Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Tvorba rozsáhlých úložišť patentových dat

Plzeň 2022 Bc. Vojtěch Danišík

Místo této strany bude zadání práce.

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V diplomové práci jsou použity názvy programových produktů, firem apod., které mohou být ochrannými známkami nebo registrovanými ochrannými známkami příslušných vlastníků.

V Plzni dne 30. dubna 2022

Bc. Vojtěch Danišík

Poděkování

Děkuji panu Doc. Ing. Daliboru Fialovi, Ph.D. za ochotu při vedení diplomové práce a rady s jejím vypracováním.

Abstract

Creation of large-scale patent data repositories. The aim of the diploma thesis is to get acquainted with the available sources of patent data and to create extensive local repositories of patent data enabling their effective searching and mining. The first part of the thesis thoroughly describes the types of patents, existing data sources and file formats in which patents are stored. Subsequently, the applicable technologies for searching and mining are described. The second part of the thesis is devoted to the selection of usable data and the implementation of selected technologies. Several queries and scenarios have been created to test efficient mining. The results of the testing are part of this work.

Abstrakt

Cílem diplomové práce je seznámit se s dostupnými zdroji dat o patentech a vytvořit rozsáhlá lokální úložiště patentových dat umožňující jejich efektivní prohledávání a vytěžování. První část práce důkladně popisuje typy patentů, existující zdroje dat a formáty souborů, ve kterých se patenty ukládají. Následně jsou popsány použitelné technologie pro prohledávání a vytěžování. Druhá část práce se věnuje výběru použitelných dat a implementaci vybraných technologií. Pro otestování efektivního vytěžování bylo vytvořeno několik dotazů a scénářů. Výsledky testování jsou součástí této práce.

Obsah

1 Úvod

První odstavec by byl o patentu. Jednoduše by se popsalo proč vlastně je potřeba patent, proč ho chce zadávající, lehce popsat co to je atd.

Druhý odstavec by byl o databázi, lehký popis jako k čemu to je, jak je to např. rozšířený atp.

Cílem této práce je se seznámit s dostupnými zdroji dat o patentech a vytvořit rozsáhlá lokální úložiště patentových dat umožňující jejich efektivní vytěžování. Zdroje dat musí poskytovat své databáze patentů (žádosti i publikace) zdarma a patenty musí obsahovat předem stanovené povinné atributy, aby je bylo možné použít. Získaná data budou následně uložena do specifického typu databáze, která bude umožňovat co nejefektivnější vytěžování uložených dat. To znamená rychlé vyhledávání správných výsledků v relativně krátkém čase pro miliony (až desítky milionů) záznamů. Import dat bude řešen pomocí jednoduché aplikace, která bude procházet všechna data a filtrovat ty patenty, kterým chybí některé povinné údaje (i přes to, že struktura obsahuje elementy, ve kterých se údaj má nacházet), nebo jsou nevalidní.

2 Patent

todo

základní informace + uložiště dat + info o zdrojích (patentové úřady atp)

2.1 Patent vs Užitný vzor

2.2 Patent vs Průmyslový vzor

2.3 Patent vs Ochranná známka

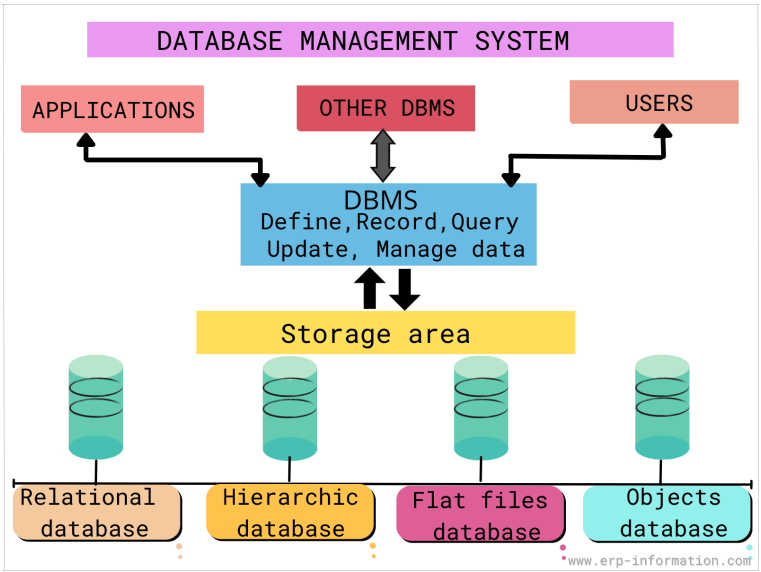
2.4 Patent vs Autorská práva

3 Databáze

Termín databáze označuje organizovanou kolekci strukturovaných informací nebo dat, která jsou typicky ukládána elektronicky v počítačovém systému. Data / informace lze nazvat jako fakta vztahující se k libovolnému uvažovanému objektu. Typický příklad objektu je člověk, jehož fakta jsou: jméno, věk, výška, váha a mnoho dalších [23].

3.1 Systém řízení báze dat

Pro správu dat v databázi a její řízení je potřeba komplexní software, který se nazývá **Systém Řízení Báze Dat** (SŘBD, anglicky DBMS). SŘBD slouží jako interface mezi samotnou databází a koncovým uživatelem (může být i program), umožňující jak vytěžování a aktualizaci dat, tak i možnosti nastavení záloh a jiných administrativních operací [1]. V dnešním světě existuje několik různých DBMS (například Relační DBMS, Objektově-orientované DBMS).



Obrázek 3.1: Systém řízení báze dat [9].

3.2 Komponenty databáze

Všechny databáze sestávají z pěti základních komponent, nehledě na použitý typ databáze [22, 23]:

- **Hardware** - Fyzické stroje (počítače, servery, pevné disky, ...) na kterých běží databázový software.
- **Software** - Databázový software poskytuje uživateli / programu kontrolu nad databází. Zahrnuje to samotný databázový software, operační systém, software pro správu sdílení dat mezi uživately a programy pro přístup k datům v databázi.
- **Data** - Nezpracované a neorganizované fakty, které je potřeba zpracovat. Administrátor databáze organizuje tyto data a dává jim význam. Data se obecně skládají hlavně z faktů, observací, percepcí, čísel, znaků a mnoho dalších.
- **Jazyk** - Typický příklad použití jazyku je přístup k datům, přidávání nových dat, úpravu již existujících dat z databáze. Uživatel / program napíše specifické příkazy v jazyku pro přístup k datům (Database Access Language) a tyto příkazy následně pošle databázi ke zpracování. Více viz kapitola č. 3.5.
- **Procedury** - Procedura obsahuje předpřipravený seznam příkazů, které se následně vykonávají po zavolání dané procedury.

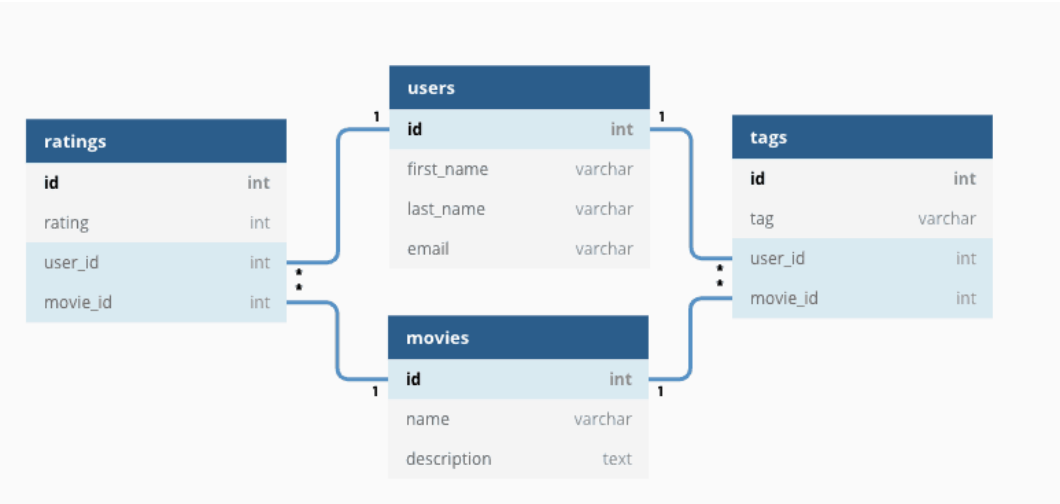
3.3 Typy databází

V dnešním světě existuje mnoho různých typů databází. Výběr nejlepšího typu databáze pro konkrétní organizaci závisí na tom, jak organizace zamýšlí data používat. V této kapitole je vypsáno pouze pár typů, protože vznikají stále nové, méně známé typy databází, které jsou tvořeny pro specifické požadavky (například finanční, věděcké) [1, 11].

3.3.1 Relační databáze

Název relační databáze pochází ze způsobu, jakým jsou data uložena, a to ve více souvisejících tabulkách. Data v tabulkách jsou uložena v řádcích a sloupcích. Relační databáze jsou velice spolehlivé a podporují všechny čtyři žádoucí vlastnosti databázových transakcí ACID. Pro co nejefektivěnjší využití tohoto typu databáze je potřeba ukládat pouze dobře strukturovaná

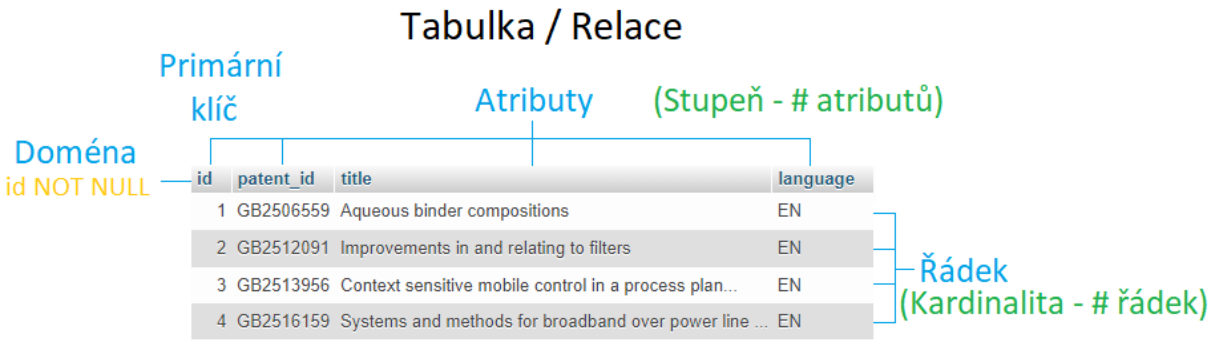
data, pro částečně strukturovaná či nestrukturovaná data je vhodné použít například grafové nebo dokumentově založené databáze. Typické relační databáze jsou například: Microsoft SQL Server, Oracle Database, MySQL. Ukázku relační databáze lze vidět na obrázku č. 3.2.



Obrázek 3.2: Ukázka relační databáze.

Datový model

Relační datový model obsahuje několik fundamentálních konceptů. Koncepty lze vidět na obrázku č. 3.3.



Obrázek 3.3: Koncepty relačního datového modelu [8].

První z konceptů se nazývá **Relace**, což je dvou-dimenzionální tabulka, která se používá pro ukládání kolekce datových elementů. Tabulka je tvořena řádky a sloupci, kde řádky reprezentují záznamy a sloupce reprezentují

atributy.

Řádka je další koncept relačního modelu, která pouze reprezentuje jeden záznam v tabulce.

Další z konceptů je **Atribut**, který reprezentuje sloupec v tabulce, neboli vlastnosti jednotlivých řádků (například jméno, příjmení, věk, ...).

Koncept **Doména atributů** slouží k definici vlastností pro každou hodnotu daného atributu. Pomocí domény lze určit, zda hodnoty daného atributu mohou být prázdné, budou dlouhé maximálně 50 znaků nebo například určit datový typ atributu (textová hodnota, číslo, ...).

Další z konceptů je **Stupeň**, který pouze určuje počet atributů v dané relaci.

Kardinalita určuje počet řádků / záznamů existujících v dané relaci.

Koncept **Relační schéma** popisuje návrh a strukturu relace. Obsahuje názvy tabulek, jejich atributy a typy atributů. Relační schéma pro naší tabulku lze vidět na obrázku č. 3.4.

```
PATENTS(id INT NOT NULL,
        patent_id VARCHAR(50),
        title VARCHAR(300),
        language VARCHAR(2)
       )
```

Obrázek 3.4: Relační schéma pro tabulku na obrázku č. 3.3.

Relační instance reprezentuje kolekci záznamů, které jsou uložené v tabulce v určitém čase.

Poslední koncept **Relační klíč** je atribut / seznam atributů, které lze využít jako unikátní identifikátor jedné entity v tabulce, případně k určení vazby mezi dvěma relacema. Existuje šest typů relačních klíčů - kandidátní, super, složený, primární, cizí, sekundární / alternativní [19].

Výhody

Výhody relační databáze jsou [18]:

- **Jednoduchost modelu** - Při porovnávání ostatních typů databází s relačním, relační databáze je o mnoho jednodušší. Díky tomu, že zde neprobíhá žádné zpracování dat, tak není potřeba využívat žádné složité dotazy.
- **Snadné použití** - Uživatelé můžou jednoduše přistupovat a získávat všechny potřebné informace v rámci sekund bez ohledu na složitost

databáze.

- **Přesnost** - Relační databáze jsou dobře uspořádané a velice striktně definované. I za pomoci primárních a cizích klíčů se v databázi udržuje unikátnost hodnot, takže se zde nevyskytují žádné duplikáty.
- **Integrita dat** - Integrita dat zajišťuje konzistentnost všech tabulek v databázi, díky čemuž lze dosáhnout vlastností jako přesnost a snadné použití.
- **Normalizace** - Normalizace je metoda, pomocí které lze rozdělit jednu informaci do několika bloků za účelem snížení velikosti.
- **Spolupráce** - Více uživatelů může přistupovat k datům ve stejný čas i v případě, že část dat je upravována.
- **Bezpečnost** - Bezpečnost je zajištěna autorizací uživatelů, kdy pouze uživatelé s právy a přístupovými údaji mohou přistupovat k datům.

Nevýhody

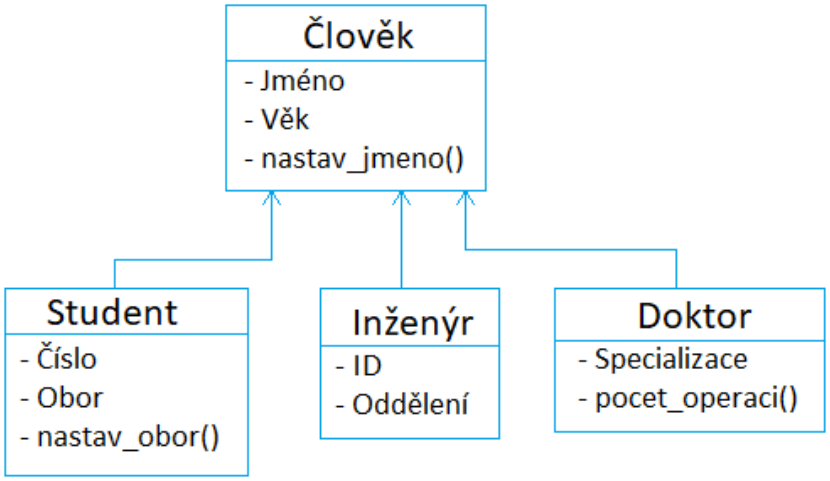
Nevýhody relační databáze jsou [18]:

- **Problém s údržbou** - Údržba relační databáze se stává postupem času náročnější vzhledem ke zvýšenému počtu uložených dat.
- **Cena** - Systém relační databáze je drahý k pořízení i pro správu. Samotná prvotní cena systému je relativně drahá pro menší byznys, ale zhoršuje se při zohlednění najímání profesionálních techniků, které musí mít dobré znalosti ohledně používaného systému.
- **Fyzické úložiště** - Relační databáze jsou složeny z řádků a sloupců, které potřebují hodně fyzické paměti, protože každá provedená operace závisí na samostatném úložišti.
- **Malá škálovatelnost** - Při používání relační databáze na více serverech se její struktura mění a stává se obtížně zvládnutelnou, zejména při velkém objemu dat. Jak se databáze zvětšuje nebo více distribuuje s větším počtem serverů, tak se zvětšuje latence a problémy s dostupností, které ovlivňují celkový výkon.
- **Složitost struktury** - Relační databáze dokáží ukládat data pouze v tabulkové formě, která neumožňuje vyobrazit složitější vazby mezi objekty. Toto může být velký problém u dost aplikací, u kterých data nelze reprezentovat pouze jednou tabulkou díky jejich aplikační logice.

- **Snížení výkonu postupem času** - S větším množstvím uložených dat a tabulek se zvětšuje i složitost systému, díky čemuž bude systém reagovat pomaleji na dotazy, případně může i spadnout v případě více dotazů od více uživatelů.

3.3.2 Objektově-orientovaná databáze

Objektově-orientovaná databáze je založena na objektově-orientovaném programování, kdy data a všechny jejich atributy a metody jsou svázány dohromady jako objekt. Stejně jako relační databáze, i objektově-orientované databáze odpovídají standardům ACID. Typické příklady jsou například: ObjectStore, ConceptBase. Ukázku objektově-orientované databáze lze vidět na obrázku č. 3.5.



Obrázek 3.5: Ukázka objektově-orientované databáze.

Datový model

V objektově-orientovaném modelu jsou data a jejich vztahy mezi sebou uložené v jediné struktuře, která se jinak nazývá objekt. Všechny objekty mají mezi sebou vícenásobné vztahy. Jednoduše řečeno, objektově-orientovaný datový model je spojením relačního databázového modelu a objektově-orientovaného programování [8].
V datovém modelu existují tyto komponenty:

- **Objekt** - Objekt je abstrakcí jakékoliv entity z reálného světa, jinak zvaná jako instance jedné třídy. Objekt zapouzdřuje data a funkční kód do celku, který poskytuje pouze datovou abstrakci, zatímco schovává

implementační detaily od uživatele. Příklad objektů z obrázku č. 3.5: *Student*, *Doktor* a *Inženýr* jsou instancí celku *Člověk*.

- **Atribut** - Atribut popisuje vlastnosti objektu. Například *Student* obsahuje atributy *Číslo* a *Obor*.
- **Metoda** - Metoda reprezentuje chování objektu. Například *Student* obsahuje metodu s názvem *nastav_obor*, pomocí které můžeme získat studovaný obor daného studenta.
- **Třída** - Třída je vlastně kolekce podobných objektů, které sdílejí strukturu (neboli atributy) a chování (neboli metody).
- **Dědičnost** - Vytvořenému objektu se říká instance třídy, která zdědí kopie / instance všech atributů a metod dané třídy. *Student*, *Doktor* i *Inženýr* dědí od celku *Člověk* atributy *Jméno*, *Věk* a metodu *nastav_jmeno*.

Výhody

Výhody objektově-orientované databáze jsou [26]:

- **Výkonnost** - Mnohonásobně výkonnější než relační databáze.
- **Rozšiřitelnost** - lze vytvářet nové datové typy z již existujících. Jako příklad lze uvést vytvořením super třídy, která bude obsahovat všechny společné atributy a metody. Tímto lze snížit redundanci v systému.
- **Podpora velkého množství datových typů** - Oproti ostatním typům databáze, objektově-orientovaná databáze podporuje ukládání různých typů dat, jako například obrázky, zvuky, video a mnoho dalších.
- **Podpora vývoje schématu** - Těsné propojení mezi daty a aplikacemi činí vývoj schématu více proveditelnějším.
- **Podpora pro dlouhotrvající transakce** - Objektově-orientovaná databáze využívá jiný protokol pro zpracovávání dlouhotrvajících transakcí než relační databáze.

Nevýhody

Nevýhody objektově-orientované databáze jsou [26]:

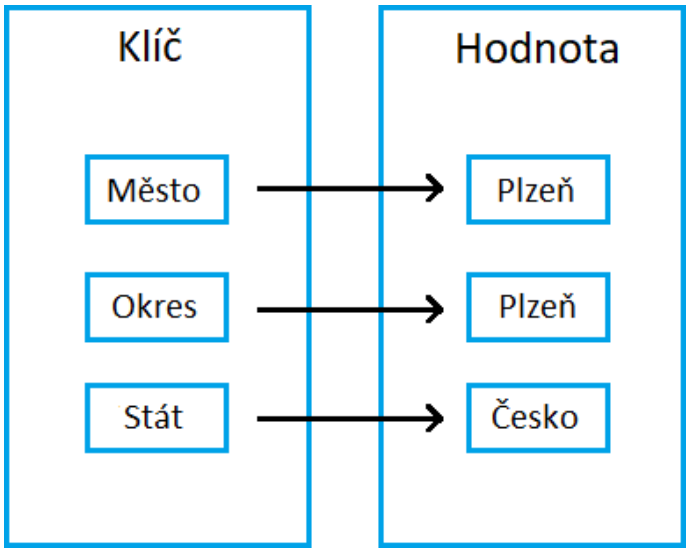
- **Neexistující univerzální datový model** - V dnešní době stále neexistuje univerzální datový model, navíc většině modelů chybí teoretický základ.
- **Nedostačující standardy** - Pro objektově-orientované databáze neexistují žádný univerzální datový model, stejně jako standardní dotazovací jazyk.
- **Složitost** - Funkcionality jako například dlouhotrvající transakce, zpráva verzí nebo evoluce schémat činí výsledný systém mnohonásobně složitější, což vede k vyšší ceně a složitějšímu používání.
- **Zabezpečení** - V databázi neexistuje adekvátní zabezpečovací systém, který by mohl přiřazovat přístupová práva na objekty nebo třídy.

3.3.3 NoSQL databáze

NoSQL je široká kategorie databází, které nepoužívají SQL jako svůj primární jazyk pro přístup k datům. Tyto typy databází jsou také někdy označovány jako nerelační databáze. V NoSQL databázích se pracuje s nestrukturovanými a polostrukturovanými sadami distribuovaných dat. Jednou z výhod je, že vývojáři mohou provádět změny databáze za běhu, aniž by to ovlivnilo aplikace, které databázi používají.

3.3.4 Databáze Klíč-Hodnota

Databáze klíč-hodnota poskytuje nejjednodušší možný NoSQL datový model. Data jsou uložená jako pár klíč - hodnota ve slovníku / mapě, kdy klíč je indexem. Hodnota může být například celé číslo, řetězec, struktura JSON nebo pole. Z vlastností databáze vyplývá, že zde není potřeba žádný dotazovací jazyk pro získávání výsledků. Typické příklady jsou: Redis, Riak, LevelDB. Ukázku databáze klíč-hodnota lze vidět na obrázku č. 3.6.



Obrázek 3.6: Ukázka databáze klíč-hodnota.

Výhody

Výhody této databáze jsou [4]:

- **Škálovatelnost** - Databázi lze škálovat jak vertikálně, tak i horizontálně.
- **Redundance** - Zabudovaná redundance zapříčiňuje větší spolehlivost databáze.
- **Rychlost** - Reakční čas je velice rychlý díky jednoduchosti struktury a jednoduchých příkazů (vlož, smaž, získej).

Nevýhody

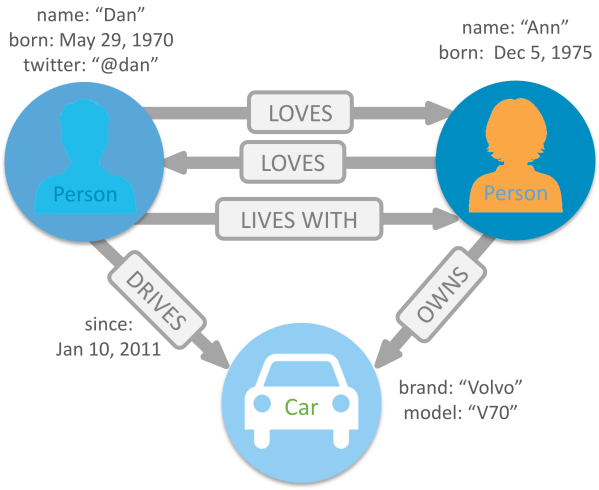
Nevýhody této databáze jsou [4]:

- **Optimalizace dat** - Optimalizace je provedena pouze pro data, kde je pouze jeden klíč a jedna hodnota. V případě ukládání složitějších struktur je potřeba parser.
- **Složité dotazy** - Nelze používat složité dotazy, pomocí kterých lze vyhledávat specifické hodnoty.

3.3.5 Grafová databáze

Grafová databáze je typem NoSQL databáze, která je založená na teorii grafů. Data jsou reprezentována jako uzly, hrany zase reprezentují vztahy

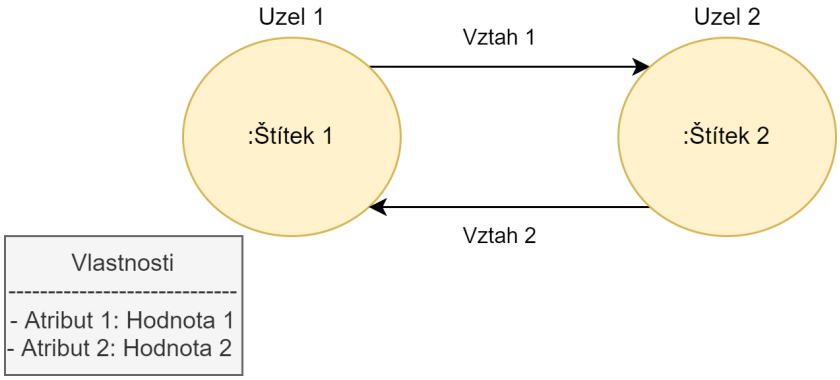
mezi daty. Graf lze procházet podél určitých typů hran nebo přes celý graf. Procházení spojení nebo relací je velmi rychlé, protože vztahy mezi uzly se nepočítají v době dotazu, ale jsou v databázi trvalé. Typické příklady jsou: Neo4j, OrientDB, Microsoft Azure CosmosDB. Ukázku grafové databáze lze vidět na obrázku č. 3.7.



Obrázek 3.7: Ukázka grafové databáze [16].

Datový model

Datový model grafové databáze se skládá ze čtyř komponent (viz obrázek č. 3.8) [7]:



Obrázek 3.8: Komponenty grafu.

- **Uzel** - Uzel je jeden ze dvou fundamentálních komponent který vytváří graf. Uzly slouží k reprezentaci entit nebo jiných doménových komponent.

- **Vztah** - Vztah propojuje dva uzly a dovoluje nám vyhledávat související uzly. Uzel, ze kterého vztah začíná, se jmenuje zdrojový, zatímco uzel, ve kterém vztah končí, se nazývá cílový (šipka ukazuje směr vztahu). Vztahy musí mít vždy jen jeden zdrojový a jeden cílový uzel, proto při mazání uzlů se mažou i všechny jeho závislosti (vstupující a vystupující vztahy).
- **Štítek** - Štítek slouží k zařazování uzlů do skupin. Všechny uzly, které jsou označené stejným štítkem, patří do jedné skupiny. Uzel může obsahovat libovolné množství štítků (0 až nekonečno). Při vyhledávání může databáze pracovat nejen s celým grafem, ale i s množinou uzlů patřící do jedné skupiny.
- **Vlastnosti** - Vlasnost je množina dvojic klíč - hodnota, které lze ukládat s každým uzlem a vztahem. Jsou podporovány skoro všechny datové typy.

Výhody

Výhody grafové databáze jsou [3]:

- Struktury jsou flexibilní a přizpůsobivé.
- Reprezentace vztahů mezi entitami je zřetelné.
- Dotazy poskytují výsledky v reálném čase. Rychlost závisí na počtu relací.

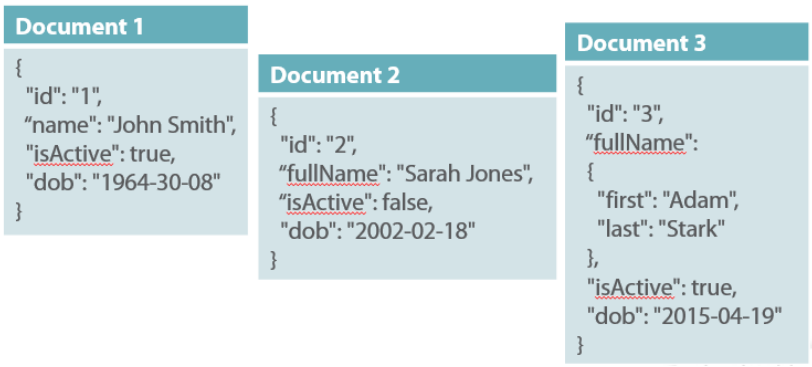
Nevýhody

Nevýhody grafové databáze jsou [3]:

- Neexistuje žádný standardizovaný jazyk. Jazyk závisí na použité platformě.
- Grafy jsou nevhodné pro transakční systémy.
- Je těžké najít podporu, protože uživatelská základna je velice malá.

3.3.6 Databáze dokumentů

Databáze dokumentů jsou typem NoSQL databáze a jsou navržené pro ukládání, načítání a správu informací orientovaných na dokumenty. Typické příklady jsou: MongoDB, Amazon DocumentDB, Elasticsearch. Ukázku dokumentové databáze lze vidět na obrázku č. 3.9.



Obrázek 3.9: Ukázka dokumentově orientované databáze [21].

Datový model

Základním prvkem dokumentové databáze je **Dokument**. Definice dokumentů se liší podle konkrétní implementace databáze, ale jedno mají společné: dokumenty kódují zapouzdřená data či informace do nějakého standardního formátu nebo kódování. Mezi typy kódování patří například JSON, XML. Dokument nemusí dodržovat pevně definovanou strukturu, takže v databázi mohou existovat dva dokumenty stejného formátu s rozdílnou strukturou dat [27].

Výhody

Výhody dokumentové databáze jsou [2]:

- **Bez schématu** - Neexistují zde žádná omezení ve formátu a struktuře ukládání dat, proto lze bez problémů uchovávat data i ve stále se měnícím systému s obrovským množstvím dat.
- **Údržba** - Po vytvoření dokumentu je vyžadována minimální údržba.
- **Nezávislost dokumentů** - Dokumenty na sobě jsou nezávislé kvůli absenci cizích klíčů.
- **Otevřené formáty** - K popisu dokumentů lze použít například formát XML, JSON a mnoho dalších.
- **Věstavěné verzování** - Díky verzování lze snižovat konflikty.

Nevýhody

Nevýhody dokumentové databáze jsou [2]:

- **Kontrola konzistence** - Je zde omezená kontrola konzistence, takže se v databázi můžou vyskytovat duplikáty.
- **Neexistence atomicity** - V případě úpravy, která ovlivňuje dvě kolekce, je potřeba spustit dva samostatné dotazy (jeden pro každou kolekci).

3.4 Existující řešení

Pro vybrané typy databáze existují mnoho databázových řešení, které lze zmínit. V této kapitole se budeme zabývat především těmi nejznámějšími pro daný typ databáze, a které jsou zdarma ke stažení a používání. Pro každý typ databáze bylo vybráno vždy jedno z nejznámějších řešení.

3.4.1 MySQL

MySQL je multiplatformní databáze uplatňující relační databázový model. Komunikace s databází (získávání dat, vytváření objektů, ...) probíhá pomocí jazyka SQL, který je rozšířen o nové funkce. Nejnovější verze MySQL je open-source, což znamená, že kdokoliv může používat a libovolně upravovat MySQL systém, aniž by musel cokoliv platit. V případě změny zdrojových kódů je potřeba nastudovat podmínky užívání definované licencí GPL [15].

Od samých počátků bylo MySQL optimalizováno především na rychlost i za cenu některých zjednodušení (například způsob zálohování dat). Díky tomuto lze provozovat jednoduché servery na počítači společně s jinýma aplikacema, případně jiné databáze. Server lze nakonfigurovat tím způsobem, že může využívat veškerou paměť, procesorový čas i vstupně výstupní kapacity.

MySQL server může být využit dvěma způsoby:

- **Klient / server** - Vícevláknový SQL server, který podporuje různé back-endy, několik různých klientských programů a knihoven a mnoho dalšího.
- **Věstavěná knihovna** - Vícevláknová věstavěná knihovna, kterou lze propojit do své aplikace a získat tím menší, rychlejší a snadněji spravovatelný samostatný produkt.

3.4.2 PostgreSQL

PostgreSQL je open-source objektově-relační databázový systém, který vznikl spojením relačního a objektově-orientovaného databázového systému. Post-

greSQL je velice silný nástroj, který používá rozšíření jazyka SQL společně s mnoha funkcemi, které bezpečně skladují a škálují většinu nejsložitějších datových úloh.

PostgreSQL přichází s mnoha funkcemi, které jsou zaměřené na pomoc vývojářům při vytváření aplikací, správu a bezpečnost dat a odolnost proti chybám v systému. Jako další výhody lze zmínit velkou rozšiřitelnost (lze tvořit vlastní datové typy a funkce), psaní kódu v jiných programovacích jazycích, podpora ACID a možnost provozovat server na všech hlavních operačních systémech [17].

PostgreSQL obsahuje mnoho funkcí, které může uživatel využít. Zde je výpis pouze část z nich:

- **Datové typy** - Primitivní (číslo, text, ...), strukturované (datum, pole, ...), dokumenty (JSON, XML, ...), geometrie (bod, kruh, ...) a vlastní datové typy.
- **Celistvost dat** - Unikátní hodnoty, primární a cizí klíče, zámky.
- **Výkonnost** - Indexování pomocí stromů, výrazů. Základní a vnořené transakce.
- **Zabezpečení** - Vícefaktorová autentikace s certifikáty, sloupcové a řádkové zabezpečení.
- **Textové vyhledávání** - Full-textové vyhledávání.

3.4.3 LevelDB

LevelDB je open-source databáze typu Klíč-hodnota, která se používá hlavně u malých přenosných aplikací a nepotřebují žádné API (rozhraní). Databáze byla vytvořena dvěma programátory Googlu, kteří byli inspirování již existující databází Bigtable (databáze typu klíč-hodnota, která je součástí platformy Google Cloud), ale chtěli vytvořit jednoduchou, lehce přenosnou databázi, kterou lze distribuovat zároveň s aplikací, která ji využívá.

Algoritmus pro ukládání databáze funguje tak, že dočasně ukládá data v *MemTable* (mezipaměť pro zpětný zápis řádků, ve které lze hledat pomocí klíče), ze které se data postupně přesouvají do *SSTable* (Sorted String Table), což je tabulka seřazených řetězců, které nelze měnit. Neměnná data jsou ukládána na disk, který může být sdílen s více clustery [20].

Výhody LevelDB jsou:

- **Jednoduché operace** - LevelDB má tři základní jednoduché operace *Get* (vrací hodnotu podle klíče), *Put* (vkládá dvojici klíč-hodnota) a *Delete* (mazání dvojice klíč-hodnota).
- **Bytové pole** - Klíče a hodnoty lze ukládat i do bytového pole, což může být užitečné v případě, kdy nechceme ukládat hodnoty jako řetězce.
- **Atomické operace** - LevelDB podporuje atomické operace, to znamená, že lze použít více operací najednou v jednom nepřerušeném volání.

3.4.4 MongoDB

MongoDB je dokumentově-orientovaná databáze, která se používá zejména tam, kde je potřeba uchovávat velké množství dat. Firma MongoDB.Inc poskytuje oficiální ovladače ke všem populárním programovacím jazykům jako je například C#, Java, C++ a mnoho dalších. Existují i neoficiální ovladače vytvořené komunitou, které pokrývají ještě více programovacích jazyků.

MongoDB je vlastně ve skutečnosti server, který umožňuje vytvářet a udržovat několik databází najednou. Každá databáze může mít své vlastní kolekce, které sdružují dokumenty. MongoDB podporuje vnořená data, což umožňuje vytvářet složité vztahy mezi dokumenty a ukládat je do stejného dokumentu, což činí práci a načítání dat extrémně efektivní ve srovnání s SQL.[12]

Výhody MongoDB jsou [5]:

- Dokumenty lze mapovat na objekty v kódu aplikace, takže se s nimi dá jednodušeji pracovat.
- Indexování, shlukování v reálném čase a ad-hoc dotazy poskytují velice výkonné způsoby přístupu k datům a jejich analýzy.
- MongoDB nabízí vysokou dostupnost, horizontální škálování a geografickou distribuci a to díky tomu, že je ve svém jádře distribuovanou databází.

3.4.5 Neo4j

Neo4j je open-source nativní grafová databáze, která efektivně implementuje vlastnosti grafového modelu až na úroveň úložiště, které je velmi výkonné.

Pomocí Neo4j lze naimplementovat každý graf, který dokážeme nakreslit na tabuli, za pomoci ukazatelů. Stejně jako pro MongoDB, i zde existují ovladače pro populární programovací jazyky, jako například Java, .NET a mnoho dalších.

Neo4j je velice populární právě z důvodu konstantních časových přechodů ve velkých grafech jak pro prohledávání do šířky, tak i do hloubky, díky efektivní reprezentaci a škálování uzlů a vztahů mezi nimi. Databáze navíc umožňuje vytvářet flexibilní schéma vlastností grafu, které se může v průběhu času přizpůsobovat, díky čemuž lze přidávat nové vztahy pro vytváření zkratek mezi uzly pro zrychlení práce s daty. Neo4j také poskytuje úplné databázové charakteristiky, které zahrnují i ACID vlastnosti, podpory clusterů a převzetí služeb při selhání za běhu [24].

3.5 Jazyky

Databázové jazyky, jinak známé jako dotazovací jazyky, jsou klasifikací programovacích jazyků, které se používají k definování a přístupu k databázím. Pomocí těchto jazyků dokáže uživatel získávat nebo spravovat data v databázích. V dnešní době se jazyky (například SQL) mohou skládat ze čtyř podjazyků, kdy každý slouží k jinému účelu v rámci vykonávání příkazů [10, 25]:

- **Data definition language** (DDL) - DDL umí vytvářet jednotlivé komponenty databázového schématu (tabulky, soubory, indexy, ...), které tvoří strukturu reprezentující organizaci dat v databázi. Dostupné příkazy pro jazyk DDL:
  - **CREATE** - Vytvoření nového objektu (tabulka, index, ...).
  - **ALTER** - Změna struktury objektu.
  - **DROP** - Smazání objektu.
  - **RENAME** - Změna názvu objektu.
  - **TRUNCATE** - Smazání podobjektů v objektu (například záznamy v tabulce).
- **Data manipulation language** (DML) - DML slouží pro manipulaci s daty, které se nachází v již existující databázi. Dostupné příkazy pro jazyk DML:
  - **SELECT** - Získání záznamů (dat) z tabulky.
  - **INSERT** - Vložení nového záznamu (dat) do tabulky.

  - **UPDATE** - Úprava existujícího záznamu v tabulce.
  - **DELETE** - Smazání záznamu z tabulky.

- **Data control language** (DCL) - Pomocí DCL lze kontrolovat přístupy a práva k datům, které jsou uloženy v databázi. Uživateli lze nastavit práva k jednotlivým DML příkazům nad tabulkama / procedurama (například uživatel bude mít přístup pouze k příkazu SELECT nad tabulkou "TABULKA"). Dostupné příkazy pro jazyk DCL:
  - **GRANT** - Přidání práv uživateli nad danou tabulkou / procedurou.
  - **REVOKE** - Odebrání práv uživateli nad danou tabulkou / procedurou.

- **Transaction control language** (TCL) - TCL spravuje transakce v databázi. Transakce obsahuje jeden či více DML příkazů nad tabulkama, které se vykonávají po sobě. Všechny příkazy musí být úspěšně provedeny, aby bylo možné transakci označit za úspěšnou. Ukázka jedné transakce viz obrázek č. 3.10. Dostupné příkazy pro jazyk TCL:
  - **COMMIT** - Potvrzení transakce, změny provedené v transakci jsou permanentní a nejdou vzít zpět.
  - **ROLLBACK** - Vezme zpět veškerou práci v aktuální transakci. Lze se vrátit na začátek transakce nebo k SAVEPOINTu.
  - **SAVEPOINT** - Nastavení bodu v transakci, ke kterému se lze v budoucnu vrátit pomocí ROLLBACK.

```
SQL> SAVEPOINT SP1;
Savepoint created.
SQL> DELETE FROM CUSTOMERS WHERE ID=1;
1 row deleted.
SQL> SAVEPOINT SP2;
Savepoint created.
SQL> DELETE FROM CUSTOMERS WHERE ID=2;
1 row deleted.
SQL> SAVEPOINT SP3;
Savepoint created.
SQL> DELETE FROM CUSTOMERS WHERE ID=3;
1 row deleted.
SQL> ROLLBACK TO SP2;
Rollback complete.
```

Obrázek 3.10: Ukázka jedné transakce (bez commitu)

Níže v kapitolách jsou popsány příklady dnešních jazyků.

3.5.1 Structured Query Language

Structured Query Language (SQL) je jazyk pro komunikaci s databázema, v dnešní době standard pro relační databázové systémy. Pomocí SQL příkazů lze například vytvářet nové objekty v databázi, upravovat existující data v tabulkách nebo vytvářet různá integritní omezení a triggery. Většina existujících databázových systémů používá upravený SQL jazyk, který navíc obsahuje dodatečná rozšíření pro splnění požadavků v jejich systémech.

Syntax

Syntaxe SQL se skládá z unikátního seznamu pravidel a směrnic. Při psaní příkazů zde nehraje roli citlivost písma (příkazy select a SELECT jsou záměnné). Dotazy lze psát na jednu nebo více řádek, které musí / můžou být zakončené středníkem (záleží na pravidlech používaného systému). Na obrázku č. 3.11 lze vidět příklad dotazu, který získá jména a příjmení uživatelů z tabulky 'user' s datem narození po roce 2000.

```
SELECT firstname, lastname FROM user WHERE birthdate_year > 2000;
```

Obrázek 3.11: Příklad SQL dotazu.

Dotazy lze zanořovat do sebe, kdy výsledek jednoho dotazu jde použit jako podmínka pro druhý dotaz, viz obrázek č. 3.12.

```
SELECT * FROM user WHERE user.id = (SELECT id_user FROM meal_orders WHERE id = 1);
```

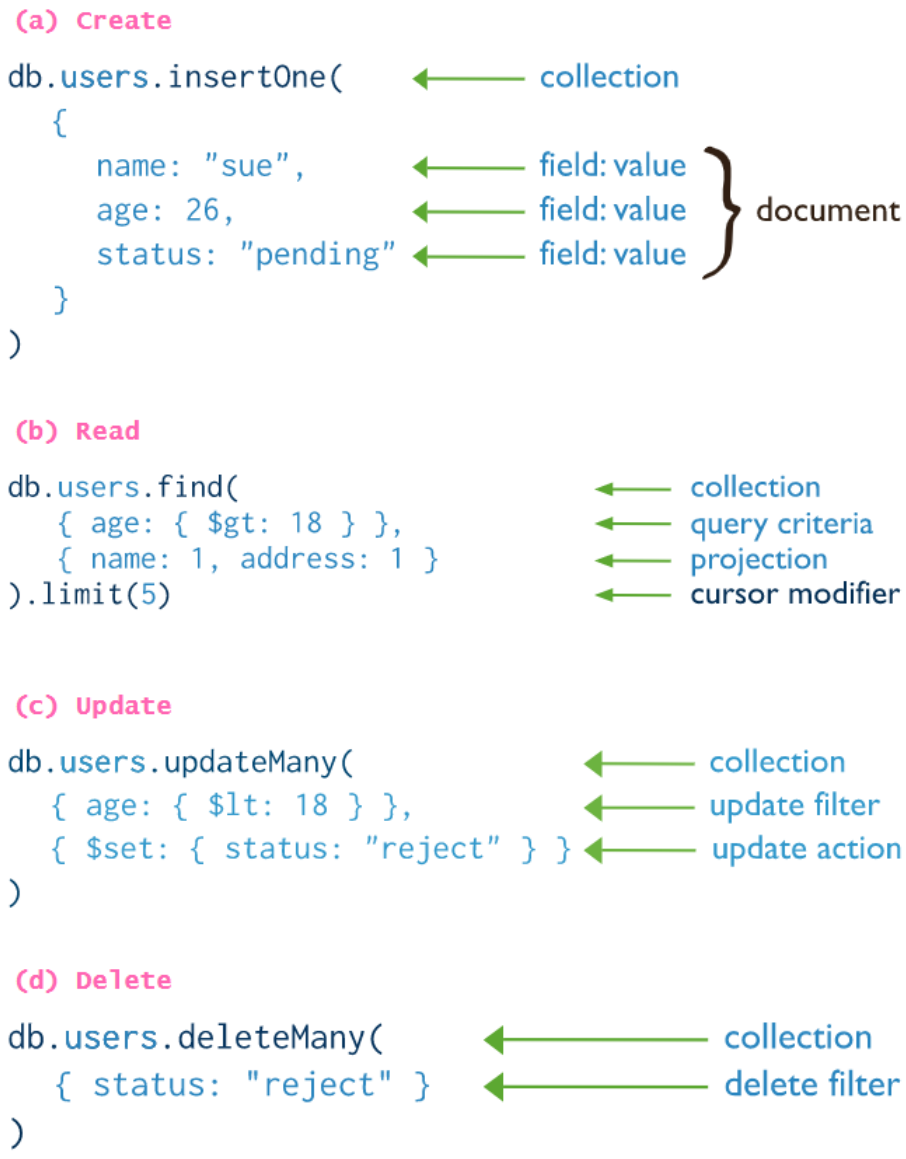
Obrázek 3.12: Příklad zanořeného SQL dotazu.

3.5.2 MongoDB Query Language

MongoDB Query Language (MQL) je jazyk pro získávání dat z MongoDB dokumentových databází. Dotazy zde poskytují jednoduchost v procesu načítání dat z databáze, stejně jako tomu je u SQL. Při provádění dotazů lze také použít kritéria nebo podmínky, kterými lze načíst konkrétní data z databáze. Jazyk také podporuje CRUD operace. Výsledky můžeme třídit, seskupovat, fitrovat a spočítat jejich četnost za pomoci agregační pipeline (zřetězeného zpracování). MQL podporuje transakce více dokumentů [13].

Syntax

Syntaxe MQL je intuitivní a jednoduchá na používání i pro velice složité dotazy, protože ta samá syntaxe se používá i pro uložené dokumenty v databázi. Příklad syntaxe pro vytváření, čtení, úpravu a mazání dokumentů (CRUD) lze vidět na obrázku č. 3.13.



Obrázek 3.13: Příklady CRUD operací v MongoDB [14].

3.5.3 Cypher Query Language

Cypher je dotazovací jazyk pro grafovou databázi Neo4j a umožňuje získávat data z grafů. Tento jazyk byl inspirován hlavně SQL - uživatel se zaměřuje pouze na to, jaká data chce získat, ne jak je má získat. Cypher je unikátní v tom, že poskytuje vizuální způsob, jak sladit vzory a vztahy [6].

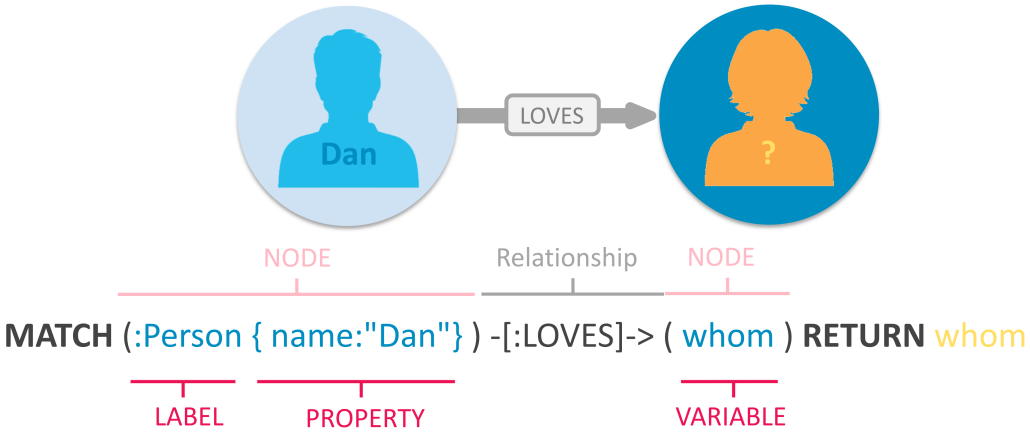
Syntax

Cypher využívá ASCII-art typ syntaxe, což je umění, které pracuje s počítačovým textem jako s výtvarným médiem (například obrázky se skládají ze znaků kódu ASCII). Syntaxi lze vidět na obrázku č. 3.14. Pro jednotlivé uzly se používají kruhové závorky, pro vztah se používá šipka s hranatýma závorkama obsahující vztah prvního uzlu s druhým uzlem.

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

Obrázek 3.14: Syntaxe jazyka Cypher.

Na obrázku č. 3.15 lze vidět jednoduchý dotaz, který hledá výsledný uzel pro vstupní uzel, kterým je člověk se jménem 'Dan', a vztahu 'LOVES' mezi uzly.



Obrázek 3.15: Dotaz v jazyce Cypher [6].

4 Návrh úložiště

Hlavní motivací pro vytvoření této práce je vytěžování patentů pro účely zjišťování existence například různých technologických vynálezů či algoritmů. Pomocí těchto informací lze zjistit, zda například má smysl vymýšlet nový algoritmus pro určitý problém a neexistuje k němu jiné, lepší řešení, případně vymyslet modifikaci, která zajistí lepší výsledky.

Dále je potřebat definovat, co vlastně znamená pojem efektivní vytěžování. Vytěžování lze označit za efektivní, pokud budou splněny tyto podmínky:

- **Rychlost** - Vyhledávání musí probíhat v rámci jednotek až desítek sekund (případně jednotky minut, záleží na celkovém počtu patentů a na hardwarové konfiguraci serveru).
- **Stabilita** - Server musí být stabilní a nesmí padat při práci s velkým množstvím dat, zvlášť při vyhledávání s použitím složitých dotazů (například hledání přes více tabulek).

todo

4.1 Výběr patentů

dopsat něco?

Při výběru patentů byly stanoveny tři podmínky, které museli být splněny:

- **Dostupnost** - Patenty musí být dostupné z online stránek / databází bez poplatků.
- **Datum** - Patentová přihláška nebo publikace patentu musí být podána alespoň v roce 2000, všechny ostatní patenty budou vyfiltrovány.
- **Atributy** - Všechny patenty musí obsahovat povinné atributy (viz kapitola č. 4.1.2).

4.1.1 Zdroje dat

V dnešním světě existuje několik desítek až stovek patentových zdrojů dat, od webových vyhledávačů v databázi až po plný export databáze s patenty. Velké organizace, jako například EPO, WIPO, USPTO, udržují jedny z největších patentových databází (desítky až stovky milionů patentů), ve kterých

lze vyhledávat velké množství informací zdarma za použití webových vyhledávačů na dané stránce organizace. Lze zde najít všechny typy patentů (přihlášky, publikace), národní patenty i patenty registrované například u EPO. V případě exportu databází, USPTO poskytuje plný export svých databází veřejnosti pro libovolné používání, zcela zdarma. Využití těchto zdrojů dat by bylo určitě skvělé, ale tyto zdroje byly nedávno použity a rozebrány v jiné diplomové práci, proto je vhodné se spíše zaměřit na národní zdroje dat patentů.

Národní databáze patentů dané země obsahuje všechny národní patenty, některé dokonce i patenty z jiných zemí registrovaných u EPO.

Země	Patentový úřad	Zkratka
Anglie	Intellectual Property Office	IPO
Arménie	Intellectual Property Office	-
Austrálie	IP Australia	-
Bělorusko	National Center of Intellectual Property	NCIP
Bulharsko	Patent Office of Republic of Bulgaria	-
Česko	Industrial Property Office of the Czech Republic	-
Čína	China National Intellectual Property Administration	CNIPA
Dánsko	Danish Patent and Trademark Office	-
Egypt	Egyptian Patent Office	-
Estonsko	The Estonian Patent Office	-
Filipíny	Intellectual Property Office of the Philippines	IPOPHL
Finsko	Finnish Patent and Registration Office	PRH
Francie	National Institute of Industrial Property	INPI
Hong Kong	Intellectual Property Department	-
Chorvatsko	State Intellectual Property Office of the Republic of Croatia	SIPO
Indie	Office of the Controller General of Patents, Designs and Trade Marks	-
Indonésie	Directorate General of Intellectual Property	DGIP
Irsko	Intellectual Property Office of Ireland	IPOI
Island	Icelandic Intellectual Property Office	ISIPO
Israel	The Israel Patent Office	ILPO
Itálie	Directorate General for the Protection of Industrial Property	-
Japonsko	Japan Patent Office	JPO
Jižní Korea	Korean Intellectual Property Office	KIPO
Kanada	Canadian Intellectual Property Office	CIPO

Tabulka 4.1: Národní patentové úřady a jejich zkratky, část první

Země	Patentový úřad	Zkratka
Kuba	Cuban Industrial Property Office	OCPI
Litva	State Patent Bureau of the Republic of Lithuania	-
Lotyšsko	Patent Office of the Republic of Latvia	-
Maďarsko	Hungarian Intellectual Property Office	HIPO
Malajsie	Intellectual Property Corporation of Malaysia	MyIPO
Mexiko	Instituto Mexicano De La Propiedad Industrial	IMPI
Moldova	State Agency on Intellectual Property	AGEPI
Německo	German Patent and Trade Mark Office	DPMA
Nizozemsko	Netherlands Patent Office	-
Norsko	Norwegian Industrial Property Office	NIPO
Nový Zéland	Intellectual Property Office of New Zealand	IPONZ
Peru	National Institute for the Defense of Competition and Protection of Intellectual Property	INDECOPI
Polsko	Urząd Patentowy Rzeczypospolitej Polskiej	UPRP
Portugalsko	Portuguese Institute of Industrial Property	-
Rakousko	Austrian Patent Office	-
Rumunsko	State Office for Inventions and Trademarks	OSIM
Rusko	Federal Service for Intellectual Property	Rospatent
Řecko	Hellenic Industrial Property Organization	HIPO
Singapur	Intellectual Property Office of Singapore	IPOS
Slovensko	Industrial Property Office of the Slovak Republic	-
Slovinsko	Slovenian Intellectual Property Office	SIPO
Srbsko	Intellectual Property Office of the Republic of Serbia	-
Španělsko	Spanish Patent and Trademark Office	OEPM
Švédsko	Swedish Intellectual Property Office	PRV
Švýcarsko	Swiss Federal Institute of Intellectual Property	-
Turecko	Turkish Patent and Trademark Office	Turkpatent
Ukrajina	Ukrainian Intellectual Property Institute	Ukrpatent

Tabulka 4.2: Národní patentové úřady a jejich zkratky, část druhá

4.1.2 Atributy

Povinné atributy

Země	Název patentu	Rok přihlášky / publikace	Autor	ID patentu
Kanada	x	x	x	x
Česko	x	x	-	x
Litva	x	x	x	x
Portugalsko	x	x	x	x
Španělsko	x	x	x	x
Švédsko	-	x	-	x
Izrael	x	x	x	x
Itálie	x	x	x	x
Mexiko	x	x	x	x
Polsko	x	x	-	-
Anglie	x	x	x	x
Rusko	x	x	x	x
Peru	x	x	x	x
Francie	x	x	x	x

Tabulka 4.3: Povinné atributy nacházející se v dostupných patentech

Nepovinné atributy

Země	Abstrakt	Slovník	Reference	Žadatel
Kanada	-	-	-	x
Česko	x	-	x	-
Litva	x	-	-	x
Portugalsko	x	-	-	x
Španělsko	x	-	-	x
Švédsko	x	-	-	-
Izrael	-	-	-	x
Itálie	-	-	-	x
Mexiko	x	-	-	x
Polsko	x	x	-	-
Anglie	-	-	-	-
Rusko	-	-	-	-
Peru	-	-	-	-
Francie	x	-	x	x

Tabulka 4.4: Nepovinné atributy nacházející se v dostupných patentech, část první

Země	Adresa	Rodina patentů	Obor	Fulltext
Kanada	x	-	x	-
Česko	-	-	x	-
Litva	-	-	x	-
Portugalsko	-	-	x	-
Španělsko	x	-	x	x
Švédsko	-	-	x	x
Izrael	x	-	-	-
Itálie	-	-	-	-
Mexiko	-	-	x	-
Polsko	-	-	-	-
Anglie	-	-	x	-
Rusko	-	-	-	-
Peru	-	-	x	-
Francie	-	-	x	-

Tabulka 4.5: Nepovinné atributy nacházející se v dostupných patentech, část druhá

4.1.3 Závěr průzkumu

Rovnou udělat kapitolu, ve který se aplikují všechny podmínky + se sepíše souhrn počtu patentů, z jakých zemí atp.

4.2 Výběr databáze

V kapitole č. 3 bylo podrobně popsáno co databáze je, jaké typy databází dnes existují (krátký výčet) i nejznámější existující řešení pro popsané typy databází. V této kapitole budou podrobně popsány rozdíly mezi jednotlivými řešeními a následně se vybere nejlepší typ databáze pro ukládání velkého objemu patentových dat. Následně, podle vybraného typu databáze, se vybere nejvhodnější existující řešení.

4.2.1 Výběr typu databáze

Abychom zajistili co nejefektivnější vytěžování, tak je potřeba vybrat co nejvhodnější typ databáze vzhledem k povaze úlohy. V budoucnu se očekává, že počet skladovaných patentů bude v řádech jednotek až desítek milionů (nelze vyvrátit i stovky milionů v případě, že se budou ukládat i patenty z jiných než národních zdrojů).

Relační databáze

Relační databáze není vhodným kandidátem pro ukládání nestrukturovaných patentových dat. V databázi sice existuje datový typ **BLOB**, který umožňuje ukládat binární soubory (v našem případě soubor s patentovými daty), ale nelze to pokládat za nejlepší řešení, když existují například dokumentové databáze. Lze zmínit i datový typ **TEXT** / **LONGTEXT**, který umožňuje ukládát velké množství textu a lze ho procházet pomocí full-text vyhledávání, ale výkonnostně a rychlostně se stejně nevyrovná NoSQL databázím.

Využití relační databáze by mělo smysl pouze v případě vytváření statistik (například počet v patentů v Kanadě za rok 2020). Tento přístup by ale vyžadoval extrahovat specifická data (například jen povinné atributy) ze souboru pomocí parseru a následné uložení hodnot do tabulek.

Objektově-orientovaná databáze

Objektově-orientovaná databáze, stejně jako relační databáze, není vhodným kandidátem pro ukládání nestrukturovaných dat. Lze argumentovat vytvo-

řením objektů odpovídající struktuře dokumentu, ale při vložení dokumentu s jinou strukturou nastává problém s uložením atributů, které se nenachází v objektu. V některých případech může vysoká složitost systému zpomalovat vyhledávání. Velká výhoda objektově-orientované databaze spočívá v jednoduchém mapování objektů při práci s objektově-orientovaném programování, které ale v našem případě nemá využití. V případě vytěžování statistik je objektově-orientovaná databáze horší volbou než relační databáze.

Databáze klíč-hodnota

Databáze klíč-hodnota je jednoduchá a velice rychlá databáze, která umožňuje ukládat i nestrukturovaná data a nepotřebuje k tomu velké množství paměti. Její nevýhoda je ale v ukládání složitých struktur, které soubory s patenty mají. Lze uložit celou strukturu patentu jako hodnotu, ale následné vyhledávání hodnot pomocí názvů parametrů je nemožné. Použití databáze klíč-hodnota v našem případě není moc vhodné.

Grafová databáze

Grafová databáze není vhodným kandidátem pro ukládání patentů, protože se zaměřuje hlavně na vztahy mezi jednotlivými daty, což u patentů nelze a ani není potřeba sledovat.

Databáze dokumentů

Databáze dokumentů, jak už název napovídá, je databáze pro efektivní ukládání dokumentů a jejich vytěžování. Umožňuje ukládat velké množství nestrukturovaných dat, její udržba je snadná a akceptuje dokumenty v několika datových formátech. Její velká nevýhoda je v kontrole konzistence, takže se v databázi mohou vyskytovat duplikáty. I přes tuto nevýhodu je dokumentová databáze vhodným kandidátem pro ukládání patentových dat.

Závěr

Dokumentová databáze bude použita jako primární databáze, protože umožňuje ukládat nestrukturované dokumenty velice efektivně. Zároveň podporuje vkládání dokumentů ve více formátech, což v případě mnoha národních zdrojů, kdy každý zdroj ukládá dat v jiném formátu, je velice vhodná vlastnost. Její nevýhoda, kontrola konzistence, může být odstraněna pomocí jednoduché aplikace, která bude kontrolovat výskyt patentu v databázi podle jeho identifikátoru (ID). Dokumentová databáze podporuje i full-text vyhledávání pro efektivnější a rychlejší vyhledávání.

Relační databáze bude použita jako sekundární databáze pro vytěžování, zaměřená hlavně na tvorbu statistik. Relační databáze je vhodnou volbou pro získávání statistik, protože vyhledávání je velice rychlé a snadné. SQL dotazy pro statistiky se můžou uložit do pohledů, které zjednoduší uživateli práci se zadáváním dotazů. Lze zmínit i nevýhody jako třeba problém s údržbou nebo potřeba velkého množství paměti. Tyto nevýhody ale nehrají velkou roli v případě jednotek až desítek milionů záznamů.

4.2.2 Výběr z existujících řešení

V dnešní době existuje mnoho dokumentových i relačních databází, placených i zdarma poskytovaných. Placené verze oproti verzím zdarma mají výhodu v lepší podpoře ze strany vývojářů, obsahují více užitečných funkcí a mají lepší zabezpečení. Pro naše účely bohatě postačí verze zdarma.

Jako existující řešení databáze dokumentů byla vybrána komunitní verze databáze MongoDB. Komunitní verze je zdarma a server lze provozovat jak lokálně, tak i na cloudu, kde MongoDB poskytuje zdarma uložiště o velikosti 512 MB. Pro lepší a spolehlivější vyhledávání v datech bude MongoDB spojena s vyhledávačem Elasticsearch. Elasticsearch je full-textový open-source vyhledávač, který nabízí vysokou dostupnost, rychlost a škálovatelnost. MongoDB sice obsahuje vlastní full-textový vyhledávač, který ale není tak výkonný jako Elasticsearch.

Jako existující řešení relační databáze byla vybrána komunitní verze databáze MySQL. MySQL je skvělá databáze, která se používá hlavně pro čtení dat. Zároveň je to jedna z nejpoužívanějších relačních databází, což znamená, že je pro ní k dispozici více nástrojů třetích stran.

5 Implementace úložiště

todo

5.1 Adresářová struktura

todo

5.1.1 Patenty

5.1.2 Docker

5.2 Implementace databáze

todo

5.2.1 MySQL

5.2.2 MongoDB

5.3 Výsledné úložiště

todo

5.3.1 Technologické požadavky

5.3.2 Docker

Images, scripty
Import dat (skripty, data soubory, ...)

5.3.3 Inicializace MySQL

5.3.4 Inicializace MongoDB

6 Rozšiřitelnost úložiště

Zadání diplomové práce sice splněno bylo, ale v blízké budoucnosti mohou být požadavky na modul změněny. Jako příklad lze uvést podporu přidávání nových patentů do databází, zjištění autorů pro české patenty, automatické stahování dat z již ověřených patentových zdrojů. V této kapitole jsou popsány 3 možné návrhy na rozšíření modulu ohledně importu dat do již existujících databází.

6.1 Přidávání nových patentů

Cílem tohoto rožšíření by bylo automatické přidávání patentů z datových souborů jak do MySQL databáze, tak i do Mongo.

Rozšíření by se dalo realizovat jako aplikace ve vyšším programovacím jazyku (např. Java, C), kdy vstupem do aplikace by byl soubor v datovém formátu JSON/XML/CSV a jiné. Vstupní soubor by se následně:

- převedl na JSON řetězec (v případě že soubor není ve formátu JSON) a vložil do Mongo databáze
- rozparsoval a extrahovali by se všechny atributy, které se ukládají v MySQL databázi (viz mysql kapitola) TODO

Jelikož je dost časté, že každý národní zdroj dat používá odlišnou strukturu patentu, tak bude potřeba aplikaci neustále upravovat (ať už v rámci přidávání nových zdrojů, nebo v případě změny struktury patentu u již podporovaných zdrojů).

Jako další velký problém lze zmínit extrakci atributů patentu ze souborů. Tím, že různé patentové soubory mají odlišnou strukturu, to znamená hloubku zanoření specifických elementů, jiné názvy elementů, tak bude obtížné naimplementovat řešení extrakce pro všechny soubory. Tento problém by se dal řešit tak, že se vytvoří soubory se slovníkama, které by obsahovali názvy elementů pro daný atribut. Slovníky by se následně použily při extrakci.

6.2 Zjišťování autorů pro české patenty

Český národní patentový úřad poskytuje data o českých patentech, které ale neobsahují autora ani instituci. Pro zjištění autora nebo instituce, která

patent registrovala, je nutné použít oficiální vyhledávač. Cílem tohoto rozšíření by bylo vytvořit aplikaci ve vyšším programovacím jazyku, která se pro všechny české patenty bude snažit najít jejich autory za pomoci využití prohledávačů webů (web crawler). Postupů řešení může být mnoho:

- Zjišťování autorů by se provedlo pro všechny existující české patenty v databázi. Z MySQL databáze se zjistí všechny identifikátory pro české patenty, které se následně použijí jako vstup pro web crawler.
- Zjišťování autorů by se provedlo pro patent/y uložené v souboru, kdy aplikace by pro všechny patenty v souboru zjistila autory a následně je dopsala do příslušnýho elementu patentu v daném souboru.
- Stejný postup jako předchozí s tím rozdílem, že po zjištění autora se patent rovnou přidá do MySQL i Mongo databáze.

6.3 Automatické stahování dat z ověřených zdrojů

Cílem tohoto rozšíření by bylo automatické stahování dat (případně i jejich parsování) z ověřených zdrojů. Ověřené zdroje by byly uloženy například v XML souboru, kdy každý zdroj by měl tyto položky:

- **Název země**
- **URL** - URL zdroje dat, na které lze stáhnout data.
- **XPath** - XPath výraz, pomocí kterého lze ze stránky vyfiltrovat a získat odkazy ke stažení dat
  (např. */html/body//a[contains(@href,'example')]/@href*)
- **Poslední verze** - Název / číslo poslední stažené verze.

XML soubor by byl následně zpracován pomocí aplikace (např. Java, C#), která by následně pro každý zdroj dat provedla následující kroky:

1. Získání seznamu odkazů na zdroje dat.
2. Stažení všech zdrojů dat, jejichž verze je větší než aktuálně uložená verze v XML.
3. V tomto bodě se dá naimplementovat cokoliv - např. lze uložená data extrahovat ze ZIP souborů, importovat patenty do databází (viz kapitola č. 6.1), pouze notifikace o stažení několika nových souborů z daty a mnoho dalšího.

4. Aktualizace verze v XML souboru.

Automatizace stahování dat by spočívala ve spouštění aplikace pro stahování dat v pravidelných intervalech (např. každé druhé úterý v 17:00). Jako příklad lze uvést použití pipeline na Jenkins serveru, který bude spouštět z lokálního uložiště spustitelnou aplikaci v daný čas (pomocí CRON). Po vykonání celého procesu může Jenkins poslat email o stavu posledního spuštění (zda se spuštění povedlo, kolik souborů byl schopen stáhnout pro jaké země, ...). Samozřejmě bohatě postačí i použití plánovače v operačním systému.

7 Ověření efektivního vytěžování

K ověření efektivního vytěžování bylo připraveno několik scénářů jak pro SQL, tak i pro Mongo + ElasticSearch.
Napsat referenční stroj na kterém se testovalo - CPU, RAM, ... todo

7.1 Mongo + ElasticSearch

7.2 MySQL

Pro MySQL bylo připraveno 10 scénářů, které testují všechny vytvořené tabulky v databázi. Každý scénář obsahuje textový popis, SQL příkaz, rychlost vykonání příkazu a ukázku výsledků. vypsat počet hodnot v tabulkách

7.2.1 Scénář č.1

Textový popis: Deset nejčastěji patentujících institucí v Izraeli v roce 2015.
SQL:

```
select count(*), count(*) * 100.0 / ((select count(*) from inventors) *
    1.0) as percentage, inventors.inventor from inventors left outer join
     patents on inventors.id_patent = patents.id where YEAR(patents.
    patent_date) = 2015 and patents.patent_id like '%IL%' group by
    inventors.inventor order by count(*) desc, percentage desc LIMIT 10;
```

Rychlost vykonání dotazu: TODO
Výsledek dotazu: TODO

count(*)	percentage	inventor
39	0.00119	DOW AGROSCIENCES LLC
29	0.00088	F. HOFFMANN-LA ROCHE AG
29	0.00088	GENENTECH, INC.
29	0.00088	NOVARTIS AG
24	0.00073	RAYTHEON COMPANY
22	0.00067	QUALCOMM INCORPORATED
20	0.00061	BIOSENSE WEBSTER (ISRAEL) LTD.
17	0.00052	MICROSOFT CORPORATION
17	0.00052	SANOFI
16	0.00049	YERKES, CARLA N.

Obrázek 7.1: Ukázka výsledku dotazu pro scénář č.1

7.2.2 Scénář č.2

Textový popis: Tři nejméně patentované obory v Kanadě od roku 2010.
SQL:

```
select count(*), count(*) * 100.0 / ((select count(*) from classification
    ) * 1.0) as percentage, classification.section from classification
    left outer join patents on patents.id = classification.id_patent
    where YEAR(patents.patent_date) >= 2010 and patents.patent_id like '%
    CA%' group by classification.section order by count(*) asc,
    percentage asc LIMIT 5;
```

Rychlost vykonání dotazu: TODO
Výsledek dotazu: TODO

count(*)	inventor
8	Gould Nigel
4	Philips Andrew
4	Lee Sangik
3	Weisman Paul Thomas
3	Dreher Andreas Josef
3	Sivik Mark Robert
3	Park Bio
3	Kim Jeongyun
3	Hamad-Ebrahimpour Alyssandrea Hope
3	Kim Seonghwan
3	Gordon Gregory Charles
3	Trokhan Paul Dennis
2	Watson David John
2	Waite David
2	Payne John Lynn

Obrázek 7.2: Ukázka výsledku dotazu pro scénář č.2

7.2.3 Scénář č.3

Textový popis: Dvacet nejčastějších klasifikací patentů za rok 2008 ve Španělsku.
SQL:

```
select count(*), count(*) * 100.0 / ((select count(*) from classification
    ) * 1.0) as percentage, classification.section, classification.class,
     classification.subclass from classification left outer join patents
    on patents.id = classification.id_patent where YEAR(patents.
    patent_date) = 2008 and patents.patent_id LIKE '%ES%' group by
    classification.section, classification.class, classification.subclass
     order by count(*) desc, percentage desc LIMIT 20;
```

Rychlost vykonání dotazu: TODO
Výsledek dotazu: TODO

count(*) ▼ 1	percentage	section	class	subclass
73	0.00481	B	65	D
57	0.00376	E	04	G
47	0.00310	A	61	K
45	0.00296	E	04	B
43	0.00283	G	01	N
34	0.00224	E	06	B
33	0.00217	F	24	J
30	0.00198	A	23	L
27	0.00178	C	02	F
27	0.00178	A	47	C
27	0.00178	B	60	R
27	0.00178	E	04	H
26	0.00171	A	47	L
26	0.00171	D	06	F
25	0.00165	F	03	D
24	0.00158	A	01	K
23	0.00152	H	05	B
23	0.00152	B	01	D
22	0.00145	B	65	B
22	0.00145	E	04	C

Obrázek 7.3: Ukázka výsledku dotazu pro scénář č.3

7.2.4 Scénář č.4

Textový popis: Deset autorů s největším počtem patentů ze všech zemí.
SQL: divnej dotaz

```
select count(*), count(*) * 100.0 / ((select count(*) from inventors) *
    1.0) as percentage, inventors.inventor from inventors left outer join
```

```
    patents on patents.id = inventors.id_patent group by inventors.
   inventor order by count(*) desc, percentage desc LIMIT 10;
```

Rychlost vykonání dotazu: TODO
Výsledek dotazu: TODO

count(*)	inventor
8	Gould Nigel
4	Philips Andrew
4	Lee Sangik
3	Weisman Paul Thomas
3	Dreher Andreas Josef
3	Sivik Mark Robert
3	Park Bio
3	Kim Jeongyun
3	Hamad-Ebrahimpour Alyssandrea Hope
3	Kim Seonghwan
3	Gordon Gregory Charles
3	Trokhan Paul Dennis
2	Watson David John
2	Waite David
2	Payne John Lynn

Obrázek 7.4: Ukázka výsledku dotazu pro scénář č.4

7.2.5 Scénář č.5

Textový popis: Pět nejméně používaných jazyků pro patenty za rok 2003.
SQL:

```
select count(*), count(*) * 100.0 / ((select count(*) from patents) *
    1.0) as percentage, patents.language from patents where patents.
    language not like '%-%' group by patents.language order by count(*)
    asc, percentage asc LIMIT 5;
```

Rychlost vykonání dotazu: TODO
Výsledek dotazu: TODO

count(*)	inventor
8	Gould Nigel
4	Philips Andrew
4	Lee Sangik
3	Weisman Paul Thomas
3	Dreher Andreas Josef
3	Sivik Mark Robert
3	Park Bio
3	Kim Jeongyun
3	Hamad-Ebrahimpour Alyssandrea Hope
3	Kim Seonghwan
3	Gordon Gregory Charles
3	Trokhan Paul Dennis
2	Watson David John
2	Waite David
2	Payne John Lynn

Obrázek 7.5: Ukázka výsledku dotazu pro scénář č.5

7.2.6 Scénář č.6

Textový popis: Deset Institucí / autorů s patenty pokrývající největší množství oborů.
SQL:

```
select count(distinct classification.section), count(*) * 100.0 / ((
    select count(*) from inventors) * 1.0) as percentage, inventors.
    inventor from inventors left outer join classification on
    classification.id_patent = inventors.id_patent where section is not
    null group by inventors.inventor order by count(distinct
    classification.section) desc, percentage desc LIMIT 10;
```

Rychlost vykonání dotazu: TODO
Výsledek dotazu: TODO

count(*)	inventor
8	Gould Nigel
4	Philips Andrew
4	Lee Sangik
3	Weisman Paul Thomas
3	Dreher Andreas Josef
3	Sivik Mark Robert
3	Park Bio
3	Kim Jeongyun
3	Hamad-Ebrahimpour Alyssandrea Hope
3	Kim Seonghwan
3	Gordon Gregory Charles
3	Trokhan Paul Dennis
2	Watson David John
2	Waite David
2	Payne John Lynn

Obrázek 7.6: Ukázka výsledku dotazu pro scénář č.6

7.2.7 Scénář č.7

Textový popis: Pět zemí s nejvíce patenty od roku 2018.
SQL:

```
select count(*), count(*) * 100.0 / ((select count(*) from patents) *
    1.0) as percentage, patents.country from patents where YEAR(patents.
    patent_date) >= 2018 group by patents.country order by count(*) desc,
     percentage desc LIMIT 5;
```

Rychlost vykonání dotazu: TODO
Výsledek dotazu: TODO

count(*)	inventor
8	Gould Nigel
4	Philips Andrew
4	Lee Sangik
3	Weisman Paul Thomas
3	Dreher Andreas Josef
3	Sivik Mark Robert
3	Park Bio
3	Kim Jeongyun
3	Hamad-Ebrahimpour Alyssandrea Hope
3	Kim Seonghwan
3	Gordon Gregory Charles
3	Trokhan Paul Dennis
2	Watson David John
2	Waite David
2	Payne John Lynn

Obrázek 7.7: Ukázka výsledku dotazu pro scénář č.7

7.2.8 Scénář č.8

Textový popis: Nejvíce používaný jazyk pro patenty ve Francii.
SQL:

```
select count(*), count(*) * 100.0 / ((select count(*) from patents) *
    1.0) as percentage, patents.language from patents where patents.
    patent_id like '%FR%' group by patents.language order by count(*)
    desc, percentage desc;
```

Rychlost vykonání dotazu: TODO
Výsledek dotazu: TODO

count(*)	inventor
8	Gould Nigel
4	Philips Andrew
4	Lee Sangik
3	Weisman Paul Thomas
3	Dreher Andreas Josef
3	Sivik Mark Robert
3	Park Bio
3	Kim Jeongyun
3	Hamad-Ebrahimpour Alyssandrea Hope
3	Kim Seonghwan
3	Gordon Gregory Charles
3	Trokhan Paul Dennis
2	Watson David John
2	Waite David
2	Payne John Lynn

Obrázek 7.8: Ukázka výsledku dotazu pro scénář č.8

7.2.9 Scénář č.9

Textový popis: Patnáct nejčastěji patentujících institucí / autorů v Anglii v textilním oboru za rok 2013.
SQL:

```
select count(*), count(*) * 100.0 / ((select count(*) from inventors) *
    1.0) as percentage, inventors.inventor from inventors left outer join
     patents on patents.id = inventors.id_patent left outer join
    classification on classification.id_patent = patents.id where
    classification.section like '%D%' and patents.patent_id like '%GB%'
    and YEAR(patents.patent_date) = 2013 group by inventors.inventor
    order by count(*) desc, percentage desc LIMIT 15;
```

Rychlost vykonání dotazu: TODO

Výsledek dotazu: TODO

count(*) ▼ 1	inventor
8	Gould Nigel
4	Philips Andrew
4	Lee Sangik
3	Weisman Paul Thomas
3	Dreher Andreas Josef
3	Sivik Mark Robert
3	Park Bio
3	Kim Jeongyun
3	Hamad-Ebrahimpour Alyssandrea Hope
3	Kim Seonghwan
3	Gordon Gregory Charles
3	Trokhan Paul Dennis
2	Watson David John
2	Waite David
2	Payne John Lynn

Obrázek 7.9: Ukázka výsledku dotazu pro scénář č.9

8 Závěr

todo

Zkratky

ACID Atomicity, Consistency, Isolation, Durability 4, 8, 16, 18

API Application Programming Interface 16

ASCII American Standard Code for Information Interchange 22

CRUD Create, Read, Update, Delete 20, 21

CSV Comma-separated values 33

DBMS Database Management Systems 3

EPO European Patent Office 23, 24

GPL GNU General Public License 15

JSON JavaScript Object Notation 10, 14, 16, 33

MQL MongoDB Query Language 20, 21

SQL Structured Query Language 10, 15, 16, 18, 20, 22, 31

URL Uniform Resource Locator 34

USPTO United States Patent and Trademark Office 23, 24

WIPO World Intellectual Property Organization 23

XML Extensible Markup Language 14, 16, 33–35

Literatura

[1] *What Is a Database?* [online]. Oracle. [cit. 21.04.2022]. Dostupné z: https://www.oracle.com/database/what-is-database/.

[2] *What is a Document Database?* [online]. phoenixNAP, 2021. [cit. 27.04.2022]. Dostupné z: https://phoenixnap.com/kb/document-database.

[3] *What Is a Graph Database?* [online]. phoenixNAP, 2021. [cit. 27.04.2022]. Dostupné z: https://phoenixnap.com/kb/graph-database.

[4] *What is a MongoDB Query?* [online]. GeeksforGeeks, 2021. [cit. 27.04.2022]. Dostupné z: https://www.geeksforgeeks.org/key-value-data-model-in-nosql/.

[5] *What Is MongoDB?* [online]. MongoDB. [cit. 28.04.2022]. Dostupné z: https://www.mongodb.com/what-is-mongodb.

[6] *Cypher Query Language* [online]. Neo4j. [cit. 26.04.2022]. Dostupné z: https://neo4j.com/developer/cypher/.

[7] *Graph Modeling Guidelines* [online]. Neo4j. [cit. 27.04.2022]. Dostupné z: https://neo4j.com/developer/guide-data-modeling/.

[8] *Basic Object Oriented Data Model* [online]. GeeksforGeeks, 2021. [cit. 27.04.2022]. Dostupné z: https://www.geeksforgeeks.org/basic-object-oriented-data-model/.

[9] *What is Database Management System (DBMS)? – FAQ, Types, and Details* [online]. erp-information. [cit. 27.04.2022]. Dostupné z: https://www.erp-information.com/database-management-system.html.

[10] *Types of Database Languages and Their Uses (Plus Examples)* [online]. indeed, 2021. [cit. 21.04.2022]. Dostupné z: https://www.indeed.com/career-advice/career-development/database-languages.

[11] *The Types of Databases (with Examples)* [online]. Matillion, 2018. [cit. 22.04.2022]. Dostupné z: https://www.matillion.com/resources/blog/the-types-of-databases-with-examples.

[12] *What is MongoDB – Working and Features* [online]. GeeksforGeeks, 2021. [cit. 28.04.2022]. Dostupné z: https://www.geeksforgeeks.org/what-is-mongodb-working-and-features/.

[13] *What is a MongoDB Query?* [online]. GeeksforGeeks, 2021. [cit. 26.04.2022]. Dostupné z: https://www.geeksforgeeks.org/what-is-a-mongodb-query/.

[14] *MongoDB Query Language* [online]. Devopedia, 2021. [cit. 27.04.2022]. Dostupné z: https://devopedia.org/mongodb-query-language.

[15] *What is MySQL?* [online]. MySQL. [cit. 26.04.2022]. Dostupné z: https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html.

[16] *What is a Graph Database?* [online]. Neo4j. [cit. 27.04.2022]. Dostupné z: https://neo4j.com/developer/graph-database/.

[17] *About PostgreSQL* [online]. PostgreSQL. [cit. 28.04.2022]. Dostupné z: https://www.postgresql.org/about/.

[18] Akhtar, Z. *Relational Database Benefits and Limitations (Advantages & Disadvantages)* [online]. DatabaseTown, 2021. [cit. 27.04.2022]. Dostupné z: https://databasetown.com/relational-database-benefits-and-limitations/.

[19] Gulati, V. *Relational Model in DBMS* [online]. Scaler, 2022. [cit. 26.04.2022]. Dostupné z: https://www.scaler.com/topics/dbms/relational-model-in-dbms/.

[20] Karki, S. *What Is LevelDB* [online]. C# Corner, 2021. [cit. 28.04.2022]. Dostupné z: https://www.c-sharpcorner.com/article/what-is-leveldb2/.

[21] Lobel, L. *Relational Databases vs. NoSQL Document Databases* [online]. WordPress, 2015. [cit. 27.04.2022]. Dostupné z: https://lennilobel.wordpress.com/2015/06/01/relational-databases-vs-nosql-document-databases/.

[22] Lutkevich, B. *database (DB)* [online]. Tech Target, 2021. [cit. 21.04.2022]. Dostupné z: https://www.techtarget.com/searchdatamanagement/definition/database.

[23] Peterson, R. *What is a Database? Definition, Meaning, Types with Example* [online]. Guru99, 2022. [cit. 21.04.2022]. Dostupné z: https://www.guru99.com/introduction-to-database-sql.html.

[24] Sekhon, S. *What is Neo4j?* [online]. DEV Community, 2020. [cit. 28.04.2022]. Dostupné z: https://dev.to/sukhbirsekhon/what-is-neo4j-8jc.

[25] Singh, C. *DBMS languages* [online]. BeginnersBook, 2015. [cit. 21.04.2022]. Dostupné z: https://beginnersbook.com/2015/04/dbms-languages/.

[26] Thakur, D. *What is Object Oriented Database (OODB)? Advantages and Disadvantages of OODBMSS.* [online]. ComputerNotes. [cit. 27.04.2022]. Dostupné z: https://ecomputernotes.com/database-system/adv-database/object-oriented-database-oodb.

[27] Williams, A. *NoSQL Document-Oriented Databases: A Detailed Overview* [online]. RavenDB, 2021. [cit. 27.04.2022]. Dostupné z: https://ravendb.net/articles/nosql-document-oriented-databases-detailed-overview.

A Uživatelská dokumentace

B Vzhled modulů