostí určenou uživatelem, maximálně však 2 GB.
- `CLOB` (character large object) má stejné vlastnosti jako `BLOB`. Jediný rozdíl je v tom, že `CLOB` je přímo určen na uložení dlouhého řetězce znaků v UNICODE kódování. Maximální délka zůstává stejná, čili 2 GB.

Získání malého vypovídajícího vzorku z databáze: V Oracle databázi jsou v zásadě dva způsoby jak dosáhnout datového řezu.

1. možnost:

```
SELECT *
FROM   (
     SELECT *
     FROM   table
     ORDER BY DBMS_RANDOM.VALUE )
WHERE  rownum < n;
```

   Číslo n zde označuje kolik řádků se má vybrat z tabulky table. Tento způsob je velmi pomalý a neefektivní pro databáze s mnoha záznamy, protože nejprve všem řádkům přiřadí náhodný index a potom vybere řádky s indexy menšími číslu n.

2. možnost:

   funkce `SAMPLE(`n`)`, kde n je přibližně procentuální podíl řádků ku celkovému počtu řádků v tabulce `table`.

```
SELECT *
FROM table SAMPLE ( n );
```

   Tento přístup je mnohem rychlejší, i když většinou nepřináší úplně náhodná data (řádky na začátku a konci tabulky jsou z nějakého důvodu algoritmem upřednostňovány), tento problém může vyřešit kombinace s další funkcí `SEED(`m`)`

```
SELECT *
FROM table SAMPLE( n ) SEED ( m );
```

   Oracle DB garantovaně pro stejné hodnoty parametru m vrátí stejné výsledky.

5.4.2 Microsoft SQL Server

Uložení binárních dat: Binární data jsou v MSSQL databázi uložená v těchto typech[32]:

- `binary[(` n `)]`, data s fixní délkou určenou při tvorbě tabulky. Parametr n udává hodnotu délky dat, možný rozsah je od 1 do 8000 bytů. Používá se, pokud jsou pro každý záznam data stejně dlouhá.
- `varbinary[(` n | `max )]` (binary variable), neboli data s proměnnou délkou. Parametr n jsou hodnoty od 1 do 8000 a `max` indikuje maximální délku úložiště $2^{31} - 1$.

- `image`, zastaralý datový typ, který bude odstraněn v některé z budoucích verzí SQL Serveru, Microsoft silně nedoporučuje tento datový typ používat a upravit všechny současné aplikace, které tento typ používají. Dá se nahradit typem `varbinary(max)`.

Získání malého vypovídajícího vzorku z databáze: Microsoft SQL Server stejně jako Oracle databáze nabízí v zásadě dvě možnosti jak dostat řez dat.

1. možnost:

```
SELECT TOP n PERCENT *
FROM table
ORDER BY NEWID()
```

   Zde je důležitá funkce `NEWID()`, ta každému řádku přiřadí nové unikátní náhodné ID, podle kterého se potom řádky seřadí a na závěr se z nich vybere horních n procent. Tato metoda je velmi neefektivní pro tabulky s mnoha záznamy a vyžaduje hodně zdrojů na provedení.

2. možnost:

```
SELECT * FROM table
WHERE ( ABS( CAST(
        ( BINARY_CHECKSUM ( * ) *
       RAND( ) ) as int ) ) % 100 ) < n
```

   Principem tohoto dotazu je vygenerování náhodného čísla pro každý řádek od 0 do 99, s tím, že závěrečné porovnání s n má za následek výstup přibližně n % řádků. Tento dotaz používá funkci `RAND()`, protože samotná funkce `BINARY_CHECKSUM()`, by při vyhodnocení dotazu dvakrát za sebou vrátila pokaždé stejné řádky. `ABS()` a `CAST()` se používají protože dotaz který obalují může vrátit desetinné i záporné číslo. Tato metoda je oproti té první mnohem rychlejší a efektivnější.

Poznámka: Pro oba databázové stroje platí, že v praxi se nepoužívá procentuální podíl, ale několik případů pro absolutní hodnoty počtu řádků. Například tabulka s malým počtem (stovkami) řádků se neřeže vůbec, a z tabulky s desetitisíci a více řádků se vybere něco kolem tisíce. Tato metoda se používá, protože použití procentuálních poměrů nedává správný počet vybraných řádků. Např. pokud bychom chtěli jedno promile všech řádků, tak pro desetitisícové tabulky by to fungovalo výborně, ale pro stořádkové by se z řezu nevrátil jediný řádek. Z tabulek s miliony záznamů by se pak vrátilo zbytečně moc řádků a řez by byl neoptimální.

5.5 Čtení binárních dokumentů v Groovy

Groovy se svojí základní sadou knihoven nedokáže číst dokumenty v jiném formátu, než je standardní textový soubor, na čtení dokumentů v požadovaných formátech je potřeba využít externí knihovny, a sice takové, které podporují co nejvíce formátů, dokážou samy rozhodnout o jaký formát se jedná a dokážou přečíst veškerá pro člověka čitelná slova v dokumentu. Jelikož jazyk Groovy funguje na Java platformě, je možné použít knihovny určené pro jazyk Java:

docx4j open source knihovna pro Javu schopna manipulovat Microsoft Open XML (.docx, .pptx, .xlsx) soubory. Je schopna číst, zapisovat i vytvářet tyto soubory a dává důraz na abstrakci a uživatelskou přívětivost. Jedinečný způsob manipulace mezi konkurencí (použíti JAXB[33] na převedení xml reprezentace vstupního souboru na skutečné Java třídy) přináší nejpřesnější výsledky a nejjednodušší manipulovatelnost. Oproti konkurenci naopak zaostává v možných formátech, podporuje skutečně jenom Microsoft Open XML soubory, čili chybí jak .pdf a .html, ale i původní .doc, .ppt, .xls formáty.[34]

Apache POI open source knihovna pro Javu, která přináši API na tvorbu, modifikaci a čtení MS Office souborů. Narozdíl od docx4j podporuje kromě nových Microsoft Open XML souborů i starší .doc / .xls / .ppt formáty. Nevýhodou je slabší backend oproti konkurenci (docx4j), takže se při čtení souborů často vrací nečistá nebo nepřesná data (nechtěné bílé znaky, může vracet nechtěná netextová data...). Nevýhodou je také chybějící podpora .pdf formátu.[35]

Apache PDFBox open source knihovna pro Javu, která dokáže vytvářet, číst, spojovat, rozdělovat, validovat nebo měnit .pdf soubory a číst jejich meta-data. Bohužel funguje jenom pro .pdf soubory a nenabízí podporu žádného jiného formátu.[36]

Apache Tika velmi obsáhlý open source framework pro Javu, který podporuje obrovskou spoustu (údajně až 1400) formátů souborů[10], mezi které patří námi vyžadované Office a Openoffice formáty, .pdf, .html, .xml, .rtf atd. Nabízí knihovnu pro Javu, s jejíž pomocí je možno z těchto souborů číst přímo proud textu nebo i jejich meta-data. Knihovna disponuje i rozpoznávací částí, která se různými algoritmy pokouší zjistit skutečný formát souboru a podle toho použít příslušnou knihovnu na čtení. Dokonce dokáže identifikovat zvukové stopy (.mp3, .wav...) a obrazová data (.jpg, .img...) a prozkoumat jejich atributy a meta-data, což se může v procesu identifikace osobních údajů rovněž hodit.[37]

[10] `http://tika.apache.org/1.11/formats.html`

iText open source knihovna pro Javu a .NET platformu. Funguje podobně jako PDFBox, podporuje pouze .pdf formát a dokáže tyto soubory číst, vytvářet a modifikovat.

Jako ideální pro potřeby aplikace jsem vybral nástroj Apache Tika, jelikož nabízí největší škálu podporovaných formátů a nabízí i rozhraní jak tyto formáty chytře identifikovat. Vybrat si kombinaci Apache PDFBox a POI by nedávalo smysl, jelikož Tika tyto dvě knihovny přímo využívá. Použití docx4j má tedy jedinou výhodu a to lepší architekturu oproti Tika, která funguje pouze na nové Microsoft Office formáty (.docx, .pptx a .xlsx). To by se dalo ve srovnání považovat jako nevýhoda nástroje POI, ale při testování a výzkumu těchto nástrojů jsem nezaznamenal situaci, že by při čtení dokumentu (což je prakticky jediná funkce relevantní pro potřeby programu) nějaká textová data chyběla. Může se stát, že se vrátí více dat, než je skutečně v původním textu (při použití extravagantního formátování ve Word dokumentu), ale to není ani zdaleka takový problém, jako kdyby některá data chyběla a program by nenalezl např. ve smlouvě důležitý osobní údaj. Další srovnání je PDFBox (který využívá i Tika) a iText, oba tyto nástroje používají rozdílné přístupy k parsování dokumentů a často se rozchází v efektivitě. Pro jeden dokument se může stát, že iText zvládne průchod a nález slov rychleji, v dalším případě má ale zase navrch PDFBox [38]. Nedá se tedy jasně určit, který z nástrojů je efektivnější, tudíž jsem se rozhodl zůstat u PDFBoxu, který je implementován v nástroji Tika.

5.5.1 Extrakce souborů z databáze

Všechny nástroje (včetně finálně zvoleného) na práci s binárními daty v Groovy vyžadují na vstupu soubor (třída `File`), je tedy potřeba záznamy, které chceme prohledat nejprve převést na tento objekt. Binární data se v databázi mohou vyskytovat v několika formách:

- Soubor uložený přímo v databázi ve formátech specifikovaných v sekci 5.4.
- Link nebo cesta na lokální filesystem.
- URL nebo jiná cesta na vzdálený server.

Je tedy potřeba vytvořit komponentu, která bude schopna vzít libovolnou z těchto forem uložení souboru a vrátit objekt `File`, který se potom bude prohledávat na případné nálezy.

Tuto práci řeší třída `AbstractFileDownloader`, která rozhodne o přítomnosti validních binárních dat podle výše uvedených možností a vrátí implementaci rozhraní `AbstractBinaryDownloader`, která dokáže vrátit `File` objekt přesně podle požadavků.

Pro samotné prohledání sloupců se používá odvozená třída `AbstractBinaryDiscoverer` třídy `AnonymizationClassDiscoverer` (základní implementace anonymizační třídy, viz 5.1). Tuto třídu potom implementují všechny anonymizační třídy znovu, tentokrát se používají pro hledání v binárních datech čili se před název původní třídy doplní slůvko `Binary`.

V původní implementaci se pravděpodobnost počítala z poměru počtu řádků splňující určující podmínku osobního údaje a počtu všech řádků s nenulovými hodnotami. Tento návrh zůstává i pro hledání v dokumentech, ale na dotazy do databáze se už nepoužívá `DiscoveryExecutor`, který implementoval pro strukturovaná data příkazy, které vracely počty validních řádků, ale všechny anonymizační třídy volají metody svého předka `AbstractBinaryDiscoverer`, který implementuje používané statistické funkce z `DiscoveryExecutor` nad binárními dokumenty.

K tomu se používá již zmiňované rozhraní `AbstractBinaryDownloader`, které vrátí pro každý řádek zkoumaného sloupce příslušný objekt `File` s binárním obsahem, který potom třída `FileDiscoverer` za pomocí nástroje Apache Tika projde a zkontroluje přítomnost osobních dat.

5.5.2 Identifikace v binárních datech

Nástroj na čtení binárních dat nám soubor převede do textu / znakového proudu, ze kterého můžeme slovo po slově vyhledávat hesla, která potřebujeme. V praxi se tento proces zahrnuje tyto možnosti

1. Hledání již zvalidovaných údajů v binárních datech

   První možnost jak hledat v těchto datech je využít již předchozí znalosti z vyhledávacího procesu. Čili místo hledání všech osobních údajů se hledají jenom takové údaje, které byly předchozími procesy nalezeny a označeny, akorát tentokrát zapsané v souboru.

   Toto může být velmi často používaná funkce, která nevyžaduje extrémní výpočetní výkon (stále se sice bude muset pro každý řádek projít soubor na všechny ostatní nálezy, ale pořád se jedná o výpočetně nejlevnější metodu průchodu souborů).

   Častý scénář bude např. tabulka lidí, ve které je uložené jméno, adresa a nějaká smlouva této osoby, ve které se právě jméno a adresa vyskytuje – právě tohle bude řešit tento režim průchodu souborů a označí sloupec se souborem jako sloupec obsahující osobní údaje jméno a adresa.

2. Nezávislé hledání údajů výpočetně nenáročných na identifikaci

   Tato možnost poprvé bere soubor jako každý normální sloupec se strukturovanými daty, akorát nehledá všechny údaje, ale pouze takové, které nemají zbytečně složitou identifikaci.

V praxi se tedy budou v této metodě hledat údaje které nevyžadují validaci pomocí slovníku, ale lze je jednoduše popsat nějakým vztahem jejich vnitřní struktury (tyto údaje jsou vypsány v tabulce 3.2 s heslem „validační funkce“ nebo „regulární výraz“ ve sloupci „validace“)

3. Nezávislé hledání libovolných údajů

   Nejsložitější možnost prohledávání, projde každý soubor a každé slovo tohoto souboru vyzkouší na shodu s každým záznamem v každém slovníku pro každý údaj. V reálném použití bez řezů tento proces může trvat velmi dlouho a nemusí přinést žádné nové výsledky, tedy co se týče efektivity (poměr výpočetní náročnosti a pravděpodobnosti nalezení osobního údaje) je na tom ze všech metod nejhůře.

   Využití ale má pokud by uživatel nutně tuto funkci vyžadoval, pokud by měl čas nebo hodně výkonný stroj, na kterém by tento režim spustil, aby měl jistotu, že se skutečně označí všechny osobní údaje ve všech sloupcích bez výjimky.

Ve finále by měl být výběr metod na identifikaci na uživateli, ať si sám zvolí, který režim odpovídá výpočetní síle jeho systému a časovým nárokům. Jako implicitní režim (bez zásahu uživatele) se dá použít číslo 2 („Nezávislé hledání údajů výpočetně nenáročných na identifikaci“), neboť je to režim s rozumnými nároky a propustí pouze předtím neobjevené údaje, které se musí validovat na shodu se slovníkem.

Kapitola 6

Testování

6.1 Unit testy

Jednotkové testy jsou vytvořené pro každou anonymizační třídu, liší se ale zásadně podle toho, jak daná anonymizační třída identifikuje osobní údaje (viz tabulka 3.2). K testování byl použit framework JUnit.

- Pokud anonymizační třída funguje na základě identifikace přes validační funkci nebo regulární výraz, potom se dá otestovat právě ta část s funkcí či výrazem, což zamezí velké většině pozdějších chyb, jelikož tato část je z pohledu identifikace ve třídě nejkritičtější.

  Ke každé třídě, která takto identifikuje údaje, je přidáno několik testovacích případů údajů, které třída označí za validní a několik případů, které označí za neplatné. Tyto případy byly navrženy tak, aby třída neoznačovala jenom tzv. „happy-path“, čili očividně validní údaje a nesmysly, ale i sporné záznamy (např. nějaké mezní hodnoty, nebo řetězce s uměle přidanými bílými znaky na testování nečistoty dat).

- Pokud anonymizační třída vyžaduje porovnávání na shodu se záznamy ve slovníku pak se dá tato funkcionalita testovat pomocí slovníků importovaných do programu (viz sekce 5.3).

  Testovat se dá vyhledávání validních a nesmyslných záznamů ve slovníku pomocí metody `find`, vyhledání záznamu ve slovníku na zadaném indexu metodou `getValueAtIndex` a také se dá testovat jestli slovníky vytváří validní SQL skripty.

6.2 Manuální testování

Kromě spolehnutí se na automatické testování programu byla funkčnost nástroje testována taky ručně. Nejjednodušším způsobem vyzkoušení funkcionality discover modulu je použít grafické rozšíření nástroje Winch – WinchAddin, ne-

boli rozšíření do programu EA[11], které dokáže jednoduše spouštět nad vytvořenými modely v EA všechny funkce nástroje. Toto rozšíření ale není součástí této práce. Druhá možnost jak ručně otestovat discovery proces je použít uživatelské rozhraní z příkazové řádky.

Po spuštění nástroje v discovery režimu nad modelem v EA se projdou všechny tabulky modelu a ve výpisu se zobrazí informace o výsledku a na pozadí budou probíhat všechny procesy popsané v sekci 4.2. Při znalosti databáze a modelu, který se testuje je možné dojít závěru, že nějaká část discovery procesu funguje špatně nebo nepřesně. Další informace o ručním testování v nástroji Winch jsou k dispozici v bakalářské práci o front-endu tohoto nástroje viz [39].

Tato metoda byla rovněž použita při testování v sekci 6.4.

6.3 Sada testovacích dat

Součástí zadání této práce je také vytvoření sady testovacích dat. Existuje několik možností jak taková data vytvořit.

1. **Ručně psaná testovací data**

   Jedna z možností jak vytvořit sadu testovacích tabulek do databáze je data ručně napsat. Je to metoda v principu nejjednodušší, ale suverénně nejpracnější. Navíc se těžko dá vypsat tolik různorodých záznamů, aby se dal nástroj Winch nějak prakticky otestovat. Pokud se ale nehledí na výsledek a na přesnost dicovery procesu, ale naopak se hledí na množství dat (například tato metoda byla použita při testování výkonu nástroje), je možné vytvořit malé množství dat, které se ručně dá zvládnout rychleji než generované, a to potom zduplikovat tolikrát kolikrát je potřeba. Výsledkem je sice velmi homogenní tabulka, která ale obsahuje velké množství záznamů a slouží v testování jako stress test.

   Další výhodou tohoto přístupu je, že takto vytvořená data jsou vždy správná, taková jaké autor zamýšlel a hodí se na testování specifických funkcí. Tato metoda byla rovněž použita při testování nově vytvořených anonymizačních tříd, protože bylo potřeba rychle testovat specifické hodnoty v databázi.

2. **Generovaná data z vlastních zdrojů**

   Další možnost je data generovat, a sice z dat, která se používají přímo v aplikaci, například generování podle regulárních výrazů nebo podle připravených slovníků pro jednotlivé osobní údaje.

   Takto vygenerovaná data ale budou zbytečně kvalitní, je potřeba do nich přidat nějakou náhodnou umělou odchylku a nečistotu, z těchto a časových důvodů nebyla tato metoda testování využita.

[11] Enterprise Architect

3. **Generovaná data za pomoci externích utilit**

   Další možnost generování je použít specializovanou externí utilitu, která tuhle problematiku řeší. Problém využívání těchto nástrojů je ale v tom, že drtivá většina z nich stojí spoustu peněz za používání, což je dost pochopitelné, neboť testování může být u aplikace jako je Winch úplně hlavní a nejtěžší část vývoje.

   Z nástrojů, které jsou k dispozici zdarma se nejlépe jeví nástroj **Mockaroo**[40], který ve své bezplatné verzi nabízí databáze o délce 1000 řádků, obsahující realistická data s možností nastavit kolik procent řádků zůstane nevyplněných pro každý sloupec a každému sloupci přidat extra nějakou „kazící“ funkci, která znečistí data tak, aby vypadala jako reálná. Nástroj nabízí množství předdefinovaných údajů, které může generovat, další údaje lze přidat jako regulární výraz, který bude generovat další záznamy. Všechny výsledky jdou stáhnout v několika formátech (jako je .csv, SQL skript nebo formát používaný v tabulkovacím programu Microsoft Excel).

Testovací data by neměla popisovat pouze tu ideální cestu, která potvrdí funkci implementace a akorát kopírují cestu předchozího jednotkového testování. Správná testovací data by měla počítat se špatnou čistotou dat, s nestandardními názvy sloupců, s různě formátovanými řetězci a s dalšími neideálními scénáři, které se často v praxi objevují.

6.4 Výsledky testování

Součástí zadání je testování výsledného řešení z pohledu rychlosti pro tabulky obsahující velké množství záznamů.

Testované byly tyto scénáře:

1. Tabulka s 6 sloupci, která vyvolá prohledávání vazebných údajů (vazba RČ – DIČ) a obsahuje 2 sloupce validované podle slovníků, 2 sloupce bez osobních dat a dvojici RČ a DIČ.

2. Tabulka s 2 sloupci, obsahující pouze křestní jméno člověka, které se zároveň nachází i v binárním souboru, tento je testovaný pouze na přítomnost již validovaného sloupce jména. V binárním souboru se nachází text navíc, obsahující jak nic neříkající věci, tak ostatní osobní data (které v tomto případě ale nejsou validovaná).

3. Stejná tabulka jako v předchozím případě, akorát se dovolí možnost nezávislého vyhledávání v binárním souboru.

Délky testovaných tabulek byly (pro každou tabulku) 200, 1000 a 10000 řádků.

V průběhu testování došel autor k závěru, že časově nejnáročnější operací je vyhledávání PSČ (viz sekce 3.1.4.1), které využívá slovník s mnoha záznamy. Tento fakt dokládá první test rychlosti (tabulka 6.1), ve kterém jsou všechny hodnoty mnohem větší a s přibývajícími řádky rostou strměji, než v případě prohledávání s vypnutou detekci PSČ (jako v případě testu z tabulky 6.2). V prvním jmenovaném testu dokonce program po 5 minutách nedoběhl a byl násilně ukončen uživatelem, z toho důvodu se v každém dalším testu detekce PSČ vypnula.

Počet řádků	**Výsledný čas**
200	6261 ms
1000	29349 ms
10000	5+ minut

Tabulka 6.1: Testování pro strukturovaná data s povoleným vyhledáváním PSČ

Počet řádků	**Výsledný čas**
200	3742 ms
1000	5094 ms
10000	20902 ms

Tabulka 6.2: Testování pro strukturovaná data bez vyhledávání PSČ

Počet řádků	**Výsledný čas**
200	9356 ms
1000	43874 ms
10000	415789 ms

Tabulka 6.3: Testování pro kontrolu nalezeného údaje v binárních datech

Počet řádků	**Výsledný čas**
200	34217 ms
1000	3 minuty
10000	netestováno

Tabulka 6.4: Testování pro kompletní vyhledávání v binárních datech

V případě neomezeného testování s binárními daty roste čas potřebný na nalezení osobních údajů velmi rychle, toto je zapříčiněno velkým množstvím operací, které musí program pro binární data udělat, kromě jednoho nebo

dvou slov jsou v dokumentu celé věty a odstavce, což velmi zvyšuje náročnost, především pak pro metody validace, které vyžadují porovnání se slovníkem.

Pro 10000 záznamů už prohledávání trvalo tak dlouho, že ani nedoběhlo do konce. Je tedy vhodné použít při kontrole binárních souborů datový řez, který vrátí něco kolem tisíce řádků, možná méně. V těchto mezích už je prohledávání únosné. Více o optimalizacích hledání v binárních dokumentech viz sekce 5.5.2.

Závěr

V této práci jsem se zabýval výzkumem osobních a citlivých údajů, který jsem dovedl do konce. Všechny (v databázích rozumně identifikovatelné) osobní údaje jsou rozebrané v rešeršní části a z nich se jich několik povedlo úspěšně implementovat. Jedná se o vyhledávače DIČ, pohlaví, PSČ, IP adresy, číslo OP a vyhledávače citlivých údajů. Winch v tuhle chvíli zvládá identifikaci na základě propojení mezi jednotlivými osobními údaji, dokáže vyhledávat osobní údaje v binárních dokumentech a počet těchto údajů, které je schopen rozpoznat, úspěšně stoupl. V budoucnosti se dá aplikace rozšířit o nějaké inteligentnější algoritmy hledání, které jsou popsané v rešerši, ale do samotné praktické práce se nedostaly (třeba vyhledávání podle délek hodnot ve sloupci), dále také o další databázové stroje, které může nástroj podporovat (PostreSQL, MySQL...).

Nakolik je implementace ideální se nejspíše zjistí až v reálném nasazení, já jsem udělal všechno proto, abych eliminoval všechny možné chyby a nepřesnosti, které mě napadly. Je možné, že v produkci s reálnými daty se přijde na problémy, které budou moji následníci vyřešit, proto jsem se snažil do textu vložit zároveň co nejvíce nápadů to šlo, i když třeba v tuhle chvíli nevypadají využitelně, abych jim práci co možná nejvíce ulehčil.

Literatura

[1] Smítka, P.: Datové řezy a anonymizace, bakalářská práce. *Praha, FIT ČVUT v Praze*, 2013: s. 40–41.

[2] Nařízení Evropského parlamentu a rady (EU) 2016/679, o ochraně fyzických osob v souvislosti se zpracováním osobních údajů a o volném pohybu těchto údajů a o zrušení směrnice 95/46/ES (obecné nařízení o ochraně osobních údajů).

[3] Zákon č. 101/2000 Sb., o ochraně osobních údajů a o změně dalších zákonů.

[4] Odbor centrálních informačních systémů: Četnost jmen a příjmení [online]. *Informační servis Ministerstva vnitra ČR*, září 2017, [cit. 2018-04-18]. Dostupné z: `http://www.mvcr.cz/clanek/cetnost-jmen-a-prijmeni-722752`

[5] International Standards Organization/International Electrotechnical Commission and others: Information Technology—Codes for the Representation of Human Sexes (ISO/IEC 5218: 2004). *Geneva, Switzerland: ISO/IEC*, 2004.

[6] Úplné znění zákona č. 133/2000 Sb., o evidenci obyvatel a rodných číslech a o změně některých zákonů (zákon o evidenci obyvatel), jak vyplývá z pozdějších změn.

[7] Český ústav zeměměřičský a katastrální: Registr územní identifikace, adres a nemovitostí (RÚIAN) [online]. duben 2018, [cit. 2018-04-18]. Dostupné z: `https://vdp.cuzk.cz/vdp/ruian/overeniadresy/vyhledej`

[8] Česká pošta: Seznam PSČ částí obcí a obcí bez částí [online]. *Zákaznické výstupy České pošty*, duben 2018, [cit. 2018-04-18]. Dostupné z: `https://www.ceskaposta.cz/ke-stazeni/zakaznicke-vystupy`

[9] International Standards Organization/International Electrotechnical Commission and others: Data elements and interchange formats – Information interchange – Representation of dates and times (ISO 8601:2004). *Geneva, Switzerland: ISO/IEC*, 2004.

[10] Zákon č. 111/2009 Sb., o základních registrech.

[11] European Commision: VAT Information Exchange System (VIES). duben 2018, [cit. 2018-04-18]. Dostupné z: `http://ec.europa.eu/taxation_customs/vies/faq.html`

[12] Bankovní Kód.cz: Telefonní předvolby [online]. duben 2018, [cit. 2018-04-18]. Dostupné z: `https://www.bankovnikod.cz/telefonni-predvolby/`

[13] Bankovní Kód.cz: Mezinárodní telefonní předvolby [online]. duben 2018, [cit. 2018-04-18]. Dostupné z: `https://www.bankovnikod.cz/mezinarodni-telefonni-predvolby/`

[14] Klyne, G.; Nottingham, M.; Mogul, J.: Registration procedures for message header fields. Technická zpráva, 2004.

[15] Postel, J.: RFC 791: Internet Protocol [online]. Září 1981, [cit. 2018-04-18]. Dostupné z: `ftp://ftp.math.utah.edu/pub/rfc/rfc791.txt`

[16] Deering, S.; Hinden, R.: RFC 8200: Internet Protocol, Version 6 (IPv6) Specification [online]. Červenec 2017, [cit. 2018-04-18]. Dostupné z: `https://tools.ietf.org/html/rfc8200`

[17] International Civil Aviation Organization: World Geodetic System – 1984 (WGS-84) Manual [online]. 2002, [cit. 2018-04-28]. Dostupné z: `https://www.icao.int/NACC/Documents/Meetings/2014/ECARAIM/REF08-Doc9674.pdf`

[18] Kubátová, R.: Systém JTSK a WGS-84, jejich charakteristika a vzájemná transformace. *Bakalářská práce. Plzeň: Západočeská univerzita v Plzni, Fakulta aplikovaných věd, katedra matematiky*, 2007.

[19] Dexter, L. M. I.; Pouratian, A. J.: Automated fingerprint identification system. Únor 9 1999, US Patent 5,869,822.

[20] Važan, R.: How does SourceAFIS algorithm work? [online]. listopad 2015, [cit. 2018-04-20]. Dostupné z: `https://sourceafis.machinezoo.com/algorithm`

[21] Kim, K.: Face Recognition using Principle Component Analysis [online]. *College Park, University of Maryland, Department of Computer Science*, [cit. 2018-04-28].

[22] Juwei Lu, K. P.; Venetsanopoulos, A.: Face Recognition Using LDA Based Algorithms [online]. *Bell Canada Multimedia Laboratory, The Edward S. Rogers Sr., University of Toronto, Department of Electrical and Computer Engineering*, květen 2002, [cit. 2018-04-28].

[23] Cole, J.: *Ethnic groups of Europe: An encyclopedia*. ABC-CLIO, 2011.

[24] AliasAlice: What are the different human races? [online]. *Answers.com*, [cit. 2018-04-28]. Dostupné z: `http://www.answers.com/Q/What_are_the_different_human_races`

[25] Ministerstvo vnitra České republiky: Kódy států [online]. únor 2015, [cit. 2018-04-28]. Dostupné z: `http://www.mvcr.cz/clanek/kody-statu.aspx`

[26] Hlavsa, Z.; Martincová, O.: *Pravidla českého pravopisu: Ústav pro jazyk český AV ČR*. Pansofia, 1993.

[27] Zákon č. 3/2002 Sb., o svobodě náboženského vyznání a postavení církví a náboženských společností a o změně některých zákonů (zákon o církvích a náboženských společnostech).

[28] KEENE, M.: Světová náboženství. Praha: KK, 2003. Technická zpráva, ISBN 80-242-0983-7.

[29] WEISS, P.; aj.: Sexuologie. Praha: Grada, 2010. Technická zpráva, ISBN 978-80-247-2492-8.

[30] Microsoft: Data Slices [online]. *TechNet*, [cit. 2018-04-28]. Dostupné z: `https://technet.microsoft.com/cs-cz/library/ms174764`

[31] Oracle: Oracle Data Types [online]. *Oracle Database Online Documentation 11g Release 1 (11.1)*, [cit. 2018-04-28]. Dostupné z: `https://docs.oracle.com/cd/B28359_01/server.111/b28318/datatype.htm`

[32] Microsoft: Data types (Transact-SQL) [online]. *SQL Server 2017 Documentation*, [cit. 2018-04-28]. Dostupné z: `https://docs.microsoft.com/en-us/sql/t-sql/data-types/data-types-transact-sql?view=sql-server-2017`

[33] Ort, E.; Mehta, B.: Java Architecture for XML Binding (JAXB) [online]. *Oracle Technology Network*, březen 2003, [cit. 2018-04-28]. Dostupné z: `http://www.oracle.com/technetwork/articles/javase/index-140168.html`

[34] Plutext Pty Ltd. and co.: docx4j 3.3.7 [online]. březen 2018, [cit. 2018-04-28]. Dostupné z: `https://www.docx4java.org`

[35] The Apache Software Foundation: Apache POI 3.17 [online]. září 2017, [cit. 2018-04-28]. Dostupné z: `https://poi.apache.org`

[36] The Apache Software Foundation: Apache PDFBox 1.8.14 [online]. duben 2018, [cit. 2018-04-28]. Dostupné z: https://pdfbox.apache.org/

[37] The Apache Software Foundation: Apache Tika 1.18 [online]. duben 2018, [cit. 2018-04-28]. Dostupné z: https://tika.apache.org/

[38] Lowagie, B.: Performance iText vs.PdfBox [online]. březen 2014, [cit. 2018-05-14]. Dostupné z: https://stackoverflow.com/questions/22340674/performance-itext-vs-pdfbox

[39] Smítka, P.: Datové řezy a anonymizace, bakalářská práce. *Praha, FIT ČVUT v Praze*, 2013: s. 71–87.

[40] Brocato, M.: mockaroo realistic data generator [online]. březen 2018, [cit. 2018-04-28]. Dostupné z: https://www.mockaroo.com/

Příloha **A**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `discovery` ..... zdrojové kódy discovery části nástroje Winch
  - `discoveryTest` ..... testovací data použitá při testování výkonu
- `text` ..... text práce
  - `BP_Skalsky_David_2018.pdf` ..... text práce ve formátu PDF
  - `src` ..... zdrojové soubory práce ve formátu LaTeX