UNIVERSIDADE DE BRASÍLIA

Faculdade do Gama

Sistemas de Banco de Dados 2

Trabalho Final (TF)

Tuning em Banco de Dados

Denniel William Roriz Lima - 170161871

Brasília, DF

2023

1. Introdução

"Tuning" é a ação que visa a otimização do desempenho da sua infraestrutura de dados. Ou seja, ele potencializa o trabalho do banco de dados de busca de uma performance máxima. na camada operacional ou no código-fonte (ARAUJO, 2020). O processo de "Tuning" vai variar e depender sempre da estrutura e regras da organização, sendo assim, de forma a se adequar a essa afirmação ele se dá da seguinte forma:

1. Entendimento do problema;
2. Desenvolvimento de um diagnóstico problema;
3. Aplicação das mudanças e técnicas de otimização, conforme indicado no diagnóstico;

"Tuning" de uma base de dados, também, pode ser definido como a ação de fazer uma aplicação de banco de dados rodar de forma mais "rápida". Essa caracterização de rapidez ele explica como um maior "throughput", que consequentemente significa um menor tempo de resposta para algumas aplicações (SHASHA; BONNET, 2003). Essas definições apresentam em como "Tuning" pode variar baseado em contexto, tecnologias e organização em um banco de dados. Araujo (2020), apresenta um conjunto de atividades caraterizadas por ele, como atividades de Tuning em um banco de dados:

1. Planejamento de performance: atividades de definição e configuração do ambiente no qual o banco de dados está instalado, sendo considerado componentes de hardware, software, Sistemas operacionais e toda infraestrutura de rede. Normalmente é relacionado com aplicações que fazem parte do seu banco de dados.
2. Tuning de instância: seriam atividades que dizem respeito ao trabalho de um DBA (Administrador de Banco de Dados), como ajustes de configuração e definição de novos parâmetros. Esta atividade deve ser conduzida por um profissional certificado, pois trata-se de uma ação bastante técnica e capaz de influenciar operações diárias da empresa.
3. SQL Tuning: Que seriam atividades de otimização de instruções SQL. Em que essa atividade deve ser igualmente conduzida por um responsável técnico especializado na linguagem de pesquisa declarativa padrão para banco de dados.

2. Objetivo

Como a própria definição transmite, o objetivo de "*Tuning"* em um banco de dados são as ações que visam a otimização de melhora do banco de dados dentro do seu modelo organizacional e adaptado ao contexto da empresa.

3. Princípios de Tuning

Apesar de Tuning ter de ser flexível a ponto de adaptar ao contexto de cada empresa, ele deve seguir um conjunto de princípios, como levantado por Sasha e Bonnet (2003). Esses princípios são:

- Pensar globalmente; consertar localmente;
- O particionamento quebra gargalos;
- Custos iniciais são altos e custos de operação são baixos;
- Renderize ao servidor o que é devido ao servidor;
- Esteja preparado para trocas.

3.1. Pensar globalmente e consertar localmente

Uma abordagem comum para um ajuste global é olhar primeiro para as estatísticas de hardware para determinar a utilização do processador, atividade de entrada-saída (E/S), paginação e assim por diante (SHASHA; BONNET, 2003). No exemplo utilizado por Shasha e Bonnet (2003). É detectado uma alta atividade de disco em que pode ser comprado hardwares para reduzi-la, porém essa poderia ser uma solução inadequada, pois a alta atividade de disco pode ser de algumas consultas frequentes que verificam a tabela ao invés de utilizar índices ou porque o compartilha uma disco com alguns dados acessados com frequência.

Outro caso apresentado por Shasha e Bonnet, também, é o caso de uma alta atividade de disco porque o administrador do banco de dados falhou ao aumentar o tamanho do buffer do banco de dados, forçando muitos acessos desnecessários ao disco.

Outro ponto interessante a ser levantado é a otimização ser realizada em consultas que ocupam a maior parte do tempo de execução. Acelerar uma consulta que ocupa 1% do o tempo de execução por um fator de dois acelerará o sistema em no máximo 0,5%. Que disse, se a consulta for crítica de alguma forma, ainda pode valer a pena. Por isso, localizar o problema em uma consulta e corrigi-la deve ser a primeira coisa a ser tentar. Mas certifique-se de que é a consulta importante. (SHASHA; BONNET, 2003)

Ao resolver um problema, você também deve pensar globalmente. Os desenvolvedores podem pedir que você leve seus consulta e "encontrar os índices que irão torná-lo mais rápido.". Frequentemente, você se sairá muito melhor se entender os objetivos da aplicação porque isso pode levar a uma solução muito mais simples. (SHASHA; BONNET, 2003).

3.2. O particionamento quebra gargalos

Um sistema lento raramente é lento porque todos os seus componentes estão saturados. Normalmente, uma parte do sistema limita seu desempenho geral. Essa parte é chamada de gargalo. (SHASHA; BONNET, 2003).

Uma analogia utilizada por Shasha e Bonnet para explicar os gargalos e as estratégias para eliminá-los seria de pensar que os gargalos seriam o engarrafamento de uma autoestrada. Há duas estratégias que podem tratar esse problema de engarrafamento:

1. Fazer os motoristas dirijam mais rápido pelo trecho da rodovia que contém menos pistas.
2. Criar mais faixas para reduzir a carga por faixa ou incentivar os motoristas a evitar os horários de pico.

A primeira estratégia corresponde a um ajuste local e deve ser a de primeira decisão. E a segunda corresponde a particionamento.

O particionamento em sistemas de banco de dados é uma técnica para reduzir a carga em um determinado componente do sistema dividindo a carga por mais recursos ou distribuindo a carga ao longo do tempo (SHASHA; BONNET, 2003).

Alguns exemplos de casos de gargalos que podem ser resolvidos via particionamento:

- Um site que possui disponibilidade em todo o território brasileiro como o Gov.br. Se o sistema for centralizado e ficar sobrecarregado uma solução natural seria de colocar os dados dos clientes de um estado X em um subsistema para esse estado X.
- Problemas de contenção de bloqueio geralmente envolvem poucos recursos. Frequentemente, a lista livre (a lista de páginas de buffer de banco de dados não utilizadas) sofre contenção antes dos arquivos de dados. Uma solução é dividir esses recursos em partes para reduzir a contenção concorrente em cada bloqueio. No caso da lista livre, isso significaria criar várias listas gratuitas, cada uma contendo ponteiros para uma parte das páginas gratuitas. Um fio precisando de uma página gratuita bloquearia e acessaria uma lista gratuita aleatoriamente. Este é um formulário de particionamento lógico (de recursos bloqueáveis). (SHASHA; BONNET, 2003).
- Um sistema com poucas transações longas que acessam os mesmos dados que muitas transações curtas. As transações ("online") terão um desempenho ruim devido à contenção de bloqueio e recurso contenção. O impasse (*deadlock*) pode forçar o cancelamento das transações longas, e as transações longas podem bloquear as mais curtas. Além disso, as transações longas podem usar grandes porções da *pool* de buffers, diminuindo assim as transações curtas, mesmo na ausência de contenção de bloqueio. Uma solução possível é realizar transações longas quando há pouca atividade de transação online e serializar essas transações longas (se forem cargas) para que não interfiram umas nas outras (partição no tempo). Uma segunda é permitir que as transações longas (se forem lidas apenas) para aplicar a dados desatualizados (particionamento no espaço) em hardware separado. (SHASHA; BONNET, 2003).

O particionamento nem sempre vai melhorar o desempenho do sistema. Por exemplo, particionar os dados por filiais pode acarretar despesas adicionais de comunicação para transações que cruzam filiais. (SHASHA; BONNET, 2003).

3.3. Custos iniciais são altos e custos de operação são baixos

Shasha e Bonnet apresentam 4 casos para explicar esse princípio:

- Iniciar uma operação de leitura em um disco é caro, mas uma vez iniciada a operação o disco pode fornecer os dados em alta velocidade. Sendo assim, ler um segmento de 64 kilobytes de um segmento de disco será duas vezes mais caro que ler 512 bytes desse mesmo segmento. Isso sugere que as tabelas frequentemente acessadas devem ser dispostas consecutivamente no disco. Isso também sugere que o particionamento vertical pode ser uma boa estratégia quando consultas importantes selecionam poucas colunas de tabelas contendo centenas de colunas.
- Em um sistema distribuído, a latência de envio de uma mensagem através de uma rede é muito alta em comparação com o custo incremental de enviar mais bytes em uma única mensagem. O resultado líquido é que enviar um pacote de 1 kilobyte será um pouco mais caro do que enviar um pacote de 1 byte. Isso implica que é bom enviar grandes blocos de dados em vez de pequenos.
- O custo de análise, execução de análise semântica e seleção de caminhos de acesso para consultas simples é significativo (mais de 10.000 instruções na maioria dos sistemas). Isso sugere que as consultas frequentemente executadas devem ser compiladas
- Suponha que um programa em uma linguagem de programação padrão como C++, Java, Perl, COBOL ou PL/1 faça chamadas para um sistema de banco de dados. Em alguns sistemas (por exemplo, a maioria dos relacionais), abrir uma conexão e fazer uma chamada incorre em uma despesa significativa. Portanto, é muito melhor para o programa executar uma única chamada SELECT e, em seguida, fazer um loop no resultado do que fazer muitas chamadas ao banco de dados (cada uma com seu próprio SELECT) dentro de um loop da linguagem de programação padrão. Como alternativa, é útil armazenar as conexões em cache.

A lição desse princípio é: obtenha o efeito desejado com o menor número possível de inicializações.

3.4. Renderize ao servidor o que é devido ao servidor

Saber distribuir as tarefas de forma essencial no servidor e aplicação é essencial. As tarefas devem ser alocadas e distribuídas baseada na principal função de cada um. Shasha e Bonnet mostram três fatores que auxiliam ao decidir em qual deve ser alocado a tarefa:

1. Os recursos de computação relativos do cliente e do servidor: Se o servidor estiver sobrecarregado, tudo o mais sendo igual, as tarefas devem ser descarregadas para os clientes.
2. Onde a informação relevante está localizada: Suponha que alguma resposta deva ocorrer (por exemplo, escrever em uma tela) quando ocorrer alguma alteração no banco de dados (por exemplo, inserções em alguma tabela do banco de dados). Em

seguida, um sistema bem projetado deve usar um recurso de gatilho dentro do sistema de banco de dados, em vez de pesquisar no aplicativo. Uma possível solução seria uma consulta periódica na tabela para ver se ela foi alterada. O gatilho, então, é acionado apenas quando a alteração realmente ocorre, acarretando muito menos sobrecarga.

3. Se a tarefa do banco de dados interage com a tela: Se interagir, então a parte que acessa a tela deve ser feita fora de uma transação. O motivo é que a interação da tela pode demorar muito (alguns segundos, pelo menos). Se uma transação T incluir o intervalo, então T impediria que outras transações acessassem os dados que T contém. Assim, a transação deve ser dividida em três etapas:

    a. Uma transação curta recupera os dados.
    b. Uma sessão interativa ocorre no lado do cliente fora de um contexto transacional (sem bloqueios mantidos).
    c. Uma segunda transação curta instala as alterações obtidas durante a sessão interativa.

3.5. Esteja preparado para trocas

Adicionar memória de acesso aleatório permite que um sistema aumente seu tamanho de buffer. Isso reduz o número de acessos ao disco e, portanto, aumenta a velocidade do sistema. Obviamente, a memória de acesso aleatório não é (ainda) gratuita. (Também não é aleatório: acessar memória sequencialmente é mais rápido do que acessar blocos amplamente dispersos.) (SHASHA; BONNET, 2003).

Adicionar um índice geralmente faz com que uma consulta crítica seja executada mais rapidamente, mas envolve mais armazenamento em disco e mais espaço na memória de acesso aleatório. Também requer mais tempo do processador e mais acessos ao disco para inserções e atualizações que não usam o índice. (SHASHA; BONNET, 2003).

Ao usar o particionamento temporal para separar consultas longas de atualizações online, você pode descobrir que muito pouco tempo é alocado para essas consultas longas. Nesse caso, você pode decidir construir um banco de dados de arquivamento separado para o qual você emite apenas consultas longas. Esta é uma solução promissora do ponto de vista de desempenho, mas pode implicar na compra e manutenção de um novo sistema de computador. (SHASHA; BONNET, 2003).

4. Vantagens e Desvantagens

Tuning em um banco de dados envolve diversas técnicas com diversas vantagens e desvantagens sendo levantados para cada caso. Para as técnicas e casos levantados previamente, durante sua explicação, foi explicado suas vantagens e desvantagens. Contudo em um aspecto geral sobre o processo de Tuning de um banco de dados, uma grande vantagem é o aumento do desempenho do banco de dados no geral, como em leituras e

consultas que podem ganhar tempo para o usuário gerando maior satisfação, uma maior escalabilidade para os desenvolvedores, melhor distribuição dos recursos, e etc.

A junção de todos os casos e técnicas explicados apresentam as vantagens de desvantagens do Tuning, assim como os exemplos de aplicação. Uma desvantagem notável que impacta e também foi levantado no primeiro tópico deste documento é a necessidade de profissionais especializados e experientes para aplicar cada uma dessas técnicas, como assim levantado em algumas delas, com a utilização delas em cenários ou em contextos errados isso pode acarretar uma piora de desempenho e causar um efeito contrário.

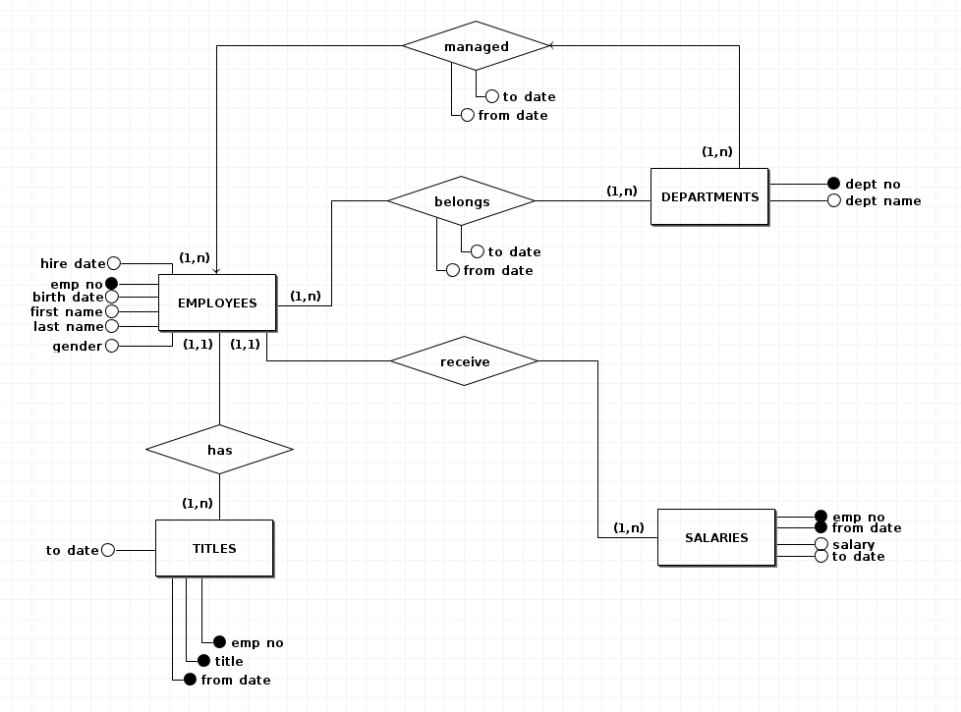
5. Cases de "Tuning"

O primeiro case, que é um case de sucesso é o da empresa Hoya que passou para a migração de seu banco de dados de servidores físicos próprios (que também estavam com seus hardwares antigos e defasados) para um sistema Cloud. Eles não só conseguiram fazer a migração com sucesso como hospedaram em servidores 4 vezes maior do que o que utilizavam. O que no caso entraria como Tuning, mas com relação a infraestrutura também.
O segundo case, sendo um case de insucesso seria o do Sistema de Seleção Unificada (SISU) do Ministério da Educação (MEC). Em 2019, os picos de usuários simultaneos chegou a atingir até 350 mil usuários o que causou uma sobrecarga e dificultou o login e acessos pelos usuários. Uma forma de resolução desse problema poderia ser de particionamento em que distribuiria seus dados em particionamento baseados nos estados de cada candidato, sendo assim, cada estado teria seu subsistema e dessobrecarregaria o servidor central do sistema.

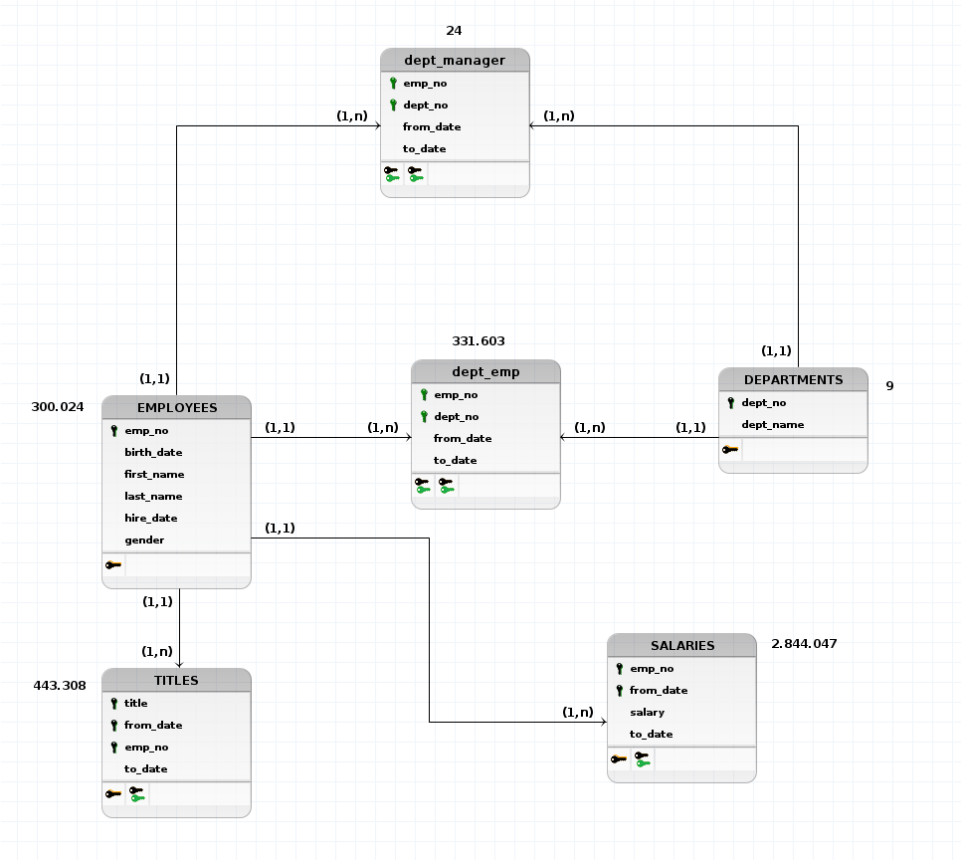
6. Base de Dados (documentação)

O sistema que será utilizado durante o trabalho final será um sistema de funcionários em uma empresa em que possui relação com departamentos e possui controle dos títulos, departamentos no qual pertencem ou gerenciam, assim como seus salários.

6.1. Diagrama Entidade-Relacionamento (D-ER)



6.2. Diagrama Lógico de Dados



6.3. Disponibilidade da Base

A base encontra-se disponível com as orientações de instalação e configuração no endereço: https://dev.mysql.com/doc/employee/en/employees-installation.html

Referência bibliográfica

- ARAUJO, J. Tuning em Banco de Dados: conheça tudo sobre o assunto. 2020. Disponível em: <https://blog.dbacorp.com.br/2020/07/30/tuning-em-banco-de-dados/>.

- RIGO, V. F. ESTUDO DA METODOLOGIA DE TUNING EM BANCO DE DADOS ORACLE. 2012.

- SHASHA, D. E.; BONNET, P. Database tuning: principles, experiments, and troubleshooting techniques. Rev. ed. San Francisco, CA: Morgan Kaufmann Publishers, 2003. (Morgan Kaufmann series in data management systems). ISBN 978-1-55860-753-8.

- SISU 2019 recebe 10 vezes mais acessos do que o esperado e site fica instável | Guia de carreiras | G1. Disponível em: <https://g1.globo.com/educacao/guia-de-carreiras/noticia/2019/01/23/sisu-2019-recebe-10-vezes-mais-acessos-do-que-o-esperado-e-site-fica-instavel.ghtml>.

- VWWEBMASTER. Case de sucesso | Hoya conquista mais estabilidade e performance com banco de dados na nuvem. 2021. Disponível em: <https://www.microserviceit.com.br/en/case-de-sucesso-hoya-conquista-mais-estabilidade-e-performance-com-banco-de-dados-na-nuvem/