UPORABA PODATKOVNE BAZE NOSQL NA PODROČJU 3D-KATASTRA

USING NOSQL DATABASES IN THE 3D CADASTRE DOMAIN

Nenad Višnjevac, Rajica Mihajlović, Mladen Šoškić, Željko Cvijetinović, Branislav Bajat

UDK: 528.4
Klasifikacija prispevka po COBISS.SI: 1.01
Prispelo: 17. 3. 2017
Sprejeto: 9. 8. 2017

DOI: 10.15292//geodetski-vestnik.2017.03.412-426
SCIENTIFIC ARTICLE
Received: 17. 3. 2017
Accepted: 9. 8. 2017

IZVLEČEK

3D-kataster zahteva podatkovne modele, ki so kompleksnejši od podatkovnih modelov za katastre dveh razsežnosti in ki omogočajo vzpostavitev obsežnejših podatkovnih zbirk. Podatki 3D-katastra bi morali biti organizirani v okviru sistema za upravljanje podatkovnih zbirk, ki bi omogočal popolnost in doslednost pri shranjevanju in vzdrževanju katastrskih podatkov. Sistem za upravljanje relacijskih podatkovnih zbirk temelji na tabelarični strukturi, podatki so shranjeni v predhodno določenih kolonah, s prav tako predhodno določenimi vrstami podatkov, kar je nekoliko neugodno za prostorske podatkovne zbirke v 3D razsežnostih. Študija je namenjena preučitvi možnosti uporabe podatkovnih zbirk NoSQL na področju 3D-katastra. Podatkovne zbirke NoSQL omogočajo shranjevanje nestrukturiranih podatkov, brez predhodne določitve vrste ali razreda podatka, kar smo preizkusili kot možnost za vzpostavitev 3D-katastra. Uporabili smo MongoDB in 3D-katastrski podatkovni model. Pripravili smo 3D-katastrske podatke, tako alfa-numerične kot grafične, ki so bili namenjeni uvozu, shranjevanju, upravljanju, poizvedovanju in posodabljanju v podatkovni bazi NoSQL. Shranjene podatke v podatkovni zbirki MongoDB smo grafično predstavili in po podatkih poizvedovali s spletnim brskalnikom, pri čemer smo uporabili Cesium knjižnico.

KLJUČNE BESEDE

3D-kataster, podatkovna zbirka NoSQL, MongoDB, RDBMS, 3D-vizualizacija, Cesium

ABSTRACT

The 3D cadastre concept brings data models, which are more complex than traditional 2D cadastral data models and could be followed by a large amount of data. The 3D cadastral data should be stored in database management systems, since the cadastral data integrity and consistency have to be satisfied. Relational database management system requires a tabular structure where data are stored within predefined columns and data types, and this could be uncomfortable for 3D cadastre until relational provides full 3D support. This study examines the possibility of using NoSQL databases in the 3D cadastre domain. NoSQL database stores unstructured data, which means that it is not required to define in advance what data types and categories will be used. From the 3D cadastre point of view, the NoSQL approach provides flexibility in data types and allows easier implementation of the 3D cadastral models. The implementation is conducted by using MongoDB and 3D Cadastral Data Model, where 3D cadastral data, including both alphanumerical and geometry part of data, are prepared for importing and then stored, managed, queried, and updated within NoSQL database. Furthermore, data stored in MongoDB are visualized and queried inside a web browser by using Cesium library.

KEY WORDS

3D Cadastre, NoSQL database, MongoDB, RDBMS, 3D visualization, Cesium

1 INTRODUCTION

During the last century, rapid urbanization created significant environmental changes that are mostly reflected by increases in densely populated urban areas. As a result, a space below and above the ground level is occupied and used by complex constructions. These changes have caused a greater importance for representation of the ownership. Traditional cadastral registration systems face many difficulties in representing complex and multilevel property situations on 2D maps (Stoter & Salzmann, 2003). Study by Marcin (2012) illustrated the difficulties in representing 3D properties on 2D maps. Additionally, Navratil (2011) stated that in 2D managed cadastre system, it is not possible to model horizontally divided situations, like in the case with the floors shared by different owners. As a result, these maps cannot be used as a base for registration of complex situations and interlocked rights in urban areas. Therefore, a system which allows easy recording and representing of all possible property situations is necessitated and such a system is called "3D cadastre". A 3D cadastre contains data of land, buildings, building parts, other construction objects, and data of the owners of such objects - including rights, restrictions, and responsibilities (RRR).

There are many reasons why the idea of the 3D cadastre is becoming increasingly widely recognized. Nowadays, more than half of the world's population lives in urban areas (www.un.org, 2014), which has resulted in an increased number of interlocked properties, complex constructions, buildings above roads, tunnels, pipelines, etc. As a consequence, current cadastral registration systems have difficulties in handling the visualization and registration of all property situations in urban areas, both methodologically and technologically (Drobež et al., 2017). Additionally, a rapid development of new technologies, especially 3D technologies, brought new possibilities of storing, manipulating, and visualizing 3D data. This initiated and enabled the development of the 3D cadastre technical approaches as well. Finally, property values increased; for example, the property values in some of the developed countries have experienced large increases of more than 200 percent from 1985 to 2006 (Goodhart & Hofmann, 2008), and all these high-valued properties, including complex structures, are represented by 2D drawings despite the fact that from the juridical point of view cadastral registration has always been 3D. Therefore, higher costs of 3D cadastre implementation can be justified by higher property values. 3D Cadastre, together with 3D city models, can improve land tax estimation and the land market by using visibility analysis (Navratil & Fogliaroni, 2014)

The 3D cadastre can be viewed from two aspects, technological and legal one. From the legal point of view, the 3D cadaster, or a system which will enable to unequivocally register rights on 3D properties, is very important. Paasch, et al. (2016) discussed the legal aspects of 3D real property, while another study, by van Oosterom (2013), describes the current research and development in 3D cadastre area altogether. In our study, we consider only the technical and technological aspects of the 3D cadastre.

According to Stoter & van Oosterom (2002), geo-database management systems (DBMS) should be the starting point for storing and maintaining data on 3D cadastre. DBMS guarantee the consistency of the data (both geometric and alphanumeric), which provides that data are modified only in allowed ways. This is very important for the cadastral data, especially for the legal part of data.

Three possible conceptual data models for the 3D cadastre were described by Stoter & Salzmann (2003). Based on the hybrid and full 3D cadastre models, it could be concluded that 3D cadastral models are

much more complex than the traditional 2D cadastral models and could result in large and complex datasets. 3D data manipulation, such as storing, processing and analyzing, presents challenges to relational database management system (RDBMS) due to the lack of full 3D support. Handling large datasets became a very big issue for RDBMS and the solution for this problem was found in scalability and data distribution where data are distributed over several servers. All this makes the whole system more complex and hence affects the operation's performance (Mohamed, et al., 2014). The 3D cadastral datasets have to be divided based on administrative units, and have to wait for further development of RDBMS to fully support the 3D cadastre requirements.

The objective of this study is to explore an alternative way of storing 3D cadastral data, with no intention to substitute the use of relational databases. Starting from the notion that 3D cadastral data should be stored in a DBMS and due to the limited 3D support in RDBMS including issues with large datasets, we analyze the use of alternative database systems called NoSQL in 3D cadastre domain. NoSQL databases are database management softwares that can handle massive data storage and they follow the BASE (Basically Available, Soft State, Eventual consistency) rules.

Section 2 summarizes the complexity of the current 3D cadastral models and the capability of relational DBMS to store 3D cadastral data. Section 3 provides a short analysis of the NoSQL databases based on comparison with the relational databases. In Section 4, we present implementation of the case study based on MongoDB and 3D Cadastral Data Model (3DCDM). Moreover, this section considers the conversion of 3DCDM GML data to JSON documents, their storage in NoSQL database, and querying, selecting and updating data within the database. In addition, 3D cadastral data stored in MongoDB are visualized and queried inside a web browser by using the Cesium library. The conclusions and directions for future work are given in Section 5.

2 3D CADASTRE AND RDBMS

The 3D cadastre could be implemented in different ways; from a full 3D cadastre, which includes 3D volumes, to the current 2D Cadastre models with tags leading to the external 3D models. The hybrid and full 3D cadastre models, together with 3DCDM proposed by Aien et al. (2013) show that operative 3D cadastral models will be much more complex than traditional 2D Cadastre models. Consequently, 3D cadastre models will store much more data.

DBMSs play a central role when it comes to storing 3D cadastral data. Within the RDBMS architecture, the spatial and non-spatial data are maintained and protected by strongly following the ACID (Atomicity, Consistency, Isolation, and Durability) rules that provide a reliable system and data consistency. Certain standards, such as CityGML (Gröger & Plümer, 2012), are very useful for the visualization and exchange of 3D models including topological relationships between different geometric aggregates (*see* Li, et al., 2016). However, they are still only XML schemas that can be easily corrupted or modified and this is not acceptable from the data consistency point of view. Therefore, the 3D cadastral data should be stored in a DBMS (Stoter & Salzmann, 2003; Lemmen & van Oosterom, 2013).

Mainstream DBMS softwares, such as Postgres/PostGIS, Microsoft SQL Server, and Oracle, have provided spatial data types and spatial operators that offer support for cadastral data. However, until recently, they

have not provided support for 3D data managing (*see* Schön, et al., 2009). With the Oracle Database Release 11.1, three-dimensional spatial data that include 3D points, 3D lines, 3D polygons, 3D surfaces, and solids can be stored with built-in datatypes. Also, PostGIS supports 3D data types such as points, lines, surfaces, TIN and solids. When it comes to topological relations, no geo-DBMS can provide support in maintaining 3D topological models (Breunig & Zlatanova, 2011). This means that the 3D spatial functionality, regarding complex 3D spatial data types, must be further defined. These advances should be developed (or extended), in order to maintain 3D objects in the database. The authors have also proposed that as soon as 3D geometry and topological models are supported, a DBMS should support functions and operations for analysis, conversion, and validity. According to these statements, there is currently no RDBMS which fully supports all 3D cadastre requirements.

Furthermore, relational databases were created for business data processing needs during the time when interfaces and hardware were different. Despite some changes, no RDBMS has been completely redesigned since that time (Stonebraker, et al., 2007). Therefore, if 3D cadastre models result in large amounts of data, this could potentially influence the handling of these data within RDBMS. Moreover, relational databases could be beaten in almost any area outside of business data processing market, if it is worth to invest in a specialized engine.

3 NOSQL DATABASES

NoSQL databases represent an alternative to the traditional relational databases. They represent the database management systems that respect three BASE principles: 1) Basically Available, 2) Soft State, and 3) Eventual consistency (Nayak, et al., 2013; Chandra, 2015). They are not based on the strict schema, and generally do not support SQL query language for data manipulation. The NoSQL databases were initially developed for the needs of companies that deal with big data and real-time application, such as Google, Yahoo, Facebook, (Mohan, 2013) but now they are used in other domains.

As Chandra (2015) has described, NoSQL database are classified into 2 categories:

— Aggregate Oriented
— Non-aggregate Oriented

Aggregate Oriented NoSQLs collect information from various nodes in the network cluster. Non-aggregate Oriented NoSQLs are the superset of Aggregate Oriented NoSQL, which heavily use relations. Hence, the difference between these two categories is that Aggregate Oriented databases divide relations.

Also, NoSQL databases use different data storage, and based on this criterion they can be categorized into 5 different types (Nayak, et al., 2013; Chandra, 2015; Moniruzzaman & Hossain, 2013):

1. *Key-value store* databases represent the most basic type of NoSQL database where data items are stored as keys. Data items are converted into keys by using a formula and when the data are queried, the key is converted back to the location of the stored data (Chandra, 2015). These databases are very similar to hash tables, and they provide fast querying of a big data storage (Nayak, et al., 2013).

2. *Column-Oriented* databases provide a multi-dimensional sorted map for storing data where each record can be stored in a different number of fields. Fields can be grouped in a way to create a column set. Data

is retrieved by a key which is associated with one column or column set. These databases are good for data mining and analytic applications (Nayak, et al., 2013).

3. *Document* databases store data as documents, which means that all the data are stored in formats such as JSON or XML. These databases can store same, similar, or dissimilar data. Each stored document gets a unique key which is used for retrieving the document (Nayak, et al., 2013). Documents can also be retrieved by any attribute within the document, including the ability to query and update values inside the document. Some databases, such as MongoDB, provide the index options, including text indexes, geospatial indexes, unique indexes, and others (website-mongoDB).

4. *Graph* databases, store data as a graph where nodes, edges and properties represent data. As Chandra (2015) has concluded, graph databases are suitable for the Location-based services and other complex network-based applications.

5. *Object-Oriented* databases store data as an object and they can be treated as a combination of database principles and object-oriented programming (Nayak, et al., 2013).

There are several studies that have compared NoSQL databases to the relational databases (Mohamed, et al., 2014; Nance, et al., 2013; Nayak, et al., 2013; Tauro, S, & A.B., 2012). Based on these publications, it can be concluded that NoSQL databases have advantages and disadvantages when compared to the relational databases—RDBMS.

Main advantages of NoSQL databases are flexibility and not strict data models which enables easy storage of different data types and allows the system to change and evolve. NoSQL databases have capabilities to store structured, semi-structured, and even unstructured data, unlike RDBMS where data have to fit into the predefined table structure. It means that it is possible to store data based on two or more data models within the NoSQL database, and this is very suitable for hybrid models. NoSQL databases are primary designed to handle big data, compared to RDBMS, where large amount of data may slow down their performance. NoSQL databases have implemented methods that improve the performance of storing and retrieving data (Mohamed, et al., 2014).

However, NoSQL databases are still immature systems, and this is the reason why many large institutions will not transfer their systems from relational databases to NoSQL databases (Nance, et al., 2013). Furthermore, they do not use standard query language. Most of the NoSQL database providers have offered different query language and the introduction of standard query language for NoSQL systems will be a big step forward (Nayak, et al., 2013). There is no unique standard user interface offered by different providers, and there is also a lack of database design tools for NoSQL databases with complex data models (Mohan, 2013). Moreover, a big disadvantage is that some of the NoSQL systems do not follow the ACID rules, and BASE rules provide less strict assurance than ACID. This is important for 3D cadastre, in the case that transactions are related to legal aspect of 3D cadastre, when high data consistency is required. Also, NoSQL databases depend on backup files when it comes to crash recovery, while some of them, such as MangoDB, provide Journal file as crash recovery (Mohamed, et al., 2014).

With regards to spatial data, a few NoSQL databases such as MongoDB, Neo4j, Bigtable, Cassandra, etc. provide basic spatial operators and spatial indexes, but there are still no 3D operators. This might be a disadvantage when compared to RDBMS due to the fact that it provides some of the 3D operators.

Nance et al., (2013) concluded that many areas can benefit from using NoSQL databases since they could be used and involved in many different platforms. Relational databases will be used in the future because NoSQL databases and relational databases do not tend to solve the same problems. In other words, there is room for the development of both approaches. For the purpose of this research, 3DCDM data are stored, queried, and updated within a NoSQL document database called MongoDB.

4 3D CADASTRE DATA IN NOSQL DATABASE

In this study, some elements of 3DCDM described formerly by Aien, et al. (2014) were used. Data are stored, managed, and processed within MongoDB, which is free and is an open-source NoSQL database. MongoDB is the most popular NoSQL database and the fourth most popular DBMS system (db-engines.com, 2016). The NoSQL Viewer application was used as a user interface.

The described physical data model of the 3DCDM has been developed as a schema of the GML language (Geography Markup Language). The GML is the XML grammar developed for geographical features by the Open Geospatial Consortium (OGC). According to this, the described 3DCDM physical data model is an XML-based schema. MongoDB stores JSON-like documents, hence for the purpose of this research the 3DCDM physical data model is converted from the XML-based schema to JavaScript Object Notation (JSON). This can be easily performed by many applications and online converters; for example, the conversion in this study was performed using codebeautify.org.

Aien, et al. (2014) defined eleven XML namespaces where each 3DCDM module has a namespace and each namespace is connected to an URI and has a suggested prefix. As a case study, we used three 3DCDM modules including: Building, Terrain and Utility modules. To elaborate, we used eight building units (including two underground units), TIN terrain model and data on two utility networks (water supply and hot water network). An example of a physical model (XML document) for two TIN triangles is as follows:

```
<physicalModel>
<UrbanModel>
<physicalObjectMember>
<trn:Terrain gml:id="TIN_1">
<trn:terrainSource>
<trn:TIN gml:id="TIN-1">
<trn:tinSource>
<gml:TriangulatedSurface gml:id="DCDM_tinSurface_1">
<gml:trianglePatches>
<gml:Triangle>
<gml:exterior>
<gml:LinearRing>
<gml:posList srsDimension="3" count="5">998.417 1018 0 1000.04 1033 3.29259 1000.75
1045.08 6.15357 998.417 1018 0</gml:posList>
</gml:LinearRing>
</gml:exterior>
</gml:Triangle>
<gml:Triangle>
<gml:exterior>
<gml:LinearRing>
```

```
<gml:posList srsDimension="3" count="5">1070.83 1039.07 6.57358 1076.54 1038.65
7.28319 1073.98 1046.35 7.9522 1070.83 1039.07 6.57358</gml:posList>
</gml:LinearRing>
</gml:exterior>
</gml:Triangle>
</gml:trianglePatches>
</gml:TriangulatedSurface>
</trn:tinSource>
</trn:TIN>
</trn:terrainSource>
</trn:Terrain>
</physicalObjectMember>
</UrbanModel>
</physicalModel>
```

The first step was converting the case study data from XML schema to JSON. The conversion was performed for all case study data: the terrain model and eight building units including two basements and two utility networks. An example of converted data (two TIN triangles) is as follows:


```
{"physicalModel": {
"UrbanModel": {
"physicalObjectMember": {
"trn:Terrain": {
"-gml:id": "TIN_1",
"trn:terrainSource": {
"trn:TIN": {
 "-gml:id": "TIN-1",
  "trn:tinSource": {
   "gml:TriangulatedSurface": {
    "-gml:id": "DCDM_tinSurface_1",
     "gml:trianglePatches": {
      "gml:Triangle": [
       {
         "gml:exterior": {
          "gml:LinearRing": {
           "gml:posList": {
            "-srsDimension": "3",
            "-count": "5",
            "#text": "998.417 1018 0 1000.04 1033 3.29259 1000.75 1045.08 6.15357
            998.417 1018 0"
            }}}},
      {
         "gml:exterior": {
          "gml:LinearRing": {
           "gml:posList": {
            "-srsDimension": "3",
             "-count": "5",
              "#text": "1070.83 1039.07 6.57358 1076.54 1038.65 7.28319 1073.98
              1046.35 7.9522 1070.83 1039.07 6.57358"
              }}}}
                    ]
         }}}}}}}}}}
```

Hereafter, as the input data, we used data from the terrain model, building units and utilities converted to JSON documents. Figure 1 shows the terrain model which has a small valley prepared for building infrastructure and which is built from the 237 triangles.

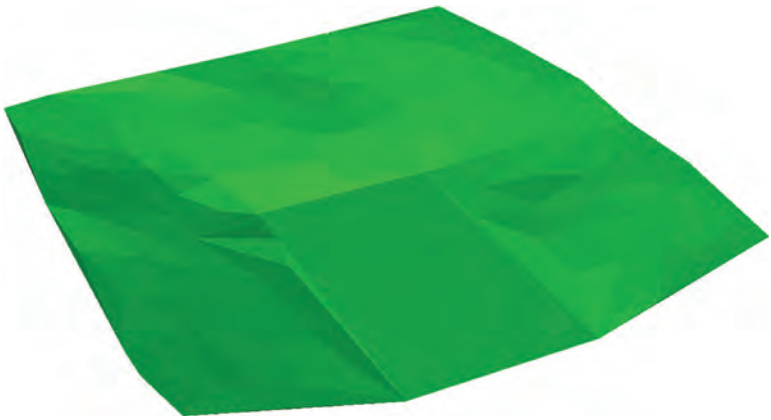

Figure 1: Visualization of the terrain model.

Figure 2 shows the terrain model and the building units. Figure 2a portrays six building units of six owners above ground whereby every building unit is colored to represent a different property holder. Figure 2b shows two underground building units — basements, and they are also colored according to property holder. Additionally, the water supply network (blue) and hot water network (red) are represented in Figure 2b. Both of them contain the main pipe and connecting pipe between the building and the main pipe.

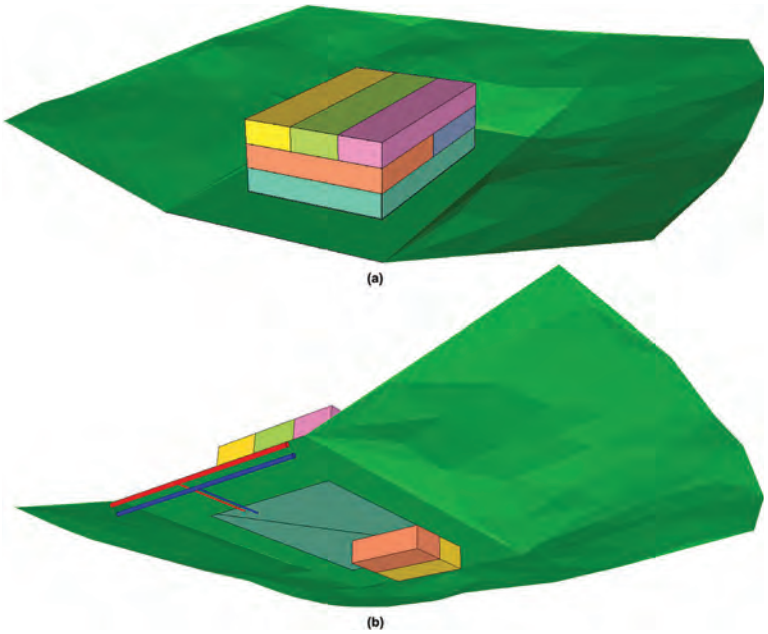

Figure 2: The terrain model and the building units. a) Six building units. b) Two underground units —basements and utilities.

The second step was to import data (JSON documents) to the MongoDB database. Imported documents are stored in the database called "3DCadastre" and within collections called "Terrain", "BuildingUnits" and "Utilities". Collections in MongoDB represent the set of documents grouped together. A collection may store documents that are not the same in structure and it can be seen as an analogy to a table of an RDBMS. The syntax for creating the database and the "Terrain" collections, as well as for importing the terrain model is as follows:

```
mongoimport --db 3DCadastre --collection Terrain --type json --file D:\3DCadastre\
Terrain\TIN.json
```

Note that a new database and collection can be created on the fly as a part of an import function without the previous definition. The syntax for importing other JSON documents related to Building Units is the same, but it can be modified to import all documents at once.

```
FOR %i IN (D:\3DCadastre\BuildingUnits\*.json) DO mongoimport --db 3DCadastre --col-
lection BuildingUnits --type json --file %i
```

By using just this simple syntax, many documents and 3D cadastral data can be easily imported and stored within the MongoDB database. Once imported, it is ready for querying, managing, and processing. Similar to RDBMS, data can be queried to define the conditions that need to be satisfied by returning documents. Additionally, it is possible to indicate the fields that should be returned including all or just the specified ones. Furthermore, it allows sorting options to be defined using extra logic. An example of returning the whole document that stores data on "Unit1" is as follows:


```
use 3DCadastre
db.BuildingUnits.find({ "UrabnCadastralModel.name.__text": "Unit1" })
```

Querying only specified fields (such as owner of the "Unit2") can be done by the following syntax:

```
use 3DCadastre
db.BuildingUnits.find(  { "UrabnCadastralModel.name.__text": "Unit2" },
{_id:0,"UrabnCadastralModel.physicalModel.UrbanModel.physicalObjectMember.Build-
ing.consistsOfBuildingPart.Unit.lpo.LegalPropertyObject.proprietor.InterestHold-
er._pName":1})
```

The query returns the following JSON document:

```
{
  "UrabnCadastralModel":{
    "physicalModel":{
      "UrbanModel":{
        "physicalObjectMember":{
          "Building":{
            "consistsOfBuildingPart":{
              "Unit":{
                "lpo":{
                  "LegalPropertyObject":{
                    "proprietor":{
                      "InterestHolder":{
                        "_pName":"Petar Markovic"
                      }}}}}}}}}}}
```

The query can be modified to return only the value, with additional options, including logic operators and sorting options. The following query returns owners of all units in ascending order by the owner's name:

```
use 3DCadastre
var myQuery = db.BuildingUnits.find( { }, {_id: 0, "UrabnCadastralModel.name.__text":
1,"UrabnCadastralModel.physicalModel.UrbanModel.physicalObjectMember.Building.
consistsOfBuildingPart.Unit.lpo.LegalPropertyObject.proprietor.InterestHolder._
pName":1}).sort( { "UrabnCadastralModel.physicalModel.UrbanModel.physicalObject-
Member.Building.consistsOfBuildingPart.Unit.lpo.LegalPropertyObject.proprietor.
InterestHolder._pName": 1 } )

myQuery.forEach(
function(myDoc) { print( "Owner of the " + myDoc.UrabnCadastralModel.name.__text +
" is " + myDoc.UrabnCadastralModel.physicalModel.UrbanModel.physicalObjectMember.
Building.consistsOfBuildingPart.Unit.lpo.LegalPropertyObject.proprietor.Interest-
Holder._pName ); }
);
```

The query returns the following results:

```
Owner of the Unit4 is Aleksa Pavlovic
Owner of the Unit5 is Jovana Petrovic
Owner of the Unit1 is Marko Jovanovic
.
.
.
```


From the cadastral point of view, the data update is very important. Data stored within MongoDB can be updated by using *db.collection.update()* function. The query that updates the owner name of Unit1 to "Janko Jankovic" is as follows:

```
use 3DCadastre
db.BuildingUnits.update({ "UrabnCadastralModel.name.__text": "Unit1"},
{$set: {"UrabnCadastralModel.physicalModel.UrbanModel.physicalObjectMember.Build-
ing.consistsOfBuildingPart.Unit.lpo.LegalPropertyObject.proprietor.InterestHolder._
pName": "Janko Jankovic" } } )
```

All these queries are connected to alphanumeric data without any spatial queries. Regarding spatial queries, MongoDB supports the following GeoJSON objects:

- Point
- LineString
- Polygon
- MultiPoint
- MultiLineString
- MultiPolygon
- GeometryCollection

Additionally, it supports inclusion queries that allow the selection of documents with geospatial data within a specified shape. Intersection queries are also supported and the $near operator is very useful for returning documents from "nearest to farthest" with regards to a specified point. MongoDB supports 2D geospatial indexes for both 2D plane data and geometries on a sphere. MongoDB does not currently support 3D geospatial indexing and 3D geospatial querying. However, some applications, such as 3DRepo (3drepo.org, 2017), use the MongoDB database for storing their 3D models.

The 3D geometry data can still be updated as coordinate pairs. The next query updates the terrain model by extending the existing TIN with inserting two additional triangles.

```
use 3DCadastre

var AdditionalTriangle1 =  {
                           "exterior":{
                             "LinearRing":{
                               "posList":{
                                 "__prefix":"gml",
                                  "__text":" 978 990 0 978 1018 0 998.417 990 0 978
990 0"},
                                "__prefix":"gml"},
                             "__prefix":"gml"},
                           "__prefix":"gml" }
var AdditionalTriangle2 =  {
                           "exterior":{
                             "LinearRing":{
                               "posList":{
                                 "__prefix":"gml",
                                  "__text":"998.417 990 0 978 1018 0 998.417 1018 0
998.417 990 0"},
                                "__prefix":"gml"},
                             "__prefix":"gml" },
                           "__prefix":"gml"}

db.Terrain.update({
    "UrabnCadastralModel.name.__text": "TERRAIN"},
    {
   $push : {    "UrabnCadastralModel.physicalModel.UrbanModel.physicalObjectMember.
Terrain.terrainSource.TIN.tinSource.TriangulatedSurface.trianglePatches.Triangle" :
{ $each: [AdditionalTriangle1, AdditionalTriangle2]}
    }})
```

The "valley" of the terrain model is extended by inserting these two additional triangles. Figure 3 shows the modified terrain model and the edges of the inserted triangles are represented in red.

Since MongoDB stores data as JSON documents, it can be easily integrated with Java Script libraries for 3D visualization or even for 3D operations. In this study, 3D Cadastral data stored in MongoDB are visualized and queried inside a web browser by using Cesium Java Script library. MongoDB provides REST (Representational State Transfer) API, and together with JSONP enables returning JSON objects from the database (server) based on client query. An example of returning a JSON object from the Mongo database can be found on the following link (osgl.grf.bg.ac.rs/mongo_rest, 2017). The received JSON object represents TIN data used in the case study.

Furthermore, received data are provided to Cesium library which visualizes interactive 3D primitives. Figure 4 shows a representation of an interface of a web application based on Cesium, for which query and read data are from MongoDB.

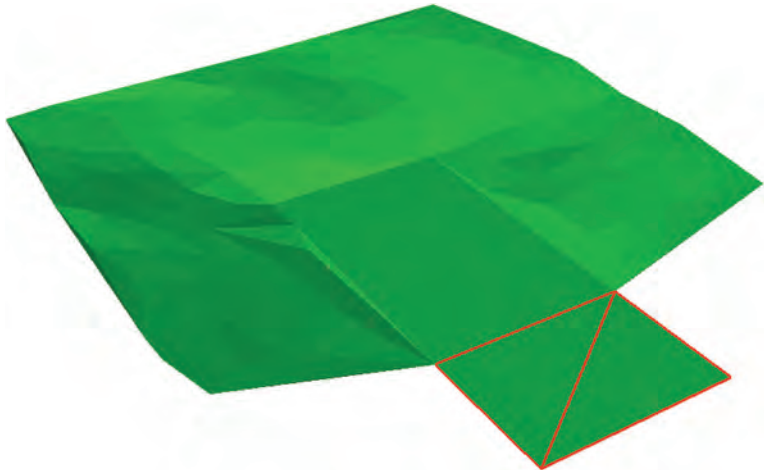

Figure 3: Visualization of the modified terrain model.

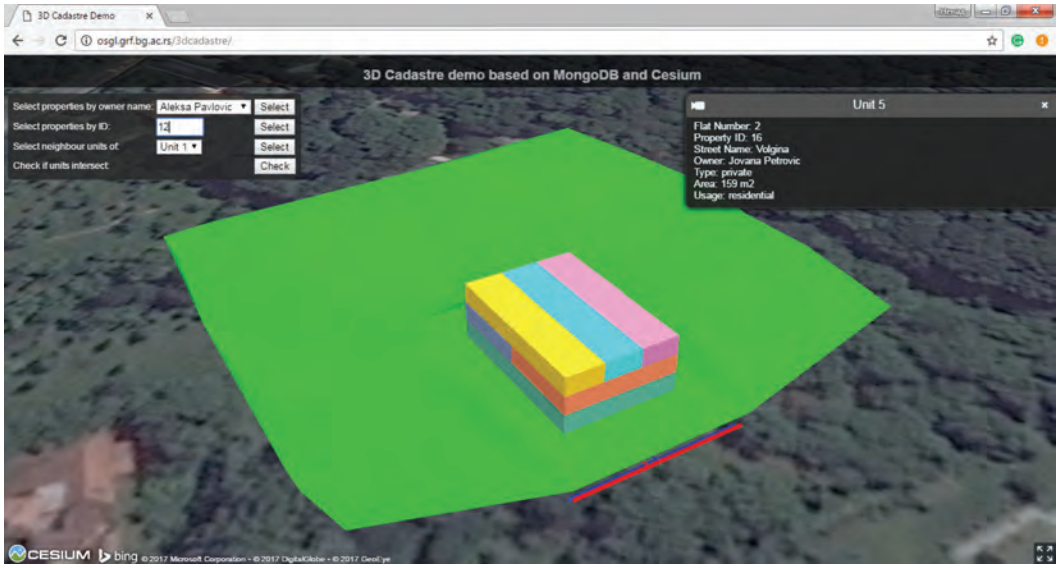

Figure 4: 3D cadastre demo based on MongoDB and Cesium.

The demo of the discussed 3D cadastre application is available at the web page URL: http://osgl.grf.bg.ac.rs/3dcadastre/ and it is optimized for the Google Chrome browser. Besides the visualization of 3D spatial data, the application provides interactive support and returns alphanumeric data stored in the database when the user clicks on a primitive. It also enables the selection and retrieval of property information based on owner name or property ID.

Regarding 3D operations, the CSG JavaScript library (github.com, 2017) was integrated into a demo application and it is used to support spatial queries. CSG (Constructive Solid Geometry) is a technique for solid modeling that uses Boolean operations to combine 3D solids. Two options are developed that

offer selecting neighbor units (touching primitives with 0.01 m tolerance) and checking if building units intersect. These operations can be used as an example of topology rules. Furthermore, topology rules can be used to check topological relationships of data in the database until DBMS extends support for 3D topological models. Checking similar topology rules on a large dataset by using external libraries should be performed on the server side; for example, node.js (JavaScript runtime environment) modules can be used rather than using JavaScript libraries on the client side. This will help to avoid sending large data sets over the network and calculations on the client side, and hence clients will get only the part of the dataset they want to see checked based on topology rules. The presented 3D cadastre demo is feasible for adaptation to use data stored in RDBMS such as Postgres/PostGIS.

MongoDB, as one of the most popular NoSQL databases, can be used for storing 3D cadastral data. Due to the schema that allows varying sets of fields, with different types selected for each field, MongoDB easily store 3D models in many different formats such as CityGML or 3DCDM model presented in this study. Merging those models with 2D cadastral data, it is possible to implement hybrid 3D cadastral models.

MongoDB supports only 2D GeoJSON data types, 2D indexing, and 2D spatial queries. This is the main disadvantage for managing 3D cadastral data, although some of the basic 3D queries could be performed based on the geometry represented as a coordinate list. MongoDB can become a promising alternative to RDBMS if further development of NoSQL databases would include the support of 3D queries and 3D indexing, especially as an architecture that supports the storage of large amounts of data.

5 CONCLUSIONS

The transition from traditional cadastral systems to 3D cadastre brings new issues on the management of spatial data and it could result in very large datasets. On one hand, some of the relational database management systems support storing 3D objects and 3D operations. Whereas, on the other hand, they still do not fully support all 3D cadastre requirements.

Considering their flexibility to support non-strict data models, the NoSQL databases can be easily used for storing 3D cadastral data in different formats such as CityGML or XML formats. This option eliminates the disadvantages of these schemes with regards to simplified modification of text documents, efficient data protection and export within NoSQL database. As a result of the free-schema approach, NoSQL databases represent a good choice when it comes to hybrid models; i.e., mixed models of the current traditional 2D cadastral registration systems and 3D cadastre elements. This is due to the fact that these models can be easily stored within a NoSQL database (unlike relational databases, where the 2D cadastre model needs to be extended with new predefined tables for the 3D elements). Their capability to handle large datasets is also interesting for subsequent investigation of using NoSQL databases as DBMS in 3D cadastre.

Analogous to RDBMS, the NoSQL database requires long-term development to fully support all 3D cadastre requirements such as 3D spatial indexing, 3D queries, and 3D topology. In the last several years, progress in developing NoSQL database has been made and it can be encouraging and promising for the development of 3D features, as well.

NoSQL databases are immature (resulting in no standard query language and user interface) and some NoSQL systems do not follow the ACID properties which might be a drawback for 3D cadastre require-

ments. Non-standard query languages and user interfaces of NoSQL are also disadvantageous when compared to relational databases.

Future work should include research on developing 3D features within NoSQL databases, accompanied with investigation that will compare capabilities of RDBMS and NoSQL database to handle large 3D cadastral datasets. Furthermore, future research should include the development of 3D cadastral models that will be adjusted to NoSQL databases. In that way, it will be possible to use all the capacity of NoSQL databases for storage and manipulation of 3D cadastral datasets.

Literature and references:

3drepo.org. (2017). http://3drepo.org/, accessed 25. 2. 2017.

Aien, A., Kalantari, M., Rajabifard, A., Williamson, I., Wallace, J. (2013). Towards integration of 3D legal and physical objects in cadastral data models. Land Use Policy, 35, 140–154. DOI: http://dx.doi.org/10.1016/j.landusepol.2013.05.014

Aien, A., Rajabifard, A., Kalantari, M., Ian, W., Shojaei, D. (2014). Development of XML Schemas for Implementation of a 3D Cadastral Data Model. 4th International Workshop on 3D Cadastres (pp. 131–158). Dubai, United Arab Emirates.

Breunig, M., Zlatanova, S. (2011). 3D geo-database research: Retrospective and future directions. Computers & Geosciences, 37, 791–803. DOI: http://dx.doi.org/10.1016/j.cageo.2010.04.016

Chandra, D. G. (2015). BASE analysis of NoSQL database. Future Generation Computer Systems, 52, 13–21. DOI: http://dx.doi.org/10.1016/j.future.2015.05.003

db-engines.com. (2016). http://db-engines.com/en/ranking, accessed 1. 6. 2016.

Drobež, P., Kosmatin Fras, M, Ferlan M., Lisec A. (2017). Transition from 2D to 3D real property cadastre: The case of the Slovenian cadastre. Computers, Environment and Urban Systems, 62, 125–135. DOI: http://dx.doi.org/10.1016/j.compenvurbsys.2016.11.002

github.com. (2017). https://github.com/evanw/csg.js/ , accessed 23. 2. 2017.

Goodhart, C., Hofmann, B. (2008). House prices, money, credit, and the macroeconomy. Oxford Review of Economic Policy, 24, 180–205. DOI: https://doi.org/10.1093/oxrep/grn009

Gröger, G., Plümer, L. (2012). CityGML – Interoperable semantic 3D city models. ISPRS Journal of Photogrammetry and Remote Sensing , 71, 12–33. DOI: http://dx.doi.org/10.1016/j.isprsjprs.2012.04.004

Lemmen, C., van Oosterom, P. (2013). The Land Administration Domain Model Standard. 5th Land Administration Domain Model Workshop. Kuala Lumpur, Malaysia.

Li, L., Luo, F., Zhu, H., Ying, S., Zhigang, Z. (2016). A two-level topological model for 3D features in CityGML. Computers, Environment and Urban Systems , 59, 11–24. DOI: http://dx.doi.org/10.1016/j.compenvurbsys.2016.04.007

Marcin, K. (2012). Registration of untypical 3D objects in Polish cadastre–do we need 3D cadastre? Geodesy and cartography, 61 (2), 75–89. DOI: https://doi.org/10.2478/v10277-012-0023-8

Mohamed, M., Altrafi, O., Ismail, M. (2014). Relational vs. NoSQL Databases: A Survey. International Journal of Computer and Information Technology, 3, 598–601 .

Mohan, C. (2013). History Repeats Itself: Sensible and NonsenSQL Aspects of the NoSQL Hoopla. Proceedings of the 16th International Conference on Extending Database Technology.

Moniruzzaman, A. B., Hossain, S. A. (2013). NoSQL Database: New Era of Databases for Big data Analytics - Classification, Characteristics and Comparison. International Journal of Database Theory and Application, 4, 1-13.

Nance, C., Losser, T., Iype, R., Harmon, G. (2013). NoSQL vs RDBMS - Why there is room for both. Proceedings of the Southern Association for Information Systems Conference. Savannah, GA, USA.

Navratil, G. (2011). Cadastral boundaries: benefits of complexity. URISA Journal-Urban and Regional InformationSystems Association, 23 (1), 19.

Navratil, G., Fogliaroni, P. (2014). Visibility analysis in a 3D Cadastre. Proceedings of Fourth International FIG Workshop on 3D Cadastres, 183–196.

Nayak, A., Poriya, A., Poojary, D. (2013). Type of NOSQL Databases and its Comparison with Relational Databases. International Journal of Applied Information Systems (IJAIS), 5, 16–19. DOI: http://doi.org/10.5120/ijais12-450888

osgl.grf.bg.ac.rs/mongo_rest. (2017). http://osgl.grf.bg.ac.rs/mongo_rest/3DCadastre/Terrain/, accessed 27. 2. 2017.

Paasch, J., Paulsson, J., Navratil, G., Vučić, N., Kitsakis, D., Karabin, M., El-Mekawy, M. (2016). Building a modern cadastre: Legal issues in describing real property in 3D. Geodetski vestnik, 6, 256–268. DOI: http://dx.doi.org/10.15292/geodetski-vestnik.2016.02.256-268

Schön, B., Laefer, D., Morrish, S., Bertolotto, M. (2009). Three-dimensional spatial information systems: state of the art review. Recent Patents on Computer Science, 2, 21–31. DOI: http://dx.doi.org/10.2174/2213275910902010021

Stonebraker, M., Madden, S., Abadi, D., Harizopoulos, S., Hachem, N., Helland, P. (2007). The End of an Architectural Era (It's Time for a Complete Rewrite). Proceedings of the 33rd international conference on Very large data bases, VLDB pp(1150–1160).

Stoter, J., van Oosterom, P. (2002). Incorporating 3D geo-objects into a 2D geo-DBMS. Proceedings ASPRS/ACSM conference. Washington, USA.

Stoter, J., Salzmann, M. (2003). Towards a 3D cadastre: where do cadastral needs and technical possibilities meet? Computers, Environment and Urban Systems, 27, 395–410. DOI: http://dx.doi.org/10.1016/S0198-9715(02)00039-X

Tauro, C. J. M., Aravindh, S, Shreeharsha, A. B. (2012). Comparative Study of the New Generation, Agile, Scalable, High Performance NOSQL Databases. International Journal of Computer Applications, 48, 1–4.

DOI: http://dx.doi.org/10.5120/7461-0336

van Oosterom, P. (2013). Research and development in 3D cadastres. Computers, Environment and Urban Systems, 40, 1–6. DOI: http://dx.doi.org/10.1016/j.compenvurbsys.2013.01.002

website-mongoDB. (2017). https://www.mongodb.com/thank-you/white-paper/nosql-considerations, accessed 27. 2. 2017.

www.un.org. (2017). http://www.un.org/en/development/desa/news/population/world-urbanization-prospects-2014.html, accessed 15. 3. 2017.

Višnjevac N., Mihajlović R., Šoškic M., Cvijetinović Ž., Bajat B. (2017). Using NoSQL databases in the 3D cadastre domain. Geodetski vestnik, 61 (3), 412-426. DOI: 10.15292/geodetski-vestnik.2017.03.412-426

Nenad Višnjevac, MSc.
University of Belgrade, Faculty of Civil Engineering
Bulevar kralja Aleksandra 73, SRB-11000 Belgrade, Serbia
e-mail: nvisnjevac@grf.bg.ac.rs

Assist. Prof. Rajica Mihajlović, Ph.D.
University of Belgrade, Faculty of Civil Engineering
Bulevar kralja Aleksandra 73, SRB-11000 Belgrade, Serbia
e-mail: rajica@grf.bg.ac.rs

Assist. Prof. Mladen Šoškić, Ph.D.
University of Belgrade, Faculty of Civil Engineering
Bulevar kralja Aleksandra 73, SRB-11000 Belgrade, Serbia
e-mail: mladens@grf.bg.ac.rs

Assist. Prof. Željko Cvijetinović, Ph.D.
University of Belgrade, Faculty of Civil Engineering
Bulevar kralja Aleksandra 73, SRB-11000 Belgrade, Serbia
e-mail: zeljkoc@grf.bg.ac.rs

Prof. Branislav Bajat, Ph.D.
University of Belgrade, Faculty of Civil Engineering
Bulevar kralja Aleksandra 73, SRB-11000 Belgrade, Serbia
e-mail: bajat@grf.bg.ac.rs