Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

VAGNER FREITAS RIGO

ESTUDO DA METODOLOGIA DE TUNING EM BANCO DE DADOS ORACLE

Assis
2012

Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

VAGNER FREITAS RIGO

ESTUDO DA METODOLOGIA DE TUNING EM BANCO DE DADOS ORACLE

Trabalho de Conclusão de Curso apresentado ao Curso de Ciências da Computação do Instituto Municipal de Ensino Superior de Assis – IMESA e a Fundação Educacional do Município de Assis – FEMA, como requisito do Curso de Graduação.

Orientador: Dr. Alex Sandro Romeo de Souza Poletto.

Área de Concentração: Sistema de Banco de Dados. Sistema Gerenciador de Banco de Dados. Tuning em Banco de Dados.

Assis
2012

FICHA CATALOGRÁFICA

RIGO, Vagner Freitas.

Estudo da Metodologia de Tuning em Banco de Dados Oracle/ Vagner Freitas Rigo. Fundação Educacional do Município de Assis – FEMA - Assis, 2012.

116 p.

Orientador: Dr. Alex Sandro Romeo de Souza Poletto.

Trabalho de Conclusão de Curso – Instituto Municipal de Ensino Superior de Assis - IMESA.

1. Banco de dados. 2. Tuning. 3. Oracle 11g. 4. Otimização.

CDD: 001.6

Biblioteca da FEMA

ESTUDO DA METODOLOGIA DE TUNING EM BANCO DE DADOS ORACLE

VAGNER FREITAS RIGO

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis – IMESA, como requisito do Curso de Graduação, analisado pela seguinte comissão examinadora:

Orientador: Dr. Alex Sandro Romeo de Souza Poletto.

Analisador (1): Fernando Cesar de Lima

Analisador (2):

Assis

2012

Dedico este trabalho a Deus,

Pela força espiritual para a realização desse trabalho.

Aos meus pais Bento Rigo e Edith Freitas Rigo,

Pelo carinho, compreensão e pela grande ajuda.

À minha esposa, meu grande e eterno amor de minha vida Daiane Lima Xarão,

Pelo eterno orgulho de nossa caminhada, pelo apoio, compreensão, ajuda, e, em especial, por todo carinho ao longo deste percurso.

Ao professor Alex Sandro Romeo de Souza Poletto,

Pela orientação deste trabalho.

RESUMO

Este trabalho apresenta um estudo sobre *tuning,* uma importante área de atuação do banco de dados, que trata do ajuste fino de desempenho, conceito este que se encontra em grande desenvolvimento e aplicação atualmente, diante do grande volume de dados que são gerados pelas empresas que utilizam da Tecnologia da Informação. Pelo fato desse grande número de informação e necessidade de constante acesso às mesmas, encontra-se a necessidade de utilização do tuning, como uma forma de aprimoramento destas consultas aos Bancos de Dados, que precisam ser realizadas da maneira mais satisfatória possível, com alto desempenho de resposta e menor tempo. Neste contexto, ajustar e otimizar uma consulta e mesmo o próprio Banco de Dados, é questão primordial. Assim, o presente trabalho, abordando o conteúdo atinente à matéria em questão, trata de descrever os conceitos básicos sobre o Sistema Gerenciador de Banco de Dados (SGBD), tecendo as informações básicas sobre Banco de Dados (BD), sua evolução histórica, conceitos, considerações e SGBD Oracle, e a importância da linguagem SQL. Trata-se ainda da metodologia e técnicas de tuning e otimização e a finalizar apresenta-se proposta de trabalho e estudo de caso com base em um SGBD Oracle 11g, com propostas para instruções DML dentro do Banco de Dados, implementação e testes. Os resultados obtidos apresentam situações de melhora ou degradação no desempenho de várias funcionalidades do software.

Palavras-chave: Banco de dados, Tuning, Oracle 11g, Otimização.

ABSTRACT

This work discusses tuning, an important area of operation of the database, which is fine-tuning performance, a concept that is in great development and application currently before the large volume of data that are generated by companies that use the Information Technology. Because of this large number of information and need for constant access to them, we find the need for use of tuning, as a way to improve these queries to databases that need to be performed most satisfactorily as possible, with high response performance and shorter. In this context, tweak and optimize a query, and even your own database, is key issue. Thus, the present work, addressing the content regards the subject matter, comes to describing the basic concepts of the System Manager Database, weaving the basic information about BD, its historical development, concepts, considerations about the DBMS, and BD Oracle DBMS, and the importance of the SQL language. We will further the methodology and techniques for tuning and optimizing and finalizing work proposal and present a case study based on an Oracle 11g DBMS, with proposals for DML within the database, implementation and testing. The results obtained are situations of improvement or degradation in performance of various software features.

Keywords: Database, Tuning, Oracle 11g, Optimization.

LISTA DE ILUSTRAÇÕES/FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

BD	Banco de Dados
DBA	Administrador de Banco de Dados
DDL	Linguagem de Definição de Dados (Data Definition Language)
DML	Linguagem de Manipulação de Dados (Data Manipulation Language)
I/O	Input/Output
PGA	Program Global Area
SBD	Sistema de Banco de Dados
SGA	System Global Area
SGBD	Sistema Gerenciador de Banco de Dados
SQL	Linguagem Estruturada de Consulta (Structure and Query Language)

Sumário

1 INTRODUÇÃO

Diante do crescente volume de dados atualmente pertencentes às Empresas que empregam Sistemas de Informação, torna-se imprescindível o papel do Banco de Dados. Frente a isso, e à necessidade constante de acesso aos dados, verifica-se em muitas vezes que a disponibilidade dos resultados, é insatisfatória. Dessa forma, vê-se a necessidade de implementação de códigos estruturados seguindo padrões de tuning para otimização dos resultados, dada a sua relevância para as Empresas. Neste sentido, que é proposto este trabalho, cuja finalidade é apresentar a importância da tecnologia (ou dos recursos) de *tuning*, a fim de proporcionar uma melhor utilização dos recursos oferecidos pelos Sistemas Gerenciadores de Banco de Dados (SGBD) no que diz respeito à recuperação de informações por meio de consultas otimizadas.

O crescimento da área é notável, ante a importância ao acesso das informações para as Empresas a fim de que sejam tomadas as decisões. O enorme volume dos dados e o espaço de tempo para a busca destas informações exigem uma atenção especial com a velocidade para o acesso aos bancos de dados. A importância para o desenvolvimento deste trabalho se dá pelo fato de ser o *tuning* um processo de refinamento que envolve modificações em vários aspectos.

1.1 OBJETIVOS

O objetivo do presente trabalho é demonstrar o desempenho e a importância do tuning, diante de sua atual necessidade para as empresas manterem um bom funcionamento no que diz respeito ao acesso de informações aos bancos de dados, sendo o tuning um ajuste fino de desempenho.

A partir disso, pretende-se expor os conceitos gerais de banco de dados e ater-se aos conceitos específicos da tecnologia tuning, bem como realizar propostas de instruções, implementações e testes na referida área, com base no Oracle 11g.

1.2 JUSTIFICATIVAS

A justificativa para propositura deste trabalho está no fato de que o presente tema está muito em comento na atualidade, tratando-se de um assunto pouco utilizado no desenvolvimento dos sistemas, sem muitos conteúdos disponíveis, sendo este, portanto, a fim de esmiuçar o conteúdo e demonstrar que a utilização de tal técnica é capaz de ajustar e aperfeiçoar uma consulta e o próprio Banco de Dados da empresa, podendo-se ter um ganho de performance satisfatória, haja vista a cobrança do mercado a respeito de desempenho e otimização.

1.3 MOTIVAÇÃO

O presente trabalho poderá servir como fonte de consulta, haja vista a escassa exploração sobre este tema pouco abordado até o momento. Ademais, imprescindível o desenvolvimento de tais pesquisas dada a importância para as empresas na modernidade, quando concentram um grande banco de dados, que precisam ser acessados constantemente para seu funcionamento, ademais, pelo fato de aumento das informações a cada ano e constantemente.

Este trabalho poderá contribuir com as empresas que desejarem buscar informações a cerca desta técnica de otimização de consultas aos seus bancos de dados, a fim de obter um melhor desempenho, sugerindo possíveis alterações.

1.4 METODOLOGIA DE PESQUISA

Para o desenvolvimento da presente pesquisa pretende-se coletar o material bibliográfico sobre o tema, a partir de livros, artigos, teses, dissertações, coletas de dados, Internet e todos os demais recursos que abranjam o tema. Utilizando-se de um Banco de Dados em Oracle 11g, será elaborado um Estudo de Caso cuja

finalidade é apresentar testes de performance e otimização de instruções DML (Data Manipulation Language).

1.5 ESTRUTURA DO TRABALHO

O trabalho será dividido em seis capítulos, como descrito abaixo:

No Capítulo 1 retrata-se um breve apontamento sobre o tema objeto da pesquisa, descrição dos objetivos, justificativa, motivação, metodologia de pesquisa e estrutura do trabalho.

No Capítulo 2 são abordados os conceitos básicos sobre o Sistema Gerenciador de Banco de Dados, discorrendo-se inicialmente sobre breves considerações de Banco de Dados, sua evolução histórica e seus vários conceitos apresentados por diferentes doutrinadores. Após, discute-se o SGBD, seus conceitos gerais e características. Em seguida o tema é delimitado tratando especificamente de Banco de Dados Oracle, expondo sucintamente a cerca da evolução histórica da Oracle e suas várias versões desenvolvidas para BD Oracle e a finalizar este capitulo trata-se do SGBD Oracle e suas características gerais.

O Capítulo 3 aborda os conceitos e importância da linguagem SQL, com a demonstração das vantagens e desvantagens de utilização da referida linguagem, suas funções e categorias, enfim, os principais aspectos desta linguagem. Após, o tema objeto da presente pesquisa é aprofundado pela fundamentação teórica sobre tuning e sua alta incidência no mercado atual, apresentando seus conceitos gerais, com fundamento em vários autores, seus objetivos, técnicas, implementação, responsáveis por sua realização, metodologia. E por fim, discorre-se a cerca da otimização, delimitando seu conceito e função, aspectos gerais, plano de execução, otimizador de consultas e Database Oracle 11g – automatic SQL tuning.

No Capítulo 4 é apresentada a Proposta do Trabalho.

No Capítulo 5 é apresentado o Estudo de Caso, propostas para instruções DML dentro do Banco de dados, implementação e testes.

O Capítulo 6 traz as Considerações Finais.

2. CONCEITOS BÁSICOS SOBRE O SISTEMA GERENCIADOR DE BANCO DE DADOS (SGBD)

2.1 BREVES CONSIDERAÇÕES SOBRE BANCO DE DADOS

Antes de se adentrar no assunto, imprescindível percorrer sobre a evolução histórica dos bancos de dados, diante da relevância do assunto em questão, simplesmente pelo fato de que, de alguma forma, com certeza se pertence a um banco de dados ou se é usuário do mesmo.

Atualmente, todos estão rodeados de ambientes informatizados, vivenciando-se a "era da informação". Mas antes desse momento, os programas se desenvolviam a fim de satisfazer as necessidades que eventualmente iam surgindo, o que gerava uma séria inconsistência de dados, dificuldade do acesso aos dados e isolamento dos dados, problemas de segurança, dentre muitos outros, diante do fato de que cada sistema seria desenvolvido especificamente para cada atividade, não havia o cruzamento de informações.

A fim de solucionar estes e outros problemas, viu-se a necessidade da criação dos bancos de dados e dos sistemas gerenciadores de banco de dados. Pelo fato de inicialmente não existir os bancos de dados, mas tão somente os sistemas de arquivos (fitas magnéticas e decks de cartão perfurado) específicos, sem qualquer forma de compartilhamento de dados entre as mesmas.

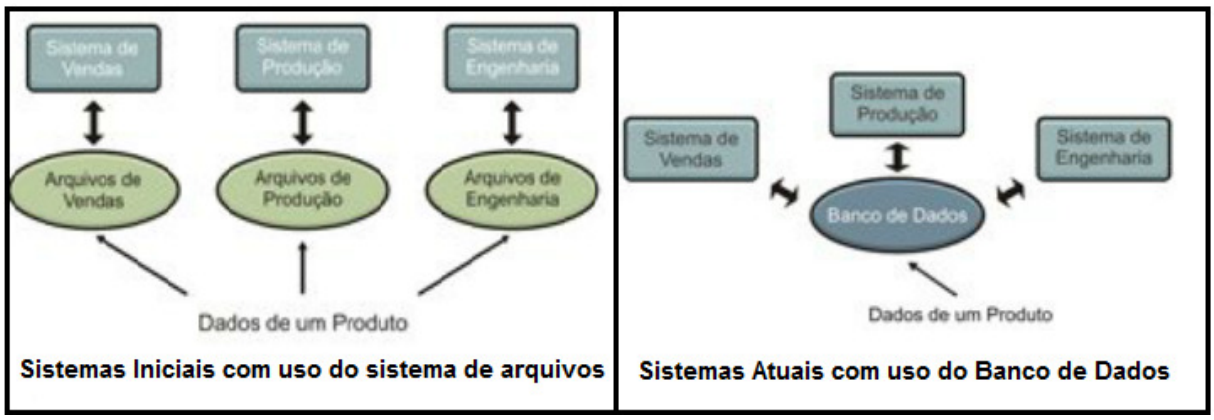


Figura 1. Sistemas antigos X Sistemas atuais

Assim como os sistemas de arquivos, os bancos de dados também evoluíram, passando por várias gerações até atingir o patamar atual, onde cada novo modelo de dados supera uma falha dos modelos anteriores. A fim de ilustrar o exposto, estão às tabelas abaixo, que resumem sucintamente as características de cada uma.

Geração	Época	Modelo	Exemplos	Comentários
Primeira	Década de 1960 e 1970	Sistema de arquivos	VMS/VSAM	Utilizado principalmente em sistemas mainframe da IBM Gerenciamento de registros, sem relacionamentos
Segunda	Década de 1970	Modelo de dados hierárquico e em rede	IMS ADABAS IDS-II	Primeiros sistemas de bancos de dados Acesso navegacional
Terceira	De meados da década de 1970 até o presente	Modelo de dados relacional	DBS Oracle MS SQL Server MySQL	Simplicidade conceitual Modelagem entidade - relacionamento (ER) e suporte a modelagem relacional de dados
Quarta	De meados da década de 1980 até o presente	Orientado a objetos Relacional estendido	Versant FastObjects.Net Objectivity/DB DB/2 UDB Oracle 10g	Suporte a dados complexos Produtos relacionais estendidos com suporte a warehouse de dados e objetos Bancos de dados na web tornam-se comuns
Próxima geração	Do presente ao futuro	XML	dbXML Tamino DB2 UDB Oracle 10g My SQL Server	Organização e gerenciamento de dados não estruturados Modelos relacionais e de objetos adicionam suporte a documentos em XML

Tabela 1. Evolução dos principais modelos de dados (ROB; CORONEL, 2011, p. 37)

Modelo de dados	Ind.* de dados	Ind.* estrutural	Vantagens	Desvantagens
Hierárquico	Sim	Não	1. Promove o compartilhamento de dados; 2. O relacionamento pai/filho promove simplicidade conceitual; 3. A segurança do banco de dados é fornecida e aplicada pelo SGBD; 4. O relacionamento pai/filho promove integridade de dados; 5. É eficiente em relacionamentos 1:M.	1. Implementação complexa exige conhecimento das características de armazenamento físico; 2. O sistema navegacional torna complexo o desenvolvimento, gerenciamento e utilização de aplicações; exige conhecimento do caminho hierárquico; 3. Alterações de estrutura exigem alterações em todos os aplicativos; 4. Há limites de implementação (não são possíveis vários pais nem relacionamentos M:N); 5. Não há linguagem de definição ou manipulação de dados no SGBD; 6. Não há padrões.
Em rede	Sim	Não	1. A simplicidade conceitual é pelo menos igual à do modelo hierárquico; 2. Lida com mais tipos de relacionamentos, como M:N e de vários pais; 3. O acesso aos dados é mais flexível do que nos modelos hierárquicos e de sistema de arquivos; 4. O relacionamento proprietário/membro promove a integridade DE DADOS; 5. Há conformidade a padrões; 6. Inclui linguagem de definição (DDL) e de manipulação (DML) de dados no SGBD.	1. A complexidade do sistema limita a eficiência – ainda é um sistema navegacional; 2. O sistema navegacional resulta em implementação, aplicação, desenvolvimento e gerenciamento complexos; 3. Alterações estruturais exigem alterações em todos os aplicativos.

Modelo de dados	Ind.* de dados	Ind.* estrutural	Vantagens	Desvantagens
Relacional	Sim	Sim	1. A independência estrutural é promovida pela utilização de tabelas independente. Alterações em uma estrutura de tabelas não afetam o acesso a dados ou os aplicativos; 2. A visualização tabular aprimora consideravelmente a simplicidade conceitual, promovendo, assim, projeto, implementação, gerenciamento e utilização mais fáceis; 3. O recurso de consultas *ad hoc* baseia-se em SQL; 4. O SGBD isola o usuário final dos detalhes do nível físico e aprimora a simplicidade de implementação e gerenciamento.	1. O SGBD exige capacidade considerável de hardware e de software do sistema; 2. A simplicidade conceitual permite que pessoas relativamente sem treino utilizem mal as ferramentas de um bom sistema e, se isso não for controlado, pode produzir as mesmas anomalias de dados encontradas em sistemas de arquivos; 3. É possível desenvolver problemas de ilhas de informação, pois indivíduos e departamentos podem desenvolver suas próprias aplicações.
Entidade-relacionamento	Sim	Sim	1. A modelagem visual resulta em excelente simplicidade conceitual; 2. A representação visual o torna um ferramenta de comunicação eficiente; 3. É integrado ao modelo relacional dominante.	1. A representação de restrições é limitada; 2. A representação de relacionamentos é limitada; 3. Não há linguagem de manipulação de dados; 4. Ocorre perda de conteúdo de informação quando atributos são removidos de entidades para evitar exibições muito poluídas.
Orientado a objetos	Sim	Sim	1. Adiciona conteúdo semântico; 2. A representação visual inclui conteúdo semântico; 3. A herança promove a integridade de dados.	1. O desenvolvimento lento de padrões fez com que os fornecedores oferecessem seus próprios aprimoramentos, eliminando, assim, um padrão amplamente aceito; 2. Trata-se de um sistema navegacional complexo; 3. Exige uma ampla aprendizagem; 4. A alta carga dos sistemas deixa as transações lentas.

Tabela 2. Vantagens e desvantagens dos diferentes modelos de BD (Ind.*: Independência)

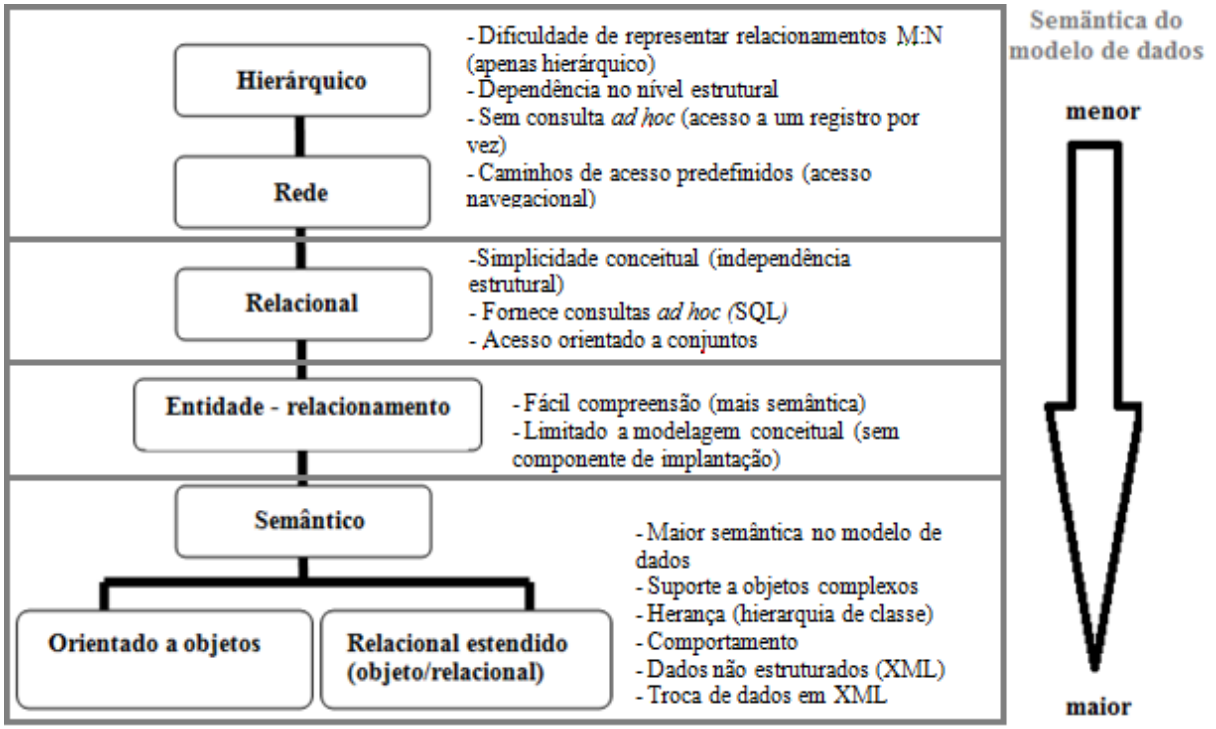


Figura 2. Desenvolvimento dos modelos de dados (ROB; CORONEL, 2011, p. 48)

Discorrido sobre a evolução dos modelos de dados, cabe adentrar no conceito de banco de dados, que é dado por diversos autores, como a seguir demonstra-se.

De acordo com Date (2003):

> Um sistema de banco de dados é basicamente apenas um sistema computadorizado de manutenção de registros. O banco de dados, por si só, pode ser considerado como o equivalente eletrônico de um armário de arquivamento; ou seja, ele é um repositório ou recipiente para uma coleção de arquivos de dados computadorizados (DATE, C. J., 2003, p. 3).

Conforme Teorey, “um banco de dados é uma coleção de dados armazenados e inter-relacionados, que atende às necessidades de vários usuários dentro de uma ou mais organizações, ou seja, coleções inter-relacionadas de muitos tipos diferentes de tabelas” (TEOREY, 2007, p. 2).

Eles são utilizados em aplicações na área da informática, cujo objetivo é registrar e manter informações consideradas significativas para as organizações (ELMSRI, 2000).

Nos ensinamentos de Silberschatz (2006):

> Os sistemas de banco de dados são projetados para gerenciar grandes blocos de informações. Esses grandes blocos de informações não existem isolados. Eles são parte da operação de alguma empresa cujo produto final pode ser informações do banco de dados ou pode ser algum dispositivo ou serviço para o qual o banco de dados desempenha apenas um papel de apoio (SILBERSCHATZ; KORTH; SUDARSHAN, 2006, p. 9).

A acrescentar, adota-se novamente os ensinamentos de Silberschatz, Korth e Sudarshan (2001), onde apresentam o objetivo do sistema de banco de dados:

> O objetivo de um sistema de banco de dados é simplificar e facilitar o acesso aos dados. Visões de alto nível ajudam a alcançar esses objetivos. Os usuários do sistema não devem ser desnecessariamente importunados com detalhes físicos relativos à implementação do sistema. Todavia, um dos fatores mais importantes de satisfação ou insatisfação do usuário com um sistema de banco de dados é justamente seu desempenho. Se o tempo de resposta é demasiado, o valor do sistema diminui. O desempenho de um sistema de banco de dados depende da eficiência das estruturas usadas para a representação dos dados, e do quanto este sistema está apto a operar essas estruturas de dados. Como acontece em outras áreas dos sistemas computacionais, não se trata somente do consumo de espaço e tempo, mas também da eficiência de um tipo de operação sobre outra. (SILBERSCHATZ; KORTH; SUDARSHAN, 2001, p. 14)

Um banco de dados tem como principais objetivos, atender as necessidades da empresa e dos usuários; garantir a segurança dos dados e das transações realizadas; agilizar os processos de consultas, alterações e exclusões de dados; armazenar os dados e garantir que fiquem armazenados permanentemente.

2.2 SISTEMA GERENCIADOR DE BANCO DE DADOS (SGBD)

O sistema de gerenciamento de bancos de dados (SGBD) é um conjunto de programas que gerenciam a estrutura do banco de dados e controlam o acesso aos dados armazenados. Até certo ponto, o banco de dados se assemelha a um arquivo eletrônico com conteúdo muito bem organizado com a ajuda de um software poderoso, conhecido como sistema de gerenciamento de banco de dados (ROB; CORONEL, 2011, p. 6).

O SGBD seria um intermediário entre o usuário e o banco de dados, e é somente por meio dele que é possível o acesso ao conjunto de dados e arquivos em geral, armazenados em sua estrutura.

Em resumo, é o software responsável pela criação e manutenção do banco de dados, ou seja, o SGBD se comunica com o BD e assim possibilita a realização de consultas e manipulações das informações pelos usuários. Um SGBD de boa qualidade deve oferecer garantia de integridade, restrição de acesso não autorizado, recuperação e backup, controle de concorrência, múltiplas interfaces e representação de relações complexas entre os dados.

Dentre as inúmeras vantagens que um SGBD pode trazer, pode-se citar como principais: o aprimoramento do compartilhamento e segurança de dados, melhoria na integração dos dados, minimização da inconsistência dos dados, aprimoramento do acesso aos dados, da tomada de decisão e aumento da produtividade do usuário final.

Dois tipos de linguagens devem ser suportadas pelo SGBD, quais sejam, a DDL (Data Definition Language) e a DML (Data Manipulation Language):

- A DDL permite a especificação do esquema da organização, ou seja, entidades com seus atributos e tipos de dados associados; os relacionamentos entre estas entidades e os índices de acesso associados aos atributos. Portanto, entende-se que se trata de uma organização lógica dos dados de uma realidade, adequados ao modelo de dados do SGBD.

- A DML permite as operações usuais de manipulação de dados, executados pelas aplicações: inclusão, alteração, exclusão e consulta.

Ao tratar do tema, Date o define como software que aborda o acesso ao BD em sua totalidade. A fim de demonstrar, por meio de exemplos, o autor adota ocorrer o seguinte:

> 1. Um usuário faz um pedido de acesso usando uma determinada sublinguagem de dados (geralmente SQL).
> 2. O SGBD intercepta o pedido e o analisa.
> 3. O SGBD, por sua vez, inspeciona o esquema externo (ou as versões objeto desse esquema) para esse usuário, o mapeamento externo/conceitual correspondente, o esquema conceitual, o mapeamento conceitual/interno e a definição do banco de dados armazenado.
> 4. O SGBD executa as operações necessárias sobre o BD armazenado.
>
> Como exemplo, considere as ações relacionadas com a busca de uma determinada ocorrência de registro externo. Em geral, serão necessários campos de várias ocorrências de registros conceituais e, por sua vez, cada ocorrência de registro conceitual exigirá campos de várias ocorrências de registros armazenados. Então conceitualmente, o SGBD deve primeiro buscar todas as ocorrências necessárias de registros armazenados, depois construir as ocorrências de registros conceituais exigidas e, em seguida, construir a ocorrência de registro externo exigida. Em cada estágio, podem ser necessárias conversões de tipos de dados ou outras conversões (Date, 2003, p. 37).

Segundo Silberschatz (2001):

> Um *Sistema Gerenciador de Banco de Dados* (SGBD) é constituído por um conjunto de dados associados a um conjunto de programas para acesso a esses dados. O conjunto de dados, comumente chamado de banco de dados, contém informações sobre uma empresa em particular. O principal objetivo de um SGBD é proporcionar um ambiente tanto *conveniente* quanto *eficiente* para a recuperação e armazenamento de informações do banco de dados. (SILBERSCHATZ; KORTH; SUDARSHAN, 2001, p. 1)

Acrescentando, os mesmos autores:

> Um SGBD é uma coleção de arquivos e programas inter-relacionados que permitem ao usuário o acesso para consultas e alterações destes dados. O maior benefício de um banco de dados é proporcionar ao usuário uma visão abstrata dos dados. Isto é, o sistema acaba por ocultar determinados detalhes sobre a forma de armazenamento e manutenção desses dados. (SILBERSCHATZ; KORTH; SUDARSHAN, 2001, p. 4)

A fim de expor as principais funções e componentes de um SGBD, está a Figura 3:

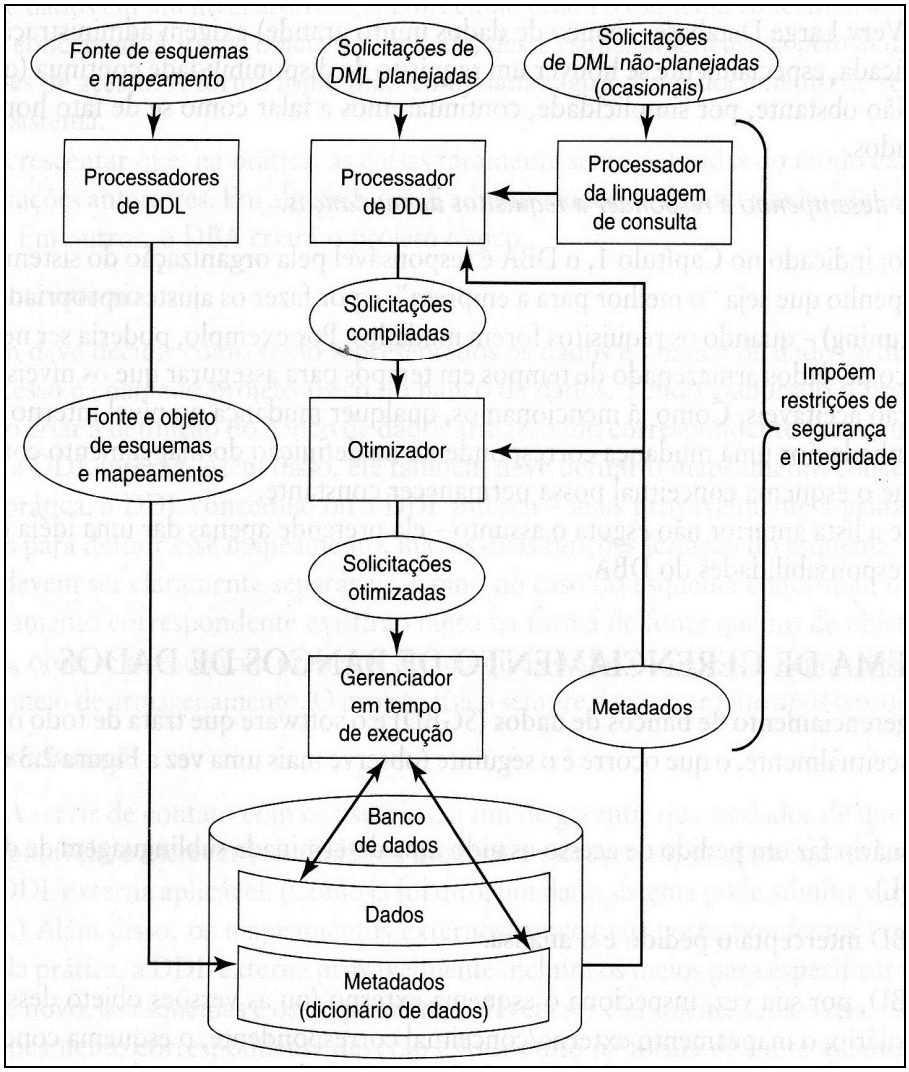


Figura 3. Principais funções e componentes de um SGBD (Date, 2003, p. 38)

A resumir todo o acima explanado importante observar a estrutura de um Sistema Gerenciador de Banco de Dados, que se dá da seguinte maneira apresentada na Figura 4:

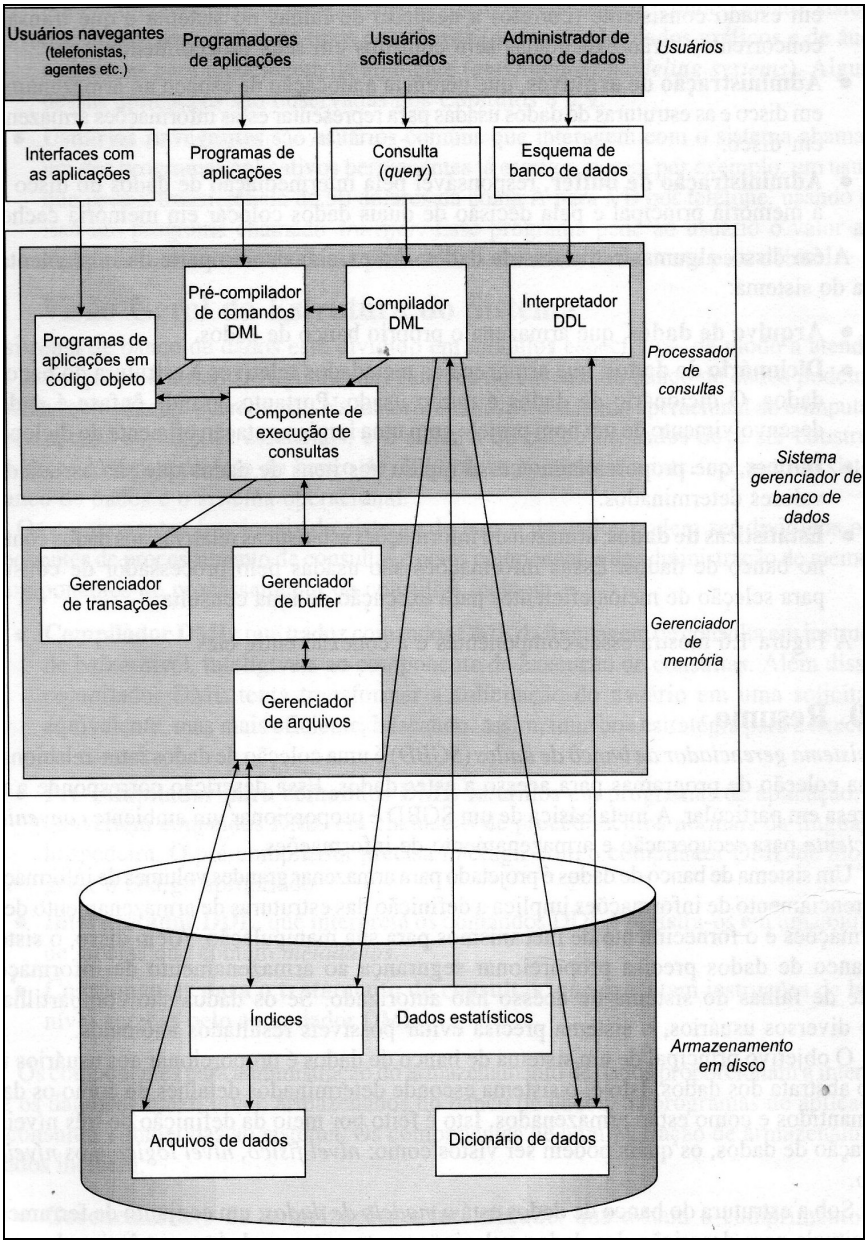


Figura 4. Estrutura do sistema (SILBERSCHATZ; KORTH; SUDARSHAN, ~~2001~~2006, p. 18)

2.3 BANCO DE DADOS ORACLE

2.3.1 Evolução Histórica da Oracle

O Oracle é um sistema gerenciador de banco de dados desenvolvido no final dos anos 70 por Lawrence J. Ellison ao descobrir uma oportunidade não abrangida por outras companhias, por um protótipo funcional de um banco de dados relacional.

A Oracle Corporation foi instituída em 1977 e desde então é líder de mercado. A linguagem criada para o processamento de transações é chamada de PL/SQL, e atualmente está-se na versão 11g.

Com o passar dos anos, a Oracle desenvolve várias versões para o banco de dados, as quais serão demonstradas por meio da tabela abaixo:

Versão	Ano de criação	Características
Oracle 2	1979	• Primeiro Lançamento Público • Funcionalidades básicas SQL (*queries and joins*)
Oracle 3	1981	• Execução automática de instruções SQL • Transações • Consultas em bloqueios
Oracle 4	1984	• Estabilidade
Oracle 5.0	1986	• Cliente-Servidor
Oracle 5.1		• Consultas distribuídas
Oracle 6	Julho 1988	• Bloqueio linha-nível • Backup de dados em linha • PL/SQL no banco de dados
Oracle 6.1		• Servidor Paralelo

Tabela 3. Quadro demonstrativo das versões de Banco de Dados Oracle (ORACLE 2 a ORACLE 6.1)

Versão	Ano de criação	Características
Oracle 7.0	Junho	• Constraints

	1992	• Armazenamento de funções e padrões • Triggers • Visualizar compilação • Manual de funções SQL definidas • Segurança • Limite de recursos – perfis • Auditoria reforçada • Replicação de dados
Oracle 7.1		• Replicação de dados simétricos • Recuperação paralela • SQL dinâmico DBMS_SQL • Consulta e criação de índices, carregamento de dados
Oracle 7.2		• Arquivos de dados redimensionável • Subconsulta na cláusula FROM • Criação de tabelas paralelas
Oracle 7.3		• Índices bitmap • Leitura assíncrona das tabelas • Opção contexto • Introdução Db_verify • Trigger compilação, depuração • Limitações de cláusulas de extensões • Histogramas • Dependenciais • Oracle Trace

Tabela 4. Quadro demonstrativo das versões de Banco de Dados Oracle ORACLE 7.0 a ORACLE 7.3

Versão	Ano de criação	Características
Oracle 8.0	**Junho 1997**	• Banco de dados objeto relacional • Padrão SQL 3 • Recuperação de partições individuais • Security Server introduzidas pela administração central do usuário. Termo de senha, senha de perfis, permitem esquema de senha personalizada. Links de banco de dados privilegiado (sem necessidade de senha a ser armazenada) • START otimizado de consultas • Organização de tabelas por índice • Verificação de restrição de integridade

		• Dados de tinta introduzida na base de dados • Novo formato ROWID • DML paralela • Fusão/partições de equilíbrio • Pool de conexão para suportar mais usuários simultâneos • Muitos melhoramentos de desempenho para o SQL/PSQL/OCI fazendo uso mais eficiente da CPU/Memória. V7 limites alargados
Oracle 8i (8.1.5)		• A recuperação rápida Iniciar – Checkpoint taxa de auto ajustada por atender roll • Reorganização de índices/tabelas de índices • Log Miner introduzidos – Permite online ou redo logs arquivados a ser visualizada através do SQL • Manual de Segurança • Gestão de propriedade de recursos • Estatísticas do otimizador • Procedimentos armazenados Java (Oracle Java VM) • Banco de dados virtual privado • Analisar as tabelas em paralelo • Dados de espera – auto transporte e aplicação de redo logs • Tablespaces transportáveis entre bancos de dados • Drop da coluna na tabela • Índices Funcional – NLS, maiúsculas e minúsculas
Oracle 8i (8.1.6)		• DBA Studio introduzida • XML Parser para Java • Novas funções SQL • Comando ALTER FREELISTS • Checksums sempre na tablespace SYSTEM • Novo PLSQL criptografar/descriptografar • Usuários e esquemas separados • Muitos aprimoramentos de desempenho
Oracle 8i (8.1.7)		• Static Servidor HTTP incluído • Java Server Pages (JSP) motor • OIS – Oracle Integration Server introduzidas • JVM Accelerator para melhorar a performance de código Java • PLSQL Gateway introduziu para implementação de PL/SQL soluções com base na Web • Enterprise Manager Enhancements – incluindo novos relatórios baseados

		em HTML e avançada funcionalidade de replicação incluído • New Character Database Set utilitário ed migração incluído

Tabela 5. Quadro demonstrativo das versões de Banco de Dados Oracle ORACLE 8.0 a ORACLE 8i (8.1.7)

Versão	Ano de criação	Características
Oracle 9i Release 1 (9.0.1)	**Junho 2001**	• Segmentos de reversão tradicional (RBS) • Oracle Parallel Server's (OPS) de escalabilidade • Consulta Flashback (dbms_flashback.enable), este recurso permitirá que os usuários para corrigir as operações sem chamar o DBA • Oracle Ultra Search para pesquisar bancos de dados, sistemas de arquivos • VI (Virtual Interface) suporte ao protocolo, uma alternativa para o TCP/IP disponíveis para o uso com o Oracle Net (SQL*Net) • Mineração de dados • Apoio de cursor de rolagem, permite a busca para trás de um conjunto de resultados • Gerenciamento de memória dinâmica • Reorganização online do índice • Build em XML Developers Kit (XKD), novos tipos de dados para XML (XMLType) • Cost Based Optimizer agora considera também a memória e CPU, Não só custos de acesso do disco • Programas PL/SQL nativa pode ser compilado para binários • Proteção de dados Deep – segurança grão fino e de auditoria • Separação – lista de particionamento de uma lista de valores • Oracle Nameserver
Oracle 9i Release 2 (9.2.0)		• Gerenciado localmente tablespaces SYSTEM • Sistema de arquivos de cluster para Windows e Linux • Compressão do segmento de dados (comprimir as chaves nas tabelas – apenas quando os dados de carga) • Criar bancos de dados standby lógicos com dados da Guarda • Melhorias na segurança – Default Install contas bloqueadas, VPD em sinônimos, AES, Usuários migrar para o diretório

Tabela 6. Quadro demonstrativo das versões de Banco de Dados Oracle ORACLE 9i Release 1 (9.0.1) a ORACLE 9i Release 2 (9.2.0)

Versão	Ano de criação	Características

Oracle 10g Release 1 (10.1.0)		• Grid Computing – uma extensão da funcionalidade de clustering • Melhorias de gerenciamento (self-tuning características) • Melhorias de desempenha e escalabilidade • Automated Storage Management (ASM) • Automatic Workload Repository (AWR) • Automatic Database Diagnostic Monitor (DAMS) • Capacidade de Undrop uma tabela a partir de uma lixeira • Flashback operações disponíveis em linha, transação, tabela ou banco de dados de nível • Capacidade de transporte de tabela em todos os tipos de máquina • Data Pump – mais rápido movimento de dados com expdp e impdp • Capacidade de mudar o nome de tabela • Declaração Nova base de dados 'drop' • Novo agendados de banco de dados – DBMS SCHEDULER • Suporte para espaços de tabela de até 8 exabytes de tamanho
Oracle 10g Release 2 (10.2.0)	Setembro 2005	• As senhas para DB são criptografas • Async • Asmcmd novo utilitário para o gerenciamento de armazenamento ASM
Oracle 10g Express Edition (Oracle XE)	Março 2006	• Liberdade para desenvolver e implementar aplicativos de muitas plataformas • Suporte para uma grande variedade de ambientes de desenvolvimento • Recursos de performance, confiabilidade e segurança • Permite que os desenvolvedores tirem total profeito do Oracle Application Express para rápido desenvolvimento e implementação de aplicativos baseados na Web

Tabela 7. Quadro demonstrativo das versões de Banco de Dados Oracle ORACLE 10g Release 1 (10.1.0) a ORACLE 10g Express Edition (ORACLE XE)

Versão	Ano de criação	Características
Oracle 11g		• Alta capacidade de cluster de banco de dados • Aceleração da automação do data Center e o gerenciamento da carga de trabalho • Grids seguros, altamente disponíveis e escaláveis de servidores e armazenamento de baixo custo • XML binário para melhor desempenho de aplicativos • Oracle Real Application Testing – inovação no teste e gerenciamento de

		alterações em ambientes de TI • Oracle Data Guard – banco de dados em standby com alto desempenho, segurança e recuperação • Gerenciamento do armazenamento orientado por regras de negócios • É possível atingir índices de compactação de 2x a 3x ou até mais para todos os dados • Registro total de todas as alterações nos dados • Oracle Flashback transaction – facilita a reversão de uma transação efetuada com erro, bem como de qualquer transação dependente • Parallel Backup and Restore – ajuda a melhorar o desempenho do backup e restauração de bancos de dados grandes • Oracle Fast Files – armazena grandes objetos • Criptografia aparente • XML mais rápido • Data Recovery Advisor – ajuda os administradores a reduzir significativamente a parada para recuperação, o que permite automatizar investigação de falhas, determinar planos de recuperação e lidar com várias situações de crise • Novo Compilador Java Just-in-time para executar procedimentos Java no banco de dados mais rapidamente • Integração nativa com o Visual Studio 2005 para desenvolvimento de aplicativos .NET no Oracle; ferramentas de migração de Acess com Oracle Application Express; e um recurso para fácil criação de consultas do SQL Developer e rápida codificação de rotinas SQL e PL/SQL • Hot patching' – alta disponibilidade do sistema ao permitir que correções sejam aplicadas sem a necessidade de interromper a operação do SGBD • Pool de conexões e cachês de resultados das consultas: produtos Query Result Caches e Database Resident Connection Pooling • Cubos OLAP incorporados

Tabela 8. Quadro demonstrativo das versões de Banco de Dados Oracle ORACLE 11g

2.3.2 SGBD Oracle

Discorrido sobre os conceitos gerais de banco de dados, sistema gerenciador de banco de dados e evolução histórica da Oracle, cabe agora ponderar sobre as características básicas de um sistema de gerenciamento de banco de dados Oracle.

Trata-se de um SGBD relacional de objeto que propicia uma abordagem aberta, ampla e integrada para o gerenciamento de informações, consistente em um banco de dados Oracle e uma instância do servidor Oracle.

Segundo Lorenzo (2011), o servidor Oracle inclui um sistema de gestão de base de dados relacional (SGBDR) que:

- Armazena grandes volumes de dados em formato texto ou binário;
- Pesquisa dados utilizando técnicas que minimizam o tempo de acesso;
- Gere a segurança de acesso aos dados para inserção, consulta e alteração de acordo com os perfis definidos para cada utilizador;
- Mantém a consistência e integridade dos dados e suas relações, incluindo mecanismos de arquivo de segurança, bloqueio para alteração e verificação de transação;
- Permite armazenamento de dados distribuídos numa rede, mantendo a integridade das relações entre os dados;
- Disponibiliza mecanismos que facilitam o backup/restore dos dados e a sua recuperação em caso de avaria de hardware ou erro humano.

Um BD Oracle possui uma estrutura física e lógica separadas, onde o armazenamento físico de seus dados pode ser facilmente gerenciado sem comprometer o acesso às estruturas lógicas de armazenamento.

A estrutura física de um BD Oracle se determina pelos arquivos do sistema operacional que constituem o BD. Cada banco de dados Oracle é formado por três

tipos de arquivos: arquivos de dados, arquivos de registros redo e arquivos de controle. A retratar a arquitetura do Oracle, pode-se verificar a figura a seguir:

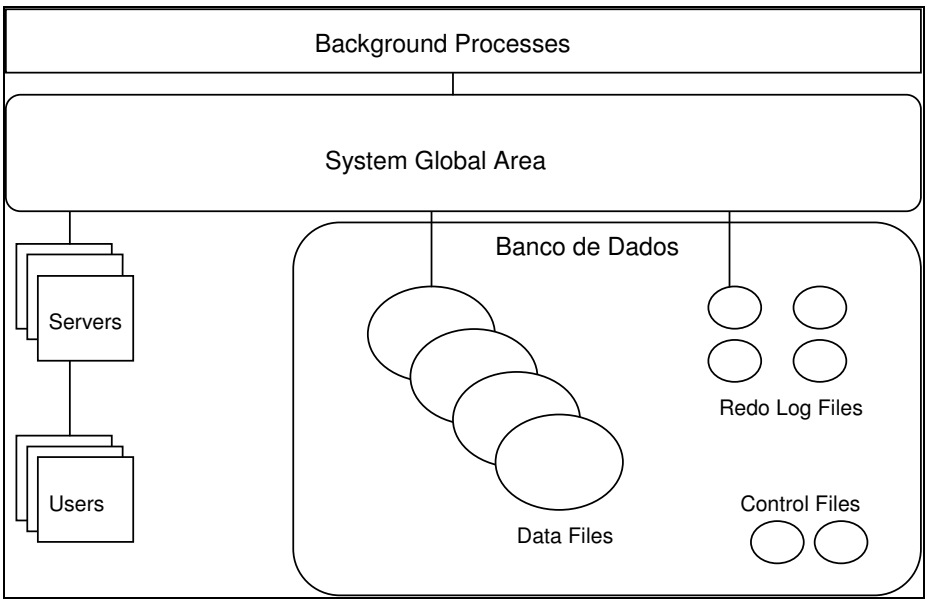


Figura 5. Arquitetura do Oracle

- Os arquivos de dados ou *datafiles* armazenam todos os dados do BD, onde os dados das estruturas lógicas de banco de dados tais como as tabelas e os índices são armazenados fisicamente nos *datafiles* alocados para um banco de dados. Contém todos os dados do banco de dados.

- Os arquivos de registro redo ou *redo log files* são formados por entradas redo, onde cada qual é um grupo de vetores de alteração que descreve uma única alteração atômica no BD. Contém todas as informações relativas às alterações efetuadas no banco de dados para permitir a recuperação.

- Os arquivos de controle ou *control files* possuem entradas que especificam a natureza física do BD, como nome do BD e nomes e localizações dos *datafiles* de um BD e dos arquivos de registro redo. Em resumo, registra a estrutura física do banco de dados.

A estrutura lógica de um BD é determinada por um ou mais *tablespaces* (área lógica de armazenamento) e por objetos de esquemas do banco de dados.

As estruturas de memória são divididas pela SGA e PGA:

- SGA (System Global Area): trata-se de uma região compartilhada de memória reservada pelo SGBD Oracle, que armazena dados e informações de controle. Assim, quando maior for a SGA, o desempenho será melhor. Seus dados serão compartilhados por todos os usuários que acessem ao BD, pois estão divididas por tipo de estrutura de memória, incluindo *database buffers, redo log buffers e shared poll*.

- PGA (Program Global Area): é uma região de memória que armazena dados e informações de controle para um processo servidor, sendo criada sempre que um processo servidor seja iniciado.

Conforme Sumit (2000):

> Toda vez que um banco de dados é iniciado, uma *system global area (SGA)* é alocada e os processos de segundo plano do Oracle são iniciados. A *system global área* é uma área da memória usada para as informações do banco de dados que são compartilhadas pelos usuários. A combinação entre os processos de segundo plano e os *buffers* de memória se chama instância do Oracle (SUMIT, 2000, p. 529).

A cerca dos processos, são divididos em: processos usuário e processos do Oracle. Onde cada usuário conectado possui um Processo Usuário, através do qual se comunica com o Oracle. Entre os processos do Oracle estão os Processos Servidores, que executam as solicitações emitidas pelos Processos Usuários, e os Processos Background.

A fim de resumir o acima explanado, um BD Oracle é uma coleção de dados tratada como unidade capaz de armazenar e recuperar as informações relacionadas. Um banco de dados Oracle pode ser aberto (acessível) ou fechado (não acessível). Em situações normais, o BD é aberto e disponibilizado para o uso, em outras ocasiões é

fechado para as funções administrativas específicas que exigem que os bancos de dados estejam indisponíveis aos usuários.

A finalizar, está a tabela abaixo que demonstra quais os tipos de dados suportados pelo Oracle:

ATRIBUTO	DESCRIÇÃO
BFILE	REFERÊNCIA DE VALOR BINÁRIO DE TAMANHO 4 GIGABYTES
BLOB	VALOR BINÁRIO DE COMPRIMENTO 4 GIGABYTES
CHAR	VALOR ALFANUMÉRICO DE COMPRIMENTO ATÉ 250 DIGITOS
CLOB	VALOR ALFANUMÉRICO DE COMPRIMENTO 4 GIGABYTES
DATE	VALOR DE DATA
FLOAT	VALOR NUMÉRICO REAL
INT	VALOR NUMÉRICO INTEIRO
LONG	VALOR ALFANUMÉRICO DE COMPRIMENTO 2 GIGABYTES
LONG RAW	DADOS BINARIOS DE COMPRIMENTO MAXIMO DE 2 GIGABYTES
NCHAR	VALOR ALFANUMÉRICO QUE OCUPA ESPAÇO DE ARMAZENAMENTO
NCLOB	TIPO OBJETO DE CARACTERES UNICOD DE TAMANHA MAXIMO 4 GB
NUMBER	VALOR NUMÉRICO REAL OU INTEIRO
NVARCHAR2	VALOR ALFANUMERICO DE TAMANHO MAXIMO 4000 BYTES
RAW	DADOS BINARIOS DE COMPRIMENTO MAXIMO 200 BYTES
REAL	VALOR NUMÉRICO REAL
ROWID	TIPO STRING DE BASE 64 QUE REPRESENTA UMA LINHA DA TABELA
TIMESTAMP	TIPO DE DATA QUE INCLUI SEGUNDOS FRACIONADOS
UROWID	TIPO STRING DE BASE 64 QUE REPRESENTA UMA LISTA DA TABELA
VARCHAR	VALOR ALFANUMÉRICO DE COMPRIMENTO ATÉ 2000 DIGITOS
VARCHAR2	VALOR ALFANUMÉRICO DE COMPRIMENTO ATÉ 4000 DIGITOS

Tabela 9. Tipos de dados suportados pelo Oracle

Retratado sobre os principais assuntos atinentes à matéria que abrange o tuning, passa-se então ao seu conteúdo e entendimento da matéria propriamente dita, em nosso próximo capitulo, tratando inicialmente da linguagem SQL e após abordando as principais características sobre tuning.

3. METODOLOGIA DE TUNING

Neste capítulo serão abordados os conceitos sobre a linguagem SQL; tuning, esta passando por sua arquitetura, topologia, vantagens e desvantagens. Também abordar-se-á sobre otimização, suas técnicas e implementação, bem como sobre o BD Oracle 11g.

3.1 LINGUAGEM SQL

Antes de iniciar-se a discussão sobre tuning, imprescindível também discorrer sobre a importância da linguagem SQL, uma vez que é por meio desta o desenvolvimento do tuning.

O SQL é a linguagem de programação que define e manipula o BD, como são relacionais, são armazenados em um conjunto de relações simples, podendo conter uma ou mais tabelas, com colunas e linhas, nos permitindo definir e manipular os dados de uma tabela com os comandos SQL. Apenas uma declaração SQL completa pode ser executada.

O SQL (Structured Quey Language – Linguagem Estruturada de Consulta) é baseado no modelo relacional, mas pode ser adaptada a qualquer ambiente não relacional, foi desenvolvida nos anos 70, inicialmente proposta por E. F. Codd.

Várias são as vantagens e desvantagens para utilização da linguagem SQL, no quadro abaixo apresenta-se algumas delas:

VANTAGENS	DESVANTAGENS
Por ser uma linguagem padronizada pode ser utilizada em quase todos os SGBD relacionais.	Diante da padronização, impossibilita o desenvolvimento de soluções, melhorias ou alterações.

Pode ser utilizada em computador pessoal, mainframe ou computadores de trabalho de médio porte.	Falta de ortogonalidade nas expressões, funções embutidas, variáveis indicadoras, referência a dados correntes, constante NULL, conjuntos vazios, etc.
Possibilita acesso rápido aos dados, redução de custo, diante de sua facilidade de manipulação (simplicidade).	Definição formal da linguagem após sua criação; discordância entre as linguagens hospedeiras; falta de algumas funções; erros, etc.

Tabela 10. Vantagens e desvantagens do SQL

As funções do SQL se dividem em duas categorias, quais sejam: DDL e DML.

- **A DDL**: é uma linguagem de definição de dados, que permite criar, modificar e excluir a estrutura de uma tabela e seus índices, ou seja, o SQL inclui comandos para criação de objetos no BD como tabelas, índices, e visualizações, e comandos para definir direitos de acesso a esses objetos. Alguns comandos desta linguagem são:

    1. CREATE SCHEMA AUTHORIZATION: Cria um esquema de banco de dados.

    2. ALTER TABLE: Modifica uma definição de tabelas (adiciona, modifica ou exclui atributos ou restrições).

    3. DROP INDEX: Exclui um índice de forma permanente.

- **A DML**: é uma linguagem de manipulação de dados, o SQL inclui comandos para inserir, atualizar, excluir e recuperar dados em tabelas de bancos de dados. Alguns comandos desta linguagem são:

    1. INSERT: Insere linhas em uma tabela.

    2. SELECT: Seleciona atributos a partir de linhas de uma ou mais tabelas ou visualizações.

3. WHERE: Restringe a seleção de linhas com base em uma expressão condicional.

4. DELETE: Exclui uma ou mais linhas de uma tabela.

5. ROLLBACK: Restaura os dados a seus valores originais.

Segundo Rob e Coronel (2011), “O coração do SQL é a consulta. [...] no ambiente do SQL, a palavra *consulta* inclui tanto perguntas como ações” (p. 242).

Apesar da extrema importância desta referida linguagem nos dias atuais e de sua imprescindibilidade para o tema objeto da presente pesquisa, ater-se-á a apenas destacar suas principais características.

Portanto, agora já discutido sobre os aspectos principais sobre a linguagem SQL, direciona-se o presente trabalho ao assunto principal, qual seja, o *tuning.*

3.2 FUNDAMENTAÇÃO TEÓRICA SOBRE TUNING

A fim de retratar o tema objeto desta pesquisa, imprescindível destacar as sábias e consistentes palavras de Rob (2011):

> Uma das principais funções de um sistema de banco de dados é fornecer resposta aos usuários finais dentro de um tempo adequado. Os usuários interagem com o SGBD por meio de consultas que geram informações, utilizando a seguinte sequência:
>
> 1. A aplicação do usuário final (extremidade do cliente) gera uma consulta.
> 2. A consulta é enviada ao SGBD (extremidade do servidor).
> 3. O SGBD (extremidade do servidor) executa a consulta.
> 4. O SGBD envia o conjunto de dados resultante para a aplicação do usuário final (extremidade do cliente).
>
> [...] ***o objetivo do desempenho de banco de dados é executar as consultas o mais rápido possível.*** **Portanto, esse desempenho deve ser**

> **estritamente monitorado e passar por sintonizações regulares. A sintonização de desempenho de banco de dados refere-se a um conjunto de atividades e procedimentos projetados para reduzir o tempo de resposta de um sistema de banco de dados** (ROB; CORONEL, 2011, p. 470/471) (grifou-se).

A alta competitividade do mercado atual faz com que as empresas adotem, cada vez mais, sistemas de informação que lhes proporcionem apoio e melhoria a seus processos.

Neste aspecto surge a necessidade de utilização dos SGBD a fim de armazenar e gerenciar informações e proporcionar que estejam disponíveis de maneira eficaz e ágil, em resposta às consultas realizadas pelos usuários.

No entanto, em sua maioria, os SGBDs não são perfeitos, e portanto, necessitam de técnicas de otimização, de aperfeiçoamento de seu funcionamento interno. Assim, tratar-se-á na próxima subseção dos conceitos gerais sobre tuning, para melhor entendimento desta técnica mencionada.

3.2.1 Conceitos gerais sobre Tuning

O termo *tuning* pode ser interpretado como *ajustar, sintonizar,* a fim de melhorar o desempenho. Assim, pode-se definir tuning como capaz de ajustar partes das aplicações do Sistema de Gerenciamento de Banco de Dados, transações do BD ou usabilidade das Redes.

Portanto, *tuning* diz respeito ao ajuste do Sistema de Gerenciamento de Banco de Dados.

Neste sentido, Ikematu (2007) afirma que “tuning” da base de dados, pode ser entendida como uma customização do sistema sob medida para que a performance atenda melhor as suas necessidades.

Como outra definição pode-se adotar as palavras de Sousa (2011):

> O termo “tuning”, que traduzido friamente significa “sintonia”, encaixou-se no cenário de tecnologia a partir do momento em que somaram-se: o uso das informações armazenadas para tomadas de decisões gerenciais estratégicas, mais o aumento no volume de dados consideravelmente, mais a preocupação com a velocidade do tempo de resposta. A soma desses três requisitos fizeram com que se intensificassem os estudos sobre aumento de desempenho e surgisse então a valorização da sintonia e do refino do banco de dados. (SOUZA, 2011, p. 30)

O tuning é utilizado e recomendado para reduzir custos, trazer agilidade, aumento de produtividade e segurança, garantindo a integridade do SGBD. É muito importante para o sucesso de todo o processo.

De acordo com Hiratsuka (2012):

> O termo performance, ou desempenho, como preferir, é uma palavra-chave quando se fala em aplicações de banco de dados, pois não só se quer obter resultados confiáveis, como é necessário que os mesmos sejam oferecidos em tempo hábil. Com o desenvolvimento das novas tecnologias de computador, os sistemas se tornaram mais complexos. Mais dados precisam ser gerenciados, consultas mais complexas precisam ser processadas, características para a utilização de redes de computadores devem ser incorporadas às aplicações, fazendo com que os problemas de performance tornem-se mais frequentes (HIRATSUKA, 2012, p. 01).

Conforme retratado por Souza (2009) *apud* Peixoto Júnior e Carvalho (2008), profissionais com experiência em BD constataram que 45% (quarenta e cinco por cento) das ações de tuning são destinadas a configuração do Sistema Operacional, 35% destinadas à configuração do SGBD e 20% destinadas ao refinamento das consultas, conforme figura abaixo:

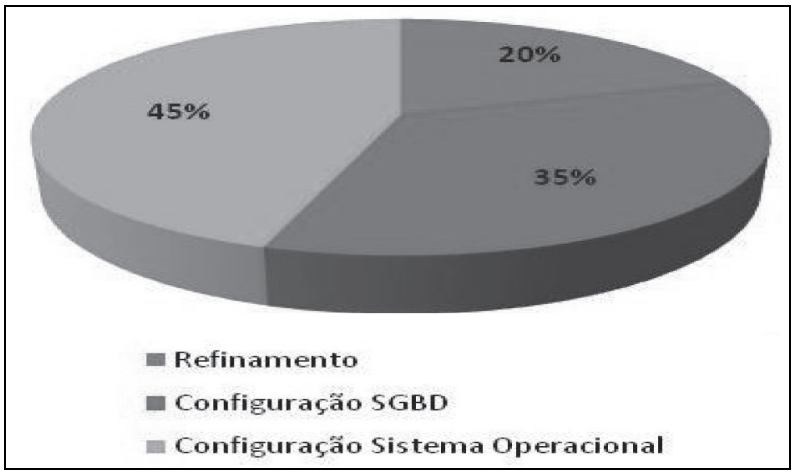


Figura 6. Fases do Tuning (Souza (2009), et al apud Peixoto Júnior e Carvalho(2008))

Adota-se as palavras de Castro (2011) a fim de demonstrar os principais objetivos do tuning, quais sejam:

> - estruturas SQL acessem o menor número possível de blocos no database
> - os blocos de dados (disponíveis /exigidos) para futuro uso são carregados na memória quando requisitados pelos usuários e que os blocos lidos pelo full table scan são removidos da memória para ajuda a liberar espaço na memória
> - estruturas de memória sejam grandes o suficiente para dados cache na memória por um período mais longo de tempo
> - usuários compartilhem códigos SQL nas áreas de memória, a fim de evitar o REPARSING das estruturas
> - processos de leitura e escrita serão realizados de forma mais rápida quanto possível
> - minimizar a concorrência entre os recursos (CASTRO, 2011, p. 01).

Cabe destacar que para se verificar a eficiência do tuning, importante que a rede seja confiável e rápida, caso contrário não será obtido o resultado esperado, outro ponto que também deve ser levado em consideração é fluxo de dados que circunda a rede, pois quanto menor, melhor será o desempenho.

O tuning envolve em sua realização os administradores de banco de dados, administradores de sistemas e desenvolvedores de aplicação, que se procede da maneira exposta na tabela abaixo:

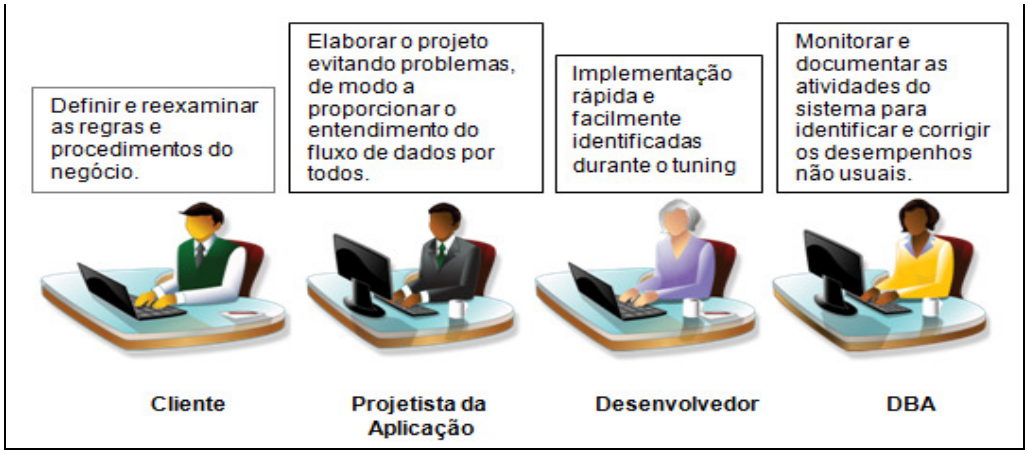


Figura 7. Responsáveis pelo tuning

Geralmente o tuning é iniciado durante a fase de aceitação do sistema, pelo fato de os problemas de desempenho estar mais visíveis, mas deve ser realizado em todas as fases (projeto, desenvolvimento e produção).

Portanto, para a realização do tuning é necessário a realização de cinco passos primordiais para a execução no servidor, como:

- Checagem da arquitetura e designo do modelo de dados do servidor;
- Checagem das aplicações do BD, definindo design eficiente para a linguagem SQL;
- Checagem das estruturas de memória e tuning do SQL e user memory;
- Tuning I/O, com monitoramento de distribuição do BD, tipos de segmentos e tablespaces nas operações I/O;
- Tuning OVERALL MEMORY, para garantia das áreas de memória que são utilizadas de maneira eficiente.

Neste mesmo sentido, Ikematu (2007), que frisa a relevância do tuning, sendo necessária a preparação de uma estratégia de performance quando da elaboração do projeto, pois no momento em que os usuários começarem a perceber os

problemas quanto ao desempenho da aplicação, já será tarde para empregar algumas das técnicas do tuning e enquanto outras técnicas somente poderão ser implementadas se o referido projeto tenha sido bem elaborado, mesmo a performance de um sistema bem projetado pode degradar com o uso, daí a sua importância de realização do tuning em todas as fases do sistema e para sua manutenção.

Adotando os pensamentos de Carneiro *et al (*2006*)*:

> Após um banco de dados ter sido desenvolvido e estar em operação, o uso real das aplicações, das transações, das consultas e das visões revela fatores e áreas de problemas que podem não ter sido considerados durante o projeto físico inicial. As informações de entrada para o projeto físico podem ser revisadas por meio da coleta de estatísticas reais sobre os padrões de uso. A utilização dos recursos, bem como o processamento interno do SGBD pode ser monitorado para revelar gargalos, tais como a disputa pelos mesmos dados ou dispositivos. Os volumes de atividades e os tamanhos dos dados podem ser bem mais estimados. (CARNEIRO, *et al*, 2006, p. 02)

Conforme se pode verificar no decorrer do presente trabalho, refere-se sempre a um banco de dados Oracle, portanto, trata-se do tuning com bases de dados em ORACLE.

Novamente, toma-se por base os notáveis ensinamentos de Hiratsuka (2012), o qual define que para o alcance de bons resultados na performance tuning, necessário seguir alguns procedimentos, os quais se dão pela forma apresentada, respectivamente:

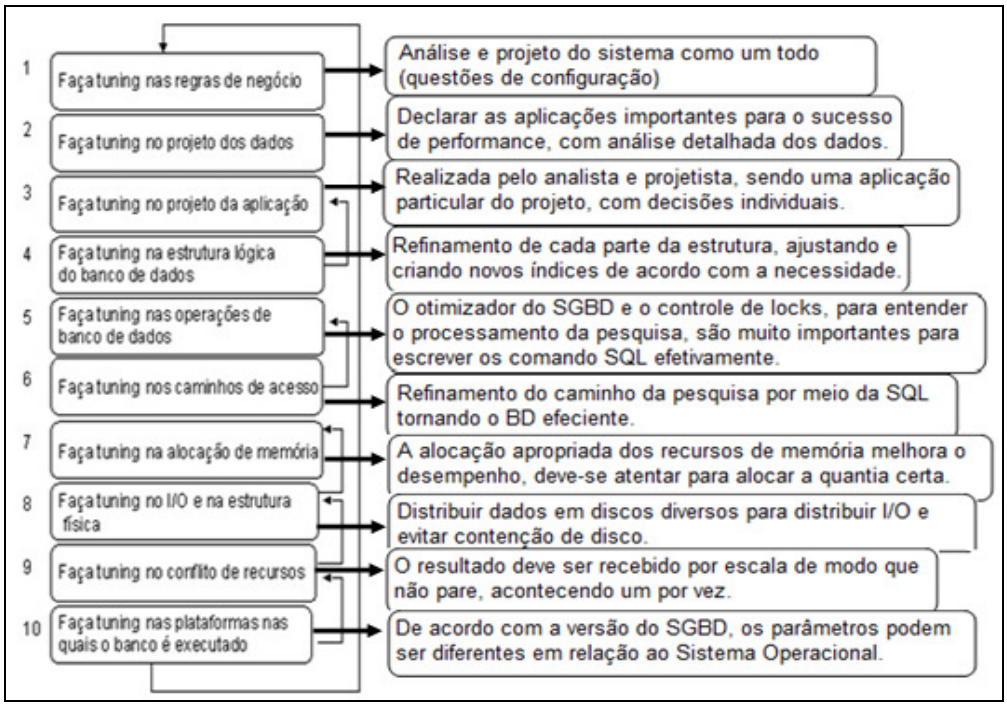


Figura 8. Metodologia para tuning

(baseado no artigo científico de Hiratsuka, disponível em: http://www.batebyte.pr.gov.br/modules/conteudo/conteudo.php?conteudo=1681)

3.2.1.1 Sintonia de índices

A opção inicial de índices pode necessitar de uma revisão, pelos seguintes fatos: algumas consultas podem demorar maior tempo para serem executadas diante da ausência de um índice, alguns índices podem, não ser usados e certos índices podem ocasionar sobrecarga exagerada por serem baseadas em um atributo que invariavelmente sofre alterações.

Segundo Rob e Coronel (2011), “os índices são fundamentais para acelerar o acesso aos dados, pois facilitam a busca, classificação e utilização de funções agregadas e, até mesmo, de operações de junção.” (p. 480)

Acrescentam ainda que, “a melhoria da velocidade de acesso aos dados se deve ao fato de que o índice ser um conjunto ordenado de valores que contém a chave de índice e ponteiros.” (ROB, Peter; CORONEL, Carlos, 2011, p. 480)

Muitos dos SGBDs possuem ou um comando ou um meio de rastreamento que pode ser utilizado para requisitar que o sistema revele como a consulta foi executada, as operações realizadas, e a ordem e estruturas de acesso. Diante da análise destes planos de execução é possível diagnosticar os motivos dos problemas citados. Certos índices podem ser excluídos e outros novos podem ser criados diante da análise de sintonia.

A finalidade da sintonização é a avaliação dinâmica dos requisitos, que podem variar durante os diferentes períodos de mês e semana, podendo reorganizá-los a fim de proporcionar a melhor performance possível. As exclusões e criações dos novos índices são justificadas em função desta melhoria de desempenho, e a atualização de uma tabela normalmente é suspensa enquanto o índice for excluído ou criado.

A importância do índice se dá pelo fato de que a sua varredura é mais eficiente que a varredura completa de uma tabela, uma vez que os dados apresentados no índice encontram-se preordenados e sua quantidade, na maioria das vezes, é relativamente menor. Assim, ao se executar buscas é quase sempre melhor que o SGBD utilize-se dos índices para acesso a tabelas e não pela varredura completa.

3.2.1.2 Sintonia de consultas

O monitoramento e a sintonia de consultas SQL é uma tarefa que retira grande tempo dos DBAs, haja vista a complexidade e por ser a maior parte dos acessos realizados no Sistema Gerenciador de Banco de Dados, no entanto, como já tratado neste trabalho, muitas as vezes, os sistemas não apresentam o desempenho esperado, seja pela falta de experiência no desenvolvimento, seja pelo nível baixo de conhecimento técnico, seja pelos prazos de entrega subdimensionados, seja pela ausência de monitoramento individual que, por consequência, tornam as consultas ineficientes.

O otimizador irá determinar um plano de execução partindo de uma consulta SQL, com margem limitada de opções sobre a utilização de operadores. Se a consulta SQL for mal elaborada, o otimizador irá se valer de um caminho nem sempre

adequado e irá desencadear um plano de execução que poderá vir a comprometer o desempenho.

Em circunstâncias onde são realizadas muitas consultas mal escritas, frequentemente pode gerar consequências drásticas, atrapalhando a utilização eficiente dos recursos disponíveis e deixando de atender, em momento oportuno os processos urgentes de execuções rápidas e prioritárias.

Neste contexto, Carneiro *et al* (2006) faz-se valer das sábias palavras de Navathe (2006), onde expõe:

> Desta forma existem algumas técnicas utilizadas pelos administradores de banco de dados a fim de alcançar desempenho. A justificativa para se usar a técnica de reescrita de consultas como uma das primeiras a ser utilizada, é que essa técnica afeta apenas uma consulta específica, não se propagando para outras aplicações que acessam as tabelas envolvidas na instrução SQL (CARNEIRO, *et al*, 2006, p. 04).

Complementando, Carneiro (2006) cita alguns critérios que podem ser utilizados quando da identificação de consultas que necessitem ser modificadas como:

- Monitorar as sessões ativas que estão sendo executadas no banco de dados;
- Separar as consultas que estão com execuções demoradas;
- Dividi-las em grupos, como: prioridade, frequência de execução e fraco desempenho;
- Implementar os ajustes reescrevendo as consultas que estão com fraco desempenho.

Cada banco de dados fornece suas ferramentas específicas para capturar as consultas citadas nos itens acima, porém a maneira mais eficiente e explícita é o próprio uso feito pelo usuário final. (CARNEIRO, 2006, p. 04).

3.3 OTIMIZAÇÃO

3.3.1 Conceito e função

Date (2003) apresenta conceituação à otimização da seguinte maneira:

> A otimização representa ao mesmo tempo um desafio e uma oportunidade para sistemas relacionais: um desafio, por que a otimização é uma exigência, caso o sistema espera atingir um desempenho aceitável; uma oportunidade, pois é precisamente um dos pontos fortes da abordagem relacional o fato de que as expressões relacionais estão em um nível semântico suficientemente alto para a que a otimização seja viável em primeiro lugar. Ao contrário, em um sistema não relacional, em que requisições dos usuários são expressas em um nível semântico mais baixo, qualquer "otimização" deve ser feita manualmente pelo usuário. E, se, o usuário tomar uma decisão equivocada não haverá nada de que o sistema possa fazer para melhorar a situação (DATE, 2003, p. 456)

Acrescenta ainda, o mesmo autor:

> [...] O otimizador é um programa; assim, por definição, é muito mais paciente que um usuário humano típico. O otimizador é capaz de considerar literalmente centenas de estratégias de implementação diferentes para requisição, enquanto é extremamente improvável que um usuário sequer levasse em consideração mais de três ou quatro (pelo menos com alguma profundidade). Por fim, o otimizador pode ser considerado, em certo sentido, como a incorporação dos conhecimentos e serviços "dos melhores" programadores. [...] (DATE, 2003, p. 457).

A otimização tem como principal objetivo permitir que as aplicações passem por tempos de respostas mais curtos com redução do uso de recursos do BD, e

consequentemente, outras aplicações do BD também serão influenciadas com o ganho de performance.

3.3.2 Aspectos gerais

De modo simplificado, a otimização pode ser entendida como eliminação de índice ineficiente, pela implementação de novos filtros ou alterações de parâmetros.

De acordo com Silberschatz *et al* (2001):

> O custo do processamento de uma consulta é determinado pelo acesso ao disco, que é lento se comparado ao acesso à memória. Normalmente, há muitas estratégias possíveis para processar uma determinada consulta, especialmente se a consulta for complexa. A diferença entre uma estratégia boa e uma estratégia ruim, em termos do número de acessos de discos exigidos, é frequentemente significativa, e pode ser de grande magnitude. Consequentemente, vale a pena para o sistema ganhar uma quantia significativa de tempo na seleção de uma estratégia boa para processar uma consulta, até mesmo se a consulta for executada somente uma vez (SILBERSCHATZ; KORTH; SUDARSHAN, 2001, p. 381).

Obviamente o sistema terá melhor funcionamento com seus recursos otimizados, com componentes de sistema suficientes para tanto.

Neste sentido, Rob *et al* (2011) apresenta tabela que demonstra as diretrizes gerais para melhor desempenho do sistema, com requisitos básicos para hardware e software, quais sejam:

	RECURSOS DO SISTEMA	**CLIENTE**	**SERVIDOR**
Hardware	CPU	O mais rápido possível CPU ***dual core*** ou superior	O mais rápido possível Vários processadores (tecnologia ***quad-core***)
	RAM	O máximo possível	O máximo possível

	Disco Rígido	Disco rígido SATA/EIDE rápido, como espaço livre suficiente.	Vários discos rígidos de alta velocidade e capacidade (SCSI / SATA / Firewire / Fibre Channel) em configuração RAID
	Rede	Conexão de alta velocidade	Conexão de alta velocidade
Software	Sistema Operacional	Sintonização fina para melhor desempenho de aplicações de cliente	Sintonização fina para melhor desempenho de aplicações de servidor
	Rede	Sintonização fina para melhor taxa de transmissão	Sintonização fina para melhor taxa de transmissão
	Aplicação	Otimizar SQL em aplicações de cliente	Otimizar servidor de SGBD para o melhor desempenho

Tabela 11. Diretrizes Gerais para melhor desempenho do sistema (ROB; CORONEL, 2011, p. 471)

Mas da mesma forma, deve-se entender que, normalmente estando diante de recursos limitados, com restrições tanto internas como externas, portanto, necessário trabalhar com o que se tem, otimizando os recursos existentes para o melhor desempenho possível, daí a motivação para a realização da otimização e do *tuning.*

Neste sentido estão os ensinamentos de Silberchatz *et al* (2006), quando o mesmo resume sucintamente os procedimentos a serem realizados para a execução de uma consulta e a ocorrência da otimização:

> A primeira ação que o sistema tem de executar em uma consulta é traduzir a consulta para sua forma interna, que (para sistemas de bancos de dados relacionais) normalmente é baseada na álgebra relacional. No processo de geração da forma interna da consulta, o analisador sintático verifica a sintaxe da consulta do usuário, verifica se os nomes das relações que aparecem na consulta são nomes de relações no banco de dados e assim por diante. Se a consulta foi expressa em termos de uma visão, o analisador sintático substitui todas as referências ao nome pela expressão da álgebra relacional para obter a visão. Dada uma consulta, geralmente há uma variedade de métodos para chegar à resposta. É responsabilidade de o

> sistema transformar a consulta, de acordo com o que foi fornecido pelo usuário, em uma consulta equivalente que pode ser calcula mais eficientemente. Essa *otimização* ou, mais precisamente, essa melhoria da estratégia de processamento de uma consulta é chamada de *otimização de consulta* (SILBERSCHATZ; KORTH; SUDARSHAN, 2001, p. 433/434).

Conforme já mencionado no início do capítulo, as atividades de sintonização serão desempenhadas por várias pessoas, mas em geral, pode-se dividí-las em cliente e servidor, da seguinte maneira:

- Para o cliente, o objetivo é criar uma consulta de SQL que retorne a resposta correta no menor espaço de tempo possível, com a quantidade mínima de recursos do servidor. Essa atividade necessária para alcance da meta descrita é nomeada como *sintonização de desempenho de SQL.*

- Para o servidor, o ambiente de DGBD deve ser configurado adequadamente para atender as solicitações/consultas dos clientes no menor período de tempo, por meio dos recursos existentes. As atividades necessárias para atingir essa fase são designadas como *sintonização de desempenho de SGBD.*

Para se ter uma visão geral a cerca do processamento de consultas, cabe destacar seus quatro estágios da otimização, que se dão na respectiva ordem:

- Conversão da consulta em alguma forma interna (geralmente, uma árvore de consulta ou uma árvore de sintaxe abstrata ou simplesmente como forma interna da álgebra relacional ou do cálculo relacional);

- Coversão para a forma canônica, usando diversas leis de transformação;

- Escolha de procedimentos de baixo nível candidatos para implementação de diversas operações na representação canônica da consulta;

- E por fim, Criar planos de consulta e definir o mais econômico, através da utilização de fórmulas de custo e ciência das estatísticas do BD.

A demonstrar o conteúdo acima exposto, cabe ilustrar com a figura abaixo, conforme preordenado por Date (2003):

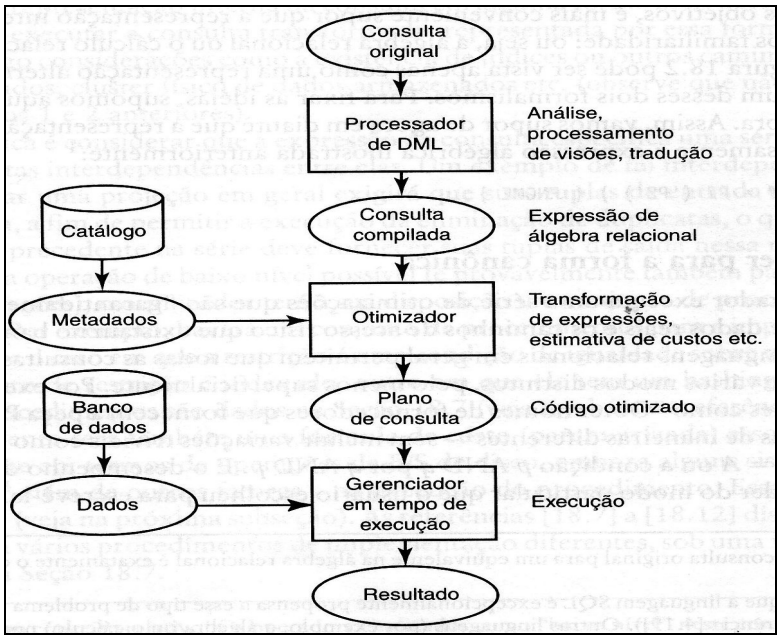


Figura 9. Visão Geral do processamento de consultas (DATE, 2003, p. 459).

Segundo Souza (2007):

> Após o otimizador cumprir a sua função de buscar o melhor desempenho para execução da expressão proposta, é enviado então ao "gerador de registros", que irá executar o plano de execução, e por fim, passá-lo para o executor de SQL, que por sua vez irá apresentar o resultado final.

A fim de entender passo a passo tais procedimentos abre-se ainda tópico específico a cerca do plano de execução e otimizador de consultas, em seguida, o tema será delimitado ao otimizador do Oracle, na versão 11g.

3.3.3 Plano de execução

Neste momento tratar-se-á de demonstrar o que é o plano de execução e para que serve, nos moldes de um SGBD Oracle.

O plano de execução de uma instrução SQL é uma sequência de operações que o Oracle realiza para executar uma instrução (PRADO, 2010, p. 03).

É exibido na forma de árvore de linhas, com as informações de ordenação de tabelas descritas na instrução, qual o método de acesso às tabelas, método join para as tabelas comprometidas pelas operações, operações de dados, otimização, particionamento e execução paralela, entre outras.

No entanto, frisa Prado (2010), que:

> O Plano de execução, por si só, não pode diferenciar instruções bem tunadas daquelas que não apresentam boa performance. O acesso a dados por meio de índices normalmente é mais rápido que acesso *full table scan*, em ambientes OLTP, porém o fato do plano de execução utilizar um índice em algum passo da execução não necessariamente significa que a instrução será executada eficientemente. Em alguns casos, índices podem ser extremamente ineficientes. Quando uma consulta irá retornar mais que 20% dos dados de uma ou mais tabelas ou quando uma consulta irá acessar tabelas pequenas (com poucas linhas), geralmente é mais rápido o acesso *full table scan* do que o acesso através de índices (index lookup). (PRADO, 2010, p. 04)

Nestes termos, embora o otimizador geralmente seja eficiente, em certos casos pode não realizar a melhor escolha do plano de execução, principalmente pelo fato de que o mesmo toma decisões a partir das estatísticas existentes, e se estas forem antigas, desatualizadas, ou mesmo atualizadas, a escolha do otimizador pode não ser a mais eficiente e então o usuário poderá alterar o modo do otimizador.

3.3.4 Otimizador de Query

Segundo Prado (2010), o otimizador de consultas Oracle estabelece, de forma dinâmica qual o plano de execução SQL mais eficiente para o caso especifico, considerando os caminhos acessíveis e as informações fundadas em estatísticas de objetos de tabelas ou índices acessados.

Muitas vezes, o otimizador pode se valer do uso de hints, que servem como instruções ao otimizador, como comentários, que podem ou não serem adotados, fornecendo determinado caminho de acesso, índice, etc.

Este será o objetivo do otimizador, na escolha do caminho mais pertinente para execução da instrução, com a utilização da quantidade menor de recursos possível e em menor tempo de resposta.

Neste sentido Rob e Coronel (2011), ao expor:

> Durante a otimização de consultas, o SGBD deve escolher quais índices utilizar, como executar operações de junção, qual tabela utilizar primeiro, etc. cada SGBD possui seus próprios algoritmos para a determinação do modo mais eficiente de acessar os dados. As duas abordagens mais comuns são a otimização com base em regras e a com base em custos.
>
> - O otimizador com base em regras utiliza regras e pontos preestabelecidos para determinar a melhor abordagem de execução de uma consulta. As regras atribuem um "custo fixo" a cada operação de SQL. Os custos são, em seguida, somados para produzir o custo do plano de execução.
>
> - O otimizador com base em custos utiliza algoritmos sofisticados com base em estatísticas sobre os objetos a serem acessados para determinar a melhor abordagem de execução de uma consulta. Nesse caso, o processo do otimizador soma os custos de processamento de E/S e de recursos (RAM e outros espaços temporários) para obter o custo total de determinado plano de execução. (ROB & CORONEL, 2011, p. 493)

De acordo com Prado (2010, p. 05), escolhido o melhor plano de execução o otimizador baseia-se basicamente em três tipos de medidas: seletividade, cardinalidade e custo.

- Seletividade: fração de linhas de um conjunto de linhas tal como uma tabela, visão ou resultado de uma ligação (join) ou operador GROUP BY.

- Cardinalidade: número de linhas em um conjunto de linhas, tal como uma tabela, visão ou resultado de uma ligação (join) ou operador GROUP BY.

- Custo: unidade de trabalho ou recursos utilizados para execução da instrução SQL como plano particular, com envolvimento de I/O, CPU e memória.

Decorrida esta fase, determinando-se o plano de execução, o otimizador irá criar um conjunto de planos para a instrução SQL conforme os caminhos de acesso e hints, estimando-se o custo de cada plano de acordo com as estatísticas do dicionário de dados para a distribuição de dados e características de armazenamento das tabelas, índices e partições acessadas pela instrução. Ao final, depois de comparar os planos realizará a escolha do plano de menor custo.

4. PROPOSTA DE TRABALHO

Segundo Andrade (2005):

> Antes de encontrar nas consultas SQL o responsável por todos os problemas de desempenho, é vital saber se realmente *há* problemas de desempenho. Se realmente existe, é preciso identificar corretamente a causa do problema. Uma vez verificado que realmente há problemas nas consultas à base de dados é preciso saná-los. E realmente consultas SQL são responsáveis por boa parte dos problemas de desempenho das aplicações que utilizam banco de dados. (ANDRADE, 2005, p. 12)

Neste contexto, apresenta-se a seguinte proposta de trabalho:

- Propostas de tunnig em escritas de consultas SQL;
- As ferramentas para tuning utilizadas no banco de dados Oracle:
    - Explain Plan;
    - SQL Loader;
    - TKPROF;
- Regras de Otimização versus custo da otimização;
- Tuning em índices e métodos básicos de acessos;
- Influência do otimizador de códigos;
- Técnicas de Tuning em views materializadas e tabelas temporárias;
- Tuning com base em técnicas alternativas de armazenamento de dados;
- O tuning ligado as consultas DML;
- O tuning em Joins e Agrupamentos de Consultas;

5. ESTUDO DE CASO

Uma vez discorrido sobre os conceitos gerais de BD, SGBD, bem como sobre o Oracle, especificamente sobre a versão 11g, e *tuning,* no presente capítulo cabe tão somente desenvolver as propostas para instruções DML no BD mencionado, e realizar as implementações e testes cabíveis.

Iniciando-se a implementação do estudo de caso, parte-se dos downloads necessários quanto aos aplicativos para o desenvolvimento das técnicas de tuning, com as descrições de cada um dos utilizados, bem como a localização de sua disponibilidade, quais sejam:

- Virtual BOX (Versão 4.1.20 r80170). Disponível em: http://www.oracle.com/technetwork/server-storage/virtualbox/downloads/index.html.
- Banco de dados ORACLE DATABASE EXPRESS edition 11g Release 2. Disponível em: http://www.oracle.com/technetwork/products/expressedition/downloads/index.html?ssSourceSiteId=ocomen.
- Sql-Developer. Disponível em: http://www.oracle.com/technetwork/developer-tools/sql-developer/downloads/index.html.
- Appliance. Disponível em: http://www.oracle.com/technetwork/database/enterprise-edition/databaseappdev-vm-161299.html.

5.1 Criação do modelo físico (modelagem e normalização):

Parte-se da criação dos tablespaces da aplicação do Banco de Dados.

Para ganho de tunning na aplicação, necessária a criação de tablespaces diferentes para dados e índices, como recomendação própria da Oracle, tem-se que destacar que nesta criação, não obteve-se resultados satisfatórios, mas tão somente diferenças de respostas de processamento ao criar a tablespaces de índices com tamanho de blocos de 32 kbytes (o padrão do banco de dados instalado é de blocos de 8 kbytes, tamanho definido de blocos do tablespaces de dados).

Assim, segue-se os scripts utilizados para criação das tablespaces conforme já exposto:

```
CREATE TABLESPACE ECOMMERCE_D
DATAFILE '/home/oracle/app/oracle/oradata/orcl/ecommerce_d.dbf' SIZE 1048576
AUTOEXTEND ON NEXT 1048576 MAXSIZE
UNLIMITED LOGGING ONLINE EXTENT MANAGEMENT LOCAL AUTOALLOCATE SEGMENT SPACE MANAGEMENT AUTO;
```

Figura 10. Script para criação da tablespaces de dados

```
CREATE TABLESPACE ECOMMERCE_I
DATAFILE 'home/ racle/app/ racle/oradata/orcl/ecommerce_i.idx' SIZE 1048576
AUTOEXTEND ON NEXT 1048576 MAXSIZE UNLIMITED NOLOGGING ONLINE PERMANENT BLOCKSIZE 32768
EXTENT MANAGEMENT LOCAL AUTOALLOCATE SEGMENT SPACE MANAGEMENT AUTO;
```

Figura 11. Script para criação da tablespaces de índices

```
CREATE USER ECOMMERCE IDENTIFIED BY ECOMMERCE DEFAULT TABLESPACE ECOMMERCE_D ACCOUNT UNLOCK ;
GRANT CONNECT, RESOURCE TO ECOMMERCE;
ALTER USER ECOMMERCE DEFAULT ROLE CONNECT, RESOURCE;
ALTER USER ECOMMERCE QUOTA UNLIMITED ON ECOMMERCE_D;
ALTER USER ECOMMERCE QUOTA UNLIMITED ON ECOMMERCE_I;
```

Figura 12. Script para criação do usuário do banco de dados e grants

5.1.1 Estrutura física das tablelas do schema ECOMMERCE_I

Para a estrutura física das tabelas para testes de performance e para melhor entendimento do tunning em consultas DML, foram criadas 6 tabelas, como um sistema básico de ECOMMERCE_I, e inserção de informação nas tabelas, os quais se deram da seguinte maneira como apresentado nas figuras abaixo:

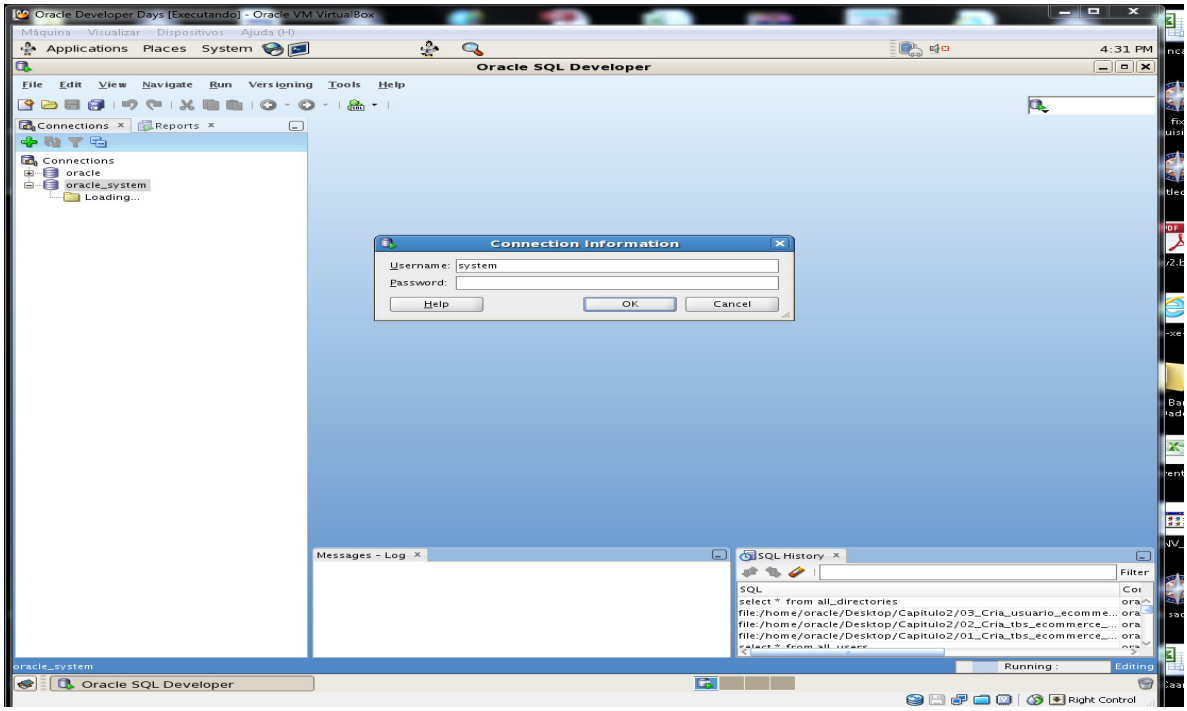


Figura 13. Conexão com o Banco de Dados

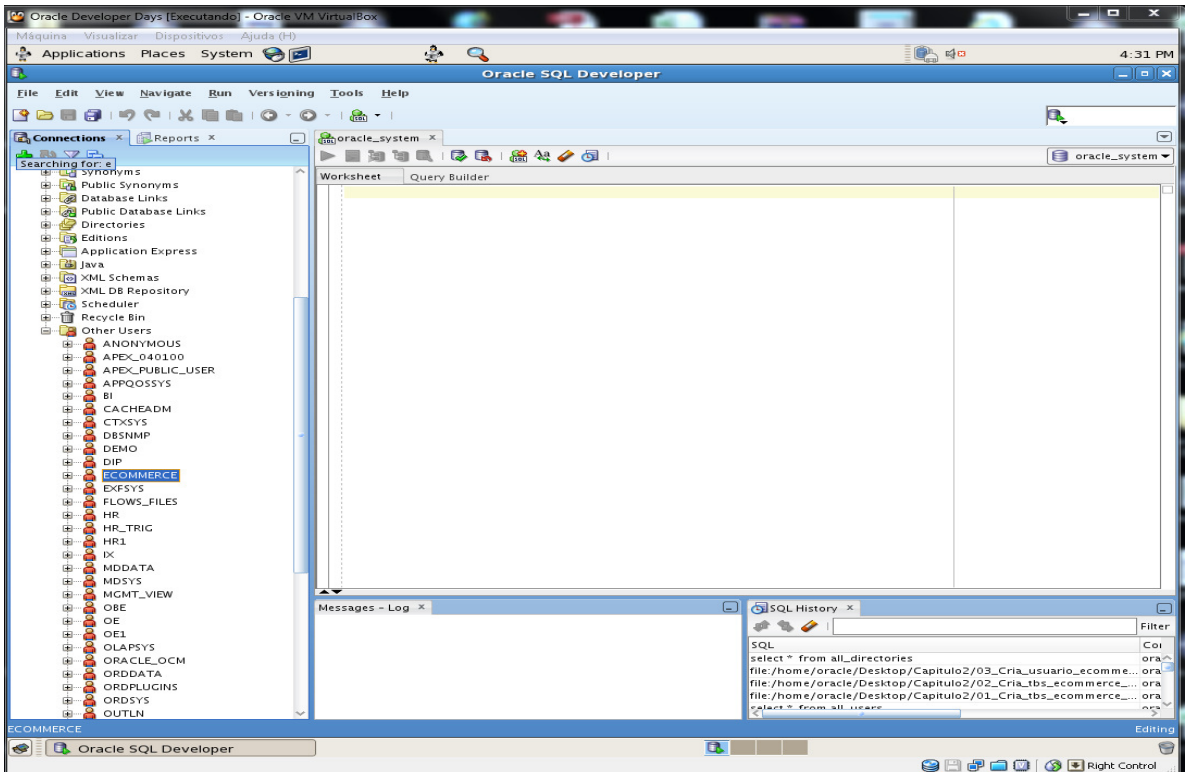


Figura 14. Schema criado ECOMMERCE_I

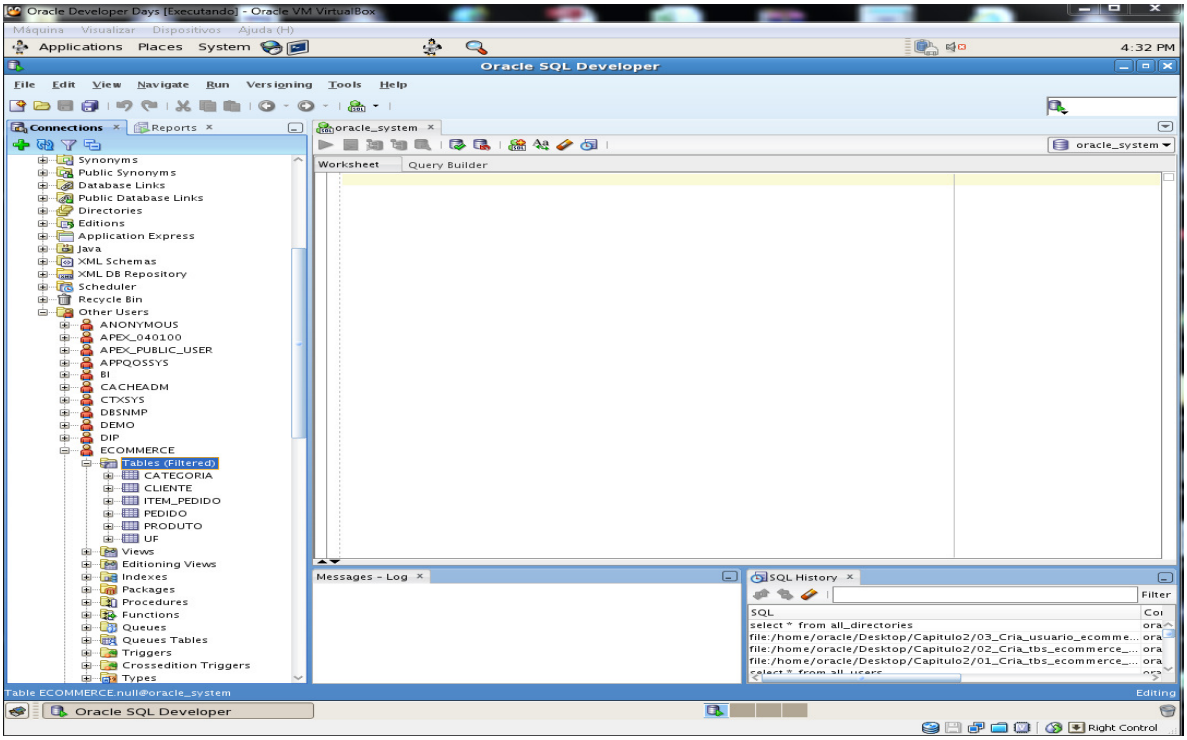


Figura 15. Tabelas para testes de tuning

5.2 Estatísticas no RDBMS Oracle

O otimizador é um programa que realiza as transformações de instruções DML em planos de execução. Com a versão do Oracle 7 introduziu-se o CBO (Cost Based Optimizer), que se baseia-se no custos das operações, nas versões 10 G e 11 G, o processo ficou default no Oracle, por meio da tarefa que é rodada no banco entre as 22:00 horas e 02:00 horas, somente se o banco de dados estiver ocioso, com baixo processamento.

Tal tarefa se mostra importante, logo pensando em um banco de dados 24x7, onde essas estatísticas estarão desatualizadas.

De acordo com LEWIS (2005), o CBO gera suas estimativas para acesso aos dados baseado nas métricas de custo, cardinalidade e seletividade, sendo influenciado por parâmetros, estatísticas e hints (diretivas de compilação). Daí a importância de se entender como as statistics funcionam e auxiliam o CBO na geração do plano de execução.

Com isso, se o banco de dados tiver uma estatística desatualizada haverá perda no desempenho das instruções.

Como apenas foram criadas tabelas e gerados processo de carga automática nas tabelas, as estatísticas estão zeradas, conforme segue:

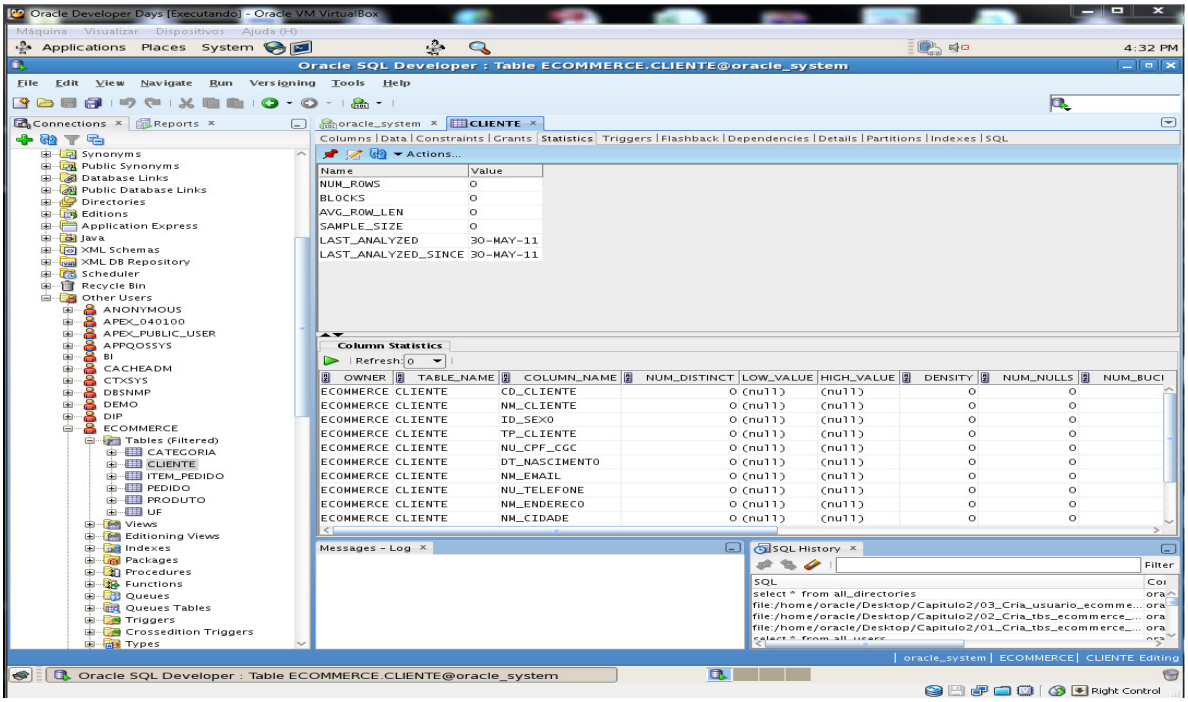


Figura 16. Estatísticas no RDBMS da tabela Cliente

A fim de contornar tal situação, pode-se rodar o processo de estatística, conforme o script abaixo:

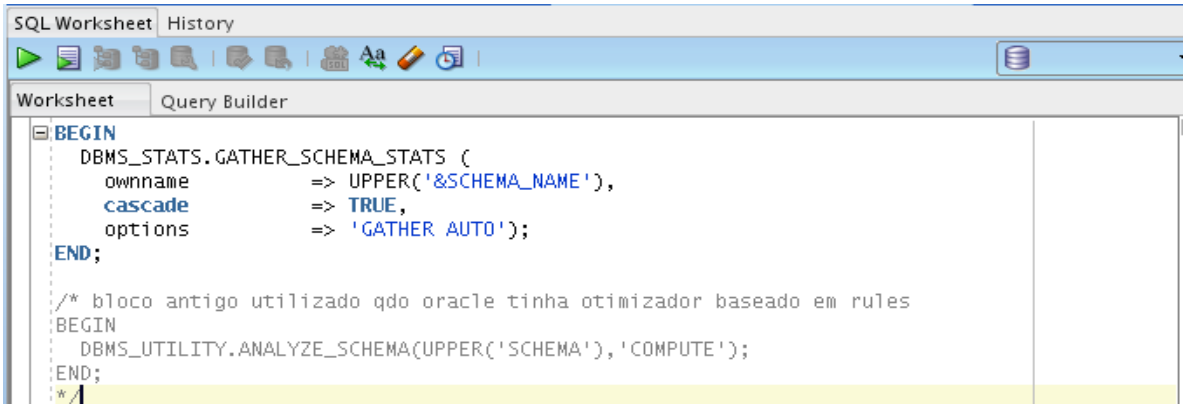


Figura 17. Script DBMS_STATS

Este script tem a função de atualizar todas as estatísticas do *schema* passado por parâmetro, através da chamada da package core DBMS_STATS, que chama a procedure core GATHER_SCHEMA_STATS que tem a função de coletar estatísticas dos objetos.

A procedure recebe os parametros owname (nome do schema), cascade (com a opção TRUE que irá gerar estatísticas de tabelas e índices, todos os objetos adjacentes) e options (caso tenha um banco de dados 24x7 com tamanho relativamente grande, deve-se ter cuidado ao gerar estatísticas para todos os objetos, pois isso afetaria o desempenho do mesmo, como o banco trabalhado possui poucos dados, a opção GATHER AUTO, tem a função de deixar o banco atualizar todas as estatísticas desatualizadas).

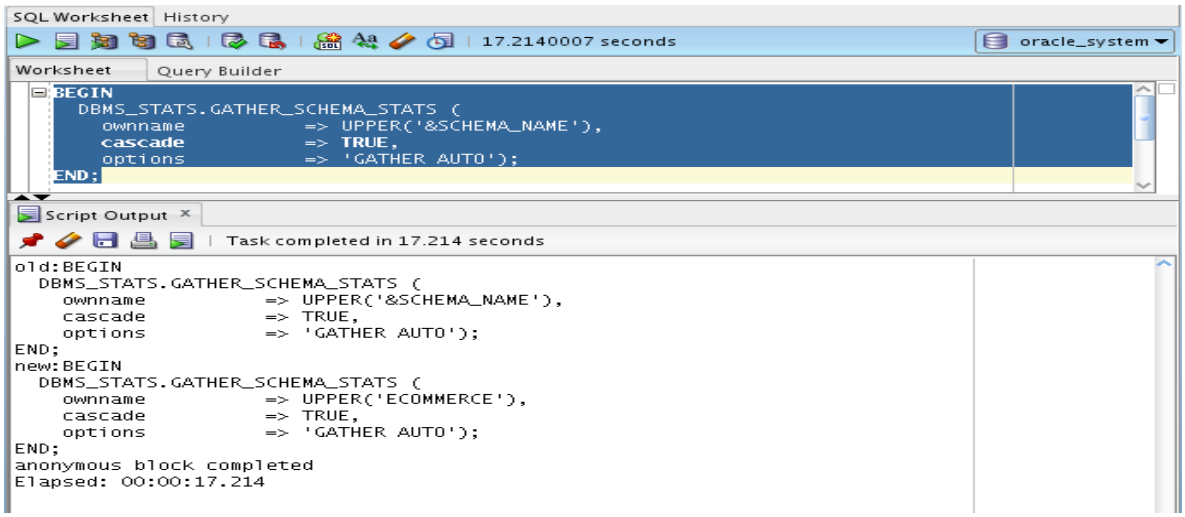


Figura 18. Terminada a coleta de estatística das tabelas

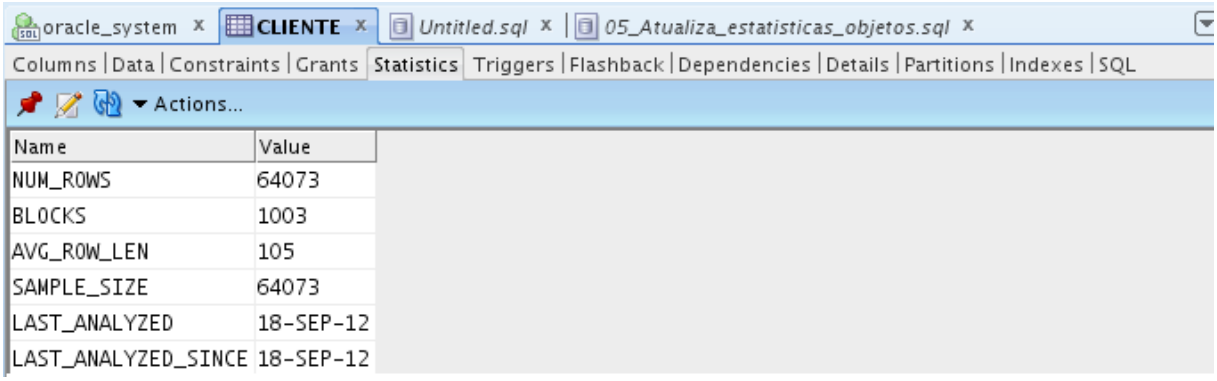


Figura 19. Tabela Atualizada

Para teste realizou-se uma pequena instrução, utilizando-se do SQL PLUS no Linux, ativando os parâmetros de timing e autotrace para verificação do tempo de processamento e seus passos, como segue:

```
Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.2.0 - Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> set timing on
SQL> set autotrace traceonly

SQL> select * from ecommerce.produto p, ecommerce.item_pedido ip where p.cd_produto =
ip.cd_produto;

SQL> /

1138048 rows selected.

Elapsed: 00:00:21.92

Execution Plan
----------------------------------------------------------
Plan hash value: 1270662005

-------------------------------------------------------------------------------------
| Id  | Operation                    | Name         | Rows  | Bytes | Cost (%CPU)
| Time     |

-------------------------------------------------------------------------------------

|   0 | SELECT STATEMENT             |              |     1 |   246 |     2   (0)
| 00:00:01 |

|   1 |  NESTED LOOPS                |              |       |       |
|          |

|   2 |   NESTED LOOPS               |              |     1 |   246 |     2   (0)
| 00:00:01 |

|   3 |    TABLE ACCESS FULL         | ITEM_PEDIDO  |     1 |    52 |     2   (0)
| 00:00:01 |

|*  4 |    INDEX UNIQUE SCAN         | PK_PRODUTO   |     1 |       |     0   (0)
| 00:00:01 |

|   5 |   TABLE ACCESS BY INDEX ROWID| PRODUTO      |     1 |   194 |     0   (0)
| 00:00:01 |

-------------------------------------------------------------------------------------


Predicate Information (identified by operation id):
---------------------------------------------------

   4 - access("P"."CD_PRODUTO"="IP"."CD_PRODUTO")

Statistics
----------------------------------------------------------
          0  recursive calls
          0  db block gets
    1292447  consistent gets
          0  physical reads
          0  redo size
  111869380  bytes sent via SQL*Net to client
    1517817  bytes received via SQL*Net from client
      75871  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
    1138048  rows processed
```

Figura 20. Antes de gerar as estatísticas

```
SQL> select * from ecommerce.produto p, ecommerce.item_pedido ip where p.cd_produto =
ip.cd_produto;

SQL> /

1138048 rows selected.

Elapsed: 00:00:21.21

Execution Plan
----------------------------------------------------------
Plan hash value: 1792694285

-------------------------------------------------------------------------------
--

| Id  | Operation          | Name        | Rows  | Bytes | Cost (%CPU)| Time
 |

-------------------------------------------------------------------------------
--

|   0 | SELECT STATEMENT   |             |  1138K|    93M|   798   (2)| 00:00:10
 |

|*  1 |  HASH JOIN         |             |  1138K|    93M|   798   (2)| 00:00:10
 |

|   2 |   TABLE ACCESS FULL| PRODUTO     |     5 |   365 |     3   (0)| 00:00:01
 |

|   3 |   TABLE ACCESS FULL| ITEM_PEDIDO |  1138K|    14M|   791   (2)| 00:00:10
 |

-------------------------------------------------------------------------------
--


Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("P"."CD_PRODUTO"="IP"."CD_PRODUTO")


Statistics
----------------------------------------------------------
          0  recursive calls
          0  db block gets
    1292447  consistent gets
          0  physical reads
          0  redo size
  111869380  bytes sent via SQL*Net to client
    1517817  bytes received via SQL*Net from client
      75871  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
    1138048  rows processed
```

Figura 21. Após atualização das estatísticas e índices do schema

Com isso, percebe-se que o processamento cai, onde o inicial foi de elapsed: 00:00:21.92, para após a atualização das estatísticas para Elapsed: 00:00:21.21, onde o Oracle decidiu por utilizar as informações de estatísticas das tabelas e diminuiu suas regras na otimização. Isso pode ser pouco significativo nessas tabelas

por conterem poucos dados, mas deve-se ter em mente tal procedimento em um sistema real, onde o resultado de melhora de processamento fica claro.

5.3 Modelagem Física do Banco de Dados

Nesta subseção tratar-se-á dos tipos de dados em uma tabela do Oracle.

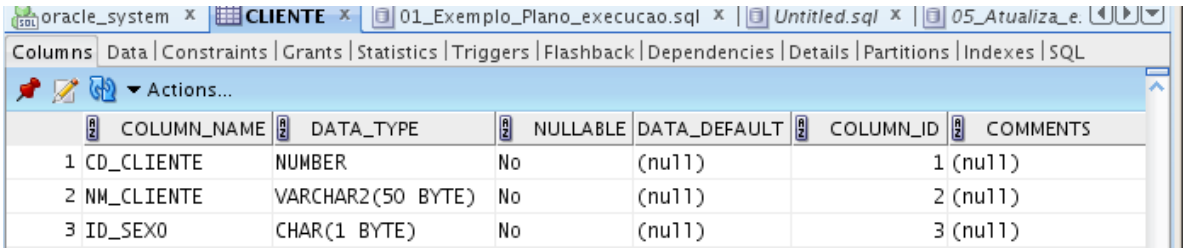


Figura 22. Trecho de campos da tabela cliente

Toda a chave primária de uma tabela deve ser criada com o tipo NUMBER, salvo exceções em que, por sua modelagem seja necessário outro tipo de dado como chave primária. Tempo de respostas de instruções DML em tabelas com chave primária são maiores do que em tabelas com chave primária VARCHAR2, por exemplo.

Na coluna 2 da figura, pode-se notar, que o nm_cliente (nome do cliente) tem tipo de dado VARCHAR2, isso por que o nome de um cliente é dinâmico, não se tem um tamanho pré definido, quando se define VARCHAR2 em uma tabela, o Oracle aloca somente o tamanho usado de dados em bytes para esse campo, ao contrário do tipo de dados CHAR, que aloca o tamanho dimensionado pelo desenvolvedor. O Tipo CHAR só deve ser usado quando se tem certeza do tamanho do dado a ser inserido no campo, como desenvolvedor utilize sempre VARCHAR2, ora para não se confundir com inúmeros campos.

Sempre que houver a necessidade de criação de um campo flag, como exemplo item ativo ou inativo no sistema, sempre usar tipo de dados NUMBER, por ser mais rápida a resposta, segue exemplo abaixo, e colocar comentário no campo para identificação do mesmo.

13	ID_STATUS	NUMBER(1,0)	No	1	13	Valores: 1='ativo', 2='inativo'

Figura 23. Campo flag NUMBER

5.3.1 Índices em tabelas

De acordo com VALIATE (2012):

> os índices são estruturas opcionais associadas a tabelas e 'clusters' que permitem que as consultas SQL sejam executadas mais rapidamente. Assim como o índice analítico ou remissivo numa revista lhe ajuda a encontrar mais rapidamente informações, um índice de Oracle fornece um trajeto de acesso mais rápido aos dados da tabela. Você pode usar índices sem reescrever nenhuma consulta, os resultados das consultas são os mesmos, mas você os vê mais rapidamente (VALIATE, 2012, p. 01).

Os aspectos negativos com criação de índices, é que degradam a performance das atualizações (INSERT, UPDATE, DELETE e MERGE), e também consomem espaço em disco. Então quanto mais índices uma tabela possuir, menor será sua performance com atualizações. Sempre que criar um índice, rever se há um grande número de consultas nessa coluna. A Oracle diz que um índice pode degradar 3 vezes a performance de uma atualização, então imaginemos agora uma tabela com 10 índices, onde a perda de performance de atualização é de 30 vezes.

5.3.1.1 Tipos de índices e dicas de criação.

Oracle 11G disponibiliza diferentes tipos de índices e opções de armazenamento dos mesmos, providenciando melhor desempenho, disponibilidade e gerenciamento.

Segue abaixo, alguns exemplos de índices no Oracle 11 G, os quais serão tratados individualmente:

- Índices B-tree;
- Índices Bitmap;
- Índices Function Based;
- Índices de domínio.

5.3.1.1.1 Índices B-Tree:

Este índice possui alta cardinalidade, ou seja, tem maior desempenho em colunas de dados que nunca ou pouco se repetem, mais indicados como exemplo para índices de PK, este índice é o default do Oracle.

Abaixo demonstra-se sua estrutura física para melhor entendimento:

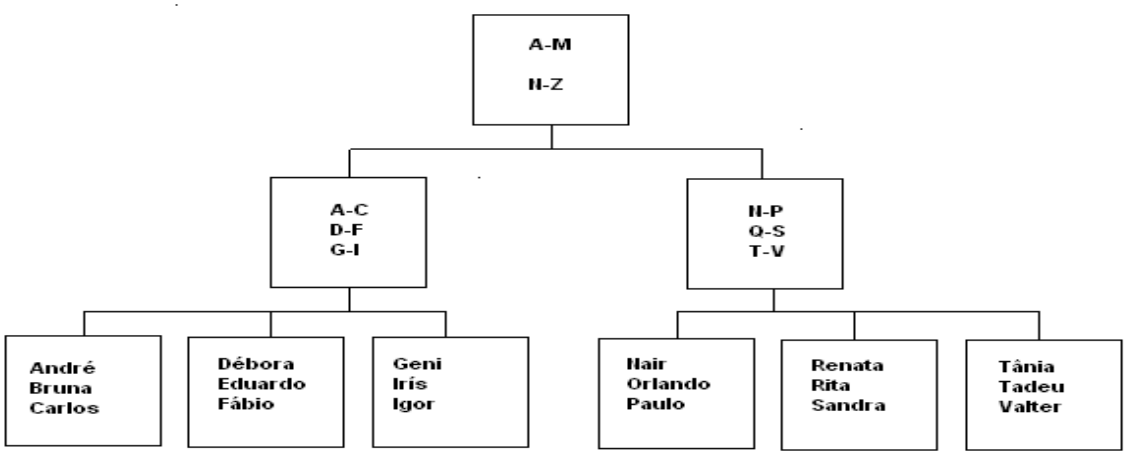


Figura 24. Estrutura do Índice B-Tree

Este índice, conforme se pode entender por seu próprio nome, baseia-se numa árvore, onde começa a divisão pelo valor através da Raiz, por exemplo, para se

chegar ao Igor, o índice se separa da busca na raiz, de A–M, separando no próximo nó dentro de G–I, até chegar na folha e encontrar o Igor, com isso são feitos três testes da raiz, do primeiro nó e da folha, reduzindo o caminho de busca.

Existem certas ocasiões onde um índice B-Tree não é apropriado e não irá melhorar a performance de consultas. Em muitas destas situações, tal como criar um índice sobre uma aplicação de data warehouse, onde a coluna indexada tem poucos valores distintos, um índice bitmap pode ser criado e melhorar bastante a performance (WHALEN, 1996).

5.3.1.1.2 Indice Bitmap.

Ao inverso do B-Tree, este índice otimiza consultas de baixa cardinalidade em colunas, ou seja, onde há pouca repetição de dados.

De acordo com (BURLESON, 1997):

> os índices Bitmap foram introduzidos com o Oracle 7.3. Em um índice bitmap, o Oracle cria um bitmap para cada valor único da coluna em questão. Cada bitmap contém um único bit (0 ou 1) para cada linha na tabela. Um “1” indica que a linha tem o valor especificado pelo bitmap e um “0” indica que não tem.

O Oracle pode rapidamente percorrer estes bitmaps para encontrar linhas que satisfazem um critério específico, também pode rapidamente comparar múltiplos bitmaps para encontrar todas as linhas que satisfazem aos múltiplos critérios.

Esses índices possuem desempenho desfavoráveis para colunas que sofrem atualizações constantes.

	LINHA																		
VALOR	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Masculino	0	0	1	1	1	0	1	0	0	0	0	1	0	1	1	1	0	1	0
Feminino	1	1	0	0	0	1	0	1	1	1	1	0	1	0	0	0	1	0	1

Figura 25. Estrutura de índice Bitmap

5.3.1.1.3 Índices Function Based

Este índice baseia-se no retorno de uma função ou expressão. Foi introduzido no Oracle 8i, pode ter a estrutura de um índice Bitmap ou B-tree. Para otimização de desempenho somente é utilizado quando se faz necessário utilizar de funções em consultas, com condições seletivas.

5.3.1.1.4 Índice de Domínio

Este grupo de índice são criados para colunas com tipos de dados especiais. Como exemplo, se possui-se uma coluna de imagem (geográficas) e deseja-se fazer consultas, o índice indicado no Oracle para esta coluna é o Oracle Spatial. Se possui-se uma coluna de Varchar2 como quantidade de texto elevado ou tipo de dado Clob, o índice indicado para esta situação é o Oracle Text.

5.4 Entendendo como o Oracle Database processa as instruções SQL.

Antes de iniciar o desenvolvimento de técnicas de tuning em instruções DML, deve-se entender como o Oracle Database processa as instruções SQL. No Oracle os principais conceitos envolvidos na execução das instruções são:

- Plano de Execução;
- Otimizador de Query;
- Métodos de Acesso aos dados;
- Métodos de ligações.

5.4.1 Plano de Execução

Define-se como uma sequência de operações que o Oracle realiza para executar as instruções DML. O plano de execução se mostra em forma de árvore de linhas, onde contém os seguintes passos:

- Ordenação das tabelas referenciadas pela instrução;
- Método de acesso para cada tabela mencionada na instrução;
- Método join para as tabelas afetadas pelas operações join da instrução;
- Operações de dados tais como filter, sort ou agregação;
- Otimização: custo e cardinalidade de cada operação;
- Particionamento: conjunto de partições acessadas;

Para se ter acesso ao plano de execução basta acrescentar a instrução o cabeçalho (EXPLAIN PLAN FOR). Nesse momento será criado os passos necessários para gerar a instrução na tabela core plan_table. Para uma melhor visualização do plano de execução utiliza-se da package core DBMS_XPLAN.DISPLAY, onde se terá em tela uma melhor visualização dos passos para geração da instrução DML, como segue melhor demonstrado na figura abaixo:

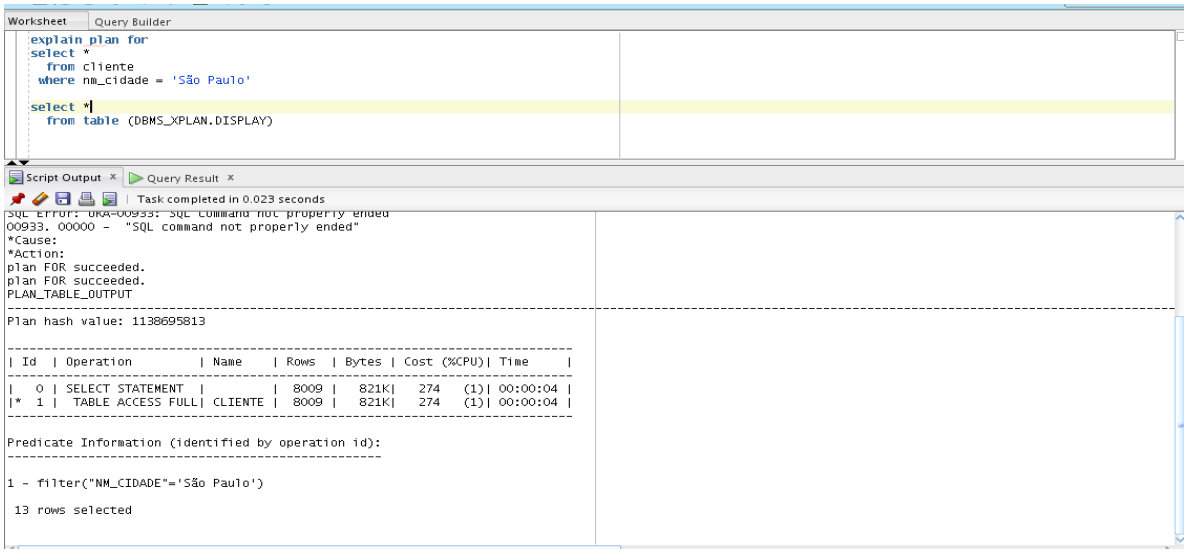


Figura 26. Plano de Execução

Através do plano de execução pode-se analisar as decisões do Otimizador de Query do Oracle, analisando assim sua performance. Através deste plano pode-se analisar se uma query acessa os dados atraves de *full table scan* (caminho de acesso em que os dados são recuperados percorrendo todas as linhas de uma tabela. Tem maior eficiência em recuperar uma grande quantidade de dados da tabela ou os dados de tabelas com poucas linhas), ou *index lookup* (Caminho de acesso em que os dados são recuperados atraves de indices. É mais eficiente para recuperar dados de um pequeno conjunto, 4 % de linhas de uma tabela), e o tipo de join que ela efetuou: seja *nested loops join* (Método de acesso de ligação entre tabelas, utilizado quando pequenos conjuntos de dados estão sendo ligados e se a condição de ligação é um caminho eficiente para acessar a segunda tabela), ou um *hash join* (Método de acesso de ligação (join) entre 2 tabelas ou origens de dados, utilizado para ligar grandes conjuntos de dados).

O plano de execução varia de acordo com o ambiente que está sendo executado, ou seja, pode-se gerar um plano de execução da mesma query em ambientes diferentes e ter diferentes valores, como exemplo de ambientes de desenvolvimento para um ambiente de produção. O plano pode ser diferente se for também gerado por schemas diferentes, pois leva em consideração o volume de dados, as estatísticas, parâmetros de bancos, parâmetros de servidor, sessão, etc.

5.5 Tunando Atualizações de dados no Oracle Database.

5.5.1 DELETE.

5.5.1.1 Comando Truncate

Quando se deseja realmente apagar todos os dados de uma tabela, deve-se ter em mente que este comando é irreversivel, pois não há ROLLBACK de suas transações, uma vez que não gera logs e nem UNDO. Este comando é

extremamente mais rápido do que o comando DELETE, pois redefine a estrutura da tabela, onde a Marca d´água vai para o início da tabela. Além de mais rápido que o comando DELETE, o comando truncate reorganiza a estrutura da tabela. O comando delete deixa espaços vazios na tabela, e estes espaços “blocos de dados” vazios fazem com que o insert na tabela demore mais.

Segue exemplo de tempo de performance entre o comando DELETE e TRUNCATE abaixo:

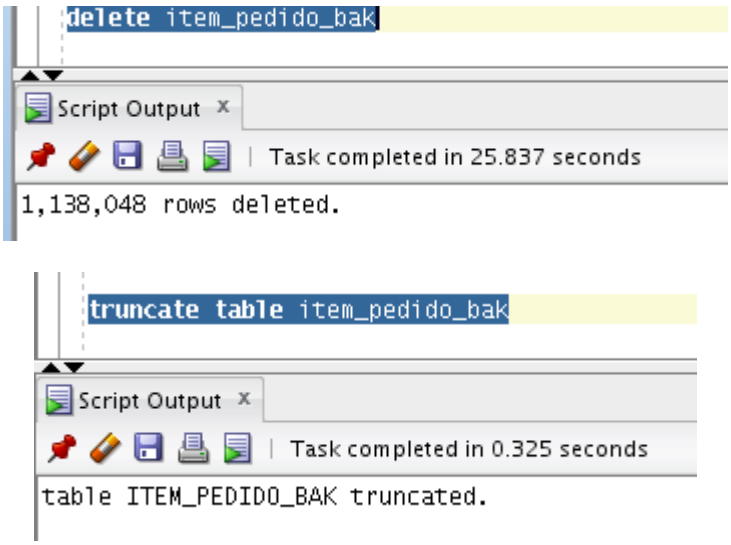


Figura 27. Tempo de resposta entre o comando DELETE e TRUNCATE

5.5.1.2 Comando Bulk Binding no DELETE

Quando há a necessidade de apagar vários registros em uma tabela dentro de um bloco autônomo de PL/SQL, deve-se utilizar do bulk binds, como exemplo o FORALL, na instrução DELETE.

É extremamente mais rápido do que um loop convencional, pois com FORALL ele irá ler toda a seleção da tabela, todas as linhas e enviar para o DELETE de uma vez, ou seja, N para 1, quando se faz um loop normal, o Oracle lê a linha e envia para o DELETE, sendo 1 para 1.

Como exemplo de ganho de performance segue abaixo um trecho de bloco em PL-SQL, nesse bloco carrega-se 10.000 (dez mil) linhas em uma coluna de 1 a 10.000 e deleta-se os dados dentro de um FOR normal e testo o tempo com um FORALL usando bulk binding.

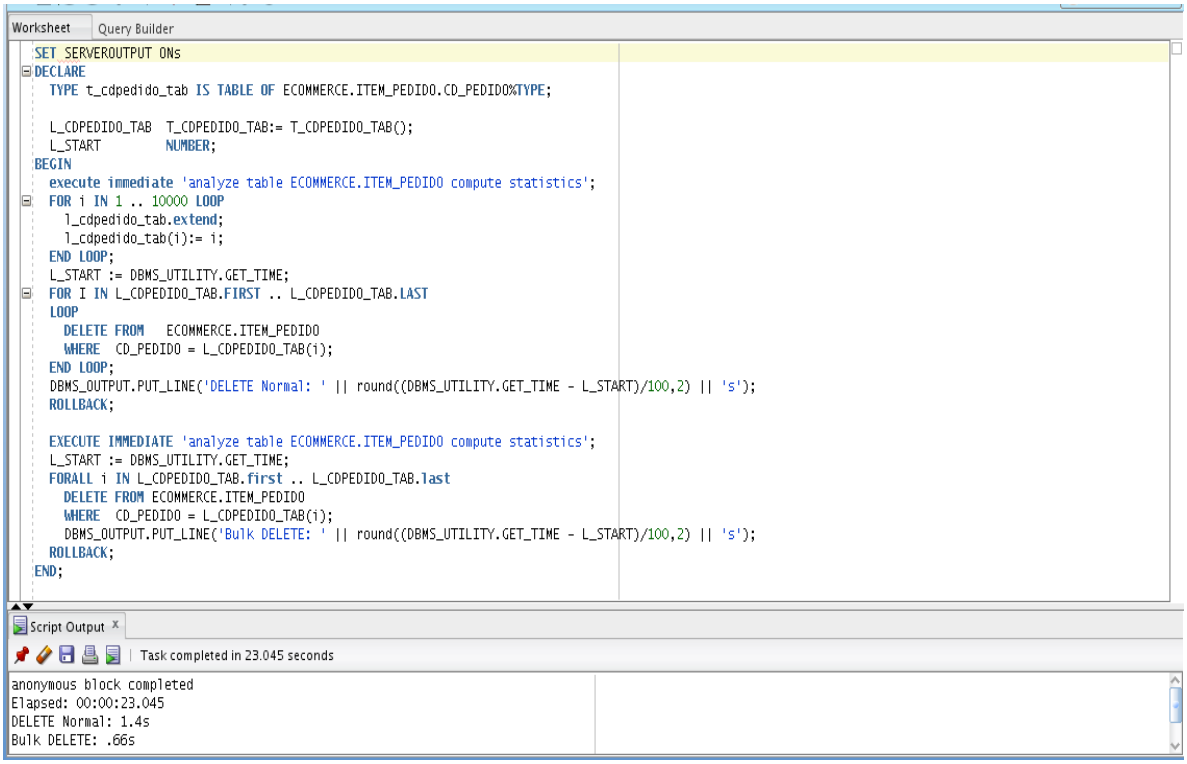


Figura 28. Teste DELETE com FORALL

Como segue figura acima, ativou-se a funcionalidade SERVEROUTPUT para on, a fim de ter a saída em tela, criando ainda uma collection para a estrutura da tabela. Coletou-se antes dos dois processos de loop normal e com FORALL, as estatísticas dos dados, com isso tendo um retorno de desempenho maior. Utilizou-se a package DBMS_UTILITY.GET_TIME, para pegar a hora atual. Assim dando o tempo de execução real desde a primeira interação do loop ate a sua finalização. E com isso percebe-se a grande diferença de desempenho entre um FOR normal e utilizando de Bulk Binding com FORALL, onde o FOR normal terminou a deleção de 10.000 linhas em um tempo de 1.4 s e com FORALL o tempo foi reduzido para 0.66 s, resultando em quase 3 vezes no ganho de performance.

5.5.1.3 Comando DELETE com PARALLEL.

Nesse comando pode-se inserir na instrução uma sugestão ao otimizador de quantos processadores usar para rodar determinada instrução, isso não quer dizer que o otimizador irá seguir essa sugestão, o mesmo irá analisar se é prudente, caso sim realizará a instrução seguida da sugestão. Exemplo segue abaixo:

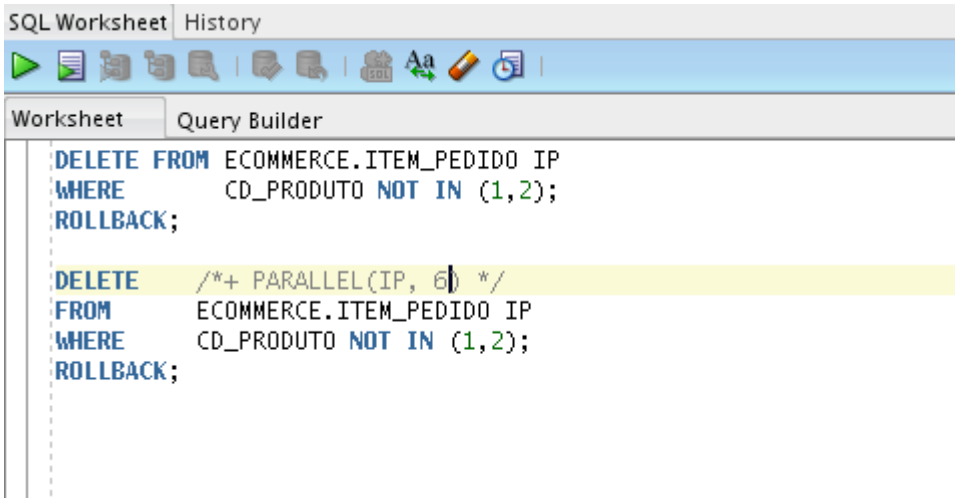


Figura 29. Utilizando DELETE com PARALLEL

Na figura acima, a dica dada para o otimizador é utilizar 6 processadores para rodar a instrução, onde os hints começam com /*+ e terminam com */, utilizando como mostra a figura o PARALLEL com dois parâmetros, e o nome da tabela que, no caso foi utilizado como apelido da tabela e a quantidade de processadores.

Neste caso, não teve como se medir o tempo de ganho que é bem significativo, pois os testes foram realizados em uma máquina virtual com Oracle Express Edition, e nesta versão não há disponibilidade da função de hint PARALLEL, logo disponível em bancos Enterprise.

5.5.2 INSERT.

5.5.2.1 Comando Insert com Direct Path

Este comando deve ser usado quando se deseja ganhar performance na inserção de muitas linhas, ou inserção dentro de um bloco PL-SQL, sempre quando se prioriza a performance da inserção, neste caso pode-se ser usado o insert com hint APPEND.

É mais rápido que um insert convencional, pois insere no fim da tabela acima da HWM (high water mark), porém, neste momento o APPEND bloqueia a tabela, então deve-se tomar cuidado com o concorrência de atualização nessa tabela. Segue abaixo o script.

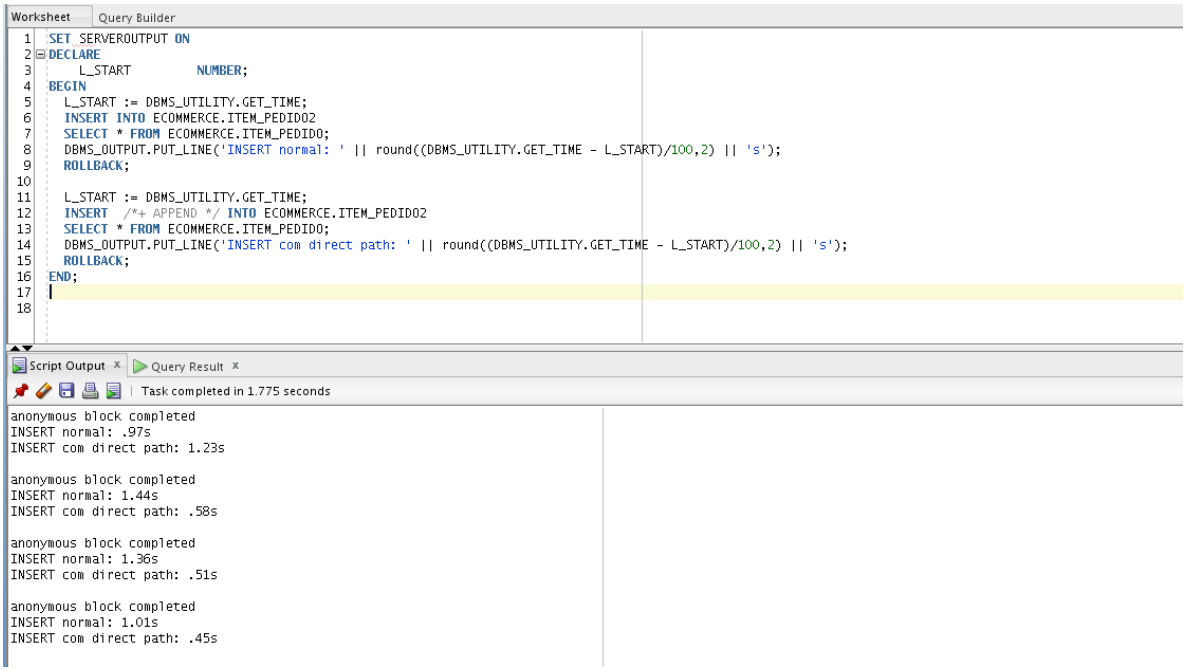


Figura 30. Insert com Direct Path

Como demonstrado na figura acima, o insert com direct path é muito mais rápido que o insert convencional, na primeira rodada, o insert foi mais rápido por que havia sido criada uma tabela vazia, e com isso não continha espaços em brancos, e assim que foi rodada pela segunda vez, o rollback criou os espaços vazios na tabela.

5.5.2.2 Comando Insert com Sequence

Segundo Prado (2011):

> ao utilizar SEQUENCES para inserir valores únicos em colunas de uma tabela, utilize SEQUENCES com CACHE. Sequences em cache ocupam pouca memória na Shared Pool e são muito eficientes para otimizar performance. O comportamento padrão do Oracle na criação de sequences, é criá-las com 20 valores em cache.

Para comprovar o que Prado afirma, segue abaixo a implementação, onde criou-se duas sequences, uma com cache padrão do Oracle de 20 e outra sequence sem cache, criou-se ainda, uma tabela, com a inserção de valores, ora utilizando sequence com cachê, ora utilizando sequence sem cache, e assim chega-se ao resultado de que a sequence com cache obtém maior performance nas inserções, conforme demonstram as figuras abaixo de criação das sequences e da execução do script:

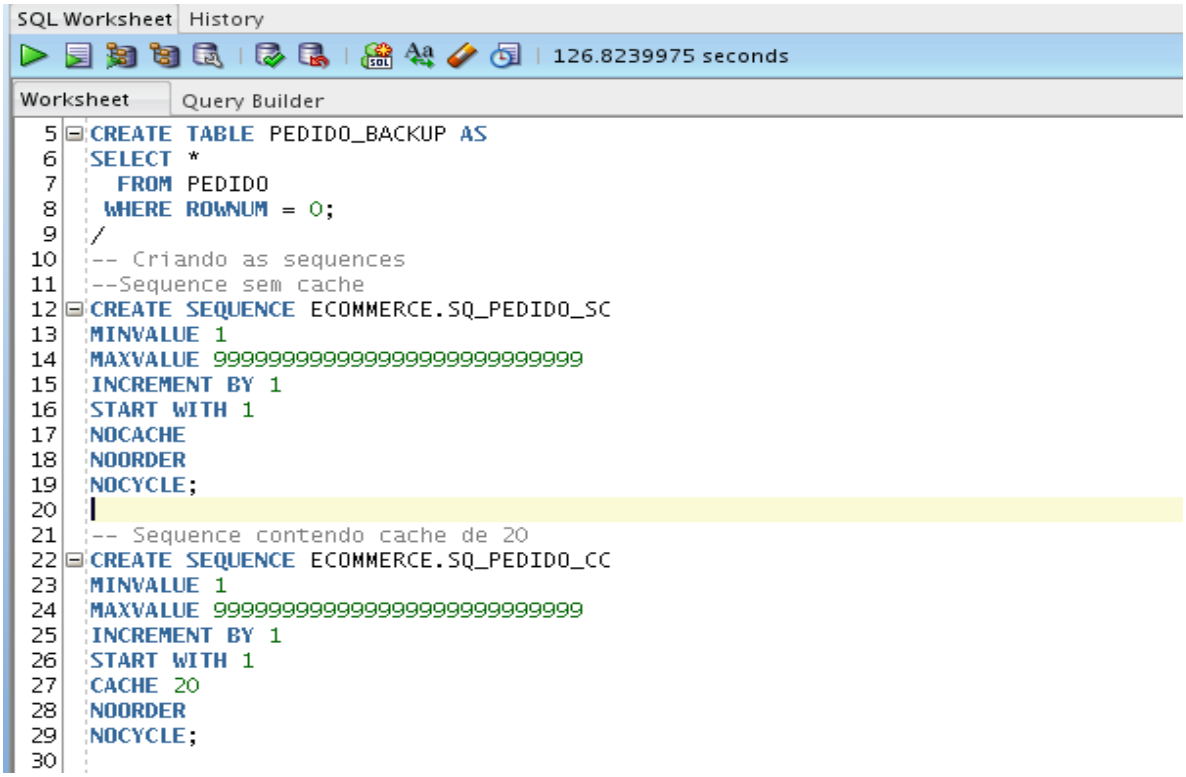


Figura 31. Criação de Sequences.

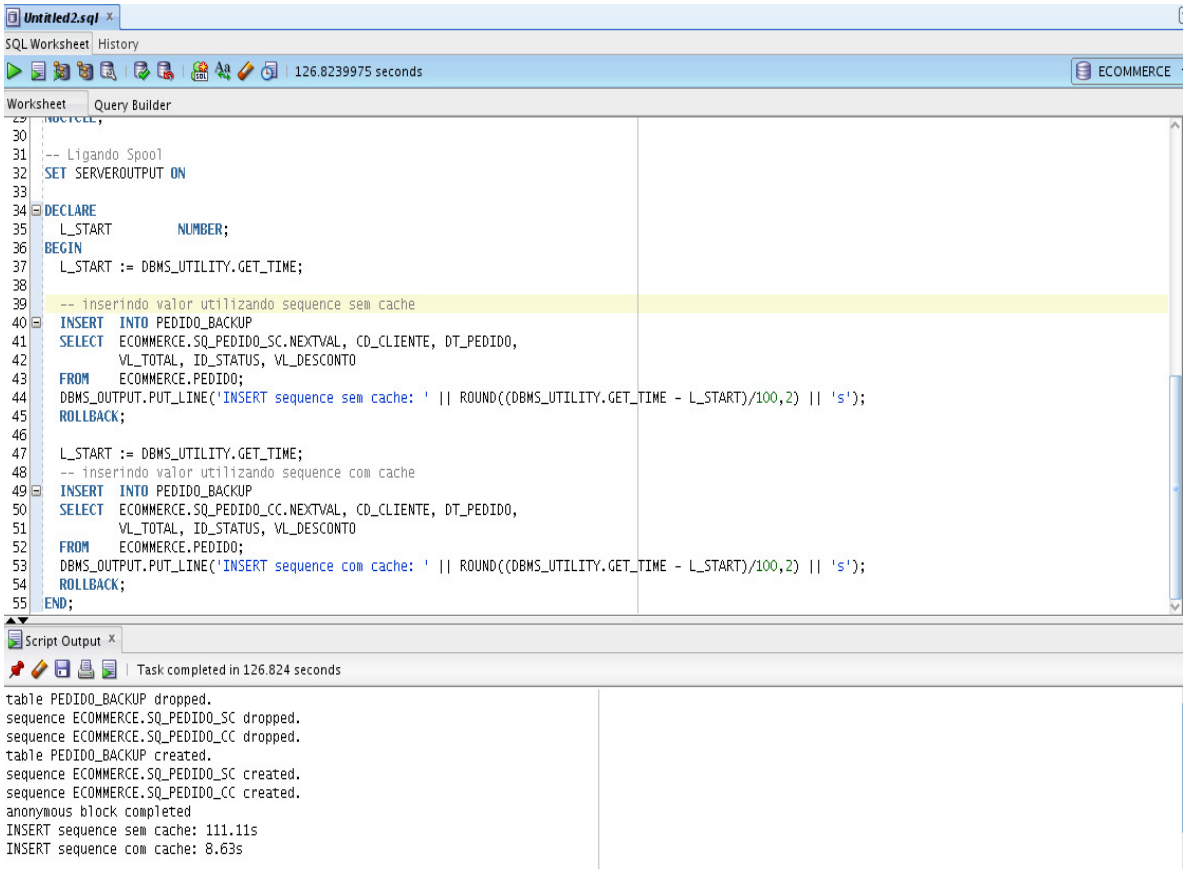


Figura 32. Tempo entre Insert com Sequence com Cache e Sem cache

5.5.2.3 Comando Insert com Bulk Collection

Quando se deseja inserir dados em uma tabela através de um loop, deve-se levar em consideração a possibilidade de fazer isso através de bulk collection, que é mais rápido que um loop convencional, pois lê N quantidades de dados para inserir, ao contrário de um loop convencional que lê 1 linha do cursor para fazer o insert sendo então 1 para 1.

Como exemplo, pode-se tomar a figura abaixo, que mostra o ganho de desempenho utilizando bulk collection no insert dentro de um bloco PL-SQL com LOOP, como pode-se perceber o ganho de performance com inserção de 100.000 linhas é em muito superior utilizando BULK COLLECTION no insert.

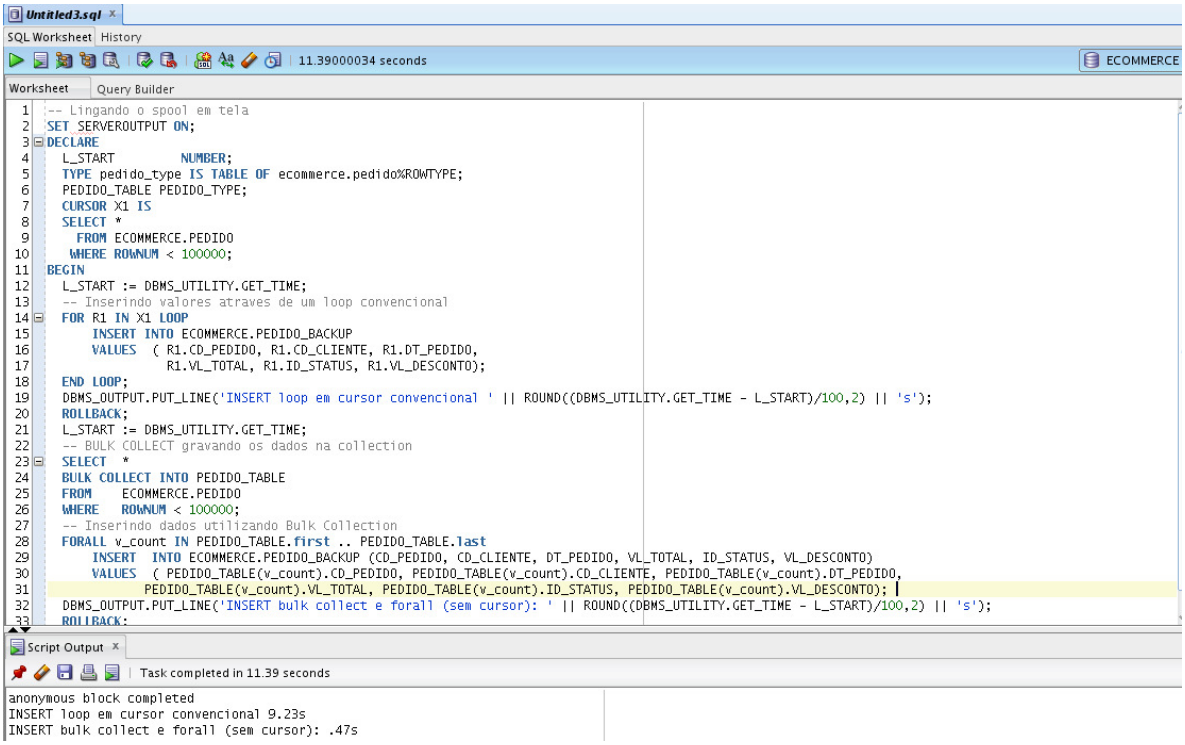


Figura 33. Insert com Bulk Collection

5.5.2.4 Comando INSERT com Pipelined Table Function

Este comando exige um nível de conhecimento mais elevado, possui ganho de performance em otimizar funções ou procedimentos complexos. Permite retornar dados como se fosse uma tabela virtual, no momento em que cada linha está sendo processada transforma os dados, excelente quando a aplicação necessita de um export de dados em formatos diferentes do banco de dados.

Dispensar-se-á a submissão de um script para testes no presente caso, pois o ganho de otimização são muito raros de serem verificados, e com bulk collection tem-se um maior ganho de desempenho.

5.5.2.5 Comando INSERT com SQL LOADER

Quando se deseja efetuar uma carga de dados em um tabela, e essa massa de dados tem um tamanho favorável, um script de insert convencional normalmente é muito demorado, levando horas para terminar. Nesse momento, pode-se tunar a aplicação utilizando o conceito de carga via SQL LOADER, através de um arquivo texto que insere-se na tabela do banco de dados os dados.

A fim de demonstrar o ganho de desempenho simulou-se uma carga da tabela de cliente, onde pode-se verificar o tempo decorrido do insert e entender os comandos.

O Primeiro passo foi a criação de um arquivo texto da tabela cliente, como segue os prints abaixo:

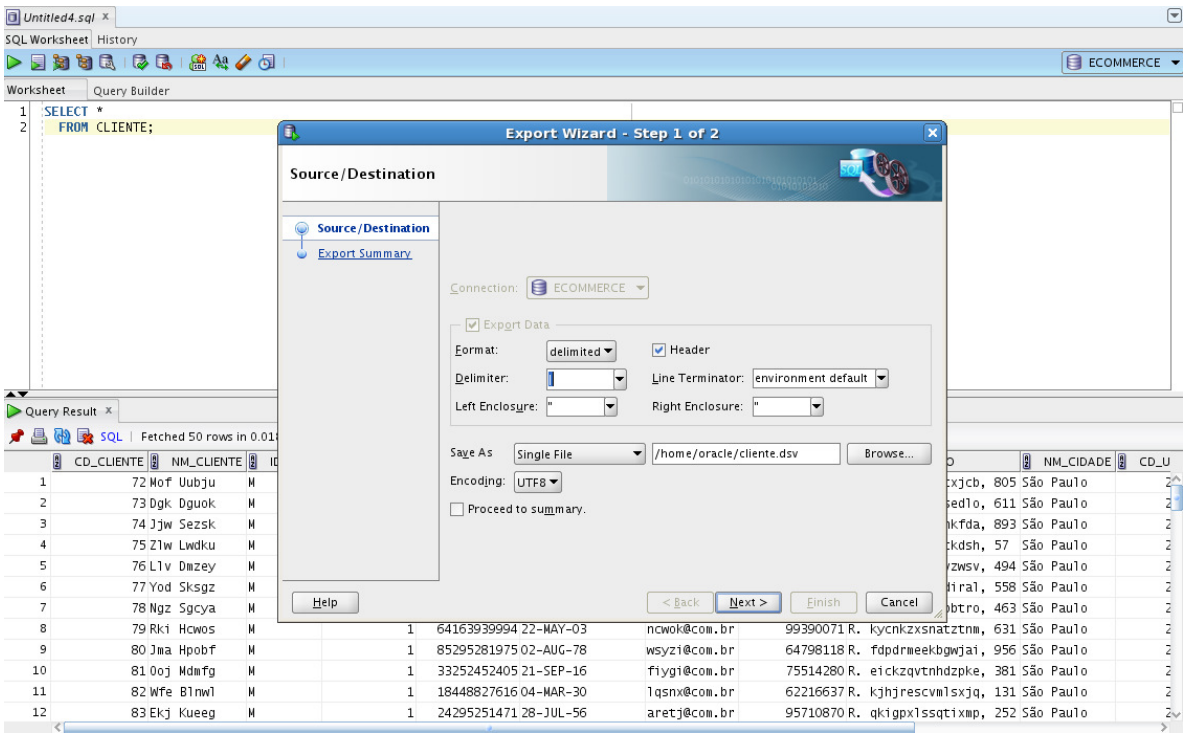


Figura 34. Criando arquivo texto, delimitado por pipe

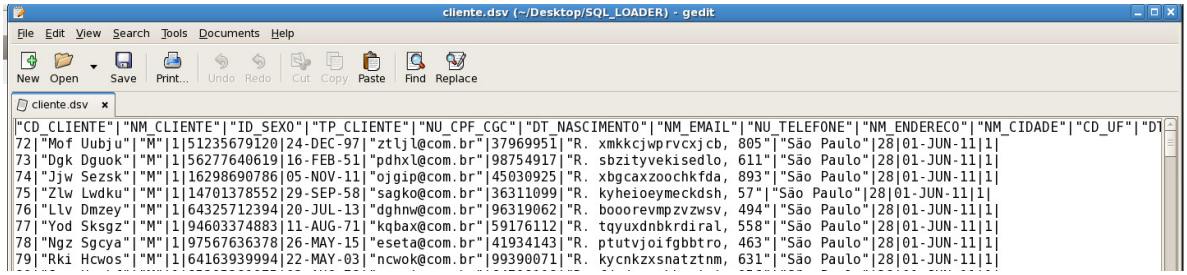


Figura 35. Estrutura do Arquivo Texto

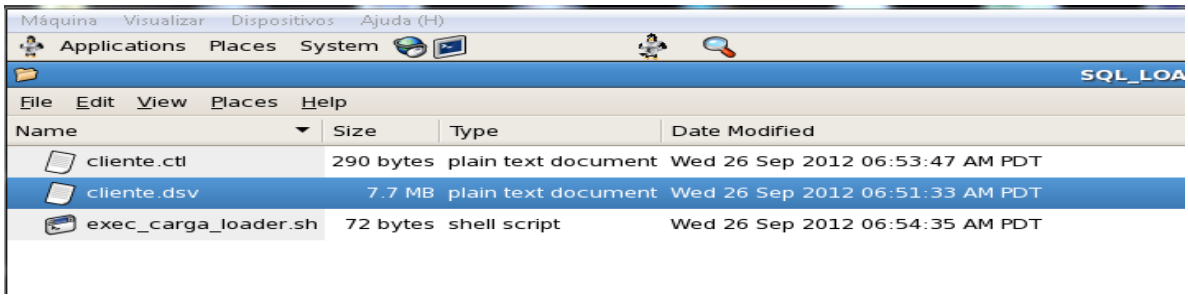


Figura 36. Arquivos necessários para carga via SQL_LOADER

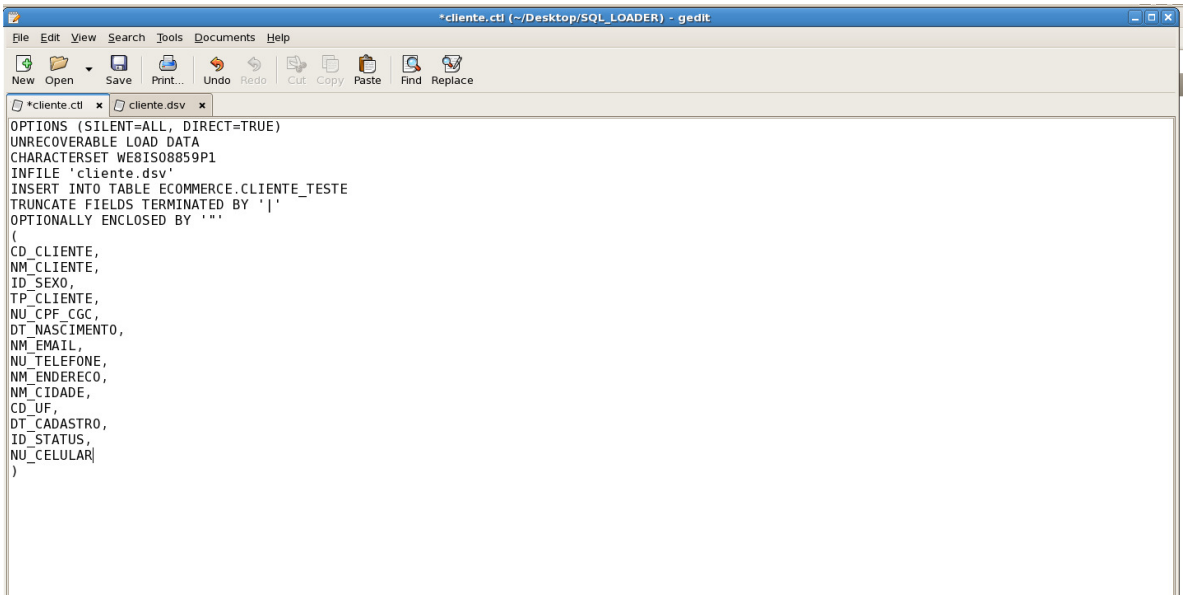


Figura 37. Cliente.ctl

A figura acima trata-se do arquivo de definição do insert, onde define-se a propriedade DIRECT = TRUE, ou seja, insert acima da HWM, e a propriedade UNRECOVERABLE, como essa propriedade é impossível voltar, o insert como ROLLBACK, define-se o tipo de caracteres que possa ter no arquivo, o nome do arquivo texto que contém os dados na propriedade INFILE, passando o insert em

qual tabela se irá fazer, dizendo que os campos são separados por pipe, que define na extração, e os campos da tabela a ser feito o insert.

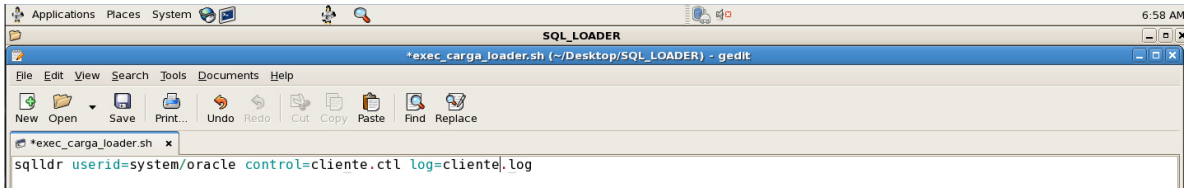


Figura 38. exec_carga_loader.sh

Na figura acima, determinou-se os parâmetros de usuário e senha, o arquivo de configuração da carga e o nome do arquivo de log que será gerado.

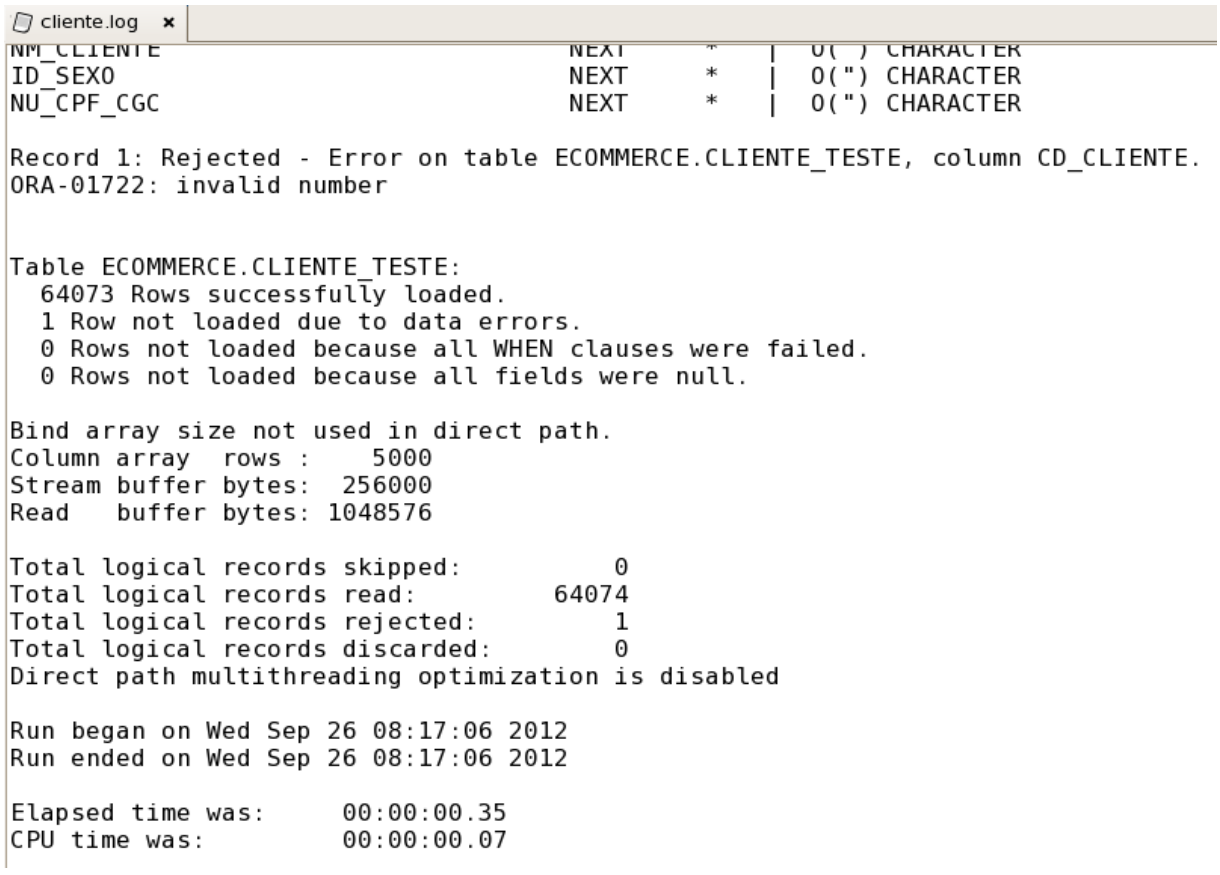


Figura 39. Tempo de carga de 35 segundos

Para cargas de mais de 64.000 (sessenta e quatro mil) linhas foram gastos apenas 35 segundos, o erro acima da linha se deu devido ao cabeçalho do arquivo.

Agora no arquivo cliente.ctl, alterou-se as configurações para geração de log e o modo agora insere nos clusters em bancos da tabela.

cliente.ctl

```
OPTIONS (SILENT=ALL, DIRECT=FALSE)
LOAD DATA
CHARACTERSET WE8ISO8859P1
INFILE 'cliente.dsv'
INSERT INTO TABLE ECOMMERCE.CLIENTE_TESTE
TRUNCATE FIELDS TERMINATED BY '|'
OPTIONALLY ENCLOSED BY '"'
(
CD_CLIENTE,
NM_CLIENTE,
ID_SEXO,
NU_CPF_CGC
)
```

Figura 40. Alteração no arquivo cliente.ctl

```
Table ECOMMERCE.CLIENTE_TESTE:
  64073 Rows successfully loaded.
  1 Row not loaded due to data errors.
  0 Rows not loaded because all WHEN clauses were failed.
  0 Rows not loaded because all fields were null.


Space allocated for bind array:                 66048 bytes(64 rows)
Read   buffer bytes: 1048576

Total logical records skipped:          0
Total logical records read:         64074
Total logical records rejected:         1
Total logical records discarded:        0

Run began on Wed Sep 26 08:18:19 2012
Run ended on Wed Sep 26 08:18:21 2012

Elapsed time was:     00:00:01.34
CPU time was:         00:00:00.19
```

Figura 41. Tempo de carga de 01 minuto e 34 segundos

Se essa carga fosse feita por um script, demoraria vários minutos, então aqui demonstra-se a eficiência da carga via SQL_LOADER, com algumas dicas para ganho de performance.

5.5.2.6 Comando insert utilizando MERGE

Para este comando utiliza-se a ideia dentro de um bloco pl_sql, no momento onde há a necessidade de saber se é um insert ou update na tabela. No comando MERGE pode-se combinar várias instruções DML como INSERT, UPDATE e DELETE. Abaixo segue sua estrutura e o ganho de desempenho comparando com um insert convencional dentro de um loop FOR.

```
SET SERVEROUTPUT ON -- ligando o spool em tela
DECLARE
  L_START         NUMBER;   -- variavel para medida de tempo
  V_COUNT         NUMBER:=1;  -- contador para loop

  CURSOR CLI IS
  SELECT *
    FROM  CLIENTE
    WHERE ROWNUM < 50000;

BEGIN
  L_START := DBMS_UTILITY.GET_TIME;  -- Funcao para pegar o tempo em minutos e segundos
  FOR X1 IN CLI LOOP
     BEGIN
        SELECT COUNT(1)
          INTO V_COUNT
          FROM CLIENTE_BACKUP  CB
         WHERE CB.CD_CLIENTE  = X1.CD_CLIENTE;
     EXCEPTION
        WHEN OTHERS THEN
            V_COUNT := 0;
     END;

      IF V_COUNT > 0 THEN
        UPDATE CLIENTE_BACKUP CB
           SET CB.ID_SEXO    = X1.ID_SEXO,
               CB.NU_CPF_CGC = X1.NU_CPF_CGC
         WHERE CB.CD_CLIENTE = X1.CD_CLIENTE;
      ELSE
          INSERT
            INTO CLIENTE_BACKUP (CD_CLIENTE, ID_SEXO, NU_CPF_CGC)
          VALUES (X1.CD_CLIENTE, X1.ID_SEXO, X1.NU_CPF_CGC);
      END IF;
  END LOOP;
  DBMS_OUTPUT.PUT_LINE('Tempo Decorrido dentro do loop convencional com FOR e IF: ' || ROUND((DBMS_UTILITY.GET_TIME - L_START)/100,2) || 's');

  ROLLBACK;

  L_START := DBMS_UTILITY.GET_TIME;

  MERGE INTO CLIENTE_BACKUP CB
  USING (SELECT *
           FROM CLIENTE
          WHERE ROWNUM < 50000) CL
             ON (CL.CD_CLIENTE = CB.CD_CLIENTE)
  WHEN  MATCHED THEN
      UPDATE
         SET CB.ID_SEXO    = CL.ID_SEXO,
             CB.NU_CPF_CGC = CL.NU_CPF_CGC
  WHEN NOT MATCHED THEN
      INSERT (CB.CD_CLIENTE, CB.ID_SEXO, CB.NU_CPF_CGC)
      VALUES (CL.CD_CLIENTE, CL.ID_SEXO, CL.NU_CPF_CGC);
  DBMS_OUTPUT.PUT_LINE('Tempo Decorrido usando MERGE: ' || ROUND((DBMS_UTILITY.GET_TIME - L_START)/100,2) || 's');
  ROLLBACK;
END;
```

```
Script Output
Task completed in 68.973 seconds
anonymous block completed
Tempo Decorrido dentro do loop convencional com FOR e IF: 64.8s
Tempo Decorrido usando MERGE: .11s
```

Figura 42. Insert Convencional X MERGE

Observa-se na figura acima, o ganho de desempenho do insert convencional para o MERGE, quando a necessidade de saber se é um INSERT ou UPDATE.

5.6 Tuning em Queries.

5.6.1 Uso do Asterisco em consultas.

Sempre quando se utilizar de busca de dados para uma aplicação, deve-se dimensionar as colunas que serão utilizadas na aplicação. Isso reduz o tempo de busca das informações, logo, se necessário duas colunas para a aplicação em uma tabela que contém trinta colunas, ao trazer somente as duas colunas necessárias, diminui-se o tráfego de dados na rede, na situação cliente x servidor, reduzindo o uso de memória do Oracle no armazenamento deste espaço. Isso pode ser demonstrado simplesmente no plano de execução, conforme vê-se abaixo:

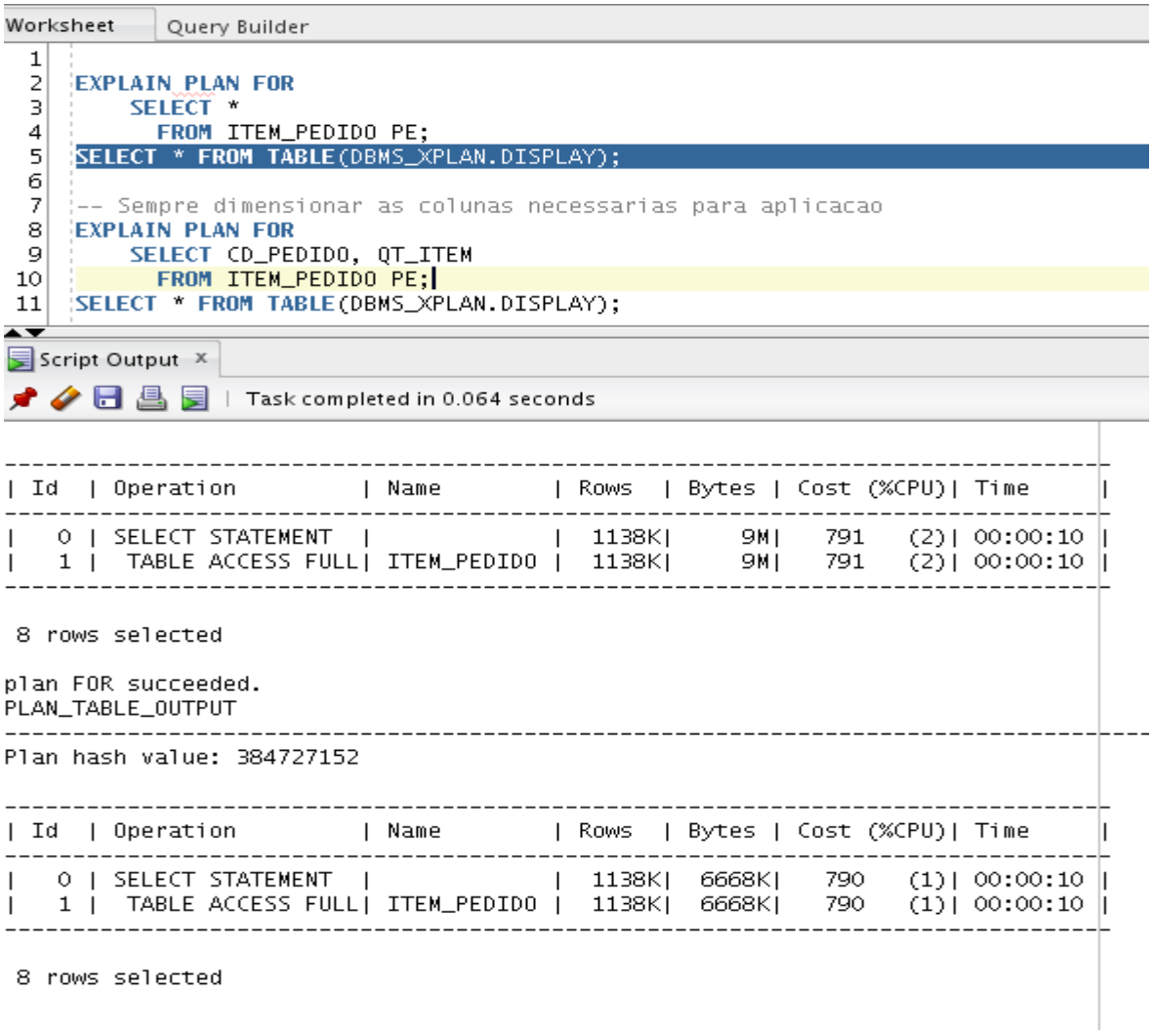


Figura 43. Evitando o uso do Asterisco

Pela figura acima pode-se verificar o plano de execução do uso do asterisco e abaixo o plano de execução da mesma tabela somente dimensionando duas colunas. Nesse caso de 9 M caiu para 6668 K de dados retornados, reduzindo o custo de processamento da CPU.

5.6.2 Queries Idênticas

Sempre que se utilizar uma mesma busca de dados para outras aplicações, deve-se tomar o cuidado de escrevê-las de forma idêntica, pois, quando executar-se no Oracle queries idênticas, na primeira vez o Oracle guarda essa execução na Shared Pool, e quando essa consulta for executada novamente o Oracle pesquisa em sua Shared Pool e vê que esta consulta existe e apenas executa a mesma (repete), tornando o processo mais rápido, pois quando o Oracle encontra a querie na Shared Pool, o otimizador utiliza o método Soft Parse apenas para execução da querie, como definido na figura abaixo. Uma dica para sempre utilizar queries idênticas nas aplicações é o uso de Storage Procedures, fazendo apenas chamadas para retorno dos dados.

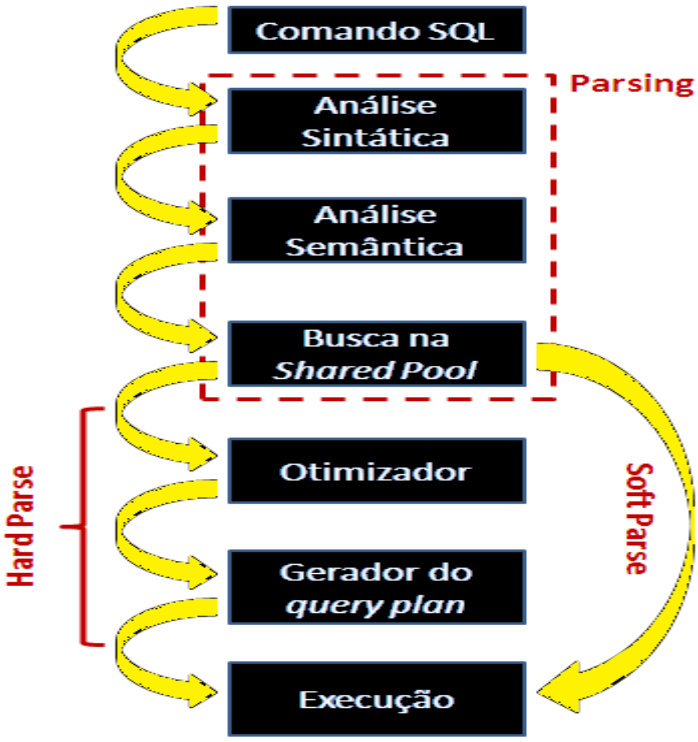


Figura 44. Processamento de SQL no Oracle

5.6.3 Queries Apelidos

Quando programar-se queries deve-se sempre trabalhar com apelidos para as colunas, vale a dica logo que, ao usar apelidos diminuí-se o número de letras nas queries, diminuindo também a alocação de memória na Shared Pool, além de ficar mais clara a query para outros desenvolvedores.

Neste momento, por ser um banco pequeno de testes e não ter concorrência de dados, apenas submetendo-se a queries não há como se medir o ganho de desempenho, mas numa situação real o ganho de performance é grande, além de ser descrito com manual de programação da própria Oracle. Segue abaixo exemplo na figura, com o modo “errado” de programação para o modo com apelidos.

```
24
25
26  SELECT CL.NM_CLIENTE       CLIENTE
27        ,ECOMMERCE.PEDIDO.CD_PEDIDO        NR_PEDIDO
28        ,ECOMMERCE.PEDIDO.DT_PEDIDO        DATA_DO_PEDIDO
29        ,ECOMMERCE.PEDIDO.VL_DESCONTO      VL_DESCONTO
30    FROM ECOMMERCE.PEDIDO
31        ,ECOMMERCE.CLIENTE CL
32   WHERE CL.NM_CLIENTE LIKE 'Vm%'
33     AND CL.CD_CLIENTE = CD_CLIENTE;
34
35
36  -- PROGRAME ASSIM
37  SELECT CL.NM_CLIENTE       CLIENTE
38        ,PE.CD_PEDIDO        NR_PEDIDO
39        ,PE.DT_PEDIDO        DATA_DO_PEDIDO
40        ,PE.VL_DESCONTO      VL_DESCONTO
41    FROM PEDIDO  PE
42        ,CLIENTE CL
43   WHERE CL.NM_CLIENTE LIKE 'Vm%'
44     AND CL.CD_CLIENTE = PE.CD_CLIENTE;
45
```

Figura 45. Queries Apelidos

5.6.4 Regras de Precedência

Como dica, a programação sempre deve ser feita pensando na regra de precedência do Oracle, onde o otimizador sempre vai seguir a precedência abaixo, de cima para baixo, com maior ganho de performance.

- Operador aritmético;
- Operador de concatenação;
- Condições de comparação;
- IS [NOT] NULL, LIKE, [NOT] IN;
- [NOT] BETWEEN;
- Não igual a...
- Negação da condição lógica;
- Condição lógica AND;
- Condição lógica OR.

5.6.5 Operadores Ou e Diferente

Sempre que possível dentro do desenvolvimento substitua os operadores OR, <> ou != por operadores NOT IN ou IN, isso além de deixar o código mais limpo e de fácil entendimento evita erros, conforme se pode verificar:

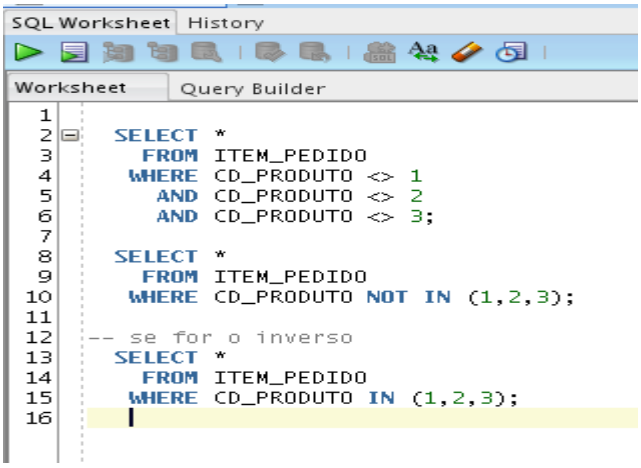


Figura 46. Dica de programação com NOT IN e IN

5.6.6 Operadores de Negação

Ao se desenvolver uma querie que haja a necessidade de excluir a busca valores com NOT IN, <> ou !=, levar em consideração que, nesses casos, o otimizador entende que a busca terá pouca seletividade e não utilizará os índices B-tree, assim, sempre informar na querie os dados que busca retornar com IN. Segue exemplo abaixo:

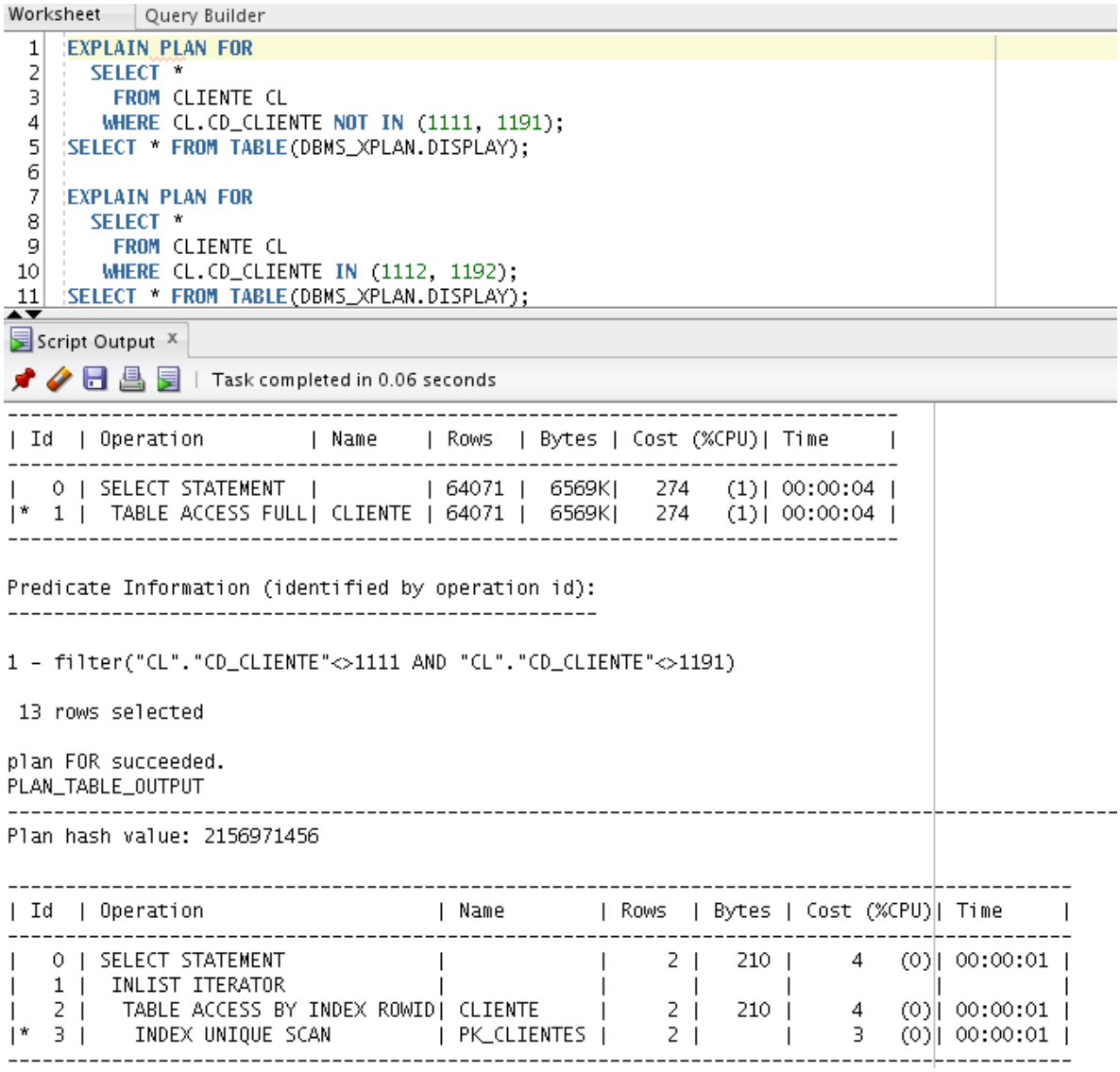


Figura 47. Operadores de negação

5.6.7 Operador Like

Quando houver a necessidade de utilizar o operador LIKE em queries, sempre evitar o uso, se possível, de curingas como (% ou _) no início de comparações, pois quando utilizado o otimizador entende-se que a consulta terá pouca seletividade e não utiliza-se os índices. Como segue exemplo de performance abaixo:

```
 1  EXPLAIN PLAN FOR
 2  SELECT *
 3    FROM PEDIDO  PE
 4        ,CLIENTE CL
 5   WHERE CL.NM_CLIENTE LIKE '%Vm%'
 6     AND CL.CD_CLIENTE = PE.CD_CLIENTE;
 7  SELECT * FROM TABLE (DBMS_XPLAN.DISPLAY);
 8
 9  -- Depois
10
11  EXPLAIN PLAN FOR
12  SELECT *
13    FROM PEDIDO  PE
14        ,CLIENTE CL
15   WHERE CL.NM_CLIENTE LIKE 'Vm%'
16     AND CL.CD_CLIENTE = PE.CD_CLIENTE;
17  SELECT * FROM TABLE (DBMS_XPLAN.DISPLAY);
```

```
-------------------------------------------------------------------------------
| Id  | Operation          | Name    | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |         | 29407 |  3762K|   978   (1)| 00:00:12 |
|*  1 |  HASH JOIN         |         | 29407 |  3762K|   978   (1)| 00:00:12 |
|*  2 |   TABLE ACCESS FULL| CLIENTE |  3204 |   328K|   274   (1)| 00:00:04 |
|   3 |   TABLE ACCESS FULL| PEDIDO  |   592K|    14M|   702   (1)| 00:00:09 |
-------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

1 - access("CL"."CD_CLIENTE"="PE"."CD_CLIENTE")
2 - filter("CL"."NM_CLIENTE" LIKE '%Vm%')

 16 rows selected

plan FOR succeeded.
PLAN_TABLE_OUTPUT
-------------------------------------------------------------------------------------------------------------
Plan hash value: 881954350

-------------------------------------------------------------------------------------------------------------
| Id  | Operation                    | Name                | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                     |   101 | 13231 |   717   (1)| 00:00:09 |
|*  1 |  HASH JOIN                   |                     |   101 | 13231 |   717   (1)| 00:00:09 |
|   2 |   TABLE ACCESS BY INDEX ROWID| CLIENTE             |    11 |  1155 |    13   (0)| 00:00:01 |
|*  3 |    INDEX RANGE SCAN          | UK_CLIENTE_NMCLIENTE |    11 |       |     2   (0)| 00:00:01 |
|   4 |   TABLE ACCESS FULL          | PEDIDO              |   592K|    14M|   702   (1)| 00:00:09 |
-------------------------------------------------------------------------------------------------------------
```

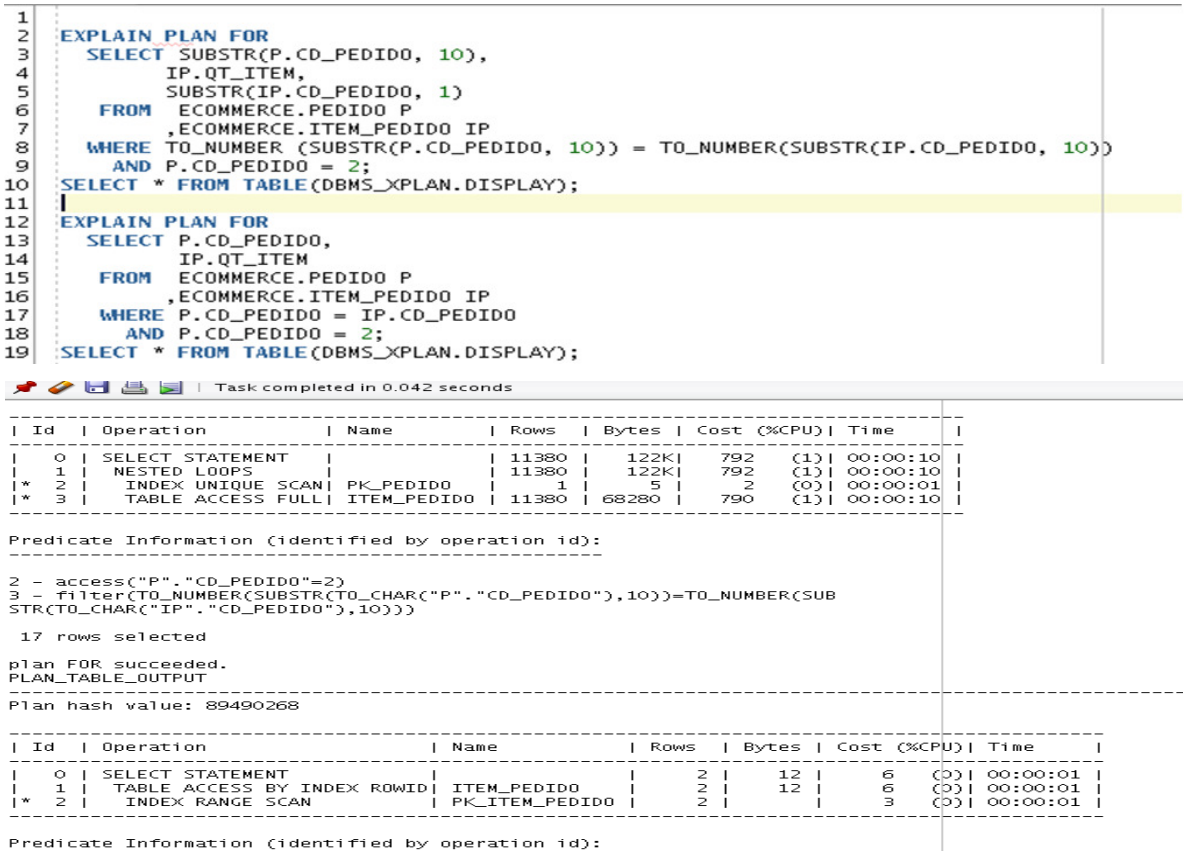
Figura 48. Evitando uso de Curingas no LIKE

5.6.8 Índices nas Foreing Key

Se uma query for utilizada com bastante frequência na aplicação, deve-se criar índices para otimização de suas consultas, uma dica que vale lembrar, de sempre criar índices em colunas FK's, o que otimiza a performance nas ligações entre JOINS.

5.6.9 Evite o uso de funções entre ligações de tabelas.

Deve-se evitar ao máximo o uso de funções de conversões de dados em ligações de JOIN, em tabelas, tal fator aumenta o uso da CPU e consequentemente diminui a performance de processamento da query. Na documentação Oracle, diz-se que o uso de funções SQL entre ligações de tabelas eliminam o uso do índice B-tree. Segue plano de execução abaixo para comprovação do fato:



```
1
2  EXPLAIN PLAN FOR
3    SELECT SUBSTR(P.CD_PEDIDO, 10),
4           IP.QT_ITEM,
5           SUBSTR(IP.CD_PEDIDO, 1)
6     FROM  ECOMMERCE.PEDIDO P
7          ,ECOMMERCE.ITEM_PEDIDO IP
8    WHERE TO_NUMBER (SUBSTR(P.CD_PEDIDO, 10)) = TO_NUMBER(SUBSTR(IP.CD_PEDIDO, 10))
9      AND P.CD_PEDIDO = 2;
10 SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);
11
12 EXPLAIN PLAN FOR
13   SELECT P.CD_PEDIDO,
14          IP.QT_ITEM
15    FROM  ECOMMERCE.PEDIDO P
16         ,ECOMMERCE.ITEM_PEDIDO IP
17    WHERE P.CD_PEDIDO = IP.CD_PEDIDO
18      AND P.CD_PEDIDO = 2;
19 SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);
```

Task completed in 0.042 seconds

```
-----------------------------------------------------------------------------------------
| Id  | Operation          | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |              | 11380 |   122K|   792   (1)| 00:00:10 |
|   1 |  NESTED LOOPS      |              | 11380 |   122K|   792   (1)| 00:00:10 |
|*  2 |   INDEX UNIQUE SCAN| PK_PEDIDO    |     1 |     5 |     2   (0)| 00:00:01 |
|*  3 |   TABLE ACCESS FULL| ITEM_PEDIDO  | 11380 | 68280 |   790   (1)| 00:00:10 |
-----------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

2 - access("P"."CD_PEDIDO"=2)
3 - filter(TO_NUMBER(SUBSTR(TO_CHAR("P"."CD_PEDIDO"),10))=TO_NUMBER(SUB
STR(TO_CHAR("IP"."CD_PEDIDO"),10)))

 17 rows selected

plan FOR succeeded.
PLAN_TABLE_OUTPUT
-----------------------------------------------------------------------------------------
Plan hash value: 89490268

-----------------------------------------------------------------------------------------
| Id  | Operation                    | Name            | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                 |     2 |    12 |     6   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| ITEM_PEDIDO     |     2 |    12 |     6   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN           | PK_ITEM_PEDIDO |     2 |       |     3   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
```

Figura 49. Comparação com funções na ligação entre tabelas

Pode-se verificar pela figura acima que a query com a utilização de funções para junções entre tabelas não utiliza o índice existente no campo CD_PEDIDO para retornar os dados, fazendo um table acess full, logo na query abaixo, o otimizador utiliza o índice fazendo um index range scan, como pode-se reparar, o ganho de performance foi grande de 10 s para 01 s, com custo de processamento de 792 para 6.

5.6.10 Conversões de dados em ligações como índices baseado em Funções.

Índices de datas podem ser de tipos baseados em funções para conversão de datas, sempre que se fizer uma query de consulta com JOINS entre colunas que possuem índices baseados em funções deve-se ter cuidado para a inserção correta da conversão, com isso utiliza-se o índice e ganha na performance de aplicação. Segue exemplo abaixo com seu plano de execução:

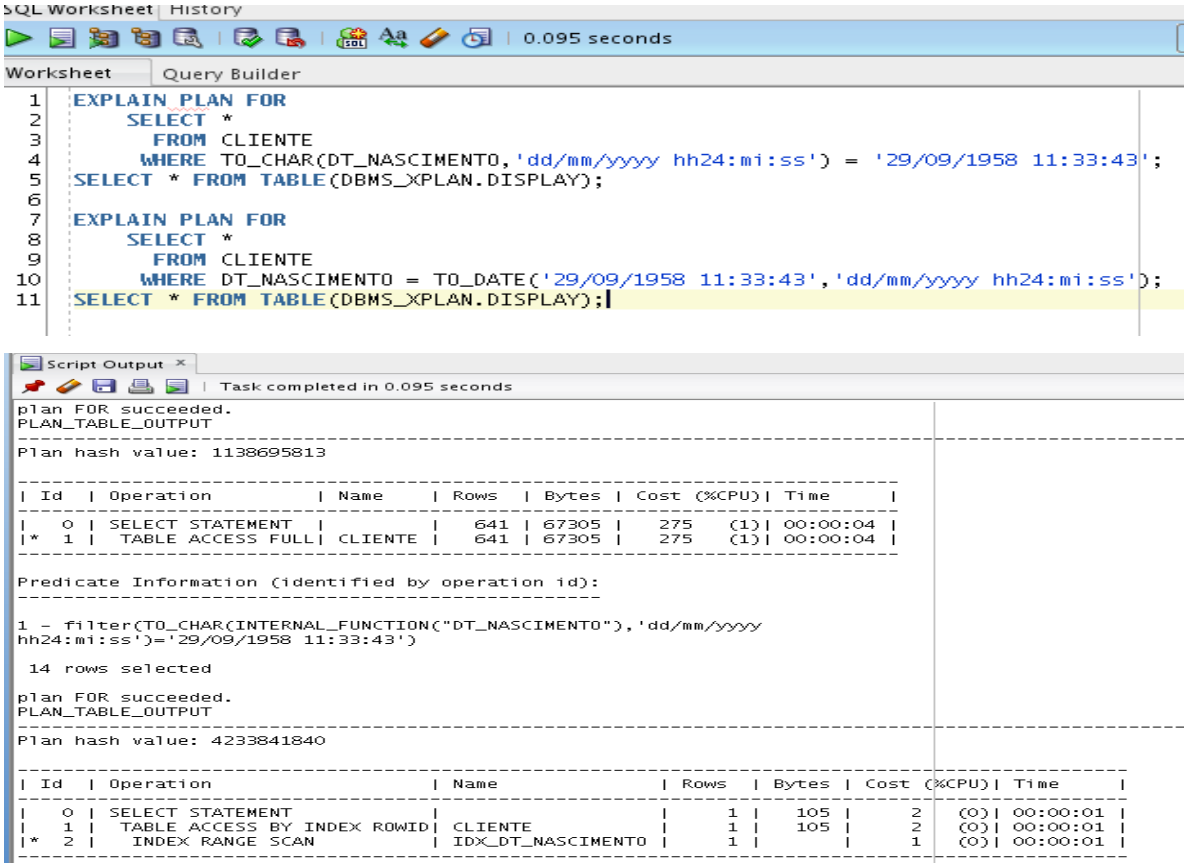


Figura 50. Plano de Execução entre JOINS com índice baseado em Funções

No exemplo acima, observa-se o ganho de performance usando a conversão correta to_date, para a consulta, logo quando utilizou-se a query com parâmetro fixo, o otimizador entendeu a necessidade de fazer um full table scan na tabela ignorando o índice já existente, quando insere-se a função to_date, afim de utilizar o índice baseado em função, o otimizador já fez index range scan, otimizando a performance de 04 s para 01 s, diminuindo o custo de 275 para 105.

5.6.11 Utilização do NULL para comparações

Deve-se evitar ao máximo comparações no WHERE com NULL, logo quando isso ocorre o otimizador não utiliza o índice para pesquisas, resultando numa perca de performance. Isto ocorre por que o otimizador do Oracle não reconhece NULL como índice, e com isso não gera as estatísticas da tabela. Segue abaixo plano de otimização:

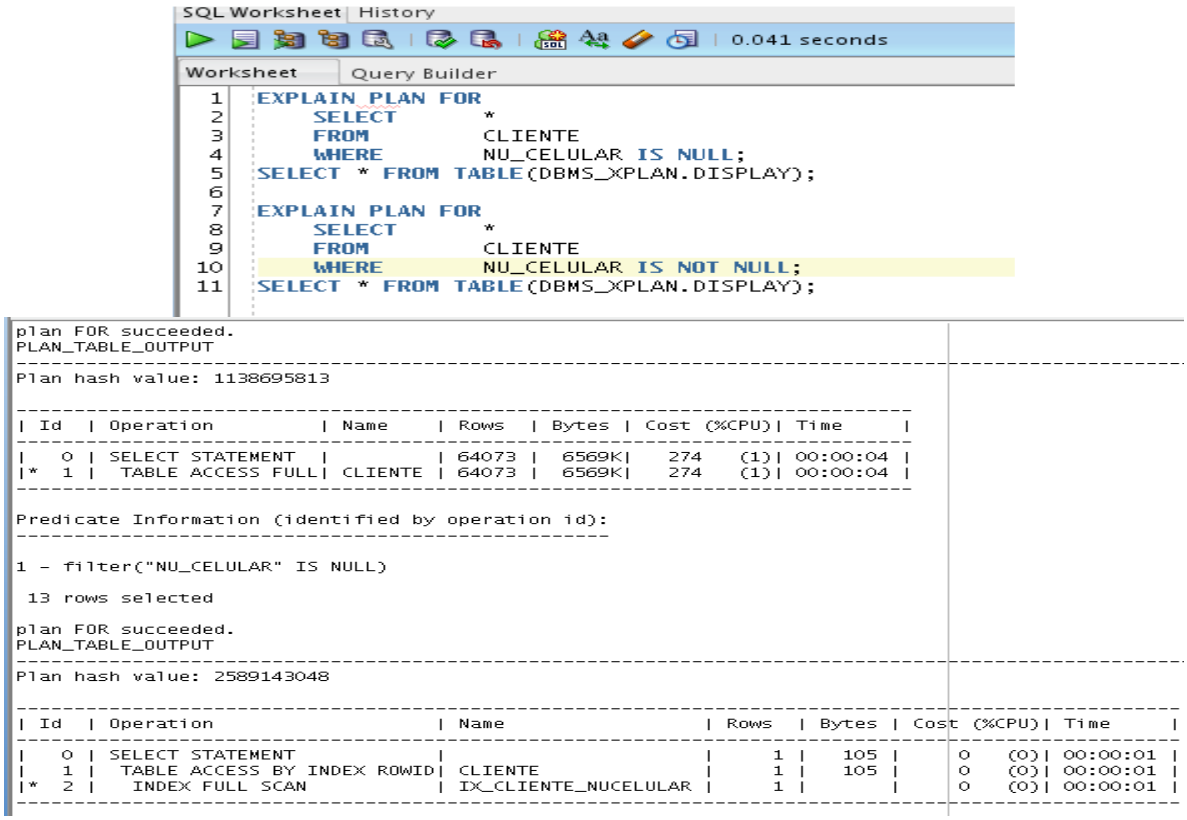


Figura 51. Comparação NULL e NOT NULL

Quando utilizado o NULL, como visto acima o otimizador fez full table scan na tabela, resultando em um custo de 274 com tempo de 04 s em uma tabela pequena, quando maior a tabela maior o tempo de retorno, menor a performance. Quando utilizado o NOT NULL, o otimizador já utilizou o índice B-Tree resultando em um custo 0 com tempo de 01s.

5.6.12 UNION e UNION ALL

Em certas ocasiões, query´s complexas podem ser resolvidas e melhor compreendidas utilizando UNION ou UNION ALL, que fazem a função de junção de resultados de duas instruções.

A diferença entre ambos é que a instrução UNION trabalha retirando as linhas repetidas e comparando-as, com isso reduz o desempenho, utilizando de um DISTINCT.

Nesses casos, sempre que possível, substitua o uso do UNION pelo UNION ALL. Segue plano de execução abaixo de duas instruções, onde a primeira foi utilizada o UNION e a próxima foi utilizada UNION ALL.

```
-------------------------------------------------------------------------------------------
| Id  | Operation             | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |                |     2 |    15 |  1408  (51)| 00:00:17 |
|   1 |  SORT UNIQUE          |                |     2 |    15 |  1408  (51)| 00:00:17 |
|   2 |   UNION-ALL           |                |       |       |            |          |
|   3 |    SORT AGGREGATE     |                |     1 |     2 |   703   (1)| 00:00:09 |
|*  4 |     TABLE ACCESS FULL | PEDIDO         |     1 |     2 |   702   (1)| 00:00:09 |
|   5 |    SORT AGGREGATE     |                |     1 |    13 |   704   (1)| 00:00:09 |
|   6 |     NESTED LOOPS      |                |     1 |    13 |   703   (1)| 00:00:09 |
|*  7 |      TABLE ACCESS FULL| PEDIDO         |     1 |     7 |   702   (1)| 00:00:09 |
|*  8 |      INDEX UNIQUE SCAN| PK_ITEM_PEDIDO |     1 |     6 |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------
| Id  | Operation             | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |                |     2 |    15 |  1406  (51)| 00:00:17 |
|   1 |  UNION-ALL            |                |       |       |            |          |
|   2 |   SORT AGGREGATE      |                |     1 |     2 |            |          |
|*  3 |    TABLE ACCESS FULL  | PEDIDO         |     1 |     2 |   702   (1)| 00:00:09 |
|   4 |   SORT AGGREGATE      |                |     1 |    13 |            |          |
|   5 |    NESTED LOOPS       |                |     1 |    13 |   703   (1)| 00:00:09 |
|*  6 |     TABLE ACCESS FULL | PEDIDO         |     1 |     7 |   702   (1)| 00:00:09 |
|*  7 |     INDEX UNIQUE SCAN | PK_ITEM_PEDIDO |     1 |     6 |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
```

Figura 52. UNION X UNION ALL

Nota-se o ganho de performance com a utilização do UNION ALL, em menos parses no plano de execução, isso deve-se ao fato de que o UNION ALL não utiliza do DISTINCT, trazendo todas as linhas.

5.6.13 Utilizando CASE para busca de dados.

A utilização da cláusula CASE permite a montagem de uma estrutura similar a IF..THEN..ELSE.

O ganho de desempenho com CASE se deve ao fato de buscar as informações no banco somente um vez, ao contrário de um conjunto de selects para trazer valores.

Para demonstrar o ganho foi construído o script abaixo, onde se busca que a aplicação receba os valores de quantos funcionários estão recebendo abaixo de R$ 2.000,00 (dois mil reais), quantos estão ganhando entre R$ 2.000,00 (dois mil reais) e R$ 4.000,00 (quatro mil reais) e quantos estão ganhando mais que R$ 4.000,00 (quatro mil reais), neste caso pode ser feito de duas formas por select com UNION ALL ou para ganho de desempenho com a utilização do CASE como observa-se abaixo:

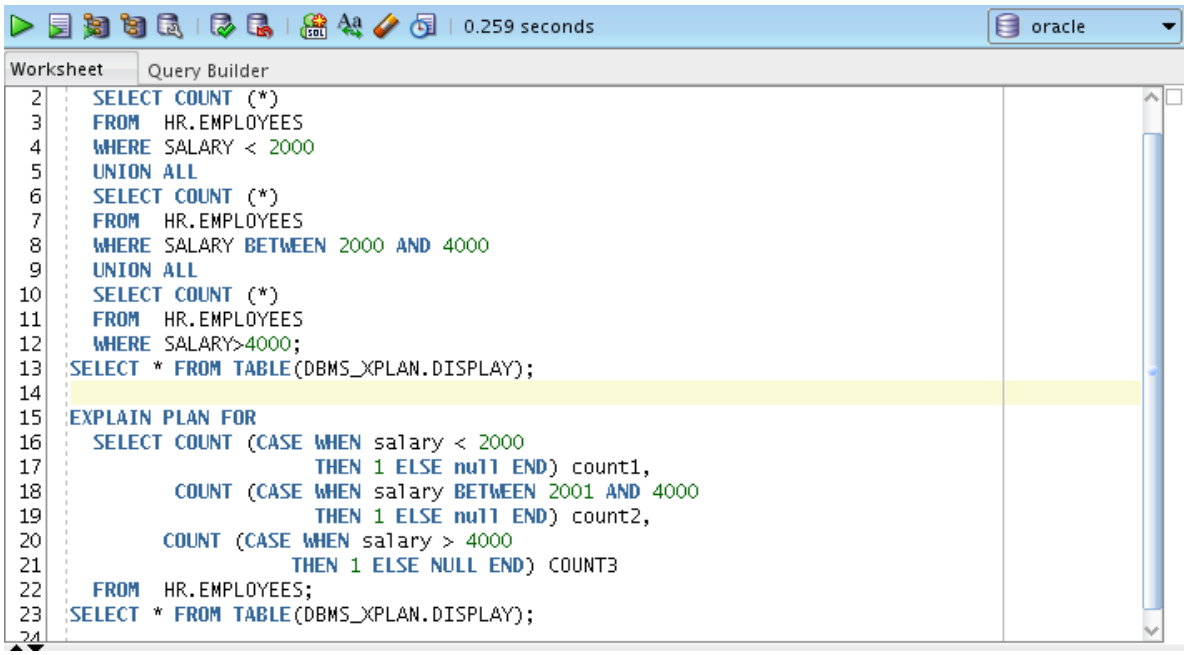


Figura 53. Utilizando CASE

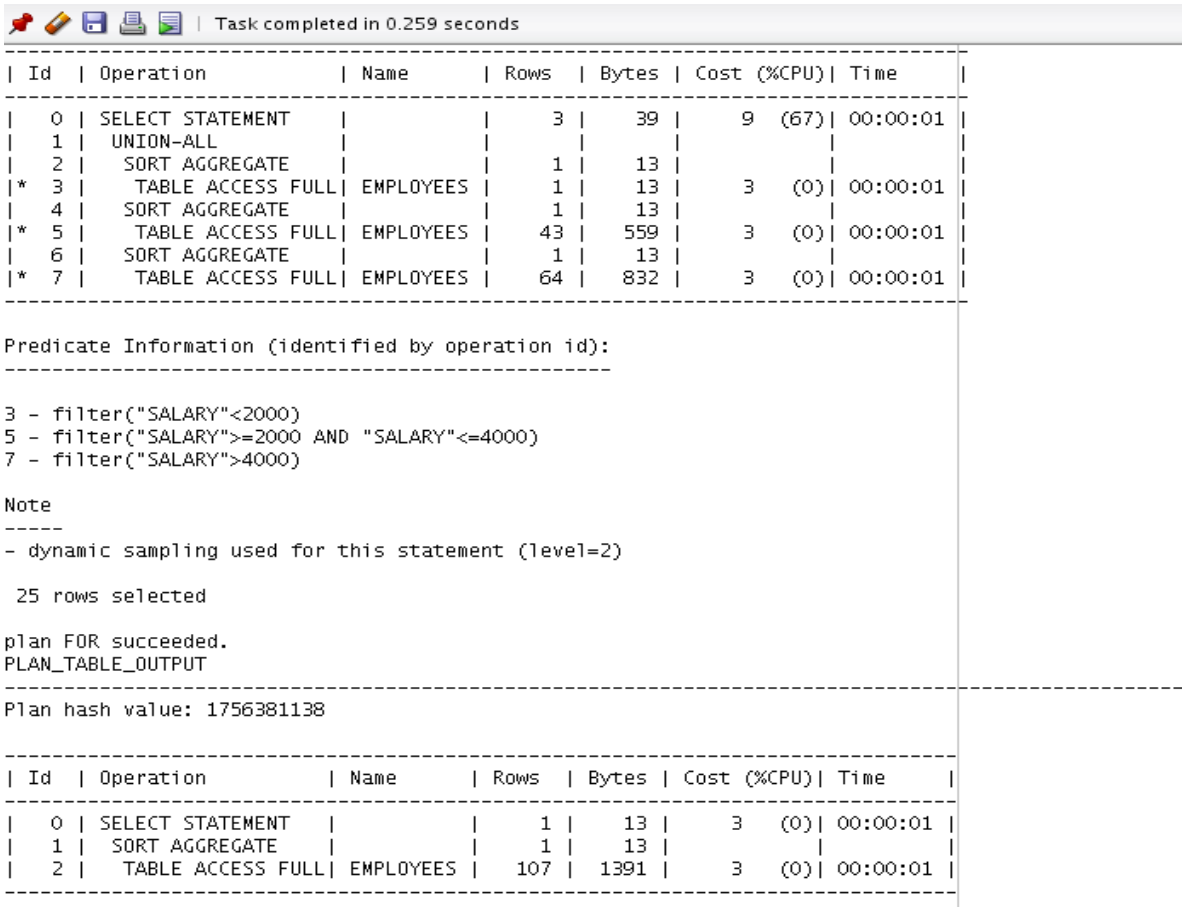


Figura 54. Desempenho do CASE X UNION ALL

De acordo com a figura acima, pode-se observar o ganho de desempenho logo que o CASE somente buscou uma única vez os dados na tabela EMPLOYEES, com a utilização do SELECT com UNION ALL foram necessárias três buscas de TABLE ACESS FULL. Com isso há uma redução no custo da CPU e logo de tempo.

5.6.14 Utilizando a Cláusula WITH para criação de tabela temporária

Muitas vezes há a situação de já estar com a query escrita, a fim de evitar a sua reescrita novamente para utilização do CASE, pode-se utilizar a cláusula WITH em algumas situações, com isso realiza-se apenas uma consulta na tabela, a cláusula WITH foi implantada no Oracle versão 9, e tem a função de criar uma tabela temporária, otimizando queries que precisam referenciar várias vezes a mesma

tabela. Segue exemplo abaixo de plano de execução para ganho de desempenho, neste caso é necessário apenas saber apenas quantos funcionários ganham menos que R$ 2.000,00 (dois mil reais) e quantos estão ganhando entre R$ 2.000,00 (dois mil reais) e R$ 4.000,00 (quatro mil reais).

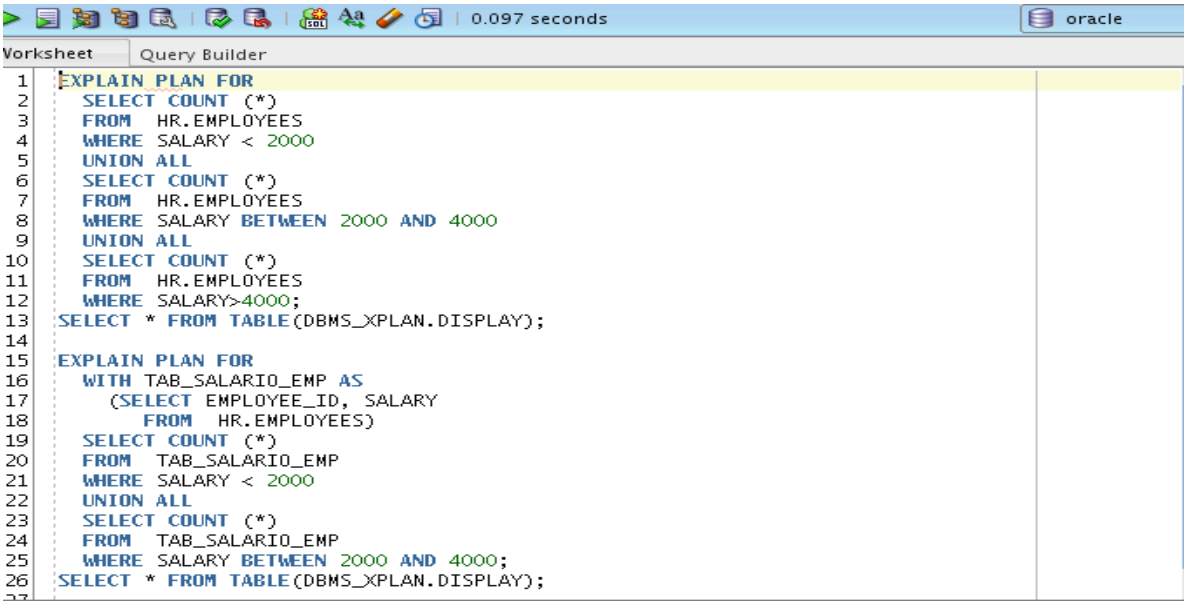


Figura 55. Utilizando a Clausula WITH

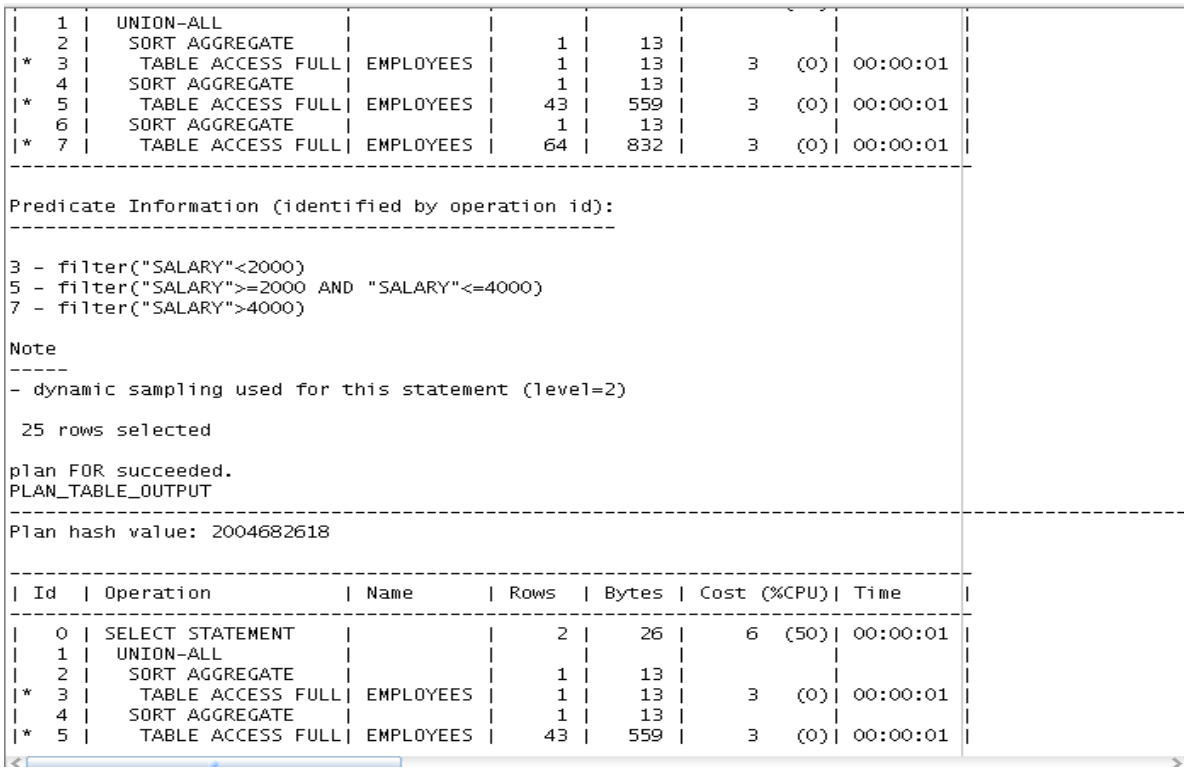


Figura 56. Comparação com a Cláusula WITH

Tendo em vista que a presente aplicação somente precisa de duas situações no WHERE, a Cláusula WITH cria uma tabela temporária somente do que precisa retornar, sem precisar ir na tabela EMPLOYEES fazer TABLE SCAN FULL, com isso há ganho de desempenho na consulta de menos linhas.

5.6.15 Utilização de Variáveis BIND

Sempre trabalhar com variáveis nas condições WHERE dos SELECTS que são reutilizados na aplicação, com isso o otimizador identifica que a SELECT está na Shared Pool, e com isso utiliza o plano de execução já existente, tendo um desempenho maior. Segue exemplo de ganho de desempenho abaixo:

```
SET SERVEROUTPUT ON
DECLARE
   V_START      NUMBER;
   V_COUNT      NUMBER;
   V_MANAGER_ID NUMBER;
BEGIN
   V_START := DBMS_UTILITY.GET_TIME;

   SELECT    COUNT(1) INTO V_COUNT
   FROM      HR.EMPLOYEES
   WHERE     MANAGER_ID = 100;

   SELECT    COUNT(1) INTO V_COUNT
   FROM      HR.EMPLOYEES
   WHERE     MANAGER_ID = 120;

   SELECT    COUNT(1) INTO V_COUNT
   FROM      HR.EMPLOYEES
   WHERE     MANAGER_ID = 121;

   SELECT    COUNT(1) INTO V_COUNT
   FROM      HR.EMPLOYEES
   WHERE     MANAGER_ID = 123;

   DBMS_OUTPUT.PUT_LINE('Tempo de execucao SQL sem BIND: ' || (DBMS_UTILITY.GET_TIME - V_START) || 'tempo em segundos de processamento.');

   V_START := DBMS_UTILITY.GET_TIME;
   V_MANAGER_ID:=100;
   SELECT    COUNT(1) INTO V_COUNT
   FROM      HR.EMPLOYEES
   WHERE     MANAGER_ID = V_MANAGER_ID;

   V_MANAGER_ID:=120;
   SELECT    COUNT(1) INTO V_COUNT
   FROM      HR.EMPLOYEES
   WHERE     MANAGER_ID = V_MANAGER_ID;

   V_MANAGER_ID:=121;
   SELECT    COUNT(1) INTO V_COUNT
   FROM      HR.EMPLOYEES
   WHERE     MANAGER_ID = V_MANAGER_ID;

   V_MANAGER_ID:=123;
   SELECT    COUNT(1) INTO V_COUNT
   FROM      HR.EMPLOYEES
   WHERE     MANAGER_ID = V_MANAGER_ID;

   DBMS_OUTPUT.PUT_LINE('Tempo de execucao SQL com BIND: ' || (DBMS_UTILITY.GET_TIME - V_START) || 'tempo em segundos de processamento.');
END;
anonymous block completed
Tempo de execucao SQL sem BIND: 280tempo em segundos de processamento.
Tempo de execucao SQL com BIND: 268tempo em segundos de processamento.
```

Figura 57. Tempo de Desempenho com variável BIND

Nota-se o ganho de 268 segundos com BIND para 280 com variáveis fixas, isso se deve ao fato comentado acima.

5.7 Utilização de HINT.

De acordo com Lewis (2005):

> Apesar de todas as informações coletadas e todo algoritmo sofisticado do otimizador, provavelmente o nosso conhecimento da informação armazenada é maior que a que o otimizador possui. Por exemplo, podemos saber que um índice é ais seletivo em certas queries que o otimizador possa determinar. Baseado nisso podemos estar mais apto a escolher um plano de execução mais eficiente que o otimizador. Neste caso, podemos usar HINTS para forçar o otimizador a usar o plano de execução que desejamos. HINTS são sugestões que fornecemos ao otimizador para um comando SQL. (LEWIS, 2005)

Nesse tópico tratar-se-á dos HINTS mais conceituados, com maior uso por programadores.

A versão 11G é mais eficaz com relação a este assunto; em testes realizados somente dois HINTS obtiveram ganho de desempenho (FIRST-ROW, FULL), os demais apresentaram o mesmo plano de execução, isso se dá pelo fato de que o Banco de Dados Oracle 11G está mais preparado para estas situações, escolhendo os melhores plano de execuções.

5.7.1 HINT FULL SCAN

Este HINT FULL (/*+FULL*/) tem a função de realizar um full table scan na tabela, deve ser utilizado quando houver a certeza que um FTS terá o melhor resultado do que um index scan.

5.7.2 HINT FULL COM PARALLEL

Conforme testado no início do capitulo 5, este HINT (*/*+ PARALLEL (p,2,2) */*) é utilizado quando se busca que a aplicação seja executada por mais de 1 processador. Deve-se conhecer a arquitetura dos servidores para poder dimensionar com quantos CPU uma query será executada, para processamento distribuído.

5.7.3 HINT FIRST_ROWS

Trata-se de uma hint muito utilizada (/* first_rows (5) */); deve ser utilizada pela busca de linhas determinadas, mas não para leitura completa da tabela.

5.7.4 HINT RULE

Este HINT (/*+ RULE */) considera as regras que o Oracle possui pré configuradas para execução da expressão, desconsiderando qualquer outra hint.

5.7.5 HINT FULL

O HINT FULL (/*+ FUL (p) */) realiza uma varredura completa da tabela, a qual deve ser realizada com cautela, e assim poderá contribuir com resultados positivos ou negativos.

5.7.6 HINT LEADING

Este HINT (/* LEADING (p) */) concorre diretamente com a “ORDERED”, prevalecendo esta. Utiliza-se a LEADING quando não há aptidão em determinar a

ordem ideal das tabelas para varredura, assim determina-se a tabela especificada e a hint realizará a sequência ideal para as tabelas seguintes envolvidas.

5.7.7 HINT USE_NL

Este hint (/* USE_NL (p) */) é mais rápido para retorno de uma linha, mas é lento para conjunto de linhas.

5.7.8 HINT USE_HASH

Este hint (/* USE_HASH(infe) */) tem como função principal subir para a memória a segunda tabela que será percorrida a fim de achar as combinações que atendem a linha selecionada na primeira tabela, com desempenho satisfatório quando há espaço considerável nas áreas de HASH_AREA_SIZE e PGA_AGGREGATE_TARGET.

5.8 Ferramentas para analise de performance e desempenho do banco.

5.8.1 Explain Plan.

O Explain Plan irá demonstrar ao desenvolvedor, através do plano de execução, os meios de acesso que o otimizador está utilizando para acessar as tabelas do banco de dados.

Como pode-se notar, esse foi o meio escolhido para demonstrar os ganhos de desempenho no tuning.

Segue abaixo o comando a ser inserido para gerar o plano de execução:

```
EXPLAIN PLAN FOR
{INSTRUÇÃO SELECT A SER AVALIADA}
```

SELECT * FROM TABLE (dbms_xplan.display)

Importante lembrar que, para ter uma bom resultado no plano de execução deve-se ter sempre os dados estatísticos atualizados das tabelas, uma vez que o plano de execução trabalha em cima de seletividade, cardinalidade e custo.

5.8.2 SQL LOADER

De acordo com VIEGAS (2012):

> O SQL*Loader é um aplicativo de importação que acompanha o Oracle, tendo como diferencial a flexibilidade na configuração, pois utiliza uma linguagem de scripts para importar os dados. Entre os recursos disponibilizados pela linguagem de script, é possível validar cada registro antes da importação, definir valores default, aplicar alterações sobre os registros lidos, separar os dados importados em duas ou mais tabelas, entre outros. O SQL Loader também permite a importação de tipos BLOB, objetos complexos e coleções de dados. O utilitário foi projetado para importação de arquivos texto que tenham formato de tabela, onde cada registro é representado por uma linha, os campos podem ter tamanho fixo ou variável e qualquer caráter pode ser utilizado para separá-los. O script de importação deve ser salvo em um arquivo texto, conhecido como arquivo de controle (Control File). O SQL*Loader é um aplicativo de linha de comando, que interpreta o arquivo de controle passado como parâmetro. A linguagem utilizada no script é específica, denominada SQL*Loader´s DDL (Data Definition Language). A documentação completa desta linguagem está disponível na instalação do Oracle.

Esta ferramenta foi utilizada nesse trabalho para dar carga em algumas das tabelas utilizadas conforme descrita no subitem 5.6.5.

5.8.3 TKPROF

De acordo com Ricardo (2009):

> Essa é uma ferramenta cuja função principal é de transformar um “arquivo de trace” em um arquivo de texto, para que assim possa permitir a nossa avaliação quanto ao desempenho das instruções de consultas abertas ou fechadas como os procedimentos armazenados (storeds procedures) ou os pacotes (packages).

O retorno dos dados convertidos, para o arquivo do tipo texto, deve ser realizado linha a linha, de maneira cuidadosa para que não passe qualquer detalhe que possa representar o verdadeiro motivo da baixa performance. O arquivo, que é denominado tecnicamente como “arquivo de trace”, normalmente está localizado nos diretórios: UDUMP, CDUMP e BDUMP e armazena todas as operações que ocorrerem dentro do banco de dados (RICARDO, 2009).

Ricardo (2009) indica ainda certos pontos que devem ser observados nos arquivos gerados pelo TKPROF:

> Comparar o número de “Parses” com o número de execuções, onde o ideal é um Parse para várias execuções. Aproveitamos esse momento para especificar que as estatísticas são divididas basicamente em 3 grupos: Parse – é o momento onde verifica a existência dos objetos citados na instrução assim como se a sintaxe está correta; Execute – é a etapa onde ocorre verdadeiramente a execução do SQL pelo Oracle; e Fetch – onde ocorre o retorno das linhas que satisfazem a instrução SQL, portanto é válida somente para select´s,
>
> • Procurar aplicar as variáveis BIND em instruções que não estejam utilizando-as, isto, por que essas variáveis permitem um ganho de desempenho considerável no momento da execução das consultas (Execute), pois permite que a instrução seja preparada uma única vez pelo otimizador do gerenciador de banco de dados e consequentemente executada inúmeras vezes;
>
> • Localizar os processos que estejam constantemente fazendo uso de Full Table Scans (FTS), pois, por consequência esses processos, como já foram

citados anteriormente, consomem muita CPU além de um alto custo de leituras a disco.

Abaixo segue os prints de geração de um arquivo TKPROF para análise:

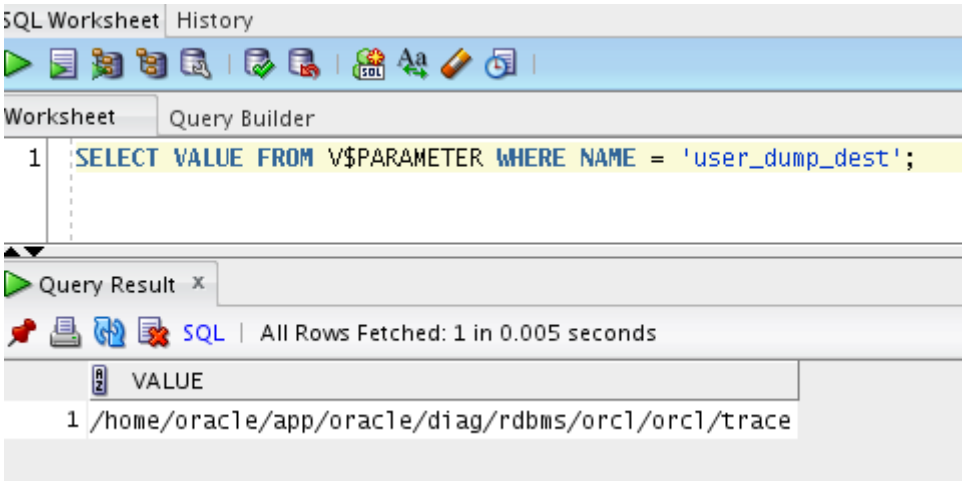


Figura 58. Local onde será gerado os arquivos do TKPROF

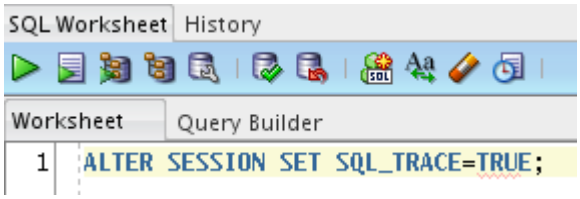


Figura 59. Habilitando o trace no ambiente

Quando habilitado o trace no ambiente deve-se ter cuidado, pois isso acarretará numa perda de desempenho, logo que serão gravados em arquivo todos os passos das instruções executadas.

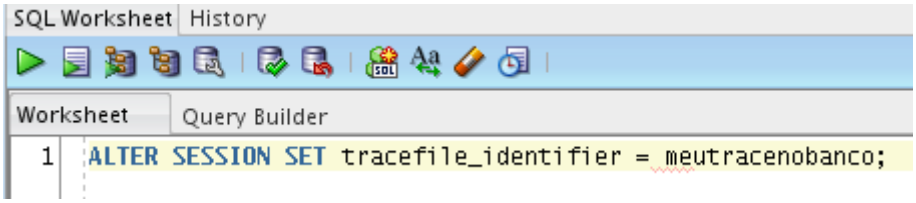


Figura 58. Nomeando o trace no ambiente.

Nomeia-se o trace a ser gerado como forma de facilitar na busca do arquivo gerado.

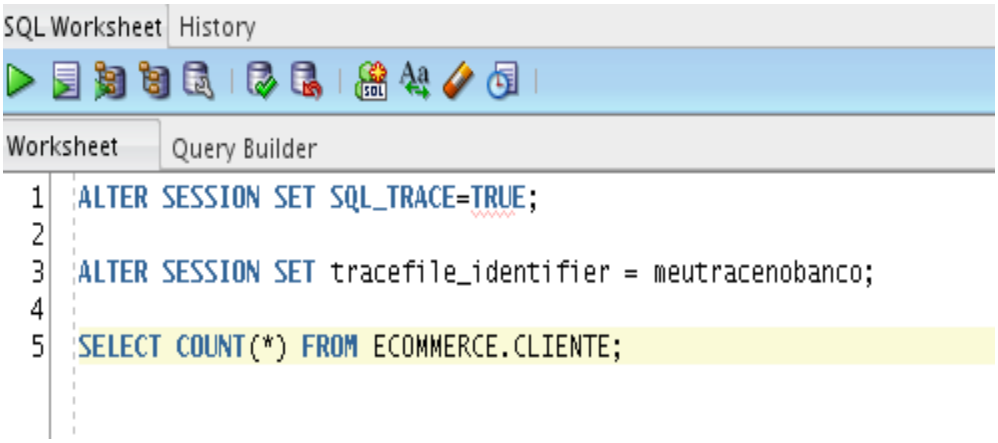


Figura 60. Instrução realizada no trace

No terminal do Linux ou no Dos do Windows, deve-se ir no caminho onde foi gravado os arquivos do TKPROF.

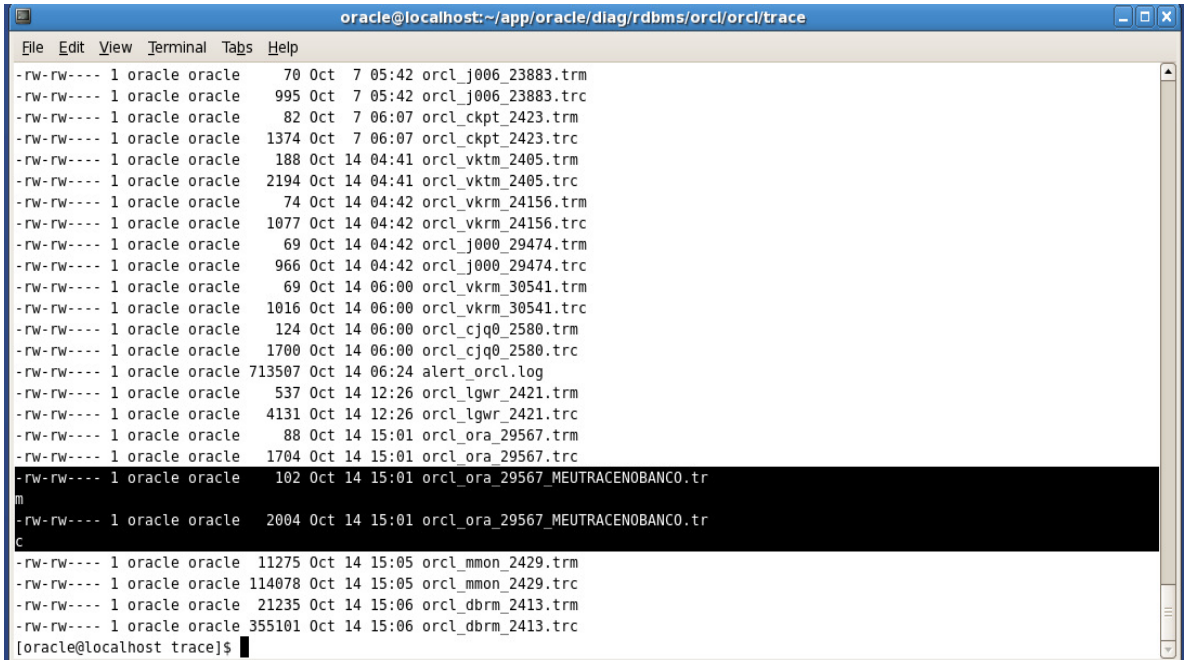


Figura 61. Localizando os arquivos gerados pelo TKPROF

```
[oracle@localhost trace]$ TKPROF orcl_ora_29567_MEUTRACENOBANCO.trc resultado.txt
bash: TKPROF: command not found
[oracle@localhost trace]$ tkprof orcl_ora_29567_MEUTRACENOBANCO.trc resultado.txt

TKPROF: Release 11.2.0.2.0 - Development on Sun Oct 14 15:08:58 2012

Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All rights reserved.


[oracle@localhost trace]$
```

Figura 62. Gerando arquivo .txt do trace gerado pelo TKPROF

```
[oracle@localhost trace]$ vi resultado.txt
\[oracle@localhost trace]$
```

Figura 63. Lendo o arquivo gerado

```
SELECT COUNT(*)
FROM
 ECOMMERCE.CLIENTE


call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        1      0.00       0.00          0        140          0           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        3      0.00       0.00          0        140          0           1

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 240
Number of plan statistics captured: 1
```

Figura 64. Parte do arquivo gerado onde encontra-se o trace da instrução executada a ser analisada

Assim finaliza-se os passos para geração do arquivo TKPROF e a leitura do mesmo. Por se tratar de uma instrução SELECT analisada, foram executados 3 passos para consulta do mesmo (PARSE, EXECUTE e FETCH).

5.9 Parâmetros de Banco de dados Oracle 11 G

Finalizando o TCC, verifica-se parâmetros em nível de sessão de usuário que setados corretamente geram maior ganho de desempenho em instruções SQL.

5.9.1 Parâmetro Cursor_Sharing

Trata-se de um parâmetro do banco de dados que afeta o nível de interpretação das sentenças SQL que são comuns entre os usuários. Quando se tem SQL idênticos no banco de dados, este é gravado na SHARED POOL, com isso na sua próxima

execução o otimizador realize a busca se o mesmo já existe em um plano de execução na SHARED POOL, se sim o otimizador o utiliza.

No Cursor_Sharing existe dois valores suportados na versão 11 G, que é o EXACT, padrão este documentado pela Oracle para aplicativos ERP – Enterprise Business Suite, onde respeita o que esta escrito na SQL, não levando em consideração as sentenças comuns, e o outro valor aceito é o FORCE, quando setado no banco de dados o otimizador levará em consideração os SQL fixo como variáveis BIND, com isso para bancos OLTP, o desempenho será maior.

Para alterar o parâmetro que está setado no banco utilize a instrução abaixo, normalmente a instalação do Banco de Dados Oracle vem com o parâmetro setado para EXACT.

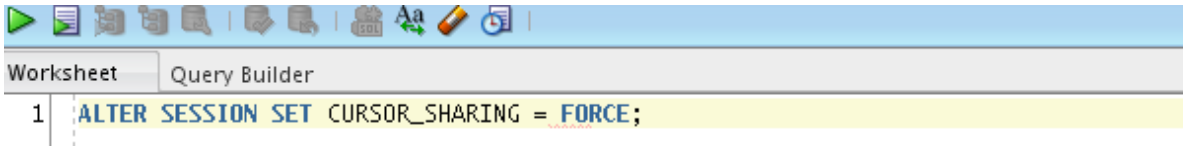


Figura 65. Alterado o parâmetro CURSOR_SHARING

Quando setado este parâmetro no banco e o desenvolvedor desejar trabalhar com o parâmetro na aplicação com EXACT, o mesmo pode utilizar do HINT /*+ CURSOR_SHARING_EXACT */.

5.9.2 Parâmetro Optimizer_Index_Cost_Adj

Este parâmetro permite mudar os custos de operações FULL TABLE SCAN X INDEX SEARCH. Para este parâmetro, os valores permitidos são entre 0 e 10.000. De default, o banco de dados Oracle vem com parâmetro setado com valor 100. Porém para muitos bancos de dados Oracle, o padrão entre 10 e 50 são os ideais para ganho de desempenho, pois nessa faixa o banco utiliza os índices com maior frequência, se o banco estiver recolhendo as estatísticas corretamente, e ter o cuidado para criação somente de índices necessários, o valor entre 10 e 50 é o mais indicado.

Para saber o valor correto a ser configurado nesse parâmetro para ganho de performance, existe a query abaixo que mostra no campo STARTING_VALUE_OICA, o valor mais indicado para este parâmetro:

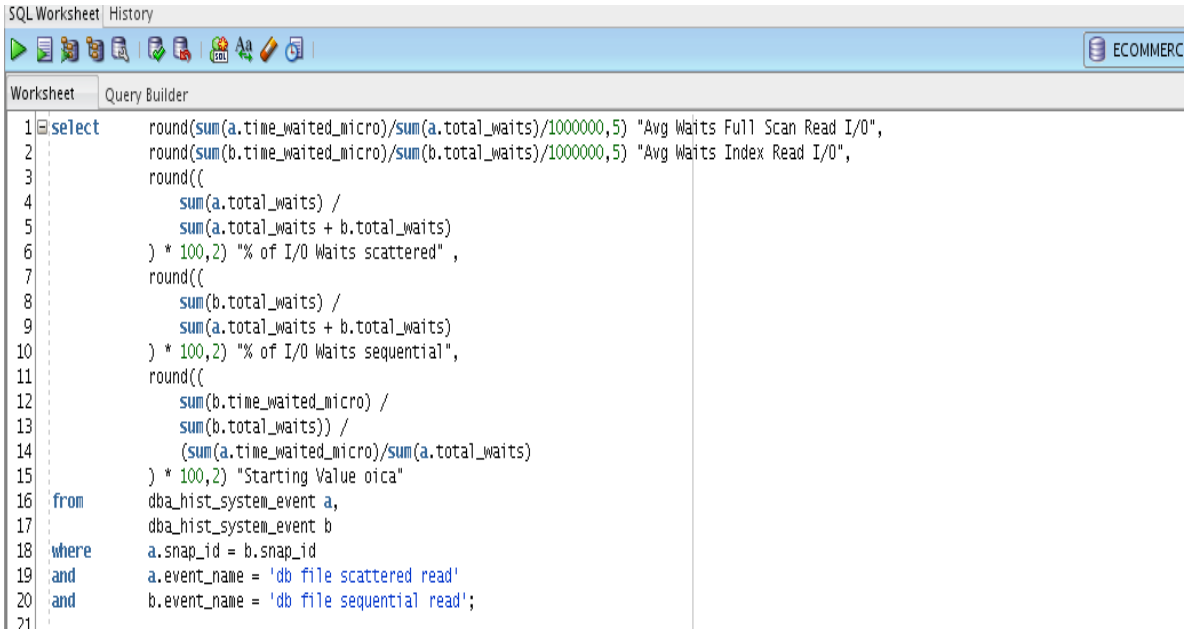


Figura 66. Parâmetro OPTIMIZER_INDEX_COST_ADJ

Para alterar o valor do parâmetro OPTIMIZER_INDEX_COST_ADJ, utilize a seguinte sentença, lembrando que somente o DBA tem acesso a este procedimento:

ALTER SESSION SET OPTIMIZER_INDEX_COST_ADJ = 50.

5.9.3 Parâmetro OPTIMIZER_INDEX_CACHING

Este parâmetro ao ser configurado, indica ao otimizador um valor percentual médio de índices que residem em cache, a fim de influenciar o plano de execução do otimizador.

Seguindo a regra de que quanto maior o valor do parâmetro irá favorecer-se a operações para NESTED LOOP JOINS, quanto menores forem os valores o otimizador irá favorecer as operações de hash joins ou sort merge joins.

Por default, o Oracle já vem setado com 0. Para definir valor para este parâmetro na instalação da instância do banco, deve ter sido criado tablespaces diferentes para dados e índices, logo que as tablespaces de índices devem obedecer o valor de 32 k de blocos, como explicado acima, esse valor dará ganho de performance no banco, somente com esta configuração este parâmetro trará ganho de desempenho ao banco de dados.

Para verificar o valor a ser inserido nesse parâmetro utilize o select abaixo no owner SYSADMIN:

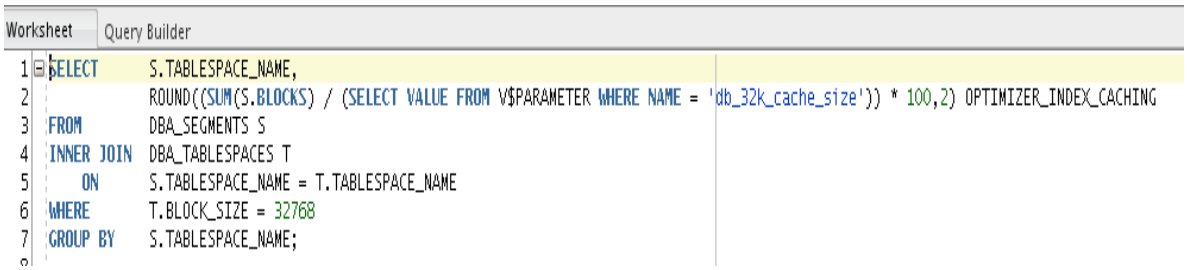


Figura 67. Obtendo valor para o parametros OPTIMIZER_INDEX_CACHING

Para alterar o valor do parâmetro utilize a instrução abaixo:

ALTER SESSION SET OPTIMIZER_INDEX_CACHING = ‘Valor desejado’.

5.9.4 Parâmetro OPTIMIZER_MODE

Esse parâmetro pode ser configurado para designar ao banco de dados a optar pelo uso de INDEX SCAN (utilização de indices), ou pelo uso de FULL TABLE SCAN (varredura completa na tabela) nas consultas do banco de dados.

É indicado para ganho de performance que o valor deste parâmetro esteja com FIRST_ROW, onde irá optar pelo uso de índices.

Para alterar este parâmetro no banco de dados utilize o script abaixo:

ALTER SESSION SET OPTIMIZER_MODE = FIRST_ROWS;

Existem inúmeros parâmetros no banco de dados, citados acima os principais parâmetros que influenciam em um melhor desempenho do banco de dados Oracle OLTP, deve-se ter muito cuidado ao se alterar os valores destes parâmetros, devendo levar em consideração todos os aspectos como sistema operacional, hardware, quantidade de dados, entre outros.

CONCLUSÃO

O fator de preocupação com o desempenho trata-se de tema relevante, mas com pouca exploração até a atualidade. Muito se sabe que o problema existe e que há formas de melhoria, porém a falta de um bom conteúdo que abranja o tema na língua portuguesa, tal assunto é deixado ao acaso, tentando ser contornado em outras perspectivas.

Em razão da grande massa de dados que compõem os bancos de dados e a necessidade cada vez maior de tempo de resposta em tempo real, faz-se necessário a elaboração de código limpo, levando em consideração as diretrizes de um banco de dados. Além disso, sempre é preciso pensar no poder de processamento do ambiente para atingir tempos de resposta mais curtos e somente os dados foco de buscas.

Pode-se observar através do estudo realizado sobre tuning ser o mesmo de grande importância para os novos desenvolvimentos bem como para manutenção das aplicações, sendo necessário monitoramento constante de desempenho, ajustes e administração do ambiente, partindo do princípio de políticas a serem implantadas e seguidas, como procedimentos integrados para o gerenciamento do desempenho do banco de dados.

Para obtenção de resultados satisfatórios, é imprescindível considerar-se as fases iniciais de implantação do banco de dados, bem como o desenho do projeto do mesmo. Com isso, evita-se a necessidade de reprojeto ou recodificação dos códigos fontes.

No transcorrer do desenvolvimento do trabalho, buscou-se demonstrar as melhores práticas de programação de SQL, as configurações de ambiente, os quais foram comprovados por meio técnicas de medição de tempo de resposta utilizando-se de um banco de dados Oracle 11G Enterprise Express Edition. Alguns ganhos de desempenho não puderam ser demonstrados em razão de se tratar de um banco de dados gratuito da Oracle, o qual possui certas limitações.

Apesar da longa discussão do tema no presente trabalho, este ainda possibilita muitas outras pesquisas e aprofundamento do assunto, como o estudo do tuning em sistemas operacionais e tuning de instância, temas estes que exigem além de todo o descrito neste trabalho, um amplo conhecimento de infraestrutura.

REFERÊNCIAS

ANDRADE, Luciano Duarte. **Otimização de Consultas de Aplicações T-SQL em Ambiente SQL Server 2000.** 2005. 52 f. Monografia (Graduação em Ciência da Computação) – Departamento de Ciência da Computação, Instituto de Matemática, Universidade Federal da Bahia, Salvador, 2005. Disponível em: http://disciplinas.dcc.ufba.br/pub/MATA67/TrabalhosSemestre20051/Monografia_Luciano_Andrade.pdf Acesso em: 16 maio 2012.

ATHREYA, Jagan R. **Banco de Dados Oracle 11g: Visão geral do Real Application Testing e da capacidade de gerenciamento.** Junho de 2007. Disponível em: http://www.oracle.com.br. Acesso em: 17 maio 2012.

BURLESON, Don, **High Performance Oracle8 Tuning**. Coriolis Group Books, 1997.

BURGO, Américo F. T. **Engenharia de Software e Banco de Dados: Tuning de banco de dados em um software de CRM**. 2007. Monografia (Especialização em Engenharia de Software e Banco de Dados) – Universidade Estadual de Londrina. Londrina, 2007.

CARNEIRO, A. P.; FREITAS, A. L. C.; MOREIRA, J. L. Artigo Científico: **TUNING - Técnicas de Otimização de Banco de Dados Um Estudo Comparativo: Mysql e Postgresql**, FURG - Rio Grande do Sul, 2006.

CASTRO, Gabriela de. **Melhorando a performance.** 2011. Oracle – support services. Disponível em: http://www.oracle.com. Acesso em: 17 maio 2012.

CASTRO, Gabriela de. **Oracle 7 Performance tuning: memory, I/O and Contention.** 2011. Oracle support services. Disponível em: http://www.oracle.com. Acesso em: 17 maio 2012.

COLELLO, David. **Arquitetura do Banco de Dados Oracle 11g no Windows.** Julho de 2007. Oracle. Disponível em: http://www.oracle.com.br Acesso em: 17 maio 2012.

DATE, C. J. **Introdução a Sistemas de Bancos de Dados.** Tradução da 8ª edição americana de Daniel Vieira. 7ª reimpressão. Rio de Janeiro: Elsevier, 2003.

ELMASRI, Ramez; NAVATHE, Shamkant. **Fundamentals of Database Systems.** Benjamim-Cummings. 3ª edição, 2000.

GIORGINO, Paula dos Santos. **Sistema de Gerenciamento de Banco de Dados – Oracle.** Disponível em: http://pt.scribd.com/doc/77492276/Artigo-Oracle. Acesso em: 04 maio 2012.

HIRATSUKA, Eleni Lumi. **Você é um Tuner?** Disponível em: http://www.batebyte.pr.gov.br/modules/conteudo/conteudo.php?conteudo=1681. Acesso em: 09 maio 2012.

IKEMATU, Ricardo Shoiti. **Realizando Tuning na Base de Aplicações.** Disponível em: http://www.batebyte.pr.gov.br/modules/conteudo/conteudo.php?conteudo=1592. Acesso em: 10 maio 2012.

LEGATTI, Eduardo. **A evolução dos bancos de dados Oracle.** Disponível em: http://eduardolegatti.blogspot.com.br/2007/04/evoluo-dos-bancos-de-dados-oracle.html. Acesso em: 04 maio 2012.

LEWIS, Jonathan. **Cost Based Oracle.** 2005.

ORACLE. **A história do Oracle: Inovação, liderança e Resultados.** Disponível em: http://www.oracle.com/br/corporate/press/story-346137-ptb.html. Acesso em: 04 maio 2012.

PRADO, Fábio. **Tunando instruções SQL no Oracle 10g.** 2010. Disponível em: http://www.fabioprado.net. Acesso em: 17 maio 2012.

RICADO, D. TKPROF – Utilizar ou não? E Porque? Disponível em: http://profissionaloracle.com.br/blogs/drbs/2009/06/30/tkprof-utilizar-ou-nao-e-porque/. Acesso em: 14 outubro 2012.

ROB, Peter; CORONEL, Carlos. **Sistemas de Banco de Dados: Projeto, implementação e administração.** Tradução da 8ª edição norte-americana. Revisão técnica Ana Paula Appel. 1ª edição. São Paulo: Cengage Learning, 2011.

SETZER, Valdemar W. **Bancos de dados: conceitos, modelos, gerenciadores, projeto lógico e projeto físico.** 3ª edição revista. São Paulo: Edgard Blucher Ltda, 1995.

SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S. **Sistemas de Banco de Dados.** Tradução de Daniel Vieira. 5ª edição – Rio de Janeiro: Elsevier, 2006.

SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S. **Sistemas de Banco de Dados.** Tradução de Marília Guimarães Pinheiro e Cláudio César Canhete. 3ª edição – São Paulo: Makron, 2001.

SOUSA, Humberto Delgado de. **O conceito de tuning aplicado em instruções SQL dentro de um Banco de Dados Oracle.** 2011. 80p. Monografia (graduação) – Universidade Nove de Julho (UNINOVE), São Paulo: Campus Memorial Barra Funda, 2011.

SOUZA, F. A.; ALONSO, A. K. P. B.; SANTOS, R., Artigo Científico: **Avaliação das principais técnicas de desempenho em banco de dados Oracle através da otimização SQL**, Policamp – São Paulo, 2007.

SOUZA, Ana Paula dos Santos; CAMPOS, Bruno Fidelis; DIAS, Carla Glênia Guedes; ALVES, Michel Batista; VIEIRA, Carlos Eduardo Costa; CARELLI, Flávio Campos; COSTA DE SÁ, Luiz Fabiano. **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. Acesso em 14 maio 2012.

SUMIT, Sarim. **Oracle DBA: Dicas e técnicas.** Rio de Janeiro: Campus, 2000.

TEOREY, Toby; LIGHTSTONE, Sam; NADEAU, Tom. **Projeto e Modelagem de Banco de Dados.** 4. ed. Tradução de Daniel Vieira. Rio de Janeiro: Editora Elsevier, 2007.

VALIATE, Pedro. **Artigo da SQL Magazine 36 - Índices no Oracle.** 2012. Disponível em: http://www.devmedia.com.br/artigo-da-sql-magazine-36-indices-no-oracle-parte-i/6835#ixzz26sPHNCK6. Acesso em: 18 setembro 2012.

WHALEN, Edward. **Oracle 8 in 21 days**, Sams Publishing, 1996.

WEBER, Flávio Augusto; ANGELOTTI, **Elaini Simoni. Otimizando Consultas SQL no ambiente SQLServer.** Disponível em: http://www.batebyte.pr.gov.br/modules/conteudo/conteudo.php?conteudo=1184. Acesso em 10 maio 2012.

YONG, Chu Shao. **Banco de Dados: organização, sistema e administração.** São Paulo: Atlas,1990.