Boas Práticas com Bancos de Dados

Boas práticas em SQL para desenvolvedores

Escrito por julianommartins em dezembro 19, 2007

Boas práticas em SQL para desenvolvedoresQuem nunca ouviu alguem reclamar: "o sistema está lento hoje!!!"? Nestes relatos de degradação de desempenho, frequentemente levantamos que esta degradação é decorrente de instruções SQL mal estruturadas ou ainda banco de dados mal planejado, o que, num efeito cascata, só é sentido conforme o sistema vai sendo utilizado, as tabelas sendo povoadas, etc. O SGDB começa a exigir muito processamento, memória e a gerar gargalos na rede, causando assim, efeitos no desempenho da aplicação e na rede!

Outro fato é que atualmente, boa parte dos desenvolvedores desenvolvem código SQL sem ter muito conhecimento sobre fundamentos de banco de dados. Tal falta de conhecimento gera a produção de código ineficiente e com baixa performance. É comum ver equipes de desenvolvimento que não tem um DBA, ficando assim, o desenvolvedor com a tarefa de criar um banco de dados.

Com estes problemas em mente, resolvi criar este post com algumas dicas para que os desenvolvedores tenham algum conteúdo básico e rápido e melhorem suas instruções SQL e a criação de banco de dados.

Tentarei ser o mais genérico possível para conseguir cobrir os bancos de dados mais utilizados atualmente (DB2, Oracle, MySql, Postgres, etc), porém, algumas dicas podem não ser aplicáveis a todos os bancos.

É importante lembrar que praticamente todas as dicas são "debatíveis" em diferentes cenários, portanto, fiquem a vontade para comentar.

Vamos lá:

1- Normalize seu banco de dados. Isso quer dizer básicamente, divida tabelas grandes em tabelas menores e remova redundancia, ou seja, que dados estejam duplicados sem real necessidade.

2. Em instruções select, evite usar "*". Seja restritivo, traga somente os campos realmente necessários, isso alivia a memória do servidor, diminue tráfego na rede, etc. Algumas pessoas defendem que tambem não devem ser criados determinados campos, por exemplo, você tem A + B e pretende guardar C onde C = A + B. Ao invéz de criar uma coluna para armazenar C, passe a utilizar "select (A+B) AS C from tabela". Esse pensamento pode não ser necessariamente válido para Dws. Vamos supor que você tem uma tabela enorme com dados sobre salário por ano. Você pode armazenar o percentual de ajuste e o valor ajustado, fazendo ai uma "desnormalização" para que o comando que vai recuperar os valores do salário não "frite" a CPU forçando-a a fazer muitas contas e perdendo muito desempenho.

3. Existe muito debate sobre essa: Não utilize seu banco de dados para armazenar imagens, ao invéz disso, armazene a URL. Vale lembrar que os bancos de dados atuais estão cada vez mais aprimorados na manipulação de imagens, portanto, aqui abre-se espaço para uma enorme discussão, benchmarck, etc.

4. Para obter maior performance, utilize chaves primárias numéricas ou ainda campos pequenos nas chaves.

5. Utilizando-se stored procedures e functions ao invéz de escrever código no seu programa, vai garantir maior desempenho e segurança para seu sistema como um todo.

6. Utilize o conceito de transações. Vários problemas podem ocorer, por exemplo, a rede cair. Aprenda sobre commit e rollback.

7. Use sempre o tipo de dados correto para armazenar os dados. Por exemplo, não armazene sexo, que vai ser M ou F em um campo Varchar, use apenas 1 caractere: CHAR(1).

8. Evite o uso de cursores, eles consomem muito tempo já que "navegam" registro por registro.

9. Otimize a clausula WHERE: Simples exemplos são o uso de ">" e ">=". Se você quer retornar todas as pessoas de uma tabela que tem idade "> 3", use no where ">=4", dessa forma o banco não fará o scan das páginas até encontrar o 3. Esse princípio é válido desde que você tenha um índice na idade.

10. Quando possivel, crie instruções SQL idênticas, pois no momento da execução de uma instrução, o banco compila a mesma e a preserva em memória, na próxima execução, não vai precisar compilar novamente. Uma ótima técnica para fazer isso é utilizar variáveis nas suas instruções ao invés de passar parametros para o banco.

11. Utilize os mecanismos do banco de dados para persistência: Primary Key, Foreign Key, etc são feitos e otimizados para isso.

12. Quando possivel, trave (lock) uma tabela para executar alguma operação que vai demandar muito acesso a esta tabela, por exemplo, se você vai alterar a estrutura de uma tabela grande ou importar dados neste tabela (falando-se de tabelas realmente grandes).

13. Sempre utilize o nome das colunas em instruções SELECT, INSERT, UPDATE evitando utilizar "*".

14. Evite utilizar o operador "LIKE", ele pode facilmente fazer o desempenho de um banco de dados ruir!

15- Utilize EXISTS ao invéz de COUNT para verificar se existe um determinado registro em uma tabela. É comum ver desenvolvedores fazendo um "select count(X) from Y" para verificar se o COUNT é maior que 0. Utilizando-se EXISTS, o sgbd vai parar no primeiro registro encontrado, se utilizar count, o banco vai varrer toda a tabela.

16- Em joins de tipos de dados diferentes, o SGBD vai ter que converter o tipo hierarquicamente inferior para o outro tipo a fim de efetuar a comparação, e assim, não vai utilizar um índice caso exista.

17. Sobre índices:

* Não crie indices em campos que são alterados constantemente, pois o banco vai ter que atualizar toda sua estrutura de índices em qualquer update feito no campo.
* Prefira criar os indices em chaves primárias e estrangeiras, e em suas queries, utilize estes índices.
* Não tenha muitos índices em seu banco, só o necessário: uma breve explicação sobre o motivo disso, é que o banco de dados mantem toda uma estrutura para gerenciar os índices, então, quanto

mais índices, mais tempo/processamento o SGBD vai utilizar para a manutenção dos mesmos.
* No momento de importação/importação de uma base de dados, não exporte/importe índices. Isso vai consumir mais tempo/processamento. Tambem pode-se não fazer backup de índices.
* Não crie índices em colunas que possuem pouca variação de valores.

18. Entenda os fundamentos de banco de dados. Uma ótima leitura é o livro "Sistema de Banco de Dados", de Abraham Silberschatz, Henry F. Korth e S. Sudarshan lançado no Brasil pela Editora Elsevier.

Práticas Recomendadas - Mauro Pichiliani

Recentemente eu estava revisando o conteúdo do M.O.C. do curso 2073A (Programming a Microsoft SQL Server 2000 Database) e notei que a cada final de capítulo (chamado de módulos) são aprensentadas várias práticas recomendadas pela Microsoft para quem utiliza o SQL Server. Como estas práticas são importantes e recomendadas pela prórpia Microsoft, esta semana resolvi reproduzir algumas destas idéias, de acordo com cada módulo.

Módulo 2: Overview Of Programming SQL Server

* Mantenha a lógica de negócio no Servidor , como Stored Procedures
* Use sintaxe SQL ANSI
* Escolha uma convenção de nomes apropriada
* Salve instruções como Scripts and comente-os bem
* Formate sua instruções Transact-SQL para ser legível a outros

Módulo 3: Creating and Managing Databases

* Faça um Backup do banco de dados Master
* Especifique um tamanho máximo do arquivo
* Especifique um incremento de autocrescimento grande
* Mude o grupo de arquivos (filegroup) padrão

Módulo 4: Creating Data Types and Tables

* Especifique tipos de dados corretos e o de tamanhos apropriados
* Sempre especifique características de coluna no CREATE TABLE
* Gere Scripts para recriar Bancos de Dados e Objetos de Bancos de Dados

Módulo 5: Implementing Data Integrity

* Use constraints por quê elas são compatível com o padrão ANSI
* Use integridade referencial em cascata ao invés de Triggers

Módulo 6: Planning Indexes

* Crie índices em colunas que fazem join em tabelas
* Use índices para reforçar unicidade
* Evite chaves cluster grandes
* Considere a utilização de índices clustered para suportar ordenação e pesquisas por intervalo
* Crie índices que suportam argumentos de pesquisa

Módulo 7: Creating and Maintaining Indexes

* Use a opção FILLFACTOR para otimizar o desempenho
* Use a opção DRP_EXISTING para a manutenção dos índices
* Execute DBCC SHOWCONTIG para medir a fragmentação
* Permita ao SQL Server criar e atualizar estatísticas automaticamente
* Considere a criação de estatísticas em colunas não indexadas para permitir planos de execução mais eficientes

Módulo 8: Implementing Views

* Utiliza uma convenção de nomes padrão
* dbo dever ser o dono de todas as views
* Verifique a dependência de objetos antes de se apagar um objeto
* Nunca apague as linhas na tabela syscomments

* Cuidadosamente avalie a criação de views baseada em views

Módulo 9: Implementing Stored Procedures

* Verifique os parâmetros de entrada (imput)
* Faça cada Stored Procedure efetuar uma única tarefa
* Valide dados antes de iniciar uma transação
* Use as mesmas configurações de conexão para todas as Stored Procedures
* Use WITH ENCRYPTION para esconder o texto de Stored Procedures

Módulo 10: Implementing User-defined Functions

* Utilize funções escalares complexas em pequenos conjuntos de resultados
* Utilize multi-statement functions ao invés de Stored Procedures que retornam tabelas
* Use in-line functions para criar views parametrizadas
* Use in-line functions para filtrar views indexadas

Módulo 11: Implementing Triggers

* Use Triggers somente quando necessário
* Mantenha as instruções de definição de triggers o mais simples possível
* Inclua instruções de checagem de término na definição de triggers recursivos
* Procure minimizar o uso da instrução ROLLBACK em triggers

Módulo 12: Programming Across Multiple Servers

* Use servidores linkados para acessos de dados remotos freqüentes
* Use consultar Ad Hoc para acessos de dados remotos não tão freqüentes
* Configure servidores linkados para executar Stored Procedures remotamente ou para executar consultas distribuídas
* Restrinja acesso a servidores linkados
* Evite duplicar contas de login em servidores diferentes
* Seleciona a coluna apropriada para definir a partição

Módulo 13: Otimizing Query Performance

* Utilize o Query Governor para prevenir consultas de longa duração de consumir recursos do sistema
* Possua um bom conhecimento dos dados e como as consultas acessam os dados
* Crie índices quem cobrem as consultas usadas mais freqüentemente
* Estabeleça estratégias de indexação para uma consulta ou múltiplas consultas
* Evite sobreescrever o otimizador de consultas

Módulo 14: Analyzing Queries

* Defina um índices em uma coluna de alta seletividade
* Garanta que índices úteis existem para todo as as colunas referenciadas no operador OR
* Minimize o uso de joins HASH

Módulo 15: Managing Transaction and Locks

* Mantenha transações curtas
* Faça transações de modo a evitar deadlocks
* Use os padrões do SQL Server para lock
* Seja cuidadoso ao utilizar opções de lock

Documentação do DBA

Olá pessoal. Na coluna desta semana falarei um pouco sobre os principais documentos com os quais um DBA deve lidar. Estes documentos são úteis não apenas para ajudar a documentar aspectos técnicos, mas também para organizar e gerenciar o trabalho do DBA (*Database Administrator*).

Ao contrário do que alguns podem pensar, a documentação não é apenas um trabalho desnecessário que toma tempo e torna tudo mais lento. Ela é um item necessário e obrigatório em qualquer projeto, servindo a vários propósitos.

Em geral, é difícil encontrar profissionais, em particular DBAs, que trabalhem com muita documentação. Isso se deve tanto à questão cultural como a questão que envolve a disponibilidade de tempo para criar e manter a documentação atualizada. De qualquer maneira, neste artigo apresento os 10 principais documentos utilizados por quem trabalha tanto como DBA como desenvolvedor.

1) MER (Modelo Entidade Relacionamento)

O MER (Modelo Entidade Relacionamento), também conhecido apenas como modelo de dados ou diagrama de entidade-relacionamento, é a principal documentação de um banco de dados. Neste diagrama são relacionadas as principais entidades (tabelas) e seus relacionamentos, além de alguns detalhes a respeito das entidades, como nome das colunas nas tabelas, tipos de dados e *constraints*. Geralmente utiliza-se uma ferramenta CASE para modelagem deste diagrama, sendo o ER-Win um dos softwares mais utilizados para elaborar este diagrama. Independente do software utilizado é imprescindível que o DBA conte com ao menos um MER para cada uma das bases que ele administra.

2) Padrões de Variáveis (Tabelas, colunas etc) e Documentação

Uma das boas práticas de desenvolvimento é contar com padrões. Saber colocar nomes significativos e explicativos em funções, variáveis, classes etc está se tornando cada vez mais um pré-requisito para quem trabalha com desenvolvimento. No que tange à banco de dados, a idéia não é diferente: possuir um padrão de nomes para tabelas, colunas, *constraints* e objetos de bancos de dados é muito importante. Aqui a idéia não é apenas possuir um padrão e utilizá-lo largamente, mas possuir um padrão eficaz que seja fácil de ser utilizado, além de fazer sentido no contexto de banco de dados. Para formalizar a adoção de um padrão, recomenda-se montar um documento explicando todos os detalhes deste padrão como, por exemplo, se o padrão é baseado na notação húngara, se os nomes devem ser em português ou se há um limite no tamanho da coluna.

3) Capacity Plan

A necessidade de prever recursos de hardware é uma das grandes responsabilidades de um DBA. Além de economizar recursos, a previsão mostra que há um controle não apenas para os recursos que estão sendo utilizados, mas também para os recursos que podem ser necessários no futuro. Para ajudar nesta previsão, o DBA deve ser responsável pela elaboração de um documento chamado *Capacity Plan*, ou Plano de Capacidade. Este documento deve listar as necessidades de espaço de armazenamento, utilização de CPU, largura de banda e outros requisitos técnicos que possam impactar o banco de dados. Com certeza, este é um documento que envolve muitos aspectos e deve ser elaborado com cuidado. Por questões práticas, muitas vezes é necessário fazer uso de estimativas e tendências ao invés de contar com informações precisas. Deste modo, o documento não precisa estar 100% correto, mas deve conter uma boa base e previsão dos principais recursos computacionais relacionados ao banco de dados.

4) Dicionário de dados

O Dicionário de dados é um documento que complementa o MER. Este documento deve conter mais detalhes a respeito das tabelas e seus relacionamentos. Por exemplo, além de listar todas as colunas de uma tabela, o documento deve fornecer também uma pequena descrição do significado desta coluna, quais são os valores possíveis, a quantidade típica de valores armazenados e quais constraints agem sobre esta coluna. Além das informações sobre colunas, este documento apresenta o nome dos objetos que dependem da tabela, como *stored procedure, triggers, views,* funções etc, e suas respectivas funções, além dos parâmetros necessários e o que é retornado. É importante notar que este documento deve sempre estar alinhado e atualizado com a base de dados atual, para evitar desencontros e desentendimentos.

5) Política de segurança

O documento contento a política de segurança é um documento não-técnico que envolve os

procedimentos, responsabilidades e atribuições relacionadas tanto à segurança das informações como do acesso à elas. Geralmente este documento contém uma política de usuários e senhas, que especificam várias regras, como as definidas abaixo:

- Troca de senha a cada três meses;
- Desabilitar as contas padrão;
- Forçar senhas com letras, números e caracteres especiais que tenham um tamanho mínimo de 10 posições;

Outras políticas gerais de senha, como o cancelamento após um algumas tentativas e horários definidos para certos usuários, também deve constar neste documento, sempre tendo em mente a utilização de sistemas e bancos de dados.

6) N.D.A (*non-disclosure agreement*) - Compromisso de sigilo

Imaginem a seguinte situação:

Somos responsáveis por uma base de dados que deve ser integrada com um sistema externo à empresa. Para discutir os detalhes desta integração, uma reunião é marcada com a equipe externa à empresa que desenvolve o sistema. Durante esta reunião são apresentadas informações sigilosas da empresa que trabalhamos, com o objetivo de discutir os aspectos da integração.

Vamos supor que na situação apresentada acima os profissionais da equipe externa hajam da má fé e utilizem as informações fornecidas para seu próprio benefício, seja comercialmente ou não. Este tipo de situação pode gerar diversos problemas, podendo chegar ao ponto onde a equipe que agiu de má fé ser acusada de roubo.

Para e proteger de situações como estas, é comum fazer uso de um documento chamado NDA (*non-disclousure agreement*), também conhecido como compromisso de sigilo. Este é o tipo de documento que protege todo mundo: tanto quem assina como quem solicita a assinatura. Em termos práticos, que assina compromete-se a não revelar nenhum detalhe da informação que lhe vai ser comunicada sob pena de ser alvo de procedimento legal.

7) S.L.A. (*Service Level Agreement*) - Acordo de nível de service (ANS)

O SLA, também conhecido como Acordo de nível de serviço - ANS, é um acordo entre a área prestadora de serviços e seus clientes. Este acordo deve deixar claro quais serviços estão sendo oferecidos (serviços específicos) e o nível de cada serviço (horas de funcionamento, *downtime*, horário do suporte etc). Geralmente este acordo é colocado na forma de um contrato que deve ser assinado na contratação do serviço. Para banco de dados, em particular, pode-se utilizar um SLA interno, onde o DBA se compromete a dar algum tipo de retorno (*feedback*) ao solicitante. Notem que este retorno não quer dizer, necessariamente, a resolução do problema ou o conserto, mas sim que o DBA está ciente da solicitação.

8) Diagrama de arquitetura

Atualmente é comum encontrar nas empresas diversos ambientes de bancos de dados. Estes ambientes são separados de acordo com a sua finalidade, isto é, seu principal objetivo. Por exemplo, é comum encontrar ambientes de desenvolvimento, onde os programadores/analistas executam diversos testes durante o processo de desenvolvimento, e ambientes de programação, onde os usuários finais trabalham com os dados reais dos sistemas.

Para documentar e organizar o gerenciamento destes ambientes, o DBA deve elaborar um diagrama de arquitetura, que indica, de forma gráfica, quais servidores pertencem ao ambiente de desenvolvimento e ao ambiente de produção, como eles estão localizados em relação aos usuários com informações sobre link, rede, zonas desmilitarizadas (DMZ), *firewalls*, roteadores, etc. Este tipo de diagrama contém informações relacionadas à estrutura arquitetural dos ambientes e é extremamente útil para quem não conhece a organização física e lógica dos componentes da rede e dos servidores. É importante lembrar que este documento pode ser flexível, ou seja, pode incluir

detalhes específicos, como endereços I.P. e senhas, ou apresentar uma visão de alto nível, onde apenas os principais servidores são apresentados.

9) Estratégia de Backup

Todo DBA profissional deve possuir uma estratégia de backup adequada. Esta estratégia deve ser montada de acordo com as necessidades de recuperação e disponibilidade do sistema, ou seja, toda a estratégia de backup vai depender do quanto de *downtime* e tempo de recuperação é aceitável.

É importante documentar a estratégia de backup utilizada, tanto para oficializar este tipo de tarefa como para conscientizar os usuários a respeito do que pode ser recuperado, quando, sob quais condições e a qualidade do que foi recuperado. Geralmente este documento contém todos os bancos de dados envolvidos na estratégia, como o backup será realizado, qual a periodicidade, qual é o procedimento para recuperação, quem são os responsáveis e os recursos envolvidos. Por fim, é importante atualizar este documento conforme as necessidades de disponibilidade e recuperação mudam de acordo com o volume de informações manipuladas pelos sistemas.

10) Procedimento para controle de chamados ou O.S. (ordem de serviço)

Este último item não é exatamente um documento, mas sim um procedimento que deve ser adotado para o controle de solicitações de serviço (ou chamados) ao DBA. Este tipo de controle evita problemas de comunicação entre quem solicitou e que realiza uma tarefa. Deixar este controle apenas a cargo do envio de e-mails é um primeiro passo, mas investir em um sistema para controlar o acesso às pessoas é algo fundamental, uma vez que este procedimento está diretamente relacionado com as regras definidas no SLA. Obviamente, este tipo de controle deve ser utilizado de forma sensata, pois existem diversos tipos de solicitações que podem exigir tratamentos diferentes, como apenas uma olhada no estado de um servidor. Mais uma vez, a idéia aqui é estabelecer um mecanismo de controle tanto para quem solicita a tarefa como para quem a executa.

Outros tipos de documentos são necessários para quem trabalha com desenvolvimento de sistemas. Atas de reunião, manuais de implementação de sistemas e *help-online* são apenas alguns exemplos disto. Em geral, o DBA é o profissional que menos tem que lidar com este tipo de documentação. Apesar disso, é importante que o profissional que trabalha com banco de dados, e também com desenvolvimento de sistemas, tenha consciência que esta documentação não necessariamente é burocracia, mas sim que ela é um artefato extremamente importante para todos os profissionais envolvidos.

Mauro Pichiliani no iMasters.