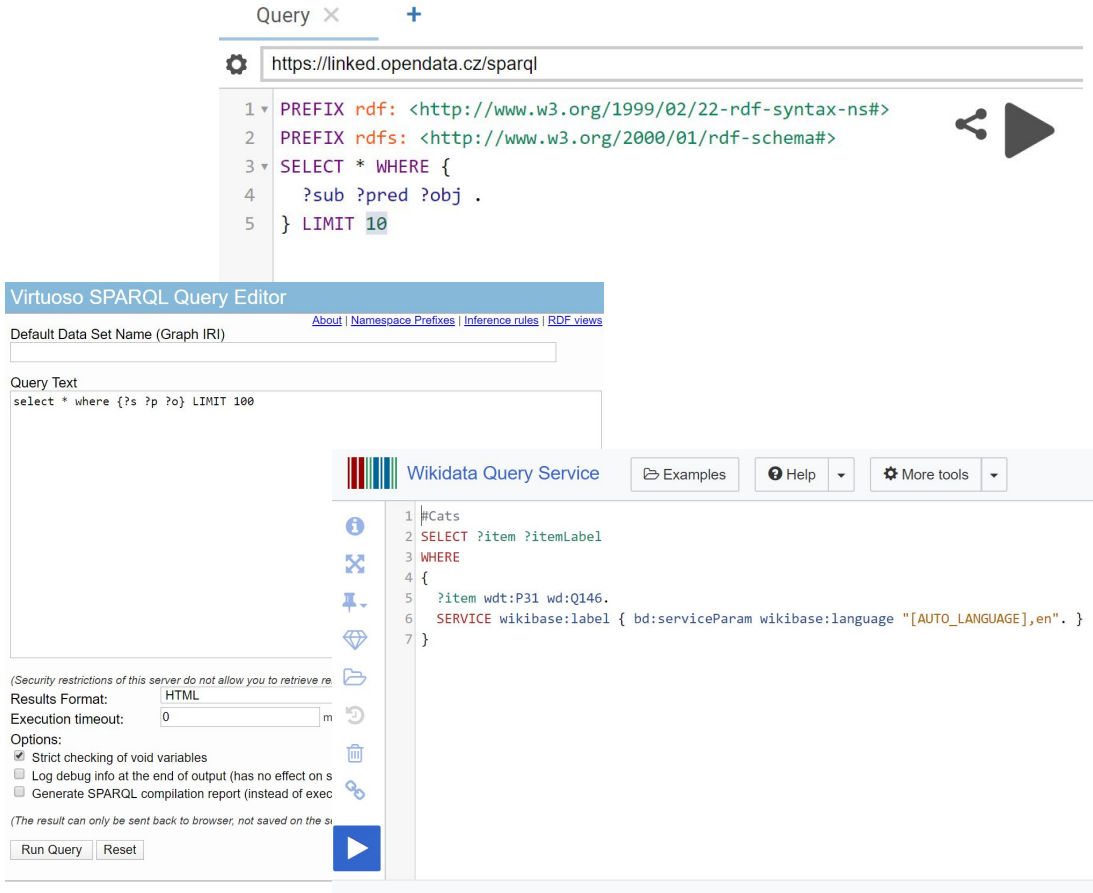
Graph data formats

SPARQL

SPARQL & SPARQL endpoints

- SPARQL
  - query language for RDF data
- SPARQL endpoint
  - HTTP-based web service
  - input: SPARQL query
  - output: data
    - RDF
    - CSV
    - JSON
    - XML
    - ...
  - typically including a user form
    - or use https://yasgui.triply.cc/

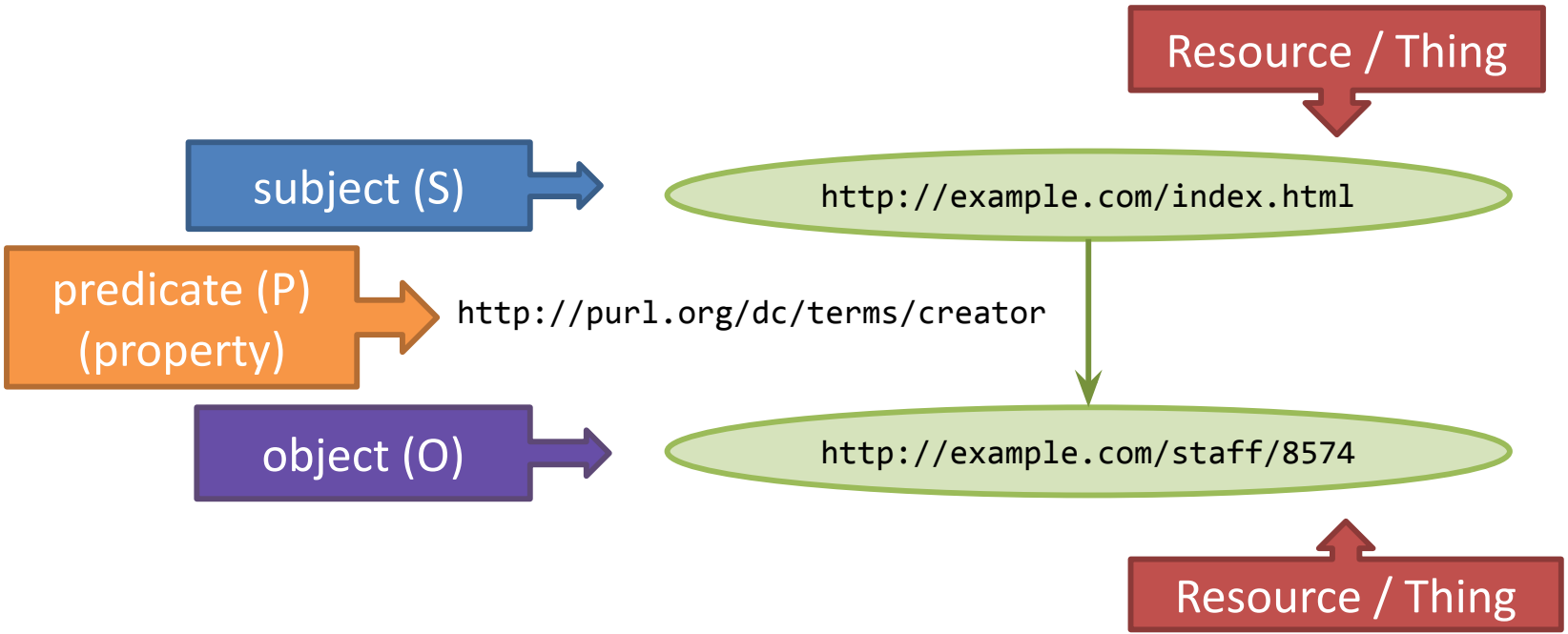


Existing SPARQL endpoint examples

- There is a list on GitHub
- Wikidata Query Service is quite popular
  - Wikidata is a community built database (like Wikipedia is a community built encyclopedia)
- The official portal for European data
  - datasets from all over Europe
- In Czechia
  - National Open Data Catalog
  - Registr práv a povinností (one of our base registries)
  - Czech Social Security Administration
- User friendly SPARQL query editor Yasgui
  - can be used instead the less friendly Virtuoso SPARQL editor

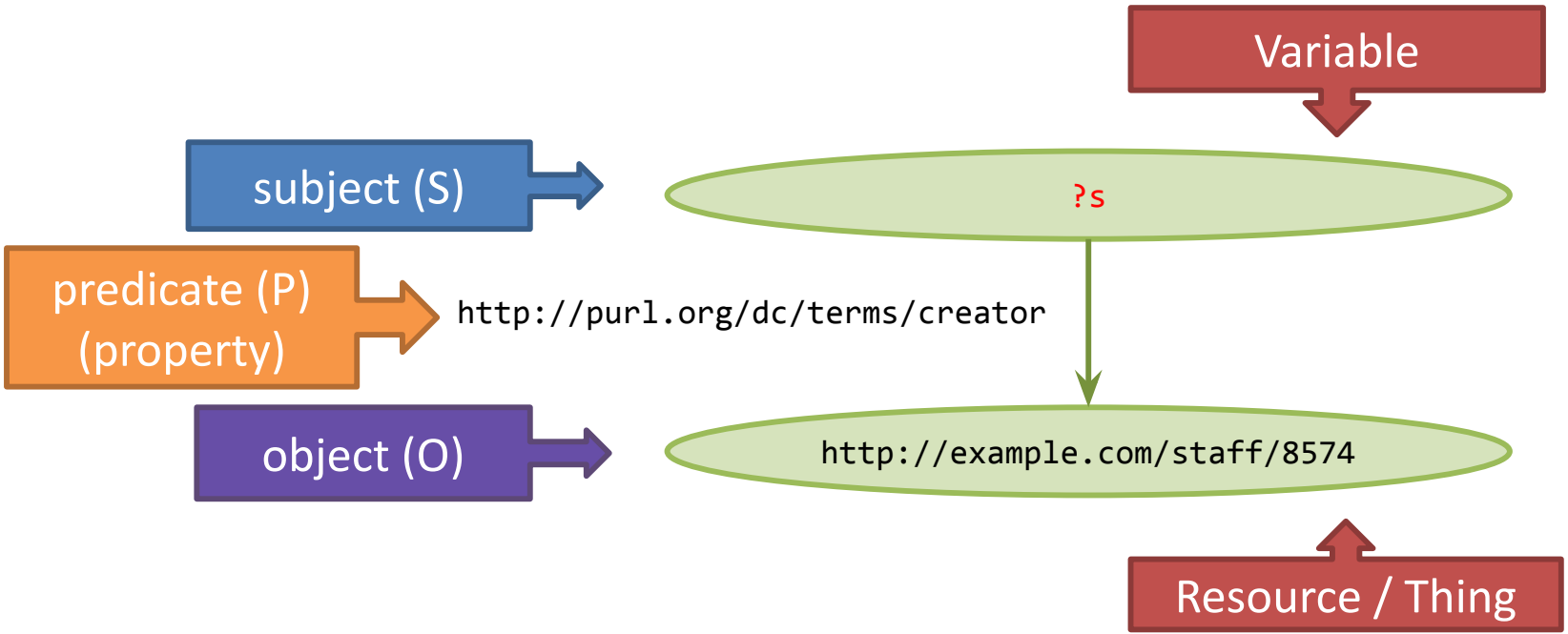
RDF model: a triple, a statement

```
<http://example.com/index.html> <http://purl.org/dc/terms/creator> <http://example.com/staff/8574> .
```

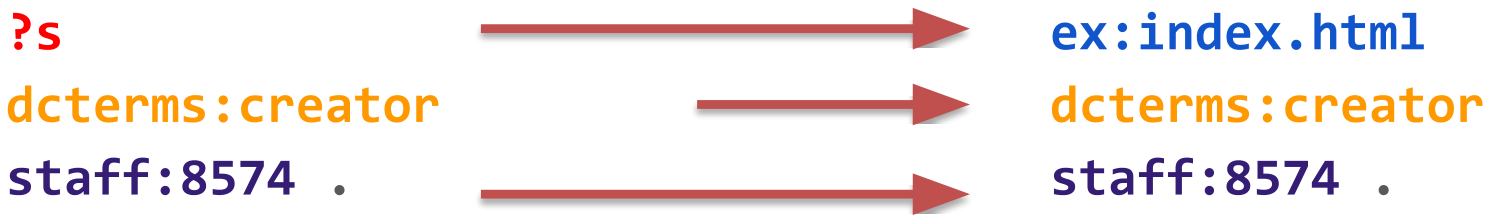


SPARQL querying - variable

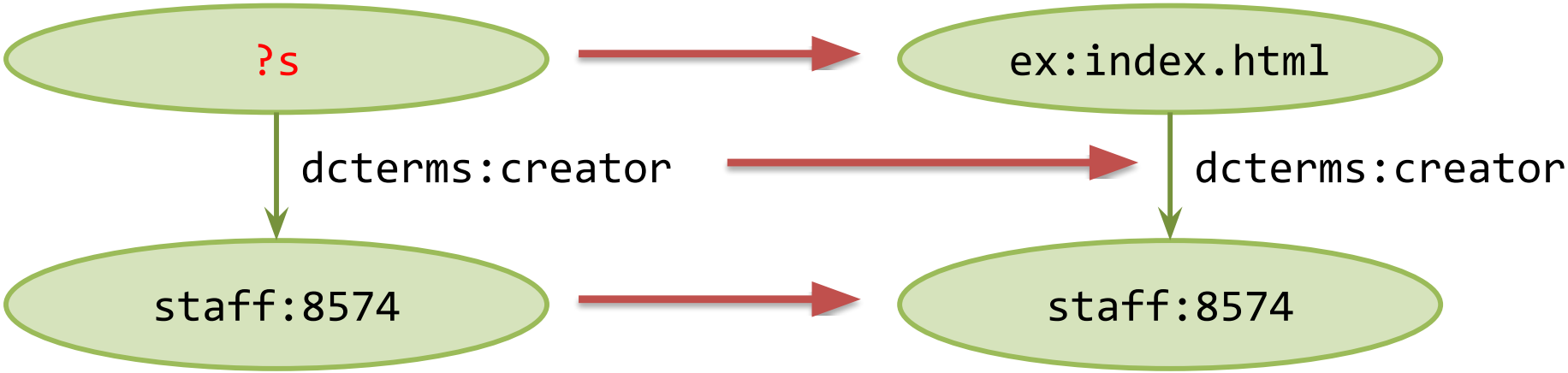
```
?s                    <http://purl.org/dc/terms/creator> <http://example.com/staff/8574> .
```



SPARQL querying - graph pattern matching



It's a match!



SPARQL querying - variable binding

```
?s
dcterms:creator
staff:8574 .
```

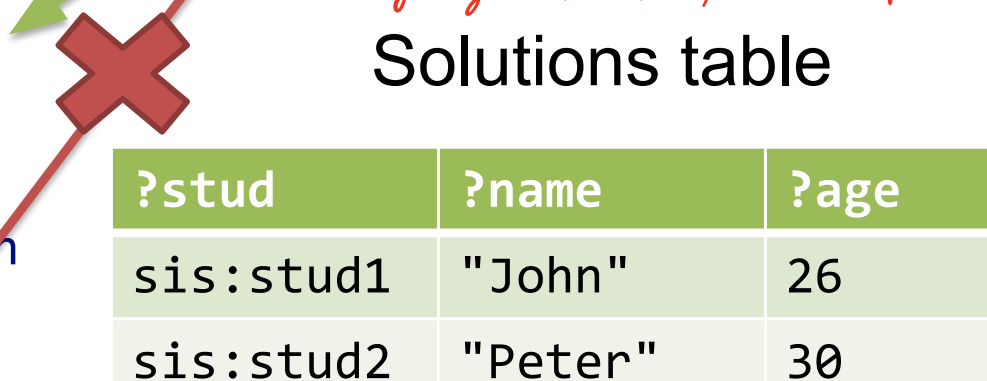
```
ex:index.html
dcterms:creator
staff:8574 .
```

?s <- ex:index.html

SPARQL querying - more triples

```
sis:stud1 sis:name "John" ;
          sis:age 26 ;
          rdf:type sis:Person
.
sis:stud2 sis:name "Peter" ;
          sis:age 30 ;
          rdf:type sis:Person
.
sis:stud3 sis:name "Martin" ;
          sis:age 20 .
```

```
?stud rdf:type sis:Person ;
      sis:name ?name ;
      sis:age ?age .
```



—> everything must match, what is fixed

Solutions table

?stud	?name	?age
sis:stud1	"John"	26
sis:stud2	"Peter"	30

SPARQL querying - optional graph pattern matching

```
sis:stud1 sis:name "John" ;
          sis:age 26 ;
          rdf:type sis:Person .

sis:stud2 sis:name "Peter" ;
          sis:age 30 ;
          rdf:type sis:Person .

sis:stud3 sis:name "Martin" ;
          sis:age 20 .
```

```
?stud sis:name ?name ;
      sis:age ?age .
OPTIONAL {
      ?stud rdf:type ?type.
}
```

not required for solution

Solutions table

?stud	?name	?age	?type
sis:stud1	"John"	26	sis:Person
sis:stud2	"Peter"	30	sis:Person
sis:stud3	"Martin"	20	NOT BOUND

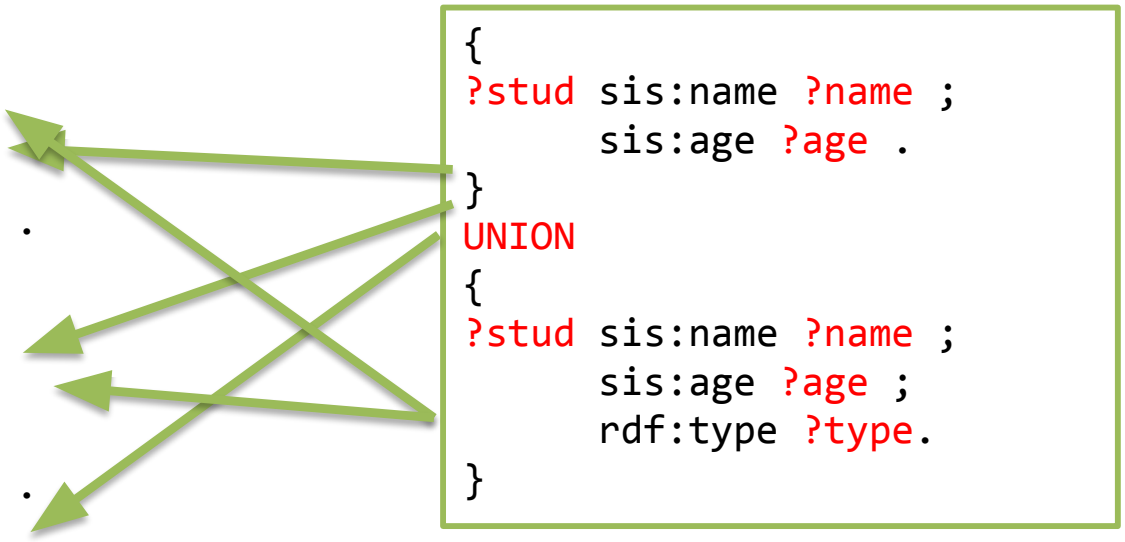
SPARQL querying - multiple optional graph patterns

```
?stud rdf:type ?type .
OPTIONAL {
    ?stud sis:age ?age .
}
OPTIONAL {
    ?stud sis:name ?name .
}
```

```
OPTIONAL { ?dataset dcterms:temporal ?temporal .
    OPTIONAL { ?temporal dcat:startDate ?startDate . }
    OPTIONAL { ?temporal dcat:endDate ?endDate . }
    OPTIONAL { ?temporal schema:startDate ?schemaStartDate . }
    OPTIONAL { ?temporal schema:endDate ?schemaEndDate . }
    BIND(IF(BOUND(?startDate), ?startDate, ?schemaStartDate) AS
?finalStartDate)
    BIND(IF(BOUND(?endDate), ?endDate, ?schemaEndDate) AS
?finalEndDate)
  }

  OPTIONAL { ?dataset dcat:contactPoint ?cp .
    ?cp a ?cptype.
    OPTIONAL { ?cp vcard:fn ?cpfn . }
    OPTIONAL { ?cp vcard:hasEmail ?cpemail . }
  }

  OPTIONAL { ?dataset foaf:page ?page . }
  OPTIONAL { ?dataset dcterms:conformsTo ?conformsTo . }
  OPTIONAL { ?dataset dcat:spatialResolutionInMeters
?spatialResolution . }
  OPTIONAL { ?dataset dcat:temporalResolution ?temporalResolution .
}
  OPTIONAL { ?dataset dcterms:isPartOf ?topDataset . }
```

SPARQL querying - union graph pattern matching

```
sis:stud1 sis:name "John" ;
          sis:age 26 ;
          rdf:type sis:Person .

sis:stud2 sis:name "Peter" ;
          sis:age 30 ;
          rdf:type sis:Person .

sis:stud3 sis:name "Martin" ;
          sis:age 20 .
```



```
{
?stud sis:name ?name ;
      sis:age ?age .
}
UNION
{
?stud sis:name ?name ;
      sis:age ?age ;
      rdf:type ?type.
}
```

?stud	?name	?age	?type
sis:stud1	"John"	26	sis:Person
sis:stud2	"Peter"	30	sis:Person
sis:stud3	"Martin"	20	NOT BOUND
sis:stud1	"John"	26	NOT BOUND
sis:stud2	"Peter"	30	NOT BOUND

SPARQL - First complete query

```
PREFIX sis: <http://is.cuni.cz/studium/sis#>
SELECT ?name ?age
WHERE {
   ?stud a ?type ;
   sis:name ?name ;
   sis:age  ?age .
}
```

Query result

?name	?age
"John"	26
"Peter"	30

SPARQL - Prologue

Prefix

Like in Turtle, but without @ and without "."

```
PREFIX name: <http://www.my.cz/>
…
WHERE {
   name:local ?p ?o.
}
```

Relative URIs

Like in Turtle, but without @ and without "."

```
BASE <http://www.my.cz/>
…
WHERE {
    <sis> <http://www.my.cz/sis> ?o
}
```

SPARQL - Blank nodes

Blank nodes have temp. IDs, therefore next request returns blank nodes with different IDs

- Blank nodes in data
  - Distinct nodes within the **document scope**
- Blank nodes in query results
  - Distinct nodes within the **result scope**

Blank node identifiers in the result may not correspond to original blank node identifiers from the given document

SPARQL - (named) graph patterns

```
PREFIX foaf:
<http://xmlns.com/foaf/0.1/>
SELECT DISTINCT ?g
WHERE {
    GRAPH ?g {

        ?s a foaf:Person .

    }
}
```

```
PREFIX foaf:
<http://xmlns.com/foaf/0.1/>
SELECT DISTINCT ?s
WHERE {
    GRAPH <https://...> {

        ?s a foaf:Person .

    }
}
```

SPARQL - literals

Language tags

- `"Praha"`
- `"Praha"@cs`
- `"Prague"@en`

Typed literals

- `1 = "1"^^xsd:integer`
- `1.5 = "1.5"^^xsd:decimal`
- `1.0E6 = "1"^^xsd:double`
- `true = "true"^^xsd:boolean`

SPARQL - Simple query with FILTER

```
PREFIX sis: <http://is.cuni.cz/studium/sis#>
SELECT ?name ?age
WHERE {
  ?stud a ?type ;
        sis:name ?name ;
        sis:age  ?age .

  FILTER (?age > 27)
}
```

Query result

?name	?age
~~"John"~~	~~26~~
"Peter"	30

Example: Simple SPARQL query

Select URI and labels of all `skos:ConceptSchemes` from a given SPARQL endpoint, which are labeled „Paragraf“.

```
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>

SELECT *
WHERE {
  ?s a skos:ConceptScheme ;
     skos:prefLabel ?label .
  FILTER (?label = "Paragraf"@cs)
}
```

= something of type skos:ConceptScheme

= Paragraf filtering

SPARQL: logical expression operators

The usual operators

- Arithmetic operators
  - Unary + –
  - Binary + – * /
- Comparison operators
  - < <= >= >
  - = !=
- Logical connectives
  - ! && ||

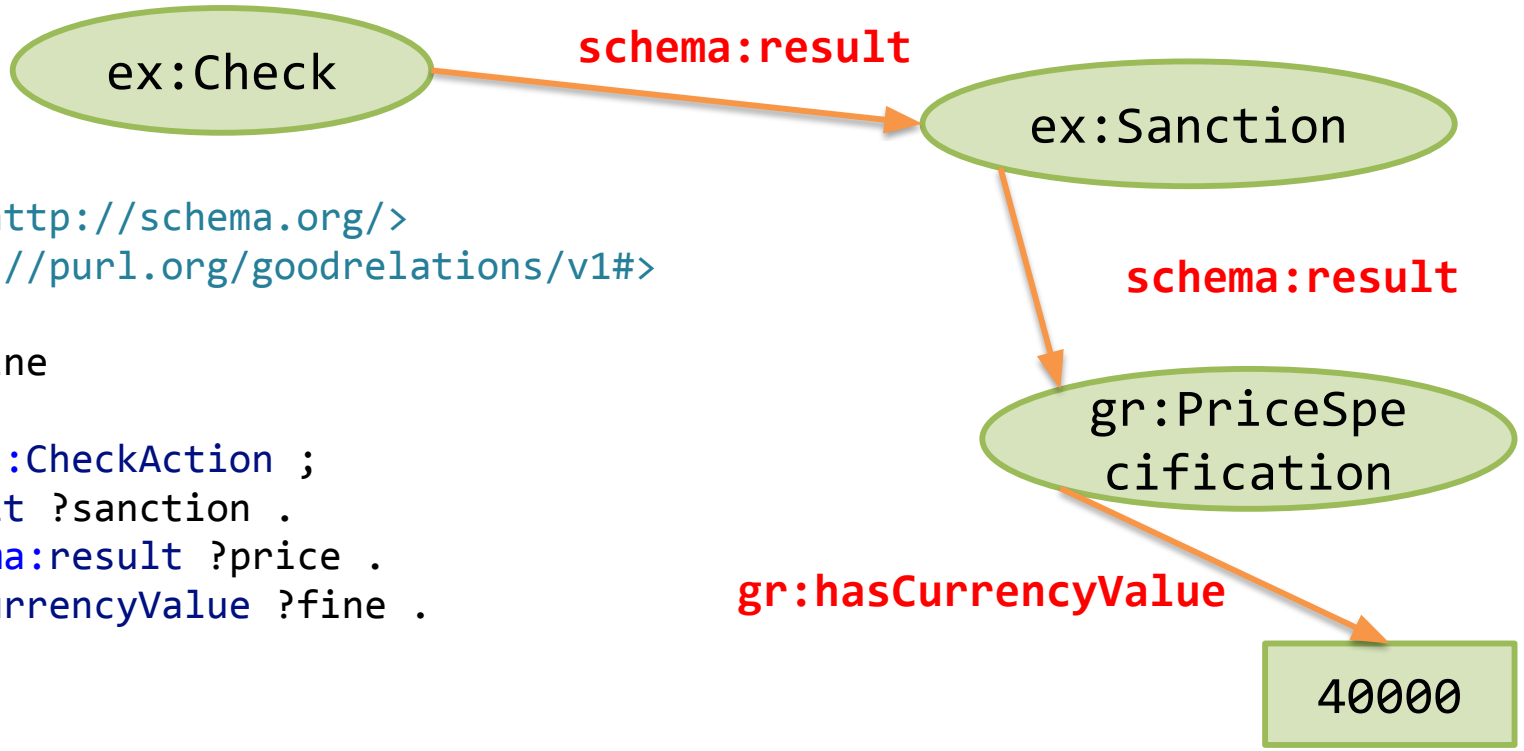
RDF specific operators

- Other operators
  - bound, isIri, isBlank, isLiteral
- Access functions
  - str, lang, dataType

SPARQL - property paths example

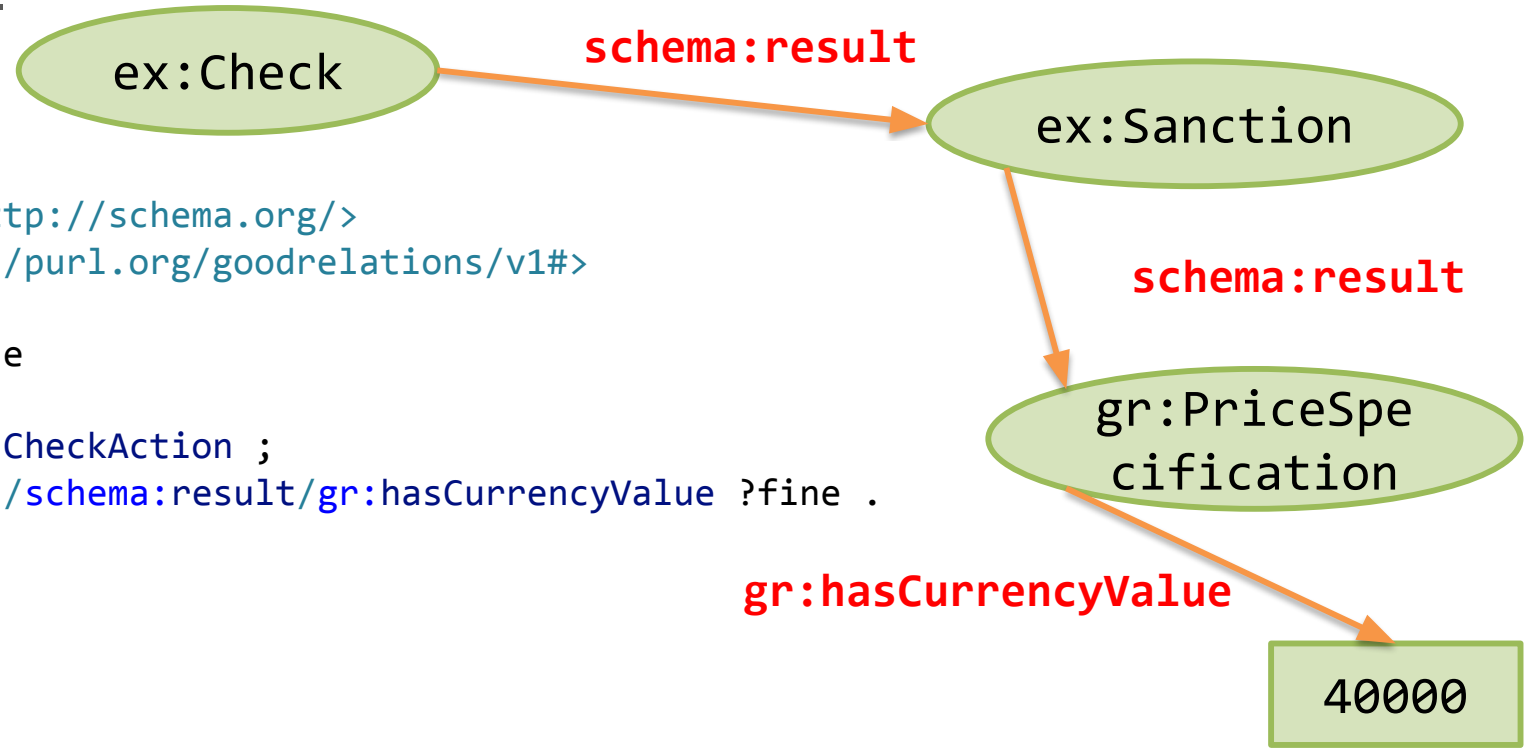
Select first 100 inspections of CTIA, which have a fine as a result.
Return its value.

```
PREFIX schema: <http://schema.org/>
PREFIX gr: <http://purl.org/goodrelations/v1#>

SELECT ?check ?fine
WHERE {
  ?check a schema:CheckAction ;
     schema:result ?sanction .
  ?sanction schema:result ?price .
  ?price gr:hasCurrencyValue ?fine .
}
LIMIT 100
```



SPARQL - property paths example

Select first 100 inspections of CTIA, which have a fine as a result.
Return its value.




```
PREFIX schema: <http://schema.org/>
PREFIX gr: <http://purl.org/goodrelations/v1#>

SELECT ?check ?fine
WHERE {
  ?check a schema:CheckAction ;
     schema:result/schema:result/gr:hasCurrencyValue ?fine .
}
LIMIT 100
```

SPARQL - property paths

- Sequences
  - `path1/path2`
- Alternatives
  - `path1|path2`
- Groups
  - `(path)`
- Negation
  - `!item`
  - `!(item1|item2|...)`

- Occurrences
  - `path*`
  - `path+`
  - `path?`
- Inverse paths (from object to subject)
  - ^path

SPARQL - aggregation example

What was the highest issued fine?

```
PREFIX schema: <http://schema.org/>
PREFIX gr: <http://purl.org/goodrelations/v1#>

SELECT (MAX (?fine) AS ?max)
WHERE
{
?check a schema:CheckAction ;
   schema:result/schema:result/gr:hasCurrencyValue ?fine .
}
```

SPARQL - aggregation

- COUNT
- SUM
- AVG
- MIN
- MAX
- GROUP_CONCAT(?x ; separator="|")
- SAMPLE

SPARQL - federated query

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?s ?name
WHERE {
     ?s a foaf:Person .
    SERVICE <https://peoplenames.org/sparql> {
          ?s foaf:name ?name .
    }
}
```

SPARQL - binding variables

```
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX ns: <http://example.org/ns#>

SELECT ?title ?price
WHERE { ?product ns:price ?p ;
                 ns:discount ?discount ;
                 dct:title ?title .
   BIND (?p * ( 1 - ?discount) AS ?price)
   FILTER(?price < 20)
}
```

SPARQL - binding variables example

Select first 100 inspections of CTIA, which have a fine as a result.
Return its value and whether it is higher than 1000.

```
PREFIX schema: <http://schema.org/>
PREFIX gr: <http://purl.org/goodrelations/v1#>

SELECT ?check ?fine ?higher
WHERE {
  ?check a schema:CheckAction ;
     schema:result/schema:result/gr:hasCurrencyValue ?fine .
  BIND(IF(?fine > 1000, true, false) AS ?higher)
}
LIMIT 100
```

SPARQL - functions

- if
  - `IF(?x = 2, "yes", "no")`
    i. evaluates first parameter
    ii. If true, returns second parameter
    iii. If false, returns third parameter
- coalesce
  - `COALESCE(1/0, ?x, ...)`
    i. returns first parameter which does not result in error
- `exists { graph pattern }`
- `not exists { graph pattern }`
- IN, NOT IN
  - `2 IN (<http://example/iri>, "str", 2.0) = true`

SPARQL - functions on RDF terms

- isIRI
- isBlank
- isLiteral
- isNumeric
- str
- lang
- datatype
- IRI
- STRDT
- STRLANG
- UUID

UUID()	<urn:uuid:b9302fb5-642e-4d3b-af19-29a8f6d894c9>

STRLANG("chat", "en")	"chat"@en

STRDT("123", xsd:integer)	"123"^^<http://www.w3.org/2001/XMLSchema#integer>
STRDT("iiii", <http://ex/roman>)	"iiii"^^<http://ex/roman>

SPARQL - functions on strings

- STRLEN
- SUBSTR
- UCASE, LCASE
- STRSTARTS, STRENDS
- CONTAINS
- STRBEFORE, STRAFTER
- ENCODE_FOR_URI
- CONCAT
- langMatches
- REGEX
- REPLACE

substr("foobar"@en, 4, 1)	"b"@en

strbefore("abc","b")	"a"
strbefore("abc"@en,"bc")	"a"@en
strbefore("abc"@en,"b"@cy)	error

encode_for_uri("Los Angeles")	"Los%20Angeles"

replace("abcd", "b", "Z")	"aZcd"
replace("abab", "B", "Z","i")	"aZaZ"
replace("abab", "B.", "Z","i")	"aZb"

SPARQL - another binding variables example

Create IRI for entities with IDs
the Frequency EU vocabulary

?freq
MONTHLY
ANNUAL
BIENNIAL
…

```
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX nkod: <https://data.gov.cz/slovník/nkod/>

CONSTRUCT {
    ?dataset dcterms:accrualPeriodicity ?frequency .
}
WHERE
{
    ?dataset nkod:accrualPeriodicity ?freq .

    BIND(IRI(CONCAT("http://publications.europa.eu/resource/authority/frequency/", ?freq)) AS ?frequency)
}
```

```
<http://publications.europa.eu/resource/authority/frequency/MONTHLY>
<http://publications.europa.eu/resource/authority/frequency/ANNUAL>
…
```

SPARQL - functions on numerics

- `abs`
- `round`
- `ceil`
- `floor`
- `RAND`

SPARQL - functions on dates and times

- now
- year
- month
- day
- hours
- minutes
- seconds
- timezone
- tz

timezone("2011-01-10T14:45:13.815-05:00"^^xsd:dateTime)	"-PT5H"^^xsd:dayTimeDuration

tz("2011-01-10T14:45:13.815-05:00"^^xsd:dateTime)	"-05:00"

now()	"2011-01-10T14:45:13.815-05:00"^^xsd:dateTime

day("2011-01-10T14:45:13.815-05:00"^^xsd:dateTime)	10

SPARQL - subqueries

Inner queries evaluated first.

```
PREFIX : <https://example.org/>
SELECT ?company ?maxFine
WHERE {
  ?company a schema:Organization .
  {
    SELECT ?company (MAX(?fine) AS ?maxFine)
    WHERE {
      ?company :fine ?fine .
    } GROUP BY ?company
  }
}
```

SPARQL - VALUES example

```
@prefix dct:   <http://purl.org/dc/terms/> .
@prefix :      <http://example.org/book/> .
@prefix ns:    <http://example.org/ns#> .

:book1  dct:title  "SPARQL Tutorial"@en ;
        ns:price  42 .
:book2  dct:title  "The Semantic Web"@en ;
        ns:price  23 .
```

```
PREFIX dct:   <http://purl.org/dc/terms/>
PREFIX :      <http://example.org/book/>
PREFIX ns:    <http://example.org/ns#>

SELECT ?book ?title ?price
WHERE {
   VALUES ?book { :book1 :book3 }
   ?book dct:title ?title ;
         ns:price ?price .
}
```

book	title	price
<http://example.org/book/book1>	"SPARQL Tutorial"	42

SPARQL - VALUES example - codelist translation

```
PREFIX dcat: <http://www.w3.org/ns/dcat#>
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX f: <http://publications.europa.eu/resource/authority/frequency/>
CONSTRUCT {
  ?dataset dcterms:accrualPeriodicity ?new.
  ?new a skos:Concept, dcterms:Frequency .
}
WHERE {
  ?dataset dcterms:accrualPeriodicity ?f .

VALUES (?f ?new) {
  ("R/P1M" f:MONTHLY)
  ("R/P1Y" f:ANNUAL)
  ("R/P2Y" f:BIENNIAL)
  …
  ("jednorázově" f:NEVER)
  ("nikdy" f:NEVER)
}
}
```

2 - tady se mutlohuje jedna strana podle druhé

```
<#dataset1> dcterms:accrualPeriodicity "R/P1M" .
<#dataset2> dcterms:accrualPeriodicity "R/P2Y" .
```

```
<#dataset1> dcterms:accrualPeriodicity f:MONTHLY .
f:MONTHLY a skos:Concept, dcterms:Frequency .

<#dataset2> dcterms:accrualPeriodicity f:BIENNIAL .
f:BIENNIAL a skos:Concept, dcterms:Frequency .
```

SPARQL - result forms - SELECT and ASK

- SELECT
  - Result: Solutions table (e.g. in CSV or JSON)
- ASK
  - Checks whether at least one result item exists
  - Result: True or false

Is there a CTIA inspection with a resulting fine higher than 5 000 000?

```
PREFIX schema: <http://schema.org/>
PREFIX gr: <http://purl.org/goodrelations/v1#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
ASK {
?check a schema:CheckAction ;
   schema:result/schema:result/gr:hasCurrencyValue ?fine .
FILTER(?fine > 5000000)
}
```

SPARQL - result forms - CONSTRUCT

Result: RDF graph constructed from a template.

```
CONSTRUCT
{ ?s sis:name ?fullName }
WHERE {
  ?s sis:firstName ?n1 ;
     sis:lastName ?n2 .
  BIND(STRLANG(CONCAT(?n1, " ", ?n2), "cs") AS ?fullName)
}
```

SPARQL - result forms - DESCRIBE

Result: RDF graph about the given resource.

Specification non-normative - various triplestores implement it differently.

Example: `DESCRIBE <http://www.my.cz/>`

SPARQL SELECT - modifiers

ORDER BY

- Orders items in the solutions table
- Hierarchical ordering criteria
  - ASC = ascending (default)
  - DESC = descending
- Unbound variable < blank node < URI < literal
- Example
  - `ORDER BY ASC(?name), DESC(?age)`

SPARQL SELECT - ORDER BY example

```
PREFIX     :  <http://example.org/ns#>
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>

SELECT ?name
WHERE { ?x foaf:name ?name ;
         :empId ?emp }
ORDER BY ?name DESC(?emp)
```

SPARQL SELECT - modifiers

LIMIT

- Limits the number of items in the result sequence
- Always should be preceded by ORDER BY modifier
  - Because RDF graph is a set with no ordering
- Used as page size
- Example: `ORDER BY ?name LIMIT 10`

OFFSET

- Index of the first reported item from the sequence
- Used for paging through results
- Example: `ORDER BY ?name LIMIT 10 OFFSET 20`

SPARQL SELECT - GROUP BY, HAVING

```
PREFIX : <http://data.example/>
SELECT (AVG(?size) AS ?asize)
WHERE {
  ?x :size ?size
}
GROUP BY ?x
HAVING(AVG(?size) > 10)
```

SPARQL 1.1

W3C Recommendation

- 21 March 2013
- Set of recommendations
  - SPARQL 1.1 Overview
  - SPARQL 1.1 Query Language
  - SPARQL 1.1 Update
  - SPARQL 1.1 Service Description
  - SPARQL 1.1 Federated Query
  - SPARQL 1.1 Query Results JSON Format
  - SPARQL 1.1 Query Results CSV and TSV Formats
  - SPARQL Query Results XML Format (Second Edition)
  - SPARQL 1.1 Entailment Regimes
  - SPARQL 1.1 Protocol
  - SPARQL 1.1 Graph Store HTTP Protocol