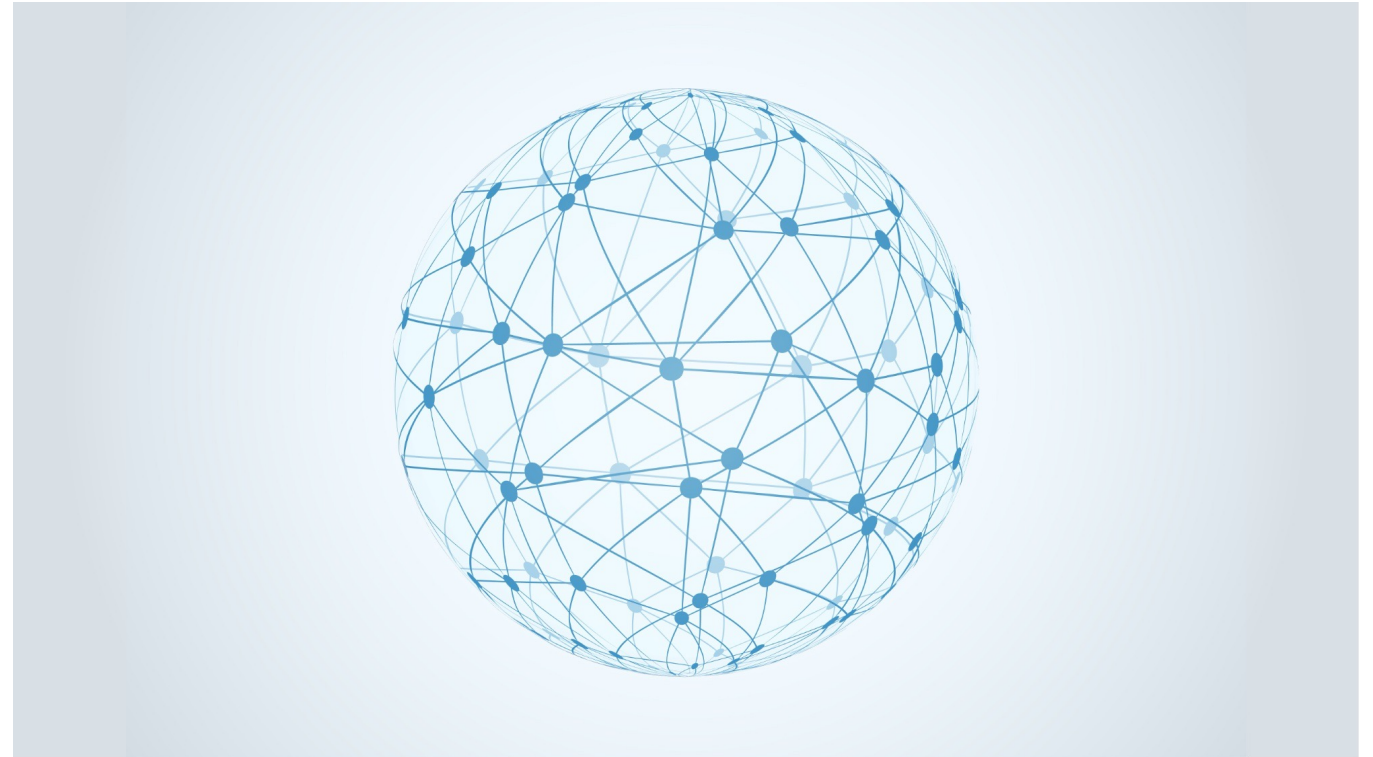


RDF and SPARQL Essentials

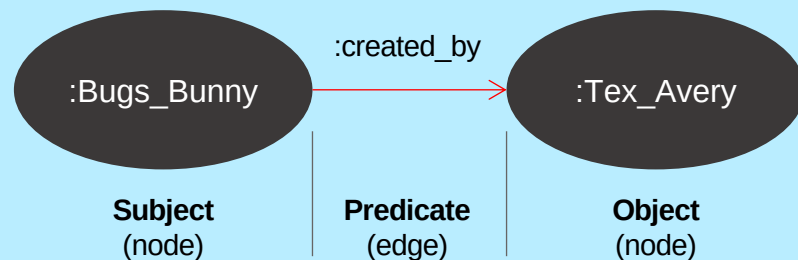
Summary



Building Blocks and Syntax

Triple

Basic building block of an RDF graph. A triple consists of three components: a subject, a predicate and an object



Turtle

A human-friendly and compact RDF serialization format (.ttl file extension)

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX : <http://looneytunes-graph.com/>

# Bugs Bunny
:Bugs_Bunny a :Looney_Tunes_Character ;
  :name "Bugs Bunny" ;
  :species "Hare" ;
  :gender "Male" ;
  :made_debut_appearance_in :A_Wild_Hare ;
  :created_by :Tex_Avery ;
  :personality_trait "Cunning" , "Charismatic" , "Smart" ;
  :known_for_catchphrase "What's up, doc?" .
```

Common RDF Serialization Formats

There are several commonly-used serialization formats for RDF graphs

- N-Quads
- N-Triples
- Notation3
- JSON
- RDF/XML
- TriG
- Turtle

TriG

A human-friendly and compact RDF serialization format for named graphs (.trig file extension)

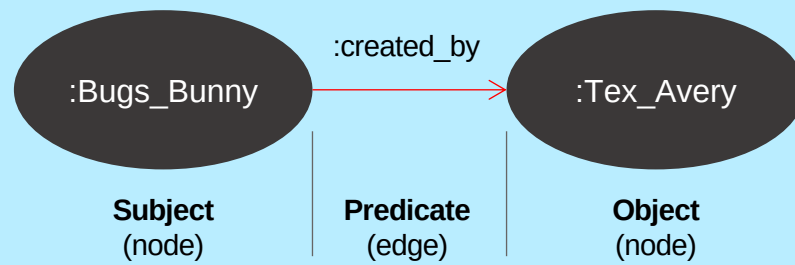
```
PREFIX : <http://looneytunes-graph.com/>
PREFIX mfg: <http://myfavs-graph.com/>

mfg:My_Favourites_Graph {
  :Bugs_Bunny mfg:has_an_appetite_for "Carrots" .
  :Daffy_Duck mfg:has_an_appetite_for "Spaghetti & Meatballs" .
  :Taz mfg:has_an_appetite_for :Bugs_Bunny .
}
```

Building Blocks and Syntax

Triple

Basic building block of an RDF graph. A triple consists of three components: a subject, a predicate and an object



IRI

Unique identifier for nodes and edges

`<http://looneytunes-graph.com/:Bugs_Bunny>` Fully-written IRI

```
PREFIX : <http://looneytunes-graph.com/>
PREFIX mfg: <http://myfavs-graph.com/>
```

Use prefix declarations to abbreviate IRIs
: is the default prefix

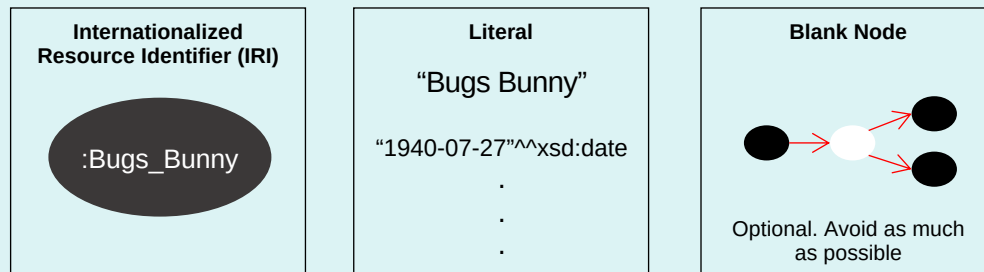
```
:Bugs_Bunny mfg:has_an_appetite_for "Carrots" .
```

```
:Bugs_Bunny a :Looney_Tunes_Character .
```

 Shortcut for `rdf:type`

RDF Nodes

There are three kinds of nodes in RDF: IRI, Literal and Blank Node



Literal

Datatype values

Serialization	Datatype
"Bugs Bunny"	xsd:string
"Bugs Bunny"^^xsd:string	xsd:string
"Bugs Bunny"@en	rdf:langstring
"1940-07-27"^^xsd:date	xsd:date
101	xsd:integer
2.0	xsd:decimal
True	xsd:boolean

Comments

Comments are treated as whitespace

```
# Bugs Bunny
```

 Single line comment

```
# Bugs Bunny
# A Wild Hare
# Tex Avery
# Daffy Duck
# Porky's Duck Hunt
```

 Comments on multiple lines

SPARQL Query Forms

SELECT *Returns all, or a subset of, the variables bound in a query pattern match – results are tabulated*

SELECT * Returns all bound variables

SELECT ?c (COUNT(?pt) AS ?ptCount)

Returns a subset of variables and any variables declared in aggregation functions

SELECT DISTINCT ?t ?p Returns distinct combinations only

ASK *Returns a Boolean indicating whether a query pattern matches or not*

```
ASK {  
  :Bugs_Bunny :created_by :Tex_Avery  
}
```

Returns **true** if pattern matches or **false** if not

These query forms return RDF graphs

CONSTRUCT *Returns an RDF graph constructed by substituting variables in a set of triple templates*

```
CONSTRUCT {  
  ?p :has_age ?age ;  
    :birth_date ?b .  
}  
WHERE {  
  ?p a :Person ;  
    :born_on ?b ;  
    :died_on ?d .  
  
  BIND(year(?b) AS ?bYear)  
  BIND(year(?d) AS ?dYear)  
  BIND((?dYear - ?bYear) AS ?age)  
}
```

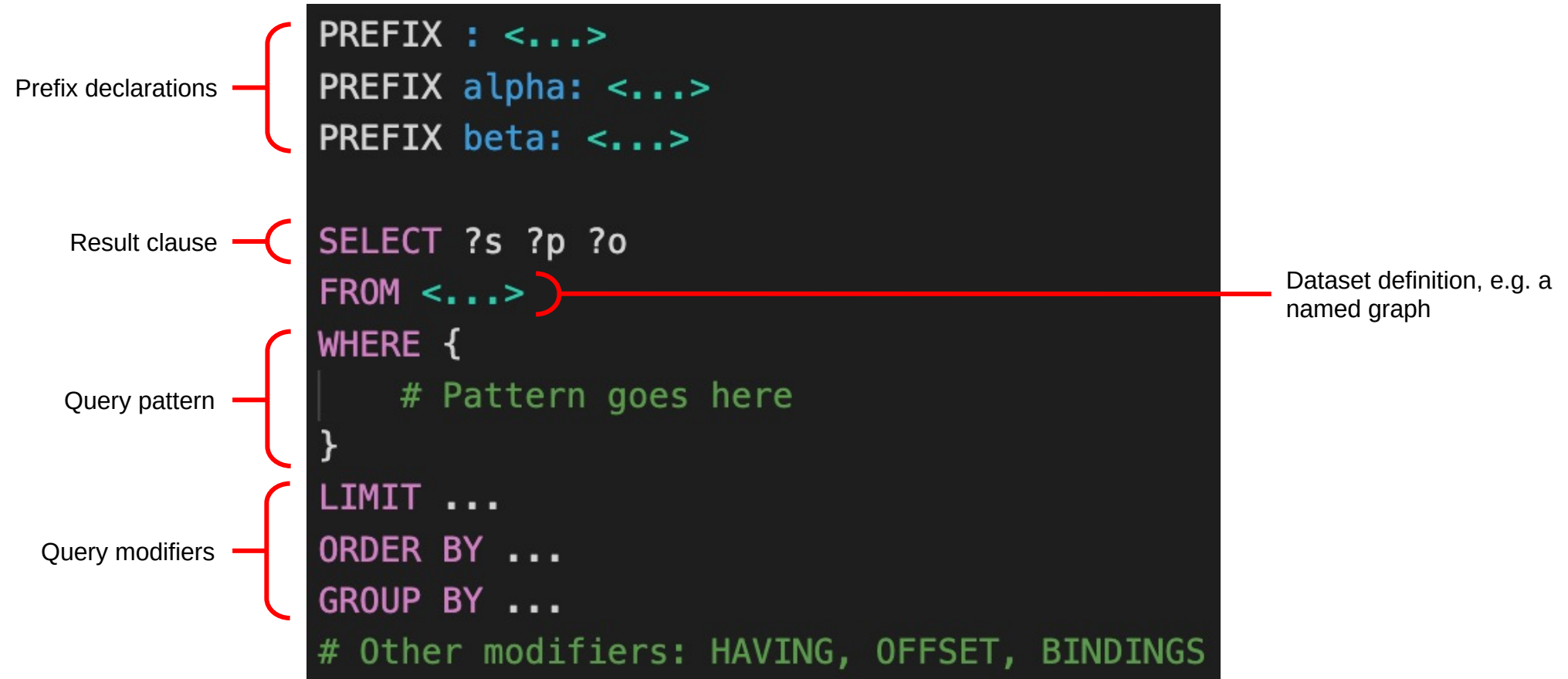
DESCRIBE *Returns an RDF graph that describes the resources found*

DESCRIBE :Bugs_Bunny

Returns all 'one-hop-away' incoming and outgoing triples for the resource

SPARQL Query Structure

Returns all, or a subset of, the variables bound in a query pattern match – results are tabulated



SPARQL Graph Pattern Combinations

X . Y

Basic graph pattern. The values of any common variables from solving X and Y are matched. Results are joined together

```
SELECT ?n
WHERE {
  :Bugs_Bunny :created_by ?p .
  ?p :name ?n
}
```

X OPTIONAL { Y }

Optional pattern matching. Solve X and solve Y. Join the results together. Keep all solutions from X regardless of whether or not there is a matching solution from Y

```
SELECT ?c ?cp
WHERE {
  ?c a :Looney_Tunes_Character .

  OPTIONAL {
    ?c :known_for_catchphrase ?cp
  }
}
```

{ X } UNION { Y }

Alternative graph patterns. Include both the results of solving X and the results of solving Y

```
SELECT ?n
WHERE {
  {
    ?c a :Looney_Tunes_Character ;
      :name ?n .
  }
  UNION
  {
    ?c a :Person ;
      :name ?n .
  }
}
```

X MINUS { Y }

Remove possible solutions. Solve X and solve Y. Find solutions for X that are not compatible with the solutions for Y

```
SELECT ?n
WHERE {
  {
    ?c a :Looney_Tunes_Character ;
      :name ?n .
  }
  MINUS {
    ?c :name "Tasmanian Devil" .
  }
}
```

SPARQL Filters, Functions and Operators

Aggregate Functions

Can only be used in **SELECT** queries for applying a function over a list of values

```
AVG
COUNT
GROUP_CONCAT
MAX
MIN
SAMPLE
SUM
```

Operators

List of common operators which are typically used in **FILTER** expressions

Comparison operators (=, !=, <, <=, >, >=)

Logical operators (&&, ||, !)

Mathematical operators (+, -, /, *)

Existence

Use **EXISTS** or **NOT EXISTS** filter operators to check for the existence or inexistence of a graph pattern, respectively

```
SELECT ?c
WHERE {
  ?c a :Looney_Tunes_Character .

  FILTER NOT EXISTS {
    ?c :known_for_catchphrase ?p
  }
}
```

Functions on RDF Terms

Examples of useful functions that can be applied to test values

```
isIRI
isLiteral
isNumeric
isBlank
```

SPARQL Property Paths

Property paths are possible routes that can be traversed between nodes in a graph

Inverse path

Reverse the direction of the path

```
SELECT ?c ?p
WHERE {
  ?p ^:created_by ?c
}
```

Sequence path

Follow a route in a specified direction of travel along predicates

```
SELECT ?c1 ?c2
WHERE {
  ?c1 :enemy_of/:rival_of ?c2
}
```

Alternative path

Try different path possibilities

```
SELECT ?c1 ?c2
WHERE {
  ?c1 :knows|:rival_of ?c2
}
```

Recursive Path

Property path of arbitrary length

Zero or more path

```
SELECT ?c
WHERE {
  ?c :knows* :Taz
}
```

One or more path

```
SELECT ?c
WHERE {
  ?c :knows+ :Taz
}
```

Zero or one path

```
SELECT ?c
WHERE {
  ?c :knows? :Taz
}
```


SPARQL Update

INSERT DATA

Operation that adds specific triples into the Graph Store

```
INSERT DATA {  
  # Triples go here  
}
```

DELETE DATA

Operation that removes specific triples from the Graph Store

```
DELETE DATA {  
  # Triples go here  
}
```

INSERT

Operation that adds triples into the Graph Store, based on some graph pattern and a specified template

```
INSERT {  
  # Template goes here  
}  
WHERE {  
  # Pattern goes here  
}
```

DELETE

Operation that removes triples from the Graph Store, based on some graph pattern and a specified template

```
DELETE {  
  # Template goes here  
}  
WHERE {  
  # Pattern goes here  
}
```

W3C Prefixes

Prefix	Namespace
rdf:	http://www.w3.org/1999/02/22-rdf-syntax-ns#
xsd:	http://www.w3.org/2001/XMLSchema#
rdfs:	http://www.w3.org/2000/01/rdf-schema#
owl:	http://www.w3.org/2002/07/owl#
skos:	http://www.w3.org/2004/02/skos/core#

Namespace lookup for RDF Developers: <http://prefix.cc/>