# ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ

---

## Μηχανική Γνώσης και Συστήματα Γνώσης

## SPARQL - Γλώσσα επερώτησης RDF

Μανώλης Μαραγκουδάκης

Τμήμα Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων

---




## Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.


## Χρηματοδότηση


- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Αιγαίου**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.




Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

# SPARQL

Γλώσσα επερώτησης RDF

# SPARQL

- **S**imple **P**rotocol **a**nd **R**DF **Q**uery **L**anguage
- Πρωτόκολλο
  - GET /sparql/?**query**=*EncodedQuery*
    &**default-graph-uri**=
    http://my.example/publishers
    &**named-graph-uri**=http://my.example/bob
    &**named-graph-uri**=http://my.example/alice
    HTTP/1.1Host: my.example
    User-agent: sparql-client/0.1
- Ρωτάει γράφους RDF (Τριπλέτες)

# Απλό παράδειγμα

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?c
WHERE
{
    ?c rdf:type rdfs:Class .
}
```

- Ανακτά όλες τις τριπλέτες για τις οποίες:

-το property είναι rdf:type

-το object είναι rdfs:Class

Δηλ. ανακτά όλες τις κλάσεις της RDF!

# Παράδειγμα 2

- Ανάκτησε όλα τα στιγμιότυπα μιας κλάσης (π.χ. course) :

(οι δηλώσεις των rdf, rdfs prefixes έχουν παραληφθεί για απλότητα)

```
PREFIX uni: <http://www.mydomain.org/uni-ns#>
SELECT ?i
WHERE
{
   ?i rdf:type uni:course .
}
```

# Implicit Join

- Ανάκτησε όλους τους lecturers και τα τηλέφωνα τους:

```
SELECT ?x ?y
WHERE
{ ?x rdf:type uni:Lecturer ;
        uni:phone ?y . }
```

- Implicit join: Επιβάλουμε στο δεύτερο σκέλος να ανακτήσουμε μόνον εκείνες τις τριπλέτες των οποίων το resource είναι η μεταβλητή **?x**
  - Συντακτική συντόμευση
    - Μια Semicolon (;) μας δηλώνει ότι η δεύτερη τριπλέτα έχειτο ίδιο subject με την προηγούμενη

# Implicit join (2)

- Εναλλακτικά, γράφουμε:

```
SELECT ?x ?y
WHERE
{
   ?x rdf:type uni:Lecturer .
   ?x uni:phone ?y .
}
```

# Τριπλέτες

**Δεδομένα**

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
 _:a foaf:name "Johnny Lee Outlaw" .
_:a foaf:mbox <mailto:jlow@example.com> .
_:b foaf:name "Peter Goodguy" .
_:b foaf:mbox <mailto:peter@example.org> .
```

**Ερώτηση**

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox
WHERE
{ ?x foaf:name ?name .
  ?x foaf:mbox ?mbox }
```

**Ομαδοποίηση Τριπλετών**

**Αποτέλεσμα**

| **name** | **mbox** |
|---|---|
| "Johnny Lee Outlaw" | <mailto:jlow@example.com> |
| "Peter Goodguy" | <mailto:peter@example.org> |

# Περιορισμοί τιμών

**Δεδομένα**

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix : <http://example.org/book/> .
@prefix ns: <http://example.org/ns#> .
:book1 dc:title "SPARQL Tutorial" .
:book1 ns:price 42 .
:book2 dc:title "The Semantic Web" .
:book2 ns:price 23 .
```

**Ερώτηση**

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX ns: <http://example.org/ns#>
SELECT ?title ?price
WHERE { ?x ns:price ?price .
        FILTER ?price < 30 .
        ?x dc:title ?title . }
```

**Αποτέλεσμα**

| title | price |
|---|---|
| "The Semantic Web" | 23 |

# Χρήση ετικέτας Optional

**Δεδομένα**

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix : <http://example.org/book/> .
@prefix ns: <http://example.org/ns#> .
:book1 dc:title "SPARQL Tutorial" .
:book1 ns:price 42 .
:book2 dc:title "The Semantic Web" .
:book2 ns:price 23 .
```

**Ερώτηση**

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX ns: <http://example.org/ns#>
SELECT ?title ?price
WHERE { ?x dc:title ?title .
        OPTIONAL { ?x ns:price ?price .
                   FILTER ?price < 30 }}
```

**Αποτέλεσμα**

| **title** | **price** |
|---|---|
| “SPARQL Tutorial“ | |
| "The Semantic Web" | 23 |

# Construct (δημιουργία τριπλετών)

**Δεδομένα:**

```
@prefix org: <http://example.com/ns#> .
_:a             org:employeeName
        "Alice" .
_:a             org:employeeId
        12345 .
_:b             org:employeeName
        "Bob" .
_:b             org:employeeId
        67890 .
```

# Construct (δημιουργία τριπλετών)

**Ερώτημα:**

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX org: <http://example.com/ns#>
CONSTRUCT
{ ?x foaf:name ?name }
WHERE
{ ?x org:employeeName ?name }
```

# Construct (δημιουργία τριπλετών)

**Αποτέλεσμα:**

@prefix org: <http://example.com/ns#> .

_:x foaf:name "Alice" .
_:y foaf:name "Bob" .

# Construct (δημιουργία τριπλετών)

**Αποτέλεσμα σε RDF**

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
    syntax-ns#" xmlns:foaf="http://xmlns.com/foaf/0.1/" >
    <rdf:Description>
        <foaf:name>Alice</foaf:name>
    </rdf:Description>
    <rdf:Description>
        <foaf:name>Bob</foaf:name>
    </rdf:Description>
</rdf:RDF>
```

# Σύνθετο παράδειγμα

- Στις παρακάτω διαφάνειες, θα συνενώσουμε 2 RDF αρχεία (μαζί τις τις RDFS κλάσεις τους) και θα εκτελέσουμε ένα σύνθετο ερώτημα σε SPARQL
  - Αρχικά θα βλέπουμε τα δεδομένα σε XML, μετά σε SPARQL-Turtle μορφή και τέλος σε RDF γράφο.

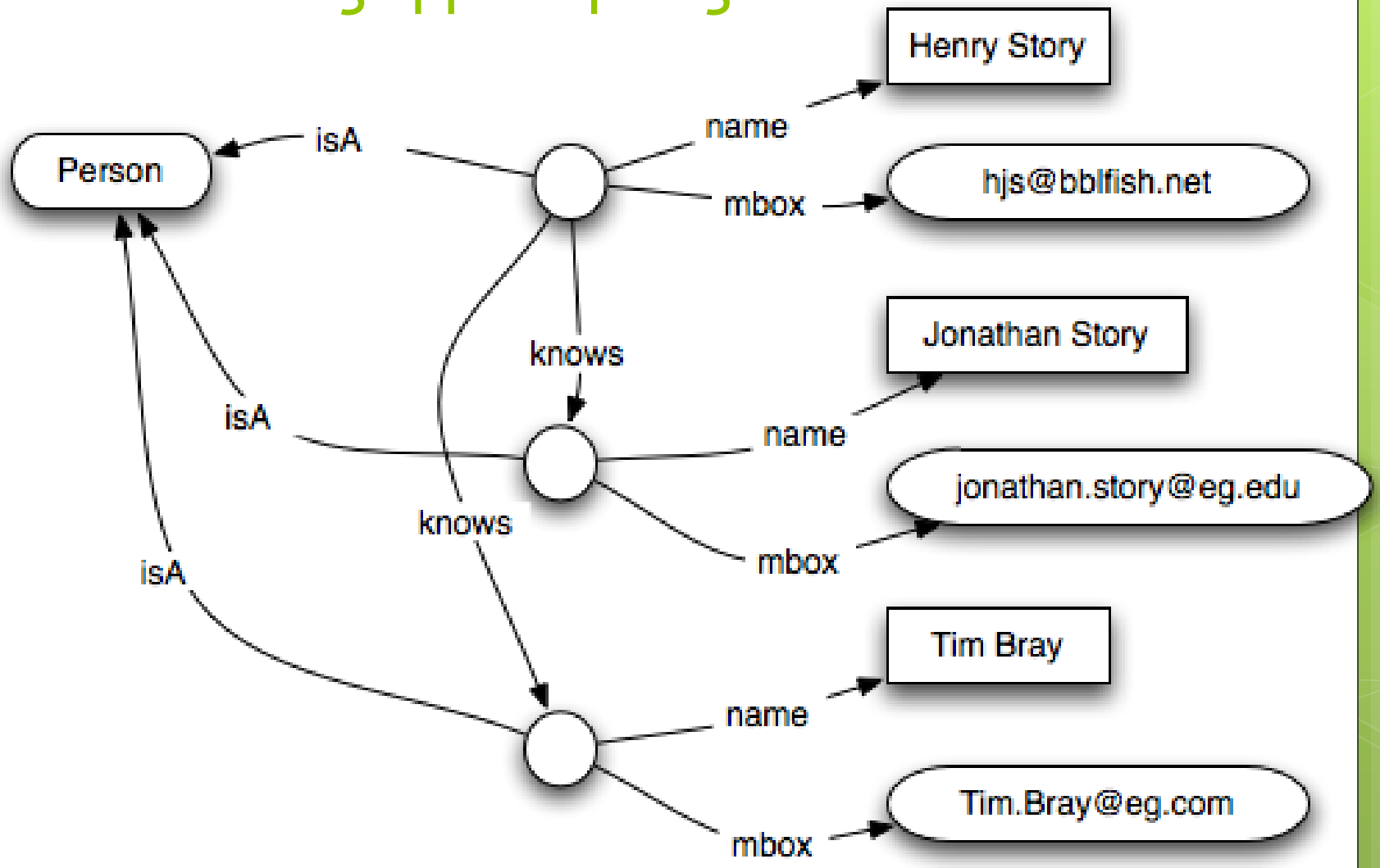
# Παράδειγμα1(απλό σε XML)

```
<Person>
      <name>Henry Story</name>
      <mbox>hs@bblfish.net</mbox>
      <knows>
          <Person>
                    <name>Tim Bray</name>
                              <mbox>tb@eg.com</mbox>
          </Person>
          <Person>
                    <name>Jonathan Story</name>
                              <mbox>js@eg.edu</mbox>
          </Person>
     </knows>
</Person>
```

# Το ίδιο σε αναπαράσταση Turtle

```
[  a     :Person;
   :name "Henry Story";
   :mbox <mailto:hs@insead.edu>;
         :knows [          a :Person;
                           :name "Tim Bray";
                           :mbox <mailto:tb@eg.com
   ];
         :knows [ a :Person;
                           :name "Jonathan Story";
                           :mbox <mailto:js@eg.edu> ];
   ] .
```

# Το ίδιο ως γράφος



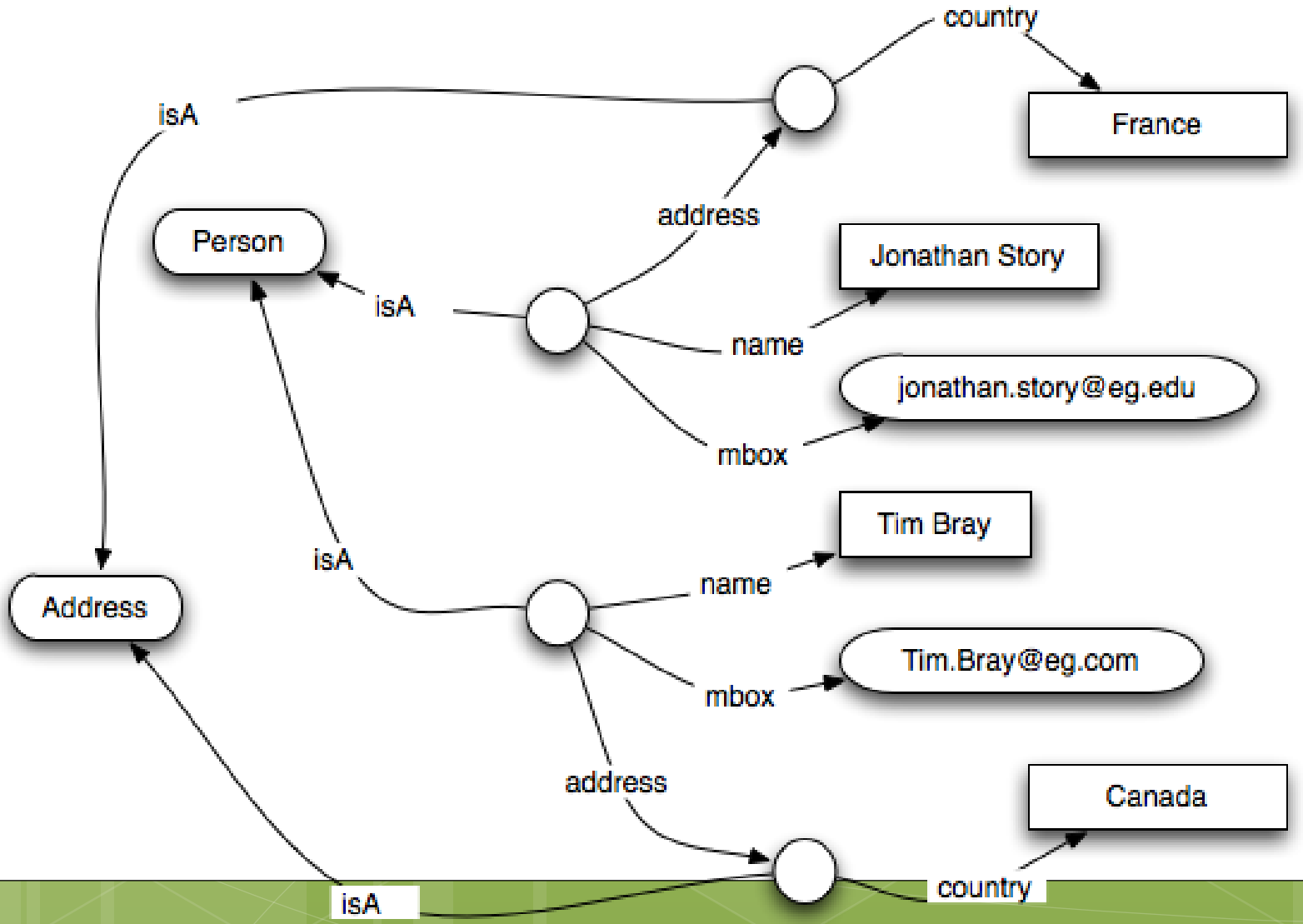
# Παράδειγμα 2 (σε XML)

```
<AddressBook>
    <Person>
            <name>Jonathan Story</name>
            <mbox>Jonathan.Story@eg.edu</mbox>
            <address>
                    <Country>France</Country>
            </address>
    </Person>
    <Person>
            <name>Tim Bray</name>
            <mbox>Tim.Bray@eg.Com</mbox>
            <address>
                    <Country>Canada</Country>
            </address>
    </Person>
 </AddressBook>
```

# Παράδειγμα 2 (σε SPARQL)

```
[       a :Person;
        :name "Tim Bray";
        :mbox <mailto:Tim.Bray@eg.com>
                :address [      a :Address;
                                :country "Canada"@en
    ]   ].
[       a :Person;
        :name "Jonathan Story";
        :mbox <mailto:Jonathan.Story@eg.edu>
        :address [
                a :Address;
                :country "France"@en ]
].
```

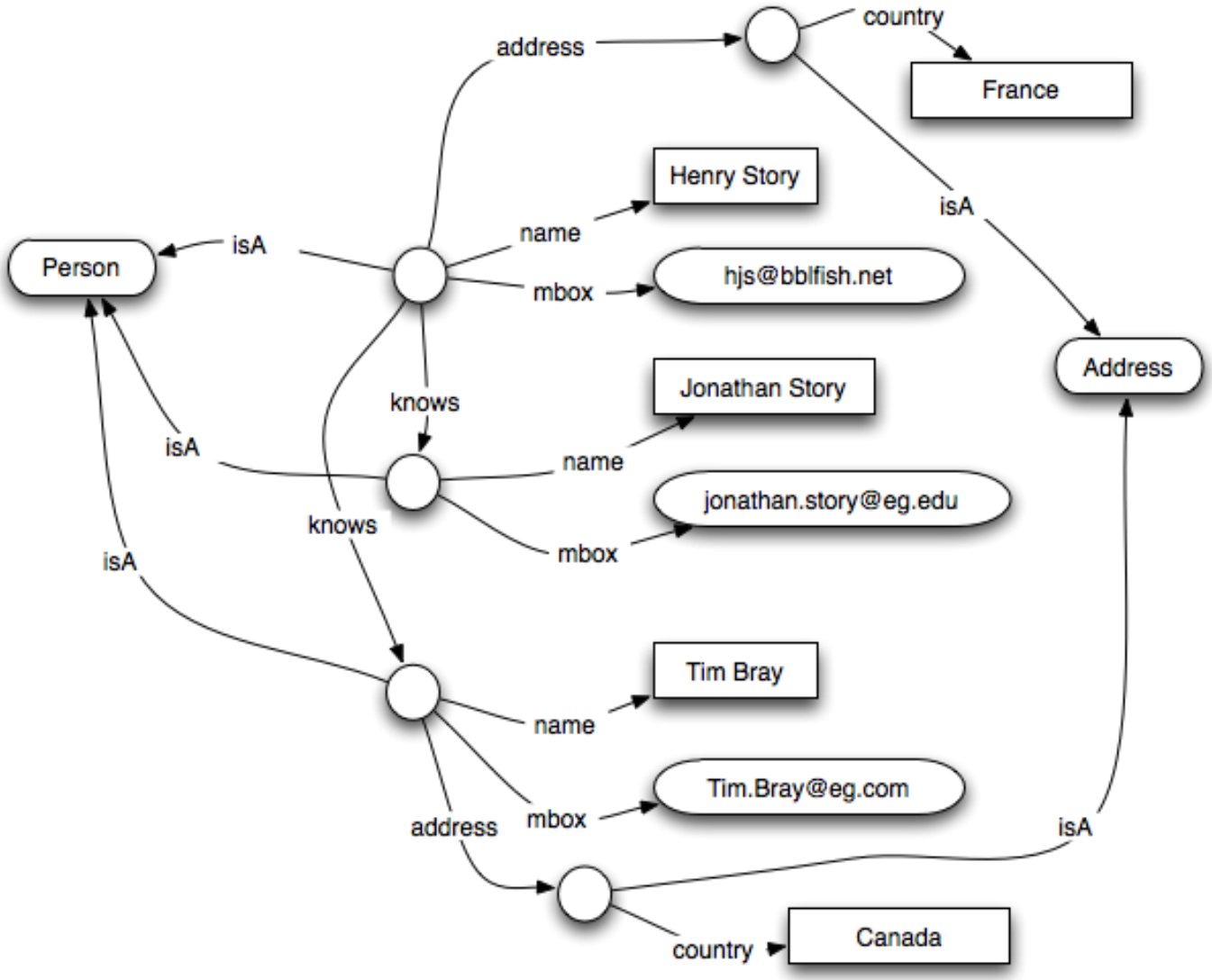
# Παράδειγμα 2 (σε γράφο)



# Επισήμανση!

- Οι 2 γράφοι μπορούν να συνενωθούν στον ακόλουθο...

# Αποτέλεσμα συνένωσης



# Παράδειγμα SPARQL

***Ποιον/ποιούς ξέρει ο Henry που ζει στον Καναδά και ποιο το email του;***

Μπορεί να απαντηθεί μόνο με SPARQL μιας και οι γλώσσες επερώτησης σε XML δουλεύουν μόνο τοπικά σε κάθε έγγραφο.

# Απάντηση σε SPARQL

```
SELECT ?name ?mail
WHERE {
        [a :Person;
        :name "Henry Story";
        :knows [        :name ?name;
                        :mbox ?mail;
                        :address [      a :Address;
                                        :country
  "Canada"@en;
        ] ] ].
 }
```