European Commission

JRC TECHNICAL REPORTS

Risk Data Hub software and data architecture

Solutions and tools for disaster risk management

Antofie, Tiberiu Eugen

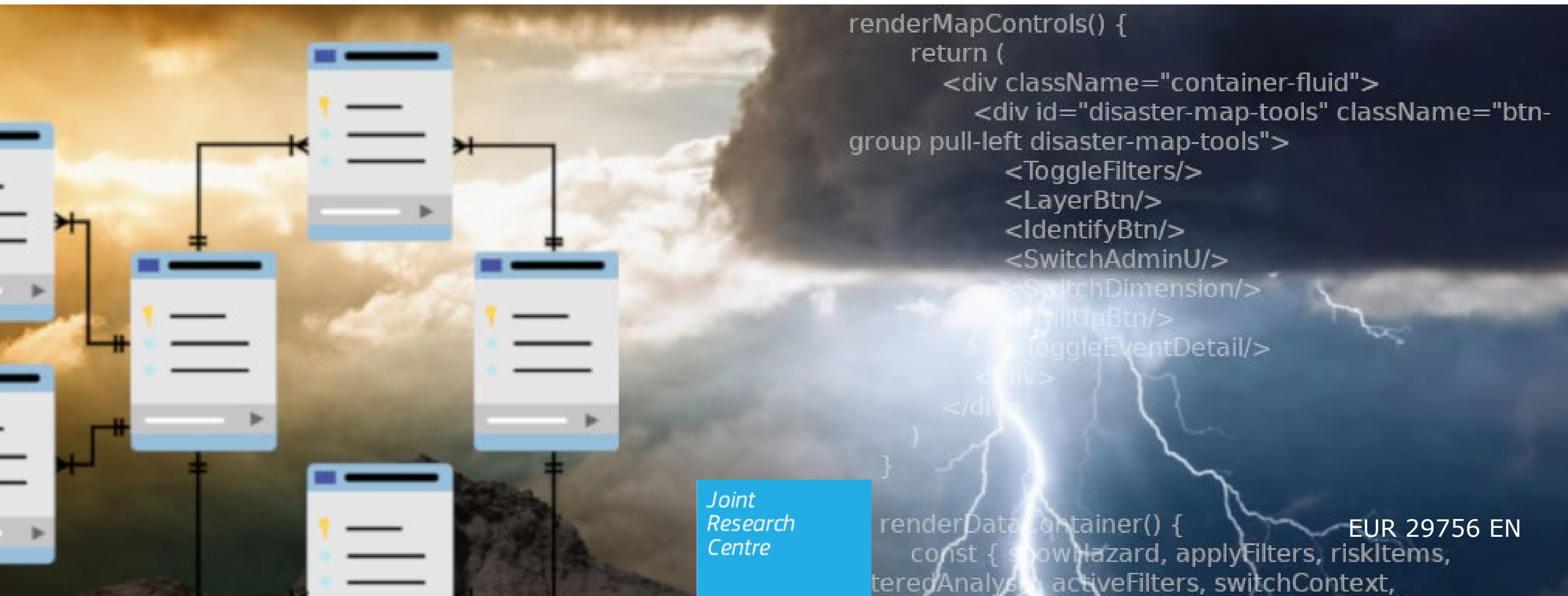
Luoni, Stefano

Faiella, Anna

Ríos Díaz, Francisco

Marín Ferrer, Montserrat

2019



Joint Research Centre

EUR 29756 EN

Contact information
Name: Tiberiu-Eugen Antofie
Address: European Commission, Joint Research Centre (JRC)-JRC.E.1, Via Enrico Fermi, 2749 – 21027 Ispra (VA), Italy
Email: Tiberiu-Eugen.ANTOFIE@ec.europa.eu
Tel.:+39 0332 789363

JRC Science Hub
https://ec.europa.eu/jrc

JRC114712

EUR 29756 EN

PDF	ISBN 978-92-76-03978-5	ISSN 1831-9424	doi:10.2760/67169

Luxembourg: Publications Office of the European Union, 2019

How to cite this report: Antofie, T., Luoni, S., Faiella, A., Rios Diaz, F., Marin-Ferrer, M., *Risk Data Hub software and data architecture*, EUR 29756 EN, Publications Office of the European Union, Luxemburg, 2019, ISBN 978-92-76-03978-5, doi:10.2760/67169, JRC114712.

Authors affiliations:

Tiberiu-Eugen Antofie, European Commission, Joint Research Centre (JRC), Ispra, Italy

Stefano Luoni, Unisystems Italy, external service provider of European Commission, Joint Research Centre (JRC), Ispra, Italy

Anna Faiella, trainee at the European Commission, Joint Research Centre (JRC), Ispra, Italy

Francisco Ríos Díaz, European Commission, Joint Research Centre (JRC), Ispra, Italy

Montserrat Marín Ferrer, European Commission, Joint Research Centre (JRC), Ispra, Italy

Contents

Abstract

The Risk Data Hub (RDH) is an initiative of the Disaster Risk Management Knowledge Centre (DRMKC) and consists of a publicly available web GIS platform intended to improve access to and sharing of curated Europe-wide risk data, tools and methodologies for fostering disaster risk management (DRM)-related actions.

The implementation of the concept is comprised of multiple steps, including the definition of the type of analysis to be presented, the design of methodologies to compute data needed, the design of database architecture and software tools and finally the development of the software.

This document will focus on the design of software architecture, starting from a high level analysis of the business needs, moving to an explanation of the solutions proposed, considering previous works on the topic of DRM and showing how the existent loss database architecture has been extended to fit the requirements of a complex and multi-context application.

Role of the authors

Tiberiu Eugen Antofie, as author, provided methodological examples, bringing together information associated with the loss and damage data, and contributed to structuring the historical event catalogue.

Stefano Luoni, as external consultant and co-author, adapted the database architecture to match the RDH needs and developed the RDH platform.

Anna Faiella, as trainee, contributed to the review of data and the report.

Francisco Ríos Díaz, as contributor, was responsible for designing the concept of the database architecture behind the RDH and helping prepare the report.

Montserrat Marin Ferrer, as the coordinator of the DRMKC projects, oversaw the whole process.

1. Introduction

Despite the great number of projects developed in the context of DRM, there are no widely shared resources to analyse disaster risk data, as every country has its own databases and organisations, with different levels of usage and effectiveness. With the RDH, the European Commission wants to offer a common platform to access innovative tools and methodologies, granting more equity to those who decide to adopt it.

To understand what this application is about, we can start with a couple of considerations.

In the first place, working with risk data means dealing with hazards: the RDH has a multi-hazard approach, implementing methodologies to present data about different hazards, both one at a time and altogether. While the first datasets introduced are related to natural hazards, technological hazards and all kinds of man-made disasters are also to be involved, with the final aim of having a complete map of risk, including both direct and indirect impacts.

The RDH is also multi-context, as it can be used to analyse exposures and vulnerabilities, as well as showing historical events. This means that on a unique platform, the user is allowed to discover the most exposed and vulnerable areas for every hazard, verify and compare real impacts, perform statistical analysis, find trends, check eligibility for solidarity fund requests and more.

The RDH is intended to be a **'second home' for research results**, satisfying the need to make them accessible. This purpose may be clarified by defining input and output for this platform.

Input is granted by scientific partnerships where this platform represents real added value, as it improves interoperability by connecting different sources and sharing their data.

Output consists of different analyses performed on available data, as the implementation of specific methodologies that should be useful for multiple policies. According to the vision for the RDH, as the main resource for accessing risk data analysis, its usage will enhance coherence across portfolios.

The rest of this document explains how these main concepts are applied by the software.

A more complete explanation of methodologies developed for the RDH and its relation to policies is included in the JRC technical report 'Risk Data Hub — web platform to facilitate management of disaster risks'.

2. Main challenges and solutions identified

Before implementation can begin, all concepts, methodologies and business needs in general have to be translated into technical requirements. This chapter contains a simple explanation of challenges and solutions proposed for the development of the software.

2.1. Dealing with uncertainty of data

It is a fact that none of the data available is 100 % correct, or gives us a certain value, because of different reasons.

First of all: the availability and accuracy of data on past disaster events are poor, mainly because data collection at local level is not homogeneous and is not shared with higher administrative levels for statistics. This is something confirmed by many research projects published on this topic and explains why it is difficult to collect and present data on damages and losses.

Then let us take the models: of course they cannot give us certainty, as they are used to predict future events and have a probabilistic approach. There is another problem, though: they are shaped by identifying trends in past events, and as data on past events are scarcely available, how much can we trust them?

This challenge, along with the solution proposed, represents a main critical concept of which most of the next points are the logical consequence.

Solution

The RDH does not offer early warning support; instead, it presents **pre-event** and **post-event** data, where pre-event data come from models and post-event data come from collections of past events.

Having stated that data always contain uncertainty, the RDH tries to give a better overview by **comparing data from different sources**. The idea is to provide the user of the platform with data from both models and archives of past events, for each event presented. This way, it is easy to spot anomalies (e.g. only one value which differs from others), identify which source is more accurate overall and give at least probabilistic values if no assessed damages are available. Moreover, having a collection of past events which is as complete a possible should help better fine-tune the models and their forecasting capacities.

2.2. Representing exposures, vulnerabilities and historical events together

The RDH aims to handle different types of datasets, making them available on a unique portal to help end users in many tasks related to DRM.

Basically, data used come from models, or archives of past events. While models tell us what could happen, past events are something that already occurred, on a specific date, under specific circumstances; hence the visualisation of these two types of data cannot be the same.

Data also refer to many **natural** hazards (**technological** hazards are not included yet, but future implementation is foreseen), and every hazard has its own peculiarities; that is why data input differs from hazard to hazard.

So the first challenge is actually doubled, and consists of storing different types of data in a single database and presenting them in a way that preserve their specificity in a single user interface.

Solution

The database should include a main entity **damage assessment** which, along with the overall flexible design, lets the system manage and present data about different types of analysis. A more detailed explanation is included in Chapter 4 of this document.

2.3. Harvesting data from multiple sources

After considering uncertainty (see Chapter 2.1) and working with such a wide area of interest, it is clear why a single source of data is not enough. The RDH works with many scientific partners that provide the application with the output of their work. These are typically models used for populating the risk analysis datasets of the RDH, but sometimes archives of past events are included.

While models have good coverage and are produced on a regular basis by scientists, the collection of loss data is something that is not homogeneous, nor well defined and structured; that is why data availability is poor, especially on a large scale.

Scientific, economic and political issues that cause this poor availability of data are not considered in this document. Technically speaking, one identified way of getting as much data as possible is to connect with different sources.

Solution

This challenge leads to the development of a dedicated data integration flow for each data source activated. The RDH has an extract, transform, load (ETL) layer that is needed to transform data extracted before inserting them into the database.

Important note: at this time, all data used by the system are stored in its own database. This approach may be considered a downside and criticised as it duplicates data already existent on external resources. As will be better explained later in this document, most of the data managed by the system, particularly events, need to be transformed and/or validated and this is not something that can be done on the fly, for different reasons. First of all, performance: complex processing of a whole dataset at every page request does not make sense as would be a waste of system resources and it would cause a dramatic fall in the overall performance. The second reason concerns the validation process: many events need to be moderated, as they often contain wrong or incomplete information and this operation should be done one at a time. The third reason concerns data availability: not every source exposes services, so data have to be downloaded en masse before being able to use them.

Having said that, using external services to access particular layers or features on the fly is something that is convenient and will be certainly integrated for specific data sources.

The RDH's main scientific partners are worth mentioning (see the 'References' section for details):

- European Flood Awareness System (EFAS)
- European Forest Fire Information System (EFFIS)
- European Drought Observatory (EDO)
- Global Human Settlement Layer (GHSL)
- Copernicus
- Global Disaster Alert and Coordination System (GDACS)
- Europe Media Monitor (EMM)

- Emergency Events Database (EM-DAT)
- Historical Analysis of Natural Hazards in Europe (HANZE) database.

2.4. Identification and classification of events

It is not clear how an event should be identified: actually it is still a matter of discussion in the scientific community, and it is not homogeneous among different hazards. From the RDH point of view, the problem is not about trusting a specific source of data; rather, since the system extracts events from multiple sources, specific criteria for identifying events are necessary to avoid duplications.

Solution

The logic used by the RDH is as follows.

- An event is identified by the **hazard**, **date** and **country,** and optionally by a smaller administrative unit. This means that, for example, a single meteorological event which covers an area shared by two countries will generate exactly two events in the system, while more events could be identified on the same date and country if the causes are different. For events harvested from external sources with no clear information on the cause, only one event would be generated per hazard, date and country.
- An **event** is a macro entity that may include multiple **phenomena**. This means that while an event can be associated for example with a whole country and can last several days (or weeks), there are single phenomena that map the event at a more specific location and on a more specific date, such as single burned areas of a vast forest fire.

2.5. Unique coding of events

Every data source has its own way of assigning a code to events; furthermore, events coming from various sources may overlap, hence a new code has to be assigned to keep a consistent archive.

Solution

In the RDH, the code composition is implemented as follows:

[Hazard] (code of two characters)

+ [Country] (ISO 3166-1 alpha-2 code of country)

+ [Begin date] (in YYYYMMDD format)

+ [Glide number] (four-digit serial number).

An example would be:

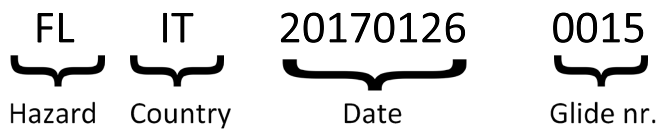


Figure 1. Example of event code.

When imported into the system, every event has 'draft' status and it needs moderation to be published. Only when the event is approved is the RDH code generated; this allows it to be consistent with the sequence of glide numbers of published events.

2.6. Country corners and user privileges

The RDH publishes Europe-wide datasets, but the whole system is designed to also work at national or regional level. This is a fundamental part of the concept of the RDH: **data should always be linked to administrative divisions and should be collected at local level**.

While the JRC is able to produce and/or find data with good coverage of the whole of Europe but that is quite generic in nature, local institutions likely have access to more detailed data and should be able to use them on the RDH platform for their own DRM purposes.

A 'country corner' works like a separate instance of the RDH, as it implements the same methodologies in a different hierarchy of locations. A single institutional user who is responsible for its country will upload data and choose whether or not to share these data with other users or groups.

Solution

The logical solution proposed is quite simple and is based on two main points:

- a user belongs to one or multiple groups and each group has some basic permissions;
- each dataset in the system has a unique owner that can set visibility and permissions for it.

Let us report a couple of examples to clarify this logic a bit.

Example 1. The group of administrators of the Austrian country corner has privileges for managing all datasets assigned to the Austria region. A user who belongs to this group uploads two layers, deciding to grant viewing rights for all groups for the first layer, and all rights only for the group 'Austria Administrators' for the second layer. After this, the first layer will be visible to every user (but not editable), even if not registered to the site (because it belongs to an anonymous group); the second layer will be visible and editable only by users in the group 'Austria_Administrators'.

Example 2. A country corner administrator uploads data for a new damage assessment, and chooses to grant editing rights to the administrator group and only viewing rights to the group of non-admin users of the country corner of reference. After this, a user who is not logged in, or a user of another country corner, will not see anything of that damage assessment.

2.7. Scalability and performance

Since the RDH is expected to store and manage large amounts of data, scalability is a matter to be addressed to keep the application healthy and responsive.

This document is not a technical guide, nor a list of design patterns in Python or any other language. Here we want just to state that performance is something taken into consideration, and in this regard there are some practices or tools already in use, as well as others to be applied in the near future.

Solution

- Database indexes.
- Optimisation of queries.
- Use of GeoWebCache: this is a tool that comes with GeoServer and caches tiles generated by Web Map Service (WMS) calls. Tiles can be cached both after a call to the WMS, or by a bulk seeding process.
- Caching of Django views: Django integrates a configurable caching system for its views, so multiple page requests would consume resources only once after the cache expiration.
- 'Reselect' tool for React: the client application keeps data retrieved from the back-end API in its own internal 'store' and would make a new call to the API only if data is not already in it; this saves both bandwidth and system resources.

The following are still to be done.

- Deployment of GeoServer on a dedicated machine.
- Use of NoSQL database: when data stored start to exceed a certain amount, old-fashioned relational databases start to suffer a degradation in their performance. The use of a NoSQL database should solve this problem, but at this point the technology selection process is not complete, as there are several constraints to be considered regarding GeoNode and GeoServer.

3. Technologies used

This chapter is about technology selection and architectural design of the RDH application.

3.1. Previous works

Having a look at works already done on the same topics is useful for identifying tools that have proven to work well and best practices for using them.

3.1.1. GeoSAFE

GeoSAFE is a web platform that makes it possible to run InaSAFE analyses online. InaSAFE is free software that produces realistic natural hazard impact scenarios for better planning, preparedness and response activities. It provides a simple but rigorous way to combine data from scientists, local governments and communities to provide insights into the likely impacts of future disaster events.

Initiative of the government of Mozambique and the World Bank. Based on GeoNode (and GeoServer).

3.1.2. Rapid Analysis and Spatialisation of Risk

The Rapid Analysis and Spatialisation of Risk (RASOR) project is developing a platform to perform multi-hazard risk analysis for the full cycle of disaster management, including targeted support and critical infrastructure monitoring. A scenario-driven query system simulates future scenarios based on existing or assumed conditions and compares them with historical scenarios. Initially available over five case study areas, RASOR will ultimately offer global services to support in-depth risk assessment and full-cycle risk management.

Developed by CIMA research foundation. Uses GeoNode as layer catalogue.

3.1.3. ThinkHazard!

ThinkHazard! provides a general view of the hazards, for a given location, that should be considered in project design and implementation to promote disaster and climate resilience. The tool highlights the likelihood of different natural hazards affecting project areas (very low, low, medium and high), provides guidance on how to reduce the impact of these hazards, and where to find more information. The hazard levels provided are based on published hazard data, provided by a range of private, academic and public organisations.

Developed by Global Facility for Disaster Reduction and Recovery (GFDRR). Uses GeoServer.

3.1.4. Afghanistan Disaster Risk

A public platform for creating, sharing and accessing geospatial data and maps for decision-making about disaster risk. It includes two modules: one for risk analysis and one for cost–benefit analysis.

Developed by GFDRR. Based on GeoNode (and GeoServer).

3.2. Overall architecture

After gathering and analysing the main requirements of the platform to be developed, it was time to choose the technologies and tools to be used. Some of these choices were anticipated by the previous chapter and they were the result of checking previous works in this fields, as they pointed out that significant projects were based on GeoNode and GeoServer.

The system architecture as a whole is quite articulate and makes use of several tools to perform all operations needed. Basically, the project is built with Django (Python web framework), using GeoNode as dependency, PostGIS as database back end and a client application developed with ReactJS.

What is GeoNode?

GeoNode is a web-based application and platform for developing geospatial information systems (GISs) and for deploying spatial data infrastructures (SDIs). It can be integrated with third-party Django apps and implements a framework for web services compliant with the Open Geospatial Consortium (OGC).

What is GeoServer?

GeoServer is an open-source server for sharing geospatial data. Designed for interoperability, it publishes data from any major spatial data source using open standards. GeoServer is an OGC-compliant implementation of a number of open standards such as Web Feature Service (WFS), Web Map Service (WMS) and Web Coverage Service (WCS).

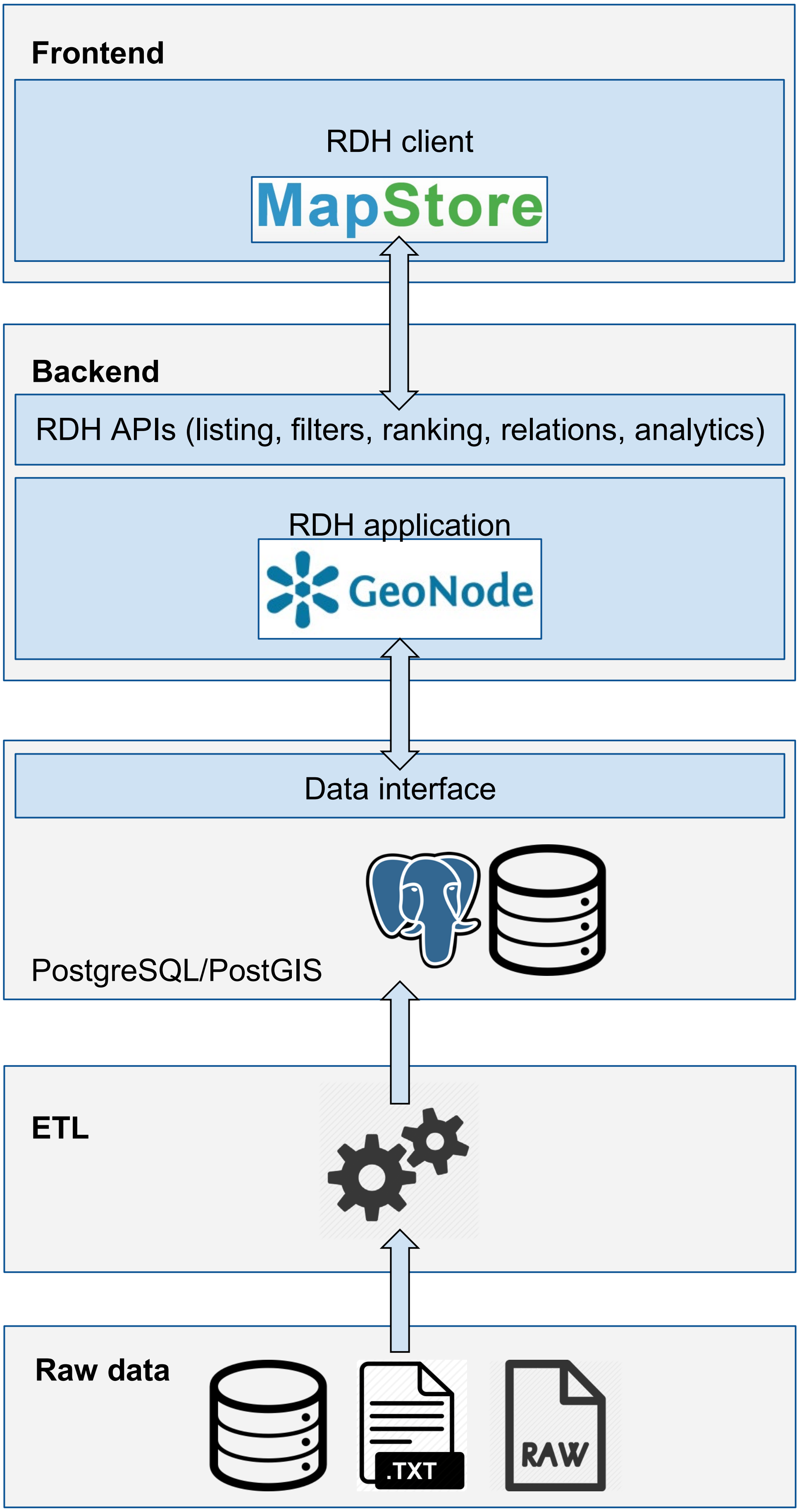


Figure 2. The RDH software architecture.

Let us a have a more detailed look at the individual pieces of architecture.

3.3. Data harvesting and extract, transform, load

Data is harvested from multiple heterogeneous sources and loaded into the RDH database by ad hoc Python scripts. Relevant operations involved in the data ingestion process are:

- definition and scheduling of importing jobs;
- a database health check;
- grouping of data into multiple layers;
- pre-calculation of relevant statistics;
- normalisation of taxonomies;
- checking and casting of geometry fields;
- creation of styles for different types of layers and geometries;
- import of GeoServer layers in GeoNode;
- insertion of keywords and categories from database view attributes;
- completion of title and description fields;
- definition of geofence rules.

3.4. Data interface

The basic operations performed by the RDH application against the PostGIS database are:

- data extraction and pre-processing (PL/pgSQL + Python code);
- spatial queries to extract spatial relations between datasets;
- extraction of administrative division boundaries.

The basic operations performed by the RDH application against GeoServer are:

- OGC/WMS service calls to view layers on map (*);
- (E)CQL to filter layers and contents on map;
- SLD for styling multiple geometries and geometry types;
- SLD filters for styling contents;
- geofence rules to restrict access to layers and services;
- GeoWebCache for tile caching;

(*) Specific layers are created in GeoServer by SQL views and are used to extract and filter data to show on the map.

3.5. Back end

GeoNode is mainly used for uploading and managing vector and raster layers. Its models and APIs are also used for:

- enriching original data with metadata and additional information (keywords and categories);
- supporting front-end functionalities;
- publishing a Catalog Service for the Web (CSW) catalogue of the layers;
- consuming GeoServer APIs for management commands;
- proxy WMS requests under access-control lists.

Inventory, analysis and loss data are loaded into a dedicated database that will be described later in this document.

3.6. Front end

The front end is based on the MapStore framework for web mapping and it uses some of its core components along with custom components to build the user interface. It is a single-page application developed with ReactJS and Leaflet maps.

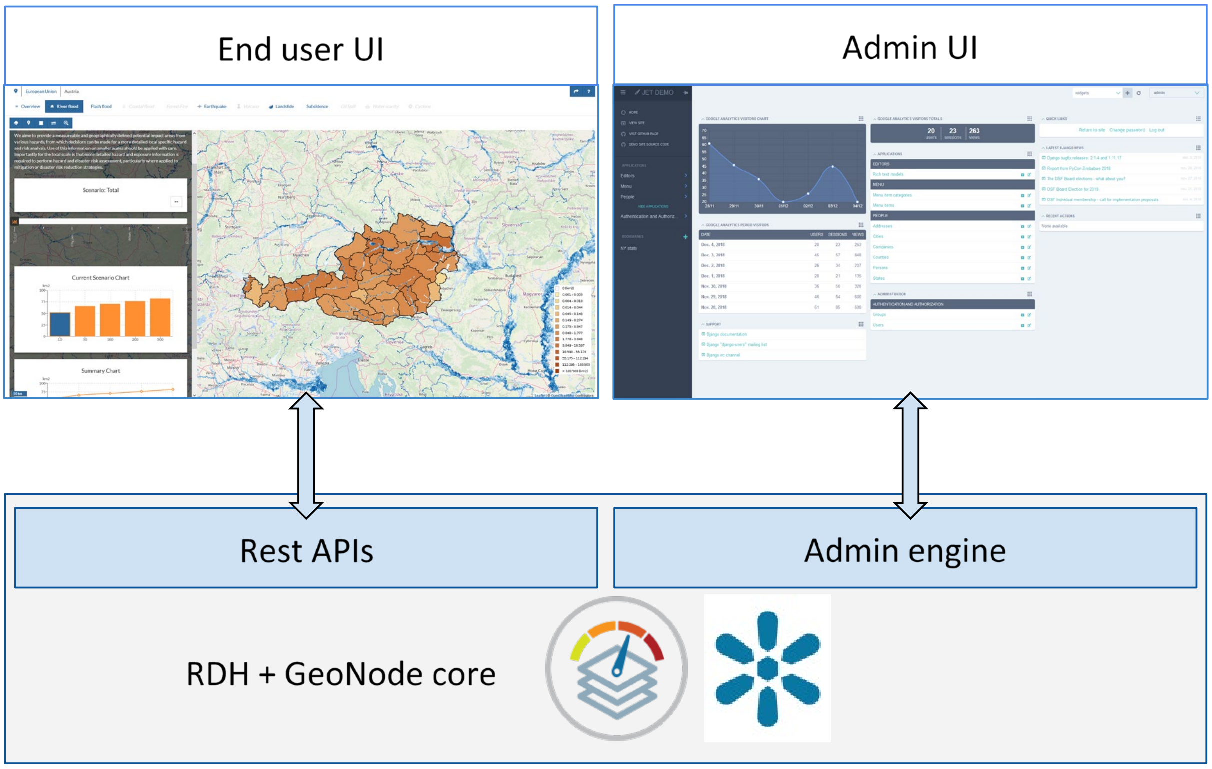


Figure 3. Interaction of user interfaces with back end.

4. Database architecture

4.1. Evolution of loss database architecture

The implementation of the base concept of the RDH required storing data for different purposes, such as risk analysis, inventory of assets and damage assessments.

The database was designed after the 'loss database for disaster risk management' proposed in a recent EU publication (http://dx.doi.org/10.2760/647488). The result was at the same time an abstraction and an extension of that model.

Changes introduced during the development of the RDH are also reported in the publication *Update of the DRMKC loss database architecture for disaster risk management*.

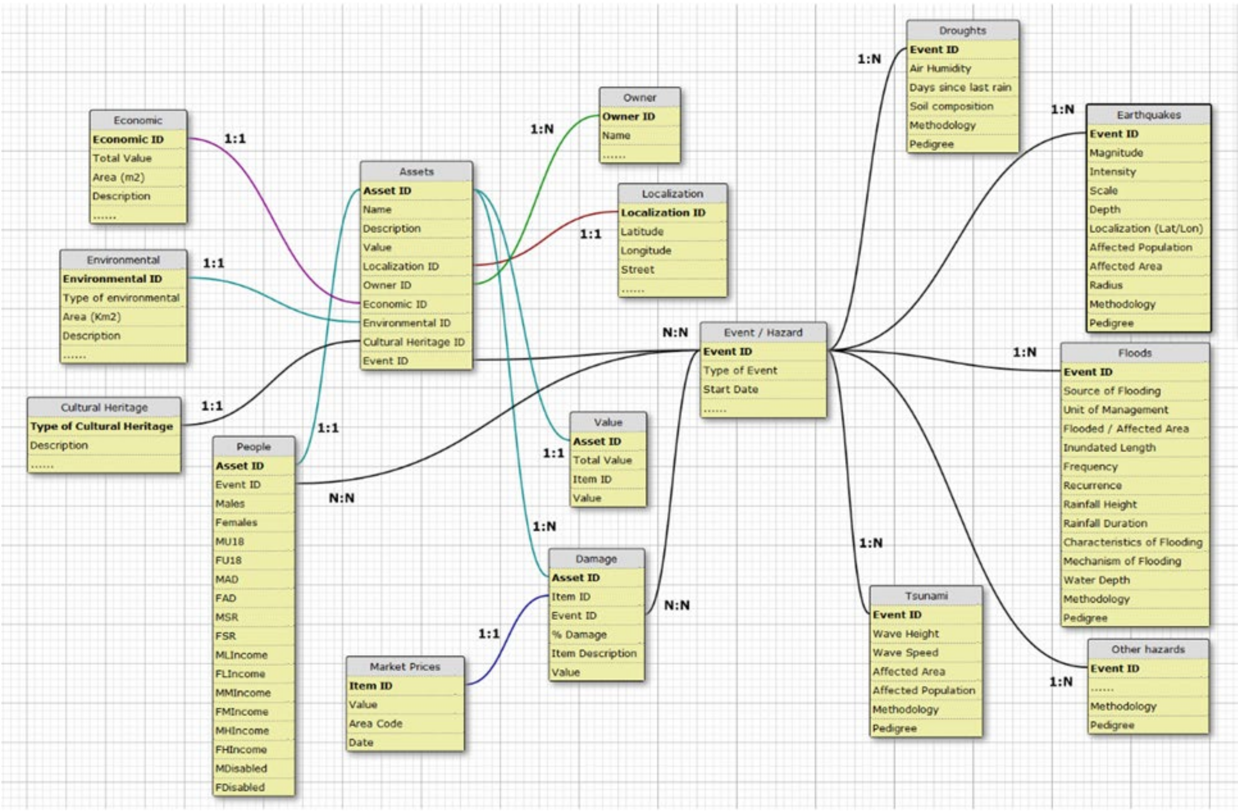


Figure 4. Loss database diagram as per EUR 29063 EN publication.

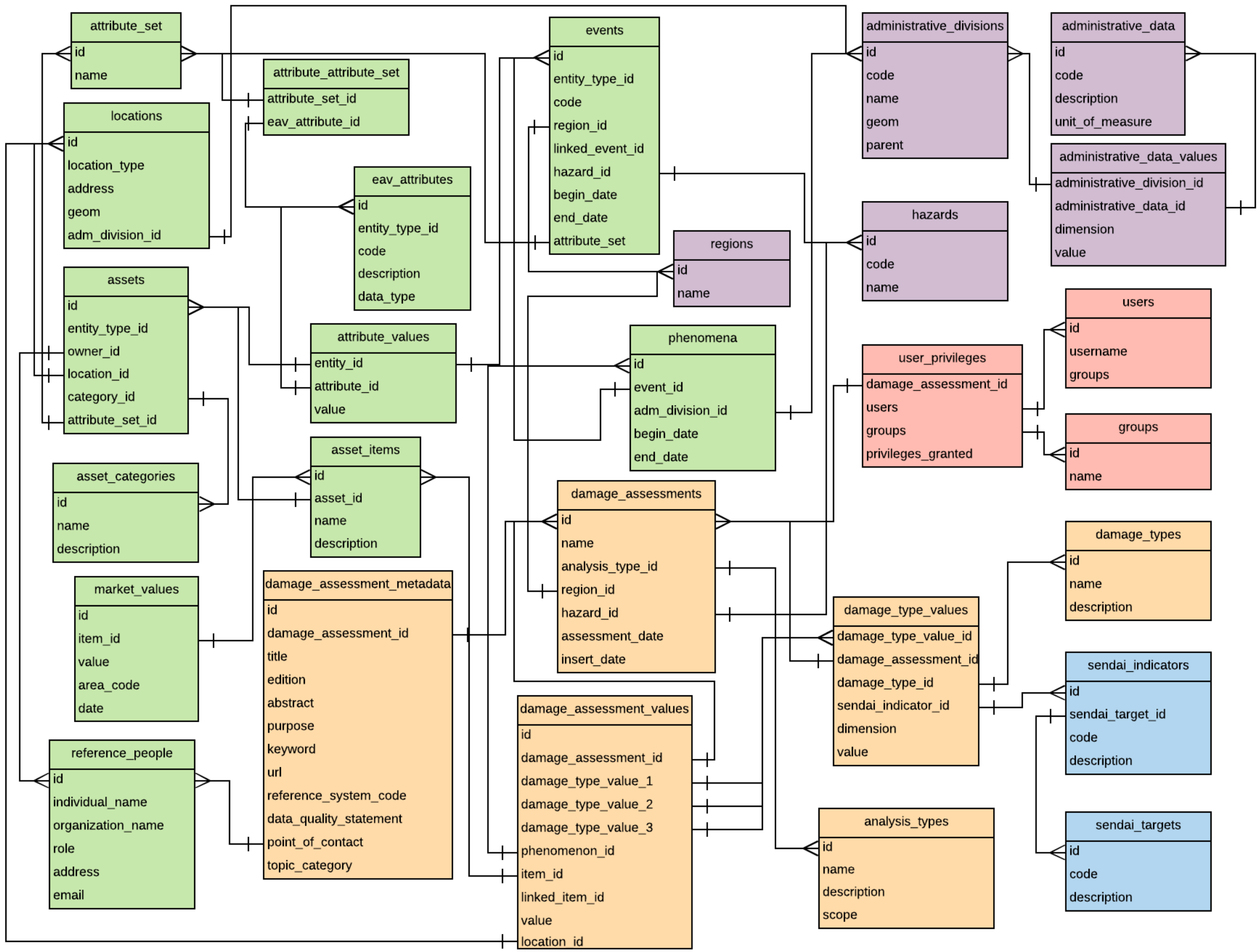


Figure 5. The RDH database diagram.

Please note that in the diagram above, tables are highlighted in different colours corresponding to specific functionalities within the application.

What has changed?

4.1.1. Introduction of entity–attribute–value data model

An entity–attribute–value model (EAV) is a data model to encode, in a space-efficient manner, entities where the number of attributes (properties, parameters) that can be used to describe them is potentially vast, but the number that will actually apply to a given entity is relatively modest. Such entities correspond to the mathematical notion of a sparse matrix. This particular model is well suited for events and assets, as the single entities have many different characterisations, depending on their type.

4.1.2. Events

The event entity has been split into a 'macro event' and a phenomenon, as explained in Chapter 1.3. An event table linked to a number of external tables (hazards) no longer exists: all event attributes are stored in a centralised table, implementing the EAV data model. Since attributes may differ from hazard to hazard, each event instance is bound to a specific attribute set that is ideally equivalent to a hazard.

4.1.3. Assets

Similarly to events, the asset entity has also been split into a 'macro asset' and an item: each asset may contain one or multiple assets (e.g. a house containing pieces of furniture). Damages are linked to items, not to macro assets. Asset attributes are not described by an additional table for every type, but they use the EAV data model and they are also divided in categories. For maximum abstraction, people are considered as a specific asset category.

4.1.4. Locations

A location entity still exists, but defines also a type (e.g. fixed location, non-fixed location, people) and it is linked to damages as well; this way every single occurrence of damage may have a specific location, as the damage location may differ from the asset location. The damage location could be a point, or a polygon that defines an extent.

As mentioned before, the RDH database implements an abstraction of the loss database, which can be identified by the inventory section (green) of the schema. There are further sections that allow all the functionalities to exist.

Below is a description of entities implemented, ordered by section (according to the colours).

4.2. Inventory section

This section describes entities that allow the storage of all inventory data needed, regarding both assets and events. The number of columns for assets and events is limited, because all possible descriptive fields are managed via the EAV data model, which allows new attributes to be defined at any time, without the need to change the database structure.

Figure 6. Inventory section of the RDH database.

locations

Description: this entity is useful for storing the location of any type of asset (fixed, non-fixed, people), or the extent of a single damage.

Fields:

- Id (int): unique identifier

- Location_type (enum): e.g. fixed asset
- Address (varchar).
- Geom (binary): geometry (could be point or polygon)
- Administrative_division_id (int): reference to administrative divisions.

assets

Description: generic entity affected by event (also includes people).

Fields:

- Id (int): unique identifier
- Entity_type (enum): defines entity type for mapping fitting attributes
- Owner_id (int): reference to reference_people
- Asset_location_id (int): reference to locations
- Asset_category_id (int): reference to categories
- Attribute_set_id: reference to attribute_set.

asset_items

Description: single item included in the asset (equal to asset in the simplest case).

Fields:

- Id (int): unique identifier
- Asset_id (int): reference to assets
- Name (varchar).

asset_categories

Description: categories for assets; e.g. buildings, infrastructure or people.

Fields:

- Id (int): unique identifier
- Name (varchar).
- Description (varchar).

market_values

Description: market value of items.

Fields:

- Id (int): unique identifier
- Item_id (int): reference to assets
- Value (decimal).
- Area_code (varchar).
- Date (datetime): start validity date.

reference_people

Description: could be the owner of an asset, author of publications, etc.

Fields:

- Id (int): unique identifier
- Individual_name (varchar).
- Organization_name (varchar).
- Role (varchar).
- Address (varchar).
- City (varchar).
- Zipcode (varchar).
- Country (varchar).
- Email (varchar).

eav_attributes

Description: attributes relevant to events and assets (and more) are defined in a single place. This feature allows new attributes to be defined at any time, without the need to change the structure of the database.

Fields:

- Id (int): unique identifier
- Entity_type_id (int): defines entity type for mapping fitting attributes
- Data_type (varchar): defines data type (varchar, text, integer, decimal, datetime)
- Name (varchar).
- Description (varchar).

attribute_values

Description: attribute values are stored in dedicated tables for each type of data (varchar, text, integer, decimal, datetime).

Fields:

- Entity_id (int): identifier of entity (event or asset)
- Attribute_id (int): identifier of eav_attribute
- Value: see note below.

The database diagram provided with this document includes a simplified view of the implemented EAV data model. Actually, a table for each data_type/entity_type exists in the database, for example event_attribute_values_varchar, event_attribute_values_text.

attribute_set

Description: attribute sets are used to link attributes to specific instances of an entity.

Fields:

- Id (int): unique identifier
- Name (varchar).

attribute_attribute_set

Description: this is a relation between attribute_set and eav_attribute, so it is basically the content of an attribute set.

Fields:

- Attribute_set_id: reference to attribute_set
- Eav_attribute_id: reference to eav_attribute.

events

Description: an event is a generic entity which may be the cause of damage.

Fields:

- Id (int): unique identifier of the event
- Entity_type_id (int): defines entity type for mapping fitting attributes
- Region_id (int): could be Europe, or any country corner
- Linked_event_id (int): optional link to an event identified as cause of the current one (chained events)
- Hazard_id (int): identifier of the hazard (e.g. flood)
- Begin_date (datetime): starting date of recognised event
- End_date (datetime): starting date of recognised event
- Attribute_set_id: reference to attribute_set.

phenomena

Description: a phenomenon is part of a major event and has a specific location and related assessed damage.

Fields:

- Id (int): unique identifier
- Event_id (int): related event
- Administrative_division (int): maps location of phenomenon
- Begin_date (datetime): starting date of recognised event
- End_date (datetime): starting date of recognised event.

4.3. Administrative data section

This section gathers entities used for basic characterisation of data stored for the damage assessments.

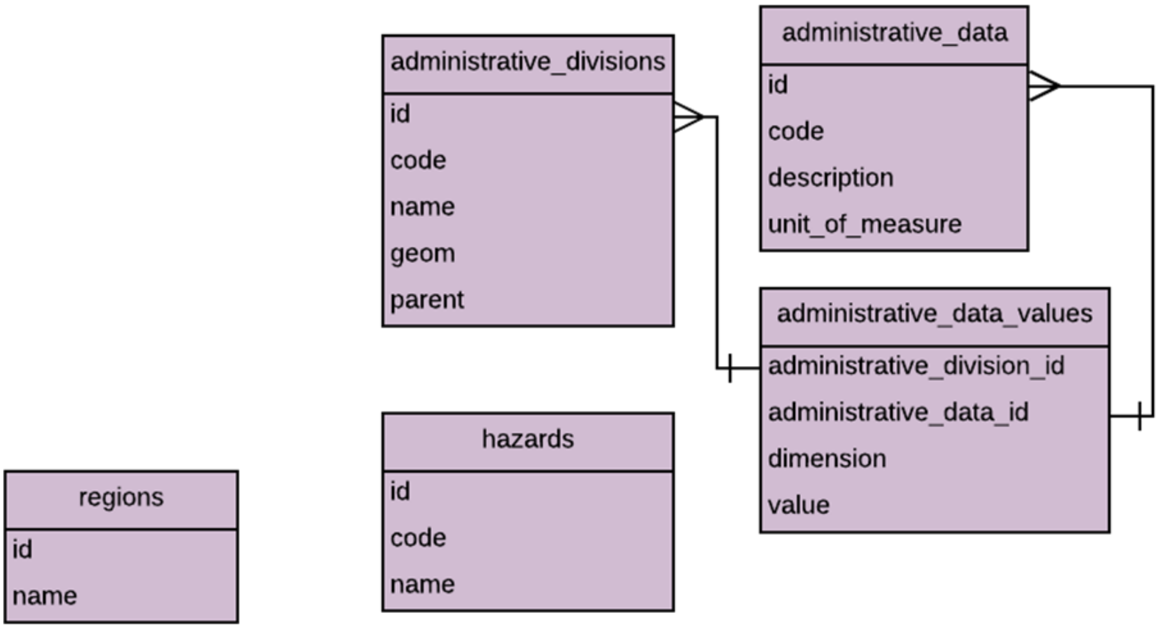


Figure 7. Administrative data section of the RDH database.

hazards

Description: definition of hazard (e.g. river flood).

Fields:

- Id (int): unique identifier
- Code (varchar): e.g. FL for flood
- Description (varchar).

administrative_divisions

Description: this entity stores basic data regarding administrative divisions.

Fields:

- Id (int): unique identifier
- Code (varchar): ISO 3166-1 alpha-2 code for countries, or relevant NUTS code according to Eurostat
- Name (varchar): name of administrative division
- Geom (binary): spatial data
- Parent_id (int): parent adm division.

regions

Description: this is crucial for ownership management of data and visibility. Each user in the system belongs to a specific region and so do the data owned by that user.

Fields:

- Id (int): unique identifier
- Name (varchar): name of region (e.g. Europe, or country corner, like Austria).

administrative_data

Description: definition of data related to administrative divisions, like gross domestic product (GDP), population, area, and so on.

Fields:

- Id (int): unique identifier
- Code (varchar): e.g. GDP
- Description (varchar): description of data
- Unit_of_measure (varchar): e.g. EUR million.

administrative_data_value

Description: relation between administrative data and administrative divisions.

Fields:

- Administrative_division_id (int).
- Administrative_data_id (int).
- Dimension (varchar): e.g. Year 2018 of GDP
- Value (decimal).

4.4. Damage assessment section

This section represents the core of the RDH, as it defines the damage assessments and how the datasets are organised. The analysis_types entity basically defines a dataset in terms of data analysed (buildings, people) and of scope (risk analysis or historical events). The damage_types defines the dimensions used to measure data within the assessment (e.g. climate change scenarios, return periods of events).

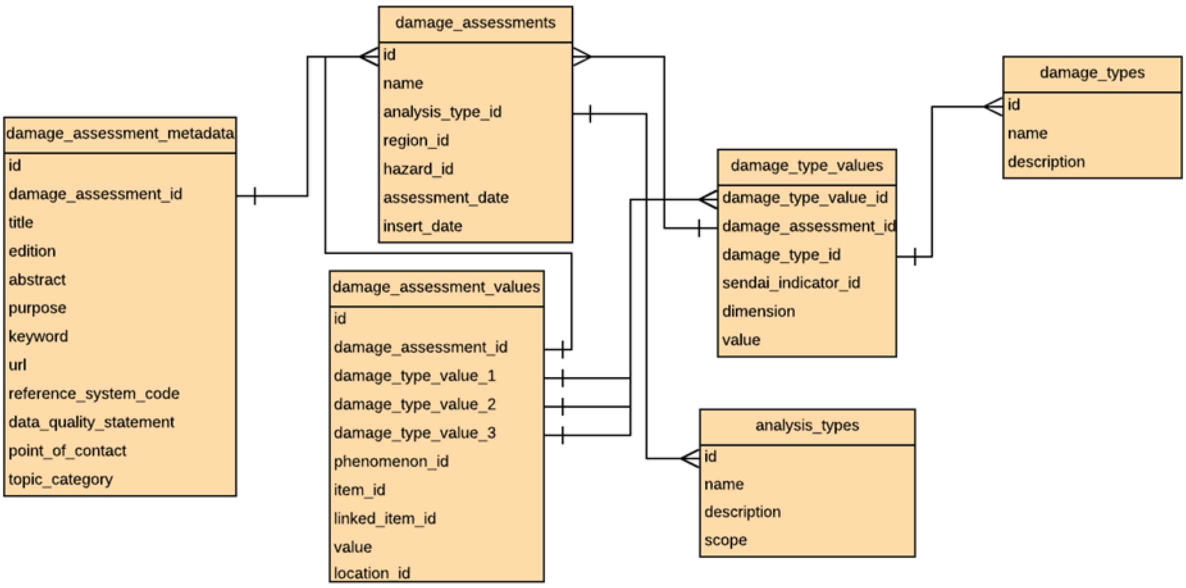


Figure 8. Damage assessment section of the RDH database.

analysis_types

Description: defines the type of data analysed (e.g. population, buildings, economic values).

Fields:

- Id (int): unique identifier
- Name (varchar).
- Description (varchar).

damage_assessments

Description: definition of data measured.

Fields:

- Id (int): unique identifier
- Name (varchar): name given (unique)
- Analysis_type_id (int): reference to analysis type
- Region_id (int): reference to region, needed for risk analyses that do not use events
- Hazard_id (int): reference to hazard, needed for risk analyses that do not use events
- Assessment_date (datetime): date declared for the assessment
- Insert_date (datetime): date of insertion in the database.

damage_types

Description: definition of considered scenario. It is useful for complex analysis with predicted values in different declinations of a given scenario (e.g. climate change).

Fields:

- Id (int): unique identifier
- Name (varchar): name given (unique)
- Description (varchar).

damage_type_values

Description: relation between damage_assessments and damage_types.

Fields:

- Id (int): unique identifier
- Damage_assessment_id (int): reference to damage assessment
- Damage_type_id (int): reference to damage type
- Sendai_indicator_id (int): reference to Sendai indicator
- Dimension (varchar): e.g. axis of a chart
- Value (varchar): value of damage type for given assessment and dimension.

damage_assessment_value

Description: value assigned to the loss for the given phenomenon, damage assessment, damage type and item.

Fields:

- Id (int): unique identifier
- Damage_assessment_id (int): reference to damage assessment
- Damage_type_value_1(2,3)_id (int): damage type specific to damage assessment
- Phenomenon_id (int): reference to phenomena
- Item_id (int): reference to asset_items
- Linked_item_id (int): e.g. allows to map people into a building
- Value (decimal).
- Location_id (int): reference to locations, to store location (extent) of the single damage.

damage_assessment_metadata

Description: complementary description of a damage assessment publication.

Fields:

- Id (int): unique identifier
- Damage_assessment_id (int): reference to damage assessment
- Title (varchar).
- Edition (varchar).
- Abstract (varchar).
- Purpose (varchar).
- Keyword (varchar).
- Url (varchar).
- Reference_system_code (varchar).
- Data_quality_statement (text).
- Point_of_contact (int): point of contact for the publication (reference_people)
- Author (int): author of publication (reference_people)
- Topic_category: e.g. environmental, structure.

4.5. Authorisation section

These entities ensure the datasets are properly managed by their owners, which may allow other users to perform operations (view, create, edit or delete).

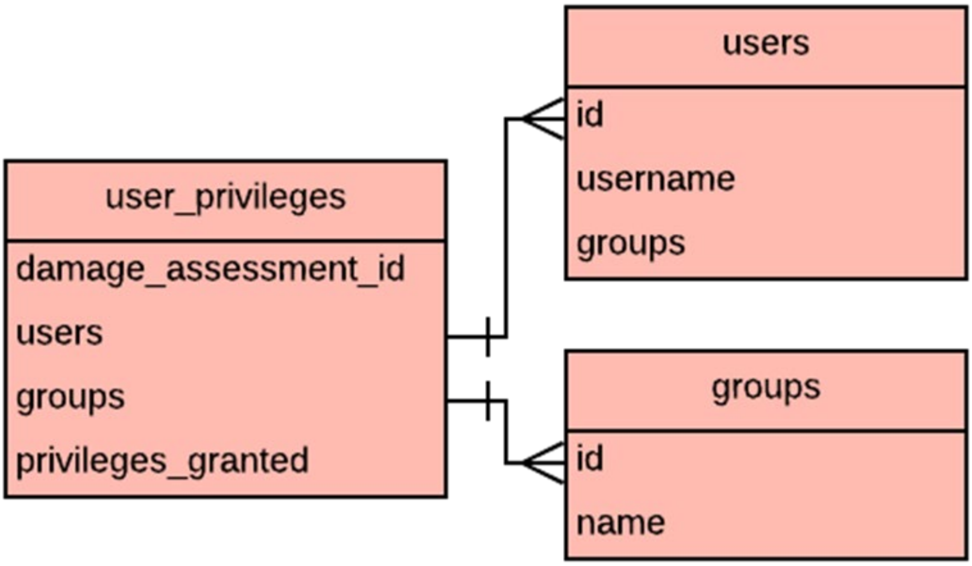


Figure 9. Authorisation section of the RDH database.

users

Description: users registered.

Fields:

- Id (int): unique identifier
- Username (varchar).
- Groups (array): list of groups the user belongs to.

groups

Description: group of users for permission purposes.

Fields:

- Id (int): unique identifier
- Name (varchar): name given (unique).

user_privileges

Description: privileges assigned to a group or single user to perform actions against a damage assessment (view, create, edit, delete).

Fields:

- Damage_assessment_id (int): reference to damage_assessment
- Users (array): list of users for current entry
- Groups (array): list of groups for current entry
- Privileges_granted (array): list of privileges granted for current entry.

4.6. Additional section

This additional section collects entities that are not strictly relevant to the main functionalities of the application. At this time, there are the definitions of Sendai targets and indicators. Please note that these tables are used only to store a mapping between the assessments performed by the RDH and the Sendai indicators, while outputs useful for Sendai reporting are generated, when data available is consistent, using a logic implemented in the source code of the application.

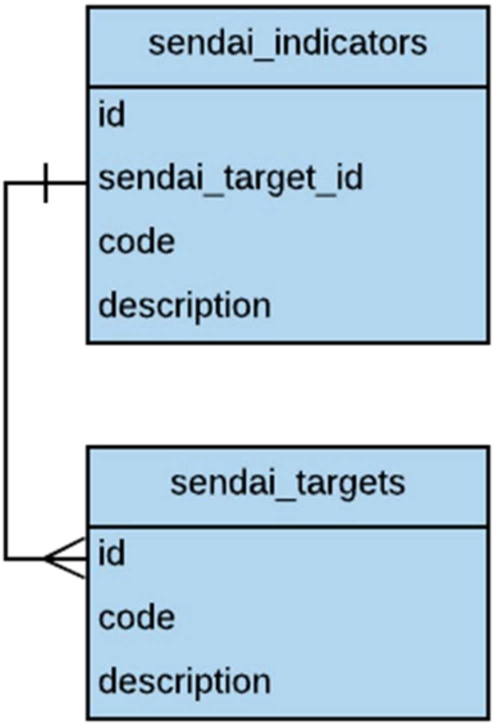


Figure 10. Additional section of the RDH database.

sendai_targets

Description: Sendai target as defined by specifications from the UN Office for Disaster Risk Reduction.

Fields:

- Id (int): unique identifier
- Code (varchar): (unique)
- Description (varchar).

sendai_indicators

Description: Sendai Indicator as defined by specifications from the UN Office for Disaster Risk Reduction.

Fields:

- Id (int): unique identifier
- Sendai_target_id (int): reference to target
- Code (varchar): (unique)
- Description (varchar).

5. Conclusions

The way the RDH is used at local level is completely up to the user, who basically has two options: activate their account on the EU-hosted platform, or deploy the whole application on a separate infrastructure of their own choice. While the first option is definitely faster to implement and does not include any costs for the user, the second one could be preferable if specific needs or constraints exist, for example restricted access to the internet, managing of extremely large datasets, customisation of base models like hazards or analysis types, or even concerns about the privacy of some sensitive data.

The application is still in the development phase and different collaborations have been established with both scientists and end users from several areas. Future work will then focus on new topics, like technological disasters, critical infrastructure and cultural heritage; this is why the identification of new features to be integrated is expected. The database architecture is by design hopefully flexible enough to handle all the complexity introduced, but as the application continues to grow, we cannot exclude that upgrades will also be needed for this part, which is typically the most critical to change.

References

Ríos Días, F. Marín Ferrer, M., *Loss database architecture for disaster risk management,* 2018.(https://ec.europa.eu/jrc/en/publication/eur-scientific-and-technical-research-reports/loss-database-architecture-disaster-risk-management).

Ríos Díaz, F.et. al., *Update of the DRMKC loss database architecture for disaster risk management,* 2018.

Antofie, T., Luoni, S., Faiella, A., Marín Ferrer, M., *Risk Data Hub — web platform to facilitate management of disaster risks,* 2019.

Wikipedia, *Entity–attribute–value model.* (https://en.wikipedia.org/wiki/Entity-attribute-value_model)

Wikipedia, *Extract, transform, load*. (https://en.wikipedia.org/wiki/Extract,_transform,_load)

GeoSAFE (https://geonode.ingc.gov.mz/)

RASOR (http://www.rasor.eu/rasor/)

ThinkHazard! (http://thinkhazard.org/en/)

Afghanistan Disaster Risk (http://disasterrisk.af)

The World Bank (https://www.worldbank.org/)

GFDRR (https://www.gfdrr.org/)

CIMA Research Foundation (http://www.cimafoundation.org/)

EFAS (https://www.efas.eu/)

EFFIS (http://effis.jrc.ec.europa.eu/)

EDO (http://edo.jrc.ec.europa.eu/)

GHSL (https://ghsl.jrc.ec.europa.eu/)

Copernicus (https://emergency.copernicus.eu/)

GDACS (http://www.gdacs.org/)

EMM (http://emm.newsbrief.eu/NewsBrief/clusteredition/en/latest.html)

EM-DAT (https://www.emdat.be/)

HANZE (https://data.4tu.nl/repository/uuid:5b75be6a-4dd4-472e-9424-f7ac4f7367f6)

Python programming language (https://www.python.org/)

Django (https://www.djangoproject.com/)

PostgreSQL (https://www.postgresql.org/)

GeoNode (http://geonode.org/)

GeoServer (http://geoserver.org/)

OGC services — Open GeoSpatial Consortium (https://www.opengeospatial.org/standards/owc)

ReactJS (https://reactjs.org/)

MapStore (https://mapstore.geo-solutions.it/)

Leaflet maps (https://leafletjs.com/)

List of abbreviations and definitions

DRMKC	Disaster Risk Management Knowledge Centre
EAV	Entity–attribute–value
EDO	European Drought Observatory
EFAS	European Flood Awareness System
EFFIS	European Forest Fire Information System
ETL	Extract, transform, load
EM-DAT	Emergency Events Database
EMM	Europe Media Monitor
GDACS	Global Disaster Alert and Coordination System
GFDRR	Global Facility for Disaster Reduction and Recovery
GHSL	Global Human Settlement Layer
HANZE	Historical Analysis of Natural Hazards in Europe
RDH	Risk Data Hub

List of figures

GETTING IN TOUCH WITH THE EU

In person

All over the European Union there are hundreds of Europe Direct information centres. You can find the address of the centre nearest you at: https://europa.eu/european-union/contact_en

On the phone or by email

Europe Direct is a service that answers your questions about the European Union. You can contact this service:

- by freephone: 00 800 6 7 8 9 10 11 (certain operators may charge for these calls),

- at the following standard number: +32 22999696, or

- by email via: https://europa.eu/european-union/contact_en

FINDING INFORMATION ABOUT THE EU

Online

Information about the European Union in all the official languages of the EU is available on the Europa website at: https://europa.eu/european-union/index_en

EU publications

You can download or order free and priced EU publications from: http://publications.europa.eu/en/publications. Multiple copies of free publications may be obtained by contacting Europe Direct or your local information centre (see https://europa.eu/european-union/contact_en).

KJ-NA-29756-EN-N

JRC Mission

As the science and knowledge service of the European Commission, the Joint Research Centre's mission is to support EU policies with independent evidence throughout the whole policy cycle.

EU Science Hub
ec.europa.eu/jrc

@EU_ScienceHub

EU Science Hub - Joint Research Centre

Joint Research Centre

EU Science Hub

Publications Office

doi:10.2760/67169

ISBN 978-92-76-03978-5