5
Conclusões e Trabalhos Futuros

Este trabalho de pesquisa abordou auto-sintonia voltada para criação, exclusão e recriação automáticas de índices. O ponto de partida foi a dissertação de Marcos Salles [32], onde apresentou-se uma proposta para criação e destruição automática de índices. Estendeu-se [32] ao propor um procedimento mais adequado para eliminação automática de índices. Ao realizar baterias de testes, observou-se que índices foram criados, destruídos e depois novamente criados. Concluiu-se que tais índices poderiam ser mais úteis, caso não fossem destruídos, mas recriados. Daí concluiu-se a necessidade em propor uma heurística que abordasse a recriação automática de índices.

Inicialmente, houve a preocupação de conferir mais robustez aos resultados obtidos em [32] ao submeter a ferramenta de criação automática de índices a uma carga de trabalho com características OLAP. Partindo de um esquema extraído do *benchmark* TPC-H, registraram-se os índices criados e comparou-se o conjunto com obtenções equivalentes em Microsoft SQL Server 2005 e Oracle 10g. Constatou-se a eficácia da ferramenta, já que o conjunto de índices por ela criados não diferiu muito dos criados em outros SGBDs.

O trabalho de comprovação da eficácia da ferramenta de criação e destruição automática de índices revelou uma deficiência ao acompanhar índices criados. Propõe-se no Capítulo 3 o mecanismo denominado "Bonus", capaz de reduzir os malefícios da deficiência. Testes comprobatórios do sucesso das novas implementações levaram à proposta da Heurística de Recriação Automática de Índices.

Após analisar o problema da fragmentação de índices, chegou-se à conclusão que índices fragmentados prejudicam o desempenho de consultas. A nova heurística analisa casos onde índices, na eminência da eliminação, podem ser refeitos utilizando configurações que retardem os efeitos negativos da fragmentação.

Contribuições da Dissertação

As principais contribuições trazidas por esta dissertação, listadas por ordem de relevância da maior para menor, foram:

1. A implementação de uma arquitetura para auto-sintonia em um SGBD de código fonte aberto envolvendo as operações de criação, eliminação e recriação de índices.
2. Melhoria do mecanismo proposto em [32] para eliminar índices automaticamente ao utilizar um acompanhamento mais realista de índices criados;
3. Comando **evaluate** para o SGBD PostgreSQL, que permite analisar uma consulta sem executá-la e, eventualmente criar índices que melhorem seu desempenho;
4. Cláusula **fillfactor** para o SGBD PostgreSQL, que permite a criação de índices com fator de preenchimento variável de páginas de um índice em seu nível folha;
5. Comando **getsize** para o SGBD PostgreSQL, que permite obter o grau de fragmentação de um índice;
6. Estudo das principais características do *benchmark* TPC-H e como pode-se utilizá-lo para comprovar a eficácia de aplicações em bancos de dados; Aplicações práticas ocorreram em três SGBDs: PostgreSQL, Microsoft SQL Server 2005 e Oracle 10g;
7. Discussão sobre **seletividade** e suas consequências na utilização de índices;
8. Discussão sobre **fragmentação** de índices e como afeta-se negativamente o desempenho de consultas;

Trabalhos Futuros

Uma extensão interessante da bateria de testes realizada no Capítulo 2, seria a verificação do comportamento do Agente de Benefícios enriquecido com o acompanhamento de índices criados, bem como a eliminação dos mesmos. Para isto, as execuções das *Refresh Functions* poderiam acontecer de maneira que muitas tuplas sejam alteradas, ou talvez uma variação do benchmark TPC-H

pudesse ser alterado (TPCH-UPD75). Uma nova leva de testes deveria ser aplicada ao Sistema composto pelos dois agentes, Benefícios e Desfragmentador.

Os testes presentes no Capítulo 2 poderiam ser refeitos, porém levando em considereção os índices previamente criados. Seria analisada a influência de execuções prévias nas seguintes, o que tornaria os testes mais realistas.

Outra melhoria ao Agente de Benefícios poderia ser a possibilidade de pré-criação de índices envolvidos com chaves primárias e estrangeiras. Dado um esquema de dados, o agente já poderia conhecer os campos envolvidos em chaves, sem precisar coletar comandos que acusassem a necessidade natural de índices em tais campos.

O aumento do percentual de captura de comandos por parte do Agente de Benefícios, atualmente em 5%, traria grandes melhorias aos trabalhos realizados pelo Agente Desfragmentador, já que não se perderiam comandos que afetassem o grau de fragmentação dos índices.

Em [7] concentra-se a discussão sobre a atribuição proporcional de benefícios durante a fase de seleção de índices, porém, nada se menciona sobre mecanismos que também avaliem o papel depreciativo de cada índice em comandos de atualização. Indubitavelmente, trata-se de uma ótima futura área de estudos.

A Heurística de Reconstrução Automática de Índices poderia decidir a reconstrução antes da eliminação iminente, isto é, não seria necessário manter um índice fragmentado até o momento em que decida-se eliminá-lo ou recriá-lo. Esta mudança seria viável caso fosse melhorado o registro de incidências de *page splits*, realizando consultas à metabase, ao invés de executar as vareduras onerosas tal como efetuadas pelo comando **getsize**. Uma alternativa interessante, seria criar alertas nos quais o Agente poderia emitir avisos informando que um determinado índice atingiu níveis preocupantes de fragmentação. Estes níveis poderiam ser pré-configurados utilizando *thresholds*.

Várias pequenas alterações no código gerado trariam grandes ganhos à implementação presente:

i. As informações coletadas pelos agentes perdem-se uma vez derrubado o SGBD. Seria de fundamental importância estabelecer algum tipo de persistência, onde seja possível armazenar e recuperar a relação

de índices criados e os hipotéticos acompanhados de seus benefícios acumulados, bem como os conhecimentos sobre graus de fragmentação.

ii. Assim como em [32] implementou-se a heurística proposta em [24], um interessante desdobramento seria implementar as idéias presentes em [7] e constatar se a nova forma de atribuir benefícios a índices candidatos traria ganhos no desempenho.

iii. O Agente de Benefícios poderia capturar ações manuais sobre índices, isto é, criações, exclusões e reconstruções deveriam ser comunicadas ao agente para que este atualize sua metabase. Em princípio, bastaria estender os comandos **create**, **drop** e **reindex index**, além das ações indiretas provocadas por **alter table** ao tratar de restrições (*constraints*).

iv. A forma como foi implementado o comando **getsize** resulta bastante oneroso, principalmente para grandes tabelas [28]. Uma variação, onde não se visitassem todas as linhas, poderia ser implementada. Haveria uma estimativa, mas possuindo margem de erro mínima. Outra alternativa seria, caso fosse possível garantir que as estatísticas da metabase estivessem constantemente atualizadas, ao invés de fazer uma checagem de todas as páginas e tuplas de uma tabela, realizar apenas uma consulta pontual à metabase.

v. Uma nova funcionalidade, CLUSTER, poderia ser aplicada em tabelas nas quais a ordenação física por um determinado campo traria benefícios excepcionais em consultas sobre ela.

vi. Criar alguma parametrização que restringisse a área destinada a novos índices.

vii. Refinar o acompanhamento de índices para que sejam decrementados benefícios daqueles que não são utilizados por longos períodos.

Finalmente, o Sistema de Agentes atual poderia ser estendido para que novas funcionalidades possam acontecer:

i. Existiria um agente encarregado de observar o nível de atividades do SGBD para que as ações de criação, eliminação e reconstrução de índices tivessem um impacto mínimo.

ii. Tratamento de outros tipos de índice, tal qual *bitmap*, por exemplo.

iii. Outras tarefas, normalmente exercidas manualmente por um DBA, poderiam ser automatizadas, tais como gerenciamento de visões materializadas ([2], [44] ou particionamento de tabelas [3].