Disciplina: Paradigmas de Programação **Semestre: 2025/2**

Trabalho - Programação Lógica

Objetivo: Aprender o básico sobre programação lógica e suas aplicações, em particular com a utilização de dados disponíveis na Web como base de fatos.

Crie um programa em Prolog que:

(1) busque por (uma quantidade razoável de) fatos na Nuvem de Dados Ligados, por exemplo, Dbpedia, usando uma consulta SPARQL conforme exemplo em anexo.
(2) O exemplo em anexo faz consultas em uma base de dados sobre Filmes. Escolham outro assunto (séries, jogos, esportes, etc) para as consultas de vocês.
(3) contenha pelo menos 8 regras significativas que usem esses dados para produzir novas informações.

- regras significativas devem produzir novas informações relevantes, e não ser apenas regras auxiliares que fazem algo muito trivial.
  Exemplo: naoAma(X,Y) :- \+ama(X,Y) <<< *Não é significativa*
  idoso(X) :- pessoa(X), idade(X,I), I > 60 <<< *Não é significativa*
  temCiumes(X,Y) :- ama(X,Z), ama(Y,Z), dif(X,Y), dif(X,Z) << Ok!
- as regras devem usar ***pelo menos 3*** dos os conceitos de Prolog a seguir:
  - negação, corte, listas, strings, -> (se então), aggregate_all, find_all

(4) forneça ***pelo menos 2*** exemplos de consultas "abertas" e "fechadas" ***para cada regra***:

  Exemplo: ?- ama(X,Y) <<< *Aberta*
  ?- ama(X,mia) <<< *Aberta*
  ?- ama(vincent,mia) <<< *Fechada*
- use of formato de exemplos do swish:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%% CONSULTAS PRE-DEFINIDAS %%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/** <examples>

?- loves(X, mia).
?- jealousOf(X, Y).
?- safeFriendship(vincent, marcellus).
?- sad(X).

*/
```

O trabalho deve ser feito em dupla e entregue até o dia 10/12/2025 às 23h59m59s. No dia 11/12/2025, durante o horário da aula, as duplas irão passar por uma breve entrevista (~5 minutos). A ordem de apresentação será sorteada e divulgada no próprio dia 11.

Envie um arquivo texto nome EA2_aluno1_aluno2.pl (pl é a extensão dada aos arquivos com programas em prolog) via e-mail blegora@inf.ufes.br no prazo estabelecido. O arquivo deve conter comentários explicando cada regra.

Obs: Plágio não será tolerado. Suspeitas de plágio serão questionadas durante a entrevista. Ao menos uma pergunta específica das regras será feita para cada membro da dupla.

Disciplina: Paradigmas de Programação **Semestre: 2025/2**

Trabalho - Programação Lógica

Anexo

Prolog (swish) possui suporte à consultas SPARQL, que é uma tecnologia básica no desenvolvimento da web semântica (interatividade entre pessoas e computadores), fornecendo consultas à bases de dados. Outros exemplos podem ser vistos no link acima.

O exemplo abaixo simplifica muito a utilização do SPARQL, apenas o suficiente para vocês realizarem o EA. Um exemplo de utilização de SPARQL em prolog (para consultar dados de filmes) é o seguinte:

```
%% este exemplo busca Filmes na base de dados dbpedia
:- data_source(
dbpedia_filmes,
sparql("PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbow: <http://dbpedia.org/ontology/Work/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

select distinct ?filme ?minutos ?orcamento ?diretor ?nascimento ?morte
   where {
       ?film_uri a dbo:Film;
               rdfs:label ?filme;
               dbow:runtime ?runtime;
               dbo:budget ?budget;
               dbo:director ?director_uri.
       ?director_uri rdfs:label ?diretor;
           dbo:birthDate ?nascimento.
       OPTIONAL {?director_uri dbo:deathDate ?morte}
   BIND (xsd:float(?runtime) as ?minutos)
   BIND (xsd:float(?budget) as ?orcamento)
   FILTER (lang(?filme) = 'en')
   FILTER (lang(?diretor) = 'en') }
   ORDER BY DESC(?minutos) ?filme
",
[ endpoint('https://dbpedia.org/sparql')])  ).

% este predicado associa as "colunas" de uma base de dados com variáveis Prolog.
filmes(Filme, Min, Orcam) :- distinct([Filme,Min,Orcam],
             (dbpedia_filmes{filme:Filme, minutos:Min, orcamento:Orcam})).

% este predicado associa as "colunas" de uma base de dados com variáveis Prolog.
filmesDiretor(Filme, Diretor, Nascimento, Morte) :-
           dbpedia_filmes{filme:Filme, diretor:Diretor,
                                nascimento:Nascimento, morte:Morte}.
```

No exemplo acima, considere que estamos consultado a base de dados dbpedia. Apenas o contéudo dentro da “função” sparql será comentado. Como dito, consultas SPARQL tem relação com a web semântica, então esses dados irão ser extraídos da web. Inicialmente, são definidos alguns apelidos para URLs longas:
PREFIX **rdf**: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX **rdfs**: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX **dbo**: <http://dbpedia.org/ontology/>

Disciplina: Paradigmas de Programação **Semestre: 2025/2**

Trabalho - Programação Lógica

PREFIX **dbow**: <http://dbpedia.org/ontology/Work/>
PREFIX **xsd**: <http://www.w3.org/2001/XMLSchema#>
Esses prefixos são usados para interpretar e/ou acessar algumas informações extraídas dos sites. Por exemplo, se os dados estão em http://dbpedia.org/ontology/Film, digitamos apenas dbo:Film.

Os principais pontos da consulta é o trecho “select distinct”. Nele, são definidas as variáveis:

- filme
- minutos
- orcamento
- diretor
- nascimento
- morte

Obs: As definições são representadas por ?<nome_var> na consulta. Acesse o link de Film ali em cima. Você verá alguns exemplos de Filmes que estão cadastrados na dbpedia. Um exemplo é o filme http://dbpedia.org/resource/A_Better_Tomorrow. Se você conseguir acessar ele (costuma ficar bastante em manutenção...), você verá algumas características, por exemplo:

- dbo:director – Diretor (no caso, é um link)
- dbo:budget – orçamento
- dbo:gross – valor arrecadado
- rds:label – o nome do conteúdo (no caso, o filme)
- dbo:Work/runtime (ou dbow:runtime) – a duração

O seguinte trecho (vocês provavelmente irão modificar) faz:

```
?film_uri a dbo:Film;
                rdfs:label ?filme;
                dbow:runtime ?runtime;
                dbo:budget ?budget;
                dbo:director ?director_uri.
```

Define “?film_uri” como sendo uma amostra do tipo “dbo:Film”. Dessa amostra (repare que film_uri não é usado para nada, mas é necessário para “filtrar” os filmes), são recuperadas 4 informações: o nome do filme (?filme), a duração (?runtime), o orçamento (?budget) e o endereço para a página do diretor. Em seguida, “exploramos” a URI do diretor:

```
?director_uri rdfs:label ?diretor;
           dbo:birthDate ?nascimento.
       OPTIONAL {?director_uri dbo:deathDate ?morte}
```

E extraímos 2 (ou 3) informações da URI do diretor: o nome (?diretor) do campo “rdf:label”, a data de nascimento (?nascimento) e opcionalmente (se houver, no caso), a data de falecimento do diretor (?morte). No caso do filme “A Better Tomorrow”, o diretor ainda é vivo e essa informação não está disponível.

Obs: o nome dos campos é descrito na página da dbpedia. Vocês que irão escolher o tema e os campos que serão acessados/recuperados.

Trabalho - Programação Lógica

Em seguida, são feitos alguns processamentos:

```
BIND (xsd:float(?runtime) as ?minutos)
   BIND (xsd:float(?budget) as ?orcamento)
   FILTER (lang(?filme) = 'en')
   FILTER (lang(?diretor) = 'en') }
   ORDER BY DESC(?minutos) ?filme
```

As duas primeiras linhas fazem um “bind” (como se fosse um cast), da string contida em ?runtime para o valor em float ?minutos. Faz o mesmo para orçamento.

As linhas de filtragem, como o próprio nome diz, restringem os resultados para aqueles cujo nome do filme (?filme) e nome do diretor (?nome) estejam marcados como sendo do idioma “en”. A função lang é nativa do SPARQL e retorna a tag de idioma da variável que ela recebe.

Por último, temos uma linha que ordena os filmes por ordem decrescente de minutos. Em caso de empate, utiliza o título para desempatar (em ordem “crescente”).

Obs: Para o caso onde há mais de um valor em uma variável (ex: 2 diretores), a consulta retorna duas entradas, onde uma possui um diretor.

Obs2: Você pode testar suas consultas SPARQL no site https://yasgui.triply.cc/ antes de usá-las no prolog.

Obs3: Você deve escolher o seu tema. Após, procure “Tema dbpedia” no google e veja se aparece algo da dbpedia/ontology.

Alternativamente, você pode fazer consultas para encontrar um tema:

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT DISTINCT ?class ?label
WHERE {
  ?class a owl:Class ;
        rdfs:label ?label .
  FILTER (lang(?label) = 'en') # se quiser trocar o idioma...
  FILTER (CONTAINS(LCASE(?label), "movie")) # a palavra-chave aqui
}
LIMIT 100
```