5
Trabalhos Relacionados

Existem ferramentas (por exemplo, (Morelli et al, 2012), (Bruno et al, 2011), (Rangaswamy et al, 2011), (Monteiro et al, 2012), (Elghandour, 2010) e (Goasdoué et al, 2011)) e abordagens (por exemplo, (Bellatreche et al, 2004) e (Bouchakri et al, 2011)), que auxiliam o DBA a selecionar as melhores estruturas e parâmetros de configurações para o SGBD. Essas ferramentas também auxiliam com outras técnicas, como a reescrita de consulta, a desnormalização e a *clusterização* entre outras.

As ferramentas e as abordagens consideradas por DBAs experientes podem sugerir a sintonia fina de acordo com a carga de trabalho (por exemplo, (Dias et al, 2005) (Alhadi et al, 2012) (Rodd et al, 2013) (Schnaitter et al, 2012) e (Jimenez et al, 2012)), podem ser orientadas a uma aplicação específica (por exemplo, (CAO et al, 2013) e (Zhang et al, 2012)) ou por comandos derivados de requisitos da aplicação que faz uso do banco de dados (Khoury et al, 2012).

Embora exista uma variedade de ferramentas semi-automáticas ou automáticas para apoiar o DBA na atividade de sintonia fina, não encontramos na literatura - ou em sistemas disponíveis publicamente - a preocupação de tornar todo o raciocínio utilizado na tomada de decisão explícito para os seus usuários. Essa transparência ajuda o DBA na decisão de confiar ou não em uma ferramenta e na argumentação para justificar a decisão final de sintonia fina.

Já existe uma preocupação, em relação aos sistemas de sintonia fina, em guardar o conhecimento sobre as razões e decisões tomadas por um administrador de dados, ou do próprio sistema de sintonia (Dias et al, 2005) (Alhadi et al, 2012). O registro do raciocínio através da instanciação de uma ontologia pode ser útil para fornecer futuras sugestões de ajustes ou demonstrar a transparência nas decisões internas do sistema.

Neste capítulo busca-se descrever os trabalhos relacionados, comparando-os, ao final, com o objetivo do *framework* proposto nesta tese. Por questões apenas de organização, decidiu-se dividir os trabalhos relacionados de acordo com

a modelagem implementada no SGBD, ressaltando as ferramentas desenvolvidas para SGBDs relacionais, que dominam o mercado atualmente. São elas: relacional, RDF (**R***esource* **D***escription* **F***ramework*), XML (EXtensible Markup Language) e *datawarehouse*.

5.1. Ferramentas para SGBDs relacionais

A ferramenta comercial *Automatic Database Diagnostic Monitor* (ADDM) do SGBD Oracle (Dias et al, 2005) realiza uma análise *top down* de um instante particular do SGBD. Ela identifica os sintomas de desempenho para, então, refiná-los de forma que descubra as reais causas dos problemas de desempenhos levantados. Seu objetivo é sugerir automaticamente uma forma de sintonia do SGBD, buscando diminuir o tempo perdido pelo servidor ao processar uma requisição do usuário, incluindo tempo de espera e CPU.

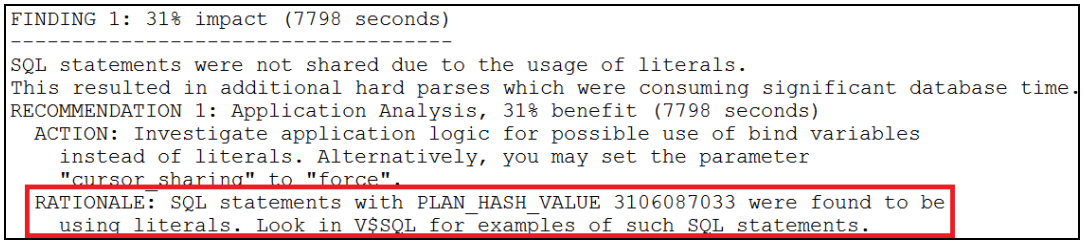

Figura 5-1 Exemplo de relatório gerado pela ferramenta ADDM

Na Figura 5-1 tem-se um exemplo de uma recomendação da ferramenta ADDM, indicando que, se a ação indicada for realizada, o ganho de tempo estimado será de 31%.

Uso de variável *bind*

```
variavelNota := 5;
SELECT matriculaAluno, nomeAluno
FROM aluno WHERE notaAluno = variavelNota;
```

Figura 5-2 Exemplo de uso de variável *bind*

A razão da sugestão é a análise das expressões em linguagem SQL que foram submetidas e consideradas pela ferramenta, para que suas condições sejam

trocadas para os tipos de variáveis *bind* (Figura 5-2) ao invés de literais (Figura 5-3).

Uso de literal

```
SELECT matriculaAluno, nomeAluno
FROM aluno WHERE notaAluno = 5;
```

Figura 5-3 Exemplo de uso de literal

É recomendável utilizar variáveis de ligação (*bind*) ao invés de valores fixos (literais) porque podem existir, por exemplo, diversas consultas iguais com alteração somente de seus valores. Com isso, substituindo tais valores fixos por referência a variáveis, e informando o valor desejado, a cada vez, como atributo da variável, o texto do SQL não muda. Portanto, não haverá verificação sintática (processo seguido para verificação e busca do valor solicitado pela consulta) repetitiva e, consequentemente, maior custo de CPU e tempo.

O mesmo plano de acesso será usado e também não haverá alocações de entradas novas no *cache*, visto que o mesmo texto será usado (variáveis de ligação) na consulta independente dos valores.

De acordo com o *rationale* do ADDM, consegue-se analisar as razões de uma determinada recomendação, fazendo com que o DBA possa ou não aceitá-la. No entanto, quando existe mais de uma recomendação e o DBA opta por uma delas, não se tem como deixar registradas as razões pelas quais se decide, por exemplo, por uma recomendação com benefício menor. Desta forma, o processo de tomada de decisão usado pelo DBA não fica registrado, para futuras sugestões da ferramenta ou para futuras análises por parte de outros DBAs. Além disso, também não ficam registrados os fatores contrários a uma determinada decisão, o que poderia, por exemplo, conduzir outros DBAs a optarem por uma recomendação de aparente menor benefício. O mesmo ocorre em outra ferramenta do mesmo SGBD mencionada em (Alhadi et al, 2012). Há preocupação em apresentar o raciocínio para as sugestões de sintonia fina (Figura 5-4) através da coluna *rationale*, explicando o motivo de cada recomendação, mas sem registro do motivo de aceitação, ou não, da recomendação, por parte do DBA.

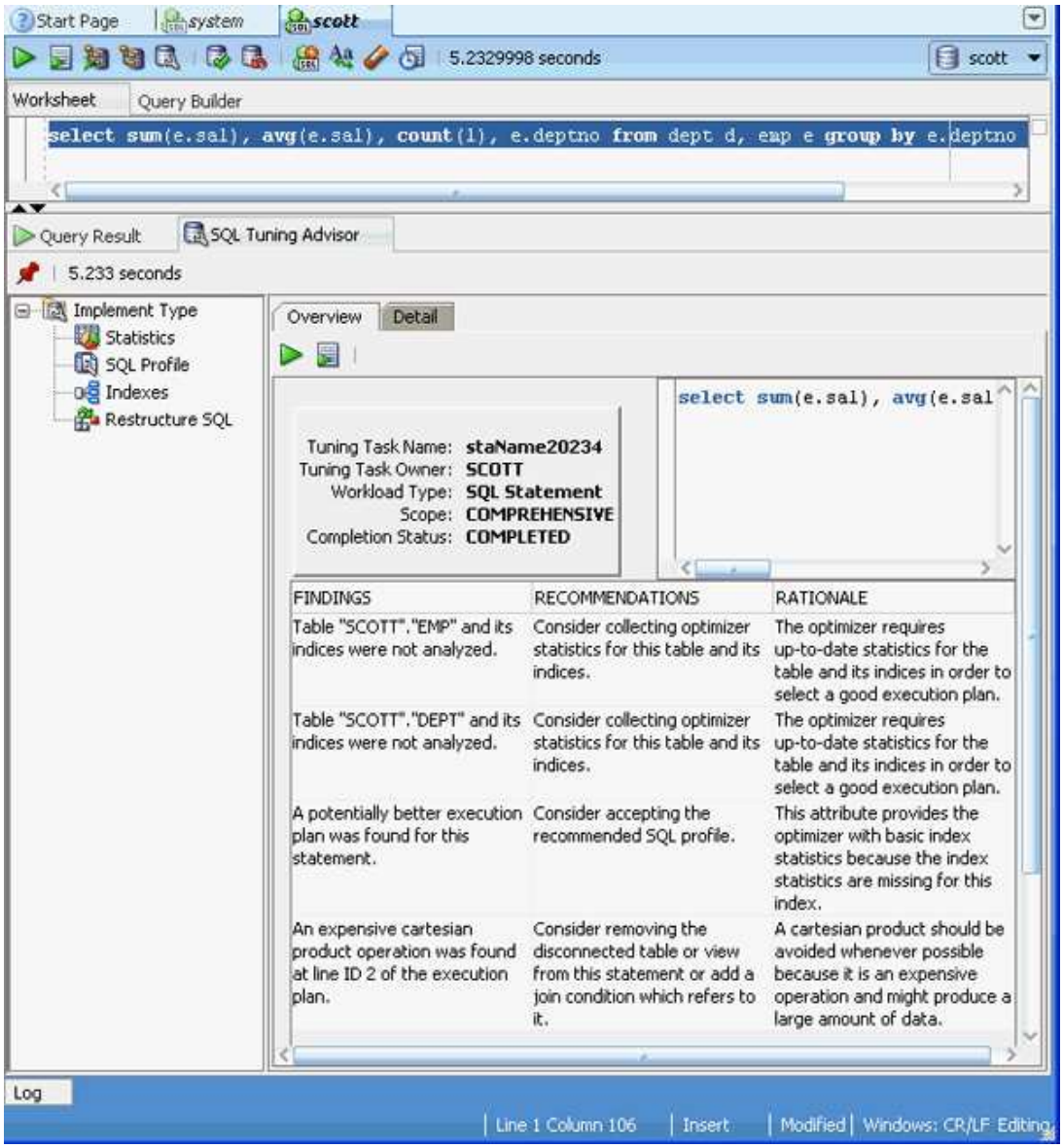

Figura 5-4 Saída da ferramenta Oracle Tuning advisor (http://www.oracle.com/webfolder/technetwork/tutorials/obe/db/sqldev/r30/TuningAdvisor/TuningAdvisor.htm)

Em outros trabalhos interessantes (Schnaitter et al, 2012) (Jimenez et al, 2012), é mencionada a preocupação com a captura do conhecimento de um DBA. Trata-se de uma ferramenta de sintonia fina, semi-automática, chamada Kaizen. Ela tem como objetivo sugerir recomendações de configuração de índices (criação e remoção) de forma *online* em função da carga de trabalho. Kaizen analisa a carga de trabalho, usa o método de dividir para conquistar no espaço de soluções e em seguida recomenda alterações no projeto físico (índices). O DBA analisa as recomendações, planeja o momento de materializar os índices e provê um *feedback* para a ferramenta a fim de refinar recomendações futuras. Para cada

sugestão de índice, a ferramenta armazena o *feedback* e preferências do DBA de duas possíveis formas: (i) explícita, quando um DBA indica se aceita ou não determinada recomendação ou (ii) implícita, que diz respeito às decisões do DBA que contrariam ações automáticas, como é o caso da remoção de um índice que tenha sido sugerido anteriormente pela ferramenta. A ferramenta infere, então, que o DBA não concordou com a sua sugestão e guarda esse conhecimento para as futuras sugestões. Pelo fato da ferramenta considerar o *feedback* do DBA para suas futuras decisões, ela sugere fortemente que sua utilização seja feita por parte de DBAs experientes.

No contexto de utilização dessa ferramenta Kaizen, pode se analisar um cenário de estudo de caso já usado por (Morelli et al, 2012) e indicado por (Bruno, 2011) como um dos desafios da área de sintonia de SGBD: a fragmentação de índices. Em casos onde se observam muitas atualizações em uma determinada tabela, um índice pode acabar sendo fragmentado e não mais usado durante a execução de consultas para as quais antes era considerado útil. O otimizador pode achar melhor realizar um *full scan* na tabela, pois esta ação acaba sendo menos custosa.

Com isso, o DBA pode optar por remover um índice e, consequentemente, a ferramenta pode atribuir um *feedback* negativo ao índice e descartá-lo em futuras sugestões. No entanto, se tal índice foi removido por estar fragmentado e, exatamente por essa razão não está sendo mais considerado em consultas, a ferramenta de sintonia fina automática provavelmente cometerá um erro. Caso seja reindexado, o mesmo índice não estaria mais fragmentado, passando a trazer benefícios para a carga de trabalho executada. Deve-se levar em consideração o cenário em que uma consulta antiga volte a ser utilizada com frequência após diversas atualizações sobre a tabela. Caso o DBA tivesse um modelo de apoio indicando que uma das razões pela qual o otimizador pode ter deixado de utilizar o índice fosse a fragmentação do mesmo, ele poderia optar por verificar o grau de fragmentação do índice antes de removê-lo.

A vantagem dessa ferramenta em relação à ADDM é que ela se preocupa em guardar um retorno do DBA. No entanto, ela não explicita as alternativas de índices que foram pensadas, o motivo da escolha final e o descarte das demais alternativas.

A ferramenta semi-automática que faz parte da abordagem descrita em (Vobugari et al, 2012) é direcionada aos sistemas OLTP (*Online Transaction Processing*). Inicialmente, define-se uma estratégia para priorizar consultas baseadas em determinados parâmetros de configuração. Dessa forma, a ferramenta libera o DBA da tarefa de selecionar as consultas mais importantes da sua carga de trabalho para análise.

Uma vez selecionado um subconjunto das consultas prioritárias, ocorre uma análise sobre o conjunto de índices que foram criados pelo DBA na configuração inicial da base de dados. Essa análise consiste em alinhar os índices com o subconjunto de consultas. Eventualmente, a ferramenta gera uma lista para o DBA com sugestões de índices a serem criados e índices considerados obsoletos, dada a taxa baixa de uso. O relatório gerado pela ferramenta indica: a coluna analisada; as consultas que referenciam tal coluna; a recomendação (ex.: índice a ser criado); um indicador do critério usado e a frequência da consulta.

No entanto, o raciocínio não é explicitado e a recomendação não possui justificativa. O DBA verifica a consulta que originou a recomendação de índice, mas não consegue verificar o raciocínio, a regra que definiu essa recomendação. Além disso, não são demonstradas quais cláusulas foram consideradas na análise das consultas nem quais regras foram usadas para a seleção das consultas mais importantes. O DBA não consegue obter garantias de que a carga de trabalho foi analisada por completo e que todas as possíveis opções de combinações de índices possivelmente úteis foram pensadas. Embora o DBA possa alterar o peso atribuído às consultas, seria importante que ele pudesse interferir na maneira automática que a ferramenta seleciona as consultas mais importantes.

No contexto de sintonia fina global, o trabalho de (Barrientos, 2004) propõe uma abordagem baseada em agentes de *software* para o ajuste automático dos parâmetros em um SGBD. Foram implementados dois mecanismos de auto-sintonia referentes aos problemas de escolha da estratégia de substituição de páginas em memória e ao de *thrashing* e contenção de dados. Esse trabalho não contempla justificativas para as decisões tomadas automaticamente e também não demonstra as possíveis alternativas de solucionar determinado problema, ou seja, não possui transparência das informações utilizadas. Além disso, o seu escopo se resume ao ajuste de parâmetros sem considerar as alternativas de criação, remoção ou recriação de estruturas de dados.

Também podemos mencionar outras ferramentas que realizam a manutenção de índices de forma semi-automática ou automática em SGBDs relacionais: a ferramenta proposta por (Morelli et al, 2012) sobre o SGBD PostgreSQL, que automaticamente cria, remove e/ou reindexa índices de forma *online* através da análise de sua carga de trabalho, atribuindo benefícios ou malefícios aos índices hipotéticos e reais de acordo com o seu uso pelo otimizador de consulta. A ferramenta Database Engine Tuning Advisor (DTA) (Bruno et al, 2011) do SGBD SQL Server da Microsoft, que sugere índices candidatos de acordo com a carga de trabalho, estimando o aumento de desempenho e o custo do índice através do uso do otimizador de consulta. A ferramenta proposta por (Rangaswamy et al, 2011) que indexa o banco de dados com a utilização de técnicas de mineração sobre a carga de trabalho de consultas. E por fim, a ferramenta proposta por (Monteiro et al, 2012), onde os autores propõem uma abordagem não-intrusiva para a manutenção automática e *on-the-fly* do projeto físico de bancos de dados, utilizando heurísticas que executam continuamente e, sempre que necessário, modificam o projeto físico corrente, reagindo a alterações na carga de trabalho de forma a considerar uma análise local do SGBD.

O trabalho descrito em (Bruno, 2011) indica o particionamento de dados como sendo uma forma de melhoria de desempenho de um SGBD que envolva grandes conjuntos de dados e carga de trabalho com consultas complexas. O particionamento pode ser utilizado em sistemas de banco de dados com processamentos paralelos, onde as partições são armazenadas em servidores autônomos diferentes. Essa estratégia pode até melhorar a gerência do servidor, por exemplo, se as tabelas e índices estiverem particionadas da mesma forma, fazendo com que operações tais como *backup/restore* possam ser drasticamente simplificadas.

(Rodd et al, 2013) apresenta um nova técnica que combina a habilidade de aprendizado da rede neural artificial com sistemas *fuzzy* para lidar com entradas imprecisas ao estimar as dimensões requeridas pela sintonia fina. Os valores estimados são analisados de acordo com uma base de conhecimento criada, usando resultados experimentais. O sistema recebe três parâmetros como entrada: número de páginas em memória, número de usuários ativos e tamanho do banco de dados. A partir desses parâmetros, o módulo *fuzzy* consulta uma base de regras e ações e gera os valores estimados para os parâmetros de sintonia fina. No artigo

são citadas algumas regras usadas. Contudo, essas regras são definidas em cima dos parâmetros iniciais recebidos pelo sistema. Não existe uma maneira formal de definir os conceitos que possam impactar a definição de parâmetros usados no momento da atividade de sintonia fina. Caso contrário, ajudaria a viabilizar um dos trabalhos futuros mencionados, que demonstra uma preocupação em analisar os impactos entre os parâmetros. Na definição das regras usadas, os autores mencionam que elas precisam ser cuidadosamente formadas e que, na maioria das vezes, requerem mudanças até que os resultados desejados sejam obtidos. No caso dessa mudança necessitar de um novo parâmetro ainda não recebido como entrada, não fica claro como o sistema iria se comportar.

Os autores em (CAO et al, 2013) e (Zhang et al, 2012) apresentam propostas para a sintonia fina do banco de dados orientado a aplicação, ou seja, combinando o desempenho da base de dados com o da aplicação. Muitas vezes o usuário da aplicação pode achar que o desempenho está sendo ruim por conta do banco, mas na verdade seria a aplicação que acessa o banco de dados que pode estar executando um comando de forma não otimizada.

A ferramenta AppSleuth (CAO et al, 2013) analisa gramaticalmente o código fonte de uma aplicação e o *log* de rastreamento das suas atividades para dois SGBDs comerciais objetivando propor melhorias de desempenho. No momento da análise, a ferramenta identifica as iterações da aplicação com o banco de dados e sugere alterações de melhorias. Por exemplo, sugerindo retiradas de laços de repetição (*loops*) que existam no código da aplicação ou reescrevendo consultas submetidas aos SGBDs, para se obter resultados com melhores desempenhos. Além disso, a AppSleuth também usa a ferramenta de sugestão de índices do SGBD Oracle com base na carga de trabalho levantada pelas consultas identificadas no código fonte da aplicação. Embora a preocupação da ferramenta seja relevante, não são apresentados os argumentos suficientes para que o responsável por melhorar o desempenho possa justificar a sua decisão ao optar por alterar algum ponto crítico da aplicação. A ferramenta também não indica explicitamente as sugestões de melhorias sobre reescrita de consultas e nem o motivo pelo qual está sugerindo uma análise de determinado trecho do código. O responsável pelo desempenho precisa saber o motivo para concordar ou não com o raciocínio empregado para o alerta.

Outro trabalho, o modelo proposto por (Zhang et al, 2012), objetiva dar suporte para a funcionalidade de sintonia fina e monitoramento de bancos de dados tradicionais. Existe um módulo responsável pelo monitoramento da aplicação, onde são levantadas as características de desempenho da aplicação. Também são registradas informações do tipo de sistema, ambiente do sistema operacional entre outras. Nesse módulo são coletadas as informações críticas da aplicação, que são passadas para o módulo de avaliação do desempenho. Paralelamente, tem-se o módulo responsável por monitorar a base de dados e coletar também suas informações.

Outro módulo definido no modelo é responsável por avaliar o desempenho. Ele recebe os parâmetros da aplicação e da base de dados, calcula e avalia o desempenho. Com o uso de algoritmos que determinam o que seria o comportamento normal da base de dados, o módulo avalia se o desempenho está bom ou ruim. Caso esteja ruim, o módulo responsável precisa identificar os problemas.

O módulo de conhecimento de otimização é o sinalizador dos pontos de falhas no desempenho da aplicação. Uma vez detectados os pontos de falha, o módulo de sintonia fina usa algoritmos e métodos para propor melhorias, fazendo uso das informações do módulo que provê avaliação de pesos para os serviços registrados de aplicações. Através desses pesos, ele consegue priorizar os recursos da base de dados de acordo com o peso atribuído à aplicação que a utiliza.

Após as sugestões de melhorias serem levantadas, as informações detalhadas sobre o processo de sintonia fina são registradas por um módulo de *log* e auditoria.

Por fim, o módulo de mineração de conhecimento de otimização recebe os dados do *log* e disponibiliza informações para serem extraídas. Essas informações podem ser usadas como referência para atividades futuras de sintonia fina ou até mesmo ser base para alguma ferramenta de auto-sintonia.

Este trabalho de pesquisa menciona que os módulos definidos no modelo foram aplicados sobre o SGBD Oracle e demonstra que os parâmetros usados para o cálculo do fator de impacto no desempenho foram definidos de acordo com o guia do SGBD. Embora exista uma tabela no artigo citando alguns parâmetros, o modelo não deixa explícito os parâmetros considerados, bem como a fórmula usada para cada cálculo realizado. Dessa maneira, o DBA não tem como confiar

completamente no processo proposto no modelo, pois ele não conhece os algoritmos usados e os parâmetros considerados. Além disso, se o modelo tiver que ser aplicado a outro SGBD, os parâmetros terão que ser revistos.

O trabalho apresentado em (Khouri et al, 2012) (Bellatreche et al, 2012) demonstra que existem dois principais atores no desenvolvimento de aplicações de *datawarehouse*: o responsável por elaborar o modelo conceitual e o administrador de dados (DBA). Neste trabalho, afirmam que as ferramentas propostas, tanto pela indústria quanto pela academia, para substituir tarefas complexas de ambos os atores, não são robustas.

Um dos propósitos desse tipo de ferramenta é recomendar estruturas de otimizações para a base de dados, tais como: índices e visões materializadas. Os autores apresentam uma metodologia que é capaz de recomendar índices *bitmap* a partir da especificação de requisitos realizada de maneira persistente. Cada requisito funcional ou não funcional é especificado usando um modelo de objetivos/metas. Os objetivos dos usuários são expressados no nível ontológico, onde são definidos um mapeamento entre coordenadas de cada objetivo (métrica, resultado e parâmetro) e as propriedades da ontologia de domínio.

A ontologia nesse trabalho é usada para prover uma visão integrada das fontes de dados do *datawarehouse*. O responsável pela modelagem conceitual seleciona as propriedades mais relevantes da ontologia para definir coordenadas de cada objetivo. A partir daí, é relacionada uma visão conceitual do *datawarehouse* aos requisitos, que são projetados para o modelo relacional. A visão conceitual é definida com base em regras especificadas em DL que provê o formalismo necessário para capturar os dados de modelagem. O conhecimento estruturado é descrito usando conceitos e regras. Dessa forma, os conceitos e propriedades do modelo conceitual são relacionados ao modelo de requisitos. As entidades (objetos e métricas) usadas para caracterizar os objetivos e as ações são relacionadas aos conceitos e regras do modelo conceitual. Em seguida, o modelo conceitual é transformado em um modelo lógico relacional através do uso de um conjunto de regras de transformações definidas. As instâncias do modelo lógico são transformadas em consultas SQL usando um processo de transformação de modelo para texto através do *plugin* Acceleo (OMG, 2013). Com isso, o algoritmo de seleção de índices (Bouchakri et al, 2011) baseado na cláusula `where` é aplicado e os índices são sugeridos.

Embora esse trabalho mantenha uma persistência no relacionamento entre os requisitos da aplicação e as sugestões de estruturas físicas para otimização do banco de dados e possua uma formalização sobre a definição dos conceitos envolvidos no domínio do *datawarehouse*, não existe a mesma preocupação com o processo de otimização. O trabalho formaliza os conceitos de domínio e as etapas de transformações dos modelos, através do uso de uma ontologia e regras DL, respectivamente. Mas ao realizar a sugestão de índices, não se preocupa em explicitar e relacionar os conceitos usados para o raciocínio do algoritmo. Sendo assim, os responsáveis pela base de dados conseguem apenas relacionar um índice ao requisito que o derivou e à semântica do requisito, mas não conseguem explicar quais índices poderiam ter sido usados, mas não foram selecionados, por exemplo.

É importante ter o rastreamento entre o requisito e a sugestão do índice, mas seria interessante que tal metodologia incorporasse o *framework* proposto nesta tese, adicionando semântica ao seu raciocínio e não só ao domínio. Assim, a tendência seria que a área de sintonia fina caminhasse cada vez mais para uma possível proveniência de otimização de dados. Com tal proveniência, todos os dados e sugestões estariam interligados e explicariam muitas situações que vão contra ao que muitos usuários possuem como premissas. Um exemplo dessa premissa é o fato de pensarem que índices são bons somente para consultas. Imaginando um requisito de manutenção de dados de uma conta corrente de uma empresa de comércio, poderia ser sugerido um índice que agilizasse tal manutenção.

5.2. Ferramentas para SGBDs em XML, RDF e Armazém de Dados

Para sistemas que consideram dados XML, tem-se a ferramenta proposta por (Elghandour, 2010) que sugere índices e visões materializadas (indexadas ou não) sobre dados XML. A ferramenta recebe como entrada a carga de trabalho de consultas (`XQuery`), o banco de dados XML, informações de sistema e regras de espaços em disco. A partir daí, enumeram-se os índices e visões relevantes segundo algumas heurísticas, simula-se o ambiente com a nova configuração (índices e visões hipotéticas) e calculam-se os benefícios através dos custos

estimados das consultas nesse novo ambiente. Ao final, a ferramenta sugere a nova configuração física, aquela com maior benefício potencial para o sistema de banco de dados.

Já no contexto de bancos de dados RDF, tem-se o assistente proposto por (Goasdoué et al, 2011) que realiza a sintonia de acordo com a carga de trabalho expressa em SPARQL sobre triplas RDF e a utilização do RDF Schema, caso exista. O assistente armazena as triplas em RDF em uma única tabela no SGBD Relacional PostgreSQL, assim como as visões materializadas sugeridas.

Inicialmente, o administrador da base de dados deve indicar o conjunto de triplas RDF (dados), respectivo conjunto de consultas e caso tenha, o RDF Schema. O RDF Schema é utilizado para responder consultas que não teriam respostas sendo submetidas somente sobre os dados, sendo necessária a utilização de semânticas associadas. Antes de sugerir a melhor configuração, o assistente de sintonia fina questiona as preferências do administrador da base, sendo por busca rápida ou por busca que mesmo mais longa tenha a garantia de solução ótima. Além disso, são atribuídos pesos, ou seja, graus de importância, aos seguintes componentes: tempo de execução de consulta, manutenção da visão e espaço necessário. Uma vez indicados todos os parâmetros necessários, o assistente reformula as consultas com uso do RDF Schema, busca similaridades entre subconsultas na carga de trabalho, submete heurísticas, materializa as visões úteis e armazena as consultas da carga de trabalho já reescritas utilizando-se as visões resultantes. Em seguida, apresenta-se as visões sugeridas, o custo de espaço e o ganho de desempenho para que o administrador da base possa optar por uma sintonia diferente da sugerida.

Já no contexto de armazéns de dados (*datawarehouse)*, encontram-se trabalhos que possuem a preocupação em combinar algumas das principais técnicas de sintonia fina. O trabalho de (Bellatreche et al, 2004) conduz experimentos e análises de situações, utilizando o *benchmark* TPC-H (TPC, 2013), onde o uso em conjunto de técnicas como índices, visões materializadas e particionamento de dados proporcionam melhor desempenho do que o uso dessas mesmas técnicas de forma isolada. O trabalho conclui que aparentemente cada técnica usada de forma isolada pode não propor o melhor resultado em questões de desempenho e custo total.

Em (Bouchakri et al, 2011) tem-se outro exemplo de combinação de técnicas de sintonia fina. Nesse caso, combinam-se as técnicas de particionamento de dados horizontal e índices *bitmap*, considerando que tais técnicas possuem em comum a competição por seleção de atributos.

5.3. Considerações finais

As ferramentas e abordagens de sintonia fina existentes buscam sugerir ações de sintonia fina, sem se preocupar em reunir, em alto nível, uma maneira de o DBA comprovar a eficácia do seu trabalho e adequar o raciocínio ao seu cenário de conhecimento. Caso o DBA discorde de algo oferecido pela ferramenta ou abordagem, pode não ser viável incluir ou alterar qualquer raciocínio.

Dessa forma, apenas um DBA experiente seria capaz de selecionar a melhor ferramenta para auxiliá-lo de acordo com o cenário imposto. E as ferramentas e abordagens que consideram mais de uma técnica não possuem flexibilidade o suficiente para considerar novas técnicas que surjam no futuro ou até mesmo que já existam e o DBA queira acrescentar, pois ainda não é considerada.

Outra desvantagem dessas ferramentas e abordagens é o fato de a maioria delas indicar apenas a melhor solução de configuração de sintonia fina. Dependendo da experiência do DBA e do seu conhecimento sobre o comportamento da base de dados, ele poderia preferir uma segunda ou terceira opção de melhoria de desempenho.

A **Tabela 5-1** apresenta um resumo comparativo entre as ferramentas levantadas no presente capítulo, considerando os requisitos descritos no capítulo 1, que motivaram o desenvolvimento do *framework* proposto nesta tese. É importante lembrar que as ferramentas levantadas não possuem o mesmo objetivo do *framework* e por isso, muitos requisitos não se aplicam às mesmas.

Ferramentas	**Explicação sobre decisão**	***Feedback* do DBA**	**Semântica**	**Apresentação de descarte de alternativas**	**Possibilidade de extensão**
ADDM	Sim	Não se aplica	Não se aplica	Não se aplica	Não se aplica
Kaizen	Não se aplica	Ações	Não se aplica	Não se aplica	Código-fonte
(Vobugari et al, 2012)	Não se aplica	Pesos (consultas)	Não se aplica	Não se aplica	Código-fonte
(Barrientos, 2004)	Não se aplica	Não se aplica	Não se aplica	Não se aplica	Código-fonte
DTA	Não se aplica	Não se aplica	Não se aplica	Não se aplica	Não se aplica
(Monteiro et al, 2012)	Não se aplica	Não se aplica	Não se aplica	Não se aplica	Código-fonte
(Bruno, 2011)	Não se aplica	Não se aplica	Não se aplica	Não se aplica	Código-fonte
(Rodd et al, 2013)	Não se aplica	Não se aplica	Não se aplica	Não se aplica	Regras
(Elghandour, 2010)	Não se aplica	Não se aplica	Não se aplica	Não se aplica	Código-fonte
(Goasdoué et al, 2011)	Não se aplica	Parâmetros	Dados	Não se aplica	Código-fonte
(Bellatreche et al, 2004)	Não se aplica	Não se aplica	Não se aplica	Não se aplica	Código-fonte
(Bouchakri et al, 2011)	Não se aplica	Não se aplica	Não se aplica	Não se aplica	Código-fonte
AppSleuth	Não se aplica	Não se aplica	Não se aplica	Não se aplica	Código-fonte
(Zhang et al, 2012)	Sim	Pesos (serviços)	Não se aplica	Não se aplica	Código-fonte
(Khouri et al, 2012	Não se aplica	Não se aplica	Dados	Não se aplica	Regras
Outer-Tuning	**Sim**	**Sim**	**Sim**	**Sim**	**Sim**

Tabela 5-1 Resumo comparativo de ferramentas de sintonia fina

Diante da tabela apresentada (**Tabela 5-1**) com a comparação do *framework* em relação às demais ferramentas de sintonia fina pesquisadas, ele pode ajudar o DBA na comprovação da eficácia do seu trabalho. O *framework* flexibiliza a

definição dos conceitos, que disponibilizam argumentos para serem usados como justificativas pelo DBA. No futuro, pode até ser implementada uma maneira de oferecer uma priorização das melhores soluções no momento da realização de sintonia fina no banco de dados. Essa ordenação das possíveis soluções pode auxiliar o DBA e/ou a ferramenta que realiza a auto-sintonia no SGBD de forma semi-automática ou totalmente automática, a guiar a sua análise sobre a melhor forma de realizar a sintonia do banco de dados bem como a comprovar a eficácia de sua decisão final.