UNIVERSIDADE FEDERAL DO PAMPA
CIÊNCIA DA COMPUTAÇÃO

AMANDA CARICATTI ESTRELA

FATORES QUE INFLUENCIAM NO DESEMPENHO DE SISTEMAS DE BANCO DE DADOS

Trabalho de Conclusão de Curso

Alegrete

2012

AMANDA CARICATTI ESTRELA

FATORES QUE INFLUENCIAM NO DESEMPENHO DE SISTEMAS DE BANCO DE DADOS

Trabalho de Conclusão de Curso apresentado como parte das atividades para obtenção do título de bacharel em Ciência da Computação na Universidade Federal do Pampa.

Orientador: Prof. Me. Alencar Machado

Co-Orientador: Prof. Me. Sergio Luis Dill

Alegrete

2012

AMANDA CARICATTI ESTRELA

FATORES QUE INFLUENCIAM NO DESEMPENHO DE SISTEMAS DE BANCO DE DADOS

Trabalho de Conclusão de Curso apresentado como parte das atividades para obtenção do título de bacharel em Ciência da Computação na Universidade Federal do Pampa.

Trabalho apresentado e aprovado em: 04 de Janeiro de 2012.

Banca examinadora:

Prof. Msc. Alencar Machado
Orientador
Ciência da Computação - UNIPAMPA

Msc. Sérgio Luis Dill
Coorientador

Prof. Msc. Daniel Welfer
Ciência da Computação - UNIPAMPA

Profa. Dra. Amanda Meincke Melo
Ciência da Computação - UNIPAMPA

AGRADECIMENTOS

Primeiramente, gostaria de agradecer a Deus. Tenho a certeza de que sem ele não estaria completando mais esta etapa.

A toda minha família, que sempre esteve presente durante a minha formação. Agradecimento especial a minha vó, a minha heroína! Sempre presente.

Aos meus amigos, irmãos de todas as horas: Isabel e Miguel.

A todos os meus professores, em especial aos professores Alencar Machado e Sérgio Dill por me orientarem no desenvolvimento do presente trabalho.

A todos vocês, meu muito obrigada!

RESUMO

Comumente, no cotidiano das organizações que usufruem de sistemas de Banco de Dados (BD), o desempenho satisfatório de tais sistemas é considerado um elemento crucial para a obtenção do sucesso nas aplicações desempenhadas pelas mesmas. Uma vez que tais sistemas frequentemente disponibilizam informações que contribuem para o processo de tomada de decisão, a realização de atividades de sintonia nesses sistemas constitui uma das principais funções de um administrador de sistemas de banco de dados (DBA). O objetivo da realização de sintonia é fazer com que o sistema de banco de dados obtenha um melhor desempenho, e com isso garantir um comportamento satisfatório e condizente com o tempo de resposta esperado para as aplicações em questão.

Neste contexto, o objetivo principal deste trabalho é realizar um estudo sobre os fatores que influenciam no desempenho dos sistemas de banco de dados. Para isso analisou-se o processo de construção de um banco de dados operacional, ao iniciar-se pela fase de modelagem conceitual dos dados, projeto lógico de banco de dados, projeto físico de banco de dados e a posterior utilização desses sistemas para a extração de informações, nesse processo o intuito foi identificar quais eram as atividades de sintonia existentes em cada fase e o impacto que elas exerciam no fator desempenho.

Por fim, após a realização do estudo teórico, foi realizada uma análise sobre uma base de dados pública da Transaction Processing Performance Council (TPC). Tal análise procurou identificar a influência que as decisões acatadas nas fases iniciais de projeto de banco de dados exerciam no momento da utilização desses sistemas. Para tal foram submetidas consultas para a extração de informações desse sistema sem a aplicação de técnicas de sintonia, os planos de consulta definidos pelo sistema gerenciador de banco de dados (SGBD) para retornar tais informações foram coletados e armazenados, posteriormente foram aplicados alguns conceitos estudados durante o desenvolvimento do presente trabalho, foram realizadas análises sobre a configuração padrão de alguns parâmetros do SGBD em relação às configurações de hardware da máquina utilizada para a realização do presente estudo, foram realizadas alterações que incluíam modificações no projeto lógico e físico da base de dados, após as modificações foram submetidas novas consultas para a extração de informações desse sistema, os planos de consulta definidos pelo SGBD para retornar tais informações foram comparados com os planos de consulta definidos pelo SGBD na fase que antecede as modificações, com isso foi possível analisar a aplicação prática de alguns conceitos estudados durante o desenvolvimento do presente trabalho e validar o estudo teórico realizado.

Palavras-chave: Banco de Dados, Sintonia, PostgreSQL.

LISTA DE FIGURAS

LISTA DE TABELAS

GLOSÁRIO

DBA - Database Administrator

DCL - Data Control Language

DDL - Data Definition Language

DML - Data Manipulation Language

E/S - Entrada/Saída

FN - Forma Normal

KB - Kilobyte

MB - Megabyte

RAM - Random access memory

R/N - Regras de Negócio

SGBD - Sistema Gerenciador de Banco de Dados

SGBDR - Sistema Gerenciador de Banco de Dados Relacionais

SQL - Structured Query Language

TI – Tecnologia da Informação

TPC- Transaction Processing Performance Council

WAL - Write Ahead Log

Sumário

1 INTRODUÇÃO

No cotidiano das organizações, a necessidade de se manipular informações e dados, faz com que os sistemas de banco de dados constituam um elemento importante no sistema de informação de qualquer organização. Segundo (MANNINO,2008) a tecnologia de banco de dados auxilia na consolidação desses dados e os transforma em informações úteis para o processo de tomada de decisão.

O processo de sintonia realizado em sistemas de Banco de Dados, "[...] é a atividade de ajustar os parâmetros, configurações e estruturas de um sistema de banco de dados às necessidades de uma determinada carga de trabalho [...]" (LIFSCHITZ et al., 2004, p. 02).

O administrador de banco de dados (DBA) considera fatores como o hardware e os softwares disponíveis ao Sistema Gerenciador de Banco de Dados (SGBD) para a realização de sintonia nesses sistemas (RAMALHO, 2002).

Melhorias no hardware podem trazer um retorno benéfico, entretanto, antes de se investir financeiramente em hardware na realização de sintonia, o DBA comumente verifica as possibilidades de obter uma melhoria no desempenho desses sistemas através dos próprios recursos disponíveis por parte de software, inclusive nem todas as melhorias de desempenho obtidas por parte de software podem ser obtidas por parte de hardware (SOUZA, 2005).

Em termos didáticos, podem-se dividir as ações de sintonia por parte de software em três grandes tipos: (1) refinamento do esquema das relações e as consultas/atualizações feitas no Banco de Dados (BD), (2) configuração do sistema operacional em uso e (3) configuração dos parâmetros do SGBD (TRAMONTINA, 2008).

As modificações realizadas no sistema operacional, geralmente incluem a troca, atualização ou modificações na configuração do sistema operacional em uso, já as realizadas no SGBD incluem mudanças nas configurações dos parâmetros originas do SGBD que são definidos no momento de sua instalação ou a troca do SGBD (SOUZA, 2005).

E, por fim, no que se refere às modificações por parte de refinamento de consultas é importante salientar que o próprio SGBD ao atender uma consulta solicitada pelo usuário, possui um componente para a realização do processamento da mesma que não trata apenas da extração de informações da base de dados, mas também das atividades que resultam na construção de um plano de execução que minimize o custo e maximize o desempenho das consultas, tal componente chama-se Processador de Consultas, ele gera várias alternativas de planos e estima qual é o plano mais eficiente para ser executado (DI LELLO e COUTINHO, 2005).

Um estudo sobre os fatores que influenciam no desempenho dos sistemas de banco de dados, as ações de sintonia existentes por parte de software e a análise de um estudo de caso utilizando um banco de dados (BD), constituiu o foco da investigação realizada durante o desenvolvimento do presente trabalho.

1.1 Motivação

Em ambientes corporativos, a Tecnologia da Informação (TI) é utilizada para obter resultados no processo de tomada de decisão e atendimento aos clientes (SILVA e TEIXEIRA, 2002), sendo comum, para isso, à utilização de sistemas de banco de dados. Uma vez que, que estes são responsáveis por armazenar e disponibilizar os dados de acordo com a necessidade dos usuários.

O volume de dados que estes sistemas precisam gerenciar cresce continuamente, um exemplo de tal crescimento é o banco de dados do Yahoo que supera a marca de 100 terabytes (TB). Logo, na utilização desses sistemas a demanda por velocidade é constante, e os fornecedores continuam a desenvolver novas formas para acelerar o desempenho dos bancos de dados (AHRENDT, 2010).

No presente contexto, um estudo sobre os fatores que influenciam no desempenho desses sistemas por parte de software, uma análise sobre a formulação de consultas que utilizam a linguagem Structured Query Language (SQL) e um estudo sobre a configuração padrão dos parâmetros do próprio SGBD, evidenciam-se como questões importantes a serem analisadas pelos DBAs, para que os sistemas de banco de dados operem com o desempenho desejado.

1.2 Objetivo

O trabalho proposto neste documento teve como objetivo estudar os conceitos inerentes à realização de sintonia em sistemas de banco de dados por parte de software. Especificamente foi analisado o impacto que as distintas formulações de consultas SQL e a alteração dos parâmetros de configuração do próprio SGBD exercem no desempenho do sistema de banco de dados.

No desenvolvimento do presente trabalho, foi realizada uma pesquisa sobre os trabalhos correlatos existentes na literatura que tratavam da realização da atividade de sintonia em sistemas de banco de dados, com ênfase em sintonia por parte de software. A análise forneceu subsídios para identificação de um conjunto de princípios que devem ser levados em consideração na realização de ajustes de sintonia.

Na sequência, para a realização de aplicações práticas dos conceitos teóricos apresentados neste trabalho de graduação. Uma simulação sobre a base de dados pública Transaction Processing Performance Council (TPC-C) utilizando o SGBD PostgreSQL foi realizada. A escolha deste SGBD é justificada no capítulo 4, e é aliada ao fato dele possuir os mecanismos de otimização de consultas desejáveis para a realização do presente trabalho. Além de ser um SGBD de código aberto, destacando-se um dos mais populares e avançados sistemas

gerenciadores de banco de dados com código aberto (MEDEIROS, 2008), estar disponível para realização dos testes e ser uma ferramenta de conhecimento prévio do aluno.

Os resultados da simulação foram analisados com o intuito de extrair novas informações e validar os conceitos estudados. Conforme (PARRA, 2005) quanto maior a base de dados, teoricamente a mesma fica cada vez mais lenta. Então neste trabalho de graduação procurou-se demonstrar que através da aplicação de alguns conceitos estudados durante o desenvolvimento do presente trabalho, é possível ter uma base de dados grande, com o desempenho satisfatório.

1.3 Estrutura

O trabalho está organizado da seguinte forma:

- O capítulo 2 faz uma abordagem de forma ampla sobre o tema central que rege o presente trabalho, a realização de Sintonia em Banco de Dados. Neste capítulo também são apresentados alguns conceitos básico sobre a área de Banco de Dados.
- O capítulo 3 apresenta uma revisão sobre alguns trabalhos presentes na literatura que discorrem sobre o presente tema.
- O capítulo 4 apresenta o experimento realizado durante o desenvolvimento do presente trabalho.
- O capítulo 5 apresenta os resultados dos experimentos realizados no capítulo 4.
- O capítulo 6 apresenta as Considerações Finais.
- O capítulo 7 apresenta as Referências Bibliográficas.

2. SINTONIA EM BANCO DE DADOS

Quando falamos sobre a realização de sintonia em sistemas de banco de dados, frequentemente nos referimos a influencia que ela exerce no desempenho das bases de dados. Segundo (SILBERCHATZ et al. 1999) no estudo do desempenho dos sistemas computacionais devemos considerar cinco fatores: workload, throughput, recursos, otimização e concorrência.

Onde o termo workload refere-se à identificação da carga de trabalho a ser submetida ao sistema, o termo throughput refere-se à medida de produtividade que corresponde à quantidade de trabalho executada pelo sistema durante um dado período de tempo, os recursos são as ferramentas do sistema e do software, o termo otimização refere-se especificamente à propriedade de melhoria de um sistema para reduzir o tempo de execução de tarefas, requisitos de memória ou de outras aplicações em particular, e por fim nesse processo quando a demanda por um recurso em particular de um determinado sistema é elevada, a disputa por este recurso pode gerar o que comumente chamamos de concorrência, logo todo processo de otimização pretende aumentar o throughput do sistema, diminuir a contenção e fazer com que o SGBD tenha à capacidade de executar uma carga maior de trabalho imposta no mesmo intervalo de tempo.

Conforme (ORACLE, 1997), o desempenho em banco de dados é algo que deve ser embutido, uma vez que os ajustes de desempenho não podem ser realizados de forma adequada após o sistema ter sido colocado em produção.

No presente contexto, a realização de atividades de sintonia diz respeito a um processo de refino de banco de dados, mais especificamente, segundo (BAPTISTA ,2008) a sintonia em banco de dados diz respeito aos ajustes realizados nos SGBD para melhor emprego dos seus recursos e proporciona um uso mais eficaz e eficiente dos SGBD, uma vez que os SGBD devem fornecer informações àqueles que as requisitam, a taxa em que o SGBD atende a demanda por informação é o que caracteriza o desempenho dos sistemas banco de dados.

Visando sempre à melhoria no desempenho desses sistemas, as realizações de atividades de sintonia podem ocorrer nas diferentes fases de implantação do mesmo, segundo (ORACLE, 1997), existe uma ordem recomendada para a realização de atividades de sintonia e um conjunto de bons princípios que devem ser analisados em cada etapa, tal ordem e suas respectivas características inerentes ao processo de sintonia são apresentadas a seguir:

Passo 1- Sintonia nas Regras de Negócio: Para que os sistemas de banco de dados obtenham um desempenho satisfatório muitas vezes é necessário que haja uma adaptação das regras de negócio (RN), nessa fase um alto nível de análise do projeto do sistema como um todo deve ser realizada, o ideal é que as regras de negócio sejam consistentes com as expectativas realistas para o número de usuários simultâneos, o tempo de resposta da transação e o número de registros armazenados on-line que o sistema pode suportar.

Passo 2- Sintonia no Projeto de Dados: Nessa fase se determina quais são os dados necessários para as aplicações, quais as relações e os atributos existentes, organizam-se as

informações estruturando-as visando atender às metas de desempenho, analisa-se a utilização de índices, partição dos dados, o processo de normalização dos dados para assegurar que não há dados redundantes, porém, por razões de desempenho posteriormente pode ser aplicado o processo de desnormalização nesses dados.

Passo 3- Sintonia no Projeto da Aplicação: Neste nível consideram-se as configurações de processos individuais, analisam- se quais são os usuários que executam o sistema e a arquitetura utilizada por eles, verifica-se a necessidade de existência de diferentes versões da aplicação, nessa fase também é analisada a utilização de cache de dados para a recuperação de informações.

Passo 4- Sintonia na Estrutura Lógica do Banco de Dados: Como nessa fase o sistema já foi projetado, analisa-se a estrutura lógica do banco de dados, que diz respeito principalmente ao projeto de índices, uma vez que uma quantidade excessiva de índices em uma tabela pode ocasionar gargalos, uma vez que a alteração em uma coluna indexada resulta em trabalho adicional para atualizar também os seus índices, em contrapartida, um número reduzido de índices aumenta o tempo de resposta das consultas, então essa fase visa garantir que os dados não serão nem sobre nem sub indexados.

Passo 5- Sintonia nas Operações de Banco de Dados: Nessa fase é importante à compreensão dos mecanismos de processamento de consultas que o SGBD utiliza, a aplicação é analisada para ver se ela obtém plena vantagem da linguagem SQL e das características projetadas pelo SGBD para aprimorar o processamento de aplicações.

Passo 6- Sintonia nos Caminhos de Acesso: Nessa fase são asseguradas questões relativas ao caminho de acesso aos dados e a sua eficiência, a tarefa de assegurar acesso eficiente pode significar a adição de índices para uma aplicação em particular ou a remoção de um determinado índice, ainda nesta fase poderá ser feita a reanálise do projeto após a construção do banco de dados.

Passo 7- Sintonia de Alocação de Memória: A alocação de recursos de memória é analisada nessa fase, a alocação apropriada dos recursos de memória melhora o desempenho do cache, reduz o *parsing* de declarações SQL e reduz a paginação e o swapping.

Passo 8- Sintonia nos dispositivos de Entrada e Saída (E/S) e Estruturas Físicas: A realização de sintonia de Entrada-Saída E/S e da estrutura física envolve procedimentos como: assegurar a distribuição dos dados de maneira que a E/S também esteja distribuída, evitar conflitos de disco, criar espaços grandes o suficiente para os dados de maneira a evitar a ampliação dinâmica de tabelas e estudar técnicas de armazenamento dos dados em blocos para melhoria do acesso.

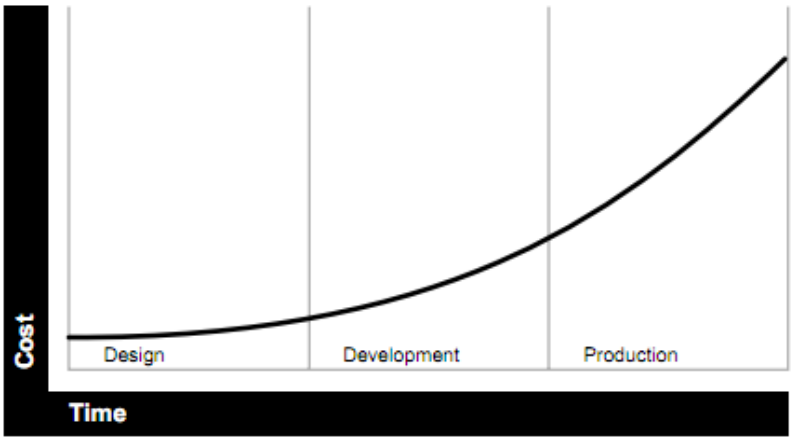
Passo 9- Sintonia no Conflito de Recursos: Nessa fase analisam-se as aplicações concorrentes e o processamento concorrente por múltiplos usuários que podem gerar conflitos por recursos.

Passo 10- Sintonia nas Plataformas de Execução do SGBD: Nessa fase de sintonia são analisadas questões referentes às características do sistema operacional utilizado para a realização das aplicações, para isso é necessário verificar a documentação do SGBD em uso para

assim descobrir quais as maneiras de realizar a sintonia na plataforma na qual o banco está sendo executado.

No presente trabalho o guia produzido pela (ORACLE, 1997) foi utilizado como um auxilio para a compreensão dos conceitos de sintonia, sendo que na realização do estudo de caso durante o desenvolvimento do presente trabalho alguns conceitos teóricos estudados foram adaptados para o SGBD PostgreSQL, tal generalização é consequente do fato do modelo de dados relacional e a linguagem SQL ter se tornado um padrão.

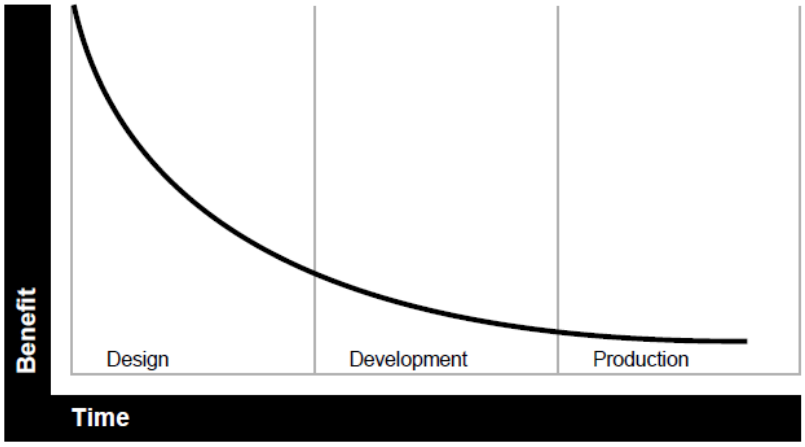
Segundo (HIRATSUKA, 2009), a realização de atividades de sintonia é um processo iterativo, logo para a obtenção de sucesso no processo de realização de sintonia em sistemas de banco de dados, o ideal é que a mesma seja realizada compreendendo todos os aspectos abrangentes possíveis, pois muitos dos problemas de desempenho são decorrentes de decisões inadequadas já no início do projeto e que se refletem nas demais fases, a Figura 1 ilustra a relação de custo da realização de sintonia durante o ciclo de vida da aplicação.



Fonte: ORACLE, 1997, p.43

FIGURA 1 - Custo da realização de sintonia durante a vida de uma aplicação

Na Figura 2, podemos visualizar que os benefícios da realização de sintonia nas aplicações durante o decorrer da sua vida é inversamente proporcional ao custo despendido, ou seja, quanto mais no inicio do ciclo de vida uma aplicação se encontrar, maior serão os benefícios da realização de sintonia, uma vez que a realização de sintonia é um processo amplo e que pode ser realizado em praticamente todas as fases do ciclo de vida das aplicações de banco de dados.

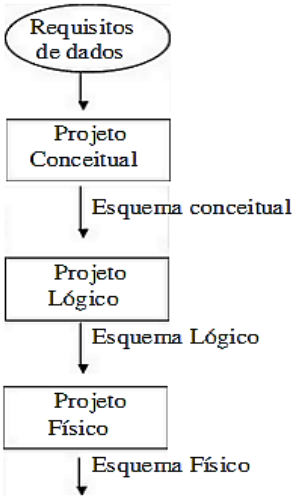


Fonte: ORACLE, 1997, p.44

FIGURA 2 - Benefícios da sintonia durante a vida de uma aplicação

2.1 Projeto de Banco de Dados

O projeto de um banco de dados conforme (SILBERSCHATZ et al., 1999) é decomposto em fases, nesta seção iremos analisar as seguintes fases: Projeto Conceitual, Projeto Lógico e Projeto Físico, a Figura 3 ilustra essas fases, onde nela podemos notar que inicialmente o que há são os requisitos dos dados, esses requisitos são repassados como entrada para a fase do projeto conceitual, onde se gera o esquema conceitual que constitui a entrada da fase do projeto lógico que os transforma no esquema lógico e por fim temos a fase do projeto físico, que tem como saída o esquema físico.



Fonte: VIEIRA, 2002, p.3

FIGURA 3 - Fases do Projeto de Banco de Dados

No projeto de banco de dados após a especificação dos requisitos, é iniciada a fase do projeto conceitual, segundo (MANNINO,2008), a fase de modelagem de dados conceitual utiliza os requisitos de dados para a produção de diagramas entidade-relacionamento (DER), tal diagrama descreve o modelo de dados, definindo uma representação gráfica para eles visando proporcionar uma melhor compreensão da estrutura da base de dados.

Após a fase do projeto conceitual, é iniciado o Projeto Lógico, ele tem por objetivo avaliar o esquema conceitual frente às necessidades de uso do banco de dados pelos usuários e as aplicações, realizando no mesmo possíveis refinamentos para alcançar maior desempenho das operações sobre o banco de dados, nessa fase nós já temos um nível de abstração mais próximo da implementação e dentre os modelos de SGBD existentes: Relacional, Hierárquico e Objeto Relacional, nessa fase define-se o tipo de SGBD que será utilizado na aplicação (VIEIRA, 2002).

O modelo relacional é o mais amplamente utilizado e de acordo com (RICARTE, 2002), para caracterizarmos um SGBD relacional ele pode ser definido como sendo um banco de dados que organiza seus dados em relações, onde cada relação pode ser vista como uma tabela, onde cada coluna corresponde a atributos da relação e as linhas correspondem aos elementos da relação.

Na fase de projeto físico de banco de dados, conforme (MANNINO, 2008) existem duas escolhas extremamente importantes, são elas: criação de índices e a disposição física dos dados, o esquema físico é uma descrição da implementação do banco de dados em memória secundária, ele descreve as estruturas de armazenamento e métodos de acesso usados para efetivamente realizar o acesso aos dados.

2.1.1 Desnormalização

De acordo com (ELMASRI e NAVATHE, 2002), a desnormalização é uma decisão de projeto que visa acelerar as consultas realizadas numa base de dados, é o processo oposto à normalização, a aplicação de processos de normalização nos dados é uma forma de verificar sua qualidade, diminuir redundâncias e incoerências do modelo de dados.

O processo de normalização aplica uma série de regras sobre as tabelas de um modelo para verificar se estas estão corretamente projetadas, uma forma normal segundo (HEUSER, 1998) é uma regra que deve ser obedecida por uma tabela para que esta seja considerada bem projetada. Embora existam mais que três formas normais, normalmente utilizam-se apenas as três primeiras formas normais, onde a Primeira Forma Normal (FN) refere-se à eliminação de atributos compostos ou multivalorados, diz-se que uma tabela está na primeira forma normal, quando ela não contém tabelas aninhadas.

Para entendermos as outras duas formas normais que serão apresentadas, é necessário o conhecimento sobre o conceito de dependência funcional, ainda segundo (HEUSER, 1998),

numa tabela relacional, diz-se que uma coluna C2 depende funcionalmente de uma coluna C1 quando, em todas as linhas presentes na tabela para cada valor de C1 que aparece na tabela o mesmo valor encontra-se em C2, a Figura 4 mostra um exemplo dessa relação.

…	Código	…	Salário	…
	E1		10	
	E3		10	
	E1		10	
	E2		5	
	E3		10	
	E2		5	
	E1		10	

Fonte: HEUSER, 1998, p.140

FIGURA 4 - Dependência Funcional

Como podemos visualizar na Figura 4, diz-se que a coluna Salário depende funcionalmente da coluna Código (ou que a coluna Código determina a coluna Salário) pelo fato de cada valor de Código estar associado sempre ao mesmo valor de Salário.

A Segunda FN determina que atributos não chave devem ser funcionalmente dependentes apenas da chave primária, dizemos que uma tabela encontra-se na segunda forma normal, quando, além de estar na 1FN, ela não contém dependências parciais e a terceira FN refere-se a atributos não chave mutuamente independentes, ou seja, que não dependam um do outro, uma tabela encontra-se na terceira forma normal, quando, além de estar na 2FN, não contém dependências transitivas, uma dependência funcional transitiva ou indireta acontece quando uma coluna não chave primária depende funcionalmente de outra coluna ou combinação de colunas não chave primária (HEUSER, 1998).

Como podemos notar, podemos dizer que uma tabela está normalizada, se ela segue o conjunto de regras citados acima, agora, justamente por não seguir essas regras, uma tabela desnormalizada é considerada uma tabela vulnerável ao surgimento de anomalias quando manipulações de atualização, inclusão, ou inserção são realizadas sobre ela, a integridade dos dados fica ameaçada.

Segundo (PARRA, 2005), alguns exemplos de utilização de técnicas de desnormalização, comumente utilizadas são:

- Tabelas Projetadas – Utilizadas, quando o custo de unir é excessivo, causando degradação de desempenho;
- Tabelas de relatórios – Utilizadas, quando as gerações dos relatórios especializados são muito caras;
- Tabelas Espelho – Utilizadas, quando tabelas são requeridas simultaneamente por dois tipos de ambientes;
- Tabelas Particionadas – Utilizadas, quando grupos distintos usam partes diferentes de uma tabela;
- Tabelas Combinadas – Utilizadas para consolidar relações um-para-um ou um-para-muitos, em uma única tabela;
- Tabelas Rápidas - Apoiam hierarquias ou informam estruturas;
- Desnormalização física - tira proveito de características de SGBD específicas.

2.1.2 Índices

De acordo com (MANNINO,2008), os índices são estrutura de arquivos secundárias que fornecem um caminho alternativo aos dados, eles podem ser organizados em árvores B, estruturas hash ou estrutura de bitmap, onde um arquivo árvore B representa uma árvore balanceada, com múltiplos caminhos, um arquivo hash representa uma estrutura de arquivo especializada em suporte à busca por chave, que transforma um valor-chave em um endereço possibilitando um acesso rápido, enquanto um índice bitmap é uma estrutura de arquivo secundária consistindo em um valor de coluna e um bitmap, um bitmap contém uma posição de bit para cada linha da tabela referenciada.

Apesar dos benefícios que podem ser obtidos através da criação de índices, antes de otimizar o banco de dados criando índices novos, é necessário o entendimento do impacto existente na criação de um índice, por isso, o DBA deve ter uma compreensão dos padrões de acesso da tabela na qual o índice será construído, essa informação é útil para avaliar questões como a porcentagem do volume de acessos e atualizações da tabela e os limitadores de desempenho fixados nas consulta, ou seja, a criação de índices apesar de ser um recurso que visa trazer melhorias no desempenho na base de dados, deve ser utilizado com cautela e um estudo amplo sobre as estruturas e a forma que ele operará na base de dados, faz-se necessário para sua implantação (CARATTI, 2004).

Segundo (WEBER e ANGELOTTI,2009) para que o banco de dados apresente um bom desempenho é necessário partir do pressuposto da existência de uma boa estrutura de banco de dados, adequadamente normalizada e que possua índices úteis, a criação e utilização de índices são tarefas importantes para a obtenção de um melhor desempenho, pois os índices podem acelerar substancialmente a recuperação e a seleção de dados .

De acordo com (POSTGRESQL, 2011), na sua versão 9.0, que é a versão utilizada neste trabalho de graduação, são suportadas quatro opções de índices: btree, hash, gist e gin, e ainda de acordo com o seu manual caso não seja escolhida uma em especifico o método padrão é o btree.

2.1.2.1 Btree

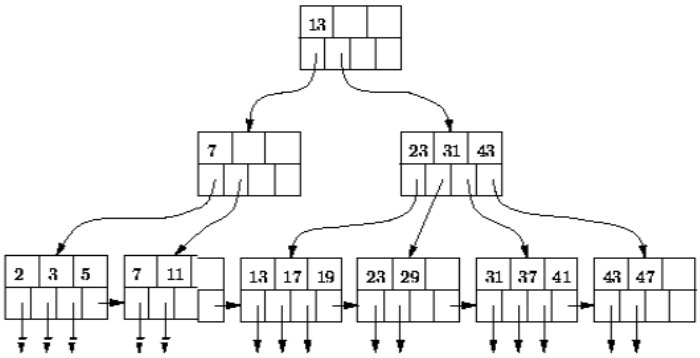
O otimizador de consultas do PostgreSQL considera a utilização de um índice Btree sempre que um atributo indexado estiver envolvido em uma comparação utilizando um dos seguintes operadores: <, <=, =, >=, >, na construção de instruções SQL equivalentes utilizando BETWEEN e IN, este tipo de estrutura também poderá ser utilizada (POSTGRESQL, 2011).

Os índices Btree, segundo (POSTGRESQL, 2011), permite os tipos de indexação, que estão sumarizados na Tabela 1.

TABELA 1 - Indexação: Índice Btree

Multi-Coluna	Único	Expressão	Parcial
Sim	Sim	Sim	Sim

Na utilização de índices Btree, temos no banco de dados uma estrutura do tipo árvore conforme mostra a Figura 5, onde nesta árvore existem vários ramos e folhas, onde os ramos são chamados de nós e representam uma ramificação de uma informação, enquanto as folhas desses nós são blocos contendo referências a vários registros ou outros índices, podemos dizer que um índice Btree é um conjunto ordenado de entradas, que contém um valor chave de pesquisa e um ponteiro para uma linha específica na coluna indexada, quando o valor chave é localizado o ponteiro identifica a linha correspondente na tabela, tal estrutura de índice é recomendada sobre as colunas que possuem valores únicos, sendo mais eficiente quando a pesquisa recupera menos que 20% das linhas de uma tabela, acima disso, a melhor opção é uma pesquisa sequencial na tabela (AVANZI, 2006).



Fonte: MOURÃO et al. 2010, p.12

FIGURA 5 - Exemplo de uma instância de uma estrutura Btree

2.1.2.2 Hash

Conforme (POSTGRESQL, 2011), os índices baseados em estruturas e funções de hash, lidam apenas com comparações de igualdade, o que limita bastante o uso deste tipo de índices, a organização hash utiliza uma função de cálculo de endereço sob a chave de pesquisa, para determinar o bloco no arquivo onde o registro será armazenado, logo, o acesso ao bloco será feito sem a necessidade de uma estrutura de índice, tal índice é recomendado para pesquisas exatas, não sendo indicado para pesquisas com intervalos (AVANZI, 2006). A Tabela 2 demonstra os tipos suportados pelo índice.

TABELA 2 - Indexação: Índice Hash

Multi-Coluna	Único	Expressão	Parcial
Não	Não	Sim	Sim

2.1.2.3 Gist

Segundo (POSTGRESQL, 2011), os índices Gist representam árvores de busca generalizada, eles não são uma estrutura de implementação por si, mas sim um framework sobre

a qual diferentes estratégias de indexação podem ser implementadas, então os operadores particulares com que um índice destes pode ser utilizado dependerá da estratégia desejada implementada como uma classe de operadores, tal índice é uma árvore balanceada generalizada que cobre o uso de várias árvores como B+trees, R-trees, B-trees, RD-trees, entre outras, dando a possibilidade de implementar qualquer uma dessas árvores para a indexação de variados tipos de dados, logo, os índices Gist não são um tipo simples de estrutura de dados, mas sim uma abstração que pode ser usada com alguns exemplos já fornecidos pelo PostgreSQL ou implementando uma estratégia customizada, tanto o processo de criação, quanto o processo de utilização de um índice Gist é algo complexo e envolve definir os operadores de comparação, os tipos suportados pelo índice gist são mostrados na Tabela 3.

TABELA 3 - Indexação: Índice Gist

Multi-Coluna	Único	Expressão	Parcial
Sim	Não	Sim	Sim

2.1.2.4 Gin

Os índices Gin, conforme (POSTGRESQL, 2011), suportam uma indexação invertida que permite o uso de chaves múltiplas, identificadas através de um *array*, assim como o Gist, suporta que o utilizador utilize várias estratégias de indexação, com os seus operadores específicos. porém, ao contrário do Gist, o Gin por padrão inclui classes de operadores para *arrays* de apenas uma dimensão, onde a Tabela 4 demonstra os tipos suportados pelo índice Gin.

TABELA 4 - Indexação: Índice Gin

Multi-Coluna	Único	Expressão	Parcial
Sim	Não	Sim	Sim

Conforme (POSTGRESQL,2011), na escolha de qual tipo de índice deve ser utilizado, Gist ou Gin, as seguintes diferenças de desempenho devem ser analisadas:

- ✓ Pesquisas de índice Gin são cerca de três vezes mais rápido que Gist.
- ✓ Os índices Gin demoram cerca de três vezes mais tempo para serem construídos em comparação aos índices Gist.
- ✓ Gin índices são cerca de dez vezes mais lento do que para atualizar Gist.
- ✓ Gin índices são de duas a três vezes maior que Gist.

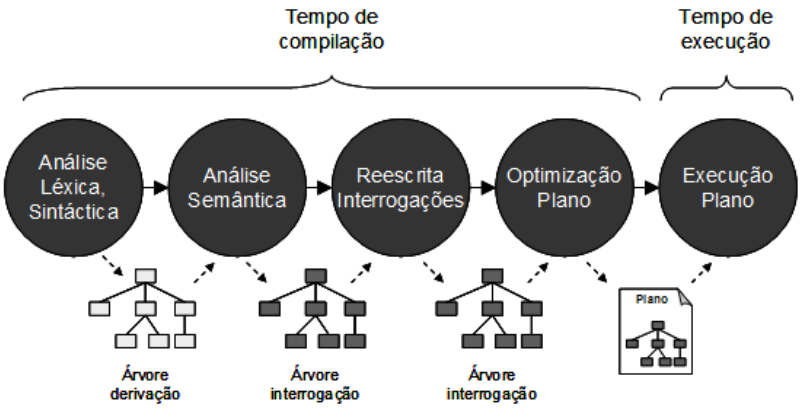
Como regra geral, os índices Gin são melhores para dados estáticos, pois para pesquisas, são considerados mais rápidos. Para dados dinâmicos, índices Gist são mais rápidos para atualização.

2.2 Plano de Execução de Consultas e Métodos de Acesso

Conforme (DI LELLO e COUTINHO, 2005), assim como existem várias formas de se formular uma consulta SQL, o SGBD também possui várias maneiras de executá-la, para isso, é necessário que a consulta seja avaliada e que se encontre uma maneira eficiente de executá-la.

Um plano de execução de consultas é um conjunto de passos usados para acessar as informações num banco de dados de um SGBD relacional. Já que SQL é uma linguagem declarativa, há, tipicamente, diversas alternativas para se executar uma consulta, sendo que o desempenho das alternativas pode variar bastante. Quando uma consulta é submetida ao SGBD, o otimizador de consultas analisará alguns dos diferentes planos possíveis para a execução da consulta e retornará aquele que for considerado a melhor alternativa, o otimizador de consultas é o componente do SGBD responsável por determinar qual é a maneira mais eficiente de se executar uma consulta, para isso ele considera alguns dos planos possíveis de execução para uma dada consulta e tenta inferir qual desses planos é o mais eficiente (SCARABELIN e DIGIAMPIETRI, 2010).

A Figura 6 mostra o processamento realizado pelo SGBD para executar uma consulta.



Fonte: SANTOS, 2009,p.2

FIGURA 6 - Fluxo de processamento de uma consulta

Conforme (POSTGRESQL, 2011), de acordo com a documentação do SGBD é disponibilizado um mecanismo para visualização destes planos de execução de consultas através do comando EXPLAIN, através dele podemos visualizar, qual foi o caminho realizado pelo SGBD para localizar os dados solicitados, sua sintaxe de utilização é apresentada na Figura 7.

```
EXPLAIN [ANALYZE] [VERBOSE]
 Instrução SQL
```

FIGURA 7 - Definição do Comando EXPLAIN

Onde na Figura 7, podemos visualizar o comando EXPLAIN que mostra o plano de execução para uma determinada consulta e os seus parâmetros ANALYZE e VERBOSE, conforme (POSTGRESQL, 2011), caso o comando EXPLAIN seja utilizado sem o parâmetro ANALYZE, a instrução SQL não é realmente executada, apenas é realizado um planejamento sobre o custo referente à sua execução, apenas quando utilizada a opção ANALYZE, é que a instrução SQL é realmente executada, e não apenas planejada, ao utilizar-se essa opção também é possível visualizar o tempo total decorrido gasto em cada nó do plano (em milissegundos) e o número total de linhas retornadas, tal opção é útil para visualizarmos se as estimativas do planejador estão próximas da realidade, já o parâmetro VERBOSE do comando EXPLAIN, mostra a representação interna completa da árvore do plano, em vez de apenas um resumo, geralmente tal opção é utilizada em finalidades especiais de depuração.

Segundo (POSTGRESQL, 2011), o SGBD utiliza métodos de acesso para poder retornar os dados, os métodos de acesso representam os tipo de acesso possíveis para obtenção dos dados solicitados de uma determinada tabela, os principais tipos de métodos de acesso aos dados são:

- ✓ *Sequential Scan*- Como o próprio nome indica, tal método de acesso, percorre todas as linhas de uma tabela, e, para cada linha, verifica se a condição é respeitada. No PostgreSQL é possível ativar ou não o uso deste algoritmo, para isso altera-se o *enable_seqscan*(*boolean*), através do arquivo *postgresql.conf*;

- ✓ *Bitmap Scan* - utiliza os diferentes índices criados numa dada tabela para otimizar e aumentar a velocidade de pesquisas mais complexas. Para ativar ou desativar esta opção usa-se o *enable_bitmapscan(boolean*);

- ✓ *Index Scan* - baseia-se numa pesquisa sobre um índice. Caso seja dado um valor inicial de pesquisa, o *index scan* apenas começará a partir deste valor, e se for dado um valor final, o *index scan* irá parar nesse valor. Todavia, os *seeks* neste algoritmo poderão ser maiores visto que a ordem de retorno dos valores não é sequencial, mas sim pela ordem dos índices. Para ativar ou desativar usa-se o *enable_indexscan(boolean*);

2.3 Otimização de Consultas SQL

Segundo (MANNINO, 2008) a linguagem SQL é uma linguagem de definição, manipulação e controle de banco de dados, logo os comandos SQL podem dividir-se em três grandes categorias: *Data Definition Language* (DDL) que representa a linguagem de definição dos dados, *Data Manipulation Language* (DML) que representa a linguagem de manipulação dos dados e *Data Control Language* (DCL) que representa a linguagem de controle dos dados, no presente trabalho foi focada a utilização das linguagens DDL e DML.

Conforme visto na seção anterior, o próprio SGBD é o responsável pela escolha do plano de acesso para cada consulta, porém pode haver casos de consultas mal formuladas que podem dificultar as escolhas do SGBD, já que segundo (SOUZA, 2005) a forma como as consultas SQL são escritas influenciam na escolha do otimizador, logo existe a possibilidade de se reescrever tais consultas para obter melhorias no desempenho.

A motivação para análise de consultas SQL, é que elas representam uma das principais causas a serem avaliadas no requisito perda de desempenho, conforme (MULLINS,1998, apud ANDRADE, 2005, p. 13) 70% a 80% de todos os problemas de desempenho em aplicações que utilizam os sistemas de banco de dados são causados por consultas SQL mal feitas.

O monitoramento e a tarefa de sintonia de consultas SQL são atividades que consomem uma grande fatia de tempo do DBA, devido a sua complexidade e por elas representarem a maior parte dos acessos realizados nos sistemas gerenciadores de banco de dados relacionais (SGBDR), em tal ambiente o que acaba ocorrendo é que muitas consultas não alcançam o desempenho esperado, devido ao fato de terem sido escritas pensando-se apenas nos resultados a serem obtidos, e não no melhor caminho de acesso para obtê-los, fatores como falta de experiência em desenvolvimento, baixo nível de conhecimento técnico, prazos de entregas subdimensionados e falta de monitoramento individual contribuem para que as consultas sejam ineficientes (PADLIPSKAS, 2011).

Na elaboração de consultas SQL, entender alguns fatores que o próprio otimizador do SGBD utiliza para otimizar tais consultas é de extrema importância, segundo (LELIS,2011), na otimização de consultas SQL, o otimizador considera a seletividade de uma determinada tabela, como sendo a primeira e mais importante medida para a realização da otimização baseada em custo. Ela representa uma fração de linhas de um conjunto resultante de uma tabela, as estatísticas presentes nas tabelas dos catálogos do SGBD são utilizadas para determinar a seletividade de um determinado predicado. A seletividade é diretamente ligada ao predicado da consulta ou a combinação de predicados. O propósito do predicado de uma consulta é limitar o escopo dela em relação a um conjunto de linhas em que estamos interessados. Portanto, a seletividade de um predicado indica quantas linhas de um conjunto vão ser filtradas por uma determinada condição.

A seletividade varia numa faixa de valores de 0.0 até 1.0 onde a seletividade de 0 indica que nenhuma linha será selecionada e 1 que todas as linhas serão selecionadas. A seletividade é igual ao numero de valores distintos que uma coluna possui.

Ainda conforme (PADLIPSKAS, 2011), dentre os fatores de análise para a elaboração de consultas SQL, na utilização da técnica de reescrita de consultas para a eliminação de consultas SQL que impactam de forma negativa no desempenho, no processo de análise das consultas para a aplicação da técnica de reescrita de consultas, alguns erros são frequentemente encontrados, são eles:

- ✓ Uso de função sobre coluna indexada;
- ✓ Subconsulta mal formulada;
- ✓ Uso errôneo de agrupamento e predicado;
- ✓ Uso indevido de operadores de conjunto;
- ✓ Incompatibilidade de tipos de dados.

Nesse processo de estudo dos conceitos para a realização de sintonia em consultas SQL, de acordo com (REZENDE,2004) a correta compreensão do funcionamento dos operadores da álgebra relacional facilita na construção de boas consultas em linguagens SQL, uma vez que o processo de otimização de consultas num SGBD, segue às regras da álgebra relacional.

A seguir são apresentados os conceitos referentes a cada operador da álgebra relacional, conforme (MANNINO,2008):

1-Seleção- Esta operação é representada pelo símbolo σ e sua função é selecionar um subconjunto de linhas de uma determinada tabela.

2-Projeção- Esta operação é representada pelo símbolo π e sua função é produzir um subconjunto de colunas de uma determinada tabela.

3-Produto Cartesiano- Esta operação é representada pelo símbolo x e sua função é relacionar informações de duas tabelas, dadas duas tabelas de entrada, o produto cartesiano de duas tabelas é uma nova tabela constituída de todas as combinações possíveis de linhas de duas tabelas de entrada.

4-União- Esta operação é representada pelo símbolo U, a utilização do operador de união entre dois conjuntos A e B (AUB) é formada por todos os elementos pertencentes a A ou B ou a ambos, ou seja, a operação de união determina todos os membros de dois conjuntos.

5-Diferença- Esta operação é representada pelo símbolo -, a utilização do operador diferença entre dois conjuntos distintos A e B (A-B) onde ambos tenham estruturas totalmente compatíveis entre si, resultará em um conjunto que possui todas as linhas que existam em A mas não existam em B, ela determina os membros únicos somente de um conjunto.

6-Junção- Extrai linhas de um produto de duas tabelas de modo que duas linhas de entrada que contribuem para qualquer linha de saída satisfaçam a uma determinada condição especificada.

7-Junção Externa- Extrai as linhas correspondentes de duas tabelas e as linhas não-correspondentes de uma ou ambas as tabelas.

8-Divisão- Cria uma tabela constituída de todos os valores de uma coluna de uma tabela binária que correspondem com todos os valores de uma tabela unária.

9-Sumarização- Organiza uma tabela em colunas de agrupamento especificadas.

10-Intersecção- Constrói uma tabela constituída de todas as linhas que aparecem em ambas as tabelas especificadas.

Conforme (CINTRA e CAPELLO, 2005), na utilização da álgebra relacional, uma regra de equivalência diz que expressões em diferentes formas podem ser equivalentes, ou seja, podemos transformar uma expressão em outra e ainda assim preservar a sua equivalência, as regras de equivalência são utilizadas pelo otimizador do SGBD para transformar expressões em outras logicamente equivalentes, esse processo é caro, tanto em termos de espaço quanto de tempo, nesse processo a ordem da execução dos operadores básicos influencia no custo da consulta, nesse contexto a operação de seleção é considerada uma das operações mais caras que um sistema de banco de dados pode realizar, uma vez que ela carrega para a memória as linhas que satisfazem a uma condição de seleção, a partir de uma relação de entrada, ela carrega para a memória todas as linhas que satisfazem a um determinado predicado, enquanto a operação de projeção tem o intuito de reduzir o tamanho das tabelas em que é aplicada essa operação, por sua vez, o operador de projeção filtra apenas as colunas da tabela que são realmente de interesse para a aplicação, se possível, logo se deve iniciar o processamento de uma consulta com os operadores de seleção e projeção para restringir o conteúdo que será utilizado em outras rotinas e a utilização dos operadores denominados binários, como junções, uniões e produto cartesiano que geram uma maior quantidade de dados como resultados devem ser adiados ao máximo.

2.4 Otimização de parâmetros no PostgreSQL

Segundo (CARNEIRO, 2009), no mercado existem diversos sistemas gerenciadores de banco de dados, onde na maioria dos casos são instalados, configurados e utilizados com todos os seus parâmetros em seus valores padrões. Sem levar em consideração o tipo de aplicação para o qual serão utilizados, o hardware, e até mesmo o sistema operacional. Desta forma nem sempre se obtém o melhor desempenho do sistema, visto que, diversos parâmetros podem ser considerados e ajustados em um SGBD para um determinado hardware e aplicação específica.

O SGBD PostgreSQL tem uma configuração padrão muito conservadora, o que consequentemente não permite o aproveitamento do potencial da máquina na qual o servidor é instalado, entretanto, é essencial para o DBA entender quais são os parâmetros que influenciam no desempenho do PostgreSQL e saber como alterá-los a seu favor, tendo em vista que os parâmetros *default* são genéricos para todo tipo de estrutura e aplicação, uma análise minuciosa sobre os parâmetros de configuração podem otimizar o desempenho do SGBD, seja melhorando

o aproveitamento do hardware disponível, ou melhorando a organização dos dados e das consultas feitas no banco (CASTOLDI,2005).

Conforme (POSTGRESQL, 2011), os parâmetros de configuração do SGBD residem em um arquivo chamado *postgresql.conf*, onde cada linha desse arquivo especifica um parâmetro, a partir de agora será realizada uma introdução aos principais parâmetros existentes no arquivo *postgresql.conf*, as definições apresentadas a seguir podem ser encontradas de forma mais detalhada na documentação do SGBD.

- ✓ max_connections

Este parâmetro, especifica o número máximo de conexões de clientes que podem existir no servidor PostgreSQL. Mais conexões podem aumentar o throughput do sistema, porém necessitam de mais memória e estruturas de controle de acesso do sistema operacional. Seu valor default é 100.

- ✓ shared_buffers

Define o espaço de memória alocado para o PostgreSQL armazenar as consultas SQL antes de devolvê-las ao buffer do sistema operacional. Um valor mais alto nesse parâmetro pode significar maior desempenho, porém um valor muito grande pode trazer resultados inversos. Seu valor default é 32 megabytes.

- ✓ temp_buffers

Neste parâmetro podemos definir o número máximo de buffer temporários utilizados por cada sessão de banco de dados. Seu valor default é 8 megabytes.

- ✓ work_mem

Especifica a quantidade de memória que pode ser utilizada pelas operações internas de classificação e tabelas de hash, antes de modificar para arquivos temporários em disco. Seu valor default é 1 megabyte.

- ✓ maintenance_work_mem

Especifica a quantidade máxima de memória a ser utilizado em operações de manutenção, como VACUUM, CREATE INDEX e ALTER TABLE ADD FOREIGN KEY. Seu valor default é 16 megabytes.

- ✓ wal_buffers

Número de buffers utilizados pelo Write Ahead Log (WAL). O WAL garante que os registros sejam gravados em LOG para possível recuperação antes de fechar uma transação.

Porém, para bancos transacionais com muitas operações de escrita, o valor padrão pode diminuir o desempenho. Seu valor default é 64 kilobytes.

- ✓ effective_cache_size

Esta configuração dita o quanto de memória Random Access Memory (RAM) será utilizada para cache efetivo (ou cache de dados) do banco de dados que se encontra disponível para uma única consulta. Seu valor default é 128 megabytes.

2.6 Monitoramento de Desempenho no PostgreSQL

No processo de análise para otimização de consultas SQL, uma fonte de informação que subsidia esse processo são os metadados do próprio SGBD, conforme (MONTEIRO et al.,2007) os metadados são frequentemente descritos como "dados sobre dados", ou seja, são um conjunto de características sobre os dados que não estão incluídas nos dados propriamente ditos, mas que fornecem informações adicionais necessárias para que os dados se tornem úteis.

O local onde o SGBD armazena os metadados dos seus esquemas chama-se catálogo do sistema, neste local concentram-se as informações referentes às tabelas, colunas, índices, tarefas e funções que são executadas pelo SGBD.

Conforme (POSTGRESQL, 2011) o PostgreSQL possui um coletor de estatísticas que permite a extração de informações sobre as atividades do servidor. Na realização de sintonia em consultas SQL utilizando este SGBD é possível que obtenhamos informações do coletor referentes ao tamanho da representação em disco de uma determinada tabela, número de linhas de uma determinada tabela, informações sobre a existência ou não de índices nas tabelas, dentre outras informações essenciais no processo de análise de consultas SQL para a realização de sintonia. Uma vez que a coleta de tais informações adiciona alguma sobrecarga ao sistema, existe a opção de configurar o sistema para coletar tais informações, ou não, isto é feito através do arquivo postgresql.conf, tal arquivo encontra-se dentro do diretório data criado no momento da instalação do PostgreSQL.

O presente capítulo descreveu alguns conceitos inerentes ao processo de realização de sintonia em sistemas de banco de dados, conforme visto na realização de sintonia nesses sistemas, uma série de fatores devem ser analisados, tais fatores estão presentes desde o momento da criação da base de dados, no princípio onde são analisadas as regras de negócio, até

o momento de utilização da base de dados, tais fatores devem ser estritamente calculados e analisados para que se consiga obter uma base de dados implantada de forma coerente e que atenda aos seus objetivos, logo muitos esforços por parte do DBA devem ser empreendidos para proporcionar as aplicações de banco de dados um desempenho adequado frente às mudanças de requisitos e adaptações que tais sistemas podem sofrer, inclusive conforme visto, o processo de sintonia é um processo iterativo, logo um acompanhamento constante na base de dados deve ser realizado, senão um sistema que hoje possa ser considerado eficiente amanhã pode tornar-se ineficaz.

3 TRABALHOS RELACIONADOS

Durante o processo de desenvolvimento do presente trabalho, foi realizada uma pesquisa sobre os principais trabalhos que abordam o foco da nossa investigação, o objetivo da análise desses trabalhos era encontrar as sugestões e os resultados obtidos por diversos autores presentes na literatura no que se refere à realização de sintonia em sistemas de banco de dados, uma vez que o intuito do presente trabalho foi analisar os fatores que influenciam no desempenho dos sistemas de banco de dados, foi realizada uma seleção sobre alguns trabalhos lidos e neste capítulo são apresentados os trabalhos que mais se relacionam com o tema estudado no presente trabalho, desta forma podemos analisar e comparar os resultados obtidos pelos autores na realização de sintonia em sistemas de banco de dados com o presente trabalho de graduação.

3.1 Trabalho Relacionado 1 – Otimização de Consultas

Em (IOANNIDIS,1996) é realizada uma análise sobre o processamento de consultas, é enfatizado no trabalho que dada uma consulta, existem muitos planos que um sistema de gerenciamento de banco de dados pode seguir para processá-lo e produzir a sua resposta. Todos os planos são equivalentes em termos de sua saída, mas variam em seus custos, ou seja, a quantidade de tempo que eles precisam para ser executado, logo a otimização de consultas, é absolutamente necessária em um SGBD.

Para a análise dos planos de consultas existentes, são retiradas informações sobre as próprias consultas SQL formuladas para o trabalho, as informações são referentes ao número de paginas que a tabela referenciada na consulta ocupa, número de tuplas que a tabela possui, índices existentes, o custo para o acesso aos dados, dentre outras informações.

É realizada uma abstração do processo de otimização de consultas em um SGBD, sobre a execução dos vários planos existentes que podem ser empregadas para responder a uma determinada consulta.

Nesse processo de análise, o trabalho aborda os conceitos da álgebra relacional, enfatizando que cada consulta é uma expressão algébrica e é representada por uma árvore cujas folhas são as próprias relações e os nós não-folha são os operadores algébricos como seleção, projeção e joins, onde os nós intermediários indicam a aplicação dos operadores correspondentes sobre as relações geradas por seus filhos, onde dada essa organização de acordo com os conceitos da álgebra relacional é possível obter uma resposta para a consulta solicitada.

Durante todo o trabalho são analisadas consultas SQL e para cada consulta são analisados os distintos planos de consultas existentes para cada uma, nesse processo são enfatizadas questões como, a importância da escolha de índices corretos, e o impacto das escolhas de distintos algoritmos de junções.

3.2 Trabalho Relacionado 2 – Seleção de Recursos para Ajustes Autonômicos em Banco de Dados

Em (OH e LEE, 2005) foi realizado um estudo sobre o desempenho em bancos de dados utilizando o SGBD Oracle, eles procuraram identificar quais as alterações realizadas nos parâmetros do SGBD Oracle que influenciam na melhor utilização dos recursos do SGBD, foram realizadas alterações em 4 parâmetros, foram eles: db_cache_size, shared_pool_size, pga_aggregate_target e dbwr_io, tais parâmetros possuíam as configurações padrões para os dados de buffer, memória compartilhada, memória privada e processos de E/S, respectivamente, as alterações nos parâmetros foram realizadas uma por vez e os resultados foram apresentados, para a realização dos testes foram utilizados os benchmarks da Transaction Processing Performance Council (TPC).

Após a identificação das alterações realizadas nos parâmetros do SGBD que eram significativamente impactantes para a utilização correta dos recursos do SGBD, foi proposto um método para a seleção de recursos autonômicos, como as mudanças para melhor utilização dos recursos podem ser reconhecidas como tendo um coeficiente de correlação e coeficiente de variação, eles realizaram os cálculos para determinar de forma precisa as alterações que deveriam ser realizadas para ocasionar um impacto positivo no desempenho de tais sistemas.

Por fim eles concluíram que a pesquisa realizada ajudaria a pavimentar o caminho para a realização de ajustes em banco de dados de forma autônoma, reduzindo a intervenção humana.

3.3 Trabalho Relacionado 3 - Tuning de Desempenho em Banco de Dados

Em (PARRA,2005) faz-se uma análise geral sobre a realização de sintonia em sistemas de banco de dados, são abordadas questões de configurações de hardware, sintonia em parâmetros de configuração dos SGBD, refinamento do projeto de banco de dados e refinamento de consultas SQL, neste trabalho o foco é o entendimento do processo de sintonia de uma forma ampla, procura-se demonstrar como o processo de sintonia é um processo contínuo e como as decisão acatadas numa determinada fase de desenvolvimento de uma aplicação podem afetar outras que a sucedem, apesar dos conceitos referente à sintonia em banco de dados serem exposto de uma forma geral, no momento da aplicação prática dos conceitos estudados, o SGBD escolhido foi o Oracle.

No trabalho foi realizada uma análise minuciosa sobre a utilização da linguagem SQL no que se refere à criação de índices adequados, escolha correta de algoritmos de junções para o processamento das consultas, reescrita de consultas, análise do caminho de acesso aos dados, onde nesse processo ele enfatizou o tema sob a perspectiva de SGBD distintos, ele citou características dos SGBD SQL Server e Oracle 9i para a demonstração dos conceitos teóricos, por fim ele analisou algumas consultas típicas realizadas na base de dados do seu estudo de caso

proposto e analisou os planos de execução das consultas comparando os ganhos de desempenho que foram obtidos.

3.4 Trabalho Relacionado 4 - Algoritmo adaptativo para ajustes de desempenho em Sistema Gerenciadores de Banco de Dados

Em (RODD e KULKRANI,2010) também foi realizado um estudo sobre o desempenho em bancos de dados utilizando o SGBD Oracle, o estudo baseou-se na análise sobre ajustes realizados em parâmetros como: shared_pool_size, buff_cache_size, porém a abordagem adotada difere da abordagem de (OH e LEE, 2005).

No trabalho eles utilizam redes neurais para estimar os ajustes adequados a serem realizados nos parâmetros do SGBD para a realização de sintonia, através da coleta de informações providas pelas estatísticas do sistema e da extração de informações de log do sistema, passavam-se essas informações como entrada para o treinamento da rede neural que então calculava um tamanho adequado para o buffer, para a realização dos testes foram utilizados benchmarchs TPC.

No final, eles enfatizam que o refinamento do sistema proposto leva em consideração o aumento súbito da carga de trabalho e também o conjunto de dados de treinamento da rede neural, além do fato do sistema ter como base um banco de dados apropriado para a caracterização.

3.5 Trabalho Relacionado 5 - Otimização de grandes consultas através do algoritmo Simulated Annealing com base em Gráficos de Aproximação

Em (CHEN et al., 2011), é realizada uma análise sobre a utilização de junções, eles enfatizam que em geral a ordem realizada para unir as tabelas nas operações de junção, é bastante sensível e tem um efeito devastadoramente negativo sobre o eficiência do SGBD.

Conforme os autores, a chave para o sucesso de um Sistema de Banco de Dados, especialmente nos que se baseiam no modelo relacional, é a eficácia do módulo de otimização de consultas do sistema, porém a otimização é um processo caro, principalmente porque o número de planos alternativos de acesso para uma consulta cresce no mínimo de forma exponencial de acordo com o número de relações participantes na consulta.

De acordo com essas informações, os autores analisam qual o melhor algoritmo a ser utilizado pelo SGBD para otimizar as consultas, principalmente as que envolvem operações de junção, para isso os autores introduzem uma série de trabalhos existentes que abordam a otimização de consultas, os autores ainda analisam o fato de existirem algoritmos utilizados

pelos otimizadores atuais que são considerados inadequados para o processamento de consultas que possuem uma grande complexidade esperada.

Segundo os autores, a explicação para a ineficiência encontrada na otimização de consultas complexas, é decorrente do fato que nos algoritmos de otimização a avaliação de desempenho das estratégias é um processo muito complexo, justamente por causa da forte influência, simultânea, de parâmetros aleatórios e outros fatores.

Dentre os algoritmos de otimização avaliados, os autores apresentam um algoritmo probabilístico para a otimização de consultas, o algoritmo apresentado é baseado no Simulated Annealing, que segundo eles é um algoritmo que explora uma analogia entre o modo como um metal se resfria e congela numa estrutura cristalina de energia mínima, os parâmetros passados para o algoritmo para o calculo da função de custo, representa o número de tuplas requisitadas por cada estratégia, para análise dos vizinhos repassados ao algoritmo, que é um processamento de junção para o caso especifico, os autores determinam um conjunto de regras de equivalência de transformação da álgebra relacional para a realização das junções.

Por fim, os autores concluem que eles reduziram o objetivo do otimizador de consultas para encontrar a melhor forma de junção existente, posterioremnte os autores discorrem sobre o efeito das junções, a importância da escolha dos predicados corretos, e o impacto que a disposição e a ordem das tabelas a serem unidas acrescentam no resultado final.

O trabalho apresentado nesta monografia realizou um levantamento sobre as contribuições existentes na literatura para a realização de sintonia em Banco de Dados, entretanto, no presente trabalho se escolheu por utilizar o SGBD PostgreSQL para a demonstração das teorias aqui apresentadas, então apesar dos trabalhos citados nesta seção já terem abordado tal tema, este trabalho visou agregar em conhecimentos, realizando a adaptação e a apresentação de alguns conceitos estudados em outros trabalhos através da perspectiva do SGBD PostgreSQL, uma vez que na literatura foram encontrados até o presente momento poucos trabalhos que aplicam os conceitos de sintonia neste SGBD.

4 REALIZAÇÃO DO EXPERIMENTO

Na realização do experimento, o Sistema Operacional (SO) utilizado foi o Windows 7 e o SGBD PostgreSQL na sua versão 9.0. O hardware utilizado foi um computador portátil com processador Intel Core i3, memória de 4GB e disco rígido de 500GB. A escolha do *hardware* é decorrente do fato do mesmo se encontrar a disposição do aluno no momento da realização do experimento. A escolha das ferramentas seguiram alguns critérios de avaliação que são apresentados no presente capítulo.

4.1 Ferramentas

Para o desenvolvimento do presente trabalho, algumas ferramentas foram avaliadas, os resultados das análises e as ferramentas escolhidas para a realização dos testes, são apresentados nas próximas seções.

4.1.1 Análise dos SGBD Gratuitos

Nesta seção será apresentada a análise realizada sobre os sistemas de banco de dados gratuitos existentes no mercado, tal análise justifica o fato da escolha do SGBD PostgreSQL para a realização do presente trabalho de graduação. Conforme (COLARES, 2007) os três principais bancos de dados livres que podem concorrer de fato com os gigantes do mercado, são: Firebird, MySQL e PostgreSQL.

- ✓ PostgreSQL

Segundo (POSTGRESQL, 2011) o SGBD PostgreSQL é um Sistema Gerenciador de Banco de Dados Objeto-Relacional (SGBDOR). Devido à sua licença aberta, o PostgreSQL pode ser utilizado, modificado e distribuído por qualquer pessoa para qualquer finalidade, seja privada, comercial ou acadêmica, livre de encargos, o SGBD roda em Linux, Windows, e uma variedade de plataformas Unix..

- ✓ MySQL

Segundo (MYSQL, 2011), o SGBD MySQL possui uma versão *open source* que é fornecida sob a licença *General Public License* (GPL), tal versão é suportada por uma comunidade ativa de desenvolvedores de código aberto e entusiastas, o SGBD é considerado um banco de dados com grande agilidade, o SGBD roda em Linux, Windows, e uma variedade de plataformas Unix..

- ✓ Firebird

Segundo (FIREBIRD, 2011), o SGBD Firebird é um banco de dados relacional que oferece muitos recursos, o SGBD roda em Linux, Windows, e uma variedade de plataformas Unix. Qualquer um pode criar uma versão personalizada do Firebird, desde que as modificações sejam disponibilizados, sob a mesma licença Initial Developer's Public License (IDPL), para outros poderem utilizar e construir.

Na escolha do SGBD a ser utilizado no presente trabalho, como o objetivo era analisar a aplicação de técnicas de sintonia existentes por parte de software, que compreendem utilizar os recursos do próprio SGBD para a realização de sintonia, os critérios de avaliação para a escolha do SGBD foram em função das funcionalidades que eram disponibilizadas pelos mesmos, foram considerados os seguintes critérios: suporte a índices, subconsultas, stored procedures, triggers e views, a Tabela 5 apresenta o resultado da análise.

TABELA 5 - Comparativo dos Critérios de Escolha do SGBD

	MySQL 5.5	PostgreSQL 9.0	Firebird 1.5
Índice	Btree e Hash	Btree, Hash e Bitmap	Btree
Stored Procedures	Restrito	Sim	Sim
Triggers	Restrito	Sim	Sim
Views	Restrito	Sim	Sim
Subconsultas	Restrito	Sim	Sim

Através da Tabela 5, que apresenta informações advindas da documentação dos SGBDs em análise, onde as referências se encontram logo acima, foi possível constatar que, o SGBD PostgreSQL possui recursos como: índices, stored procedures, triggers, views e subconsultas, oferecendo mais opções dentre os SGBD gratuitos analisados, além dos recursos disponibilizados pelo mesmo foi também realizada uma análise sobre o mercado conquistado pelo SGBD. Conforme (PGBR, 2011) cada vez mais o SGBD vem crescendo ao redor do mundo, sendo que empresas de grande porte como CAIXA, Skype, BASF e Cisco já tem apostado na ferramenta.

Ainda na análise de mercado sobre a utilização do SGBD PostgreSQL, outros casos também foram analisados. Segundo (ACESSASP, 2011) que é o programa de inclusão digital do Governo do Estado de São Paulo, coordenado pela Secretaria de Gestão Pública, com gestão da Companhia de Processamento de Dados do Estado de São Paulo (PRODESP), a companhia de Metrô do estado de São Paulo realizou a migração para o SGBD PostgreSQL e o utiliza desde

2001. Para tal foram realizados treinamento de administradores de bancos de dados para o gerenciamento do ambiente PostgreSQL, também segundo (OPENGEO, 2008) que é uma empresa voltada ao desenvolvimento de soluções corporativas com inteligência espacial e software livre, a Diretoria de Obras Militares do Exército (DOM), utiliza tecnologias geoespaciais em software livre para o desenvolvimento do Sistema Unificado do Processo de Obras (OPUS). O novo sistema de gestão de obras que atenderá todas as organizações militares do Exército, sendo que o sistema é desenvolvido na linguagem Java, com a base de dados estruturada no PostgreSQL/PostGIS.

Logo, de acordo com a análise das funcionalidades do SGBD apresentadas na Tabela 5 e com base nas informações de mercado sobre os utilizadores do SGBD, foi decidido utilizar o SGBD PostgreSQL para o desenvolvimento do presente trabalho.

4.1.2 Análise das ferramentas de Benchmark

Nessa seção será apresentada a análise realizada sobre as ferramentas de Benchmark gratuitas existentes no mercado, conforme (MANNINO, 2008), podem ser utilizados benchmarks para a avaliação do desempenho de um SGBD, sendo que um benchmark representa uma carga de trabalho para avaliar o desempenho de um sistema ou produto, logo o DBA pode utilizar os resultados dos benchmarks para obter estimativas razoáveis sobre o desempenho de um SGBD em particular. A análise das ferramentas de Benchmark gratuitas existentes no mercado, justifica o fato da escolha da ferramenta BenchmarkSQL para a realização do presente trabalho de graduação, as ferramentas pesquisadas foram: TPCC-UVA, PolePosition, BenchmarkSQL e JMeter.

A justificativa da utilização da ferramenta, é que de acordo com (O'NEIL, 1993), na prática cada vez que uma nova metodologia de benchmark é apresentada, existe também a necessidade do desenvolvimento das ferramentas que a executam. Logo, para o presente trabalho como foi utilizada a metodologia de benchmark TPC-C, para a realização de sintonia nos parâmetros do SGBD, houve a necessidade de haver uma análise sobre algumas ferramentas de benchmark existentes no mercado, que suportassem a metodologia utilizada.

- ✓ TPCC-UVa

A ferramenta TPCC-Uva, conforme (UVA, 2011) é uma implementação (não oficial) do Transaction Processing Performance Council (TPC-C) de referência, destinado para fins de pesquisa, TPCC-UVa é inteiramente escrito em linguagem C, e utiliza o banco de dados PostgreSQL.

- ✓ BenchmarkSQL

A ferramenta Benchmarks, conforme (SOURCEFORGE, 2011) faz parte de um projeto Open Source desenvolvido em 2004, foi desenvolvida utilizando a linguagem de programação Java. Utiliza as bibliotecas JDBC para realizar a comunicação com diferentes SGBD por meio da linguagem de programação SQL.

- ✓ PolePosition

A ferramenta PolePosition, de acordo com (POLEPOSITION, 2011) é uma suíte de testes de benchmark para comparar mecanismos de desempenho em banco de dados, a ferramenta é fornecida sob a licença General Public License (GPL).

- ✓ JMeter

A ferramenta JMeter, de acordo com (APACHE, 2011), é uma ferramenta utilizada para testes de carga de serviços oferecidos por sistemas computacionais, a ferramenta é parte do projeto Jakarta da Apache Software Foundation, suporta requisições Java Database Connectivity (JDBC), para executar consultas em sistemas de banco de dados.

Na escolha da ferramenta de Benchmark para ser utilizada no presente trabalho, alguns critérios foram adotados, dentre os trabalhos relacionados com este trabalho de graduação, os trabalhos de (RODD e KULKRANI,2010) e (OH e LEE, 2005) utilizaram benchmarks TPC para a realização dos testes de desempenho, devido à confiabilidade que tais benchmark apresentam. Então, na escolha da ferramenta de benchmark para analisar o desempenho global do SGBD em relação a configuração dos seus parâmetros, o principal critério adotado foi a utilização da metodologia TPC, para a análise de consultas SQL, o principal critério adotado foi que a ferramenta deveria possibilitar a utilização de testes de consulta SQL de elaboração própria, uma vez que conforme (IOANNIDIS,1996) a linguagem SQL pode ser formulada de distintas formas, já que é uma linguagem declarativa, porém cada formulação acarreta a escolha de um plano de consulta distinto, a Tabela 6 apresenta o resultado da análise.

TABELA 6 - Comparativo das Ferramentas de Benchmark

	TPCC-UVa	BenchmarkSQL	Poleposition	JMeter
Metodologia	TPC-C	TPC-C	Barcelona	Independente
S.O Compatível	GNU/Linux	Independente	Independente	Indepedente
SGBD Compatível	PostgreSQL	Vários	Vários	Vários
Simulação	Sim	Sim	Sim	Sim

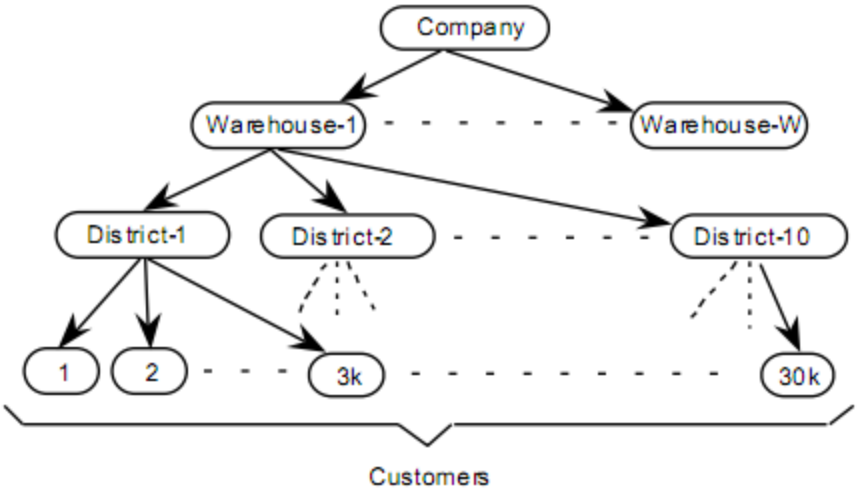
É importante salientar que cada uma das ferramentas analisadas apresentam as suas particularidades, logo deve ser analisado o que se deseja fazer e o que a ferramenta disponibiliza, para o presente trabalho foi analisada qual a ferramenta mais adequada para a realização dos testes pretendidos, posteriormente foi escolhida a ferramenta BenchmarkSQL para os testes referentes a alteração dos parâmetros do SGBD, uma vez que a ferramenta simula a sua própria carga de trabalho através da utilização do benchmark TPC-C, foi também escolhida a ferramenta JMeter para a análise de consultas SQL de elaboração própria.

4.1.3 A Estrutura do Banco de Dados

De acordo com (TPC, 2010), o benchmark TPC-C, destina-se a avaliação de sistemas com processamento de transações on-line (OLTP), o benchmark é composto de uma série de transações projetadas para simular um típico sistema de aquisição de produtos, com as funcionalidades comumente encontradas nessas aplicações, tais como criação de novos pedidos, realização de pagamentos, registro de saída de produtos, verificação do estado de um pedido, entre outras.

A empresa representada no benchmark é uma fornecedora atacadista que possui um determinado número de distritos de vendas e lojas associadas geograficamente.

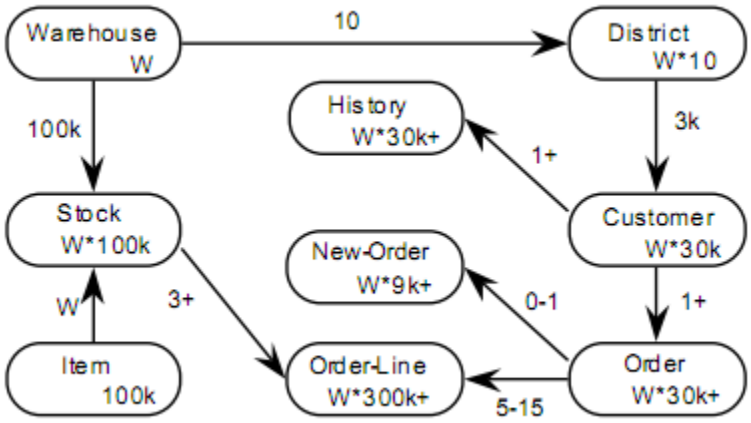
Conforme a empresa cresce, novas lojas e distritos são criados. Cada distrito tem capacidade para atender 3000 clientes. Existem também os armazéns regionais, sendo que cada armazém atende 10 distritos. Todas as lojas mantêm estoque de 100K produtos que são vendidos pela empresa. A Figura 8 ilustra essa hierarquia.



Fonte: TPC, 2010, p.10

FIGURA 8 - Hierarquia das Relações do TPC-C

A cardinalidade das tabelas do benchmark TPC-C varia em função do número de registros na tabela WAREHOUSE. Apenas a tabela ITEM possui um número fixo de registros. O relacionamento e a cardinalidade entre as tabelas são apresentados pela Figura 9, onde o k corresponde ao número de warehouses gerados, que para o nosso caso foi warehouses=1.



Fonte: TPC, 2010, p.11

FIGURA 9 - Diagrama de entidade-relacionamento do TPC-C

A especificação TPC-C define a maneira como cada coluna de cada tabela deve ser carregada e o número de registros que devem ser inseridos. A Tabela 7 apresenta a quantidade de tuplas que cada tabela deve conter para cada warehouse configurado.

TABELA 7 - Cardinalidade Inicial das Tabelas do TPC-C

Tabela	Tuplas	Tamanho da Tupla (bytes)	Tamanho da Tabela (em 1 KB)
WAREHOUSE	1	89	0.089
DISTRICT	10	95	0.950
CUSTOMER	30K	655	19.650
HISTORY	30K	46	1.380
ORDER	30K	24	720
NEW-ORDER	9K	8	72
ORDER-LINE	300K	54	16.200
STOCK	100K	306	30.600
ITEM	100K	82	8.200

4.2 Otimização na Configuração dos Parâmetros do SGBD

Após analisada a especificação e a configuração padrão, dos parâmetros do SGBD, que conforme (POSTGRESQL, 2011) exercem um maior impacto no fator desempenho, foi possível notar que a configuração padrão do SGBD estava ajustada para um hardware com características inferiores ao hardware utilizado, uma vez que os parâmetros em suas configurações padrões estavam ideais para uma configuração de hardwares que disponibilizavam 1 GB de memória, para as operações, como na realização do experimento, possuíamos uma quantidade de memória um pouco superior, para a realização das operações do SGBD, foram realizadas as alterações nos seguintes parâmetros apresentados na Tabela 8.

TABELA 8 - Análise dos Parâmetros do SGBD

	Padrão	Sintonia (Tuning)
shared_buffers	32 MB	64 MB
temp_buffers	8 MB	16 MB
work_mem	1 MB	2 MB
wal_buffers	64 KB	128 KB
effective_cache_size	128 MB	256 MB

A Tabela 9 apresenta os resultados obtidos através da ferramenta BenchmarkSQL, o teste foi realizado para um warehouse e dez terminais, é importante salientar que a ferramenta que disponibiliza essa opção, como para o presente experimento havia sido gerado apenas um warehouse, posteriormente nos testes com a ferramenta BenchmarkSQL foi informado o número exato de warehouses gerados, ou seja número de warehouses igual a um, a opção de colocar mais terminais foi justamente para verificar o desempenho do sistema, analisando o seu comportamento frente a um número maior de usuários, tal qual como ocorre comumente na utilização desses sistemas em ambientes reais.

TABELA 9 - Comparativa sobre a Média de Transações Por Minuto

	Padrão	Sintonia (Tuning)
Número de Transações	3732	3698
Média de Transações por Minuto	3684.17	4420.02

No total foram realizados sete testes para cada configuração, primeiramente com os parâmetros em sua configuração padrão e posteriormente com os parâmetros já modificados, a seguir na Figura 10 os resultados obtidos são ilustrado graficamente.

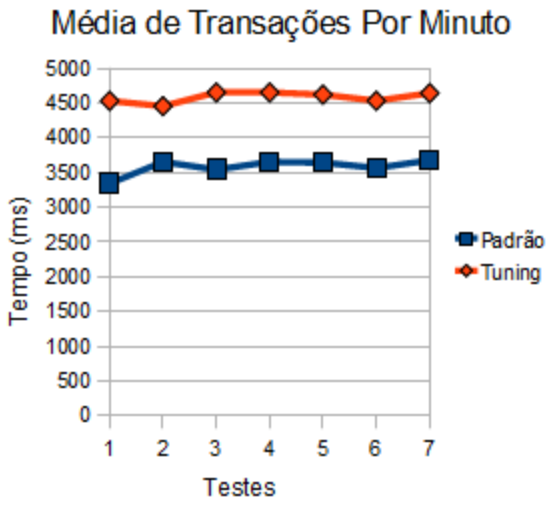


FIGURA 10 - Média de Transações Por Minuto

Conforme (TPC, 2011) o TPC-C executado, realizou uma carga de trabalho para ambientes transacionais, este benchmark mistura transações de apenas leitura com transações que realizam intensivas atualizações, um maior esclarecimento sobre as características das consultas SQL executadas na base de dados, estão disponíveis na seção apêndice, é importante salientar que a amostra de transações TPC-C, conforme a (TPC, 2011) não representam a transação real, mas sim, um exemplo dos tipos de transações executadas pelo TPC-C.

4.3 Otimização em Consultas SQL e no Projeto de Banco de Dados

Esta seção realiza uma aplicação prática sobre as técnicas de sintonia existentes para a elaboração de consultas SQL, juntamente com a análise das decisões de projeto de banco de dados, os resultados encontrados são apresentados a seguir.

4.3.1 Definição de Índices Seletivos

A Figura 11 apresenta uma consulta realizada sobre a tabela customer, a consulta é restringida pela coluna c_city da tabela, esta coluna possui uma grande quantidade de valores distintos, inicialmente será apresentado o custo para a execução da consulta, sem a utilização de um índice sobre a coluna c_city, logo após um índice será criado sobre esse campo.

```
EXPLAIN ANALYZE SELECT *
FROM customer
WHERE c_city LIKE 'A%'
```

FIGURA 11 - Consulta restringida pelo campo c_city

A Tabela 10 apresenta o plano de consulta padrão para a consulta apresentada na figura 15 e o plano de consulta modificado, já utilizando o índice criado.

TABELA 10 - Plano da Consulta através do Comando EXPLAIN ANALYZE

	Padrão	Sintonia (Tuning)
Tipo de Acesso	Sequential Scan	Bitmap Index Scan
Custo Estimado	0.00360	36.27
Total de Linhas Retornadas	558	558
Tempo Total de Execução	29.425ms	2.029ms

Para a consulta apresentada na Figura 15, o resultado da aplicação dos testes sem a definição do índice e o resultado da aplicação dos testes após a definição do índice através do software JMeter são demonstrados na Tabela 11, no presente teste o objetivo foi identificar o desempenho do SGBD no processamento das consultas com a utilização de índices, foram escolhidas um número de amostras igual a 100 para a realização dos testes.

TABELA 11 - Comparativo sobre Índice através da ferramenta JMeter

	Padrão	Sintonia (Tuning)
Número de Amostras	100	100
Tempo Médio de Resposta	4627ms	1780ms
Mediana	4546ms	1929ms
Desvio-Padrão	1109	639
Throughput	794.597/minuto	1608.148/minuto

A Figura 12 ilustra os resultados obtidos referentes ao desempenho da consulta apresentada na Figura 11, primeiramente na sua forma padrão e posteriormente na sua forma modificada, os testes foram gerados através da utilização da ferramenta JMeter.

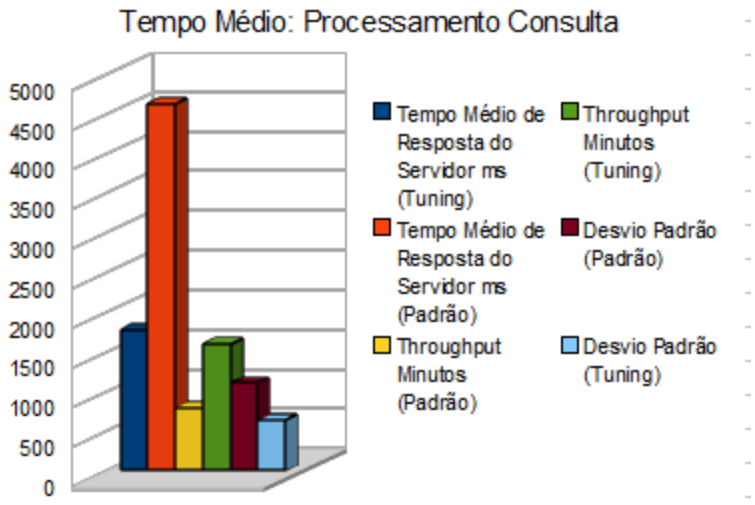


FIGURA 12 - Definição de Índice sobre a Coluna c_city

Nesta seção foi abordada a utilização de um típico especifico de índice, os índices seletivos, na próxima seção, é realizada uma abordagem sobre os índices não seletivos.

4.3.2 Definição de Índices Não Seletivos

A Figura 13 apresenta uma consulta realizada sobre a tabela customer, semelhante a consulta realizada na seção anterior 4.3.1, porém desta vez foi definido um novo índice, para uma coluna denominada não-seletiva.

A consulta é restringida pela coluna c_delivery da tabela, esta coluna possui uma baixa quantidade de valores distintos, inicialmente será apresentado o custo para a execução da consulta, sem a utilização de um índice sobre a coluna c_delivery, logo após um índice será criado sobre esse campo.

```
EXPLAIN ANALYZE SELECT *
FROM customer
WHERE c_delivery_cnt=1
```

FIGURA 13 - Consulta restringida pelo campo c_delivery_cnt

A Tabela 12 apresenta o plano de consulta padrão para a consulta apresentada na Figura 17 e o plano de consulta modificado, já utilizando o índice criado.

TABELA 12 - Plano da Consulta através do Comando EXPLAIN ANALYZE

	Padrão	Sintonia (Tuning)
Tipo de Acesso	Sequential Scan	Bitmap Index Scan
Custo Estimado	0.00310	201.81
Total de Linhas Retornadas	10.523	10.523
Tempo Total de Execução	28.321ms	14.211ms

Para a consulta apresentada na Figura 13, os resultados obtidos através do software JMeter são apresentados na Tabela 13.

TABELA 13 - Comparativo sobre Índice através da ferramenta JMeter

	Padrão	Sintonia (Tuning)
Número de Amostras	100	100
Tempo Médio de Resposta	9195ms	9206ms
Mediana	10000 ms	10001ms
Desvio-Padrão	1969	1957
Throughput	529.614 /minuto	539.084/minuto

A Figura 14 ilustra os resultados obtidos, no presente teste o objetivo foi identificar o desempenho do SGBD no processamento das consultas com a utilização de índices, foram escolhidas um número de amostras igual a 100 para a realização dos testes.

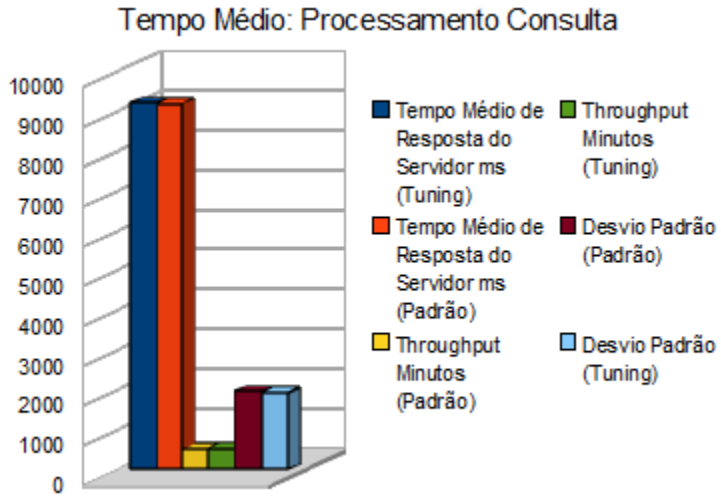


FIGURA 14 - Definição de Índice sobre a Coluna c_delivery_cnt

Nesta seção foi abordada a utilização de índices não seletivos, na próxima seção serão realizados testes que abrangem a decisão de aplicar técnicas de desnormalização nos dados.

4.3.3 Desnormalização

Para a execução deste teste, foi criada uma nova tabela chamada *customer_order*, ela é uma tabela resultante da desnormalização das tabelas *customer* e *oorder*, a Figura 15 apresenta uma consulta que primeiramente solicita dados provenientes destas duas tabelas.

```
EXPLAIN ANALYZE SELECT *
FROM customer c, oorder o
WHERE c.c_id = o.o_c_id
```

FIGURA 15 - Consulta que solicita junção de duas tabelas

A Tabela 14 apresenta o plano de consulta para a consulta apresentada na Figura 15, foi realizado um teste através da utilização do comando EXPLAIN ANALYZE, o tempo gasto pelo SGBD no processamento da consulta, e o acesso ao dados escolhido foram identificados, primeiramente foi realizado o teste realizando a junção das tabelas para obtenção dos dados e posteriormente foi realizado o teste com a tabela customer_order resultante da desnormalização das tabelas customer e oorder

TABELA 14 - Plano da Consulta através do Comando EXPLAIN ANALYZE

	Padrão	Sintonia (Tuning)
Tipo de Acesso	Sequential Scan	Sequential Scan
Custo Estimado	5528	0.1230
Total de Linhas Retornadas	72189	72189
Tempo Total de Execução	856.948ms	22.434ms

Os resultados obtidos através do software JMeter são apresentados na Tabela 15, no presente teste o objetivo foi identificar o desempenho do SGBD no processamento das consultas através da modificação do projeto realizando a desnormalização dos dados, foram escolhidas um número de amostras igual a 100 para a realização dos testes.

TABELA 15 - Comparativo sobre Desnormalização através da ferramenta JMeter

	Padrão	Sintonia (Tuning)
Número de Amostras	100	100
Tempo Médio de Resposta	10034ms	9583ms
Mediana	10015 ms	10015ms
Desvio-Padrão	45	1572
Throughput	56.492 /minuto	369.276/minuto

A Figura 16 ilustra os resultados obtidos no processamento da consulta apresentada na Figura 15, os resultados obtidos foram gerados através da utilização da ferramenta JMeter.

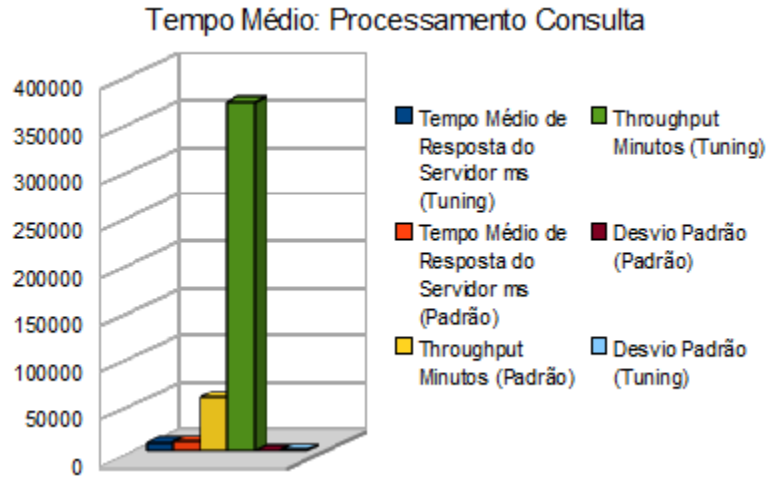


FIGURA 16 - Desnormalização

Nesta seção foi abordada a utilização da técnica de desnormalização nos dados, na próxima seção serão avaliados os distintos algoritmos de junção.

4.3.4 Junções

Na realização deste teste foi submetida a base de dados a seguinte consulta apresentada na Figura 17.

```
EXPLAIN ANALYZE SELECT  *
 FROM customer c,district d,oorder o
 WHERE c.c_id = o.o_c_id
 AND      d.d_id=c.c_d_id
```

FIGURA 17 - Consulta que solicita junção de duas tabelas

O objetivo do presente teste foi identificar através do comando EXPLAIN ANALYZE qual o melhor algoritmo de junção para o processamento da consulta apresentada na Figura 17, para testar os algoritmos de junção disponíveis foi realizada uma alteração nos parâmetros relativos aos algoritmos de junção do SGBD, desse modo o otimizador foi forçado a computar os custos com os três algoritmos de junções disponíveis nesse SGBD, foi observado neste teste que antes da modificação nos parâmetros o otimizador do PostgreSQL decidiu utilizar por padrão o algoritmo de junção Hash Join para processar a consulta, os resultados obtidos são apresentados na Tabela 16.

TABELA 16 - Plano da Consulta através do Comando EXPLAIN ANALYZE

	Hash Join	Merge Join	Nested Loop
Tipo de Acesso	Seq. Scan	Sequential e Index	Sequential e Index
Custo Estimado	6420.25	36941.35	0.644
Total de Linhas Retornadas	721.890	721.890	721.890
Tempo Total de Execução	1079.275	871.535	76899.21

Para a consulta apresentada na Figura 17, foram também realizados testes no software JMeter, os resultados obtidos são apresentados na Tabela 17.

TABELA 17 - Comparativo sobre Algoritmos de Junção através da ferramenta JMeter

	Hash Join	Merge Join	Nested Loop
Número de Amostras	100	100	100
Tempo Médio de Resposta	10258ms	10026ms	10928ms
Mediana	10011ms	10015ms	11026ms
Desvio-Padrão	398	29	273
Throughput	473.19/min	537.71 /min	123.7/min

A Figura 18 ilustra o tempo médio de resposta do servidor para solicitar a requisição referente à consulta apresentada na Figura 17.

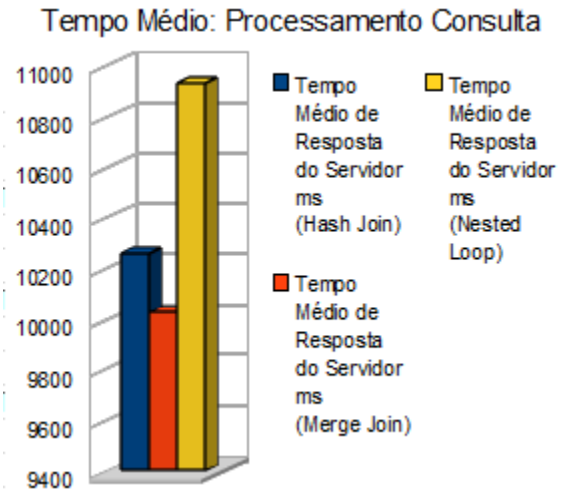


FIGURA 18 - Tempo de Resposta: Algoritmos de Junção

A Figura 19 ilustra o impacto ocasionado no throughput através da utilização de distintos algoritmos de junção para o processamento da consulta apresentada na Figura 17.

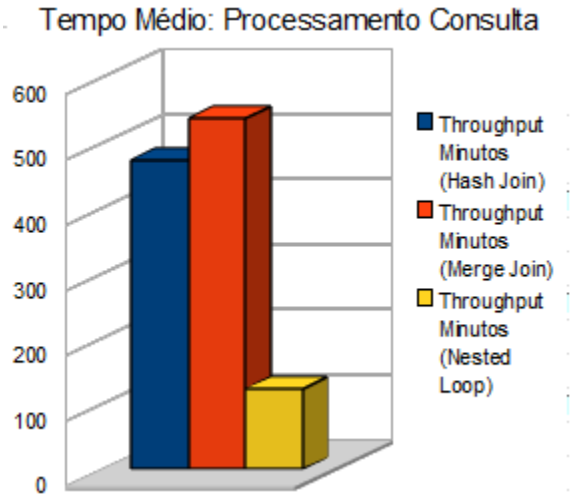


FIGURA 19 - Throughput: Algoritmos de Junção

É importante salientarmos, que a diferença acarretada no fator desempenho ocasionada na utilização dos três algoritmos, foi devido ao fato, que nesse em caso em específico, as características das tabelas utilizadas na junção, que já estavam ordenadas, devido a criação de índices, favoreceram o algoritmo Merge Join, uma vez que o algoritmo iria realizar a ordenação

implicitamente. Enquanto o tamanho das tabelas, apresentando uma grande quantidade de dados, desfavoreceram a varredura sequencial comumente realizada na utilização do algoritmo Nested Loop.

Nesta seção foi abordada a utilização de distintos algoritmos de junção, na próxima seção é avaliada a utilização de subconsultas.

4.3.5 Subconsultas

Para a realização deste teste foi submetida à base de dados a seguinte consulta apresentada na Figura 20 e logo em seguida foi submetida à consulta apresentada na Figura 21, ambas as consultas retornam os mesmos resultados, porém o plano de execução de ambas apresenta grandes diferenças como pode ser observado na Tabela 18.

```
EXPLAIN ANALYZE SELECT c_city
FROM customer c
WHERE NOT EXISTS
 (SELECT d.d_id from district d
 WHERE  d.d_id=2
 AND d.d_id=c.c_d_id)
```

FIGURA 20 - Subconsulta utilizando NOT EXISTS

```
EXPLAIN ANALYZE SELECT c_city
FROM customer c
WHERE c_d_id NOT IN
 (SELECT d.d_id
 FROM district d
 WHERE  d.d_id=2
 AND d.d_id=c.c_d_id)
```

FIGURA 21 - Subconsulta utilizando NOT IN

TABELA 18 - Plano da Consulta através do Comando EXPLAIN ANALYZE

	NOT EXISTS	NOT IN
Tipo de Acesso	Seq. e Bitmap Index	Seq. e Index Scan
Custo Estimado	8.33	0.1930
Total de Linhas Retornadas	27000	27000
Tempo Total de Execução	59.152ms	75.277ms

Para as consultas apresentadas nas Figuras 20 e 21, foram também realizados testes no software JMeter, os resultados obtidos são apresentados na Tabela 19.

TABELA 19 - Comparativo em Subconsultas através da ferramenta JMeter

	NOT EXISTS	NOT IN
Número de Amostras	100	100
Tempo Médio de Resposta	4ms	5ms
Mediana	1ms	1ms
Desvio-Padrão	10	9
Throughput	5639.098/minuto	5520.276/minuto

A Figura 22 ilustra os resultados obtidos no processamento das consultas apresentadas nas Figuras 20 e 21, através da utilização do software JMeter.

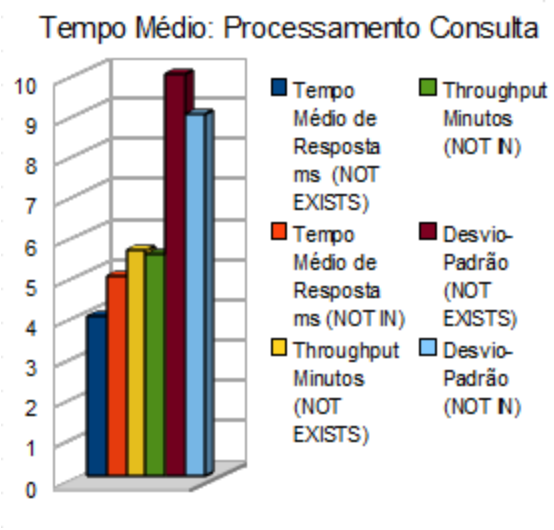


FIGURA 22 - Comparativo Subconsultas

Nesta seção foram realizados testes com a utilização de distintas formulações para subconsultas, na próxima seção é abordada a utilização de views.

4.3.6 View

Para a realização deste teste foi submetida à base de dados a seguinte consulta apresentada na Figura 23, posteriormente foi submetida a consulta apresentada na Figura 25.

```
EXPLAIN ANALYZE SELECT *
FROM customer c,district d
WHERE d.d_id=c.c_d_id
```

FIGURA 23 - Consulta sem View

Após a criação da view apresentada na Figura 24, foi submetida a consulta apresentada na Figura 25, para assim analisar a influência ocasionada no desempenho através da utilização de views.

```
CREATE TEMP VIEW j_view AS
SELECT *
FROM customer c,district d
WHERE d.d_id=c.c_d_id
```

FIGURA 24 - Criação da View

As consultas apresentadas nas Figuras 23 e 25 retornam os mesmos resultados, porém o plano de execução de ambas apresentou algumas particularidades, como pode ser observado na Tabela 20.

```
EXPLAIN ANALYZE SELECT *
FROM  j_view
```

FIGURA 25 - Consulta com View

TABELA 20 - Plano da Consulta através do Comando EXPLAIN ANALYZE

	Consulta sem View	Consulta Utilizando View
Tipo de Acesso	Sequential	Sequential
Custo Estimado	15.33	15.20
Total de Linhas Retornadas	30000	30000
Tempo Total de Execução	59.152ms	49.277ms

Para as consultas apresentadas nas Figuras 23 e 25, foram também realizados testes no software JMeter, os resultados obtidos são apresentados na Tabela 21.

TABELA 21 - Comparativo View através da ferramenta JMeter

	Padrão	VIEW
Número de Amostras	100	100
Tempo Médio de Resposta	4ms	3ms
Mediana	2ms	1ms
Desvio-Padrão	8	8
Throughput	5600.11/minuto	5612.88/minuto

A Figura 26 ilustra os resultados obtidos no processamento das consultas apresentadas nas Figuras 23 e 25, através da utilização do software JMeter.

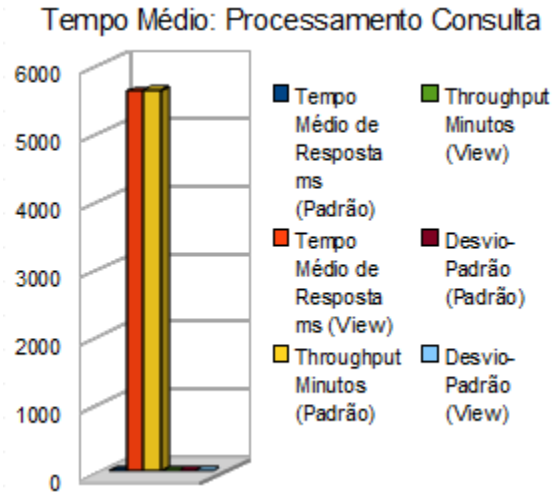


FIGURA 26 - Consulta com View

Nesta seção foi abordada a utilização de views, na próxima seção é abordada a utilização de Stored Procedures.

4.3.7 Utilização de Stored Procedure

Para a realização deste teste foi submetida à base de dados a seguinte consulta apresentada na Figura 27.

```
EXPLAIN ANALYZE
SELECT o_c_id
FROM oorder o, order_line ol
WHERE o.o_ol_cnt=ol.ol_number
AND ol_o_id=3862
```

FIGURA 27 - Consulta Padrão

Posteriormente foi decidido criar uma Stored Procedured para a consulta, onde a consulta apresentada na Figura 28 demonstra sua criação.

```
CREATE OR REPLACE FUNCTION
order_or_line ( )
returns SETOF INTEGER AS
'SELECT o_c_id
FROM oorder o, order_line ol
WHERE o.o_ol_cnt=ol.ol_number
AND ol_o_id=3862'
LANGUAGE 'SQL';
```

FIGURA 28 - Criação da Stored Procedure

A consulta apresentada na Figura 29, retorna o mesmo resultado que a consulta apresentada na Figura 27.

```
EXPLAIN ANALYZE
SELECT order_or_line ()
```

FIGURA 29 - Consulta utilizando Stored Procedures

Apesar de ambas as consultas retornarem o mesmo resultado, na Tabela 22 é possível observar as particularidades existentes entre o plano de consulta definido para cada consulta em específico.

TABELA 22 - Plano da Consulta através do Comando EXPLAIN ANALYZE

	Consulta Padrão	Stored Procedure
Tipo de Acesso	Sequential	Inexistente
Custo Estimado	19867.45	0.0026
Total de Linhas Retornadas	72.189	72.189
Tempo Total de Execução	450.845ms	359.86ms

Para as consultas apresentadas nas Figuras 27 e 29, foram também realizados testes no software JMeter, os resultados obtidos são apresentados na Tabela 23.

TABELA 23 - Comparativo Junção através da ferramenta JMeter

	Consulta Padrão	Stored Procedures
Número de Amostras	100	100
Tempo Médio de Resposta	9556ms	9224ms
Mediana	100000ms	100000ms
Desvio-Padrão	1637	1508
Throughput	512.25/minuto	525.88/minuto

A Figura 30 ilustra os resultados obtidos no processamento das consultas apresentadas nas Figuras 27 e 29, através da utilização do software JMeter.

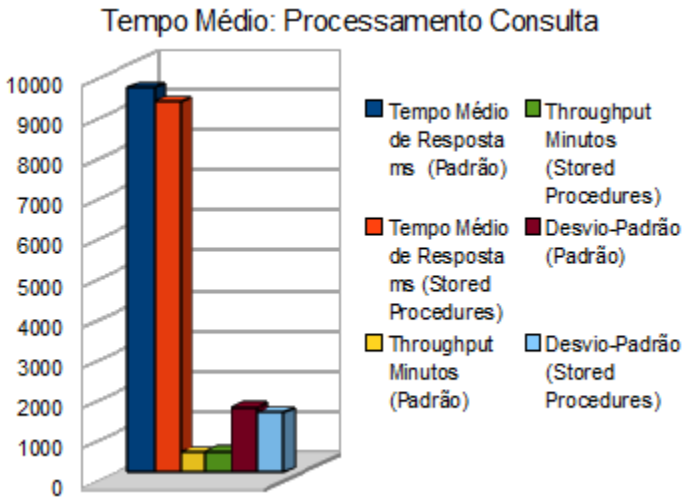


FIGURA 30 - Consulta com Stored Procedures

Esta seção encerrou a demonstração dos experimentos realizados no desenvolvimento do presente trabalho de graduação, a próxima seção realiza uma comparativo deste trabalho com os trabalhos relacionados apresentados no capítulo 3.

4.4 Comparativo: Trabalhos Relacionados

Esta seção tem a finalidade de realizar um comparativo sobre o presente trabalho de graduação e os trabalhos relacionados apresentados no capítulo 3, como foram abordadas no desenvolvimento deste trabalho as técnicas existentes para a realização de sintonia por parte de software, foi decidido utilizar no comparativo os três tipos de ações de sintonia existentes por parte de software, assim como também o SGBD escolhido em cada trabalho, a Tabela 24

apresenta os resultados do comparativo realizado e posteriormente há uma descrição das variáveis utilizadas nos critérios de comparação apresentados pela tabela.

TABELA 24 - Comparativo Trabalhos Relacionados

	Critério 1	Critério 2	Critério 3	Critério 4
(IOANNIDIS,1996)	X			Genérico
(OH e LEE, 2005)		X		Oracle
(PARRA,2005)	X	X	X	Oracle
(RODD e KULKRANI,2010)		X		Oracle
(CHEN et al., 2011)		X		Genérico
(Presente Trabalho)	X	X		PostreSQL

Onde, na Tabela 24, o critério 1 verifica se o presente trabalho abordou o refinamento do esquema das relações e consultas, o critério 2 verifica se o presente trabalho abordou a configuração dos parâmetros do SGBD, o critério 3 verifica se o presente trabalho abordou a configuração do sistema operacional e por fim o critério 4 verifica qual foi o SGBD utilizado para a aplicação das técnicas de sintonia existentes.

Em relação ao trabalho de (IOANNIDIS,1996), o presente trabalho possui em comum as ações de refinamento do esquema das relações e consultas, especificamente a análise e otimização de consultas SQL para a elaboração de distintos planos de consulta, o presente trabalho apresenta como contribuição em relação a este trabalho, a utilização do SGBD PostgreSQL para realizar a otimização de consultas SQL, apresentando suas potencialidades e técnicas existentes para a elaboração de consultas que aproveitam de uma melhor forma os recursos providos pelo SGBD.

Em relação aos trabalhos de (OH e LEE,2005) e (RODD e KULKRANI,2010), o presente trabalho possui em comum a ação de ajustes na configuração dos parâmetros e apresenta como contribuição a ação de refinamento do esquema das relações e consultas através da utilização do SGBD PostgreSQL.

Em relação ao trabalho de (PARRA,2005), o presente trabalho possui em comum as ações de refinamento do esquema das relações e consultas e configuração dos parâmetros do SGBD, o presente trabalho apresenta como diferencial a utilização do SGBD PostgreSQL para realização das ações de sintonia, a tabela 23 demonstra tal comparativo.

Em relação ao trabalho de (CHEN et al., 2011), o presente trabalho possui em comum as ações de refinamento do esquema das relações e consultas, especificamente a análise de operações de junção, porem o trabalho de (CHEN et al., 2011), no processo de análise das distintas operações de junção existentes, no impacto ocasionado no desempenho em relação a

ordem das tabelas em que são realizadas operações de junção, propõe um algoritmo diferenciado para ser utilizado pelos otimizadores dos SGBD, enquanto no presente trabalho é abordado o impacto que os distintos algoritmos de junção ocasionam no SGBD PostgreSQL.

Esta seção apresentou os experimentos realizados durante o desenvolvimento do presente trabalho e apresentou um comparativo deste trabalho com os trabalhos relacionados no capítulo 3, o próximo capítulo apresenta a análise dos resultados dos experimentos realizados no presente capítulo.

5. ANÁLISE DOS RESULTADOS

Nesta seção, foi realizada uma análise a partir dos resultados obtidos com a realização dos experimentos apresentados no capítulo 4, a análise foi inferida seguindo alguns critérios, sendo que conforme (MANNINO, 2008) para obter melhorias no desempenho de um SGBD específico é necessário estudar e entender as suas opções atentamente, posteriormente foi analisada a medida de desempenho de cada instrução, conforme (CLICIA, 2011) nos sistemas de computadores o desempenho pode ser definido de maneiras diferentes, podemos considerar como o sistema de melhor desempenho aquele que executa em menor tempo uma determinada tarefa, ou ainda, aquele sistema que executa o maior número de tarefas num determinado intervalo de tempo. Dessa forma, estamos considerando dois aspectos diferentes para avaliar o desempenho. O primeiro é o tempo de resposta ou tempo de execução da aplicação. O segundo é a avaliação da própria documentação do SBGB para a aplicação em questão.

No presente trabalho para medição de desempenho, foi aplicada a primeira abordagem, onde o melhor desempenho é aquele que executa em menor tempo uma determinada tarefa, onde sua fórmula é apresentada a seguir: Desempenho de A = 1/ Tempo de Execução de A.

Os resultados inferidos para todos os experimentos realizados no capítulo 4, são apresentados a seguir.

5.1 Otimização na Configuração dos Parâmetros do SGBD

Na otimização realizada nos parâmetros do SGBD, foi possível observar que ao utilizar o SGBD para a realização dos testes com os parâmetros nas suas versões originais, o mesmo não aproveitava de forma correta os recursos de hardware disponível, tal constatação é fundamentada de acordo com (POSTGRESQL, 2011), onde os parâmetros nas suas configurações originais pressupõe um cálculo de porcentagem em relação ao hardware utilizado, é importante salientar que não é fornecido um valor exato para a alteração dos parâmetros, mas de acordo com as recomendações para um servidor de banco de dados dedicado com 1 Gigabyte (GB) de RAM ou mais, um valor razoável de partida para shared_buffers é de 25% da memória do sistema, sendo a mesma análise aplicada aos outros parâmetros, logo foi verificado que a alteração nos parâmetros apresentou melhorias significativas, uma vez que o hardware utilizado suportava um valor um pouco mais alto que o valor padrão.

Foram submetidas a base de dados, consultas de inserção antes da alteração dos parâmetros, onde o valor de desempenho foi 0.76 ms, após a alteração o valor passou para 3.2 ms, para as consultas de deleção o valor encontrado foi 0.009 ms, após a alteração o valor passou para 0.010 ms, para as consultas de atualização o valor encontrado foi 1.31 ms, após a alteração

passou para 12.9 ms, para as consultas de recuperação de informação o valor encontrado foi 0.008 ms, após a alteração o valor passou para 0.010 ms.

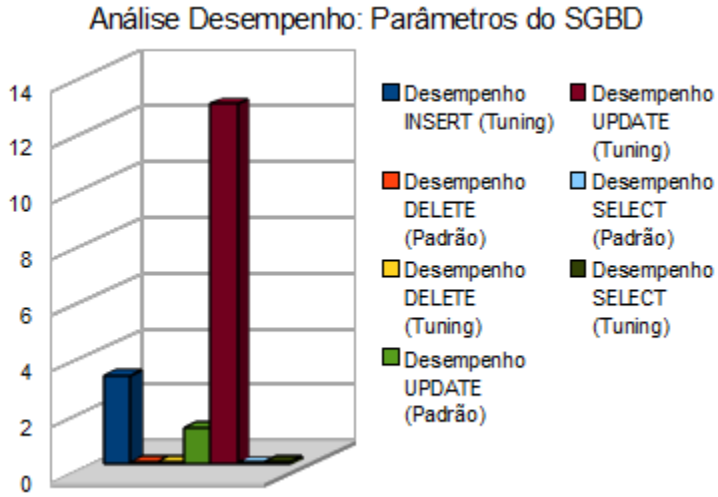


FIGURA 31 - Análise Desempenho: Parâmetros

Onde, na Figura 31 foi ilustrado o impacto ocasionado no desempenho, no momento que antecede a alteração dos parâmetros e posteriormente com os parâmetros já alterados.

Esta seção apresentou a análise realizada sobre a configuração dos parâmetros do SGBD, primeiramente na sua versão original e posteriormente com os valores alterados para o presente hardware em que foi desenvolvido este trabalho de graduação.

5.2 Definição de Índices Seletivos

No presente teste foi possível observar que na utilização do índice criado para a consulta apresentada na Figura 11 no capítulo 4, o mesmo possibilitou uma grande melhoria no desempenho, uma vez que a coluna na qual o índice foi criado possuía uma grande quantidade de valores distintos, isso possibilitou a criação de um índice bastante seletivo.

Na utilização do índice para a solicitação dos dados que referenciavam a coluna do índice, foi observado que no plano de consulta o tipo de varredura foi modificado, o otimizador do PostgreSQL não realizou uma varredura do tipo Sequential Scan onde se percorre todas as linhas de uma tabela, e, para cada linha, verifica se a condição é respeitada, invés disso ele utilizou o índice criado, minimizando o tempo total de execução da consulta. O valor encontrado para o desempenho da consulta apresentada antes da criação do índice foi 0.0339 ms, após a criação do índice o valor passou para 0.492 ms, a Figura 32 ilustra tais resultados.

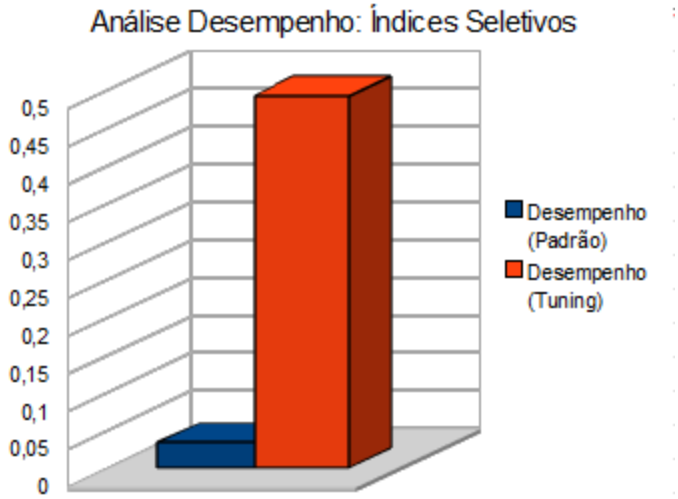


FIGURA 32 - Análise Desempenho: Índices Seletivos

Esta seção apresentou a análise sobre índices seletivos, na próxima seção é analisada a utilização de índices não seletivos.

5.3 Definição de Índices Não Seletivos

No presente teste foi possível observar que na utilização do índice criado para a consulta apresentada na Figura 13 no capítulo 4, como o índice era um índice menos seletivo que o índice criado para o campo c_city apresentado na consulta anterior, devido a existência de um número reduzido de valores distintos no campo , podemos observar que o tempo total de execução também foi minimizado para este caso, porém, numa proporção inferior ao índice criado para a consulta anterior, que possuía uma grande quantidade de valores distintos, para este caso foi analisado que tal índice ao se analisar a relação custo beneficio que ele propicia pode não ser recomendado, uma vez que a alteração em uma coluna indexada resulta em trabalho adicional para atualizar também os seus índices.

O valor encontrado para o desempenho da consulta apresentada antes da criação do índice foi 0.0335 ms, após a criação do índice o valor passou para 0.070 ms, a Figura 33 ilustra tais resultados.

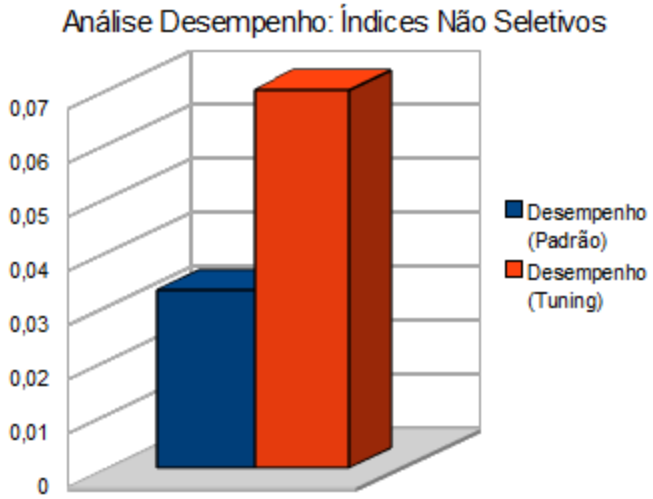


FIGURA 33 - Análise Desempenho: Índices Não Seletivos

Esta seção apresentou a análise sobre índices não seletivos, na próxima seção é analisado o processo de desnormalização dos dados.

5.4 Desnormalização

Na decisão de aplicar a desnormalização nos dados provenientes das tabelas, foi possível notar que o benefício obtido com a criação da nova tabela proveniente da desnormalização é que ela eliminou a necessidade de junção entre duas tabelas para fornecer o retorno dos dados, como a operação de junção é considerada uma operação muito custosa para o SGBD, o que foi inferido nesta análise é que a decisão de aplicar o processo de desnormalização nos dados, deve ser realizada quando a característica da utilização das tabelas em que os dados foram desnormalizados seja somente leitura, uma vez que a operação de inserção em dados desnormalizados é bem mais complexa.

O valor encontrado para o desempenho da consulta apresentada antes da desnormalização foi 0.0011 ms, após a desnormalização o valor passou para 0.044 ms, a Figura 34 ilustra tais resultados.

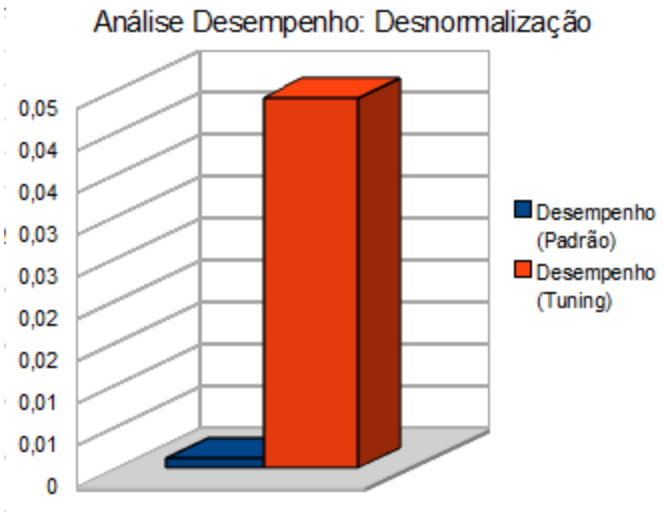


FIGURA 34 - Análise Desempenho: Desnormalização

Esta seção apresentou a análise sobre o processo de desnormalização, na próxima seção é analisada a utilização de distintos algoritmos de junção.

5.5 Junção

Na utilização de algoritmos de junção distintos, foi possível notar que há um grande diferença em relação aos custos e o tempo total de execução conforme o tipo de algoritmo de junção utilizado, para este caso o algoritmo de junção que apresentou o maior custo foi o algoritmo Merge Join, tal custo pode ser entendido ao analisar-se o funcionamento do algoritmo Merge Join, que ordenada as tabelas de junção quando elas não estão previamente ordenadas e isso ocasiona um acréscimo no custo.

Ao utilizarmos o algoritmo de junção Nested Loop as pesquisas lineares foram substituídas por pesquisas em índices, no presente algoritmo para cada registo exterior, o índice foi utilizado para encontrar os registos interiores que satisfaziam a condição de seleção e foram comparados de forma unitária.

O valor encontrado para o desempenho da consulta apresentada com a utilização do algoritmo Merge Join foi 0.0011 ms, na utilização do algoritmo Hash Join o valor encontrado foi 0.00009.26 ms, o valor encontrado na utilização do algoritmo Nested Loop foi 0.000001.30 ms, a Figura 35 ilustra tais resultados.

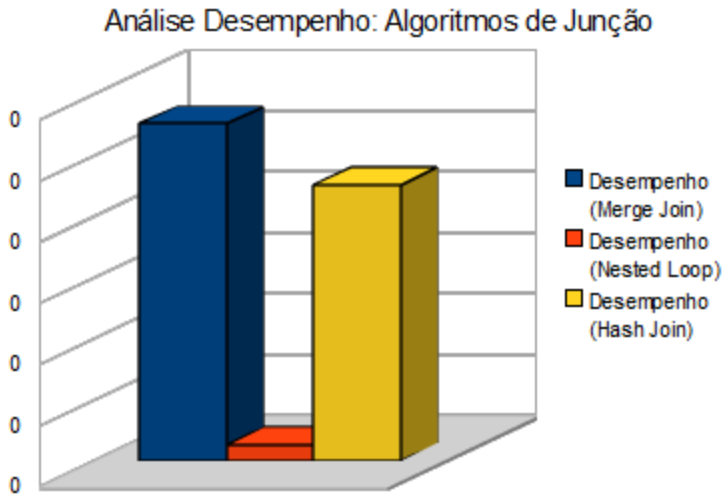


FIGURA 35 - Análise Desempenho: Junção

Esta seção apresentou a análise sobre distintos algoritmos de junção, na próxima seção é analisada a utilização de subconsultas.

5.6 Subconsultas

O otimizador do PostgreSQL adotou planos diferentes para executar as subconsultas e na utilização do operador NOT IN, para que fossem retornadas as linhas solicitadas, o otimizador realizou uma varredura sequencial na tabela e realizou um filtro em comparação com as linhas visitadas na outra tabela, porém esta operação demonstrou ser uma operação lenta e se compararmos o tempo total de execução de ambas as subconsultas podemos visualizar que para este caso a utilização do operador NOT EXISTS mostrou-se mais eficaz.

O valor encontrado para o desempenho da consulta apresentada através da utilização do predicado NOT EXISTS foi 0.016 ms, na utilização do predicado NOT IN o valor passou para 0.013 ms, a Figura 36 ilustra tais resultados.

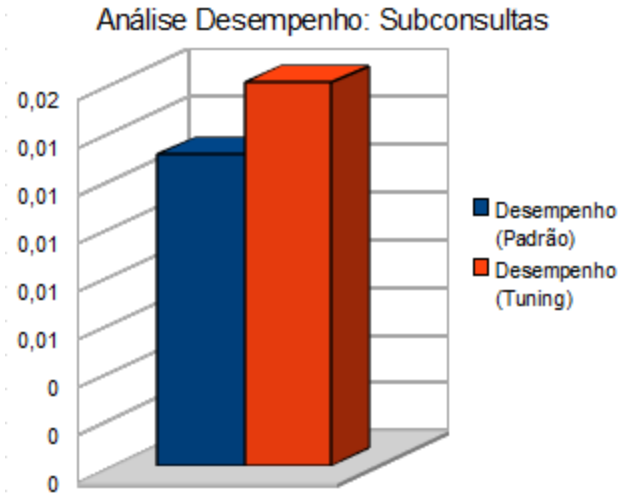


FIGURA 36 - Análise Desempenho: Subconsultas

Esta seção apresentou a análise sobre a utilização de subconsultas, na próxima seção é analisada a utilização de views.

5.7 View

Na consulta solicitada utilizando view na Figura 25, foi apresentada uma melhoria no desempenho em relação a solicitação da consulta sem a utilização da view, porem, de acordo com (MANNINO,2008) é necessário que haja um cuidado extra na utilização de view, pois na utilização de views complexas, sua utilização pode prejudicar significativamente o desempenho, ou seja antes de utilizar views deve ser realizada uma comparação com a solicitação da consulta diretamente da tabela base, na utilização da view para a realização do presente trabalho, a view não degradou o desempenho, podendo ser utilizada para o caso especifico em que ela foi aplicada.

O valor encontrado para o desempenho da consulta apresentada sem a utilização da view foi 0.016 ms, com a utilização da view o valor passou para 0.020 ms, a Figura 37 ilustra tais resultados.

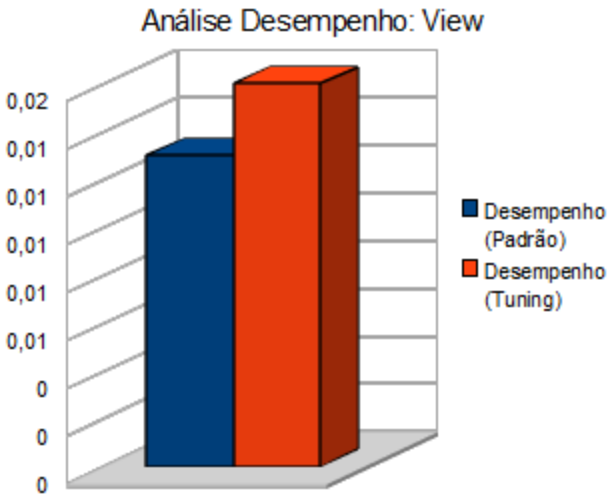


FIGURA 37 - Análise Desempenho: View

Esta seção apresentou a análise sobre a utilização de views, na próxima seção é analisada a utilização de stored procedures.

5.8 Stored Procedures

Na consulta solicitada utilizando Stored Procedures na Figura 29, foi apresentada uma melhoria no desempenho em relação a solicitação da consulta sem a utilização dessa funcionalidade, tal fato é explicado de acordo com a própria definicação de Stored Procedure, que conforme (POSTGRESQL, 2011) a utilização de Stored Procedures, na verdade é a utilização de módulos de programa armazenados pelo SGBD no servidor de banco de dados, sendo que executar um programa no servidor pode reduzir a transferência de dados e consequentemente o custo de comunicação entre cliente e servidor.

O valor encontrado para o desempenho da consulta apresentada sem a utilização da Stored Procedures foi 0.0022 ms, com a utilização da Stored Procedures o valor passou para 0.0027 ms, a Figura 38 ilustra tais resultados.

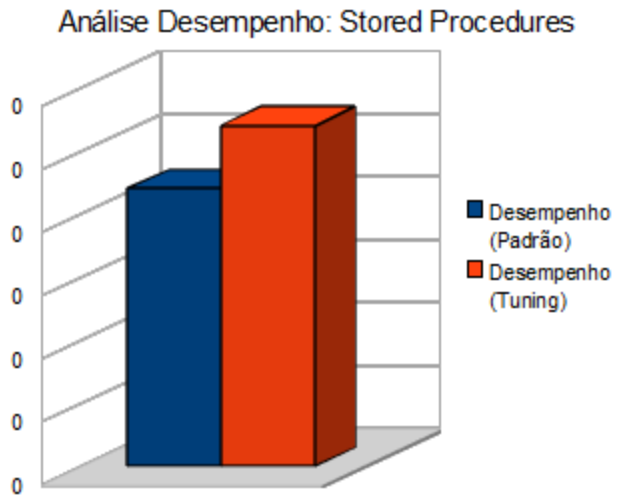


FIGURA 38 - Análise Desempenho: Stored Procedures

A partir deste ponto, é importante salientarmos que, todas os testes realizados tem o intuito de identificar no SGBD PostgreSQL, quais os recursos existentes para a realização de sintonia. Logo, para que o trabalho tenha alcançado o seu objetivo, espera se que nesse exato momento, o leitor, já tenha compreendido como realizar a busca para detecção de possíveis problemas de performance no PostgreSQL, as alternativas de melhorias que o SGBD disponibiliza, e as ferramentas e comandos utilizados, para testes de verificação de melhoria de desempenho.

Para que seja mais fácil visualizar o impacto ocasionado no desempenho de uma forma geral, a Tabela 25, apresenta os principais recursos, que foram utilizados no SGBD PostgreSQL, e o impacto ocasionado no fator desempenho.

TABELA 25 - Comparativo Geral de Desempenho

	Desempenho (Padrão)	Desempenho (Tuning)
Índices Seletivos	0.0339 ms	0.492 ms
Índices Não Seletivos	0.0335 ms	0.070 ms
Desnormalização	0.0011 ms	0.044 ms
Subconsultas	0.013 ms	0.016 ms
View	0.016 ms	0.020 ms
Stored Procedures	0.0022 ms	0.002 ms

Esta seção, encerra a análise sobre os experimentos demonstrados no capitulo 4, a próxima seção apresenta as considerações finais do presente trabalho.

6 CONSIDERAÇÕES FINAIS

Este capítulo conclui a apresentação deste trabalho. Na seção 5.1 são citadas as principais contribuições do trabalho realizado. Finalmente, a seção 5.2 propõe um trabalho futuro como forma de continuação do presente trabalho.

6.1 Principais Contribuições

No presente trabalho, observou-se que a realização de sintonia em sistemas de Banco de Dados é uma atividade que pode ser realizada em diferentes fases do ciclo de vida das aplicações de sistemas de Banco de Dados, tal atividade pode ser realizada tanto em nível de hardware quanto em nível de software.

Neste trabalho, a principal contribuição, foram as investigações das formas de sintonia existentes por parte de software, no SGBD PostgreSQL, para tal foram concretizadas as ações de refinamento do esquema das relações e consultas. Onde, também, foram realizadas alterações na configuração dos parâmetros do SGBD, a simulação foi realizada sobre a base de dados da TPC-C, onde na análise foi observada a influência sobre as diferentes formas de esquemas e consultas, onde foi observado o impacto que os experimentos em geral, acarretariam no SGBD PostgreSQL.

Na realização deste trabalho procurou-se descrever de forma minuciosa todos os passos realizados para a concretização do mesmo e os resultados obtidos foram apresentados no capitulo 5, com isso espera-se que o presente trabalho possa auxiliar nos conceitos que envolvem a compreensão das atividades de sintonia realizadas em sistemas de Banco de Dados e colaborar no estudo sobre os fatores que influenciam no desempenho de sistemas de Banco de Dados, especificamente no SGBD PostgreSQL.

6.2 Trabalhos Futuros

No contexto atual os SGBD, não são sistemas totalmente automatizados e independentes dos seus usuários para a realização de atividades de sintonia. O estudo sobre o gerenciamento autônomo desses sistemas, capazes de realizar atividades de sintonia, é um campo que apresenta constante crescimento na área de Banco de Dados, trabalhos como o de (WIESE e RABINOVITCH, 2009) realizam estudos sobre uma arquitetura baseada no conhecimento necessário para a construção de mecanismos de gerenciamento autônomo capazes de realizar atividades de sintonia.

Logo, uma sugestão é o desenvolvimento de aplicativos que auxiliem os DBAs na administração de tais sistemas, uma vez que o serviço de monitoramento em banco de dados é um serviço 24x7, ou seja, é exigido do DBA uma dedicação exclusiva de 24 horas, durante os 7 dias da semana, no que se refere aos serviços de monitoramento que visam propiciar a manutenção e um melhor desempenho na base de dados, uma alternativa de estudo é promover o desenvolvimento de um aplicativo para uso dos DBAs em dispositivos móveis, a justificativa da escolha da utilização de tecnologia móvel é oriunda do fato deste tipo de tecnologia permitir o acesso aos dados e as informações em qualquer momento e em qualquer lugar, ou seja, acredita-se que ela pode auxiliar de forma benéfica a rotina dos DBAs devido a essa característica de mobilidade que ela propicia.

7 REFERENCIAS

ACESSASP, Metrô de SP economiza U$$ 900 mil com software livre, 2007. Disponível em: <http://www.acessasp.sp.gov.br/modules/news/article.php?storyid=326>. Acesso em: 03 nov.2011, 15:30.

AHRENDT, Eric. Bancos de dados extremos: Os maiores e mais rápidos, 2010. Disponível em: <http://www.ibm.com/developerworks/br/data/library/dmmag/DBMag_2010_Issue1/DBMag_Issue109_Extreme/>. Acesso em: 03 nov.2011, 15:30.

ANDRADE, L. D. Otimização de Consultas de Aplicações T-SQL em Ambiente SQL Server 2000, 2005. 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: 05 maio 2011, 11:00.

APACHE, The Apache Software Foundation. Apache JMeter, 2011. Disponível em: < http://jmeter.apache.org/>. Acesso em: 05 maio 2011, 11:00.

AVANZI, Edson. Estendendo UML profile de banco de dados para projeto físico, 2006. Disponível em: < https://www.unimep.br/phpg/bibdig/aluno/down. php?cod=74>. Acesso em: 05 nov.2011, 17:00.

BAPTISTA, Cláudio. Administração de Sistemas de Gestão de Banco de Dados, 2008.

CARATTI, Ricardo. Monitoração de Sistemas de Banco de Dados. SQL Magazine, Rio de Janeiro: DevMedia, p. 32-35, 2004.

CARNEIRO, Alessandro; MOREIRA, Julinao; FREITAS, André. TUNING - Técnicas de Otimização de Banco de Dados: Um Estudo Comparativo: Mysql e Postgresql, V Escola Regional de Banco de Dados, 2009.

CASTOLDI, Leandro. Sintonia em Banco de Dados sob diferentes sistemas de arquivos, 2005.

CHEN, Yongheng; ZUO, Wanli; HE, Fenglin; CHEN, Kerui. Optimizing Large Query by Simulated Annealing Algorithm Based On Graph-Based Approach, Journal of Software, vol. 6, n. 9, 2011.

CLICIA, Maria. Avaliação de Desempenho, 2011. Disponível em:< http://www.ime.uerj.br/professores/Mariaclicia/Oc1/Cap3_desempenho.pdf>. Acesso em: 07 jun. 2011, 14:30.

COLARES, Flávio. Análise Comparativa de Banco de Dados Gratuitos, Monografia apresentada à coordenação do curso de Ciências da Computação da Faculdade Lourenço Filho, como requisito para obtenção do grau de Bacharel em Ciência da Computação, 2007.

DI LELLO, João; COUTINHO, Marcelo. Avaliação de Consultas- Capítulo 12 Resumo, 2005. Disponível em: <http://www.ic.unicamp.br/~geovane/mo410-091/Ch12-Resumo.pdf>. Acesso em: 03 jun.2011, 15:30.

ELMASRI, Ramez; NAVATHE, Shamkant. Sistemas de Banco de Dados - Fundamentos e Aplicações. Americana: LTC, 2002.

FIREBIRD, About Firebird, 2011. Disponível em: <http://www.firebirdsql.org/en/about-firebird/>. Acesso em: 03 jun.2011, 15:30.

HEUSER, Carlos. Projeto de Banco de Dados. Porto Alegre: Sagra Luzzato, 1998.

HIRATSUKA Eleni. Você é um Tuner?, 2009. Disponível em: <http://www.batebyte.pr.gov.br/modules/conteudo/conteudo.php?conteudo=1681> . Acesso em: 20 jun. 2011, 11:00.

IOANNIDIS, Yannis E. Query Opitimization, Journal ACM Computing Surveys, Volume 28 n.1, 1996.

LELIS H.; Boas Praticas De Programação PLSQL - PARTE 1, 2011 <http://www.profissionaloracle.com.br/modules. php?name=News&file=print&sid=3>. Acesso em: 03 jun.2011, 16:00.

LIFSCHITZ, Sérgio; MILANÉS, Anolan; SALLES, Marcos. Estado da Arte em Auto-Sintonia de SGBD Relacionais, 2004. Disponível em: <http://www.dbis.ethz.ch/people/vmarcos/LMS04.pdf>. Acesso em: 03 jun. 2011, 16:30.

MANNINO, Michael. Projeto, Desenvolvimento de Aplicações & Administração de Banco de Dados. São Paulo: McGraw-Hill, 2008.

MEDEIROS, Anderson. Manipulando banco de dados no PostgreSQL sem fazer uso de comandos SQL, 2008. Disponível em: <http://www.clickgeo.com.br/PostgreSQLsemusarSQL.pdf>. Acesso em: 22 jun. 2011, 13:20.

MONTEIRO José; LIFSCHITZ Sérgio; BRAYNER Ângelo. Extraindo Metadados de SGBDs, Monografias em Ciência da Computação, Pontífica Universidade Católica do Rio de Janeiro 2007;

MOURÃO, André; COSTA, Eduardo; MARQUES, Ricardo. Trabalho Final- PostgreSQL 2010. Disponível em:< http://ssdi.di.fct.unl.pt/sbd/1011/func/trabalho/entregues/files/G10_relatorioBD.pdf>. Acesso em: 23 jul. 2011, 11:00.

MYSQL, MySQL Community Edition, 2011. Disponível em:< http://www.mysql.com/products/community/>. Acesso em: 12 jul. 2011, 15:00.

OH, Jeong Seok; LEE, Sang Ho. Resource Selection for Autonomic Database Tuning. Proceedings of the 21st International Conference on Data Engineering (ICDE'05), 2005.

O'NEIL, Patrick, The set query benchmark. The Benchmark Handbook. 1993.

OPENGEO, Evento no Quartel General do Exército irá abordar Software Live para Geotecnologias, 2008. Disponível em: < http://www.geolivre.org.br/?q=node/558>. Acesso em: 03 set. 2011, 12:00.

ORACLE Corporation, Oracle8 Tuning, 1997. Disponível em: < http://docs.oracle.com/cd/A64702_01/doc/server.805/a58246.pdf>. Acesso em: 01 out. 2011, 12:00.

PADLIPSKAS, S.; Tuning – Técnica de Reescrita de Consultas, 2011. Disponível em: <http://www.devmedia.com.br/post-4501-Tuning-tecnica-de-Reescrita-de-Consultas.html>. Acesso em: 03 jun.2011, 12:00.

PARRA, Leonardo. Tuning de Desempenho em Banco de Dados, 2005. Disponível em: < www2. dc.uel.br/nourau/document/?down=568>. Acesso em: 10 jun. 2011, 11:00.

PGBR, Conferência Brasileira de PostgreSQL, 4º Evento Brasileiro de PostgreSQL, 2011. Disponível em: < http://pgbr.postgresql.org.br/2011/capt.pt.pdf >. Acesso em: 15 out. 2011, 11:00.

POLEPOSITION, PolePositon, 2011. Disponível em: < http://polepos.org/>. Acesso em: 10 jun. 2011, 11:00.

POSTGRESQL, The PostgreSQL Global Development Group. PostgreSQL 9.1.2 Documentation, 2011. Disponível em:< http://www.postgresql.org/docs/current/static/>. Acesso em: 23 jul 2011, 11:00.

RAMALHO, José Antonio. ORACLE 9i. São Paulo: Berkeley Brasil, 2002.

REZENDE, Ricardo. Álgebra Relacional – Parte II Linguagem de consulta formal, 2004. Disponível em: <http://www.sqlmagazine.com.br/Colunistas/RicardoRezende/05_AlgebraRelacional_P2.asp> Acesso em: 29 abr. 2011, 11:30.

RODD,S. F., KULKRANI U. P., Adaptive Tuning Algorithm for Performance tuning of Database Management System, International Journal of Computer Science and Information Security, Vol. 8, No. 1, 2010.

SANTOS, Francisco. Optimização de Interrogações no PostgreSQL, 2009. Disponível em: < https://dspace.ist.utl.pt/bitstream/2295/51822/1/TDPostgreSQLProcOptimInterrogacoesFinal.doc >. Acesso em: 12 nov 2011, 14:00.

SCARABELIN, Fernando; DIGIAMPIETRI, Luciano. Uso de índices e Visões para a Otimização de Banco de Dados. In: 18º Simpósio de Iniciação Científica da USP (SIICUSP), 2010, São Paulo, Brasil. Anais do 18º Simpósio de Iniciação Científica da USP, 2010.

SILBERSCHATZ, Abraham; KORTH, Henry; SUDARSHAN, S. Sistemas de Bancos de Dados, São Paulo: Makron Books, 1999.

SILVA, Maria; TEIXEIRA, Rivanda. Gerenciamento da tecnologia da informação para tomada de decisão em supermercados, Revista de Ciências da Administração, Florianópolis, v.4, n.6, p.69-80, 2002;

SOUZA, Luis. Uma análise de tipos de consultas e esquemas para realização de sintonia em banco de dados, 2005. Disponível em: < http://www.cin.ufpe.br/~tg/2005-1/>. Acesso em: 03 jul. 2011, 12:30.

SOURCEFORGE, BenchmarkSQL, 2011. Disponível em: <http://sourceforge.net/projects/benchmarksql/>. Acesso em: 02 nov. 2011, 12:00.

TRAMONTINA, Gregório. Database Tuning: Configurando o Interbase e o PostgreSQL. Campinas, 2008. Disponível em: <http://www.ic.unicamp.br/~geovane/mo410-091/Ch20-ConfigInterbasePosgres-art.pdf >. Acesso em: 29 abr. 2011, 11:30.

.

TPC, TPC BENCHMARK™C Standard Specification Revision 5.11, 2010. Disponível em: < http://www.tpc.org/tpcc/spec/tpcc_current.pdf>. Acesso em: 29 out. 2011, 11:30.

UVA, Universidad de Valladolid. TPCC-UVa: A free, open-source implementation of the TPC-C Benchmark, 2011. Disponível em: < http://www.infor.uva.es/~diego/tpcc-uva.html>. Acesso em: 29 out. 2011, 11:30.

VIEIRA, Marina; Projeto de Banco de Dados, <http://pt.scribd.com/doc/6795108/2/Processo-de-Projeto-de-Banco-de-Dados>< Acesso em: 03 jun.2011>

WEBER, Flavio; ANGELOTTI, Elaini. Otimizando Consultas SQL no ambiente SQLServer, 2009. Disponível em:< http://www.batebyte.pr.gov.br/modules/conteudo/conteudo.php?conteudo=1184 >. Acesso em: 22 jul 2011, 11:00.

APÊNDICE - A

AMOSTRA DE TRANSAÇÕES TPC-C - NEW-ORDER

```
int neword()
{
EXEC SQL WHENEVER NOT FOUND GOTO sqlerr;
EXEC SQL WHENEVER SQLERROR GOTO sqlerr;
gettimestamp(datetime);
EXEC SQL SELECT c_discount, c_last, c_credit, w_tax
INTO :c_discount, :c_last, :c_credit, :w_tax
FROM customer, warehouse
WHERE w_id = :w_id AND c_w_id = w_id AND
c_d_id = :d_id AND c_id = :c_id;
EXEC SQL SELECT d_next_o_id, d_tax INTO :d_next_o_id, :d_tax
FROM district
WHERE d_id = :d_id AND d_w_id = :w_id;
EXEC SQL UPDATE district SET d_next_o_id = :d_next_o_id + 1
WHERE d_id = :d_id AND d_w_id = :w_id;
o_id=d_next_o_id;
EXEC SQL INSERT INTO ORDERS (o_id, o_d_id, o_w_id, o_c_id,
o_entry_d, o_ol_cnt, o_all_local)
VALUES (:o_id, :d_id, :w_id, :c_id,
:datetime, :o_ol_cnt, :o_all_local);
EXEC SQL INSERT INTO NEW_ORDER (no_o_id, no_d_id, no_w_id)
VALUES (:o_id, :d_id, :w_id);
for (ol_number=1; ol_number<=o_ol_cnt; ol_number++)
{
ol_supply_w_id=atol(supware[ol_number-1]);
if (ol_supply_w_id != w_id) o_all_local=0;
ol_i_id=atol(itemid[ol_number-1]);
ol_quantity=atol(qty[ol_number-1]);
EXEC SQL WHENEVER NOT FOUND GOTO invaliditem;
EXEC SQL SELECT i_price, i_name , i_data
INTO :i_price, :i_name, :i_data
FROM item
price[ol_number-1] = i_price;
strncpy(iname[ol_number-1],i_name,24);
EXEC SQL WHENEVER NOT FOUND GOTO sqlerr;
EXEC SQL SELECT s_quantity, s_data,
s_dist_01, s_dist_02, s_dist_03, s_dist_04, s_dist_05
s_dist_06, s_dist_07, s_dist_08, s_dist_09, s_dist_10
INTO :s_quantity, :s_data,
:s_dist_01, :s_dist_02, :s_dist_03, :s_dist_04, :s_dist_05
:s_dist_06, :s_dist_07, :s_dist_08, :s_dist_09, :s_dist_10
FROM stock
WHERE s_i_id = :ol_i_id AND s_w_id = :ol_supply_w_id;
```

```
pick_dist_info(ol_dist_info, ol_w_id); // pick correct s_dist_xx
stock[ol_number-1] = s_quantity;
if ( (strstr(i_data,"original") != NULL) &&
(strstr(s_data,"original") != NULL) )
bg[ol_number-1] = 'B';
else
bg[ol_number-1] = 'G';
if (s_quantity > ol_quantity)
s_quantity = s_quantity - ol_quantity;
else
s_quantity = s_quantity - ol_quantity + 91;
EXEC SQL UPDATE stock SET s_quantity = :s_quantity
WHERE s_i_id = :ol_i_id
AND s_w_id = :ol_supply_w_id;
ol_amount = ol_quantity * i_price * (1+w_tax+d_tax) * (1-c_discount);
amt[ol_number-1]=ol_amount;
total += ol_amount;
EXEC SQL INSERT
INTO order_line (ol_o_id, ol_d_id, ol_w_id, ol_number,
ol_i_id, ol_supply_w_id,
ol_quantity, ol_amount, ol_dist_info)
VALUES (:o_id, :d_id, :w_id, :ol_number,
:ol_i_id, :ol_supply_w_id,
:ol_quantity, :ol_amount, :ol_dist_info);
} /*End Order Lines*/
EXEC SQL COMMIT WORK;
return(0);
invaliditem:
EXEC SQL ROLLBACK WORK;
printf("Item number is not valid");
return(0);
sqlerr:
error();
}
```

APÊNDICE – B

AMOSTRA DE CONTEÚDO - TABELA CUSTOMER

	c_w_id [PK] integer	c_d_id [PK] integer	c_id [PK] integer	c_discount numeric(4,4)	c_credit character(2)	c_last character var	c_first character var	c_credit_lim numeric(12,2	c_balance numeric(12,2	c_ytd_payme double precis	c_payment_c integer
1	1	1	1	0.2731	GC	PRIESEANTI	vrBFHolFzArLNZ	50000.00	10817.20	10	1
2	1	1	2	0.2757	GC	OUGHTCALLYAT	eKwpIuzkK	50000.00	9493.70	10	1
3	1	1	3	0.3245	GC	ESEOUGHTBAR	GFmYjLxv	50000.00	3110.42	10	1
4	1	1	4	0.3708	BC	EINGEINGPRES	HdJvPHHxd	50000.00	1231.36	10	1
5	1	1	5	0.4719	GC	PRESBARPRES	CenOQROrfmiG	50000.00	6702.60	10	1
6	1	1	6	0.4400	GC	EINGEINGBAR	UyuBRtCbL	50000.00	4022.78	10	1
7	1	1	7	0.1088	GC	PRIABLEOUGHT	rueQWgVzvSca	50000.00	224564.96	10	1
8	1	1	8	0.2456	GC	ESEABLEABLE	TBkXFJZK	50000.00	42254.00	10	1
9	1	1	9	0.0134	GC	PRESANTIATIOI	MgLOVcjk	50000.00	10737.88	10	1
10	1	1	10	0.3553	GC	PRICALLYATION	rqrlXntiGNo	50000.00	8665.80	10	1
11	1	1	11	0.3834	GC	EINGABLEATION	dnYDkYrKhOM	50000.00	67648.30	10	1
12	1	1	12	0.2923	GC	ATIONBARATIO	eUTVkfj	50000.00	7689.49	10	1
13	1	1	13	0.1377	GC	BARANTIABLE	onixgbS	50000.00	41903.62	10	1
14	1	1	14	0.3124	GC	PRICALLYOUGH	pEEBnnSXuuKb	50000.00	6611.33	10	1
15	1	1	15	0.0163	GC	BAREINGANTI	VUJpiDMv	50000.00	6623.18	10	1
16	1	1	16	0.0142	GC	ANTIABLEPRI	rjsRvtKnCW	50000.00	3004.78	10	1
17	1	1	17	0.3460	BC	CALLYPRESCALL	glxbnrXXO	50000.00	9708.80	10	1
18	1	1	18	0.3274	GC	ABLEPRESCALL	CeqZMGVc	50000.00	67605.04	10	1
19	1	1	19	0.3460	GC	ANTIANTIEING	nWsYVLFBXQ	50000.00	832.61	10	1
20	1	1	20	0.0623	GC	PRESATIONABLI	xdaTKdAmnfAM	50000.00	3559.66	10	1

FIGURA 39 - Amostra da Tabela CUSTOMER Parte-1

c_delivery_c integer	c_street_1 character var	c_street_2 character var	c_city character var	c_state character(2)	c_zip character(9)	c_phone character(16	c_since timestamp w	c_middle character(2)	c_data character var
1	ypLmRQavzBcfe	icvRHUsqXPcae	iMYMQOPOUJEc	FO	123456789	(732)744-1700	2011-12-06 16:	OE	bHgOzGRCLaGz
3	TfKxFOvauIcTQ	WYnMPDefvCST	CayNsEdMeWW	GY	123456789	(732)744-1700	2011-12-06 16:	OE	hoKMUxbLwlrXE
0	fuDnZBhUKstAH	NvbKDKcPH	XRQVmucSypy	FT	123456789	(732)744-1700	2011-12-06 16:	OE	XXjqbvWwGtlzT
1	UKNdqfBDrwxS	TufNofpblveN	IQPFsmnfkAHN	VB	123456789	(732)744-1700	2011-12-06 16:	OE	IcLoTxSaBhlgGu
0	gKzEoGRkf	AGpcjvJDjTVVT	ydUzTHpMVvdjf	VE	123456789	(732)744-1700	2011-12-06 16:	OE	IjrojIAWhYeTXl
0	lfKnYBaZvwgpS	gTagdTAroNOh	aeivbBGODk	EX	123456789	(732)744-1700	2011-12-06 16:	OE	vyDYCUFyMCZh
4	GHapsjDthfH	SXjWpPHxjRbA	inAfjQZGqgiLSp	HC	123456789	(732)744-1700	2011-12-06 16:	OE	DbcvMrzurVOzlt
1	EBridqNYCR	KZpQExMcQrAf	MxXYbeLaFs	OY	123456789	(732)744-1700	2011-12-06 16:	OE	oahwkaUzDvtqL
2	PxzzrSxfMdAgr	cOLVEQKtkLZoI	GudBGmxbB	XD	123456789	(732)744-1700	2011-12-06 16:	OE	GqfXfvlPQVkFoh
0	lxbHfCzgql	nBWWwqodcIn	lOekdRdXcpogS	TA	123456789	(732)744-1700	2011-12-06 16:	OE	VqJpvIZWfJBjnj
1	FjQSffOtTXQPD	HOaiikBqhNsAy	fXCWpgRKqDlF	LE	123456789	(732)744-1700	2011-12-06 16:	OE	gemVqCOiLeiixr
2	ebqqTAuyF	BwGlqFdeGBpra	eRYsRtWnXRAV	BA	123456789	(732)744-1700	2011-12-06 16:	OE	LCOCDJRQbkvJ
4	GNIKQvlbPbbyy	FbGOZcpuuA	PeewYYVQDDBt	SJ	123456789	(732)744-1700	2011-12-06 16:	OE	sdPfUJwGpqRac
2	HuOGwNSMkKP	recDScJqhMiUbr	wBuakdsPfvxRV	RM	123456789	(732)744-1700	2011-12-06 16:	OE	NwMgnNCHrbGy
1	jAylmrDhQsRuM	fWnjMbcHCKLgE	rTLtYaMjDfEPDT	NT	123456789	(732)744-1700	2011-12-06 16:	OE	NhwQbMisHKsH
0	EUlzNQtgNK	ndEOIlveIZJoph	yvziXjWCHZEZA	KL	123456789	(732)744-1700	2011-12-06 16:	OE	cumFyxHTfmTqu
2	BwNrZTeniBAeE	qwFEWfNXWFD	hxWCXjTJzL	GF	123456789	(732)744-1700	2011-12-06 16:	OE	17 1 1 1 1 3036.
3	AvesvrDJvLqTv	bEjPpHHTAl	cqYWuZbgOfiDJ	ND	123456789	(732)744-1700	2011-12-06 16:	OE	QAqKlgzBKpcHP
0	wxQLsJvVkN	ndugkddDvMuXf	UpKGLdbZqXraR	DK	123456789	(732)744-1700	2011-12-06 16:	OE	tyNOdnirAyAgtn
0	GnmSMZbFAl	iEpfNDPzHiU	wYHkYffzgFtblO	ZS	123456789	(732)744-1700	2011-12-06 16:	OE	dNdwCpPtTwfLi

FIGURA 40 - Amostra da Tabela CUSTOMER Parte-2

AMOSTRA DE CONTEÚDO - TABELA OORDER

	o_w_id [PK] integer	o_d_id [PK] integer	o_id [PK] integer	o_c_id integer	o_carrier_id integer	o_ol_cnt numeric(2,0)	o_all_local numeric(1,0)	o_entry_d timestamp without time zone
1	1	1	1	126	10	14	1	2011-12-06 16:54:55.162
2	1	1	2	2455	7	12	1	2011-12-06 16:54:55.162
3	1	1	3	700	2	15	1	2011-12-06 16:54:55.162
4	1	1	4	2212	9	7	1	2011-12-06 16:54:55.163
5	1	1	5	430	7	14	1	2011-12-06 16:54:55.163
6	1	1	6	2148	3	13	1	2011-12-06 16:54:55.163
7	1	1	7	821	2	11	1	2011-12-06 16:54:55.163
8	1	1	8	2952	6	6	1	2011-12-06 16:54:55.163
9	1	1	9	2883	7	13	1	2011-12-06 16:54:55.163
10	1	1	10	1491	8	13	1	2011-12-06 16:54:55.163
11	1	1	11	1792	3	5	1	2011-12-06 16:54:55.164
12	1	1	12	345	8	6	1	2011-12-06 16:54:55.164
13	1	1	13	270	6	10	1	2011-12-06 16:54:55.164
14	1	1	14	2718	2	7	1	2011-12-06 16:54:55.164
15	1	1	15	1977	5	6	1	2011-12-06 16:54:55.164
16	1	1	16	533	1	9	1	2011-12-06 16:54:55.164
17	1	1	17	2080	1	7	1	2011-12-06 16:54:55.164
18	1	1	18	351	10	13	1	2011-12-06 16:54:55.164
19	1	1	19	1925	6	8	1	2011-12-06 16:54:55.164
20	1	1	20	849	6	11	1	2011-12-06 16:54:55.164

FIGURA 41 - Amostra da Tabela OORDER

AMOSTRA DE CONTEÚDO - TABELA HISTORY

	h_c_id integer	h_c_d_id integer	h_c_w_id integer	h_d_id integer	h_w_id integer	h_date timestamp w	h_amount numeric(6,2)	h_data character var
1	1	1	1	1	1	2011-12-06 16:	10.00	xmhrVXRtpOdH
2	2	1	1	1	1	2011-12-06 16:	10.00	NXdAoOdTHLoM
3	3	1	1	1	1	2011-12-06 16:	10.00	ADbDXJpHJYCG
4	4	1	1	1	1	2011-12-06 16:	10.00	yoaAEAeNERpiiL
5	5	1	1	1	1	2011-12-06 16:	10.00	uLoAtGsKjBWrv
6	6	1	1	1	1	2011-12-06 16:	10.00	eGxWnTFPx
7	7	1	1	1	1	2011-12-06 16:	10.00	pONxTCSZRjqZg
8	8	1	1	1	1	2011-12-06 16:	10.00	byvJNUoEeTZQ
9	9	1	1	1	1	2011-12-06 16:	10.00	IaXBjsJYHMVwol
10	10	1	1	1	1	2011-12-06 16:	10.00	HguVagFHROvjU
11	11	1	1	1	1	2011-12-06 16:	10.00	vUUnbwrNAZFir
12	12	1	1	1	1	2011-12-06 16:	10.00	oXvzgSlDMOAF
13	13	1	1	1	1	2011-12-06 16:	10.00	dYvvkGvQjALaf
14	14	1	1	1	1	2011-12-06 16:	10.00	dCTHLUwkduMM
15	15	1	1	1	1	2011-12-06 16:	10.00	lOmShucSg
16	16	1	1	1	1	2011-12-06 16:	10.00	PhkCIaZqSPSvK
17	17	1	1	1	1	2011-12-06 16:	10.00	kxkFDaPGSK
18	18	1	1	1	1	2011-12-06 16:	10.00	XmKYaMtpg
19	19	1	1	1	1	2011-12-06 16:	10.00	avAEQhbNxzwo
20	20	1	1	1	1	2011-12-06 16:	10.00	slbaZAHzZWJ

FIGURA 42 - Amostra da Tabela HISTORY

AMOSTRA DE CONTEÚDO - TABELA DISTRICT

	d_w_id [PK] integer	d_id [PK] integer	d_ytd numeric(12,2	d_tax numeric(4,4)	d_next_o_id integer	d_name character var	d_street_1 character var	d_street_2 character var	d_city character var	d_state character(2)	d_zip character(9)
1	1	1	10211903.95	0.1191	7134	ZmMZg	ptkMtiTuMmKsw	cmqvzpLTbBFRu	tBfPAVIDP	YI	123456789
2	1	2	10001999.87	0.0864	7256	uoSLykG	jovaEkeFOWiAF	AuYgsEMHhcxf	mRNXvjtIYdYoq	ZJ	123456789
3	1	3	9930972.53	0.1899	7065	CEJioFy	FgFOpDHFOiTxt	FmhoagPFldWjo	uOoZUtPkaGduh	XS	123456789
4	1	4	10364434.48	0.1971	7279	qBjjqf	MxExlbrPihu	NcHtUPmgBdxat	hAUupiUUGmpKl	XW	123456789
5	1	5	10180578.11	0.1697	7360	lSWwCR	BEkjQGCcHvDFp	JQJrXFVuhLnztk	pgljqNpArpnFvV	ST	123456789
6	1	6	10076182.07	0.0450	7291	YzrhRFoir	kwSoVrhns	HfAarsorHUwAb	oRrshqNfbsIGxF	ZY	123456789
7	1	7	10183881.33	0.0139	7124	TZyJlc	BkQHPEDqiou	hTudrMukhA	EgWykfCFyxPI	SU	123456789
8	1	8	10049156.67	0.1240	7321	wQFnEqc	RyfHCDGqJE	nWZLCojvaIlHp	fCBVeQbUEPu	OS	123456789
9	1	9	10041732.48	0.0103	7185	OuXPp	DWIYcTJJIjZ	jZebeqQPN	cEwaYTCxDawD	QO	123456789
10	1	10	10312736.06	0.0205	7184	GClzmizDP	ZqWWXENDMVY	YdgmKNSFeNCh	spEyLVJOIWvv	LQ	123456789

FIGURA 43 - Amostra da Tabela DISTRICT

AMOSTRA DE CONTEÚDO - TABELA ITEM

	i_id [PK] integer	i_name character var	i_price numeric(5,2)	i_data character var	i_im_id integer
1	1	MxRuhFisOGFDe	43.34	cFpiEwGBuifOgJ	4478
2	2	GnJjlvIucxAHtz	29.21	oBMIRRlpNAigG	6048
3	3	ldnLSbcBiWUIph	94.97	ppmTqCVIKSZHe	4492
4	4	hhOzlbeFOuEUr	25.78	ZNGgoQWOztO	7903
5	5	xpCpASpkyDrkw	86.76	PLNfLzjIGKLjGT	4240
6	6	ftOhFwUoOBfml	3.43	mSkJPvodhhXDA	3278
7	7	AQPdIjoVAJKmc	40.57	uSnBMJceltspyfl	990
8	8	LrxtBfLShUcHCk	23.50	fDAmvQXQvgsfl	838
9	9	EfVNbVRTCqHnc	16.65	OSdhhweqZBLd	3709
10	10	ZneXMlpCIqQZlF	40.12	llNjnuEEmmoIRB	4596
11	11	veTjATZnHTDPY	33.62	llvjFqaRziERYHV	1981
12	12	HJKdzhxlJephMy	10.63	daVtFvfPdsFHz	433
13	13	KYBdgpLrsNcItS	72.42	JovfgTVORIGIN	5787
14	14	HAoNPPOjjxavk	64.95	BPqmBIiWDojlyb	337
15	15	UMOzwwmAFAa	44.01	xaXLbQPtmhIZb	9888
16	16	hAUvAdiALeEYK	86.10	KwrkOCLOFYTbl	1894
17	17	IJwNQSKnLvTlcl	5.97	JZurvCiyRwpde	761
18	18	hCSlGpJwzyioYJ	36.93	LLprvaKOPjufdy	3107
19	19	htFiANbPQHxCA	53.31	DToehNfdBcJLK	8716
20	20	ovZLpugoeaWm	87.17	wUzBPAVUwMjp	7045

FIGURA 44 - Amostra da Tabela ITEM

AMOSTRA DE CONTEÚDO - TABELA STOCK PARTE-1

	s_w_id [PK] integer	s_i_id [PK] integer	s_quantity numeric(4,0)	s_ytd numeric(8,2)	s_order_cnt integer	s_remote_cn integer	s_data character var	s_dist_01 character(24	s_dist_02 character(24
1	1	1	77	36.00	0	0	veEqZqmXQulkb	YnpVVqrDOfOM	lgrUWdmFZtwSil
2	1	2	36	60.00	0	0	kVdKXBnuORIGI	FogCDuabAkYZr	zsNoaPKSnCNAI
3	1	3	37	31.00	0	0	UEQtLvnTOXpQ	xhcCVhjsrOafzC	ABcfRWlqpOTxc
4	1	4	19	3.00	0	0	AvWLFUWmaSt	CBndOvkxKnWE	iFeTdYILVhIgom
5	1	5	25	1.00	0	0	AzcHRqcoURbM	YHMZOabNvVtG	vosBxxDaBjZroL
6	1	6	43	11.00	0	0	vOLJKAJyUrRqh	ydxHxEeUcccTA	frsmHLRsHIyom
7	1	7	90	40.00	0	0	GMbiIslWdwESE	kMGrDePgaNvqc	KJWWJXdqGfKv
8	1	8	83	4.00	0	0	LfojgObJYPfipM	UjBeLNvaZWzpz	AUKHfCCENpoq
9	1	9	61	15.00	0	0	GfFyJwcnNIRfTl	VDPbudjwRWtX	EneFdDeNUPnw
10	1	10	79	11.00	0	0	DTFMxKkvujbDz	TzyaUYdRjRHNJ	GRkDXLtkPIgmJs
11	1	11	12	26.00	0	0	VOOLVOWbwta	NsJazkRvSfCAlJ	WesvsUMUKOrll
12	1	12	79	27.00	0	0	IpSeeZtWSTpAu	giEzkgmPlNqWB	FuHfmJTnRXehH
13	1	13	48	19.00	0	0	DBLojdNuPsFxxl	inMTTvHEhbSmE	IfDWOUjXlKYbrç
14	1	14	47	34.00	0	0	iHSJMtMrZFlsmM	wwWSKbKdzVIg	TRkjDuRnDxsPs
15	1	15	33	26.00	0	0	IRLblhAMgqCuR	PfGChMBYADnh	jeIqMLkaDzddBr
16	1	16	13	15.00	0	0	MNqGgfZhKjmut	IonKaTSbJyPYX	mqmESnhnisGhH
17	1	17	37	17.00	0	0	FbuiYYccNhsVvj	BBMjnclPQzsVkk	kutVyxirousvwY
18	1	18	46	44.00	0	0	eVsmisIWDbExZ	EIpZBjuGyCERk	DmCbFtYIxcoJq
19	1	19	52	19.00	0	0	ssNJWhwuBKlDC	JPtepZXLTqrUEI	BdsLGVKIpvFaE
20	1	20	97	12.00	0	0	OZIFLTvTFGdqH	szprJIQhzdAQlF	INBnpXYVRVLvY

FIGURA 45 - Amostra da Tabela STOCK Parte-1

AMOSTRA DE CONTEÚDO - TABELA STOCK PARTE-2

s_dist_02 character(24	s_dist_03 character(24	s_dist_04 character(24	s_dist_05 character(24	s_dist_06 character(24	s_dist_07 character(24	s_dist_08 character(24	s_dist_09 character(24	s_dist_10 character(24
lgrUWdmFZtwSil	NQwNyNcNlUZp	QpOsbDrzzTfixF	nZQBWCGyUoKi	KIwIjmawvwKJc	gtKHSlWtxDoAg	kqIDjNPbaUMgit	mXeBmlDduKhAt	jmASLetbNBKCJ
zsNoaPKSnCNAI	frdZVfblecNlGrfp	zaoFYRDtNBIXH	OdtQeaazjcgoU	BePTBYTLujcaaT	bYSqLTINejkVvz	CmEGptbqzJiOo	hGqtErfjdFjwgdi	mVoFvMsaxKzOl
ABcfRWlqpOTxc	wDQheKlSnELYJ	QSswrsFXlRkjzb	iKXVsxEdMeiYjjt	ZshqEGMXjqKKh	DTyUPrIsvpQPw	hRIDTuBnZIVYG	LBnoQlUVDtuXh	wRrweBronEkijf
iFeTdYILVhIgom	QRRiXIpUiDhXbf	CUHQsgMfHnte	kUblOlGDJrxKnT	XVLdviJqHNWbC	SqDaTgSxwbRP	hLNlZmVsNaSujL	tzvooGHVOsXOl	jFhADvYehITJBc
vosBxxDaBjZroL	zpNJuohUyggCç	vpZdvSuHrNQD	yDJDAVgXhTncj	jDojdbQWjJqpyl	oOuVbQMBtTLQ	muWnwLQLMFY	HhSCgKHygsRY	AkuBqDeClDhLD
frsmHLRsHIyom	gREHYWEuQyT	rYKSAtzDyWREl	WTBaBziuVoCaF	UuOjJCIZKXlZriV	biqYWOQHTBFw	fGTWtaVvIvUjtz	lqvcSfXgBUXPJC	MofwNLXRjDsqk
KJWWJXdqGfKv	RhNrGjaNLtrSFI	PcUZDzAUYruY	PlQCmRGUEKAP	LyyutCdrRPfHS	YWEvTzOlCXpcI	cXWqAfRUuKbd	rgWwOGwFZekJ	zchaXThsornLBd
AUKHfCCENpoq	mMYaRkSoDAFL	hLOnbSzCXUjsT	iqdxQRQoKNZH	TyXDwdxKjnqvl	cPHfDaQDAQVK	OdalkMQvFfLZq	nhsGORtAvbVLT	PTJAFjgeOUtgyi
EneFdDeNUPnw	EmsLshdGNiAVX	jrjurUbaCONUz	fvQbaqPJTgcWi	fpPrvrKbuMWbk	jXSwWuecppSlt	QtzSkSWZFkZB	UHmFBIwaeVBC	RLmurBKgnDGxe
GRkDXLtkPIgmJs	fvcsJFtXsLTOgz	NLFJMoAqgtIwv	HydYDljgagRlna	pzBClwQBMhlXR	CrmKhfXbsCQbz	ffGBrozelnymAP	UeODzuwSeNgk	MHWXNlCiuNucy
WesvsUMUKOrll	QBCKhielQSECS	FhDHuUxRlLoHF	ANQjrPdFSSLkX	apEiYlEChyyEsZ	mxgcVtPfcoBkV	elsLYYgVJYdFW	KEQdRDOLqamc	uHgxSkHyBQBM
FuHfmJTnRXehH	cGtyimhEtoeHaA	LNYactBMcLxyel	THkJXqApNlzHA	YxTSzTTmYLtHiN	CrEBeypKhswVA	SmrgFGioSulViOl	uvTWrtzBoITuh	wGkhumrUCOGC
IfDWOUjXlKYbrç	vZtgEBHVkNNtfr	icetYFaCLzNUIp	PUwYcVPnaKhIX	NAsnzUtjAfpjktF	SjiiQXXkniwWVr	QUhZAxeOWMc	MEHKYLfdtXQOt	evnMhAZLODzB
TRkjDuRnDxsPs	fALzyeLVDZQee	mpLWqevlqOVM	zHKzzpuaAgzoC	zQJLYcgJGmGia	JmDAKxrEBIKIO	GYEOCJyUyLVFi	ouWxPkLcKAHo	fukeQvDCbDmrH
jeIqMLkaDzddBr	yVAgxriITMATKl	IsOogbULejrlOb	arVnhNCEddBAE	QTVUVFbgYJkZp	mhZzLTtwhOcilz	gfRGyRffavJXN	cwMEyhCbgGpls	NxsgvmAYOEHg
mqmESnhnisGhH	HdZjRKSySgsskt	UfbOluSicVpBIo	YPiuLmHhHWINr	eqWywbcJsWpC	iDbNldDQGKdrBu	AqGzCGEjJaDhH	bcenmGckEydvç	HlAkbIOrzGUQU
kutVyxirousvwY	cCBPnitJEBRkvK	KtZYpcNRCgBhX	ptZJWXeoenQYl	kCHlblpVaFXdVK	DTEoGxkechBCy	PnRRvUYTMsPX	iTtuwZJYaScnYf	jDBUZKXOpIJOw
DmCbFtYIxcoJq	qfOsOBNlNdnSp	gkQEHaTAqBaLc	IyzYYfeFgTiyzyh	axqCkelhfBWgx	soAbXDJtVpuKC	luScSpECbSgzKF	VVwunUpdFycqh	zfSXDRuAWtQq
BdsLGVKIpvFaE	AIIeerjEImjuRjlh	cYbnBINvrcJUN	sWMvShbgrHLe	pUCIwtBsDiuCB	evItqefQCUnqP	pYwMICQcKythç	qrfAQMNHeCBv	sVhheCgNhOeSj
INBnpXYVRVLvY	zusuukELDzMafl	xMbGHsJgocNgc	YDQCPxvcaVWc	bfQLXqJejuKEBM	tXbqmVNFhccoV	rgEFBaWWSNOl	ywFkCprdVsdka	sqgIMtGygcaYS

FIGURA 46 - Amostra da Tabela STOCK Parte-2

AMOSTRA DE CONTEÚDO - TABELA WAREHOUSE

	w_id [PK] integer	w_ytd numeric(12,2	w_tax numeric(4,4)	w_name character var	w_street_1 character var	w_street_2 character var	w_city character var	w_state character(2)	w_zip character(9)
1	1	101353577.55	0.0228	YOECEanF	zPlEwVUgktKZI	pDDggbcSsmuG	IJePzcawPRWG\	SG	123456789

FIGURA 47 - Amostra da Tabela WAREHOUSE

AMOSTRA DE CONTEÚDO - TABELA NEW_ORDER

	no_w_id [PK] integer	no_d_id [PK] integer	no_o_id [PK] integer
1	2	2	2

FIGURA 48 - Amostra da Tabela NEW_ORDER

AMOSTRA DE CONTEÚDO – TABELA ORDER_LINE

	ol_w_id [PK] integer	ol_d_id [PK] integer	ol_o_id [PK] integer	ol_number [PK] integer	ol_i_id integer	ol_delivery_d timestamp w	ol_amount numeric(6,2)	ol_supply_w_ integer	ol_quantity numeric(2,0)	ol_dist_info character(24
1	1	1	1	1	31388	2011-12-06 16:	0.00	1	5	FjsrykwkeTcsAY
2	1	1	1	2	21834	2011-12-06 16:	0.00	1	5	wnJIeyrVuaKBDl
3	1	1	1	3	15724	2011-12-06 16:	0.00	1	5	zmNraYhTulVXsZ
4	1	1	1	4	38432	2011-12-06 16:	0.00	1	5	DNsZcykcUlKyCh
5	1	1	1	5	97365	2011-12-06 16:	0.00	1	5	IjNdwyRrJwKOJ
6	1	1	1	6	456	2011-12-06 16:	0.00	1	5	OvYBdxEgiUUQy
7	1	1	1	7	55107	2011-12-06 16:	0.00	1	5	plqQksvkqKGQD
8	1	1	1	8	38031	2011-12-06 16:	0.00	1	5	GLvahHIUWwKy
9	1	1	1	9	89343	2011-12-06 16:	0.00	1	5	FHaqdksDeOfgrl
10	1	1	1	10	88622	2011-12-06 16:	0.00	1	5	IQxPAMkkJdwnE
11	1	1	1	11	86706	2011-12-06 16:	0.00	1	5	oBJrRmDKzXCO1
12	1	1	1	12	59944	2011-12-06 16:	0.00	1	5	lHBsNPmOyaMM
13	1	1	1	13	9134	2011-12-06 16:	0.00	1	5	NnskVFceXWKRj
14	1	1	1	14	74493	2011-12-06 16:	0.00	1	5	FGXbdBuCFeZY(
15	1	1	2	1	96460	2011-12-06 16:	0.00	1	5	JMWRhftQCTwN
16	1	1	2	2	63859	2011-12-06 16:	0.00	1	5	hxqAowkmnBWg
17	1	1	2	3	84168	2011-12-06 16:	0.00	1	5	oQJhzLAozzBKX
18	1	1	2	4	11944	2011-12-06 16:	0.00	1	5	FeQFtEADAGGx
19	1	1	2	5	74112	2011-12-06 16:	0.00	1	5	vhOwYPtnqWFg
20	1	1	2	6	45639	2011-12-06 16:	0.00	1	5	LhmCcdofMfmjQ

FIGURA 49 - Amostra da Tabela ORDER_LINE

APÊNDICE – C

SCRIPT CRIAÇÃO DA TABELA CUSTOMER

```
-- Table: customer

-- DROP TABLE customer;

CREATE TABLE customer
(
  c_w_id integer NOT NULL,
  c_d_id integer NOT NULL,
  c_id integer NOT NULL,
  c_discount numeric(4,4),
  c_credit character(2),
  c_last character varying(16),
  c_first character varying(16),
  c_credit_lim numeric(12,2),
  c_balance numeric(12,2),
  c_ytd_payment double precision,
  c_payment_cnt integer,
  c_delivery_cnt integer,
  c_street_1 character varying(20),
  c_street_2 character varying(20),
  c_city character varying(20),
  c_state character(2),
  c_zip character(9),
  c_phone character(16),
  c_since timestamp without time zone,
  c_middle character(2),
  c_data character varying(500),
  CONSTRAINT pk_customer PRIMARY KEY (c_w_id, c_d_id, c_id)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE customer OWNER TO postgres;

-- Index: c_ind

-- DROP INDEX c_ind;

CREATE INDEX c_ind
  ON customer
  USING btree
```

```
  (c_payment_cnt);

-- Index: cd_tipo

-- DROP INDEX cd_tipo;

CREATE INDEX cd_tipo
  ON customer
  USING btree
  (c_city varchar_pattern_ops);

-- Index: ndx_customer_name

-- DROP INDEX ndx_customer_name;

CREATE INDEX ndx_customer_name
  ON customer
  USING btree
  (c_w_id, c_d_id, c_last, c_first);
```

SCRIPT CRIAÇÃO DA TABELA DISTRICT

```
-- Table: district

-- DROP TABLE district;

CREATE TABLE district
(
  d_w_id integer NOT NULL,
  d_id integer NOT NULL,
  d_ytd numeric(12,2),
  d_tax numeric(4,4),
  d_next_o_id integer,
  d_name character varying(10),
  d_street_1 character varying(20),
  d_street_2 character varying(20),
  d_city character varying(20),
  d_state character(2),
  d_zip character(9),
  CONSTRAINT pk_district PRIMARY KEY (d_w_id, d_id)
)
WITH (
  OIDS=FALSE
);
```

```
ALTER TABLE district OWNER TO postgres;
```

SCRIPT CRIAÇÃO DA TABELA HISTORY

```
-- Table: history

-- DROP TABLE history;

CREATE TABLE history
(
  h_c_id integer,
  h_c_d_id integer,
  h_c_w_id integer,
  h_d_id integer,
  h_w_id integer,
  h_date timestamp without time zone,
  h_amount numeric(6,2),
  h_data character varying(24)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE history OWNER TO postgres;
```

SCRIPT CRIAÇÃO DA TABELA ITEM

```
-- Table: item

-- DROP TABLE item;

CREATE TABLE item
(
  i_id integer NOT NULL,
  i_name character varying(24),
  i_price numeric(5,2),
  i_data character varying(50),
  i_im_id integer,
  CONSTRAINT pk_item PRIMARY KEY (i_id)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE item OWNER TO postgres;
```

SCRIPT CRIAÇÃO DA TABELA NEW_ORDER

```
-- Table: new_order

-- DROP TABLE new_order;

CREATE TABLE new_order
(
  no_w_id integer NOT NULL,
  no_d_id integer NOT NULL,
  no_o_id integer NOT NULL,
  CONSTRAINT pk_new_order PRIMARY KEY (no_w_id, no_d_id, no_o_id)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE new_order OWNER TO postgres;
```

SCRIPT CRIAÇÃO DA TABELA OORDER

```
-- Table: oorder

-- DROP TABLE oorder;

CREATE TABLE oorder
(
  o_w_id integer NOT NULL,
  o_d_id integer NOT NULL,
  o_id integer NOT NULL,
  o_c_id integer,
  o_carrier_id integer,
  o_ol_cnt numeric(2,0),
  o_all_local numeric(1,0),
  o_entry_d timestamp without time zone,
  CONSTRAINT pk_oorder PRIMARY KEY (o_w_id, o_d_id, o_id)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE oorder OWNER TO postgres;

-- Index: ndx_oorder_carrier

-- DROP INDEX ndx_oorder_carrier;

CREATE UNIQUE INDEX ndx_oorder_carrier
```

```
  ON oorder
  USING btree
  (o_w_id, o_d_id, o_carrier_id, o_id);
```

SCRIPT CRIAÇÃO DA TABELA ORDER_LINE

```
-- Table: order_line

-- DROP TABLE order_line;

CREATE TABLE order_line
(
  ol_w_id integer NOT NULL,
  ol_d_id integer NOT NULL,
  ol_o_id integer NOT NULL,
  ol_number integer NOT NULL,
  ol_i_id integer NOT NULL,
  ol_delivery_d timestamp without time zone,
  ol_amount numeric(6,2),
  ol_supply_w_id integer,
  ol_quantity numeric(2,0),
  ol_dist_info character(24),
  CONSTRAINT pk_order_line PRIMARY KEY (ol_w_id, ol_d_id, ol_o_id, ol_number)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE order_line OWNER TO postgres;
```

SCRIPT CRIAÇÃO DA TABELA STOCK

```
-- Table: stock

-- DROP TABLE stock;

CREATE TABLE stock
(
  s_w_id integer NOT NULL,
  s_i_id integer NOT NULL,
  s_quantity numeric(4,0),
  s_ytd numeric(8,2),
  s_order_cnt integer,
  s_remote_cnt integer,
  s_data character varying(50),
```

```
  s_dist_01 character(24),
  s_dist_02 character(24),
  s_dist_03 character(24),
  s_dist_04 character(24),
  s_dist_05 character(24),
  s_dist_06 character(24),
  s_dist_07 character(24),
  s_dist_08 character(24),
  s_dist_09 character(24),
  s_dist_10 character(24),
  CONSTRAINT pk_stock PRIMARY KEY (s_w_id, s_i_id)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE stock OWNER TO postgres;
```

SCRIPT CRIAÇÃO DA TABELA WAREHOUSE

```
-- Table: warehouse

-- DROP TABLE warehouse;

CREATE TABLE warehouse
(
  w_id integer NOT NULL,
  w_ytd numeric(12,2),
  w_tax numeric(4,4),
  w_name character varying(10),
  w_street_1 character varying(20),
  w_street_2 character varying(20),
  w_city character varying(20),
  w_state character(2),
  w_zip character(9),
  CONSTRAINT pk_warehouse PRIMARY KEY (w_id)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE warehouse OWNER TO postgres;
```