UNIVERSIDADE DO SAGRADO CORAÇÃO

GUSTAVO SANTOS

O USO DE TABELAS AUXILIARES NA OTIMIZAÇÃO DE PERFORMANCE DE BANCO DE DADOS UTILIZANDO TECNOLOGIA POSTGRES

BAURU
2015

GUSTAVO SANTOS

O USO DE TABELAS AUXILIARES NA OTIMIZAÇÃO DE PERFORMANCE DE BANCO DE DADOS UTILIZANDO TECNOLOGIA POSTGRES.

Trabalho de conclusão de curso apresentado ao Centro de Ciências Exatas e Sociais Aplicadas da Universidade do Sagrado Coração, como parte dos requisitos para obtenção do título de Bacharel em Ciência da Computação, sob orientação do Profº Drº Elvio Gilberto da Silva.

BAURU
2015

Santos, Gustavo
S2374u

O uso de tabelas auxiliares na otimização de performance de banco de dados utilizando tecnologia postgres / Gustavo Santos. -- 2015.
36f. : il.

Orientador: Prof. Dr. Élvio Gilberto da Silva.

Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) – Universidade do Sagrado Coração – Bauru – SP.

1. Banco de dados relacional. 2. Otimização e Performance. 3. Postgres. 4. Software livre. I. Silva, Élvio Gilberto da. II. Título.

GUSTAVO SANTOS

O USO DE TABELAS AUXILIARES NA OTIMIZAÇÃO DE PERFORMANCE DE BANCO DE DADOS UTILIZANDO TECNOLOGIA POSTGRES.

Trabalho de Conclusão de Curso apresentado ao Centro de Ciências Exatas e Sociais Aplicadas da Universidade do Sagrado Coração, como parte dos requisitos para obtenção do título de Bacharel em Ciência da Computação, sob orientação do Profº Drº Elvio Gilberto da Silva.

Banca examinadora:

_____

Prof. Dr. Elvio Gilberto da Silva
Universidade Sagrado Coração

_____

Prof. Me. Henrique Pachioni Martins
Universidade Sagrado Coração

_____

Prof. Me. Marcio Henrique Castilho Cardim
Universidade Sagrado Coração

Bauru, 2 de dezembro de 2015.

AGRADECIMENTOS

Agradeço primeiramente a Deus, por ter concedido força e perseverança para que eu pudesse iniciar e terminar meu curso de ensino superior.

Aos meus pais e minha irmã, Ricardo, Leila e Mariana, que por seu amor incondicional apostaram na minha capacidade, me proveram conforto nos momentos turbulentos, lutaram junto comigo para minha formação e nunca me deixaram desistir de meus objetivos.

A minha namorada, Jennifer por ter paciência e ser minha companheira em todo esse tempo de estudo.

Aos meus avós paternos, Ivo e Lili, que me deram lições de que a maior virtude de um homem é sua dignidade, honestidade, determinação, humildade e sobre tudo sua família.

Ao corpo docente da Universidade do Sagrado Coração, em especial ao meu professor orientador Prof. Dr. Elvio Gilberto da Silva, que em todos esses anos transmitiram o máximo de conhecimento que puderam, e não só por isso, por terem contribuído com meu desenvolvimento pessoal dentro e fora da universidade.

A todos, ficam meus agradecimentos de elevada estima e consideração.

“Uma paixão forte por qualquer objetivo assegurará o sucesso, porque o desejo pelo objetivo mostrará os meios. [...]”.

William Hazlitt

RESUMO

Empresas que utilizam sistema segmentado de informação para gerir a regra de negócio, geram grandes quantidades de informações que são armazenadas de forma coerente e coesa no banco de dados. As informações precisam ser entregue praticamente que instantaneamente para quando feito alguma pesquisa, mas devido ao volume crescente constantemente, essa entrega costuma a ficar cada vez mais morosa a ponto de inviável a utilização de determinado recurso devido ao tempo gasto para se gerar um relatório por exemplo. Um software legado, que se presume ter sido desenvolvido numa tecnologia antiga, sofre ainda mais com o quesito de lentidão na entrega das consultas, e isso faz com que a vida útil do sistema legado seja antecipada sendo um dano inestimável simplesmente deixar de utilizar o sistema devido a lentidão para gerar um relatório ou até mesmo para trabalhar. Em várias situações, o problema de lentidão não está relacionado com infraestrutura, sistema operacional ou mesmo periféricos, mas sim na modelagem do banco de dados relacional e a escrita das consultas que são feitas dentro do BD. A constante verificação das queries, índices, constraints e atualização das tabelas de estatísticas do banco de dados (independente da plataforma), podem resultar num ganho aceitável de performance. Cada plataforma de banco de dados tem o seu SGDB padrão fornecido na maioria das vezes pelo próprio fabricante ou até mesmo pela comunidade no caso de um software livre como é o caso do Postgres. O SGDB é amparado com ferramentas úteis para o estudo das consultas, rotinas de backup, organização dos índices, desfragmentação dos registros e atualização das tabelas de estatísticas. O principal objetivo do presente trabalho é verificar se o uso de tabelas temporárias e auxiliares podem aumentar a performance do banco de dados para executar uma consulta em entidades com grandes quantidades de informações com relacionamento N-para-M, e se esse ganho é efetivo e aceitável.

Palavras-chave: Banco de dados relacional. Otimização e Performance. Postgres. Software livre.

ABSTRACT

Companies using targeted information system to manage business rule, generate large amounts of information that are stored coherently and cohesively in the database. Information needs to be delivered almost instantly to when doing some research, but due to the increasing volume constantly, this delivery usually getting increasingly time consuming to the point of impossible to use certain resource due to the time required to generate a report for example . A legacy software, which is presumed to have been developed on an old technology, further suffers from the Question of slowness in delivering these consultations, and this makes the life of the legacy system is anticipated to be a inestimable damage just stop using the system due to slow to generate a report or even to work. In many situations, the slowness problem is not related to infrastructure, operating system or even peripheral, but in the modeling of the relational database and writing the queries that are made within the database. The constant checking of queries, indexes, constraints and update the database statistics tables (regardless of platform) could result in an acceptable performance gain. Each database platform has its standard DBMS provided mostly by the manufacturer or even the community in the case of free software such as the Postgres. The DBMS is supported with useful tools for the study of queries, backup routines, organization of indices, defragmentation of records and updating the statistics tables. The main objective of this study is to verify that the use of temporary and auxiliary tables can increase database performance to run a query for entities with large amounts of information with relationship N-to-M, and this gain is effective and acceptable.

Keywords: Relational Database. Optimization and Performance. Postgres. Free software.

LISTA DE ABREVIATURAS E SIGLAS

DCL – Data Control Language

DDL – Data Definition Language

DML – Data Manipulation Language

ER – Entidade-Relacionamento

GUI – Graphical User Interface

PHP – Personal Home Page

SGDB – Sistema de Gerenciamento de Banco de Dados

SQL – Structured Query Language

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

É comum no mercado atual, existir migrações de sistemas legados para sistemas segmentados de informações. Como uma remodelagem de um sistema legado se torna inviável devido ao tempo que será investido em uma tecnologia que se presume ultrapassada, e, por conta da metodologia de desenvolvimento empregada, o schema arcaico que possa ter sido desenvolvido no sistema, as fábricas de software tendem a cada vez mais procurar uma solução ágil e eficaz para que um programa antigo não seja descontinuado, tendo em vista que sistemas como esse aqui descrito, contém vários módulos e podem abrigar toda a regra de negócio da empresa. Os estudos consistem em utilizar algumas funcionalidades pouco exploradas pelas empresas desenvolvedoras que podem sem custo nenhum, dar tempo a mais de vida para esse sistema e aperfeiçoar sucintamente o modelo relacional.

Será verificado se a aplicabilidade dos conceitos que aqui serão expostos, podem ou não melhorar a performance do banco de dados relacional, uma vez que o software (legado ou não) atinge um volume muito grande (devido ao grande número de registros) e extremamente complexo a execução da remodelagem das informações no meio de um projeto, portanto, serão efetuadas modificações graduais no schema proposto para que possa ser colhido os resultados e analisado com uma visão técnica o quesito de performance e organização. Como resultado desse trabalho, o objetivo é verificar se é válido afirmar que quanto antes houver o estudo, melhor será o resultado obtido, a manutenção será bem mais simples e assertiva, e poupará eventuais transtornos no que diz respeito a problemas futuros por falta de conexões (contraints, primary Keys, foreing Keys, índices...) que deveriam ter sido previamente analisados.

Com base no contexto, este trabalho tem o propósito de demonstrar por meio de testes que a utilização de técnicas de tabelas temporárias, a reescrita das queries e modelagem do banco de dados pode reduzir o uso dos recursos físicos dos servidores possibilitando até a retenção de recursos financeiros ao se investir em servidores de grande porte de processamento e memória.

O trabalho apresentado será divido em cinco etapas: A primeira etapa consistirá na criação do ambiente (virtualizado) e a criação do banco de dados na

plataforma Postgres 9.4. A segunda etapa consistirá na definição e modelagem de uma parte de um sistema que controla o back-office de uma empresa, tal que o modelo relacional de banco de dados atenda a necessidade e a população dos registros serão com dados fictícios. A terceira etapa será estipulada uma consulta no banco de dados criado que simulará um relatório que se relaciona com várias entidades do schema, essa query será escrita no padrão SQL99, que tem por característica principalmente a ausência de aliases e uso do comando USING. A quarta etapa consistirá na criação de primary keys, foreing keys, constraints, afim de que o nosso banco comece a ser otimizado e posteriormente a isso executaremos a mesma query criada na etapa anterior e armazenaremos os resultados de ambas para comparação. A query será reescrita utilizando a tabela temporária para armazenar os registros dentro de uma função e mais uma vez armazenaremos os resultados. Na última etapa, será colhido os resultados das execuções das queries e exposto num gráfico o consumo e tempo gasto para execução de cada.

2 OBJETIVOS

Para alcançar os objetivos, serão executados testes gradativos, e armazenados os resultados a cada etapa afim de mensurar os dados finais e apurar de forma específica a performance, desempenho e organização.

2.1 OBJETIVO GERAL

Demonstrar por meio de testes que a utilização de técnicas de tabelas temporárias e auxiliares, em conjunto da escrita correta da query e a modelagem otimizada do banco de dados, podem poupar recursos financeiros ao se investir em máquinas de servidores de grande porte de processamento e memória.

2.2 OBJETIVOS ESPECÍFICOS

- Efetuar um levantamento bibliográfico acerca dos temas que norteiam a pesquisa;
- Criar um banco de dados Postgres com grande volume de dados e amarrações simples;
- Extrair informações contidas no banco de dados e armazenas as estatísticas obtidas a fim de compará-los com outros resultados obtidos no decorrer dos testes;
- Criar índices nas tabelas do banco de dados;
- Utilizar técnicas diferentes para a escrita da query e armazenar as estatísticas dos resultados obtidos;
- Expor os resultados obtidos a fim de comprovar se essa técnica é usável para otimização de performance de banco de dados;
- Comparar as técnicas apresentadas.

3 FUNDAMENTAÇÃO TEÓRICA

3.1 BANCO DE DADOS

As tecnologias referentes a banco de dados e manipulação de registros crescem a cada dia mais e está ligado praticamente a todas nossas ações cotidianas abrangendo os mais diversos segmentos.

> É viável afirmar que eles representam um papel crítico em quase todas as áreas em que os computadores são utilizados, incluindo negócios, comércio eletrônico, engenharia, medicina, direito, educação e as ciências da informação, para citar apenas algumas delas. (ELMASRI; NAVATHE, 2006, p. 4).

Bancos de dados são conjuntos de informação que são construídos e desenvolvidos, visando à resolução de determinada situação ou problema através de abstração da realidade, transpondo todos esses aspectos de forma lógica e organizada, pois para Elmasri (2006, p. 4), “um banco de dados representa alguns aspectos do mundo real, sendo chamado às vezes, de minimundo ou de universo de discursos (UoD). As mudanças do minimundo são refletidas em um banco de dados em tempo real”.

Só é classificado como banco de dados, a partir do momento em que a coleção de informações armazenadas são relacionadas umas com as outras, fazendo dela uma grande rede de registros que manipulados de forma coesa dentro de um contexto, torna-o efetivo e inteligente, e que supra total ou parcialmente a necessidade imposta.

Ainda, segundo o autor citado anteriormente, um banco de dados é uma coleção lógica e coerente de dados com algum significado inerente. Uma organização de dados ao acaso (randômica) não pode ser corretamente interpretada como um banco de dados, pois para tal deve ser projetado, construído e povoado por dados, atendendo a uma proposta específica. Possui um grupo de usuários definido e algumas aplicações preconcebidas, de acordo com o interesse desse grupo de usuários. (ELSMARI, 2006).

Oppel (2009) afirma ainda que banco de dados é uma coleção inter-relacionada de informações gerenciada por uma unidade simples e que esta

definição é deliberadamente ampla, pois existem infinitas variedades em toda diversidade de sistemas fornecedores de informações de forma estruturada e objetiva.

Do ponto de vista de Sharma (2010), bancos de dados estão entre os domínios do conhecimento mais pesquisados pelos profissionais em Ciência da Computação, e que essa ferramenta é um repositório de registros, projetado para armazenar, recuperar e prover a devida manutenção em suas informações; e o autor supracitado ainda cita alguns tipos de bancos de dados específicos para armazenamento de arquivos binários, documentos, imagens, vídeos, dados relacionais, dados multidimensionais, dados transacionais, analítica, dados geográficos entre diversos outros.

3.1.1 MODELO DE DADOS RELACIONAL

Tomando como base o principal modelo de estrutura de dados em uso hoje do mercado, será aplicado os testes desse trabalho no modelo relacional.

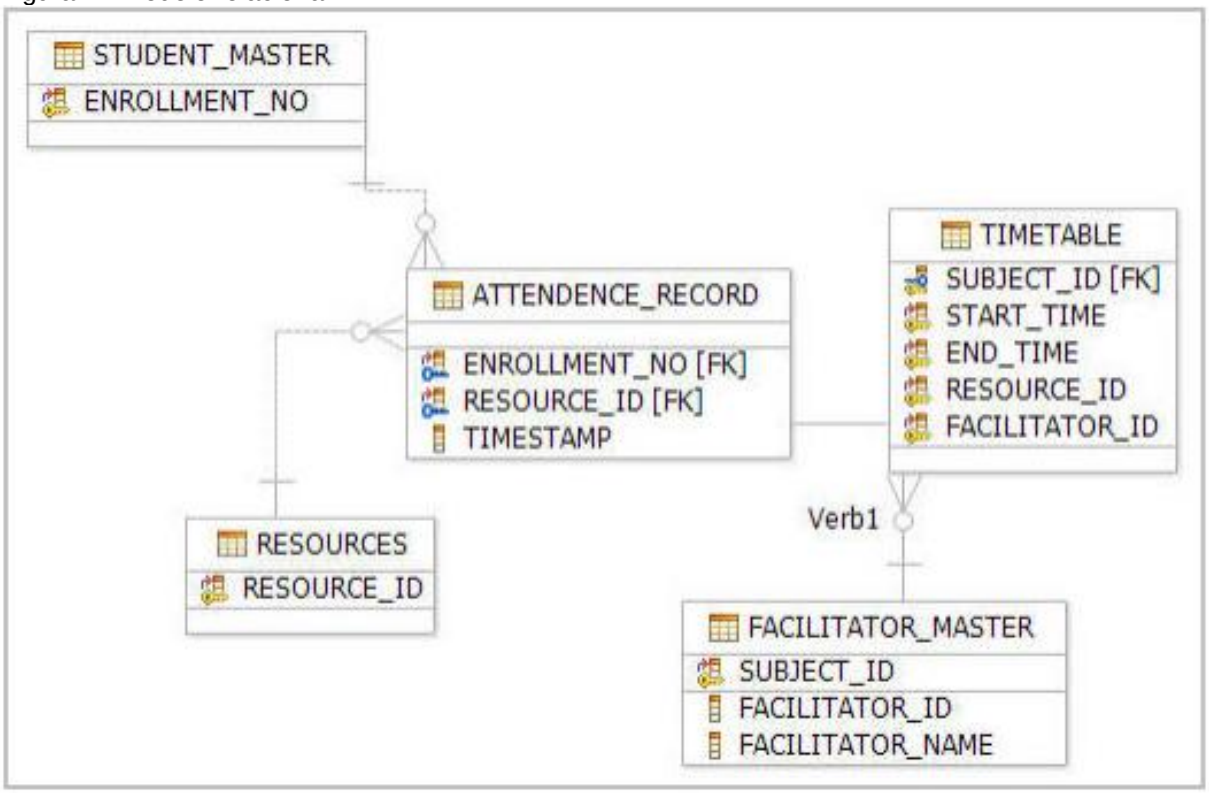
Uma característica da estrutura desse banco de dados é a sua fácil interpretação a respeito de seus atributos e relacionamentos, o que torna a plataforma leve e eficiente. Aplicativos que utilizam esse tipo de modelagem atendem vários segmentos e em qualquer escala o que deixa o desenvolvimento da solução mais clara e com baixo custo.

Na interpretação de Sharma (2010), o modelo relacional é simples e elegante. Com base em matemática sólida, teoria dos conjuntos e cálculos de predicados, o modelo Entidade-Relacionamento (ER) é tido como o mais usado do mercado.

Ainda no entendimento de Sharma (2010), uma estrutura de dados simples, como o modelo relacional, é mais objetivo o fornecimento de dados de forma lógica e independente.

A Figura 1 mostra um exemplo de uma Entidade-Relacionamento (ER), onde esse diagrama é representado por entidades (tabelas) e suas relações.

Figura 1 - Modelo relacional.



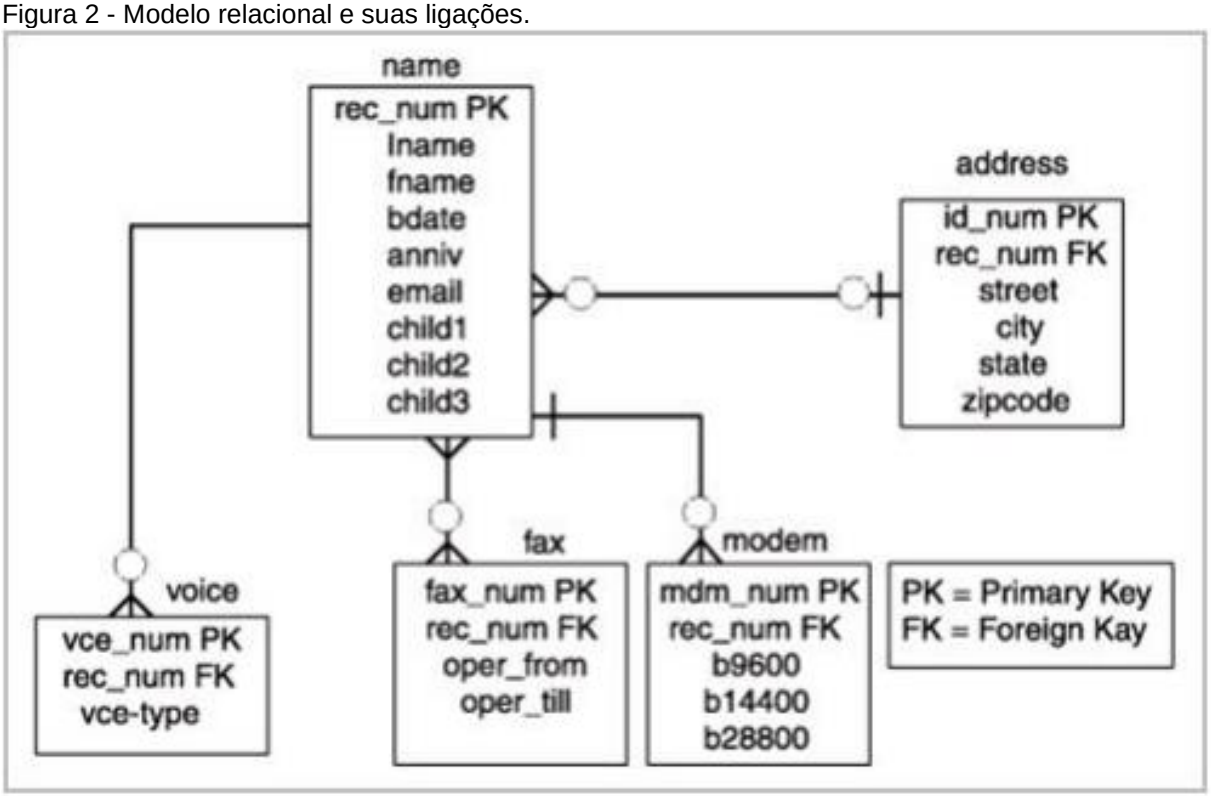
Fonte: Sharma, 2010, p. 26.

Meados de 1970, Peter Chen propôs o modelo de dados entidade-relacionamento (ER) para que fosse uma alternativa para o então método mais utilizado relacional. As informações seriam distribuídas de forma hierárquicas, organizadas e correlacionadas entre si formando um conjunto de informações.

Uma entidade não necessita da existência de outra, esses objetos possuem atributos que definem os elementos de cada dado que caracterizam uma entidade, pois Sharma (2010) cita que entidades são objetos tem uma existência independente de quaisquer outras entidades do banco de dados e que estas possuem atributos e elementos que as definem.

Por fim surgem ainda os relacionamentos, visto que todas as informações armazenadas no banco de dados tem relação umas com as outras, que Chen (1970) já havia dito e Sharma (2010) confirma que, a maior característica de um banco normalizado relacional, são as propriedades de suas ligações que podem ser 1-a-1, 1-para-N, N-a-1 ou de M-de-N dependendo do conceito utilizado e a lógica de negócio desenvolvida para resolver tal situação ou desenvolver uma ferramenta para determinada finalidade. A Figura 2 representa um exemplo de diagrama entidade-relacionamento com exemplos das ligações.

Figura 2 - Modelo relacional e suas ligações.



Fonte: Sharma, 2010, p. 31.

3.2 SISTEMA DE GERENCIAMENTO DE BANCO DE DADOS (SGDB)

No tocante a parte de administração do banco de dados e manipulação de registros, se faz necessário o uso de ferramentas específicas para que se possa executar todas as rotinas necessárias de tratamento, inclusão, exclusão, manutenção e segurança das informações em tempo real, inibindo o máximo o uso do servidor e da máquina de aplicação, e garantindo a confiabilidade do banco de dados. Para isso é criado um ambiente de desenvolvimento onde somente a equipe de produz a ferramenta tem acesso completo ao código fonte da aplicação, restringindo o usuário de informações técnicas e ainda como método de segurança interno.

> Os programas não contêm todo o código referente a exibição dos dados na interface, mas utilizam gerenciadores de interface de usuário, conjuntos de rotinas que incluem as funcionalidades que um programador vai necessitar frequentemente, ao construir uma interface de usuário. (HEUSER, 1998)

3.3 TECNOLOGIA POSTGRESQL

A tecnologia que será utilizada nesse trabalho, atende a todos os requisitos. Foi desenvolvida no final da década de 1970, pela Universidade de Berkeley na Califórnia.

> PostgreSQL deu início a sua árvore por volta de 1977 na Universidade de Berkeley na Califórnia (UCB). (MATTHEW, Begining Database with PostgreSQL, 2005)

É comum no início da carreira o desenvolvedor, buscar ferramentas open-source devido à quantidade de material fornecido bem como a grande quantidade de recursos que oferecem, pois de acordo com Matthew (2005) os sistemas de código fonte aberto, além de ser gratuito, o que é vantagem, contam com uma altíssima qualidade e desempenho, afinal quem presta manutenção nessas ferramentas é a própria comunidade contribuinte.

Matthew (2005) destaca um ponto importante a respeito desse tipo de licença, é que o próprio desenvolvedor pode corrigir uma falha do sistema e submeter à comunidade, ou ainda adaptar para a regra de negócio existente.

A aplicabilidade do banco de dados se dá praticamente a todo sistema, e Matthew (2005) expõe a definição de banco de dados baseado em termo técnico e faz uma analogia de que não é um ambiente de trabalho, mas sim como fonte de informação para diversos sistemas nos mais diversos ramos.

O Postgres além de ser multiplataforma (Windows, Linux, Unix, BSD...), compreende desde sua primeira versão o SQL e para manipulação das informações são utilizadas as DML, DDL e DCL e Matthew (2005) dá a definição de cada uma.

- Data Manipulation Language (DML): Usado para manutenção dos registros do banco de dados, usado na maioria do tempo. É composta pelos comandos de INSERT, UPDATE, DELETE e SELECT.

- Data Definition Language (DDL): Comandos utilizados para criação de tabelas, relacionamentos e controles do aspecto estrutural da modelagem.

- Data Control Language (DCL): Principais comandos para controlar níveis de permissões, direitos de acessos e manutenção preventiva e corretiva. Esses comandos são pouco utilizados pelas companhias fabricantes de software, uma vez que para funcionamento correto se faz necessário a presença de um profissional em administração de banco de dados para controlar todo o fluxo.

3.3.1 PRÁTICAS DE PERFORMANCE

Focando na tecnologia PostgreSQL e suas funcionalidades no quesito performance, serão utilizadas algumas terminologias que não possuem tradução ao pé da letra e, por esse motivo, serão expostos alguns termos que necessitam ser bem definidos para compreender o emprego da técnica posteriormente. Para isso, Smith (2010) expõe:

- Bottleneck ou limiting fator: Ambos os termos são utilizados para referir-se à limitações físicas ou lógicas que está dificultando um melhor desempenho da aplicação.
- Benchmarking: É a execução de um teste para determinar o quão rápido uma determinada operação pode ocorrer, isso geralmente é feito para encontrar gargalos no sistema.
- Profiling: Monitoramento de partes específicas do programa e tratativa da rotina que está consumindo maior recurso ao executar uma operação difícil, tais como valor de referência. Geralmente é utilizado antes e depois das alterações para verificação de resultados e é usado em conjunto com testes de desempenhos onde vão forçar o gargalo. Ferramentas de criação de perfil no nível de código são gprof, oprofile e dtrace.

4 METODOLOGIA

Para demonstrar o uso dessa técnica, foi seguido determinadas etapas afim de manter uma ordem cronológica e coesa. A primeira etapa consiste na criação de uma máquina virtual com recursos que serão detalhados a seguir como servidor do banco de dados. Logo após, foi criado um banco de dados relacional que foi exposto o schema em forma de figura.

Começando pelo ambiente virtual que simulou o servidor do banco de dados de uma corporação de médio porte (quanto a quantidade de registros no banco de dados), todos os exemplos utilizados aqui serão fictícios.

A máquina virtual foi criada com recursos lógicos limitados para que quando o banco de dados estivesse populado com uma grande quantidade de registros, e esses recursos permanecesse próximo ao limite de processamento para executar tais tarefas. Essa máquina virtual foi constituída de apenas um núcleo de processador, 4GB de memória RAM (alocados dinamicamente entre sistema operacional e aplicações) e 40GB de espaço no disco rígido. Os recursos limitados da máquina virtual foram propositais, pois assim, foi simulado um servidor com alta carga de processamento, e a execução da query (que será modelada no decorrer do trabalho) do banco de dados concorrendo com o sistema operacional e demais aplicativos exemplificou o objetivo proposto, e a tecnologia de virtualização forneceu exatamente esse tipo de manipulação de recursos.

Essa máquina qual foi designada como servidor foi contemplada com Linux Ubuntu 14.04.03 LTS na sua última versão. Foram instalados somente os recursos básicos do sistema operacional para servidor, e a GUI (Graphical User Interface) não foi implementada.

Foi instalado no servidor o PostgreSQL 9.4 que atualmente é a última versão estável disponível para ser o banco de dados, onde foram expostas as técnicas propostas afim de cumprir o objetivo do trabalho. O PostgreSQL desde a sua versão 8.0, concede aos desenvolvedores um SGDB estável, leve, prático e com recursos que facilitam obter estatísticas a cada etapa do plano.

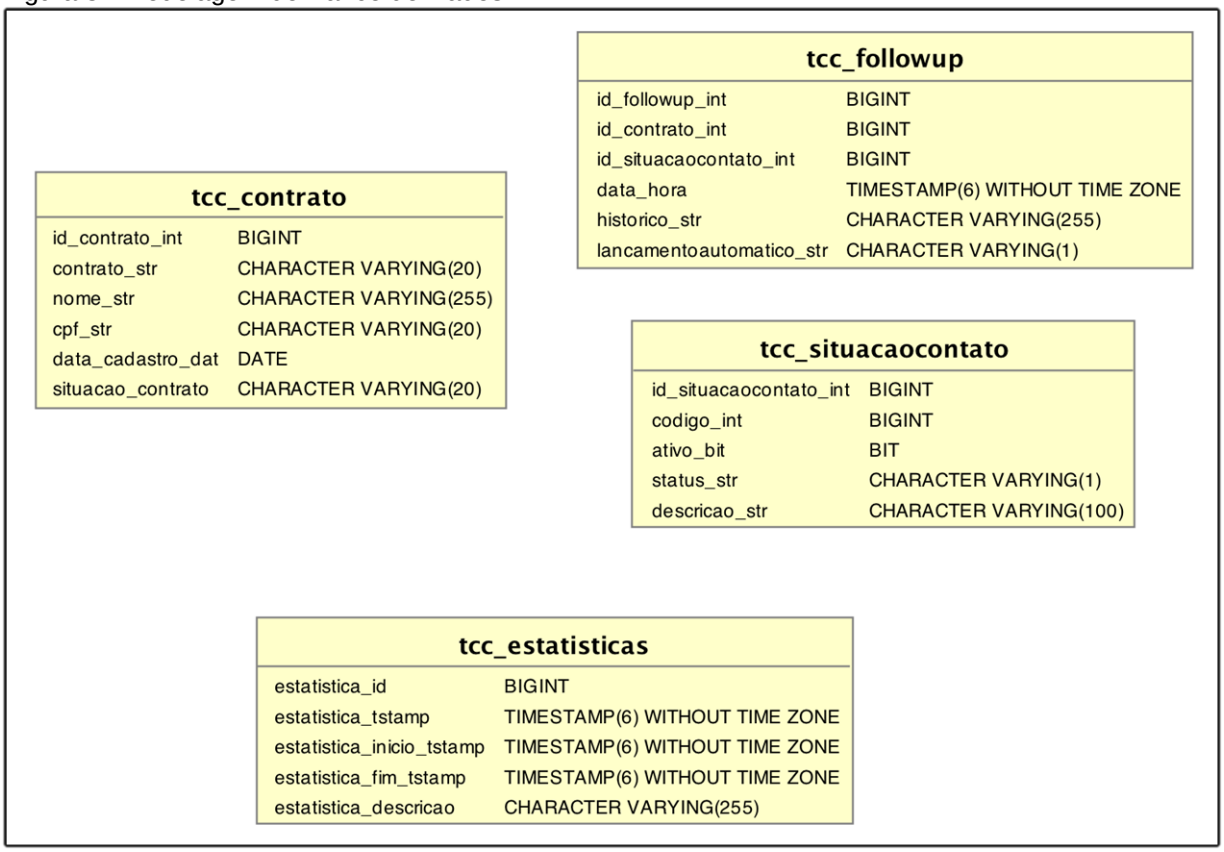
O servidor continha ainda tecnologias a parte para que pudesse auxiliar determinados procedimentos e tornar o ambiente de testes mais amigável. Essas tecnologias foram: PHP, Apache, SSH e UFW. Foi criado um simples portal em PHP, e este foi conectado com o banco de dados, para que pudesse popular o banco de

dados de forma dinâmica e oferecer de forma visível a diferença entre as etapas do processo. Quanto a tecnologia SSH, foi somente para acesso ao servidor virtual via modo texto, e o UFW é um firewall nativo do Ubuntu que simplesmente foi habilitado todos os acessos de entrada e saída. Mostrar como funciona cada uma dessas tecnologias não é o objetivo, mas vale ressaltar que até este ponto tudo o que se tem é um servidor virtual com o serviço do banco de dados online com acesso remoto a ele.

Foram criadas algumas entidades de uma pequena parte do banco de dados de uma empresa fictícia, a qual presta serviço de back-office para outras corporações, e visando essa modelagem foi obtido um schema de tabelas onde foram armazenadas todas as informações pertinentes ao negócio numa única fonte. Para demonstrar a aplicação desse método de performance, foram feitas modificações gradativas no banco de dados e foram armazenados todos os resultados para que pudessem ser comparados posteriormente com as informações obtidas e assim demonstrar a viabilidade de adaptar-se a técnica de uso de tabela temporária ou não.

Para criação do banco de dados, foi utilizado o PGAdmin III 1.20.0, ferramenta que pertence ao SGDB padrão da tecnologia PostgreSQL, embora pudesse ser usado qualquer editor de SQL que aceite conexão com esse tipo de banco. Inicialmente foi criado um banco de dados apenas com as tabelas normalizadas, que por hora não será contemplado com amarrações ou índices; em sistemas legados raramente há tratamentos como esses, e o desejado é expor exatamente esse cenário. A Figura 3 ilustra as entidades que foram criadas para aplicar os conceitos que aqui foram estudados e a entidade tcc_estatisticas apresentada na Figura 4 funcionou a parte da modelagem desse banco, e o seu uso foi somente para armazenar os resultados obtidos a cada etapa que foi avançada, e no final dessa técnica, ela abrigou todo o resultado da otimização.

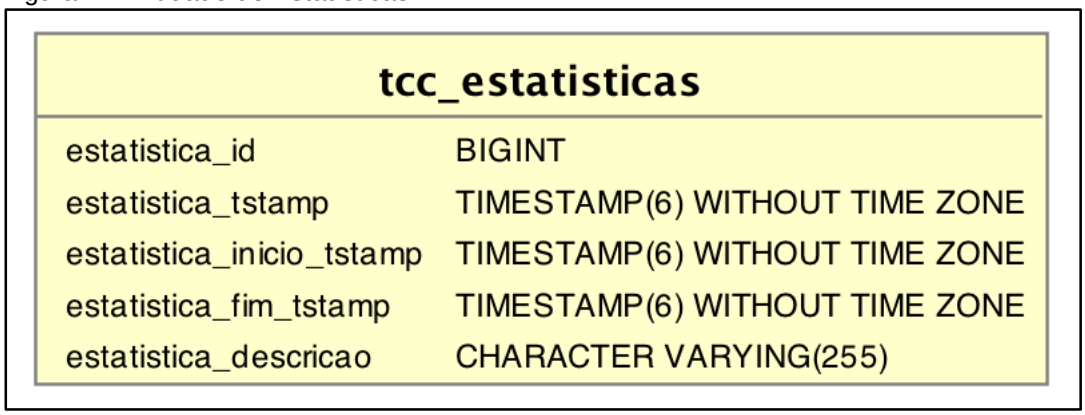
Figura 3 - Modelagem do Banco de Dados.



Fonte: Elaborada pelo autor.

A entidade tcc_estatisticas apresentada na Figura 4, foi uma tabela a parte do sistema proposto, e nela que foram armazenados todos os resultados das etapas que foram executadas no decorrer do desenvolvimento, e estes foram estudados e confrontados posteriormente. Essa tabela tem atributos básicos para que possa ser feito um levantamento assertivo como demonstrado na Figura 4.

Figura 4 - Entidade de Estatísticas.

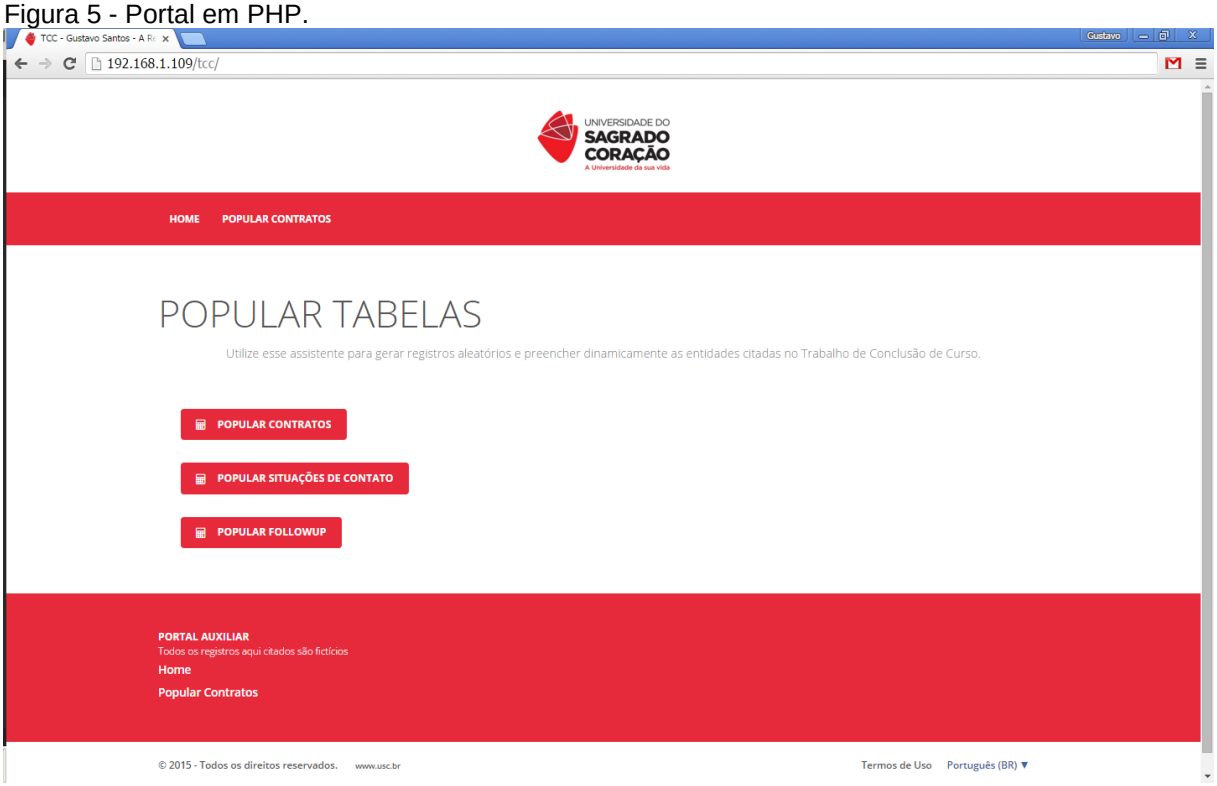


Fonte: Elaborada pelo autor.

Com todas as tabelas criadas e a modelagem definida, foi dado início a uma rotina utilizando a tecnologia PHP para automatizar a população fictícia das entidades desse banco. Para exemplificar esse conceito aqui abordado, foi necessário que o banco de dados fosse populado com muitos registros no banco, pois ficaria totalmente inviável alimentar dados manualmente para testes, e por esse motivo foi utilizada a tecnologia PHP para auxiliar no processo.

Foi criado um micro portal utilizando a conexão com o banco de dados e este micro portal é ilustrado na Figura 5.

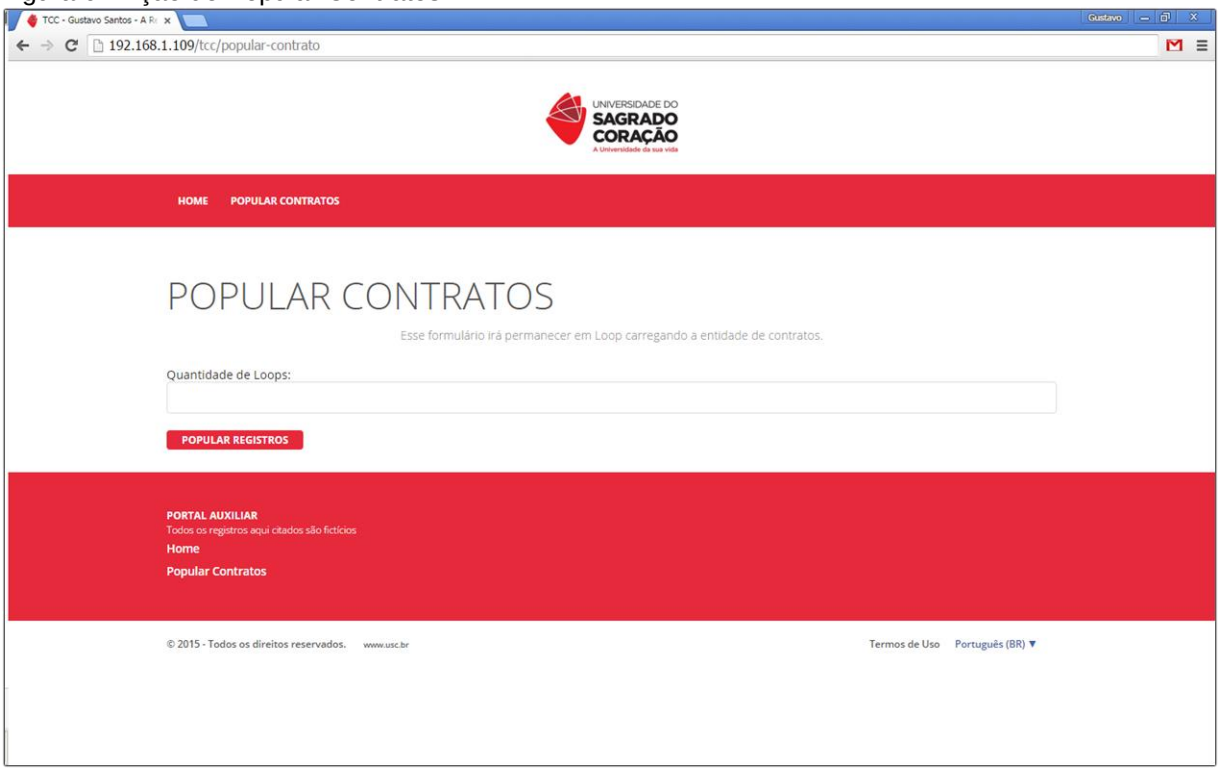
Figura 5 - Portal em PHP.



Fonte: Elaborada pelo autor.

Ao clicar em “POPULAR CONTRATOS” foi redirecionado para uma página que exibiu um formulário com um campo para digitar o número da quantidade de contratos que deseja-se que fosse cadastrado no banco de dados e um botão como mostra a Figura 6.

Figura 6 - Ação de Popular Contratos.



Fonte: Elaborada pelo autor.

A ação desse botão foi a criação de registros baseados em vetores aleatórios e o uso da DML INSERT no banco de dados populando a entidade tcc_contrato. Essa página em PHP permaneceu em loop, até a quantidade digitada no formulário, atribuindo valores a algumas variáveis randômicas montando registros e alimentando o banco de dados. Foi distribuída a informação para essas tabelas de forma lógica e organizada. O processo pode ser repetido quantas vezes forem necessário, mas o objetivo era prover informação no banco de dados até que ele tenha uma quantidade de registros o suficiente para executar os testes de performance. Não há um limite para essa quantidade, e nem uma forma clara de especificar um limite, tudo isso serve apenas de exemplo para que possa ser criada a estrutura para exemplificar o objetivo desse trabalho. Essa quantidade de registros comparado ao tempo de geração pode variar de servidor para servidor, o que se tem aqui apresentado é uma máquina virtual com poucos recursos. Num ambiente físico, com processador específico para processamento de dados e uma grande quantidade de memória, a quantidade aqui proposta pode não apresentar grande variação, mas caso o servidor trabalhe próximo ao seu nível limite, deve ser verificado se os conceitos aqui descritos poupam recursos e diminuem o load do servidor.

Para alimentar a tabela tcc_contrato, foram criados vetores com vários valores e o loop atribuiu posições randômicas dos vetores e montou a inserção no banco de dados. Para exemplificar o processo, a entidade tcc_contrato foi alimentada com vetores da seguinte forma:

- id_contrato_int: Foi um campo sequencial do banco de dados.
- contrato_str: Foi constituído de 10 caracteres numéricos aleatórios de 0 à 9 e este não pode se repetir no banco de dados.
- nome_str: Foi criado dois vetores com vários nomes e sobrenomes, exemplo vNome[‘Gustavo’, ‘Paulo’, ‘Gisele’, ...] e vSobreNome [‘Santos’, ‘Souza’, ‘Fernandes’, ...] e o loop atribuiu posições aleatórias desses vetores e retornou nomes como Paulo Fernandes, Gustavo Santos ou Patrícia Souza.
- data_cadastro_dat: Foi Limitado um intervalo de datas de quando esse registro foi cadastrado que será entre 01/01/2010 e 01/05/2015 e atribuído uma data válida entre esse intervalo.
- situacaoContrato: valor aleatório entre Aberto, Baixado e Standby.

A mesma lógica foi utilizada para manipular as demais tabelas do banco de dados, e onde necessitar do id_contrato_int da tabela tcc_contrato, obviamente que somente valores já cadastrados no banco de dados foram aceitos.

Esse mesmo conceito foi aplicado nas demais entidades, e o resultado foi essa parte do banco de dados proposto, alimentado com os dados que foram obtidos ao longo da população do banco de dados. A quantidade de registros de cada entidade é apresentado na Figura 7:

Figura 7 - Tabela com totalizar de registros.

Nome da Tabela	Quantidade de registros
tcc_contrato	471.057 registros
tcc_situacaocontato	481 registros
tcc_followup	13.685.953 registros

Fonte: Elaborada pelo autor.

A partir desse momento foram aplicados alguns conceitos de performance e do banco.

Tendo esse ambiente proposto online e disponível para trabalho, foi dado início a próxima etapa do projeto. Foi fixado a necessidade de se extrair

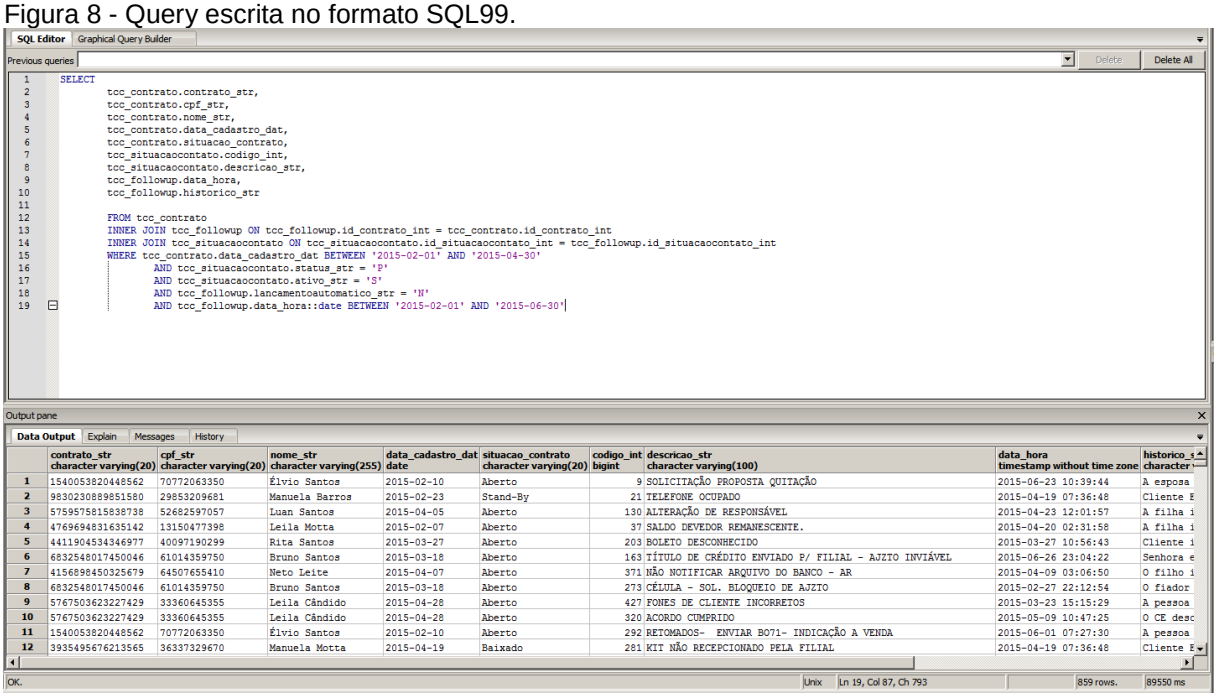
determinados registros desse banco de dados e montado uma query para executar tal rotina. Os parâmetros foram estabelecidos obedecendo a modelagem dessa camada do banco de dados, portante foi estipulado que deveria ser extraído os seguintes dados:

- Contratos com situação Aberto e Stand-By
- Clientes cadastrados entre 01/02/2015 e 30/04/2015
- Tenham recebido um follow-up POSITIVO
- Que o follow-up POSITIVO esteja ATIVO
- Que o follow-up não seja AUTOMÁTICO
- Que o follow-up tenha sido lançado entre 01/02/2015 e 30/06/2015
- Com as colunas contrato, cpf, nome, data de cadastro, situação do contrato, código da situação do contato, descrição da situação do contato, data do follow-up e histórico.

Primeiramente, foi executado uma query não otimizada, no formato SQL99 para trazer essas informações, essa mesma query foi reformulada várias vezes a cada etapa do trabalho, e esta deverá ser obtida exatamente as mesmas informações. Vale salientar, que não havia no banco nenhuma restrição de acesso, índices, amarrações com outras entidades e em nenhum momento as configurações do servidor virtual e os parâmetros do banco de dados não foram alterados para maior assertividade das informações. Foram armazenadas as informações referentes a essa geração na entidade tcc_estatisticas.

Moldando a query no formato SQL99, foi obtido o seguinte comando ilustrado na Figura 8:

Figura 8 - Query escrita no formato SQL99.



```
SELECT
        tcc_contrato.contrato_str,
        tcc_contrato.cpf_str,
        tcc_contrato.nome_str,
        tcc_contrato.data_cadastro_dat,
        tcc_contrato.situacao_contrato,
        tcc_situacaocontato.codigo_int,
        tcc_situacaocontato.descricao_str,
        tcc_followup.data_hora,
        tcc_followup.historico_str

        FROM tcc_contrato
        INNER JOIN tcc_followup ON tcc_followup.id_contrato_int = tcc_contrato.id_contrato_int
        INNER JOIN tcc_situacaocontato ON tcc_situacaocontato.id_situacaocontato_int = tcc_followup.id_situacaocontato_int
        WHERE tcc_contrato.data_cadastro_dat BETWEEN '2015-02-01' AND '2015-04-30'
                AND tcc_situacaocontato.status_str = 'P'
                AND tcc_situacaocontato.ativo_str = 'S'
                AND tcc_followup.lancamentoautomatico_str = 'N'
                AND tcc_followup.data_hora::date BETWEEN '2015-02-01' AND '2015-06-30'
```

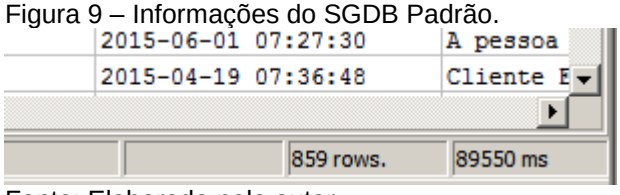
	contrato_str character varying(20)	cpf_str character varying(20)	nome_str character varying(255)	data_cadastro_dat date	situacao_contrato character varying(20)	codigo_int bigint	descricao_str character varying(100)	data_hora timestamp without time zone	historico_s character
1	1540053820448562	70772063350	Élvio Santos	2015-02-10	Aberto	9	SOLICITAÇÃO PROPOSTA QUITAÇÃO	2015-06-23 10:39:44	A esposa
2	9830230889851580	29853209681	Manuela Barros	2015-02-23	Stand-By	21	TELEFONE OCUPADO	2015-04-19 07:36:48	Cliente E
3	5795758158838738	52682597057	Luan Santos	2015-04-05	Aberto	130	ALTERAÇÃO DE RESPONSÁVEL	2015-04-23 12:01:57	A filha i
4	4769694831635142	13150477398	Leila Motta	2015-02-07	Aberto	37	SALDO DEVEDOR REMANESCENTE.	2015-04-20 02:31:58	A filha i
5	4411904534346977	40097190299	Rita Santos	2015-03-27	Aberto	203	BOLETO DESCONHECIDO	2015-03-27 10:56:43	Cliente i
6	6832548017450046	61014359750	Bruno Santos	2015-03-18	Aberto	163	TÍTULO DE CRÉDITO ENVIADO P/ FILIAL - AJZTO INVIÁVEL	2015-06-26 23:04:22	Senhora e
7	4156898450325679	64507655410	Neto Leite	2015-04-07	Aberto	371	NÃO NOTIFICAR ARQUIVO DO BANCO - AR	2015-04-09 03:06:50	O filho i
8	6832548017450046	61014359750	Bruno Santos	2015-03-18	Aberto	273	CÉLULA - SOL. BLOQUEIO DE AJZTO	2015-02-27 22:12:54	O fiador
9	5767503623227429	33360645355	Leila Cândido	2015-04-28	Aberto	427	FONES DE CLIENTE INCORRETOS	2015-03-23 15:15:29	A pessoa
10	5767503623227429	33360645355	Leila Cândido	2015-04-28	Aberto	320	ACORDO CUMPRIDO	2015-05-09 10:47:25	O CE desc
11	1540053820448562	70772063350	Élvio Santos	2015-02-10	Aberto	292	RETOMADOS- ENVIAR BO71- INDICAÇÃO A VENDA	2015-06-01 07:27:30	A pessoa
12	3935495676213565	36337329670	Manuela Motta	2015-04-19	Baixado	281	KIT NÃO RECEPCIONADO PELA FILIAL	2015-04-19 07:36:48	Cliente E

Fonte: Elaborada pelo autor.

A Figura 8 mostra o comando da query que foi desenvolvida utilizando o padrão SQL99, bem como todos os filtros que foram citados. Quando executada uma query no SGDB padrão, o retorno da consulta foi uma tabela com os registros que atendiam a necessidade imposta.

Note que no rodapé do PGAdmin III, mostra duas informações cruciais para as estatísticas conforme mostra a Figura 9, que é o tempo em milissegundos de geração da consulta SQL e a quantidade de registros que essa query retornou os resultados. A cada passo que foi avançado, foram armazenadas essas informações numa tabela a parte (tcc_estatisticas) para que no final possa ser avaliado o desempenho em cada uma das situações.

Figura 9 – Informações do SGDB Padrão.



Fonte: Elaborada pelo autor.

A quantidade de registros, o tempo em milissegundos e a descrição dessa execução foram armazenadas na entidade tcc_estatisticas, a Figura 10 mostra como ficou o primeiro registro que foi incluso nessa entidade.

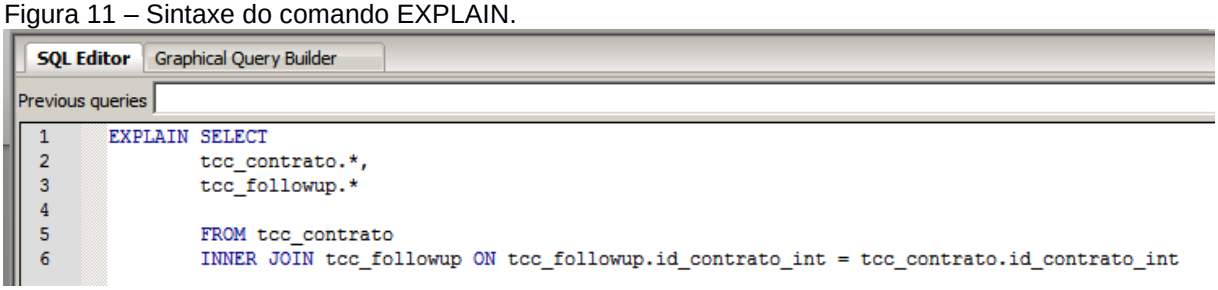
Figura 10 - Registro Inserido da tcc_estatisticas.

estatistica_tstamp timestamp without time zone	estatistica_descricao character varying(200)	estatistica_tempo_geracao character varying(50)	estatistica_quantidade_registro character varying(50)
2015-11-13 09:51:29.362	1) SELECT do relatório formato SQL99	89550 milissegundos	859 registros

Fonte: Elaborada pelo autor.

Um recurso nativo do Postgres que auxilia no entendimento do comportamento do servidor ao executar uma query, a plataforma fornece o comando EXPLAIN, que basicamente informa o comportamento do núcleo do banco de dados para que possam prever algumas situações, e até mesmo indicar possíveis problemas na estrutura em que foi modelado, esse método específico da tecnologia indica como organizar, classificar e prever os registros da base. Veja na Figura 11 um exemplo do uso do comando EXPLAIN com JOIN entre duas entidades.

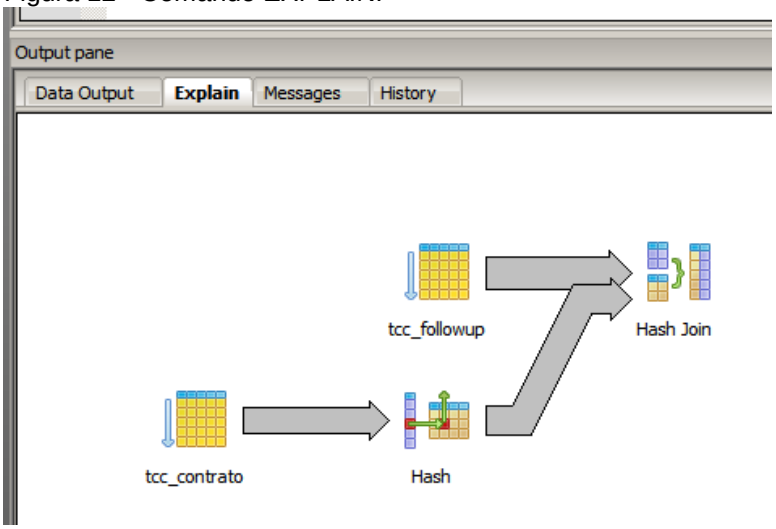
Figura 11 – Sintaxe do comando EXPLAIN.



Fonte: Elaborada pelo autor.

Assim que executada a rotina, bastou adicionar o comando EXPLAIN antes da consulta SQL para que o PostgreSQL retornasse o comportamento do servidor como mostra a Figura 12.

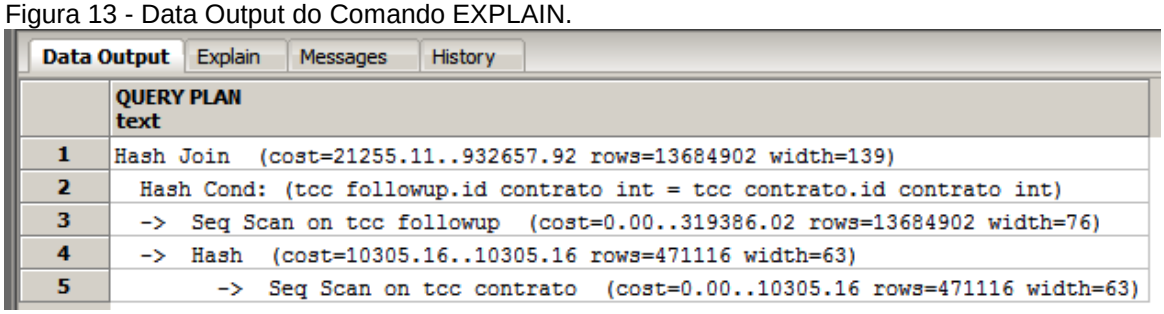
Figura 12 - Comando EXPLAIN.



Fonte: Elaborada pelo autor.

A Figura 13 mostra a saída no modo texto do comando EXPLAIN, e pode se extrair muitas informações desse recurso principalmente quando for uma query extensa e deseja verificar se os índices estão sendo computados, para alimentar a entidade tcc_estatisticas, foi utilizado somente o tempo em milissegundos e a quantidade de registros retornados, portanto a análise das informações contidas nessa Figura 13 não foram utilizadas.

Figura 13 - Data Output do Comando EXPLAIN.



Data Output | Explain | Messages | History

	QUERY PLAN text
1	Hash Join (cost=21255.11..932657.92 rows=13684902 width=139)
2	Hash Cond: (tcc followup.id contrato int = tcc contrato.id contrato int)
3	-> Seq Scan on tcc followup (cost=0.00..319386.02 rows=13684902 width=76)
4	-> Hash (cost=10305.16..10305.16 rows=471116 width=63)
5	-> Seq Scan on tcc contrato (cost=0.00..10305.16 rows=471116 width=63)

Fonte: Elaborada pelo autor.

O comando EXPLAIN (nativo do Postgres que aqui foi utilizado) é na verdade uma pergunta para o SGDB de como seria o comportamento ao se executar determinada consulta.

O comando EXPLAIN usa também como base, sua própria fonte de informação (“pg_stats()”), que registra dados de sua modelagem, quantidade de registros, posicionamento de índices, estrutura de funções, dados de restrição de usuário entre outras informações que devem ser atualizadas constantemente junto ao processo de backup do banco de dados. Para o caso do SQL do relatório proposto de acordo com a Figura 13, lendo o retorno do SGBD, teve o custo estimado entre 21255.11 e 932657.92 leituras de páginas de disco com tamanho igual a 139 bytes. Ter conhecimento sobre esse recurso poupa tempo de debug e ajuda o desenvolvedor diagnosticar onde está uma possível lentidão antes mesmo de acontecer e uma escrita assim ir para o ambiente de produção do software. A verificação constante do comando EXPLAIN visa monitorar a desenvoltura e comportamento do banco de dados ainda num ambiente de teste e homologação.

Para manter essa base de tuplas (leitura de páginas de disco) bem como as tabelas de estatísticas nativas do PostgreSQL, foi utilizado um comando que deve ser monitorado pelo DBA da instituição por se tratar de um comando que normalmente é executado logo após o backup do banco de dados e pelo motivo de que enquanto o comando estiver em execução, as tabelas consultadas estarão com

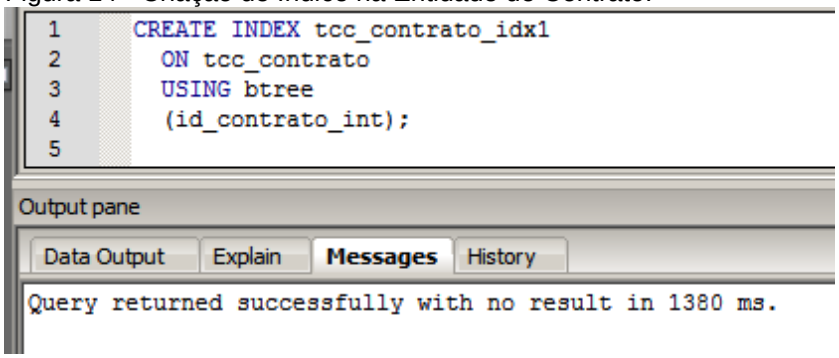
LOCK e não podem receber nenhuma DML de INSERT E UPDATE. Será utilizado o comando VACUUM. Esse comando possui uma árvore de derivações que deve ser cuidadosamente estudada pelo administrador do banco de dados, e entender exatamente o conceito de cada uma de suas ramificações, mas no caso será utilizado o VACUUM ANALYSE [table], quando omitido o termo [table] o mesmo será aplicado a todas as entidades pertencentes a esse banco.

Esse comando fez uma varredura do banco de dados, organizou os registros, atualizou a tabela de estatísticas e desfragmentar os registros. O procedimento é aconselhado que seja executado após a rotina de backup do servidor (que deveria ser diário segundo ISO 27000 e 27001). A utilização do comando EXPLAIN em conjunto com o comando VACUUM são recursos pouco empregados nos sistemas de software house desenvolvidos com essa tecnologia de banco de dados. Até o momento não alterado significativamente o banco de dados para que tenha uma melhor performance, apenas apresentado o que pode ser feito para auxiliar na busca uma possível lentidão de banco de dados.

Uma otimização notável de grande diferença no quesito performance, são os índices. Índices no banco de dados servem os quais como referência para que um servidor encontre o registro necessário num menor espaço de tempo. Esse atributo, geralmente se aplica a campos numéricos, chaves primárias e chaves estrangeiras do banco de dados para que quando houver uma interação entre as tabelas o núcleo do nosso servidor possa manipular as informações de forma mais rápida.

Foi criado então o primeiro índice na tabela tcc_contrato, no campo idContrato, e este serviu de chave estrangeira para as demais entidades do modelo relacional, como mostra a Figura 14.

Figura 14 - Criação de Índice na Entidade de Contrato.

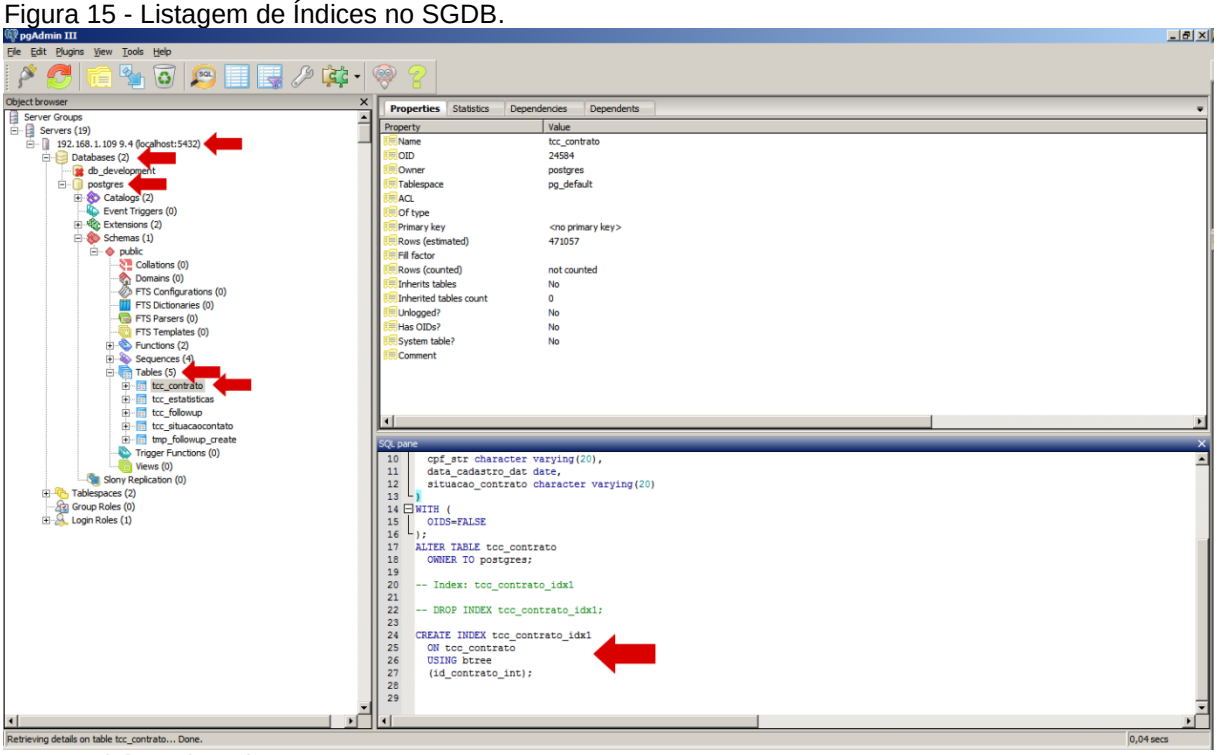


Fonte: Elaborada pelo autor.

Logo foi exibida a mensagem de sucesso no SGDB padrão.

Para saber quais índices foram empregados na entidade, basta navegar pelo PGAdmin III até encontrar a entidade, e o comando do índice estará no rodapé da estrutura.

Figura 15 - Listagem de Índices no SGDB.



Fonte: Elaborada pelo autor.

Foram criados os índices nas demais entidades do banco de dados como mostra a Figura 16:

Figura 16 - Criação dos índices necessários.

```
1  CREATE INDEX tcc_contrato_idx2 ON tcc_contrato USING btree (data_cadastro_dat);
2  CREATE INDEX tcc_followup_idx1 ON tcc_followup USING btree (id_contrato_int);
3  CREATE INDEX tcc_followup_idx2 ON tcc_followup USING btree (id_situacaocontato_int);
4  CREATE INDEX tcc_followup_idx3 ON tcc_followup USING btree (data_hora);
5  CREATE INDEX tcc_situacaocontato_idx1 ON tcc_situacaocontato USING btree (id_situacaocontato_int);
```

Fonte: Elaborada pelo autor.

Para verificar o que mudou no banco de dados, foi executada a mesma query da Figura 8 para a consulta das informações em conjunto com o comando EXPLAIN e foram analisados os resultados.

Foram armazenadas essas informações na tabela tcc_estatisticas. Com isso foi notada significativamente o ganho de performance na geração do relatório. Veja a comparação entre os dois registros que foram inclusos na entidade tcc_estatisticas.

Figura 17 - Entidade tcc_estatisticas.

estatistica_tstamp timestamp without time zone	estatistica_descricao character varying(200)	estatistica_tempo_geracao character varying(50)	estatistica_quantidade_registro character varying(50)
2015-11-13 09:51:29.362	1) SELECT do relatório formato SQL99	89550 milissegundos	859 registros
2015-11-13 10:35:44.239	2) SELECT do relatório formato SQL99 com índices	27831 milissegundos	859 registros

Fonte: Elaborada pelo autor.

Depois de todas as entidades já indexadas, foi executado o comando VACUUM ANALYSE para que as alterações fossem consideradas pelo pg_stats() no PostgreSQL.

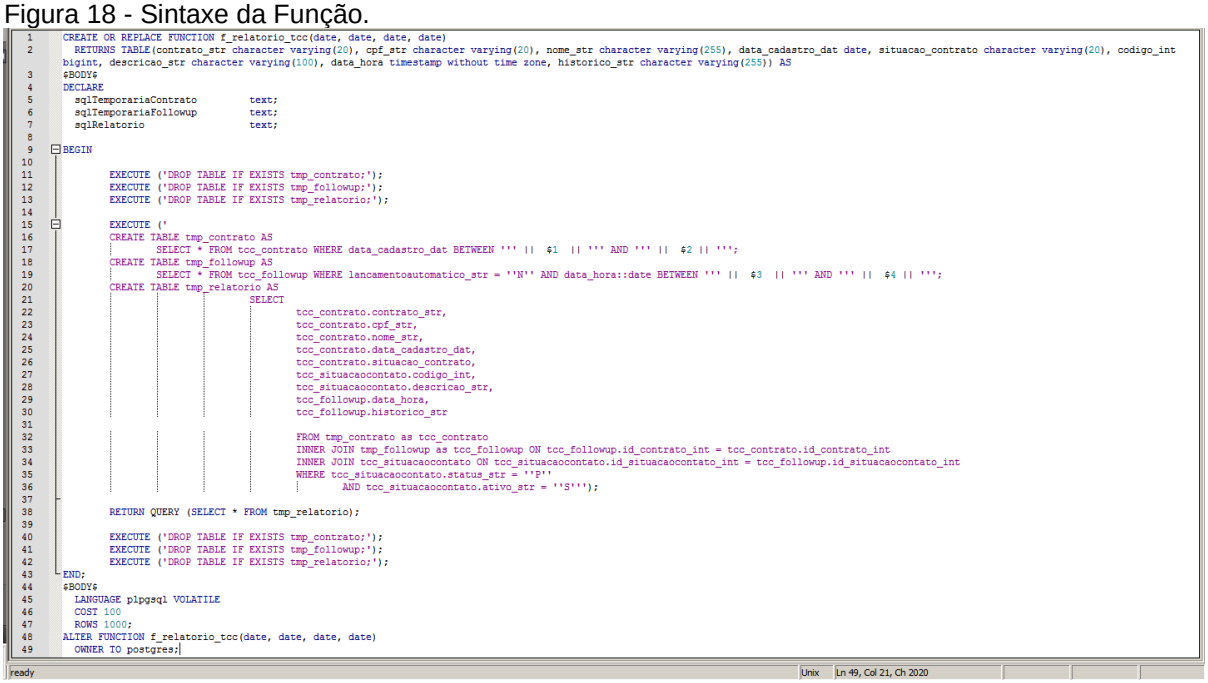
Foi executada novamente a primeira versão da query de consulta aos registros juntamente com o comando EXPLAIN, e analisado os resultados obtidos até então e se houve ou não o ganho de performance com as entidades indexadas e as estatísticas atualizadas. Foram novamente armazenadas essas informações na entidade tcc_estatisticas e adicionada a observação de que o comando VACUUM ANALYSE fora executado.

A partir desse momento, foi manipulada a forma de escrita de consulta ao banco de dados, foi alterada a forma de escrita para que fosse tirado um maior proveito da modelagem. Essa reescrita do comando, teve que obrigatoriamente devolver os mesmos registros da consulta anterior, e foi verificado se o tempo de geração teve alteração para mais ou menos tempo. Foi utilizado agora ao invés de uma simples consulta em SQL, o uso de FUNCTIONS para que pudesse dar mais liberdade e recursos na escrita desse relatório.

Foi elaborado então, o conceito da consulta visando as tabelas auxiliares para que possa chegar no resultado do trabalho. Levando em consideração que há muita informação no banco de dados e se faz necessário extrair apenas alguns poucos registros em determinada situação, acaba não sendo útil, executar grandes queries complexas para obter essas informações.

Para filtrar as informações de follow-up dos contratos que foram cadastrados em determinado período, não é uma boa prática unir as duas tabelas com milhares de registros, pois isso seria muito custoso para o servidor resolver a query com essa quantidade de registros envolvidos, portanto foram criadas então as tabelas temporárias que até o momento são físicas e foram sendo gravadas no disco, mas com os registros que foram utilizados, essa criação teve que seguir o modelo relacional lógico e procedural, e o comando obtido foi ilustrado na Figura 18:

Figura 18 - Sintaxe da Função.

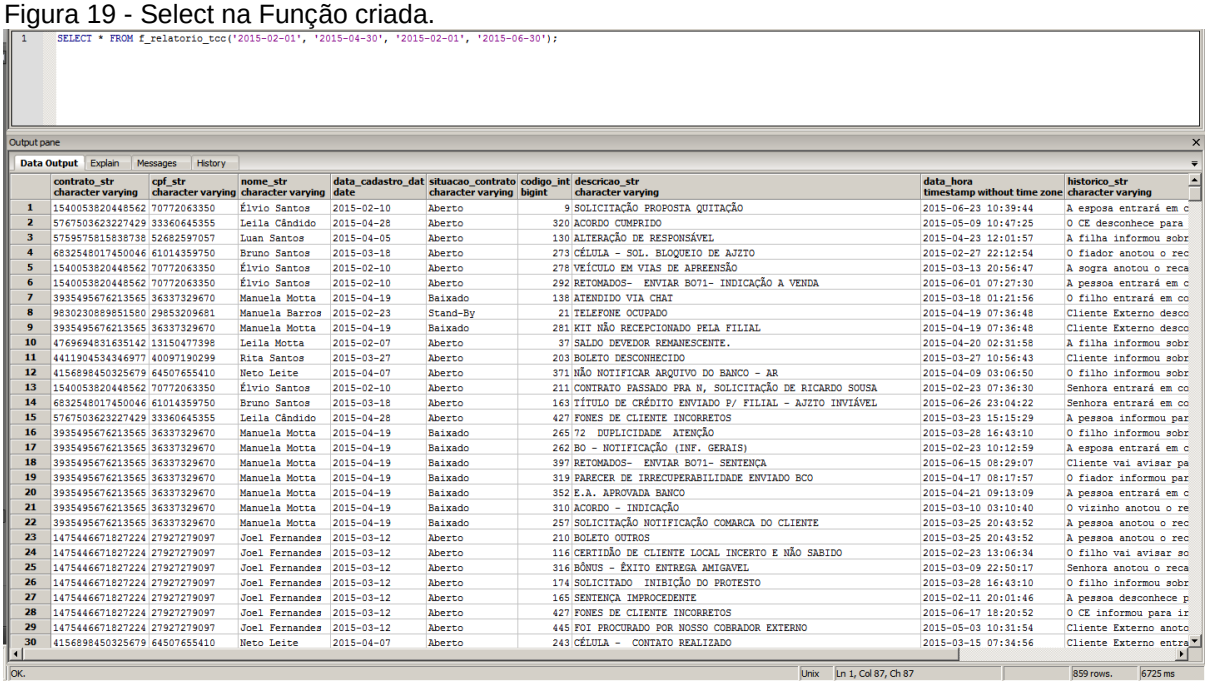


```
CREATE OR REPLACE FUNCTION f_relatorio_tcc(date, date, date, date)
  RETURNS TABLE(contrato_str character varying(20), cpf_str character varying(20), nome_str character varying(255), data_cadastro_dat date, situacao_contrato character varying(20), codigo_int
bigint, descricao_str character varying(100), data_hora timestamp without time zone, historico_str character varying(255)) AS
$BODY$
DECLARE
  sqlTemporariaContrato         text;
  sqlTemporariaFollowup         text;
  sqlRelatorio                  text;

BEGIN

        EXECUTE ('DROP TABLE IF EXISTS tmp_contrato;');
        EXECUTE ('DROP TABLE IF EXISTS tmp_followup;');
        EXECUTE ('DROP TABLE IF EXISTS tmp_relatorio;');

        EXECUTE ('
        CREATE TABLE tmp_contrato AS
                SELECT * FROM tcc_contrato WHERE data_cadastro_dat BETWEEN ''' ||  $1  || ''' AND ''' ||  $2 || ''';
        CREATE TABLE tmp_followup AS
                SELECT * FROM tcc_followup WHERE lancamentoautomatico_str = ''N'' AND data_hora::date BETWEEN ''' ||  $3  || ''' AND ''' ||  $4 || ''';
        CREATE TABLE tmp_relatorio AS
                        SELECT
                                tcc_contrato.contrato_str,
                                tcc_contrato.cpf_str,
                                tcc_contrato.nome_str,
                                tcc_contrato.data_cadastro_dat,
                                tcc_contrato.situacao_contrato,
                                tcc_situacaocontato.codigo_int,
                                tcc_situacaocontato.descricao_str,
                                tcc_followup.data_hora,
                                tcc_followup.historico_str

                                FROM tmp_contrato as tcc_contrato
                                INNER JOIN tmp_followup as tcc_followup ON tcc_followup.id_contrato_int = tcc_contrato.id_contrato_int
                                INNER JOIN tcc_situacaocontato ON tcc_situacaocontato.id_situacaocontato_int = tcc_followup.id_situacaocontato_int
                                WHERE tcc_situacaocontato.status_str = ''P''
                                        AND tcc_situacaocontato.ativo_str = ''S''');

        RETURN QUERY (SELECT * FROM tmp_relatorio);

        EXECUTE ('DROP TABLE IF EXISTS tmp_contrato;');
        EXECUTE ('DROP TABLE IF EXISTS tmp_followup;');
        EXECUTE ('DROP TABLE IF EXISTS tmp_relatorio;');
END;
$BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100
  ROWS 1000;
ALTER FUNCTION f_relatorio_tcc(date, date, date, date)
  OWNER TO postgres;
```

Fonte: Elaborada pelo autor.

Para executar o relatório a partir desse momento, foi executado uma DML SELECT na função que foi criada, os parâmetros são obrigatórios para a chamada da função e foi ilustrado na Figura 19:

Figura 19 - Select na Função criada.



```
SELECT * FROM f_relatorio_tcc('2015-02-01', '2015-04-30', '2015-02-01', '2015-06-30');
```

Output pane — Data Output | Explain | Messages | History

	contrato_str character varying	cpf_str character varying	nome_str character varying	data_cadastro_dat date	situacao_contrato character varying	codigo_int bigint	descricao_str character varying	data_hora timestamp without time zone	historico_str character varying
1	15400538204485 62	70772063350	Élvio Santos	2015-02-10	Aberto	9	SOLICITAÇÃO PROPOSTA QUITAÇÃO	2015-06-23 10:39:44	A esposa entrará em c
2	57675036232274 29	33360645355	Leila Cândido	2015-04-28	Aberto	320	ACORDO CUMPRIDO	2015-05-09 10:47:25	O CE desconhece para
3	57595758158387 38	52682597057	Luan Santos	2015-04-05	Aberto	130	ALTERAÇÃO DE RESPONSÁVEL	2015-04-23 12:01:57	A filha informou sobr
4	68325480174500 46	61014359750	Bruno Santos	2015-03-18	Aberto	273	CÉLULA – SOL. BLOQUEIO DE AJZTO	2015-02-27 22:12:54	O fiador anotou o rec
5	15400538204485 62	70772063350	Élvio Santos	2015-02-10	Aberto	278	VEÍCULO EM VIAS DE APREENSÃO	2015-03-13 20:56:47	A sogra anotou o reca
6	15400538204485 62	70772063350	Élvio Santos	2015-02-10	Aberto	292	RETOMADOS- ENVIAR BO71- INDICAÇÃO A VENDA	2015-06-01 07:27:30	A pessoa entrará em c
7	39354956762135 65	36337329670	Manuela Motta	2015-04-19	Baixado	138	ATENDIDO VIA CHAI	2015-03-18 01:21:56	O filho entrará em co
8	98302308898515 80	29853209681	Manuela Barros	2015-02-23	Stand-By	21	TELEFONE OCUPADO	2015-04-19 07:36:48	Cliente Externo desco
9	39354956762135 65	36337329670	Manuela Motta	2015-04-19	Baixado	281	KIT NÃO RECEPCIONADO PELA FILIAL	2015-04-19 07:36:48	Cliente Externo desco
10	47696948316351 42	13150477398	Leila Motta	2015-02-07	Aberto	37	SALDO DEVEDOR REMANESCENTE.	2015-04-20 02:31:58	A filha informou sobr
11	44119045343469 77	40097190299	Rita Santos	2015-03-27	Aberto	203	BOLETO DESCONHECIDO	2015-03-27 10:56:43	Cliente informou sobr
12	41568984503256 79	64507655410	Neto Leite	2015-04-07	Aberto	371	NÃO NOTIFICAR ARQUIVO DO BANCO – AR	2015-04-09 03:06:50	O filho informou sobr
13	15400538204485 62	70772063350	Élvio Santos	2015-02-10	Aberto	211	CONTRATO PASSADO PRA N, SOLICITAÇÃO DE RICARDO SOUSA	2015-02-23 07:36:30	Senhora entrará em co
14	68325480174500 46	61014359750	Bruno Santos	2015-03-18	Aberto	163	TÍTULO DE CRÉDITO ENVIADO P/ FILIAL – AJZTO INVIÁVEL	2015-06-26 23:04:22	Senhora entrará em co
15	57675036232274 29	33360645355	Leila Cândido	2015-04-28	Aberto	427	FONES DE CLIENTE INCORRETOS	2015-03-23 15:15:29	A pessoa informou par
16	39354956762135 65	36337329670	Manuela Motta	2015-04-19	Baixado	265	72 DUPLICIDADE ATENÇÃO	2015-03-28 16:43:10	O filho informou sobr
17	39354956762135 65	36337329670	Manuela Motta	2015-04-19	Baixado	262	BO – NOTIFICAÇÃO (INF. GERAIS)	2015-02-23 10:12:59	A esposa entrará em c
18	39354956762135 65	36337329670	Manuela Motta	2015-04-19	Baixado	397	RETOMADOS- ENVIAR BO71- SENTENÇA	2015-06-15 08:29:07	Cliente vai avisar pa
19	39354956762135 65	36337329670	Manuela Motta	2015-04-19	Baixado	319	PARECER DE IRRECUPERABILIDADE ENVIADO BCO	2015-04-17 08:17:57	O fiador informou par
20	39354956762135 65	36337329670	Manuela Motta	2015-04-19	Baixado	352	E.A. APROVADA BANCO	2015-04-21 09:13:09	A pessoa entrará em c
21	39354956762135 65	36337329670	Manuela Motta	2015-04-19	Baixado	310	ACORDO – INDICAÇÃO	2015-03-10 03:10:40	O vizinho anotou o re
22	39354956762135 65	36337329670	Manuela Motta	2015-04-19	Baixado	257	SOLICITAÇÃO NOTIFICAÇÃO COMARCA DO CLIENTE	2015-03-25 20:43:52	A pessoa anotou o rec
23	14754466718272 24	27927279097	Joel Fernandes	2015-03-12	Aberto	210	BOLETO OUTROS	2015-03-25 20:43:52	A pessoa anotou o rec
24	14754466718272 24	27927279097	Joel Fernandes	2015-03-12	Aberto	116	CERTIDÃO DE CLIENTE LOCAL INCERTO E NÃO SABIDO	2015-02-23 13:06:34	O filho vai avisar so
25	14754466718272 24	27927279097	Joel Fernandes	2015-03-12	Aberto	316	BÔNUS – ÊXITO ENTREGA AMIGAVEL	2015-03-09 22:50:17	Senhora anotou o reca
26	14754466718272 24	27927279097	Joel Fernandes	2015-03-12	Aberto	174	SOLICITADO INIBIÇÃO DO PROTESTO	2015-03-28 16:43:10	O filho informou sobr
27	14754466718272 24	27927279097	Joel Fernandes	2015-03-12	Aberto	165	SENTENÇA IMPROCEDENTE	2015-02-11 20:01:46	A pessoa desconhece p
28	14754466718272 24	27927279097	Joel Fernandes	2015-03-12	Aberto	427	FONES DE CLIENTE INCORRETOS	2015-06-17 18:20:52	O CE informou para ir
29	14754466718272 24	27927279097	Joel Fernandes	2015-03-12	Aberto	445	FOI PROCURADO POR NOSSO COBRADOR EXTERNO	2015-05-03 10:31:54	Cliente Externo anoto
30	41568984503256 79	64507655410	Neto Leite	2015-04-07	Aberto	243	CÉLULA – CONTATO REALIZADO	2015-03-15 07:34:56	Cliente Externo entra

OK. Unix Ln 1, Col 87, Ch 87 859 rows. 6725 ms

Fonte: Elaborada pelo autor.

Não se faz necessário o uso do comando EXPLAIN a partir desse momento, tendo em vista que o SGDB agora forneceu apenas uma consulta a FUNCTION

criada, e não o que está sendo manipulado e consultado dentro da FUNCTION especificadamente.

Foi alterada então a função para que utilizasse TEMPORARY TABLE. Uma informação importante é que a tabela temporária, diferentemente da tabela física, fica armazenada na memória e cache do servidor, e por esse motivo a manipulação é mais rápida e ela só existe enquanto a conexão estiver ativa, após a conexão ser fechada, a tabela será desprezada da memória e do cache, o código fonte da função ficou como ilustrado na Figura 20:

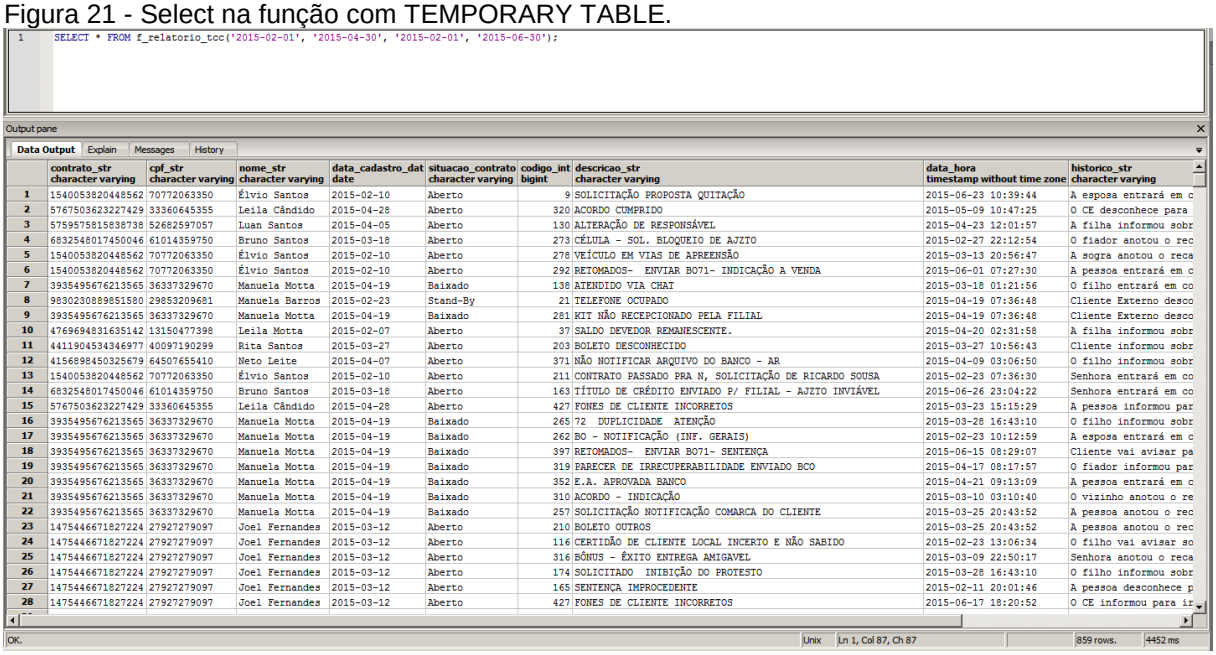
Figura 20 - Sintaxe da função com TEMPORARY TABLE.

```
CREATE OR REPLACE FUNCTION f_relatorio_tcc(date, date, date, date)
  RETURNS TABLE(contrato_str character varying(20), cpf_str character varying(20), nome_str character varying(255), data_cadastro_dat date, situacao_contrato character varying(20), codigo_int
bigint, descricao_str character varying(100), data_hora timestamp without time zone, historico_str character varying(255)) AS
$BODY$
DECLARE
  sqlTemporariaContrato         text;
  sqlTemporariaFollowup         text;
  sqlRelatorio                  text;

BEGIN

        EXECUTE ('
        CREATE TEMPORARY TABLE tmp_contrato AS
                SELECT * FROM tcc_contrato WHERE data_cadastro_dat BETWEEN ''' ||  $1  || ''' AND ''' ||  $2 || ''';

        CREATE TEMPORARY TABLE tmp_followup AS
                SELECT * FROM tcc_followup WHERE lancamentoautomatico_str = ''N'' AND data_hora::date BETWEEN ''' ||  $3  || ''' AND ''' ||  $4 || ''';

        CREATE TEMPORARY TABLE tmp_relatorio AS
                        SELECT
                                tcc_contrato.contrato_str,
                                tcc_contrato.cpf_str,
                                tcc_contrato.nome_str,
                                tcc_contrato.data_cadastro_dat,
                                tcc_contrato.situacao_contrato,
                                tcc_situacaocontato.codigo_int,
                                tcc_situacaocontato.descricao_str,
                                tcc_followup.data_hora,
                                tcc_followup.historico_str

                                FROM tmp_contrato as tcc_contrato
                                INNER JOIN tmp_followup as tcc_followup ON tcc_followup.id_contrato_int = tcc_contrato.id_contrato_int
                                INNER JOIN tcc_situacaocontato ON tcc_situacaocontato.id_situacaocontato_int = tcc_followup.id_situacaocontato_int
                                WHERE tcc_situacaocontato.status_str = ''P''
                                        AND tcc_situacaocontato.ativo_str = ''S''');

        RETURN QUERY (SELECT * FROM tmp_relatorio);

END;
$BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100
  ROWS 1000;
ALTER FUNCTION f_relatorio_tcc(date, date, date, date)
  OWNER TO postgres;
```

Fonte: Elaborada pelo autor.

Novamente foi executado a DML SELECT na função passando os mesmos parâmetros que haviam sido utilizados anteriormente, e o resultado da geração da query foi ilustrador na Figura 21:

Figura 21 - Select na função com TEMPORARY TABLE.



```
SELECT * FROM f_relatorio_tcc('2015-02-01', '2015-04-30', '2015-02-01', '2015-06-30');
```

Output pane — Data Output | Explain | Messages | History

	contrato_str character varying	cpf_str character varying	nome_str character varying	data_cadastro_dat date	situacao_contrato character varying	codigo_int bigint	descricao_str character varying	data_hora timestamp without time zone	historico_str character varying
1	1540053820448562	70772063350	Élvio Santos	2015-02-10	Aberto	9	SOLICITAÇÃO PROPOSTA QUITAÇÃO	2015-06-23 10:39:44	A esposa entrará em c
2	5767503623227429	33360645355	Leila Cândido	2015-04-28	Aberto	320	ACORDO CUMPRIDO	2015-05-09 10:47:25	O CE desconhece para
3	5759575815838738	52682597057	Luan Santos	2015-04-05	Aberto	130	ALTERAÇÃO DE RESPONSÁVEL	2015-04-23 12:01:57	A filha informou sobr
4	68325480174500046	61014359750	Bruno Santos	2015-03-18	Aberto	273	CÉLULA - SOL. BLOQUEIO DE AJZTO	2015-02-27 22:12:54	O fiador anotou o rec
5	1540053820448562	70772063350	Élvio Santos	2015-02-10	Aberto	278	VEÍCULO EM VIAS DE APREENSÃO	2015-03-13 20:56:47	A sogra anotou o reca
6	1540053820448562	70772063350	Élvio Santos	2015-02-10	Aberto	292	RETOMADOS- ENVIAR BO71- INDICAÇÃO A VENDA	2015-06-01 07:27:30	A pessoa entrará em c
7	39354956762135 65	36337329670	Manuela Motta	2015-04-19	Baixado	138	ATENDIDO VIA CHAT	2015-03-18 01:21:56	O filho entrará em co
8	98302308898515 80	29853209681	Manuela Barros	2015-02-23	Stand-By	21	TELEFONE OCUPADO	2015-04-19 07:36:48	Cliente Externo desco
9	39354956762135 65	36337329670	Manuela Motta	2015-04-19	Baixado	281	KIT NÃO RECEPCIONADO PELA FILIAL	2015-04-19 07:36:48	Cliente Externo desco
10	47696948316351 42	13150477398	Leila Motta	2015-02-07	Aberto	37	SALDO DEVEDOR REMANESCENTE.	2015-04-20 02:31:58	A filha informou sobr
11	44119045343469 77	40097190299	Rita Santos	2015-03-27	Aberto	203	BOLETO DESCONHECIDO	2015-03-27 10:56:43	Cliente informou sobr
12	41568984503256 79	64507655410	Neto Leite	2015-04-07	Aberto	371	NÃO NOTIFICAR ARQUIVO DO BANCO - AR	2015-04-09 03:06:50	O filho informou sobr
13	1540053820448562	70772063350	Élvio Santos	2015-02-10	Aberto	211	CONTRATO PASSADO PRA N, SOLICITAÇÃO DE RICARDO SOUSA	2015-02-23 07:36:30	Senhora entrará em co
14	68325480174500046	61014359750	Bruno Santos	2015-03-18	Aberto	163	TÍTULO DE CRÉDITO ENVIADO P/ FILIAL - AJZTO INVIÁVEL	2015-06-26 23:04:22	Senhora entrará em co
15	5767503623227429	33360645355	Leila Cândido	2015-04-28	Aberto	427	FONES DE CLIENTE INCORRETOS	2015-03-23 15:15:29	A pessoa informou par
16	39354956762135 65	36337329670	Manuela Motta	2015-04-19	Baixado	265	72 DUPLICIDADE ATENÇÃO	2015-03-28 16:43:10	O filho informou sobr
17	39354956762135 65	36337329670	Manuela Motta	2015-04-19	Baixado	262	BO - NOTIFICAÇÃO (INF. GERAIS)	2015-02-23 10:12:59	A esposa entrará em c
18	39354956762135 65	36337329670	Manuela Motta	2015-04-19	Baixado	397	RETOMADOS- ENVIAR BO71- SENTENÇA	2015-06-15 08:29:07	Cliente vai avisar pa
19	39354956762135 65	36337329670	Manuela Motta	2015-04-19	Baixado	319	PARECER DE IRRECUPERABILIDADE ENVIADO BCO	2015-04-17 08:17:57	O fiador informou par
20	39354956762135 65	36337329670	Manuela Motta	2015-04-19	Baixado	352	E.A. APROVADA BANCO	2015-04-21 09:13:09	A pessoa entrará em c
21	39354956762135 65	36337329670	Manuela Motta	2015-04-19	Baixado	310	ACORDO - INDICAÇÃO	2015-03-10 03:10:40	O vizinho anotou o re
22	39354956762135 65	36337329670	Manuela Motta	2015-04-19	Baixado	257	SOLICITAÇÃO NOTIFICAÇÃO COMARCA DO CLIENTE	2015-03-25 20:43:52	A pessoa anotou o rec
23	1475446671827224	27927279097	Joel Fernandes	2015-03-12	Aberto	210	BOLETO OUTROS	2015-03-25 20:43:52	A pessoa anotou o rec
24	1475446671827224	27927279097	Joel Fernandes	2015-03-12	Aberto	116	CERTIDÃO DE CLIENTE LOCAL INCERTO E NÃO SABIDO	2015-02-23 13:06:34	O filho vai avisar so
25	1475446671827224	27927279097	Joel Fernandes	2015-03-12	Aberto	316	BÔNUS - ÊXITO ENTREGA AMIGAVEL	2015-03-09 22:50:17	Senhora anotou o reca
26	1475446671827224	27927279097	Joel Fernandes	2015-03-12	Aberto	174	SOLICITADO INIBIÇÃO DO PROTESTO	2015-03-28 16:43:10	O filho informou sobr
27	1475446671827224	27927279097	Joel Fernandes	2015-03-12	Aberto	165	SENTENÇA IMPROCEDENTE	2015-02-11 20:01:46	A pessoa desconhece p
28	1475446671827224	27927279097	Joel Fernandes	2015-03-12	Aberto	427	FONES DE CLIENTE INCORRETOS	2015-06-17 18:20:52	O CE informou para ir

OK. | Unix | Ln 1, Col 87, Ch 87 | 859 rows. | 4452 ms

Fonte: Elaborada pelo autor.

E por mais uma vez, as estatísticas foram recolhidas e inseridas na entidade tcc_estatisticas onde foram confrontados os dados de todo o processo e foi demonstrado no próximo capítulo.

Os registros obtidos na entidade tcc_estatisticas estão expostos na Figura 22 onde a primeira vista já informa um ganho de desempenho considerável.

Figura 22 - Registros da entidade tcc_estatisticas.

estatistica_tstamp timestamp without time zone	estatistica_descricao character varying(200)	estatistica_tempo_geracao character varying(50)	estatistica_quantidade_registro character varying(50)
2015-11-13 09:51:29.362	1) SELECT do relatório formato SQL99	89550 milissegundos	859 registros
2015-11-13 10:35:44.239	2) SELECT do relatório formato SQL99 com índices	27831 milissegundos	859 registros
2015-11-13 16:42:44.761	3) SELECT na função com tabela física temporária	6725 milissegundos	859 registros
2015-11-13 16:52:55.067	4) SELECT na função com tabela temporária em cache	4452 milissegundos	859 registros

Fonte: Elaborada pelo autor.

6 Resultados

Para demonstrar a efetividade da técnica de uso de tabelas temporárias é efetivo ou não, foram contabilizados os registros em tabelas e gráficos e analisar se a mesma quantidade de registros obtidos foi a mesmo e qual foi a janela de tempo entre as etapas de geração das queries.

A Figura 23 exibe os registros que foram obtidos no decorrer das etapas de geração de cada query, é possível ver ganho significativo.

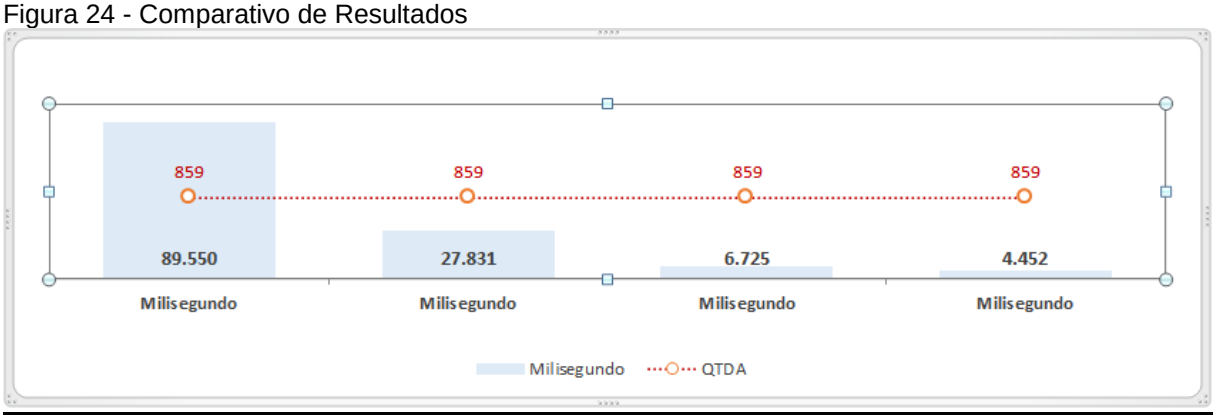
Figura 23 - Tabela de Resumo das Estatísticas

	A	B	C
1	#	Tempo em ms	Qtde Registros
2	1º Geração	89.550	859
3	2º Geração	27.831	859
4	3º Geração	6.725	859
5	4º Geração	4.452	859

Fonte: Elaborada pelo autor.

Na Figura 23 pode-se notar que a quantidade de registros manteve-se inalterada, porém o tempo de execução baixou consideravelmente a cada item que a otimização.

Figura 24 - Comparativo de Resultados



Fonte: Elaborada pelo autor.

Confrontando o tempo obtido em cada geração, e comparando como forma de economia de tempo entre elas, foi constatado um ganho de 95% da 4º geração

da query em relação a 1º como ilustra a Figura 25. Foi feito também uma comparação em relação à geração anterior, e a economia de tempo entre elas.

Figura 25 - Economia de Tempo (ms) entre as Gerações

	A	B	C	D
1		Tempo Total (ms)	Ganho em relação a Geração Anterior(%)	Ganho em relação a 1º Geração(%)
2	1º Geração	89.550	0%	0%
3	2º Geração	27.831	-69%	-69%
4	3º Geração	6.725	-76%	-92%
5	4º Geração	4.452	-34%	-95%

Fonte: Elaborada pelo autor.

A Figura 25 mostra na coluna C, a porcentagem de ganho de tempo em relação à geração anterior, e como a 1º geração não havia registro anterior, ficou como 0%, e logo que as primeiras modificações foram executadas, houve uma economia no tempo geração o que foi chamado de ganho de 69% do tempo, ou seja, 61.719 milissegundos a menos para executar uma consulta dos mesmos registros.

A diferença se tornou ainda mais significativa quando comparado os extremos cenários (o cenário de início e de término) onde o ganho de tempo alcançou a casa dos 95%. Isso significa que a query foi executada em apenas 5% do tempo original totalizando em apenas 4.452 milissegundos.

7 Considerações Finais

A utilização de tabelas temporárias são sim efetivas para aumentar a performance do banco de dados relacional, tendo em vista o baixo consumo para resolver as queries internas de uma função e o tempo gasto com uma um banco sem índice, sem atualização das tabelas de estatísticas (pg_stats()) e sem uma query otimizada pode levar a uma finalização precoce de um sistema seja ele legado ou não.

Foi aferido que sem alteração nenhuma no hardware é possível ter um ganho significativo de ganho de desempenho o que poderia prolongar a vida útil de uma ferramenta, ou ainda remodelar módulos isolados de determinado sistema visando uma melhor organização do código fonte bem como mantendo dentro do novo paradigma.

O Postgres foi executado dentro de uma máquina virtual com poucos recursos e mesmo assim mostrou um desempenho altíssimo tendo em vista a quantidade de registros que foram alocados e ainda processando aplicações a parte do sistema operacional e o serviço do banco de dados (como o PHP, Apache, UFW entre outros). A máquina virtual estava rodando num disco SSD com alta taxa de leitura e gravação quando comparado com disco SATAII, mas a técnica apresentada e os recursos utilizados podem ser utilizados em qualquer tipo de disco e praticamente em qualquer plataforma de banco de dados, desde que obedeça as suas particularidades. Bancos de dados pagos, como o caso do SQL SERVER da Microsoft ou ORACLE, oferecem ferramentas próprias para atualização de tabelas de estatísticas e para backup do banco, portanto onde foi usado comando EXPLAIN e VACCUM, deve ser pesquisado por comandos similares em cada plataforma.

O SGDB padrão do Postgres, o PgAdmin III 1.20.0 mostrou ser bastante estável e leve mesmo quando executado uma query com grande quantidade de registros.

REFERÊNCIAS

ELMASRI, R. E NAVATHE, S. B. **Sistema de Banco de Dados**, São Paulo: Pearson Education, 2006.

HEUSER, C. A. **Projeto de Banco de Dados**, Rio Grande do Sul: Sagra Luzzato, 1998.

LAUREANO, M. **Máquinas Virtuais e Emuladores - Conceitos, Técnicas e Aplicações**, Novatec, 2006. Disponível em: < http://www.mlaureano.org/aulas_material/so/livro_vm_laureano.pdf> Acesso em: 25 nov, 2015.

MATTHEW, N. E STONES, R. **Beginning databases with PostgreSQL**, New York: Springer, 2005.

OPPEL, A. J. A **Beginner´s Guide Database**, New York: McGraw-Hill, 2009.

POSTGRES SQL – **Oficial Documentation**, Berkley. Disponível em <http://www.postgresql.org/docs/9.4/static/index.html>. Acesso em: 25 nov. 2015;

SHARMA, N., PERNIU, L., CHONG, R. F., NANDAN, C., MITEA, A., NONVINKERE, M. E DANUBIANU, M. **Database Fundamentals**, Markham: IBM Corporation, 2010.

SMITH, G. **PostgreSQL 9.0 High Performance**, Mumbai: Birmingham, 2010.