ISSN 21-76-8498

PRÁTICAS E FERRAMENTAS PARA OTIMIZAÇÃO DO DESEMPENHO DE BANCO DE DADOS SQL SERVER

Renan Gonçalves JAYME[1]
Ana Paula Ambrosio ZANELATO[2]

RESUMO: O presente artigo tem como objetivo descrever algumas práticas e ferramentas que podem ser utilizadas para otimizar o desempenho de consultas em Bancos de Dados Microsoft SQL Server, permitindo assim a execução de forma eficiente de grandes volumes de dados gerenciados pelo Sistema Gerenciador de Banco de Dados, e visando principalmente suprir as necessidades das empresas em tomar decisões de forma rápida e precisa.

Palavras-chave: otimização, desempenho, banco de dados, SQL Server, SQL, Sistema Gerenciador de Banco de Dados.

1 INTRODUÇÃO

Atualmente, as empresas lidam com uma grande quantidade de dados e informações em seus sistemas, com o objetivo de melhorar seus processos, o que torna de extrema importância o uso de um sistema gerenciador de Banco de Dados. Além disso, a competitividade do mercado atual, exige cada vez mais, tomada de decisões mais ágeis e precisas. Neste contexto, os dados precisam ser acessados constantemente, a todo instante, podendo ocorrer lentidão e uma demora no processo da empresa, por isso a questão da performance do Banco de Dados se tornou também de muita importância.

Os Sistemas de Gerenciamento de Banco de Dados (SGBDs), como o SQL Server, têm como objetivo geral armazenar e gerenciar os dados e informações, além de garantir sua disponibilidade de forma rápida e eficaz. Mas para garantir isto, é necessário despender esforços a fim de aperfeiçoar

[1] Discente do 7º ano do curso de Sistemas de Informação das Faculdades Integradas "Antonio Eufrásio de Toledo" de Presidente Prudente. renangj19@gmail.com
[2] Docente do curso de Sistemas de Informação das Faculdades Integradas "Antonio Eufrásio de Toledo" de Presidente Prudente. anapaula@unitoledo.br. Orientador do trabalho.

seu funcionamento, uma vez que desta forma podemos colaborar com a performance.

Os problemas de performance podem estar não só relacionados a infraestrutura, hardware ou sistemas operacionais, mas também no modelo físico e configurações do banco de dados e na forma com que as instruções SQL (Structured Query Language, ou Linguagem de Consulta Estruturada, uma linguagem de pesquisa declarativa padrão para banco de dados relacional) são escritas.

O presente artigo tem como objetivo descrever algumas práticas de otimização do modelo físico e da aplicação além de ferramentas que podem ser utilizadas para aprimorar o desempenho de consultas em Bancos de Dados Microsoft SQL Server, que trata-se de um gerenciador de banco de dados relacional desenvolvido pela Microsoft e amplamente utilizado no mercado comercial.

Portanto, o presente trabalho encontra-se divido em três partes, sendo que a primeira parte aborda a metodologia utilizada, na segunda parte o desenvolvimento desta pesquisa, e na terceira parte a conclusão.

2 METODOLOGIA

Este artigo foi desenvolvido por meio de pesquisas bibliográficas realizadas em artigos científicos de outros autores sobre assuntos relacionados à otimização do desempenho de Bancos de Dados.

O Sistema de Gerenciamento de Banco de Dados SQL Server foi escolhido para exemplificar o uso das práticas e ferramentas de otimização por ser um SGBD usado em um grande número de empresas, além de estar amplamente incluído na grade curricular do curso de Sistemas de Informação, para qual o artigo será utilizado como forma de avaliação para a disciplina de Metodologia do Trabalho Científico, e o autor do presente artigo reter certa experiência com o uso do SGBD.

3 OTIMIZAÇÃO

A necessidade das empresas por Sistemas de Gerenciamento de Bancos de Dados cresce de maneira continua, assim como cresce também o volume de dados a serem gerenciados por estes sistemas e o nível de complexidade de suas aplicações. Desta forma, executar operações eficientemente, sobre este grande volume de dados é fundamental, uma vez que usasse a sua eficiência diante de consultas e alterações para medir o desempenho de um Sistema Gerenciador de Banco de Dados.

De acordo com Carneiro et al (s.d):

> O primeiro caminho para conseguir um desempenho adequado de um sistema de banco de dados é tomar boas decisões durante o projeto desse. Várias considerações deverão ser feitas durante a fase de projeto, entre elas: o volume esperado de dados em cada relação do sistema e quais consultas serão realizadas com mais freqüência. Mas, percebe-se na maioria dos sistemas é que seu real desempenho só pode ser conseguido após algum tempo de uso, e muitas das considerações que os projetistas haviam feito podem mostrar-se incorretas. Portanto, uma fase subseqüente de ajuste do sistema torna-se necessária, com base em dados reais de seu comportamento, com o objetivo de maximizar o desempenho e a estabilidade.

Assim podemos afirmar que sempre serão necessários ajustes no sistema para que este apresente o desempenho e estabilidade desejados, uma vez que de acordo com o tempo de uso as considerações feitas no projeto podem se mostrar erradas.

3.1 OTIMIZAÇÃO DO MODELO FÍSICO DOS DADOS

Um dos fatores que tem maior influência no desempenho de aplicações SQL é a definição de chave primária das tabelas, que é o identificador único de um registro. Outro fator importante é a criação de índices. Um índice é uma estrutura do banco de dados usada pelo servidor para localizar rapidamente uma linha de uma tabela (Loney, 2002, apud Pletsch, 2005). Cada entrada de índice consiste em um valor de chave e um endereço

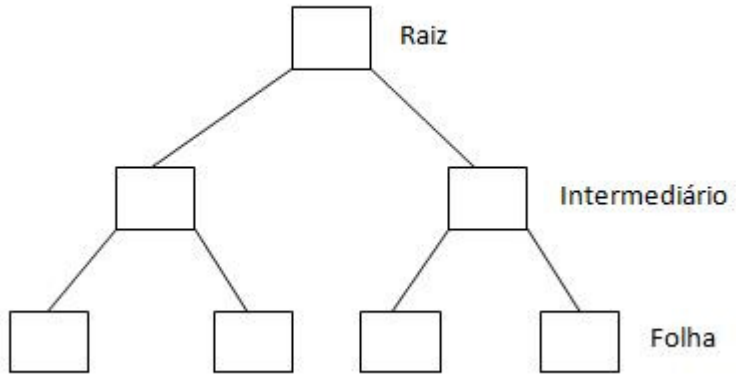
físico do registro. Segundo Pletsch (2005), os índices podem ser divididos em três tipos básicos:

> - **índices de *cluster***: armazenam os valores de chave de uma tabela *cluster*. Uma tabela cluster é criada quando existem tabelas que são acessadas juntas com frequência, fazendo com que elas sejam armazenadas fisicamente juntas;
> - **índices de tabela**: armazenam os valores das linhas de uma tabela juntamente com a localização física na qual a linha está localizada, é o tipo de índice mais utilizado;
> - **índices de bitmap**: tipo especial de índice de tabela, criado para dar suporte às consultas das tabelas grandes com colunas que têm poucos valores distintos.

3.1.1 ÍNDICES

O SQL Server grava as informações dos índices em uma estrutura chamada de B-Tree, que trata-se de uma estrutura que contém um nó-raiz com uma única página de dados, uma ou mais páginas de níveis intermediários e uma ou mais páginas de níveis folhas. Abaixo temos um exemplo desta estrutura.

Figura 1: Estrutura B-Tree.



Fonte: Lopes, s.d.

Segundo Lopes (s.d.)

> Uma busca pelo índice inicia-se no nível raiz percorrendo todas as linhas até achar a cadeia de valores a qual o mesmo se encaixa e

> através do ponteiro pular para a página do nível intermediário que o mesmo se refere. No nível intermediário repete o mesmo processo até achar a cadeia de valores e pular para a página de nível folha conforme o ponteiro. No nível folha novamente repete-se o processo até achar o valor desejado e nesse momento é localizado os dados necessários.

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 necessária a criação de novos índices e quais serão os critérios adotados para a sua criação.

De acordo com Lopes (s.d.), 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 nos registros da tabela. Isto porque o índice é atualizado após a 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.

3.2 OTIMIZAÇÃO DA APLICAÇÃO

Pletsch (2005) afirma que a maior parte dos problemas de desempenho em banco de dados relacionais, assim como no SQL Server, 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 a otimização das consultas.

3.2.1 USO DE FILTROS

De acordo com Araujo (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.

3.2.2 USO DA INSTRUÇÃO COUNT

Araujo (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.

3.2.3 USO DA INSTRUÇÃO IN

Segundo Araujo (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.

3.2.4 USANDO OPERADORES 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.

3.2.5 TEMPDB

O SQLServer utiliza o banco TempDB para realizar varias operações, ele armazena de forma temporária os objetos do banco. Trata-se de um recurso compartilhado entre todos os usuários do sistema, por isso, enquanto um processo utiliza o TempDB, outro processo precisa aguardar. Além disso, a execução de operações longas pode gerar lentidão do sistema, pois exigem um maior tempo de ocupação do TempDB.

Fortes (2013) defende que se existir no sistema vários TempDBs independentes, organizando o uso dos recursos do banco de dados, a chance de concorrência entre processos diminui, ganhando desempenho, segundo ele o ideal é começar com poucos arquivos e monitorar o acesso e uso do TempDB, e assim decidir entre incluir ou não mais um arquivo.

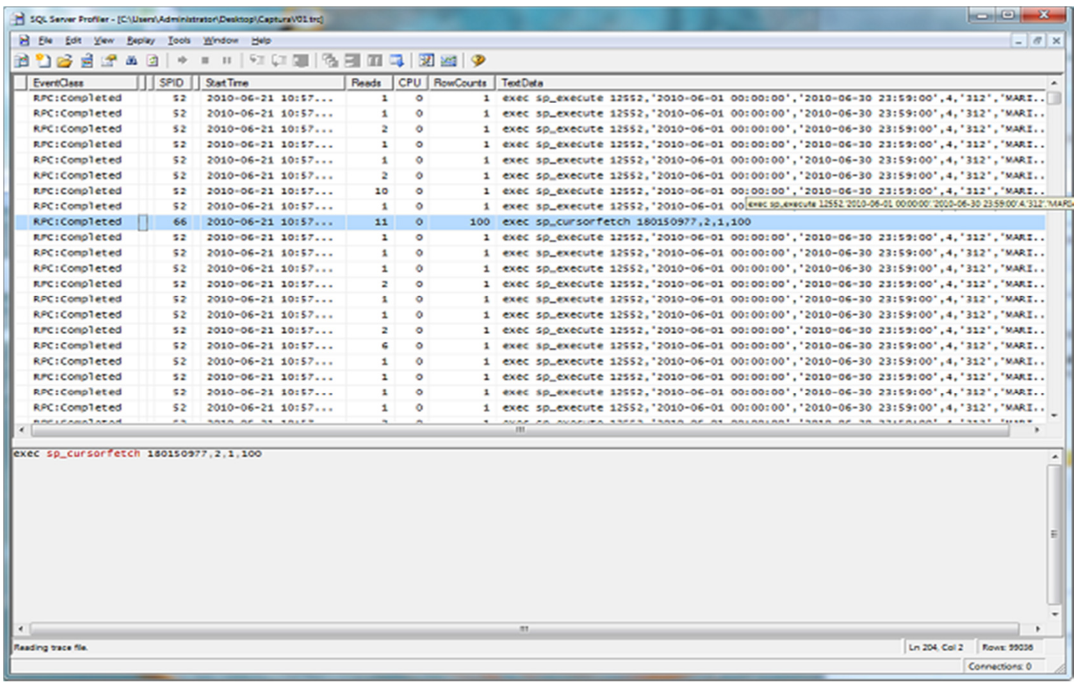
4 EXEMPLOS DE FERRAMENTA

Neste capitulo, citaremos algumas ferramentas do SQL Server, que podem ser utilizadas com o objetivo de auxiliarem na busca pela melhor performance.

4.1 SQL SERVER PROFILER

O SQL Server Profiler é uma interface avançada para criar e gerenciar rastreamentos, produzindo resultados que podem ser analisados. Por exemplo, é possível monitorar um ambiente de produção para analisar quais procedimentos estão afetando a performance devido à lentidão na execução. É possível capturar e salvar os dados sobre cada evento em um arquivo, ou tabela para análise posterior. Esta ferramenta possui uma interface gráfica para o usuário, que o rastreamento do SQL pode usar para monitorar uma instância do Mecanismo de Banco de Dados ou do Analysis Services (conjunto de serviços para gerenciamento de dados em data marts e data warehouses). Além disso, o SQL Server Profiler também oferece apoio à execução de auditoria das ações relacionadas à segurança, executadas em instâncias do SQL Server, para análise posterior por um administrador de segurança (http://msdn.microsoft.com/pt-br/library/ms181091.aspx). Segue abaixo figura que ilustra a ferramenta:

Figura 2: traza-SQL-Profiler-con-cursores-small

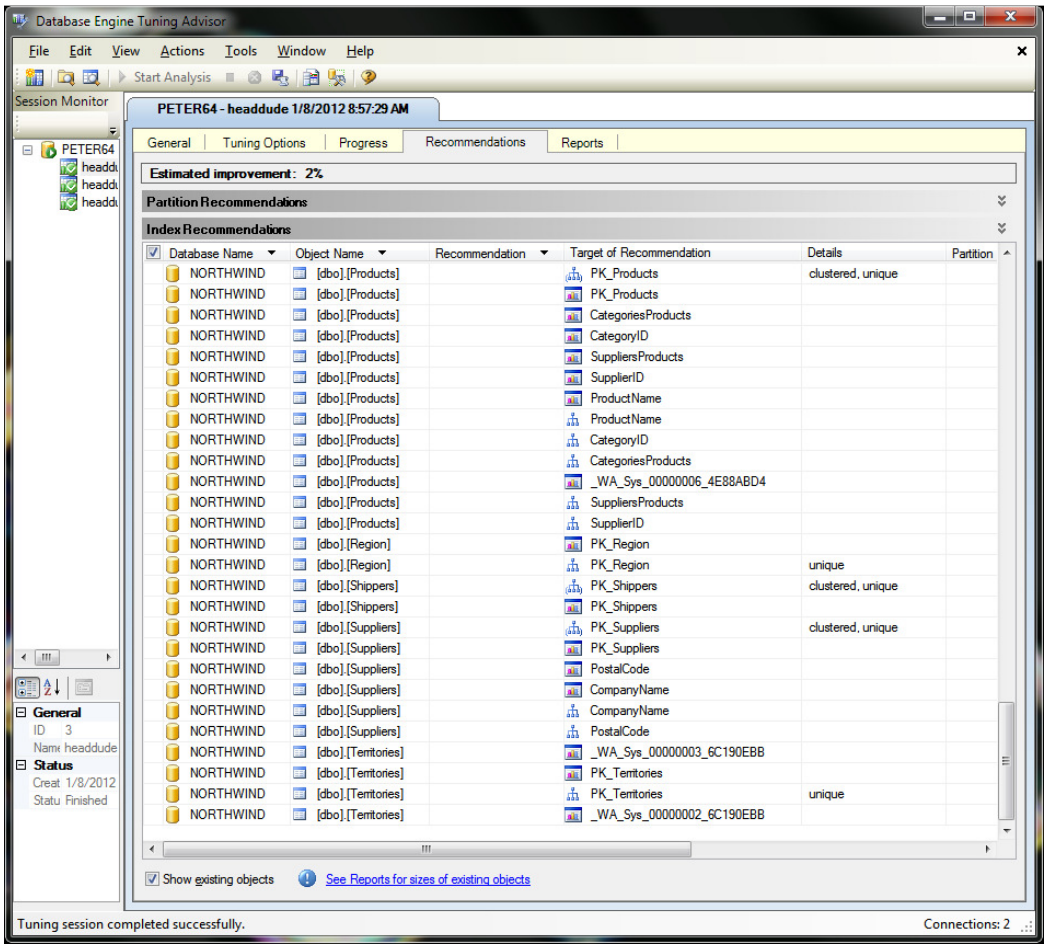


Fonte: Informações sobre o Serviço SolidQ TSQL-CSI

4.2 DATABASE ENGINE TUNING ADVISOR

Segundo (Moraes, 2007, apud Souza, 2009), o Database Engine Tuning Advisor (DTA) é uma importante ferramenta que ajuda os Administradores de Banco de Dados na seleção adequada do projeto físico de uma banco de dados Microsoft SQL Server. Pode ser utilizado para ajustar consultas simples que possam estar mal escritas e que possam proporcionar problemas de desempenho, ou mesmo para consultas complexas e que necessitam de um maior conhecimento do Administrador de banco de Dados em relação ao projeto físico do banco de dados. Segue abaixo figura ilustrando a ferramenta:

Figura 3: 1145921998_northwind2

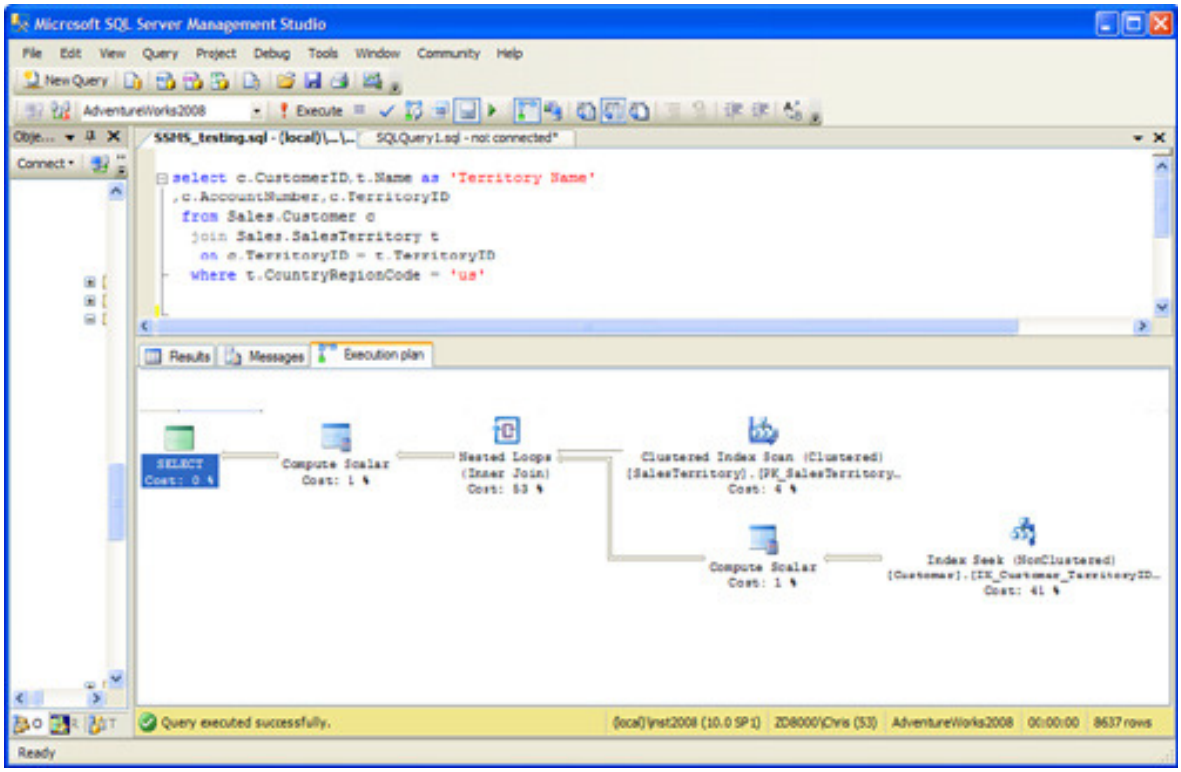


Fonte: Peter Bromberg

4.3 EXECUTION PLAN

Segundo (Moraes, 2007, apud Souza, 2009), o Execution Plan fornece um diagrama visual da execução de determinada consulta, quais os índices utilizados, além de outras informações que podem auxiliar os Administradores de Banco de Dados no processo de otimização de índices ou outros dados demandados pelo SQL Server para aprimorar o desempenho das consultas. Segue abaixo figura ilustrando a ferramenta:

Figura 4: Displaying an actual execution plan in Query Editor.



Fonte: Microsoft SQL Server 2008 R2 : SQL Server Management Studio - Development Tools (part 2)

5 BOAS PRÁTICAS EM CONSULTAS T-SQL

Durante o desenvolvimento de instruções SQL, algumas praticas podem colaborar com o desempenho da consulta. Abaixo, citamos algumas:

BOAS PRÁTICAS EM CONSULTAS T-SQL
Procurar escrever consultas levando em consideração boas práticas de desenvolvimento, tais como indentação e comentários (/* */ ou --).
Informar os campos que devem aparecer na consulta, evitando utilizar o famoso select * from <NomeDaTabela>.
Quando for utilizar operadores de comparação, evitar usar NOT em condições de pesquisa, pois podem diminuir a velocidade de recuperação de dados porque todos os registros em uma tabela são avaliados.
Usar de forma restritiva a cláusula WHERE, pois é uma grande causadora de problemas em relação ao processador
Para testes de existência é sempre mais eficiente utilizar EXISTS ao invés de COUNT. Quando se utiliza COUNT, o banco de dados não sabe que está sendo feito um teste de existência e continua pesquisando todas as linhas qualificadas.Já utilizando EXISTS, o banco de dados sabe que é um teste de existência e interrompe a pesquisa quando encontra a primeira linha qualificada.
Analisar a necessidade de utilização de CHAR e VARCHAR.

Fonte: Souza et al (2009)

Tabela 3: Boas Práticas na Elaboração de Consultas

6 CONCLUSÃO

Em virtude dos fatos apresentados neste artigo, chegamos à conclusão que há várias técnicas que podemos utilizar a fim de otimizar o desempenho das consultas de um banco de dados SQL Server, e que elas são necessárias, pois a demanda de acesso a dados hoje vem crescendo a cada dia, e o Sistema Gerenciador de Banco de Dados deve responder de forma ágil e confiável diante das requisições dos usuários. Mas ao mesmo tempo, é necessário atentar-se a verdadeira necessidade do uso da prática ou ferramenta para otimização, e se ela está sendo aplicada da forma correta, pois caso seja administrada de forma errada, ao invés de melhorar a performance do banco de dados, pode tornar seu desempenho ainda pior.

REFERÊNCIAS BIBLIOGRÁFICAS

ALENCAR, Thiago Carlos de. **Dicas de Administração de Consultas.** Disponível em: http://www.oficinadanet.com.br/artigo/529/dicas_de_otimizacao_de_consultas. Acesso em: 31 mai. 2013.

ALLCOMPUTERS. **Microsoft SQL Server 2008 R2 : SQL Server Management Studio - Development Tools (part 2).** Disponível em: http://allcomputers.us/windows_server/microsoft-sql-server-2008-r2---sql-server-management-studio---development-tools-(part-2).aspx. Acesso em: 07 jun. 2013.

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

CARNEIRO, Alessandro Pinto; MOREIRA, Julinao Lucas; FREITAS, André Luis Castro de. **TUNING - Técnicas de Otimização de Banco de Dados, Um Estudo Comparativo: Mysql e Postgresql.** Trabalho de Graduação **-** Centro de Ciências Computacionais – Universidade Federal do Rio Grande (FURG).

FORTES, Charles. **Melhorando o desempenho de seu banco de dados com o TempDB.** Disponível em: http://www.100loop.com/destaque/melhorando-o-desempenho-de-seu-banco-de-dados-com-o-tempdb/ Acesso em: 06 jun. 2013.

LOPES, Nicholas. **Índices no SQL Server.** Disponível em: http://www.devmedia.com.br/indices-no-sql-server/18353. Acesso em: 31 mai. 2013.

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

SOLIDQ. **Informações sobre o Serviço SolidQ TSQL-CSI.** Disponível em: http://www.solidq.com/br-pt/services/sqlserver-relacional/Pages/Informa%C3%A7%C3%B5es-sobre-o-Servi%C3%A7o-SolidQ-TSQL-CSI.aspx. Acesso em: 06 jun. 2013.

SOUZA, Ana Paula dos Santos; CAMPOS, Bruno Fidelis, DIAS, Carla Glênia Guedes; ALVES, Michel Batista; VIEIRA, Carlos Eduardo Costa, CARELLI, Flávio Campos. E LUIZ FABIANO COSTA DE SÁ. Tuning em Banco de Dados. **Cadernos UniFOA**. Volta Redonda, ano IV, n. 10, agosto. 2009. Disponível em: http://www.unifoa.edu.br/portal_pesq/caderno/edicao/10/19.pdf.