Nuts & Bolts

URIs

Write full URIs:

```
<http://this.is.a/full/URI/written#out>
```

Abbreviate URIs with prefixes:

```
PREFIX foo: <http://this.is.a/URI/prefix#>
… foo:bar …
    ⇨  http://this.is.a/URI/prefix#bar
```

Shortcuts:

```
a  ⇨  rdf:type
```

Literals

Plain literals:

```
"a plain literal"
```

Plain literal with language tag:

```
"bonjour"@fr
```

Typed literal:

```
"13"^^xsd:integer
```

Shortcuts:

```
true ⇨ "true"^^xsd:boolean
3    ⇨ "3"^^xsd:integer
4.2  ⇨ "4.2"^^xsd:decimal
```

Variables

Variables:

```
?var1, ?anotherVar, ?and_one_more
```

Comments

Comments:

```
# Comments start with a '#'
# continue to the end of the line
```

Triple Patterns

Match an exact RDF triple:

```
ex:myWidget ex:partNumber "XY24Z1" .
```

Match one variable:

```
?person foaf:name "Lee Feigenbaum" .
```

Match multiple variables:

```
conf:SemTech2009 ?property ?value .
```

Common Prefixes

prefix…	…stands for
rdf:	http://xmlns.com/foaf/0.1/
rdfs:	http://www.w3.org/2000/01/rdf-schema#
owl:	http://www.w3.org/2002/07/owl#
xsd:	http://www.w3.org/2001/XMLSchema#
dc:	http://purl.org/dc/elements/1.1/
foaf:	http://xmlns.com/foaf/0.1/

More common prefixes at http://prefix.cc

Anatomy of a Query

Declare prefix shortcuts *(optional)* →

```
PREFIX foo: <…>
PREFIX bar: <…>
…
SELECT …
```

← Query result clause

Define the dataset *(optional)*

```
FROM <…>
FROM NAMED <…>
WHERE {
  …
}
```

← Query pattern

Query modifiers *(optional)*

```
GROUP BY …
HAVING …
ORDER BY …
LIMIT …
OFFSET …
BINDINGS …
```

4 Types of SPARQL Queries

SELECT queries

Project out specific variables and expressions:

```
SELECT ?c ?cap (1000 * ?people AS ?pop)
```

Project out all variables:

```
SELECT *
```

Project out distinct combinations only:

```
SELECT DISTINCT ?country
```

Results in a table of values (in XML or JSON):

?c	?cap	?pop
ex:France	ex:Paris	63,500,000
ex:Canada	ex:Ottawa	32,900,000
ex:Italy	ex:Rome	58,900,000

CONSTRUCT queries

Construct RDF triples/graphs:

```
CONSTRUCT {
   ?country a ex:HolidayDestination ;
        ex:arrive_at ?capital ;
        ex:population ?population .
}
```

Results in RDF triples (in any RDF serialization):

```
ex:France a ex:HolidayDestination ;
    ex:arrive_at ex:Paris ;
    ex:population 635000000 .
ex:Canada a ex:HolidayDestination ;
    ex:arrive_at ex:Ottawa ;
    ex:population 329000000 .
```

ASK queries

Ask whether or not there are any matches:

```
ASK
```

Result is either "true" or "false" (in XML or JSON):

```
true, false
```

DESCRIBE queries

Describe the resources matched by the given variables:

```
DESCRIBE ?country
```

Result is RDF triples (in any RDF serialization) :

```
ex:France a geo:Country ;
  ex:continent geo:Europe ;
  ex:flag <http://…/flag-france.png> ;
  …
```

Combining SPARQL Graph Patterns

*Consider **A** and **B** as graph patterns.*

A Basic Graph Pattern – one or more triple patterns

```
A . B
```

⇨ Conjunction. Join together the results of solving A and B by matching the values of any variables in common.

Optional Graph Patterns

```
A OPTIONAL { B }
```

⇨ Left join. Join together the results of solving A and B by matching the values of any variables in common, if possible. Keep all solutions from A whether or not there's a matching solution in B

Combining SPARQL Graph Patterns

*Consider **A** and **B** as graph patterns.*

Either-or Graph Patterns

```
{ A } UNION { B }
```

⇨ Disjunction. Include both the results of solving A and the results of solving B.

"Subtracted" Graph Patterns (SPARQL 1.1)

```
A MINUS { B }
```

⇨ Negation. Solve A. Solve B. Include only those results from solving A that are *not compatible* with any of the results from B.

SPARQL Subqueries *(SPARQL 1.1)*

*Consider **A** and **B** as graph patterns.*

```
A .
{
  SELECT …
  WHERE {
    B
  }
}
C .
```

⇨ Join the results of the subquery with the results of solving A and C.

SPARQL Filters

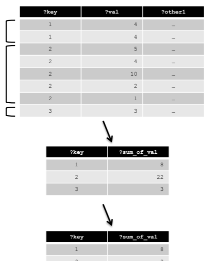
- *SPARQL **FILTER**s eliminate solutions that do not cause an expression to evaluate to true.*
- *Place **FILTER**s in a query inline within a basic graph pattern*

```
A . B . FILTER ( …expr… )
```

Category	Functions / Operators	Examples
Logical	`!, &&, ||, =, !=, <, <=, >, >=`	`?hasPermit || ?age < 25`
Math	`+, -, *, /`	`?decimal * 10 > ?minPercent`
Existence *(SPARQL 1.1)*	`EXISTS, NOT EXISTS`	`NOT EXISTS { ?p foaf:mbox ?email }`
SPARQL tests	`isURI, isBlank, isLiteral, bound`	`isURI(?person) || !bound(?person)`
Accessors	`str, lang, datatype`	`lang(?title) = "en"`
Miscellaneous	`sameTerm, langMatches, regex`	`regex(?ssn, "\\d{3}-\\d{2}-\\d{4}")`

Aggregates *(SPARQL 1.1)*

1. Partition results into groups based on the expression(s) in the **GROUP BY** clause
2. Evaluate projections and aggregate functions in **SELECT** clause to get one result per group
3. Filter aggregated results via the **HAVING** clause



?key	?val	?other1
1	4	…
1	4	…
2	5	…
2	4	…
2	10	…
2	2	…
2	1	…
3	3	…

?key	?sum_of_val
1	8
2	22
3	3

?key	?sum_of_val
1	8
3	3

SPARQL 1.1 includes: COUNT, SUM, AVG, MIN, MAX, SAMPLE, GROUP_CONCAT

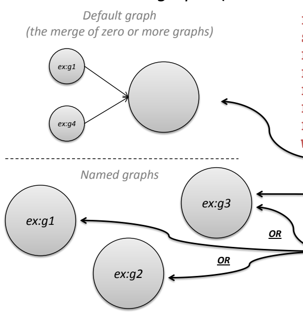
Property Paths *(SPARQL 1.1)*

- Property paths allow triple patterns to match arbitrary-length paths through a graph
- Predicates are combined with regular-expression-like operators:

Construct	Meaning
`path1/path2`	Forwards path (`path1` followed by `path2`)
`^path1`	Backwards path (object to subject)
`path1|path2`	Either `path1` or `path2`
`path1*`	`path1`, repeated zero or more times
`path1+`	`path1`, repeated one or more times
`path1?`	`path1`, optionally
`path1{m,n}`	At least `m` and no more than `n` occurrences of `path1`
`path1{n}`	Exactly `n` occurrences of `path1`
`path1{m,}`	At least `m` occurrences of `path1`
`path1{,n}`	At most `n` occurrences of `path1`

RDF Datasets

A SPARQL queries a *default graph* (normally) and zero or more *named graphs* (when inside a **GRAPH** clause).



```
PREFIX ex: <…>
SELECT …
FROM ex:g1
FROM ex:g4
FROM NAMED ex:g1
FROM NAMED ex:g2
FROM NAMED ex:g3
WHERE {
   … A …
   GRAPH ex:g3 {
     … B …
   }
   GRAPH ?g {
     … C …
   }
}
```

SPARQL Over HTTP (the SPARQL Protocol)

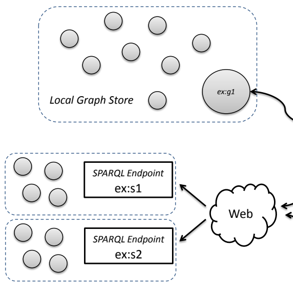
`http://host.domain.com/sparql/endpoint?`*<parameters>*

where *<parameters>* can include:

`query=`*<encoded query string>*
  e.g. `SELECT+*%0DWHERE+{…`
`default-graph-uri=`*<encoded graph URI>*
  e.g. `http%3A%2F%2Fexmaple.com%2Ffoo…`
  n.b. zero of more occurrences of `default-graph-uri`
`named-graph-uri=`*<encoded graph URI>*
  e.g. `http%3A%2F%2Fexmaple.com%2Fbar…`
  n.b. zero of more occurrences of `named-graph-uri`

HTTP `GET` or `POST`. Graphs given in the protocol override graphs given in the query.

Federated Query *(SPARQL 1.1)*



```
PREFIX ex: <…>
SELECT …
FROM ex:g1
WHERE {
   … A …
  SERVICE ex:s1 {
     … B …
  }
  SERVICE ex:s2 {
     … C …
  }
}
```

SPARQL 1.1 Update

SPARQL Update Language Statements
`INSERT DATA { triples }`
`DELETE DATA {triples}`
`[ DELETE { template } ] [ INSERT { template } ] WHERE { pattern }`
`LOAD <uri> [ INTO GRAPH <uri> ]`
`CLEAR GRAPH <uri>`
`CREATAE GRAPH <uri>`
`DROP GRAPH <uri>`

[…] denotes optional parts of SPARQL 1.1 Update syntax

Some Public SPARQL Endpoints

Name	URL	What's there?
SPARQLer	http://sparql.org/sparql.html	General-purpose query endpoint for Web-accessible data
DBPedia	http://dbpedia.org/sparql	Extensive RDF data from Wikipedia
DBLP	http://www4.wiwiss.fu-berlin.de/dblp/snorql/	Bibliographic data from computer science journals and conferences
LinkedMDB	http://data.linkedmdb.org/sparql	Films, actors, directors, writers, producers, etc.
World Factbook	http://www4.wiwiss.fu-berlin.de/factbook/snorql/	Country statistics from the CIA World Factbook
bio2rdf	http://bio2rdf.org/sparql	Bioinformatics data from around 40 public databases

Cheatography

Semantic Web Cheat Sheet
by asselin via cheatography.com/433/cs/201/

xsd Types (cont)
xsd:double
Decimal-derived
xsd:integer
xsd:nonNegativeInteger
xsd:positiveInteger
xsd:nonPositiveInteger
xsd:negativeInteger
xsd:long
xsd:int
xsd:short
xsd:byte
xsd:unsignedLong
xsd:unsignedInt
xsd:unsignedShort
xsd:unsignedByte
Binary
xsd:hexBinary
xsd:base64Binary
Date/Time-related
xsd:dateTime
xsd:time
xsd:date
xsd:gYearMonth
xsd:gYear
xsd:gMonthDay
xsd:gDay
xsd:gMonth
Resource
xsd:anyURI

Classes, Parents, Instances of

Class	Subclass of	Instance of
rdfs:Resource	rdfs:Resource	rdfs:Class
rdfs:Class	rdfs:Resource	rdfs:Class
rdfs:Literal	rdfs:Resource	rdfs:Class
rdfs:Datatype	rdfs:Class	rdfs:Class
rdf:XMLLiteral	rdfs:Literal	rdfs:Datatype
rdf:Property	rdfs:Resource	rdfs:Class
rdfs:Container	rdfs:Resource	rdfs:Class
rdf:Alt	rdfs:Container	rdfs:Class
rdf:Bag	rdfs:Container	rdfs:Class
rdf:Seq	rdfs:Container	rdfs:Class
rdfs:ContainerMembershipProperty	rdf:Property	rdfs:Class
rdf:List	rdfs:Resource	rdfs:Class
rdf:Statement	rdfs:Resource	rdfs:Class

SHACL Core Cheat Sheet

1) "Core Core" *(note: there's no such thing as "core core," we invented that)*

a) Node shapes
 i) sh:NodeShape
b) Property shapes
 i) sh:property
 ii) sh:path
c) Constraint components
 i) Cardinality
  (1) sh:minCount
  (2) sh:maxCount
 ii) Value types
  (1) sh:datatype
   (a) xsd:
   (b) custom
  (2) sh:class
  (3) sh:nodeKind
   (a) sh:IRI
   (b) sh:BlankNode
   (c) sh:Literal
   (d) sh:BlankNodeOrLiteral
   (e) sh:BlankNodeOrIRI
   (f) sh:IRIOrLiteral
  (4) *Sets*: sh:in
  (5) *Specific value*: sh:hasValue
 iii) Value ranges
  (1) sh:minInclusive
  (2) sh:maxInclusive
  (3) sh:minExclusive
  (4) sh:maxExclusive
 iv) String-based
  (1) sh:minLength
  (2) sh:maxLength
  (3) sh:length
  (4) sh:pattern
   (a) *optional*: sh:flags
 v) Language-based
  (1) sh:languageIn
  (2) sh:uniqueLang
 vi) Logical
  (1) sh:and
  (2) sh:or
  (3) sh:not
  (4) sh:xone
 vii) Shape-based
  (1) sh:node
  (2) *(See "Intermediate Core" below):*
   (a) sh:property
   (b) sh:qualifiedValueShape
   (c) sh:qualifiedValueShapeDisjoint
   (d) sh:qualifiedMinCount
   (e) sh:qualifiedMaxCount
 viii) Closed shape Choisir l'affichage de la barre latérale *"Intermediate Core" below)*
 ix) Property pairs *(see "Intermediate Core" below)*
 x) Non-validating *(see "Intermediate Core" below)*
d) Target declarations
 i) sh:targetNode
 ii) sh:targetClass
 iii) sh:targetSubjectsOf
 iv) sh:targetObjectsOf
e) Validation reporting
 i) sh:message
 ii) sh:severity

2) Intermediate Core *(note: there's no such thing as "intermediate core,"*

f) Importing and referencing (Gayo 5.6.6)
 i) owl:imports
 ii) sh:deactivated
g) Combining logical operators (Gayo 5.11.5)
 i) If-then
 ii) If-then-else
h) Shape based constraints (Gayo 5.12)
 i) The constraints:
  (1) sh:node
  (2) sh:property
  (3) qualified value shapes:
   (a) sh:qualifiedValueShape
   (b) sh:qualifiedValueShapeDisjoint
   (c) sh:qualifiedMinCount
   (d) sh:qualifiedMaxCount
 ii) Shape references and recursion
i) Closed shapes
 i) sh:closed
 ii) sh:ignoredProperties
j) Property pair constraints
 i) sh:equals
 ii) sh:disjoint
 iii) sh:lessThan
 iv) sh:lessThanOrEquals
k) Non-validating constraints
 i) sh:name
 ii) sh:description
 iii) sh:order
 iv) sh:group
 v) sh:defaultValue
l) SHACL paths

SHACL path	SPARQL path
schema:name	schema:name
[sh:inversePath schema:knows]	^schema:knows
(schema:knows schema:name)	schema:knows/schema:name
[sh:alternativePath (schema:knows schema:follows)]	schema:knows| schema:follows
[sh:zeroOrOnePath schema:knows]	schema:knows?
[sh:oneOrMorePath schema:knows]	schema:knows+
([sh:zeroOrMorePath schema:knows] schema:name)	schema:knows*/schema:name

Cheatography

Semantic Web Cheat Sheet
by asselin via cheatography.com/433/cs/201/

Predefined RDF classes

Class	Description
rdfs:Resource	The class resource, everything.
rdfs:Class	The class of classes.
rdf:Property	The class of RDF properties.
rdf:Container	Superclass of all container types
rdf:Bag	Unordered container
rdf:Seq	Sequentially ordered container
rdf:Alt	Container of alternatives
rdf:List	List of items
rdfs:ContainerMembershipProp-erty	The class of container membership properties, rdf:_1, rdf:_2, ...
rdfs:Datatype	The class of RDF datatypes.
rdf:XMLLiteral	The class of XML literals values.
rdfs:Literal	The class of literal values, e.g. textual strings and integers.
rdf:Statement	The class of RDF statements.

rdfs:Resource properties

Property Name	Type (Range)	Description
rdf:type	rdfs:Class	The subject is an instance of a class.
rdfs:label	rdfs:Literal	A human-readable name for the subject.
rdfs:comment	rdfs:Literal	A description of the subject resource.
rdfs:member	rdfs:Resource	A member of the subject resource.
rdfs:seeAlso	rdfs:Resource	The definition of the subject resource.
rdfs:isDefinedBy	rdfs:Resource	Namespace where it is defined
rdf:value	rdfs:Resource	Idiomatic property used for structured values

rdfs:Class properties

Property Name	Type (Range)	Description
rdfs:subClassOf	rdfs:Class	The subject is a subclass of a class.

owl:Class properties

Property Name	Type (Range)	Description
owl:disjointWith *	owl:Class	Disjoint Classes
owl:intersectionOf -	rdf:List	Intersection of 2 classes

owl:Class properties (cont)

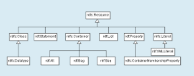
owl:unionOf *	rdf:List	Union of 2 classes
owl:complementOf *	owl:Class	Complement of a class
owl:oneOf *	rdf:List	Enumerated class
owl:equivalentClass -	owl:Class	Equivalent Classes

- Denotes a construct available in OWL Lite. * denotes a construct available in OWL DL.

rdf:Property properties

Property Name	Type (Range)	Description
rdfs:domain	rdfs:Class	A domain of the subject property.
rdfs:range	rdfs:Class	A range of the subject property.
owl:equivalentProperty	rdf:Property	
rdfs:subPropertyOf	rdf:Property	The subject is a subproperty of a property.

Inheritance Diagram



Sponsored by **Readability-Score.com**
Measure your website readability!
https://readability-score.com

Cheatography

Semantic Web Cheat Sheet
by asselin via cheatography.com/433/cs/201/

rdf:List properties

Property Name	Type (Range)	Description
rdf:first	rdfs:Resource	The first item in the subject RDF list.
rdf:rest	rdfs:List	The rest of the subject RDF list after the first item.

rdf:Statement properties

Property Name	Type (Range)	Description
rdf:subject	rdfs:Resource	The subject of the subject RDF statement.
rdf:predicate	rdfs:Resource	The predicate of the subject RDF statement.
rdf:object	rdfs:Resource	The object of the subject RDF statement.

URL prefixes

prefix...	...stands for
rdf:	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs:	http://www.w3.org/2000/01/rdf-schema#
owl:	http://www.w3.org/2002/07/owl#
xsd:	http://www.w3.org/2001/XMLSchema#
dc:	http://purl.org/dc/elements/1.1/
foaf:	http://xmlns.com/foaf/0.1/

owl:Thing properties

Property Name	Type (Range)	Description
owl:sameAs	owl:Thing	
owl:differentFrom	owl:Thing	

owl:AllDifferent properties

Property Name	Type (Range)	Description
owl:distinctMembers	rdf:List	

owl:Restriction properties

Property Name	Type (Range)	Description
owl:allValuesFrom	rdfs:Class	
owl:someValuesFrom	rdfs:Class	
owl:hasValue *	(none)	
owl:cardinality -	xsd:nonNegativeInteger	
owl:maxCardinality -	xsd:nonNegativeInteger	
owl:minCardinality -	xsd:nonNegativeInteger	
owl:onProperty	rdf:Property	

- Denotes a construct available in OWL Lite. * denotes a construct available in OWL DL.

owl:ObjectProperty properties

Property Name	Type (Range)	Description
owl:inverseOf	owl:ObjectProperty	

owl:Ontology properties

Property Name	Type (Range)	Description
owl:imports	owl:Ontology	
owl:incompatibleWith	owl:Ontology	
owl:priorVersion	owl:Ontology	
owl:versionInfo	(none)	
owl:backwardCompatibleWith	owl:Ontology	

xsd Types
Strings
xsd:string
xsd:normalizedString
xsd:token
xsd:language
xsd:NMTOKEN
xsd:Name
xsd:NCName
Boolean
xsd:boolean
Numerical
xsd:decimal
xsd:float

Sponsored by **Readability-Score.com**
Measure your website readability!
https://readability-score.com