



SPARQL

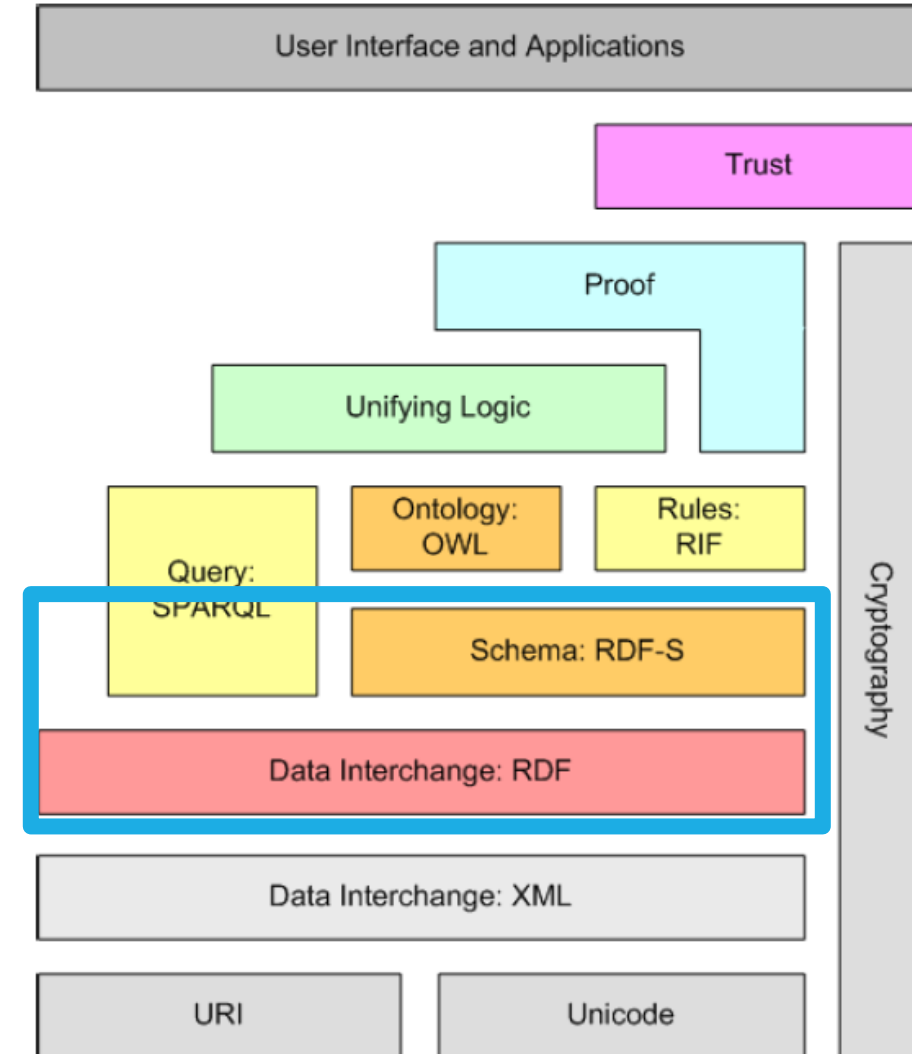
Σημασιολογικός Ιστός και Ευφυείς Εφαρμογές – 7^ο Εξάμηνο
Γεώργιος Μεδίτσκος, Επίκουρος Καθηγητής, Τμήμα Πληροφορικής ΑΠΘ

Αρχιτεκτονική ΣΙ

■ RDF Schema

- Είναι μια στοιχειώδης γλώσσα οντολογιών
- Παρέχει συγκεκριμένα θεμελιώδη στοιχεία μοντελοποίησης
 - κλάσεις, σχέσεις υποκλάσης, οι ιδιότητες, οι σχέσεις υποϊδιότητας, και οι περιορισμοί στο πεδίο ορισμού και στο σύνολο τιμών
- Στην ουσία περιγράφει το λεξιλόγιο (schema) και τις συσχετίσεις ανάμεσα στις διάφορες έννοιες

```
:AllStarPlayer rdf:type rdfs:Class.  
:MajorLeaguePlayer rdf:type rdfs:Class.  
:Surgeon rdf:type rdfs:Class.  
:Staff rdf:type rdfs:Class.  
:Physician rdf:type rdfs:Class.
```

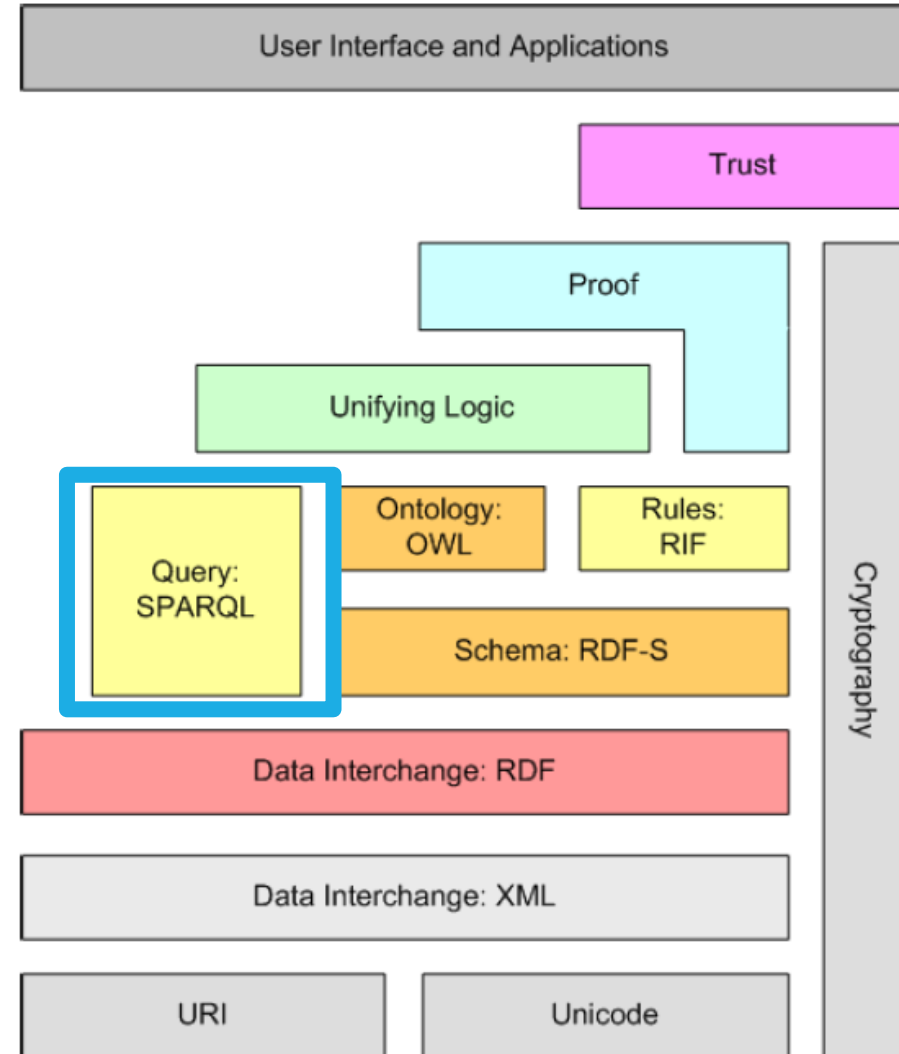


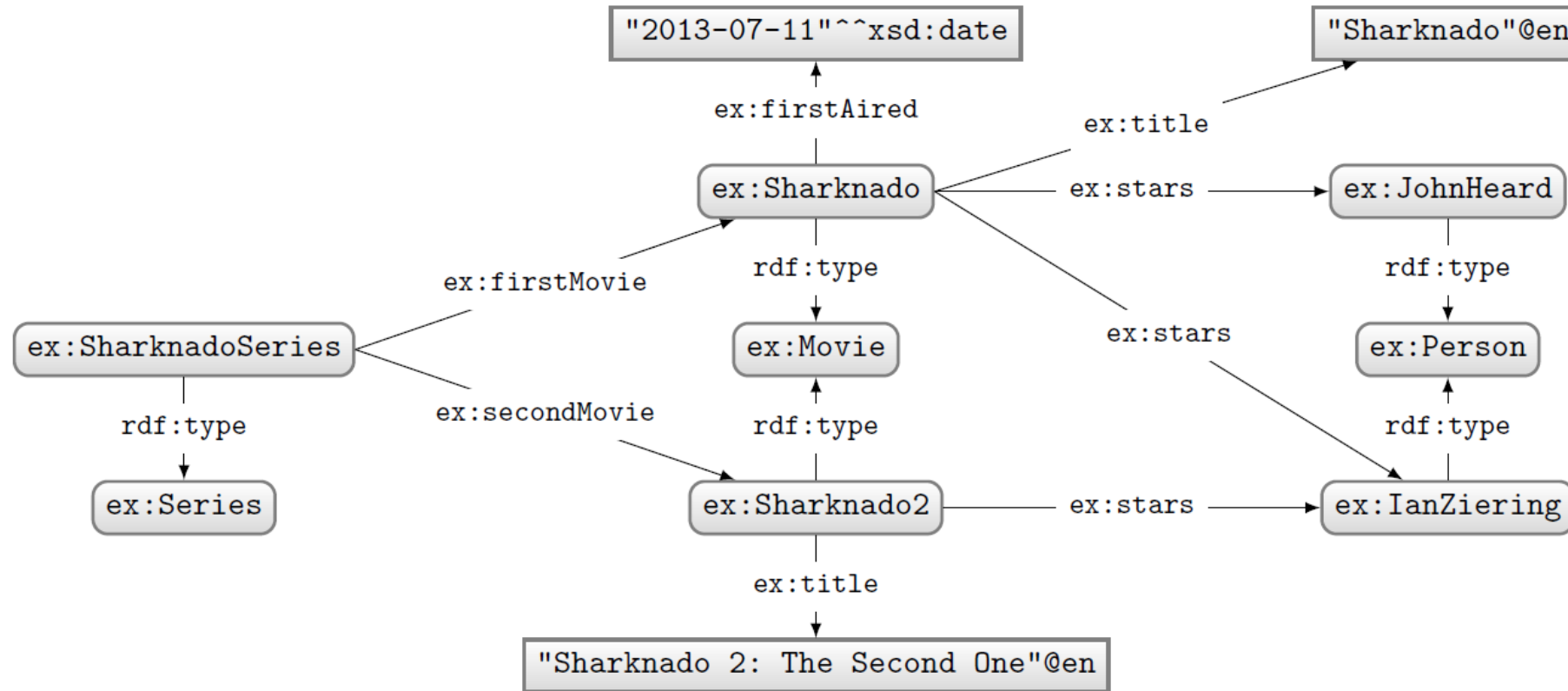
Αρχιτεκτονική ΣΙ

■ SPARQL

- Γλώσσα ερωτημάτων για την RDF
- select, construct, ask, describe

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name
       ?email
WHERE
{
  ?person a      foaf:Person .
  ?person foaf:name ?name .
  ?person foaf:mbox ?email .
}
```





“Who stars in ‘Sharknado’?”

```

PREFIX ex: <http://ex.org/voc#>
SELECT *
WHERE {
    ex:Sharknado ex:stars ?star .
}
  
```

?star

ex:JohnHeard
ex:IanZiering

SPARQL

- SPARQL Protocol and RDF Query Language
- Γλώσσα ερωτημάτων σε RDF γράφους
- Τα ερωτήματα διατυπώνονται σε μια παραλλαγή της Turtle
- SPARQL 1.0, SPARQL 1.1



SPARQL 1.1 Overview

W3C Recommendation 21 March 2013

This version:

<http://www.w3.org/TR/2013/REC-sparql11-overview-20130321/>

Latest version:

<http://www.w3.org/TR/sparql11-overview/>

Previous version:

<http://www.w3.org/TR/2012/PR-sparql11-overview-20121108/>

Editor:

The W3C SPARQL Working Group, see [Acknowledgements](#) <public-rdf-dawg-comments@w3.org>

Please refer to the [errata](#) for this document, which may include some normative corrections.

See also [translations](#).

Copyright © 2013 W3C® (MIT, ERCIM, Keio, Beihang), All Rights Reserved. W3C [liability](#), [trademark](#) and [document use](#) rules apply.

Abstract

This document is an overview of SPARQL 1.1. It provides an introduction to a set of W3C specifications that

Status of this Document

May Be Superseded

This section describes the status of this document at the time of its publication. Other documents may supersede it at <http://www.w3.org/TR/>.

Set of Documents

Ανατομία του ερωτήματος SPARQL

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX msc: <http://www.myExample.gr/dataset#>
```

Ορισμός Προθεμάτων

Τύπος Ερωτήματος

```
SELECT ?title
```

Μεταβλητές (τί θέλουμε να βρούμε)

```
WHERE
```

```
{
```

```
  ?x rdf:type msc:SomeClass .
```

```
  ?dataset rdf:title ?title .
```

```
  FILTER(...)
```

```
}
```

```
Group By ...
```

```
Having ...
```

```
Order By ...
```

```
Limit ...
```

```
Offset ...
```

```
Bindings ...
```

Πρότυπος γράφος (οι πρότυπες
τριπλέτες που πρέπει να
ικανοποιηθούν) και φίλτρα στις τιμές

Προαιρετικοί τελεστές στα
αποτελέσματα

Τύποι ερωτημάτων SPARQL

- **SELECT**

Επιστρέφει σε πίνακα τα αποτελέσματα X, Y, κλπ. που ικανοποιούν τον πρότυπο γράφο

- **CONSTRUCT**

Εντοπίζει τα X, Y, κλπ. που ικανοποιούν τον πρότυπο γράφο και τα τοποθετεί σε νέο πρότυπο γράφο για την κατασκευή RDF τριπλετών (γράφου).

- **DESCRIBE**

Εντοπίζει τριπλέτες που παρέχουν πληροφορία για συγκεκριμένους πόρους

- **ASK**

Εξετάζει αν υπάρχουν X, Y, κλπ. στα δεδομένα, τέτοια ώστε να ικανοποιούν τον πρότυπο γράφο. Επιστρέφει yes ή no

Σύνταξη Ερωτήματος: **SELECT**

- Έχει δύο μέρη
 - Μια σειρά από ερωτηματικές λέξεις (π.χ. ?x)
 - Και ένα πρότυπο γράφου ερώτησης - **WHERE** (query graph pattern)

```
PREFIX ex: <http://ex.org/voc#>
SELECT *
WHERE {
    ex:Sharknado ex:stars ?star .
}
```

```
SELECT ?what WHERE { :JamesDean :playedIn ?what . }
```

```
SELECT ?who WHERE { ?who :playedIn :Giant . }
```

```
SELECT ?what WHERE { :JamesDean ?what :Giant . }
```


Πολλαπλές τιμές

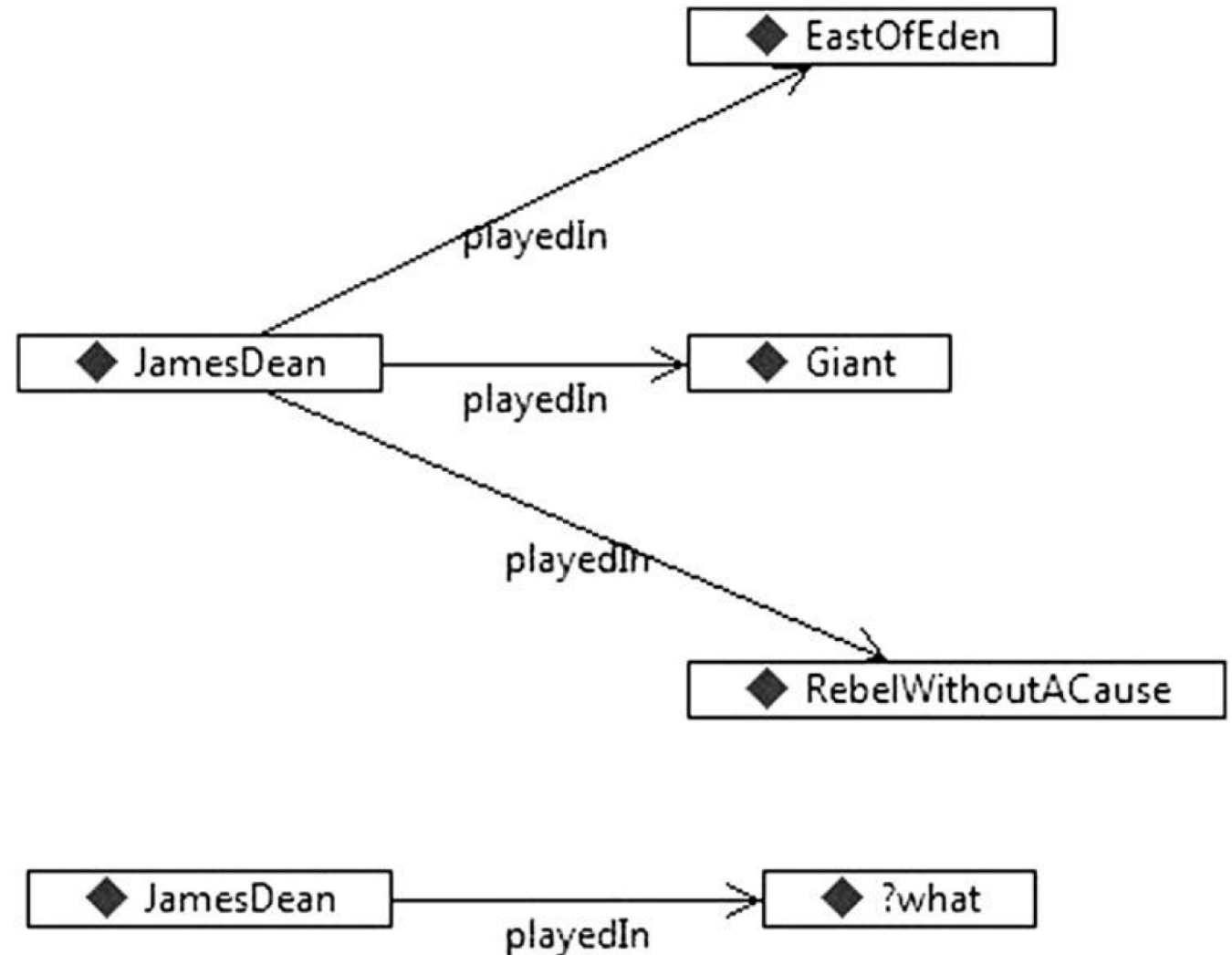
```
:JamesDean :playedIn :Giant .  
:JamesDean :playedIn :EastOfEden .  
:JamesDean :playedIn :RebelWithoutaCause .
```

Ερώτηση: `SELECT ?what WHERE { :JamesDean :playedIn ?what }`

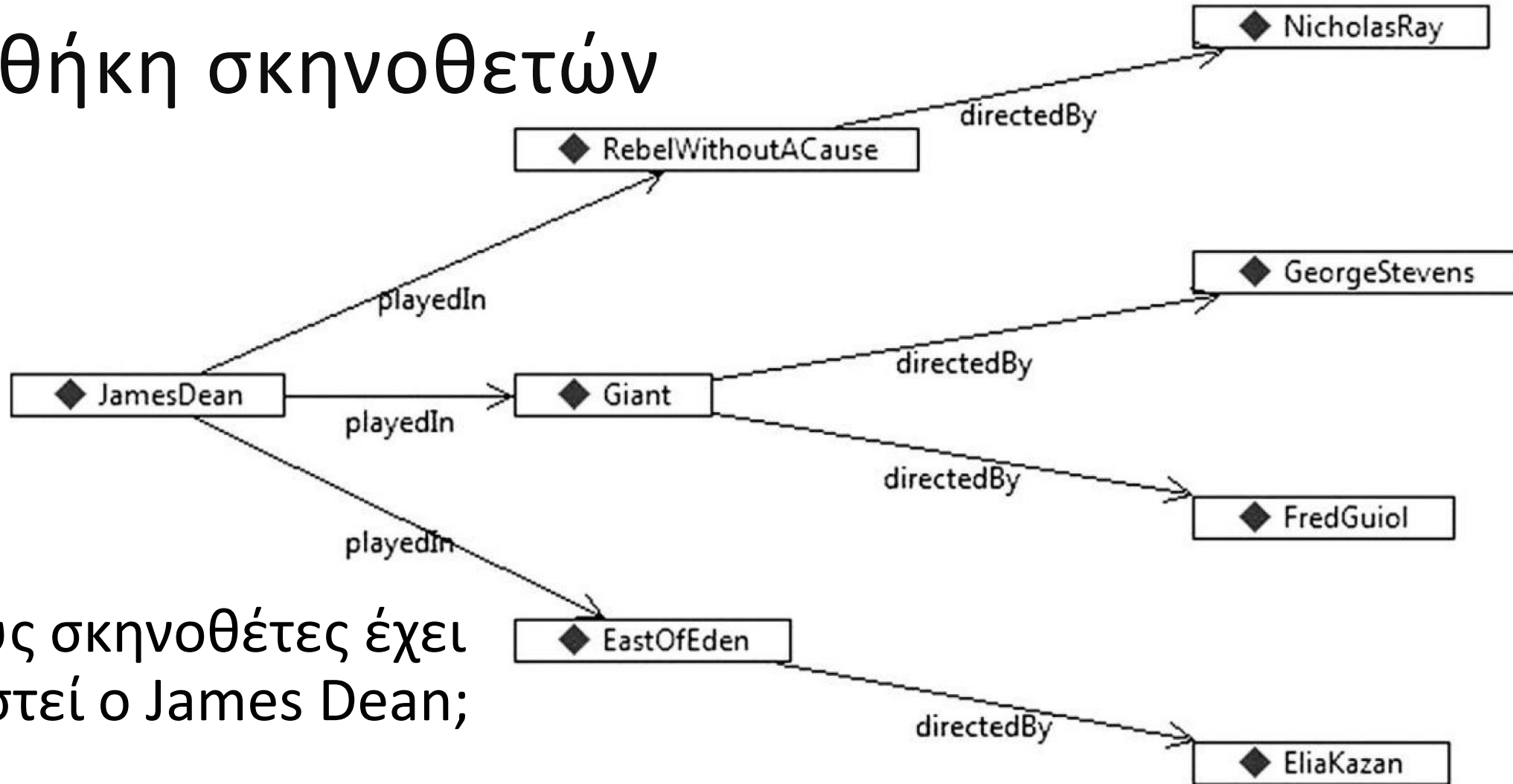
Απάντηση: `:Giant, :EastOfEden, :RebelWithoutaCause.`

Μορφότυπο Γράφου **WHERE** (Graph Pattern)

- Στην ουσία το WHERE αντιστοιχεί σε ένα πρότυπο γράφο δεδομένων
- Δουλειά του συστήματος εκτέλεσης ερωτημάτων είναι να «ταιριάξει» το πρότυπο με τον γράφο δεδομένων



Προσθήκη σκηνοθετών



- Με ποιους σκηνοθέτες έχει συνεργαστεί ο James Dean;

:JamesDean :playedIn ?what .

?what :directedBy ?who .

Προσθήκη σκηνοθετών

?what=:Giant

?what=:Giant

?what=:EastOfEden

?what=:RebelWithoutaCause

?who=:GeorgeStevens

?who=:FredGuiol

?who=:EliaKazan

?who=:NicholasRay

- Επειδή έχουμε >1 ερωτηματικές λέξεις (μεταβλητές), επιστρέφονται όλες οι αναθέσεις
- Στο SELECT μπορούμε να δηλώσουμε ποιες θέλουμε

Ερώτηση:

```
SELECT ?who
WHERE { :JamesDean :playedIn ?what .
        ?what :directedBy ?who . }
```

Απάντηση:

```
:GeorgeStevens, :EliaKazan, :NicholasRay,
:FredGuiol
```

Ορισμός Ερωτηματικών Λέξεων

- Δηλώνονται με ?
- Μπορούμε να χρησιμοποιήσου με όποια λέξη θέλουμε

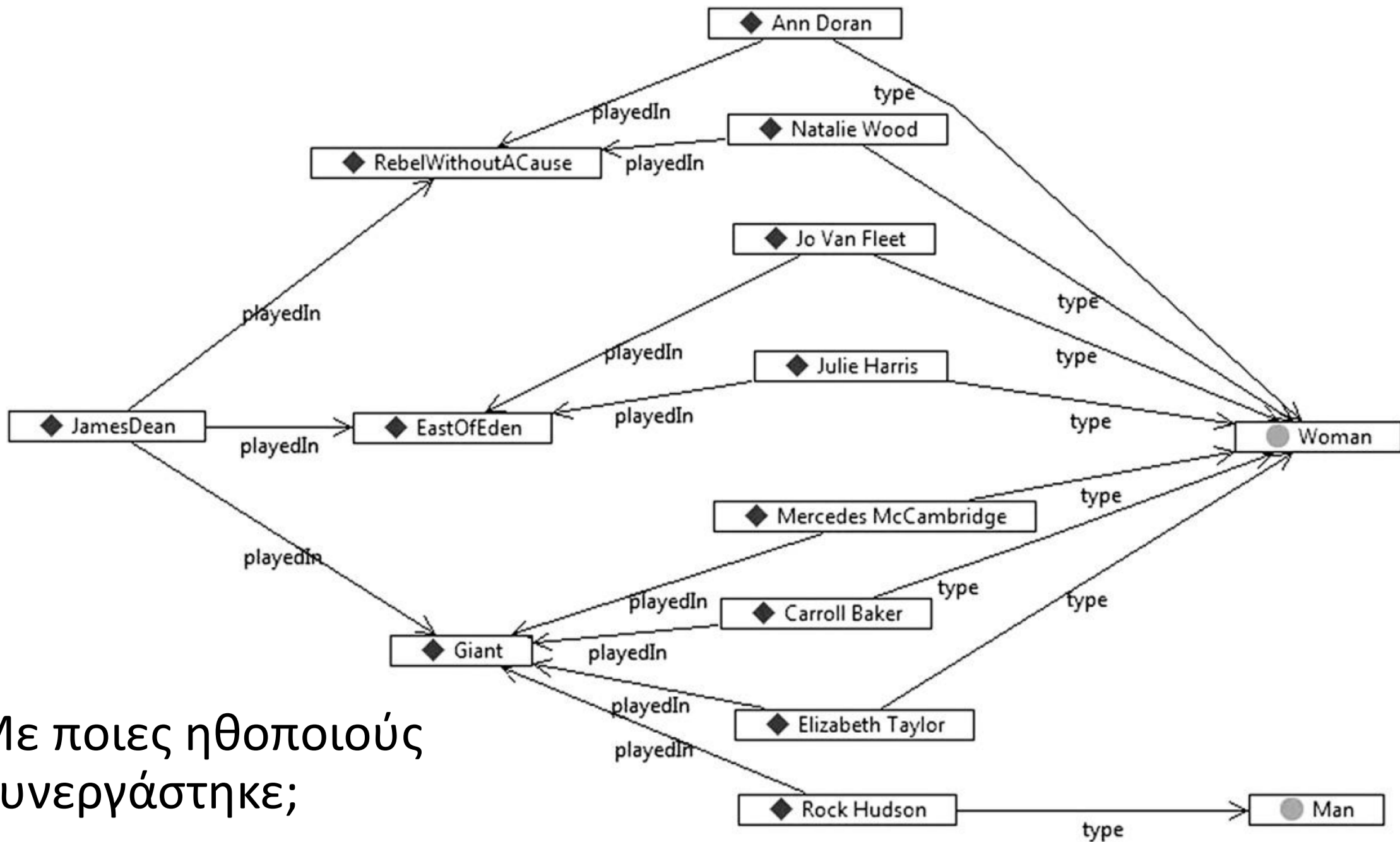
Ask:

```
SELECT ?movie ?director
WHERE { :JamesDean :playedIn ?movie .
        ?movie :directedBy ?director . }
```

Answer:

| ?movie | ?director |
|---------------------|----------------|
| :Giant | :GeorgeStevens |
| :Giant | :FredGuiol |
| :EastOfEden | :EliaKazan |
| :RebelWithoutaCause | :NicholasRay |

Προσθήκη ηθοποιών



Με ποιες ηθοποιούς
συνεργάστηκε;

Προσθήκη ηθοποιών

```
SELECT ?actress ?movie
WHERE {
    :JamesDean :playedIn ?movie .
    ?actress :playedIn ?movie .
    ?actress rdf:type :Woman .
}
```

?actress

- :AnnDoran
- :ElizabethTaylor
- :CarrollBaker
- :JoVanFleet
- :JulieHarris
- :MercedesMcCambridge
- :NatalieWood

- Ο Rock Hudson δεν επιστρέφεται γιατί δεν ικανοποιεί το πρότυπο τριπλέτας: ?actress rdf:type :Woman

Προσθήκη ηθοποιών

- Αν στο παράδειγμά μας το playedIn μπορούσε να αναφερθεί και σε άλλους τύπους εκτός από movies:

```
SELECT ?actress
WHERE {
    :JamesDean :playedIn ?movie .
    ?movie rdf:type :Movie .
    ?actress :playedIn ?movie .
    ?actress rdf:type :Woman
}
```

- Το όνομα της μεταβλητής δεν παίζει κανένα απολύτως ρόλο (δεν έχει σημασιολογία)

Σειρά Τριπλετών σε SPARQL

- Δεν παίζει ρόλο η σειρά των τριπλετών στην RDF
- Το ίδιο ισχύει και στην SPARQL για τα πρότυπα γράφου
 - Θα επιστραφούν τα ίδια αποτελέσματα
- Όμως μπορεί να υπάρξουν άλλες συνέπειες, π.χ. χρόνος εκτέλεσης



- Το ?q1 αντιστοιχίζεται σε 3 ταινίες
- Αν κάθε ταινία έχει N ηθοποιούς, τότε η δεύτερη τριπλέτα θα αντιστοιχίζεται σε $N \times 3$ ηθοποιούς



```
SELECT ?q3
WHERE {
    :JamesDean :playedIn ?q1 .
    ?q3 :playedIn ?q1 .
    ?q3 :playedIn ?q2 .
    ?q2 :directedBy :JohnFord .
}
```

- Η πρώτη τριπλέτα θα αντιστοιχισθεί σε όλα τα ζεύγη τιμών για τα ?q3 και ?q1 ($N \times M$)
- Η δεύτερη τριπλέτα θα «κόψει» ζεύγη

```
SELECT ?q3
WHERE {
    ?q3 :playedIn ?q1 .
    :JamesDean :playedIn ?q1 .
    ?q3 :playedIn ?q2 .
    ?q2 :directedBy :John Ford.
}
```

Αναζήτηση Ιδιοτήτων και Σχημάτων (Schemas)

- Ερωτηματικές λέξεις μπορούν να μούνε σε οποιοδήποτε μέρος της τριπλέτας: subject, predicate, object

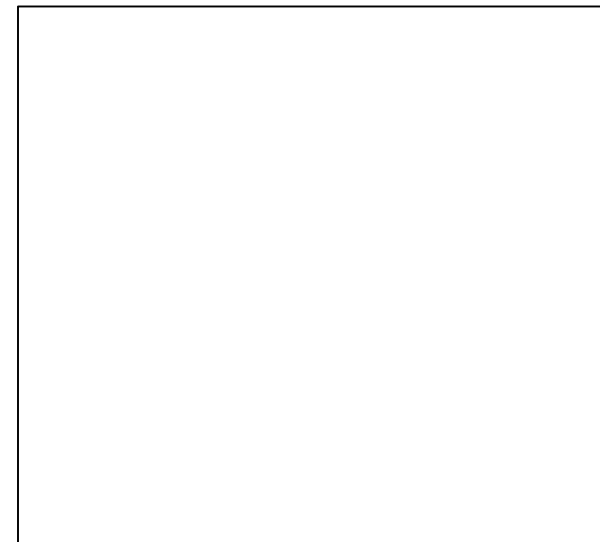
```
SELECT ?property ?value  
WHERE { :JamesDean ?property ?value }
```

| ?property | ?value |
|------------|---------------------|
| :bornOn | 1931-02-08 |
| :diedOn | 1955-09-30 |
| :playedIn | :RebelWithoutaCause |
| :playedIn | :EastOfEden |
| :playedIn | :Giant |
| rdf:type | :Man |
| rdfs:label | "James Dean" |



Αναζήτηση Ιδιοτήτων και Σχημάτων

```
SELECT ?property  
WHERE { :JamesDean ?property ?value }
```





Αναζήτηση Ιδιοτήτων και Σχημάτων

```
SELECT ?property  
WHERE { :JamesDean ?property ?value }
```

```
?property  
:bornOn  
:diedOn  
:playedIn  
:playedIn  
:playedIn  
rdf:type  
rdfs:label
```



Αναζήτηση Ιδιοτήτων και Σχημάτων

```
SELECT DISTINCT ?property  
WHERE { :JamesDean ?property ?value }
```

■ DISTINCT

- Φιλτράρισμα διπλών αποτελεσμάτων

```
?property  
:bornOn  
:diedOn  
:playedIn  
rdf:type  
rdfs:label
```

Αναζήτηση Ιδιοτήτων και Σχημάτων

```
SELECT DISTINCT ?property
WHERE {
    ?q0 a :Actor .
    ?q0 ?property ?object .
}
```

```
?property
:bornOn
:diedOn
:playedIn
rdf:type
rdfs:label
:produced
:sang
:wrote
```

- Μπορεί να «υπάρχουν» και άλλες ιδιότητες οι οποίες δεν έχουν χρησιμοποιηθεί ακόμα, οπότε δεν θα επιστραφούν

Αναζήτηση Ιδιοτήτων και Σχημάτων

```
SELECT DISTINCT ?class  
WHERE {?class rdfs:subClassOf :Person}
```

```
?class  
:Actor  
:Actress  
:Man  
:Woman  
:Politician  
:Producer
```

Αναζήτηση Ιδιοτήτων και Σχημάτων

```
SELECT DISTINCT ?class  
WHERE {?q0 a ?class}
```

```
SELECT DISTINCT ?property  
WHERE {?q0 ?property ?q1}
```

- Όλες οι κλάσεις και οι ιδιότητες που χρησιμοποιούνται



- Ποιοι ηθοποιοί που έπαιξαν στο Giant έζησαν πάνω από πέντε χρόνια μετά το γύρισμα της ταινίας (24/11/1956);

```
SELECT ?actor ?deathdate
WHERE {
    ?actor :playedIn :Giant ;
           :diedOn ?deathdate .
}
```

| ?actor | ?deathdate |
|---------------|-------------------|
| RockHudson | 1985-10-02 |
| JamesDean | 1955-10-30 |
| ... | |

- Πρέπει να φιλτράρουμε τις ημερομηνίες

FILTER

- **FILTER(condition)**
 - Boolean έλεγχος (AND, OR, REGEX, τύποι δεδομένων)

```
SELECT ?actor
WHERE {
    ?actor :playedIn :Giant .
    ?actor :diedOn ?deathdate .
    FILTER (?deathdate > "1961-11-24"^^xsd:date)
}
```



Είναι αυτό σωστό;

```
SELECT ?actor
WHERE {
  ?actor :playedIn :EastOfEden .
  FILTER (?birthday > "1930-01-01"^^xsd:date)
}
```

- Το όνομα της μεταβλητής δεν έχει καμιά σημασιολογία

```
SELECT ?actor
WHERE {
  ?actor :playedIn :EastOfEden .
  ?actor :bornOn ?birthday .
  FILTER (?birthday > "1930-01-01"^^xsd:date)
}
```

Προαιρετικές Αντιστοιχίες (Optional)

- Έχουμε δει ότι: *κάθε πρότυπο τριπλέτας (triple pattern) σε ένα πρότυπο γράφο (WHERE graph pattern) πρέπει να μπορεί να αντιστοιχίζεται στον γράφο δεδομένων*

```
SELECT ?actor ?deathdate
WHERE {?actor :playedIn :Giant .
?actor :diedOn ?deathdate .}
```

| ?actor | ?deathdate |
|------------|------------|
| RockHudson | 1985-10-02 |
| JamesDean | 1955-10-30 |
| . . . | |

- Για παράδειγμα, η Elisabeth Taylor δεν επιστρέφεται γιατί δεν έχει πεθάνει, οπότε δεν υπάρχει τριπλέτα :diedOn



- «Ποιος έπαιξε στο *Giant* και πότε πέθαινε (αν ισχύει);»

```
SELECT ?actor ?deathdate
WHERE {
    ?actor :playedIn :Giant .
    ?actor :diedOn ?deathdate .
}
```



- «Ποιος έπαιξε στο *Giant* και πότε πέθαινε (αν ισχύει);»

```
SELECT ?actor ?deathdate
WHERE {
    ?actor :playedIn :Giant .
    ?actor :diedOn ?deathdate .
}
```

- Αν κάποιος που έχει παίξει εκεί δεν έχει πεθάνει, τότε δεν θα επιστραφεί

OPTIONAL

- Προσδιορίζει ένα πρότυπο γράφου που δεν είναι απαραίτητο να αντιστοιχίζεται στον γράφο δεδομένων

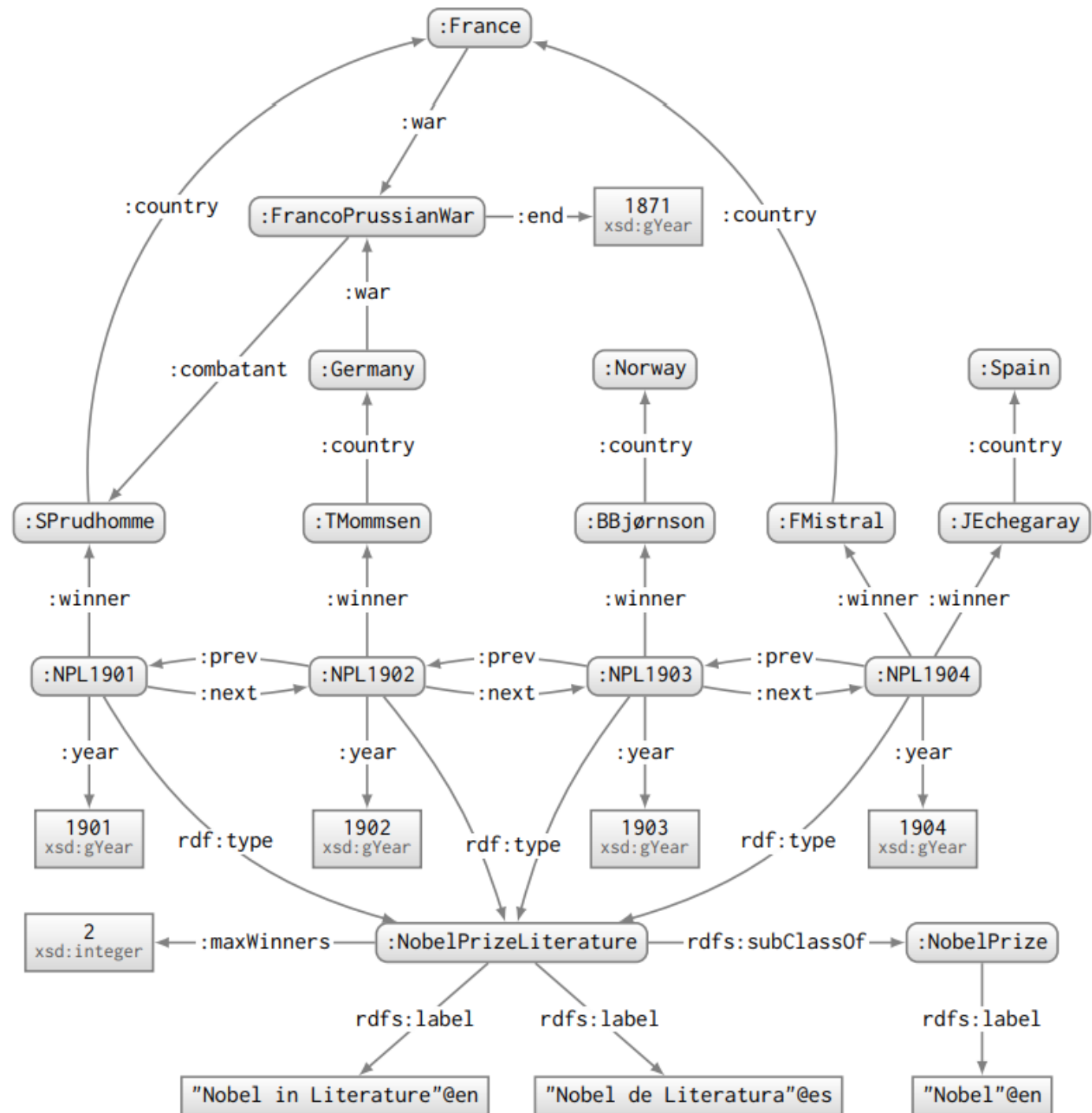
```
SELECT ?actor ?deathdate
WHERE {
    ?actor :playedIn :Giant .
    OPTIONAL {?actor :diedOn ?deathdate .}
}
```

| ?actor | ?deathdate |
|------------------|--------------|
| :RockHudson | 1985-10-02 |
| :JamesDean | 1955-10-30 |
| :ElizabethTaylor | (no binding) |
| . . . | |

Exercise 1

Write a SPARQL query to find wars in which Nobel winners in Literature were combatants

Combatants of wars of their home countries that have won a Nobel Prize in Literature



Άρνηση (SPARQL 1.1)

- Ποιες τριπλέτες (πρότυπα τριπλετών) δεν υπάρχουν στον γράφο



«Ποιοι ηθοποιοί του Giant είναι ακόμα ζωντανοί;»

```
SELECT ?actor
WHERE {
    ?actor :playedIn :Giant .
    UNSAID {?actor :diedOn ?deathdate .} }
```



- «Ηθοποιοί που δεν είναι παραγωγοί»

```
SELECT ?actor
WHERE {
    ?actor a :Actor .
    UNSAID {?actor a :Producer}
}
```

- Θεώρηση του Κλειστού Κόσμου (**Closed-world Assumption**)
 - Αφού δεν υπάρχει τριπλέτα <?actor a :Producer> θεωρούμε ότι δεν είναι Παραγωγός
- Κανονικά όμως, έχουμε βρει τους ηθοποιούς για τους οποίους δεν υπάρχουν ακόμα δεδομένα (**Θεώρηση Ανοιχτού Κόσμου – Open-world Assumption**)

Ερωτήματα Ναι/Όχι

- Μέχρι τώρα είδαμε ερωτήματα με SELECT
- Υπάρχουν και Boolean ερωτήματα (True/False)
- «Έχει πεθάνει η Elisabeth Taylor;»

```
ASK WHERE {  
    :ElizbethTaylor :diedOn ?any .  
}
```



- «Είναι ζωντανή η Elisabeth Taylor;»

```
ASK WHERE {  
  UNSAID { :ElizabETHTaylor :diedOn ?any }  
}
```



- «Γεννήθηκε κάποιος ηθοποιός του Giant μετά το 1950;»

```
ASK WHERE {  
  ?any :playedIn :Giant.  
  ?any :bornOn ?birthday .  
  FILTER (?birthday > "1950-01-01"^^xsd:date)  
}
```

CONSTRUCT

- SELECT -> αναθέσεις σε μεταβλητές
- ASK -> TRUE / FALSE
- CONSTRUCT -> επιστρέφεται γράφος
- «Όλοι οι σκηνοθέτες ταινιών»

```
?director  
:EliaKazan  
:FredGuio1  
:GeorgeCukor  
:GeorgeStevens  
:NicholasRay  
etc.
```

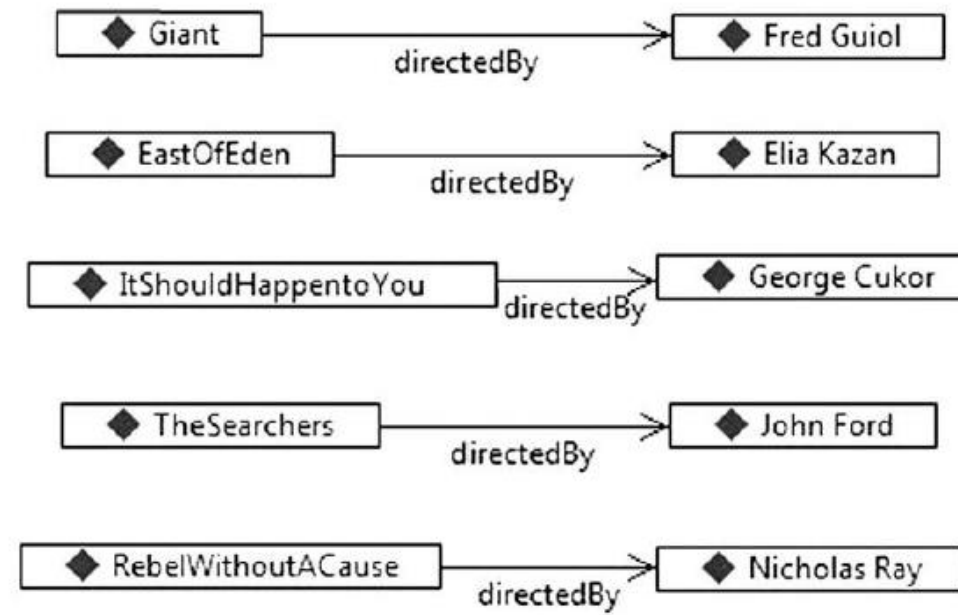
```
SELECT ?director  
WHERE {?m :directedBy ?director}
```


CONSTRUCT

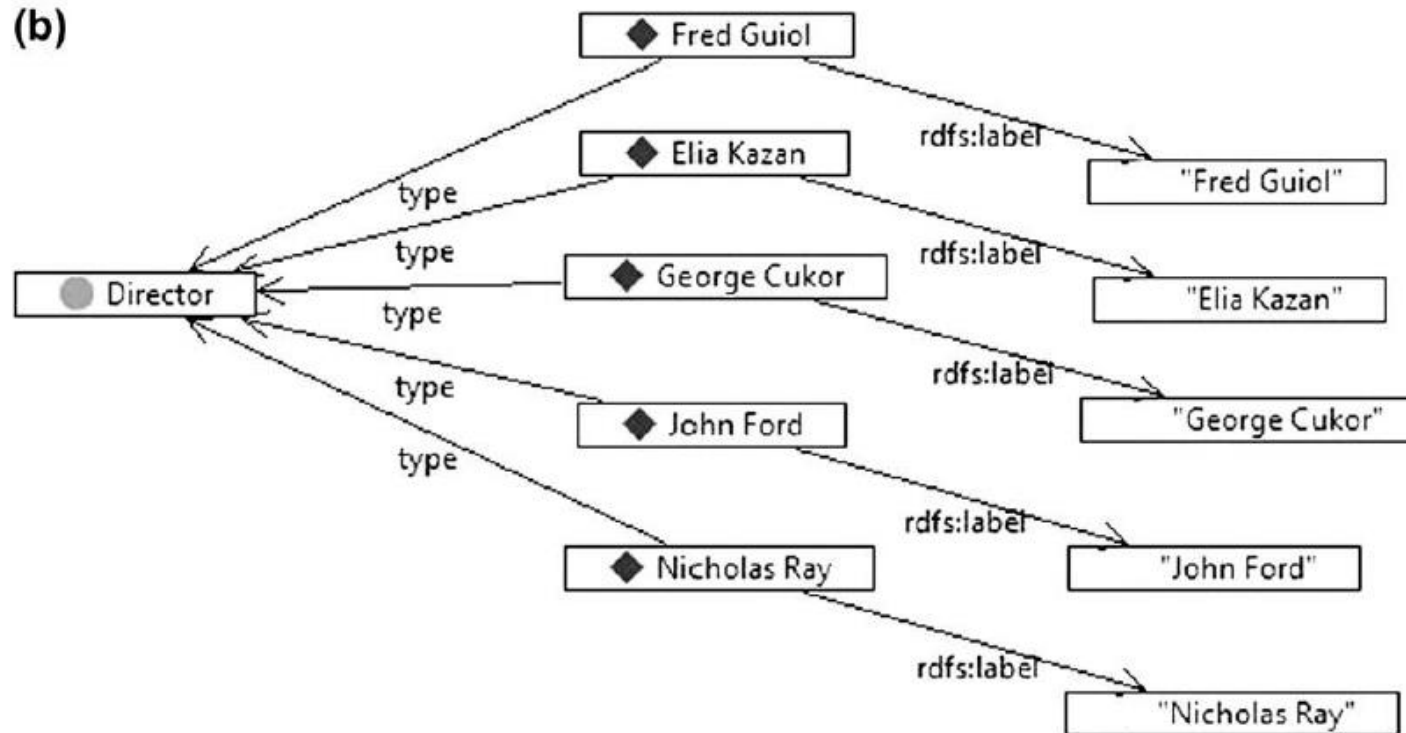
- Έστω ότι θέλουμε το αποτέλεσμα να είναι πιο «πλούσιο»
 - Να επιστρέψουμε και τον τύπο
 - Μια συμβολοσειρά αντί για το qname
 - Κτλ.

```
CONSTRUCT {  
    ?d rdf:type :Director .  
    ?d rdfs:label ?name .  
}  
WHERE {  
    ?any :directedBy ?d .  
    ?d rdfs:label ?name .  
}
```

(a)



(b)



CONSTRUCT και Κανόνες

- Μπορούν να χρησιμοποιηθούν ως κανόνες
 - Κανόνες πληρότητας: Αν ο πατέρας του John είναι ο Joe, τότε ο γιος του Joe είναι ο John
 - Κανόνες λογικής: Αν ο Σωκράτης είναι άνδρας και όλοι οι άνδρες είναι θνητοί, τότε ο Σωκράτης είναι θνητός
 - Κανόνες ορισμών: Αν η αδελφή του Ted είναι η μητέρα της Maria, τότε ο Ted είναι θείος της Maria
 - Business rules: οι πελάτες των οποίων ο τζίρος που έχουν κάνει στην εταιρία ξεπερνά τα 5000 ευρώ, είναι προτιμητέοι.

Παράδειγμα

```
:John a :Man.  
:Joe a :Man.  
:Eunice a :Woman .  
:Maria a :Woman .  
:Caroline a :Woman .  
:Ted a :Man .  
:Socrates a :Man .  
:Caroline :hasFather :John .  
:Ted :hasBrother :John .  
:John :hasFather :Joe .  
:Maria :hasMother :Eunice .  
:Maria :hasFather :Sargent .  
:Ted :hasSister :Eunice .
```

```
CONSTRUCT {?q1 :hasSon :q2 .}  
WHERE {?q2 :hasFather ?q1}
```

:Joe :hasSon :John .

:Sargent :hasSon :Maria .



```
CONSTRUCT {?q1 :hasSon :q2 .}  
WHERE {  
    ?q2 a :Man .  
    ?q2 :hasFather ?q1 .  
}
```

:Joe :hasSon :John .

Παράδειγμα

```
:John a :Man.  
:Joe a :Man.  
:Eunice a :Woman .  
:Maria a :Woman .  
:Caroline a :Woman .  
:Ted a :Man .  
:Socrates a :Man .  
:Caroline :hasFather :John .  
:Ted :hasBrother :John .  
:John :hasFather :Joe .  
:Maria :hasMother :Eunice .  
:Maria :hasFather :Sargent .  
:Ted :hasSister :Eunice .
```

```
CONSTRUCT {?q1 a :Mortal}  
WHERE {?q1 a :Man}
```

```
:John a :Mortal.  
:Joe a :Mortal.  
:Ted a :Mortal.  
:Socrates a :Mortal.
```

Maria, Eunice?



- Επιλογές
 - SPARQL UNION
 - Κλάση Human
 - Δύο κανόνες

Παράδειγμα

```
:John a :Man.  
:Joe a :Man.  
:Eunice a :Woman .  
:Maria a :Woman .  
:Caroline a :Woman .  
:Ted a :Man .  
:Socrates a :Man .  
:Caroline :hasFather :John .  
:Ted :hasBrother :John .  
:John :hasFather :Joe .  
:Maria :hasMother :Eunice .  
:Maria :hasFather :Sargent .  
:Ted :hasSister :Eunice .
```

```
CONSTRUCT {?q1 :hasUncle ?q2}  
WHERE {?q2 :hasSister ?s .  
?q1 :hasMother ?s .}
```

:Maria :hasUncle :Ted .

:Caroline :hasUncle :Ted?



```
CONSTRUCT {?q1 :hasUncle ?q2}  
WHERE {  
    ?q2 :hasSibling ?parent .  
    ?q2 a :Man .  
    ?q1 :hasParent ?parent  
}
```

Παράδειγμα

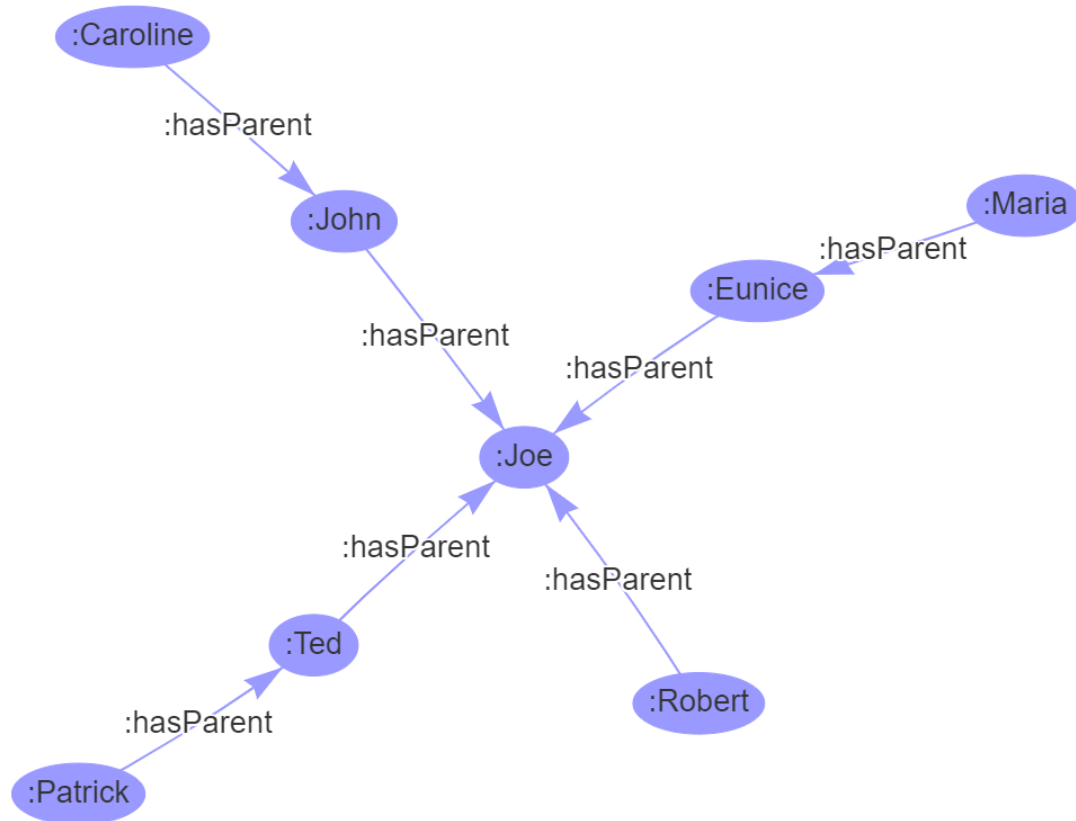
```
:ACME :totalBusiness 5253.00 .  
:PRIME :totalBusiness 12453.00 .  
:ABC :totalbusiness 1545.00 .
```

```
CONSTRUCT {  
    ?c a :PreferredCustomer  
}  
WHERE {  
    ?c :totalBusiness ?tb .  
    FILTER (?tb > 5000)  
}
```

```
:ACME a :PreferredCustomer .  
:PRIME a :PreferredCustomer .
```

Μεταβατικά Ερωτήματα

- “όλα τα μέλη της οικογένειας του Joe”
- Παιδιά του:

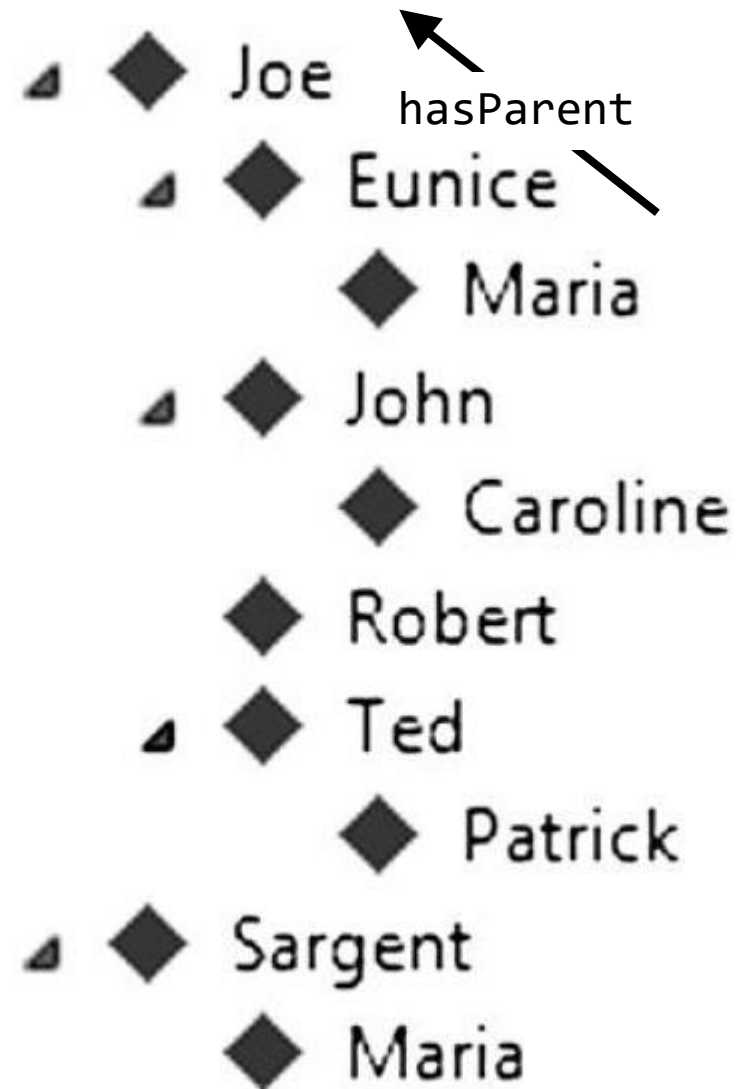


Μεταβατικά Ερωτήματα

- “όλα τα μέλη της οικογένειας του Joe”
- Παιδιά του:

```
SELECT ?member  
WHERE {?member :hasParent :Joe}
```

?member
Eunice
John
Robert
Ted



Μεταβατικά Ερωτήματα

- “όλα τα μέλη της οικογένειας του Joe”
- Εγγόνια του:

```
SELECT ?member
WHERE {
    ?int :hasParent :Joe .
    ?member :hasParent ?int .
}
```

?member
Maria
Caroline
Patrick



Μεταβατικά Ερωτήματα

- “όλα τα μέλη της οικογένειας του Joe”
- Δισέγγονα:



Τελεστής Μεταβατικότητας (SPARQL 1.1)

```
SELECT ?member  
WHERE {?member :hasParent* :Joe .}
```

- Αν έχουμε το * μετά το όνομα μιας ιδιότητας, το πρότυπο τριπλέτας αντιστοιχίζεται σε οποιοδήποτε αριθμό διασυνδεδεμένων εμφανίσεων της ίδιας ιδιότητας

?member

Joe

Eunice

Maria

John

Caroline

Robert

Ted

Patrick

| Syntax Form | Property Path Expression Name | Matches |
|---|-------------------------------|--|
| <i>iri</i> | PredicatePath | An IRI. A path of length one. |
| $\wedge elt$ | InversePath | Inverse path (object to subject). |
| <i>elt1</i> / <i>elt2</i> | SequencePath | A sequence path of <i>elt1</i> followed by <i>elt2</i> . |
| <i>elt1</i> <i>elt2</i> | AlternativePath | A alternative path of <i>elt1</i> or <i>elt2</i> (all possibilities are tried). |
| <i>elt</i> * | ZeroOrMorePath | A path that connects the subject and object of the path by zero or more matches of <i>elt</i> . |
| <i>elt</i> + | OneOrMorePath | A path that connects the subject and object of the path by one or more matches of <i>elt</i> . |
| <i>elt</i> ? | ZeroOrOnePath | A path that connects the subject and object of the path by zero or one matches of <i>elt</i> . |
| ! <i>iri</i> or !(<i>iri</i> ₁ ... <i>iri</i> _{<i>n</i>}) | NegatedPropertySet | Negated property set. An IRI which is not one of <i>iri</i> _{<i>i</i>} . ! <i>iri</i> is short for !(<i>iri</i>). |
| ! $\wedge iri$ or !($\wedge iri_1$... $\wedge iri_n$) | NegatedPropertySet | Negated property set where the excluded matches are based on reversed path. That is, not one of <i>iri</i> ₁ ... <i>iri</i> _{<i>n</i>} as reverse paths. ! $\wedge iri$ is short for !($\wedge iri$). |
| !(<i>iri</i> ₁ ... <i>iri</i> _{<i>j</i>} $\wedge iri_{j+1}$... $\wedge iri_n$) | NegatedPropertySet | A combination of forward and reverse properties in a negated property set. |
| (<i>elt</i>) | | A group path <i>elt</i> , brackets control precedence. |

Όρια και Ταξινόμηση

- «Ταινίες που έχει παίξει ο James Dean και τις ημερομηνίες κυκλοφορίας»

```
SELECT ?movie ?date
WHERE {
  :JamesDean :playedIn ?m.
  ?m rdfs:label ?movie .
  ?m dc:date ?date .
}
```

| ?movie | ?date |
|--------------------|-------|
| Giant | 1956 |
| EastOfEden | 1955 |
| RebelWithoutaCause | 1955 |

- Δεν υπάρχει κάποια σειρά στα αποτελέσματα...

ORDER BY

- Μπαίνει μετά από το πρότυπο γράφου
- Προσδιορίζει τις μεταβλητές που θα χρησιμοποιηθούν για την σειρά

```
SELECT ?title ?date
WHERE {
    :JamesDean :playedIn ?movie.
    ?movie rdfs:label ?title .
    ?movie dc:date ?date .
}
ORDER BY ?date
```

| ?title | ?date |
|--------------------|--------------|
| EastOfEden | 1955 |
| RebelWithoutaCause | 1955 |
| Giant | 1956 |

ORDER BY


- Μπαίνει μετά από το πρότυπο γράφου
- Προσδιορίζει τις μεταβλητές που θα χρησιμοποιηθούν για την σειρά

```
SELECT ?title ?date
WHERE {
    :JamesDean :playedIn ?movie.
    ?movie rdfs:label ?title .
    ?movie dc:date ?date .
}
ORDER BY ?title
```

| ?title | ?date |
|--------------------|-------|
| EastOfEden | 1955 |
| Giant | 1956 |
| RebelWithoutaCause | 1955 |

LIMIT

- Περιορισμός του αριθμού των αποτελεσμάτων που επιστρέφονται
 - Π.χ. θέλω μόνο τα top-10, web interface
- Συνδυάζεται με το ORDER BY
 - Αν όχι, η SPARQL μηχανή αποφασίζει ποια θα επιστρέψει
- «Ποια είναι η παλιότερη ταινία του James Dean;»



```
SELECT ?title
WHERE {
    :JamesDean :playedIn ?m.
    ?m rdfs:label ?title .
    ?m dc:date ?date .
}
ORDER BY ?date
LIMIT 1
```



- «Και οι τρεις πρωταγωνιστές στο *Rebel without a Cause* πέθαναν νέοι. Ποιος πέθανε πρώτος;»

:playedIn
:diedOn

```
SELECT ?first
WHERE {
    ?who :playedIn :RebelWithoutaCause .
    ?who rdfs:label ?first .
    ?who :diedOn ?date .
}
ORDER BY ?date
LIMIT 1
```

DESC

- Η προεπιλογή είναι τα αποτελέσματα να επιστρέφονται σε αύξουσα σειρά
- «Ηθοποιός που πέθανε τελευταίος»

```
SELECT ?first
WHERE {
    ?who :playedIn :RebelWithoutaCause .
    ?who rdfs:label ?first .
    ?who :diedOn ?date .
}
ORDER BY DESC(?date)
LIMIT 1
```

Συναθροίσεις και Ομαδοποιήσεις

- COUNT, MIN, MAX, AVG, SUM
- «Σε πόσες ταινίες έχει παίξει ο James Dean;»

```
SELECT (COUNT (?movie) AS ?howmany)
WHERE {
    :JamesDean :playedIn ?movie .
}
```

?howmany
3



Πόσα είναι τα παιδιά του John?

SUM

- «Ποιος είναι ο τζίρος που κάνουν οι πελάτες κάθε χρόνο;»

| Company | Amount | Year |
|---------|--------|------|
| ACME | \$1250 | 2010 |
| PRIME | \$3000 | 2009 |
| ABC | \$2500 | 2009 |
| ABC | \$2800 | 2010 |
| PRIME | \$1950 | 2010 |
| ACME | \$2500 | 2009 |
| ACME | \$3100 | 2010 |
| ABC | \$1500 | 2009 |
| ACME | \$1250 | 2009 |
| PRIME | \$2350 | 2009 |
| PRIME | \$1850 | 2010 |

Συνολικός τζίρος

```
:row1 a :Sale .  
:row1 :company :ACME .  
:row1 :amount 1250 .  
:row1 :year 2010 .
```

```
SELECT (SUM (?val) AS ?total)  
WHERE {  
    ?s a :Sale .  
    ?s :amount ?val  
}
```

- Έστω ότι θέλουμε να επιστρέψουμε το σύνολο ανά έτος

GROUP BY

- Ορίζει πώς η συνάθροιση θα ομαδοποιήσει τα σύνολα
- Αθροίζονται τα αποτελέσματα των συνόλων

```
SELECT ?year (SUM (?val) AS ?total)
WHERE {
    ?s a :Sale .
    ?s :amount ?val .
    ?s :year ?year
}
GROUP BY ?year
```

| ?year | ?total |
|-------|----------|
| 2009 | 13100.00 |
| 2010 | 10950.00 |

GROUP BY

- Μπορούμε να έχουμε >1 μεταβλητές
- «Τζίρος ανά πελάτη ανά χρονιά»

```
SELECT ?year ?company (SUM (?val) AS ?total)
WHERE {
    ?s a :Sale .
    ?s :amount ?val .
    ?s :year ?year .
    ?s :company ?company .
}
GROUP BY ?year ?company
```

| ?year | ?company | ?total |
|-------|----------|---------|
| 2009 | ACME | 3750.00 |
| 2009 | ABC | 4000.00 |
| 2009 | PRIME | 5350.00 |
| 2010 | ACME | 4350.00 |
| 2010 | PRIME | 3800.00 |
| 2010 | ABC | 2800.00 |

HAVING


- Εμφάνιση μερικών συνόλων
- «Πελάτες που σημείωσαν >5000 τζίρο κάποια χρονιά»

```
SELECT ?year ?company (SUM (?val) AS ?total)
WHERE {
    ?s a :Sale .
    ?s :amount ?val .
    ?s :year ?year .
    ?s :company ?company .
}
GROUP BY ?year ?company
HAVING (?total > 5000)
```

| ?year | ?company | ?total |
|-------|----------|---------|
| 2009 | PRIME | 5350.00 |

UNION

- Σε ένα πρότυπο γράφου, υπονοείται το ΚΑΙ μεταξύ των τριπλετών
- Κάποιες φορές είναι χρήσιμο να μπορούμε να αναπαραστήσουμε την ένωση γράφων
- «Όλοι οι ηθοποιοί που έπαιξαν είτε στο *Giant* είτε στο *Rebel Without a Cause*»



```
SELECT ?actor
WHERE {
  {
    ?actor :playedIn :Giant .
  }
  UNION
  {
    ?actor :playedIn :RebelWithoutaCause .
  }
}
```

?actor

Ann Doran
Carroll Baker
Elizabeth
Taylor
James Dean
James Dean
Jim Backus
Mercedes
McCambridge
Natalie Wood
Rock Hudson
Sal Mineo
Sal Mineo

UNION και CONSTRUCT

Παράδειγμα

```
CONSTRUCT {  
    ?q1 a :Mortal  
}  
WHERE {  
    ?q1 a :Man .  
}
```

```
CONSTRUCT {  
    ?q1 a :Mortal  
}  
WHERE {  
    {?q1 a :Man .}  
    UNION  
    {?q1 a :Woman .}  
}
```

Αναθέσεις

- Υπολογισμός ειδικού σκοπού στα πλαίσια ενός ερωτήματος
- Στην ουσία ορίζεται η τιμή μιας μεταβλητής αντί να αντιστοιχιστεί με κάποια τιμή από τα δεδομένα (γράφο)
- Κάτι αντίστοιχο είχαμε δει στις συναθροίσεις

SELECT (*expression* **AS** ?var)

Ανάθεση στο SELECT

```
SELECT ?title (?p*(1-?discount) AS ?price)
{
  ?x :price ?p .
  ?x :title ?title .
  ?x :discount ?discount .
}
```

Ανάθεση στο SELECT

```
SELECT  ?title (?p AS ?fullPrice)
        (?fullPrice*(1-?discount) AS ?customerPrice)
{ ?x ns:price ?p .
  ?x dc:title ?title .
  ?x ns:discount ?discount
}
```


Ανάθεση σε πρότυπο γράφου

```
SELECT ?title ?price
{
  ?x ns:price ?p .
  ?x ns:discount ?discount
  BIND (?p*(1-?discount) AS ?price)
  FILTER(?price < 20).
  ?x dc:title ?title .
}
```

Παράδειγμα

- Έστω ότι έχουμε τα παρακάτω δεδομένα

```
:DeanAllemang rdf:type :Person ;  
  :firstName "Dean" ;  
  :lastName "Allemang" .
```

```
:JimHendler rdf:type :Person ;  
  :firstName "James" ;  
  :lastName "Hendler" .
```

```
:WorkingOntologist rdf:type :Book ;  
  rdfs:label "Semantic Web for the Working Ontologist" ;  
  dc:creator :DeanAllemang , :JimHendler .
```

Πώς μπορούμε να
επιστρέψουμε το πλήρες
όνομα του συγγραφέα;



Παράδειγμα

- `fn:concat`
 - Συνένωση αλφαριθμητικών

?fullname
James Hendler
Dean Allemang

```
SELECT (fn:concat (?first, " ", ?last) AS ?fullname)
WHERE {
    :WorkingOntologist dc:creator ?author .
    ?author :firstName ?first .
    ?author :lastName ?last .
}
```

Ομόσπονδα Ερωτήματα (federated)

- Μέχρι τώρα υποθέσαμε ότι όλη η πληροφορία βρισκόταν σε έναν γράφο
- Υπάρχουν περιπτώσεις που δεν ισχύει αυτό
 - π.χ. Είναι πολλά τα δεδομένα ώστε να γίνει κάποια συγχώνευση, δεν έχουμε πρόσβαση στα δεδομένα
- Τα δεδομένα πολλές φορές γίνονται διαθέσιμα μέσω endpoints
 - π.χ. SPARQL endpoints (π.χ. <https://query.wikidata.org/>, <https://dbpedia.org/sparql>)
- Υπάρχει δυνατότητα εκτέλεσης και συνδυασμού αποτελεσμάτων από διαφορετικές πηγές (endpoints)




```
SELECT ?entry
WHERE {
    ?actor :playedIn :Giant .
    ?actor rdfs:label ?name .
    SERVICE http://dbpedia.org/sparql
    {
        ?entry rdfs:label ?name .
    }
}
```



```
SELECT DISTINCT ?name ?realname
WHERE {
    ?actor :playedIn :Giant .
    ?actor rdfs:label ?name .
    SERVICE <http://dbpedia.org/sparql>
    {
        ?entry rdfs:label ?name .
        ?entry dbpedia:birthname ?realname.
    }
}
```

Σημασιολογία RDFS και Κανόνες SPARQL

- Έχουμε ήδη δει ότι μέσω CONSTRUCT μπορούμε να εκφράσουμε κανόνες σε SPARQL
- Επίσης έχουμε δει τη σημασιολογία της RDFS
 - π.χ. τη σημασιολογία του `rdfs:domain`

`?P rdfs:domain ?D .`  `?x rdf:type ?D .`
`?x ?P ?y .`

- Στην ουσία το παραπάνω είναι ένας κανόνας
- Η σημασιολογία της RDFS μπορεί να εκφραστεί με κανόνες
- Η υλοποίηση μπορεί να βασιστεί σε κανόνες SPARQL

RDFS Rules

- Σχέσεις if-then πάνω σε δεδομένα για την παραγωγή έγκυρων συμπερασμών (valid inferences)

| ID | if G matches | then G RDFS _{D} -entails |
|--------|---|---|
| rdfD1 | $?x ?p ?l .$ ($?l$ a literal with datatype IRI $dt(?l) \in D$) | $?x ?p _ : b . _ : b \text{ a } dt(?l) .$ |
| rdfD2 | $?x ?p ?y .$ | $?p \text{ a } \text{rdf:Property} .$ |
| rdfs1 | $?u \in D$ | $?u \text{ a } \text{rdfs:Datatype} .$ |
| rdfs2 | $?p \text{ rdfs:domain } ?c . ?x ?p ?y .$ | $?x \text{ a } ?c .$ |
| rdfs3 | $?p \text{ rdfs:range } ?c . ?x ?p ?y .$ | $?y \text{ a } ?c .$ |
| rdfs4a | $?x ?p ?y .$ | $?x \text{ a } \text{rdfs:Resource} .$ |
| rdfs4b | $?x ?p ?y .$ | $?y \text{ a } \text{rdfs:Resource} .$ |
| rdfs5 | $?p \text{ rdfs:subPropertyOf } ?q . ?x ?p ?y .$ | $?x ?q ?y .$ |
| rdfs6 | $?p \text{ a } \text{rdf:Property} .$ | $?p \text{ rdfs:subPropertyOf } ?p .$ |
| rdfs7 | $?p \text{ rdfs:subPropertyOf } ?q . ?q \text{ rdfs:subPropertyOf } ?r .$ | $?p \text{ rdfs:subPropertyOf } ?r .$ |
| rdfs8 | $?c \text{ a } \text{rdfs:Class} .$ | $?c \text{ rdfs:subClassOf } \text{rdfs:Resource} .$ |
| rdfs9 | $?c \text{ rdfs:subClassOf } ?d . ?x \text{ a } ?c .$ | $?x \text{ a } ?d .$ |
| rdfs10 | $?c \text{ a } \text{rdfs:Class} .$ | $?c \text{ rdfs:subClassOf } ?c .$ |
| rdfs11 | $?c \text{ rdfs:subClassOf } ?d . ?d \text{ rdfs:subClassOf } ?e .$ | $?c \text{ rdfs:subClassOf } ?e .$ |
| rdfs12 | $?p \text{ a } \text{rdfs:ContainerMembershipProperty} .$ | $?p \text{ rdfs:subPropertyOf } \text{rdfs:member} .$ |
| rdfs13 | $?d \text{ a } \text{rdfs:Datatype} .$ | $?d \text{ rdfs:subClassOf } \text{rdf:Literal} .$ |


```
CONSTRUCT {  
    ?x a ?c2 .  
}  
WHERE {  
    ?c1 rdfs:subClassOf ?c2 .  
    ?x a ?c1 .  
}
```

rdfs9

```
?c rdfs:subClassOf ?d . ?x a ?c .  
-> ?x a ?d .
```

```
CONSTRUCT {  
    ?c1 rdfs:subClassOf ?c3 .  
}  
WHERE {  
    ?c1 rdfs:subClassOf ?c2 .  
    ?c2 rdfs:subClassOf ?c3 .  
}
```

rdfs11

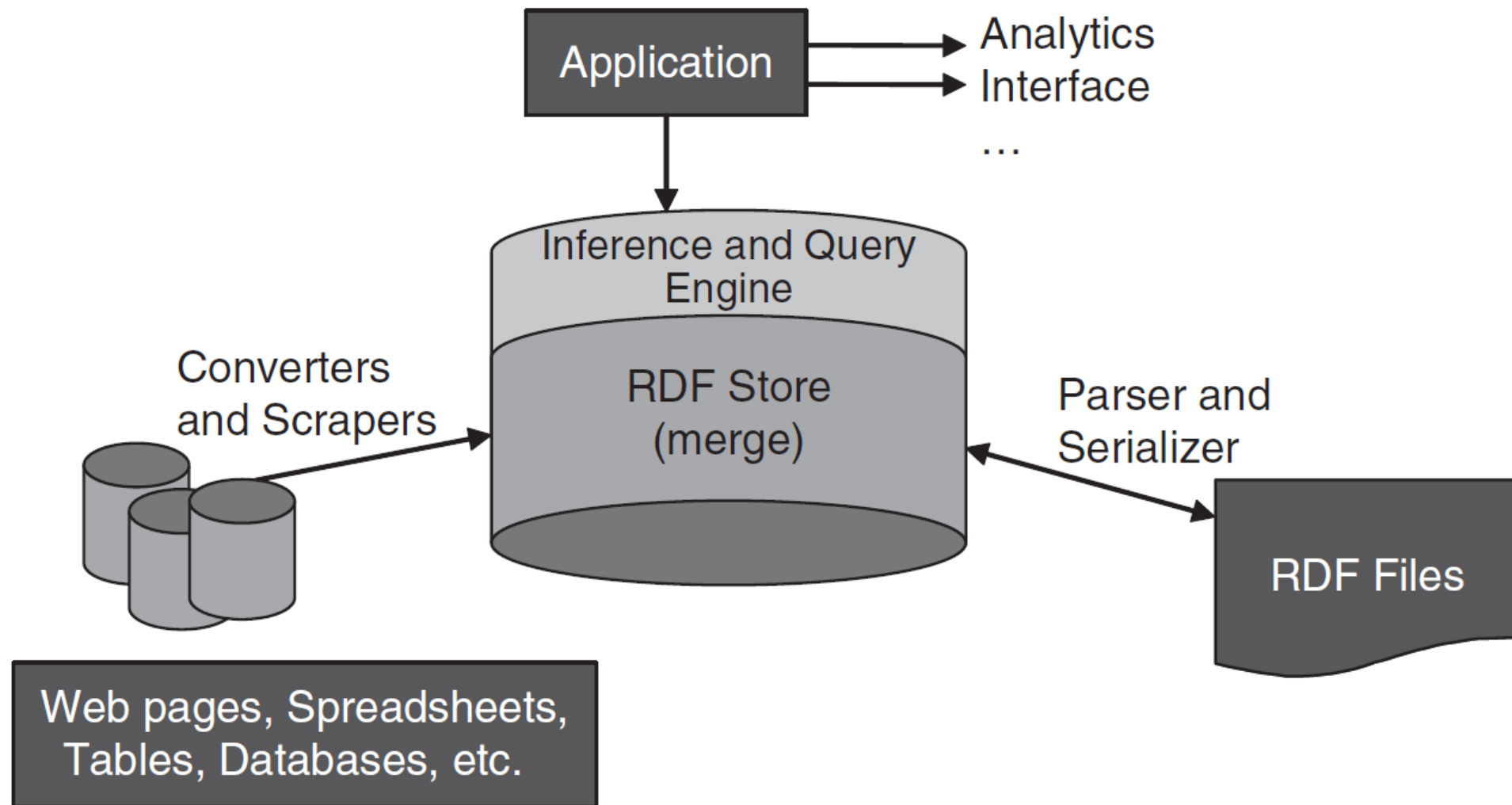
```
?c rdfs:subClassOf ?d . ?d rdfs:subClassOf ?e .  
-> ?c rdfs:subClassOf ?e .
```

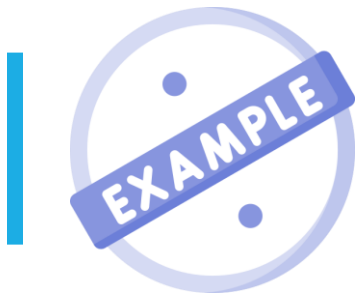
```
CONSTRUCT {  
    ?x ?p2 ?y .  
}  
WHERE {  
    ?p1 rdfs:subPropertyOf ?p2 .  
    ?x ?p1 ?y .  
}
```

rdfs5

```
?p rdfs:subPropertyOf ?q . ?x ?p ?y  
-> ?x ?q ?y .
```

Αρχιτεκτονική Σημασιολογικής Εφαρμογής





```
shop:Henleys rdfs:subClassOf shop:Shirts.  
shop:ChamoisHenley rdf:type shop:Henleys.
```

```
SELECT ?item  
WHERE {?item rdf:type shop:Shirts . }
```

- Τι θα επιστρέψει το ερώτημα;
- **Τίποτα!** Χρειαζόμαστε πρώτα να τρέξουμε έναν μηχανισμό RDFS συμπερασμού

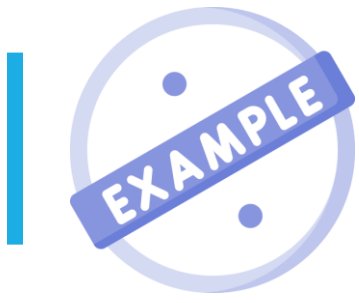
Δηλωμένες / Συναγώμενες Τριπλέτες

■ Δηλωμένες

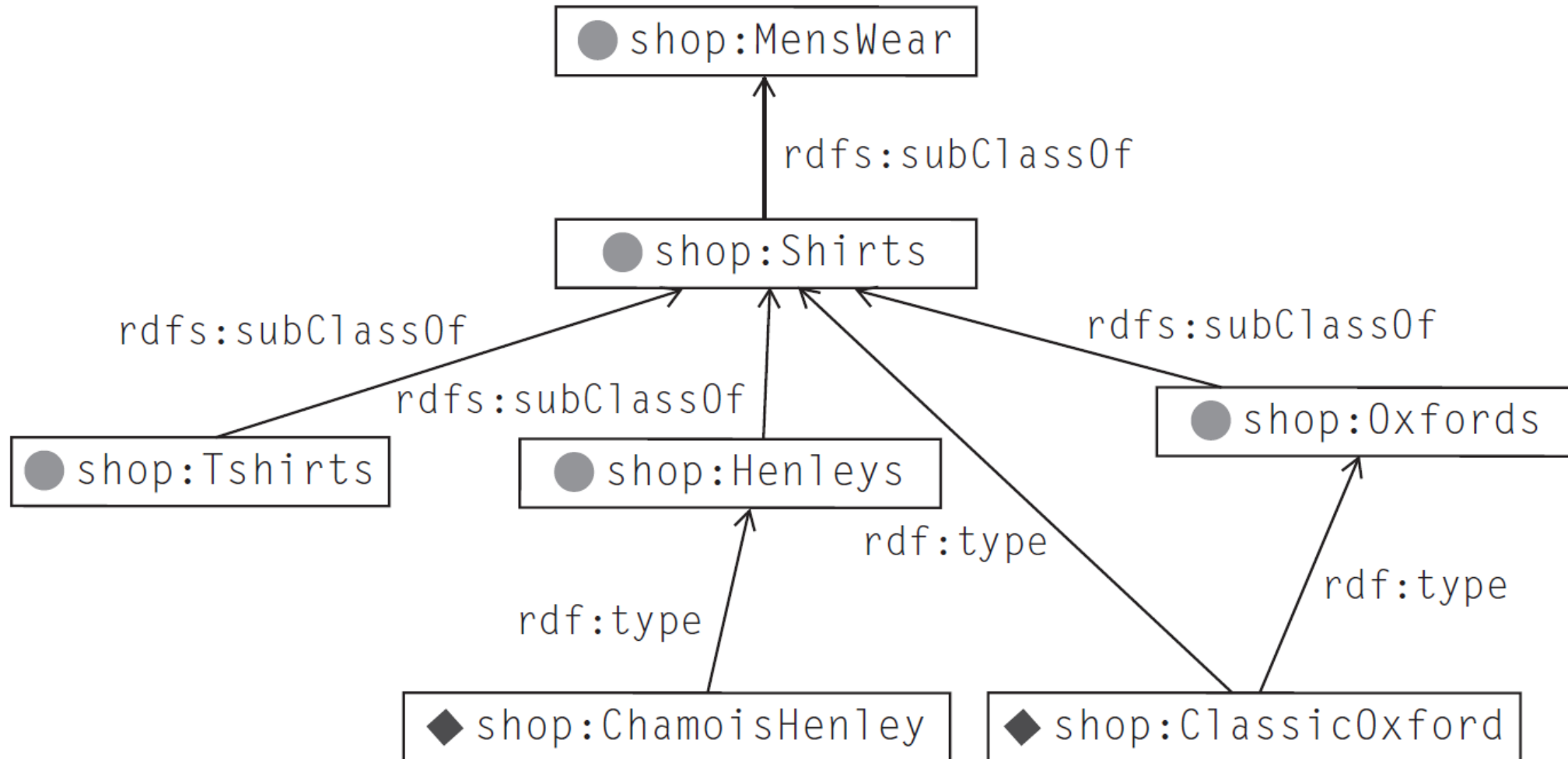
- Τριπλέτες που εισήχθησαν στην σημασιολογική βάση (RDF αποθήκη)

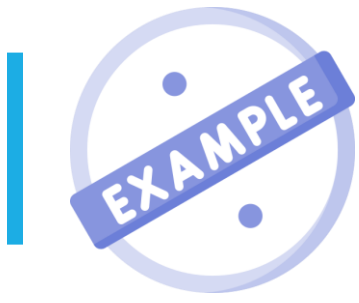
■ Συναγόμενες

- Πρόσθετες τριπλέτες που έχουν εξαχθεί ως συμπεράσματα
- Υπάρχει περίπτωση ο μηχανισμός εξαγωγής συμπερασμάτων να εξάγει μια τριπλέτα που έχει ήδη δηλωθεί
 - Εξακολουθεί να θεωρείται δηλωμένη



```
shop:Henleys rdfs:subClassOf shop:Shirts.  
shop:Shirts rdfs:subClassOf shop:MensWear.  
shop:Blouses rdfs:subClassOf shop:WomensWear.  
shop:Oxfords rdfs:subClassOf shop:Shirts.  
shop:Tshirts rdfs:subClassOf shop:Shirts.  
shop:ChamoisHenley rdf:type shop:Henleys.  
shop:ClassicOxford rdf:type shop:Oxfords.  
shop:ClassicOxford rdf:type shop:Shirts.  
shop:BikerT rdf:type shop:Tshirts.  
shop:BikerT rdf:type shop:MensWear.
```

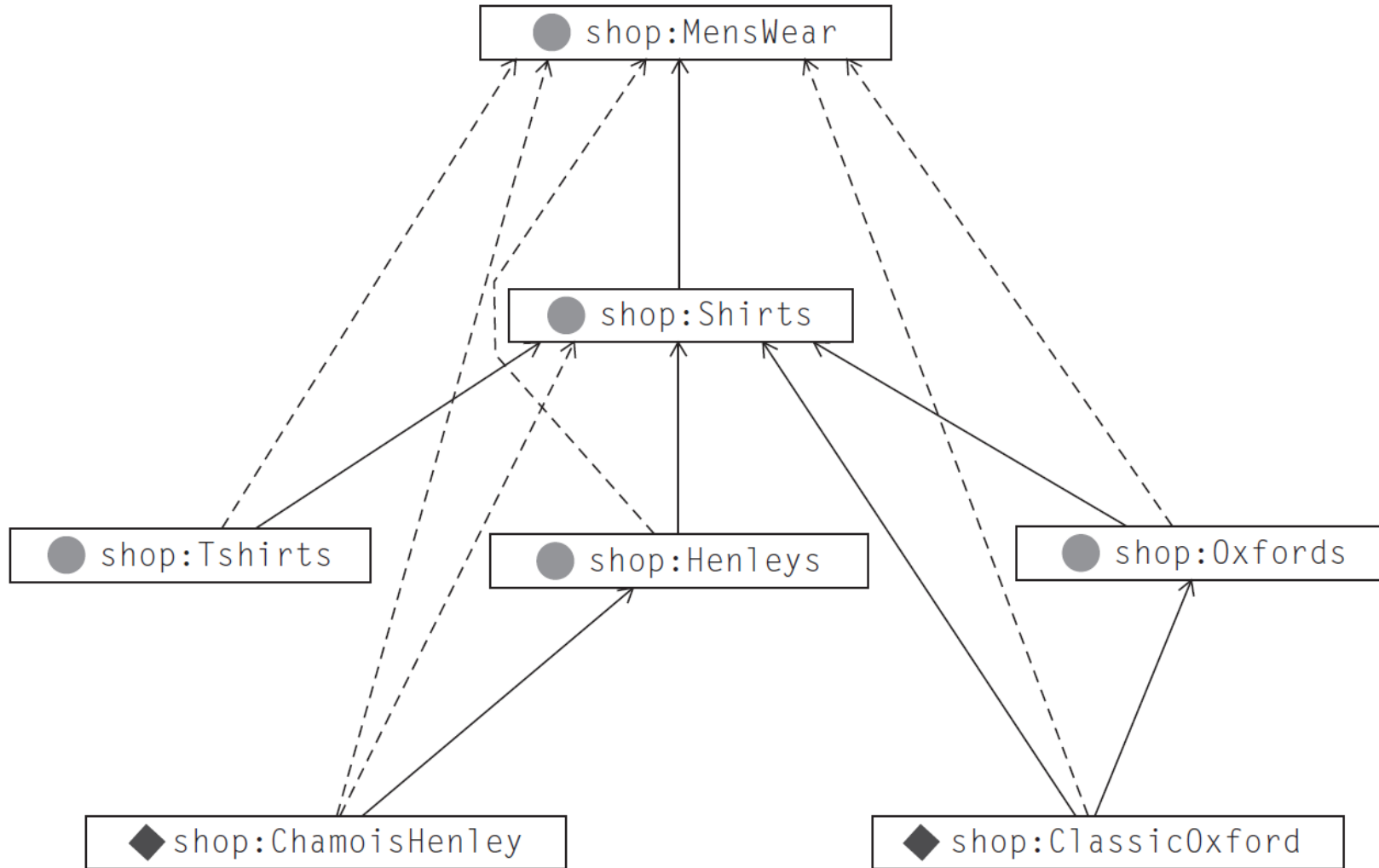




shop:ChamoisHenley rdf:type shop:Shirts.
shop:ChamoisHenley rdf:type shop:MensWear.
shop:ClassicOxford rdf:type shop:Shirts.
shop:ClassicOxford rdf:type shop:MensWear.
shop:BikerT rdf:type shop:Shirts.
shop:BikerT rdf:type shop:MensWear.



Όλες οι τριπλέτες



Πότε συμβαίνει η εξαγωγή συμπεράσματος;

- Materialisation (cached inferencing – αποθηκευμένος συμπερασμός)
 - Μόλις εισαχθεί μια τριπλέτα, τρέχουν οι κανόνες που ικανοποιούνται και το αποτέλεσμα εισάγεται πάλι στην αποθήκη
 - Απλή στην υλοποίηση
 - Πληθυσμιακή έκρηξη
 - Τι θα γίνει αν αλλάξει κάτι στην βάση;
- In-time Inferencing (έγκαιρος συμπερασμός)
 - Δεν αποθηκεύονται οι τριπλέτες – Ενεργοποιείται μόνο για την απάντηση ερωτημάτων
 - Δεν μεγαλώνει η βάση
 - Δεν επηρεάζεται από αλλαγές
 - Είναι σχετικά πιο δύσκολο να υλοποιηθεί
 - Λόγω τη μη αποθήκευσης των αποτελεσμάτων, συμπερασμοί μπορεί να επαναλαμβάνονται
 - Ο συμπερασμός γίνεται «on the fly» οπότε ίσως αργήσει το αποτέλεσμα

Ξανά....


```
shop:Henleys rdfs:subClassOf shop:Shirts.  
shop:ChamoisHenley rdf:type shop:Henleys.
```

```
SELECT ?item  
WHERE {?item rdf:type shop:Shirts . }
```

- Έστω ότι δεν έχουμε διαδικασία συμπερασμού, ποιο θα μπορούσε να είναι το ερώτημα για να βρει τα πουκάμισα τύπου Shirt;

```
SELECT ?item
WHERE {
    ?class :subClassOf :Shirts .
    ?item a ?class .
}
```

```
SELECT ?item
WHERE {
    ?class :subClassOf* :Shirts .
    ?item a ?class .
}
```

- 
- Παράδειγμα στο TopBraid...
 - shopping.ttl
 - SELECT (property paths)
 - CONSTRUCT (εισαγωγή συμπερασμών – κανόνες)

Αναφορές - Πηγές



Αναφορά Δημιουργού - Μη Εμπορική
Χρήση - Παρόμοια Διανομή 4.0 Διεθνές
(CC BY-NC-SA 4.0)

- Allemang Dean, Hendler Jim, 2020. Ο Σημασιολογικός Ιστός για τους Δημιουργούς Οντολογιών. Επιμέλεια ελληνικού κειμένου: Βούρος Γεώργιος, Κώτης Κωνσταντίνος, Σαντιπαντάκης Γεώργιος. Εκδόσεις ΔΙΣΙΓΜΑ. ISBN 978-618-202-007-4.
- The Web of Data (Book), by Aidan Hogan
- Introduction to the Semantic Web – Tutorial
(<https://www.w3.org/2009/Talks/0615-SanJose-tutorial-IH/Slides.pdf>)