Duomenų struktūros aprašas

Dokumente pateikiama informacija kaip parengti **atvirų duomenų rinkinio struktūros aprašą** (toliau – ADSA). ADSA rengiamas inventorinimo metu įkeliant jį į Lietuvos atvirų duomenų katalogą (ADK). Inventorinimas yra pirmas žingsnis, kurį turi padaryti institucija, siekdama atverti savo sukauptus duomenis.

Institucijos parengtas atvirų duomenų struktūros aprašas turi tiksliai atitikti atvertų duomenų struktūrą ir yra parengiamas pirminio duomenų šaltinio struktūros aprašo (toliau – ŠDSA) pagrindu.

ADSA ir ŠDSA sudaromi tokiais pačiais principais, skiriasi tik aprašomas duomenų turinys. Toliau dokumente, jei kalbama bendrai apie duomenų struktūros aprašą bus naudojamas DSA trumpinys.

Vieninga aprašų struktūros parengimo metodika sudaro galimybę automatizuoti duomenų atvėrimo ir publikavimo procesus bei užtikrinti atveriamų duomenų kokybę ir vientisumą ir viso proceso stebėseną.

Šaltinio duomenų struktūros aprašas (ŠDSA)

Jei įmanoma ŠDSA variantas turi būti generuojamas automatiškai, skaitant šaltinio duomenų struktūrą. Vėliau automatiškai generuotas ŠDSA papildomas rankiniu būdu, suskirstant duomenis į duomenų rinkinius, nurodant kurie duomenų laukai gali būti atverti, pateikiant duomenų laukų aprašymus, atliekant nuasmeninimo transformacijas ir pan.

Jei automatiškai generuoti ŠDSA galimybės nėra, tada ŠDSA rengiamas rankiniu būdu.

Galiausiai publikavimui paruoštas ŠDSA automatiškai transformuojamas į ADSA, pašalinant vidinės struktūros informaciją ir paliekant tik viešinimui skirtą informaciją. ADSA skaidomas į vieno duomenų rinkinio lenteles ir automatiškai įkeliamas į ADK. ADSA turėtų būti publikuojamas ADK dar prieš publikuojant pačius duomenis.

Duomenys publikuojami taip pat automatiškai, naudojant ŠDSA metaduomenis. Duomenų publikavimas gali būti atliekamas kopijuojant duomenis į centrinę duomenų saugyklą (toliau ADS), kopijuojami į įstaigos atvirų duomenų saugyklą arba teikiami tiesiai iš įstaigos duomenų bazių ar paprasčiausiai eksportuojami vienu iš atvirų duomenų formatų.

Po to, kai duomenys publikuojami, turi būti atnaujinamas ir ADSA, pateikiant informaciją apie tai kur ir kaip pasiekti publikuotus duomenis.

Jei ŠDSA apimtis yra didelė, tada ŠDSA paruošimo publikavimui darbus reikėtų atlikti palaipsniui. Pirmiausiai publikavimui reikėtų parengti tuos duomenis, kuriems yra išreikštas didžiausias poreikis.

ŠDSA parengimas publikavimui yra pagrindinė veikla atveriant duomenis, kadangi

didelė dalis duomenų atvėrimo proceso gali būti automatizuojama naudojant ŠDSA pateiktus metaduomenis.

Jei įstaiga jau yra atvėrusi ir publikavusi duomenis, tada ŠDSA rengti nereikia, tačiau reikia parengti ADSA.

Atvertų duomenų struktūros aprašas (ADSA)

Įprastai ADSA turėtu būti generuojamas automatiškai ŠDSA pagrindu. Tačiau jei įstaiga jau yra atvėrusi duomenis ir neturi ŠDSA, tada ADSA jei įmanoma, automatiškai generuojamas atvertų duomenų pagrindu.

ADSA lentelėje gali būti aprašyta daug duomenų rinkinių, tačiau publikuojant duomenų rinkinių metaduomenis į ADK ADSA lentelė turi būti suskaidoma į dalis pagal domenų rinkinius. Vienoje dalyje turi būti tik vieno duomenų rinkinio aprašas.

Nepriklausomai kur ir kaip publikuojami atverti duomenys ADSA dėka visi duomenys galiausiai kopijuojami į centrinę valstybinę duomenų saugyklą iš kurios duomenys teikiami įvairiais formatais, sudarant galimybę duomenis pasiekti per centrinį valstybinį API.

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 ar kitas pasirinktas priemones. Kadangi DSA turi aiškią ir griežtą struktūrą, lentelėje pateiktus duomenis taip pat gali lengvai nuskaityti ir interpretuoti kompiuterinės programos.

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

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

Orientacinis įgyvendinimas

DSA sudarytas remiantis praktine patirtimi, įgyvendinant priemones skirtas automatizuotam duomenų atvėrimui. Minėtos priemonės gali būti naudojamos, kaip orientacinis pavyzdys kuriant automatizuotas priemones. Projekto kodą galima rasti šiuo adresu:

https://gitlab.com/atviriduomenys/spinta/

Šios priemonės bus naudojamas tikrinimui ar DSA atitinka specifikaciją ir ar patys atverti duomenys atitinka DSA.

Priemonės įgyvendintos naudojant Python programavimo kalbą ir priemonių kodas teikiamas pagal atviro kodo MIT licencijos sąlygas. Projekto dokumentaciją galima rasti šiuo

adresu:

https://spinta.readthedocs.io/

Šis dokumentas yra sukurtas projekto metu paruoštos dokumentacijos pagrindu, kurią galima rasti šiuo adresu:

https://atviriduomenys.readthedocs.io/

Šiame dokumentacijoje galite rasti daugiau informacijos ir pavyzdžius apie tai, kaip rengti DSA konkrečiais atvejais.

Lentelės struktūra

Rengiant duomenų struktūros aprašus darbas vyksta su viena lentele. Lentelė sudaryta iš 15 stulpelių. Ką reiškia kiekvienas stulpelis paaiškinta žemiau.

Stulpelis	Pavadinimas	Aprašymas
`id`	Eilutės identifikatorius	Unikalus elemento numeris, gali būti sveikas, monotoniškai didėjantis skaičius arba UUID. Svarbu užtikrinti, kad visi elementai turėtu unikalų `id`.
`dataset`	Duomenų rinkinys	Kodinis duomenų rinkinio pavadinimas. Atitinka dcat:Dataset prasmę. Žiūrėti Duomenų rinkinys.
`resource`	Duomenų šaltinis	Kodinis duomenų šaltinio pavadinimas. Atitinka dcat:Distribution prasmę. Žiūrėti Duomenų šaltinis.
`base`	Modelio bazė	Kodinis bazinio modelio pavadinimas. Atitinka rdfs:subClassOf prasmę (`model rdfs:subClassOf base`). Žiūrėti Modelio bazė.
`model`	Modelis (lentelė)	Kodinis modelio pavadinimas. Atitinka rdfs:Class prasmę. Žiūrėti Duomenų modelis.
`property`	Savybė (stulpelis)	Kodinis savybės pavadinimas. Atitinka rdfs:Property prasmę. Žiūrėti Duomenų laukas.
`source`	Šaltinis	Duomenų šaltinio struktūros elementai. Žiūrėti Duomenų šaltiniai.
`prepare`	Formulė	Formulė skirta duomenų atrankai, nuasmeninimui, transformavimui, tikrinimui ir pan. Žiūrėti Formulės.
`type`	Tipas	Prasmė priklauso nuo sluoksnio. Žiūrėti Duomenų laukų tipai.
`ref`	Ryšys	Prasmė priklauso nuo sluoksnio. Žiūrėti Ryšiai tarp modelių.

`level`	Brandos lygis	Duomenų brandos lygis, atitinka 5 Star Data. Žiūrėti Brandos lygiai.
`access`	Prieiga	Duomenų prieigos lygis. Žiūrėti Prieigos lygiai.
`uri`	Žodyno atitikmuo	Sąsaja su išoriniu žodynu. Žiūrėti Sąsaja su išoriniais žodynais.
`title`	Pavadinimas	Elemento pavadinimas.
`description`	Aprašymas	Elemento aprašymas. Galima naudoti Markdown sintaksę.

Duomenų struktūros aprašo lentelėje laukas `id` turi būti visada užpildytas. `id` reikšmė turi sutapti tiek ŠDSA tiek ADSA.

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

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

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 lentelės stulpelių pavadinimai turi būti užrašyti tiksliai taip, kaip specifikacijoje, kad kompiuterio programos galėtų juos atpažinti.

Visuose sluoksnių stulpeliuose ir kitose vietose kuriuose nurodyta naudoti kodinius pavadinimus keliamas reikalavimas, kad pavadinimai atitiktų šią reguliariąją išraišką:

```
[a-z][a-z0-1_]+
```

Tai reiškia, kad pavadinimai turi būti užrašyti mažosiomis raidėmis ir pirma raidė turi būti skaičius, o sekančios raidės gali būti raidės, skaičiai ir pabraukimo simbolis skirtas žodžiams atskirti, jei pavadinimą sudaro daugiau nei vienas žodis. Visos raidės turi būti mažosios. Kodiniuose pavadinimuose gali būti tik lotyniškos raidės, lietuviškų raidžių kodiniuose pavadinimuose neturi būti.

Ypatingas dėmesys turi būti kreipiamas suteikiant pavadinimus `dataset`, `model` ir `property` stulpeliuose. Šiuose stulpeliuose pateikti pavadinimai naudojami identifikuojant konkrečias duomenų struktūros vietas, taip pat šie pavadinimai bus naudojami publikuojant

duomenis, tai reiškia, kad šiuos pavadinimus naudos ir duomenų naudotojai. Po to, kai duomenys publikuojami minėtų `dataset`, `model` ir `property` pavadinimu reikėtų vengti keisti, kad duomenų naudotojams nereikėtų taisyti jau padarytų integracijų su atvertais duomenimis.

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.

Vardų erdvė	Aprašymas
`/<standard>/`	**Globali vardų erdvė**. Globalią vardų erdvę rekomenduojama formuoti egzistuojančių standartų pagrindu suteikiant vardų erdvei `standard` standarto sutrumpintą pavadinimą. Pavyzdžiui atvirų duomenų katalogo metaduomenys turėtų keliauti į `/dcat/` vardų erdvę. Standartų sutrumpintus pavadinimus rekomenduojame imti iš Linked Open Vocabularies katalogo.
`/datasets/`	Vardų erdvė skirta įstaigų atveriamiems duomenų rinkiniams.
`/datasets/<type>/`	Vardų erdvė skirta konkrečiam organizacijos tipui. Organizacijų tipai gali būti tokie: `gov` – valstybinės įstaigos, `com` – verslo įmonės.
`/datasets/<type>/<org>/`	Konkrečios organizacijos **vietinė vardų erdvė**. Rekomenduojama `org` reikšmei naudoti organizacijos trumpinį, kad bendras modelio pavadinimas nebūtų per daug ilgas.
`/datasets/<type>/<org>/<dataset>/`	Įstaigos **duomenų rinkinio vardų erdvė** į kurią patenka visi įstaigos duomenų modeliai.

Naujai atveriamų duomenų struktūros aprašai sudaromi originalaus duomenų šaltinio struktūros pagrindu. Originalūs duomenų šaltiniai įprastai nėra kuriami vadovaujantis standartais. Vidinės struktūros dažniausiai kuriamos vadovaujantis sistemai keliamais reikalavimais. Todėl naujai atveriamų duomenų rinkiniai turi keliauti į duomenų rinkinio vardų erdvę `/datasets/<type>/<org>/<dataset>/`, išlaikant pirminę duomenų struktūrą ir

neprarandant duomenų.

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

Vardų erdvės pavadinimai gali būti globalūs arba vietiniai. Globalūs vardų erdvės pavadinimai turi prasidėti `/` simboliu, vietiniai vardų erdvės pavadinimai neturi prasidėti `/` simboliu.

Modeliai gali būti aprašomi duomenų rinkinio kontekste arba nepriklausomai nuo duomenų rinkinio. Jei modelis aprašomas duomenų rinkinio kontekste ir modelio pavadinimas neprasideda `/` simboliu, tada pilnas modelio pavadinimas formuojamas jungiant vietinį modelio pavadinimą prie duomenų rinkinio vardų erdvės. Tačiau jei modelio pavadinimas prasideda `/` simboliu, tada pilnas modelio pavadinimas nėra jungiamas prie duomenų rinkinio vardų erdvės.

Kaip tai atrodo DSA lentelėje iliustruota žemiau:

id	d	r	b	m	property
1				**dcat/dataset**	
2					title
3	**datasets/gov/ivpk/adk**				
4		adk			
5			**/dcat/dataset**		
6				**dataset**	
7					title

Šiame pavyzdyje matome, kad pirmoje eilutėje yra apibrėžtas `dcat/dataset` modelis, kuris nėra susietas duomenų rinkiniu, tai reiškia, kad modelis yra globalus. `dcat/dataset` modelio pavadinimas neturi `/` simbolio priekyje, todėl pilnas modelio pavadinimas bus `/dcat/dataset`, nes šis modelis neturi duomenų rinkinio konteksto. Modelio pavadinime `dcat` reiškia standarto arba domeno (srities) pavadinimą.

Toliau lentelėje yra aprašytas duomenų rinkinys `datasets/gov/ivpk/adk`, kur `gov` yra valstybinio sektoriaus duomenų erdvė, `ivpk` yra konkrečios įstaigos sutrumpintas pavadinimas, o `adk` yra atvirų duomenų katalogo duomenų rinkinio sutrumpintas pavadinimas.

Toliau 5 eilutėje nurodyta modelio bazė `/dcat/dataset`. Kadangi modelio bazės pavadinimas prasideda `/` simboliu, tai modelio pavadinimas nesiejamas su duomenų rinkinio vardų erdve ir rodo į pirmoje eilutėje apibrėžtą modelį.

6 eilutėje pateiktas modelio pavadinimas `dataset` neturi priekyje `/`, todėl yra siejamas su duomenų rinkinio vardų erdve. Pilnas 6 eilutėje aprašyto modelio pavadinimas bus `/datasets/gov/ivpk/adk/dataset`.

Visose vietose, kur naudojamas modelio pavadinimas, jei eilutė yra `dataset` sluoksnio sudėtyje, tada modelio pavadinimas bus jungiamas prie duomenų rinkinio vardų erdvės, nebent modelio pavadinimas prasideda `/` simboliu.

Sluoksniai

Stulpeliai `dataset`, `resource`, `base`, `model` ir `property` yra naudojami kaip DSA sluoksniai. `dataset` yra aukščiausias sluoksnis, `property` žemiausias. Sluoksniai apibrėžia duomenų metaduomenų detalumo lygį. `dataset` ir `resource` sluoksniai atitinka DCAT standartą ir užtikrina trečia duomenų brandos lygį, o žemiau esantys `base`, `model` ir `property` atitinka RDFS standartą ir užtikrina penktą duomenų brandos lygį. Vienoje lentelės eilutėje gali būti užpildytas ne daugiau kaip vienas sluoksnio stulpelis. Užpildytasis sluoksnio stulpelis nustato visų kitų stulpelių prasmę.

id	d	r	b	m	property	title
1	datasets/gov/ivpk/adk					Atvirų duomenų katalogas
2		adk				Atvirų duomenų katalogo duomenų bazė
3			/dcat/dataset			Duomenų rinkinys
4				dataset		Duomenų rinkinys
5					title	Duomenų rinkinio pavadinimas

Pavyzdyje aukščiau, taupant vietą, dalies sluoksnių pavadinimai sutrumpinti iki vienos raidės ir įtraukti ne visi stulpeliai, o tik sluoksnių, `id` ir `title` stulpeliai. Pavyzdyje matome, kad vienoje eilutėje užpildytas tik vienas sluoksnio stulpelis, o `title` stulpelio prasmė keičiasi priklausomai nuo sluoksnio reikšmės. Toliau specifikacijoje konkretaus sluoksnio stulpeliai įvardijami pateikiant tiek sluoksnį, tiek stulpelį, kad būtų aiškiau apie kurio sluoksnio stulpelį kalbama, pavyzdžiui `model.title` leidžia suprasti kad kalbama apie „Duomenų rinkinys“ reikšmę 4-oje eilutėje.

Be minėtų sluoksnių stulpelių DSA lentelėje gali būti naudojami papildomi sluoksniai, kai nurodoma `type` reikšmė ir nepateikiama nei viena sluoksnio reikšmė. Pavyzdžiui:

id	d	r	b	m	property	type	ref	uri
1						prefix	dcat	http://www.w3.org/ns/dcat#

Šiuo atveju `prefix` tampa dar vienu sluoksniu ir kalbant apie `dcat` reikšmę nurodomas `prefix.ref` stulpelis.

Eilučių eiliškumas lentelėje yra svarbus, kadangi žemiau esančios eilutės priklauso aukščiau esančiam sluoksniui. Tas pats galioja ir pagalbiniams sluoksniams.

Nors lentelėje yra tik 15 stulpelių, tačiau pasitelkiant 5 pagrindinius sluoksnis ir keletą papildomų sluoksnių atsiranda galimybė išsamiai aprašyti visą duomenų šaltinio struktūrą.

Duomenų rinkinys

DSA lentelėje duomenų rinkinys nurodomas tam, kad būtų išlaikomas ryšys tarp DSA ir ADK. Atliekant duomenų inventorizaciją, automatiškai generuota DSA lentelė turi būti suskirstoma į duomenų rinkinius. Tada automatizuotos priemonės pagalba ADK portale automatiškai sukuriami pirminiai ADK metaduomenys apie duomenų rinkinius, kuriuos vėliau reikia papildyti rankiniu būdu prisijungus prie ADK. Automatizuota priemonė sukūrus duomenų rinkinių įrašus ADK, papildys DSA lentelę, į `dataset.ref` įrašant ADK sukurto duomenų rinkinio identifikatorių. Tokiu būdu, sekantį kartą vykdant sinchronizaciją, jei `dataset.ref` yra užpildytas, bus atnaujinami jau sukurti ADK duomenų rinkinių įrašai.

Į ADK turi būti publikuojami tik tie duomenų rinkiniai iš DSA, kurių `dataset.access` reikšmė yra `public` arba `open`.

Stulpelis		Aprašymas
`dataset`		
	`source`	Jei nenurodyta, naudoti `https://data.gov.lt/` adresą.
	`prepare`	Nenaudojama.
	`type`	Jei nenurodyta, naudoti `ivpk` reikšmę. `type` nurodo API formatą, kuriuo automatiškai pildomi duomenų rinkinių metaduomenys atvirų duomenų portale.
	`ref`	Duomenų rinkinio duomenų kataloge identifikatorius.
	`level`	Viso duomenų rinkinio brandos lygis. Paveldimas.
	`access`	Viso duomenų rinkinio prieigos lygis. Paveldimas.
	`title`	Jei nenurodyta, naudoti „Lietuvos atvirų duomenų portalas“.
	`description`	Jei nenurodyta, palikti tuščią.

Skaidymas į duomenų rinkinius turi būti atliekamas tokiu principu, kad visi tarpusavyje susiję modeliai patektų į vieną duomenų rinkinį. Teoriškai, absoliučiai visi modeliai gali būti susiję tarpusavyje, skaidymą reikėtų daryti pagal tematinį modelių tarpusavio ryšį, o ne pagal reliacinius ryšius.

Jei duomenys yra išskaidyti pagal laiką, vietove ar kitus kriterijus į skirtingus duomenų šaltinius, tokie duomenys turėtų būti apjungti į vieną modelį `base` stulpelio pagalba ir turėtų priklausyti vienam duomenų rinkiniui. Tą pačią semantinę prasmę turintys duomenys

neturėtų būti išskaidyti keliuose duomenų rinkiniuose.

Duomenų šaltinis

Vidinio DSA atveju duomenų šaltinis bus vidinis duomenų bazių serveris, kažkoks vidinis katalogas kuriame yra lentelių failai ar koks nors vidinis API.

Išorinio DSA atveju, duomenų šaltinis gali būti nenurodytas, tai reiškia, kad duomenų rinkinio duomenys dar nėra publikuoti. Jei duomenys jau yra publikuoti, tada išorinio DSA duomenų šaltinis turi rodyti į publikuotus atvertus duomenis, tai gali būti nuorodos į CSV failus, į JSON API ir pan.

Duomenų šaltinio įrašas taip pat naudojamas tam, kad automatiškai atnaujinti ADK esančius duomenų rinkinius, pateikiant konkrečias nuorodas į konkrečius duomenų failus. Analogiškai kaip ir `dataset` atveju, `resource.ref` stulpelyje nurodomas duomenų šaltinio identifikatorius iš ADK.

Stulpelis		**Aprašymas**
`resource`		
	`type`	Duomenų šaltinio tipas.
	`ref`	Duomenų šaltinio duomenų kataloge identifikatorius. Priklauso nuo `dataset.type` reikšmės.
	`level`	Viso duomenų šaltinio brandos lygis. Paveldimas.
	`access`	Viso duomenų šaltinio prieigos lygis. Paveldimas.
	`title`	Duomenų šaltinio pavadinimas.
	`description`	Duomenų šaltinio aprašymas.

Duomenų šaltinio `resource.type` reikšmė apibrėžia kokią ETL priemonę naudoti skaitant duomenis iš duomenų šaltinio. Automatizuota duomenų priemonė skirta įstaigos duomenų atvėrimui turėtų palaikyti tik tokius duomenų šaltinius, kurie naudojami įstaigos vidinėje infrastruktūroje.

Išorinis DSA palaiko tokius duomenų šaltinius:

`resource.type`	**Pavadinimas**
`sql`	Reliacinės duomenų bazės
`csv`	CSV lentelės
`tsv`	TSV lentelės
`json`	JSON resursai

`jsonl`	JSON lines resursai
`xml`	XML resursai
`html`	HTML puslapiai
`xlsx`	Excel lentelės (naujasis OOXML formatas)
`xls`	Excel lentelės (senasis formatas)
`ods`	ODT skaičiuoklės formatas
`wsdl`	WSDL servisas.

Esant poreikiui gali būti įgyvendintas palaikymas naujiems duomenų šaltiniams.

Modelio bazė

Modelio bazė naudojama kelių modelių (lentelių) susiejimui arba apjungimui. Kadangi įvairiuose duomenų šaltiniuose dažnai pasitaiko duomenų, kuriuose saugomos tą pačią semantinę prasmę turinčios lentelės, `base` stulpelyje galima nurodyti kaip skirtingos lentelės siejasi tarpusavyje.

`base.type` stulpelyje nurodoma kokiu būdu lentelės yra susiję. ETL priemonė vadovaujantis `base` informacija duomenis automatiškai transformuoja ir sujungia kelias lenteles į vieną.

Modeliai ne tik susiejami semantiškai tarpusavyje, bet taip pat suliejami ir dviejų modelių duomenys naudojant laukų sąrašą nurodytą `base.ref` stulpelyje. `base.ref` stulpelyje nurodyti laukai naudojami norint unikaliai identifikuoti `model` lentelėje esančią eilutę, kuri atitinka `base` lentelėje esančią eilutę.

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.

`model.property` laukai turi sutapti su `base` modelio laukais, tačiau `model` gali turėti ir papildomų laukų, kurių nėra `base` modelyje. Visi `base.ref` laukai turi būti aprašyti tiek `base`, tiek `model` modeliuose.

Stulpelis		**Aprašymas**
`base`		
	`source`	Nenaudojamas.
	`prepare`	Išimtiniais atvejais, kai nėra galimybės lentelių susieti ar apjungti įprastiniais metodais, galima pasitelkti formules, kurių pagalba galima įgyvendinti nestandartinius lentelių apjungimo atvejus.

	`type`	Lentelių susiejimo tipas. Jei nenurodyta naudoti `alias`.
	`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.
	`level`	Baziniam modeliui priskirtų modelių brandos lygis. Paveldimas.
	`access`	Baziniam modeliui priskirtų modelių prieigos lygis. Paveldimas.
`base.type`		
	`submodel`	Išplečia `base` ir saugo tik tų `property` duomenis, kurių neturi `base`.
	`partition`	Naudojama, kai reikia vieno modelio duomenis išskaidytus pagal datą ar vietą, sujungti į vieną bazę.
	`alias`	Naudojama, kai tą pačią semantinę prasmę duomenys saugomi skirtingose vietose.
	`proxy`	Naudojama tada, kai kelių modelių duomenys yra identiški vienam `base`.

`base.type` savybių matrica:

	Savybės			
`base.type`	**Išplečia**	**Papildo**	**Perduoda**	**Dubliuoja**
`submodel`	**taip**	**ne**	**ne**	**ne**
`partition`	**taip**	**taip**	**ne**	**taip**
`alias`	**taip**	**ne**	**ne**	**taip**
`proxy`	**ne**	**taip**	**taip**	**ne**

Paaiškinimas, ką reiškia kiekviena savybė.

Savybė	**Aprašymas**
Išplečia	`model` gali turėti `property` eilučių, kurių neturi `base`.
Papildo	`model` gali papildyti `base` naujais objektais, jei joks objektas neatitinka `base.ref`.
Perduoda	`model` duomenys perduodami į `base`, pats `model` duomenų nesaugo.
Dubliuoja	`model` saugo kopiją `property` reikšmių, kurios saugomos `base`.

Duomenų modelis

Duomenų modelis apibrėžia duomenų grupę turinčią tas pačias savybes. Skirtinguose duomenų šaltiniuose ir formatuose, duomenų modelis gali būti išreikštas skirtingomis formomis, pavyzdžiui `sql` duomenų šaltinio atveju, modelis aprašo vieną duomenų bazės lentelę.

Kiekvienas modelis turi turėti pirminį raktą, unikalų modelio duomenų identifikatorių. Pirminis raktas aprašomas pateikiant vieną ar kelias `model.property` reikšmes `model.ref` stulpelyje, kurios kartu unikaliai identifikuoja kiekvieną duomenų eilutę.

Išimtiniais atvejais, kai modelio duomenų laukų reikšmės turi būti generuojamos dinamiškai ar kitais nestandartiniais atvejais yra galimybė nurodyti `model.type` reikšmę. Jei `model.type` yra pateiktas, tada už modelio duomenų generavimą, įeinančių duomenų tikrinimą ir visos kitos su modeliu susijusios dalys gali būti pritaikytos konkretaus modelio atvejui. Tačiau, jei reikia keisti tik duomenų pateikimą, užtenka naudoti `model.prepare` formules.

Stulpelis		Aprašymas
`model`		
	`source`	Modelio pavadinimas šaltinyje. Prasmė priklauso nuo `resource.type`.
	`prepare`	Formulė skirta duomenų filtravimui ir paruošimui, iš dalies priklauso nuo `resource.type`.
	`type`	Jei nurodytą, naudoti išplėstą modelio variantą, jei nenurodyta palikti tuščią. Jei tuščia, naudoti standartinį modelio variantą.
	`ref`	Kableliu atskirtas sąrašas `model.property` reikšmių, kurios kartu unikaliai identifikuoja vieną duomenų eilutę (pirminis lentelės raktas).
	`level`	Modeliui priklausančių laukų brandos lygis. Paveldimas.
	`access`	Modeliui priklausančių laukų prieigos lygis. Paveldimas.
	`uri`	Sąsaja su išoriniu žodynu.
	`title`	Modelio pavadinimas.
	`description`	Modelio aprašymas.

Duomenų laukas

Stulpelis	Aprašymas
`property`	

	`source`	Duomenų lauko pavadinimas šaltinyje. Prasmė priklauso nuo `resource`.`type`.
	`prepare`	Formulė skirta duomenų tikrinimui ir transformavimui arba statinės reikšmės pateikimui.
	`type`	Duomenų tipas. Žiūrėti skyrių Duomenų laukų tipai.
	`ref`	Šį lauką reikia pildyti `ref`, `backref` ir `generic` `property`.`type` atvejais. Šiame stulpelyje reikia nurodyti `model` pavadinimą, kas nurodo, kad lauko reikšmė rodo į kitą modelį. Žiūrėti Ryšiai tarp modelių.
	`level`	Nurodo duomenų lauko brandos lygį. Žiūrėti Brandos lygiai.
	`access`	Nurodo prieigos prie duomenų lygį. Žiūrėti skyrių Prieigos lygiai.
	`uri`	Sąsaja su išoriniu žodynu.
	`title`	Duomenų lauko pavadinimas.
	`description`	Duomenų lauko aprašymas.

Duomenų laukų tipai

Duomenų laukų tipai nurodomi `property`.`type` stulpelyje.

Tipas	Pavadinimas	Aprašymas
`boolean`	Loginė reikšmė	
`integer`	Sveikas skaičius	
`number`	Realusis skaičius	
`string`	Simbolių eilutė	
`text`	Tekstas	Žmonių kalba užrašytas tekstas, nurodant kalbą naudojant ISO 639-1 kodus. Tekstas gali būti pateiktas keliomis kalbomis.
`binary`	Dvejetainiai duomenys	
`date`	Data	Data formatas turi atitikti ISO 8601.
`datetime`	Data ir laikas	Datos ir laiko formatas turi atitikti ISO 8601.
`geometry`	Erdviniai duomenys	Duomenys pateikiami WKT, WKB arba suderinamu formatu, kartu nurodant ir SRID.
`currency`	Valiuta	Saugomas valiutos kiekis, nurodant tiek sumą, tiek valiutos kodą naudojant ISO 4217 kodus.
`file`	Failas	`file` tipas yra sudėtinis ir turi tokius metaduomenis:

		`id` – failo UUID. `file_name` – failo pavadinimas. `cotent_type` – failo media tipas. `size` – failo turinio dydis baitais. `content` – failo turinys.
`image`	Paveiksliukas	`image` tipas turi tokias pačias savybes kaip `file` tipas.
`ref`	Ryšys su modeliu	Šis tipas naudojamas norint pažymėti, kad lauko reikšmė yra `ref` stulpelyje nurodyto modelio `id`.
`backref`	Atgalinis ryšys su modeliu	Šis tipas naudojamas norint pažymėti, kad tam tikras kitas modelis turi `ref` tipo lauką, kuris rodo į šį modelį. Šis laukas pats duomenų neturi, tai tik papildomas metaduomuo, padedantis geriau suprasti ryšius tarp modelių.
`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.
`object`	Kompozicinis tipas	Šis tipas naudojamas apibrėžti sudėtiniams duomenims, kurie aprašyti naudojant kelis skirtingus tipas. Kompozicinio tipo atveju `property` stulpelyje komponuojami pavadinimai atskiriami taško simboliu. Sudarant duomenų modelį, rekomenduojama laikytis plokščios struktūros ir komponavimą įgyvendinti siejant modelius per `ref`, `backref` ar `generic` tipus.
`array`	Masyvas	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. Rekomenduojama vengti naudoti šį tipą, siekiant išlaikyti plokščią duomenų modelį. Vietoje `array` tipo rekomenduojama naudoti `backref`.

Ryšiai tarp modelių

Ryšiai tarp modelių nurodomi `ref`, `backref` ir `generic` `property.type` pagalba. Pats ryšys pateikiamas `property.ref` stulpelyje. `property.ref` stulpelyje ryšį su modeliu galima nurodyti tokiais būdais:

`property.ref`	**Aprašymas**

`rmodel`	`rmodel` nurodo kito `model` pavadinimas kurio `rmodel.ref` siejamas su `property`. Jei `rmodel.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.
`rmodel[rproperty]`	Tais atvejais, kai `property` duomenys nesutampa su siejamo `rmodel.ref`, galima nurodyti `rproperty` iš `rmodel`.
`rmodel[*rproperty]`	Jei susiejimui reikia daugiau nei vieno duomenų lauko ir jie nesutampa su `rmodel.ref`, tada galima nurodyti kelias `rproperty` 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.

Brandos lygiai

Duomenų brandos lygis nurodomas `level` stulpelyje.

Duomenų brandos lygis atitinka 5 ★ Open Data skalę, tačiau adaptuota duomenų struktūros aprašo kontekstui. Papildomai įtrauktas nulinis lygis, kai duomenys nekaupiami, tačiau yra reikalingi ir yra parengtas jų duomenų struktūros aprašas.

Lygis	Pavadinimas	Aprašymas
`0`	Nekaupiama	Duomenys nekaupiami. Duomenų rinkinys užregistruotas atvirų duomenų portale. Užpildyta `dataset` eilutė.
`1`	Publikuojama	Duomenys publikuojami bet kokia forma. Užpildyta `resource` eilutė.
`2`	Struktūruota	Duomenys kaupiami struktūruota, mašininiu būdu nuskaitoma forma, bet kokiu formatu. Užpildytas `source` stulpelis.
`3`	Standartizuota	Duomenys saugomi atviru, standartiniu formatu. Užpildytas `property.type` stulpelis ir duomenys atitinka nurodytą tipą.
`4`	Identifikuojama	Duomenų objektai turi aiškius, unikalius identifikatorius. Užpildytas `ref` stulpelis.
`4.5`	Susieta	Duomenys susieti su vieningu valstybiniu žodynu. Užpildyta `base` eilutė ir `base` modelis yra iš globalios vardų erdvės.

5	Susieta su išoriniu žodynu	Duomenys susieti su išoriniais žodynais. Užpildytas uri stulpelis.

Daugeliu atveju brandos lygis gali būti nustatomas automatiškai pagal tai ar yra užpildyti tam tikri stulpeliai. Automatiškai brandos lygio negalima nustatyti tarp 2 ir 3 brandos lygio, todėl automatinės priemonės visada turėtų parinkti žemesnį 2 brandos lygį, jei nenurodyta kitaip.

Brandos lygis gali būti paveldimas iš aukštesnio sluoksnio. Jei žemesniame sluoksnyje nėra nurodytas joks brandos lygis, jis yra paveldimas iš aukštesnio sluoksnio.

Prieigos lygiai

Duomenų prieigos lygis nurodomas access stulpelyje.

Lygis	Pavadinimas	Aprašymas
private	Vidiniam naudojimui	Duomenys skirti tik vidiniam konkrečios sistemos naudojimui.
protected	Pakartotiniam naudojimui	Duomenys gali būti naudojami integracijai su išorinėmis sistemomis.
public	Viešam naudojimui	Duomenys skirti viešam naudojimui, tačiau duomenų panaudojimo tikslai ribojami.
open	Atviram naudojimui	Duomenys skirti viešam naudojimui, neribojant panaudojimo tikslo.

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

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

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

Prieigos lygiai gali būti paveldimi iš aukštesnio sluoksnio. Tačiau žemesnis sluoksnis apsprendžia realų prieigos lygį. Pavyzdžiui jei dataset.access yra private, o to dataset sluoksnyje esantis property yra open, tada visi property aukštesni sluoksniai taip pat tampa open, nors visi kiti sluoksniai yra private, nes paveldi datases.access reikšmę.

Nuasmeninimas

Duomenų laukų reikšmių nuasmeninimas atliekamas `property.prepare` stulpelio pagalba.

`property.prepare`	**Aprašymas**
`randomize(n)`	Reikšmės keičiamos parenkant atsitiktinę vertę ±`n` intervale nuo tikrosios vertės.
`permutate()`	Atsitiktine tvarka sumaišomos duomenų reikšmės.
`hash(name)`	Taikyti numatytą maišos funkciją.
`hash(name)`	Taikyti konkrečią `name` maišos funkciją.
`sample(n)`	Atsitiktine tvarka atrenkama `n` procentų žodžių naudojamų tekste.
`group(n)`	Pakeiti originalias reikšmes į intervalų grupes taip, kad į vieną intervalą patektų ne mažiau nei `n` reikšmių. Jei viena konkreti reikšmė pasikartoja daugiau nei `n` kartų, tada intervalas nekuriamas, reikšmė paliekama tokia kokia yra šaltinyje.

Duomenų paruošimas

Duomenų bazės

Duomenų bazės URI formuojamas naudojant tokį ABNF šabloną:

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

Šablone naudojamų kintamųjų aprašymas:

Parametras	**Aprašymas**
`type`	Duomenų bazių serverio pavadinimas: • `sqlite` • `postgresql` • `mysql` • `oracle` • `mssql`
`driver`	Konkretaus duomenų bazių serverio tvarkyklė naudojama komunikacijai su duomenų baze.

`user`	Naudotojo vardas jungimuisi prie duomenų bazės.
`password`	Duomenų bazės naudotojo slaptažodis.
`host`	Duomenų bazių serverio adresas.
`port`	Duomenų bazių serverio prievadas.
`database`	Konkrečios duomenų bazės pavadinimas.
`params`	Papildomi parametrai Query string formatu.

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.

Stulpelis	**Aprašymas**
`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`	
`extract(type, path)`	Jei `resource.source` rodo į archyvo failą, tada papildomai galima nurodyti koks archyvo tipas ir koks failo kelias archyvo viduje. `type` reikšmės gali būti tokios: • `zip` • `tar` • `rar`
`decompress(type)`	Taikomas srautinis failo glaudinimo filtras. Galimos `type` reikšmės: • `gz` • `bz2` • `xz`

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.

Stulpelis		Aprašymas
`model`.`prepare`		
	`header(null)`	Lentelėje eilučių pavadinimų nėra. Tokiu atveju, `property`.`source` stulpelyje reikia pateikti stulpelio numerį, pradedant skaičiuoti nuo `0`.
	`header(line)`	Nurodomas eilutės numeris, pradedant eilutes skaičiuoti nuo `0`, kur yra pateikti lentelės stulpelių pavadinimai. Pagal nutylėjimą stulpelių pavadinimų ieškoma pirmoje eilutėje.
	`header(*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 įrašymas lentelės pirmajame `line` argumente.
	`prepare`	Jei naudojamas `header(*line)`, žiūrėti Kompleksinės struktūros.

Kompleksinės struktūros

Daugelis duomenų šaltiniu turi galimybę saugoti kompleksines struktūros. Jei duomenys yra kompleksiniai, tada `property`.`source` stulpelyje galima nurodyti tik duomens pavadinimą iš pirmojo lygmens, gilesniuose lygmenyse esančius duomenis galima aprašyti naudojant formules `property`.`prepare` stulpelyje.

Analogiškai duomenų atranką galima daryti ir `model` eilutėse, jei tai leidžia duomenų šaltinis.

Kaip pavyzdį naudosime tokią JSON duomenų struktūrą:

```
{
    "result": {
```

```
            "count": 1,
            "results": [
                {
                    "type": "dataset",
                    "tags": ["CSV"]
                }
            ]
        }
    }
```

property.prepare	Rezultatas
self.result.count	1
self["result"]["count"]	1
self.result.results[0].type	"dataset"
self.result.results[tags = "CSV"].type	["dataset"]
self.result.results[item(tags) = "CSV"].type	["dataset"]
self.result.results[].tags[]	["CSV"]

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)`.

Duomenų atranka

Duomenų filtravimui naudojamas `model.prepare` stulpelis, kuriame galima naudoti tokius filtrus:

model.prepare	Aprašymas
a = b	a ir b reikšmės yra lygios.
a != b	a nelygu b.
a > b	a daugiau už b.

`a < b`	`a` mažiau už `b`.
`a >= b`	`a` daugiau arba lygu `b`.
`a <= b`	`a` mažiau arba lygu `b`.
`a.in(b)`	`a` lygi bent vienai iš `b` sekos reikšmių.
`a.notin(b)`	`a` nelygi nei vienai iš `b` sekos reikšmių.
`a.contains(b)`	`a` seka savyje turi `b` seką.
`a.startswith(b)`	`a` seka prasideda `b` seka.
`a.endswith(b)`	`a` seka baigiasi `b` seka.
`a & b`	`a` ir `b`.
`a | b`	`a` arba `b`.
`sort(+a, -b)`	Rūšiuoti didėjimo tvarka pagal `a` ir mažėjimo tvarka pagal `b`.

Periodiškumas

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

`model.prepare`	**Aprašymas**
`cron(line)`	Duomenų atnaujinimo laikas, analogiškas cron formatui.
`hourly()`	`cron('0m')`
`daily()`	`cron('0m 0d')`
`weekly()`	`cron('0m 0h 0w')`
`monthly()`	`cron('0m 0h 1d')`
`yearly()`	`cron('0m 0h 1d 1M')`

`cron(line)` `line` argumentas aprašomas taip:

`line`	**Aprašymas**
`nm`	`n`-toji minutė, `n` ∈ 0-59.
`nh`	`n`-toji valanda, `n` ∈ 0-23.
`nd`	`n`-toji mėnesio diena, `n` ∈ 1-31.

`$d`	Paskutinė mėnesio diena.
`nM`	`n`-tasis mėnuo, `n` ∈ 1-12.
`nw`	`n`-toji savaitės diena, `n` ∈ 0-6 (sekmadienis-šeštadienis).
`n#iw`	`n`-toji savaitės diena, `i`-toji mėnesio savaitė, `i` ∈ 1-6.
`n$iw`	`n`-toji savaitės diena, `i`-toji savaitė nuo mėnesio galo, `i` ∈ 1-6.
`,`	Kableliu galim atskirt kelias laiko vertes.
`-`	Brūkšneliu galima atskirti laiko verčių intervalą.
`/`	Pasvyruoju brūkšniu galima atskirti laiko verčių kartojimo žingsnį.

Laiko vertės atskiriamos tarpo simbolių. Jei laiko vertė nenurodyta, reiškia įeina visos įmanomos laiko vertės reikšmės.

Statinės reikšmės

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

Fiksuotų reikšmių variantai

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 `choice`.

`property.choice`		**Aprašymas**
	`source`	Pateikiama originali reikšmė, taip kaip ji saugoma duomenų šaltinyje.
	`prepare`	Pateikiama reikšmė, tokia kuri bus naudojama atveriant duomenis. `model.prepare` filtruose taip pat bus naudojama būtent ši reikšmė.
	`type`	`choice`
	`title`	Fiksuotos reikšmės pavadinimas.
	`description`	Fiksuotos reikšmės aprašymas.

Pagal nutylėjimą, jei `property.prepare` yra tuščias ir `property` turi `choice`

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)`.

Dinaminių reikšmių variantai

Tam tikrais atvejais duomenis tenka normalizuoti parenkant tam tikrą reikšmę jei tenkinama nurodyta sąlyga. Tokias situacijas galima aprašyti pasitelkiant `switch` sluoksnį, kuris patenka į `property` kontekstą.

`property.switch`		**Aprašymas**
	`type`	`switch`
	`source`	Reikšmė, kuri bus atveriama.
	`prepare`	Sąlyga, naudojant einamojo modelio laukus. Jei sąlyga tenkinama, tada laukui priskiriama `swtich.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.
	`type`	Nurodoma `switch` reikšmė.

Transformavimas

`property.prepare` stulpelyje gauta šaltinio reikšmė gali būti pasiekiama per `self` kintamąjį.

`property.prepare` formulėje gali būti aprašomos kelios reikšmės atskirtos kableliu, tai naudojama ryšio laukams, kai ryšiui aprašyti reikia daugiau nei vieno duomenų lauko.

Formulėje galima naudoti kitus to pačio modelio `property` pavadinimus, kai aprašomo `property` reikšmės formuojamos dinamiškai naudojant viena ar kelis jau aprašytus laukus.

`property.prepare` stulpelyje galima naudoti tokias formules:

`property.prepare`	**Tipas**	**Aprašymas**
`null()`	`null`	Grąžina `null` reikšmę, jei toliau einančios transformacijos grąžina `null`.
`replace(old, new)`	`string`	Pakeičia visus `old` į `new` simbolių eilutėje.
`re(pattern)`	`string`	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)`	`any`	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()`	`string`	Dalina simbolių eilutę naudojant `\s+` reguliariąją išraišką. Grąžina masyvą.
`strip()`	`string`	Pašalina tarpo simbolius iš pradžios ir pabaigos.
`lower()`	`string`	Verčia visas raides mažosiomis.
`upper()`	`string`	Verčia visas raides didžiosiomis.
`len()`	`string`	Grąžina elementų skaičių sekoje.
`choose()`	`any`	Grąžina vieną iš `property.choice` reikšmių, jei duomenų šaltinio reikšmė nėra viena iš `property.choice`, tada grąžinama klaida.
`choose(default)`	`any`	Jei šaltinio reikšmė nėra viena iš `property.choice`, tada grąžinama `default` reikšmė.
`switch()`	`any`	Grąžina vieną iš `switch.source` reikšmių, tenkinančių `switch.prepare` sąlygą.

`switch(`
`  case(cond, value),`
`  case(default),`
`)`
	`any`	Grąžina `value`, jei tenkina `cond` arba `default`. Jei `case(default)` nepateiktas, tada grąžina pradinę reikšmę.

`return()`	`any`	Nutraukia transformacijų grandinę ir grąžina reikšmę.
`set(name)`	`any`	Išsaugo reikšmę į kintamąjį `name`.
`uri()`	`string`	Skaido URI į objektas 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` – URI dalis einanti tarp ? Ir #.

		`fragment` – URI dalis einanti po #.
`query()`	`string`	Skaido URI `query` dalį į parametrus.
`path()`	`string`	Skaido failų sistemos arba URI kelią į tokias savybes: `parts` – skaido kelią į dalis (plačiau). `drive` – diskas (plačiau). `root` – šaknis (plačiau).

Duomenų šaltiniai

SQL

Stulpelis		Aprašymas
`resource`		
	`source`	Duomenų bazės URI, žiūrėti Duomenų bazės.
	`prepare`	Formulė skirta papildomiems veiksmams reikalingiems ryšiui su duomenų baze užmezgimui ir duomenų bazės paruošimui, kad būtų galima skaityt duomenis.
	`type`	SQL duomenų bazių atveju nurodomas `sql` reikšmė.
`resource.prepare`		
	`schema(name)`	Naudojama tais atvejais, kai reikia aktyvuoti tam tikrą duomenų bazės schemą.
`model`		
	`source`	Duomenų bazėje esančios lentelės pavadinimas.
`property`		
	`source`	Lentelės stulpelio pavadinimas.

CSV

Stulpelis	Aprašymas
`resource`	

	`type`	`csv`, `tsv`.
	`source`	Žiūrėti Failai.
`resource.prepare`		
	`sep(separator)`	Nurodoma kaip CSV faile atskirti stulpeliai. Pagal nutylėjimą `separator` reikšmė yra `","`.
`model`		
	`source`	Nenaudojama.
	`prepare`	Žiūrėti Stulpeliai lentelėje.
`preperty`		
	`source`	Žiūrėti Stulpeliai lentelėje.

JSON

Stulpelis	**Aprašymas**
`resource`	
`type`	`json`, `jsonl`.
`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` sluoksnyje. Jei JSON objektas yra kompleksinis žiūrėti Kompleksinės struktūros.
`preperty`	
`source`	JSON objekto savybė, kurioje pateikiami aprašomo stulpelio duomenys.
`prepare`	Žiūrėti Kompleksinės struktūros.

XML

Stulpelis	**Aprašymas**
`resource`	

	`type`	`xml`, `html`.
	`source`	Žiūrėti Failai.
`model`		
	`source`	XPath iki elementų sąrašo kuriame yra modelio duomenys.
`preperty`		
	`source`	XPath iki elemento kuriame yra duomenys.

Skaičiuoklių lentelės

Stulpelis		**Aprašymas**
`resource`		
	`type`	`xlsx`, `xls` arba `odt`.
	`source`	Žiūrėti Failai.
`model`		
	`source`	Skaičiuoklės faile esančio lapo pavadinimas.
	`prepare`	Žiūrėti Stulpeliai lentelėje.
`preperty`		
	`source`	Žiūrėti Stulpeliai lentelėje.

WSDL

Stulpelis		**Aprašymas**
`resource`		
	`type`	`wsdl`.
	`source`	WSDL URI.
`model`		
	`source`	Nenaudojamas.
	`prepare`	WSDL funkcijos iškvietimas.
`model.prepare`		

	`service(`
	`  name,`
	`  *args,`
	`  **kwargs,`
	`)`
		WSDL funkcijos `name` iškvietimas.

	`wsdl(`
	`  type,`
	`  **kwargs,`
	`)`
		Inicializuoja nurodytą `type` WSDL tipą.

`preperty`		
	`source`	Rezultato objekto atributas.
	`prepare`	Žiūrėti Kompleksinės struktūros.

Parametrizacija

Duomenys gali būti publikuojami fragmentuotai, kai vieno modelio duomenys skaidomi į daug šaltinių, pavyzdžiui CSV failai gali būti suskaidyti pagal metus arba API atveju, vienas API prieigos taškas gali grąžinti eilučių identifikatorius, o kitas prieigos taškas pačius eilutės duomenis. Tokiais atvejais reikalinga parametrizacija, kuri leidžia naudoti dinaminius parametrus `source` ir `prepare` laukuose.

Parametrai aprašomi pasitelkiant papildomą `param` sluoksnį:

`param`	**Aprašymas**
`ref`	Parametro pavadinimas.
`prepare`	Formulė, kuri grąžina sąrašą reikšmių aprašomam parametrui. Jei užpildytas `param`.`source` stulpelis, tada `param`.`prepare` naudojamas nurodyto modelio duomenų filtravimui.
`source`	Parametro reikšmės imamos iš nurodyto modelio duomenų.

Kai konkretus hierarchijos lygmuo yra parametrizuotas, konkretus hierarchijos lygmuo dinamiškai kartojamas tiek kartu kiek parametras turi reikšmių, o `source` ir `prepare` stulpeliuose yra prieinama parametro reikšmė nurodytu `param`.`ref` pavadinimu.

Pavyzdžiui, jei `param`.`ref` stulpelyje būtų nurodytas pavadinimas `x`, tada šio parametro reikšmės gali būti pasiekiamos taip:

Stulpelis	**Reikšmė**
`source`	`{x}`
`prepare`	`x` arba `param(x)`

Parametrų generavimui galima naudoti tokias formules:

param.prepare	Aprašymas
range(stop)	Sveikų skaičių generavimas nuo 0 iki stop.
range(start, stop)	Sveikų skaičių generavimas nuo start iki stop.
scalar(name)	Jei nurodytas param.source, tada imama tik name lauko reikšmė, o ne visi modelio laukai.

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:

Stulpelis	Reikšmė
source	{x.field}
prepare	x.field arba param(x).field

Sąsaja su išoriniais žodynais

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		Aprašymas
	type	prefix
	ref	Prefikso pavadinimas.
	source	Žodyno URI prefiksas.

Rekomenduojama naudoti LOV prefiksus.

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

Formulės

Formulės pildomas 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 mechanizmo, siekiant suvaldyti visą šią įvairovę DSA formulės leidžia vieningai aprašyti veiksmus su duomenimis. Vėliau formulės verčiamos į vieningą AST, kurį gali interpretuoti automatizuotos priemonės, priklausomai nuo duomenų šaltinio ir konteksto ir DSA sluoksnio.

Formulių sintaksė atitinką šią ABNF gramatiką:

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

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:

Savybė	**Aprašymas**
`name`	Funkcijos pavadinimas.
`args`	Funkcijos argumentų sąrašas, kurį gali sudaryti konkrečios reikšmės ar kiti medžio elementai, veiksmai.

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

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

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

Išraiška	**Abstraktus medis**	**Aprašymas**
`a`	`bind('a')`	Rodo į reikšmę iš konteksto: • parametras, kai naudojama parametrizacija, • modelio savybė, • sąrašo objekto savybė.
`prop(a)`	`prop(bind('a'))`	Naudojamas tais atvejais, kai `a` persidengia su rezervuotu pavadinimu arba parametru.
`param(a)`	`param(bind('a'))`	Naudojamas tais atvejais, kai parametras persidengia su rezervuotu pavadinimu.
`var(a)`	`var(bind('a'))`	Naudojama tais atvejais, kai norima pasiekti kintamąjį sukurtą `set()` funkcijos pagalba ir a pavadinimas persidengia su `param` ar `prop` pavadinimu.
`item(a)`	`item(bind('a'))`	Naudojamas tais atvejais, kai norima pasiekti sąraše esančių objektų atributą ar raktą.
`self`	`bind('self')`	Rodo į aktyvią reikšmę.

`a | b | c`	`or(`
	`  bind('a'),`
	`  bind('b'),`
	`  bind('c'),`
	`)`
		Grąžiną pirmą netuščią reikšmę. Pirmoji netuščia reikšmė nutraukia argumentų interpretavimą.

`a & b & c`	`and(`
	`  bind('a'),`
	`  bind('b'),`
	`  bind('c'),`
	`)`
		Grąžina pirmą tuščią reikšmę arba paskutinę reikšmę, jei prieš tai esančios reikšmės netuščios.

`!a`	`not(bind('a'))`	Jei `a` tuščia grąžina `true`, priešingu atveju `false`.

`a = b`	`eq(`
	`  bind('a'),`
	`  bind('b'),`
	`)`
		`a` lygus `b`.

`a != b`	`ne(` `  bind('a'),`	`a` nelygus `b`.

	`    bind('b'),` `)`	

`a < b`	`lt(`
	`    bind('a'),`
	`    bind('b'),`
	`)`
		`a` mažiau už `b`.

`a <= b`	`le(`
	`    bind('a'),`
	`    bind('b'),`
	`)`
		`a` mažiau arba lygu už `b`.

`a > b`	`gt(`
	`    bind('a'),`
	`    bind('b'),`
	`)`
		`a` daugiau už `b`.

`a >= b`	`ge(`
	`    bind('a'),`
	`    bind('b'),`
	`)`
		`a` daugiau arba lygu už `b`.

`a + b`	`add(`
	`    bind('a'),`
	`    bind('b'),`
	`)`
		`a` ir `b` suma.

`a - b`	`sub(`
	`    bind('a'),`
	`    bind('b'),`
	`)`
		`a` ir `b` skirtumas.

`a * b`	`mul(`
	`    bind('a'),`
	`    bind('b'),`
	`)`
		`a` ir `b` sandauga.

`a / b`	`div(`
	`    bind('a'),`
	`    bind('b'),`
	`)`
		`a` ir `b` dalyba.

`a % b`	`mod(`
	`    bind('a'),`
	`    bind('b'),`
	`)`
		`a` ir `b` modulis.

`+a`	`positive(`
	`    bind('a'),`
	`)`
		Gali būti interpretuojamas skirtingai, priklausomai nuo konteksto. Įprastiniu atveju keičia skaičiaus ženklą.

`-a`	`negative(`
	`    bind('a'),`
	`)`
		Gali būti interpretuojamas skirtingai, priklausomai nuo konteksto. Įprastiniu atveju keičia skaičiaus ženklą.

`()`	`tuple()`	Tuščia grupė.

`(a, b)`	`tuple(`
	`    bind('a'),`
	`    bind('b'),`
		Grupė reikšmių arba veiksmų.

	`)`	

`a, b`	`tuple(`
	`  bind('a'),`
	`  bind('b'),`
	`)`
		Grupė reikšmių arba veiksmų.

`[a, b]`	`list(`
	`  bind('a'),`
	`  bind('b'),`
	`)`
		Sąrašas reikšmių.

`a(b, c)`	`a(`
	`  bind('b'),`
	`  bind('c'),`
	`)`
		Vykdomas veiksmas `a` su argumentais `b` ir `c`.

`b.a(c)`	`a(`
	`  bind('b'),`
	`  bind('c'),`
	`)`
		Vykdomas veiksmas `a` su argumentais `b` ir `c`.

`a.b.c`	`getattr(`
	`  getattr(`
	`  bind('a'),`
	`  bind('b'),`
	`  ),`
	`  bind('c'),`
	`)`
		Gaunamos reikšmės pagal atributą arba raktą.

`a["b"]["c"]`	`getitem(`
	`  getitem(`
	`  bind('a'),`
	`  'b',`
	`  ),`
	`  'c',`
	`)`
		Gaunamos reikšmės pagal atributą arba raktą.

`a[b > c]`	`getitem(`
	`  bind('a'),`
	`  gt(`
	`  bind('b'),`
	`  bind('c'),`
	`  )`
	`)`
		`getitem` gali būti interpretuojamas kaip sąrašo reikšmių filtras.

`a(b: c)`	`a(`
	`  bind(`
	`  'b',`
	`  bind('c'),`
	`  ),`
	`)`
		Veiksmo argumentai gali turėti su argumentu susietas reikšmes.

`{a: b}`	`dict(`
	`  bind(`
	`  'a',`
	`  bind('c'),`
	`  )`
	`)`
		Sudėtinė duomenų struktūra.

`{a, b}`	`set(` ` bind('a'),	Reikšmių aibė.

	`    bind('b'),` `)`	
`a(*)`	`a(op('*'))`	Operatoriai gali būti naudojami kaip argumentai.