Selection #1264629 - The Risk Data Library: Integration of the MOVER vulnerability schema and database

D6 Final Report

Authors: P. Henshaw, E. Verrucci, T. Rossetto

Date: 07 February 2020

Abstract

This document summarizes the activities undertaken as part of this project, namely the production of a simplified and updated version of the MOVER vulnerability database schema, and the revision of existing database schemas and supporting software tools for hazard, exposure and loss databases. An important part of the schema revision activities was the incorporation of common database elements developed in previous Challenge Fund projects and adopted by the hazard, exposure (GED4ALL) and loss databases, and the evolution of these elements to include occupancy categories and a more structured and robust model of hazard intensity measure types (IMTs).

We first describe the changes applied to the MOVER vulnerability database schema outlining key changes compared to the previous version, and the newly developed import / export software tools. We then describe the changes to the other Challenge Fund database schemas, and their supporting software tools. This is followed by considerations for database deployment and some discussion of scoring in the revised database.

We conclude with a brief discussion of limitations and possible future directions including opportunities for continued collaboration with other initiatives such as the UK Space Agency METEOR project, and the work of the IDF/RMSG.

Keywords

vulnerability, hazard, exposure, losses, databases, intensity measure types, building taxonomy, scenarios, hazards footprints, disaster risk reduction, open-source, open-data.

Intended Audience and Terminology

Large portions of this document are targeted at readers with database programming and/or administration skills and assumes knowledge of general database constructs, the SQL language (particularly the PostgreSQL implementation), and geospatial operations (specifically PostGIS). We also assume familiarity with the Challenge Fund Round 3 Hazard, Exposure, Loss and Vulnerability databases and the concepts associated with natural hazards and risk assessment.

We use the PostgreSQL `SCHEMA` construct to implement namespaces. In order to avoid confusion, in this document we will use the term 'schema' to indicate the complete set of related database elements used to implement a database and the term 'namespace' to refer to the specific PostgreSQL implementation of a logical grouping of tables and related elements. For example, the GED4ALL exposure database schema uses two namespaces: *cf_common* for common database elements and *ged4all* for those specific to GED4ALL.

Introduction

In 2017, the Global Facility for Disaster Risk reduction and Recovery (GFDRR) and the UK Department for International Development (DFID) have financed, through the activities of the Challenge Fund Round 2, the development of three data schemas for the development of a robust, open, accessible and expandable database designed for use in both developed and developing countries to store, visualise, and download Exposure, Hazard, and Vulnerability data. These data schemas were presented at the Challenge Fund Workshop held in Dar es Salaam in March 2018. Here it was found that the Hazard, and the Exposure data schemas shared the same terminology, but the Vulnerability schema - also known as the Multi-Hazard Open Vulnerability Platform for Evaluating Risk (MOVER) - presented some differences in the use of intensity measure types and asset data attributes. The following phase of integration of the data schemas, and the development of a new Loss data schema, also brought to light the difficulty to display vulnerability, fragility, and data-to-loss models in spatial terms. To overcome these difficulties and thus further the use and development of the schema, the GFDRR/DFID Challenge Fund seeks to support work to harmonize the MOVER schema with the other components, ensuring complete linkage by using consistent intensity parameters and asset data attributes, and enabling users to efficiently search across all four risk data components by ensuring the vulnerability schema includes aligned project and dataset information. The planned web portal – the Risk Data Library (RDL) –which is the user' link access point to the schema and stored data, cannot fully meet users' needs without this step.

The objective of this exercise is to integrate the existing MOVER vulnerability data schema into the same database architecture and format as the hazard, exposure, and loss schema. The integrated schema will share common tables and enumerated types enabling linked data (e.g. data types from the same project, or applicable for the same hazard or asset type) to be linked across the schema and discovered together by user search in the RDL and will, ultimately, fulfil the technical requirements for the applicability of the use cases identified during the Inception meeting call on 23rd October 2019.

This document presents the results of the project and describes the strategies adopted by this consortium to achieve these goals.

Deliverables

Since many of the deliverables specified by the ToR and referenced in the Workplan are changes to database schema and Python code rather than written reports, in this section we provide links to the public github source code repositories where the relevant code artefacts can be found.

Deliverable	Type	Link / Notes
D1	Inception Report	Delivered 11 Nov 2019
D2	SQL Schema (final_schema.sql, final_data.sql)	https://github.com/enricaverrucci/movercf/blob/master/schema.sql
D3	Python scripts	https://github.com/enricaverrucci/movercf/tree/master/python
D4	Python scripts and changes to SQL code.	Updated versions of the Python scripts and SQL code are available via github. Please see the dedicated section below for further details
D5	SQL VIEW / Stored Procedure or Python (integrated in final_schema.sql)	https://github.com/enricaverrucci/movercf/blob/master/schema.sql
D6	Final report	This document

Revised MOVER Database Schema (D1)

This section describes the conceptual changes required for the refactoring of the MOVER database schema. The revised MOVER focuses solely on the representation of physical vulnerability and on its direct or indirect assessment by means of the fragility functions, vulnerability functions and damage to loss (DtL) ratios. Hence, it does not include tables or data pertaining to the representation of social or physical vulnerability by means of social, physical or hybrid vulnerability indicators and/or indices, as in the previous version of MOVER. However, since the modular design of the schema works in favour of future expansions, a revised social vulnerability module - as well as new modules - may be added at a later stage. Figure 1 offers an overview of the revised MOVER vulnerability database schema.

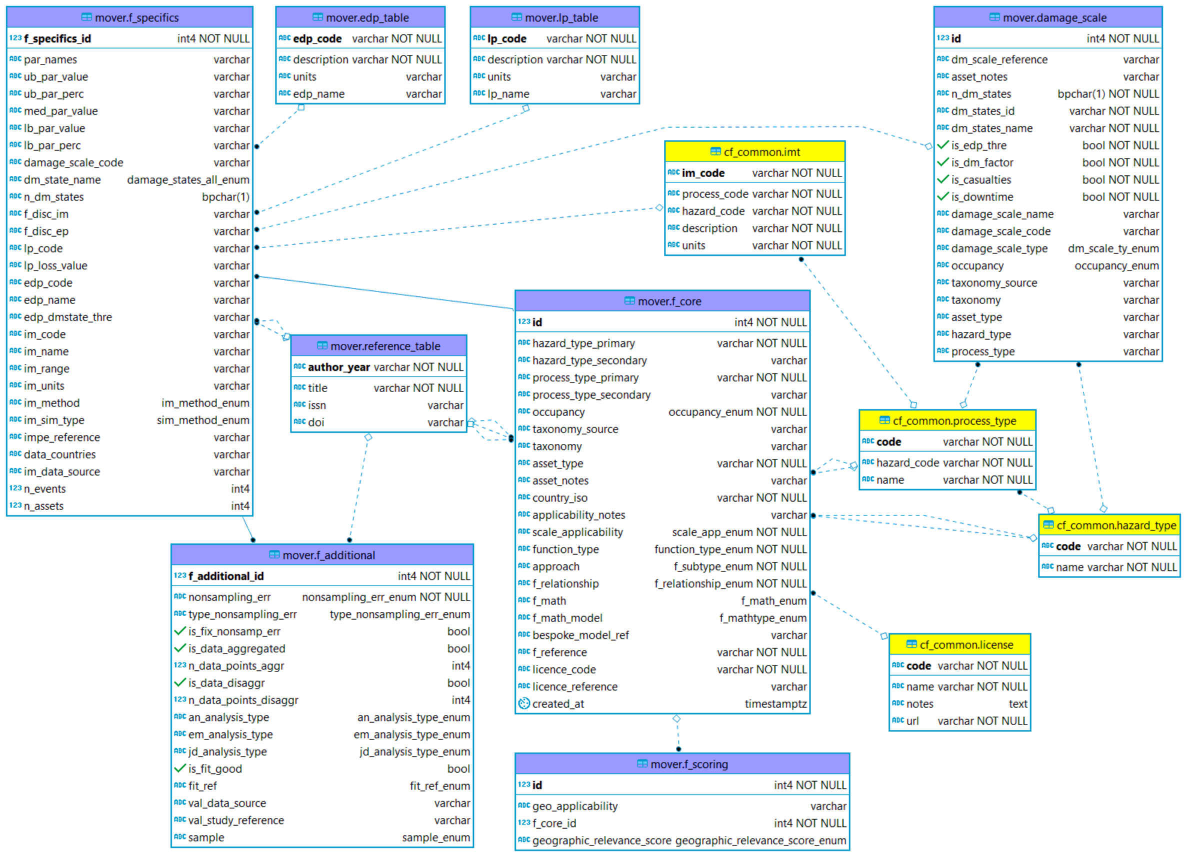

mover.f_specifics
f_specifics_id int4 NOT NULL
par_names varchar
ub_par_value varchar
ub_par_perc varchar
med_par_value varchar
lb_par_value varchar
lb_par_perc varchar
damage_scale_code varchar
dm_state_name damage_states_all_enum
n_dm_states bpchar(1)
f_disc_im varchar
f_disc_ep varchar
lp_code varchar
lp_loss_value varchar
edp_code varchar
edp_name varchar
edp_dmstate_thre varchar
im_code varchar
im_name varchar
im_range varchar
im_units varchar
im_method im_method_enum
im_sim_type sim_method_enum
impe_reference varchar
data_countries varchar
im_data_source varchar
n_events int4
n_assets int4
mover.edp_table
edp_code varchar NOT NULL
description varchar NOT NULL
units varchar
edp_name varchar
mover.lp_table
lp_code varchar NOT NULL
description varchar NOT NULL
units varchar
lp_name varchar
cf_common.imt
im_code varchar NOT NULL
process_code varchar NOT NULL
hazard_code varchar NOT NULL
description varchar NOT NULL
units varchar NOT NULL
mover.damage_scale
id int4 NOT NULL
dm_scale_reference varchar
asset_notes varchar
n_dm_states bpchar(1) NOT NULL
dm_states_id varchar NOT NULL
dm_states_name varchar NOT NULL
is_edp_thre bool NOT NULL
is_dm_factor bool NOT NULL
is_casualties bool NOT NULL
is_downtime bool NOT NULL
damage_scale_name varchar
damage_scale_code varchar
damage_scale_type dm_scale_ty_enum
occupancy occupancy_enum
taxonomy_source varchar
taxonomy varchar
asset_type varchar
hazard_type varchar
process_type varchar
mover.reference_table
author_year varchar NOT NULL
title varchar NOT NULL
issn varchar
doi varchar
mover.f_core
id int4 NOT NULL
hazard_type_primary varchar NOT NULL
hazard_type_secondary varchar
process_type_primary varchar NOT NULL
process_type_secondary varchar
occupancy occupancy_enum NOT NULL
taxonomy_source varchar
taxonomy varchar
asset_type varchar NOT NULL
asset_notes varchar
country_iso varchar NOT NULL
applicability_notes varchar
scale_applicability scale_app_enum NOT NULL
function_type function_type_enum NOT NULL
approach f_subtype_enum NOT NULL
f_relationship f_relationship_enum NOT NULL
f_math f_math_enum
f_math_model f_mathtype_enum
bespoke_model_ref varchar
f_reference varchar NOT NULL
licence_code varchar NOT NULL
licence_reference varchar
created_at timestamptz
cf_common.process_type
code varchar NOT NULL
hazard_code varchar NOT NULL
name varchar NOT NULL
cf_common.hazard_type
code varchar NOT NULL
name varchar NOT NULL
cf_common.license
code varchar NOT NULL
name varchar NOT NULL
notes text
url varchar NOT NULL
mover.f_additional
f_additional_id int4 NOT NULL
nonsampling_err nonsampling_err_enum NOT NULL
type_nonsampling_err type_nonsampling_err_enum
is_fix_nonsamp_err bool
is_data_aggregated bool
n_data_points_aggr int4
is_data_disaggr bool
n_data_points_disaggr int4
an_analysis_type an_analysis_type_enum
em_analysis_type em_analysis_type_enum
jd_analysis_type jd_analysis_type_enum
is_fit_good bool
fit_ref fit_ref_enum
val_data_source varchar
val_study_reference varchar
sample sample_enum
mover.f_scoring
id int4 NOT NULL
geo_applicability varchar
f_core_id int4 NOT NULL
geographic_relevance_score geographic_relevance_score_enum

Figure 1: MOVER vulnerability database schema

In the refactored MOVER schema, the **f_core**, **f_specifics,** and the **f_additional** tables store all attributes included in the **ff_table**, **vf_table** and **dlt_table** of the previous version of MOVER. As such, they constitute the core module of the refactored schema. These new tables refer to supporting tables describing damage scales (DMs), intensity measures types(IMTs), loss parameters (LPs), and engineering demand parameters (EDPs). They also link to a **scoring_table**, which has been simplified with respect to previous versions of MOVER, and a **reference_ table** - which remains unchanged. The damage scale, LP, and EDP table remain part of the mover schema. The IMTs table has been moved in the common namespace schema (**cf_common**) and the intensity measures types have been harmonised, to be used seamlessly across both the hazard and the vulnerability schemas. **Cf_common** stores all common tables across the hazard, exposure, vulnerability, and loss schemas.

The **hazard_table** and **asset_table** were used in the previous version of MOVER as placeholders until a common namespace across the hazard, exposure and vulnerability data schemas had been established. As for the IMTs table, these tables have now been replaced by new tables in **cf_common**. At the time of writing, the **cf_common** schema includes the **hazard_type** table, a **process_type** table, the **licence** table, and the **imt_table** but more tables can be added to it to better support the integration across the current schemas and potential new modules. As noted in previous discussions we would feel that strict constraints preventing the contribution of innovative approaches to vulnerability would be counterproductive, so the list of IMTs is intended to be used by a future user-interface to propose suggestions but not to restrict input.

A summary of the main structural changes between the old MOVER data schema and the current version are summarised in Table 1. These changes are further described in the following text. It is highlighted that a **timestamp** has been added to the tables of the core module to track contributions.

Table 1. Summary of changes across the two MOVER versions (Key: X *- Deleted,* → *- Kept with revisions).*

Module	**Table name in MOVER_OLD**	**Action**	**Table name in Refactored MOVER / Notes**
Asset	asset_table	X	Replaced by occupancy field in f_core
Hazard	hazard_table	X	Replaced by cf_common.hazard_type cf_common.process_type
Social and Physical Vulnerability - Indicators and Indices	Soc_v_cat Soc_v_char Socvul_indicators Soc_ind_scoring Phy_v_cat Phy_v_char Phyvul_indicators Phy_ind_scoring Indx_table indx_scoring	X	Outside the scope of this project. This module could be revised to be integrated to the refactored MOVER in the future. The original Social Vulnerability module remains active in the original MOVER database.
CORE module - Fragility, Vulnerability, DtL functions	ff_table vf_table dtl_table	→	f_core f_specifics f_additional
Scoring	ff_scoring_table vf_scoring_table	→	f_scoring
	data_table	X	LInked to the old scoring system. It has not been replaced
Supporting tables in MOVER	damage_scale edp_table lp_table	→	damage_scale edp_table lp_table
Supporting table in cf_common	im_table	X	Replaced by cf_common.imt
Reference table	reference_table	→	reference_table (unchanged)

Changes to the Asset table

The **asset_table** of MOVER has been dropped in the refactoring as information about the asset is now managed as fields of the **f_core** table. New and modified fields include the new **occupancy_type** as well as the taxonomy source, to help integration with the exposure data schema.

Changes to the Hazard table

The **hazard_table** presented in the previous version of MOVER has been replaced by the **hazard_type** and the **process_type** tables in the **cf_common** schema. The **hazard_type.code** and the **process_type.code** unequivocally identify the type of hazard/s and process/es for which a function has been developed.

When developing a function, it is possible that data is collected after an event characterized by more than one hazard or process e.g., damage caused by a tsunami happening after an earthquake would for instance refer to two different hazard_types according to the current nomenclature described in the **cf_common.hazard_type** table, and in previous versions of MOVER this could not be captured. Fragility and vulnerability curves that look at sequential hazards are becoming more common. Furthermore, empirical fragility and vulnerability are known to commonly inherently contain damage or losses from a combination of hazards, e.g. earthquake ground shaking and induced liquefaction. In order to capture this, the refactored version of MOVER is now equipped with the capacity of recording up to two hazards and process types. The **f_core** table and the **damage_scale** table present both **hazard_type_primary** and **hazard_type_secondary** fields as well as **process_type_primary** and **process_type_secondary** fields. These refer to the hazard_type_code and the process_type _code, respectively. In addition, a new multi_hazards field has been added. This allows the clearer identification of fragility and vulnerability functions and DtL that are derived for more than one hazard. In empirical fragility, vulnerability and DtL models, the observational data is often collected following a number of linked events (e.g. earthquake and tsunami, or earthquake and strong aftershock). This new field allows this to be specified. It also captures new trends in the development of analytical fragility and vulnerability functions for sequential hazards.

Changes to the core module

The design of the core module has been substantially simplified. In the previous version of MOVER, the Fragility, Vulnerability and Damage-to-Loss functions were stored into three separate tables, one for each function type, and common fields were repeated across the three tables. This design solution originated from the need to represent all the different functions subtypes appropriately and to guide the user in the correct population of the schema by setting constraints. By keeping the tables separate, the constraints applied to each field were more specific to the function subtype and, in turn, the number of fields that could be left 'Null' by the user was also constrained. But having three separate tables inevitably resulted in some duplication. Once the MOVER schema was populated, it became easier to investigate which specific fields were often left unpopulated by the user due to lack of data and, thus, to make more case-fitting considerations about the trade-off between having some more nullable fields versus having common fields repeated across different tables. It was observed that some fields were more difficult to populate than others, especially in developing countries where there is a greater paucity of functions and information about the function attributes is often sparse. All the fields for which data were difficult to find had to be left as 'Nullable' in the schema design, and this brought to the realisation that the attributes of the functions could be more efficiently summarised in three simplified tables with no repeated fields.

In the current version, the **f_core** table now stores the descriptive attributes of the functions (e.g., **function_type**, **country_ISO**), which can be intended as the metadata of the functions. It is expected that the great majority of the fields of this table will be populated at all times. The **f_specifics** stores the attribute of functions that are linked to the mathematical form of the function. As the functions differentiate across subtypes, it is expected that the fields of the table with be populated in part but in a mutually exclusive fashion (i.e., if the user populates the attributes that describe a parametric function then they will leave empty the attributes of the bespoke and discrete functions).

The **f_additional** table stores attributes linked to the method used for building a function, and to the existence of a validation study carried out with independent data (or lack thereof). The decision of separating these fields from the **f_specific table** derives from the observation that, while the fields of the **f_specific** table will always be found in the scientific paper describing the function, the fields describing the method and the validation may not be present in paper at all, or may be subjective to the judgement of the user (i.e., is the function fit a good fit?). It is therefore expected that these fields will be populated less often and, for the sake of a clearer visualisation of the **f_specifics** table, it was appropriate to have a separate table for these attributes. Documentation is provided in the appendix (see section on Function types) to guide the user in the population of the table of the functions core module.

In order to simplify the **ff_table**, **vf_table**, and **dtl_table** into two main tables (**f_core** and **f_specifics**) and one additional table (**f_additional**), whilst also harmonising the nomenclatures across the four schemas, some alterations of the data types describing the functions have been necessary and new fields have been created ad-hoc for these data. These are described in Table 2. Also, Figure 2 (a,b) show a comparison of the Entity - Relationship (ER) diagrams of the tables in the core module before and after the refactoring .

Table 2. New fields and data type required for the MOVER refactoring. Fields and data type marked as (*) are new, the ones marked as (**) existed in the previous version of MOVER but have been either modified and/or expanded.

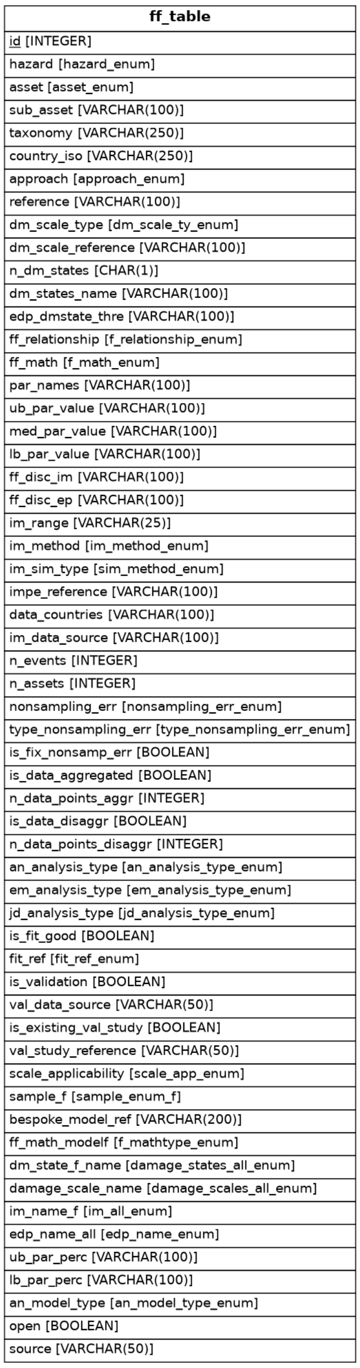
Fields	**Table**	**Data type**	**Description**
hazard_type_primary (*) hazard_type_secondary (*)	f_core	VARCHAR	CS EQ TS VO CF FL LS WI ET DR WF MH
process_type_primary (*) process_type_secondary (*)	f_core	VARCHAR	QLI QGM Q1R Q2R TSI VAF VLH VPF VBL VLV VFH FSS FCF FFF FPF LAV LSL TCY ETC EHT ECD DTS DTM DTH DTA WFI TOR

Table 2 - continued. New fields and data type required for the MOVER refactoring. Fields and data type marked as (*) are new, the ones marked as (**) existed in the previous version of MOVER but have been either modified and/or expanded.

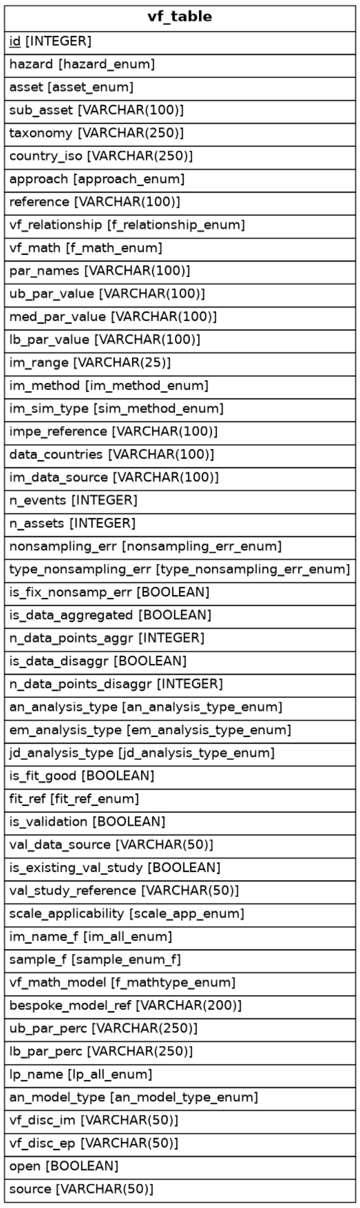
Fields	**Table**	**Data type**	**Description**
occupancy (*)	f_core	cf_common.occupancy_enum,	Residential, Commercial, Industrial, Infrastructure, Healthcare, Educational, Government, Crop, Livestock, Forestry, Mixed
taxonomy_source (*)	f_core	VARCHAR	E.g., GEM, GED4ALL, OSM
applicability_notes	f_core	VARCHAR Please note that we have reverted this field into a VARCHAR after initially considering having an Enumerative field listing the regions identified by the WB plus “Global”. This decision was made because the WB regions have very large spatial extents and there would not be functions applicable to such large areas.	This field is defined as a specific sub-area within a country and/or region. E.g., East-Africa
function_type (*)	f_core	function_type_enum	Fragility, Vulnerability, Damage-to-Loss

Table 2 - continued. New fields and data type required for the MOVER refactoring. Fields and data type marked as (*) are new, the ones marked as (**) existed in the previous version of MOVER but have been either modified and/or expanded.

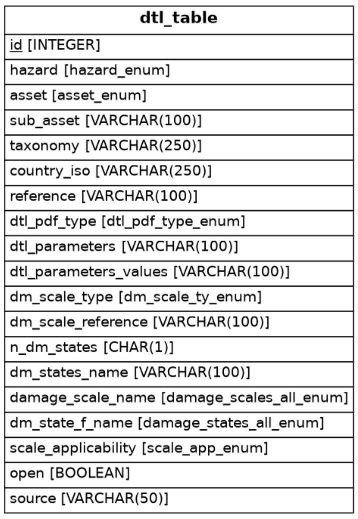
Fields	**Table**	**Data type**	**Description**
approach (**)	f_core	f_subtype_enum (**)	Empirical, Analytical, Judgement, Hybrid - Analytical/Empirical, Hybrid - Analytical/Judgement, Hybrid - Empirical/Judgement, Hybrid - Analytical HF/LF, **Code - based (**)**
f_math_model (**)	f_core	f_mathtype_enum	Cumulative Lognormal, Cumulative Normal, Exponential, Bespoke, **DtL - Beta PDF (**)**, **DtL - Normal PDF (**)**, **DtL - Lognormal PDF (**)**, **DtL - Uniform PDF (**)**, **DtL - Bespoke PDF (**)**
licence_code (**)	f_core	VARCHAR linked in interface with licence_code	Creative Commons CCZero (CC0), Open Data Commons Public Domain Dedication and Licence (PDDL), Creative Commons Attribution 4.0 (CC BY 4.0) Open Data Commons Attribution License(ODC-By), Creative Commons Attribution Share-Alike 4.0 (CC BY-SA 4.0,), Open Data Commons Open Database License (ODbL)



ff_table
id [INTEGER]
hazard [hazard_enum]
asset [asset_enum]
sub_asset [VARCHAR(100)]
taxonomy [VARCHAR(250)]
country_iso [VARCHAR(250)]
approach [approach_enum]
reference [VARCHAR(100)]
dm_scale_type [dm_scale_ty_enum]
dm_scale_reference [VARCHAR(100)]
n_dm_states [CHAR(1)]
dm_states_name [VARCHAR(100)]
edp_dmstate_thre [VARCHAR(100)]
ff_relationship [f_relationship_enum]
ff_math [f_math_enum]
par_names [VARCHAR(100)]
ub_par_value [VARCHAR(100)]
med_par_value [VARCHAR(100)]
lb_par_value [VARCHAR(100)]
ff_disc_im [VARCHAR(100)]
ff_disc_ep [VARCHAR(100)]
im_range [VARCHAR(25)]
im_method [im_method_enum]
im_sim_type [sim_method_enum]
impe_reference [VARCHAR(100)]
data_countries [VARCHAR(100)]
im_data_source [VARCHAR(100)]
n_events [INTEGER]
n_assets [INTEGER]
nonsampling_err [nonsampling_err_enum]
type_nonsampling_err [type_nonsampling_err_enum]
is_fix_nonsamp_err [BOOLEAN]
is_data_aggregated [BOOLEAN]
n_data_points_aggr [INTEGER]
is_data_disaggr [BOOLEAN]
n_data_points_disaggr [INTEGER]
an_analysis_type [an_analysis_type_enum]
em_analysis_type [em_analysis_type_enum]
jd_analysis_type [jd_analysis_type_enum]
is_fit_good [BOOLEAN]
fit_ref [fit_ref_enum]
is_validation [BOOLEAN]
val_data_source [VARCHAR(50)]
is_existing_val_study [BOOLEAN]
val_study_reference [VARCHAR(50)]
scale_applicability [scale_app_enum]
sample_f [sample_enum_f]
bespoke_model_ref [VARCHAR(200)]
ff_math_modelf [f_mathtype_enum]
dm_state_f_name [damage_states_all_enum]
damage_scale_name [damage_scales_all_enum]
im_name_f [im_all_enum]
edp_name_all [edp_name_enum]
ub_par_perc [VARCHAR(100)]
lb_par_perc [VARCHAR(100)]
an_model_type [an_model_type_enum]
open [BOOLEAN]
source [VARCHAR(50)]

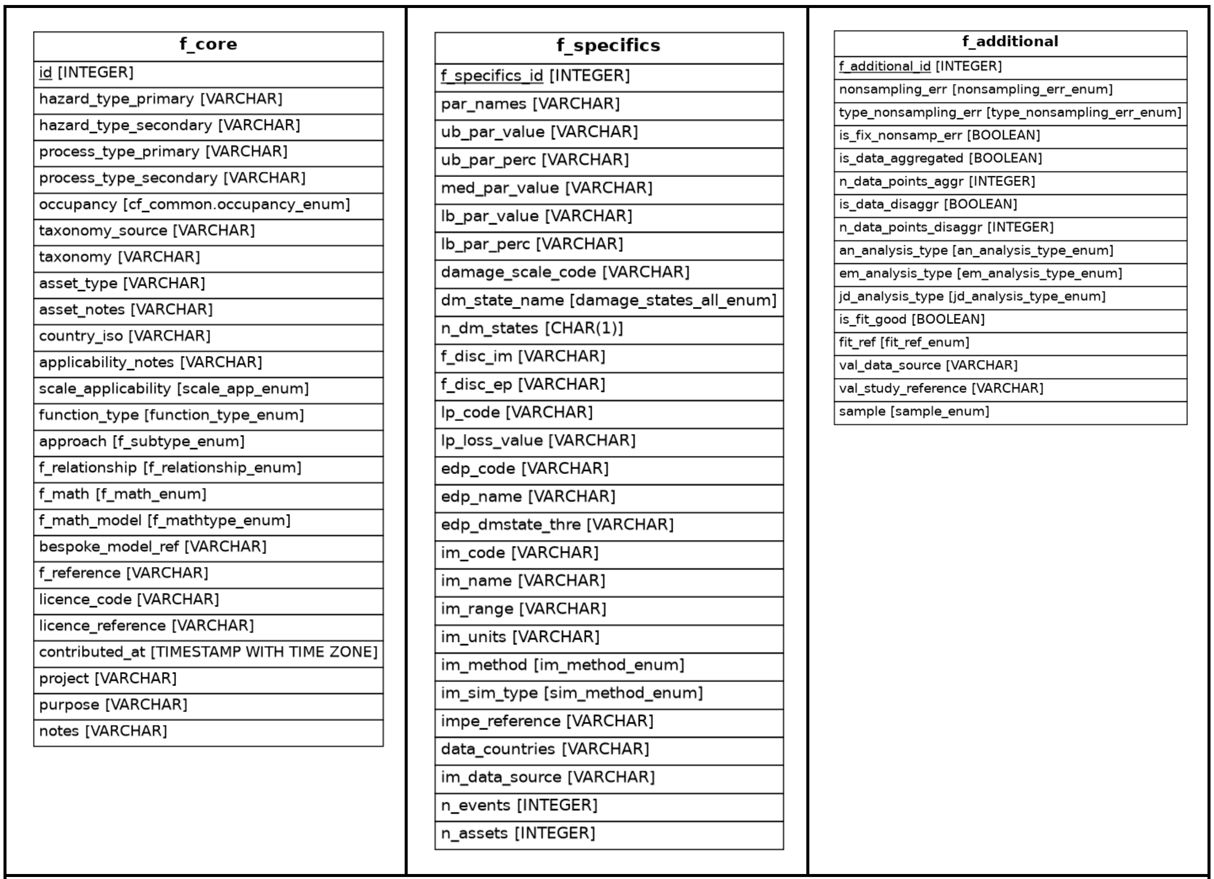


vf_table
id [INTEGER]
hazard [hazard_enum]
asset [asset_enum]
sub_asset [VARCHAR(100)]
taxonomy [VARCHAR(250)]
country_iso [VARCHAR(250)]
approach [approach_enum]
reference [VARCHAR(100)]
vf_relationship [f_relationship_enum]
vf_math [f_math_enum]
par_names [VARCHAR(100)]
ub_par_value [VARCHAR(100)]
med_par_value [VARCHAR(100)]
lb_par_value [VARCHAR(100)]
im_range [VARCHAR(25)]
im_method [im_method_enum]
im_sim_type [sim_method_enum]
impe_reference [VARCHAR(100)]
data_countries [VARCHAR(100)]
im_data_source [VARCHAR(100)]
n_events [INTEGER]
n_assets [INTEGER]
nonsampling_err [nonsampling_err_enum]
type_nonsampling_err [type_nonsampling_err_enum]
is_fix_nonsamp_err [BOOLEAN]
is_data_aggregated [BOOLEAN]
n_data_points_aggr [INTEGER]
is_data_disaggr [BOOLEAN]
n_data_points_disaggr [INTEGER]
an_analysis_type [an_analysis_type_enum]
em_analysis_type [em_analysis_type_enum]
jd_analysis_type [jd_analysis_type_enum]
is_fit_good [BOOLEAN]
fit_ref [fit_ref_enum]
is_validation [BOOLEAN]
val_data_source [VARCHAR(50)]
is_existing_val_study [BOOLEAN]
val_study_reference [VARCHAR(50)]
scale_applicability [scale_app_enum]
im_name_f [im_all_enum]
sample_f [sample_enum_f]
vf_math_model [f_mathtype_enum]
bespoke_model_ref [VARCHAR(200)]
ub_par_perc [VARCHAR(250)]
lb_par_perc [VARCHAR(250)]
lp_name [lp_all_enum]
an_model_type [an_model_type_enum]
vf_disc_im [VARCHAR(50)]
vf_disc_ep [VARCHAR(50)]
open [BOOLEAN]
source [VARCHAR(50)]



dtl_table
id [INTEGER]
hazard [hazard_enum]
asset [asset_enum]
sub_asset [VARCHAR(100)]
taxonomy [VARCHAR(250)]
country_iso [VARCHAR(250)]
reference [VARCHAR(100)]
dtl_pdf_type [dtl_pdf_type_enum]
dtl_parameters [VARCHAR(100)]
dtl_parameters_values [VARCHAR(100)]
dm_scale_type [dm_scale_ty_enum]
dm_scale_reference [VARCHAR(100)]
n_dm_states [CHAR(1)]
dm_states_name [VARCHAR(100)]
damage_scale_name [damage_scales_all_enum]
dm_state_f_name [damage_states_all_enum]
scale_applicability [scale_app_enum]
open [BOOLEAN]
source [VARCHAR(50)]

Figure 2a - Diagrams of the ff_table, vf_table and dtl_table (left to right) as in the previous version of MOVER. The images show that many of the fields used in the ff_table and vf_table repeat. This issue has been addressed in the refactored schema.



f_core
id [INTEGER]
hazard_type_primary [VARCHAR]
hazard_type_secondary [VARCHAR]
process_type_primary [VARCHAR]
process_type_secondary [VARCHAR]
occupancy [cf_common.occupancy_enum]
taxonomy_source [VARCHAR]
taxonomy [VARCHAR]
asset_type [VARCHAR]
asset_notes [VARCHAR]
country_iso [VARCHAR]
applicability_notes [VARCHAR]
scale_applicability [scale_app_enum]
function_type [function_type_enum]
approach [f_subtype_enum]
f_relationship [f_relationship_enum]
f_math [f_math_enum]
f_math_model [f_mathtype_enum]
bespoke_model_ref [VARCHAR]
f_reference [VARCHAR]
licence_code [VARCHAR]
licence_reference [VARCHAR]
contributed_at [TIMESTAMP WITH TIME ZONE]
project [VARCHAR]
purpose [VARCHAR]
notes [VARCHAR]

f_specifics
f_specifics_id [INTEGER]
par_names [VARCHAR]
ub_par_value [VARCHAR]
ub_par_perc [VARCHAR]
med_par_value [VARCHAR]
lb_par_value [VARCHAR]
lb_par_perc [VARCHAR]
damage_scale_code [VARCHAR]
dm_state_name [damage_states_all_enum]
n_dm_states [CHAR(1)]
f_disc_im [VARCHAR]
f_disc_ep [VARCHAR]
lp_code [VARCHAR]
lp_loss_value [VARCHAR]
edp_code [VARCHAR]
edp_name [VARCHAR]
edp_dmstate_thre [VARCHAR]
im_code [VARCHAR]
im_name [VARCHAR]
im_range [VARCHAR]
im_units [VARCHAR]
im_method [im_method_enum]
im_sim_type [sim_method_enum]
impe_reference [VARCHAR]
data_countries [VARCHAR]
im_data_source [VARCHAR]
n_events [INTEGER]
n_assets [INTEGER]

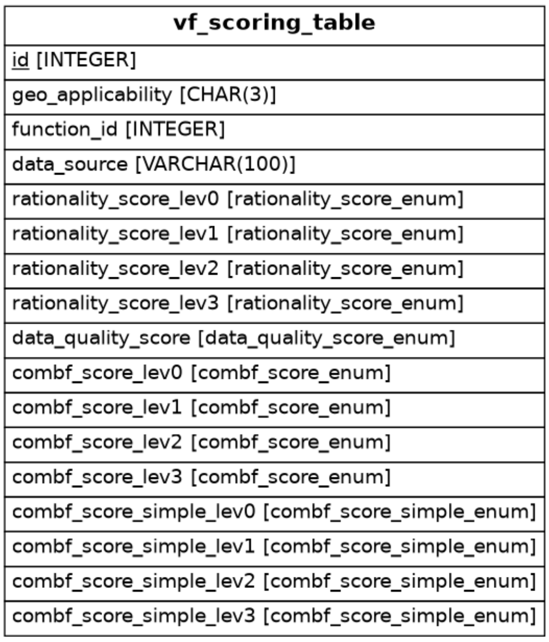
f_additional
f_additional_id [INTEGER]
nonsampling_err [nonsampling_err_enum]
type_nonsampling_err [type_nonsampling_err_enum]
is_fix_nonsamp_err [BOOLEAN]
is_data_aggregated [BOOLEAN]
n_data_points_aggr [INTEGER]
is_data_disaggr [BOOLEAN]
n_data_points_disaggr [INTEGER]
an_analysis_type [an_analysis_type_enum]
em_analysis_type [em_analysis_type_enum]
jd_analysis_type [jd_analysis_type_enum]
is_fit_good [BOOLEAN]
fit_ref [fit_ref_enum]
val_data_source [VARCHAR]
val_study_reference [VARCHAR]
sample [sample_enum]

Figure 2b - Diagrams of the f_core, f_specidics and f_additional tables (left to right), no longer showing repeated fields and with a more coherent, data-driven structure.

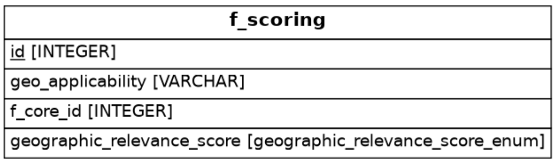
Changes to the scoring table

In the previous version of MOVER the fragility functions and the vulnerability functions tables linked to two separate scoring tables. Following the refactoring of the MOVER schema and the simplified design of the core module, the **f_core** table links to a single **scoring_table**. The fields of the scoring table have also been updated to reflect the changed scoring system. As already mentioned in the Inception report, the scoring of the functions in the previous version of MOVER was a combination of a rationality_score and a data_quality score. The revised scoring system instead focuses only on the geographic relevance of the function, which essentially describes how good a choice a function developed for one country is when applied to another. The scoring considers the scale applicability as well as the geographical applicability (e.g. if a vulnerability function is developed for use at national level, it may not be appropriate for the loss assessment of a single asset). The components of the geographical scoring remain those of the original MOVER data schema.

Figure 3 (a,b) show a comparison of the ER diagrams of the **scoring_table** before and after the refactoring



vf_scoring_table
id [INTEGER]
geo_applicability [CHAR(3)]
function_id [INTEGER]
data_source [VARCHAR(100)]
rationality_score_lev0 [rationality_score_enum]
rationality_score_lev1 [rationality_score_enum]
rationality_score_lev2 [rationality_score_enum]
rationality_score_lev3 [rationality_score_enum]
data_quality_score [data_quality_score_enum]
combf_score_lev0 [combf_score_enum]
combf_score_lev1 [combf_score_enum]
combf_score_lev2 [combf_score_enum]
combf_score_lev3 [combf_score_enum]
combf_score_simple_lev0 [combf_score_simple_enum]
combf_score_simple_lev1 [combf_score_simple_enum]
combf_score_simple_lev2 [combf_score_simple_enum]
combf_score_simple_lev3 [combf_score_simple_enum]



f_scoring
id [INTEGER]
geo_applicability [VARCHAR]
f_core_id [INTEGER]
geographic_relevance_score [geographic_relevance_score_enum]

Figure 3 a,b - Diagrams of the old scoring table (left) which structure applied to both ff_table and vf_table and the new simplified scoring system (right), focusing on the geographical relevance of the function.

Changes to the supporting DMs, LPs, EDPs tables in the MOVER schema and IMTs table in the cf_common schema

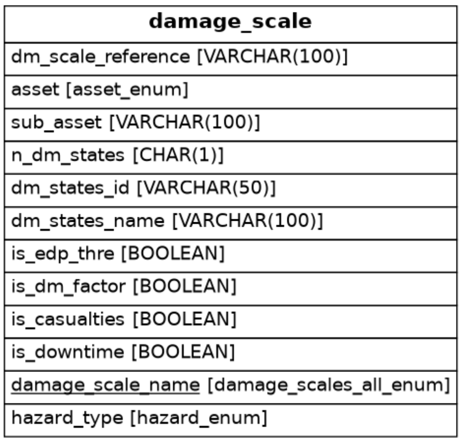
In the original version of MOVER, the damage scale, IM, EDP and LP supporting tables were used as dictionaries for storing pre-defined entries and their descriptions.

The names of the damage scales (DMs), intensity measures (IMs -now renamed intensity measures types or IMTs), Engineering Demand Parameters (EDPs) and Loss Parameters (LPs) were enumerated and set as primary keys. In the refactored MOVER, the enumerative data type has been relaxed so that users can contribute to these tables with new entries. The refactored schema is thus more flexible but also more future-proof, as the new design allows the schema to store data from new upcoming and future studies working with new DMs, LPs or EDPs. In the refactored MOVER, the **IM** table has been replaced by a new table in the **cf_common** schema listing the harmonised intensity measure types **(cf_common.imt)**. Like the DMs, LPs, and EDPs tables, **cf_common.imt** table is designed to accept user contributions.

From a technical standpoint, the **ENUM** fields storing the names of the damage scales, LPs, IMTs and EDPs have been reverted to a **VARCHAR** data type. For the LPs, IMTs and EDPs, the **symbol** fields has been now renamed as **code** and the LPs, IMTs and EDPs codes are now set as the primary keys. In the **cf_common.imt** table the im_code is now linked specifically to the process_type.

The damage scale table did not have a code in the previous MOVER version. The damage scale names are also long and thus unsuitable to be a primary key. To solve this issue, whilst a **damage_scale_code** has been created for standardisation of the supporting tables, the primary key of the **damage_scale** table is a **SERIAL ID**. This slightly different design can be masked once the interface is created.

Figure 4 (a,b), Figure 5 (a,b), and Figure 6 (a,b) respectively show a comparison of the ER diagrams of the **damage_scale** table, of the **lp_table** and of the **edp_table** before and after the refactoring. Figure 7 (a, b) compares the old **im_table** and the **cf_common.imt** table.



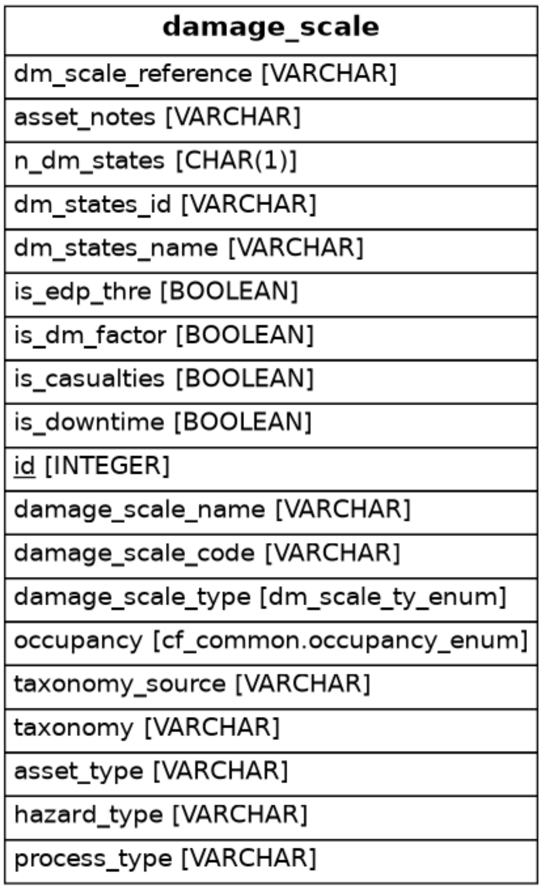
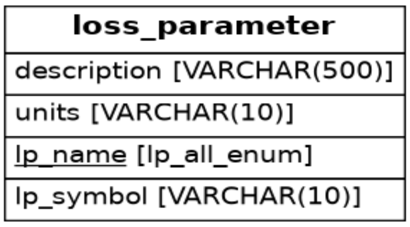


Figure 4 a,b - Diagrams of the old damage scale table (left) and the new one (right), which allows damage scales for multiple hazard type and process types. A damage scale code was also created for consistency with the structure of the other supporting tables.



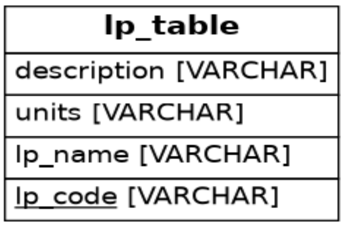
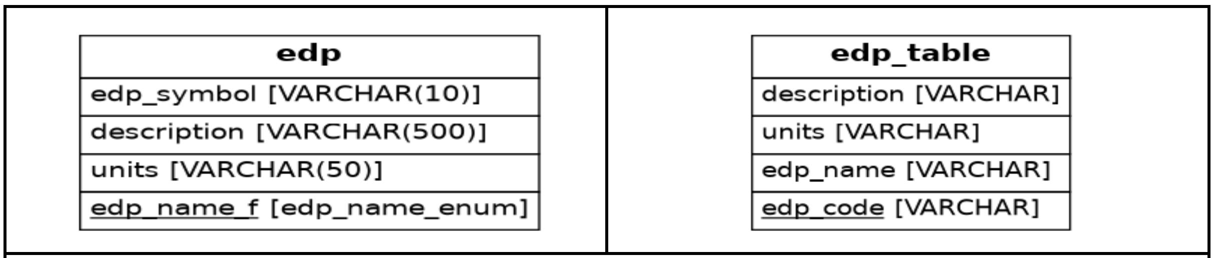


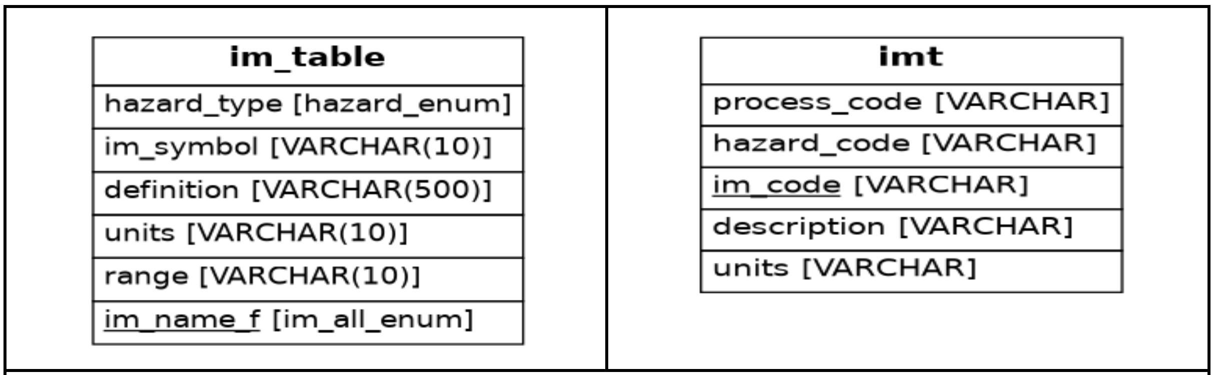
Figure 5 a,b - Diagrams of the old loss parameter table (left) and the new one (right). The lp_code field has replaced the lp_symbol.



edp
edp_symbol [VARCHAR(10)]
description [VARCHAR(500)]
units [VARCHAR(50)]
edp_name_f [edp_name_enum]

edp_table
description [VARCHAR]
units [VARCHAR]
edp_name [VARCHAR]
edp_code [VARCHAR]

Figure 6 a,b - Diagrams of the old loss EDP table (left) and the new one (right). The edp_code field has replaced the edp_symbol.



im_table
hazard_type [hazard_enum]
im_symbol [VARCHAR(10)]
definition [VARCHAR(500)]
units [VARCHAR(10)]
range [VARCHAR(10)]
im_name_f [im_all_enum]

imt
process_code [VARCHAR]
hazard_code [VARCHAR]
im_code [VARCHAR]
description [VARCHAR]
units [VARCHAR]

Figure 7 a,b - Diagrams of the old loss EDP table (left) and the new one (right). The edp_code field has replaced the edp_symbol.

MOVER Import / Export Scripts (D3)

The MOVER Import/Export have been developed in Python 3 and work on a table-by-table basis. The assumption here is that, in order to fulfil the dependencies of the functions_core module on the supporting tables, the data input has to follow a specific workflow to ensure that the damage scales, IMTs, EDPs and LPS values that the functions refer to already exist when the function is inserted in MOVER. If they do not exist, they need to be created before the new function is added.

The Import/ Export scripts are available on the GitHub Repository, see the related documentation here: https://github.com/enricaverrucci/movercf#importexport

Updates to other Databases and scripts (D4)

As described in D1, the contribution tables of the Hazard, Exposure and Loss databases have all been updated to include a free-text *project* field and a *contributed_at* timestamp field which is automatically set to the current time when a new contribution is added. The *cf_common.license* table was also updated to use the license code (e.g "CC BY-SA 3.0" or "ODbL) as the primary key rather than a numeric *id* field, and the corresponding *contribution* tables updated to replace the *license_id* field with *license_code*. This change simplifies the schema and import/export code and also makes it easier to inspect the data manually.

One of the more important changes applied to the database schema was the introduction of the *cf_common.imt* table and the use of database constraints to prevent the use of hazard intensity types not present in the table. A discussion of the motivation for this change and the agreed list of valid intensity types is presented in the “Hazard Intensity Measure Types” section later in this report.

The existing Python import / export scripts were updated to accommodate the changes to the schema also revised to support Python 3, since Python 2 having reached “end of life” on January 1st 2020. Previous versions of the import/export scripts made use of the Python Django library to manage database connections, however we have removed this dependency and instead make use of the PsycoPG2 database driver directly. We also took advantage of the occasion to test the schema and Python code with more recent versions of PostgreSQL; in particular versions 10 and 11 both with PostGIS version 2.4. The “Deployment Considerations” section contains a brief discussion of issues encountered and recommendations for those looking to deploy the database in an operational context.

We also took advantage of this opportunity to perform some code cleanup in both Python and SQL code; the utility scripts for creating and testing initial database instances are now both more uniform and easier to apply to different target PostgreSQL installations. In addition, the GED4ALL *level2* namespace has been renamed to *ged4all*. While this change was not strictly speaking necessary to meet the project requirements, we feel that the old name was misleading.

All the Python and SQL code is made available under the terms of an open licence from the following github repositories:

https://github.com/gem/hazard_scenario_database
https://github.com/gem/ged4all
https://github.com/gem/loss_database

For all three repositories, revised import/export Python code can be found in “python” folders, SQL code in “sql” folders.

Unified Challenge Fund Database Schema

Following discussions regarding handling of common hazard, process and intensity measure types, it became clear that combining the four database schema into a single database instance would be not merely beneficial but indispensable in order to guarantee consistency across the four database schema.

The figure below depicts the relationships between the tables in the resulting unified database. Tables from the *hazard* schema are shown in red, those from *ged4gem* in green, *loss* tables are shown in violet, *mover* tables are shown in blue and *cf_common* tables are shown in yellow. While all tables are displayed, most fields are hidden, only “key fields” (those being used as a primary or foreign database lookup key) are displayed.

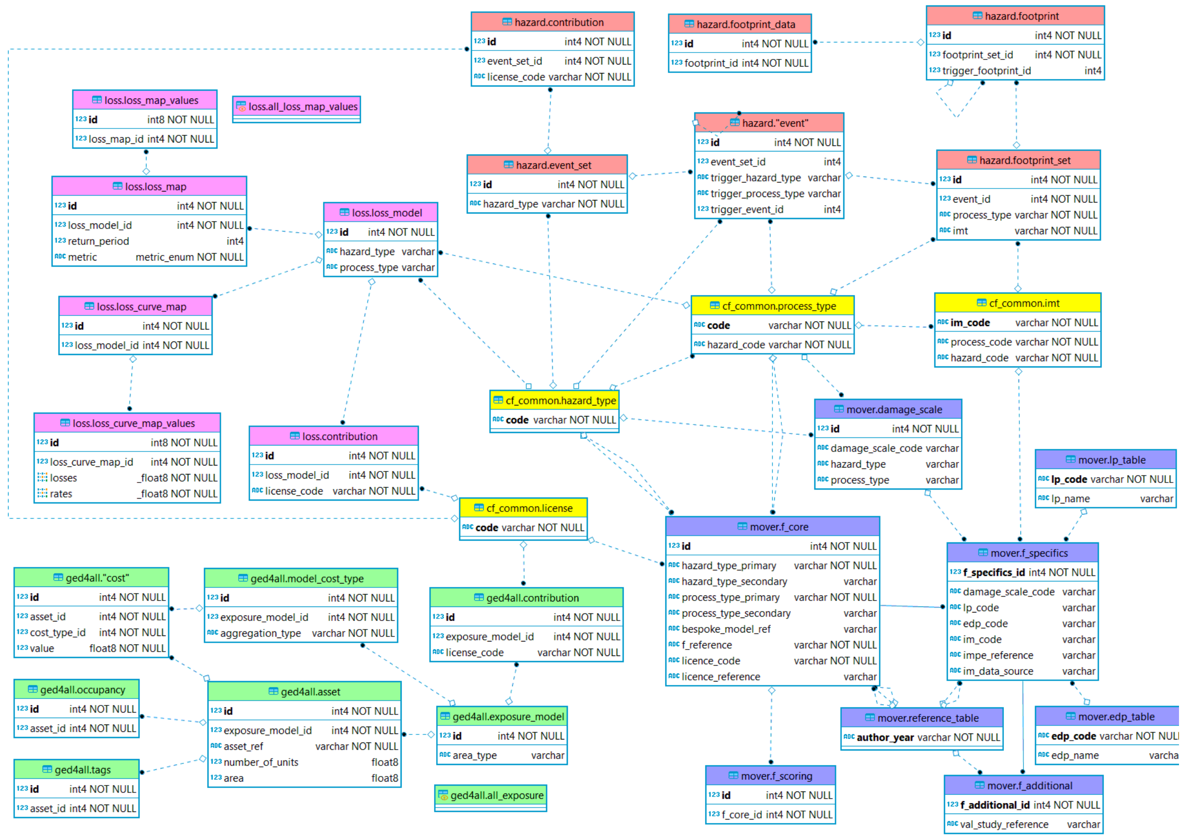


Figure 8: Simplified E-R overview of the combined Challenge Fund Database showing relationships between tables.

This diagram emphasizes the interconnected nature of the four database schema, and in particular the central role of the *cf_common* tables: *hazard_type*, *process_type* and *imt* as touchpoints between hazard, loss and vulnerability.

The following github repository contains the SQL and Python code for a unified database containing all four Challenge Fund databases, with a single copy of the *cf_common* namespace: https://github.com/gem/cf_db

Hazard Intensity Measure Types

The Terms of Reference document notes the need for consistency regarding Intensity Measure Types (IMT) across the various database schemas. This was a topic of discussion and activity during the previous database harmonization exercise, however at the time this project was started a number of problems remained unsolved, in our technical proposal document we stated:

"It would be desirable to include one or more database level constraints to ensure that the specified IMT is valid for the given hazard; however as described earlier, we have not been able to find an elegant solution for the hazard database and are not currently able to propose a solution for MOVER. Applying strict constraints which complicate or prevent contributors from sharing useful datasets is unlikely to

further the goal of encouraging a standard unless there is critical mass of support for adopting a standard and tools available to facilitate conversion."

During the execution of this project we continued discussions on this topic between GEM, UCL and GFDRR. In particular when discussing the use of existing database contents for use cases in the context of the proposed Risk Data Library portal, we found that we were forced to reconsider the implications of adopting a more structured approach to intensity types enforced by strict database constraints.

For some IMTs such as Peak Ground Acceleration (PGA) or Spectral Acceleration of a given Period, T (SA(T)) there are multiple units of measure commonly in use: meters per second squared (m/s2), the Earth's local gravitational acceleration (g), and USGS ShakeMaps use a percentage of g. It must be clear to the user which units are used when selecting and using data. It must also be clear whether or not unit conversion is required when combining hazard with vulnerability. We also wish to avoid "artificial" incompatibilities caused by e.g different ways of spelling common IMT codes. Furthermore, the original MOVER database relies on a more structured approach to IMTs including a FOREIGN KEY constraint to link database tables and prevents insertion of vulnerability and fragility functions using invalid intensity types. While we remain in favour of keeping the data insertion process as simple as possible to avoid discouraging potential contributors, the project team, in agreement with Stuart Fraser of GFDRR were of the opinion that a more rigorous approach to IMTs would reduce the possibility of misinterpreted values and units and limits attempts to use incompatibile datasets.

That said, It is also important to note that intensity measures remain a simplified representation of the hazard (e.g., what is it meant by 'depth'?, Where do you measure it from?). As intensity measures rarely follow standards, even differentiating by hazard type / process type does not guarantee that two IMTs called the same actually refer to the exact same measure.

The resulting IMT codes thus combine in indication of the hazard intensity and an indication of the unit used. We have made us of existing intensity codes where their use is commonplace, such as PGA or SA(T), in other cases we have created new intensity codes referring to the hazard process and unit, for example "v_wi(1m):km/h" for "1-min at 10m sustained wind speed (kph)". A complete list of the IMT codes and their associated hazard and process codes can be found in the appendix at the end of this report.

Hazard Database

The figure below depicts the revised entity-relationship diagram of the hazard database schema.

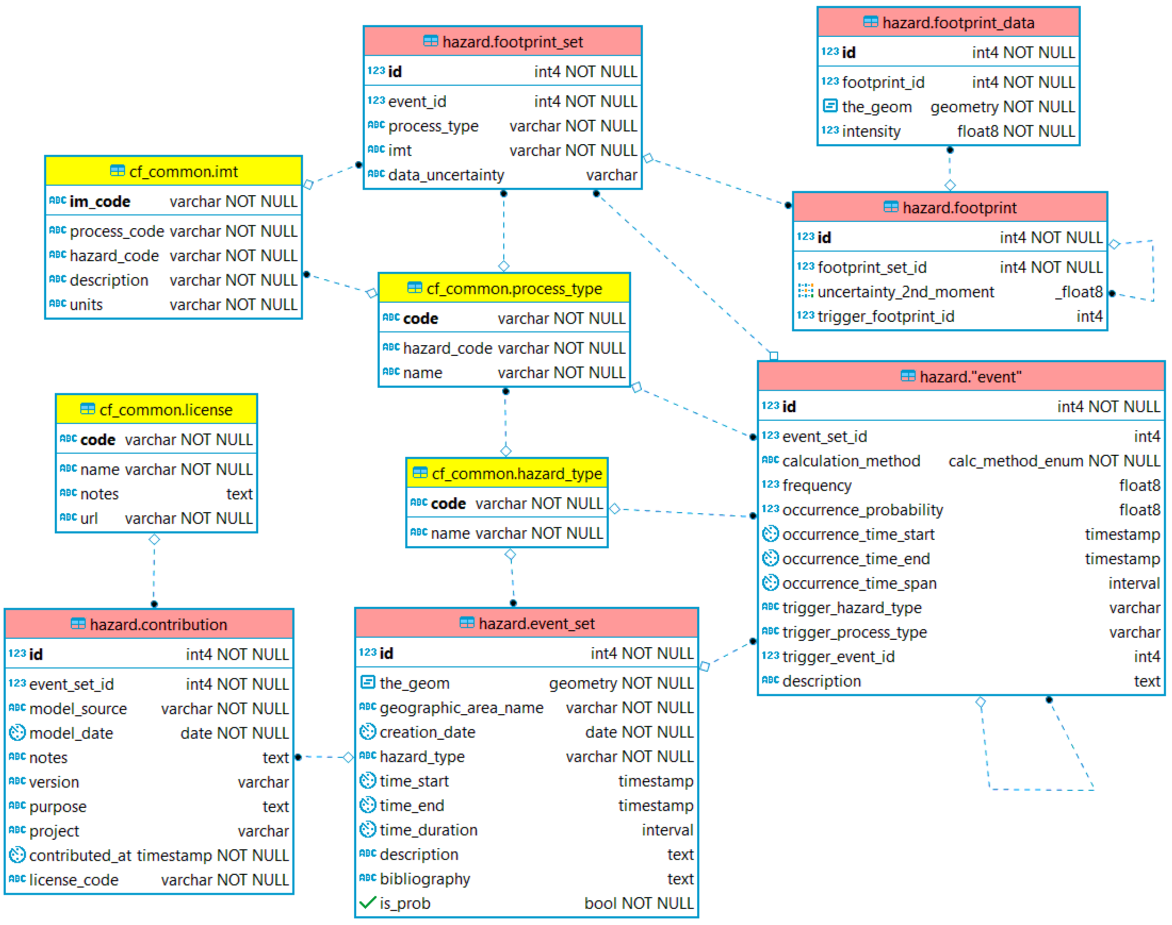


Figure 9: Revised Hazard Schema Entity Relationship Diagram

As expected, there were no significant changes to the structure of the schema, however the introduction of the IMT table and enforceable database constraints implied a number of changes to the database content in order to bring all existing data into line with the agreed IMT and unit codes. The SQL statements used to apply changes to the existing datasets can be found in the `sql/historical/ 2020-01` folder or from the following link:
https://github.com/gem/hazard_scenario_database/tree/master/sql/historical/2020-01_moverint

We also updated the JSON format input files found in the `data/swio` folder. During this phase we also identified some errors in descriptive fields for some datasets, these too were corrected in both JSON files and via SQL.

GED4ALL Exposure Database

As with the hazard database, there were no significant structural changes to the exposure specific part of the schema, but the *cf_common* schema now includes the *hazard_type*, *process_type* and *imt* tables that are not used by ged4all directly but are present so that all four databases have the same structure. Figure 10 shows the revised GED4ALL schema:

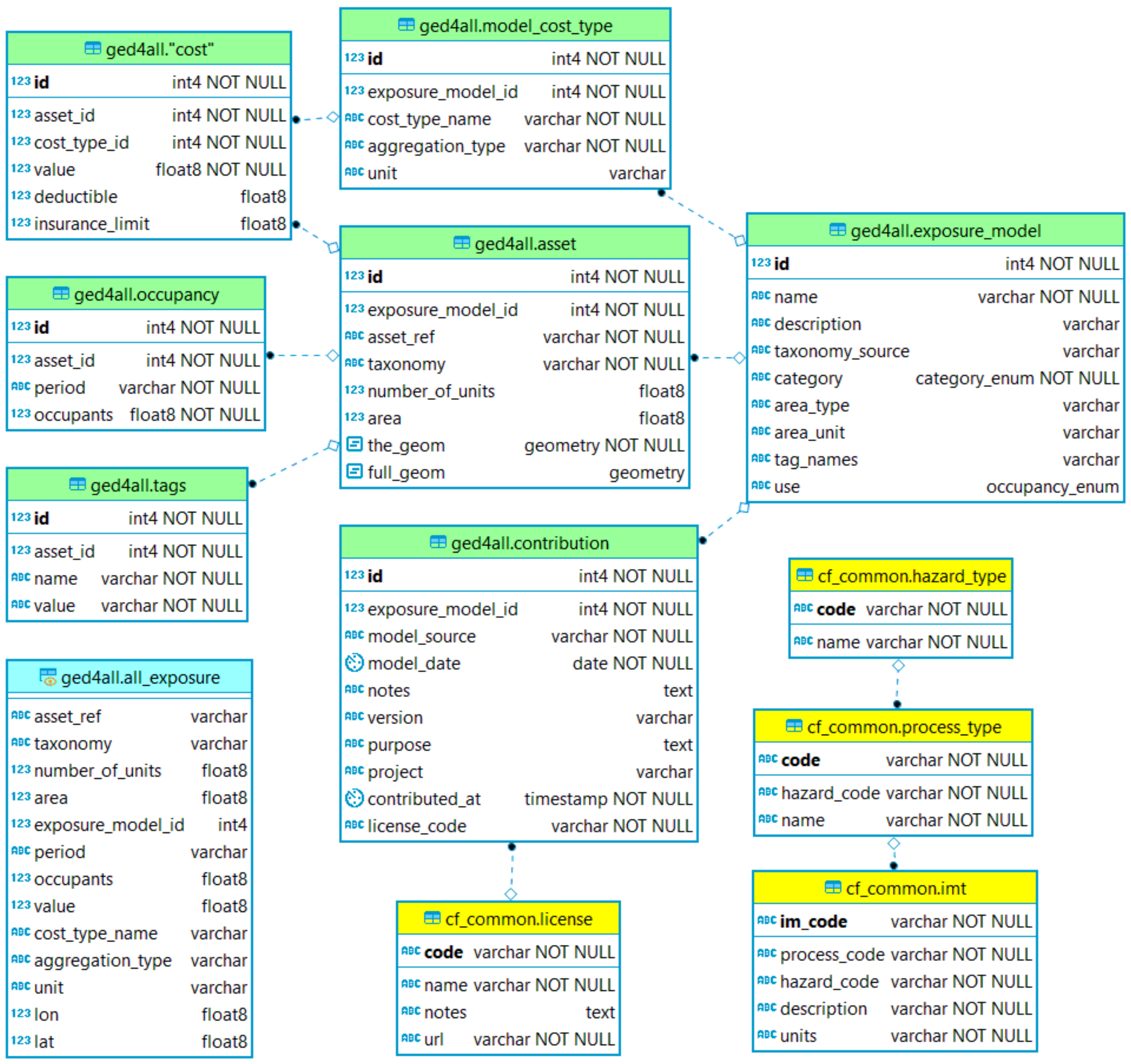


Figure 10: Revised GED4ALL schema

Although not visible on the diagram another new element in the *cf_common* schema, is a stored procedure *taxonomy2human* which was developed in response to the request in D4b for more human readable outputs. This stored procedure uses PostgreSQL's support for calling Python language code to parse valid GEM Building Taxonomy 2.0 strings into English language text. More details on the stored procedure and underlying Python code are provided in the "GEM Building Taxonomy Explainer" section.

While testing the explainer we discovered that some of the GEM Taxonomy strings present in the database were not strictly compliant with the standard due to trivial typos or use of an incorrect separator character; these cases have been fixed with SQL UPDATE statements present in the sql/historical folder.

Loss Database

Unlike the hazard database, the IMT related changes in *cf_common* did not require any changes to the loss schema since the loss_model table does not refer to IMTs and already had FOREIGN KEY database constraints to prevent use of invalid hazard and process codes. The only structural changes required were related to the new fields in the *contribution* table and the use of *license_code* rather than *id*. The figure below depicts the revised *loss* schema.

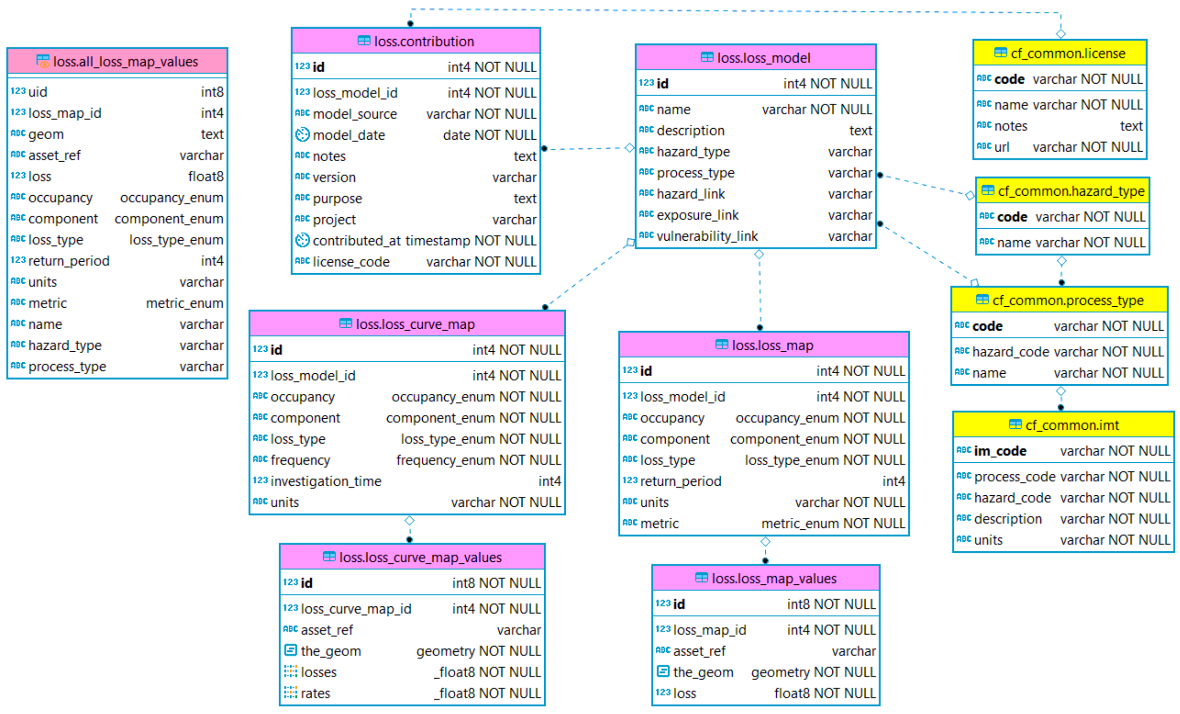


Figure 11: Revised loss schema

Python import / export scripts

As described earlier, all pre-existing input/export scripts we updated to support Python 3 and to remove Django dependencies and, of course, to work with the revised database schemas. In addition, by creating a single github repository for all four databases, it also became possible to organize the code into packages so that the common code for handling database connections or contribution meta-data can be shared between the import/export tools rather than each repository having its own copy.

GEM Building Taxonomy Explainer

In order to provide a more human readable version of the GEM Building Taxonomy strings used in both mover and ged4all databases, we produced a stored procedure cf_common.taxonomy2human which uses plpython3u to call a Python library. We modified a pre-existing interactive software tool build by GEM to provide a package suitable for use as a standalone library. The code for this tool is available on github from the following link:
https://github.com/gem/oq-platform-taxtweb/tree/master/openquake/taxonomy

The following example shows how the stored procedure may be used in a SQL SELECT expression:

```
SELECT a.taxonomy, cf_common.taxonomy2human(a.taxonomy) FROM
ged4all.asset WHERE id=336761;
```

Example outputs (all taken from GED4ALL and MOVER databases) are shown in the table below:

Taxonomy String	**Explainer output**
CR/LWAL/HBET:1,5	Material type: Concrete, reinforced; Type lateral load-resisting system: Wall; Number of storeys above ground - Range of the number of storeys: between 1 and 5.
CR/LWAL/HBET:6,10	Material type: Concrete, reinforced; Type lateral load-resisting system: Wall; Number of storeys above ground - Range of the number of storeys: between 6 and 10.
CU/HBET:1,2/RSH2	Material type: Concrete, unreinforced; Number of storeys above ground - Range of the number of storeys: between 1 and 2; Roof shape: Pitched with gable ends.
EU+ETR/HBET:1,3/	Material type: Earth, unreinforced; Material technology: Rammed earth; Number of storeys above ground - Range of the number of storeys: between 1 and 3.
CR+CIP/LFM+DNO/HEX:6/YBET:1980,1989	Material type: Concrete, reinforced; Material technology: Cast-in-place concrete; Type lateral load-resisting system: Moment frame; System ductility: Non-ductile; Number of storeys above ground - Exact number of storeys: 6; Date of construction or retrofit - Bounds for the date of construction or retrofit: between 1980 and 1989.
CR+CIP/LPB	Material type: Concrete, reinforced; Material technology: Cast-in-place concrete; Type lateral load-resisting system: Post and beam.
MCF+CBH+MOC/HBET:1,4	Material type: Masonry, confined; Material technology: Concrete blocks, hollow; Material properties: Cement mortar; Number of storeys above ground - Range of the number of storeys: between 1 and 4.
SRC+CIP/LPB/HBET:2,5	Material type: Concrete, composite with steel section; Material technology: Cast-in-place concrete; Type lateral load-resisting system: Post and beam; Number of storeys above ground - Range of the number of storeys: between 2 and 5.
SRC+CIP/LPB/HBET:6,10	Material type: Concrete, composite with steel section; Material technology: Cast-in-place concrete; Type lateral load-resisting system: Post and beam; Number of storeys above ground - Range of the

	number of storeys: between 6 and 10.

Deployment Considerations

Throughout the execution of this project we have made use of PostgreSQL 11.6 with PostGIS 2.5 as the underlying software stack via the "postgis 11.0-2.5" Docker image distributed by Kartoza: https://hub.docker.com/r/kartoza/postgis/. This combination has worked well however it does not include native support for the plpython3u extension required to run the Taxonomy Explainer. Furthermore, it should be noted that since Python does not have a "sandbox" feature to limit access to system resources PostgreSQL considers all plpython extensions "untusted", some systems administrators may be uncomfortable about installing this extension in an operational database. For this reason, we consider the installation of the *taxonomy2human* stored procedure an optional extra; it is possible to omit this feature without otherwise compromising the behaviour of the database.

We also note that the latest Ubuntu LTS, 18.04 "Bionic" provides packages for PostgreSQL 10 and PostGIS 2.4 as well as a compatible plpython extension, but support for PostgreSQL 11 is not available unless Ubuntu 19.04 or later is used. We trust that the forthcoming Ubuntu 20.04 release will provide an Ubuntu LTS solution with official packages for all three required packages. An alternative approach might be to use PostgreSQL 10 and PostGIS 2.4 on Ubuntu 18.04 initially and then upgrade both operating system and software stack once LTS versions are available.

Storage and Tablespaces

Some of the datasets contained in the hazard, exposure and loss databases can be very large with several million records for a single contribution. At present the database occupies approximately 100 GB of space but if GFDRR expects to add more contributions, then some thought should be given to future storage needs, particularly if high-resolution datasets such as flood maps are to be included.

PostgreSQL provides the TABLESPACE mechanism to allow database administrators to specify where data for specific tables should be stored. On a system with, for example, small fast drives as well as larger, slower drives, the DBA could create two TABLESPACEs, one for each drive type and use the slower storage for the largest tables (*ged4all.asset, hazard.footprint_data, loss.loss_map_values, loss.loss_curve_map_values*) while using the faster drives to store indices (including those referring to the larger tables) as well as the data for the smaller tables. Further information is available from https://www.postgresql.org/docs/11/manage-ag-tablespaces.html

Scoring Considerations

In the refactored version of MOVER, the scoring of the fragility and vulnerability functions and DtL has been simplified to a system for scoring geo-applicability of these relationships.

In terms of allocating a score to the scale applicability, a set of rules is required for determining how applicable a function derived at a given scale is at other scales. For example, how well a fragility function derived for a single asset can represent the fragility of assets at regional, sub-national and national level. A scoring matrix can be drawn to facilitate this. Such a matrix could be proposed by the authors or by an expert committee assembled by the World Bank, but should be relatively straight forward.

Instead the geo-applicability scores in terms of countries and regions is more complex. Assumptions need to be made in relating the similarity of assets represented in a given function with the assets in other countries and regions. This is highly subjective and the similarity relationships between countries may change according to the asset discussed. For example, a building fragility function for the UK would not be applicable to India, due to strong differences in building construction, however, fragility relationships for bridges might be similar due to the Indian use of British Standards in bridge building. In the future population of the data schema, scoring of the functions needs to be carefully considered. The development of scoring rules for each possible combination of countries and assets is outside the scope of the current project. It requires the determination of a specific methodology, additional research and consultation with several experts.

Limitations and possible future Developments

1) The refactored MOVER already contemplates the possibility to distinguish between primary and secondary hazard types and/or process types when adding a function to the core module. This new functionality was added to be able to provide additional information for functions that look, for instance, at sequential hazards but also to represent correctly empirical fragility and vulnerability function that inherently contain damage or losses from a combination of hazards.

At the time of writing, it has also become apparent that in the near future functions referring to more than one intensity measure and/or damage scale may become available. Due to the lack of test data, it is difficult to foresee which attributes these functions would require to be appropriately described and, in turn, what adjustments will need to be made to MOVER to accommodate the possibility of adding such functions.
Technical solutions to extend/expand the capabilities of MOVER in this direction can be however easily implemented due to the modular design of the schema. For the time being, the coding of these alterations remains outside the scope of this project.

2) In the inception report we had initially foreseen the possibility to mark some fields of the function core module by default as “N/a - Not applicable” for specific function types/subtypes (e.g., damage scale fields being not applicable to vulnerability functions).

This suggestion was based on the idea that the refactored MOVER would have a similar design to the original MOVER, in which Fragility, Vulnerability, and DtL functions were stored in three separate tables. During the course of the project, we have expanded the definition of the function types to include new types that are being used more frequently (e.g., code-based functions) and we have also concluded that the function types used to define fragility and vulnerability functions can be also applied to DtL functions. These revisions have brought us to reconsider the need for function-type specific tables and to opt, instead, from a slimmer and more integrated design. The main trade-off of representing Fragility, Vulnerability and DtL functions with a single **f_core** table and its two dependent tables (**f_specific** and **f_additional**) is that we have less control on the constraints of the fields that are not common to all types of functions. This trade-off was weighed against the benefit of the clearer representation of the functions in the refactored MOVER. It was also considered that having a table for each function type would have not necessarily helped impose more case-fitting constraints for all the fields of the table, as each function type differentiates in several subtypes. The option of using a table inheritance design, which would have allowed for the definition of a hierarchy for each function types and subtypes was at first considered. However, after some tests, it was concluded that this approach would have generated too many tables, fragmented the design and overall rendered the query process more convoluted and impractical.

In lieu of the hard-coded constraints, appropriate and detailed documentation to guide the user in the compilation of the function tables has been provided in the Appendix (See the Function_Types Tables).

3) *In the inception report, we also mentioned the possibility to apply the ON DELETE CASCADE option to foreign key constraints to maintain the referential integrity in the functions table and the supporting reference tables.*

When coding the refactored MOVER, it became apparent that the ON DELETE CASCADE could not be applied to all the foreign keys. As the supporting tables are now open for contributing and alteration from the users, it is more prudent to avoid the automatic ON DELETE CASCADE option. Hence, it will not be possible for functions to be deleted when, for instance, an IMT is deleted. The ON DELETE CASCADE option has been used however in the f_core table and its dependent tables (f_additional, f_specifics and f_scoring). The function ID of the f_core table is referenced in all the dependent tables. In fact, the ID of the f_core table will always find a (1:1) correspondence in the f_specific table (f_core_ID = f_specific_ID) and it may find a (1:1) correspondence in the f_additional table (f_specific_ID = f_additional_ID), depending on data availability. As there might be more than a scoring value associated with each function (e.g., a function is scored for several countries), subjected to data availability, there may be a (1:many) correspondence in the f_scoring table. By deleting a function in the f_core tables, all the attributes of the function stored in any of the associated tables are also deleted.

Appendix - Documentation

IMT Table

Table 3: All valid IMT codes

process_code	hazard_code	im_code	description	units
QGM	EQ	PGA:g	Peak ground acceleration in g	g
QGM	EQ	PGA:m/s2	Peak ground acceleration in m/s2 (meters per second squared)	m/s2
QGM	EQ	PGV:m/s	Peak ground velocity in m/s	m/s
QGM	EQ	SA(0.2):g	Spectral acceleration with 0.2s period	g
QGM	EQ	SA(0.3):g	Spectral acceleration with 0.3s period	g
QGM	EQ	SA(1.0):g	Spectral acceleration with 1.0s period	g
QGM	EQ	SA(3.0):g	Spectral acceleration with 3.0s period	g
QGM	EQ	SA(0.2):m/s2	Spectral acceleration with 0.2s period	m/s2
QGM	EQ	SA(0.3):m/s2	Spectral acceleration with 0.3s period	m/s2
QGM	EQ	SA(1.0):m/s2	Spectral acceleration with 1.0s period	m/s2
QGM	EQ	SA(3.0):m/s2	Spectral acceleration with 3.0s period	m/s2
QGM	EQ	Sd(T1):m	Spectral displacement	m
QGM	EQ	Sv(T1):m/s	Spectral velocity	m/s
QGM	EQ	PGDf:m	Permanent ground deformation	m

QGM	EQ	D_a5-95:s	Significant duration a5-95	s
QGM	EQ	D_a5-75 :s	Significant duration a5-75	s
QGM	EQ	IA:m/s	Arias intensity (IÎ±) or (IA) or (Ia)	m/s
QGM	EQ	Neq:-	Effective number of cycles	-
QGM	EQ	EMS:-	European macroseismic scale	-
QGM	EQ	AvgSa:m/s2	Average spectral acceleration	m/s2
QGM	EQ	I_Np:m/s2	I_Np by BojÃ³rquez and Iervolino	m/s2
QGM	EQ	MMI:-	Modified Mercalli Intensity	-
QGM	EQ	CAV:m/s	Cumulative absolute velocity	m/s
QGM	EQ	D_B:s	Bracketed duration	s
FFF	FL	d_fff:m	Flood water depth	m
FPF	FL	d_fpf:m	Flood water depth	m
FFF	FL	v_fff:m/s	Flood flow velocity	m/s
FPF	FL	v_fpf:m/s	Flood flow velocity	m/s
TCY	WI	v_tcy(3s):km/h	3-sec at 10m sustained wind speed (kph)	km/h
ETC	WI	v_ect(3s):km/h	3-sec at 10m sustained wind speed (kph)	km/h
TCY	WI	v_tcy(1m):km/h	1-min at 10m sustained wind	km/h

			speed (kph)	
ETC	WI	v_ect(1m):km/h	1-min at 10m sustained wind speed (kph)	km/h
TCY	WI	v_tcy(10m):km/h	10-min sustained wind speed (kph)	km/h
ETC	WI	v_etc(10m):km/h	10-min sustained wind speed (kph)	km/h
TCY	WI	PGWS_tcy:km/h	Peak gust wind speed	km/h
ETC	WI	PGWS_ect:km/h	Peak gust wind speed	km/h
LSL	LS	d_lsl:m	Landslide flow depth	m
LSL	LS	I_DF:m3/s2	Debris-flow intensity index	m3/s2
LSL	LS	v_lsl:m/s2	Landslide flow velocity	m/s2
LSL	LS	MFD_lsl:m	Maximum foundation displacement	m
LSL	LS	SD_lsl:m	Landslide displacement	m
LSL	LS	LSI:-	Landslide susceptibility Index	-
LSL	LS	haz_lsl:-	Landslide hazard index	-
TSI	TS	Rh_tsi:m	Tsunami wave runup height	m
TSI	TS	d_tsi:m	Tsunami inundation depth	m
TSI	TS	MMF:m4/s2	Modified momentum flux	m4/s2
TSI	TS	F_drag:kN	Drag force	kN
TSI	TS	Fr:-	Froude number	-
TSI	TS	v_tsi:m/s	Tsunami velocity	m/s

TSI	TS	F_QS:kN	Quasi-steady force	kN
TSI	TS	MF:m3/s2	Momentum flux	m3/s2
TSI	TS	h_tsi:m	Tsunami wave height	m
TSI	TS	Fh_tsi:m	Tsunami Horizontal Force	kN
VAF	VO	h_vaf:m	Ash fall thickness	m
VAF	VO	L_vaf:kg/m2	Ash loading	kg/m2
FSS	CF	v_fss:m/s	Maximum water velocity	m/s
FSS	CF	d_fss:m	Storm surge inundation depth	m
DTA	DR	CMI:-	Crop Moisture Index	-
DTM	DR	PDSI:-	Palmer Drought Severity Index	-
DTM	DR	SPI:-	Standard Precipitation Index	-

Function Types Tables

Table 4 lists all fields of the **f_core**, **f_specific**, **f_additional**, and **f_scoring** tables color-coded based on how they apply to the function types and subtypes. The table offers a better understanding of the constraints applied to each field and guides the user in the compilation of the tables of the functions core module.

All the table are to be interpreted based on the Legend here below:

Key	Description	Applicability of NULL/NOT NULL Constraint	User-guide for data input
🟩	Common to all subtypes and compulsory	NOT NULL	Must fill in
🟩	Common to all subtypes but not compulsory	Has to be Nullable by design	Can leave it blank/fill in depending on data availability
🟨	Not common to all subtypes but compulsory for some	Has to be Nullable by design	Must fill info only if relevant to the subtype – Mark as N/a for the others
N/A	No applicable to a specific function type	Has to be Nullable by design	Mark as N/a

Table	**Fields**	**Fragility Functions**					
		Empirical	**Analytical**	**Judgment**	**Hybrid A/E**	**Hybrid A/J**	**Hy E**
f_core	id	🟩	🟩	🟩	🟩	🟩	🟩
	hazard_type_primary	🟩	🟩	🟩	🟩	🟩	🟩
	hazard_type_secondary	▨	▨	▨	▨	▨	▨
	process_type_primary	🟩	🟩	🟩	🟩	🟩	🟩
	process_type_secondary	▨	▨	▨	▨	▨	▨
	occupancy	🟩	🟩	🟩	🟩	🟩	🟩
	taxonomy_source	▨	▨	▨	▨	▨	▨
	taxonomy	▨	▨	▨	▨	▨	▨
	asset_type	🟩	🟩	🟩	🟩	🟩	🟩
	asset_notes	▨	▨	▨	▨	▨	▨
	country_iso	🟩	🟩	🟩	🟩	🟩	🟩
	applicability_notes	▨	▨	▨	▨	▨	▨
	scale_applicability	🟩	🟩	🟩	🟩	🟩	🟩
	function_type (ff/vf/dtl)	🟩	🟩	🟩	🟩	🟩	🟩
	approach (8subtypes)	🟩	🟩	🟩	🟩	🟩	🟩
	f_relationship (math/disc)	🟩	🟩	🟩	🟩	🟩	🟩
	f_math (par/bespoke)	N/A to discrete functions					
	f_math_model	N/A to discrete or Mathematical bespoke funct					
	bespoke_model_ref	N/A to discrete and to Mathematical NON besp					
	f_reference	🟩	🟩	🟩	🟩	🟩	🟩
	licence	🟩	🟩	🟩	🟩	🟩	🟩
	licence_reference	▨	▨	▨	▨	▨	▨
	contributed_at (timestamp)	🟩	🟩	🟩	🟩	🟩	🟩
	project	🟩	🟩	🟩	🟩	🟩	🟩
	purpose	🟩	🟩	🟩	🟩	🟩	🟩
	notes	🟩	🟩	🟩	🟩	🟩	🟩

Table	Fields	Fragility Functions						
		Empirical	**Analytical**	**Judgment**	**Hybrid A/E**	**Hybrid A/J**	**Hybrid E/J**	**H**
f_specifics	f_specifics_id	🟩	🟩	🟩	🟩	🟩	🟩	🟩
	par_names	N/A to discrete						
	ub_par_value	N/A to discrete						
	ub_par_perc	N/A to discrete						
	med_par_value	N/A to discrete						
	lb_par_value	N/A to discrete						
	lb_par_perc	N/A to discrete						
	damage_scale_code	🟩	🟩	🟩	🟩	🟩	🟩	🟩
	dm_state_name	🟩	🟩	🟩	🟩	🟩	🟩	🟩
	n_dm_states	🟩	🟩	🟩	🟩	🟩	🟩	🟩
	f_disc_im	N/A to Mathematical functions						
	f_disc_ep	N/A to Mathematical functions						
	lp_code	N/A						
	lp_code_value	N/A						
	edp_code	N/A	🟩	N/A	🟩	🟩	N/A	🟩
	edp_name	N/A	🟩	N/A	🟩	🟩	N/A	🟩
	edp_dmstate_thre	N/A	🟩	N/A	🟩	🟩	N/A	🟩
	im_code	🟩	🟩	🟩	🟩	🟩	🟩	🟩
	im_name	🟩	🟩	🟩	🟩	🟩	🟩	🟩
	im_units	🟩	🟩	🟩	🟩	🟩	🟩	🟩
	im_range	🟩	🟩	🟩	🟩	🟩	🟩	🟩
	im_method	🟩	🟩	🟩	🟩	🟩	🟩	🟩
	im_sim_type	🟩	🟩	🟩	🟩	🟩	🟩	🟩
	impe_reference	🟩	🟩	🟩	🟩	🟩	🟩	🟩
	data_countries	🟩	🟩	🟩	🟩	🟩	🟩	🟩
	im_data_source	🟩	🟩	🟩	🟩	🟩	🟩	🟩
	n_events	🟩	🟩	🟩	🟩	🟩	🟩	🟩
	n_assets	🟩	🟩	🟩	🟩	🟩	🟩	🟩

Table	**Fields**	**Fragility Functions**					
		Empirical	**Analytical**	**Judgment**	**Hybrid A/E**	**Hybrid A/J**	**Hybrid E/J**
f_additional	f_additional_id	🟩	🟩	🟩	🟩	🟩	🟩
	nonsampling_err	🟩	🟩	🟩	🟩	🟩	🟩
	type_nonsampling_err	🟩	🟩	🟩	🟩	🟩	🟩
	is_fix_nonsamp_err	🟩	🟩	🟩	🟩	🟩	🟩
	is_data_aggregated	🟩	🟩	🟩	🟩	🟩	🟩
	n_data_points_aggr	🟩	🟩	🟩	🟩	🟩	🟩
	is_data_disaggr	🟩	🟩	🟩	🟩	🟩	🟩
	n_data_points_disaggr	🟩	🟩	🟩	🟩	🟩	🟩
	an_analysis_type	🟩	N/A	N/A	🟩	🟩	N/A
	an_model_type	🟩	N/A	N/A	🟩	🟩	N/A
	em_analysis_type	N/A	N/A	N/A	🟩	N/A	🟩
	jd_analysis_type	N/A	N/A	🟩	N/A	🟩	N/A
	is_fit_good	🟩	🟩	🟩	🟩	🟩	🟩
	fit_ref	🟩	🟩	🟩	🟩	🟩	🟩
	is_validation	🟩	🟩	🟩	🟩	🟩	🟩
	val_data_source	🟩	🟩	🟩	🟩	🟩	🟩
	is_existing_val_study	🟩	🟩	🟩	🟩	🟩	🟩
	val_study_reference	🟩	🟩	🟩	🟩	🟩	🟩
	sample	🟩	🟩	🟩	🟩	🟩	🟩

f_scoring	geographic_relevance_score	🟩	🟩	🟩	🟩	🟩	🟩
	geo_applicability	🟩	🟩	🟩	🟩	🟩	🟩

Table	**Fields**	**Vulnerability Functions**					
		Empirical	**Analytical**	**Judgment**	**Hybrid A/E**	**Hybrid A/J**	**Hybrid E/**
f_core	id	🟩	🟩	🟩	🟩	🟩	🟩
	hazard_type_primary	🟩	🟩	🟩	🟩	🟩	🟩
	hazard_type_secondary	▨	▨	▨	▨	▨	▨
	process_type_primary	🟩	🟩	🟩	🟩	🟩	🟩
	process_type_secondary	▨	▨	▨	▨	▨	▨
	occupancy	🟩	🟩	🟩	🟩	🟩	🟩
	taxonomy_source	▨	▨	▨	▨	▨	▨
	taxonomy	▨	▨	▨	▨	▨	▨
	asset_type	🟩	🟩	🟩	🟩	🟩	🟩
	asset_notes	▨	▨	▨	▨	▨	▨
	country_iso	🟩	🟩	🟩	🟩	🟩	🟩
	applicability_notes	▨	▨	▨	▨	▨	▨
	scale_applicability	🟩	🟩	🟩	🟩	🟩	🟩
	function_type (ff/vf/dtl)	🟩	🟩	🟩	🟩	🟩	🟩
	approach (8subtypes)	🟩	🟩	🟩	🟩	🟩	🟩
	f_relationship (math/disc)	🟩	🟩	🟩	🟩	🟩	🟩
	f_math (par/bespoke)	N/A to discrete functions					
	f_math_model	N/A to discrete or Mathematical bespoke functi					
	bespoke_model_ref	N/A to discrete and to Mathematical NON besp					
	f_reference	🟩	🟩	🟩	🟩	🟩	🟩
	licence	🟩	🟩	🟩	🟩	🟩	🟩
	licence_reference	▨	▨	▨	▨	▨	▨
	contributed_at (timestamp)	🟩	🟩	🟩	🟩	🟩	🟩
	project	🟩	🟩	🟩	🟩	🟩	🟩
	purpose	🟩	🟩	🟩	🟩	🟩	🟩
	notes	🟩	🟩	🟩	🟩	🟩	🟩

Table	**Fields**	**Vulnerability Functions**					
		Empirical	**Analytical**	**Judgment**	**Hybrid A/E**	**Hybrid A/J**	**Hybrid E/**
f_specifics	f_specifics_id	🟩	🟩	🟩	🟩	🟩	🟩
	par_names	N/A to discrete					
	ub_par_value	N/A to discrete					
	ub_par_perc	N/A to discrete					
	med_par_value	N/A to discrete					
	lb_par_value	N/A to discrete					
	lb_par_perc	N/A to discrete					
	damage_scale_code	N/A					
	dm_state_name	N/A					
	n_dm_states	N/A					
	f_disc_im	N/A to Mathematical functions					
	f_disc_ep	N/A to Mathematical functions					
	lp_code	🟩	🟩	🟩	🟩	🟩	🟩
	lp_code_value	🟩	🟩	🟩	🟩	🟩	🟩
	edp_code	▨	▨	▨	▨	▨	▨
	edp_name	▨	▨	▨	▨	▨	▨
	edp_dmstate_thre	▨	▨	▨	▨	▨	▨
	im_code	🟩	🟩	🟩	🟩	🟩	🟩
	im_name	🟩	🟩	🟩	🟩	🟩	🟩
	im_units	🟩	🟩	🟩	🟩	🟩	🟩
	im_range	🟩	🟩	🟩	🟩	🟩	🟩
	im_method	🟩	🟩	🟩	🟩	🟩	🟩
	im_sim_type	🟩	🟩	🟩	🟩	🟩	🟩
	impe_reference	🟩	🟩	🟩	🟩	🟩	🟩
	data_countries	🟩	🟩	🟩	🟩	🟩	🟩
	im_data_source	🟩	🟩	🟩	🟩	🟩	🟩
	n_events	🟩	🟩	🟩	🟩	🟩	🟩
	n_assets	🟩	🟩	🟩	🟩	🟩	🟩

Table	Fields	Vulnerability Functions					
		Empirical	Analytical	Judgment	Hybrid A/E	Hybrid A/J	Hybr E/J
f_additional	f_additional_id	🟩	🟩	🟩	🟩	🟩	🟩
	nonsampling_err	🟩	🟩	🟩	🟩	🟩	🟩
	type_nonsampling_err	🟩	🟩	🟩	🟩	🟩	🟩
	is_fix_nonsamp_err	🟩	🟩	🟩	🟩	🟩	🟩
	is_data_aggregated	🟩	🟩	🟩	🟩	🟩	🟩
	n_data_points_aggr	🟩	🟩	🟩	🟩	🟩	🟩
	is_data_disaggr	🟩	🟩	🟩	🟩	🟩	🟩
	n_data_points_disaggr	🟩	🟩	🟩	🟩	🟩	🟩
	an_analysis_type	🟩	N/A	N/A	🟩	🟩	N/A
	an_model_type	🟩	N/A	N/A	🟩	🟩	N/A
	em_analysis_type	N/A	N/A	N/A	🟩	N/A	🟩
	jd_analysis_type	N/A	N/A	🟩	N/A	🟩	N/A
	is_fit_good	🟩	🟩	🟩	🟩	🟩	🟩
	fit_ref	🟩	🟩	🟩	🟩	🟩	🟩
	is_validation	🟩	🟩	🟩	🟩	🟩	🟩
	val_data_source	🟩	🟩	🟩	🟩	🟩	🟩
	is_existing_val_study	🟩	🟩	🟩	🟩	🟩	🟩
	val_study_reference	🟩	🟩	🟩	🟩	🟩	🟩
	sample	🟩	🟩	🟩	🟩	🟩	🟩

f_scoring	geographic_relevance _score	🟩	🟩	🟩	🟩	🟩	🟩
	geo_applicability	🟩	🟩	🟩	🟩	🟩	🟩

Table	Fields	Damage-to-Loss				
		Empirical	**Analytical**	**Judgment**	**Hybrid A/E**	**Hybr**
f_core	id	🟩	🟩	🟩	🟩	🟩
	hazard_type_primary	🟩	🟩	🟩	🟩	🟩
	hazard_type_secondary	▨	▨	▨	▨	▨
	process_type_primary	🟩	🟩	🟩	🟩	🟩
	process_type_secondary	▨	▨	▨	▨	▨
	occupancy	🟩	🟩	🟩	🟩	🟩
	taxonomy_source	▨	▨	▨	▨	▨
	taxonomy	▨	▨	▨	▨	▨
	asset_type	🟩	🟩	🟩	🟩	🟩
	asset_notes	▨	▨	▨	▨	▨
	country_iso	🟩	🟩	🟩	🟩	🟩
	applicability_notes	▨	▨	▨	▨	▨
	scale_applicability	🟩	🟩	🟩	🟩	🟩
	function_type (ff/vf/dtl)	🟩	🟩	🟩	🟩	🟩
	approach (8subtypes)	🟩	🟩	🟩	🟩	🟩
	f_relationship (math/disc)	🟩	🟩	🟩	🟩	🟩
	f_math (par/bespoke)	N/A to discrete functions				
	f_math_model	N/A to discrete or Mathematical bespoke fun				
	bespoke_model_ref	N/A to discrete and to Mathematical NON be				
	f_reference	🟩	🟩	🟩	🟩	🟩
	licence	🟩	🟩	🟩	🟩	🟩
	licence_reference	▨	▨	▨	▨	▨
	contributed_at (timestamp)	🟩	🟩	🟩	🟩	🟩
	project	🟩	🟩	🟩	🟩	🟩
	purpose	🟩	🟩	🟩	🟩	🟩
	notes	🟩	🟩	🟩	🟩	🟩

Table	**Fields**				**Damage-to-Loss**	
		Empirical	**Analytical**	**Judgment**	**Hybrid A/E**	**Hybrid A/J**
f_specifics	f_specifics_id	🟩	🟩	🟩	🟩	🟩
	par_names				N/A to discrete	
	ub_par_value				N/A to discrete	
	ub_par_perc				N/A to discrete	
	med_par_value				N/A to discrete	
	lb_par_value				N/A to discrete	
	lb_par_perc				N/A to discrete	
	damage_scale_code	🟩	🟩	🟩	🟩	🟩
	dm_state_name	🟩	🟩	🟩	🟩	🟩
	n_dm_states	🟩	🟩	🟩	🟩	🟩
	f_disc_im				N/A	
	f_disc_ep				N/A	
	lp_code	🟩	🟩	🟩	🟩	🟩
	lp_code_value	🟩	🟩	🟩	🟩	🟩
	edp_code				N/A	
	edp_name				N/A	
	edp_dmstate_thre				N/A	
	im_code				N/A	
	im_name				N/A	
	im_units				N/A	
	im_range				N/A	
	im_method				N/A	
	im_sim_type				N/A	
	impe_reference				N/A	
	data_countries				N/A	
	im_data_source				N/A	
	n_events	🟩	N/A	N/A	🟩	N/A
	n_assets	🟩	N/A	N/A	🟩	N/A

Table	**Fields**	**Damage-to-Loss**				
		Empirical	**Analytical**	**Judgment**	**Hybrid A/E**	**Hybri**
f_additional	f_additional_id	🟩	🟩	🟩	🟩	🟩
	nonsampling_err	🟩	🟩	🟩	🟩	🟩
	type_nonsampling_err	🟩	🟩	🟩	🟩	🟩
	is_fix_nonsamp_err	🟩	🟩	🟩	🟩	🟩
	is_data_aggregated	🟩	🟩	🟩	🟩	🟩
	n_data_points_aggr	🟩	🟩	🟩	🟩	🟩
	is_data_disaggr	🟩	🟩	🟩	🟩	🟩
	n_data_points_disaggr	🟩	🟩	🟩	🟩	🟩
	an_analysis_type	🟩	N/A	N/A	🟩	🟩
	an_model_type	🟩	N/A	N/A	🟩	🟩
	em_analysis_type	N/A	N/A	N/A	🟩	N
	jd_analysis_type	N/A	N/A	🟩	N/A	🟩
	is_fit_good	🟩	🟩	🟩	🟩	🟩
	fit_ref	🟩	🟩	🟩	🟩	🟩
	is_validation	🟩	🟩	🟩	🟩	🟩
	val_data_source	🟩	🟩	🟩	🟩	🟩
	is_existing_val_study	🟩	🟩	🟩	🟩	🟩
	val_study_reference	🟩	🟩	🟩	🟩	🟩
	sample	🟩	🟩	🟩	🟩	🟩

f_scoring	geographic_relevance _score				N/A	
	geo_applicability				N/A	