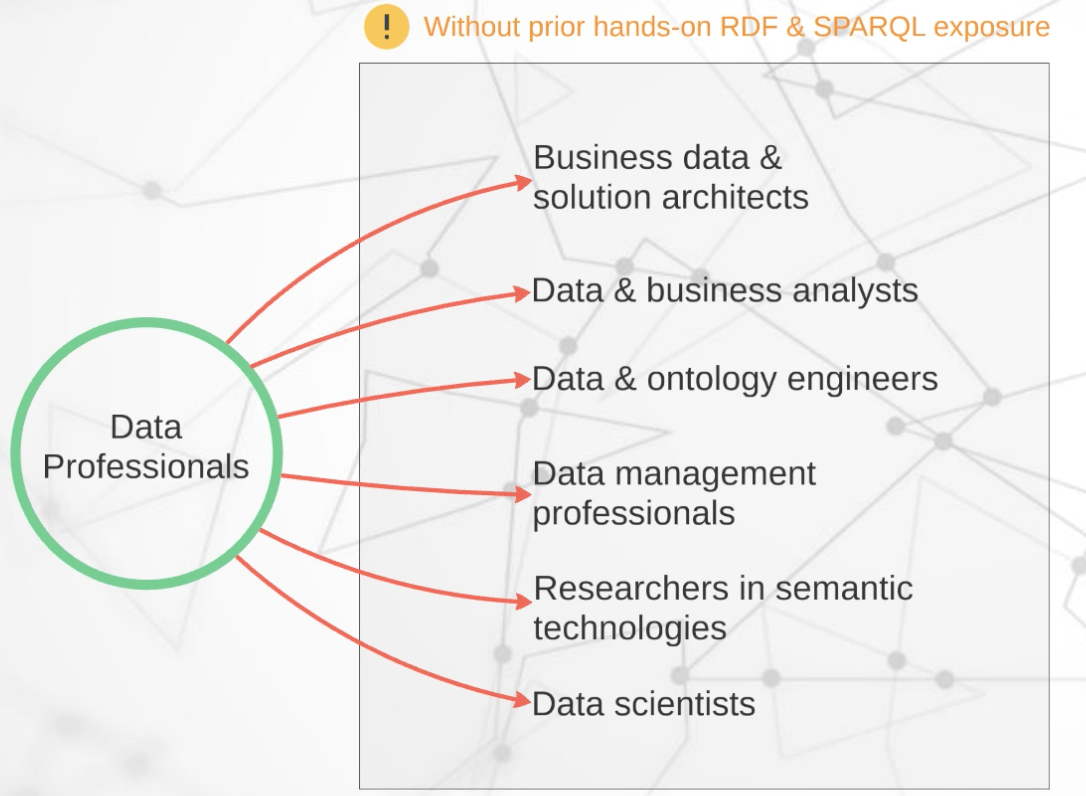
# Udemy: RDF & SPARQL Essentials

RDF: representing data in graphs

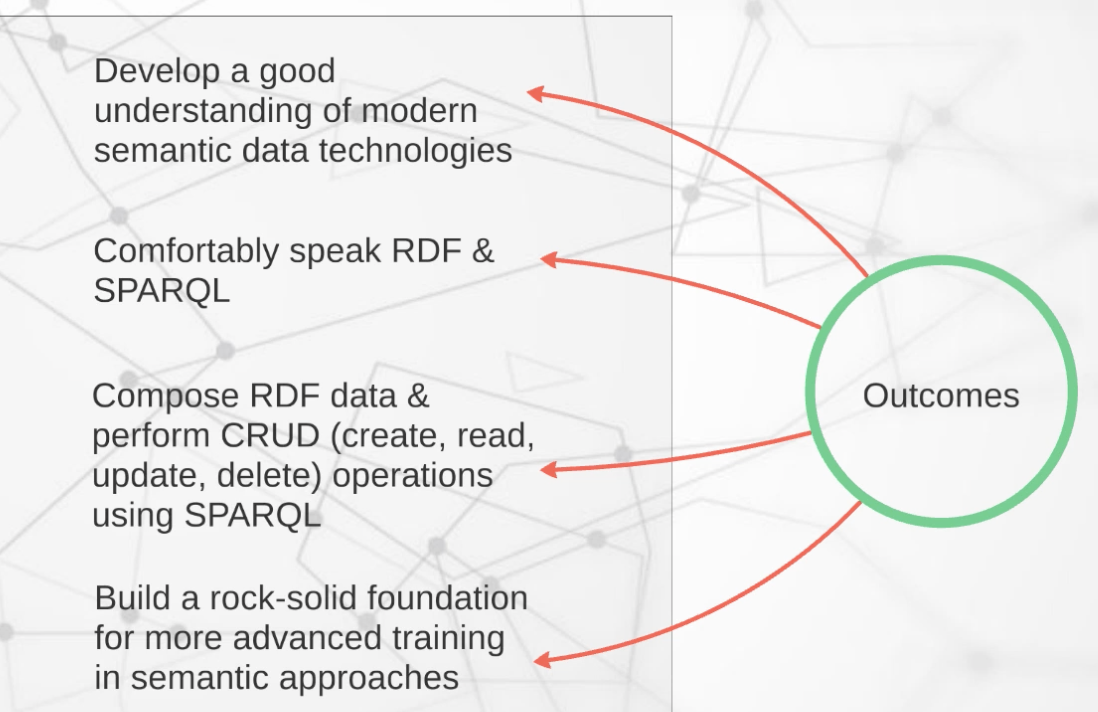
SPARQL: CRUD - creating, reading, updating, and deleting data

Graphs: a network of connected data

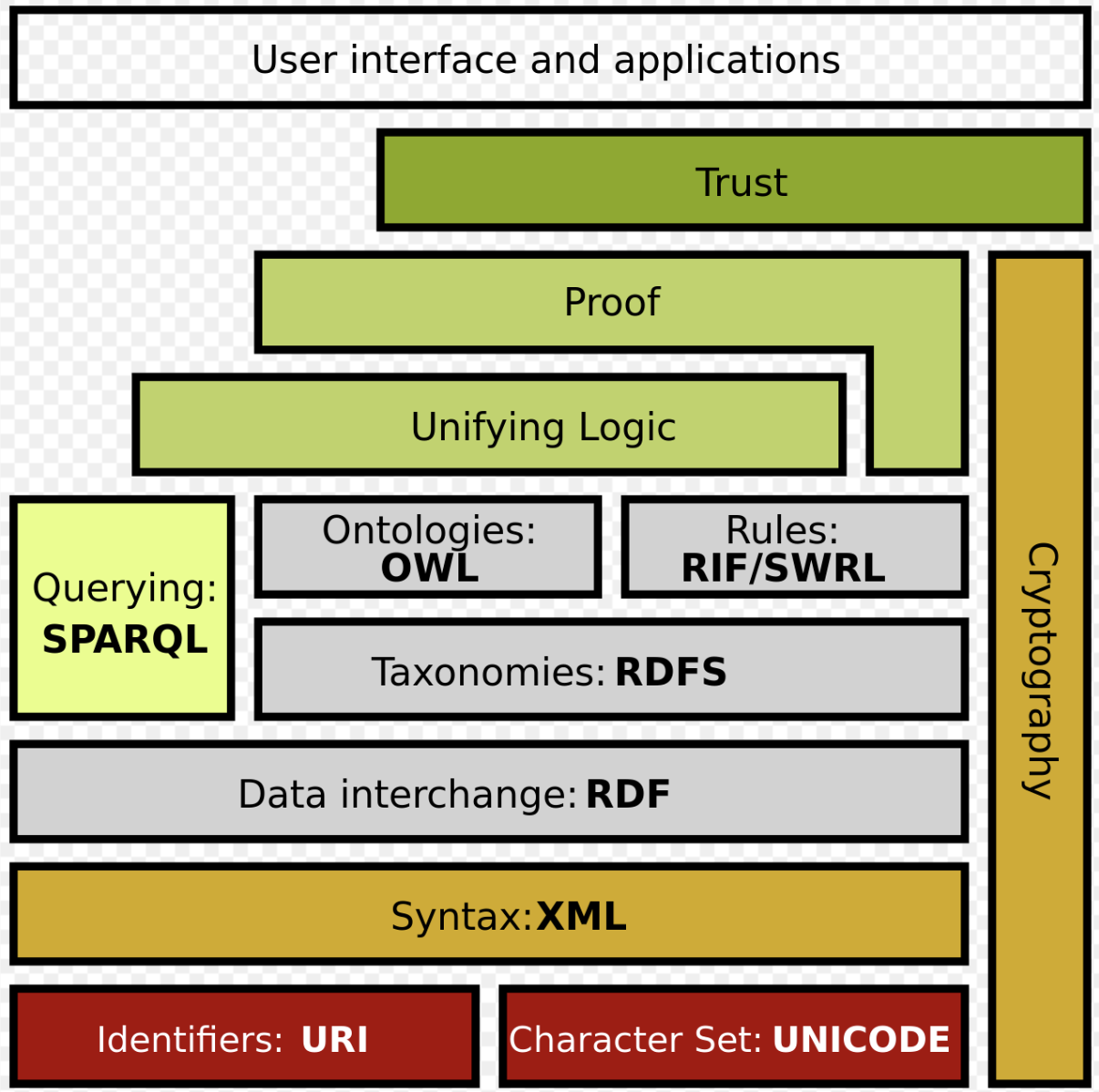
Audience:



Learning objectives:

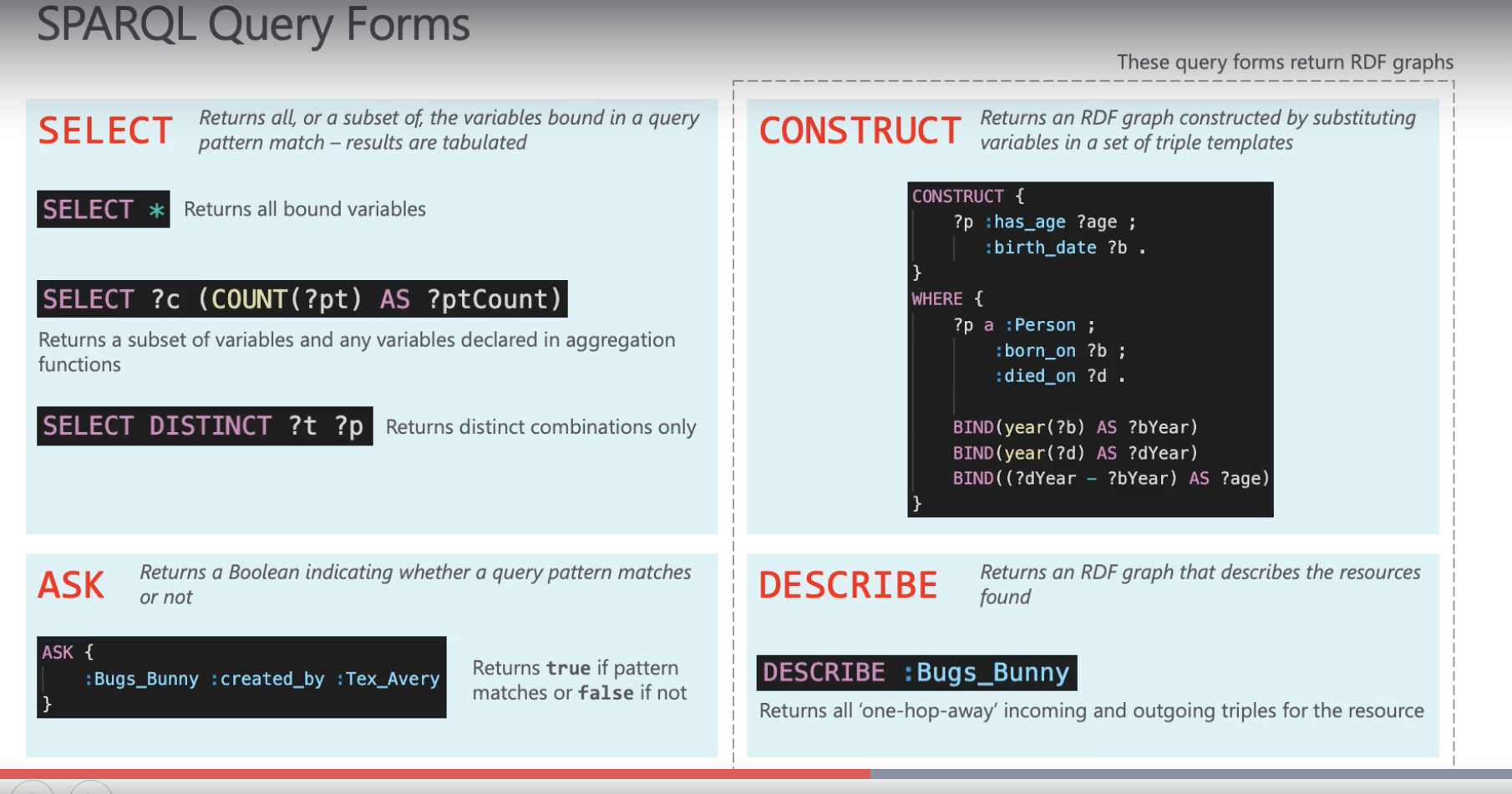


Higher level semantic ontology units: OWL and SWRL



You will learn: queries with SPARQL





Most work: typing RDF data and query with SPARQL language

## RDF update

Download from:

<https://github.com/blazegraph/database/wiki/Quick_Start>

Writing RDF

Sample1:

PREFIX : <http://looneytunes-graph.com/>

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

# 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?" .

# A Wild Hare

:A\_Wild\_Hare a :Short ;

:release\_date "1940-07-27"^^xsd:date .

# Tex Avery

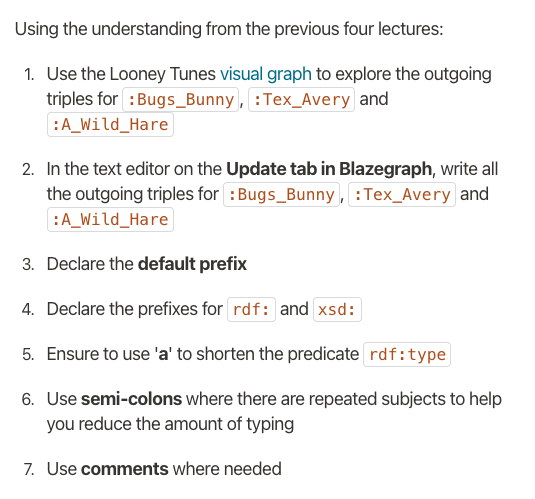
:Tex\_Avery a :Person ;

:name "Frederick Bean Avery" ;

:born\_on "1908-02-26"^^xsd:date ;

:died\_on "1980-08-26"^^xsd:date .

Sample2:



## SPARQL

? precedes variable

**# search subj, pred, and obj**

SELECT ?s ?p ?o

WHERE {

?s ?p ?o

}

LIMIT 20

**# What is Bugs Bunny's name?**

PREFIX : <http://looneytunes-graph.com/>

SELECT ?n

WHERE {

:Bugs\_Bunny :name ?n

}

**# Who is Bugs Bunny's creator?**

PREFIX : <http://looneytunes-graph.com/>

SELECT ?c

WHERE {

:Bugs\_Bunny :created\_by ?c

}

**# What are Bugs Bunny's personality traits?**

PREFIX : <http://looneytunes-graph.com/>

SELECT ?t

WHERE {

:Bugs\_Bunny :personality\_trait ?t

}

# What is/are the name(s) of the creator(s) of Bugs Bunny?

PREFIX : <http://looneytunes-graph.com/>

SELECT ?n

WHERE {

:Bugs\_Bunny :created\_by ?p .

?p :name ?n

}

**# What is the release date when Bugs Bunny made his debut appearance?**

PREFIX : <http://looneytunes-graph.com/>

SELECT ?d

WHERE {

:Bugs\_Bunny :made\_debut\_appearance\_in ?m . ***#the dot signified an end***

?m :release\_date ?d

}

**# What is/are the name(s) of the creator(s) of Bugs Bunny?**

PREFIX : <http://looneytunes-graph.com/>

SELECT ?n

WHERE {

:Bugs\_Bunny :created\_by ?p .

?p :name ?n

}

**# What is the release date when Bugs Bunny made his debut appearance? #compounding expressions**

PREFIX : <http://looneytunes-graph.com/>

SELECT ?d

WHERE {

:Bugs\_Bunny :made\_debut\_appearance\_in ?m .

?m :release\_date ?d

}

**# Birth dates of the creators of Taz**

PREFIX : <http://looneytunes-graph.com/>

SELECT ?d

WHERE {

:Taz :created\_by ?p .

?p :born\_on ?d

}

**# What are the debut release dates for our Looney Tunes characters?**

PREFIX : <http://looneytunes-graph.com/>

SELECT ?n ?d

WHERE {

?c a :Looney\_Tunes\_Character ;

:name ?n ;

:made\_debut\_appearance\_in ?m .

?m :release\_date ?d

}

**# Looney Tunes characters, their creators and their birth dates**

PREFIX : <http://looneytunes-graph.com/>

SELECT ?cn ?pn ?d

WHERE {

?c a :Looney\_Tunes\_Character ;

:name ?cn ;

:created\_by ?p .

?p :name ?pn ;

:born\_on ?d

}

**# How many characters were created by Tex Avery?**

PREFIX : <http://looneytunes-graph.com/>

SELECT (COUNT(?c) AS ?cCount)

WHERE {

?c :created\_by :Tex\_Avery .

}

# Count of number of Looney Tunes characters

PREFIX : <http://looneytunes-graph.com/>

SELECT (COUNT(?c) AS ?cCount)

WHERE {

?c a :Looney\_Tunes\_Character

}

# remove duplicates from the result

**# How many types of movies have our Looney Tunes characters made debut appearances in?**

PREFIX : <http://looneytunes-graph.com/>

SELECT (COUNT(DISTINCT ?mt) AS ?mtCount)

WHERE {

?c :made\_debut\_appearance\_in ?m .

?m a ?mt

}

**# How many types of movies have our Looney Tunes characters made debut appearances in?**

PREFIX : <http://looneytunes-graph.com/>

SELECT (COUNT(DISTINCT ?mt) AS ?mtCount)

WHERE {

?c :made\_debut\_appearance\_in ?m .

?m a ?mt

}

**[Activity] Count of distinct types of creators**

PREFIX : <http://looneytunes-graph.com/>

SELECT (COUNT(DISTINCT ?ptype) AS ?ptypeCount)

WHERE {

?c :created\_by ?p .

?p a ?ptype

}

**# Find Looney Tunes characters that don't have a catchphrase**

PREFIX : <http://looneytunes-graph.com/>

SELECT ?c

WHERE {

?c a :Looney\_Tunes\_Character .

FILTER NOT EXISTS {

?c :known\_for\_catchphrase ?p

}

}

[Activity] Names of Looney Tunes characters with existing catchphrase

PREFIX : <http://looneytunes-graph.com/>

SELECT ?n

WHERE {

?c a :Looney\_Tunes\_Character ;

:name ?n

FILTER EXISTS {

?c :known\_for\_catchphrase ?p

}

}

**# Who are the characters who were co-created? → More than 1 triple involving ‘created\_by’ predicate for which the objects are different**

PREFIX : <http://looneytunes-graph.com/>

SELECT DISTINCT ?c

WHERE {

?c :created\_by ?p1 , ?p2 .

FILTER (?p1 != ?p2)

}

**# What are the earliest and latest release dates of the Shorts?**

PREFIX : <http://looneytunes-graph.com/>

SELECT (MIN(?date) AS ?earliest) (MAX(?date) AS ?latest)

WHERE {

?m a :Short ;

:release\_date ?date #*if no select … this returns all the dates*

}

**# How old was Tex Avery when he died?**

**BIND: stores info of variables**

PREFIX : <http://looneytunes-graph.com/>

SELECT ?age

WHERE {

:Tex\_Avery :born\_on ?b ;

:died\_on ?d .

BIND(year(?b) AS ?bYear)

BIND(year(?d) AS ?dYear)

BIND((?dYear - ?bYear) AS ?age)

}

**[Activity] Lifespan of each Looney Tunes character creator**

PREFIX : <http://looneytunes-graph.com/>

SELECT ?p ?age

WHERE {

?p :born\_on ?b ;

:died\_on ?d

BIND(year(?b) AS ?bYear)

BIND(year(?d) AS ?dYear)

BIND((?dYear - ?bYear) AS ?age)

}

**# Who was the longest living creator?**

PREFIX : <http://looneytunes-graph.com/>

SELECT ?p ?age

WHERE {

?p :born\_on ?b ;

:died\_on ?d .

BIND(year(?b) AS ?bYear)

BIND(year(?d) AS ?dYear)

BIND((?dYear - ?bYear) AS ?age)

}

ORDER BY DESC(?age)

LIMIT 1

**# What is the average age of all Looney Tunes character creators combined?**

PREFIX : <http://looneytunes-graph.com/>

SELECT (AVG(?age) AS ?avgAge)

WHERE {

?p :born\_on ?b ;

:died\_on ?d .

BIND(year(?b) AS ?bYear)

BIND(year(?d) AS ?dYear)

BIND((?dYear - ?bYear) AS ?age)

}

**# Using BIND and nested IF conditions**

PREFIX : <http://looneytunes-graph.com/>

SELECT ?n ?result

WHERE {

?c :made\_debut\_appearance\_in ?m ;

:name ?n .

?m :release\_date ?d .

BIND(year(?d) AS ?dYear)

BIND (

IF(?dYear >= 1946 && ?dYear <= 1950, "Released during post-war era",

IF(?dYear >= 1939 && ?dYear <= 1945, "Released during WW2",

IF(?dYear >= 1918 && ?dYear <= 1938, "Released during interwar period", "Other era"

)

)

) AS ?result

)

}

ORDER BY ASC(?n)

**# Using BIND, COALESCE and IF**

PREFIX : <http://looneytunes-graph.com/>

SELECT ?n ?result

WHERE {

?c :made\_debut\_appearance\_in ?m ;

:name ?n .

?m :release\_date ?d .

BIND(year(?d) AS ?dYear)

BIND(

COALESCE(

IF(?dYear >= 1946 && ?dYear <= 1950, "Released during post-war era", 1/0),

IF(?dYear >= 1939 && ?dYear <= 1945, "Released during WW2", 1/0),

IF(?dYear >= 1918 && ?dYear <= 1938, "Released during interwar period", 1/0),

"Other era"

) AS ?result

)

}

ORDER BY ASC(?n)

PREFIX : <http://looneytunes-graph.com/>

SELECT DISTINCT ?n ?result

WHERE {

?c :created\_by ?p .

?p :born\_on ?b ;

:name ?n

BIND(year(?b) AS ?bYear)

BIND(

COALESCE(

IF(?bYear < 1900, "Born pre-1900", 1/0),

IF(?bYear >= 1900, "Born post-1900", 1/0),

"NA"

) AS ?result

)

}

ORDER BY ASC(?bYear)

PREFIX : <http://looneytunes-graph.com/>

SELECT DISTINCT ?n ?result

WHERE {

?c :created\_by ?p .

?p :born\_on ?b ;

:name ?n

BIND(year(?b) AS ?bYear)

BIND(

COALESCE(

IF(?bYear < 1900, "Born pre-1900", 1/0),

IF(?bYear >= 1900, "Born post-1900", 1/0),

"NA"

) AS ?result

)

}

ORDER BY ASC(?bYear)

**# What are the catchphrases of all the different characters (include those that may not have catchphrases)?**

PREFIX : <http://looneytunes-graph.com/>

SELECT ?c ?cp

WHERE {

?c a :Looney\_Tunes\_Character .

OPTIONAL {

?c :known\_for\_catchphrase ?cp

}

}

**[Activity] Catchphrase mania**

PREFIX : <http://looneytunes-graph.com/>

SELECT ?c ?cp

WHERE {

?c a :Looney\_Tunes\_Character .

OPTIONAL {

?c :known\_for\_catchphrase ?cp

}

}

**# What is the full list of all the names of things in our dataset? (i.e. names of Looney Tunes characters and names of their creators as a single stacked list)**

PREFIX : <http://looneytunes-graph.com/>

SELECT ?n

WHERE {

{

?c a :Looney\_Tunes\_Character ;

:name ?n .

}

UNION

{

?c a :Person ;

:name ?n .

}

}

ORDER BY ASC(?n)

**[Activity] List all the date values present in the graph**

PREFIX : <http://looneytunes-graph.com/>

SELECT DISTINCT ?d

WHERE {

{

?m :release\_date ?d .

}

UNION

{

?p :born\_on ?d .

}

UNION

{

?p :died\_on ?d .

}

}

ORDER BY DESC(?d)

**# Pull a list of all Looney Tunes character names making sure to remove "Tasmanian Devil" from the list - use of MINUS**

PREFIX : <http://looneytunes-graph.com/>

SELECT ?n

WHERE {

{

?c a :Looney\_Tunes\_Character ;

:name ?n .

}

MINUS {

?c :name "Tasmanian Devil" .

}

}

ORDER BY ASC(?n)

**# What is the count of recorded personality traits for each of our Looney Tunes Characters?**

PREFIX : <http://looneytunes-graph.com/>

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

WHERE {

?c :personality\_trait ?pt

}

GROUP BY ?c

**[Activity] Count of creators per Looney Tunes character**

PREFIX : <http://looneytunes-graph.com/>

SELECT ?n (COUNT(?p) AS ?pCount)

WHERE {

?c :created\_by ?p ;

:name ?n

}

GROUP BY ?n

**# DESCRIBE query - What are the incoming and outgoing statements for Bugs Bunny?**

**It returns a subportion of the graph**

PREFIX : <http://looneytunes-graph.com/>

DESCRIBE :Bugs\_Bunny

**[Activity] Outgoing and incoming RDF triples for Sylvester**

PREFIX : <http://looneytunes-graph.com/>

DESCRIBE :Sylvester

# ASK query - It is true that Bugs Bunny was created by Text Avery? → same as BOOL

PREFIX : <http://looneytunes-graph.com/>

ASK {

:Bugs\_Bunny :created\_by :Tex\_Avery

}

PREFIX : <http://looneytunes-graph.com/>

ASK {

:Sylvester :created\_by ?p .

?p :name "Isadore Frelang"

# There is a typo in the name string above. Try "Isadore Freleng"

}

**# CONSTRUCT query - Return an RDF sub-graph which contains specific triples patterns (persons and their birth and death dates)**

PREFIX : <http://looneytunes-graph.com/>

CONSTRUCT WHERE {

?p a :Person ;

:born\_on ?b ;

:died\_on ?d .

}

More:

**# CONSTRUCT query - Return an RDF sub-graph which contains specific triples patterns (persons and their average age)**

PREFIX : <http://looneytunes-graph.com/>

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)

}

**[Activity] Is "Isadore Frelang" Sylvester's creator's name?**

PREFIX : <http://looneytunes-graph.com/>

CONSTRUCT {

?c :debuted\_in ?d

}

WHERE {

?c a :Looney\_Tunes\_Character ;

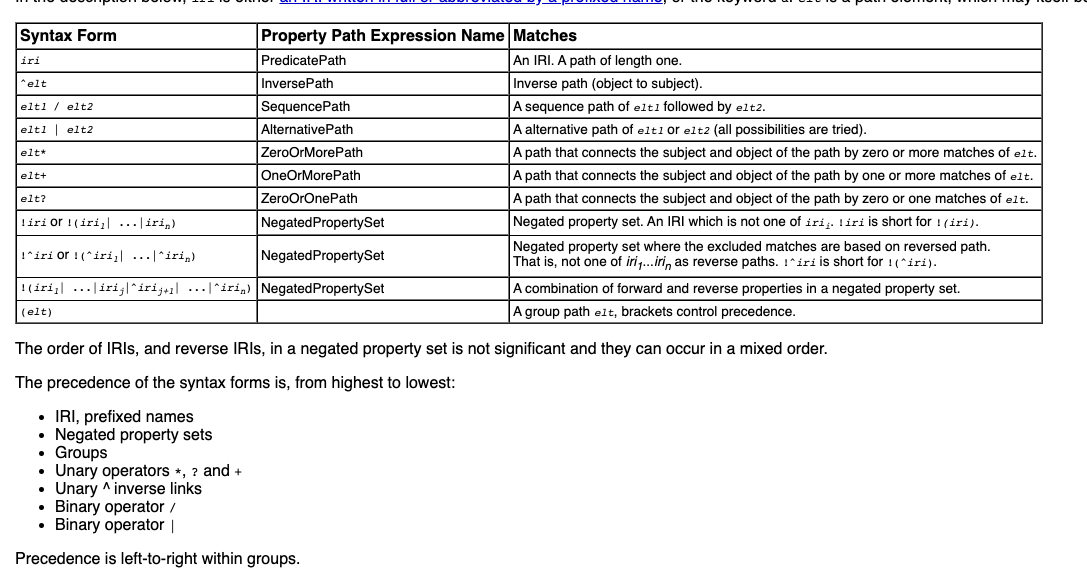
:made\_debut\_appearance\_in ?m .

?m :release\_date ?d

}

## Property paths

<https://www.w3.org/TR/sparql11-query/#propertypaths>



Inverse path

## Inverse path

PREFIX : <http://looneytunes-graph.com/>

SELECT ?c ?p

WHERE {

?p ^:created\_by ?c

}

^ means flip:



Sequence path

## Sequence path - **List characters who are related via a path that involves the predicates**

For example: list characters who are enemies of other characters who themselves are rivals

:enemy\_of followed by :rival\_of → /

PREFIX : <http://looneytunes-graph.com/>

SELECT ?c1 ?c2

WHERE {

?c1 :enemy\_of/:rival\_of ?c2

}

PREFIX : <http://looneytunes-graph.com/>

SELECT ?c1n ?c2n

WHERE {

?c1 ^:rival\_of/^:enemy\_of ?c2 ;

:name ?c1n .

?c2 :name ?c2n

}

Recursive path

## Recursive paths - Who knows Taz (directly or transitively) through zero or more occurrences of "knows"?

PREFIX : <http://looneytunes-graph.com/>

SELECT ?c

WHERE {

?c :knows\* :Taz

}

All possible forward path

## What are the possible paths I can take between two nodes (e.g. between Bugs Bunny & Daffy Duck)?

PREFIX : <http://looneytunes-graph.com/>

SELECT DISTINCT ?subject ?predicate ?object

WHERE {

VALUES ?start { :Bugs\_Bunny }

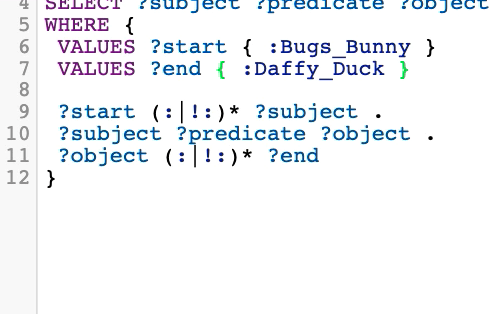
VALUES ?end { :Daffy\_Duck }

?start (:|!:)\* ?subject .

?subject ?predicate ?object .

?object (:|!:)\* ?end

}



PREFIX : <http://looneytunes-graph.com/>

SELECT DISTINCT ?subject ?predicate ?object

WHERE {

VALUES ?start { :Sylvester }

VALUES ?end { :Taz }

?start (:|!:)\* ?subject .

?subject ?predicate ?object .

?object (:|!:)\* ?end

}

## SPARQL updates: delete, insert,

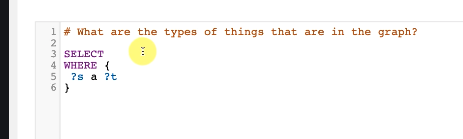
## 

## SPARQL: Name a graph

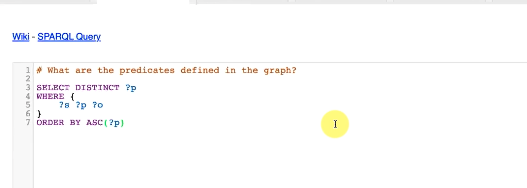
TRIG format

## Explore the schema (basic info)

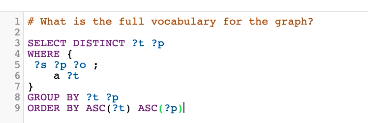
Check the hierarchy/type knowledge



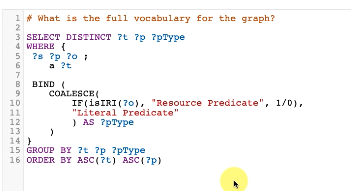
Check the predicates



Check the full vocab



Con’t: tell the difference between resource and literal predicate



Materials:

Concepts and Abstract syntax

<https://www.w3.org/TR/rdf11-concepts/>

Turtle

<https://www.w3.org/TR/turtle/>

Trig

<https://www.w3.org/TR/trig/>

Query language

<https://www.w3.org/TR/sparql11-query/>

Update

<https://www.w3.org/TR/sparql11-update/>

Property paths

<https://www.w3.org/TR/sparql11-query/#propertypaths>

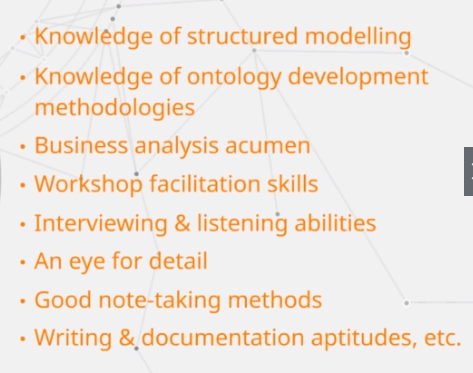
Datatypes

<https://www.w3.org/TR/xmlschema-2/#built-in-datatypes>

Web ontology language

<https://en.wikipedia.org/wiki/Web_Ontology_Language>

skills:



Challenge:

1. Pick up a project at a manageable scale
2. Conduct information gathering workshops
3. Model your ontology
4. Apply the ontology for analysis purposes (for example making queries)