Seleção e Otimização de Fontes

1.

Introdução

- Muitos dados disponíveis
- Não há garantia de relevância
- Muitos acessos (custoso)

O Autor propõe uma ideia para otimizar o processamento: A indexação e seleção de fontes relevantes para a consulta.

Seleção de Fontes

- Modelo de Consulta Genérico
- Processar consultas diretamente nos dados
- Determinar fontes com conteúdo relevante para a consulta
- Etapa pré-processamento

Resumo do **PROCESSO**

Estrutura Compacta de Indexação

Sumário de Dados

Selecionar Fontes

Fetch dos Dados

Processar Consulta

DIFERENTES ABORDAGENS

O autor apresenta diferentes abordagens para avaliar uma query e tratar a seleção de fontes.

- **Direct Lookup (DL)**
- **Schema Level Indexing (SLI)**
- **Inverted URI Indexing (II)**
- **Multidimensional Histograms (MDH)**
- **QTree (QT)**

DIFERENTES ABORDAGENS

Direct Lookup (DL)

Somente as URIs informadas na query ou as URIs obtidas em resultados parciais serão processadas. Não requer estrutura de indexação.

Schema-Level Indexing (SLI)

Mantém um índice do schema, para consultar as propriedades de cada fonte.

Inverted URI Indexing (II)

Cobre as ocorrências de uma URI (seed) e obtêm todas as que apresentam uma ligação (indexação invertida, usado por ferramentas de busca).

DIFERENTES ABORDAGENS

Multidimensional Histograms (MDH)

Combina elementos de instância e esquema, para sumarizar o conteúdo das fontes por meio de histogramas. Representa uma aproximação de todos os dados (reduz consumo de espaço).

QTree (QT)

Também combina elementos de instância e esquema. Ao contrário do MDH, QTree não possui regiões de tamanho fixo. QTree é uma árvore de dados no qual as regiões de tamanho variável cobrem os conteúdos das fontes.

Sumário de Dados

Sumário de Dados

- Alternativa para “apresentar” os dados das fontes
- Como resumir (sumarizar) dados numéricos é mais eficiente que strings, precisamos transformar as triplas RDF: utilizamos funções Hash
- Para cada hash, anexamos a fonte de origem, para que possamos identificar depois (seleção)

Multidimensional Histograms (MDH)

- Espaço numérico é particionado em regiões denominadas **buckets**
- Cada bucket, contém dados estatísticos sobre os itens de cada região
- Essas regiões são n-dimensionais, sendo n o número dos tipos de dado (no nosso caso, 3, ou seja, s, p e o)

O autor usa a expressão “data itens” para a “triplas” em alguns momentos.

Multidimensional Histograms (MDH)

- Para uma tripla, aplicamos o mesmo hash (usado para construir o histograma) e identificamos o bucket responsável.
- Adicionamos uma as triplas, uma de cada vez, e mantemos apenas um número de triplas por região e as fontes que contribuem

QTREE

- Combinação dos Histogramas multidimensionais e Árvores R
- Estrutura Hierárquica em que cada região é representada por um nodo e cada tripla uma folha
- A diferença é que QTree só cobrem as regiões que contém dados (vide imagem)

QTREE

Construção é dividida em 2 etapas

- Insert: buscar o bucket, atualizar dados estatísticos e criar registro de fonte. Conferir as regras de número máximo de nós filhos e de buckets é respeitada (do contrário, merge e cria um novo)
- Lookup: É necessário uma busca em árvore, até identificar o bucket com a tripla correspondente

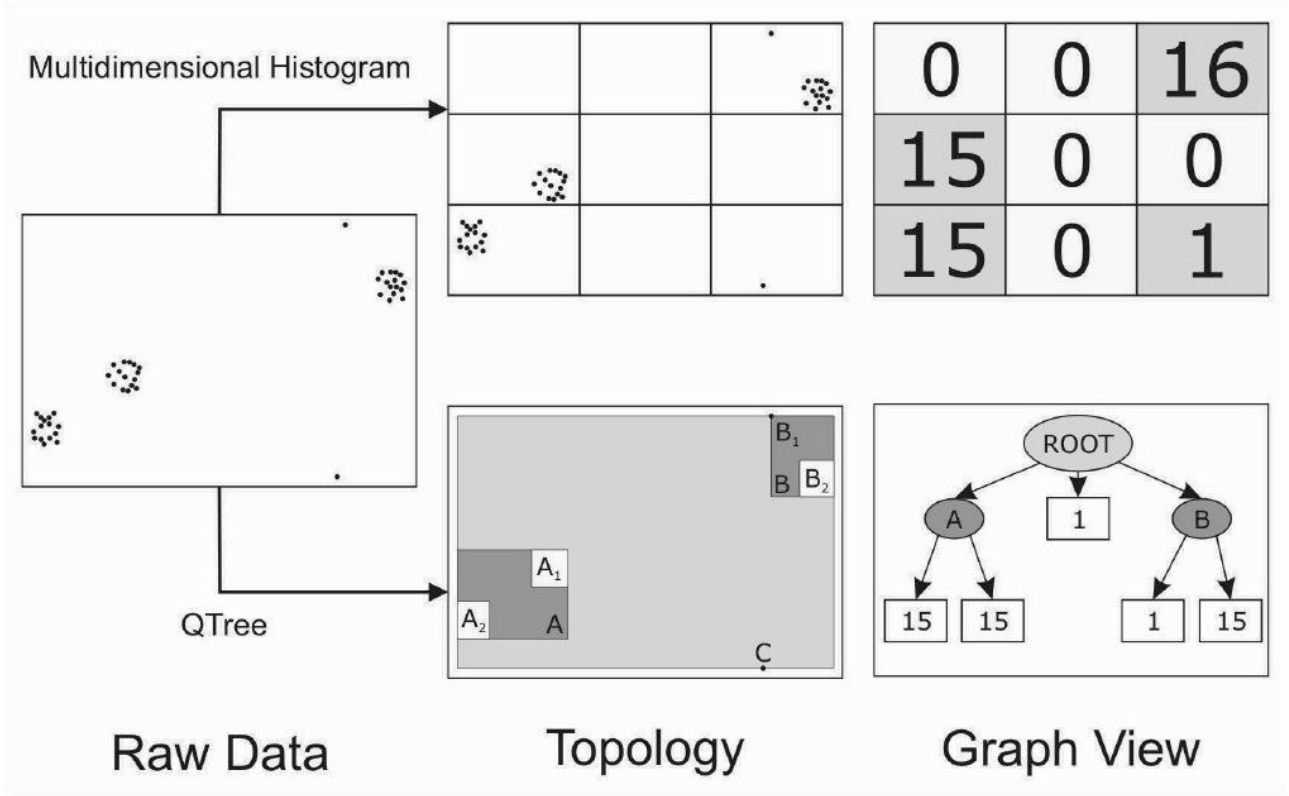

FIGURE 12.1: Example of a data summary. The left column shows the coordinates corresponding to hash values for the data items to insert. The middle column shows the bucket regions, and the right column shows the buckets with the assigned statistical data and, in the case of the QTree, the hierarchy between inner nodes and buckets.

Construção e Manutenção

Dividido em 2 fases:

- Inicial

- Expansão

Construção e Manutenção

Fase inicial é baseada na versão do sumário de dados (seed). Duas formas:

- Pre-Fetch: buscar conjunto através de Web Crawler
- SPARQL Queries: usa as queries e coleta as fontes, para indexar, através da resposta. Podemos utilizar o LTBQE para recuperar as URIs durante a consulta

Construção e Manutenção

Fase de Expansão: integrar outras fontes ainda não adicionadas. Duas formas:

- Push: envolve um agente para ativar a expansão (notificação de serviço sobre uma nova URI)
- Pull: implementa um fetch preguiçoso durante a execução (desreferenciar todas as novas URIs durante a consulta)

Importância do HASH

Importância do HASH

É possível se utilizar de uma variedade de funções hash para mapear os dados. Contudo, um método bastante comum é normalizar os valores dos hash, criando um range numérico da função escolhida em um range menor. Isso aprimora o agrupamento (clustering) e distribuição dos dados

Importância do HASH

Para definir o range, é preciso considerar 2 casos especiais:

- range grande: dados esparsos. Logo, a maioria dos valores numéricos não será usado, o que torna-se um problema para os MDH (não trata dados esparsos muito bem)
- range pequeno: colisão de valores. Falsos positivos durante a decisão de seleção de fontes.

Importância do HASH

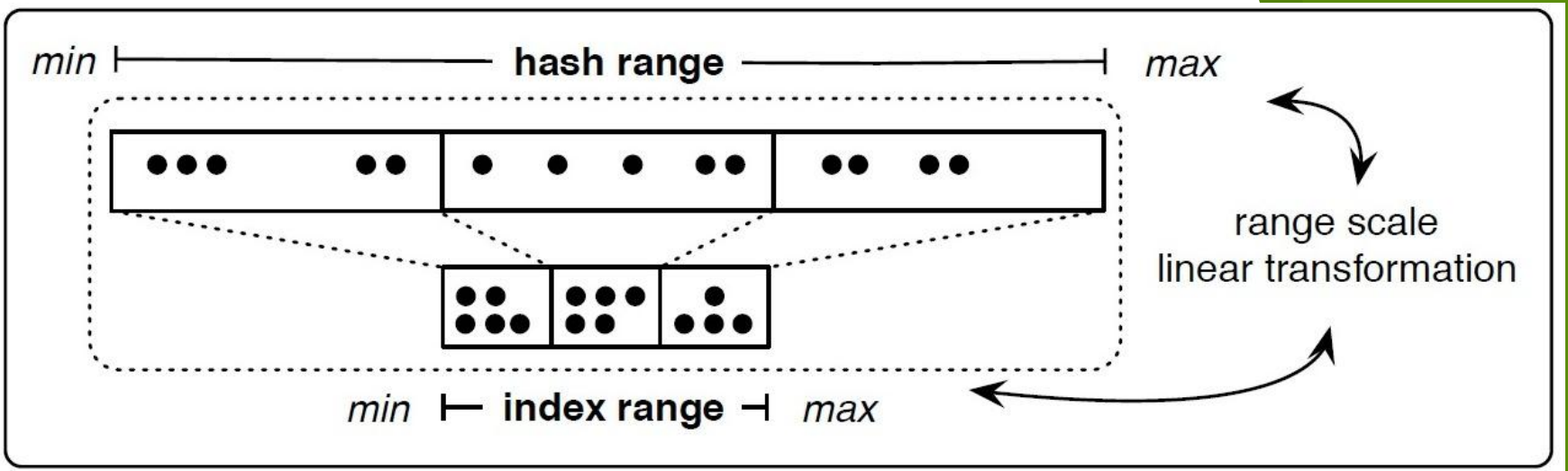

FIGURE 12.2: Example of range scaling.

Funções HASH

- String Hashing (STR): gera um hash diferente para cada string e utiliza-se mais do range de valores de hash

- Prefix Hashing (PRE): árvore de prefixos para mapear os valores da string para o prefixo que mais próximo e providencia um bom agrupamento de strings (similares). Requer mais espaço para manter a árvore de prefixos

Funções HASH

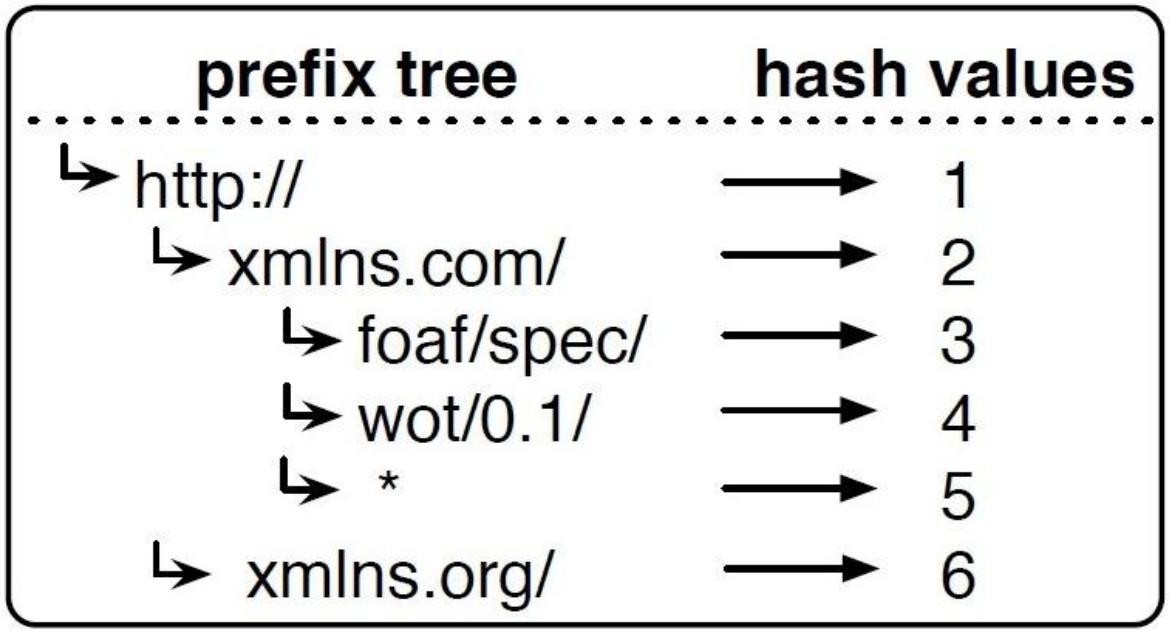

FIGURE 12.3: Prefix Hashing.

Funções HASH

- Mixed Hashing (MIX): Combina as estratégias das duas abordagens anteriores: para o sujeito e objeto, aplica-se prefix hashing; para o predicado, aplica-se o string hashing. O motivo dessa escolha é que dentro da WEB existem poucos predicados distintos, comparado com o número de sujeitos e objetos. Se aplicássemos o prefix hash no predicado, iríamos perder muita informação, pois vários seriam mapeados para o mesmo valor.

Escolha das Fontes

Escolha das Fontes

- Determinamos fontes relevantes para agregarmos na busca, definimos um padrão de tripla e o convertemos para um conjunto de coordenadas, aplicando as operações hash escolhidas. Com esse valor, podemos determinar qual bucket/região possui candidatos relevantes
- No MDH, verifica todos buckets
- QTree, busca a partir da raiz

Escolha das Fontes (Otimizar)

- O autor afirma que podemos otimizar ainda mais o processo
- Tudo depende dos seguintes fatores
  - A ordem dos joins
  - O processo em si de join
  - Determinar quais regiões fazer o join

Determinar Regiões

- Uma questão crucial é como determinar fontes irrelevantes e descartá-las. Contudo, se um bucket apresenta sobreposição, não podemos omitir nenhuma das fontes, pois, não sabemos qual fonte está contribuindo para aquele bucket.
- Uma opção seria considerar todas as fontes como relevantes. Fontes podem ser descartadas se todo o restante do bucket for descartado.

Join das Regiões

- **União por Loop Aninhado**: algoritmo irá determinar as sobreposições de cada bucket e adicioná-los ao conjunto de saída.

- **União por Indexação**: Podemos melhorar a performance utilizando índices especiais de junção. Uma opção é usar o “índice invertido de bucket”, que armazenado mapeamentos das regiões de cada bucket.

Join das Regiões

Algorithm 12.1 Principle of nested-loop join.

Input: left input $\mathcal{L}$, right input $\mathcal{R}$
Output: join space containing overlapping buckets

```
for all L ∈ 𝓛 do
    for all R ∈ ℛ do
        𝒥.add(determineOverlap(L, R))
    end for
end for
return 𝒥
```

Join das Regiões

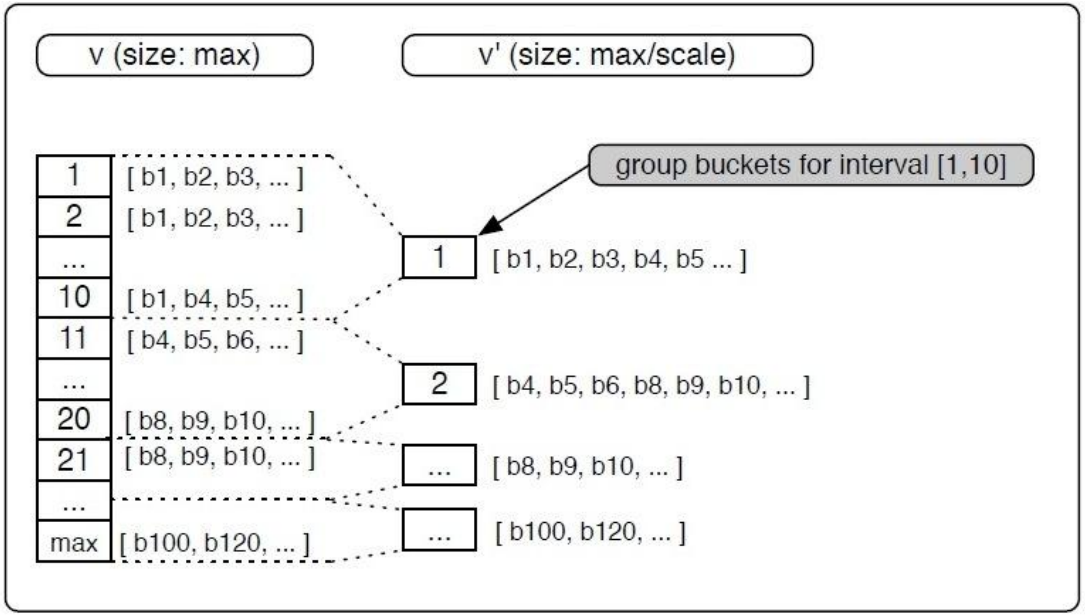

FIGURE 12.7: Illustration of an inverted bucket index.

Estimativa e Cardinalidade

- A Seleção é uma “aproximação” de um conjunto relevante. Existe a possibilidade dele ser grande, o que contraria a proposta, ou, apresentar falso positivo. Para evitar este tipo de problema, é necessário rankear as fontes, para determinar o conjunto mais relevante.

- Uma opção, seria utilizar da cardinalidade fornecida do resumo/sumário de dados. Nele, é possível estimar a contribuição que cada uma das fontes fornece para a seleção.

Conclusão

- solução para otimizar o processamento de consultas SPARQL sobre estruturas RDF publicadas com dados integrados na Web. O problema principal se tratava da quantidade de URIs que deveriam ser acessadas, e esse processo tornava-se extremamente custoso. Portanto, a solução foi aplicar a seleção de fontes relevantes dentro do conjunto, para reduzir esse número de acessos.

Conclusão

- A seleção é uma “aproximação” e não é 100% eficaz: (falsos positivos, fonte irrelevante ou fonte relevante “ignorada”).

- O autor sugere que ainda existem outras formas de otimizar o processo, como por exemplo, evitar as inserções um-a-um na QTree (processo custoso) através, se possível, de técnicas pré-clusterização.

Exercícios

1.

O processo de otimização pode ser dividido em algumas etapas. Apresente essas etapas e comente um **pouco** sobre elas.

2.

Comente sobre os diferentes tipos de abordagens citados para **seleção de fontes**.

jonatankorello@gmail.com

Dúvidas, sugestões e críticas são sempre bem-vindas.