



Eduardo Maria Terra Morelli

Recriação Automática de Índices em um SGBD Relacional

Dissertação de Mestrado

Dissertação apresentada ao Programa de Pós-graduação em Informática da PUC-Rio como requisito parcial para obtenção do grau de Mestre em Informática.

Orientador: Prof. Sérgio Lifschitz

Rio de Janeiro, setembro de 2006



Eduardo Maria Terra Morelli

Recriação Automática de Índices em um SGBD Relacional

Dissertação apresentada como requisito parcial para obtenção do título de Mestre pelo Programa de Pós-Graduação em Mestrado em Informática do Departamento de Informática do Centro Técnico e Científico da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Sérgio Lifschitz

Orientador
PUC-Rio

Prof. Rubens Nascimento Melo

Departamento de Informática - PUC-Rio

Prof. Renato Fontoura de Gusmão Cerqueira

Departamento de Informática - PUC-Rio

Prof. Geraldo Zimbrão da Silva

Departamento da Ciência da Computação - UFRJ

Prof. Asterio Kiyoshi Tanaka

Departamento de Informática Aplicada - UNIRIO

Prof. José Eugenio Leal

Coordenador(a) Setorial do Centro Técnico Científico - PUC-Rio

Rio de Janeiro, 19 de setembro de 2006

Todos os direitos reservados. É proibida a reprodução total ou parcial do trabalho sem autorização da universidade, do autor e do orientador.

Eduardo Maria Terra Morelli

Formou-se em Tecnólogo em Processamento de Dados pela PUC-Rio, em 1990. Trabalha desde 1997 como administrador de bases de dados corporativas. Autor dos livros Oracle 8: SQL, PL/SQL e Administração (2000), SQL Server 2000 Fundamental (2001) e Oracle 9i Fundamental (2002), todos pela Editora Érica. Desde 1986 está envolvido com tarefas de treinamento, tendo ministrado cerca de 6.400 horas e desenvolvido mais de trinta apostilas. Também contribui para revistas do ramo tal como SQL Magazine.

Ficha Catalográfica

Morelli, Eduardo Maria Terra

Recriação automática de índices em um SGDB relacional / Eduardo Maria Terra Morelli ; orientador: Sérgio Lifschitz. – Rio de Janeiro : PUC, Departamento de Informática, 2006.

120 f. : il. ; 30 cm

Dissertação (mestrado) – Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática.

CDD: 004

À Monick, Leonardo e Thiago,
cúmplices em cada pequena conquista que viabilizou este grande trabalho.

Agradecimentos

Ao longo do período no qual desenrolou-se esta dissertação, aconteceram inúmeros momentos de plena euforia e outros de completo desânimo. Felizmente, contei com precisas orientações do Professor Sergio Lifschitz, responsável direto pela manutenção dos trabalhos no rumo adequado. Os incontáveis encontros proporcionaram debates muito interessantes, fazendo as várias horas parecerem alguns minutos.

Devo agradecer a Marcos Salles pela sua excelente dissertação, que lançou as bases a partir das quais pude edificar este trabalho.

Obrigado a Anolan Milanés pelos inúmeros pedidos de socorro atendidos. Sua dissertação também contribuiu para determinar os rumos a seguir.

À amiga Janaína Oleinik, pelas horas a fio de estudos e pelo apoio constante.

Agradeço aos meus valorosos ajudantes, Felipe Rondon e Daniel Seabra, por valiosas contribuições de resultados experimentais.

Finalmente, não poderia esquecer de José Maria Monteiro, cujas dicas foram inestimáveis em diversos momentos.

Resumo

Morelli, Eduardo Maria Terra; Lifschitz, Sérgio. **Recriação Automática de Índices em um SGBD Relacional**. Rio de Janeiro, 2006. 120p. Dissertação de Mestrado - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Uma dentre as muitas tarefas desempenhadas por DBAs consiste em tentar garantir que os tempos de respostas dos comandos submetidos por usuários a um grande SGBDR não excedam valores previamente acordados. Esta dissertação segue uma linha de estudos denominada auto-sintonia de índices, que preconiza a realização de ajustes automáticos na execução de consultas SQL, visando reduzi-lhes tempos de resposta, a partir de alterações no conjunto de índices: criação, eliminação e recriação. Este trabalho teve como ponto de partida a dissertação de Marcos Salles [32], que seguiu a mesma linha, propondo um mecanismo automático de criação de índices. Esta dissertação estende [32], primeiro submetendo sua implementação a uma carga de trabalho alternativa e depois realizando eliminações e reconstruções de índices automáticas, levando em consideração níveis de preenchimento de páginas alternativos. Também foram realizados testes utilizando ferramentas comerciais, Oracle 10g e SQL Server 2005, para avaliar quão eficaz comportou-se a implementação proposta em [32]. Vale ressaltar que os testes realizados limitaram-se à criação de índices, já que as ferramentas não oferecem facilidades de reconstrução automática. Diferentemente dos trabalhos publicados nessa linha de estudos e das ferramentas comerciais disponíveis, foi criado um protótipo que não se limita a sugerir novos índices; também são eliminados os que deixaram de ser interessantes, porém, antes ocorre uma avaliação para verificar se a reconstrução não seria mais adequada. Criou-se, inclusive, uma heurística rudimentar que avalia um índice a ser destruído e recomenda sua reconstrução, caso atenda a determinados requisitos.

Palavras-chave

Auto-sintonia; SGBD; Índices; PostgreSQL; Fragmentação

Abstract

Morelli, Eduardo Maria Terra; Lifschitz, Sérgio. **Automatic Reindexing in Relational Databases**. Rio de Janeiro, 2006, 120p. MSc. Dissertation – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

One of the most important tasks of Database Administrators certainly is to guarantee optimal response times to statements submitted by users of big RDBMS. Our dissertation deals with Index Self-tuning, which means creating, dropping or recreating indexes automatically, in order to decrease SQL queries durations. We start from Marcos Salles' dissertation [32], which proposed an automatic way of creating indexes. We extend [32] in many ways: first using a different workload, TPC-H like. Second, following created indexes inspecting its usage. Finally, we have gotten to drop and, mostly, recreate indexes using different fillfactor in leaf pages. Also, we have elaborated many tests using commercial tools, Microsoft SQL Server 2005 and Oracle 10g in order to ratify [32] ideas. Unfortunately, we could not test automatic dropping and recreating in these tools, as long as they do not offer this kind of functionalities. Unlike related work and commercial tools, we have created a code prototype that not only suggests new indexes creations, but also drops and recreates indexes using an own heuristics. To validate our ideas we have used a TPC-C like workload, but we had to make some changes to increase updates and force reindexing.

Keywords

Self-tuning; RDBMS; Indices; PostgreSQL; Fragmentation

Conteúdo

1	Introdução	14
1.1.	Contexto	14
1.2.	O Problema	16
1.3	Estrutura da Dissertação	18
2	Avaliação da Eficácia da Heurística de Benefícios com Cargas OLAP	20
2.1	Trabalhos Correlatos	20
2.2	Auto-sintonia Global de SGBDs	22
2.3	Criação Autônoma de Índices em Bancos de Dados	24
	Heurística de Benefícios	24
	Arquitetura da Implementação	26
	Resultados Obtidos	28
2.4	Agente de Benefícios e TPC-H	29
	EVALUATE	30
	Seletividade	32
	Trabalhos Relacionados com Avaliações TPC-H	34
2.5	Resultados Experimentais com PostgreSQL	36
2.6	Resultados Experimentais com SQL Server 2005	38
2.7	Resultados Experimentais com Oracle 10g	40
2.8	Comentários Finais	41
3	Extensões à Heurística de Benefícios	43
3.1	Atribuição Cumulativa de Benefícios a Índices	43
3.2	Acompanhamento de Índices Criados: Bônus	44
	Malus	47
3.3	Eliminação Automática de Índices	47
3.4	Implementações: Bônus e Eliminação Automática de Índices	48
	Camada Crença	48
	Camada Raciocínio	50

Camada Ação	51
3.5 Resultados Experimentais	51
Ambiente	51
Estrutura dos Testes	51
Conclusões	55
3.6 Comentários Finais	55
4 Reconstrução Automática de Índices	57
4.1 Bases da Heurística de Reconstrução Automática de Índices	57
Grau de Fragmentação	59
Tamanho	60
Varreduras	60
4.2 Fator de Preenchimento	61
4.3 GETSIZE	62
4.4 Heurística de Reconstrução Automática de Índices	65
4.5 Estrutura Funcional	67
Camada Sensor	68
Camada Crença	69
Camada Raciocínio	70
Camada Ação	70
Camada Colaboração	71
4.6 Interações entre Agentes	73
4.7 Implementações e Resultados Obtidos	77
Estrutura dos Testes	78
4.8 Comentários Finais	80
5 Conclusões e Trabalhos Futuros	82
Contribuições da Dissertação	83
Trabalhos Futuros	83
Bibliografia	87
Apêndice A Benchmark TPCH	91
A.1 TPC	91

A.2 TPCH	92
A.3 Consultas Escolhidas	95
Consulta 1	96
Consulta 2	97
Consulta 4	98
Consulta 10	99
Consulta 17	100
A.3 Demais Consultas	102
Apêndice B Avaliação de Ferramentas de Apoio ao DBA	103
B.1 - Introdução	103
B.2 - BMC	105
B.3 - Embarcadero	112
B.4 - Idera	112
B.5 - Veritas	116

Lista de figuras

Figura 1.1: consulta realizando cinco varreduras sobre uma tabela	17
Figura 1.2: índice não fragmentado.	17
Figura 1.3: índice fragmentado	17
Figura 2.1: estratégia de Avaliação de consultas.	25
Figura 2.2: estratégia de Avaliação de atualizações	26
Figura 2.3: framework de construção de agentes de [KPP+99].	27
Figura 2.4: exemplo de execução de evaluate , cujo argumento (select) tem sua execução inibida.	31
Figura 2.5: o otimizador proporciona ganhos decrescentes à medida que piora a seletividade	34
Figura 3.1: estrutura da tabela Venda	45
Figura 3.2: comando executado sobre a tabela Venda	45
Figura 3.3: outro comando sobre a tabela Venda	45
Figura 3.4: estratégia de avaliação de consultas	50
Figura 3.5: estratégia de avaliação de atualizações	50
Figura 4.1: nível folha de um índice recém criado.	58
Figura 4.2: nível folha de um índice que sofreu um <i>page-split</i> .	58
Figura 4.3: nível folha de um índice com alto grau de fragmentação.	59
Figura 4.4: fórmula que obtém o grau de fragmentação de um índice	59
Figura 4.5: consulta efetuando várias varreduras sobre a tabela Venda .	60
Figura 4.6: investigação do tamanho de uma tabela em PostgreSQL	62
Figura 4.7: comando fragmentando índice sobre o campo num .	62
Figura 4.8: novos valores obtidos em consulta à metabase, após fragmentar o índice.	63
Figura 4.9: exemplo de utilização da função pgstattable .	63
Figura 4.10: exemplo de utilização da função pgstatindex .	64
Figura 4.11: criação de um índice parcialmente preenchido	64
Figura 4.12: recriação de um índice utilizando fator de preenchimento maior	65
Figura 4.13: fórmula para obtenção do fator de preenchimento	66
Figura 4.14: diagrama de classes envolvidas na configuração das camadas	67

Figura 4.15: diagrama de classes na camada <i>Sensor</i>	68
Figura 4.16: diagrama de classes na camada <i>Crença</i>	69
Figura 4.17: diagrama de classes na camada <i>Raciocínio</i>	70
Figura 4.18: diagrama de classes na camada <i>Ação</i>	71
Figura 4.19: diagrama de classes na camada <i>Colaboração</i>	72
Figura 4.20: interação entre agentes via camada <i>Colaboração</i>	73
Figura 4.21: Interação causada pelo serviço first_usage : primeira parte	75
Figura 4.22: Interação causada pelo serviço first_usage : segunda parte	76
Figura A.1: esquema de Dados base de TPC-H	92
Figura A.2: totais consolidados para transações financeiras ocorridas em um determinado período	96
Figura A.3: seleciona qual fornecedor deve ser escolhido para realizar um pedido de um determinado componente em uma dada região	97
Figura A.4: verifica eficácia do sistema de controle de prioridades de pedidos e checa níveis de satisfação de clientes.	98
Figura A.5: mostra clientes que podem ter tido problemas com entregas.	99
Figura A.6: determina a possível perda decorrente caso alguns pedidos tenham reduzidas pequenas quantidades de componentes. Isto permitiria reduzir o nível de <i>overhead</i> , permitindo maior concentração nas grandes compras.	100
Figura A.7: exibe descontos produzidos segundo diferentes estratégias utilizadas.	101
Figura A.8: nova versão para consulta 19.	102
Figura B.1: ambiente da ferramenta BMC SQL Explorer	106
Figura B.2: consulta típica acessando várias tabelas	107
Figura B.3: resultado de análise de comando SQL	108
Figura B.4: resultado de um Index Advisor.	109
Figura B.5: comandos em busca de índices hipotéticos	110
Figura B.6: comparação entre dois comandos: um com índice hipotético e outro sem.	111
Figura B.7: ambiente da ferramenta Idera SQL Diagnostic Manager	113
Figura B.8: identificando uma consulta agressiva	115
Figura B.9: Agent Manager	117
Figura B.10: ambiente da ferramenta InDepth	118

Lista de tabelas

Tabela 2.1: 6 consultas escolhidas e 8 índices propostos pelo <i>toolkit</i> .	37
Tabela 2.2: 6 consultas escolhidas, 9 índices criados pelo agente e quantas vezes foi necessário executar com evaluate cada consulta para chegar às criações.	37
Tabela 2.3: 6 consultas escolhidas, 11 índices criados pelo SQL Server 2005.	39
Tabela 2.4: 6 consultas escolhidas, 10 índices criados pelo Oracle 10g.	41
Tabela 3.1: sequência de comandos revelando um caso típico de promoção de um índice hipotético em real.	46
Tabela 3.3: transações disparados pelo <i>toolkit</i> e respectivas quantidades de comandos	53
Tabela 3.4: transações alteradas com intuito de forçar a eliminação de índices	54
Tabela 4.1: relação de serviços caracterizando a cooperação entre agentes.	72
Tabela 4.2: valores de parâmetros relevantes de inicialização do PostgreSQL	77
Tabela 4.3: resumo da bateria de testes	80
Tabela A.1: testes constituindo o <i>benchmark</i> TPC-H	94
Tabela B.1: relação de fabricantes visitados.	103
Tabela B.2: relação de módulos da família SmartDBA do fabricante BMC	105
Tabela B.3: elementos da interface do produto BMC SQL Explorer	107
Tabela B.4: resumo de funcionalidades do produto BMC SQL Explorer	111
Tabela B.5: serviços oferecidos pela ferramenta SQL Diagnostic Manager	115
Tabela B.6: resumo de funcionalidades do produto Idera SQL Diagnostic Manager	116
Tabela B.7: principais elementos da ferramenta InDepth	118
Tabela B.8: resumo de funcionalidades do produto Veritas InDepth	119