Efficient Query Processing in RDF Databases

AUTORES:

Andrey Gubichev
Munich, Germany

Thomas Neumann
Munich, Germany

José Ramalho
Apresentação capítulo 5
Curitiba, 10 de novembro de 2016

Introdução

Itens Abordados:

- Análise das consultas SPARQL.
- Tipos de indexação de dados RDF.
- Formas de armazenamento.
- Exemplos de sistemas RDF.

Introdução

Considerações Iniciais:

- Aumento expressivo na web nos últimos anos do Linked Data.
- Existência de diversos conjuntos de dados contendo milhões de triplas: biomedicina (Uniprot, PubMed), bases de conhecimento (DBpedia, YAGO), governo (Data.gov), entretenimento (MusicBrainz), etc.

De acordo com **Linked Open Data** existem mais de

31 bilhões

de triplas publicadas online

Introdução

Motivação:

- Números expressivos de recursos RDF indicando a necessidade da busca por uma **indexação eficiente**.
- Preocupação com o processamento de consultas SPARQL.
- Dificuldades identificadas no RDF: **armazenamento, indexação e consulta**.

Desafios

Como indexar os diversos conjuntos de dados RDF publicados na Web?

A ausência de um esquema global e a diversidade de recursos torna esta tarefa desafiadora.

Desafios

Como encontrar o plano de execução ideal para consultas SPARQL?

Uma vez que o modelo de dados consiste essencialmente de triplas, a extração de dados algumas vezes necessita de um grande número de junções.

Desafios

Como otimizar consultas com estimativas precisas de resultados?

Isso normalmente requer estruturas de dados específicas que manipulam as estimativas de **seletividade** e **cardinalidade** para dados RDF.

Desafios

Como executar com eficiência as junções e os caminhos a percorrer?

Mesmo depois de encontrar o plano de execução ideal, diversas junções precisam ser executadas. Os caminho percorrido também tornar-se um desafio em grandes conjuntos de dados, uma vez que acessam diversas informações.

Armazenamento de dados RDF

Duas possíveis abordagens de acordo com os autores:

- Baseada no Modelo Relacional
- Baseado em Grafos

Armazenamento de dados RDF

Abordagem baseada no modelo relacional

A maioria dos sistemas mapeiam o conjunto de dados RDF para tabelas relacionais e utilizam três possíveis estratégias: **Triple Stores**, **Tabelas de Propriedades e Tabelas de Propriedades Clusterizadas.**

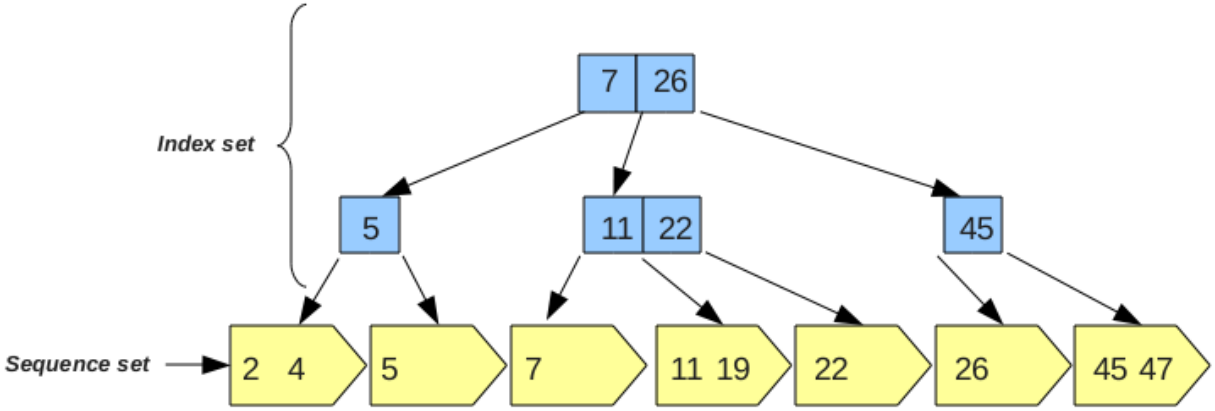
Armazenamento de dados RDF

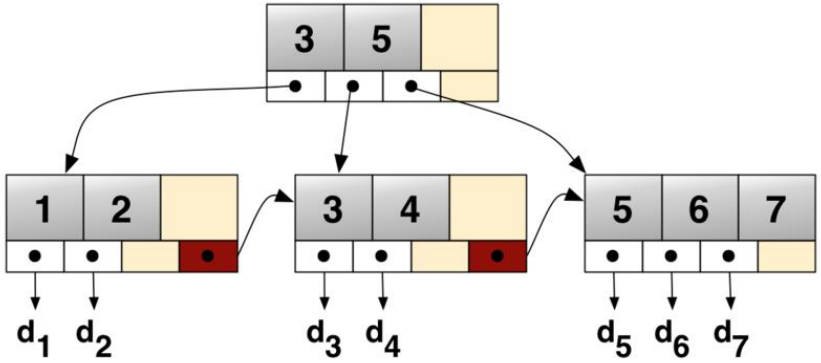
Estratégia 1: Triple Stores

- As triplas são salvas em grandes tabelas cotendo sujeito, predicado e objeto.
- Em alguns sistemas é criada uma **coluna adicional** para salvar o nome do grafo que contém a tripla.

Indexação

Exemplos Árvores B+





Indexação

Baseado no modelo relacional

Cinco estratégias de índices usando ***B+ tree***:

- Índices **separados** para o S, P e O;
- Índice **agrupado** com o S e P;
- Índice **agrupado** com o S e P e o índice do O **separado**;
- Índice **agrupado** com o P e O;
- Íncide **agrupado** com o S, P e O

S: Sujeito
P : Predicado
O: Objeto

Indexação

Baseado no modelo relacional

Experiências com **consultas simples** ou **consultas com poucas junções** mostraram que a primeira estratégia obteve melhor desempenho, no caso, **índices separados para o S, P e O**;

S: Sujeito
P : Predicado
O: Objeto

Indexação

Estratégia 1: Triple Stores

Umas das primeiras propostas para melhorar o desempenho das consultas foi usar **índices não clusterizados** com diversas combinações de **Sujeito, Predicato e Objeto** salvos em *triples table*.

Indexação

No modelo relacional

Quando uma consulta tiver uma complexidade maior, **índices clusterizados** são indicados pois possuem uma performance melhor.

Indexação


Criação de novos índices
Indexação Excessiva
Espaço em Disco

Indexação

Compressão de Dados em Banco de Dados

Para manter os requisitos de espaço em disco devido a indexação em larga escala, os sistemas normalmente usam a **Compressão de Dados por Dicionário**.

Indexação

Compressão de Dados em Banco de Dados

A compressão de dados é o ato de reduzir o espaço ocupado por dados num determinado sistema. Essa operação pode ser realizada através de diversos algoritmos de compressão, reduzindo a quantidade de bytes para representar um dado, podendo esse dado ser um recurso RDF.

Indexação

Como funciona a compressão de dados por dicionário?

- Cada URI de uma tripla é mapeado para um **ID exclusivo**.
- Os dados são salvos como triplas de inteiros e não como strings.
- Duas maneiras de atribuir os IDs: incremental ou através de função hash.

Indexação

Como funciona a compressão de dados por dicionário?

- **ID de forma incremental**: para cada novo dado que deve ser salvo no BD é requisitado o próximo ID incremental disponível.
- O mapeamento das strings para os IDs e dos IDs para as strings é salvo em uma estrutura de árvore B+.

Indexação

Como funciona a compressão de dados por dicionário?

- Segunda opção para geração de IDs é através de **função hash**.
- Um ID para a String é gerado por um hash.
- Apenas o mapeamento do ID para a sequência de caracteres precisa ser armazenado.

Indexação

Exemplos de implementações

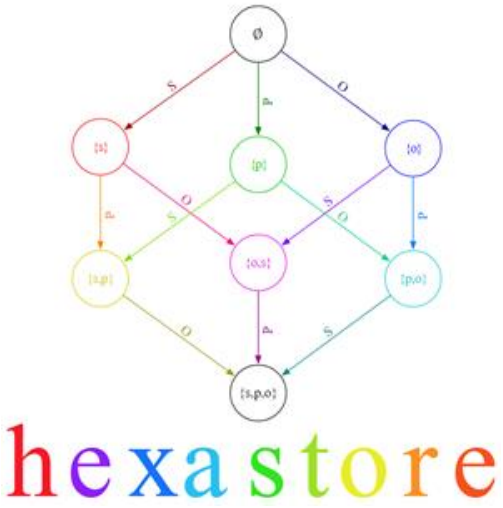
- Virtuoso: emprega um esquema de indexação parcial, onde apenas dois índices quádruplos (PSOG, POGS) são armazenados.
- Além disso, três índices duplos são mantidos: SP, OP e GS.

S: Sujeito
P : Predicado
O: Objeto
G: Grafo

Indexação

Exemplos de implementações

- Hexastore: emprega seis diferentes índices de árvores B+ agrupados em todas as possíveis permutações de S, P e O: SPO, SOP, PSO, POS, OSP, OPS.



S: Sujeito
P : Predicado
O: Objeto

Indexação

Exemplos de implementações

- RDF-3X: emprega seis diferentes índices de árvores SPO, SOP, PSO, POS, OSP, OPS além dos subconjuntos SP, SO, PS, PO, OS, OP resultando em seis índices adicionais.

S: Sujeito
P : Predicado
O: Objeto

Armazenamento de dados RDF

Estratégia 2 e 3: Property e Cluster-property Tables

Os autores analisaram o armazenamento de dados RDF em tabelas que são definidas por um ou vários predicados.

Armazenamento de dados RDF

Baseado no modelo relacional

Estratégia 2: Property Tables

- As triplas são agrupadas pelo nome do predicado.
- Todas as triplas com o mesmo predicado são armazenadas em uma tabela separada.

Armazenamento de dados RDF

Baseado no modelo relacional

Estratégia 3: Cluster-property Tables

- As triplas são agrupadas em classes com base na similaridade dos predicados.
- Cada tripla é armazenada em uma tabela que corresponde a sua classe.
- As classes são normalmente determinadas por um algoritmo de agrupamento.

Armazenamento

Property e Cluster-property Tables

Abordagem Extrema

- Usa uma tabela separada para cada (P).
- Sujeitos são conectados a vários objetos através do mesmo (P) e resultam em múltiplas linhas com o mesmo (S) e diferentes (O)

S: Sujeito
P : Predicado
O: Objeto

Armazenamento

Property e Cluster-property Tables

Abordagem Extrema

- A coluna do sujeito é classificada para permitir *joins* otimizados.
- Os objetos podem ser indexados através de árvores B+.

Armazenamento

Cluster-property Tables

Exemplo

- Implementação da Apache em Java que suporta tabelas de propriedades *clusterizadas*.

Armazenamento e Indexação

Abordagem baseada em grafos

- Os autores abordam de **forma sucinta** o assunto.
- Limitaram em conceituar o G-Store: sistema em que o conjunto de dados RDF é tratado como grafos.

Armazenamento e Indexação

G-Store

- Sistema de gerenciamento de dados RDF.
- Mantém a estrutura de grafo de acordo com o dado RDF original.
- O modelo de dados é baseado em grafos com várias arestas onde cada vértice corresponde a um sujeito ou objeto.

Armazenamento e Indexação

G-Store

- Possui um algoritmo que particiona o grafo em subgrafos e organiza os subgrafos no disco.
- Minimiza a distância no disco entre vértices adjacentes.
- Implementa um índice chamado VS-Tree que é uma árvore de altura balanceada com técnicas para pesquisa de dados nos subgrafos.

Processamento e Otimização de Consultas SPARQL

- Traduzindo consultas SPARQL em grafos de consulta
- Melhor forma para otimizar consultas
- Análise de junções e operadores

Traduzindo Consultas SPARQL

Para uma consulta SPARQL, o mecanismo RDF (*engine*) executa os seguintes passos:

I. Analisa a consulta e constrói a AST (*);
II. Traduz a AST para o **grafo de consulta**;
III. Constrói o plano de consulta otimizado a partir do grafo de consulta;
IV. Executa o plano de consulta.

(*) Árvore de Sintaxe Abstrata

Traduzindo Consultas SPARQL

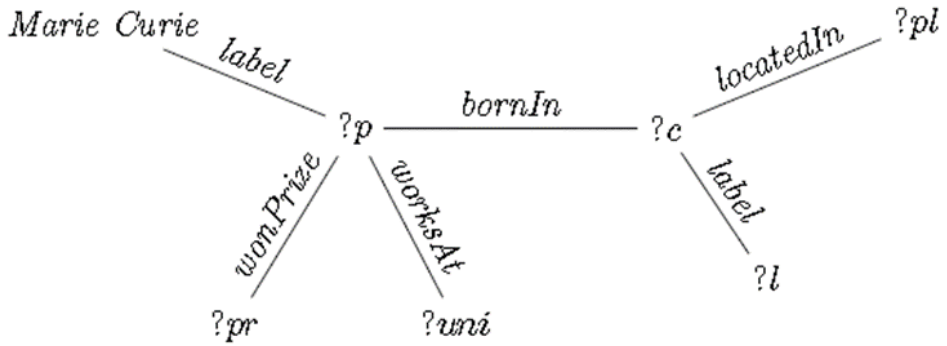
Após a construção do Grafo de Consulta

- No momento da execução, essa consulta é transformada em um conjunto de triplas.
- Cada uma dessas triplas consistem de recursos e literais que foram mapeados de acordo com os IDs no dicionário de dados.

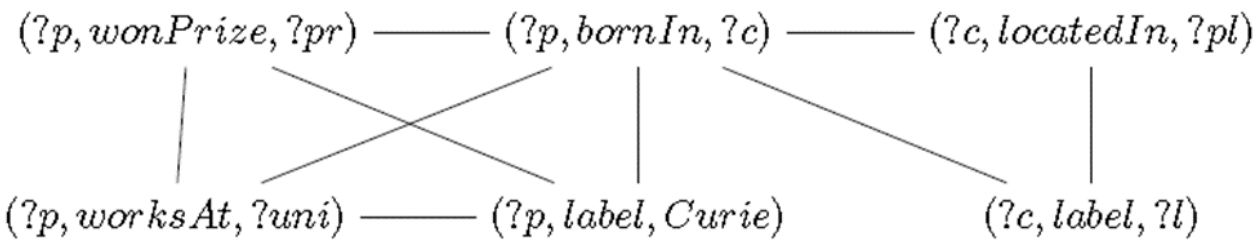
Consulta SPARQL e respectivos grafos

```
select *
where {
   ?p yago:hasName "Marie Curie".
   ?p yago:hasWonPrize ?pr.
   ?p yago:worksAt ?uni.
   ?p yago:wasBornIn ?c.
   ?c yago:isLocatedIn ?pl.
   ?c rdfs:label ?l }
```

(a) SPARQL query



(b) RDF query graph



(c) Join Query Graph

Traduzindo Consultas SPARQL

Árvore de Junção

- De acordo com o grafo criado, o mecanismo de consulta (*engine*) constrói sua representação algébrica junto com operadores de junções, conhecida como **árvore de junção**.
- Pode ser representada por uma árvore binária.
- Esta árvore junto com os índices e junções serve de base para o **plano de execução da consulta**.

Traduzindo Consultas SPARQL

Exemplos

- Dataset: YAGO2
- System: RDF-3X
- Algorithm: DP (Dynamic Programming)
- Não descreveram a configuração de hardware.

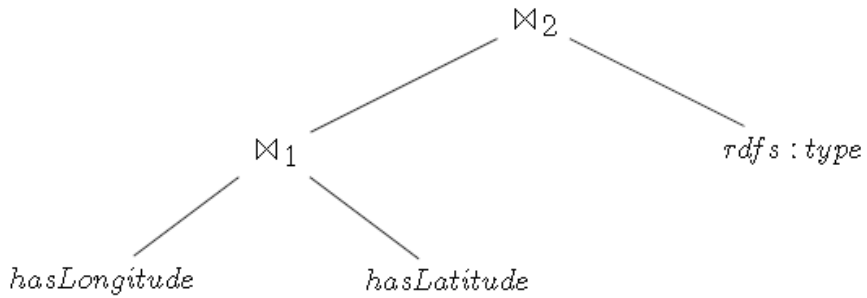
```
1  select ?s ?t where {
2          ?s yago:hasLongitude ?long.
3          ?s yago:hasLatitude ?lat.
4          ?s rdfs:type ?t.
5  }
```

Traduzindo Consultas SPARQL

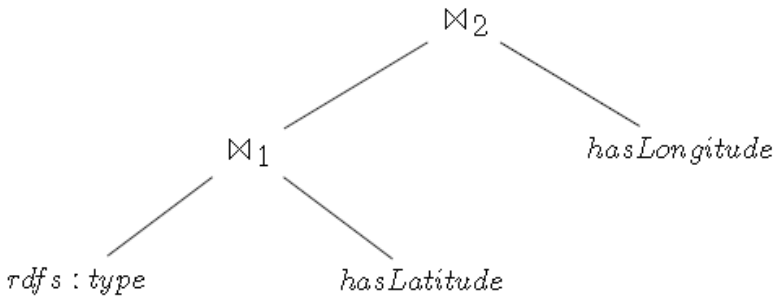
Exemplos

A execução da consulta levando em consideração:

- a cardinalidade resultou na árvore de junção (a)
- plano otimizado resultou na árvore (b).



(a)

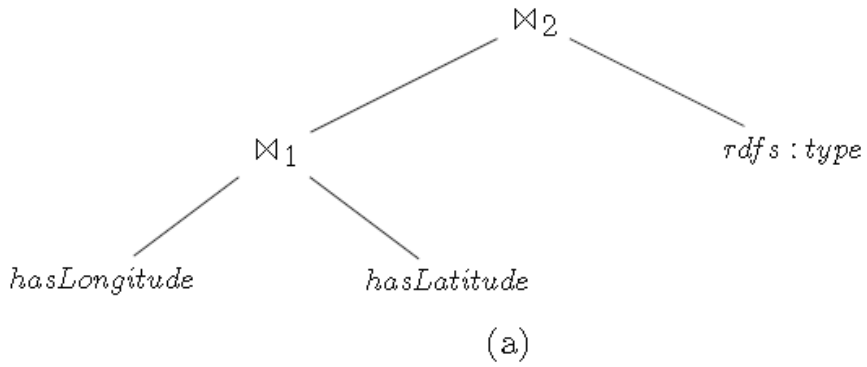


(b)

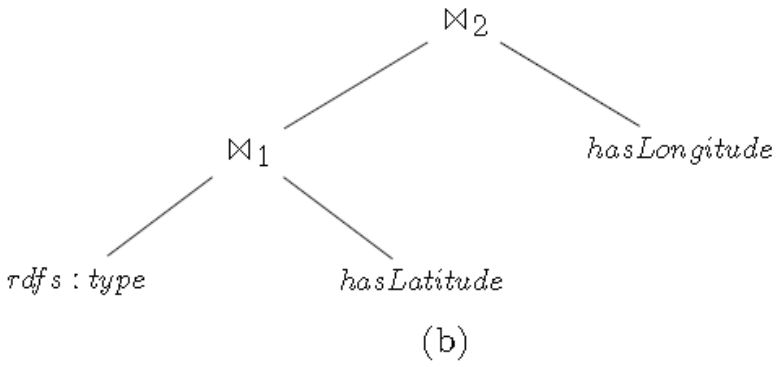
Traduzindo Consultas SPARQL

Exemplos

Os tempos de execução resultantes foram de 2100 ms para o plano que levou em consideração a cardinalidade (a) e 700 ms para o plano otimizado (b).



(a)

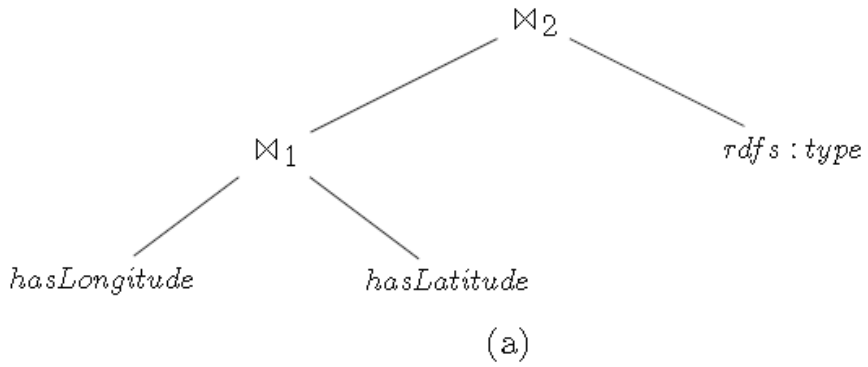


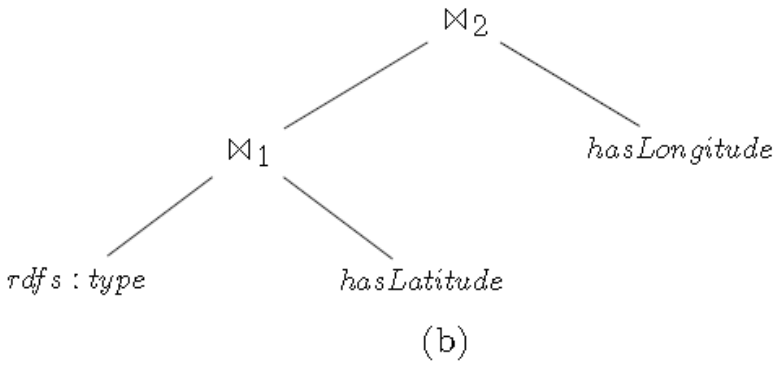
(b)

Traduzindo Consultas SPARQL

Exemplos

Sugestão de implementação do plano de execução: dar preferência para abordagens *bottom-up*, em vez de abordagens *top-down*.





Traduzindo Consultas SPARQL

Exemplos

- Artigo: Simpósio Brasileiro de Bancos de Dados - SBBD 2012
- Autores: Macedo Sousa Maia , João Carlos Pinheiro , Regis Pires Magalhães , José Maria da Silva Monteiro Filho , Vânia Maria Ponte Vidal
- Algorithm 1: PHJ (Pipeline Hash-join)
- Algorithm 2: SBJ (Set-bind-join)
- Dataset: DrugBank e Dailymed

http://www.lbd.dcc.ufmg.br/colecoes/sbbd/2012/0016.pdf

Traduzindo Consultas SPARQL

Junções e Caminhos

- Levando em consideração as junções e os caminhos que devem ser percorridos, mesmo depois que o otimizador encontre o plano ideal para uma consulta, uma execução ainda pode enfrentar dificuldades e demorar.
- Para executar um simples *select*, podem ocorrer grandes varreduras e diversos caminhos percorridos nas sub-árvores para retornar os dados desejados.

Traduzindo Consultas SPARQL

Restrição *Filter* na cláusula *Where*

Query:

```
PREFIX  dc:  <http://purl.org/dc/elements/1.1/>
PREFIX  ns:  <http://example.org/ns#>
SELECT  ?title ?price
WHERE   { ?x ns:price ?price .
          FILTER (?price < 30.5)
          ?x dc:title ?title . }
```

Query Result:

title	price
"The Semantic Web"	23

https://www.w3.org/TR/rdf-sparql-query

Conclusão

- As Triple Stores possuem um melhor desempenho quando utilizados com consultas simples ou consultas com poucas junções.
- Índices separados para o Sujeito, Predicado e Objeto são mais indicados do que índices agrupados.
- É importante termos atenção na construção das estruturas em SPARQL e no uso da indexação mais apropriada para o objetivo da pesquisa.

Obrigado!

Referências:

Hexastore:
http://crubier.net/Hexastore

RDF-3X:
http://dl.acm.org/citation.cfm?id=1731354

Virtuoso:
https://virtuoso.openlinksw.com

G-Store:
http://g-store.sourceforge.net

Jena:
https://jena.apache.org

http://www.lbd.dcc.ufmg.br/colecoes/sbbd/2012/0016.pdf

Perguntas

1. É correto afirmar que para consultas complexas em SPARQL devemos usar índices não clusterizados? Explique a sua resposta e porque devemos ou não usar esse tipo de índice.

2. Defina seletividade e cardinalidade e porque são importantes esses conceitos no contexto de banco de dados RDF.

ramalhoneto@gmail.com