

# Ukládání rozsáhlých dat v NoSQL databázích

František Koleček, [xkolec08@stud.fit.vut.cz](mailto:xkolec08@stud.fit.vut.cz)

Tomáš Moravčík, [xmorav41@stud.fit.vut.cz](mailto:xmorav41@stud.fit.vut.cz)

David Sladký, [xsladk07@stud.fit.vut.cz](mailto:xsladk07@stud.fit.vut.cz)

zima 2022

# Obsah

<b>I</b>	<b>Analýza zdrojových dat a návrh jejich uložení v NoSQL databázi</b>	<b>1</b>
<b>1</b>	<b>Analýza zdrojových dat</b>	<b>2</b>
<b>2</b>	<b>Návrh způsobu uložení dat</b>	<b>3</b>
2.1	Definice entit a vazeb mezi nimi . . . . .	3
2.1.1	File . . . . .	3
2.1.2	PA . . . . .	4
2.1.3	TR . . . . .	5
2.1.4	Station . . . . .	5
2.1.5	Day . . . . .	5
<b>3</b>	<b>Zvolená NoSQL databáze</b>	<b>7</b>
<b>II</b>	<b>Návrh, implemetace a použití aplikace</b>	<b>8</b>
<b>4</b>	<b>Návrh aplikace</b>	<b>9</b>
4.1	Zpracování souborů XML . . . . .	9
4.1.1	Struktura XML souborů . . . . .	9
4.1.2	Zvolené Prostředky . . . . .	9
4.1.3	Způsob implementace . . . . .	10
<b>5</b>	<b>Způsob použití</b>	<b>11</b>
<b>6</b>	<b>Experimenty</b>	<b>12</b>
6.1	Rychlost zpracování XML souborů . . . . .	12

## 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 konkrétně zde. Detailní popis formátu dokumentů datové sady lze najít na stejném místě.

Datová sada se skládá z XML souborů. Tyto soubory jsou zveřejňovány 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, kterými 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 prováděn.

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 identifikátor vůči normálním spojům.

## Kapitola 2

# Návrh způsobu uložení dat

### 2.1 Definice entit a vazeb mezi nimi

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

#### 2.1.1 File

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

:File
filename
creation

#### PRECEDES

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



#### DEFINES

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



**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

**2.1.2 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

**SERVER BY**

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

:PA – :SERVED BY -> :TR

**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

**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

**IS IN**

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

:PA – :IS IN -> :Station

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

### 2.1.3 TR

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

:TR
Core
Company
Variant
TimetableYear
NetworkSpecificParameters

### 2.1.4 Station

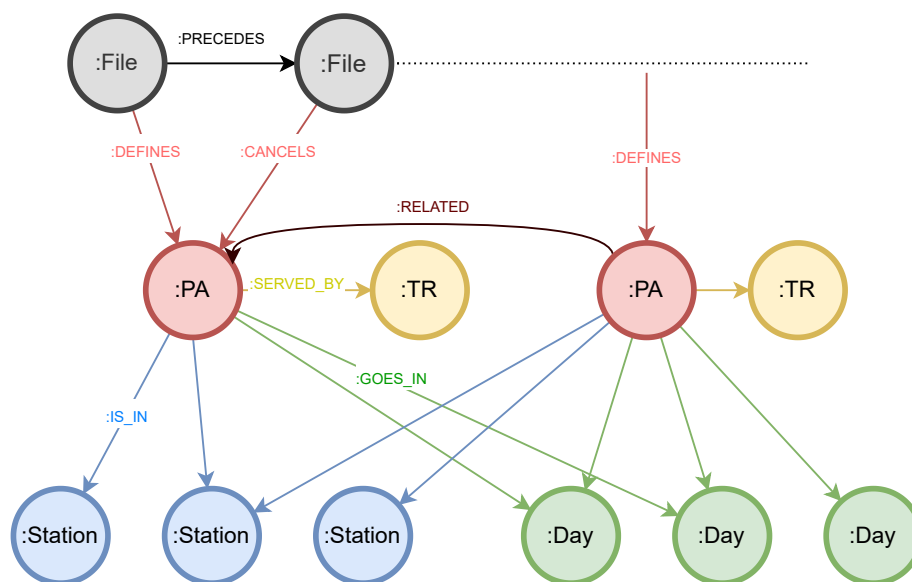
V XML souborech jsou také definovány stanice, kterými vlak projíždí. Tyto stanice budou modelovány pomocí entit :Station.

:Station
LocationPrimaryNumber
PrimaryLocationName
CountryCodeISO

### 2.1.5 Day

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

:Day
Date



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



## 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 entitami 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.

**Cíl:** Rozhrnout a zdůvodnit jaký druh NoSQL databáze je vhodný (zdůvodnění plyne částečně již z předchozího návrhu) a jaký konkrétní produkt NoSQL databáze bude použit.

**Obsah:** Určit typ databáze a konkrétní produkt NoSQL, vypsát jeho vlastnosti, které jsou pro toto řešení užitečné (a jiné než u jiných typů a produktů NoSQL) a zdůvodnit jejich vhodnost v kontextu předchozího návrhu.

**Prostředky:** Stručný volný text (až několik kratších odstavců) s případným vyznačením podstatných částí.

**Fáze projektu:** Zakončování návrhu a přechod k implementaci.

## 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 stanicích 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

**Cíl:** Poskytnout stručnou dokumentaci pro zprovoznění databáze a aplikace.

**Obsah:** Stručně popsat, jak celé řešení zprovoznit, tj. nasadit databázi i aplikaci vč. způsobu volání aplikace (příkazový řádek, parametry) pro úlohy předzpracování a nahrání dat ze zdroje do databáze a pro úlohy hledání nad databází tak, jak byly definovány v zadání.

**Prostředky:** Stručný text obsahující návod (popis) s ukázkami způsobu volání aplikace (např. pro skripty by to byl kód příkazového řádku).

**Fáze projektu:** Dokončování implementace, chystání dokumentace pro předání výsledného systému zákazníkovi.

## 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.