Eduard Cidraque Sala

Informative document

This is a document outlining what has been done in the code development since now. It is not a detailed explanation as should be (API documentation style) but it aims to serve as an informal summary of the code and the work done over the past few weeks.

There are three packages, one for each functionality: *HBaseToWIMEDSDB*, *WIMEDSbonitaToHBase* and *HBaseToWIMEDSDB.*
Link to Github repository on WISCENTD-UPC organization:
https://github.com/educisa/enablingDescrAnalysis-WIMEDSdata

INDEX

Sprint 1. HBaseToWIMEDS.

This sprint is related to Objective 1. Having a proper data synchronization between the master repository and WIMEDS, Administration Units exportation.

Work done:

The code of this part is in *HBaseToWIMEDSDB* package inside the Github repository. There is the code which implements the connector for reading data from the Organisation Units table in HBase and exporting it to WIMEDS (PostgreSQL). The logic implemented for updating the data that conflicts with the master data stored in HBase is also there.

Then, the code there can be divided into full extraction functionality and update functionality.

Java files full extraction: *HBase_orgUnitExtraction.java, organisationUnitsExtraction.java*
Java files update: *updateDB_orgUnits.java*, *organisationUnitsUpdate.java*

a) The main steps full extraction code follows are:

- scanner creation over the HBase Table
- use the scanner to traverse and extract data from the HBase Table.
- for each batch returned by the scanner, do the logic for loading data into a PostgreSQL database, that is an SQL Query.
- once the scanner has finished, delete it.
- update extraction times in control.properties file (*thisExtraction, prevExtraction).*
- data loading in Postgres database

b)The main steps for update code are similar to the full extraction ones. The only thing changing is the way the scanner is defined and the SQL queries (updates).

- scanner creation over the HBase Table with startTime and endTime.
- use the scanner to traverse and extract data from the HBase Table that has been modified since the last extraction.
- for each batch returned by the scanner, do the logic for loading data into a PostgreSQL database, that is an SQL Query.
- once the scanner has finished, delete it.
- update extraction times in control.properties file (*thisExtraction, prevExtraction).*
- data loading in Postgres database

control.properties values used in these functionallities:

PostgresSQL(WIMEDS): *DBurl, DBusr, DBpwd*
HBase(WICD-landing zone): *url, batch*
thisExtraction i *prevExtraction* values are both used for full extraction and update code.

Possible improvements:

- Store also last full extraction and update times separately from *thisExtraction* and *prevExtraction*. This information may be needed.
- Think of a way to parametrize attributes needed in the Administration Unit WIMEDS table. By the moment the attributes stored are: *parentid, name, shortname, datelastupdated, leaf, levelnumber i url.*

Sprint 2.WIMEDSbonitaToHBase.

This sprint is related to Objective 2. Loading application data to the data lake.

Work done:

The code of this part is in the WIMEDSbonitaToHBase package inside the Github repository. Java files: *WIMEDSextraction.java, IndividualRequestExtraction.java, HbaseUtilities.java, ReqMedProcessMetadataExtraction, BonitaRESTAuth.java.*

The implemented code exports data related to the REQ_RequestMedicines process from WIMEDS to the WICD landing zone. Specifically the code export the data from (WIMEDS tables names):
Request, ShipmentR, RequestDocument, Disease, RequestStatus, MedicalSupply and *Manufacturer.*

These are the steps the code follows in order to export WIMEDS *REQ_RequestMedicines* process data to HBase:

a) create the *WIMEDS-Table-Data* in HBase if it has not been created yet.
b) ask Bonita REST API for *REQ_RequestMedicines* process metadata last update date.
b.1) if (*last update date* > l*ast WIMEDS data extraction)* then update metadata version (for the row key). No WIMEDS metadata is stored in HBase at this moment. If this happens a new metadata extraction would be needed.
b.2) else nothing has to be done with respect to metadata
c) export all the data from the WIMEDS tables mentioned before. *RequestMedicines* process data is stored in WIMEDS-Table-Data (HBase) under the same key. Each table in a different column-family. (Data:Request, Data:Disease, etc.)
d) update last WIMEDS extraction time

control.properties values used in these functionalities are the ones commented there as *--HBase_RequestMedicineProcess* and *--BonitaWIMEDS*.

Possible improvements:

- More efficient update logic. Only the modified tables/values should be extracted. Now, each time an update in the WIMEDS-Table-Data table in HBase is needed, a full data extraction from the WIMEDS database is done, that is another WIMEDSextraction.java execution is done and another row is created in WIMEDS-Table-Data with the row key format:
  Subsystem$Individualdata$process$metadataVersion$extractionDateTime
  WIMEDS$Individual$RequestMedicine$1$2020-12-29T16:59:55.789
- parametrize which WIMEDS tables have to be exported to HBase.
- take into account that some WIMEDS tables may have a lot of data (e.g., Request) and a split of this data would be necessary. To extract it and also to export it into HBase tables. Now, all the request data is extracted with one REST API call, when

the table is bigger, more REST API calls would be needed. For storing it in HBase a similar technique as the one used for storing some WIDP Forms should be used.
- Store *REQ_RequestMedicines* process metadata

To correct, bug:
Between steps a) i b) should be another step. The one related with the Bonita Authentication. The code is implemented in *BonitaRESTAuth.java* and it stores jsessionid and X-Bonita-API-Token in control.properties file.
I can get the value from jsessionid but the problem is that i can not get the X-Bonita-API-Token with the java code I implemented and I would need to install another version of Bonita to be able to configure and retrieve this token. Now each time I execute the code I get the jsessionid and X-Bonita-API-Token from Postman. The code implemented should work with a proper Bonita configuration.

Sprint 3. HBaseToWIMEDS.

This sprint is related to Objective 3. The focus will be on data pre-processing and transformation of the data lake data.

Work done:
The code of this part is in the *HBaseLZtoPostgresTZ* package inside the Github repository. Java files: *mainToTransformedZone.java, organisationUnitsExtractionToTZ.java, toTransformedZone.java, tz_mainUpdate.java, tz_updateAdminUnit.java* and *PostgresSqlTransformedZone.java*

The java code implements the ETL steps for loading the data from the WICD landing zone (HBase) to the modeled WICD transformed zone (PostgreSQL). The main steps are:
a) create Tables and Materialized Views in Postgres transformed zone if needed
b) perform Administration Units extraction from HBase landing zone table and export it to transformed zone DB (ETL process)
c) extract data from WIMEDS-Table-Data in landing zone and export it to transformed zone DB (ETL process)
d) set extraction times
e) populate medicalSupply_manufacturer table (it is populated with data from medicalSupply and manufacturer table)
f) refresh materialized views

Step d) is basically needed for updating the AdministrationUnits table in the transformed zone. The logic used is the same as the one used in *HBaseToWIMEDSDB.*

Requests table is the fact table where the measures come from. Other tables are used as dimension tables, filters for the analysis.
ShipmentR table is there but still not used by the Tableau (which uses materialized views).
In the repository there is a *createTables.txt* with the SQL queries used in the *PostgresSqlTransformedZone.java* code.

Possible improvements:

- update logic for fact and dimension tables. Only AdministrationUnit (geographical dimension) has a proper update logic. The other tables are extracted from WIMEDS-Table-Data each time and this should be only done if the dateTime (or timestamp stored by HBase) in the row key has changed from the past exportation from landing zone to transformed zone.
- parametrize tables wanted in the transformed zone, that is WIMEDS Tables stored in WIMEDS-Table-Data HBase table.
- only refresh materialized views if needed. Now it is always done since always the code is executed, an update into the transformed zone is done (it would be not needed if data in the landing zone has not changed since last update).
- refactor tables creation SQL queries (add FK restrictions)
- create a materialized view with shipmentR data to enable further analysis over shipments data.

Sprint 4. Tableau. Multidimensional analysis.

This sprint is related to Objective 4. The focus will be on bringing descriptive analysis over WIMEDS data to end users.

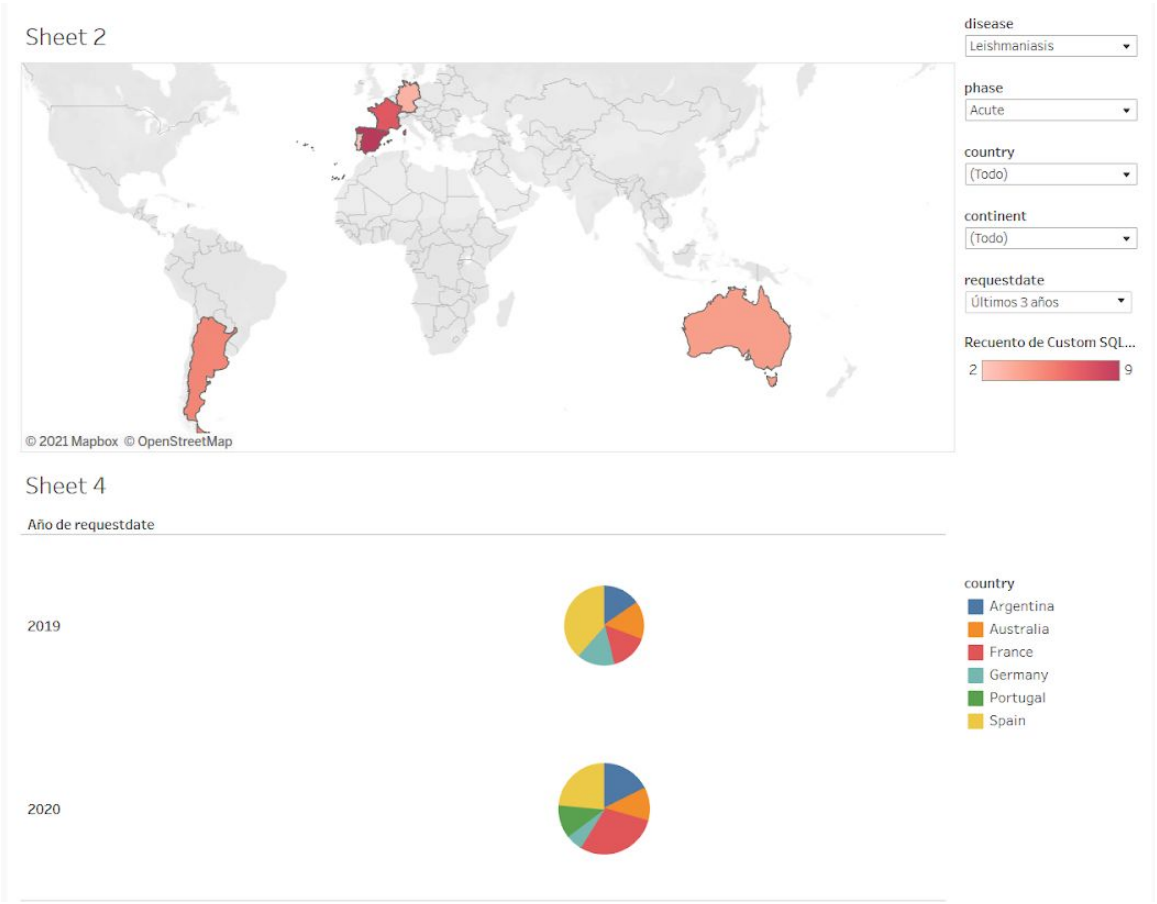
Tableau is the OLAP tool used. Dashboards and Sheets done are published in my personal Tableau Online account. Here in the next pages I attach some figures of the Dashboards created from materialized views in the PostgreSQL database that simulates the transformed zone.

In order to get this results that can be seen in the dashboards an insertion in the Requests Table from the transformed zone was made. The inserts can be found in *insertsRequestsTesting.txt* in the Github repository.

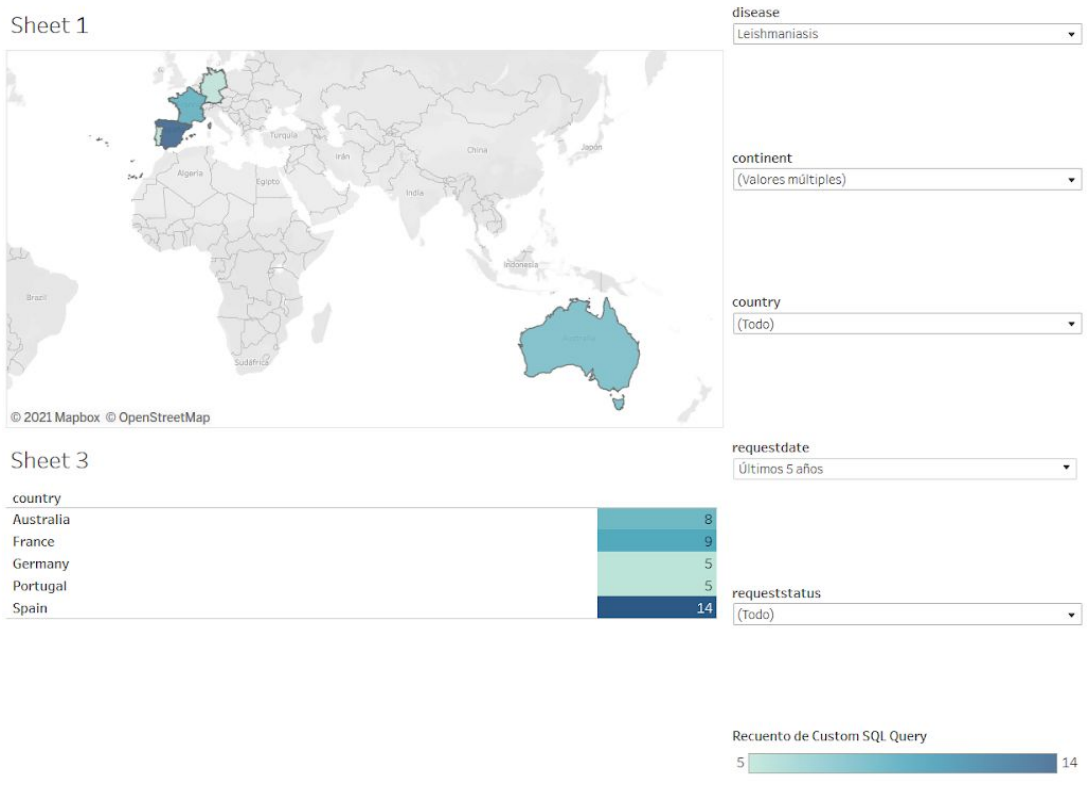
Possible improvements:

- add analysis for Shipments.
- find a way to extract or generate more detailed geographical dimension from Request data.

Eduard Cidraque Sala



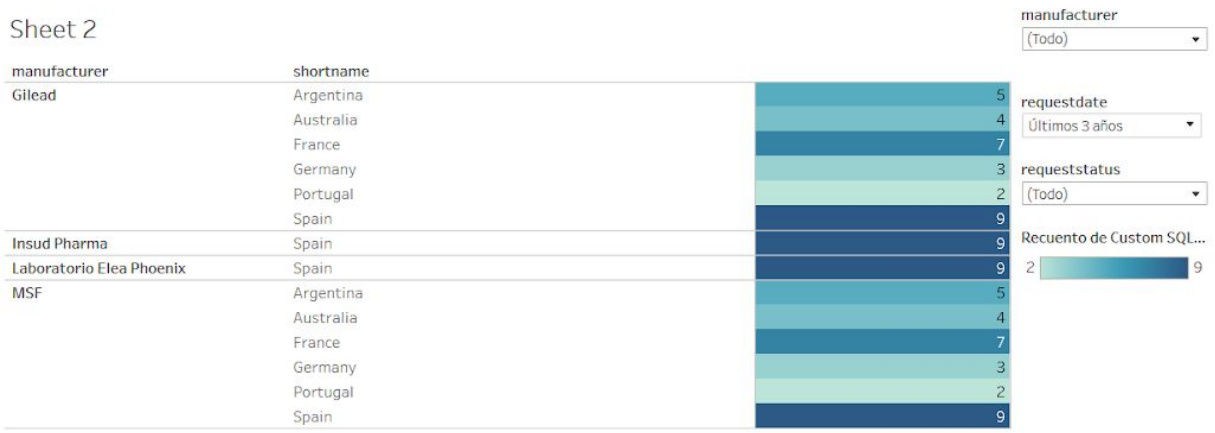
Dashboard 1



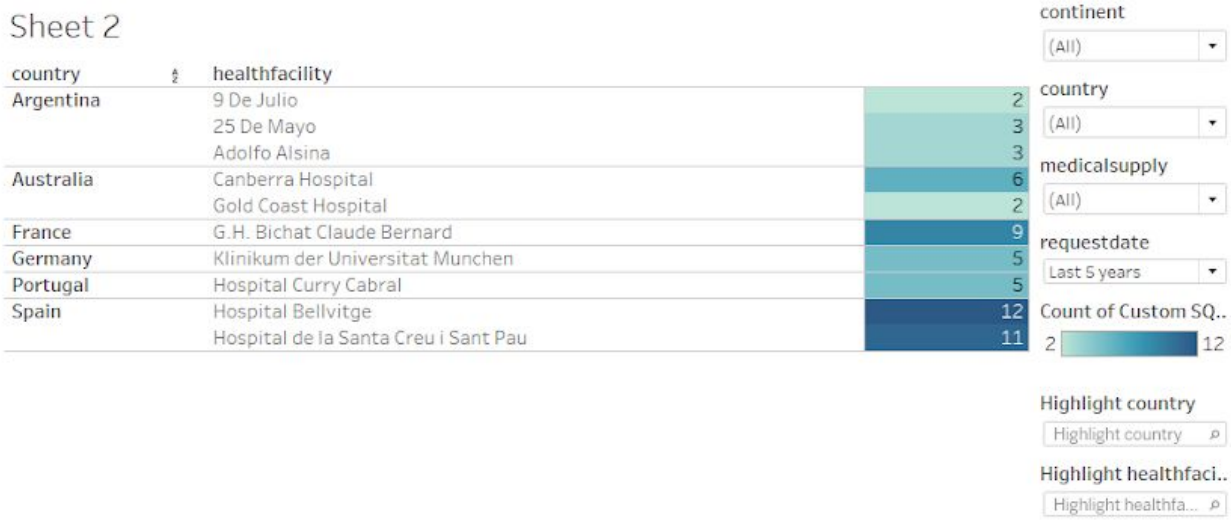
Dashboard 2


Sheet 1
Recuento de Custom SQL Query
patient_age
continent
(Todo)
country
(Todo)
disease
Leishmaniasis
phase
Acute
Sheet 3
Recuento de Custom SQL Query
patient_weigthinkg

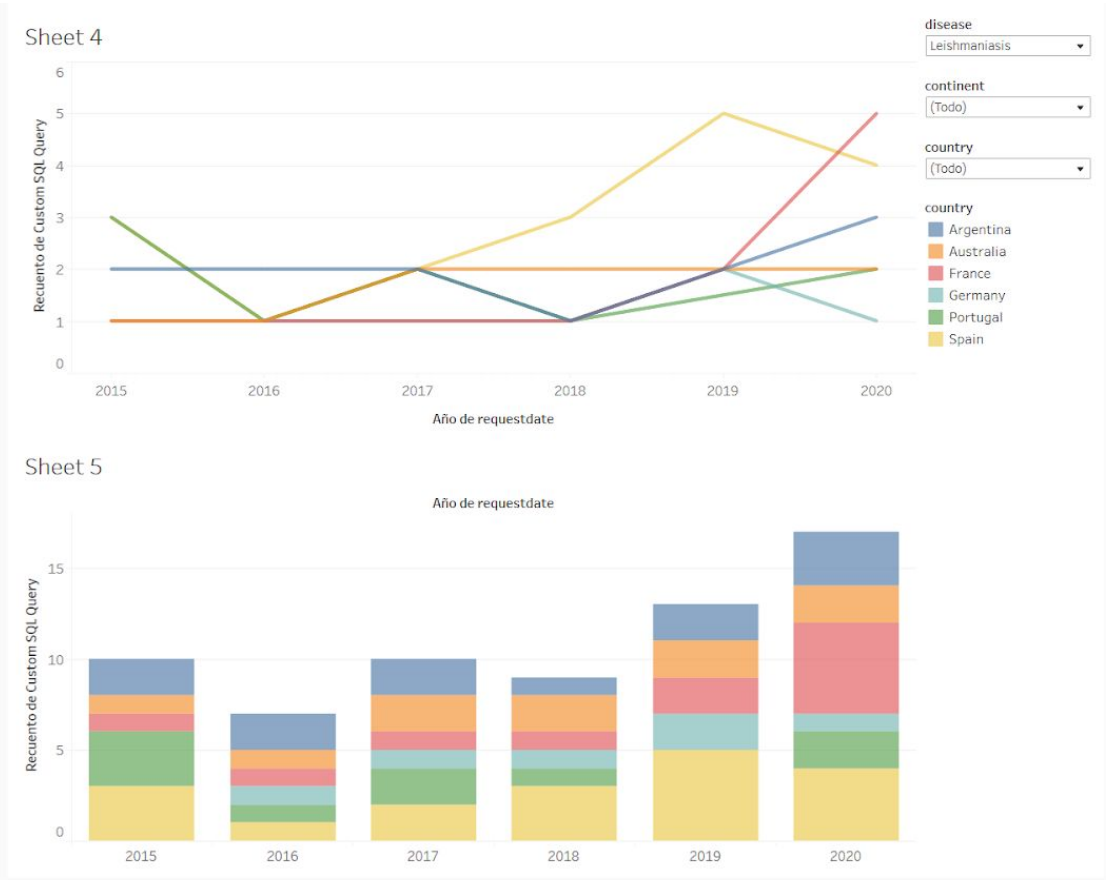
Dashboard 3



Dashboard 4



Dashboard 5


Sheet 4
Recuento de Custom SQL Query
6
5
4
3
2
1
0
2015
2016
2017
2018
2019
2020
Año de requestdate
disease
Leishmaniasis
continent
(Todo)
country
(Todo)
country
Argentina
Australia
France
Germany
Portugal
Spain
Sheet 5
Año de requestdate
Recuento de Custom SQL Query
15
10
5
0
2015
2016
2017
2018
2019
2020

Dashboard 6