1
Introdução

Esta dissertação estuda como prover mecanismos automáticos para a sintonia de índices em sistemas de bancos de dados relacionais. Nosso trabalho de pesquisa faz parte de um conjunto de iniciativas para auto-sintonia de sistemas de bancos de dados [36]. A seguir, iremos comentar brevemente os objetivos e o contexto de desenvolvimento de pesquisa em auto-sintonia de sistemas gerenciadores de bancos de dados (SGBDs), revisar alguns conceitos referentes a índices em bancos de dados, descrever o problema endereçado por esta dissertação, e explicitar como o restante do texto está organizado.

Contexto

A sintonia de bancos de dados (*database tuning*) é a atividade de tornar a execução de uma aplicação de bancos de dados mais rápida [53]. Por mais rápida podemos entender que a aplicação terá maior vazão (*throughput*) em termos das transações que executa por unidade de tempo ou que algumas de suas transações terão menores tempos de resposta.

Durante a atividade de sintonia, o administrador do banco de dados (DBA) ou o especialista em desempenho podem ter que modificar parâmetros do SGBD e do sistema operacional, criar novas estruturas de dados na base de dados, ou alterar a forma como a aplicação foi escrita. Em nossa visão, o objetivo da sintonia deve ser o de obter o melhor desempenho possível para a aplicação sem realizar melhorias nos recursos de *hardware* disponíveis [36][1].

O profissional que realiza a sintonia deve possuir conhecimento não só sobre o SGBD, mas também sobre como este interage com o seu ambiente.

[1] Shasha e Bonnet [53], por outro lado, pregam que o objetivo da sintonia é o de obter um desempenho aceitável para a aplicação, conforme percebido pelos seus usuários, mesmo que isto implique a adição de novo *hardware*.

Por ter que reunir diversos conhecimentos e experiência prática, a curva de aprendizado deste tipo de profissional tende, em geral, a ser longa.

Por outro lado, estimativas para o número de profissionais de tecnologia de informação necessários globalmente para dar suporte a um bilhão de usuários e milhões de organizações conectadas através da Internet - uma situação a que pode se chegar na próxima década - chegam a impressionantes 200 milhões [33]. Os avanços no *hardware* e o desenvolvimento da Web levaram pesquisadores da área de bancos de dados a crer que ocorrerá um crescimento de ordens de magnitude na quantidade de sistemas de bancos de dados instalados e no volume de dados gerenciado [2].

As arquiteturas utilizadas para os atuais SGBDs demandam um trabalho de sintonia que tende a se tornar impraticável no longo prazo. O custo relativo dos recursos computacionais e da atenção humana se inverteu, sendo agora a atenção humana o recurso mais precioso [2]. Nos últimos anos, diversos trabalhos de pesquisa (por exemplo, [5, 10, 37, 42]) proporam automatizar aspectos da sintonia de bancos de dados, buscando tornar os sistemas capazes de auto-sintonia.

Um estudo de trabalhos sobre auto-sintonia de sistemas de bancos de dados relacionais foi conduzido no contexto desta dissertação e documentado em [36]. Referimos o leitor a este estudo para uma visão aprofundada de trabalhos correlacionados ao nosso. Nesta dissertação, iremos focar nossa atenção na busca de soluções para o problema específico de auto-sintonia de índices em SGBDs relacionais.

Índices

Antes de abordamos o problema tratado por esta dissertação, cabe realizar uma revisão de conceitos relativos à criação de índices em sistemas de bancos de dados relacionais. Nossa revisão é baseada na apresentação de Shasha e Bonnet [53].

Um índice é uma organização de dados que permite acelerar o processamento de consultas interessadas em um ou mais registros de uma tabela. Uma chave de um índice é uma seqüência de atributos. O índice define um mapeamento entre os valores desta seqüência de atributos e seus registros correspondentes. Em sistemas relacionais, a estrutura mais comumente utilizada para representar índices são as árvores B+ (para informações sobre árvores B+ e algumas variantes, veja [16]). Nesta dissertação, iremos restringir nossa atenção a este tipo de estrutura de indexação.

Uma árvore B+ é uma árvore balanceada cujas folhas contêm uma seqüência de pares chave-ponteiro. As chaves são ordenadas pelo seu valor. Podemos ver um exemplo deste tipo de estrutura na Figura 1.1.

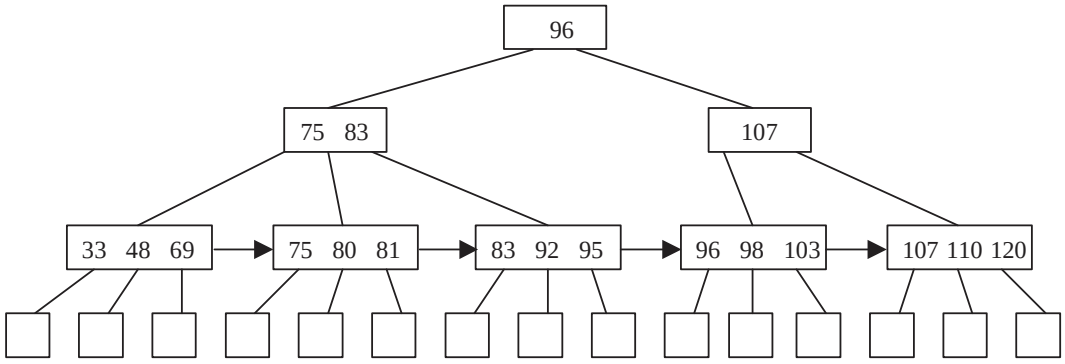

Figura 1.1: Exemplo de árvore B+

Uma característica que torna as árvores B+ interessantes para o uso em sistemas de bancos de dados é o armazenamento de diversas chaves por nó, criando uma árvore de grau alto. Isto implica que a altura da árvore tende a ser pequena mesmo para um conjunto considerável de chaves. Tipicamente, é possível indexar milhões de chaves mantendo a altura da árvore inferior a três. Como a obtenção de cada nó é feita por um acesso a disco, podemos consultar informações com uma determinada chave após alguns poucos acessos.

Existem algoritmos eficientes que permitem inserções e remoções de chaves de árvores B+. Estes algoritmos eventualmente precisam fazer reestruturações locais na árvore para manter sua propriedade de balanceamento. A quantidade de reestruturações feitas por estes algoritmos durante uma atualização dependem do quanto cada nó da árvore está preenchido em relação à sua capacidade total. Em algumas sistemas de bancos de dados, como o Oracle [45] e o Microsoft SQL Server [41], o DBA pode especificar um parâmetro de fator de preenchimento para as páginas dos índices representados como árvores B+. Este fator de preenchimento regula o quanto cada nó da árvore poderá ficar preenchido, facilitando ou dificultando atualizações.

Outro ponto interessante a observar é o de que as folhas da árvore são ordenadas pela chave do índice e ligadas por ponteiros. Isto permite que consultas que acessam um intervalo de valores da chave sejam processadas de forma também eficiente. O acesso às tuplas da relação é feito seguindo os ponteiros associados a cada chave no nível folha.

Os ponteiros das chaves no nível folha da árvore podem apontar tanto para os registros da tabela subjacente quanto para os blocos em que estes registros estão armazenados. Chamamos o índice de denso quando este se encaixa no primeiro caso e de esparso quando se encaixa no segundo.

Podemos ainda classificar os índices em primários e secundários. Um índice primário é aquele que determina que a ordem dos registros na tabela subjacente será idêntica à ordem das chaves contidas nas folhas do índice. Já um índice secundário é aquele em que não há relação entre a ordem das chaves nas folhas do índice e dos registros armazenados na tabela. Consultas por intervalos que acessam informações da tabela são mais eficientes quando realizadas com um índice primário.

Um índice primário pode ser denso ou esparso; já um índice secundário somente pode ser denso. Isto ocorre pois índices esparsos não mantêm um ponteiro para cada registro da tabela e sim para cada página. A localização dos registros dentro das páginas da tabela é feita levando em consideração a ordem em que estes estão dispostos, que deve ser a mesma ordem das chaves do índice.

Em alguns sistemas de bancos de dados, como por exemplo o PostgreSQL [46], é possível ainda criar índices completos ou parciais. Índices completos são aqueles que indexam todos os registros de uma determinada tabela. Já índices parciais indexam apenas um subconjunto dos registros definido por um predicado especificado na criação do índice. Índices parciais podem ser úteis quando é necessário indexar colunas que possuem distribuições de dados não-uniformes, eliminando a representação de valores com grandes quantidades de repetições.

Descrição do Problema

Uma atividade comumente realizada por administradores de sistemas de bancos de dados para acelerar o desempenho de consultas submetidas a um SGBD relacional é a seleção de índices sobre as tabelas presentes na base de dados. Para compreendermos melhor as questões envolvidas na escolha de índices, vamos iniciar nossa apresentação com um exemplo. Tomemos uma carga de trabalho que envolva os seguintes comandos SQL sobre uma tabela `Venda`:

```
(1) insert into Venda (prodNum, data, qtd, valor)
    values (4, current_timestamp, 20, 348);
```

```
(2) select prodNum, data, sum(valor) as total
    from   Venda
    where  valor > 1500000 and
           data between '20040101' and '20040131'
    group by prodNum, data;
```

Suponha, inicialmente, que os comandos são submetidos ao SGBD de forma concorrente e que temos uma freqüência de comandos do tipo (2) muito maior do que a de comandos do tipo (1). Neste tipo de cenário, um índice sobre as colunas de `valor` e `data` da tabela `Venda` pode trazer benefícios de desempenho no processamento das transações em que estes comandos estão inseridos. Como os comandos do tipo (1) são menos freqüentes, espera-se um pequeno custo decorrente da atualização do índice sobre a tabela de vendas[2]. Este custo de atualização pode ser amplamente compensado pelas vantagens de desempenho trazidas nas consultas, uma vez que o índice pode trazer grandes benefícios ao processarmos a restrição sobre `valor` e `data`.

Se o DBA decidir criar o índice, será necessário determinar o momento apropriado para esta atividade. Em sistemas como o PostgreSQL [46], a criação de um novo índice obtém bloqueios que impedem o processamento de transações concorrentes que desejam fazer escritas sobre a tabela base.

Se tivermos a situação inversa, com uma freqüência de comandos do tipo (1) muito superior do que a freqüência de comandos do tipo (2), a criação de índices sobre a tabela de vendas pode não ser recomendável. Neste cenário, podemos ter um custo significativo associado à atualização dos índices presentes sobre a tabela. A atualização dos índices é decorrente de comandos que realizam modificações sobre os dados da tabela, como inserções, remoções e atualizações que envolvem as colunas indexadas. Numa situação intensiva em atualizações, os benefícios trazidos pelos índices nas consultas podem não compensar os custos de manutenção envolvidos.

Podemos ter ainda uma situação em que as transações envolvendo os comandos dos tipos (1) e (2) estão particionadas no tempo [53]. Os comandos do tipo (1) podem ocorrer em uma carga noturna, enquanto as consultas do tipo (2) ocorrem durante o dia. Neste cenário, pode ser interessante remover os índices da tabela `Venda` antes da carga noturna e voltar a criá-los antes do período diurno de consultas.

[2]O custo de atualização de um índice pode variar de acordo com o valor definido para o fator de preenchimento de suas páginas. Nesta dissertação, não abordaremos a questão da sintonia do fator de preenchimento de páginas. Nosso foco está na escolha de índices adequados para a carga de trabalho processada pelo SGBD.

Estes exemplos ilustram alguns pontos importantes que devem ser considerados quando escolhemos os índices a criar ou remover de uma base de dados. São eles:

1. *Freqüência e tipos de comandos executados.* O processamento de cargas de trabalho que misturam atualizações e consultas, como é comum em sistemas de processamento *on-line* de transações (OLTP), dificulta a decisão de criação de novos índices. Para um melhor uso dos recursos computacionais disponíveis, devemos considerar se o custo de manutenção de cada índice será realmente compensado pelos benefícios que este traz em consultas.

2. *Escolha de quais tabelas e colunas devem ser indexadas.* Antes mesmo de conseguirmos determinar se criar um índice é benéfico, precisamos saber quais índices podem ser criados. O número de índices possíveis sobre uma base de dados tende a crescer rapidamente à medida que mais colunas e tabelas estão presentes [37]. Conforme veremos no Capítulo 2, este é um dos aspectos mais estudados na literatura sobre o problema de sintonia de índices.

3. *Periodicidade e perfis da carga de trabalho.* Algumas cargas de trabalho tendem a possuir variações sazonais em seus comandos. Isto pode trazer a necessidade de criar diferentes índices para atender aos distintos perfis e sazonalidades da carga de trabalho.

4. *Obtenção da carga de trabalho.* Apesar de a carga de trabalho ser um insumo básico para determinarmos se estamos indexando bem uma dada base de dados, raramente é uma tarefa simples obtê-la. Em outras propostas da literatura [36], esta atividade é deixada nas mãos do administrador do sistema de bancos de dados (DBA).

5. *Determinação do instante em que os comandos de criação ou remoção dos índices serão executados.* Como a criação ou remoção de índices podem implicar em bloqueios, devemos ser cautelosos para que transações importantes não sejam impedidas por estas atividades.

Os trabalhos hoje presentes na literatura, como [10, 37], abordam fundamentalmente os primeiros dois pontos acima. Há um esforço para a criação de ferramentas que apoiem o DBA na tarefa de seleção de índices, mas os trabalhos não propõem arquiteturas que possam eliminar completamente a intervenção humana da tarefa de sintonia de índices.

Em estudos anteriores do nosso grupo de pesquisa [40, 8, 42], aplicamos a abstração de agentes de *software* para aumentar a autonomia de SGBDs e implementar mecanismos para auto-sintonia. Esta dissertação estuda a construção de arquiteturas de agentes de *software* que permitam a completa automatização das atividades relacionadas à sintonia de índices em SGBDs relacionais. Apresentamos duas arquiteturas que lidam com diferentes graus de complexidade envolvidos no problema. Para ilustrar a utilidade de nossa proposta, implementamos uma dessas arquiteturas embutindo agentes em um SGBD de código fonte aberto, o PostgreSQL [46].

Na discussão acima sobre o problema de sintonia de índices, existe uma suposição básica importante. Em princípio, consideramos todas as transações envolvidas igualmente importantes para os usuários do sistema. Isto não é sempre verdadeiro na prática. Por exemplo, poderíamos ter uma restrição de tempo máximo de resposta para os comandos do tipo (2). Se este fosse o caso, a criação do índice que acelera a consulta poderia ser obrigatória, mesmo que não traga um benefício para a vazão do sistema como um todo. Esta dissertação não levará em conta este tipo de restrição sobre as transações processadas pelo sistema. Iremos supor que o objetivo da sintonia de índices é diminuir o consumo de recursos do sistema em termos da quantidade total de acessos a disco e, assim, aumentar a sua vazão. Alguns exemplos de trabalhos na área de auto-sintonia que levam em conta restrições de objetivos de desempenho por classes de transações são [39, 5].

Estrutura da Dissertação

O restante desta dissertação está organizado nos seguintes capítulos:

- O Capítulo 2 revisa trabalhos da literatura que estudam a auto-sintonia de SGBDs, em especial a seleção de índices, ou que abordam como agentes de *software* podem ser usados em conjunto com sistemas de bancos de dados.
- O Capítulo 3 apresenta nossa proposta de arquiteturas de agentes de *software* para auto-sintonia de índices em SGBDs relacionais. Discutimos a infra-estrutura que os SGBDs devem prover para a implementação das arquiteturas e mostramos como construir os agentes envolvidos.
- O Capítulo 4 discute a implementação realizada para uma das arquiteturas de agentes de *software* propostas. Apresenta, ainda, resultados

experimentais obtidos com a execução da arquitetura no SGBD PostgreSQL.

– O Capítulo 5, por fim, apresenta as conclusões de nosso estudo e indica algumas direções em que o mesmo pode ser estendido.