FACULDADE DE ADMINISTRAÇÃO E NEGÓCIOS DE SERGIPE – FANESE
MBA EM ADMINISTRAÇÃO DE BANCO DE DADOS

INGRID GRASIELLY NOGUEIRA SANTOS

ESTUDO DE OTIMIZAÇÃO DE BANCO DE DADOS

Aracaju - SE
2016/01

INGRID GRASIELLY NOGUEIRA SANTOS

ESTUDO DE OTIMIZAÇÃO DE BANCO DE DADOS

Trabalho de Conclusão de Curso apresentado ao Núcleo de Pós-Graduação e Extensão – NPGE, da Faculdade de Administração de Negócios de Sergipe – FANESE, como requisito para a obtenção do título de Especialista em Banco de Dados.

_____

Nome completo do Avaliador

_____

Nome completo do Coordenador de Curso

_____

Nome completo do Aluno

Aprovado (a) com média: _____

Aracaju (SE), ____ de _____ de 2016.

ESTUDO DE OTIMIZAÇÃO DE BANCO DE DADOS

Ingrid Grasielly Nogueira Santos*

RESUMO

Com o aumento do uso da tecnologia, as empresas estão aderindo ao uso de sistema de banco de dados. Para obter uma excelente performances da sua aplicação e gerenciar os registros de suas tabelas.

Desta forma se faz necessário uma análise periódica no desempenho da aplicação, para que dai possa ser feito um levantamento de quais regras de otimização serão aplicadas. Este trabalho tem a finalidade de mostrar que podemos melhorar os problemas dentro de um ambiente que trabalha com transações me tempo real.

Palavras-chave: Banco de dados; desempenho; otimização.

---

* Formada em Gestão de Sistema da Informação, pela Faculdade de Administração e Negócios de Sergipe e Analista de Sistemas na Empresa Medsaude Aracaju/SE, grasiellyingrid@gmail.com

1 INTRODUÇÃO

Atualmente com o avanço da tecnologia, existem diversos sistemas utilizando banco de dados, com as suas respectivas configurações para atender a demanda de cada aplicação. Verifiquei na empresa na qual trabalho que os clientes que adquiram ao sistema de banco de dados foi buscando a segurança dos seus dados e um nível de performance elevado do mesmo.

O banco de dados é uma parte fundamental para o desenvolvimento de software, pois ficam armazenados os dados gerados pelos usuários, possibilitando as ações de consulta, alteração, inclusão e exclusão.

Segundo uma das definições de Elmasri e Navathe:

> "Um banco de dados é projetado, construído e povoado por dados, atendendo a uma proposta específica. Possui um grupo de usuários definido e algumas aplicações preconcebidas, de acordo com o interesse desse grupo de usuários." (2005, p.4)

A complexidade e o tamanho de um banco de dados são variáveis. Eles também podem ser criados e mantidos manualmente ou então computadorizados através de um grupo de aplicações como um sistema gerenciador de banco de dados (SGBD).

Os Sistemas Gerenciadores de Banco de Dados (SGBDs) têm como principal função proteger o banco de dados contra ataques maliciosos ou acessos não autorizados e também precisam garantir que não haja mau funcionamento. Ou falhas, sendo estas hardware ou software. Outra função importante é quanto à manutenção dos dados, sendo que estes precisam estar protegidos com relação aos danos causados pelos usuários, quer sejam intencionais ou não.

O desempenho do banco de dados implica diretamente na performance do software, deve ser uma preocupação desde o momento da modelagem, seguindo as regras para otimização do mesmo. Podemos levar em consideração que as configurações da maquina que foi instalado o sistema facilite bastante o desempenho, mas não podemos deixar as aplicações dependentes do hardware. Por isso é necessário fazer analises periódicas

desses dados para saber entender o problema, elaborar o diagnostico e montar planos e técnicas de otimização. Desta forma podemos executar 3 atividades iniciais para o processo de otimização, para que no momento da extração dos dados seja de forma rápida e segura:

- Planejamento de performance: define e configuração do ambiente do banco de dados será instalado, considerando: Hardware, Software, Sistema operacional e Redes.
- Otimização de instância e banco de dados: ajuste de parâmetro e configuração.
- SQL: Otimização de instruções SQL.

Devido a essa grande importância se faz necessário fazer todo um estudo de como será aplicado essa otimização de forma correta e coerente:

- Ler sobre conceitos e técnicas de otimização de banco de dado;
- Discriminar os métodos a serem utilizados na otimização dos dados para que não haja perda de informações;
- Apresentar fluxograma de todo o processo de otimização dos dados;

É evidente que a velocidade de acesso a dados e o tempo de resposta de consultas é um fator decisivo. O usuário atual não está mais disposto a uma longa espera para conseguir as informações que deseja. Mesmo que a potência das máquinas atuais facilitem bastante não se deve deixar aplicações dependentes do hardware em uso, assim se faz necessário a otimização de consulta de forma independente do hardware.

Melhorar consultas nem sempre é uma tarefa fácil, as vezes podendo consumir mais tempo. Nenhum otimizador é perfeito a ponto de substituir completamente um DBA.

Uma aplicação bem otimizada irá continuar a responder bem mesmo com o aumento de usuários ou volume de dados. Uma aplicação mal otimizada irá apresentar degradação de performance, podendo esta degradação ser linear, exponencial ou abrupta.

As falhas de não se otimizar as aplicações no início do desenvolvimento das mesmas normalmente leva um maior esforço para que se obtenha um resultado semelhante quando há otimização é feita nas fases final do desenvolvimento. Quase sempre, sistemas não aperfeiçoados precisam de hardware mais caros para rodar tão bem quanto um sistema lapidado. Desta forma a medida que se avança no ciclo de vida do desenvolvimento de um software, o custo da potencialização aumenta.

A otimização é aplicada a todas as consultas e a qualquer outro comando (por exemplo alteração, inserção, remoção) que contém uma consulta ou uma clausula where.

Diante disso, é de extrema importância realizar o processo de aperfeiçoamento, pois a seleção de uma boa estratégia faz toda diferença e, em termos do tempo de avaliação, a diferença de custo pode chegar a várias ordens de grandeza.

2 OTIMIZANDO BANCO DE DADOS

2.1 Bando de dados

Segundo Silberschatz,

> "Um modelo de dados é uma coleção de ferramentas conceituais para descrever dados, relações de dados, semântica de dados e restrições de consistência." (2006, p.5).

Existem três maneiras para se descrever um projeto de um banco de dados como no nível físico, lógico e de visão, que são divididos em quatro categorias diferentes de classificação de modelos:

- **Modelo relacional:** são modelos baseados em tabelas. São formadas por uma estrutura de registros de tamanhos fixos e vários tipos. Cada tipo de registro define quantidade de números de campos ou atributos.
- **Modelo de entidade/relacionamento:** é uma coleção de objetos de entidades e de relação entre eles.

- **Modelo de dados baseados em objetos:** é uma extensão do modelo de dados relacional e com orientação a objeto.
- **Modelo de dados semi-estruturado:** é um modelo que os itens de dados são individuais do mesmo tipo e podem ter diferentes conjunto de atributos.

Segundo Silberschatz,

> "O principal objetivo de um banco de dados é fornecer um ambiente que seja tanto conveniente quanto eficiente para as pessoas usarem na recuperação e armazenamento das informações, oferecendo aos usuários apenas o resultado final através de uma visão abstrata dos dados e assim, ocultando os detalhes de como os dados são armazenados e mantidos". (2006,p.1).

2.2 Linguagem de consulta – SQL

Em 1986, foi publicado um padrão SQL chamado de SQL-86 pelo American National Standards Institute (ANSI) e a International Organization for Standardization (ISO). Em 1989, o ANSI publicou o SQL-89 e depois surgiram as versões SQL-92 e SQL-1999. Atualmente a versão utilizada é o SQL-2008. A linguagem SQL possui várias partes:

- Linguagem de definição de dados (DDL): é possível definir esquema de relação, excluir relações e modificar esquemas.
- Linguagem de manipulação de dados interativa (DML): linguagem baseada na álgebra relacional e no calculo relacional de tabela.

Devido ao fato de ter se tornado padrão para os bancos de dados relacionais, os usuários não têm tanta preocupação ao migrar suas aplicações para outros tipos de sistemas de banco de dados.

2.3 Otimização

Otimização é o processo de selecionar o plano de avaliação mais eficiente de uma consulta. Otimizar nada mais é do que localizar uma estratégia que se aproxima do ótimo. Na otimização da consulta, vários planos equivalentes por consulta são gerados e a partir daí escolhe-se o menos oneroso dentre eles. Vale a pena para o sistema, gastar um certo tempo na

seleção de uma estratégia boa ao processar determinada consulta, até mesmo se ela for executada uma única vez [Elmasri e Navathe, 2000].

Para que esse processo ocorra se faz necessário antes da implementação do sistema, fazer o levantamento de requisitos, para saber quais informações devem estar disponíveis na aplicação e nas operações que devem ser realizadas sobre o mesmo. O levantamento de requisitos pode ser representado de várias formas, em linguagens textuais ou linguagens gráficas (HEUSER, 2001).

De acordo com as contribuições de Souza (2009) o otimizador tem a finalidade geral de escolher uma estratégia eficaz para avaliar determinada expressão relacional.

Desta forma uma das principais atividades de um DBA é verificar periodicamente o desempenho do sistema. E saber escolher as alternativas que melhorem o desempenho e com baixo custo.

Para que esse processo de otimização seja bem implementado depois do levantamento de requisitos o controle de acesso e armazenamento em um Sistema Gerenciador de Banco de Dados (SGBDs), desempenhe as principais funções:

- Processamento e Otimização de Consultas: deve-se primeiro, ser analisada, validada e otimizada, para que possa ser executada;
- Processamento de Transações: garantir que transações tenha propriedades ACID (Atomicidade, Consistência, Independência e Durabilidade);
- Recuperação de Falhas: manter a consistência e mesmo depois de falhas, a partir de arquivo de log;

Um dos fatores que mais influenciam no desempenho de aplicações de SQL são definidas como chave primária das tabelas. Outro fator importante é a criação de índices.

2.3.1 Índices

Índices são estruturas de banco de dados associados com tabelas e são criados para acessar rapidamente dentro de uma tabela. Índices são transparentes para aplicações e usuários pois com sua presença ou ausência não requer alterações no SQL. Além disso, índices necessitam de espaços adicional de armazenamento, sendo comum este tomar mais espaço que a tabela à qual ele referencia.

Os índices podem ser divido em três tipos básicos:

- Índices de cluster: armazenam os valores de chave de uma tabela cluster.
- Índices de tabela: armazena os valores das linhas de uma tabela com a localização física.
- Índices de bitmap: tipo de índice especial criado para da suporte às consultas.

Quando a tabela não possui índices clusterizados seus dados ficam armazenados numa estrutura denominada *heap*. Denomina-se *heap* porque os dados não têm uma ordem lógica e são gravados nas páginas que tem espaço disponível.

Segundo Pletsch (2005), a escolha correta de quais colunas irão formar o índice é algo fundamental para a sua utilidade. Cabe ao projetista definir se é necessário a criação de novos índices e quais serão os critérios adotados para a sua criação.

Cardinalidade é o número máximo e mínimo de ocorrências de uma entidade que estão associadas às ocorrências de outra entidade que participa do relacionamento. Ou seja, a cardinalidade é importante para ajudar a definir o relacionamento, pois ela define o número de ocorrências em um relacionamento. Podemos dividir em dois tipos:

- Cardinalidade mínima – indica o número mínimo de ocorrências, que pode ser representada por 0 relacionamento opcional e 1 associação obrigatória.

- Cardinalidade máxima – é o numero máximo de ocorrências de uma entidade associadas a outras entidades relacionadas. Podendo ser representado por 1(uma ocorrência) ou n(varias ocorrências).

Os campos para serem indexados a fim de ganhar desempenho são as Chaves Primárias, as Chaves Estrangeiras, as colunas acessadas por ranges (between) e os campos utilizados em group by ou order by. Além do mais, os campos que não devem ser indexados são os campos dos tipos: text, image, decimais, os que são calculados, e com alta cardinalidade.

Ainda segundo Pletsch (2005), é indispensável entender que cada índice representa custo de desempenho na ocasião em que é efetuada uma alteração, e esta operação demanda tempo de processamento. Toda vez que for inserido ou excluído um registro de uma tabela que possui um índice, o mesmo precisará ser atualizado, afim de que continue a realizar sua função de maneira correta. Já na execução de uma atualização, só deverá ser atualizado o índice caso a coluna deste for alterada.

Quando uma coluna indexada é NULL, a linha correspondente não terá entrada no índice. Ou seja, *nulls não são indexados.*

Quando se utiliza o condição IS NULL na clausula Where de uma consulta é necessário considerar o impacto na performance.

Para resolver este problema pode ser resolvido se a coluna indexada for definida como NOT NULL.

Quando o otimizador não encontra um índice disponível, ele realiza a verificação de tabela para que assim possa criar e projetar índices que serão apropriados. Essa é outra importante função do Otimizador de consulta.

Geralmente os índices contêm poucas colunas por linha e ficam classificadas em ordem, tornando assim, bem mais rápida a busca do que se fosse feita na tabela. Quando a pesquisa é feita através do índice, o otimizador busca as colunas de chave e encontra o local pelo qual estão armazenadas as linhas para aquela consulta específica.

Outra técnica que pode ser utilizada é a aplicação de regras baseadas em heurísticas.

2.3.2 Heurística

Regras heurísticas modificam a representação interna da consulta e melhoram o desempenho da mesma. Quando ocorre o uso da heurísticas ocorre o *paser*, que significa interpretar e analisar um comando SQL.

Executar essas operações de seleção o mais cedo possível, para eliminar tuplas que não preenchem a condição de seleção;

- Combinar seleções com um produto cartesiano anterior, de forma a produzir uma junção natural, que é bem mais econômica do que um produto cartesiano;
- Aplicar operações de projeção de forma antecipada, isto é, transferir ou inserir projeções sempre que necessário;
- Procurar expressões comuns em uma árvore, devendo computá-las uma única vez e guardá-las em uma tabela temporária [Elmasri e Navathe, 2000].

Depois que o *parser* gera a representação inicial , e criado uma plano de execução de consulta para executar grupos de operações com caminhos de acessos disponíveis.

A árvore de consulta representa a entrada de relações da consulta através de nós e também representa as operações da álgebra relacionai com nós internos. A execução da árvore de consulta consiste em executar operações de nós internos sempre que seus operandos estão disponíveis e substitui aquele nó interno pela relação resultante da execução da operação. A execução termina quando os nós da raiz são executados e produzem uma relação resultante para a consulta.

2.3.3 Custos

Um aperfeiçoamento de consulta não depende somente das regras de heurísticas para otimizar uma consulta, pois ele pode também calcular e comparar os custos da execução da referida consulta e testar as diferentes estratégias de execução, escolhendo a que possuir menor estimativa de custo. Um otimizador baseado no custo gera uma faixa de planos de avaliação a partir de uma determinada consulta, usando as regras de equivalência, e escolhe aquele de menor custo.

Para otimizar se faz necessário estimar e comparar as diferentes estratégias de execução. Uma otimização mais elaborada é indicada em consultas compiladas, que são consultas na qual a otimização é feita em tempo de compilação, sendo consultas interpretadas indicadas para otimização parcial, que ocorrem em tempo de execução.

Alguns componentes são necessários para o custo da execução de uma consulta, como:

- Custo de acesso ao armazenamento secundário: refere-se ao custo da busca, leitura e escrita de blocos de dados que residem em armazenamento secundário, principalmente em discos. Fatores como blocos de arquivos alocados de maneira adjacente no mesmo cilindro ou disco afetam o custo de acesso.
- Custo de armazenamento: custo de arquivos temporários que são gerados por uma estratégia de execução de consulta.
- Custo de computação: compõe o custo de realização, na memória, referente às operações de buffers de dados durante a execução da consulta.
- Custo do uso de memória: custo referente ao número de buffers de memória que são solicitados durante a execução da consulta.
- Custo de comunicação: custo que se refere ao transporte da consulta e de seus resultados de um site de banco de dados para o site ou terminal onde a consulta originou-se.

Segundo Elmasri e Navathe, “Para desenvolver funções de custo razoavelmente precisas para operações JOIN, é preciso ter uma estimativa do tamanho (número de tabelas) do arquivo que resulta após a operação JOIN”. (2005, p.378). Dá-se o nome de seletividade da junção (js), a proporção entre o

tamanho do arquivo de junção resultante e o tamanho do arquivo do produto cartesiano.

A redução ainda maior do número de planos de execução pode ser conseguida através de uma escolha heurística de um bom plano e estimando seu custo. Essas otimizações podem diminuir significativamente a sobrecarga da otimização de consulta.

2.3.4 Full Table Scan

Pode ser sempre uma opção a depender dos índices criados sobre a tabela. Pois é uma consulta que trabalha com uma leitura sequencial, bloco a bloco por toda tabela especifica. Neste caso são verificados todos os registros com critérios de seleção (cláusula where).

Mesmo depois de definir toda estrutura para um bom funcionamento da consulta. A escrita do comando SQL pode fazer com que o otimizador escolha um caminho errado para executá-la, buscando valores null ou a utilização de funções que desabilitam o índice provocando erro de acesso a tabela.

2.4 Otimização da Aplicação

Pletsch (2005) afirma que a maior parte dos problemas de desempenho em banco de dados relacionais, são causados por instruções SQL mal escritas, pois a forma que o SGBD utilizará para responder uma requisição de um aplicativo será definida através da forma que a instrução SQL, é elaborada. A forma mais simples para otimizar uma instrução é o uso de índices, porém nem sempre eles são o melhor caminho para otimização das consultas.

2.4.1 Uso de filtros

De acordo com Araújo (2007), a menos que seja realmente necessário, é importante evitar informar todos os campos de uma tabela em uma consulta. A consulta utilizando-se de todos os campos da tabela pode ser executada de forma rápida caso a tabela da base de dados contiver poucos registros, mas para poder adquirir uma performance melhor da consulta recomenda-se filtrar apenas os campos que serão utilizados.

2.4.2 Uso de instrução count

Araújo (2007) afirma que não há a necessidade de utilização desta instrução, pois ao executá-la, será contado um registro de cada vez. Para esta operação existem as tabelas '*sysobjects*' e '*sysindexes*', sendo possível com essas tabelas obter muitas informações de todos os objetos existentes na base de dados.

3.4.3 Uso da instrução in

Segundo Araújo (2007), outra instrução que exige um maior processamento é a instrução *IN,* geralmente utilizada quando é preciso fazer o filtro de um vetor de dados. Outra forma de se informar estes registros seria através da utilização da instrução EXISTS.

2.4.4 Usando operações de comparação

Alencar (2007) relata que quando houver necessidade de usar operadores de comparação, deve-se evitar usar a instrução *NOT* em condições de pesquisa, pois pode ocorrer a diminuição da velocidade de recuperação de dados, uma vez que todos os registros em uma tabela serão avaliados. Além do mais é recomendável sempre usar condições de pesquisa positivas ao invés de negativas, como *NOT BETWEEN*, *NOT IN* e *IS NOT NULL*, pois retardam as consultas.

3 PROCEDIMENTOS METODOLÓGICOS

Esta é uma pesquisa qualitativa do tipo bibliográfico, fundamentada em material textual e sites da internet. Foi utilizados livros e um computador com acesso à internet, para realizarmos consultas e buscar as melhores práticas de otimização de Banco de Dados, visando a integridade dos dados, viabilizando a melhor maneira de entendimento e esclarecimento.

4 CONSIDERAÇÕES FINAIS

Este trabalho teve como objetivo mostrar as principais regras e métodos utilizados para obter um bom desempenho do banco de dados.

A otimização de consultas é apenas um dos vários aspectos que devem ser considerados quando deseja-se construir uma aplicação de acesso a um banco de dados bem otimizada. Para se conseguir melhores resultados critérios de desempenho devem ser considerados nas fazes iniciais do desenvolvimento de software. Com isso também evita-se que seja necessário um grande esforço com re-projeto ou re-codificação para se atingir um nível de desempenho satisfatório, e nem sempre consegue-se o mesmo resultado que se teria conseguido se os critérios de performance fossem considerados desde o início do desenvolvimento.

Otimizar um bando de dados como vimos não é uma tarefa fácil, pois consume grande parte do tempo de um DBA. Por isso é necessário que o mesmo tenha um conhecimento avançado da base de dados que esta otimizando.

Quando se otimiza um banco de dados deve-se observar melhores configurações e hardware adequado. Considera-se também, verificar quais consultas estão lentas, porque estão lentas e a melhor forma de otimizá-las. Sendo assim o primeiro passo é o monitoramento para identificar o problema de desempenho.

Enfim, é de extrema importância que os desenvolvedores tenham conhecimento específico na estruturação das consultas, pois é a partir delas que o plano de execução será gerado pelo otimizador de consultas, e, uma consulta mal estruturada gera um plano de execução deficiente, acarretando em baixo desempenho, sobrecarga e nas consultas, causando e sobrecarga de recursos.

ABSTRACT

With an increasing number of companies are adhering to database systems, with the main objective to preserve your data and have a excellent performance of your application. Thus if a periodic analysis on the application performance is required. To apply the appropriate rules database optimization. Such as making changes to the database schema to improve performance and most importantly always preserve the semantics of the data.

Keywords: Database; performance; optimization.

REFERÊNCIAS

SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S.. Sistema de Banco de Dados. 5ª ed. Rio de Janeiro: Campus - Elsevier, 2006.

PLETSCH, Edson Luís. Avaliação das Técnicas de Desempenho em Sistemas Gerenciadores de Banco de Dados Relacionais. Trabalho de Conclusão do Curso – Centro de Universitário Feevale Instituto de Ciências Exatas e Tecnológicas Curso de Ciência da Computação, 2005.

ELMASRI, Ramez; NAVATHE, Shamkant B. Sistemas de Banco de Dados. 4. ed. São Paulo: Pearson, 2005. 724 p.

ARAUJO, Alexandre. Como criar consultas SQL mais rápidas. Disponível em: https://sqlcomoumtodo.wordpress.com/tag/otimizacao/ Acesso em: 04 jun. 2013