From consistency to flexibility: handling spatial information schema thanks to a middleware in a 3D city-modeling context

Gilles-Antoine Nys [1,]* and Roland Billen [1]

[1] Geomatics Unit, UR Spheres, University of Liège (ULiège), Allée du six Août, 19, 4000 Liège, Belgium; ganys/rbillen@uliege.be (R.B.)

* Correspondence: ganys@uliege.be (G.A.N.)

Abstract: Twinning elements of reality gains a growing interest in support of decision-making, learning and simulations: a single and shared model should provide a unique integrative basis for managing assets of any replica of the real world. From a technical viewpoint, sharing and opening information requires both an exchange format and a high degree of freedom and flexibility. It should allow an important number of users to manage this information, to modify it, etc. Storing and manipulating spatial information concerning the urban built context currently focuses on ensuring consistency thanks to relational databases and predefined schemas. Following a paradigm shift from a relational database to a NoSQL database, a schema validation middleware is proposed to improve the flexibility storage by conceding a share of its consistency. The flexibility improvements thus provide users a common basis that is able to evolve all along the lifecycle of their models and applications as required for twinning things. It allows users and their applications to take advantage of new storage features such as common: versioning, partitioning, prioritization, applications profiles, etc. The middleware and their new capabilities are illustrated thanks to the CityJSON encoding and its simplified schema for a document-oriented database.

Keywords: Schemaless database, NoSQL, middleware, 3D city model, CityJSON

---

1. Introduction

The digitization of real world elements improves activities and applications in many domains. This could be achieved, for example, through the creation of a digital model providing a single integration basis for all these activities. However, in practice, even if a conceptual schema allows structuring elements around a common base, different competing models might exist and provide a different representation of the same reality: not all users are interested in the same aspect or the same details.

The design of a digital model is a long and difficult process that requires compromises. As the needs of each application are not the same, one often prefers to use a specific model, close to the needs of the application, which itself is also specific. It is often the reason that does not allow the implementation of a shared digital model: what is the vision of reality that is necessary but also sufficient regarding all the users' activities around the model? Should applications that are considered more complex make concessions by using a generic model or should we impose complexity on applications that can be limited to something simpler? This would lead to a potential disconnection with reality on the one hand due to a loss of information in a generalization. On the other hand, situations where interactions would be cumbersome and too expensive in terms of resources without reason can appear.

Therefore, several questions remain unanswered and requires a response whether it can be illustrated in a paradigm shift in the management of the data, in the technique and its applications or by using the new capabilities offered by the recent technological advances in hardware:

a) Is this choice still relevant nowadays?
b) Given that there is only one ground truth but an infinite number of potential digital models, why should there be any compromise?
c) Could we not propose a solution that would allow storing a unique digital reality while making the representation we make of it dynamic according to our needs?

In addition to the substantial investment involved in designing a digital model, saving the model, often in relational mode, seems to explain some of the compromises made. In a web application, the server, which allows exchanges and a part of the processing, and the client, which is the data consumer, do not impose any concrete limitation. Structured beforehand and thus guarantying the application consistency, the rigidity and inertia of the relational model make it a change-resistant solution (complicated addition of heterogeneous data, modifications of the basic schema are always at the expense of some performance, difficult maintenance, complex horizontal scalability, etc.).

Would it not be possible to propose a storage solution that allows various independent applications to store and search relevant information in a single place? The guarantee on consistency would then be carried over to the server and the interoperability would be ensured using exchange standards. In this type of architecture, a document-oriented NoSQL database would allow completely free-shared storage of information without prior structuring. The server would then be responsible for the information structure by filtering it during exchanges with the various clients (both for storage but client requests).

The contribution of this paper is twofold. On the one hand, from a purely technical point of view, it provides a middleware, which acts as a bi-directional filter on server queries and would then filtrate information following CityJSON semi-structured schemas. Added to the simplified database schema for the storage of CityJSON models in a document-oriented database, it provides the core basis of an accessible storage solution. Provided in a convenient and well document framework, it should allow people developing their own use keeping in mind standardization. This flexible but still consistent data management helps developers to make bridges between the constituting parts of much greater city models management platforms.

On the other hand, in a dynamic that is always moving towards greater openness and information sharing, a new solution is proposed as an alternative to traditional solutions. Digital Twinning, a unique and digital 3D replica of a city, is now possible by using this first assumption. It is illustrated relying on a storage paradigm still too little used in our opinion: NoSQL databases. NoSQL databases and web-related technologies gained interest in the scope of 3D city modelling. However, most of the time, the new propositions are framed in a succession of improvements of a recognized tools limited to the purpose they had when they were set up. The new solutions are still too often neglected in favor of traditional solutions without addressing the problem from the start: the design of the tool.

All the principles and ideas developed in this paper are illustrated in the context of three-dimensional urban modelling and city digital twinning. The contribution is therefore not about the concept of middleware itself, but also in the answers to the recent questions formulated above.

The paper is structured as follows: the main topics studied are the exchanges standards and the role of the database in a GIS architecture. First, the various standardized way of querying and accessing geographical information, the city modelling standards, their semantic data model (CityGML and CityJSON) and the usage of these standards in shared or unshared web architectures are presented. The state of the art is assessed to frame this research in storing 3D city models in databases and deliver them on the web. Then, after a quick presentation on what NoSQL databases are and their differences with relational databases, the paradigm shift to a NoSQL database and its basic specifications are evaluated. The principle and benefits of a schemaless database are discussed afterward. Insights are also given concerning the usage of middleware in geospatial data management. Different methods of accessing information through features query services are presented in parallel with the major contribution of this paper: a bi-directional filter that simplifies the recording of information but guarantees the consistency of exchanges. Finally, future developments are considered as improvements and new possibilities that can be developed thanks to this new paradigm and the middleware.

2. Related works

Related works are divided in two different but interconnected parts: "Exchanges and standardization" and "Role of the database". While the first presents standards for structuring information in the urban built environment, the second part is a focus on the role of the database and its various shapes. The logical articulation of this part goes from a more general section to a more specific section that places our contribution in its context.

2.1. Exchanges and standardization

A "Digital Twin" is defined as "a virtual representation of a physical asset enabled through data and simulators for real-time prediction, monitoring, control and optimization of the asset for improved decision making throughout the life cycle of the asset and beyond" (Rasheed, San, and Kvamsdal 2019). On a conceptual level, especially in city modelling, the prospective potential of reality twinning is large (Shahat, Hyun, and Yeom 2021). Even if a wholly mirrored city is yet not available, improvements are relatively fast. In particular, improved data processing would make it easier to use the models and find information, but also to share it. Above all, the pooling of information from all kinds of sources is the main advantage of twinning. At this stage, all these considerations are anticipated to accurately reflect and affect the city and model's functions: data management, visualization, situational awareness, planning and prediction, integration and collaboration. Consequently, the search and processing of data must be simple and attractive (Schrotter and Hürzeler 2020). Indeed, supporting decision processes should be made in a comprehensible way all along the lifecycle management. The focus is made on the contrast between the static of the relational databases and the continuous evolution of users' needs. Behind the idea that the digitalization enhances the communication, The World Avatar (TWA) is a project led by the CARES center of the University of Cambridge in Singapore (Mei Qi et al. 2021). The TWA intends to capture the idea of representing every aspect of the real world in a digital model. It is thus a large-scale project gathering various researchers in a wide range of research areas. In concrete terms, it takes the form of a dynamic knowledge graph (dKG) that should improve the interoperability between heterogeneous data formats, software and applications (Chadzynski et al. 2021).

In GIS architecture, many efforts have been made on the database tier (Zlatanova and Stoter 2006). However, there is still much room for improvement. For instance, relational databases do not support co-existing schema versions natively. It is thus complex to develop tools without imposing them to be created prior of any production launch. Smart solutions need to be found in order to allow concurrent versioning. Among these solutions, a bidirectional database evolution language provides a solution for the co-existence of schema versions using *delta-code* (Herrmann et al. 2017). This language allows increasing the freedom to easily change the physical table schema but at the expense of some performance. Once the schema of a relational database has evolved, the stored data should also comply with the new structure. It imposes to guarantee the usability of the newly ordered database but also its completeness. A formal basis, which helps developers with the expensive and error-prone task of manual co-evolution (of both schema and data) is compulsory (Herrmann et al. 2018).

The consumption of performance is highlighted in comparison between the features of the relational versus the NoSQL databases. An empirical comparison of their average execution times gives insight on their specific advantages (Baralis et al. 2017). The number of concurrent users and dataset cardinalities have been also considered as they represent the great advantages of NoSQL. Among all the NoSQL database variations that exist, the document-oriented databases allow a great flexibility regarding the information structuration and their modifications. It allows storing documents in many convenient ways without imposing any predefined and strict schema, as would a relational database. Research is being carried out on the automatic creation of structures based on UML diagrams. However, it ensures the storage flexibility as it is the main asset of these NoSQL stores. A validation scenario presents the creation, its complexity metrics and states on the NoSQL assets (Gómez, Roncancio, and Casallas 2021).

Indeed, modelling a relational database might become a tremendous process: all requirements must be assessed beforehand in order to build an application that meets all user needs. In the NoSQL environment, there is no equivalent to the Unified Modelling Language (UML) used by relational databases. Some could use new notation based on UML or Entity Relationship (ER), eXtensible Markup Language (XML), etc. (Vera-Olivera et al. 2021). A systematic review on NoSQL databases explores the current state of research regarding their design methods (Roy-Hubara and Sturm 2020). One of its findings states that database design should meet non-functional requirements. It means that database design should not state on what to do or must do but how to do things: in other words, the absence of predefined schema is an opportunity and must be taken to its advantage.

A middleware is a piece of software that implements communication solutions for an operating system. It is commonly used in distributed architecture to support input/output between stacks. In the scope of GIS architecture, it allows merging multiple and heterogeneous data sources (Cha et al. 1999) and multi-storage paradigm architectures (Wong, Swartz, and Sarkar 2002; Li et al. 2018). Handling inputs and outputs also favors data integration without impacting on performance (Haas et al. 1999). For example, some propose to facilitate the merging of city modelling and building information modelling standards through a dedicated middleware (Schultz and Bhatt 2013).

Looking at the large family of NoSQL databases, the validation of exchanges using schemas is nothing new. For example for knowledge graphs such as these based on the RDF model uses the Shapes Constraint Language (SHACL) (Knublauch and Kontokostas 2017). In the context of spatial validation, it emphasizes recursive filtering and validation (Corman, Reutter, and Savković 2018) and the reusability of validation schemes (Debruyne and McGlinn 2021). Such a technical solution is one of the basic pillars of the work towards a global European infrastructure (Huang et al. 2019). These principles are commonly called "application profiles".

In the same way but at a different level of the web architecture and a more global ingestion process, GraphQL is an API layer that allows people querying and mutating already existing data. It is the closest thing to a universal method of questioning. The request defines itself the desired structure of the answer. Recent improvements on GraphQL demonstrate their usage in network bandwidth optimization (Brito, Mombach, and Valente 2019). However, like any new technology, it comes with drawbacks (Hartig and Pérez 2018; Wittern et al. 2019). However, there is no official spatial features nor capabilities.

Geospatial data are data about objects, events, or phenomena that have a location on the surface of the earth. It combines location information, which can be static or dynamic (usually coordinates or combinations and complex arrangements of them) and attribute information (characteristics and knowledge of the object). Given all these considerations, the exchange and the storage of such information imposes the usage of dedicated tools: spatial standards and spatial databases. The Open Geospatial Consortium (OGC) has a mission to improve geodata accessibility providing standards and normative exchanges formats. These standards are global resources that are publicly available and free to use. Among others, the Web Features Service (WFS) Interface Standard provides an interface allowing requests for 2D geographical features. A new version has recently been published in a legacy review (Clemens, Panagiotis, and Charles 2019). It has been done as to allow platformindependent calls across the web. This review is part of a new bigger family: “OGC APIs”. These APIs are developed in order to make it easy for anyone to provide geospatial data on the web but in a standardized way. The different APIs are meant to provide building blocks that can be used to build APIs that are novel and more complex. Along with the maps, coverage and processing services, the features are part of the improvements brought in this new standards family. The “OGC API - Features - Part 1: Core” is restricted to read-access and describes the mandatory capabilities to implement a data access interface (Clemens et al. 2019). Future capabilities such as creation and modification of existing features but also additional coordinate references should be developed in future parts. Alongside, 3D Tiles is designed for streaming and rendering of massive 3D content (Patrick, Sean, and Gabby 2019). It should not be confused with the OGC API - Features as the second concerns a way to serve information on a specific element and all its semantic information: attributes, versioning, etc.

In addition to the exchange protocols, the OGC standards also provide standards for the exchanges and representation of knowledge. CityGML is the most widely used standard for 3D city modelling (Gröger and Plümer 2012). Recent developments are related to extending the standard features: linking with other common standards (Biljecki et al. 2021), wind simulations (Deininger et al. 2020), heating demand prediction (Rossknecht and Airaksinen 2020), etc. Among other solutions, 3DCityDB is a software package that consists of a database schema for spatially enhanced relational databases. It improves the database with a set of procedures and software tools allowing to import, manage, analyze, visualize, and export CityGML models (Yao et al. 2018). Another CityGML data model usage consists of a compact and developers-friendly encoding alternative of this data model: CityJSON (Ledoux et al. 2019). Besides its simplicity and easiness to handle city models, many advantages derive from the JSON encoding and its semi-opened structure: native support of metadata and refined levels-of-detail (Nys, Poux, and Billen 2020), easier integration in common GIS tools (Vitalis, Arroyo Ohori, and Stoter 2020), lightweight and scalable base to support complex web applications (Virtanen et al. 2021), usage of combinatorial maps in topology structure (Stelios Vitalis, Ohori, and Stoter 2019), etc. This new encoding solution opens possibilities by reducing the cost of modifying data but also facilitates its exchange. It is part of a dynamic that is increasingly focused on the web and the pooling of knowledge: servicification. This dynamic is the process to migrate code and applications to a modular and service-oriented architecture. This results in the production of reusable and decoupled components while also reducing duplication. It finally results in a better usage of resources and the sharing of capabilities and information. Servicification in geographical systems is well illustrated in SOA architecture (Service-Oriented Architecture) (Allah Bukhsh, van Sinderen, and Singh 2015; Nys and Billen 2021). A flexible architecture allows the composition and sequencing of data processing. The geospatial intelligence provided by such services is a proper solution to most of the geospatial application problems (Fricke, Döllner, and Asche 2018).

2.2. Role of the database

It is understood that 3D city models are great integrating bases for complex studies in various fields. This can be seen from the ever-increasing number of application domains extensions (ADEs) for CityGML (Biljecki, Kumar, and Nagel 2018): energy, noise, 3D cadaster, etc. However, even if the semantic information is well integrated in such models, their usability in simulations is not straightforward: this kind of linkage is often studied by the actors in the field of 3D modelling and not simulation experts. The method of storage is not necessarily responsible (Widl, Agugiaro, and Peters-Anders 2021). One is proposing to review the way in which the information, recorded in a relational database, is accessed and thus linked to the simulation tools (Yao et al. 2018). Without modifying the base, this solution makes it possible to spread the use of city models and their linked information.

The management of versions and history within 3D city modelling, which can be generalized by allowing different views on the same information, can be done through the use of an ADE of CityGML (Chaturvedi et al. 2017). This independent extension considers new aspects as managing multiple temporal interpretations of a city and its features. It is now part of the CityGML 3.0 data model and should thus be implemented in its various uses (Kutzner, Chaturvedi, and Kolbe 2020). Despite the proposed solutions for versioning, several issues remain (S. Vitalis et al. 2019; Kutzner et al. 2020). Six issues were evaluated and discussed among the data providers' incentives, the database implementation, etc. but more specifically: the need to collect additional lifecycle and versioning information (Eriksson and Harrie 2021). The problem highlighted on the additional information is that it requires a substantial restructuring of the technical solution and work processes. In addition, the increasing complexity of the database implementation increases with the number of versioning features included (Eriksson et al. 2021).

Besides the relational databases, the vast panel of NoSQL databases offer complementary solutions. NoSQL databases propose to review the storage structure of relational database. Among others, when the links between the elements are preponderant, graph databases are the most suitable. For instance, thanks to the graph isomorphism tools, even if they are resources consuming, change detection is made between versions of CityGML models (Nguyen and Kolbe 2020). Moreover, a much precise definition of the change types is given based on the graph structure. As it has been said, the graphs are useful for modelling the relationships between the city features. More precisely, the translation of these relations in Resource Description Framework (RDF) triples structures the semantic information of the urban built environment: the only inconvenient is that the geometric information is neglected (Malinverni et al. 2020). It is worth mentioning that ontologies are preserved during data conversions and can therefore be queried afterwards. It opens up fusion possibilities for city models with various sources using a NoSQL graph database: IFC, IndoorGML, etc. Structuring information in graphs also provide solution for bi-directional transformations. It allows deriving models from real CityGML models and instrument modelling and analysis facilities for digital models (Visconti et al. 2021).

Document-oriented NoSQL databases offer interesting possibilities. Besides any processing efficiency, the whole data structure has been reformed. It is much simpler than relational databases that use joint keys for example (Bartoszewski, Piorkowski, and Lupa 2019). Changing the user's perspective on data can improve or even rethink the basic idea of relational databases. The database design itself gives an answer to the multipurpose needs for WebGIS (Sutanta and Nurnawati 2019). Without providing a complete solution compared to what relational solutions offer, the NoSQL databases offer premises of spatial data management on the web (da Costa Rainho and Bernardino 2018). Especially in 3D city modelling, the shift from consistency to flexibility opens many possibilities (Nys and Billen 2021). In this research, a combination between CityJSON and the NoSQL document-oriented database provides an alternative to the traditional geodata management. The parallel can be drawn with 3DCityDB, which proposes a data schema for storage in a relational database. The comparison between the two tools was made in terms of performance but also in terms of their capabilities. In short, it improves the modularity of information thanks to the lack of schema for the database. Gains of performances and capabilities are remarkable kiss-cool effects too. For instance, proposing new extensions, and thus improving and adding features to the schema, is easier and supported in a convenient way thanks to the schema and its translation in the semi-open database structure (Nys et al. 2021).

3. Schemaless database

This definition of Rasheed et al. for "Digital Twin", even if it remains vague on the "virtual representation" term, focuses on the long-term usage and lifecycle of the information. This representation should therefore be required to be modular and flexible in order to adapt to current but also future needs. Without going for a complete avatar, a digital replica whose main characteristic is its shared uniqueness is a point worth studying. Even if relational databases provide solutions and capabilities, those are not suited for development in line with modifications in usage needs and horizontal scaling. It can therefore be considered that they do not address the root of the problem: the flexibility of schemes and thus the whole architecture modularity.

Tacking a step back, a web GIS architecture is constituted of three components: a client, a server and a database. While there is no limitation on the number for each tier, it should be at least one element for each. Thus, a wide range of combinations is possible. Moreover, the elements are not always parts of the same whole; they might be under responsibility of different organization, located in various places, etc. Most of the time, the server and the database are closely linked and why not installed on the same physical machine (the architecture thus become a "two-tier architecture"). A brief explanation of the usefulness of each tier provides a better understanding of the paradigm shift proposed started in previous research in which this contribution fits (Nys and Billen 2021).

The client is the consumer of the data. It can be a viewer, a GIS standalone software, a web application, etc. Since the "frontend's" capabilities are evolving, clients support more and more processing. For instance, the web browsers, thanks to the creation of the V8 JavaScript Engine (Chromium Project of Google), handle more and more capabilities (Kulawiak, Dawidowicz, and Pacholczyk 2019): heavy graphics computations, graphs manipulation, etc.

The server takes care of the processing part, or at least part of it, as the frontend improves as mentioned above. It manages the database connections and receive the clients' queries (Wagemann et al. 2018). It is possible for a client to query a database directly, but the presence of a server makes it possible to improve security, set up statistics, structure and guarantee the consistency of exchanges. With the database, it is part of what is called "backend".

The database saves information; it structures the data and allows its accessibility. For example, the relational mode structures information in tables and defines the relationships between them thanks to associations and cardinalities. Therefore, a predefined schema is mandatory so that the defined boxes and their links can be filled in later. It is the main advantage of using relational databases: the guarantee of consistency. Still, one can suffer of the predefinition of such framework. The users' needs and applications capabilities might evolve and no longer fit this schema. It could then be interesting to provide an alternative that concedes a loss of consistency to improve the architecture flexibility. A partial answer to this problem is to shift the use of a traditional database and move towards a NoSQL solution (Nys and Billen 2021). This contribution is in line with this answer and proposes to make a step further from the consistency to the flexibility of databases in the scope of modeling urban environments.

3.1.NoSQL paradigm

Before considering NoSQL solutions, attempts to improve the relational model are worth mentioning. One of these is the BiDEL language (Herrmann et al. 2017). However, these solutions gets around the problem without tackling its root. The language acts like an additional layer that improve the relational database capabilities. The database itself is not adequate to handle specific features. For instance, thanks to BiDEL, the versioning is simplified but it imposes to manage a new technology that adds complexity and potential problems. Tackling the rigid structure of the relational databases is avoided but not solved. It would be more interesting to find an integrated solution.

The research topics of the TWA project study the formalization, the evaluation and the repair of ontologies based on the CityGML and many other data models (in field such as environment, weather, etc.) (Mei Qi et al. 2021). Their integrated and dynamic knowledge graph structures information from a semantic point of view at least. As a complementary layer, the 3D geometric information brings unavoidable information concerning urban management. Undoubtedly, it should find an interest in developing a geometry support, if not at the beginning, at least at some point. This project nevertheless illustrates an important need: NoSQL databases not only offer new capabilities but also provide a very new storage paradigm and many advantages. Subsequent to it, it is not only the arrangement of the data that changes; it is the whole perception of it.

At this point, an explanation on the NoSQL storage paradigm should be given. NoSQL solutions (Not Only SQL) are defined as "everything that is not relational". In fact, it is much more complex than that. The NoSQL family responds to capabilities that are indeed different from the relational databases but still correspond to a set of definitions. The main difference between relational databases and NoSQL solutions lies in the management of their schemes. NoSQL databases, without going into the details of their various families, do not limit the data to be filled in predefined boxes. In other words, the database does not impose a schema for the data to be stored. NoSQL databases are "schema less databases". Besides the ACID characteristics of traditional databases (Atomicity, Consistency, Isolation and Durability), the NoSQL databases follows the BASE principles:

- Basically Available: the data are always available; there is no downtime despite any network failure or temporary inconstancies. A "non-response" is impossible from the store. Whether it is a success or an error, there is always an answer to every request.
- Soft state: even without any input, the system state could change over time. This characteristic is required for the following "eventually consistent" property.
- Eventual consistency: if no further updates are made to an item for a long enough period, all users will see the same value for the updated item. In the meantime, anything can happen. The system will eventually become consistent once it stops receiving input.

The "eventual consistency" characteristic is the linchpin. The soft state characteristic is one of its requirements and the availability is a quality of life asset but does not have any link with the consistency. The third characteristic is indeed the most interesting one: the eventual consistency means that the consistency is not set by the database itself and might not be always guaranteed. The database could deliver different information to various users in some state. The compromises made on consistency and the above-mentioned responses' heterogeneity can be considered as potentially harmful. This is true if the database is considered as an isolated component. The server, and why not, the clients, might have a role to play in the consistency assessments.

As they have been defined in the previous section, clients are passive consumers and thus free regarding the data structure. Both databases and clients should be independent services but clients must be able to work with what the databases provide, as they are more flexible. Even if they can support a part of the computations, clients should not require to control and validate the server responses. They just visualize or process the data but does not restructure or modify it. Otherwise, they will become an active component and the server may have neglected some of its responsibility.

The key idea of this contribution is to take the opposite view of the "schema less" database and to take advantage of this actual flexibility. Since no schema is mandatory by the database, the opportunity is to store data without any restrictions beside technical constraints: format, encoding, etc. Any shape of information can thus be stored in the same place. Taking the assumption that an infinite number of record variants can be stored in a unique database, one can consider that some records will share a common basis or correspond to a common structure. Where several pieces of information relate to the same real object, the use of a single and unequivocal identifier should allow connecting these pieces. Moreover, each element might have common attributes and/or ways of representation with one another (versions, extensions, etc.). They are actually different copies or views of the same entity. Stating on a common basis and referring to real objects uniquely, one can consequently define the foundation of a shared but limitless model.

Every city is unique. It has its own history, its specific space, its citizens and their own lifestyles, etc. Many public services and stakeholders have their own views of the city and its assets. However, they should not be allowed to harm or modify those of others. In addition, if interactions should be possible, they should be done at least under pre-established conditions.

Back to the data store framework and its infinite theoretical set of city models, a common basis should determine the constituting elements of a city and their relationships: it is the main purpose of standards such as the CityGML data model. Since CityGML is a semi-open standard, it consist not only of a shared ground for city modelling but also for extensions and future applications. It thus offers the possibility to reuse the compliant data in different fields and applications that are themselves compliant to the data model. However, the majority of recent developments in 3D city modeling accept the relational storage mode and its advantages without questioning its initial capabilities. Hence, they focus on developing extensions proposing new features, new attributes and new relationships without considering any use of a unique and common digital model. Such a model, whose core can itself evolve as improvements are made, has been little studied. Among others, the ACID characteristics are part of these limitations.

3.2. Architecture specifications

Before presenting the architecture capabilities and the benefits of the new component, a specific point of its features should be discussed regarding the paradigm shift. As defined in the previous sections, its groundwork relies on three things: the usage of a NoSQL database that improves flexibility at the expense of some consistency, the usage of a common definition basis such as the CityGML data model and finally a new component that filtrates information. These specifications are available thanks to the simplified database schema for the management of CityJSON 3D city models in a document-oriented store (Nys and Billen 2021).

Thence, a first step towards ensuring consistency is done by implicitly choosing that all applications must be standard compliant. In the case of urban modeling and JSON-related technologies, CityJSON 1.1 is unavoidable. It is here worth mentioning that it remains an unspoken consensus for some tier: the database itself is not structured following any schema. No conditions are set during creation and modification on records about any cardinalities, document structure, document size limitations, etc. This is the role of the proposed component, which is mounted on the existing application server, and only it. The server can thus be used to lock users' exchanges and structure queries on the server but nowhere else.

While the current applications developed around this simplified database schema of CityJSON concerned the storage of multiple city models in a unique store, the new architecture will make an additional hypothesis: the store remains unique but the unicity is now generalized to the stored model also. Note that the number of frontend elements is still limitless. In summary, one database is shared by several server, or Application Programming Interfaces (APIs), that are themselves receiving queries from an unlimited number of clients on the web. All this is done under the assumption that the hardware is not a limited resource. The Figure 1~~Figure 1~~ illustrates the architecture of the shared database.

Formatted: Font: Not Bold

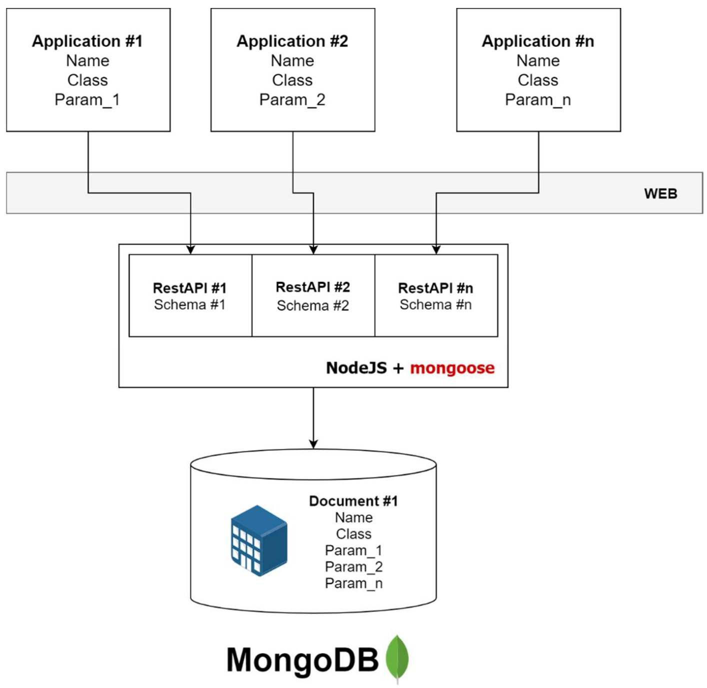

Figure 1. *Architecture of a shared and unique database*

Two requirements depicted in the Figure 1~~Figure 1~~ need an explanation: accordingly, in the simplified schema, a document, or record, corresponds to a model of a city or to an element of the urban built environment (i.e. an *AbstractCityObject*). Hence, in order to be able to refer to the correct record, a document stored in the database must be defined by a unique way of identification. For example the "name" attribute in each level of the architecture should be formatted in a similar manner. Secondly, a *Class*, which specifies the city object family, might also be given to objects in order to simplify the various queries. These classes are used to manage the different schemes by the server. Examples of *Classes* are *CityModel*, *Buildings*, *SolitaryVegetationObject*, etc. according to the CityGML model specifications. Other parameters (*Param_1*, *Param_2*, etc.) might also be defined in the core specification but also come from extensions. For instance, since the stored documents should implicitly comply with the specifications of CityJSON, it is thus possible for an application to query a *Building* object knowing beforehand some of its attribute: *address*, *roofType*, etc.

Formatted: Font: Not Bold

These considerations are generic to any number of clients and applications. As a result, information can be derived from a theoretically infinite number of architecture elements except for the database, which is deliberately intended to be unique. Hence, the whole architecture can be abstracted by a tree, so that the database would be the trunk and the clients the leaves. The servers will then be the tree branches (see Figure 2~~Figure 2~~). For the growth of the tree, the only limitations are network and hardware considerations since the data is constrained.

Formatted: Font: Not Bold

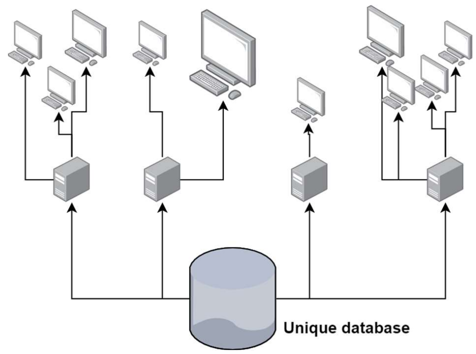

Figure 2. *Abstraction of the architecture in tree form*

Besides the semantic formalization of the shared information, this component could be the basis of more complex mechanisms. One can imagine that the unique model is used by various users from the same city government. While each city service is working on its specific aspect of the city model, details can be brought thanks to the filters (and thus versions, extensions, etc.). Besides, the security issues of a non-strict user, several concurrent schemas might be used by the same user community limiting accessibility in respect of the grade or hierarchy like application profiles do. This choice is left to the developers, as they may be interested in developing stand-alone applications or in routing users through a larger application.

Concerning the versioning, the new architecture not only ease data versions management but also the data model versioning. It is customary that a database and thus the stored information comply with a unique data model version. Updating the data when a new version of the data model is released can lead to the database becoming obsolete if there has been no insight on backwards compatibility. As a result, in this architecture, there is no need for data update in the database, only technical maintenance is mandatory. The filter will then serve information in respect of the desired version picking relevant information from the semi-structured whole. The same information can therefore be easily shared by different versions or different applications.

Thanks to the BASE characteristics of the NoSQL databases, the data are always available. This means that the database should always be operational and able to respond at any time. As the server component is independent of the database, any maintenance on a specific application will not limit the usage of the others. It therefore improve the flexibility of the whole architecture and provides a modularly solution for further developments.

As a comparison with a current good practice, GraphQL is an API layer that lies between a server and clients. It allows querying and mutating data in a generalized way (Wittern et al. 2019). Several notable differences are to be noted: first, GraphQL only allows a single endpoint on a database and imposes developing new features in the same way. Server functions and data manipulation are limited (Brito et al. 2019). Such a limitation would have limited us in the development of the application clients and especially with its links to the OGC API. Moreover, the document collections in the predefined simplified schema are built keeping in mind performances and most common queries. Staying with the development of advanced capabilities, GraphQL lacks of temporal and spatial features (Hartig and Pérez 2018). Such features are mandatory in the scope of city modeling. Mutations have another major concern because the tool was not originally created for this purpose: mutating functions are not conducted in parallel: each change waits until the previous one is finished. It is a major drawback when it comes to open the architecture to the many. It affects users experience in a very negative way. Finally, errors handling can become tremendous for developers since HTTP requests only serve 200 status queries (or 5xx if the server is not available at all). Avoiding these drawbacks and allowing developers to create the best endpoints they need is an imperative.

Since maintaining the data consistency is no more the responsibility of the database, it is now the role of another architecture tier: the server. In the proposed architecture, because of the storage paradigm shift, this guarantee is transferred to the server or at least to one of its components. The present improvements are made in line with the previous: it proposes a proof of concept using JavaScript libraries. The new component is hosted on a NodeJS server, a JavaScript runtime environment. The component is built on the *mongoose* library, an open source solution that provides built-in type casting, schema validation, query formalization and building, business logic hooks, etc. (https://www.npmjs.com/package/mongoose). Among these features, formalizing a query and the built-in type casting do not concern any aspect of storage consistency. By contrast, the schema validation is the cornerstone of the proposed improvement. In practical terms, mongoose acts like a bi-directional coat that filtrates information between the client, the server and the database. It acts both as a mediator and as a wrapper (i.e. in both directions). A predefined schema on the server maps a requested resource to a collection of documents in the database and serves relevant information to client and vice versa. It also allows verifying the format and encoding of any exchanged resource. This second feature does not play a role in the consistency of schemes besides technical considerations.

As far as semantic information is concerned, thanks to the discriminated schemas of mongoose and its inheritance capabilities (which are not possible in JSON schemas); some variations can be added to the schema definitions without having to modify the initial requirements. For instance, a *Building* is nothing more than an *AbstractCityObject* with an address, a roofType, etc. The added information is still compliant with the *AbstractCityObject* schema. A *SolitaryVegetationObject* is an *AbstractCityObject* that might have a *specie*, a *trunkDiameter*, etc. Nevertheless, there is no requirement for each application to have the same exact definition of what a type of feature exactly is. It is a matter of agreeing on the common basis from the CityJSON specifications. The notion of hierarchy being absent from JSON schemas, this point reinforces the demonstration of the architecture flexibility as it simplify modifications without damaging the already existing schemas. Note that JSON schemas require checking several concurrent schemas rather than offering the possibility to specialize them.

By going further into the technical definition of the architecture: such a filter stands as a middleware. A middleware is a software that lies between an operating system (i.e. the server) and the applications running on it. Common middleware provide security layers (limiting the number of request, cryptography, etc.), cross-origin requests managers (accessing restricted resources from a remote domain), authentication layers (checking tokens and/or registered users, etc.), etc.

Beside these technical features, it also allows for the removal of excess information sent by clients or delivered to them but also the databases. The component filters information and maps it to the documents collections in the document-oriented database. This is done so that the semi-structured database is not polluted by incorrectly structured or unwanted information. This mapping consist of a collection of features schemas (not JSON schemas), themselves built on the CityJSON specifications. In addition to the schemas, inheritance is added between collection elements thanks to the mongoose features. In the proposed architecture, all the capabilities of a middleware are used such as it acts like a bidirectional filter:

(a) In one direction, based on the queries made by the clients (i.e. in a writing way), it filters the city objects and their attributes before any storage and/or potential updates. For the reminder, every API might be independent and built in such a way as to allow flexibility. They offer several different types of requests with different accesses, different connections, and different schemas as a result. For instance (see Figure 3~~Figure 3~~), the application #2 is not allowed to modify (and perhaps damage) the objects and attributes handled by the application #1. Be aware, however, that some objects might be shared by several applications and the same thing for the attributes of shared objects. The value of a common standardized and documented basis is again demonstrated here. It is a major prerequisite.

(b) In the other direction, for the documents queried by the clients from the database, the filter works exactly the same way. There are not only the format and the encoding that are verified but also the semantic information thanks to the schema specifications. It is retrieved from the database given that the attributes are checked and validated by the application-related schema. No feature object or attribute that has not been defined beforehand in the schema will be served as a response. It is

Formatted: Font: Not Bold

important to note that while format consistency can easily be checked, logical consistency, i.e. the compliance of values with their semantics, cannot be verified regarding coherent meaning. This could limit the applications interaction and information retrieving but it is part of the responsibility of each data producer and its policy on whether or not to open the data and document it. In this context, the technical elements necessary for this sharing are provided without taking a position on this last aspect.

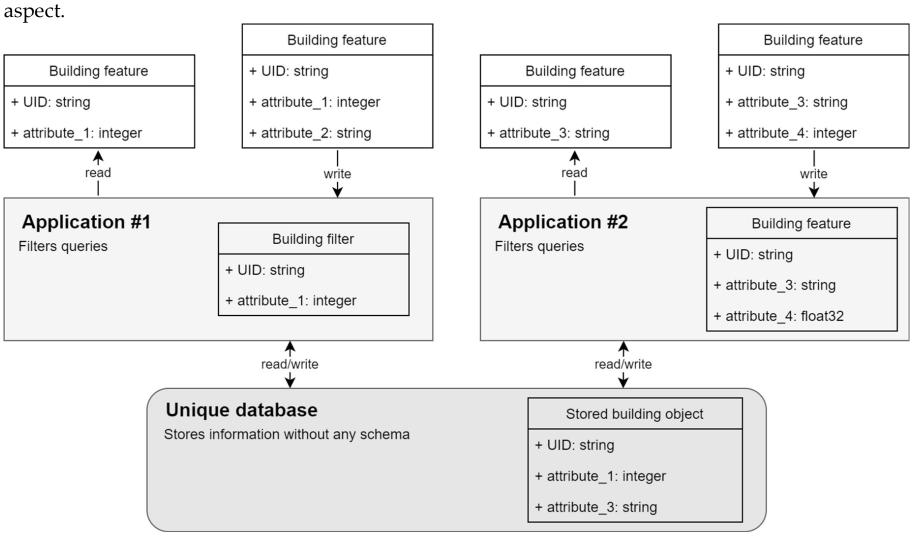

Figure 3. *Illustrated example of the bi-directional filter principles*

In the Figure 3~~Figure 3~~, the example represents two applications that want to register and retrieve information on the same object of the same class. Considering that both the unique identifiers are the same (whether they are URIs (Uniform Resource Identifier), UUID (Universally Unique Identifier), etc.), the object is defined by four attributes: two attributes handled by each applications. Some points are noticeable:

Formatted: Font: Not Bold

- The *attribute_2* is not stored in the database because it is not allowed in the server schema of the first application. As it does not pass the filter in a writing way, the information is not send to the database and thus cannot be queried afterwards.
- The *attribute_4* is not stored because it is not properly formatted with respect to what is required in the second application's schema. If this attribute has been correctly formatted, it will be stored in the database and made searchable by clients.
- Both *attribute_1* and *attribute_3* are stored in the database but their use is limited to the separate framework of the two applications.

In the example above, the attributes are "basic and common" data types: string, integer, float32, etc. In the context of spatial information, and in the even more specific 3D city modelling, "spatial" data types require a dedicated management to handle their specificities. In addition to the complex representation of the built environment, the formatting of geographical information and features geometries imposes conditions. It should be noted that geometries must follow well-defined patterns most of which are defined by international standards. Among others, the concept of level of detail needs to be discussed and addressed (Biljecki 2013).

As defined in the simplified CityJSON schema for document-oriented databases, the geometries are managed in a dedicated mass-collection regardless of their type, the number per element and their level of detail (Nys and Billen 2021). For the reminder, every type of geometry has its own validation schema whether it is a *Solid*, a *MultiSurface*, etc. They all share a common basis but some specificities are brought in their specific sub-schema definitions by inheritance. It works the same way as the *AbstractCityObject* and the *Buildings* schemas. The schemas are independent of any level of detail and all types of geometries can be arranged to create various types of levels of detail. A unique identifier refers these geometry documents, one for each level of detail, to the feature documents in the city objects collection. While storing and thus writing a geometry in the database, each element and level are checked against their scheme. On the contrary, in order to optimize and better adapt to the users' needs, querying a specific geometry can be done specifying the desired level-of-detail. All attributes of the geometry are served since it is a feature in its own right.

In practice, given that CityJSON handles the "refined levels-of-detail" (Biljecki, Ledoux, and Stoter 2016; Nys et al. 2020), the geometries can be queried in a compound manner. Either, the specified LoD is itself a refined one and thus can be retrieved if it exists. Either, if it does not exist or is a broader one: the most detailed one is recovered while remaining in a coherent level order. For instance, for a geometry stored in 2.1 and 2.2 levels, querying a unique geometry for the 2nd level will respond with only the 2.2 document. Besides it, one cannot retrieve a LoD greater than the expected. It is done in way to reduce exchange weight and providing redundant information.

3.3. OGC API - Features

Besides the operations presented above and their request mode, it could be interesting for the users to handle features in a more standardized way. It is important to allow every user to have a view of what is stored in the database. It must be done in a reading-limited way so as not to compromise the database. Therefore, the new OGC API - Features has been implemented in order to provide a normalized and convenient way to do so (Clemens et al. 2019). It thus guarantees the consistency of the database keeping things secure and avoiding data mutation. Again, this could be done not through the database itself but thanks to the middleware. It is worth mentioning that not all information should be requested by everyone: the idea is to "see" what can be obtained. This must of course be done within the access limitation, security, hierarchy, etc.

The Figure 4~~Figure 4~~ depicts how the OGC API service is connected to the architecture. While all the other well-separated applications are limited to their own part of the data (and to the shared parts), the OGC API - Features service has access to everything (under conditions of safety, regulations, etc.). It is important to clarify that this service is a read-only protocol and should be considered as a view on the stored model.

Formatted: Font: Not Bold

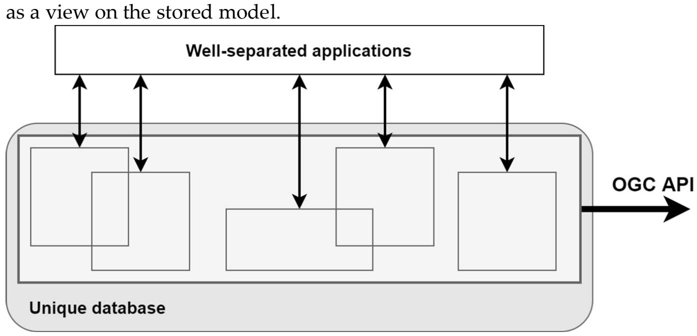

Figure 4. *Implementation of the OGC API - Features service*

A problem arises from the fact that most of the exchange standards, protocols and OGC services (Web Features Service, Web Map Service, etc.) are suited for two-dimensional geodata. CityJSON, as a 3D modelling standard, cannot be queried in a convenient way using the standards with a few exceptions. Moreover, the document-oriented database does not support any 3D indexing methods. It was thus necessary to build workaround solutions in order to allow spatial filtering at least in two dimensions.

Initially, OGC standards serve features with a single geometry. However, a CityJSON object can have an undefined number of geometries. These geometries provide a wide range of information corresponding to various levels-of-detail. An alternative is proposed to limit misunderstandings. Besides the limit, offset and bounding boxes parameters already used in the specifications, a new attribute is added to the query parameters: the *lod* (level-of-detail). As a reminder, the geometries are stored independently of any feature as a bulk in a dedicated collection. The simplified schema separates them in several documents even if they concern the same object. This is justified by the fact that the level of detail plays an important and very specific role in urban information management but also because of the spatial indexing capabilities of the database. The *lod* is thus introduced as a parameter in its own right in requests. In any case, where this parameter is not supported by an application, it is simply neglected and it is the greatest *lod* that is served.

In the official schema of the *AbstractCityObject*, the *geographicalExtent* attribute stores the 3D boundary box of the distinct features. Projecting the 3D box, a new 2D attribute *bbox* is created on the fly during the storage process. This new spatial extent is then used to build the spatial indexing in the database. This extent is also created for the whole city model itself. An important condition is imposed by MongoDB for spatial indexing: the coordinate of the any spatial information should be expressed as a GeoJSON object (RFC 7946). It thus imposes the use of the World Geodetic System 1984 (WGS84 - EPSG:4326) and thus the projection of the *bbox* attribute. If no reference system is provided in the model, it is considered as already expressed in WGS84 by default. This is a major drawback for the management of spatial information in document-oriented databases.

Future work should include the improvements brought by the added parts of the OGC API – Features family: Coordinate Reference Systems by Reference, Filtering and the Common Query Language and the Simple Transactions. This should be handled by the middleware, as it is neither the responsibility of the database provider.

Second possible improvement coming from the semantic web, the Uniform Resource Identifier (URI) is a great candidate for the unique identifier format. This URI identifier is a unique sequence of characters that identifies a logical or physical resource used by web technologies. While the purpose of the URI is to allow data extracted from various databases to be linked and to be identified unambiguously, it could also improve the management of the legitimacy of data. Such an identifier could be part of a certification process in which the responsibility of the city objects is part of the prerogatives of the city services. Its identity and its responsibility can thus be translated in the URI syntax in one way or another.

4. Conclusion

This paper makes a step towards a paradigm shift for the storage of geographical information: it provides a technical component and insights that concedes a loss of consistency in favor of more flexibility. It is illustrated in the context of 3D city modelling thanks to the implementation of a schema validation middleware. This could be performed, among other approaches, with the replacement of a relational database by a NoSQL document-oriented database. The main characteristic of the database considers that it does not handle any data schema. It therefore does not require filling in predefined boxes or meeting non-technical requirements as does relational. Conversely, the logical, conceptual and physical models are not prerequisites. The consistency management is then shifted to the server and more specifically to a filter layer: a schema validation middleware.

With a more focused view to this new architecture, the database can be considered as the principal foundation of a more complex whole: the database is unique but allows an undefined number of applications to retrieve information. A condition is however imposed in order to shift the consistency guarantee from the database to the server: exchanges should comply with a common standard. In the particular context of 3D city modelling, and keeping in mind the simplicity of use, applications and their exchanges should favor the CityJSON specifications.

Technically, the server filters all requests in both directions: from clients to the database and from the database to clients. This bi-directional filter allows storing and updating elements on the database by restricting them to predefined semantic information. The other way, it limits the information requested from the database depending on the users' right access, versions, etc. A view on the actual state of the database can be given thanks to the OGC API - Features exchange standard. Restricted to the read access, this view allows users to get generic information on the models elements. In summary, such type of filter can be used in order to implement security layers, versioning and above all to enclose the users' possibilities.

The consistency counterbalanced by the middleware implementation will open many possibilities in application development and digital twining. City stakeholders should benefit from a single data store that can be shared across all their activities and responsibilities. Without any limitations or compromises made on previous storage capabilities, the city models will become real integrating bases for all the city services activities.

Back to the introduction, the answer to the first question on the relevancy of a new generation is nuanced. It is important to provide a new solution for the management of a "unique and digital 3D replica of a city" that improves the applications flexibility but the usage of the traditional solutions is not outdated. An ecosystem based on several solutions should provide a relevant answer. At the same time a document-oriented solution for its flexibility and accessibility, a knowledge graph solution for the support of contextualization and semantic both linked to a relational solution, which has already demonstrated its capabilities in handling spatial methods, should meet current requirements and tackle future needs. The product resulting from the fusion of these storage modes could ideally take advantage of the benefits of each while attempting to offset their disadvantages. Therefore, few compromises should be made considering them in the very beginning of the conception rather than providing partial solutions on a succession of choices. We believe that this contribution makes a step further towards such a hybrid architecture. Shifting the storage paradigm should then not be seen as a complete reverse but rather as a more global vision that would allow reaching a better management of what "digital twins" are intend to be.

Remaining challenges could also be divided in improvements specific to the middleware and improvements specific to the vision of a unique replica and its contextualization. Developments should concern the identity of a feature through its lifecycle (creation, modifications, etc.). The middleware and its various schemas could suffer from a lack of management added to the unicity of the stored information. A dedicated study thus need to be conducted on the optimized way to identify city models and their elements. While CityJSON features are commonly identified by UUID or GML_ID, the Uniform Resource Identifier is freer in its use. However, it should be considered as a very relevant solution since an URI can take whatever shape needed as long as it provides a means of locating (on the web, not spatial). One can for instance create a formatted URI translating the identity of the data provider. The data responsibility could then be established and both documentation and support could be released in a very convenient manner.

Since we considered GraphQL and SHACL as related works, specific access methods could be developed to propose them as alternatives to our middleware and the OGC API services. Just like the latter, it would be normalized windows on the data stored by the provider no matter the users' habits. This consideration, alongside with the proposed usage of URIs, could lead to a hybrid storage solution based on document and graph oriented databases in which the identification of an object and its uniqueness would be guaranteed. We assume it will consist a good base to climb the Semantic Web Stacks: in our opinion, the principal requirement to reach the dreamed "Digital Twins". Finally, still with this objective in mind, data integration should also be one of the main future developments following this contribution. Regardless of the access method chosen, different levels of integration must be considered: Does the base model need to be enhanced? Should new attributes be created? Is this part of the data model's mission or should applications handle the integration themselves?

References

Allah Bukhsh, Zaharah, Marten van Sinderen, and P. M. Singh. 2015. 'SOA and EDA: A Comparative Study - Similarities, Differences and Conceptual Guidelines on Their Usage': Pp. 213–20 in *Proceedings* *of the 12th International Conference on e-Business*. Colmar, Alsace, France: SCITEPRESS - Science and and Technology Publications.

Baralis, Elena, Andrea Dalla Valle, Paolo Garza, Claudio Rossi, and Francesco Scullino. 2017. 'SQL versus NoSQL Databases for Geospatial Applications'. Pp. 3388–97 in. IEEE.

Bartoszewski, Dominik, Adam Piorkowski, and Michal Lupa. 2019. 'The Comparison of Processing Efficiency of Spatial Data for PostGIS and MongoDB Databases'. Pp. 291–302 in *Beyond Databases, Architectures and Structures. Paving the Road to Smart Data Processing and Analysis*. Vol. 1018, *Communications in Computer and Information Science*, edited by S. Kozielski, D. Mrozek, P. Kasprowski, B. Małysiak-Mrozek, and D. Kostrzewa. Cham: Springer International Publishing.

Biljecki, F. 2013. *The Concept of Level Detail in 3D City Models: PhD Research Proposal*. Delft University of Technology.

Biljecki, Filip, Kavisha Kumar, and Claus Nagel. 2018. 'CityGML Application Domain Extension (ADE): Overview of Developments'. *Open Geospatial Data, Software and Standards* 3(1):13. doi: 10.1186/s40965-018-0055-6.

Biljecki, Filip, Hugo Ledoux, and Jantien Stoter. 2016. 'An Improved LOD Specification for 3D Building Models'. *Computers, Environment and Urban Systems* 59:25–37. doi: 10.1016/j.compenvurbsys.2016.04.005.

Biljecki, Filip, Joie Lim, James Crawford, Diana Moraru, Helga Tauscher, Amol Konde, Kamel Adouane, Simon Lawrence, Patrick Janssen, and Rudi Stouffs. 2021. 'Extending CityGML for IFC-Sourced 3D City Models'. *Automation in Construction* 121:103440. doi: 10.1016/j.autcon.2020.103440.

Brito, Gleison, Thais Mombach, and Marco Tulio Valente. 2019. 'Migrating to GraphQL: A Practical Assessment'. Pp. 140–50 in *2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. Hangzhou, China: IEEE.

Cha, Sang Kyun, Ki Hong Kim, Chang Bin Song, Joo Kwan Kim, and Yong Sik Kwon. 1999. 'A Middleware Architecture for Transparent Access to Multiple Spatial Object Databases'. Pp. 267–82 in *Interoperating Geographic Information Systems*, edited by M. Goodchild, M. Egenhofer, R. Fegeas, and C. Kottman. Boston, MA: Springer US.

Chadzynski, Arkadiusz, Nenad Krdzavac, Feroz Farazi, Mei Qi Lim, Shiying Li, Ayda Grisiute, Pieter Herthogs, Aurel von Richthofen, Stephen Cairns, and Markus Kraft. 2021. 'Semantic 3D City Database — An Enabler for a Dynamic Geospatial Knowledge Graph'. *Energy and AI* 6:100106. doi: 10.1016/j.egyai.2021.100106.

Chaturvedi, Kanishk, Carl Stephen Smyth, Gilles Gesquière, Tatjana Kutzner, and Thomas H. Kolbe. 2017. 'Managing Versions and History Within Semantic 3D City Models for the Next Generation of CityGML'. Pp. 191–206 in *Advances in 3D Geoinformation*, edited by A. Abdul-Rahman. Cham: Springer International Publishing.

Clemens, Portele, Vretanos Panagiotis, and Heazel Charles. 2019. ‘OGC API - Feature - Part 1: Core’.

Corman, Julien, Juan L. Reutter, and Ognjen Savković. 2018. ‘Semantics and Validation of Recursive SHACL’. Pp. 318–36 in *The Semantic Web – ISWC 2018*. Vol. 11136, *Lecture Notes in Computer Science*, edited by D. Vrandečić, K. Bontcheva, M. C. Suárez-Figueroa, V. Presutti, I. Celino, M. Sabou, L.-A. Kaffee, and E. Simperl. Cham: Springer International Publishing.

da Costa Rainho, Filipe, and Jorge Bernardino. 2018. ‘Web GIS: A New System to Store Spatial Data Using GeoJSON in MongoDB’. Pp. 1–6 in *2018 13th Iberian Conference on Information Systems and Technologies (CISTI)*. Caceres: IEEE.

Debruyne, Christophe, and Kris McGlinn. 2021. ‘Reusable SHACL Constraint Components for Validating Geospatial Linked Data’. in *Reusable SHACL Constraint Components for Validating Geospatial Linked Data*. CEUR.

Deininger, Martina E., Maximilian von der Grün, Raul Piepereit, Sven Schneider, Thunyathep Santhanavanich, Volker Coors, and Ursula Voß. 2020. ‘A Continuous, Semi-Automated Workflow: From 3D City Models with Geometric Optimization and CFD Simulations to Visualization of Wind in an Urban Environment’. *ISPRS International Journal of Geo-Information* 9(11):657. doi: 10.3390/ijgi9110657.

Eriksson, Helen, and Lars Harrie. 2021. ‘Versioning of 3D City Models for Municipality Applications: Needs, Obstacles and Recommendations’. *ISPRS International Journal of Geo-Information* 10(2):55. doi: 10.3390/ijgi10020055.

Eriksson, Helen, Jing Sun, Väino Tarandi, and Lars Harrie. 2021. ‘Comparison of Versioning Methods to Improve the Information Flow in the Planning and Building Processes’. *Transactions in GIS* 25(1):134–63. doi: 10.1111/tgis.12672.

Fricke, Andreas, Jürgen Döllner, and Hartmut Asche. 2018. ‘Servicification – Trend or Paradigm Shift in Geospatial Data Processing?’ Pp. 339–50 in *Computational Science and Its Applications – ICCSA 2018*. Vol. 10962, *Lecture Notes in Computer Science*, edited by O. Gervasi, B. Murgante, S. Misra, E. Stankova, C. M. Torre, A. M. A. C. Rocha, D. Taniar, B. O. Apduhan, E. Tarantino, and Y. Ryu. Cham: Springer International Publishing.

Gómez, Paola, Claudia Roncancio, and Rubby Casallas. 2021. ‘Analysis and Evaluation of Document-Oriented Structures’. *Data & Knowledge Engineering* 134:101893. doi: 10.1016/j.datak.2021.101893.

Gröger, Gerhard, and Lutz Plümer. 2012. ‘CityGML – Interoperable Semantic 3D City Models’. *ISPRS Journal of Photogrammetry and Remote Sensing* 71:12–33. doi: 10.1016/j.isprsjprs.2012.04.004.

Haas, L., Renée J. Miller, B. Niswonger, M. Roth, P. Schwarz, and E. Wimmers. 1999. ‘Transforming Heterogeneous Data with Database Middleware: Beyond Integration’. *IEEE Data Engineering Bulletin* 22:31–36.

Hartig, Olaf, and Jorge Pérez. 2018. 'Semantics and Complexity of GraphQL'. Pp. 1155–64 in *Proceedings of the 2018 World Wide Web Conference on World Wide Web - WWW '18*. Lyon, France: ACM Press.

Herrmann, Kai, Hannes Voigt, Andreas Behrend, Jonas Rausch, and Wolfgang Lehner. 2017. 'Living in Parallel Realities -- Co-Existing Schema Versions with a Bidirectional Database Evolution Language'. *Proceedings of the 2017 ACM International Conference on Management of Data* 1101–16. doi: 10.1145/3035918.3064046.

Herrmann, Kai, Hannes Voigt, Jonas Rausch, Andreas Behrend, and Wolfgang Lehner. 2018. 'Robust and Simple Database Evolution'. *Information Systems Frontiers* 20(1):45–61. doi: 10.1007/s10796-016-9730-2.

Huang, Weiming, Syed Amir Raza, Oleg Mirzov, and Lars Harrie. 2019. 'Assessment and Benchmarking of Spatially Enabled RDF Stores for the Next Generation of Spatial Data Infrastructure'. *ISPRS International Journal of Geo-Information* 8(7):310. doi: 10.3390/ijgi8070310.

Knublauch, Holger, and Dimitris Kontokostas. 2017. *Shapes Constraint Language (SHACL)*. W3C.

Kulawiak, Marcin, Agnieszka Dawidowicz, and Marek Emanuel Pacholczyk. 2019. 'Analysis of Server-Side and Client-Side Web-GIS Data Processing Methods on the Example of JTS and JSTS Using Open Data from OSM and Geoportal'. *Computers & Geosciences* 129:26–37. doi: 10.1016/j.cageo.2019.04.011.

Kutzner, Tatjana, Kanishk Chaturvedi, and Thomas H. Kolbe. 2020. 'CityGML 3.0: New Functions Open Up New Applications'. *PFG – Journal of Photogrammetry, Remote Sensing and Geoinformation Science*. doi: 10.1007/s41064-020-00095-z.

Ledoux, Hugo, Ken Arroyo Ohori, Kavisha Kumar, Balázs Dukai, Anna Labetski, and Stelios Vitalis. 2019. 'CityJSON: A Compact and Easy-to-Use Encoding of the CityGML Data Model'. *ArXiv:1902.09155 [Cs]*.

Li, Dingding, Wande Chen, Mingming Pan, He Li, Hai Liu, and Yong Tang. 2018. 'DBHUB: A Lightweight Middleware for Accessing Heterogeneous Database Systems'. Pp. 408–19 in *Cloud Computing and Security*. Vol. 11063, *Lecture Notes in Computer Science*, edited by X. Sun, Z. Pan, and E. Bertino. Cham: Springer International Publishing.

Malinverni, Eva Savina, Berardo Naticchia, Jose Luis Lerma Garcia, Alban Gorreja, Joaquin Lopez Uriarte, and Francesco Di Stefano. 2020. 'A Semantic Graph Database for the Interoperability of 3D GIS Data'. *Applied Geomatics*. doi: 10.1007/s12518-020-00334-3.

Mei Qi, Lim, Wang Xiaonan, Inderwildi Oliver R., and Kraft Markus. 2021. *The World Avatar - a World Model for Facilitating Interoperability*. 277. Cambridge.

Nguyen, S. H., and T. H. Kolbe. 2020. 'A MULTI-PERSPECTIVE APPROACH TO INTERPRETING SPATIO-SEMANTIC CHANGES OF LARGE 3D CITY MODELS IN CITYGML USING A GRAPH

DATABASE'. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences* VI-4/W1-2020:143–50. doi: 10.5194/isprs-annals-VI-4-W1-2020-143-2020.

Nys, Gilles-Antoine, and Roland Billen. 2021. 'From Consistency to Flexibility: A Simplified Database Schema for the Management of CityJSON 3D City Models'. *Transactions in GIS* tgis.12807. doi: 10.1111/tgis.12807.

Nys, Gilles-Antoine, Abderrazzaq Kharroubi, Florent Poux, and Roland Billen. 2021. 'AN EXTENSION OF CITYJSON TO SUPPORT POINT CLOUDS'. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLIII-B4-2021:301–6. doi: 10.5194/isprs-archives-XLIII-B4-2021-301-2021.

Nys, Gilles-Antoine, Florent Poux, and Roland Billen. 2020. 'CityJSON Building Generation from Airborne LiDAR 3D Point Clouds'. *ISPRS International Journal of Geo-Information* 9(9):521. doi: 10.3390/ijgi9090521.

Patrick, Cozzi, Lilley Sean, and Getz Gabby. 2019. '3D Tiles Specification 1.0'.

Rasheed, Adil, Omer San, and Trond Kvamsdal. 2019. 'Digital Twin: Values, Challenges and Enablers'. *ArXiv:1910.01719 [Eess]*.

Rossknecht, Maxim, and Enni Airaksinen. 2020. 'Concept and Evaluation of Heating Demand Prediction Based on 3D City Models and the CityGML Energy ADE—Case Study Helsinki'. *ISPRS International Journal of Geo-Information* 9(10):602. doi: 10.3390/ijgi9100602.

Roy-Hubara, Noa, and Arnon Sturm. 2020. 'Design Methods for the New Database Era: A Systematic Literature Review'. *Software and Systems Modeling* 19(2):297–312. doi: 10.1007/s10270-019-00739-8.

Schrotter, Gerhard, and Christian Hürzeler. 2020. 'The Digital Twin of the City of Zurich for Urban Planning'. *PFG – Journal of Photogrammetry, Remote Sensing and Geoinformation Science* 88(1):99–112. doi: 10.1007/s41064-020-00092-2.

Schultz, Carl, and Mehul Bhatt. 2013. 'InSpace3D: A Middleware for Built Environment Data Access and Analytics'. *Procedia Computer Science* 18:80–89. doi: 10.1016/j.procs.2013.05.171.

Shahat, Ehab, Chang T. Hyun, and Chunho Yeom. 2021. 'City Digital Twin Potentials: A Review and Research Agenda'. *Sustainability* 13(6):3386. doi: 10.3390/su13063386.

Sutanta, E., and E. K. Nurnawati. 2019. 'The Design of Relational Database for Multipurpose WebGIS Applications'. *Journal of Physics: Conference Series* 1413:012029. doi: 10.1088/1742-6596/1413/1/012029.

Vera-Olivera, Harley, Ruizhe Guo, Ruben Cruz Huacarpuma, Ana Paula Bernardi Da Silva, Ari Melo Mariano, and Holanda Maristela. 2021. 'Data Modeling and NoSQL Databases - A Systematic Mapping Review'. *ACM Computing Surveys* 54(6):1–26. doi: 10.1145/3457608.

Virtanen, Juho-Pekka, Kaisa Jaalama, Tuulia Puustinen, Arttu Julin, Juha Hyyppä, and Hannu Hyyppä. 2021. 'Near Real-Time Semantic View Analysis of 3D City Models in Web Browser'. *ISPRS International Journal of Geo-Information* 10(3):138. doi: 10.3390/ijgi10030138.

Visconti, Ennio, Christos Tsigkanos, Zhenjiang Hu, and Carlo Ghezzi. 2021. 'Model-Driven Engineering City Spaces via Bidirectional Model Transformations'. *Software and Systems Modeling* 20(6):2003–22. doi: 10.1007/s10270-020-00851-0.

Vitalis, S., A. Labetski, K. Arroyo Ohori, H. Ledoux, and J. Stoter. 2019. 'A DATA STRUCTURE TO INCORPORATE VERSIONING IN 3D CITY MODELS'. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* IV-4/W8:123–30. doi: 10.5194/isprs-annals-IV-4-W8-123-2019.

Vitalis, Stelios, Ken Arroyo Ohori, and Jantien Stoter. 2020. 'CityJSON in QGIS: Development of an Open-source Plugin'. *Transactions in GIS* tgis.12657. doi: 10.1111/tgis.12657.

Vitalis, Stelios, Ken Ohori, and Jantien Stoter. 2019. 'Incorporating Topological Representation in 3D City Models'. *ISPRS International Journal of Geo-Information* 8(8):347. doi: 10.3390/ijgi8080347.

Wagemann, Julia, Oliver Clements, Ramiro Marco Figuera, Angelo Pio Rossi, and Simone Mantovani. 2018. 'Geospatial Web Services Pave New Ways for Server-Based on-Demand Access and Processing of Big Earth Data'. *International Journal of Digital Earth* 11(1):7–25. doi: 10.1080/17538947.2017.1351583.

Widl, Edmund, Giorgio Agugiaro, and Jan Peters-Anders. 2021. 'Linking Semantic 3D City Models with Domain-Specific Simulation Tools for the Planning and Validation of Energy Applications at District Level'. *Sustainability* 13(16):8782. doi: 10.3390/su13168782.

Wittern, Erik, Alan Cha, James C. Davis, Guillaume Baudart, and Louis Mandel. 2019. 'An Empirical Study of GraphQL Schemas'. Pp. 3–19 in *Service-Oriented Computing*. Vol. 11895, *Lecture Notes in Computer Science*, edited by S. Yangui, I. Bouassida Rodriguez, K. Drira, and Z. Tari. Cham: Springer International Publishing.

Wong, S. H., S. L. Swartz, and D. Sarkar. 2002. 'A Middleware Architecture for Open and Interoperable GISs'. *IEEE Multimedia* 9(2):62–76. doi: 10.1109/93.998065.

Yao, Zhihang, Claus Nagel, Felix Kunde, György Hudra, Philipp Willkomm, Andreas Donaubauer, Thomas Adolphi, and Thomas H. Kolbe. 2018. '3DCityDB - a 3D Geodatabase Solution for the Management, Analysis, and Visualization of Semantic 3D City Models Based on CityGML'. *Open Geospatial Data, Software and Standards* 3(1). doi: 10.1186/s40965-018-0046-7.

Zlatanova, Sisi, and Jantien Stoter. 2006. 'The Role of DBMS in the New Generation GIS Architecture'. Pp. 155–80 in *Frontiers of Geographic Information Technology*, edited by S. Rana and J. Sharma. Berlin/Heidelberg: Springer-Verlag.