Patty / Mapping the Via Appia in 3D

Via Appia 3D GIS: Software User Manual

E. Ranguelova
R. Goncalves
R. van Haren
O. Martinez-Rubi
Netherlands eScience Center,
Science Park 140 (Matrix 1), 1098 XG Amsterdam, the Netherlands

March 18, 2015

Contents

1 Overview

The subject of this project is an archaeological study of an area full of funerary monuments between the fifth and sixth miles of the Via Appia Antica. The data gathered from this area are of different resolution and different types (modalities): point clouds, meshes, photos, historical images and attribute information of the monuments of interest. The data are managed, processed and visualized by the Via Appia 3D GIS. The GIS combines a 3D viewer and a database (DB) and allows the user to refine which data are visualized based on some attributes. The system, which can be updated when new data is available, allows multiple researchers in different locations to analyze and study the area in 3D and aims to be an example for other complex archaeological study areas. This document is the user manual for the data management part of the system.

The Via Appia 3D/4D GIS has client-server architecture described in Section 2.The data are heterogeneous in nature both due to their conceptual difference and acquisition modality. The conceptual data description can be found in Section 3. The storage of the data is organized according to a predefined hierarchical convention as described in Section 4. The DB, storing the actual data location as well as many attribute data (meta-data) of archaeological interest, is explained in Section 5. The software used in the 3D/4D GIS is described in Section 6.

2 System architecture

The developed 3D/4D GIS system has a two-tier *client-server* architecture (Figure 2). The *server* contains the master copy of the data and a PostgreSQL database called *viaappiadb*. The *clients* download or request the data required for visualization and run the 3D/4D viewer locally or via a Web application which connects to the remote database when required.

A diagram of the data preparation framework which is executed in the server is shown in Figure 1. The acquired data for the road itself (background) are usually point clouds, while for the monuments (sites) the data are of more modalities- point clouds, meshes, pictures. Depending on the preferred application on the client side, Windows desktop or Web-based, the raw data are handled in one of the two ways.

In the former case the raw data is converted to the binary format of the OpenSceneGraph (`http://www.openscenegraph.org`). In the latter case, the point clouds are converted using the POTREE (`potree.org`) WebGL point cloud converter. The *viaappiadb* database is filled with meta-data information of the location of the raw data and the converted data. The archaeological information with attribute data for the several sites is provided in a Microsoft Access file. It needs to be converted to the PostgreSQL format before being imported into the main database. The footprints are provided in a PostgreSQL dump file and are imported into the *viaappiadb* database as well. The altitude ranges are derived from the PC raw data in combination with the footprints and added to the Database. On Figures 1 and 2 the solid arrows indicate data flow, while the dashed arrows indicate meta-data flow.

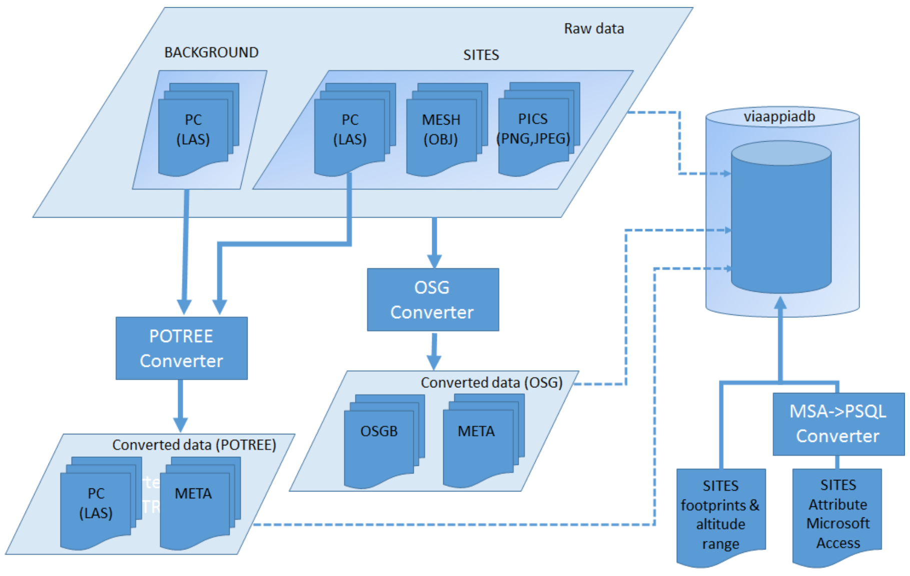

Figure 1: Data preparation framework executed on the Via Appia Linux serer

Once the data preparation framework has been executed on the server the several clients can start using the 3D/4D GIS system. In the case of Windows desktops or laptops, local copies of the OSG data required for visualization need to be downloaded. A *launcher* tool based on the Xenon library (`http://nlesc.github.io/Xenon/`), developed by the Netherlands eScience Center (NLeSc), is used for this purpose. The tool also downloads the configuration file required by the 3D viewer. Once both the data and the configuration file are locally available in the client (Windows computer) the launcher tool automatically starts the 3D viewer. There is also a Web client which can use the POTREE converted point clouds. In this case no local copy of the data is needed for the 4D viewer, but the necessary data are pulled from the server on request via NGINX Web server (`nginx.com`). The viewer also needs a JSON configuration file containing the meta-data for the background and the sites obtained from the *viaappia* DB. The 4D viewer have been developed by NLeSc. Figure 2 illustrates the two-tier architecture and shows the steps performed at the client side.

Both the execution of the data preparation framework in the server and the steps in the client have been automated. The user is referred to Section 6 for instruction of how to use the architecture.

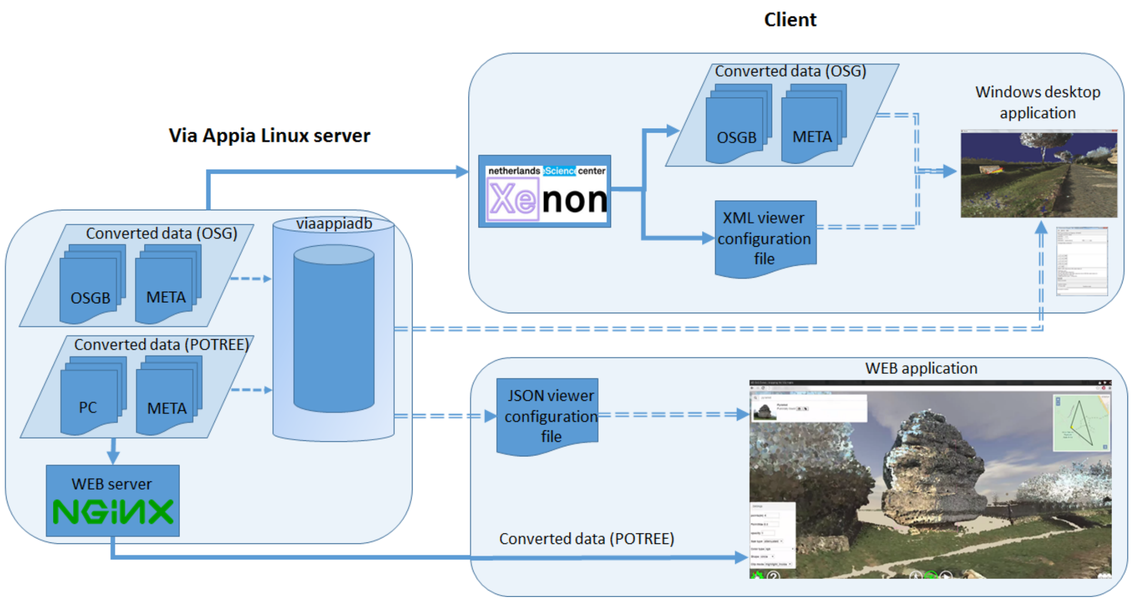

Figure 2: Two-tier architecture of the Via Appia 3D/4D GIS and the steps executed on the client side.

3 Conceptual description

The gathered data of interest originate from the surface of an area corresponding to one mile of the ancient Via Appia in Rome. Conceptually these are divided into road data (also referred as to the *background*), which serve as a reference system and context and monument data, which represent the objects of interest, aligned with the road data. The monuments, are referred to as *sites*.

The data, depending on the way of acquisition, are of several types:

- *Point clouds (PC)*
  - A low resolution PC of the research area that has been generated making use of Fugro's DRIVE-MAP services.
  - High resolution PCs of the different objects of interest (monuments, sites) generated using photogrammetric technologies.
- *Meshes* (sites reconstructions) for different historical epochs.
- Contemporary and historical *pictures* or paintings.
- *Attributes* data for the sites and their parts, which are gathered by field observations.

3.1 Point clouds

Using the DRIVE-MAP service of Fugro the part of the surface of the Via Appia between the fifth and sixth miles was scanned and a *PC* was produced. A point in the PC has 3D coordinates (x, y, z), in respect to a given Earth reference system known at scanning time, color and possibly other measured attributes. The resolution (number of points per area or volume) of this point cloud is not enough to see detailed features of the sites, furthermore, they were only scanned from the main road. Thus, the back of the sites is missing. This is illustrated on Fig.3.

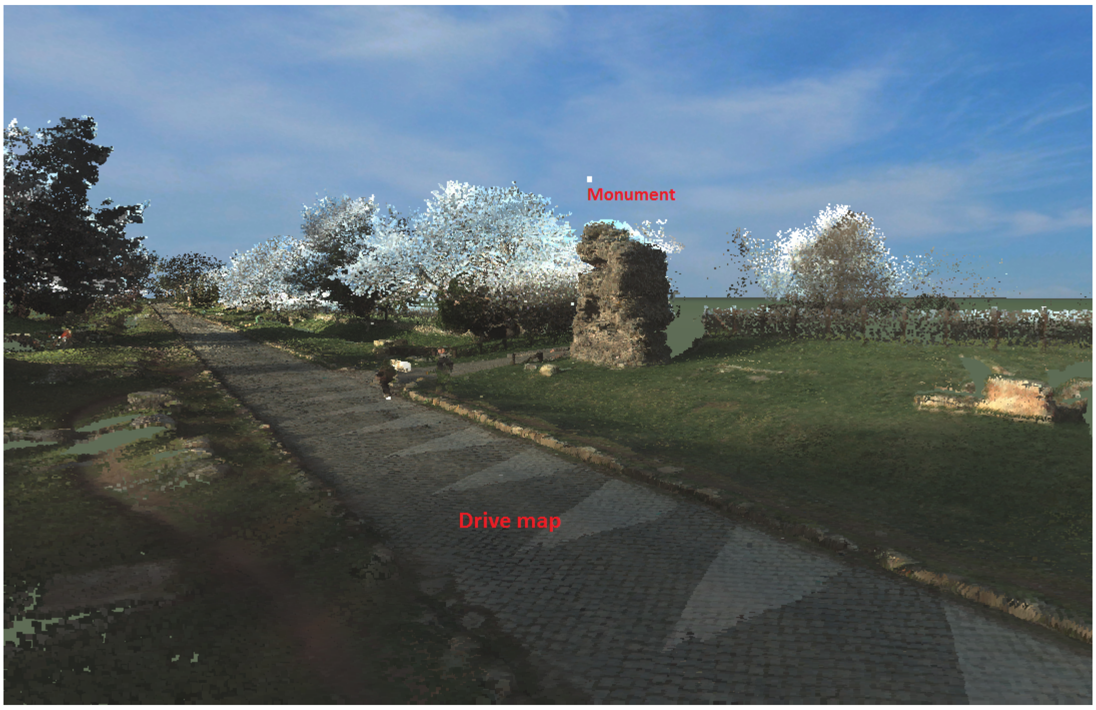

Figure 3: Point cloud data gathered along Via Appia. The Drive map is of low resolution and covers only part of the monuments (sites) facing the road.

Usually there is one drive-map used as a background at a time, while there can be multiple drive-maps obtained at different times or acquisition devices. Or there can be different versions of the same drive-map where some post-processing have been performed like points classification (e.g. sky, tree, road, etc.) and possibly non-interesting points filtering (e.g. tree and grass removal).

Additional point clouds of higher resolution covering also the back of the monuments have been obtained using photogrammetric software from photographic pictures of the sites from different camera locations (viewpoints).

3.2 Pictures

The pictures of a site can be both contemporary (current) or historical pictures or paintings. An illustration of visualizing a contemporary photo of a monument next to its point cloud data can be seen on Fig.4. The photo was one of the photos used to generate the high resolution site PC and visualizing it next to the location of the monument can be extra informative or can be used as thumbnail. Usually there are multiple pictures per site.

3.3 Meshes

For the purpose of archaeological research, multiple reconstructions of the sites of interest are often performed. These reconstructions, or meshes, can also be current or historical. An illustration of visualizing a contemporary mesh of a monument aligned to its point cloud data can be seen on Fig.4.

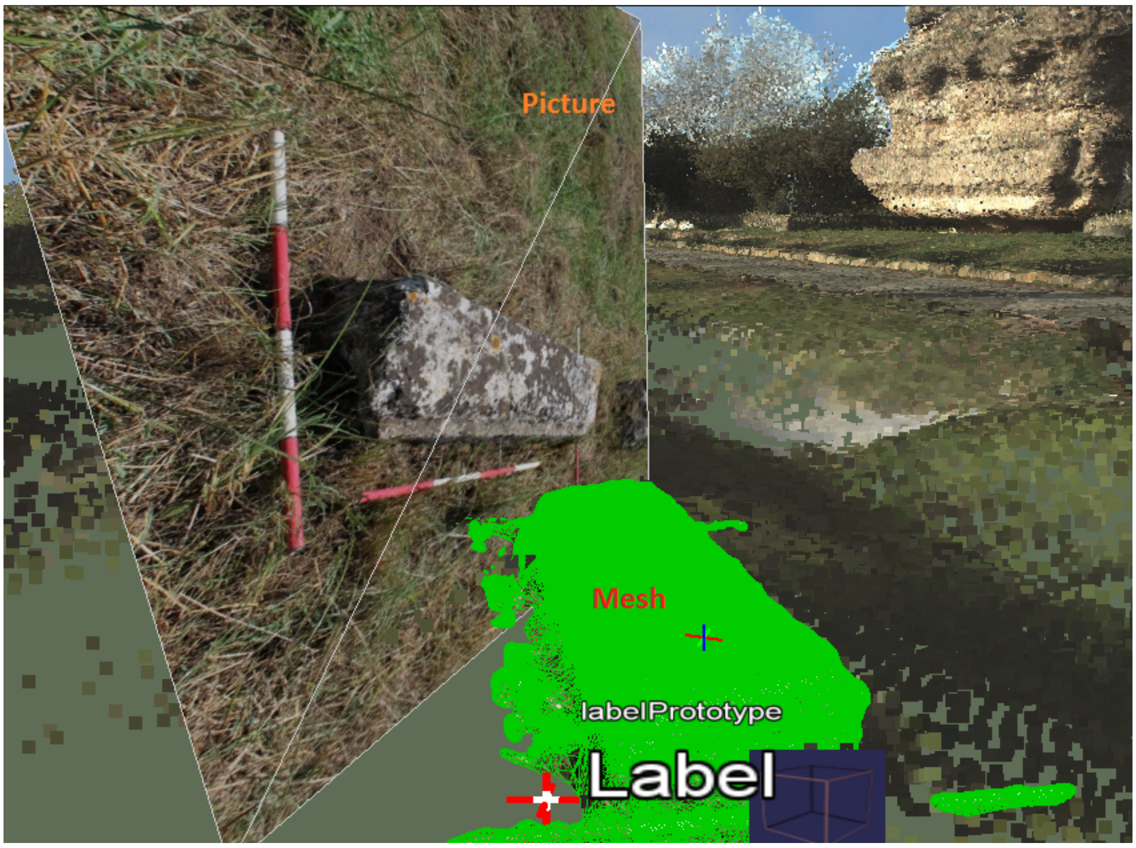

Figure 4: Reconstruction meshes and picture of a site overlayed on the point cloud data.

The point clouds, meshes and pictures are stored according to the data structure convention (Section 4).

3.4 Attributes

These data are collected on the site from archaeologists. They are relevant to the archaeological research question, and can be, for example material composition, condition, possible interpretation, description of the different elements or sub-parts, etc.

The attribute data are considered as additional descriptive (meta) data and are stored in a database along with the pointers to the other data types (Section 5).

4 Data structure

4.1 Description of the data structure

The raw data is **only** stored in the Via Appia Linux server. There is python script that needs to be used to generate the converted OSG data, *createosgdata.py*. Another python script exists that needs to be used to generate the converted POTree data, *CreatePOTreeConfig.py*. The raw data directory is */home/vadata/DATA/RAW*. OSG data is stored in */home/vadata/DATA/OSG* and POTREE data is stored in */home/vadata/DATA/POTREE* (figure 5).

The next subsections explores the RAW data tree in more detail and discusses how to modify and list the data in the RAW data tree.

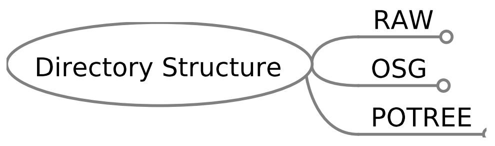

Figure 5: Overview of the data structure

4.1.1 Raw data

Point clouds

Higher resolution point clouds are stored in the subdirectories *PC/BACK* and *PC/SITE*, for respectively backgrounds and sites. For backgrounds, the subdirectory contains different folders with point clouds for each background. For sites, the subdirectory contains a separate folder for each site (e.g. *PC/SITE/S162* for site 162), which in turn contain different folders for each point cloud of the site. These folders contain LAS files of different point clouds of the site.

The LAS files contained in each site subfolder may have been pre-aligned (using CloudCompare for example). In that case the LAS file name must be ***_ALIGNED_*BGNAME**** where *BGNAME* must be the background name (as contained in the folder *PC/BACK/*). Some point clouds generation tools store the color information in 8 bits instead of the usual 16 bits. In that case the folder name must be ***_8BC**. The effect of having an undeclared LAS file with 8 bit color is that the converted data will be black and white. Note that these properties are cumulative, for example *S162_ALIGNED_DRIVE_1_V3_8BC* is a valid name for a folder containing a LAS file with 8 bit color information and aligned points.

Meshes

Meshes are stored in the directories *MESH/BACK* and *MESH/SITE*, for respectively backgrounds and sites. There are two types of meshes: (a) current mesh representations, and (b) archeological reconstructions. For backgrounds, they are stored in respectively *MESH/BACK/CURR* and *MESH/BACK/ARCH_REC*. For sites they are stored in respectively *MESH/SITE/CURR* and *MESH/SITE/ARCH_REC*. For sites, each of these directories contain different folders for each site. For example the folder *MESH/SITE/CURR/S162* contains mesh representations of the current state of the site 162.

For each site there must be different folders for the several meshes. Inside these folders the files for the meshes are stored (OBJ, MTL, JPEG). For example *MESH/SITE/CURR/S162* could contain two folders called *162_curr_1* and *162_curr_2* and each of them contain a OBJ, a MTL and several JPEG files.

Similarly to the LAS files the meshes can also be aligned. In this case the folder name for a certain mesh must be ***_ALIGNED_*BGNAME****.

Pictures

Pictures are stored in the subdirectory *PICT*. Pictures can be either of a background or of a site. For both types a further subdivision is made between (a) pictures of the current state of the sites, and (b) historical pictures and paintings. These are stored in respectively *PICT/BACK/CURR* and *PICT/BACK/HIST* for backgrounds, or *PICT/SITE/CURR* and *PICT/SITE/HIST* for sites. For sites, each of these directoryes contains a separate folder for each site. For example the folder *PICT/SITE/CURR/S162* contains pictures of the current state of the site 162.

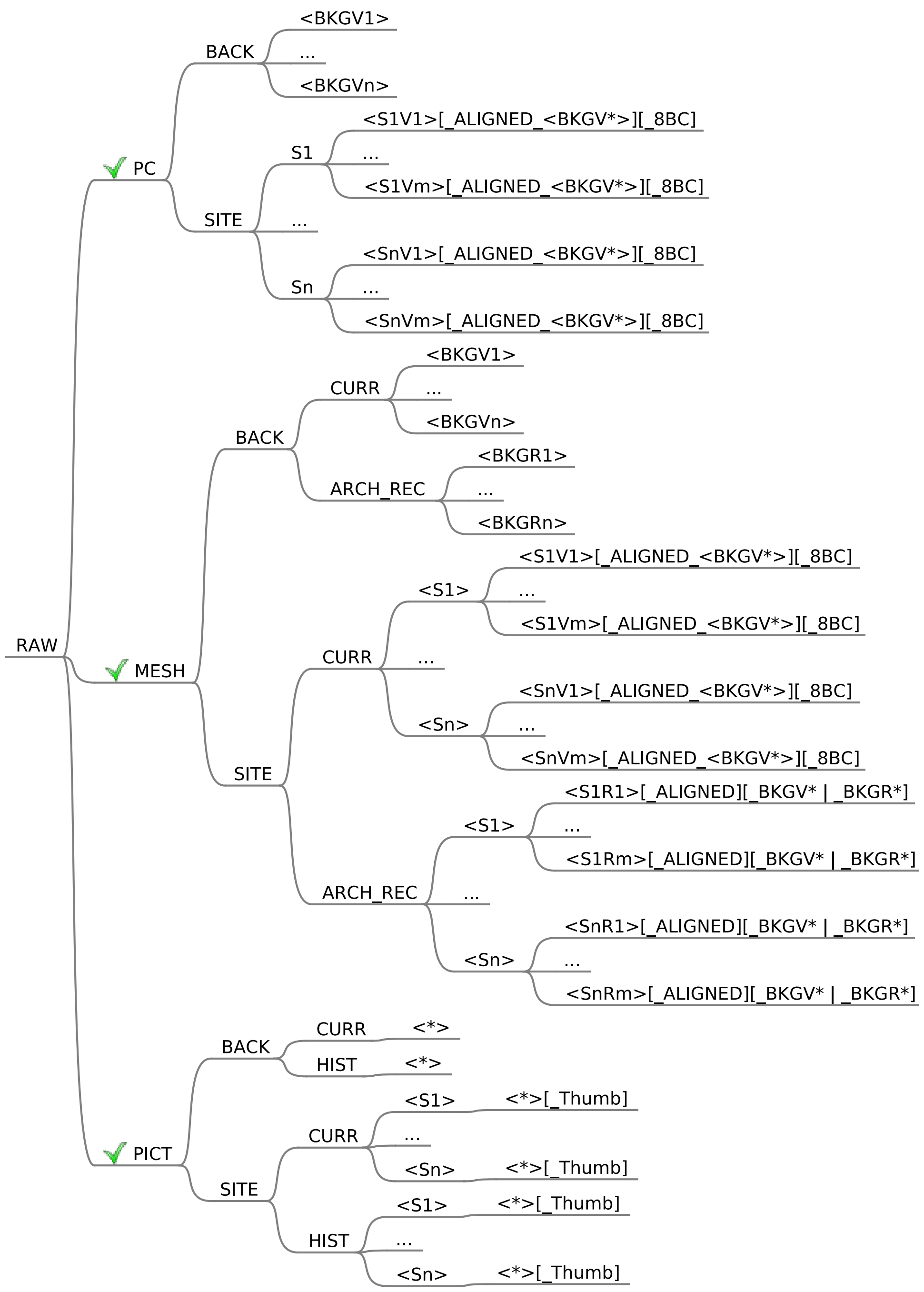
RAW
PC
BACK
<BKGV1>
...
<BKGVn>
SITE
S1
<S1V1>[_ALIGNED_<BKGV*>][_8BC]
...
<S1Vm>[_ALIGNED_<BKGV*>][_8BC]
...
Sn
<SnV1>[_ALIGNED_<BKGV*>][_8BC]
...
<SnVm>[_ALIGNED_<BKGV*>][_8BC]
MESH
BACK
CURR
<BKGV1>
...
<BKGVn>
ARCH_REC
<BKGR1>
...
<BKGRn>
SITE
CURR
<S1>
<S1V1>[_ALIGNED_<BKGV*>][_8BC]
...
<S1Vm>[_ALIGNED_<BKGV*>][_8BC]
...
<Sn>
<SnV1>[_ALIGNED_<BKGV*>][_8BC]
...
<SnVm>[_ALIGNED_<BKGV*>][_8BC]
ARCH_REC
<S1>
<S1R1>[_ALIGNED][_BKGV* | _BKGR*]
...
<S1Rm>[_ALIGNED][_BKGV* | _BKGR*]
...
<Sn>
<SnR1>[_ALIGNED][_BKGV* | _BKGR*]
...
<SnRm>[_ALIGNED][_BKGV* | _BKGR*]
PICT
BACK
CURR <*>
HIST <*>
SITE
CURR
<S1> <*>[_Thumb]
...
<Sn> <*>[_Thumb]
HIST
<S1> <*>[_Thumb]
...
<Sn> <*>[_Thumb]

Figure 6: Overview of the data structure: RAW data items

4.1.2 OSG data

The raw data can be converted to the OSGB format. For that purpose a python script *GenerateOSG.py* as been created. This script is discussed in section 4.2.5.

In order to view the data in your Windows machine you need to download a version of the converted OSG data in your local machine. We use a NLeSC tool called Xenon.

4.1.3 POTree data

The raw data can be converted to the POTree format. For that purpose a python script *GeneratePOTree.py* has been created. This script is discussed in section 4.2.4.

4.2 Operation on the data structure

Different operations can be performed on the data structure. The different operations that can be performed are (1) adding raw data items (section 4.2.1); (2) removing raw data items (section 4.2.2); and (3) listing raw data items (section 4.2.3).

4.2.1 Adding raw data items

This section describes the script that should be used to add raw data items to the directory structure: *AddRawDataItem.py.* To run the script, the user needs, depending on the exact datatype, supply additional arguments to the script. An overview of the possible arguments are found in the verbatim below. The script automatically creates a destination folder with the required naming, as specified in section 4.1, using the supplied arguments.

For MESH and PICT, the (*-f, --file*) argument can be either a file, or a folder containing files with the correct extension. For a PC the argument must always be a folder.

```
 usage: AddRawDataItem.py [-h] [-i DATA] -k BACK,SITE -t PC,MESH,PICT -f FILE
              [-p CURR,HIST,ARCH_REC] [-s SRID] [--eight]
              [-l debug,info,warning,error,critical] [--site SITE]

Add Raw data item to the file structure.

optional arguments:
  -h, --help            show this help message and exit
  -i DATA, --data DATA  RAW data folder [default /home/pattydat/DATA/RAW]
  -s SRID, --srid SRID  spatial reference system SRID [only for MESH SITE]
  --eight               8 bit color [only for PC SITE or MESH]
  -l debug,info,warning,error,critical, --log debug,info,warning,error,critical
                        Log level

required arguments:
  -k BACK,SITE, --kind BACK,SITE
                        Type of item
  -t PC,MESH,PICT, --type PC,MESH,PICT
                        Type of data
  -f FILE, --file FILE  Input file/directory name to copy

required arguments for MESH and PICT:
  -p CURR,HIST,ARCH_REC, --period CURR,HIST,ARCH_REC
                        Period (choose from MESH:CURR,ARCH_REC;
                        PICT:CURR,HIST)
```

```
required arguments for SITE:
  --site SITE           Site number
```

4.2.2 Removing raw data items

This section describes the script that should be used to remove raw data items from the file structure: *RemoveRawDataItem.py*.

```
usage: RemoveRawDataItem.py [-h] -i ITEMID [-d DBNAME] [-u DBUSER] [-p DBPASS]
                            [-t DBHOST] [-r DBPORT]
                            [-l debug,info,warning,error,critical]

Removes a list of Raw data items and their related converted data from the
file structure.

optional arguments:
  -h, --help            show this help message and exit
  -d DBNAME, --dbname DBNAME
                        PostgreSQL DB name viaappiadb]
  -u DBUSER, --dbuser DBUSER
                        DB user [default ronald]
  -p DBPASS, --dbpass DBPASS
                        DB pass
  -t DBHOST, --dbhost DBHOST
                        DB host
  -r DBPORT, --dbport DBPORT
                        DB port
  -l debug,info,warning,error,critical, --log debug,info,warning,error,critical
                        Log level

required arguments:
  -i ITEMID, --itemid ITEMID
                        Comma-separated list of Raw Data Item Ids Raw data
                        item id (with ? the available raw data items are
                        listed)
```

4.2.3 Listing raw data items

This section describes the script that lists the raw data items currently in the file structure: *ListRawDataItem.py*.

```
usage: ListRawDataItem.py [-h] [-i ITEMID] [-d DBNAME] [-u DBUSER] [-p DBPASS]
                          [-t DBHOST] [-r DBPORT]
                          [-l debug,info,warning,error,critical]

List the Raw data items that are in the DB.

optional arguments:
  -h, --help            show this help message and exit
  -i ITEMID, --itemid ITEMID
                        List the Raw Data Item Ids related to a list of items
                        (comma-separated) [default list all raw data items]
  -d DBNAME, --dbname DBNAME
                        PostgreSQL DB name viaappiadb]
  -u DBUSER, --dbuser DBUSER
                        DB user [default $USERNAME]
  -p DBPASS, --dbpass DBPASS
                        DB pass
  -t DBHOST, --dbhost DBHOST
                        DB host
  -r DBPORT, --dbport DBPORT
                        DB port
  -l debug,info,warning,error,critical, --log debug,info,warning,error,critical
                        Log level
```

4.2.4 Generating OSG data

This script should be run with the *vadata* user and should be run when some data has changed in the raw data directory. The POTree data is copied automatically in the POTree data tree, with the naming as specified in figure 7.

```
 usage: GenerateOSG.py [-h] [-i ITEMID] [-d DBNAME] [-u DBUSER] [-p DBPASS]
                      [-t DBHOST] [-r DBPORT] [-o OSGDIR]
                      [-l debug,info,warning,error,critical]

Generates the OSG data for a raw data item.

optional arguments:
  -h, --help            show this help message and exit
  -i ITEMID, --itemid ITEMID
                        Comma-separated list of Raw Data Item Ids [default is
                        to convert all raw data items related to sites that do
                        not have a related OSG data item] (with ? the
                        available raw data items are listed, with ! the list
                        all the raw data items without any related OSG data
                        item)
  -d DBNAME, --dbname DBNAME
                        Postgres DB name [default viaappiadb]
  -u DBUSER, --dbuser DBUSER
                        DB user [default $USERNAME]
  -p DBPASS, --dbpass DBPASS
                        DB pass
  -t DBHOST, --dbhost DBHOST
                        DB host
  -r DBPORT, --dbport DBPORT
                        DB port
  -o OSGDIR, --osgDir OSGDIR
                        OSG data directory [default /home/pattydat/DATA/OSG]
  -l debug,info,warning,error,critical, --log debug,info,warning,error,critical
                        Log level
```

4.2.5 Generating POTree data

This script should be run with the *vadata* user and should be run when some data has changed in the raw data directory. The POTree data is copied automatically in the POTree data tree, with the naming as specified in figure 8.

```
usage: GeneratePOTree.py [-h] [-i ITEMID] [-d DBNAME] [-u DBUSER] [-p DBPASS]
                         [-t DBHOST] [-r DBPORT] [-o POTREEDIR]
                         [--levels LEVELS]
                         [-l debug,info,warning,error,critical]

Generates the POTree data for a raw data item (ONLY FOR PCs)
```

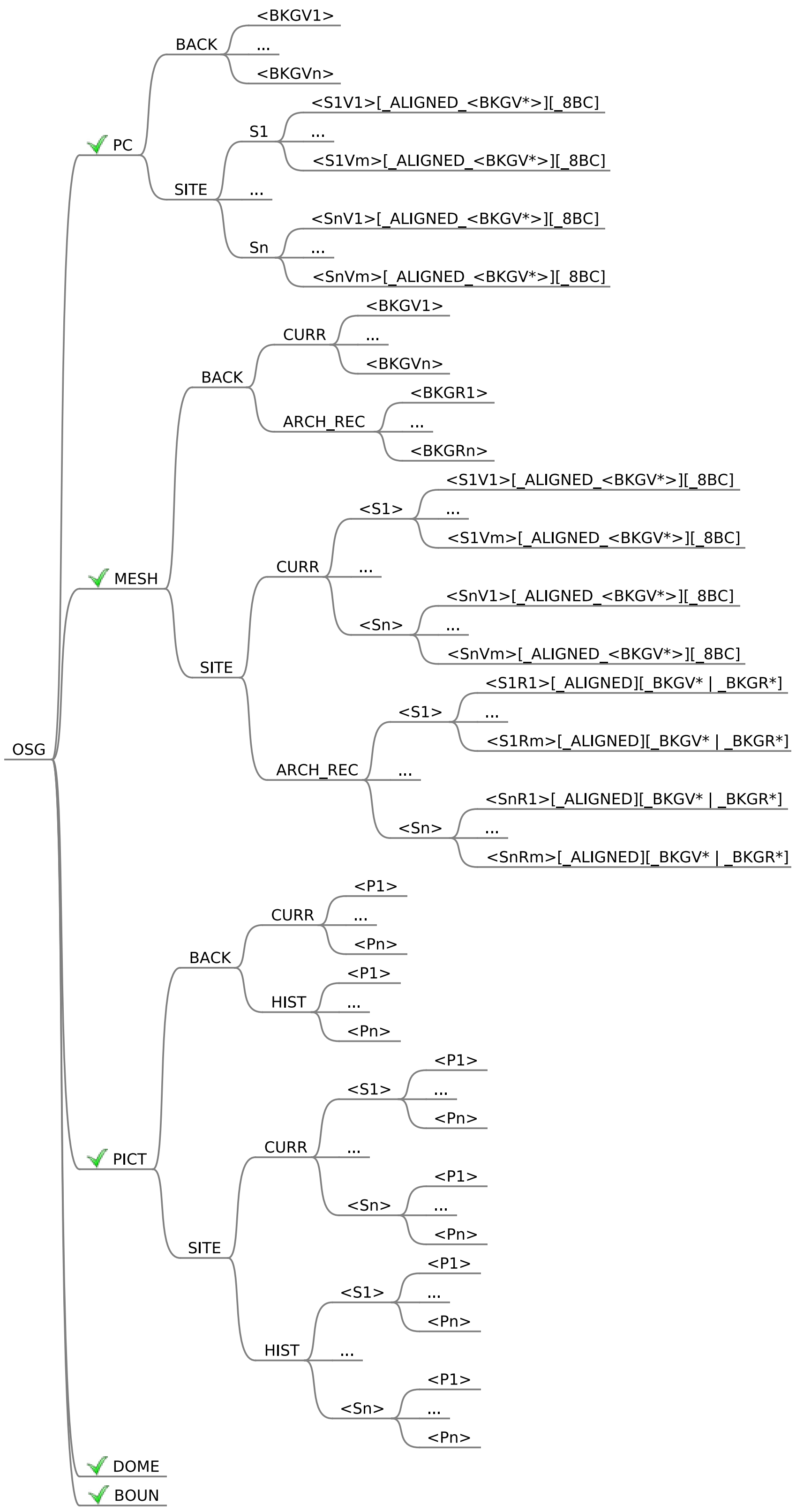
OSG
✓ PC
BACK
<BKGV1>
...
<BKGVn>
SITE
S1
<S1V1>[_ALIGNED_<BKGV*>][_8BC]
...
<S1Vm>[_ALIGNED_<BKGV*>][_8BC]
...
Sn
<SnV1>[_ALIGNED_<BKGV*>][_8BC]
...
<SnVm>[_ALIGNED_<BKGV*>][_8BC]
✓ MESH
BACK
CURR
<BKGV1>
...
<BKGVn>
ARCH_REC
<BKGR1>
...
<BKGRn>
SITE
CURR
<S1>
<S1V1>[_ALIGNED_<BKGV*>][_8BC]
...
<S1Vm>[_ALIGNED_<BKGV*>][_8BC]
...
<Sn>
<SnV1>[_ALIGNED_<BKGV*>][_8BC]
...
<SnVm>[_ALIGNED_<BKGV*>][_8BC]
ARCH_REC
<S1>
<S1R1>[_ALIGNED][_BKGV* | _BKGR*]
...
<S1Rm>[_ALIGNED][_BKGV* | _BKGR*]
...
<Sn>
<SnR1>[_ALIGNED][_BKGV* | _BKGR*]
...
<SnRm>[_ALIGNED][_BKGV* | _BKGR*]
✓ PICT
BACK
CURR
<P1>
...
<Pn>
HIST
<P1>
...
<Pn>
SITE
CURR
<S1>
<P1>
...
<Pn>
...
<Sn>
<P1>
...
<Pn>
HIST
<S1>
<P1>
...
<Pn>
...
<Sn>
<P1>
...
<Pn>
✓ DOME
✓ BOUN

Figure 7: Overview of the data structure: OSG data items

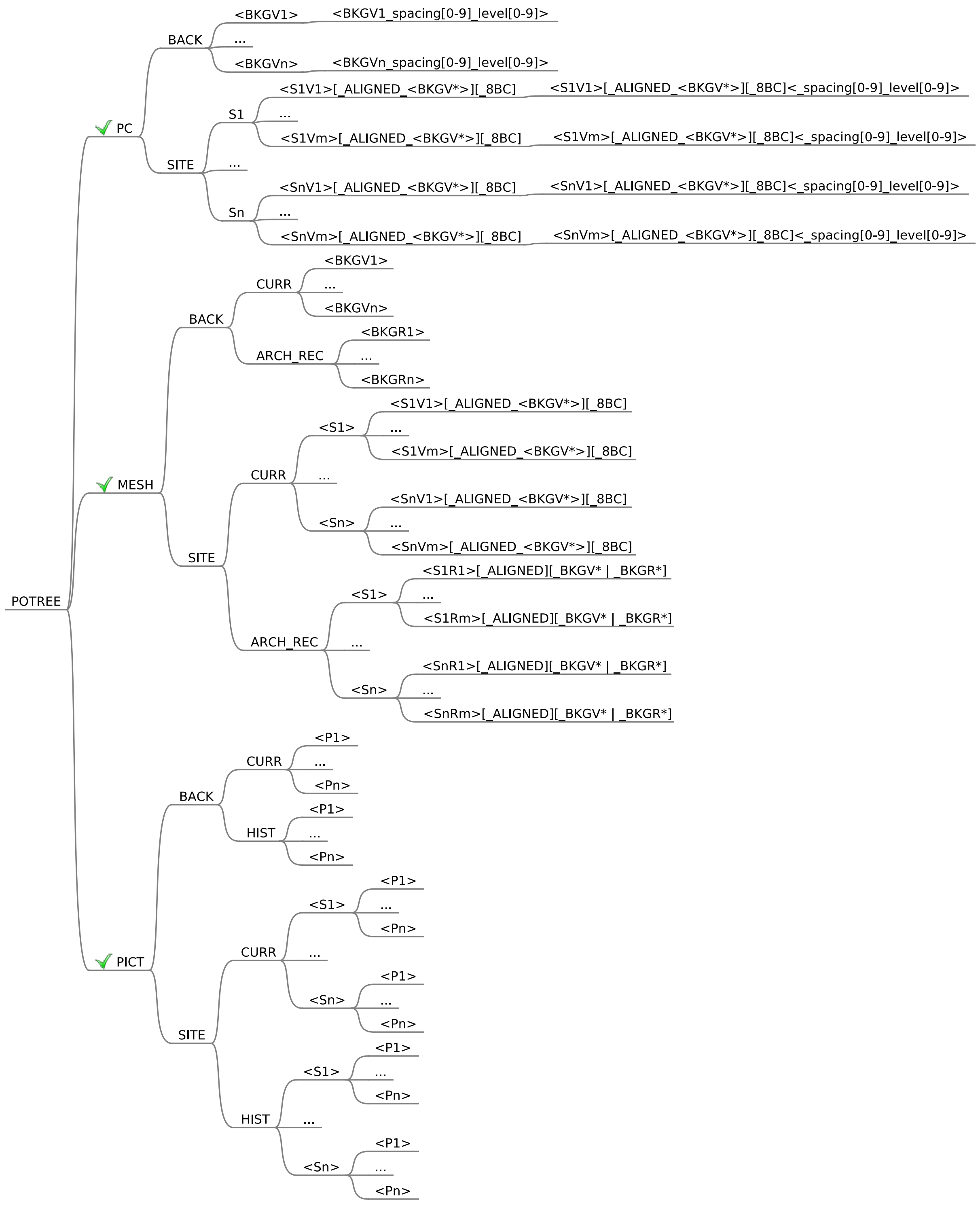
POTREE
PC
BACK
<BKGV1>
<BKGV1_spacing[0-9]_level[0-9]>
...
<BKGVn>
<BKGVn_spacing[0-9]_level[0-9]>
SITE
S1
<S1V1>[_ALIGNED_<BKGV*>][_8BC]
<S1V1>[_ALIGNED_<BKGV*>][_8BC]<_spacing[0-9]_level[0-9]>
...
<S1Vm>[_ALIGNED_<BKGV*>][_8BC]
<S1Vm>[_ALIGNED_<BKGV*>][_8BC]<_spacing[0-9]_level[0-9]>
...
Sn
<SnV1>[_ALIGNED_<BKGV*>][_8BC]
<SnV1>[_ALIGNED_<BKGV*>][_8BC]<_spacing[0-9]_level[0-9]>
...
<SnVm>[_ALIGNED_<BKGV*>][_8BC]
<SnVm>[_ALIGNED_<BKGV*>][_8BC]<_spacing[0-9]_level[0-9]>
MESH
BACK
CURR
<BKGV1>
...
<BKGVn>
ARCH_REC
<BKGR1>
...
<BKGRn>
SITE
CURR
<S1>
<S1V1>[_ALIGNED_<BKGV*>][_8BC]
...
<S1Vm>[_ALIGNED_<BKGV*>][_8BC]
...
<Sn>
<SnV1>[_ALIGNED_<BKGV*>][_8BC]
...
<SnVm>[_ALIGNED_<BKGV*>][_8BC]
ARCH_REC
<S1>
<S1R1>[_ALIGNED][_BKGV* | _BKGR*]
...
<S1Rm>[_ALIGNED][_BKGV* | _BKGR*]
...
<Sn>
<SnR1>[_ALIGNED][_BKGV* | _BKGR*]
...
<SnRm>[_ALIGNED][_BKGV* | _BKGR*]
PICT
BACK
CURR
<P1>
...
<Pn>
HIST
<P1>
...
<Pn>
SITE
CURR
<S1>
<P1>
...
<Pn>
...
<Sn>
<P1>
...
<Pn>
HIST
<S1>
<P1>
...
<Pn>
...
<Sn>
<P1>
...
<Pn>

Figure 8: Overview of the data structure: POTree data items

```
optional arguments:
  -h, --help            show this help message and exit
  -i ITEMID, --itemid ITEMID
                        Comma-separated list of PointCloud Raw Data Item Ids
                        [default is to convert all raw data items that do not
                        have a related POtree data item] (with ? the available
                        raw data items are listed, with ! the list all the raw
                        data items without any related POTree data item)
  -d DBNAME, --dbname DBNAME
                        Postgres DB name [default viaappiadb]
  -u DBUSER, --dbuser DBUSER
                        DB user [default $USERNAME]
  -p DBPASS, --dbpass DBPASS
                        DB pass
  -t DBHOST, --dbhost DBHOST
                        DB host
  -r DBPORT, --dbport DBPORT
                        DB port
  -o POTREEDIR, --potreeDir POTREEDIR
                        POTREE data directory [default
                        /home/pattydat/DATA/POTREE]
  --levels LEVELS       Number of levels of the Octree, parameter for
                        PotreeConverter. [default is 4 for Sites and 8 for
                        Backgrounds]
  -l debug,info,warning,error,critical, --log debug,info,warning,error,critical
                        Log level
```

5 Database

The *viaappiadb* database is running in the Via Appia Linux server. See Section 6 on how to get an account in the database.

This database (DB) has conceptually two major parts and five categories: (a.) data management information and (b.) the attribute data. The data management part has 4 categories: ITEM, RAW, POTREE and OSG and the attribute data are represented in category ATTRIBUTE.

In Figure 9 the Entity-Relationship diagram (ERD) of the *viaappiadb* is shown. In next Subsections each category is described briefly and illustrated with its part of the ERD also indicating the connections (only direct links are indicated) to the other categories.

Note that some of the nodes of the relationships are *0:n* or *0:1* (with black points) instead of the usual *1:n* or *1:1*. This is to illustrate that some sites and objects may have entries in some tables but not in others. For example it is possible to have a site in the item table which has no entry in the attributes table (tbl1_site).

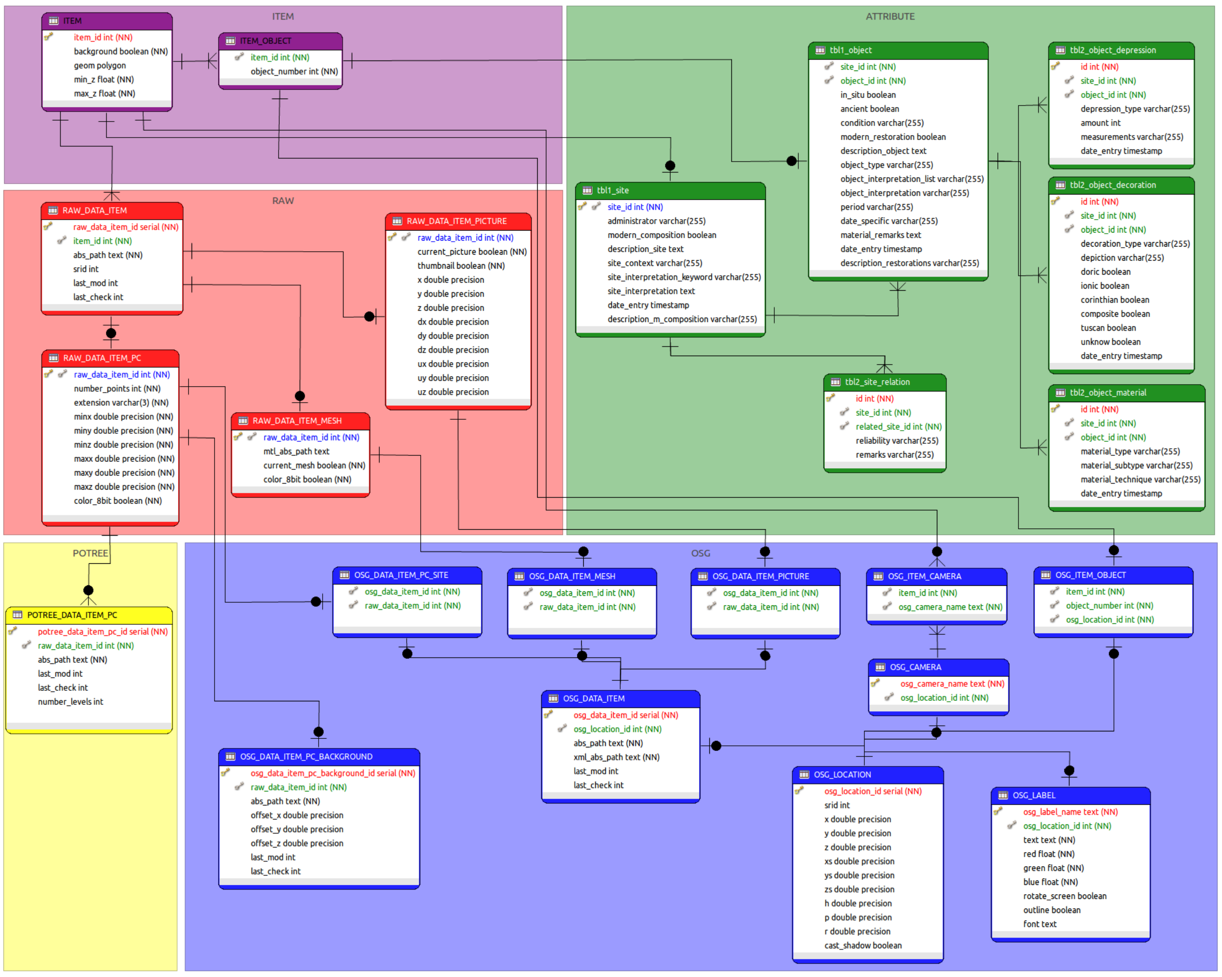

Figure 9: Entity-Relationship diagram of the *viaappiadb* database with its five categories.

5.1 Data management

This part of the DB contains tables with the locations of the raw and converted data, visualization-related data and the footprints (geometries) of the sites.

5.1.1 ITEM

Conceptually this is a category containing an information about an *item*. Item is any entity of interest- background (then the logical field *background* is set to true (1)) or site (0). Is the item of interest is an archaeological site (monument) it can contain one or more objects (parts of the site) which is reflected by the field *object_numer*. Figure 10 shows the relationship of the ITEM category with the other categories. The ITEM category is on the top of the categories' hierarchy.

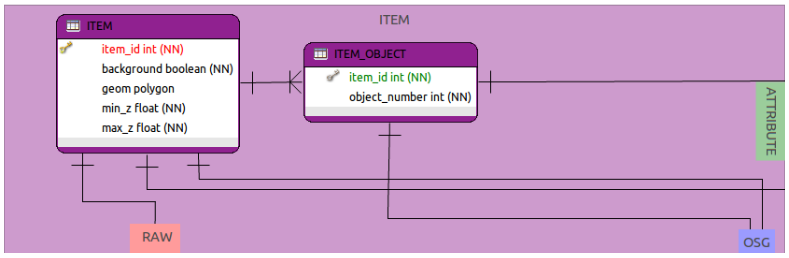

Figure 10: Entity-Relationship diagram of category ITEM and indicated connections with other categories.

5.1.2 RAW

The RAW category contains all the *raw data* gathered for given item. The most important meta-data is the location in the Data structure (Section 4), stored in the *abs_path* field. These data can be either point clouds (PC), meshes (MESH) or pictures (PICTURE). Each of these types is represented by a separate table with the specific for that data type properties. The RAW category is related to the derived data categories OSG and POTREE as indicated on Figure 11.

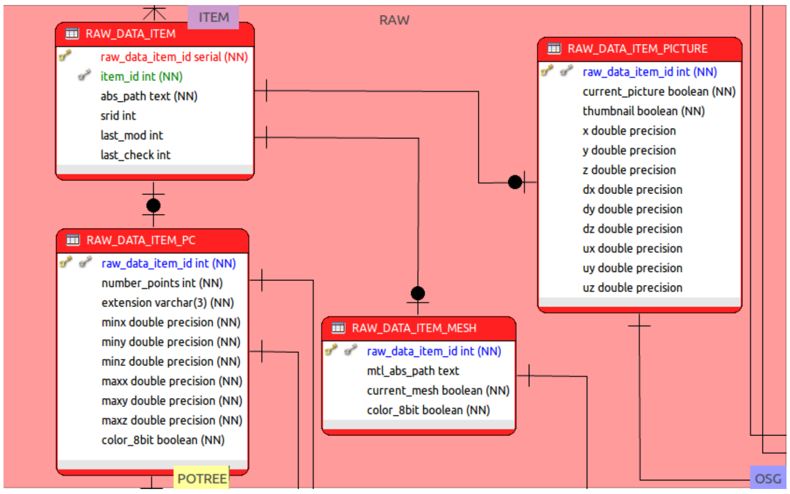

Figure 11: Entity-Relationship diagram of category RAW and indicated connections with other categories.

5.1.3 OSG

This category represents the OSG *converted data* used for the Windows desktop/laptop client application. As seen by the ERD on Figure 12 it is related to the RAW data (from where it is derived) and to the ITEM category. The tables in this category reflect the possible data types: PC, meshes and pictures. Most importantly, this category contains information needed for the *visualization*, like specifics of the background and items, the camera, location (bounding box) and label.

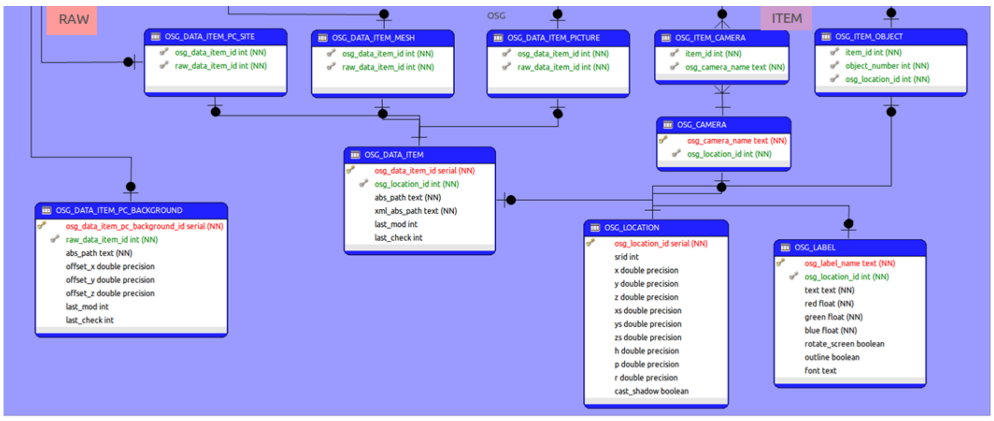

Figure 12: Entity-Relationship diagram of category OSG and indicated connections with other categories.

5.1.4 POTREE

This category is illustrated on Figure 13. As it can be seen, it is related only to the RAW category from which it is derived using the POTREE converter and stores only data of the PC type and conversion parameters (*number_levels*).

5.2 Attribute data

The attribute part of the DB is represented only by one category (Figure 14). It is connected only to the ITEM category. These are the meta-data collected during the field work and are primarily of research interest to the archaeologists as it contains all domain-related data enabling browsing and filtering of sub-parts of the data of interest.

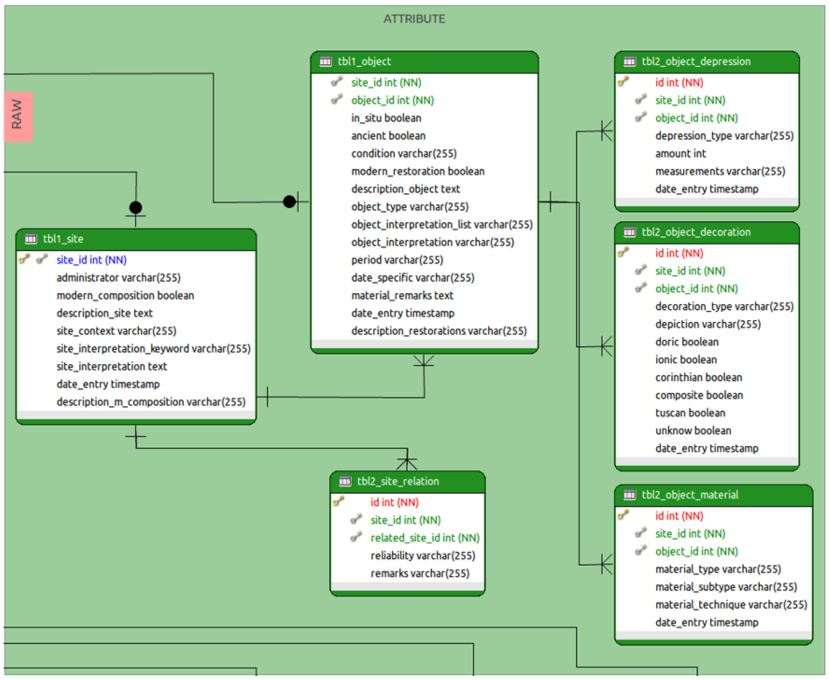

Figure 14: Entity-Relationship diagram of category ATTRIBUTE and indicated connections with other categories.

6 Software

As described in section 2 the 3D GIS system has a two tier architecture, the server and the clients.

6.1 Server software

The *viaappiadb* database running in the Via Appia Linux server is a PostgreSQL 9.2.8 with PostGIS 2.1.2 and GDAL 1.10.0.

Specific point cloud libraries are also used for the management and processing of point cloud data, concretely LASzip, libLAS and LAStools. In LAStools and libLAS many of their applications share the same name and in these cases the usage of the LAStools ones is preferred. In order to guarantee this we need to set the *PATH* environment variable accordingly, i.e. by setting the LAStools bin folder prior to the libLAS one.

As described in section 4 the data stored in the server needs to be converted to the specific formats required by the two supported visualizations, i.e. the web viewer based on Potree and the Windows desktop viewer/editor based on Open Scene Graph. To perform these conversions we use the *PotreeConverter* (https://github.com/potree/PotreeConverter) and the *OSG-Converter* (https://github.com/NLeSC/Via-Appia/tree/master/converter) which requires the Open Scene Graph (we currently use 3.2.1). Both converters use Boost (our installed version is 1.55).

An NGINX web server is also used to serve files (static content) to the web-based visualization.

6.2 Client software

The web visualization viewer only requires the installation of a modern web browser in the client laptop/desktop (we have used Chrome/Chromium).

For the desktop-based visualization we use a tool based on OpenSceneGraph (OSG) so the point cloud data and also the pictures and meshes have to be converted to OSG format. This conversion also happens in the Via Appia server and it is executed by a

For the usage of the OSG desktop viewer some additional libraries have to be installed

pasted form old documentaiton- to be edited!

In order to use the new 3D GIS system the end-user only needs to do:

- (a.) Obtain an account in the Via Appia Linux server and in the *viaappiadb* database and generate a SSH key pair (see Subsections **??** and **??**).
- (b.) Download, install and configure the *launcher* tool in his Windows laptop or desktop (see Subsection **??**). The *launcher* tool also contains the 3D viewer. It is also recommend to install two tools for the communication between the Linux server and the Windows computers: PuTTY and WinSCP (see Subsubsections **??** and **??**).

The database is a PostgreSQL 9.2.8 (version correct?) instance running with PostGIS 2.1.2.

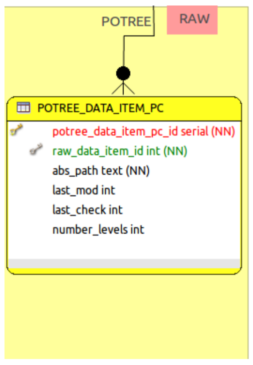

Figure 13: Entity-Relationship diagram of category POTREE and indicated connections with other categories.