

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

## **Diplomová práce**

# **Tvorba rozsáhlých úložišť patentových dat**

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 12. května 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

<b>1</b>	<b>Úvod</b>	<b>1</b>
<b>2</b>	<b>Patent</b>	<b>2</b>
2.1	Typy patentů . . . . .	3
2.1.1	Užitný patent . . . . .	3
2.1.2	Návrhový patent . . . . .	4
2.1.3	Patent rostlin . . . . .	5
2.2	Obory . . . . .	5
<b>3</b>	<b>Databáze</b>	<b>7</b>
3.1	Systém řízení báze dat . . . . .	7
3.2	Komponenty databáze . . . . .	8
3.3	Typy databází . . . . .	8
3.3.1	Relační databáze . . . . .	8
3.3.2	Objektově-orientovaná databáze . . . . .	12
3.3.3	NoSQL databáze . . . . .	14
3.3.4	Databáze Klíč-Hodnota . . . . .	14
3.3.5	Grafová databáze . . . . .	15
3.3.6	Databáze dokumentů . . . . .	18
3.4	Existující řešení . . . . .	19
3.4.1	MySQL . . . . .	19
3.4.2	PostgreSQL . . . . .	20
3.4.3	LevelDB . . . . .	20
3.4.4	MongoDB . . . . .	21
3.4.5	Neo4j . . . . .	22
3.5	Jazyky . . . . .	22
3.5.1	Structured Query Language . . . . .	24
3.5.2	MongoDB Query Language . . . . .	25
3.5.3	Cypher Query Language . . . . .	26
<b>4</b>	<b>Návrh úložiště</b>	<b>28</b>
4.1	Výběr patentů . . . . .	28
4.1.1	Zdroje dat . . . . .	29
4.1.2	Atributy . . . . .	32
4.1.3	Analýza dat . . . . .	36
4.1.4	Závěr průzkumu . . . . .	39

4.2	Výběr databáze . . . . .	41
4.2.1	Výběr typu databáze . . . . .	41
4.2.2	Výběr z existujících řešení . . . . .	43
<b>5</b>	<b>Implementace úložiště</b>	<b>44</b>
5.1	Implementace databáze . . . . .	44
5.1.1	MongoDB . . . . .	44
5.1.2	MySQL . . . . .	45
5.2	Nasazení úložiště . . . . .	47
5.2.1	Docker . . . . .	47
5.2.2	Inicializace MySQL . . . . .	50
5.2.3	Inicializace MongoDB . . . . .	50
5.2.4	Propojení MongoDB a Elasticsearch . . . . .	50
<b>6</b>	<b>Rozšiřitelnost úložiště</b>	<b>53</b>
6.1	Přidávání nových patentů . . . . .	53
6.2	Zjišťování autorů pro české patenty . . . . .	54
6.3	Automatické stahování dat z ověřených zdrojů . . . . .	54
<b>7</b>	<b>Ověření efektivního vytěžování</b>	<b>56</b>
7.1	Mongo + ElasticSearch . . . . .	56
7.1.1	Scénář č.1 . . . . .	56
7.1.2	Scénář č.2 . . . . .	57
7.1.3	Scénář č.3 . . . . .	57
7.2	MySQL . . . . .	58
7.2.1	Scénář č.1 . . . . .	58
7.2.2	Scénář č.2 . . . . .	59
7.2.3	Scénář č.3 . . . . .	60
7.2.4	Scénář č.4 . . . . .	61
7.2.5	Scénář č.5 . . . . .	62
7.2.6	Scénář č.6 . . . . .	63
7.2.7	Scénář č.7 . . . . .	64
7.2.8	Scénář č.8 . . . . .	64
7.2.9	Scénář č.9 . . . . .	65
<b>8</b>	<b>Závěr</b>	<b>66</b>
	<b>Zkratky</b>	<b>67</b>
	<b>Literatura</b>	<b>68</b>

<b>A</b>	<b>Uživatelská dokumentace</b>	<b>72</b>
A.1	Adresářová struktura . . . . .	72
A.2	Instalace úložiště . . . . .	73
A.2.1	Smazání setup kontejnerů . . . . .	74
A.3	Import dat do MySQL . . . . .	76
A.4	Import dat do MongoDB . . . . .	79



# 1 Úvod

Patenty jsou v dnešní době nedílnou součástí zákonné ochrany nových technologií a vynálezů v průmyslu a jejich odvětví. Vlastník takového patentu má výhradní právo využívat vynález a poskytovat souhlas o využití patentu jiným stranám. Patenty jsou registrovány a spravovány jak v rámci jednotlivých zemí (národní patentové instituce), tak i mezinárodně (například Evropský patentový úřad EPO). Tyto instituce veřejně poskytují informace o registrovaných patentech pomocí vyhledávačů na jejich webových stránkách. Pro vědce a vynálezce jsou tyto vyhledávače velice důležité právě proto, aby si mohli zkontrolovat, zda jejich vynález je jedinečný, nebo už ve světě existuje v lehce upravené podobě a je patentován.

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. Stažená data budou filtrována na základě specifikovaných atributů, které musí obsahovat (protože ve struktuře existuje element pro povinný atribut ještě neznamená, že pro něj existuje reálná hodnota).

## 2 Patent

Patent je grant od vlády, který dává vynálezci právo vyloučit ostatní z výroby, používání, prodeje, dovozu nebo nabízení vynálezu pro prodej na dobu určitou. Vynález může mít široký význam. Je to jakýkoliv nový článek, stroj, složení hmoty, proces nebo nové použití vyvinuté člověkem. Slovo **proces** je definováno zákonem jako proces, akt nebo metoda a zahrnuje především průmyslové nebo technické procesy. Termín **výroba** se vztahuje na výrobky, které jsou vyrobeny, a zahrnuje všechny vyrobené předměty. Termín **složení hmoty** se vztahuje k chemickému složení a může zahrnovat směsi složek i nové chemické sloučeniny. Tyto třídy předmětů dohromady zahrnují prakticky vše, co je vyrobeno člověkem a procesy výroby výrobků [22, 36].

Patentový zákon stanovuje, že předmět musí být "užitečný". Výraz "užitečný" v této souvislosti odkazuje na podmínku, že předmět má užitečný účel a zahrnuje také operativnost. To znamená, že stroj, který nebude fungovat k plnění zamýšleného účelu, by nebyl nazván užitečným, a proto by mu nebyl udělen patent. Patent nelze udělit ani na zákony přírody, fyzikální jevy a abstraktní myšlenky.

Patent nelze získat na základě pouhého nápadu nebo návrhu. Patent se uděluje na nový stroj, výrobu a jiné, ale ne na myšlenku nebo návrh nového stroje. Vyžaduje se úplný popis skutečného stroje nebo jiného předmětu, pro který se žádá o patent [22].

Patenty vydává vždy národní patentový úřad nebo mezinárodní organizace zabývajícími se patenty. Pro vytvoření patentu je nutné vyplnit žádost. Žádost lze rozdělit na dva typy: provizorní a neprovizorní. Provizorní žádost je jednodušší forma té neprovizorní a slouží jen pro oddálení vytvoření patentu (hlavně z důvodu nedokončeného vývoje vynálezu). Pro přechod z provizorní na neprovizorní je potřeba vytvořit novou žádost, která může vést k publikaci patentu v případě, že splňuje všechny podmínky pro vytvoření.

Patentové právo trvá 20 let od podání přihlášky, po skončení patentového práva může kdokoli volně kopírovat vynález. Patent je forma osobního majetku a proto může být prodáván, nebo jeho vlastník může dát komukoli povolení (neboli licenci) k používání vynálezu, které se může taky platit jako licenční poplatky. Patent také může být převeden darem, vůlí a nebo ho lze získat dědičně podle zákonů o zákonné dědické posloupnosti (bez vůle) státu [36].

## 2.1 Typy patentů

Patenty jsou klasifikovány do tří hlavních typů: **Užitný patent**, **Návrhový patent** a **Patent rostlin**. V následujících kapitolách jsou tyto typy podrobně popsány a srovnány s jejich právně jednoduššími protějšky.

### 2.1.1 Užitný patent

Užitný patent, nebo také patent na vynález, poskytuje právní ochranu lidem, kteří vynalézají nový a užitečný proces, výrobní předmět, stroj, složení hmoty nebo užitečné vylepšení. Užitný patent přetrvává 20 let od data podání, pokud jsou placeny udržovací poplatky. Užitné patenty jsou nejběžnějším typem patentů, a lze ho dále rozdělit na 2 podtypy: **Trvalý** a **Dočasný**. [24, 36].

#### Struktura přihlášky užitného patentu

Struktura užitného patentu obsahuje sedm různých sekcí (v případě USPTO), které patent musí obsahovat (pro dočasný patent jsou povinné pouze dvě sekce) [41]:

- **Pozadí** - Sekce **Pozadí** popisuje současný stav techniky týkající se navrhovaného vynálezu a neměl by se o tomto vynálezu zmiňovat. **Pozadí** musí být velmi stručné a nesmí být zveřejněno více informací, než je požadováno.
- **Stručný přehled** - Sekce **Stručný přehled** shrnuje oddíl **Podrobný popis** do několika odstavců.
- **Stručný popis nákresů** - Sekce **Stručný popis nákresů** obsahuje široký přehled nákresů v oddílu **Nákresy**. Lze zde zmínit a identifikovat části vynálezu v každém z obrázků nebo například uvádět různé pohledy nákresů (perspektivní, pohled zleva a zprava, ...).
- **Podrobný popis** - Sekce **Podrobný popis** popisuje různá alternativní provedení a vlastnosti vynálezu a měla by obsahovat referenční číslce, které odkazují na obrázky v oddílu **Nákresy**. Tato je povinná i pro dočasnou přihlášku.
- **Nároky** - Sekce **Nároky** popisuje rozsah ochrany poskytované patentem nebo ochranu požadovanou v patentové přihlášce.
- **Abstrakt** - Sekce **Abstrakt** slouží jako krátké shrnutí o čem patent je. Abstrakt nesmí být delší jak 150 slov.

- **Nákresy** - Sekce **Nákresy** obsahují nákresy popisující vlastnosti vynálezu v různých náhledech (perspektivní, ...). Tato je povinná i pro dočasnou přihlášku.

## **Užitný patent vs Užitný vzor**

Patentová ochrana není jedinou formou průmyslově právní ochrany vynálezu. Pro vynálezy s nižší vynálezeckou úrovní je možné zvolit **Užitný vzor**, který je jednodušší, rychlejší a méně nákladnou alternativou k užitnému patentu.

### **2.1.2 Návrhový patent**

Návrhový patent, jinak zvaný jako Průmyslový vzor, je patent vydaný pro originální, nové a ornamentální vzory pro vyráběné výrobky. Vzhledem k tomu, že se návrhový patent projevuje vzhledem, atk se předmět patentové přihlášky může týkat konfigurace nebo tvaru předmětu, povrchové výzdoby aplikované na předmět nebo kombinace konfigurace a povrchové výzdoby. Návrh povrchové výzdoby je neoddělitelný od předmětu, na který je aplikován, a nemůže existovat sám. Lze získat návrhový i užitný patent na jeden produkt v případě, že z technologického a designového hlediska jsou neoddělitelné. Návrhový patent má platnost (stejně jako u ostatních patentů) od 5 až do 25 let (každá země může mít jinak nastavenou platnost) [21, 36].

### **Struktura přihlášky návrhového patentu**

Struktura přihlášky návrhového patentu by měla obsahovat tyto prvky (v případě USPTO):

- Preambule s uvedením jména přihlašovatele, názvu návrhového patentu a stručného popisu použití předmětu, ve kterém je tento návrh ztělesněn.
- Křížový odkaz na související užitkovou žádost.
- Prohlášení týkající se federálně sponzorovaného výzkumu nebo vývoje.
- Popis obrázků a výkresů.
- Popis funkce.
- Jeden nárok na návrh.
- Výkresy a fotografie.

- Vykonaný slib nebo přísaha.

## Návrhový patent vs Ochranná známka

Ochranná známka je označení pro schopné grafické znázornění, tvořené zejména slovy, písmeny, číslicemi, barvou, kresbou nebo tvarem výrobku či jeho obalu, určené k rozlišení výrobků nebo služeb [28], zatímco návrhový patent se vztahuje na okrasný vzhled výrobku jako takového, který není spojen s identifikací zdroje zboží [33].

### 2.1.3 Patent rostlin

Patent na rostlinu je vydáván na novou nebo odlišnou odrůdu rostliny, která byla vynalezena nebo objevena a asexuálně rozmnožována. Tyto patenty se udělují na 20 let od data podání přihlášky a neplatí se žádné udržovací poplatky [23, 36]. Účelem těchto patentů je zamezení duplikování rostlin nebo jejich nabízení k prodeji (jako celek nebo po částech).

#### Struktura přihlášky patentu rostlin

Struktura patentu pro rostliny (v případě USPTO) je až jednu vyjímku stejná jako pro užitný patent (viz kapitola č. 2.1.1). Vyjímka se týká podrobného popisu, který musí obsahovat kompletní botanický popis dané rostliny a rozdíly, které tuto rostlinu odlišují od ostatních, již existujících rostlin.

## 2.2 Obory

Každý patent je po podání přihlášky klasifikován do jednoho oboru podle klasifikačního systému IPC. **International Patent Classification** (IPC) je hierarchický systém, který používá jazykově nezávislé symboly pro klasifikaci patentů a užitných vzorů podle oblasti technologie, ke které se vztahují [9]. Tento systém je používán ve více než 100 zemích.

IPC schéma definuje celkem osm hlavních oblastí technologie, do kterých lze patent přiřadit. Každá oblast je symbolizována jedním písmenem (viz tabulka č. 2.1).

Kód	Popisek
A	Lidské potřeby - jídlo, léky, oblečení, ...
B	Operace a Doprava - tisk, auta, koleje, nanotechnologie, ...
C	Chemie a Hutnictví - sklo, cement, železo, ...
D	Textílie a Papír - výroba papíru, provazy, tkalcovství
E	Pevné konstrukce - stavby, dveře, zámky, dolování, ...
F	Strojírenství, Osvětlení, Vytápění, Zbraně, Odstřelování
G	Fyzika - měření, testování, optika, nukleární fyzika, ...
H	Elektřina - základní elektrické elementy (kabely, rezistory, ...), techniky elektrické komunikace, ...

Tabulka 2.1: Základní IPC klasifikace oborů [12].

Klasifikační symboly definované IPC jsou tvořeny písmenem označující hlavní oblast technologie (neboli sekce), následovaným dvěmi číslicemi označující třídu, poté písmenem označujícím podtřídu. Po podtřídě následuje proměnná čítající jednu až čtyři číslice, která označuje hlavní skupinu. Následuje dopředné lomítko a číslo čítající dvě až šest číslic označující podskupinu [9]. Příklad klasifikace lze vidět v tabulce č. 2.2.

Divize	Počet divizí	Symbol	Titulek
Sekce	8	G	Fyzika
Třída	120	G01	Měření; Testování
Podtřída	628	G01G	Vážení
Skupina	+ - 6 900	G01G 21	Podrobnosti o vážení aparátu
Podskupina	+ - 62 100	G01G 21/02	Podrobnosti o vážení ložisek s ostřím nože

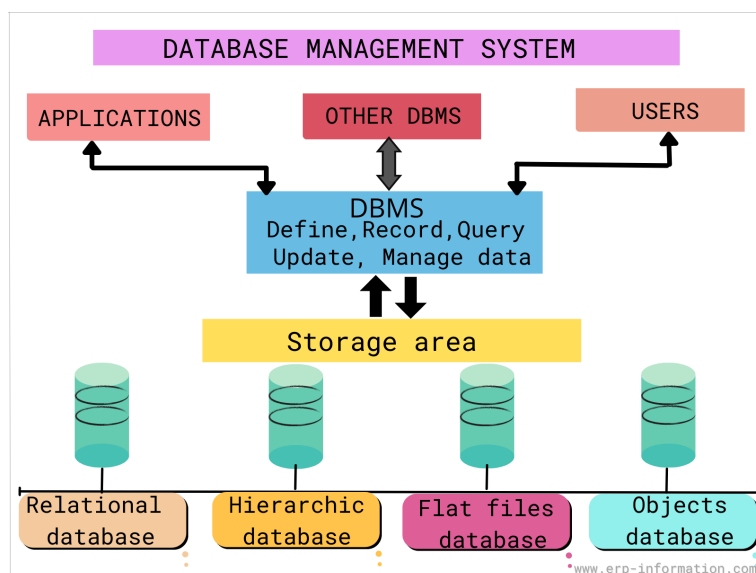
Tabulka 2.2: Příklad IPC klasifikace [31].

## 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 [27].

### 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 [10].

## 3.2 Komponenty databáze

Všechny databáze sestávají z pěti základních komponent, nehledě na použitý typ databáze [35]:

- **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 uživateli 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, percepce, 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

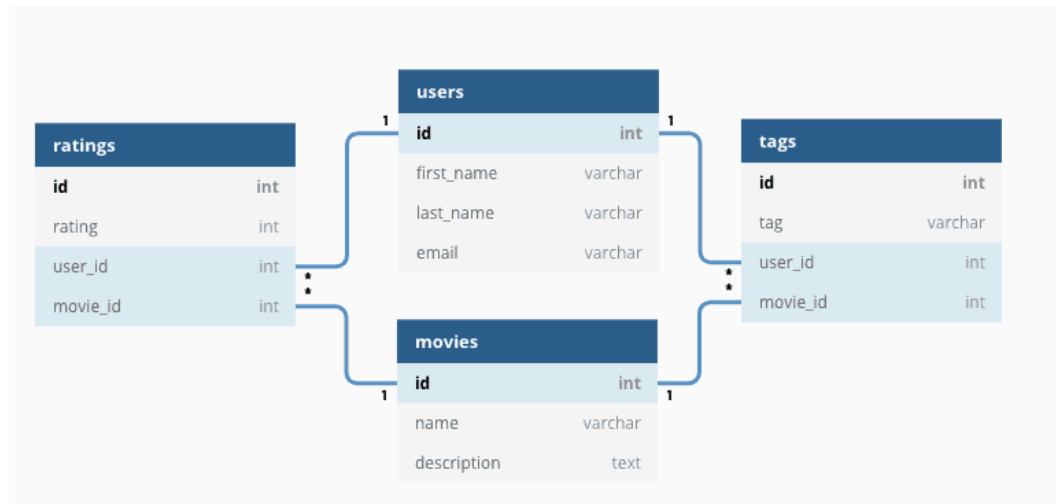
V dnešním světě existuje mnoho různých typů databází. Výběr nejlepšího typu databáze pro konkrétní organizaci závisí na tom, jak organizace zamýšlí data používat. V této kapitole je vypsáno pouze pár typů, protože vznikají stále nové, méně známé typy databází, které jsou tvořeny pro specifické požadavky (například finanční, vědecké) [1, 13].

### 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 [29]. Pro co nejefektivnější využití tohoto typu databáze je potřeba ukládat pouze dobře strukturovaná



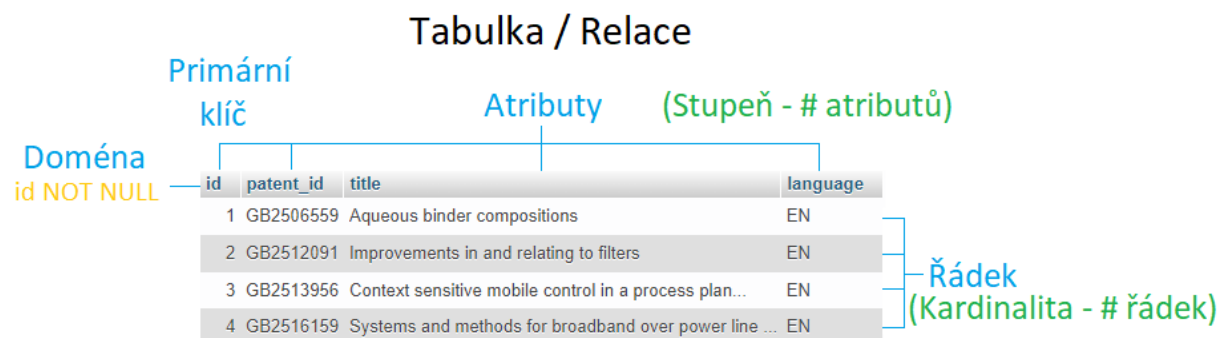
data, pro částečně strukturovaná či nestrukturovaná data je vhodné použít například grafové nebo dokumentově založené databáze. Typické relační databáze jsou například: Microsoft SQL Server, Oracle Database, MySQL. Ukázkou relační databáze lze vidět na obrázku č. 3.2.



Obrázek 3.2: Ukázkou 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 sloupce, 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 naši 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 uloženy 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 relacemi. Existuje šest typů relačních klíčů - kandidátní, super, složený, primární, cizí, sekundární / alternativní [30].

## Výhody

Výhody relační databáze jsou [25]:

- **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 mohou 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

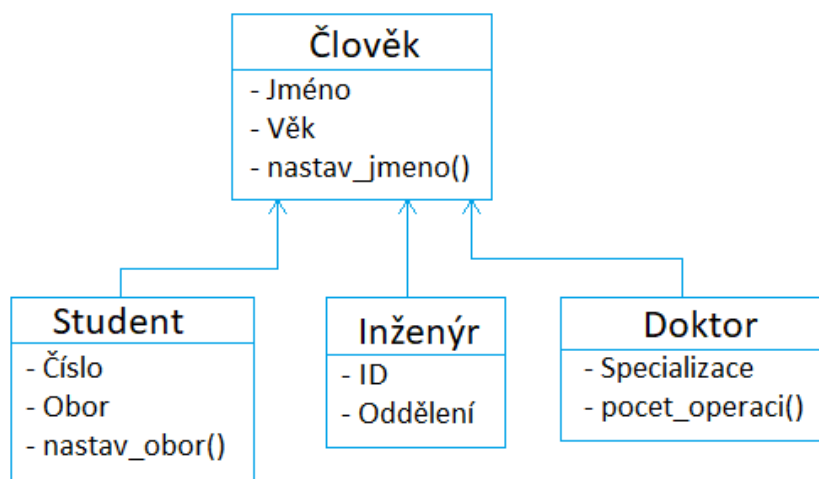
Nevýhody relační databáze jsou [25]:

- **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. Ukázkou objektově-orientované databáze lze vidět na obrázku č. 3.5.



Obrázek 3.5: Ukázkou objektově-orientované databáze.

### Datový model

V objektově-orientovaném modelu jsou data a jejich vztahy mezi sebou uloženy 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 [38]:

- **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 [38]:

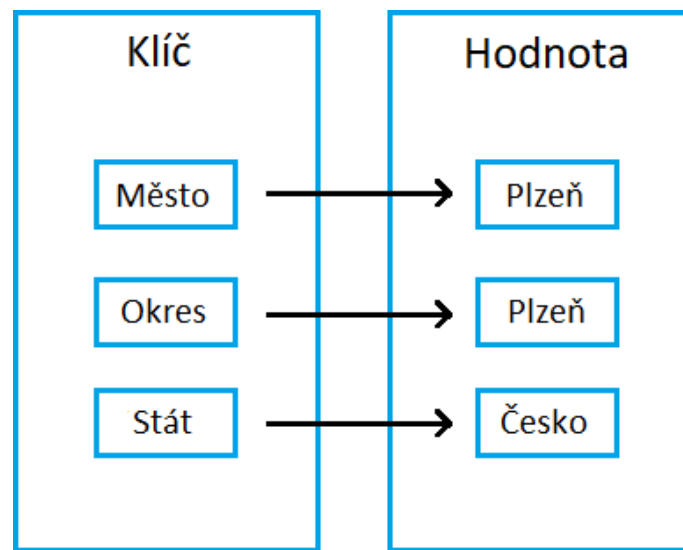
- **Neexistující univerzální datový model** - V dnešní době stále neexistuje univerzální datový model, navíc většině modelů chybí teoretický základ.
- **Nedostačující standardy** - Pro objektově-orientované databáze neexistují žádné univerzální datový model, stejně jako standardní dotazovací jazyk.
- **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 [39].

### 3.3.4 Databáze Klíč-Hodnota

Databáze klíč-hodnota poskytuje nejjednodušší možný NoSQL datový model. Data jsou uložena 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 [39]. Typické příklady jsou: Redis, Riak, LevelDB. Ukázkou 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 (vložit, smazat, získat).

### Nevýhody

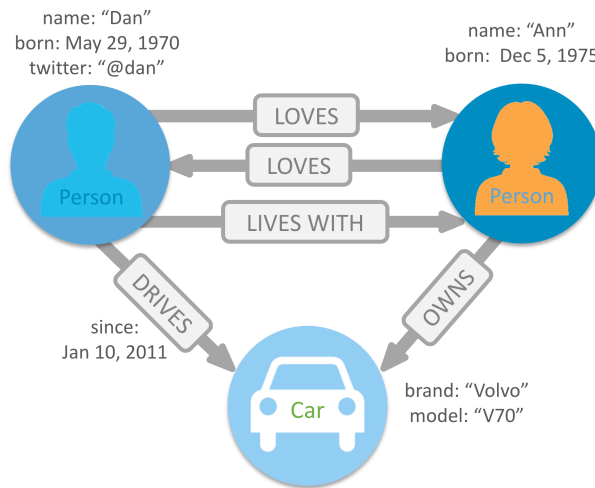
Nevýhody této databáze jsou [4]:

- **Optimalizace dat** - Optimalizace je provedena pouze pro data, kde je pouze jeden klíč a jedna hodnota. V případě ukládání složitějších struktur je potřeba parser.
- **Složitě 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

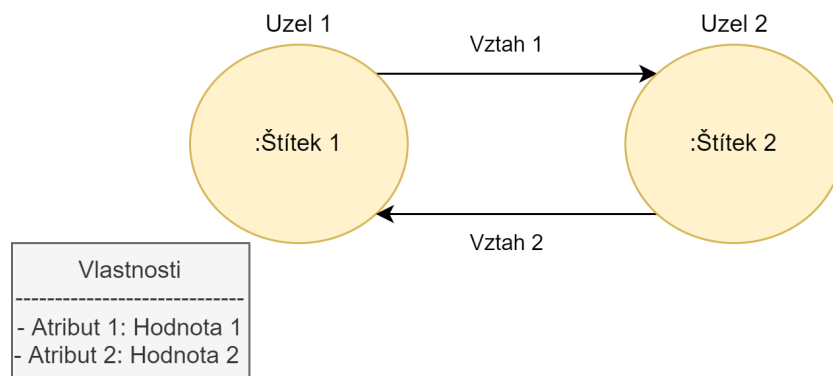
mezi daty. Graf lze procházet podél určitých typů hran nebo přes celý graf. Procházení spojení nebo relací je velmi rychlé, protože vztahy mezi uzly se nepočítají v době dotazu, ale jsou v databázi trvalé [39]. Typické příklady jsou: Neo4j, OrientDB, Microsoft Azure CosmosDB. Ukázku grafové databáze lze vidět na obrázku č. 3.7.



Obrázek 3.7: Ukázka grafové databáze [18].

## 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 označeny 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** - Vlastnost 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 navrženy pro ukládání, načítání a správu informací orientovaných na dokumenty. Typické příklady jsou: MongoDB, Amazon DocumentDB, Elasticsearch. Ukázkou dokumentové databáze lze vidět na obrázku č. 3.9.

Document 1	Document 2	Document 3
<pre>{   "id": "1",   "name": "John Smith",   "isActive": true,   "dob": "1964-30-08" }</pre>	<pre>{   "id": "2",   "fullName": "Sarah Jones",   "isActive": false,   "dob": "2002-02-18" }</pre>	<pre>{   "id": "3",   "fullName": {     "first": "Adam",     "last": "Stark"   },   "isActive": true,   "dob": "2015-04-19" }</pre>

Obrázek 3.9: Ukázka dokumentově orientované databáze [34].

#### 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 [39, 40].

#### 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.

- **Věstavené 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 kdokoli může používat a libovolně upravovat MySQL systém, aniž by musel cokoli platit. V případě změny zdrojových kódů je potřeba nastudovat podmínky užívání definované licencí GPL [17].

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 jinými aplikacemi, 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.
- **Věstavená knihovna** - Vícevláknová věstavená 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. PostgreSQL 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 [20].

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 inspirováni 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 [32].

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 nepřerušném 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.[14]

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 zkratk 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 [37].

## 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 základních podjazyků, kdy každý slouží k jinému účelu v rámci vykonávání příkazů [11, 26]. V některé literatuře se jazyk TCL také označuje jako podjazyk, společně s jazyky DDL, DML, DQL, DCL:

- **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:
  - **INSERT** - Vložení nového záznamu (dat) do tabulky.
  - **UPDATE** - Úprava existujícího záznamu v tabulce.
  - **DELETE** - Smazání záznamu z tabulky.
- **Data Query Language (DQL)** - Příkazy jazyka DQL slouží k provádění dotazů na data v rámci objektů schématu. Jejich účelem je získat vztah schématu na základě dotazu, který mu byl předán.
  - **SELECT** - Získání záznamů (dat) z tabulky.
- **Data Control Language (DCL)** - Pomocí DCL lze kontrolovat přístupy a práva k datům, které jsou uloženy v databázi. Uživatelé lze nastavit práva k jednotlivým DML příkazům nad tabulkami / procedurami (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 tabulkami, 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 triggeru [29]. 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 / mohou 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 použít 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, filtrovat a spočítat jejich četnost za pomoci agregační pipeline (zřetěženého zpracování). MQL podporuje transakce více dokumentů [15].

#### Syntax

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

**(a) Create**

```
db.users.insertOne(  ← collection
{
  name: "sue",        ← field: value
  age: 26,             ← field: value
  status: "pending"   ← field: value
}                    } document
)
```

**(b) Read**

```
db.users.find(
  { age: { $gt: 18 } }, ← collection
  { name: 1, address: 1 } ← query criteria
).limit(5)              ← projection
                        ← cursor modifier
```

**(c) Update**

```
db.users.updateMany( ← collection
  { age: { $lt: 18 } }, ← update filter
  { $set: { status: "reject" } } ← update action
)
```

**(d) Delete**

```
db.users.deleteMany( ← collection
  { status: "reject" } ← delete filter
)
```

Obrázek 3.13: Příklady CRUD operací v MongoDB [16].

### 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].

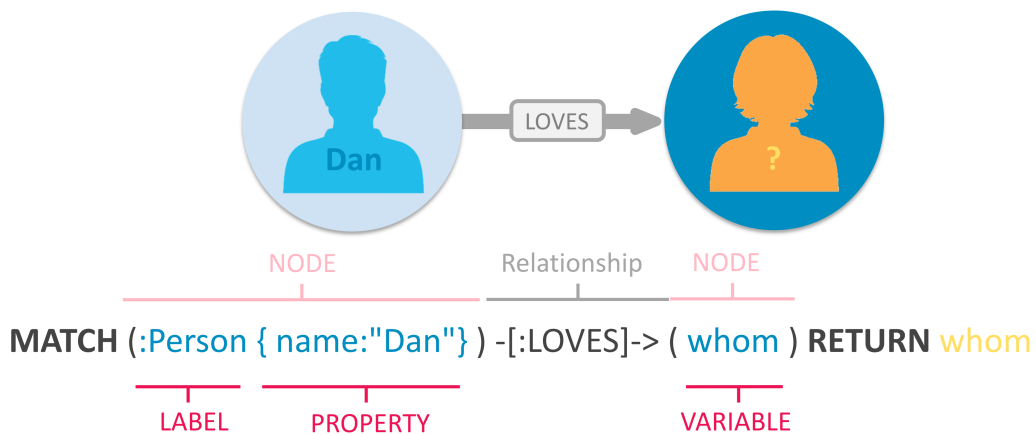
## Syntax

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 potřeba 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, zvláště při vyhledávání s použitím složitých dotazů (například hledání přes více tabulek).

V následujících kapitolách bude popsán postup výběru zdrojů patentů a patentových dat. Následně budou vybrány typy databáze, které budou vhodné pro uložení vybraných patentových dat a následné zvolení existujících řešení.

### 4.1 Výběr patentů

Při výběru patentů byly stanoveny čtyři 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).
- Alternativy jako Užitný a Průmyslový vzor (Návrhový patent) nebudou brány v potaz.

### 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 (příhlášky, 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.

Při průzkumu bylo zkoumáno celkem 51 národních zdrojů dat (patentových úřadů). V tabulkách č. 4.1 a 4.2 lze vidět název země, název patentového úřadu v dané zemi, zkratku patentového úřadu (pokud nějakou má) a jestli patentový úřad poskytoval data nebo ne. U každého patentového úřadu byl procházen její oficiální web a zkoumán na dostupnost patentových dat. Většina úřadů má na svých stránkách vyhledávač pro procházení vlastní databáze patentů, ale jen zlomek z nich poskytoval použitelná data zadarmo. Tyto data byla většinou schována pod neodpovídajícím názvem článku / příspěvku, a některé dokonce poskytovaly odkazy ke stažení dat na svých stránkách pouze v národním jazyce (neexistující článek s daty v anglické verzi webu).

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

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

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

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

Z celkových 51 patentových zdrojů nám pouze 19 zdrojů poskytuje data. V případě Itálie nám data neposkytuje přímo patentový úřad, ale výzkumný úřad PATIRIS, který poskytuje patentová data z univerzit a veřejných výzkumných ústavů v Itálii.

Bohužel ne všechny patentové úřady poskytují svá data zdarma. Celkem tři úřady - Austrálie, Německo, Nový Zéland chtěli za svá data zaplatit.

Ještě je potřeba zmínit Japonsko, které svá data poskytuje, ale je potřeba vyplnit formulář, ve kterém bylo potřeba naskenovat oficiální dokument potvrzující adresu školy. Z tohoto důvodu jsme bylo Japonsko jako zdroj dat zavrhnuto. Z původních 19 zdrojů dat poskytující data zůstalo nakonec jen 14 zdrojů dat, které poskytují svá data zadarmo.

### 4.1.2 Atributy

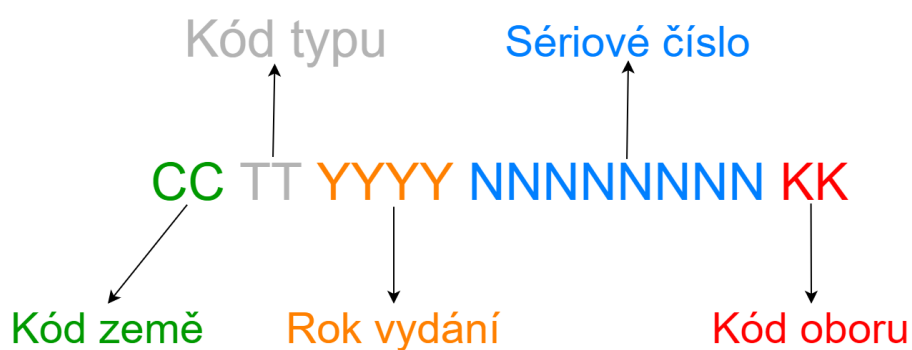
Při zadávání práce byly definovány podmínky pro výběr platných zdrojů dat, a jednou z nich bylo i povinnost mít důležité atributy ve struktuře dat patentu. Celkem byly definovány čtyři povinné atributy společně s osmi nepovinnými. Povinné a nepovinné atributy jsou podrobně popsány níže.

#### Povinné atributy

Se zadavatelem bylo domluveno, že validní zdroj patentových dat musí poskytovat patenty obsahující tyto atributy:

- **Titulek** - Titulek patentu, který říká o čem daný patent.
- **Rok přihlášky / publikace** - Patent musí obsahovat alespoň rok přihlášky / publikace. Publikace / přihláška musí být minimálně z roku 2000, patenty před rokem 2000 budou zamítnuty.
- **Autor** - U patentu bude nutné vědět jeho autor (jméno autora, případně název instituce).
- **ID patentu** - Patent musí mít nějaké kódové označení / identifikátor, podle kterého ho lze vyhledávat. Identifikátor se bude držet formátu viz obrázek č. 4.1, ale nemusí obsahovat všechny položky, protože každá země může některé položky zanedbávat, případně měnit počet znaků v položce, viz tabulka č. 4.3.





Obrázek 4.1: Základní formát pro identifikátor patentu [19].

<b>Země</b>	<b>CC</b>	<b>TT</b>	<b>YYYY</b>	<b>NNNNNNNN</b>	<b>KK</b>
Austrálie	AU		4 znaky	6 znaků	ANO
Kanada	CA			7 znaků	ANO
Čína	CN	1 znak		8 znaků	ANO
EPO	EP			7 znaků	ANO
Německo	DE	2 znaky	4 znaky	6 znaků	ANO
Francie	FR			7 znaků	ANO
Velká Británie	GB			7 znaků	ANO
Nizozemsko	NL			7 znaků	ANO
Japonsko	JP		4 znaky	6 znaků	ANO
Korea	KR	2 znaky	4 znaky	7 znaků	ANO
Rusko	RU		4 znaky	6 znaků	ANO
USA	US		4 znaky	7 znaků	ANO
WIPO	PCT		4 znaky	6 znaků	ANO

Tabulka 4.3: Aktuálně používané formáty pro patenty z různých zemí [19].

V tabulce č. 4.4 lze vidět patenty, poskytované národními patentovými úřady zdarma, obsahují povinné atributy.

Země	Titulek 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	-	-
Velká Británie	x	x	x	x
Rusko	x	x	x	x
Peru	x	x	x	x
Francie	x	x	x	x

Tabulka 4.4: Povinné atributy nacházející se v dostupných patentech.

### Nepovinné atributy

Při průzkumu byly zjišťovány i nepovinné atributy, které nemají vliv na výběr zdrojů dat, ale je dobré vědět co který patent z daného patentového zdroje poskytuje za atributy. Nepovinné atributy jsou:

- **Abstrakt** - Stručný výtah patentu, který popisuje o čem daný patent je.
- **Klíčová slova** - Klíčová slova nebo fráze spojené s patentem. Mohou sloužit při vyhledávání patentů se stejným zaměřením.
- **Reference** - Reference na podobné typy patentů nebo na související patenty (například odkaz na základní verzi algoritmu).
- **Žadatel** - Žadatel a autor může být tatáž osoba, ale v některých případech je žadatelem někdo jiný (například autor je zaměstnanec firmy, žadatelem je samotná firma).
- **Adresa autora / instituce** - Adresa autora nebo instituce.
- **Rodina patentů** - Rodina patentů je kolekce patentových žádostí, které se zaměřují na stejný nebo alespoň podobný technický obsah.

- **Obor** - Obor, který daný patent pokrývá.
- **Full-text** - Zdroje dat poskytují veškerá data o patentu (nejenom to co je v přihláškách / publikacích, například různé poznámky, obrázky).

V tabulkách č. 4.5, č. 4.6 lze vidět patenty, poskytované národními patentovými úřady zdarma, obsahující nepovinné atributy.

Země	Abstrakt	Klíčová slova	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	-	-
Velká Británie	-	-	-	-
Rusko	-	-	-	-
Peru	-	-	-	-
Francie	x	-	x	x

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

<b>Země</b>	<b>Adresa</b>	<b>Rodina patentů</b>	<b>Obor</b>	<b>Full-text</b>
Kanada	x	x	x	-
Česko	-	x	x	-
Litva	-	x	x	-
Portugalsko	-	-	x	-
Španělsko	x	-	x	x
Švédsko	-	x	x	x
Izrael	x	x	-	-
Itálie	-	-	-	-
Mexiko	-	-	x	-
Polsko	-	-	-	-
Velká Británie	-	-	x	-
Rusko	-	-	-	-
Peru	-	-	x	-
Francie	-	x	x	-

Tabulka 4.6: Nepovinné atributy nacházející se v dostupných patentech, část druhá.

### 4.1.3 Analýza dat

Z tabulky č. 4.4 lze vidět tři země, jejichž poskytovaná data neobsahují povinné atributy - Česko, Švédsko a Polsko. Pro země, jejichž data obsahují povinné atributy, bude následně provedena hlubší analýza dat - formát dat, počet souborů s daty, struktura dat a počet vyfiltrovaných patentů.

#### Velká Británie

Velká Británie poskytuje svá data v žurnálu ve formátu XML nebo PDF v týdenních intervalech od roku 2008. Součástí žurnálu jsou záznamy o přihláškách patentů, jejich publikacích, udělených patentech, evropských patentech a patentech, kterým skončila platnost.

Celkem bylo staženo 715 žurnálů ve formě XML souborů, ze kterých byly použity pouze udělené patenty (granted). Struktura žurnálů zůstává neměnná od roku 2008 až do současnosti. Při filtrování byl nalezen pouze jeden záznam o patentu, který neobsahoval všechny povinné atributy, z celkového počtu 88 032 patentů.

## **Francie**

Francie poskytuje svá data pouze uživatelům, kteří provedli registraci na stránkách francouzské patentové instituce a vyplnili formulář k poskytnutí dat. Data jsou dostupná jak z webové stránky patentové instituce, tak i na FTP, ze které si data lze stáhnout. Dostupných typů dat o patentech bylo několik, ale nejdůležitější byly tyto - bibliografická data o patentech a evropské patenty.

Celkem bylo staženo 2 729 111 souborů s patenty ve formátu XML, ze kterých bylo 746 899 unikátních s datem přihlášení od roku 2000. Struktura patentů byla rozdílná jak v rámci typů dat (bibliografické a evropské), tak i v rámci bibliografických dat, kdy struktura patentů v roce 2017 byla jiná než v roce 2021. Bibliografická data obsahovala mnoho duplikátů a patentů, kterým chyběl vždy alespoň jeden povinný atribut (většinou titulek). Většina evropských patentů byla také odfiltrována, stejně jako u bibliografických zde chyběl hlavně titulek. Celkem bylo vyfiltrováno 474 850 patentů.

## **Itálie**

Data pro Itálii poskytuje pouze výzkumný úřad PATIRIS, která jsou poskytována jako plný export relační databáze s tabulkama ve formátu SQL. Databáze obsahuje celkem 9 tabulek a 21 367 patentů, ze kterých pouze 7 624 obsahuje všechny povinné atributy.

## **Izrael**

Izrael poskytuje svá data všem uživatelům v žurnálu ve formátu XML v ročních intervalech od roku 2000. Celkem bylo staženo 19 žurnálů, kdy všechny mají stejnou strukturu dat. Žurnál obsahuje údaje jak o publikaci patentu, tak i o jeho žádosti, ale analýza probíhala pouze nad publikacema. Celkem bylo vyfiltrováno 326 patentů kvůli chybějícímu autoru a titulku z celkových 116 380 patentů.

## **Kanada**

Kanada poskytuje svá data všem uživatelům ve formátu XML v týdenních intervalech od roku 2019. Kanadský patentový úřad poskytuje dva typy patentových dat - bibliografické a full-text.

Celkem bylo staženo 1 566 425 souborů s patenty, ze kterých bylo 936 464 unikátních s datem přihlášení od roku 2000. Struktura dat se v průběhu let jednou změnila, konkrétně v roce 2022. V bibliografických datech byly

údaje pouze pro publikace, zatímco pro full-text se zde nacházel celý popis patentu i jeho abstrakt. Celkem bylo vyfiltrováno 119 696 patentů kvůli chybějícímu titulku a autorům.

### **Litva**

Litva poskytuje svá data všem uživatelům ve formátu XML v dvoutýdenních intervalech od konce roku 2021. Patentový úřad v Litvě poskytuje národní i evropské patenty. Celkem bylo staženo 869 souborů s patenty, kde žádnému z nich nechyběl žádný povinný atribut.

### **Peru**

Peru poskytuje svá data všem uživatelům ve formátu XLSX. Pouze jeden soubor s daty z roku 2019 byl nalezen na webové stránce patentové instituce v Peru. Soubor obsahuje celkem 23 157 patentů, ze kterých pouze 1 805 je validních, ostatní patenty mají přihlášku před rokem 2000.

### **Portugalsko**

Patentová data z Portugalska jsou poskytována na webovém portálu *data.europa.eu*, který spravuje Evropská Unie. Na portále existuje pouze jeden soubor ve formátu XML, který obsahuje celkem 69 patentů.

### **Rusko**

Rusko poskytuje svá data všem uživatelům ve formátu CSV v měsíčních intervalech (od roku 2020) a nepravidelných intervalech (od roku 2017 do roku 2019).

Celkem bylo staženo 30 souborů, u kterých zůstala struktura dat neměnná. Ve všech souborech existuje celkem 15 920 786 záznamů o patentech, ze kterých pouze 753 970 patentů je unikátních. Celkem bylo vyfiltrováno 139 937 patentů, kde většina z nich měla datum registrace před rokem 2000.

### **Španělsko**

Španělsko poskytuje svá data všem uživatelům ve formátu XML v denních a měsíčních intervalech od roku 2021, pro data před rokem 2021 existují roční reporty až do roku 1987. Data jsou poskytována ve formě full-textu.

Celkem bylo staženo 458 566 souborů s patenty, u kterých zůstala struktura dat neměnná po všechny roky (od roku 2004). Z dat bylo odfiltrováno celkem 345 676 patentů, hlavně kvůli chybějícímu autoru.

## Mexiko

Mexiko poskytuje svá data všem uživatelům ve formátu XML v měsíčních intervalech, ale ke stažení jsou pouze data z předcházejícího měsíce. Struktura souborů se za poslední měsíce nezměnila, ale data o patentech obsažená v souborech jsou nevyhovující, protože se zde nachází patentové přihlášky, které jsou zamítnuté, zrušené z důvodu nezaplacení nebo jim vypršela platnost.

### 4.1.4 Závěr průzkumu

Průzkum národních zdrojů zahrnoval celkem 51 národních patentujících institucí, ze kterých pouze 10 poskytovalo svá data zdarma a splňovala všechny podmínky. V tabulce č. 4.7 lze vidět souhrn výsledků.

Popis	Počet	Poměr
Nedostupné	33	64,70 %
Nepoužitelné	4	7,85 %
Za peníze	4	7,85 %
Použitelné	10	19,60 %

Tabulka 4.7: Souhrn průzkumu národních patentujících institucí.

Pro všechny použitelné národní zdroje bylo kromě výskytu atributů dále sledováno: formát uložených dat, počet patentů po roce 2000 (včetně roku 2000) obsahující všechny povinné atributy, počet patentů před rokem 2000 a počet duplikátů. Duplikátem se myslí jiná verze daného patentu, protože v průběhu let se mohl měnit obsah patentu a v databázi nechceme ukládat žádné starší verze jednoho patentu (výsledky vyhledávání v databázi nebudou validní). V tabulce č. 4.8 lze vidět všechny validní národní zdroje.

<b>Země</b>	<b>Formát dat</b>	<b>Počet patentů (rok &gt;= 2000)</b>	<b>Počet patentů (rok &lt; 2000)</b>	<b>Počet duplikátů</b>
Velká Británie	XML	88 032	141	19
Francie	XML	746 899	192 630	1 140 084
Israel	XML	116 380	0	9 956
Itálie	SQL	7 624	4 150	9 593
Kanada	XML	936 464	130 279	499 682
Litva	XML	869	0	0
Peru	XLSX	1 805	21 352	0
Portugalsko	XML	69	0	0
Rusko	CSV	614 256	139 714	15 166 816
Španělsko	XML	381 713	37 612	39 241
<b>Souhrn</b>		<b>2 894 111</b>	<b>525 878</b>	<b>16 865 391</b>

Tabulka 4.8: Seznam všech validních národních zdrojů.

V tabulce č. 4.9 lze vidět výsledný počet patentů (po filtraci všech patentů neobsahující povinné atributy - ID, titulek, autor, datum).

<b>Země</b>	<b>Počet patentů před filtrací</b>	<b>Filtrace</b>	<b>Počet patentů po filtraci</b>
Velká Británie	88 032	1	88 031
Francie	746 899	474 850	272 049
Israel	116 380	326	116 054
Itálie	7 624	530	7 094
Kanada	936 464	119 696	816 768
Litva	869	0	869
Peru	1 805	0	1 805
Portugalsko	69	0	69
Rusko	614 256	223	614 033
Španělsko	381 713	308 064	73 649
<b>Souhrn</b>	<b>2 894 111</b>	<b>903 690</b>	<b>1 990 421</b>

Tabulka 4.9: Výsledný počet patentů po filtraci.



## 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 ukládat 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 vytvořením 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 databáze spočívá v jednoduchém mapování objektů při práci s objektově-orientovaným programováním,

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 uložení 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

Výsledné řešení by mělo obsahovat dvě databáze, pomocí kterých bude možno vyhledávat patenty, ať už podle určitých atributů (ID, země, obor, a jiné), nebo pomocí full-textu. Vybrané databáze jsou uzpůsobeny pro ukládání velkého množství patentových dat, které byly staženy z národních zdrojů.

Import dat bude řešen pomocí vlastní aplikace, která bude filtrovat nevalidní patenty (neexistující povinné atributy, nevalidní XML struktura, ...) a importovat validní patenty do databází. Pro MySQL databázi bude potřeba z patentu získat pouze specifické atributy.

Výsledné řešení bude potřeba jednoduše nasadit na produkční server / počítač, aniž by bylo potřeba instalovat vícero aplikací. Řešení se postará o automatickou instalaci obou databází, jejich inicializaci (vytvoření tabulek, kolekcí, pohledů, ...) a zajistí propojení mezi databázemi **MongoDB** a full-textovým vyhledávačem **Elasticsearch**. S databázemi se zároveň nainstalují i aplikace pro jejich správu.

### 5.1 Implementace databáze

Pro databázi **MySQL** bylo navrženo a vytvořeno databázové schéma tak, aby bylo možno použít co nejvíce atributů při filtrování patentů pro statistiky. V **MongoDB** bude vytvořena databáze s jednou kolekcí, která bude obsahovat všechny vyfiltrované patenty.

#### 5.1.1 MongoDB

Pro MongoDB byla použita jedna z nejnovějších verzí komunitní edice (verze 5.0.6). Programový systém **Mongo-express** (verze 0.54.0) bude použit jako nástroj pro spravování MongoDB databáze, ke kterému lze přistoupit pomocí webového prohlížeče.

V databázovém schématu byla vytvořena databáze s názvem **patents**, která obsahuje pouze jednu kolekci s názvem **patents**. V této kolekci budou uloženy všechny vyfiltrované patenty ze všech zemí. Vzhledem k tomu, že MongoDB bude sloužit jen jako úložiště patentových dat, tak není potřeba vytvářet více kolekcí nebo databází. Jako další důvod pro zvolení pouze jedné kolekce je počet indexů v Elasticsearch, kdy byl použit pouze jeden index pro zaindexování všech dat. Důvod je takový, že chceme vždy prohle-

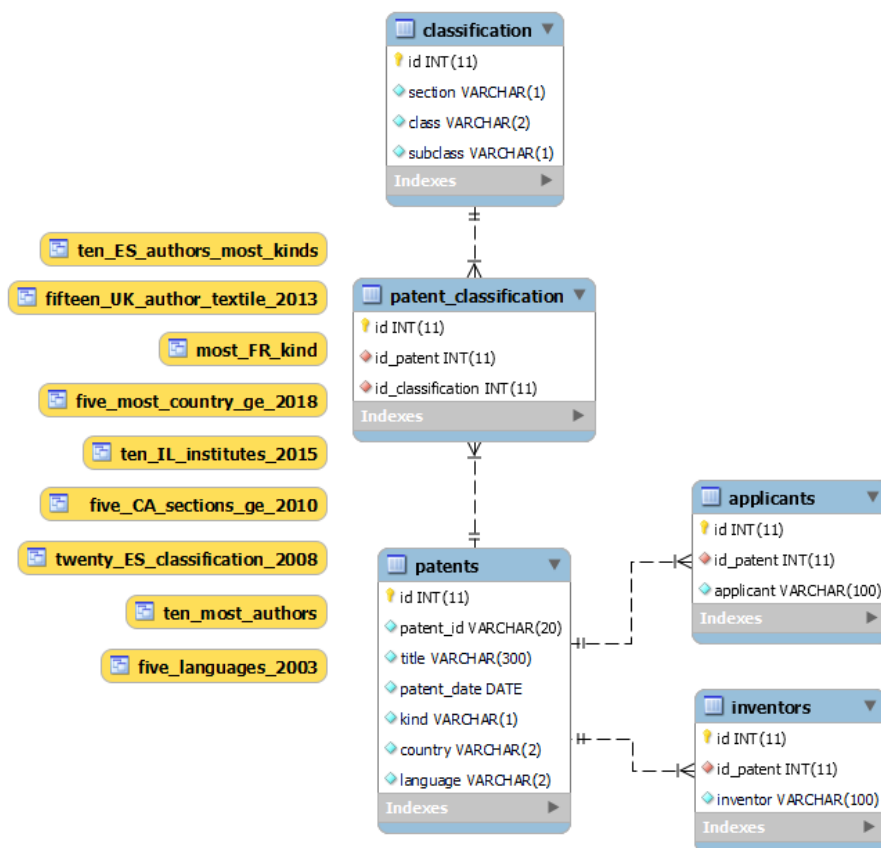
dávat všechny patenty a ne jen jejich část, takže vícero indexů by akorát způsobovalo pomalejší zpracování dotazů pro vyhledávání.

### 5.1.2 MySQL

Pro MySQL byla použita verze komunitního serveru (verze 8.0.16). Programový systém **phpMyAdmin** (verze 5.1.3) bude použit jako nástroj pro spravování MySQL databáze, ke kterému lze přistoupit pomocí webového prohlížeče. Lze použít i nástroj **MySQL Workbench** pro správu MySQL databáze, ale pro naše účely bohatě postačí **phpMyAdmin**.

Schéma splňuje pouze druhou normální formu. Je to z toho důvodu, že tabulka **classification** obsahuje některé duplicitní hodnoty ve sloupcích. Duplicitu samozřejmě lze odstranit, ale zvýšilo by to složitost celého systému - SQL dotazy by byly složitější a pomalejší v případě příkazu *SELECT*, který by vyhledával data z několika tabulek, spojených příkazem *JOIN*.

Databázové schéma pro MySQL lze vidět na obrázku č. 5.1.



Obrázek 5.1: Schéma pro MySQL databázi.

### Tabulka **patents**

Tabulka **patents** uchovává všechny národní patenty, které byly poskytnuty zdarma a obsahují všechny povinné atributy.

### Tabulka **classification**

Tabulka **classification** slouží k uchovávání IPC klasifikace daného patentu, konkrétně jeho sekci, třídu a podtřídu (skupina a podskupina vybrána nebyla, protože se vyskytovala v patentech jen ojediněle).

### Tabulka **patent\_classification**

Tabulka **patent\_classification** má kardinalitu  $M:N$  a slouží jako propojení tabulky **patents** a tabulky **classification**. Jeden patent může mít více klasifikací (například se mohlo změnit označení v průběhu let, nebo patent pokrývá více oborů) a jedna klasifikace může být u více patentů.

### Tabulka **inventors**

Tabulka **inventors** slouží k uchovávání jmen autorů patentů. Kardinalita mezi tabulkou **patents** a tabulkou **inventors** je  $1:N$ , protože pro jeden patent může existovat více autorů.

### Tabulka **applicants**

Tabulka **applicants** slouží k uchovávání jmen žadatelů patentů. Ačkoli existuje tabulka pro autory, tak může existovat scénář, ve kterém bude potřeba vyhledávat žadatele pro daný patent. Kardinalita mezi tabulkou **patents** a tabulkou **applicants** je  $1:N$ , protože pro jeden patent může existovat více žadatelů.

### Pohledy

Pohled je databázový objekt, který uživateli poskytuje data ve stejné podobě jako tabulka. Stručněji řečeno, je to struktura uchovávající SQL dotaz, který se většinou dotazuje dané tabulky na specifická data.

V databázi pro patenty bylo vytvořeno celkem devět pohledů, kdy každý z nich reprezentuje jeden scénář, který je použit při ověřování efektivního vytěžování (viz kapitola č. 7.2).

## 5.2 Nasazení úložiště

Celý modul našeho úložiště čítá celkem 5 nástrojů - MySQL, MongoDB, Elasticsearch, phpMyAdmin a Mongo-express. Zároveň bude potřeba nastavit připojení mezi MongoDB a Elasticsearch tak, aby při přidání nového patentu do databáze byl následně zaindexován ve full-textovém vyhledávači. Pokud bychom měli po každém uživateli chtít instalaci všech těchto nástrojů a k tomu stahovat další nástroje pro vytvoření připojení mezi MongoDB a Elasticsearch, tak to zabere mnoho času a existuje zde velká pravděpodobnost že některé nástroje nebudou kompatibilní s aktuální verzí operačního systému. Z těchto důvodů byl zvolen software **Docker**, pomocí kterého se provede veškerá instalace a nastavení automaticky.

### 5.2.1 Docker

**Docker** je jeden z nejznámějších open-source nástrojů pro dodání aplikací v balíčkách zvaných *kontejner*. **Docker** využívá virtualizaci na úrovni operačního systému, čímž je výrazně snížena režie na rozdíl od klasických virtuálních strojů. Existují i jiné alternativy než docker, ale díky velké popularitě a fanouškovské základně byl vybrán právě docker.

Definice a instalace aplikací je zajištěna pomocí nástroje **Docker compose**. Definice probíhá pomocí souboru s názvem *docker-compose.yml* (viz obrázek č. 5.2), který využívá YAML formát pro serializaci strukturovaných dat. V *docker-compose.yml* souboru pro tento modul je definováno celkem devět aplikací a dva nastavovací moduly. Data nejsou součástí inicializace kontejnerů, je potřeba je importovat dodatečně (dále viz kapitola **Uživatelská dokumentace**).

```

1  version: '3.7'
2
3  services:
4    elasticsearch:
24   elasticvue:
31   mongo-setup:
44   mongo:
55   mongo-express:
66   zookeeper:
77   broker:
102  connect:
141  connect-setup:
156  mysql:
157    image: mysql:8.0.16
158    container_name: mysql
159    restart: always
160    ports:
161      - 6033:3306
162    command: --init-file /mysql_init.sql
163    volumes:
164      - ./init/mysql/mysql_init.sql:/mysql_init.sql
165    environment:
166      - MYSQL_ROOT_PASSWORD=password
167
168  phpmyadmin:
169    image: phpmyadmin:5.1.3
170    container_name: phpmyadmin
171    restart: always
172    ports:
173      - 8082:80
174    links:
175      - mysql
176    environment:
177      PMA_HOST: mysql
178      PMA_PORT: 3306
179      PMA_ARBITRARY: 1
180      UPLOAD_LIMIT: 600M

```

Obrázek 5.2: Ukázka souboru docker-compose.yml.

## Docker - Elasticsearch

Pro Elasticsearch byly nadefinovány dvě aplikace:

- **elasticsearch** - Oficiální image full-textového vyhledávače Elasticsearch ve verzi 8.2.0. V nastavení byla vypnuto zabezpečení kvůli problémům se sadou replik v MongoDB.
- **elasticvue** - Grafické uživatelské rozhraní pro Elasticsearch ve verzi 0.39.0, dostupné na localhostu na portu 8080.

## Docker - MongoDB

Pro MongoDB byly nadefinovány dvě aplikace a jeden nastavovací modul:



- **mongo** - Oficiální image databáze MongoDB ve verzi 5.0.6. V databázi bylo nutné nastavit sadu replik, aby bylo možné vytvořit propojení mezi MongoDB a Elasticsearch.
- **mongo-express** - Grafické uživatelské rozhraní pro MongoDB ve verzi 0.54.0, dostupné na localhostu na portu 8081.
- **mongo-setup** - Nastavovací modul pro MongoDB, který po inicializaci databáze inicializuje sadu replik a následně vytvoří novou databázi a kolekci pro patenty.

## Docker - MongoDB a Elasticsearch konektor

Pro konektor mezi MongoDB a Elasticsearch byly nadefinovány tři aplikace a jeden nastavovací modul:

- **broker** - Komunitní verze Apache Kafka od firmy Confluent ve verzi 6.1.0.
- **zookeeper** - Nástroj pro Apache Kafka ve verzi 6.1.0, který funguje jako centralizovaná služba a slouží k údržbě jmenných a konfiguračních dat. Zároveň zajišťuje i flexibilní a robustní synchronizaci v rámci distribuovaných systémů.
- **connect** - Nástroj pro Apache Kafka ve verzi 6.1.0, který má za úkol propojit externí systémy s Apache Kafka za pomoci konektorů.
- **connect-setup** - Nastavovací modul pro Kafka connect, který po inicializaci Kafka connect vytvoří dva konektory:
  - *mongo-source-connector* - Konektor, který aktivně získává nová data z MongoDB a vkládá je do Kafka brokeru.
  - *elasticsearch-sink-connector* - Konektor, který aktivně získává nová data z Kafka brokeru a vkládá je do Elasticsearch.

## Docker - MySQL

Pro MySQL byly nadefinovány dvě aplikace:

- **mysql** - Oficiální image databáze MySQL ve verzi 8.0.16. Při inicializaci databáze se vytvoří všechny tabulky i pohledy.
- **phpmyadmin** - Grafické uživatelské rozhraní pro MySQL ve verzi 5.1.3, dostupné na localhostu na portu 8082. V nastavení byl navýšen limit pro import souborů na 600 MB z původních 2 MB.

### 5.2.2 Inicializace MySQL

Inicializace MySQL probíhá po vytvoření databáze v rámci docker kontejneru pro MySQL. Pro inicializaci byl vytvořen soubor **mysql\_init.sql**, který obsahuje příkazy pro vytvoření všech pěti tabulek (názvy tabulek, názvy sloupců, jejich omezení a klíče) a všech devíti pohledů.

### 5.2.3 Inicializace MongoDB

Inicializace MongoDB probíhá po vytvoření databáze v rámci docker kontejneru **mongo-setup**, protože MongoDB nemá žádnou možnost jak vložit inicializační soubor při startu kontejneru. Pro inicializaci byly vytvořeny dva soubory:

- **create\_\_mongo\_\_database.js** - Javascriptový soubor, který obsahuje příkazy pro vytvoření kolekce v databázi pro patenty.
- **mongo\_init.sh** - Shell skript, který inicializuje sadu replik v databázi a následně spustí javascriptový soubor **create\_\_mongo\_\_database.js**.

### 5.2.4 Propojení MongoDB a Elasticsearch

Pro propojení MongoDB a Elasticsearch byly vyzkoušeny dva nástroje: **Mongo-connector** a **Apache Kafka**.

#### Mongo-connector

Mongo-connector je obecný připojovací systém vyvinutý firmou MongoDB Inc. v roce 2012, který slouží pro integraci databáze MongoDB s jiným systémem, který podporuje CRUD operace. Mongo-connector byl následně udržován pouze komunitně přibližně do roku 2018, kdy byla vydána poslední verze tohoto nástroje (verze 3.1.1).

V původní verzi úložiště byl mongo-connector použit k propojení MongoDB ve verzi 5.0.6 a Elasticsearch ve verzi 7.17.0. Konektor pracoval rychle a bez problémů do té doby, než bylo potřeba použít novější verzi Elasticsearch (verze 8.x.x), kdy konektor nedokázal navázat žádné spojení a tím pádem nebylo možné využívat vyhledávač.

Přechod na novější verzi Elasticsearch bylo potřeba z důvodu různých struktur patentů. Národní patentové úřady poskytují svá patentová data v různých strukturách, které se mohou měnit v průběhu let, což byl i jeden z důvodů pro výběr databáze MongoDB. V případě Elasticsearch je ale nutné, aby data měla stále stejnou strukturu, alespoň v rámci MongoDB kolekcí,

protože Elasticsearch si data z kolekcí musí nejdříve zaindexovat pomocí mapperu a pak až v nich lze vyhledávat. V nejnovější verzi Elasticsearche (verze 8.x.x) je použit jiný způsob indexování souborů, takže data mohou mít libovolné struktury. Z tohoto důvodu byl mongo-connector zamítnut a musel být použit nový nástroj Apache Kafka.

## Apache Kafka

Apache Kafka je distribuované úložiště událostí a platforma pro zpracování datových proudů (streamů). V diplomové práci byla použita Apache Kafka od firmy Confluent, která obsahuje dodatečné komunitní a komerční funkce navržené tak, aby vylepšily streamování operátorů i vývojářů v produkčním prostředí.

Inicializace konektorů pro Apache Kafka probíhá v nastavovacím modulu **connect-setup**, ve kterém je stažen a nainstalován program **CURL**, který je využit shellovým skriptem **connect.sh**. Tento skript posílá dvě HTTP POST metody na brokera (Apache Kafka), pomocí kterých se vytváří nové konektory. Na obrázku č. 5.3 lze vidět data ve formátu JSON pro POST metodu, která vytvoří spojení z Apache Kafka do Elasticsearch, a na obrázku č. 5.4 lze vidět data pro vytvoření spojení z MongoDB do Apache Kafka.

```
{
  "name": "elasticsearch-sink-connector",
  "config": {
    "connector.class": "io.confluent.connect.elasticsearch.ElasticsearchSinkConnector",
    "key.converter": "org.apache.kafka.connect.storage.StringConverter",
    "value.converter": "org.apache.kafka.connect.json.JsonConverter",
    "connection.url": "http://elasticsearch:9200",
    "schema.ignore": "true",
    "value.converter.schemas.enable": "false",
    "producer.buffer.memory": "200",
    "offset.flush.interval": "500",
    "flush.timeout.ms": "1000",
    "flush.synchronously": "true",
    "batch.size": "20",
    "topics": "patents.patents"
  }
}
```

Obrázek 5.3: Data pro vytvoření konektoru z Apache Kafka do Elasticsearch.

```
{
  "name": "mongo-source-connector",
  "config": {
    "tasks.max": "1",
    "connector.class": "com.mongodb.kafka.connect.MongoSourceConnector",
    "connection.uri": "mongodb://mongo:27017",
    "key.converter": "org.apache.kafka.connect.json.JsonConverter",
    "value.converter": "org.apache.kafka.connect.json.JsonConverter",
    "database": "patents",
    "copy.existing" : "true",
    "poll.max.batch.size": "20",
    "poll.await.time.ms": "2000",
    "poll.batch.size": "20",
    "errors.log.enable": "true",
    "errors.log.include.message": "true",
    "collection": "patents"
  }
}
```

---

Obrázek 5.4: Data pro vytvoření konektoru z MongoDB do Apache Kafka.

## 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 tři možné návrhy na rozšíření modulu ohledně importu dat do již existujících databází.

### 6.1 Přidávání nových patentů

Cílem tohoto rozšíření 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 (například Java, C), kdy vstupem do aplikace by byl soubor v datovém formátu JSON/XML/CSV a jiné. Vstupní soubor by se následně:

- Převodl 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 kapitola č. 5.1.2).

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 obsahovaly 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 prohlížečů 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 příslušného 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  
(například `/html/body//a[contains(@href,'example')]/@href`)
- **Poslední verze** - Název / číslo poslední stažené verze.

XML soubor by byl následně zpracován pomocí aplikace (například 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 - například 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 (například každé druhé úterý v 17:00). Jako příklad lze uvést použití pipeline na Jenkins serveru, který bude spouštět z lokálního uložště 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 s využitím vyhledávače Elasticsearch. Z výsledků scénářů můžeme poté usoudit, jak moc efektivní vytěžování je vzhledem k technickým parametrům stroje a použitým technologiím.

### Testovací stroj - technické parametry

Testovací stroj, který byl použit pro kontrolu efektivity vytěžování, má tyto technické parametry:

- **Procesor** - Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz (8 CPUs).
- **RAM** - 8192 MB
- **Disk** - SSD 256 GB
- **Grafická karta 1** - Intel(R) UHD Graphics 620
- **Grafická karta 2** - NVIDIA GeForce 940MX
- **Operační systém** - Windows 10 Home 64bit (10.0, build 19044).

## 7.1 Mongo + ElasticSearch

todo

### 7.1.1 Scénář č.1

todo

Textový popis:

Dotaz:

Rychlost vykonání dotazu:  $\pm X$  sekund

Výsledek dotazu:



count(*)	percentage	section
7043	0.93196	D
41788	5.52955	E
60052	7.94631	F
71359	9.44249	H
101316	13.40652	G

Obrázek 7.1: Ukázka výsledku dotazu pro scénář č.2

### 7.1.2 Scénář č.2

todo

Textový popis:  
Dotaz:  
Rychlost vykonání dotazu:  $\pm X$  sekund  
Výsledek dotazu:

count(*)	percentage	section
7043	0.93196	D
41788	5.52955	E
60052	7.94631	F
71359	9.44249	H
101316	13.40652	G

Obrázek 7.2: Ukázka výsledku dotazu pro scénář č.2

### 7.1.3 Scénář č.3

todo

Textový popis:  
Dotaz:  
Rychlost vykonání dotazu:  $\pm X$  sekund  
Výsledek dotazu:

count(*)	percentage	section
7043	0.93196	D
41788	5.52955	E
60052	7.94631	F
71359	9.44249	H
101316	13.40652	G

Obrázek 7.3: Ukázka výsledku dotazu pro scénář č.2

## 7.2 MySQL

Pro MySQL bylo připraveno devět scénářů, které testují všechny vytvořené tabulky v databázi. Každý scénář obsahuje textový popis, název pohledu, pod kterým je scénář uložen, SQL příkaz, rychlost vykonání příkazu a ukázkou výsledků. Každý scénář odpovídá jednomu pohledu v databázi.

### 7.2.1 Scénář č.1

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

**Název pohledu:** ten\_IL\_institutes\_2015

**SQL:**

```
select count(*), count(*) * 100.0 / ((select count(*) from inventors left
    outer join patents on inventors.id_patent = patents.id where YEAR(
    patents.patent_date) = 2015 and patents.patent_id like '%IL%') * 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:** ±1,4 sekundy

**Výsledek dotazu:**

count(*)	percentage	inventor
39	1.00309	DOW AGROSCIENCES LLC
29	0.74588	NOVARTIS AG
29	0.74588	F. HOFFMANN-LA ROCHE AG
29	0.74588	GENENTECH, INC.
24	0.61728	RAYTHEON COMPANY
22	0.56584	QUALCOMM INCORPORATED
20	0.51440	BIOSENSE WEBSTER (ISRAEL) LTD.
17	0.43724	MICROSOFT CORPORATION
17	0.43724	SANOFI
16	0.41152	YERKES, CARLA N.

Obrázek 7.4: Ukáзка výsledku dotazu pro scénář č.1

### 7.2.2 Scénář č.2

**Textový popis:** Pět nejméně patentovaných oborů v Kanadě od roku 2010.

**Název pohledu:** five\_CA\_sections\_ge\_2010

**SQL:**

```
select count(*), count(*) * 100.0 / ((select count(*) from
    patent_classification left outer join patents on patents.id =
    patent_classification.id_patent where YEAR(patents.patent_date) >=
    2010 and patents.patent_id like '%CA%') * 1.0) as percentage,
    classification.section from classification left outer join
    patent_classification on classification.id = patent_classification.
    id_classification left outer join patents on patents.id =
    patent_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:** ±10 sekund

**Výsledek dotazu:**

count(*)	percentage	section
7043	0.93196	D
41788	5.52955	E
60052	7.94631	F
71359	9.44249	H
101316	13.40652	G

Obrázek 7.5: Ukázka výsledku dotazu pro scénář č.2

### 7.2.3 Scénář č.3

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

**Název pohledu:** twenty\_ES\_classification\_2008

**SQL:**

```
select count(*), count(*) * 100.0 / ((select count(*) from
    patent_classification left outer join patents on patents.id =
    patent_classification.id_patent where YEAR(patents.patent_date) =
    2008 and patents.patent_id LIKE '%ES%') * 1.0) as percentage,
    classification.section, classification.class, classification.subclass
    from classification left outer join patent_classification on
    classification.id = patent_classification.id_classification left
    outer join patents on patents.id = patent_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:** ±1,3 sekundy

**Výsledek dotazu:**

count(*)	percentage	section	class	subclass
56	2.73571	B	65	D
49	2.39375	E	04	G
40	1.95408	E	04	B
39	1.90523	A	61	K
35	1.70982	G	01	N
27	1.31900	F	24	J
27	1.31900	E	06	B
25	1.22130	E	04	H
24	1.17245	B	60	R
23	1.12360	A	23	L
23	1.12360	C	02	F
23	1.12360	A	01	K
22	1.07474	D	06	F
21	1.02589	B	01	D
21	1.02589	A	47	L
20	0.97704	F	03	D
19	0.92819	E	04	C
19	0.92819	A	47	C
18	0.87934	E	04	F
18	0.87934	G	02	B

Obrázek 7.6: Ukázka výsledku dotazu pro scénář č.3

#### 7.2.4 Scénář č.4

**Textový popis:** Deset autorů s největším počtem patentů ze všech zemí.

**Název pohledu:** ten\_most\_authors

**SQL:**

```
select count(*), count(*) * 100.0 / ((select count(*) from inventors) *  
    1.0) as percentage, inventors.inventor from inventors group by  
    inventors.inventor order by count(*) desc, percentage desc LIMIT 10;
```

**Rychlost vykonání dotazu:**  $\pm 8$  sekund

**Výsledek dotazu:**

count(*)	percentage	inventor
26987	0.65261	Квасенков Олег Иванович (RU)
5123	0.12389	Щепочкина Юлия Алексеевна (RU)
1932	0.04672	Кочетов Олег Савельевич (RU)
1675	0.04051	QUALCOMM INCORPORATED
1430	0.03458	Квасенков Олег Иванович
1158	0.02800	ASTRAZENECA AB
1113	0.02692	NOVARTIS AG
1035	0.02503	F. HOFFMANN-LA ROCHE AG
948	0.02293	BASF SE
904	0.02186	Consiglio Nazionale delle Ricerche - CNR

Obrázek 7.7: Ukázka výsledku dotazu pro scénář č.4

### 7.2.5 Scénář č.5

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

**Název pohledu:** five\_languages\_2003

**SQL:**

```
select count(*), count(*) * 100.0 / ((select count(*) from patents where
patents.language not like '%-%') * 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:** ±5,3 sekund

**Výsledek dotazu:**

count(*)	percentage	language
69	0.00348	PT
869	0.04381	LT
75489	3.80605	ES
299732	15.11205	FR
614033	30.95865	RU

Obrázek 7.8: Ukázka výsledku dotazu pro scénář č.5

### 7.2.6 Scénář č.6

**Textový popis:** Deset Institucí / autorů s patenty pokrývající největší množství oborů ve Španělsku.

**Název pohledu:** ten\_ES\_authors\_most\_kinds

**SQL:**

```
select count(distinct classification.section), count(*) * 100.0 / ((
    select count(*) from inventors left outer join patents on patents.id
    = inventors.id_patent left outer join patent_classification on
    patents.id = patent_classification.id_patent left outer join
    classification on classification.id = patent_classification.
    id_classification where classification.section is not null and
    patents.country like '%ES%') * 1.0) as percentage, inventors.inventor
    from inventors left outer join patents on patents.id = inventors.
    id_patent left outer join patent_classification on patents.id =
    patent_classification.id_patent left outer join classification on
    classification.id = patent_classification.id_classification where
    classification.section is not null and patents.country like '%ES%'
    group by inventors.inventor order by count(distinct classification.
    section) desc, percentage desc LIMIT 10;
```

**Rychlost vykonání dotazu:** ±2,2 sekundy

**Výsledek dotazu:**

count(distinct classification.section)	percentage	inventor
7	0.09014	PORRAS VILA FCO. JAVIER
6	0.09833	PORRAS VILA F. JAVIER
6	0.06555	TRENCH ROCA LLUIS
6	0.05736	ALET VIDAL JOSEP
6	0.03687	GUTIERREZ MIGUELEZ ANGEL
6	0.03687	LLOVERAS MACIA JOAQUIM
5	0.14340	LLORENTE GONZALEZ JOSE IGNACIO
5	0.08604	ALONSO ESTEBAN RAFAEL
5	0.06146	MANDALUNIZ BILBAO JOSEBA
5	0.04917	SEGADE ROBLEDA ABRAHAM

Obrázek 7.9: Ukázka výsledku dotazu pro scénář č.6

### 7.2.7 Scénář č.7

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

**Název pohledu:** five\_most\_country\_ge\_2018

**SQL:**

```
select count(*), count(*) * 100.0 / ((select count(*) from patents where  
YEAR(patents.patent_date) >= 2018) * 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:** ±2,4 sekundy

**Výsledek dotazu:**

count(*)	percentage	country
122299	39.44658	RU
112202	36.18987	CA
56954	18.37007	FR
7295	2.35294	IL
5769	1.86075	UK

Obrázek 7.10: Ukázka výsledku dotazu pro scénář č.7

### 7.2.8 Scénář č.8

**Textový popis:** Nejvíce používaný typ patentu ve Francii.

**Název pohledu:** most\_FR\_kind

**SQL:**

```
select count(*), count(*) * 100.0 / ((select count(*) from patents where  
patents.patent_id like '%FR%' and patents.kind not like '%-%') * 1.0)  
as percentage, patents.kind from patents where patents.patent_id  
like '%FR%' and patents.kind not like '%-%' group by patents.kind  
order by count(*) desc, percentage desc;
```

**Rychlost vykonání dotazu:** ±5,5 sekund

**Výsledek dotazu:**

count(*)	percentage	kind
264824	97.65293	A

Obrázek 7.11: 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.

**Název pohledu:** fifteen\_UK\_author\_textile\_2013

**SQL:**

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

**Rychlost vykonání dotazu:** ±0,3 sekundy

**Výsledek dotazu:**

count(*)	percentage	inventor
8	6.20155	Gould Nigel
4	3.10078	Philips Andrew
4	3.10078	Lee Sangik
3	2.32558	Trokhan Paul Dennis
3	2.32558	Kim Seonghwan
3	2.32558	Weisman Paul Thomas
3	2.32558	Dreher Andreas Josef
3	2.32558	Sivik Mark Robert
3	2.32558	Kim Jeongyun
3	2.32558	Hamad-Ebrahimpour Alyssandrea Hope
3	2.32558	Park Bio
3	2.32558	Gordon Gregory Charles
2	1.55039	Hilhorst Ronald Jozephus
2	1.55039	Schmidt Michael A
2	1.55039	Paszke Gary

Obrázek 7.12: Ukázka výsledku dotazu pro scénář č.9

## 8 Závěr

V rámci této práce se autor seznámil s dostupnými národními zdroji dat o patentech. Celosvětové patentové instituce nebyly prostudovány z důvodu již existující diplomové práce, která byla zaměřená právě na tyto celosvětové instituce.

Celkem bylo prostudováno 51 národních patentových institucí z celého světa. U institucí byla zkoumána hlavně dostupnost patentových dat (zda patentová instituce poskytuje svá data ke stažení zdarma nebo za peníze) a validita dat. Data byla vyhodnocena jako validní tehdy, když obsahovali všechny povinné atributy, kterými jsou: ID patentu, titulek patentu, datum přihlášení a existující autor. Z 51 národních patentových institucí splňovalo tyto dvě podmínky pouze 10 institucí, které poskytly necelé dva miliony validních patentových dat.

Vzhledem k získaným datům z patentových institucí byly prozkoumány možné typy databází, které by umožňovali jejich efektivní vytěžování. Při průzkumu bylo porovnáváno pouze šest nejznámějších typů databází, které by mohli ukládat patentová data. Z šesti typů dat byly nakonec vybrány dva typy databází, každý pro jiný účel. První typ, relační databáze, poslouží k rychlému získávání statistik o patentech. Druhý typ, databáze dokumentů, poslouží k ukládání celých souborů s patentovými daty. Jako existující řešení pro relační databázi bylo vybráno MySQL, pro databázi dokumentů zase MongoDB, které se pomocí Apache Kafka propojilo s full-textovým vyhledávačem Elasticsearch.

Zadání práce bylo splněno ve všech bodech. Zvolená databázová řešení umožňují efektivní vytěžování patentových dat pomocí full-textového vyhledávače Elasticsearch v případě MongoDB, a pomocí SQL dotazů v MySQL pro získávání statistik. Efektivní vytěžování bylo otestováno na třech scénářích pro MongoDB, a devíti scénářích pro MySQL. Zvolená řešení lze jednoduše nainstalovat pomocí Dockeru.

# Zkratky

**ACID** Atomicity, Consistency, Isolation, Durability 8, 12, 20, 22

**API** Application Programming Interface 20

**ASCII** American Standard Code for Information Interchange 27

**CRUD** Create, Read, Update, Delete 25, 26, 50

**CSV** Comma-separated values 38, 53

**DBMS** Database Management Systems 7

**EPO** European Patent Office 1, 29

**FTP** File Transfer Protocol 37

**GPL** GNU General Public License 19

**HTTP** Hypertext Transfer Protocol 51

**IPC** International Patent Classification 5, 6, 46

**JSON** JavaScript Object Notation 14, 18, 20, 53

**SQL** MongoDB Query Language 25

**PDF** Portable Document Format 36

**SQL** Structured Query Language 14, 19, 20, 22, 24–26, 37, 42, 45, 46, 66

**URL** Uniform Resource Locator 54

**USPTO** United States Patent and Trademark Office 3–5, 29

**WIPO** World Intellectual Property Organization 29

**XML** Extensible Markup Language 18, 20, 36–39, 44, 53–55

**YAML** YAML Ain't Markup Language 47

# 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] *International Patent Classification (IPC)* [online]. Espacenet, 2016. [cit. 02.05.2022]. Dostupné z: [https://is.espacenet.com/help?locale=en\\_IS&method=handleHelpTopic&topic=ipc](https://is.espacenet.com/help?locale=en_IS&method=handleHelpTopic&topic=ipc).
- [10] *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>.
- [11] *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>.
- [12] *IPC Publication* [online]. WIPO, 2022. [cit. 01.05.2022]. Dostupné z: <https://ipcpub.wipo.int/?menulang=en>.

- [13] *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>.
- [14] *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/>.
- [15] *What is a MongoDB Query?* [online]. GeeksforGeeks, 2021. [cit. 26.04.2022]. Dostupné z: <https://www.geeksforgeeks.org/what-is-a-mongodb-query/>.
- [16] *MongoDB Query Language* [online]. Devopedia, 2021. [cit. 27.04.2022]. Dostupné z: <https://devopedia.org/mongodb-query-language>.
- [17] *What is MySQL?* [online]. MySQL. [cit. 26.04.2022]. Dostupné z: <https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html>.
- [18] *What is a Graph Database?* [online]. Neo4j. [cit. 27.04.2022]. Dostupné z: <https://neo4j.com/developer/graph-database/>.
- [19] *PATENT NUMBER, PUBLICATION NUMBER* [online]. IamIP, 2019. [cit. 01.05.2022]. Dostupné z: <http://patentwiki.iamip.com/publication-number>.
- [20] *About PostgreSQL* [online]. PostgreSQL. [cit. 28.04.2022]. Dostupné z: <https://www.postgresql.org/about/>.
- [21] *Design Patent Application Guide* [online]. USPTO, 2017. [cit. 02.05.2022]. Dostupné z: <https://www.uspto.gov/patents/basics/types-patent-applications/design-patent-application-guide#def>.
- [22] *General information concerning patents* [online]. USPTO, 2018. [cit. 02.05.2022]. Dostupné z: <https://www.uspto.gov/patents/basics/general-information-patents>.
- [23] *General Information About 35 U.S.C. 161 Plant Patents* [online]. USPTO, 2017. [cit. 02.05.2022]. Dostupné z: <https://www.uspto.gov/patents/basics/types-patent-applications/general-information-about-35-usc-161#heading-1>.
- [24] *Nonprovisional (Utility) Patent Application Filing Guide* [online]. USPTO, 2018. [cit. 02.05.2022]. Dostupné z: <https://www.uspto.gov/patents/basics/types-patent-applications/nonprovisional-utility-patent#heading-1>.

- [25] 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/>.
- [26] CHATHAM, M. *Structured Query Language By Example - Volume I: Data Query Language*. lulu.com, 2012. ISBN 1291199519.
- [27] DATE, C. J. *An introduction to database systems*. Pearson/Addison Wesley, 2004. ISBN 0321197844.
- [28] FRANKLOVÁ, M. *Patenty a jejich klasifikace* [online]. ČVUT. [cit. 03.05.2022]. Dostupné z: [http://pspev.cvut.cz/PSPEV\\_CD/V11/main.html?ID=0](http://pspev.cvut.cz/PSPEV_CD/V11/main.html?ID=0).
- [29] GARCIA-MOLINA, e. a. H. *Database Systems: The Complete Book 2nd Edition*. Pearson, 2009. ISBN 0136067018.
- [30] 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/>.
- [31] JOHNSTONE, e. a. N. Renewable Energy Policies and Technological Innovation: Evidence Based on Patent Counts. *Environmental and Resource Economics*. November 2009, 3, 1, s. 38. doi: 10.1007/s10640-009-9309-1. Dostupné z: [https://www.researchgate.net/publication/225430825\\_Renewable\\_Energy\\_Policies\\_and\\_Technological\\_Innovation\\_Evidence\\_Based\\_on\\_Patent\\_Counts](https://www.researchgate.net/publication/225430825_Renewable_Energy_Policies_and_Technological_Innovation_Evidence_Based_on_Patent_Counts).
- [32] KARKI, S. *What Is LevelDB* [online]. C# Corner, 2021. [cit. 28.04.2022]. Dostupné z: <https://www.c-sharpcorner.com/article/what-is-leveldb2/>.
- [33] LEBER, e. a. C. *The Difference Between Trademarks and Design Patents: What you need to know* [online]. Alt Legal, 2021. [cit. 03.05.2022]. Dostupné z: <https://www.altlegal.com/blog/the-difference-between-trademarks-and-design-patents-what-you-need-to-know/>.
- [34] 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/>.
- [35] LUTKEVICH, B. *database (DB)* [online]. Tech Target, 2021. [cit. 21.04.2022]. Dostupné z: <https://www.techtarget.com/searchdatamanagement/definition/database>.

- [36] PRESSMAN, R. D. S. *Nolo's Patents for Beginners*. Nolo, 2004. ISBN 1413300715.
- [37] SEKHON, S. *What is Neo4j?* [online]. DEV Community, 2020. [cit. 28.04.2022]. Dostupné z: <https://dev.to/sukhbirsekhon/what-is-neo4j-8jc>.
- [38] THAKUR, D. *What is Object Oriented Database (OODB)? Advantages and Disadvantages of OODBMS*. [online]. ComputerNotes. [cit. 27.04.2022]. Dostupné z: <https://ecomputernotes.com/database-system/adv-database/object-oriented-database-oodb>.
- [39] VAISH, G. *Getting Started with NoSQL*. Packt Publishing, 2013. ISBN 1849694982.
- [40] 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>.
- [41] YANG, J. *Parts of a Utility Patent Application (Chapter 11)* [online]. OC Patent Lawyer, 2018. [cit. 03.05.2022]. Dostupné z: <https://ocpatentlawyer.com/lesson/basics-utility-patent-application-sections/>.

# A Uživatelská dokumentace

Pro instalaci celého úložiště je potřeba mít nainstalovanou aplikaci **Docker**.

## A.1 Adresářová struktura

Adresářová struktura je následující:

- **Aplikace\_a\_knihovny**
  - **docker**
    - \* **init** - Inicializační skripty pro MySQL, MongoDB a Apache Kafka.
    - \* **mongo\_import** - Soubory pro import dat do MongoDB.
    - \* **docker-compose.yml**
    - \* **Dockerfile**
  - **projects** - Java projekty použité pro extrakci a import dat do MongoDB a MySQL.
- **Poster**
- **Text\_prace**
- **Vstupni\_data**
  - **mongodb**
    - \* **patents** - Složka se soubory obsahující patentová data.
  - **mysql**
    - \* **patents.sql** - SQL soubor s patentovými daty.
- **Readme.txt**



## A.2 Instalace úložiště

Instalace úložiště a všech potřebných aplikací se provádí následovně:

1. Otevřít konzoli ve složce `./Aplikace_a_knihovny/docker`.
2. Spustit příkaz **docker-compose up**. Docker začne stahovat všechny aplikace (jejich image), následně pro ně vytvoří kontejnery, které poté spustí (viz obrázky č. A.1, A.2).

```
C:\Windows\System32\cmd.exe - docker-compose up
Microsoft Windows [Version 10.0.19044.1645]
(c) Microsoft Corporation. Všechna práva vyhrazena.

C:\Users\vojda\Desktop\A19N0028P_prilohy\Aplikace_a_knihovny\docker>docker-compose up
Creating network "docker_default" with the default driver
Pulling elasticsearch (elasticsearch:8.2.0)...
8.2.0: Pulling from library/elasticsearch
e0b25ef51634: Pull complete
860caabdf263: Pull complete
9fbc6bc43ac5: Pull complete
9d4f6737f430: Pull complete
10f01841fd3e: Pull complete
dae1e3bba098: Pull complete
0a3767e40ef9: Pull complete
7d786dfd085d: Pull complete
7ce904f28ed3: Pull complete
Digest: sha256:6bd33a35f529d349d8d385856b138d73241555abf2851287c055665494680b8d
Status: Downloaded newer image for elasticsearch:8.2.0
Pulling elasticvue (cars10/elasticvue:0.39.0)...
0.39.0: Pulling from cars10/elasticvue
cbdbe7a5bc2a: Pull complete
c554c602ff32: Pull complete
67bcb44c5d45: Pull complete
5bdcca788878: Pull complete
ae5e52a175e2: Pull complete
Digest: sha256:81fcbd272f5a91cc62daefda7ee39a597a852b94ce32c2cafdc30a6fa0a6172d
Status: Downloaded newer image for cars10/elasticvue:0.39.0
Pulling mongo (mongo:5.0.6)...
5.0.6: Pulling from library/mongo
e0b25ef51634: Already exists
c7a086fc80ea: Pull complete
7a6592c2fb05: Pull complete
5dad2281c276: Pull complete
34073132290c: Pull complete
ec793bc1a2ae: Pull complete
660887418c9b: Pull complete
97291b67bd8e: Downloading [==> 9.096MB/211.1MB
65956d0b19fc: Download complete
b07897fe2a77: Download complete
```

Obrázek A.1: Docker compose a stahování aplikací.

```
Creating mysql ... done
Creating zookeeper ... done
Creating mongo-express ... done
Creating elasticvue ... done
Creating elasticsearch ... done
Creating mongo ... done
Creating mongo-setup ... done
Creating broker ... done
Creating phpmyadmin ... done
Creating connect ... done
Creating connect-setup ... done
Attaching to mongo, zookeeper, mysql, elasticvue, elasticsearch, mongo-express, mongo-setup, phpmyadmin, broker, connect, connect-setup
broker | ==> User
```

Obrázek A.2: Vytvoření kontejnerů v dockeru.

3. Veškerá instalace a nastavování končí ve chvíli, kdy konektor mezi MongoDB a Apache Kafka začne posílat zprávy (viz obrázek č. A.3).

```

{"@version": "1.2.0", "service.name": "ES_ECS", "event.dataset": "elasticsearch.server", "process.thread.name": "elasticsearch[767b27f51b30][generic][t12]", "log.logger": "org.elasticsearch.ingest.geop.GeopDownloader", "elasticsearch.cluster.uid": "sbvzyt89Ru0MkR5yy0Qw", "elasticsearch.node.id": "KG3lmcwS4u6FPNqHqQw", "elasticsearch.node.name": "767b27f51b30", "elasticsearch.cluster.name": "docker-cluster"}
{"@timestamp": "2022-05-12 09:11:47,242", "log.level": "INFO", "message": "Copying existing data on the following namespaces: [patents.patents] (com.mongodb.kafka.connect.source.MongoCopyDataManager)"
{"@timestamp": "2022-05-12 09:11:47,248", "log.level": "INFO", "message": "Started MongoDB source task (com.mongodb.kafka.connect.source.MongoSourceTask)"
{"@timestamp": "2022-05-12 09:11:47,248", "log.level": "INFO", "message": "WorkerSourceTask(id-mongo-source-connector-0) Source task finished initialization and start (org.apache.kafka.connect.runtime.WorkerSourceTask)"
{"@timestamp": "2022-05-12 09:11:47,357", "log.level": "INFO", "message": "retrieve geop database [geolite2-Country.mmdb] from [geop_databases] to [/tmp/elasticsearch-12878839149275137266/geop-databases/KG3lmcwS4u6FPNqHqQw/Geolite2-Country.mmdb.gr]", "ecs.version": "1.2.0", "service.name": "ES_ECS", "event.dataset": "elasticsearch.server", "process.thread.name": "elasticsearch[767b27f51b30][clusterApplierServiceUpdateTask][t11]", "log.logger": "org.elasticsearch.ingest.geop.DatabaseNodeService", "elasticsearch.cluster.uid": "sbvzyt89Ru0MkR5yy0Qw", "elasticsearch.node.id": "KG3lmcwS4u6FPNqHqQw", "elasticsearch.node.name": "767b27f51b30", "elasticsearch.cluster.name": "docker-cluster"}
{"@timestamp": "2022-05-12 09:11:47,361", "log.level": "INFO", "message": "Successfully downloaded geop database [geolite2-Country.mmdb]", "ecs.version": "1.2.0", "service.name": "ES_ECS", "event.dataset": "elasticsearch.server", "process.thread.name": "elasticsearch[767b27f51b30][generic][t12]", "log.logger": "org.elasticsearch.ingest.geop.GeopDownloader", "elasticsearch.cluster.uid": "sbvzyt89Ru0MkR5yy0Qw", "elasticsearch.node.id": "KG3lmcwS4u6FPNqHqQw", "elasticsearch.node.name": "767b27f51b30", "elasticsearch.cluster.name": "docker-cluster"}
{"@timestamp": "2022-05-12 09:11:47,443", "log.level": "INFO", "message": "GroupCoordinator 1: Stabilized group connect-elasticsearch-sink-connector-0, groupId=connect-elasticsearch-sink-connector-0, groupId=connect-elasticsearch-sink-connector-0, consumerOffsets=32 (kafka.coordinator.group.GroupCoordinator)"
{"@timestamp": "2022-05-12 09:11:47,444", "log.level": "INFO", "message": "Consumer clientId=connector-consumer-elasticsearch-sink-connector-0, groupId=connect-elasticsearch-sink-connector-0 Successfully joined group with generation Generation{generationId=1, memberId=connector-consumer-elasticsearch-sink-connector-0-72319f21-eb75-4b59-bd55-9c090b090ede, protocol=range} (org.apache.kafka.clients.consumer.internals.AbstractCoordinator)"
{"@timestamp": "2022-05-12 09:11:47,448", "log.level": "INFO", "message": "Consumer clientId=connector-consumer-elasticsearch-sink-connector-0, groupId=connect-elasticsearch-sink-connector-0, groupId=connect-elasticsearch-sink-connector-0 Finished assignment for group at generation 1: {connector-consum", "ecs.version": "1.2.0", "service.name": "ES_ECS", "event.dataset": "elasticsearch.server", "process.thread.name": "elasticsearch[767b27f51b30][generic][t12]", "log.logger": "org.elasticsearch.ingest.geop.DatabaseNodeService", "elasticsearch.cluster.uid": "sbvzyt89Ru0MkR5yy0Qw", "elasticsearch.node.id": "KG3lmcwS4u6FPNqHqQw", "elasticsearch.node.name": "767b27f51b30", "elasticsearch.cluster.name": "docker-cluster"}
{"@timestamp": "2022-05-12 09:11:47,457", "log.level": "INFO", "message": "GroupCoordinator 1: Assignment received from leader for group connect-elasticsearch-sink-connector-0 (kafka.coordinator.group.GroupCoordinator)"
{"@timestamp": "2022-05-12 09:11:47,460", "log.level": "INFO", "message": "Consumer clientId=connector-consumer-elasticsearch-sink-connector-0, groupId=connect-elasticsearch-sink-connector-0, groupId=connect-elasticsearch-sink-connector-0 Successfully synced group in generation Generation{generationId=1, memberId=connector-consumer-elasticsearch-sink-connector-0-72319f21-eb75-4b59-bd55-9c090b090ede, protocol=range} (org.apache.kafka.clients.consumer.internals.AbstractCoordinator)"
{"@timestamp": "2022-05-12 09:11:47,461", "log.level": "INFO", "message": "Consumer clientId=connector-consumer-elasticsearch-sink-connector-0, groupId=connect-elasticsearch-sink-connector-0, groupId=connect-elasticsearch-sink-connector-0 Notifying assignor about the new Assignment(partitions=[patents.patents-0]) (org.apache.kafka.clients.consumer.internals.ConsumerCoordinator)"
{"@timestamp": "2022-05-12 09:11:47,461", "log.level": "INFO", "message": "Consumer clientId=connector-consumer-elasticsearch-sink-connector-0, groupId=connect-elasticsearch-sink-connector-0, groupId=connect-elasticsearch-sink-connector-0 Adding newly assigned partitions: patents.patents-0 (org.apache.kafka.clients.consumer.internals.ConsumerCoordinator)"
{"@timestamp": "2022-05-12 09:11:47,463", "log.level": "INFO", "message": "Successfully loaded geop database file [geolite2-Country.mmdb]", "ecs.version": "1.2.0", "service.name": "ES_ECS", "event.dataset": "elasticsearch.server", "process.thread.name": "elasticsearch[767b27f51b30][generic][t17]", "log.logger": "org.elasticsearch.ingest.geop.DatabaseNodeService", "elasticsearch.cluster.uid": "sbvzyt89Ru0MkR5yy0Qw", "elasticsearch.node.id": "KG3lmcwS4u6FPNqHqQw", "elasticsearch.node.name": "767b27f51b30", "elasticsearch.cluster.name": "docker-cluster"}
{"@timestamp": "2022-05-12 09:11:47,464", "log.level": "INFO", "message": "Consumer clientId=connector-consumer-elasticsearch-sink-connector-0, groupId=connect-elasticsearch-sink-connector-0, groupId=connect-elasticsearch-sink-connector-0 Found no committed offset for partition patents.patents-0 (org.apache.kafka.clients.consumer.internals.ConsumerCoordinator)"
{"@timestamp": "2022-05-12 09:11:47,488", "log.level": "INFO", "message": "Consumer clientId=connector-consumer-elasticsearch-sink-connector-0, groupId=connect-elasticsearch-sink-connector-0, groupId=connect-elasticsearch-sink-connector-0 Resetting offset for partition patents.patents-0 to position FetchPosition(offset=0, offsetEpoch=Optional.empty, currentLeader=LeaderAndEpoch{leader=Optional{broker:38892 (id: 1 rack: null)}, epoch=0}) (org.apache.kafka.clients.consumer.internals.SubscriptionState)"
{"@timestamp": "2022-05-12 09:11:47,488", "log.level": "INFO", "message": "Successfully loaded geop database file [geolite2-city.mmdb]", "ecs.version": "1.2.0", "service.name": "ES_ECS", "event.dataset": "elasticsearch.server", "process.thread.name": "elasticsearch[767b27f51b30][generic][t18]", "log.logger": "org.elasticsearch.ingest.geop.DatabaseNodeService", "elasticsearch.cluster.uid": "sbvzyt89Ru0MkR5yy0Qw", "elasticsearch.node.id": "KG3lmcwS4u6FPNqHqQw", "elasticsearch.node.name": "767b27f51b30", "elasticsearch.cluster.name": "docker-cluster"}
{"@timestamp": "2022-05-12 09:11:52,249", "log.level": "INFO", "message": "Shutting down executors (com.mongodb.kafka.connect.source.MongoSourceTask)"
{"@timestamp": "2022-05-12 09:11:52,249", "log.level": "INFO", "message": "Finished copying existing data from the collection(s). (com.mongodb.kafka.connect.source.MongoSourceTask)"
{"@timestamp": "2022-05-12 09:11:52,250", "log.level": "INFO", "message": "Watching for collection changes on 'patents.patents' (com.mongodb.kafka.connect.source.MongoSourceTask)"
{"@timestamp": "2022-05-12 09:11:52,265", "log.level": "INFO", "message": "Resuming the change stream after the previous offset: {'data': '82627CCF4E00000002B0229296E04'} (com.mongodb.kafka.connect.source.MongoSourceTask)"
{"@timestamp": "2022-05-12 09:11:55,289", "log.level": "INFO", "message": "Committing offsets (org.apache.kafka.connect.runtime.WorkerSourceTask)"
{"@timestamp": "2022-05-12 09:11:55,298", "log.level": "INFO", "message": "Flushing 0 outstanding messages for offset commit (org.apache.kafka.connect.runtime.WorkerSourceTask)"

```

Obrázek A.3: Úspěšná instalace a nastavení všech aplikací.

## A.2.1 Smazání setup kontejnerů

Nastavovací moduly slouží pouze pro prvotní nastavení MongoDB a Apache Kafka Connect, takže po jejich úspěšném vykonání je lze smazat. Smazání se provede následujícím způsobem (ukázka na obrázku č. A.4):

1. Kdekoliv v systému otevřít konzoli a zadat příkaz **docker container ls -a**, pomocí kterého zjistíme všechny existující kontejnery v dockeru.
2. Zjistíme ID kontejneru pro oba nastavovací moduly. Jména nastavovacích modulů jsou **connect-setup** a **mongo-setup**.
3. Zadáme příkaz **docker rm CONTAINER\_ID**, kde *CONTAINER\_ID* je ID kontejneru nastavovacího modulu.

```

C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19044.1645]
(c) Microsoft Corporation. Všechna práva vyhrazena.

C:\Users\vojda\Desktop\A19N0028P_prilohy\Aplikace_a_knihovny\docker>docker container ls -a

```

CONTAINER ID	IMAGE	COMMAND	NAMES	CREATED	STATUS	PORTS
468ba1ad7933	docker_connect-setup	"sh /connect.sh"	connect-setup	14 seconds ago	Up Less than a second	
13dae19824dd	confluentinc/cp-kafka-connect:6.1.0	"bash -c 'echo \"Inst...\""	connect	20 seconds ago	Up 14 seconds (health: starting)	8083/tcp, 909
8db21e2d5814	phpmyadmin:5.1.3	"/docker-entrypoint..."	phpmyadmin	21 seconds ago	Exited (0) 2 seconds ago	
7cb30d10a4d4	confluentinc/cp-kafka:6.1.0	"/etc/confluent/dock..."	broker	23 seconds ago	Up 19 seconds	0.0.0.0:9092-
59092/tcp, 0.0.0.0:9101->9101/tcp, 0.0.0.0:29092->29092/tcp	mongo:5.0.6	"/mongo_init.sh"	mongo-setup	24 seconds ago	Exited (0) 2 seconds ago	
7b332e361e32	mysql:8.0.29	"docker-entrypoint.s..."	mysql	27 seconds ago	Up 21 seconds	33060/tcp, 0.
0.0.0:6033->3306/tcp	cars10/elasticvue:0.39.0	"nginx -g 'daemon of..."	elasticvue	27 seconds ago	Exited (0) 3 seconds ago	
6a8cb27b6615	mongo:5.0.6	"docker-entrypoint.s..."	mongo	27 seconds ago	Up 23 seconds	0.0.0.0:27017
831cc2bb8075	mongo-express:0.54.0	"tini -- /docker-ent..."	mongo-express	27 seconds ago	Exited (143) 3 seconds ago	
->27017/tcp	elasticsearch:8.2.0	"/bin/tini -- /usr/l..."	elasticsearch	27 seconds ago	Exited (143) 3 seconds ago	
90561a165f82	confluentinc/cp-zookeeper:6.1.0	"/etc/confluent/dock..."	zookeeper	27 seconds ago	Up 23 seconds	2888/tcp, 0.0
084b5e57d358						
8403b7b54809						
0.0.0:2181->2181/tcp, 3888/tcp						

```

C:\Users\vojda\Desktop\A19N0028P_prilohy\Aplikace_a_knihovny\docker>docker rm 7b332e361e32 468ba1ad7933
7b332e361e32
468ba1ad7933

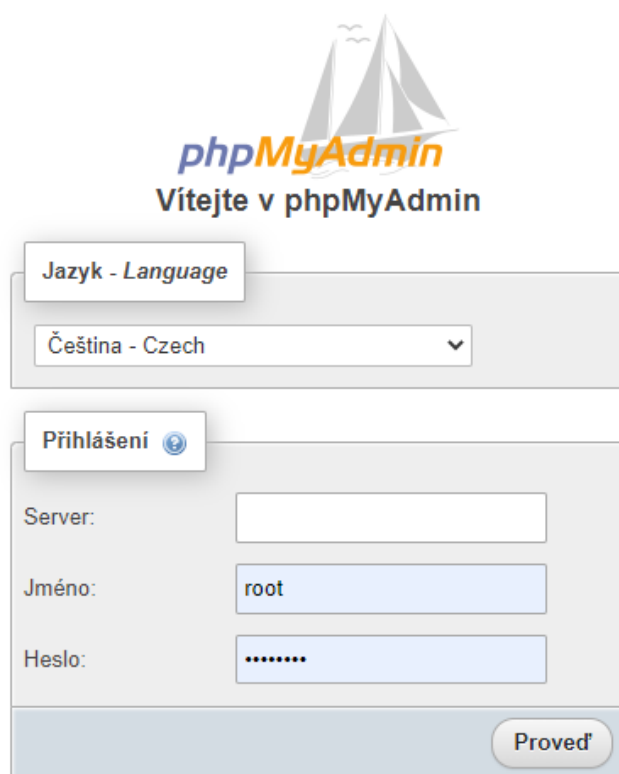
```

Obrázek A.4: Smazání kontejnerů pro nastavovací moduly.

## A.3 Import dat do MySQL

Import dat se provádí pomocí PHPMyAdmin na adrese **http://localhost:8082/**.  
Postup pro import je následující:

1. Ve webovém prohlížeči otevřít stránku **http://localhost:8082/**.
2. Přihlásit se s následujícími údaji: **Jméno:** *root*, **Heslo:** *password* (viz obrázek č. A.5)



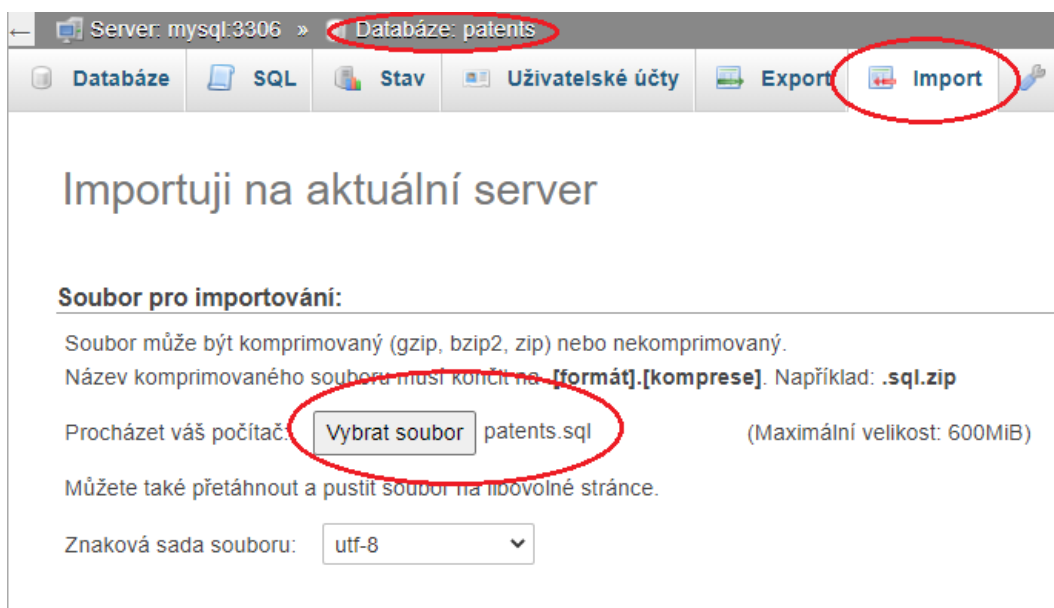
Obrázek A.5: PHPMyAdmin přihlášení.

3. Po přihlášení kliknout na databázi **patents** (viz obrázek č. A.6).



Obrázek A.6: Databáze v PHPMyAdmin.

4. Na horní liště kliknout na záložku **Import**.
5. Kliknout na tlačítko **Vybrat soubor** a zvolit soubor `./Vstupni_data/mysql/patents.sql` (viz obrázek č. A.7).



Obrázek A.7: Import dat v PHPMyAdmin.

6. Kliknout na tlačítko **Proved'** a počkat, než se naimportují všechna data do všech tabulek.
7. Po úspěšném importu dat se zobrazí hláška (viz obrázek č. A.8).

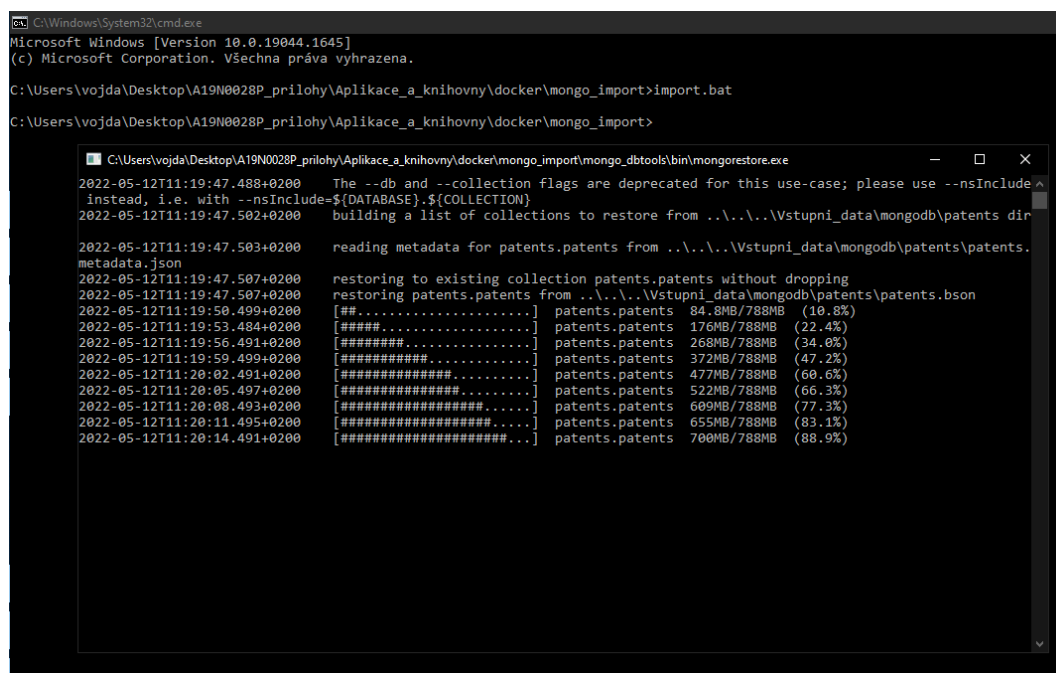


Obrázek A.8: Úspěšný import dat v PHPMyAdmin.

## A.4 Import dat do MongoDB

Import dat pro MongoDB se provádí pomocí oficiální aplikace od MongoDB s názvem **mongo\_restore.exe**. Postup pro import je následující:

1. Otevřít konzoli ve složce **./Applikace\_a\_knihovny/docker/mongo\_import**.
2. Spustit dávkový soubor **import.bat** a počkat na import dat (viz obrázek č. A.9).



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19044.1645]
(c) Microsoft Corporation. Všechna práva vyhrazena.

C:\Users\vojda\Desktop\A19N0028P_prilohy\Applikace_a_knihovny\docker\mongo_import>import.bat

C:\Users\vojda\Desktop\A19N0028P_prilohy\Applikace_a_knihovny\docker\mongo_import>

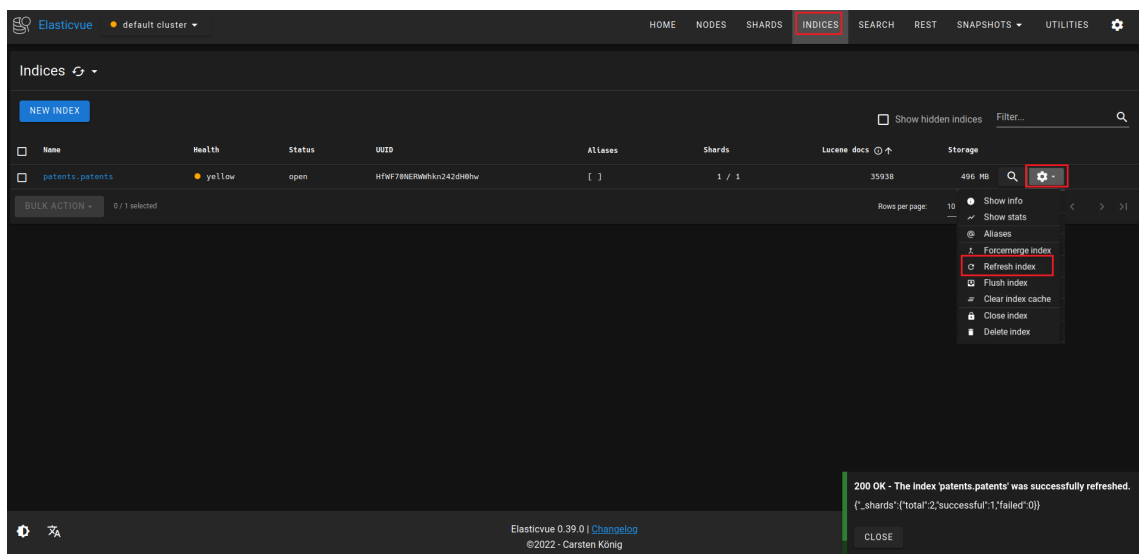
C:\Users\vojda\Desktop\A19N0028P_prilohy\Applikace_a_knihovny\docker\mongo_import\mongo_dbtools\bin\mongorestore.exe
The --db and --collection flags are deprecated for this use-case; please use --nsInclude instead, i.e. with --nsInclude=${DATABASE}.${COLLECTION}
building a list of collections to restore from ..\..\..\Vstupni_data\mongodb\patents dir
reading metadata for patents.patents from ..\..\..\Vstupni_data\mongodb\patents\patents.metadata.json
restoring to existing collection patents.patents without dropping
restoring patents.patents from ..\..\..\Vstupni_data\mongodb\patents\patents.bson
[##.....] patents.patents 84.8MB/788MB (10.8%)
[#####] patents.patents 176MB/788MB (22.4%)
[#####] patents.patents 268MB/788MB (34.0%)
[#####] patents.patents 372MB/788MB (47.2%)
[#####] patents.patents 477MB/788MB (60.6%)
[#####] patents.patents 522MB/788MB (66.3%)
[#####] patents.patents 609MB/788MB (77.3%)
[#####] patents.patents 655MB/788MB (83.1%)
[#####] patents.patents 700MB/788MB (88.9%)
```

Obrázek A.9: Import dat do MongoDB.

- Zároveň bude probíhat i import dat do Elasticsearch. Import dat do elasticu lze pozorovat v konzoli kde jsou spuštěné naše docker kontejnery (viz obrázek č. A.10). Apache Kafka vypisuje warning hlášky že se nepodařilo vykonat bulk request kvůli *java.lang.NullPointerException*. Toho se není potřeba obávat, protože to je pouze bug v rámci Apache Kafka Connect, který pouze vypisuje tyto hlášky, ale samotný import do Elasticsearch funguje.







Obrázek A.12: Aktualizace indexu v Elasticsearch.