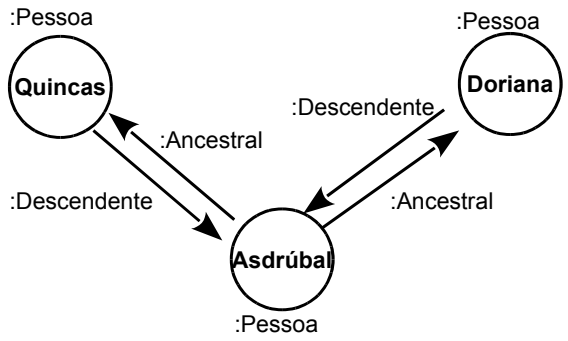
Lista de Exercícios

MC536 – Bancos de Dados: Teoria e Prática Instituto de Computação Universidade Estadual de Campinas		Bancos de Dados de Grafos Ontologias e Web Semântica 2015 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)-[:Ancestral]->(qu),
       (qu)-[:Descendente]->(as),
       (as)-[:Ancestral]->(dr),
       (dr)-[:Descendente]->(as)
```



Escreva sentenças em Cypher para atender às seguintes requisições:

a) Dada uma pessoa X, retorne seus acestrais.

b) Dada uma Pessoa X, retorne todos os irmãos de X.

c) Dada uma Pessoa X, retorne todos os primos de X.

d) Dadas duas pessoas X e Y, indique todos os descendentes comuns entre elas, se existirem.

e) 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.

f) Dada uma Pessoa X, retorne o seu pai.

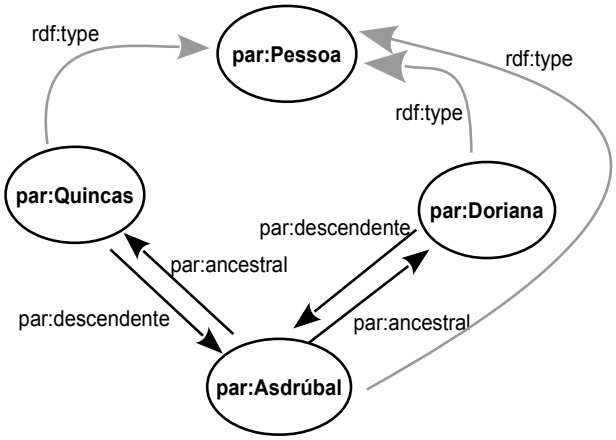
g) Dada uma Pessoa X, retorne seus primos por parte de mãe.

Questão 2 – RDF / SPARQL

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

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

O rótulo `:Pessoa` da questão anterior se tornou uma classe `par:Pessoa` RDF e os rótulos `:Descendente` e `:Ancestral` se tornaram as propriedades `par:ancestral` e `par:descente`. O grafo RDF correspondente está ilustrado a seguir.



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

```
par:Adrubal rdf:type par:Pessoa .
par:Adrubal par:ancestral par:Doriana .
par:Asdrubal par:ancestral par:Quincas .

par:Doriana rdf:type par:Pessoa .
par:Doriana par:descendente par:Adrubal .

par:Quincas rdf:type par:Pessoa .
Par:Quincas par:descendente :Adrubal .
```

a) 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.

b) 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

a) 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.

b) Escreva uma consulta SQL equivalente à letra (b) da Questão 1.

c) 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`.

```
CREATE TABLE Produto (
  codigo_produto VARCHAR(5),
  nome_produto VARCHAR(80),
  PRIMARY KEY (codigo_produto)
);
```

```
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)  );
```

```
CREATE TABLE Receita (
  codigo_receita VARCHAR(5),
  nome_receita VARCHAR(80),
  PRIMARY KEY (codigo_receita)  );
```

Esquema Relacional:
Produto(codigo_produto, nome_produto)
Receita(codigo_receita, nome_receita)
Ingrediente(codigo_receita, codigo_produto)

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"
```