

Tecnologías para la Web Semántica

Ingeniería Ontológica II
Lenguajes de consultas

RDQL

RDQL (RDF Data Query Language)

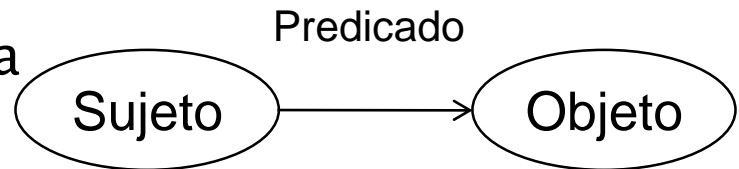
- ▶ Lenguaje de consulta del estilo SQL para consultar sobre las tripletas RDF.
- ▶ Permite especificar patrones que son contrastados con las tripletas del modelo para retornar un resultado.
- ▶ RDQL se utiliza para verificar si las preguntas de competencia se pueden responder con la ontología diseñada.

RDQL: RDF Data Query Language

► Sintaxis RDQL básica:

SELECT vars

Especifica la variable a ser retornada



FROM documents

Indica la fuente RDF a ser consultada

Especifica el modelo por URI

WHERE Expressions

Es la parte más importante de la expresión RDQL. Indica las restricciones de las tripletas RDF (sujeto, predicado, objeto)

RDQL: RDF Data Query Language

AND Filters

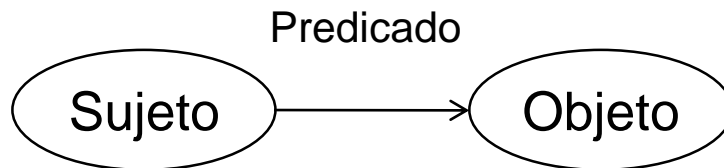
Especifica expresiones booleanas

Indica restricciones que las variables RDQL deben seguir

USING Namespace declarations

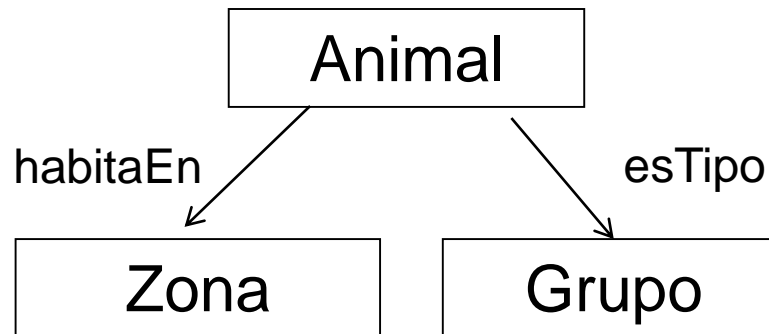
Declara todos los espacios de nombre

Mecanismo de abreviación para URIs a través de la definición de prefijos



RDQL – Ejemplo

¿Dónde habitan los leones?



```
SELECT ?x  
WHERE (<animal:leon>, <animal:habitaEn>, ?x)  
FROM <animal.rdf>  
USING animal FOR <http://www.owl-ontologies.com/unnamed.owl>
```

Cláusulas en profundidad: SELECT

- ▶ **SELECT** permite indicar qué variables RDQL se necesita que la consulta retorne,
- ▶ Si se usa SELECT ?x,?y,?z se recibirá un arreglo de tuplas conteniendo valores para ?x,?y y ?z.
- ▶ Se pueden utilizar otras variables en la consulta como ?a1,?p,?foo pero no van a tener devolución dado que no están presentes en la parte select de la consulta.

```
SELECT ?x
WHERE (<animal:leon>, <animal:habitaEn>, ?x)
FROM <animal.rdf>
USING animal FOR <http://www.owl-ontologies.com/unnamed.owl>
```

Cláusulas en profundidad: FROM

- ▶ **FROM** indica las fuentes RDF que son consultadas, cada fuente se encierra por *angle brackets* (<&>).
- ▶ Si se incluyen más de una fuente, se separa por comas.

```
SELECT ?x  
WHERE (<animal:leon>, <animal:habitaEn>, ?x)  
FROM <animal.rdf>  
USING animal FOR <http://www.owl-ontologies.com/unnamed.owl>
```

Cláusulas en profundidad: WHERE

- ▶ **WHERE** es la parte más importante de la expresión RDQL. En esta parte se indican restricciones que las tripletas RDF (sujeto, predicado, objeto) deben cumplir para que sean retornadas.
- ▶ Se expresa por una lista de restricciones separadas por comas.
- ▶ Cada restricción toma la forma: (sujeto, predicado, objeto) donde el sujeto, predicado y objeto pueden ser un valor literal o una variable RDQL.

Descripción de la forma del grafo

```
SELECT ?x
WHERE (<animal:leon>, <animal:habitaEn>, ?x)
FROM <animal.rdf>
USING animal FOR <http://www.owl-ontologies.com/unnamed.owl>
```


Cláusulas en profundidad: WHERE

Para el predicado se pueden expresar nombres de propiedades utilizando un espacio de nombres declarado en la sección **USING**.

Por ejemplo:

<dc:name> que indica que el predicado debe equiparar a "name" nombre local para el espacio de nombre declarado como "dc" en la parte USING.

```
SELECT ?x
WHERE (<animal:leon>, <animal:habitaEn>, ?x)
FROM <animal.rdf>
USING animal FOR <http://www.owl-ontologies.com/unnamed.owl>
```

Cláusulas en profundidad: USING

- ▶ **USING** declara todos los espacios de nombre que serán usados para las propiedades RDF
- ▶ Las declaraciones se separan por comas

```
SELECT ?x  
WHERE (<animal:leon>, <animal:habitaEn>, ?x)  
FROM <animal.rdf>  
USING animal FOR <http://www.owl-ontologies.com/unnamed.owl>
```

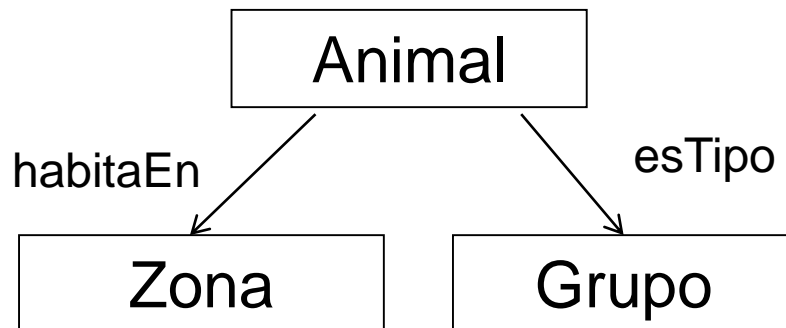
Cláusulas en profundidad: AND

- ▶ **AND** indica restricciones que las variables RDQL deben cumplir.

RDQL – Ejemplo

¿Cuántos mamíferos hay?

```
SELECT ?x ?grupo
WHERE (?x, <animal:esTipo>, ?grupo)
AND (?grupo, <rdf:type>, <animal:mamifero>)
FROM <animal.rdf>
USING rdf FOR <http://www.w3.org/1999/02/22-rdf-syntax-ns>
animal FOR http://www.owl-ontologies.com/unnamed.owl
```



Where: indica restricciones para las tripletas RDF (sujeto, predicado, objeto)

AND: indica restricciones que las variables RDQL deben cumplir

Implementación– Jena

Framework desarrollado por HP Labs para manipular metadatos desde una aplicación Java.

- ▶ Dos versiones:
 - JENA 1
 - Principalmente soporte para RDF.
 - Capacidades de razonamiento limitadas.
 - JENA 2
 - Incluye además una API para el manejo de Ontologías.
 - Soporta el lenguaje OWL.

Implementación– Jena

- ▶ Incluye varios componentes:
 - ARP: Un Parser de RDF.
 - API RDF.
 - API de Ontologías con soporte para OWL, DAML y RDF Schema.
 - Subsistema de Razonamiento.
 - Soporte para Persistencia.
 - RDQL: Lenguaje de consultas de RDF.

Implementación– Jena

- ▶ Es posible ejecutar consultas RDQL desde una aplicación Java.
- ▶ Se utilizan las siguientes clases:
 - Query – La consulta misma
 - QueryExecution – la interface del algoritmo de ejecución
 - QueryEngine – algoritmo de ejecución local
 - QueryResults – el iterador de resultados
 - ResultBinding – colección de variable bindings

API RDF de Jena

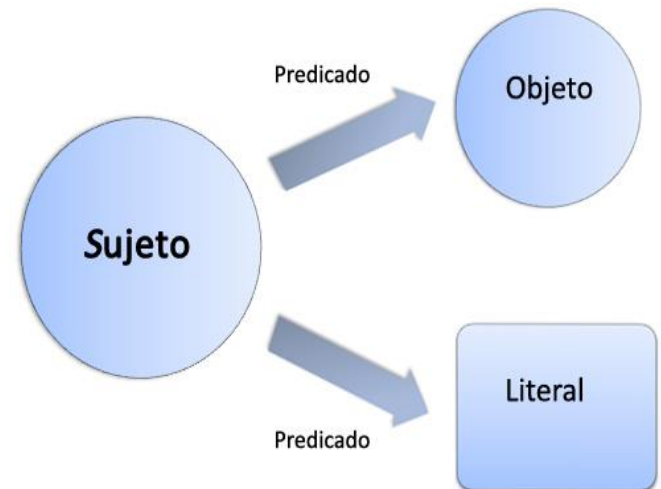
- ▶ Permite crear y manipular modelos RDF desde una aplicación Java.
- ▶ Proporciona clases java para representar:
 - Modelos.
 - Recursos.
 - Propiedades.
 - Literales.
 - Declaraciones.

RDF

- ▶ **Recurso:**
 - Todo aquello que se puede describir por una expresión RDF.
- ▶ **Propiedad:**
 - Una característica, atributo o relación usada para describir un recurso.
- ▶ **Literal:**
 - Un tipo de datos simple (String, Integer, etc).
- ▶ **Declaración:**
 - Un recurso junto con una propiedad y con un valor asociado.

Declaraciones

- ▶ Cada declaración tiene tres partes:
 - Sujeto, Predicado y Objeto.



- ▶ Un modelo RDF es un conjunto de

Implementación– Jena

```
Model model = ModelFactory.createDefaultModel() ;  
model.read(new FileInputStream("Animal.owl"),
```

```
    "http://protege.stanford.edu/", "RDF/XML") ;  
String queryString = "SELECT ?x "+  
"WHERE (<animal:leon>, <animal:habitaEn>, ?x) "+  
"USING animal FOR <http://www.owl-ontologies.com/unnamed.owl>";
```

```
Query query = new Query(queryString) ;  
query.setSource(model);
```

```
QueryExecution qe = new QueryEngine(query) ;
```

```
QueryResults results = qe.exec() ;
```

```
System.out.println("");
```

```
System.out.println("¿Dónde habitan los leones?");
```

```
for ( Iterator iter = results ; iter.hasNext() ; )
```

```
{  
    ResultBinding res = (ResultBinding)iter.next() ;
```

```
    Object x = res.get("x") ;
```

```
    System.out.println(x) ;
```

```
}
```

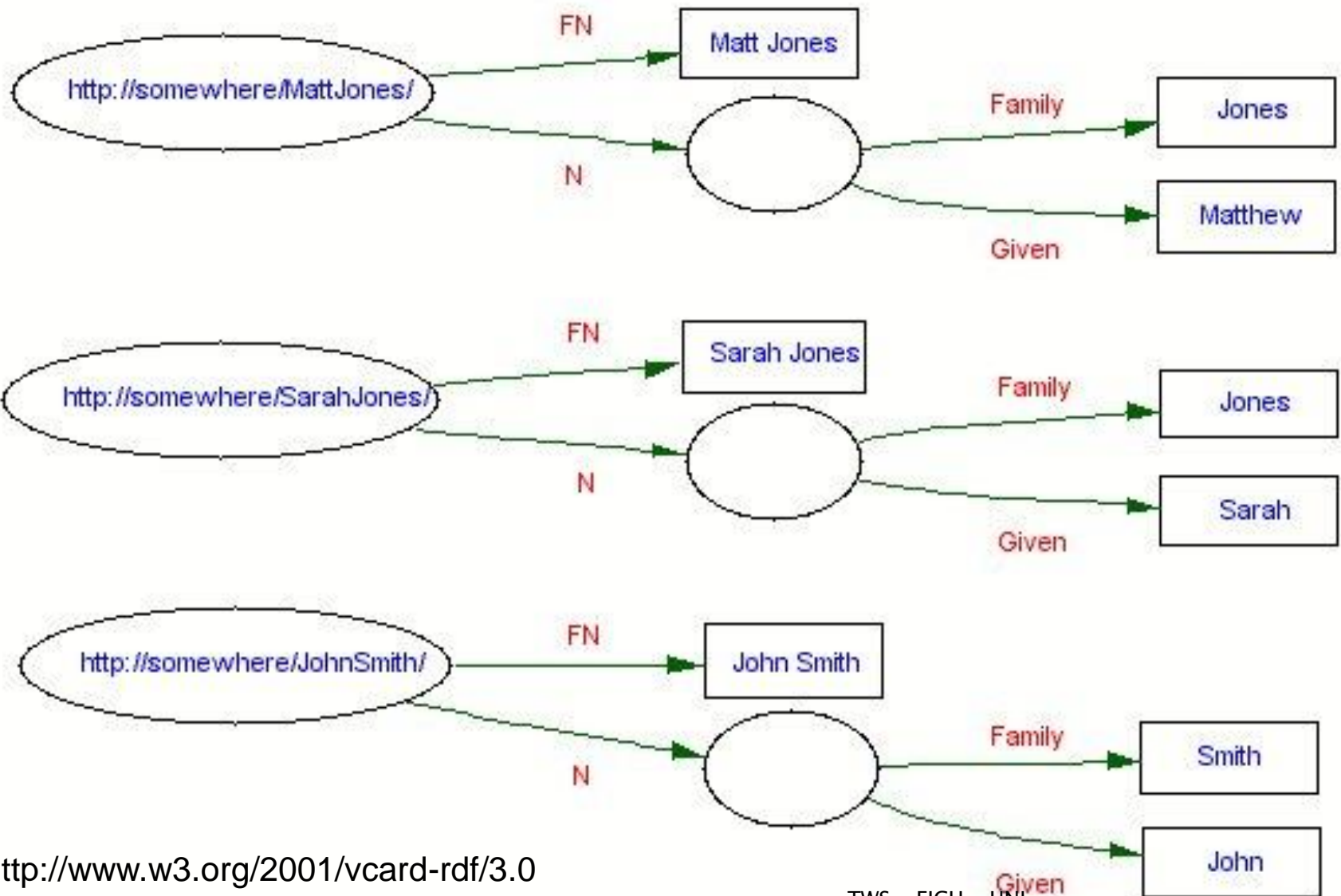
```
results.close() ;
```

Una consulta se crea pasando un string a la clase Query

Los resultados se devuelven con un iterador

Se imprime cada resultado como una línea simple

Ejemplo



Ejercicio 1

Retornar todos los recursos que tienen la propiedad FN con valor “John Smith”

Ejercicio 1

Q1: Retornar todos los recursos que tienen la propiedad FN con valor “John Smith”:

```
SELECT ?x
WHERE (?x <http://www.w3.org/2001/vcard-rdf/3.0#FN> "John Smith")
```

Ejecutar q1 con un modelo m1.rdf:

```
java jena.rdfquery --data m1.rdf --query q1
```

La salida es:

```
x
=====
<http://somewhere/JohnSmith/>
```

Ejercicio 2

Retornar todos los recursos que tienen la propiedad FN y su valor asociado:

Ejercicio 2

Retornar todos los recursos que tienen la propiedad FN y su valor asociado:

```
SELECT ?x, ?fname
WHERE (?x <http://www.w3.org/2001/vcard-rdf/3.0#FN> ?fname)
```

La salida es:

```
      x
      | fname
=====
http://somewhere/JohnSmith/           | "John Smith"
<http://somewhere/RebeccaSmith/>      | "Becky Smith"
<http://somewhere/SarahJones/>        | "Sarah Jones"
http://somewhere/MattJones/           | "Matt Jones"
```


Ejercicio 2

Retornar todos los recursos que tienen la propiedad FN y su valor asociado:

```
SELECT ?x, ?fname  
WHERE (?x,<vcard:FN>, ?fname)  
FROM vcard.rdf  
USING vcard FOR <http://www.w3.org/2001/vcard-rdf/3.0#N>
```

Ejemplo

```
SELECT ?x, ?fname  
WHERE (?x <http://www.w3.org/2001/vcard-rdf/3.0#FN> ?fname)
```

```
Query query = new Query("SELECT...") ;  
query.setSource(model);  
QueryExecution qe = new QueryEngine(query) ;  
QueryResults results = qe.exec();  
  
for (Iterator iter = results; iter.hasNext();)  
{  
    ResultBinding res = (ResultBinding) iter.next();  
    Resource x = (Resource) res.get("x");  
    Literal fname = (Literal) res.get("fname");  
    System.out.println("x: " + x + " fname: " + fname);  
}
```

Ejercicio 3

- a) Retornar los nombres de los Jones
- b) Retornar los recursos y su nombre (givenName)

Ejercicio 3

Retornar los nombres de los Jones:

```
SELECT ?givenName
WHERE (?y <http://www.w3.org/2001/vcard-rdf/3.0#Family> "Jones"),
      (?y <http://www.w3.org/2001/vcard-rdf/3.0#Given> ?givenName)
```

La salida es:

```
givenName
=====
"Matthew"
"Sarah"
```

Crear un modelo RDF

```
//some definitions
String personURI    = "http://somewhere/JohnSmith";
String givenName     = "John";
String familyName    = "Smith";
String fullName      = givenName + " " + familyName;

//create an empty model
Model model = ModelFactory.createDefaultModel();

//create the resource
Resource johnSmith = model.createResource(personURI);

//add the property
johnSmith.addProperty(VCARD.FN, fullName);

johnSmith.addProperty(VCARD.N, model.createResource()
    .addProperty(VCARD.Given, givenName)
    .addProperty(VCARD.Family, familyName));
```

Escribir y Leer un modelo

- ▶ Para serializar el modelo a XML:

```
model.write(System.out);
```

- ▶ Para cargar un modelo en memoria:

```
Model model = ModelFactory.createDefaultModel();
```

```
InputStream in = Tutorial05.class.getClassLoader().  
    getResourceAsStream(ejemplo.owl);
```

```
model.read(new InputStreamReader(in), "");
```

Navegar un modelo

► A partir de la URI de un recurso:

```
// recuperar el recurso John Smith
String johnSmithURI = "http://somewhere/JohnSmith";
Resource jSmith = model.getResource(johnSmithURI);

// recuperar el valor de la propiedad N
Resource name = (Resource) jSmith.getProperty(VCARD.N)
                                                         .getObject();

// recuperar el valor de la propiedad FN
String fullName = (String) jSmith.getProperty(VCARD.FN)
                                                         .getObject();
```

Consultar un modelo

► Buscar información en un modelo:

```
// recuperar todos los recursos de tipo vcard (asumiendo que
// solo ellos tienen la propiedad FN y que todos la tienen)
ResIterator it = model.listSubjectsWithProperty(VCARD.FN);
while (it.hasNext()) {
    Resource r = it.nextResource();
    System.out.println(r);
}
```

► Consultas más avanzadas:

- Utilizar la primitiva `listStatements(Selector s)`.
- Utilizar el lenguaje de consulta RDQL.

Operaciones sobre modelos

- ▶ Los modelos son conjuntos de declaraciones.
- ▶ Podemos realizar las siguientes operaciones:
 - Unión.
 - Intersección.
 - Diferencia.

```
// leer los modelos de los archivos RDF
model1.read(new InputStreamReader(in1), "");
model2.read(new InputStreamReader(in2), "");
```

```
// unir los modelos
Model model = model1.union(model2);
```

Ejemplo

- ▶ q1 contiene la consulta:

```
SELECT ?x
WHERE (?x <http://www.w3.org/2001/vcard-rdf/3.0#FN> "John Smith")
```

- ▶ Para ejecutar q1 sobre el modelo m1.rdf:

```
java jena.rdfquery --data m1.rdf --query q1
```

- ▶ La salida es:

```
x
=====
<http://somewhere/JohnSmith/>
```

SPARQL

Active Ontology
Entities
Classes
Object Properties
Data Properties
Individuals
OWL Viz
DL Query
SPARQL Query

SPARQL query:

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX aonet: <http://www.owl-ontologies.com/Ontology1328898742.owl#>
SELECT ?activity ?reactive ?instrument ?tipo
WHERE {
    ?assessment aonet:isComposedBy ?activity.
    ?activity aonet:isComposedByReactive ?reactive.
    ?reactive aonet:uses ?instrument.
    ?instrument rdf:type ?tipo
    FILTER regex(str(?tipo), 'SimpleChoice')}

```

activity	reactive	instrument	tipo
componentes_modelo_simulacion_activity	componente_modelo_simulacion_reactive	componente_modelo_simulacion_mch	SimpleChoice
modelo_DEVS_clasico_acoplado_activity	modelo_DEV_clasico_reactive	modelo_DEV_clasico_sch	SimpleChoice
ciclo_de_simulacion_DEVS_activity	ciclo_de_simulacion_DEVS_reactive	ciclo_de_simulacion_DEVS_sch	SimpleChoice

SPARQL

- ▶ Similar a SQL para RDF
- ▶ Lenguaje de consultas Basado en RDQL
- ▶ Modelo = patrones sobre grafos

SPARQL

- ▶ Conjunto de patrones triples
 - Patrón triple: similar a una tripleta RDF (sujeto, predicado, objeto) pero cualquier componente puede ser una variable de consulta; se permiten sujetos literales.
 - Macheo de un patrón triple a un grafo: vínculos entre variables y términos RDF.

```
SELECT ?x ?v WHERE { ?x ?x ?v }
```

rdf:type rdf:type rdf:Property

x	v
rdf:type	rdf:Property

SPARQL

Data @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> .

PREFIX foaf: <http://xmlns.com/foaf/0.1/> Query

SELECT ?name ?mbox

WHERE

{ ?x foaf:name ?name .

 ?x foaf:mbox ?mbox }

Query Result

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

SPARQL

Multiple Optional Blocks

@prefix foaf: <http://xmlns.com/foaf/0.1/> .

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

 @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

 _:a foaf:name "Alice" .

 _:a foaf:homepage <http://work.example.org/alice/> .

 _:b foaf:name "Bob" .

 _:b foaf:mbox <mailto:bob@work.example> .

PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?name ?mbox ?hpage

Query

WHERE { ?x foaf:name ?name .

 OPTIONAL { ?x foaf:mbox ?mbox }.

OPTIONAL { ?x foaf:homepage ?hpage } }

Query Result

name	Mbox	hpage
"Alice"		<http://work.example.org/alice/>
"Bob"	<mailto:bob@example.com>	

SPARQL

Alternative Graph Patterns

Data @prefix dc10: <http://purl.org/dc/elements/1.0/> .
 @prefix dc11: <http://purl.org/dc/elements/1.1/> .
 _:a dc10:title "SPARQL Query Language Tutorial" .
 _:b dc11:title "SPARQL Protocol Tutorial" .
 _:c dc10:title "SPARQL" .
 _:c dc11:title "SPARQL (updated)" .

PREFIX dc10: <http://purl.org/dc/elements/1.0/>

PREFIX dc11: <http://purl.org/dc/elements/1.1/>

SELECT ?x ?y

WHERE { { ?book dc10:title ?x } UNION { ?book dc11:title ?y } }

Query

Query Result

x	y
	"SPARQL (updated)"
	"SPARQL Protocol Tutorial"
"SPARQL"	
"SPARQL Query Language Tutorial"	

Relationship between Named and Background Graphs (I)

Background graph

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .  
<http://example.org/bob> dc:publisher "Bob" .  
<http://example.org/alice> dc:publisher "Alice" .
```

Graph: <http://example.org/bob>

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .  
_:a foaf:name "Bob" .  
_:a foaf:mbox <mailto:bob@oldcorp.example.org> .
```

Graph: <http://example.org/alice>

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .  
_:a foaf:name "Alice" .  
_:a foaf:mbox <mailto:alice@work.example.org>  
.
```

Relationship between Named and Background Graphs (II)

Background graph

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .  
_:x foaf:name "Bob" .  
_:x foaf:mbox <mailto:bob@oldcorp.example.org> .  
_:y foaf:name "Alice" .  
_:y foaf:mbox <mailto:alice@work.example.org> .
```

Graph: <http://example.org/bob>

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .  
_:a foaf:name "Bob" .  
_:a foaf:mbox <mailto:bob@oldcorp.example.org> .
```

Graph: <http://example.org/alice>

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .  
_:a foaf:name "Alice" .  
_:a foaf:mbox <mailto:alice@work.example.org> .
```

Consulta Dataset

Graph: <http://example.org/foaf/aliceFoaf>

@prefix foaf: <<http://xmlns.com/foaf/0.1/>> .

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

@prefix rdfs: <<http://www.w3.org/2000/01/rdf-schema#>> .

_:a foaf:name "Alice" .

_:a foaf:mbox <mailto:alice@work.example> .

_:a foaf:knows _:b .

_:b rdfs:seeAlso <<http://example.org/foaf/bobFoaf>> .

<<http://example.org/foaf/bobFoaf>> rdf:type foaf:PersonalProfileDocument .

_:b foaf:name "Bob" .

_:b foaf:mbox <mailto:bob@work.example> .

_:b foaf:age 32 .

Graph: <http://example.org/foaf/bobFoaf>

@prefix foaf: <<http://xmlns.com/foaf/0.1/>> .

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

@prefix rdfs: <<http://www.w3.org/2000/01/rdf-schema#>> .

_:1 foaf:mbox <mailto:bob@work.example> .

_:1 rdfs:seeAlso <<http://example.org/foaf/bobFoaf>> .

_:1 foaf:age 35 .

<<http://example.org/foaf/bobFoaf>> rdf:type foaf:PersonalProfileDocument .

Consulta Dataset

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?src ?bobAge
WHERE { GRAPH ?src
        { ?x foaf:mbox <mailto:bob@work.example>
          .
          ?x foaf:age ?bobAge }
      }
```

src	bobAge
<http://example.org/foaf/aliceFoaf>	32
<http://example.org/foaf/bobFoaf>	35

Consulta Dataset

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX data: <http://example.org/foaf/>
SELECT ?age
WHERE
{
    GRAPH data:bobFoaf {
        ?x foaf:mbox <mailto:bob@work.example> .
        ?x foaf:age ?age }
}
```

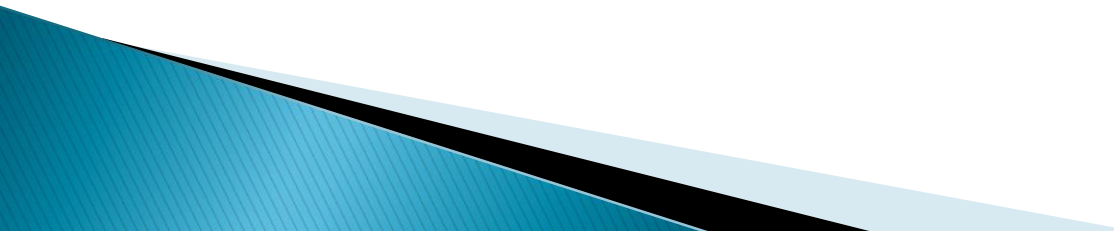
age
35

Consulta Dataset

```
PREFIX data: <http://example.org/foaf/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?mbox ?age ?ppd
WHERE
{ GRAPH data:aliceFoaf
  { ?alice foaf:mbox <mailto:alice@work.example> ;
    foaf:knows ?whom .
    ?whom foaf:mbox ?mbox ;
    rdfs:seeAlso ?ppd .
    ?ppd a foaf:PersonalProfileDocument . } .
  GRAPH ?ppd { ?w foaf:mbox ?mbox ;
    foaf:age ?age } }
```

mbox	age	ppd
<mailto:bob@work.example>	35	<http://example.org/foaf/bobFoaf>

Consulta:

- **SELECT**
 - returns all, or a subset of the variables bound in a query pattern match
 - formats : XML or RDF/XML
 - **CONSTRUCT**
 - returns an RDF graph constructed by substituting variables in a set of triple templates
 - **DESCRIBE**
 - returns an RDF graph that describes the resources found.
 - **ASK**
 - returns whether a query pattern matches or not.
- 

CONSTRUCT Ejemplos(I)

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
```

```
_:a foaf:name "Alice" .
```

```
_:a foaf:mbox <mailto:alice@example.org> .
```

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
```

```
PREFIX vcard: <http://www.w3.org/2001/vcard-rdf/3.0#>
```

```
CONSTRUCT { <http://example.org/person#Alice> vcard:FN  
?name }
```

```
WHERE { ?x foaf:name ?name }
```

```
@prefix vcard: <http://www.w3.org/2001/vcard-rdf/3.0#>.
```

```
<http://example.org/person#Alice> vcard:FN "Alice" .
```

#extracting a whole graph from the target RDF dataset

```
CONSTRUCT { ?s ?p ?o }
```

```
WHERE { GRAPH <http://example.org/myGraph> { ?s ?p ?o } . }
```


CONSTRUCT Ejemplos(II)

accessing a graph conditional on other information contained in the metadata about named graphs in the dataset

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
```

```
PREFIX app: <http://example.org/ns#>
```

```
CONSTRUCT { ?s ?p ?o }
```

```
WHERE { GRAPH ?g { ?s ?p ?o } .
```

```
    { ?g dc:publisher <http://www.w3.org/> } .
```

```
    { ?g dc:date ?date } .
```

```
    FILTER app:myDate(?date) > "2005-02-8T00:00:00Z"^^xsd:dateTime.
```

```
}
```



DESCRIBE

PREFIX ent: <http://myorg.example/employees#>

DESCRIBE ?x

WHERE { ?x ent:employeeId "1234" }

@prefix foaf: <http://xmlns.com/foaf/0.1/> .

@prefix vcard: <http://www.w3.org/2001/vcard-rdf/3.0> .

@prefix myOrg: <http://myorg.example/employees#> .

_:a myOrg:employeeId "1234" ;

foaf:mbox_sha1sum "ABCD1234" ;

vcard:N [vcard:Family "Smith" ;

vcard:Given "John"] .

foaf:mbox_sha1sum rdf:type owl:InverseFunctionalProperty

.

ASK

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .  
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .  
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .  
_:a foaf:name "Alice" .  
_:a foaf:homepage <http://work.example.org/alice/> .  
_:b foaf:name "Bob" .  
_:b foaf:mbox <mailto:bob@work.example> .
```

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>  
ASK { ?x foaf:name "Alice" } .
```