Pontifícia Universidade Católica
do Rio de Janeiro

Luciana de Sá Silva Perciliano

Inferência de Tuning através da OnDBTuning

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre pelo Programa de Pós-graduação em Informática da PUC-Rio.

Orientador: Prof. Sérgio Lifschitz

Rio de Janeiro
Dezembro de 2021

PUC-Rio - Certificação Digital Nº 1912717/CA

Pontifícia Universidade Católica
do Rio de Janeiro

Luciana de Sá Silva Perciliano

Inferência de Tuning através da OnDBTuning

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre pelo Programa de Pós–graduação em Informática da PUC-Rio. Aprovada pela Comissão Examinadora abaixo:

Prof. Sérgio Lifschitz
Orientador
Departamento de Informática – PUC-Rio

Prof. Edward Hermann Haeusler
Departamento de Informática – PUC-Rio

Profa. Ana Carolina Brito de Almeida
Instituto de Matemática e Estatística-UERJ

Prof. José Maria da Silva Monteiro Filho
Departamento de Computação-UFC

Rio de Janeiro, 21 de Dezembro de 2021

PUC-Rio - Certificação Digital Nº 1912717/CA

Todos os direitos reservados. A reprodução, total ou parcial do trabalho é proibida sem a autorização da universidade, da autora e do orientador.

Luciana de Sá Silva Perciliano

Possui especialização em Análise e Projeto de Sistemas (2011) pela PUC-Rio. Graduação em Ciências Atuariais (2007) pela Sociedade Unificada de Ensino Superior e Cultura (SUESC). Também possui graduação em Informática pela Universidade Estácio de Sá (2002). Atualmente é Analista de TI na Empresa de Tecnologia e Informações da Previdência Social (DATAPREV). Tem experiência na área de Ciência da Computação, com ênfase em Sistemas de Computação.

PUC-Rio - Certificação Digital Nº 1912717/CA

Ficha Catalográfica

de Sá Silva Perciliano, Luciana

Inferência de Tuning através da OnDBTuning / Luciana de Sá Silva Perciliano; orientador: Sérgio Lifschitz. – 2021.

60 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2021.

Inclui bibliografia

1. Banco de Dados – Teses. 2. Informática – Teses. 3. Ontologia. 4. Sintonia Fina. 5. SPIN. 6. SPARQL. I. Lifschitz, Sérgio. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Agradecimentos

Em primeiro lugar, agradeço a Deus e depois aos meus pais, José Donizetti Silva (*in memorian*) e Rosangela de Sá Silva, pois sem eles todo o conhecimento e cultura adquiridos não teriam sido alcançados. Também agradeço ao meu esposo, Luiz Perciliano, pelo apoio dado durante o período do mestrado.

Ao meu orientador, Professor Sérgio Lifschitz, pelo estímulo e parceria na realização deste trabalho. A todos os professores e funcionários da PUC-Rio, pelos ensinamentos e pela ajuda.

Aos meus colegas da PUC-Rio, em especial a Veronica Santos, por toda parceria, pelas importantes contribuições e palavras de apoio.

Aos professores Edward Haeusler Hermann e Ana Carolina Santos pelas contribuições que foram fundamentais para o andamento dessa pesquisa.

Aos professores que participaram da Comissão examinadora.

À professora Fernanda Baião e aos professores e colegas da UNIRIO, os quais fizeram parte do início da minha jornada acadêmica.

A todos os meus colegas de trabalho da Dataprev pelo estímulo, em especial a Nicomar Oliveira, Eduardo Fritzen e Patricia Zudio.

A todos os amigos e familiares que me estimularam ou ajudaram.

Ao CNPq e à PUC-Rio, pelos auxílios concedidos, sem os quais este trabalho não poderia ter sido realizado.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

PUC-Rio - Certificação Digital Nº 1912717/CA

Resumo

de Sá Silva Perciliano, Luciana; Lifschitz, Sérgio. **Inferência de Tuning através da OnDBTuning**. Rio de Janeiro, 2021. 60p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

OnDBTuning é uma ontologia de *tuning* (semi-automático) de banco de dados relacional. Ontologias são artefatos que representam o conhecimento de um domínio específico e podem ser usadas para se inferir conhecimentos. No entanto, em geral, a maioria das aplicações envolve apenas uma descrição formal e estática de conceitos. Além disso, como *tuning* de banco de dados envolve muitas regras baseadas na experiência e em algoritmos de caixa preta, torna-se um desafio descrever esse processo de inferência. Esse trabalho de pesquisa apresenta primeiramente a solução OnDBTuning que é uma ontologia no domínio de *tuning*. Em seguida, ele propõe uma implementação de regras em SPARQL *Inferencing Notation* (SPIN) na OnDBTuning. Por fim, mostra uma avaliação prática da solução para recomendação de índices e visões materializadas.

PUC-Rio - Certificação Digital Nº 1912717/CA

Palavras-chave

Ontologia; Sintonia Fina; SPIN; SPARQL.

Abstract

de Sá Silva Perciliano, Luciana; Lifschitz, Sérgio (Advisor). **Tuning inference through OnDBTuning**. Rio de Janeiro, 2021. 60p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

OnDBTuning is a relational database (semi-automatic) tuning ontology. Ontologies are artifacts that represent specific domain knowledge and can be used to infer knowledge. However, in general, most applications include only a formal and static description of concepts. Moreover, as database tuning involves many rules-of-thumb and black-box algorithms, it becomes challenging to describe these inference procedures. This research work first presents the OnDBTuning ontology solution focusing on the inference of tuning actions. Next, it proposes an implementation of the OnDBtuning rules using SPARQL Inferencing Notation (SPIN). Finally, it shows a practical evaluation of our solution concerning index and materialized views recommendations.

PUC-Rio - Certificação Digital Nº 1912717/CA

Keywords

Ontology; Tuning; SPIN; SPARQL.

Sumário

1 Introdução **12**
1.1 Motivação 12
1.2 Objetivos e Escopo da Dissertação 13
1.3 Estrutura da Dissertação 14

2 Conceitos Básicos e Trabalhos Relacionados **15**
2.1 Sintonia Fina e Otimização 15
2.2 Ontologia 16
2.3 Inferência 18
2.4 Contexto e Trabalhos Relacionados 19
2.5 Resumo do Capítulo 22

3 Inferência na OnDBTuning **23**
3.1 OnDBTuning 23
3.2 Decomposição SQL 25
3.3 Modelagem das Regras de Tuning 26
3.3.1 Regra do Índice Simples e Composto 26
3.3.2 Regra do Índice Parcial 29
3.3.3 Regra da Visão Materializada 30
3.4 Indicadores 32
3.4.1 Bônus 32
3.4.2 Utilidade 33
3.5 Resumo do Capítulo 34

4 Estudo de Caso **35**
4.1 Ambiente e Carga de Trabalho 35
4.2 Regras de Inferência de Bônus e Utilidade 36
4.3 Regras de Inferência de Índices Simples e Composto 37
4.4 Inferências de Índices Simples e Composto 39
4.5 Comparativo do Bônus e Utilidade das Cargas de Trabalho 42
4.6 Inferência de Índices Parciais 44
4.7 Inferência de Visões Materializadas 46
4.8 Escopo do Estudo de Caso 47
4.9 Conclusões 47

5 Conclusão **48**
5.1 Contribuições da Pesquisa 48
5.2 Trabalhos Futuros 49

Referências bibliográficas **50**

A Consultas TPC-H **54**
A.1 Consulta TPC-H Q1 54
A.2 Consulta TPC-H Q3 - 1 55
A.3 Consulta TPC-H Q3 - 2 55

PUC-Rio - Certificação Digital Nº 1912717/CA

A.4 Consulta TPC-H Q4 56
A.5 Consulta TPC-H Q5 56
A.6 Consulta TPC-H Q8 57

B Regra da Visão Materializada 58

PUC-Rio - Certificação Digital Nº 1912717/CA

Lista de figuras

Figura 2.1 Exemplo da Estrutura de Tripla RDF 17
Figura 2.2 Regra Representada em SPARQL 19
Figura 2.3 Regra Representada em RDF 19

Figura 3.1 Fragmento da OnDBTuning [3] 24
Figura 3.2 Decomposição do WhereClause 25
Figura 3.3 Diagrama de Estados de Instâncias da Ontologia 26
Figura 3.4 Diagrama de Atividades - Índice Simples 27
Figura 3.5 Diagrama de Atividades - Índice Composto 28
Figura 3.6 29
Figura 3.7 Diagrama de Atividades - Índice Parcial Simples 30
Figura 3.8 Diagrama de Atividades - Visão Materializada 31
Figura 3.9 Descrição da Visão Materializada 32
Figura 3.10 Recorte da Consulta Q5 33
Figura 3.11 Recorte da Consulta Q8 33

Figura 4.1 Regra do Bônus 36
Figura 4.2 Regra de Utilidade - Índice Simples 37
Figura 4.3 Regra de Utilidade - Índice Composto 37
Figura 4.4 *RuleHypSimpleIndex* 38
Figura 4.5 *RuleHypCompositeIndex* 39
Figura 4.6 Inferência de RuleHypSimpleIndex 40
Figura 4.7 Inferência de RuleHypCompositeIndex 41
Figura 4.8 Índice LINEITEM_L_PARTKEY_FKEY 42
Figura 4.9 Bônus dos Índices Hipotéticos 43
Figura 4.10 Utilidade dos Índices Reais 44
Figura 4.11 Regra do Índice Parcial 45
Figura 4.12 Índice Parcial Inferido 45
Figura 4.13 Visão Materializada Hipotética 46

Figura A.1 Consulta Original e Instanciada - TPC-H Q1 54
Figura A.2 Consulta Original e Instanciada - TPC-H Q3 55
Figura A.3 Consulta Original e Instanciada - TPC-H Q3 55
Figura A.4 Consulta Original e Instanciada / Reescrita - TPC-H Q4 56
Figura A.5 Consulta Original e Instanciada - TPC-H Q5 56
Figura A.6 Consulta Original e Instanciada / Reescrita - TPC-H Q8 57

Figura B.1 Regra *RuleHypViewAdapted* (parte 1) 58
Figura B.2 Regra *RuleHypViewAdapted* (parte 2) 59
Figura B.3 Regra *RuleHypViewAdapted* (parte 3) 60

PUC-Rio - Certificação Digital Nº 1912717/CA

Lista de tabelas

Tabela 3.1 Sugestões de Índices Simples e Índices Compostos 29
Tabela 3.2 Expressão Condicional de Índice Parcial 30

PUC-Rio - Certificação Digital Nº 1912717/CA

Lista de Abreviaturas

APM – Application Performance Monitoring

AST – Abstract Syntax Trees - Árvore sintática abstrata

DBA – Database Administrator - Administrador de Banco de Dados

DML – Data Manipulation Language - Linguagem de Manipulação de Dados

IA – Inteligência Artificial

OLAP – Online Analytical Processing

OnDBTuning – Database Tuning Ontology

OWL – Web Ontology Language

ML – Machine Learning - Aprendizado de Máquina

RDF – Resource Description Framework

RDF-S – RDF-Schema

RL – Deep Reinforcement Learning

RN – Redes Neurais

SGBD – Sistema de Gerenciamento de Banco de Dados

SGBDR – Sistema de Gerenciamento de Banco de Dados Relacional

SHACL – Shapes Constraint Language

SPARQL – SPARQL Protocol and RDF Query Language

SPIN – SPARQL Inferencing Notation

SQL – Structured Query Language

SWRL – Semantic Web Rule Language

URI – Universal Resource Identifier

W3C – World Wide Web Consortium

PUC-Rio - Certificação Digital Nº 1912717/CA

1
Introdução

As ações de sintonia fina (*tuning*) em um banco de dados são importantes na melhora do desempenho das aplicações, pois elas visam a diminuir o tempo de resposta de requisições e/ou aumentar a vazão (*throughput*) das transações [1]. Essas ações compreendem os seguintes ajustes: de configurações, de parâmetros dos Sistemas de Gerenciamento de Banco de Dados (SGBD) e do projeto físico dos Sistemas de Bancos de Dados envolvidos. Uma das formas de ajuste no projeto físico é realizar a criação de estruturas de acesso, tais como índices e visões materializadas.

Esse tema tem grande relevância na área de banco de dados relacionais, sendo pesquisado pelo grupo BioBD da PUC-Rio por mais de uma década, conforme descrito na seção 2.4.

PUC-Rio - Certificação Digital Nº 1912717/CA

1.1
Motivação

As decisões sobre *tuning* de banco de dados são na maioria das vezes baseadas na experiência de um Administrador de Banco de Dados (DBA) ou em ferramentas que proporcionam indicadores e métricas que apoiam a tomada de decisão desse especialista. Porém, o raciocínio da sintonia fina recomendada na maioria dos cenários não fica com sua explicabilidade expressa para entendimento de outros analistas. Da mesma maneira, a justificativa de uma sintonia fina também não tem argumentos e exemplos que corroborariam para sua implementação. A solução é a implementação de uma ontologia, que torna possível modelar o raciocínio, sequenciamento e fundamentação da sintonia fina.

As ontologias definem conceitos, relacionamentos, instâncias, restrições e regras de um determinado domínio [2], além de ser possível através delas inferir conhecimentos.

A ontologia OnDBTuning [3] utilizada neste trabalho de pesquisa possui conceitos e propriedades no domínio de sintonia fina de banco de dados relacionais.

O Outer-tuning é um *framework* que tem como propósito facilitar o trabalho do DBA ou de outras pessoas que venham interagir com banco de

dados relacionais, propondo ações de sintonia fina, tais como a criação de índices e visões materializadas, através da exibição dos custos e benefícios que essas ações proporcionam, assim como os comandos SQL que são beneficiados em conjunto com essa ou essas ações propostas [4].

Por conta do tamanho e da complexidade da OnDBTuning, alguns desafios importantes, científicos e tecnológicos, foram enfrentados pelo *framework* Outer-tuning e a implementação dessa ferramenta não contemplou a inferência automática das ações de sintonia fina por meio dessa ontologia.

Sendo assim, esse trabalho de pesquisa procurou viabilizar o mecanismo de inferência da OnDBtuning, permitindo um processo e funcionamento declarativo, ou seja, de forma a explicitar o raciocínio utilizado para as inferências e extensível, pois não depende de carga de trabalho específica e SGBD, podendo ser integrado posteriormente com uma ferramenta para entrada dos dados (*input*) e para análise do resultado (*output*) das sugestões de ações de inferências da ontologia.

1.2 Objetivos e Escopo da Dissertação

Esse trabalho de dissertação de mestrado pretende explorar a possibilidade de se inferir ações de sintonia fina através de uma ontologia específica, tendo como resultado uma sugestão equivalente ao que seria sugerido por um especialista em *tuning* de sistemas de bancos de dados relacionais. Formalizar o conhecimento desse especialista para inferir ações de *tuning*, torna-se um desafio. Para que o objetivo fosse alcançado foi realizado um estudo de caso que considera a ontologia OnDBTuning existente e as seguintes contribuições dessa dissertação:

- modelagem de regras;
- decomposição SQL;
- inferência de ações de *tuning*;
- indicadores.

A construção da modelagem das regras de *tuning*, foram baseadas nos conceitos presentes na ontologia. Depois de modeladas as regras, foi necessário identificar as partes que compõe o comando SQL que correspondem aos conceitos da ontologia. As regras de *tuning* modeladas e as decomposições SQL foram descritas na linguagem SPIN e estendidas na Ontologia com a inclusão dessas regras. Metadados e consultas de uma determinada carga de trabalho foram instanciadas, ou seja, adicionadas na ontologia. Sugestões de

PUC-Rio - Certificação Digital Nº 1912717/CA

ações de *tuning* foram inferidas, através da execução de uma máquina de regras específica. E também a criação de indicadores denominados bônus e utilidade, como novas propriedades na OnDBTuning.

Dentre os resultados desse estudo de caso, espera-se a inferência de sugestões de ações de *tuning*, tais como índices simples hipotéticos, índices compostos hipotéticos, índices parciais e visões materializadas hipotéticas, além de valores de bônus para os índices hipotéticos e utilidade para os índices reais.

Cabe salientar que o escopo dessa dissertação não objetiva a escolha da melhor estrutura de acesso para uma carga de trabalho (*workload*), mas de sugestões de ações de *tuning* e indicadores que auxiliem na tomada de decisão.

1.3 Estrutura da Dissertação

A estrutura desta dissertação está organizada da seguinte forma:

- O Capítulo 2 descreve os conceitos fundamentais, o contexto e os trabalhos relacionados desta pesquisa.
- O Capítulo 3 apresenta a ontologia OnDBTuning, o processo de decomposição SQL, a modelagem do processo de instanciação da OnDBTuning e das regras de *tuning* para: índice simples, índice composto, índice parcial e visão materializada; e os indicadores denominados bônus e utilidade.
- O Capítulo 4 descreve os resultados experimentais, apresentando o ambiente e a carga de trabalho utilizada, as inferências de índices simples e índice composto, o bônus e utilidade calculadas, as inferências das cargas de trabalho, a inferência de índices parciais e visões materializadas; e o escopo do estudo de caso.
- O Capítulo 5 apresenta a conclusão, com as contribuições da pesquisa e os trabalhos futuros.
- O apêndice A lista as consultas do TPC-H e no apêndice B apresenta a regra da visão materializada.

PUC-Rio - Certificação Digital Nº 1912717/CA

2
Conceitos Básicos e Trabalhos Relacionados

Este capítulo descreve conceitos básicos para melhor entendimento dessa dissertação de mestrado, o contexto e trabalhos relacionados.

2.1
Sintonia Fina e Otimização

A sintonia fina (*tuning*) de banco de dados tem por objetivo proporcionar menor tempo de consulta nas transações e/ou maior vazão (*throughput*) em um banco de dados [1]. Normalmente, isso ocorre através de um conjunto de ações propostas por um especialista, também conhecido como DBA, com base na sua experiência prévia ou por métricas obtidas por ferramentas de monitoramento de sistemas (*Application Performance Monitoring* -APM).

O *tuning* compreende o ajuste de configurações, parâmetros e aspectos do projeto físico que pode ser realizado de forma automática (também denominado auto-sintonia de banco de dados, sintonia fina automática de banco de dados e *self-tuning* de banco de dados) [3] ou de forma manual. Uma das formas de ajuste no projeto físico é realizar a criação de estruturas de acesso, tais como visões materializadas e índices.

Uma visão (*view*) "é uma relação derivada, definida em termos de relações de base". Uma visão é considerada materializada quando é armazenada fisicamente no banco de dados, ao invés de computada a partir das relações de base em resposta às consultas [5].

Os índices são estruturas de acesso que podem acelerar e oferecer alternativas na recuperação dos registros [6] e podem ser classificados como reais ou hipotéticos. Enquanto os índices reais existem fisicamente no banco de dados, os índices hipotéticos estão presentes somente nos metadados [7]. Esse mesmo entendimento pode ser estendido para a visão materializada. Os metadados, também chamados descritores, podem ser considerados uma base de dados isolada do sistema que contém todas as definições dos objetos sistêmicos, detalhando as informações sobre estes dados [8].

Um índice é parcial se indexar um subconjunto de tuplas de uma tabela determinada por uma expressão condicional. Esse índice parcial é criado a partir de um CREATE INDEX com restrição de cláusula WHERE [9]. Como

PUC-Rio - Certificação Digital Nº 1912717/CA

exemplo, no atributo C_MKTSEGMENT da tabela CUSTOMER, pode ser definido um índice parcial denominado CUSTOMER_C_MKTSEGMENT:

```
CREATE INDEX CUSTOMER_C_MKTSEGMENT
ON CUSTOMER (C_MKTSEGMENT)
WHERE C_MKTSEGMENT = 'BUILDING'
```

Os SGBDs do tipo relacional, normalmente possuem um otimizador encarregado em avaliar, dentro de um grande conjunto de alternativas possíveis, o caminho de menor custo na execução de uma consulta, podendo escolher uma ou várias das estruturas de acesso [3].

As estruturas de acesso selecionadas durante o processo de sintonia fina estão associadas a heurísticas que explicitam o conhecimento do DBA especialista em *tuning*. Essas heurísticas abrangem decisões sobre a criação (o que criar e quando criar), além da manutenção das estruturas de acesso. Como exemplo, podemos citar pesquisas que possuem regras para sugestão de criação, eliminação e recriação automática de índices [10]. Outro exemplo seriam heurísticas para escolha e seleção de índices candidatos a serem criados em ações de sintonia fina [7] e heurísticas para criação de visão materializada [11]. Essas heurísticas podem ser modeladas através de uma ontologia. Assim, conseguimos formalizar conhecimentos práticos dos DBAs para que seja viável a sintonia fina automática com apoio de programa de maneira declarativa e extensível.

PUC-Rio - Certificação Digital Nº 1912717/CA

2.2 Ontologia

Uma ontologia, sob o viés computacional, é reconhecida como um artefato capaz de representar formalmente uma conceituação compartilhada de um domínio [12]. Ela pode ser representada como uma tupla $O = \langle C, R, I, A \rangle$, onde:

- C é a representação de um conjunto de conceitos de um domínio;
- R é o conjunto de relacionamentos - ou associações - entre esses conceitos;
- I é o conjunto de instâncias de classes;
- A é um conjunto de axiomas do domínio usados para modelar restrições e regras do domínio [2].

As ontologias podem modelar diversos tipos de domínios. Como exemplo, na área de banco de dados, a ontologia Relational Database System Ontology

(RDBS-O) [13] modela um sistema de banco de dados. E a OnDBTuning modela a sintonia fina de banco de dados que será melhor detalhada na seção 3.1.

O *Resource Description Framework* (RDF)[1] é um modelo de dados especificado pelo *World Wide Web Consortium* (W3C) para descrever metadados. Esse modelo utiliza uma estrutura de triplas do tipo <sujeito> <predicado> <objeto>, sendo o <sujeito> e <objeto> os nós e o <predicado> a relação ou propriedade entre eles, conforme exemplo da Figura 2.1. O conjunto destas estruturas em triplas é denominado de grafo RDF [14].

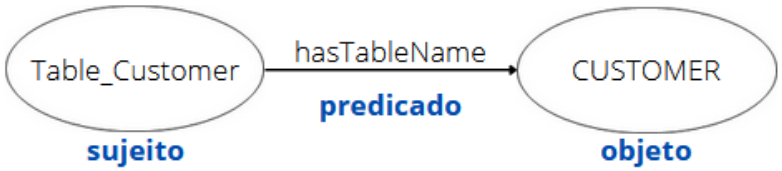

Figura 2.1: Exemplo da Estrutura de Tripla RDF

Uma Árvore Sintática Abstrata (AST) é uma representação estruturada (árvore) de um código de alguma linguagem [15]. A estrutura AST de consultas SQL podem ser modeladas através de grafos RDF. Logo, uma carga de trabalho de um banco de dados pode ser representada por um grafo RDF, possibilitando que a estrutura sintática seja anotada com metadados adicionais, a identificação das consultas através da *Universal Resource Identifier* (URI) e o mapeamento por meio de nós dos tipos de recursos [16].

O RDF-S (ou RDF-Schema) é uma especificação que permite descrever com poucos elementos as ontologias, através da definição de classes, propriedades e seus relacionamentos [14].

A *Web Ontology Language* (OWL)[2] é uma linguagem recomendada pela W3C, amplamente utilizada para representar formalmente ontologias, sendo baseada nas especificações RDF/RDF-S [14].

O RDF-S e o OWL são úteis para modelar classes, propriedades e relacionamentos entre esses conceitos, ou seja, são usadas para modelar a estrutura estática do conhecimento, incluindo as definições axiomáticas das estruturas de dados. Outras linguagens são necessárias para descrever o comportamento e inferência de objetos.

O *Semantic Web Rule Language* (SWRL)[3] cobre este escopo através da combinação de OWL com regras lógicas *horn*. Apesar dele ser uma linguagem

[1] `https://www.w3.org/RDF`
[2] `https://www.w3.org/OWL/`
[3] `https://www.w3.org/Submission/SWRL/`

PUC-Rio - Certificação Digital Nº 1912717/CA

bem difundida e suportada por editores de ontologia, não é uma recomendação W3C [17].

O *SPARQL Protocol and RDF Query Language* (SPARQL) é uma linguagem declarativa designada para manipular triplas RDF. O SPARQL suporta quatro tipos de consultas: SELECT (selecionar variáveis e expressões específicas), CONSTRUCT (construir triplas/grafos RDF), ASK (consulta e retorna se algo é verdadeiro ou falso), e DESCRIBE (descreve os recursos correspondentes as variáveis informadas, recuperando o nó/arestas adjacentes). Particularmente, o formulário de consulta CONSTRUCT pode ser utilizado para mapear sujeitos, predicados e objetos de triplas armazenados para classes, relacionamentos e atributos da ontologia de domínio ou inferir novas triplas [18].

De acordo com [17], SPIN (SPARQL Inferencing Notation)[4] é um padrão da indústria para representar regras e restrições na Web Semântica, construída com base na linguagem SPARQL, geralmente usada para consultas e processamento *Linked Open Data*, que também é conhecido como SPARQL *Rules*. A especificação SPIN define um vocabulário como uma coleção de propriedades RDF para representar consultas SPARQL como triplas RDF. O vocabulário SPIN inclui construtores, tais como *spin:rule*, *spin:constraint* e *spin:constructor*, possibilitando consultas, definição de regras e restrições.

PUC-Rio - Certificação Digital Nº 1912717/CA

2.3 Inferência

A inferência pode ser definida como a capacidade de explicitar conhecimentos, ou seja, gerar novas triplas, com base nas instâncias iniciais (triplas anteriores) utilizando uma máquina de regras [17]. As máquinas de regras (*reasoning engine, rules engine, reasoner*) são componentes que inferem consequências lógicas e devem inferir fatos conhecidos [19].

Uma regra de inferência pode ser definida através do uso do SPARQL CONSTRUCT no *spin:rule*, onde se as condições previstas na cláusula WHERE forem satisfeitas, então as triplas presentes no CONSTRUCT serão criadas. Nas figuras 2.2 e figura 2.3 pode-se observar um exemplo de uma regra de inferência que cria uma tripla do tipo *tuning:QueryStatement* (Figura 2.2 - linha 1 a 3) se a condição descrita no bloco WHERE (Figura 2.2 - linha 4 a 8) for atendida, nesse caso é verificado se a descrição da consulta *tuning:hasDMLDescription* se inicia com "SELECT" (Figura 2.2 - linha 7). A Figura 2.3 exibe a representação RDF do SPARQL CONSTRUCT da Figura 2.2.

[4]https://spinrdf.org/

```
spin:rule
(1) CONSTRUCT {
(2)   ?this a tuning:QueryStatement .
(3) }
(4) WHERE {
(5)   ?this a tuning:AtomicStatement .
(6)   ?this tuning:hasDMLDescription ?desc .
(7)   FILTER STRSTARTS(UCASE(?desc), "SELECT") .
(8) }
```

Figura 2.2: Regra Representada em SPARQL

```
spin:rule [
      rdf:type sp:Construct ;
      sp:templates (
          [
            sp:object tuning:QueryStatement ;
            sp:predicate rdf:type ;
            sp:subject spin:_this ;
          ]
        ) ;
      sp:where (
          [
            sp:object tuning:AtomicStatement ;
            sp:predicate rdf:type ;
            sp:subject spin:_this ;
          ]
          [
            sp:object [
                sp:varName "desc" ;
              ] ;
            sp:predicate tuning:hasDMLDescription ;
            sp:subject spin:_this ;
          ]
          [
            rdf:type sp:Filter ;
            sp:expression [
                rdf:type sp:strstarts ;
                sp:arg1 [
                    rdf:type sp:ucase ;
                    sp:arg1 [
                        sp:varName "desc" ;
                      ] ;
                  ] ;
                sp:arg2 "SELECT" ;
              ] ;
          ]
        ) ;
    ] ;
```

Figura 2.3: Regra Representada em RDF

2.4 Contexto e Trabalhos Relacionados

A sintonia fina é algo muito importante para o desempenho de um Sistema de Banco de Dados. Entretanto, nem sempre a justificativa da escolha

PUC-Rio - Certificação Digital Nº 1912717/CA

de uma ação fica documentada de forma declarativa e semântica, sendo necessário um prévio conhecimento do domínio do negócio ou experiência no assunto.

Algumas pesquisas realizadas pelo grupo BioBD da PUC-Rio têm trabalhado na automatização das escolhas de ações de tuning, através de heurísticas que detalham o raciocínio utilizado por um DBA ou especialista na área de sintonia fina. Dentre esses trabalhos de pesquisa, podemos citar o trabalho de Eduardo Morelli [10], que segue a linha de estudos da auto-sintonia de índices, com ênfase na criação, eliminação e recriação de índices que estendeu o trabalho de Marcos Salles [7]. Ambos recorreram a agentes de *software* para automatização das atividades relacionadas a sintonia de índices. Além disso, na linha de criação de agentes, o trabalho [20] cria estratégias automáticas de manutenção no projeto físico de banco de dados, inclusive em tempo de execução de forma não intrusiva, a abordagem independe do SGBD, sendo utilizado em estruturas de índices, visões materializadas, particionamentos de tabelas e estruturas físicas duplicadas. Rafael de Oliveira [21] propôs a combinação de ações de sintonia fina automática para banco de dados relacional.

Outro trabalho do grupo é o do *framework* Outer-tuning, que é um *framework* cujo propósito é facilitar o trabalho do DBA ou de outras pessoas que venham interagir com banco de dados relacionais. Esse *framework* propõe ações de sintonia fina, tais como a criação de índices e visões materializadas, através da exibição dos custos e benefícios que essas ações proporcionam, assim como os comandos SQL beneficiados em conjunto com ações propostas. Esse *framework* tem a opção de ser executado de forma semi-automática (só propõe e não executa as ações no banco de dados) ou automática (propõe e executa as ações no banco de dados) [4].

O diferencial do trabalho proposto em relação aos demais do mesmo grupo está na utilização da ontologia com regras em SPIN no processo de inferência de estrutura de acessos, tais como os índices e as visões materializadas.

Outros trabalhos encontrados na literatura usam ontologia para a inferência de conhecimento. Dentre eles podemos citar uma ontologia de Diretrizes da prática Clínica (CPGs) que demonstrou eficiência em relação aos fatos inferidos e ao tempo de execução dessas inferências através de regras SPIN [22]. Outro trabalho é o [23], que usa uma ontologia de fitoterapia tradicional para inferir, através de regras em SWRL, recomendações para fitoterápicos tradicionais de forma semelhante à que um especialista faria.

Atualmente, trabalhos de *tuning* de banco de dados estão aplicando técnicas de aprendizado de máquina (*Machine Learning* - ML). Dentre eles podemos citar:

PUC-Rio - Certificação Digital Nº 1912717/CA

- O OtterTune, uma ferramenta automática de *tuning* independente de SGBD que utiliza a combinação de métodos supervisionados e não supervisionados de aprendizado de máquina que recomenda configurações para os parâmetros do SGBD [24].
- O CDBTune+, um sistema de *tuning* automático que usa modelo de *Deep Reinforcement Learning* (RL), com objetivo de recomendar configurações para os parâmetros os bancos de dados em nuvem [25].
- O sintonizador proposto no trabalho [26], que recomenda índices comparando o custo de dois planos de execução realizado por uma tarefa de classificação de ML. A abordagem usa também uma tarefa de regressão em ML que minimiza o erro de previsão para cada um dos planos. Com um classificador treinado, vetorizado e sincronizado com o otimizador, o sintonizador obtém planos para configurações hipotéticas e promove o classificador a prever se o plano de uma determinada consulta é menos custoso em comparação com a configuração existente.
- Os módulos utilizando sintonizadores baseados em aprendizado por reforço desenvolvidos no trabalho [27], que ajustam em tempo real as configurações dos parâmetros dos serviços de banco de dados relacionais conforme as cargas de trabalhos do momento. O módulo monitora o banco de dados, aprende com as novas cargas de trabalhos ajustando as métricas existentes, aplica e valida as recomendações oriundas dos sistemas em produção.
- O Sistema de Classificação de Aprendizagem (LCS) utilizado no trabalho [28], que são classificadores representados por regras de inferência, combinando aprendizagem por reforço e algoritmos genéticos. Além de criar e excluir índices, a abordagem também movimenta um índice localizado em um HDD e transfere para um SSD conforme a necessidade da carga de trabalho. Para cada ação proposta e realizada, o sistema informa os benefícios adquiridos. A proposta permite seu uso em outros Sistemas de Gerenciamento de Banco de Dados Relacional (SGBDRs), requerendo poucos ajustes. O trabalho [28] minimiza algumas complexidades para um DBA na criação e manutenção de índices principalmente em ambientes de bancos de dados com armazenamento híbrido, ora HDD, ora SSD ou ambos para uma mesma solução.

Ainda usando ML com Inteligência Artificial (IA), há um trabalho que propõe um banco de dados totalmente automatizado (autoconfiguração, auto otimização, automonitoramento, autodiagnóstico, autocorreção e automontagem), tornando-se um banco de dados totalmente automatizado nativo em

PUC-Rio - Certificação Digital Nº 1912717/CA

IA [29]. Percebe-se que, nos últimos anos, muitos trabalhos de otimização de banco de dados e sintonia fina estão usando ML, IA e Redes Neurais (RN). O trabalho [30] propõe um plano de consulta ideal baseado em RN representando codificação com semântica.

O trabalho [31] desenvolveu diversos componentes que automatizam para criação e sintonia de índices para o Azure SQL Database. A implementação está em funcionamento desde 2019 com resultados satisfatórios para seus milhões de clientes, sintonizando banco de dados também como serviço onde cada cliente tem negócios e restrições bem distintas. Esta proposta de trabalho, além de criar índices automatizados, realiza ajustes nos equipamentos físicos e gerenciamento de cada banco de dados, geralmente criando uma réplica deste, efetuando validações, testes e disponibilização transparentemente e sem o cliente perceber as trocas. Todos os serviços podem ser operados manualmente por usuários permitindo ou não cada ajuste. A implementação é componetizada de forma assíncrona com o Azure.

Nessa pesquisa de dissertação foi investigado como uma ontologia pode ser útil como um artefato de raciocínio e abordagem baseada em regras para sugerir possíveis ações de sintonia fina para DBAs com sua explicabilidade.

PUC-Rio - Certificação Digital Nº 1912717/CA

2.5 Resumo do Capítulo

Neste capítulo foram abordados os conceitos de sintonia fina, otimização, ontologia e, por fim, o contexto e trabalhos relacionados com essa pesquisa.

No capítulo 3 será apresentada a ontologia OnDBTuning, a decomposição SQL, as modelagens utilizadas para as regras de *tuning*, utilidade e bônus.

3
Inferência na OnDBTuning

O presente capítulo apresenta a ontologia utilizada nesse trabalho de pesquisa, a OnDBTuning, e as contribuições deste trabalho a esta ontologia, que foram a decomposição SQL, a modelagem do processo de instanciação, o desenvolvimento das regras de ações de *tuning* e os indicadores, conforme os conceitos presentes na ontologia OnDBTuning. Estas modelagens serviram de base para a descrição das regras em SPIN acrescentadas na OnDBTuning.

3.1
OnDBTuning

OnDBTuning [1], the Database Tuning Ontology, foi proposta inicialmente por Ana Carolina Almeida em sua tese em 2013 [3], e vem sendo redefinida e modelada com a colaboração de outros pesquisadores ao longo dos anos, sendo hoje uma ontologia bem fundamentada [32]. Essa ontologia define conceitos, propriedades e relacionamentos usados no domínio de sintonia fina de banco de dados. Entre esses conceitos, podemos destacar o *tuning:DMLStatement*, que representa a Linguagem de Manipulação de Dados (DML). Esta linguagem acessa e manipula dados das tabelas de um banco de dados relacional, conforme a Figura 3.1, que mostra um fragmento desta ontologia.

[1] `https://www.ime.uerj.br/ondbtuning/`

PUC-Rio - Certificação Digital Nº 1912717/CA

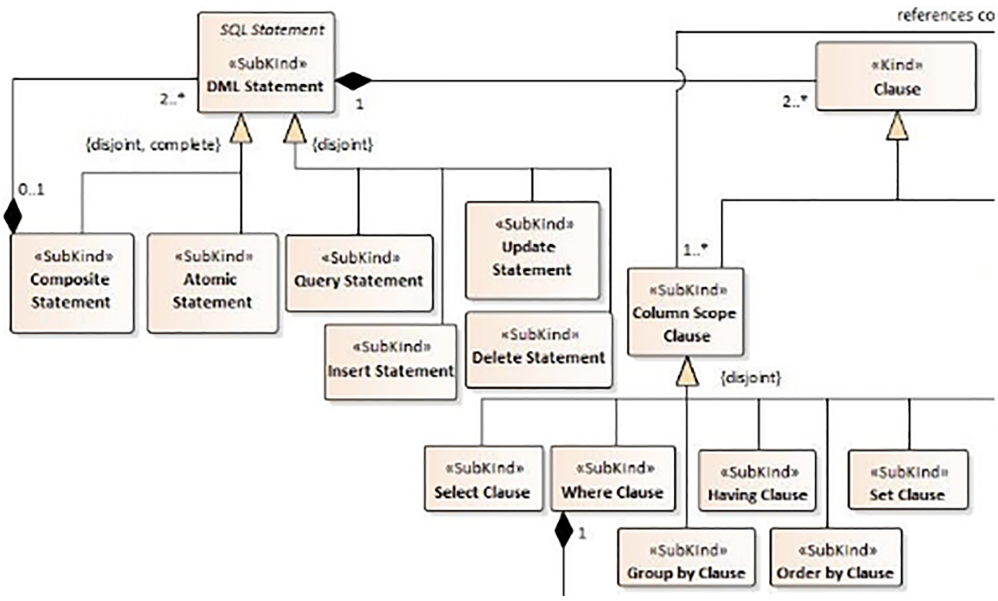

Figura 3.1: Fragmento da OnDBTuning [3]

Inicialmente, as regras para inferência na OnDBTuning foram escritas em SWRL [33]. No entanto, a escolha pelo uso do SPIN para descrever regras de *tuning* foi devido às restrições tecnológicas e de expressividade do SWRL. Uma das restrições tecnológicas encontradas com o SWRL foi o erro conhecido como estouro de pilha [34]. Mesmo sendo executado em uma máquina com 32 GB de RAM e também com o módulo do SWRLTab[2] separado do Protégé[3] , o erro aconteceu. Ocorreu também erro na execução, por conta da ordenação das referências nas regras no Protégé em suas versões mais recentes (superiores à versão 4).

Outra restrição é não haver possibilidade de se considerar o operador lógico "ou" para regras SWRL e também existe uma limitação no tamanho (quantidade de caracteres) das regras. Sendo assim, é necessária a utilização de várias regras, uma para cada tipo de combinação de cláusulas possíveis em uma consulta DML, tornando bem complexa a criação e manutenção dessas regras [3].

Devido aos recursos existentes na SPARQL, tais como FILTER, OPTIONAL, UNION e a possibilidade de definir novas funções e *templates* no próprio SPIN, o SPIN é mais expressivo do que o SWRL. Outra característica é em relação ao desempenho na execução das regras, pois no SPIN as regras são verificadas quando ocorrem novas instâncias ou alterações dessas, e no SWRL é necessário verificar todas as regras a cada execução realizada pela máquina de regras [17].

[2]https://github.com/protegeproject/swrltab
[3]https://protege.stanford.edu/

PUC-Rio - Certificação Digital Nº 1912717/CA

3.2 Decomposição SQL

As consultas da carga de trabalho instanciada na ontologia foram submetidas ao processo de decomposição (*parsing*), visto que as heurísticas de *tuning* dependem da estrutura dessas consultas. Esse processo usado para dividir SQL em nós indivisíveis foram compostos por 6 (seis) *spin:rules*. Foi utilizado o comando CONSTRUCT para inferir novas triplas, categorizadas por seus nós e relacionamentos baseados nas classes e propriedades da OnDBTuning. A Figura 3.2 mostra o resultado de uma regra de decomposição de uma cláusula WHERE (*tuning:WhereClause*), sendo:

- na linha 1 - a URI, cujo NAME é *tuning:DML_TPCH_4_WHERE*;
- da linha 2 a 6 - as colunas referenciadas na cláusula WHERE (*tuning:clauseReferencesColumn*);
- na linha 7 - a identificação da consulta de origem (*tuning:componentOfDMLSt*);
- na linha 8 - a descrição da cláusula WHERE (*tuning:hasClauseDescription*);
- na linha 9 - a identificação a qual conceito pertence (*rdf:type tuning:WhereClause*).

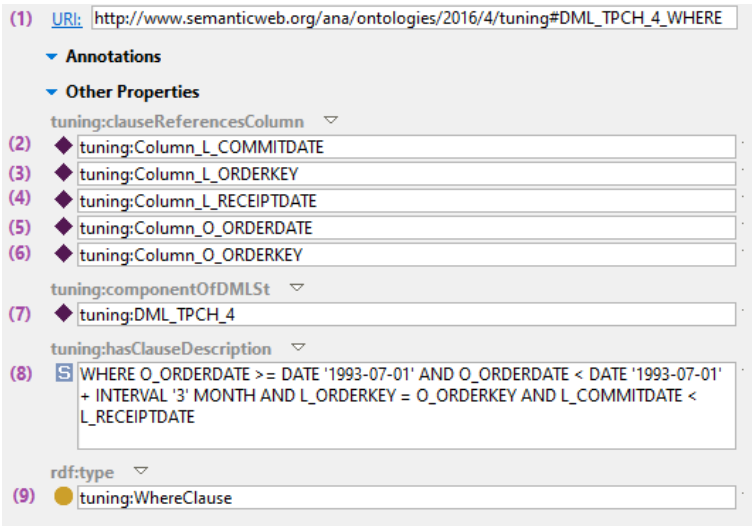

Figura 3.2: Decomposição do WhereClause

Instanciar conceitos da ontologia através de regras associadas às suas classes torna o raciocínio explícito e consistente. Isso também torna a ontologia auto-contida e com forte coesão, reduzindo a dependência externa e aumentando a reutilização. Na presença de novas heurísticas de *tuning* de banco de dados que se baseiam em um conceito não definido previamente, é bastante

PUC-Rio - Certificação Digital Nº 1912717/CA

simples estender a ontologia com as classes, propriedades, e as suas respectivas regras de *parse*.

3.3 Modelagem das Regras de Tuning

Para melhor elucidação das etapas necessárias para inferência das regras, foi modelado um diagrama de estados (Figura 3.3) que representa os estados e eventos do processo de instanciação na OnDBTuning. No estado inicial, os indivíduos da ontologia são representados pelas classes e propriedades do domínio de sintonia fina. Ao receber uma carga de trabalho composta pelas consultas e os metadados do banco de dados, novas triplas são criadas (*asserted*) como indivíduos com suas classes e respectivas propriedades na ontologia. Depois que ocorre a execução das regras (*run rules*) por uma máquina especializada, novas triplas são criadas (*inferred*). Essas inferências são resultados das regras de decomposição da linguagem SQL e das regras de ações de *tuning*. Cabe observar que estamos sugerindo ações de *tuning*. A efetiva decisão sobre a ação, depende de um DBA ou sistema programado para isso.

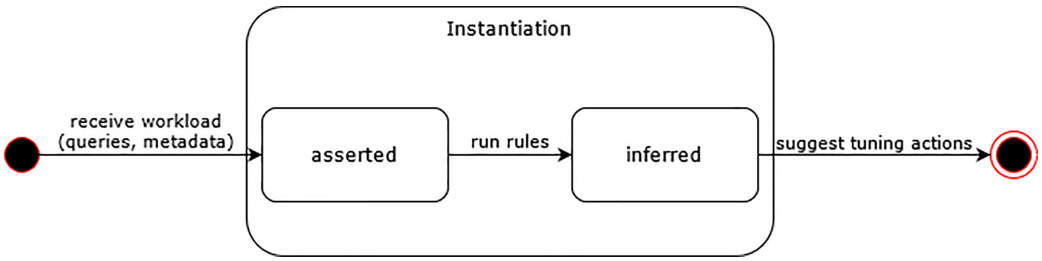

Figura 3.3: Diagrama de Estados de Instâncias da Ontologia

3.3.1 Regra do Índice Simples e Composto

Os diagramas de atividades das Figuras 3.4 e 3.5, mostram respectivamente os passos das regras usadas para sugerir índices simples (não composto) e índices compostos. Segue a descrição dos passos presentes nesses diagramas:

- **get DMLStatement**: para cada instância de uma *query*, verifica se existe a cláusula "WHERE";
- **get ReferencedColumn**: seleciona cada coluna referenciada na cláusula "WHERE". No diagrama do índice composto, seleciona um par de colunas referenciadas na cláusula "WHERE" e verifica se essas duas colunas diferem e se pertencem a mesma tabela;

PUC-Rio - Certificação Digital Nº 1912717/CA

– **get CompositeExpression**: para cada expressão composta da cláusula "WHERE", verifica se essa possui um conector lógico "OR";

– **get Index**: verifica se a coluna selecionada tem um índice (físico). Caso seja um índice composto, verifica se o par de colunas selecionado tem um índice (físico);

– **suggest Index**: sugere um índice baseado na coluna selecionada, caso não exista essa coluna indexada. No diagrama do índice composto, sugere um índice baseado no par de colunas selecionadas, caso não exista esse par de colunas indexada.

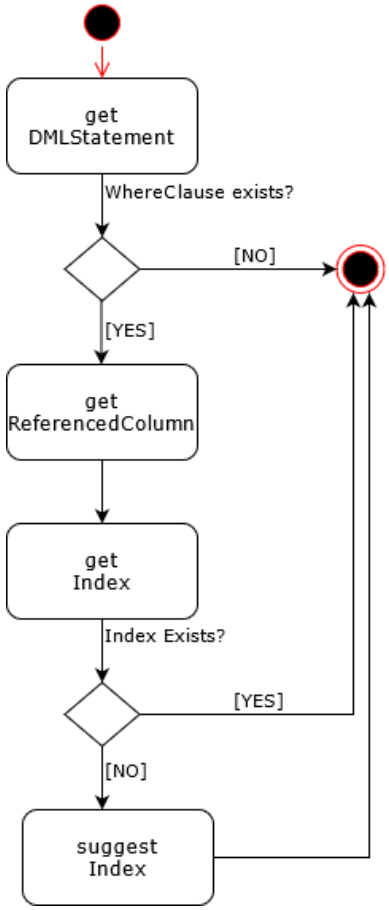

Figura 3.4: Diagrama de Atividades - Índice Simples

PUC-Rio - Certificação Digital Nº 1912717/CA

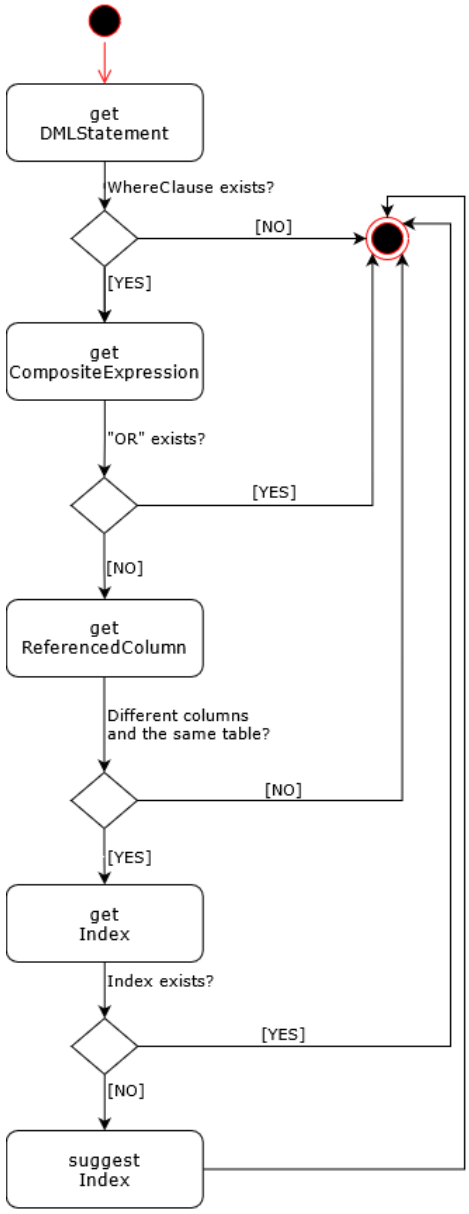

Figura 3.5: Diagrama de Atividades - Índice Composto

Por exemplo, a consulta da Figura 3.6 possui sete colunas referenciadas na cláusula WHERE (linha seis a dez), tendo como colunas indexadas: C_CUSTKEY, O_CUSTKEY, L_ORDERKEY e O_ORDERKEY. Após aplicar as regras das Figuras 3.4 e 3.5, temos, respectivamente, sugestões de três índices simples e dez índices compostos listados na Tabela 3.1. Sendo a nomenclatura para o índice simples, constituída do prefixo "HYP_" + "nome da coluna indexada" e para o índice composto do prefixo "HYP_" + "nome da primeira coluna indexada" + "_nome da segunda coluna indexada".

PUC-Rio - Certificação Digital Nº 1912717/CA

```
1  SELECT      L_ORDERKEY,
2              SUM(L_EXTENDEDPRICE + (1 - L_DISCOUNT)) AS REVENUE,
3              O_ORDERDATE,
4              O_SHIPPRIORITY
5  FROM        CUSTOMER, ORDERS, LINEITEM
6  WHERE       C_MKTSEGMENT = 'BUILDING' AND
7              C_CUSTKEY = O_CUSTKEY AND
8              L_ORDERKEY = O_ORDERKEY AND
9              O_ORDERDATE < DATE '1995-03-15' AND
10             L_SHIPDATE > DATE '1995-03-15'
11 GROUP BY    L_ORDERKEY, O_ORDERDATE, O_SHIPPRIORITY
12 ORDER BY    REVENUE DESC, O_ORDERDATE;
```

Figura 3.6: Consulta Instanciada Q3- TPC-H[4]

ÍNDICE SIMPLES	ÍNDICE COMPOSTO
HYP_C_MKTSEGMENT	HYP_C_CUSTKEY_C_MKTSEGMENT
HYP_L_SHIPDATE	HYP_C_MKTSEGMENT_C_CUSTKEY
HYP_O_ORDERDATE	HYP_L_ORDERKEY_L_SHIPDATE
	HYP_L_SHIPDATE_L_ORDERKEY
	HYP_O_CUSTKEY_O_ORDERDATE
	HYP_O_ORDERDATE_O_CUSTKEY
	HYP_O_CUSTKEY_O_ORDERKEY
	HYP_O_ORDERKEY_O_CUSTKEY
	HYP_O_ORDERDATE_O_ORDERKEY
	HYP_O_ORDERKEY_O_ORDERDATE

Tabela 3.1: Sugestões de Índices Simples e Índices Compostos

3.3.2 Regra do Índice Parcial

O diagrama de atividades da Figura 3.7 mostra os passos da regra utilizada para sugerir um índice parcial simples. O passo **get DMLStatement** foi definido na subseção 3.3.1. Abaixo segue descrição dos outros passos:

- **get AtomicExpression**: para cada expressão atômica da cláusula "WHERE" verifica se essa tem um literal;
- **suggest PartialIndex**: sugere um índice parcial baseado na coluna selecionada e no literal.

[4]http://www.tpc.org/tpch/

PUC-Rio - Certificação Digital Nº 1912717/CA

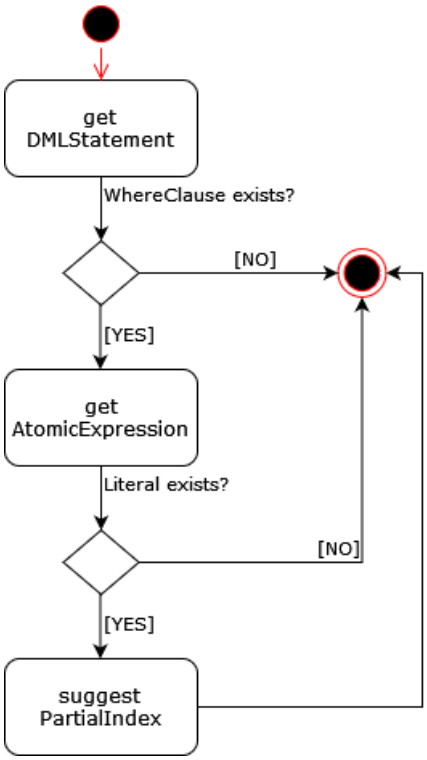

Figura 3.7: Diagrama de Atividades - Índice Parcial Simples

Como exemplo de aplicação da regra da Figura 3.7, a consulta da Figura 3.6 sugere índices parciais com as expressões condicionais listadas na Tabela 3.2.

EXPRESSÃO CONDICIONAL
WHERE C_MKTSEGMENT = 'BUILDING'
WHERE L_SHIPDATE >DATE '1995-03-15'
WHERE O_ORDERDATE <DATE '1995-03-15

Tabela 3.2: Expressão Condicional de Índice Parcial

3.3.3 Regra da Visão Materializada

O diagrama de atividades da Figura 3.8 mostra os passos da regra utilizada para sugerir uma visão materializada. Essa regra foi uma adaptação dos passos do algoritmo DefineView [11] utilizado para definir uma visão.

- **get DMLStatement**: para cada instância de uma *query*, verifica se existe uma cláusula "WHERE" e um "GROUP BY";
- **get SelectClause**: seleciona a descrição da cláusula "SELECT";

PUC-Rio - Certificação Digital Nº 1912717/CA

- **get ReferencedColumn from HavingClause**: se houver a cláusula "HAVING", seleciona a coluna referenciada dessa cláusula;
- **get filter ReferencedColumn from AtomicExpress**: seleciona as colunas referenciadas na cláusula "WHERE" que são comparadas com um literal;
- **get FromClause**: seleciona a descrição da cláusula "FROM";
- **get join AtomicExpression**: seleciona expressões atômicas que contenham JOIN;
- **get non-aggregated ReferencedColumn from SelectClause**: seleciona colunas referenciadas no **SelectClause** e não agregadas (que não estejam na parte com SUM e COUNT, por exemplo); e
- **suggest Materialized View**: sugere uma visão materializada.

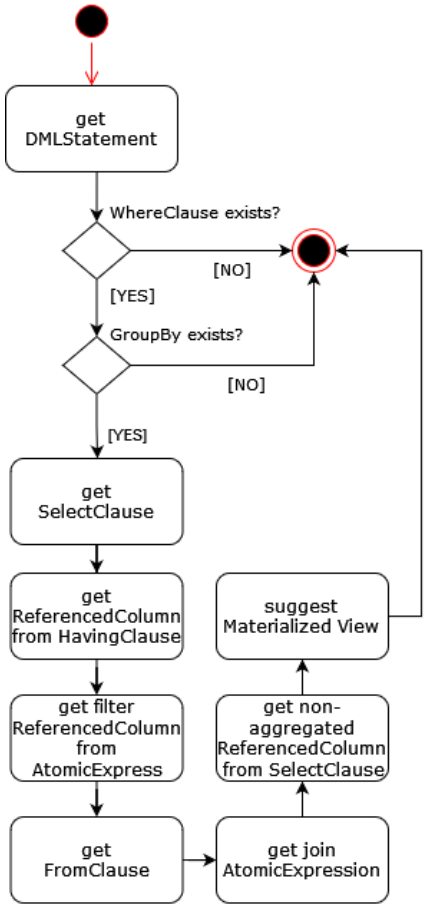

Figura 3.8: Diagrama de Atividades - Visão Materializada

A Figura 3.9 contém a descrição da visão materializada sugerida pela aplicação da regra da Figura 3.8 na consulta da Figura 3.6.

PUC-Rio - Certificação Digital Nº 1912717/CA

```
1   SELECT      L_ORDERKEY,
2               SUM(L_EXTENDEDPRICE + (1 - L_DISCOUNT)) AS REVENUE,
3               O_ORDERDATE,
4               O_SHIPPRIORITY,
5               C_MKTSEGMENT,
6               L_SHIPDATE
7   FROM        CUSTOMER, ORDERS, LINEITEM
8   WHERE       C_CUSTKEY = O_CUSTKEY AND
9               L_ORDERKEY = O_ORDERKEY
10  GROUP BY    L_ORDERKEY, O_ORDERDATE, O_SHIPPRIORITY, C_MKTSEGMENT, L_SHIPDATE
```

Figura 3.9: Descrição da Visão Materializada

3.4 Indicadores

Nesse trabalho de pesquisa, foi percebida a necessidade de criar indicadores que permitissem uma tomada de decisão mais qualificada, tendo em vista que as ações de *tuning* inferidas nesse estudo de caso são somente de sugestões. Esses indicadores foram o bônus para os índices hipotéticos e a utilidade para os índices reais criados como novas propriedades na OnDBTuning. Eles foram inspirados na Heurística de Benefícios do grupo de pesquisa BioBD, que contém cálculos que avaliam o benefício de um índice, de modo a decidir quais índices devem ser criados ou removidos [7].

3.4.1 Bônus

O bônus é um indicador que objetiva verificar o quão benéfico um índice hipotético sugerido pode ser para uma carga de trabalho. O cálculo do bônus refere-se ao total de consultas que originam o índice hipotético sugerido dividido pela soma das consultas que referenciam a tabela correspondente. A ideia contida é que quanto mais próximo o valor do bônus estiver de 1, melhor. Isso indica que pode ser útil na maioria das consultas presentes na carga de trabalho acumulada. Acima de um limite definido pelo usuário, esse bônus pode indicar serem índices hipotéticos candidatos a se tornarem índices reais. Para cada índice hipotético sugerido, calculamos seu bônus com base na equação 3-1:

$$bonus = \frac{count(hypIndex \rightarrow queries)}{count(hypIndex \rightarrow column \leftarrow table \leftarrow queries)} \quad (3\text{-}1)$$

Como exemplo de aplicação do cálculo do bônus, se tivermos um índice hipotético que tenha sido sugerido por somente uma consulta de uma carga de trabalho que contenha um total de cinco consultas que referenciam a tabela

PUC-Rio - Certificação Digital Nº 1912717/CA

a qual ele pertence, o valor do bônus será de 0,16667. Nesse caso, esse valor indica que ele não beneficia a maioria das consultas dessa carga de trabalho.

3.4.2 Utilidade

O indicador denominado utilidade foi criado para verificar o quanto um índice real pode vir a ser útil para uma carga de trabalho. A utilidade é em função da coluna indexada estar presente na cláusula **WHERE** de um DML, ou seja, não está sendo verificado se o índice real será utilizado quando a consulta é executada em um SGBD e sim que poderá vir a ser.

O cálculo da utilidade do índice real é obtido pela divisão do total das colunas indexadas na cláusula **WHERE** pelo total de DML que referencia a sua tabela. Quanto mais próximo o resultado estiver de 1, melhor, pois indica que o índice real pode ser útil para a maioria das DMLs da carga de trabalho acumulada. Valores próximos a 0 sugerem índices candidatos a serem eliminados, pois eles requerem manutenção e não estão sendo usados.

Por exemplo, as Figuras 3.10 e 3.11 mostram o recorte de duas consultas de uma carga de trabalho. Essas consultas com suas descrições completas estão presentes nas Figuras A.5 e A.6 no apêndice A. Nelas, a coluna indexada **S_SUPPKEY**, destacada em azul, está presente somente nessas duas consultas, e a tabela a qual ela pertence, **SUPPLIER**, destacada em verde, só aparece na cláusula FROM dessas consultas. Com isso, ao se aplicar o cálculo da utilidade, teremos como resultado o valor 1, ou seja, esse índice real poderá vir a ser utilizado e ser útil para essa carga de trabalho.

```
SELECT ...
FROM CUSTOMER, ORDERS, LINEITEM,
SUPPLIER, NATION, REGION

WHERE
C_CUSTKEY = O_CUSTKEY AND
L_ORDERKEY = O_ORDERKEY AND
L_SUPPKEY = S_SUPPKEY AND
C_NATIONKEY = S_NATIONKEY ...
```

Figura 3.10: Recorte da Consulta Q5

```
SELECT ...
FROM PART, SUPPLIER, LINEITEM, ORDERS,
CUSTOMER, NATION, REGION

WHERE
P_PARTKEY = L_PARTKEY AND
S_SUPPKEY = L_SUPPKEY AND
L_ORDERKEY = O_ORDERKEY ...
```

Figura 3.11: Recorte da Consulta Q8

PUC-Rio - Certificação Digital Nº 1912717/CA

3.5 Resumo do Capítulo

Este capítulo apresentou a ontologia utilizada nesse trabalho de pesquisa, a OnDBTuning, e as contribuições deste trabalho a esta ontologia, que foram a decomposição SQL, a modelagem do processo de instanciação, o desenvolvimento das regras de ações de *tuning* utilizadas para inferência nesse trabalho de pesquisa e os indicadores.

No próximo capítulo será abordado o estudo de caso com seu ambiente e carga de trabalho, as regras utilizadas para inferência e os resultados das inferências de índices hipotéticos simples e composto, bônus, utilidade, índice parcial e visões materializadas.

PUC-Rio - Certificação Digital Nº 1912717/CA

4
Estudo de Caso

Nesse capítulo abordaremos o ambiente, carga de trabalho e regras utilizadas para inferências de uma ontologia no domínio em sintonia fina de banco de dados, visando a sugerir ações de sintonia fina e indicadores para uma carga de trabalho, de modo a apoiar a tomada de decisão.

4.1
Ambiente e Carga de Trabalho

Neste experimento foi utilizado o Sistema Operacional Windows 10 Home Single Language, Intel Core i7 com 2.40GHz, 8GB de RAM e 1TB de disco rígido.

Nas alterações da ontologia OnDBTuning utilizou-se a ferramenta TopBraid Composer Free Edition[1] na versão 6.0.1, que através de sua máquina de regras denominada TopSPIN possibilitou a inferência de novos indivíduos da ontologia. Apesar de ser um *reasoner* comercial, foi possível inferir com a versão gratuita do *software* nesse trabalho de pesquisa. De acordo com [35], o TopBraid é uma das ferramentas mais utilizadas na criação das ontologias, assim como o Protégé, OWLGrED[2] e SWOOP[3]. Essa ferramenta fornece uma *interface* gráfica que facilita a manutenção e visualização das instâncias e inferências, bem como permite a visualização e edição das regras em outras extensões, tais como RDF/XML, TURTLE, N-TRIPLE e JSON-LD. Também permite a exportação do resultado das execuções SPARQL nos formatos JSON, RDF e XML, o que auxilia em uma futura interoperabilidade sistêmica.

Para descrever as regras de ações de *tuning* modeladas na seção 3.3 e a decomposição SQL foi utilizada a linguagem SPIN. Além disso foi instanciado um subconjunto do esquema do TPC-H[4] contendo sete tabelas e 30 colunas, além de 13 índices reais constituído por sete chaves primárias e seis chaves estrangeiras existentes. Também foram instanciadas cinco consultas (Q1, Q3, Q4, Q5 e Q8) do TPC-H, que possui uma carga de trabalho analítica (*Online*

[1] `https://www.topquadrant.com/`
[2] http://owlgred.lumii.lv/
[3] https://www.w3.org/2001/sw/wiki/SWOOP
[4] http://www.tpc.org/tpch/

PUC-Rio - Certificação Digital Nº 1912717/CA

Analytical Processing - OLAP), conforme apêndice A. Esses elementos foram instanciados, ou seja, adicionados na OnDBTuning.

Foram realizadas duas cargas de trabalho na ontologia. A primeira foi composta de cinco consultas do TPC-H e após a execução das regras pela máquina de regras, suas inferências foram persistidas na ontologia. Uma segunda carga foi instanciada na ontologia, composta por uma consulta do TPC-H que já havia sido contemplada na carga anterior, porém com literais diferentes. Essas duas cargas de trabalho tiveram como objetivo analisar o resultado das inferências, a fim de verificar se haveria conceitos duplicados inferidos e também a evolução dos valores inferidos para o bônus e a utilidade.

4.2 Regras de Inferência de Bônus e Utilidade

Para o cálculo do bônus (*tuning:hasHypBonus*) foi utilizada a regra da Figura 4.1. Essa utilizou a função *spif:countMatches*, que soma as ocorrências de uma tripla. Nessa figura, observa-se a soma das quantidades de consultas que originam o índice hipotético (*tuning:originatesIndSpec*) e a soma das consultas que referenciam a tabela correspondente (*tuning:clauseReferencesTable*).

PUC-Rio - Certificação Digital Nº 1912717/CA

spin:rule

```
# Total de DML relacionados ao índice hipotético /
# Total de DML que acessam a tabela do índice hipotético
CONSTRUCT {
  ?this tuning:hasHypBonus ?valor .
}
WHERE {
  ?this a tuning:HypotheticalIndex .
  BIND (spif:countMatches(?this, tuning:originatesIndSpec, ?dml) AS ?bonus) .
  ?this tuning:hypIndexesColumn ?col .
  ?t1 tuning:constitutedOf ?col .
  BIND (spif:countMatches(?s1, tuning:clauseReferencesTable, ?t1) AS ?total) .
  BIND ((?bonus / ?total) AS ?valor) .
}
```

Figura 4.1: Regra do Bônus

Para cada índice real, foi realizado o cálculo da sua utilidade (*tuning:hasIndexUsefulness*), dividindo o total *tuning:WhereClause*, das colunas indexadas presentes, pelo total *tuning:DMLStatment*, que faz referência a sua tabela. As Figuras 4.2 e 4.3 demonstram as regras utilizadas para o cálculo da utilidade de um índice real simples e composto, respectivamente.

```
# para índice simples
CONSTRUCT {
  ?this tuning:hasIndexUsefulness ?use .
}
WHERE {
  {
    SELECT ((?conta / ?total) AS ?use) ?this
    WHERE {
      ?this a tuning:Index .
      ?this tuning:FirstColumn ?col .
      ?t1 a tuning:ReferencedTable .
      ?t1 tuning:constitutedOf ?col .
      BIND (spif:countMatches(?s3, tuning:clauseReferencesTable, ?t1) AS ?total) .
      {
        SELECT ?this (COUNT(?s) AS ?conta)
        WHERE {
          ?this a tuning:Index .
          ?this tuning:FirstColumn ?col .
          ?s a tuning:WhereClause .
          ?s tuning:clauseReferencesColumn ?col .
          FILTER NOT EXISTS {
            ?this tuning:SecondColumn ?col2 .
          } .
        }
        GROUP BY ?this
      } .
    }
  } .
}
```

Figura 4.2: Regra de Utilidade - Índice Simples

```
# para indice composto
CONSTRUCT {
  ?this tuning:hasIndexUsefulness ?use .
}
WHERE {
  {
    SELECT ((?conta / ?total) AS ?use) ?this
    WHERE {
      ?this a tuning:Index .
      ?this tuning:FirstColumn ?col .
      ?t1 a tuning:ReferencedTable .
      ?t1 tuning:constitutedOf ?col .
      BIND (spif:countMatches(?s3, tuning:clauseReferencesTable, ?t1) AS ?total) .
      {
        SELECT (COUNT(?s) AS ?conta) ?this
        WHERE {
          ?this a tuning:Index .
          ?this tuning:FirstColumn ?col .
          ?this tuning:SecondColumn ?col2 .
          ?s a tuning:WhereClause .
          ?s tuning:clauseReferencesColumn ?col .
          ?s tuning:clauseReferencesColumn ?col2 .
        }
        GROUP BY ?this
      } .
    }
  } .
}
```

Figura 4.3: Regra de Utilidade - Índice Composto

4.3 Regras de Inferência de Índices Simples e Composto

Foram consideradas funções inerentes do SPIN em conjunto com SPARQL na implementação da decomposição das cláusulas SQL e das regras de *tuning*. No processo da decomposição, a função *spin:split* permitiu a divisão de algumas *strings* em *substrings*.

PUC-Rio - Certificação Digital Nº 1912717/CA

As regras de sugestões de ações de sintonia fina denominadas *tuning:RuleHypCompositeIndex* e *tuning:RuleHypSimpleIndex* foram instanciadas na OnDBTuning, sendo essas instâncias de *tuning:HeuristicDefinitionRule* e implementadas por *spin:rules* anexadas ao conceito *tuning:DMLStatement*. Com a utilização da propriedade *tuning:isGeneratedBy* da tripla gerada por cada regra, o usuário pode identificar e até mesmo filtrar inferências de uma determinada regra de *tuning*.

Na regra de *tuning* denominada *tuning:RuleHypSimpleIndex* da Figura 4.4, conforme modelagem da subseção 3.3.1, para cada coluna referenciada na cláusula WHERE (*tuning:WhereClause*) de um padrão SQL, foi verificado se existe índice para aquela coluna (FILTER NOT EXISTS); caso não exista, essa coluna é sugerida como um índice hipotético simples e novas triplas no bloco CONSTRUCT (Figura 4.4 - linha 2 a 7) são inferidas após a execução dessa regra.

```
(1)  CONSTRUCT {
(2)    ?uri a tuning:HypotheticalIndex .
(3)    ?uri tuning:hasHypotheticalIndexName ?hypName .
(4)    ?uri tuning:originatesIndSpec ?dml .
(5)    ?uri tuning:hypIndexesColumn ?col .
(6)    ?uri tuning:FirstColumn ?col .
(7)    ?uri tuning:isGeneratedBy tuning:RuleHypSimpleIndex .
(8)  }
(9)  WHERE {
(10)   ?s a tuning:WhereClause .
(11)   ?s tuning:componentOfDMLSt ?dml .
(12)   ?s tuning:clauseReferencesColumn ?col .
(13)   FILTER NOT EXISTS {
(14)     ?s1 a tuning:Index .
(15)     ?s1 tuning:FirstColumn ?col .
(16)     FILTER NOT EXISTS {
(17)       ?s1 tuning:SecondColumn ?col2 .
(18)     } .
(19)   } .
(20)   ?col tuning:hasColumnName ?nameCol .
(21)   BIND (CONCAT("HYP_", UCASE(?nameCol)) AS ?hypName) .
(22)   BIND
       (URI(CONCAT("http://www.semanticweb.org/ana/ontologies/2016
       /4/tuning#", ?hypName)) AS ?uri) .
(23) }
```

Figura 4.4: *RuleHypSimpleIndex*

Na regra de *tuning* denominada *tuning:RuleHypCompositeIndex* da Figura 4.5, conforme modelagem para índice composto da subseção 3.3.1, para cada par de coluna referenciada na cláusula WHERE (*tuning:WhereClause*), é verificado se já existe um índice (FILTER NOT EXISTS - linha 25 a 29) contendo esse par. Por exemplo, na figura 4.8, o índice LINEITEM_L_PARTKEY_FKEY (*tuning:hasIndexName*) tem como o par (*tuning:Column_L_PARTKEY*) na primeira posição (*tuning:FirstColumn*) e (*tuning:Column_L_SUPPKEY*) na segunda posição (*tuning:SecondColumn*). Nesse caso, essa regra só irá sugerir um índice composto para esse

PUC-Rio - Certificação Digital Nº 1912717/CA

par de colunas, (*tuning:Column_L_SUPPKEY*) na primeira posição (*tuning:FirstColumn*) e (*tuning:Column_L_PARTKEY*) na segunda posição (*tuning:SecondColumn*), tendo em vista que a ordem oposta já existe em um índice.

```
(1)  CONSTRUCT {
(2)     ?uri a tuning:HypotheticalIndex .
(3)     ?uri tuning:hasHypotheticalIndexName ?hypName .
(4)     ?uri tuning:hypIndexesColumn ?col1 .
(5)     ?uri tuning:hypIndexesColumn ?col2 .
(6)     ?uri tuning:originatesIndSpec ?dml .
(7)     ?uri tuning:FirstColumn ?col1 .
(8)     ?uri tuning:SecondColumn ?col2 .
(9)     ?uri tuning:isGeneratedBy tuning:RuleHypCompositeIndex .
(10) }
(11) WHERE {
(12)    ?s a tuning:WhereClause .
(13)    ?s tuning:componentOfDMLSt ?dml .
(14)    ?s tuning:clauseReferencesColumn ?col1 .
(15)    ?col1 tuning:hasColumnName ?nameCol1 .
(16)    ?tab tuning:constitutedOf ?col1 .
(17)    ?s tuning:clauseReferencesColumn ?col2 .
(18)    ?col2 tuning:hasColumnName ?nameCol2 .
(19)    ?tab tuning:constitutedOf ?col2 .
(20)    FILTER (?col1 != ?col2) .
(21)    ?composite a tuning:CompositeExpression .
(22)    ?composite tuning:hasPredicateDescription ?p .
(23)    BIND (IF(CONTAINS(UCASE(?p), " OR "), "", ?composite) AS ?c) .
(24)    ?c tuning:componentOfWhere ?s .
(25)    FILTER NOT EXISTS {
(26)       ?s1 a tuning:Index .
(27)       ?s1 tuning:FirstColumn ?col1 .
(28)       ?s1 tuning:SecondColumn ?col2 .
(29)    } .
(30)    BIND (CONCAT("HYP_", ?nameCol1, "_", ?nameCol2) AS ?hypName) .
(31)    BIND (URI(CONCAT("http://www.semanticweb.org/ana/ontologies/2016/4/tuning#", ?hypName)) AS ?uri) .
(32) }
```

Figura 4.5: *RuleHypCompositeIndex*

4.4 Inferências de Índices Simples e Composto

Para a primeira carga de trabalho, composta pelas cinco consultas do TPC-H, foram sugeridos 40 índices hipotéticos, sendo: nove índices simples pela regra *tuning:RuleHypSimpleIndex* e 31 índices compostos por duas colunas pela regra *tuning:RuleHypCompositeIndex*. Ademais, valores de bônus foram calculados (*tuning:hasHypBonus*) para cada índice hipotético.

A Figura 4.6 mostra um índice hipotético denominado *tuning:HYP_R_NAME* , inferido através da regra *tuning:RuleHypSimpleIndex*, identificado por seu URI (linha 1) e a descrição de suas propriedades, sendo:

- na linha 2 - *tuning:Coluna_R_NAME* é a primeira coluna referenciada (*tuning:FirstColumn*);
- na linha 3 - 1.0 é o valor do bônus (*tuning:hasHypBonus*);

PUC-Rio - Certificação Digital Nº 1912717/CA

– na linha 4 - HYP_R_NAME é o nome do índice hipotético (*tuning:hasHypotheticalIndexName*);

– na linha 5 - *tuning:Coluna_R_NAME* é a coluna indexada (*tuning:HypIndexesColumn*);

– na linha 6 - *tuning:RuleHypSimpleIndex* indica a regra de *tuning* (*tuning:isGeneratedBy*) que deu origem ao índice hipotético (*rdf:type tuning:HypotheticalIndex*);

– na linha 7 - *tuning:DML_TPCH_5* e *tuning:DML_TPCH_8* são os nomes das consultas beneficiadas que deram origem (*tuning:originatesIndSpec*) e são instâncias de *tuning:QueryStatement*;

– na linha 8 - *tuning:HypotheticalIndex* identifica a qual conceito o indivíduo (índice hipotético) pertence (*rdf:type tuning:HypotheticalIndex*).

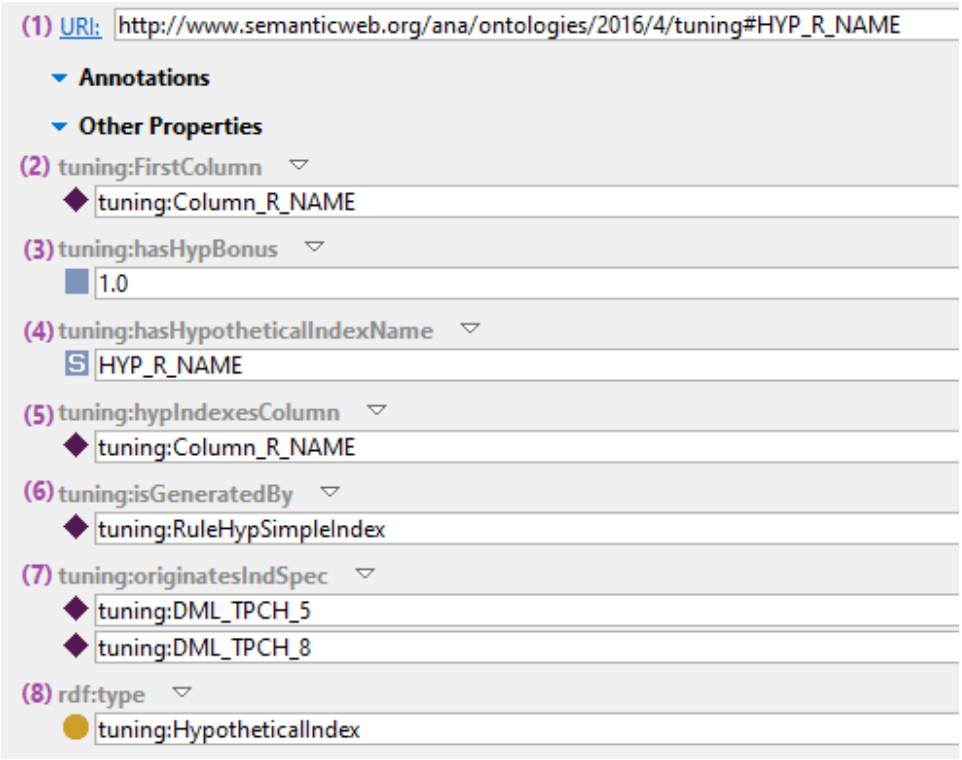

Figura 4.6: Inferência de RuleHypSimpleIndex

A Figura 4.7 mostra um índice composto hipotético denominado *HYP_L_SUPPKEY_L_PARTKEY*, identificado por seu URI (linha 1), inferido através da regra *tuning:RuleHypCompositeIndex*, sendo a descrição de suas propriedades:

– nas linhas 2 e 3, a descrição da primeira (*tuning:FirstColumn*) e da segunda coluna (*tuning:SecondColumn*) utilizada na geração do

PUC-Rio - Certificação Digital Nº 1912717/CA

índice composto, ou seja, por *tuning:Column_L_SUPPKEY* e *tuning:Column_L_PARTKEY*, respectivamente;

- na linha 4 - 0.2 é o valor do *bonus* (*tuning:hasHypBonus*);
- na linha 5 - *HYP_L_SUPPKEY_L_PARTKEY* é o nome do índice hipotético composto (*tuning:hasHypotheticalIndexName*);
- na linha 6 - *tuning:Column_L_PARTKEY* e *tuning:Column_L_SUPPKEY* são as colunas indexadas (*tuning:HypIndexesColumn*);
- na linha 7 - *tuning: RuleHypCompositeIndex* indica a regra de *tuning* (*tuning:isGeneratedBy*) que gerou este índice hipotético;
- na linha 8 - *tuning:DML_TPCH_8* (*tuning:originatesIndSpec*) é a consulta que deu origem ao índice;
- na linha 9 - *tuning:HypotheticalIndex* identifica a qual conceito o indivíduo pertence (*rdf:type tuning:HypotheticalIndex*).

A ontologia inferiu apenas um índice hipotético composto para este par de colunas (*tuning:Column_L_SUPPKEY* e *tuning:Column_L_PARTKEY*), dado que já existe um índice composto (*rdf:type tuning:Index*) para o mesmo par de colunas em ordem inversa, conforme Figura 4.8. Nessa figura, podemos observar essas colunas na ordem *tuning:Column_L_PARTKEY* e *tuning:Column_L_SUPPKEY*, nas linhas 2 e 3 respectivamente, além do valor de sua utilidade (*tuning:hasIndexUsefulness*) na linha 5, que foi de 0.2.

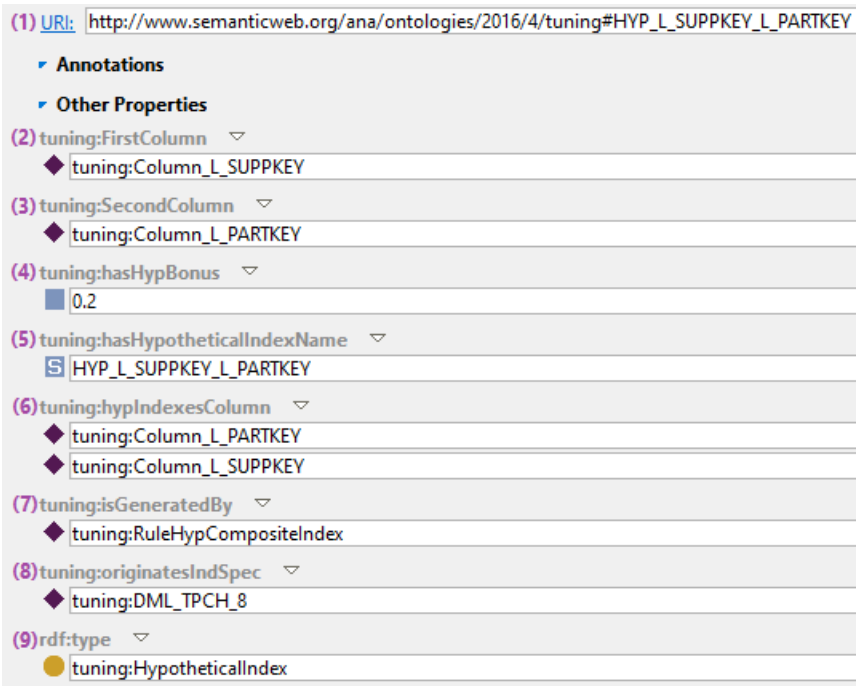

Figura 4.7: Inferência de RuleHypCompositeIndex

PUC-Rio - Certificação Digital Nº 1912717/CA

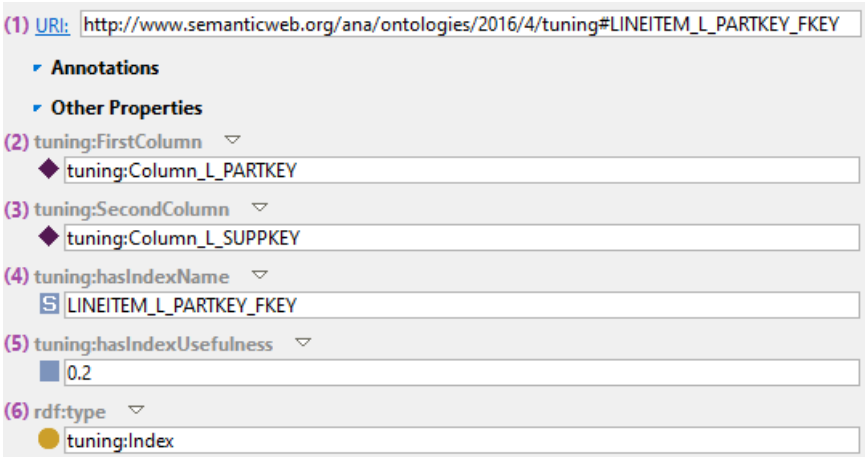

Figura 4.8: Índice LINEITEM_L_PARTKEY_FKEY

Após a execução da primeira carga de trabalho, as novas triplas inferidas foram persistidas na OnDBTuning, e uma segunda carga de trabalho composta pela consulta Q3 do TPC-H foi instanciada na OnDBTuning. Em seguida, as regras foram executadas para essa nova carga de trabalho. Com isso, permitiu-se verificar que não ocorreram novas instâncias de índices simples e compostos na segunda carga, pois não houve alteração das colunas referenciadas na cláusula WHERE em comparação com a primeira carga. O que a nova consulta poderia sugerir já foi contemplado na carga de trabalho anterior acumulada.

4.5 Comparativo do Bônus e Utilidade das Cargas de Trabalho

No gráfico da Figura 4.9, um comparativo dos valores dos bônus da primeira e segunda carga de trabalho é mostrado. No eixo vertical, a representação dos valores do bônus, no eixo horizontal, o nome dos 40 índices hipotéticos inferidos, sendo as barras de cor vermelha, a primeira carga, (*Workload*1) e as barras de cor azul, a segunda carga de trabalho (*Workload*2). Com isso, podemos observar que o valor do bônus (*tuning:hasHypBonus*) para alguns índices hipotéticos diminuíram, aumentaram ou mantiveram-se inalterados após a execução da segunda carga de trabalho. Por exemplo, para a instância *tuning: HYP_L_SUPPKEY*, o valor do bônus anteriormente era 0,4 e diminuiu para 0,3333333. Esse resultado mostrou que houve uma diminuição no número de consultas beneficiadas por esse índice hipotético.

PUC-Rio - Certificação Digital Nº 1912717/CA

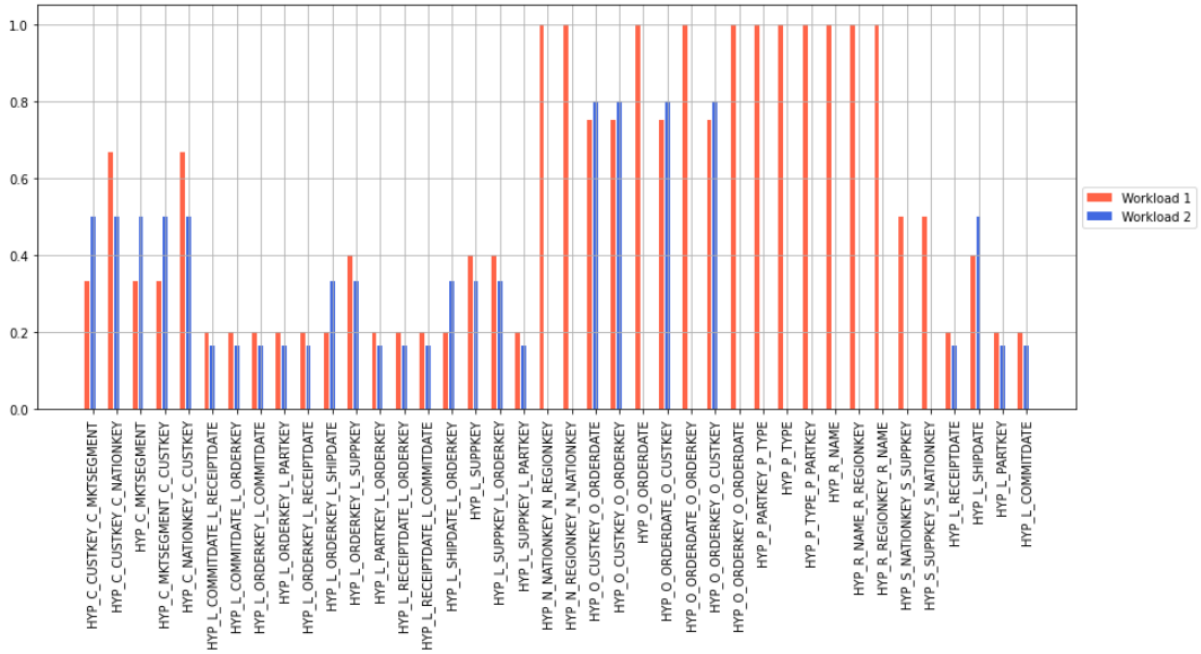

Figura 4.9: Bônus dos Índices Hipotéticos

No gráfico da Figura 4.10, vemos um comparativo dos valores de utilidade da primeira e segunda carga de trabalho. No eixo Y, os valores de utilidade, no eixo X o nome dos índices reais, sendo as barras de cor verde, a primeira carga (*Workload*1) e as barras de cor azul, a segunda carga de trabalho (*Workload*2). Com isso, podemos observar que os valores de utilidade (*tuning:hasIndexUsefulness*) dos índices diminuíram, aumentaram ou mantiveram-se inalterados, após a execução da segunda carga de trabalho. Como exemplo, podemos observar o índice *tuning:LINEITEM_L_PARTKEY_FKEY* que obteve o valor de 0,2 na primeira carga de trabalho e na segunda carga, o valor diminuiu para 0,16666667, evidenciando que tem baixo potencial para aumentar o desempenho das consultas da carga de trabalho.

PUC-Rio - Certificação Digital Nº 1912717/CA

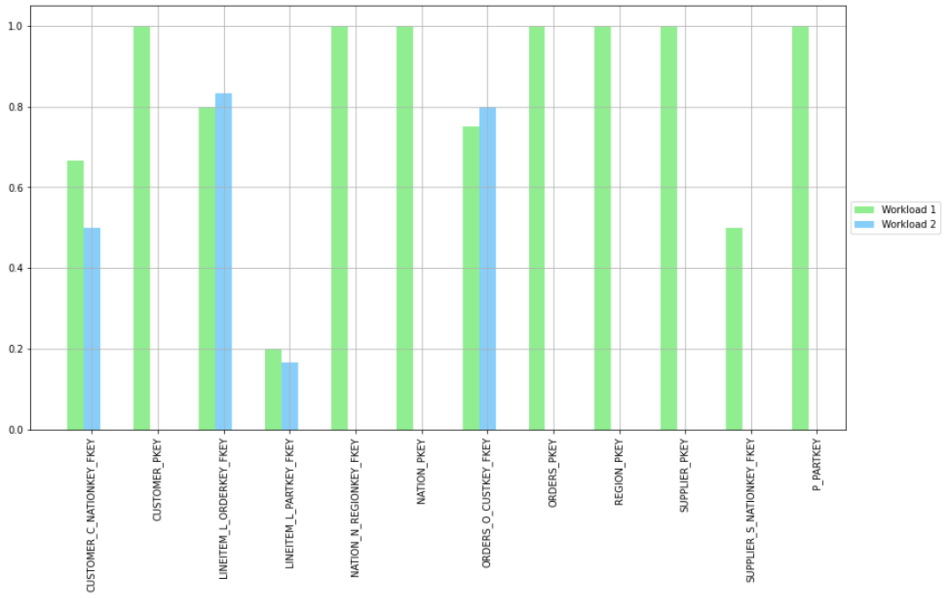

Figura 4.10: Utilidade dos Índices Reais

É fundamental mencionar que, na segunda carga de trabalho, novas triplas foram criadas apenas para aqueles índices que tiveram alteração do valor de utilidade (*tuning:hasIndexUsefulness*) e o mesmo comportamento para os índices hipotéticos com relação ao bônus (*tuning:hashasHypBonus*), devido à nova consulta. No SPIN, as regras são verificadas quando ocorrem novas instâncias.

4.6 Inferência de Índices Parciais

Foram instanciados 15 índices parciais após as duas cargas de trabalho executadas, conforme modelagem da regra na subseção 3.3.2 e regra na Figura 4.11.

A Figura 4.12 demonstra o resultado de uma inferência de índice parcial, sendo:

- na linha 2 - *tuning:Coluna_C_MKTSEGMENT* é a coluna referenciada (*tuning:FirstColumn*);
- na linha 3 - WHERE = 'BUILDING' é a expressão condicional do índice parcial (*tuning:hasAConditionalExpression*);
- na linha 4 - *PARTIAL_C_MKTSEGMENT_ec940310e2025d61bae1c6ef4d7b3738* é o nome do índice parcial (*tuning:hasIndexName*);
- na linha 5 - *tuning:RuleSimplePartialIndex* indica a regra de *tuning* (*tuning:isGeneratedBy*) que gerou este índice parcial (*rdf:type tuning:PartialIndex*);

PUC-Rio - Certificação Digital Nº 1912717/CA

– na linha 6 - *tuning:DML_TPCH_3* é a consulta que deu origem ao índice parcial (*tuning:originatesIndSpec*);

– na linha 7 - *tuning:PartialIndex* identifica a qual conceito pertence (*rdf:type tuning:PartialIndex*).

```
CONSTRUCT {
  ?uri a tuning:PartialIndex .
  ?uri tuning:hasAConditionalExpression ?compLiteral .
  ?uri tuning:hasIndexName ?nameIndex .
  ?uri tuning:FirstColumn ?c .
  ?uri tuning:isGeneratedBy tuning:RuleSimplePartialIndex .
  ?uri tuning:originatesIndSpec ?d .
}
WHERE {
  ?s a tuning:AtomicExpression .
  ?l tuning:componentOfAtomicExp ?s .
  ?c tuning:componentOfAtomicExp ?s .
  ?s tuning:hasPredicateDescription ?atomic .
  BIND (MD5(?atomic) AS ?suf) .
  ?s tuning:componentOfWhere ?where .
  ?where tuning:componentOfDMLSt ?d .
  FILTER regex(?atomic, UCASE(?literal)) .
  ?l tuning:hasLiteralDescription ?literal .
  FILTER regex(?atomic, UCASE(?columnName)) .
  ?c tuning:hasColumnName ?columnName .
  BIND (CONCAT("WHERE ", STRAFTER(?atomic, ?columnName)) AS ?compLiteral) .
  BIND (CONCAT("PARTIAL_", ?columnName, "_", ?suf) AS ?nameIndex) .
  BIND (URI(CONCAT("http://www.semanticweb.org/ana/ontologies/2016/4/tuning#", ?nameIndex)) AS ?uri) .
}
```

Figura 4.11: Regra do Índice Parcial

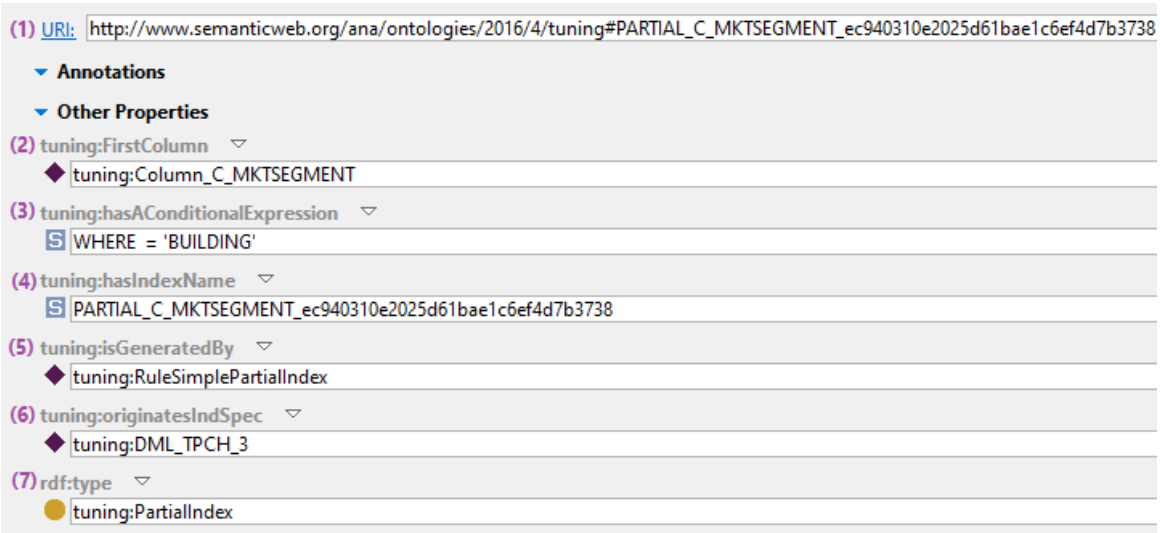

Figura 4.12: Índice Parcial Inferido

PUC-Rio - Certificação Digital Nº 1912717/CA

4.7 Inferência de Visões Materializadas

Foi inferida uma visão materializada hipotética para cada consulta (*tuning:DMLStatement*) da carga de trabalho informada, conforme modelagem da regra na subseção 3.3.3 e regra no Apêndice B. A Figura 4.13 mostra o resultado de uma inferência de visão materializada hipotética denominada HYP_VIEW_DML_TPCH_3, identificado por seu URI (linha 1), sendo:

- na linha 2 - HYP_VIEW_DML_TPCH_3 é o nome da visão materializada (*tuning:hasHypotheticalMaterializedViewName*);
- na linha 3 - a descrição da visão materializada (*tuning:hasViewDescription*);
- na linha 4 - *tuning:RuleHypViewAdapted* indica a regra de *tuning* (*tuning:isGeneratedBy*) que gerou esta visão materializada;
- na linha 5 - *tuning:DML_TPCH_3* (*tuning:originatesViewSpec*) é a consulta que deu origem a visão materializada;
- na linha 6 - *tuning:HypotheticalMaterializedView* identifica a qual conceito pertence (*rdf:type tuning:HypotheticalMaterializedView*).

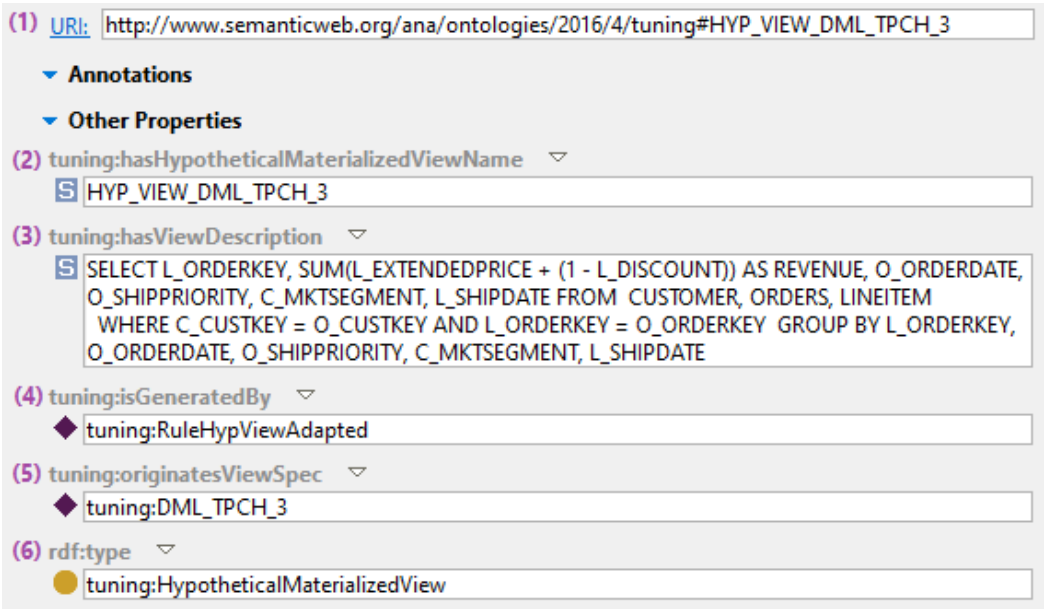

Figura 4.13: Visão Materializada Hipotética

PUC-Rio - Certificação Digital Nº 1912717/CA

4.8 Escopo do Estudo de Caso

Devido à limitação de tempo, para o escopo deste estudo de caso:

– não foi possível abranger todas as decomposições SQL nessa implementação, sendo contemplado somente os seguintes operadores (< , >, =, >= , <=);

– as regras utilizadas nessa pesquisa utilizaram os modelos da seção 3.3. Esses modelos consideraram a cláusula WHERE, porém novas heurísticas e regras, considerando outras cláusulas como o SELECT, podem ser modeladas e instanciadas na OnDBTuning;

– não foi considerada a estrutura hipotética para índices parciais, mas a inferência das expressões condicionais candidatas a índices parciais, para a carga de trabalho informada.

4.9 Conclusões

As inferências obtidas nesse capítulo foram sugestões de ações de sintonia fina através de uma ontologia, cabendo a tomada de decisão a um especialista em *tuning* de banco de dados ou ferramenta específica.

Esse trabalho não se limitou a uma carga de trabalho específica, podendo ser instanciada na ontologia outros *benchmarks* e também não foi de um determinado SGBD, pois as regras foram modeladas e implementadas de forma genérica SQL, podendo abranger outras cargas de trabalho.

PUC-Rio - Certificação Digital Nº 1912717/CA

5
Conclusão

Neste trabalho de pesquisa foi projetada a inferência de sugestões de *tuning* baseada na heurística de índices e visões materializadas. As quatro regras, a saber, *tuning:RuleHypSimpleIndex*, *tuning:RuleHypCompositeIndex*, *tuning:RulePartialIndex* e *tuning:RuleHypViewAdapted* foram adicionadas na ontologia OnDBTuning, e sugestões de índice hipotéticos, índices parciais e visões materializadas foram inferidas com a carga de trabalho do TPC-H. Essas sugestões apoiam um DBA em suas ações de sintonia fina com explicações lógicas que as ferramentas de caixas pretas não suportam. Parte deste trabalho de pesquisa foi publicado antes da defesa desta dissertação [36].

5.1
Contribuições da Pesquisa

A principal contribuição dessa pesquisa é propor uma implementação real de uma ferramenta declarativa de sintonia fina automática, com a utilização de uma ontologia que modela conceitos e heurísticas de ações de sintonia fina em conjunto com uma máquina de regras.

Portanto, foi necessária a adição de novos conceitos e propriedades na OnDBTuning, a identificação das partes que compõe o comando SQL que correspondem aos conceitos da Ontologia, a modelagem das regras de *tuning*, a elaboração e instanciação das regras na ontologia, instanciação de uma carga de trabalho composta pelos seus metadados e consultas na ontologia e execução das regras através de uma máquina de regras. Tudo isso foi feito para inferir sugestões de ações de sintonia fina potencialmente úteis na melhoria do desempenho de um banco de dados relacional, independentemente do SGBD.

A sugestão de índices hipotéticos e de visões materializadas são exemplos de ações de sintonia fina. Além disso, o processo de inferência também permitiu a criação de indicadores para auxiliar na tomada da decisão de sintonia fina, como o cálculo do bônus dos índices hipotéticos e da utilidade da estrutura de acesso.

PUC-Rio - Certificação Digital Nº 1912717/CA

5.2 Trabalhos Futuros

Por conta do tempo para a defesa dessa dissertação, não foi possível abranger as seguintes atividades:

1. novas regras com heurísticas sobre índices e visões materializadas;
2. regra com a inclusão de mais colunas nos índices hipotéticos;
3. regra com a verificação da existência da descrição de visão materializada, a fim de evitar a sugestão duplicada de uma visão materializada;

A extensão da inferência de ações de tuning na OnDBTuning, como trabalho futuro, poderia contemplar:

1. regras com heurísticas sobre estratégias de particionamento;
2. novos conceitos e propriedades relacionados a sintonia fina de banco de dados NoSQL;
3. *parsing* SQL abrangendo consultas aninhadas e outros operadores SQL;
4. uso do *Meta-Modeling Vocabulary*[1] nas regras SPIN;
5. descrição das regras em *Shapes Constraint Language* (SHACL)[2], sendo uma linguagem que se tornou uma recomendação W3C e possui termos equivalentes com a linguagem SPIN, a fim de verificar se possui características úteis e mais expressivas no processo de inferência de *tuning*;
6. integração da ontologia e uma máquina de regras com uma ferramenta que captura automaticamente a carga de trabalho de um SGBDR e execute as inferências de sugestões de sintonia fina para uma avaliação completa do desempenho do banco de dados.

[1]https://spinrdf.org/spin.html
[2]https://www.w3.org/TR/shacl/

PUC-Rio - Certificação Digital Nº 1912717/CA

Referências bibliográficas

[1] SHASHA, D.; BONNET, P.. **Database Tuning: Principles, Experiments and Troubleshooting Techniques**, volumen 33. Morgan Kaufmann, 1st edition, 01 2002.

[2] STAAB, S.; STUDER, R.. **Handbook on Ontologies**. Springer Sci & Bus. Media, 2010.

[3] DE ALMEIDA, A. C. B.. **Framework para apoiar a sintonia fina de banco de dados (in portuguese)**. PhD thesis, PUC-Rio, 2013.

[4] DE OLIVEIRA, R. P.; BAIÃO, F.; ALMEIDA, A. C.; SCHWABE, D. ; LIFSCHITZ, S.. **Outer-tuning: an integration of rules, ontology and rdbms**. In: PROCEEDINGS OF THE XV BRAZILIAN SYMPOSIUM ON INFORMATION SYSTEMS, p. 1–8, 2019.

[5] QUASS, D.; GUPTA, A.; MUMICK, I. S. ; WIDOM, J.. **Making views self-maintainable for data warehousing**. In: PROCEEDINGS OF THE FOURTH INTERNATIONAL CONFERENCE ON PARALLEL AND DISTRIBUTED INFORMATION SYSTEMS, DECEMBER 18-20, 1996, MIAMI BEACH, FLORIDA, USA, p. 158–169. IEEE Computer Society, 1996.

[6] ELMASRI, R.; NAVATHE, S. B.. **Sistemas de banco de dados**. Pearson Education do Brasil, 6th edition, 09 2011.

[7] SALLES, M. A. V.. **Criação Autônoma de Indices em Bancos de Dados**. PhD thesis, PUC–Rio, 2004.

[8] DATE, C. J.. **Introdução a sistemas de bancos de dados**. Ed. Campus, 8th edition, 2003.

[9] STONEBRAKER, M.. **The case for partial indexes**. ACM SIGMOD Record, 18(4):4–11, 1989.

[10] MORELLI, E. M. T.. **Recriação automática de índices em um SGDB relacional**. PhD thesis, PUC–Rio, 2006.

[11] CARVALHO, A. W.. **Criação Automática de Visões Materializadas em SGBDs Relacionais**. PhD thesis, PUC-Rio, 2011.

PUC-Rio - Certificação Digital Nº 1912717/CA

[12] GRUBER, T. R.. **A translation approach to portable ontology specifications**. Knowledge Acquisition, 5(2):199–220, 1993.

[13] DE AGUIAR, C. Z.; DE ALMEIDA FALBO, R. ; SOUZA, V. E. S.. **Ontological representation of relational databases**. In: Carbonera, J. L.; Guizzardi, G., editors, PROCEEDINGS OF THE XI SEMINAR ON ONTOLOGY RESEARCH IN BRAZIL AND II DOCTORAL AND MASTERS CONSORTIUM ON ONTOLOGIES, SÃO PAULO, BRAZIL, OCTOBER 1ST-3RD, 2018, volumen 2228 de **CEUR Workshop Proceedings**, p. 140–151. CEUR-WS.org, 2018.

[14] ISOTANI, S.; BITTENCOURT, I. I.. **Dados Abertos Conectados: Em busca da Web do Conhecimento**. Novatec Editora, 1st edition, 09 2015.

[15] FOLLENFANT, C.; CORBY, O.. **Sql abstract syntax trees vocabulary**. https://ns.inria.fr/ast/sql/index.html, 2014. Accessed: 2021-11-24.

[16] FOLLENFANT, C.; CORBY, O.; GANDON, F. ; TRASTOUR, D.. **Rdf modelling and sparql processing of sql abstract syntax trees**. In: PSW-1ST WORKSHOP ON PROGRAMMING THE SEMANTIC WEB, 2012.

[17] BASSILIADES, N.. **A tool for transforming semantic web rule language to SPARQL infererecing notation**. Intl. Journal Semantic Web Information Systems, p. 87–115, 2020.

[18] STEVE HARRIS, G.; SEABORNE, A.. **Sparql 1.1 query language**. https://www.w3.org/TR/sparql11-query/, 03 2013. Accessed: 2021-10-29.

[19] MEENA, K.; SIVAKUMAR, R.. **Human-computer interaction**. PHI Learning Pvt. Ltd., 01 2014.

[20] MONTEIRO, J.. **Uma abordagem não intrusiva para a manutenção automática do projeto físico de bancos de dados**. PhD thesis, Tese de Doutorado, Departamento de Informática, PUC-Rio, 2008.

[21] DE OLIVEIRA, R. P.. **Combinação e seleção automática de ações de sintonia fina**. PhD thesis, Tese de Doutorado, Departamento de Informática, PUC-Rio, 2019.

[22] DOULAVERAKIS, C.; KOUTKIAS, V.; ANTONIOU, G. ; KOMPATSIARIS, I.. **Applying sparql-based inference and ontologies for modelling and execution of clinical practice guidelines: a case study on**

PUC-Rio - Certificação Digital Nº 1912717/CA

hypertension management. In: KNOWLEDGE REPRESENTATION FOR HEALTH CARE, p. 90–107. Springer, 2016.

[23] PROMKOT, A.-N.; ARCH-INT, S. ; ARCH-INT, N.. **The personalized traditional medicine recommendation system using ontology and rule inference approach**. In: 4TH INTL. CONF. ON COMPUTER AND COMMUNICATION SYSTEMS, p. 96–104. IEEE, 2019.

[24] AKEN, D. V.; PAVLO, A.; GORDON, G. J. ; ZHANG, B.. **Automatic database management system tuning through large-scale machine learning**. In: Salihoglu, S.; Zhou, W.; Chirkova, R.; Yang, J. ; Suciu, D., editors, PROCEEDINGS OF THE 2017 ACM INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, SIGMOD CONFERENCE 2017, CHICAGO, IL, USA, MAY 14-19, 2017, p. 1009–1024. ACM, 2017.

[25] ZHANG, J.; ZHOU, K.; LI, G.; LIU, Y.; XIE, M.; CHENG, B. ; XING, J.. **Cdbtune+: An efficient deep reinforcement learning-based automatic cloud database tuning system**. VLDB, p. 1–29, 2021.

[26] DING, B.; DAS, S.; MARCUS, R.; WU, W.; CHAUDHURI, S. ; NARASAYYA, V.. **Ai meets ai: Leveraging query executions to improve index recommendations**. p. 1241–1258, 2019. cited By 31.

[27] TIWARY, M.; MISHRA, P.; JAIN, S. ; SAHOO, K.. **Autodbaas: Autonomous database as a service for managing backing services**. volumen 2021-March, p. 600–610, 2021. cited By 0.

[28] PEDROZO, W.; NIEVOLA, J. ; RIBEIRO, D.. **An adaptive approach for index tuning with learning classifier systems on hybrid storage environments**. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 10870 LNAI:716–729, 2018. cited By 9.

[29] LI, G.-L.; ZHOU, X.-H.. **Xuanyuan: An ai-native database systems**. Ruan Jian Xue Bao/Journal of Software, 31(3):831–844, 2020. cited By 2.

[30] XU, S.; ZHANG, Y.; BIAN, R.; CHENG, F. ; LI, X.. **Deepplaner: Query optimization using recurrent neural network**. Advances in Intelligent Systems and Computing, 1185:379–388, 2021. cited By 0.

[31] DAS, S.; GRBIC, M.; ILIC, I.; JOVANDIC, I.; JOVANOVIC, A.; NARASAYYA, V.; RADULOVIC, M.; STIKIC, M.; XU, G. ; CHAUDHURI, S.. **Automatically indexing millions of databases in microsoft azure sql database**. p. 666–679, 2019. cited By 22.

PUC-Rio - Certificação Digital Nº 1912717/CA

[32] GUIZZARDI, G.; FALBO, R. ; GUIZZARDI, R.. **A importância de ontologias de fundamentação para a engenharia de ontologias de domínio: o caso do domínio de processos de software**. IEEE Latin America Transactions, 6(3):244–251, 2008.

[33] ALMEIDA, A. C.; CAMPOS, M. L. M.; BAIÃO, F. A.; LIFSCHITZ, S.; DE OLIVEIRA, R. P. ; SCHWABE, D.. **An ontological perspective for database tuning heuristics**. In: INT. CONF. ON CONCEPTUAL MODELING (ER), p. 240–254. Springer, 2019.

[34] DEITEL, H. M.; DEITEL, P. J.. **Java: como programar**. Pearson, 6th edition, 2005.

[35] SUGANYA, G.; PORKODI, R.. **Ontology based information extraction-a review**. In: 2018 IEEE CONF. ON CURRENT TRENDS TOWARDS CONVERGING TECH. (ICCTCT), p. 1–7, 2018.

[36] PERCILIANO, L. D. S. S.; DOS SANTOS, V.; BAIÃO, F.; HAEUSLER, E. H.; LIFSCHITZ, S. ; ALMEIDA, A. C.. **Inferencing relational database tuning actions with ondbtuning ontology**. In: ANAIS DO XXXVI SIMPÓSIO BRASILEIRO DE BANCOS DE DADOS, p. 157–168. SBC, 2021.

PUC-Rio - Certificação Digital Nº 1912717/CA

A
Consultas TPC-H

As próximas seções deste apêndice exibem as consultas do TPC-H na sua forma original e a sua instância e reescrita utilizada na ontologia.

A.1
Consulta TPC-H Q1

Consulta Original

```
SELECT
        L_LINESTATUS,
        SUM(L_QUANTITY) AS SUM_QTY,
        SUM(L_EXTENDEDPRICE) AS SUM_BASE_PRICE,
        SUM(L_EXTENDEDPRICE*(1-L_DISCOUNT)) AS SUM_DISC_PRICE,
        SUM(L_EXTENDEDPRICE*(1-L_DISCOUNT)*(1+L_TAX)) AS SUM_CHARGE,
        AVG(L_QUANTITY) AS AVG_QTY,
        L_RETURNFLAG,
        AVG(L_EXTENDEDPRICE) AS AVG_PRICE,
        AVG(L_DISCOUNT) AS AVG_DISC,
        COUNT(*) AS COUNT_ORDER
FROM    LINEITEM
WHERE   L_SHIPDATE <= DATE '1998-12-01' - INTERVAL '[DELTA]' DAY (3)
GROUP BY L_RETURNFLAG, L_LINESTATUS
ORDER BY L_RETURNFLAG, L_LINESTATUS;
```

Consulta Instanciada

```
SELECT
        L_RETURNFLAG,
        L_LINESTATUS,
        SUM(L_QUANTITY) AS SUM_QTY,
        SUM(L_EXTENDEDPRICE) AS SUM_BASE_PRICE,
        SUM(L_EXTENDEDPRICE * (1 - L_DISCOUNT)) AS SUM_DISCPRICE,
        SUM(L_EXTENDEDPRICE * (1 - L_DISCOUNT) * (1 + L_TAX)) AS SUM_CHARGE,
        AVG(L_QUANTITY) AS AVG_QTY,
        AVG(L_EXTENDEDPRICE) AS AVG_PRICE,
        AVG(L_DISCOUNT) AS AVG_DISC,
        COUNT(*) AS COUNT_ORDER
FROM    LINEITEM
WHERE   L_SHIPDATE <= DATE'1998-12-01'-INTERVAL'87_DAYS'
GROUP BY L_RETURNFLAG, L_LINESTATUS
ORDER BY L_RETURNFLAG, L_LINESTATUS;
```

Figura A.1: Consulta Original e Instanciada - TPC-H Q1

PUC-Rio - Certificação Digital Nº 1912717/CA

A.2
Consulta TPC-H Q3 - 1

Consulta Original		Consulta Instanciada	
SELECT		**SELECT**	
	L_ORDERKEY,		L_ORDERKEY,
	SUM(L_EXTENDEDPRICE*(1-L_DISCOUNT)) **AS** REVENUE,		**SUM**(L_EXTENDEDPRICE + (1 - L_DISCOUNT)) **AS** REVENUE,
	O_ORDERDATE,		O_ORDERDATE,
	O_SHIPPRIORITY		O_SHIPPRIORITY
FROM	CUSTOMER, ORDERS, LINEITEM	**FROM**	CUSTOMER, ORDERS, LINEITEM
WHERE	C_MKTSEGMENT = '[SEGMENT]'	**WHERE**	C_MKTSEGMENT = 'BUILDING'
AND	C_CUSTKEY = O_CUSTKEY	**AND**	C_CUSTKEY = O_CUSTKEY
AND	L_ORDERKEY = O_ORDERKEY	**AND**	L_ORDERKEY = O_ORDERKEY
AND	O_ORDERDATE < DATE '[DATE]'	**AND**	O_ORDERDATE < DATE '1995-03-15'
AND	L_SHIPDATE > DATE '[DATE]'	**AND**	L_SHIPDATE > DATE '1995-03-15'
GROUP BY	L_ORDERKEY, O_ORDERDATE, O_SHIPPRIORITY	**GROUP BY**	L_ORDERKEY, O_ORDERDATE, O_SHIPPRIORITY
ORDER BY	REVENUE DESC, O_ORDERDATE;	**ORDER BY**	REVENUE DESC, O_ORDERDATE;

Figura A.2: Consulta Original e Instanciada - TPC-H Q3

PUC-Rio - Certificação Digital Nº 1912717/CA

A.3
Consulta TPC-H Q3 - 2

Consulta Original		Consulta Instanciada	
SELECT		**SELECT**	
	L_ORDERKEY,		L_ORDERKEY,
	SUM(L_EXTENDEDPRICE*(1-L_DISCOUNT)) **AS** REVENUE,		**SUM**(L_EXTENDEDPRICE + (1 - L_DISCOUNT)) **AS** REVENUE,
	O_ORDERDATE,		O_ORDERDATE,
	O_SHIPPRIORITY		O_SHIPPRIORITY
FROM	CUSTOMER, ORDERS, LINEITEM	**FROM**	CUSTOMER, ORDERS, LINEITEM
WHERE	C_MKTSEGMENT = '[SEGMENT]'	**WHERE**	C_MKTSEGMENT = 'AUTOMOBILE'
AND	C_CUSTKEY = O_CUSTKEY	**AND**	C_CUSTKEY = O_CUSTKEY
AND	L_ORDERKEY = O_ORDERKEY	**AND**	L_ORDERKEY = O_ORDERKEY
AND	O_ORDERDATE < DATE '[DATE]'	**AND**	O_ORDERDATE < DATE '1995-03-30'
AND	L_SHIPDATE > DATE '[DATE]'	**AND**	L_SHIPDATE > DATE '1995-03-30'
GROUP BY	L_ORDERKEY, O_ORDERDATE, O_SHIPPRIORITY	**GROUP BY**	L_ORDERKEY, O_ORDERDATE, O_SHIPPRIORITY
ORDER BY	REVENUE DESC, O_ORDERDATE;	**ORDER BY**	REVENUE DESC, O_ORDERDATE;

Figura A.3: Consulta Original e Instanciada - TPC-H Q3

A.4
Consulta TPC-H Q4

Consulta Original

```
SELECT
        O_ORDERPRIORITY,
        COUNT(*) AS ORDER_COUNT
FROM    ORDERS
WHERE   O_ORDERDATE >= DATE '[DATE]'
AND     O_ORDERDATE < DATE '[DATE]' + INTERVAL '3' MONTH
AND     EXISTS (
        SELECT *
        FROM LINEITEM
        WHERE L_ORDERKEY = O_ORDERKEY
        AND L_COMMITDATE < L_RECEIPTDATE
)
GROUP BY O_ORDERPRIORITY
ORDER BY O_ORDERPRIORITY;
```

Consulta Instanciada

```
SELECT
        O_ORDERPRIORITY
        ,COUNT(*) AS ORDER_COUNT
FROM    ORDERS, LINEITEM
WHERE   O_ORDERDATE >= DATE '1993-07-01'
AND     O_ORDERDATE < DATE '1993-07-01' + INTERVAL '3' MONTH
AND     L_ORDERKEY = O_ORDERKEY
AND     L_COMMITDATE < L_RECEIPTDATE
GROUP BY O_ORDERPRIORITY
ORDER BY O_ORDERPRIORITY;
```

Figura A.4: Consulta Original e Instanciada / Reescrita - TPC-H Q4

A.5
Consulta TPC-H Q5

Consulta Original

```
SELECT
        N_NAME,
        SUM(L_EXTENDEDPRICE * (1 - L_DISCOUNT)) AS REVENUE
FROM    CUSTOMER, ORDERS, LINEITEM, SUPPLIER, NATION, REGION
WHERE   C_CUSTKEY = O_CUSTKEY
AND     L_ORDERKEY = O_ORDERKEY
AND     L_SUPPKEY = S_SUPPKEY
AND     C_NATIONKEY = S_NATIONKEY
AND     S_NATIONKEY = N_NATIONKEY
AND     N_REGIONKEY = R_REGIONKEY
AND     R_NAME = '[REGION]'
AND     O_ORDERDATE >= DATE '[DATE]'
AND     O_ORDERDATE < DATE '[DATE]' + INTERVAL '1' YEAR
GROUP BY N_NAME
ORDER BY REVENUE DESC;
```

Consulta Instanciada

```
SELECT
        N_NAME,
        SUM(L_EXTENDEDPRICE * (1 - L_DISCOUNT)) AS REVENUE
FROM    CUSTOMER, ORDERS, LINEITEM, SUPPLIER, NATION, REGION
WHERE   C_CUSTKEY = O_CUSTKEY
AND     L_ORDERKEY = O_ORDERKEY
AND     L_SUPPKEY = S_SUPPKEY
AND     C_NATIONKEY = S_NATIONKEY
AND     S_NATIONKEY = N_NATIONKEY
AND     N_REGIONKEY = R_REGIONKEY
AND     R_NAME = 'ASIA'
AND     O_ORDERDATE >= DATE '1994-01-01'
AND     O_ORDERDATE < DATE '1994-01-01' + INTERVAL '1' YEAR
GROUP BY N_NAME
ORDER BY REVENUE DESC;
```

Figura A.5: Consulta Original e Instanciada - TPC-H Q5

PUC-Rio - Certificação Digital Nº 1912717/CA

A.6
Consulta TPC-H Q8

Consulta Original

```
SELECT
        O_YEAR,
        SUM(CASE WHEN NATION = '[NATION]'
           THEN VOLUME
           ELSE 0
        END) / SUM(VOLUME) AS MKT_SHARE
FROM    (SELECT
            EXTRACT(YEAR FROM O_ORDERDATE) AS O_YEAR,
            L_EXTENDEDPRICE * (1-L_DISCOUNT) AS VOLUME,
            N2.N_NAME AS NATION
        FROM PART, SUPPLIER, LINEITEM, ORDERS, CUSTOMER, NATION N1, NATION N2, REGION
        WHERE P_PARTKEY = L_PARTKEY
        AND S_SUPPKEY = L_SUPPKEY
        AND L_ORDERKEY = O_ORDERKEY
        AND O_CUSTKEY = C_CUSTKEY
        AND C_NATIONKEY = N1.N_NATIONKEY
        AND N1.N_REGIONKEY = R_REGIONKEY
        AND R_NAME = '[REGION]'
        AND S_NATIONKEY = N2.N_NATIONKEY
        AND O_ORDERDATE BETWEEN DATE '1995-01-01' AND DATE '1996-12-31'
        AND P_TYPE = '[TYPE]'
) AS    ALL_NATIONS
GROUP BY O_YEAR
ORDER BY O_YEAR;
```

Consulta Instanciada

```
SELECT
        SUM(CASE WHEN N_NAME = 'BRAZIL'
        THEN L_EXTENDEDPRICE * (1 - L_DISCOUNT)
        ELSE 0
        END) / SUM((L_EXTENDEDPRICE * (1 - L_DISCOUNT))) AS MKT_SHARE,
        DATE_PART('YEAR', O_ORDERDATE) AS O_YEAR
FROM    PART, SUPPLIER, LINEITEM, ORDERS, CUSTOMER, NATION, REGION
WHERE   P_PARTKEY = L_PARTKEY
AND     S_SUPPKEY = L_SUPPKEY
AND     L_ORDERKEY = O_ORDERKEY
AND     O_CUSTKEY = C_CUSTKEY
AND     C_NATIONKEY = N_NATIONKEY
AND     N_REGIONKEY = R_REGIONKEY
AND     R_NAME = 'AMERICA'
AND     O_ORDERDATE
BETWEEN DATE '1995-01-01' AND DATE '1996-12-31'
AND     P_TYPE = 'ECONOMY ANODIZED STEEL'
GROUP BY O_YEAR
ORDER BY O_YEAR;
```

Figura A.6: Consulta Original e Instanciada / Reescrita - TPC-H Q8

PUC-Rio - Certificação Digital Nº 1912717/CA

B
Regra da Visão Materializada

As Figuras B.1, B.2 e B.3 na sequência, descrevem a regra denominada *RuleHypViewAdapted* que foi utilizada para gerar a sugestão da visão materializada hipotética.

```
CONSTRUCT {
   ?uri a tuning:HypotheticalMaterializedView .
   ?uri tuning:hasHypotheticalMaterializedViewName ?hypViewName .
   ?uri tuning:hasViewDescription ?view .
   ?uri tuning:originatesViewSpec ?s .
   ?uri tuning:isGeneratedBy tuning:RuleHypViewAdapted .
}
WHERE {
   ?s tuning:hasDMLDescription ?dml .
   ?f a tuning:FromClause .
   ?f tuning:hasClauseDescription ?from .
   ?f tuning:componentOfDMLSt ?s .
   {
      SELECT (GROUP_CONCAT(DISTINCT COALESCE(?columnName, ""); SEPARATOR=', ') AS ?select) ?s
      WHERE {
         {
            {
               SELECT DISTINCT ?s ?columnName
               WHERE {
                  ?sl a tuning:SelectClause .
                  ?sl tuning:componentOfDMLSt ?s .
                  ?sl tuning:hasClauseDescription ?desSelect .
                  BIND (REPLACE(UCASE(spif:trim(?desSelect)), "SELECT", "") AS ?sele) .
                  ?selec spif:split ( ?sele "," ) .
                  BIND (spif:trim(?selec) AS ?columnName) .
               }
            } .
         }
```

Figura B.1: Regra *RuleHypViewAdapted* (parte 1)

PUC-Rio - Certificação Digital Nº 1912717/CA

```
        UNION
        {
          {
            SELECT DISTINCT ?s ?columnName

            WHERE {
              ?w a tuning:WhereClause .
              ?w tuning:componentOfDMLSt ?s .
              OPTIONAL {
                ?a a tuning:AtomicExpression .
                ?a tuning:componentOfWhere ?w .
                ?a tuning:hasPredicateDescription ?atomic .
                FILTER regex(?atomic, UCASE(?literal)) .
                ?l tuning:hasLiteralDescription ?literal .
                FILTER regex(?atomic, UCASE(?columnName)) .
                ?col tuning:hasColumnName ?columnName .
              } .
            }
          } .
        }
UNION
        {
          {
            SELECT DISTINCT ?s ?columnName
            WHERE {
              ?h a tuning:HavingClause .
              ?h tuning:componentOfDMLSt ?s .
              ?h tuning:clauseReferencesColumn ?col .
              ?col tuning:hasColumnName ?columnName .
            }
          } .
        } .
      }
      GROUP BY ?s
    } .
    {
      SELECT (GROUP_CONCAT(DISTINCT COALESCE(?columnName, ""); SEPARATOR=', ') AS ?group) ?s
      WHERE {
        {
          {
            SELECT DISTINCT ?s ?columnName
            WHERE {
              ?sl a tuning:SelectClause .
              ?sl tuning:componentOfDMLSt ?s .
              ?sl tuning:hasClauseDescription ?desSelect .
              BIND (REPLACE(UCASE(spif:trim(?desSelect)), "SELECT", "") AS ?sele) .
              ?selec spif:split ( ?sele "," ) .
              BIND (spif:trim(?selec) AS ?columnName) .
              ?k tuning:hasColumnName ?columnName .
            }
          } .
        }
```

Figura B.2: Regra *RuleHypViewAdapted* (parte 2)

PUC-Rio - Certificação Digital Nº 1912717/CA

```
        UNION
        {
          {
            SELECT DISTINCT ?s ?columnName
            WHERE {
              ?w a tuning:WhereClause .
              ?w tuning:componentOfDMLSt ?s .
              OPTIONAL {
                ?a a tuning:AtomicExpression .
                ?a tuning:componentOfWhere ?w .
                ?a tuning:hasPredicateDescription ?atomic .
                FILTER regex(?atomic, UCASE(?literal)) .
                ?l tuning:hasLiteralDescription ?literal .
                FILTER regex(?atomic, UCASE(?columnName)) .
                ?col tuning:hasColumnName ?columnName .
              } .
            }
          } .
        }
         UNION
        {
          {
            SELECT DISTINCT ?s ?columnName
            WHERE {
              ?h a tuning:HavingClause .
              ?h tuning:componentOfDMLSt ?s .
              ?h tuning:clauseReferencesColumn ?col .
              ?col tuning:hasColumnName ?columnName .
            }
          } .
        } .
      }
      GROUP BY ?s
    } .
    OPTIONAL {
      {
        SELECT (GROUP_CONCAT(DISTINCT ?atomic; SEPARATOR=' AND ') AS ?where) ?s
        WHERE {
          ?w a tuning:WhereClause .
          ?w tuning:componentOfDMLSt ?s .
          ?a a tuning:AtomicExpression .
          ?a tuning:componentOfWhere ?w .
          ?a tuning:hasPredicateDescription ?atomic .
          FILTER NOT EXISTS {
            FILTER regex(?atomic, UCASE(?literal)) .
            ?l tuning:hasLiteralDescription ?literal .
          } .
          FILTER regex(?atomic, UCASE(?column)) .
          ?c tuning:hasColumnName ?column .
        }
        GROUP BY ?s
      } .
    } .
    BIND (spif:localName(?s) AS ?name) .
    BIND (CONCAT("HYP_VIEW_", ?name) AS ?hypViewName) .
    BIND (IF(COALESCE(?where, ""), CONCAT(" WHERE ", ?where), " ") AS ?w) .
    BIND (COALESCE(?select, "") AS ?newSelect) .
    BIND (COALESCE(?group, "") AS ?gr) .
    BIND (CONCAT("SELECT ", ?newSelect, " ", ?from, " ", ?w, " ", " GROUP BY ", ?gr) AS ?view) .
    BIND (URI(CONCAT("http://www.semanticweb.org/ana/ontologies/2016/4/tuning#", ?hypViewName)) AS ?uri) .
}
```

Figura B.3: Regra *RuleHypViewAdapted* (parte 3)

PUC-Rio - Certificação Digital Nº 1912717/CA