

Dokumentace k semestrální práci předmětu
BI-BIG

Michal Konečný

16. prosince 2018

Obsah

1	Úvod	3
1.1	Zadání semestrální práce	3
1.1.1	Požadavky	3
1.1.2	Forma	3
2	Business pohled na data	4
2.1	Popis dat z jednotlivých tabulek	4
2.1.1	Tabulka rybáři	4
2.1.2	Tabulka sumář	4
2.1.3	Tabulka ryba	5
2.2	Ukázka dat z jednotlivých tabulek	5
3	Import dat do HDFS	6
4	Agregace - 1	7
4.1	Ukázka výsledné agregace	7
5	Agregace - 2	7
5.1	Ukázka výsledné agregace	8
6	Agregace - 3	8
6.1	Ukázka výsledné agregace	9
7	Logstash + Elasticsearch	10
7.1	Import dat a vytvoření indexu	10
7.2	Dotazy	11
7.2.1	Filtrování	11
7.2.2	Třídění	11
7.2.3	Wildcard	12
8	Dashboard + vizualizace	13
8.1	Zastoupení jednotlivých druhů ryb	13
8.2	Úspěšnost jednotlivých nástrah	13
8.3	Průměrná délka jednotlivých druhů	14
8.4	Průměrná váha jednotlivých druhů	14
8.5	Výsledný dashboard	15
9	Závěr	16
10	Přílohy	17
10.1	Logstash config	17

1 Úvod

1.1 Zadání semestrální práce

1.1.1 Požadavky

- vybrat si min. 3 různé datasety (možno i vygenerovat smysluplná data náhodně)
 - každý vstupní dataset bude mít nejméně 50.000 záznamů (pokud budete předvádět v učebně, zvolte data tak, aby bylo časově reálné úlohu ukázat)
- tyto (min.) 3 datasety nainportovat do databáze či distribuovaného file-systemu (např. HDFS)
- vytvořit nový dataset, který bude agregovat data z jednoho původního datasetu
- vytvořit nový dataset, který bude agregovat data ze dvou původních datasetů najednou
- vytvořit nový dataset, který bude agregovat data ze dvou datasetů najednou, z čehož jeden bude výsledkem předchozí agregace a uložit ho zpět do databáze/na file systém
- vytvořit nad kterýmkoliv datasetem index v ElasticSearch (či podobném enginu) a připravit 3 různé dotazy do tohoto indexu (nestačí index databáze, je potřeba použít indexovací engine jako je ElasticSearch)
 - využít filtrování
 - využít třídění
 - použít wildcard hledání (www.soft.com)
- k indexu připojit vizualizační nástroj (např. Kibana) a udělat dashboard s 4 smysluplnými pohledy na vaše data
- po dohodě (dopředu schváleno) je možné použít i jiné technologie než je Spark a ElasticSearch

1.1.2 Forma

- k semestrální práci je potřeba zpracovat kompletní dokumentaci (odevzdat ve formátu PDF), která bude obsahovat minimálně:
 - klasickou strukturu včetně hlavičky, rejstříku, úvodu, hlavní části, závěru
 - businessový pohled na to, jaká data se budou používat
 - popis a ukázkou dat z použitých datasetů (ukázka = několik jednotek, maximálně desítek řádků z každého datasetu)

- kompletní příkazy použité pro jednotlivé transformace
- v příloze použité konfigurační soubory (např. pro search engine)
- na základě dokumentace by mělo být možné se vstupními datasety kompletně replikovat vaší práci!

2 Business pohled na data

Data, která jsem použil ve své semestrální práci, jsou náhodně generovaná data a mají simulovat data Českého rybářského svazu. Jde o data rybářů a jejich úlovků za posledních 5 let - mezi lety 2014-2018, vždy od 1.3. do 1.12 daného roku. Dataset rybáři obsahuje 50000 záznamů, známe jméno, příjmení, pohlaví, datum narození a telefon jednotlivce. Dataset ryby obsahuje též 50000 záznamů, u jednotlivých ryb evidujeme rodové jméno - kapr/amur/štika/sumec/candát/pstruh, délku v cm a váhu v kg. Poslední dataset sumář obsahuje přes 120000 záznamů a jde o dataset, který simuluje spojení mezi rybáři a jejich úlovky. Zaznamenáváme id rybáře a id ryby, dále datum ulovení, typ vody, typ nástrahy a okres, ve kterém byla ryba chycena.

2.1 Popis dat z jednotlivých tabulek

2.1.1 Tabulka rybáři

- id - automaticky generované id
- jmeno - jméno rybáře
- prijmeni - příjmení rybáře
- pohlavi - pohlaví rybáře - může nabývat hodnot M(muž)/Z(žena)
- telefon - telefonní číslo rybáře(bez předvolby)
- datum_narozeni - datum narození rybáře (tvar d.m.yyyy)

2.1.2 Tabulka sumář

- id_rybar - id rybáře, které odpovídá id z tabulky rybari
- id_ryba - id ryby, které odpovídá id z tabulky ryby
- datum_uloveni - datum ulovení ryby (tvar d.m.yyyy)
- typ_vody - informace jestli šlo o úlovek na svazové nebo soukromé vodě (může nabývat hodnot svazova/soukroma)
- nastraha - na jakou nástrahu byla daná ryba ulovena, může nabývat těchto hodnot:
 - boilies, peleta, zizala, pecivo, cervi, mrtva rybka, ziva rybka, guma, trpytka, rousnice, kukurice, rotacka, wobler

- okres - v jakém okresu byla ryba ulovena, může nabývat hodnot - všechny okresy v ČR

2.1.3 Tabulka ryba

- id - id ryby, generováno automaticky
- rodove_jmeno - rodové jméno ryby, může nabývat hodnot - kapr, amur, stika, sumec, candat, pstruh
- delka - délka ryby v cm
- vaha - váha ryby v kg

2.2 Ukázka dat z jednotlivých tabulek

id	meno	prijmeni	pohlavi	datum_narozeni	telefon
1	Doubravka	Sloukova	Z	16.10.1969	603946861
2	Krystof	Jiracek	M	3.9.1986	732983944
3	Radmila	rehorova	Z	22.7.1983	775222416
4	Ivana	Francova	Z	6.2.1969	773227594
5	stefan	Dite	M	24.4.1968	777861126
6	Viola	Belkova	Z	11.2.1996	736123456

(a) Tabulka rvhari

id_rybar	id_ryba	datum_uloveni	typ_vody	nastraha	okres
40282	32879	23.3.2016	svazova	cervi	Kolin
375	6952	28.11.2016	svazova	zizala	Chrudim
22169	19348	21.4.2016	soukroma	cervi	Strakonice
39838	9408	2.7.2016	soukroma	cervi	Beroun
7628	18151	22.7.2016	svazova	kukurice	Prachatice
29033	36711	28.5.2016	svazova	boilies	cesky Krumlov
31401	6141	10.8.2016	svazova	peleta	Nachod

(b) Tabulka sumar

id	rodove_jmeno	delka	vaha
1	kapr	60	2
2	kapr	54	5
3	kapr	49	4
4	kapr	44	6
5	kapr	42	4

(c) Tabulka ryby

Obrázek 1: Ukázka z jednotlivých tabulek

3 Import dat do HDFS

Při importu dat do HDFS jsem postupoval stejným způsobem, jako na cvičení č.5.

<https://courses.fit.cvut.cz/BI-BIG/tutorials/05/index.html>

Container pro SPARK master a worker:

```
docker build -f spark.df -t spark .
docker-compose up
```

Container pro spark-shell:

```
docker run -it -p 8088:8088 -p 8042:8042 -p 4041:4040 --
name driver -h driver spark:latest bash
```

Připojení spark-shell na master:

```
spark-shell --master spark://<IP adresa mastera>:7077
```

Container pro HDFS:

```
docker run --name hadoop -t -i sequenceiq/hadoop-docker /
etc/bootstrap.sh -bash
```

Vytvoření složek a import dat:

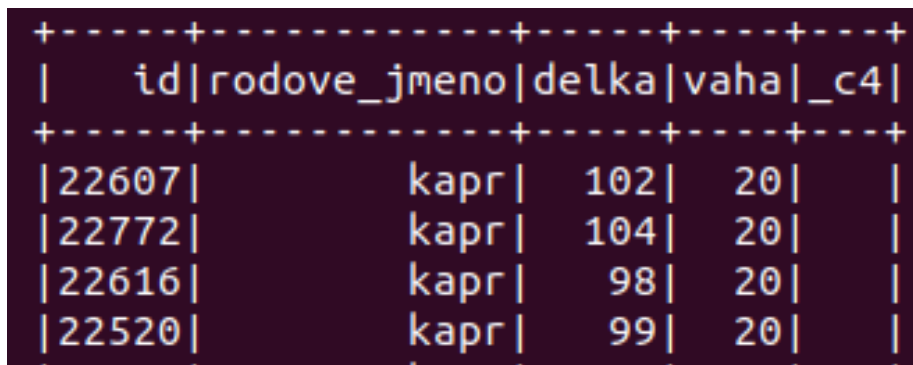
```
export PATH=$PATH:/usr/local/hadoop/bin/
hdfs dfs -mkdir /semestralka
hdfs dfs -mkdir /semestralka/dataset
curl https://gitlab.fit.cvut.cz/konecni4/bi-big/tree/
master/rybari.csv --output rybari.csv
curl https://gitlab.fit.cvut.cz/konecni4/bi-big/tree/
master/ryby.csv --output ryby.csv
curl https://gitlab.fit.cvut.cz/konecni4/bi-big/tree/
master/sumar.csv --output sumar.csv
hdfs dfs -put ./rybari.csv /semestralka/dataset/
hdfs dfs -put ./ryby.csv /semestralka/dataset/
hdfs dfs -put ./sumar.csv /semestralka/dataset/
```

4 Agregace - 1

V první agregaci jsme nejprve načetli dataset ryby z HDFS do sparku. Dále jsme z něj vytvořili tabulku se sloupci id, rodove_jmeno, delka a vaha. Následně jsme z této tabulky vybrali kapry s délkou nad 90cm a váhou větší než 14kg a výsledek setřídili sestupně. A uložili nový dataset zpátky na HDFS.

```
val ryby = spark.sqlContext.read.format("csv").option("header", "true").option("delimiter", ";").option("inferSchema", "true").load("hdfs://172.17.0.5:9000/semestralka/dataset/ryby.csv")
ryby.createOrReplaceTempView("ryby")
val nejvetsi_kapri = spark.sqlContext.sql("SELECT * FROM ryby WHERE rodove_jmeno = 'kapr' delka > 90 AND vaha >= 15 ORDER BY vaha DESC")
nejvetsi_kapri.write.option("header", "true").csv("hdfs://172.17.0.5:9000/semestralka/dataset/nejvetsi_kapri.csv")
```

4.1 Ukázka výsledné agregace



id	rodove_jmeno	delka	vaha	_c4
22607	kapr	102	20	
22772	kapr	104	20	
22616	kapr	98	20	
22520	kapr	99	20	

Obrázek 2: Největší kapři

5 Agregace - 2

Ve druhé agregaci jsme načetli datasety rybari a sumar. Ve sparku z nich vytvořili tabulku s odpovídajícími sloupci jako v původním datasetu. A následně nad nimi provedli JOIN přes sloupec id v tabulce rybari a sloupec id_rybar v tabulce sumar.

```
val rybari = spark.sqlContext.read.format("csv").option("header", "true").option("delimiter", ";").option("inferSchema", "true").load("hdfs://172.17.0.5:9000/semestralka/dataset/rybari.csv")
```

```

rybari.createOrReplaceTempView("rybari")
val sumar = spark.sqlContext.read.format("csv").option("
    header", "true").option("delimiter", ";").option("
    inferSchema", "true").load("hdfs://172.17.0.5:9000/
    semestralka/dataset/sumar.csv")
sumar.createOrReplaceTempView("sumar")
val rybari_sumar = spark.sqlContext.sql("SELECT r.id, r.
    jmeno, r.prijmeni, pohlavi, s.datum_ulozeni, id_ryba,
    nastraha, typ_vody, okres FROM rybari r JOIN sumar s
    ON r.id = s.id_rybar")
rybari_sumar.write.option("header", "true").csv("hdfs
    ://172.17.0.5:9000/semestralka/dataset/rybari_sumar.
    csv")

```

5.1 Ukázka výsledné agregace

id	jmeno	prijmeni	datum_ulozeni	id_ryba	nastraha	typ_vody	okres
1	Doubravka	Sloukova	24.10.2015	45727	guma	svazova	Tabor
1	Doubravka	Sloukova	8.7.2018	16601	peleta	svazova	Cheb
1	Doubravka	Sloukova	16.7.2017	36927	cervi	svazova	Jicin
1	Doubravka	Sloukova	30.11.2016	14702	pecivo	soukroma	Melnik
2	Krystof	Jiracek	3.4.2014	49361	trpytka	svazova	Kladno
3	Radmila	rehorova	24.9.2018	42464	ziva rybka	svazova	ceske Budejovice

Obrázek 3: Spojení tabulky rybari a sumar

6 Agregace - 3

V poslední, třetí agregaci jsme načetli tabulku z agregace číslo 2, tedy tabulku rybari_sumar. A následně provedli JOIN nad touto tabulkou a tabulkou ryby, přes sloupce id_ryba v tabulce rybari_sumar a id v tabulce ryby. Výsledek jsme zapsali zpět na HDFS k dalšímu použití.

```
val rybari_sumar_new = spark.sqlContext.read.format("csv")
    .option("header", "true").option("inferSchema", "true")
    .load("hdfs://172.17.0.5:9000/semestralka/dataset/rybari_sumar.csv")
rybari_sumar_new.createOrReplaceTempView("rybari_sumar")
val rybari_sumar_ryby = spark.sqlContext.sql("SELECT rs.id AS id_rybar, rs.jmeno, rs.prijmeni, rs.pohlavi, rs.datum_ulozeni, id_ryba, f.rodove_jmeno, f.delka AS delka_v_cm, f.vaha AS vaha_v_kg, nastraha, typ_vody, okres FROM rybari_sumar rs JOIN ryby f ON rs.id_ryba = f.id")
rybari_sumar_ryby.write.option("header", "true").csv("hdfs://172.17.0.5:9000/semestralka/dataset/rybari_sumar_ryby.csv")
```

6.1 Ukázka výsledné agregace

id_rybar	jmeno	prijmeni	pohlavi	datum_ulozeni	id_ryba	rodove_jmeno	delka_v_cm	vaha_v_kg	nastraha	typ_vody	okres
1	Doubravka	Sloukova	Z	24.10.2015	45727	sumec	175	51	guna	svazova	Tabor
1	Doubravka	Sloukova	Z	8.7.2018	16601	kapr	77	10	peleta	svazova	Chebi
1	Doubravka	Sloukova	Z	16.7.2017	36927	amur	90	13	cervi	svazova	Jicin
1	Doubravka	Sloukova	Z	30.11.2016	14702	kapr	60	8	pectvo	soukroma	Melnik
2	Krystof	Jiracek	M	3.4.2014	49361	pstruh	44	2	trpytka	svazova	Kladno
3	Radnilla	rehorova	Z	24.9.2018	42464	stika	95	10	ziva rybka	svazova	ceske Budejovice

Obrázek 4: Spojení tab z AG2 a tab ryby

7 Logstash + ElasticSearch

Nad datasetem z agregace č.3 vytvořím index v ElasticSearch a připravím 3 různé dotazy do tohoto indexu. Postupuji podobným způsobem jako na cvičení č.9.

<https://courses.fit.cvut.cz/BI-BIG/tutorials/09/index.html>

7.1 Import dat a vytvoření indexu

Pro import dat do ElasticSearch využiji nástroj Logstash. Vytvořím adresář elastic, do kterého stáhnou připravený balíček ze cvičení č.9.

<https://drive.google.com/open?id=1PqEtoRUxRjWXWkQQOR-201tMh6DY7M0i>

Do složky elastic/logstash/datasets zkopíruji dataset

`rybari_sumar_ryby.csv`

Jediný soubor, který je třeba upravit je v

`elastic/logstash/pipeline-logstash.conf`

Jeho přesnou podobu můžete najít v příloze. Spustíme docker a jakmile projde příkaz, otevřeme si logy z logstash, abychom viděli v jakém stavu je import.

```
docker-compose up -d
```

```
docker logs -f logstash
```

Po dokončení importu již můžeme otevřít Kibanu a vytvořit Index Pattern založený na již vytvořeném indexu v ElasticSearch.

- Panel Management
- Index Pattern
- Create Index Pattern
- Next step a vybrat, že nechceme použít časový filtr.
- Create Index Pattern

Nyní můžeme přejít do Dev tools a začít s dotazy do indexu.

7.2 Dotazy

7.2.1 Filtrování

Ulovení kapři s délkou větší než 90 cm a váhou větší než 15 kg.

```
GET _search
{
  "query": {
    "bool": {
      "must": [
        { "match": { "rodove_jmeno": "kapr" } }
      ],
      "filter": [
        { "range": { "vaha": { "gte": 15 } } } },
        { "range": { "delka": { "gte": 90 } } } }
      ]
    }
  }
}
```

7.2.2 Třídění

Najde všechny ulovené sumce a seřadí je dle data ulovení(od nejstaršího po nejaktuálnější)

```
GET _search
{
  "query": {
    "match": {
      "rodove_jmeno": "sumec"
    }
  },
  "sort": {
    "datum_ulozeni.keyword": { "order": "asc" }
  }
}
```

7.2.3 Wildcard

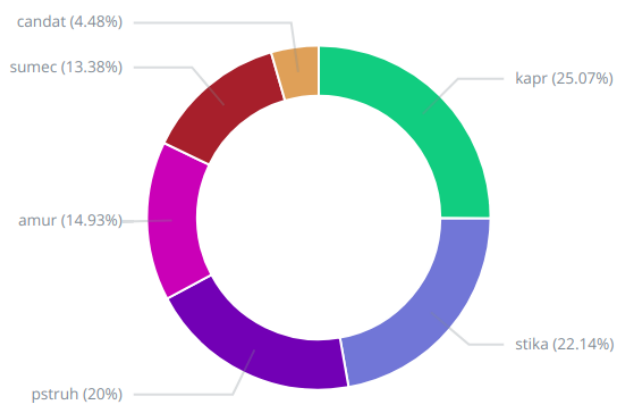
Hledá rybářky jejichž příjmení obsahuje koncovku ova.

```
GET _search
{
  "query": {
    "wildcard": {
      "prijmeni" : "*ova*"
    }
  }
}
```

8 Dashboard + vizualizace

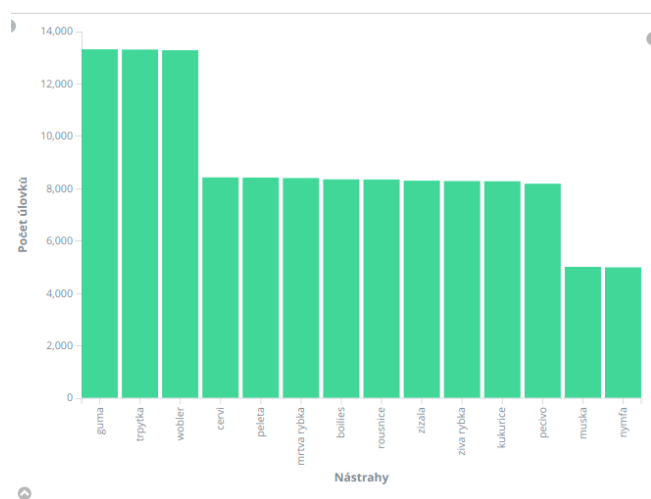
Nakonec vytvoříme nový dashboard a provedeme pár vizualizací.

8.1 Zastoupení jednotlivých druhů ryb



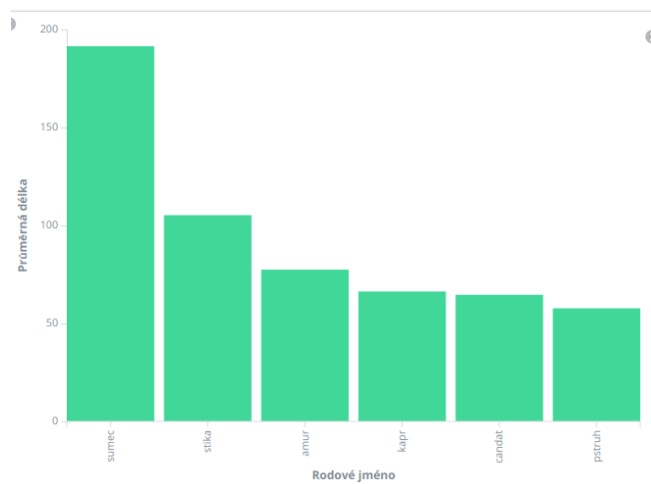
Obrázek 5: Zastoupení druhů

8.2 Úspěšnost jednotlivých nástrah



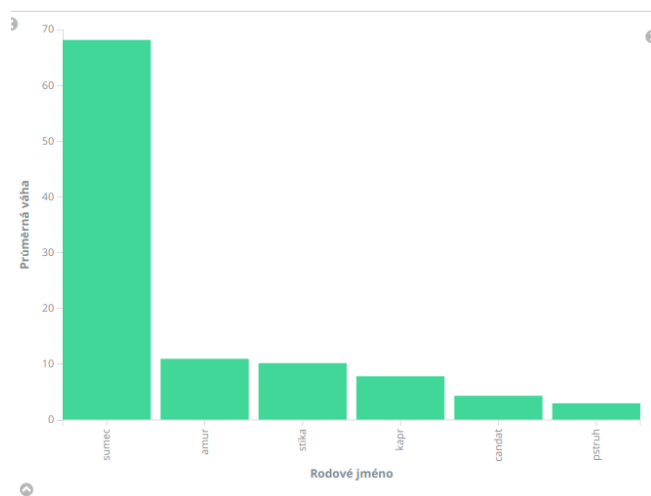
Obrázek 6: Úspěšnost nástrah

8.3 Průměrná délka jednotlivých druhů



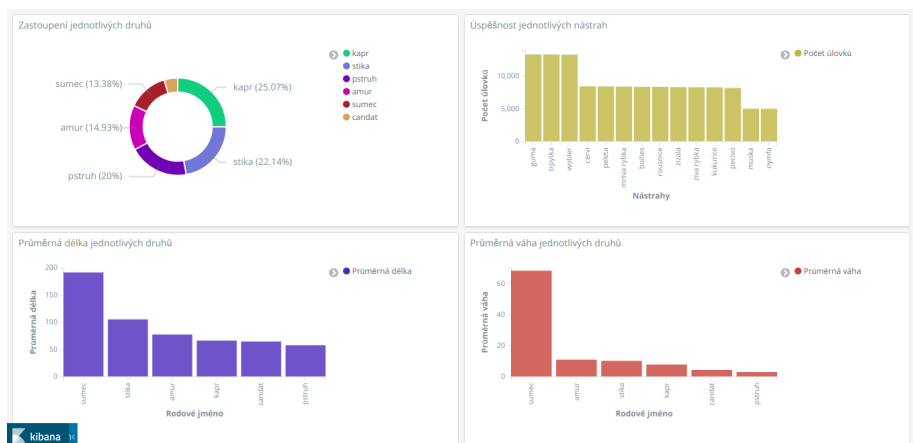
Obrázek 7: Průměrná délka

8.4 Průměrná váha jednotlivých druhů



Obrázek 8: Průměrná váha

8.5 Výsledný dashboard



Obrázek 9: Dashboard

9 Závěr

Cílem této semestrální práce bylo prozkoumat základní použití technologií využívaných v předmětu BI-BIG. To se podařilo. Bohužel vzhledem k tomu, že jsem použil náhodně generovaná data (která spolu sice souvisí, ale není jich dostatek), nejsou konečné statistiky zcela relevantní. Pro další projekt bych tedy raději použil nějaká reálná data.

10 Přílohy

10.1 Logstash config

```
input {
  file {
    path => "/datasets/ryb_sumar_ryb.csv"
    start_position => "beginning"
    codec => plain { charset=>"Windows-1250" }
  }
}

filter {
  csv{
    separator => ";"
    columns => [" datum_uloveni",
               " id_rybar",
               " id_ryba",
               " typ_vody",
               " nastraha",
               " okres",
               " jmeno",
               " prijmeni",
               " pohlavi",
               " telefon",
               " rodove_jmeno",
               " delka",
               " vaha" ]
  }
  mutate { convert => [" delka", " integer" ] }
  mutate { convert => [" vaha", " integer" ] }
}

output {
  elasticsearch {
    hosts => "http://elasticsearch:9200"
    index => "vsechno"
  }
}
```