

Gospodarska
zbornica
Slovenije



Chamber of Commerce
and Industry of Slovenia



KCUI
Kompetenčni
center za
umetno
inteligenco

Slovenian
National
AI Competence
Centre

Semantična priprava podatkov za učinkovitejšo uporabo umetne inteligence

<https://ul-fri-kcui.github.io/semantic-data/>

dr. Dejan Lavbič in dr. Slavko Žitnik (UL FRI)



REPUBLIKA SLOVENIJA
MINISTRSTVO ZA NOTRANJE ZADEVE
IN JAVNO UPRAVO



REPUBLIKA SLOVENIJA
MINISTRSTVO ZA IZOBRAŽEVANJE, ZNANOST
IN MLADINO



REPUBLIKA SLOVENIJA
MINISTRSTVO ZA GOSPODARSTVO
DELO IN ŠPORT



REPUBLIKA SLOVENIJA
MINISTRSTVO ZA OBRAMBO







Agenda

1. Uvod: Zakaj semantično opisovanje podatkov?
2. Od izvornih podatkov (1 ★) do odprtih podatkov (3 ★)
3. Raven 4 ★: semantični opis podatkov z RDF
4. Formalizacija podatkovnega modela z ontologijami
5. Raven 5 ★: povezani podatki in zunanji viri
6. Uporaba semantično opisanih podatkov s SPARQL
7. Sklepanje: prehod od podatkov k znanju
8. Za konec: kdaj se semantika izplača



1. Uvod: Zakaj semantično opisovanje podatkov?



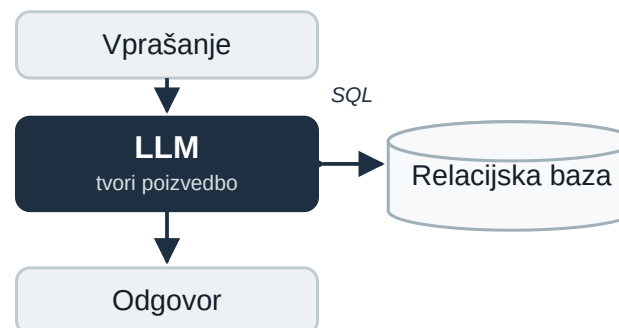
GenUI in semantične tehnologije

a Samo LLM



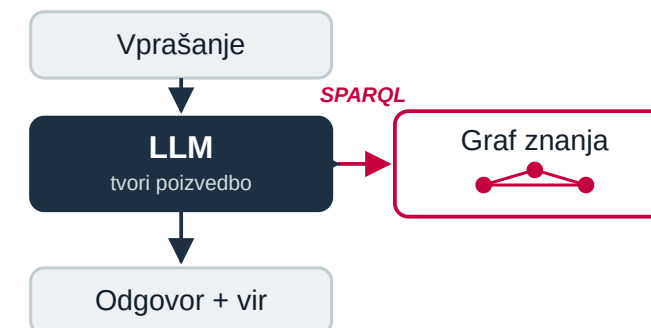
- ⚠ znanje je iz učenja in lahko zastarelo,
- ⚠ ni vira in ni sledljivosti odgovora.

b LLM z relacijsko bazo



- ✓ prave in aktualne vrednosti iz baze,
- ⚠ shema sama ne nosi pomena (POL_TYP_CD).

c LLM z grafom znanja



- ✓ pojmi imajo pomen in so povezani med viri,
- ✓ odgovor je sledljiv do izvornega trojčka.

d LLM kot del agentnega sistema (MCP)



- ✓ agent hkrati doseže več virov in sam izbere orodje,
- ✓ poleg odgovora lahko izvede tudi dejanje,
- ⚠ zato potrebuje tipiziran model objektov,
- ⚠ opredeljene dovoljene akcije in revizijsko sled.

Semantika tu ni okras, ampak pogoj za zaupanje dejanju.



Namen delavnice

Problem > tehnologija

- Zakaj objava podatkov ni dovolj?
- Kaj dejansko rešuje semantika?
- Pomen interoperabilnosti in ponovne uporabe podatkov.



Motivacija

Podatki obstajajo – odgovori na vprašanje pa ne

- Razpršeni podatki,
- različni viri,
- različni pomeni.



Od podatkov do znanja

Struktura ≠ pomen

- **Podatek** (npr. 298.000),
- **Informacija** (Število prebivalcev Ljubljane.),
- **Metapodatek** (Kaj ta številka pomeni, za katero leto, vir, metodologija.),
- **Znanje** (Sklepanje, povezovanje, odgovori na kompleksna vprašanja, npr. Ljubljana je največje mesto v Sloveniji z 298.000 prebivalci (podatki iz leta 2023, SURS).).



Zakaj je to pomembno?

Interoperabilnost in ponovna uporaba

- veliko podatkov,
- veliko podvajanja,
- malo povezovanja.



Kaj ni semantično opisovanje?

Pogoste zmote

- ✗ *“Samo za raziskovalce.”*
- ✗ *“Najprej moramo razviti ontologijo.”*
- ✗ *“Preveč kompleksno za praktično uporabo.”*



Ključni pojmi - poenostavljeno

Enostavne analogije

- URI
- RDF
- Ontologija
- SPARQL
- Linked Data



Primer iz prakse

Semantika dandanes

- Spletni članki,
- iskanje,
- povezovanje vsebin.



INTERAKTIVNA VAJA

Kaj pove JSON-LD? ↗

```
{
  "@context": "https://schema.org",
  "@type": "NewsArticle",
  "mainEntityOfPage": {
    "@type": "WebPage",
    ...
  },
  "headline": ...,
  "image": ...,
  "datePublished": "2026-09-16 06:53:59",
  "author": { ... },
  "publisher": {
    "@type": "Organization",
    "name": "RTV Slovenija",
    "logo": { ... }
  }
}
```



2. Od izvornih podatkov (1 ★) do odprtih podatkov (3 ★)



Model 5 ★ odprtosti podatkov

Stopnja	Zvezdice	Raven	Značilnosti podatkov
1	★	Objavljeni podatki	PDF, HTML, poročila
2	★★	Strukturirani podatki	Excel, lastni formati
3	★★★	Odprti podatki	CSV, strojno berljivi formati
4	★★★★	Semantično opisani podatki	RDF, URI-ji, ontologije
5	★★★★★	Povezani podatki	Zunanji viri, Linked Data



Izvorni podatki (1 ★ in 2 ★): ko podatki obstajajo, a niso uporabni

- Podatki CRP iz  NIO [↗](#),
 -  Nabor podatkov NIO CRP.xls [↗](#),
 -  Sifranti IO CRP.xls [↗](#),
- pomanjkljivosti,
 - razpršeni po datotekah in zavihkih,
 - pomeni stolpcev ni jasen,
 - brez konteksta,
 - iste vrednosti → različen koncept.



Prehod v odprti format (3 ★)

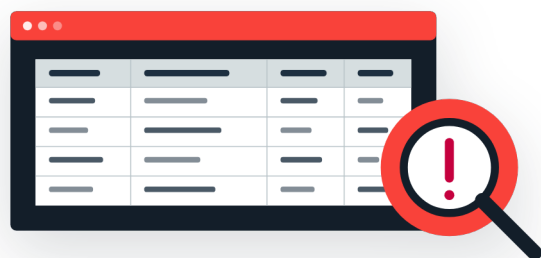
- Pretvorba v **CSV** (odprt in strojno berljiv format), kot prvi korak k tehnični interoperabilnosti,
- preprost 2D besedilni zapis (vrstice, stolpci, ločila),
- izboljša **dostopnost**, ne pa še **pomena**.



Podatki SURS (3 ★)

- Podatki o prebivalcih po občinah ( [SURS ↗](#), 2025),
- prenos v **odprtem formatu (CSV/TSV)**,
- vrednosti in kratice stolpcev ( [data.tsv ↗](#)),
- pomen stolpec zapisan **ločeno** ( [info.txt ↗](#)),
- **odprti podatki brez vgrajene semantike.**

INTERAKTIVNA VAJA



Kaj nam TSV ne pove? [↗](#)



Omejitve ravni 3 ★

- Odprti in strojno berljivi podatki (CSV),
- brez skupnih identifikatorjev,
- različna imena in strukture,
- pomen ni formalno opredeljen,
- **struktura ≠ semantika.**

CRP · izvoz iz ( Excelsa ↗)

OBC_ID	OBC_IME_SLO_MLC
1	Ajdovščina

SURS · ( data.tsv ↗)

ob_id	ob_ime	tot_p
001	Ajdovščina	19895

1 ↔ 001: ista občina, različna imena stolpcev in različna oblika identifikatorja.



3. Raven 4 ★: semantični opis podatkov z RDF



Zakaj CSV ni več dovolj?

- Odprti in strojno berljivi podatki (3 ★),
- brez pomena za računalnik,
- enaki nizi ≠ ista entiteta,
- manjkajo enolični identifikatorji,
- potrebujemo **URI-je**, **RDF** in **formalni pomen**.





RDF: opis podatkov v obliki trojčkov

- **Osebek** – kaj opisujemo,
- **povedek** – katero lastnost opisujemo,
- **predmet** – vrednost ali povezava,
- eksplicitno izražen pomen,
- podatki kot **graf**, ne tabela.

SURS ·  [data.tsv ↗](#)

ob_id	ob_ime	tot_p
001	Ajdovščina	19895

Ajdovščina (osebek) → tot_p (povedek) → 1985 (vrednost).



URI: enolična identifikacija pojmov

Od ID-ja v tabeli do globalnega pomena

- Enolični in stabilni identifikatorji,
- globalno naslovljivi (splet),
- osnova povezovanja podatkov.

 <https://sl.wikipedia.org/wiki/Ajdovščina> ↗

 <https://dbpedia.org/page/Ajdovščina> ↗



Zapis RDF v obliki Turtle (TTL)

- RDF ima več zapisov (serializacij),
- **Turtle (TTL)** je najbolj berljiv,
- pogosto uporabljen v praksi.

```
<https://onto.mdp.gov.si/obcina/ajdovscina> <https://onto.mdp.gov.si/shema/steviloPrebivalcev> 19895 .
```

```
@prefix obcina: <https://onto.mdp.gov.si/obcina/> .  
@prefix shema: <https://onto.mdp.gov.si/shema/> .  
  
obcina:ajdovscina shema:steviloPrebivalcev 19895 .
```

```
<?xml version="1.0"?>  
<rdf:RDF  
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"  
  xmlns:shema="https://onto.mdp.gov.si/shema/"  
  <rdf:Description rdf:about="https://onto.mdp.gov.si/obcina/ajdovscina">  
    <shema:steviloPrebivalcev rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">  
      19895  
    </shema:steviloPrebivalcev>  
  </rdf:Description>  
</rdf:RDF>
```



TSV → RDF

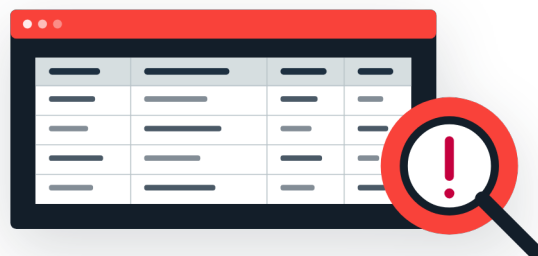
Vsaka vrstica → več trojčkov

- Vrstica = entiteta,
- stolpci = lastnosti,
- RDF preseže tabelo.

 [data.tsv ↗](#)

ob_id	ob_ime	tot_p
001	Ajdovščina	19895
213	Ankaran	3446
...	...	...

INTERAKTIVNA VAJA



[TSV v RDF-trojčke ↗](#)



RDF trojčki – polni zapis

 data.tsv ↗

ob_id	ob_ime	tot_p
001	Ajdovščina	19895
...	...	...



- **Subjekt**
 - **Povedek Predmet** .

```
<https://onto.mdp.gov.si/obcina/ajdovscina>
  <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <https://onto.mdp.gov.si/shema/Obcina> .

<https://onto.mdp.gov.si/obcina/ajdovscina>
  <https://onto.mdp.gov.si/shema/idObcinaSurs> "001" .

<https://onto.mdp.gov.si/obcina/ajdovscina>
  <https://onto.mdp.gov.si/shema/naziv> "Ajdovščina" .

<https://onto.mdp.gov.si/obcina/ajdovscina>
  <https://onto.mdp.gov.si/shema/steviloPrebivalcev> 19895 .
```



Predpone (*angl. prefix*)

- Skrajšanje URI-jev (`rdf:type` → `a`), bolj berljiv zapis, standardna praksa.

```
<https://onto.mdp.gov.si/obcina/ajdovscina>
  <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <https://onto.mdp.gov.si/shema/Obcina> .

<https://onto.mdp.gov.si/obcina/ajdovscina>
  <https://onto.mdp.gov.si/shema/idObcinaSurs> "001" .

<https://onto.mdp.gov.si/obcina/ajdovscina>
  <https://onto.mdp.gov.si/shema/naziv> "Ajdovščina" .

<https://onto.mdp.gov.si/obcina/ajdovscina>
  <https://onto.mdp.gov.si/shema/steviloPrebivalcev> 19895 .
```



```
@prefix obcina: <https://onto.mdp.gov.si/obcina/> .
@prefix shema: <https://onto.mdp.gov.si/shema/> .

obcina:ajdovscina a shema:Obcina .
obcina:ajdovscina shema:idObcinaSurs "001" .
obcina:ajdovscina shema:naziv "Ajdovščina" .
obcina:ajdovscina shema:steviloPrebivalcev 19895 .
```



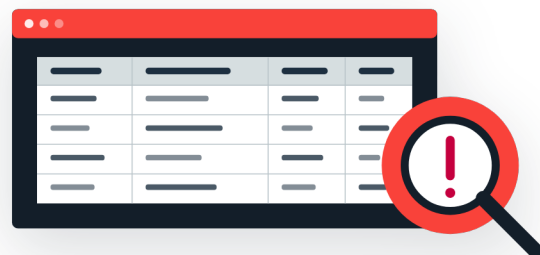
Krajši zapis (TTL sintaksa)

- Podpičje ; za isti subjekt,
- bolj jedrnato.

```
@prefix obcina: <https://onto.mdp.gov.si/obcina/> .  
@prefix shema:  <https://onto.mdp.gov.si/shema/> .  
  
obcina:ajdovscina a shema:Obcina .  
obcina:ajdovscina shema:idObcinaSurs "001" .  
obcina:ajdovscina shema:naziv "Aj dovščina" .  
obcina:ajdovscina shema:steviloPrebivalcev 19895 .
```



INTERAKTIVNA VAJA



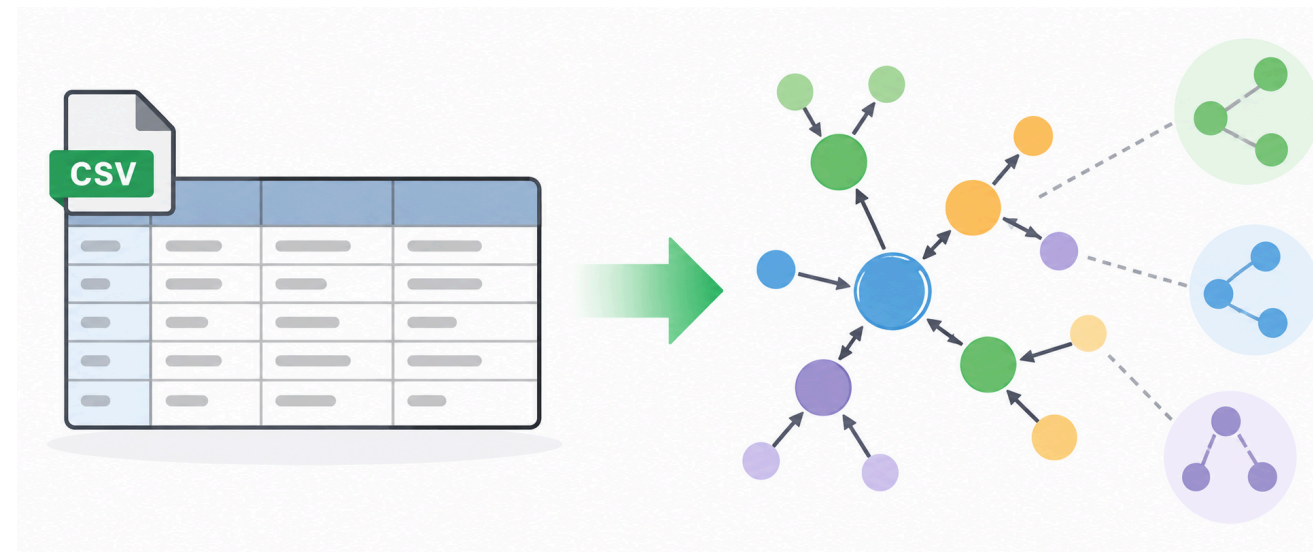
[Popravi Turtle ↗](#)

```
@prefix obcina:  
  <https://onto.mdp.gov.si/obcina/> .  
@prefix shema:  
  <https://onto.mdp.gov.si/shema/> .  
  
obcina:ajdovscina a shema:Obcina ;  
  shema:idObcinaSurs "001" ;  
  shema:naziv "Aj dovščina" ;  
  shema:steviloPrebivalcev 198
```



Od tabele do grafa

- **CSV** = 2D tabela,
- **RDF** = graf,
- priprava na povezovanje.





Kako določimo URI-je?

- Stabilni in trajni identifikatorji,
- berljivi in enolični,
- brez posebnih znakov,
- niso vezani izključno na interne ID-je.

✓ Dober, delujoč URI	✗ Slab URI	Zakaj?
https://dbpedia.org/resource/Ljubljana	https://primer.si/mesto/Ljubljana 2026	Brez presledkov in časovno spremenljivih dodatkov.
https://dbpedia.org/resource/Slovenia	https://primer.si/drzava?id=17	URI jasno identificira entiteto, ne le internega ID-ja.
https://dbpedia.org/resource/Berlin	https://primer.si/mesto/Berlin!	Brez nepotrebnih posebnih znakov.
https://dbpedia.org/resource/Semantic_Web	https://primer.si/pojem/Semantic Web	Stabilna in konsistentna struktura URI-ja.
https://dbpedia.org/resource/The_Beatles	https://interni-sistem.local/entity/84723	Javni in trajni identifikator, ne vezan na interni sistem.



Vzorec URI-jev na delavnici

- Entitete: `/obcina/{naziv}`,
- pojmi (shema): `/shema/{pojem}`,
- jasna ločitev pomena.



Kaj dobimo z ravnjo 4 ★?

- **Enolična identifikacija** entitet,
- **povezljivost** med različnimi viri,
- **grafovska struktura** podatkov,
- osnova za **poizvedovanje** in **sklepanje**,
- priprava na **ontologije**.





4. Formalizacija podatkovnega modela z ontologijami



Zakaj RDF sam po sebi ni dovolj?

- RDF zagotovi zapis in povezljivost,
- pomen pojmov ni formalno opredeljen,
- pravila so implicitna,
- računalnik/agent jih ne razume,
- potrebujemo **ontologijo**.



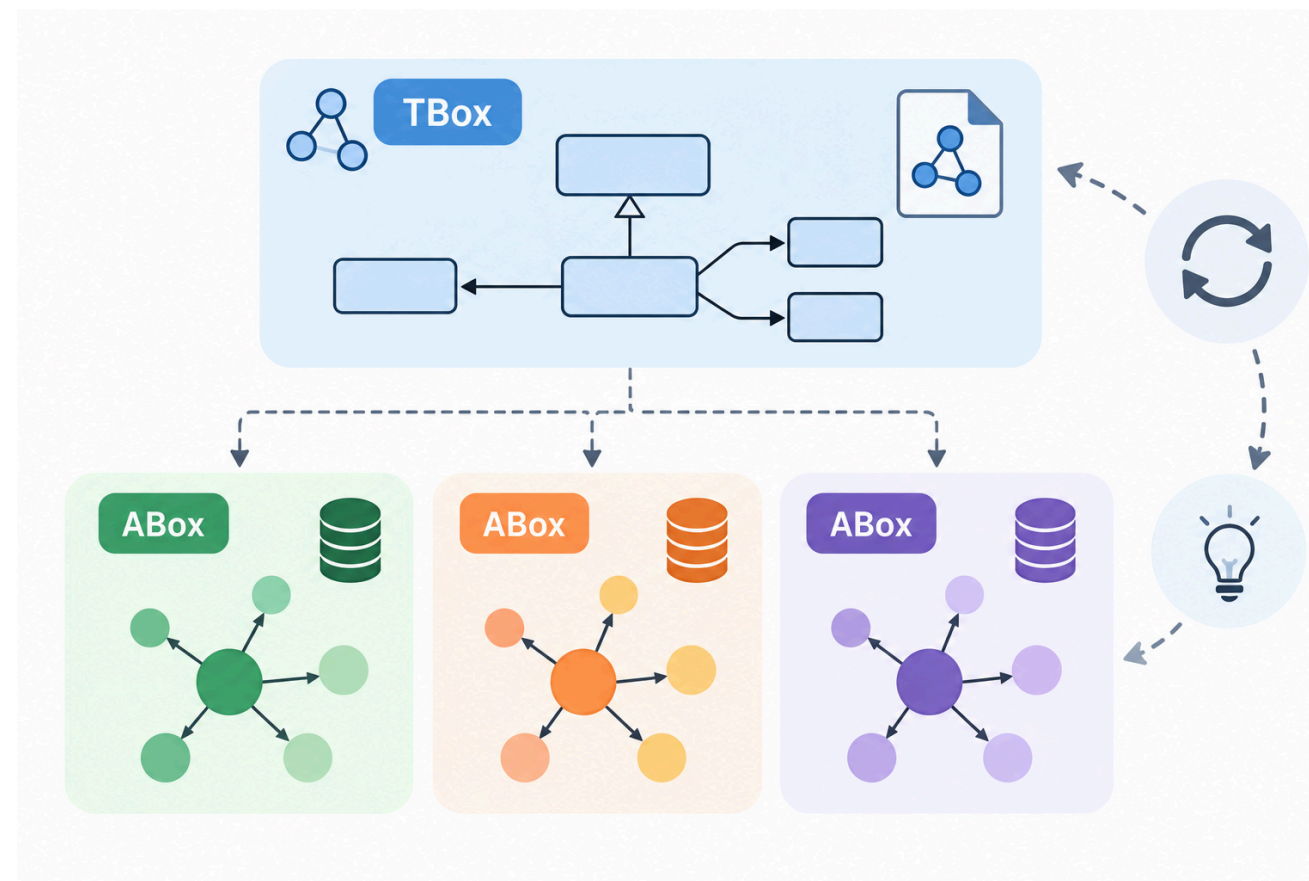
Kaj je ontologija (*v kontekstu delavnice*)?

- Formalni opis pomena,
- razredi, lastnosti in relacije,
- pravila in omejitve,
- **model pomenskega področja** (ne podatki).



Ločevanje modela in podatkov: TBox in ABox

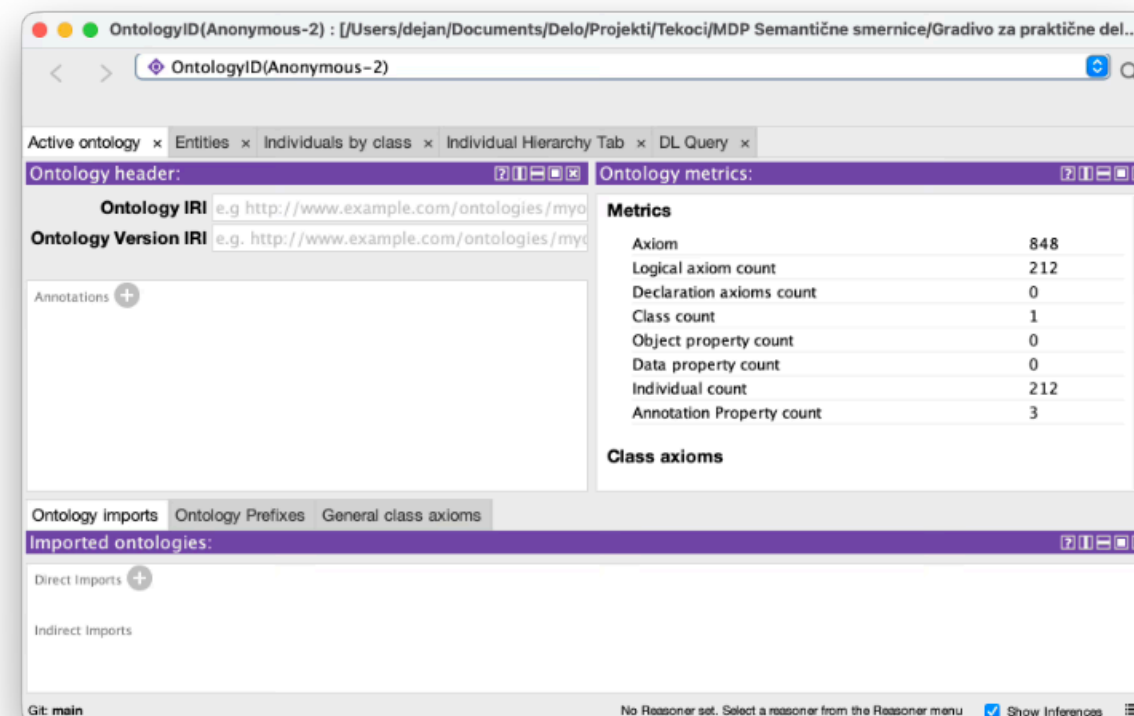
- **TBox** – model, struktura in pravila,
- **ABox** – konkretni podatki in primerki,
- isti model, različni podatki,
- lažje vzdrževanje in nadgradnje.





Uvoz podatkov TTL v Protégé

- Začnemo z **realnimi podatki SURS** ( **SURS_obcine.ttl** ↗)
 - 848 trditev (axioms),
 - 212 primerkov (občin),
 - 3 lastnosti (še brez formalnega pomena),
- RDF/TTL brez formalnega modela,
- fokus na podatkih (ABox).

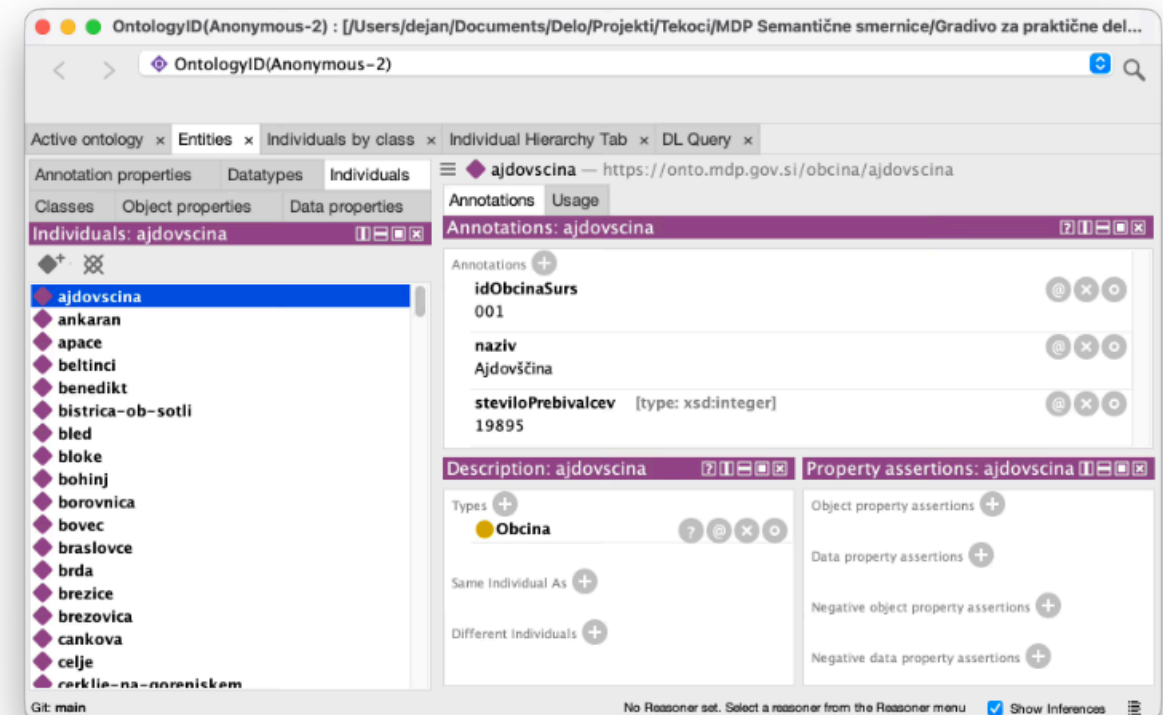


Orodje **Protégé** je odprtokodno in je na voljo na <https://protege.stanford.edu/>.



Pregled podatkov: Individuals

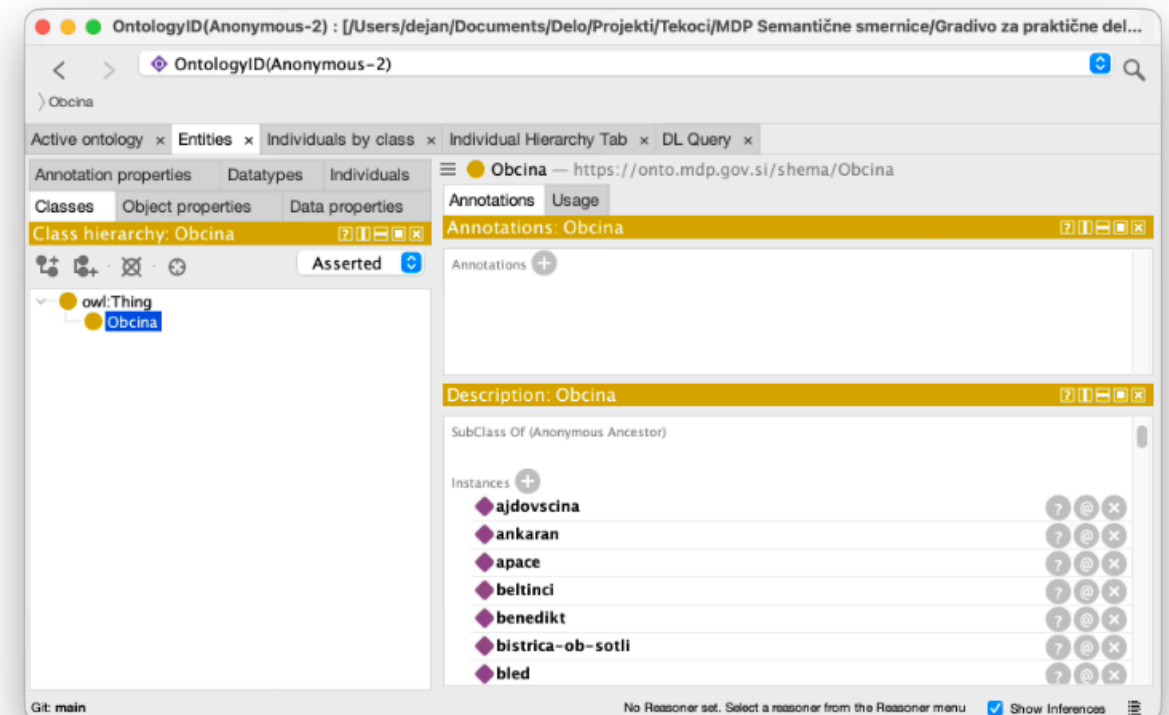
- Vsi primerki v zavihku **Individuals**,
- brez formalnega opisa modela.





Pregled podatkov: Classes

- En sam razred: **Obcina**,
- brez formalnega opisa modela.





Prvi korak formalizacije: opis razreda

- Iz podatkov izluščimo pojme,
- v središču: **Občina**,
- prehod iz implicitnega v formalni model.



Razred Občina

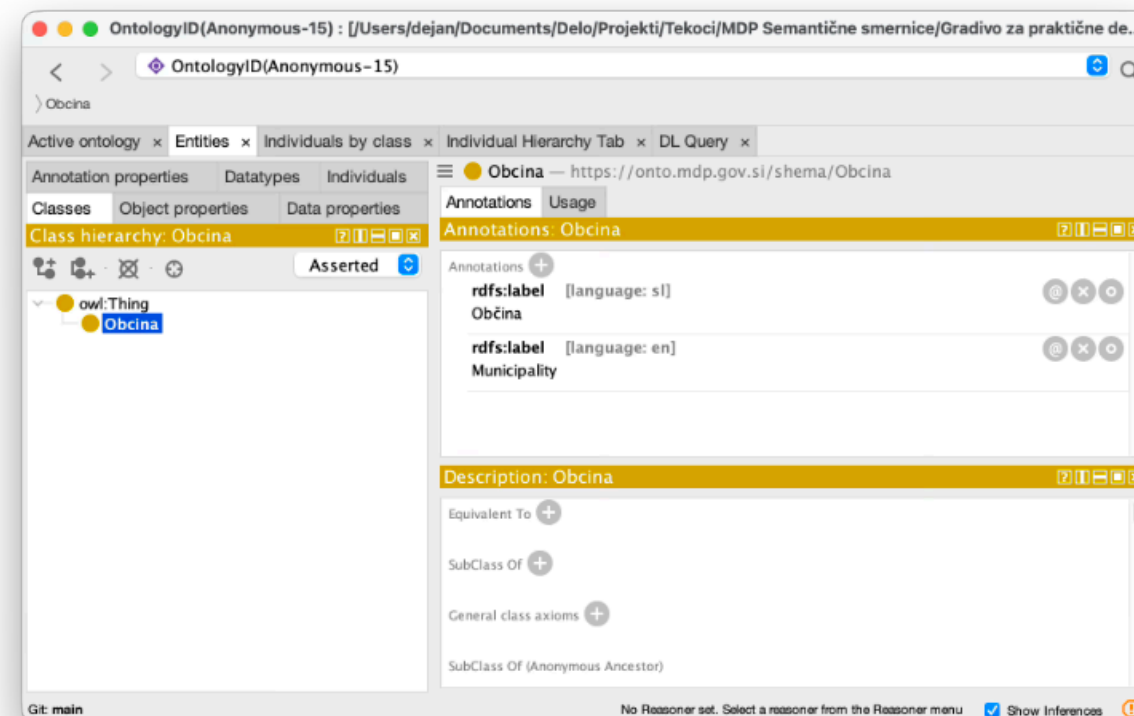
Implicitna opredelitev (🐢 SURS_obcine.ttl ↗)

```
obcina:ajdovscina a shema:Obcina .
```



Formalna opredelitev

```
shema:Obcina rdf:type owl:Class ;  
  rdfs:label "Občina"@sl ;  
  rdfs:label "Municipality"@en .
```





Dodajanje oznak (večjezičnost)

- `rdfs:label`,
- jezikovne oznake
 - ročno (`@sl`, `@en`),
 - GUI (`@es`).

```
schema:Obcina rdf:type owl:Class ;  
  rdfs:label "Municipality"@en , "Občina"@sl ,  
  "Municipio"@es .
```

The screenshot shows a window titled "Obcina" with a tabbed interface. The "Literal" tab is selected. On the left, a list of properties is shown, with "rdfs:label" highlighted. On the right, the "Value" field contains the text "municipio". Below the value field, the "Language Tag" is set to "es" and the "Datatype" is set to "xsd:string". At the bottom right, there are "Cancel" and "OK" buttons.



Kaj Protégé doda samodejno (1/2)?

- Lastnosti še niso formalno opredeljene,
- zato se interpretirajo kot **oznake**.

```
schema:idObcinaSurs rdf:type owl:AnnotationProperty .  
schema:naziv rdf:type owl:AnnotationProperty .  
schema:steviloPrebivalcev rdf:type owl:AnnotationProperty .
```



Kaj Protégé doda samodejno (2/2)?

Primerki kot `owl:NamedIndividual`

```
obcina:ajdovscina rdf:type owl:NamedIndividual ,  
                    shema:Obcina .
```

Osnovni imenski prostor in ontologija

```
@base <https://onto.mdp.gov.si/shema/> .  
<https://onto.mdp.gov.si/shema/> a owl:Ontology .
```



Podatkovne lastnosti

- Formalizacija obstoječih lastnosti,
- prehod iz opisov v **podatkovne lastnosti**,
- lastnosti pripadajo razredu **Občina**.

SURS · ( [data.tsv ↗](#))

ob_id	ob_ime	tot_p
001	Ajdovščina	19895
...	...	...



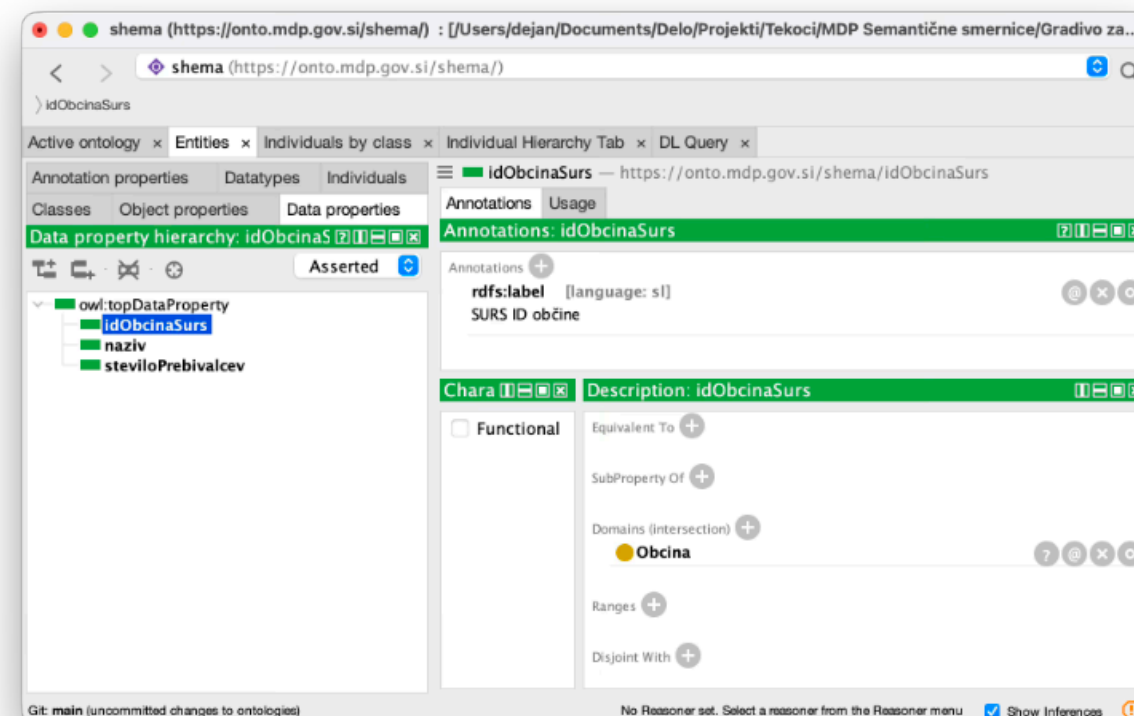
Dodajanje podatkovnih lastnosti v Protégé

- Entities → Data properties,
- lastnosti imajo **domeno** in **obseg** (tip podatka).

```
schema:idObcinaSurs a owl:DatatypeProperty ;  
  rdfs:domain schema:Obcina ;  
  rdfs:range xsd:string ;  
  rdfs:label "SURS ID občine"@sl .
```

```
schema:naziv a owl:DatatypeProperty ;  
  rdfs:domain schema:Obcina ;  
  rdfs:range xsd:string ;  
  rdfs:label "naziv"@sl .
```

```
schema:steviloPrebivalcev a  
  owl:DatatypeProperty ;  
  rdfs:domain schema:Obcina ;  
  rdfs:range xsd:integer ;  
  rdfs:label "število prebivalcev"@sl .
```



Končna verzija s podatkovnimi lastnostmi je na voljo v 🐢 [SURS_obcine_dp.ttl](#) ↗.



Zakaj potrebujemo objektne lastnosti?

- Število prebivalcev ni trajna lastnost občine,
- gre za **opazovanje v času**,
- ena občina → več meritev,
- graf omogoča naravno razširitev.



```
schema:steviloPrebivalcev a owl:DatatypeProperty ;  
  rdfs:domain schema:Obcina ;  
  rdfs:range xsd:integer ;  
  rdfs:label "število prebivalcev"@sl .
```

```
obcina:ajdovscina schema:steviloPrebivalcev 19895 .
```



Model z objektno lastnostjo

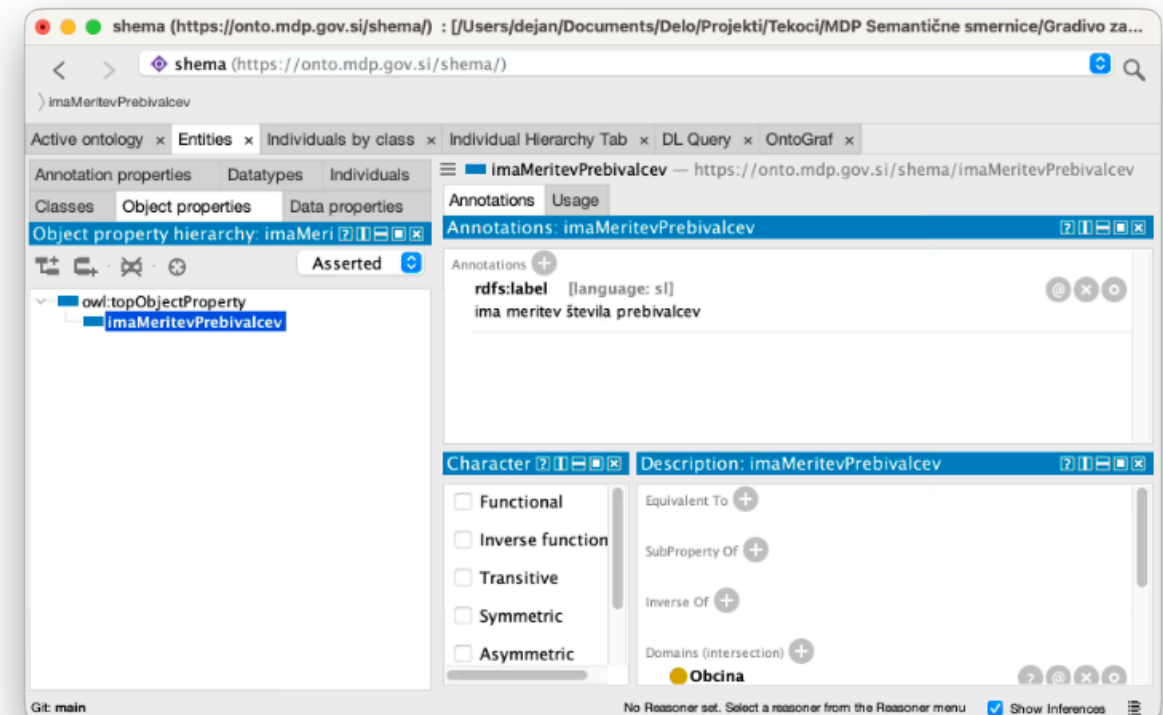
- Nov razred: **MeritevPrebivalcev**,
- lastnosti: **leto**, **vrednost**,
- objektna lastnost: **imaMeritevPrebivalcev**.

```
schema:MeritevPrebivalcev a owl:Class ;  
  rdfs:label "Meritev števila prebivalcev"@sl .
```

```
schema:leto a owl:DatatypeProperty ;  
  rdfs:domain schema:MeritevPrebivalcev ;  
  rdfs:range xsd:gYear ;  
  rdfs:label "leto"@sl .
```

```
schema:vrednost a owl:DatatypeProperty ;  
  rdfs:domain schema:MeritevPrebivalcev ;  
  rdfs:range xsd:integer ;  
  rdfs:label "vrednost"@sl .
```

```
schema:imaMeritevPrebivalcev a  
  owl:ObjectProperty ;  
  rdfs:domain schema:Obcina ;  
  rdfs:range schema:MeritevPrebivalcev ;  
  rdfs:label  
    "ima meritev števila prebivalcev"@sl .
```



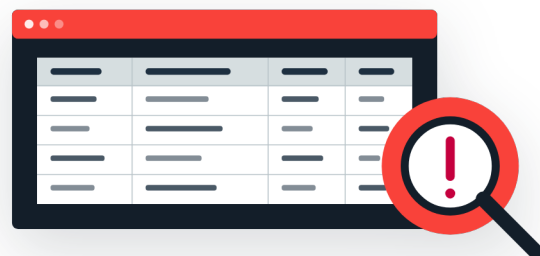


Preoblikovanje podatkov z novo strukturo

- Vrednosti → povezave na meritve,
- več let brez spremembe razreda **Obcina**,
- podatki postanejo časovno razširljivi.

Končna verzija s podatkovnimi lastnostmi je na voljo v 🐢 [SURS_obcine_dp_op.ttl](#) ↗.

INTERAKTIVNA VAJA



Formaliziraj občino ↗

```
obcina:ajdovscina shema:steviloPrebivalcev  
19895 .
```



```
obcina:ajdovscina a shema:Obcina ;  
  shema:imaMeritevPrebivalcev  
    obcina:ajdovscina-stevilo-prebivalcev-2024,  
    obcina:ajdovscina-stevilo-prebivalcev-2025 .
```

```
obcina:ajdovscina-stevilo-prebivalcev-2024  
  a shema:MeritevPrebivalcev ;  
  shema:leto "2024"^^xsd:gYear ;  
  shema:vrednost 19891 .
```

```
obcina:ajdovscina-stevilo-prebivalcev-2025  
  a shema:MeritevPrebivalcev ;  
  shema:leto "2025"^^xsd:gYear ;  
  shema:vrednost 19895 .
```

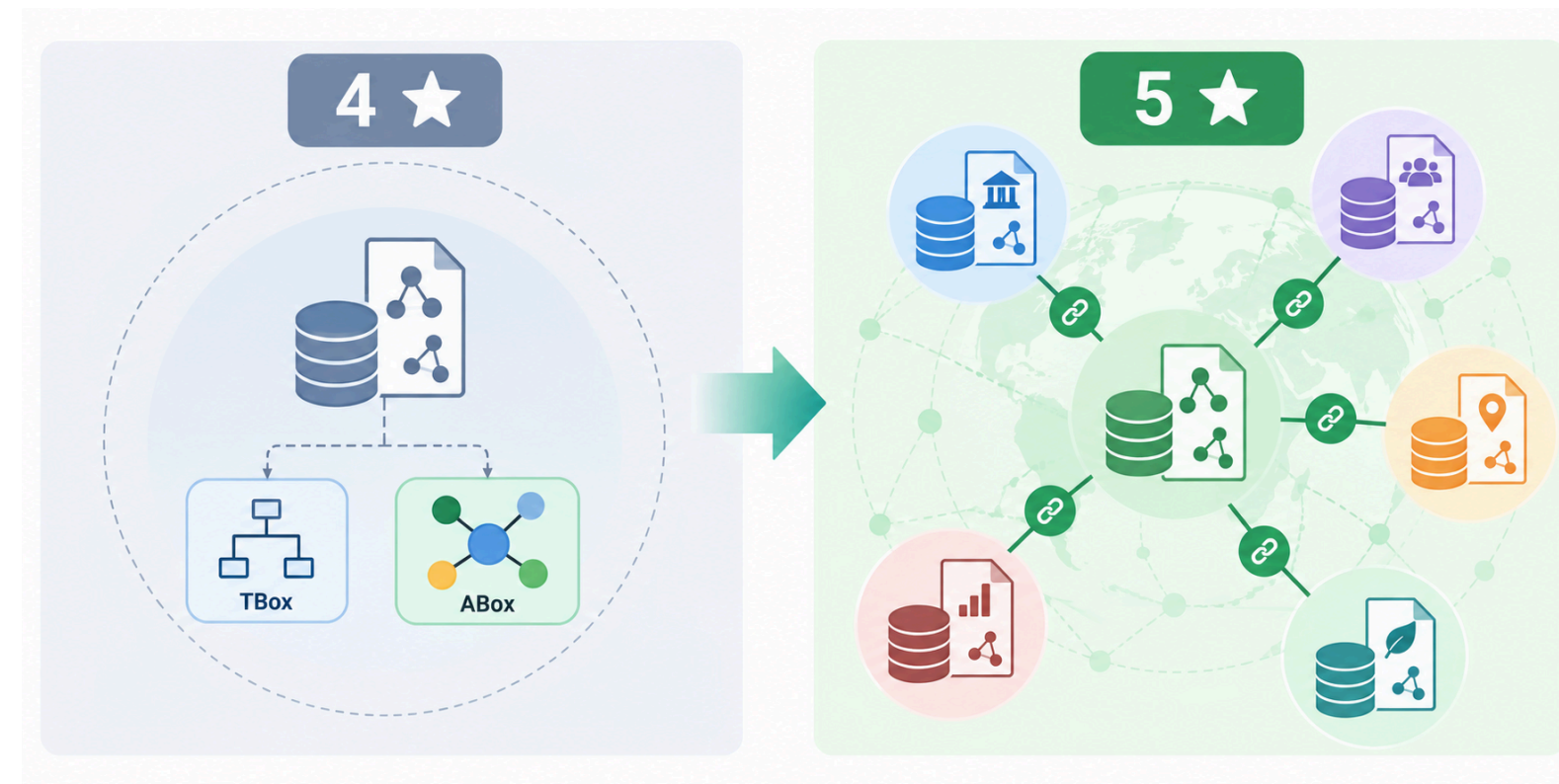


5. Raven 5 ★: povezani podatki in zunanji viri



Kaj pomeni raven 5 ★ v praksi?

- RDF + URI + ontologija (4 ★),
- podatki so še vedno izolirani,
- manjka povezava z drugimi viri,
- 5 ★ = **povezani podatki**.





Zunanji viri za povezovanje (5 ★) (1/2)

- **Podatkovni modeli (TBox)**

- Centralni besednjak – Platforma za semantično interoperabilnost  NIO ↗,

- **Splošne ontologije (TBox)**

- Schema.org –  <https://schema.org/>
- FOAF –  <http://xmlns.com/foaf/spec/>
- Dublin Core –  <http://purl.org/dc/elements/1.1/>
- SKOS –  <http://www.w3.org/2004/02/skos/core#>
- DCAT –  <http://www.w3.org/ns/dcat#>



Zunanji viri za povezovanje (5 ★) (2/2)

- **Splošni podatkovni viri (ABox)**
 - DBpedia –  <https://wiki.dbpedia.org/>
 - Wikidata –  <https://www.wikidata.org/>
 - LinkedGeoData –  <http://linkedgeodata.org/>
- **Domenski besednjaki**
 - EUROVOC –  <https://eurovoc.europa.eu/>
 - GEMET –  <https://www.eionet.europa.eu/gemet/>
 - INSPIRE –  <https://inspire.ec.europa.eu/glossary>
 - GeoNames –  <https://www.geonames.org/>

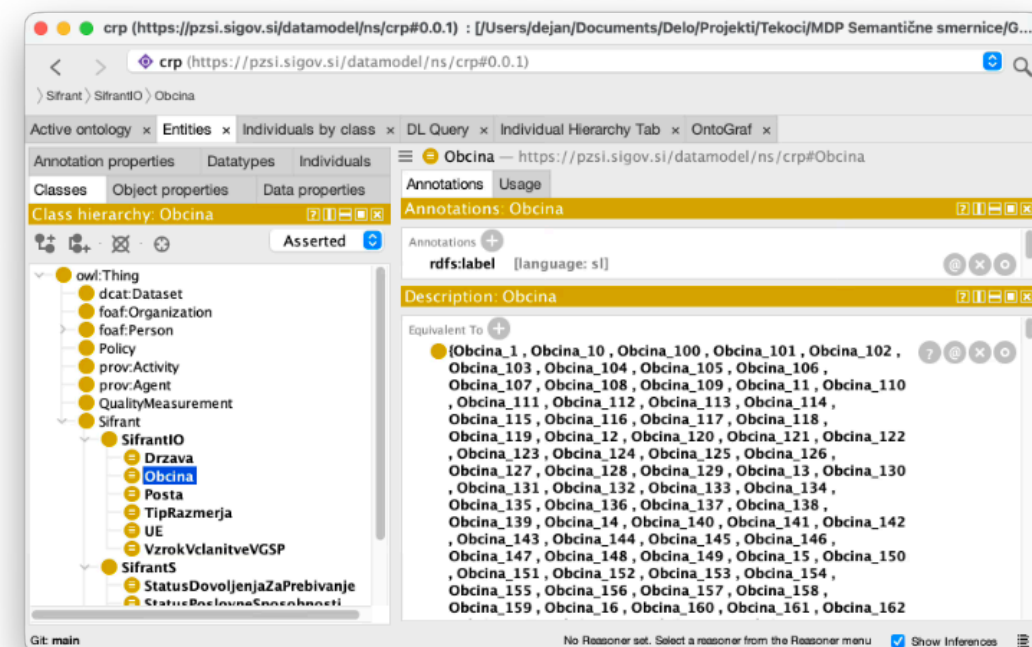


Povezovanje s hierarhijo pojmov (rdfs:subClassOf)

- Razredi so organizirani hierarhično,
- podrazredi dedujejo pomen nadrazredov,
- omogoča splošnejše poizvedbe,
- temelj za sklepanje.
- **cnbs:Sifrant**
 - **:SifrantIO**
 - **:Obcina**



CRP.ttl ↗



```
@prefix : <https://pzsi.sigov.si/datamodel/ns/crp#> .
@prefix cnbs: <http://onto.mju.gov.si/centralni-besednjak-sifranti#> .

cnbs:Sifrant rdfs:type owl:Class .
:SifrantIO rdfs:type owl:Class ;
            rdfs:subClassOf cnbs:Sifrant .
:Obcina rdfs:type owl:Class ;
         rdfs:subClassOf :SifrantIO .
```

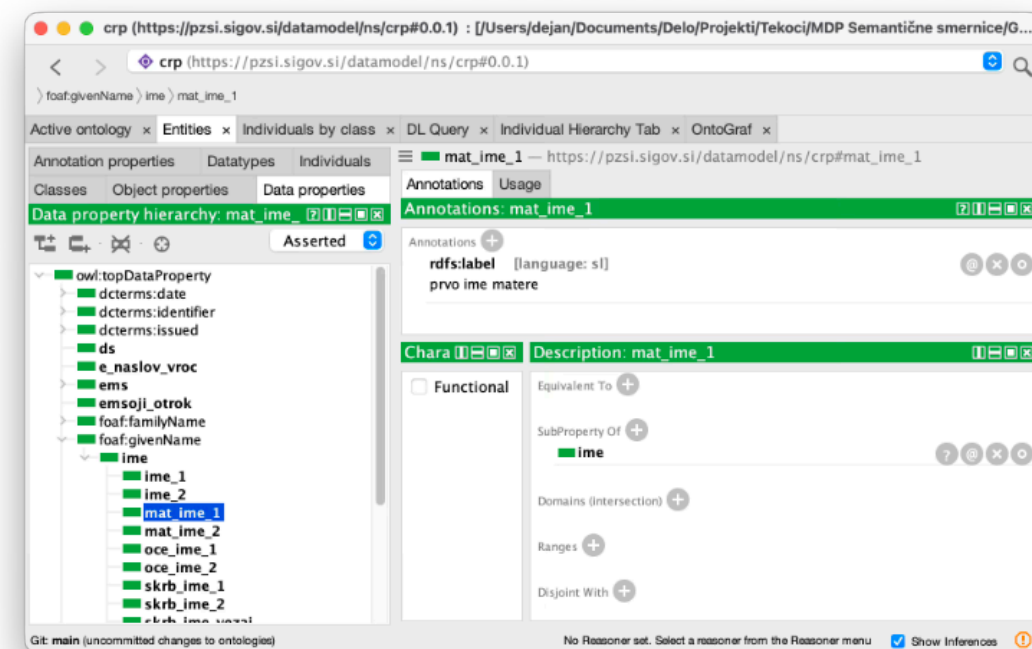


Hierarhija lastnosti (rdfs:subPropertyOf)

- Podobno kot pri razredih, tudi lastnosti tvorijo hierarhijo,
- splošna lastnost → bolj specifične podlastnosti,
- omogoča splošnejše poizvedbe in sklepanje.
- `foaf:givenName`
 - `:ime`
 - `:mat_ime_1`



CRP.ttl ↗



```
@prefix : <https://pzsi.sigov.si/datamodel/ns/crp#> .  
@prefix cnb: <http://onto.mju.gov.si/centralni-besednjak-core#> .  
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
```

```
:ime rdf:type owl:DatatypeProperty ;  
    rdfs:subPropertyOf foaf:givenName ;  
    rdfs:domain cnb:FizicnaOseba .  
:mat_ime_1 rdf:type owl:DatatypeProperty ;  
    rdfs:subPropertyOf :ime ;  
    rdfs:label "prvo ime matere"@sl .
```

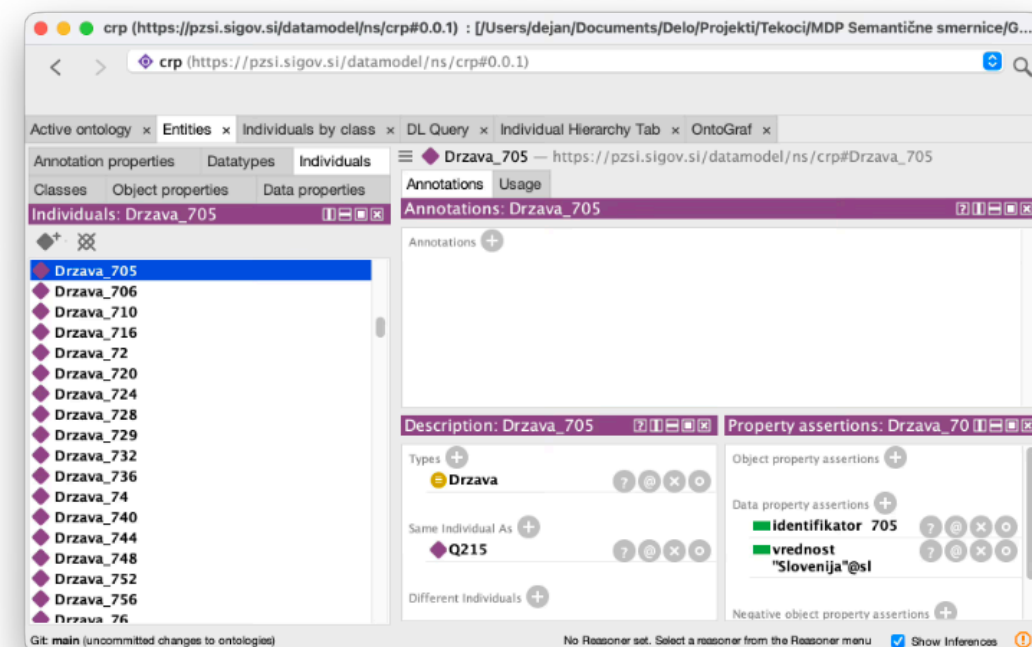


Povezovanje istih primerkov (owl:sameAs)

- Včasih vemo, da dva URI-ja opisujeta **isto entiteto**,
- entiteta je ista, razlikuje se le **vir podatkov**,
- **owl:sameAs** pove: *to je popolnoma isto*.

Primer:

- CRP: :Obcina_1 (🐢 [CRP.ttl ↗](#)),
- Wikidata: 🌐 [wd:Q331701 ↗](#) (Ajdovščina).



```
@prefix : <https://pzsi.sigov.si/datamodel/ns/crp#> .  
@prefix wd: <https://www.wikidata.org/entity/> .
```

```
:Obcina_1 rdf:type owl:NamedIndividual ,  
            :Obcina ;  
            :identifikator 1 ;  
            :vrednost "Ajdovščina"@sl ;  
            owl:sameAs wd:Q331701 .
```

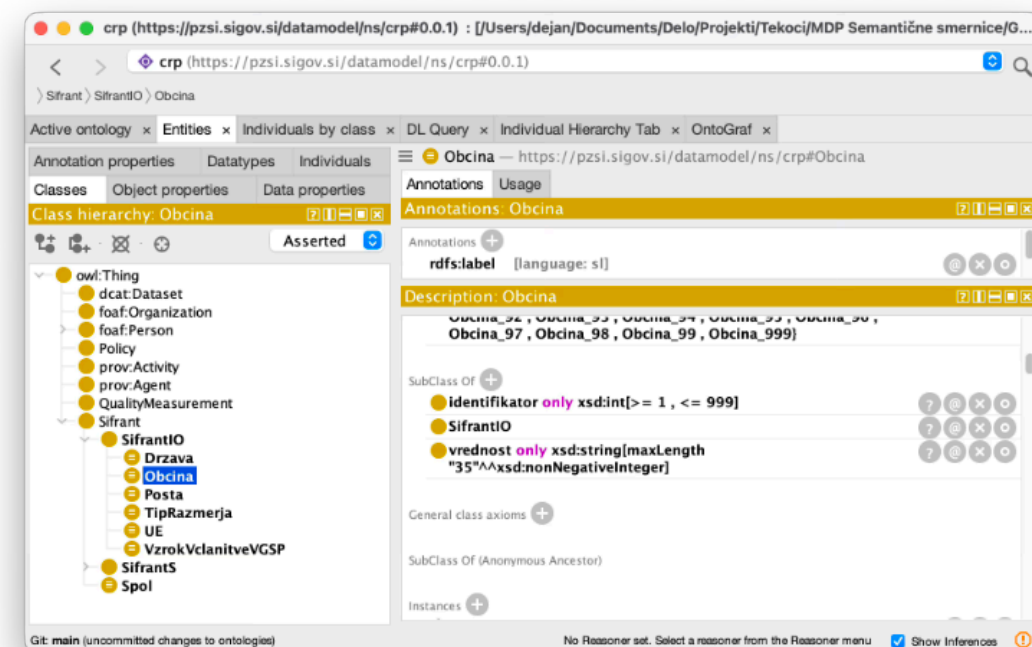


Povezovanje pojmov (owl:equivalentClass)

- Povezujemo **pojme / razrede**, ne primerkov,
- različni slovarji → isti pomen,
- **owl:equivalentClass** pomeni: *pojma sta semantično enakovredna*.

Primer:

- **:Obcina** (🐢 CRP.ttl ↗),
- **Municipality** v drugem slovarju / ontologiji,
- opredelitev vseh vrednosti.



```
:Obcina rdf:type owl:Class ;
  owl:equivalentClass [ rdf:type owl:Class ;
    owl:oneOf ( :Obcina_1
      :Obcina_10
      :Obcina_100
      ... ) ] ;

rdfs:subClassOf :SifrantIO ,
  [ rdf:type owl:Restriction ; ... ] ,
  [ rdf:type owl:Restriction ; ... ] .
```



Kako povezati CRP in SURS?

- 2 neodvisna vira:
 - **CRP** – uradni šifranti, stabilni identifikatorji (🐢 [CRP.ttl ↗](#)),
 - **SURS** – statistični kazalniki (npr. prebivalstvo) (🐢 [SURS_obcine_dp_op.ttl ↗](#)),
- opisujeta **iste občine**,
- cilj:
 - ne združujemo podatkov v eno datoteko,
 - omogočimo **skupno uporabo** v poizvedbah in sklepanju,
- integracija v ločeni datoteki 🐢 [Integracija_CRP_SURS.ttl ↗](#).





Povezovanje primerkov (CRP ↔ SURS)

- Najbolj neposreden pristop: **owl:sameAs**,
- primer: CRP: **crp:Obcina_1** (CRP) in **sursABox:ajdovscina** (SURS),
- pomensko gre za **isto občino**.



CRP.ttl ↗

```
@base
  <https://pzsi.sigov.si/datamodel/ns/crp#> .

:Obcina_1 rdf:type :Obcina ;
  :identifikator 1 ;
  :vrednost "Ajdovščina"@sl ;
  owl:sameAs wd:Q331701 .
```



SURS_obcine_dp_op.ttl ↗

```
@prefix obcina:
  <https://onto.mdp.gov.si/obcina/> .

obcina:ajdovscina a shema:Obcina ;
  shema:idObcinaSurs "001" ;
  shema:naziv "Ajdovščina" ;
  shema:imaMeritevPrebivalcev .
```



Integracija_CRP_SURS.ttl ↗

```
@prefix crp:      <https://pzsi.sigov.si/datamodel/ns/crp#> .
@prefix sursABox: <https://onto.mdp.gov.si/obcina/> .

crp:Obcina_1 owl:sameAs sursABox:ajdovscina .
```



Povezovanje pojmov in širši ekosistem

- Povezovanje **pojmov (TBox)**:
 - previdno: `rdfs:subClassOf`,
 - strogo: `owl:equivalentClass`,
- primer (previdnejša varianta) →

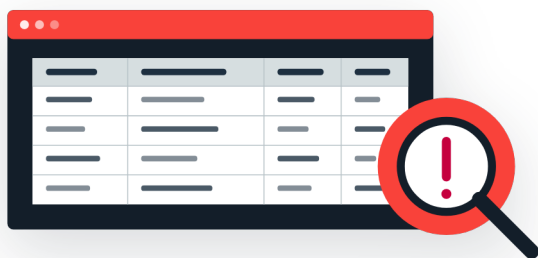


Integracija_CRP_SURS.ttl ↗

```
@prefix crp:
  <https://pzsi.sigov.si/datamodel/ns/crp#> .
@prefix surSTBox:
  <https://onto.mdp.gov.si/shema/> .

surSTBox:Obcina rdfs:subClassOf crp:Obcina .
```

INTERAKTIVNA VAJA



Poveži podatkovna vira ↗



Razširitev povezovanja – več občin, en ekosistem

- Ajdovščina je že povezana z **Wikidata**,
- dodamo še druge občine: **Celje**, Ljubljana, Maribor,
- CRP ↔ SURS ↔ Wikidata,
- podatkov **ne podvajamo**, ampak jih **logično povežemo**.



Integracija_CRP_SURS.ttl ↗

```
sursTBox:Obcina rdfs:subClassOf crp:Obcina .
```

```
crp:Obcina_11 owl:sameAs sursABox:celje .
```

```
crp:Obcina_11 owl:sameAs wd:Q3441823 .
```



6. Uporaba semantično opisanih podatkov s SPARQL

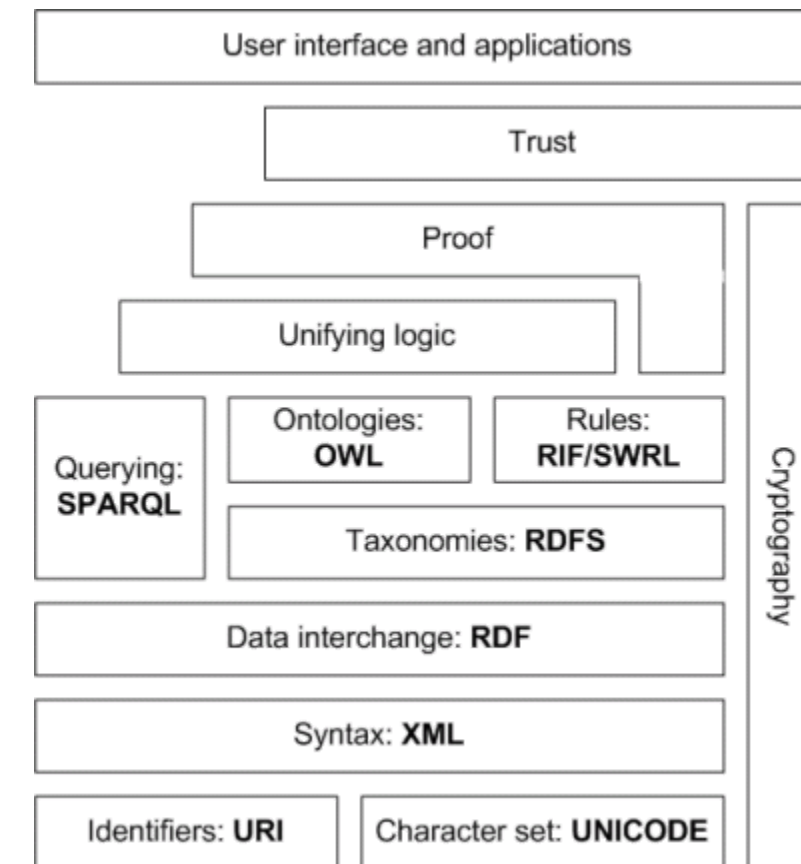


SPARQL je poizvedovalni sloj semantičnega spleta

- Podatkovna izmenjava: **RDF**
- Taksonomije: **RDFS**
- Ontologije: **OWL**
- Pravila: **RIF/SWRL**
- Poizvedovanje: **SPARQL**

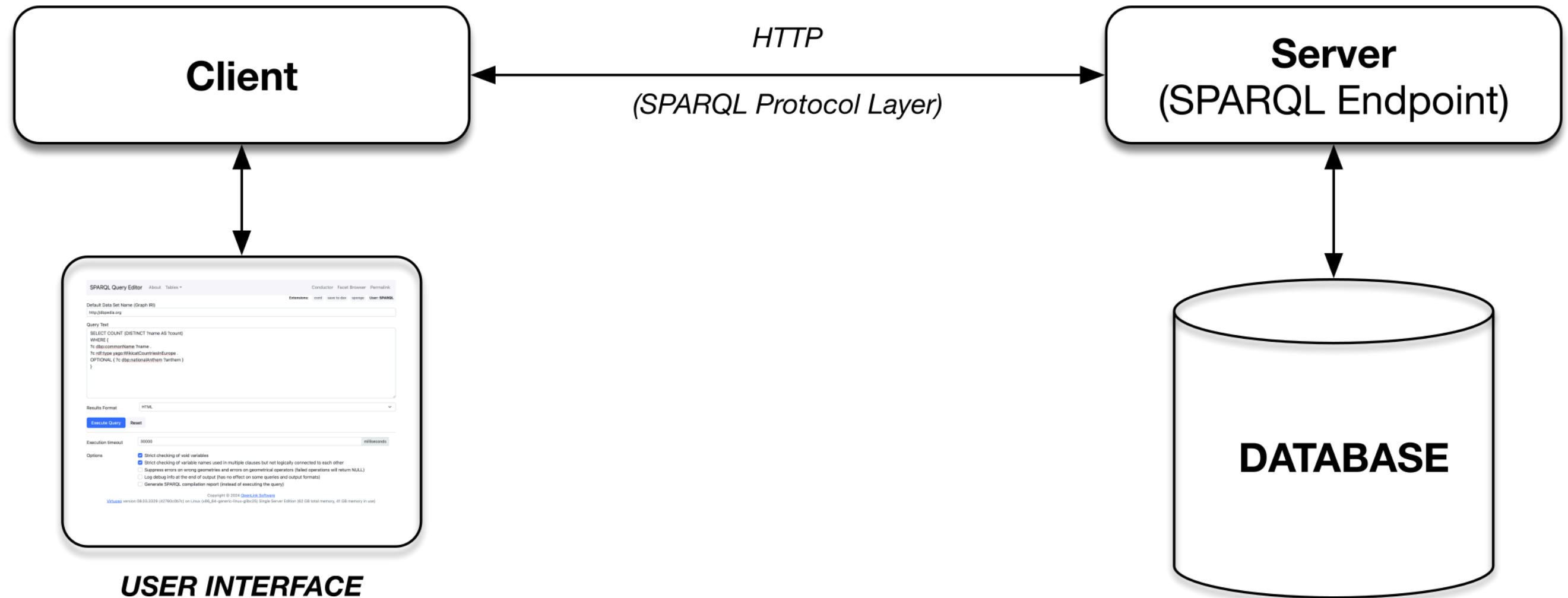
Različice standarda

- SPARQL 1.0 (2008)
- SPARQL 1.1 (2013)





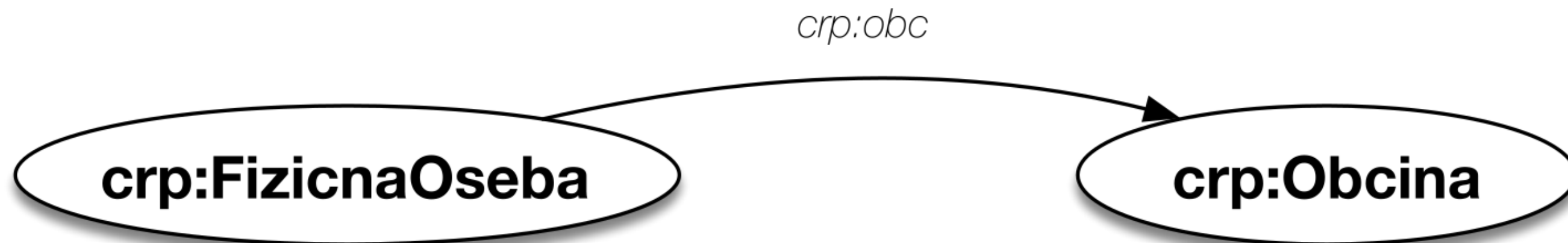
SPARQL poveže odjemalca s podatkovnim grafom





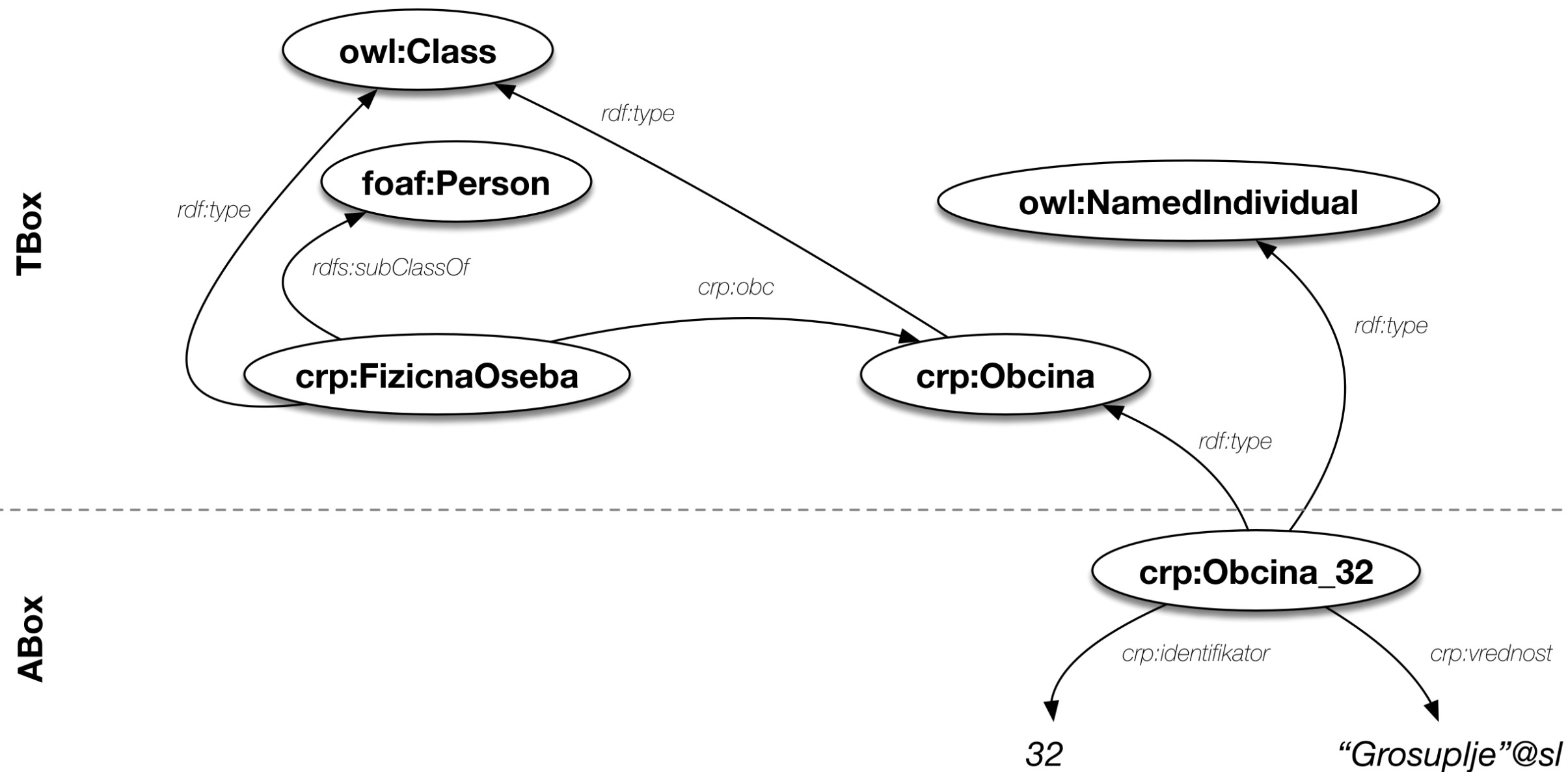
Predstavitev CRP kot graf

```
### https://pzsi.sigov.si/datamodel/ns/crp#obc
:obc rdf:type owl:ObjectProperty ;
      rdfs:domain cnb:FizicnaOseba ;
      rdfs:range :Obcina .
```





Širša predstavitev CRP v okolici specifičnega objekta





Poizvedovanje s SPARQL nad CRP

Okolje Protégé ponuja tudi možnost poizvedovanja s SPARQL (Window -> Tabs -> SPARQL Query).

V Protégé odprimo datoteko 🐢 **CRP.ttl** ↗ in izvedimo spodnjo poizvedbo:

```
SELECT ?osebek ?povedek ?predmet  
  
WHERE {  
    ?osebek ?povedek ?predmet .  
}
```



Kaj predstavlja rezultat poizvedbe?

The screenshot shows the Protégé SPARQL Query interface. The query is:

```
SELECT ?osebek ?povedek ?predmet  
  
WHERE {  
    ?osebek ?povedek ?predmet .  
}
```

The results are displayed in a table with three columns: osebek, povedek, and predmet. The first row shows the results for the query.

osebek	povedek	predmet
xsd:integer	rdf:type	rdfs:Datatype
8a1f49a4_5ede_460b_8283_a4d547bd5a9	hashCode	"-678389819"^^<http://www.w3.org/2001
8a1f49a4_5ede_460b_8283_a4d547bd5a9	sourceOntology	9d94dddc_cd92_4bcf_bfeb_a9afb065dafa
Drzava_50	vrednost	"Bangladeš"@sl
vrednost	rdf:type	owl:DatatypeProperty
Drzava_50	rdf:type	owl:NamedIndividual
rdf:PlainLiteral	rdf:type	rdfs:Datatype

At the bottom of the interface, there is an "Execute" button and a status bar indicating "No Reasoner set. Select a reasoner from the Reasoner menu" and "Show Inferences" is checked.



Osnovno iskanje vsebine v CRP

- Izrazi s **?** so **spremenljivke**,
- SPARQL poišče **vse podgrafe**, ki se ujemajo z vzorcem,
- poiščimo vse povezave, ki izhajajo iz **crp:Obcina_32**:

```
SELECT ?povedek ?predmet

WHERE {

  <https://pzsi.sigov.si/datamodel/ns/crp#Obcina_3
    ?povedek ?predmet .
}
```

- Pri daljših poizvedbah si tudi tu pomagamo s **predponami**:

```
PREFIX crp:
<https://pzsi.sigov.si/datamodel/ns/crp#>

SELECT ?povedek ?predmet

WHERE {
  crp:Obcina_32 ?povedek ?predmet .
}
```



Zakaj sta obe poizvedbi enakovredni?



Rezultat iskanja po občini crp:Obcina_32

```
SELECT ?povedek ?predmet

WHERE {

  <https://pzsi.sigov.si/datamodel/ns/crp#Obcina_3
    ?povedek ?predmet .
}
```

```
PREFIX crp:
<https://pzsi.sigov.si/datamodel/ns/crp#>

SELECT ?povedek ?predmet

WHERE {
  crp:Obcina_32 ?povedek ?predmet .
}
```

povedek
rdf:type
rdf:type
rdf:type
identifikator
rdf:type
rdf:type
vrednost
rdf:type

predmet
Obcina
owl:NamedIndividual
owl:NamedIndividual
"32"^^<http://www.w3.org/2001/XMLSchema#integer>
owl:NamedIndividual
owl:NamedIndividual
"Grosuplje"@sl
owl:NamedIndividual

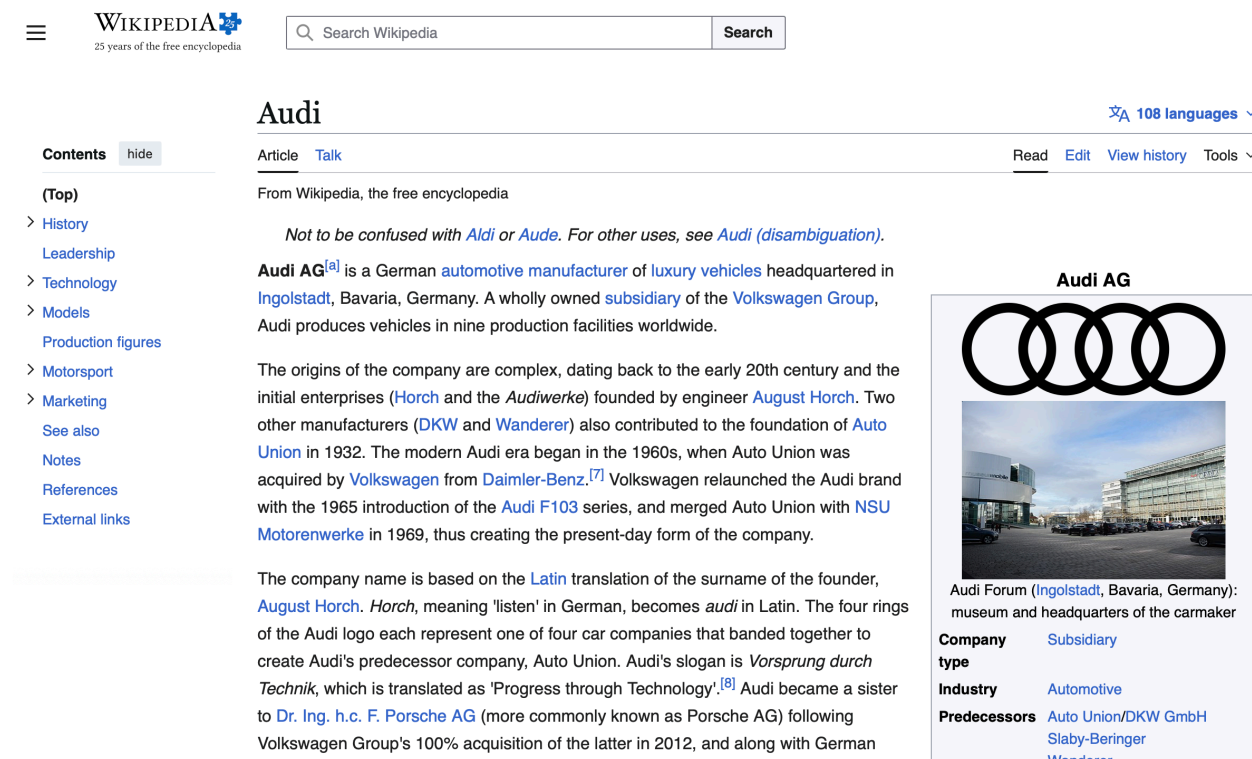


Od Wikipedije do DBpedie

- **Wikipedija** – prosta enciklopedija:
 - vsebina je namenjena **ljude**m,
 - strukturirani so predvsem podatki v okencih (*angl. infobox*).
- **DBpedia** – javna **semantična baza**:
 - vsebino črpa iz teh okenc,
 - vsaka stran dobi svoj **URI**,
 - podatke lahko izpišemo tudi v obliki **Turtle**.

Ista vsebina, dva pogleda:

-  Wikipedija: Audi ↗
-  DBpedia: Audi ↗

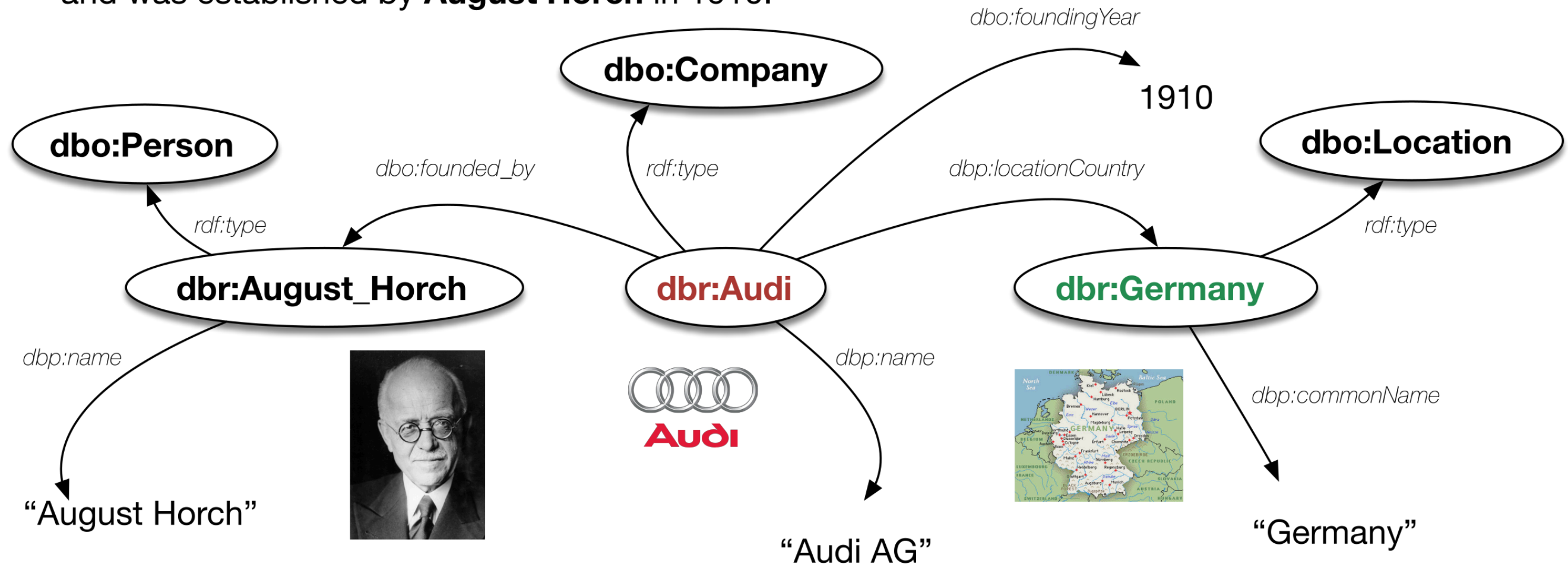


The screenshot shows the Wikipedia article for Audi. The top navigation bar includes the Wikipedia logo, a search bar, and a language selector set to "108 languages". The article title "Audi" is prominently displayed. Below the title, there are tabs for "Article" and "Talk", and a "Read Edit View history Tools" menu. The main text area contains a disambiguation note: "Not to be confused with *Aldi* or *Aude*. For other uses, see *Audi (disambiguation)*." The introductory paragraph states: "Audi AG^[a] is a German automotive manufacturer of luxury vehicles headquartered in Ingolstadt, Bavaria, Germany. A wholly owned subsidiary of the Volkswagen Group, Audi produces vehicles in nine production facilities worldwide." A sidebar on the left lists various sub-topics like History, Leadership, Technology, Models, Production figures, Motorsport, Marketing, See also, Notes, References, and External links. On the right, the Audi AG infobox is visible, featuring the Audi logo (four interlocking rings) and a photograph of the Audi Forum in Ingolstadt. Below the photo, it identifies the location as "Audi Forum (Ingolstadt, Bavaria, Germany): museum and headquarters of the carmaker". A table at the bottom of the infobox lists company details: Company type is "Subsidiary", Industry is "Automotive", and Predecessors are "Auto Union/DKW GmbH", "Slaby-Beringer", and "Wanderer".



Primer iz javnega grafa DBpedia

Audi is an automaker that makes luxury cars and SUVs. **The company** was born in **Germany** and was established by **August Horch** in 1910.





Javni endpoint omogoča poizvedovanje po DBpedii

SPARQL Query Editor About Tables ▾ Conductor Facet Browser Permalink

Extensions: cxml save to dav sponge User: SPARQL

Default Data Set Name (Graph IRI)

Query Text

```
SELECT COUNT (DISTINCT ?name AS ?count)
WHERE {
  ?c dbp:commonName ?name .
  ?c rdf:type yago:WikicatCountriesInEurope .
  OPTIONAL { ?c dbp:nationalAnthem ?anthem }
}
```

Results Format

Execution timeout milliseconds

Options

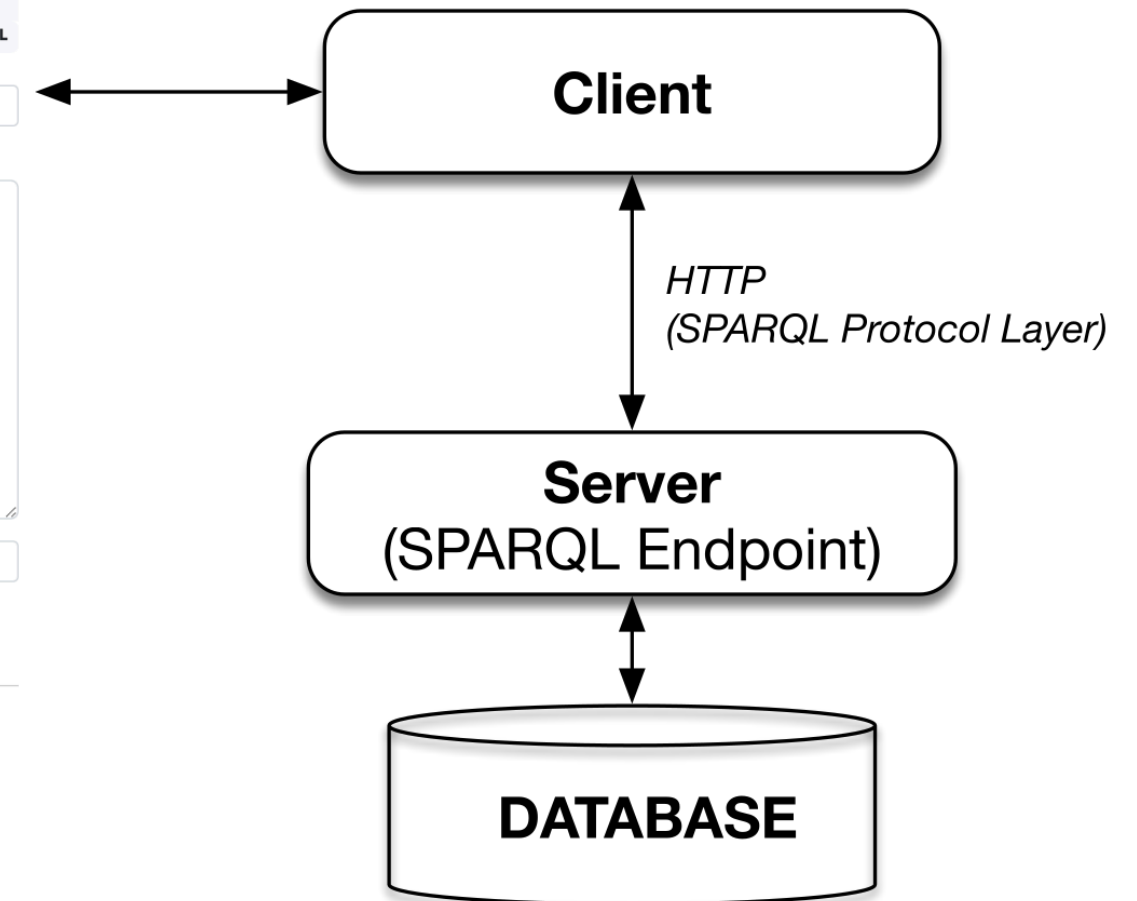
- ☒ Strict checking of void variables
- ☒ Strict checking of variable names used in multiple clauses but not logically connected to each other
- ☐ Suppress errors on wrong geometries and errors on geometrical operators (failed operations will return NULL)
- ☐ Log debug info at the end of output (has no effect on some queries and output formats)
- ☐ Generate SPARQL compilation report (instead of executing the query)

Copyright © 2024 [OpenLink Software](#)

[Virtuoso](#) version 08.03.3329 (42780c0b7c) on Linux (x86_64-generic-linux-glibc25) Single Server Edition (62 GB total memory, 41 GB memory in use)

USER INTERFACE

Odpri DBpedia SPARQL endpoint →





SPARQL poizveduje z ujemanjem vzorcev

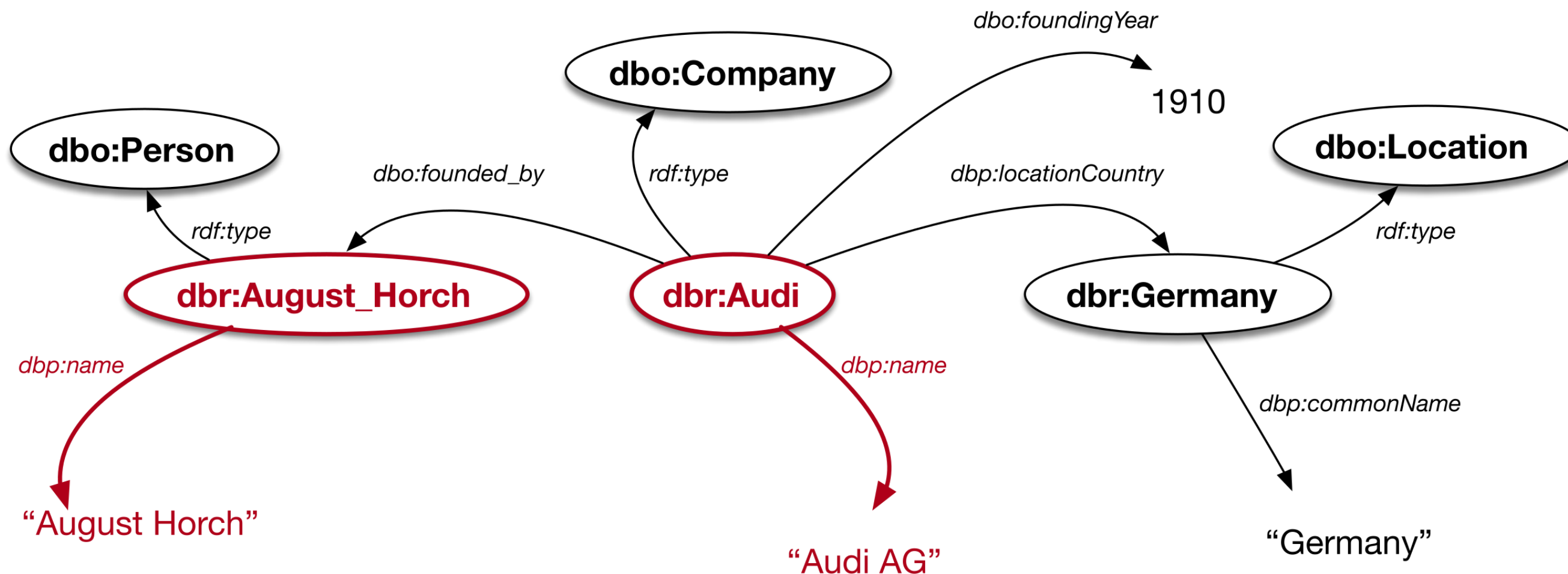
- Vzorec trojčka zapišemo podobno kot trojček v obliki Turtle.
- Neznane dele označimo s spremenljivkami.

```
?subject dbp:name ?object .
```

?subject	?object
dbr:August_Horch	"August Horch"
dbr:Audi	"Audi AG"



Vzorec se ujema neposredno z deli grafa

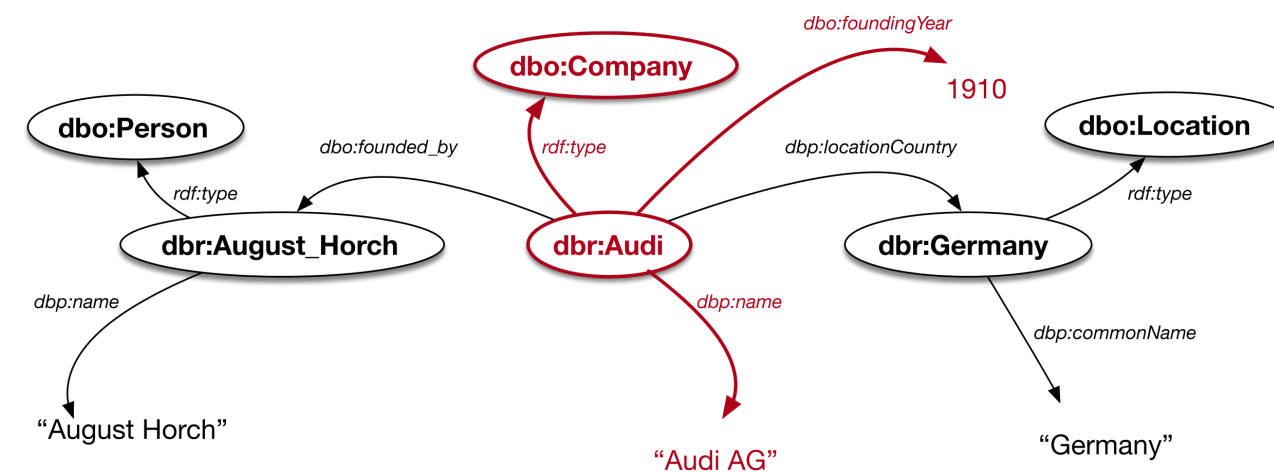




Konjunkcija poveže več vzorcev v en rezultat

```
?subject dbp:name ?name .  
?subject rdf:type dbo:Company .  
?subject dbo:foundingYear ?foundingYear .
```

?subject	?name	?foundingYear
dbr:Audi	"Audi AG"	1910





Poizvedba SPARQL ima štiri osnovne dele

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT ?name ?foundingYear

FROM <http://dbpedia.org/>

WHERE {
  ?subject dbp:name ?name .
  ?subject rdf:type dbo:Company .
  ?subject dbo:foundingYear ?foundingYear .
}
```

- **PREFIX** določi kratice imenskih prostorov.
- **SELECT** določi rezultatne spremenljivke.
- **FROM** izbere graf.
- **WHERE** opiše iskane vzorce.

Izvedite zgornjo poizvedbo na DBPediti.
Kakšne rezultate dobite? So pričakovani?



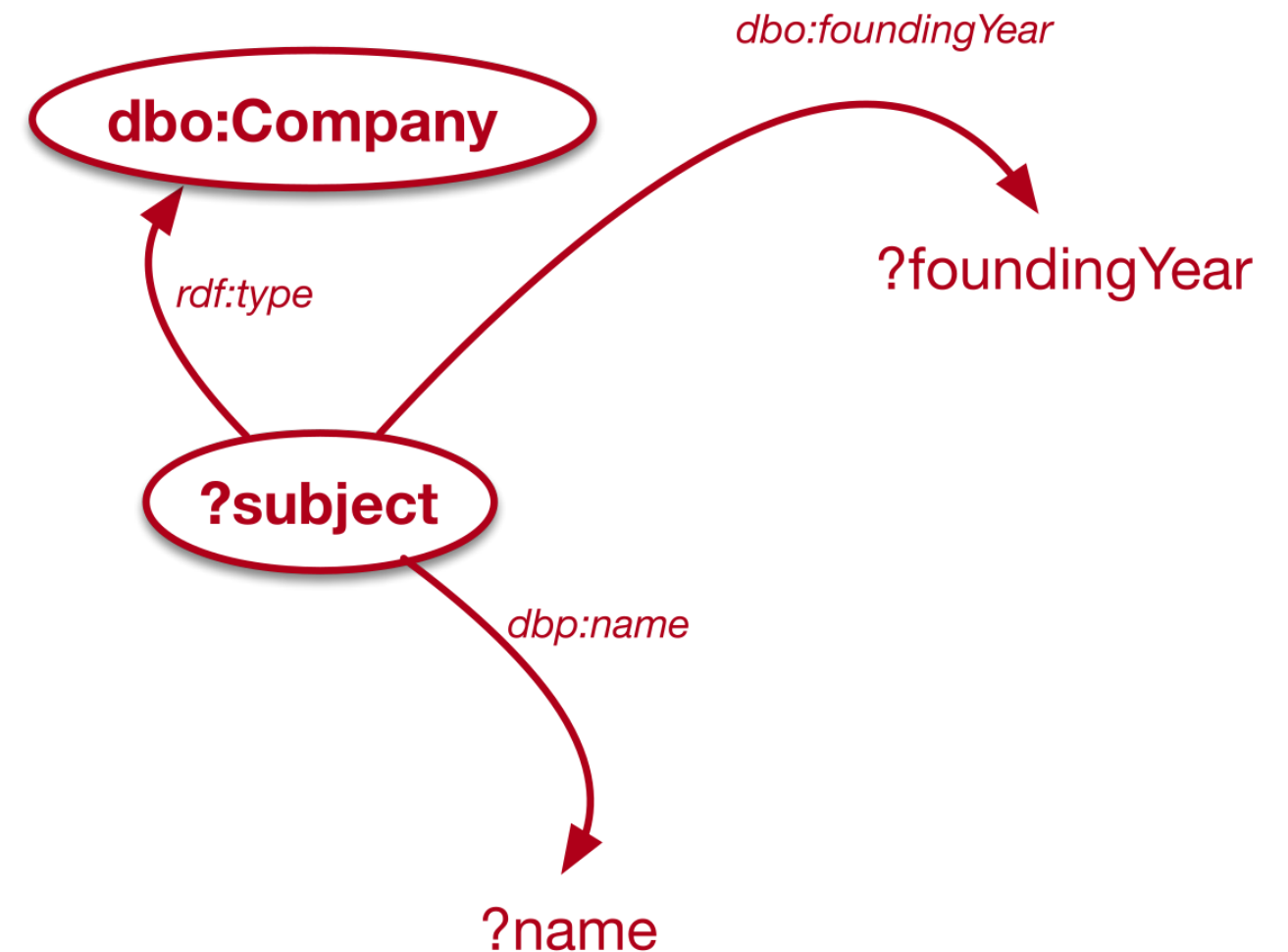
DBpedia ponuja javno storitev SPARQL

- Endpoint: <https://dbpedia.org/sparql>
- Približno **900 milijonov dejstev**
- Približno **400.000 organizacij**

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT ?name ?foundingYear

FROM <http://dbpedia.org/>

WHERE {
  ?subject dbp:name ?name .
  ?subject rdf:type dbo:Company .
  ?subject dbo:foundingYear ?foundingYear .
}
```





ORDER BY in LIMIT urejata rezultate

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT ?name ?foundingYear

FROM <http://dbpedia.org/>

WHERE {
    ?subject dbp:name ?name .
    ?subject rdf:type dbo:Company .
    ?subject dbo:foundingYear ?foundingYear .
}

ORDER BY ASC(?name)
LIMIT 100
```



OFFSET preskoči začetni del rezultatov

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT ?name ?foundingYear

FROM <http://dbpedia.org/>

WHERE {
    ?subject dbp:name ?name .
    ?subject rdf:type dbo:Company .
    ?subject dbo:foundingYear ?foundingYear .
}

ORDER BY ASC(?name)
LIMIT 100
OFFSET 14800
```



FILTER omeji rezultate na izbrane vire

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT ?name ?foundingYear

FROM <http://dbpedia.org/>

WHERE {
    ?subject dbp:name ?name .
    ?subject rdf:type dbo:Company .
    ?subject dbo:foundingYear ?foundingYear .

    FILTER (?subject IN (dbp:Audi))

}

LIMIT 100
```



FILTER REGEX filtrira besedilo

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT ?name ?foundingYear

FROM <http://dbpedia.org/>

WHERE {
    ?subject dbp:name ?name .
    ?subject rdf:type dbo:Company .
    ?subject dbo:foundingYear ?foundingYear .

    FILTER REGEX(?name, "^audi", "i")
}

LIMIT 100
```



Obvezni vzorec izloči rezultate brez trgovalnega imena

Opis razreda `dbo:Company` →

```
SELECT ?name ?foundingYear ?tradingName

FROM <http://dbpedia.org/>

WHERE {
  ?subject dbp:name ?name .
  ?subject rdf:type dbo:Company .
  ?subject dbo:foundingYear ?foundingYear .

  ?subject dbo:tradingName ?tradingName .
  FILTER REGEX(?name, "^audi", "i")

}

LIMIT 100
```



OPTIONAL ohrani rezultate brez trgovalnega imena

Opis razreda dbo:Company →

```
SELECT ?name ?foundingYear ?tradingName
FROM <http://dbpedia.org/>
WHERE {
  ?subject dbp:name ?name .
  ?subject rdf:type dbo:Company .

  ?subject dbo:foundingYear ?foundingYear .
  OPTIONAL { ?subject dbo:tradingName ?tradingName . }

  FILTER REGEX(?name, "^audi", "i")
}

LIMIT 100
```



Vzorce lahko sestavljamo

Primer: Audi v DBpedii →

```
SELECT ?carCompanyName  
  
FROM <http://dbpedia.org/>  
  
WHERE {  
    ?carCompany rdfs:label ?carCompanyName .  
    ?carCompany dbo:industry dbr:Automotive_industry .  
    ?carCompany dbp:locationCountry dbr:Germany .  
  
    FILTER (LANG(?carCompanyName) = "en")  
}
```



UNION združi alternativne vzorce

```
SELECT ?carCompanyName

FROM <http://dbpedia.org/>

WHERE {
  ?carCompany rdfs:label ?carCompanyName .
  ?carCompany dbo:industry dbr:Automotive_industry .

  { ?carCompany dbp:locationCountry dbr:Germany . }
  UNION
  { ?carCompany dbp:locationCountry dbr:Slovenia . }

  FILTER (LANG(?carCompanyName) = "en")
}
```



Avtomobilska podjetja po državah

```
SELECT ?carCompanyName ?countryName

WHERE {
  ?carCompany rdfs:label ?carCompanyName .
  ?carCompany dbo:industry dbr:Automotive_industry .
  ?carCompany dbp:locationCountry ?country .
  ?country rdfs:label ?countryName .

  FILTER (
    LANG(?carCompanyName) = "en" &&
    LANG(?countryName) = "en"
  )
}
```



COUNT izračuna agregat nad rezultati

```
SELECT (COUNT(DISTINCT ?carCompanyName) AS ?carCompanyNameCount)

WHERE {
  ?carCompany rdfs:label ?carCompanyName .
  ?carCompany dbo:industry dbr:Automotive_industry .
  ?carCompany dbp:locationCountry ?country .
  ?country rdfs:label ?countryName .

  FILTER (
    LANG(?carCompanyName) = "en" &&
    LANG(?countryName) = "en"
  )
}
```



GROUP BY izračuna rezultat za vsako državo

```
SELECT ?countryName
      (COUNT(DISTINCT ?carCompanyName) AS ?carCompanyNameCount)

WHERE {
  ?carCompany rdfs:label ?carCompanyName .
  ?carCompany dbo:industry dbr:Automotive_industry .
  ?carCompany dbp:locationCountry ?country .
  ?country rdfs:label ?countryName .

  FILTER (LANG(?carCompanyName) = "en" && LANG(?countryName) = "en")
}

GROUP BY ?countryName
ORDER BY DESC(?carCompanyNameCount)
```



GROUP_CONCAT združi imena znotraj skupine

```
SELECT ?countryName
      (COUNT(DISTINCT ?carCompanyName) AS ?carCompanyNameCount)
      (GROUP_CONCAT(?carCompanyName) AS ?carCompanyNames)

WHERE {
  ?carCompany rdfs:label ?carCompanyName .
  ?carCompany dbo:industry dbr:Automotive_industry .
  ?carCompany dbp:locationCountry ?country .
  ?country rdfs:label ?countryName .

  FILTER (LANG(?carCompanyName) = "en" && LANG(?countryName) = "en")
}

GROUP BY ?countryName
ORDER BY DESC(?carCompanyNameCount)
```



HAVING filtrira izračunane skupine

```
SELECT ?countryName
      (COUNT(DISTINCT ?carCompanyName) AS ?carCompanyNameCount)
      (GROUP_CONCAT(?carCompanyName) AS ?carCompanyNames)

WHERE {
  ?carCompany rdfs:label ?carCompanyName .
  ?carCompany dbo:industry dbr:Automotive_industry .
  ?carCompany dbp:locationCountry ?country .
  ?country rdfs:label ?countryName .

  FILTER (LANG(?carCompanyName) = "en" && LANG(?countryName) = "en")
}

GROUP BY ?countryName
HAVING (COUNT(?carCompanyName) > 5)
ORDER BY DESC(?carCompanyNameCount)
```



SPARQL 1.1 podpira tudi druge vrste operacij

- Oblike poizvedb:
 - SELECT
 - CONSTRUCT
 - ASK
 - DESCRIBE
- Posodabljanje podatkov: **INSERT DATA**
- Zahtevnejši vzorci
 - gnezdene poizvedbe
 - **BIND** za izračun novih vrednosti
 - poti lastnosti (angl. property paths) za sprehode po grafu
 - **SERVICE** za federirane poizvedbe
- Preverjanje podatkov s **SHACL**



SPARQL lahko uporabljamo tudi iz Pythona

- Poizvedbe izvajamo nad oddaljenimi storitvami SPARQL.
- Rezultate lahko obdelamo v običajnem podatkovnem postopku Python.

Odpri zvezek [SPARQL_in_Python.ipynb](#) →

Če nimate nameščenega Pythona, lahko zvezek poženete v storitvi [Google Colab](#) → .



Semantične podatke hranimo v namenskih bazah

- Primeri baz RDF s podporo SPARQL:
 - **GraphDB** (*uporabimo na delavnici*),
 - Apache Jena Fuseki,
 - Amazon Neptune,
 - Oracle Graph Database.
- Kaj pridobimo v primerjavi s Protégé?
 - **trajno hrambo** večjih grafov,
 - **storitev SPARQL** za odjemalce,
 - **sklepanje** ob uvozu podatkov,
 - **federirane poizvedbe** preko več virov.

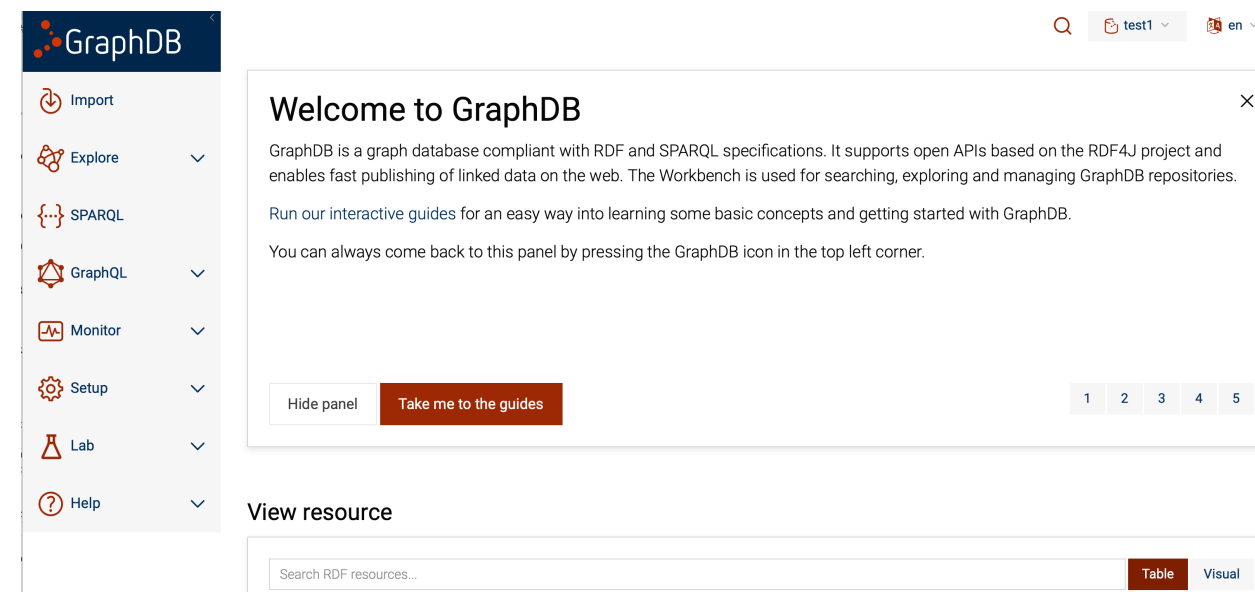


Zagon baze GraphDB v okolju Docker

```
docker run -p 127.0.0.1:7200:7200 \
  --name graphdb-instance-name-11 \
  -t ontotext/graphdb:11.3.1
```

- Vmesnik:  <http://localhost:7200>
- Ponovni zagon istega vsebnika:

```
docker start graphdb-instance-name-11
```



Za uporabo brezplačne verzije potrebujete licenco, ki jo lahko pridobite na spletni strani <https://www.ontotext.com/products/graphdb>.



Nov repozitorij in nastavitve sklepanja

- Ustvarimo nov **GraphDB Repository**,
- v razdelku *Inference and Validation*:
 - **odznačimo** *Disable owl:sameAs*,
 - ohranimo *RDFS-Plus (Optimized)*.



Zakaj je prav ta nastavev pomembna za nadaljevanje?

Inference and Validation

Ruleset



RDFS-Plus (Optimized)



Custom ruleset...

☐ Disable owl:sameAs

☐ Enable consistency checks

☐ Enable SHACL validation > SHACL options



Uvoz podatkov CRP in SURS

- V meniju *Import* izberemo repozitorij in naložimo:
 - 🐢 CRP.ttl ↗,
 - 🐢 SURS_obcine_dp_op.ttl ↗.
- Uvoženih **10.575 trojčkov** (*Explicit*),
- dodatno izpeljanih **7.507 trojčkov** (*Inferred*).

Import ⓘ

User data Server files ⓘ

Upload RDF files
All RDF formats, up to 1.0 GB

Get RDF data from a URL
All RDF formats

Import
Type of

Explore other file import options - Server files import and the GraphDB REST API.

Repository Test

Location: Local
Type: Graphdb
Access: Read/Write
Total statements: 18,082
Explicit: 10,575
Inferred: 7,507
Expansion ratio (total/explicit): 2

Type to filter by name

Name ↕	Size ↕	Uploaded ↕	Imported ↕	Context ↕
☐ SURS_obcine_dp_op.ttl Imported successfully in less than a second. Added 2358 statements	102.79 kb	2026-02-11, 22:40	2026-02-11, 22:40	ⓘ × 🗑️ ⏮️
☐ CRP.ttl Imported successfully in less than a second. Added 8217 statements	368.27 kb	2026-02-11, 22:39	2026-02-11, 22:39	ⓘ × 🗑️ ⏮️



Ista poizvedba, druga baza

```
PREFIX crp:
<https://pzsi.sigov.si/datamodel/ns/crp#>

SELECT ?povedek ?predmet

WHERE {
    crp:0bcina_32 ?povedek ?predmet .
}
```

- Poizvedba je **nespremenjena**,
- spremenilo se je le **okolje izvajanja**,
- podatki so zdaj v **bazi**, ne v urejevalniku.



Kaj to pomeni za ponovno uporabo poizvedb?



Federirane poizvedbe povežejo več grafov

- Ukaz **SERVICE** izvede del poizvedbe nad **oddaljeno storitvijo** SPARQL,
- podatkov **ni treba prenesti** v lastno bazo,
- rezultati se **združijo v en odgovor**.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX dbp: <http://dbpedia.org/property/>

SELECT ?subject ?name ?foundingYear

WHERE {
  SERVICE <https://dbpedia.org/sparql> {
    ?subject dbp:name ?name .
    ?subject rdf:type dbo:Company .
    ?subject dbo:foundingYear ?foundingYear .
    FILTER (?subject IN (dbr:Audi))
  }
}
```



Kdo je župan občine Grosuplje?

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX crp:
<https://pzsi.sigov.si/datamodel/ns/crp#>
PREFIX dbp: <http://dbpedia.org/property/>

SELECT ?idObcine ?zupan

WHERE {
    SERVICE <https://dbpedia.org/sparql> {
        ?subject dbp:officialName "Grosuplje"@en
    }
    ?subject dbp:leaderName ?zupan .
}
?obcina rdf:type crp:Obcina .
?obcina crp:identifikator ?idObcine .
?obcina crp:vrednost "Grosuplje"@sl .
}
```

- V CRP imamo **identifikator in ime** občine,
- podatka o županu **nimamo**,
- DBpedia ga ima:  **Grosuplje** ↗.

?idObcine	?zupan
32	"Peter Verlič"



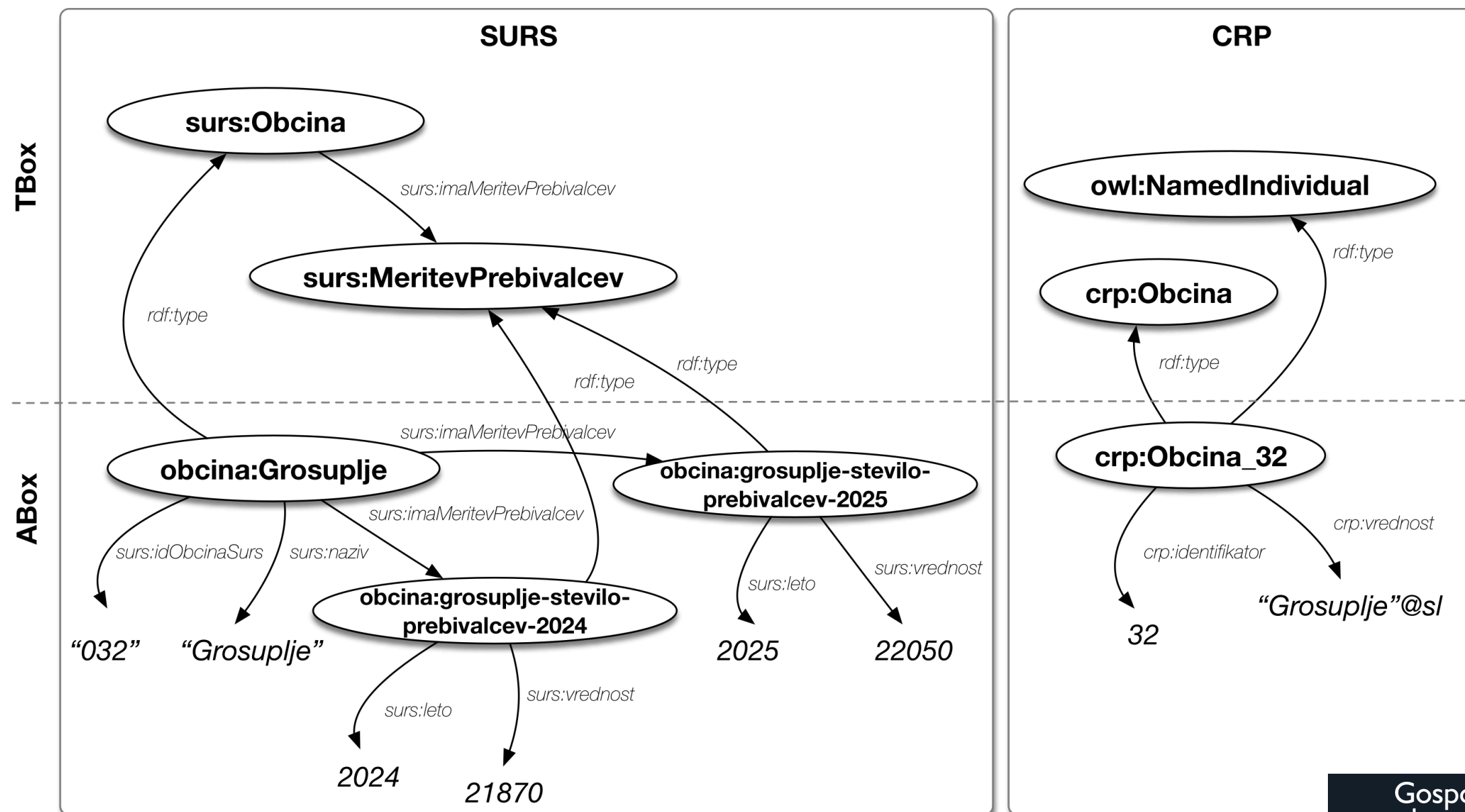
7. Sklepanje: prehod od podatkov k znanju

Uporaba več podatkovnih virov in materializacija – SURS in CRP



Sklepanje poveže podatke SURS in CRP

Vira CRP in SURS nista povezana in primer podatkov izgleda sledeče:





Vsak vir zase odgovori na svoj del vprašanja



CRP.ttl ↗

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX crp:
<https://pzsi.sigov.si/datamodel/ns/crp#>

SELECT ?imeObcineCRP ?idCRP
WHERE {
  ?obcinaCRP crp:vrednost ?imeObcineCRP .
  ?obcinaCRP crp:identifikator ?idCRP .
  ?obcinaCRP rdf:type crp:Obcina .
  FILTER (?imeObcineCRP = "Grosuplje"@sl)
}
```



SURS_obcine_dp_op.ttl ↗

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX surs: <https://onto.mdp.gov.si/shema/>

SELECT ?imeObcineSURS ?idSURS
        ?letoMeritve ?steviloPrebivalcev
WHERE {
  ?obcinaSURS surs:naziv ?imeObcineSURS .
  ?obcinaSURS surs:idObcinaSurs ?idSURS .
  ?obcinaSURS rdf:type surs:Obcina .
  ?obcinaSURS surs:imaMeritvePrebivalcev ?
  meritev .
  ?meritev surs:leto ?letoMeritve .
  ?meritev surs:vrednost ?steviloPrebivalcev .
  FILTER (?imeObcineSURS = "Grosuplje")
}
```





Skupna poizvedba – rezultat je videti pravilen

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX crp:
<https://pzsi.sigov.si/datamodel/ns/crp#>
PREFIX surs: <https://onto.mdp.gov.si/shema/>

SELECT ?imeObcineCRP ?idCRP ?imeObcineSURS
       ?idSURS ?letoMeritve ?steviloPrebivalcev
WHERE {
  ?obcinaCRP crp:vrednost ?imeObcineCRP .
  ?obcinaCRP crp:identifikator ?idCRP .
  ?obcinaCRP rdf:type crp:Obcina .
  FILTER (?imeObcineCRP = "Grosuplje"@sl)

  ?obcinaSURS surs:naziv ?imeObcineSURS .
  ?obcinaSURS surs:idObcinaSurs ?idSURS .
  ?obcinaSURS rdf:type surs:Obcina .
  ?obcinaSURS surs:imaMeritevPrebivalcev ?
  meritev .
  ?meritev surs:leto ?letoMeritve .
  ?meritev surs:vrednost ?steviloPrebivalcev .
  FILTER (?imeObcineSURS = "Grosuplje")
}
```

- Poizvedba združi **oba vzorca** v enem stavku **WHERE**,
- rezultat vsebuje **identifikator CRP** in **število prebivalcev**.



Je poizvedba res pravilna?





POZOR: to je kartezični produkt

- Vzorca se ujemata nad **dvema nepovezanimi** komponentama grafa,
 - med njima **ni nobene skupne spremenljivke**,
 - rezultat je **kartezični produkt** $\text{CRP} \times \text{SURS}$.
- Tu deluje **po naključju**:
 - filter v CRP pusti **eno samo** rešitev.
 - Brez filtra bi dobili **vse pare** občin CRP in SURS.





Enakost zapišemo eksplicitno



Integracija_CRP_SURS_SPARQL.ttl ↗

```
@prefix owl: <http://www.w3.org/2002/07/owl#>
.
@prefix surs: <https://onto.mdp.gov.si/shema/>
.
@prefix crp:
<https://pzsi.sigov.si/datamodel/ns/crp#> .

surs:Obcina owl:equivalentClass crp:Obcina .
surs:naziv owl:sameAs crp:vrednost .
```

- **Dva trojčka**, ki povesta, da gre za isti pojem:
 - `owl:equivalentClass` – **razreda** sta enakovredna,
 - `owl:sameAs` – **lastnosti** imata isti pomen.
- Datoteko **uvozimo** v GraphDB.





Materializacija: baza zapiše izpeljana dejstva

- Uvožena **2 trojčka** (*Explicit*),
- izpeljanih **1.713 novih trojčkov**:
 - iz **7.507** na **9.220** (*Inferred*).
- Izpeljana dejstva so **shranjena**, ne izračunana sproti.

The screenshot shows a web interface with a search icon, a 'Test' button, and a language selector 'en'. Below is a 'Repository Test' panel with the following data:

Location:	Local
Type:	Graphdb
Access:	Read/Write
Total statements:	19,797
Explicit:	10,577
Inferred:	9,220
Expansion ratio (total/explicit):	2



Posledica: viri se v poizvedbi obnašajo kot eden

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX crp:
<https://pzsi.sigov.si/datamodel/ns/crp#>
PREFIX surs: <https://onto.mdp.gov.si/shema/>

SELECT * WHERE {
  ?obcina rdf:type surs:Obcina .
  ?obcina surs:naziv ?nazivSURS .
  ?obcina crp:vrednost ?nazivCRP .
  ?obcina surs:imaMeritevPrebivalcev ?meritev .
  ?meritev surs:leto ?letoMeritve .
  ?meritev surs:vrednost ?vrednost .
}
```

- **Ena sama** spremenljivka **?obcina** za oba vira,
- lastnost **crp:vrednost** deluje tudi nad **primerki SURS**,
- **ni** kartezičnega produkta.



Kje v poizvedbi je zdaj povezava med viroma?



Kaj smo pridobili s sklepanjem?

Brez sklepanja

- Povezavo med viri nosi **poizvedba**,
- vsak nov vir pomeni **daljšo poizvedbo**,
- tveganje **kartezičnega produkta**.

S sklepanjem

- Povezavo nosijo **podatki**,
- pravilo zapišemo **enkrat**, velja za **vse primerke**,
- nova dejstva so **preverljiva** in **izsledljiva**.



8. Za konec: kdaj se semantika izplača



Kdaj se podjetju semantika izplača?

Izplača se, ko ...

- isti pojem živi v **več sistemih** (ERP, CRM, PLM, spletna trgovina),
- podatke **izmenjujete navzven** – z dobavitelji, kupci, regulatorjem,
- se model **pogosto spreminja** (nove kategorije, trgi, jeziki),
- nad podatki poganjate **LLM** in potrebujete **preverljive** odgovore.

Ne izplača se, ko ...

- gre za **en sam sistem** s stabilno shemo,
- podatkov **ne izmenjujete** navzven,
- so poizvedbe preproste in jih **SQL dobro pokrije**,
- so podatki **enkratni** in se ne uporabijo ponovno.



Utemeljitev 1: ontologija potroji točnost LLM nad bazo

- Vprašanja v naravnem jeziku nad **poslovno bazo SQL**,
 - domena zavarovalništva, vprašanja različnih težavnosti.
- **GPT-4 neposredno nad shemo SQL**: ~16 %
pravih,
- ista vprašanja prek **grafa znanja** (ontologija +
preslikave): ~54 %,
- **isti model**, drugačen opis podatkov,
- z **ontološkim preverjanjem** poizvedbe in
samodejnim popravkom: do ~73 %
(nadaljevanje študije).

Razlika ni v modelu, ampak v **kontekstu**, ki ga
model dobi.

- [arXiv:2311.07509 ↗](#) (osnovni benchmark)
- [arXiv:2405.11706 ↗](#) (preverjanje in popravki)



Utemeljitev 2: semantični sloj prinese več kot menjava modela

- 100 poslovnih vprašanj nad trgovsko shemo (*Contoso*), 5 težavnostnih stopenj,
- primerjava: **gola shema DDL** proti **shemi + 4 KB opisa** metrik in poslovnih pravil,
- preizkušeni **tri jezikovni modeli**.

Kontekst	Točnost
Samo shema baze	~46–51 %
Shema + semantični opis	~ 68 %

- **+17 do +23** odstotnih točk, $p < 0,002$,
- **z opisom** so vsi trije modeli **statistično neločljivi**.
- [arXiv:2604.25149 ↗](#) · [koda ↗](#)



Utemeljitev 3: LinkedIn – podpora strankam z grafom znanja

- Iz **zgodovinskih zahtevkov** zgradili graf znanja,
 - ohranili **strukturo** zahtevka in **povezave** med njimi,
- ob vprašanju poiščejo **podgraf**, ne le podobnega besedila.

Rezultati proti klasičnemu RAG

- **+77,6 %** pri **MRR** (*kakovost priklica*),
- v produkciji **~6 mesecev**:
 - **-28,6 %** mediana časa do rešitve zahtevka.
- [🔗 SIGIR 2024, arXiv:2404.17723 ↗](#)



Utemeljitev 4: finančni dokumenti – graf in vektorji skupaj

- Transkripti **objav poslovnih rezultatov** (NVIDIA in BlackRock),
- **vektorski RAG**: dober nad prostim besedilom, šibek pri **razmerjih med entitetami**,
- **graf znanja**: močan pri strukturi, šibkejši pri **nestrukturiranem besedilu**,
- **kombinacija prekaša oba** pristopa posamično.

Graf znanja ni **zamenjava** za vektorsko iskanje, ampak njegova **dopolnitev**.

- [arXiv:2408.04948 ↗](#)
- Sorodno – Microsoft **GraphRAG** (odprta koda):
 - [arXiv:2404.16130 ↗](#)



Utemeljitev 5: najcenejši vstop – schema.org

- **Lahka semantika** na spletnih straneh:
 - izdelki, recepti, dogodki, delovna mesta,
- brez ontologij in brez baze RDF,
- Googlove **objavljene študije primerov**:
 - Rotten Tomatoes: **+25 %** CTR na 100.000 straneh,
 - Nestlé: **+82 %** CTR za strani z bogatimi rezultati,
 - Food Network: **+35 %** obiskov.

Isti strukturirani zapis danes berejo tudi **jezikovni modeli in agenti**.

-  developers.google.com/search/case-studies
↗
-  schema.org ↗



Kje to že deluje: trije zreli primeri

Farmacija – najbolj zrel trg

- AstraZeneca: graf povezanih biomedicinskih virov za hitrejšo preverjanje hipotez,
- lasten *Biomedical Insights KG* iz besedil in strukturiranih virov.
- [🔗 Ontotext: študija primera ↗](#)

Proizvodnja – najbližje slovenski publiki

- Bosch: graf znanja za **nadzor varjenja** v avtomobilski proizvodnji (ISWC 2022),
- danes: **LLM pomaga pisati preslikave** podatkov v graf (TGDK 2025).
- [🔗 Bosch: varjenje ↗](#) · [LLM preslikave ↗](#)



Kaj je danes mogoče kupiti?

Kategorija	Primeri
Semantični sloj z agentom	Stardog Voicebox ↗
Platforme RDF	Graphwise ↗ (<i>GraphDB + PoolParty</i>), metaphacts, eccenca
GraphRAG nad grafi lastnosti	Neo4j ↗ , FalkorDB
Ontologija kot jedro platforme	Palantir Foundry / AIP ↗
Semantični sloj za poslovno analitiko	Cube ↗ , dbt Semantic Layer, AtScale



„Ali ni modeliranje predrago?”

Ugovor je bil dolga leta upravičen

- Ontologije so gradili **redki strokovnjaki**,
- ročno pisanje **preslikav** je bilo drago,
- vzdrževanje je zaostajalo za spremembami.

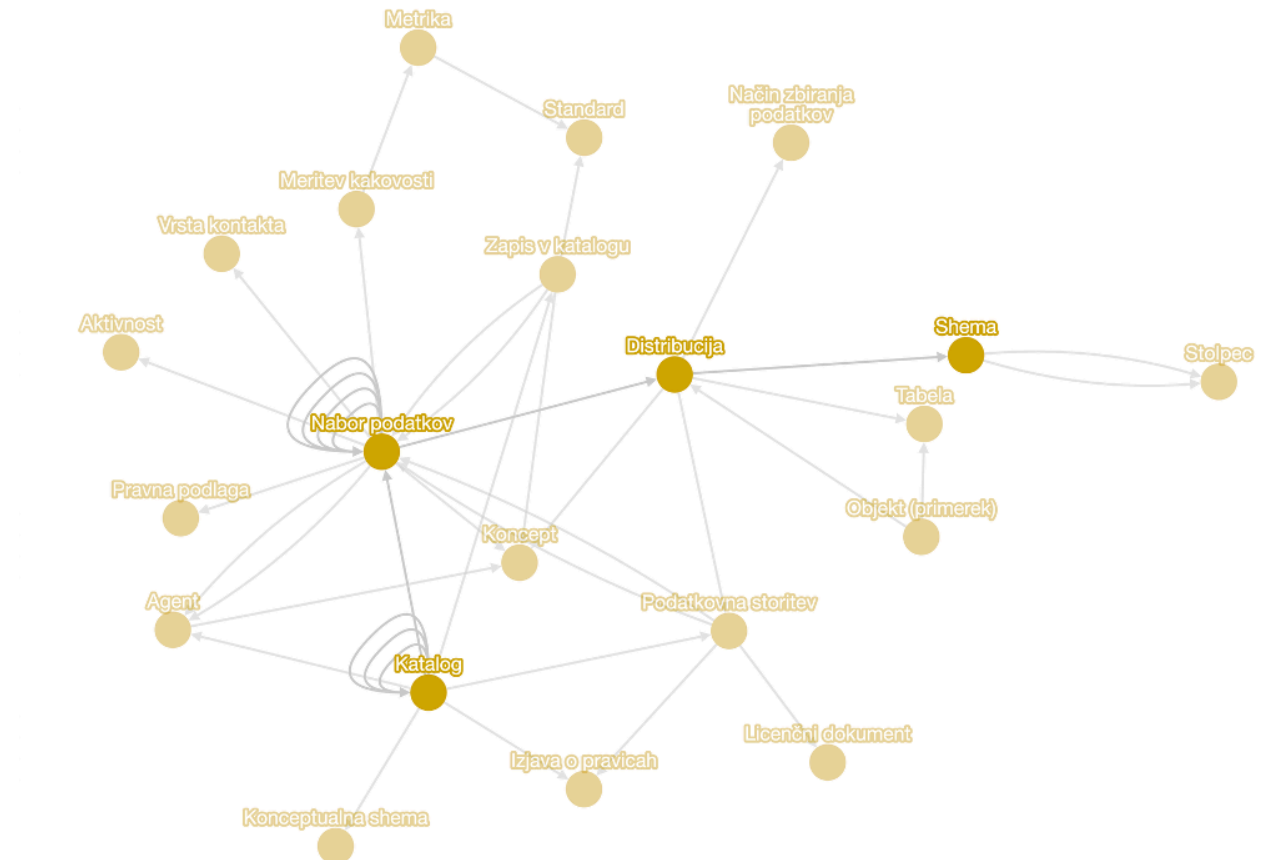
Kar se je spremenilo

- LLM-ji **predlagajo** razrede, lastnosti in hierarhije,
 - izziv **LLMs4OL** ↗ pri konferenci ISWC,
- LLM-ji **pišejo preslikave** iz virov v graf (*Bosch, 2025*),
- človek **preverja**, ne tipka.



Primer 1: Podatkovni zemljevid (UL FRI in MDP)

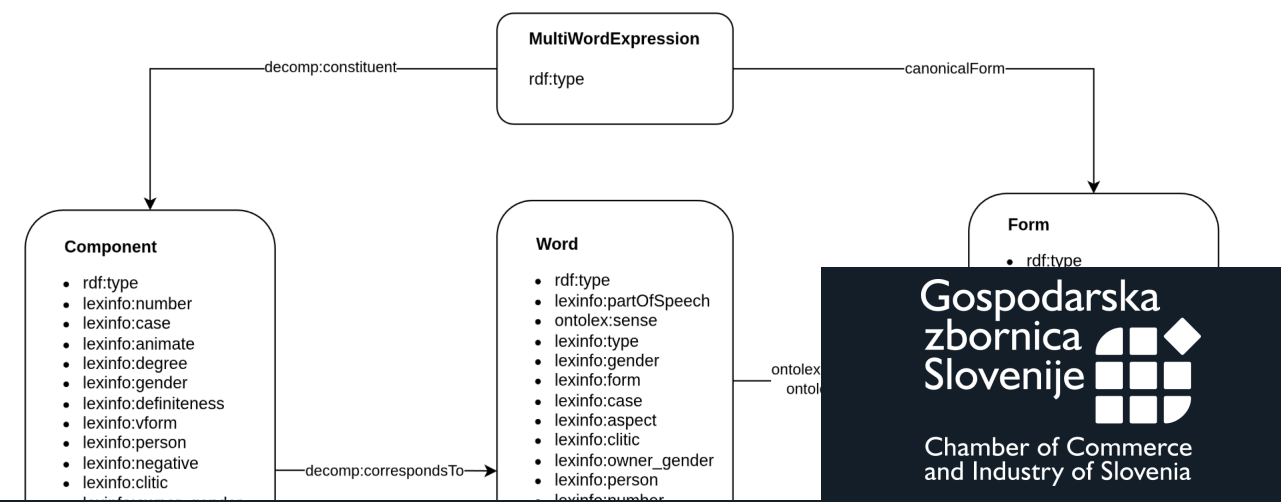
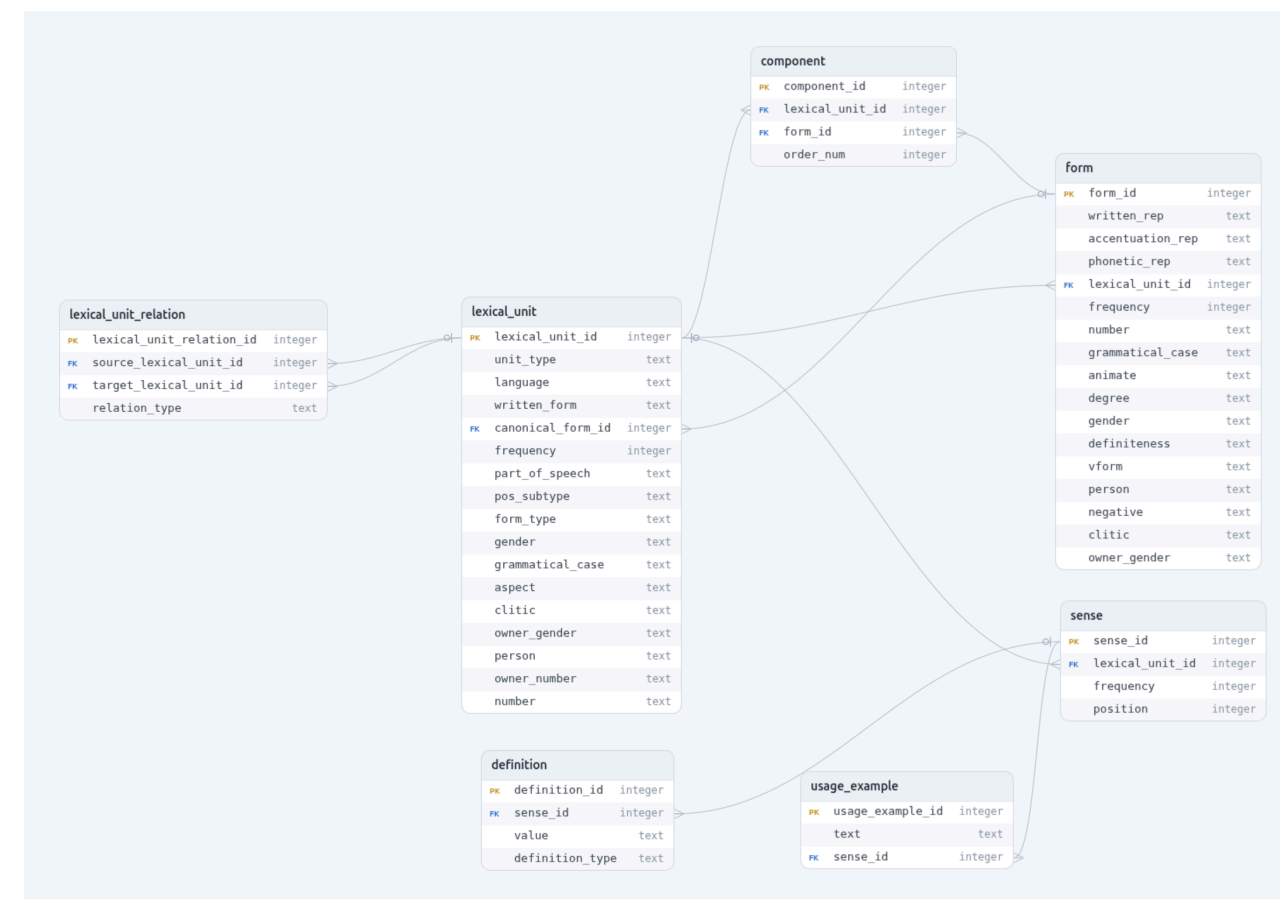
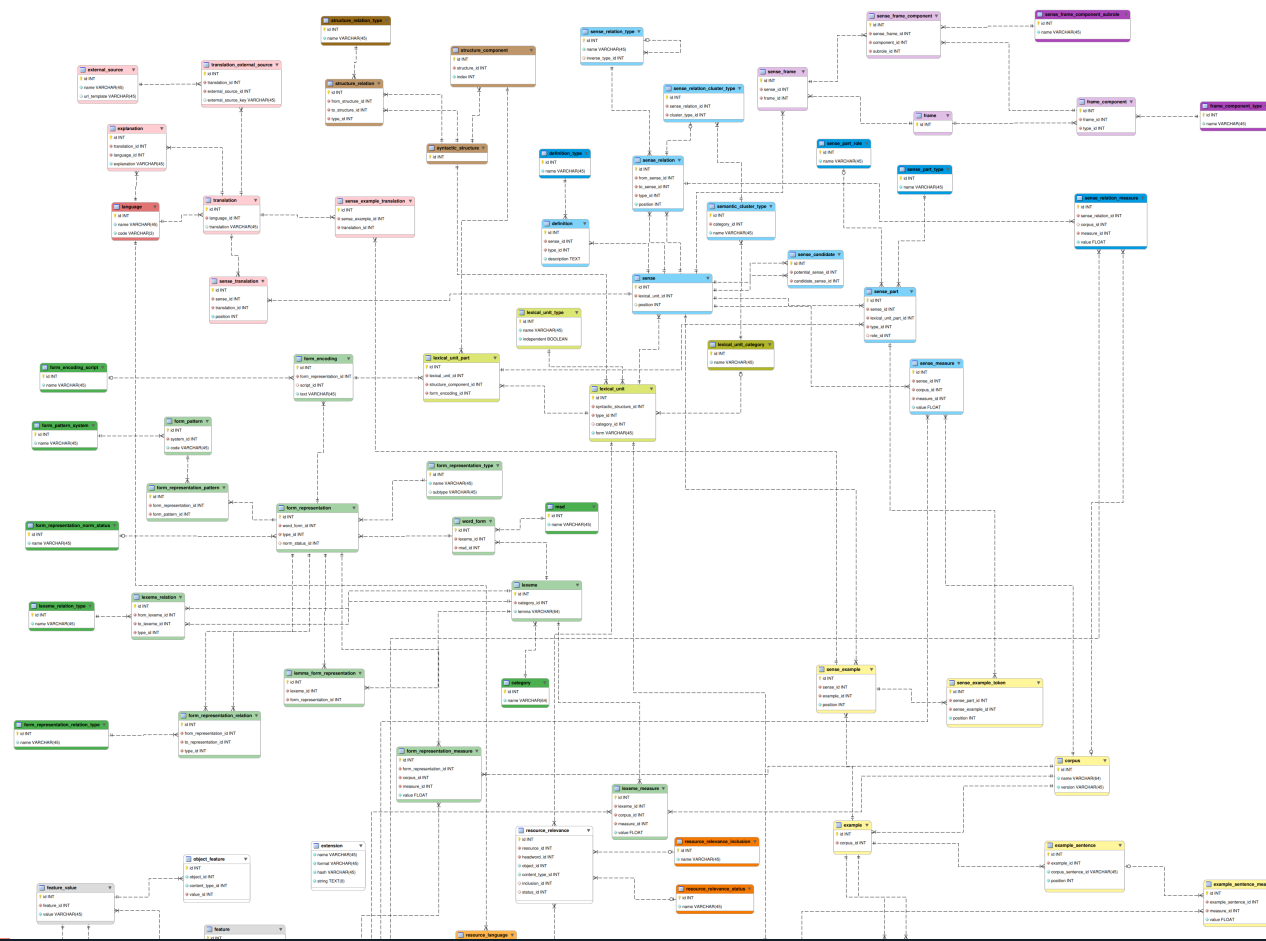
- Prototip za **Ministrstvo za digitalno preobrazbo**,
- nad slovenskim **portalom odprtih podatkov** (CKAN),
- v **enem pogledu** prikaže:
 - **shemo** (ontologijo) in
 - **podatke** (primerke).
- Temelji na **DCAT** in sorodnih slovarjih,
- podatke bere iz vmesnika **SPARQL**.
-  [Demo Podatkovnega zemljevida](#) ↗
-  github.com/FRI-MDP/ckan-datamap ↗





Primer 2: Jezikoslovni podatki (UL FRI, projekt LLM4DH)

- znanje o slovenskem jezikoslovju
-  **Relacijska baza** ↗
- cilj - odgovarjati na *Kako bi razložili pojem »skupnost«?*, *Kaj so deležniške oblike glagola »skuhati«?*, *Navedi tožilnik množine za samostalnik »obraz«*.





Zakaj je semantika pomembna za javno upravo?

Medsektorsko

- Isti pojem živi v **več registrih**:
 - občina v CRP, SURS, prostorskih podatkih,
- vsak register ima **svoj identifikator** in svoje ime,
- brez skupnega pomena je vsaka izmenjava **nov projekt**.

Meddržavno

- EU zahteva **pomensko interoperabilnost**:
 - *Interoperable Europe Act* – **Uredba (EU) 2024/903** ↗,
 - načelo **samo enkrat** (*angl. once-only*),
- skupni slovarji: **DCAT-AP**, *Core Vocabularies*,
-  data.europa.eu ↗



Ontologija ni le za odgovore, ampak za agente

- Ko agent le **odgovarja**, je napaka neprijetna,
- ko agent **ukrepa** – naroči, rezervira, spremeni zapis – je napaka **draga**.
- Zato potrebuje:
 - **tipiziran** objektni model,
 - **dovoljene akcije** nad objekti,
 - **revizijsko sled**, od kod podatek izvira.

To je natanko tisto, kar smo risali na prvi prosojnici – le da je zdaj to prodajni argument, ne raziskovalna ideja.





Podatkovni prostori: semantika kot pogoj za vstop

- **Catena-X** (*avtomobilska industrija*) in **Manufacturing-X**:
 - podatkovni modeli so **standardizirani** (SAMM),
 - odprta koda: **Eclipse Tractus-X**.
- **Digitalni potni list izdelka (DPP)**:
 - Uredba o baterijah (EU) 2023/1542 ↗,
 - od **18. 2. 2027**: QR in potni list za baterije na trgu EU.

Za dobavitelja to ni več tehnološka izbira, ampak **pogoj za posel**.

-  catena-x.net ↗
-  [eclipse-tractusx](https://eclipse-tractusx.com) ↗



Napotki: kako in kdaj začeti

Kdaj

1. Začnite pri **enem vprašanju**, ki danes zahteva ročno združevanje virov.
2. Semantiko dodajte **nad obstoječe sisteme**, ne namesto njih.
3. Najprej **dogovor o pojmi**, šele nato tehnologija.
4. Če podatkov ne izmenjujete in je shema statična – **počakajte**.

Kako

5. **Uporabite obstoječe slovarje** – schema.org, DCAT, SAMM, domenske ontologije.
6. **Stabilni identifikatorji** so pomembnejši od izbire formata.
7. Pri LLM: **graf za dejstva in sledljivost**, vektorji za besedilo.
8. Spremljajte **podatkovne prostore svoje panoge** – modeli so tam že dogovorjeni.



9. Diskusija: Ali lahko identificirate kak primer, kjer bi uporaba semantike bila za vaše podjetje ali organizacijo koristna?

