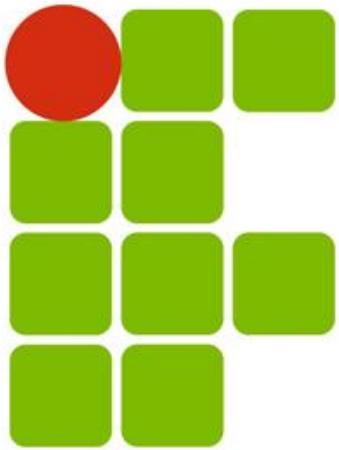


LEDS

Laboratório de Extensão de
Desenvolvimento de Sistemas

Jena

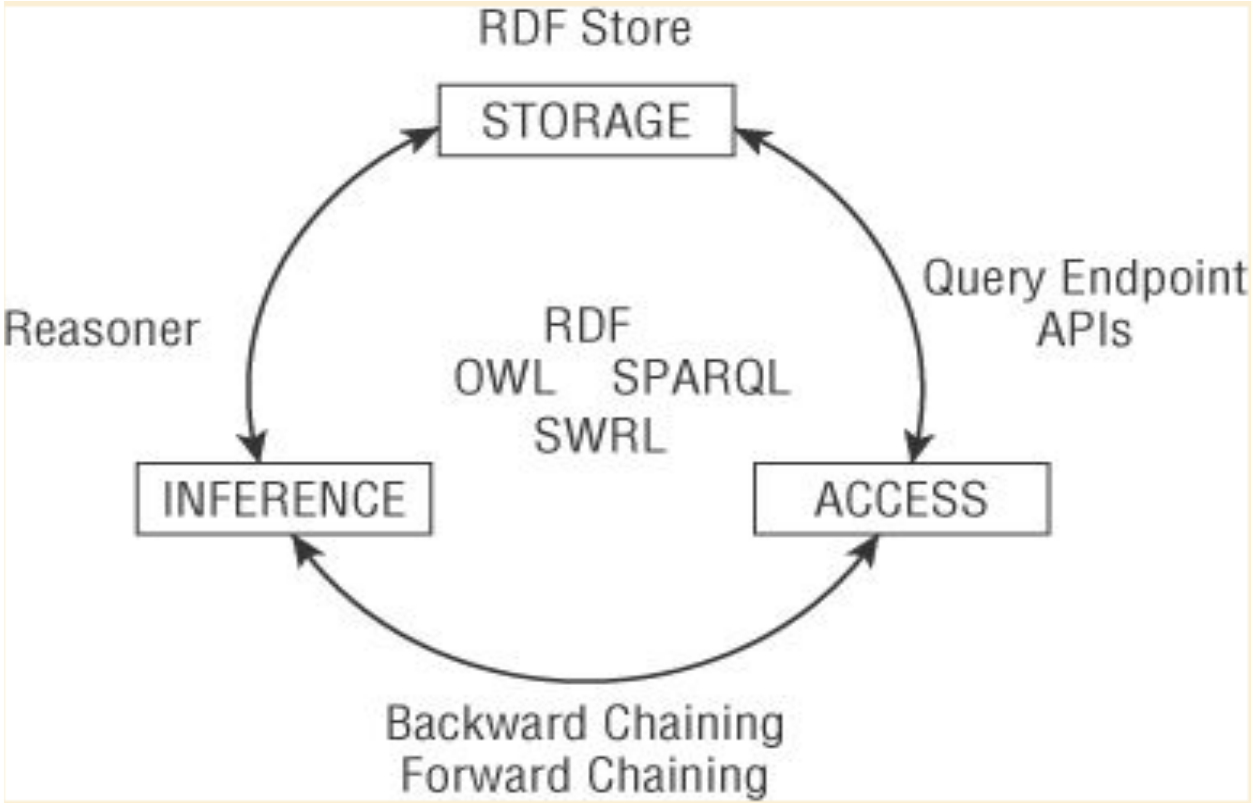
Paulo Sérgio dos Santos Júnior

Agenda

- Semantic Web Framework
- Jena
- RDF
- SPARQL
- Ontologias

Semantic Web Framework

Semantic Web Framework



INSTITUTO FEDERAL
ESPIRITO SANTO

Semantic Web Framework

- Componentes básicos:
  - **Storage (Knowledge base)**: é um repositório de documentos (rdf ou owl);
  - **Access:** responsável por manipular as informações (e.g, processadores de queries);
  - **Inferences (reasoning engines ou resoner):** responsável pela interpretação semântica as informações armazenadas

Semantic Web Framework

- Principais Semantic Web Framework:

Framework	Language/Interface	Semantic Level
Jena	Java	RDF to OWL 2
Sesame	Java & RESTful web service	RDF
OWL API	Java	OWL 2
RAP—RDF API	PHP	RDF
Redland	C, Python, Ruby, Perl, & PHP	RDF
LinqToRDF	.NET	RDF

INSTITUTO FEDERAL
ESPIRITO SANTO

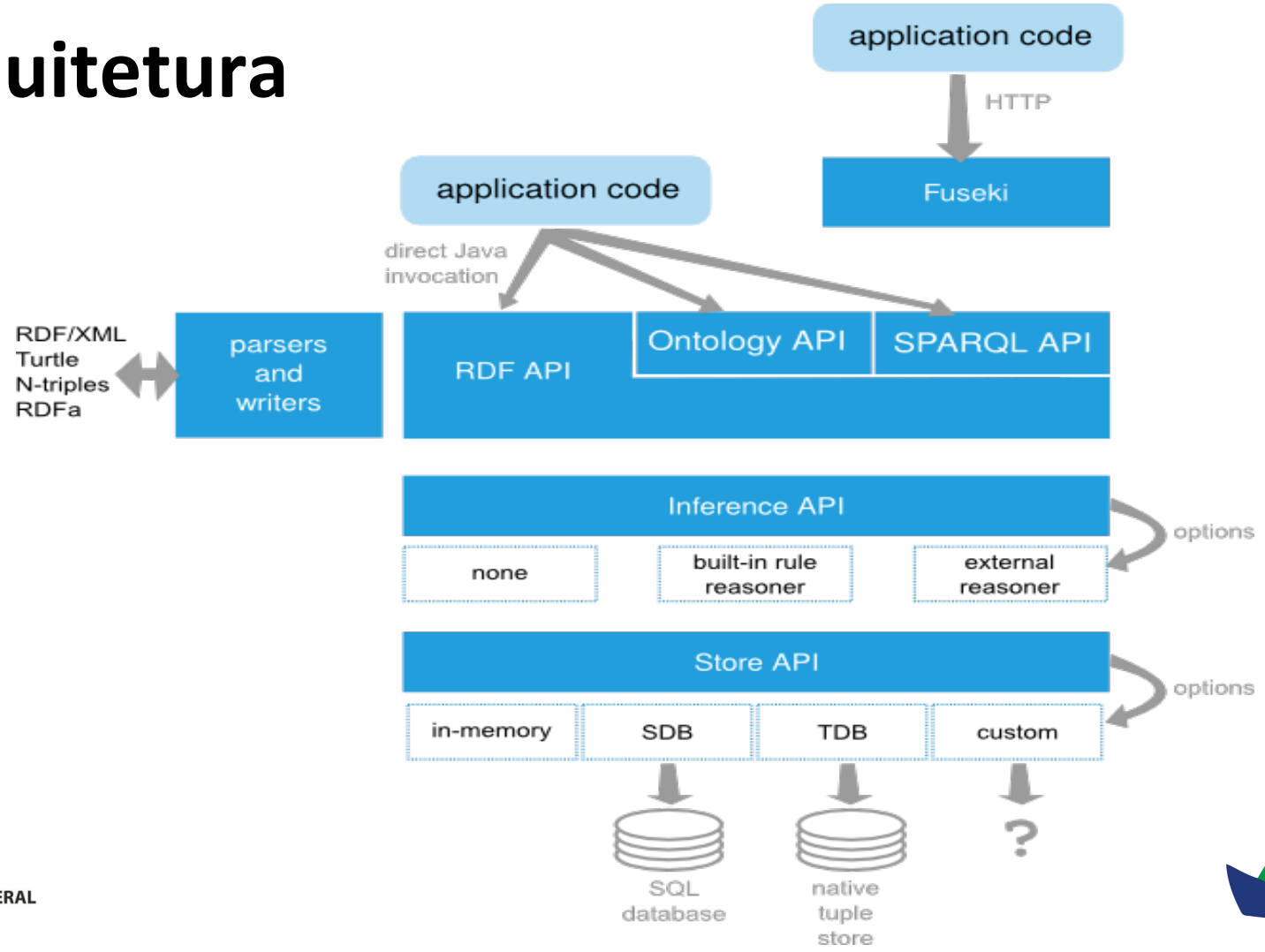
JENA

Jena

- **Introdução:**
  - Desenvolvido no HP Lab em 2000;
  - Criado para o desenvolvimento de aplicações de Linking Data e Semantic Web;
  - Permite minupular dados em OWL e RDF;

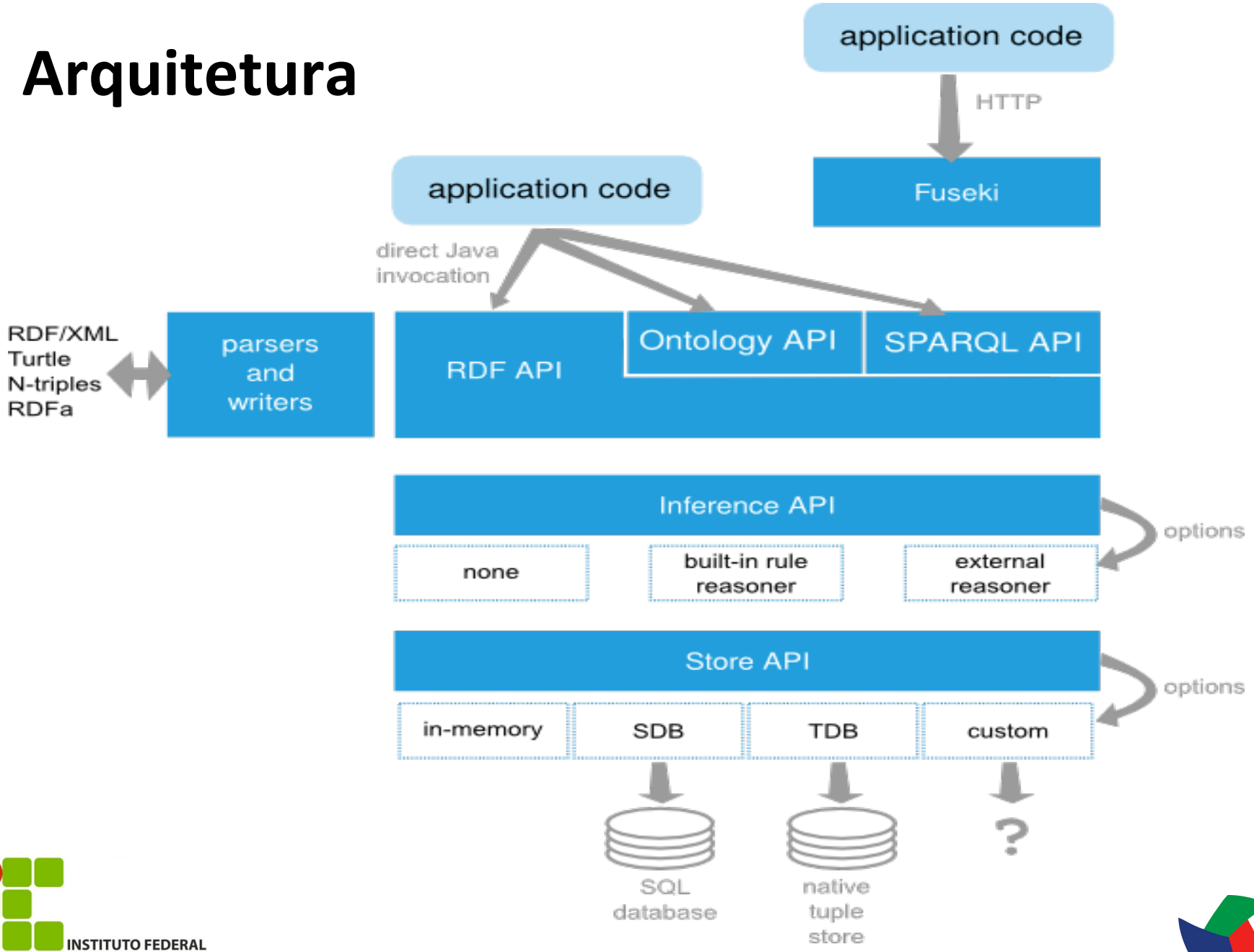
Jena

- **Arquitetura**



INSTITUTO FEDERAL
ESPIRITO SANTO

Arquitetura



INSTITUTO FEDERAL
ESPIRITO SANTO

Jena

- **Componentes da Arquitetura:**
  - **RDF API**: permite criar, ler e manipular dados em RDF;
  - **ARQ**: é uma *engine* de consultas do JENA;
  - **TDB:** permite armazenar os dados. É um banco de dados de triplas nativo do JENA;
  - **Fuseki:** é um "servidor rest-style" que permite criar *end-point;*
  - **Ontology API**: permite manipular RDFS e OWL;
  - **Inference API:** permite realizar inferências sobre os modelos OWL e RDFS;

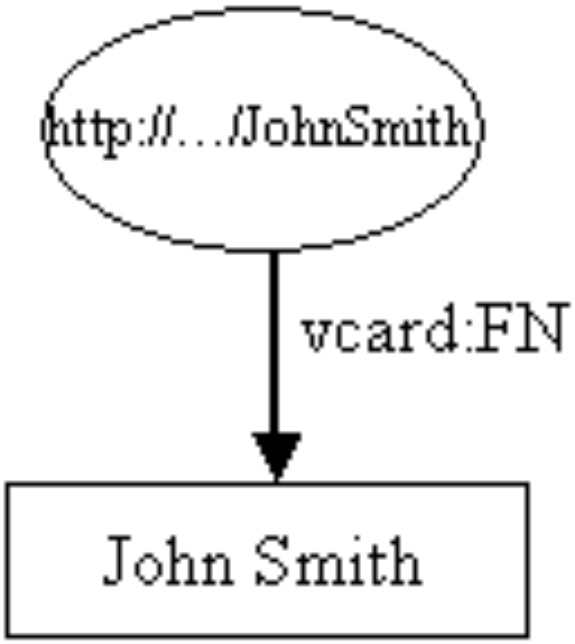
JENA e RDF

Jena e RDF

- **RDF:**
  - *Resource Description Framework* (RDF) é uma recomendação da W3C para descrever recursos;
  - Recursos pode ser qualquer coisas:
    - Uma página web;
    - Um tutorial;
    - Um filme e etc;
  - **Recursos** possuem **propriedades**;
  - Cada **propriedade** possui um **valor**;

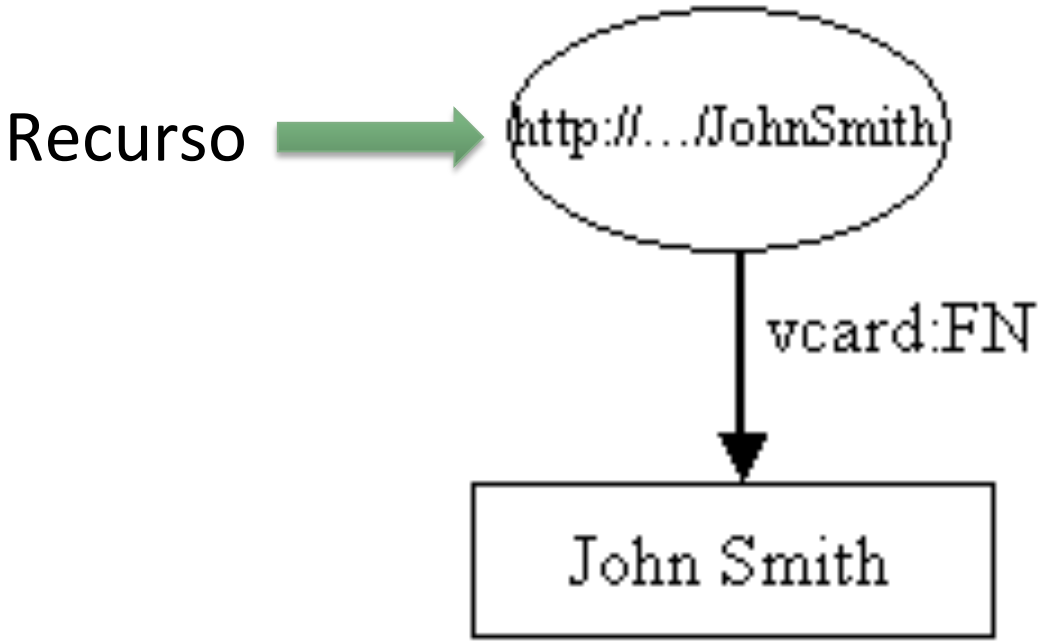
Jena e RDF

- **RDF:**



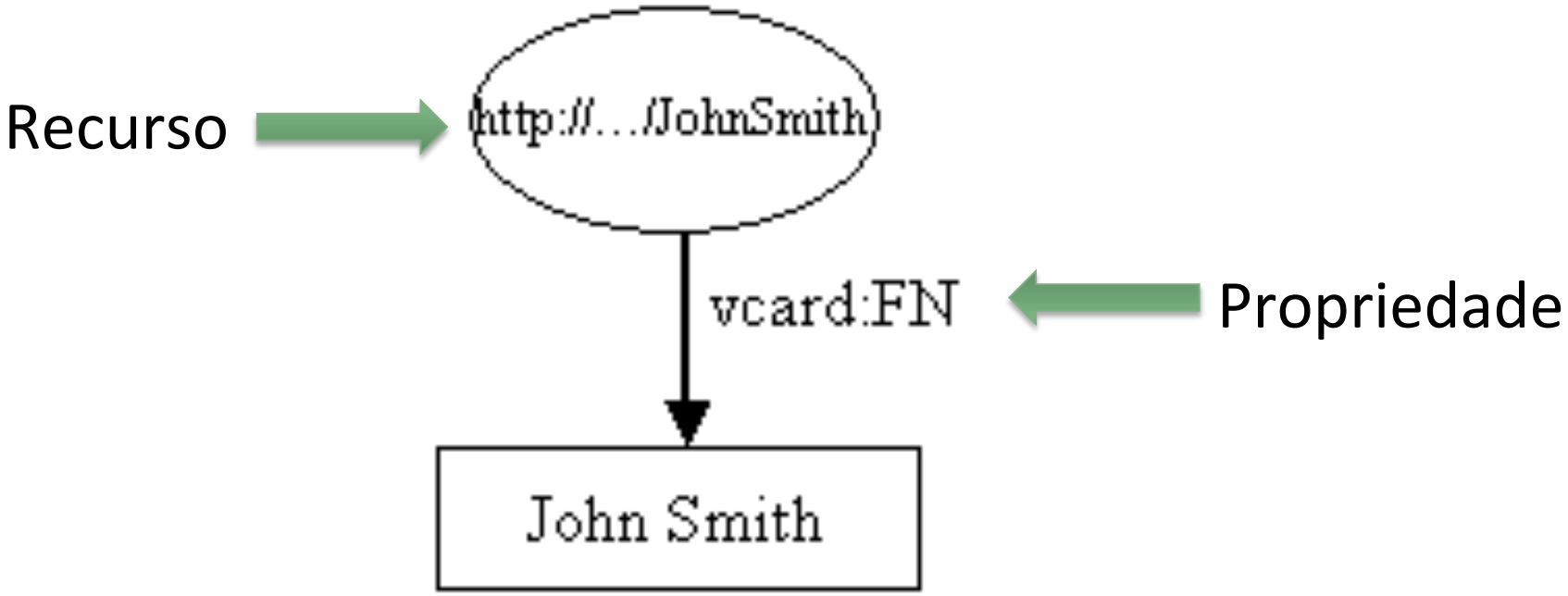
INSTITUTO FEDERAL
ESPIRITO SANTO

Jena e RDF



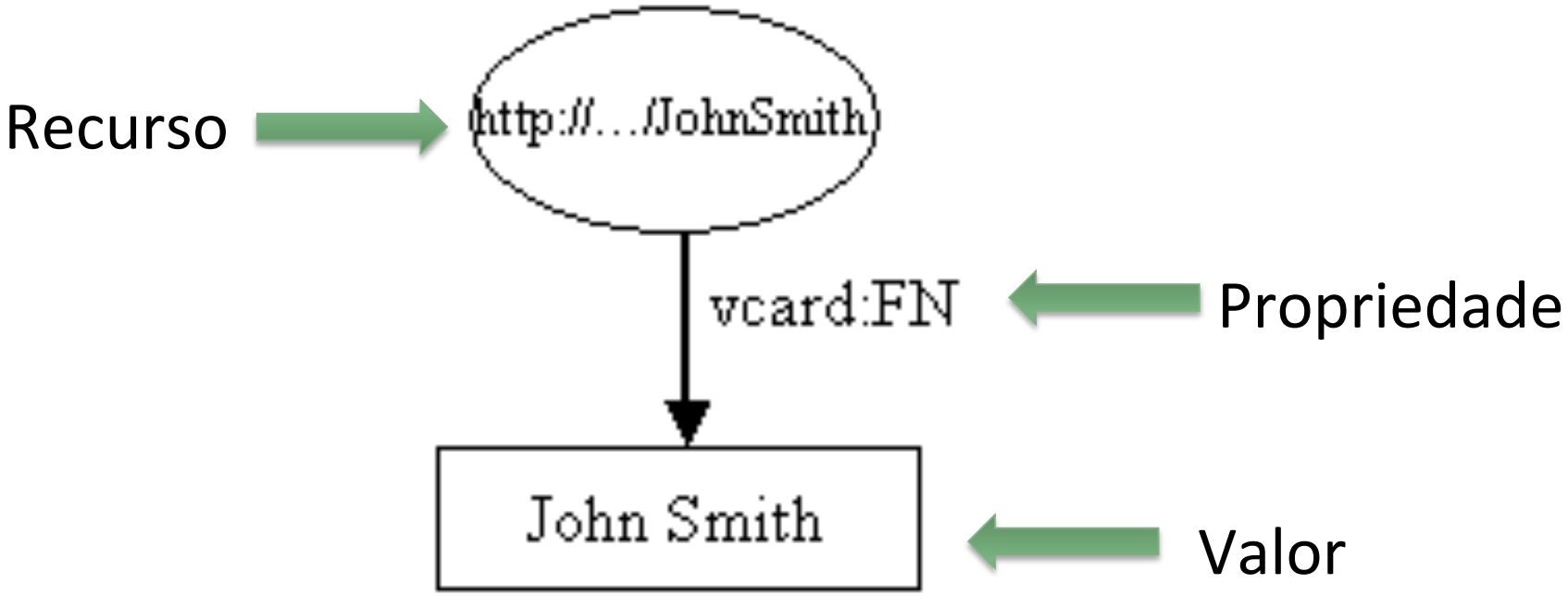
INSTITUTO FEDERAL
ESPIRITO SANTO

Jena e RDF



INSTITUTO FEDERAL
ESPIRITO SANTO

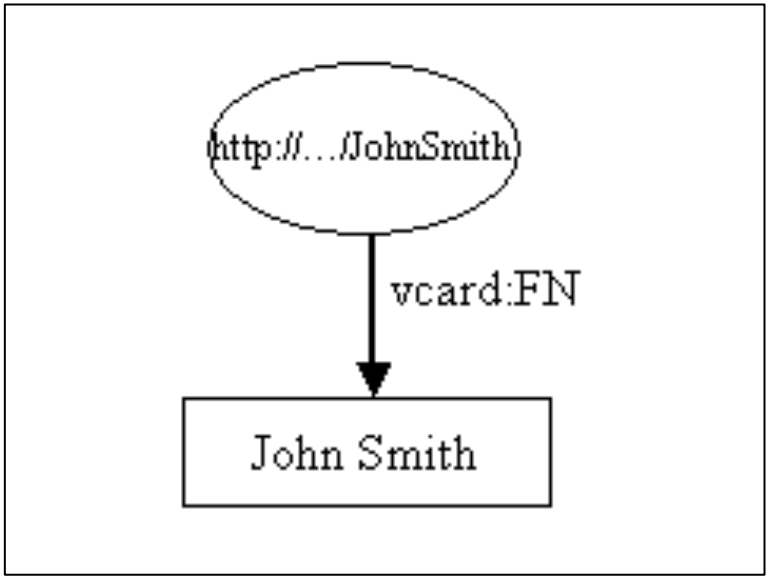
Jena e RDF



INSTITUTO FEDERAL
ESPIRITO SANTO

Jena e RDF

Como criar este modelo no Jena?



INSTITUTO FEDERAL
ESPIRITO SANTO

Jena e RDF

- **Como criar este modelo no Jena?**

```
static String personURI    = "http://somewhere/JohnSmith";
static String fullName     = "John Smith";

public static void main(String[] args)
{
    // create an empty Model
    Model model = ModelFactory.createDefaultModel();
    // create the resource
    Resource johnSmith = model.createResource(personURI);
    // add the property
    johnSmith.addProperty(VCARD.FN, fullName);
    model.write(System.out, "Turtle");
}
```

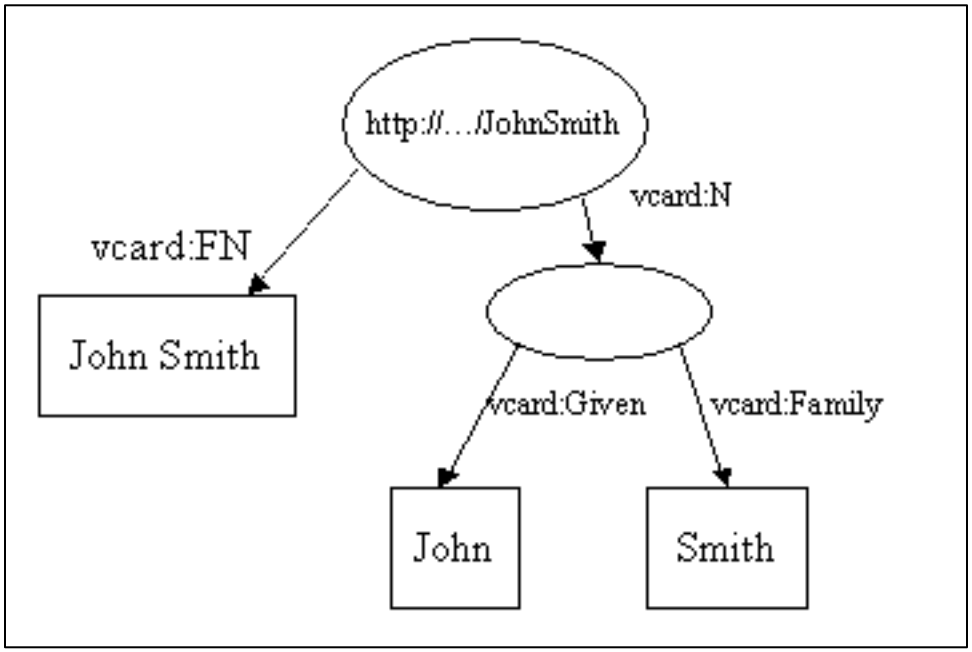

```
static String personURI    = "http://somewhere/JohnSmith";
static String fullName     = "John Smith";

public static void main(String[] args)
{
        // create an empty Model
        Model model = ModelFactory.createDefaultModel();
        // create the resource
        Resource johnSmith = model.createResource(personURI);
        // add the property
        johnSmith.addProperty(VCARD.FN, fullName);
        model.write(System.out, "Turtle");
}
```

```
<http://somewhere/JohnSmith>
        <http://www.w3.org/2001/vcard-rdf/3.0#FN>
                "John Smith" .
```

INSTITUTO FEDERAL
ESPIRITO SANTO

Jena e RDF

Como faremos este modelo?



INSTITUTO FEDERAL
ESPIRITO SANTO

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

public static void main(String[] args) {
        // create an empty Model
        Model model = ModelFactory.createDefaultModel();
        // create the resource
        Resource johnSmith = model.createResource(personURI);
        // add the property
        johnSmith.addProperty(VCARD.FN, fullName);

        Resource n = model.createResource();

        n.addProperty(VCARD.Given, givenName);

        n.addProperty(VCARD.Family,familyName);

        johnSmith.addProperty(VCARD.N, n);

        model.write(System.out,"RDF/XML-ABBREV");
}
```


```
static String personURI    = "http://somewhere/JohnSmith";
...
public static void main(String[] args) {
    // create an empty Model
    ...
    model.write(System.out,"RDF/XML-ABBREV");
}
```

```
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:vcard="http://www.w3.org/2001/vcard-rdf/3.0#">
  <rdf:Description rdf:about="http://somewhere/JohnSmith">
    <vcard:N rdf:parseType="Resource">
      <vcard:Family>Smith</vcard:Family>
      <vcard:Given>John</vcard:Given>
    </vcard:N>
    <vcard:FN>John Smith</vcard:FN>
  </rdf:Description>
</rdf:RDF>
```

Jena e RDF

- A “combinação” de um **recurso**, **propriedade** e **valor** é chamado de ***statement***;

- No ***statement***
  - o **recurso** é chamado de **sujeito**;
  - a **propriedade** é chamada de **predicado**;
  - o **valor** é chamado de **objeto;**

Jena: RDF

```
// list the statements in the Model
StmtIterator iter = model.listStatements();

// print out the predicate, subject and object of each statement
while (iter.hasNext()) {
    Statement stmt      = iter.nextStatement();  // get next statement
    Resource  subject   = stmt.getSubject();     // get the subject
    Property  predicate = stmt.getPredicate();   // get the predicate
    RDFNode   object    = stmt.getObject();      // get the object

    System.out.print(subject.toString());
    System.out.print(" " + predicate.toString() + " ");
    if (object instanceof Resource) {
       System.out.print(object.toString());
    } else {
        // object is a literal
        System.out.print(" \"" + object.toString() + "\"");
    }

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

Jena e RDF

- Lendo modelos RDF em disco:

```
Model m2 = ModelFactory.createDefaultModel();
m2.read( "vc-db-1.rdf" );
```

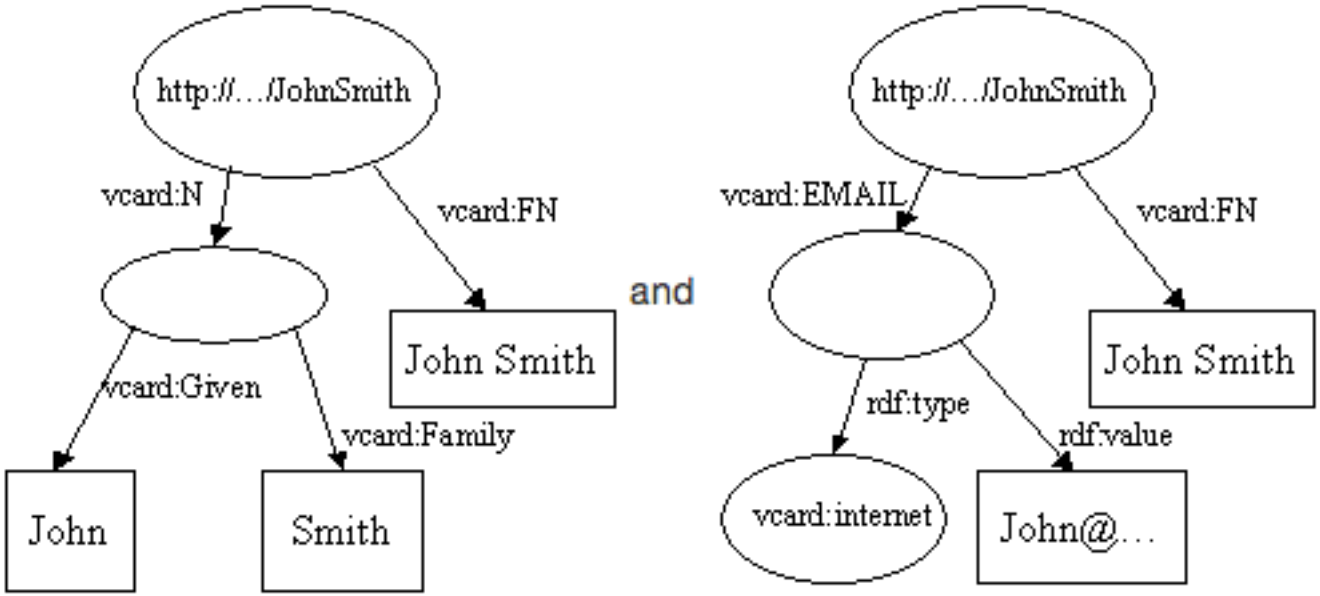
OU

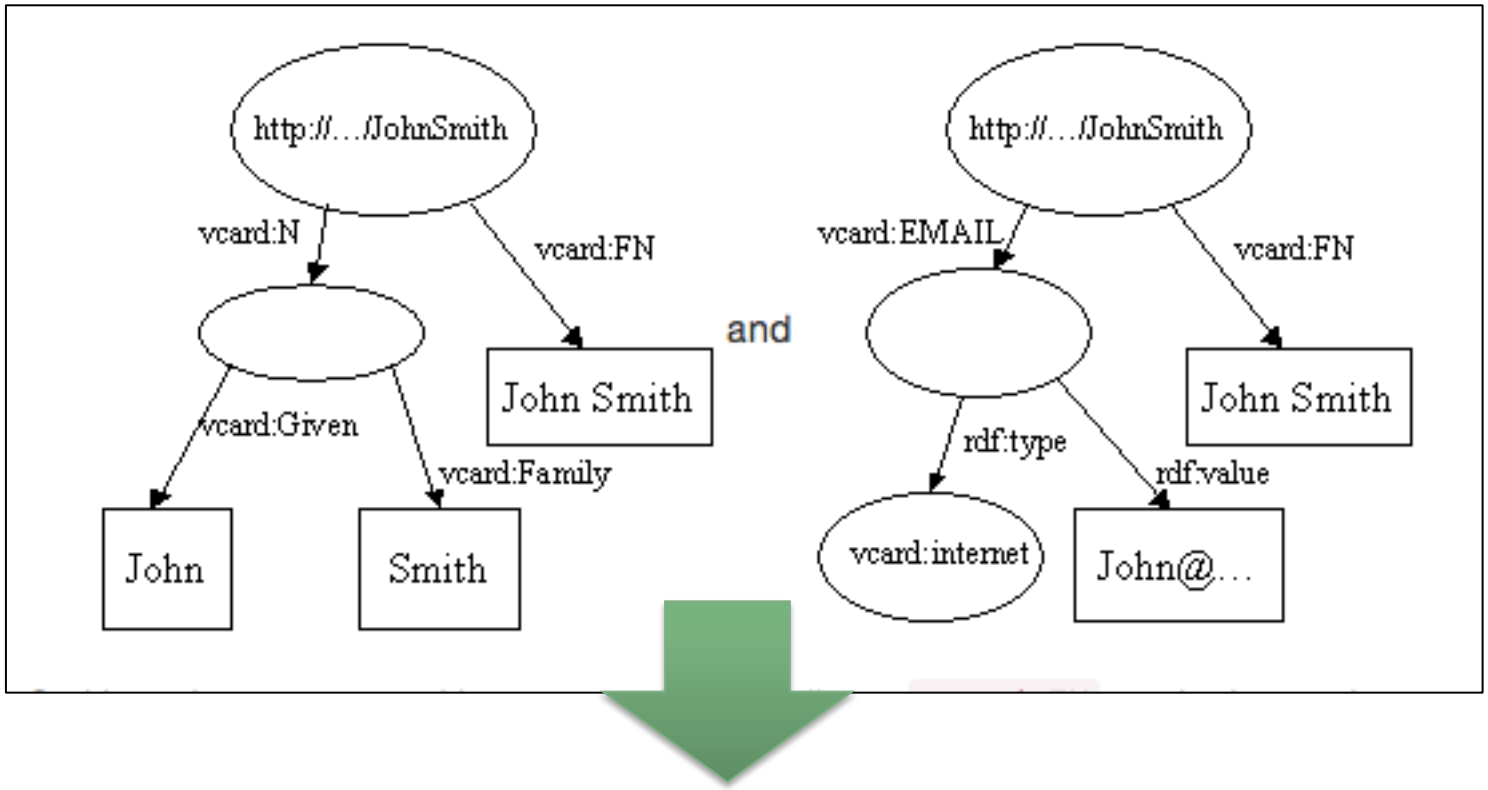
```
Model model1 = ModelFactory.createDefaultModel();
InputStream in1 = FileManager.get().open(inputFileName1);
if (in1 == null) {
        throw new IllegalArgumentException( "File: " + inputFileName1 + " not found");
                }
model1.read( in1, "" );
```

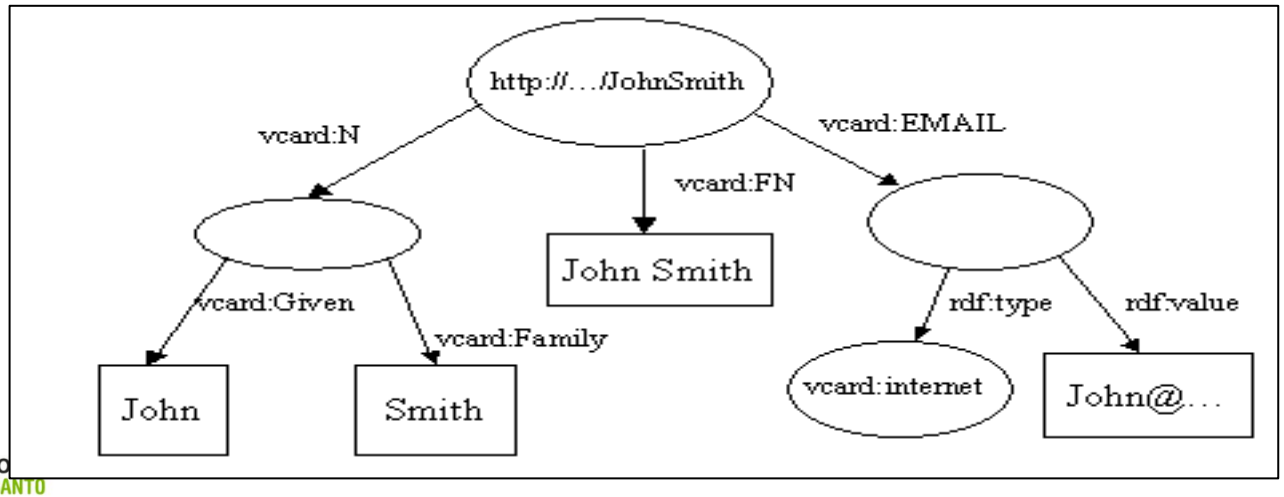
INSTITUTO FEDERAL
ESPIRITO SANTO

Jena e RDF

- Operações com Modelos
  - É possível realizar operações de união, interseção e diferença entre modelos?







INSTITUTO ESPIRITO SANTO

Jena e RDF

- Operações com Modelos

```
Model model1 = ModelFactory.createDefaultModel()
Model model2 = ModelFactory.createDefaultModel();
model1.read( “vc-db-3.rdf”);
model2.read( “vc-db-4.rdf”);

//União de modelos
Model model = model1.union(model2);
model.write(System.out, "RDF/XML-ABBREV");

//Interseção de modelos
model = model1.intersection(model2);
model.write(System.out, "RDF/XML-ABBREV");

// Diferença entre modelos
model = model1. difference(model2);
model.write(System.out, "RDF/XML-ABBREV");
```


Jena e RDF

- **Consultas simples nos modelos:**
  - Jena possui um conjunto de classes que permitem fazer consultas simples no modelo.
  - Para realizar consultas mais complexas é necessário utilizar RQDL (RDF Query Language) ou SPARQL;

Jena e RDF

- **Consultas simples nos modelos:**
  - **Model.listSubjectWithProperty (Property p, RDFNode o)**: permite encontrar um sujeito através de uma propriedade e/ou objeto;

```
// list vcards
ResIterator iter = model.listSubjectsWithProperty(VCARD.FN);
while (iter.hasNext()) {
    Resource r = iter.nextResource();
    ...
}
```

INSTITUTO FEDERAL
ESPIRITO SANTO

Jena e RDF

- **Consultas simples nos modelos:**
  - Interface **Selector:** permite ao programador definir o sujeito, predicado ou objeto que deseja buscar;

```
Selector selector = new SimpleSelector(null, VCARD.FN, null);
```

JENA e SPARQL

Jena e SPARQL

- **SPARQL:**
  - É um linguagem utilizada para realizar consultas em documentos RDF;
  - É padronizada pela W3C e possui um grande uso na comunidade;
  - SPARQL é tanto uma linguagem de consulta quanto uma protocolo de comunicação;
  - O protocolo de comunicação é utilizado para comunicar entre o cliente SPARQL e endpoint/processor (e.g., http://dbpedia.org/sparql)

Jena e SPARQL

- **Hello World:**

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

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

INSTITUTO FEDERAL
ESPIRITO SANTO

Jena e SPARQL

- **Explicando o Hello World:**

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

  - O SPARQL tenta casar o padrão da tripla presente na clausula **WHERE** com as triplas presentes no grafo RDF;
  - O padrão é sujeito, predicado e objeto
  - No nosso exemplo, queremos saber todo os **sujeitos** que possuem o **predicado** FN (FullName) igual ao **objeto "John Smith"**

Jena e SPARQL

- **Executando o Hello World no Jena:**

  1) Carregue o modelo RDF na memória;

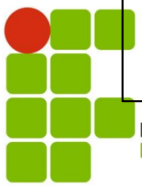
```
Model model = ModelFactory.createDefaultModel();

InputStream in = FileManager.get().open(inputFileName);

        if (in == null) {
            throw new IllegalArgumentException("File: " + inputFileName
                    + " not found");
        }

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


Jena e SPARQL

- **Executando o Hello World no Jena:**

  2) Execute a query ;

```
String queryString =  "SELECT ?x WHERE { ?x  <http://www.w3.org/2001/vcard-rdf/3.0#FN> \"John Smith\" }"

Query query = QueryFactory.create(queryString);

QueryExecution queryExecution = QueryExecutionFactory.create(query,model);

ResultSet results = queryExecution.execSelect();

ResultSetFormatter.out(System.out, results, query) ;

queryExecution.close();
```



Jena e SPARQL

- **Nova Consulta:**

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

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

INSTITUTO FEDERAL
ESPIRITO SANTO

Jena e SPARQL

- **Padrões Básicos:**
  - Um padrão básico é um conjunto de padrões de triplas.

```
SELECT ?givenName
WHERE
  { ?y  <http://www.w3.org/2001/vcard-rdf/3.0#Family>  "Smith" .
    ?y  <http://www.w3.org/2001/vcard-rdf/3.0#Given>  ?givenName .
  }
```

  - O "." é utilizado para indicar que o sujeito será reutilizado por outra tripla;
  - O ";" é utilizado para indicar que o sujeito será reutilizado por duas ou mais triplas;
  - A "," é utilizado para indicar que o sujeito e predicado serão reutilizados por duas ou mais triplas;

INSTITUTO FEDERAL
ESPIRITO SANTO

Jena e SPARQL

- **Padrões Básicos:**
  - O ";" é utilizado para indicar que o sujeito será reutilizado por duas ou mais triplas;

```
SELECT ?job ?birthLoc ?picture
WHERE {
    ?person rdfs:label "George Washington"@en;
            dbprop:occupation ?job;
            dbprop:birthPlace ?birthLoc;
            foaf:img ?picture
}
```

INSTITUTO FEDERAL
ESPIRITO SANTO

Jena e SPARQL

- **QName:**
  - É um mecanismo de prefixagem que permite simplificar as consultas;

```
SELECT ?givenName
WHERE
  { ?y  <http://www.w3.org/2001/vcard-rdf/3.0#Family>  "Smith" .
    ?y  <http://www.w3.org/2001/vcard-rdf/3.0#Given>  ?givenName .
  }
```

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

SELECT ?givenName
WHERE
 { ?y vcard:Family "Smith" .
   ?y vcard:Given  ?givenName .
 }
```


Jena e SPARQL

- **Filtros:**
  - Permite que o usuário faça uma seleção tendo como base um certo padrão;
  - **Busca por Strings**:

```
FILTER regex(?x, "pattern" [, "flags"])
```

    - **?x**: variável que deseja procurar;
    - **“pattern”**: padrão que deseja procurar;
    - **flags**: determinar alguma configuração de busca;

Jena e SPARQL

- **Filtros:**
  - Permite que o usuário faça uma seleção tendo como base um certo padrão;
  - **Busca por Strings**:

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

SELECT ?g
WHERE
{ ?y vcard:Given ?g .
  FILTER regex(?g, "r", "i") }
```

    - **i**: determina que a busca é case-insensitivo;

Jena e SPARQL

- **Filtros:**
  - Permite que o usuário faça uma seleção tendo como base um certo padrão;
  - **Busca por Valores**:

```
PREFIX info: <http://somewhere/peopleInfo#>

SELECT ?resource
WHERE
  {
    ?resource info:age ?age .
    FILTER (?age >= 24)
  }
```


Jena e SPARQL

- **Filtros:**
  - Permite que o usuário faça uma seleção tendo como base um certo padrão;
  - **Operador IN**: verificar se o valor está entre um conjunto de valores pré-definidos;

```
PREFIX dm:  <http://learningsparql.com/ns/demo#>
PREFIX db: <http://dbpedia.org/resource/>

SELECT ?s ?cost ?location
WHERE
{
  ?s dm:location ?location ;
     dm:cost ?cost .
  FILTER (?location IN (db:Montreal, db:Lisbon)) .
}
```

INSTITUTO FEDERAL
ESPIRITO SANTO

Jena e SPARQL

- **Filtros:**
  - Permite que o usuário faça uma seleção tendo como base um certo padrão;
  - **Operador IN**: verificar se o valor está entre um conjunto de valores pré-definidos;

```
PREFIX dm:  <http://learningsparql.com/ns/demo#>
PREFIX db: <http://dbpedia.org/resource/>

SELECT ?s ?cost ?location
WHERE
{
  ?s dm:location ?location ;
     dm:cost ?cost .
  FILTER (?location NOT IN (db:Montreal, db:Lisbon)) .
}
```


Jena e SPARQL

- **Limitadores:**
  - O comando **LIMIT** determina o número máximo de resultados que serão retornados;
  - O comando **OFFSET** determina o número de resultados que serão "pulados" do resultado da consulta

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

SELECT ?label
WHERE
{ ?s rdfs:label ?label . }
OFFSET 3
LIMIT 1
```


Jena e SPARQL

- **Limitadores:**
  - O comando **LIMIT** determina o número máximo de resultados que serão retornados;
  - O comando **OFFSET** determina o número de resultados que serão "pulados" do resultado da consulta

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

SELECT ?label
WHERE
{ ?s rdfs:label ?label . }
OFFSET 3
LIMIT 1
```


Jena e SPARQL

- **Limitadores:**
  - O comando **LIMIT** determina o número máximo de resultados que serão retornados;
  - O comando **OFFSET** determina o número de resultados que serão “pulados” do resultado da consulta

Jena e SPARQL

- **Consultas“dinâmicas”:**
  - Em muitos casos é necessário passar parâmetros para as consultas;
  - JENA permita fazer o mapeamento entre uma variável da consulta com um valor desejado;
  - O mapeamento é feito utilizando as seguintes classes:
    - **ParameterizedSparqlString;**
    - **QuerySolutionMap;**

Jena e SPARQL

- **Consultas“dinâmicas”:**
  - **ParameterizedSparqlString:**

```
ParameterizedSparqlString queryStr = new ParameterizedSparqlString(queryString);

queryStr.setLiteral(parameter, value);

Query query = QueryFactory.create(queryStr.toString());

QueryExecution queryExecution = QueryExecutionFactory.create(query,model);
```

INSTITUTO FEDERAL
ESPIRITO SANTO

Jena e SPARQL

- **Consultas“dinâmicas”:**
  - **QuerySolutionMap:**

```
RDFNode node = model.createResource(value);

QuerySolutionMap map = new QuerySolutionMap();

map.add(parameter, node);

Query query = QueryFactory.create(queryString);

QueryExecution queryExecution = QueryExecutionFactory.create(query,model, map);
```

INSTITUTO FEDERAL
ESPIRITO SANTO

Jena e SPARQL

- **Conectando a banco de dados externos:**
  - O JENA permite que o desenvolvedor acesse outros bancos de dados;
  - Para realizar o acesso é necessário utilizar o seguinte comando:

"QueryExecutionFactory.*sparqlService("<Server sparql>", query)"*

Jena e SPARQL

- **Conectando a banco de dados externos:**
  - Conectando no **DBPedia**:

```
QueryExecution queryExecution = QueryExecutionFactory.sparqlService("http://
dbpedia.org/sparql", query);

ResultSet results = queryExecution.execSelect();
```

INSTITUTO FEDERAL
ESPIRITO SANTO

```
String query1 ="      PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>"+
                                                "PREFIX rdf: <http://www.w3.org/
1999/02/22-rdf-syntax-ns#> "+
                                                " PREFIX : <http://dbpedia.org/resource/>"+
                                                " PREFIX d: <http://dbpedia.org/ontology/> "+
                                                " SELECT distinct ?albumName ?artistName "+
                                                " WHERE "+
                                                " { "+
                                                " ?album d:producer :Timbaland . "+
                                                " ?album d:musicalArtist ?artist ."+
                                                " ?album rdfs:label ?albumName . "+
                                                " ?artist rdfs:label ?artistName ." +
                                                " FILTER ( lang(?artistName) = \"en\")" +
                                                " FILTER ( lang(?albumName) = \"en\" )" +
                                                " }";

ResultSet results = SPARQLUtil.INSTANCE.dbpediaQuery(query1)
ResultSetFormatter.out(System.out, results) ;
```

INSTITUTO FEDERAL
ESPIRITO SANTO

Jena e SPARQL

- **Optional:**
  - Apresenta o dado caso ele exista

```
SELECT ?first ?last ?workTel
WHERE
{
  ?s ab:firstName ?first ;
     ab:lastName ?last .
  OPTIONAL
  { ?s ab:workTel ?workTel . }
}
```

```
-----------------------------------------
| first     | last       | workTel      |
=========================================
| "Craig"   | "Ellis"    | "(245) 315-5486" |
| "Cindy"   | "Marshall" |                  |
| "Richard" | "Mutt"     |                  |
-----------------------------------------
```

INSTITUTO FEDERAL
ESPIRITO SANTO

Jena e SPARQL

- **Optional:**
  - Apresenta o dado caso ele exista

```
SELECT ?first ?last ?workTel ?nick
WHERE
{
  ?s ab:firstName ?first ;
     ab:lastName ?last .
  OPTIONAL { ?s ab:workTel ?workTel . }
  OPTIONAL { ?s ab:nick ?nick .  }

}
```

first	last	workTel	nick
"Craig"	"Ellis"	"(245) 315-5486"	
"Cindy"	"Marshall"		
"Richard"	"Mutt"		"Dick"

INSTITUTO FEDERAL
ESPIRITO SANTO

Jena e SPARQL

- **Optional com Filtro:**

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

SELECT ?name ?age
WHERE
{
    ?person vcard:FN  ?name .
    OPTIONAL { ?person info:age ?age . FILTER ( ?age > 24 ) }
}
```

Jena e SPARQL

- **UNION:**

```
PREFIX ab: <http://learningsparql.com/ns/addressbook#>
PREFIX d:  <http://learningsparql.com/ns/data#>

SELECT *
WHERE
{
   { ?person ab:firstName ?first ; ab:lastName ?last . }

   UNION

   { ?course ab:courseTitle ?courseName . }

}
```

INSTITUTO FEDERAL
ESPIRITO SANTO

```
PREFIX ab: <http://learningsparql.com/ns/addressbook#>
PREFIX d:  <http://learningsparql.com/ns/data#>

SELECT *
WHERE
{
   { ?person ab:firstName ?first ; ab:lastName ?last . }

   UNION

   { ?course ab:courseTitle ?courseName . }

}
```

person	first	last	course	courseName
d:i8301	"Craig"	"Ellis"		
d:i9771	"Cindy"	"Marshall"		
d:i0432	"Richard"	"Mutt"		
			d:course85	"Updating Data with SPARQL"
			d:course59	"Using SPARQL with non-RDF Data"
			d:course71	"Enhancing Websites with RDFa"
			d:course34	"Modeling Data with OWL"

Jena e SPARQL

- **UNION:**

```
SELECT ?first ?last ?instrument
WHERE
{
  { ?person ab:firstName ?first ;
            ab:lastName ?last ;
            ab:instrument "trumpet" ;
            ab:instrument ?instrument .
  }

  UNION

  { ?person ab:firstName ?first ;
            ab:lastName ?last ;
            ab:instrument "sax" ;
            ab:instrument ?instrument .
  }

}
```

```
-----------------------------------
| first     | last    | instrument |
===================================
| "Craig"   | "Ellis" | "trumpet"  |
| "Richard" | "Mutt"  | "clarinet" |
| "Richard" | "Mutt"  | "sax"      |
-----------------------------------
```

INSTITUTO FEDERAL
ESPIRITO SANTO

Jena e SPARQL

- O SPARQL possui 4 “tipos” de consultas:
  - **Select**: é um projeção identifica quais variáveis nomeadas estão no conjunto resultado;
  - **Construct**: monta um RDF baseado num grafo template. O grafo template pode ter variáveis que são definidas na clausula WHERE;
  - **Describe**: monta um RDF baseado em um grafo que é montado pelo processador da query;
  - **Ask**: retorna um booleano, true se o padrão for casado, ou false caso contrário;

Jena e SPARQL

- Construct:

```
PREFIX ab: <http://learningsparql.com/ns/addressbook#>
PREFIX d:  <http://learningsparql.com/ns/data#>

SELECT ?person ?p ?o
WHERE
{
  ?person ab:firstName "Craig" ;
          ab:lastName  "Ellis" ;
          ?p ?o .
}
```

```
PREFIX ab: <http://learningsparql.com/ns/addressbook#>
PREFIX d:  <http://learningsparql.com/ns/data#>

CONSTRUCT
{ ?person ?p ?o . }

WHERE
{
  ?person ab:firstName "Craig" ;
          ab:lastName  "Ellis" ;
          ?p ?o .
}
```

INSTITUTO FEDERAL
ESPIRITO SANTO

Jena e SPARQL

- Construct:

```
PREFIX ab: <http://learningsparql.com/ns/addressbook#>
PREFIX d:  <http://learningsparql.com/ns/data#>

CONSTRUCT
{ ?p ab:hasGrandfather ?g . }
WHERE
{
  ?p ab:hasParent ?parent .
  ?parent ab:hasParent ?g .
  ?g ab:gender d:male .
}
```

INSTITUTO FEDERAL
ESPIRITO SANTO

Jena e SPARQL

- ASK:

```
PREFIX dm: <http://learningsparql.com/ns/demo#>

ASK WHERE
{
  ?s dm:location ?city .
  FILTER (!(isURI(?city)))
}
```

INSTITUTO FEDERAL
ESPIRITO SANTO

Jena e SPARQL

- ASK:

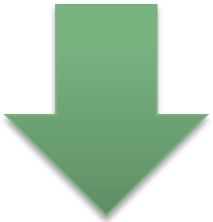
```
PREFIX dm:  <http://learningsparql.com/ns/demo#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

ASK WHERE
{
  ?item dm:amount ?amount .
  FILTER ((datatype(?amount)) != xsd:integer)
}
```

INSTITUTO FEDERAL
ESPIRITO SANTO

Jena e SPARQL

- **Describe**:
  - O processador da query que decide qual informação será retornada

```
DESCRIBE  <http://learningsparql.com/ns/data#course59>
```



```
@prefix d:       <http://learningsparql.com/ns/data#> .
@prefix ab:      <http://learningsparql.com/ns/addressbook#> .

d:course59
      ab:courseTitle  "Using SPARQL with non-RDF Data" .
```

INSTITUTO FEDERAL
ESPIRITO SANTO

JENA e Ontologias

Jena e Ontologias

- Jena permite trabalhar com:

1. RDFS (*Resource Description Framework Schema*);

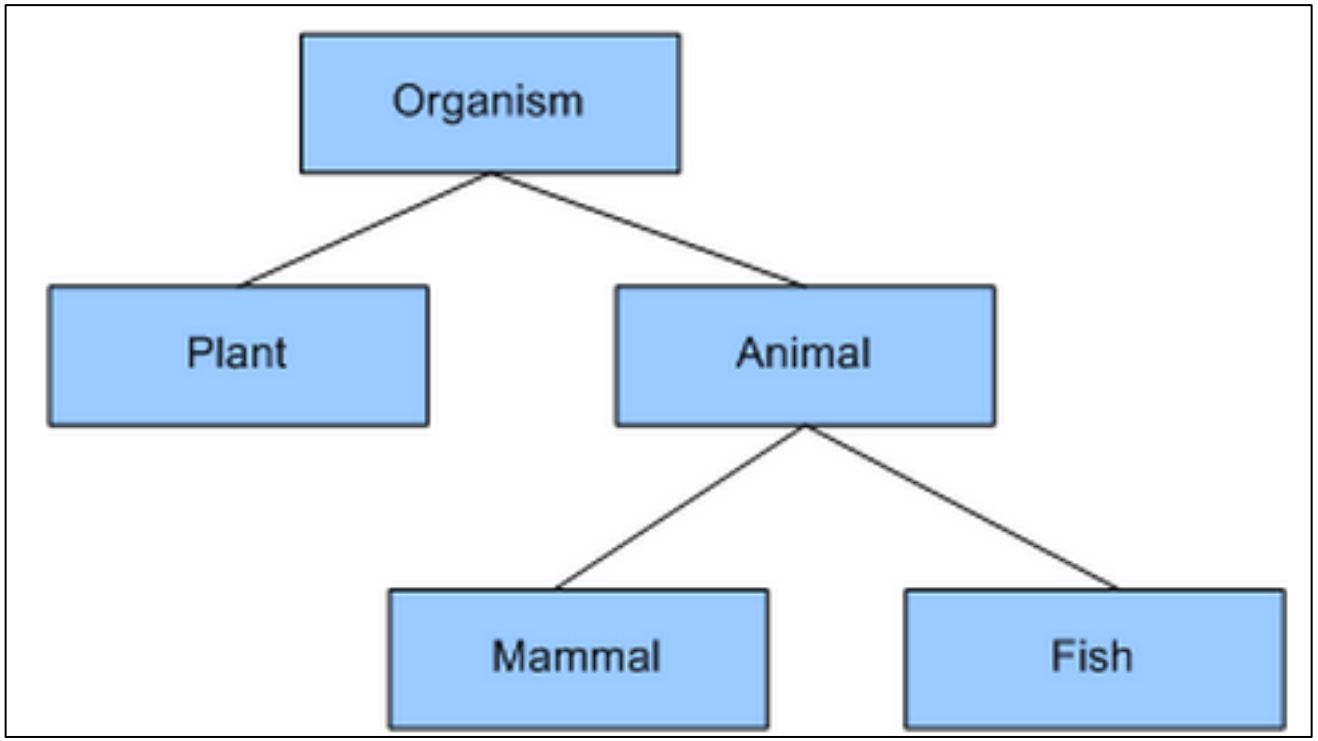
2. OWL (*Web Ontology Language*) e suas variações;

Jena e Ontologias

- RDFS:
  - É uma linguagem de construção ontologia “fracas”.
  - É dita uma linguagem “fraca” pois permite construir apenas simples hierarquias de conceitos e de relacionamentos;

Jena e Ontologias

- RDFS:



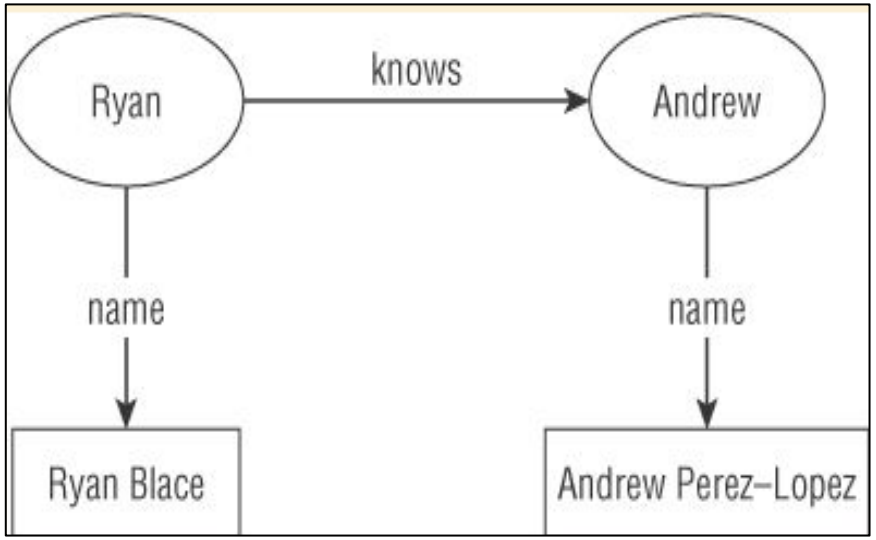
INSTITUTO FEDERAL
ESPIRITO SANTO

Jena e Ontologias

- **OWL:**
  - Primeira versão criada em 2004, pelo W3C Web Ontology (WebOnt);
  - Segunda versão foi lançada em 2009, chamada de OWL 2;
  - A OWL é uma linguagem que foi derivada das linguagens DAML e OIL;
  - A OWL também estende o vocabulário do *Resource Description Framework Schema* (RDFS);
  - Principais perfís: (i) OWL Full , (ii) OWL DL e (iii) OWL Lite;
  - Outros perfís: (i) OWL EL, (ii) OWL QL e (iii) OWL RL

Jena e Ontologias

- **OWL:**
  - Composta de três blocos semânticos:
    - **Classes**;
    - **indivíduos**;
    - **Propriedades** (Object Properties e datatype properties);



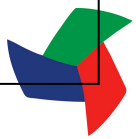
INSTITUTO FEDERAL
ESPIRITO SANTO

Jena e Ontologias

- **OWL: Classes:**

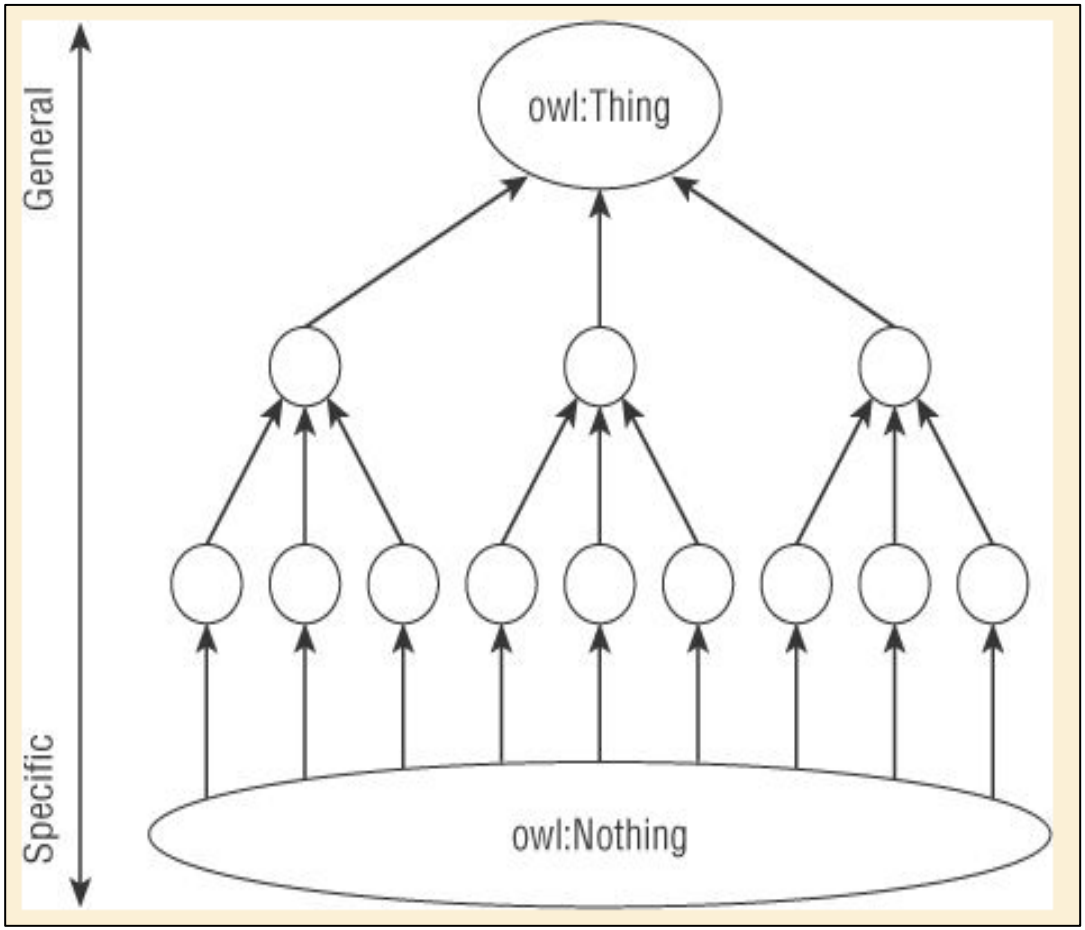
```
#Canine

ex: Canine owl: Class.

ex: Toto ex: Canine
```

```
#Canine

ex: Mammal owl: Class

ex: Canine owl: class; rdfs: subClassOf ex: Mammal

ex: Toto ex: Canine
```

Jena e Ontologias

- **OWL: Classes:**



INSTITUTO FEDERAL
ESPIRITO SANTO

Jena e Ontologias

- **OWL: Propriedade:**

As propriedades fundamentais em OWL são: OWL: ObjectProperty e owl:DataTypeProperty.

```
#Canine

ex: registeredName owl: DataTypeProperty;

rdfs:subPropertyof ex:name;
```


Jena e Ontologias

- **OWL:**

Constructs	Supported by	Notes
rdfs:subClassOf, rdfs:subPropertyOf, rdf:type	all	Normal RDFS semantics supported including meta use (e.g. taking the subPropertyOf subClassOf).
rdfs:domain, rdfs:range	all	Stronger if-and-only-if semantics supported
owl:intersectionOf	all	
owl:unionOf	all	Partial support. If C=unionOf(A,B) then will infer that A,B are subclasses of C, and thus that instances of A or B are instances of C. Does not handle the reverse (that an instance of C must be either an instance of A or an instance of B).
owl:equivalentClass	all	
owl:disjointWith	full, mini	
owl:sameAs, owl:differentFrom, owl:distinctMembers	full, mini	owl:distinctMembers is currently translated into a quadratic set of owl:differentFrom assertions.
Owl:Thing	all	
owl:equivalentProperty, owl:inverseOf	all	
owl:FunctionalProperty, owl:InverseFunctionalProperty	all	
owl:SymmeticProperty, owl:TransitiveProperty	all	

Jena e Ontologias

- **OWL:**

owl:someValuesFrom	full, (mini)	Full supports both directions (existence of a value implies membership of someValuesFrom restriction, membership of someValuesFrom implies the existence of a bNode representing the value). Mini omits the latter "bNode introduction" which avoids some infinite closures.
owl:allValuesFrom	full, mini	Partial support, forward direction only (member of a allValuesFrom(p, C) implies that all p values are of type C). Does handle cases where the reverse direction is trivially true (e.g. by virtue of a global rdfs:range axiom).
owl:minCardinality, owl:maxCardinality, owl:cardinality	full, (mini)	Restricted to cardinalities of 0 or 1, though higher cardinalities are partially supported in validation for the case of literal-valued properties. Mini omits the bNodes introduction in the minCardinality(1) case, see someValuesFrom above.
owl:hasValue	all	

INSTITUTO FEDERAL
ESPIRITO SANTO

Jena e Ontologias

- **OWL:**

Relacionamento	Definição
Owl:symetricproperty	(A p B) implica que (B p A)
Owl: Asymetricproperty	(A p B) implica que não (B p A)
OWL: reflexiveProperty	Implica que (A p A), para todo A.
OWL: irreflexiveProperty	Implica que não há (A p A), para todo A.
OWL: transitiveProperty	Se (A p B) e (B p C) então (A p C).
OWL: functionalProperty	Se (A p x) e (C p y) então x = y.
OWL: ImverseFunctionalProperty	Se (A p B) e (B p C) então A = C

INSTITUTO FEDERAL
ESPIRITO SANTO

Jena e Ontologias

- **Criando uma Ontologia:**

```
OntModel m = ModelFactory.createOntologyModel();
```

  - A configuração padrão é:
    - OWL-FULL;
    - Armazenamento em memória;
    - RDFS inferência;

OntModelSpec	Language profile	Storage model	Reasoner
OWL_MEM	OWL full	in-memory	none
OWL_MEM_TRANS_INF	OWL full	in-memory	transitive class-hierarchy inference
OWL_MEM_RULE_INF	OWL full	in-memory	rule-based reasoner with OWL rules
OWL_MEM_MICRO_RULE_INF	OWL full	in-memory	optimised rule-based reasoner with OWL rules
OWL_MEM_MINI_RULE_INF	OWL full	in-memory	rule-based reasoner with subset of OWL rules
OWL_DL_MEM	OWL DL	in-memory	none
OWL_DL_MEM_RDFS_INF	OWL DL	in-memory	rule reasoner with RDFS-level entailment-rules
OWL_DL_MEM_TRANS_INF	OWL DL	in-memory	transitive class-hierarchy inference
OWL_DL_MEM_RULE_INF	OWL DL	in-memory	rule-based reasoner with OWL rules
OWL_LITE_MEM	OWL Lite	in-memory	none
OWL_LITE_MEM_TRANS_INF	OWL Lite	in-memory	transitive class-hierarchy inference
OWL_LITE_MEM_RDFS_INF	OWL Lite	in-memory	rule reasoner with RDFS-level entailment-rules
OWL_LITE_MEM_RULES_INF	OWL Lite	in-memory	rule-based reasoner with OWL rules
RDFS_MEM	RDFS	in-memory	none
RDFS_MEM_TRANS_INF	RDFS	in-memory	transitive class-hierarchy inference
RDFS_MEM_RDFS_INF	RDFS	in-memory	rule reasoner with RDFS-level entailment-rules

Jena e Ontologias

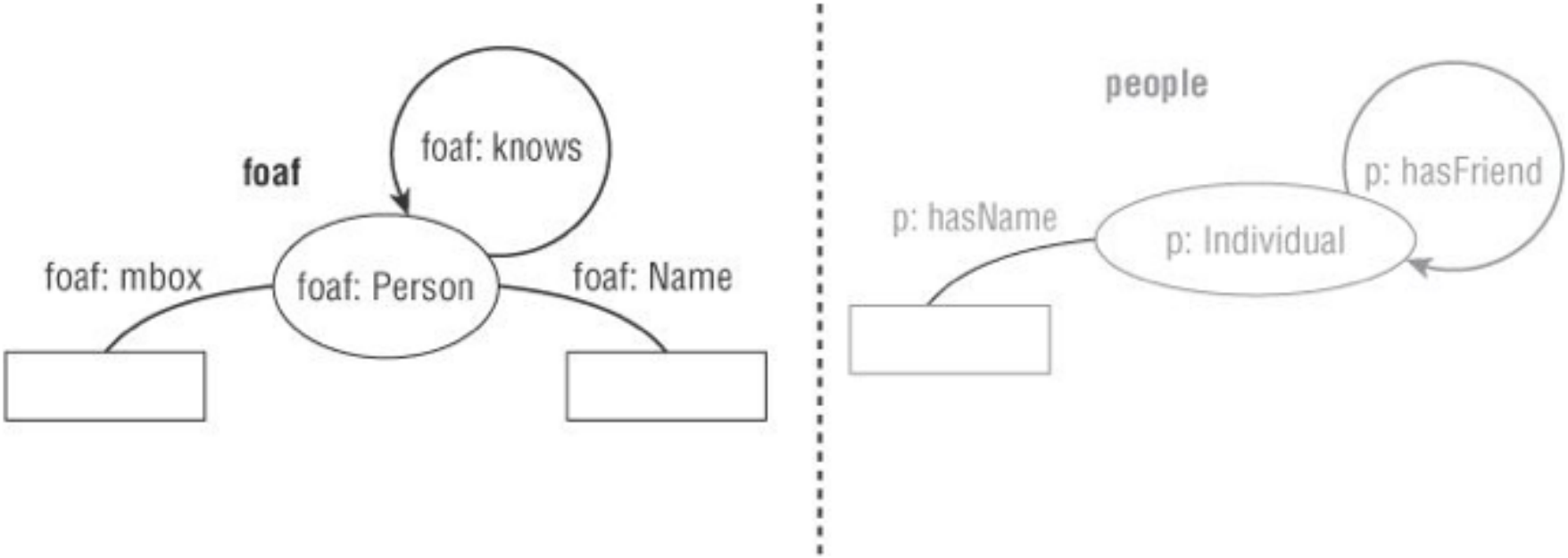
- **Carregando um modelo:**

```
public OntModel _friends = null;
public OntModel schema = null;
public InfModel inferredFriends = null;

public void populateFOAFFriends(){
        _friends = ModelFactory.createOntologyModel();
        InputStream inFoafInstance = FileManager.get().open("Ontologies/FOAFFriends.rdf");
        _friends.read(inFoafInstance,defaultNameSpace);
}
```

INSTITUTO FEDERAL
ESPIRITO SANTO

Jena e Ontologias

Como alinhar duas ontologias distintas?

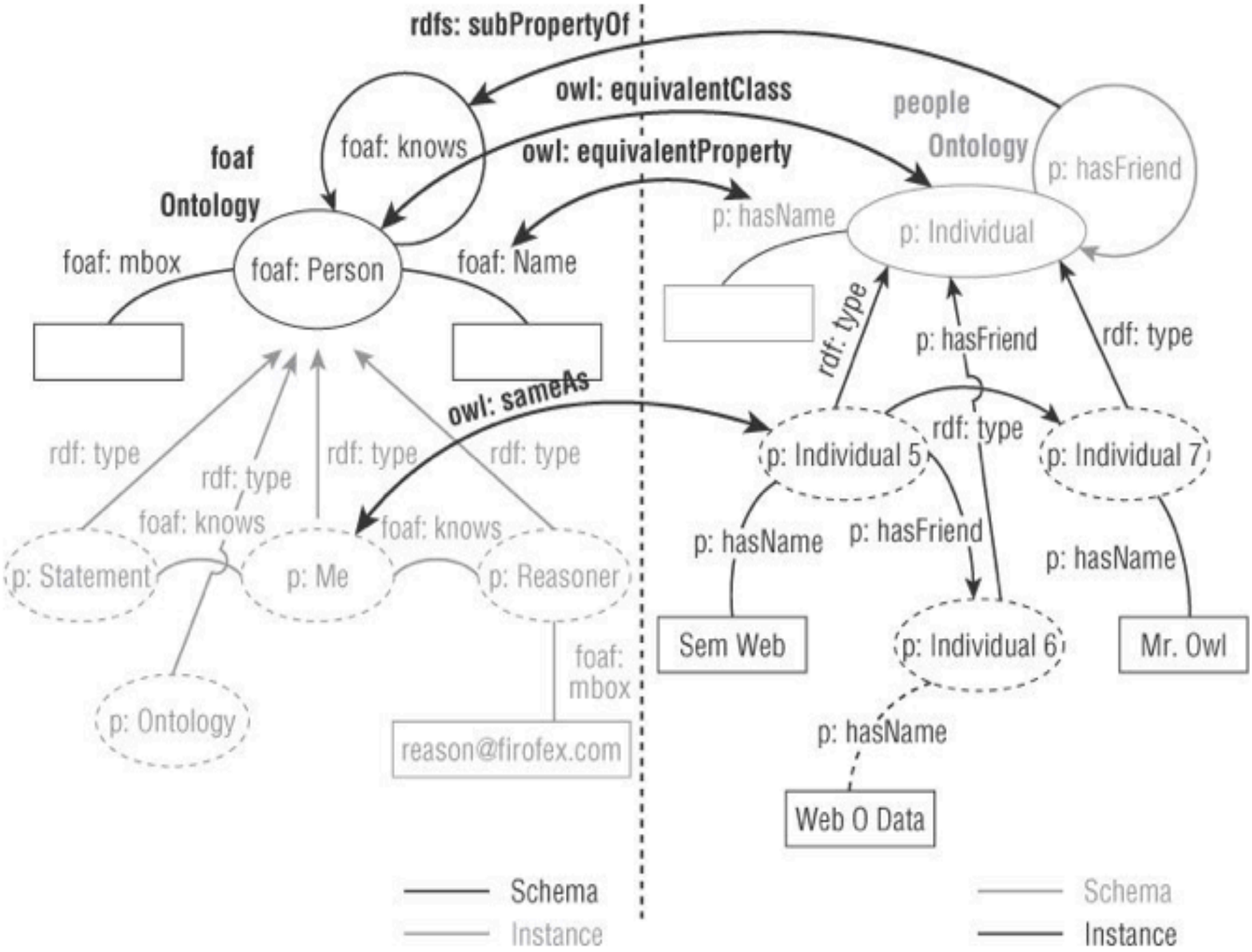


INSTITUTO FEDERAL
ESPIRITO SANTO

Jena e Ontologias

Como alinhar duas ontologias distintas?

- Antes de realizar o alinhamento é necessário decidir o nível do alinhamento:
  1. Nível de Classe;
  2. Nível de propriedade;
  3. Nível de individuo;


rdfs: subPropertyOf
owl: equivalentClass
owl: equivalentProperty
foaf
Ontology
foaf: knows
foaf: mbox
foaf: Person
foaf: Name
people
Ontology
p: hasFriend
p: hasName
p: Individual
rdf: type
p: hasFriend
rdf: type
owl: sameAs
rdf: type
rdf: type
rdf: type
rdf: type
p: Individual 5
rdf: type
p: Individual 7
foaf: knows
foaf: knows
p: hasName
p: hasFriend
p: hasName
p: Statement
p: Me
p: Reasoner
Sem Web
p: Individual 6
Mr. Owl
foaf:
mbox
p: Ontology
reason@firofex.com
p: hasName
Web O Data
Schema
Instance
Schema
Instance

Jena e Ontologias

Como alinhar duas ontologias distintas?

1. Carregue as ontologias com a semântica dos elementos;

2. Realizar o alinhamento entre os elementos das ontologias;

3. Aplique um *reasoner* para realizar o alinhamento;

INSTITUTO FEDERAL
ESPIRITO SANTO

Jena e Ontologias

Como alinhar duas ontologias distintas?

1. Carregue as ontologias com a semântica dos elementos;

```
public void populateFOAFSchema() throws IOException{
        InputStream inFoaf = FileManager.get().open("Ontologies/foaf.rdf");
        schema = ModelFactory.createOntologyModel();
        schema.read(inFoaf, defaultNameSpace);
        inFoaf.close();

}

public void populateNewFriendsSchema() throws IOException {
        InputStream inFoafInstance = FileManager.get().open("Ontologies/additionalFriendsSchema.owl");
        _friends.read(inFoafInstance,defaultNameSpace);
        inFoafInstance.close();
}
```

INSTITUTO FEDERAL
ESPIRITO SANTO

Jena e Ontologias

Como alinhar duas ontologias distintas?

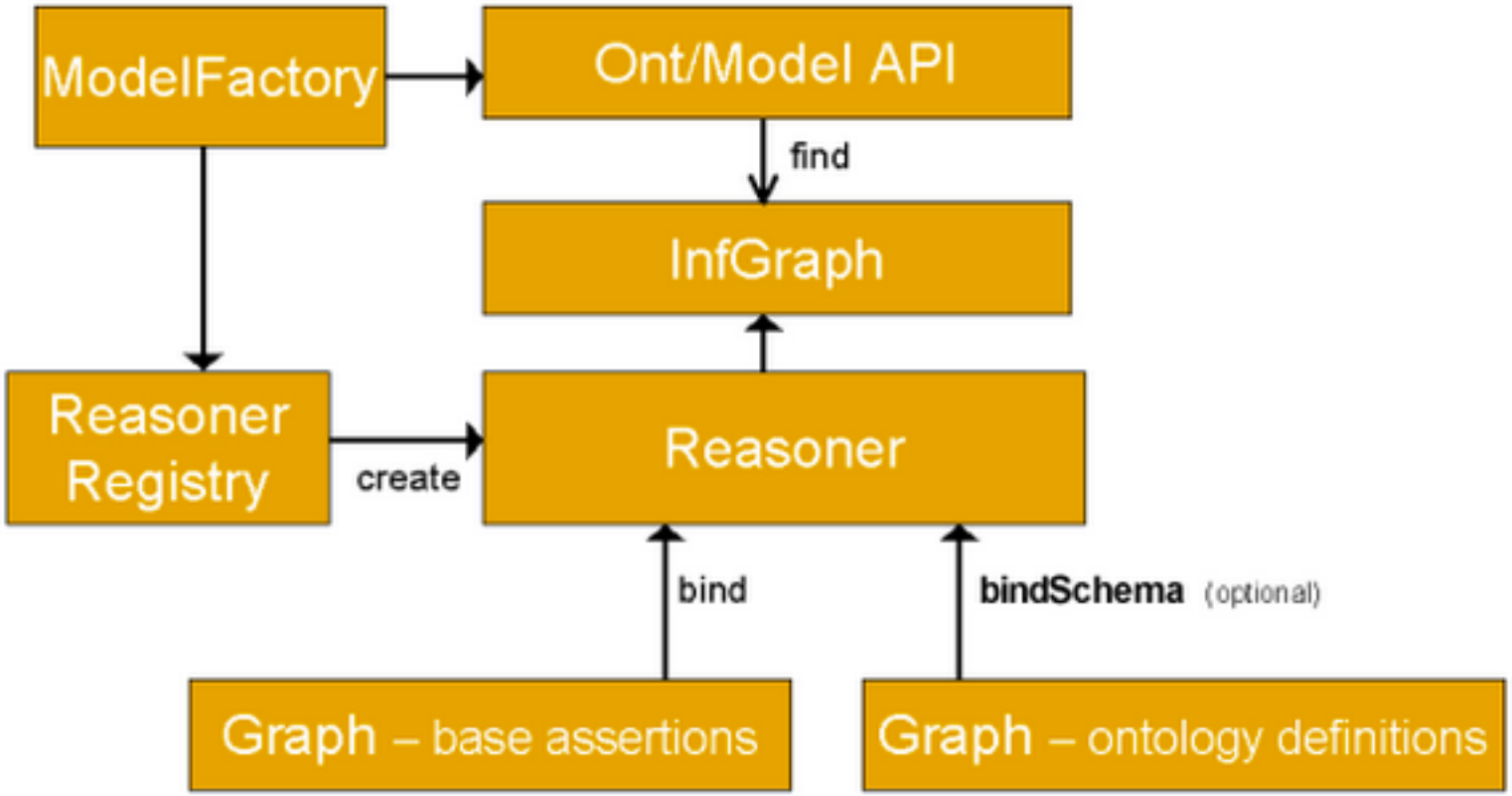
2. Realizar o alinhamento entre os elementos das ontologias;

```
// State that :individual is equivalentClass of foaf:Person
Resource resource = schema.createResource(defaultNameSpace + "Individual");
Property prop = schema.createProperty("http://www.w3.org/2002/07/owl#equivalentClass");
Resource obj = schema.createResource("http://xmlns.com/foaf/0.1/Person");
schema.add(resource,prop,obj);

//State that :hasName is an equivalentProperty of foaf:name
resource = schema.createResource(defaultNameSpace + "hasName");
//prop = schema.createProperty("http://www.w3.org/2000/01/rdf-schema#subPropertyOf");
prop = schema.createProperty("http://www.w3.org/2002/07/owl#equivalentProperty");
obj = schema.createResource("http://xmlns.com/foaf/0.1/name");
schema.add(resource,prop,obj);

//State that :hasFriend is a subproperty of foaf:knows
resource = schema.createResource(defaultNameSpace + "hasFriend");
prop = schema.createProperty("http://www.w3.org/2000/01/rdf-schema#subPropertyOf");
obj = schema.createResource("http://xmlns.com/foaf/0.1/knows");
schema.add(resource,prop,obj);


//State that sem web is the same person as Semantic Web
resource = schema.createResource("http://org.semwebprogramming/chapter2/people#me");
prop = schema.createProperty("http://www.w3.org/2002/07/owl#sameAs");
obj = schema.createResource("http://org.semwebprogramming/chapter2/people#Individual_5");
schema.add(resource,prop,obj);
```

INSTITUTO FEDERAL
ESPIRITO SANTO

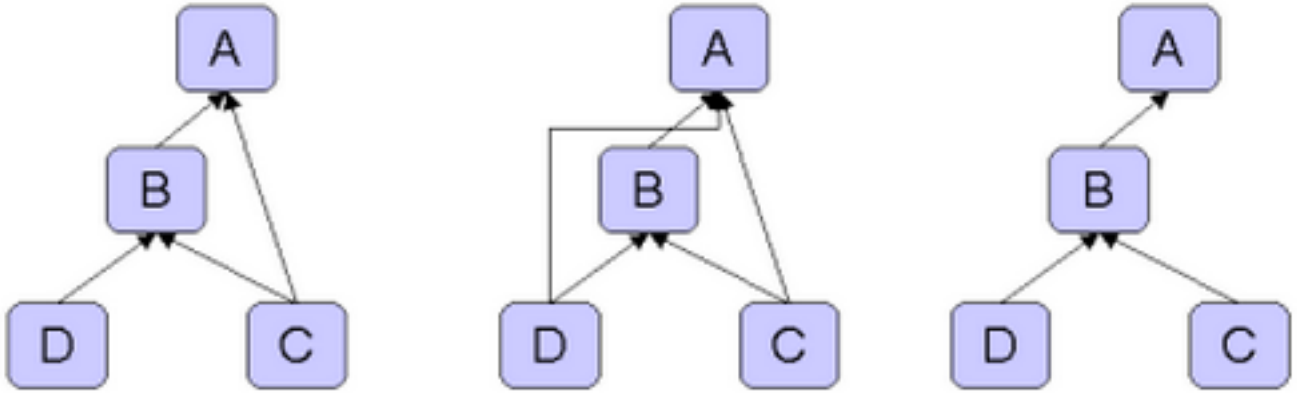
```
// State that :individual is equivalentClass of foaf:Person
Resource resource = schema.createResource(defaultNameSpace + "Individual");
Property prop = schema.createProperty("http://www.w3.org/2002/07/owl#equivalentClass");
Resource obj = schema.createResource("http://xmlns.com/foaf/0.1/Person");
schema.add(resource,prop,obj);

//State that :hasName is an equivalentProperty of foaf:name
resource = schema.createResource(defaultNameSpace + "hasName");
//prop = schema.createProperty("http://www.w3.org/2000/01/rdf-schema#subPropertyOf");
prop = schema.createProperty("http://www.w3.org/2002/07/owl#equivalentProperty");
obj = schema.createResource("http://xmlns.com/foaf/0.1/name");
schema.add(resource,prop,obj);

//State that :hasFriend is a subproperty of foaf:knows
resource = schema.createResource(defaultNameSpace + "hasFriend");
prop = schema.createProperty("http://www.w3.org/2000/01/rdf-schema#subPropertyOf");
obj = schema.createResource("http://xmlns.com/foaf/0.1/knows");
schema.add(resource,prop,obj);


//State that sem web is the same person as Semantic Web
resource = schema.createResource("http://org.semwebprogramming/chapter2/people#me");
prop = schema.createProperty("http://www.w3.org/2002/07/owl#sameAs");
obj = schema.createResource("http://org.semwebprogramming/chapter2/people#Individual_5");
schema.add(resource,prop,obj);
```

Jena e Ontologias



INSTITUTO FEDERAL
ESPIRITO SANTO

Jena e Ontologias



INSTITUTO FEDERAL
ESPIRITO SANTO

Jena e Ontologias

Como alinhar duas ontologias distintas?

3. Aplique um *reasoner* para realizar o alinhamento;

```
public void bindReasoner(){
    Reasoner reasoner = ReasonerRegistry.getOWLReasoner();
    reasoner = reasoner.bindSchema(schema);
    inferredFriends = ModelFactory.createInfModel(reasoner, _friends);

}
```

INSTITUTO FEDERAL
ESPIRITO SANTO

Jena e Ontologias

Regras:

- Suporte a regras;
- padrão If-then;
- O componente permite a execução de regras de tipo forward chaining e backward chaining.

emailChange:

(?person foaf:mbox ?email), strConcat (?email, ? lit),

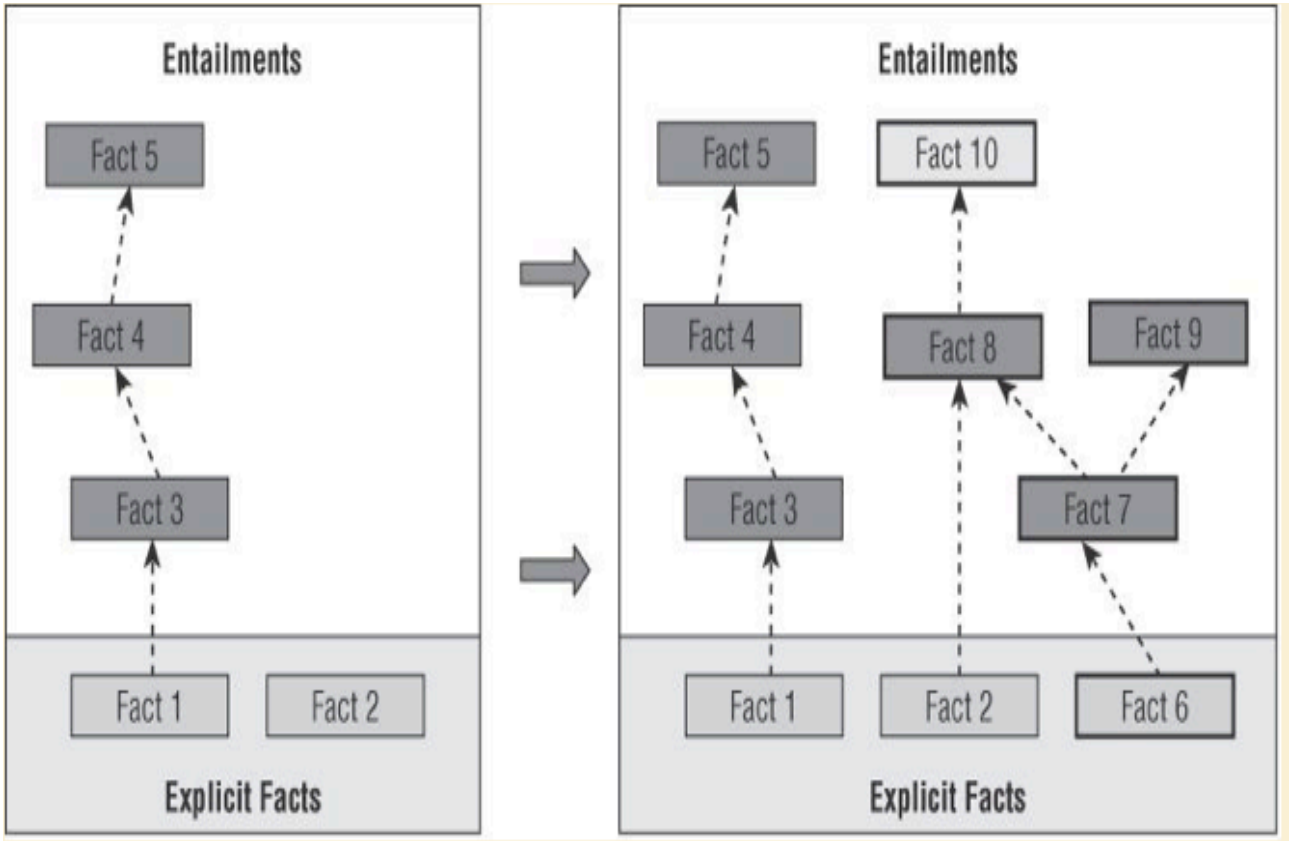
regex (? lit, '(.*gmail.com)')

→(?person rdf: type people:GmailPerson)

INSTITUTO FEDERAL
ESPIRITO SANTO

Jena e Ontologias

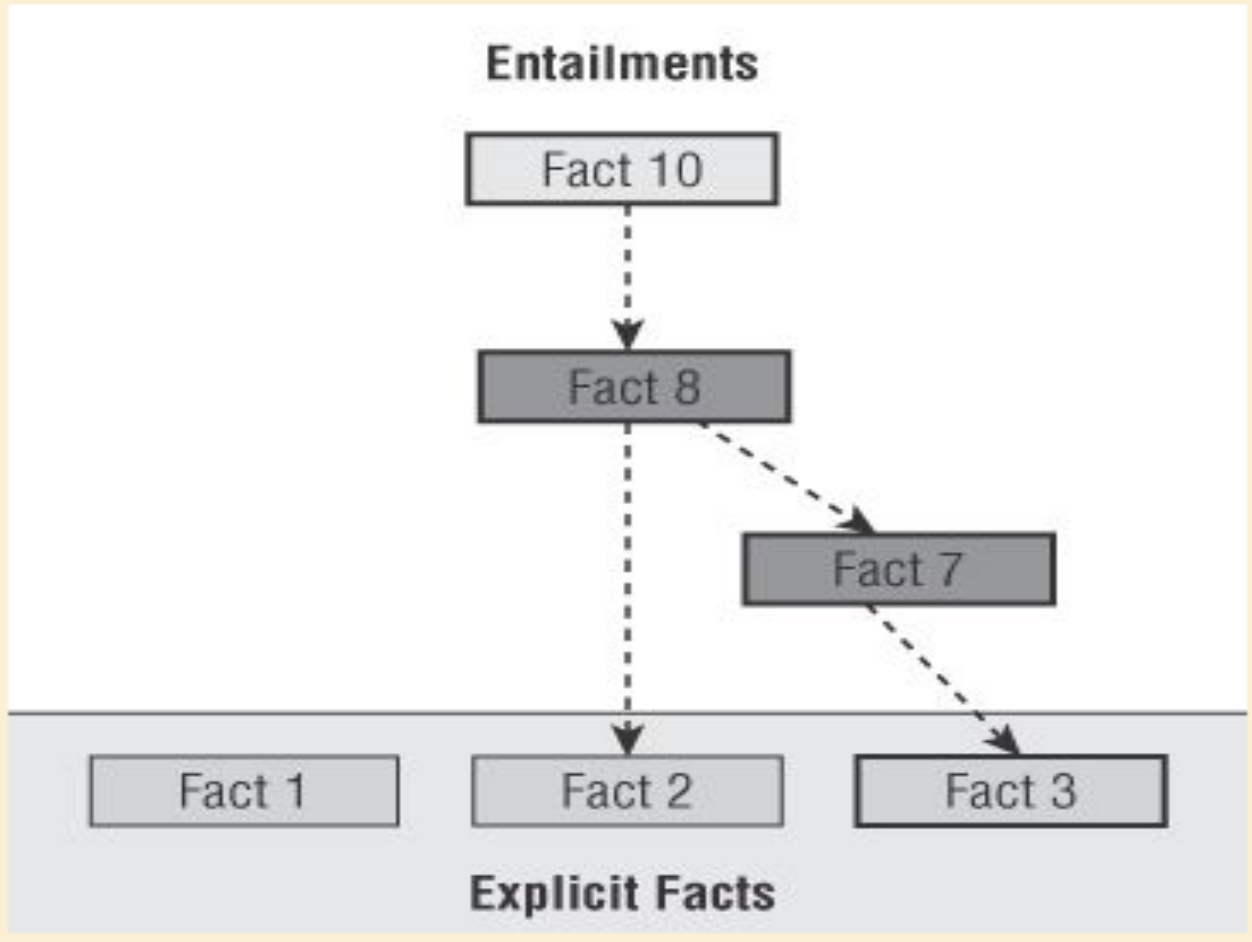
Regras : Forward Chainning



INSTITUTO FEDERAL
ESPIRITO SANTO

Jena e Ontologias

- **Regras :** Backward Chainning



Jena e Ontologias

Regras:

```
private void runJenaRule(Model model){
String rules = "[emailChange:
   (?person foaf:mbox ?email),strConcat(?email,?lit),
    regex( ?lit, "(.*@gmail.com)')
    -> (?person rdf:type> People:GmailPerson)]";

Reasoner ruleReasoner                                    =
new GenericRuleReasoner(Rule.parseRules(rules));
ruleReasoner = ruleReasoner.bindSchema(schema);
inferredFriends  =  ModelFactory.createInfModel(ruleReasoner,
 model);
}
```

Referências

[1] Hebeler, John; Fisher, Mattew; Blace, Ryan; Perez-Lopez, Andrew. 2009.**Semantic Web Programming**. Wiley Publishing, ISBN: 9780470418017.

[2] Jena, 2014. **https://jena.apache.org/**. Acessado em 23/06/2014.

[3] DuCharme, Bob. **Learing SPARQL**. 2 Ed.

[4] Gómez-Pérez, Asunción; Fernández-López, Mariano; Corcho, Oscar. 2004.**Ontology Engineering**. Spriger. ISBN: 1852335513.

INSTITUTO FEDERAL
ESPIRITO SANTO