USING GEOSERVER AND POSGRESQL FOR CHANGE DETECTION OF SATELITE IMAGES

Denis Byalev[1], Egnar Özdikililer[1,2]

[1] *University of Telecommunications and Post, Sofia, Bulgaria denisbyalev@gmail.com*
[2] *Istanbul Technical University, Faculty of Aeronautics and Astronautics, Istanbul, Turkey ozdikililer@itu.edu.tr*

Abstract

The object of investigation is the use of relational DB for storing data for change detection analysis. The purpose of this paper is to: 1) to demonstrate the use of relational DB as storage for raster data for better performance while doing change detection; 2) propose relational DB schema for storing vectorized information of raster images.

In the scope of this paper is shown massive potential in storing vectorized raster data in relational DB.

Keywords: Change Detection, Remote Sensing, Satellite Images, Sentinel-2, NDWI

INTRODUCTION

Introduction Satellite change detection provides analysis based on the changes that have been detected during the time between two or more satellite acquisitions for an area of interest (AOI). The base for the project is using Setninel-2 satellite imaging as a base for creating change detection. The satellite system Sentinel-2 is equipped with MultiSpectral Instrument (MSI) instrument, which permits global coverage within a period of 6-8 days. For the region of Europe, this period is 4-5 days with 13 spectral bands in the visible up to the shortwave infrared spectrum (0.443–2.190 μm) with the width of the scene is 290 km. The spatial resolution is 10m for the visible (red, green, blue and near infrared bands), 20m for red-edge and shortwave infrared bands and 60m for the 3 bands dedicated for atmospheric corrections.

The images are processed to create NDWI index of water bodies in Kozloduy area. These raster images are stored in the DB for further use in change detection algorithm.

MATERIALS AND METHODS

The area of interest described in the paper is *Kozloduy* area. Satellite images used for analysis are Setninel-2 images. The images are processed using Normalized Difference Water Index (NDWI), expressed with the following formula:

$$NDWI = \frac{(GREEN - SWIR11)}{(GREEN + SWIR11)} \quad (1)$$

Equation 1 NDWI formula

where GREEN and SWIR11 stands for the spectral reflectance measurements acquired in the visible (GREEN) and short-wave infrared regions (SWIR11) respectively. The two tasks are the following:

1) Storing the raster data as vectorized points in relational DB.

2) Visualizing vectorized data and performing change detection algorithm.

The base approach of storing the data in DB is the following Figure 1 (DB schema for storing vectorized raster data)

As seen in Figure 1 the base idea is storing raster metadata in 'geo_raster_index_ndwi_water' table. The metadata include Sentinel - 2 tile id and date of the taken image. The table 'geo_raster_values_ndwi_water' stores all possible values of the raster image, 'map_pixel_coords' is the table for storing grid information and 'map_pixel_coords_value_ndwi_water' table

for storing value of the raster for specified point on the grid.

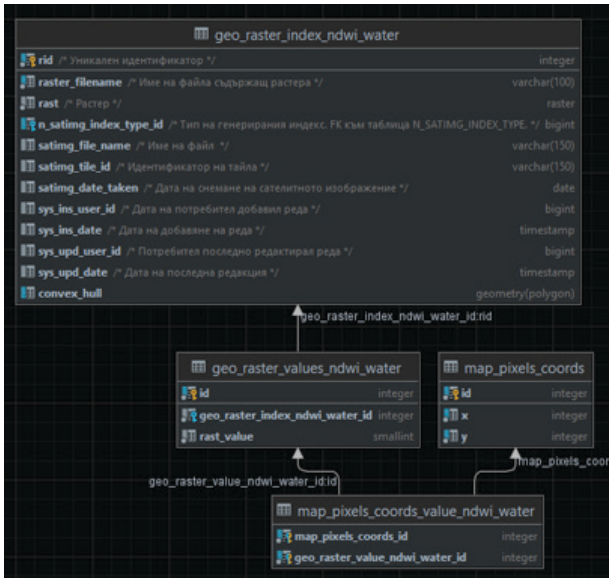

Figure 1 DB schema for storing vectorized raster data

NDWI raster image is typically in the range from -1 to 1 we multiply this value by 100 to create index in the range of -100 to 100 so that we could store it in smallint variable to save space. The image is with precision of 20m/20m per pixel. This means we store around 25 000 000 points in the database. The original SAFE file from sentinel is with the size of ~1GB, when we create NDWI raster the image size is typically around ~80MB. This is significantly lower. When we store this image in the DB we multiply this size multiple times, the result is that one image is stored in ~1 GB with the private key and indexes taking respectively ~1GB. The tradeoff storing the data in the DB is having the possibility of querying 161024 in less than a second as shown in Figure 2. The change detection (CD) is created using this formula:

$$CD = OLD - NEW \qquad (2)$$

Equation 2 Change detection formula

In (Equation 2) Change detection formula where the values are the older satellite image (OLD) and newer satellite image (NEW) respectively, this is represented as query shown in Figure 2.

The insertion of the data in the database is colossal as it takes around 8 hours to insert one raster image in the DB as vectorized data. As other point is the size of the data. For one tile from Sentinel-2 we could estimate the count of raster images for one year using the formula: N=(YD\D)*Q=(365\7)*1 ≈52, where we divide days in the year (YD) by average time new satellite image is published (Y) and multiply this my average percentage of non-cloudy images, these images have cloud coverage lower than 15%, we calculate worst case scenario where we accept all images as usable by the system. Using the average count of images for a tile we calculate average count for Bulgaria using the formula:TN = N * TL = 52 * 24 = 1248, where we multiply tile average count (N) by total tile count to cover Bulgaria (TL). Having this information, we could estimate the total size of storing the satellite images for Bulgaria for the period of one year by using the formula:

$$TS = (DATA_{SIZE} + INDEX_{SIZE}) * TN =$$
$$= (1158MB + 1042MB) * 1248$$
$$= 2200 * 1248$$
$$= 2745600MB$$
$$= \mathbf{2745.6GB \approx 2.75TB}$$

where we have the size of the data (DATASIZE) and indexes and primary keys (INDEXSIZE) multiplied by total satellite image count.

```
select ST_POINT(pg.x, pg.y, 32635) as geom, pd1.rast_value - pd2.rast_value as value
from map_pixels_coords pg
        join map_pixels_coords_ndwi_water pd1
             on pg.id = pd1.map_pixels_coords_id and pd1.geo_raster_index_ndwi_water_id = 5
        join map_pixels_coords_ndwi_water pd2
             on pg.id = pd2.map_pixels_coords_id and pd2.geo_raster_index_ndwi_water_id = 9
where pg.x >= 233135
  and pg.x <= 244012
  and pg.y >= 4850339
  and pg.y <= 4856259
[2022-07-13 22:18:33] 500 rows retrieved starting from 1 in 910 ms
                      (execution: 871 ms, fetching: 39 ms)
```

Figure 2 query used in creating change detection

The data is then visualized using Geoserver as a tool for visualizing spatial data using web map service (WMS). In Geoserver the select from Figure 2 is published as a layer with 6 parameters for the 2 satellite images we want visualize change detection and the upper left and bottom right corners represented with upper left x and upper left y and bottom left x

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<StyledLayerDescriptor version="1.0.0"
                       xsi:schemaLocation="http://www.opengis.net/sld http://schemas.opengis.n
/StyledLayerDescriptor.xsd"
                       xmlns="http://www.opengis.net/sld"
                       xmlns:ogc="http://www.opengis.net/ogc"
                       xmlns:xlink="http://www.w3.org/1999/xlink"
                       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <NamedLayer>
        <Name/>
        <UserStyle>
            <Title>NDWI_WATER</Title>
            <FeatureTypeStyle>
                <Rule>
                    <Title>NDWI_WATER</Title>
                    <PointSymbolizer uom="http://www.opengeospatial.org/se/units/metre">
                        <Graphic>
                            <Mark>
                                <WellKnownName>square</WellKnownName>
                                <Fill>
                                    <CssParameter name="fill">
                                        <ogc:Function name="Interpolate">
                                            <!-- Property to transform -->
                                            <ogc:PropertyName>value</ogc:PropertyName>
                                            <!-- Mapping curve definition pairs (input, output
                                            <ogc:Literal>-100</ogc:Literal>
                                            <ogc:Literal>#d7191c</ogc:Literal>
                                            <ogc:Literal>-50</ogc:Literal>
                                            <ogc:Literal>#fdae61</ogc:Literal>
                                            <ogc:Literal>0</ogc:Literal>
                                            <ogc:Literal>#ffffbf</ogc:Literal>
                                            <ogc:Literal>50</ogc:Literal>
                                            <ogc:Literal>#abdda4</ogc:Literal>
                                            <ogc:Literal>100</ogc:Literal>
                                            <ogc:Literal>#2b83ba</ogc:Literal>
                                            <!-- Interpolation method -->
                                            <ogc:Literal>color</ogc:Literal>
                                            <!-- Interpolation mode - defaults to linear -->
                                        </ogc:Function>
                                    </CssParameter>
                                </Fill>
                            </Mark>
                            <Size>20</Size>
                        </Graphic>
                    </PointSymbolizer>
                </Rule>
            </FeatureTypeStyle>
        </UserStyle>
    </NamedLayer>
</StyledLayerDescriptor>
```

Figure 3 SLD for NDWI_WATER visualization

and bottom left y respectively. The visualization of layers in Geoserver is done using style layer descriptor (SLD). The value of the coordinate is being interpolated and based on the color table shown in Figure 3 and Figure 4 the specified color is shown.

The visualization relies on render on request to calculate the color, different approach would have been to store the color table inside the DB but this would have added more overhead in the data retrieval step. The request time rendering does not add significant performance bottleneck and this the bottleneck is still in the querying of the data.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<StyledLayerDescriptor version="1.0.0"
                       xsi:schemaLocation="http://www.opengis.net/sld http://schemas.opengis.net/sld
/StyledLayerDescriptor.xsd"
                       xmlns="http://www.opengis.net/sld"
                       xmlns:ogc="http://www.opengis.net/ogc"
                       xmlns:xlink="http://www.w3.org/1999/xlink"
                       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <NamedLayer>
        <Name/>
        <UserStyle>
            <Title>CHANGE_DETECTION</Title>
            <FeatureTypeStyle>
                <Rule>
                    <Title>CHANGE_DETECTION</Title>
                    <PointSymbolizer uom="http://www.opengeospatial.org/se/units/metre">
                        <Graphic>
                            <Mark>
                                <WellKnownName>square</WellKnownName>
                                <Fill>
                                    <CssParameter name="fill">
                                        <ogc:Function name="Interpolate">
                                            <!-- Property to transform -->
                                            <ogc:PropertyName>value</ogc:PropertyName>
                                            <!-- Mapping curve definition pairs (input, output) -->
                                            <ogc:Literal>-60</ogc:Literal>
                                            <ogc:Literal>#4bf2f4</ogc:Literal>
                                            <ogc:Literal>-50</ogc:Literal>
                                            <ogc:Literal>#d30c08</ogc:Literal>
                                            <ogc:Literal>-45</ogc:Literal>
                                            <ogc:Literal>#f31611</ogc:Literal>
                                            <ogc:Literal>-40</ogc:Literal>
                                            <ogc:Literal>#fd3908</ogc:Literal>
                                            <ogc:Literal>-35</ogc:Literal>
                                            <ogc:Literal>#fd5108</ogc:Literal>
                                            <ogc:Literal>-30</ogc:Literal>
                                            <ogc:Literal>#ffe817</ogc:Literal>
                                            <ogc:Literal>0</ogc:Literal>
                                            <ogc:Literal>#ffffff</ogc:Literal>
                                            <ogc:Literal>30</ogc:Literal>
                                            <ogc:Literal>#cce98e</ogc:Literal>
                                            <ogc:Literal>35</ogc:Literal>
                                            <ogc:Literal>#b2de76</ogc:Literal>
                                            <ogc:Literal>40</ogc:Literal>
                                            <ogc:Literal>#92cf64</ogc:Literal>
                                            <ogc:Literal>45</ogc:Literal>
                                            <ogc:Literal>#61ac50</ogc:Literal>
                                            <ogc:Literal>50</ogc:Literal>
                                            <ogc:Literal>#34873c</ogc:Literal>
                                            <ogc:Literal>60</ogc:Literal>
                                            <ogc:Literal>#0c471d</ogc:Literal>
                                            <!-- Interpolation method -->
                                            <ogc:Literal>color</ogc:Literal>
                                            <!-- Interpolation mode - defaults to linear -->
                                        </ogc:Function>
                                    </CssParameter>
                                </Fill>
                            </Mark>
                            <Size>20</Size>
                        </Graphic>
                    </PointSymbolizer>
                </Rule>
            </FeatureTypeStyle>
        </UserStyle>
```

Figure 4 SLD for CHANGE_DETECTION visualization

The published layers are all in the spatial reference system (SRS) of EPSG:32635. This system was chosen because it uses metric units and the data shown in this paper falls inside the area of use of the SRS. The background in the demo uses Open Street Map service as it provides free background road maps.

RESULTS AND DISCUSSIONS

The published layers in Geoserver provide visualization for the NDWI data shown in Figure 5 and Figure 6 respectively. We use change detection to visualize the change visualized in Figure 9 and Figure 10.

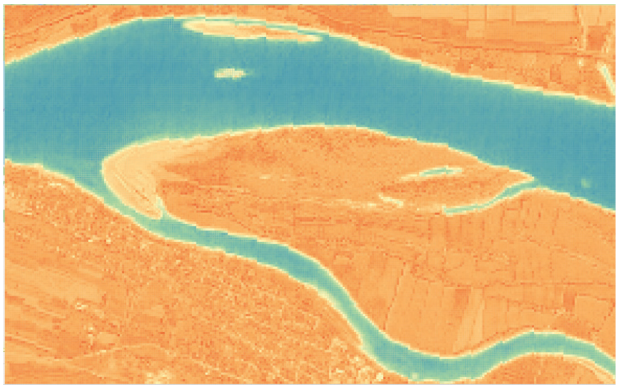

Figure 5 NDWI of Kozloduy on 2021/09/15

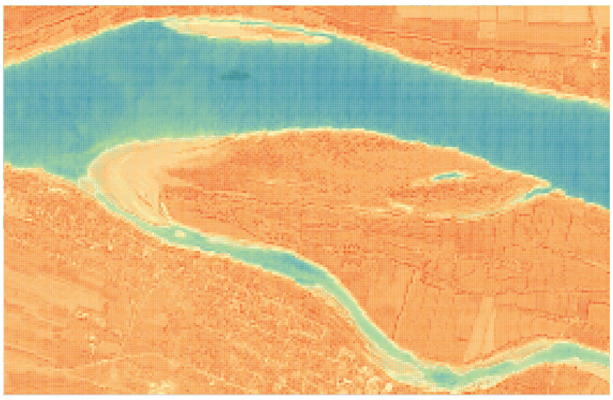

Figure 6 NDWI of Kozloduy on 2021/09/25

In the images we can determine that Figure 6 shows more land mass or we could say less water level than Figure 5. To visualize the change in the water level we perform change detection algorithm described at Equation 2, where Figure 6 and Figure 5 is used as OLD and NEW respectively in the formula.

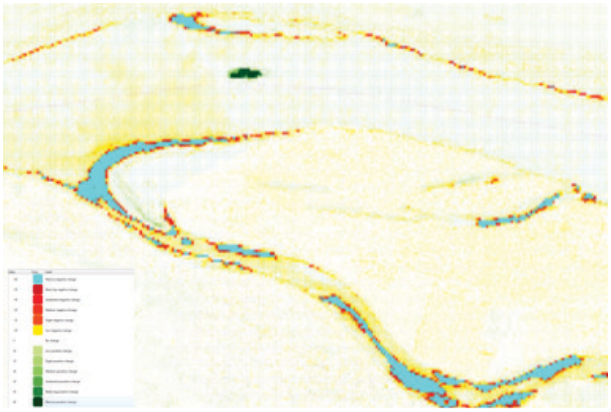

Figure 7 Change detection between Figure 5 and Figure 6 include color range(mid Figure 8)

The result of the algorithm is shown in Figure 7. The image shows that sky blue colored areas have massive negative change to the water level as well as some red areas. The full range of colors and their meaning could be seen at Figure 8.

Figure 8 Change detection between 2021/09/25 and 2021/10/20

Figure 9 Change detection between 2021/10/20 and 2021/12/19

From Figure 9 and Figure 10 we can see the increase in water level of Danube River. The visualization of the data is ~2 seconds including network traffic. This is visualization represents the tradeoff of storing more data to increase the speed of calculation to near real time. The area presented in Figure 9 and Figure 10 is about ~168km2. This provides an example if we have enough storage to store the data, we could potentially provide real time change detection algorithm for use in remote sensing systems.

There is potential in using non-relational DB in storing the data to increase read and write speed of the satellite data. One potential is using Apache Cassandra for storing the data and GeoMessa for creating spatial indexes as they claim to be able to query up to 80 000 points under one second. As the solution described in the paper is able to query near this amount this switch of storage could potentially improve the read speed to increase area of interest, we could analyze with the change detection algorithm.

CONCLUSION

The result of having the raster image data stored in relational DB as vectorized data provides significant increase in speed when calculating change detection. This paper provide example of one such implementation. The biggest disadvantage of this implementation is the increase in data being stored. This being said it's safe to assume that with enough storage space this implementation could provide real time change detection to mass audience of people. This being said it is highly recommended to experiment with other realization on this idea to increase performance or save storage space without sacrificing performance or the nature of the real time system.

REFERENCE

[1] Roumenina E., V. Vassilev, V. Naydenova, K. Ruskov. 2009. Land use dynamics of areas threatened with floods using object-oriented classification of very high resolution imagery. In: Proceedings of 5-th Scientific Conference with International Participation "Space, Ecology, Nanotechnology, Safety" - SENS2009, 2–4 November 2009, Sofia, Bulgaria. pp. 116-124

[2] Vassilev V. 2013. Crop identification using SPOT-VGT NDVIs S10 time-series data for the arable land on the territory of republic of Bulgaria. Aerospace Research in Bulgaria. Published by SRTI-BAS, Issue 15, ISSN: 1313-0927 pp.172-182.

[3] Launay, M., and M. Guerif. 2005. Assimilating remote sensing data into a crop model to improve predictive performance for spatial applications. Agriculture, Ecosystems and Environment, 111, 321−339.
[4]Lang, S., 2008. Object-based image analysis for remote sensing applications: Modeling reality, dealing with complexity. In: Blaschke, T., Lang, S., Hay, G.J. (Eds.), Object Based Image Analysis. Springer, Heidelberg, Berlin, New York, pp. 1-25.
[5]Lillesand, T.M., R.W. Kiefer, J.W. Chipman. 2004. Remote Sensing and Image Interpretation, fifth ed. Wiley, New York, NY, 763pp.