Chapter

4

(Auto) Sintonia-fina em Sistemas de Bancos de Dados nas Organizações

Ana Carolina Brito de Almeida, Sérgio Lifschitz

Abstract

In a vast quantity of data (Big Data), organizations are increasingly demanding agility in processing database queries. In this mini-course we intend to address the issue of the database (self) tuning in the organizations. Database tuning is a complex task that improves database performance. This technique demands deep and specific knowledge on architecture, parameters and design of the database system. The course is intended for presentation on two main topics: an overview of the (self) tuning process in Relational Database Management Systems (DBMSs) and Applications of self (tuning), in practice, by organizations.

Resumo

Na atual era de grandes volumes de dados (big data), as organizações estão demandando cada vez mais a agilidade no processamento de requisições aos bancos de dados. Neste minicurso pretendemos abordar o tema de (auto) sintonia fina (self-tuning) em sistemas de bancos de dados nas organizações. A sintonia fina de banco de dados é uma tarefa complexa que visa melhorar o tempo de resposta de operações realizadas na base de dados. Essa técnica exige grande conhecimento da arquitetura, parâmetros e projeto do sistema de banco de dados. O curso está pensado para apresentação sobre dois tópicos principais: uma visão geral do processo de (auto) sintonia em Sistemas de Gerência de Bancos de Dados (SGBDs) Relacionais e Aplicações de auto (sintonia), na prática, pelas organizações.

4.1. Introdução

Os sistemas de informações estão cada vez mais complexos e gerando grandes volumes de dados (*big data*) para serem armazenados em SGBDs. *Big Data* é definido por três

V's: volume, variedade e velocidade [Berman 2013]. O volume refere-se a grande quantidade de dados; a variedade significa que tais dados podem vir em formatos diferentes, incluindo as bases de dados tradicionais, documentos, imagens e registros considerados complexos; e por fim, a velocidade indica que o conteúdo dos dados é atualizado constantemente, através da absorção de coleções de dados complementares, dados previamente arquivados, dados legados e dados de diversas fontes.

Diante disso, torna-se imprescindível para os sistemas de informações críticos obterem um bom desempenho do SGBD com respostas ágeis e precisas sobre esse grande volume de dados. A tarefa de *tuning* de bancos de dados ou sintonia fina surgiu com o objetivo de atingir o bom desempenho, buscando diminuir o tempo de resposta das consultas submetidas aos bancos de dados bem como aumentar a vazão (*throughput*), ou seja, aumentar o número de consultas executadas no mesmo período de tempo [Shasha and Bonnet 2003].

É necessário ressaltar a diferença entre a tarefa de sintonia fina de banco de dados e a tarefa de otimização de consultas, em particular no contexto de SGBDs relacionais. A otimização de consultas é o processo realizado por um SGBD para converter um comando declarativo, escrito, por exemplo, na linguagem SQL (*Structured Query Language*), em um plano de execução. Nesse plano, são especificados todos os operadores que são aplicados aos dados e de que forma eles são ordenados e compostos. Assim, a otimização de consultas é um processo automático, realizado por componentes do SGBD. Já a atividade de sintonia fina é realizada, na maioria das vezes, pelo administrador do banco de dados (DBA - *DataBase Administrator*) ou por um especialista em ajuste de desempenho.

A sintonia fina descreve uma série de atividades que podem ser realizadas para otimizar o desempenho dos bancos de dados, sendo considerada uma tarefa complexa. Realizam-se ajustes das configurações dos bancos de dados, parâmetros e projeto físico, seleção de estruturas de acesso, duplicação de estruturas físicas, sempre de acordo com a carga de trabalho submetida ao banco. Entende-se carga de trabalho como sendo um conjunto de consultas ou atualizações, ponderadas por suas frequências de ocorrência.

Por ser uma tarefa complexa, torna-se inviável, em muitos casos, monitorar um sistema de banco de dados, tomando decisões em tempo real, sem apoio de ferramentas especialistas. É importante que tais ferramentas sejam capazes de adaptar-se dinamicamente às modificações da carga de trabalho [Bruno 2011] [Lifschitz et al. 2004]. Diante dessa necessidade de automação, surge o conceito da auto-sintonia de banco de dados, sintonia fina automática de banco de dados ou *self-tuning* de banco de dados. Auto-sintonia de banco de dados significa o ajuste automático de configurações que buscam melhorar o desempenho através da diminuição do tempo de resposta sobre as execuções de transações ou comandos submetidos ao banco de dados. Praticamente todos os SGBDs de mercado oferecem apoio de sintonia automática, mas poucos são os profissionais que realmente conhecem as ferramentas ou mesmo têm condição de utilizá-las. Existem diversas propostas de ferramentas que auxiliam a sintonia fina através da automatização de estratégias para a seleção de índices e visões materializadas, particionamento, reescrita de consultas, entre outras.

O estudo da auto-sintonia de sistemas computacionais não é recente, como pode-se constatar em [Ferrari, 1978]. Entretanto, a sua relevância para a área de banco de

dados e sistemas de informações aumentou significativamente nos últimos anos. Espera-se que, no futuro, os sistemas demandem de uma quantidade menor de profissionais especializados para a manutenção de seu desempenho e de sua operação. Em um cenário ideal, os sistemas conseguiriam alcançar um desempenho adequado sem qualquer necessidade de ajuste manual. Mais do que isto, à medida que as cargas de trabalho submetidas ao sistema mudassem, este iria procurar se adaptar dinamicamente para continuar apresentando um desempenho aceitável.

Além disso, o próprio desenvolvedor de sistemas pode contribuir para a melhoria do desempenho, com conhecimentos de boas práticas em sintonia fina, não precisando aguardar por uma intervenção do administrador do banco de dados. Por exemplo, o desenvolvedor pode realizar reescrita das consultas que são elaboradas, de forma automática, por *frameworks* de desenvolvimento.

Este minicurso está organizado da seguinte maneira:

- ✓ **Introdução e visão geral de (auto) sintonia fina de bancos de dados:** apresentação dos principais conceitos envolvidos, em particular processamento e otimização de consultas e atividades de sintonia fina (*tuning*) de bancos de dados. Pretende-se abordar vários aspectos de sintonia fina, porém dando ênfase ao *tuning* de projeto físico.

- ✓ **Aplicações de auto (sintonia) nas organizações:** apresentação de ferramentas e alguns exemplos práticos, ilustrando diversas situações reais e frequentemente encontradas no dia a dia de uma empresa. Estes exemplos envolvem cenários de otimizações de consultas, demonstrando o custo e desempenho de consultas antes e após a sua reescrita de forma simples. Além disso, sugere-se boas práticas para serem adotadas tanto por equipes de desenvolvimento de sistemas de informação quanto por equipes de administradores de bancos de dados. Destaca-se o cuidado que os desenvolvedores precisam ter ao adotar a utilização de *frameworks* que elaboram consultas de forma automática para serem submetidas aos bancos de dados.

O presente minicurso objetiva complementar e aprofundar os conhecimentos de estudantes e profissionais interessados na área de banco de dados e em sistemas de informações que fazem uso de bancos de dados. Embora a ênfase seja dada no modelo relacional, diversos assuntos abordados e discutidos se aplicam também a outros modelos de dados.

4.2. Processamento e Otimização de Consultas

O SGBD é um sistema considerado complexo e que consiste de muitos módulos. Entre esses módulos inclui-se o módulo de implementação do catálogo, do processador de linguagem de consultas, da interface, do controle de concorrência, de segurança e recuperação entre outros [Elmasri and Navathe 2016].

Para contextualizar o presente minicurso, descreve-se o módulo do processador de linguagem de consultas, pois é o mais impactante em termos de sintonia-fina, no escopo de projeto físico do sistema de banco de dados. Nesse módulo que os comandos

DML (*Data Manipulation Language* – Linguagem de Manipulação de Dados) submetidos aos bancos de dados são avaliados e um plano de acesso e execução é gerado.

O processo de avaliação de uma consulta, de uma forma geral, é composto por cinco etapas (Figura 4.1):

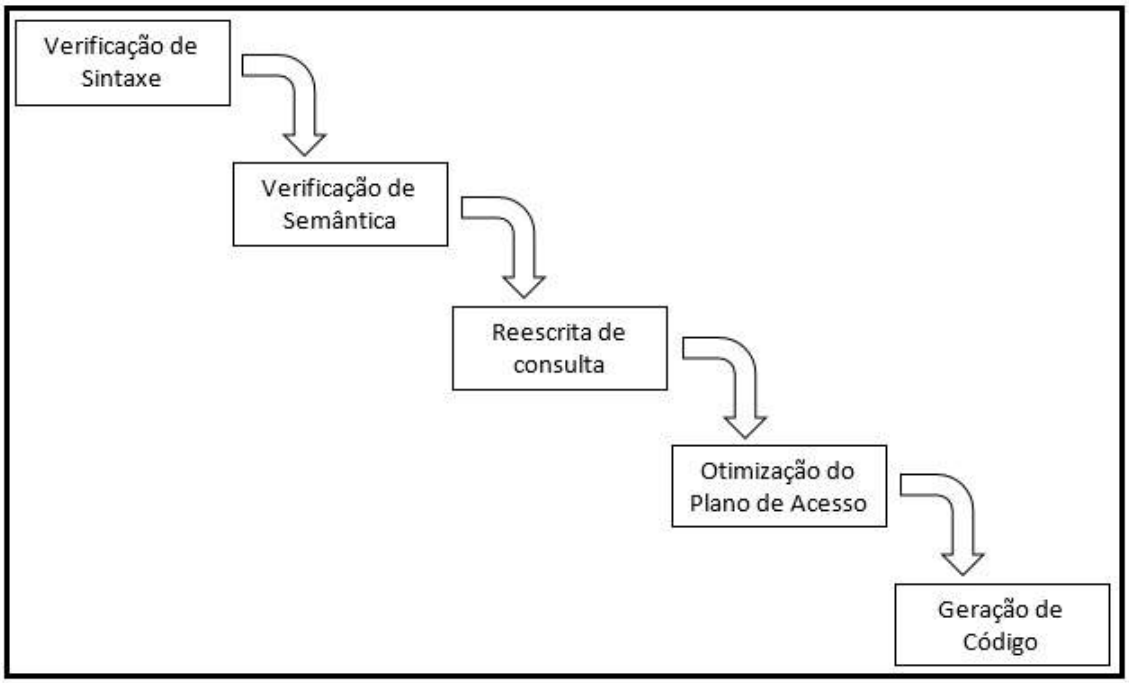

Figura 4.1 Etapas do processo de avaliação de consultas

- ✓ **Verificação de sintaxe** – verifica-se a sintaxe da consulta e detecta-se os erros sintáticos baseados nos *tokens* conhecidos (ex.: SELECT, FROM) pelo SGBD em sua gramática;
- ✓ **Verificação de semântica** – verifica se os objetos referenciados no comando SQL existem e se estão acessíveis para o usuário que submeteu o comando;
- ✓ **Reescrita de consulta** – o comando SQL é reescrito a partir de uma linguagem em alto nível (SQL) para uma representação mais adequada para sua manipulação interna (Álgebra relacional).
- ✓ **Otimização do plano de acesso** – para a execução do comando SQL, o SGBD deve possuir sua representação em unidades mais simples, as quais ele seja capaz de mapear, isoladamente, para suas rotinas básicas. Uma combinação destas unidades, em uma determinada ordem de execução, é chamada de plano de acesso e execução. Usualmente, vários planos de acesso e execução levam ao resultado desejado, mas, quando executados, possuem diferentes tempos de execução. Ao SGBD cabe montar o plano de acesso e execução que seja mais eficiente. De acordo com o número de tabelas utilizadas e a complexidade da consulta (que pode incluir várias subconsultas, por exemplo), o espaço de busca pode ser bastante grande. É importante que o tempo de otimização da consulta

não seja superior ao tempo que pode ser ganho com a escolha de um plano mais adequado. A otimização só vale a pena se o custo associado da execução do plano selecionado for melhor do que a execução de um plano qualquer escolhido aleatoriamente;

- ✓ **Geração de código** – o plano de acesso e execução selecionado na etapa anterior é transformado para chamadas às rotinas básicas do SGBD, as quais são executadas pelo mesmo.

Nesse ponto cabe observar que é possível mostrar que o processo de otimização de consultas em um banco de dados relacional é NP-difícil no número de relações envolvidas. Assim, sabe-se que, na prática, os SGBDs relacionais não determinam a solução ótima, mas sim, a melhor solução possível baseada em heurísticas.

4.3. Desempenho do Sistema de Banco de Dados

[Shasha and Bonnet 2003] enumeram cinco princípios básicos que permeiam ao considerar desempenho (Figura 4.2). São eles: (i) Pense globalmente, corrija localmente; (ii) Particione para quebrar os gargalos; (iii) Custos iniciais são altos, custos de execução são baixos; (iv) Deixe no servidor o que é do servidor; (v) Esteja preparado para conflitos em escolhas (*Trade-Offs*).

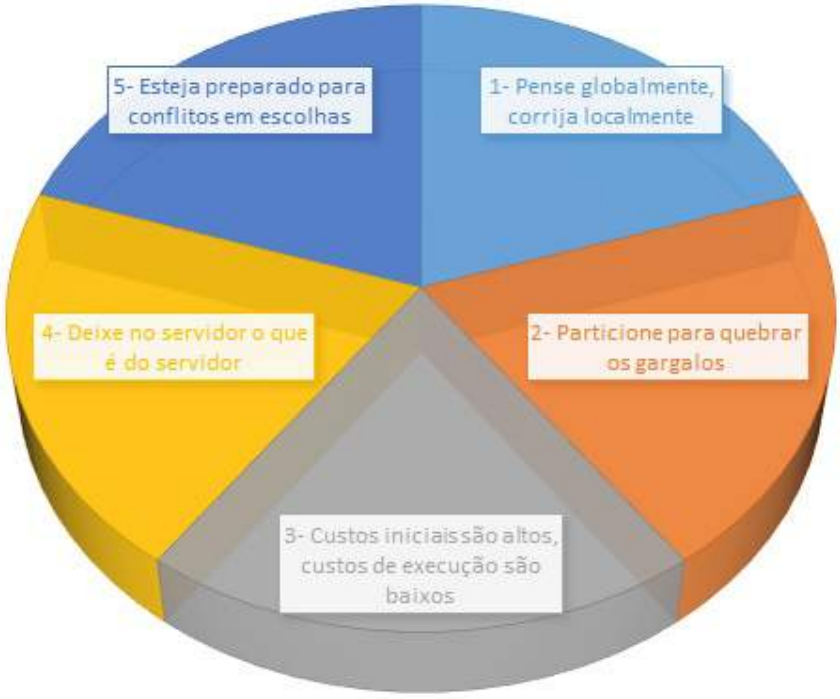

Figura 4.2 Princípios básicos – Desempenho do banco de dados

O primeiro princípio alerta que poucas intervenções simples, se aplicadas corretamente, podem aumentar o desempenho de forma considerável no sistema de banco de dados como um todo. Um exemplo clássico, citado por [Shasha and Bonnet 2003], é que o DBA pode analisar as estatísticas de *hardware* para verificar a utilização do processador, a atividade de entrada/saída (I/O), paginação entre outros itens. Caso o

valor de requisições ao disco seja alto, o DBA pode optar por comprar mais disco rígido para diminuir esse valor. No entanto, essa decisão pode não resolver o problema. A causa do problema pode ser uma consulta, frequentemente executada, que realiza muitas varreduras em uma tabela. Nesse caso, bastaria que o DBA criasse um índice ou movesse os arquivos de dados para discos diferentes. Essa solução seria mais barata e mais efetiva do que comprar mais *hardware*.

O segundo princípio ressalta que quando o DBA identificar um gargalo no banco de dados, deve tentar primeiro acelerá-lo. Caso não consiga, deve particioná-lo. Esse particionamento pode ser por espaço, recurso lógico ou por tempo. Por exemplo, o particionamento de recursos físicos por espaço pode ser útil em um cenário de um banco nacional com várias agências espalhadas pelo Brasil. Considerando que a maioria dos clientes acessam os dados de sua conta bancária próximo ao local de onde reside, uma solução natural seria colocar os dados dos clientes em um local L dentro de um subsistema L. Esse local L poderia ser cada região do Brasil (Sudeste, Sul, Nordeste etc), onde existiria um servidor de banco de dados com dados sobre contas bancárias somente da sua região. Dessa forma, os clientes estariam próximos aos dados das suas contas bancárias, não impactando ou concorrendo com consultas de outros clientes do restante do Brasil.

O terceiro princípio indica o uso de menos recursos de inicialização, considerado mais caro. Por exemplo, é muito comum o cenário onde um programador desenvolve um sistema de informação que acessa o banco de dados. Nesse caso, é melhor que o programador opte por uma única chamada de conexão ao banco de dados, se o comando de consulta ao banco de dados envolver o mesmo conjunto de dados e posteriormente, crie um bloco de comando de repetição para percorrer o conjunto resultado da consulta. Essa solução é melhor do que criar um bloco de repetição, onde em cada interação é aberta uma conexão com o banco de dados e cada chamada tenha o seu próprio comando de consulta. Imaginando que o comando de consulta retorne dez milhões de linhas, é melhor submeter o comando uma única vez ao banco de dados e posteriormente, percorrer o seu conjunto de linhas resultante do que realizar dez milhões de chamadas ao banco de dados.

O quarto princípio visa o equilíbrio das tarefas entre o servidor de banco de dados e os demais servidores (ex.: servidor de aplicação) e clientes. Por exemplo, é importante levantar onde reside a informação relevante. Imagina-se o seguinte cenário: um sistema de informação precisa de alguma resposta do banco de dados toda vez que ocorrer uma determinada mudança no registro do cliente para que seja disparado um e-mail para o mesmo. Dado que a informação necessária e que deve ser acompanhada reside no banco de dados, é melhor que seja criado um gatilho (*trigger*) no servidor do banco, que seja disparado toda vez que ocorrer alguma mudança do que um método no sistema de informação que fique consultando a tabela periodicamente com requisições ao banco. O método, no servidor de aplicação, pode submeter requisições ao banco sem necessidade. Já o gatilho só vai ser disparado quando, de fato, ocorrer alguma alteração nos dados da tabela de cliente.

O quinto e último princípio chama atenção para os possíveis impactos de uma decisão ou escolha. A melhoria de desempenho de um sistema requer a combinação de memória, disco ou recursos computacionais. Por exemplo, a criação de um índice para

agilizar uma determinada consulta requer mais espaço em disco e mais espaço na memória de acesso randômico. Além disso, em momentos de inserções e atualizações no banco de dados requer mais tempo de processamento e mais acessos ao disco, quando não usa o índice. Dessa forma, o DBA deve estar ciente sobre os conflitos de interesse em suas escolhas e decisões.

4.4. Sintonia-fina de Bancos de Dados

A sintonia-fina, no contexto deste minicurso, envolve a realização de ajustes em sistemas de banco de dados de forma a obter um tempo de resposta menor e/ou aumentar a vazão para determinada aplicação. A tarefa de sintonia-fina é considerada difícil, pois requer conhecimento profundo da aplicação que utiliza os dados, do SGBD que gerencia os bancos de dados, do sistema operacional e da parte física do *hardware*.

Para realizar a tarefa de sintonia fina do banco de dados, de forma adequada, é essencial ter um bom entendimento dos planos de acesso e da maneira como o SGBD os elabora. É através desse plano que se pode verificar a validade de determinadas operações de sintonia fina, comparando-se os planos de acesso obtidos antes e após suas realizações.

Os planos de acesso são montados a partir da utilização dos métodos de acesso e das operações de manipulação de dados. Para a escolha das operações a serem utilizadas e a elaboração dos planos de acesso, os SGBDs devem tomar algumas decisões. Tais decisões podem ser baseadas em regras ou em custos.

Os otimizadores baseados em regras utilizam um conjunto de regras para escolher a ordem das operações que devem realizar. Essas regras são hierarquizadas de forma a fazer com que as operações potencialmente mais restritivas sejam realizadas antes das demais. Por exemplo, o otimizador prioriza uma operação que busque por uma única linha baseada no identificador dessa linha do que uma operação que precise realizar uma varredura completa na tabela.

Já os otimizadores baseados em custo atribuem um valor de custo para cada operação a ser realizada. O plano escolhido é aquele que apresenta o menor custo total. Todos esses custos são calculados com o uso de informações extraídas das estatísticas armazenadas no metadado do banco. Tais estatísticas podem compreender informações referentes às tabelas como um todo (ex.: número de tuplas e de blocos e o tamanho médio de uma tupla), aos atributos (ex.: número de valores distintos em cada um) e aos índices (ex.: número de níveis). No entanto, nem sempre as estatísticas estão atualizadas no momento da geração do plano de acesso, podendo acarretar em um plano mal estruturado. Manter todas as estatísticas atualizadas apresenta um custo que, em ambientes com grande quantidade de atualização e elevado nível de concorrência por recursos, pode ser bastante alto. A geração e utilização das estatísticas de forma adequada são fatores que afetam diretamente o desempenho de um SGBD. Dessa forma, o DBA deve avaliar, cuidadosamente, quando atualizar as estatísticas.

Quando são utilizados otimizadores baseados em custos, esses custos atribuídos aos diversos planos podem ser usados como um parâmetro de comparação. Já quando o otimizador baseado em regras é utilizado, faz-se necessário maior conhecimento das várias operações que são realizadas e dos modos como podem ser reordenadas. Em

ambos os casos, deve-se avaliar as alternativas existentes, que vão desde a alteração de parâmetros de inicialização até a utilização de dicas, para induzir o SGBD a adotar um plano de execução que tenha melhor desempenho.

A sintonia fina em bancos de dados busca, na grande maioria das vezes, reduzir a quantidade de operações de E/S (Entrada e Saída), que são comparativamente lentas em relação às operações em memória principal, sendo a sua realização o principal gargalo no processamento de consultas e atualizações em SGBDs. Nesse sentido, pode-se realizar alterações que sejam pontuais, afetando um comando específico ou operações que apresentam maior abrangência, podendo favorecer ou prejudicar diversas operações do SGBD.

Pode-se dividir a tarefa de sintonia-fina de acordo com o componente que se deseja otimizar. São eles: consulta, transação, memória e projeto físico.

4.4.1. Sintonia-fina de consulta

A sintonia-fina de consulta é a menos impactante de todas, pois não onera outro recurso. Existem dois caminhos para detectar que uma consulta está com o desempenho lento no banco de dados: (i) a consulta está realizando muitos acessos ao disco, por exemplo, uma consulta que varre uma tabela inteira; (ii) se o plano de execução da consulta não estiver usando índices que são considerados relevantes para a sua execução.

Caso seja detectado esse tipo de lentidão, é necessário analisar se é possível reescrever a consulta de forma que o otimizador possa interpretá-la de outra forma e optar por um caminho mais barato para executá-la. Na etapa de aplicações de sintonia-fina apresenta-se alguns exemplos de melhorias de desempenho apenas com a reescrita de consultas.

4.4.2. Sintonia-fina de transação

O conceito de transação provê um mecanismo para descrever unidades lógicas de processamento do banco de dados [Elmasri and Navathe 2016]. Uma transação inclui uma ou mais operações de acesso ao banco de dados – operações de inserção, remoção, atualização ou recuperação.

A sintonia-fina de transação inclui atividades que gerenciam o controle concorrente das transações no banco de dados. Entre essas atividades estão: eliminar bloqueios quando são desnecessários através da modificação da ordem dos comandos que seguram os bloqueios; usar o nível de isolamento apropriado para as transações; usar as características de multi-versionamento no sistema para consultas longas e que apenas leem dados; gerenciar a granularidade dos bloqueios; analisar o intervalo de verificação de *deadlocks*.

4.4.3. Sintonia-fina de memória

Os SGBDs utilizam a memória para armazenar os dados mais recentemente acessados. Dessa forma, as operações de atualização que são realizadas por usuários podem fazer com que existam, em memória, várias versões da mesma informação. Além disso, a memória armazena informações sobre as árvores de execução de comandos SQL, sobre

a instância utilizada e a base de dados acessada e os processos do servidor e seus parâmetros de controle.

Os SGBDs utilizam diversas áreas da memória (ex.: buffer pool, redo log buffer). A sintonia-fina de memória busca dimensionar corretamente essas áreas de memória para aumentar desempenho. Usualmente, busca-se aumentar o *hit ratio*, ou seja, a taxa de acerto de se encontrar um dado necessário já na memória. A proporção do número de ocorrências é dividida pela referência total da memória da CPU e é chamada de *hit ratio*.

4.4.4. Sintonia-fina de projeto físico

O projeto físico do banco de dados descreve como os dados são armazenados como arquivos no computador através da representação da informação, tais como: formatos de registro, ordenação dos registros e caminhos de acesso [Elmasri and Navathe 2016]. Caminho de acesso é uma estrutura de busca que realiza a busca para determinados registros de forma mais eficiente (ex.: índices).

O problema do projeto físico é formalizado por [Bruno 2011] como:

> *Dado uma carga de trabalho **W** consistindo de consultas e atualizações e um limite de armazenamento **A**, obtenha a configuração de índices **C** que caiba em **A** e resulte em consultas na **W** executando tão eficientemente quanto possível.*

Assim, a sintonia-fina de projeto físico inclui a seleção das estruturas de índices adequadas, mas também as visões materializadas a serem utilizadas, as tabelas a serem particionadas e os tipos de particionamento mais adequados e até mesmo, a desnormalização de tabelas, como forma de acelerar o desempenho das consultas.

Índices são estruturas auxiliares que visam permitir o acesso mais rápido aos dados. Para isso, os índices são organizados de forma diferente da utilizada para os dados. Estes podem estar armazenados em diversos tipos de arquivos, sendo, os mais comuns, os arquivos de tipo *heap*, onde os dados estão armazenados em sequência aleatória, e ordenado, onde os dados são mantidos fisicamente ordenados por um conjunto de um ou mais atributos (ou colunas) de uma tabela.

As visões materializadas são réplicas dos dados originais em formatos mais adequados para a realização de consulta. Neste caso, tem-se que estar atento para o espaço em disco disponível e a atualização dos dados. Como visões materializadas são réplicas dos dados, a cada atualização dos dados originais as visões devem ser atualizadas também, gerando mais atividade na base de dados. Alguns SGBDs provêem mecanismos automáticos para a realização destas atualizações. Em alguns casos, porém, é desejável realizar a atualização das visões materializadas de forma assíncrona com a atualização das tabelas que a originam.

O particionamento é outra técnica que pode ser utilizada para reduzir a quantidade de operações de E/S. O particionamento horizontal, implementado como um recurso em muitos SGBDs, permite que sejam criados subconjuntos da tabela (ou

índice) original. Em consultas onde o atributo utilizado para realização da partição é utilizado como filtro, os otimizadores podem acessar somente a partição adequada, reduzindo o espaço de busca e a quantidade de operações de E/S. As partições podem ser alocadas, inclusive, em diferentes discos.

Outra forma de realizar a sintonia-fina de projeto físico é a desnormalização, pois a realização de junções pode gerar um problema de desempenho. Em algumas situações é aceitável recorrer a essa técnica no esquema do banco de dados. Isso ocorre, muitas vezes, em ambientes onde são realizadas poucas ou nenhuma operação de atualização de dados. É importante ressaltar que a desnormalização pode trazer vários problemas e sua realização requer uma avaliação detalhada dos impactos sobre as aplicações que acessam a base de dados.

4.5. Auto Sintonia-fina de Bancos de Dados

A auto sintonia fina de bancos de dados refere-se ao ajuste, de forma automática, dos parâmetros, configurações e estruturas de um SGBD com o objetivo de processar mais eficientemente uma determinada carga de trabalho.

Na literatura, existe um conjunto de princípios básicos para os componentes de auto sintonia. Alguns deles são:

- ✓ Definição do ambiente em que o componente de auto sintonia está inserido. Esse ambiente significa determinar o escopo em que a auto sintonia deve ser aplicada, ou seja, de forma local ou global. No escopo local, o objetivo do componente é automatizar uma atividade de sintonia específica e no escopo global, considera-se a sintonia do sistema de banco de dados como um todo.

- ✓ Modificação do sistema através de um ciclo de controle de realimentação. Um ciclo de controle de realimentação é um método para controlar o efeito das decisões de sintonia sobre o desempenho do sistema. O método se divide em três fases: observação, predição e reação. Na fase de observação ocorre a coleta de métricas e estatísticas que permitem avaliar se existe algum problema de desempenho em algum componente específico ou no sistema como um todo. A fase de predição consiste em estimar o desempenho futuro do sistema com base nas métricas coletadas e tomar decisões sobre quais ações podem ser executadas caso seja necessário sintonizar o sistema. Por fim, na fase de reação, o componente de auto sintonia implementa as ações de sintonia do sistema.

- ✓ Coleção de estatísticas e informações de forma que haja pouco impacto no desempenho do sistema. Determinar se há um problema de desempenho no sistema que possa ser resolvido através de ações de sintonia exige que seja realizado um diagnóstico. Este diagnóstico é baseado em um conjunto de métricas e estatísticas coletadas durante a operação do sistema.

- ✓ Utilização de modelos matemáticos e estimativas para prever o desempenho futuro do sistema. Um componente de auto sintonia precisa prever o desempenho futuro do sistema com base em métricas coletadas periodicamente. Os modelos formais sobre o desempenho do sistema podem ajudar com informações valiosas sobre o comportamento do mesmo.

✓ Realização de ajustes simples *on-line* e ajustes complexos *off-line*. A enumeração de ações possíveis para sintonizar o sistema e a sua execução podem, em algumas situações, impactar muito negativamente o desempenho do sistema. Portanto, realizar este tipo de computação *on-line*, enquanto o sistema processa requisições, pode ser impraticável. Para não ocorrer esse tipo de impacto negativo, pode-se registrar a necessidade de tomar a ação para posteriormente, executá-la em *off-line*, ou seja, em um momento em que o processamento de requisições do sistema não seja comprometido.

Os itens citados anteriormente servem para ilustrar os princípios que podem ser aplicados na construção de componentes de auto sintonia em geral. Outras técnicas relevantes podem ser consultadas em [Barrientos et al., 2004].

[Lifschitz et al. 2004] sugerem duas direções principais para a pesquisa em auto sintonia-fina: global e local em concordância com o primeiro item abordado durante a descrição dos princípios.

A auto sintonia-fina global possui como objetivo a construção de sistemas que possam, automaticamente, manter um equilíbrio entre os recursos usados pelos componentes do SGBD para atingir um bom desempenho e vazão como um todo.

A auto sintonia-fina local tenta solucionar problemas específicos de desempenho de banco de dados, tais como: seleção e gerenciamento de índices, localização dos dados físicos, políticas de substituição de páginas no buffer entre outros. Os trabalhos de auto sintonia local concentram os seus esforços em resolver os seguintes problemas:

✓ Projeto físico: o problema consiste em escolher a melhor organização física para um dado esquema e uma carga de trabalho específica.

✓ Alocação de dados: dados N elementos de armazenamento, devemos determinar como podemos alocar e realocar dinamicamente fragmentos de arquivos ou relações nestes elementos de tal forma que o equilíbrio de carga seja o melhor possível, mesmo diante de mudanças nos padrões de acesso da carga de trabalho.

✓ Controle de carga: em um sistema que processa múltiplas transações de forma concorrente e que se utiliza de bloqueios (*locks*) para garantia das propriedades ACID (Atomicidade, Consistência, Isolamento e Durabilidade), o objetivo é regular o nível de multiprogramação do sistema de forma a evitar que a sua vazão (*throughput*) caia devido a conflitos de bloqueio (*thrashing* de contenção de sados).

✓ Substituição de páginas: o problema é manter em memória o conjunto de páginas mais populares do banco de dados, mesmo quando a carga de trabalho a que o sistema é submetido muda.

✓ Ajuste de *buffers*: procura-se estudar quantos *buffers* distintos devem ser configurados no banco de dados e de que forma devem ser distribuídos os objetos (tabelas e índices) para estes buffers para que o desempenho aumente.

✓ Refino de estatísticas: o objetivo é escolher quais estatísticas devem ser criadas no banco de dados e refinar estas estatísticas sem necessidade de executar comandos específicos para coleta no sistema.

4.6. (Auto) Sintonia-fina na Prática

Existem diversas ferramentas comerciais e acadêmicas para auxiliar o DBA na tarefa de (auto) sintonia-fina.

Pode-se considerar que as ferramentas de auto-sintonia de banco de dados envolvem três componentes principais: espaço de busca, modelo de custo e estratégia de enumeração [Bruno 2011].

✓ **Espaço de busca:** no contexto de sintonia fina de projeto físico, usando a estrutura de acesso de índice, considera-se como espaço de busca, o conjunto de índices candidatos para um determinado banco de dados e uma dada carga de trabalho. Em uma visão mais ampla, podemos considerar como sendo o conjunto de estruturas de acesso candidatas para uma base de dados de acordo com uma determinada carga de trabalho.

✓ **Modelo de custo:** é elaborado fora do otimizador. Ele avalia qual seria o custo de execução dos comandos que fazem parte de uma determinada carga de trabalho sobre uma configuração simulada durante a tarefa de sintonia fina. Quando os custos de execução são estimados sobre configurações hipotéticas, ou seja, que consideram estruturas sem estarem realmente materializadas, chamamos de camada de otimização *what-if*.

✓ **Estratégia de enumeração:** considera as limitações do ambiente em que o SGBD se encontra para enumerar configurações candidatas. Por exemplo, a estratégia pode considerar um espaço em disco limitado disponível para a criação de índices. Com a limitação de espaço físico, a estratégia pode simplificar o espaço de busca, passando a considerar apenas índices simples.

Considerando os resultados complexos da área e o aumento da sofisticação das máquinas de consultas e cargas de trabalho, as técnicas recentes de sintonia fina usam heurísticas para simularem configurações que sejam relevantes no espaço de busca analisado. Na próxima seção, cita-se alguns trabalhos que fazem uso de heurísticas.

4.6.1. Ferramentas

Existem diversas ferramentas acadêmicas (ex.: [Morelli et al. 2012], *DBX* [Almeida et al. 2015], *Outer-Tuning* [Almeida et al. 2018] e comerciais (ex.: *Automatic Database Diagnostic Monitor – ADDM* [Oracle 2018], *SQL Server Database Engine Tuning Advisor* [Microsoft 2017a]), 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.

A *DBX* é uma ferramenta para a manutenção automática do projeto físico em bancos de dados relacionais baseada em planos de execução hipotéticos (Figura 4.3 e Figura 4.4) [Almeida et al. 2015]. Essa ferramenta é implementada na linguagem Java e totalmente desacoplada de qualquer SGBD específico. Ela realiza a manutenção de índices e visões materializadas, reagindo dinamicamente às alterações na carga de trabalho. O diferencial da ferramenta é que ela considera tanto índices completos quanto índices parciais, além de detectar a fragmentação de um índice e sugerir a sua recriação ou remoção. As visões materializadas também são acompanhadas durante todo o seu ciclo de vida, ou seja, quando deixam de proporcionar benefícios, a ferramenta sugere a sua remoção.

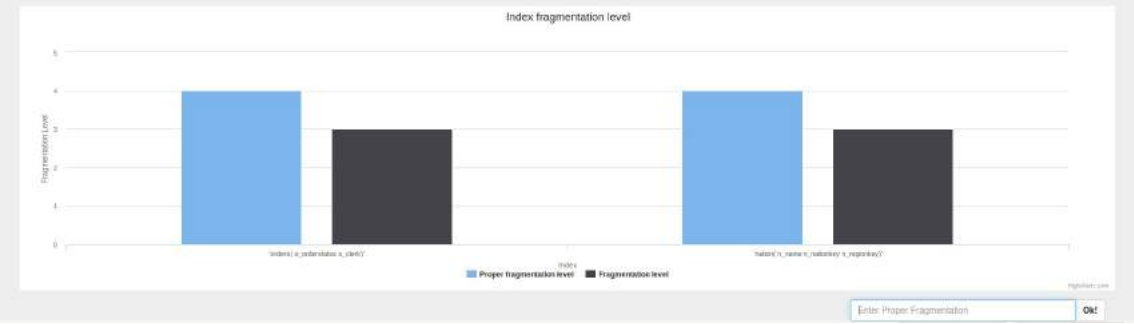

Figura 4.3 DBX – Gráficos sobre o nível de fragmentação de índices

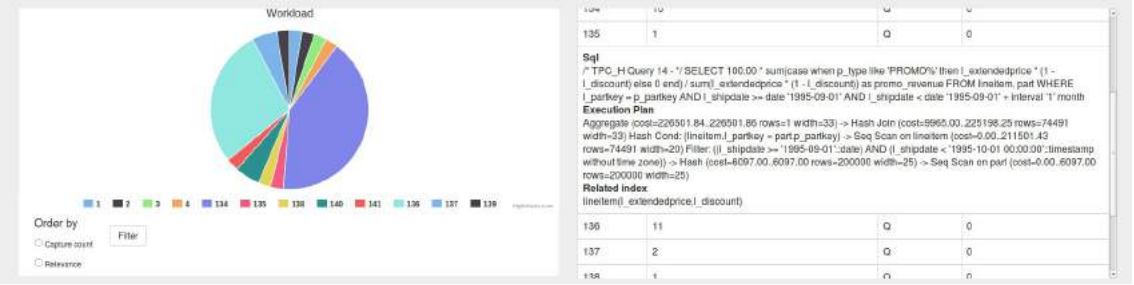

Figura 4.4 DBX – Detalhes sobre a carga de trabalho e índice sugerido para consulta

A *Outer-Tuning* é uma ferramenta que faz uso de uma ontologia de aplicação para apoiar a tarefa de (auto) sintonia-fina de banco de dados [Almeida et al. 2018]. Essa ferramenta pode ser instanciada ou estendida com qualquer heurística, desde que os conceitos utilizados por ela sejam definidos na ontologia de domínio. Atualmente, a ferramenta possui heurísticas que sugerem índices e visões materializadas e que podem ser aplicadas aos SGBDs PostgreSQL e Oracle.

Inicialmente, na Figura 4.5(A), o usuário do 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 de consulta ou atualização de dados é executado no banco de dados e a sua duração, em segundos (Figura 4.5(B)). Caso o usuário queira detalhes maiores sobre a execução do comando, ele pode verificar mais abaixo na

mesma tela (Figura 4.5(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 4.5(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 beneficiados 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.

Figura 4.5 Outer-Tuning – Telas

As ferramentas automáticas de sintonia-fina oferecidas pela Oracle para o SGBD Oracle 18c são: *Automatic Database Diagnostic Monitor – ADDM*, *SQL Tuning Advisor*, *SQL Access Advisor*, *SQL Plan Management* e *SQL Performance Analyzer* [Oracle 2018].

A *ADDM* 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-fina do SGBD, buscando diminuir o tempo perdido pelo servidor ao processar uma requisição do usuário, incluindo tempo de espera e CPU. Essa ferramenta provê recomendações de correção e quantifica os benefícios esperados. Na Figura 4.6 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%. Além disso, a figura apresenta o raciocínio da ferramenta para sugerir a recomendação.

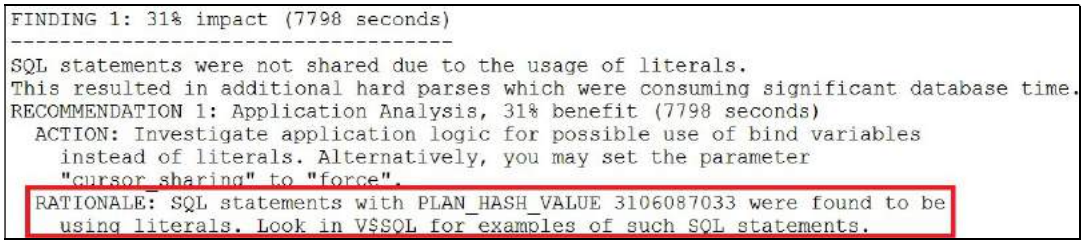

Figura 4.6 Exemplo de relatório gerado pela ferramenta ADDM

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 4.7) ao invés de literais (Figura 4.8).

Uso de variável *bind*

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

Figura 4.7 Exemplo de uso de variável *bind*

Uso de literal

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

Figura 4.8 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 *SQL Tuning Advisor* (Figura 4.9) detecta comandos SQL considerados problemáticos e recomenda melhorias para o comando. Essa ferramenta verifica se as estatísticas estão obsoletas ou se não existem; constrói perfis do SQL, ou seja, um conjunto de informações específicas para o comando SQL; analisa se algum caminho de acesso alternativo pode melhorar o desempenho de forma significativa; e identifica comandos SQL que estão sendo executados com planos de execução abaixo do ideal. As recomendações relatam as estatísticas dos objetos, sugerem a criação de índices novos, a reestruturação do comando SQL ou a criação do perfil do SQL.

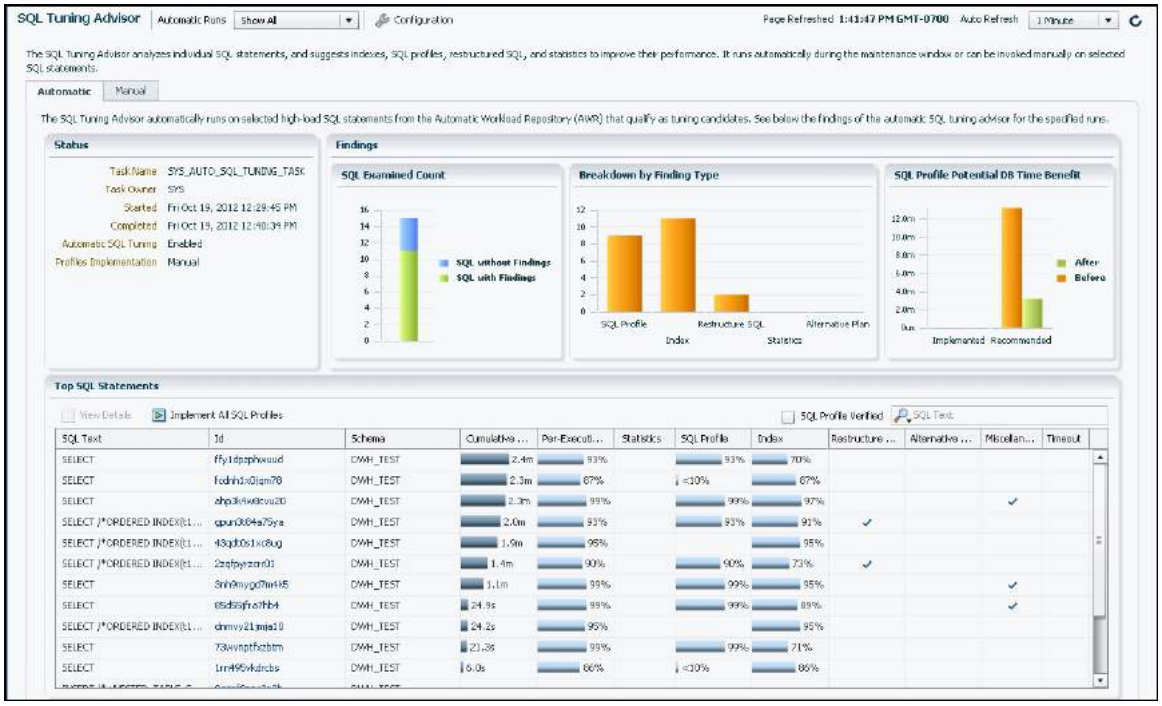

Figura 4.9 SQL Tuning Advisor – Tela [Oracle 2017]

O *SQL Access Advisor* analisa a carga de trabalho real ou pode atuar sobre uma carga de trabalho hipotética do esquema. Essa ferramenta analisa o conflito entre as possíveis escolhas (*trade-offs*). Ela avalia o custo-benefício entre o uso de espaço e o desempenho das consultas, recomendando a configuração de visões materializadas e índices (Figura 4.10).

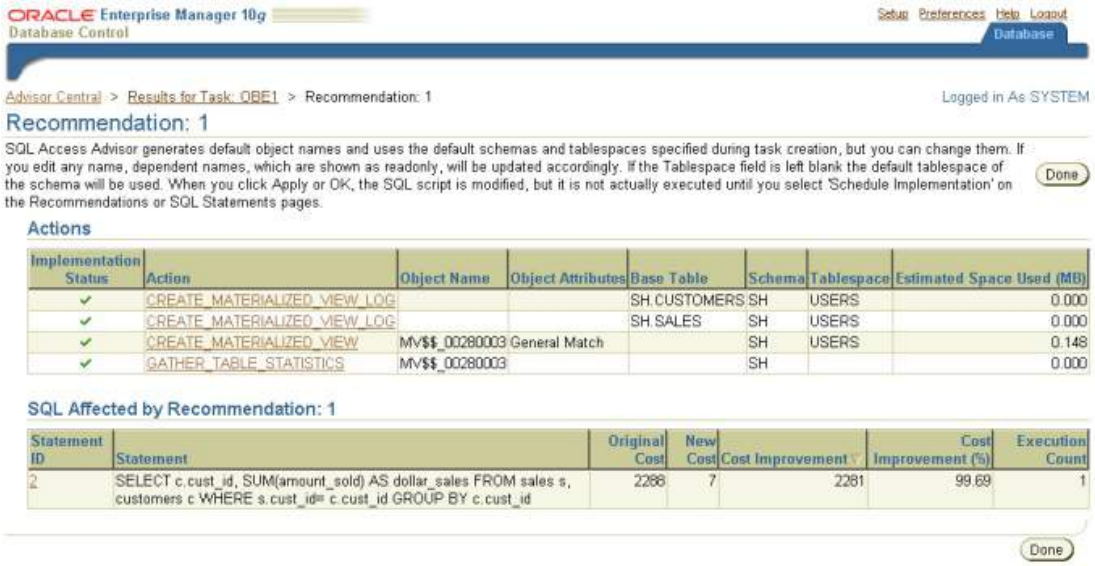

Figura 4.10 *SQL Access Advisor* - Exemplo da tela de recomendação [Oracle 2006]

O *SQL Plan Management* é um mecanismo de prevenção que permite que o otimizador gerencie os planos de execução, automaticamente, garantindo que o banco de dados usa somente planos conhecidos e já verificados (Figura 4.11).

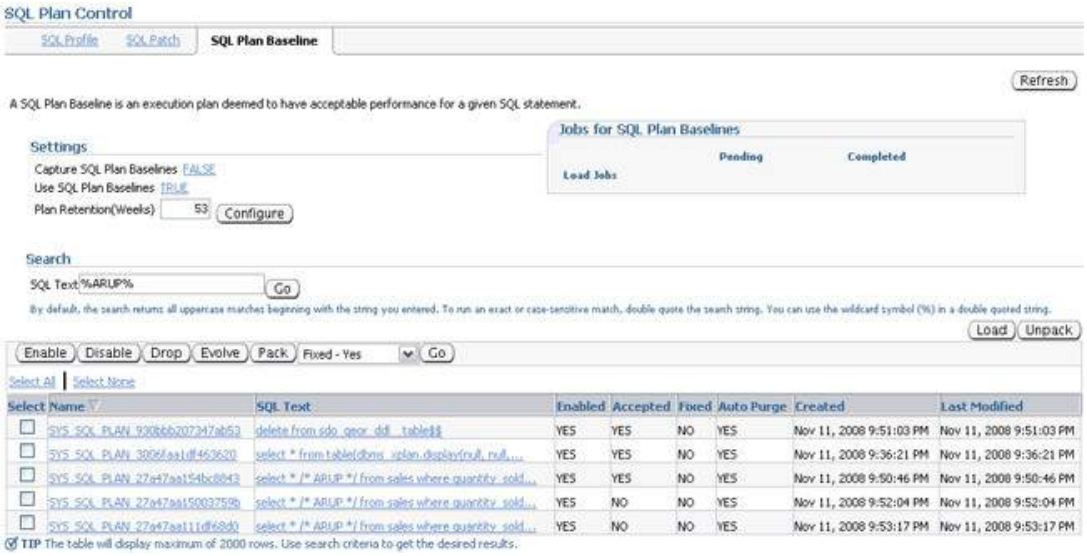

Figura 4.11 Baselines de planos de execução para serem usados pelo SQL Plan Management [Nanda 2009]

O *SQL Performance Analyzer* identifica o efeito de uma mudança na carga de trabalho SQL através da identificação de divergência de desempenho para cada comando SQL (Figura 4.12). Essa ferramenta apresenta informações sobre a tarefa, o conjunto de sintonia-fina que foi usado e se os comandos SQL encontraram algum erro. Além disso, exibe estatísticas globais baseada na comparação de métricas usadas para análises, a melhoria global na carga de trabalho e o impacto de regressão. De acordo com a Figura 4.12, tem-se que a melhoria sugerida não impacta negativamente em nenhum outro comando SQL, pois os itens textuais (item 1) e gráficos (item 2) referentes ao item de regressão (*Regression Impact*) constam zerados.

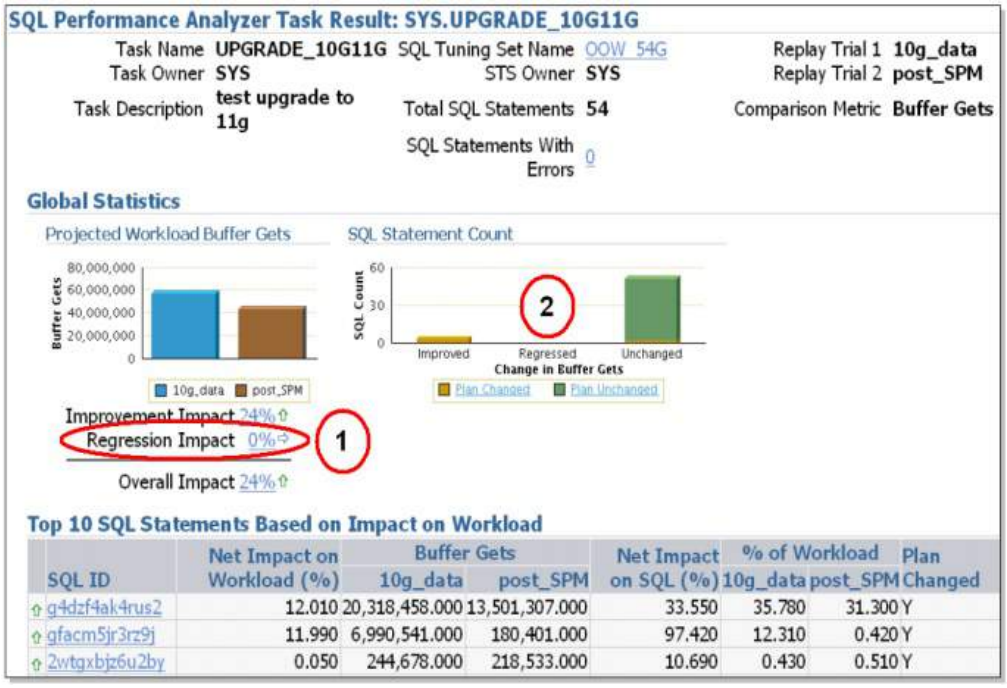

Figura 4.12 SQL Performance Analyzer - Exemplo de resultado de análise [Yagoub et al., 2007]

A ferramenta comercial da Microsoft para o SGBD SQL Server 2017 chama-se *Database Engine Tuning Advisor - DTA* [Microsoft 2017a]. Essa ferramenta analisa a carga de trabalho e sugere a criação de índices, visões indexadas ou partições de tabelas sem a necessidade de o usuário possuir entendimento sobre as estruturas do banco de dados ou detalhes internos do SQL Server (Figura 4.13).

Existem diversas outras ferramentas e abordagens de sintonia fina não citadas neste minicurso. As ferramentas citadas são apenas exemplos da existência de auxílios para o DBA. A maioria das ferramentas busca sugerir uma ou mais ações de sintonia fina e considera que o usuário seja um DBA, ou seja, saiba interpretar os resultados e as ações dessas ferramentas. As ferramentas comerciais citadas aqui não permitem qualquer extensão.

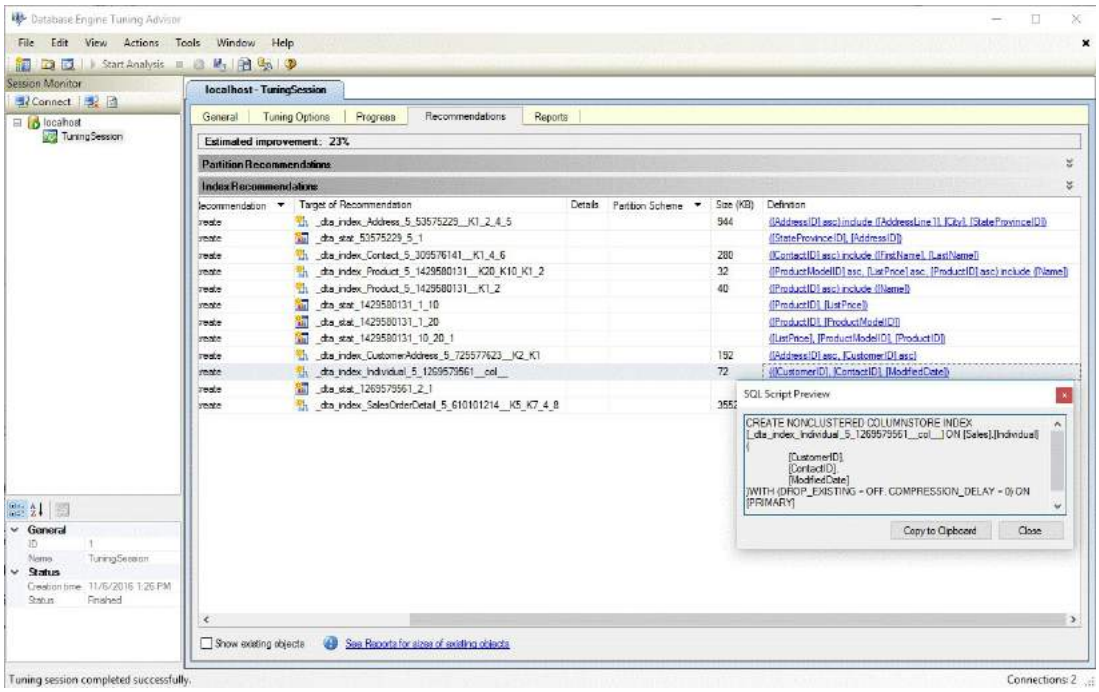

Figura 4.13 *Database Engine Tuning Advisor* – Exemplo de recomendação [Microsoft 2017b]

4.6.2. Estudos de Caso e Boas Práticas

Nesta subseção são descritos alguns exemplos de melhorias de desempenho no banco de dados e por consequência, relatadas algumas boas práticas.

Para todos os exemplos, usa-se o esquema de bancos de dados de uma locadora [Morelli, 2009] implementado no SGBD PostgreSQL 9.6.9[1] (Figura 4.14):

[1] https://www.postgresql.org/

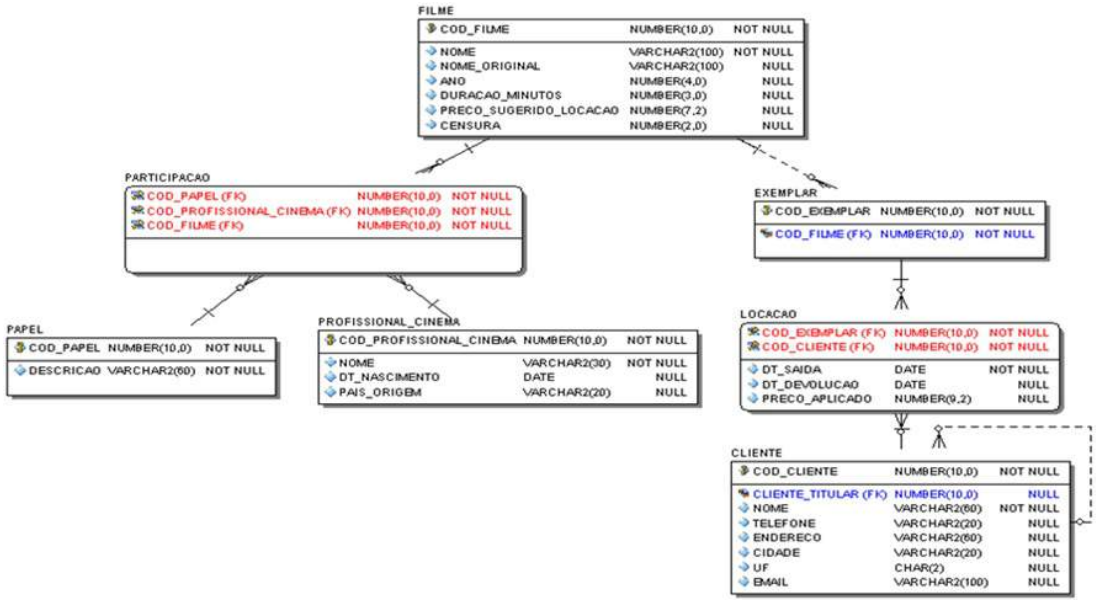

Figura 4.14 Esquema de Locadora [Morelli, 2009]

Para simular um grande volume de dados, cria-se uma tabela chamada de **HISTORICO_LOCACAO** com as mesmas características da tabela **LOCACAO**, mas contendo em torno de 15.000.000 de linhas a mais do que a existente (**LOCACAO**).

4.6.2.1 Uso do DISTINCT

A cláusula DISTINCT deve ser evitada. O DBA deve sempre avaliar a lógica a aplicação para verificar se o uso do DISTINCT é realmente necessário. No caso de consultas sobre uma única tabela, o DBA deve verificar se o resultado da consulta contém alguma coluna definida como chave única. Nesse caso, se existir, não há necessidade do uso do DISTINCT, visto que o comando retornará todas as linhas, pois sempre terá uma coluna com valor distinto em todas as linhas (chave única). Para consultas com junções, o DBA deve verificar o tipo de junção, as colunas da junção e as colunas do resultado.

Considerando a seguinte consulta (Figura 4.15):

```
SELECT distinct cod_exemplar, cod_cliente, dt_saida
FROM historico_locacao;
```

Figura 4.15 Consulta com uso do DISTINCT

O Plano de execução original dessa consulta, gerado pelo otimizador do SGBD é apresentado na Figura 4.16.

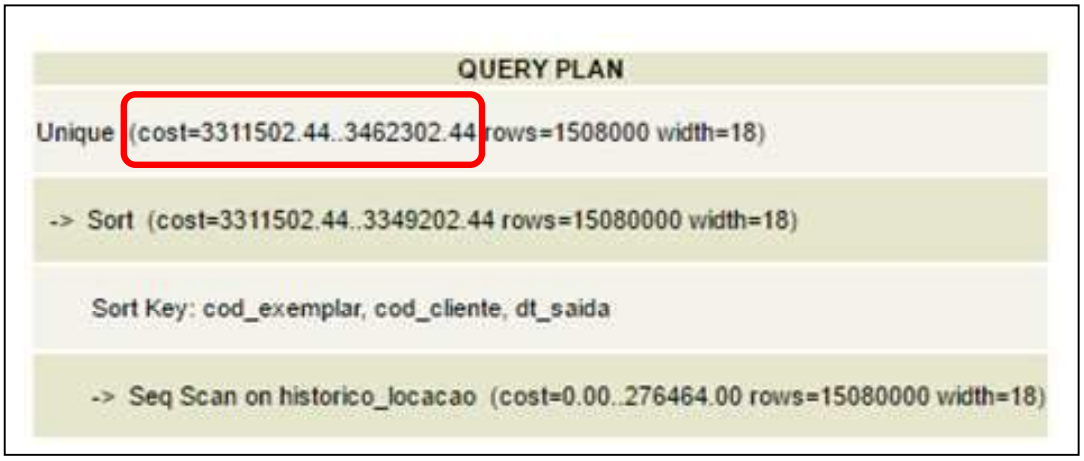

Figura 4.16 Plano de execução da consulta com uso do DISTINCT

Ao analisar consulta, nota-se que as colunas da cláusula de projeção (SELECT) resultante são as `COD_EXEMPLAR` e `COD_CLIENTE`. Essas colunas, segundo o esquema apresentado, corresponde as colunas que compõem a chave primária da tabela. Por definição, a chave primária de uma tabela no SGBD relacional deve ser única, logo, não é necessário usar a cláusula DISTINCT. Ele é totalmente desnecessário. Ao retirar a cláusula DISTINCT e solicitar novamente um plano de execução ao otimizador, obtém-se um custo total bem menor (Figura 4.17), sem as operações de ordenação.

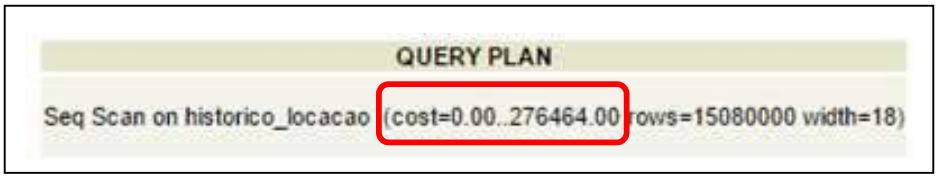

Figura 4.17 Plano de execução otimizado da consulta que usava DISTINCT

4.6.2.2 Uso do HAVING

Se a cláusula HAVING puder ser substituída pela cláusula WHERE, tal substituição deve ser realizada. Essa boa prática é indicada, pois a cláusula WHERE elimina linhas antes da operação de agrupamento e facilita a utilização de índices na comparação dos valores de busca.

Considerando a seguinte consulta (Figura 4.18):

```
SELECT max(preco_aplicado)
FROM historico_locacao
GROUP BY cod_exemplar
HAVING cod_exemplar = 15;
```

Figura 4.18 Consulta com uso do HAVING

O Plano de execução original dessa consulta, gerado pelo otimizador do SGBD é apresentado na Figura 4.19.

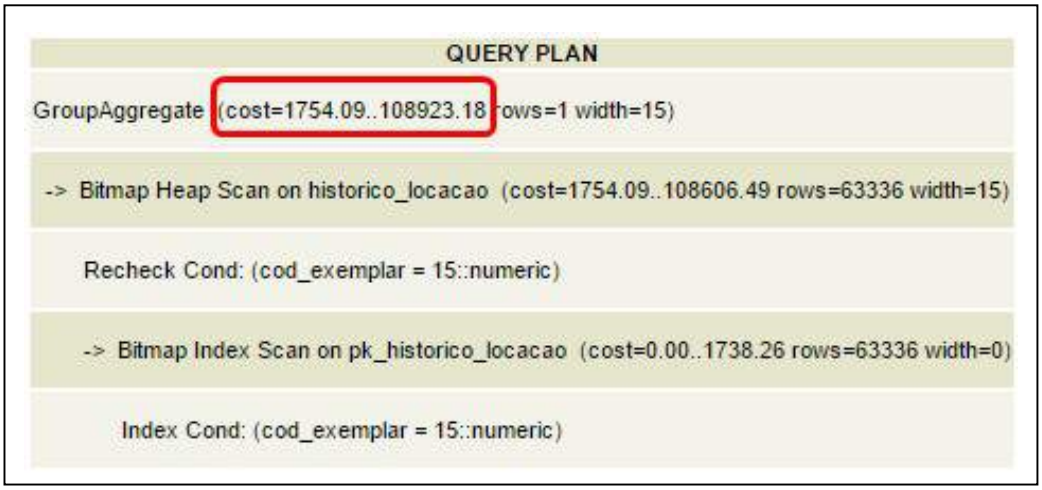

Figura 4.19 Plano de execução da consulta com uso do HAVING

Ao analisar o plano de execução, verifica-se que existe um custo muito alto de agregação, mesmo usando um índice *bitmap* da chave primária. Analisando a consulta, percebe-se que a restrição solicitada na cláusula HAVING não depende do agrupamento, ou seja, ela pode ser executada na cláusula WHERE, para cada linha da tabela. Ao realizar essa modificação, o otimizador gera um plano de execução com custo menor, apresentado na Figura 4.20.

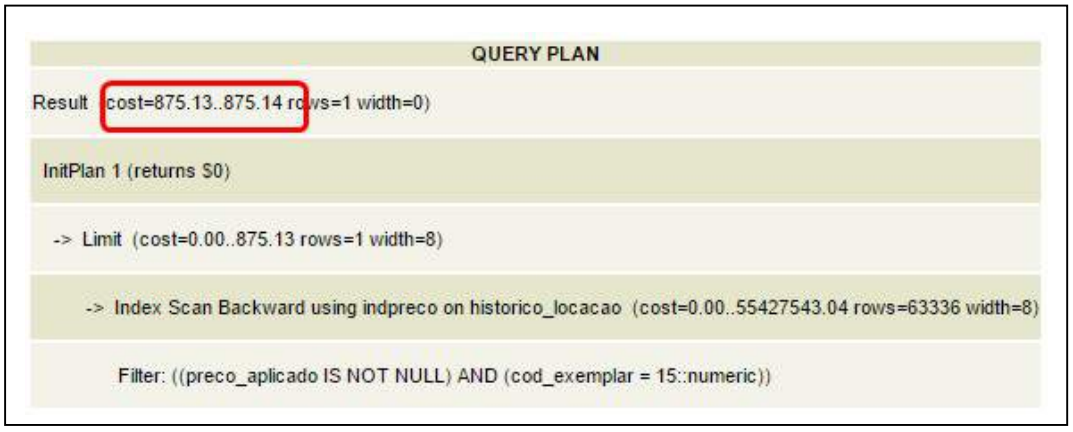

Figura 4.20 Plano de execução otimizado para a consulta que usava a cláusula HAVING

Conforme ilustrado nos exemplos anteriores, a técnica de reescrita de consultas pode ser usada para melhorar o desempenho do banco de dados, diminuindo o custo total de execução de consultas. É importante alertar que o uso de *frameworks*, por desenvolvedores, deve ser cuidadosamente avaliado. O *framework* pode facilitar o desenvolvimento de sistemas de informações, mas pode prejudicar o desempenho do banco de dados. Alguns *frameworks* trazem a facilidade de elaborar as suas próprias consultas, criando diversas subconsultas de forma desnecessária. Com isso, ao submeter essas requisições ao banco de dados, pode-se degradar o desempenho de forma

considerável. Ao analisar as consultas geradas pelo *framework*, o próprio desenvolvedor pode otimizá-la de forma a não prejudicar o banco de dados.

4.6.2.3 Uso de Estruturas Auxiliares

Além da técnica de reescrita de consultas, o DBA também pode optar por criar novas estruturas de acesso. Por exemplo, considera-se a consulta descrita na Figura 4.21.

```
SELECT c.nome, SUM(hl.preco_aplicado) Pago
FROM cliente c
      INNER JOIN historico_locacao hl
      ON c.cod_cliente = hl.cod_cliente
WHERE c.uf = 'ES'
      AND hl.preco_aplicado = 2
GROUP BY c.nome;
```

Figura 4.21 Consulta com junção

O plano de execução, gerado pelo otimizador do SGBD, é apresentado na Figura 4.22.

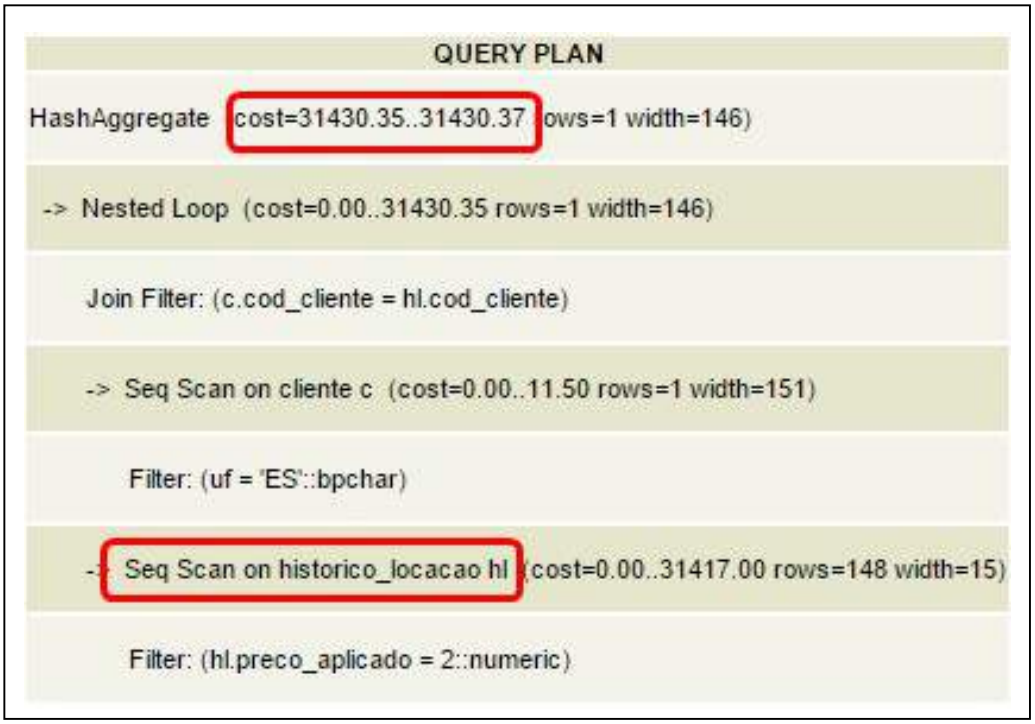

Figura 4.22 Plano de execução da consulta com junção

Analisando o plano de execução da consulta, observa-se que o otimizador realiza uma varredura (*Seq Scan*) na tabela `HISTORICO_LOCACAO`, para executar o filtro

sobre a coluna **PRECO_APLICADO** (`hl.preco_aplicado = 2`). Para diminuir o custo da varredura, o DBA pode optar pela criação de um índice sobre a coluna **PRECO_APLICADO**, desde que haja espaço e esse índice não prejudique as atualizações da tabela. O comando usado para a criação do índice se encontra na Figura 4.23.

```
CREATE INDEX indhist ON historico_locacao(preco_aplicado);
```

Figura 4.23 Comando SQL para criação de índice

Ao criar o índice no banco de dados e analisar o novo plano de execução da consulta (Figura 4.24), certifica-se de que o índice realmente beneficia a consulta e proporciona um decréscimo considerável no custo total do plano. Além disso, a criação do índice proporciona uma agilidade bem maior no tempo de resposta da consulta.

Figura 4.24 Plano de execução otimizado da consulta com junção

4.6.2.4 Uso de Subconsultas

O DBA deve estar atento ao uso de alternativa de subconsultas não correlacionadas, subconsultas correlacionadas e junções. Nas subconsultas não correlacionadas, não existe menção à consulta externa na subconsulta. As subconsultas correlacionadas

referenciam algum atributo de uma das tabelas da consulta externa. As junções permitem realizar a eventual recomposição de dados que foram separados por projeto.

Entre essas três alternativas, tem-se que as subconsultas não correlacionadas apresentam uma tendência pela não utilização de índices sobre a coluna selecionada na subconsulta (e que é a comparada entre a consulta e a subconsulta), enquanto as junções apresentam, usualmente, melhor desempenho.

Por exemplo, imagine uma consulta para verificar os exemplares de filmes que estão locados. Pode-se avaliar três formas de consultas possíveis. Uma forma poderia usar consulta não correlacionada (Figura 4.25), outra correlacionada (Figura 4.26) e outra por junção (Figura 4.27). Todas elas com os seus respectivos planos de execução e custos.

Nota-se que o otimizador gerou o mesmo plano de execução, com utilização de um algoritmo de junção *hash*, para os comandos com a presença das subconsultas correlacionada e não correlacionada. O plano de execução para o caso da utilização da junção também utiliza uma junção *hash*, mas a ordem de acesso as tabelas é invertida com relação à quando são utilizadas subconsultas. Além disso, tem-se que o custo da realização das consultas que utilizam as cláusulas IN e EXISTS é de, aproximadamente, 29.61 unidades, enquanto o custo do comando com a utilização de junção é de, aproximadamente, 3.34 unidades. Dessa forma, verifica-se que apenas com a reescrita da consulta, a mesma pode ter um ganho de mais de 85% em seu custo.

```
SELECT *
FROM exemplar e
WHERE e.cod_exemplar in
  (SELECT cod_exemplar
   FROM locacao l
   WHERE dt_devolucao
         is null);
```

QUERY PLAN
Hash Join (cost=29.61..32.81 rows=78 width=10)
Hash Cond: (e.cod_exemplar = l.cod_exemplar)
-> Seq Scan on exemplar e (cost=0.00..2.04 rows=104 width=10)
-> Hash (cost=28.63..28.63 rows=78 width=5)
-> HashAggregate (cost=27.85..28.63 rows=78 width=5)
Group Key: l.cod_exemplar
-> Seq Scan on locacao l (cost=0.00..24.08 rows=1508 width=5)
Filter: (dt_devolucao IS NULL)

Figura 4.25 Primeira forma de consulta – Exemplares locados

```
SELECT *
FROM exemplar e
WHERE exists
 (SELECT *
  FROM locacao l
  WHERE l.cod_exemplar
       = e.cod_exemplar
       and dt_devolucao
           is null);
```

```
QUERY PLAN
Hash Join  (cost=29.61..32.81 rows=78 width=10)
Hash Cond: (e.cod_exemplar = l.cod_exemplar)
-> Seq Scan on exemplar e  (cost=0.00..2.04 rows=104 width=10)
-> Hash  (cost=28.63..28.63 rows=78 width=5)
   -> HashAggregate  (cost=27.85..28.63 rows=78 width=5)
      Group Key: l.cod_exemplar
      -> Seq Scan on locacao l  (cost=0.00..24.08 rows=1508 width=5)
         Filter: (dt_devolucao IS NULL)
```

Figura 4.26 Segunda forma de consulta – Exemplares locados

```
SELECT *
FROM exemplar e,
  locacao l
WHERE l.cod_exemplar
     = e.cod_exemplar
     and dt_devolucao
         is null;
```

```
QUERY PLAN
Hash Join  (cost=3.34..48.16 rows=1508 width=42)
Hash Cond: (l.cod_exemplar = e.cod_exemplar)
-> Seq Scan on locacao l  (cost=0.00..24.08 rows=1508 width=32)
   Filter: (dt_devolucao IS NULL)
-> Hash  (cost=2.04..2.04 rows=104 width=10)
   -> Seq Scan on exemplar e  (cost=0.00..2.04 rows=104 width=10)
```

Figura 4.27 Terceira forma de consulta – Exemplares locados

4.7. Conclusão

Este minicurso apresenta uma discussão sobre os principais conceitos envolvidos na (auto) sintonia-fina de bancos de dados relacionais. Além disso, apresenta-se algumas ferramentas acadêmicas e comerciais que podem auxiliar o DBA nessa tarefa considerada complexa. Porém parece claro que as ferramentas, isoladamente, não conseguem resolver todas as situações práticas e o papel do DBA nas organizações continua fundamental no dia a dia. Cabe ao DBA saber se adaptar para utilizar mais e melhor as ferramentas de apoio e poder dedicar mais tempo às atividades mais complexas que são mais dependentes do conhecimento do negócio [Morelli and Lifschitz, 2004].

Como uma forma de demonstrar casos práticos que possam ser aplicados para a melhoria de desempenho de bancos de dados e por consequência, de sistemas de informações, apresenta-se exemplos reais de melhorias com uso de técnicas de sintonia-fina.

Os profissionais especializados na atividade de sintonia-fina de bancos de dados precisam conhecer bem desde a etapa de projeto lógico do banco de dados até estruturas de armazenamento e índices disponíveis. Em particular, sabe-se que há, por vezes, dificuldades existentes exclusivamente por desconhecimento de fundamentos básicos dos sistemas de bancos de dados. Por exemplo, pode ser interessante modificar estruturas de tabelas na base de dados de estruturas *heap* para estruturas de tipo *hash* caso o padrão de acesso aos dados seja mais adequado para acessos diretos (ou aleatórios). Em alguns SGBDs, isso é realizado com um simples comando (como o *modify* do Ingres) e os ganhos de desempenho são notáveis. Entretanto, o desconhecimento do comando do SGBD e, principalmente, do princípio de independência de dados (mudanças no nível físico não devem impactar o nível conceitual ou lógico), faz com que tarefas relativamente simples se tornem complicadas e que sistemas de banco de dados fiquem pouco eficientes desnecessariamente.

Espera-se que o participante tenha uma visão geral sobre (auto) sintonia-fina de bancos de dados, tema importante para a área de bancos de dados e para todos os sistemas de informações que fazem uso de bancos de dados, reforçando a formação básica dos profissionais e estudantes da área.

Referências

Almeida, A. C., Brayner, A., Filho, J. M. S. M., Lifschitz, S., Oliveira, R. P. (2015) "DBX: Um Framework para Auto-sintonia fina baseado em planos hipotéticos", 30º Simpósio Brasileiro de Bancos de Dados - SBBD - Demos and Applications Session, p. 149 – 154, Petrópolis, RJ, Brasil.

Almeida, A. C., Haeusler, E. H., Lifschitz, S., Oliveira, R. P., Schwabe, D. (2018) "Outer-Tuning: sintonia fina automática baseada em ontologia", 33º Simpósio Brasileiro de Bancos de Dados - SBBD - Demos and Applications Session, p. 29 – 34, Rio de Janeiro, RJ, Brasil.

Barrientos, A. (2004). "Uma arquitetura para auto-sintonia global de sgbds baseada em agentes". Dissertação de Mestrado do Departamento de Informatica, Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio).

Berman, J. J. (2013) "Principles of big data: preparing, sharing, and analyzing complex information". Morgan Kaufmann, Elsevier.

Bruno, N. (2011) "Automated Physical Database Design and Tuning". CRC Press.

Elmasri, R., Navathe, S. (2016) "Fundamentals of database systems". 7th Edition, London: Pearson.

Ferrari, D. (1978). "Computer Systems Performance Evaluation". Prentice Hall.

Lifschitz, S., Milanés, A., Salles, M. (2004) "Estado da arte em auto-sintonia de SGBD relacionais. ", Relatório Técnico, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), MCC35.

Microsoft (2017a) “Database Engine Tuning Advisor”, https://docs.microsoft.com/pt-br/sql/relational-databases/performance/database-engine-tuning-advisor?view=sql-server-2017, Setembro.

Microsoft (2017b) “Columnstore index recommendations in Database Engine Tuning Advisor (DTA)”, https://docs.microsoft.com/en-us/sql/relational-databases/performance/columnstore-index-recommendations-in-database-engine-tuning-advisor-dta?view=sql-server-2017, Setembro.

Morelli, E. (2009) “Oracle DBA Essencial Vol. 1 - SQL”. Editora Brasport.

Morelli, E., Lifschitz, S. (2004) “Auto-sintonia em SGBDs: o fim do DBA?”. SQL Magazine 8.

Nanda, A. (2009) “Use SQL plan management in Oracle Database 11*g*to optimize execution plans”, https://blogs.oracle.com/oraclemagazine/baselines-and-better-plans, acessado em Outubro, 2018.

Oracle (2006) “Performance Tuning using the SQL Access Advisor”, https://www.oracle.com/technetwork/database/manageability/twp-manage-tuning-using.pdf, acessado em Outubro, 2018.

Oracle (2017) “Configuring the Automatic SQL Tuning Advisor”, https://docs.oracle.com/database/121/ADMQS/GUID-AB077F2A-033B-4881-A916-ADBA1B20A8AC.htm#ADMQS1038, Setembro.

Oracle (2018) “SQL Tuning Guide”, https://docs.oracle.com/en/database/oracle/oracle-database/18/tgsql/introduction-to-sql-tuning.html#GUID-B770B7C7-2D0D-4E8A-9A29-811A5E2A9D4F, acessado em Setembro, 2018.

Shasha, D., Bonnet, P. (2003) “Database tuning – Principles, experiments, and troubleshooting techniques”. Morgan Kaufmann publishers.

Yagoub, K., Gongloor, P. (2007) “SQL Performance Analyzer”, https://www.oracle.com/technetwork/database/database-technologies/performance/spa-white-paper-ow07-132047.pdf, acessado em Outubro, 2018.

Autores

Ana Carolina Brito de Almeida é professora adjunta da Universidade do Estado do Rio de Janeiro (UERJ), alocada no Departamento de Informática e analista judiciário com especialidade em Informática no Tribunal Regional Federal da 2ª Região (TRF2), alocada na Seção de Administração de Bancos de Dados. Realizou pós-doutorado com ênfase em *Big Data* na Universidade Federal do Rio de Janeiro (UFRJ) (Out/2014 a Abr/2015). Doutora em Informática com especialização em *Tuning* de Bancos de Dados pela Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio) (2008 a 2013). Mestre em Sistemas e Computação com especialização em Bioinformática pelo Instituto Militar de Engenharia (IME/RJ) (2004 a 2006). Pesquisadora na área de bancos de dados com ênfase em: (i) sistemas contemplando (auto) sintonia de bancos de dados e (ii) ontologias. Já foi consultora em banco de dados e ministrou cursos na Universidade Petrobras. Detalhes em `http://lattes.cnpq.br/8306729029606464`.

Sérgio Lifschitz é professor associado da Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), alocado no Departamento de Informática. Doutor em Informática com especialização em Bancos de Dados pela École Nationale Supérieure des Télécommunications, ENST Paris, França (Out/1990 a Jan/1994). Mestre em Engenharia Elétrica (Especialização: Sistemas e Geometria Computacional - Mar/1987 a Ago/1990) e Engenheiro Eletricista (Especialização em Sistemas - Mar/1981 a Julho/1986), ambos pela PUC-Rio. Pesquisador na área de bancos de dados com ênfase em (i) computação autônoma e sistemas contemplando auto-sintonia e auto-gerenciamento e (ii) ferramentas e sistemas de gerência de dados para aplicações em bioinformática. Detalhes em `http://lattes.cnpq.br/8164403687403639`.