Ukladanie a príprava dátových cestovných poriadkov

Bc. Michal Šedý

Bc. Martina Chrípková

Bc. Martin Novotný Mlinárcsik

21.10.2022

Obsah

Časť I

Analýza zdrojových dat a návrh jejich uložení v NoSQL databázi

Kapitola 1

Analýza zdrojových dat

Cieľom aplikácie je stiahnutie dátových cestovným poriadkov a ich následné spracovanie a uloženie. Aplikácia bude taktiež poskytovať možnosť vyhľadania spojení v ním špecifikovanom dni.

Dáta sú verejne distribuované pomocou webovej stránky `https://portal.cisjr.cz/pub/draha/celostatni/szdc/` a popis týchto dát je dostupný na `https://portal.cisjr.cz/pub/draha/celostatni/szdc/Popis%20DJ%C5%98_CIS_v1_09.pdf`.

Všetky dáta sú serializované vo formáte XML, ktoré sú na distribučnej webovej stránke uložené v osobitných adresároch podľa roku a mesiaca, kedy boli vytvorené. Jednotlivé XML dokumenty sa vytvárajú priebežne, podľa potrieb prevádzkovateľa železníc. Niektoré cestovné poriadky majú rozsah platnosti stanovený na jeden deň, iné majú svoj rozsah platnosti špecifikovaný na mesiac. Cestovné poriadky s väčším rozsahom platnosti majú navyše dodatočne špecifikované, v ktoré dni je vlak vypravený a v ktoré nie. epar XML záznamy môžu byť dvoch typov:

1. Prvým typom záznamu sú záznamy špecifikujúce vlastnosti cestovných poriadkov pre vlaky premávajúce na nejakej trase. Konkrétny vlak premávajúci po trase je špecifikovaný unikátnou kombináciou identifikátoru vlaku a identifikátoru dráhy. Vlastnosti tohto cestovného poriadku sú, napríklad, rozsah jeho platnosti alebo zoznam zastávok na trase vlaku.

2. Druhým typom záznamu je záznam typu **cancel**, ktorý upravuje rozsah platnosti prvého záznamu. Záznam špecifikuje dni, kedy daný vlak na danej trase nebude vypravený.

Názov prvého typu záznamu obsahuje názov trasy, po ktorej daný vlak premáva. Dôležitými štrukturálnymi súčasťami záznamu prvého typu sú:

1. Element **Identifiers**, obsahujúci identifikátor trasy **(PA)** a identifikátor vlaku **(TR)**. Táto kombinácia identifikátorov je vždy unikátna.

2. Element **CZPTTCreation**, špecifikujúci dátum vytvorenia dokumentu.

3. Element **CZPTTInformation**, pričom primárnym zdrojom informácií je subelement **CZPTTLocation**, ktorý obsahuje informácie o zastávkach vlaku na danej trase, ako napríklad názov zastávky, časy príchodu a odchodu alebo typ úkonu vykonávaného na konkrétnej stanici. Dôležitým subelementom je taktiež **PlannedCalendar**, ktorý obsahuje rozsah platnosti cestovného poriadku pre danú trasu a taktiež špecifikuje dni, v ktoré je vlak na danej trase vypravený a v ktoré nie.

   Ak sa v elemente **CZPTTLocation** nachádza subelement **TimingAtLocation.Offset**, bude tento element upresňovať počet polnocí, ktoré vlak na svojej trase prekonal.

4. Element **RelatedPlannedTransportIdentifiers** určuje vlakové spojenie, ktoré daný záznam nahrádza. V takom prípade by mal byť pôvodný spoj zrušený správou **CZCanceledPTTMessage**. V istých prípadoch avšak táto správa chýba, čo je v rozpore s dokumentáciou poskytnutou CISJR. V nájdení týchto inkonzistencií je správa CZCanceledPTTMessage vygenerovaná aplikáciou.

Tieto súčasti sú dôležité z pohľadu tvorby databázy a ich úloha bude popísaná v ďalších kapitolách dokumentácie. Všetky vyššie spomenuté súčasti musia byť povinne vyplnené, keďže ide o kritické informácie o trase. Iné, pre potreby aplikácie a tvorby databázy menej dôležité elementy jednotlivých dokumentov, je možné vyhľadať v oficiálnej dokumentácií popisu dátových cestovných poriadkov.

Kapitola 2

Návrh způsobu uložení dat

Zo zadania projektu je známe, že užívatelia sa na trasy budú dotazovať pomocou názvov počiatočnej a koncovej stanice v zvolený deň. Z tohto dôvodu sú dôležité elementy **CZPTTLocation**, ktoré obsahujú zoznam staníc pre vlak na danej trase a elementy **PlannedCalendar**, ktoré špecifikujú, ktoré dni je vlak vypravený. Keďže zoznam lokácií, ktorými bude vlak prechádzať, je zoradený podĺa poradia staníc, v akom nimi bude vlak prechádzať, je pomerne jednoduché v databáze nájsť dokumenty, ktoré budú obsahovať počiatočnú a koncovú stanicu v užívateľom špecifikovanom poradí. Zároveň nám subelementy **BitmapDays** a **ValidityPeriod** elementu **PlannedCalendar** umožňujú vyhľadat iba tie spoje, ktoré budú premávať v užívateľom specifikovanom dni. Konkrétne subelement **BitmapDays** je transformovaný do podoby dvoch polí:

1. Pole **Valid** obsahuje dátumy uložené vo formáte YYYYMMDD, ktoré určujú, ktoré dni je vlak vypravený, tj. kedy **BitmapDays** obsahujú 1.

2. Pole **Cancel** obsahuje dátumy vo formáte YYYYMMDD, ktoré určujú dni, v ktoré bola výprava vlaku zrušená správou CZCanceledPTTMessage. Prienik množín Valid a Cancel je prázdny. V konkrétnom prípade, kedy by existovala správa CZCanceledPTTMessage, ktorá by upravovala zmeny vytvorené inou, staršou správou rovnakého typu CZCanceledPTTMessage, pole Cancel umožňuje jednoducho prepísať zmeny staršej správy správou novšou.

Jednotlivé dokumenty v kolekciách sú vytvorené podľa jedného dátového cestovného poriadku a jednotlivé časti dokumentu sa budú zhodovať s elementmi pôvodných XML dokumentov. Tu bude využitá vlastnosť NoSQL databáz, ktorá nám umožňuje vytvárať heterogénne dátové štruktúry, keďže často nemajú jednotlivé dátové cestovné poriadky rovnaký počet elementov **CZPTTLocation**, alebo niektoré neobsahujú elementy **NetworkSpecificParameters**.

Pri počiatočnom spracovaní a nahrávaní dát do databázy sú všetky XML súbory spracované a ich obsah nahratý do kolekcie **CZPTTCISMessage**. Táto kolekcia je teda primárnou kolekciou, ktorá obsahuje väčšinu dôležitých dát pre dotazovanie a vyhľadávanie trás.

Ako pomocné kolekcie sú vytvorené:

1. **DBInfo**, ktorá obsahuje dokumenty o poslednej aktualizácií databázy.
2. **Location**, ktorá obsahuje dokumenty s informáciami o jednotlivých lokáciách, ktoré sa vyskytujú v elementoch **CZPTTLocation**. Táto kolekcia pomáha znížiť pamäťovú náročnosť znížením duplicitných výskytov popisu každej lokácie v každom dokumente. Namiesto toho sa môže databáza odkazovať na túto kolekciu.
3. **Downloaded** obsahuje adresy na jednotlivé stiahnuté XML dokumenty. Pri aktualizácií databázy teda aplikácií stačí porovnať, ktoré XML súbory už boli stiahnuté a stiahnuť iba nové. V prípade použitia prepínača –**force** v klientskej aplikácií je možné vynútiť opätovné stiahnutie záznamov.

Vrámci predspracovania dát a zníženie pamäťovej náročnosti sa záznamy typu **cancel** neukladajú, ale pri ich spracovávaní sú z nich extrahované kľúčové informácie o trase, ktorej rozsah platnosti upravujú. Tieto identifikátory sú porovnané s identifikátormi v kolekcii **CZPTTCISMessage** a pri zhode budú dokumenty v tejto kolekcií príslušne upravené, aby sa pri vyhľadávaniach užívateľom nezobrazovali. Keďže sa vo fáze iniciálneho vytvorenia a nahratia databázy bude prehľadávať celý dátový priestor môže toto predspracovanie trvať istý čas, avšak ušetrí čas pri neskôršom vyhľadávaní, keďže nebude potrebné opakovane kontrolovať, či pre vyhľadanú trasu existuje záznam typu **cancel**. Avšak aj toto prehľadávanie dátového priestoru môže byť odľahčené za použitia indexovania podľa subelementov elementu **Identifiers**.

Kapitola 3

Zvolená NoSQL databáze

Ako druh NoSQL databázy bola zvolená dokumentová databáza, konkrétne multiplatformná open-source dokumentová databáza **MongoDB**. Kedže naše vstupné dáta sú tvorené XML súbormi s popismi jednotlivých trás, môžeme vďaka dokumentovej štruktúre MongoDB zachovať túto podobu a XML súbory previesť na jednotlivé dokumenty, ktoré budú vkladané do kolekcie.

MongoDB nám ako dokumentová databáza umožňuje vytvárať vnorené dokumenty a tak uchovávať všetky dôležité informácie o trasách v jednom dokumente. Takto sú dáta obsiahnuté v jednej ucelenej jednotke, pričom sa k týmto dátam môžeme jednoducho dostať.

MongoDB poskytuje využitie tzv. **agregačnej pipeline**, ktorá umožňuje efektívne dotazovanie. Táto pipeline môže byť zostavená z viacerých krokov/úrovní, ktoré budú postupne redukovať množstvo dokumentov, nad ktorými sa bude aplikácia dotazovať. Správnou voľbou MongoDB príkazov vo vyhľadávacej pipeline môžeme značne zlepšiť časy jednotlivých vyhľadávaní, keďže sa výsledky jednotlivých úrovní berú ako vstup úrovní následujúcich. Túto úlohu čiastočne za programátora preberá aj samotné MongoDB, ktoré je schopné poradie príkazov v agregačnej pipeline samo optimalizovať tak, aby bol výsledný dotaz efektívnejší.

V prípade vyhľadávania trás môžeme efektívne eliminovať trasy, ktoré nebudú obsahovať jednu z užívateľom zadaných staníc a znížiť počet porovnávaní v následujúcich úrovniach pipeline. V následujúcom kroku eliminujeme všetky spoje, ktoré nemajú hodnotu DATE v elemente VALID. Vďaka predchádzajúcim úpravám - aplikáciám správ typu Cancel - už nie je potrebné kontrolovať, či bol vlak zrušený.

Časť II

Návrh, implemetace a použití aplikace

Kapitola 4

Návrh aplikace

Aplikácia je tvorená kolekciou skriptov napísaných v jazyku **Python**. Databáza je uložená lokálne a jej prepojenie s aplikáciou zabezpečuje softvér **Docker**. Skripty napísane v jazyku Python využívajú množstvo externých knižníc pre spracovanie a nahratie dát do databázy, najmä knižnice **bs4**, **shutil**, **tempfile**, **gzip**, **zipfile** a knižnicu **xml.etree.ElementTree** pre prácu s XML dokumentmi.

Python skripty využívané pre prácu s dátami a databázou:

1. **fill_db.py** riadi spúštanie následných modulov, ktoré sa starajú o jednotlivé úkony pri príprave databázy.

2. **downloader.py** na začiatku stiahne všetky URL adresy jednotlivých XML súborov a uloží ich to zoznamu adries. Z týchto adries sú následne stiahnuté všetky súbory zabalené vo formáte **zip** alebo **gzip**. Tieto archívy sú uložené do adresáru špecifikovaného užívateľom a v prípade nešpecifikácie je vytvorený dočasný adresár, ktorý sa po vykonaní skriptu vymaže. Skript downloader.py využíva knižnice **pqdm** pre možnosť paralelného sťahovania archívov, čím znižuje časovú náročnosť aplikácie. V prípade aktualizácie databázy skript stiahne iba nové XML súbory, pričom ich následne pridá do databázy. Opätovné stiahnutie záznamov je možné s využitím prepínača –**force**.

3. **parser.py** spracováva jednotlivé XML dokumenty. Skript postupne prechádza elementmi XML dokumentu a ich obsah prevádza to dátovej štruktúry typu slovník. Po spracovaní celého dokumentu je novovytvorený slovník pridaný do jedného zo zoznamov slovníkov, ktoré sú troch typov:

   (a) **CZPTTCISMessage**

   (b) **CZCanceledPTTMessage**

   (c) **Locations**

   Skript parser.py taktiež využíva knižnicu **pqdm** pre paralelné spracovanie XML súborov.

4. **uploader.py** zoznamy slovníkov vytvorené skriptom parser.py vloží ich do databázy. Pri vkladaní dát do databázy sú využívané funkcie knižnice **pymongo** pre prácu s MongoDB v jazyku Python. Tieto príkazy sú využívané pre vkladanie dát do správnych kolekcií a pri overovaní duplicitných dokumentov. Pokiaľ dôjde ku kolízií jednoznačných identifikátorov (Identifikátory v elemente RelatedTrainIdentifiers) je ponechaný iba najaktuálnejší záznam.

5. **database_api.py** poskytuje API pre prácu s databázou skrz knižnicu **pymongo** a zjednodušuje volania funkcií tejto knižnice.

Kapitola 5

Spôsob použitia

Inštalácia na závislosti a vytvorení kontajneru

Aplikácia cestovných poriadkov je implementovaná v jazyku Python (3.10+) s využitím NoSQL databázy MongoDB (6.0). Databáza je nasadená v Docker (20.10.18) kontajneri. Všetky príkazy z tejto sekcie sú vykonávané v koreňovej zložke projektu.

Inštalácia podporných balíkov:

```
python3 -m venv upa-env
source ./upa-env/bin/activate
pip3 install -r requirements.txt
```

Spustenie kontajneru:

```
docker run -d -p 27017:27017 --name upa_container upa_image
```

Mongo databáza je po spustení kontajneru prístupná z:

```
mongodb://localhost:27017
```

Užívateľské rozhranie

Skripty vykonávajúce prevod dát z cestovných poriadkov do Mongo databázy (**fill_db.py**) a vyhľadávanie vlakových spojov (**find.py**) sa nachádzajú v zložke src. Všetky spustiteľné skripty obsahujú nápovedy.

fill_db.py

Stiahne archívy cestovných poriadkov z repozitára poskytovaného CISJR a následne ho extrahuje, spracuje a nahrá do databázy.

```
fill_db.py [-h] [--parallel-download N_THR]
           [--parallel-parse N_THR]
           [-f] [workdir]
```

Pričom jednotlivé prepínače znamenajú:
`-h --help` - Nápoveda.

`--parallel-download N_THR` - Súbory je možné z CISJR repozitára stahovať paralelne s použitím $0 <$ N_THR vlákien. Pri paralelnom stahovaní je nutné uviesť pracovný adresár (workdir), kam sa budú ukladať stiahnuté súbory.

`--parallel-parse N_THR` - Dáta uložené v XML súbore je možné taktiež spracovávať paralelným spôsobom s využitím $0 <$ N_THR vlákien.

`-f --force` - Budú stiahnuté všetky dáta z CISJR a to bez ohľadu na obsah kolekcie Downloaded v databázi.

`workdir` - Pracovná zložka, do ktorej sa budú ukladať stiahnuté a extrahované archívy. Pokiaľ nie je uvedené, bude vytvorená dočasná zložka v /tmp, ktorá bude po dokončení odstránená. Pri použití –parallel-download musí byť zložka workdir špecifikovaná.

printer.py

Jednoduchý vizualizér na formátovanie výsledkov. Vypisuje nájdené spoje na príkazový riadok a generuje HTML súbor `results.html`, ktorý zobrazuje výsledky prehľadne.

find.py

Nájde spoje medzi zadanými stanicami v daný dátum a čas. Následne ich pomocou jednoduchého vizualizéru 5 zobrazí. Mená staníc obsahujúce biele znaky musia byť uzatvorené do "úvodzoviek". Formát dátumu a času je yyyy-ddTHH:MM:SS.

Použitie: `find.py FROM TO DATE_TIME`

Pričom:

`FROM` - Meno počiatočnej stanice.

`TO` - Meno cieľovej stanice.

`DATE_TIME` - Dátom a čas odjazdu vo formáte yyyy-mm-ddTHH:MM:SS

drop_db.py

Zmaže Mongo databázu.

Použitie: `drop_db.py`

Kapitola 6

Experimenty

Experimenty boli vykonané na systéme s následovnými technickými parametrami:

- CPU: AMD Ryzen 7 5800H (8-jadier|16-vlákien)
- RAM: 16GB
- Rychlosť internetového pripojenia: ping 1ms; upload 1000Gb; download 1000Gb

1. Rýchlosť sťahovania v závislosti na počte použitých paralelných vlákien:

THR	TIME
1	634 sec
2	288 sec
4	146 sec
8	64 sec
16	33 sec
32	22 sec
64	20 sec

2. Rýchlosť parsovania XML dokumentov v súbore v závislosti na počte použitých paralelných vlákien:

THR	TIME
1	167 sec
2	57 sec
4	41 sec
6	36 sec
8	36 sec
16	38 sec
32	40 sec

3. Rýchlosť nahrávania (upsertovania) dát do databázy v závislosti na nastavení/nenastavení indexácie podľa subelementu Identifiers v elemente PlannedTransportIdentifiers:

IDX	TIME
ON	106 sec
OFF	1345 sec

4. Rýchlosť vykonania dotazu na hľadaný spoj:

```
Ostrava hl. n. -> Ostrava-Svinov v 2022-10-18T00:00:00: 0.57 sec
```