BellaDati

Data cleaning Demo POC

March 16th, 2016

Created by:
Ondrej Spalek
BellaDati HQ EU
Prague, Czech Republic

Table of Contents

1 Purpose of the document

The purpose of this document is to describe the solution and summarize results of given Proof-of-Concept (PoC) delivered by BellaDati for **Customer**.

2 PoC requirements

BellaDati received the following resources:

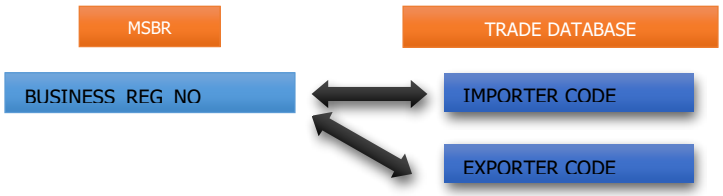
- PoC requirements specification
- Sample data files

File name	Description
export_sabah_ogos.txt	Export trade table for Semenanjung, Sabah dan Sarawak. July 2015.
import_sabah_ogos.txt	Import trade table Semenanjung, Sabah dan Sarawak. July 2015.
import_zb_julai.txt	Import trade table for Semenanjung sahaja. July 2015.
MSBR_JUL_AUG_SEPT15_POC.xlsx	Business Register for July, August and September 2015.

- completed BellaDati Questionnaire document.

The requirements can be summarized into the following points:

1. Combine given data tables (trade import/export tables with BR) based on **key identifiers matching**. Example illustrating the requirement:



Business Registration	Spec	Example	Right Format	Check Digit
ROB	Alphanumeric	12345-M	000012345	M
	9 Digit	JM0034567-V	JM0034567	V
ROC	Numeric	3456-V	3456	V
	1-8 Digit	12-M	12	M

Columns containing the appropriate keys are identified as:

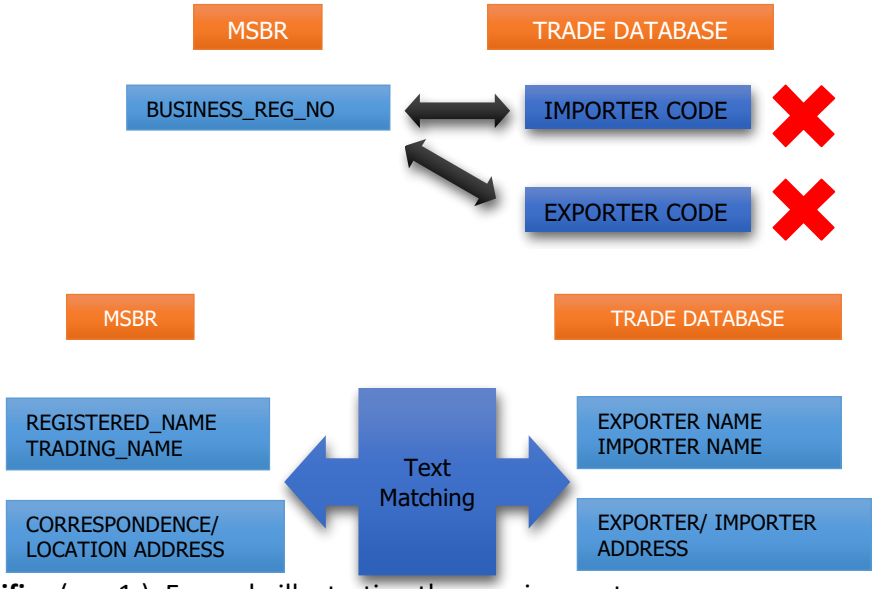
File name	Column name	Description
export_sabah_ogos.txt **import_sabah_ogos.txt**	Exporter/Consignor Code	Exporter code. Example B137811A
export_sabah_ogos.txt	Importer/Consignee Code	Importer code. Example:

import_sabah_ogos.txt		B137811A
import_zb_julai.txt	IMPORTER CODE	Importer code. Example: 526896-V
import_zb_julai.txt	EXPORTER CODE	Exporter code. Example: 305119-V
MSBR_JUL_AUG_SEPT15_POC.xlsx	BUSINESS_REG_NO	Business Registration Number. Example: 418817/ IP0274828

The requirement is to find appropriate records in the BR based on matching Importer/Exporter codes with BUSINESS_REG_NO. Identifier key values may be in two formats (detected either by "check digit" or by textual analysis):

- 9 digits alphanumeric
- 1-8 digits numeric

2. Combine given data tables (trade import/export tables with BR) based on **text matching** of particular columns. This approach should be applied for the records, which were not matched by the **key identifier** (see 1.). Example illustrating the requirement:



Columns containing the appropriate values for text matching are identified as:

export_sabah_ogos.txt import_sabah_ogos.txt	import_zb_julai.txt	MSBR_JUL_AUG_SEPT15_POC.xlsx
Exporter/Consignor Name		REGISTERED_NAME or TRADING_NAME
Importer/Consignee Name		REGISTERED_NAME or TRADING_NAME
	IMPORTER NAME	REGISTERED_NAME or

		TRADING_NAME
	EXPORTER NAME	REGISTERED_NAME or TRADING_NAME
Exporter/Consignor Address		Correspondence address = (STREET_1_A, STREET_2_A, TOWN_A, POSTCODE_A, STATE_A) or Location address = (STREET_1_B, STREET_2_B, TOWN_B, POSTCODE_B, STATE_B)
Importer/Consignee Address		Correspondence address = (STREET_1_A, STREET_2_A, TOWN_A, POSTCODE_A, STATE_A) or Location address = (STREET_1_B, STREET_2_B, TOWN_B, POSTCODE_B, STATE_B)
	EXPORTER ADDRESS	Correspondence address = (STREET_1_A, STREET_2_A, TOWN_A, POSTCODE_A, STATE_A) or Location address = (STREET_1_B, STREET_2_B, TOWN_B, POSTCODE_B, STATE_B)
	IMPORTER ADDRESS	Correspondence address = (STREET_1_A, STREET_2_A, TOWN_A, POSTCODE_A, STATE_A) or Location address = (STREET_1_B, STREET_2_B, TOWN_B, POSTCODE_B, STATE_B)

3. Create sample reports and visualizations based on the resulting data sets.
4. Demonstrate the readiness of BellaDati for applying the solution for Big Data.

3 Solution in BellaDati

BellaDati is able to address all key requirements specified in PoC. Based on the fact, that the desired results are heavily depending on advanced data processing, BellaDati will utilize mainly the it's **ETL** and **Machine learning** capabilities do deliver the solution. The solution has been divided into several phases:

1. Analyzing the data structure
    a. Textual analysis of the key identifiers
    b. Textual analysis of the columns to be used for matching
2. Identifying necessary steps to be executed in BellaDati to deliver desired results
    a. Define number and content of iterations resulting into desired results

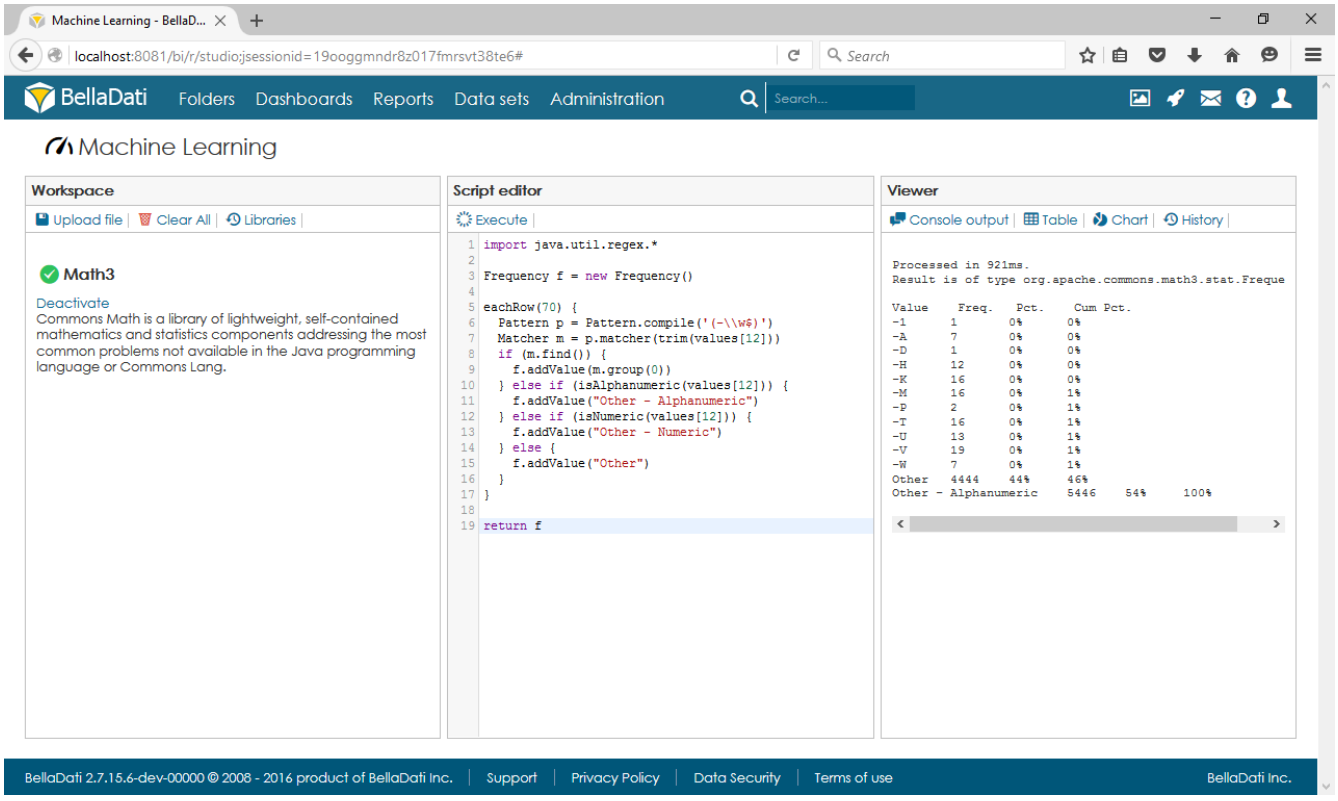
   b. Identify the desired transformation rules to put the values into normalized format, data cleaning
3. Importing and transforming data, creating appropriate data sets (data models)
4. Combining imported data sets (several iterations)
   a. Key identifiers matching
   b. Text matching
   c. GEO mapping
5. Creating reports

3.1 Data structure analysis

Quick analysis of the data files showed the high level of uncertainty of the key identifier values and content of the columns dedicated to the text matching. Examples:

- Inconsistent importer/exporter codes
- Invalid symbols presented (e.g. spaces, quotes, &, …)
- Inconsistent parts of the address

To be able to realize what is the range of possible matching issues and identify possible solutions, BellaDati did the textual and statistical analysis of the file content. **BellaDati Machine Learning** module was used. As an example, here is a code used for analyzing the import/export trade files.



```
1.   import java.util.regex.*
2.
3.   Frequency f = new Frequency()
4.
5.   eachRow(70) {
6.     Pattern p = Pattern.compile(a'(-\\w$)')
7.     Matcher m = p.matcher(trim(values[12]))
8.     if (m.find()) {
9.       f.addValue(m.group(0))
10.    } else if (isAlphanumeric(values[12])) {
11.      f.addValue("Other - Alphanumeric")
12.    } else if (isNumeric(values[12])) {
13.      f.addValue("Other - Numeric")
14.    } else {
15.      f.addValue("Other")
16.    }
17.  }
18.
19.  return f
```

Code used:

And the results were:

Value	Freq.	Pct.	Cum Pct.
-1	1	0%	0%
-A	7	0%	0%
-D	1	0%	0%
-H	12	0%	0%
-K	16	0%	0%
-M	16	0%	1%
-P	2	0%	1%
-T	16	0%	1%
-T	13	0%	1%
-U	19	0%	1%
-W	7	0%	1%
Alphanumeric	5446	44%	46%
Other	4444	54%	100%

After similar analysis has been done for all key columns, we found out the following transformation will be needed:

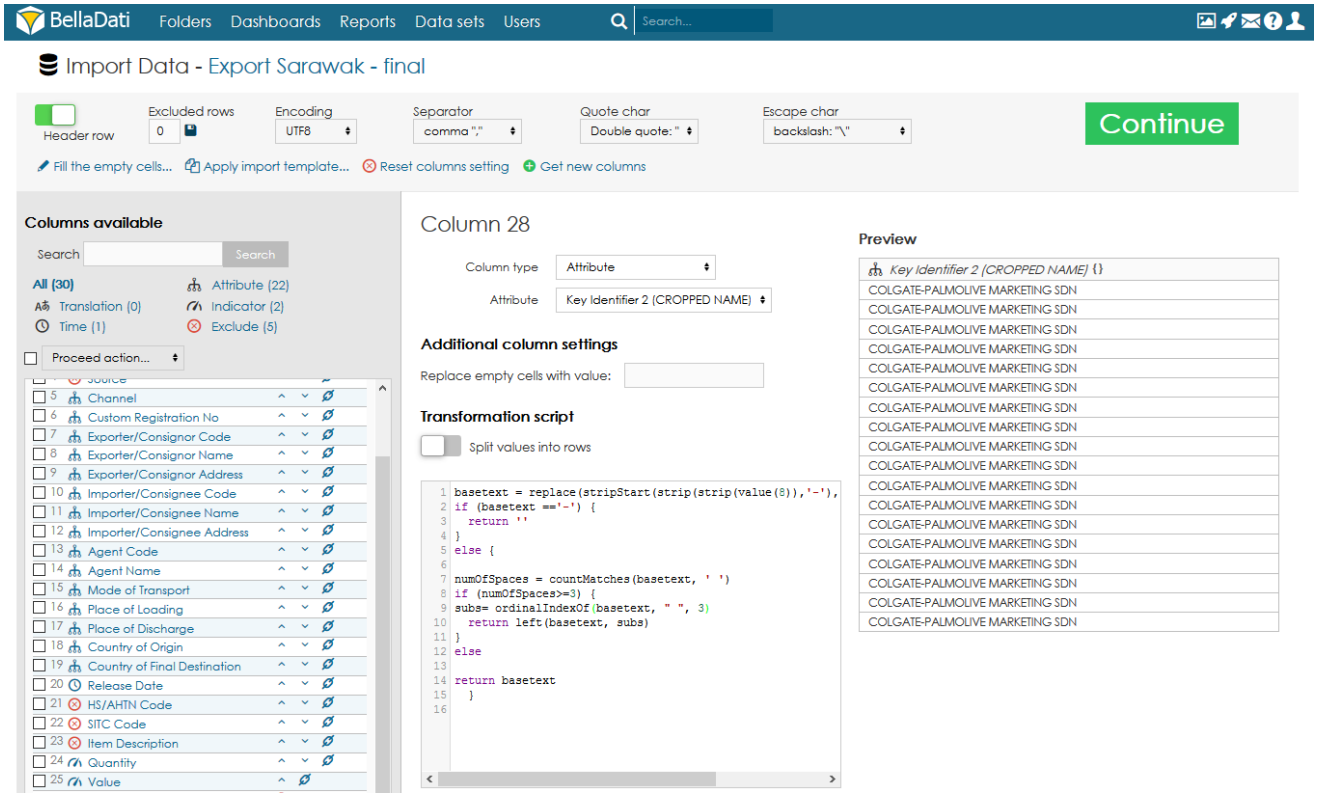
1. **Basic cleaning** - whitespaces were stripped from the start and end of strings. Other special characters, such as dash, quotes etc. were also stripped from the start if necessary.
2. **Importer/Exporter code transformation**
    a. remove XNIL values. Check if it contains dash (check digit after dash)
        i. if not, return code as it is
        ii. If yes, split the string by dash
    b. Check which part is longer -> longer part is code, shorter part is check digit
    c. check whether it is numerical or alphanumerical
        i. If numerical - > return as i tis
        ii. If alphanumerical – add correct number of leading zeros so the string has 9 chars
    d. character on beginning or at the end – this character has to be removed.

3. **Importer/Exporter name normalizing**
    a. removing invalid characters, stripping values
    b. transforming to uppercase
    c. trim up to three words - to eliminate majority of misspellings and differences in exporter name representation, only up to three words from the beginning of the name were used for matching
4. **Importer/Exporter address normalizing**
    a. addresses in MSBR file are separated into multiple columns (street, town, postcode, state), but in trade files, they are in one column. Strip all spaces, dots and commas and merge all four columns in MSBR into one ADDRESS columns which will be used for matching.
    b. addresses in trade files have various formats – some are with commas, some with dots, some with spaces only.

3.2 Importing and transforming data

Imported data from import/export trade files has been transformed according the transformation rules identified in previous chapter (1-5). As an example, here is the code used for normalizing of one of the key identifiers (exporter name):

```
basetext = replace(stripStart(strip(strip(value(8)),'-'),'"'), ' &amp;', '&')
if (basetext =='-') {
  return ''
} else {
  numOfSpaces = countMatches(basetext, ' ')
}
if (numOfSpaces>=3) {
  subs= ordinalIndexOf(basetext, " ", 3)
  return left(basetext, subs)
}  else  {
  return basetex
```



All provided data files has been transformed and imported into BellaDati. They are now represented as 3 independent data sets: **BR**, **Exports** and **Imports**. Appropriate data model has been created as well.

3.3 Combining data (records matching)

To match records from BR table and Trade table, native graphical user interface available in BellaDati was used. To reflect requirements for matching, custom joining condition was used and the process is divided into 4 iterations:

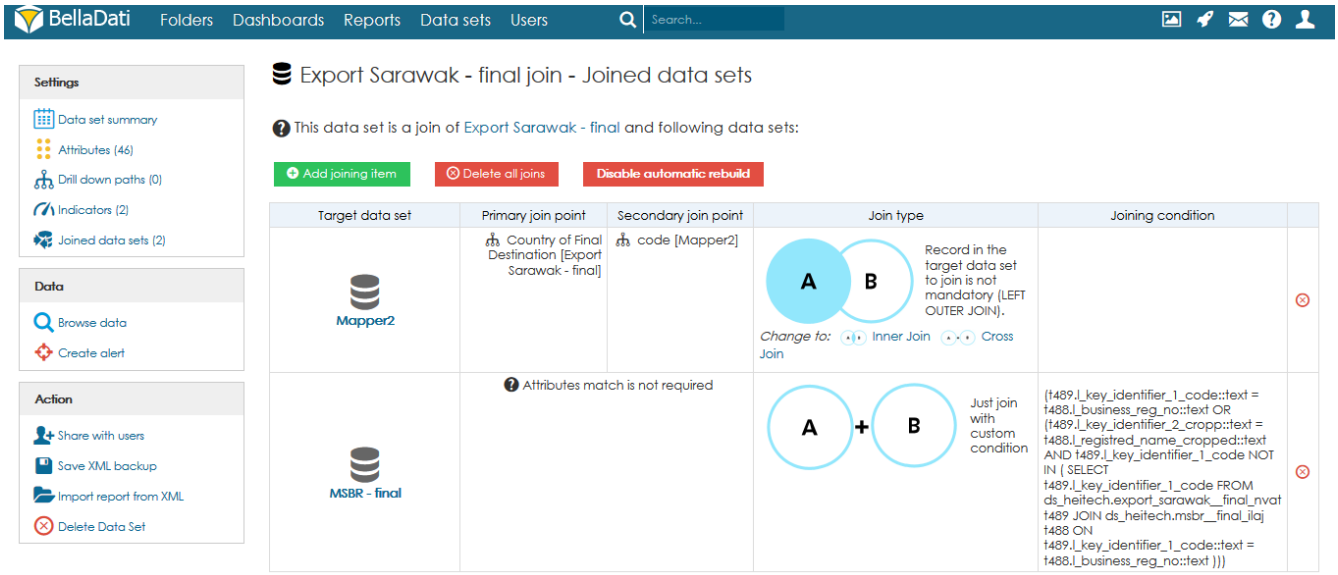
1. In first iteration, BellaDati is trying to find match between **Exporter code** and **Business registration number**.
2. In the second iteration, for unmatched records, second condition is applied. This condition is trying to find match between **Exporter name** and **Registered name**.
3. Third iteration is trying to match records by the **Trading name**
4. Fourth iteration is trying to match records by the **Address** or **Location**.

These four iterations have been executed several times with different condition parameters and matching algorithm. Based on the first results (built on plain text matching) and very low rates (around 1-5%) of matched records, we decided to apply advanced method to match the data – for the PoC purposes, we used the **Fuzzy matching** algorithm. Except the fuzzy matching, BellaDati supports the following methods:

- **Levenstein** - calculates the Levenstein distance between two strings
- **Metaphone** – it's similar to Fuzzy matching and calculates the representative code string. Two strings are then deemed similar if they have the same codes.

By using the **Fuzzy matching** method we increased the matching rates up to 60%.

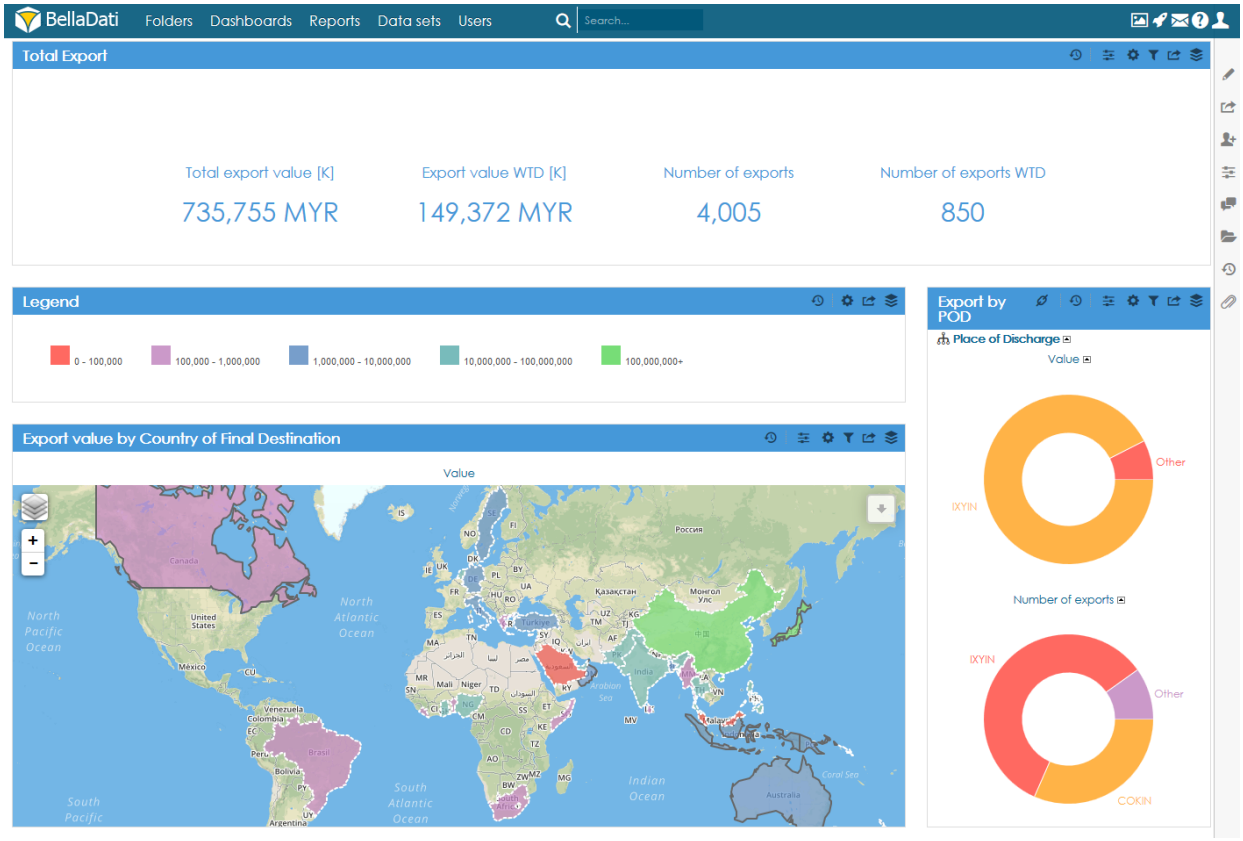
The screenshot below shows BellaDati UI containing joined data set combining trade table with MSBR based on plain text matching.



3.4 Reports

Based on the final data set, a sample report was created to demonstrate abilities of BellaDati. The report contains multiple visualization, filters and other advanced features of BellaDati, such as associated views.

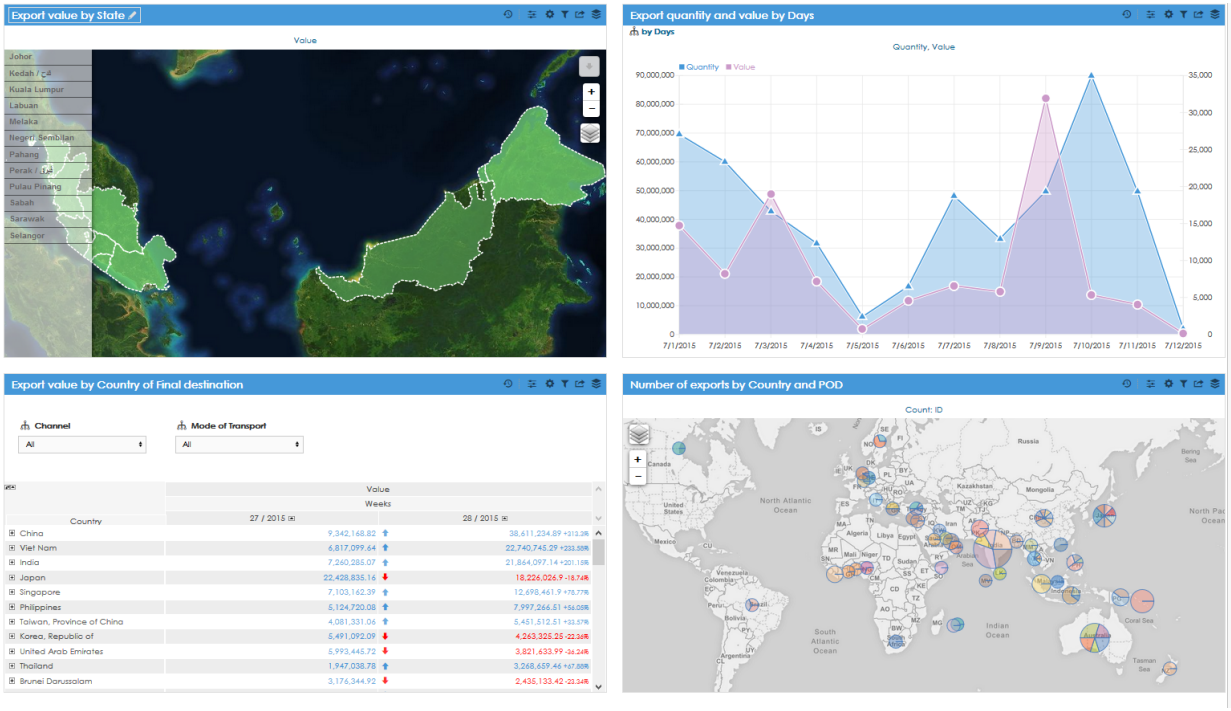
First part of report contains four basic KPIs, displaying total export value, total number of exports, export value WTD (week-to-date) and number of exports WTD. Second part of report shows a world map with total export value for each country and a two pie charts displaying values for each Place of discharge. It is possible to select specific country by clicking on the map and filter the pie charts for that country.



Picture 1: First and second part of report

Third part of report contains another map which displays export value for each state of Malaysia. Next to the map there is a line chart showing total export quantity and total export value for each day.

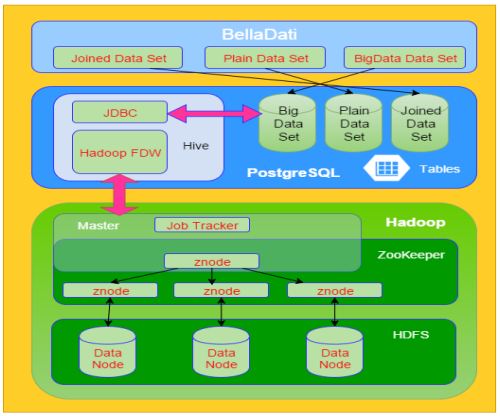
Fourth part of sample report is used for showing the change (growth or decay) of export between weeks. Multidimensional table with drill-down by weeks and by Country of final destination was used. Conditional formatting was applied on the table to highlight the growth and decay of each values. Last visualization is another map. In this case a point-based map was used instead of shape-based map. Each point describes number of exports for different country and each point is divided in slides for each point of discharge.



3.5 Big Data Support

BellaDati supports several ways how to store the data and how the data are represented on the database level. Here is a brief overview:

1. **Plain data sets up to 100GB** – these data sets are represented as physical data structures (partitioned tables, materialized views) inside the BellaDati reporting database (e.g. PostgreSQL)
2. **Joined data sets up to 100GB** – same as plain data sets, but different changes calculation method is used (eager algorithms).
3. **Big Data data sets (up to several TB)** – BellaDati utilizes the Hadoop platform and can access it natively over the Foreign Data Wrapper (Hadoop FDW) directly from PostgreSQL database subsystem (see the illustration below).



BellaDati can support another Big Data engines like SAP HANA (BellaDAti is certified partner of SAP HANA, SAP BW using the connectors available in BellaDati.