Ukládání a příprava dat

2022/2023

Ukládání rozsáhlých dat v NoSQL databázích

Kulíšek Vojtěch(xkulis03)
Plevač Lukáš(xpleva07)
Šesták Pavel(xsesta07)

Brno, 25. října 2022

Obsah

1 Analýza zdrojových dat a návrh jejich uložení v NoSQL databázi **2**
1.1 Analýza zdrojových dat . . . 2
1.2 Návrh způsobu uložení dat . . . 2
1.3 Zvolená NoSQL databáze . . . 2

2 Návrh, implementace a použití aplikace **3**
2.1 Návrh aplikace . . . 3
2.1.1 DataUpdater . . . 3
2.1.2 FileDownloader . . . 3
2.1.3 FileParser . . . 3
2.1.4 MongoClient . . . 3
2.1.5 Provedené optimalizace . . . 4
2.1.6 Dotaz na existující vlakové spojení . . . 4
2.2 Způsob použití . . . 5
2.3 Experimenty . . . 5

A Časy 50ti běhů s výpočtem průniků pomocí agregační pipeliny **6**

B Časy 50ti běhů s výpočtem průniků v paměti **6**

C Časy 10ti běhů s dotazem do databáze pro každý vlak zastávky **6**

D Časy 10ti běhů s před stažením všech záznamů vlaků v průniku **6**

E Použitá pipeline pro MongoDb **7**

Seznam obrázků

1 Diagram kolekcí pro uložení do NoSQL databáze . . . 2
2 Diagram architektury aplikace . . . 3
3 Ukázka kolekcí s velikostmi a počtem záznamů . . . 5

1 Analýza zdrojových dat a návrh jejich uložení v NoSQL databázi

V této části textu se zaměříme na vstupní data, jejich interpretaci a navrhneme pomocné struktury v rámci NoSQL databáze, které nám umožní rychlejší dotazování nad vstupními daty.

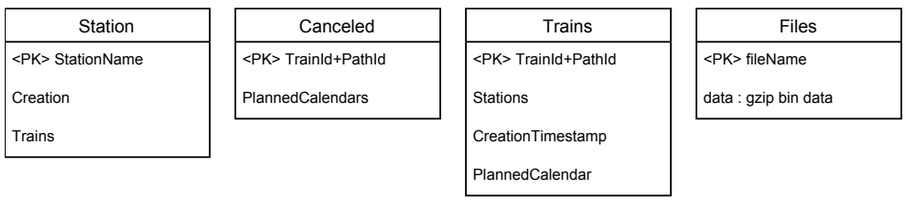
1.1 Analýza zdrojových dat

Zdrojová data jsou dostupná na adrese: `https://portal.cisjr.cz/pub/draha/celostatni/szdc`. Data jsou členěná podle roku a měsíce do podsložek. Každá složka s rokem obsahuje archiv s prvotním plánem spojů a následně složky pro jednotlivé měsíce obsahují soubory se změnami v jízdním řádu. Ke změně jízdního řádu může dojít v případě zrušení či odklonění daného spoje.

Jednotlivé soubory obsahují data ke spojům ve formátu xml a jsou komprimované pomocí gzip. Každý soubor obsahuje zprávu s informacemi o jednom konkrétním spoji. Každý spoj je jednoznačně identifikován pomocí identifikátoru cesty a vlaku. Zpráva dále obsahuje informace o jednotlivých zastávkách. Zda vlak v dané zastávce skutečně zastavuje je uvedeno v atributu TrainActivityType. Na konci zprávy je informace o časovém intervalu, pro který je tato zpráva platná.

1.2 Návrh způsobu uložení dat

V rámci aplikace budeme chtít ve stažených datech vyhledávat spoj mezi dvěma zastávkami v konkrétní čas. Datové soubory jsou ve formátu, že pro každý vlak definují seznam zastávek, nicméně toto uložení by nebylo z hlediska vyhledávání optimální. Jelikož nevyužijeme všechna vstupní data pro naše vyhledávání a mohly by jsme data do budoucna potřebovat pro případně rozšíření aplikace tak původní soubory ukládáme v tabulce files v komprimované podobě. Tato tabulka nám dále bude sloužit pro ověření zda jsme již daný soubor zpracovali. Jelikož chceme vyhledat spoj mezi dvěma zastávkami potřebujeme ukládat do tabulky zastávek, která nám umožní rychle pomocí id zjistit jaké vlaky projíždí danou zastávkou. Následně musíme vytisknout jízdní řád daného vlaku, který splňuje podmínky, takže potřebujeme tabulku vlaků pro výsledný seznam zastávek daného vlaku. Dále musíme kontrolovat zda daný vlak je platný, tj. není odkloněn nebo zrušen a k tomu použijeme tabulku canceled.



Obrázek 1: Diagram kolekcí pro uložení do NoSQL databáze

1.3 Zvolená NoSQL databáze

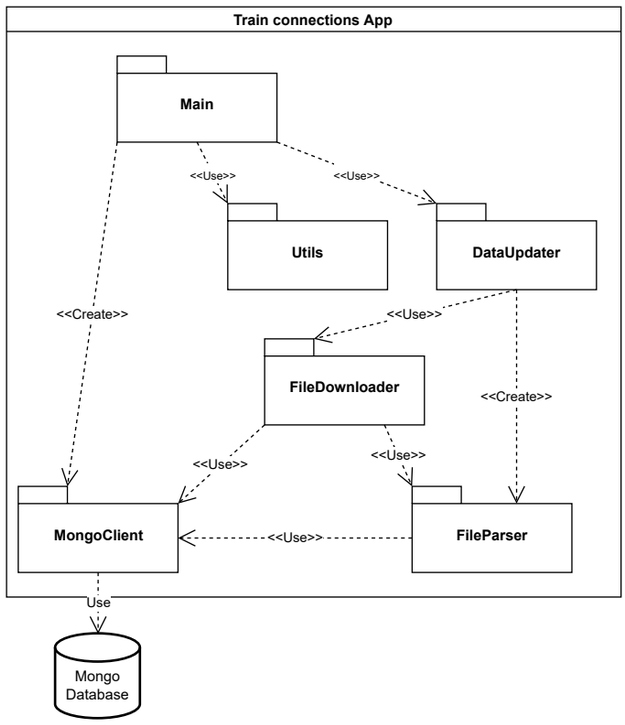
Pro tento projekt jsme si zvolili NoSQL databázi Mongo DB. Jedná se o dobře škálovatelnou objektově orientovanou databázi, která nemá pevně dané schéma. Ke všemu přistupuje jako k dokumentům, které shlukujeme do kolekcí. Jednotlivé dokumenty se ukládají ve formátu JSON, což využijeme jelikož vstupní data máme ve formátu XML, který není náročné převést na JSON. Jednotlivé kolekce jsou uloženy jako hashovací tabulka což využijeme pro rychlé vyhledání pomocí identifikátoru. Databázi budeme hostovat v cloudu přímo poskytovaným mongem, který je dostupný na adrese `https://cloud.mongodb.com/`. Jelikož databáze nemá pevně dané schéma, tak není třeba nic předem definovat.

2 Návrh, implementace a použití aplikace

V předchozí sekci jsme si zanalyzovali vstupní data, navrhli jak a kde je budeme ukládat. Nyní přejdeme k návrhu a implementaci klientské aplikace, která bude s databází komunikovat.

2.1 Návrh aplikace

Aplikace bude implementována v interpretovaném multiplatformním jazyce Python. V rámci hlavního skriptu je zahrnuto parsování argumentů a dotaz do databáze na vlakové spojení. Nad rámec zadání byly přidány argumenty pro vyhledání korektního názvu zastávky v databázi podle regulárního výrazu a více možností pro zobrazení tabulek vlakových spojů. Původní zadání zobrazovalo pouze celou trasu vlaku, nicméně nám přišlo praktické vyfiltrovat zastávky pouze od nástupní do výstupní stanice. Dále aplikace podporuje minimalistický režim zobrazení tabulek vlakových spojů, kde zobrazí pouze nástupní a výstupní zastávku.



Obrázek 2: Diagram architektury aplikace

2.1.1 DataUpdater

Jedná se pouze o wrapper pro stahování souborů, který instanciuje FileParser a FileDownloader. Následně spustí stahování souborů.

2.1.2 FileDownloader

Tento modul se stará o připojení k ftp serveru, jeho procházení stahování a následnou dekompresi souborů. Po kontrole zda-li jsme již daný soubor nezpracovali se provede jeho zpracování. Uživatel je informován o aktuálním stavu stahování pomocí výpisů, které informují o aktuálně stahovaném souboru a zobrazují progress bar, který indikuje kolik procent daného souboru je již zpracováno.

2.1.3 FileParser

FileParser provádí samotné parsování dat. Napřed zjistí o jaký typ zprávy se jedná. Dále vytáhne identifikátory dané zprávy. V případě, že se jedná o odklonový vlak, tak z identifikátorů určíme jaký vlak nahrazuje a původní vložíme do zrušených. Časové údaje převedeme na časové značky. V případě, že TrainActivityType není 0001 tak daný vlak neukládáme do Stations, ale musíme zastávky uložit v kolekci Trains.

2.1.4 MongoClient

Třída, která zabezpečuje připojení k MongoDb pomocí driveru Pymongo. MongoClient obsahuje připojení k databázi, definici kolekcí a metody pro práci s těmito kolekcemi. Většina update funkcí

je řešena pomocí upsert, což je funkce, která v případě že záznam již existuje aktualizuje data a v případě, že záznam se zadaným identifikátorem neexistuje tak jej vytvoří. Některé záznamy je nutné přidávat podmíněným upsertem, který ještě kontroluje čas vytvoření, aby jsme nepřepsali novější záznam starším.

2.1.5 Provedené optimalizace

První optimalizace se týkala prednačtení vlaků a zrušených vlaků pro všechny vlaky ve stanicích, jelikož budeme potřebovat všechny data pro všechna vlaky a tímto snížíme režijní náklady na komunikaci s databází. Tato optimalizace podle naměřených hodnot uvedených v přílohách vedla k snížení průměrné doby dotazu z 12.13 na 1.63 sekund.

Další pokus o optimalizaci byl pomocí agregační pipeliny s výpočtem průniků v databázi, která je uvedena v příloze. Podle naměřených hodnot byl výsledek výrazně horší. Po padesáti běžích jsme změřili průměrnou doba běhu bez pipeliny, která byla 4.43 sekundy a s pomocí pipeliny pro výpočet průniku 10.73 sekundy.

2.1.6 Dotaz na existující vlakové spojení

Algoritmus 1: Funkce pro nalezení spojů mezi dvěma zastávkami v zadaném čase.

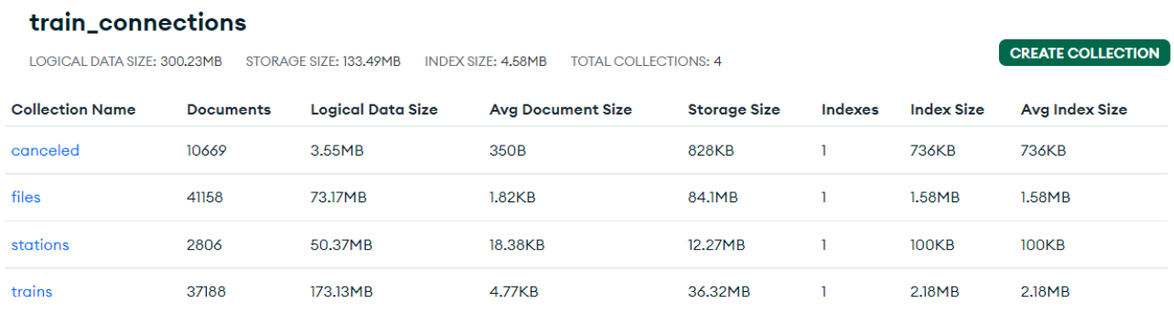
```
1  function trainConnection(start, end, time);
   Input  : Dvě zastávky a čas kdy má vlak jet
   Output: Vytiskne jízdní řády vlaků, který projíždí zadanými zastávkami v daný čas.
2  startStation = DB.Stations.find_one(start);
3  endStation = DB.Stations.find_one(end);
4  if startStation != None and endStation != None then
5      trainIds = getTrainsIntersectionAndReturnTrainIds(startStation,endStatio);
6      canceledTrains = DB.Canceled.find(trainIds);
7      trains = DB.Trains.find(trainIds);
8      forall train in trains do
9          if checkIfItsNotCanceled(canceledTrains, train, date) then
10             continue;
11         end
12         if not checkIfStartStationIsBeforeEndStation(train, start, end) then
13             continue;
14         end
15         if not checkIfTrainIsInValidInvertal(train,dateObj) then
16             continue;
17         end
18         train.PrintTimeTable();
19     end
20 end
```

2.2 Způsob použití

Pro spuštění aplikace je nutné mít nainstalovaný interpret Python, který je dostupný na adrese `https://www.python.org/downloads`.
Dále je nutné doinstalovat závislosti pomocí příkazu:
pip install -r requirements.txt
V projektu je dostupný dockerfile, který slouží pro tvorbu referenčního prostředí.
Po úspěšném vyřešení závislostí je možno skript spustit pomocí:
main.py [-h] [--update] [--start start] [--end end] [--time time]
možnosti:
-h, --help Zobrazí nápovědu aplikace
--update Provede aktualizaci záznamů spojů v databázi
--cut Zobrazí pouze zastávky hledané trasy
--less Zobrazí pouze nástupní a výstupní zástavku daného spoje
--start start Počáteční stanice cesty
--end end Cílová stanice cesty
--time time Čas zahájení cesty
--findStation regex Najít stanici v DB pomocí regex

2.3 Experimenty

Jelikož hlavní motivace pro úpravu struktury dat byla optimalizace rychlosti dotazování, tak nyní musíme ověřit jak efektivně je schopna se naše aplikace dotazovat databázového serveru.



Collection Name	Documents	Logical Data Size	Avg Document Size	Storage Size	Indexes	Index Size	Avg Index Size
canceled	10669	3.55MB	350B	828KB	1	736KB	736KB
files	41158	73.17MB	1.82KB	84.1MB	1	1.58MB	1.58MB
stations	2806	50.37MB	18.38KB	12.27MB	1	100KB	100KB
trains	37188	173.13MB	4.77KB	36.32MB	1	2.18MB	2.18MB

Obrázek 3: Ukázka kolekcí s velikostmi a počtem záznamů

Aplikace ověřována vůči vyhledávači vlakových spojů IDOS. Jako nejnáročnější dotaz jsme uvážili spoj mezi Brnem a Prahou, kde jsme očekávali největší počet vlakových spojů. Podle naměřených dat, které byly prezentovány již v části optimalizace trval průměrný dotaz pro Brno a Prahu okolo 2 sekund. Pro méně frekventované zastávky je vyhledání rychlejší.

A Časy 50ti běhů s výpočtem průniků pomocí agregační pipeliny

2.924574281999867, 3.0573588349998317, 3.2475778010002614, 3.3001856920000137, 3.491702355000143, 3.5335225500002707, 3.603201302999878, 3.656008280000151, 3.6862602940000215, 3.902841193000313, 4.112898724999923, 4.3346124869999585, 4.444156488000317, 4.707166889000291, 4.719769804999942, 4.828381604000242, 4.879923328000132, 5.022791206999955, 5.06264706400043, 5.146066542999961, 5.165435516000343, 5.2050638690002415, 5.255031523000071, 5.637228986000082, 6.368912635000015, 6.720831203999751, 6.859749643000214, 7.504848121000123, 7.626514622000286, 7.66132651199996, 8.149364758000047, 8.233227810999779, 9.451005719000023, 9.906503590000284, 10.080303243999879, 10.618985890999738, 10.94678371000009, 13.246543549999842, 13.280898620999778, 14.478144192999935, 16.217136443000072, 18.93498137100005, 20.25867224600006, 21.90657002199987, 22.95680854400007, 23.005688453999937, 24.868744892999985, 39.11488523600019, 40.59555155199996, 54.78033259299991

B Časy 50ti běhů s výpočtem průniků v paměti

2.2021792989999085, 2.320791115999782, 2.3337598040002376, 2.4098905129999366, 2.4126717399999507, 2.4374632920003023, 2.4461288990000867, 2.4829228459998376, 2.5440030649997425, 2.580405458000314, 2.589375504000145, 2.597247810999761, 2.612624278000112, 2.618643774999782, 2.64844838099998, 2.6804811269998936, 2.6823528380000425, 2.68947157599996, 2.7050201960000777, 2.7081278380001095, 2.732677782999872, 2.743166742000085, 2.755620024000109, 2.7632616280002367, 2.8076216380000005, 2.8322255610000866, 2.907910403000187, 2.9631162569999105, 2.972556162999083, 2.97586589499997, 3.0300565800002914, 3.069677436999882, 3.184725644000082, 3.236167845000182, 3.2378000909998264, 3.3349211169997943, 3.3729584190000423, 3.3825472969997463, 3.519972362000317, 3.5240186979999635, 3.5620927089998986, 4.012301646000196, 4.094976012999723, 4.263049490999947, 4.752061190999939, 4.923696798000037, 15.071882550000282, 18.38388517500016, 22.30927824499986, 28.01714842399997

C Časy 10ti běhů s dotazem do databáze pro každý vlak zastávky

8.8779071559984, 9.070468407997396, 9.793442759000754, 10.172086459999264, 11.15996500300389, 11.161229926998203, 12.225097766000545, 12.594505382003263, 16.21027339399734, 20.047925796003256

D Časy 10ti běhů s před stažením všech záznamů vlaků v průniku

0.7180842860034318, 0.9811189899992314, 1.2778079510026146, 1.3033881970040966, 1.3561894049998955, 1.3647056940026232, 1.6712773550025304, 1.728203165002924, 2.1876460290004616, 3.740619752999919

E Použitá pipeline pro MongoDb

```
query = dbClient.stations.aggregate([
    {
        "$facet": {
            "startStation": [{
                "$match":
                    {
                        "$and":[
                            {"_id": start},
                            {"Trains.ValidFrom": {"$gte": date}}
                        ]
                    }
            },
            { "$limit": 1 }
            ],
            "endStation": [{
                "$match":
                    {
                        "$and":[
                            {"_id": end},
                            {"Trains.ValidTo": {"$lte": date}}
                        ]
                    }
            },
            { "$limit": 1 },

            ]
        }
    },
    { "$addFields": {
            "to": { "$first": "$endStation" },
            "from": { "$first": "$startStation" }
        }
    },
    {
        "$project": {
            "from": "$from",
            "to": "$to",
            "intersection": {
                "$setIntersection": [
                    "$from.Trains._id",
                    "$to.Trains._id"
                    ]
                },
            "_id": 0
        }
    }
])
```