OTIMIZAÇÃO DE CONSULTAS NO SGBD ORACLE 11G

BRENO MARCELO DE SOUZA[1]

IREMAR NUNES DE LIMA[2]

Resumo: Este artigo apresenta diversas técnicas e ferramentas para otimizar consultas SQL no Sistema Gerenciador de Banco de Dados ORACLE 11g.

Palavras-chave: Banco de Dados, Otimização, SQL, Oracle 11g.

1.0 INTRODUÇÃO

Um fator muito importante para garantir a continuidade do negócio das empresas nos dias atuais é sua capacidade de analisar, planejar e reagir com rapidez às mudanças ocorridas no mercado. Para isso, é importante que as organizações disponham cada vez mais de informações para o auxílio à tomada de decisões, otimizando seus processos.

No ambiente competitivo em que as empresas estão inseridas, onde a globalização vem derrubando barreiras comerciais, extrair e inserir informações confiáveis e em menor tempo possível passa a ser recomendável para aquelas empresas que buscam permanecer no mercado.

Neste contexto, é indicada a aplicação de novas técnicas e a sua necessidade começa a ser sentida a partir do momento em que as empresas adotam uma postura de trabalho mais voltada à gestão da informação. Dessa forma, a otimização de um sistema de banco de dados se faz necessária pelo fato de estar fortemente ligada ao tempo de resposta e,

[1] Analista de Sistemas e Especialista em Banco de Dados e Business Intelligence (breno.m.souza@gmail.com).
[2] DBA, Mestre em informática e Professor do Centro Universitário Newton Paiva (iremar.prof@uol.com.br).

conseqüentemente, é o elemento mais perceptível ao usuário final. Sabe-se que muito dos gargalos em um sistema de banco de dados derivam de consultas ou modelo de dados mal estruturado.

O objetivo desse trabalho é demonstrar que mesmo um SGBD de grande porte como Oracle 11g, que utiliza mecanismos de otimização automática, pode escolher planos de execução para sentenças SQL que não sejam as melhores. As consultas podem ser melhoradas com a aplicação manual de técnicas que possibilitem um desempenho maior com menor custo, provendo às empresas a capacidade de extraírem as informações mais rápidas do banco de dados, independente da ferramenta ou do grau de complexidade exigido nas consultas.

O restante do artigo está organizado da seguinte forma: na seção 2 serão apresentadas algumas características da linguagem SQL. Na seção 3 será abordado o Oracle 11g, descrevendo sua arquitetura e os componentes envolvidos na execução da instrução SQL, explicando o plano de execução e estatística. Na seção 4 mostrará as técnicas de boa prática na otimização de instrução SQL, focando em modelagem de dados, variáveis *bind*, índices e ajuste de SQL. Na seção 5 é apresentada a conclusão deste trabalho.

2.0 CARACTERÍSTICAS DA LINGUAGEM SQL

SQL (Structured Query Language) é uma linguagem de alto nível para manipulação de dados dentro do modelo relacional. É de tal ordem sua importância para a indústria dos bancos de dados relacionais que ela acabou por se tornar o mecanismo mais popular de acesso aos grandes bancos de dados cliente/servidor.

A linguagem SQL é dividida em cinco tipos de instruções, descritas a seguir (PRICE, 2009):

- Data Query Language (DQL): recuperam linhas armazenadas nas tabelas do banco de dados e é representada pela instrução Select.
- Data Manipulation Language – (DML): modificam o conteúdo das tabelas. Existem três instruções DML: Insert, Delete e Update.
- Data Definition Language – (DDL): definem as estruturas de dados, como as tabelas, que compõem um banco de dados. Existem cinco tipos básicos de instruções DDL : Create, Alter, Drop, Rename e Truncate.
- Transaction Control - (TC): registram permanentemente as alterações feitas em linhas ou desfazem essas alterações. Existem três instruções TC: Commit, Rollback e Savepoint.
- Data Control Language –(DCL): alteram as permissões nas estruturas de banco de dados. Existem duas instruções DCL: Grant, Revoke.

3.0 ARQUITETURA DO ORACLE 11G

O conhecimento da arquitetura interna do SGBD ORACLE é importante para a compreensão das técnicas de otimização do produto. O SGBD Oracle é composto de duas entidades: o banco de dados e a instância. O banco de dados é formado por arquivos no disco, ou seja, a estrutura física. Já a instância é formada por estruturas de memória e os processos. No processo de inicialização, a instância é criada e inicializada primeiro que o Banco de Dados. Eles são separados, porém conectados.

Segundo WATSON (2010), uma instância Oracle é composta por um bloco de memória compartilhada conhecida como área global de sistema (SGA), e uma área de memória não compartilhável associado a cada processo do servidor, conhecida com área global de programa (PGA).

A PGA é o buffer de memória que contêm dados e algumas informações de controle de uma sessão de um usuário. Dentro da PGA existem 2 estruturas : uma contendo variáveis de sessão e outras informações, outra contendo dados sobre a sessão do usuário, tais como áreas privadas SQL.

Já a estrutura de dados da SGA é composta por:

- O cache de buffer do banco de dados: concentra todos os blocos lidos para aumentar a velocidade de leitura evitando I/O.
- O Redo log buffer: contém informações o suficiente para refazer o banco de dados. É uma área de segurança, tudo é gravado lá independente de *Commit*. É uma área que pode causar problemas de performance.
- O shared pool: é uma área de cache especializada em armazenar os seguintes dados em cache: código PLSQL compilado, planos de execução, dicionário de dados, permissões e etc...
- Um large pool: é uma área opcional que, se criada, será usada automaticamente por vários processos que ocupariam a memória do shared pool. Ela é destinada a I/O, backup e restore.
- Um Java pool: é uma área destinada para executar procedures java armazenada no banco de dados. Sem ela a Console não inicia.
- Um Streams pool: é uma área destinada extrair vetores de alteração do redo log,e a partir deles, reconstrói as instruções que foram executadas ou instruções que teriam o mesmo efeito.

As estruturas físicas que compõem um banco de dados Oracle são :

- Data File : são os arquivos de dados armazenados.
- Redo Log File: guardam informações para restaurar o banco de dados.

- Control File: armazenam informações sobre as estruturas físicas do banco de dados (nome, checkpoints, informações sobre backups, localização física,...). Todos os data files e redo log files são identificados no control file, bem como o nome do banco de dados.

3.1 Componentes Envolvidos na Execução da Instrução SQL.

Quando submetemos um comando de SQL para o banco de dados, quatro etapas são realizadas para a implementação do comando: Parse, Bind (opcional), Execute e Fetch. A forma como estes componentes interagem entre si é ilustrada pela Figura 1.

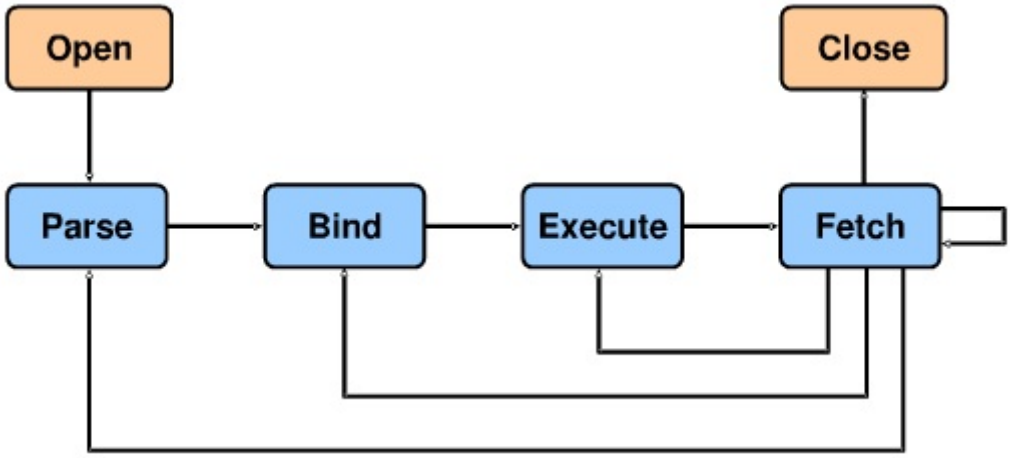

Figura 01: Fluxo de execução de uma consulta SQL
Fonte: http://www.cs.tau.ac.il/~boim/courses/databases2011/slides/moreinfo/SQL%20tuning.pdf

Prepare (Parse)

Durante a fase prepare, a instrução SQL é enviada pelo usuário para o servidor de banco de dados para ser analisada. Em seguida, é carregada na área de compartilhamento do SQL. Este processo de análise e preparação do SQL consiste em:

- Validação da sintaxe do comando SQL.
- Pesquisa a existência do comando em memória – o Oracle verifica se o comando de SQL que está sendo analisado já foi submetido anteriormente e se o resultado desta submissão ainda está em memória. Caso o comando seja encontrado, significa que

o plano de execução já está traçado, não havendo necessidade de refazermos esta ação; sendo assim, será iniciada a fase EXECUTE.

- Pesquisa o dicionário de dados - caso o comando de SQL não seja encontrado na memória, o Oracle verifica no dicionário de dados se existem as tabelas e colunas mencionadas no comando e se as permissões para acesso aos objetos desejados são suficientes.
- Construção do plano de execução – após a pesquisa no dicionário de dados o otimizador poderá traçar um caminho de acesso para cada tabela presente no comando, montando um plano de execução para obtenção e/ou atualização dos dados do comando SQL submetido.
- Alocação da memória – com o plano traçado, o Oracle inclui este plano na memória para que sua execução seja possível e para permitir que este plano seja reaproveitado por outro comando idêntico ao atual.

A fase de análise é executada apenas uma vez, independentemente da qualidade de vezes que a instrução é executada, desde que a instrução analisada esteja na área de compartilhamento do SQL.

Bind (Variáveis de Ligação)

É opcional, caso haja utilização de variáveis na consulta SQL. Com uso de variáveis bind para representar valores de coluna, pode-se garantir que uma instrução seja idêntica, ocorrendo a reutilização de uma instrução SQL já armazenada em memória, e a redução do tempo de execução, pois o plano de execução já está traçado, não havendo necessidade de refazer o *parse*.

Execute

Nesta fase, o servidor de banco de dados já detém todos os recursos e informações necessárias para executar a instrução e aplicar o plano de execução, podendo fazer leituras

físicas ou lógicas. Caso a instrução seja um comando select ou insert, nenhuma linha precisa ser bloqueada, já que nenhum dado está sendo alterado. Caso a instrução seja update, delete, select for update ou with lock, todas as linhas afetadas pela instrução são bloqueadas, impedindo que outros usuários alterem-nas até o próximo commit, rollback ou savepoint da transação, garantindo assim, a integridade do dado.

Fetch

Nesta fase, as linhas que satisfizerem ao resultado de uma consulta são recuperadas e enviadas para a aplicação que as requisitou.

3.2 Plano de Execução

O software de banco de dados Oracle usa um subsistema conhecido como Otimizador para gerar o caminho mais eficiente para acessar os dados armazenados nas tabelas. O caminho gerado pelo otimizador é conhecido como plano de execução.

Para executar um comando SQL (DML), o Oracle pode ter de executar diversos passos. Cada um destes passos pode recuperar, fisicamente, linhas das tabelas referenciadas no comando.

Para cada tabela envolvida no comando SQL haverá um caminho de acesso para obtenção dos dados daquela tabela, que é apresentado abaixo:

- **Full table scans**
  - ✓ Lê todos blocos da tabela usada para armazenar linha. Faz I/O multiblocos.
  - ✓ Pode ser causado por: falta de índice, pouca seletividade das colunas e em tabelas pequenas, mesmo com índice.
- **Row ID Scans**
  - ✓ Acessa linha e reconhece o RowID , que são pseudocolunas que retorna o endereço da linha.
  - ✓ Acesso pela chave (**PK).**

- ✓ Melhor forma de acesso aos dados, pois são identificadores únicos para linhas em uma tabela.

- **Index Scan**
  - ✓ Acesso aos dados usando-se um índice.

  Principais tipos de índices existentes no Oracle:

  - ✓ Index Join: Join entre índices.
  - ✓ Unique Scan: PK (a = 10) é o acesso a um índice B-Tree sobre colunas únicas (primary keys ou uniques) para recuperação do RowId de um registro.
  - ✓ Range Scan: por faixa (a > 10 < 20) ou quando não se tem chave única
  - ✓ Full-scan: quando ordena pelo campo indexado
  - ✓ Fast-Full scan: sem garantia de ordem

- **Nested Loop Joins**
  - ✓ Para cada linha da tabela externa, faz uma busca na tabela mais interna.

- **Hash Joins**
  - ✓ Aplicar o algoritmo de hash join em um inner join de duas relações funciona da seguinte maneira: primeiro, é preparada uma hash para a relação menor, aplicando uma função hash para cada linha , na columa a ser usada no join. Então, a relação maior é escaneada, em busca das linhas relevantes, procurando pela hash table. Usado com INNER JOIN, OUTER JOIN.
  - ✓ Na junção hash (hash join), a fonte de dados interna é colocada em uma tabela hash, utilizando a chave da junção. Cada registro da fonte de dados externa também é colocada na tabela hash, gerando os registros que atendem à junção. Caso o número de registros da fonte de dados interna seja muito grande, são criadas partições, contendo apenas parte dos registros. Cada registro da fonte de dados externa será colocado na tabela hash a fim de verificar os registros que

atendem as condições de junção. Este procedimento é realizado para cada registro da fonte de dados externa. O custo do “hash join” pode ser obtido pela expressão (custo de acesso da fonte externa * número de partições) + custo de acesso da fonte interna.

  - ✓ Quando há relacionamento entre duas tabelas, utilizando full-table-scan.

- **Sort-Merg Joins**
  - ✓ Usado quando não se tem índices.
  - ✓ Cada uma das tabelas ou conjuntos resultados é ou já deve estar ordenado.
  - ✓ A partir disto, é realizado o percurso balanceado de ambas as tabelas.
  - ✓ Normalmente o hash join possui desempenho superior, mas o sort merge join é particularmente útil em junções por desigualdade (<>, <, >, <=, >=, like, etc.).

3.3 Estatística

Nos dias de hoje os bancos de dados Oracle vivem e respiram estatísticas (FREEMAN,2009).

O otimizador recolhe os dados estatísticos sobre a base de dados e objetos no banco de dados. Essas estatísticas são utilizadas pelo otimizador para escolher o melhor plano de execução de cada comando SQL.

Para diagnosticar problemas de desempenho com eficiência, a estatística deve estar disponível. O Oracle gera muitos tipos de estatísticas, como: estatística de tabelas, colunas, índices e sistema.

A partir da versão 10g, já era possível recolher automaticamente as estatística, algo que antes era feito manualmente, reduzindo com isto significativamente as chances de algum objeto ficar sem estatísticas ou com as mesmas obsoletas causando algum problema no plano de execução. Porém em alguns casos, se faz necessário a coleta manual, como por exemplo, em tabelas que são modificadas em operações com volume de dados muito

grande e são acessadas de forma significativa. Já o Oracle 11g oferece novas funções associada às estatísticas, como: estatísticas pendentes e publicadas, recuperando estatísticas anteriores e estatísticas estendidas (FREEMAN,2009).

As estatísticas pendentes e publicadas oferece a opção de publicar estatísticas após a sua coleção, sendo o comportamento padrão, ou você pode ter estatísticas recém coletadas salvas em um estado pendente (FREEMAN,2009).

Para determinar se o banco de dados irá publicar as estatísticas quando elas são coletadas ou se elas serão mantidas em um estado pendente, você deverá usar a função dbms_stats.get_prefs, retornando TRUE as estatísticas serão publicadas automaticamente ou FALSE elas não serão publicadas, sendo valor default TRUE. Se você determinar não publicar a estatística pendente, basta removê-las através do procedimento dbms_stats.delete_pending_stats, tendo a opção de remover a estatística pendente do banco inteiro ou somente de um esquema ou de um objeto especifico em um esquema. E para determinar se uma estatística pendente será ou não publicada, o mesmo poderá ser feito através do procedimento dbms_stats.set_schema_prefs para esquema ou dbms_stats.set_table_prefs para uma tabela, acrescentando a opção FALSE ou TRUE. E o procedimento dbms_stats.publish_pending_stats ira publicar todas as estatísticas pendentes marcadas para publicar, podendo fazer isso para um banco inteiro, ou para uma esquema ou até mesmo um objeto especifico em um esquema. (FREEMAN,2009).

Uma das vantagens da coleta e publicação, que permite uma coleta quando o sistema é menos usado, por exemplo, á noite e as publicando pela tarde. Se acontecer algo errado, basta restaurar a estatística antiga, e ainda, se você quiser testar as estatísticas coletadas antes de publicá-las, existe um parâmetro optimizer_use_pending_statistics, que fará que o otimizador sempre use as estatísticas pendentes se houver em vez das estáticas publicadas. Caso não haja estáticas pendentes o otimizador irá usar as publicadas, e se houver algum

problema basta mudar esse parâmetros para o otimizador usar as estatísticas publicadas, sendo essa opção bastante útil para testar como o sistema vai reagir as novas estatísticas.

Já recuperando estatísticas anteriores o Oracle 11g permite que você restaure versões anteriores de estatísticas, através de vários procedimentos no pacote dbms_stats, com base em um timestamp especifico. O Oracle irá administrar o repositório de estatísticas históricas, removendo a estatísticas antigas regularmente, de modo padrão a cada 31 dias (FREEMAN, 2009).

As estatísticas estendidas fornecem a habilidade para juntar estatísticas adicionais, como as estatísticas de múltiplas colunas e estáticas de expressão (FREEMAN, 2009).

Antes do Oracle 11g, não existia estatísticas de múltiplas colunas não havendo modo algum para compreender a relação dos dados dentro de múltiplas colunas de uma cláusula where. Agora com as estatísticas de múltiplas colunas podemos fornecer ao Oracle uma melhor informação para que ele possa saber que essas colunas são agrupadas em conjunto, e para que possa gerar um plano de execução que reflita mais precisamente a realidade da consulta (FREEMAN, 2009).

Já as estatísticas de expressão permitem incluir funções como predicativo na cláusula where, e determinar a seletividade dessa função. Lembrando que essa estatística não pode ser aplicada se você tiver um índice baseado na função dessa estatística de expressão que você está querendo criar (FREEMAN, 2009).

Teste cuidadosamente todas as funções novas e certifique-se que elas estejam funcionando, da maneira que você espere que elas funcionem antes de aplicá-las no banco de produção.

4.0 Técnicas de Otimização de Instruções SQL

A otimização é o processo de escolha do modo mais eficiente para a execução de um comando SQL.

Podem existir diferentes formas para o Oracle executar um comando SQL: de acordo com a ordem em que as tabelas ou índices sofrerão acesso, de acordo com as restrições estabelecidas, de acordo com a quantidade de linhas das tabelas envolvidas, de acordo com os índices disponíveis, etc.

De acordo com VENNAPUSA (2004), são vários os fatores que podem influenciar no custo e velocidade das consultas SQL: criação e deleção de índices, alteração de parâmetros, remodelagem física de tabelas, geração de estatísticas, reescrita de códigos SQL, entre outros. Não existindo uma fórmula precisa para aumento de desempenho (VENNAPUSA, 2004).

As consultas são objetos de constantes revisões e otimizações, visto que é o ponto crucial em um sistema de banco de dados, pois constituem a maior parte das transações envolvidas em uma aplicação.

4.1 Modelagem de Dados

A modelagem de dados é importante para o sucesso da aplicação. Isto deve ser feito de uma maneira eficiente e precisa para representar a regra de negócio. Deve-se aplicar esforços para modelagem das entidades de maior impacto, ou seja, as utilizadas com maior frequência. Utilizar ferramentas de modelagem pode ajudar nas definições dos modelos e agilizar na definição dos protótipos (VENNAPUSA, 2004).

Normalizando tabelas, evitam-se duplicações de registros, pelo menos até a terceira forma normal. Quando os dados são normalizados, você tem uma imagem clara das chaves e relacionamentos, tornando mais fácil realizar a próxima etapa de criação de tabelas, constraints e índice. Um bom modelo, em última analise, significa que as suas transações serão interpretadas de modo mais eficiente (VENNAPUSA, 2004).

4.2 Variáveis Bind

Toda instrução SQL submetida ao banco de dados Oracle é colocada em cache. Uma instrução SQL colocada no cache é reutilizada se uma instrução idêntica é enviada para o banco de dados. Quando ocorre a reutilização de uma instrução, o tempo de execução é reduzido, pois o plano de execução já está traçado, não havendo necessidade de refazer o Parse. Entretanto, a instrução SQL deve ser absolutamente idêntica para ser reutilizada. Isso significa que:

- Todos os caracteres na instrução SQL devem ser iguais.
- Todas as letras na instrução SQL devem ter a mesma caixa.
- Todos os espaços na instrução SQL devem ser iguais.

Você pode garantir que uma instrução seja idêntica utilizando variáveis bind para representar valores de coluna. Elas funcionam como parâmetros em instruções SQL, possibilitando a atribuição de valores dinâmicos nos comandos SELEC, UPDATE, DELETE e INSERT. Com a utilização de variáveis Bind o Oracle faz o reuso de instrução SQL já armazenada em memória. Dessa forma, variáveis bind servem de ponte para a execução de uma instrução SQL, já preparada na memória do servidor, onde são enviadas apenas os valores nela contido.

4.3 Índices

A criação de índices deve ser avaliada de acordo com a frequência de uso de determinados dados e tabelas. Um índice pode ser um instrumento chave para atender aos objetivos da otimização. Entretanto, o mesmo pode ser capaz de interferir no desempenho de forma negativa. Uma boa regra geral segundo PRICE (2009) é: crie um índice quando uma consulta recuperar até 10% do total de linhas em uma tabela.

Isso significa, que o critério para a escolha de índices é a seletividade, portanto quando maior for a seletividade melhor será o índice. Uma coluna que contém valor exclusivo para cada linha (por exemplo, um número de CPF), é um atributo candidato á indexação.

Assim, os índices são criados com intuito de eliminar a leitura de toda a tabela na intenção de encontrar o registro, mas cuidado, possuir mais índices em uma tabela não significa que as consultas serão aceleradas. Cada operação DML que seja submetida a commit em uma tabela com índices significa que os índices devem ser atualizados. Quando mais índices associados a uma tabela você tiver, maior será o esforço realizado pelo Oracle para atualizar todos os índices após uma instrução DML.

O Oracle possui diversos tipos de índices, como os normais (árvore B-Tree é a mais utilizada), índices Bitmap que armazenam os rowids juntamente com um valor chave em formatos de bits, índices particionados, índices baseados em funções, índices de chaves reversas e índices IOT.

4.4 Ajuste de SQL

Segundo PRICE (2009), uma das principais vantagens da linguagem SQL é que você não precisa dizer ao banco de dados exatamente como ele deve obter os dados solicitados. Basta executar uma consulta especificando as informações desejadas e o software de banco de dados descobre a melhor maneira de obtê-las.

Através do estudo do desempenho e otimização de consultas SQL, procuramos minimizar o tempo de resposta do servidor de banco de dados, podendo às vezes, diminuir o tempo de execução de suas consultas. A seguir serão apresentadas algumas boas práticas na escrita da instrução SQL, buscando a melhoria do desempenho dessas instruções.

- Evite utilizar “Select *”, pois quando você faz isso, o Oracle tem que ir no Dicionário de dados, ler a tabela DBA_TABLE_COLUMN para saber quais colunas pertencem essa tabela e retornar as informações, causando um trabalho adicional desnecessário, além de ler a página de dados de cada linha para obter os valores dos atributos que não fazem parte do índice (VENNAPUSA, 2004).

- Use referência de coluna totalmente qualificada ao fazer joins, pois economiza tempo ao gerenciador para localizar a tabela que pertence o campo (PRICE , 2009).

  Exemplo:

```
Select p.name, pt.name, p.description, p.price
From products p, product_types pt
Where p.product_type_id=pt.product_type_id
And p.product_id=1;
```

- Use expressões case, em vez de várias consultas, quando precisar efetuar muitos cálculos nas mesmas linhas em uma tabela (PRICE , 2009). Exemplo:

```
Select
  Count(Case When price < 13 then 1 else null end) Low,
  Count(Case When price between 13 and 15 then 1 else null end) med,
   Count(Case When price > 15 then 1 Else null end) high
From products;
```

- Use “Exists” em vez de “IN”. O “Exists” verifica apenas a existência da linha, enquanto o “IN” verifica valores reais. Normalmente o “Exists” oferece melhor desempenho do que “IN” com subconsultas. Porém, se você utilizar “IN” em uma subconsulta e comparar o campo pesquisado dentro da subconsulta, o “IN” oferece melhor desempenho que o “Exists” (PRICE ,2009). Exemplo:

```
Select a.product_id,a.name
From products a
Where a.product_id in (select b.product_id From purchases b
                       Where a.product_id=b.product_id);
```

- Se a Aplicação valida a entrada nos campos, retirar as Constraints de Check na tabela, pois economiza tempo para o gerenciador e evita o consumo de recurso desnecessário, pois não haverá necessidade de consulta ao dicionário de dados, para validar a entrada (VENNAPUSA, 2004).

- Evitar a utilização de “%TYPE”, pois o Oracle terá que consultar no dicionário de dados o tipo dessa coluna e retornar a informação, causando um trabalho adicional desnecessário (VENNAPUSA, 2004).
- Evitar a utilização de operadores como NOT, NOT EXISTS, NOT LIKE, NOT IN, LIKE “%X” ou <> . Nesses casos o índice não é utilizado, pois a pesquisa não pode ser limitada, sendo necessário avaliar todas as linhas (PRICE,2009).
- Utilização de “HINTS”, que tem o como principio, passar dicas para o otimizador influenciando na escolha do plano de execução. A dica correta pode melhorar o desempenho de uma instrução SQL. Você pode verificar se uma dica foi eficiente avaliando o plano de execução de uma instrução SQL com ou sem Dica (PRICE, 2009).

- Evitar o uso de funções na clausula WHERE, pois aumenta o tempo de execução das Instruções SQL. Toda vez que houver uma função na coluna, o índice não será usado (PRICE, 2009). Exemplo:

  Use:

  Where NOME=upper(‘breno’);

  Ao invés de:

  Where lower(NOME)= ‘breno’;

  Ideal:

  Where NOME=’BRENO’;

- Evitar fazer comparações com dados incompatíveis. O Oracle converte automaticamente os campos char e number, mas para isso é consumido recurso desnecessário, pois o mesmo terá que consultar no dicionário de dados o tipo da

coluna e tentar fazer o conversão utilizando os comandos To_char(), To_Number() (VENNAPUSA, 2004).

- Na clausula “Where”, sempre que for possível escolher os campos que fazem parte da chave primária. E sempre colocar as condições mais restritivas por ultimo na clausula “Where”, pois a maioria dos otimizadores lê uma consulta da parte inferior da clausula “Where” para cima (PRICE, 2009).
- Usar “UNION ALL” em vez de “UNION” quando for possível, pois o “UNION ALL” obtém todas as linhas recuperadas por duas consultas, incluindo as linhas duplicadas se houver. Já “UNION” remove as linhas duplicadas se houver, degradando o desempenho da consulta, pois mesmo não havendo linhas duplicadas ele irá verificar (PRICE, 2009).

5.0 CONCLUSÃO

A otimização de um sistema no banco de dados deve ser considerado desde o inicio do projeto (modelagem do banco de dados), com intuito de conseguir resultados mais satisfatórios, evitando retrabalho futuro com correção de problemas que inviabilizem o desempenho esperado, sabendo-se que muito dos gargalos em um sistema de banco de dados derivam de consultas ou modelo de dados mal estruturados.

Existem muitos fatores que podem influenciar o problema de desempenho, existindo casos complexos que fogem a regra mesmo utilizando todas as técnicas e recursos disponíveis, tornando o processo de otimização de consultas uma tarefa não muito fácil. No entanto, apesar dos grandes progressos do Oracle 11g já realizou, utilizando mecanismos de otimização automática e novas funções de estatísticas, ele pode escolher planos de execução para sentença SQL que não sejam as melhores e que não satisfaçam a nossa expectativa, porém quem detém o conhecimento da informação ainda é o analista e as decisões de acesso são de responsabilidade de quem está programando, tornando-se

necessário o conhecimento de acessos aos dados da Oracle e como influenciá-lo obtendo comandos SQL mais eficientes.

O processo de otimização é algo que deve ser constante e implantado de forma proativa, pois pequenos ajustes acarretam grandes saltos de desempenho, entretanto, todas as alterações devem ser premeditadas de modo a não prejudicar outros aspectos do sistema.

REFERÊNCIAS

PRICE, Jason. ***Oracle Database 11g SQL:*** *Domine SQL E PL/SQL no banco de dados Oracle*.Tradução João Eduardo Nóbrega Tortello. Porto Alegre,RS: Bookman, 2009.

WATSON, John. ***OCA Oracle Database 11g:*** *Administração I.* Tradução Altair Caldas Dias de Moraes. Porto Alegre,RS: Bookman, 2010.

VENNAPUSA, Priya. ***Oracle Database 10g:*** *SQL Tuning Workshop*. Student Guide Volume 1. Oracle University. 2004.

VALIATI, Pedro***. Índices no Oracle – Parte I:*** *Conceito*. Revista SQL Magazine. Edição 36, 2006.

FREEMAN, Robert G. ***Oracle Database 11g*** *: Novos Recursos*. Tradução Arcanjo Miguel. Rio de Janeiro, RJ: Alta Books, 2009.

RANCONI, Vinícius. *O otimizador do Oracle para desenvolvedores*. Disponível em <http://www.linhadecodigo.com.br/Artigo.aspx?id=724> Acessado em: 28 mar. 2011.

DORNELAS Carlos Alberto. ***SQL*** *– Structured Query Language*, 2008. Disponível em <http://www.cadcobol.com/sql_hist.htm>. Acesso em: 22 fev. 2011.