Dokumentace k projektu z předmětu UPA

Andrea Chimenti(xchime00)
Jan Klhůfek(xklhuf01)
Jiří Václavič(xvacla31)

zima 2022

Obsah

Část I

Analýza zdrojových dat a návrh jejich uložení v NoSQL databázi

Kapitola 1

Analýza zdrojových dat

Popis struktury dat: Zdrojová data vlakových spojení jsou poskytována ve formátu XML. S využitím syntaxe XML jsou informace týkající se jednotlivých spojů strukturovány do elementů, jež jsou mnohonásobně zanořené. Je zapotřebí bližšího pochopení dat a jejich předzpracování, aby bylo dosaženo efektivního uložení dat do databáze s podporou rychlého vyhledávání. Odkaz na úložiště datových souborů pro tento projekt je poskytnut na stránkách ministerstva dopravy[1].

Rodičovský uzel dokumentu identifikuje typ zprávy vztahující se na konkrétní jízdní řád (dále JŘ). Název rodičovského uzlu může být CZPTTCISMessage nebo CZCanceledPTTMessage. Struktura zpráv se lehce liší, viz 1.2 a 1.2. Obě zprávy obsahují položky sloužící k jednoznačné identifikaci JŘ, skrývající se v elementu Identifiers, respektive PlannedTransportIdentifiers.

Element Identifiers, respektive PlannedTransportIdentifiers obsahuje idenfikátor objektu datového JŘ – **PAID** a identifikátor objektu vlaku – **TRID**. Ty společně slouží k jednoznačné identifikaci JŘ, ke kterému se daná zpráva vztahuje.

CZPTTCreation a CZPTTCancelation značí čas a datum vytvoření zprávy, významné v rámci odlišení nejaktuálnějších dat od zastaralých.

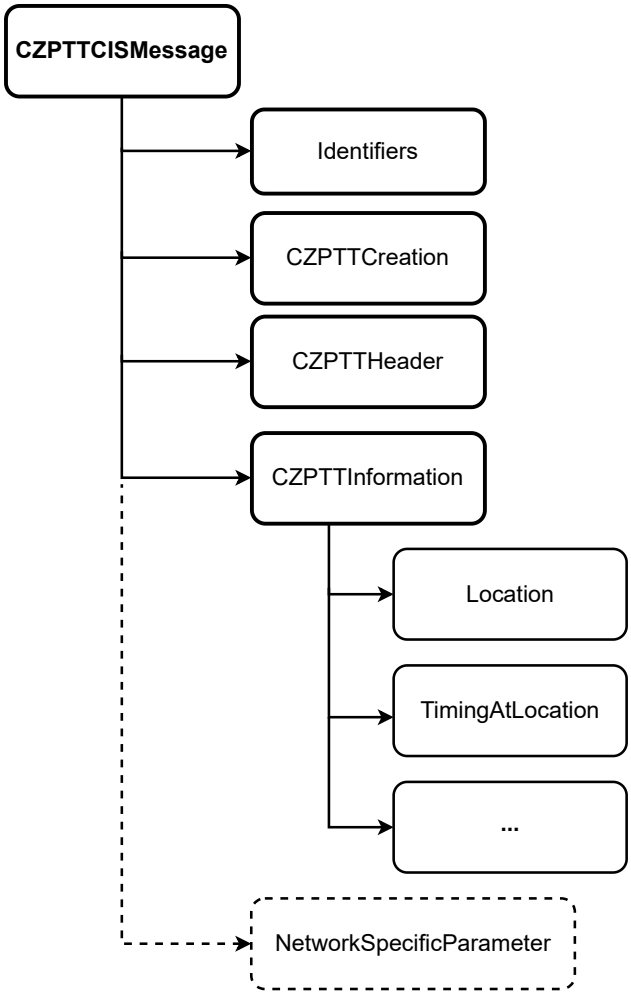
CZPTTHeader obsahuje informace o spojích jejichž trasa má počátek nebo cíl mimo území ČR. Mimo názvu zahraniční lokality je uvedeno i kódové označení cizí země, apod.

CZPTTInformation je element s největší výpovědní hodnotou. Obsahuje informace o lokacích, kterými vlak projíždí (minimálně počáteční a cílová destinace), dále také časový rozvrh příjezdu/odjezdu z/do stanice či velmi důležitou informaci o zastavení vlaku v dané lokaci.

V NetworkSpecificParameters jsou uloženy dodatečné informace, které nebyly mezinárodně schválené a tudíž jsou libovolné.

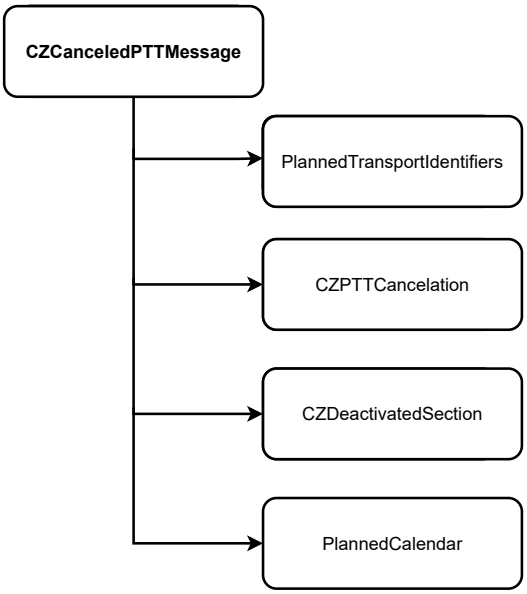
[1] `https://portal.cisjr.cz/pub/draha/celostatni/szdc/`

Zprávy o JŘ jsou do systému aktualizovány třikrát týdně. Pakliže se změní JŘ, aktualizují se také seznamy železničních stanic, zastávek a železničních tratí[2]. Kromě vhodného získávání a uložení dat do NoSQL databáze je třeba, aby uživatel mohl nad daty vyhledávat jednotlivé vlakové spoje mezi destinacemi v zadaný datum.
Podrobnější popis dat je k dostání v poskytnuté dokumentaci[3].



Obrázek 1.1: Struktura zprávy CZPTTCISMessage. Přerušovanou čarou je značený element obsahující informace nepodstatné pro účely projektu.

[2]Zdroj: `https://www.mdcr.cz/getattachment/Dokumenty/Verejna-doprava/Jizdni-rady,-kalendare-pro-jizdni-rady,-metodi-(1)/Jizdni-rady-verejne-dopravy/popis-seznamy-cis-(1).pdf.aspx?lang=cs-CZ`
[3]`https://portal.cisjr.cz/pub/draha/celostatni/szdc/Popis%20DJ%C5%98_CIS_v1_09.pdf`



Obrázek 1.2: Struktura zprávy CZCanceledPTTMessage.

Kapitola 2

Návrh způsobu uložení dat

Při zohlednění analýzy z předchozí kapitoly je nutné, aby navržená databáze splňovala následující vlastnosti:

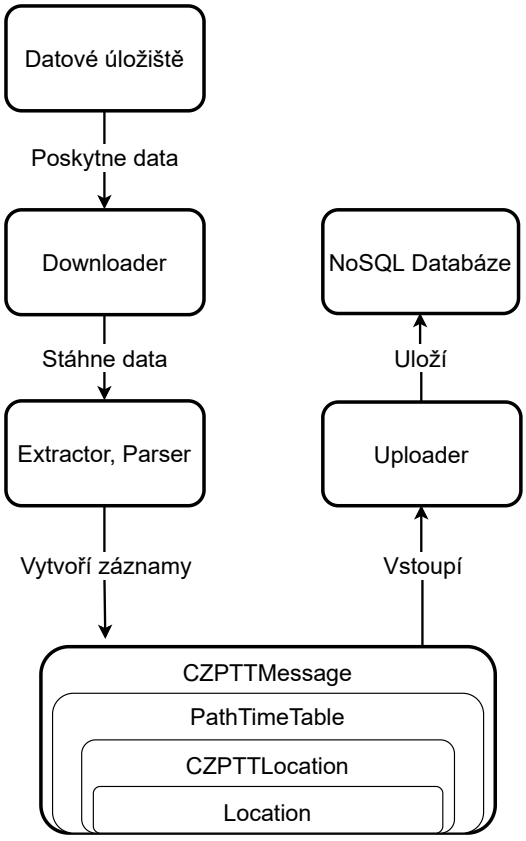
- Podpora skriptů pro stahování nových datových položek.
- Vytvoření databáze nad větším objemem dat (maximálně stovky MB) v přiměřeném čase (v rámci jednotek až desítek hodin).
- Možnost efektivního provedení dotazů pro vyhledávání spojů.
- Možnost Data Shardingu, kvůli rychlému provádění dotazů.
- Aktualizace jednotlivých částí dat, bez nutnosti smazání položek databáze.
- Snadná implementace a multiplatformní podpora.

Na základě analýzy bylo řešení defragmentováno na jednotlivě řešitelné části, jež jsou znázorněny diagramem na obrázku 2.1. Všechny diagramem vyobrazené části (kromě datového úložiště, databáze) budou implementovány pomocí tříd, nad kterými se vytvoří vhodné metody, aby byly splněny požadované funkcionality. Zvoleným jazykem byl Python pro jeho objektově orientované paradigma.

1. Downloader v rámci inicializace databáze stáhne všechna dostupná data z úložiště na serveru Ministerstva dopravy[1] pro aktuální rok 2022.
2. Stažené komprimované soubory budou následně Extractorem extrahovány a v paměti (bez nutnosti uložení extrahovaných souborů) zpracovány Parserem jednotlivých zpráv, který XML obsah transformuje do podoby Python slovníků pro uložení do databáze.

[1] `https://portal.cisjr.cz/pub/draha/celostatni/szdc/`

3. O ukládání dat do databáze se postará MongoUploader. Podle typu předzpracované zprávy se vykoná:
   - Pro zprávu **CZPTTCISMessage** dojde k pokusu o vložení nového záznamu o JŘ do kolekce v databázi, pokud v ní již není. Navíc dojde k aktualizaci a případnému přidání nové lokace ze seznamu lokací daného JŘ do kolekce lokací v databázi, které u sebe obsahují informaci o JŘ, ve kterých se vyskytují.
   - U zprávy **CZCanceledPTTMessage** dojde k aktualizaci kalendáře JŘ v databázi, ke kterému se podle identifikace vztahuje.



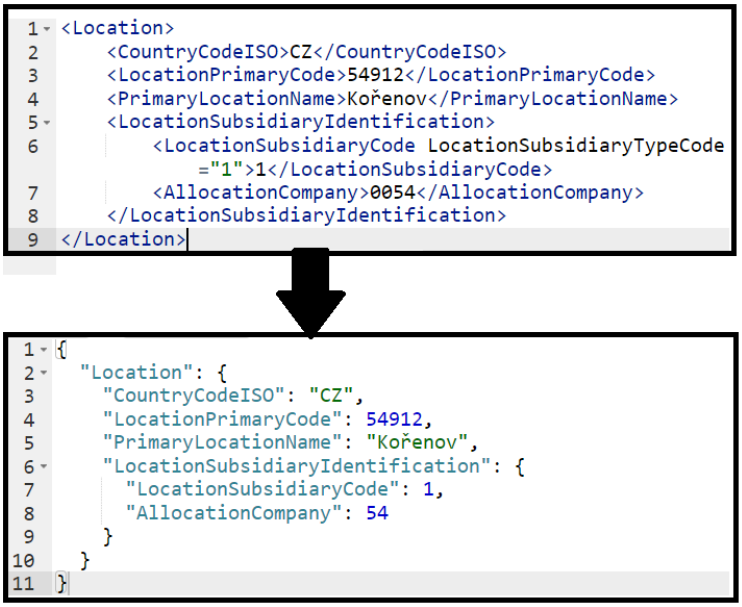
Obrázek 2.1: Dataflow diagram

Kapitola 3

Zvolená NoSQL databáze

Na základě předchozího návrhu byla vybrána NoSQL databáze MongoDB, která patří do třídy dokumentově orientovaných databází.

Dokumentově orientované databáze jsou vhodné pro zpracování velkého objemu dat, kde jsou vztahy mezi jednotlivými položkami definované pomocí zanoření v souboru typu JSON. Jelikož jsou vstupní soubory ve formátu XML je potřeba převodu do formátu JSON. V případě Pythonu je převod realizovatelný snadno a to do podoby slovníku. Na obrázku 3.1 je vidět převod konkrétních dat o lokaci, které jsou obsažené v datových souborech.



Obrázek 3.1: Transformace XML popisu loskace do formátu JSON.

MongoDB je vhodné pro tvorbu ad-hoc (nedeterministických) dotazů, indexování, rozložení zátěže, spouštění skriptů na straně serveru. Záznamy jsou řazeny do strukturovaných dokumentů BSON (binární JSON), jejichž části se dělí vždy na klíč (_id) a hodnotu.

Nad daty je prováděn automatický sharding a s přibývajícími položkami jsou informace rovnoměrně rozloženy mezi všechny uzly bez potřeby změny schématu navržené databáze.

Konzistence dat je zajištěna Master uzlem. Záznamy jsou z primární databáze distribuovány do sekundárních, které slouží jako záloha v případě zotavení se z chyb. V případě selhání Master uzlu je sekundární uzel zvolen jako hlavní, tato změna trvá řádově jednotky až desítky minut a tedy během této doby není databáze dostupná [1]. Navzdory výše zmíněné nevýhodě splňuje MongoDB všechny stanovené požadavky, a proto byla databáze vybrána jako prostředek pro vypracování zadaného projektu.

[1] Zdroj: `https://www.techtarget.com/searchdatamanagement/definition/MongoDB`

Část II

Návrh, implemetace a použití aplikace

Kapitola 4

Návrh aplikace

V této kapitole je proveden popis jednotlivých bloků řešení sekvenčně dle obrázku 2.1. K celkové implementaci je použit jazyk python3 a jeho knihovny.

Stažení dat: V souboru DOWNLOADER.PY je implementována třída DataDownloader s potřebnými metodami pro stažení obsahu ze serveru do lokální složky DOWNLOADED_DATA se zachováním adresářové hierarchie. Třída využívá pro stahování dat knihovnu BEAUTIFULSOUP[1], která je použita pro získání odkazů na jednotlivé archivy XML souborů z webových stránek. Odkazy jsou zpracovány s využítím lambda výrazu, který je aplikován na HTML značky pro hypertextové odkazy `<a>`, touto HTML filtrací jsou z webu získány pouze potřebné odkazy na XML soubory.
Pro vytvoření a zaslání http dotazu na webový server je využita knihovna REQUEST. Při připojení na server jsou také využity informace z předchozích relací, které jsou ukládány v podobě webových cookies za využití modulu HTTP.COOKIEJAR.

Parsování dat: Pro parsování získaných XML souborů je implementován modul MESSAGE_PARSER.PY, který v první řadě definuje strukturu parsovaných souborů (viz třídy CZPTTCISMESSAGE, PATHTIMETABLE, ...) a implementuje třídy CZPTTCISMESSAGEPARSER a CZCANCELEDPTTMESSAGEPARSER, které obě nabízí veřejné rozhraní (metodu `parse_from_element()`), pomocí kterého lze xml soubor převést do struktury objektů, která využívá slovníků a seznamů (záznamů a kolekcí). Výsledné objekty jsou již v podobě, která umožňuje jejich přímé uložení do MongoDB databáze.

Interakce s databází: Pro přímou interakci s databází slouží modul MONGO_COMMUNICATOR.PY. Ten je rozdělen na dvě logicky oddělené jednotky (implementované jako třídy MONGOUPLOADER a MONGODOWNLOADER), které se starají o nahrávání a získávání dat z MongoDB databáze. Způsob nahrání

[1] Dokumentace ke knihovně: `https://www.crummy.com/software/BeautifulSoup/bs4/doc/`

dat záleži na typu zprávy. Tedy jestli se jedná o CIS zprávu a vytvoření nového záznamu, nebo o Cancel zprávu a aktualizaci již vytvořeného záznamu.

- **Vytvoření nového záznamu:** Do databáze je vždy vkládan jeden záznam trasy společně (zároveň) se záznamy lokací, které trasa obsahuje.

  1. Při tvorbě nového záznamu do kolekce `path_time_tables` je postup velmi jednoduchý. Struktura, která je získáná z parseru je jednoduše nahrána do databáze funkcí `insert_one()`.
  2. Zajímavějším případem je nahrání záznamu do kolekce `locations`. U takových záznamů je časté, že záznam pro lokaci již v kolekci je. Proto se provádí operace upsert, kdy k vložení nového záznamu lokace dojde pouze pokud v kolekci ještě není.
  3. Pokud záznam lokace již v kolekci je, provede se aktualizace tohoto záznamu. Konkrétně se k seznamu ID tras (který udává, ve kterých trasách se lokace nachází) přidá ID nové trasy (pouze pokud tam již není) ke kterému se insert operace vztahuje.

- **Úprava již existujícícho záznamu:** Vždy pouze ruší jízdy vlaků v definovaných dnech již existujících tras.

  1. Z kolekce `path_time_tables` se získá záznam, jenž má být upraven.
  2. Aktualizuje se kalendář. Tedy dny které Cancel zpráva vypiná budou nastaveny na False. Zbytek zůstane nedotčen.
  3. Nová podoba aktualizovaného kalendáře se nahraje k danému záznamu do databazé pomocí funkce `update_one()`.
  4. Aktualizuje se čas aktualizace záznamu (na základě času uvedeného v Cancel zprávě).

- **Získání tras:** Při vytvoření požadavku na získání dostupných vlakových spojení na základě místa odjezdu, destinace a dne se provádí následující kroky:

  1. `query_candidate_ptt()`: Na základě klíče (názvu místa) se v kolekci `locations` vyhledá příslušný záznam. V tomto záznamu je uložen seznam ID tras, které obsahují dané místo.
  2. Vyhledání se provede jak pro místo odjezdu, tak pro destinaci.
  3. Získají se dva seznamy ID tras, z jejichž obsahu se provede průnik.
  4. Výsledný seznam obsahuje ID všech tras, které obsahují obě dvě vyhledaná místa.
  5. `query_ptt_ids()`: Z kolekce `path_time_tables` se získají všechny záznamy, jejichž ID se vyskytlo v seznamu vytvořeném v předchozím kroku.

6. Následně každý záznam prochází systémem filtrů, který vyřadí neplatné záznamy.
7. První filtr zkontroluje zda-li vlak v daný den jede. Kontrola se provádí na základě bool hodnoty v kalendáři pro daný den.
8. Druhý filtr kontroluje zda-li vlak jede správným směrem. Získá se index v poli lokalit pro obě lokality a porovná se jejich velikost. Index místa odjezdu musí být menší než index destinace.
9. Poslední filtr ověří, že vlak v daných stanicích zastaví. Tedy že aktivita vlaku ve stanici je označena kodem `0001`.
10. Vrací se seznam dokumentů tras, které splňují všechny požadavky. Dokumenty jsou dále zpracovány v klientskou aplikací do čitelné podoby.

Kapitola 5

Způsob použití

Zprovoznění řešení vyžaduje interpret jazyka python3 ve verzi 3.10 (nižší verze nejsou podporovány!) a zprovozněnou databázi MongoDB a connection string, pomocí kterého se k databázi bude aplikace připojovat. Pro účely projektu byla vytvořena databáze na službě Mongo Atlas, která poskytuje i zdarma řešení s omezenými prostředky. Connection string je poskytnut, doporučuje se však při nasazení aplikace u klienta vytvořit vlastní instanci databáze a dbát na vyšší zabezpečení.

Pro zprovoznění řešení je potřeba provést následující kroky:

1. Úprava connection stringu a názvu databáze v souboru `static/config.ini`
2. Spuštění scriptu `downloader.py` (bez parametrů) pro stažení dat z serveru železniční správy.
3. Spuštění scriptu `extractor.py` (bez parametrů) pro nahrání dat do databáze. Pozor, tento krok může být časově náročný.
4. Spuštění scriptu `query.py` pro získání výsledků vyhledání. Např. `./query.py -f "Karlovy Vary"-t "Olomouc hl. n."-d "2022-05-31"`

Kapitola 6

Experimenty

Výkonnost byla testována na neúplné databázi o 9000 záznamech, nad kterými byly provedeny dotazy. Průměrná doba provedení dotazu nepřekročila 2 vteřiny. Některé dotazy vrátily desítky relevantních JŘ, z nichž část obsahovala více než 150 různých lokalit. Usuzujeme z toho, že je program velice rychlý.

Například doba provedení dotazu na následující údaje byla 1.8s:
Od: Karlovy Vary
Kam: Olomouc hl. n.
Den: 2022-05-30

Ukázka dotazu je na obrázku 6.1 níže.

```
./query.py -f "Karlovy Vary" -t "Olomouc hl. n." -d "2022-05-31"
PA0054KT----29244A002022TR1154KASO--34441A012022
Stanice                     Příjezd                     Odjezd
Karlovy Vary                -                           05:52:00
Sedlec odbočka              05:53:30                    05:53:30
Karlovy Vary-Dvory          05:55:30                    05:55:30
Chodov zhlaví               05:59:00                    05:59:00
Nové Sedlo u Lokte          06:01:00                    06:01:00

------------------------------------
```

Obrázek 6.1: Ukázka části výsledku dotazu.