3
Extensões à Heurística de Benefícios

A ferramenta desenvolvida a partir das idéias propostas no trabalho de pesquisa presente em [32] cria índices automaticamente, segundo preceitos da Heurística de Benefícios. Apesar de discutida em [32], a eliminação automática de índices não consta na ferramenta. Assim, decidiu-se implementá-la e refazer os testes realizados em [32] com intuito de verificar o impacto de tal extensão. Uma vez concluída esta verificação, percebeu-se a necessidade em alterar a Heurística de Benefícios no que tange à escolha dos índices a serem eliminados, já que julgou-se inadequado o critério utilizado nesta decisão.

O presente capítulo divide-se em seis seções: a primeira discute o critério utilizado pela Heurística de Benefícios para eliminação de índices e justifica o argumento de classificá-lo como inadequado. A segunda seção apresenta uma nova proposta para este critério; A terceira discorre sobre a eliminação automática de índices. A 3.4 apresenta detalhes sobre a implementação das idéias presentes nas duas seções anteriores (3.2 e 3.3).

A quinta seção discute resultados obtidos a partir da realização dos testes propostos em [32], porém levando-se em conta a eliminação automática de índices. Constatou-se não apenas a eliminação de índices, como também o fato de ocorrerem novas criações de índices outrora eliminados. Encerra-se o capítulo discutindo questões suscitadas a partir da decisão de recriar ou eliminar índices.

3.1 Atribuição Cumulativa de Benefícios a Índices

Em [32] e [24] propõe-se que todo índice participante de um comando receba o mesmo benefício de todos os índices presentes no comando (Figuras 2.1 e 2.1). Esta abordagem pode criar distorções, já que um índice, que pouco tenha contribuído para baixar o custo de um comando, tenha sido selecionado em um comando com um índice com significativa contribuição, terá um benefício acumulado falacioso.

Por exemplo, imagine que o custo de uma determinada consulta seja 2.000, mas, graças à utilização de dois índices, caia para 500. Suponha também que, dos 1.500 obtidos de benefício, 1.490 sejam devidos apenas ao primeiro índice. Ora, segundo [32] e [24], os dois índices receberiam o benefício de 1.500, o que criaria uma flagrante distorção, já que o índice contribuindo apenas com 10, ganharia um benefício desproporcional à sua participação.

Uma alternativa mais justa consistiria em atribuir a cada índice uma parcela de ganhos proporcional à sua contribuição. Ao invés de cada índice participante de um comando receber um benefício fixo, tal qual o ganho total do comando, como proposto em [32] e [24], seria concedido um valor que dependeria da real contribuição do tal índice ao comando no qual estaria participando.

[7] apresenta um algoritmo que analisa uma carga de trabalho e propõe um conjunto de índices, cujo benefício seja máximo. O algoritmo proposto baseia-se no problema da Mochila e atribui pesos a índices candidatos. Desta forma, associam-se a cada índice ganhos proporcionais a sua contribuição para reduzir o custo global da carga.

A ferramenta de criação automática de índices apresentada a partir do trabalho de pesquisa realizado em [32] implementou apenas a atribuição cumulativa de benefícios em índices hipotéticos. Como o acompanhamento de índices criados não foi implementado, resolveu-se fazê-lo, porém adotando uma estratégia intermediária entre [32] e [7]. Os detalhes aparecem a seguir.

3.2 Acompanhamento de Índices Criados: Bônus

Segundo a Heurística de Benefícios, a contínua verificação da real utilidade de um índice requer um acompanhamento de sua participação em comandos. Desta forma, caso ele participe de forma positiva, deve ganhar benefícios. E, analogamente, quando sua presença acarretar em maior custo para um comando, deve ter diminuída sua carga de benefícios acumulada.

Decidiu-se realizar o acompanhamento de índices criados: durante a fase como hipotético, enquanto o índice vai acumulando benefícios, também registra-se a quantidade de vezes em que ele foi útil. No momento em que deixa de ser hipotético para ser real, o índice ganha um bônus resultante da divisão entre o

benefício acumulado, ou seja, igual ou um pouco superior ao custo de criação, pela quantidade de vezes em que foi utilizado.

Por exemplo, imagine que tenha sido aplicado o comando exibido na Figura 3.2 sobre a tabela **Venda**, descrita na Figura 3.1.

```
Coluna   |     Tipo
---------+--------------
 num     | integer
 prodnum | integer
 data    | date
 qtd     | integer
 valor   | numeric(10,2)
```

Figura 3.1: estrutura da tabela **Venda**

```
select prodnum, data, sum(valor) as total
from venda
where valor > 2000000
and data between '20040101' and '20040102'
group by prodnum, data;
```

Figura 3.2: comando executado sobre a tabela **Venda**

Apurou-se um custo de 60.000, mas a criação de um índice sobre o atributo **data** proporcionou uma economia de 10.112. Após a sexta execução do comando acima, o índice candidato acumulou um benefício de 60.672, portanto, justificando sua criação. E, como foi utilizado seis vezes, ganhou um bônus de 10.112. Durante sua fase como índice criado, toda vez que seja utilizado, seu benefício acumulado receberá um valor idêntico ao bônus. Desta forma, após a terceira utilização, seu benefício já estaria em 91.008 (60.672 + 3 x 10.112). Vale ressaltar que os benefícios apurados para um índice candidato podem variar de um comando a outro. Assim, o comando apresentado na Figura 3.3 poderia conceder uma carga de benefícios distinta:

```
select prodnum, valor
from venda
where valor > 2000000
and data = '20040101';
```

Figura 3.3: outro comando sobre a tabela **Venda**

A Tabela 3.1 apresenta um exemplo onde inicialmente cria-se um índice hipotético; depois ocorre sua promoção para índice real e analisam-se benefícios acumulados com base na ocorrência de comandos onde o índice tenha participação positiva, bem como em outros onde o índice prejudique a execução.

Instante	Fato	Status do Sistema
1	Comando, **C1**, cujo custo foi 1.000 gerou a criação de um índice hipotético, **IH1**, com benefício 500 e custo de criação 2.000.	Armazena-se **IH1** com benefício acumulado 500.
2	Segunda execução de **C1**.	Benefício acumulado de **IH1** atualizado para 1.000.
3	Comando, **C2**, cujo custo foi 800, mas graças a **IH1**, seu custo seria de 200.	Como **IH1** proporcionaria um ganho de 600, atualiza-se o benefício acumulado para 1.600.
4	Segunda execução de **C2**.	Benefício de **IH1** alcança 2.200; justifica-se a "promoção" do índice de hipotético para real (**IR1**). Seu bônus será de 550 (resultado entre a divisão do benefício acumulado e vezes utilizado).
5	Execução de outro comando cuja participação de **IR1** tenha contribuído para diminuir seu custo.	Independente de quanto tenha sido sua contribuição, os benefícios acumulados de **IR1** ganham 550. Novo valor: 2.750.
6	Comando de atualização que afeta índice recém criado.	O ônus da atualização gera uma subtração no benefício acumulado do índice real.

Tabela 3.1: sequência de comandos revelando um caso típico de promoção de um índice hipotético em real.

Na Heurística de Benefícios calcula-se o ônus de uma atualização, levando-se em conta o tamanho da tabela afetada, bem como as tuplas envolvidas na alteração. Precisamente, este valor representa o montante a ser subtraído dos benefícios acumulados de um índice, seja candidato ou não.

Analisando a Tabela 3.1, se depois do instante 6 predominarem os comandos de atualização, nos quais **IR1** não proporcione ganho algum, **IR1** terá

seus benefícios reduzidos até um valor no qual decida-se destruí-lo. Este valor, negativo, eqüivale, em módulo, ao custo de criação.

Malus

Assim como premia-se um índice real com um bônus calculado em sua existência como índice hipotético, também poderia existir um "bônus negativo", ou, aproveitando um vocábulo latim, *malus* [37], que determinasse um valor a ser subtraído dos benefícios acumulados. Esta ação seria mais coerente com as idéias discutidas nesta seção, pois não seriam diminuídos benefícios iguais de todos os índices criados em um comando. Isto aconteceria obedecendo um critério estabelecido a partir do histórico de participações dos índices em comandos anteriores.

Diversos experimentos levaram à conclusão que não seria prático implementar o conceito de *malus*, pois seu cálculo seria muito subjetivo. Não haveria elementos que embasassem o cálculo do montante de benefício negativo acumulado durante a fase de índice hipotético. Concluiu-se que não se pode comparar prejuízos entre índices criados e hipotéticos porque estes não existem enquanto o índice não tiver sido criado fisicamente.

Outra questão levantada a partir da discussão de bônus e *malus*, reside na possibilidade de alterar a Heurística de Benefícios para que mude seu mecanismo de atribuição de benefícios a índices hipotéticos. Novamente, concluiu-se que o estabelecimento de um bônus para índices que ainda não tenham sido criados, representa um trabalho de aproximação que dificilmente proporcionaria resultados melhores do que os proporcionados por [32]. Além disto, deve-se frisar que a quantidade de índices hipotéticos criados é muito maior que a de índices reais, graças à Heurística de Escolha de Índices Candidatos (seção 2.3).

3.3 Eliminação Automática de Índices

Ainda que mencionada em [32], a eliminação automática de índices não foi implementada na ferramenta. Decidiu-se, nesta dissertação, implementar tal funcionalidade, porém realizando uma pequena alteração. Segundo a Heurística de Benefícios, a decisão de eliminar um índice seria tomada quando os benefícios acumulados alcançassem valores negativos que, em módulo, superassem os custos

para criação, mais o ônus da destruição do índice. Este ônus, entretanto, revelou-se insignificante nos testes práticos, já que pôde-se concluir que trata-se, em última instância, da exclusão de uma tupla na metabase. Desta forma, resolveu-se desconsiderar este ônus no cálculo do benefício que indica a necessidade em destruir um índice.

Aproveitando a implementação realizada, decidiu-se acrescentar um detalhe de caráter prático: uma vez excluído um índice real, ele volta a ser hipotético, porém seu benefício acumulado começa com uma carga negativa equivalente ao módulo de seu custo de criação. Esta decisão corrige a distorção de equiparar índices hipotéticos, que nunca causaram malefícios, com outros que já tenham prejudicado comandos durante a fase como índice real. Assim, na Tabela 3.1, quando o benefício acumulado chegar a –2.000, **IR1** será destruído e um novo índice hipotético, **IH1**, será criado. O novo benefício atribuído será negativo (–2.000), inferior ao benefício angariado no comando que teria viabilizado sua criação, como no instante 4.

3.4 Implementações: Bônus e Eliminação Automática de Índices

As implementações do acompanhamento de índices reais com bônus e da eliminação automática de índices seguiu a mesma linha de [Sal01] adotando o mesmo *framework* baseado em agentes proposto em [19] (Figura 2.3).

Mudanças ocorreram nas camadas Crença, Raciocínio e Ação.

Camada Crença

Devido à necessidade de armazenamento de dados históricos, foi necessário estender o conjunto das estruturas de dados utilizadas em [32] composto pelos cinco vetores a seguir:

i. **_indexSet**: membros da *struct* **AgentIndex**, que possui nome do índice candidato, bem como lista de colunas e um ponteiro para a tabela à qual pertence;
ii. **_benefitSet**: elementos representando os benefícios acumulados de cada índice candidato armazenado em **_indexSet**;
iii. **_tableRef**: grupo redundante informando nomes de tabelas contendo os índices candidatos. Como possui a mesma cardinalidade de **_indexSet**,

rapidamente permite-se descobrir a qual tabela pertence um dado índice. Por exemplo, a quinta posição deste vetor revela o nome da tabela à qual pertence o índice ocupando a quinta posição em **_ indexSet**;

iv. **_indexCreationCosts**: elementos representando custos de criação de cada índice candidato armazenado em **_indexSet**. Quando um valor destes for superado pelo benefício acumulado correspondente, acontece a criação do índice e a consequente remoção nestes quatro vetores;

v. **_tables**: membros da *struct* **AgentTable**, que possui nome da tabela, bem como lista de índices hipotéticos;

Além desses vetores, ainda guardam-se atributos numéricos: **_indexCount** e **_tableCount**, (quantidades de índices candidatos e tabelas, respectivamente). Finalmente, o atributo **_lastStatementCost** representa o custo do último comando executado. Este valor será comparado com o custo hipotético do comando acrescido de índices hipotéticos. Caso perceba-se diferença, esta será atribuída em forma de benefícios aos índices candidatos envolvidos no comando. O Apêndice C exemplifica utilizações das estruturas aqui descritas.

Uma nova coluna foi acrescentada ao vetor **_benefitSet** para que sejam registradas quantas vezes utilizou-se um índice. Este dado servirá para cálculo do bônus (razão entre benefícios acumulados e vezes utilizado), quando o índice em questão passe de hipotético para real.

Três novos vetores foram criados: **_createdIndexSet, _benefitCreatedSet** e **_tableCreatedIndexRef**. O primeiro possui membros da *struct* **AgentIndex** para representar índices criados. O atributo **_createdIndexCount** representa a quantidade de índices reais.

O vetor **_ benefitCreatedSet** armazena elementos representando os benefícios acumulados, bônus e custos de eliminação de cada índice criado armazenado em **_createdindexSet**. Finalmente, **_tableCreatedIndexRef** permite rapidamente localizar a tabela na qual foi criado um índice presente em **_ createdIndexSet**.

De acordo com a Heurística de Benefícios, o custo de eliminação obtém-se multiplicando-se por (-1) o custo de criação. Quando os benefícios acumulados forem inferiores ao custo de eliminação, deve-se destruir o índice.

Camada Raciocínio

A Figura 3.4 revela a nova estratégia adotada pela Heurística de Benefícios, para consultas cujo custo tenha diminuído graças à presença de índices. Observar que leva-se em consideração o bônus discutido na seção 3.2.

Avaliação de Consultas

```
1.  Para cada índice hipotético da consulta faça:
2.      Atualize benefícios acumulados
3.      Incremente vezes utilizadas
4.      Caso benefícios acumulados superem custo de criação:
5.          Calcule bônus
6.          Crie índice real
7.          Elimine índice hipotético
8.  Para cada índice real da consulta faça:
9.      Incremente benefícios acumulados com bônus
```

Figura 3.4: estratégia de avaliação de consultas

Observações:

- O cálculo do bônus (linha 5) obtém-se dividindo-se benefícios acumulados por vezes em que o índice foi utilizado;
- Diferente da linha 2, na linha 9 atualizam-se os benefícios do índice criado em questão com base em seu bônus, e não com base na redução do custo da consulta.

A Figura 3.5 revela a estratégia adotada, pela Heurística de Benefícios, para atualizações cujo custo tenha aumentado graças à presença de índices. Observar que não se leva em consideração o bônus discutido na seção 3.2.

Avaliação de Atualizações

```
1.  Para cada índice afetado pela atualização faça:
2.      Decremente benefícios acumulados
3.      Caso índice seja real:
4.          Caso índice deva ser removido:
5.              Remova índice criado
6.              Crie índice hipotético
```

Figura 3.5: estratégia de avaliação de atualizações

Observações:

- A verificação quanto à eliminação de um índice criado (linha 4) testa se os benefícios acumulados são inferiores ao custo de eliminação;
- A remoção do índice real implica na criação de outro hipotético possuindo benefícios acumulados equivalentes ao custo de eliminação.

Camada Ação

Finalmente, houve a necessidade de implementar a requisição para que um índice real fosse destruído (ação para eliminação de índices reais).

Vale ressaltar que decidiu-se não capturar o custo da eliminação de índices reais, já que testes práticos e estudos no código fonte indicaram durações ínfimas, visto que tratava-se, em última instância, de uma simples atualização da metabase.

3.5 Resultados Experimentais

Ambiente

Os testes foram conduzidos utilizando o SGBD PostgreSQL numa estação de trabalho com 512 MB RAM, processador Pentium IV com 3,0 Ghz e disco rígido de 75 GB. O Sistema Operacional utilizado foi Fedora Linux 3, baseado no kernel versão 2.6.12.1. A versão do *toolkit* DBT-2 foi 0.21.1.

Estrutura dos Testes

Como foram realizadas alterações substanciais no protótipo apresentado em [32], julgou-se oportuno repetir os testes realizados em [32] e verificar a criação dos mesmos índices, além de averiguar a ocorrência de eliminação de índices previamente criados.

Seguindo diretrizes traçadas em [32], criou-se uma infra-estrutura com ajuda do *toolkit Database 2 (*DBT-2) oferecido pela ODSL, a mesma organização que disponibiliza o *toolkit* DBT-3, utilizado no capítulo anterior. O *toolkit* DBT-2

apresenta uma arquitetura de três camadas que se comunicam através de rede seguindo o protocolo TCP/IP:

i. Emuladores de terminais;
ii. Concentradores
iii. SGBD

O esquema de dados representa um problema típico de pedidos disparados por clientes e compostos por itens localizados em armazéns que atendem a um certo número de distritos. Cada armazém possui 100.000 itens. Há dez distritos, que por sua vez atendem a 3.000 compradores; Cada armazém cobre dez emuladores de terminal. Assim, com quatro armazéns haveria 40 emuladores a serem distribuídos pelos concentradores e estes conectam-se ao SGBD.

O *toolkit* gera inúmeras transações com base em três variáveis:

i. Quantidade de Armazéns;
ii. Número de concentradores;
iii. Tempo em segundos.

Adquire-se escalabilidade variando-se a quantidade de armazéns. Por exemplo, a mudança de dois para quatro armazéns aumentaria de 20 para 40 os emuladores. Deve-se determinar uma quantidade razoável de concentradores que garantam atendimento a todos os emuladores. Se o número de concentradores for pequeno, haverá contenção no atendimento de requisições dos emuladores. Por outro lado, determinando uma grande quantidade de concentradores acarretará em desperdício de recursos de máquina.

Constataram-se erros irrecuperáveis no *toolkit* quando aumentou-se em demasia a quantidade de armazéns. Após vários experimentos, chegou-se a uma configuração estável composta por 2 armazéns e 5 concentradores em testes com duração de 90 minutos. Isto permitiu-nos observar a criação dos índices relacionados na Tabela 3.2 que, como esperado, são os mesmos índices indicados por [32].

Índice	Tabela	Colunas
1. Ri_customer_0_1_2	Customer	C_id, c_d_id, c_w_id
2. Ri_customer_1_2_5_3	Customer	C_d_id, c_w_id, c_last, c_first
3. Ri_district_1	District	D_w_id
4. Ri_item_0	Item	I_id
5. Ri_new_order_0_1_2	New_Order	No_o_id, no_d_id, no_w_id
6. Ri_new_order_2	New_Order	No_w_id
7. Ri_order_line_0_1_2	Order_Line	Ol_o_id, ol_d_id, ol_w_id
8. Ri_orders_0_1_2	Orders	O_id, o_d_id, o_w_id
9. Ri_orders_1_2_3_0	Orders	O_d_id, o_w_id, o_c_id, o_id
10. Ri_stock_0_1	Stock	S_i_id, s_w_id

Tabela 3.2: 10 índices criados pelo Agente de Benefícios

Os nomes dos índices revelam os campos constituintes e suas posições ocupadas na tabela. (zero equivale ao primeiro campo da tabela).

Ao analisar o registro de operações ocorridas durante os noventa minutos, constatou-se que nenhuma destruição de índices havia ocorrido. A explicação para este fato deve-se à maior incidência de consultas em relação a comandos de atualização, o que pôde ser entendido após examinar a Tabela 3.3. Além disto, todo comando de atualização possui cláusula com filtro, o que provoca a ocorrência de uma consulta previamente à atualização.

Transação	Selects	Inserts	Updates	Deletes	Percentual
Novo Pedido (*New-Order*)	5	2	3	0	45
Pagamento (*Payment)*	4	1	4	0	43
Entrega (*Delivery)*	3	0	3	1	4
Estado de Pedido (*Order-Status)*	4	0	0	0	4
Nível de Estoque *(Stock-Level)*	2	0	0	0	4

Tabela 3.3: transações disparados pelo *toolkit* e respectivas quantidades de comandos

Como não ocorreram destruições de índices, a realização desse teste não pôde avaliar se as alterações efetuadas no Agente de Benefícios foram eficazes. Assim, decidiu-se realizar pequenas alterações na relação de comandos que constituem as transações listadas na Tabela 3.3. A Tabela 3.4 relaciona as alterações efetuadas.

Transação	**Novos comandos**
Entrega	2 *updates* sobre a coluna **i_id** da tabela **Item** 2 *updates* sobre a coluna **o_id** da tabela **Orders**
Estado de Pedido	2 *updates* sobre a coluna **i_id** da tabela **Item** 1 *update* sobre a coluna **o_id** da tabela **Orders**
Nível de Estoque	2 *updates* sobre a coluna **i_id** da tabela **Item** 1 *update* sobre a coluna **o_id** da tabela **Orders**

Tabela 3.4: transações alteradas com intuito de forçar a eliminação de índices

Comparando as informações reveladas pelas Tabelas 3.2 e 3.4, pode-se observar que apenas dois índices foram afetados: **Ri_item_0**, **Ri_orders_0_1_2** e **Ri_orders_1_2_3_0**. Após nova execução, agora acrescida dos comandos listados na Tabela 3.4, observaram-se os fatos a seguir:

- O índice **Ri_orders_1_2_3_0** não chega nem a ser criado, permanecendo como hipotético;
- O índice **Ri_item_0** foi destruído quatro vezes e criado cinco;
- O índice **Ri_orders_0_1_2** foi destruído duas vezes e criado três.

Neste ponto, deve-se destacar uma limitação importante da ferramenta automática de criação (e agora, eliminação) de índices: nem todos os comandos executados pelo SGBD são capturados pelo Agente de Benefícios. Segundo [32], para testes acima de 60 minutos, são capturados apenas 5% dos comandos submetidos ao SGBD. Se por um lado esta limitação não constituiu entrave para criação de índices em [32], o mesmo não se pode afirmar para os testes aqui desenvolvidos. Este fato representa grave problema, já que não há como garantir que determinados comandos sejam obrigatoriamente capturados pelo agente. Conseqüêntemente, foi necessário a realização de vários testes para observar as eliminações e recriações de **Ri_item_0** e **Ri_orders_0_1_2**. Um trabalho futuro

de grande relevância, poderia ser a tentativa de aumento deste percentual de captura de comandos pelo Agente de Benefícios.

Conclusões

Aumentando a duração dos testes levava à maior incidência de eliminações e criações. As numerosas recriações dos índices levantaram dois questionamentos:

i. Já que um índice será recriado em um momento do futuro, terá sido a destruição uma boa escolha?.
ii. É fato notório que os malefícios causados pela fragmentação de um índice [28]. Não seria mais adequado reconstruir um índice, ao invés de removê-lo, evitando períodos como hipotético após a primeira criação?

Certamente a destruição não constitui boa escolha para índices úteis no futuro. Durante o período em que o índice não existe, as consultas que poderiam se beneficiar dele sofrerão com aumentos de custos de execução.

Para responder a segunda questão, deveriam ser verificadas as utilizações dos índices nos comandos das transações listadas na Tabela 3.3. De fato, descobriu-se que **Ri_orders_0_1_2** foi utilizado na terceira consulta da transação "Estado de Pedido", na cláusula de ordenação. Isto significa que, caso o índice esteja fragmentado, o desempenho da consulta será pior. Conclui-se, então, que o índice deveria estar sempre presente e, periodicamente, criar algum mecanismo que elimine os malefícios de sua fragmentação.

A reconstrução periódica de índices utilizados em operações nas quais seja necessário realizar operações de leitura em muitos blocos, impede a degradação do desempenho de consultas utilizando estes índices [28].

3.6 Comentários Finais

Este capítulo discutiu as extensões à heurística de benefícios, proposta em [32], apresentando funcionalidades que permitem acompanhar a utilização de índices criados. Aproveitando idéias de [32], foi realizada a implementação da destruição automática de índices criados, o que motivou a repetição da bateria de testes presente em [32] com objetivo de verificar quantos índices seriam

destruídos. Observou-se, entretanto, que, dada a característica da carga de trabalho utilizada, não ocorreram eliminações. Desta forma, foi necessário ajustar os testes para que fosse possível comprovar a eficácia das novas funcionalidades.

Uma vez alterada a carga de trabalho, perceberam-se várias destruições, seguidas de novas criações. Este fato levantou o questionamento sobre a validade de reconstruir um índice, ao invés de eliminá-lo e, somente após um determinado período, recriá-lo.

As ferramentas de *tuning* automático oferecidas no mercado, limitam-se à sugestão de criação de novos índices [23]. A facilidade de recriação automática constitui um recurso inovador.

No próximo capítulo propõe-se uma heurística que complementa a de Benefícios, avaliando se um determinado índice, em vias de eliminação, não poderia ser recriado. Desta forma, será criada uma ferramenta completa que cria, elimina e reconstrói índices.