ROTEIRO WORKSHOP LOD (LINKED OPEN DATA)

Facilitadores

Jandson Ribeiro
Lucas Lara
Matheus Lessa
Renato Silva
Tiago Gonçalves

Pré-requisitos

O participante deve ter o conhecimento prévio de:

- RDF
- Ontologias
- SPARQL

O equipamento utilizado pelo participante deve conter:

- Acesso a internet
- Protegé versão 3. 5 ou superior
- IDE Eclipse + Java JDK
- API Jena

Ao término deste workshop o participante será capaz de:

- Compreender o que são dados ligados;
- Compreender o projeto Linked Open Data;
- Consultar qualquer informação disponível na Semantic Web
- Publicar dados ligados.

PARTE I - INTRODUÇÃO (estimativa : 10 min)

1. **Linked Data**

Linked Data é o uso da Web para criar ligações entre os dados a partir de diferentes fontes. Tecnicamente, Linked Data refere-se aos dados publicados na web, de forma que seja légivel por máquina, na qual o significado esses dados são definidos explicitamentes e ligados a outros conjuntos de dados externos.

Enquanto que a unidade primária da Web de hipertexto é o HTML,documentos conectados por hiperlinks sem tipo , Linked Data conta com documentos que contêm dados em formato RDF . Como resultado temos a Web de Dados, de forma mais precisa pode ser descrito como uma teia de coisas no mundo.

2. **Vocabulários**
    a. FOAF (Friend-of-a-Friend) para descrição de pessoas e suas redes
    b. sociais.
    c. SIOC (Semantically-Interlinked Online Communities) para descrição
    d. de forums e blogs.
    e. SKOS (Simple Knowledge Organization System) para representação

f. de taxonomias de tópicos.
g. DC (Dublin Core)
h. VoiD (Vocabulary of Interlinked Datasets) para expressar metadados
i. gerais sobre datasets.
j. Organization Ontology para descrever estrutura de organizações.
k. GoodRelations para descrição de produtos e entidades de negócio.
l. Music Ontology para artistas, álbuns, e shows.
m. Review Vocabulary termos para representação de opiniões.
n.

PARTE II - LINKED OPEN DATA (estimativa: 1,2,3 : 10 min; total:50 min)

1. Sobre o projeto

Linking Open Data é uma comunidade criada pelo projeto W3C SWEO (Ligando Dados Abertos) que tem como objetivo promover a distribuição livre de dados ligados na Web, através da publicação de vários datasets (conjuntos de dados) abertos e do estabelecimento de ligações entre eles, criando uma nuvem de dados ligados (LOD Cloud).

Fatos LOD Cloud em 2007:

- DBPedia: Versão "RDFizada" da Wikipiedia;
- muitos links de entradas e saídas
- Base de dados relacionados a Música
- Grandes bases de dados incluem FOAF,dados de censo americano: Tamanho aprox. 1 billhão de triplas, 250k links.

Factos LOD Cloud em 2008:

- Mais do que 35 base de dados interligados
- Players comerciais se juntando na nuvem, p. ex.,BBC
- Organizações começam a publicar e hospedar banco de dados, p. ex. OpenLink, Talis, e Garlik.
- Tamanho aprox. 2 bilhões de triplas, 3 milhões de links

Factos LOD Cloud em 2009:

- Grande parte da nuvem Linking OpenData e o projeto BIO2RDF
- Banco de dados importantes: Freebase,OpenCalais, ACM/IEEE
- Tamanho > 10 bilhões triplas

Oito princípios da Open Data

- Completo: todos os dados públicos devem ser disponibilizados. Os dados não podem estar sujeitados a limitações de privacidade, segurança ou privilégio.
- Primário: os dados são coletados da fonte, com o mais alto nível possível de detalhamento, não de forma modificada.
- Referência de tempo: os dados devem ser disponibilizados rapidamente, pois é necessário preservar o seu valor.

- Acessível: os dados devem estar disponíveis para a maior rede possível de pessoas, para serem utilizados com as mais diversas finalidades.
- Processo automatizado: dados razoalvelmente estruturados para permitir que o processo seja automatizado.
- Não é discriminatório: os dados devem estar disponíveis a qualquer pessoa, sem necessidade de registo.
- Não possuir proprietário: os dados devem ser disponibilizados em um formato sobre o qual nenhuma entidade tem o controle exclusivo.
- Licença Livre: os dados não estão sujeitos a qualquer direito autoral, patente, marca ou regulamento de segredo comercial.

2. **Escolhendo os end nodes**
    a. DbPedia
    b. Linkedmdb
    c. Geonames
    d. Yago
3. **Consultando Linked Open Data** (estimativa: 30 min)
    a. DBpedia

O DBpedia é um esforço colaborativo da comunidade para extrair informações estruturadas do Wikipedia e fazê-las disponíveis na internet. O DBpedia te permite consultar esses dados através de queries sofisitcadas e ligar diferentes *data sets* da internet aos dados do Wikipedia.

i. DBPedia em números

- 4 milhões de coisas na versão inglesa, incluindo 832 mil pessoas, 639 mil lugares, 372 mil *creative works*(116 mil albuns de música, 78 mil filmes e 18.500 videogames), 209 mil organizações(incluindo 49 mil companhias e 45 mil instituições de educacionais), 226 mil espécies e 5.600 doenças;
- Existem versões do DBpedia em 119 línguas, totalizando 24.9 milhões de coisas;
- 24 milhões de *links* para imagens;
- 27 .6 milhões de *links* para páginas externas;
- 45 milhões de *links* dentro de outros *datasets*;
- 67 milhões para as categorias do Wikipedia;
- 41.2 milhões para as categorias do YAGO;
- Existem 2.46 bilhões de triplas RDF

ii. Datasets ligados ao DBpedia

Um artigo no DBpedia consiste de texto livre, mas também de diferentes tipos de informações estruturadas como *infobox templates*, imagens, geocoordenadas e links para páginas externas. A figura abaiso mostra o código fonte e a visualização da *infobox template* contendo as informações estruturadas a

respeito da cidade de Innsbruck.

b. Virtuoso

O Virtuoso é um servidor de dados que diferentemente dos outros bancos de dados que armazenam seus dados em linhas, colunas ou pares de chave-valor, armazena todas as informações em forma de grafos, ou seja, em uma rede de nós e arestas. As arestas representam o relacionamento entre os nós que representa os objetos. Devido aos nós e arestas serem representados como objetos (os quais os desenvolvedores estão acostumados) é possível definir atributos (também chamado de propriedades) a eles. Adicionando uma direção para uma aresta cria o conhecido grafo de propriedades que representa a explícita estrutura de dados dentro de um banco de dados de grafo. No caso de aplicações na web semântica o Virtuoso pode ser resumidamente explicado como sendo um SGD onde é possivel executar querys SPARQL e acessar dados de ontologias disponiveis na web.

Assim como os SGBD's relacionais, o Vituoso pode ser instalado localmente. Além dessa opção, também é possivel acessa-lo online no link dbpedia.org/sparql

Referência:

http://planeta-globo.com/2010/12/31/sparql-%E2%80%93-parte-i-%E2%80%93-como-instalar-o-virtuoso-banco-de-dados-de-triplas

Query 0 - query padrão do virtuoso

Por padrão, no virtuoso vem a seguinte query:

```
select distinct ?Concept where {[] a ?Concept} LIMIT 100
```

A palavra chave 'a' do Sparql é um atalho para o predicado rdf:type informando a classe de um recurso.

Query 1 - Buscando todos os filmes da base do Dbpedia

```
SELECT ?subject
WHERE {
        ?subject rdf:type <http://dbpedia.org/ontology/Film>.
}
```

A query acima retorna todos os sujeitos cujo o tipo é filme.

OBS: Experimente substituir rdf:type por 'a'. O resultado será o mesmo.

A propriedade rdf:type é proveniente do vocabulário http://www.w3.org/1999/02/22-rdf-syntax-ns#. A importação no virtuoso dos vocabulários do dbpedia e do endereço http://www.w3.org/1999/02/22-rdf-syntax-ns# é opcional, pois já são importados automaticamente pelo virtuoso.
Também para não buscarmos informações demais, podemos limitar a quantidade de resultados.
A query completa se encontra abaixo:

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

SELECT ?subject
WHERE {
?subject a <http://dbpedia.org/ontology/Film>.
}
LIMIT 100
```

Query 2 - 1ª Atividade
Retornar todos os filmes e suas respectivas estrelas

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

SELECT ?subject ?starring
WHERE {
        ?subject rdf:type <http://dbpedia.org/ontology/Film>.
        ?subject dbpedia-owl:starring ?starring.
}
LIMIT 100
```

Query 3 - 2ª Atividade
Retornar todos os filmes estrelados por Tom Cruise

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbpedia: <http://dbpedia.org/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?subject, ?film_name
WHERE {
?subject rdf:type <http://dbpedia.org/ontology/Film>.
?subject dbpedia-owl:starring ?starring.
?subject foaf:name ?film_name.
```

```
FILTER(?starring = rsc:Tom_Cruise)

}
LIMIT 100
```

Ao executar essa query ocorre a seguinte mensagem de erro:

```
Virtuoso 37000 Error SP030: SPARQL compiler, line 12: Undefined namespace prefix at '' before ')'

SPARQL query:
define sql:big-data-const 0
#output-format:text/html
define sql:signal-void-variables 1 define input:default-graph-uri <http://dbpedia.org> PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbpedia: <http://dbpedia.org/>


SELECT ?subject ?starring
WHERE {
?subject rdf:type <http://dbpedia.org/ontology/Film>.
?subject dbpedia-owl:starring ?starring.

FILTER(?starring = rsc:Tom_Cruise)

}
LIMIT 100
```

Isso ocorre pois o prefixo rsc: antes de Tom Cruise não foi definido anteriormente. A query correta segue abaixo:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbpedia: <http://dbpedia.org/>
PREFIX rsc: <http://dbpedia.org/resource/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?subject ?film_name
WHERE {
?subject rdf:type <http://dbpedia.org/ontology/Film>.
?subject dbpedia-owl:starring ?starring.
?subject foaf:name ?film_name.
```

```
FILTER(?starring = rsc:Tom_Cruise)

}
LIMIT 100
```

QUERY 4 - 3ª atividade: Quem fez mais filmes? Angelina Jolie ou Brad Pitt?

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

SELECT ?actor (COUNT(DISTINCT ?movie) as ?quantOfMovies)
WHERE {
?movie a <http://dbpedia.org/ontology/Film>.
?movie dbpprop:starring ?actor

FILTER (?actor = rsc:Brad_Pitt || ?actor =  rsc:Angelina_Jolie)
}
```

O **COUNT** é similar ao do SQL. Ele conta quantos recursos **?movie** foram retornados em relação ao recurso **?actor**.

QUERY 5 - Quem é mais velho? Tom Cruise ou Brad Pitt?

```
ASK
{
  <http://dbpedia.org/resource/Tom_Cruise> dbpedia-owl:birthDate ?tom .
  <http://dbpedia.org/resource/Brad_Pitt> dbpedia-owl:birthDate ?brad .
  FILTER(?tom > ?brad) .
}
```

A clásula ASK nos permite fazer perguntas do tipo acima e retorna um valor boolean true ou false.

Query 6 - 4ª Atividade:

Descrubra se Angelina Jolie já fez mais filmes que o seu marido Brad Pitt.

```

```

Query 7 - Como buscar todos os diretores americanos de descendência italiana

```
SELECT ?subject
WHERE {
?subject a <http://dbpedia.org/class/yago/AmericanFilmDirectorsOfItalianDescent>.
}
```

Informações específicas como essa podem ser buscadas através da base de conhecimento YAGO. No ano de 2012, o Yago tinha mais de 10 milhões de entidades e mais de 120 milhões de fatos a respeito dessas entidades. As informações do YAGO são extraídas do Wikipedia, Wordnet (dicionário de sinônimos e hiperônimos da Universidade de Princenton) e da GeoNames (base de conhecimento de informações geográficas). Veja o exemplo abaixo:

```
PREFIX rsc:<http://dbpedia.org/resource/>

SELECT ?subject ?birthDate ?birthPlace ?sameAs
WHERE {
?subject a <http://dbpedia.org/class/yago/AmericanFilmDirectorsOfItalianDescent>.
?subject  dbpedia-owl:birthDate ?birthDate.
?subject dbpprop:birthPlace ?birthPlace.
?subject owl:sameAs ?sameAs.

FILTER(?subject = rsc:Francis_Ford_Coppola)

}
```

c. Fuseki

Fuseki é um servidor SPARQL que provê atualização *REST-style SPARQL HTTP*, query *SPARQL* e atualização *SPARQL* usando o protocolo do SPARQL via HTTP.

Instalação

Baixe o arquivo jena-fuseki-1.0.0-distribution.zip em:
http://www.apache.org/dist/jena/binaries/.

Configuraçao:

1. Descompacte o arquivo

2. Abra o terminal
3. Vá até a pasta onde os arquivos foram descompactados
4. execute o seguinte comando:

    ./fuseki-server --update --mem /ds
5. Abra o navegador e digite http://localhost:3030
6. Escolha a opção *control panel*
7. selecione a opção ds (única diponível)

Para mais informações consulte http://jena.apache.org/documentation/serving_data/#download-fuseki.

Query 8 - Buscando filmes e seus locais de gravação

O http://data.linkedmdb.org/ contém dados abertos sobre filmes. Podemos, por exemplo, listar os filmes o seus locais de gravação.

Não podemos, no entanto, utilizar o virtuoso para obtermos os dados desta consulta. Pois, este consulta somente os dados presentes no DBpedia. Para fazermos uma consulta no data.linkedmdb.org deveremos utilizar um outro servidor, como o Fuseki.

Agora podemos consultar o data.linkedmdb.org. O endereço de seu SPARQL endpoint é http://data.linkedmdb.org/sparql. E é a partir dele que faremos a nossa consulta, para isto, é suficiente utilizarmos a cláusula SERVICE que permite que enviemos a query dentro de seu bloco para um servidor em específico. Observe o exemplo abaixo.

Buscar filmes e seus respectivos locais de gravação:

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX movie: <http://data.linkedmdb.org/resource/movie/>
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>

SELECT ?label  ?locname WHERE {
        SERVICE <http://data.linkedmdb.org/sparql> {
        ?film a movie:film .
        ?film rdfs:label ?label .
        ?film  movie:featured_film_location  ?location.
        ?location movie:film_location_name ?locname.
}
```

Query 9 - Retornando informações de duas bases de conhecimento distintas: DBpedia e Linkedmdb

Imaginemos agora que queiramos saber as sinopses dos filmes. Observe, no entanto, que o http://data.linkedmdb.org/ não tem tal informação. O DBPedia, por outro lado, tem estas informações, podemos assim, fazer uma ligação entre os dois *datasets* a fim de obter tais informações.

Para isto, precisamos ligar os dois *datasets*. Isto pode ser feito através da propriedade **sameAs.** Para fazermos a ligação com o DBPedia, é suficiente adicionarmos mais um SERVICE apontando para o mesmo e escrever a consulta a fim de retornar as propriedades desejadas.

Nome dos filmes, seus locais de gravação e comentários sobre os mesmos.

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX movie: <http://data.linkedmdb.org/resource/movie/>
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbpprop: <http://dbpedia.org/property/>

SELECT ?label  ?locname ?comment WHERE {
        SERVICE <http://data.linkedmdb.org/sparql> {
        ?film a movie:film .
        ?film rdfs:label ?label .
        ?film  movie:featured_film_location  ?location.
        ?location movie:film_location_name ?locname.
        ?film owl:sameAs ?dbpediaLink.

FILTER(regex(str(?dbpediaLink), "dbpedia", "i")).


}
SERVICE <http://dbpedia.org/sparql> {
        ?dbpediaLink rdfs:comment ?comment.

        }

}

LIMIT 50
```

Querry 10 - 4ª Atividade

Utilizando o Linkedmdb e o DBpedia, para o filme Casino Royale, retorne o título do filme, onde o mesmo foi baseado, o orçamento e o nome dos atores.

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
```

```
PREFIX movie: <http://data.linkedmdb.org/resource/movie/>
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbpprop: <http://dbpedia.org/property/>
PREFIX rsc:<http://dbpedia.org/resource/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?label ?bnear  ?actorName ?nomes WHERE {
        SERVICE <http://data.linkedmdb.org/sparql> {
        ?film a movie:film .
        ?film rdfs:label ?label .

         OPTIONAL{
              ?film foaf:based_near ?bnear.
              ?film movie:actor ?actor.
             ?actor movie:actor_name ?actorName.
          }

?film owl:sameAs ?dbpediaLink.
FILTER(regex(str(?dbpediaLink), "dbpedia", "i")).

}

SERVICE <http://dbpedia.org/sparql> {
        ?dbpediaLink rdfs:label ?nomes.

}


}

LIMIT 100
```

d. Usando Jena

A API Jena nos permite realizar consultas SPARQL em endpoints distintos. Como a API é em Java e podendo ser utilizada em qualquer aplicação Java, a mesma pode ser utilizada para alimentar aplicações Java com dados abertos e ligados.

Exemplo Jena - Projeto

Já sabemos que o DBpedia é uma base de conhecimentos que extrai informações do Wikipedia. Sabemos também que o Linkedmdb é a base de conhecimento do Imdb, um dos maiores e melhores sites de informações cinéfilas do mundo. O projeto XXX propõe a união dessas duas bases combinando seus conjuntos de dados.

[mostrar o projeto que ainda não existe, hehe]
[mostrar os trechos importantes de códig

4. **Publicação de dados ligados** (estimativa: 5 min)
    a. Ferramentas
        i. Triplify
            - Plugin para aplicações web PHP, Ruby/Python
            - Disponibiliza dados de bancos de dados relacionais em triplas (RDF, JSON e Linked Data)
            - “SELECT id, name AS ‘foaf:name’ FROM users”
            - imagem db_to_tripladb_to_tripla
            - tem um exemplo aqui
        ii. D2RQ e D2R Server
            - Permitir que aplicações consigam ter uma RDF-view em um banco de dados non-RDF
            - Plataforma desenvolvida em Java para publicação de dados de BD em grafo RDF (RDF/XML,N3, N-TRIPLE)
            - Suporta Oracle, MySQL, PostGreeSQL, Microsoft SQL Server, fontes de dados ODBC)
            - Consultas em SPARQL
            - D2RQ Mapping Language
                a. Lingaugem declarativa que descreve o mapeamento
            - D2R Server
                a. Servidor HTTP que fornece uam visão Linked datae permite constultas SPARQL
            - D2RQ Engine
                a. Plug in para Jena
        iii. Virtuoso RDF Views
            - Dados relacionais em RDF e expô-lo no Virtuoso-hosted SPARQL endpoint.
            - Gera RDF/XML
            - Oracle, MS Server, DB2, Informix, Progress, MySQL, Ingres, Firebird, PostgreSQL e ODBC ou JDBC

APÊNDICE A - VOCABULÁRIO

PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX movie: <http://data.linkedmdb.org/resource/movie/>
PREFIX dcterms: <http://purl.org/dc/terms/>

PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbpprop: <http://dbpedia.org/property/>
PREFIX rsc:<http://dbpedia.org/resource/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

Propriedade

DBPedia

http://dbpedia.org/ontology/Film = todos os filmes

Filme
dbpedia-owl:starring = estrelas de um determinado filme
foaf:name = nome do filme

Ator

dbpedia-owl:birthDate data de nascimento
dbpprop:birthPlace Local de nascimento

LinkedMDB

http://data.linkedmdb.org/resource/movie/

film = todos os filmes

rdfs:label = rotulo da entidade (Filme e Ator)
movie:featured_film_location = local de gravação do filme (geolocation)
movie:film_location_name = nome do local de gravação do filme