Horizon 2020
European Union Funding
for Research & Innovation

Deliverable D 2.3
Data Management Standard

Project acronym:	GoSAFE RAIL
Starting date:	01/10/2016
Duration (in months):	36
Call (part) identifier:	H2020-S2R-OC-CCA-04-2015
Grant agreement no:	730817
Due date of deliverable:	Month 21
Actual submission date:	12-07-2018
Responsible/Author:	Damian Harasymczuk, Timo Hartmann, CT
Dissemination level:	PU/CO
Status:	Issued

Reviewed: (yes)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the GoSAFE RAIL grant agreement No 730817.

Document history		
Revision	*Date*	*Description*
1	29.06.2018	First issue
2	11.07.2018	Editing and additions for finalization for review by WP leader
3	26.08.2018	Added discussion about BIG DATA
4	10.01.2020	Addressed the request for revision: - Added the GA number in the footer - Added information on funding and disclaimer - Rephrased sentence on page 6 to clarify relation between standard and format
5	17.01.2020	Added disclaimer on the JU's non-responsibility

Report contributors		
Name	*Beneficiary Short Name*	*Details of contribution*
Damian Harasymczuk	CT	First issue
Timo Hartmann	CT	Second and third version

Table of Contents

1. Executive Summary

This report proposes a standard for railway asset information management. The standard suggests basing the storage of all information about and around a railway network on a graph database. Graph databases allow for an intuitive representation of a network as nodes and relationship between nodes. Therefore, graph databases allow for the quick querying of network data and for the attachment of all other relevant data either directly as data items within a graph database or as links to other database.

To store additional information within the proximity of railway network objects, the standard suggests including an additional graph-based tree structure that describes certain neighbours within hierarchical levels of details. Railway objects and other objects within a specific proximity can then be linked to these neighbourhood nodes and be easily accessed through graph-based queries of the tree structure.

The document here illustrates the elements of the suggested standard using the example of the GoSafe Railway information management system that was developed on this project.

2. Abbreviations and acronyms

Abbreviation / Acronyms	Description
JSON	JavaScript Object Notation
Geojson	Geographic JavaScript Object Notation
XML	eXtended Markup Language
OGC	Open Geospatial Consortium
OCS	Operational Control System
IS	Infrastructure
TT	Timetable and Rostering
IL	Interlocking Subschema
CO	Class Common
TB	Start Node
TE	End Node
SW	Switches
GIS	Geographic Information System
LOD	Level of Detail
IFCRail	Industry Foundation Classes Rail
CSV	Comma Separated Values
TSV	Tab Separated Values

3. Background

Railway agencies, companies and institutions utilize several different data sources and formats in asset management and decisions making process. Even if the given dataset describes the same part of the railway network as another dataset, it can be difficult to integrate, merge and process all information together in one information management system. However, complicating the situation often different data sets describe partly overlapping information. In the end, the data management systems at railway agencies need to be able to integrate more than one type of data source.

Many formats exist to store data, such as JSON, XML, binary formats, or delimited text files. Additionally, many database systems exist. To manage a consistent data model, the existing data sources and new input must be organized into a fast, robust, and easy to understand information management system.

In this document a holistic information management system standard that integrates the most widely used data formats in the railway industry is introduced. At the core of the standard is a graph database management system that allows to integrate all other existing data formats, like RailML, CityGML, Vector formats, and less popular and custom formats in an easy an intuitive manner. Such state of the art graph-based database systems are highly scalable and fast tools to store network related in the form of graphs, and allows to streamline the management of large parts of the available data sources around a network into an overarching Information Management System. Information in graph-based database are stored as nodes, which can have numerous properties and can relate to any other node by using explicit relationships. Graph query languages are used that allow to obtain information about nodes and relations in the network.

Together with these storage and mapping related features, graph databases offer huge advantages in storing data around and about railway networks as they allow for the modelling of the topological features of a railway network (with distinguished beginnings, ends and connection points of elements). Furthermore, graph-based databases offer spatial extension that allow to organize nodes into spatial layers with spatial indices, which result in significant performance improvement for a vast array of spatial queries. The user interfaces of graph databases allow for the easy identification of parts of the graph and support to perform modifications in an uncomplicated and intuitive manner.

The introduced data management standard can be implemented in any system that supports railway infrastructure management. This standard report will draw upon the graph-based database Neo4J that is used in the GoSafe information management platform.

4. Objective/Aim

This document has been prepared to provide an overview of the data management standard developed within task 2.4 of the GoSAFE Rail project. This standard was thoroughly build upon existing standards (CITYGML, IfcRail, RailML, OGC standars, etc.) and provides clear guidelines for data management and maintenance for railway agencies for all data relevant to understand safety and mobility related aspects of their networks.

One important problem that this data standard will address is the integration of BIG DATA within the asset management IT systems of railway organizations. BIG DATA is characterized by one or more of the following characteristics:

- Volume – the quantity of the data is relatively high
- Variety – data comes in many different formats and aggregation levels
- Velocity – data is generated at high speed and causes continuous changes in the available data
- Veracity – the quality of captured data is varying greatly

To integrate BIG DATA within information management systems a much more flexible approach to data management is required than is implemented in current information management systems. To this respect, it is important that volume – the sheer size of a data set - is not the most problematic characteristic that requires a significant shift in how we design information systems. With the exponential increase in computational power even relatively standard computers can by now handle extremely large data sets. What makes the development of information management systems for BIG DATA difficult are the other three characteristics of BIG DATA. To store and query BIG DATA it is important that flexible information management systems are established that can cope with the variety, velocity, and veracity of the data.

To this end, information management systems need to be developed under the philosophy of distributed data base systems that do not store much data centrally any longer, but that allow to quickly integrate different existing data sources available in different formats. A sound integration requires a general framework that allows to virtually organize a railway network and its related structures. This data standard suggests such a structure around graph-based database systems that can integrate a wide range of different data sets in various formats and that at the same time allows to meaningfully access this data intuitively.

5. Data Management Standard

This section describes the overall structure of the suggested standard. At the core of the structure is a graph database that allows to intuitively and quickly query different parts of a railway network. Using dedicated relations additional data from many different sources can be attached to the nodes of the network modelled within the graph database. This information system can then be used to manage all information and data available to railway agencies and allows for the development of dedicated decision support, safety management, and asset management applications to support tasks within the railway agency and of external organizations that work with a railway agency. This section describes each of the parts of the suggested information management standard, starting with introducing the principles of graph databases.

5.1. Graph Database

A graph database management system is an online database management system that exposes a graph data model [1]. Graphs are collections of vertices and edges (nodes and relationships). In graph databases these elements can represent any objects and relations or connections between them. Unlike in relational database systems, there are no collections and nodes are not grouped into containers. There are four types of features (Figure 1), that create a graph:

- Nodes – main elements, contain information to describe an object. Nodes are equivalents of rows in a relational database.
- Relationships – elements which connect two nodes. Relationships have a name and can have different properties.
- Properties – key-value pairs describing nodes or relationships.
- Labels – additional properties that allow for grouping nodes into sets.

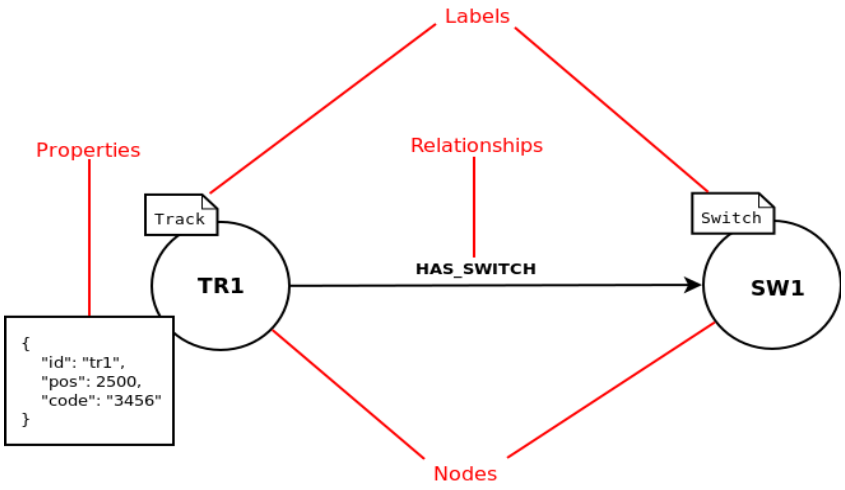

Figure 1 Graph elements: nodes, relationships, labels and properties

Graph databases support specific query languages, which can be used to perform a variety of operations. In this document the described database system which has been used to develop the GoSafe information management system, neo4J, for example is supported by the Cypher query language. It is an easy to read and understandable query language. Cypher allows to find data that matches a specific pattern. All requests and functions implemented in the GoSafe system, which interact with the database, are performed by executing Cypher queries. The next section of this report will explain how to integrate the most common standards used within Railway Asset Management using a graph database such as Neo4J.

5.2. Supported formats

An information management system for Railway agencies need to provide modules to convert a variety of different formats so that they can be stored in a graph database. Depending on the format, data can be connected to the railway network graph or spatial layers representing the closest neighbourhood of a track. Main supported formats are RailML and CityGML. Vector data formats like Shapefiles or Geojson are supported as well. Because the IFCRail project has been suspended, this document cannot describe how to integrate IFCRail, but future versions of the standard can be easily integrating this suspended standard in the future.

5.3. RailML

RailML (railway Markup Language) is an open, XML based data exchange format for data interoperability of railway applications. Currently there are three subschemas for productive use: Infrastructure (IS), Timetable and Rostering (TT) and Rollingstock (RS). The Interlocking subschema (IL) is in active development and can be obtained for development and test purposes. Elements that do not fit into this structure are subsumed in the Class Common (CO) [2].

The Infrastructure subschema contains complex information about a given railway network. Each track from the network is described with its topology, track elements and operational control system (OCS) elements. Each element is assigned to a track, so it can be considered as a tree structure. This is a huge advantage because a complete topological model can be created without providing a geographical position. Obviously non-georeferenced objects can't be displayed on a map or 3D globe. There is no other format or data source that can provide more detailed and accurate information about the railway network. We developed the GoSafe Rail Converter module to convert a RailML document to a graph and save it in the graph database.

Figure 2 illustrates an example RailML track description within the XML format. There are nested sections, each one is a container for elements. XML tags have various property, in most cases if an id attribute exists, it is an object that exists in real world. This property is used in the conversion process to create graph nodes.

```
<track id="tr30">
        <trackTopology>
        <trackBegin id="x2507" pos="0.000000" absPos="434513">
                <bufferStop id="bs2507" name=""/>
        </trackBegin>
        <trackEnd id="y2491" pos="184.000000" absPos="434697">
            <connection id="co2491_2" ref="co2491_1"/>
        </trackEnd>
    </trackTopology>
    <trackElements>
        <speedChanges>
                <speedChange id="spu2507" pos="0.000000" absPos="434513"
↪   dir="up" profileRef="sppr4" vMax="200"/>
            <speedChange id="spd2491" pos="184.000000" absPos="434697"
↪   name="SKR9" dir="down" profileRef="sppr4" vMax="200"/>
        </speedChanges>
        <gradientChanges>
                  <gradientChange id="gr2507" pos="0.000000"
↪   absPos="434513" slope="0.400"/>
        </gradientChanges>
        <radiusChanges>
            <radiusChange id="ra2507" pos="0.000000" absPos="434513"
↪   radius="0.000000"/>
        </radiusChanges>
    </trackElements>
    <ocsElements>
            <signals>
            <signal id="si2499" pos="121.000000" absPos="434634"
↪   dir="up" name="ISK2" ocpStationRef="ocp-HLSK" type="other:45"
↪   sight="400" virtual="false" code="45"/>
            <signal id="si2501" pos="121.000000" absPos="434634"
↪   dir="down" name="ISK2" ocpStationRef="ocp-HLSK" type="other:45"
↪   sight="400" virtual="false" code="45"/>
        </signals>
    </ocsElements>
</track>
```

Figure 2 An example RailML track description (source: 1a hleskovac-draganici.railML)

RailML files are the main data source for generating the initial graph-based database of the GoSafe data management system as it provides information about tracks and all directly related elements. To this end, the RailML Infrastructure subschema needs to be converted to a graph.

For the graph to be correct, it needs to be connected so that paths exists between all nodes

stored in the database. Figure 5 shows an example line of three 3 tracks T1, T2, T3. Each track has a start node (TB) and an end node (TE). Tracks can also have track elements or connection elements like switches (SW) (T1 has SW1 and T3 begins at SW1). A mapping of this information from a RailML file into the model of a graph database can be found in Figure 3.

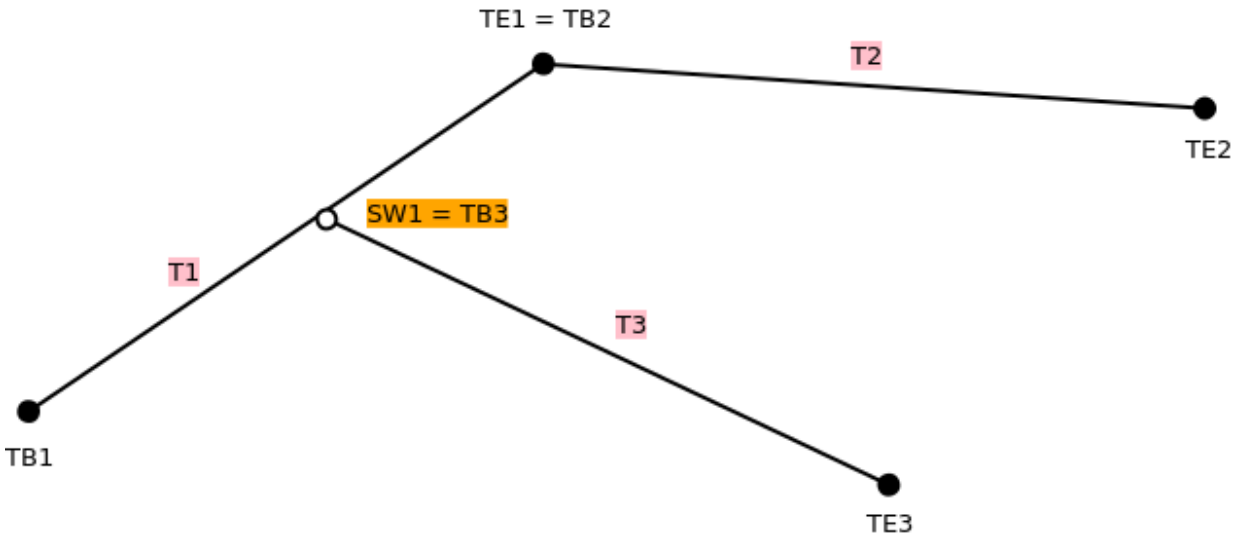

Figure 3: An example railway network represented by connected tracks and other elements.

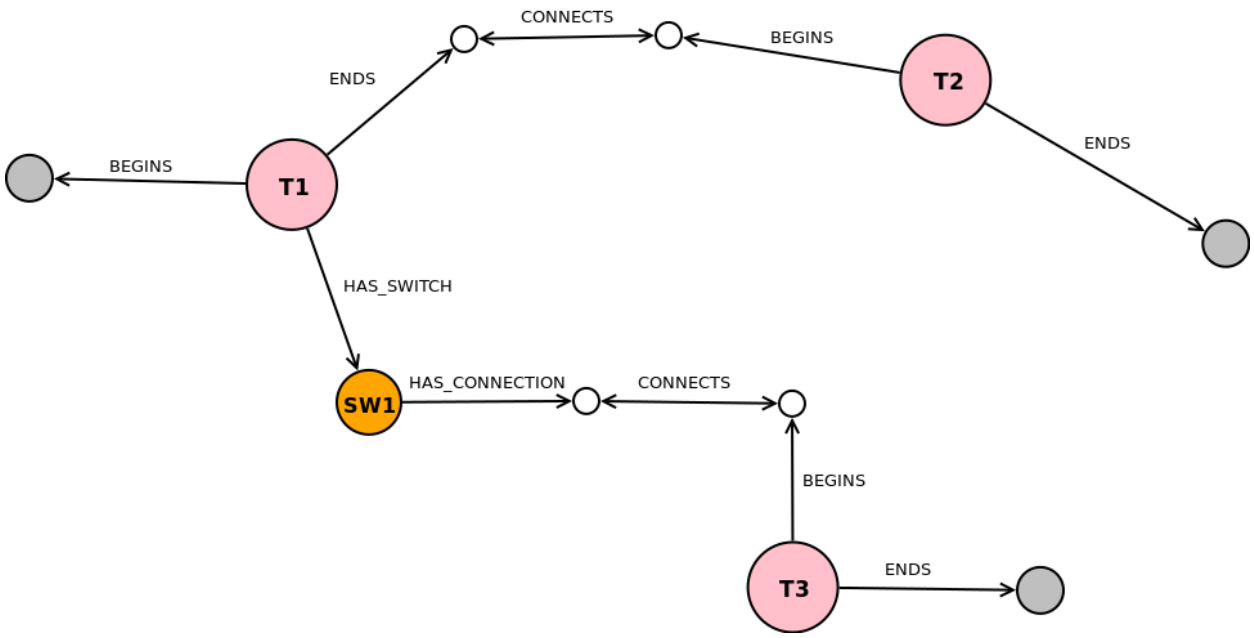

Figure 4: An example railway network converted to the graph.

Figure 4 depicts eleven nodes. three track nodes, one switch node, four connections nodes, and

three buffer stop nodes. Arrows indicate a relationship between two given nodes. In the graph database a node is a simple JSON object with key-value pairs, where both (key and value) are strings, such as the following example illustrates:

```
{
        "id": "ABD-DEF-123",
        "name": "Track 1",
        "description": "Berlin-Munich",
        "geometry": "LINESTRING(0 0, 0 2, 1 5, 3 10)"
}
```

The above properties were extracted from the following tag within the RailML XML file:

<track id="ABD-DEF-123" name="Track 1" description="Berlin-Munich">

We suggest including one additional property “geometry” to store additional information with a node that allows for the visualization of the node in 2D and 3D GIS systems.

Next to modelling start and end points, we also suggest that each connection between these points (track or section) are modelled as nodes. In this way a topological model, can be established. In such topological models, finding connected objects is fast and efficient and can be performed by a simple Cypher query (Figure 7) instead of a complex spatial query as in traditional GIS systems. Table 1 summarizes all suggested elements that should be modelled within a graph-based database system.

Table 1: Suggested nodes and relations to represent a railway network witin a graph database

Name	From	To
BEGINS	Track	Buffer Stop, Connection, Open End, Macroscopic Node
ENDS	Track	Buffer Stop, Connection, Open End, Macroscopic Node
HAS_SWITCH	Track	Switch
HAS_CROSSING	Track	Crossing
HAS_TRACK_ELEMENT	Track	Axle Weight Change, Bridge, Clearance Gauge Change, Electrification Change, Gauge Change, Gradient Change, Level Crossing, Operation Mode Change, Owner Change, Platform Edge, Power Transmission Change, Radius Change, Service Section, Speed

		Change, Track Condition, Train Protection Change, Tunnel
HAS_OCS_ELEMENT	Track	Signal, Train Detection Element, Balise, Train Protection Element, Stop Post, Derailer, Train Radio Change
HAS_CONNECTION	Switch	Connection
CONNECTS	Connection	Connection
HAS_TRACK	Line	Track
HAS_ATTR_GROUP	Line	Infra Attr Group
HAS_INFRA_ATTRS	Infra Attr Group	Infra Attributes
INFRA_ATTR	Infra Attributes	Axle Weight, Electrification, Gauge, Clearance Gauge, Operation Mode, Owner, Power Transmission, Speeds, Train Radio, Train Protection
HAS_SPEED	Speeds	Speed

In summary, the above suggested graph-based approach provides a hierarchy, which guarantee that all nodes are organised into ordered lines which are collections of tracks with all related assets. Described relationships connect only nodes from the Infrastructure Subschema. Extending a graph with other subschemas can be done without changing the existing structure as the following sub-sections will illustrate for the main data standards and sources required for railway asset management.

5.4. CityGML

CityGML is an open standardised data model and exchange format to store digital 3D models of cities and landscapes. It defines ways to describe most of the common 3D features and objects found in cities (such as buildings, roads, rivers, bridges, vegetation and city furniture) and the relationships between them. It also defines different standard levels of detail (LODs) for the 3D representation of these objects, which allows to represent objects for different applications and purposes [3]. Within the overall data management standard these objects can be stored within the geometry property of the nodes within the graph database. An alternative to directly storing the nodes, is to store a link to objects stored within a dedicated CityGML server.

There are 11 modules[1] in CityGML and most of them describe features that are not directly related to a railway network, but that are important for asset managers:

- Bridge: bridge-related structures, possibly split into parts
- Building: the exterior and possibly the interior of buildings with individual surfaces
- that represent doors, windows, etc.
- CityFurniture: benches, traffic lights, signs, etc.

1

12th is the Appearance module, but it contains only visual aspects of features from other schemas

- CityObjectGroup: groups of objects of other types
- Generics: other types that are not explicitly covered
- LandUse: areas that reflect different land uses, such as urban, agricultural, etc.
- Relief: the shape of the terrain
- Transportation: roads, railways and squares
- Tunnel: tunnels, possibly split into parts
- Vegetation: areas with vegetation or individual trees
- WaterBody: lakes, rivers, canals, etc.

Of these objects, the objects Building, LandUse, Vegetation, WaterBody, Tunnel are the most valuable for the railway management process and should, therefore, included within a information management system.

Each feature contains various properties and geometry. Geometry depends on specific LODs. For example, the geometry of buildings in LOD 0 or 1 is only described by a footprint, while in LOD 2 to 5 there are much more detailed surfaces that describes different parts of the building, such as roofs or walls. In the current state, the developed GoSafe Converter extracts always only the footprint which allows for the visualization of buildings as simple height extrusions (Figure 5 LOD1 buildings (visualized in the GoSafe). However, more complex geometrical information about areas and polygons of exact triangulated shapes can also be extracted and integrated within information management systems (Figure 4).

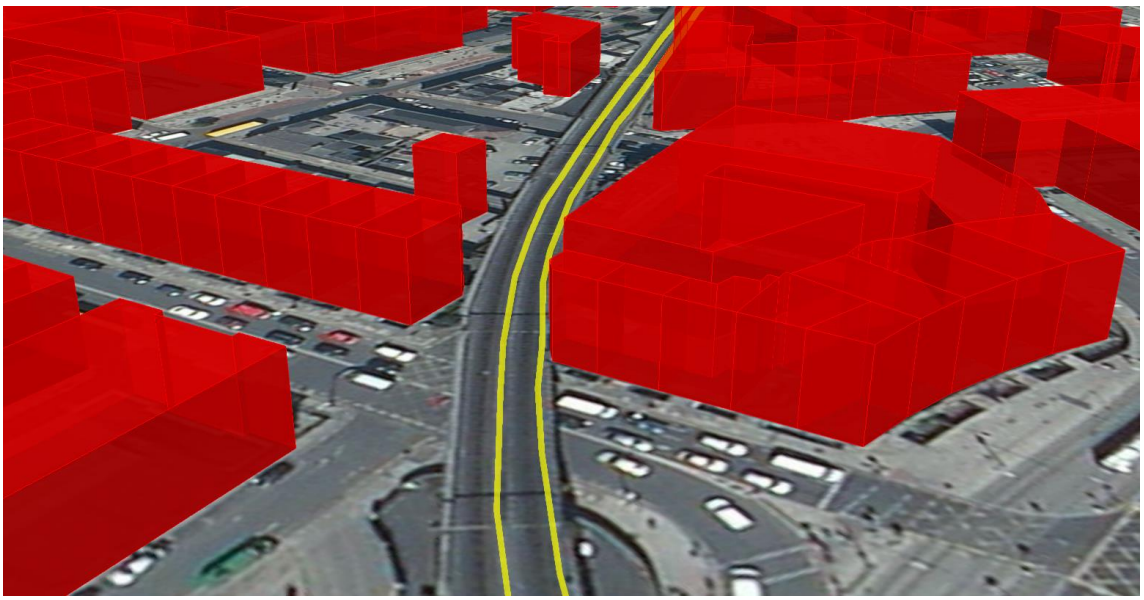

Figure 5 LOD1 buildings (visualized in the GoSafe Information Management Platform)

The extraction of geometry describing specific areas are also represented in different LODs within the CITYGML standard. In LOD0 surfaces are lines while LOD 1 to 5 represents these objects by more and more accurate polygons. Independent of the type of objects, the geometry

and additional information that is represented about objects can be linked to the graph database. This allows for the targeted querying of this information.

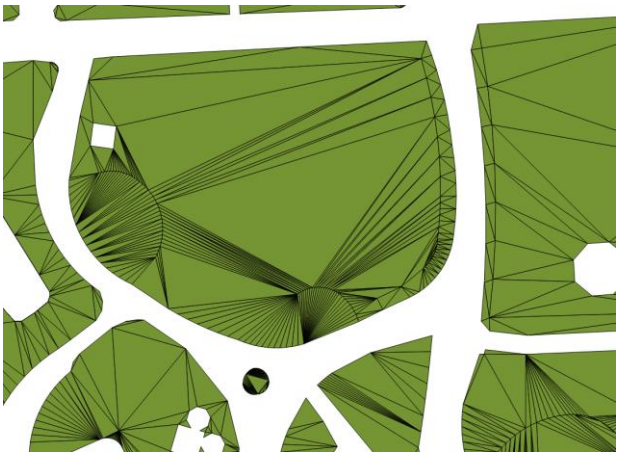

Figure 6 Triangulated surfaces displayed in GIS software

5.4.1.1. IFCRail

As mentioned, at this stage of development the IFCRail project is delayed. However, the data model can be extended with this type of information without making any changes to existing structure.

5.5. Vector formats

The graph-based database structure also allows to integrate Geojson and Esri Shapefiles to the neo4j spatial layer. For each feature within these vector format files all properties and geometry are saved. It is possible to extend the number of available formats as the data structure only needs to store the geometry and additional properties associated as a table or any other format. We assume that these layers are not directly related to the railway network and we can get features only when we need them with a simple spatial query.

In the data model objects extracted from these data sources are treated the same way as objects extracted form CityGML files and can be attached as geometry properties to nodes.

5.6. Other data sources

Delimited text files like CSV (comma separated values) or TSV (tab separated values) can be ingested into existing graph nodes. To increase the amount of information describing the single object. Each line (row) in a delimited text file represents an array of values. Identifying a target node is possible due to unique id. Each file must contain a column with unique IDs, then if a node with the given ID exists, new properties will be added. This approach is an automatized way to edit nodes. Users can carry out the same operation using only the user interface and modifying

nodes manually.

5.7. Objects within the proximity of a railway network

Different data sources than those directly connected to objects within a railway network can also be converted to the graph and stored in the same database as railway network objects. To this end, a link between railway objects and these other objects within the neighbourhood of the railway object can be established through their geographical position. For the graph-based database Neo4J we use to develop the GoSafe Railway platform spatial extension modules are available that allow for the inclusion of such additional objects. These extensions provide dedicated query languages to find all objects from within given tracks neighbourhood (for example all objects within a 200m distance of a specific railway node). For these queries to be fast, spatial indices are created by these extensions (such as the RTree index used by the Neo4J spatial extension).

To support the generation of these indices, additional spatial layers need to be created in the database. These layers are structures, wich provide bounding boxes for multiple nodes stored in the database graph. Each additional feature can then be connected to such a bounding box. Finally, these RTree structures then allow to find spatially close objects very fast, without the requirement to fully search an entire database and to do sophisticated spatial comparisons. Figure 7 depicts a spatial layer for Buildings within a graph-based database.

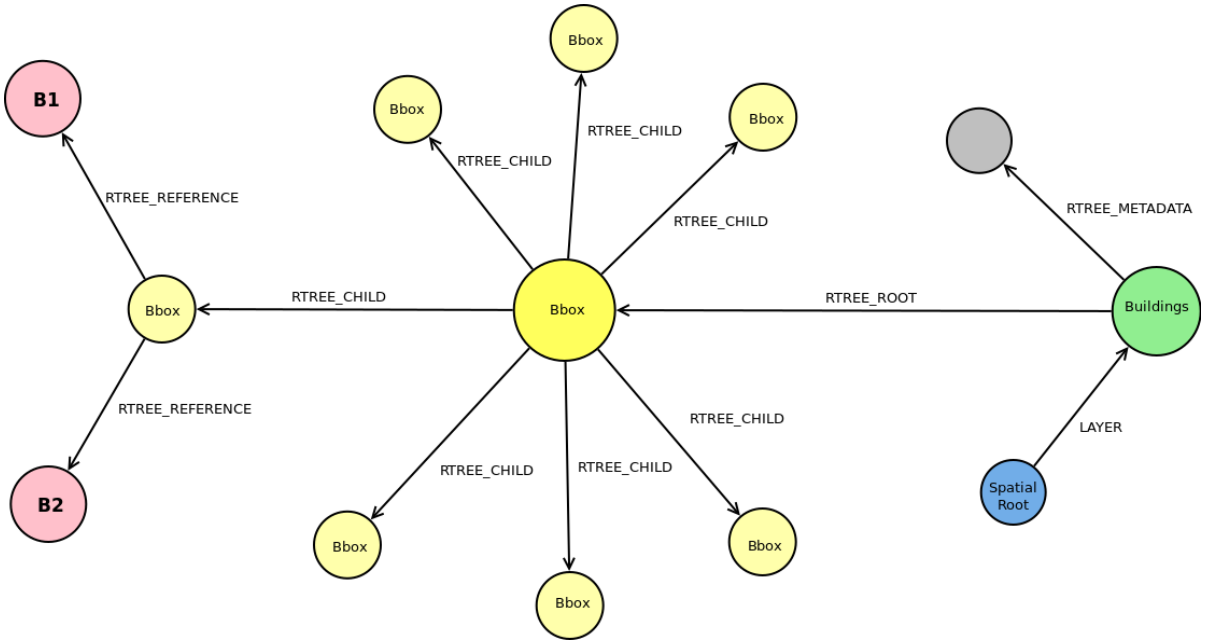

Figure 7 Spatial layer Buildings organised into RTree structure.

In this figure, the B1 and B2 nodes represent buildings that are within a given bounding box.

Spatial root nodes connect all spatial layers in the database. Such RTree structures are very useful when we want to find all neighbours of a given track node, because to do this we need to search only one bounding box, instead of iterating over all objects. A query can then quickly find the bounding box nodes that are related to a specific railway object and extract all elements that are stored within this bounding box and that are in a specific neighbourhood of the respective railway object.

6. Conclusions

The here suggested data management standard that builds upon graph databases can ingest different formats containing railway network information and spatial data. We suggest that RailML will be used as the main data source describing the railway network. Other objects related to railway networks can be extracted from CityGML or other GIS based sources.

Converted files create graphs with general structure illustrated in the picture below (Figure 9):

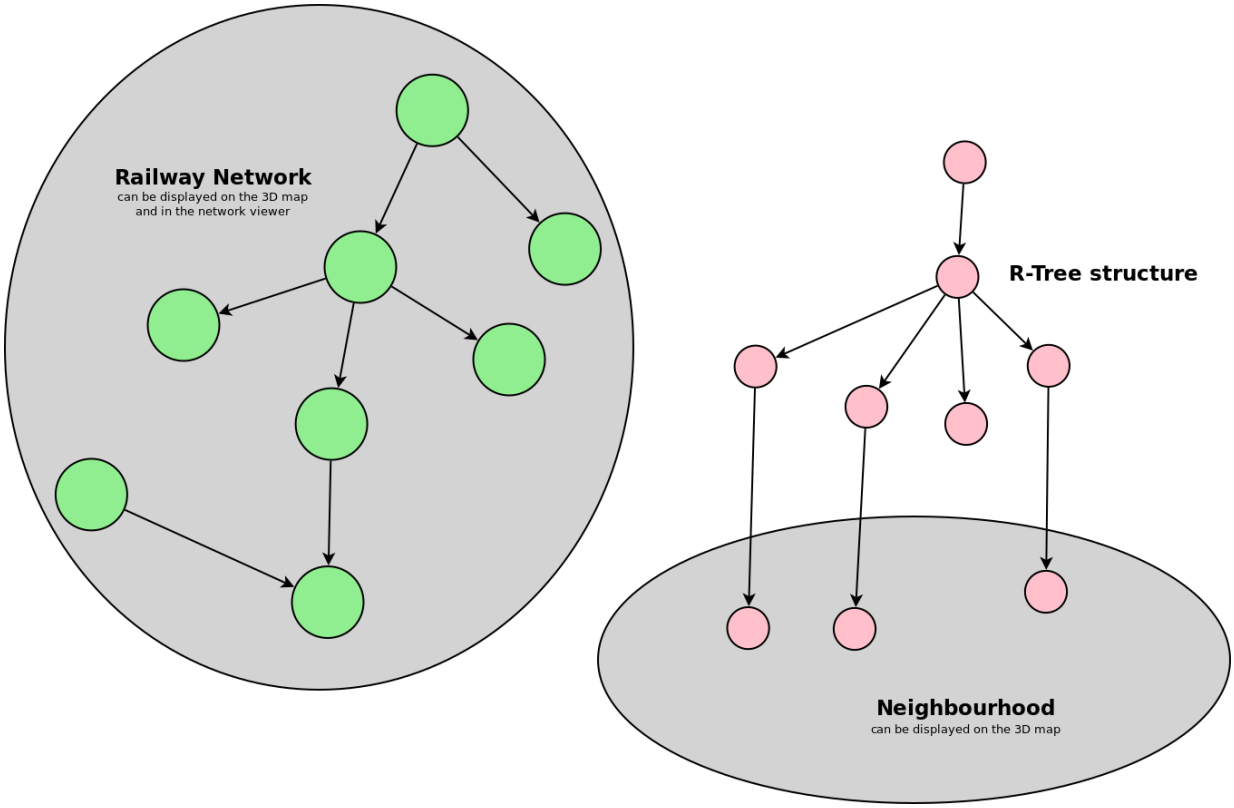

Figure 8 Graph model structure. Railway network elements and the neighbourhood.

Graph structures provide an easy to understand and navigate interface. Raw RailML and CityGML files that usually contain the asset related information about and around railway networks are usually big and their structure is hard to understand without a proper documentation. Graphs are more user friendly and can be modified in a simple graphical interface. The data model is very elastic and can be extended by adding new nodes and relationships between them and these nodes which exist at the current stage. Any type of file format and database can be integrated to the graph model by using additional properties and relationships. Objects around railway networks can be integrated in the neighbourhood tree structure. In the future the most desirable improvements will be the integration of IFCRail to describe physical railway network information more accurate than RailML allows to.

7. References

[1] Neo4j official website, retreived 30.06.2018 from https://neo4j.com

[2] RailML wiki, retrieived 30.06.2018 from https://wiki.railml.org/index.php?title=Main_Page

[3] CityGML official website, retreived 30.06.2018 from https://www.citygml.org