Uma Ferramenta Não-Intrusiva para a Manutenção Automática de Índices

José Maria Monteiro[1], **Sérgio Lifschitz**[1], **Ângelo Brayner**[2]

[1]Departamento de Informática – Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio)

[2]Centro de Ciências Tecnológicas – Universidade de Fortaleza (Unifor)

Resumo. *Este trabalho apresenta o "Dbx Automated Index Tuning", uma ferramenta não-intrusiva para a manutenção automática de índices. O Dbx é independente de SGBD e executa suas tarefas de forma contínua e sem intervenção humana. Dessa forma, o Dbx pode modificar o projeto físico corrente (criando, removendo ou recriando índices) reagindo a alterações na carga de trabalho. A implementação do protótipo foi realizada inteiramente em Java e de forma desacoplada do SGBD. As técnicas desenvolvidas apresentam baixa sobrecarga e levam em consideração vários aspectos importantes como as restrições de espaço de armazenamento, a eventual fragmentação dos índices e a manutenção de índices primários e secundários.*

1. Introdução

As aplicações de bancos de dados têm se tornado cada vez mais complexas e variadas. Estas podem ser caracterizadas pelo grande volume de dados e pela demanda elevada em relação ao tempo de resposta das consultas e à vazão (*throughput*) das transações. Neste contexto, uma das principais atividades realizadas pelos administradores de bancos de dados (*Database Administrators -DBAs*) para acelerar o desempenho da carga de trabalho (conjunto de consultas e atualizações) submetida a um SGBD (Sistema de Gerenciamento de Banco de Dados) consiste em manter um conjunto de índices sempre adequado. Contudo, esta tarefa requer um profundo conhecimento acerca dos detalhes de implementação dos SGBDs, das características dos dados armazenados, das aplicações e da carga de trabalho [Bruno and Chaudhuri 2007, Luhring et al. 2007].

Os principais fabricantes de SGBDs oferecem ferramentas para suportar a sintonia automática (ou auto-sintonia) de índices (e.g., [Bruno and Chaudhuri 2005]). A idéia é auxiliar os DBAs através da análise automática da carga de trabalho e, com base nesta análise, recomendar a criação ou remoção de índices. Entretanto, tais ferramentas adotam uma abordagem estática (*offline*) na solução deste problema e transferem para o DBA tarefas de sintonia importantes e decisões relativas à quando modificar o esquema e quais de fato são as modificações a serem efetuadas. Além disso, são soluções proprietárias, voltadas para SGBDs específicos e que exigem participação de DBAs especialistas.

Neste artigo, apresentamos a implementação de uma ferramenta não-intrusiva para a manutenção automática e contínua (*on-the-fly*) de índices denominada "*Dbx Automated Index Tuning*". O *Dbx* é completamente desacoplado do código do SGBD utilizado (denominado SGBD alvo), o que permite sua execução com qualquer SGBD. Além disso, dispensa interações com os DBAs. Sempre que necessário, o *Dbx* modifica o projeto físico corrente (criando, removendo ou reconstruindo índices) reagindo a alterações na carga de

trabalho. Vale destacar que as técnicas desenvolvidas para a implementação do Dbx apresentam baixa interferência na carga de trabalho total do SGBD e levam em consideração vários aspectos importantes como as restrições de espaço de armazenamento, a eventual fragmentação dos índices e a manutenção de índices primários e secundários.

O restante deste artigo está organizado conforme descrito a seguir. Na seção 2 discutimos as principais soluções, encontradas na literatura, para a manutenção automática de índices. A seção 3 apresenta a ferramenta proposta. Na seção 4 discutimos os testes de desempenho realizados e os resultados obtidos. A seção 5 conclui este trabalho e aponta direções para trabalhos futuros.

2. Motivação e Trabalhos Relacionados

Algumas iniciativas recentes apresentam descrições de protótipos que implementam algumas funcionalidades na direção da sintonia automática e contínua de índices. Alguns desses trabalhos procuram estender o SGBD PostgreSQL adicionando funcionalidades de auto-sintonia (por ex. [Salles and Lifschitz 2005, Morelli 2006, Schnaitter et al. 2006, Luhring et al. 2007]. Os fabricantes de SGBDs como DB2, Oracle e SQL Server também têm procurado investigar novas funcionalidades que possibilitam a manutenção automática do projeto físico, como por exemplo, o SQL Server 2005 [Bruno and Chaudhuri 2007].

Contudo, estas ferramentas são consideradas intrusivas. As soluções intrusivas são aquelas que exigem alterações no código do SGBD, ou seja, que estão fortemente acopladas ao código do SGBD. Uma ferramenta intrusiva pode até apresentar bons ganhos de desempenho em situações particulares mas sempre terá um custo mais alto de manutenção, além de potenciais incompatibilidades com novos *releases* dos SGBDs, o que pode comprometer sua evolução e aplicabilidade. Outra desvantagem importante consiste no fato dos componentes de sintonia automática executarem em conjunto com os demais módulos do SGBD (por exemplo, no mesmo *host*), o que pode gerar sobrecarga e comprometer o desempenho do SGBD. Por outro lado, as soluções não-intrusivas, como a ferramenta proposta neste artigo, são aquelas que não requerem modificações no código do SGBD, estando desacopladas deste código. Logo, as soluções não-intrusivas não são afetadas por atualizações na implementação do SGBD sendo, portanto, independentes em relação às novas versões.

3. O Dbx

De um modo geral, a arquitetura do Dbx propõe a realização da auto-sintonia de índices através da colaboração entre agentes de *software*, os quais utilizam *drivers* para obter e manipular informações do SGBD utilizado. De posse desses dados, os agentes utilizam heurísticas, que encapsulam o conhecimento de especialistas em sintonia de projeto físico, com a finalidade de manter um conjunto de índices sempre adequado.

Nesta arquitetura um *driver* consiste em uma interface Java que contém um conjunto de métodos abstratos. Assim, para cada SGBD que se deseja utilizar torna-se necessário instanciar estes *drivers*. Instanciar um determinado *driver* consiste em codificar uma classe Java que implementa a interface correspondente ao *driver*. Contudo, nossa ferramenta já fornece *drivers* instanciados para os SGBDs: PostgreSQL 8.2, Oracle 10g e SQL Server 2005.

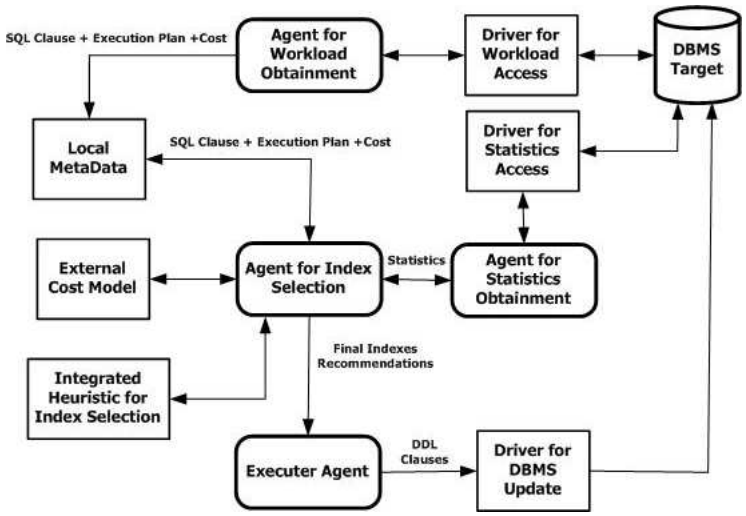

Figure 1. Arquitetura do Dbx

Os principais componentes desta arquitetura são:

- *Agent for Workload Obtainment (AWO):* Este agente captura periodicamente a carga de trabalho submetida ao SGBD alvo. Para isso, o agente consulta a metabase do SGBD e recupera as cláusulas SQL que foram ou estão sendo executadas, juntamente com seus respectivos planos de execução e estimativas de custo. Em seguida, cada *tripla* <cláusula SQL, plano de execução, custo> capturada é armazenada na metabase local (*LM*).
- *Local MetaData (LM):* A metabase local armazena a carga de trabalho capturada pelo agente *AWO* e um conjunto de informações acerca dos índices gerenciados pela ferramenta. Para cada índice são armazenados: um identificador (*id*), o tipo do índice (primário ou secundário), o nome da tabela, as colunas que compõem o índice, o estado do índice (real ou hipotético), o custo estimado de criação do índice e o benefício acumulado do índice, ou seja, quanto o índice contribui (caso real) ou poderia contribuir (caso hipotético) para reduzir o custo da carga de trabalho como um todo. Vale ressaltar que um índice hipotético, ao contrário dos índices reais, existe apenas na metabase local.
- *Driver for Workload Access (DWA): Driver* que permite recuperar a carga de trabalho submetida ao SGBD alvo.
- *Agent for Index Selection (AIS):* Este agente consulta periodicamente a metabase local para verificar se novas *triplas* <cláusula SQL, plano de execução, custo> foram adicionadas. Em caso afirmativo, para cada nova *tripla* adicionada, o agente aplica uma heurística integrada para seleção e acompanhamento de índices (*HISAI*). Esta heurística determinará os melhores índices (reais ou hipotéticos) para cada *tripla* analisada e seus respectivos benefícios (ou seja, quanto cada índice contribui ou poderia contribuir para reduzir o custo da cláusula SQL, especificamente).
- *External Cost Model (ECM):* Modelo de custos externo, ou seja independente de SGBD. Este modelo de custos é utilizado pela heurística *HISAI* e possibilita estimar, por exemplo, o custo de uma busca seqüencial, o custo de uma seleção utilizando um índice primário, etc.

- *Agent for Statistics Obtainment (ASO):* Este agente acessa o SGBD alvo e recupera informações estatísticas, tais como o número de páginas ocupadas por uma tabela, a quantidade de *tuplas* de uma tabela, a altura de um índice, etc. Estas informações são requisitadas e utilizadas pelo agente *AIS*.
- *Driver for Statistics Access (DSA): Driver* que permite recuperar as estatísticas do SGBD alvo.
- *Executer Agent (EA):* Este agente é responsável pelas atualizações no esquema do SGBD alvo, isto é, pela criação, remoção ou re-organização de índices. O agente executor recebe uma recomendação do agente *AIS* e executa esta recomendação no SGBD alvo.
- *Driver for DBMS Update (DDU): Driver* responsável por executar os comandos necessários para atualizar o projeto físico do SGBD alvo.

A abordagem de funcionamento utilizada pelo Dbx segue o ciclo clássico de auto-sintonia: Observação, Predição e Reação, como descrito inicialmente em [Weikum et al. 1994]. Estas fases são continuamente executadas com a finalidade de monitorar o comportamento corrente do sistema, tomar decisões sobre o comportamento futuro e, caso mudanças sejam necessárias, aplicar estas mudanças a fim de alterar as propriedades e o comportamento do sistema. A seguir descrevemos cada uma dessas fases:

- *Observação:* Nesta fase, a carga de trabalho submetida ao SGBD é capturada e armazenada em uma metabase (através do agente *AWO*). Em seguida, a carga é analisada, utilizando-se a heurística *HISAI*, a fim de se determinar os melhores índices e seus respectivos benefícios (tarefa realizada pelo o agente *AIS*). Além disso, acompanha-se a utilização dos índices reais a fim de monitorar a sua utilidade e a necessidade de re-organização (*reindex*).

  A heurística *HISAI* analisa cada *tripla <cláusula SQL, plano de execução, custo>* capturada pelo agente *AWO*. As informações armazenadas, para cada tripla, permitem fazer inferências sobre as características e custos de planos de execução alternativos e equivalentes que poderiam ser obtidos caso variássemos o projeto físico, por exemplo, criando ou removendo determinados índices.

  A idéia básica da *HISAI* consiste em procurar substituir um determinado sub-plano p, pertencente ao plano de execução original, por um sub-plano alternativo e equivalente $p^{'}$, obtido através da utilização de índices hipotéticos, porém com custo menor, resultando assim em um plano alternativo válido e equivalente ao plano original.

  O custo do sub-plano original p ode ser obtido a partir da metabase local. Já o custo o custo do sub-plano alternativo $p^{'}$ pode ser estimado através do modelo de custos externo. Logo, podemos inferir quanto o desempenho do plano de execução original pode ser melhorado (ou degradado) se substituíssemos um determinado sub-plano p por um outro sub-plano alternativo $p^{'}$.

- *Predição:* Esta fase busca obter uma configuração de índices ótima, ou seja, um conjunto de estruturas de índices que maximize o benefício para a carga de trabalho como um todo, levando em consideração as restrições de espaço físico.

  Esta fase é iniciada sempre que o benefício acumulado de um índice hipotético k qualquer ultrapassar o seu custo estimado de criação, indicando que a existência

do índice iria contribuir para diminuir o custo de processamento da carga de trabalho submetida ao SGBD, ou quando o benefício acumulado de um índice real k' alcança um valor negativo que supera, em módulo, o seu custo de criação, o que indica que a existência de k' não está contribuindo para diminuir o custo de processamento da carga de trabalho.

- *Reação:*

  Ao final da fase de predição, caso seja detectada a necessidade de mudanças na configuração de índices ou a existência de índices fragmentados, a fase de reação é iniciada. Assim, obtida a nova configuração de índices, torna-se necessário: (i) criar fisicamente os índices hipotéticos que pertencem à esta configuração mas que ainda não existem fisicamente, (ii) remover os índices reais que não pertencem à nova configuração de índices e (iii) re-organizar os índices reais que pertencem à nova configuração de índices mas que se encontram fragmentados.

4. Resultados Experimentais

A fim de mensurar a eficácia de nossa ferramenta utilizamos o *benchmark* TPC-H, o qual consiste em um *benchmark* voltado para aplicações de suporte a decisão. O TPC-H é formado por um conjunto de 23 consultas *ad-hoc* e possui uma estrutura padrão composta por oito tabelas. Destas, seis são tabela dimensionais (*Region*, *Nation*, *Supplier*, *Part*, *Customer* e *Partsupp*) e duas de fatos (*Orders* e *Lineitem*). Dentre as 23 consultas do TPC-H escolhemos a 6 mais significativas, ou seja, aquelas que apresentam tempo de resposta maiores, justificando a utilização das estruturas de índices [Morelli 2006]. Para a realização dos testes foi utilizada uma base de dados de 1G, onde a tabela *Lineitem* tem 6.000.000 de *tuplas*, por exemplo. O ambiente de execução utilizado foi uma estação *Athlon* 64 x2 de 2Ghz, com 2G *ram* e 250G de Hd, com SO *Windows XP*. O SGBD utilizado nos testes foi o PostgreSQL 8.2.

Como medida da eficiência de nossa solução utilizamos como referência o conjunto dos índices definidos para o *benchmark* TPC-H (Tabela 1). Assim, quanto mais próximo deste grupo estiver a coleção de índices mantida pela abordagem proposta, maior será sua eficiência da solução proposta.

Cada uma das seis consultas escolhidas foi executada em um ambiente desprovido de índices, ou seja, antes de cada execução de uma determinada consulta eliminamos os índices existentes. A Tabela 1 apresenta os índices criados automaticamente pela nossa implementação e a quantidade de execuções do comando SQL que foram necessárias para que nossa ferramenta decidisse pela criação do índice.

O índice a mais, criado por nossa abordagem (sobre a tabela *Part*, consulta 17) e não sugerido pelo *benchmark* TPC-H proporcionou ganhos expressivos. O custo caiu 62,62% e a sua duração obteve um ganho da ordem de 61,45%. Este valor foi obtido executando-se a consula 17 primeiramente sem e depois com a presença do referido índice.

5. Conclusões e Trabalhos Futuros

Neste trabalho, apresentamos uma ferramenta não-intrusiva para o problema da manutenção automática e *on-the-fly* de índices, denominada *Dbx*. A ferramenta proposta pode ser utilizada com qualquer SGBD e executa de forma contínua e independente de

Consulta	Tabela	Campos	TPC-H	Execuções
1	Lineitem	L_shipdate	Sim	5
2	Partsupp	Ps_partkey	Sim	1
2	Supplier	S_suppkeye	Sim	1
4	Orders	O_orderdate	Sim	1
4	Lineitem	L_orderkey	Sim	2
10	Orders	O_orderdate	Sim	2
10	Lineitem	L_orderkey	Sim	2
10	Customer	C_custkey	Sim	1
17	Lineitem	L_partkey	Não	5
17	Part	P_brand, P_container, P_partkey	Sim	1
19	Part	P_partkey	Sim	3

Table 1. Resultados dos Testes Realizados.

intervenção humana. Os testes de desempenho realizados com o *benchmark* TPC-H comprovaram a eficácia do *Dbx*.

Como trabalhos futuros pretendemos avaliar a pré-criação de índices relacionados às chaves primárias e estrangeiras, além de propor alternativas para que este processo de auto-sintonia leve em consideração a generalidade e a relevância dos índices a serem selecionados.

References

Bruno, N. and Chaudhuri, S. (2005). Automatic physical database tuning: a relaxation-based approach. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, pages 227–238, New York, NY, USA. ACM.

Bruno, N. and Chaudhuri, S. (2007). An online approach to physical design tuning. In *Proceedings of the ICDE Conference (ICDE07)*.

Luhring, M., Sattler, K., Schmidt, K., and Schallehn, E. (2007). Autonomous management of soft indexes. In *Proceedings of the 23rd IEEE International Conference on Data Engineering Workshop (ICDE'07)*.

Morelli, E. M. T. (2006). *Recriação Automática de Índices em um SGBD Relacional (in portuguese)*. PhD thesis, Department of Informatics, PUC-Rio.

Salles, M. V. and Lifschitz, S. (2005). Autonomic index management. In *In Proceedings of the International Conference on Autonomic Computing (ICAC)*.

Schnaitter, K., Abiteboul, S., Milo, T., and Polyzotis, N. (2006). Colt: continuous on-line tuning. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD06)*, pages 793–795, New York, NY, USA. ACM.

Weikum, G., Hasse, C. Monkeberg, A., and Zabback, P. (1994). The COMFORT automatic tuning project, invited project review. *Information Systems*, 19(5):381–432.