International Journal of Geo-Information

MDPI

Article

A Spatio-Temporal Building Exposure Database and Information Life-Cycle Management Solution

Marc Wieland [1,2,*] and Massimiliano Pittore [1]

[1] Helmholtz Centre Potsdam GFZ German Research Centre for Geosciences, Centre for Early Warning Systems, Telegrafenberg, Potsdam D-14473, Germany; pittore@gfz-potsdam.de

[2] Department of Zoology, University of Oxford, Tinbergen Building, South Parks Road, Oxford OX1 3PS, UK

* Correspondence: marc.wieland@zoo.ox.ac.uk

Academic Editor: Wolfgang Kainz

Received: 30 January 2017; Accepted: 5 April 2017; Published: 8 April 2017

Abstract: With an ever-increasing volume and complexity of data collected from a variety of sources, the efficient management of geospatial information becomes a key topic in disaster risk management. For example, the representation of assets exposed to natural disasters is subjected to changes throughout the different phases of risk management reaching from pre-disaster mitigation to the response after an event and the long-term recovery of affected assets. Spatio-temporal changes need to be integrated into a sound conceptual and technological framework able to deal with data coming from different sources, at varying scales, and changing in space and time. Especially managing the information life-cycle, the integration of heterogeneous information and the distributed versioning and release of geospatial information are important topics that need to become essential parts of modern exposure modelling solutions. The main purpose of this study is to provide a conceptual and technological framework to tackle the requirements implied by disaster risk management for describing exposed assets in space and time. An information life-cycle management solution is proposed, based on a relational spatio-temporal database model coupled with Git and GeoGig repositories for distributed versioning. Two application scenarios focusing on the modelling of residential building stocks are presented to show the capabilities of the implemented solution. A prototype database model is shared on GitHub along with the necessary scenario data.

Keywords: database; life-cycle management; exposure; disaster risk management

1. Introduction

Natural hazards impose a threat to millions of people all over the world. Yet, if hazards can be so different, the exposed assets are mostly the same; people, buildings, infrastructure, and the natural environment. Despite the fact that exposure is often treated as a static entity, it should be regarded as a dynamic process in which exposed assets undergo continuous changes [1]. This is particularly relevant for exposed building stocks. Considering the disaster risk management cycle, exposed assets are subject to changes throughout the different phases of risk management, reaching from pre-disaster mitigation to the immediate response after an event and the long-term recovery of affected assets. Such dynamics typically appear on different spatial and temporal scales. It can, for example, be linked to continuous changes such as urbanization [2] or to changes of the residential building quality through retrofitting or other risk-reducing measures [3] as part of the mitigation phase. Damage and collapse of exposed buildings are typical abrupt changes that appear in consequence of a disastrous event [4]. Construction of temporary shelters or reconstruction of buildings are other examples of rapid and continuous exposure dynamics that are typically linked to the short- and long-term post-disaster phases [5].

With respect to the collection of exposure and/or damage data at a per-building scale, typical methods involve in-situ screening surveys [6]. Often aggregated census data are used for regional to national scale risk studies [7]. Lately, an increasing demand for remote sensing studies at these scales can be observed, with a shift going from simple (single timestamp) mapping to (multi-temporal) monitoring of exposed building stocks and damages from satellite images [8]. Due to the increasingly large variety of possible exposure and/or damage data sources and their inherent spatio-temporal nature, database models in support of disaster risk management need to consider aspects of data harmonization and integration and be able to store and efficiently manage data coming from different sources, at varying scales, with different quality, and changing over time and space. The crucial point of data storage and management is, however, rarely mentioned in the context of disaster risk management, despite the fact that it is becoming increasingly important with the increase in data volume and complexity. Pittore et al. (2016) [9] provide an overview of the challenges of implementing an exposure model for different geo-hazards at a global scale within a dynamic and scalable framework. Specific points that need to be jointly addressed by a database model in the context of disaster risk management include spatio-temporal data modelling, the use of standard taxonomies, multi-representation, information life-cycle management, and distributed version control [10], each of which can be considered separate research domains. A large body of literature exists on the different topics (see Section 2); however, as yet, no publicly available data management technology exists, to best of the authors' knowledge that matches these requirements in a single solution with particular focus on disaster risk management.

The objective of this work is, therefore, to provide a conceptual and technical solution to store and manage heterogeneous building exposure data in a dynamic and spatially referenced manner throughout their life cycle. The proposed solution is implemented on the basis of free and open source software components. It combines best practices from different research domains into a single solution that specifically addresses the data management needs throughout different phases of the disaster risk management cycle.

The paper is structured as follows. Section 2 provides an overview of existing work in the different research domains that are touched by this study. Choices made and advances achieved in order to support disaster risk management efforts are addressed. Section 3 introduces the database model. Sections 4 and 5 describe the information life-cycle management solution and distributed version control. Section 6 builds up on the concepts and solutions presented in the previous sections and describes a realistic scenario related to building exposure data management during different phases of an earthquake risk management cycle. Sections 7 and 8 close the paper with discussion and conclusions.

2. Exposure Data Management in Support of Disaster Risk Management

This section provides an overview of the most relevant work in the different research domains tackled by this study and identifies the choices that were being made in order to implement a prototype database and information life-cycle management solution.

2.1. Spatio-Temporal Databases

Spatio-temporal databases are characterized by both spatial and temporal semantics and deal with real-world applications in which spatial changes occur over time. A data model must, therefore, allow for spatial, temporal, and spatio-temporal queries to be performed. Changes over time can apply to the geometry, topology, or attribute characteristics of an object, leading to eight possible change type scenarios [11], as depicted in Figure 1. Earlier work in the direction of spatio-temporal database design was characterized by separate research in either the spatial [12] or temporal [13] domain. Since the emerging of combined spatio-temporal databases, numerous approaches have been proposed for data modeling, and reviews have categorized and compared the existing work. Langran (1992) [14] was one of the first authors who specifically addressed the temporal domain in Geographic Information

Systems (GIS). Different approaches to spatio-temporal data modelling have been proposed since then, including object-oriented data modelling [15] or cell tuple-based data models, as proposed for example in [16]. Also relational models have been proposed to handle spatio-temporal data such as in [17,18]. Comprehensive literature reviews on the topic can be found, amongst others, in [19–21]. Relational database models are generally considered to be less efficient than object-oriented approaches when dealing with very large data volumes and semi-structured data. However, due to its simplicity, its ability to deal with transactions, its extensive join capabilities, and the fact that it is a well-known and standard data modelling approach, a relational approach is followed in this study.

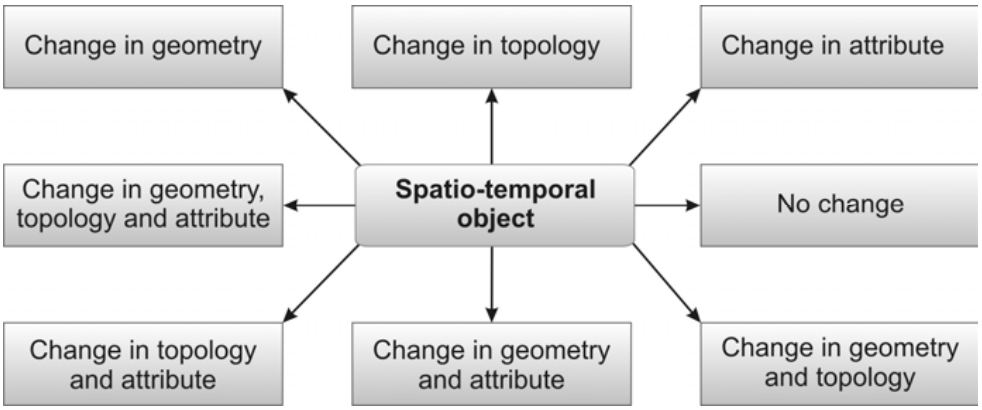


Figure 1. Possible types of changes of a spatio-temporal object. Modified after [11].

In spatio-temporal databases, different time representations are possible and the decision of what to timestamp or on which level may vary. Timestamping an entire geographical object (e.g., a building as a whole), for example, may be a valuable choice if storage capacity is limited, but it does not allow for a description of the temporal variation of its individual attributes (e.g., number of storeys) over its lifespan. Timestamping at the object primitive level (records) takes more storage capacity but allows for a more detailed description of the temporal variability within objects with a finer granularity. In this context, bi-temporal data models that support transaction time (or registration time) and valid time (or real world time) are flexible and allow object identity to be represented in a way that is suitable for the envisaged application. When modelling changes are made to an exposed building stock, information updates (new information becomes available as part of database updates) need to be distinguishable from real-world changes (actual modifications to buildings) in order to identify and track the object identity, meaning the lifespan of an object. This becomes particularly important when crowd-sourced data acquisition is involved [22]. Therefore, a bi-temporal data model with timestamping at the most detailed object primitive level is selected. Changes to the building stock are recorded in discrete transactions, as is the case in cadastral land register systems.

2.2. Multi-Representation of Spatial Objects

A multi-representation database stores different views of the same physical objects or phenomena and links them with each other, which leads to differences in the semantics and in the geometry of the objects [23]. The main reasons for introducing a multi-representation into a data model include how such information can be analyzed with respect to information provided in another representation, such as another scale or resolution. Moreover, linking different representations allows for propagating updates between, for example, different map scales [24]. A data model that supports multiple representations of spatial objects can also be defined as a multi-resolution or multi-scale database.

Multi-resolution models mainly evolved in response to requirements imposed by web mapping applications that deal with auto-generalization between different zoom levels [25]. CityGML [26], for example, provides a comprehensive data model to represent 3D city models under consideration of

five different detail levels. Two general cases of multi-resolution datasets can be distinguished. First, datasets at different resolutions are generated independently and need to be linked in the database by matching procedures. Second, lower resolution representations are derived from higher resolution data by generalization functions and their link is established by the generalization process. In the context of disaster risk management, multi-resolution is of high relevance, mainly due to the fact that exposed assets and/or damages are often described by data from different sources and at different scales. In previous work of the authors [27], for example, point- and path-wise visual screenings of single buildings are used in combination with large area remote sensing observations to derive a multi-scale exposure model. Due to the fact that exposure models often combine datasets from different sources at different scales, particular emphasis was given to implement independent matching procedures.

Sester et al. (2004) [23] describe three possible structures for matching different resolutions in the database: attribute-based structure, additional attribute, and additional linking tables. In an attribute-based structure, everything is stored in one dataset and additional attributes are used to describe the different appearances. Another option is to link the datasets by using additional attributes that store only identifiers (IDs) that refer to the corresponding objects at the lower resolutions. One attribute per resolution level would be required for this option and the different resolution objects could be stored either within the same table or in different tables. The third option is based on the additional attribute structure but stores the resolution IDs in a separate table. To minimize the complexity of Structured Query Language (SQL) queries and the number of database accesses, the additional attribute linking structure was selected, and attributes are added for each resolution level in the same table.

2.3. Taxonomy

Given the complex elements composing the exposure of the built human environment (for instance, the structural components of a building), a clearly defined taxonomy is essential in order to overcome possible misunderstandings amongst data collectors and analysts, especially at transnational and global scales. The preparation of a taxonomy involves the creation of an exhaustive and structured set of mutually exclusive and well-described attributes, which can be structured hierarchically or faceted and encoded in numeric or alphabetical codes [28]. A taxonomy suitable for describing exposure at large-scale should be international in scope, detailed, collapsible, extendable, user friendly, and suitable to describe different types of exposed assets with enough detail to match the requirement of the related vulnerability models [29]. Moreover, ideally an exposure model could accommodate structural and non-structural features relevant for different natural hazards, therefore setting the base for an efficient, integrated, multi-risk exposure modelling. The harmonization of information between taxonomies is of particular importance in order to be able to combine or compare data that have been collected about different inventories, or over different time periods, and which may potentially have involved different data collection methods and/or spatial units. Harmonization can be achieved by applying consistent standards and taxonomies across different data sets and by establishing look-up tables mapping the homologous categories of different taxonomies.

Several taxonomies have been developed mainly in the field of earthquake engineering in order to classify and characterize building inventories in standardized and comparable ways. Widely used taxonomies that aim at describing the structural components of buildings include HAZUS [30], the European Macroseismic Scale 1998 (EMS-98) [31], Prompt Assessment of Global Earthquakes for Response (PAGER) [32], and the World Housing Encyclopedia [33]. A structural taxonomy is just one of several taxonomies that together contain all the relevant data about a particular exposed asset. Other (non-structural) taxonomies are used to encode further information, including non-structural features, occupancy, population, or economic values. A recently established structural taxonomy that also includes few non-structural attributes (e.g., occupancy) is the Global Earthquake Model (GEM) Building Taxonomy [29]. It follows the concept of a faceted taxonomy [34]. In comparison to hierarchical taxonomies, where the data are organized in a tree-like structure, the hierarchy of classes

in faceted taxonomies is not fixed and can be changed. Faceted taxonomies allow information to be navigated along multiple paths corresponding to different arrangements of the facets and items in more than one dimension. Due to its clear and flexible structure, its complexity and comprehensiveness and the fact that it may also be applicable to hazards other than earthquakes, the GEM Building Taxonomy Version 2.0 was used as the basis to structure the taxonomy schema of the proposed database model.

2.4. Information Life-Cycle Management

The concept of a life cycle implies that information elements (in the following referred to as objects) have a definable point of creation and a similar point of destruction or disposal. Life cycle development focuses on the change of the information itself and on the change of its value over time. Value transformation occurs when information is created, updated, or disposed. Information value is often defined in financial terms as the amount a decision maker would be willing to invest for information prior to making a decision. However, the types of information value can be of different natures, reaching from financial to operational to intrinsic. Chen (2005) [35], for example, presents an information valuation approach that quantifies the value of a given piece of information in an automated manner based on an analysis of its usage over time. In the context of building exposure data management, the information value is closely linked to information quality, wherein map and thematic accuracy are widely used proxies to value the information. Map accuracy can be distinguished into location and classification accuracy. Location accuracy answers the question if the objects that are mapped and the data sources themselves are located on the earth and to one another. Classification accuracy quantifies if a material on the earth or a condition is mapped correctly [36]. The thematic accuracy is less well constrained and can be approached from different directions. A possible way to quantify thematic accuracy is to assign a degree of belief, as is described in detail by Huber (2006) [37]. In the context of this study, the information value is associated with map accuracy for spatial attributes and with thematic accuracy (here defined as degree of belief) for thematic attributes.

In database models, the information life cycle indicates the history of objects that are stored in a database both in terms of their real-world evolution and in terms of the database transactions referring to them. This means that a database model should be able to keep track of and archive the real-world changes of an object (from creation to destruction). At the same time, it should be able to register any modifications to an object at database level by logging transactions that are linked to a specific object. In order to do this, a bi-temporal database model is implemented in this study as described above.

Information collected from different sources is, moreover, subject to uncertainties that not only vary between sources but also over time. Therefore, life-cycle management should be uncertainty aware. The integration and use of spatio-temporal information affected by uncertainty is a topic gaining attention since when Geographical Information Systems (GIS) started becoming widespread. For instance, Shi (2007) [38] describes an object-oriented meta-data error database for capturing quality information and temporal information, information about sources of data, processing steps, and cartographic generalizations on the datasets. Tu et al. (2006) [39] use hierarchical and hybrid Bayesian networks to integrate information among multiple agencies. Butenuth et al. (2007) [40] introduce integration algorithms for vector and raster data sets in federated databases. In previous work, the authors used Bayesian networks to integrate proxy information extracted from different image sources to derive posterior probability distributions of structural vulnerability classes for buildings [41].

2.5. Distributed Version Control

With respect to non-distributed version control systems, where clients have to connect to a central server to coordinate the versioning, a distributed version control system takes a peer-to-peer approach. In a distributed version control system, each client's copy of a repository is a repository on its own, able to synchronize with any other peer to exchange patches and further software development [10]. Recently, a new class of distributed version control systems, which have evolved in the domain of software development, can be considered mature enough to be used for handling

geospatial information. The most popular and widely used among these systems is Git, which was originally developed by Linus Torwald for managing Linux source-code. With the focus on source-code versioning, it can only be applied to version information that is stored in text-based formats such as GeoJSON or Comma Separated Value (CSV) files. GeoGig is a system that builds up on the general principals of Git and explicitly supports the versioning of geospatial information. GeoGig is developed and maintained by Boundless on a free and open-source basis. The founders of GeoGig support a shift away from treating geospatial information as strict data (stores of rows and columns optimized to be sliced in different ways, copied extensively, and occasionally updated), towards treating it as software developers treat source code; as a material living in collaborative infrastructure that constantly tracks its origin and evolution, even when copied and edited by disparate users [10]. With distributed version control systems like Git or GeoGig major problems that have affected users of geospatial information can be addressed, including multi-organization/multi-user collaboration, crowd-sourced vs. authoritative data collection, or meta-data sharing and updating. The capabilities of such distributed version control systems in disaster risk management have recently been demonstrated during Typhoon Yolanda where crowd-sourced damage mapping activities have been version controlled by GeoGig [10]. In this study, we use both the Git and GeoGig workflows to test the distributed version control of database release states.

3. Database Model

This section describes the spatio-temporal database model that has been designed to efficiently store and manage data represented at different spatial scales and changing over time, while being able to keep track of the source and quality of the single attributes. The implementation of the database model is based on the free and open-source relational database management system PostgreSQL. It supports most SQL constructs, including sub selects, transactions, and user-defined types and functions as well as many standard data types, including date/time types. For spatial functions, the PostGIS extension is applied. It adds support for geographic objects to the PostgreSQL database. PostGIS follows the OpenGIS Simple Features Specification for SQL, with version >2.0 supporting both vector and raster objects and any related spatial query capabilities. Also, topological models are supported to handle objects with shared boundaries. To deal with the multi-representation of spatial objects, a bottom-up approach is followed in which datasets of different resolutions are linked by using additional attributes, which identify the corresponding objects in the lower levels of detail. A bi-temporal representation of time has been implemented, where a trigger function allows database transactions to be logged along with the transaction time and optional additional information for user-defined tables and attributes. The temporal support has been implemented on the spatial object level to allow for a more detailed description of the temporal variability within objects with a finer granularity.

The database model is structured into three schemas. The object schema (Figure 2 left) contains the main description of the database objects, their spatial reference, and representation at different resolutions. In addition to the spatial properties, the object details are characterized by attribute and qualifier values, which are defined within the taxonomy schema (Figure 2 right). The history schema (Figure 3) holds the general structure for logging database transactions and provides, therefore, the transaction time component of the bi-temporal data model. In the following, the taxonomy implementation, the multi-resolution spatial support, and the multi-temporal extension are explained in further detail.

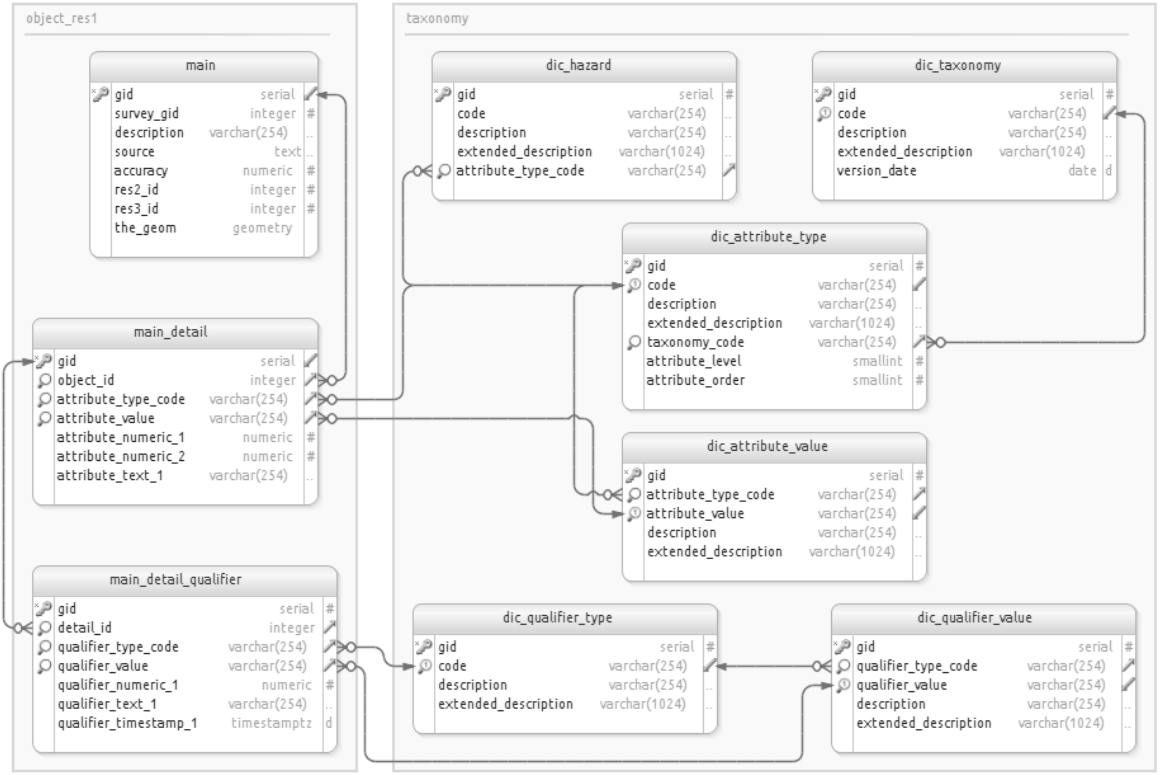


Figure 2. Database model prototype, with object schema (**left**) and taxonomy schema (**right**).

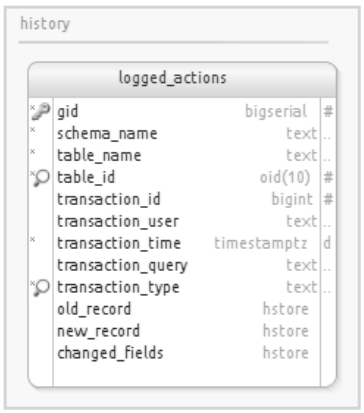


Figure 3. History schema.

3.1. Taxonomy

The GEM Building Taxonomy Version 2.0 was used as basis to structure the taxonomy schema of the current database model prototype. It was attempted to design the database schema structure in a way to keep it flexible and open to any other taxonomies. Taxonomy attributes are organized in a dictionary table (*taxonomy.dic_attribute_type*), where each attribute type is assigned a unique ID in alphanumeric format and linked to a textual description. Categorical attribute values (*taxonomy.dic_attribute_value*) are stored in a separate dictionary table and are also defined by unique IDs and linked with a textual description. Numeric or textual attribute values are inserted for the single object primitives directly in the table *object.main_detail*. Each attribute type needs to be associated with one or more hazard types (*taxonomy.hazard*) and be linked to a specific taxonomy (*taxonomy.dic_taxonomy*) in order to clearly categorize the attributes and to identify their source and application.

In addition to attribute types and values, qualifiers are defined within the taxonomy schema of the database prototype. Qualifiers in the context of this work refer to additional descriptors of attributes that are independent of the type of hazard and the application. Potential qualifiers to be supported could include accuracy, precision, quality, or valid time. Qualifier types (*taxonomy.dic_qualifier_type*) are defined by unique IDs and linked with a textual description. The specific states of the qualifier types are given by values that are stored in a separate dictionary table (*taxonomy.dic_qualifier_value*) in case of categorical variables. Numeric or textual qualifier values are inserted for the single object primitives directly in the table *object.main_detail_qualifier*.

3.2. Multi-Resolution Spatial Support

Comprehensive spatial support to the database is provided by the PostGIS extension to PostgreSQL. A particular requirement for the database, however, was the support of multiple object representations. Due to the focus on geographical objects derived from imaging techniques and mapping approaches, multi-representation can be seen in terms of multi-resolution or multi-scale datasets, where different generalization levels of the objects are provided. However, also multiple representations of datasets at the same resolution are possible when, for example, data from different sources are integrated (e.g., building footprints from cadastral maps together with footprints extracted by satellite image analysis).

In the current implementation of the database, a bottom-up approach is followed where datasets of different resolution are linked by additional attributes, which refer to the corresponding objects in the lower levels of detail. A trigger function *object.update_resolution_ids()* has been implemented to identify for each object at each resolution level its corresponding representations at the other resolution levels and to update the resolution IDs based on the spatial relationship between the geometries. More specifically, the function identifies for each object with a given polygon geometry type a point inside the polygon that is closest to its centroid and, based on this, performs a spatial join with the polygon geometries at the other resolution levels to obtain their corresponding IDs. The function is called from a row level after the INSERT or UPDATE trigger of the geometry column (*the_geom*) and, therefore, updates the IDs on the fly whenever the geometry of any record at any resolution level is modified or a new record is inserted. A row level after DELETE trigger handles the ID updates for record deletes. The resolution ID updates are only performed for records that are actually affected by the INSERT, UPDATE, or DELETE triggers. This is done to avoid affecting negatively the database performance by having to re-compute ID relations for the whole dataset.

In its current implementation, the data model supports three resolution levels, based on the assumption that the database will be populated mostly with information derived from census reports, satellite image analysis, and in-situ screening techniques. The data model can be easily extended to other numbers of resolution levels, and level definitions are not fixed or bound to specific conditions but can be customized depending on the specific application.

In order to derive new (lower resolution) data sets from existing (higher resolution) ones, a generalization function needs to be defined to describe this action. The choice of the generalization function depends upon the type of objects to be created and the desired resolution level. For example, a generalization function that derives aggregated building blocks from higher resolution building footprints may be different from a function that creates a built-up area footprint from a set of building blocks. To exemplify the capabilities of the data model, a simple prototype generalization function (*simplify_buildings()*) has been implemented for the geometrical simplification of building footprints to aggregated building blocks.

3.3. Multi-Temporal Support

A bi-temporal representation of time has been implemented on the primitive spatial object level to allow for a detailed description of the temporal variability within objects. Valid time is considered an object qualifier and is therefore specified for each record by qualifier timestamps. Valid time refers

to real world timestamps, and, therefore, unlike the transaction time, it needs to be set for each record by the user or by the application that inserts or updates the records as a consequence of a real world change. Therefore, even continuous real world changes are detected in discrete steps in the database. This is, however, in line with the considered data acquisition methods for which the database is intended. In-situ surveys or remote sensing analysis for example depend on (stepwise) acquisition times of the surveys or the satellite images acquired over an area of interest. Satellites have specific revisit periods or temporal resolutions, which depend upon their orbit, sensor specifications, and number of satellites (in the case of a satellite constellation), which in practice define the temporal granularity. Similarly, in-situ surveys are usually carried out iteratively at discrete time steps.

In order to keep track of the history of the real world objects, a live-history approach is followed. This means that deleted or changed records are archived in the history schema (*history.logged_actions*) along with additional information about the corresponding database transaction. A generic trigger function *history.if_modified_func()* archives the records and logs the transaction times and additional information about changes to selected tables and attributes. The function is based on the PostgreSQL Audit Trigger and was modified according to the requirements of this work. The logging of transactions can be done at a statement level or at a row level. Control is logged separately for each table, and the data columns to be logged can be specified individually. Row values are recorded as *hstore* fields instead of *text*, which allows for more sophisticated queries on the history and reduces query complexity and storage space.

The combination of a bi-temporal data model with a live-history approach allows for a straightforward recovery of former states of the database at defined transaction or valid times and for sophisticated temporal queries. Two functions have been implemented that allow views of the transaction and valid time history, which can be easily queried for spatial, attribute, valid, and transaction time components, to be recovered.

- *ttime_gethistory()*: This function selects from *history.logged_actions* all records that have been modified in the logged tables. It provides the transaction time history.
- *vtime_gethistory()*: This function selects from *history.logged_actions* all records that have been subject to a real world change. For each record, the latest version at each valid time is selected. It provides the valid time history.

Examples that use transaction and valid time history functions are given in Section 6.2.

4. Information Life-Cycle Management Solution

Figure 4 shows a schema of the information life cycle exemplified within this work and implemented at database level in PostgreSQL. New information enters the database workflow in the creation phase. Along with the actual object to be added to the database, several meta-data attributes need to be defined. Among these are the source of the information item and its value. The use of data entry forms for user driven information creation can enforce these additional meta-data attributes. Once new information is created, it is integrated with already existing information. In the case of geospatial information, it is checked if geoinformation already exists in the database at the same location of the newly created item. A spatial overlap between new and old items forces their values to be compared, and the information item with the lower value is replaced with the higher value item. The overall values of the database are then updated accordingly, and the information is ready for usage for further analysis and interpretation by the user. Several customized spatio-temporal query functions and views are implemented in order to improve information accessibility during the usage phase of the life-cycle model. The database model supports transaction logging and record archival. Once information is modified or disposed, its previous state can be archived and the database transaction can be logged. In case the modification refers to an update with information with higher value (e.g., better information became available), the archiving mechanism allows previous database states to be retrieved or recovered at any given transaction time. In case of the modification or disposal

referring to real-world changes, this mechanism allows the full history of an object from its creation (e.g., construction of a building), through modifications (e.g., retrofitting of a building) to its disposal (e.g., destruction of a building) to be archived. The different parts of the database workflow are described in greater detail in the following paragraphs.

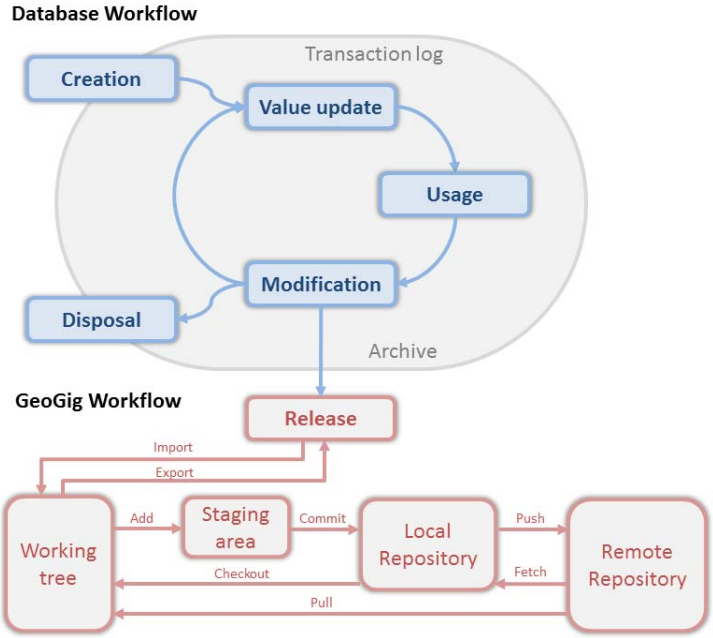


Figure 4. Schematic information life cycle depicting the information flow as supported by the database model. The model is coupled with Git and GeoGig for the versioned release of database states.

The content of the database can be released and versioned at any time in order to distribute and share it in a decentralized way. Information to be released can be the whole database or subsets of it in the form of summary views. Once information is released, it enters a new workflow. For releases and distributed versioning, we currently use Git [42] and GeoGig [10]. A possible release workflow is schematized in Figure 4 and is further discussed in Section 5.

4.1. Information Creation

Information creation includes discovery, accumulation, and aggregation. It is the point where information is captured and enters the managed environment. The issues governing information creation include data entry, acquisition standards, selection of attributes and taxonomies, and documentation (e.g., meta-data creation).

4.2. Information Usage

At this stage of the life cycle, information is actively or semi-actively in use. It is assumed to be current and the most useful and valuable (accurate) to the application. Issues governing information usage include accessibility, transformations and the effectiveness of delivery (query functionality), and distribution (speed, mapping, etc.). Views are used in the database model to interact with the database content, which natively is stored in distributed schemas and tables. An editable view is used to access a comprehensive description of the content and to insert, update, and delete records. The modifications to the view are distributed to the relevant underlying tables. Read-only views are further generated to create a meta-data summary of the current state of the database content and a simplified data view that is readily formatted for mapping and release.

4.3. Information Modification and Value Updating

In the database model, the information is described at the attribute level using qualifiers. In principle, different kinds of value descriptors are possible, depending on the user-driven definition of the information value. This could be costs, accuracy, and degree of belief; time; or other application specific values. In the context of this work, information value is associated with quality in the form of map accuracy for geometrical information and degree of belief for thematic information.

Table 1 shows an excerpt of a dynamically created meta-data summary view. The view summarizes the currently available attributes, their sources, and their average value (here average degree of belief in percent). Whenever information is updated in the database, its value is also updated, and, accordingly, the average value of the overall information content of the database is adjusted. The value of information can be used as a proxy to guide the information integration, as proposed, for example, by Tu et al. (2006) [39] or by previous work of the authors, in which Bayesian networks are used for automated information integration and the probabilistic vulnerability estimation of structures [41]. In its current implementation, the database model provides all the necessary attributes and updating mechanisms to set up automated integration functions under consideration of uncertainties.

Table 1. Excerpt of a dynamically created meta-data summary table that summarizes the currently available attributes, their sources, and the evolution of their average value (here degree of belief in percent) from release R1 to R3.

Attribute Type	Description	Source	Value Type	Avg Value R1	Avg Value R2	Avg Value R3
HEIGHT	Height	RRVS	BP	0.00	0.00	47.52
LLRS	Lateral Load-Resisting System	RRVS	BP	0.00	0.00	46.88
MAT_PROP	Material Property	RRVS	BP	0.00	0.00	45.58
MAT_TECH	Material Technology	RRVS	BP	0.00	0.00	55.29
MAT_TYPE	Material Type	RRVS	BP	0.00	0.00	52.82
OCCUPY	Building Occupancy Class	RRVS	BP	0.00	0.00	46.76
OCCUPY_DT	Building Occupancy Class—Detail	RRVS	BP	0.00	0.00	49.35
THE_GEOM	Object geometry	EO, OSM, OF	BP	73.00	84.04	86.91
YR_BUILT	Date of Construction or Retrofit	RRVS	BP	0.00	0.00	0.00

4.4. Information Disposal and Archiving

In the case of geospatial information management, information about a geographic object can be disposed and archived because (1) it is replaced with information of higher value (e.g., accuracy) or (2) the object that the information is referring to (e.g., building) has been modified or destructed in reality. In both cases, the obsolete information is disposed and archived for possible later retrieval. The difference between the two cases being that in case (1) the archived information refers to the database transaction life-cycle of the object, whereas in case (2) the archived information relates to the real-world life-cycle of the object.

Depending on the aim of the information system, the type and duration of archiving disposed information may vary. Archived information relating to the real-world changes of an object is an essential part of the life cycle of the actual object under observation. It is not disposed because the information lost value but because the object changed or lost its existence in the real world. The information referring to this object may therefore still be of great importance for further analysis even after its disposal. On the contrary, archived information that was disposed because better information became available has, by definition, less value than the new information. Therefore, the decision on whether to archive the disposed information or not and, in this case, for how long to archive it may differ significantly from the previous case.

In the database model, both cases of information disposal and archiving are supported. A specific PL/pgSQL (Procedural Language/PostgreSQL) function (history.history_table()) has been implemented in order to activate or deactivate the logging of database transactions and to archive

modifications and disposal of information. The function can be applied to specific tables or database views as a whole, or the transaction logging can be activated only for specific attributes of a table (e.g., log transactions only if attribute x is modified). If, for example, only the attributes related to the valid time of the object are activated for logging, the system would archive only the real-world change (case 2) and not the information updates (case 1).

5. Distributed Version Control and Release

The database model provides rich functionality to log database transactions, archive records, and query object histories in space and time. It provides a framework for information integration, management, and versioning in a non-distributed manner, as it only offers collaboration of multiple users around a single centralized database. In order to maintain its intrinsic benefits (e.g., quick and comprehensive queries across different states of the database and its records), while making it accessible and manageable in a distributed multi-user environment, the database model can be coupled with Git and GeoGig version control systems.

The schema of a typical versioning workflow is provided as part of Figure 4. The principal workflows and commands are similar between Git and GeoGig. The content is stored in a repository, which has three areas: the working tree, the staging area, and the local repository.

- The working tree is the area of the repository where the work is actually done on the data. Data in the working tree is not part of a defined version but instead can be edited and altered before turning it into a new version that will be safely stored.
- The staging area is an intermediate area where data is stored before moving it to the database.
- The local repository is where the history of the repository is stored, as well as all the versions that have been defined.
- The remote repository is a remote copy of a repository that allows for collaborative work between different people. In a collaborative environment, the remote repository holds the reference history. Users can clone it, work on the cloned repository locally, and push their changes back to the remote repository.

The process of versioning geospatial data consists of the following steps:

1. Importing data into the working tree so it can be managed by GeoGig (note that this step is additional with respect to a typical Git workflow).
2. Adding data from the working tree to the staging area.
3. Committing to the local repository.
4. Pushing local changes to the remote repository and/or fetching changes from the remote repository to the local repository in order to synchronize them. As new data are added to the repository database, new versions are created that define the history of the repository. While some versioning systems store the differences between consecutive versions, GeoGig stores the full set of objects that comprise each version. For instance, if the geometry of a feature has been modified, GeoGig will store the definition of that feature, which will be kept in the database along with the previous version of the same feature. For features not modified from one version to another, the corresponding objects are not stored again, and the new version points to the same previous object. So while each version is a new set of objects, the data for these objects are only stored once.

Where the database archiving functionality is designed to ease storage and querying of object histories and to provide a framework for information integration and updating in a non-distributed manner, Git and GeoGig are designed to ease collaboration among people working on the same data. The repository can accept changes from different people working on the same data, and the changes done by different users can be shared. Instead of a single repository, there can be a number of connected remote repositories, each of them working independently but communicating and interacting when needed.

6. Application Scenarios

In order to test the capabilities of the proposed database and life cycle management solution with a coupled Git/GeoGig workflow for distributed version control of releases, two application scenarios have been created.

6.1. Scenario I: Information Integration and Value Updating

The aim of the first scenario is to integrate three different data sources to a unified dataset that covers the whole Area of Interest (AOI) while improving the overall value for the final inventory. The location and geometries considered in the scenario are taken from different sources over a district of a German city (Figure 5). Geometries are iteratively inserted, updated, and deleted with the aim to simulate both real-world changes to the building objects (construction, modification, destruction) and information updates, as result of better information becoming available about existing objects during their life-time. Attribute values for updates are randomly chosen from a set of allowed values according to the GEM Building Taxonomy for randomly selected objects.

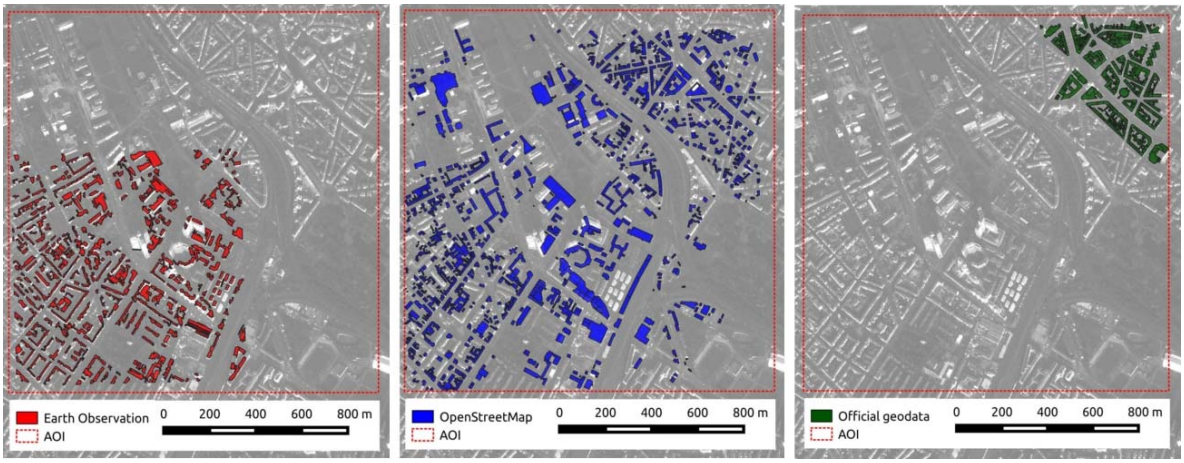


Figure 5. Scenario Area of Interest (AOI) with the different sources for creating a unified building inventory.

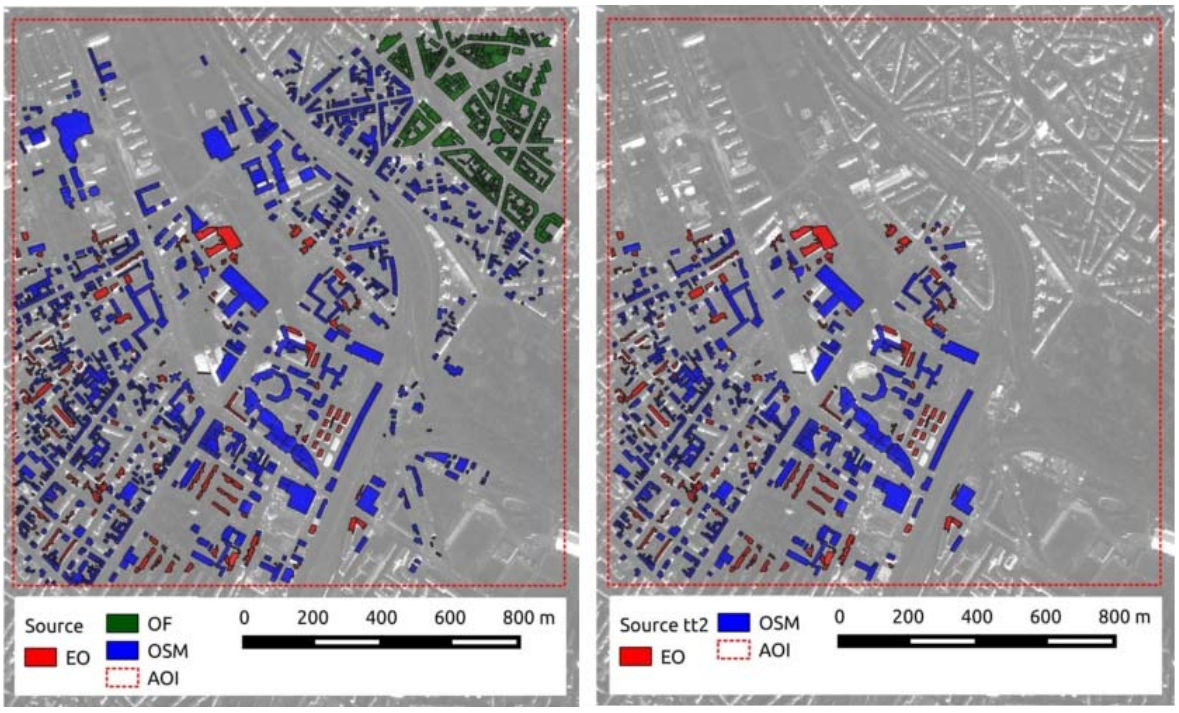
The data sources used for this application scenario are 422 building footprints extracted from Earth Observation (EO) data using remote sensing tools [43], 1539 buildings from OpenStreetMap (OSM), and cadastral data from an Official source (OF) that holds 541 buildings. The value of the sources (here defined by the map accuracy) increases from EO to OSM to OF.

During the integration phase, the database transactions shall be logged and changes shall be archived for possible later retrieval. At each critical stage of the integration, a release is issued via Git and GeoGig and pushed to a remote repository. The following database transactions were performed in order to integrate the available data:

- Transaction tt1: Insert object geometries from EO data analysis.
- Release R1: EO data.
- Transaction tt2: Update EO geometries with geometries from OSM where (ValueOSM > ValueEO) $\wedge$ (GeomOSM $\cap$ GeomEO).
- Transaction tt3: Insert OSM objects where not GeomOSM $\cap$ GeomEO.
- Release R2: EO and OSM data.
- Transaction tt4: Update attributes of 10 buildings with values following the standards for RRVS data entry.

- Transaction tt5: Update EO and OSM geometries (EOSM) with geometries from OF where (ValueOF > ValueEOSM) ∧ (GeomOF ∩ GeomEOSM).
- Transaction tt6: Insert OF objects where not GeomOF ∩ GeomEOSM.
- Realease R3: EO, OSM and OF data enriched with RRVS attributes.

Table 1 shows the evolution of the average value for selected attributes as the dataset evolves over time. Figure 6a shows the integrated building inventory at release 3 (R3), with colors indicating the different sources of the building geometries. Figure 6b shows the database state at transaction time 2 after the update of EO geometries with geometries from OSM. This state lies in between two releases (R1 and R2) and has been recovered using the transaction time history query (ttime_history()) of the database model.



(**a**) (**b**)

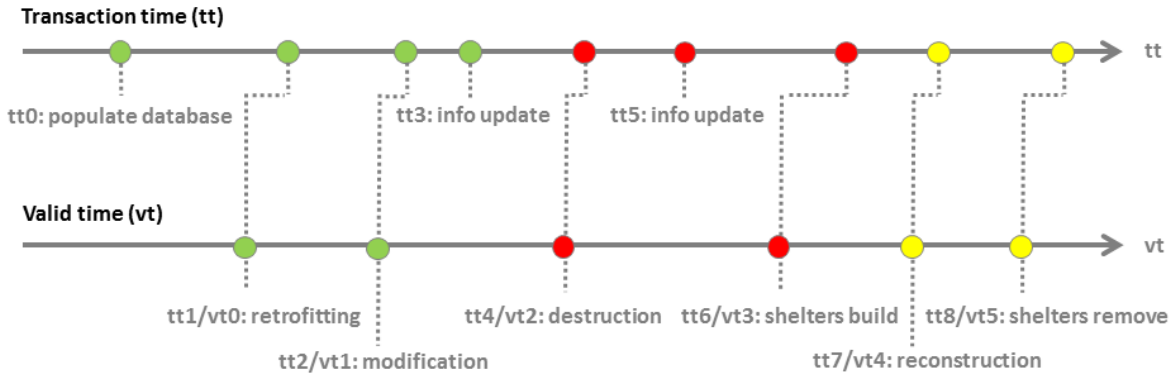
Figure 6. (**a**) Shows the integrated building inventory at release 3 (R3); (**b**) shows the database state at transaction time 2 after the update of Earth Observation (EO) geometries with geometries from OpenStreetMap (OSM).

6.2. Scenario II: Real-World Changes and Object Life-Cycle

In the second scenario, the building stock of a city is modelled in space and time throughout different phases of disaster risk management, from pre-event mitigation to post-event response and long-term recovery. Real world changes to the building stock (e.g., retrofit of buildings) and information updates (e.g., new information becomes available about existing buildings) are considered in the scenario. The GEM Building Taxonomy has been fully implemented into the database structure to characterize the building stock in a standardized and comparable way. Additional attributes relating to a building's structural vulnerability and damage grades were implemented based on the European Macro-seismic Scale 1998 (EMS-98) [31]. Data acquisition is simulated, exploiting different information sources at varying spatial resolution. The data that are used in this scenario have been imported from independent acquisition campaigns. The combination of building attributes and geometries does not represent the actual inventory of a real city but has been assembled to clearly exemplify several application scenarios of interest for risk management. The building footprints have been imported

from OpenStreetMap (OSM) by selecting 500 buildings from a German city. The building attributes were extracted from a database of buildings that were inspected by structural engineers by using the Remote Rapid Visual Screening (RRVS) procedure described in [27]. Further attributes were derived from census data.

Figure 7 depicts the timeline of events that are considered in the scenario. Both transaction (tt) and valid (vt) timelines are represented. At tt0, the database is populated with 500 building geometries from OSM and attribute values from RRVS at per-building resolution and with a zonation of the city from Earth Observation (EO) data at a neighborhood resolution. At tt1/vt0, a Rapid Visual Screening (RVS) survey [6] indicates 50 buildings that did not comply with the available building code and that were retrofitted to improve their seismic resistance. At tt2/vt1, another RVS survey highlights 20 buildings that were modified since the previous survey. The owners added an additional story to their house, which is not compliant with the existing building code for seismic resistant construction. At tt3, more information about the buildings becomes available from a newly released census report that, for the first time, includes information about the occupancy of buildings in the city. The information is used to update the database. At tt4/vt2, a major earthquake occurs in the region, which leads to structural failures and total collapses as well as a large number of damaged buildings in the city. As part of a first damage detection survey, only total collapses (125 buildings) are recorded and stored in the database. From a second damage survey, more information about the damage grades of 200 buildings relating to the same event (vt2) becomes available and is stored in the database at tt5. As response to the disaster, at tt6/vt3, 100 temporary shelters are constructed to host people who lost their homes as a consequence of the earthquake. As part of the long-term recovery, permanent housing starts and 95 new buildings are constructed at tt7/vt4. Finally, after all displaced people received support from the government and were able to settle down in permanent housing, the temporary shelters are removed at tt8/vt5. The information about the evolution of the recovery phase could be detected by change detection analysis of EO data and in-situ surveys, as is described in [44].



Time	Short description	Description	Data source	Phase
tt0	Populate database	Available data is added to the database	OSM, RRVS, EO	Mitigation
tt1/vt0	Retrofitting	50 buildings are retrofitted	RVS	Mitigation
tt2/vt1	Modification	20 buildings are modified by owners	RVS	Mitigation
tt3	Information update	Information about occupancy of the buildings becomes available	Census	Mitigation
tt4/vt2	Destruction	An earthquake occurs , buildings are damaged and destroyed	RVS (1st survey)	Response
tt5	Information update	Information about damage grades of buildings becomes available	RVS (2nd survey)	Response
tt6/vt3	Shelters build	Temporary shelters are constructed	EO	Response
tt7/vt4	Reconstruction	New buildings are constructed	EO, RRVS	Recovery
tt8/vt5	Shelters remove	Temporary shelters are removed	EO	Recovery

Figure 7. Transaction and valid timelines of the application scenario and overview of the events considered in the application scenario.

Figure 8 shows the evolution of the integrated building inventory over different transaction times (tt0, tt4, tt6, and tt8). The different database states have been recovered using the transaction time history query (ttime_gethistory()), which creates a view of the transaction time history.

```
SELECT * FROM history.ttime_gethistory('object_res1.ve_resolution1', 'history.ttime_history');
```

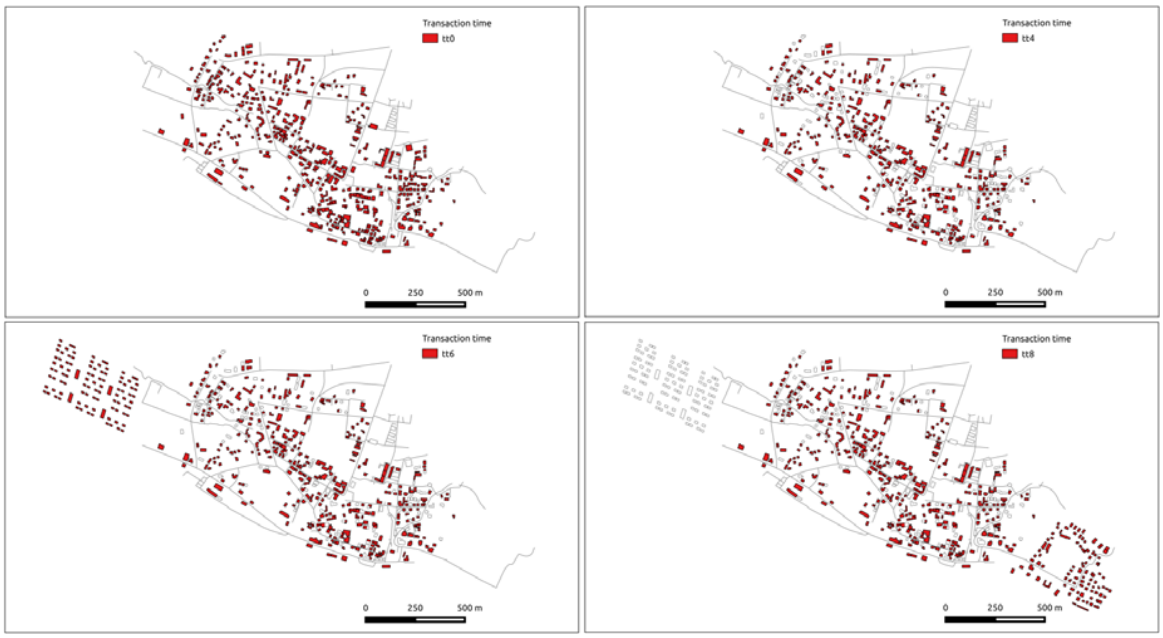


Figure 8. Integrated building inventory at different transaction times as recovered from the database archive by the ttime_gethistory() function.

Figure 9 shows the full valid time history of the database as recovered from the archive of the database with the valid time history function (vtime_gethistory()). It includes for each object a full lifecycle with the type of real-world change (creation, modification, and destruction), valid time of the changes, and recovered attributes at any given valid time. The magnified view and table in Figure 9 show the lifecycle of a selected object (gid: 480). It can be seen that this building was built on 06-01-1996, modified on 06-01-2014, and finally destroyed as a result of the earthquake on 12-01-2014. For each timestamp, a small subset of building attributes and their values are displayed to show how the building has changed over time. These changes include both real-world changes, such as the modification applied to the building by adding an additional floor, but also information updates, such as the change of the occupancy (OCCUPY) from unknown (OC99) to residential (RES99) and the occupancy detail (OCCUPY_DT) from unknown (OCCDT99) to single dwelling (RES1). The building is of material type (MAT_TYPE) unreinforced masonry (MUR) with material properties (MAT_PROP) adobe (ADO). Its height (HEIGHT) is defined as being in number of floors above ground (H) and changed from one floor to two floors (HEIGHT_NUM). For a comprehensive description of building attributes and their values, the reader is referred to the GEM Building Taxonomy [29]. The following query creates a view of the valid time history.

```
SELECT * FROM history.vtime_gethistory('object_res1.ve_resolution1', 'history.vtime_history',
 'yr_built_vt', 'yr_built_vt1');
```

In the following, a spatio-temporal query example is shown, which involves spatial, attribute, valid, and transaction time components. The query selects from the valid time history view all the buildings that were modified in the year before the earthquake (between 2013-12-01 and 2014-12-01) that are located inside a buffer of 50 m to a street, that are of material type reinforced masonry (MR), and that got an information update within the last week from the issue of the query.

```
SELECT DISTINCT ON (a.gid) a.* FROM history.vtime_history as a, streets as b
   WHERE a.yr_built_vt1 >= '2013-12-01' AND a.yr_built_vt1 <= '2014-12-01' AND
     a.yr_built_vt = 'MODIF' AND
     ST_INTERSECTS(ST_BUFFER(b.the_geom,50), a.the_geom) AND
     a.mat_type = 'MR' AND
     a.transaction_timestamp > (timestamp 'now' - interval '1 week')
   ORDER BY a.gid, a.transaction_timestamp DESC;
```

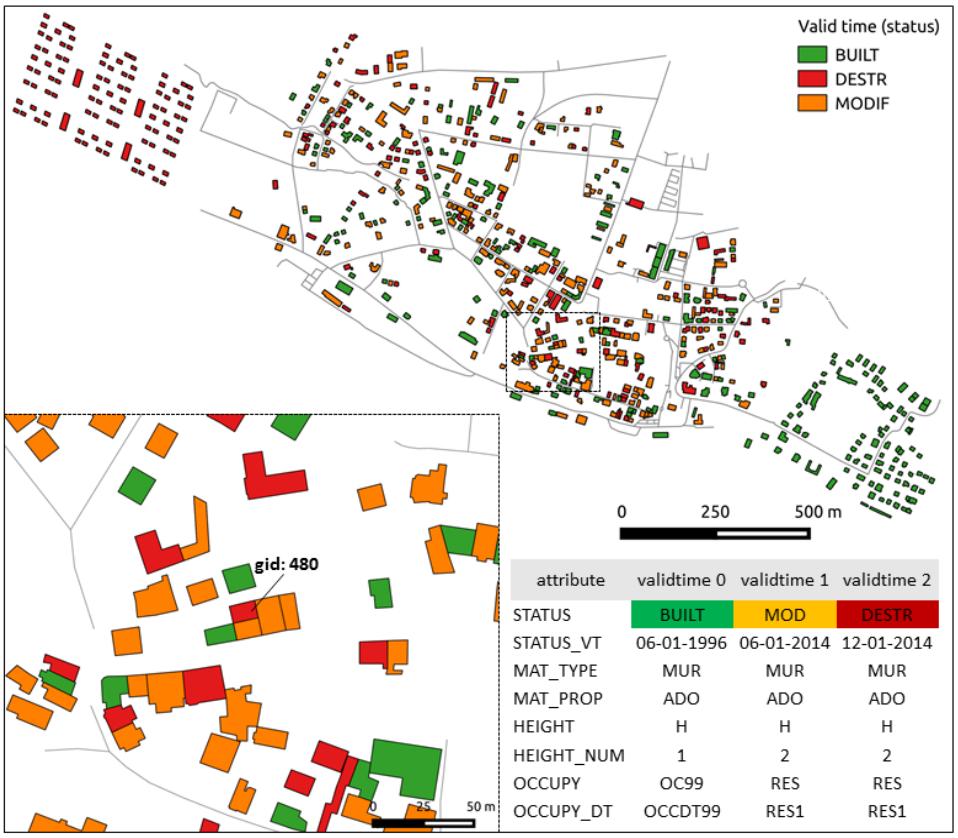


Figure 9. Valid time history of the database, which includes for each object an information lifecycle with the type of real-world change, valid time of the changes, and recovered attributes at any given valid time. The magnified view and table show the lifecycle of a selected object.

7. Discussion

This work provides a conceptual and technological framework for spatio-temporal integration, management, and versioning of exposure information throughout different phases of the disaster risk management cycle. Application scenarios have been provided to showcase the capabilities of the database and lifecycle management solution. Yet, a comprehensive testing of the full capabilities of the solution and the presented workflows on large real-world datasets remains an open task. A dynamic integration of large area OSM data with versioning and history support is envisaged and will be a good opportunity to further test the potentials and limitations of the proposed solutions and, in particular, to assess its scalability.

The database model and life cycle management solution provide a solid backend for data and metadata management in order to integrate data from different sources with varying quality, vintage, and reliability. In its current form, no explicit integration algorithm is implemented at the database level. However, several promising approaches for probabilistic information integration algorithms

are available [39,41] that could be combined with the proposed solution. The decision criterion that matches objects between different datasets and that triggers the check whether an object should be considered for an update or not is currently based on a spatial intersection of existing and new object geometries. This works well when object geometries are simple and of generally high accuracy and/or distributed with enough proximity to each other. In case of an integration of complex, large scale, and low accuracy datasets this, however, could introduce object mismatches. In such a case where pure geometrical matching would be error-prone and unique object identifiers are not available, more sophisticated approaches that match objects between different datasets by using several attributes and fuzzy matching criteria may be required.

Queries about location, spatial properties and relationships, time, temporal properties and relationships, and spatio-temporal behaviors and relationships are supported. Multi-representation in the form of a multi-resolution approach with dynamic linkage between resolutions is also included. However, some functionalities such as the generalization of geometries between different resolutions should be further extended and different generalization functions should be made available. Only a simple generalization function for polygon geometries is currently implemented in the database model. Modifications and extensions can, however, be applied without major changes to the basic design.

8. Conclusions

The consideration for the information life cycle, from collection and storing to updating and disposal, is rarely mentioned in the context of disaster risk management. Yet, ever-increasing amounts of information are being made available. With the provision of high-quality EO data, the increasing availability of data from open-sources, and the results of the collection effort of many non-interacting players (e.g., governmental institutions, NGOs, research institutions), a progressive layering of information with different quality, vintage, and reliability unfolds. In order to make these growing and increasingly dynamic datasets accessible and actionable in the framework of disaster risk management, consistent spatio-temporal data management plays a key role. The proposed database and information life cycle solution for building exposure data provides a conceptual and technological solution to cope with this development. The specific points that were jointly addressed by this study include spatio-temporal data modelling, the use of facetted taxonomies, multi-representation, and information life-cycle management. From each of these research domains, the most promising approaches have been selected and combined into a comprehensive solution that meets the needs of current and future disaster risk management practices. The presented solution is free and open-source and can be accessed from GitHub [45] along with the scenario data and an extended set of sample queries. Database design is an iterative process, and the proposed solution is to be considered a prototype that still needs to be further tested in real world applications. Currently, the proposed solution is used to manage a multi-scale exposure model for Central Asia [46]. Future work will be dedicated to probabilistic information integration and to the implementation of multi-hazard integrated taxonomies aimed at the prompt assessment of risk in complex urban environments.

Supplementary Materials: The presented solution is distributed on a free and open-source basis and is available online at https://github.com/MWieland/sensum_db.

Acknowledgments: The authors would like to thank the editors and the anonymous reviewers for suggestions that helped to improve this paper. Furthermore, the authors would like to thank the Boundless team and all contributors for their great efforts and achievements with the development of GeoGig and the OpenStreetMap community for their enthusiasm and mapping efforts worldwide. This research has been supported by the SENSUM project (Grant Agreement Number 312972).

Author Contributions: Marc Wieland designed and implemented the solution, conducted the experiments and wrote the paper. Massimiliano Pittore supervised the research activity and provided guidance and suggestions for the database design and revisions during the writing of the paper.

Conflicts of Interest: The authors declare no conflict of interest. The founding sponsors had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Aubrecht, C.; Fuchs, S.; Neuhold, C. Spatio-temporal aspects and dimensions in integrated disaster risk management. *Nat. Hazards* **2013**, *68*, 1205–1216. [CrossRef]
2. Bilham, R. The seismic future of cities. *Bull. Earthq. Eng.* **2009**, *7*, 839–887. [CrossRef]
3. Spence, R. Saving lives in earthquakes: Successes and failures in seismic protection since 1960. *Bull. Earthq. Eng.* **2007**, *5*, 139–251. [CrossRef]
4. Uprety, P.; Yamazaki, F.; Dell'Acqua, F. Damage detection using high-resolution SAR imagery in the 2009 L'Aquila, Italy, earthquake. *Earthq. Spectr.* **2013**, *29*, 1521–1535. [CrossRef]
5. Brown, D.; Saito, K.; Liu, M.; Spence, R.; So, E.; Ramage, M. The use of remotely sensed data and ground survey tools to assess damage and monitor early recovery following the 12.5.2008 Wenchuan earthquake in China. *Bull. Earthq. Eng.* **2012**, *10*, 741–764. [CrossRef]
6. Federal Emergency Management Agency (FEMA). *Rapid Visual Screening of Buildings for Potential Seismic Hazards: A Handbook*, 2nd ed.Applied Technology Council: Washington, DC, USA, 2002.
7. Silva, V.; Crowley, H.; Varum, H.; Pinho, R. Seismic risk assessment for mainland Portugal. *Bull. Earthq. Eng.* **2015**, *13*, 429–457. [CrossRef]
8. Geiß, C.; Taubenböck, H. Remote sensing contributing to assess earthquake risk: From a literature review towards a roadmap. *Nat. Hazards* **2013**, *68*, 7–48. [CrossRef]
9. Pittore, M.; Wieland, M.; Fleming, K. Perspectives on global dynamic exposure modelling for geo-risk assessment: From remote sensing to crowd-sourcing. *Nat. Hazards* **2016**. [CrossRef]
10. GeoGig by LocationTech. Available online: http://geogig.org/ (accessed on 3 January 2017).
11. Roshannejad, A.; Kainz, W. Handling identities in spatio-temporal databases. In Proceedings of the International Symposium on Computer-Assisted Cartography, Charlotte, NC, USA, 27–29 February 1995.
12. Paredaens, J.; Van den Bussche, J.; Van Gucht, D. Towards a theory of spatial database queries (extended abstract). In Proceedings of the Thirteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, Minneapolis, MN, USA, 24–27 May 1994.
13. Snodgrass, R.T. Temporal databases. In *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*; Frank, A.U., Campari, I., Formentini, U., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1992; pp. 22–64.
14. Langran, G. *Time in Geographic Information Systems*; Taylor & Francis: Oxford, UK, 1992.
15. Worboys, M. A unified model for spatial and temporal information. *Comput. J.* **1994**, *37*, 1–9. [CrossRef]
16. Raza, A.; Kainz, W. Cell tuple based spatio-temporal data model: An object oriented approach. In Proceedings of the 7th ACM International Symposium on Advances in Geographic Information Systems, Kansas City, MO, USA, 2–6 November 1999.
17. Raza, A. Working with spatio-temporal data type. In Proceedings of the XXII Congress of the International Society for Photogrammetry and Remote Sensing, Melbourne, Australia, 25 August–1 September 2012.
18. Zhao, L.; Jin, P.; Zhang, L.; Wang, H.; Lin, S. Developing an Oracle-based spatio-temporal information management system. *Lect. Notes Comput. Sci.* **2011**, *6637*, 168–176.
19. Abraham, T.; Roddick, J.F. Survey of spatio-temporal databases. *GeoInformatica* **1999**, *3*, 61–99. [CrossRef]
20. Peuquet, D.J. Making space for time: Issues in space-time data representation. *GeoInformatica* **2001**, *5*, 11–32. [CrossRef]
21. Pelekis, N.; Theodoulidis, B.; Kopanakis, I.; Theodoridis, Y. Literature review of spatio-temporal database models. *Knowl. Eng. Rev.* **2004**, *19*, 235–274. [CrossRef]
22. Goodchild, M.F.; Glennon, J.A. Crowdsourcing geographic information for disaster response: A research frontier. *Int. J. Digit. Earth* **2010**, *3*, 231–241. [CrossRef]
23. Sester, M.; Sarjakoski, T.; Harrie, L.; Hampe, M.; Koivula, T.; Sarjakoski, T.; Lehto, L.; Birgit, E.; Nivala, A.-M.; Stigmar, H. *Real-Time Generalisation and Multiple Representation in the GiMoDig Mobile Service*; Lund University: Lund, Sweden, 2004.
24. Stoter, J.E.; Morales, J.M.; Lemmens, R.L.G.; Meijers, B.M.; van Oosterom, P.J.M.; Quak, C.W.; Uitermark, H.T.; Brink, L. Data model for multi-scale topographical data. In *Headway in Spatial Data Handling*; Ruas, A., Gold, C., Eds.; Lecture Notes in Geoinformation and Cartography; Springer: Berlin/Heidelberg, Germany, 2008; pp. 233–254.

25. Hampke, M.; Sester, M. Real-time integration and generalization of spatial data for mobile applications. *Geowiss. Mitteilungen* **2002**, *60*, 1–13.
26. Kolbe, H. CityGML, KML und das Open Geospatial Consortium. In Proceedings of the 13th Münchner Fortbildungsseminar Geoinformationssysteme, Munich, Germany, 26–28 February 2008.
27. Wieland, M.; Pittore, M.; Parolai, S.; Begaliev, U.; Yasunov, P.; Tyagunov, S.; Moldobekov, B.; Saidiy, S.; Ilyasov, I.; Abakanov, T. A Multiscale Exposure Model for Seismic Risk Assessment in Central Asia. *Seismol. Res. Lett.* **2015**, *86*, 210–222. [CrossRef]
28. Hoffmann, E.; Chamie, M. *Standard Statistical Classifications: Basic Principles*; United Nations Statistics Division: New York, NY, USA, 1999.
29. Brzev, S.; Scawthorn, C.; Charleson, A.W.; Allen, L.; Greene, M.; Jaiswal, K.; Silva, V. *GEM Building Taxonomy v2.0*; GEM Building Taxonomy Global Component; Global Earthquake Model: Pavia, Italy, 2013.
30. Federal Emergency Management Agency (FEMA). *Multi-Hazard Loss Estimation Methodology*; FEMA: Washington, DC, USA, 2003.
31. Grünthal, G.; Musson, R.M.W.; Schwarz, J.; Stucchi, M. *European Macroseismic Scale 1998 (EMS-98)*; Cahiers du Centre Européen de Géodynamique et de Séismologie 15; Centre Européen de Géodynamique et de Séismologie: Luxembourg, 1998.
32. Wald, D.J.; Earle, P.S.; Allen, T.I.; Jaiswal, K.; Porter, K.; Hearne, M. Development of the US Geological Survey's PAGER system (Prompt Assessment of Global Earthquakes for Response). In Proceedings of the 14th World Conference on Earthquake Engineering, Beijing, China, 12–17 October 2008.
33. World Housing Encyclopedia. Available online: http://www.world-housing.net/ (accessed on 4 January 2013).
34. Broughton, V. Faceted classification as a basis for knowledge organization in a digital environment: The bliss bibliographic classification as a model for vocabulary management and the creation of multidimensional knowledge structures. *New Rev. Hypermedia Multimedia* **2002**, *7*, 67–102. [CrossRef]
35. Chen, Y. Information valuation for information lifecycle management. In Proceedings of the Second IEEE International Conference on Autonomic Computing, Seattle, WA, USA, 13–16 June 2005.
36. Congalton, R.G.; Green, K. *Assessing the Accuracy of Remotely Sensed Data: Principles and Practices*; CRC Press: Boca Raton, FL, USA, 2002.
37. Huber, F.; Schmidt-Petri, C. *Degrees of Belief*; Springer: Heidelberg, Germany, 2008.
38. Shi, W. Towards uncertainty-based geographic information science–theories of modelling uncertainties in spatial analyses. In *Advances in Spatio-Temporal Analysis*; Taylor and Francis: London, UK, 2007.
39. Tu, H.; Allanach, J.; Singh, S.; Pattipati, K.R.; Willett, P. Information integration via hierarchical and hybrid Bayesian networks. *IEEE Trans. Syst. Man Cybern. Part A Syst. Hum.* **2006**, *36*, 19–33.
40. Butenuth, M.; Gösseln, G.v.; Tiedge, M.; Heipke, C.; Lipeck, U.; Sester, M. Integration of heterogeneous geospatial data in a federated database. *ISPRS J. Photogramm. Remote Sens.* **2007**, *62*, 328–346. [CrossRef]
41. Pittore, M.; Wieland, M. Toward a rapid probabilistic seismic vulnerability assessment using satellite and ground-based remote sensing. *Nat Hazards* **2013**, *68*, 115–145. [CrossRef]
42. GitHub. Available online: https://github.com/ (accessed on 3 January 2017).
43. Wieland, M.; Pittore, M.; Parolai, S.; Zschau, J. Exposure Estimation from Multi-Resolution Optical Satellite Imagery for Seismic Risk Assessment. *ISPRS Int. J. Geo-Inf.* **2012**, *1*, 69–88. [CrossRef]
44. Wang, S.; So, E.; Smith, P. Detecting tents to estimate the displaced populations for post-disaster relief using high resolution satellite imagery. *Int. J. Appl. Earth Obs. Geoinform.* **2015**, *36*, 87–93. [CrossRef]
45. SENSUM Database GitHub Repository. Available online: https://github.com/MWieland/sensum_db (accessed on 14 March 2017).
46. Wieland, M.; Pittore, M.; Parolai, S.; Begaliev, U.; Yasunov, P.; Niyazov, J.; Tyagunov, S.; Moldobekov, B.; Saidiy, S.; Ilyasov, I.; et al. Towards a cross-border exposure model for the Earthquake Model Central Asia. *Ann. Geophys.* **2015**, *58*, S0106.