Ukládání rozsáhlých dat v NoSQL databázích

František Koleček, xkolec08@stud.fit.vut.cz
Tomáš Moravčík, xmorav41@stud.fit.vut.cz
David Sladký, xsladk07@stud.fit.vut.cz

zima 2022

Obsah

Část I

Analýza zdrojových dat a návrh jejich uložení v NoSQL databázi

Kapitola 1

Analýza zdrojových dat

Použitá datová sada se nachází na stránkách ministerstva dopravy a to konrétně zde. Detailní popis formátu dokumentů datové sady lze najít na stejném místě.

Datová sada se skládá z `XML` souborů. Tyto soubory jsou zveřejňovaný na začátku roku pro celý rok ve složce `GVD`. Dále je pak každý měsíc zveřejňována datová sada pro daný měsíc s aktualizacemi pro spoje. Každý soubor má element `CZPTTCreation`, který určuje jeho vytvoření.

`XML` soubory lze rozdělit do následujících tří skupin:

- Definující spoj
- Rušící spoj
- Definující náhradní spoj

Soubory definující spoje mají jako kořenový element `CZPTTCISMEssage`. První důležité informace se nachází v elementu `Identifiers`, kde jsou uvedeny identifikátory pro definované spojení a vlak, který ho bude provádět. Dále element `CZPTTHeader` určuje, zda spoj přijíždí nebo pokračuje z/do zahraniční stanice. Elementy `CZPTTLocation` obsahují jednotlivé stanice, kterýma vlak projíždí. Zde jsou uvedeny i další informace ke stanici. Nejvýznamnější z nich jsou: čas příjezdu/odjezdu, typ aktivity. Po uvedení všech stanic následuje element `PlannedCalendar`, který určuje výčet dní, kdy je tento spoj prováděň.

Soubory rušící spoje mají kořenový element `CZCanceledPTTMessage`. Podobně jako soubory definující spoje obsahují identifikaci spoje, který se ruší a výčet dní, kdy se ruší. Dále už nenesou žádné informace.

Soubory definující náhradní spoje mají stejnou strukturu jako soubory definující spoje jenom s jediným rozdíle. Obsahují element `RelatedPlannedTransportIdentifiers`, který určuje, jaký spoj nahrazují. Tyto spoje mají unikátní indentifikátor vůči normálním spojům.

Kapitola 2

Návrh způsobu uložení dat

Z datové sady poskytnuté XML soubory lze definovat několik entit a vazeb mezi nimi vhodných pro uložení do NoSQL databáze.

2.1 File

Samotné soubory lze považovat za entity pro uložení. Tyto entity ponesou informace o jméně souboru, který reprezentují a času, kdy byl soubor vytvořeny z elementu CZPTTCreation.

:File
filename creation

2.1.1 PRECEDES

Novější soubory aktualizují stav definovaný předcházejícími soubory, proto mezi jednotlivými entitami :File budou vazby :PRECEDES, které spojí soubor s jeho nejbližším následníkem.

:File – :PRECEDES –> :File

2.1.2 DEFINES

Pokud se nejedná o soubor rušící spoj, tak tento soubor definuje o jakém spoji podává informace. Tuto vlastnost bude modelována vazbou :DEFINES.

:File – :DEFINES –> :PA

2.1.3 CANCELS

Pokud se jedná o soubor rušící spoj, tak tento soubor definuje jakého spoje ruší jízdy. Tuto vlastnost bude modelována vazbou :CANCELS.

:File – :CANCELS –> :PA

2.2 PA

Další významnou entitou jsou spoje. Tyto spoje budou označovány jako PA – stejně jako v souborech v elementu Object Type.

:PA
Core
Company
Variant
TimetableYear
NetworkSpecificParameters
Header

2.2.1 SERVER BY

Každý spoje je prováděn vlakem. Tento vlak je vždy uveden v souboru se spojem, který provádí. Tuto vlastnost definujeme jako :SERVED BY.

:PA – :SERVED BY –> :TR

2.2.2 RELATED

Pokud se jedná o spoj, který nahrazuje jiný spoj, tak je také definováno jaký spoj je nahrazován. Tuto vlastnost modelujeme jako :RELATED.

:PA – :RELATED –> :PA

2.2.3 GOES IN

U každého spoje je také uvedeno, v jaké dny jezdí. Informaci v jaké dny jeden reprezentujeme vazbou :GOES IN mezi spojem a dnem.

:PA – :GOES IN –> :Day

2.2.4 IS IN

Každý spoj projíždí minimálně dvěmi stanicemi. Tuto vlastnost modelujeme vztahem :IS IN mezi spojem a stanicí. Tato vazbna také neve řadu parametrů definující příjez, odjezd, prováděné akce a podobné.

:PA – :IS IN –> :Station

:IS IN
LocationSubsidiaryCode
AllocationCompany
LocationSubsidiaryName
ALA
ALAOffest
ALD
ALDOffset
DwellTime
ResponsibleRU
ResponsibleIM
TrainType
TrafficType
CommercialTrafficType
TrainAvtivityType
OperationalTrainNumber
NetworkSpecificParameters

2.3 TR

Soubory jsou také definovány vlaky, které provádí spoje. Tyto vlaky budou znázorněny entitami :TR.

:TR
Core
Company
Variant
TimetableYear
NetworkSpecificParameters

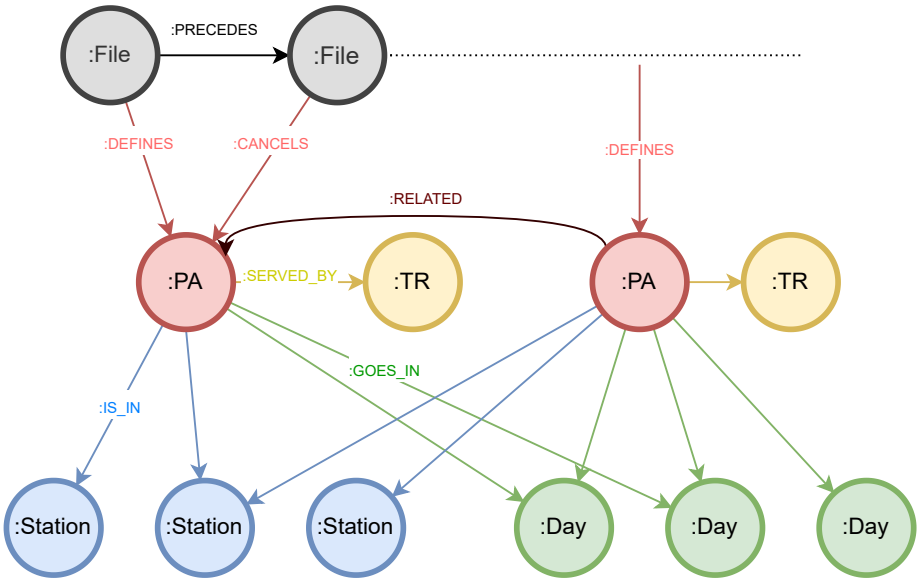
2.4 Station

V XML souborech jsou také definováné stanice, kterými vlak projíždí. Tyto stanice budou modelovány pomocí entit :Station.

:Station
LocationPrimaryNumber
PrimaryLocationName
CountryCodeISO

2.5 Day

U každého spoje je také určeno, v jaké dny jezdí. Tyto dny budou existovat jeko entity :Day.



Obrázek 2.1: Finální schéma entit a vazeb meiz nimi v databázi.

:Day
Date

Kapitola 3

Zvolená NoSQL databáze

Pro řešení tohoto problému jsme zvolili grafovou databázi, konkrétně Neo4j. Grafová databáze nám umožní reprezentovat entity jako uzly v grafu a vztahu mezi entitaby jako cesty mezi těmito uzly. Zároveň je možné u uzlů i cest definovat další vlastnosti typu klíč:hodnota, co zase umožní uložit veškeré další potřebné informace vztahující se k jednotlivým objektům. Neo4j poskytuje webové rozhraní pro databázi, které umožňuje jednoduché zadávání požadavků na databázi. S kombinací s tím, že toto webové rozhraní také zobrazuje získané výsledky do grafu, tak nám to umožní snadnější opravování chyb v aplikaci a snadnější vytváření potřebných požadavků na databázi.

Část II

Návrh, implemetace a použití aplikace

Kapitola 4

Návrh aplikace

4.1 Zpracování souborů XML

Informace o vlakových spojích jsou získávány ze souborů ve formátu XML. Tyto soubory je třeba zpracovat – nahrát data do strukturované vnitřní reprezentace programu, pro efektivní nahrávání do databáze. Každý soubor obsahuje informace o jednom konkrétním vlakovém spoji, jeho cesta a časy ve stanicích jsou vždy stejné, jsou zde definované dny, ve kterých tento spoj jede. Nejdůležitější zpracovávaná data jsou:

- Identifikátory
- Navštívené stanice a časy příjezdu a odjezdu
- Dny, ve kterých spoj jede

4.1.1 Struktura XML souborů

Každá s těchto informací se nachází ve vlastní „větvi“ v souboru. Nachází se u nich samozřejmě i dodatečné informace – detaily lokace, činnost vlaku ve stanici atd. Platné dny jsou určeny pomocí dvou atributů – začátek a konec platnosti a bitmapa. Bitmapou je myšlen řetězec jedniček a nul, kde jedničky vyjadřují platnost v jednotlivých dnech. Tímto způsobem lze vyjádřit platné dny na rok dopředu pomocí řetězce dlouhém 365 znaků.

4.1.2 Zvolené Prostředky

Zpracovávání souborů je implementováno v jazyce Python v souboru `parser.py`. Je využito modulu `xml`, který je součástí základní instalace Pythonu, není třeba jej dodatečně instalovat. Tento modul obsahuje třídu `ElementTree`, která umožňuje snadné nahrávání dat ze stromové struktury souboru. Velmi užitečnou funkcí je možnost adresování uzlů pomocí „cesty“ - obdobným způsobem jako adresování souborů v souborovém systému.

4.1.3 Způsob implementace

Nejdůležitější roli při zpracovávání hraje funkce `node_to_dict`, která rekurzivně prohledává daný uzel a převádí jeho obsah na slovníkový datový typ.

Protože platné dny jsou v naší databázi ukládány jako vlastní uzly, na které pak odkazují spoje, které jsou platné v daný den, je potřeba původní reprezentaci platných dní (popsána výše) převést na seznam konkrétních kalendářních dat. Pro implementaci této funkcionality byl využit modul `datetime`, který je opět součástí základní instalace pythonu. Tento modul mimo jiné umožňuje vykonávat „aritmetické operace“ s kalendářními daty. V našem případě se jedná o sečtení data začátku platnosti a indexu jedničky v příslušné bitmapě. Implementace je ve funkci `cal_to_listofdays`.

Kapitola 5

Způsob použití

5.1 Neo4j

Pro spuštění je potřeba mít přístup k Neo4j databázi. Toho se dá dosáhnout několika způsoby, zde však budou uvedeny jenom dva nejpřímočařejší.

5.1.1 Databáze na cloudu

Pro tuto variantu je potřeba si založit účet na stránkách Neo4j -> Get Started -> Start Free -> Sign up. Po vytvoření účtu je možné se dostat ke správci cloudových databází obdobným způsobem opět z Neo4j -> Get Started -> Start Free -> Přihlášení. Po přihlášení lze vytvořit novou databázi pomocí New Instance. Nezapomeňte si poznamenat přihlašovací údaje do databáze.

5.1.2 Lokální databáze

Pro lokální databázi je třeba stáhnout Neo4j Community. Po rozbalení staženého archivu zadejte následující příkaz z kořenového adresáře: `./bin/neo4j console`. Databáze bude poskytovat webové rozhraní na http://localhost:7474/. Výchozí login i heslo jsou `neo4j`.

5.2 Spuštění

```
python3 app.py login heslo uri
```

- `login` - přihlašovací jméno k databázi
- `heslo` - heslo k databázi
- `uri` - uri databáze

Po spuštení programu se bude program dotazovat na několik informací:

1. `Do you want to download data? Y/N` - Tato volba způsobí stažení zazipovaných xml do složky zips a jejich následovné rozbalení do složky `xml_data`. Pokud nechcete stahovat data, umístěte vlastní xml soubory do `xml_data`.
2. `Do you want to load local data? Y/N` - Projde soubory v `xml_data` a nahraje jejich grafovou reprezentaci do databáze.
3. `Please put starting station` - Zadejte výchozí stanici.
4. `Please put terminal station` - Zadejte konečnou stanici.
5. `Please put since date` - Zadejte čas od kdy má program vyhledávat ve formátu `YYYY-MM-DDTHH:MM:SS`. `T` je oddělovač data a času.

Po zadání všech informací program vyhledá nejbližší přímí spoj, pokud takový existuje, mezi stanicemi. Posléze budete dotázání, zda chcete provést další vyhledávání.

Příklad výstupu programu.

```
user@pc:/path/to/root/folder$ python3 app.py neo4j neo4j bolt://localhost:7687
Do you want to download data? Y/N
n
Do you want to load local data? Y/N
n
Please put starting station:
Polanka nad Odrou
Please put terminal station:
Sedlnice
Please put since date:
2022-07-13T9:00:00
----------------------------------------
Polanka nad Odrou -- 09:43:00
Jistebník -- 09:46:00
Studénka -- 09:51:00
Sedlnice-Bartošovice -- 09:56:00
Sedlnice -- 09:57:00
It took 0.004521846771240234 s to complete this query.
----------------------------------------
Do you wish to continue?
n
```

Kapitola 6

Experimenty

Cíl: Změřit, jak aplikace a databáze fungují v praxi.

Obsah: Popis výchozí konfigurace aplikace a nasazení databáze stroje, kde budou experimenty probíhat (HW a SW). Popis experimentů typicky představující nahrání dat ze zdroje do databáze či dotazy ze zedání s výslednými časy jejich provedení. Případné poznámky k výsledkům experimentů.

Prostředky: Strukturvaný text, případně tabulka či graf s doprovodným textem.

Fáze projektu: Testování řešení před předáním výsledného systému zákazníkovi.

6.1 Rychlost zpracování XML souborů

Funkčnost a efektivita skriptu `parser.py` byla testována nad složkou `GVD2022`, která obsahuje přibližně 12300 souborů formátu XML. Všechny soubory byly v experimentu zpracovány a výpis důležitých informací vytisknut na standardní výstup. Tato operace trvala 93 sekund, tedy cca 132 zpracovaných souborů za sekundu.