Outer-Tuning: sintonia fina automática baseada em ontologia

Ana Carolina Almeida[1], Edward Hermann Haeusler[2], Sérgio Lifschitz[2], Rafael Pereira de Oliveira[2], Daniel Schwabe[2]

[1] Depto de Informática e Ciência da Computação UERJ
ana.almeida@ime.uerj.br

[2]Departamento de Informática PUC-Rio
{hermann,sergio,rpoliveira,dschwabe}@inf.puc-rio.br

Resumo. *Apresentamos neste trabalho o framework Outer-Tuning, que visa dar apoio à sintonia fina (semi) automática de sistemas de bancos de dados relacionais por meio de uma ontologia específica ao domínio. Neste trabalho apresentamos os principais aspectos de sua arquitetura baseada em componentes, a máquina de regras de inferência e uma visão geral da ferramenta na prática.*

1. Introdução

O trabalho de administração e sintonia fina (*tuning*) de bancos de dados é complexo e exige especialização e conhecimentos fundamentais nesta área da computação. Busca-se um melhor desempenho para o banco de dados, isto é, maior eficiência ou vazão (*throughput*) de suas transações. Para isso, realizam-se ajustes de suas configurações, parâmetros e projeto físico, seleção de estruturas de acesso, redundância de estruturas físicas, sempre de acordo com a carga de trabalho executada no banco.

Nosso grupo de pesquisa já desenvolveu a ferramenta *DBX*[1] para a manutenção automática e contínua do projeto físico de um banco de dados relacional. Neste artigo, apresentamos a arquitetura de *software*, aspectos funcionais e práticos do *framework* Outer-Tuning [Almeida 2013], uma ferramenta baseada em ontologia criada para apoiar DBAs e desenvolvedores em geral na tomada de decisão envolvida na atividade de sintonia fina.

O restante deste artigo está organizado conforme descrito a seguir. Na Seção 2 motivamos a construção da ferramenta e o uso de ontologias no processo de sintonia fina. A Seção 3 apresenta o projeto de arquitetura e a Seção 4 ilustra o funcionamento básico do *framework* aqui proposto. A Seção 5 conclui este trabalho.

2. Motivação: ontologia e alternativas para sintonia fina

O trabalho de [Almeida 2013] propôs originalmente o *framework* Outer-Tuning, visando apoiar o trabalho de sintonia fina (semi)automática em sistemas de bancos de dados relacionais (SBDR). O nome da ferramenta originou-se do fato da mesma apresentar todas as alternativas analisadas pelas heurísticas e não somente aquela considerada a melhor ação de sintonia fina. Faz-se uma analogia à operação relacional de *outer join*, que retorna todas as tuplas das tabelas envolvidas e não somente aquelas que satisfazem à condição de junção. A ideia básica do Outer-Tuning consiste em oferecer mecanismos que permitam ao DBA uma tomada de decisão mais consistente com relação às possíveis ações de sintonia fina, baseada em múltiplas alternativas e respectivos custos.

[1]https://github.com/BioBD/dbx

Nas ferramentas de sintonia-fina existe uma falta de clareza sobre as decisões e as ações que são tomadas de forma automática. Dessa forma, o Outer-Tuning propõe o uso de uma ontologia de aplicação para a sintonia fina (automática ou não) que proporciona uma abordagem formal para decisões e inferências. A contribuição inovadora dessa abordagem é oferecer transparência e confiabilidade acerca das alternativas disponíveis para possíveis cenários no SGBD, por meio de justificativas concretas para as decisões que foram tomadas definidas semanticamente.

Foram encontrados apenas dois trabalhos preocupados com semântica [Goasdoué et al. 2011][Khouri et al. 2012], mas ambos somente relacionados aos dados e não aos conceitos de sintonia-fina. Além disso, as ferramentas não são flexíveis o suficiente para incorporar novas heurísticas. Através do uso de uma ontologia específica busca-se gerar automaticamente novas práticas de sintonia fina, a partir das práticas existentes (uso de inferências) ou de novas regras e conceitos que venham a surgir no futuro. Esta abordagem permite também combinações de heurísticas de sintonia fina.

A partir da ontologia do domínio de sintonia fina, proposta em [Almeida 2013] e estendida em [Oliveira 2015], buscou-se atender os desafios de especificar os componentes e integrá-los com uma máquina de regras. Usaremos Visões Materializadas (VM) para ilustrar as vantagens do uso da ontologia (Figura 1) em todo o processo. Supondo um usuário *User_1* que submete um comando DML *DML_1*. Este é classificado automaticamente como sendo *DMLCommand - SingleStatement - QueryStatement* através de regras definidas na própria ontologia. Esse comando possui como propriedade *hasDescription* o comando SQL derivado do benchmark TPC-H[2]: *SELECT no_o_id FROM new_order WHERE no_w_id = 1 AND no_d_id = 1;*. Continuando na figura, a ontologia já infere, por regras, as cláusulas *SELECT*, *FROM* e *WHERE* definidas em tal comando. A ontologia de domínio completa pode ser encontrada na URL[3].

Um exemplo de regra definida na ontologia pode ser visto na Figura 2. Para utilizar uma determinada heurística de VM é necessário estimar o custo de criação da visão materializada. Este é obtido, através de regra SWRL definido na ontologia, como sendo o custo de uma varredura simples da visão materializada mais o custo de gravação das páginas em memória secundária, estimada como sendo equivalente a 2 (duas) vezes a quantidade de páginas hipotéticas da VMH [Oliveira 2015]. Dessa forma, tem-se que dado o conceito de VM hipotética (VMH), se existe alguma VMH (linha 1 - Figura 2), e ela produz um plano (linha 2) do tipo *PlanoExecucaoReal* (linha 3); tal plano possui as propriedades de *temCustoExecucao* (linha 4) e *temNumeroPaginasHipoteticas* (linha 5) com valores já previamente calculados no metadado do banco e por outra regra, respectivamente. Em seguida, ocorre a multiplicação das páginas por dois (linha 6) e o custo de execução é então, adicionado a esse resultado (linha 7). Caso o valor total seja maior do que zero (linha 8), a propriedade *temValorCustoEstimadoCriacao* recebe esse valor (linha 9). Com a definição do cálculo pela regra usando conceitos do próprio domínio do DBA, acredita-se que melhora o entendimento das regras usadas na decisão de sintonia-fina da ferramenta bem como facilita a definição de novas regras e heurísticas.

[2]https://www.tpc.org/tpch/

[3]`http://www.inf.puc-rio.br/~postgresql/conteudo/projeto4/webvowl/index.html`

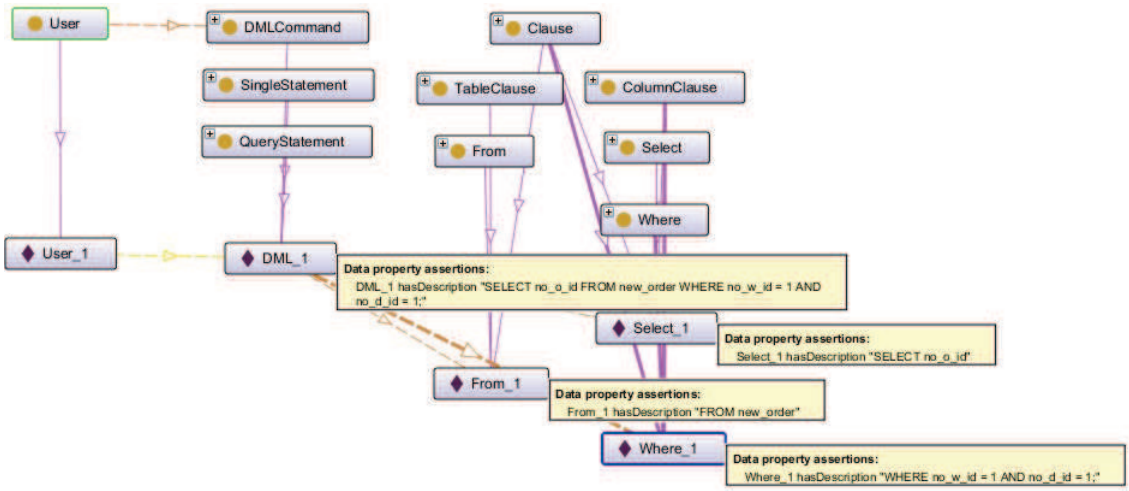

Figura 1. Fragmento da ontologia de domínio instanciada

```
1  VisaoMaterializadaHipotetica(?vmH) ∧
2     produz(?vmH, ?plan) ∧
3  PlanoExecucaoReal(?plan) ∧
4     temCustoExecucao(?plan, ?valorConsultar) ∧
5     temNumeroPaginasHipoteticas(?vmH, ?pagHipo) ∧
6     swrlb:multiply(?pagHipoMult, ?pagHipo, 2) ∧
7     swrlb:add(?total, ?pagHipoMult, ?valorConsultar) ∧
8     swrlb:greaterThan(?total, 0) →
9  temValorCustoEstimadoCriacao(?vmH, ?total)
```

Figura 2. Regra SWRL para cálculo do custo estimado de criação de uma VMH

3. Framework Outer-Tuning e arquitetura baseada em componentes

A arquitetura escolhida para o Outer-Tuning (Figura 3) é baseada em componentes. Devido ao caráter experimental da ferramenta e as múltiplas tecnologias envolvidas (e.g. SGBDs, máquinas de regras, ontologia, bibliotecas), decidiu-se que os componentes poderiam facilitar a comunicação entre as partes e deixar cada uma das fases de execução independentes. Assim, caso necessário, poderiam ser substituídos (ou mantidos) de forma compartimentalizada sem a propagação de *bugs*.

Como resultado da escolha de uma arquitetura baseada em componentes, decidiu-se que o Outer-Tuning seria desenvolvido como um *framework* de aplicação, que por definição é uma aplicação semicompleta, construída com uma coleção organizada de componentes de *software* reusáveis [Oliveira et al. 2011]. A escolha desse tipo de *framework* com uso de componentes, foi feita para possibilitar futura evolução para *software* orientado a serviços com baixo acoplamento entre as partes do *software*.

De acordo com o fluxo de execução proposto, o Outer-Tuning teve seus componentes definidos e especializados para cada etapa do ciclo. Na Figura 3, são enumerados os principais elementos da arquitetura proposta, descritos brevemente a seguir:

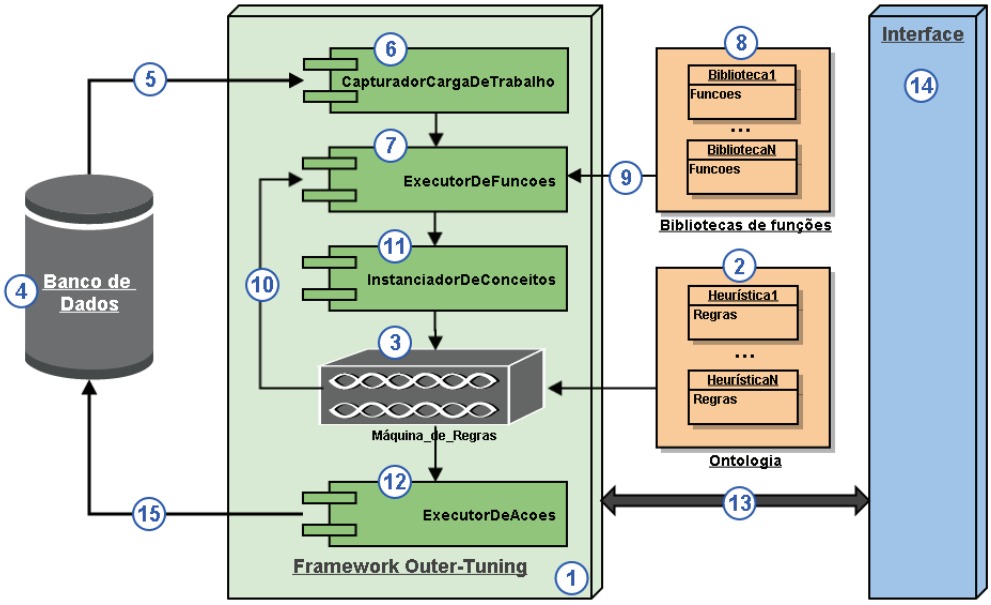

Figura 3. Arquitetura da Outertuning

(1) Base: Formada por uma biblioteca de funções para que os componentes possam compartilhar funções ordinárias e redundantes, além do acesso ao log compartilhado.

(2) Ontologia: A ontologia de aplicação contém a ontologia de domínio (conceitos instanciados pela carga de trabalho) e a ontologia de tarefas (heurísticas através de regras "se-então"). É a principal parte extensível do *framework*. As regras são definidas em uma linguagem declarativa (SWRL - Semantic Web Rules Language).

(3) Máquina de regras: Uma máquina de regras (ou motor de inferência) é o componente pelo qual as regras definidas na ontologia de tarefas são selecionadas e executadas. Para a implementação foi utilizada a máquina de regras Jess [4].

(4) Banco de dados: Qualquer banco de dados gerenciado por um SGBD possui sua comunicação com o *framework* (5) e (15) através dos *drivers* de conexão usados pelos componentes *CapturadorCargaDeTrabalho* (6) e *ExecutorDeFuncoes* (7).

(5) Carga de trabalho: A carga de trabalho capturada consiste em comandos SQL do tipo DML (*Data Manipulation Language* - Linguagem de manipulação de dados), os seus respectivos planos de execução e sua frequência.

(6) *CapturadorCargaDeTrabalho*: obtém a carga de trabalho com um intervalo de tempo pré-determinado pelo DBA para realizar a sintonia fina, através de driver JDBC [5].

(7) *ExecutorDeFuncoes*: Extrai informações da carga de trabalho e gera os indivíduos dos conceitos, que são pré-condições das heurísticas e que devem ser instanciados na máquina de regras para que haja inferência de ações de sintonia fina.

(8) Bibliotecas de funções: Sua função é aglutinar bibliotecas de código fonte compiladas, responsáveis por extrair conceitos da carga de trabalho. As bibliotecas são

[4]http://www.jessrules.com/ acesso em 25/05/2018
[5]http://www.oracle.com/ acesso em 25/05/2018

lidas e executadas em tempo de execução, sem intervenção no código fonte do *framework*.

(9) Comunicação entre bibliotecas e *ExecutorDeFuncoes*: realizada através de interface definida no *ExecutorDeFuncoes* que busca no repositório as funções desejadas.

(10) Conceitos pré-condições das heurísticas: O componente *ExecutorDeFuncoes* recebe da máquina de regras os conceitos que são pré-condições e a assinatura das funções contidas na biblioteca de funções que extraem conceitos.

(11) *InstanciadorDeConceitos*: Deve instanciar os indivíduos de pré-condições gerados pela execução das funções pelo *ExecutorDeFuncoes* na máquina de regras.

(12) *ExecutorDeAcoes*: Monitora a máquina de regras, captura as ações de sintonia fina inferidas e as executa no banco de dados de acordo com a escolha do DBA.

(13) Comunicação *framework* – interface: uso do padrão de projeto quadro-negro (*blackboard*)[Khosla et al. 2004], pela sua facilidade de implementação.

(14) Interface: Responsável pela interação com o DBA.

4. Outer-tuning na prática

Devido à limitação de espaço, vamos ilustrar brevemente o uso do Outer-Tuning. O vídeo em `http://www.inf.puc-rio.br/~postgresql/conteudo/projeto4/video/outertuning.mp4` mostra um possível uso da ferramenta.

Inicialmente, na Figura 4(A), o usuário do *framework* pode visualizar as heurísticas definidas na ontologia de tarefa e selecionar aquelas que deseja considerar para as futuras sugestões de sintonia fina. Posteriormente, o usuário informa para a ferramenta o modo que deseja trabalhar: semiautomático ou automático (sem intervenção humana). A ferramenta inicia a captura da carga de trabalho, em tempo real e apresenta, de forma gráfica, o momento em que o comando DML é executado no banco de dados e a sua duração, em segundos (Figura 4(B)). Caso o usuário queira detalhes maiores sobre a execução do comando, ele pode verificar mais abaixo na mesma tela (Figura 4(C)).

Caso o usuário queira acompanhar as ações de sintonia fina que estão sendo analisadas e sugeridas pela ferramenta, pode fazer isso através do menu *Tuning actions*. Nessa tela (Figura 4(D)), o usuário pode visualizar um gráfico com o cruzamento das informações de ganho esperado com a ação de sintonia fina (eixo x) e custo estimado de criação da estrutura de acesso (eixo y). O tamanho do círculo indicado no gráfico representa a quantidade de consultas SQL que a ação pode beneficiar. Quanto maior o tamanho do círculo, maior será o número de comandos beneficados pela ação de sintonia fina na carga de trabalho. O pop-up apresentado é um resumo sobre a ação de sintonia fina proposta com as seguintes informações: ganho esperado, custo de criação, tipo de ação (ex.: índice ou visão materializada) e número de comandos beneficiados pela ação.

As heurísticas que propõem VMs foram executadas e tiveram seus resultados avaliados e comparados através do Outer-Tuning. Para a geração da carga de trabalho durante os testes foi utilizado o *benchmark* TPC-H , propício para a avaliação de ferramentas de seleção de VMs por ser OLAP. Nota-se que a ferramenta apresenta tanto as avaliações positivas quanto as negativas. Algumas sugestões positivas foram implementadas e trouxeram benefícios conforme o esperado. Mais detalhes em [Oliveira 2015].

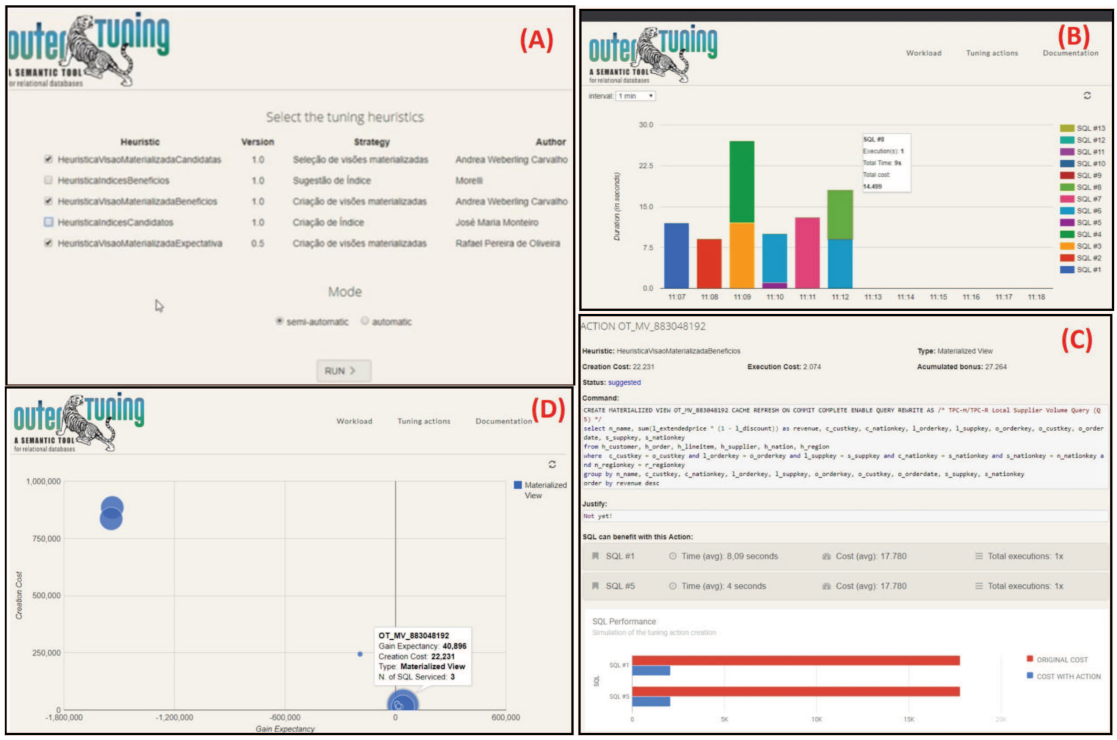

Figura 4. Amostra de Telas da Ferramenta Outer-Tuning

5. Comentários Finais

Apresentamos o *framework* Outer-Tuning para apoio à decisão sobre sintonia fina de bancos de dados relacionais. A ferramenta é apoiada por uma ontologia específica ao domínio que explicita conceitos e permite inferências. É possível visualizar as heurísticas instanciadas e o uso da sintonia fina de forma semiautomática.

References

[Almeida 2013] Almeida, A. C. B. d. (2013). *Framework para apoiar a sintonia fina de banco de dados*. PhD thesis, Depto. Informática - PUC-RIO.

[Goasdoué et al. 2011] Goasdoué, F., Karanasos, K., Leblay, J., and Manolescu, I. (2011). View selection in semantic web databases. *PVLDB*, 5(2):97–108.

[Khosla et al. 2004] Khosla, R., Ichalkaranje, N., and Jain, L. C. (2004). *Design of Intelligent Multi-Agent Systems: Human-Centredness, Architectures, Learning and Adaptation*. Studies in Fuzziness and Soft Computing. Springer.

[Khouri et al. 2012] Khouri, S., Bellatreche, L., Boukhari, I., and Bouarar, S. (2012). More investment in conceptual designers: Think about it! In *Computational Science and Engineering (CSE), 2012 IEEE 15th International Conference on*, pages 88–93. IEEE.

[Oliveira et al. 2011] Oliveira, J. L. D., Fernando, L., Loja, B., Larissa, S., Vicente, V., and Neto, G. (2011). Um Componente para Gerência de Processos de Negócio em Sistemas de Informação. *VII Simpósio Brasileiro de Sistemas de Informação*, pages 250–261.

[Oliveira 2015] Oliveira, R. P. d. (2015). Sintonia fina baseada em ontologias: o caso das visões materializadas. Master's thesis, Depto. Informática - PUC-RIO.