

---

# DSA Specifikacija

**VSSA**

**2024-12-17**



<b>1</b>	<b>Specifikacijos</b>	<b>3</b>
<b>2</b>	<b>Turinys</b>	<b>5</b>
2.1	Koncepcinis modelis . . . . .	5
2.2	Lentelės formatas . . . . .	12
2.3	Dimensijos . . . . .	18
2.4	Papildomos dimensijos . . . . .	30
2.5	Duomenų tipai . . . . .	44
2.6	Kodiniai pavadinimai . . . . .	56
2.7	Matavimo vienetai . . . . .	59
2.8	Apibendrinimas . . . . .	65
2.9	Asociacija . . . . .	65
2.10	Funkciniai modeliai . . . . .	79
2.11	Brandos lygiai . . . . .	86
2.12	Prieigos lygiai . . . . .	100
2.13	Duomenų schemas . . . . .	100
2.14	Duomenų šaltiniai . . . . .	115
2.15	Vardų erdvės . . . . .	122
2.16	Išoriniai žodynai . . . . .	123
2.17	Formulės . . . . .	127
2.18	Sąvokos . . . . .	144
2.19	Keitimų istorija . . . . .	154
	<b>Python Module Index</b>	<b>155</b>
	<b>Indeksas</b>	<b>157</b>



**Ši versija:**

Naujausias redaktoriaus juodraštis

**Klaidų sekimas:**

Github

**Redaktorius:**

VSSA

**Vertimas (nenorminis):**

Nėra

**Naujausia paskelbta versija:**

Naujausia paskelbta versija

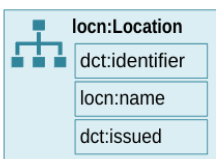
**Naujausias redaktoriaus juodraštis:**

Naujausias redaktoriaus juodraštis

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.

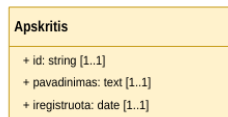
Semantinis modelis (OWL)



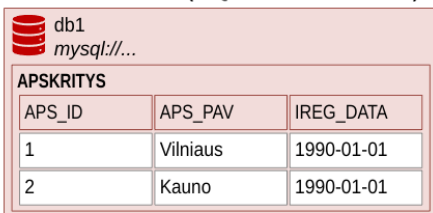
Veiklos modelis



Koncepcinis modelis (UML)

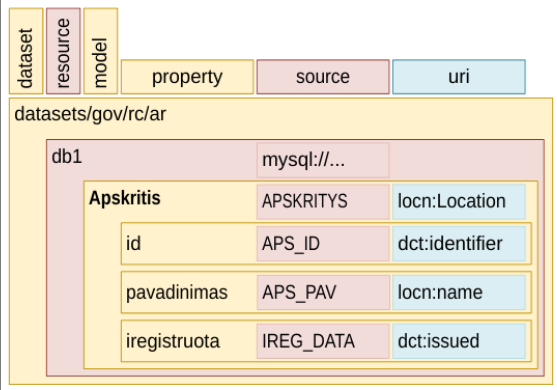


Fizinis modelis (SQL, XML, CSV, ...)

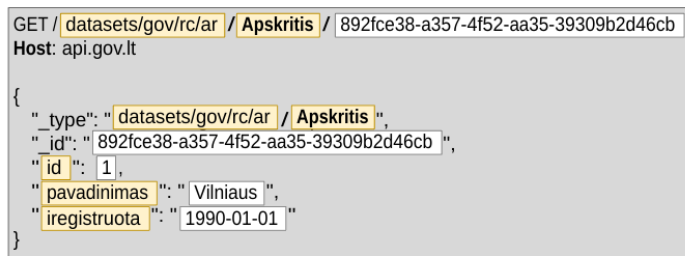


**Duomenų struktūros aprašas (DSA)**

Loginis modelis (UDTS)



API (UDTS)



*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 žodymais 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 žodymais.

# SKYRIUS 1

---

## Specifikacijos

---

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





## 2.1 Konceptcinis modelis

Prieš pradedant darbą su struktūros aprašais, būtina pasirengti konceptcinio modelio UML diagramą, vadovaujantis [Conceptual model conventions \(UML\)](#) reikalavimais.

Konceptcinis 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 pasitvirtinti duomenų modelį, kuris vėliau bus taikomas rengiant duomenų struktūros aprašus.

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

Kaip pavyzdį, naudosime žemiau pateiktą konceptcinį duomenų modelį:

Pagal šį konceptcinį modelį, DSA lentelė atrodytu taip:

d	r	m	property	type	ref	prepare
datasets/gov/example						
			<b>Administracija</b>		kodas	
			kodas	string		
			pavadinimas	string		
			tipas	string		
				enum		"APSKRITIS"
						"SAVIVALDYBE"
			<b>Gyvenvietė</b>		id	
			id	integer		
			pavadinimas	string		
			savivaldybė	ref	<b>Savivaldybė</b>	
			<b>Apskritis</b>	<b>Administracija</b>	kodas, tipas	
			kodas	string		
			pavadinimas	string		
			tipas	string		"APSKRITIS"
			<b>Savivaldybė</b>	<b>Administracija</b>	kodas, tipas	
			kodas	string		
			pavadinimas	string		
			tipas	string		"SAVIVALDYBE"
			apskritis	ref	<b>Apskritis</b>	

Pavadinimai nurodyti koncepciniame modelyje, turi būti identiški sutapti su pavadinimai nurodytais DSA lentelės loginio modelio model, property, type, ref ir prepare stulpeliuose.

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 Gyvenvieta objektai gali būti:

- Vilnius
- Kaunas
- Klaipėda

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

Imant duomenų lentelę iš Gyvenvieta modelio, gausime tokius duomenis.

id	pavadinimas	savivaldybe
1	Vilnius	10
2	Kaunas	11
3	Klaipėda	12

Šioje lentelėje yra trys objektai.

Objekto pavyzdys UML diagramoje:

UML diagramoje turime tris objektus Vilnius, Kaunas ir Klaipėda, priskirti klasei Gyvenvieta.

Skirtingi objektai gali būti klasifikuojami į klases arba esybes.

### 2.1.2 Klasė

Klasė arba Esysbė yra vienodas savybės 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 Gyvenvieta.

Klasės pavyzdys UML diagramoje:

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

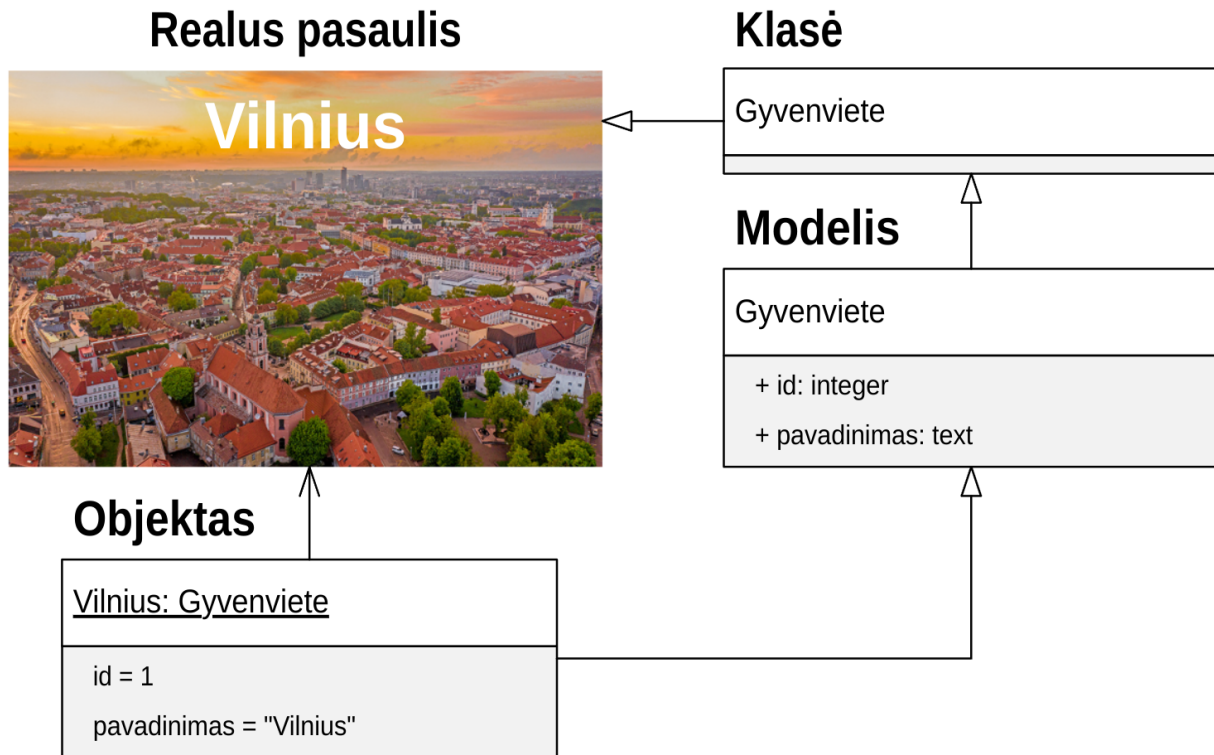
Tarkime modelis Gyvenvietė turi savybę pavadinimas, tačiau tai nėra klasės savybė, todėl, kad pavadinimas yra duomenų atributas, kuris nėra klasę apibūdinanti savybė.

Nurodžius savybes prie klasės, iškeliamas griežtas reikalavimas, visiems modeliams ir sub-klasėms, 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 apibrezia pačią klasę ir tampa klasės dalimi, tačiau modelio savybės neturi įtakos klasės semantiniam apibrėžimui, tai yra tiesiog duomenų laukų sąrašas pateikiamas 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 Apibendrinimas

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 kryptis iš labiau specializuotos siauresnės prasmės turinčios klasės, į labiau apibendrintą, platesnę prasmę turinčią klasę, pavyzdžiui:

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

### 2.1.5 Identifikatoriai

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 lokaliūs 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 Gyvenvieta, lokaliūs 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, žiūrėti *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.
- \* - lauko reikšmė gali būti pateikta daugiau nei vieną kartą.

Pavyzdys:

UML diagramoje matote Gyvenvieta 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 sąrašas, savybės, kuri yra pateikta prie rodyklės neberodoma.

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 viešajame 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/Gyvenvieta
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:Gyvenvieta
ar:id
ar:pavadinimas
```

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

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 priemonės. 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 sąjaja.

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

Dimensijos				Metaduomenys	
dataset	resource	model	property	type	source
datasets/gov/rc/jar/ws					
db				sql	sql://host/db
JuridinisAsmuo					JA
kodas				integer	ja_kodas
pavadinimas				string	ja_pavadinimas
adresas				string	adresas
forma				integer	forma_kodas
				enum	1
					2



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 lotyniškėmis raidėmis, žodžiams atskirti gali būti naudojamas \_ simbolis.

Pagal semantinę prasmę atitinka *dc: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 lotyniškėmis raidėmis, žodžiai skiriami \_ simboliu.

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

Pagal semantinę prasmę atitinka [dcat:Distribution](#) arba [rml:logicalSource](#).

### Pavyzdys

resource1  
db1

### Taip pat žiūrėkite

*resource*

*Duomenų šaltiniai*

## base

### Modelio bazė

Kodinis bazinio modelio pavadinimas. Atitinka [rdfs:subClassOf](#) prasmę (*model* [rdfs:subClassOf](#) *base*).

Šiame stulpelyje įrašomas kito *model* stulpelyje įrašyto modelio kodinis pavadinimas.

Galima nurodyti absoliutų modelio pavadinimą, kuris prasideda / simboliu, taikoma, kai nurodomas bazinis modelis iš kito duomenų rinkinio, pavyzdžiui:

### Pavyzdys

/datasets/gov/example/Country

Arba galima nurodyti reliatyvų pavadinimą, kuris neprasideda / simboliu, taikoma, kai bazinis modelis yra tame pačiame duomenų rinkinyje, pavyzdžiui:

### Pavyzdys

Country

Jei base stulpelis neužpildytas, tada visi modeliai, neturintys base yra laikomi baziniais modeliais.

**Taip pat žiūrėkite**

*Baziniai modeliai*  
*Apibendrinimas*

**model**

**Modelis (lentelė)**

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

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

**Pavyzdys**

Gyvenvieta  
AdministracijosTipas

**Taip pat žiūrėkite**

*model*  
*Modelis*

**property**

**Savybė (stulpelis)**

Kodinis savybės pavadinimas, užrašomas mažosiomis lotyniškais 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 pridėdamas 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 turėtų 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
<i>Asociacija</i> <i>Matavimo vienetai</i> <i>enum</i>

source

**Šaltinis**

Duomenų šaltinio struktūros elementai.

Taip pat žiūrėkite
<i>Duomenų šaltiniai</i>

prepare

**Formulė**

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

Taip pat žiūrėkite
<i>Formulės</i>

level

**Brandos lygis**

Duomenų brandos lygis, atitinka 5 Star Data.

Taip pat žiūrėkite
<i>Brandos lygiai</i>

access

**Prieiga**

Duomenų prieigos lygis.

Taip pat žiūrėkite
<i>Prieigos lygiai</i>

### 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 lentelės būtų lengviau skaityti.

## 2.3 Dimensijos

Dimensijos leidžia vienoje lentelėje sutalpinti kelias skirtingas lenteles turinčias bendrą 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
<b>Gyvenvietė</b>				Esybė
pavadinimas				Savybė

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

Daugiamatė lentelė pateikta viršuje, atitiktą tokią vienamąją lentelę:

type	name	title
dataset	datasets/gov/rc/ar/ws	Duomenų rinkinys
resource	db	Duomenų teikimo paslauga
model	<b>Gyvenvietė</b>	Esybė
property	pavadinimas	Savybė

Kadangi DSA lentelė yra daugiamačė, nurodant stulpelį, jei kalbama apie konkrečios dimensijos stulpelį, reikia nurodyti ir dimensiją, pavyzdžiui `dataset.title` nurodo būtent apie `dataset` dimensijos `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 ke-  
liamų 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	Registru centras
	datasets/gov/rc/ar			ns	Adresu 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 konfigūracijoje, 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

Jeigu duomenų šaltinis reikalauja naudotojo vardo ir slaptažodžio, rekomenduojama nurodyti 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.http(*method='GET', body=form*)

Papildomi parametrai, reikalingi 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 sąrašė.

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 rak-tus, 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 sąrašė, 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 modelį.

## Pavyzdžiai

### Pavyzdys

dataset	base	model	property	type	ref	source
example						
		Location			id	
			id	integer		
			name@lt	text		
			population	integer		
	Location				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**

Gyvenvieta  
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 turėtų 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				
	<b>Location</b>			code
		code	integer	
		name@en	string	

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

### `model.ref`

Kableliu atskirtas sąrašas *`model.property`* duomenų laukų pavadinimų, kurie kartu unikalčiai 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 unikalčiai 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/
<b>Location</b>				code	
		code	integer		ar:code
datasets/gov/ivpk/dp/api		name@en	string		ar:name
			prefix	ar	https://data.gov.lt/id/adresai/

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.distinct()`

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

### Pavyzdys

Turint tokius duomenis duomenų šaltinyje:

CITY	COUNTRY
Vilnius	Lithuania
Kaunas	Lithuania

Ir struktūros aprašą, kuriame COUNTRY aprašytas, kaip atskiras modelis:

model	property	type	ref	source	prepare	level	access
<b>Country</b>			name@en	CITIES	distinct()	4	
	name@en	string		COUNTRY		4	open
<b>City</b>			name@en	CITIES		4	
	name@en	string		CITY		4	open
	country	ref	<b>Country</b>	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 tikėtės 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.

Duomenų tipų pavyzdžiai:

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

#### property.ref

Priklauso nuo `property.type`, nurodo matavimo vienetą, 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	proper- ty	type	ref	uri
dataset1							
					pre- fix	spinta	https://github.com/atviriduomenys/spinta/issues/
						dsa	https://ivpk.github.io/dsa/
						dct	http://purl.org/dc/dcmitype/
dataset2							
					pre- fix	dcat	http://www.w3.org/ns/dcat#
						dct	http://purl.org/dc/terms/
						dcty- pe	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://PLACES						
3	Place						id							
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://PLACES						
6	Place						id							
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	proper- ty	type	ref	source	prepare
			datasets/example/cities places	csv		https://example.com/{ }.c	
			<b>Country</b>		id	countries	
			code	string		CODE	
			title	string		TITLE	
			<b>City</b>		country, title	cities/{code}	
				pa- ram	code	<b>Country</b>	re- ad().code
			count- ry	ref	<b>Country</b>	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 naudojama konstanta nurodyta *param.source* stulpelyje.

#### Pavyzdys

Jei *param.prepare* pateikta formulė *param.read()*, o *param.source* nurodyta Count-ry, 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 pateiktos *source* ir *prepare* reikšmės. Pavyzdžiui anksčiau pateiktą pavyzdį galima būtų perdaryti taip:

data-set	re-source	mo-del	pro-property	type	ref	source	prepare
datasets/example/cities							
	places			csv		https://example.com/{	
		<b>Country</b>			id	countries	
			code	string		CODE	
			title	string		TITLE	
		<b>City</b>			country, title	cities/{country}	
				pa-ram	country	lt	
						lv	
						ee	
			count-ry	ref	<b>Country</b>		pa-ram(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.read(model)`

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

`param.range(stop)`

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

`param.range(start, stop)`

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

`param.path(name, value)`

Parametras pateikia URI path dalies parametą.

*resource.source* URI path parametrų vieta pateikiama {} skliaustų viduje, pavyzdžiui /cities/{id} nurodytas parametras id.

Funkcija gali būti iškviesti tokiais būdais:

- path(value)
- path(name, value)

### Argumentai

#### name

Parametro pavadinimas nurodytas {} riestinių skliaustelių viduje. Jei nenurodyta, tada naudojamas *param.source*, jei šis nenurodytas, tada naudojamas *param.ref* pavadinimas.

#### value

Parametro reikšmė, gali būti statinė arba dinaminė.

#### Pavyzdys

resource	type	ref	source	prepare
resource1	json		/cities/{id}/	
	param	id		path(42)

Bus konstruojamas toks URI:

/cities/42/

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

Parametras pateikia URI query dalies parametą.

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

### 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.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.body(name, value, parent, type)`

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

### Argumentai

#### name

*JSONPath* arba **XPath** išraiška, priklauso nuo *resource.prepare* nurodytos *resource.http()* body tipo.

Nurodoma *param.source* stulpelyje.

#### value

Reikšmė suteikiama name elementui http užklauso 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")

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

### Pavyzdys (XML)

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

```
<?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 atveriamą.

**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 pavadinimą į pavadinimas@lt, o type į text.

### **comment.level**

Nurodoma, kad patenkinus keitimo siūlymą, kuris nurodytas *comment.prepare* stulpelyje, 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 prefiksus. Žiūrėti skrių *Išoriniai žodynai*.

### **comment.title**

Komentaro data, *ISO 8601* formatu.

### **comment.description**

Komentaro tekstas.

## **Pavyzdys**

d	r	b	m	prop- erty	ty- pe	ref	prepare	le- vel	ac- cess	uri
example										
					pre- fix	spin- ta				https://github.com/atviriduomenys
						dsa				https://ivpk.github.io/dsa/
			Imone					2		
					com- ment	ba- se	update(base: "/jar/JuridinisAsmuo" ref: "id")	4	open	spinta:205, mani- fest:1290
					com- ment	ref	update(ref: "id")	4	open	vado- vas:dsa/dimensijos.html#model.re
			id		in- te- ger			4	open	
				pa- va- dini- mas	string			2	open	
					com- ment	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 sąrašė neatsispindi.

id	d	r	b	m	pro- per- ty	type	ref	prepare	le- vel	title	description
1						migra te				2021-12- 21 16:29	Pirmoji migracija.
2						migra te	1			2021-12- 21 16:33	Antroji migracija.
3						migra te	2			2022-06- 21 16:41	Trečioji migracija.
datasets/example/migrate											
Country						id					
id						inte- ger	4				
code						string	3				
						migra te	1	cre- ate(level: 2			
						migra te	3	upda- te(level: 3)			
na- me						string					
						migra te	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(when)**

Naudojamas *property* kontekste, kai vykdoma duomenų migracija. Nurodo, kad migracija taikoma tik *when* 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 ( <i>false</i> ).
1	Teigiama reikšmė ( <i>true</i> ).

#### 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 turētu būti 1, 0.

### type.**integer**

Sveikas skaičius.

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

#### Brandos lygis

##### **L102: Nėra vientisumo**

Duomenys pateikti skirtingais vienetais, pavyzdžiui dalis duomenų pateikti metrais, dalis kilometrais ir dalis milimetrais.

##### **L105: Vienetų konvertavimo paklaida**

Amžius pateiktas atskirai, nurodant metus, mėnesius 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 turētu 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 turėtų 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**

Dvejjetainiai duomenys. Bendras baitų skaičius turi būti ne didesnis nei 1G.

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

**type.string**

Simbolių eilutė. Neriboto dydžio, tačiau fiziškai simbolių eilutė turėtų būti ne didesnė, nei 1G.

Simbolių eilutė turėtų 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 žymę, *property.ref* galima pateikti teksto formatą, naudojant vieną iš šių formatų:

html	tekstas pateiktas HTML formatu.
md	tekstas pateiktas Markdown formatu.
rst	tekstas pateiktas 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 formavimo sintaksę, kuri nėra nurodyta [property.ref](#) stulpelyje.

### type.text

Natūraliąja 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

data-set	model	property	type	ref	uri	title
example						
			pre-fix	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 kokią 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 vieno 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 turėtų 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* stulpelyje 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 reikšmės 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	integer	id
	independence	date	Y

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

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

Šiuo atveju, kadangi datos tikslumas yra metai, -01-01 dalis datoje neturi jokios reikšmės ir yra pateikiama tik tam, kad reikšmė atitiktų 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 reikšmės 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

- 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/longitude) reikšmės;
- 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](https://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 24<sup>o</sup> rytų meridianio, atėmus 500km. Teikiant duomenis, taškai turėtų atrodyti taip: 6000000 500000, pirmas ilgesnis, antras trumpesnis.

Ašinio meridianio 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 absicės pradžia pusiaujuje ir y rytinės ordinatės reikšme 24<sup>o</sup> meridianie 500 000 metrų.

---<https://www.e-tar.lt/portal/lt/legalAct/TAR.6D575923F94A>

Prieš publikuojant duomenis, galite patikrinti, ar koordinačių ašys pateikiamos teisinga

tvarka, naudotami taško atvaizdavimo įrankį.

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

[https://get.data.gov.lt/\\_srid/3346/6061789/582964](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 koordinatinių 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) koordinatinių sistemoje:

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

#### Brandos lygis

##### **L101: Neaiški struktūra**

Pateiktas adresas, nenurodant adreso koordinatinių.

##### **L102: Nėra vientisumo**

Nenurodytas koordinatinių 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 koordinacių sistema, tačiau visi duomenys pateikti naudojant vieno-  
dą koordinacių sistemą.

**L210: Išskaidyta atskirais komponentais**

Taško koordinatės pateiktos, kaip du atskiri duomenų laukai.

**L303: Nenurodytas duomenų tikslumas**

Nenurodytas *property.ref*, kuriame turētu 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 reikšmės 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	proper- ty	type	ref	source	prepare
example								
				Product			PRODUCT	
				amount			PRICE	
				curren- cy			CURREN- CY_CODE	
				price	mo- ney			money(amount, curren- cy)

Šio tipo duomenys pateikiami viena iš šių formų:

123  
123.45

(continues on next page)



(tęsinys iš praeito puslapio)

123 EUR  
123.45 EUR

## type.file

Šis duomenų tipas yra sudėtinis, susidedantis iš tokių duomenų:

### id

Laukas, kuris unikalčiai 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	pro- perty	ty- pe	source	prepare	ac- cess
datasets/example								
				Country				
				name	string	NAME		open
				flag_file_	string	FLAG_FILI		pri- va- te
				flag_file_	bi- na- ry	FLAG_FILI		pri- va- te
				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	prepa- re	ac- cess
datasets/example								
				Country				
				name	string	NAME		open
				flag	file			open
				flag._name		FLAG_FILE_NAME		open
				flag._content		FLAG_FILE_DATA		open

### type.image

Paveikslukas. 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***Asociacija***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
<b>Country</b>	id	integer	id
	name@lt	string	
	cities[]	backref	<b>City</b>
<b>City</b>	id	integer	id
	name@lt	string	
	country	ref	<b>Country</b>

**Taip pat žiūrėkite***Atgalinis ryšys***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**

*Sudėtinis tipas*, apjungiantis kelias savybes į grupę, po vienu pavadinimu.

Šis tipas naudojamas apibrėžti sudėtiniams duomenims, kurie aprašyti naudojant kelis skirtingus tipus. 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**

*Sudėtinis 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](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 subjekto identifikatorius.

[https://en.wikipedia.org/wiki/Uniform\\_Resource\\_Identifier](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 lotyniškąs 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 turėtų 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 veinaskaitos 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 atspindi 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 [\*Jungtinis modelis\*](#).

### **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

geometrijos 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 <sup>2</sup>	4
				turis	number	m <sup>3</sup>	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<sup>2</sup>  
 m<sup>2</sup>  
 km<sup>10</sup>  
 kg·m<sup>2</sup>·s<sup>-3</sup>·A<sup>-1</sup>  
 kg\*m<sup>2</sup>\*s<sup>-3</sup>\*A<sup>-1</sup>  
 8kg·m<sup>2</sup>·s<sup>-3</sup>·A<sup>-1</sup>  
 mg/l  
 g/m<sup>2</sup>  
 mg/m<sup>3</sup>  
 mm  
 U/m<sup>2</sup>  
 U/m<sup>3</sup>  
 %  
 ha  
 min  
 h  
 bar

U  
 $10^6\text{s}$   
 $10^6\text{s}$   
 $\mu/\text{m}^3$   
yr  
3mo  
yr 2mo 4wk  
 $^{\circ}\text{C}$   
 $^{\circ}$

### 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
$\mu$	$10^{-6}$	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

#### Specialieji 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
$\Omega$	ohm
S	siemens
Wb	weber
T	tesla
H	henry
$^{\circ}\text{C}$	degree Celsius
lm	lumen
lx	lux
Bq	becquerel
Gy	gray
Sv	sievert
kat	katal

**Kiti vienetai**

Žymėjimas	Pavadinimas
au	astronomical unit
$^{\circ}$	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
$\hbar$	natural unit of action
$m_e$ arba $m_e$	natural unit of mass

continues on next page

Table 1 – tęsinys iš praeito puslapio

Žymėjimas	Pavadinimas
e	atomic unit of charge
a <sub>0</sub> arba a <sub>0</sub>	atomic unit of length
E <sub>h</sub>	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 Apibendrinimas

Duomenų apibendrinimas, generalizavimas arba paveldimumas yra būdas nurodyti, kad keli skirtingi duomenų modeliai priklauso tai pačiai vienai koncepcinio modelio klasei.

### Taip pat žiūrėkite

- *Koncepcinis modelis / Apibendrinimas*
- *Dimensijos / base*

## 2.9 Asociacija

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.

### 2.9.1 Kompozicija

Kompozicija arba duomenų normalizavimas yra duomenų modeliavimo principas, kai atskiros duomenų klasės yra pateikiamos kaip atskiri duomenų modeliai, kurie gali būti jungiami tarpusavyje, taikant įvairius *duomenų jungimo būdus*.

Pavyzdžiui turint tokią koncepcinio modelio klasių diagramą:

Duomenų modelis, taikant kompoziciją atrodytų taip:

2 Table : Kompozicijos pavyzdys

dataset	model	property	type	ref	level
datasets/gov/rc/ar/ws					
	Country			code	4
		code	string		4
		name@en	string		4
	City				3
		name@en	string		4
		country	ref	Country	4

Taikant kompozicijos principą, kiekvienai klasei kuriamas atskiras nepriklausomas duomenų modelis, o atskiri modeliai jungiami tarpusavyje ryšiais.

### 2.9.2 Jungtinis modelis

Jungtinis modelis yra toks duomenų modelis, kuriame apjungiamos savybės iš daugiau nei vienos koncepcinio modelio klasės. Jungtinis modelis yra priešingas dalykas *kompozicijai*, kur kompozicija išskaido klases į atskirus modelius, jungtinis modelis atvirkščiai apjungia klases į vieną modelį.

Jungtinis modelis dar yra vadinamas agregatu arba denormalizuotu duomenų modeliu.

Jungtinis modelis turi vieną šakninį modelį (angl. *aggregate root*), prie kurio prijungiami kiti modeliai naudojant vieną iš *duomenų jungimo būdų*.

Žiamiau pateikiamas jungtinio modelio pavyzdys.

3 Table : Jungtinio modelio pavyzdys

dataset	model	property	type	ref	level
datasets/gov/rc/ar/ws					
	Country			code	4
		code	string		4
		name@en	string		4
	City				3
		name@en	string		4
		country	ref	Country	4
		country.code			4
		country.name@en			4
		country.name@lt	string		4

Pavyzdyje City yra jungtinis modelis, kadangi City išvardintos ne tik City klasei priklausančios savybės, tačiau įtraukiamos ir kitos klasės Country savybės.

City šiame pavyzdyje yra šakninis modelis, o Country yra modelis, prijungtas per City/country savybę.

### Taip pat žiūrėkite

*Daliniai modeliai (/ :part)*

### Savybių įtraukimas

Kad jungtiniame modelyje nereikėtų kartoti prijungiamo modelio savybių, galima pateikti `expand()` funkciją *model.prepare* arba *property.prepare* stulpeliuose.

Perrašant *Jungtinio modelio pavyzdys* lentelę su `expand()`, gautume, tokią trumpesnę struktūros aprašo lentelę:

4 Table : Jungtinio modelio pavyzdys su expand()

dataset	model	property	type	ref	prepare	level
datasets/gov/rc/ar/ws						
	Country			code		4
		code	string			4
		name@en	string			4
	City					3
		name@en	string			4
		country	ref	Country	expand()	4
		country.name@lt	string			4

Ši lentelė yra lygiai tokia pati kaip ir *Jungtinio modelio pavyzdys*.

Prie City/country nurodyta expand() funkcija į City modelį įtraukia visas Country savybes.

Jei norima įtraukti ne visas Country savybes, reikia naudoti include() funkciją, pateikiant sąrašą savybių, kurias norima įtraukti, pavyzdžiui include(code) - bus įtraukta tik viena Country/code savybė. Kelias savybes galima išvardinti, atskiriant savybių pavadinimus kab-  
leliu.

Prie City modelio yra įtraukta ir City/country.name@lt savybė, kurios nėra Country mode-  
lyje.

expand() įtraukia visas savybes, kurios išvardintos prie modelio, įskaitant ir jungtinio modelio savybes iš kitų klasių.

### 2.9.3 Daugiareikšmiškumas

#### 2.9.4 Duomenų kilmė

Įprastai duomenys yra registruojami vieną kartą pirminiame šaltinyje ir daug kartų pernaudo-  
jami išvestiniuose šaltiniuose. Informacija apie tai iš kokio pirminio šaltinio duomenys pateko  
į išvestinį šaltinį, vadinama duomenų kilme.

Struktūros aprašuose, duomenų kilmė pažymima nepildant *property.type* stulpelio. Jei  
*property.type* yra neužpildytas, nurodoma, kad modelis kuriame pateikta savybė, nėra pir-  
minis šios savybės šaltinis.

Jei modelis yra kito jungtinio modelio dalis arba pirminio modelio dalis, *property.type* stul-  
pelis yra nepildomas, jei savybės modelyje yra pateikiamos tik skaitymui, be galimybės keisti  
savybių reikšmių.

- jei *property.type* užpildytas, tada nurodoma, kad ši savybė yra *pirminis duomenų šal-  
tinis*, tos savybės duomenys gali būti keičiami duomenų modelyje prie kurios savybė yra  
pateikta,
- jei type nenurodytas, tada nurodoma, kad ši savybė nėra *pirminis duomenų šaltinis* ir gali  
būti naudojama tik skaitymui, be galimybės keisti savybės reikšias per modelį, kuriame  
savybė pateikta.



Pavyzdžiui turime jungtinį City modelį:

dataset	model	property	type	ref	level
datasets/gov/rc/ar/ws					
	Country			code	4
		code	string		4
		name@en	string		4
	City				3
		name@en	string		4
		country	ref	Country	4
		country.code			4
		country.name@en			4
		country.name@lt	string		4

Kuriame prie City prijungiama Country klasė. `country.code` ir `country.name@en` neturi *property.type*, nurodant, kad City jungtinis modelis nėra šių savybių pirminis šaltinis ir šios savybės gali būti naudojamos tik skaitymo tikslais.

Tačiau City/country.name@lt turi *property.type*, todėl City jungtinis modelis yra šios savybės pirminis šaltinis.

Jei ta pati savybė turi daugiau nei vieną pirminį šaltinį, tada savybės, kuri nurodo *property.type* ir yra pateikta prie išvestinio arba jungtinio modelio, brandos lygis yra 2, kadangi negali būti du pirminiai duomenų šaltiniai viename objektui.

## 2.9.5 Jungimo būdai

### 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 atrodytu 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.

### **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 koku modeliu jungiame, o code nurodo su koku Country stulpeliu jungiame. Jei pateiktas tik modelis, tada jungiama per to modelio pirminį raktą, jei pateiktas stulpelis laužtiniuose skliaustuose, tada jungiama per nurodytą stulpelį.

### 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.

## Atgalinis ryšys

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
<b>Country</b>			id	4
	id	integer		4
	name@lt	string		4
	cities[]	backref	<b>City</b>	4
<b>City</b>			id	4
	id	integer		4
	name@lt	string		4
	country	ref	<b>Country</b>	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]`.

## 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 apibrėžti nereikia.

### 2.9.6 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 igaus 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	string		4
			City			name	4
				name@lt	string		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. string.

Tačiau, metaduomenyse išliks informacija, apie tai, kad šios lentelės yra susijusios. Dėl prasto duomenų brandos lygio, 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	string		4
				name@en	string		0
			City			name	4
				name@en	string		4
				country	ref	Country[name@en]	1



Šioje vietoje `City.country` tampa `country@en`, kurio tipas yra `string`. O `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 dubliotis, 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	string		4
			City			name	4
				name@lt	string		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 turėtų 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	string		4
				City		name	4
				name@lt	string		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 turėtų 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			
_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		
_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	string		4
				City		name	4
				name@lt	string		4
				country	ref	Country	4

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

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

example/City _id	name@lt	country._id
096e054e-7a4c-44cc-8f27-98af815080d5	Vilnius	4dbb1b77-a930-4f2a-8ef4-f05b89f0fcfe

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

## 2.10 Funkciniai modeliai

Loginis duomenų modelis formuojamas *model* stulpelyje įvardinant *konceptinio modelio* klasės pavadinimą.

*Konceptinis modelis* nėra siejamas su jokių konkrečių duomenų šaltiniu, tačiau loginis modelis yra siejamas su konkrečiu duomenų šaltiniu, tačiau išlaiko koncepciniame modelyje apibrėžtą duomenų struktūrą.

Dažnai viena koncepcinio modelio klasė, loginio modelio pagalba yra siejama su keliais skirtingais duomenų šaltinio prieigos taškais, kurie įgyvendina skirtingas duomenų gavimo funkcijas arba veiksmus.

Tarkime, jei turime tokį koncepcinį modelį:

Ir turime duomenų šaltinį, kuris leidžia duomenis gauti skirtingais metodais:

dataset	resource	model	property	type	ref	source	prepare
datasets/gov/example							
		<b>City</b>			code		
			code	integer			
			name@lt	string			
	get_all_cities			json		/cities	
	get_city_by_code			json		/cities/{code}	

Šiame pavyzdyje turime duomenų struktūros aprašą, kuriame yra City modelis, atitinkantis koncepcinį modelį, nesusietas su jokių duomenų šaltinius ir du duomenų šaltiniai `get_all_cities` ir `get_city_by_code`, nesusieti su loginiu modeliu.

Norint susieti `get_all_cities` ir `get_city_by_code` duomenų šaltinius su loginiu modeliu, mums reikia panaudoti funkcinis modelius, kadangi duomenų šaltinis įgyvendina tik dalį funkcionalumo duomenims gauti.

Galutinis pilnai susietas struktūros aprašas atrodys taip:

dataset	resource	model	property	type	ref	source	prepare
datasets/gov/example							
		<b>City</b>			code		
			code	integer		code	
			name@lt	string		title	
	get_all_cities			json		/cities	
		City:getall				data	
	get_city_by_code			json		/cities/{code}	
		City:getone					
			code	integer			path()

`City:/getall` ir `City:/getone` yra funkciniai modeliai, nurodantys, kad duomenų modelis City yra siejamas su duomenų šaltiniu, kuris įgyvendina tam tikras duomenų skaitymo funkcijas.

Duomenų skaitymo funkcijos sutampa su [UDTS](#) specifikacijoje aprašytais veiksmams, kuriuos galima atlikti su duomenimis.

Funkciniai modeliai paveldi visas savybes iš pagrindinio modelio, tačiau gali papildyti pagrindinį modelį naujomis savybėmis, arba pateikti pagrindinio modelio savybes su kitais meta-duomenimis, tarkim nurodant kitokią *property*.*source* reikšmę.

Jei nenurodytas joks funkcinis modelis, daroma prielaida, kad šaltinis palaiko visas **UDTS** funkcijas.

### 2.10.1 Funkcijos

#### getall

Nurodo, kad duomenų šaltinis leidžia gauti visus klasės objektus, netaikant jokių filtrų.

##### Pavyzdys

dataset	resource	model	property	type	ref	source	prepare
datasets/gov/example							
		<b>City</b>			code		
			code	integer		code	
			name@lt	string		title	
	get_all_cities			json		/cities	
		City/:getall				data	

#### getone

Nurodo, kad duomenų šaltinis leidžia gauti vieną klasės objektą nurodžius objekto identifikatorių.

##### Pavyzdys

dataset	resource	model	property	type	ref	source	prepare
datasets/gov/example							
		<b>City</b>			code		
			code	integer		code	
			name@lt	string		title	
	get_city_by_code			json		/cities/{code}	
		City/:getone					
			code	integer			path()

getone veiksmo atveju, modelis turi turėti pirminį raktą, nurodytą *model.ref* stulpelyje. Pirminis raktas gali būti paveldimas iš pagrindinio modelio arba gali būti nurodomas kitas pirminis raktas prie funkcinio modelio.

Sąsajai su duomenų šaltiniu, pirminio rakto savybės turėtų nurodyti funkciją, kuri siejama su duomenų šaltinio parametrais. Konkrečiai, pavyzdyje aukščiau prie `City/:getone/code` savybės yra nurodyta `path()` funkcija `property.prepare` stulpelyje, kuri nurodo, kad `code` savybė yra naudojama kaip duomenų šaltinio URI `path` parametras, pažymėtas *resource.source* stulpelyje `{}` riestiniuose skliausteliuose, tuo pačiu pavadinimu, kaip ir *property* pavadinimas.

Per UDTS protokolą, bus tikimasi gauti tokią užklausą:

```
GET /datasets/gov/example/City/87a1d91a-e00d-4981-8287-d1810243d160 HTTP/1.1
```

Tokia UDTS užklausa, pagal pateiktą duomenų struktūros aprašo pavyzdį, bus konvertuota į tokią duomenų šaltiniui skirtą užklausą:

```
GET /cities/42 HTTP/1.1
```

Kad tai veiktų, duomenų agentas, turi saugoti išorinių ir vidinių identifikatorių lentelę, kurioje yra susietas išorinis `87a1d91a-e00d-4981-8287-d1810243d160` su vidiniu `42`, kas leidžia konvertuoti tarp vidinių ir išorinių identifikatorių.

5 Table : Identifikatorių susiejimas

_id (išorinis)	id (vidinis)
87a1d91a-e00d-4981-8287-d1810243d160	42

Visų duomenų šaltinių lokalūs identifikatoriai, siejami su vienu esybės išoriniu identifikatoriumi.

### search

Nurodo, kad duomenų šaltinis leidžia gauti ne visus klasės objektus, o tam tikrą objektų imtį, pagal nurodytą duomenų filtrą.

### Statiniai filtrai

Statiniai filtrai nurodo, kad duomenys pateikiami naudojant konkrečias filtrų reikšmes nurodytas struktūros apraše.

#### Pavyzdys

data-set	resource	model	property	type	ref	source	prepare
datasets/gov/example							
		<b>Country</b>			code		
			code	string			
		<b>City</b>			code		
			code	integer		city_code	
			name@lt	string		city_name	
			country	ref	Country		
			country.code	string			"lt"
	get_all_cities			json		/cities	
		City?country.code="lt"					

Pavyzdyje nurodyta, kad funkcinis `City?country.code="lt"` modelis grąžina ne visus duomenis, o tik Lietuvos miestų duomenis.

### Dinaminiai filtrai

Dinaminiai filtrai nurodo, kad duomenys pateikiami naudojanti filtrų reikšmes, kurias pateikia duomenų naudotojas, per [UDTS](#) užklausa, pateikti duomenys konvertuojami ir perduodami duomenų šaltiniui.

Dinaminiai filtrai veikia lygiai taip pat, kaip ir statiniai filtrai, tik dinaminio filtro atveju, ne-nurodoma statinė reikšmė.

Tarkime statinis `country.code="lt"` filtras gali būti dinaminis pašalinus `"lt"` dalį ir paliekant tik `country.code`.

### Pavyzdys

data-set	resource	model	property	type	ref	source	prepare
datasets/gov/example							
		<b>Country</b>			code		
			code	string			
		<b>City</b>			code		
			code	integer		city_code	
			name@lt	string		city_name	
			country	ref	Country		
			country.code	string		country_code	
	get_cities_by_country			json		/cities	
		City?country.code					
			country.code	string		country	query()

Šiame pavyzdyje turime `City?country.code` funkcinį modelį, kuriame naudojamas dinaminis filtras pagal `country.code`.

Kadangi `City?country.code` modelis nurodo `country.code` savybę su *query()* formule *property.prepare* stulpelyje, tai reiškia, kad konvertuojant užklausą į duomenų šaltinio protokolą, `country.code` reikšmė bus perduota kaip URI query parametras.

Per UDS protocolą, bus tikimasi gauti tokią užklausą:

```
GET /datasets/gov/example/City?country.code="lt" HTTP/1.1
```

Tokia UDS užklausa, pagal pateiktą duomenų struktūros aprašo pavyzdį, bus konvertuota į tokią duomenų šaltiniui skirtą užklausą:

```
GET /cities?country="lt" HTTP/1.1
```

## Operatoriai

Funkciniam modeliui galima perduoti daugiau nei vieną filtro parametą, naudojant skirtingus filtravimui skirtus operatorius.

### Taip pat žiūrėkite

*Duomenų atranka*

Pavyzdžiui funkcinis modelis naudojantis dinaminį filtravimą pagal du kriterijus atrodytų taip:



City?country.code&code

Šiuo atveju, duomenys pateikiami naudojant filtrą pagal šalies ir miesto kodus.

## select

Nurodo, kad duomenų šaltinis grąžina ne visas klasės savybes, o tik tam tikras.

### Pavyzdys

dataset	resource	model	property	type	ref	source	prepare
datasets/gov/example							
		City			code		
			code	integer		city_code	
			name@lt	string		city_name	
	get_cities			json		/cities	
		City?select(code)					

Pavyzdyje nurodoma, kad `get_cities` duomenų šaltinis grąžina ne visas `City` klasės savybes (`code` ir `name`), o tik vieną `code`.

`select()` funkcijai galima nurodyti kelias savybes, atskiriant jas kableliu:

`select(code, name@lt)`

Taip pat galima naudoti ir kitas savybių atrankos funkcijas.

### Taip pat žiūrėkite

`select()`

Kitos savybių atrankos funkcijos:

`include()`

`exclude()`

`expand()`

## sort

Nurodo, kad duomenų šaltinis grąžina surūšiuotus duomenis, pagal tam tikras savybių reikšmes.

**Taip pat žiūrėkite**
`sort()`
**part**

Nurodo, kad duomenų šaltinis neleidžia tiesiogiai pasiekti modelio duomenų ir šis modelis yra naudojamas tik kaip sudėtinė, vieno ar kelių kitų *jungtinių modelių*.

**Pavyzdys**

dataset	resource	model	property	type	ref	source	prepare
datasets/gov/example							
		<b>City</b>			code		
			code	integer		city_code	
			name@lt	string		city_name	
	get_cities			json		/cities	
		City?select(code)					
			id	integer		page	query()

## 2.11 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.

### 2.11.1 L000: Duomenų nėra

#### Nekaupiami

Duomenys nekaupiami. Duomenų rinkinys užregistruotas duomenų portale. Užpildyta *data-set* 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							
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.

**2.11.2 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 imones_id	imones_pavadinimas	rusis
42	UAB "Įmonė"	1

Filialas ikurimo_data	atstumas	imones_id_id	imo- 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	„IMONĖ UAB	1“, 037034536522

Struktūros aprašas	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 duomenų 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.

### 2.11.3 L200: Nestandartinis pateikimas

Duomenys kaupiami struktūruota, mašiniu būdu nuskaityama forma, bet koku 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		
imones_id	imones_pavadinimas	rusis
42	UAB "Įmonė"	1

Filialas				
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 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, turėtų būti date.
- imones\_pavadinimas - nurodytas string, turėtų būti text.
- atstumas - nurodytas string, turėtų 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, turėtų būti YYYY-MM-DD.
- atstumas - nurodyta X m., turėtų būti X.
- tel\_nr - nurodytas (XX) XXX XXXXX, turėtų 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, turėtų būti `id`.
- `imones_pavadinimas` - dubliuojamas modelio pavadinimas, turėtų būti pavadinimas.
- `ikurimo_data` - dubliuojamas tipo pavadinimas, turėtų 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` turėtų būti `Filialas.imone.imones_id`.
- `Filialas.imones_pavadinimas` turėtų būti `Filialas.imone.imones_pavadinimas`.

Plačiau apie denormalizuotus duomenis skaitykite skyriuje *Jungtinis modelis*.

### 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 turėtų 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` turėtų būti `Imone.kodas`, kad sutaptu su baze (`JuridinisAsmuo.kodas`).
- `Imone.imones_pavadinimas` turėtų 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.

**2.11.4 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 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, turētu būti D - vienos dienos tikslumas.

**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.

**2.11.5 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 *Jungimo būdai*).
- Jei duomenų laukas yra kiekybinis ir turi matavimo vienetų (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 meta-duomenimis, kurie lydi duomenis.

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

**Pavyzdžiai**

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

Filialas _id	id	ikur- ta	at- stu- mas	imone._id	imo- ne.id	imo- ne.pavadinis	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 d r b m	property	type	ref	le- vel	prepa- re	title
example						
	JuridinisAsmuo		kodas	4		
		kodas	inte- ger	4		
		pavadinimas@lt	text	4		
	JuridinisAsmuo			4		
	Imone		kodas	4		
		id	inte- ger	4		
		pavadinimas@lt	text	4		
		rusis	inte- ger	4		
			enum		1	Juridi- nis
					2	Fizinis
/						
	Filialas		id	4		
		id	inte- ger	4		
		ikurta	date	D	4	
		atstumas	inte- ger	km	4	
		imone	ref	Imo- ne	4	
		imone.id		4		
		imo- 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.

### 2.11.6 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	ikur- ta	at- stu- mas	imone._id	imo- ne.id	imo- ne.pavadinis	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							
		pre- fix	foaf		http://xmlns.com/		
			dct		http://purl.org/dc/		
			sche- ma ko- das		http://schema.org		
JuridinisAsmuo				4			
	kodas	inte- ger		4			
	pavadinimas@lt	text		4			
JuridinisAsmuo				4			
Imone			id	5	fo- af:Organization		
	id			5	dct:identifier		
	pavadinimas@lt			5	dct:title		
	rusis	inte- ger		4			
		enum				1	Juri- dinis
						2	Fizi- nis
/							
Filialas			id	5	sche- ma:LocalBusiness		
	id	date	1D	5	dct:identifier		
	ikurta	date	1D	5	dct:created		
	atstumas	inte- ger	km	5	sche- ma:distance		
	imone	ref	Imo- ne	5	fo- af:Organization		
	imone.id	inte- ger		5	dct:identifier		
	imo- ne.pavadinimas	text		5	dct:title		
	tel_nr	string		5	foaf:phone		

## 2.12 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 viešajame 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 turėtų 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.

## 2.13 Duomenų schemas

Duomenų struktūra aprašoma schemas pagalba.

Žemiau rasite sąrašą schemų kalbų su aprašymais, kaip kiekviena schemų kalba siejama su duomenų struktūros aprašu.

### 2.13.1 Schemų sąrašas

#### **XSD**

`XML Schema Definition (XSD)` yra schemų kalba skirta XML duomenims aprašyti.



XSD specifikacija susideda iš šių dalių:

- W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures
- W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes

## Jungtinis modelis

Jungtiniai modeliai (angl. *Aggregate Model*) yra modelis, kuris yra sujungtas iš vieno ar kelių papildomų modelių.

## Šakninis modelis

Šakninis modelis (angl. *Aggregate Root*) yra jungtinio modelio pradžios taškas, per kurį pasiekiami kitų jungtinio modelio sudėtyje esančių modelių duomenys.

## Dalinis modelis

Dalinis modelis (angl. *Aggregate Part*) yra Jungtinio modelio sudedamoji dalis ir atskirai nenaudojamas duomenims gauti. Dalinio modelio duomenys yra teikiami tik kaip jungtinio modelio dalis.

Dalinio modelio atveju, nėra pildomas *model.source*, kadangi dalinio modelio duomenys gali būti pasiekiami tik per jungtinį modelį.

### Pavyzdys

#### Duomenys

```
<country name="Lietuva">
  <city name="Vilnius" />
</country>
```

#### Schema

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:complexType name="city">
    <xs:attribute name="name" type="xs:string" />
  </xs:complexType>

  <xs:element name="country">
    <xs:complexType>
      <xs:sequence>
        <xs:element type="city" maxOccurs="unbounded" />
        <xs:attribute name="name" type="xs:string" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

#### Struktūros aprašas

dataset	model	property	type	ref	source	prepare	level
			schema	xsd	country.xsd		
xsd							
	<b>Country</b>				/country		0
		name	string		@name		
		cities[]	backref	City	city	expand()	
	<b>City::part</b>						0
		name	string		@name		0

Pavyzdyje:

- Country modelis yra *Jungtinis modelis* ir jungtinio modelio *Šakninis modelis*.
- City yra *Dalinis modelis*, kadangi tai žymi `/:part` žymė, taip pat City neturi užpildyto *model.source* stulpelio, tai reiškia, kad tiesiogiai City duomenų gauti galimybės nėra, juos galima gauti tik per Country jungtinį modelį, kurio sudėtyje yra ir City, prieinamas per Country/cities savybę.
- Country/cities savybė turi *expand()* funkciją įrašytą į *property.prepare*, kuri įtraukia visas tiesiogines City savybes į jungtinį Country modelį.

## Elementai

### element

XSD *element* atitinka DSA loginio modelio:

- *model* - jei elemento tipas yra *complexType*,
- *property* - jei elemento tipas yra *simpleType*.

Jei *xsd\_element* tipas nėra nurodytas, tada pagal XSD specifikaciją elemento tipas yra *xs:anyType*. DSA neturi *xs:anyType* analogo, todėl tokiu atveju turėtų būti naudojamas DSA *string* tipas, kur *xs:anyType* reikšmė yra pateikiama, kaip tekstinė reprezentacija.

Priklausomai nuo to, kur *element* yra deklaruotas *complexType* atžvilgiu, pagal nutylėjimą atliekama sekanti XSD interpretacija:

- Jei *element* (*complexType* tipo) yra *complexType* sudėtyje, laikoma, kad modelis yra kito *Jungtinis modelis* dalis, todėl pagal nutylėjimą nenurodomas *model.source* ir šis modelis žymimas kaip dalinis, prie jo pavadinimo pridedant žymę ``/:part`.
- Jei *element* (*complexType* tipo) nėra *complexType* sudėtyje ir deklaruotas atskirai, bet XSD schemoje yra bent vienas *complexType*, kurio viduje yra elementas, turintis atributą *ref*, kurio reikšmė yra šio elemento pavadinimas, tada laikoma, kad modelis yra kito *Jungtinis modelis* dalis, todėl pagal nutylėjimą nenurodomas *model.source* ir šis modelis žymimas kaip dalinis, prie jo pavadinimo pridedant žymę `/:part`.
- Jei *element* (*complexType* tipo) nėra *complexType* sudėtyje ir yra deklaruotas atskirai, tada laikoma, kad modelis gali būti pasiekiamas tiesiogiai ir tokio modelio *model.source*

yra pildomas bei šis modelis nežymimas kaip dalinis.

- Jei *simpleType* tipo *element* yra *complexType* sudėtyje, iš jo sukuriamas *property*, ir pridedamas prie *model*, kuris kuriamas iš elemento, kurį aprašo šis *complexType*. Šios *property.source* formuojamas iš elemento pavadinimo, prie jo pridedant `/text()`
- Jei *simpleType* tipo *element* nėra *complexType* sudėtyje, bet yra deklaruotas atskirai, bei nėra nei vieno kito elemento, kuris per `ref` ar `type` referuotų į šį elementą, iš jo sukuriamas *property*, ir pridedamas prie specialaus Resource *model*.

### Pavyzdys

#### Duomenys

```
<Country>
  <name>France</name>
  <numberOfMunicipalities>35</numberOfMunicipalities>

  <City>
    <name>Paris</name>
  </City>

  <City>
    <name>Lyon</name>
  </City>

</Country>
```

#### Schema

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="Country">
    <xs:complexType>
      <xs:sequence>

        <xs:element name="name" type="xs:string" />

        <xs:element name="numberOfMunicipalities">
          <xs:simpleType>
            <xs:restriction base="xs:integer">
              <xs:minInclusive value="1" />
            </xs:restriction>
          </xs:simpleType>
        </xs:element>

        <xs:element ref="governance"/>

        <xs:element name="City" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="name" type="xs:string" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>

      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="governance">
    <xs:complexType>
```

### Struktūros aprašas

Pavyzdyje:

- Country element tampa modeliu, nes jis yra pirminio lygio, ir jo tipas yra *complexType*. XML struktūroje jis tampa šakniniu elementu, todėl iš jo kilęs modelis irgi nurodomas kaip šakninis modelis, galintis eiti atskirai, ir nėra žymimas `/:part`.
- name element tampa *model* Country *property*, nes jis yra viduje *complexType*, kuris yra viduje

**Country element ir pats yra *simpleType*. Jo tipas šiuo atveju nurodomas paties elemento aprašyme**

ir yra string. Šis string tipas DSA taip pat tampa string tipu.

- numberOfMunicipalities element taip pat tampa Country modelio *property*. Jam tipas nurodytas atskirame *simpleType*, kuriame nurodoma, kad jo pagrindas (base) yra integer, ir nurodyti apribojimai (restriction). Šis base tipas ir yra konvertuojamas į DSA tipą, šiuo konkrečiu atveju - į integer tipą. Kadangi DSA netaiko apribojimų reikšmėms, tai visi apribojimai, kurie yra nurodyti restriction (šiam pavyzdyje, minInclusive) ignoruojami. Kadangi *property* pavadinimas turi būti sudarytas iš mažųjų raidžių, o tarpai tarp žodžių atskiriami pabraukimais (\_), tai *property* pavadinimas tampa number\_of\_municipalities.
- Sekantis elementas, `<xs:element ref="governance"/>`, neturi pavadinimo, bet jame yra atributas ref, kas nurodo, kad jo aprašymas referuojamas kitam, globaliam elementui, pavadinimu governance. Šiuo atveju iš šio element kuriamai *property* suteikiamas pavadinimas pagal ref atributą, o į jo ref stulpelį įrašomas modelio, sukurto iš referuojamo element, pavadinimas.
- iš atskirai apibrėžto elemento `<xs:element ref="governance"/>` sukuriamas *model* Governance.

### Santrauka: XSD elementų ir DSA Atitikimas

#### attribute

XSD *attribute* atitinka DSA loginio modelio *property*.

Iš *attribute* atributo name formuojamas *property* pavadinimas. Jei name susideda iš kelių žodžių, *property* pavadinimas taip pat susidės iš kelių žodžių, tačiau jie bus mažosiomis raidėmis ir atskirti pabraukimo ženklų (\_).

*property.source* yra formuojamas iš *attribute* atributo name, priekyje pridant @.

Jei attribute tipas yra nurodytas attribute elemente esančiu *type* atributu, tai *property* tipas formuojamas iš *attribute*, naudojantis `xd_type_conversion`.

Jei attribute tipas aprašytas *simpleType*, tai *property.type* formuojamas iš šio simpleType viduje esančio *restriction base* nurodyto *type*.

Jei attribute elemento sudėtyje yra annotation, iš jo formuojamas aprašymas - *property.description*.

Jei attribute turi atributą *use* su reikšme required, tai prie *property* pavadinimo pridedama required žymė, reiškianti, kad ši *property* yra privaloma.

**Pavyzdys****Duomenys**

```
<country name="France" capital="Paris" />
```

**Schema**

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:element name="country">
    <xs:complexType>
      <xs:attribute name="name" type="xs:string" use="required"/>
      <xs:attribute name="capital" type="xs:string" use="required"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

**Struktūros aprašas**

dataset	model	property	type	ref	source	prepare	level
			schema	xsd	country.xsd		
xsd							
	<b>Country</b>				/country		0
		name	string		@name		
		capital	string		@capital		

Pavyzdyje:

- XSD attribute kurio name reikšmė yra name tampa *property* su pavadinimu name. Jo tipas yra string, tai konvertuojasi į DSA *property.type* string. *property.source* padaromas iš attribute name name, prie jo pridedant @ ir tampa @name.
- XSD attribute kurio name reikšmė yra capital tampa *property* su pavadinimu capital. Jo tipas yra string, todėl konvertuojasi į DSA *property.type* string. *property.source* padaromas iš attribute name capital, prie jo pridedant @ ir tampa @capital.

**simpleType**

Jei elemento ar atributo tipas aprašytas naudojant simpleType, į DSA tipą jis konvertuojamas naudojant konvertavimo lentelę `xsd_type_conversion`.

simpleType viduje gali būti restriction arba extension. Jie abu naudojami smulkesniam simpleType aprašymui. Dauguma jų naudojami duomenų validavimui, o DSA duomenų validavimo taisyklės netaikomos, tai šie apribojimai dažniausiai yra ignoruojami. Jų aprašymus rasite žemiau.

Jei simpleType elementas turi *annotation*, jo turinys pridedamas prie iš šį simpleType naudojančio elemento sukurtos property aprašymo: *property.description*.

simpleType gali būti aprašomas ir atskirai. Tokiu atveju, iš jo nustatytas *property.type* bus pridėtas toms *property*, kurios sukurtos iš į šį tipą referuojančių elementų arba atributų.

### Pavyzdys

#### Duomenys

```
<numberOfMunicipalities>5</numberOfMunicipalities>
```

#### Schema

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="country">
    <xs:complexType>
      <xs:element name="population">
        <xs:simpleType>
          <xs:restriction base="xs:integer">
            <xs:minInclusive value="1" />
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

#### Struktūros aprašas

Pavyzdyje:

- iš simpleType, kurio viduje nurodytas *restriction*, kurio *base* yra string, gaunamas DSA *property.type* string.

## complexType

*complexType* gali būti arba *element* sudėtyje, arba atskirai. Jei *complexType* yra *element* sudėtyje, iš jų abiejų kartu kuriamas *model*. *model* struktūra nustatoma iš *complexType*, o pavadinimas - iš *element* atributo name.

Jei *complexType* yra aprašytas atskirai, iš jo kuriamas *model*, kurio pavadinimas nustatomas iš *complexType* pavadinimo.

*complexType* gali turėti atributą *mixed*. Jis reiškia, kad šiuo *complexType* aprašytas *element* turės galimybę viduje turėti teksto. Tokiu atveju, prie *model* pridedama *property* su pavadinimu text ir tipu string. Jos *property.source* yra text().

Jei *complexType* sudėtyje yra *choice* elementas ir šio elemento atributas *maxOccurs* yra daugiau, nei 1 arba yra unbounded, iš šio *complexType* kuriama po vieną *model* kiekvienam *choice* pasirinkimui, kai šis pasirinkimas pridedamas prie kitų *property*.

Jei *complexType* viduje yra *complexContent*, kurio viduje yra *extension*, kurio *base* rodo į kitą, atskirai apibrėžtą elementą, tai prie *model*, stulpelyje prepare, nurodoma funkcija **function: `extends()`**, jos parametru nurodžius *model*, kuris buvo sukurtas iš to tipo.

*complexType* sudėtyje gali būti įvairios konstrukcijos, aprašančios atributus ir elementus, iš kurių šiam *model* formuojamos *property*:

- attribute
- sequence
- choice
- all
- simpleContent
- complexContent

## sequence

sequence elementas būna complexType sudėtyje. Jis nurodo element elementų seką. Kiekvienas sequence viduje esantis element yra apdorojamas, ir iš jo sukurta savybė ar savybės pridedamos prie iš complexType sukurto modelio.

Jei sequence viduje yra *choice*, kurio *maxOccurs* yra "1", tai kiekvienam šio *choice* pasirinkimui iš jo ir likusių savybių kuriamas atskiras modelis.

Jei sequence turi atributą *maxOccurs* ir jo reikšmė yra daugiau nei 1 ar yra "unbounded", tai kiekviena *property*, sukurto iš sequence viduje esančių elementų, tampa masyvu, kas reiškia, kad prie jos pavadinimo prisideda `[]`, o jei jos tipas būtų buvęs ref, jis pasikeičia į backref.

## choice

Jei choice elemento atributas *maxOccurs* yra lygus "1", tai choice verčiamas į DSA lygiai taip pat, kaip ir *sequence*.

Jei choice elemento atributas *maxOccurs* yra daugiau nei "1" arba yra "unbounded", tai su kiekvienu šio choice viduje esančiu pasirinkimu (tai gali būti *element*, *sequence* ar kitas choice) bus kuriamas atskiras *model* iš *complexType*, kurio sudėtyje yra šis choice (tiesiogiai, ar kito *sequence* ar choice viduje).

### Pavyzdys

#### Duomenys

Pirmas variantas:

```
<country>
  <population>800000</population>
  <area>700.5</area>
  <king_or_queen>Elžbieta II</king_or_queen>
</country>
```

Antras variantas:

```
<country>
  <population>1000000</population>
  <area>500.0</area>
  <president>Ona Grybauskaitė</president>
</country>
```

#### Schema

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="country">
    <xs:complexType>
      <xs:element name="population" type="xs:integer" />
      <xs:element name="area" type="xs:decimal" />
      <xs:choice maxOccurs="1">
        <xs:element name="president" type="xs:string" />
        <xs:element name="king_or_queen" type="xs:string" />
      </xs:choice>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

**\*\*Struktūros aprašas - \*\***

## all

Elementas *all* reiškia, kad jo viduje aprašyti elementai turi eiti nurodyta tvarka, ir maksimaliai gali būti po 1 kartą. Minimaliai gali būti taip, kaip nurodyta prie kiekvieno elemento naudojant *minOccurs*. Taigi, iš kiekvieno *all* viduje esamų elementų bus kuriama savybė, ir galbūt modelis, kaip nurodyta *element*.

## complexContent

*complexContent* būna *complexType* viduje ir aprašo sudėtinį turinį.

*complexContent* viduje būna *extension*, kuris turi atributą *base*. Šis atributas nurodo, kio kito tipo pagrindu kuriamas šis tipas. Iš *base* nurodomo tipo sukurtas modelis įdedamas į iš šio *complexContent* tėvinio *complexType* prepare stulpelyje nurodomą funkciją **:function:extends()**.

*extension* viduje gali būti *sequence*, *choice* ir *all*, o taip pat *attribute*.

Iš šių *attribute* bei iš *sequence*, *choice* ir *all* viduje esančių *element* kuriamos savybės, ir pridamos prie iš *complexType* sukurto modelio, pagal tas pačias taisykles, kaip ir iš tiesiogiai *complexType* esančių tokių pačių elementų.

### Pavyzdys

#### Duomenys

```
<example>
  <country id="C1">
    <name>Lithuania</name>
    <capital>Vilnius</capital>
  </country>

  <city id="CT1">
    <name>Kaunas</name>
    <country>Lithuania</country>
```



```
</city>
</example>
```

### Schema

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault=
  →"qualified">
```

```
  <xs:complexType name="Place">
    <xs:sequence>
      <xs:element name="name" type="xs:string"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" use="required"/>
  </xs:complexType>
```

```
  <xs:complexType name="Country">
    <xs:complexContent>
      <xs:extension base="Place">
        <xs:sequence>
          <xs:element name="capital" type="xs:string"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
```

```
  <xs:complexType name="City">
    <xs:complexContent>
      <xs:extension base="Place">
        <xs:sequence>
          <xs:element name="country" type="xs:string"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
```

```
  <xs:element name="country" type="Country"/>
  <xs:element name="city" type="City"/>
```

```
</xs:schema>
```

**\*\*Struktūros aprašas - \*\***

dataset	model	property	type	ref	source	prepare	level
xsd							

Pavyzdyje:

- complexType Place tampa *model* Place, o complexType Place viduje esantis *element* name ir *attribute* id tampa jo savybėmis (*property*).
- complexType City tampa *model* City, o complexType City viduje esančiame **:rex: `xsd\_extension`** esantis *element* country tampa *property* country.  
*extension base* atributas nurodo į complexType Place, todėl iš jo sukurtas *model* Place nurodomas *model* City prepare stulpelyje esančioje **:function: `extends`** funkcijoje. Tai reiškia, kad vėliau, interpretuojant šią DSA, visos *model* Place esančios *property* įtraukiamos į *model* City.
- analogiškai su Country.

### simpleContent

simpleContent elementas būna viduje complexType elemento. Viduje simpleContent elemento gali būti arba *restriction* arba *extension* elementas.

Jei simpleContent viduje naudojamas *extension*, tai *extension* viduje nurodomi *attribute*. Iš kiekvieno jų kuriama *property* ir pridedama prie *model*, sukurto iš *xsd\_complex\_type*, kurio viduje yra. Taip pat, prie modelio, sukurto iš *xsd\_complex\_type* pridedama *property* pavadinimu text ir jai priskiriamas tipas, kuris gaunamas iš *base*, pagal tipų siejimo lentelę *xsd\_type\_conversion*.

Jei simpleContent viduje naudojamas *restriction*, tai reiškia, kad tipas, kurio viduje yra šis mazgas, yra apribojamas. Apribojimai gali būti tokie, kaip minimalios ar maksimalios reikšmės, ilgis ar kitos duomenų validacijos taisyklės. Dauguma jų yra ignoruojami, nes DSA duomenų reikšmių apribojimui įrankių neturi. Tačiau, jei *restriction* viduje yra enumeration, tai išvardintos reikšmės perkeliama į *enum*. Išsamiau paaiškinta prie *enumeration*.

### enumeration

enumeration išvardija reikšmes, iš kurių gali būti pasirenkama *element* arba *xsd\_attribute* reikšmė. DSA jo atitiktis yra *enum*. enumeration būna *xsd\_simple\_type* sudėtyje esančio *restriction* viduje, o šis *xsd\_simple\_type* aprašo *element* arba *attribute* tipą. Taigi, iš enumeration gautas reikšmių sąrašas perkeliama į DSA savybės, suformuotos iš *element* arba *attribute* *enum* reikšmės.

#### Pavyzdys

##### Duomenys

Pirmas variantas:

```
<country>
  <population>800000</population>
  <area>700.5</area>
  <king_or_queen>Elžbieta II</king_or_queen>
</country>
```

Antras variantas:

```
<country>
  <population>1000000</population>
  <area>500.0</area>
  <president>Ona Grybauskaitė</president>
</country>
```

## Schema

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDe-
fault="qualified">
  <xs:element name="Country">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="name" type="xs:string" /> <xs:element na-
me="head_of_state">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="President" /> <xs:enumeration va-
lue="Monarch" /> <xs:enumeration value="PrimeMinister" />
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

**\*\*Struktūros aprašas - \*\***

dataset	model	property	type	ref	source	prepare	level
			schema	xsd	country.xsd		
xsd							
	<b>Country</b>				/country		0
		name	string		name/text()		
		head_of_state	string		head_of_state/text()		
			enum		President		
					Monarch		
					PrimeMinister		

### annotation

annotation viduje būna informacija apie elementą, kurio viduje jis yra. Jo viduje gali būti elementai *documentation* ir *union*. *union* elementas ignoruojamas, o *documentation* viduje esantis tekstas perkeliamas į lauką *property.description* arba į *model.description*, kuris kuriamas iš element ar attribute, kurio viduje annotation yra.

#### Pavyzdys

##### Duomenys

```
<Country name="Lithuania">
  <Capital>Vilnius</Capital>
</Country>
```

##### Schema

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="Country">
    <xs:annotation>
      <xs:documentation>
        Represents a country, with its name as and its capital.
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Capital" type="xs:string">
          <xs:annotation>
            <xs:documentation>
              Represents the capital city of the country.
            </xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="name" type="xs:string" use="required">
        <xs:annotation>
          <xs:documentation>
            Specifies the name of the country.
          </xs:documentation>
        </xs:annotation>
      </xs:attribute>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

#### Struktūros aprašas

### documentation

documentation elementas visada būna viduje *annotation* elemento ir iš jų abiejų kartu formuojamas aprašymas - *model.description* arba *model.description*. Daugiau informacijos prie *annotation*

**restriction**

**restriction** yra taikomas galimų duomenų reikšmių ribojimui, kaip pavyzdžiui, minimalioms ar maksimalioms reikšmėms, teksto ilgio ribojimams. DSA šie ribojimai neaprašomi, taigi dauguma šių žymių ignoruojama.

Vienintelis atvejis, kai **restriction** perkeliamas į DSA, yra, kai jis naudojamas kartu su *enumeration*. Tai naudojama aprašyti išvardijamoms galimoms reikšmėms. Daugiau aprašyta prie *enumeration*.

**extension**

**extension** mazgas visada eina viduje `xsd_simple_type` arba `xsd_complex_type`. Kaip jis veikia šiuose mazguose, aprašyta prie jų.

**union****union****Atributai****base**

**base** naudojamas viduje *extension* arba *restriction*, kai norima išplėsti arba susiaurinti tam tikro tipo reikšmes. Daugiau apie **base** naudojimą aprašyta prie *complexContent* ir prie *simpleContent*.

**unique****minOccurs**

**minOccurs** naudojamas nurodyti elemento minimalų pasikartojimų skaičių. Pagal šį atributą taip pat galim nustatyti, ar elementas privalomas, ar ne. Jei **minOccurs** yra lygus 0, tai elementas neprivalomas, o jei didesnis nei 0, elementas privalomas. DSA privalomumas nurodomas prie `property.name` pridėdant `required` jei ji yra privaloma, arba nepridėdant nieko, jei neprivaloma.

Jei *element* turi atributą **minOccurs**, kurio reikšmė lugi '0', reiškia iš šio elemento sukurta savybė yra neprivaloma, ir žymė `required` nepridedama, o jei **minOccurs** atributo reikšmė yra 1 arba didesnė, arba jei šis atributas visai nenurodytas (pagal nutylėjimą jo reikšmė lygi 1), reiškia, kad iš jo sukurta savybė yra privaloma ir prie jos pavadinimo pridėdama žymė `required`.

**maxOccurs**

**maxOccurs** žymi, kiek daugiausiai kartų elementas gali pasikartoti. Jei ši reikšmė yra 1, arba jei šis atributas iš viso nenurodytas (jo numatytoji reikšmė yra 1), reiškia elementas gali būti tik vieną kartą. Tokiu atveju, DSA tai yra įprasta, vieną objektą ar savybę žyminti *property*. Jei **maxOccurs** reikšmė yra daugiau nei 1 arba unbounded, tai reiškia, kad elementas gali pasikartoti daug kartų, tai iš jo padaryta *property* bus masyvas, ir prie jos pavadinimo bus pridėti laužtiniai skliaustai ([ ]), o jei tai yra į kitą *model* rodanti savybė, tai jos tipas bus ne *ref*, bet *backref*.

**nillable****type**

XSD tipas gali būti nurodytas pačiame elemente, nurodant atributą `type`, arba aprašytas po jo einančiame `simpleType` arba `complexType`.

Jei tipas aprašytas pačiame elemente ar attribute, į DSA *property.type* jis konvertuojamas naudojant konvertavimo lentelę `xsd_type_conversion`.

Taip pat elemento tipas gali būti aprašytas naudojant `simpleType` ir `complexType`.

**use**

`use` naudojamas aprašant *attribute*, ir nurodo, ar elementas yra privalomas, ar ne. Jei `use` nenurodytas, naudojama jo numatytoji reikšmė, kuri yra `optional`, ir tai reiškia, kad *attribute* nėra privalomas, taigi DSA jis taip pat nežymimas kaip privalomas. Jei `use` reikšmė yra "required", reiškia, kad šis *attribute* yra privalomas, ir DSA prie jo pavadinimo pridedama žymė `required`.

**Duomenų tipai**

XSD tipas (type)	DSA tipas (type)
string	string
boolean	boolean
decimal	number
float	number
double	number
duration	string
dateTime	datetime
time	time
date	date
gYearMonth	date;enum;M
gYear	date;enum;Y
gMonthDay	string
gDay	string
gMonth	string
hexBinary	string
base64Binary	binary;prepare;base64
anyURI	uri
QName	string
NOTATION	string
normalizedString	string
token	string
language	string
NMTOKEN	string
NMTOKENS	string
Name	string
NCName	string
ID	string
IDREF	string
IDREFS	string

continues on next page

Table 6 – tęsinys iš praeito puslapio

XSD tipas (type)	DSA tipas (type)
ENTITY	string
ENTITIES	string
integer	integer
nonPositiveInteger	integer
negativeInteger	integer
long	integer
int	integer
short	integer
byte	integer
nonNegativeInteger	integer
unsignedLong	integer
unsignedInt	integer
unsignedShort	integer
unsignedByte	integer
positiveInteger	integer
yearMonthDuration	integer
dayTimeDuration	integer
dateTimeStamp	datetime
	string

## 2.14 Duomenų šaltiniai

### 2.14.1 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

Konkreto 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 veiksams reikalingiems ryšiui su duomenų baze užmegzti ir duomenų bazės paruošimui, kad būtų galima skaityti 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.

### 2.14.2 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*.

**2.14.3 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*.

**2.14.4 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
<b>Country</b>	id	integer	id	<b>countries/country</b> @id
	name@en	string		@name
	cities[]	backref	<b>City</b>	cities/city
	cities[].id	integer		@id
	cities[].name@en	string		@name
	cities[].country	ref	<b>Country</b>	../@id
	cities[].streets[]	backref	<b>Street</b>	streets/street
	cities[].streets[].id	integer		@id
	cities[].streets[].name@en	string		@name
	cities[].streets[].city	ref	<b>City</b>	../@id
<b>City</b>	id	integer	id	<b>countries/country/cities/city</b> @id
	name@en	string		@name

118	country	ref	<b>Country</b>	../@id	<b>Skyrius 2. Turinys</b>
	<b>Street</b>		id	<b>countries/country/cities/city/streets/street</b>	
	id	inte-		@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 žymymos [] 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"
        }
      },
      {
        "_type": "Street",
        "_id": "5b04fecb-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": "5b04fecb-5fff-48f6-8674-7cc6da840281",
      "id": 103,
      "name": {"en": "Daukanto st."},
      "city": {
        "_type": "City",
        "_id": "0fee7d9a-6827-4931-bbea-d44d197faef2"
      },
    },
  ]
}

```

### 2.14.5 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*.

## 2.15 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 turėtų būti perkelti į globalią duomenų erdvę. Į globalią duomenų erdvę pirmiausiai turėtų būti perkelti 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.

### 2.15.1 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	<b>dcat</b>					ns
2				<b>dataset</b>		
3					title	
4	<b>datasets/gov/ivpk/adk</b>					
5		adk				
6			<b>/dcat/dataset</b>			alias
7				<b>dataset</b>		
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.

## 2.16 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	proper- ty	type	ref	uri
						prefix	locn dbpedia- owl	http://www.w3.org/ns/locn# http://dbpedia.org/ontology/
								datasets/example/geo
						sql		salys
				Country			id	locn:Location
				id		inte- ger		dct:identifier
				na- me@lt		text		locn:geographicName
				capital		ref	City	dbpedia-owl:capital
				City			id	locn:Location
				id		inte- ger		dct:identifier
				na- me@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-
    ↪7bble468b165"
    dct:identifier="1">
    <locn:geographicName xml:lang="lt">Lietuva</locn:geographicName>
    <dbpedia-owl:capital
      rdf:resource="datasets/example/geo/City/b54c21f6-08b8-4bdd-b785-
    ↪belcb2e93a98" />
    </locn:Location>

    <locn:Location
      rdf:about="datasets/example/geo/City/b54c21f6-08b8-4bdd-b785-
    ↪belcb2e93a98"
      dct:identifier="1">
      <locn:geographicName xml:lang="lt">Vilnius</locn:geographicName>
      <dbpedia-owl:country
        rdf:resource="datasets/example/geo/Country/eb09946c-26e1-4698-
    ↪a298-7bble468b165" />
      </locn:Location>
    </rdf:RDF>

```

Išoriniai žodynai suteikia galimybę eksportuoti duomenis *RDF* formatu.

Jei struktūros apraše nėra užpildytas *uri* stulpelis, tada, turētu 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/> .

<datasets/example/geo/Country/eb09946c-26e1-4698-a298-7bble468b165>
  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-belcb2e93a98> .

<datasets/example/geo/City/b54c21f6-08b8-4bdd-b785-belcb2e93a98>
  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-a298-7bble468b165> .

```

### 2.16.1 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 įtrauktas 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	integer	id	locn:Location
	uri	uri		locn:Location
	name@lt	text		
City	id	integer	id	locn:Location
	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-belcb2e93a98>
  a locn:Location ;
  <City/id> 1 ;
  <City/name> "Vilnius"@lt ;
  <City/country> <https://sws.geonames.org/597427/> .
```

## 2.17 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*, kuri gali interpretuoti automatizuotos priemonės, priklausomai nuo duomenų šaltinio ir konteksto ir *DSA* sluoksnio.

### 2.17.1 Gramatika

Formulių sintaksė atitinką šią *ABNF* gramatiką:

```
formula      = testlist
testlist     = test *(", " test) *1", "
test         = or
or           = and *(" | " and)
```

(continues on next page)

(tęsinys iš praeito puslapio)

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

## 2.17.2 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, veiksmi.

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

```
{
  "name": "test",
  "args": ["a", "b"],
}
```

### 2.17.3 Funkcijų iškvietimas

Formulės susideda iš vykdomų funkcijų sekos. Pavyzdžiui funkcijos pavadinimu `test` vykdomas formulėje atrodys taip:

```
test()
```

Aukščiau pavyzdyje pateikta formulė vykdo funkciją `test`, be argumentų. Tačiau funkcijos gali turėti pozicinius ir vardinius argumentus.

### 2.17.4 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.

### 2.17.5 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.

### 2.17.6 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).

## 2.17.7 Standartinės funkcijos

Priklausomai nuo duomenų šaltinio ar konteksto gali būti naudojami skirtingi veiksmai, tačiau žemiau yra pateikti bendrosios paskirties veiksmai:

**bind**(*name*)

Rodo į reikšmę pavadinimu *name* iš konteksto. Reikšmės ieškoma tokia tvarka:

- *var()*
- *param()*
- *item()*
- *prop()*

**prop**(*name*)

Modelio savybė pavadinimu *name* iš *property* stulpelio.

**item**(*name*)

Sąrašo elemento savybė pavadinimu *name*.

**param**(*name*)

Parametras pavadinimu *name*. Žiūrėti *param*.

**var**(*name*)

Kintamasis apibrėžtas *set()* funkcijos pagalba.

**self**()

Rodo į aktyvią reikšmę, naudojamas *property.prepare* formulėse.

**or**(*\*args*)

Taip pat galima naudoti tokia išraiška:

a | b | c

Grąžina pirmą netuščią reikšmę. Pirmoji netuščia reikšmė nutraukia sekančių args argumentų interpretavimą.

**and**(*\*args*)

Taip pat galima naudoti tokia išraiška:

a & b & c

Grąžina pirmą tuščią reikšmę arba paskutinę reikšmę, jei prieš tai esančios reikšmės netuščios.

**not**(*arg*)

Taip pat galima naudoti tokia išraiška:

!arg

Jei *arg* tuščia grąžina *true*, priešingu atveju *false*.

**eq**(*a*, *b*)

Taip pat galima naudoti tokia išraiška:

a = b

a lygus b.

**ne**( $a, b$ )

Taip pat galima naudoti tokia išraiška:

 $a \neq b$ 

a nelygus b.

**lt**( $a, b$ )

Taip pat galima naudoti tokia išraiška:

 $a < b$ 

a mažiau už b.

**le**( $a, b$ )

Taip pat galima naudoti tokia išraiška:

 $a \leq b$ 

a mažiau arba lygu už b.

**gt**( $a, b$ )

Taip pat galima naudoti tokia išraiška:

 $a > b$ 

a daugiau už b.

**ge**( $a, b$ )

Taip pat galima naudoti tokia išraiška:

 $a \geq b$ 

a daugiau arba lygu už b.

**add**( $a, b$ )

Taip pat galima naudoti tokia išraiška:

 $a + b$ 

a ir b suma.

**sub**( $a, b$ )

Taip pat galima naudoti tokia išraiška:

 $a - b$ 

a ir b skirtumas.

**mul**( $a, b$ )

Taip pat galima naudoti tokia išraiška:

 $a * b$ 

a ir b sandauga.

### **div**(*a*, *b*)

Taip pat galima naudoti tokia išraiška:

`a / b`

*a* ir *b* dalyba.

### **mod**(*a*, *b*)

Taip pat galima naudoti tokia išraiška:

`a % b`

*a* ir *b* modulis.

### **positive**(*a*)

Taip pat galima naudoti tokia išraiška:

`+a`

Gali būti interpretuojamas skirtingai, priklausomai nuo konteksto. Įprastiniu atveju keičia skaičiaus ženklą.

### **negative**(*a*)

Taip pat galima naudoti tokia išraiška:

`-a`

Gali būti interpretuojamas skirtingai, priklausomai nuo konteksto. Įprastiniu atveju keičia skaičiaus ženklą.

### **tuple**(*\*args*)

Taip pat galima naudoti tokia išraiška:

`(*args)`

Grupė argumentų.

**()**

Tuščia grupė.

**a, b**

Tas pats, kas `tuple(a, b)`.

### **list**(*\*args*)

Taip pat galima naudoti tokia išraiška:

`[*args]`

Sąrašas reikšmių.

### **getattr**(*object*, *attr*)

Taip pat galima naudoti tokia išraiška:

`object.attr`

Gaunamos reikšmės pagal atributą arba raktą.



### **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]
```

### **dict(\*\*kwargs)**

Taip pat galima naudoti tokia išraiška:

```
{a: b}
```

Sudėtinė duomenų struktūra.

### **set(\*\*kwargs)**

Taip pat galima naudoti tokia išraiška:

```
{a, b}
```

Reikšmių aibė.

### **op(operator)**

Taip pat galima naudoti tokia išraiška:

```
a(*)
```

Operatoriai gali būti naudojami kaip argumentai.

### **stack(columns, values, exclude)**

Visus stulpelius išskyrus `exclude` verčia į vieną stulpelių eilutę 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

### **datetime(str, format)**

Išgaunama data ir laikas iš `str`, naudojant `strftime` formatą.

**date**(*str, format*)

išgaunama data iš *str*, naudojant *strftime* formatą.

**date**(*datetime*)

Gražinama data iš datos ir laiko.

### 2.17.8 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**

**file**(*resource, encoding: 'utf-8'*)

#### Parametrai

- **resource** -- Kelias arba URI iki failo.
- **encoding** -- Failo koduotė.

Ši funkcija leidžia nurodyti failo koduotę, jei failas yra užkoduotas kita, nei UTF-8 koduote. Pilną palaikomų koduočių sąrašą galite rasti [šiuose sąraše](#).

**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.

**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**

### 2.17.9 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**

**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.

**head(n)**

Praleisti n einančių po stulpelių pavadinimų eilutės.

**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 įrašytas lentelės pirmajame line argumente.

#### **property.prepare**

Jei naudojamas *header(\*line)*, žiūrėti *Kompleksinės struktūros*.

### 2.17.10 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 turėtų atrodyti taip:

d	r	b	m	property	type	ref	source	prepare
datasets/example/countries								
			Country		sql		sqlite://	
				code			COUNTRIES	<b>code = "lt"</b>
				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 lentelės 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	<b>country.code = "lt"</b>

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**

**eq(a, b)**

Sutrumpinta forma:

**a = b**

Sąlyga tenkinama, jei a ir b reikšmės yra lygios.

**ne(a, b)**

Sutrumpinta forma:

```
a != b
```

Sąlyga tenkinama, jei **a** ir **b** reikšmės nėra lygios.

**gt**(*a*, *b*)

Sutrumpinta forma:

```
a > b
```

Sąlyga tenkinama, jei **a** reikšmė yra didesnė už **b**.

Ši funkcija veikia tik su kiekybiniais duomenimis, kuriuos galima palyginti.

**lt**(*a*, *b*)

Sutrumpinta forma:

```
a < b
```

Sąlyga tenkinama, jei **a** reikšmė yra mažesnė už **b**.

Ši funkcija veikia tik su kiekybiniais duomenimis, kuriuos galima palyginti.

**ge**(*a*, *b*)

Sutrumpinta forma:

```
a >= b
```

Sąlyga tenkinama, jei **a** reikšmė yra didesnė arba lygi **b** reikšmei.

Ši funkcija veikia tik su kiekybiniais duomenimis, kuriuos galima palyginti.

**le**(*a*, *b*)

Sutrumpinta forma:

```
a <= b
```

Sąlyga tenkinama, jei **a** reikšmė yra mažesnė arba lygi **b** reikšmei.

Ši funkcija veikia tik su kiekybiniais duomenimis, kuriuos galima palyginti.

**in**(*a*, *b*)

Sutrumpinta forma:

```
a.in(b)
```

Sąlyga tenkinama, jei **a** lygi bent vienai iš **b** sekos reikšmių.

**notin**(*a*, *b*)

Sutrumpinta forma:

```
a.notin(b)
```

Sąlyga tenkinama, jei **a** nelygi nei vienai iš **b** sekos reikšmių.

**contains**(*a*, *b*)

Sutrumpinta forma:

```
a.contains(b)
```

Sąlyga tenkinama, jei bent viena *a* sekos reikšmių sutampa su *b* reikšme.

**startswith**(*a*, *b*)

Sutrumpinta forma:

```
a.startswith(b)
```

Sąlyga tenkinama, jei *a* simbolių eilutė prasideda *b* simbolių eilute.

**endswith**(*a*, *b*)

Sutrumpinta forma:

```
a.endswith(b)
```

Sąlyga tenkinama, jei *a* simbolių eilutė baigiasi *b* simbolių eilute.

**and**(*a*, *b*)

Sutrumpinta forma:

```
a & b
```

Sąlyga tenkinama, jei abi *a* ir *b* sąlygos tenkinamos.

**or**(*a*, *b*)

Sutrumpinta forma:

```
a | b
```

Sąlyga tenkinama, jei bent viena *a* arba *b* sąlyga tenkinama.

**sort**(*+a*, *-b*)

Rūšiuoti didėjimo tvarka pagal *a* ir mažėjimo tvarka pagal *b*.

**select**(*\*props*)

Pateikiant duomenis, grąžinamos tik nurodytos savybės.

Jei nenurodyta jokia savybė, įtraukia visas, išskyrus daugiareikšmes savybes.

**include**(*\*props*)

Prie grąžinamų savybių, papildomai įtraukiamos nurodytos savybės. Gali būti naudojama kartu su `select()`, papildomai įtraukiant daugiareikšmes ar sudėtinių tipų savybės.

**exclude**(*\*props*)

Pašalina savybę iš `select()` ar `expand()` funkcijos pagalba atrinktų savybių sąrašo.

**expand**(*\*props*)

Veikia panašiai, kaip `include()`, tačiau įtraukia visas nurodyto sudėtinio tipo savybes.

Jei nenurodyta konkreti savybė, įtraukia visų sudėtinių tipų savybes.

`expand()` veikia tik su tiesiogiai modeliui priklausančiomis savybėmis.

**Taip pat žiūrėkite**

- [Savybių įtraukimas](#)

**extends**(*model*, *\*props*)

Įtraukia nurodyto model modelio props savybes.

Jei nenurodytas model ir props, įtraukia visas [base](#) modelio savybes.

Jei nenurodyta props, įtraukia, visas model modelio savybes.

**limit**(*num*)

Riboja gražinamų objektų skaičių iki nurodyti num skaičiaus.

Gali būti naudojamas su [page\(\)](#).

**page**(*\*props*)

Nurodo sekančio next puslapio reikšmę.

### 2.17.11 Periodiškumas

Periodiškumui nurodyti naudojamas model.prepare stulpelis, kuriame galima naudoti tokias formules:

**model.prepare****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 atskirti 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.

**hourly()**

*cron('0m')*

**daily()**

*cron('0m 0d')*

**weekly()**

*cron('0m 0h 0w')*

**monthly()**

*cron('0m 0h 1d')*

**yearly()**

*cron('0m 0h 1d 1M')*

### 2.17.12 Statinės reikšmės

Statinės reikšmės arba konstantos duomenų laukams gali būti nurodomos *property.prepare* stulpelyje naudojant formulės sintaksę. Plačiau apie formules žiūrėti *Formulės* skyrelyje.

### 2.17.13 Transformavimas

*property.prepare* stulpelyje gauta šaltinio reikšmė gali būti pasiekama 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**

**null()**

Grąžina `null` reikšmę, jei toliau einančios transformacijos grąžina `null`.

**replace(*old*, *new*)**

Pakeičia visus `old` į `new` simbolius eilutėje.

**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.

**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.



**split()**

Dalina simbolių eilutę naudojant `s+ reguliariąją išraišką`. Grąžina masyvą.

**strip()**

Pašalina tarpo simbolius iš pradžios ir pabaigos.

**lower()**

Verčia visas raides mažosiomis.

**upper()**

Verčia visas raides didžiosiomis.

**len()**

Grąžina elementų skaičių sekoje.

**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.

**switch(\*cases)****case(cond, value)****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.

**swap(old, new)**

Pakeičia old reikšmę new, jeigu self atitinka old.

**return()**

Nutraukia transformacijų grandinę ir grąžina reikšmę.

**set(name)**

Išsaugo reikšmę į kintamąjį name.

**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 #.

**query()**

Funkcija skirta darbui su URI query, skaido URI query dalį į parametrus arba leidžia pakeisti duomenų šaltinio URI query naujomis reikšmėmis.

**Taip pat žiūrėkite**

*param.query()*

**path()**

Skaido failų sistemos arba URI kelią į tokias savybes:

**parts**

Skaido kelią į dalis ([plačiau](#)).

**drive**

Diskas ([plačiau](#)).

**root**

Šaknis ([plačiau](#)).

**Taip pat žiūrėkite**

*param.path()*

## 2.17.14 Kompleksinės struktūros

Daugelis duomenų šaltinių turi galimybę saugoti kompleksines struktūras. Jei duomenys yra kompleksiniai, tada *property.source* stulpelyje galima nurodyti tik duomenų 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"]
      }
    ]
  }
}
```

(continues on next page)

(tęsinys iš praeito puslapio)

```
}
  }
}
}
```

### property.prepare

**getattr(object, name)**

Grąžina object savybę name.

```
>>> self.result.count
1
```

**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"]
```

**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)`.

## 2.18 Sąvokos

### ADK

Lietuvos atvirų duomenų katalogas, prieinamas adresu [data.gov.lt](https://data.gov.lt).

### ADP

Atvirų duomenų portalas, sudarytas iš *atvirų duomenų katalogo* ir *duomenų saugyklos*.

### ADS

Atvirų duomenų saugykla, skirta pakartotinio panaudojimo duomenų publikavimui, valstybinė atvirų duomenų saugykla pasiekama [get.data.gov.lt](https://get.data.gov.lt) adresu.

### ADSA

*DSA* lentelė, kurioje aprašomi jau atverti ir viešai prieinami duomenys.

### aplinkos kintamasis

Angliškai tai vadinama *environment variables*, tai yra operacinės sistemos aplinkos kintamieji.

Plačiau apie tai skaitykite [Vikipedijoje](#).

### 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.

### 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).

### 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 koku 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 koku tikslumu, nėra galimybės automatinėmis priemonėmis patikimai nuskaityti tokių duomenų.
- 2 Publikuojami duomenys turi aiškią, mašininį būdu nuskaitymą 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 specifikacijoje*. 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 unikalios 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 unikalios identifikatorius 6868eca7-0ae1-4390-83d0-7af642a62863, o šalies tipo objekto „Lietuva“ duomenų lauko „sostinė“ reikšmė yra objekto „Vilnius“ unikalios 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.

#### daugiareikšmis tipas

Duomenų tipas nurodantis, kad viena savybė gali turėti daugiau nei vieną, to pačio tipo reikšmę.

Struktūros aprašuose daugiareikšmės savybės žymimos [/] simboliais, užrašomais savybės pavadinimo pabaigoje.

### Taip pat žiūrėkite

- `type.array`

### duomenų valdymo aktas

2020 m. lapkričio 25 d. Europos Parlamento ir Tarybos reglamento (ES) pasiūlymas [2020/0340](#) dėl Europos duomenų valdymo (Duomenų valdymo aktas).

### duomenų katalogas

Lietuvos duomenų portalo sudedamoji dalis, skirta metaduomenims apie duomenų šaltinius registruoti.

Duomenų katalogas prieinamas adresu [data.gov.lt](https://data.gov.lt).

### 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

### duomenų serializavimo formatai

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:

```
gqR0eXBlp3Byb2plY3SkgmFtZapNYW5pZmVzdGFz
```

Visuose šiuose pavyzdžiuose yra pateikti tie patys duomenys, tačiau naudojami skirtingi duomenų serializavimo formatai, koduotės ir skirtingi žodynai.

## 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.

## EIF

### Europos sąveikumo karkasas

**Rekomendacijų rinkinys** apie tai, kaip užtikrinti didesnę skaitmeninę 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.

## IVPK

Informacinės visuomenės plėtros komitetas.

### Pastaba

IVPK pavadininimas 2024 metais pakeistas į [VSSA](#).

## jungtinis modelis

Jungtinis modelis yra duomenų modelis, kuriame yra apjungtos kelių skirtingų klasių savybės į vieną duomenų modelį.

Toks apjungiamas yra vadinamas duomenų denormalizavimu.

### Taip pat žiūrėkite

- [Jungtinis modelis](#)

## 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 unikalčiai identifikuoti įmonę yra gan sudėtinga.

Tačiau turint autoritetingus kanoninius duomenis:

Įmonės kodas	Įmonės pavadinimas
111111111	UAB "Duomesta"

Užduotis unikalčiai identifikuoti įmonę pasidaro paprastesnė. Todėl kanoniniai duomenys yra labai svarbūs.

**kodinis pavadinimas**

Pavadinimas, kuriam keliama tam tikri apribojimai.

**kompozicinis raktas**

Lokalus objekto identifikatorius susidedantis iš daugiau nei vienos reikšmės.

**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 turėtų 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 servisas](#) kurti.

Plačiau skaitykite [Vikipedijoje](#).



**web servisas**

Web servisas yra interneto paslauga skirta automatizuotiems robotams. Interneto svetainės dažniausiai yra skirtos žmonėms, tačiau web servais 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:

```
container:
  name: value
```

YAML yra sukurtas JSON formato pagrindu, siekiant palengvinti darbą su JSON serializuotais duomenimis žmonėms. Analogiškas pavyzdys JSON formatu atrodo taip:

```
{"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*).

**sudėtinis tipas**

Duomenų tipas apjungiantis kelias savybes į vieną grupę. Sudėtiniam tipui priskirtos savybės gali būti pasiekiamos . (taško) pagalba.

**Taip pat žiūrėkite**

- *type.ref*
- *type.backref*
- *type.object*

**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 šaltinis****pirminis duomenų šaltinis**

Duomenų šaltinis, kuriame duomenys pateikiami arba įrašomi pirmą kartą.

Įprastai, kas būtų užtikrinamas duomenų vientisumas, duomenų objektai yra registruojami vienoje vietoje, tai yra numatyta ir *VIIVĮ* įstatyme, kuriame nurodyta, kad Informacinės sistemos objektai, gali būti registruojami tik vienoje vietoje.

**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 naudojimą,

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 prasmų, priklausomai nuo konteksto. Šioje dokumentacijoje, modelis yra duomenų struktūros aprašo dalis leidžianti aprašyti duomenis pateiktus įvairiais formatais.

Tiksliai 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 veiksmu, 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*).

### 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 užklauskos 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 vieno-dą metaduomenų aprašymo formatą, taip pat vieno langelio principu institucijų ir valstybės valdomų subjektų sudarytiems duomenų rinkiniams ir jų metaduomenims ieškoti, peržiūrėti, parsisiųsti, pareiškėjų užklauskoms 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 apskusti 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 apskusti 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.

#### ŠDSA

DSA lentelė, kurioje aprašoma neatvertų, *pirminio duomenų šaltinio* duomenų struktūra.

#### VSSA

Valstybės skaitmeninių sprendimų agentūra.

#### VIIVĮ

Valstybės informacinių išteklių valdymo įstatymas.

## 2.19 Keitimų istorija

### 2.19.1 1.1.0 (neišleista)

- Pridėtos `select()`, `expand()`, `include()`, `exclude()` ir `extends()` funkcijos.
- Naujas skyrius *Funkciniai modeliai*.

### 2.19.2 1.0.0 (2024-10-22)

### 2.19.3 0.1.0 (2022-03-03)

Pirmoji duomenų struktūros aprašo versija.

### b

base, [23](#)

### c

comment, [39](#)

### d

dataset, [19](#)

### e

enum, [31](#)

### l

lang, [41](#)

### m

migrate, [41](#)

model, [28](#)

### p

param, [35](#)

prefix, [30](#)

property, [29](#)

### r

resource, [22](#)

### s

switch, [39](#)

### t

type, [44](#)





## A

absent (*modulje type*), 44  
 access (*modulje base*), 23  
 access (*modulje comment*), 40  
 access (*modulje dataset*), 20  
 access (*modulje enum*), 33  
 access (*modulje model*), 27  
 access (*modulje property*), 29  
 access (*modulje resource*), 22  
 access (*įtaisytytis kintamasis*), 17, 100  
 add()  
     built-in function, 131  
 ADK, **144**  
 ADP, **144**  
 ADS, **144**  
 ADSA, **144**  
 and()  
     built-in function, 130, 138  
 API, **152**  
 aplinkos kintamasis, **144**  
 array (*modulje type*), 56  
 atvirų duomenų direktyva, **144**  
 aukštos vertės duomenys, **146**

## B

backref (*modulje type*), 55  
 base  
     module, 23  
 base (*įtaisytytis kintamasis*), 14  
 bazė, **150**  
 BDAR, **144**  
 binary (*modulje type*), 45  
 bind()  
     built-in function, 130  
 body() (*modulyje param*), 37  
 boolean (*modulje type*), 44  
 brandos lygis, **144**  
 built-in function  
     add(), 131

and(), 130, 138  
 bind(), 130  
 case(), 141  
 cast(), 140  
 choose(), 141  
 connect(), 116  
 contains(), 137  
 cron(), 139  
 daily(), 140  
 date(), 133, 134  
 datetime(), 133  
 decompress(), 134  
 dict(), 133  
 div(), 131  
 endswith(), 138  
 eq(), 130, 136  
 exclude(), 138  
 expand(), 138  
 extends(), 139  
 extract(), 134  
 file(), 134  
 first(), 143  
 ge(), 131, 137  
 getattr(), 132, 143  
 getitem(), 132, 143  
 gt(), 131, 137  
 head(), 135  
 header(), 135  
 hourly(), 140  
 in(), 137  
 include(), 138  
 item(), 130  
 le(), 131, 137  
 len(), 141  
 limit(), 139  
 list(), 132  
 lower(), 141  
 lt(), 131, 137  
 mod(), 132  
 monthly(), 140

mul(), 131  
 ne(), 131, 136  
 negative(), 132  
 not(), 130  
 notin(), 137  
 null(), 140  
 op(), 133  
 or(), 130, 138  
 page(), 139  
 param(), 130  
 path(), 142  
 positive(), 132  
 prop(), 130  
 query(), 142  
 re(), 140  
 replace(), 140  
 return(), 141  
 select(), 138  
 self(), 130  
 set(), 133, 141  
 sort(), 138  
 split(), 140  
 stack(), 133  
 startswith(), 138  
 strip(), 141  
 sub(), 131  
 swap(), 141  
 switch(), 141  
 tabular(), 116  
 tail(), 135  
 tuple(), 132  
 upper(), 141  
 url(), 141  
 var(), 130  
 weekly(), 140  
 xpath(), 117  
 yearly(), 140

## C

case()  
     built-in function, 141  
 cast()  
     built-in function, 140  
 choose()  
     built-in function, 141  
 comment  
     module, 39  
 connect()  
     built-in function, 116  
 contains()  
     built-in function, 137  
 create() (*modulyje migrate*), 43  
 cron()  
     built-in function, 139

## D

daily()  
     built-in function, 140  
 dataset  
     module, 19  
 dataset (*įtaisyttas kintamas*), 13  
 date (*modulje type*), 47  
 date()  
     built-in function, 133, 134  
 datetime (*modulje type*), 46  
 datetime()  
     built-in function, 133  
 daugiareikšmis tipas, 145  
 DCAT, 149  
 DCAT-AP, 149  
 decompress()  
     built-in function, 134  
 delete() (*modulyje migrate*), 43  
 description (*modulje comment*), 40  
 description (*modulje dataset*), 20  
 description (*modulje enum*), 33  
 description (*modulje lang*), 41  
 description (*modulje migrate*), 43  
 description (*modulje model*), 28  
 description (*modulje prefix*), 30  
 description (*modulje property*), 29  
 description (*modulje resource*), 22  
 description (*įtaisyttas kintamas*), 18  
 dict()  
     built-in function, 133  
 didelės vertės duomenys, 146  
 dimensija, 150  
 distinct() (*modulyje model*), 28  
 distribucija, 150  
 div()  
     built-in function, 131

DOS, 152

DSA, 146

duomenų katalogas, 146

duomenų rinkinys, 150

duomenų serializavimo formatas, 146

duomenų valdymo aktas, 146

duomenų šaltinis, 152

## E

EIF, 147

endswith()  
     built-in function, 138

enum  
     module, 31

enum (*modulje property*), 29

eq()  
     built-in function, 130, 136

ETL, 152

Europos sąveikumo karkasas, **147**

exclude()  
    built-in function, **138**  
expand()  
    built-in function, **138**  
extends()  
    built-in function, **139**  
extract()  
    built-in function, **134**

## F

file (*modulje type*), **53**  
file()  
    built-in function, **134**  
filter() (*modulyje migrate*), **43**  
first()  
    built-in function, **143**

## G

ge()  
    built-in function, **131, 137**  
generic (*modulje type*), **55**  
geometry (*modulje type*), **49**  
getattr()  
    built-in function, **132, 143**  
getitem()  
    built-in function, **132, 143**  
gt()  
    built-in function, **131, 137**

## H

head()  
    built-in function, **135**  
header()  
    built-in function, **135**  
header() (*modulyje param*), **37**  
hourly()  
    built-in function, **140**  
http() (*modulyje resource*), **22**

## I

id (*modulje comment*), **39**  
id (*modulje dataset*), **19**  
id (*modulje migrate*), **42**  
id (*modulje resource*), **20**  
id (*įtaisytytis kintamasis*), **16**  
image (*modulje type*), **54**  
in()  
    built-in function, **137**  
include()  
    built-in function, **138**  
integer (*modulje type*), **44**  
item()  
    built-in function, **130**

iteratorius, **152**  
IVPK, **147**

## J

JSON, **152**  
jungtinis modelis, **147**

## K

kanoniniai duomenys, **147**  
kodinis pavadinimas, **148**  
kompozicinis raktas, **148**

## L

lang  
    module, **41**  
le()  
    built-in function, **131, 137**  
len()  
    built-in function, **141**  
level (*modulje base*), **23**  
level (*modulje comment*), **40**  
level (*modulje dataset*), **20**  
level (*modulje model*), **27**  
level (*modulje property*), **29**  
level (*modulje resource*), **22**  
level (*įtaisytytis kintamasis*), **17**  
limit()  
    built-in function, **139**  
list()  
    built-in function, **132**  
lower()  
    built-in function, **141**  
lt()  
    built-in function, **131, 137**

## M

manifestas, **148**  
metaduomenys, **148**  
migrate  
    module, **41**  
mod()  
    built-in function, **132**  
model  
    module, **24, 28**  
model (*įtaisytytis kintamasis*), **15**  
modelis, **151**  
module  
    base, **23**  
    comment, **39**  
    dataset, **19**  
    enum, **31**  
    lang, **41**  
    migrate, **41**  
    model, **24, 28**

- param, 33, 35
- prefix, 30
- property, 29
- resource, 20, 22
- switch, 39
- type, 44
- money (*modulje type*), 52
- monthly()
  - built-in function, 140
- mul()
  - built-in function, 131
- N**
- ne()
  - built-in function, 131, 136
- negative()
  - built-in function, 132
- normalizavimas, **148**
- not()
  - built-in function, 130
- notin()
  - built-in function, 137
- null()
  - built-in function, 140
- number (*modulje type*), 45
- O**
- object (*modulje type*), 56
- objektas, **151**
- op()
  - built-in function, 133
- open (*įtaisytytis kintamasis*), 100
- or()
  - built-in function, 130, 138
- P**
- page()
  - built-in function, 139
- param
  - module, 33, 35
- param()
  - built-in function, 130
- path()
  - built-in function, 142
- path() (*modulyje param*), 35
- pirminis duomenų šaltinis, **149**
- pirminis šaltinis, **149**
- positive()
  - built-in function, 132
- POSIX, **152**
- prefix
  - module, 30
- prepare (*modulje comment*), 39
- prepare (*modulje dataset*), 20
- prepare (*modulje enum*), 33
- prepare (*modulje migrate*), 43
- prepare (*modulje model*), 26
- prepare (*modulje param*), 34
- prepare (*modulje property*), 29
- prepare (*įtaisytytis kintamasis*), 17
- prieigos taškas, **148**
- private (*įtaisytytis kintamasis*), 100
- prop()
  - built-in function, 130
- property
  - module, 29
- property (*modulje model*), 28
- property (*įtaisytytis kintamasis*), 15
- protected (*įtaisytytis kintamasis*), 100
- public (*įtaisytytis kintamasis*), 100
- Q**
- query()
  - built-in function, 142
- query() (*modulyje param*), 36
- R**
- range() (*modulyje param*), 35
- RDF, **152**
- re()
  - built-in function, 140
- read() (*modulyje param*), 35
- ref (*modulje base*), 23
- ref (*modulje comment*), 39
- ref (*modulje dataset*), 20
- ref (*modulje enum*), 33
- ref (*modulje lang*), 41
- ref (*modulje migrate*), 43
- ref (*modulje model*), 26
- ref (*modulje param*), 34
- ref (*modulje prefix*), 30
- ref (*modulje property*), 29
- ref (*modulje resource*), 21
- ref (*modulje type*), 54
- ref (*įtaisytytis kintamasis*), 17
- reguliarioji išraiška, **152**
- replace()
  - built-in function, 140
- resource
  - module, 20, 22
- resource (*įtaisytytis kintamasis*), 14
- REST API, **148**
- return()
  - built-in function, 141
- RFC
  - RFC 4180, 12
  - RFC 9562, 9, 16

## S

savybė, **151**

select()

built-in function, **138**

self()

built-in function, **130**

set()

built-in function, **133, 141**

sisteminis pavadinimas, **149**

sort()

built-in function, **138**

source (modulje comment), **39**

source (modulje dataset), **20**

source (modulje enum), **33**

source (modulje model), **26**

source (modulje param), **34**

source (modulje property), **29**

source (modulje resource), **21**

source (įtaisyta kintamasis), **17**

split()

built-in function, **140**

stack()

built-in function, **133**

startswith()

built-in function, **138**

string (modulje type), **45**

strip()

built-in function, **141**

sub()

built-in function, **131**

subjektas, **151**

sudėtinis tipas, **149**

swap()

built-in function, **141**

switch

module, **39**

switch()

built-in function, **141**

switch.prepare (modulje switch), **39**

switch.source (modulje switch), **39**

## T

tabular()

built-in function, **116**

tail()

built-in function, **135**

text (modulje type), **46**

TGIĮ, **152**

time (modulje type), **48**

title (modulje comment), **40**

title (modulje dataset), **20**

title (modulje enum), **33**

title (modulje lang), **41**

title (modulje migrate), **43**

title (modulje model), **28**

title (modulje prefix), **30**

title (modulje property), **29**

title (modulje resource), **22**

title (įtaisyta kintamasis), **18**

tuple()

built-in function, **132**

type

module, **44**

type (modulje dataset), **19**

type (modulje model), **25**

type (modulje property), **29**

type (modulje resource), **20**

type (įtaisyta kintamasis), **16**

## U

update() (modulje migrate), **43**

upper()

built-in function, **141**

URI, **152**

uri (modulje comment), **40**

uri (modulje model), **27**

uri (modulje prefix), **30**

uri (modulje property), **29**

uri (modulje type), **56**

uri (įtaisyta kintamasis), **17**

url (modulje type), **56**

url()

built-in function, **141**

## V

var()

built-in function, **130**

viešasis žodynas, **149**

VIIIV, **153**

VSSA, **153**

## W

web servisas, **149**

weekly()

built-in function, **140**

## X

xpath()

built-in function, **117**

## Y

YAML, **149**

yearly()

built-in function, **140**



ŠDSA, **153**

žodynas, **151**