

# **IMPLEMENTACIJA INSPIRE STANDARDA U OBLASTI GEOPROSTORNIH PODATAKA**

## **UVOD**

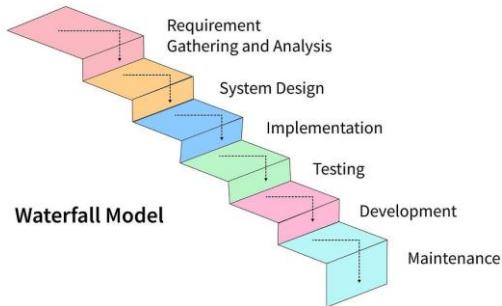
Direktiva INSPIRE opisuje 34 teme prostornih podataka koji su potrebni u aplikacijama vezanim za zaštitu životne sredine. Teme su podeljenje u tri aneksa. Aneks I sadrži osnovne teme podataka na osnovu kojih se kreiraju druge teme. Za svaku od tema postoji specifikacija koja uključuje UML model date teme. INSPIRE standard je pastraktni standard, na istom nivou kao i ISO/TC 19100 serija, što znači da definiše De Jure dnosno neki vid skupa podataka kojim navodi šta je sve predmet budućeg sistema a šta ne pripada domenu budućeg sistema, kako su elementi domena povezani, rečnik pojmoveva i sl. Dakle navodi šta sve u tzv. universe of discourse spada u predmet, katalozira za buduću implementaciju. Zato je upravo prvo poglavlje elaborata (razrade) konceptualno modelovanje. Svako poglavlje pokriveno je uvodom koji ima za cilj da objasni šta je cilj kojeg poglavlja (koraka). Svaki korak povlači prethodni. Na kraju svakog poglavlja je dat ili zaključak ili referenca na sledeći korak.

## 1. KONCEPTUALNA ŠEMA - UML

Modeli se izrađuju u cilju boljeg razumevanja sistema, pomažu projektnom timu u prikazu sistema koji se formira i omogućuju evidentiranje raznovrsnih ograničenja na sistemu, eventualnih grešaka.

Tehnički opis UML-om je precizan, nedvosmislen i potpun, pa na taj način omogućava razumevanja strukture i ponašanje sistema uz smanjeni rizik pogrešnog tumačenja. Tako da će u ovom poglavlju sve biti dato u notama/buletićima kao nezavisno obješnjje nekog modela sa slike bez ulazeњa u dubinu semantike.

U cilju interoperabilnosti, efikasnosti, manjih troškova obezbeđen je set modela, takozvanih tema u sklopu INSPIRE standarda. Ovo je izuzetno korisno jer sistemi za upravljanje katastrom su izuzetno kompleksni i obimni (aneks II to pokazuje) i u slučaju greške teško se vratiti, skupo je i rizično (waterfall model), pa je smanjen rizik.

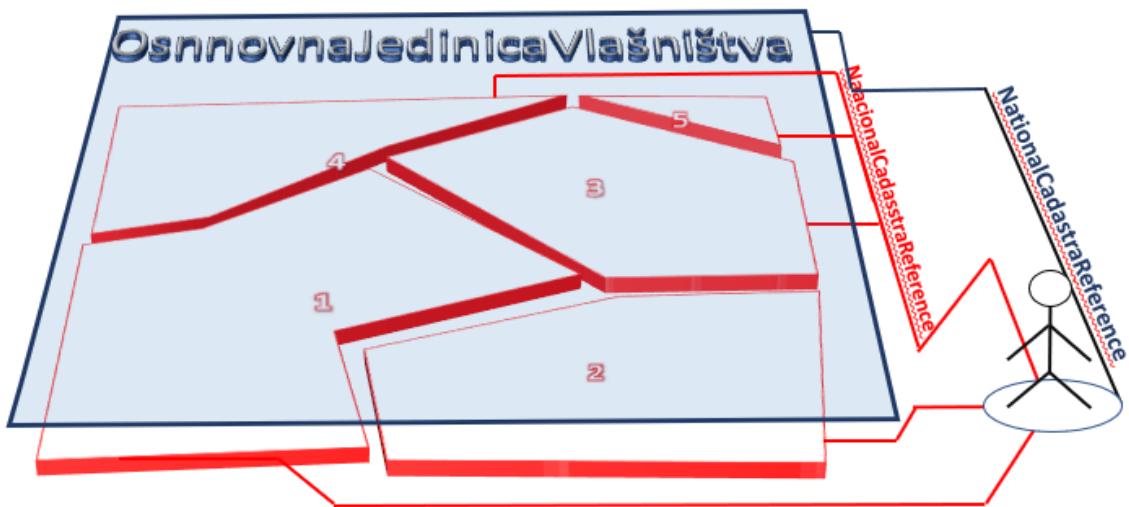


Slika 1. Waterfall model za velike sisteme

### 1.1.OsnovnaJedinicaVlašništva

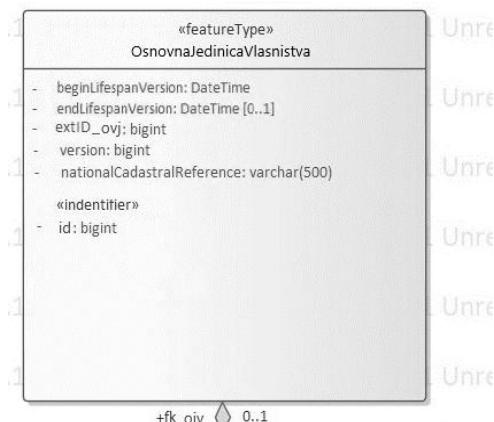
Osnovna jedinica vlašništva je zabeležena u zemljišnim registrima. U nekim zemljama je ovo veza sa nacionalnim registrima, što je navedeno atributom „NationalCadastralReference“, ostvarena preko ovih jedinica, a ne preko prostorno referenciranih parcela (iduća klasa). Zapravo reč je samo o drugčijem uređenju sistema, od države do države, zavisi od zatečenog stanja, što je prikazano na slici 1. U nekim

zemljama je ovo zapravo parcela, pa je odatle minimalni kardinalitet kod pola na strani klase „KatastarskaParcela“ 0. Primjer upotrebe je Norveška ili Finska.



**Slika 2.** Intuicija za razliku dva sistema

Katastarske jedinice su u biti definisane kao jedinstvena područja Zemljine površine pod homogenim pravima nad nepokretnostima i jedinstvenim vlašništvom, pri čemu je ovo primer razdvajanja.



**Slika 3.** Osnovna jedinica vlašništva

Atributi klase su:

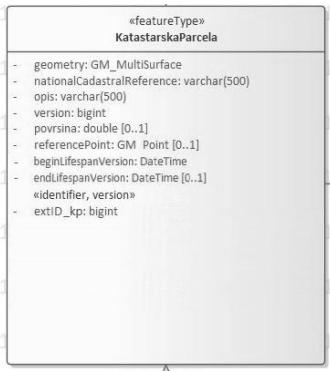
- beginLifeSpanVersion - definiše vreme i datum (timestamp) kada je jedinica postala aktivna
- endLifeSpanVersion – definiše kada je, pri komasaciji ili sl.jedinica postala neaktivna.

- extId\_ovj – klase i atributi sa prefiksom ext se odnose na vrednosti za koje se očekuju podatci iz eksternalnih izvora, registara.
- Id – jedinstveni identifikator sa obzirom da extId\_ovj nije jak kandidat za ključ da bi bio primarni, jer se extId\_ovj može ponoviti zmeđu opština, biti pogrešan i sl.
- Version – ISO 19019/INSIRE – ako neke od geoprostornih objekata (featuretype) budu eliminisani iz klase koja ih agregira, onda identifikator klase može ostati isti ali sa različitnom verzijom. u obzir da se verzioniranje projekta vrši kroz neki software koji je bolji za to, kao git, jer on kompletno evidentira sve promjene.
- nationalCadastralReference – ključan atribut, upućuje (URI, URL, URN) na neki spoljni izvor, tj. nacionalni registar ako je postojan nacionalni registari dobijeni iz baza kao što su grutovnice, zamljišne knjige, katastar nepokretnosti i sl. Dakle registri koji su korišteni prije, standarizovani u okviru pojedine države.

Svaka klasa ima stereotip, prema ISO 19109 i ISO19103 klase čiji su objekti geoprostorni podatci imaju stereotip <<featureType>>, dok su ostali entiteti koji nemaju geografsku referencu označeni sa <<dataType>> (u mom slučaju nemam takvih entiteta).

Bilo je potrebno dodeliti stereotip <<featureType>> i klasi OsnovnaJedinicaVlašništva, jer u slučaju reference na KatastarskuParcela klasu koja jeste prostorno određena, onda je i ova kasa sa prostornim značajem, odnosno tretira se kao da je prenesen ključ atribut koji je geoprostornog tipa. Višestrukoštiti atributa se navode klikom na atribut, odabivom kartice „potperies“. Ako je max i min multiplicity 1 onda se ni ne prikazuje. Ako je 0 onda znači da nije obavezan a ako 1 znači da je obavezan. Stereotip <<indifier>> se definiše kao „za modele baze podataka atribut predstavlja polje primarnog ključa“.

## 1.2.KatastarskaParcela



**Slika 5.** Katastarska parcela

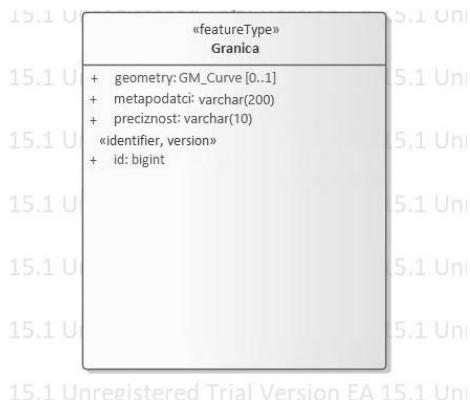
Referenca na semantiku ove klase je data sa osnovnom jedinicom vlasnistva. Kako naziv kaže, to je pojedina parcela, što znači da ima i svoj prostorni identitet, nalik na specijalizaciju od prethodne klase.

- Geometry – ISO19125 – tipa GM\_MultiSurface tip odgovara PostGIS tipu MultiPoygon. Međutim ja ћu koristiti samo Polygon jer mi GIS pri testu neomogućuje upotrebu takvog tipa.
- NationalCadastralReference – kao i prethodno, metapodatci i podatci na nekom spoljnem izvoru
- Povrsina – pošto imam geometriju na osnovu nje mogu dobiti i površinu , double odgovara tipu decimal ,preciznost 2 demale (decimetar kvadratni)
- referencePoint – ISO19125 tip GM Pint koji se prevodi kao Point a predstavlja središnju tačku parcele. Korisno za vizuelizaciju velikog broja parcela.

Za usporedbu sa LADAM, katastarska parcela odgovara klasi LA\_SpatialUnit koja ima dve specilaizacije, međutim zgrade i mreže se u INSPIRE razmatraju u drugim temama.

### 1.3.Granica

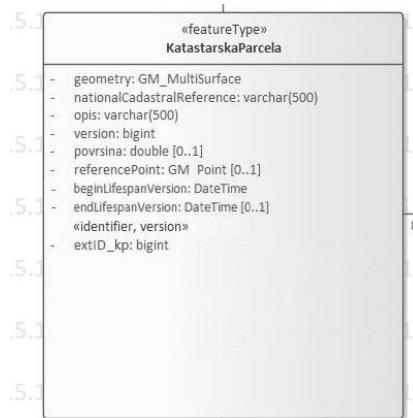
Koristi se za predstavljanje granica katastarskih čestica.



Slika 6. Klasa granica

Ključni su atributi metapodatci i preciznost. Metapodatci govore o tome kako je granica izmjerena, obilježena, da li ima sporova i sl. Preciznost je u suština zašto se postavlja ova klasa, jer granica kao ni bio šta drugo nije tačna, merena i obilježena sa poznatim ili nepoznatim geškama. Preciznost se može izraziti kao  $\pm 1m$  ili postotak npr. 95% pouzdanosti.

### 1.4.CadastralZoning



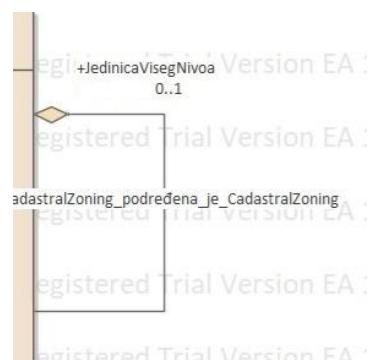
Slika 7. Klasa hirerarhije administrativnih jedinica

Odnosi se na regjone u okviru države, i samu državu.

- Level – pored naziva admin. jedinice neophodno je radi upita i sigurnosti u ispravnost unosa unjeti i brojčanu vrednost nivoa. Nivi su od 1 od 3, gdje je 1 najnjiži a 3 najviši nivo, ondono K.O. i država.
- levelName – naziv levela, varira od nacie do nace, kod nas za primjer to će biti država, politicka opstina i katastarskaopstina

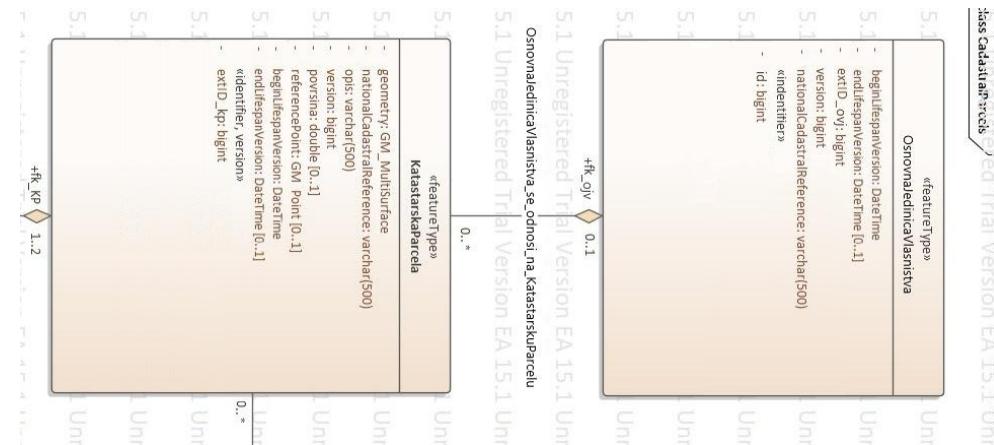
- GeografskiLokNaziv – naziv admin. jednica, npr. Surčin, ili Banjaluka, Kalnik i sl.
- beginLifeSpanVersion - definiše vreme i datum (timestamp) kada je jedinica postala aktivna
- endLifeSpanVersion – definiše kada je, pri komasaciji ili sl.jedinica postala neaktivna.
- Version – ISO 19019/INSIRE – ako neke od geoprostornih objekata (featuretype) budu eliminisani iz klase koja ih agregira, onda indifikator klase može ostati isti ali sa različitnom verzijom.
- Povrsina – pošto imam geometriju na osnovu nje mogu dobiti i površinu , preciznost 2 decimalne
- Geometry – GM\_MultiSurface ISO19125 je ključno za navesti, jer taj tip omogućuje kolekciju poligona u jednu temu, a to omogućuje admin. jedinice kao što su entitet Republika Srpsa koje imaju „prekid“, imaju svoje djelove. Odgovara PostGIS tipu Multipolygon. Međutim ovaj tip nije podržan kada budem testirao bazu u QGIS-u , podržan je Polygon. Ključno je da ovakav entitet ima rupe (hole)

Ako postoji više nivoa, mora postojati veza agregacije između dve instancije klase. Minimalni kardinalitet 0 znači da najviša jedinica u hierarhiji ne može imati naredenu višu, a maksimalan kardinalitet 1 znači da svaka jedinica mora biti potpuno sadržana u jednoj i samo jednoj administrativnoj jedinici, što nije slučaj za parcele. To će biti poseban fokus implementacije. Sa druge strane, svaka admin. jedinica agregira neku nižu jedinicu, sem najniže. Zato je ovo veza agregacije.



**Slika 8.** Prethodno opisana agregacija hirerarhije

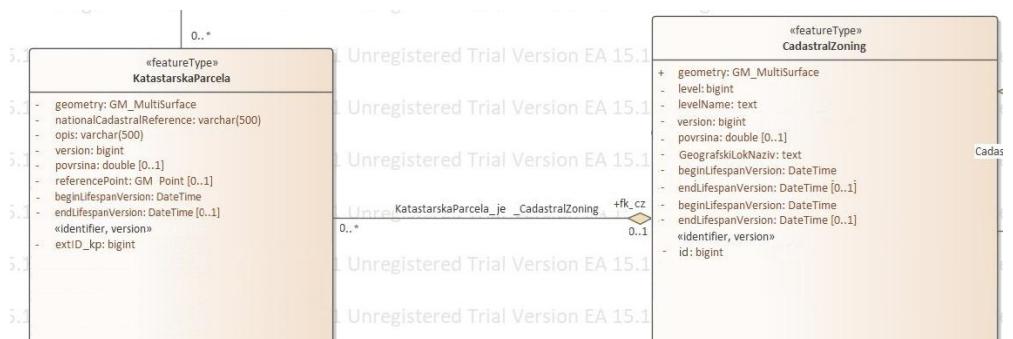
## 1.5.OsnovaJedinicaVlašništva\_se\_odnosi\_na\_KatastarskuParcelu



Slika 9. Agregacija klase katastarska parcela u osnovnoj jeninici vlašništva

Pol **fk\_ojv** ima min. Karninalitet (minOccurace) 0 što je već navđeno u poglavlju OsnovnaJedinicaVlašništva. Maksimalni kardinalitet \* je u skladu sa slikom 1. Cjelina nije dakle obavezna na djelove i zato je veza agregacije. Sa druge strane, pojedna katastarska parcela se može odnositi tačno na jednu osnovnu jedinicu, odnosno obuhvaćena tačno jednim, jednistvenim pravom (pravo vlašništva jednog vlasnika ili više suvlasnika). Pošto je minimalni kardinaliter sa obe strane 0, a sa strane djela je max. 1, tada se ključ koji osigurava referencjalni integritet prenosi djelu KatastarskaParcela, odnosno tamo gdje je jednistven.

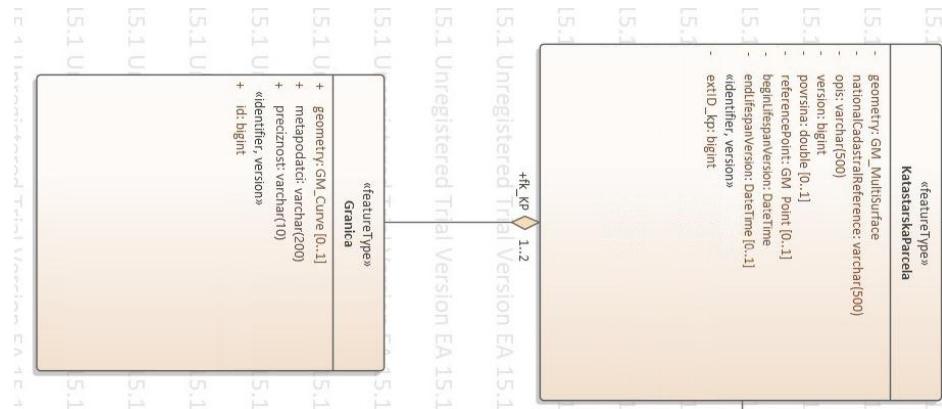
## 1.6.KatastarskaParcela\_je\_CadastralZoning



Slika 10. Veza agregacije između dve klase

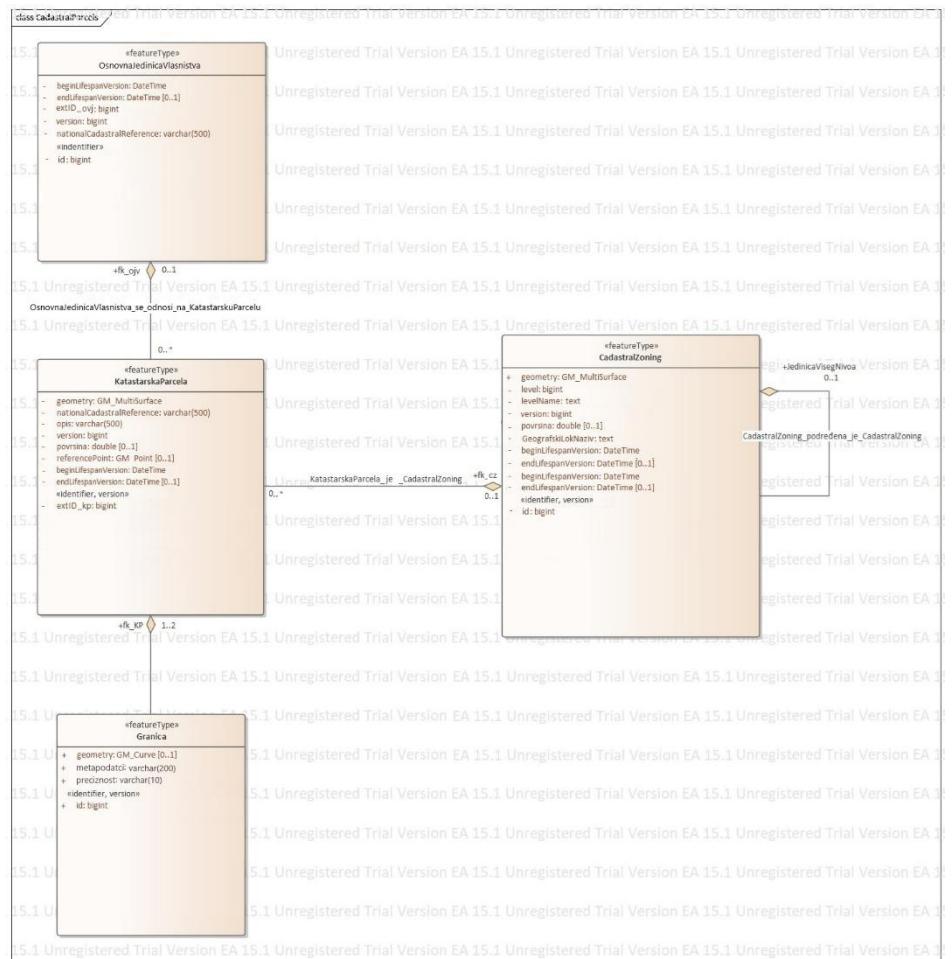
KatastarskaParcela može imati (i ne mora) dodeljenju zonu. Katastarska zona također može imati doeljenu parcelu. Minimalni kardinaliteti sa obe strane su 0 jer se dozvoljava preuređivanj, brisanje a u takvim slučajevima podaci moraju ostati sačuvani. Ne bi bilo dobro da uslijed brisanja K.O. zbog promjene granica nestanu svi podaci o katastarskim parcelama, isto tak vredi i za brisanje parcela koje su možda pogrešno unjete, K.O. još postoji, zato nije veza kompozicije. Fk, pošto je max. kard. sa strane kat. parcele 1, se prenosi u kat. parcelu jer je tu jedinstven.

## 1.7. Veza KatastarskaParcela - Grаница



Slika 11. i granice su bitan atribut parcele

Katastarska parcela je ograničena granicom. Granicu mogu da dele do dvi parcele. Granica uslijed raznih grešaka, kartiranja, merenja, obilježavanja, starih registara i slično nije potpuno ona koja je prikazana poligonom parcele. Zbog navedenih a i drugih okolnosti moguć je i spor. Sve to se evidentira u atributima objekta klase Granica. Zbog prvog navedenog min. kard je 1 a max. je 2. Nije neophodno za svaku ivicu parcele voditi evidenciju pa zato nije kompozicija već veza agrgacije. Ključ je zbog kax. Kardinaliteta jedinstven samo u obj. klase Granica pa se tu prenosi.



Slika 12. Celokupan model

Također, prema inspire za sve atribute se navodi i LifeCycleInfo. Ovaj stereotip bi trebao da pri transformaciji ukaže na to da tablica treba da ima i tri kolone, tipa „creationDate“, „modificationDate“ i „status“, međutim takvu funkcionalnost pružaju i programi za verzioniranje, npr. program koji ja krisim je git (program komadne linije).

Sada kada su dodeljeni odgovarajući stereotipi je moguće izvršiti transformaciju sa nekim alatima za transformaciju. Takav model se naziva aplikativni model. Međutim u trenutku izrade elaborata sam imao problema sa shapechange-om zbog nekompaktibilnosti verzije jave i enterprise arhitecta, jer je zahtjevan .qea format a ne .eap, međutim .qea je format koji je podržan u verzijama nakon 15, a transformator iz .eap u qea format također. U svakom slučaju cilj konceptualnog modelovanja po definiciji je postignut, a model transformacije je opcion i od njega varira cijena, efikasnost, vreme jer ručno transformisanje modela bi bilo jako sporo i složeno.

## 2. SQL , POSTGIS I QGIS

Ovdje će biti izložen algoritam i specifikacije baze podataka, na kraju napomene vezane za specifičnost SQL koda, i zatim upiti.

### 2.1.Instalacija PostGIS-a

Za početak je potrebno kreirati ekstenziju koja omogućuje mnogobrojne PostGIS alate, idući *Servers/PostgreSQL 15/Databases/gis/Extensions/Create/Extension* i iz padajućeg menija u kartici General odabratи proširenje „postgis“, te stisnuti OK. Ili jednostavno klauzulom CREATE EXTENSION postgis.

Kontrola da li je PostGIS instaliranje se izvodi pokretanjem klauzule SELECT postgis\_full\_version() kao na slici 13.

The screenshot shows a PostgreSQL query editor interface. The top section is labeled 'Query' and contains the SQL command: '1 select postgis\_full\_version()'. Below this is the 'Data Output' tab, which displays the results of the query. The results table has two columns: 'postgis\_full\_version' and 'text'. The first row shows the value 'POSTGIS="3.5.0 3.5.0" [EXTENSION] PGSQL="150" GEOS="3.13.0-CAP1-1.19.0" PROJ="8.2.1 NE"'.

Slika 13. potvrđan odziv

### 2.2.Tipovi podataka

Korišteni su sljedeći tipovi sa sledećim razlozima:

- BISERIAL – ovo nije tip podataka, ovo je mehanizam koji stvara sequence generator ili prosto sequence (sufiks serial) koja menađira polje jedinstvenog identifikatora id sa monotono rastućim pozitivnim celim brojevima tipa bigint (prefiks big). Bigint je prikladniji za baze podataka sa milione parcela, pri čemu je maksimalan broj parcela 9223372036854775807 ( maksimum koji stane u 8 bajta).

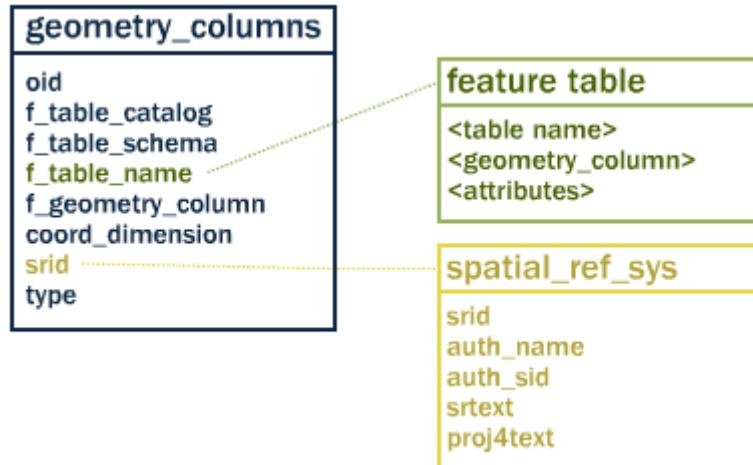
- DECIMAL – vrednost površine velikih teritorija bi trebala da može biti izražena sa 38 znamenki pre decimalnog mesta. Preciznost površine je druga decimala, decimetar kvadratni.
- Ostale geomtrije – geometrije su prikazane u litingu pogleda geometry\_columns slika17.

Pregled geometrije baze podataka se može dobiti preko objedinjenog ogleda.

Query History														
1	select * from geometry_columns													
2														
	Data Output Messages Geometry Viewer Notifications													
geometry_columns														
	f_table_catalog	f_table_schema	f_table_name	f_geometry_column	coord_dimension	srid	type							
1	gis	public	cadastralzone	geometry		2	3908 POLYGON							
2	gis	public	Katastarskaparcela	geometry		2	3908 POLYGON							
3	gis	public	parecele	geom		2	26918 POLYGON							
4	gis	public	Katastarskaparcela	referencePoint		2	3908 POINT							
5	gis	public	granica	geometry		2	3908 LINESTRING							

Slika 14. Objedinjeni pogled, bitno za provjeru koordinatnih sistema

Konceptualno, geometry\_column join-a dve tablice , slika 15.

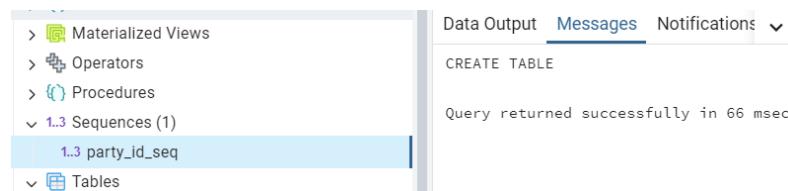


Slika 15. pogled iz listinga

Primjer kreiranja sequence i kako se na koristi dat je u listinzima.

```
Query  Query History
1 ✓ CREATE TABLE public.Party
2   ("id" BIGSERIAL, "naziv" VARCHAR(500))
```

Slika 16. Kreiranje



Slika 17. gdje su vidljive sekvence

Sequence su, u najširem pogledu, specijalne tablice sa jednim zapisom, a mimo BIGSERIAL sekvenca bih se kreirala klauzulom CREATE SEQUENCE ime\_sekvence. Slika 16 je definicija ovog objekta.

The screenshot shows the pgAdmin interface with the 'Data Output' tab selected. It displays a table with two rows and an associated SQL query. A red box highlights the value '2' in the second row of the table, which corresponds to the value in the 'last\_value' column of the sequence's internal table.

	<b>id</b> bigint	<b>naziv</b> character varying (500)
1	1	Lil Nas X
2	2	X AE 12

1    **SELECT** \* **FROM** public.party\_id\_seq

Data Output Messages Notifications

Showing rows: 1 to 1 Page No: 1

	<b>last_value</b> bigint	<b>log_cnt</b> bigint	<b>is_called</b> boolean
1	2	31	true

Slika 18. Primer tablica sequence

PAŽNJA! Ovaj trivijalan primer sam naveo sa razlogom. Naime, očekivaću sledeći tok aktivnosti:

1. Kreiram 3 zapisa, sa id. = 1,2,3 redom
2. Obrišem zadnji zapis sa id.= 3
3. Restartujem sekvencu na last\_value n tako da se ponaša kao da brisanja nije bilo
4. Kreiram novi zapis, što znači da ima id.3

Ako bih n uzeo mahom da je 2 za last\_value tj. restart jer je to prethodni id. desilo bi se sledeće:

1. Kreirao bih novi zapis (prethodni korak 4.)
2. Prije dodavanja u bazu, sequence vidi da last\_value 2 nije dodeljen – id\_called = false i dodeljuje taj broj kao id.
3. Novi zapis sada nema id.=3 kao očekivano već id.=2
4. Sistem javlja upozorenje ograničenja PRIMARY KEY

Neće biti trivijalno za imati u vidu da se last\_value trebao biti 3 a ne 2.

### 2.3.Problemi

Problemi navedenih tipova su sledeći:

- Topologija poligona adimn. nije osigurana, zahtjevi su:
  1. jedinica nižeg ne smije biti van granica jedinice višeg nivoa
  2. parcele se ne smiju sjeći
  3. admin. jedinice se ne smiju sjeći na istom nivou
    - U slučaju brisanja zapisa imamo:
      1. prekid u nizu indekasa
      2. ako su svi zapisi izbrisani, prvi sledeći id nije 1

Problemi referencijalnih integriteta, veza:

1. u slučaju brisanja K.O., npr.izmena i dopuna granica, brišu se i svi preneseni ključevi u sadržanim parcelama ( problem agregacije), problem ponovnog manuelnog unosa
2. u slučaju brisanja više admin. jednice, sve niže sadržane jedinice potrebno je opet dodefinisati novom višom jedinicom, slično problemu parcela
3. veliki broj veza geoobjekata otežava hand-handling update verzija celina

Problem površina:

1. ST\_Area poligona izuzima rupe iz površine Parcele morju imati rupe, a rupe ne ulaze u površinu parcele
2. A'priori računanje površine za velik broj parceli je previše vremenski skupo
3. Parcele morju imati rupe, a rupe ne ulaze u površinu parcele

## 2.4. Baza

```
Query Query History
1 ✓ create table public.CadastralZone(
2   "Id_cz" BIGSERIAL PRIMARY KEY,
3   "geometry" geometry("POLYGON",3908) null,
4   "levelName" varchar(100) not null,
5   "level" integer not null check("level">>0 and "level"<4),
6   "jedinicaVisegNivoa" bigint null--min Occurance = 0
7   -- nema referencijalnog integrata preko clause FOREIGN KEY(localname)
8   --REFERENCES parent(jidname) on delete ... ,omogucuje izmjene nizih hir., npr. uredjenja nekih opština, kada je potrebno objediniti ili stvoriti novu...
9   -- referencijalni integritet se održava preko triggera
10  );
11 ✓ create table public.KatastarskaParcela(
12   "extId_kp" BIGSERIAL PRIMARY KEY,
13   "geometry" geometry("POLYGON",3908) null,
14   "fk_cz" bigint null;
15  CREATE TABLE public.OsnovnaJedinicaVlasnistva (
16    "id" BIGSERIAL PRIMARY KEY,
17    "extId_ov" varchar(100) not null, --složjni izvor
18    "nationalCadastralReference" varchar(200) not null,-- referencia na RRR
19    "beginLifespanVersion" timestamp default now(),
20    "endLifespanVersion" timestamp null check("endLifespanVersion">>"beginLifespanVersion"),-- null jer je min Occurance 0
21    "version" integer default 1;
22 ✓ ALTER TABLE public.cadastralzone
23   add column "beginLifeSpan" timestamp not null DEFAULT now() ,
24   add column "endLifeSpan" timestamp null,
25   add column "version" bigint DEFAULT 1,
26   add column "povrsina" decimal(13,2),
27   add "lokalniNazivZone" varchar(1000) not null;
28 ✓ ALTER TABLE public.katastarskaparcela
29   ADD COLUMN "extBr.parcele" varchar(100), -- broj parcele
30   ADD COLUMN "nationalCadastralReference" varchar(500) null,
31   ADD COLUMN "opis" varchar(500) null,
32   ADD COLUMN "version" bigint DEFAULT 1,
33   ADD COLUMN "povrsina" decimal(13,2),
34   ADD COLUMN "referencePoint" geometry('POINT',3908) constraint topologija_molim check(st_within("referencePoint", "geometry")),
35   ADD COLUMN "beginLifeSpan" timestamp default now(),
36   ADD COLUMN "endLifeSpan" timestamp null CONSTRAINT vremenskatoTopologija_molicu check("endLifeSpan">>"beginLifeSpan"),
37   ADD COLUMN "fk_ovj" bigint null; -- jer je min Occurance =0
38 ✓ CREATE TABLE Granica ("id" BIGSERIAL PRIMARY KEY,"geometry" geometry("LINESTRING",3908),"metapodatci" varchar(500) not null,"preciznost" varchar(100) not null,
39   "fk_ljeva" bigint null, -- i jedno i drugo je nil jer parcela može da ne granici sa drugom, npr uz granicu drzave...
40   "fk_desna" bigint null
41 );
```

Slika 19. Kreiranje tablica

Primetiti treba da se u upitima za prostorne odnose ST\_Within,ST\_Contains koristi WKT format unosa geometrije parcele. WKT je npr. POLYGON((0 0, 0 10, 10 10, 10 0, 0 0)) ili POINT( 3242232,123232). Serijaliziran oblik geometrije u tablici se prevodi u WKT sa ST\_AsText(geometry) prema tome ne specificira koordinatni sistem SRID, što mi olakšava relativnu usporedbu kad znam da su sve geom u istom sistemu.

Poseban fokus je na topologiji, gdje se referentna tacka mora nalaziti u poligonu, ogranicenje je prokazano na slici 19. u crvenom okviru.

## 2.5. Automatizacija i funkcionalnost

Osnovna ideja je da se sva poslovna pravila i ogranicenja odvijaju na nivou sloja podataka.

Pošto ovo ima za cilj da jednog dana služi u sklopu IGP servisa treba znati se ispravno pozicionirati. Servisi IGP su deo sistema baziranog na servisno orientisanoj arh. (SOA), kroz realizaciju arhitekture od tri sloja:

1. Podataka
2. Servisa

### 3. Aplikacija

Tako da servisi služe za dobavljanje i razmjenu podataka sa sloja podataka po zahtjevu korisnika na apk. sloju, apk. sloj je IO panel za korisnike i ne bi se tu tek trebali rešavati problemi vezani za unutanje procedure vezane za podatke, kao što su ograničenja, indeksi, površine . Neće biti korišteni plugin's iz QGIS-a ili pisan python kod za isto, zbog istog, odnosno nije cilj biti vezan za QGIS kao aplikativni sloj već da baza bude nezavisna i spremna za uključivanje sisteme sa SOA arhitekturom.

#### 2.5.1. *Stvaranje prostornih indekasa*

Prostorni indeksi su u suštini boundingbox's geometrije koji menja geometriju i koji pomažu algoritmu poređenja da sa manje podataka tj upita odredi da li je geometrija npr. unutar druge ili sječe drugu. Pošto mogu imati više država, p.o., k.o., svakako ne želim svaku porediti sa svakom a pogotovo je bitno pri proveri da li se parcela preklapa sa drugom.

Funkcije koje budem koristio

- ST\_Within
- ST\_Dwithin
- ST\_Overlaps
- ST\_Intersect
- ST\_Touches
- ST\_Contains

Sve automatski koriste prostorne indekse. Da automatski koriste spatial index znači da koriste spatial index geometrijske kolone koja se navodi u argumentu, ne na način da one formiraju. To znači da geometrijske kolone prethodno moraju da imaju kreirane prostorne indekse!

```
CREATE [ UNIQUE ] INDEX [ CONCURRENTLY ] [ [ IF NOT EXISTS ] name ] ON [ ONLY ] table_name [ USING method ]
  ( { column_name | ( expression ) } [ COLLATE collation ] [ opclass [ ( opclass_parameter = value [, ... ] ) ] ] [ WHERE condition ] )
```

Sika 20. Sinopsis za kreiranje indeksa

Bitno mi je sledeće:

- UNIQUE – ne dopuštam kreiranje npr. prekopljениh parcela, **gist** ne dopušta ovu opciju

- IF NOT EXISTS – da ne vraća upozorenje ako je prostorni indeks kreiran pod istim nazivom, ovo neće biti navedeno kako bi se osigurao od grešaka jer će naziv name biti dodeljivan ručno
- USING metod – postoji više metoda za stvaranje prostornog indeksa, različitih oblika , npr. btree, hash,gist i sl. a ovdje je to **gist**
- Column\_name - naziv geometrijske kolone

Tako da sljedi

```

1  DROP INDEX IF EXISTS "my_katastarskaparcela_polygon";
2  DROP INDEX IF EXISTS "my_katastarskaparcela_point";
3  DROP INDEX IF EXISTS "my_cadastral_zone";
4  CREATE INDEX "my_katastarskaparcela_polygon" ON public.katastarskaparcela USING GIST("geometry");
5  CREATE INDEX "my_katastarskaparcela_point" ON public.katastarskaparcela USING GIST("referencePoint");
6  CREATE INDEX "my_cadastral_zone" ON public.cadastralzone USING GIST("geometry");

```

**Slika 21.** kreiranje prostornih inekasa

Da zaključim poglavljje, ideja spatial index's je da je brži algoritam koji poredi četiri tačke sa četri tačke nego n sa m. Pažnja, to ne znači da se neće poređati i n sa m, već na taj način brže izdvaja one zapise, kandidate za poređenje složenijim algoritmom! Postoje **operator's** koji doslovno samo koriste spatial index's , npr **&&**.

#### 2.5.2. Jedinica nižeg ne smije biti van granica jedinice višeg nivoa

Ovdje su bitne sledeće pretpostavke:

- Najviša jedinica nema višu, time se za nju uslov ne proverava
- Svaka jedinica niža u hirerarhiji treba da ima fk više jedinice
- Pošto su admin. jedinice na istom sloju u QGIS-u moraću da kreiram rupe za vidljivost

Ovaj i sledeći problem znače uslov da jedinica bude u dodana u bazu, pa se stoga koristi BEFORE INSERT trigger, a zato je i ime funkcije trigger upravo **adjustNewCadastralZone**.

```

204
205 CREATE OR REPLACE FUNCTION my_adjustNewCadastralZone() RETURNS TRIGGER AS
206   $$
207   DECLARE
208     C4 CURSOR (mojaklasa BIGINT) FOR SELECT * FROM public.cadastralzone WHERE public.cadastralzone."jedinicaVisegNivoa"=mojaklasa;
209     r record;
210   begin
211     if my_upperZoneId(NEW."level",NEW."geometry") is null and set NEW."level"=1 then PERFORM my_hirearhijaJedinicaKjeDobra();
```

...ovom zapisu da se sekvenca ponosa kan da nije ni dobrejia nema "id\_cz"

```

212     else NEW."jedinicaVisegNivoa" = my_upperZoneId(NEW."level",NEW."geometry");
```

**end if;**
213
214 **OPEN** (**C4**(**new**."**jedinicaVisegNivoa**"));
215 **FETCH FIRST** **FROM** **c4** **INTO** **r**;
216 **while** **FOUND**
217 **loop**
218 **IF** **st\_overlaps**(**new**."**geometry**",**r**."**geometry**") **THEN** **PERFORM** my\_jedinicasePreklapaju();
**END IF;**
219 **FETCH NEXT** **FROM** **c4** **INTO** **r**;
220 **end loop;**
221
222 **CLOSE** **c4**;
223 **return** **NEW**;
224 **end;**
225 \$\$;
226 **language plpgsql;**
227
228
229 **CREATE OR REPLACE TRIGGER** bf\_adjustNewCadastralZoneTrigger
230 **BEFORE INSERT** **ON** CadastralZone
231 **FOR EACH ROW** **EXECUTE FUNCTION** my\_adjustviewCadastralZone();
232
233
234

**Slika 22.** Realizacija provere sadržanosti u hirerarhiji

Dakle, jedinica koja nije u višoj jedinici ne dobija ključ iste, samo jednostavno. Funkcija koja se ovdje koristi je na slici 21.

```

146 CREATE OR REPLACE FUNCTION my_upperZoneId(myLevel integer, geom geometry("MULTIPOLYGON",3908)) RETURNS BIGINT AS
147   $$
148
149   begin
150
151   return "Id_cz" FROM public.CadastralZone WHERE CadastralZone."level" = myLevel+1 AND
152   ST_Within(ST_AsText(geom),concat('POLYGON(',concat(substr(ST_AsText(ST_InteriorRingN(CadastralZone."geometry",1)),11),'))'));
153   end;
154   $$;
155   language plpgsql;
156

```

**Slika 23.a** pozvana funkcija 1.

```

166 CREATE OR REPLACE FUNCTION my_hirearhijaJedinicaKjeDobra() RETURNS VOID AS
167   $$
168
169   begin
170   perform setval"cadastralzone_Id_cz_seq";regclass.currentval"cadastralzone_Id_cz_seq";Valid;
171   raise exception 'Hirerarhija jedinica nije topoloski dobar!';
172   end;
173   $$;
174   language plpgsql;
175

```

**Slika 23.b** pozvana funkcija 2

### 2.5.3. administrativne jedinice se ne smiju sjeći na istom nivou

Optimizacija je moguća, tako da se testira na overlap samo sa postojećim u svojoj klasi ekvivalencije (jedinici), kako je već obezbeđeno da je potpuno unutar svoje jedinice, a čime prvi put i potreba za hirerarhijom dolazi do primene.

```

204
205 CREATE or replace FUNCTION my_adjustNewCadastralZone() returns TRIGGER as
206   $$
207   DECLARE
208   c4 CURSOR (mojaklasa bigint) FOR SELECT * FROM public.cadastralzone WHERE public.cadastralzone."jedinicaViseNivoa"=mojaklasa;
209   r record;
210   begin
211
212   if my_upperZoneId(NEW."level",NEW."geometry") is null and not NEW."level"=3 then PERFORM my_hirearhijaJedinicaNijeDobra();
213   --ovo znaci da se sekvenca ponosi kao da niti ni dodjela nov."id_cz"
214   else NEW."jedinicaViseNivoa" = my_upperZoneId(NEW."level",NEW."geometry");
215   end if;
216
217   OPEN c4(NEW."jedinicaViseNivoa");
218   FETCH FIRST FROM c4 INTO r;
219   while FOUND
220   loop
221   IF st_overlaps(new."geometry", r."geometry") THEN PERFORM my_jediniceSePreklapaju();
222   END IF;
223   FETCH NEXT FROM c4 INTO r;
224   end loop;
225
226   CLOSE c4;
227
228   RETURN NEW;
229 end;
230 $$;
231 language plpgsql;
232
233
234 CREATE or replace TRIGGER b1_adjustNewCadastralZoneTrigger
235 BEFORE INSERT ON CadastralZone
236 FOR EACH ROW EXECUTE FUNCTION my_adjustNewCadastralZone();
237
238 CREATE or replace FUNCTION my_adjustUpdatedCadastralZone() returns TRIGGER as
239   $$
240   begin
241   NEW."jedinicaViseNivoa" = my_upperZoneId(NEW."level",NEW."geometry");
242   RETURN NEW;
243   end;
244

```

Slika 24.a Provjera granica admin. jedinica

Bez poznavanja atributa osnovnaJedinicaVlašništva **ovo ne bi bilo moguće.**

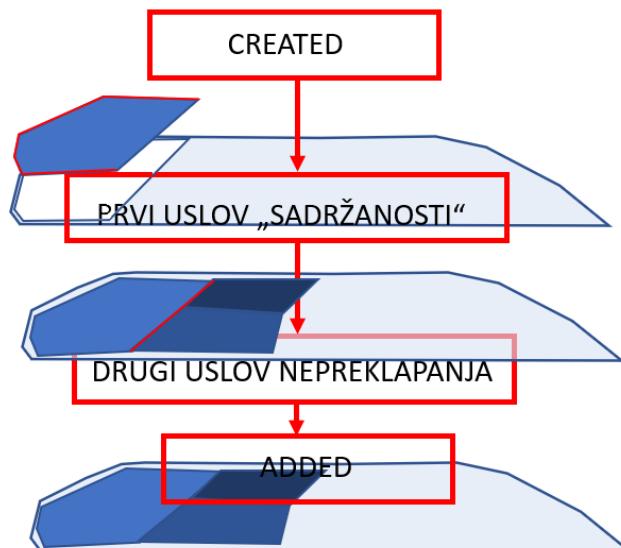
Pozvana funkcija ima oblik

```

178 create or replace function my_jediniceSePreklapaju() returns void as
179   $$
180
181   begin
182
183   perform setval(''cadastralzone_Id_cz_seq''::regclass,currval(''cadastralzone_Id_cz_seq''::regclass),false);
184   raise exception 'Jedinice se preklapaju!';
185
186
187   end;
188   $$;
189   language plpgsql;
...

```

Slika 24.b Pozvana funkcija za oporavak i poruku o grešci



Slika 25. Naglasak na integraciji

#### 2.5.4. Parcele se ne smiju seći

Ovdje su bitne sledeće prepostavke:

- Veliki broj parcela n (milion, bilion) je spor za poređenje 1:n
- Parcele nemoraju pripasti nijednoj K.O. ako se nalaze na granici K.O.

Predpostavke već su rešene kreiranjem prostornih indekasa.

```
85 ✓ create or replace function parcelaPreklapaNekuDruguParcelu() returns void as
86 $$
87 declare
88 begin
89 perform setval('katastarskaparcela_extId_kp_seq'||::regclass,currval('katastarskaparcela_extId_kp_seq'||::regclass),false);
90 raise exception 'Parcelska preklapa neku drugu parcelu!';
91 end;
92 $$ language plpgsql;
93
94 ✓ CREATE or replace FUNCTION my_topoParcela() returns trigger as
95 $$
96 declare
97 c5 CURSOR (geom geometry('POLYGON',3986)) FOR SELECT * FROM public.katastarskaparcela WHERE ST_Overlaps(geom,katastarskaparcela."geometry");
98 begin
99    OPEN c5;
100   loop
101      if found then
102        IF FOUND THEN PERFORM parcelaPreklapaNekuDruguParcelu(); -- ponasa se kao da nije ni dodijelen, tj. is_called = false
103      END IF;
104      CLOSE c5;
105      return new;
106   end loop;
107   $$ language plpgsql;
108
109 ✓ CREATE or replace TRIGGER bi_topoParcelaTrigger
110 BEFORE INSERT ON public.katastarskaparcela
111 FOR EACH ROW EXECUTE FUNCTION my_topoParcela();
112
```

Slika 26. Imp. algoritma problema preklapanja parcela

Dakle BEFORE INSERT i ROW kao parametri kako bih za svaku parcelu to proverio.

#### 2.5.5. prekid u nizu indekasa

Primenjeni su sledeći koncepti:

- dinamičko kreiraњe i execute
- while loop kontrolna struktura

Bitna su sledeća ograničenja za planiran algoritam:

- promenjiva tipa record (ili %rowtype) je samo indikatorska promenjiva , stvarni pristup zapisu na koji ukazuje cursor se ostvaruje kroz current of
- svi kurzori u međusobno povezanim metodama (metode delegiraju u drugim metodama) moraju imati jedinstven naziv (npr. c1,c2,c3 ... cn) kako ne bi došlo do sudaranja portala/konekcija

Sledeći slučajevi su uzeti u obzir:

- Izbrisani je zadnji zapis
- Izbrisani je zapis koji nije zadnji
- Izbrisani je zapis nakon čega je tablica prazna

Iz ovih navigacija sljedi rešenje prikazano na slikama koje sljede.

```

288 ✓ create or replace function my_normalizeIsEmpty() returns trigger as
289   $$
290   DECLARE
291   r record;
292   c CURSOR FOR SELECT * FROM public.cadastralzone;
293 ✓ begin
294   OPEN c;
295   FETCH FIRST FROM c INTO r;
296
297   IF not FOUND THEN ALTER SEQUENCE "cadastralzone_Id_cz_seq" RESTART WITH 1;
298   end if;
299   close c;
300   return old;
301 end;
302 $$
303 language plpgsql;
304
305 ✓ create or replace TRIGGER ad_normalizeIsEmptyTrigger
306 after delete on cadastralzone
307 for each statement execute function my_normalizeIsEmpty();
308
309 ✓ create or replace function my_sortiraj(i bigint) returns void as
310   $$
311   declare
312   c2 CURSOR (x bigint) FOR SELECT * FROM public.cadastralzone WHERE public.cadastralzone."Id_cz">>x ;
313   r4 public.cadastralzone%rowtype;
314   idx bigint :=1;
315 ✓ begin
316   OPEN c2(idx);
317
318
319   FETCH FIRST FROM c2 INTO r4;
320
321   while FOUND
322   loop
323     UPDATE public.cadastralzone SET "Id_cz"=idx WHERE CURRENT OF c2;
324     idx:=idx+1;
325     FETCH NEXT FROM c2 INTO r4;
326     end loop;
327
328   EXECUTE 'ALTER SEQUENCE "cadastralzone_Id_cz_seq" RESTART WITH'|| idx;-- zašto ne idx-1 je objeњen
329   CLOSE c2;
330   end;
331   $$
332 language plpgsql;
333
334 ✓ CREATE or replace function my_normalizeIdCadastralZone() returns trigger as
335   $$
336   declare
337   c1 CURSOR FOR SELECT * FROM public.cadastralzone;-- zapisi nakon brisanja
338   r1 public.cadastralzone%rowtype;
339   pom bigint = old."Id_cz"-1;
340   begin
341   OPEN c1;
342   FETCH LAST FROM c1 INTO r1;
343   IF not FOUND THEN ALTER SEQUENCE "cadastralzone_Id_cz_seq" RESTART WITH 1;
344   ELSIF r1."Id_cz">=pom THEN EXECUTE 'ALTER SEQUENCE "cadastralzone_Id_cz_seq" RESTART WITH'|| old."Id_cz"
345   ELSE PERFORM my_sortiraj(OLD."Id_cz");
346   END IF;
347   CLOSE c1;
348   return old;
349 end;
350 $$
351 language plpgsql;
352
353 ✓ CREATE or replace TRIGGER bd_normalizeIdInCadastralZoneTrigger
354 AFTER DELETE ON cadastralzone
355   -- KLJUCNO ZBOG OGРАНИЧЕЊА PRIMATY KEY, DA SE NEDESI DA JOS UVJEK ZAPIS KOJI SE SMATRA IZBRISANIM SA O
356   -- FOR EACH ROW EXECUTE FUNCTION my_normalizeIdCadastralZone();
357
358

```

Slika 27. SQL

```

362 ✓ create or replace function my_sortirajParcele(i bigint) returns void as
363   $$
364   declare
365     c19 CURSOR (x bigint) FOR SELECT * FROM public.katastarskaparcela WHERE public.katastarskaparcela."extId_kp">x ;
366     r4 record;
367     idx bigint :=i;
368   ✓ begin
369     OPEN c19(idx);
370   371
372     FETCH FIRST FROM c19 INTO r4;
373   374
375   ✓ while FOUND
376     loop
377       UPDATE public.katastarskaparcela SET "extId_kp"=idx WHERE CURRENT OF c19;
378       idx:=idx+1;
379       FETCH NEXT FROM c19 INTO r4;
380       end loop;
381
382       EXECUTE 'ALTER SEQUENCE "katastarskaparcela_extId_kp_seq" RESTART WITH '|| idx;-- zašto ne idx-1 je obješnjeno u elaboratu, prij
383       CLOSE c19;
384     end;
385   $$
386   language plpgsql;
387
388
389   ✓ CREATE OR REPLACE FUNCTION my_normalizeIdCadastralParcel() RETURNS TRIGGER AS
390   $$
391   declare
392     c20 CURSOR FOR SELECT * FROM public.katastarskaparcela;-- zapisi nakon brisanja
393     r1 record;
394     pom bigint = old."extId_kp"-1;
395   ✓ begin
396     OPEN c20;
397     FETCH LAST FROM c20 INTO r1;
398     IF not FOUND THEN ALTER SEQUENCE public."katastarskaparcela_extId_kp_seq" RESTART WITH 1;
399     ELSEIF r1."extId_kp=pom THEN EXECUTE 'ALTER SEQUENCE "katastarskaparcela_extId_kp_seq" RESTART WITH '|| old."extId_kp"; -- zašto
400     ELSE PERFORM my_sortirajParcele(OLD."extId_kp");
401   END IF;
402   CLOSE c20;
403   return old;
404   end;
405   $$
406   language plpgsql;
407
408   ✓ CREATE OR REPLACE TRIGGER ad.ParcelIdTrigger
409   AFTER DELETE ON public.katastarskaparcela -- KLJUCNO ZBOG OGRANICENJA PRIMATY KEY,
410   -- DA SE NEDESI DA JOS UVJEK ZAPIS KOJI SE SMATRA IZBRISANIM SA OLD NALAZI U TABLI
411   FOR EACH ROW EXECUTE FUNCTION my_normalizeIdCadastralParcel();
412   -- OVO ZA STATEMENT MENI JE POSEBNO ZANIMLJIVO, VOLIO BIH SKRENUTI PAZNJU NA ODBRANI
413   -- NA ZNACENJE (konkretno optimizacija)
414

```

Slika 28. SQL

Sva objašnjenja su data u komentarima. Bitno je da je parametar trigger's AFTER jer je id označen kao PRIMARY KEY pri kreiranju baze, tako da ne dolazi do duplih indeksa prilikom penošenja id obrisanog zapisa sledećem u redu.

Pažnja, postojala je greška u kodu, da se provjeri je li zapis iz portala dodeljen u promenjivu tipa record ili table%rowtype koristi se FOUND, ne 'record is not null'.

### 2.5.6. Problem referencijalnog integriteta i hierarhije

Osnovna funkcija koja se koristi je ST\_Dwithin, nikako &&.

Problemi referencijalnih integriteta, veza:

- u slučaju brisanja K.O., npr.izmena i dopuna granica, brišu se i svi preneseni ključevi u sadržanim parcelama ( problem agregacije), problem ponovnog manuelnog unosa
- u slučaju brisanja više admin. jednice, sve niže sadržane jedinice potrebno je opet dodefinisati novom višom jedinicom, slično problemu parcela
- veliki broj veza geoobjekata otežava hand-handling update verzija celina

Predpostavke za kod su:

- Niža hiearhija pripada potpuno višoj
- Najviša hiearhija nema višu
- Parcele imaju kao višu hir. K.O.
- Pri izmjeni id tokom normalizacije npr. K.O. parcele trebaju biti ažurirane
- Administrativne jednice su u suštini rupe, pogledaj test pprimjer

Dakle parametri za svaki trigger's su:

- AFTER DELETE
- AFTER UPDATE

```

128 ✓ CREATE or replace FUNCTION my_zoniranje(geom geometry("POLYGON",3908), visinivo integer) returns bigint as
129   $$
130 begin
131   return "Id_cz" from CadastralZone where ST_Within(geom,CadastralZone."geometry") and CadastralZone.level=visinivo;
132   ntje reformiran poligon jer mi razvojene klase omogucuju razvojene slojeve i time nepreklopanje
133 end;
134 $$;
135 language plpgsql;
136
137 ✓ CREATE or replace FUNCTION my_setDefaultZoneForCadastralParcel() returns TRIGGER as
138   $$
139 begin
140   NEW."fk_cz" := my_zoniranje(NEW.geometry,1);
141   return NEW;--zapis koji je insertovan pa izmenjen ovom metodom
142   --sada mozes da cinis tablicu KatastarskaParcela, NEW!
143   return NEW;
144 end;
145 $$;
146 language plpgsql;
147
148 ✓ CREATE or replace TRIGGER bi_setDefaultZoneForCadastralParcelTrigger
149 BEFORE INSERT ON public.KatastarskaParcela
150 FOR EACH ROW EXECUTE FUNCTION my_setDefaultZoneForCadastralParcel();
151
152 ✓ CREATE or replace FUNCTION my_upperZoneId(myLevel integer, geom geometry("MULTIPOLYGON",3908)) returns bigint as
153   $$
154 begin
155   return "Id_cz" from public.CadastralZone where CadastralZone."level" = myLevel+1 and
156   ST_Within(ST_AsText(geom),concat('POLYGON(',concat(substr(ST_AsText(ST_InteriorRingN(CadastralZone."geometry"),1),11),'')));
157 end;
158 $$;
159 language plpgsql;
160
161 ✓ CREATE or replace FUNCTION my_lowerZoneId(myLevel integer, geom geometry("MULTIPOLYGON",3908)) returns TABLE( "Idx" bigint ) as
162   $$
163 begin
164   if ST_NRings(geom)=2 then return query select"Id_cz" from public.CadastralZone where CadastralZone."level" = myLevel-1 and ST_Within(ST_AsText(CadastralZone."geometry"),
165   end if;-- ovaj if je zbog k.o. koje nemaju ring
166   end;
167   $$;
168 language plpgsql;
169
170 ✓ CREATE or replace FUNCTION my_adjustNewCadastralZone() returns TRIGGER as
171   $$
172 declare
173   c4 CURSOR (mojaKlasa bigint) FOR SELECT * FROM public.cadastralzone WHERE public.cadastralzone."jedinicaViseNivoa"=mojaKlasa;
174   r record;
175   begin
176   if my_upperZoneId(NEW."level",NEW."geometry") is null and not NEW."level"=3 then PERFORM my_hiearhijaJedinicaNijeDobra();
177   else
178   zbroj_znaci da se servisna ponosa kao da nije ni dodala nov "id_cz"
179   else New."jedinicaViseNivoa" = my_upperZoneId(NEW."level",NEW."geometry");
180   end if;
181
182   OPEN c4(new."jedinicaViseNivoa");
183   FETCH FIRST FROM c4 INTO r;
184   while FOUND
185   loop
186   IF st_overlaps(new."geometry", r."geometry") THEN PERFORM my_jedinicaSePreklapaju();
187   END IF;
188   FETCH NEXT FROM c4 INTO r;
189   end loop;
190
191   CLOSE c4;
192   return NEW;
193 end;
194 $$;
195 language plpgsql;
196
197 ✓ CREATE or replace TRIGGER bi_adjustNewCadastralZoneTrigger
198 BEFORE INSERT ON CadastralZone
199 FOR EACH ROW EXECUTE FUNCTION my_adjustNewCadastralZone();
200

```

Slika 29.a Inicijalizacija, početna dodatak identifikatora pri kreiranju

Dakle ovdje su parametri za trigger's samo BEFORE i INSERT.

Sada ide serija slika koje pokazuju kod za ponovnu dodelu prenesenih ključeva nakon DELETE, UPDATE ili naknadnog umetanja tj. INSERT nekog zapisa.

```

199
200 ✓ CREATE or replace FUNCTION my_updateLoewrLevelCadastralParcel() returns TRIGGER as
201 $$ begin
202
203     UPDATE cadastralzone
204     SET "jedinicaviseNvoa" = new."Id_cz"
205     where "Id_cz" in (select my_lowerZoneId(new."level", new."geometry"));
206     return new;
207 end;
208 $$;
209
210 language plpgsql;
211
212
213
214
215
216 ✓ CREATE or replace TRIGGER ai_updateLowerLevelCadastralParcelTrigger-- zeznuo, trbalo je ići LevelZone, ne LevelCadastralParcel
217 after INSERT ON CadastralZone -- za isti zapis koji je prethodno INSERTOVAN , sirono
218 FOR EACH ROW EXECUTE procedure my_updateLoewrLevelCadastralParcel();
219
220 --ovjajanjenje sledećg triggera se nalazi u slijenom bloku triggera koji se odnosi na parcele
221 ✓ CREATE or replace TRIGGER au_updateLowerLevelCadastralParcelTrigger
222 AFTER UPDATE ON CadastralZone -- ukratko, UPDATE će biti samo za indeks od strane triggera normalize
223 FOR EACH ROW EXECUTE FUNCTION my_updateLoewrLevelCadastralParcel();
224
225
226 ✓ CREATE or replace FUNCTION my_updateCadastralParcelToNewZone() returns TRIGGER as
227 $$ begin
228
229     UPDATE KatastarskaParcela SET "fk_cz" = NEW."Id_cz" where ST_Within(KatastarskaParcela."geometry",NEW."geometry") and NEW."level"=1; -- jedinica višeg nivoa koja je verov
230     return NEW;-- imao sam katastrofalni previd, propust, nisam vidio da ako ne stavim dodatni uslov NEW."Id_cz"=1 može se desiti da se i pri novoj političkoj opštini dodanc
231 end;
232 $$;
233 language plpgsql;
234
235
236 ✓ CREATE or replace TRIGGER ai_updateCadastralParcelToNewZoneTrigger
237 AFTER INSERT ON CadastralZone -- za isti zapis koji je prethodno INSERTOVAN , sirono
238 FOR EACH ROW EXECUTE procedure my_updateCadastralParcelToNewZone();
239 -- medutim, parcele treba azurirati i pri projen zone pod kojom se nalaze, tj normalizaciji idx-a
240 ✓ CREATE or replace TRIGGER au_updateCadastralParcelToNewZoneTrigger
241 AFTER UPDATE ON CadastralZone -- opet after da bi sve bilo sirono, a NEW u trigger funkciji je update-ovana zona, slično kao prethodno novodata
242 FOR EACH ROW EXECUTE FUNCTION my_updateCadastralParcelToNewZone();

```

**Slika 30.a** Oporavak referenci

Dakle svi trigger's su parametrizirani za:

- AFTER INSERT
- AFTER UPDATE

Ne i nakon AFTER DELETE jer mi ne treba pogrešna referenca. AFTER jer je ovo posljedica dodavanja, nikako BEFORE jer dok npr. Politicka Opstina bude dodana mora proći proces provere topologije opisan prije (BEFORE).

```

15 ✓ create or replace function my_verzioniranjeINSPIRE() returns trigger as
16 $$ declare
17     crs1 CURSOR FOR SELECT * FROM public.osnovnjedinicavlasnistva;
18     crs2 CURSOR FOR SELECT * FROM public.cadastralzone;
19     r1 record;
20     r2 record;
21
22 begin
23     OPEN crs1;
24     OPEN crs2;
25     FETCH first FROM crs1 INTO r1;
26     FETCH first FROM crs2 INTO r2;
27
28     -- nakon brisanja parcele koja pripada jedinici
29     -- menjam verziju toj admin. jedinici, jedinici vlasnistva
30     ✓ IF r1 is not null THEN UPDATE cadastralzone SET cadastralzone."version" = cadastralzone."version"+1 WHERE ST_Contains(cadastralzone."geometry",OLD."geometry") and
31     cadastralzone."level"=1;--nije reformiran poligon jer mi razdvojene klase
32     END IF;
33
34     --omoguju razdvojene slojeve i time neprekapanje
35     ✓ IF r2 is not null THEN UPDATE OsnovnaJedinicaVlasnistva SET OsnovnaJedinicaVlasnistva."version"=OsnovnaJedinicaVlasnistva."version"+1 FROM katastarskaparcela WHERE
36     OsnovnaJedinicaVlasnistva."id"=katastarskaparcela."fk_olv";
37     END IF;
38
39     CLOSE crs1;
40     CLOSE crs2;
41     return OLD;
42 end;
43 $$;
44 language plpgsql;
45
46 ✓ CREATE or replace TRIGGER ad_INSPIRE_CadastralZonalVersion
47 AFTER DELETE ON public.katastarskaparcela
48 FOR EACH ROW EXECUTE FUNCTION my_verzioniranjeINSPIRE();-- ključno je da bude za svaki zapis, ne operaciju

```

**Slika 31.** Verzioniranje

### 2.5.7. Problem površina

Bitna je predpostavka da poligon može ili mora imati rupu. Funkcije korištene su:

- ST\_Area
- ST\_AsText
- Concat
- Substr
- ST\_ExteriorRing
- ST\_InteriorRingN
- ST\_Nrings

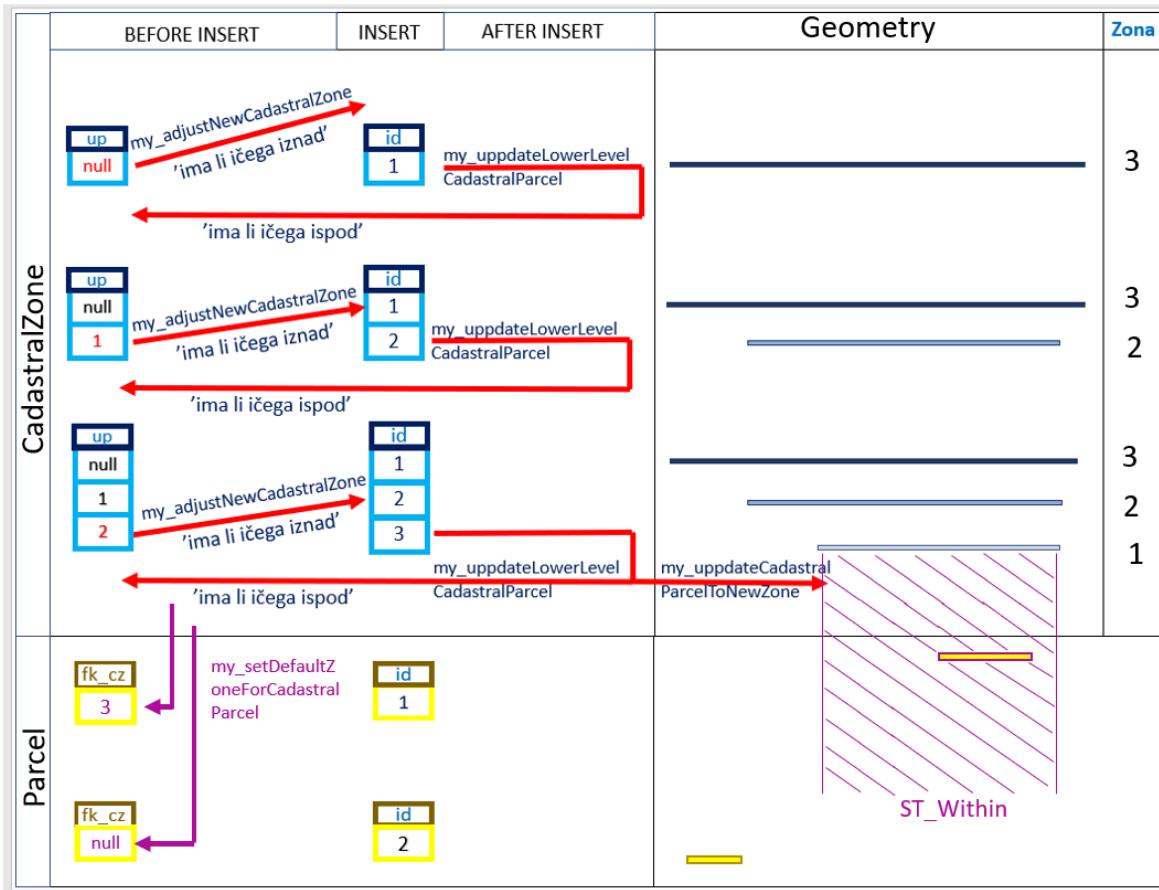
Na osnovu toga sljedi kod na slici sa komentarima.

```

1
2 ✓ CREATE or replace FUNCTION my_povrsina_cz() returns trigger as
3   $$ begin
4
5   IF ST_NRings(NEW."geometry")=2 THEN
6     NEW."povrsina":= ST_Area(concat('POLYGON(',concat(substr(ST_AsText(ST_ExteriorRing(NEW."geometry")),11),'))'));
7   ELSE NEW."povrsina":=ST_Area(NEW."geometry");
8   END IF;
9   return NEW;
10 end;
11 $$ language plpgsql;
12
13 ✓ CREATE or replace FUNCTION my_povrsina_cp() returns trigger as
14   $$ declare
15   base decimal(36,2);
16   begin
17
18   IF ST_NRings(new."geometry">>1 THEN |
19     base = ST_Area(concat('POLYGON(',concat(substr(ST_AsText(ST_ExteriorRing(NEW."geometry")),11),'))'));
20
21   FOR i IN 1..ST_NRings(new."geometry")-1
22     LOOP
23       base = base-ST_Area(concat('POLYGON(',concat(substr(ST_AsText(ST_InteriorRingN(NEW."geometry",i)),11),'))'));
24   END LOOP;
25   NEW."povrsina" = base;
26   ELSE NEW."povrsina" = ST_Area(NEW."geometry");
27   END IF;
28
29   return NEW;
30 end;
31 $$ language plpgsql;
32
33 CREATE or replace TRIGGER bi_povrsina_CZ BEFORE INSERT ON public.cadastralzone FOR EACH ROW EXECUTE FUNCTION my_povrsina_cz();
34 CREATE or replace TRIGGER bi_povrsina_CP BEFORE INSERT ON public.katastarskaparcela FOR EACH ROW EXECUTE FUNCTION my_povrsina_cp();
35 CREATE or replace TRIGGER bu_povrsina_CP BEFORE UPDATE ON public.katastarskaparcela FOR EACH ROW EXECUTE FUNCTION my_povrsina_cp();
36
37
38
39
40

```

**Slika 32.** Račun površina prije kreiranja parcele



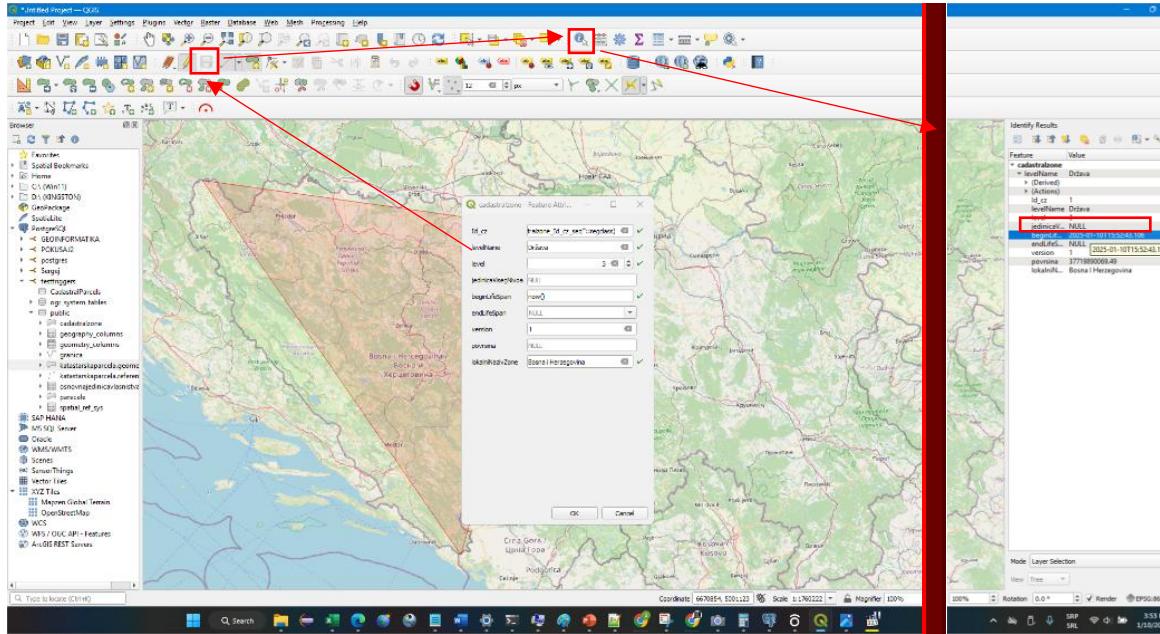
Slika 33. Diagram sljeda aktivnosti sql koda (ne UML diagram aktivnosti) u cilju bolje intuicije

## 2.6. QGIS konekcija i TEST

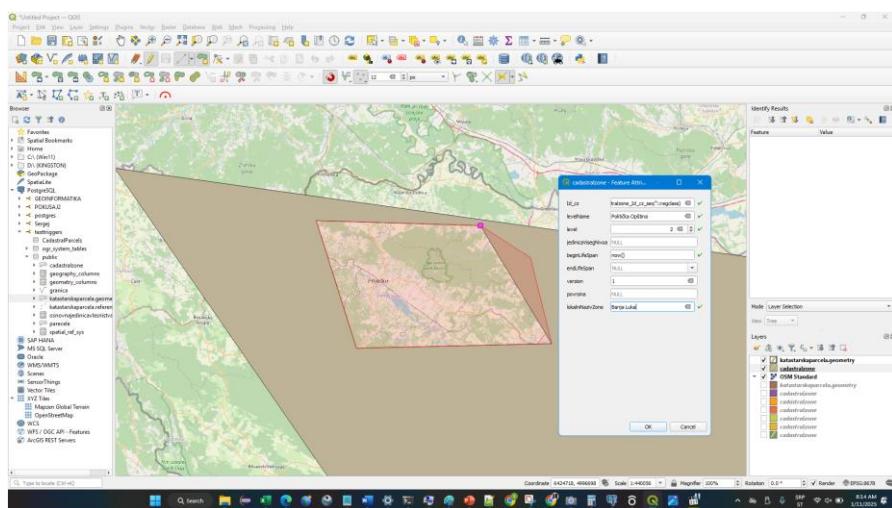
Za početak sljedim sledeće korake:

- Kreiranje jedinica – slika 34,
- Test algoritma sa slika 22-26. na neispravnim topo. – slike od 34 do 37
- Test triggera sa slike 29. za autometsko proslijedivanje fk– slike 37,38
- Test triggera, funkcija i procedura sa slika 27 i 28 – slike 39 do 41.
- Slika 42 test funkcije zoniranje parcela sa slike 29, slika 45. je test svih funkcija sa slike 26. za topologiju parcele, 46. je primena elemenata objašnjениm u slopu slike 28. tj. neprekidnost indekasa parcela

- Slika 47. pokazuje kako funkcioniše au/ai-updateCadastralPrcelToNewZoneTrigger i au/ai – updateLowerLevelCadastralParcelTrigger
- Slika 48. i Slika 49. predstavljaju test sql koda sa slike 32.

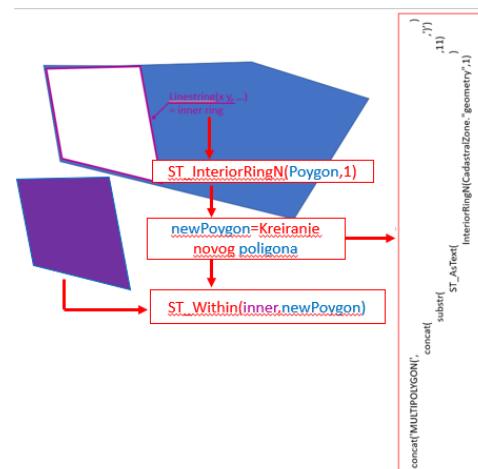


Slika 34. Primer kreiranja zone

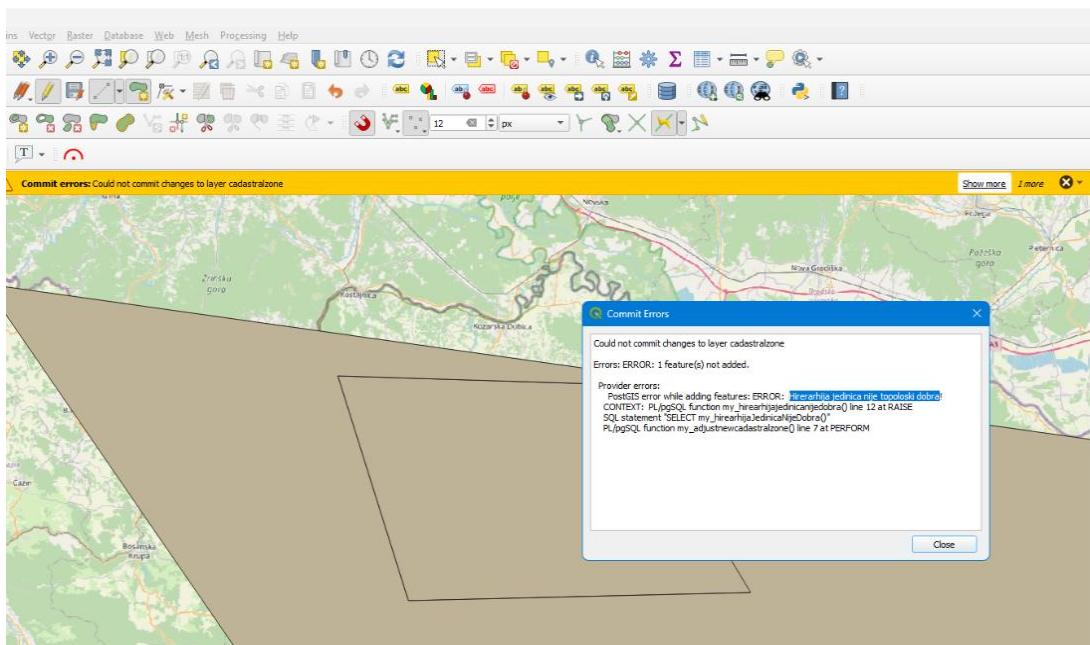


Slika 34.a Primer pogrešne topologije, pogledaj narednu sliku

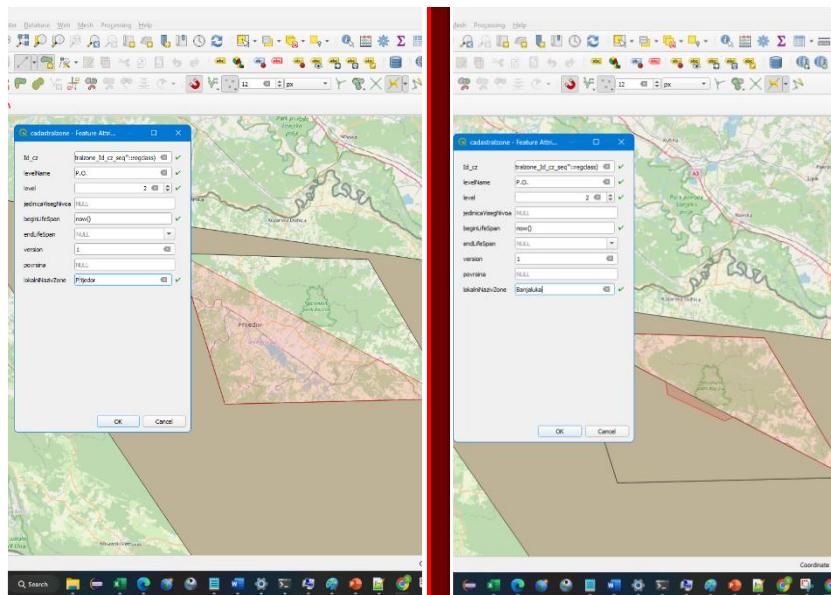
Radi bolje intuicije, pristup koji je omogućio predhodna dva-tri koraka , tj da se bez obzira na rupu dobije poređenje sadržanosti u vanjskom poligону je



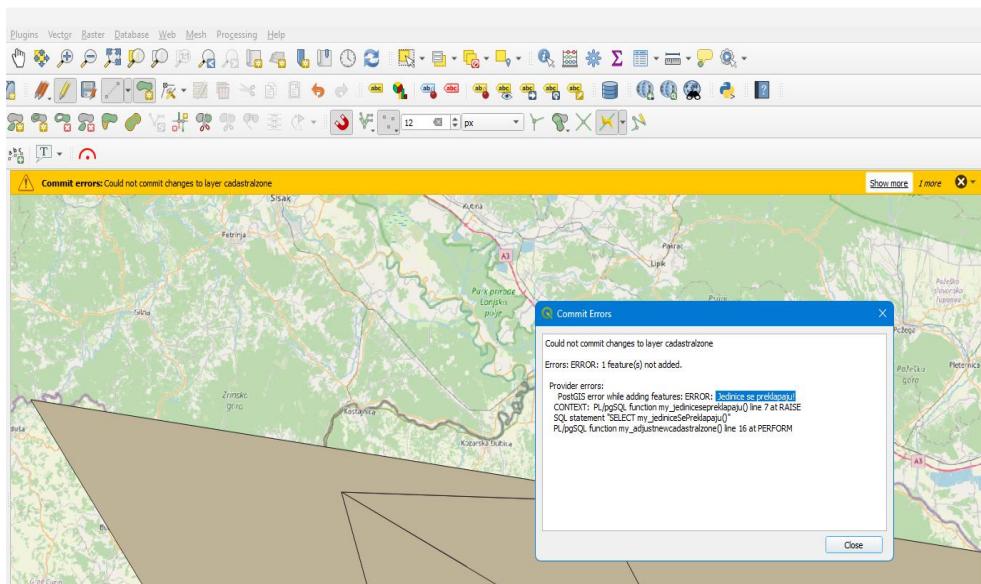
**Slika 34.b** Način na koji je dobijena prostorna relacija (poređenje)



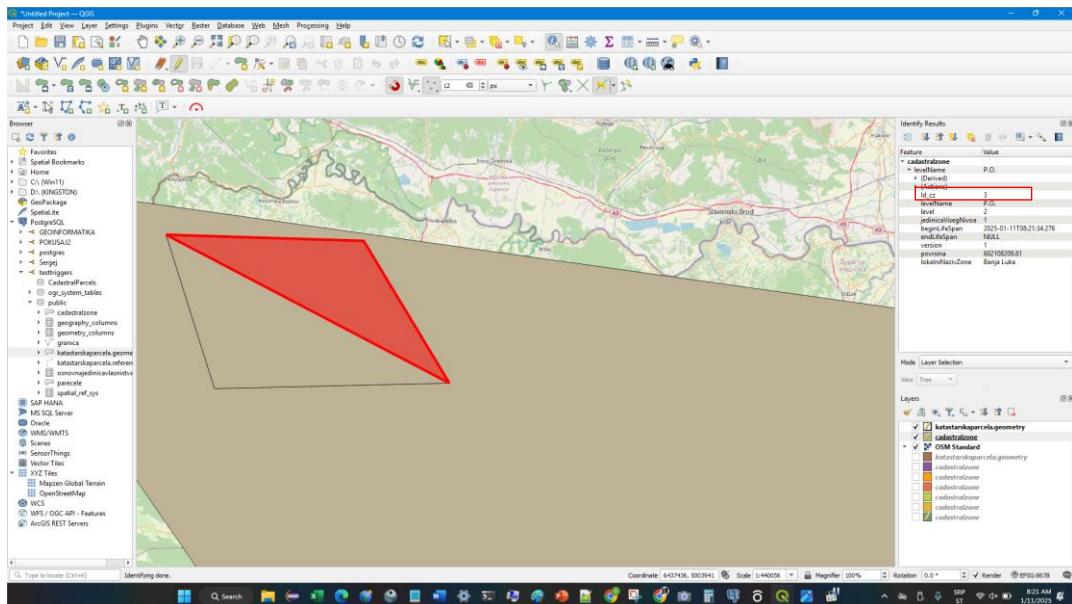
**Slika 35.** Onemogućen unos geoobjekta sa pogrešnom topo. - poruka



Slika 36. Drugi vid pogrešne topologije unutar granica

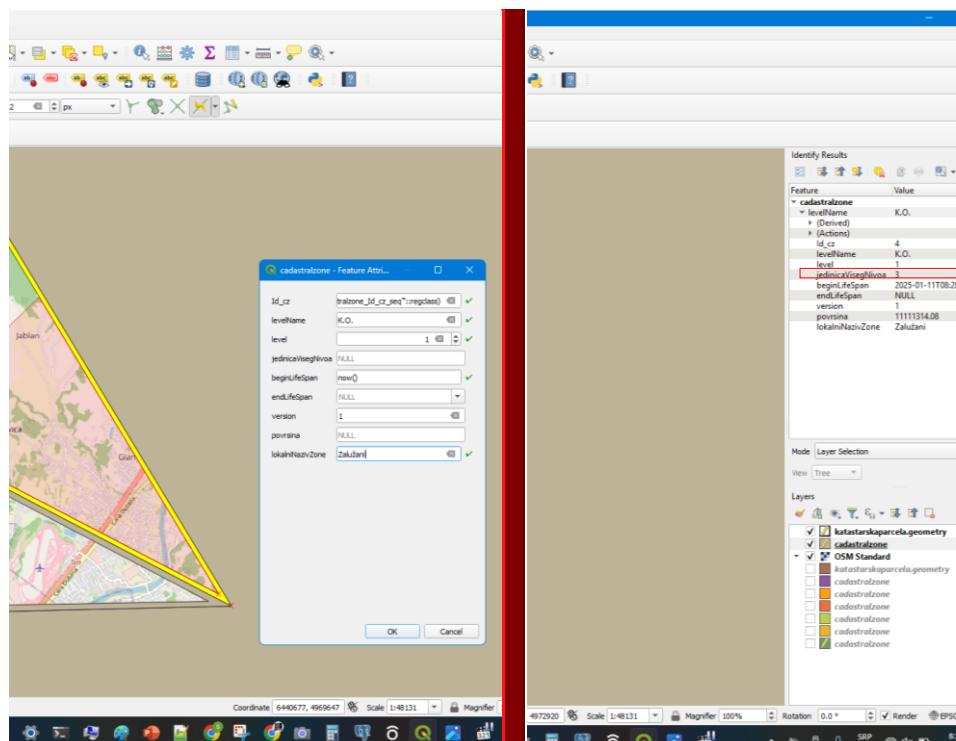


Slika 37. Onemogućen unos i poruka o tipu greške Kreiraču opet P.O. Banjaluka ali unutar granica.

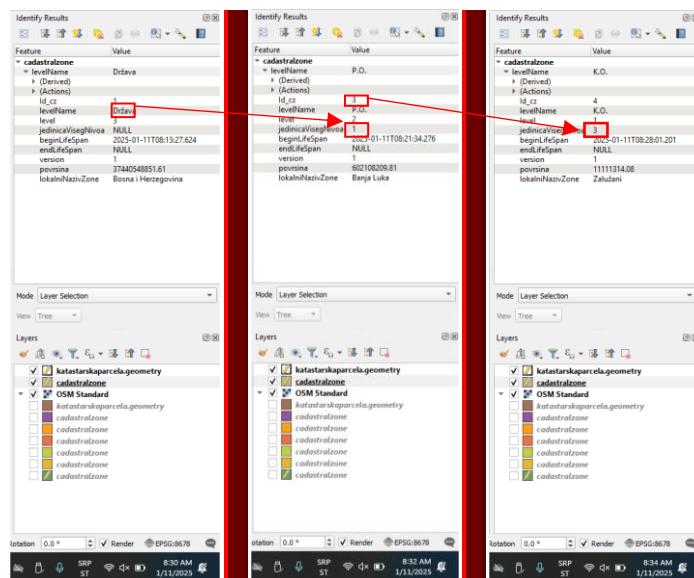


Slika 38.a Primer delovanja triggera normalizeIdInCadastralZoneTrigger

Dakle nakon 1 'otkazanog' koji je trebao imati id =3, ipak će sledeći imati id =3 a ne id =4 što bi se desilo bez ikakvog algoritma u bazi jer bi se sekvenca monotno rastući nastavila inkrementovat smatrajući da je id 3 već dodeljen.



Slika 38.b primjer triggera bi\_adjustNewCadastralZoneTrigger



Slika 38.c Retrospektiva

Još uvjek imam zapis sa id. = 2 ali ta P.O. je kreirana samo za primer presecanja jedinica pa je postrane, tako da će je obrisati:

cadastralzone — Features Total: 4, Filtered: 4, Selected: 1								
123Id_cz	levelName	level	edinicaVisegNivoz	beginLifeSpan	endLifeSpan	version	povrsina	lokalmNazivZone
1	Država	3	NULL	2025-01-11T08...	NULL	1	37440548851.61	Bosna i Herze...
2	P.O.	2	1	2025-01-11T08...	NULL	1	795359530.39	Prijedor
3	P.O.	2	1	2025-01-11T08...	NULL	1	602108209.81	Banja Luka
4	K.O.	1	3	2025-01-11T08...	NULL	1	11111314.08	Zalužani

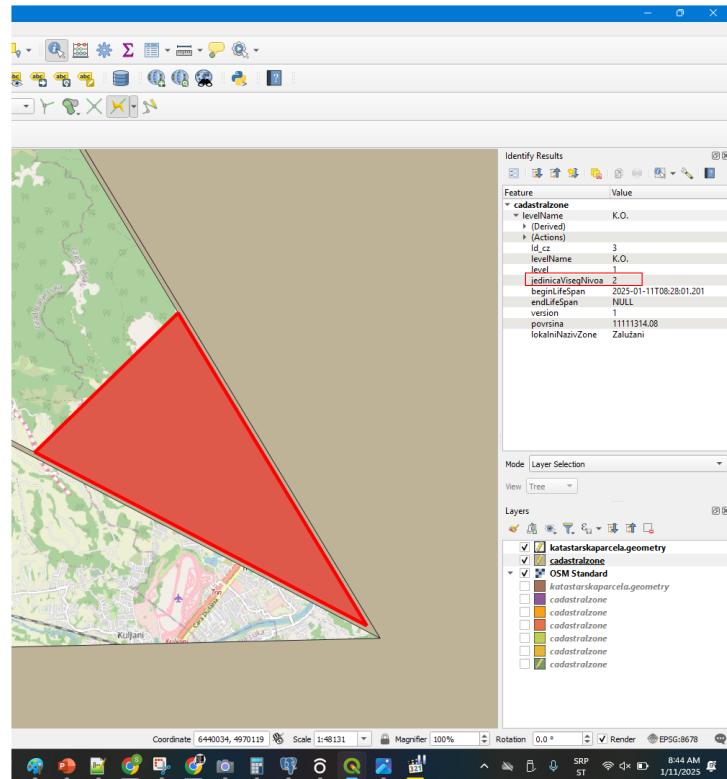
Slika 39. Stanje prije brisanja jedinice Prijedor

Nakon što obrišem taj zapis stanje će se samo ažurirati

123Id_cz	levelName	level	edinicaVisegNivoz	beginLifeSpan	endLifeSpan	version	povrsina	lokalmNazivZone
1	Država	3	NULL	2025-01-11T08...	NULL	1	37440548851.61	Bosna i Herze...
2	P.O.	2	1	2025-01-11T08...	NULL	1	602108209.81	Banja Luka
3	K.O.	1	2	2025-01-11T08...	NULL	1	11111314.08	Zalužani

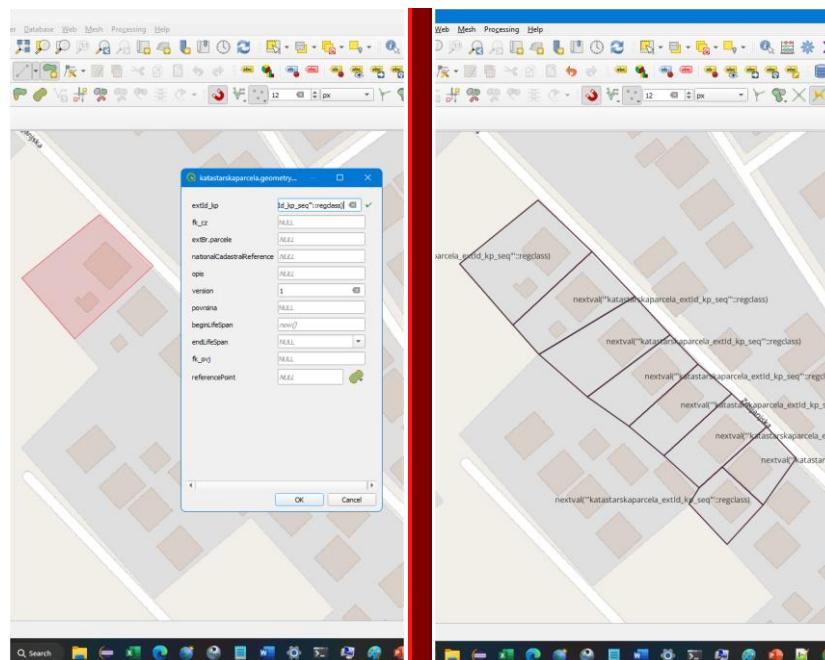
Slika 40. Stanje poslje brisanja međuzapisa Prijedor

Također se u tom slučaju vrši update svih prenesenih ključeva



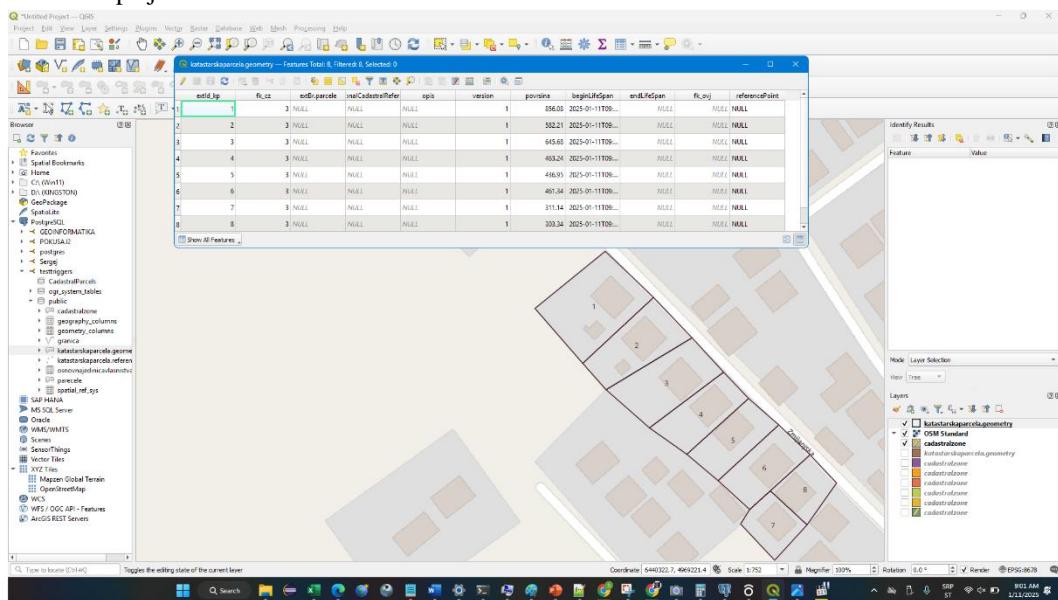
**Slika 41.** Primer kako se katastarska ažurirala, dobila nov id

Za daljni nastavak kao prvi layer postavljam OSM mapu, kako bih ispravno obišao konture nekih parcela ili objekata.

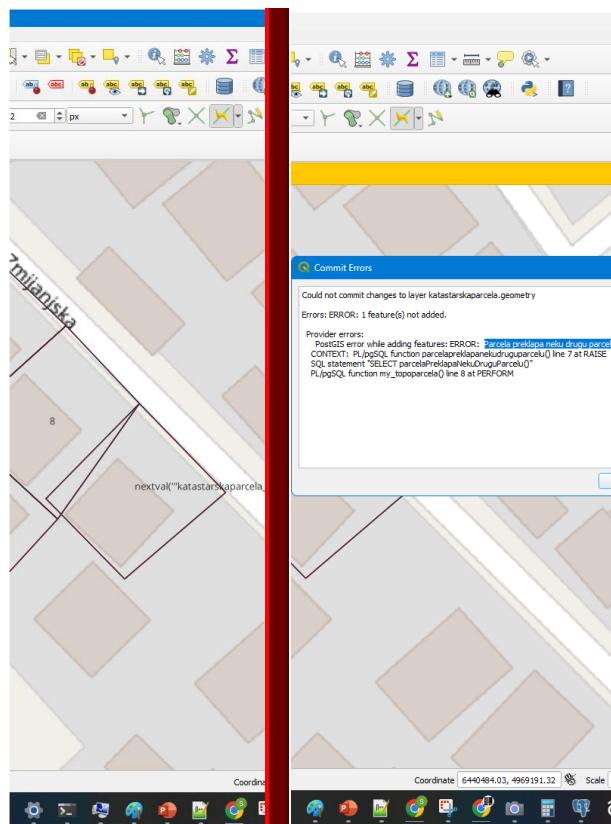


**Slika 42.** Primjer korisne primjene sequence objekta

Odabiranjem opcije **save**

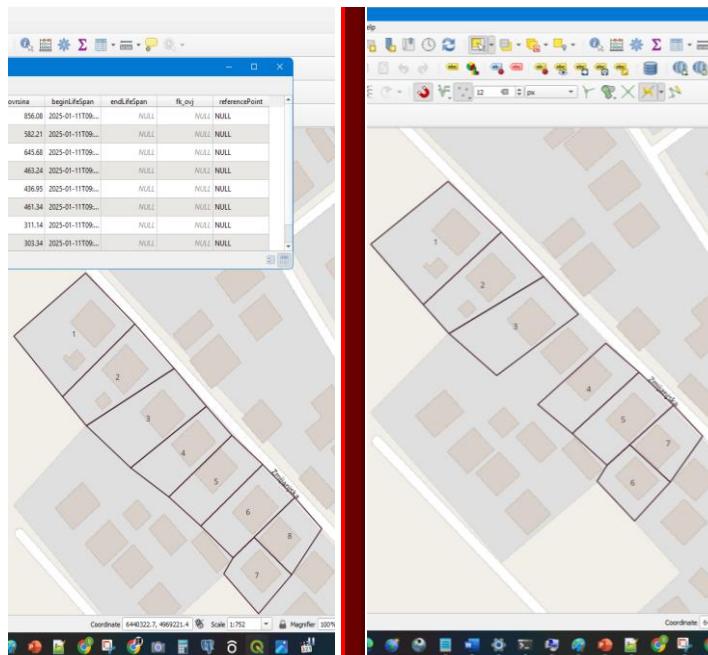


Slika 43. commit da bi vidljili rezultat



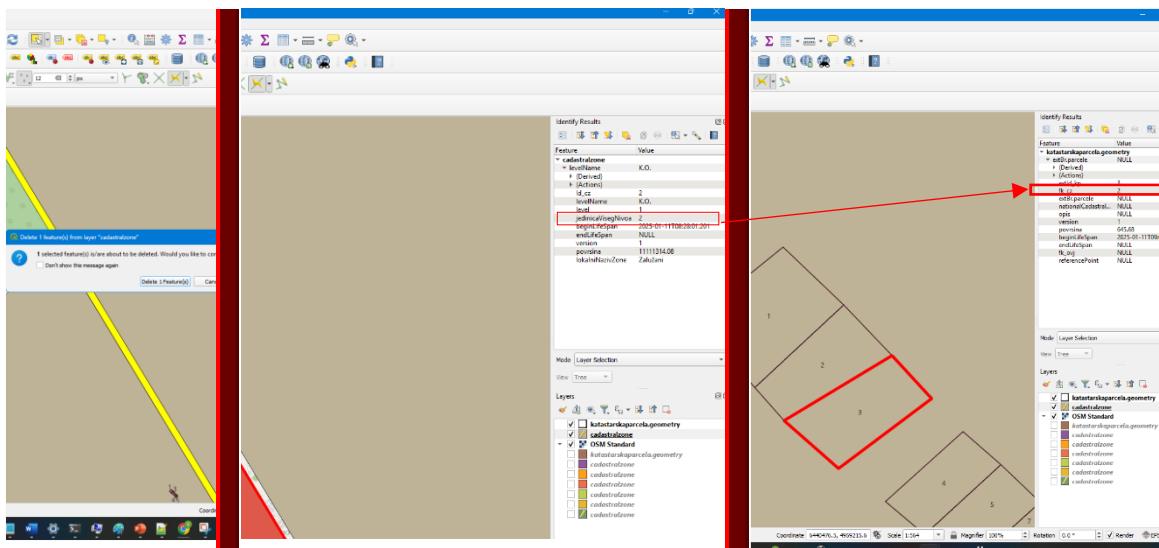
Slika 44. Pokušaj

U slučaju brisanja neke parcele, npr. one sa id =4, nakon odabira opcije **save** sljedi



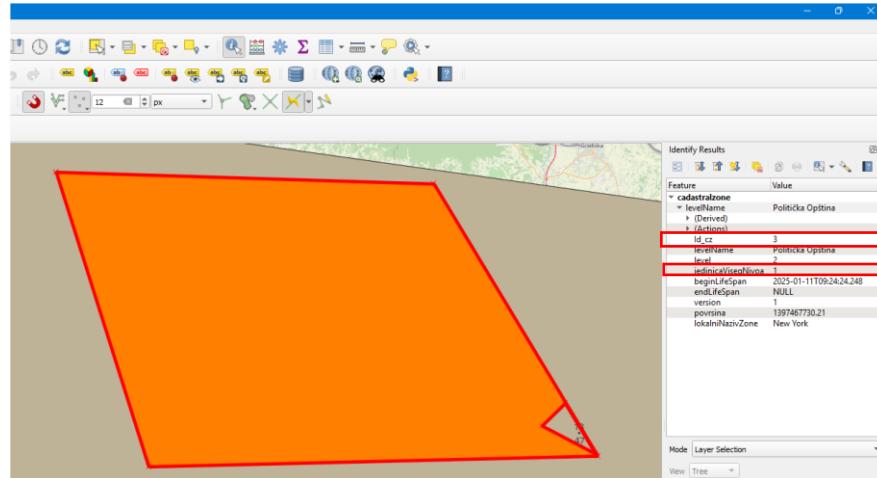
Slika 45. prije i poslije brisanja

Također, nakon promjene indetifikaotra (id) neke opštine, uslijed neke druge promjene ili direktno dolazi do update-a prenesenog ključa koji govori u kojoj se K.O. parcele nalaze.



Slika 46. normalizovanje indeksa parcela

Također se može opet dodeliti politička opština i ona će 'povći' prenesen ključ postojeće države i ažurirati niže jedinice nakon svog kreiranja, naravno koje sadrži u potpunosti.

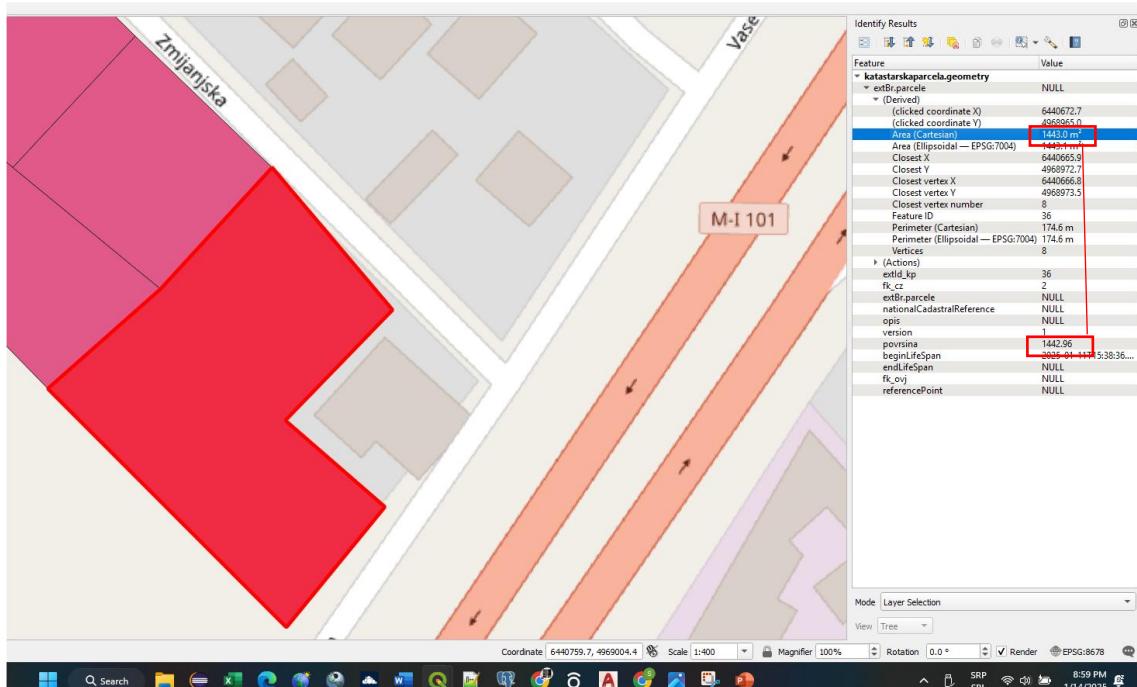


**Slika 47.** au/ai-updateCadastralPrceIToNewZoneTrigger i au/ai – updateLowerLevelCadastralParcelTrigger

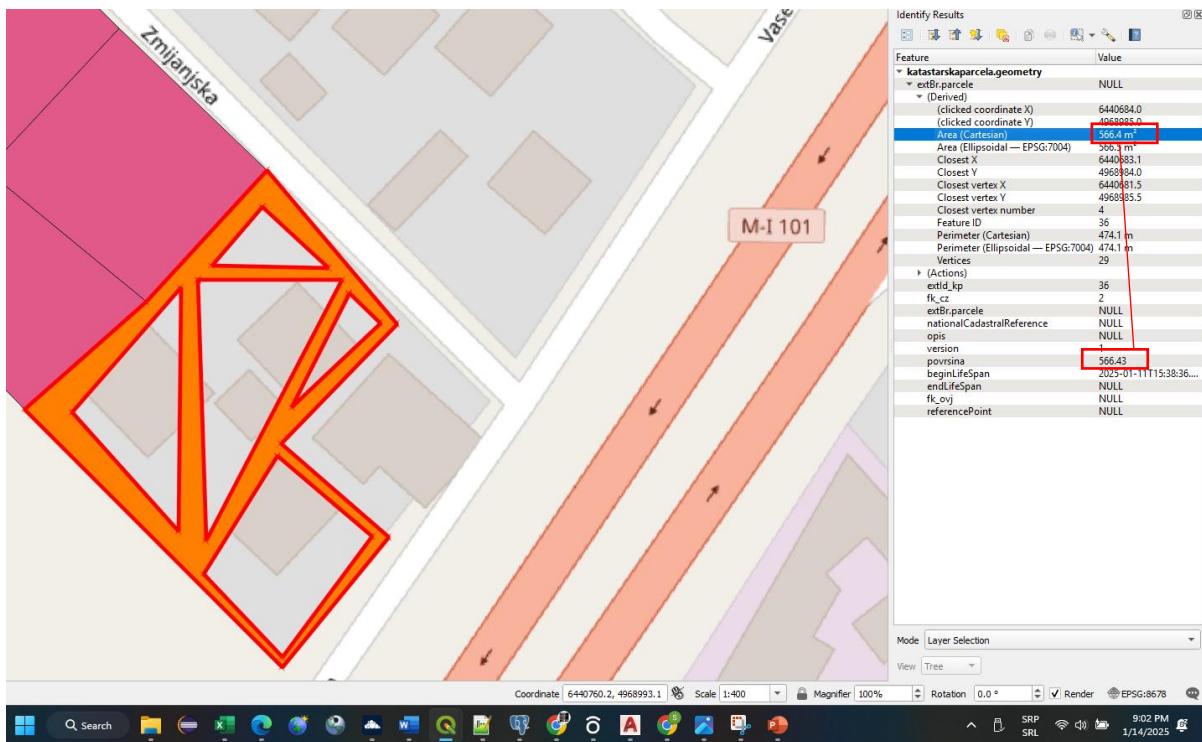
a da vratim pažnju još jednom, id je također ispravno dodeljen bez obzira na sve izmene.

Kao što je prikazano u prethodnom poglavljtu, površine se ododeljuju pri samom kreiranju parcele. Ovo je značajno, jer funkcije \$Area u QGIS-u su strogo vezane za QGIS sučelje. Ova baza podataka je time spremnija za direktnu integraciju u SOA serversku arhitekturu, bazi podataka se prisupa preko nekog drugog aplikativnog nivoa koji nije QGIS. Sljedeće slike:

- Slika 48. – prikazuje površinu parcele izračunatu pre dodavanja rupa i usporedbu sa površinom koju daje QGIS (Derived) radi kontrole – naznačeno crvenim pravougaoncima
- Slika 49. – prikazuje površinu parcele ažuriranu posle dodavanja niza rupa (InteriorRings) i usporešuje sa izračunatom u QGIS-u – naznačeno crvenim pravougaoncima



**Slika 48.** before update



Slika 49. after adding hole

### 3. UPITI

- Prikazati sve parcele kroz koje prolazi osa saobraćajnice idejnog projekta u cilju planiranja komasacije i elaborata isplativosti.

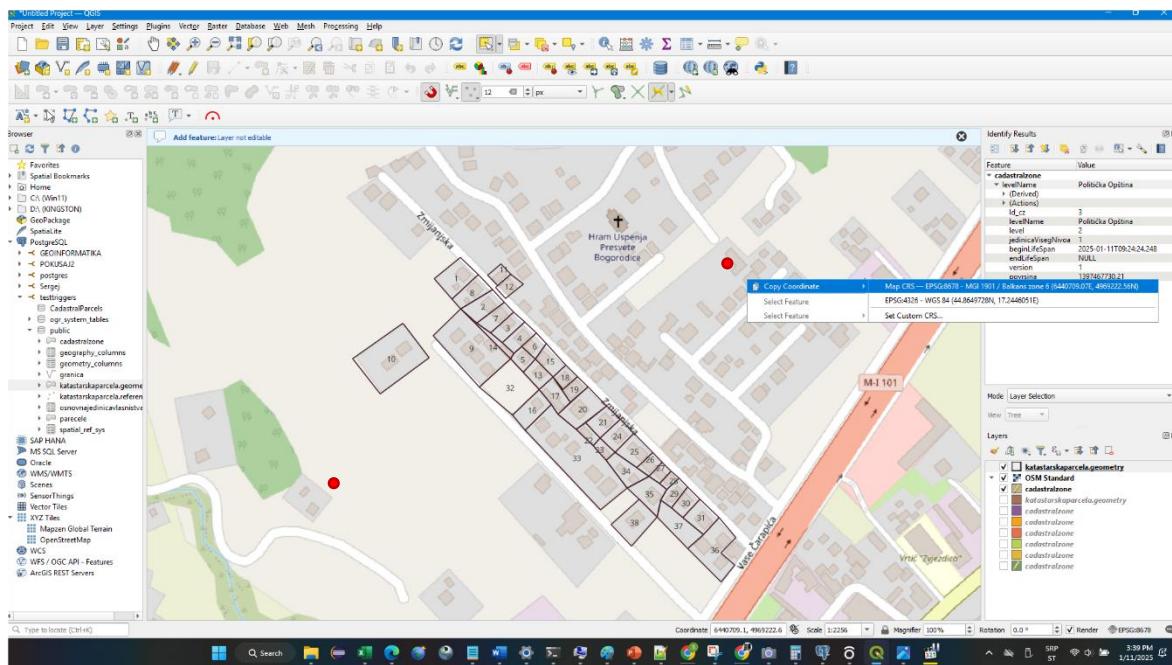
```

8 ✓ create or replace function komasacija(pocX decimal(18,2),pocY decimal(18,2),krajX decimal(18,2),krajY decimal(18,2)) returns TABLE("idx" bigint) as
9   $$ 
10  declare
11    line_text = concat('LINESTRING(',concat(concat(pocX::text,' '),concat(pocY::text,concat(' ',concat(concat(krajX,' '),concat(krajY::text,''))))))));
12    SAO geometry('LINESTRING',3908) := ST_GeomFromText(line,3908);
13  begin
14    --intersects nepozne jer bi ukljucio i touches ondoso dodirivanje linije i parcele a to nije problem
15    return query select "extId_kp" from public.katastarskaparcela where ST_Crosses(geometry,SAO);--merenje rastojanja ima smisao samo u crs=3908
16
17
18  end;
19  $$ 
20 language plpgsql;

```

Slika 50 . Rešenje za prvi upit

Sada je potrebno prikupiti koordinate na OSM mapi, biram opciju select i desnim klikom na praznu površinu odabiram koordinate, i to u crs3908.



Slika 51. Pripreme za upit

Sada da vidimo koje su to parcele

```

select komarsacija(6440427.32,4969875.12,6440680.42,4969234.50);
--6440427.32,4969875.12
--6440680.42,4969234.50

```

komarsacija	bigint
1	18
2	15
3	32

Slika 52. Parcele koje treba komasirati ako trasa bude odabrana

, i ovo rešenje se očigledno vidi sa slike 51.

2. Neka je minimalna dozvoljena površine k.p.  $250 \text{ m}^2$ . Pronaći sve parcele u K.O. Zalužani koje su manje od dozvoljene površine.

```

--prikazati sve parcele u opštini Zalužani koje su manje od dozvoljene min površine 200 m2 po kartezijanskim koordinatama
SELECT "extId_kp",katastarskaparcela."povrsina",katastarskaparcela."beginLifeSpan","opis" FROM public.katastarskaparcela,public.cadastralzone
WHERE katastarskaparcela."povrsina"<250.00 and "fk_cz"="Id_cz" and UPPER("lokalnaNazivZone") IN ('ZALUZANI','ZALUZANI') ORDER BY povrsina ASC;

```

	extId_kp	povrsina	beginLifeSpan	endLifeSpan	opis
1	28	160.09	2025-01-11 15:37:26.26209		[null]
2	23	211.74	2025-01-11 15:37:26.26209		[null]
3	27	215.65	2025-01-11 15:37:26.26209		[null]
4	14	220.63	2025-01-11 15:34:49.831136		[null]
5	22	229.47	2025-01-11 15:37:26.26209		[null]
6	11	237.30	2025-01-11 15:34:35.018121		[null]

**Slika 53.** Rezultat upita za limit