COMPETITIVENESS AND INNOVATION FRAMEWORK PROGRAMME

CIP-ICT-PSP-2013-7 Pilot Type B

WP3 – Service platform integration and deployment in cloud infrastructure

D3.1.2: Heterogeneous data repositories and related services

Deliverable Lead: ATOS

Deliverable due date: 29/02/2016

Actual submission date: 29/02/2016

Version: 1.3

This project is partially funded under the ICT Policy Support Programme (ICT PSP) as part of the Competitiveness and Innovation Framework Programme by the European Commission under grant agreement no. 621074

Document Control Page

Title	D3.1.2: Heterogeneous data repositories and related services
Creator	Miguel Ángel Esbrí (ATOS)
Description	This document introduces the second prototype of the different geospatial data repositories and their associated services
Publisher	FOODIE Consortium
Contributors	Mario Núñez Jiménez
Creation date	28/02/2015
Type	Text
Language	en-GB
Rights	copyright "FOODIE Consortium"
Audience	☐ internal ☒ public ☐ restricted
Review status	☐ Draft ☐ WP leader accepted ☐ Technical Manager accepted ☒ Coordinator accepted
Action requested	☐ to be revised by Partners ☐ for approval by the WP leader ☐ for approval by the Technical Committee ☐ for approval by the Project Coordinator
Requested deadline	

About the project

FOODIE project aims at creating a platform hub on the cloud where spatial and non-spatial data related to agricultural sector is available for agri-food stakeholders groups and interoperable. It will offer: an infrastructure for the building of an interacting and collaborative network; the integration of existing open datasets related to agriculture; data publication and data linking of external agriculture data sources, providing specific and high-value applications and services for the support of planning and decision-making processes.

FOODIE project is addressed to four basic groups of users: a) stakeholders from the agriculture sector as end-users of final applications, b) public sector for communication with farmers about taxation, subsidies and regulation, c) researchers for large scale experimentation on real data and d) ICT companies for the development of new applications for agriculture and food sector, mainly using implemented tools

FOODIE specifically works on three pilots:

- Pilot 1: Precision Viticulture (Spain) will focus on the appropriate management of the inherent variability of crops,
- Pilot 2: Open Data for Strategic and Tactical Planning (Czech Republic) will focus on improving future management of agricultural companies (farms) by introducing new tools and management methods,
- Pilot 3: Technology allows integration of logistics via service providers and farm management including traceability (Germany).

Contact information

Miguel Angel Esbrí

Project Coordinator

Atos Spain, Madrid, Spain

E-mail: miguel.esbri@atos.net

URL: http://www.foodie-project.eu

Twitter: https://twitter.com/FOODIE_Project

Glossary

The glossary of terms used in this deliverable can be found in the public document "FOODIE_Glossary.pdf" available at: http://www.foodie-project.eu

Abbreviations and Acronyms

Abbreviation / Acronym	Description
API	Application Programming Interface
HTTP	Hypertext Transfer Protocol
OGC	Open Geospatial Consortium
ORDBMS	Object-Relational Database Management System
RDBMS	Relational Database Management System
RDF	Resource Description Framework
SOS	Sensor Observation Service
WFS	Web Feature Service
WCS	Web Coverage Service
WMS	Web Map Service

Table 1: Abbreviations and Acronyms

Executive Summary

This document introduces the second prototype of different geospatial databases and access services that were deployed in FOODIE platform during the second year of the project.

1 Components and related services overview

As conceptually conceived from the start of the project, FOODIE service platform comprises a set of diverse repositories necessary for the storage, retrieval and processing of different sources of information such as satellite imagery, cartography (e.g., roads, cadastral information), observations collected from sensors (e.g., meteorological stations provided by farmers) as well as other sources of structured and unstructured VGI data (e.g., individual crop production from farmers, statistics at European level, etc.).

Due to the heterogeneity of the data sources and formats, different storage approaches have to be integrated in order to provide an effective, efficient and scalable data storage solution over the time. These combine the use of files (e.g., for the storage of reports in the form of Excel and Word documents) and relational databases.

In addition, the access to these data repositories is done through open and standardized interfaces by using OGC services (i.e., WMS, WFS, WCS and SOS) but also through a set of lightweight APIs developed by FOODIE in order to encapsulate the complexity of the OGC ones and facilitate the access to the resources stored in the platform for the non-expert users.

2 Implementation

Efforts where focused on the establishment of a basic set of various database components and access services (that will be increased enhanced in an iterative manner in the next months) from which to start building the rest of capabilities/features offered by FOODIE platform to its users.

As such, in its current status, the data storage and access services comprise the following components (deployed in various VMs for increased performance).

2.1 Databases

- Postgres-XL: clusterized version of PostgreSQL database, which enables horizontal scalability, being flexible enough to handle varying database workloads. Used (in combination with PostGIS database extension) for storing vector base geospatial information.
- Rasdaman database: an array database system, which provides flexible, fast, scalable geo services for multi-dimensional spatio-temporal sensor, image, simulation, and statistics data of unlimited volume. Currently used to store the satellite imagery (in contrast with the typical storage in the file system)
- Virtuoso: it is a hybrid product that combines the functionality of RDBMS, ORDBMS, XML, RDF, web application server and content management tool. In FOODIE we are mainly interested in the RDF store for the provisioning of a semantic database (triplestore) as a service for the storage of semantic annotations and other RDF data, as well as for their publication as linked data.

For further information about the deployment details of these database components, please refer to deliverable D3.6.2 Deployment and Integration Report.

2.2 Services

- OGC Web Map Service (WMS): The service provides a simple HTTP interface for requesting geo-registered map images from one or more distributed geospatial databases. A WMS request defines the geographic layer(s) and area of interest to be processed. The response to the request is one or more geo-registered map images (returned as JPEG, PNG, etc.) that can be displayed.

  The service is provided through the following open-source software implementations: Mapserver, Geoserver and Rasdaman Petascope WMS plugin.

- OGC Web Feature Service (WFS): The basic Web Feature Service allows querying and retrieval of features. A transactional Web Feature Service (WFS-T) allows creation, deletion, and updating of features.

  The service is provided through the following open-source software implementations: Mapserver, for non-transactional operations and Geoserver for transactional operations – insert, delete, update features in the geospatial database.

- OGC Web Coverage (WCS): provides a client access to coverages – that is, digital geospatial information representing space/time-varying phenomena (typically represented as raster or gridded data, being common for satellite imagery).

  The service is provided through the following open-source software implementations: Mapserver, Geoserver and Rasdaman Petascope WCS plugin.

- Orion Context Broker: It is a FI-WARE Generic Enabler that provides RESTful API to managing context information. Orion provides methods to register and discover context producers and provides mechanism to

notify changes on context information. Orion implements NGSI9/NGSI10 protocols that are used as standards in FI-WARE.

Orion is provided as standalone installation and runs as a backend service daemon.

- OGC Sensor Observation Service (SOS): provides access to sensor data and provides querying and retrieval of observations and sensor system description. The requests and responses are in form of XML documents defined by OGC standard. The OGC SOS service is suitable for any type of sensor systems.

  The service is provided through the following open-source software implementations: MapLog and SensLog.

- Data Management Service: this service allows uploading geospatial information – through its REST interface – to different types of data repositories (e.g., file system, databases) and publishing its associated metadata in an OGC compliant catalogue as well as publishing the data as a new OGC WMS and WFS layer. Besides, the service provides a web user interface, which simplifies for the non-expert user the data upload and layer management process.

  The service is provided through the following open-source software implementations: LayMan (the Layer Manager) and LayMan Client.

For further information about the deployment details of these service components, please refer to deliverable D3.6.2 Deployment and Integration Report.

2.3 Other tools

In addition to the above databases and services, two complementary back office services/tools have been implemented for supporting the maintenance of the spatial repositories:

- Data Harvester component: used for collecting – in a periodic manner - external data sources which are used to feed FOODIE data repositories. The component is deployed on Apache Tomcat server and in its current version is able to download LANDSAT 8 imagery[1] (used in a later stage for performing other operations with them, such as vegetation, water or moisture index extraction) for the whole country areas of Portugal, Spain, Czech Republic and Germany.
- Satellite imagery pre-processing scripts: used for performing (once the LANDSAT-8 imagery is downloaded) some sanitation operations in the raw raster data previous to be usable by the processing services. The scripts are implemented using Python and relies on GRASS software for performing the specialized operations on the satellite imagery:
  - i.landsat.toar → convert DN to top of atmosphere radiance
  - i.colors.enhance (GRASS 7.x) → auto-enhance colors
  - i.landsat.acca → cloud cover assessment

2.4 API details

- OGC Web Map Service (WMS): OpenGIS Web Map Service (WMS) Implementation Specification (http://portal.opengeospatial.org/files/?artifact_id=14416)
- OGC Web Feature Service (WFS): OGC Web Feature Service 2.0 Interface Standard (https://portal.opengeospatial.org/files/09-025r2)

[1] From EarthExplorer: http://earthexplorer.usgs.gov/

- OGC Web Coverage (WCS): OGC WCS 2.0 Interface Standard - Core, version 2.0.1 (https://portal.opengeospatial.org/files/09-110r4)
- OGC Sensor Observation Service (SOS): OGC Sensor Observation Service 1.0 Interface Standard (http://portal.opengeospatial.org/files/?artifact_id=26667)
- Data Harvester: no plans during the first year for having an API
- Data Management Service: See FOODIE Open API specification [2].
- MapLog: See FOODIE Open API specification [2].
- Orion Context Broker: See FOODIE Open API specification [2]

3 Datasets available in FOODIE repositories

- LANDSAT-8 imagery for the Spanish, Czech Republic and German pilot areas
- Statistics from Eurostat in RDF format (virtuoso)
- Road Network Model for all Europe (taken from OSM and made INSPIRE compliant)
- Czech Republic pilot
    - LPIS data
    - Machinery Tracking measurements
    - Weather Stations observations
    - Land user
- Spanish pilot:
    - related to FOODIE Data Model: Treatments, Treatment Plans, Fertilizers, Plots, Issues, Annotations, Crop Production, Prunes
    - Weather Stations observations
- German pilot
    - Cadaster data
    - LPIS data

Annex A - LANDSAT-8 storage details

The Landsat 8 satellite images the entire Earth every 16 days in an 8-day offset from Landsat 7. Data collected by the instruments on-board the satellite are available to download **at no charge** from GloVis (http://glovis.usgs.gov), EarthExplorer (http://earthexplorer.usgs.gov), or via the LandsatLook Viewer (http://landsatlook.usgs.gov) within 24 hours of reception.

Planned parameters for Landsat 8 standard products

- Product type: Level 1T (terrain corrected)
- Output format: GeoTIFF
- Pixel size: 15 meters/30 meters/100 meters (panchromatic/multispectral/thermal)
- Map projection: UTM (Polar Stereographic for Antarctica)
- Datum: WGS 84
- Orientation: North-up (map)
- Resampling: Cubic convolution
- Accuracy:
    - OLI: 12 meters circular error, 90-percent confidence
    - TIRS: 41 meters circular error, 90-percent confidence

OLI Spectral Bands

OLI collects data from nine spectral bands. Seven of the nine bands are consistent with the Thematic Mapper (TM) and Enhanced Thematic Mapper Plus (ETM+) sensors found on earlier Landsat satellites, providing for compatibility with the historical Landsat data, while also improving measurement capabilities. Two new spectral bands, a deep blue coastal / aerosol band and a shortwave-infrared cirrus band, will be collected, allowing scientists to measure water quality and improve detection of high, thin clouds.

Spectral Band	Wavelength	Resolution
Band 1 - Coastal / Aerosol	0.433 - 0.453 µm	30 m
Band 2 - Blue	0.450 - 0.515 µm	30 m
Band 3 - Green	0.525 - 0.600 µm	30 m
Band 4 - Red	0.630 - 0.680 µm	30 m
Band 5 - Near Infrared	0.845 - 0.885 µm	30 m
Band 6 - Short Wavelength Infrared	1.560 - 1.660 µm	30 m
Band 7 - Short Wavelength Infrared	2.100 - 2.300 µm	30 m
Band 8 - Panchromatic	0.500 - 0.680 µm	15 m
Band 9 - Cirrus[2]	1.360 - 1.390 µm	30 m

Table 2: Landsat8 OLI spectral bands

[2] Allows the improved detection of high, thin clouds

Thermal InfraRed Sensor

The Thermal InfraRed Sensor (TIRS), built by the NASA Goddard Space Flight Center, conducts thermal imaging and supports emerging applications such as evapotranspiration rate measurements for water management.

Spectral Band	Wavelength	Resolution
Band 10 - Long Wavelength Infrared	10.30 - 11.30 µm	100 m
Band 11 - Long Wavelength Infrared	11.50 - 12.50 µm	100 m

Table 3: Landsat8 thermal infraRed bands

Derived products

The Landsat program only offers the raw Landsat 8 imagery, implying that any derived product (e.g., vegetation indexes such as VDI, NDVI, etc.) must be calculated by the interested end-user.

Note: It seems that it is possible to access and download some of the vegetation and moisture indexes through Google Earth Engine (https://earthengine.google.org), but it is required to sign in, which is currently limited to organizations in their Trusted Tester program. To apply to be a trusted tester, we would have send a brief description of our organization and how you would like to use Google Earth Engine to: earthengine-beta@google.com.

In principle, most of the vegetation and moisture indexes (NDVI, EVI, NDWI, NDI7, SIWSI, SWIRR, etc.) can be derived from Landsat 8 imagery. However, a series of pre-processing operations should be applied to the raw data previous to the calculations of the indices in order to obtain accurate results.

Among the most relevant ones we can find:

- Convert the Digital Numbers (DNs) to Top-of-Atmosphere Radiances/Reflectances (ToARs)
- (optionally) Correct for atmospheric effects, that is, accounting for distorting atmospheric effects and estimating actual reflectances as they would have been measured on the ground
- (optionally) Assessing cloud covers in order to detect and remove clouds shadows as well

The calculations of all these indexes (as well as the above mentioned pre-processing operations) can be performed manually or automatically using commercial or open source solutions such as ESRI ArcView, GRASS GIS[3] or Orfeo ToolBox, which in some cases have specialized modules and operations to extract specifically most of these indexes from Landsat 8 imagery (see http://courses.neteler.org/processing-landsat8-data-in-grass-gis-7, https://www.orfeo-toolbox.org or https://www.mapbox.com/blog/processing-landsat-8).

3

Storage approaches

There are two approaches for storing and accessing to the satellite imagery: Rasdaman array database and file system plus traditional relational database.

Rasdaman

The following data type has been defined in Rasdaman database in order to store the LANDSAT-8 band information:

```
## definition of landsat8 type (create a lansat8.dl file)
struct Landsat8Pixel {unsigned short coastal, blue, green, red, nir, swir1, swir2, pan, cirrus, tirs1, tirs2; };
typedef marray <Landsat8Pixel, [*:*,*:*]> Landsat8Image;
typedef set <Landsat8Image> Landsat8ImageSet;
```

The data type is inserted in the type definition table using the following command from Rasdaman: "rasdl --basename RASBASE --read landsat8.dl –insert"

The following is an example on how to insert an example landsat 8 image using the rasql command provided by Rasdaman (once the former command has been successfully executed in order to create the LANDSAT-8 definition in the database):

"rasql -q 'insert into LANDSAT values inv_tiff($1, "sampletype=ushort")' -f /usr/local/rasdaman/share/rasdaman/examples/images/landsat_untiled_subset_small_nh.tif --user rasadmin --passwd rasadmin"

File system

Landsat 8 imagery can be downloaded by using "landsat-util", an open source command line utility (https://github.com/developmentseed/landsat-util) that makes it easy to search, download, and process Landsat imagery (in contraposition to the manual methods required by the official sites).

Operations

- **Searching**: Using landsat-api (https://github.com/developmentseed/landsat-api), the tool can search all Landsat-8 metadata and find the images you are looking for. You can limit your search to specific date ranges, filter by cloud coverage, and look within specific rows and paths.
  Landsat-util also makes it easier to find the imagery for a specific area. You can point it to a local shapefile and landsat-util selects all images that cover your shapefile. If you give a country, landsat-util selects all images that cover that country.
- **Downloading**: Landsat-util uses imagery from Google Storage and Amazon AWS to download results faster than USGS Earth Explorer. Both, Google and Amazon, in partnership with USGS and NASA, store raw Landsat imagery on their Google Earth Engine (http://earthengine.google.org) and Amazon AWS (http://aws.amazon.com/public-data-sets/landsat) servers and offers them to the public for free. Landsat-util automatically downloads all of the SceneIDs that fit your search.
  Landsat-util will download a zip file that includes all the bands. You have the option of specifying the bands you want to down. In this case, landsat-util only downloads those bands if they are available online.

- **Image Processing:** Landsat-util can do much of the processing required to make Landsat images useful in a project. It generates natural color images that are ready to be used on mapping tools. All images are adjusted for quality, colour, and contrast, and have incredible details (pansharpening increases pixel resolution 2x). They are WGS84 Web-Mercator (EPSG: 3857) georeferenced and can easily be added as a layer to web-based maps.

Imagery retrieval:

This approach relies on the development of a small bash script. This script will be configured with Ubuntu Cron to run periodically. The script uses a small file with all the path rows for the Iberian Peninsula and using landsat-tools.it searches in the current date for each of the path rows for satellite imagery results.

```
landsat search --start $DATE --end $DATE --pathrow $PATHROW
```

Every match will result on the download of all the bands rasters by using a specific landsat8 id.

```
landsat download $SCENEID --bands 1110987654321
```

In addition, an RGB raster will be generated by combining some of the bands thorugh another landsat-tools command.

```
landsat process /home/atos/landsat/downloads/$SCENEID --bands 432
```

After all this the script will transform each raster to a standard coordinate reference system (EPSG:4326) in order to guarantee homogeneity of the GIS data.

```
gdalwarp -t_srs EPSG:4326 /home/atos/landsat/downloads/$SCENEID/"$SCENEID"_B1.TIF
/home/atos/landsat/downloads/$SCENEID/band1_"$YEAR"_"$MONTH"_"$DAY"_"$PATHROW".tif
```

Once transformation is achieved another GDAL library command will be used to tile each raster into a set of smaller pieces in order to improve WMS or WCS performance.

```
gdal_retile.py -v -levels 1 -ps 2172 1754 -targetDir /data/foodie/landsat/$YEAR/$MONTH/$DAY/$PATHROW/B1/
/home/atos/landsat/downloads/$SCENEID/band1_"$YEAR"_"$MONTH"_"$DAY"_"$PATHROW".tif -tileIndex in-
dex_band1_"$YEAR"_"$MONTH"_"$DAY"_"$PATHROW"
```

At the same time the tiles are stored in a specific set of folders created by the script, this way each tile set is classified by date, path row and band.

Imagery indexation:

With all the imagery downloaded, another bash script will index the tiles using a shapefile:

```
gdaltindex index.shp band_folder/*.tif
```

As a result several files are created:

- Index.dbf
- Index.shp
- Index.prj
- Index.shx

The bash script will use a command to append a new register per tile to database:

```
sudo -u postgres shp2pgsql –a -W LATIN1 -D -I -s 4326 index.shp tablename | sudo -u postgres psql –h virtualmachine6_ip –U foodie-landsat -d foodie-landsat_db
```

Or:

```
ogr2ogr -f PostgreSQL PG:"host=virtualmachine6_ip user=foodie-landsat dbname=foodie-landsat_db password=*******" index.shp;
```

This will add the tile geometry and the file system path to the raster file:

- *FILE_PATH*
- GEOM

Into the corresponding database specific band table:

- Band1
- Band2
- Band3
- Band4
- Band5
- Band6
- Band7
- Band8
- Band9
- Band10
- Band11
- Band432

After that the database will run a specific trigger to calculate bounding box of each register/tile

- *CORNER_UL_LON*
- *CORNER_UR_LAT*
- *CORNER_UR_LON*
- *CORNER_LL_LAT*
- *CORNER_LL_LON*

- *CORNER_LR_LAT*
- *CORNER_LR_LON*

Each table register will be updated by the script to include some more fields by running a simple SQL Update sentence:

- *DATE_ACQUIRED*

Some others are still under discussion:

- CLOUD_COVER
- IMAGE_QUALITY_OLI
- IMAGE_QUALITY_TIRS
- ROLL_ANGLE
- SUN_AZIMUTH
- SUN_ELEVATION
- EARTH_SUN_DISTANCE
- GROUND_CONTROL_POINTS_VERSION
- GROUND_CONTROL_POINTS_MODEL
- GEOMETRIC_RMSE_MODEL
- GEOMETRIC_RMSE_MODEL_Y
- GEOMETRIC_RMSE_MODEL_X
- GROUND_CONTROL_POINTS_VERIFY
- GEOMETRIC_RMSE_VERIFY
- WMS_SERVICE_URL
- WCS_SERVICE_URL

Imagery publication:

Using Mapserver, all band layers are published. To achieve this Mapserver requires the configuration of two separate layers for each band. One is a tile index layer mapping corresponding database band table and the other one is a raster that refers to the tile index layer.

The raster index layer definition includes the Postgis database connection and the SQL query to get the band table contents

```
LAYER
  name raster_tindex
  TYPE polygon
  PROJECTION
    "init=epsg:4326"
  END
  CONNECTIONTYPE POSTGIS
  CONNECTION "host=database.foodie-cloud.org port=5432 dbname=foodie-landsat_db user=foodie-landsat password=******"
  DATA "geom from (SELECT * FROM band432) as subquery using unique id using srid=4326"
END
```

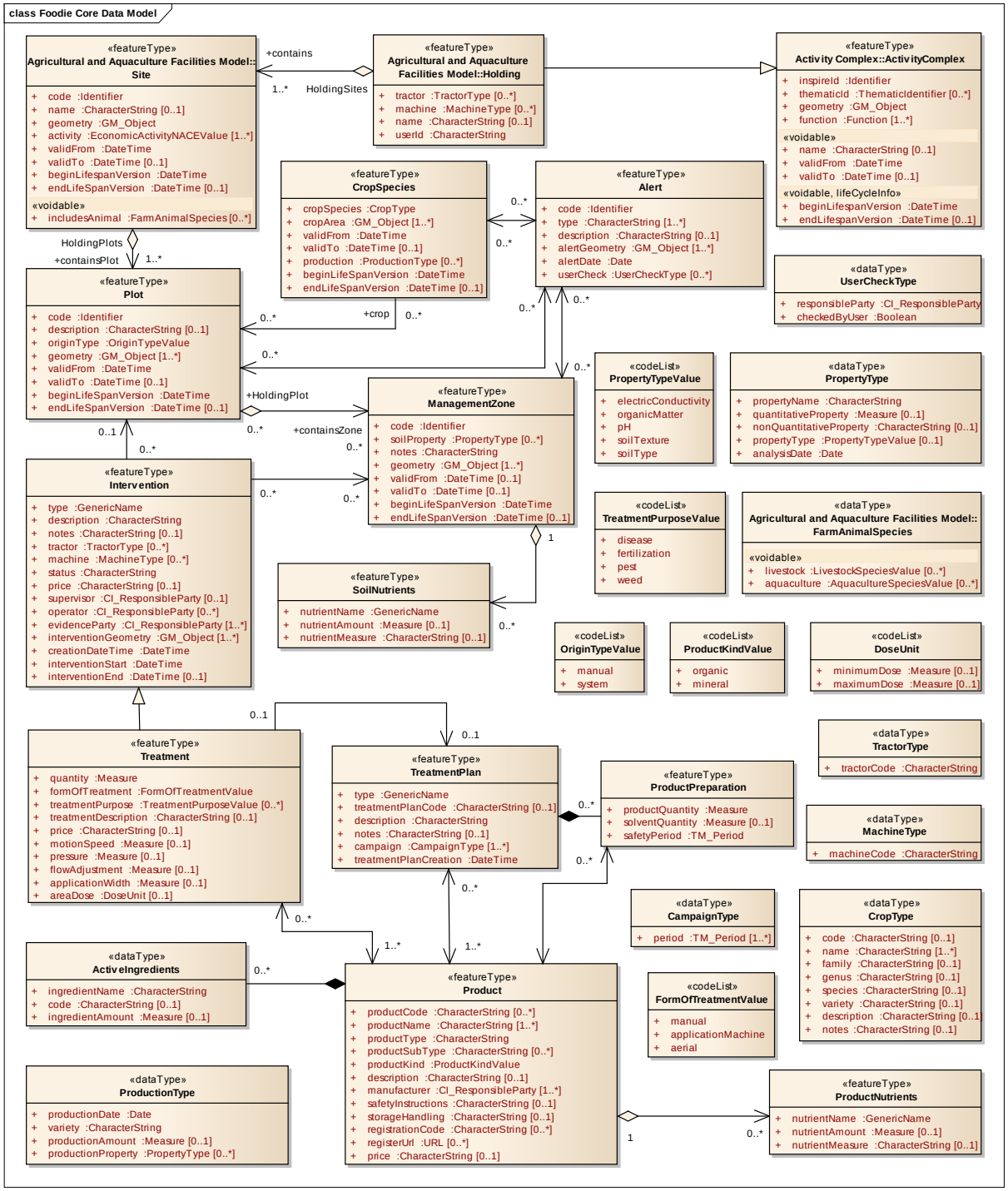
The raster layer includes time dimension definition, using the band tables DATA_ADQUIRED column.

The time dimension extent will be updated by the bash script each time there is new landsat rasters for the specific band.

```
LAYER
  name raster_layer
  TYPE RASTER
  PROJECTION
    "init=epsg:4326"
  END
  METADATA
   “wms_timeextent” “2015-07-01/2016-02-01/P16D”
   “wms_timeitem” “DATE_ADQUIRED”
   “wms_timedafault” “2015-07-01”
  END
  TILEINDEX "raster_tindex"   #THIS NAME MUST MATCH THE TILEINDEX LAYER NAME
  TILEITEM "location"   #not actually needed if column is named location
  STATUS OFF
  OFFSITE 0 0 0
END
```

Annex B - FOODIE Data Model storage details

The following UML class diagram and associated SQL schema (including the extension developed for the Spanish specific needs) shows how FOODIE UML Data Model (please, refer to UML diagram provided in the Information Viewpoint of FOODIE Architecture [3]) has been adapted to Postgres database.



FOODIE CORE DATA MODEL

```
DROP SCHEMA IF EXISTS foodie CASCADE;

CREATE SCHEMA foodie;

SET search_path TO foodie, public;

CREATE TABLE holding (
        holding_id bigint,
        inspire_id_code text,
        inspire_id_code_space text,
        inspire_id_code_version timestamp(0) without time zone,
        geometry geometry,
        holding_name text,
        valid_from timestamp(0) without time zone,
        valid_to timestamp(0) without time zone,
        begin_life_span_version timestamp(0) without time zone,
        end_life_span_version timestamp(0) without time zone,
        user_id text
        ) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE holding ADD CONSTRAINT pk_holding
        PRIMARY KEY (holding_id);

CREATE TABLE holding_thematic_id (
        holding_thematic_id bigint,
        thematic_id text,
        holding_id bigint
        ) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE holding_thematic_id ADD CONSTRAINT pk_holding_thematic_id
        PRIMARY KEY (holding_thematic_id);

CREATE TABLE  holding_function(
        holding_function_id bigint,
        holding_id bigint,
        holding_function text
        ) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE holding_function ADD CONSTRAINT pk_holding_function_id
        PRIMARY KEY (holding_function_id );

CREATE TABLE site(
        site_id bigint,
        code text,
        code_space text,
        code_version timestamp(0) without time zone,
        geometry geometry,
        valid_from timestamp(0) without time zone,
        valid_to timestamp(0) without time zone,
        begin_life_span_version timestamp(0) without time zone,
        end_life_span_version timestamp(0) without time zone,
        holding_id bigint, --contains (aggregation)
        site_name text
        ) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE site ADD CONSTRAINT pk_site
        PRIMARY KEY (site_id);
```

```
CREATE TABLE site_activity(
        site_activity_id bigint,
        site_id bigint,
        economic_activity_nace_value_id bigint
        ) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE site_activity add CONSTRAINT pk_site_activity
        PRIMARY KEY (site_activity_id);

CREATE TABLE economic_activity_nace_value(
        economic_activity_nace_value_id bigint,
        economic_activity_nace_value_name text
        ) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE economic_activity_nace_value add CONSTRAINT pk_economic_activity_nace_value
        PRIMARY KEY (economic_activity_nace_value_id);

CREATE TABLE plot (
        plot_id bigint,
        code text,
        code_space text,
        code_version timestamp(0),
        valid_from timestamp(0)without time zone,
        valid_to timestamp(0)without time zone,
        begin_life_span_version timestamp(0)without time zone,
        end_life_span_version timestamp(0)without time zone,
        geometry geometry,
        description text,
        origin_type_value_id bigint,
        site_id bigint --containsPlot (aggregation)
        ) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE plot ADD CONSTRAINT pk_plot
        PRIMARY KEY (plot_id);

CREATE TABLE origin_type_value (
        origin_type_value_id bigint,
        origin_type_value_name text
        ) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE origin_type_value ADD CONSTRAINT pk_origin_type_value
        PRIMARY KEY (origin_type_value_id);

CREATE TABLE crop_species (
        crop_species_id bigint,
        crop_type_id bigint,
        valid_from timestamp(0)without time zone,
        valid_to timestamp(0)without time zone,
        begin_life_span_version timestamp(0)without time zone,
        end_life_span_version timestamp(0)without time zone,
        crop_area geometry
        --plot_id bigint --speciesPlot
        ) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE crop_species ADD CONSTRAINT pk_crop_species
        PRIMARY KEY (crop_species_id);

CREATE TABLE  crop_species_plot(
        crop_species_plot_id bigint,
```

```
	crop_species_id bigint,
	plot_id bigint
	) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE crop_species_plot ADD CONSTRAINT pk_crop_species_plot
	PRIMARY KEY (crop_species_plot_id);

CREATE TABLE crop_type (
	crop_type_id bigint,
	code text,
	description text,
	crop_type_family text,
	genus text,
	species text,
	notes text,
	variety text
	) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE crop_type ADD CONSTRAINT pk_crop_type
	PRIMARY KEY (crop_type_id);

CREATE TABLE crop_type_name (
	crop_type_name_id bigint,
	crop_type_id bigint,
	crop_type_name text
	) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE crop_type_name ADD CONSTRAINT pk_crop_type_name
	PRIMARY KEY (crop_type_name_id);

CREATE TABLE production_type ( --CHANGED NAME, WAS: production
	production_type_id bigint,
	production_date date,
	variety text,
	production_amount_value double precision,
	production_amount_uom_name text
	--crop_species_id bigint --production in CropSpecies
	) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE production_type ADD CONSTRAINT pk_production_type
	PRIMARY KEY (production_type_id);

CREATE TABLE  production_type_production_property(
	production_type_production_property_id bigint,
	production_type_id bigint,
	property_type_id bigint
	) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE production_type_production_property ADD CONSTRAINT pk_production_type_production_property
	PRIMARY KEY (production_type_production_property_id);

CREATE TABLE  crop_species_production_type(
	crop_species_production_type_id bigint,
	crop_species_id bigint,
	production_type_id bigint
	) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE crop_species_production_type ADD CONSTRAINT pk_crop_species_production_type
	PRIMARY KEY (crop_species_production_type_id);
```

```
CREATE TABLE property_type ( --CHANGED NAME: WAS production_property
        property_type_id bigint,
        property_name text,
        analysis_date date,
        non_guantitative_property text,
        quantitative_property_value double precision,
        quantitative_property_uom_name text,
        property_type_value_id bigint
        ) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE property_type ADD CONSTRAINT pk_property_type
        PRIMARY KEY (property_type_id);


CREATE TABLE alert (
        alert_id bigint,
        code text,
        code_space text,
        code_version timestamp (0) without time zone,
        description text,
        alert_date date,
        alert_geometry geometry
        ) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE alert ADD CONSTRAINT pk_alert
        PRIMARY KEY (alert_id);

CREATE TABLE alert_type (
        alert_type_id bigint,
        alert_id bigint,
        alert_type text
        ) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE alert_type ADD CONSTRAINT pk_alert_type
        PRIMARY KEY (alert_type_id);

CREATE TABLE alert_plot ( --alertPlot, plotAlert
        alert_plot_id bigint,
        alert_id bigint,
        plot_id bigint
        ) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE alert_plot ADD CONSTRAINT pk_alert_plot
        PRIMARY KEY (alert_plot_id);

CREATE TABLE alert_crop_species ( --alertSpecies, speciesAlert
        alert_crop_species_id bigint,
        alert_id bigint,
        crop_species_id bigint
        ) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE alert_crop_species ADD CONSTRAINT pk_alert_crop_species
        PRIMARY KEY (alert_crop_species_id);

CREATE TABLE management_zone(
        management_zone_id bigint,
        code text,
        code_space text,
        code_version timestamp(0) without time zone,
        valid_from timestamp(0) without time zone,
```

```
            valid_to timestamp(0) without time zone,
            begin_life_span_version timestamp(0) without time zone,
            end_life_span_zone timestamp(0) without time zone,
            geometry geometry,
            notes text,
            plot_id bigint --containsZone (aggregation)
            ) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE management_zone ADD CONSTRAINT pk_management_zone
            PRIMARY KEY (management_zone_id);

CREATE TABLE management_zone_soil_property (
            management_zone_soil_property_id bigint,
            management_zone_id bigint,
            property_type_id bigint
            ) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE management_zone_soil_property ADD CONSTRAINT pk_management_zone_soil_property
            PRIMARY KEY (management_zone_soil_property_id);

CREATE TABLE alert_management_zone ( --alertZone, zoneAlert
            alert_management_zone_id bigint,
            alert_id bigint,
            management_zone_id bigint
            ) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE alert_management_zone ADD CONSTRAINT pk_alert_management_zone
            PRIMARY KEY (alert_management_zone_id);

CREATE TABLE intervention
( intervention_id bigint,
  description text,
  notes text,
  price text,
  status text,
  creation_date_time timestamp(0) without time zone,
  intervention_start timestamp(0) without time zone,
  intervention_end timestamp(0) without time zone,
  intervention_geometry geometry,
  intervention_type_value_id bigint,
  plot_id bigint --interventionPlot
  ) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE intervention ADD CONSTRAINT pk_intervention
            PRIMARY KEY (intervention_id);

CREATE TABLE treatment
( treatment_id bigint,
  price text,
  quantity_value double precision,
  quantity_uom_name text,
  motion_speed_value double precision,
  motion_speed_oum_name text,
  pressure_value double precision,
  pressure_uom_name text,
  flow_adjustment_value double precision,
  flow_adjustment_uom_name text,
  application_width_value double precision,
  application_width_uom_name text,
  area_dose_minimum_value double precision,
```

```
 area_dose_minimum_uom_name text,
 area_dose_maximum_value double precision,
 area_dose_maximum_uom_name text,
 form_of_treatment_value_id bigint,
 treatment_plan_id bigint, --plan
 treatment_description text,
 intervention_id bigint not null
 ) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE treatment ADD CONSTRAINT pk_treatment
        PRIMARY KEY (treatment_id);

CREATE TABLE tractor_type ( --NAME CHANGED WAS: tractor_id
        tractor_type_id bigint,
        tractor_code text
        ) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE tractor_type ADD CONSTRAINT pk_tractor_type
        PRIMARY KEY (tractor_type_id);

CREATE TABLE machine_type ( --NAME CHANGED WAS: machine_id
        machine_type_id bigint,
        machine_code text
        );--DISTRIBUTE BY REPLICATION;

ALTER TABLE machine_type ADD CONSTRAINT pk_machine_type
        PRIMARY KEY (machine_type_id);

CREATE TABLE intervention_machine ( --NAME CHANGED WAS: machine_id
        intervention_machine_id bigint,
        intervention_id bigint,
        machine_type_id bigint
        );--DISTRIBUTE BY REPLICATION;

ALTER TABLE intervention_machine ADD CONSTRAINT pk_intervention_machine
        PRIMARY KEY (intervention_machine_id);

CREATE TABLE holding_machine ( --NAME CHANGED WAS: machine_id
        holding_machine_id bigint,
        holding_id bigint,
        machine_type_id bigint
        );--DISTRIBUTE BY REPLICATION;

ALTER TABLE holding_machine ADD CONSTRAINT pk_holding_machine
        PRIMARY KEY (holding_machine_id);

CREATE TABLE intervention_tractor ( --NAME CHANGED WAS: machine_id
        intervention_tractor_id bigint,
        intervention_id bigint,
        tractor_type_id bigint
        );--DISTRIBUTE BY REPLICATION;

ALTER TABLE intervention_tractor ADD CONSTRAINT pk_intervention_tractor
        PRIMARY KEY (intervention_tractor_id);

CREATE TABLE holding_tractor ( --NAME CHANGED WAS: machine_id
        holding_tractor_id bigint,
        holding_id bigint,
        tractor_type_id bigint
        );--DISTRIBUTE BY REPLICATION;
```

```
ALTER TABLE holding_tractor ADD CONSTRAINT pk_holding_tractor
        PRIMARY KEY (holding_tractor_id);

CREATE TABLE intervention_management_zone ( --NAME CHANGED WAS: machine_id
        intervention_management_zone_id bigint,
        management_zone_id bigint,
        intervention_id bigint
        );--DISTRIBUTE BY REPLICATION;

ALTER TABLE intervention_management_zone ADD CONSTRAINT pk_intervention_management_zone
        PRIMARY KEY (intervention_management_zone_id);

CREATE TABLE product(
        product_id bigint,
        product_kind_value_id bigint,
        description text,
        safety_instructions text,
        storage_handling text,
        product_type text,
        price text
        ) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE product ADD CONSTRAINT pk_product
        PRIMARY KEY (product_id);

CREATE TABLE product_code(
        product_code_id bigint,
        product_id bigint,
        product_code text
        ) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE product_code ADD CONSTRAINT pk_product_code
        PRIMARY KEY (product_code_id);

CREATE TABLE product_name(
        product_name_id bigint,
        product_id bigint,
        product_name text
        ) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE product_name ADD CONSTRAINT pk_product_name
        PRIMARY KEY (product_name_id);

CREATE TABLE product_sub_type(
        product_sub_type_id bigint,
        product_id bigint,
        product_sub_type text
        ) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE product_sub_type ADD CONSTRAINT pk_product_sub_type
        PRIMARY KEY (product_sub_type_id);

CREATE TABLE product_register_url(
        product_register_url_id bigint,
        product_id bigint,
        register_url text
        ) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE product_register_url ADD CONSTRAINT pk_product_register_url
```

```
        PRIMARY KEY (product_register_url_id);

CREATE TABLE product_registration_code(
        product_registration_code_id bigint,
        product_id bigint,
        registration_code text
        ) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE product_registration_code ADD CONSTRAINT pk_product_registration_code
        PRIMARY KEY (product_registration_code_id);

CREATE TABLE product_treatment( --productTreatment, treatmentProduct
        product_treatment_id bigint,
        product_id bigint,
        treatment_id bigint
        ) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE product_treatment ADD CONSTRAINT pk_product_treatment
        PRIMARY KEY (product_treatment_id);

CREATE TABLE treatment_plan(
        treatment_plan_id bigint,
        treatment_plan_code text,
        description text,
        treatment_plan_creation timestamp(0) without time zone,
        notes text,
        treatment_plan_type_value_id bigint
        ) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE treatment_plan ADD CONSTRAINT pk_treatment_plan
        PRIMARY KEY (treatment_plan_id);

CREATE TABLE campaign_type(
        campaign_type_id bigint,
        campaign_begin timestamp(0) without time zone,
        campaign_end timestamp(0) without time zone
        ) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE campaign_type ADD CONSTRAINT pk_campaign
        PRIMARY KEY (campaign_type_id);

CREATE TABLE treatment_plan_campaign(
        treatment_plan_campaign_id bigint,
        treatment_plan_id bigint,
        campaign_type_id bigint
        ) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE treatment_plan_campaign ADD CONSTRAINT pk_treatment_plan_campaign
        PRIMARY KEY (treatment_plan_campaign_id);

CREATE TABLE product_treatment_plan( --productPlan, planProduct
        product_treatment_plan_id bigint,
        product_id bigint,
        treatment_plan_id bigint
        ) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE product_treatment_plan ADD CONSTRAINT pk_product_treatment_plan
        PRIMARY KEY (product_treatment_plan_id);

CREATE TABLE product_preparation(
```

```
        product_preparation_id bigint,
        product_quantity_value double precision,
        product_quantity_uom_name text,
        solvent_quantity_value double precision,
        solvent_quantity_uom_name text,
        safety_period_begin timestamp(0) without time zone,
        safety_period_end timestamp(0) without time zone,
        treatment_plan_id bigint, --preparationPlan (aggregation->special)
        product_id bigint
        ) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE product_preparation ADD CONSTRAINT pk_product_preparation_id
        PRIMARY KEY (product_preparation_id);


CREATE TABLE soil_nutrients(
        soil_nutrients_id bigint,
        management_zone_id bigint, --zoneNutrients (aggregation)
        nutrient_amount_value double precision,
        nutrient_amount_uom_name text,
        nutrients_measure text,
        nutrient_name text
        ) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE soil_nutrients ADD CONSTRAINT pk_soil_nutrients
        PRIMARY KEY (soil_nutrients_id);

CREATE TABLE product_nutrients(
        product_nutrients_id bigint,
        product_id bigint, --nutrient (aggregation)
        nutrient_amount_value double precision,
        nutrient_amount_uom_name text,
        nutrients_measure text,
        nutrient_name text
        ) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE product_nutrients ADD CONSTRAINT pk_product_nutrients
        PRIMARY KEY (product_nutrients_id);

CREATE TABLE form_of_treatment_value( --for treatment
        form_of_treatment_value_id bigint,
        form_of_treatment_value_name text
        ) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE form_of_treatment_value ADD CONSTRAINT pk_form_of_treatment_value
        PRIMARY KEY (form_of_treatment_value_id);

CREATE TABLE treatment_purpose ( --for treatment CHANGED NAME WAS treatment_purpose
        treatment_purpose_id bigint,
        treatment_id bigint,
        treatment_purpose_value_id bigint
        ) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE treatment_purpose ADD CONSTRAINT pk_treatment_purpose
        PRIMARY KEY (treatment_purpose_id);

CREATE TABLE treatment_purpose_value(
        treatment_purpose_value_id bigint,
        treatment_purpose_value_name text
        ) ;--DISTRIBUTE BY REPLICATION;
```

```
ALTER TABLE treatment_purpose_value ADD CONSTRAINT pk_treatment_purpose_value
        PRIMARY KEY (treatment_purpose_value_id);

CREATE TABLE product_kind_value( --for product
        product_kind_value_id bigint,
        product_kind_value_name text
        ) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE product_kind_value add CONSTRAINT pk_product_kind_value
        PRIMARY KEY (product_kind_value_id);

CREATE TABLE property_type_value( --for property_type
        property_type_value_id bigint,
        property_type_value_name text
        ) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE property_type_value add CONSTRAINT pk_property_type_value
        PRIMARY KEY (property_type_value_id);

CREATE TABLE cl_responsible_party(
        cl_responsible_party_id bigint,
        cl_contact_id bigint,
        individual_name text,
        organisation_name text,
        position_name text,
        cl_responsible_party_role_id bigint
        ) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE cl_responsible_party add CONSTRAINT pk_cl_responsible_party
        PRIMARY KEY (cl_responsible_party_id);

CREATE TABLE cl_contact(
        cl_contact_id bigint,
        cl_address_id bigint,
        contact_instructions text,
        hours_of_service text,
        cl_online_resource_id bigint,
        cl_phone_id bigint
        ) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE cl_contact add CONSTRAINT pk_cl_contact
        PRIMARY KEY (cl_contact_id);

CREATE TABLE cl_telephone(
        cl_telephone_id bigint
        ) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE cl_telephone add CONSTRAINT pk_cl_telephone
        PRIMARY KEY (cl_telephone_id);

CREATE TABLE facsimile(
        facsimile_id bigint,
        cl_telephone_id bigint,
        facsimile  text
        ) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE facsimile add CONSTRAINT pk_facsimile
        PRIMARY KEY (facsimile_id);
```

```
CREATE TABLE voice(
        voice_id bigint,
        cl_telephone_id bigint,
        voice  text
        ) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE voice add CONSTRAINT pk_voice
        PRIMARY KEY (voice_id);

CREATE TABLE cl_address(
        cl_address_id bigint,
        administrative_area text,
        city text,
        country text,
        postal_code text
        ) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE cl_address add CONSTRAINT pk_cl_address
        PRIMARY KEY (cl_address_id);

CREATE TABLE delivery_point(
        delivery_point_id bigint,
        cl_address_id bigint,
        delivery_point text
        ) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE delivery_point add CONSTRAINT pk_delivery_point
        PRIMARY KEY (delivery_point_id);

CREATE TABLE electronic_mail_address(
        electronic_mail_adress_id bigint,
        cl_address_id bigint,
        electronic_mail_adress text
        ) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE electronic_mail_address add CONSTRAINT pk_electronic_mail_adress
        PRIMARY KEY (electronic_mail_adress_id);

CREATE TABLE cl_online_resource(
        cl_online_resource_id bigint,
        description text,
        cl_online_function_code_id bigint,
        linkage text,
        application_profile text,
        cl_online_resource_name text,
        protocol text
        ) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE cl_online_resource add CONSTRAINT pk_cl_conline_resource
        PRIMARY KEY (cl_online_resource_id);

CREATE TABLE cl_online_function_code(
        cl_online_function_code_id bigint,
        cl_online_function_code_name text
        ) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE cl_online_function_code add CONSTRAINT pk_cl_online_function_code
        PRIMARY KEY (cl_online_function_code_id);

CREATE TABLE cl_role_code(
```

```
	cl_role_code_id bigint,
	cl_role_code_name text
	) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE cl_role_code add CONSTRAINT pk_cl_role_code
	PRIMARY KEY (cl_role_code_id);

CREATE TABLE intervention_supervisor(
	supervisor_id bigint,
	intervention_id bigint,
	cl_responsible_party_id bigint
	) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE intervention_supervisor add CONSTRAINT pk_intervention_supervisor
	PRIMARY KEY (supervisor_id);

CREATE TABLE intervention_evidence_party(
	evidence_party_id bigint,
	intervention_id bigint,
	cl_responsible_party_id bigint
	) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE intervention_evidence_party add CONSTRAINT pk_intervention_evidence_party
	PRIMARY KEY (evidence_party_id);

CREATE TABLE intervention_operator(
	operator_id bigint,
	intervention_id bigint,
	cl_responsible_party_id bigint
	) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE intervention_operator add CONSTRAINT pk_intervention_operator
	PRIMARY KEY (operator_id);

CREATE TABLE product_manufacturer (
	product_manufacturer_id bigint,
	product_id bigint,
	cl_responsible_party_id bigint
	) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE product_manufacturer add CONSTRAINT pk_product_manufacturer
	PRIMARY KEY (product_manufacturer_id);

CREATE TABLE active_ingredients (
	active_ingredients_id bigint,
	code text,
	code_space text,
	code_version timestamp(0) without time zone,
	ingredient_name text,
	ingredient_amount_value double precision,
	ingredient_amount_uom_name text,
	product_id bigint --ingredientProduct (aggregation->special)
	) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE active_ingredients add CONSTRAINT PK_active_ingredients
	PRIMARY KEY (active_ingredients_id);

CREATE TABLE user_check_type (
	user_check_type_id bigint,
	alert_id bigint,
```

```
	cl_responsible_party_id bigint,
	checked_by_user boolean

	) ;--DISTRIBUTE BY REPLICATION;


ALTER TABLE user_check_type add CONSTRAINT PK_user_check_type
	PRIMARY KEY (user_check_type_id);

CREATE TABLE treatment_plan_type_value (
	treatment_plan_type_value_id bigint,
	treatment_plan_type_value_name text

	) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE treatment_plan_type_value add CONSTRAINT PK_treatment_plan_type_value
	PRIMARY KEY (treatment_plan_type_value_id);

CREATE TABLE intervention_type_value (
	intervention_type_value_id bigint,
	intervention_type_value_name text

	) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE intervention_type_value add CONSTRAINT PK_intervention_type_value_id
	PRIMARY KEY (intervention_type_value_id);

ALTER TABLE site
 ADD CONSTRAINT fk_site__holding FOREIGN KEY (holding_id)
    REFERENCES holding (holding_id);

ALTER TABLE site_activity
	ADD CONSTRAINT fk_site_activity__site FOREIGN KEY (site_id)
    REFERENCES site(site_id);

ALTER TABLE site_activity
	ADD CONSTRAINT fk_site_activity__economic_activity_nace_value FOREIGN KEY (economic_activity_nace_value_id)
    REFERENCES economic_activity_nace_value(economic_activity_nace_value_id);

ALTER TABLE plot
 ADD CONSTRAINT fk_plot__site FOREIGN KEY (site_id)
    REFERENCES site (site_id);

ALTER TABLE plot
	ADD CONSTRAINT fk_plot__origin_type_value FOREIGN KEY (origin_type_value_id)
		REFERENCES origin_type_value (origin_type_value_id);


ALTER TABLE crop_species_plot
	ADD CONSTRAINT fk_crop_species_plot__crop_species FOREIGN KEY (crop_species_id)
		REFERENCES crop_species (crop_species_id);

ALTER TABLE crop_species_plot
	ADD CONSTRAINT fk_crop_species_plot__plot FOREIGN KEY (plot_id)
		REFERENCES plot (plot_id);

ALTER TABLE crop_species_production_type
 ADD CONSTRAINT fk_crop_species_production_type__crop_species FOREIGN KEY (crop_species_id)
    REFERENCES crop_species (crop_species_id);
```

```
ALTER TABLE crop_species_production_type
 ADD CONSTRAINT fk_crop_species_production_type__production_type FOREIGN KEY (production_type_id)
    REFERENCES production_type (production_type_id);

ALTER TABLE crop_species
 ADD CONSTRAINT fk_crop_species__crop_type FOREIGN KEY (crop_type_id)
    REFERENCES crop_type (crop_type_id);

ALTER TABLE crop_type_name
 ADD CONSTRAINT fk_crop_type_name__crop_type FOREIGN KEY (crop_type_id)
    REFERENCES crop_type (crop_type_id);

-- ALTER TABLE production_property
--  ADD CONSTRAINT fk_production_property__productin FOREIGN KEY (production_id)
--     REFERENCES production (production_id);

ALTER TABLE alert_plot
 ADD CONSTRAINT fk_alert_plot__alert FOREIGN KEY (alert_id)
    REFERENCES alert (alert_id);

ALTER TABLE alert_plot
 ADD CONSTRAINT fk_alert_plot__plot FOREIGN KEY (plot_id)
    REFERENCES plot(plot_id);

ALTER TABLE alert_crop_species
 ADD CONSTRAINT fk_alert_crop_species__crop_species FOREIGN KEY (crop_species_id)
    REFERENCES crop_species(crop_species_id);

ALTER TABLE alert_crop_species
 ADD CONSTRAINT fk_alert_crop_species__alert FOREIGN KEY (alert_id)
    REFERENCES alert(alert_id);

ALTER TABLE management_zone
            ADD CONSTRAINT fk_management_zone__plot FOREIGN KEY (plot_id)
    REFERENCES plot(plot_id);

ALTER TABLE         management_zone_soil_property
            ADD CONSTRAINT fk_management_zone_soil_property__management_zone FOREIGN KEY (management_zone_id)
    REFERENCES management_zone(management_zone_id);

ALTER TABLE alert_management_zone
 ADD CONSTRAINT fk_alert_management_zones__management_zone FOREIGN KEY (management_zone_id)
    REFERENCES management_zone(management_zone_id);

ALTER TABLE alert_management_zone
 ADD CONSTRAINT fk_alert_management_zone__alert FOREIGN KEY (alert_id)
    REFERENCES alert(alert_id);

ALTER TABLE intervention
 ADD CONSTRAINT fk_intervention_treatement__plot FOREIGN KEY (plot_id)
    REFERENCES plot (plot_id);

ALTER TABLE product_code
 ADD CONSTRAINt fk_product_code__product FOREIGN KEY (product_id)
    REFERENCES product (product_id);

ALTER TABLE product_name
 ADD CONSTRAINt fk_product_name__product FOREIGN KEY (product_id)
    REFERENCES product (product_id);
```

```
ALTER TABLE product_sub_type
 ADD CONSTRAINt fk_product_sub_type__product FOREIGN KEY (product_id)
    REFERENCES product (product_id);

ALTER TABLE product_register_url
 ADD CONSTRAINt fk_product_register_url__product FOREIGN KEY (product_id)
    REFERENCES product (product_id);

ALTER TABLE product_registration_code
 ADD CONSTRAINt fk_product_registration_code__product FOREIGN KEY (product_id)
    REFERENCES product (product_id);

ALTER TABLE product_treatment
 ADD CONSTRAINt fk_product_treatment__treatment FOREIGN KEY (treatment_id)
    REFERENCES treatment (treatment_id);

ALTER TABLE product_treatment
 ADD CONSTRAINt fk_product_intervention_treatement__product FOREIGN KEY (product_id)
    REFERENCES product (product_id);

 ALTER TABLE treatment_plan_campaign
 ADD CONSTRAINt fk_treatment_plan_campaignn__treatment_plan FOREIGN KEY (treatment_plan_id)
    REFERENCES treatment_plan (treatment_plan_id);

 ALTER TABLE treatment_plan_campaign
 ADD CONSTRAINt fk_treatment_plan_campaignn__campaign_type FOREIGN KEY (campaign_type_id)
    REFERENCES campaign_type (campaign_type_id);

 ALTER TABLE product_treatment_plan
 ADD CONSTRAINt fk_product_treatment_plan__product FOREIGN KEY (product_id)
    REFERENCES product (product_id);

 ALTER TABLE product_treatment_plan
 ADD CONSTRAINt fk_product_treatment_plan__treatment_plan FOREIGN KEY (treatment_plan_id)
    REFERENCES treatment_plan (treatment_plan_id);

 ALTER TABLE treatment
 ADD CONSTRAINt fk_treatment__treatment_plan FOREIGN KEY (treatment_plan_id)
    REFERENCES treatment_plan (treatment_plan_id);

 ALTER TABLE product_preparation
 ADD CONSTRAINt fk_product_preparation__treatment_plan FOREIGN KEY (treatment_plan_id)
    REFERENCES treatment_plan (treatment_plan_id);

 ALTER TABLE product_preparation
 ADD CONSTRAINt fk_product_preparation__product FOREIGN KEY (product_id)
    REFERENCES product (product_id);

 ALTER TABLE soil_nutrients
 ADD CONSTRAINt fk_soil_nutrients__management_zone FOREIGN KEY (management_zone_id)
    REFERENCES management_zone (management_zone_id);

 ALTER TABLE product_nutrients
 ADD CONSTRAINt fk_product_nutrients__product FOREIGN KEY (product_id)
    REFERENCES product (product_id);

ALTER TABLE treatment
        ADD CONSTRAINT fk_treatment__form_of_treatment_value FOREIGN KEY (form_of_treatment_value_id)
    REFERENCES form_of_treatment_value(form_of_treatment_value_id);
```

```
ALTER TABLE treatment_purpose
        ADD CONSTRAINT fk_treatment_purpose__treatment_purpose_value FOREIGN KEY (treatment_purpose_value_id)
    REFERENCES treatment_purpose_value(treatment_purpose_value_id);

ALTER TABLE treatment_purpose
        ADD CONSTRAINT fk_treatment_purpose__treatment FOREIGN KEY (treatment_id)
    REFERENCES treatment(treatment_id);

ALTER TABLE product
        ADD CONSTRAINT fk_product__product_kind_value FOREIGN KEY (product_kind_value_id)
    REFERENCES product_kind_value(product_kind_value_id);

ALTER TABLE management_zone_soil_property
        ADD CONSTRAINT fk_management_zone_soil_property__property_type_value FOREIGN KEY (property_type_id)
    REFERENCES property_type(property_type_id);

ALTER TABLE property_type
        ADD CONSTRAINT fk_property_type__property_type_value FOREIGN KEY (property_type_value_id)
    REFERENCES property_type_value(property_type_value_id);

ALTER TABLE cl_responsible_party
        ADD CONSTRAINT fk_cl_responsible_party__cl_role_code FOREIGN KEY (cl_responsible_party_role_id)
    REFERENCES cl_role_code(cl_role_code_id);

ALTER TABLE cl_responsible_party
        ADD CONSTRAINT fk_cl_responsible_party__cl_contact FOREIGN KEY (cl_contact_id)
    REFERENCES cl_contact(cl_contact_id);

ALTER TABLE cl_contact
        ADD CONSTRAINT fk_cl_contact__cl_online_resource FOREIGN KEY (cl_online_resource_id)
    REFERENCES cl_online_resource(cl_online_resource_id);

ALTER TABLE cl_contact
        ADD CONSTRAINT fk_cl_contact__cl_address FOREIGN KEY (cl_address_id)
    REFERENCES cl_address(cl_address_id);

ALTER TABLE cl_contact
        ADD CONSTRAINT fk_cl_contact__cl_telephone FOREIGN KEY (cl_phone_id)
    REFERENCES cl_telephone(cl_telephone_id);

ALTER TABLE cl_online_resource
        ADD CONSTRAINT fk_cl_online_resource__cl_online_function_code FOREIGN KEY (cl_online_function_code_id)
    REFERENCES cl_online_function_code(cl_online_function_code_id);

ALTER TABLE intervention_supervisor
        ADD CONSTRAINT fk_intervention_supervisor__cl_responsible_party FOREIGN KEY (cl_responsible_party_id)
    REFERENCES cl_responsible_party(cl_responsible_party_id);

ALTER TABLE intervention_supervisor
        ADD CONSTRAINT fk_intervention_supervisor__intervention FOREIGN KEY (intervention_id)
    REFERENCES intervention(intervention_id);

ALTER TABLE intervention_evidence_party
        ADD CONSTRAINT fk_intervention_evidence_party__cl_responsible_party FOREIGN KEY (cl_responsible_party_id)
    REFERENCES cl_responsible_party(cl_responsible_party_id);

ALTER TABLE intervention_evidence_party
        ADD CONSTRAINT fk_intervention_evidence_party__intervention FOREIGN KEY (intervention_id)
    REFERENCES intervention(intervention_id);
```

```
ALTER TABLE intervention_operator
        ADD CONSTRAINT fk_intervention_operator__cl_responsible_party FOREIGN KEY (cl_responsible_party_id)
    REFERENCES cl_responsible_party(cl_responsible_party_id);

ALTER TABLE intervention_operator
        ADD CONSTRAINT fk_intervention_operator__intervention FOREIGN KEY (intervention_id)
    REFERENCES intervention(intervention_id);

ALTER TABLE product_manufacturer
        ADD CONSTRAINT fk_product_manufacturer__product FOREIGN KEY (product_id)
    REFERENCES product(product_id);

ALTER TABLE product_manufacturer
        ADD CONSTRAINT fk_product_manufacturer__cl_responsible_party FOREIGN KEY (cl_responsible_party_id)
    REFERENCES cl_responsible_party(cl_responsible_party_id);

ALTER TABLE delivery_point
        ADD CONSTRAINT fk_delivery_point__cl_address FOREIGN KEY (cl_address_id)
    REFERENCES cl_address(cl_address_id);

ALTER TABLE electronic_mail_address
        ADD CONSTRAINT fk_electronic_mail_address__cl_address FOREIGN KEY (cl_address_id)
    REFERENCES cl_address(cl_address_id);

ALTER TABLE facsimile
        ADD CONSTRAINT fk_facsimile__cl_telephone FOREIGN KEY (cl_telephone_id)
    REFERENCES cl_telephone(cl_telephone_id);

ALTER TABLE voice
        ADD CONSTRAINT fk_voice__cl_telephone FOREIGN KEY (cl_telephone_id)
    REFERENCES cl_telephone(cl_telephone_id);

ALTER TABLE holding_thematic_id
        ADD CONSTRAINT fk_holding_thematic_id__holding FOREIGN KEY (holding_id)
    REFERENCES holding(holding_id);

ALTER TABLE alert_type
        ADD CONSTRAINT fk_alert_type__alert FOREIGN KEY (alert_id)
    REFERENCES alert(alert_id);

ALTER TABLE holding_function
        ADD CONSTRAINT fk_holding_function__holding FOREIGN KEY (holding_id)
    REFERENCES holding(holding_id);

ALTER TABLE active_ingredients
        ADD CONSTRAINT fk_active_ingredients__product FOREIGN KEY (product_id)
    REFERENCES product(product_id);

ALTER TABLE production_type_production_property
        ADD CONSTRAINT fk_production_type_property_type__production_type FOREIGN KEY (production_type_id)
    REFERENCES production_type(production_type_id);

ALTER TABLE production_type_production_property
        ADD CONSTRAINT fk_production_type_property_type__property_type FOREIGN KEY (property_type_id)
    REFERENCES property_type(property_type_id);

ALTER TABLE intervention_management_zone
        ADD CONSTRAINT intervention_management_zone__intervention FOREIGN KEY (intervention_id)
    REFERENCES intervention(intervention_id);
```

```
ALTER TABLE intervention_management_zone
        ADD CONSTRAINT intervention_management_zone__management_zone FOREIGN KEY (management_zone_id)
    REFERENCES management_zone(management_zone_id);

ALTER TABLE user_check_type
        ADD CONSTRAINT user_check_type__alert FOREIGN KEY (alert_id)
    REFERENCES alert(alert_id);

ALTER TABLE user_check_type
        ADD CONSTRAINT checked_by_user__cl_responsible_party FOREIGN KEY (cl_responsible_party_id)
    REFERENCES cl_responsible_party(cl_responsible_party_id);

ALTER TABLE treatment_plan
        ADD CONSTRAINT treatment_plan__treatment_plan_type_value FOREIGN KEY (treatment_plan_type_value_id)
    REFERENCES treatment_plan_type_value(treatment_plan_type_value_id);

ALTER TABLE treatment
        ADD CONSTRAINT treatment__intervention FOREIGN KEY (intervention_id)
    REFERENCES intervention(intervention_id);

ALTER TABLE intervention
        ADD CONSTRAINT intervention__intervention_type_value FOREIGN KEY (intervention_type_value_id)
    REFERENCES intervention_type_value(intervention_type_value_id);

ALTER TABLE intervention_tractor
        ADD CONSTRAINT intervention_tractor__tractor_type FOREIGN KEY (tractor_type_id)
    REFERENCES tractor_type(tractor_type_id);

ALTER TABLE intervention_tractor
        ADD CONSTRAINT intervention_tractor__intervention FOREIGN KEY (intervention_id)
    REFERENCES intervention(intervention_id);

ALTER TABLE holding_tractor
        ADD CONSTRAINT holding_tractor__tractor_type FOREIGN KEY (tractor_type_id)
    REFERENCES tractor_type(tractor_type_id);

ALTER TABLE holding_tractor
        ADD CONSTRAINT holding_tractor__holding FOREIGN KEY (holding_id)
    REFERENCES holding(holding_id);

ALTER TABLE intervention_machine
        ADD CONSTRAINT intervention_machine__machine_type FOREIGN KEY (machine_type_id)
    REFERENCES machine_type(machine_type_id);

ALTER TABLE intervention_machine
        ADD CONSTRAINT intervention_machine__intervention FOREIGN KEY (intervention_id)
    REFERENCES intervention(intervention_id);

ALTER TABLE holding_machine
        ADD CONSTRAINT holding_machine__machine_type FOREIGN KEY (machine_type_id)
    REFERENCES machine_type(machine_type_id);

ALTER TABLE holding_machine
        ADD CONSTRAINT holding_machine__holding FOREIGN KEY (holding_id)
    REFERENCES holding(holding_id);
```

SPANISH PILOT EXTENSION TO FOODIE CORE DATA MODEL

```
SET search_path TO foodie, public;

alter table site add column main_site boolean;

alter table site add column cl_address_id bigint;

COMMENT ON COLUMN site.cl_address_id IS 'location';


ALTER TABLE site
        ADD CONSTRAINT fk_site__cl_address FOREIGN KEY (cl_address_id)
        REFERENCES      cl_address(cl_address_id);

alter table  management_zone add column management_zone_name text;

alter table  management_zone add column origin_type_value_id bigint;

ALTER TABLE management_zone
        ADD CONSTRAINT fk_management_zone__origin_type_value FOREIGN KEY (origin_type_value_id)
        REFERENCES      origin_type_value(origin_type_value_id);

alter table  management_zone add column crop_type_id bigint;

ALTER TABLE management_zone
        ADD CONSTRAINT fk_management_zone__crop_type FOREIGN KEY (crop_type_id)
        REFERENCES crop_type(crop_type_id);

alter table  campaign_type add column holding_id bigint;

ALTER TABLE campaign_type
        ADD CONSTRAINT fk_campaign_type__holding FOREIGN KEY (holding_id)
        REFERENCES holding(holding_id);

CREATE TABLE site_responsible_party(
        site_responsible_party_id bigint,
        site_id bigint,
        cl_responsible_party_id bigint
        ) ;--DISTRIBUTE BY REPLICATION;

ALTER TABLE site_responsible_party ADD CONSTRAINT pk_site_responsible_party
        PRIMARY KEY (site_responsible_party_id);

ALTER TABLE site_responsible_party
        ADD CONSTRAINT fk_site_responsible_party__site FOREIGN KEY (site_id)
    REFERENCES site(site_id);

ALTER TABLE site_responsible_party
        ADD CONSTRAINT fk_site_responsible_party__cl_responsible_party FOREIGN KEY (cl_responsible_party_id)
        REFERENCES      cl_responsible_party(cl_responsible_party_id);

alter table product_name add column country_code text;
alter table product_code add column country_code text;
alter table product_register_url add column country_code text;
alter table product_registration_code add column country_code text;
```

References

	References
01	Palma R., et al "Deployment and Integration Report". D3.6.2 FOODIE Deliverable. February 2016
02	Suarez I., et al "D3.2.2. Open API specification". D3.2.2. Deliverable. February 2016
03	Esbrí M., et al "D2.2.2 Platform Specification Report". D2.2.2 Deliverable. February 2016

Table 4: References