# DSA Specifikacija

**VSSA**

**2024-12-17**

# Turinys

**Ši versija:**
DSA 1.0

**Klaidų sekimas:**
Github

**Redaktorius:**
VSSA

**Vertimas (nenorminis):**
Nėra

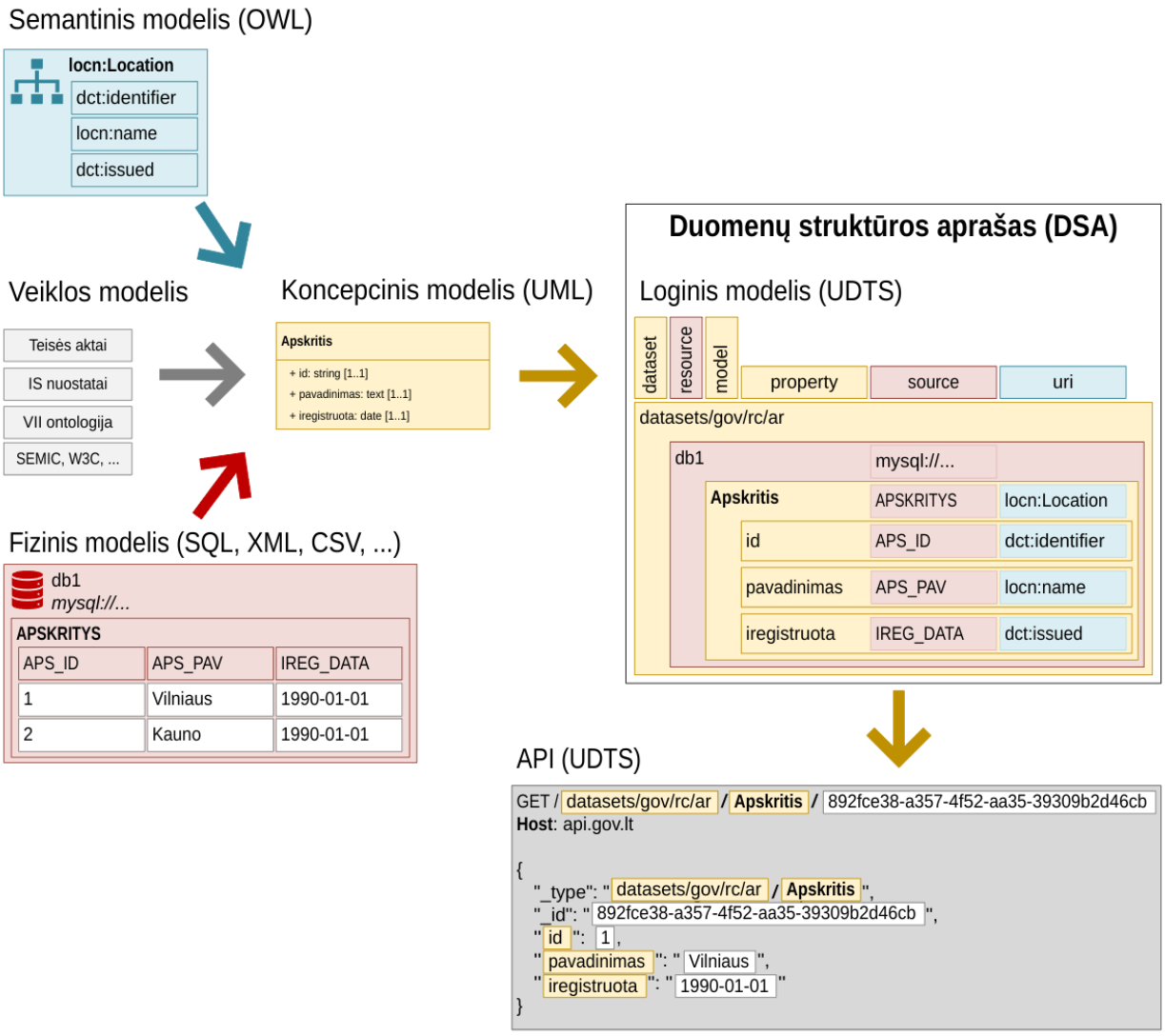
**Naujausia paskelbta versija:**
Naujausia paskelbta versija

**Naujausias redaktoriaus juodraštis:**
Naujausias redaktoriaus juodraštis

**Šis dokumentas taip pat pateikiamas šiais nenorminiais formatais:**
PDF

Čia rasite *Duomenų struktūros aprašo* (*DSA*) lentelės specifikaciją.

*Duomenų struktūros aprašas* yra lentelė skirta fizinio, loginio ir semantinio duomenų modelių susiejimui, prieigos lygio nustatymui ir duomenų brandos lygio vertinimui.




*Koncepcinis modelis* yra UML klasių diagrama, sudaryta laikantis Conceptual model conventions (UML) reikalavimų. Koncepcinis modelis laikomas kaip vienintelis tiesios šaltinis ir yra sudaromas remiantis teisės aktais, informacinės sistemos nuostatais, semantiniais žodynais ir duomenų modeliu iš duomenų šaltinio.

*Fizinis modelis* šio dokumento kontekste yra duomenų schema apibūdinanti kur ir kaip duomenys yra saugomi ir kaip juos pasiekti. Schema apibrėžianti duomenų modelį priklauso nuo duomenų saugojimo formato. Jei duomenys saugomi SQL duomenų bazėse, tada DSA lentelėje nurodomi lentelių ir stulpelių pavadinimai, XML atveju nurodomos XPath išraiškos, JSON atveju nurodomos JSONPath išraiškos. DSA lentelėje fizinis modelis nurodomas source stulpelyje.

*Loginis modelis* yra duomenų schema, kuri naudojama duomenų apsikeitimui UDTS protokolu, loginis modelis rengiamas pagal koncepcinį modelį ir yra artimas semantiniam modeliui, tačiau skirtas duomenų publikavimui per API. Loginis modelis siejamas su fiziniu ir semantiniu modeliais.

*Semantinis modelis* yra nepriklausomas nuo to, kaip duomenys saugomi ar perduodami fiziškai, siejamas su tarptautiniais standartais ir plačiai naudojamais sąvokų žodynais.

SKYRIUS 1

# Specifikacijos

- Universali duomenų teikimo sąsaja (UDTS)
- Duomenų katalogo Lietuvos taikymo profilis (DCAT-AP-LT)

SKYRIUS 2

# Turinys

## 2.1 Koncepcinis modelis

Prieš pradedant darbą su strūktūros aprašais, būtina pasirengti koncepcinio modelio UML diagramą, vadovaujantis Conceptual model conventions (UML) reikalavimais.

Koncepcinis modelis vienareikšmiškai apibrėžia duomenų modelį grafine forma ir naudojamas, kaip vienintelis tiesos šaltinis, kadangi vizualinę duomenų modelio reprezentaciją nesunkiai gali suprasti skirtingose srityse dirbandys žmonės ir pasitivrtinti duomenų modelį, kuris vėliau bus taikomas rengiant duomenų struktūros aprašus.

Reikia atkreipti dėmesį, kad koncepcinis modelis yra vienas, o jį atitinkančių duomenų šaltinių gali būti daug.

Kaip pavyzdį, naudosime žemiau pateiktą koncepcinį duomenų modelį:

Pagal šį koncepcinį modelį, DSA lentelė atrodytu taip:

| d | r | m | property | type | ref | prepare |
|---|---|---|---|---|---|---|
| datasets/gov/example | | | | | | |
| | | **Administracija** | | | kodas | |
| | | | kodas | string | | |
| | | | pavadinimas | string | | |
| | | | tipas | string | | |
| | | | | enum | | "APSKRITIS" |
| | | | | | | "SAVIVALDYBE" |
| | | **Gyvenviete** | | | id | |
| | | | id | integer | | |
| | | | pavadinimas | string | | |
| | | | savivaldybe | ref | **Savivaldybe** | |
| | | **Apskritis** | | **Administracija** | kodas, tipas | |
| | | | kodas | string | | |
| | | | pavadinimas | string | | |
| | | | tipas | string | | "APSKRITIS" |
| | | **Savivaldybe** | | **Administracija** | kodas, tipas | |
| | | | kodas | string | | |
| | | | pavadinimas | string | | |
| | | | tipas | string | | "SAVIVALDYBE" |
| | | | apskritis | ref | **Apskritis** | |

Pavadinimai nurodyti koncepciniame modelyje, turi identiškai sutapti su pavadinimai nurodytais DSA lentelės loginio modelio `model`, `property`, `type`, `ref` ir `prepare` stulpeluose.

DSA lentelėje fizinio modelio, `source` stulpelyje nurodyti pavadinimai skirtinguose šaltiniuose gali skirtis, tačiau loginio modelio pavadinimai turi išlikti tokie patys.

### 2.1.1 Objektas

*Objektas* yra viena duomenų eilutė, arba vienas duomenų įrašas ar atvejis. Kalbant apie objektus, naudojamas pavyzdys žymėjimas.

Pavyzdžiui iš aukščiau pateikto duomenų modelio, klasės `Gyvenviete` objektai gali būti:

- Vilnius
- Kaunas
- Klaipeda

Sąvoka *objektas* kalba apie konkretų individualų atvejį ar pavyzdį.

Imant duomenų lentelę iš `Gyvenviete` modelio, gausime tokius duomenis.

| id | pavadinimas | savivaldybe |
|---|---|---|
| 1 | Vilnius | 10 |
| 2 | Kaunas | 11 |
| 3 | Klaipeda | 12 |

Šioje lentelėje yra trys objektai.

Objekto pavyzdys UML diagramoje:

UML diagramoje turime tris objetus Vilnius, Kaunas ir Klaipeda, priskirti klasei `Gyvenviete`.

Skirtingi objektai gali būti klasifikuojami į klases arba esybes.

### 2.1.2 Klasė

Klasė arba Esybė yra vienodas savybes ir vienodą apibrėžimą turinčių objektų aibė, kuriems suteikiamas tam tikras pavadinimas.

Tarkime Vilniaus, Kauno ir Klaipėdos objektus galime priskirti vienai klasei ir suteikti tai klasei pavadinimą `Gyvenviete`.

Klasės pavyzdys UML diagramoje:

Klasė gali neturėti jokių savybių, arba gali turėti savybes, kurios apibūdina pačią klasę.

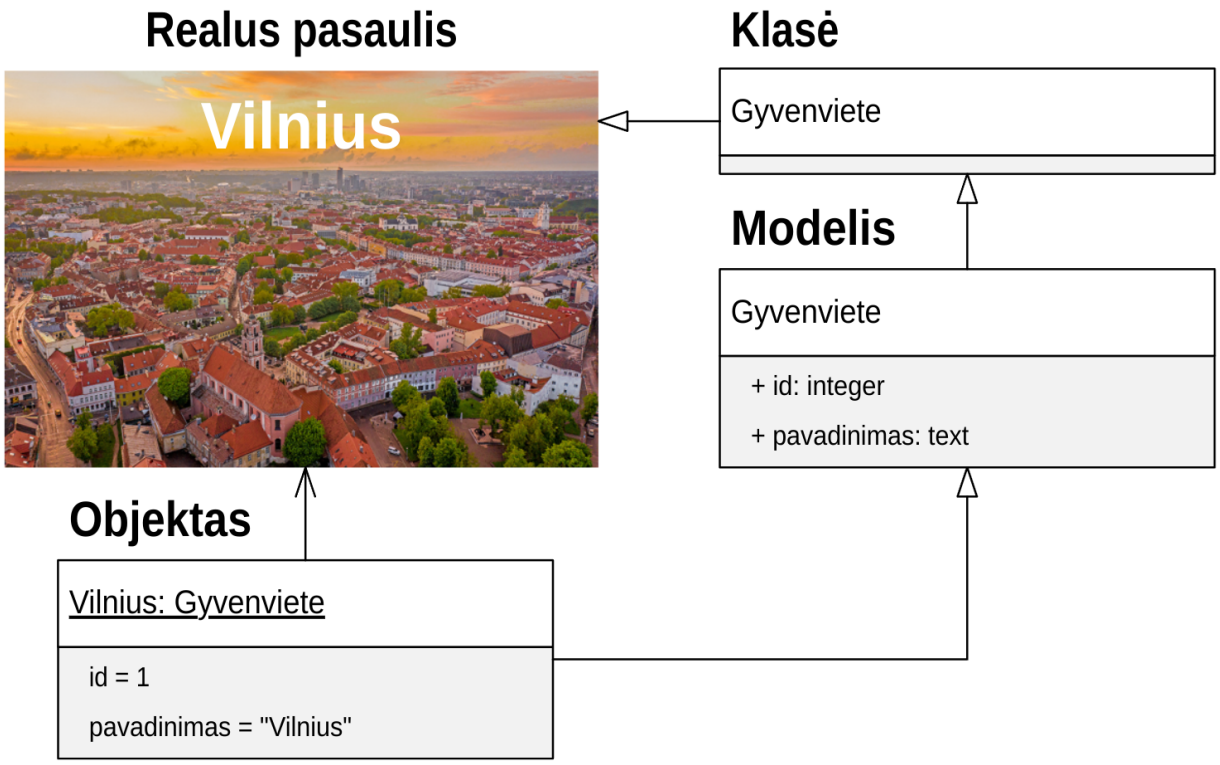
Tarkime modelis `Gyvenvietė` turi savybę `pavadinimas`, tačiau tai nėra klasės savybė, todėl, kad `pavadinimas` yra duomenų atributas, kuris nėra klasę apibūdinanti savybė.

Nurodžius savybes prie klasės, iškeliamas griežtas reikalavimas, visiems modeliams ir subklasėms, atitikti visas klasės savybes.

Tuo tarpu duomenų modelis, gali atitikti tam tikrą klasę, bet gali būti pateikiamas su skirtingomis savybėmis.

Sudarant ontologijas, pateikiami klasių apibrėžimai, dažniausiai be savybių, kad neriboti klasės taikymo. Tačiau tam tikrais atvejais, ontologijoje klasės pateikiamos ir su keliomis savybėmis, kurios apibrėžia pačią klasę.

### 2.1.3 Modelis




Klasės savybės apibrežia pačią klasę ir tampa klasės dalimi, tačiau modelio savybės neturi įtakos klasės semantiniam apibrėžimui, tai yra tiesiog duomenų laukų sąrašas pateikiams su klase.

Viena klasė gali turėti daug skirtingų modelių, su skirtingomis savybėmis arba su skirtingais duomenų laukais.

Modelis, schema arba profilis yra konkretus savybių, duomenų tipų sąrašas, kuriame nurodoma kurios savybės yra privalomos, kurios gali turėti daugiau nei vieną reikšmę ir kitas detales.

Sudarant taikymo profilius (angl. *Application profile*) UML klasių diagramoje pateikiami konkretūs duomenų modeliai, su konkrečiomis savybėmis ir jų tipais.

Modelio pavyzdys UML diagramoje:

Modelis atvaizduojas lygiai taip pat, kaip ir klasės. Ar tai yra klasės ar modelis galima atskirti tik pagal diagramos pavadinimą, jei diagrama vaizduoja ontologiją, tada joje yra klasės, jei taikymo profilį, tada diagramoje yra modeliai.

Jei UML diagramose prie klasių yra pateikti pilni sąrašai savybių su tipais, tada tai greičiausiai yra taikymo profilis.

### 2.1.4 Generalizacija

Objektai gali būti skirstomi į klases, tačiau pačios klasės gali būti skirstomos į bendresnes klases, toks apibendrinimo procesas vadinamas generalizacija.

UML diagramose gneralizacija žymima užpildyta rodykle, kurios krypts iš labiau specializuotos siauresnę prasmę turinčios klasės, į labiau apibendrintą, platesnę prasmę turinčią klasę, pavyzdžiui:

Šiame pavyzdyje nurodome, kad `Savivaldybe` yra `Administracija` poaibis. Arba `Administracija` yra platesnė klasė, o `Savivaldybę` yra siauresnė, labiau specifinę prasmę nurodanti klasė.

### 2.1.5 Identifikatorius

Kad galėtume vienareikšmiškai įvardinti ar nurodyti tam tikrą objektą, visi objektai privalo turėti unikalius identifikatorius.

Kiekvienam objektui priskiriamas vienas globalus identifikatorius **UUID** formatu, tačiau objektas gali turėti vieną ar daugiau lokalius identifikatorius.

Globalūs identifikatoriai priskiriami esybei ir atspindi vieną realaus pasaulio objektą, lokalus identifikatorius yra siejams su konkrečiu duomenų modeliu ar duomenų šaltiniu ir skirtinguose modeliuose gali būti naudojami keli skirtingi lokalūs identifikatoriai rodantys į vieną realaus pasaulio objektą.

UML diagramoje aukščiau turime du skirtingus duomenų objektus, kurie turi vienodą globalų identifikatorių `dd79d2a6-d3d6-4fc2-83bb-da9dd15b2a89`, tačiau skirtingus lokalius `id = 7` ir `kodas = 23`.

Globalus identifikatorius suteikiamas esybei `Gyvenviete`, lokalūs identifikatoriai suteikiami konkrečiam duomenų modeliui ir konkrečiam duomenų šaltiniui.

Rengiant *DSA* lentelę globalūs identifikatoriai žymimi *`model.ref`* stulpelyje arba rezervuotu savybės pavadinimu `_id` ir yra privalomas.

### 2.1.6 Savybė

UML diagramos savybės žymimos sutartine forma:

---

**Sintaksė**

**access property** : **type** [ **cardinality** . . **multiplicity** ]

---

**access**
: Prieigos lygis. Gali būti naudojami tokie žymėjimai:

  - \+ - atviri duomenys, žiūrėti *`open`*.
  - \# - vieši duomenys, žiūrįti *`public`*.
  - ~ - duomenys teikiami pagal sutartį, žiūrėti *`protected`*.
  - \- - nepublikuojami duomenys, žiūrėti *`private`*.

**property**
: Savybė, žiūrėti *`property`*. Nurodoma savybės URI forma.

**type**
: Duomenų tipas, žiūrėti *Duomenų tipai*. UML diagramose, jei duomenų tipas yra *`ref`* arba `backref`, tada nurodomas modelio pavadinimas, URI forma, su kuriuo daroma asociacija.

**cardinality**
: Nurodo ar laukas yra privalomas:

  - `0` - laukas yra neprivalomas.
  - `1` - laukas yra privalomas.

**multiplicity**
: Nurodo kiek kartų gali būti pateikta lauko reikšmė.

  - `1` - lauko reikšmė gali būti pateikta tik vieną kartą.
  - * - laukė reikšmė gali būti pateikta daugiau nei veiną kartą.

Pavyzdys:

UML diagramoje matote `Gyvenviete` duomenų modelį, kuris turi dvi savybes:

```
+ id: integer [1..1]
+ pavadinimas: text [1..1]
```

Abi savybės turi atvirą prieigos lygmenį, `id` ir `pavadinimas` kodinius savybės pavadinimus, `integer` ir `text` duomenų tipus ir abi savybės yra privalomos ir gali turėti tik vieną reikšmę.

### 2.1.7 Asociacija

#### Per duomenų tipą

UML diagramose nurodant ryšį su kitomis esybėmis, galima naudoti įprastą savybių žymėjimo formą `+ savivaldybe: Savivaldybe [1..1]`, kur po `:` dvitaškio nurodomas kitas modelis, su kuriuo daroma asociacija.

Tokia asociacija daroma, kai siejame su išoriniais modeliais, arba kai turime per daug asociacijų ir norime UML diagramoje sumažinti rodyklių skaičių.

#### Tiesioginė

Tiesioginė asociacija nurodoma rodyklės pagalba, jei yra pateikta rodyklė, tada savybių sąraše, savybės, kuri yra pateikta prie rodyklės neberodome.

Rodyklės kryptis visada rodo iš modelio, prie kurio savybė yra apibrėžta, į kitą modelį, su kuriuo savybė yra siejama.

Tiesioginė asociacija *DSA* yra nurodoma *`type.ref`* pagalba.

#### Atvirkštinė

Asociacijai gali būti naudojami ir atvirkštiniai ryšiai, pavyzdžiui:

Šiuo atveju nurodome *type.backref* tipo atvirkštinę asociaciją, rodyklės kryptis ir daugia-reikšmiškumas keičiasi, turime vieną savyvaldybę, kuri gali turėti daug gyvenviečių.

### 2.1.8 Klasifikatorius

Klasifikatoriai arba kontroliuojami žodynai, yra galimų reikšmių sąrašas naudojamas tam tikrai savybei.

UML diagramoje klasifikatoriai pateikiami naudojant `<<enumeration>>` stereotipą ir punktyrinę priklausomybės rodyklę:

`AdministracijosTipas` yra klasifikatorius, turintis kontroliuojamą žodyną, kuriame apibrėžtos dvi galimos reikšmės `APSKRITIS` ir `SAVIVALDYBE`.

Struktūros apraše klasifikatoriai aprašomi naudojant *enum* dimensiją.

### 2.1.9 Žodynas

Visos klasės ir savybės (*sąvokos*) yra skirstomos į žodynus. Dažnai viename duomenų modelyje yra naudojamos sąvokos iš skirtingų žodynų.

Kad atskirti, kuri sąvoka yra iš kokio žodyno, naudojami žodyno prefiksai.

Žodyno prefiksai gali būti naudojami tiek klasės pavadinime, tie savybių ir tipų pavadinimuose.

Jei žodyno prefiksas nėra nurodytas, tai reiškia, kad naudojamas esamas žodynas, kuris yra apibrėžtas duomenų modelyje.

Žodynai taip pat gali būti nurodomi naudojant UML paketus arba vardų erdves:

Sąvokoms, kurios yra vardų erdvės rėmuose, žodyno prefiksai nenurodomi. Žodyno prefiksai nurodomi tik tuo atveju, jei sąvoka yra iš kito žodyno.

### 2.1.10 IRI

Visos sąvokos, tokios kaip klasės, savybės, duomenų tipai, taip pat yra objektai, turintys savo identifikatorius.

UML diagramose nurodomi būtent sąvokų identifikatoriai sutrumpinta IRI forma.

IRI yra identifikatorius schema sudaryta iš sekančių komponentų:

**scheme** `://` **host** / **path** ? **query** # **fragment**

Lietuvos viešąjame sektoriuje naudojama sekanti URI schema:

`https://data.gov.lt/id/` **vocab** / **term** [ / **id** ]

**vocab**
: Žodyno kodinis pavadinimas.

**term**
: Sąvokos kodinis pavadinimas.

**id**
: Objekto identifikatorius.

Jei mūsų kuriamam žodynui būtų suteiktas kodinis pavadinimas `adresai`, tada mūsų sąvokoms būtų suteikti tokie IRI identifikatoriai:

```
https://data.gov.lt/id/adresai/Gyvenviete
https://data.gov.lt/id/adresai/id
https://data.gov.lt/id/adresai/pavadinimas
```

Kadangi pilnas IRI yra gan ilgas, UML diagramose naudojame sutrumpintą IRI formą su prefiksu. Šiuo atveju, galime deklaruoti, kad `ar` prefiksas atitinka `https://data.gov.lt/id/adresai/` URI, todėl sutrumpinta forma atrodys taip:

```
@prefix ar: <https://data.gov.lt/id/adresai/>

ar:Gyvenviete
ar:id
ar:pavadinimas
```

UML diagramoje naudojame sutrumpintus URI pavadinimus, tačiau kartu su diagrama būtina pateikti ir prefiksų sąrašą, kad būtų ašku, ką reiškia kiekvienas prefikas:

| Prefiksas | Vardų erdvės IRI |
|---|---|
| ar | https://data.gov.lt/id/adresai/ |
| locn | http://www.w3.org/ns/locn# |
| dct | http://purl.org/dc/terms/ |
| rdfs | http://www.w3.org/2000/01/rdf-schema# |

## 2.2 Lentelės formatas

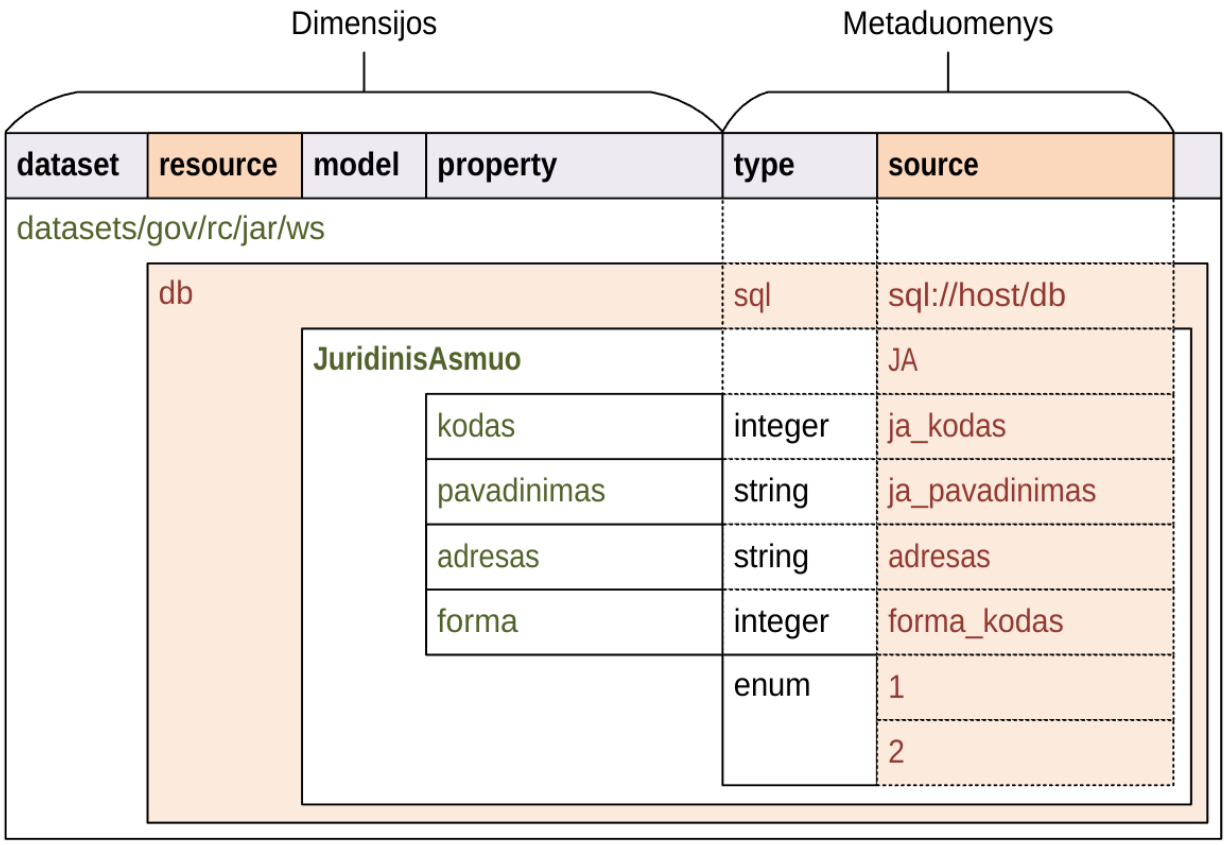
*DSA* yra sudarytas taip, kad būtų patogu dirbti tiek žmonėms, tiek programoms. Žmonės su *DSA* lentele gali dirbti naudojantis, bet kuria skaičiuoklės programa (Excel, LibreOffice Calc) ar kitas pasirinktas priemones. Kadangi *DSA* turi aiškią ir griežtą struktūrą, lentelėje pateiktus duomenis taip pat gali lengvai nuskaityti ir interpretuoti kompiuterinės programos.

Tais atvejais, kai su *DSA* lentele dirba žmonės, lentelė gali būti saugoma įstaigos pasirinktos skaičiuoklės programos ar kitų priemonių formatu.

Automatizuotoms priemonėms *DSA* turi būti teikiamas CSV formatu laikantis **RFC 4180** taisyklių, failo koduotė turi būti UTF-8.

DSA lentelė gali būti importuojama į Duomenų katalogą, kuriame DSA lentelės turinys gali būti tvarkomas naudojantis grafine naudotojo sąsaja.

Rengiant duomenų struktūros aprašus darbas vyksta su viena lentele. Lentelės stuleliai sudaryti iš dimensijų ir metaduomenų.




Lentelė sudaryta hierarchiniu principu. Kiekvienas metaduomenų stulpelis gali turėti skirtingą prasmę, priklausomai nuo dimensijos. Todėl toliau dokumentacijoje konkrečios dimensijos stulpelis yra žymimas nurodant tiek dimensijos, tiek metaduomenų pavadinimus, pavyzdžiui *`property.type`*, kuris nurodo *`type`* metaduomenų stulpelį, esantį *`property`* dimensijoje.

### 2.2.1 Dimensijos

Duomenų struktūros aprašo lentelė sudaryta hierarchiniu principu. Kiekvienos lentelės eilutės prasmę apibrėžia *Dimensijos* stulpelis.

Kiekvienoje eilutėje gali būti užpildytas tik vienas dimensijos stulpelis.

Be šių penkių dimensijų, yra kelios *papildomos dimensijos*, jos nurodomos *`type`* stulpelyje, neužpildžius nei vieno dimensijos stulpelio.

**`dataset`**

**Duomenų rinkinys**

Kodinis duomenų rinkinio pavadinimas. Naudojant duomenų rinkinio kodinį pavadinimą formuojamas API.

Duomenų rinkinio kodinis pavadinimas užrašomas pagal tokį šabloną:

`datasets/` **forma** / **organizacija** / **katalogas** / **rinkinys**

Visi duomenų rinkinio pavadinimo komponenta užrašomi mažosiomis raidėmis, jei reikia žodžiai atskiriami _ simbolio pagalba.

**forma**
Nurodo organizacijos teisinę formą, galimi variantai:

**gov** - Viešasis sektorius.
**com** - Privatusis sektorius.

**organizacija**
Organizacijos pavadinimo trumpinys. Viena organizacija gali turėti vieną trumpinį, kuris yra registruojamas *duomenų kataloge*.

**katalogas**
Organizacijos informacinės sistemos trumpinys.

**rinkinys**
Informacinės sistemos teikiamas duomenų rinkinys.

Visi pavadinimai užrašomi mažosiomis lotyniškomis raidėmis, žodžiams atskirti gali būti naudojamas _ simbolis.

Pagal semantinę prasmę atitinka dcat:Resource.

---

**Pavyzdys**

```
datasets/gov/rc/jar/ws
datasets/gov/ivkp/adp/adk
```

---

**Taip pat žiūrėkite:**

*dataset*
*Kodiniai pavadinimai*

`resource`
**Duomenų šaltinis**

Kodinis duomenų šaltinio pavadinimas, užrašomas mažosiomis lotyniškomis raidėmis, žodžiai skiriami _ simboliu.

Duomenų šaltinis yra duomenų failas, duomenų bazė ar API, per kurį teikiami duomenys.

Pagal semantinę prasmę atitinka dcat:Distribution arba rml:logicalSource.

---

**Pavyzdys**

```
resource1
db1
```

**Taip pat žiūrėkite:**

*resource*
*Duomenų šaltiniai*

**base**

**Modelio bazė**

Nebepalaikoma nuo 0.2 versijos: Atskira modelio bazė naikinama. Nuo 0.2 versijos, modelio bazė nurodoma *model.type* stulpelyje.

Kodinis bazinio modelio pavadinimas. Atitinka rdfs:subClassOf prasmę (*model* `rdfs:subClassOf` *base*). Žiūrėti *base*.

**model**

**Modelis (lentelė)**

Kodinis modelio pavadinimas, užrašomas lotyniškomis raidėmis, kiekvieno žodžio pirma raidė didžioji, kitos mažosios, žodžiai atskiriami didžiąja raide.

Pagal semantinę prasmę atitinka rdfs:Class arba r2rml:SubjectMap.

**Pavyzdys**

```
Gyvenviete
AdministracijosTipas
```

**Taip pat žiūrėkite:**

*model*
*Modelis*

**property**

**Savybė (stulpelis)**

Kodinis savybės pavadinimas, užrašomas mažosiomis lotyniškomis raidėmis, žodžiai atskiriami _ simboliu.

Savybių pavadinimai prasidedantys _ simboliu yra rezervuoti ir turi apibrėžtą prasmę.

Savybės pavadinime gali būti naudojami tokie specialūs simboliai:

- (taško simbolis) nurodo objektų kompoziciją. Naudojamas su *ref* ir *object* duomenų tipais.

  **Pavyzdys**

```
adresas.gatve
```

---

**[]**

Duomenų masyvas arba sąrašas, gali būti naudojamas su visais tipais.

---

**Pavyzdys**

```
miestai[]
```

---

**@**

Kalbos žymė, naudojama su *text* tipu.

---

**Pavyzdys**

```
pavadinimas@lt
pavadinimas@en
```

---

Pagal semantinę prasmę atitinka rdfs:Property, r2rml:PredicateObjectMap.

**Taip pat žiūrėkite:**

*property*

### 2.2.2 Metaduomenys

Kaip ir minėta aukščiau, kiekvienos metaduomenų eilutės prasmė priklauso nuo *Dimensijos*. Todėl, toliau dokumentacijoje, kalbant apie tam tikros dimensijos stulpelį, stulpelis bus įvardinamas pridedant dimensijos pavadinimą, pavyzdžiui *model.ref*, kas reikštų, kad kalbama apie *ref* stulpelį, *model* dimensijoje.

**id**

**Eilutės identifikatorius**

Unikalus elemento numeris, gali būti sveikas, monotoniškai didėjantis skaičius arba **UUID**. Svarbu užtikrinti, kad visi elementai turėtu unikalų id.

Šis stulpelis pildomas automatinėmis priemonėmis, siekiant identifikuoti konkrečias metaduomenų eilutes, kad būtų galima atpažinti metaduomenis, kurie jau buvo pateikti ir po to atnaujinti.

Šio stulpelio pildyti nereikia.

`type`
: **Tipas**

  Prasmė priklauso nuo dimensijos.

  Jei nenurodytas nei vienas *dimensijos stulpelis*, tuomet šiame stulpelyje nurodoma *papildoma dimensija*.

  **Taip pat žiūrėkite:**

  *Duomenų tipai*

`ref`
: **Ryšys**

  Prasmė priklauso nuo dimensijos.

  **Taip pat žiūrėkite:**

  *Ryšiai tarp modelių*
  *Matavimo vienetai*
  *enum*

`source`
: **Šaltinis**

  Duomenų šaltinio struktūros elementai.

  **Taip pat žiūrėkite:**

  *Duomenų šaltiniai*

`prepare`
: **Formulė**

  Formulė skirta duomenų atrankai, nuasmeninimui, transformavimui, tikrinimui ir pan.

  **Taip pat žiūrėkite:**

  *Formulės*

`level`
: **Brandos lygis**

  Duomenų brandos lygis, atitinka 5 Star Data.

  **Taip pat žiūrėkite:**

  *Brandos lygiai*

`access`
: **Prieiga**

  Duomenų prieigos lygis.

  **Taip pat žiūrėkite:**

  *Prieigos lygiai*

`uri`

**Žodyno atitikmuo**

Sąsaja su išoriniu žodynu.

**Taip pat žiūrėkite:**

*Išoriniai žodynai*

`title`

**Pavadinimas**

Elemento pavadinimas.

`description`

**Aprašymas**

Elemento aprašymas. Galima naudoti Markdown sintaksę.

Visi stulpeliai lentelėje yra neprivalomi. Stulpelių tvarka taip pat nėra svarbi. Pavyzdžiui jei reikia apsirašyti tik globalių modelių struktūrą, nebūtina įtraukti *`dataset`*, *`resource`* ir *`base`* stulpelių. Jei norima apsirašyti tik prefiksus naudojamus *`uri`* lauke, užtenka turėti tik prefiksų aprašymui reikalingus stulpelius.

Įrankiai skaitantys *DSA*, stulpelius, kurių nėra lentelėje turi interpretuoti juos kaip tuščius. Taip pat įrankiai neturėtų tikėtis, kad stulpeliai bus išdėstyti būtent tokia tvarka. Nors įrankių atžvilgiu stulpelių tvarka nėra svarbi, tačiau rekomenduotina išlaikyti vienodą stulpelių tvarką, tam kad lenteles būtų lengviau skaityti.

## 2.3 Dimensijos

Demensijos leidžia vienoje lentelėje sutalpinti kelias skirtingas lenteles turinčias bendrų savybių.

DSA lentelėje turime tokius dimensijų stulpelius:

- *dataset*
- *resource*
- *model*
- *property*

| dataset | resource | model | property | title |
|---|---|---|---|---|
| datasets/gov/rc/ar/ws | | | | Duomenų rinkinys |
| | db | | | Duomenų teikimo paslauga |
| | | **Gyvenviete** | | Esybė |
| | | | pavadinimas | Savybė |

Pavyzdyje aukščiau turime tris lenteles, turinčias vieną bendrą stulpelį `title`.

Daugiamatė lentelė pateikta viršuje, atitiktų tokią vienamatę lentelę:

| type | name | title |
|---|---|---|
| dataset | datasets/gov/rc/ar/ws | Duomenų rinkinys |
| resource | db | Duomenų teikimo paslauga |
| model | **Gyvenviete** | Esybė |
| property | pavadinimas | Savybė |

Kadangi DSA lentelė yra daugiamatė, nurodant stulpelį, jei kalbama apie konkrečios dimensijos stulpelį, reikia nurodyti ir dimensiją, pavyzdžiui `dataset.title` nurodo būtent apie `dataset` diemensijos `title` stulpelį.

### 2.3.1 dataset

Duomenų rinkinys struktūros apraše nurodomas tam, kad būtų galimybė susieti duomenų struktūros elementus su duomenų rinkiniais registruotais duomenų kataloge. Toks susiejimas atliekamas naudojant duomenų rinkinio kodinį pavadinimą.

**Brandos lygis**

***L203: Nestandartiniai kodiniai pavadinimai***
Duomenų rinkinio ar duomenų erdvės kodinis pavadinimas neatitinka reikalavimų keliamų kodiniams pavadinimams.

---

**Pastaba:** Nurodytas duomenų rinkinio ar vardų erdvės kodinis pavadinimas turi būti unikalus tarp visų duomenų struktūros aprašų.

---

**Taip pat žiūrėkite:**

*Vardų erdvės*
*Kodiniai pavadinimai*

`dataset.`**`id`**
Duomenų rinkinio arba duomenų erdvės identifikatorius.

`dataset.`**`type`**
Jei nenurodyta, pagal nutylėjimą naudojama `dataset` reikšmė, kuri nurodo duomenų rinkinio kodinį pavadinimą nurodyta *duomenų kataloge*.

Galimos reikšmės:

**ns**
Vardų erdvė.

**dataset**
Duomenų rinkinys.

---

**Pavyzdys**

| dataset | resource | model | property | type | title |
|---|---|---|---|---|---|
| datasets/gov/rc | | | | ns | Registrų centras |
| datasets/gov/rc/ar | | | | ns | Adresų registras |
| datasets/gov/rc/ar/ws | | | | | Duomenų teikimo paslauga |

`dataset.`**`ref`**
: Duomenų rinkinio identifikatorius duomenų kataloge. Alternatyviai, galima naudoti *`dataset.source`*.

  Nenaudojamas jei *`dataset.type`* yra `ns`.

`dataset.`**`source`**
: Nuoroda į duomenų rinkinio puslapį duomenų kataloge.

  Nenaudojama, jei `dataset.type` yra `ns`.

`dataset.`**`prepare`**
: Nenaudojama.

`dataset.`**`level`**
: Nenaudojamas.

  Duomenų rinkinio brandos lygis yra išskaičiuojamas iš *`model.level`* ir *`property.level`*.

`dataset.`**`access`**
: Prieigos lygis, naudojamas pagal nutylėjimą viesiems šios vardų erdvės elementams.

`dataset.`**`title`**
: Duomenų rinkinio ar vardų erdvės pavadinimas.

`dataset.`**`description`**
: Duomenų rinkinio ar vardų erdvės aprašymas.

### 2.3.2 resource

Fizinis duomenų šaltinis, kuriame saugomi duomenys.

Kiekvienam duomenų šaltiniui suteikiamas kodinis pavadinimas, kuris nėra naudojamas formuojant API URI, tačiau naudojamas identifikuojant patį duomenų šaltinį.

Nurodytas duomenų šaltinio kodinis pavadinimas turi būti unikalus duomenų rinkinio kontekste.

**Taip pat žiūrėkite:**

*Duomenų šaltiniai*

`resource.`**`id`**
: Duomenų šaltinio unikalus identifikatorius UUID formatu.

`resource.`**`type`**
: Duomenų šaltinio tipas. Galimos reikšmės:

| | |
|---|---|
| `sql` | Reliacinės duomenų bazės |
| `csv` | CSV lentelės |
| `json` | JSON resursai |
| `xml` | XML resursai |

`resource.`**`ref`**

Identifikatorius, naudojamas konfiguracijoje, kurioje pateikiamas pilnas resurso adresas ir kiti parametrai, tokie kaip slaptažodžiai ar prisijungimo vardai.

Alternatyviai resurso pilną adresą galima nurodyti *`resource.source`* stulpelyje.

`resource.`**`source`**

Pilnas resurso adresas URI formatu.

> **Įspėjimas:** Jei duomenų šaltinis reikalauja naudotojo vardo ir slaptažodžio, rekomenduojama nerodyti URI struktūros apraše, vietoj to prisijungimo duomenis prie šaltinio pateikti atskirame konfigūraciniame faile, naudojant *`resource.ref`* stulpelį.

**dialect** [ + **driver** ] `://` [ **user** `:` **password** `@` ] **host** [ `:` **port** ] `/` **path** [ `?` **params** ]

**dialect**
: Duomenų šaltinio dialektas arba protokolas, kuriuo teikiami duomenys, galimi variantai:

| | |
|---|---|
| `postgresql` | PostgreSQL duomenų bazė. |
| `mysql` | MySQL duomenų bazė. |
| `mariadb` | MariaDB duomenų bazė. |
| `sqlite` | SQLite duomenų bazė. |
| `oracle` | Oracle duomenų bazė. |
| `mssql` | Microsoft SQL Server duomenų bazė. |
| `http`, `https` | Duomenų failas publikuojamas HTTP protokolu. |

**driver**
: Priklauso nuo **dialect** ir nuo naudojamo duomenų agento.

**user**
: Duomenų šaltinio naudotojo vardas, jei duomenų šaltinis to reikalauja.

**password**
: Duomenų šaltinio slaptažodis, jei duomenų šaltinis to reikalauja.

**host**
: Duomenų šaltinio serverio adresas, jei duomenų šaltinis yra nuotoliniame serveryje.

**port**
: Nuotolinio serverio prievado numeris.

**path**
: Duomenų bazės pavadinimas arba kelias iki duomenų failo.

**params**
: Papildomi parametrai, priklauso nuo naudojamo **driver**.

`resource.`**`level`**
: Duomenų šaltinio *brandos lygis*, vertinant tik pagal formatą, nežiūrint į šaltinyje esančių duomenų turinį.

`resource.`**`access`**
: Duomenų šaltinio *prieigos lygis*.

  Pildyti neprivaloma, jei nurodytas, tada visoms žemesnio lygio dimensijoms, pagal nutylėjimą taikomas nurodytas šaltinio prieigos lygis.

`resource.`**`title`**
: Duomenų šaltinio pavadinimas.

`resource.`**`description`**
: Duomenų šaltinio aprašymas.

### Funkcijos

`resource.prepare.`**`http`**(*method='GET', body=form*)
: Papildomi parametrai, reikaling konstruojant HTTP užklausas.

  **Argumentai**

  **method (vardinis)**
  : HTTP methodas.

  **body (vardinis)**
  : HTTP užklausos perduodamų duomenų formatas.

    Galimi variantai:

| | |
|---|---|
| `json` | Duomenys perduodami JSON formatu. |
| `xml` | Duomenys perduodami XML formatu. |
| `from` | Duomenys perduodami `application/x-www-form-urlencoded` arba `multipart/form-data` (jei formoje pateikiami failai) formatu. |

---

**Pavyzdys**

| resource | type | ref | source | prepare |
|---|---|---|---|---|
| resource1 | json | | https://example.com/ | |
| | param | name1 | NAME1 | `query("value1")` |
| | | name2 | NAME2 | `query("value2")` |

Bus konstruojamas toks URI:

```
https://example.com/?NAME1=value1&NAME2=value2
```

---

### 2.3.3 base

Modelio bazė naudojama objekto identifikatoriams susieti, kai keli skirtingi duomenų modeliai aprašo tą pačią realaus pasaulio esybę.

`base.`**`ref`**

*`model.property`* reikšmė, kurios pagalba *`model`* objektai siejami su *`base`* objektais. Jei susiejimas pagal vieną *`model.property`* yra neįmanomas, galima nurodyti kelis *`model.property`* pavadinimus atskirtus kableliu.

Galima naudoti tik tuos *`model.property`*, kurie neturi nurodyto *`property.type`*, kas reiškia, kad toks pat laukas turi būti tiek *`base`*, tiek *`model`* laukų sąraše.

Tais atvejais, kai *`base.ref`* rodo į modelio lauką, kuris turi tipą, tada *`base.level`* negali būti didesnis nei 3, kadangi jei modelio laukas turi tipą, tai reiškia, kad jo duomenys nesutampa su bazės duomenimis ir todėl jungimas negali būti daromas.

`base.`**`level`**

*Brandos lygis*, nurodantis modelio susiejamumą su nurodytu baziniu modeliu. Plačiau žiūrėti *Ryšiai tarp modelių | Brandos lygis*.

Jei brandos lygis yra žemesnis nei 3, tada identifikatorių siejimas nėra atliekamas, tokiu būdu tiesiog nurodomas semantinis susiejimas metaduomenų, o ne duomenų lygmenyje.

`base.`**`access`**

Nenaudojamas.

#### Išoriniai identifikatoriai

Modelis ir jo bazė turi vienodus išorinius identifikatorius, nors vidiniai šaltinio identifikatoriai gali skirtis.

Siejant *`model`* ir *`base`* duomenis tarpusavyje, *`model`* lentelė įgauna lygiai tokius pačius unikalius identifikatorius, kurie yra *`base`* lentelėje. Tai reiškia, kad *`model`* lentelėje negali būti duomenų, kurių nėra *`base`* lentelėje.

Identifikatorių apjungimas atliekamas pagal *`model.ref`* ir *`base.ref`* pateiktus pirminius raktus, kurie turi sutapti.

Visi *`base.ref`* laukai turi būti aprašyti tiek *`base`*, tiek *`model`* modeliusoe.

#### Paveldimumas

*`model`* paveldi visus laukus iš *`base`*, įskaitant ir tuos, kurie nėra nurodyti prie *`model`* laukų sąrašo. Tai reiškia, kad galima skaityti ir rašyti duomenis į *`base`*, per *`model`*. Jei skaitomas ar rašomas laukas, kurio nėra *`model`* laukų sąraše, tada to lauko duomenys skaitomi iš arba rašomi į *`base`* modelį.

Skaitymas ir rašymas iš base įmanomas tik tuo atveju, jei tai palaiko duomenų šaltinis.

### Duomenų lokalumas

Visi modelio laukai, kurie neturi *property.type*, fiziškai saugomi *base* modelio šaltinyje.

Jei *base* stulpelyje nurodoma / reikšmė, tai reiškia, kad *model* neturi bazės, arba modelio bazė yra panaikinama. / naudojamas tais atvejais, kai norima vieną ar kelis modelius prijungti prie vienos bazės, tačiau sekantys modeliai nebeturi priklausyti jokiai bazei.

### Persidengimas

Tais atvejais, kai *property* yra saugomas tiek *base*, tiek *model* lentelėse, norint gauti persidengiančios savybės duomenis iš *base*, reikia naudoti `_base.` prefiksą.

`_base` rodo į bazinį modelė.

### Pavyzdžiai

**Pavyzdys**

| dataset | base | model | property | type | ref | source |
|---|---|---|---|---|---|---|
| example | | | | | | |
| | | Location | | | id | |
| | | | id | integer | | |
| | | | name@lt | text | | |
| | | | population | integer | | |
| | Loocation | | | | name@lt | |
| | | City | | | name@lt | CITY |
| | | | name@lt | | | NAME |
| | | | population | | | POPULATION |
| | / | | | | | |
| | | Village | | | name@lt | VILLAGE |
| | | | name@lt | | | VILLAGE |
| | | | population | | | POPULATION |
| | | | region | ref | Location | REGION |
| | / | | | | | |

Šiame pavyzdyje esminis skirtumas yra tas, kad nurodyta kaip daromas jungimas. `City` ir `Village` su `Location` jungiame per `name@lt` lauką.

### 2.3.4 model

Duomenų modelio *kodinis pavadinimas*. Užrašomas vienaskaitos forma iš didžiosios raidės, jei pavadinimas iš kelių žodžių, žodžiai atskiriami didžiąja raide.

**Pavyzdžiai**

```
Gyvenviete
AdministracinisTipas
```

Modelis yra siejamas su realaus pasaulio esybėmis. Viena esybė gali turėti kelis skirtingus duomenų modelius, su skirtingomis savybėmis, tačiau skirtingi vienos esybės modeliai turi turėti vienodus identifikatorius.

**Brandos lygis**

***L203: Nestandartiniai kodiniai pavadinimai***
Modelio kodinis pavadinimas neatitinka reikalavimų keliamų kodiniams pavadinimams.

`model.`**`type`**

Pakeista 0.2 versijoje: Nuo 0.2 versijos nurodo modelio bazę.

Nurodo modelio bazę arba esybę, kurios pagalba skirtingiems modeliams suteikiami vienodi identifikatoriai.

Jei nurodyta modelio bazė, *`model.ref`* nurodytas pirminis raktas turi sutapti su bazinio modelio pirminiu raktu.

Taip pat turi sutapti ir modelio savybės su baziniu modeliu. Tačiau modelis gali turėti ir papildomų savybių, kurių nėra baziniame modelyje. Vienintelis privalomas reikalavimas yra pirminio rakto susiejimas, kad modelis ir bazinis modelis turėtu vienodus identifikatorius.

**Brandos lygis**

***L103: Neįmanomas jungimas***
Modelis yra susietas su bazinio registro esybe metaduomenų lygmeniu, tačiau nėra tokio identifikatoriaus kuris leistu susieti ir pačius duomenis.

***L209: Nenurodyta modelio bazė***
Modelis nėra susietas su baziniame registre apibrėžta esybe.

**Pavyzdys**

**Duomenų modelis**

**Struktūros aprašas**

| dataset | model | property | type | ref |
|---|---|---|---|---|
| locn | | | | |
| | **Location** | | | code |
| | | code | integer | |
| | | name@en | string | |
| ns1 | | | | |
| | **Gyvenviete** | | **/locn/Location** | code |
| | | code | integer | |
| | | name@lt | string | |
| ns2 | | | | |
| | **Gyvenviete** | | **/locn/Location** | code |
| | | code | integer | |
| | | name@lt | string | |

Pavyzdyje turime tris modelius iš skirtingų duomenų rinkinių, `ns1:Gyvenviete` ir `ns2:Gyvenviete` nurodo `locn:Location` kaip šių modelių bazę, tai reiškia, kad visi trys modeliai realiame pasaulyje yra viena esybė, turinti vienodus identifikatorius skirtinguose modeliuose.

`model.`**`ref`**

Kableliu atskirtas sąrašas *`model.property`* duomenų laukų pavadinimų, kurie kartu unikaliai identifikuoja vieną duomenų eilutę (pirminis lentelės raktas arba identifikatorius).

Jei nurodytas *`model.type`*, pirminis raktas būtinai turi sutapti su *`model.type`* pirminiu raktu.

Jei modelio objektą unikaliai identifikuoja keli duomenų laukai, *`model.ref`* stulpelyje galima nurodyti kelis duomenų laukus atskirtus kableliu.

**Brandos lygis**

***L003: Nėra identifikatoriaus***
Nenurodytas objekto identifikatorius.

***L104: Identifikatorius nėra unikalus***
Nurodytas objekto identifikatorius nėra unikalus, turi pasikartojančių reikšmių.

***L204: Nepatikimi identifikatoriai***
Nurodytas objekto identifikatorius yra unikalus, tačiau nepatikimas, kadangi nurodytas duomenų laukas, kuris gali keistis, tarkime pavadinimas.

***L301: Nėra globalaus objekto identifikatoriaus***
Nurodytas objekto identifikatorius yra patikimas, tačiau nėra siejamas su globaliu objekto identifikatoriumi.

`model.`**`source`**

Modelio duomenų šaltinis, vieta ar pavadinimas fiziniame duomenų modelyje.

Kas įrašoma į šį stulpelį priklauso nuo duomenų šaltinio *`resource.type`*.

SQL atveju, tai bus lentelės pavadinimas, XML atveju - XPath išraiška, JSON atveju - JSONPath išraiška, skirtingi duomenų šaltiniai gali naudoti skirtingą sintaksę vietai (kur fiziškai saugomi duomenys) apibūdinti.

Jei duomenys publikuojami vidinėje saugykloje, *`model.source`* pildyti nereikia, kadangi vidinės saugyklos fizinio ir loginio modelio pavadinimai yra tokie patys.

**Brandos lygis**

***L004: Duomenų nėra***
Nenurodytas modelio duomenų šaltinis *`model.source`* ir duomenys nėra publikuojami vidinėje saugykloje.

`model.`**`prepare`**

Formulė skirta duomenų filtravimui ir paruošimui, iš dalies priklauso nuo *`resource.type`*.

**Taip pat žiūrėkite:**

*Formulės*
*Duomenų atranka*

`model.`**`level`**

Modelio *brandos lygis*, nusakantis pačio modelio brandos lygį, pavyzdžiui ar nurodytas pirminis raktas, ar modelio pavadinimas atitinka kodiniams pavadinimams keliamus reikalavimus.

**Taip pat žiūrėkite:**

*Brandos lygis*

`model.`**`access`**

Modeliui priklausančių laukų *prieigos lygis*.

Modelio prieigos lygis yra išskaičiuojamas iš modeliui priskirtų duomenų laukų, imant didžiausią prieigos lygmenį nurodytą prie duomenų lauko. Pavyzdžiui, jei bent vienas duomenų laukas turi aukščiausią `open` prieigos lygmenį, tada ir viso modelio prieigos lygis tampa `open`.

**Taip pat žiūrėkite:**

*Prieigos lygiai*

`model.`**`uri`**

Sąsaja su OWL, RDFS ontologijomis ar SKOS kontroliuojamais žodynais.

Jei nenurodyta, generuojamas pavadinimas pagal tokį šabloną:

`https://data.gov.lt/id/` **dataset** `/` **model**

**Pavyzdys**

```
https://data.gov.lt/id/datasets/gov/rc/ar/ws/Location
```

Struktūros apraše galima nurodyti automatiškai generuojamus URI.

**Pavyzdys**

| dataset | model | property | type | ref | uri |
|---|---|---|---|---|---|
| adresai | | | | | |
| | | | prefix | ar | https://data.gov.lt/id/adresai/ |
| | **Location** | | | code | |
| | | code | integer | | ar:code |
| | | name@en | string | | ar:name |
| datasets/gov/ivpk/dp/api | | | | | |
| | | | prefix | ar | https://data.gov.lt/id/adresai/ |
| | **Gyvenviete** | | | code | ar:Location |
| | | code | integer | | ar:code |
| | | name@lt | string | | ar:name |

Šiame pavyzdyje `ar:Location` yra URI, kuris yra automatiškai generuojamas `adresai` duomenų rinkinyje.

**Taip pat žiūrėkite:**

*Išoriniai žodynai*

`model.`**`title`**

Trumpas modelio pavadinimas pirmas žodis iš didžiosios raidės, pavadinimo gale taško nereikia.

Pavadinime nereikia kartoti duomenų rinkinio pavadinimo. Modelio pavadinimas rašomas duomenų rinkinio kontekste.

`model.`**`description`**

Modelio aprašymas.

`model.`**`property`**

Modeliui priklausantis duomenų laukas.

### Funkcijos

`model.prepare.`**`distinct`**`()`

Jei *`model.ref`* pirminis raktas nėra unikalus ir norma panaikinti besidubliuojančias reikšmes, galima nurodyti `distinct()` funkciją, kuri panaidins objktus su besidubliuojančiais pirminiais raktais.

---

**Pavyzdys**

Turint tokius duomenis duomenų šaltinyje:

| CITY | COUNTRY |
|---|---|
| Vilnius | Lithuania |
| Kaunas | Lithuania |

Ir struktūros aprašą, kuriame `COUNTY` aprašytas, kaip atskiras modelis:

| model | property | type | ref | source | prepare | level | access |
|---|---|---|---|---|---|---|---|
| **Country** | | | name@en | CITIES | `distinct()` | 4 | |
| | name@en | string | | COUNTRY | | 4 | open |
| **City** | | | name@en | CITIES | | 4 | |
| | name@en | string | | CITY | | 4 | open |
| | country | ref | **Country** | COUNTRY | | 3 | open |

`distinct()` funkcija panaikina besidubliuojančius objektus ir grąžina tik vieną šalį.

---

### 2.3.5 property

Savybė yra duomenų laukas, modelio atributas.

`property.`**`source`**
: Duomenų lauko pavadinimas šaltinyje. Prasmė priklauso nuo *`resource.type`*.

`property.`**`prepare`**
: Formulė skirta duomenų tikrinimui ir transformavimui arba statinės reikšmės pateikimui.

`property.`**`type`**
: Nurodomas loginis duomenų tipas. Dėl galimų tipų sąrašo žiūrėti *Duomenų tipai*.

  Loginis duomenų tipas yra toks tipas, kurį tikitės gauti publikuojant duomenis per API. Loginis tipas gali skirtis nuo duomenų šaltinio tipo.

  Visi duomenų tipai gali turėti tokius parametrus:

  - `required` - nurodo, kad šis duomenų laukas yra privalomas, tai reiškia, kad šio duomenų lauko reikšmė visada turi būti pateikta. Pagal nutylėjimą visi modelio duomenų laukai yra neprivalomi.

  Kai kurie duomenų tipai, gali turėti konkrečiam duomenų tipui pateikiamus papildomus parametrus, tokie parametrai nurodomi skliausteliuose.

  Dupmenų tipų pavyzdžiai:

  - `integer`
  - `integer required`
  - `geometry`
  - `geometry(linestringm, 3345) required`

`property.`**`ref`**
: Priklauso nuo `property.type`, nurodo matavimo vienetus, laiko ar vietos tikslumą, *klasifikatorių* arba *ryšį su kitais modeliais*. Ką tiksliai reiškia šis laukas, patikslinta skyrelyje *Duomenų tipai*.

`property.`**`level`**
: Nurodo duomenų lauko brandos lygį. Žiūrėti *Brandos lygiai*.

`property.`**`access`**
: Nurodo prieigos prie duomenų lygį. Žiūrėti skyrių *Prieigos lygiai*.

`property.`**`uri`**
: Sąsaja su išoriniu žodynu. Žiūrėti *Išoriniai žodynai*.

`property.`**`title`**
: Duomenų lauko pavadinimas. Šis pavadinimas yra skirtas skaityti žmonėms ir bus rodomas duomenų laukų sąrašuose ir antraštėse. Jei nenurodyta, bus naudojamas *`property`* kodinis pavadinimas.

`property.`**`description`**
: Duomenų lauko aprašymas.

`property.`**`enum`**
: Žiūrėti *enum*.

## 2.4 Papildomos dimensijos

### 2.4.1 prefix

Sąsają su išoriniais žodynais galima pateikti *model.uri* ir *property.uri* stulpeliuose. Tačiau prieš naudojant žodynus, pirmiausia reikia apsirašyti žodynų prefiksus. Žodynų prefiksai aprašomi taip:

`prefix.`**`ref`**
: Prefikso pavadinimas.

  Rekomenduojama naudoti prefix.cc paslaugą URI prefiksų pavadinimams.

`prefix.`**`uri`**
: Išorinio žodyno URI.

`prefix.`**`title`**
: Prefikso antraštė.

`prefix.`**`description`**
: Prefikso aprašymas.

Rekomenduojama naudoti LOV prefiksus.

Aprašyti prefiksai gali būti naudojami *model.uri* ir *property.uri* stulpeliuose tokiu būdu: `prefix:name`.

Pavyzdys:

| d | r | b | m | property | type | ref | uri |
|---|---|---|---|---|---|---|---|
| dataset1 | | | | | | | |
| | | | | | prefix | spinta | https://github.com/atviriduomenys/spinta/issues/ |
| | | | | | | dsa | https://ivpk.github.io/dsa/ |
| | | | | | | dct | http://purl.org/dc/dcmitype/ |
| dataset2 | | | | | | | |
| | | | | | prefix | dcat | http://www.w3.org/ns/dcat# |
| | | | | | | dct | http://purl.org/dc/terms/ |
| | | | | | | dctype | http://purl.org/dc/dcmitype/ |
| | | | | | | foaf | http://xmlns.com/foaf/0.1/ |
| | | | | | | owl | http://www.w3.org/2002/07/owl# |
| | | | | | | prov | http://www.w3.org/ns/prov# |
| | | | | | | rdf | http://www.w3.org/1999/02/22-rdf-syntax-ns# |
| | | | | | | rdfs | http://www.w3.org/2000/01/rdf-schema# |
| | | | | | | sdo | http://schema.org/ |
| | | | | | | skos | http://www.w3.org/2004/02/skos/core# |
| | | | | | | vcard | http://www.w3.org/2006/vcard/ns# |
| | | | | | | xsd | http://www.w3.org/2001/XMLSchema# |

Prefiksai turi būti apibrėžti duomenų rinkinio kontekste, kadangi skirtingi duomenų rinkiniai gali naudoti skirtingus prefiksus, tiems patiems URI. Pavyzdžiui abiejuose rinkinių pavyzdžiuose aukščiau, `dct` iš `dataset1` ir `dctype` iš `dataset2` rodo į tą patį URI.

### 2.4.2 enum

Tam tikri duomenų laukai turi fiksuotą reikšmių variantų aibę. Dažnai duomenų bazėse fiksuotos reikšmės saugomos skaitine forma ar kitais kodiniais pavadinimais. Tokias fiksuotas reikšmes duomenų struktūros apraše galima pateikti neužpildant hierarchinių stulpelių ir nurodant `type` reikšmę `enum`, pavyzdžiui:

| id | d | r | b | m | property | type | ref | source | prepare | level | access | uri | title | description |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | datasets/example/places | | | | | | | | | | | | | |
| 2 | | places | | | | sql | | sqlite:// | | | | | | |
| 3 | | | | Place | | | id | PLACES | | | | | | |
| 4 | | | | | id | integer | | ID | | 3 | open | | | |
| 5 | | | | | type | string | | CODE | | 3 | open | | | |
| 6 | | | | | | enum | | 1 | "city" | | | | City | |
| 7 | | | | | | | | 2 | "town | | | | Town | |
| 8 | | | | | | | | 3 | "village" | | | | Village | |
| 9 | | | | | name | string | | NAME | | 3 | open | | | |

Šiame pavyzdyje `Place.type` laukas yra klasifikatorius, kurio reikšmės yra kodai 1, 2 ir 3, kurios duomenų struktūros apraše keičiamos į `city`, `town` ir `village`, papildomai `title` stulpelyje nurodant reikšmės pavadinimą.

Jei tas pats klasifikatorius gali būti naudojamas keliose skirtingose vietose, tada galima iškelti klasifikatorių ir suteikti jam pavadinimą, pavyzdžiui:

| id | d | r | b | m | property | type | ref | source | prepare | level | access | uri | title | description |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | datasets/example/places | | | | | | | | | | | | | |
| 2 | | | | | | enum | place | 1 | "city" | | | | City | |
| 3 | | | | | | | | 2 | "town | | | | Town | |
| 4 | | | | | | | | 3 | "village" | | | | Village | |
| 5 | | places | | | | sql | | sqlite:// | | | | | | |
| 6 | | | | Place | | | id | PLACES | | | | | | |
| 7 | | | | | id | integer | | ID | | 3 | open | | | |
| 8 | | | | | type | string | place | CODE | | 3 | open | | | |
| 9 | | | | | name | string | | NAME | | 3 | open | | | |

Šiuo atveju, klasifikatoriui buvo suteiktas pavadinimas `place` įrašytas `enum.ref` stulpelyje, 2-oje eilutėje. O `Place.type` laukui, `property.ref` stulpelyje nurodyta, kad šis laukas naudoja vardinį `place` klasifikatorių.

`enum.`**`ref`**

Pasirinkimų sąrašo pavadinimas.

`enum.`**`source`**

Pateikiama originali reikšmė, taip kaip ji saugoma duomenų šaltinyje. Pateiktos reikšmės turi būti unikalios ir negali kartotis.

Jei pageidaujama aprašyti tuščią šaltinio reikšmę, tada *`property.prepare`* celėje reikia nurodyti formulę, kuri tuščią reikšmę pakeičia, į kokią nors kitą. Formulės pavyzdys:

```
swap('', '-')
```

`enum.`**`prepare`**

Pateikiama reikšmė, tokia kuri bus naudojama atveriant duomenis. *`model.prepare`* filtruose taip pat bus naudojama būtent ši reikšmė.

`enum.prepare` reikšmės gali kartotis, tokiu būdu, kelios skirtingos `enum.source` reikšmės bus susietos su viena `enum.prepare` reikšme.

`enum.`**`access`**

Klasifikatoriams galima nurodyti skirtingas prieigos teises, tokiu atveju, naudotojas turintis `open` prieigą matys tik tuos duomenis, kurių klasifikatorių reikšmės turi `open` prieigos teises, visi kiti bus išfiltruoti.

`enum.`**`title`**

Fiksuotos reikšmės pavadinimas.

`enum.`**`description`**

Fiksuotos reikšmės aprašymas.

Pagal nutylėjimą, jei *`property.prepare`* yra tuščias ir *`property`* turi *`enum`* sąrašą, tada jei šaltinis turi neaprašytą reikšmę, turėtų būti fiksuojama klaida.

Jei yra poreikis fiksuoti tik tam tikras reikšmes, o visas kitas palikti tokias, kokios yra šaltinyje, tada *`property.prepare`* stulpelyje reikia įrašyti `self.choose(self)`.

### 2.4.3 param

Parametrai leidžia iškelti tam tikras duomenų paruošimo operacijas į parametrus kurie gali būti naudojami *Dimensijos*, kurioje apibrėžtas parametras kontekste. Parametrai gali gražinti *iteratorius*, kurių pagalba galima dinamiškai kartoti *`resource`* duomenų skaitymą, panaudojant aprašytus parametrus. Taip pat parametrų pagalba galima sudaryti reikšmių sąrašus, kurių pagalba galima kartoti *`resource`* su kiekviena reikšme.

Parametrai dažniausiai naudojami žemesnio brandos lygio duomenų šaltiniams aprašyti, o taip pat API atvejais, kai duomenys atiduodami dinamiškai.

Parametrai aprašomi pasitelkiant papildomą *param* dimensiją.

| d | r | m | property | type | ref | source | prepare |
|---|---|---|---|---|---|---|---|
| datasets/example/cities | | | | | | | |
| | places | | | csv | | https://example.com/{}.c | |
| | | **Country** | | | id | countries | |
| | | | code | string | | CODE | |
| | | | title | string | | TITLE | |
| | | **City** | | | country, title | cities/{code} | |
| | | | | param | code | **Country** | read().code |
| | | | country | ref | **Country** | code | param() |
| | | | title | string | | TITLE | |

`param.`**`ref`**

Parametro *kodinis pavadinimas*.

`param.`**`prepare`**

Formulė, kuri grąžina sąrašą reikšmių aprašomam parametrui.

`param.`**`source`**

Nurodoma parametro reikšmė šaltinyje, kuri yra pateikiama kaip pirmas *`param.prepare`* funkcijos argumentas.

Jei *`param.prepare`* nenurodyta jokia formulė, tada bus naudojam konstanta nurodyta *`param.source`* stulpelyje.

**Pavyzdys**

Jei *param.prepare* pateikta formulė `read()`, o *param.source* nurodyta `Country`, tai formulė bus iškviesta kaip `read("Country")`.

Jei parametro reikšmė yra *iteratorius*, tada *dimensija*, kurios kontekste yra aprašytas *parametras* yra kartojama tiek kartų, kiek reikšmių grąžina *iteratorius*.

Jei yra keli *param* grąžinantys *iteratorius*, tada iš visų *iteratorių* sudaroma Dekarto sandauga ir *resource* dimensija vykdoma su kiekviena sandaugos rezultato reikšme.

Jei sekančioje *DSA* eilutėje, einančioje po eilutės, kurioje aprašytas *param*, nenurodytas *type* ir neužpildytas joks kitas *dimensijos* stulpelis, tada parametras tampa *iteratoriumi*, kurio reikšmių sąrašą sudaro sekančiose eilutėse patektos *source* ir *prepare* reikšmės. Pavyzdžiui anksčiau pateiktą pavyzdį galima būtų perdaryti taip:

| dataset | resource | model | property | type | ref | source | prepare |
|---|---|---|---|---|---|---|---|
| datasets/example/cities | | | | | | | |
| | places | | | csv | | https://example.com/{ | |
| | | **Country** | | | id | countries | |
| | | | code | string | | CODE | |
| | | | title | string | | TITLE | |
| | | **City** | | | country, title | cities/{country} | |
| | | | | param | country | lt | |
| | | | | | | lv | |
| | | | | | | ee | |
| | | | country | ref | **Country** | | `param(country)` |
| | | | title | string | | TITLE | |

Šiame pavyzdyje, parametras `country` grąžins tris šalies kodus: lt, lv ir ee, kurie bus panaudojami `cities/{country}` pavadinime, pakeičiant `{country}` dalį.

*param* reikšmės pasiekiamos naudojant pavadinimą įrašytą *param.ref* stulpelyje. Pavyzdžiui, jei *param.ref* stulpelyje įrašyta `x`, tada `x` parametro reikšmę galima gauti taip:

**source**
: `{x}`.

**prepare**
: `x` arba `param(x)`.

### Funkcijos

Parametrų generavimui galima naudoti tokias formules:

`param.prepare.`**`read`**(*model*)

Sukuriama priklausomybė nuo kito modelio, skaitomi duomenys iš kito modelio ir su kiekvienu objektu, kreipiamasi į *`resource.source`*, panaudojant nuskaitytą objektą kaip parametrą formuojant šaltinio užklausą.

`param.prepare.`**`range`**(*stop*)

Sveikų skaičių generavimas nuo 0 iki `stop`, `stop` neįeina.

`param.prepare.`**`range`**(*start*, *stop*)

Sveikų skaičių generavimas nuo `start` iki `stop`, `stop` neįeina.

`param.prepare.`**`query`**(*name*, *value*)

Parametras pateikia URI query dalies parametrą.

Jei *`resource.source`* jau turi query parametrus, jei bus papildyti.

**Argumentai**

**name**
Nurodo URI query parametro pavadinimą, nurodomas *`param.source`* stulpelyje.

**value**
URI query paramtro reikšmė.

---

**Pavyzdys**

| resource | type | ref | source | prepare |
|---|---|---|---|---|
| resource1 | json | | https://example.com/ | |
| | param | name1 | NAME1 | `query("value1")` |
| | | name2 | NAME2 | `query("value2")` |

Bus konstruojamas toks URI:

```
https://example.com/?NAME1=value1&NAME2=value2
```

---

`param.prepare.`**`header`**(*name*, *value*)

Parametras pateikiamas, kaip HTTP antraštė.

**Argumentai**

**name**
Nurodo HTTP antraštės pavadinimą, nurodomas *`param.source`* stulpelyje.

**value**
HTTP antraštės reikšmė.

---

**Pavyzdys**

| resource | type | ref | source | prepare |
|---|---|---|---|---|
| resource1 | json | | https://example.com/ | |
| | param | name1 | X-Name1 | `header("value1")` |
| | | name2 | X-Name2 | `header("value2")` |

Bus konstruojama tokia HTTP užklausa:

```
GET / HTTP/1.1
X-Name1: value1
X-Name2: value2
```

---

`param.prepare.`**`body`**`(`*name*, *value*, *parent*, *type*`)`

Generuoja XML, JSON ar kito formato dokumentą, kuris pateikiamas HTTP užklausos metu.

**Argumentai**

**name**
JSONPath arba **XPath_** išraiška, priklauso nuoo `resource.prapare` nurodytos *`resource.prepare.http()`* `body` tipo.

Nurodoma *`param.source`* stulpelyje.

**value**
Reikšmė suteikiama `name` elementui http užklausos struktūroje.

**parent (neprivalomas)**
Parametro pavadinimas, kurio pagrindu konstruojamas naujas dokumentas.

**type (neprivalomas, vardinis)**
Naudojamas konstruojant naują dokumentą, jei nurodytas, kiti argumentai turi būti nepateikti.

---

**Pavyzdys (JSON)**

| resource | type | ref | source | prepare |
|---|---|---|---|---|
| resource1 | json | | https://example.com/ | `http(body: json)` |
| | param | name1 | NAME1 | `body("value1")` |
| | | name2 | NAME2 | `body("value2")` |
| | | name3 | NESTED.NAME3 | `body("value3")` |
| | | name4 | ARRAY[].NAME4 | `body("value4")` |
| | | name5 | MATRIX[] | `body()` |
| | | | MATRIX[][] | `body("value5")` |
| | | | MATRIX[] | `body()` |
| | | | MATRIX[][] | `body("value6")` |
| | | | MATRIX[][] | `body("value7")` |

```
{
    "NAME1": "value1",
    "NAME2": "value2",
    "NESTED": {
        "NAME3": "value3",
    },
    "ARRAY": [
        {"NAME4": "value4"}
    ],
    "MATRIX": [
        ["value5"],
        ["value6", "value7"]
    ]
}
```

**Pavyzdys (XML)**

| resource | type | ref | source | prepare |
|---|---|---|---|---|
| resource1 | json | | https://example.com/ | `http(body: xml)` |
| | param | name1 | DATA/@NAME1 | `body("value1")` |
| | | name2 | DATA/NAME2 | `body("value2")` |
| | | name3 | DATA/NESTED/NAME3 | `body("value3")` |
| | | name4 | DATA/ARRAY/NAME4 | `body("value4")` |
| | | name5 | DATA/ARRAY/NAME4 | `body("value5")` |

```xml
<?xml version="1.0" encoding="utf-8"?>
<DATA NAME1="value1">
    <NAME2>value2</NAME2>
    <NESTED>
        <NAME3>value3</NAME3>
    </NESTED>
    <ARRAY>
        <NAME4>value4</NAME4>
        <NAME4>value5</NAME4>
    </ARRAY>
</DATA>
```

**Pavyzdys (maišytas)**

| resource | type | ref | source | prepare |
|---|---|---|---|---|
| resource1 | json | | https://example.com/ | `http(body: xml)` |
| | param | name1 | DATA/@NAME1 | `body("value1")` |
| | | name2 | DATA/NAME2 | `body(name3)` |
| | | name3 | | `body(type: json)` |
| | | name4 | NAME4 | `body("value4", name3)` |
| | | name5 | NAME5 | `body("value5", name3)` |

```xml
<?xml version="1.0" encoding="utf-8"?>
<DATA NAME1="value1">
    <NAME2><![CDATA[
        {
            "NAME4": "value4",
            "NAME5": "value5"
        }
    ]]></NAME2>
</DATA>
```

Jei užpildytas *param.source* stulpelis, tada *param.prepare* stulpelyje galima naudoti filtrą nurodyto *param.source* modelio duomenims filtruoti, o naudojant parametrus galima nurodyti ir modelio laukų pavadinimus, pavyzdžiui:

**source**
: `{x.field}`.

**prepare**
: `x.field` arba `param(x).field`.

### 2.4.4 switch

Tam tikrais atvejais duomenis tenka normalizuoti parenkant tam tikrą reikšmę jei tenkinama nurodyta sąlyga. Tokias situacijas galima aprašyti pasitelkiant *switch* dimensiją.

`switch.`**`source`**
: Reikšmė, kuri bus atveriama.

`switch.`**`prepare`**
: Sąlyga, naudojant einamojo modelio laukus. Jei sąlyga tenkinama, tada laukui priskiriama *switch.source* reikšmė. Jei sąlyga netenkinama, tada bandoma tikrinti sekančią sąlygą. Parenkama ta reikšmė, kurios pirmoji sąlyga tenkinama.

  Jei *switch.prepare* yra tuščias, tada sąlyga visada teigiama ir visada grąžinama *switch.source* reikšmė.

### 2.4.5 comment

Dirbant su *DSA* yra galimybė komentuoti eilutes, naudojant papildomą *comment* dimensiją, kurią galima naudoti bet kurios kitos dimensijos kontekste.

`comment.`**`id`**
: Komentaro numeris.

`comment.`**`ref`**
: Komentuojamo vieno ar kelių kableliu atskirtų *property* pavadinimai. Galima nurodyti ne tik stulpelio pavadinimą, bet ir dimensiją.

`comment.`**`source`**
: Komentaro autorius.

`comment.`**`prepare`**
: Keitimo pasiūlymas, naudojant `create()`, `update` ir `delete()` funkcijas. Pavyzdžiui:

```
update(property: "pavadinimas@lt", type: "text")
```

  Šiuo atveju nurodoma, kad siūloma keisti `property` pavadnimą į `pavadinimas@lt`, o `type` į `text`.

`comment.`**`level`**
: Nurodoma, kad patenkinus keitimo siūlymą, kuris nurodytas *comment.prepare* stulplyje, komentuojamai eilutei gali būti suteiktas nurodytas brandos lygis.

`comment.`**`access`**
: Nurodoma, ar komentaras gali būti publikuojamas viešai.

  **private**
  : Komentaras negali būti publikuojamas viešai. Šis prieigos lygis naudojamas pagal nutylėjimą.

  **open**
  : Komentaras gali būti publikuojamas viešai.

`comment.`**`uri`**

Viena ar kelios kableliu atskirtos šaltinio nuorodos, kurios pateikia daugiau informacijos apie tai, kas komentuojama. Taip pat gali būti nurodytas kito komentaro *`comment.id`*, nurodant, kad tai yra atsakymas į ankstesnį komentarą.

URI pateikiami sutrumpinta forma, naudojant prefikstus. Žiūrėti skrių *Išoriniai žodynai*.

`comment.`**`title`**

Komentaro data, ISO 8601 formatu.

`comment.`**`description`**

Komentaro tekstas.

**Pavyzdys**

| d | r | b | m | property | type | ref | prepare | level | access | uri |
|---|---|---|---|---|---|---|---|---|---|---|
| example | | | | | | | | | | |
| | | | | | prefix | spinta | | | | https://github.com/atviriduomenys |
| | | | | | | dsa | | | | https://ivpk.github.io/dsa/ |
| | | | Imone | | | | | 2 | | |
| | | | | | comment | base | update(base: "/jar/JuridinisAsmuo ref: "id") | 4 | open | spinta:205, manifest:1290 |
| | | | | | comment | ref | update(ref: "id") | 4 | open | vadovas:dsa/dimensijos.html#model.re |
| | | | | id | integer | | | 4 | open | |
| | | | | pavadinimas | string | | | 2 | open | |
| | | | | | comment | ref | update(property: "pavadinimas@lt", type: "text") | 4 | open | spinta:204 |

## 2.4.6 lang

Nebepalaikoma nuo 0.2 versijos.

*`title`* ir *`description`* stulpeliuose tekstas rašomas lietuvių kalba, tačiau galima pateikti tekstą ir kita kalba, panaudojus papildomą *`lang`* dimensiją, kurią reikia naudoti prieš eilutę, kuriai pateikiamas tekstas kita kalba.

`lang.`**`ref`**

ISO 639-1 dviejų simbolių kalbos kodas.

`lang.`**`title`**
: Pavadinimas *`lang.ref`* stulpelyje nurodyta kalba.

`lang.`**`description`**
: Aprašymas *`lang.ref`* stulpelyje nurodyta kalba.

### 2.4.7 migrate

Nebepalaikoma nuo 0.2 versijos.

Laikui einant, pirminių duomenų šaltinių arba jau atvertų duomenų struktūra keičiasi, papildoma naujais *modeliais* ar *savybėmis*, keliant duomenų brandos lygį seni duomenys keičiami naujais, aukštesnio brandos lygio duomenimis.

Visi šie struktūros ar pačių duomenų pasikeitimai fiksuojami papildomos *`migrate`* dimensijos pagalba, kuri gali būti naudojama, bet kurios kitos dimensijos kontekste.

---

**Pastaba:** Migracijos naudojamos tik tuo atveju, kai keičiasi duomenų struktūra arba patys duomenys. Jei keičiasi tik metaduomenys, tai migracijų sąraše neatsispindi.

---

| id | d | r | b | m | property | type | ref | prepare | level | title | description |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | | | | | migrate | | | | 2021-12-21 16:29 | Pirmoji migracija. |
| 2 | | | | | | migrate | 1 | | | 2021-12-21 16:33 | Antroji migracija. |
| 3 | | | | | | migrate | 2 | | | 2022-06-21 16:41 | Trečioji migracija. |
| | datasets/example/migrate | | | | | | | | | | |
| | | | | Country | | | id | | | | |
| | | | | | id | integer | | | 4 | | |
| | | | | | code | string | | | 3 | | |
| | | | | | | migrate | 1 | create(level: 2 | | | |
| | | | | | | migrate | 3 | update(level: 3) | | | |
| | | | | | name | string | | | | | |
| | | | | | | migrate | 2 | create() | | | |

Pavyzdyje aukščiau matome, kad šis duomenų struktūros aprašas turi tris migracijas:

1. Pirmosios migracijos metu sukuriamas pradinis duomenų struktūros variantas. Pirmoji migracija nežymima prie modelių ir duomenų laukų, nebent daromas keitimas, tuomet

įtraukiam ir pirmoji migracija, kad būtų matoma, kas keitėsi. Būtent toks atvejis parodytas prie `Country.code` lauko, kuri trečiojo migracijoje keičiamas brandos lygis.

2. Antrosios migracijos metu buvo įtrauktas naujas duomenų laukas `Country.name`.
3. Trečiosios migracijos metu, buvo keičiami `Country.code` lauko duomenys, pakeitimo metu brandos lygis buvo pakeltas iki trečio. Atkreipkite dėmesį, kad metaduomenų pasikeitimas, kaip šiuo atveju, žymimas migracijose tik tuo atveju, jei tai yra susiję su pačių duomenų pasikeitimu.

   Jei brandos lygis būtų pakeistas, nekeičiant pačių duomenų, tuomet tokio pakeitimo nereikėtų įtraukti į migracijų sąrašą.

   Kadangi trečiojoje migracijoje buvo atliktas su ankstesne versija nesuderinamas pakeitimas, tai šios migracijos data yra 6 mėnesiai ateityje, kadangi nesuderinamos migracijos pirmiausia paskelbiamos, o įgyvendinamos tik praėjus 6 mėnesiams nuo paskelbimo.

`migrate.`**`id`**

Migracijos numeris (UUID). Kiekvienos migracijos metu gali būti atliekama eilė operacijų, visos operacijos fiksuojamos naudojant migracijos numerį.

Visų migracijų sąrašas pateikiamas, kai *`migrate`* nepriklauso jokiam dimensijos kontekstui.

`migrate.`**`ref`**

Ankstesnės migracijos numeris, pateiktas *`migrate.id`* stulpelyje, arba tuščia, jei prieš tai jokių kitų migracijų nebuvo.

Naudojamas jei *`migrate`* nepatenka į jokios dimensijos kontekstą.

Jei *`migrate`* aprašomas dimensijos kontekste, tada šis stulpelis nenaudojamas.

`migrate.`**`prepare`**

Migracijos operacija. Galimos tokios operacijos:

`migrate.`**`create`**`()`

Priklausomai nuo dimensijos konteksto, prideda naują modelį, arba savybę.

Funkcijai galima perduoti `ref` ir kitus vardinius argumentus, kurie atitinka *DSA* lentelės metaduomenų stulpelių pavadinimus.

`migrate.`**`update`**`()`

Taikomas tik duomenų laukams ir nurodo, kad buvo pakeistos esamų duomenų reikšmės, keičiant reikšmių dimensiją, matavimo vienetus, formatą ir kita.

Funkcijai galima perduoti `ref` ir kitus vardinius argumentus, kurie atitinka *DSA* lentelės metaduomenų stulpelių pavadinimus.

Perduodami tik tie vardiniai argumentai, kuriuos atitinkantys metaduomenys keičiasi.

`migrate.`**`delete`**`()`

Priklausomai nuo dimensijos konteksto, šalina modelį ar savybę.

Pašalinto modelio ar savybės *`type`* keičiamas į `absent` reikšmę.

`migrate.`**`filter`**`(`*`where`*`)`

Naudojamas *`property`* kontekste, kai vykdoma duomenų migracija. Nurodo, kad migracija taikoma tik `where` sąlygą tenkinantiems duomenims.

Be šių pagrindinių migracijos operacijų, galima naudoti kitas duomenų transformavimo operacijas, kurios vykdomos su kiekviena duomenų eilute ir atlikus pateiktas transformacijos funkcijas, pakeista reikšmė išsaugoma.

`migrate.`**`title`**

Migracijos įvykdymo data ir laikas. Migracijos laikas ir data gali būti ir ateityje, tuo atveju, jei daromas nesuderinamas keitimas.

Naudojamas tik tada, kai *`migrate`* nepatenka į jokios dimensijos kontekstą.

`migrate.`**`description`**

Migracijos atliekamo pakeitimo trumpas aprašymas.

## 2.5 Duomenų tipai

`type.`**`absent`**

Nebepalaikoma nuo 0.2 versijos: Šis tipas buvo naudojamas migracijoms ir versijavimui, tačiau nuo 0.2 versijos, versijavimo struktūros aprašuose atsisakyta.

Žymi *savybę*, kuri buvo ištrinta ir nebenaudojama.

**Taip pat žiūrėkite:**

*migrate*

`type.`**`boolean`**

Loginė reikšmė, pateikiama skaitine forma:

| | |
|---|---|
| `0` | Neigiama reikšmė (*false*). |
| `1` | Teigiama reikšmė (*true*). |

**Brandos lygis**

***L102: Nėra vientisumo***

Duomenyse nėra vientisumo, kartais `true` pateikta kaip `1`, kartais kaip `taip`, arba `yes`.

***L202: Nestandartinis formatas***

Visi duomenys pateikti vienoda, tačiau nestandartine forma. Tarkime duomenys pateikti `true`, `false` reikšmėmis, tačiau turėtu būti `1`, `0`.

`type.`**`integer`**

Sveikas skaičius.

*`property.ref`* stulpelyje, nurodomi *Matavimo vienetai*.

**Brandos lygis**

***L102: Nėra vientisumo***
Duomenys pateikti skirtingais vienetais, pavyzdžui dalis duomenų pateikti metrais, dalis kilometrais ir dalis milimetrais.

***L105: Vienetų konvertavimo paklaida***
Amžius pateiktas atskirai, nurodant metus, menesius ir dienas. Šiuo atveju brandos lygis yra 1, kadangi neaišku, kiek yra dienų metuose ir mėnesiuose, kadangi skirtingi metai ir skirtingi mėnesiai turi skirtingą dienų skaičių. Tokiais atvejais duomenis reikia pateikti dienomis.

***L202: Nestandartinis formatas***
Duomenys pateikti išskaidant vieną reikšmę į kelias reikšmes, skirtingais vienetais. Duomenys turėtu būti pateikti mažiausiu detalumu, viename duomenų lauke. Pavyzdžiui atstumas pateiktas atskirais duomenų laukais, kur viename nurodomas atstumas kilometrais, kitame metrais, trečiame milimetrais. Šiuo atveju, duomenys turėtu būti pateikiami milimetrais.

***L302: Nenurodyti matavimo vienetai***
Duomenys yra kiekybiniai, tačiau *property.ref* stulpelyje nenurodyti vienetai.

`type.`**`number`**
Realusis skaičius, apvalinamas naudojant slankiojo kablelio aritmetiką.

*property.ref* stulpelyje, nurodomi *Matavimo vienetai*.

Sveikoji dalis atskiriama `.` simbolių.

`type.`**`binary`**
Dvejetainiai duomenys. Bendras baitų skaičius turi būti ne didesnis nei 1G.

Jei reikšmė yra didesnė nei 1G reikųtu naudoti *type.file*.

`type.`**`string`**
Simbolių eilutė. Neriboto dydžio, tačiau fiziškai simbolių eilutė turėtu būti ne didesnė, nei 1G.

Simboliu eilutė turėtu būti pateikta UTF-8 koduote.

Šiuo tipu žymimi duomenų laukai, kuriuose tekstas pateiktas ne žmonių kalba. Tai gali būti įvairūs kategoriniai duomenys, identifikatoriai ar kito pobūdžio simbolių eilutės, kurios nėra užrašytos natūraliąja žmonių kalba.

Jei *property* pavadinimas turi kalbos žymę `@`, tada `string` tipas tampa `text` tipo dalimi. Kablos kodas nurodomas naudojant ISO 639-1 kodų sąrašą.

Jei tekstas turi kalbos žyme, *property.ref* galima pateikti teksto formatą, nadojant vieną iš šių formatų:

| | |
|---|---|
| `html` | tekstas pateiktas HTML formatu. |
| `md` | tekstas pateiktas Markdown formatu. |
| `rst` | tekstas pateitkas reStructuredText formatu. |
| `tei` | tekstas pateiktas TEI formatu. |

**Pavyzdys**

| d | r | b | m | property | type | ref |
|---|---|---|---|---|---|---|
| example | | | | | | |
| | | | Country | | | |
| | | | | name@lt | string | |
| | | | | description@lt | string | html |
| | | | | description@en | string | html |

Šiame pavyzdyje `@lt` nurodo, kad šalies pavadinimai ir aprašymai pateikti Lietuvių kalba, tačiau laukas `description` papildomai turi vertimą į anglų kalbą. Papildomai, šalies aprašymo teksto formatas yra HTML tipo.

---

**Brandos lygis**

***L202: Nestandartinis formatas***
Tekstas yra užrašytas natūralia žmonių kalba, tačiau neturi kalbos žymės.

***L202: Nestandartinis formatas***
Duomenys pateikti nestandartine koduote. Standartinė koduotė yra UTF-8.

***L202: Nestandartinis formatas***
Duomenys pateikti UTF-8 koduote, tačiau pats tekstas naudoja tam tikrą formatavimo sintaksę, kuri nėra nurodyta *`property.ref`* stulpelyje.

`type.`**`text`**

Natūraliaja žmonių kalba užrašytas tekstas, susidedantis iš vieno ar kelių `string` tipo duomenų laukų, pateikiant atskirą duomenų lauką, kiekvienai kalbai.

Dažniausiai tiesiogiai `text` tipas nenaudojamas, kadangi jei `string` tipas turi kalbos žymę, tai duomenų laukas yra interpretuojamas kaip `text` tipo.

Pavyzdžiui jei *`property`* pavadinimas yra `title@lt`, tada `title` duomenų laukas yra `text` tipo.

Atskirai `text` tipo duomenų laukas gali būti nurodomas tais atvejais, kai reikia pateikti aprašymą ir *URI* pačiam `text` tipo laukui, o ne vienam iš vertimų.

---

**Pavyzdys**

| dataset | model | property | type | ref | uri | title |
|---|---|---|---|---|---|---|
| example | | | | | | |
| | | | prefix | rdfs | http://www.w3.org/ 2000/01/rdf-schema# | |
| | Country | | | name@lt | | |
| | | name | text | | rdfs:label | Pavadinimas |
| | | name@lt | string | | | |

`type.`**`datetime`**

Data ir laikas atitinkantis ISO 8601.

Mažiausia galima reikšmė: `0001-01-01T00:00:00`.

Didžiausia galima reikšmė: `9999-12-31T23:59:59.999999`.

Pagal ISO 8601 standartą, data gali būti pateikta tokia forma:

```
YYYY-MM-DD[*HH[:MM[:SS[.fff[fff]]]][+HH:MM[:SS[.ffffff]]]]
```

Simbolis * reiškia, kad galima pateikti bet kokį vieną simbolį, dažniausiai naudojamas tarpo simbolis, arba raidė `T`.

*`property.ref`* stulpelyje, nurodomas datos ir laiko tikslumas sekundėmis. Tikslumą galima nurodyti laiko vienetais, pavyzdžiui `Y`, `D`, `S`, arba `5Y`, `10D`, `30S`. Visi duomenys turi atitikti vienodą tikslumą, tikslumas negali varijuoti. Galimi vienetų variantai:

| Reikšmė | Prasmė |
|---|---|
| `Y` | Metai |
| `M` | Mėnesiai |
| `Q` | Metų ketvirčiai |
| `W` | Savaitės |
| `D` | Dienos |
| `H` | Valandos |
| `T` | Minutės |
| `S` | Sekundės |
| `L` | Milisekundės |
| `U` | Mikrosekundės |
| `N` | Nanosekundės |

**Brandos lygis**

***L101: Neaiški struktūra***

Data ir laikas pateikti laisvu tekstu, pavyzdžiui `2020 paskutinę pirmo mėnesio dieną`.

***L102: Nėra vientisumo***
Data ir laikas pateikti naudojant skirtingus formatus, pavyzdžiui `2020-01-31`, `01/31/2020`, `31.1.20`.

***L202: Nestandartinis formatas***
Duomenys pateikti nestandartiniu formatu, tačiau visi duomenys pateikti vienodu formatu. Pavyzdžiui visi duomenys pateikti `01/31/2020` formatu, tačiau datos turi būti pateiktos ISO 8601 formatu.

***L210: Išskaidyta atskirais komponentais***
Duomenys pateikti atskiruose laukuose, pavyzdžiui metai pateikti viename `integer` tipo lauke, o ketvirtis, kitame `integer` tipo lauke. Norint didesnio brandos lygio, duomenys turi būti viename `date` tipo lauke su *`property.ref`* = `Q`.

***L303: Nenurodytas duomenų tikslumas***
Nenurodytas *`property.ref`*, kuriame turėtu būti pateiktas duomenų tikslumas.

`type.`**`date`**

Tas pats kas `datetime` tik dienos tikslumu. Šio tipo reikšmės taip pat turi atitikti ISO 8601:

```
YYYY-MM-DD
```

*`property.ref`* stulpeyje nurodomas datos tikslumas:

| Reikšmė | Prasmė |
|---|---|
| Y | Metai |
| M | Mėnesiai |
| Q | Metų ketvirčiai |
| W | Savaitės |
| D | Dienos |

Jei duomenys pateikti žemesniu nei dienos tikslumu, tada datos rekšmės turi būti nurodytos `YYYY-MM-DD` formatu, pakeičiant `MM` ir arba `DD~ į `01`.

**Pavyzdys**

Turint tokį struktūros aprašą:

| model | property | type | ref |
|---|---|---|---|
| Country | | | id |
| | id | integer | |
| | independence | date | Y |

Nors `independence` duomenų lauko tiksluas yra metų, tačiau pateikiant duomenis būtina nurodyti mėnesį ir dieną taip pat:

```
{
    "id": 1,
    "independence": "1990-01-01",
}
```

Šiuo atveju, kadangi datos tiksluas yra metai, `-01-01` dalis datoje neturi jokios reikšmės ir yra pateikiama tik tam, kad reikšmė atitiktu ISO 8601 reikalavimus.

---

`type.`**`time`**

Dienos laikas, be konkrečios datos. Šio tipo reikšmės, kaip ir kiti su laiku susiję tipai turi atitikti ISO 8601:

```
HH[:MM[:SS[.fff[fff]]]][+HH:MM[:SS[.ffffff]]]
```

Jei norima nurodyti žemesnio nei sekundžių tikslumo laiką, tada vietoj minučių ir/ar sekundžių galima naudoti `00` ir *`property.ref`* stulpelyje nurodyti tikslumą:

| Reikšmė | Prasmė |
|---|---|
| `H` | Valandos |
| `T` | Minutės |
| `S` | Sekundės |
| `L` | Milisekundės |
| `U` | Mikrosekundės |
| `N` | Nanosekundės |

`type.`**`temporal`**

Nebepalaikoma nuo 0.2 versijos.

Apibrėžtis laike.

Šis tipas atitinka `datetime`, tačiau nurodo, kad visas model yra apibrėžtas laike, būtent pagal šią savybę. Tik viena model savybė gali turėti `temporal` tipą. Pagal šios savybės reikšmes apskaičiuojamas ir įvertinamas dct:temporal.

`type.`**`geometry`**

Erdviniai duomenys. Duomenys pateikiami WKT formatu, naudojant EPSG duomenų bazės parametrus, skirtingoms projekcijoms išreikšti.

*`property.ref`* stulpelyje nurodomas tikslumas metrais. Tikslumą galima pateikti naudojanti SI vienetus, pavyzdžiui `m`, `km` arba `10m`, `100km`.

`geometry` tipas gali turėti du argumentus `geometry(form, crs)`:

- `form` - geometrijos forma
- `crs` - koordinačių sistema

Pats tipas gali būti pateiktas vienu iš šių variantų:

- `geometry(form, crs)` - nurodant formą ir koordinačių sistemą
- `geometry(crs)` - nurodant tik koordinačių sistemą
- `geometry(form)` - nurodant tik formą
- `geometry` - be argumentų.

**Geometrijos forma** (`form`)

Galimi tokie geometrijos tipai:

- `point` - taškas.
- `linestring` - linija.
- `polygon` - daugiakampis (pradžios ir pabaigos taškai **turi** sutapti).
- `multipoint` - keli taškai.
- `multilinestring` - kelios linijos.
- `multipolygon` - keli daugiakampiai (kiekvieno daugiakampio pradžios ir pabaigos taškai **turi** sutapti).

Kiekviena iš formų gali turėti tokias galūnes nurodančias papildomą dimensiją:

- `z` - aukštis.
- `m` - pasirinktas matmuo (pavyzdžiui laikas, atstumas, storis ir pan.)
- `zm` - aukštis ir pasirinktas matmuo.

Jei geometrijos forma nenurodyta, tada duomenys gali būti bet kokios geometrinės formos. Jei forma nurodyta, tada visi duomenys turi būti tik tokios formos, kokia nurodyta.

**Koordinačių sistema** (`crs`)

Antrasis `geometry` argumentas nurodomas pateikiant SRID numerį, kuris yra konkrečios koordinačių sistemos identifikacinis numeris EPSG duomenų bazėje. Jei koordinačių sistemos numeris nenurodytas, tuomet daroma prielaida, kad erdviniai duomenys atitinka 4326 (**WGS84_**) koordinačių sistemą.

Svarbu, kad pateikiant duomenis, koordinačių ašių eiliškumas atitiktų tokį eiliškumą, kuris nurodytas EPSG parametrų duomenų bazėje, konkrečiai koordinačių sistemai, kuria pateikiami duomenys.

Svarbu, kad pateikiant duomenis, koordinačių ašių eiliškumas atitiktų tokį eiliškumą, kuris nurodytas EPSG parametrų duomenų bazėje, konkrečiai koordinačių sistemai, kuria pateikiami duomenys.

---

**Pavyzdys**

a) pateikiant duomenis LKS 94 (SRID:3346) ir WGS84 (SRID:4326) koordinačių sistemose į ADP Saugyklą turi būti laikomasi eiliškumo: pirmiausia pateikiama X (į šiaurę/platumos/latitude), o po to Y (į rytus/ilgumos/longitute) reikšmės;

b) tačiau pateikiant duomenis WGS84/Pseudo-Merkator (SRID:3857) koordinačių sistemoje jau atvirkščiai – pirmiausia pateikiama rytų ilgumos, o po to šiaurės platumos reikšmės.

---

Pilną SRID kodų sąrašą galite rasti epsg.io svetainėje. Keletas dažniau naudojamų SRID kodų:

| SRID | CRS | Pavyzdys | ašys | orientacija |
|---|---|---|---|---|
| 4326 | WGS84 | POINT(54.6981 25.2738) | lat, lon | north, east |
| 3346 | LKS94 | POINT(6063156 582111) | north, east (x, y) | north, east |
| 3857 | WGS84 / Pseudo-Merctor | POINT(2813472 7303494) | lon, lat | east, north |
| 4258 | ETRS89 | POINT(54.6981 25.2738) | lat, lon | north, east |

**Pastaba:** Atkreipkite dėmesį, kad LKS94 koordinačių sistemoje geometrinės ašys neatitinka įprastinio ašių eiliškumo naudojamo GIS sistemose. LKS94 pirmas skaičius yra apytiksliai 6,000,000 metrų nuo pusiaujo į šiaurę, o antrasis skaičius apytiksliai 500,000 metrų į rytus, skaičiuojant nuo 24º rytų meridiano, atėmus 500km. Teikiant duomenis, taškai turėtu atrodyti taip: `6000000 500000`, pirmas ilgesnis, antras trumpesnis.

> Ašinio meridiano projekcija yra abscisių (x) ašis. Šios ašies teigiamoji kryptis nukreipta į šiaurę. Ordinačių (y) ašies teigiamoji kryptis nukreipta į rytus.
>
> Išvyniojus cilindrą, gaunamos stačiakampės koordinatės su x šiaurinės abscisės pradžia pusiaujuje ir y rytinės ordinatės reikšme 24°C meridiane 500 000 metrų.
>
> ---https://www.e-tar.lt/portal/lt/legalAct/TAR.6D575923F94A

Prieš publikuojant duomenis, galite pasitikrinti, ar koordinačių ašys pateikiamos teisinga tvarka, naudotami taško atvaizdavimo įrankį.

Pavyzdžiui, norint patikrinti Vilniaus Katedros varpinės bokšto taško koordinates, LKS94 (EPSG:3346) sistemoje, galite naršyklės adreso juostoje pateikti šį adresą:

https://get.data.gov.lt/_srid/3346/6061789/582964

Jei ašių eiliškumas teisingas, gausite tašką ten kur tikėjotės, jei ašys sukeistos vietomis, tada taškas žemėlapyje gali būti visai kitoje vietoje, nei tikėjotės.

Adreso formatas:

```
/_srid/{srid}/{ašis1}/{ašis2}
```

- `{srid}` - EPSG duomenų bazėje esančios koordinačių sistemos SRID kodas
- `{ašis1}` - pirmosios ašies reikšmė (kryptis priklauso nuo `{srid}`)
- `{ašis2}` - antrosios ašies reikšmė (kryptis priklauso nuo `{srid}`)

**Pavyzdinės `property.type` reikšmės**

- `geometry` - WGS84 projekcijos, bet kokio tipo geometriniai objektai.
- `geometry(3346)` - LKS94 projekcijos, bet kokio tipo geometriniai objektai.
- `geometry(point)` - GWS84 projekcijos, bet `point` tipo geometriniai objektai.

- `geometry(linestringm, 3345)` - LKS94 projekcijos, `linestringm` tipo geometriniai objektai su pasirinktu matmeniu, kaip trečia dimensija.

---

**Pavyzdys (duomenys)**

Vilniaus Katedros varpinės bokšto taškas, LKS94 (EPSG:3346) koordinačių sistemoje:

```
{
    "koordinates": "POINT (6061789 582964)"
}
```

---

**Brandos lygis**

***L101: Neaiški struktūra***
Pateiktas adresas, nenurodant adreso koordinačių.

***L102: Nėra vientisumo***
Nenurodytas koordinačių sistema ir duomenys pateikti skirtingomis koordinatėmis.

***L102: Nėra vientisumo***
Sumaišytos ašys, pavyzdžiui vieni duomenys pateikiami x, y, kiti y, x.

***L102: Nėra vientisumo***
Sumaišyti vienetai, pavyzdžiui vieni duomenys pateikti metrais, kiti laipsniais.

***L201: Nestandartiniai duomenų tipai***
Nenurodyta koordinačių sistema, tačiau visi duomenys pateikti naudojant vienodą koordinačių sistemą.

***L210: Išskaidyta atskirais komponentais***
Taško koordinatės pateiktos, kaip du atskiri duomenų laukai.

***L303: Nenurodytas duomenų tikslumas***
Nenurodytas *`property.ref`*, kuriame turėtu būti pateiktas duomenų tikslumas metrais.

`type.`**`spatial`**

Nebepalaikoma nuo 0.2 versijos.

Apibrėžtis erdvėje.

Šis tipas atitinka `geometry`, tačiau nurodo, kad visas model yra apibrėžtas erdvėje, būtent pagal šią savybę. Tik viena model savybė gali turėti `spatial` tipą. Pagal šios savybės reikšmes apskaičiuojamas ir įvertinamas dct:spatial.

`type.`**`money`**

Valiuta. Saugomas valiutos kiekis, nurodant tiek sumą, tiek valiutos kodą naudojant ISO 4217 kodus.

Valiutos kodas nurodomas *`property.ref`* stulpelyje.

Pavyzdys:

| d | r | b | m | property | type | ref | source |
|---|---|---|---|---|---|---|---|
| example | | | | | | | |
| | | | Product | | | | PRODUCT |
| | | | | price | money | EUR | PRICE |

Jei valiutos suma ir pavadinimas saugomi atskirai, tuomet valiutą galima aprašyti taip:

| d | r | b | m | property | type | ref | source | prepare |
|---|---|---|---|---|---|---|---|---|
| example | | | | | | | | |
| | | | Product | | | | PRODUCT | |
| | | | | amount | | | PRICE | |
| | | | | currency | | | CURRENCY_CODE | |
| | | | | price | money | | | money(amount, currency) |

Šio tipo duomenys pateikiami viena iš šių formų:

```
123
123.45
123 EUR
123.45 EUR
```

`type.`**`file`**

Šis duomenų tipas yra sudėtinis, susidedantis iš tokių duomenų:

**id**
: Laukas, kuris unikaliai identifikuoja failą, šis laukas duomenų saugojimo metu pavirs failo identifikatoriumi, jam suteikiant unikalų UUID.

**name**
: Failo pavadinimas.

**type**
: Failo media tipas.

**size**
: Failo turinio dydis baitais.

**content**
: Failo turinys.

Šiuos metaduomenis galima perduoti `file()` funkcijai, kaip vardinius argumentus.

---

**Pavyzdys**

| d | r | b | m | property | type | source | prepare | access |
|---|---|---|---|---|---|---|---|---|
| datasets/example | | | | | | | | |
| | | | Country | | | | | |
| | | | | name | string | NAME | | open |
| | | | | flag_file_ | string | FLAG_FILE | | private |
| | | | | flag_file_ | binary | FLAG_FILE | | private |
| | | | | flag | file | | file(name: flag_file_name, content: flag_file_data) | open |

Šiame pavyzdyje, iš `flag_file_name` ir `flag_file_data` laukų padaromas vienas `flag` laukas, kuriame panaudojami duomenys iš dviejų laukų. Šiuo atveju, `flag_file_name` ir `flag_file_data` laukai tampa pertekliniais, todėl *`access`* stulpelyje jie pažymėti `private`.

Analogiškai, tokius pačius duomenis galima aprašyti ir nenaudojant formulių:

| d | r | b | m | property | type | source | prepare | access |
|---|---|---|---|---|---|---|---|---|
| datasets/example | | | | | | | | |
| | | | Country | | | | | |
| | | | | name | string | NAME | | open |
| | | | | flag | file | | | open |
| | | | | flag._name | | FLAG_FILE_NAME | | open |
| | | | | flag._content | | FLAG_FILE_DATA | | open |

`type.`**`image`**

Paveiksliukas. `image` tipas turi tokias pačias savybes kaip `file` tipas.

`type.`**`ref`**

Ryšys su modeliu. Šis tipas naudojamas norint pažymėti, kad lauko reikšmė yra *`property.ref`* stulpelyje nurodyto modelio objektas.

Pagal nutylėjimą, jungimas su kito modelio objektais daromas per siejamo pirminį raktą (*`model.ref`*), tačiau yra galimybė nurodyti ir kitą, nebūtinai pirminį raktą.

Jei jungimas daromas, ne per pirminį raktą, tuomet, laukai per kuriuos daromas jungimas nurodomi *`property.ref`* stulpelyje laužtiniuose sklaustuose, pavyzdžiui:

```
Country[code]
```

Čia jungiama su `Country` modeliu, per `Country` modelio `code` duomenų lauką.

Jei laukas, per kurį daromas jungimas nenurodytas, pavyzdžiui:

```
Country
```

Tada, jungimas daromas per `Country` modelio pirminį raktą, kuris nurodytas *`model.ref`* stulpelyje.

Šio objekto reikšmės yra pateikiamos, kaip dalis objekto į kurį rodoma. Jei `ref` tipo lauko brandos lygis (*`property.level`*) yra 4 ar didesnis, tuomet šio duomenų tipo reikšmės atrodo taip:

```
{"_id": "69c98b0f-9e4e-424b-9575-9f601d79b68e"}
```

Jei brandos lygis (*`property.level`*) yra žemesnis nei 4, tada reikšmė atrodo taip:

```
{"id": "69c98b0f-9e4e-424b-9575-9f601d79b68e"}
```

Čia `id` yra *`model.ref`* arba *kitas laukas*, per kurį daromas jungimas. Jei nenurodytas nei *`model.ref`*, nei *kitas laukas*, tada jungimas daromas per `_id`, tačiau netikrinama ar toks `_id` egzistuoja jungiamame modelyje.

**Taip pat žiūrėkite:**

*Ryšiai tarp modelių*

`type.`**`backref`**

Atgalinis ryšys su modeliu.

Jei ryšys tarp dviejų modlių yra daug su vienu, tada `property` pavadinimas nurodomas su `[]` simboliu.

---

**Pavyzdys**

Koncepcinis modelis

Struktūros aprašas

| model | property | type | ref |
|---|---|---|---|
| **Country** | | | id |
| | id | integer | |
| | name@lt | string | |
| | cities[] | backref | **City** |
| **City** | | | id |
| | id | integer | |
| | name@lt | string | |
| | country | ref | **Country** |

**Taip pat žiūrėkite:**

*Jungimas atgaliniu ryšiu*

`type.`**`generic`**

Dinaminis ryšys su modeliu.

Šis tipas naudojamas tada, kai yra poreikis perteikti dinaminį ryšį, t. y. duomenys siejami ne tik pagal id, bet ir pagal modelio pavadinimą. Tokiu būdu, vieno modelio laukas gali būti siejamas su keliais modeliais.

**Taip pat žiūrėkite:**

*Polimorfinis jungimas*

Šis duomenų tipas yra sudėtinis, susidedantis iš tokių duomenų:

**object_model**
: Pilnas modelio pavadinimas, su kuriuo yra siejamas objektas.

**object_id**
: `object_model` modelio objekto id.

`type.`**`object`**

Kompozicinis tipas, apjuniants kelias savybes į grupę, po vienu pavadinimu.

Šis tipas naudojamas apibrėžti sudėtiniams duomenims, kurie aprašyti naudojant kelis skirtingus tipas. Kompozicinio tipo atveju property stulpelyje komponuojami pavadinimai atskiriami taško simboliu.

Sudarant duomenų modelį, rekomenduojama laikytis plokščios struktūros ir komponavimą įgyvendinti siejant modelius per `ref` ar `generic` tipus.

`type.`**`array`**

Kompozicinis duomenų tipas, nurodo reikšmių masyvą.

Šis tipas naudojamas apibrėžti duomenų masyvams. Jei masyvo elementai turi vienodus tipus, tada elemento tipas pateikiamas property pavadinimo gale prirašant [] sufiksą, kuris nurodo, kad aprašomas ne pats masyvas, o masyvo elementas.

`type.`**`url`**
: Unikali resurso vieta (URL) (angl. *Uniform Resource Locator*).

  Šis tipas naudojamas pateikiant nuorodas į išorinius šaltinius.

  https://en.wikipedia.org/wiki/Uniform_Resource_Locator

`type.`**`uri`**
: Universalus resurso identifikatorius (URI) (angl. *Universal Resource Identifier*).

  Šis tipas naudojamas tais atvejais, kai pateikiamas išorinio resurso identifikatorius, RDF duomenų modelyje tai yra subjeto identifikatorius.

  https://en.wikipedia.org/wiki/Uniform_Resource_Identifier

## 2.6 Kodiniai pavadinimai

Kadangi *DSA* lentelė skirta naudoti tiek žmonėms tiek automatizuotoms priemonėms, tam tikros lentelės dalys privalo naudoti sutartinius kodinius pavadinimus. Kodiniams pavadinimams keliami griežtesni reikalavimai, kadangi šiuos pavadinimus interpretuos automatizuotos priemonės.

Visi *DSA* lentelės stulpelių pavadinimai turi būti užrašyti tiksliai taip, kaip nurodyta, kad kompiuterio programos galėtų juos atpažinti.

Kodiniai pavadinimai rašomi naudojant tik lotyniškas raidas. Lietuviškų raidžių naudoti negalima, todėl geriausia pavadinimus užrašyti anglų kalba, arba pakeičiant lietuviškas raides į lotyniškos raidės analogą.

Deja, vis dar pasitaiko vietų, kuriose palaikoma tik lotyniška abėcėlė, todėl ir keliamas toks reikalavimas, siekiant užtikrinti maksimalų suderinamumą tarp skirtingų sistemų.

Pavadinimai turėtu būti rašomi laikantis tokio stiliaus:

### 2.6.1 Vardų erdvių

Pavyzdys: `datasets/gov/abbr/short/word`

Visos mažosios raidės, stengiantis naudoti vieno žodžio trumpus pavadinimus arba žodžio trumpinius. Kadangi vardų erdvė rašoma prie kiekvieno modelio pavadinimo, todėl reikia stengtis vardų erdvių ir duomenų rinkinių pavadinimus išlaikyti kiek įmanoma trumpesnius.

Vardų erdvės pavadinimai užrašomi daugiskaita ir turi prasidėti mažąja raide.

### 2.6.2 Modeliai

Pavyzdys: `UpperCamelCase`

Kiekvieno modelio pavadinimo pirma raidė didžioji, kitos mažosios. Pavadinimo žodžiai atskiriami juos užrašant iš didžiosios raidės. Tarp žodžių neturi būti nei tarpų, nei kitų skyrybos ženklų.

Modelio pavadinimas įprastai užrašomas veinaskaitos forma.

Modelio kodinius pavadinimus užrašome taip, kaip pavadintume vieną objektą, kuriam yra taikomas duomenų modelis. Tarkime jei aprašome pastatus, tai vienas pastatas būtų vadinamas vienaskaitos forma `Pastatas`. Tačiau, jei vienas objektas yra patatų grupė, kuriuos jungia bendra paskirtis, tada galima pavadinti `Pastatai` arba `PastatuKompleksas`.

Modelio pavadinimas turi atspindėti duomenų subjekto tipą. Duomenų subjektas yra dalykas turintis pavadinimą ar unikalų identifikatorių. Duomenų subjekto tipas yra subjektų grupė priklausančių tai pačiai kategorijai ar klasei.

#### Nekartojame vardų erdvės

Modelio pavadinime nekartojamos vardų erdvės, kurioje yra modelis.

Pavyzdys, kaip nereikėtų daryti: `example/planets/EarthPlanet`. Šioje vietoje nereikia kartoti `Planet`, kadangi tai atsispindi vardų erdvės pavadinime `planets`.

### 2.6.3 Duomenų laukai

Pavyzdys: `snake_case`

Visi duomenų lauko žodžiai rašomi mažosiomis raidėmis, atskiriami pabraukimo ženklu `_`.

Duomenų lauko pavadinimas turi prasidėti mažąja raide.

#### Ryšiai tarp modelių

*`ref`* tipo laukai rašomi be `id` ar `_id` galūnės, kadangi jis yra perteklinis.

*`ref`* tipo laukai atspindi ne konkretų identifikatorių, o visą objektą. Konkretus identifikatorius yra rezervuotas pavadinimas ir duomenų struktūros apraše nenurodomas.

Pavyzdžiui vietoje `country_id`, kurio tipas yra `ref`, reikėtų rašyti `country`.

| m | property | type | ref |
|---|---|---|---|
| Country | | | |
| | name@lt | text | |
| City | | | |
| | name@lt | text | |
| | country | ref | Country |

Tais atvejais, kai duomenys yra denormalizuoti, duomenų lauko pavadinimas užrašomas su tašku, nurodant duomenų lauką iš siejamo modelio. Plačiau apie tai *Denormalizuoti duomenys*.

### Nekartojame modelio pavadinimo

Visi modelio duomenų laukai yra konkretaus modelio laukai, todėl nereikia kartoti duomenų laukuose modelio pavadinimo, pavyzdžiui vietoje tokių pavadinimų:

| m | property |
|---|---|
| City | |
| | city_id |
| | city_name |

Reikėtų rašyti taip:

| m | property |
|---|---|
| City | |
| | id |
| | name |

Jei kiti modeliai siejami su `City`, tada nurodant tarkim `city_name` iš kito modelio, reikėtų rašyti `city.city_name`. Todėl `city.name` yra aiškesnis pavadinimas, kuriame nesikartoja modelio pavadinimas.

### Nekartojame duomenų tipo pavadinimo

Duomenų lauko pavadinime nereikia kartoti duomenų tipo pavadinimo.

Pavyzdžiui taip nereikėtų daryti:

| m | property | type |
|---|---|---|
| City | | |
| | founded_date | date |

Reikėtų rašyti taip:

| m | property | type |
|---|---|---|
| City | | |
| | founded | date |

Nėra prasmės kartoti duomenų tipo, lauko pavadinime.

## 2.7 Matavimo vienetai

### 2.7.1 Apibrėžtis laike

`date` ir `datetime` tipo duomenų laukams gali būti žymimas ir laiko duomenų tikslumas, pavyzdžiui:

| d | r | b | m | property | type | ref | level |
|---|---|---|---|---|---|---|---|
| datasets/example | | | | | | | |
| | | | Matavimas | | | id | |
| | | | | id | integer | | 4 |
| | | | | laikas | datetime | 1S | 4 |
| | | | | vieta | geometry(point, 4326) | 1m | 4 |

Šiuo atveju, nurodyta, kad laukas `laikas` yra 1 sekundės tikslumu, o `vieta` 1 metro tikslumu.

Žymint laiko tikslumą, galite naudoti tokius sutartinius simbolius (atkreipkite dėmesį, kad šie vienetai veikia tik su `date` ir `datetime` tipais):

| Simbolis | Prasmė |
|---|---|
| Y | Metai |
| Q | Metų ketvirčiai |
| M | Mėnesiai |
| W | Savaitės |
| D | Dienos |
| H | Valandos |
| T | Minutės |
| S | Sekundės |
| L | Milisekundės |
| U | Mikrosekundės |
| N | Nanosekundės |

### 2.7.2 Apibrėžtis erdvėje

`geometry` tipo duomenų laukams gali būti žymimas erdvinių duomenų tikslumas, pavyzdžiui:

| Simbolis | Prasmė |
|---|---|
| nm | Nanometrai ($10^{-9}$ m) |
| mm | Milimetrai ($10^{-3}$ m) |
| cm | Centimetrai ($10^{-2}$ m) |
| m | Metrai |
| km | Kilometrai ($10^{3}$ m) |

### 2.7.3 Kokybiniai duomenys

Kokybiniai duomenys skirstomi į dvi kategorijas:

- pavadinimai ir identifikatoriai
- kategoriniai duomenys

Pavadinimai ir identifikatoriai *`property.ref`* stulpelyje neturi jokio žymėjimo ir jiems suteikiamas 4 brandos lygis.

Kategoriniai duomenys žymimi naudojant papildomą `enum` dimensiją, kurioje išvardinamos visos galimos kategorinių duomenų reikšmės.

Jei kategoriniai duomenys yra palyginami, pavyzdžiui:

- puikiai
- gerai
- vidutiniškai
- blogai

Tada, tokiems duomenims, turi būti naudojamas `integer` tipas, kad nebūtų prarastos palyginamosios savybės.

Jei duomenys yra nepalyginami, pavyzdžiui:

- raudona
- geltona
- žalia
- mėlyna

tada nebūtina naudoti `integer` tipą.

### 2.7.4 Kiekybiniai duomenys

Matavimo vienetai naudojant SI simbolius, išvestinius SI simbolius ir simbolius patvirtintus naudojimui su SI, pateikiami *`property.ref`* stulpelyje.

Pateikus vienetus, laukui gali būti suteikiamas 4-as brandos lygis.

Pavyzdys:

| d | r | b | m | property | type | ref | level |
|---|---|---|---|---|---|---|---|
| datasets/example | | | | | | | |
| | | | Matavimas | | | id | |
| | | | | id | integer | | 4 |
| | | | | temperatura | number | °C | 4 |
| | | | | svorlis | number | kg | 4 |
| | | | | plotas | number | $m^2$ | 4 |
| | | | | turis | number | $m^3$ | 4 |
| | | | | greitis | number | km/h | 4 |

Vienetai užrašomi naudojant matematinę notaciją, kurioje galima naudoti skaičius, daugybos ir dalybos simbolius, kėlimą laipsniu ir atskirų vienetų sudėtį:

| Žymėjimas | | Reikšmė |
|---|---|---|
| . . * | | Daugyba |
| / | | Dalyba |
| (tarpas) | | Sudėtis |
| ^(+-)(skaičius) | arba $^{+}$ $^{-}$ $^{0}$ $^{1}$ $^{2}$ $^{3}$ $^{4}$ $^{5}$ $^{6}$ $^{7}$ $^{8}$ $^{9}$ | Kėlimas laipsniu |

Pavyzdžiai:

m
1m
10m
m^2
$m^2$
$km^{10}$
$kg \cdot m^2 \cdot s^{-3} \cdot A^{-1}$
kg*m^2*s^-3·A^-1
$8kg \cdot m^2 \cdot s^{-3} \cdot A^{-1}$
mg/l
g/m^2
mg/m^3
mm
U/m^2
U/m^3
%
ha
min
h
bar

U
10^6s
$10^6$s
µ/$m^3$
yr
3mo
yr 2mo 4wk
°C
°

### Prefiksai

Kiekybiniai matavimo vienetai gali turėti tokius prefiksus:

| Žymėjimas | $10^n$ | Priešdėlis |
|---|---|---|
| Y | $10^{24}$ | yotta |
| Z | $10^{21}$ | zetta |
| E | $10^{18}$ | exa |
| P | $10^{15}$ | peta |
| T | $10^{12}$ | tera |
| G | $10^{9}$ | giga |
| M | $10^{6}$ | mega |
| k | $10^{3}$ | kilo |
| h | $10^{2}$ | hecto |
| da | $10^{1}$ | deca |
| d | $10^{-1}$ | deci |
| c | $10^{-2}$ | centi |
| m | $10^{-3}$ | milli |
| µ | $10^{-6}$ | micro |
| n | $10^{-9}$ | nano |
| p | $10^{-12}$ | pico |
| f | $10^{-15}$ | femto |
| a | $10^{-18}$ | atto |
| z | $10^{-21}$ | zepto |
| y | $10^{-24}$ | yocto |

### Vienetai

#### Specialiejie vienetai

| Žymėjimas | Pavadinimas |
|---|---|
| U | vienetai (keikis vienetais) |
| % | procentai |

#### Laiko vienetai

Naudojami tik tais atvejais, kai matuojamas laiko kiekis, o ne data ir laikas. Datos ir laiko (`date` ir `datetime` tipai) tikslumui žymėti, naudojamos kitos žymės.

| Žymėjimas | Pavadinimas |
|---|---|
| s | sekundė |
| min | minutė |
| h | valanda |
| d | diena (24 valandos) |
| wk | savaitė (7 dienos) |
| mo | mėnuo (28-31 diena arba 4 savaitės) |
| yr | metai (354.37 dienos arba 12 mėnesių) |

#### SI Baziniai vienetai

| Žymėjimas | Pavadinimas |
|---|---|
| m | metre |
| g | gram |
| s | second |
| A | ampere |
| K | kelvin |
| mol | mole |
| cd | candela |

### SI Išvestiniai vienetai

| Žymėjimas | Pavadinimas |
|---|---|
| Hz | hertz |
| rad | radian |
| sr | steradian |
| N | newton |
| Pa | pascal |
| J | joule |
| W | watt |
| C | coulomb |
| V | volt |
| F | farad |
| Ω | ohm |
| S | siemens |
| Wb | weber |
| T | tesla |
| H | henry |
| °C | degree Celsius |
| lm | lumen |
| lx | lux |
| Bq | becquerel |
| Gy | gray |
| Sv | sievert |
| kat | katal |

### Kiti vienetai

| Žymėjimas | Pavadinimas |
|---|---|
| au | astronomical unit |
| ° | degree |
| ′ | arcminute |
| ″ | arcsecond |
| ha | hectare |
| l | litre |
| L | litre |
| t | tonne |
| Da | dalton |
| eV | electronvolt |
| Np | neper |
| B | bel |
| dB | decibel |
| Gal | gal (acceleration) |
| u | unified atomic mass unit |
| var | volt-ampere reactive |
| pc | parsec |
| $c_0$ arba c_0 | natural unit of speed |
| ħ | natural unit of action |

continues on next page

Table 1 – tęsinys iš praeito puslapio

| Žymėjimas | Pavadinimas |
|---|---|
| $m_e$ arba m_e | natural unit of mass |
| e | atomic unit of charge |
| $a_0$ arba a_0 | atomic unit of length |
| E_h | atomic unit of energy |
| M | nautical mile |
| kn | knot |
| Å | ångström |
| a | are |
| b | barn |
| bar | bar |
| atm | standard atmosphere |
| Ci | curie |
| R | roentgen |
| rem | rem |
| erg | erg |
| dyn | dyne |
| P | poise |
| st | stokes |
| Mx | maxwell |
| G | gauss |
| Oe | ørsted |
| sb | stilb |
| ph | phot |
| Torr | torr |
| kgf | kilogram-force |
| cal | calorie |
| µ | micron |
| xu | x-unit |
| γ | gamma (mass, magnetic flux density) |
| λ | lambda |
| Jy | jansky |
| mmHg | millimetre of mercury |

## 2.8 Ryšiai tarp modelių

Pateikiant metaduomenis apie ryšius tarp modelių, duomenų *brandos lygis* pakeliamas iki ketvirto lygio.

Ryšiai tarp modelių aprašomi tais atvejais, kai vienoje duomenų lentelėje naudojami identifikatoriai iš kitos lentelės.

### 2.8.1 Jungimas per pirminį raktą

Pavyzdžiui, jei turime tokias dvi duomenų lenteles:

| Country | | |
|---|---|---|
| id | name | code |
| 1 | Lietuva | lt |
| 2 | Latvija | lv |

| City | | |
|---|---|---|
| id | name | country |
| 1 | Vilnius | lt |
| 2 | Kaunas | lt |
| 3 | Ryga | lv |

Šiuo atveju, jei norime parengti aukščiau pateiktų duomenų struktūros aprašą, jis atrodytų taip:

| id | d | r | b | m | property | type | ref | level |
|---|---|---|---|---|---|---|---|---|
| 1 | datasets/gov/example/countries | | | | | | | |
| 2 | | | | Country | | | code | 4 |
| 3 | | | | | id | integer | | 4 |
| 4 | | | | | name | string | | 4 |
| 5 | | | | | code | string | | 4 |
| 6 | | | | City | | | id | 4 |
| 7 | | | | | id | integer | | 4 |
| 8 | | | | | name | string | | 4 |
| 9 | | | | | country | ref | Country | 4 |

Šiame duomenų struktūros apraše, 9-oje eilutėje `country` stulpelio tipas yra `ref`, tai reiškia, kad šis stulpelis yra kito modelio išorinis raktas. `property.ref` stulpelyje nurodyta kurio modelio išorinis raktas šis stulpelis yra. Šiuo atveju, tai yra `Country` modelis, kuris apibrėžtas 2-oje eilutėje.

Pagal nutylėjimą, ryšys su kitu modeliu nustatomas naudojant kitos lentelės pirminį raktą nurodytą *`model.ref`* stulpelyje. Šiuo atveju, `City .country` yra jungiamas per `Country.code`. Tai reiškia, kad `City.country` duomenų tipas turi sutapti su `Country.code` duomenų tipu, kuris yra `string`.

`property.ref` reikšmė gali būti pateikiama vienu iš šių variantų:

**`property.ref`**

**`model`**

`model` nurodo kito *`model`* pavadinimą kurio *`model.ref`* siejamas su *`property`*.

Jei *`model.ref`* pirminiam raktui naudoja daugiau nei vieną lauką, tada *`property.source`* laukas turi būti tuščias, o *`property.prepare`* turi būti pateikiamos kableliu atskirtos property reikšmės, kurios bus naudojamos susiejimui.

**model[property]**

Tais atvejais, kai *property* duomenys nesutampa su siejamo *model.ref*, galima nurodyti *property* iš *model*.

**model[*property]**

Jei susiejimui reikia daugiau nei vieno duomenų lauko ir jie nesutampa su model.ref, tada galima nurodyti kelias property reikšmes atskirtas kableliu. Tačiau šiuo atveju taip pat būtina nurodyti ir *property.prepare* kelias reikšmes atskirtas kableliu, o *property.source* reikšmė turi būti tuščia. *property.prepare* stulpelyje nurodomi kiti modelio *property* pavadinimai iš kurių duomenų reikšmių turi būti formuojamas sudėtinis raktas.

### 2.8.2 Jungimas per nepirminį raktą

Jei modelius reikia jungti ne per pirminį raktą, o per kitus laukus, tada naudojama `model[property]` forma.

Pavyzdžiui, jei turime tokius duomenis:

| Country | | |
|---|---|---|
| id | name | code |
| 1 | Lietuva | lt |
| 2 | Latvija | lv |

| City | | |
|---|---|---|
| id | name | country |
| 1 | Vilnius | lt |
| 2 | Kaunas | lt |
| 3 | Ryga | lv |

Kur `Country` pirminis raktas yra `id` ir norime jungti `City.country` per `Country.code`, tuomet duomenų struktūros aprašas atrodys taip:

| d | d | r | b | m | property | type | ref | level |
|---|---|---|---|---|---|---|---|---|
| 1 | datasets/gov/example/countries | | | | | | | |
| 2 | | | | Country | | | id | 4 |
| 3 | | | | | id | integer | | 4 |
| 4 | | | | | name | string | | 4 |
| 5 | | | | | code | string | | 4 |
| 6 | | | | City | | | id | 4 |
| 7 | | | | | id | integer | | 4 |
| 8 | | | | | name | string | | 4 |
| 9 | | | | | country | ref | Country[code] | 4 |

9-oje eilutėje `property.ref` stulpelyje pateikta `Country[code]` reikšmė, kuri `Country` nurodo su kokiu modeliu jungiame, o `code` nurodo su kokiu `Country` stulpeliu jungiame. Jei pateiktas tik modelis, tada jungiama per to modelio pirminį raktą, jei pateiktas stulpelis laužtiniuose skliausteliuose, tada jungiama per nurodytą stulpelį.

### 2.8.3 Jungimas per kompozicinį raktą

Jei modelius reikia jungti per kelis laukus, tada naudojama `model[*property]` forma, kur laužtiniuose skliaustuose pateikiami keli stulpeliai atskirti kableliais.

Pavyzdžiui, jei turime tokius duomenis:

| Country | | |
|---|---|---|
| id | name | code |
| 1 | Lietuva | lt |
| 2 | Latvija | lv |

| City | | | |
|---|---|---|---|
| id | name | country | country_id |
| 1 | Vilnius | lt | 1 |
| 2 | Kaunas | lt | 1 |
| 3 | Ryga | lv | 2 |

Kur `City` su `Country` yra jungiamas per du `country` ir `country_id` stulpelius, tuomet reikia įtraukti išvestinį duomenų lauką, kuriame formulės įrašomos į *`property.prepare`* pagalba apjungiami keli laukai į vieną kompozicinį raktą. Šiuo atveju duomenų struktūros aprašas atrodys taip:

| d | d | r | b | m | property | type | ref | prepare | level |
|---|---|---|---|---|---|---|---|---|---|
| 1 | datasets/gov/example/countries | | | | | | | | |
| 2 | | | | Country | | | id | | 4 |
| 3 | | | | | id | integer | | | 4 |
| 4 | | | | | name | string | | | 4 |
| 5 | | | | | code | string | | | 4 |
| 6 | | | | City | | | id | | 4 |
| 7 | | | | | id | integer | | | 4 |
| 8 | | | | | name | string | | | 4 |
| 9 | | | | | country_code | string | | | 4 |
| 10 | | | | | country_id | integer | | | 4 |
| 11 | | | | | country | ref | Country[id,code] | country_id, country_code | 4 |

Čia matome, kad 11-oje eilutėje buvo įtrauktas išvestinis laukas `country`, kuris išskaičiuojamas apjungiant `country_id` ir `country_code`. O ryšiui su `Country`, laužtiniuose skliausteliuose nurodyti du laukai iš jungiamo `Country` modelio. Abiejų jungiamų pusių pateiktas laukų sąrašas turi būti vienodo eiliškumo, o jungiami laukai turi turėti vienodus tipus.

Jei `Country` pirminis raktas būtų kompozicinis, pavyzdžiui `id, code`, tuomet, 11-oje eilutėje `property.ref` užtektu nurodyti tik `Country`.

### 2.8.4 Jungimas atgaliniu ryšiu

Jungiant modelius atgaliniu ryšiu kuriamas išvestinis arba virtualus laukas, kuriame analogiškai kaip ir paprasto ryšio atveju, apjungiami du modeliai, tik šiuo atveju kuriamas daug su vienas tipo ryšys.

Pavyzdžiui, jei turime tokius duomenis:

Country

| id | name |
|---|---|
| 1 | Lietuva |
| 2 | Latvija |

City

| id | name | country |
|---|---|---|
| 1 | Vilnius | 1 |
| 2 | Kaunas | 1 |
| 3 | Ryga | 2 |

Ir šiuos duomenis atitinkantį duomenų modelį:

Tai norint sukurti atgalinį ryšį iš `City` modelio į `Country` modelį, duomenų struktūros aprašas atrodys taip:

| model | property | type | ref | level |
|---|---|---|---|---|
| **Country** | | | id | 4 |
| | id | integer | | 4 |
| | name@lt | string | | 4 |
| | cities[] | backref | **City** | 4 |
| **City** | | | id | 4 |
| | id | integer | | 4 |
| | name@lt | string | | 4 |
| | country | ref | **Country** | 4 |

Čia atgalinis ryšys nurodytas 5-oje eilutėje, pateikiant virtualų `Country.cities` lauką, kuris jungiamas per `City.country` lauką, kadangi `City.country` turi ryšį su `Country`.

Jei `City` modelyje būtų pateikti keli stulpeliai susieti su `Country`, tada 5-oje eilutėje `property.ref` reikšmė turėtų nurodyti konkretų lauką, per kurį jungiama, pavyzdžiui `City[country]`.

### 2.8.5 Polimorfinis jungimas

**Pastaba:** Tokio tipo jungimas kol kas dar nėra įgyvendintas.

Polimorfinis jungimas yra toks ryšys tarp modelių, kai vieno modelio laukas yra siejamas su daugiau nei vienu kitu modeliu. Tokiam ryšiui nurodyti polimorfinis laukas turi dvi reikšmes, išorinio modelio pavadinimą ir to modelio stulpelio per kurį jungiama reikšmę.

Country

| id | name |
|---|---|
| 1 | Lietuva |
| 2 | Latvija |

City

| id | name | country |
|---|---|---|
| 1 | Vilnius | 1 |
| 2 | Ryga | 2 |

Event

| id | name | object_id | object_model |
|---|---|---|---|
| 1 | Įkūrimas | 1 | datasets/gov/example/countries/Country |
| 2 | Įkūrimas | 2 | datasets/gov/example/countries/Country |
| 3 | Įkūrimas | 1 | datasets/gov/example/countries/City |
| 4 | Įkūrimas | 2 | datasets/gov/example/countries/City |

Pavyzdyje aukščiau matome, kad yra du modeliai `Country` ir `City`, kuriuos jungia `Event` modelis per `object_id` ir `object_model` laukus. Pavyzdžiui `Event` kurio `id` yra 1, siejamas su `Country` modeliu, kurio `id` yra 1.

Tokių duomenų struktūros aprašas atrodys taip:

| d | d | r | b | m | property | type | ref | prepare | level |
|---|---|---|---|---|---|---|---|---|---|
| 1 | datasets/gov/example/countries | | | | | | | | |
| 2 | | | | Country | | | id | | 4 |
| 3 | | | | | id | integer | | | 4 |
| 4 | | | | | name | string | | | 4 |
| 5 | | | | | cities[] | backref | City | | 4 |
| 6 | | | | City | | | id | | 4 |
| 7 | | | | | id | integer | | | 4 |
| 8 | | | | | name | string | | | 4 |
| 9 | | | | | country | ref | Country | | 4 |
| 10 | | | | Event | | | id | | 4 |
| 11 | | | | | id | integer | | | 4 |
| 12 | | | | | name | string | | | 4 |
| 13 | | | | | object_id | integer | | | 4 |
| 14 | | | | | object_model | string | | | 4 |
| 15 | | | | | object | generic | Country | object_model, object_id | 4 |
| 16 | | | | | | | City | | |

15-oje eilutėje įtrauktas virtualus `Event.object` laukas, kuris 15-oje ir 16-oje eilutėse, *`property.ref`* stulpelyje išvardina du modelius `Country` ir `City`, su kuriais jungiamas šis laukas, per `object_model` ir `object_id` laukus, kurie aprašyti atskirai.

`object_id` ir `object_model` aprašomi atskirai tik todėl, kad duomenys ateina iš išorinio šaltinio. Jei duomenys rašomi tiesiogiai į Saugyklą, tada atskirai `generic` laukų apsirašyti nereikia.

### 2.8.6 Denormalizuoti duomenys

Denormalizuoti duomenų laukai yra tokie laukai, kurie pateikti viename modelyje, tačiau pagal semantinę prasmę priklauso skirtingiems modeliams.

Dažniausiai duomenų normalizavimas atveriant duomenis yra nepageidaujamas ir duomenų struktūra turėtu būti transformuojama į skirtingus modelius, pagal semantinę prasmę. Plačiau apie duomenų normalizavimą galite skaityti skyriuje norm.

Tačiau tais atvejais, kai vis dėlto norima pateikti duomenis denormalizuotoje formoje, duomenų struktūros apraše galima nurodyti, kurie duomenų laukai yra denormalizuoti.

Pavyzdys, kaip atrodo denormalizuotų duomenų laukų žymėjimas:

| d | r | b | m | property | type | ref | level |
|---|---|---|---|---|---|---|---|
| example | | | | | | | |
| | | | Country | | | code | 4 |
| | | | | code | string | | 4 |
| | | | | name@en | text | | 4 |
| | | | City | | | | 3 |
| | | | | name@en | text | | 4 |
| | | | | country | ref | Country | 4 |
| | | | | country.code | | | 2 |
| | | | | country.name@en | | | 2 |
| | | | | country.name@lt | text | | 2 |

Šiame pavyzdyje turime tokius laukus:

**`country`**
: Šis laukas yra `ref` tipo, tai reiškia, kad šiame lauke saugomas `Country` modelio identifikatorius, kurio pagalba `City` galima susieti su `Country`.

  `ref` tipo duomenys yra sudėtiniai, tai reiškia, kad per `ref` tipo lauką galima pasiekti siejamo modelio laukus, nurodant kito modelio laukus po taško.

  Todėl pagal nutylėjimą `country ref Country` yra tas pats, kas `country._id ref Country`, tik `._id` dalis nenurodoma.

**`country.code` ir `country.name@en`**
: Šie laukai yra denormalizuoti, tai reiškia, kad jie priklauso `Country` modeliui, tačiau duomenys yra dubliuojami ir pateikiami dviejose vietose, prie `Country` ir prie `City.country`.

  Kadangi `City.country` yra `ref` tipo, tai po taško, galima nurodyti kitus šiam siejamam modeliui priklausančius laukus iš kito modelio.

  Atkreipkite dėmesį, kad denormalizuotiems laukams nepildomas `type` stulpelis, kadangi šių laukų tipas turi sutapti su siejamo modelio laukų tipais, taip pat turi sutapti ir laukų pavadinimai.

**`country.name@lt`**
: Tais atvejais, kai siejamame modelyje (šiuo atveju `Country` modelyje) nėra tam tikrų laukų, tuomet galima juose pateikti ir prie `City.country`, tačiau tokiu atveju, būtina nurodyti `type`.

### 2.8.7 Brandos lygis

Apibrėžiant ryšius tarp modelių, brandos lygis įrašomas `level` stulpelyje atlieka svarbų vaidmenį. Nuo brandos lygio, priklauso, kaip turi būti interpretuojamas išorinis raktas, siejamas su kitu modeliu.

**1 brandos lygis: Susiejimas neįmanomas**
Duomenys pateikti tokia forma, kurios pagalba dviejų modelių jungimas nėra įmanomas.

Pavyzdžiui, pateikta adreso tekstinė forma, kuri nesutampa su tekstine forma pateikiama oficialiame adresų registre arba naudojamas toks tam tikras identifikatorius, kuris nėra surištas su siejamo modelio pirminiu raktu.

**2 brandos lygis: Susiejimas nepatikimas**
Duomenys pateikiami tam tikra forma, kuri neužtikrina patikimo duomenų susiejimo, tačiau siejimas atliekamas pagal siejamo modelio atributą, kuris negarantuoja unikalaus objekto identifikavimo.

Pavyzdžiui siejimas atliekamas pagal pavadinimą, kuris gali keistis arba ne visais atvejais sutampa.

**3 brandos lygis: Susiejimas ne per pirminį raktą**
Duomenims susieti naudojamas patikimas identifikatorius, kuris yra surištas su siejamo modelio pirminiu raktu, tačiau naudojamas ne pirminis raktas, o kitas identifikatorius.

**4 brandos lygis: Susiejimas per pirminį raktą**
Susiejimas daromas per pirminį raktą.

#### Susiejimas neįmanomas

Jei `ref` tipui nurodytas 1 arba žemesnis brandos lygis, tai reiškia, duomenų jungimas nėra įmanomas. Tokiu atveju, atveriant duomenis, `property` įgaus tokį tipą, koks yra lauko su kuriuo siejamas ryšys tipas.

Pavyzdžiui:

| d | r | b | m | property | type | ref | level |
|---|---|---|---|---|---|---|---|
| example | | | | | | | |
| | | | Country | | | name@lt | 4 |
| | | | | name@lt | text | | 4 |
| | | | City | | | name | 4 |
| | | | | name@lt | text | | 4 |
| | | | | country | ref | Country | 1 |

Šiuo atveju, `City.country` yra siejamas su `Country.name`. Kadangi `City.country` brandos lygis yra 2, tai reiškia, kad `City.country` ir `Country.name` pavadinimai nesutampa ir jungimo atlikti neįmanoma. Tokiu atveju, `City.country` tipas bus ne `ref`, o toks pat, kaip `Country.name`, t.y. `text`.

Tačiau, metaduomenyse išliks informacija, apie tai, kad šios lentelės yra susijusios. Dėl prasto duomenų brandos lygios, realus susiejimas nėra įmanomas.

Jei modeliai yra susiję, tačiau, tokio duomenų lauko, per kurį galima būtų atlikti susiejimą iš vis nėra, tuomet, tokį lauką galima sukurti, nurodant brandos lygį 0. Pavyzdžiui:

| d | r | b | m | property | type | ref | level |
|---|---|---|---|---|---|---|---|
| example | | | | | | | |
| | | | Country | | | name@lt | 4 |
| | | | | name@lt | text | | 4 |
| | | | | name@en | text | | 0 |
| | | | City | | | name | 4 |
| | | | | name@en | text | | 4 |
| | | | | country | ref | Country[name@en] | 1 |

Šioje vietoje `City.country` tampa `country@en`, kurio tipas yra `text`. O į `Country` yra įtrauktas papildomas laukas `name@en`, per kurį ir atliekamas susiejimas, t.y. per kurį galėtų būti atliktas susiejimas, jei toks laukas egzistuotų ne tik `City.country`, bet ir `Country.name@en`.

### Susiejimas nepatikimas

Jei `ref` tipui suteiktas 2 brandos lygis, tai reiškia, kad susiejimas yra įmanomas, tačiau nėra garantijos, kad jis veiks visais atvejais.

Susiejimas laikomas nepatikimu, tada, kai siejimas atliekamas ne patikimo unikalaus identifikatoriaus pagalba, o per pavadinimą ar panašiais būdais.

Pavadinimai gali keistis, gali dubliuotis, gali skirtis jų užrašymo forma, todėl toks jungimas laikomas nepatikimu.

Toks jungimas ir 2 brandos lygio žymėjimas taikomas tik tais atvejais, kai jungimas daromas, per jungiamo modelio atributą. Pavyzdžiui:

| d | r | b | m | property | type | ref | level |
|---|---|---|---|---|---|---|---|
| example | | | | | | | |
| | | | Country | | | name@lt | 4 |
| | | | | name@lt | text | | 4 |
| | | | City | | | name | 4 |
| | | | | name@lt | text | | 4 |
| | | | | country | ref | Country | 2 |

Šiuo atveju, kadangi `City.country` brandos lygis yra 2, tai reiškia, kad `City.country` duomenys yra realiai paimti iš `Country.name@lt`. Jei `City.country` būtų paimti ne iš `Country.name@lt`, o iš kokio nors kito šaltinio ir gali nesutapti, tada brandos lygis turėtu būti 1.

Tai reiškia, kad 2 brandos lygis žymimas tik tais atvejais, kai išorinis raktas yra paimtas iš siejamo modelio atributo.

### Susiejimas ne per pirminį raktą

Jei `ref` tipui suteiktas 3 ar didesnis brandos lygis, vadinasi susiejimas yra patikimas. Duomenys siejami naudojant patikimus unikalius identifikatorius, kurie nesidubliuoja, nesikeičia ir užrašomi visada vienodai.

Dažniausiai patikimais identifikatoriais laikomi sveiki skaičiai, tam tikri sutartiniai kodai ir kiti specializuoti identifikatoriai, tokie kaip UUID.

Tačiau, naudojamas ne pirminis raktas, o kitas duomenų laukas. Pavyzdžiui:

| d | r | b | m | property | type | ref | level |
|---|---|---|---|---|---|---|---|
| example | | | | | | | |
| | | | Country | | | id | 4 |
| | | | | id | integer | | 4 |
| | | | | code | string | | 4 |
| | | | | name@lt | text | | 4 |
| | | | City | | | name | 4 |
| | | | | name@lt | text | | 4 |
| | | | | country | ref | Country[code] | 3 |

Skirtumas tarp 3 ir 4 brandos lygio iš esmės susijęs su duomenų saugojimu Saugykloje ar kitoje vietoje, kur pirminiai raktai yra generuojami ir jų negalima keisti. Jei naudojamas 3 brandos lygis, tuomet saugykloje saugomas, ne išorinis saugyklos identifikatorius UUID, o vidinis duomenų rinkinio identifikatorius.

Publikuojant duomenis iš tam tikro šaltinio, išorinis raktas visada turėtu būti konvertuojamas į išorinį pirminį raktą, tačiau tais atvejais, jei dėl tam tikrų priežasčių tas nėra daroma, tuomet žymimas 3 brandos lygis ir publikuojami ne išoriniai pirminiai raktai, o šaltinio vidiniai.

Pavyzdžiui, jei turime tokius duomenis:

| example/Country<br>_id | id | code | name@lt |
|---|---|---|---|
| 4dbb1b77-a930-4f2a-8ef4-f05b89f0fcfe | 1 | lt | Lietuva |

Ir jei `City.country` turi brandos lygį 3, tada `City` duomenys atrodys taip:

| example/City<br>_id | name@lt | country._id |
|---|---|---|
| 096e054e-7a4c-44cc-8f27-98af815080d5 | Vilnius | lt |

### Susiejimas per pirminį raktą

Šiuo atveju, brandos lygis žymimas 4 ir skirtumas nuo 3 brandos lygio yra toks, kad duomenyse naudojamas išorinis pirminis raktas. Pavyzdžiui:

| d | r | b | m | property | type | ref | level |
|---|---|---|---|---|---|---|---|
| example | | | | | | | |
| | | | Country | | | id | 4 |
| | | | | id | integer | | 4 |
| | | | | code | string | | 4 |
| | | | | name@lt | text | | 4 |
| | | | City | | | name | 4 |
| | | | | name@lt | text | | 4 |
| | | | | country | ref | Country | 4 |

Turint tokį struktūros aprašą, kur `City.country` brandos lygis yra 4, duomenys atrodys taip:

| example/Country<br>_id | id | code | name@lt |
|---|---|---|---|
| 4dbb1b77-a930-4f2a-8ef4-f05b89f0fcfe | 1 | lt | Lietuva |

| example/City<br>_id | na-<br>me@lt | country._id |
|---|---|---|
| 096e054e-7a4c-44cc-8f27-<br>98af815080d5 | Vilnius | 4dbb1b77-a930-4f2a-8ef4-<br>f05b89f0fcfe |

Matome, kad `City.country._id` yra `Country` pirminis raktas. Tai reiškia, kad vidiniai duomenų rinkinio raktai konvertuojami į išorinius.

## 2.9 Brandos lygiai

Duomenų brandos lygis nurodomas `level` stulpelyje.

Duomenų brandos lygis atitinka 5 ★ Open Data skalę, tačiau ji yra adaptuota duomenų struktūros aprašo kontekstui, brandos lygius pritaikant ir uždariems duomenims.

Pirmasis brandos lygis, kuris nurodo, kad duomenys turi būti publikuojami pagal atvirą licenciją, DSA kontekste, šis reikalavimas išplėstas ir apima ne tik atviras licencijas, bet ir bet kokias duomenų teikimo sąlygas.

Todėl reikia atkreipti dėmesį, kad duomenų brandos lygis ar formatas nėra susijęs su duomenų atvirumu. Duomenys gali būti pateikti aukščiausiu 5 brandos lygiu, tačiau pats duomenų prieinamumas gali būti visiškai uždaras, siauram naudotojų ratui, kurie turi ribotą ir saugų prieigos kanalą prie duomenų.

Papildomai įtrauktas nulinis lygis, kai duomenys nekaupiami, tačiau yra reikalingi ir yra parengtas jų *duomenų struktūros aprašas*.

Tim Berners Lee brandos lygius aprašo, kaip pavyzdį pasitelkiant duomenų distribucijų formatus. Tačiau duomenų distribucijų formatai yra labai netikslus pavyzdys. Rengiant duomenų struktūros aprašą, brandos lygis vertinamas kiekvienam duomenų laukui atskirai, todėl tarkime CSV failas, gali būti didesnio arba mažesnio nei trečias brandos lygio, priklausomai nuo CSV faile esančių duomenų turinio. Tačiau grubiai vertinant, vidutiniškai CSV failai turi daugiau ar mažiau 3 brandos lygį, koks ir yra nurodytas Tim Berners Lee pavyzdžiuose.

Rengiant duomenų struktūros aprašą, reikėtų nurodyti ne šaltinio duomenų brandos lygį, o galutinį brandos lygį, kuris yra gaunamas atlikus visas duomenų struktūros apraše nurodytas transformacijas.

### 2.9.1 L000: Duomenų nėra

**Nekaupiama**

Duomenys nekaupiami. Duomenų rinkinys užregistruotas duomenų portale. Užpildyta `dataset` eilutė.

Plačiau apie brandos lygio kėlimą skaitykite skyriuje to-level-0.

**Pavyzdžiai**

Imone

| imones_id | imones_pavadinimas | rusis |
|---|---|---|
| 42 | UAB "Įmonė" | 1 |

Filialas

| ikurimo_data | atstumas | imones_id | imones_pavadinimas._id | tel_nr |
|---|---|---|---|---|

| Struktūros aprašas<br>d r b m property | type | ref | level |
|---|---|---|---|
| example | | | |
| Imone | | imones_id | 4 |
| imones_id | integer | | 4 |
| imones_pavadinimas | string | | 2 |
| rusis | integer | | 2 |
| Filialas | | | 0 |
| ikurimo_data | string | | 0 |
| atstumas | string | | 0 |
| imone | ref | Imone | 0 |
| tel_nr | string | | 0 |

### L001: Duomenys nepublikuojami

Nuliniu brandos lygiu žymimi duomenys, kurie fiziškai egzistuoja, tačiau nėra publikuojami jokia forma.

### L002: Ribojamas duomenų naudojimas

Nuliniu brandos lygiu yra žymimi duomenys, kurie yra saugomi duomenų bazėse, ir yra poreikis juos naudoti, tačiau duomenys nėra teikiami už informacinės sistemos ribų.

### L003: Nėra identifikatoriaus

Duomenų šaltinis neturi jokio unikalaus objekto identifikatoriaus.

### L004: Duomenų nėra

Apibrėžtas duomenų modelis, tačiau pačių duomenų kol kas nėra.

### 2.9.2 L100: Nenuskaitoma mašininiu būdu

Duomenys publikuojami bet kokia forma. Užpildyta *resource* eilutė.

Plačiau apie brandos lygio kėlimą skaitykite skyriuje to-level-1.

Pirmu brandos lygiu žymimi duomenų laukai, kurių reikšmės neturi vientisumo, tarkime ta pati reikšmė gali būti pateikta keliais skirtingais variantais.

**Pavyzdžiai**

| Imone<br>imones_id | imones_pavadinimas | rusis |
|---|---|---|
| 42 | UAB "Įmonė" | 1 |

| Filialas<br>ikurimo_data | atstumas | imones_id._id | imo-<br>nes_pavadinimas | tel_nr |
|---|---|---|---|---|
| vakar | 1 m. | 1 | Įmonė 1 | +370 345 36522 |
| 2021 rugpjūčio 1 d. | 1 m | 1 | UAB Įmonė 1 | 8 345 36 522 |
| 1/9/21 | 1 metras | 1 | Įmonė 1, UAB | (83) 45 34522 |
| 21/9/1 | 0.001 km | 1 | „ĮMONĖ 1“, UAB | 037034536522 |

| Struktūros aprašas | | | | | | | |
|---|---|---|---|---|---|---|---|
| d | r | b | m | property | type | ref | level |
| example | | | | | | | |
| | | | Imone | | | id | 4 |
| | | | | imones_id | integer | | 2 |
| | | | | imones_pavadinimas | string | | 2 |
| | | | | rusis | integer | | 2 |
| | | | Filialas | | | | 3 |
| | | | | ikurimo_data | string | | 1 |
| | | | | atstumas | string | | 1 |
| | | | | imones_id | ref | Imone | 1 |
| | | | | imones_pavadinimas | string | | 1 |
| | | | | tel_nr | string | | 1 |

### L101: Neaiški struktūra

Pirmu brandos lygiu žymimi duomenys, kuriuose nėra aiškios struktūros, pavyzdžiui `ikurta` datos formatas nėra vienodas, kiekviena data užrašyta vis kitokiu formatu.

### L102: Nėra vientisumo

Pirmu brandos lygiu žymimi duomenys, kuruose nėra vientisumo, pavyzdžiui `atstumas` užrašytas laikantis tam tikros struktūros, tačiau skirtingais vienetais.

### L103: Neįmanomas jungimas

Pirmu brandos lygiu žymimi duomenys, kurių neįmanoma arba sudėtinga sujungti. Pavyzdžiui `Filialas` duomnų laukas `imone` naudoja tam tikrą identifikatorių, kuris nesutampa nei su vienu iš `Imone` atributų, pagal kuriuose būtų galima identifikuoti filialo įmonę.

#### L104: Identifikatorius nėra unikalus

Objekto identifikatorius nėra unikalus, turi pasikartojančių reikšmių.

#### L105: Vienetų konvertavimo paklaida

Tam tikrais atvejais, kai kiekybiniai duomenys pateikiami išskaidant į atskirus duomenų laukus, gali būti prarandamas duomenų tikslumas.

---

**Pavyzdys**

| model | property | type | ref | level |
|---|---|---|---|---|
| Assmuo | | | id | 4 |
| | id | integer | | 4 |
| | metai | integer | yr | 4 |
| | menesiai | integer | mo | 1 |
| | dienos | integer | d | 1 |

Šiame pavyzdyje nurodytas asmens amžius pateikiant atskirai metus, mėnesius ir dienas, tarkim *25 metai, 5 mėnesiai, 29 dienos*.

Jei norėtume gauti amžių dienomis, rezultatas $25 * 365 + 5 * 30 + 29 = 9304$ būtų netikslus, kadangi metai ir mėnesiai turi skirtingus dienų skaičius, todėl konvertuojant rezultatą į dienas, gausime netikslų rezultatą.

Kadangi nustatyti nėra galimybės nustatyti galutinio tikslaus mėnesio ir dienų skaičiaus, nurodomas 1 brandos lygis.

---

### 2.9.3 L200: Nestandartinis pateikimas

Duomenys kaupiami struktūruota, mašininiu būdu nuskaitoma forma, bet kokiu formatu. Užpildytas `property.source` stulpelis.

Plačiau apie brandos lygio kėlimą skaitykite skyriuje to-level-2.

Antru brandos lygiu žymimi duomenų laukai, kurie pateikti vieninga forma arba pagal aiškų ir vienodą šabloną. Tačiau pateikimo būdas nėra standartinis. Nestandartinis duomenų formatas yra toks, kuris neturi viešai skelbiamos ir atviros formato specifikacijos arba, kuris nėra priimtas kaip standartas, kurį prižiūri tam tikra standartizacijos organizacija.

**Pavyzdžiai**

| Imone<br>imones_id | imones_pavadinimas | rusis |
|---|---|---|
| 42 | UAB "Įmonė" | 1 |

| Filialas<br>ikurimo_data | atstumas | imones_id | imones_pavadinimas._id | tel_nr |
|---|---|---|---|---|
| 1/9/21 | 1 m. | 1 | UAB "Įmonė" | (83) 111 11111 |
| 2/9/21 | 2 m. | 1 | UAB "Įmonė" | (83) 222 22222 |
| 3/9/21 | 3 m. | 1 | UAB "Įmonė" | (83) 333 33333 |
| 4/9/21 | 4 m. | 1 | UAB "Įmonė" | (83) 444 44444 |

| Struktūros aprašas<br>d | r | b | m | property | type | ref | level |
|---|---|---|---|---|---|---|---|
| example | | | | | | | |
| | | | JuridinisAsmuo | | | kodas | 4 |
| | | | | kodas | integer | | 4 |
| | | | | pavadinimas@lt | text | | 4 |
| | | | Imone | | | imones_id | 2 |
| | | | | imones_id | integer | | 2 |
| | | | | imones_pavadinimas | string | | 2 |
| | | | | rusis | integer | | 2 |
| | | | Filialas | | | | 3 |
| | | | | ikurimo_data | string | | 2 |
| | | | | atstumas | string | | 2 |
| | | | | imones_id | integer | | 2 |
| | | | | imones_pavadinimas | string | | 2 |
| | | | | tel_nr | string | | 2 |

### L201: Nestandartiniai duomenų tipai

Antru brandos lygiu žymimi duomenys, kurių nurodytas tipas neatitinka realaus duomenų tipo. Pavyzdžiui:

- `ikurimo_data` - nurodytas `string`, turėtu būti `date`.
- `imones_pavadinimas` - nurodytas `string`, turėtu būti `text`.
- `atstumas` - nurodytas `string`, turėtu būti `integer`.

### L202: Nestandartinis formatas

Antru brandos lygiu žymimi duomenys, kurie pateikti nestandartiniu formatu. Standartinis duomenų pateikimas nurodytas prie kiekvieno duomenų tipo skyriuje *Duomenų tipai*. Pavyzdžiui:

- `ikurimo_data` - nurodytas `DD/MM/YY`, turėtu būti `YYYY-MM-DD`.
- `atstumas` - nurodyta `X m.`, turėtu būti `X`.
- `tel_nr` - nurodytas `(XX) XXX XXXXX`, turėtu būti `+XXX-XXX-XXXXX`.

### L203: Nestandartiniai kodiniai pavadinimai

Antru brandos lygiu žymimi duomenys, kurių kodiniai pavadinimai, neatitinka *standartinių reikalavimų keliamų kodiniams pavadinimams*. Pavyzdžiui:

- `imones_id` - dubliuojamas modelio pavadinimas, turėtu būti `id`.
- `imones_pavadinimas` - dubliuojamas modelio pavadinimas, turėtu būti `pavadinimas`.
- `ikurimo_data` - dubliuojamas tipo pavadinimas, turėtu būti `ikurta`.

**Taip pat žiūrėkite:**

*Kodiniai pavadinimai*

### L204: Nepatikimi identifikatoriai

Antru brandos lygiu žymimi duomenys, kurių `ref` tipui naudojami nepatikimi identifikatoriai, pavyzdžiui tokie, kaip pavadinimai, kurie gali keistis arba kartotis. Pavyzdžiui:

- `imones_pavadinimas` - jungimas daromas per įmonės pavadinimą, tačiau šiuo atveju kito varianto nėra, nes `Filialas.imones_id` nesutampa su `Imone.imones_id`.

### L205: Denormalizuoti duomenys

Antru brandos lygiu žymimi duomenys, kurie dubliuoja kito modelio duomenis ir yra užrašyti nenurodant, kad tai yra duomenys dubliuojantys kito modelio duomenis. Pavyzdžiui:

- `Filialas.imones_id` turėtu būti `Filialas.imone.imones_id`.
- `Filialas.imones_pavadinimas` turėtu būti `Filialas.imone.imones_pavadinimas`.

Plačiau apie denormalizuotus duomenis skaitykite skyriuje *Denormalizuoti duomenys*.

### L206: Nenurodytas susiejimas

Antru brandos lygiu žymimi duomenys, kurie siejasi su kitu modeliu, tačiau tokia informacija nėra pateikta metaduomenyse. Pavyzdžiui:

- `Filialas.imone` - `Filialas` siejasi su `Imone`, per `Filialas.imones_pavadiniams`, todėl turėtu būti nurodytas `imone ref Imone` ryšys su `Imone`.

### L207: Neatitinka modelio bazės

Antru brandos lygiu žymimi duomenys, kurie priklauso vienai semantinei klasei, tačiau duomenų schema nesutampa su bazinio modelio schema. Pavyzdžiui:

- `Imone` - priklauso semantinei klasei `JuridinisAsmuo`, tačiau tai nėra pažymėta metaduomenyse.
- `Imone.imones_id` turėtu būti `Imone.kodas`, kad sutaptu su baze (`JuridinisAsmuo.kodas`).
- `Imone.imones_pavadinimas` turėtu būti `Imone.pavadinimas@lt`, kad sutaptu su baze (`JuridinisAsmuo.pavadinimas@lt`).

### L208: Nenurodytas enum kodinėms reikšmėms

Antru brandos lygiu žymimi kategoriniai duomenys, kurių reikšmės pateiktos sutartiniais kodais, kurių prasmė nėra aiški. Pavyzdžiui:

- `Imone.rusis` - įmonės rūšis žymima skaičiais, tačiau nėra aišku, koks skaičius, ką reiškia, todėl reikia pateikti `enum` sąrašą, kuriame būtų nurodyta, ką koks skaičius reiškia. Plačiau skaityti *enum*.

### L209: Nenurodyta modelio bazė

Modelis atitinka registre apibrėžtą esybę, tačiau nėra su ja susietas.

### L210: Išskaidyta atskirais komponentais

*DSA* turi sudėtinius tipus, tokius kaip `date`, `datetime` ir `geometry`, kuri apima kelis atskirus duomenų komponentus, kurie pateikiami kaip viena reikšmė, nustatytu formatu.

Jei komponentai yra išskaidyti į atskirus duomenų laukus, tuomet tai yra nestandartinis duomenų pateikimas žymimas 2 brandos lygiu.

### 2.9.4 L300: Nėra identifikatoriaus

Duomenys saugomi atviru, standartiniu formatu. Užpildytas *property.type* stulpelis ir duomenys atitinka nurodytą tipą.

Plačiau apie brandos lygio kėlimą skaitykite skyriuje to-level-3.

Trečias brandos lygis suteikiamas tada, kai duomenys pateikti vieninga forma, vieningu masteliu, naudojamas formatas yra standartinis, tai reiškia, kad yra viešai skelbiama ir atvira formato specifikacija arba pats formatas yra patvirtintas ir prižiūrimas kokios nors standartizacijos organizacijos.

**Pavyzdžiai**

| Imone | | |
|---|---|---|
| id | pavadinimas@lt | rusis |
| 42 | UAB "Įmonė" | juridinis |

| Filialas | | | | |
|---|---|---|---|---|
| ikurta | atstumas | imone._id | imone.pavadinimas@lt | tel_nr |
| 2021-09-01 | 1 | 42 | UAB "Įmonė" | +37011111111 |
| 2021-09-02 | 2 | 42 | UAB "Įmonė" | +37022222222 |
| 2021-09-03 | 3 | 42 | UAB "Įmonė" | +37033333333 |
| 2021-09-04 | 4 | 42 | UAB "Įmonė" | +37044444444 |

| Struktūros aprašas<br>d | r | b | m | property | type | ref | level | prepare | title |
|---|---|---|---|---|---|---|---|---|---|
| example | | | | | | | | | |
| | | | JuridinisAsmuo | | | kodas | 4 | | |
| | | | | kodas | integer | | 4 | | |
| | | | | pavadinimas@lt | text | | 4 | | |
| | | JuridinisAsmuo | | | | | 4 | | |
| | | | Imone | | | kodas | 4 | | |
| | | | | kodas | | | 4 | | |
| | | | | pavadinimas@lt | | | 4 | | |
| | | | | rusis | string | | 3 | | |
| | | / | | | | | | | |
| | | | Filialas | | | | 3 | | |
| | | | | ikurta | date | | 3 | | |
| | | | | atstumas | integer | | 3 | | |
| | | | | imone | ref | Imone | 3 | | |
| | | | | imone.kodas | | | 4 | | |
| | | | | imone.pavadinimas@lt | | | 4 | | |
| | | | | tel_nr | string | | 4 | | |

### L301: Nėra globalaus objekto identifikatoriaus

Nėra naudojamas globalus objekto identifikatorius, objektas identifikuojamas naudojant tik lokalų identifikatorių. Tokiu atveju, objektas negali būti nuskaitomas tiesiogiai, gali būti vykdoma tik atranka, nurodant filtrą, pagal lokalų identifikatorių.

- `Filialas.imone` - siejimas atliekamas per `Imone.kodas`, o ne per `Imone._id`.

#### L302: Nenurodyti matavimo vienetai

Trečiu brandos lygiu žymimi kiekybiniai duomenys, kuriems nėra nurodyti matavimo vienetai *property.ref* stulpelyje. Pavyzdžiui:

- `atstumas` - nenurodyta, kokiais vienetais matuojamas atstumas.

#### L303: Nenurodytas duomenų tikslumas

Trečiu brandos lygiu žymimi laiko ir erdviniai duomenys, kuriems nėra nurodytas matavimo tikslumas. Matavimo tikslumas nurodomas `property.ref` stulpelyje. Pavyzdžiui:

- `ikurta` - nenurodytas datos tikslumas, turėtu būti `D` - vienos dienos tiksumas.

#### L304: Neaprašyti kategoriniai duomenys

Trečiu brandos lygiu žymimi kategoriniai duomenys, kurių reikšmės pačios savaime yra aiškios, tačiau neišvardintos struktūros apraše. Pavyzdžiui:

- `Imone.rusis` - įmonės rūšies kategorijos duomenys yra pateikta tekstine forma, tačiau, struktūros apraše nėra išvardintos visos galimos kategorijos ir pats duomenų laukas nėra pažymėtas, kaip kategorinis.

### 2.9.5 L400: Nesusieata su žodynais

Duomenų objektai turi aiškius, unikalius identifikatorius. Užpildyti *model.ref* ir *property.ref* stulpeliai.

---

**Pastaba:** *property.ref* stulpelis pildomas šiais atvejais:

- Jei duomenų laukas yra išorinis raktas (žiūrėti ref-types).
- Jei duomenų laukas yra kiekybinis ir turi matavimo vienetus (žiūrėti *Matavimo vienetai*).
- Jei duomenų laukas žymi laiką ar vietą (žiūrėti temporal-types ir spatial-types).

---

Plačiau apie brandos lygio kėlimą skaitykite skyriuje to-level-4.

Ketvirtas duomenų brandos lygis labiau susijęs ne su pačių duomenų formatu, bet su metaduomenimis, kurie lydi duomenis.

Duomenų struktūros apraše *model.ref* stulpelyje, pateikiamas objektą unikaliai identifikuojančių laukų sąrašas, o *property.type* stulpelyje įrašomas `ref` tipas, kuris nurodo ryšį tarp dviejų objektų.

**Pavyzdžiai**

| Imone<br>_id | id | pavadinimas@lt | rusis |
|---|---|---|---|
| 26510da5-f6a6-45b0-a9b9-27b3d0090a58 | 42 | UAB "Įmonė" | 1 |

| Filialas _id | id | ikurta | atstumas | imone._id | imone.id | imone.pavadinii | tel_nr |
|---|---|---|---|---|---|---|---|
| 63161bd2-158f-4d62-9804-636573abb9c7 | 1 | 2021-09-01 | 1 | 26510da5-f6a6-45b0-a9b9-27b3d0090a58 | 42 | UAB "Įmonė" | +37011111111 |
| 65ec7208-fb97-41a8-9cfc-dfedd197ced6 | 2 | 2021-09-02 | 2 | 26510da5-f6a6-45b0-a9b9-27b3d0090a58 | 42 | UAB "Įmonė" | +37022222222 |
| 2b8cdfa6-1396-431a-851c-c7c6eb7aa433 | 3 | 2021-09-03 | 3 | 26510da5-f6a6-45b0-a9b9-27b3d0090a58 | 42 | UAB "Įmonė" | +37033333333 |
| 1882bb9e-73ee-4057-b04d-d4af47f0aae8 | 4 | 2021-09-04 | 4 | 26510da5-f6a6-45b0-a9b9-27b3d0090a58 | 42 | UAB "Įmonė" | +37044444444 |

| Struktūros aprašas<br>d | r | b | m | property | type | ref | le-<br>vel | prepa-<br>re | title |
|---|---|---|---|---|---|---|---|---|---|
| example | | | | | | | | | |
| | | | JuridinisAsmuo | | | kodas | 4 | | |
| | | | | kodas | inte-<br>ger | | 4 | | |
| | | | | pavadinimas@lt | text | | 4 | | |
| | | JuridinisAsmuo | | | | | 4 | | |
| | | | Imone | | | kodas | 4 | | |
| | | | | id | inte-<br>ger | | 4 | | |
| | | | | pavadinimas@lt | text | | 4 | | |
| | | | | rusis | inte-<br>ger | | 4 | | |
| | | | | | enum | | | 1 | Juridi-<br>nis |
| | | | | | | | | 2 | Fizinis |
| | | / | | | | | | | |
| | | | Filialas | | | id | 4 | | |
| | | | | id | inte-<br>ger | | 4 | | |
| | | | | ikurta | date | D | 4 | | |
| | | | | atstumas | inte-<br>ger | km | 4 | | |
| | | | | imone | ref | Imo-<br>ne | 4 | | |
| | | | | imone.id | | | 4 | | |
| | | | | imo-<br>ne.pavadinimas@lt | | | 4 | | |
| | | | | tel_nr | string | | 4 | | |

#### L401: Nesusieta su standartiniu žodynu

Ketvirtu brandos lygiu žimimi duomenys, kurie nėra susieti su standartiniais žodynais ar ontologijomis. Siejimas su žodynais atliekamas `model.uri` ir `property.uri` stulpeluose.

### 2.9.6 L500: Trūkumų nėra

Modeliai iš įstaigų duomenų rinkinių vardų erdvės susieti su modeliais iš standartų vardų erdvės, užpildyta *`base`* eilutė. Standartų vardų erdvėje esantiems *modeliams* ir jų *savybėms* užpildytas *`uri`* stulpelis.

Daugiau apie vardų erdves skaitykite skyrelyje: vardu-erdves.

Plačiau apie brandos lygio kėlimą skaitykite skyriuje to-level-5.

Penkto brandos lygio duomenys yra lygiai tokie patys, kaip ir ketvirto brandos lygio, tačiau penktame brandos lygyje, duomenys yra praturtinami metaduomenimis, pateikiant nuorodas į išorinius žodynus arba bent jau pateikiant aiškius pavadinimus ir aprašymus, užpildant `title` ir `description` stulpelius.

Penktame brandos lygyje visas dėmesys yra sutelkiamas į semantinę duomenų prasmę.

**Pavyzdžiai**

| Imone_id | id | pavadinimas@lt | rusis |
|---|---|---|---|
| 26510da5-f6a6-45b0-a9b9-27b3d0090a58 | 42 | UAB "Įmonė" | 1 |

| Filialas_id | id | ikurta | atstumas | imone._id | imone.id | imone.pavadini | tel_nr |
|---|---|---|---|---|---|---|---|
| 63161bd2-158f-4d62-9804-636573abb9c7 | 1 | 2021-09-01 | 1 | 26510da5-f6a6-45b0-a9b9-27b3d0090a58 | 42 | UAB "Įmonė" | tel:+37011111111 |
| 65ec7208-fb97-41a8-9cfc-dfedd197ced6 | 2 | 2021-09-02 | 2 | 26510da5-f6a6-45b0-a9b9-27b3d0090a58 | 42 | UAB "Įmonė" | tel:+37022222222 |
| 2b8cdfa6-1396-431a-851c-c7c6eb7aa433 | 3 | 2021-09-03 | 3 | 26510da5-f6a6-45b0-a9b9-27b3d0090a58 | 42 | UAB "Įmonė" | tel:+37033333333 |
| 1882bb9e-73ee-4057-b04d-d4af47f0aae8 | 4 | 2021-09-04 | 4 | 26510da5-f6a6-45b0-a9b9-27b3d0090a58 | 42 | UAB "Įmonė" | tel:+37044444444 |

| Struktūros aprašas | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| d | r | b | m | property | type | ref | level | uri | prepare | title |
| example | | | | | | | | | | |
| | | | | | prefix | foaf | | http://xmlns.com/ | | |
| | | | | | | dct | | http://purl.org/dc/ | | |
| | | | | | | schema | | http://schema.org | | |
| | | | JuridinisAsmuo | | | kodas | 4 | | | |
| | | | | kodas | integer | | 4 | | | |
| | | | | pavadinimas@lt | text | | 4 | | | |
| | | JuridinisAsmuo | | | | | 4 | | | |
| | | | Imone | | | id | 5 | foaf:Organization | | |
| | | | | id | | | 5 | dct:identifier | | |
| | | | | pavadinimas@lt | | | 5 | dct:title | | |
| | | | | rusis | integer | | 4 | | | |
| | | | | | enum | | | | 1 | Juridinis |
| | | | | | | | | | 2 | Fizinis |
| | | / | | | | | | | | |
| | | | Filialas | | | id | 5 | schema:LocalBusiness | | |
| | | | | id | date | 1D | 5 | dct:identifier | | |
| | | | | ikurta | date | 1D | 5 | dct:created | | |
| | | | | atstumas | integer | km | 5 | schema:distance | | |
| | | | | imone | ref | Imone | 5 | foaf:Organization | | |
| | | | | imone.id | integer | | 5 | dct:identifier | | |
| | | | | imone.pavadinimas | text | | 5 | dct:title | | |
| | | | | tel_nr | string | | 5 | foaf:phone | | |

## 2.10 Prieigos lygiai

Duomenų prieigos lygis nurodomas *access* stulpelyje.

**access**

**open**

**Atviri duomenys**

Duomenys skirti viešam naudojimui, neribojant panaudojimo tikslo, pagal vieną iš atvirų duomenų licencijų.

**public**

**Ribotos prieigos duomenys**

Duomenys skirti viešam naudojimui, tiek privačiame tiek viešąjame sektoriuje, pagal duomenų teikimo sutartį.

**protected**

**Valstybinio sektoriaus duomenys**

Duomenys skirtin naudoti tik tarp valstybinio sektoriaus institucijų, neteikiami privačiam sektoriui.

**private**

**Vidiniai duomenys**

Duomenys skirti tik vidiniam konkrečios sistemos naudojimui, neteikiami už vienos sistemos ribų.

Viešam pakartotiniam naudojimui gali būti teikiami tik `public` ir `open` prieigos lygio duomenys.

`public` duomenys gali būti teikiami tik autorizuotiems duomenų valdytojams, kurie yra susipažinę ir sutinka su duomenų naudojimo taisyklėmis ir naudoja duomenis tik nurodytu tikslu (*purpose limitation*), laikosi BDAR reikalavimų. Asmens duomenys gali būti viešinami tik public ar žemesniu prieigos lygiu.

`open` duomenys turėtu būti teikiami atvirai be jokios autorizacijos ir neribojant duomenų naudojimo tikslo. Asmens duomenys negali būti teikiami `open` prieigos lygiu.

Prieigos lygiai gali būti paveldimi iš aukštesnės dimensijos. Tačiau žemesnė dimensija apsprendžia realų prieigos lygį. Pavyzdžiui jei *dataset.access* yra `private`, o toje *dataset* dimensijoje esantis *property* yra `open`, tada visos to *property* aukštesnės dimensijos taip pat tampa `open`, nors visos kitos dimensijos yra `private`, nes paveldi *dataset.access* reikšmę.

## 2.11 Duomenų šaltiniai

### 2.11.1 SQL

**resource.source**

Duomenų bazės URI. Duomenų bazės URI formuojamas naudojant tokį ABNF šabloną:

```
uri = type ["+" driver] "://"
      [user [":" password] "@"]
      host [":" port]
      "/" database ["?" params]
```

Šablone naudojamų kintamųjų aprašymas:

**type**
: Duomenų bazių serverio pavadinimas:

  **sqlite**

  **postgresql**

  **mysql**

  **oracle**

  **mssql**

**driver**
: Konkretaus duomenų bazių serverio tvarkyklė naudojama komunikacijai su duomenų baze.

**user**
: Naudotojo vardas jungimuisi prie duomenų bazės.

**password**
: Duomenų bazės naudotojo slaptažodis.

**host**
: Duomenų bazių serverio adresas.

**port**
: Duomenų bazių serverio prievadas.

**database**
: Konkrečios duomenų bazės pavadinimas.

**params**
: Papildomi parametrai `Query string` formatu.

**resource.prepare**
: Formulė skirta papildomiems veiksmams reikalingiems ryšiui su duomenų baze užmegzti ir duomenų bazės paruošimui, kad būtų galima skaityt duomenis.

**resource.type**
: Galimos reikšmės: `sql`.

**resource.prepare**

**connect**(*dsn*, *schema: str = None*, *encoding: str = 'utf-8'*)

**Parametrai**

- **dsn** -- Duomenų bazės URI, kaip nurodyta *resource.source*.
- **schema** -- Duomenų bazės schema.
- **encoding** -- Duomenų bazės koduotė.

Naudojama tais atvejais, kai jungiantis prie duomenų bazės reikia perduoti papildomus parametrus.

**model.source**
Duomenų bazėje esančios lentelės pavadinimas.

**property.source**
Lentelės stulpelio pavadinimas.

### 2.11.2 CSV

**resource.type**
Galimos reikšmės: `csv`, `tsv`.

**resource.source**
Žiūrėti *Failai*.

**resource.prepare**

**tabular**(*sep:* ',')
Nurodoma kaip CSV faile atskirti stulpeliai. Pagal nutylėjimą `separator` reikšmė yra `,`.

**model.source**
Nenaudojama, kadangi CSV resursas gali turėti tik vieną lentelę.

**model.prepare**
Žiūrėti *Stulpeliai lentelėje*.

**property.source**
Žiūrėti *Stulpeliai lentelėje*.

### 2.11.3 JSON

**resource.type**
Galimos reikšmės: `json`, `jsonl`.

**resource.source**
Žiūrėti *Failai*.

**model.source**
JSON objekto savybės pavadinimas, kuri rodo į masyvą reikšmių, kurios bus naudojamos kaip modelio duomenų eilutės. Kiekvienas masyvo elementas atskirai aprašomas *property* dimensijoje. Jei JSON objektas yra kompleksinis žiūrėti *Kompleksinės struktūros*.

**property.source**
JSON objekto savybė, kurioje pateikiami aprašomo stulpelio duomenys.

**property.prepare**
Žiūrėti *Kompleksinės struktūros*.

### 2.11.4 XML

**`resource.type`**
Galimos reikšmės: `xml`, `html`.

**`resource.source`**
Žiūrėti *Failai*.

**`model.source`**
XPath iki elementų sąrašo kuriame yra modelio duomenys.

**`model.prepare`**
Jei neužpildyta, vykdoma `xpath(self)` funkcija.

**`xpath`**(*expr*)
Vykdo nurodyta `expr`, viso XML dokumento kontekste.

**`property.source`**
XPath iki elemento kuriame yra duomenys.

XPath nurodomas reliatyvus modeliui, arba kitai daugiareikšmei savybei, kurios sudėtyje savybė yra. Daugiareikšmės savybės žymimos `[]` simboliais savybės kodiniame pavadinime, įprastai tai yra `array` tipo savybės.

**`model.prepare`**
Jei neužpildyta, vykdoma `xpath(self)` funkcija, iš *`model`* gauto elemento kontekste.

**Pavyzdys**

```
<countries>
    <country id="1" name="Lithuania">
        <cities>
            <city id="10" name="Vilnius">
                <streets>
                    <street id="100">Gedimino st.</street>
                    <street id="101">Konstitucijos st.</street>
                </streets>
            </city>
            <city id="11" name="Kaunas">
                <streets>
                    <street id="102">Laisves st.</street>
                    <street id="103">Daukanto st.</street>
                </streets>
            </city>
        </cities>
    </country>
</countries>
```

Pagal aukščiau duotus duomenis ir koncepcinį modelį, struktūros aprašas atrodys taip:

| model | property | type | ref | source |
|---|---|---|---|---|
| **Country** | | | id | **countries/country** |
| | id | integer | | @id |
| | name@en | string | | @name |
| | cities[] | backref | **City** | cities/city |
| | cities[].id | integer | | @id |
| | cities[].name@en | string | | @name |
| | cities[].country | ref | **Country** | ../../@id |
| | cities[].streets[] | backref | **Street** | streets/street |
| | cities[].streets[].id | integer | | @id |
| | cities[].streets[].name@e | string | | @name |
| | cities[].streets[].city | ref | **City** | ../../@id |
| **City** | | | id | **countries/country/cities/city** |
| | id | integer | | @id |
| | name@en | string | | @name |
| | country | ref | **Country** | ../../@id |
| **Street** | | | id | **countries/country/cities/city/streets/street** |
| | id | integer | | @id |
| | name@en | string | | @name |
| | country | ref | **City** | ../../@id |

Struktūros apraše matome du variantus, kaip gali būti aprašomi duomenys. Pirmu atveju `Country` modelyje naudojama objektų kompozicija, kur vieno `Country` objekto apimtyje, pateikiami ir kiti objektai.

Reikia atkreipti dėmesį, kad savybės esančios kitos daugiareikšmės savybės sudėtyje, *`property.source`* stulpelyje nurodo XPath išraišką reliatyvią daugereikšmei savybei. Daugiareikšmės savybės žymimos `[]` žyme.

Pavyzdyje `cities[].id` *`property.source`* stulpelyje nurodo `@id`, kuris yra reliatyvus `cities[]` savybės `streets/street` atžvilgiu.

Pagal struktūros aprašą pateiktą aukščiau, kreipiantis į `/Country`, gausime tokius UDTS specifikaciją atitinkančius duomenis:

```
{
    "_type": "Country",
    "_id": "29df0534-389d-4eac-a048-799ac64d5103",
    "id": 1,
    "name": {"en": "Lithuana"},
    "cities": [
        {
            "_type": "City",
            "_id": "4a7a3214-e6c3-4a5b-99a8-04be88eac3d4",
            "id": 10,
            "name": {"en": "Vilnius"},
            "country": {
                "_type": "Country",
                "_id": "29df0534-389d-4eac-a048-799ac64d5103"
            },
            "streets": [
                {
                    "_type": "Street",
                    "_id": "c1380514-549f-4cdd-b258-6fecc3a5bbda",
                    "id": 100,
                    "name": {"en": "Gedimino st."},
                    "city": {
                        "_type": "City",
                        "_id": "4a7a3214-e6c3-4a5b-99a8-04be88eac3d4"
                    },
                },
                {
                    "_type": "Street",
                    "_id": "5c02f700-6478-43a0-a147-959927cb3c1c",
                    "id": 101,
                    "name": {"en": "Konstitucijos st."},
                    "city": {
                        "_type": "City",
                        "_id": "4a7a3214-e6c3-4a5b-99a8-04be88eac3d4"
                    },
                }
            ]
        },
        {
            "_type": "City",
            "_id": "0fee7d9a-6827-4931-bbea-d44d197faef2",
            "id": 11,
            "name": {"en": "Kaunas"},
            "country": {
                "_type": "Country",
                "_id": "29df0534-389d-4eac-a048-799ac64d5103"
            },
            "streets": [
                {
                    "_type": "Street",
                    "_id": "399a37d6-63a7-43a4-82de-d3d5c75f5d02",
                    "id": 102,
                    "name": {"en": "Laisves st."},
                    "city": {
                        "_type": "City",
                        "_id": "0fee7d9a-6827-4931-bbea-d44d197faef2"
                    },
```


(continues on next page)

(tęsinys iš praeito puslapio)

```
                },
                {
                    "_type": "Street",
                    "_id": "5b04fecd-5fff-48f6-8674-7cc6da840281",
                    "id": 103,
                    "name": {"en": "Daukanto st."},
                    "city": {
                        "_type": "City",
                        "_id": "0fee7d9a-6827-4931-bbea-d44d197faef2"
                    },
                }
            ]
        }
    ]
}
```

Analogiškai, jei kreiptumėmės į `/Street`, gautume visas gatves iš visų miestų:

```
{
    "_data": [
        {
            "_type": "Street",
            "_id": "c1380514-549f-4cdd-b258-6fecc3a5bbda",
            "id": 100,
            "name": {"en": "Gedimino st."},
            "city": {
                "_type": "City",
                "_id": "4a7a3214-e6c3-4a5b-99a8-04be88eac3d4"
            },
        },
        {
            "_type": "Street",
            "_id": "5c02f700-6478-43a0-a147-959927cb3c1c",
            "id": 101,
            "name": {"en": "Konstitucijos st."},
            "city": {
                "_type": "City",
                "_id": "4a7a3214-e6c3-4a5b-99a8-04be88eac3d4"
            },
        },
        {
            "_type": "Street",
            "_id": "399a37d6-63a7-43a4-82de-d3d5c75f5d02",
            "id": 102,
            "name": {"en": "Laisves st."},
            "city": {
                "_type": "City",
                "_id": "0fee7d9a-6827-4931-bbea-d44d197faef2"
            },
        },
        {
            "_type": "Street",
            "_id": "5b04fecd-5fff-48f6-8674-7cc6da840281",
            "id": 103,
            "name": {"en": "Daukanto st."},
            "city": {
```

(continues on next page)

(tęsinys iš praeito puslapio)

```
                "_type": "City",
                "_id": "0fee7d9a-6827-4931-bbea-d44d197faef2"
            },
        }
    ]
}
```

### 2.11.5 XLSX

**`resource.type`**
: Galimos reikšmės: `xlsx`, `xls` arba `odt`.

**`resource.source`**
: Žiūrėti *Failai*.

**`model.source`**
: Skaičiuoklės faile esančio lapo pavadinimas.

**`model.prepare`**
: Žiūrėti *Stulpeliai lentelėje*.

**`property.source`**
: Žiūrėti *Stulpeliai lentelėje*.

## 2.12 Vardų erdvės

*`dataset`* ir *`model`* esantys pavadinimai turi būti globaliai (Lietuvos mastu) unikalūs. Kad užtikrinti pavadinimų unikalumą *`dataset`* ir *`model`* pavadinimai formuojami pasitelkiant vardų erdves.

**`/{vocabulary}/`**
: **Standartinių žodynų vardų erdvė**

  Standartinių žodynų vardų erdvė formuojama egzistuojančių standartų ir išorinių žodynų pagrindu suteikiant vardų erdvei `{vocabulary}` žodyno sutrumpintą pavadinimą. Pavyzdžiui duomenų katalogo metaduomenų žodynas DCAT turėtų keliauti į `/dcat/` vardų erdvę. Standartų sutrumpintus pavadinimus rekomenduojame imti iš Linked Open Vocabularies katalogo.

**`/datasets/{form}/{org}/`**
: **Įstaigų vardų erdvė**

  Konkrečios organizacijos vietinė rinkinio vardų erdvė. Rekomenduojama `{org}` reikšmei naudoti organizacijos trumpinį, kad bendras modelio pavadinimas nebūtų per daug ilgas.

  Galimos `{form}` reikšmės:

| | |
|---|---|
| `gov` | Valstybinės įstaigos. |
| `com` | Verslo įmonės. |

`/datasets/{form}/{org}/{catalog}/`

**Informacinių sistemų vardų erdvė**

Informacinės sistemos vardų erdvė, kuri teikia duomenų rinkinius.

`/datasets/{form}/{org}/{catalog}/{dataset}/`

**Įstaigų duomenų rinkinių vardų erdvė**

Įstaigos duomenų rinkinio vardų erdvė į kurią patenka visi įstaigos duomenų modeliai.

Naujai atveriami *duomenų struktūros aprašai* sudaromi *ŠDSA* pagrindu. Įprastai duomenų bazių struktūra nėra kuriama vadovaujantis standartais. Vidinės struktūros dažniausiai kuriamos vadovaujantis sistemai keliamais reikalavimais. Todėl naujai atveriamų duomenų rinkiniai turi keliauti į duomenų rinkinio vardų erdvę `/datasets/{form}/{org}/{catalog}/{dataset}/`, išlaikant pirminę duomenų struktūrą ir neprarandant duomenų.

Tačiau su laiku, dalis įstaigos duomenų iš duomenų rinkinio vardų erdvės turėtu būti perkeliami į globalią duomenų erdvę. Į globalią duomenų erdvę pirmiausiai turėtų būti perkeliami tie duomenys, kurie yra plačiai naudojami. Perkėlimas į globalią duomenų erdvę nepanaikina duomenų rinkinio iš ankstesnės vardų erdvės, tiesiog duomenų rinkinio duomenų pagrindu kuriama kopija į globalią duomenų erdvę.

### 2.12.1 Reliatyvūs pavadinimai

Modelio pavadinimas gali būti absoliutus arba reliatyvus. Absoliutūs pavadinimai prasideda `/` simboliu, reliatyvūs pavadinimai prasideda be `/` simbolio ir yra jungiami su vardų erdvės pavadinimu, kurios kontekste yra apibrėžtas modelis.

Pavyzdžiui, turinti tokį duomenų struktūros aprašą:

| id | d | r | b | m | property | type |
|---|---|---|---|---|---|---|
| 1 | **dcat** | | | | | ns |
| 2 | | | | **dataset** | | |
| 3 | | | | | title | |
| 4 | **datasets/gov/ivpk/adk** | | | | | |
| 5 | | adk | | | | |
| 6 | | | **/dcat/dataset** | | | alias |
| 7 | | | | **dataset** | | |
| 8 | | | | | title | |

Matome, kad yra apibrėžti du modeliai:

- `dcat/dataset`
- `datasets/gov/ivpk/adk/dataset`

Vienas `dataset` modelis yra apibrėžtas `dcat` vardų erdvės kontekste, kitas `datasets/gov/ivpk/adk` vardų erdvės kontekste.

Kai modelio pavadinimas yra naudojamas vardų erdvės kontekste ir pavadinimas neprasideda `/` simboliu, tada tai yra reliatyvus modelio pavadinimas. Reliatyvus modelio pavadinimas yra jungiamas su vardų erdvės pavadinimu, kurios kontekste yra apibrėžtas modelis.

Jei tam tikros vardų erdvės kontekste norime įvardinti modelį, kuris yra už tos vardų erdvės konteksto ribų, būtina naudoti absoliutų modelio pavadinimą, kuris prasideda `/` simboliu.

Taip yra padaryta 6-oje eilutėje, kur nurodyta, kad `datasets/gov/ivpk/adk/dataset` bazė yra `dcat/dataset` modelis iš kitos vardų erdvės.

Visais atvejais, kai modelio pavadinimas naudojamas nenurodant jokio vardų erdvės konteksto, `/` simbolio pavadinimo pradžioje naudoti nereikia. Pavyzdžiui šiame tekste įvardinti `dcat/dataset` ir `datasets/gov/ivpk/adk/dataset` modelių pavadinimai neprasideda `/` simboliu.

## 2.13 Išoriniai žodynai

Išorinių žodynų pagalba, galima susieti aprašomus duomenis su išoriniais žodynais. Susiejimas atliekamas *`model.uri`* ir *`property.uri`*, naudojant išorinių žodynų URI prefiksus.

Pavyzdžiui turint tokį duomenų šaltinį:

| country | |
|---|---|
| id | name |
| 1 | Lietuva |

| city | | |
|---|---|---|
| id | name | country |
| 1 | Vilnius | 1 |

Ir šį šaltinį atitinkančią *DSA* lentelę:

| id | d | r | b | m | property | type | ref | uri |
|---|---|---|---|---|---|---|---|---|
| | | | | | | prefix | locn | http://www.w3.org/ns/locn# |
| | | | | | | | dbpedia-owl | http://dbpedia.org/ontology/ |
| | datasets/example/geo | | | | | | | |
| | | salys | | | | sql | | |
| | | | | Country | | | id | locn:Location |
| | | | | | id | integer | | dct:identifier |
| | | | | | name@lt | text | | locn:geographicName |
| | | | | | capital | ref | City | dbpedia-owl:capital |
| | | | | City | | | id | locn:Location |
| | | | | | id | integer | | dct:identifier |
| | | | | | name@lt | text | | locn:geographicName |
| | | | | | country | ref | Country | dbpedia-owl:country |

Jei *`property`* pavadinimai turi `@` žymes, tada generuojant RDF, prie reikšmės pridedama atitinkama kalbos žymė.

Galima duomenis eksportuoti RDF Turtle formatu, kas atrodytų taip:

```
@base <https://get.data.gov.lt/> .
@prefix dct: <http://purl.org/dc/terms/> .
@prefix locn: <http://www.w3.org/ns/locn#> .
@prefix dbpedia-owl: <http://dbpedia.org/ontology/> .

<datasets/example/geo/Country/eb09946c-26e1-4698-a298-7bb1e468b165>
    a locn:Location ;
    dct:identifier 1 ;
    locn:geographicName "Lietuva"@lt ;
    dbpedia-owl:capital <datasets/example/geo/City/b54c21f6-08b8-4bdd-b785-
↪be1cb2e93a98> .

<datasets/example/geo/City/b54c21f6-08b8-4bdd-b785-be1cb2e93a98>
    a locn:Location ;
    dct:identifier 1 ;
    locn:geographicName "Vilnius"@lt ;
    dbpedia-owl:country <datasets/example/geo/Country/eb09946c-26e1-4698-a298-
↪7bb1e468b165> .
```

Analogiškai, tie patys duomenys gali būti eksportuojami RDF/XML formatu:

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF
    xml:base="https://get.data.gov.lt/"
    xmlns:dct="http://purl.org/dc/terms/"
    xmlns:locn="http://www.w3.org/ns/locn#"
    xmlns:dbpedia-owl="http://dbpedia.org/ontology/"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">

    <locn:Location
        rdf:about="datasets/example/geo/Country/eb09946c-26e1-4698-a298-7bb1e468b165"
        dct:identifier="1">
        <locn:geographicName xml:lang="lt">Lietuva</locn:geographicName>
        <dbpedia-owl:capital
            rdf:resource="datasets/example/geo/City/b54c21f6-08b8-4bdd-b785-
↪be1cb2e93a98" />
    </locn:Location>

    <locn:Location
        rdf:about="datasets/example/geo/City/b54c21f6-08b8-4bdd-b785-be1cb2e93a98"
        dct:identifier="1">
        <locn:geographicName xml:lang="lt">Vilnius</locn:geographicName>
        <dbpedia-owl:country
            rdf:resource="datasets/example/geo/Country/eb09946c-26e1-4698-a298-
↪7bb1e468b165" />
    </locn:Location>
</rdf:RDF>
```

Išoriniai žodynai suteikia galimybę eksportuoti duomenis *RDF* formatu.

Jei struktūros apraše nėra užpildytas *uri* stulpelis, tada, turėtu būti generuojamas tokie RDF duomenys:

```
@base <https://get.data.gov.lt/> .
@prefix dct: <http://purl.org/dc/terms/> .
@prefix locn: <http://www.w3.org/ns/locn#> .
@prefix dbpedia-owl: <http://dbpedia.org/ontology/> .
```

(continues on next page)

(tęsinys iš praeito puslapio)

```
<datasets/example/geo/Country/eb09946c-26e1-4698-a298-7bb1e468b165>
    a <datasets/example/geo/Country> ;
    <datasets/example/geo/Country/id> 1 ;
    <datasets/example/geo/Country/name> "Lietuva"@lt ;
    <datasets/example/geo/Country/capital> <datasets/example/geo/City/b54c21f6-08b8-
↪4bdd-b785-be1cb2e93a98> .

<datasets/example/geo/City/b54c21f6-08b8-4bdd-b785-be1cb2e93a98>
    a <datasets/example/geo/City> ;
    <datasets/example/geo/City/id> 1 ;
    <datasets/example/geo/City/name> "Vilnius"@lt ;
    <datasets/example/geo/City/country> <datasets/example/geo/Country/eb09946c-26e1-
↪4698-a298-7bb1e468b165> .
```

### 2.13.1 Subjekto URI

Pagal nutylėjimą *subjekto* URI yra automatiškai generuojamas ir atrodo taip:

```
https://get.data.gov.lt/datasets/example/geo/Country/eb09946c-26e1-4698-a298-
↪7bb1e468b165
```

Tačiau naudojant kontroliuojamus žodynus, galima nurodyti kitą identifikatorių tokiu būdu:

| m | property | type | ref | uri |
|---|---|---|---|---|
| Country | | | id | locn:Location |
| | id | integer | | |
| | uri | uri | | locn:Location |
| | name@lt | text | | |

Jei *property.uri* sutampa su *model.uri* ir *property.type* yra *uri*, tada formuojant duomenis RDF formatu naudojame ne generuotą subjekto URI, o naudojame lauko reikšmę, kurio *property.uri* sutampa su *model.uri*.

Gali būti ne daugiau kaip vienas *property.uri* su *property.type uri*, kuris sutampa su *model.uri*.

Jei yra keli *uri* tipo laukai, kurie identifikuoja tą patį subjektą, tada kitiems atvejams reikia naudoti ne *model.uri*, o owl:sameAs.

Jei `uri` reikšmė bus `https://sws.geonames.org/597427/`, tada gautume tokius RDF duomenis:

```
@base <https://get.data.gov.lt/example/> .
@prefix locn: <http://www.w3.org/ns/locn#> .

<https://sws.geonames.org/597427/>
    a locn:Location ;
    <Country/id> 1 ;
    <Country/name> "Lietuva"@lt .
```

Atkreipkite dėmesį, kad pats `uri` laukas nėra įtrautkas į RDF duomenis.

Analogiškai, jei *`ref`* tipo laukas rodo į modelį, kurio *`model.uri`* sutampa su *`property.uri`* kuris yra *`ref`* tipo, tada *`ref`* lauko reikšmė taip pat įgyja ne generuotą URI, o URI iš duomenų.

Pratęsiant tą patį pavyzdį:

| m | property | type | ref | uri |
|---|---|---|---|---|
| Country | | | id | locn:Location |
| | id | integer | | |
| | uri | uri | | locn:Location |
| | name@lt | text | | |
| City | | | id | locn:Location |
| | id | integer | | |
| | name@lt | text | | |
| | country | ref | Country | locn:Location |

Gautume tokius duomenis:

```
@base <https://get.data.gov.lt/example/> .
@prefix locn: <http://www.w3.org/ns/locn#> .

<https://sws.geonames.org/597427/>
    a locn:Location ;
    <Country/id> 1 ;
    <Country/name> "Lietuva"@lt .

<City/b54c21f6-08b8-4bdd-b785-be1cb2e93a98>
    a locn:Location ;
    <City/id> 1 ;
    <City/name> "Vilnius"@lt ;
    <City/country> <https://sws.geonames.org/597427/> .
```

## 2.14 Formulės

Formulės rašomos *`prepare`* stulpelyje. Formulių pagalba galima atlikti įvairius duomenų transformavimo, nuasmeninimo, filtravimo ir kokybės tikrinimo veiksmus.

Kadangi yra labai didelė įvairovė duomenų formatų ir duomenų valdymo mechanizmų, siekiant suvaldyti visą šią įvairovę *DSA* formulės leidžia vieningai aprašyti veiksmus su duomenimis. Vėliau formulės verčiamos į vieningą AST, kurį gali interpretuoti automatizuotos priemonės, priklausomai nuo duomenų šaltinio ir konteksto ir DSA sluoksnio.

### 2.14.1 Gramatika

Formulių sintaksė atitinką šią ABNF gramatiką:

```
formula     = testlist
testlist    = test *("," test) *1","
test        = or
or          = and *("|" and)
and         = not *("&" not)
not         = "!" not / comp
comp        = expr *(compop expr)
expr        = term *(termop term)
term        = factor *(factorop factor)
factor      = sign factor / composition
composition = atom *trailer
atom        = "(" *1group ")"
            / "[" *1list "]"
            / func / value / name
group       = test *("," test) *1","
list        = test *("," test) *1","
trailer     = "[" *1filter "]" / method / attr
func        = name call
method      = "." name call
call        = "(" *1arglist ")"
arglist     = argument *("," argument) *1","
argument    = test / kwarg
kwarg       = name ":" test
filter      = test *("," test) *1","
attr        = "." name
value       = null / bool / integer / number / string / star
compop      = ">=" / "<=" / "!=" / "=" / "<" / ">"
termpop     = "+" / "-"
factorop    = "*" / "/" / "%"
sign        = "+" / "-"
star        = "*"
name        = ~/[a-z_][a-z0-9_]*/i
number      = ~/\d+(\.\d+)?/
integer     = ~/0|[1-9]\d*/
bool        = "false" / "true"
null        = "null"
string      = ~/(?!"").*?(?<!\\)(\\\\)*?"|'(?!'').*?
                (?<!\\)(\\\\)*?'/i
```

### 2.14.2 Sintaksės medis

Formulės verčiamos į vieningą abstraktų sintaksės medį. Vieningas abstraktus sintaksės medis leidžia atskirti formulės skaitymo ir interpretavimo veiklas.

Abstraktus sintaksės medis sudarytas iš vienodų elementų turinčių tokias savybes:

**`name`**
: Funkcijos pavadinimas.

**`args`**
: Funkcijos argumentų sąrašas, kurį gali sudaryti konkrečios reikšmės ar kiti medžio elementai, veiksmai.

Visos formulėje naudojamos išraiškos sintaksės medyje verčiamos į funkcijų ir argumentų medį. Pavyzdžiui `test("a", "b")` bus verčiamas į:

```
{
    "name": "test",
    "args": ["a", "b"],
}
```

### 2.14.3 Funkcijų iškvietimas

Formulės susideda iš vykdomų funkcijų sekos. Pavyzdžiui funkcijos pavadinimu `test` vykdymas formulėje atrodys taip:

```
test()
```

Aukščiau pavyzdyje pateikta formulė vykdo funkciją `test`, be argumentų. Tačiau funkcijos gali turėti pozicinius ir vardinius argumentus.

### 2.14.4 Poziciniai argumentai

Poziciniai argumentai perduodami taip:

```
test(a, b, c)
```

Pavyzdyje, funkcijai `test` perduodami trys argumentai `a`, `b` ir `c`. Šioje dokumentacijoje, tais atvejais, kai funkcijos pozicinių argumentų skaičius nėra fiksuotas, naudojama *`args` išraiška, kur * nurodo, kad pozicinių argumentų gali būti 0 ar daugiau.

### 2.14.5 Vardiniai argumentai

Vardiniai argumentai funkcijai perduodami taip:

```
test(a: 1, b: 2, c: 3)
```

Pozicinius argumentus būtina perduoti tiksliai tokia tvarka, kokios tikisi funkcija. Tačiau vardinius argumentus, galima perduoti, bet kuria tvarka.

Jei vardinių argumentų sąrašas nėra fiksuotas, dokumentacijoje toks argumentų sąrašas užrašomas **`kwargs` forma, kur ** nurodo, kad vardinių argumentų gali būti 0 ar daugiau.

### 2.14.6 Alternatyvus funkcijos iškvietimas

Funkcijų iškvietimas gali būti užrašomas įprastiniu būdu, pavyzdžiui:

```
test(test(test(a), b), c)
```

Arba funkcijų grandinės (angl. Method chain) būdu:

```
a.test().test(b).test(c)
```

Kadangi formulės dažnai naudojamos tam tikros reikšmės transformavimui, todėl dažnai formulė yra lengviau skaitoma, naudojant funkcijų grandinę.

`test(a)` yra `a.test()` arba `test(a, b)` ir `a.test(b)` yra ekvivalentūs (UFCS).

### 2.14.7 Standartinės funkcijos

Priklausomai nuo duomenų šaltinio ar konteksto gali būti naudojami skirtingi veiksmai, tačiau žemiau yra pateikti bendrosios paskirties veiksmai:

`func.`**`bind`**`(`*name*`)`

Rodo į reikšmę pavadinimu `name` iš konteksto. Reikšmės ieškoma tokia tvarka:

- *`var()`*
- *`param()`*
- *`item()`*
- *`prop()`*

`func.`**`prop`**`(`*name*`)`

Modelio savybė pavadinimu `name` iš *`property`* stulpelio.

`func.`**`item`**`(`*name*`)`

Sąrašo elemento savybė pavadinimu `name`.

`func.`**`param`**`(`*name*`)`

Parametras pavadinimu `name`. Žiūrėti *`param`*.

`func.`**`var`**`(`*name*`)`

Kintamasis apibrėžtas *`set()`* funkcijos pagalba.

`func.`**`self`**`()`

Rodo į aktyvią reikšmę, naudojamas *`property.prepare`* formulėse.

`func.`**`or`**`(`**args*`)`

Taip pat galima naudoti tokia išraiška:

```
a | b | c
```

Grąžiną pirmą netuščią reikšmę. Pirmoji netuščia reikšmė nutraukia sekančių `args` argumentų interpretavimą.

`func.`**`and`**`(`**args*`)`

Taip pat galima naudoti tokia išraiška:

```
a & b & c
```

Grąžina pirmą tuščią reikšmę arba paskutinę reikšmę, jei prieš tai esančios reikšmės netuščios.

`func.`**`not`**`(`*arg*`)`

Taip pat galima naudoti tokia išraiška:

```
!arg
```

Jei `arg` tuščia grąžina `true`, priešingu atveju `false`.

`func.`**`eq`**(*a*, *b*)

Taip pat galima naudoti tokia išraiška:

```
a = b
```

a lygus b.

`func.`**`ne`**(*a*, *b*)

Taip pat galima naudoti tokia išraiška:

```
a != b
```

a nelygus b.

`func.`**`lt`**(*a*, *b*)

Taip pat galima naudoti tokia išraiška:

```
a < b
```

a mažiau už b.

`func.`**`le`**(*a*, *b*)

Taip pat galima naudoti tokia išraiška:

```
a <= b
```

a mažiau arba lygu už b.

`func.`**`gt`**(*a*, *b*)

Taip pat galima naudoti tokia išraiška:

```
a > b
```

a daugiau už b.

`func.`**`ge`**(*a*, *b*)

Taip pat galima naudoti tokia išraiška:

```
a >= b
```

a daugiau arba lygu už b.

`func.`**`add`**(*a*, *b*)

Taip pat galima naudoti tokia išraiška:

```
a + b
```

a ir b suma.

`func.`**`sub`**(*a*, *b*)

Taip pat galima naudoti tokia išraiška:

```
a - b
```

a ir b skirtumas.

`func.`**`mul`**(*a*, *b*)

Taip pat galima naudoti tokia išraiška:

```
a * b
```

`a` ir `b` sandauga.

`func.`**`div`**(*a*, *b*)

Taip pat galima naudoti tokia išraiška:

```
a / b
```

`a` ir `b` dalyba.

`func.`**`mod`**(*a*, *b*)

Taip pat galima naudoti tokia išraiška:

```
a % b
```

`a` ir `b` modulis.

`func.`**`positive`**(*a*)

Taip pat galima naudoti tokia išraiška:

```
+a
```

Gali būti interpretuojamas skirtingai, priklausomai nuo konteksto. Įprastiniu atveju keičia skaičiaus ženklą.

`func.`**`negative`**(*a*)

Taip pat galima naudoti tokia išraiška:

```
-a
```

Gali būti interpretuojamas skirtingai, priklausomai nuo konteksto. Įprastiniu atveju keičia skaičiaus ženklą.

`func.`**`tuple`**(**args*)

Taip pat galima naudoti tokia išraiška:

```
(*args)
```

Grupė argumentų.

**()**
: Tuščia grupė.

**a, b**
: Tas pats, kas `tuple(a, b)`.

`func.`**`list`**(**args*)

Taip pat galima naudoti tokia išraiška:

```
[*args]
```

Sąrašas reikšmių.

`func.`**`getattr`**(*object*, *attr*)

Taip pat galima naudoti tokia išraiška:

```
object.attr
```

Gaunamos reikšmės pagal atributą arba raktą.

`func.`**`getitem`**(*object*, *item*)

Taip pat galima naudoti tokia išraiška:

```
a[item]
```

Gaunamos reikšmės pagal atributą arba raktą.

*`getitem()`* gali būti interpretuojamas kaip sąrašo reikšmių filtras:

```
a[b > c]
```

`func.`**`dict`**(***kwargs*)

Taip pat galima naudoti tokia išraiška:

```
{a: b}
```

Sudėtinė duomenų struktūra.

`func.`**`set`**(***kwargs*)

Taip pat galima naudoti tokia išraiška:

```
{a, b}
```

Reikšmių aibė.

`func.`**`op`**(*operator*)

Taip pat galima naudoti tokia išraiška:

```
a(*)
```

Operatoriai gali būti naudojami kaip argumentai.

`func.`**`stack`**(*columns*, *values*, *exclude*)

Visus stulpelius išskyrus `exclude` verčia į vieną stulpelių eilutei suteikiant `columns` pavadinimą, o reikšmių stulpeliui `values` pavadinimą. Pavyzdžiui:

| vertinimas | 2015P2 | 2016P2 |
|---|---|---|
| Neigiamai | 0 | 1 |
| Teigiamai | 39 | 28 |

Tokiai lentelei pritaikius `stack("data", "rodiklis", ["vertinimas"])` transformaciją, gausime tokį rezultatą:

| vertinimas | data | rodiklis |
|---|---|---|
| Neigiamai | 2015P2 | 0 |
| Neigiamai | 2016P2 | 1 |
| Teigiamai | 2015P2 | 39 |
| Teigiamai | 2016P2 | 28 |

`func.`**`datetime`**(*str*, *format*)

Išgaunama data ir laikas iš str, naudojant strftime formatą.

`func.`**`date`**(*str*, *format*)

išgaunama data iš str, naudojant strftime formatą.

`func.`**`date`**(*datetime*)

Gražinama data iš datos ir laiko.

### 2.14.8 Failai

Dažnai duomenys teikiami failų pavidalu, kurie gali būti saugomi tiek lokaliai failų sistemoje, tiek nuotoliniame serveryje. Failai gali būti suspausti ir patalpinti į archyvo konteinerius. *DSA* leidžia aprašyti įvairius prieigos prie duomenų, saugomų failuose, atvejus.

**`resource.source`**

Nutolusiame serveryje saugomo failo *URI* arba kelias iki lokalaus katalogo. Lokalaus katalogo kelias gali būti pateikiamas tiek *POSIX*, tiek *DOS* formatais, priklausomai nuo to, kokioje operacinėje sistemoje failai saugomi.

**`resource.prepare`**

`func.`**`file`**(*resource*, *encoding: 'utf-8'*)

**Parametrai**

- **`resource`** -- Kelias arba URI iki failo.
- **`encoding`** -- Failo koduotę.

Ši funkcija leidžia nurodyti failo koduotę, jei failas yra užkoduotas kita, nei UTF-8 koduote. Pilną palaikomų koduočių sąrašą galite rasti šiame sąraše.

`func.`**`extract`**(*resource*, *type*)

**Parametrai**

- **`resource`** -- Kelias arba URI iki archyvo failo arba failo objektas.
- **`type`** -- Archyvo tipas.

Išpakuoja archyvą, kuriame saugomi failai. Galimos `type` reikšmės:

**`zip`**

**`tar`**

**`rar`**

Funkcijos rezultatas yra archyvo objektas, kuris leidžia pasiekti esančius archyvo failus *`getitem()`* funkcijos pagalba.

`func.`**`decompress`**(*resource*, *type*)

**Parametrai**

- **`resource`** -- Kelias arba URI iki archyvo failo arba failo objektas.
- **`type`** -- Archyvo tipas.

Taikomas srautinis failo glaudinimo filtras. Galimos `type` reikšmės:

**gz**

**bz2**

**xz**

### 2.14.9 Stulpeliai lentelėje

CSV ar skaičiuoklių lentelėse stulpelių pavadinimai pateikiami pačioje lentelėje. Eilutė, kurioje surašyti pavadinimai nebūtinai gali būti pirma. Stulpelių pavadinimai gali būti pateikti keliose eilutėse iš kurių formuojamos kompleksinės struktūros (žiūrėti *Kompleksinės struktūros*). Įvairias situacijas galima aprašyti formulių pagalba.

**model.prepare**

func.**header**(**line*)

**null**

Lentelėje eilučių pavadinimų nėra. Tokiu atveju, *property.source* stulpelyje reikia pateikti stulpelio numerį, pradedant skaičiuoti nuo 0.

**line**

Nurodomas eilutės numeris, pradedant eilutes skaičiuoti nuo 0, kur yra pateikti lentelės stulpelių pavadinimai. Pagal nutylėjimą stulpelių pavadinimų ieškoma pirmoje eilutėje.

***line**

Jei lentelė turi kompleksinę stulpelių struktūrą, tada galima pateikti daugiau nei vieną eilutės numerį iš kurių bus nustatomi stulpelių pavadinimai.

func.**head**(*n*)

Praleisti n einančių po stulpelių pavadinimų eilutės.

func.**tail**(*n*)

Ignoruoti n eilučių failo pabaigoje.

**property.source**

Jei naudojamas *header(null)*, tada nurodomas stulpelio numeris, pradedant nuo 0.

Jei naudojamas *header(line)*, tada nurodomas stulpelio pavadinimas, toks koks įrašytas lentelės line eilutėje.

Jei naudojamas *header(*line)*, tada nurodomas stulpelio pavadinimas, toks koks įrašymas lentelės pirmajame line argumente.

**property.prepare**

Jei naudojamas header(*line), žiūrėti *Kompleksinės struktūros*.

### 2.14.10 Duomenų atranka

Duomenų filtravimui naudojamas *model.prepare* stulpelis, kuriame galima apriboti iš šaltinio skaitomų duomenų imtį.

Tarkime, jei turime tokias dvi duomenų lenteles:

| COUNTRIES | |
|---|---|
| COUNTRY | CODE |
| Lietuva | lt |
| Latvija | lv |

| CITIES | | |
|---|---|---|
| ID | CITY | COUNTRY |
| 1 | Vilnius | lt |
| 2 | Kaunas | lt |
| 3 | Ryga | lv |

Jei norėtume atveri ne visų šalių duomenis, o tik Lietuvos, tada duomenų struktūros aprašas turėtu atrodyti taip:

| d | r | b | m | property | type | ref | source | prepare |
|---|---|---|---|---|---|---|---|---|
| datasets/example/countries | | | | | | | | |
| | salys | | | | sql | | sqlite:// | |
| | | | Country | | | code | COUNTRIES | **code = "lt"** |
| | | | | name | string | | COUNTRY | |
| | | | | code | string | | CODE | |
| | | | City | | | id | CITIES | |
| | | | | id | integer | | ID | |
| | | | | name | string | | CITY | |
| | | | | country | ref | Country | COUNTRY | |

Kaip ir visur, formulės reikia naudoti pavadinimus ne iš *source* stulpelio, o iš *property*, *model* arba *dataset*.

Jei lentelės yra susijusios ryšiais tarpusavyje, užtenka filtrą nurodyti tik vienoje lentelėje, visose kitose susijusios lentelėse filtrai bus taikomi automatiškai, kad užtikrinti duomenų vientisumą.

Nurodant filtrus yra galimybė naudoti ne tik vienos lentelės laukus, bet ir susijusių lentelių laukus, pavyzdžiui yra galimybė nurodyti tokį filtrą:

| d | r | b | m | property | type | ref | source | prepare |
|---|---|---|---|---|---|---|---|---|
| | | | City | | | id | CITIES | **country.code = "lt"** |

Tačiau šiuo atveju, toks filtras būtų perteklinis, nes toks filtras generuojamas automatiškai ir susijusio `Country` modelio, kadangi negalime publikuoti Latvijos miestų, jei publikuojama tik Lietuvos šalis.

Pilnas galimų filtrų sąrašas:

**model.prepare**

**a = b**
: a ir b reikšmės yra lygios.

**a != b**
: a nelygu b.

**a > b**
: a daugiau už b.

**a < b**
: a mažiau už b.

**a >= b**
: a daugiau arba lygu b.

**a <= b**
: a mažiau arba lygu b.

**a.in(b)**
: a lygi bent vienai iš b sekos reikšmių.

**a.notin(b)**
: a nelygi nei vienai iš b sekos reikšmių.

**a.contains(b)**
: a seka savyje turi b seką.

**a.startswith(b)**
: a seka prasideda b seka.

**a.endswith(b)**
: a seka baigiasi b seka.

**a & b**
: a ir b.

**a | b**
: a arba b.

**sort(+a, -b)**
: Rūšiuoti didėjimo tvarka pagal a ir mažėjimo tvarka pagal b.

### 2.14.11 Periodiškumas

Periodiškumui nurodyti naudojamas model.prepare stulpelis, kuriame galima naudoti tokias formules:

**model.prepare**

func.**cron**(*line*)
: Duomenų atnaujinimo laikas, analogiškas cron formatui.

  `line` argumentas aprašomas taip:

**nm**
: n-toji minutė, n ∈ 0-59.

**nh**
: n-toji valanda, n ∈ 0-23.

**nd**
: n-toji mėnesio diena, n ∈ 1-31.

**$d**
: Paskutinė mėnesio diena.

**nM**
: n-tasis mėnuo, n ∈ 1-12.

**nw**
: n-toji savaitės diena, n ∈ 0-6 (sekmadienis-šeštadienis).

**n#iw**
: n-toji savaitės diena, i-toji mėnesio savaitė, i ∈ 1-6.

**n$iw**
: n-toji savaitės diena, i-toji savaitė nuo mėnesio galo, i ∈ 1-6.

**,**
: Kableliu galim atskirt kelias laiko vertes.

**-**
: Brūkšneliu galima atskirti laiko verčių intervalą.

**/**
: Pasvyruoju brūkšniu galima atskirti laiko verčių kartojimo žingsnį.

Laiko vertės atskiriamos tarpo simbolių. Jei laiko vertė nenurodyta, reiškia įeina visos įmanomos laiko vertės reikšmės.

`func.`**`hourly`**`()`
```
cron('0m')
```

`func.`**`daily`**`()`
```
cron('0m 0d')
```

`func.`**`weekly`**`()`
```
cron('0m 0h 0w')
```

`func.`**`monthly`**`()`
```
cron('0m 0h 1d')
```

`func.`**`yearly`**`()`
```
cron('0m 0h 1d 1M')
```

### 2.14.12 Statinės reikšmės

Statinės reikšmės arba konstantos duomenų laukams gali būti nurodomos *property.prepare* stulpelyje naudojant formulės sintaksę. Plačiau apie formules žiūrėti *Formulės* skyrelyje.

### 2.14.13 Transformavimas

*property.prepare* stulpelyje gauta šaltinio reikšmė gali būti pasiekiama per `self` kintamąjį.

*property.prepare* formulėje gali būti aprašomos kelios reikšmės atskirtos kableliu, tai naudojama ryšio laukams, kai ryšiui aprašyti reikia daugiau nei vieno duomenų lauko.

Formulėje galima naudoti kitus to pačio modelio property pavadinimus, kai aprašomo *property* reikšmės formuojamos dinamiškai naudojant vieną ar kelis jau aprašytus laukus.

*property.prepare* stulpelyje galima naudoti tokias formules:

**property.prepare**

`func.`**`null`**`()`
: Grąžina `null` reikšmę, jei toliau einančios transformacijos grąžina `null`.

`func.`**`replace`**`(`*old*, *new*`)`
: Pakeičia visus `old` į `new` simbolius eilutėje.

`func.`**`re`**`(`*pattern*`)`
: Grąžina atitinkančią reguliariosios išraiškos `pattern` reikšmę arba pirmos grupės reikšmę jei naudojama tik viena grupė arba reikšmių grupę jei `pattern` yra daugiau nei viena grupė.

`func.`**`cast`**`(`*type*`)`
: Konvertuoja šaltinio tipą į nurodytą `type` tipą. Tipų konvertavimas yra įmanomas tik tam tikrais atvejais. Jei tipų konvertuoti neįmanoma, tada metodas turėtų grąžinti klaidą.

`func.`**`split`**`()`
: Dalina simbolių eilutę naudojant `s+` *reguliariąją išraišką*. Grąžina masyvą.

`func.`**`strip`**`()`
: Pašalina tarpo simbolius iš pradžios ir pabaigos.

`func.`**`lower`**`()`
: Verčia visas raides mažosiomis.

`func.`**`upper`**`()`
: Verčia visas raides didžiosiomis.

`func.`**`len`**`()`
: Grąžina elementų skaičių sekoje.

`func.`**`choose`**`(`*default*`)`
: Jei šaltinio reikšmė nėra viena iš *enum*, tada grąžinama default reikšmė.

  Jei `default` nepateiktas, grąžina vieną iš *property.enum* reikšmių, jei duomenų šaltinio reikšmė nėra viena iš *property.enum*, tada grąžinama klaida.

`func.`**`switch`**`(*`*cases*`)`

`func.`**`case`**(*cond*, *value*)

`func.`**`case`**(*default*)

Grąžina `value`, jei tenkina `cond` arba `default`. Jei `case(default)` nepateiktas, tada grąžina pradinę reikšmę.

Jei, `cases` nepateikti, grąžina vieną iš `switch.source` reikšmių, tenkinančių switch prepare sąlygą.

`func.`**`swap`**(*old*, *new*)

Pakeičia `old` reikšmę `new`, jeigu `self` atitinka `old`.

`func.`**`return`**()

Nutraukia transformacijų grandinę ir grąžina reikšmę.

`func.`**`set`**(*name*)

Išsaugo reikšmę į kintamąjį `name`.

`func.`**`url`**()

Skaido URI į objektą turintį tokias savybes:

**scheme**
URI schema.

**netloc**
Visada URI dalis tarp scheme ir path.

**username**
Naudotojo vardas.

**password**
Slaptažodis.

**host**
Domeno vardas arba IP adresas.

**port**
Prievado numeris.

**path**
Kelias.

**query**
URL dalis einanti tarp ? ir #.

**fragment**
URL dalis einanti po #.

`func.`**`query`**()

Skaido URI query dalį į parametrus.

`func.`**`path`**()

Skaido failų sistemos arba URI kelią į tokias savybes:

**parts**
Skaido kelią į dalis (plačiau).

**drive**
Diskas (plačiau).

**root**
Šaknis (plačiau).

### 2.14.14 Kompleksinės struktūros

Daugelis duomenų šaltiniu turi galimybę saugoti kompleksines struktūras. Jei duomenys yra kompleksiniai, tada *property.source* stulpelyje galima nurodyti tik duomens pavadinimą iš pirmojo lygmens, gilesniuose lygmenyse esančius duomenis galima aprašyti naudojant formules *property.prepare* stulpelyje.

Analogiškai duomenų atranką galima daryti ir `model` eilutėse, jei tai leidžia duomenų šaltinis.

Kaip pavyzdį naudosime tokią *JSON* duomenų struktūrą:

```
{
    "result": {
        "count": 1,
        "results": [
            {
                "type": "dataset",
                "tags": ["CSV"]
            }
        ]
    }
}
```

**property.prepare**

func.**getattr**(*object*, *name*)

Grąžina `object` savybę `name`.

```
>>> self.result.count
1
```

func.**getitem**(*object*, *item*)

Grąžina `object` objekto `item` savybę arba `object` masyvo `item` elementą.

```
>>> self["result"]["count"]
1
```

*getitem()* ir *getattr()* gali būti naudojami kartu.

```
>>> self.result.results[0].type
"dataset"
```

*getitem()* gali būti naudojamas, kaip masyvo elementų filtras pateikiant filtro sąlygą.

```
>>> self.result.results[tags = "CSV"].type
["dataset"]

>>> self.result.results[item(tags) = "CSV"].type
["dataset"]
```

Norint gauti visus masyvo elementus, galima naudoti tokią išraišką:

```
>>> self.result.results[].tags[]
["CSV"]
```

`func.`**`first`**`(`*`object`*`, `*`default`*`)`

Grąžina pirmą `object` sąrašo reikšmę, jei sąrašas tuščias, tada grąžina `default` reikšmę, jei `default` nenurodytas, tada nutraukia vykdymą su klaidą.

```
>>> self.result.results[].tags.first()
"CSV"
```

Jei `self.result.results[].tags būtų tuščias, tada:

```
>>> self.result.results[].tags.first(null)
null
```

Analogiška struktūra gali būti gaunama ir lentelėse, kai stulpelių pavadinimai nurodyti keliose eilutėse, pavyzdyje pateiktą struktūrą atitiktų tokia lentelė:

| result count | results type | tags |
|---|---|---|
| 1 | dataset | CSV |

Šioje lentelėje stulpelių pavadinimai pateikti trijose eilutėse, todėl `model.prepare` reikėtų naudoti *`header(0, 1, 2)`*.

## 2.15 Sąvokos

**TGIĮ**

Teisės gauti informaciją ir duomenų pakartotinio naudojimo įstatymas.

Šis įstatymas įpareigoja valstybės ir savivaldybių institucijas ir joms pavaldžius subjektus atverti duomenis.

Kelios citatos iš įstatymo:

**4 straipsnis**

1. Institucijos ir valstybės valdomi subjektai privalo teikti pareiškėjams ar jų atstovams duomenis, įskaitant pakartotiniam naudojimui skirtus duomenis, išskyrus šio įstatymo ir kitų įstatymų nustatytus atvejus.

**15 straipsnis**

1. Visi institucijos ar valstybės valdomo subjekto duomenys turi būti inventorizuoti laikantis principo, kad duomenys gali būti skelbiami pakartotinai naudoti, jeigu tai neprieštarauja šiam ir kitiems įstatymams. Inventorizuotų duomenų sąrašas turi būti skelbiamas Lietuvos atvirų duomenų portale.

2. Institucijos ir valstybės valdomi subjektai turi sudaryti duomenų, dėl kurių yra pateiktos užklausos Lietuvos atvirų duomenų portale arba kurių pakartotinis naudojimas, institucijos ir valstybės valdomo subjekto vertinimu, gali kurti pridėtinę vertę, rinkinius ir juos skelbti šiame portale, jeigu tai neprieštarauja šiam ir kitiems įstatymams.

**17 straipsnis**

1. Lietuvos atvirų duomenų portalas yra valstybės informacinė sistema, skirta duomenų rinkiniams ir jų metaduomenims sisteminti ir skelbti naudojant vienodą metaduomenų aprašymo formatą, taip pat vieno langelio principu institucijų ir valstybės valdomų subjektų sudarytiems duomenų rinkiniams ir jų metaduomenims ieškoti, peržiūrėti, parsisiųsti, pareiškėjų užklausoms registruoti ir kitoms paslaugoms, susijusioms su šios informacinės sistemos paskirtimi, teikti.

5. Institucijos ir valstybės valdomi subjektai privalo užtikrinti, kad inventorizuotų duomenų sąrašai ir sudaryti duomenų rinkiniai Lietuvos atvirų duomenų portale bus surasti ir pasiekiami šio portalo tvarkytojo nustatyta tvarka ir priemonėmis.

**18 straipsnis.**

Pareiškėjo teisės gali būti ginamos šiais būdais:

1) pareiškėjas turi teisę apskųsti institucijos veiksmus, neveikimą ar administracinį sprendimą, taip pat institucijos vilkinimą atlikti jos kompetencijai šiuo įstatymu priskirtus veiksmus Viešojo administravimo įstatymo nustatyta tvarka;

2) pareiškėjas turi teisę apskųsti valstybės valdomo subjekto veiksmus ar neveikimą, taip pat valstybės valdomo subjekto vilkinimą atlikti jo kompetencijai šiuo įstatymu priskirtus veiksmus tam pačiam valstybės valdomam subjektui arba bendrosios kompetencijos teismui.

**Europos sąveikumo karkasas**

Rekomendacijų rinkinys apie tai, kaip užtikrinti didesnį skaitmeninį sąveikumą tarp Europos šalių.

Rekomendacijų sąrašas:

2. Publish the data you own as open data unless certain restrictions apply.

3. Ensure a level playing field for open source software and demonstrate active and fair consideration of using open source software, taking into account the total cost of ownership of the solution.

41. Establish procedures and processes to integrate the opening of data in your common business processes, working routines, and in the development of new information systems.

42. Publish open data in machine-readable, non-proprietary formats. Ensure that open data is accompanied by high quality, machine-readable metadata in non-proprietary formats, including a description of their content, the way data is collected and its level of quality and the licence terms under which it is made available. The use of common vocabularies for expressing metadata is recommended.

43. Communicate clearly the right to access and reuse open data. The legal regimes for facilitating access and reuse, such as licences, should be standardised as much as possible.

44. Put in place catalogues of public services, public data, and interoperability solutions and use common models for describing them.

45. Where useful and feasible to do so, use external information sources and services while developing European public services.

**atvirų duomenų direktyva**

2019 m. birželio 20 d. Europos Parlamento ir Tarybos direktyva (ES) 2019/1024 dėl

atvirųjų duomenų ir viešojo sektoriaus informacijos pakartotinio naudojimo.

**duomenų valdymo aktas**
: 2020 m. lapkričio 25 d. Europos Parlamento ir Tarybos reglamento (ES) pasiūlymas 2020/0340 dėl Europos duomenų valdymo (Duomenų valdymo aktas).

**aplinkos kintamasis**
: Angliškai tai vadinama *environment variables*, tai yra operacinės sistemos aplinkos kintamieji.

  Plačiau apie tai skaitykite Vikipedijoje.

**ADP**
: Atvirų duomenų portalas, sudarytas iš *atvirų duomenų katalogo* ir *duomenų saugyklos*.

**ADK**
: Lietuvos atvirų duomenų katalogas, prieinamas adresu data.gov.lt.

**duomenų katalogas**
: Lietuvos duomenų portalo sudedamoji dalis, skirta metaduomenims apie duomenų šaltinius registruoti.

  Duomenų katalogas prieinamas adresu data.gov.lt.

**ADS**
: Atvirų duomenų saugykla, skirta pakartotinio panaudojimo duomenų publikavimui, valstybinė atvirų duomenų saugykla pasiekiama get.data.gov.lt adresu.

**DSA**
: Duomenų struktūros aprašas yra lentelė, kurioje išsamiai aprašyta tam tikro duomenų šaltinio duomenų struktūra. DSA lentelę sudaro penkios dimensijos (duomenų rinkinys, resursas, bazė, modelis, savybė) ir dešimt metaduomenų stulpelių.

**ADSA**
: *DSA* lentelė, kurioje aprašomi jau atverti ir viešai prieinami duomenys.

**ŠDSA**
: *DSA* lentelė, kurioje aprašoma neatvertų, *pirminio duomenų šaltinio* duomenų struktūra.

**didelės vertės duomenys**
**aukštos vertės duomenys**
: Duomenys apibrėžti *atvirų duomenų direktyvos* 5 skyriuje.

  Aukštos vertės duomenų sritys yra šios:

  - Geoerdviniai duomenys
  - Aplinka ir žemės stebėjimai
  - Meteorologiniai duomenys
  - Statistika (demografiniai ir ekonominiai rodikliai)
  - Įmonės ir įmonių savininkai
  - Judumas

**BDAR**
: 2016 m. balandžio 27 d. Europos Parlamento ir Tarybos reglamentas (ES) 2016/679 dėl fizinių asmenų apsaugos tvarkant asmens duomenis ir dėl laisvo tokių duomenų judėjimo ir kuriuo panaikinama Direktyva 95/46/EB (Bendrasis duomenų apsaugos reglamentas).

**duomenų serializavimo formatas**
: Duomenys gali būti serializuojami įvairiais formatais, pavyzdžiui YAML formatu:

```
type: project
title: Manifestas
```

JSON formatu:

```
{"type": "project", "title": "Manifestas"}
```

Turtle formatu:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
<http://atviriduomenys.lt> a foaf:Project;
    rdfs:label "Manifestas" .
```

MessagePack dvejetainiu formatu, kurio turinys pateiktas naudojant BASE64 koduotę:

```
gqR0eXBlp3Byb2plY3SkbmFtZapNYW5pZmVzdGFz
```

Visuose šiuose pavyzdžiuose yra pateikti tie patys duomenys, tačiau naudojami skirtingi duomenų serializavimo formatai, koduotės ir skirtingi žodynai.

**brandos lygis**

Duomenų brandos lygiai yra apibrėžti 5 ★ Open Data svetainėje. Viso yra penki brandos lygiai, tačiau papildomai verta įtraukti ir nulinį brandos lygį, kai duomenų poreikis yra, tačiau duomenys nekaupiami arba negali būti publikuojami dėl teisinių ar kitų apribojimų.

5 ★ Open Data svetainėje brandos lygia apibrėžti, kaip pavyzdį nurodant formatus. Nors formatus galima naudoti kaip pavyzdį labai abstrakčiai apibūdinant ką reiškia brandos lygiai, tačiau tikslus brandos lygis gali būti suteiktas tik atskiriems duomenų laukams, o ne formatui.

Duomenų brandos lygiai yra tokie:

**0**

Duomenys nekaupiami, tačiau poreikis tokiems duomenims yra. Gali būti ir tokių atvejų, kai duomenys yra kaupiami, tačiau dėl teisinių ar kitų priežasčių negali būti publikuojami.

**1**

Duomenys kaupiami ir publikuojami viešai, bet kokia forma ir bet kokiu formatu. Pavyzdžiui datos tipo laukas gali būti pateikiamas įvairiais formatais „Pirmadienis“, „2021 gegužės 10 d.“, „5/10/21“ ir pan. Kadangi šiuo atveju data gali būti užrašyta bet kokia forma ir bet kokiu tikslumu, nėra galimybės automatinėmis priemonėmis patikimai nuskaityti tokių duomenų.

**2**

Publikuojami duomenys turi aiškią, mašininiu būdu nuskaitomą struktūrą, tačiau pateikiami nestandartiniu arba nuosavybiniu formatu. Pavyzdžiui datos tipo lauko duomenys pateikiami nestandartiniu formatu, tačiau visos reikšmės pateiktos naudojant tą patį formatą, „5/10/21“, „6/10/21“ ir pan. Šiuo atveju, automatiškai nuskaityti tokius duomenis įmanoma tik papildomai įgyvendinant duomenų nuskaitymo priemones, kuriose yra įgyvendintas būtent tokio nestandartinio formato duomenų skaitymas.

**3**

Duomenys pateikiami naudojant standartinį formatą. Lietuvos atvirų duomenų kontekste, *standartiniai formatai yra apibrėžti duomenų struktūros aprašo specifikacijo-*

*je*. Pavyzdžiui datos tipo lauko duomenys pateikiami standartiniu ISO 8601 formatu. Kadangi duomenys yra pateikti standartiniu formatu, pačio formato specifikacija yra atvira ir viešai publikuojama, o duomenų nuskaitymo priemonės tokį atvirą formatą palaiko, todėl tokių duomenų nuskaitymui nereikia įdėti jokio papildomo darbo.

**4**

Kiekvienas publikuojamų duomenų *objektas* turi unikalų identifikatorių ir naudojant tokius unikalius objektų identifikatorius, skirtingų tipų objektai siejami tarpusavyje. Kartu su duomenimis pateikiami ir metaduomenys apie tai, kaip skirtingų tipų objektai siejasi tarpusavyje.

Pavyzdžiui miesto tipo objektui „Vilnius“ yra suteiktas unikalus identifikatorius *6868eca7-0ae1-4390-83d0-7af642a62863*, o šalies tipo objekto „Lietuva“ duomenų lauko „sostinė“ reikšmė yra objekto „Vilnius“ unikalus identifikatorius *6868eca7-0ae1-4390-83d0-7af642a62863*.

Turint tokį brandos lygį, duomenis galima ne tik nuskaityti, bet ir jungti tarpusavyje, o jungiant skirtingus duomenis tarpusavyje atsiveria daugiau galimybių juos naudoti įvairiuose taikymuose.

**5**

Kartu su publikuojamais duomenimis, pateikiami ir metaduomenys apie tai, kaip publikuojami duomenys siejasi su kitais viešaisiais duomenų žodynais (ontologijomis). Pavyzdžiui datos duomenų laukas yra susiejamas su „Dublin Core Metadata Initiative“ publikuojama ontologija, nurodant, kad datos lauko semantinė prasmė yra tokia pati, kaip apibrėžta dcterms:created ontologijoje. Šiuo atveju, nurodoma, kad datos laukas būtent yra tam tikro resurso sukūrimo data.

Kai duomenys yra susieti su išoriniais žodynais, atsiranda galimybė įgyvendinti tokias priemones, kurios veiktų universaliai, nepriklausomai nuo duomenų šaltinio ar duomenų kilmės.

**kanoniniai duomenys**

Kanoniniai duomenys yra tarsi duomenų etalonas, kuris nusako kokios duomenų reikšmės yra teisingos. Pavyzdžiui įmonės pavadinimas gali būti užrašomas įvairiausiomis formomis, pavyzdžiui:

| Įmonės kodas | Įmonės pavadinimas |
|---|---|
| - | UAB "Duomesta" |
| - | UAB „Duomesta“ |
| - | Duomesta |
| - | DUOMESTA |
| - | Uždaroji akcinė bendrovė Duomesta |
| - | Duomesta, UAB |
| - | DSTA UAB |

Jei duomenų rinkinyje nėra pateiktas įmonės registracijos kodas, tada unikaliai identifikuoti įmonę yra gan sudėtinga.

Tačiau turint autoritetingus kanoninius duomenis:

| Įmonės kodas | Įmones pavadinimas |
|---|---|
| 111111111 | UAB "Duomesta" |

Užduotis unikaliai identifikuoti įmonę pasidaro paprastesnė. Todėl kanoniniai duomenys yra labai svarbūs.

**kodinis pavadinimas**
Pavadinimas, kuriam keliami tam tikri apribojimai.

**manifestas**
Atvirų duomenų manifestas yra *DSA* lentelių rinkinys, kuriuose aprašyti duomenų šaltiniai ir juose esančių duomenų struktūra.

Žodis manifestas yra kilęs iš programavimo srityje naudojamo termino Manifesto failas, kuriame pateikiami metaduomenys apie programinio paketo sandarą.

Duomenų kontekste, žodis manifestas turėtu būti suprantamas, kaip metaduomenų lentelė apie įvairiuose duomenų šaltiniuose publikuojamus duomenis.

**metaduomenys**
Duomenys apie duomenis yra vadinami metaduomenimis. Pavyzdžiui duomenų struktūros aprašas konkrečiam CSV duomenų failui gali būti vadinamas CSV failo metaduomenimis.

**normalizavimas**
Duomenų normalizavimas yra duomenų struktūros transformavimo procesas taikant taip vadinamas normalines formas, tam kad sumažinti duomenų pasikartojimą.

Plačiau apie tai skaitykite Vikipedijoje.

**prieigos taškas**
Prieigos taškas yra *REST API* terminas, nurodantis URL kelio dalį iki tam tikro resurso.

Plačiau skaitykite Vikipedijoje.

**REST API**
Representational State Transfer (REST) yra taisyklių ir rekomendacijų rinkinys sirtas *web servisams* kurti.

Plačiau skaitykite Vikipedijoje.

**web servisas**
Web servisas yra interneto paslauga skirta automatizuotiems robotams. Interneto svetainės dažniausiai yra skirtos žmonėms, tačiau web servisai yra skirti mašinoms, kurios gali komunikuoti viena su kita.

Plačiau skaitykite Vikipedijoje.

**YAML**
YAML yra *duomenų serializavimo formatas*, kuris skirtas ne tik mašininiam skaitymui, bet su šio formato turiniu tiesiogiai gali dirbti ir žmogus. YAML formato pavyzdys:

```yaml
container:
  name: value
```

YAML yra sukurtas JSON formato pagrindu, siekant palengvinti darbą su JSON serializuotais duomenimis žmonėms. Analogiškas pavyzdys JSON formatu atrodo taip:

```json
{"container": {"name": "value"}}
```

**viešasis žodynas**
Viešieji žodynai, dar vadinami ontologijomis, šie žodynai dažnai yra gerai dokumentuoti ir skelbiami viešai, jie yra skirti globaliam susietųjų duomenų tinkui kurti (angl. *linked data*).

**sisteminis pavadinimas**
: Sisteminis pavadinimas yra naudojamas objektų identifikavimui ir yra naudojamas URL nuorodose ir visur kitur, kur reikia nurodyti ryšį su objektu, naudojamas to objekto sisteminis pavadinimas.

  Sisteminis pavadinimas sudaromas tik iš lotyniškų raidžių ir -_/ simbolių.

**pirminis duomenų šaltinis**
: Įstaigos ar kitos organizacijos pagrindinis duomenų šaltinis.

**DCAT**
: Duomenų katalogo žodynas (angl. Data Catalog Vocabulary) yra standartas skirtas duomenų rinkiniams aprašyti. Aprašant duomenis DCAT standartu reikėtų vadovautis *DCAT-AP* specifikacijomis.

**DCAT-AP**
: DCAT-AP (DCAT Application Profile) yra specifikacija, detalizuojanti DCAT naudojima, nurodant kurios DCAT klasės ir savybės yra privalomos, kurios rekomenduojamos ir kaip jas naudoti.

**dimensija**
: Dimensija yra metaduomenų, aprašomų DSA lentelėje, grupė. DSA lentelėje metaduomenys skirstomi į tokias dimensijas:

  - duomenų rinkinys
  - resursas
  - bazė
  - modelis
  - savybė

  Kiekviena dimensija turi skirtingą metaduomenų detalumo lygį.

  Plačiau apie dimensijas: *Dimensijos*.

**duomenų rinkinys**
: Duomenų rinkinys apibrėžia turimus arba pageidaujamus duomenis, reikalingus konkrečios organizacijos, konkrečiai veiklai vykdyti.

  Duomenų rinkinys gali būti registras, informacinės sistemos duomenų bazė, interneto svetainės duomenų bazė, skaičiuoklės lentelė, dokumentų katalogas arba duomenys, kurie dar nėra kaupiami, tačiau yra reikalingi tam tikrai veiklai vykdyti.

  Duomenų rinkinio fizinė reprezentacija, tai yra patys duomenys yra vadinami *distribucija*. Duomenų rinkinyje gali būti daugiau nei viena distribucija, jei fiziškai duomenys yra suskaidyti pagal vietos, laiko, detalumo, struktūros elementus, natūralios kalbos ar kitus kriterijus.

  Dažnai duomenų rinkinys painiojamas su distribucija. Duomenų rinkinys apibrėžia tam tikrą grupę duomenų, kurie nebūtinai fiziškai egzistuoja, tuo tarpu distribucija yra fiziniai duomenys įeinantys į duomenų rinkinio sudėtį.

  Duomenų rinkiniai neskaidomi pagal vietos, laiko, detalumo, struktūros ar kitus kriterijus.

  Plačiau apie tai, kaip duomenų rinkiniai aprašomi duomenų struktūros apraše skaitykite skyriuje *dataset*.

  Duomenų rinkinys atitinka dcat:Dataset apibrėžimą.

**distribucija**
: Distribucija yra duomenų rinkinio fizinė reprezentacija. Vienas duomenų rinkinys gali būti sudarytas iš kelių distribucijų, tuos pačius duomenis pateikiant skirtingais formatais, suskaidant duomenis pagal laiko, vietos ar kitus kriterijus, tuos pačius duomenis pateikiant skirtingu detalumu arba pateikiant agreguotus duomenis įvairiais pjūviais.

  Duomenų struktūros aprašo kontekste, distribucija yra tas pats, kas resource.

  Distribucija atitinka dcat:Distribution apibrėžimą.

**bazė**
: Bazė arba loginė klasė yra modelių grupė turinčių bendras savybes ir vienodą semantinę prasmę.

  Dažnai skirtingų organizacijų veikloje naudojami duomenų rinkiniai turi vienodą semantinę prasmę. Pavyzdžiui, daugelis organizacijų turi naujienų duomenis. Norint visų organizacijų naujienų duomenis aprašyti vieningai, galima pasitelkti vieną bazę, arba vieną duomenų rinkinį, kurio struktūrą naudoja visi kiti rinkiniai. Tai bazė būtent ir būtų struktūros šablonas pagal kurį būtų sudaromos visų kitų analogiškų rinkinių struktūros.

  Bazė yra tas pats, kas *modelis* arba tiksliau modelio šablonas.

  Duomenų struktūros aprašo kontekste apie bazę plačiau skaitykite skyriuje *base*.

**modelis**
: Modelis yra gan plati sąvoka turinti daug prasmių, priklausomai nuo konteksto. Šioje dokumentacijoje, modelis yra duomenų struktūros aprašo dalis leidžianti aprašyti duomenis pateiktus įvairiais formatais.

  Tiksli modelio prasmė priklauso nuo duomenų šaltinio, kurio duomenys yra aprašomi:

  - CSV failo atveju, modelis yra CSV faile esanti lentelė,
  - Excel failo atveju, modelis yra kiekviena lentelė (arba lapas) esanti Excel faile,
  - SQL duomenų bazių atveju, modelis yra viena duomenų bazės lentelė,
  - JSON dokumento atveju, modelis yra kiekvienas masyvas esantis JSON dokumente,
  - XML atveju, modelis yra kiekvienas elementų masyvas esantis XML faile.

  Duomenų rinkiniai aprašo konkretaus autoriaus duomenis, skirtingi autoriai gali naudoti tuos pačius duomenis, todėl duomenys skirtinguose rinkiniuose gali dubliuotis. Tuo tarpu modeliai aprašo duomenis pagal jų semantinę prasmę, nepriklausomai nuo autoriaus, tai leidžia apjungti skirtingų autorių naudojamus duomenis, pagal jų semantinę prasmę, modelių pagalba.

  *DSA* lentelėje atitinka *`model`*. Duomenų modelį atitinkanti fizinė reprezentacija nurodoma *`source`* stulpelyje. *`source`* gali būti duomenų bazės lentelė, CSV failas ar kita, priklauso nuo duomenų šaltinio tipo. Sąsaja su išoriniais žodynais pateikiama *`uri`* stulpelyje. Siejant su išoriniais žodynais, pateikiama nuoroda į rdfs:Class.

**savybė**
: *Duomenų modeliui* priklausančių informacinių *objektų* savybė, pavyzdžiui miesto pavadinimas, šalis kuriai priklauso miestas. *DSA* lentelėje atitinka *`property`*. Atitinka rdfs:Property arba lentelės stulpelį.

**subjektas**
: Subjektas lietuvių kalboje vadinamas veiksniu, duomenų kontekste įvardija objektą apie kurį eina kalba.

  Tarkime saknyje „Namas turi stogą“ subjektas yra Namas, todėl, kad kalba eina apie namą.

**objektas**
: Vienas duomenų įrašas sudarytas iš savybių ir savybėms priskirtų reikšmių. Informacinis objektas turi turėti unikalų identifikatorių. Atitinka rdfs:Resource arba lentelės vieną eilutę.

  Plačiau apie objektą: *Objektas*.

**žodynas**
: Duomenų kontekste, žodynas yra susitarimas, kokiais pavadinimais vadinami objektai ir jų savybės. Dažniausiai kiekvienas duomenų rinkinys turi savo vidinį naudojamą žodyną, visas Lietuvos atvirų duomenų modelis turi savo vidinį žodyną, kuris suvienodina skirtingus duomenų rinkinių naudojamus žodynus. Yra *viešieji žodynai*, dar vadinami ontologijomis, kurie yra skelbiami viešai ir skirti globaliam susietųjų duomenų tinklui kurti.

  Duomenų kontekste, žodynas yra tiesiog *modelių* ir *savybių* pavadinimų rinkinys. Skirtingi duomenų šaltiniai dažniausiai naudoja skirtingus žodynus, t.y. naudoja skirtingus *modelių* ir *savybių* pavadinimus.

  *Duomenų struktūros aprašas* leidžia skirtinguose duomenų šaltiniuose naudojamus pavadinimus suvienodinti, taip, kad visi šaltiniai naudotų vieningą žodyną.

  Vieningo žodyno sudarymas yra gan sudėtinga užduotis, todėl, *DSA* leidžia prie vieningo žodyno pereiti palaipsniui:

  - pirmiausia sudaromas vieno duomenų rinkinio žodynas,
  - kuris palaipsniui transformuojamas į Lietuvos vieningą žodyną,
  - o Lietuvos vieningas žodynas palaipsniui transformuojamas į globalų žodyną, nurodant sąsajas su išoriniais žodynais ir standartais.

  Žodynai sudaromi pasitelkiant vardų erdves.

**API**
: Programavimo sąsaja (*angl. Application Programming Interface*).

**duomenų šaltinis**
: Resursas, kuriame saugomi duomenys. Toks resursas tampa duomenų šaltiniu, kai tokius duomenis norima pakartotinai panaudoti, tokiu atveju, iš pakartotinio panaudojimo perspektyvos toks resursas tampa duomenų šaltiniu.

**ETL**
: Duomenų ištraukimas, transformavimas ir užkrovimas (*angl. Extract Transform Load*).

**iteratorius**
: Tam tikra funkcija, kuri grąžina keletą elementų, tačiau ne visus iš karto, o po vieną.

**URI**
: Universalus resurso identifikatorius (*angl. Universal Resource Identifier*).

**POSIX**
: Universali operacinių sistemų sąsaja (*angl. Portable Operating System Interface*) - standartas apibrėžiantis operacinių sistemų sąsają, kad skirtingos operacinės sistemos būtų suderinamos tarpusavyje.

  https://en.wikipedia.org/wiki/POSIX

**DOS**
: MS-DOS.

**reguliarioji išraiška**
: Simbolių seka apibrėžianti tam tikrą šabloną tekste (angl. Regular Expression).

**JSON**
: Atviras duomenų formatas (angl. JavaScript Object Notation).

**RDF**
: Duomenų modelis sudarytas iš subjekto, predikato ir objekto tripletų (angl. Resource Description Framework).

**IVPK**
: Informacinės visuomenės plėtros komitetas.

## 2.16 Keitimų istorija

### 2.16.1 0.2.0 (neišleista)

- Panaikinta *base* dimensija, baziniai modeliai perkeliami į *model.type*.
- Panaikinta *param* dimensija, parametrizavimas perkeltas prie *property*, *property.type* stulpelyje nurodant `param`.

### 2.16.2 0.1.0 (2022-03-03)

Pirmoji duomenų struktūros aprašo versija.

# Python Module Index

# Indeksas

## E



## F



## G



## H



## I



## J



## K



## L



## M

## T



## U



## V



## W



## X



## Y



##