Ukládání rozsáhlých dat v NoSQL databázích: projekt do předmětu UPA (Ukládání a příprava dat)

Vladyslav Varfolomeiev, xvarfo00@stud.fit.vut.cz
Kristián Heřman, xherma33@stud.fit.vut.cz
Jiří Zbořil, xzbori21@stud.fit.vut.cz

zima 2022

Obsah

Část I

Analýza zdrojových dat a návrh jejich uložení v NoSQL databázi

Kapitola 1

Analýza zdrojových dat

Dostupnost: Pro zadanou finální úlohu v projektu, kterou bylo vyhledávat v připravené databázi spojení na dané trase v daný čas, bylo potřeba prozkoumat datové sady, dostupné na portálu *Celostátního Informačního Systému o jízdních řádech veřejné osobní dopravy* (dále jen CISJŘ). K této analýze bylo využito specifikace datových souborů pro CISJŘ dostupná na webu. V kořenové složce portálu je k dispozici hlavní balík (ve formátu `.zip`) s datovými sadami na celý rok, přesněji vždy na období od prosince do prosince. Kromě tohoto souboru jsou k dispozici doplňující datové sady, které jsou vydávány v průběhu ročního období, a proto jsou pro přehlednost uspořádány do jednotlivých podsložek podle měsíce a roku, v kterých byly vydány.

U hlavní sady (jmenovitě `GVD2022.zip`) je počítáno s aktualizací jednou ročně, vždy při vydání JŘ pro nové období. Naproti tomu datové sady v jednotlivých složkách jsou doplňovány průběžně podle předem plánovaných výluk, odklonů, ale mohou být také v některých případech aktualizovány během nenadálých situací pro nejbližší dny. Hlavní datová sada je komprimována do formátu ZIP, průběžně doplňující sady jsou jednotlivě komprimovány do formátu GZIP.

Charakteristika: Všechny datové sady jsou serializována do formátu `XML` a dělí se v zásadě na 2 typy zpráv. Ty jsou rozlišené názvem svého kořenového elementu. Zpráva typu `CZPTTCISMessage` obsahuje informace o daném spoji na dané trase s konkrétními údaji o časech v jednotlivých místech, kterými projíždí. Druhý typ zprávy `CZCanceledPTTMessage` pouze říká, který spoj pro které období bude odřeknut. Pokud se jedná o zprávu, pojednávající o výluce či odklonu, je tato zaslána jako zpráva typu `CZPTTCISMessage`, ovšem navíc obsahuje podelement `RelatedPlannedTransportIdentifiers`, kterým popisuje svůj vztah k již existujícímu spoji – tedy například, že k nějakému základnímu jízdnímu řádu vznikne výlukový JŘ s datem platnosti pouze v části roku.

Právě pro specifikaci platnosti jízdního řádu je ve zprávách obou typů zaveden element `PlannedCalendar`, uvnitř kterého je uvedena platnost tohoto JŘ. Ta je řečena ve formátu počátečního a koncového data platnosti společně s tzv. *BitmapDays*. Tato bitmapa je stejně dlouhá jako délka specifikovaného období

a říká, jestli se daného dne v období tata zpráva CISJŘ týká, nebo ne. Tedy například u zpráv o odřeknutých spojích znamená hodnota `1` v bitmapě, že spoj je odřeknut.

Pokud zpráva obsahuje informace o konkrétní trase spoje s časovými údaji v jednotlivých místech, jsou tyto informace strukturovány jako pole elementů `CZPTTLocation`, jenž může obsahovat kromě mezinárodně unikátního identifikátoru lokace, jejího české názvu, času příjezdu/odjezdu také element `TrainActivity`, který značí nějaký proces spoje v dané zastávce. Navíc pokud obsahuje subelement `TrainActivityType` s hodnotou `0001`, je tímto řečeno, že spoj v daném umístění a v daný čas zastavuje pro nástup a výstup cestujících, což je další ze zásadních hodnot pro následné vyhledávání.

Při ukládání dat do databáze bude potřeba brát do úvahy také datové typy některých položek. Většinu hodnot můžeme uchovávat jako obyčejné textové řetězce. V případech, kdy ale budeme při následném dotazování potřebovat hodnoty porovnávat mezi sebou, musíme je takto již v databázi uložit. Toto se týká údajů o datu, čase a případně některých číselných hodnot, které bychom mezi sebou chtěli porovnávat jako celá čísla. Datum je v původních datových sadách ukládáno podle normy ISO 8601, tedy `YYYY-MM-DDThh:mm:ssTZD`. Podobně časy jsou datového formátu `hh:mm:ss.mmmmmmm+UTC offset`.

Data lze třídit podle hodnot mnoha sloupců. Výhodné by ale pro další dotazování mohlo být seskupení podle některých časových úseků roku (pro zmenšení rozsahu prohledávání) nebo podle stanic, kterými jednotlivé spoje projíždí.

Se zdroji dat by se mělo zacházet velmi obezřetně, a to vzhledem jejich rozsáhlosti co se počtu souborů týče, tak vzhledem k celkové velikosti, která se může pohybovat až do řádu gigabajtů a není předem známá, protože záleží na aktuálních podmínkách výluk a odklonů.

Kapitola 2

Návrh způsobu uložení dat

Po prozkoumání datových sad a dat, která jsou v nich ukládána, lze přistoupit k návrhu uložení dat do databáze. V první řadě je na tomto místě vhodné zmínit, že bychom mohli uvažovat také o využití běžné SQL databáze, ale nejen kvůli zadání projektu bude využita databáze NoSQL. Jedním z faktorů je například variabilita schematu databáze, která se u dat CISJŘ částečně vyskytuje, což by mohlo práci s SQL databází komplikovat.

Zisk dat: Samotné získání dat je třeba realizovat tak, abychom na jeho konci dostali kompletní sadu datových souborů, které budeme moct za pomocí vhodné knihovny či software nahrát do NoSQL databáze. Všechny tyto soubory nám budou představovat jednotlivé objekty pro vložení do databáze.

Formát uložení dat: Vzhledem k rozsáhlosti vstupních dat bychom měli využít databázi, která nám umožní dobrou škálovatelnost databáze, současně rychlé vkládání do databáze. Toho lze dosáhnout tak, že budeme minimálně předzpracovávat data a měnit jejich strukturu. Právě naopak – pokud se zamyslíme nad formátem vstupních dat ve formátu XML, měli bychom této strukturovanosti do jednotlivých položek a podpoložek využít.

Můžeme uvažovat o ukládání v původních strukturách se zanořeními, ve kterých se nachází objekty či pole objektů stejného formátu. Při nahrávání dat do databáze ze zdroje by bylo vhodné vytvořit pomocné tabulky, které byly popsány v předchozí kapitole, jelikož by mohly při výsledném vyhledávání pomocí dotazování databáze výrazně zkrátit dobu dotazu. **Jednotlivé dokumenty nám budou v NoSQL databázi představovat jednotlivé datové soubory.**

Konkrétní struktury: V hlavní kolekci budeme uchovávat všechny dokumenty zpráv typu `CZPTTCISMessage`. V pomocných kolekcích pak budeme strukturu dokumentu vytvářet podle potřebných uchovávaných informací, což budou duplikáty některých hluboce zanořených elementů ve zprávě. Co se týče odřeknutých spojů, vytvoříme si další pomocnou tabulku, v které budeme moct uchovávat

odkaz na identifikátor konkrétního odřeknutého spoje společně s datovým období, v kterém je odřeknutí platné.

Kapitola 3

Zvolená NoSQL databáze

Druh databáze: Jak již bylo diskutováno v předchozí kapitole návrhu, pro naše zadání ukládání dat jízdních řádů ve formátu XML je nejvýhodnější ukládat dokumenty v té strukturované podobě, v jaké je máme.

Pro toto využití byla zvolena dokumentová NoSQL, která je schopna ukládat dokumenty ve formátu XML, JSON a podobně a následně v nich vyhledávat hodnoty pomocí klíčů jakožto posloupnosti jednotlivých vnořených elementů.

Databázový produkt: Jako konkrétní databázový produkt **byla vybrána NoSQL databáze MongoDB**. Byla vybrána z důvodů, mezi kterými lze zmínit například vynikající úroveň dokumentace, která je zapříčiněna velkou oblíbeností tohoto produktu, z čehož vychází také mnoho diskuzních fór i uživatelských tutoriálů, z kterých lze čerpat, protože autoři tento typ NoSQL databází využívají při tvorbě projektu poprvé.

Dalším důvodem využití je znalost databáze z přednášek předmětu UPA. Databáze probrané v UPA více do hloubky tak byly výchozím bodem při rozhodování.

Mezi dalšími důvody musíme zmínit také formát ukládání dat v MongoDB, který je velmi podobný formátu *JSON*, do kterého lze bez ztráty jakýchkoli dat převést naše data z formátu XML, tudíž takto můžeme počítat s převodem dat $1 : 1$. **Databázový produkt MongoDB byl také vybrán z důvodu, že je multiplatformní**. Autoři tak mohli s pohodlností pracovat současně na OS Linux i OS Windows a na obou těchto produkt MongoDB fungoval bez problémů.

Část II

Návrh, implemetace a použití aplikace

Kapitola 4

Návrh aplikace

Architekturní návrh: Aplikace byla navržena a rozdělena na 3 skripty v jazyce Python, kde:

- první skript `download.py` stáhne všechna data z portálu CISJŘ a rozbalí je ze ZIP/GZIP komprese tak, že na konci běhu skriptu jsou v předem nastavené složce k dispozici všechny datové soubory pro aktuální rok ve formátu XML;
- druhý skript `set_db.py` zpracuje všechna data v zadané složce, iteruje přes každý nalezený soubor, provede jeho parsing, případné konverze datových typů a nahraje výsledná data do NoSQL Mongo databáze;
- závěrečný skript `db_search_query.py` podle zadaných parametrů provede vyhledávání pomocí dotazování Mongo databáze. Pokud nalezne spojení vyhovující zadaným podmínkám, vrací jej na standardní výstup uživateli.

Použité technologie: Jak bylo řečeny, skripty byly napsány v jazyce Python společně s dalšími knihovnami. Mezi nejdůležitějšími pro správný běh aplikace uveďme tyto:

- `bs4 - BeautifulSoup` pro scraping odkazu na všechny podsložky s datovými sadami na portálu
- `requests` pro zasílání HTTP požadavků typu GET na server portálu CISJŘ, což nám vrací iterátor na zadaný soubor na serveru, pomocí kterého jej zapíšeme do vlastního souboru uloženého lokálně;
- `concurrent.futures` pro snazší paralelizaci stahování souborů – pokud bychom jednotlivé soubory stahovali sekvenčně, může se doba stahování celé datové sady zvýšit v závislosti na rychlosti internetu i několikanásobně;
- `pymongo` – knihovna pro práci s databází Mongo – pomocí modulu lze navázat připojení a dále se nad databází dotazovat;

- `xmltodict` – protože databáze Mongo pracuje s JSON formátem, který lze v Pythonu uchovávat v datovém typu `dictionary`, tedy stačí využít tuto knihovnu, která pro nás tento převod zajistí.

Stahovací skript: Skript umožňuje stáhnout všechna data, která jsou k dispozici na portálu CIJŘ pro aktuální letošní rok. Je možné zvolit přepínač a pouze vizualizovat počet souborů, které byly na serveru nalezeny ke stažení. Pokud je skript spuštěn ke stahování, nejprve si vytvoří seznam URL se všemi datasety, následně nad tímto seznamem spustí proces stahování.

Ten je paralelizován pomocí knihovny `concurrent.futures`, která zajistí, že je současně zasíláno více HTTP požadavků na server a tudíž celý proces stahování trvá kratší dobu. Současně knihovna řeší, že pokud je soubor úspěšně stažený, extrahuje jeho obsah do zadané lokální složky. Extrakce nejprve otestuje, jestli je souborový formát ZIP, potom použije dekompresi ZIP, v opačném případě využije GZIP dekompresi.

Skript pro přípravu databáze: Skript pracuje nad stejnou složkou, do které jsou staženy všechny XML datové soubory. Nad těmito následně iteruje a pro každý soubor provede zpracování.

Nejprve je vstupní XML otevřen pro čtení, převeden pomocí knihovny `xmltodict` na Python datový typ `dict`. S tím můžeme následně pracovat jako s „mnohodimenizonálním polem“ – například pokud chceme přistoupit k položce, která byla v původním poli v elementu `StartDateTime`, lze toto dosáhnout pomocí zápisu jednotlivých zanoření jako kdybychom přistupovali klíčem do pole, tedy `["CZCanceledPTTMessage"]["PlannedCalendar"]["ValidityPeriod"]["StartDateTime"]` (detaily viz kód).

```
<?xml version="1.0"?>
<CZCanceledPTTMessage xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  ...
  <CZPTTCancelation>2022-10-21T21:20:04</CZPTTCancelation>
  <PlannedCalendar>
    <BitmapDays>1</BitmapDays>
    <ValidityPeriod>
      <StartDateTime>2022-10-21T00:00:00</StartDateTime>
      <EndDateTime>2022-10-21T00:00:00</EndDateTime>
    </ValidityPeriod>
  </PlannedCalendar>
</CZCanceledPTTMessage>
```

Pokud se soubor týká odřeknutého spoje (toto bylo popsáno v předchozích kapitolách), jsou data souboru uložena do kolekce `canceled`, která udržuje všechny vazby na odřeknuté spoje. Pokud se soubor netýká odřeknutí, je v každém případě uložen do hlavní kolekce `main`, v níž jsou uchována všechna data týkající se celé trasy spoje. Nadto v případě, že soubor specifikuje odklon/výluku

(tedy obsahuje element `RelatedPlannedTransportIdentifiers`), je vytvořen záznam v kolekci `related`, která uchovává vazby identifikátorů původního spoje a spoje, který jej nahrazuje novým JŘ.

Dotazovací skript: Podle zadaných argumentů `odkud`, `kam` a `datum-cas` vypíše všechny validní spojení a k ním příslušný seznam zastávek s časem zastavení.

Nejprve se z databázové kolekce `locations` vyberou všechny dokumenty, které mají `location` rovno `odkud` a následně rovno `kam`. Průnik těchto dvou množin definuje pouze takové vlaky, které projíždí zadanými stanicemi, tedy i v nesprávném pořadí. Prvky výsledné množiny jsou případně nahrazeny náhradním vlakovým spojem z kolekce `related`. Logika nahrazení prvku je následovná. Jestliže má prvek nastavenou hodnotu `IsRelated` na `true` hledá se podle `PA ID` záznam a když má platný časový záznam pro zadané datum, je původní vlakový záznam nahrazený náhradním. Naopak pokud má nastavenou hodnotu `IsRelated` na `false` a jeho odkazovaný náhradní záznam není platný pro zadané datum, není nahrazeno. V opačném případě je takový záznam smazán.

Podle zadaného časového období jsou eliminovány vlakové spojení, které nejedou v konkrétní den. V případě, že vlaky sice danými stanicemi projíždí, ale v nesprávném pořadí, jsou ze seznamu odstraněny. Na závěr se kontroluje zda není výsledný vlakový spoj zrušen. Jednotlivé vlaky jsou nalezeny podle `TRAIN ID` v kolekci `canceled` a v případě, že platí pro daný den je záznam vymazán. Zbylé, nesmazané prvky jsou validní pro zadané parametry.

Kapitola 5

Způsob použití

Pokud některý skript nelze spustit, je vhodné zkontrolovat využívané knihovny zmíněné v předchozí kapitole, které lze v případě, že nejsou na počítači přítomny, stáhnout pomocí Python balíčkovacího systému `pip`.

Stažení souborů a upload do databáze: Pro správnou funkčnost aplikace je nejprve potřeba stáhnout a nahrát všechna data do databáze. To zajistíme spuštěním skriptů `download.py` a `set_db.py` v tomto pořadí. Ještě před spuštěním je však potřeba nastavit konfigurační soubor se složkami a adresou Mongo databáze. Konfiguraci nalezneme v souboru `config_upa.py`. V tomto souboru je potřeba nastavit hodnoty:

- `ZIPFOLDER` – do které složky budou uložena data ve skriptu `download.py`,
- `FOLDER` – ze které složky budou čteny XML soubory pro import do databáze (tato složka bude ve většině případech stejná jako hodnota `ZIPFOLDER`) a
- `MONGOSERVER` – adresa serveru, na kterém běží instance databáze Mongo.

Skript pro stažení lze spustit takto: `python download.py [-s]`. Přepínač `-s` je volitelný a udává, že chceme skript spustit pouze v režimu simulace – na standardní výstup jsou vypsány soubory, které by skript stáhl. Stahování však neproběhne.

Pokud máme v námi zadané složce stažené připravené XML soubory s jízdními řády a současně máme v konfiguračním souboru `config_upa.py` nastavenou správnou adresu serveru (proměnná `MONGOSERVER`), můžeme spustit skript pro nahrávání do databáze. Skript spustíme:
`python set_db.py`
Po spuštění nám několik minut poběží import dat do databáze, což bude skript indikovat průběžným výpisem stavu do konzole.

Dotazování: Skript má tři vstupní argumenty:

- `-c/-odkud` – Zastávka, kterou má vlakové spojení projíždět jako první

- `-d/-do` – Zastávka, kterou má vlakové spojení projíždět po první zastávce
- `-c/-cas` – Udává den, pro který vlakové spojení hledáme

Dle zadaných argumentů jsou vypsány vlakové spojení pro daný den. K jednotlivým vlakům jsou vypsány všechny zastávky, kterými bude vlak projíždět a současně odhadovaný příjezd.

Kapitola 6

Experimenty

Technické parametry:

- HW: 8 GB RAM, Intel Core i5g, disk 512 GB SSD, OS Windows
- Rychlost internetu cca 100 Mb/s
- Python, verze 3.10.7
- MongoDB 6.0.2

Jako první bylo testováno stahování všech datových sad z portálu CISJŘ. Bez nasazení paralelizace trvalo velmi dlouho – řádově desítky minut. Dokonce než bylo započato využití knihovny `requests`, byla tato doba stahování mnohem vyšší. Při přepracování kódu a nasazení paralelizace ale čas downloadu klesnul na cca 5 minut.

Následné nahrávání do databáze se časově pohybovalo kolem 15 minut, což vzhledem k celkové velikosti dat kolem 1GB a vytváření pomocných tabulek je asi přiměřená doba.

Doba získání dat z databáze dle zadaných parametrů průměrně trvala 1.2s uživatelského času a 0.1s systémového