

Lista de Exercícios

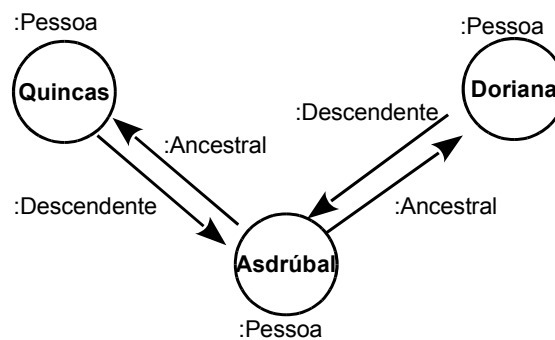
MC536 - Bancos de Dados: Teoria e Prática
Instituto de Computação
Universidade Estadual de Campinas

Bancos de Dados de Grafos
Ontologias e Web Semântica
2014
André Santanchè

Questão 1 - Cypher

Considere um grafo criado em Cypher que registra relações de parentesco entre pessoas. Cada nó rotulado como `:Pessoa` representa uma pessoa; se uma pessoa X é filha(o) de outra pessoa Y, X possui uma aresta rotulada como `:Ancestral` para Y e Y possui uma aresta `:Descendente` para X. Veja a seguir um exemplo:

```
CREATE (as:Pessoa {id: "Asdrubal"}),  
(qu:Pessoa {id: "Quincas"}),  
(dr:Pessoa {id: "Doriana"}),  
(as)-[:Descendente]->(qu),  
(qu)-[:Ancestral]->(as),  
(as)-[:Descendente]->(dr),  
(dr)-[:Ancestral]->(as)
```



Escreva sentenças em Cypher para atender às seguintes requisições:

- Dada uma pessoa X, retorne seus ancestrais.
- Dada uma Pessoa X, retorne todos os irmãos de X.
- Dada uma Pessoa X, retorne todos os primos de X.
- Dadas duas pessoas X e Y, indique todos os descendentes comuns entre elas, se existirem.
- No atual modelo não é possível distinguir o pai da mãe de uma pessoa, já que ambos possuem o mesmo tipo de aresta. Modifique o modelo do grafo para que seja possível distingui-los, mas de tal modo que os algoritmos anteriores (da letra a até a letra d) continuem funcionando sem modificações. A partir destas modificações responda os dois itens subsequentes.
- Dada uma Pessoa X, retorne o seu pai.
- Dada uma Pessoa X, retorne seus primos por parte de mãe.

Questão 2 - RDF / SPARQL

Considere que o grafo da questão anterior está escrito em RDF. Para isto é definido o namespace padrão conforme segue:

```
@base <http://purl.org/andresantanche/learning/graph/examples> .
```

Por ser o namespace padrão, todas as URIs que o usam são precedidas apenas de dois-pontos.

O rótulo `:Pessoa` da questão anterior se tornou uma classe RDF e os rótulos `:Descendente` e `:Ancestral` se tornaram propriedades como segue:

```
:Pessoa rdf:type rdfs:Class .
:ancestral rdf:type rdfs:Property .
:descendente rdf:type rdfs:Property .
```

No exemplo da questão anterior, Asdrubal, Quincas e Doriana são instâncias de pessoas, conforme segue:

```
:Asdrubal rdf:type :Pessoa ;
           :ancestral :Doriana ,
             :Quincas .

:Doriana rdf:type :Pessoa ;
         :descendente :Asdrubal .

:Quincas rdf:type :Pessoa ;
         :descendente :Asdrubal .
```

- Escreva em SPARQL sentenças para responder os itens (a) até (g) da questão anterior. Identifique se há alguma query que você não é capaz de responder em SPARQL e justifique a sua resposta.
- Como regras de inferência poderiam ser utilizadas tanto para simplificar queries da letra (a), quanto para permitir que sejam executadas queries que não foram possíveis com o SPARQL (considerando que o SPARQL será capaz de acessar os resultados da inferência).

Questão 3 - Modelo Grafo x Relacional

- Elabore um modelo relacional equivalente ao modelo de grafo da Questão 1 e ao modelo RDF da Questão 2. Comente que diferenças podem haver entre ambos.
 - Escreva uma consulta SQL equivalente à letra (b) da Questão 1.
 - Identifique se há alguma query da Questão 1 que você não é capaz de responder em SQL e justifique a sua resposta.
-

Questão 4 (questão de prova)

Considere os comandos SQL a seguir para criar tabelas que controlam produtos e receitas, bem como o respectivo esquema relacional simplificado. A tabela de `Produto` mantém um cadastro de produtos, com seu código e nome. Cada `Receita` tem um código e nome. Cada entrada nesta tabela `Ingrediente` indica que um `Produto` é componente de uma `Receita`.

<pre>CREATE TABLE Produto (codigo_produto VARCHAR(5), nome_produto VARCHAR(80), PRIMARY KEY (codigo_produto));</pre>	<pre>CREATE TABLE Ingrediente (codigo_receita VARCHAR(5), codigo_produto VARCHAR(5), PRIMARY KEY (codigo_receita, codigo_produto), FOREIGN KEY (codigo_receita) REFERENCES Receita (codigo_receita), FOREIGN KEY (codigo_produto) REFERENCES Produto (codigo_produto));</pre>
<pre>CREATE TABLE Receita (codigo_receita VARCHAR(5), nome_receita VARCHAR(80), PRIMARY KEY (codigo_receita));</pre>	<p>Esquema Relacional:</p> <p><code>Produto</code>(<u>codigo_produto</u>, nome_produto) <code>Receita</code>(<u>codigo_receita</u>, nome_receita) <code>Ingrediente</code>(<u>codigo_receita</u>, <u>codigo_produto</u>)</p>

Considere os comandos INSERT abaixo para inserir dados nas tabelas acima:

```
INSERT INTO Produto values ('BOLO', 'Bolo chocolate');
INSERT INTO Produto values ('SORV', 'Sorvete de creme');
INSERT INTO Receita values ('PETI', 'Petit Gateau');
INSERT INTO Ingrediente values ('PETI', 'BOLO');
INSERT INTO Ingrediente values ('PETI', 'SORV');
```

A partir do esquema e inserções apresentados:

a) Escreva uma sequência de comandos em Cypher para construir um grafo, com dados e relações equivalentes àqueles inseridos nas tabelas `Produto`, `Ingrediente` e `Receita`.

b) Considere o seguinte comando SQL:

```
SELECT R.nome_receita FROM Receita R, Ingrediente I, Produto P
WHERE R.codigo_receita = I.codigo_receita and
      I.codigo_produto = P.codigo_produto and
      P.nome_produto = "Sorvete de Creme";
```

Escreva uma consulta em Cypher que produza um resultado equivalente a partir dos dados do grafo. É importante notar que além dos dados inseridos no exemplo, pode haver mais dados seguindo o modelo e a consulta deve ser capaz de considerá-los.

Referência Cypher

```
CREATE (var:Label { id: "value", ... } ),
      (var)-[var:Label]->(var), ...

MATCH (var:Label) WHERE <condition>

MATCH (var:Label)-[var:Label]->(var:Label) WHERE <condition>

RETURN <variável/propriedades>, ...

<condition> - expressões lógicas com <variável/propriedades> e <constantes>
             <variável/propriedade> - <var>.<propriedade>
             <constante> - "string"
```