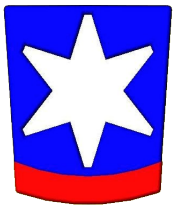


UNIVERSIDADE DE CAXIAS DO SUL
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO
TRABALHO DE CONCLUSÃO DE CURSO

MIGRAÇÃO E TESTES DE UM ERP DO SGDB

SQL SERVER PARA O SGDB POSTGRESQL

Wilian Ivo Selzlein

CAXIAS DO SUL
2009

WILIAN IVO SELZLEIN

MIGRAÇÃO E TESTES DE UM ERP DO SGDB SQL SERVER PARA O SGDB POSTGRESQL

Trabalho de Conclusão de Curso

Documento de especificação para avaliação na disciplina TCC apresentado à Universidade de Caxias do Sul, no curso de Sistemas de Informação.

Orientador: Prof. Msc. Daniel Luis Notari

CAXIAS DO SUL
2009

Balanço da vida

Eis-me aqui, chegado o fim da noite, a examinar minha vida em todos os detalhes. O que fiz de bem, o que fiz de mal, o quanto tinha de ter feito mais de bem, o quanto tinha de ter evitado de causar o mal.

A verdadeira arte da vida é viver. E não há vida sem sofrer. Não importa que soframos, enquanto estivermos sofrendo temos a garantia de que estamos vivendo.
Tem sido dura a caminhada, mas ela não teria sentido se não fosse penosa, áspera, ingrata, quase intransponível.
Eis-me aqui, passado mais um dia, traçando o balanço da minha vida.
É quase um milagre, ainda estou vivo, isto é o que importa.

Estou purificado pela vida, é possível que não tenha atingido meu ideal, nem sei se o tive.
O que interessa é que resisti. E que ainda resta lugar para a esperança.
O que importa é que muitas vezes comovi os outros e ainda tenho forças para continuar me comovendo.

As lágrimas formam os melhores momentos da minha vida. Senti-me humano quando as derramei por dor. Senti-me aproximado de Deus quando as verti por alegria.
Sem as lágrimas, minha vida não teria sentido. Sem as lágrimas de sofrimento, desespero e desamparo que a vida me submeteu, sem as lágrimas de emoção que consegui provocar nos outros, minha vida não teria sentido.

Nem graça teria a vida se não houvesse a ingratidão. Quando topamos com a ingratidão, isso nos deve servir de incentivo a que façamos outra obra de bem.
Não teria graça a vida se não trombássemos com a inveja. A inveja é a mais clara demonstração de que fomos bem-sucedidos.
Infelizes os homens que não conviveram com a inveja, eles podem considerar que suas vidas foram inúteis.

Nem atração nenhuma teria a vida se tivéssemos, como tolamente pretendemos, longos períodos de felicidade. A vida se tornaria mais ainda tediosa do que já o é, se fôssemos ditosos.
Neste fim de noite de intensas reflexões, quase atinjo a verdade da vida: o que me move, o que me sustenta é essa incerteza de como será o dia de amanhã. Isso é que me anima, é possível que amanhã ou depois de amanhã um amigo, um parente, todas as pessoas com quem convivo reconheçam finalmente que lhes quis bem e não fiz outra coisa senão querer-lhes bem, embora não tivesse sido compreendido.

Se eu tivesse que pedir uma última coisa à vida, solicitaria que ela me concedesse antes de morrer que os meus circunstantes reconhecessem que fui um tipo inesquecível.
Mas tem de ser antes de morrer. Para que eu possa morrer em paz.

Eu preciso que as pessoas próximas de mim se assegurem de que eu faço falta a elas antes de morrer.
Depois de morrer, pouco se me dá que eu faça ou não falta às pessoas próximas de mim.
Mesmo sem esse propósito, construí minha vida na esperança de que eu fizesse falta às pessoas.
Só se os outros sentissem a minha falta, a necessidade de mim, a imprescindibilidade de mim, minha vida teria adquirido sentido.

Eu necessito vitalmente de que as pessoas sintam saudade de mim antes de eu morrer. Eu tenho de ter a certeza de que as pessoas tenham lembrança boa de mim enquanto estou vivo.
Minha vida terá sido um lastimoso fracasso se ninguém sentir falta, saudade ou lembrança de mim. Mas enquanto eu estiver vivo. Depois que eu estiver morto, mesmo os meus sofrimentos e dedicações mais ridículos se tornarão respeitáveis.
Eu almejo respeitabilidade em vida. Só assim considerarei que minha vida valeu a pena.

Paulo Sant'Ana

AGRADECIMENTOS

Agradeço ao meu orientador Prof. Ms. Daniel Luis Notari, pela orientação segura e a confiança que me passou ao longo da elaboração deste trabalho.

Aos amigos e companheiros, que passaram por mim esse tempo e deixaram lembranças pela amizade e, que de maneira direta ou indireta, contribuíram com a realização deste trabalho; obrigado a todos pela força.

Aos professores da Universidade de Caxias do Sul por todo o aprendizado ao longo dos anos e pela amizade. Aos membros da banca, agradeço as contribuições.

Pela confiança que me passou, a eSafety Sistemas e Tecnologia, por acreditar no meu potencial e me deixar utilizar de seu todo para realizar mais esse desafio.

À família Guindani, minha segunda família. Especialmente para meu primo Roberto, pelo imenso apoio.

Aos meus pais Rudi e Dena, meu irmão Ulysses e irmã Clara. Meu muito obrigado pela força e ajuda que vocês sempre me deram.

E por fim, a minha namorada Cris, pelo amor, carinho, amizade, confiança, paciência e dedicação que tem me dado.

RESUMO

O objetivo do trabalho está baseado principalmente em realizar diversos testes para concluir, não qual o melhor SGDB, mas sim qual dos dois SGDBs se encaixa melhor para o cliente e também para o sistema da empresa eSafety Sistemas e Tecnologia Ltda. Essa tarefa começa com a conversão do SGDB e a migração dos dados através de ferramentas disponíveis no mercado. Os testes realizados através de ferramentas permitirão disponibilizar a quem seja útil, conclusões sobre os melhores desempenhos, dos mais variados aspectos, possibilitando uma forma de avaliar o relacionamento que os SGDBs têm com o sistema, trabalhando de maneira conjunta e, acima de tudo, obtendo uma melhor compreensão de suas funcionalidades.

Palavras-chave: SGDB, Banco de Dados, Sistemas, Testes.

ABSTRACT

The objective of this work is based mainly on conducting various tests to complete, not the best SGDB, but which of the two SGDBs fits best for the client and also the system of the company eSafety Systems and Technology Ltda. This task begin with the conversion of SGDB and migration of data through tools available on the market. Tests performed by the tools available which will be useful, conclusions on the best performance, a variety of ways, allowing a way to assess the relationships that have SGDBs with the system, working on a joint and, above all, getting a better understanding of its features.

Key-words: SGDB, Data Base, System, Tests.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

SGDB	*Database Management System*	Sistema Gerenciador de Bancos de Dados
SQL	*Structured Query Language*	Linguagem de Consulta Estruturada
ODBC	*Open Data Base Connectivity*	Padrão para Acesso a Sistemas Gerenciadores de Bancos de Dados
CRM	*Customer Relationship Management*	Gestão de Relacionamento com o Cliente
ERP	*Enterprise Resource Planning*	Planejamento dos Recursos da Empresa
ANSI	*American Nacional Standards Institute*	Instituto Americano de Padrões
BI	*Business Intelligence*	Inteligência de Negócios
ISO	*International Standards Organization*	Organização Internacional de Padrões.
DBA	*Database Administrator*	Administrador do Banco de Dados
DRE	*Statement of Profit and Exercise.*	Demonstrativo de Resultado e Exercício.
EDI	*Electronic Data Interchange*	Troca Eletrônica de Arquivos
CPU	*Central Processing Unit*	Unidade Central de Processamento
DLL	*Dynamic-link library*	Biblioteca de ligação dinâmica
RAD	*Rapid Application Development*	Aplicação de Desenvolvimento Rápido
TDS	*Tabular Data Stream*	Fluxo de Dados Tabulares

SUMÁRIO

1 INTRODUÇÃO

Os Sistemas Gerenciadores de Banco de Dados (SGBD) desempenham um papel fundamental na era tecnológica. Sabe-se que a informação tornou-se um bem muito valioso, fundamental para decisões estratégicas e está presente em todo lugar. Devido a isso, ela tem que estar disponível sempre que necessário (DATE, 2003).

O papel dos SGBDs é permitir a administração de bancos de dados de modo a garantir que a informação esteja disponível em tempo hábil e em condições de ser lida e entendida, auxiliando dessa forma a administração da empresa (DATE, 2003).

Com o avanço dos SGDBs e a necessidade de se disponibilizar soluções cada vez mais complexas e com custos variados, foi abordada a importância do cliente optar pela escolha do SGDB de sua preferência ou que se encaixe melhor a sua estrutura física e financeira. A partir de seus fundamentos teóricos, a empresa disponibiliza a escolha de dois SGDB Relacionais: Microsoft SQL Server e PostgreSQL.

A escolha desses dois SGDBs, parte do princípio de que, como os clientes da eSafety são desde grandes empresas até empresas muito pequenas, é necessário que se sugira duas opções: A primeira para quem pode pagar a licença do Microsoft SQL Server, tendo suporte do Microsoft e também que prefiram trabalhar com o Sistema Operacional Microsoft Windows. Já os clientes menores, com baixo faturamento, podem optar por uma solução de servidor Linux e também com um SGDB robusto e de licença livre como o PostgreSQL.

Para disponibilizar totais informações para seus clientes, falta para a empresa disponibilizar informações reais sobre esses SGDBs. Para isso, é necessário testar os dois SGDBs, para que se auxilie a projetar o sistema de maneira correta. Entre os mais variados testes, pode-se citar alguns como testes de espaço de armazenamento, de qualidade, de confiabilidade, de segurança, de desempenho e disponibilidade das informações contidas.

Como a massa de dados utilizados pelo sistema atual é grande, gerando uma enorme quantidade de registros a cada dia, esse trabalho servirá para uma, melhor questão de segurança, visto que por se tratar de empresas sérias e com dados de natureza crítica e sigilosa, a informação contida no seu SGDB é algo infinitamente importante.

1.1 Objetivo

O objetivo do trabalho é selecionar, testar e validar qual é o SGDB mais apropriado para o cliente da empresa eSafety para o uso do sistema ERP.

Os testes visam obter conclusões sobre os melhores desempenhos, melhor segurança, validação dos dados, etc. Ou seja, os testes a serem realizados são: teste de validação, teste de sistema, teste de segurança, testes de estresse e teste de desempenho. Para demonstrar os resultados, será realizada a procura de ferramentas que ajudem no processo de validação e também na divulgação, gerando gráficos e tabelas (PRESSMAN, 2007).

Um dos objetivos desse trabalho será, a partir de um estudo de caso e validações, propor em um projeto quais as melhores especificações para um melhor desempenho do sistema com os dois SGDBs. Dessa forma, facilitando e gerando um projeto de implantação com conceitos e práticas a partir de estudos de casos, com uma completa análise de requisitos neste ambiente (necessidades, funcionalidades, características, etc.).

Atualmente o sistema está disponível apenas no SGDB SQL Server, sendo também parte desse trabalho a migração desse SGDB para o PostgreSQL e adaptações nos códigos fonte atuais do sistema.

Tendo em vista que o objetivo deste trabalho é pesquisar, desenvolver e gerar resultados, será feito um projeto de pesquisa com implementação. Realizar testes é um dos objetivos - um conjunto de testes que tenham uma alta probabilidade de revelar defeitos no software e/ou no SGDB.

1.2 Estrutura

O capítulo dois fará um breve apanhado sobre a situação atual da empresa, falando da sua estrutura atual, seu banco de dados e seu sistema. Será feita uma análise sobre como ficaria o seu sistema ERP com o SQL Server e o PostgreSQL.

O terceiro capítulo refere-se a atividade de teste em *softwares*, quais os objetivos que esse apanhado realiza, as metas e os resultados que pode trazer. Será abordado uma estratégia de teste de *software*, explicando os testes que serão realizados nesse trabalho. Para realizar os testes, ferramentas serão utilizadas. E para concluir, uma explicação sobre as medições e métricas de software.

O quarto capítulo, apresentará o desenvolvimento da solução do problema levantado, a conversão do SGDB, a migração dos dados e as alterações no *software*.

Para cada ferramenta utilizada, será abordado as características de cada uma delas, suas telas e descrições.

O quinto capítulo demonstrará os resultados obtidos com o primeiro estudo de caso: A conversão do esquema, a migração dos dados e as alterações no sistema SistranNet.

No sexto capítulo, é apresentado o segundo estudo de caso que corresponde aos testes executados tanto no sistema como nos SGDBs.

O último capítulo é descrito as conclusões do trabalho, resultados obtidos, discutidas as principais contribuições e sugestões para trabalhos futuros. Após a conclusão, cinco anexos completam o trabalho.

2 LEVANTAMENTO DA SITUAÇÃO ATUAL

Este capítulo tem como objetivo levantar requisitos necessários para a execução das atividades propostas e uma melhor execução do cronograma das atividades que serão desenvolvidas durante todo o trabalho de conclusão de curso. Primeiramente o capítulo irá descrever toda a estrutura atual da empresa, falando sobre o seu ERP e sua arquitetura. A seguir, será realizado levantamentos de como o sistema deve funcionar com ambos os SGDBs, e por fim, uma tabela explicando detalhadamente cada componente dos SGDBs, necessários para uma melhor tomada de decisão.

2.1 Estrutura atual da eSafety

A eSafety é uma empresa que desde 1993 apresenta soluções inovadoras em Gestão de Transportes, através do desenvolvimento de sistemas e pelo apoio ao gestor através de sua consultoria (ESAFETY, 2009).

O sistema SistranNet é um sistema para empresas de transportes rodoviário de cargas e foi iniciado em 1998 com a versão do Borland Delphi 3. Após a evolução da ferramenta da Borland, o sistema foi migrado para a versão do Borland Delphi 7 e mantêm-se atualmente.

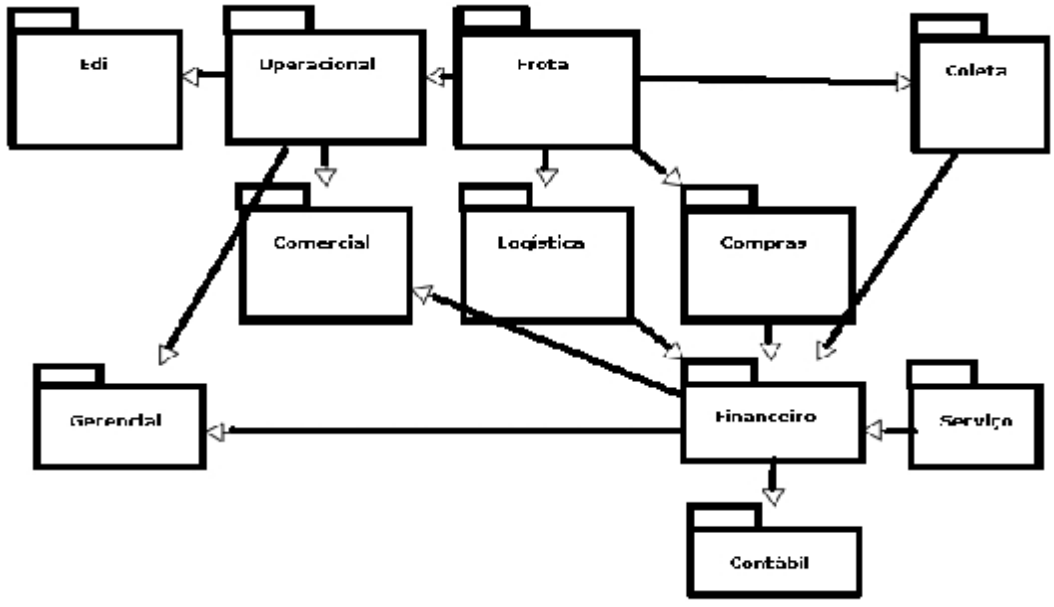


Figura 1 : Módulos do Sistema SistranNet.

Abaixo a apresentação de cada módulo:

- Módulo Comercial**:** referente a integração com o cliente, possui cotação de preço, agenda de visitas, vendedores, tabelas de frete, atendimento e reclamações e informações ao cliente, como por exemplo acesso pelo *Web.*
- Módulo Compras: Esse módulo gerencia toda a parte de compras da empresa, realizando o processo nessa ordem: Requisição, Cotações, .Orçamentos, Ordem de Compra e Recebimento. Como o sistema é voltado para o ramo de transportes, há nesse módulo também um item de Ordem de Abastecimento.
- Módulo Frota: Importante para o ramo de transportes é o custo dos seus veículos, esse módulo gerencia essa parte, desde o cadastro do veículo, como seu controle de quilometragem, ordem de serviços, sua produção e custos.
- Módulo Operacional: Aqui encontra-se a parte braçal do sistema, .disponibilidades do veículo, impressão de conhecimentos, relação de entregas e viagens.
- Módulo Coletas: O módulo coletas é como se fosse uma extensão do operacional, só que ao invés de realizar grandes viagens, o veículo faz coletas e entregas dentro da cidade, esse módulo contém a Emissão das Coletas, Pedido de Coletas e Coletas Automáticas.
- Módulo Financeiro: Após realizar o processo dos transportes, sendo ele uma coleta e entrega ou uma viagem é necessário cobrar pelo serviço. O financeiro envolve o faturamento do serviço, contas a pagar, controle de cheques e boletos bancários.
- Módulo Contábil**:** O módulo contábil é um auxílio para o contador da empresa, contendo a parte básica de um controle contábil, como lançamentos contábeis, partidas múltiplas, o plano de contas da empresa, integrações com órgãos e outros sistemas, como por exemplo seguradoras, e por fim relatórios como Razão, Balancete, DRE, etc.
- Módulo Logística: A parte de armazenagem e impressão de notas é realizado pelo setor de logística, sendo assim o módulo contém controle sobre a Área de Armazenagem, Romaneios de Separação, *Cross Docking*, Montagem de Kits e Notas Fiscais de Entrada/Saídas. Por fim, o cadastro, controle, movimentação e estoque dos produtos transportados.

- Módulo EDI: A sigla EDI vem de *Electronic Data Interchange*, que significa movimento eletrônico de documentos. É um padrão de negócio entre, ou dentro, de empresas, para transmitir arquivos automaticamente.
- Módulo Serviço: Esse módulo inclui apenas a parte de prestação de serviço, depois de lançar e imprimir a nota de serviço, o faturamento da mesma é realizado.
- Módulo Gerencial: Apenas os usuários gerencias tem acesso a esse módulo, que contém relatórios analíticos e sintéticos de nível gerencial. Mostrando o desempenho financeiro e operacional da empresa. É um módulo que ajuda muito na tomada de decisão dos administradores.

Hoje o sistema encontra-se com o SGDB SQL Server 2000 da Microsoft, funciona através de DataSnap a conexão com o SGDB, que "é usada para desenvolver aplicativos personalizados de cliente e servidor em um ambiente três camadas" (CANTÙ, 2003).

Nessa arquitetura, a apresentação, o processamento de aplicações e o gerenciamento de dados são processos logicamente separados (SOMMERVILLE, 2003).

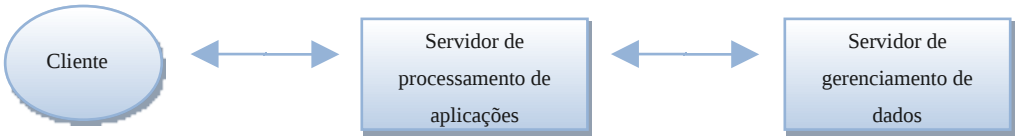


Figura 2 : Uma arquitetura cliente-servidor de três camadas.
(SOMMERVILLE, 2003)

Uma arquitetura de software cliente-servidor de três camadas não significa necessariamente que haja três sistemas de computador conectados à rede. Um único servidor pode executar tanto o processamento de aplicações quanto o gerenciamento de dados de aplicações, como servidores lógicos separados. Contudo, se a necessidade aumentar, será relativamente simples separar o processamento de aplicações e o gerenciamento de dados e executá-los em processadores separados (SOMMERVILLE, 2003).

A comunicação entre bases pode ser feita através de linha discada (dependendo da quantidade de transações), ADSL ou *link* dedicado para melhor segurança. Importante salientar que não há necessidade de nenhuma ferramenta adicional de

terceiros (*CITRIX* ou *TERMINAL SERVICE*) para comunicação entre as bases.

2.2 ERP com SQL SERVER

O SQL Server é uma plataforma abrangente de SGDB que fornece recursos de gerenciamento de dados de classe empresarial com ferramentas de BI (*Business Intelligence*) integradas. O mecanismo de banco de dados do SQL Server oferece um armazenamento tanto para dados relacionais quanto estruturados, permitindo criar e gerenciar aplicativos de dados altamente disponíveis e eficientes para uso em seus negócios (MICROSOFT,2009).

Conforme Sommerville (2002), o SQL Server "é um conjunto de componentes e produtos que se complementam, como um sistema cliente/servidor, para atender os requisitos de armazenamento, recuperação e análise de dados de qualquer entidade ou empresa. Sente-se igualmente confortável em ambientes variando entre as maiores empresas e sites *Web* comerciais enviando milhões de transações por dia para pequenas empresas e até mesmo aplicativos individuais e especializados que necessitam de um armazenamento de dados resistente e um serviço persistente.

Quando a Microsoft distribuiu o SQL Server 7.0, ficou claro que o SQL Server 6.5 e versões anteriores se tornariam produtos de herança, aplicativos essencialmente da era pré-Internet. Um trio de princípios e tornou o tema central do SQL Server 7.0: o produto tinha de ser compatível com a Internet, ser altamente escalonável (para competir com os sistemas de alcance médio e de mainframe) e disponibilizar acesso mais rápido ao mercado. Em sua saga, para ser a melhor, a equipe de desenvolvimento da Microsoft para o SQL Server ficou obcecada com esses temas. Os bancos de dados SQL Server são totalmente compatíveis a definição de banco de dados do SQL-92 da ANSI (SHAPIRO, 2002).

2.3 ERP com PostgreSQL

O PostgreSQL é um SGBDR que está baseado nos padrões SQL ANSI-92, 96 e 99, de alta performance, de fácil administração e utilização em projetos e também é multiplataforma (PEREIRA NETO, 2003).

A abertura completa de seu código-fonte se deu em meados de 1996. E neste momento, esse software de SGDBR já era conhecido como PostgreSQL. Tratava-se do código-fonte original do *Postgres* construído pela Universidade de Berkeley, acrescido

do código-fonte do interpretador SQL, construído por Andrew Yu e Jolly Chen. Desde então, várias contribuições têm sido agregadas ao produto a fim de posicioná-lo de forma extremamente competitiva em nível de funcionalidade com os mais modernos SGBDRs do mercado, com uma vantagem extremamente importante: é totalmente livre, de código-aberto (PEREIRA NETO, 2003).

O PostgreSQL foi projetado para ser uma ferramenta de SGDBR simples de administrar, robusta e capaz de atender a uma grande gama de transações simultâneas, além de ser multiplataforma. Associada a um *engine* capaz de oferecer alta disponibilidade de dados para processamento e suporte à decisão de transações simultaneamente (PEREIRA NETO, 2003).A figura abaixo demonstra a arquitetura do PostgreSQL:

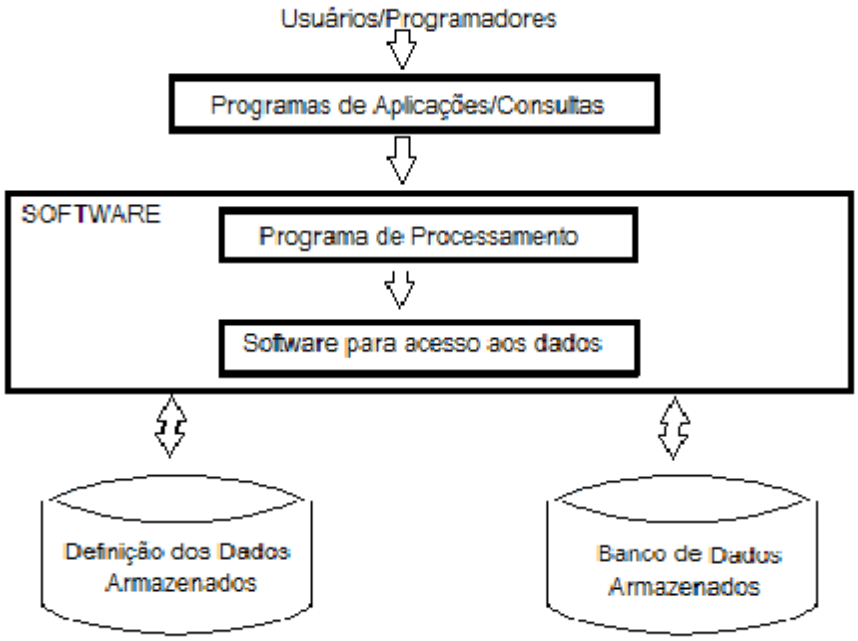


Figura 3: Configuração de um SGDB simplificado.

(ELMASRI & NAVATHE, 2005).

O PostgreSQL é conhecido por ser uma solução OpenSource, o qual é bem flexível a questões de de limitações como storage, processador e plataforma. Pode ser instalado em qualquer tipo de hardware com diversos tipos de sistemas operacionais de forma estável. Como desvantagens, as principais são em relação a ferramentas de administração e sua própria administração, onde algumas vezes é necessário possuir um conhecimento elevado de sistema operacional para realizar algumas atividades.

2.4 Tabela Comparativa entre SGDB

Após a análise dos dois SGDBs, é importante demonstrar resumidamente quais as principais características de ambos os SGDBs escolhidos para o roteiro desse trabalho: SQL Server e PostgreSQL. Abaixo segue uma tabela com as funcionalidades de cada banco e se são aplicadas aos bancos de dados grátis existentes no mercado: DB2, Firebird , MySQdL, PostgreSQL, Oracle e SQL Server.

Tabela 1 : Comparativo entre os atuais SGDBs grátis do mercado.
Legenda:
S/R:Sem restrições.
2-N/1:Não Informado.
3-Disponível apenas através de ferramentas de terceiros.
4-Atuando apenas como target DataBase,nunca como source DataBase
5-Apenas para desenvolvimento de relatórios do Reporting Services.
6-O Reporting Services da Microsoft pode ser utilizado gratuitamente.
Fonte: (SQL MAGAZINE 37, 2006).

Item	Sistema de Gerenciamento de Banco de Dados					
	DB2	Firebird	MySQL	Postgre SQL	Oracle	SQL Server
Limitações						
Número Máximo de Processadores	2	S/R1	S/R1	S/R1	1	1
Tamanho máximo de memória RAM	4 GB	S/R1	S/R1	S/R1	1 GB	1 GB
Plataforma	S/R1	S/R1	S/R1	S/R1	S/R1	Windows
Tamanho máximo do Banco de Dados	S/R1	7 TB	S/R1	S/R1	4 GB	4 GB
Número máximo de Conexões	S/R1	1024	S/R2	S/R1	N/12	32.767
Recursos de Desenvolvimento						
Trigger	Sim	Sim	Sim	Sim	Sim	Sim
Stored Procedure	Sim	Sim	Sim	Sim	Sim	Sim
Function	Sim	Sim	Sim	Sim	Sim	Sim
XML	Sim	Não	Sim	Sim	Sim	Sim
Java (escrita de procedures)	Sim	N/12	N/12	Sim	Não	N/12
Orientação a Objetos	Sim	Não	Não	Sim	Sim	N/12
Compatibilidade ao padrão ANSI						
Administração						
Gerenciamento de Usuários	Sim	Sim	Sim	Sim	Sim	Sim
Gerenciamento de Armazenamento	Sim	Sim	Sim	Sim	Sim	Sim
Gerenciamento de Recursos	Sim	Não	Sim	Não	Sim	Sim
Tuning automático	Sim	Não	Não	Não	Sim	Sim
Agendamento de Jobs	Sim	Sim[3]	Sim	Sim[3]	Sim	Sim[3]
Backup						
Off-Line (Cold Backup)	Sim	Sim	Sim	Sim	Sim	Sim
On-Line (Hot Backup)	Sim	Sim	Sim	Sim	Sim	Sim
Incremental	Sim	Sim	Sim	Não	Sim	Sim
Lógico (dump)	Sim	Não	Sim	Sim	Sim	N/12

Item	Sistema de Gerenciamento de Banco de Dados					
	DB2	Firebird	MySQL	Postgre SQL	Oracle	SQL Server
Restore						
Recuperação Total	Sim	Sim	Sim	Sim	Sim	Sim
Recuperação Parcial	Sim	Sim	Sim	Sim	Sim	Sim
Recuperação em ponto no tempo	Sim	Não	Sim	Sim	Sim	Sim
Replicação	Sim	Sim[3]	Sim	Sim[3]	Sim	Sim[4]
Segurança						
Conexão	Sim	Sim	Sim	Sim	Sim	Sim
Privilégios	Sim	Sim	Sim	Sim	Sim	Sim
Auditoria	Sim	Não	Sim	Sim	Sim	Sim
Criptografia	Sim	Não	Sim	Sim	Sim	Sim
Ferramentas						
Ferramentas de Administração	Sim	Sim	Sim	Sim	Sim	Sim
Ferramentas de Desenvolvimento	Sim	Sim[3]	Sim	Sim	Sim	Sim[5]
Ferramentas de terceiros	Sim	Sim	Sim	Sim	Sim	Sim
Outros						
Suporte a compactação de linha	Sim	Não	Não	N/12	Sim	N/12
Gerenciamento dinâmico de memória	Sim	Sim	Sim	N/12	Sim	Sim
Particionamento de Tabelas	Sim	Não	Sim	N/12	Sim	Não
Tabelas Temporárias	Sim	Não	Sim	Sim	Sim	Sim
Views Materializadas	Sim	Não	N/12	Sim	Sim	Sim
Bloqueio otimista em nível de linha	Sim	Sim	Sim	N/12	Sim	Sim
Índice Full Text (Full Text Search) (Nota 1)	Sim	Sim[3]	Sim	N/12	Sim	Sim
Suporte a Siste. De Inf. Geográficas (GIS)	Sim	Não	Sim	Sim	Não	N/12
Flashback	Não	Não	Não	Não	Sim	Não
Serviço de Relatórios	Sim[3]	Sim[3]	Sim[3]	Sim[3]	Sim[3]	Sim[6]
Índices Cluester	N/12	Não	Sim	Sim	Sim	Sim
Embedded database (BD Embutido)	Não	Sim	Não	Não	Não	Não
Suporte a Bancos de Dados Distribuidos	Não	N/12	Sim	Sim[3]	Sim	Sim
Trace Log	Sim	N/12	Sim	Não	Sim	Sim

Conforme a equipe da SQL MAGAZINE Ano 3 e Edição 37. Cada item foi elaborado por um autor diferente, uma consulta foi feita a cada um deles para confirmar a aderência ou não a cada uma das características listadas na tabela. Essa comparação dos SGDBs foi dividida em tópicos que foram explorados:

- Limitações: Quais as limitações impostas pelo SGDB, o qual deverá suportar o ambiente que ele deverá tomar conta, levando em consideração a memória, capacidade de armazenamento, utilização de recursos de CPU e quais as plataformas que poderão acolher o SGDB.

- Recursos de Desenvolvimento: Conhecer quais os recursos cada SGDB oferece. *Triggers*, *Stored Procedures e Functions* são recursos já consagrados no desenvolvimento de sistemas.

- Compatibilidade do padrão ANSI: A portabilidade dos sistemas tem sido primordial no desenvolvimento, e a compatibilidade com o padrão ANSI é fundamental.

- Administração: Extrema importância o papel do DBA.

- *Backup*: Apesar de todos os problemas que podem acontecer, é necessário uma boa política de *backups*.

- *Restore*: Tão importante quanto a política de *backup*, a de *restore* deve funcionar, para dar valor ao *backup*.

- Replicação: Levando em consideração banco de dados distribuídos e atualizações on-line matriz/filial.

- Segurança: Verificar os problemas que pode-se enfrentar em relação à proteção de dados, importantes a organização.

- Ferramentas: Ligados a produtividade, eficiência e eficácia no desenvolvimento/ administração do sistema.

- Desvantagens: Pontos fracos e desvantagens na utilização do SGDB.

- Outros: características relevantes de cada SGDB.

- Serviços agregados: Serviços que estão fora do padrão do SGDB.

- Por que utilizar? Ponto de vista sobre motivos para usar a solução do SGDB.

2.5 Considerações Finais

O PostgreSQL, apesar de rodar em máquinas antigas, é ideal para empresas de médio e grande porte, soluções Enterprise por exemplo. Uma grande vantagem é a documentação em português gratuita na internet, proporciona desempenho e é possível obter uma grande economia de software já que pode-se usa-lo com um sistema operacional e ferramenta de desenvolvimento utilizando software livre (SQL Magazine, 37).

Para as próximas etapas, será possível ver que a conclusão do trabalho e o seu principal resultado que é o sistema ERP funcionando juntamente com o PostgreSQL é possível, visto suas características. No próxima capítulo serão vistos os testes de *softwares*, os quais irão garantir a qualidade do projeto.

3 TESTES EM SOFTWARES

A atividade de teste de *software* é um elemento crítico da garantia de qualidade de *software* e representa a última e a primeira revisão de especificação, projeto e codificação. O destaque crescente do *software* como elemento de sistema e os “custos” envolvidos associados às falhas de *software* são forças propulsoras para uma atividade de teste cuidadosa e bem planejada. Não é incomum que uma organização de software gaste 40% do esforço de projeto total em teste. No extremo, a atividade de teste de software de sistemas, dos quais dependam vidas humanas, pode custar de três a cinco vezes mais que todos os outros passos da engenharia de softwares juntos (PRESSMAN, 2007).

O teste de software é responsável pela maior porcentagem de esforço técnico no processo de desenvolvimento do software. Contudo, é de suma importância a sua realização, visto que o tempo utilizado para testes, é ganho no avanço de outros processos. Conforme Sommerville (2002) os testes de softwares envolvem executar uma implementação do software com os dados de teste e examinar as saídas dele e seu comportamento operacional, a fim de verificar se ele está sendo executado conforme o esperado. Os testes são uma técnica dinâmica de verificação e validação porque trabalham com uma representação executável do sistema.

A meta definitiva do processo de verificação e validação é estabelecer a confiança de que o sistema de software é adequado a seu propósito. Isso não significa que o programa tenha de ser inteiramente livre de defeitos. Em vez disso, significa que o sistema deve ser suficientemente bom para o uso pretendido (SOMMERVILLE 2003).

Conforme Pressman (2007), as seguintes regras podem servir bem como objetivos de teste:

1. A atividade de teste é o processo de executar um programa com a intenção de descobrir um erro;

2. Um bom caso de teste é aquele que tem uma elevada probabilidade de revelar um erro ainda não descoberto;

3. Um teste bem-sucedido é aquele que reverá um erro ainda não descoberto.

Os casos de testes são especificações das entradas para o teste e da saída esperada do sistema. Os dados de testes podem, algumas vezes, ser gerados automaticamente. Conclui Pressman (2007) que “o objetivo principal do projeto de casos de teste é derivar um conjunto de testes que tenha uma alta probabilidade de revelar defeitos no *software*”.

A seguir, serão abordado os testes de *softwares* e estratégias dos mesmos, explicando cada um dos testes detalhadamente.

3.1 Estratégias de teste de *softwares*

Uma estratégia de teste de *software* integra técnicas de projeto de casos de teste em uma série bem-definida de passos que resultam na construção bem-sucedida de *software*. Outra coisa importante é que uma estratégia de teste de *software* proporciona uma base para o desenvolvedor de *software*, para a organização de garantia de qualidade e para o cliente – descreve os passos a serem dados como parte da atividade de teste, quando esses passos são planejados e depois executados, e quanto esforço, tempo e recursos são exigidos. Por conseguinte, qualquer estratégia de teste deve incorporar planejamento de teste, projeto de casos de teste, execução de teste e a resultante coleta e avaliação de dados (PRESSMAN, 2007).

Existem duas preocupações no teste de *software* conforme Gustafson (2003): (1) quais casos de teste usar e (2) quantos casos de teste são necessários. A seleção de casos de teste pode ser baseada nas especificações, na estrutura do código, no fluxo de dados ou na seleção randômica de casos de teste.

Um exemplo de estratégia de *software*, pode ser visto na figura 4, onde mostra um fluxo desde o produto liberado, até o final do ciclo - fase de requisitos. Tornando assim quase que um trabalho sem fim.

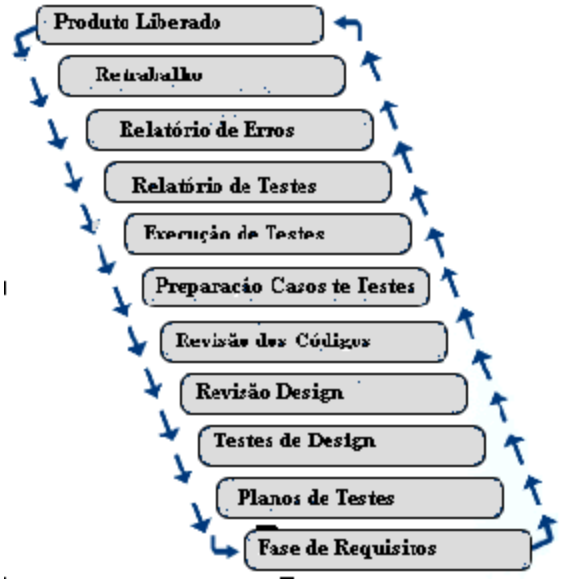


Figura 4: Ciclo de Teste de Software
Fonte: http://www.akibsoftech.com/Testing_Services.html

Conforme Pressman (2007) uma estratégia de teste de software deve ser flexível o bastante para promover a criatividade e a customização necessárias para testar adequadamente todos os grandes sistemas baseados em *software*. Ao mesmo tempo, a estratégia deve ser rígida o bastante para promover um razoável planejamento e rastreamento administrativo à medida que o projeto progride.

Um teste, conforme SILBERSCHATZ; KORTH & SUDARSHAN (1999) tem de chegar a um estado aceitável até mesmo na presença de falhas de sistema, caso contrário, deve-se abortar a tarefa.

3.2 Teses

Nos próximos sub tópicos serão apresentados os testes que serão utilizados nesse trabalho. Será descrito cada teste tentando exemplificar um por um, demonstrando alguma ferramenta que auxilie nessa tarefa.

3.2.1 Teste de Validação

O teste de validação é bem-sucedido quando o *software* funciona de uma maneira razoavelmente esperada pelo cliente (PRESSMAN, 2007).

Para termos a certeza que a validação está correta, afirma Pressman (2007) que "[...] tanto o plano como o procedimento são projetados para ter a garantia de que todos os requisitos funcionais são satisfeitos, todos os requisitos de desempenho são conseguidos, a documentação está correta e passou por um trabalho de engenharia humana e outros requisitos são cumpridos (por exemplo, portabilidade, compatibilidade, remoção de erros e manutenibilidade)."

3.2.2 Teste de Sistema

Um sistema não é apenas o *software* em si, e sim toda a estrutura, componentes e elementos que o interagem. Segundo Pressman (2007), o teste de sistema é "uma série de diferentes testes, cujo propósito primordial é pôr completamente à prova o sistema baseado em computador. Não obstante cada teste tenha uma finalidade diferente, todo o trabalho deve verificar se todos os elementos do sistema foram adequadamente integrados e realizam as funções atribuídas".

Conclui Pressman (2007) que "os passos dados durante a atividade de projeto e teste do *software* melhoram grandemente a probabilidade de uma integração bem-sucedida com o sistema mais amplo."

3.2.3 Teste de Segurança

Segundo Pressman (2007) "o teste de segurança tenta verificar se todos os mecanismos de proteção embutidos num sistema o protegerão, de fato, de acessos indevidos. Desde que tenha tempo e recursos suficientes, um bom teste de segurança penetrará por fim num sistema. O papel do projetista do sistema é fazer com que o acesso custe mais do que o valor da informação que será obtida."

SILBERSCHATZ; KORTH & SUDARSHAN (1999) citam falhas de segurança podem causar:

- Quedas durante o processamento de transações;
- Anomalias causadas por acesso concorrentemente ao banco de dados;
- Anomalias causadas pela distribuição de dados pelos diversos computadores;
- Erros lógicos que violam as regras impostas para que as transações

preservem as restrições de consistência do banco de dados;

- Leitura não autorizada de dados (roubo de informação);
- Modificação não autorizada de dados;
- Destruição não autorizada de dados.

3.2.4 Teste de Estresse

Segundo Pressman (2007) “o teste de *software* é realizado para confrontar os programas com situações anormais. O teste de *software* executa o sistema de uma forma que exige recursos em quantidade, frequência ou volume anormais.”

A função do teste de estresse, conforme Sommerville (2003), é testar “o comportamento de falha do sistema. Podem surgir determinadas circunstâncias por meio de uma combinação inesperada de eventos, quando a carga colocada no sistema excede a carga máxima prevista. Nessas circunstâncias, é importante que a falha do sistema não cause a corrupção de dados ou a perda inesperada de serviços do usuário. Os testes de estresse verificam se, sobrecarregando o sistema, ele ‘apresenta uma falha mais leve’, em vez de entrar em colapso sob essa carga.

Para Koscianski (2006) um teste de estresse pode ser feito para responder algumas perguntas como:

- O sistema consegue atingir o objetivo?
- Qual o número máximo de transações realmente possível?
- Se a plataforma de execução se degradar, como o sistema se comportará?

3.2.5 Teste de Desempenho

Segundo Pressman (2007) o teste de desempenho é “idealizado para testar o desempenho em tempo de execução do *software* dentro do contexto de um sistema integrado. O teste de desempenho ocorre ao longo de todos os passos do processo de teste. [...] Apenas quando todos os elementos de sistema estão plenamente integrados é que o desempenho real de um sistema pode ser verificado.” (PRESSMAN, 2007)

Ajustar o desempenho de um sistema envolve ajustar diversos parâmetros e projetar escolhas para melhorar seu desempenho para um aplicação específica. Vários aspectos de um projeto de banco de dados afetam o desempenho de uma aplicação. Cada um desses aspectos pode ser ajustado de forma que o desempenho seja melhorado. Alguns itens que podem ser ajustados são: Eliminação de Gargalos, Ajuste de Esquemas, Ajuste de Índices e Ajuste de Transações (SILBERSCHATZ; KORTH & SUDARSHAN, 1999).

Os testes de desempenho precisam ser projetados para assegurar que o sistema possa processar sua carga pretendida. Isso envolve planejar uma série de teste em que a carga é constantemente aumentada, até que o desempenho do sistema se torne inaceitável (SOMMERVILLE,2003).

Para finalizar, Pressman (2007) diz que “combinado com teste de estresse frequentemente exige instrumentação de *hardware* e *software*. Ou seja, muitas vezes é necessário medir a utilização de recursos rigorosamente.”

3.3 Ferramentas de Teste

Muitas das diversas que compõem o ciclo de vida de um *software* podem ser apoiadas por ferramentas, que automatizam tarefas e aumentam a produtividade (KOSCIANSKI, 2006).

Uma vez que o teste de *software* frequentemente é responsável por até 40% de todo o esforço despendido num projeto de desenvolvimento de *software*, as ferramentas que podem reduzir o tempo de teste (sem reduzir a eficácia) são muito valiosas. Reconhecendo os benefícios potenciais, pesquisadores e profissionais desenvolveram uma primeira geração de ferramentas de testes automatizadas (PRESSMAN, 2007).

Algumas categorias de ferramentas de testes serão utilizadas como:

1. Analisadores estáticos: Esses sistemas de análise de programa suportam a “comprovação” de afirmações estáticas;

2. Gerador de dados de teste: Esses sistemas de análise automatizados auxiliam o usuário a selecionar dados de teste que fazem um programa comportar-se de uma forma particular;

3. Comparadores de saída: Esta ferramenta torna possível a comparação de um conjunto de saídas de um programa com outro conjunto (previamente arquivado) para determinar a diferença entre elas;

4. Analisadores de fluxo de dados: Essa ferramenta rastreia o fluxo de dados mediante um sistema (semelhante em muitos aspectos aos analisadores de caminho) e tenta descobrir referências a dados, indexação incorreta e outros erros relacionados a dados.

3.4 Ferramentas utilizadas

Ferramentas de automatização de teste estão sendo lançadas cada vez mais no mercado para automatizar as atividades em teste. Existem várias ferramentas dessa categoria, e é improvável que uma única ferramenta seja capaz de executar todas as atividades de teste. A maioria das ferramentas enfoca uma determinada atividade ou grupo de atividades, e algumas só abordam um aspecto de uma atividade.

Após testes em diversas ferramentas e vendo que todas atendem bem a proposta do trabalho, foi selecionado a ESF Database Convert Standard Edition na versão 5.9.66 para a conversão do banco de dados. Essa conversão pode ser verificada no Anexo A desse trabalho. Para a migração dos dados, a ferramenta Migrate Database foi configurada conforme o Anexo B e utilizada conforme o Anexo C.

Como parte do objetivo desse trabalho, será realizada a migração do SGDB do SQL SERVER 2000 da Microsoft para o PostgreSQL. Primeiramente foi utilizada a ferramenta MIGRATE DATABASE (Ferraça, 2006). Após algumas pesquisas, forão levantadas mais duas ferramentas para conversão: ESF Database Convert Standard Edition versão 5.9.66 e a Full Convert Enterprise.

Abaixo a descrição básica de cada ferramenta:

- Migrate Database: Ferramenta gratuita, desenvolvida pelo aluno Maycon Ferraça, maiores informações no seu site: http://sourceforge.net/projects/migratedatabase.

- ESF Database Convert Standart Edition: Testada a versão 5.9.66 da ferramenta, desenvolvida pela empresa Easy From. Maiores informações no seu site: http://www.easyfrom.net.

- Full Convert Entreprise: Para instalar é necessário o Microsoft .Net versão 2.5. Maiores informações no seu site: http://www.spectralcore.com/fullconvert/.

No SistranNet algumas alterações também precisam ser feitas para acessar o novo SGDB. Atualmente o SistranNet acessa o SQL Server com o driver da CoreLab. Para acessar o PostgreSQL, será utilizado o driver da Vita Voom Software: o pgExpress Driver. Maiores informações no seu site http://www.vitavoom.com/products/pgExpress_Driver/index.html.

É consenso que a atividade de teste de *software* não pode prescindir de ferramentas que a automatize ao menos em partes. Para a próxima etapa, após todo o processo de conversão e migração estar concluído, será realizado testes nos SGDBs para a conclusão do trabalho e também as alterações no SistranNet, realizando o objetivo específico desse trabalho.

Para o ensaio do teste de *software*, as ferramentas levantadas tiveram grande auxílio devido os sites OpenSourceTesting e Tigris, que pretendem reforçar o perfil das ferramentas *open source* destinada ao teste de *software. O*s sites proporcionam uma fácil porta de entrada para informações sobre a ampla gama dessas ferramentas disponíveis. Um último site levantado foi o http://www.aptest.com/resources.html, porém, a maioria de suas ferramentas eram pagas, por isso a prioridade para os sites OpenSourceTesting e Tigris (KOSCIANSKI, 2006).

> - “ é um software cujo código fonte é publicado abertamente, e muitas vezes os esforços desenvolvidos por voluntários e está normalmente disponível gratuitamente sob uma licença definida pela Open Source Initiative.” (OPENSOURCETESTING, 2009).

Para esse ponto, algumas ferramentas foram levantadas:

- Jmetter: O JMeter é uma ferramenta open-source do grupo Jakarta Apache, desenvolvida totalmente com tecnologia Java, para a execução de testes de Stress, Performance e de Caixa-preta. Um de seus atrativos é o fato de permitir a execução de plano de testes que podem ser configurados graficamente. Maiores informações no site: http://jakarta.apache. org/jmeter/usermanual/index.html.

- DBExplorer: Com o DBExplorer você pode comparar 2 bancos de dados e gerar um relatório de diferenças de estrutura e de dados. O programa permite que você sincronize as diferenças entre eles, podendo ser feito em dados ou campos selecionados. Maiores informações no site: http://www.dbexplorer.com.

- Test data generator TDG 1.2c: TDG trabalha com SQL ODBC e é capaz de acessar diversos bancos de dados. Ele gera dados de teste ou dados de massa e tem a possibilidade de ver os dados atualizados por um visualizador de banco de dados integrado. Maiores informações no site: http://www.igs-edv.de/.

- Bugzilla: É uma ferramenta integrada à base de esquemas de segurança, inter-dependentes com gráficos de erros, relatórios avançados de capacidades e extensa configurabilidade. disponível para integração com sistemas de gerenciamento de verssionamento automático de software, incluindo Perforce e o CVS (através da interface checkin / checkout de scripts). http://www.bugzilla.org/

- Benerator: É usado para criar um quadro de um alto volume de dados para teste, utilizados para testar e validar configurações. Os dados podem ser importados e exportados a partir de arquivos e sistemas. É altamente personalizável com *plugins* e opções de configuração. http://databene.org/databene-benerat HYPERLINK "http://p-unit.sourceforge.net/"or

- Database Opensource Test Suite (DOTS): É um conjunto de casos de teste concebidos para o teste de estresse no servidor de banco de dados. Utilizado para medir o desempenho e confiabilidade. http://ltp.sourceforge.net/

- DBMonster: É uma aplicação que gera dados aleatórios que servem para testar os bancos de dados SQL de diversas aplicações, utilizando um carga pesada de dados. http://sourceforge.net/projects/dbmonster/

- P-unit: É um framework *open source* para testes de unidade e desempenho, que foi iniciada por Andrew Zhang, sob licença GPL. Tem suporte a *threads e multi-threads*, *tracks memory* e tempo consumo, gerando o resultado em forma de texto simples, imagem ou arquivo pdf. http://p-unit.sourceforge.net/

- Soap Stone: Aplicação de *benchmark* de rede que pode colocar toda a rede em gravação de atividades. http://soap-stone.sourceforge.net/

- Data Generator: *Open source*, escrito em Javascript, PHP e MySQL que permite que gere grandes volumes de dados personalizados em uma variedade de formatos para uso em testes *software* e bases de dados. http://www.generatedata.com/

- QATraq: Abrange tudo que defina planos de teste, escrevendo casos de teste e gravação resultados. http://www.testmanagement.com/

- RTH: Ferramenta projetada para gerenciar requisitos, testes, resultados de teste e defeitos em todo o ciclo de vida da aplicação. A ferramenta oferece uma abordagem estruturada para testar softwares e aumenta a visibilidade do processo de realização do teste, criando um repositório comum, incluindo requisitos, casos de testes, planos de teste, e os resultados do teste. https://sourceforge.net/projects/rth/

- Tesly: É uma aplicação Web escrita em PHP que ajuda a criar, executar e apresentar relatórios sobre o plano de teste. Pode-se acompanhar o andamento dos testes, bem como testadores usar a interface para conclusão do relatório de casos de testes. http://tesly.sourceforge.net/

- Lreport: Ferramenta de linha de comando para comparar CSV e bases de dados (em especial a nível de seleção). http://lreport.sourceforge.net/

- Om: É uma ferramenta open-source de linha de comando que gera dados para testes aleatórios Ela reduz um grande esforço quando é preciso gerar textos de dados de teste. Om pode gerar dados em três maneiras diferentes usando os seus três modos de operação. O usuário pode fornecer uma opção para a geração de dados constituído de diferentes tipos de caracteres como letras maiúsculas, minúsculas, números, etc. Om também vem com manual do usuário. http://omfortesting.110mb.com/

- Wireshark: Anteriormente conhecido como Ethereal, é utilizada por profissionais de rede para a solução de problemas de análise, desenvolvimento de software e protocolos. http://www.wireshark.org/

- SPUnit: É uma aplicação do conceito xUnit que permite testes unitários no SQL Server. http://spunit.sourceforge.net

- TsqlTest: É um simples, leve quadro de teste para o Microsoft SQL Server. Ele utiliza apenas duas tecnologias: T-SQL e scripts. TSqlTest pode ser usado para testar *stored procedures*, *triggers* e *functions*. http://www.tsqltest.org/

- The Delphi Unit Test Expert: É um *AddIn* para o Delphi que cria testes em units do Delphi. O projeto include classes com procedimentos de testes na sua infraestrutura. http://utex.tigris.org/

3.4.1 Característica e comparação das ferramentas

Conforme Ferraça (2008), a tabela “apresenta um comparativo entre as principais características das ferramentas recém analisadas. Estas características são baseadas conforme descrição dos sites dos desenvolvedores, qualquer informação não contida lá, mas mesmo que haja suporte, foi mencionado como sem. Algumas outras características consideradas como somente “comerciais” para exaltar os produtos, tais como “facilidade na utilização do produto” ou “maior gama de recursos”, foram ignoradas. Estas restrições foram aplicadas desta maneira porque a intenção desta comparação não é eleger uma melhor ferramenta, mas sim descobrir as características mais importantes para serem suportadas como base no desenvolvimento da ferramenta proposta neste trabalho.”

Tabela 2: Comparativo de características entre as ferramentas.

Fontes: http://sourceforge.net/projects/migratedatabase, http://www.easyfrom.net, http://www.spectralcore.com/fullconvert/.

Características	**Ferramentas**		
	Migrate Database	ESF Full Convert	Full Convert Enterprise
Suporte a conversão de tabelas	✔	✔	✔
Suporte a conversão de *check constraints*	✔	✔	✔
Suporte a conversão de visões	✔	✔	✔
Suporte a conversão de dados	✔	✔	✔
Suporte a conversão de colunas auto-incremento	✔	✔	✔
Suporte a conversão de chaves únicas	✔	✔	✔
Suporte a conversão de chaves estrangeiras	✔	✔	✔
Suporte a conversão de índices	✔	✔	✔
Suporte a execução de *scripts* pré-conversão e pós-conversão	✘	✔	✔
Suporte a conversão de campos BLOB/CLOB	✔	✔	✔
Suporte a conversão de tipos de dados	✔	✔	✔
Suporte a conversão de nomes de objetos	✔	✔	✔
Suporte a conversão de procedimentos armazenados, funções e gatilhos	✔	✔	✔
Ferramenta de código aberto ou licença gratuita	✔	✘	✘

As características apresentadas na tabela 2 são conforme Ferraça (2008):

Suporte a conversão de tabelas, chaves primárias e *check constraints*: Operações responsáveis por converterem a estrutura das tabelas, incluindo, tipo dos dados, valores *default* das colunas, definição de coluna nula ou não, chaves primárias e *check constraints*;

Suporte a conversão de visões: Responsável pela conversão de visões utilizadas no banco de dados de origem para o banco de dados de destino;

Suporte a conversão de dados: Conversão dos dados de uma base para outra;

Suporte a conversão de colunas auto-incremento: Suporte a conversão de colunas do tipo auto-incremento entre as bases, pois cada SGBD implementa diferencialmente este recurso. Por exemplo, o *Firebird* implementa por gatilhos e *sequences*, sendo necessária a criação destes após a conversão da tabela. Já o ASA, na definição da cláusula *default* de uma coluna [Ferraça, 2007];

Suporte a conversão de chaves únicas: Responsável pela criação de *constraints* do tipo *unique*, que além implementarem consistência dos dados, são necessárias para a conversão de chaves estrangeiras, pois é possível, segundo normas da ANSI-SQL, a criação de chaves estrangeiras através destas [ANSI-SQL 92; ANSI-SQL 99];

Suporte a conversão de chaves estrangeiras: Responsável pela criação de *constraints* do tipo *foreign key*;

Suporte a conversão de índices: Suporte a conversão de índices únicos e não únicos;

Suporte a execução de *scripts* pré-conversão e pós-conversão: Estas operações permitem a execução de scripts para a pré-conversão e pós-conversão dos bancos de dados de origem e destino, de acordo com a necessidade para cada conversão. Importantes, por exemplo, para a necessidade de criação de visões para compatibilidade utilizada no projeto de estágio, que consistiram na criação de visões para cada tabela do banco, com a diferença que todos os identificadores ficaram delimitados por aspas duplas [Ferraça, 2007];

Suporte a conversão de campos BLOB/CLOB: Suporte para a conversão de campos binários e arquivos de textos longos;

Suporte a alteração de tipos de dados: Possibilidade da ferramenta em converter um tipo de dados para outro, por exemplo, a conversão de um tipo de dados de CHAR no banco de dados de origem para VARCHAR no banco de dados de destino [Ferraça, 2007];

Suporte a alteração de *default* de colunas: Possibilidade da ferramenta em converter um valor padrão de uma coluna para outro, por exemplo, a conversão do *default* de

“CURRENT TIME” no banco de dados de origem para “CURRENT_TIME” no banco de dados de destino [Ferraça, 2007];

Suporte a caracteres *unicode*: Oferecer suporte ao conjunto de dados *unicode*;

Suporte a conversão de nomes de objetos: Possibilidade da conversão de nomes de objetos de maiúsculo para minúsculo, maiúsculo para minúsculo, ou nenhuma conversão;

Suporte a conversão de comentários de objetos: Suporte para a conversão para comandos que definem comentários para objetos;

Suporte a conversão de procedimentos armazenados, funções e gatilhos:

Todas as ferramentas analisadas que oferecem este recurso, apenas convertem quando os bancos de dados de origem e destino são os mesmos;

Suporte a controle de tamanho máximo de tipos de objetos entre banco de dados distintos: Alguns bancos de dados possuem limites de tamanho de nomes para objetos [Ferraça, 2007];

Suporte a conversão total de múltiplos bancos de dados relacionais (conversão de dados e metadados): Ferramentas que seja independentes de uma relação finita de bases de origem e destino;

Ferramenta de código aberto ou licença gratuita: Ferramentas que sejam de código aberto ou de licença gratuita.

3.4.2 Bancos de Dados suportados pelas ferramentas

Na Tabela 3 são explicados os bancos de dados suportados por cada uma das ferramentas:

Tabela 3: Comparativo de características entre as ferramentas.

Fontes: http://sourceforge.net/projects/migratedatabase, http://www.easyfrom.net, http://www.spectralcore.com/fullconvert/.

Banco de Dados	Ferramenta		
	Migrate Database	ESF Full Convert	Full Convert Enterprise
Oracle	✗	✓	✓
MySQL	✗	✓	✓
Microsoft SQL Server	✓	✓	✓
PostgreSQL	✓	✓	✓
DB2	✗	✓	✗
Access	✗	✓	✓
Excel	✗	✓	✓
Visual Foxpro	✗	✓	✓
SQLite	✗	✓	✗
Firebird/Interbase	✓	✓	✓
CSV/Text	✗	✓	✓
Paradox	✗	✗	✓
Delimited text files (CSV)	✗	✗	✓
XML	✗	✗	✓
Dbase	✗	✗	✓
Sybase	✓	✗	✗

Com a escolha dos testes e ferramentas que garantirão o funcionamento correto e qualidade do sistema ERP, que irá funcionar juntamente com o PostgreSQL, alguns resultados serão obtidos.

3.5 Considerações Finais

Cada teste será fundamental para a conclusão do trabalho, utilizando as ferramentas disponíveis. O próximo passo será a construção de um plano de projeto, para demonstrar os resultados esperados.

4 PROJETO DE SOFTWARE

O projeto de *software* visa com mais detalhes a organização dos procedimentos necessários, mostrando como será organizada as atividades, fazendo uma modelagem do desenvolvimento do trabalho. Essa parte visa estruturar um conjunto de atividades a serem realizadas. Utilizando a especificação dos requisitos para definir o problema para depois determinar a solução adequada para o problema dos requisitos especificados (PFLEEGER, 2004).

Os requisitos são obtidos no início do desenvolvimento, e o objetivo é determinar a natureza do problema. Os documentos de definição e especificação de requisitos descrevem como o sistema interagirá com o seu ambiente. Os documentos de definição e especificação de requisitos deverão descrever o problema deixando a escolha da solução (PFLEEGER, 2004).

Primeiramente para o projeto de solução do problema abordado, será realizada a conversão do SGDB SQL Server para o PostgreSQL e Firebird. Após a realização dessa tarefa será feita a migração dos dados para os novos SGDBs e as alterações no sistema, para poder executar utilizando o outro SGDB e, por fim, os testes de integração do sistema com os novos SGDBs.

4.1 Requisito da Aplicação

Esta seção apresenta os requisitos de alto nível da aplicação. Os requisitos são mapeados na forma de casos de uso.

Um caso de uso descreve a funcionalidade específica que um sistema, supostamente, deve desempenhar ou exibir, por meio da modelagem do diálogo que um usuário, um sistema externo ou outra entidade terá com o sistema (PFLEEGER, 2004). Exemplo de caso de uso é a Figura 5 que explica o diagrama do caso de uso das tarefas que serão desenvolvidas.

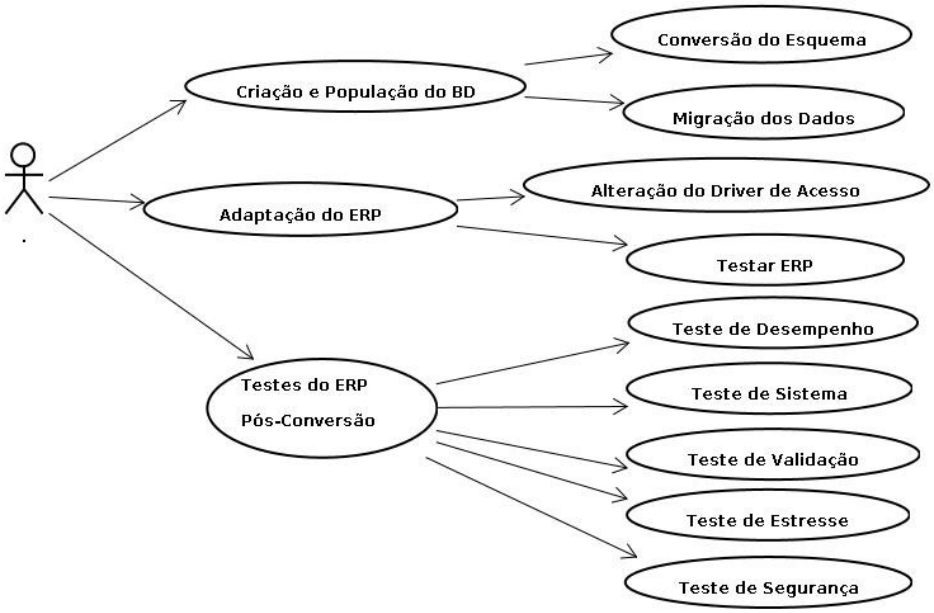


Figura 5: Diagrama de Caso de Uso das tarefas a serem desenvolvidas.

Para cada situação da Figura 5, será criado um Caso de Uso, utilizando como base o modelo de descrição, com base de PFLEEGER (2004). Como primeiro passo e objetivo específico desse trabalho que é a conversão do SGDB, serão realizados os passos conforme podem serem vistos na Tabela 4:

Tabela 4: Caso de Uso de Conversão de Esquema

Caso de Uso:	Conversão do Esquema
Objetivo:	Criar o esquema do banco de dados no PostgreSQL e Firebird com a mesma estrutura atual existente no SQL Server.
Descrição breve:	O propósito desse caso de uso é descrever as atividades envolvidas e a ferramenta utilizada na ação da conversão do SGDB SQL Server para o PostgreSQL e Firebird.
Ferramenta utilizada:	ESF Database Convert Standart Edition e Migrate Database.
Requisitos especiais:	SQL Server, PostgreSQL e ferramenta de conversão instalados.
Pré-condições:	Os mesmos que o requisito especial especifica.
Pós-condições:	Banco de dados no PostgreSQL e Firebird criado e pronto para a migração dos dados e acesso pelo SistranNet.

Após a conversão do esquema do SGDB, a migração dos dados é necessária, mostrando o Caso de Uso na Tabela 5.

Tabela 5: Caso de Uso de Migração de Dados.

Caso de Uso:	Migração dos Dados
Objetivo:	Migrar os dados atuais do SQL Server para o banco de dados criado no PostgreSQL e Firebird.
Descrição breve:	O propósito desse caso de uso é descrever as atividades envolvidas e a ferramenta utilizada na migração dos dados de um SGDB para outro, visando possíveis problemas que possam acontecer em alguns registros.
Ferramenta Utilizada:	Migrate Database e EMS Database Management.
Requisitos especiais:	SQL Server, PostgreSQL, Firebird e a ferramenta de migração instalados juntamente com os *drivers* ODBC de ambos os bancos.
Pré-condições:	Esquema do banco de dados com a estrutura correta do PostgreSQL concluída.
Pós-condições:	Esquema do banco de dados no PostgreSQL e Firebirde com os dados migrados.

Após o esquema do banco de dados estar concluído e os dados migrados, algumas alterações são necessárias no ERP SistranNet para o novo acesso ao SGDB, conforme será explicado na Tabela 6.

Tabela 6: Caso de Uso das Alterações no Driver de Acesso

Caso de Uso:	Alteração no Driver de Acesso
Objetivo:	Realizar alteração no *driver* de acesso do SGDB para o PostgreSQL e Firebird e verificar o bom funcionamento no sistema (cadastros, consultas e relatórios) através de teste, fazendo seu fluxo normal de ações (inserção, exclusão e consultas), buscando por erros.
Descrição breve:	O propósito desse caso de uso é descrever as atividades envolvidas no acesso do sistema SistranNet no novo banco de dados criado no PostgreSQL e Firebird.
Ferramenta Utilizada:	Vitavoom Driver.
Requisitos especiais:	Instalado o novo *driver* de acesso ao SGDB.
Pré-condições:	Banco de dados com a estrutura correta do PostgreSQL e Firebird concluídas.
Pós-condições:	Sistema acessando o novo banco de dados.

Feita a alteração no Driver de Acesso ao SGDB PostgreSQL e Firebird, algumas inconsistências de SQLs, por exemplo, podem acontecer, e para isso outra tarefa a ser executada é um teste simples no sistema, para apenas realizar tarefas rotineiras do sistema, a procura de algum erro que possa acontecer, conforme Tabela 7.

Tabela 7: Caso de Uso das Alterações no Sistema ERP.

Caso de Uso:	Testar ERP
Objetivo:	Após alteração no *driver* de acesso ao SGDB para o PostgreSQL e Firebird, deve-se verificar o bom funcionamento no sistema (cadastros, consultas e relatórios) através de teste, fazendo seu fluxo normal de ações (inserção, exclusão e consultas), buscando por erros.
Descrição breve:	O propósito desse caso de uso é descrever as atividades envolvidas no acesso do sistema SistranNet no novo banco de dados criado no PostgreSQL e Firebird. Possíveis erros de sintaxe e problemas na execução de operações no SGDB.
Ferramenta Utilizada:	The Delphi Unit Test Expert.
Requisitos especiais:	Instalado o novo *driver* de acesso ao SGDB.
Pré-condições:	Banco de dados com a estrutura correta no PostgreSQL e Firebird concluídas.
Pós-condições:	Sistema acessando o novo banco de dados ser erros.

Agora começarão serem exemplificados os Casos de Uso referentes ao teste de conversão. O primeiro dos testes a ser realizado é o teste de desempenho que pode ser verificado na Tabela 8.

Tabela 8: Caso de Uso referente ao teste de Desempenho.

Caso de Uso:	Teste de Desempenho
Objetivo:	Medir a utilização dos recursos rigorosamente.
Descrição breve:	Verificar a quantidade de carga até que o desempenho do sistema se torne inaceitável.
Ferramenta Utilizada:	The Delphi Unit Test Expert.
Requisitos especiais:	Banco de dados criado e alterações no SistranNet já concluídas.
Pré-condições:	Banco de dados com a estrutura correta e concluída no PostgreSQL e Firebird, dados migrados e alterações no SistranNet efetuadas.
Pós-condições:	Gráficos/resultados dos testes efetuados.

O segundo dos Casos de Uso referentes ao teste de conversão é o teste de sistema que pode ser verificado na Tabela 9.

Tabela 9: Caso de Uso referente ao teste de Sistema.

Caso de Uso:	Teste de Sistema
Objetivo:	Pôr completamente à prova o sistema baseado em computador.
Descrição breve:	Testar todo o círculo do *software* em si, toda a estrutura, componentes e elementos que interagem com o mesmo.
Ferramenta Utilizada:	The Delphi Unit Test Expert.
especiais:	Banco de dados criado e alterações no SistranNet já concluídas.
Pré-condições:	Banco de dados com a estrutura correta e concluída no PostgreSQL e Firebird, dados migrados e alterações no SistranNet efetuadas.
Pós-condições:	Gráficos/resultados dos testes efetuados.

O terceiro dos Casos de Uso referentes ao teste de conversão é o teste de validação que pode ser verificado na Tabela 10.

Tabela 10: Caso de Uso referente ao teste de Validação.

Caso de Uso:	Teste de Validação
Objetivo:	Garantir que o sistema funciona de maneira esperada.
Descrição breve:	Verificar se os requisitos funcionais do sistema estão corretos.
Ferramenta Utilizada:	The Delphi Unit Test Expert.
Requisitos especiais:	Banco de dados criado e alterações no SistranNet já concluídas.
Pré-condições:	Banco de dados com a estrutura correta e concluída no PostgreSQL e Firebird, dados migrados e alterações no SistranNet efetuadas.
Pós-condições:	Todo o sistema funcionando e gráficos/resultados dos testes efetuados.

O quarto dos Casos de Uso referentes ao teste de conversão é o teste de estresse que pode ser verificado na Tabela 11.

Tabela 11: Caso de Uso referente ao teste de Estresse.

Caso de Uso:	Teste de Estresse
Objetivo:	Verificar circunstâncias que podem acontecer com uma combinação inesperada de eventos.

Descrição breve:	É importante confrontar o sistema com situações anormais, verificando que o sistema não cause perdas e danos em vez de entrar em colapso.
Ferramenta Utilizada:	The Delphi Unit Test Expert.
Requisitos especiais:	Banco de dados criado e alterações no SistranNet já concluídas.
Pré-condições:	Banco de dados com a estrutura correta e concluída no PostgreSQL e Firebird, dados migrados e alterações no SistranNet efetuadas.
Pós-condições:	Todo o sistema funcionando e gráficos/resultados dos testes efetuados.

O quinto dos Casos de Uso referentes ao teste de conversão é o teste de segurança que pode ser verificado na Tabela 12.

Tabela 12: Caso de Uso referente ao teste de Segurança.

Caso de Uso:	Teste de Segurança
Objetivo:	Verificar se os mecanismos de segurança embutidos no sistema o protegerão.
Descrição breve:	O teste de segurança deve evitar que acessos indevidos possam acessar principalmente os dados, os quais são as informações mais importante do *software*.
Ferramenta Utilizada:	Soap Stone, Wireshark.
Requisitos especiais:	Banco de dados criado e alterações no SistranNet já concluídas
Pré-condições:	Banco de dados com a estrutura correta e concluída no PostgreSQL e Firebird, dados migrados e alterações no SistranNet efetuadas.
Pós-condições:	Todo o sistema funcionando e gráficos/resultados dos testes efetuados.

4.2 Arquitetura do Sistema

Para a realização dos testes, representada na Figura 6, será montada uma estrutura de simulação utilizando máquinas virtuais com bancos de dados SQL Server 2005 (versão testada do sistema com o SGDB) e PostgreSQL. A utilização de máquinas virtuais reduz o tempo de instalação da infra-estrutura de *racks* assim, como também, facilita a correção de eventuais erros que possam vir a ocorrer durante o processo de instalação. Uma vez que a instalação seja mal sucedida basta retornar o último *backup*

da máquina virtual, sem que haja preocupação de reconfigurar as parametrizações. A estrutura montada pode ser verificada na Figura 7, Arquitetura de Hardware.

Figura 6: Arquitetura do Software.

Figura 7: Arquitetura de Hardware

Inicialmente o ambiente de teste será estruturado com a utilização de máquinas virtuais rodando no XenServer 5.0. Será feita uma simulação inicial nessa estrutura, para que após verificado os riscos que possam vir acontecer, sejam testados finalmente no servidor do sistema. O servidor que futuramente será realizado os testes contém a seguinte configuração:

Tabela 13 : Configuração do Servidor.

Servidor:	Servidor PowerEdge 1900
Processador:	02 processadores Xeon dual core 3.06 gz
Disco:	03 HDs 250 GB em raid 5
Sistema Operacional:	Xenserver 5.0

Memória:	08 GB Ram

A divisão das máquinas virtuais foram feitas com as seguintes configurações:

Tabela 14: Configuração de cada máquina virtual no servidor (Terminal Server).

Servidor:	Arquivo e Terminal Server
Processador:	01 Processador Xeon Dual Core 3.06
Disco:	100 GB HD
Sistema Operacional:	Windows Server 2008
Memória:	3 GB de Ram

Tabela 15: Configuração de cada máquina virtual no servidor (FTP).

Servidor:	FTP
Processador:	01 Processador Xeon Dual Core 3.06
Disco:	20 GB HD
Sistema Operacional:	Debian 4.0
Memória:	512 MB de Ram

Tabela 16: Configuração de cada máquina virtual no servidor (E-Mail).

Servidor:	E-Mail
Processador:	01 Processador Xeon Dual Core 3.06
Disco:	120 GB HD
Sistema Operacional:	CentOS 5.2
Memória:	03 GB de Ram

Tabela 17: Configuração de cada máquina virtual no servidor (Sistema).

Servidor:	Sistema
Processador:	01 Dual Core AMD Opteron (tm)
Disco:	250 GB HD
Sistema Operacional:	Windows Server 2003
Memória:	04 GB de Ram

Após a instalação dos sistemas operacionais, Linux e Microsoft Windows, foram realizadas configurações como instalação dos SGDBs e restauração do banco de dados.

4.3 Fluxo de Atividades

Neste ambiente de simulação será avaliada a performance para execução dos testes que representam a base desse trabalho. Foi criada uma sequência para designar como serão realizados os testes, os quais podem ser verificados na Figura 8.

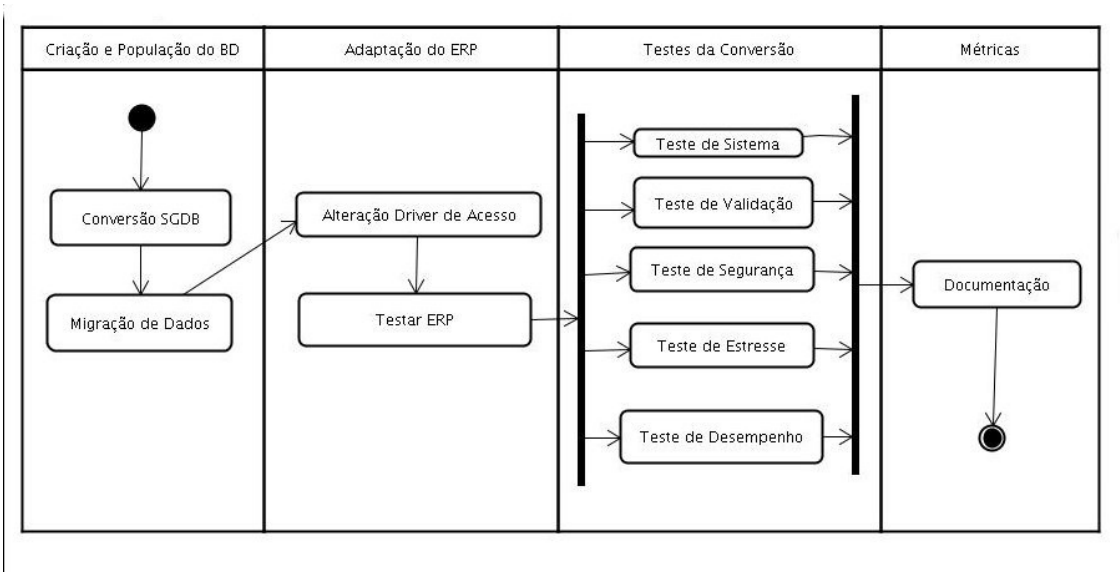


Figura 8: Diagrama de Atividades dividido em cada etapa do desenvolvimento.

4.4 Considerações Finais

Após uma detalhada análise feita nesse capítulo, pode se ter uma boa base para a realização do trabalho que será demonstrada no próximo capítulo, onde será feita toda a modificação nos fontes do sistema, bem como os testes, resultados e conclusões.

5 ESTUDO DE CASO I – MIGRAÇÃO DE DADOS DO ERP SISTRANNET

Após a pesquisa de metodologias que serão aplicadas ter sido feita, deve-se validar a orientação da proposta. Como escolha, a validação será feita através de uso de ferramentas de software juntamente com os estudos de caso já concluídos.

Este capítulo apresenta a execução das atividades demonstrando cada caso de teste separadamente, seguindo um projeto de *software*, sua execução, as ferramentas utilizadas e por fim, são descrito os principais problemas encontrados no desenvolvimento, quais as soluções para estas situações e resultados alcançados.

Para cada caso de teste, será utilizada a arquitetura montada para os testes descrita na seção 4.2. Além disso cada caso de teste utiliza uma tecnologias, e essas ferramentas utilizadas para sua implementação serão analisadas.

5.1 Conversão de Esquema

Seguindo a ordem dos casos de uso, o primeiro é a conversão de ambos os bancos de dados. A conversão foi realizada com a ferramenta ESF Database Convert Full por ser a primeira ferramenta testada e que a sua execução tenha atingido êxito. Essa conversão pode ser acompanhada no Anexo A. A Tabela 18 segue o caso de teste de conversão do esquema:

Tabela 18: Caso de Teste de Conversão de Esquema

Caso de Teste:	Conversão do Esquema
Objetivo Atingido:	Validado a conversão do esquema do banco de dados do SQL Server para o PostgreSQL e Firebird.
Descrição breve:	Descrever e validar as atividades envolvidas e a ferramenta utilizada na conversão do SGDB SQL Server para o PostgreSQL e Firebird.
Condições de Execução:	Utilizar a ferramenta que execute a ação de criar o esquema para o PostgreSQL: ESF Database Convert Standart Edition.
Pré-condições:	SQL Server, PostgreSQL, Firebird e ferramenta de conversão instalados.
Pontos de Observações	Tabelas e campos com estrutura familiar ao SQL Server.
Resultados esperados:	Conversão do Esquema do banco de dados para o PostgreSQL concluído.

Foi realizada a conversão no PostgreSQL 8.2 e Firebird 2.5. Ambos os bancos de dados foram concluídos com sucesso em até 2 minutos e não teve nenhuma dificuldade, principalmente pelo fato de o banco de dados não possuir *views*, *triggers, generators* e *stored procedures*. Todos os bancos foram concluídos com sucesso e os campos corretamente identificados. A Figura 9 demonstra o tamanho dos bancos de dados depois da conversão realizada no Firebird 2.5 e PostgreSQL, nota-se a diferença devido a forma de cada SGDB alocar o espaço em disco.

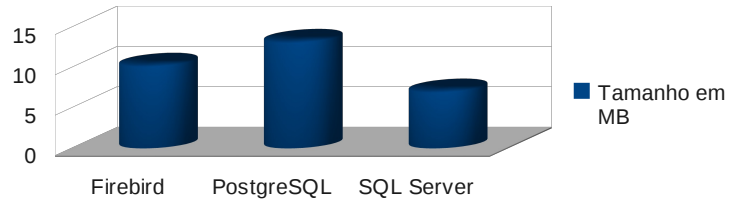


Figura 9: Tamanho dos bancos de dados após a conversão

5.2 Migração dos Dados

O segundo caso de teste realizado foi o da migração dos dados do SQL Server para o PostgreSQL e Firebird. A ferramenta utilizada foi o Migrate Database e na tabela 19, segue o detalhamento do caso de teste. Todo o processo de migração pode ser acompanhado no Anexo B e C.

Tabela 19 : Caso de Teste de Migração de Dados.

Caso de Teste:	Migração dos Dados
Objetivo Atingido:	Validado a Migração dos dados atuais do SQL Server para os banco de dados criado no PostgreSQL e Firebird.
Descrição breve:	O propósito desse caso de teste é descrever e validar as atividades envolvidas com a ferramenta utilizada na migração dos dados de um SGDB para outro, visando possíveis problemas que possam acontecer em alguns registros.
Condições de Execução:	Utilizar a ferramenta que execute a ação de migrar todos os dados para o novo SGDB: Migrate Database.
Pré-condições:	Esquema do banco de dados com a estrutura correta e concluída no PostgreSQL e Firebird.
Pontos de Observações	Observar os dados que não foram exportados, cuidar para transferir-los depois.
Resultados esperados:	Todos os dados migrados do banco de dados do SQL Server para os SGDBs PostgreSQL e Firebird.

O processo de migração foi divido em duas partes. A primeira parte foi a instalação e configuração dos *drivers* ODBC para todos os SGDBs (Anexo B). A segunda parte foi a execução da ferramenta Migrate Database (Anexo C). Duas tentativas de migração foram realizadas, a primeira foi executando SQL por SQL - devido a ferramenta Migrate não obedecer as regras de integridade referencial - após a migração e a segunda foi desativando os *foreign key* e ativando-os novamente após toda a migração.

Nas próximas seções, é descrito as duas migrações realizadas, a primeira utilizando uma ferramenta e a segunda desabilitando as chaves estrangeiras. Por fim é descrito os resultados obtidos durante o processo de migração.

5.2.1 Primeira Migração

Como a ferramenta de migração Migrate Database não obedece as regras de chaves estrangeiras, foi alterado os fontes da ferramenta e modificado a parte de gravação do arquivo texto de *log*, na Figura 10 mostra-se essas alterações. A necessidade dessa alteração foi para corrigir a falta de uma sequência de chaves estrangeiras a ser obedecida.

```
function TLog.WriteLogDMLError(Line: Integer; Error, DML,
SourceDBMS, TargetDBMS: String): Boolean;
var
     F: TextFile;
     Aux, Msg: String;
begin
Result := True;
try
     {$i-}
     Aux := Util.SystemPath + '\dml.log';
     AssignFile(F, Aux);
     if FileExists(Aux) then
          Append(F)
     else
          Rewrite(F);
     Msg := '--------------------------' + #13#10 +
            'Timestamp: ' + DateTimeToStr(Now) + #13#10 +
            'Source DBMS: ' + SourceDBMS +  #13#10 +
            'Target DBMS: ' + TargetDBMS +  #13#10 +
            'Line: ' + IntToStr(Line) +  #13#10#13#10 +
            Copy(DML, 1, Length(DML) - 2)  + #13#10#13#10 +
            Error + #13#10;

     // Writeln(F, Msg);
     WriteLn(F, Copy(DML, 1, Length(DML) - 2));
     CloseFile(F);
     {$i+}
except
     Result := False;
end;
end;
```

Figura 10 : Alterações na ferramenta Migrate Database.

A alteração na ferramenta foi a de trocar da forma de gravação do arquivo de *log*. Em vez de gravar a mensagem completa, com a hora, base de origem e destino, linha, SQL de execução e erro, gravou apenas a SQL de execução. Depois de terminada a execução da ferramenta, foi gerado o *log* que pode ser visto na Figura 11.

```
INSERT INTO CIDADE (codigodacidade, uf, cidade, cepdalocalidade, tipode...
INSERT INTO CIDADE (codigodacidade, uf, cidade, cepdalocalidade, tipode...
INSERT INTO CLIENE (cpfcnpj, razaosocial, fantasia...
```

Figura 11 : Novo modelo de *log* salvo após as alterações na ferramenta.

Mesmo após a execução da ferramenta Migrate, alguns dados não foram transferidos, para contornar essa situação, foi criada uma ferramenta em Delphi 7, que lê o *log* gravado, e vai executando as SQLs até o fim, marcando o que foi executado, e retornando até o início e novamente fazendo o processo, até que todos as linhas tenham sido executadas.

Para contornar o problema de SQLs que não foram executadas pela ferramenta Migrate Database durante o caso de teste de migração dos dados, foi criada uma ferramenta que leia o *log* e execute-o linha a linha.

A Figura 11 mostra o procedimento principal dessa ferramenta, e a próxima imagem, a Figura 12, mostra a única tela existente na ferramenta, ainda em tempo de desenvolvimento.

Unit1

```
procedure TForm1.Button1Click(Sender: TObject);
var j: integer;
begin
  DBTarget.Connected := true;
  CheckListBox1.items.LoadFromFile('dml.log');
  memo1.Lines.Clear;
  memo1.Lines.add(timetostr(now));
  i := CheckListBox1.Items.Count - 1;
  while i > 0 do
  begin
    Application.ProcessMessages;
    for j := 0 to CheckListBox1.Items.Count - 1 do
    begin
      CheckListBox1.ItemIndex := j;
      if not CheckListBox1.Checked[j] then
      begin
        Application.ProcessMessages;
        try
          //SQLConnection1.ExecuteDirect(CheckListBox1.Items[j])
          Query1.sql.Text := CheckListBox1.Items[j];
          Query1.ExecSQL;

          CheckListBox1.Checked[j] := true;
        except
          on e: exception do
            Memo1.lines.add(e.Message);
        end;
      end;
    end;
  end;
  memo1.Lines.add(timetostr(now));
end;
```

Figura 12: Procedimento principal da execução da ferramenta.

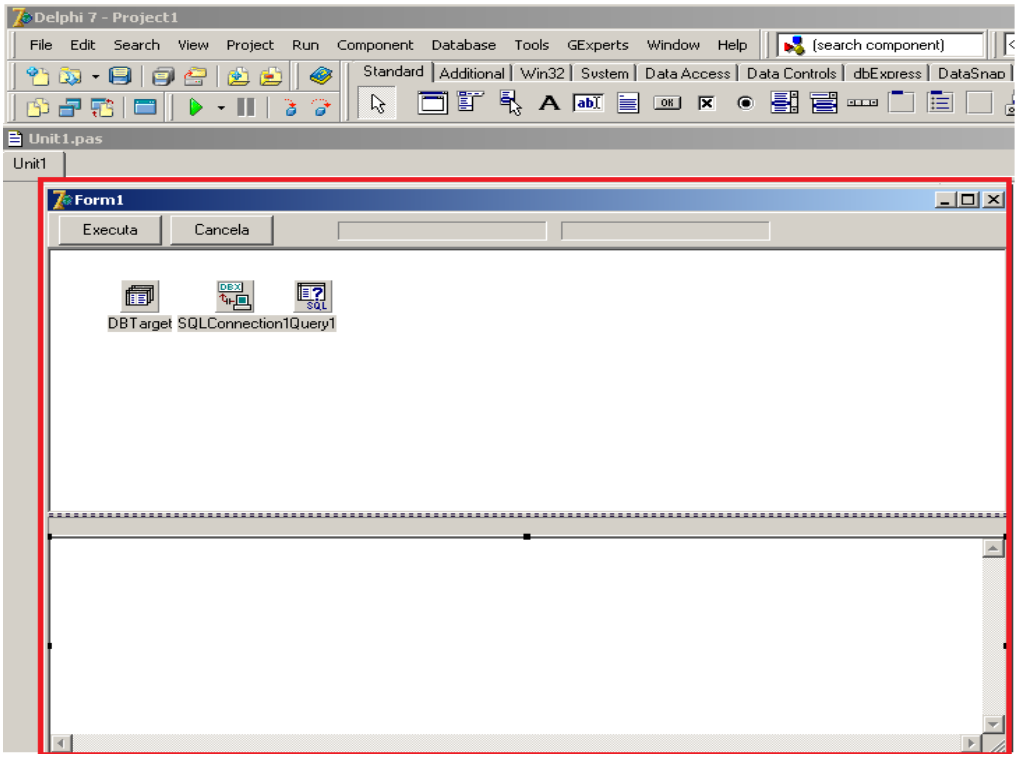


Figura 13: Única tela da ferramenta de auxílio.

A base de execução da ferramenta de auxílio é iniciá-la e clicar no botão “Executa”. Enquanto ela é executada as SQLs que geram algum erro são incluídas na parte de baixo, com a mensagem do erro e, pode-se cancelar o processo, clicando no botão “Cancelar”, caso o usuário queira.

5.2.2 Segunda Migração

A segunda forma de migração dos dados foi a de exclusão dos *foreign key* do banco de dados e recriação dos mesmo após a completa transferência dos dados. O processo foi realizado com os seguintes passos:

1. Extraído os metadados dos banco de dados;
2. Alterado o arquivo e retirada a parte de criação dos *foreign key*;
3. Recriado os banco de dados;
4. Executado a ferramenta de migração, a qual não retornou nenhum erro;
5. Executo as SQLs de criação dos *foreign key*.

5.2.3 Resultados

Para descrever os resultados, em cada gráfico é utilizado siglas, que são explicadas da seguinte forma:

- Cli X : Banco de dados de um Cliente, denominado pelo número que vem após, exemplo: Cliente 1, Cliente 2 e Cliente 3.
- FB : Firebird.
- PG : PostgreSQL.
- MS : Microsoft SQL Server.

A Figura 14 mostra o tempo gasto pela ferramenta para transferir os dados nos dois SGDBs. São amostragens de dois clientes, com bases de dados diferentes.

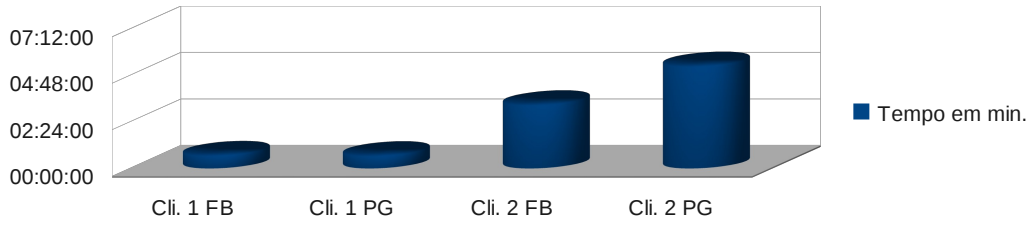


Figura 14 : Gráfico de tempo em minutos da migração dos dados do SQL Server para o PostgreSQL e Firebird em dois clientes.

Após a migração completa de todos os dados e tabelas, os bancos de dados obtiveram tamanhos variados que podem serem verificados no gráfico da Figura 15.

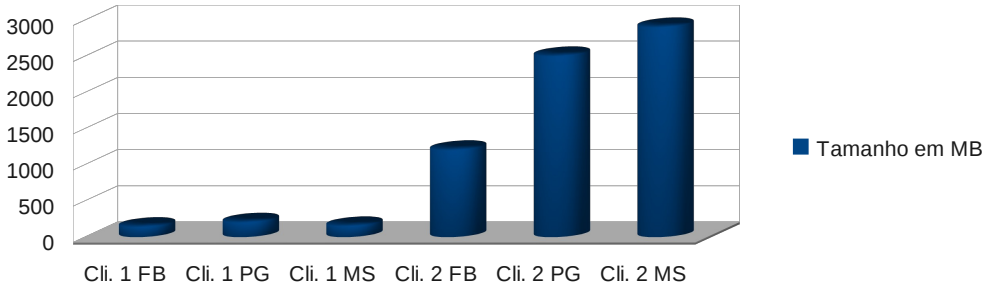


Figura 15 : Tamanho dos bancos de dados após a migração em Bytes.

Na Figura 16, tem-se o tamanho dos arquivos de *backups* feitos diariamente em ambos os SGDBs. Todos são do mesmo cliente.

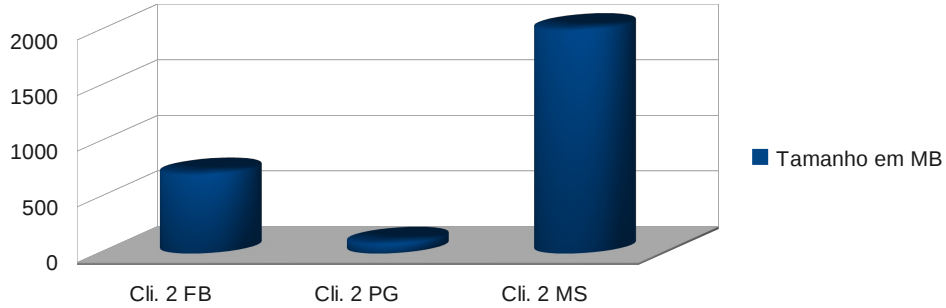


Figura 16 : Tamanho dos arquivos de *backups*.

Concluindo, percebe-se o tamanho do banco de dados muito maior no SQL Server, bem como o tempo de processamento do PostgreSQL ser maior que o do Firebird.

5.3 Alteração no Driver de Acesso

Depois do esquema do banco de dados estar pronto no SGDB PostgreSQL e Firebird, e os dados migrados, há uma última etapa antes da realização dos testes: as alterações na programação dos fontes do sistema SistranNet. Como a arquitetura do sistema é em três camadas, visto na Figura 17, o trabalho é facilitado por precisar alterar apenas a parte servidora, segunda camada, e SQLs de execução de consultas que estejam localizadas na camada cliente da aplicação. Deve-se ter um cuidado maior por se tratar de um sistema ERP, para não disponibilizar versões com erros para os clientes. O caso de testes é demonstrado na Tabela 20.

Tabela 20: Caso de Teste das Alterações no Driver de Acesso

Caso de Teste:	Alteração no Driver de Acesso
Objetivo atingido:	Validado a alteração no *driver* de acesso ao SGDB para o PostgreSQL e Firebird.
Descrição breve:	O propósito desse caso de teste é descrever e validar as atividades envolvidas no acesso do sistema SistranNet nos novos esquemas criados no PostgreSQL e no Firebird.
Condições de Execução:	Utilizar a ferramenta que execute a ação de acesso do novo *driver* para o PostgreSQL e Firebird. Vitavoom Driver.
Pré-condições:	Instalado o novo driver de acesso ao SGDB.
Pontos de Observações	Observar DLLs e arquivos especiais para a configuração do novo *driver* de acesso.
Resultados esperados:	Sistema acessando as novas bases de dados no PostgreSQL e Firebird.

Foi instalado o PostgreSQL 8.2 juntamente com o *driver* da VitaVoom versão 4.1.1, o qual é compatível com o PostgreSQL versão 8.x. O processo de instalação baseia-se em alterar os arquivos de conexões do dbExpress e copiar suas respectivas DLLs para a pasta de instalação do sistema operacional. Mais detalhes da instalação pode ser conferido no próprio site do fabricante: http://www.vitavoom.com/products/pgedriver/docs/installation.html.

A plataforma dbExpress é uma biblioteca de bancos de dados (DBX), disponível para plataformas Windows e Linux. Chama-se biblioteca e não mecanismo de banco de dados, porque ao contrário das outras soluções para o Delphi, o dbExpress usa uma estratégia leve e basicamente não exige nenhuma configuração nas máquinas dos usuários finais, e, permite escrever um aplicativo que pode acessar muitos mecanismos

de banco dados diferente (InterBase/Firebird, Oracle, DB2, MySql, Informix, Microsoft SQL Server, PostgreSQL) sem muita modificação no código fonte (CANTÙ, 2003).

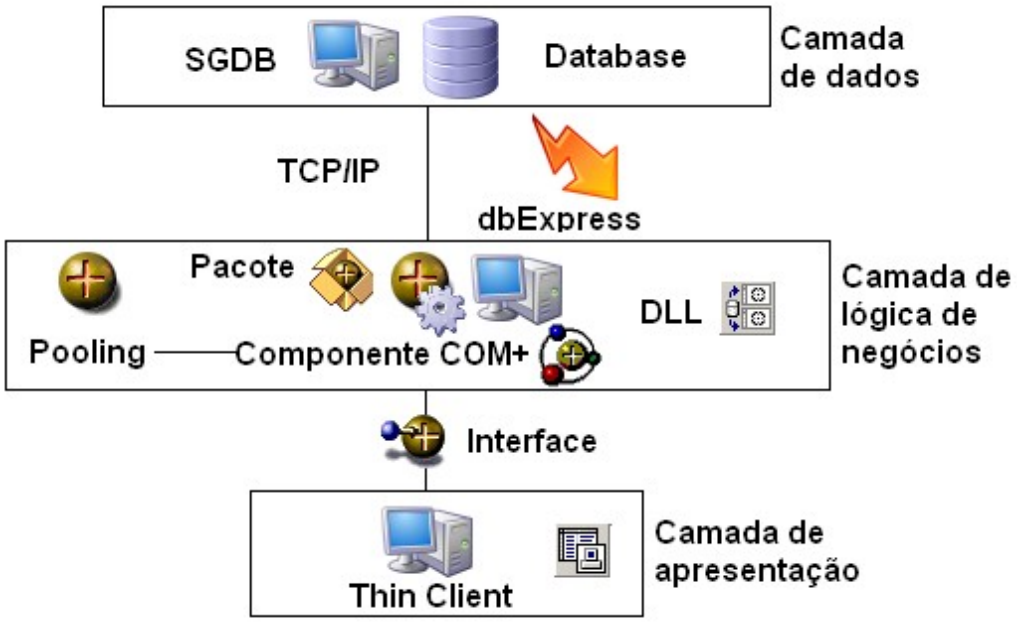


Figura 17 : Solução Multi-camadas.
Legenda: Pooling: Otimizador de acessos aos dados através de uma utilização racional das conexões feitas com uma fonte de dados.
Fonte: (CLUBE DELPHI, 2006).

No caso da eSafety, a biblioteca do dbExpress, foi utilizada para escrever além da camada cliente, a camada servidora também personalizada. Um melhor entendimento, pode ser conferido na Figura 18.

As alterações começam na arquivo de inicialização do sistema, antes esse arquivo possuia apenas configurações para acesso ao SQL Server, agora ele passa a conter também as configurações para os SGDBs PostgreSQL e Firebird, juntamente com uma constante inicial, que terá o valor do SGDB ativo (FB para Firebird, PG para PostgreSQL e MS para SQL Server).

A Figura 19 mostra como ficou o arquivo que fica salvo na pasta System32 do Windows: é importante notar que esses arquivos de inicialização são usados apenas em tempo de projeto. Quando seleciona-se o um *driver* ou conexão em tempo de projeto, os valores desses arquivos são copiados nas propriedades correspondentes ao componente de conexão.

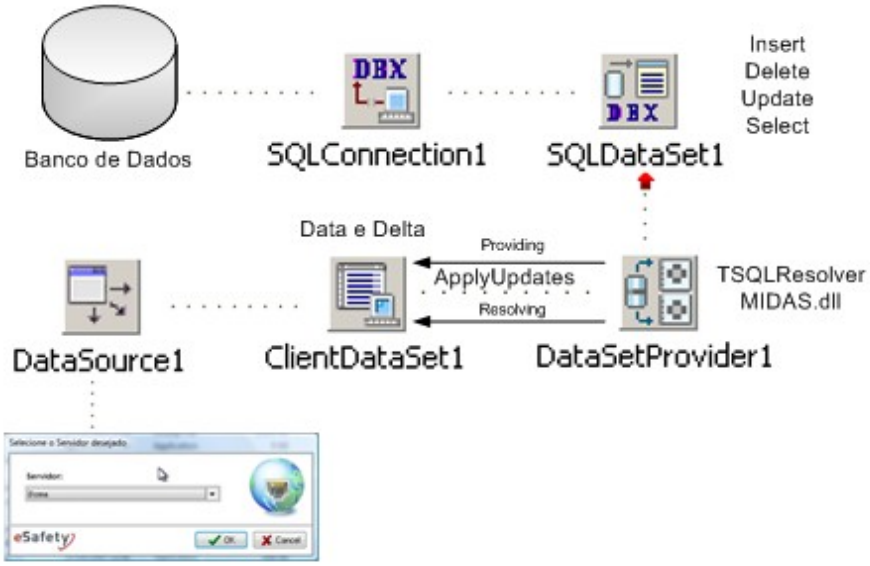


Figura 18 : Funcionamento do dbExpress e DataSnap.
Fonte: (BLUMM *at. Al*, 2006).

O componente de conexão ao SGDB, usa as informações disponíveis nos arquivos *dbxdrivers.ini* e *dbxconnections.ini*, os quais são arquivos de configuração do dbExpress (ambos do Delphi). O primeiro arquivo lista os *drivers* disponíveis, um para cada banco de dados suportado. Para cada *driver*, existe um conjunto de parâmetros de conexão padrão. Os parâmetros do arquivo *dbxdrivers.ini* indicam a DLL de *drivers*, a função de entrada a ser usada, a biblioteca-cliente do fornecedor e outros parâmetros específicos dependentes de cada SGDB. Já os parâmetros do arquivo *dbxconnections.ini* fornece a descrição específica do banco de dados. Essa lista associa as configurações a um nome, e pode inserir detalhes de múltiplas conexões para cada *driver* de banco de dados. A conexão descreve um banco de dados físico ao qual deseja-se conectar (CANTÙ, 2003).

Importante salientar que a junção desses dois arquivos resultam no arquivo representado na Figura 19, ou seja, os *drivers* instalados juntamente com as configurações de cada conexão. Para a realização da conexão, foi inserido a configuração de apenas três SGDBs: Microsoft SQL Server, PostgreSQL e Firebird.

```
[Banco]
Ativo=PG
```

[Sql Server]	[PostgreSQL]	[Firebird]
BlobSize=-1	BlobSize=32	BlobSize=-1
HostName=.	HostName=localhost	DataBase=D:\SistranNetFB\SIS
DataBase=SistranNet	DataBase=SistranNet	TRANNET.FDB

```
DriverName=SQLServer          DriverName=DevartPostgreSQL   DriverName=Interbase
User_Name=                    User_Name=postgres            User_Name=SYSDBA
Password=                     Password=postgres             Password=masterkey
LongString=True               LongString=True               LongString=True
EnabledBCD=True               EnabledBCD=True               EnabledBCD=True
FetchAll=True                 FetchAll=True                 RoleName=RoleName
Prepared=True                 Prepared=True                 CommitRetain=False
Connection Timeout=10         Connection Timeout=10         Interbase
                                                            TransIsolation=ReadCommited
                                                            SQLDialect=3
```

Figura 19: Arquivo de inicialização do sistema, contendo as configurações de acesso de cada SGDB.

Como o sistema é dividido em módulos, e também está na arquitetura de três camadas, todos os módulos tiveram que ser recompilados devida a alteração na parte de leitura da configuração de inicialização do sistema.

Conforme BLUMM *at. al* (2006), alguns cuidados devem ser tomados para criar uma aplicação multi banco de dados em Delphi utilizando o dbExpress. Na Tabela 21 estão relacionados os tipos mais comuns em cada banco de dados e o equivalente nos outros SGDBs de dados. É importante também cuidar de tipos de dados específicos, os quais não são utilizados na eSafety, sendo assim, facilitando as consultas realizadas pelo sistema.

Tabela 21 :Tipos de dados dos SGDBs.
Fonte: (BLUMM *at. Al*, 2006).

Tipo	**Oracle**	**MS SQL Server**	**Firebird**	**PostgreSQL**
Alfanumérico (variável)	Varchar2	Varchar	Varchar	Varchar
Alfanumérico (fixo)	Char	Char	Char	Char
Binário	Long Raw	Image	Blob	Bytes
Data/Hora	Date	DateTime	Date	Date
Inteiro	Int/Integer	Int	Integer	Int8
Numérico	Number	Decimal	Decimal	Numeric
Text	Long	Text	Varchar	Text

Para solucionar o problema de incompatibilidades nos tipos de campos, é utilizado a função c*ast* disponível em todos os SGDBs aqui utilizados, essa função transforma o tipo do valor de uma expressão, deixando os valores resultantes da consulta em um padrão reconhecido pelo dbExpress, exemplo:

```
Select CR.CodigoDaReclamacao,
cast(Count(CR.CodigoDaReclamacao) as integer) as QtdeReclamacao,
cast(Max(CRT.DescricaoReclamacao) as char(50)) as DescricaoReclamacao
from CLIENTERECLAMACAO CR
left join CLIENTETIPODERECLAMACAO CRT
```

```
on CR.CodigoDaReclamacao=CRT.CodigoDaReclamacao
group by CR.CodigoDaReclamacao;
```

Além dessas outras situações foram tratadas refente aos SGDBs nos fontes do sistema. A primeira foi a transformação de todas as SQLs contidas no sistema, na parte de servidor (segunda camada) para minúsculas. Isso porque o *driver* não faz como que o PostgreSQL reconheça as tabelas, caso o nome da tabela em um simples *select * from NOME_DA_TABELA* for maiúscula, um erro é retornado dizendo que a *Relation* mais o nome da tabela não é reconhecido pelo SGDB. A segunda alteração cabe aos "apelidos" nas consultas, muitas dessas consultas não tinham a palavra-chave "as", a qual é obrigatória no PostgreSQL, e nos demais opcional.

Foi necessário privar-se de usufruir de muitos bons recursos dos SGDBs (como por exemplo funções, procedimentos, operadores) para deixar a solução portável, na Tabela 22 é apresentado as funções que eram utilizadas com padrão no SGDB SQL Server, e que tiveram que ser retiradas e tratadas diretamente nos fontes do sistema. As funções padrões como o *Sum, Count, Min. Max* e *Avg* foram mantidas por serem universais a todos os SGDBs utilizados no sistema.

Tabela 22 : Tipos de funções dos SGDBs.
Fonte: (CLUBE DELPHI 75, 2006).

Objetivo	**SQL Server**	**Firebird**	**PostgreSQL**
Selecionar campos nulos	Isnull ou Coalesce	Coalesce	Coalesce
Separar o ano de uma data	Year(coluna)	Extract(Year from coluna)	Extract(Year from coluna)
Separar o mês de uma data	Day(coluna)	Extract(Day from coluna)	Extract(Day from coluna)
Concatenação	String + String	String || String	String || String

Durante a execução deste caso de teste, apareceram as seguintes divergências:

- Problema Concatenação: Para o problema de concatenação de *strings*, foi utilizado os chamados "campos calculados", campos que são computados para cada registro e recalculados sempre que o registro é carregado em um *buffer* interno. (CANTÙ, 2003). Na prática esses campos foram criados para resolver o problema de selecionar dois campos juntos em uma instrução SQL, assim, foi separado os campos, selecionando os dois e concatenado-os em tempo de execução pelo próprio componente de acesso do dbExpress, no caso o ClientDataSet.

- *Scripts* de Atualização: As atualizações no sistema, eram encarregadas por um segundo sistema, Figura 20, o qual carrega um arquivo XML contento as linhas de SQL a serem atualizadas no banco de dados da conexão atual, ou seja, quando for necessária a atualização de algum banco de dados de um cliente, conecta-se no banco de dados e executa esse sistema. Pelo motivo dos comandos SQLs de criação de tabelas, adição de colunas, exclusão de tabelas, etc., serem diferentes para cada banco de dados, a solução foi criar outro arquivo XML, específico para cada banco de dados. O esquema do arquivo XML no formato *XMLschema* que contém os comandos a serem executados no banco de dados, pode ser visto na Figura 21.

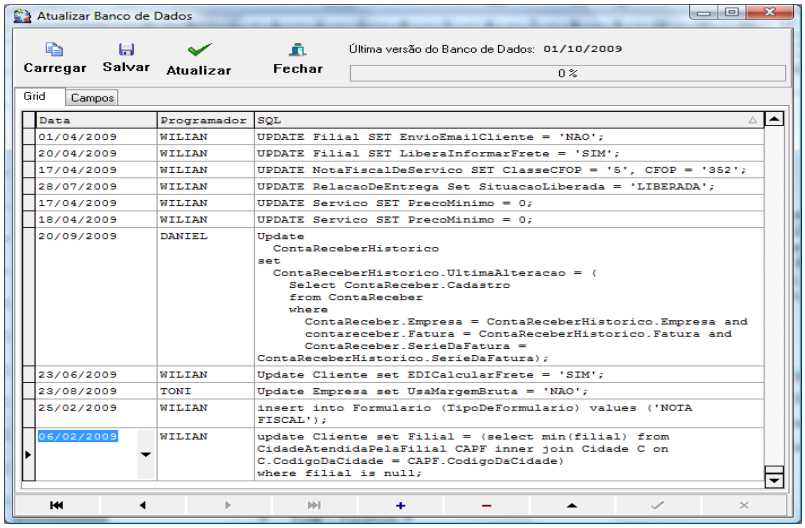


Figura 20: Sistema para atualização de bancos de dados da eSafety.

```
<?xml version="1.0" encoding="utf-8" ?>
- <xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
  - <xs:element name="DATAPACKET">
    - <xs:complexType>
      - <xs:sequence>
        - <xs:element name="METADATA">
          - <xs:complexType>
            - <xs:sequence>
              - <xs:element name="FIELDS">
                - <xs:complexType>
                  - <xs:sequence>
                    - <xs:element maxOccurs="unbounded" name="FIELD">
                      - <xs:complexType>
                          <xs:attribute name="attrname" type="xs:string" use="required" />
                          <xs:attribute name="fieldtype" type="xs:string" use="required" />
                          <xs:attribute name="WIDTH" type="xs:unsignedByte" use="optional" />
                          <xs:attribute name="SUBTYPE" type="xs:string" use="optional" />
                        </xs:complexType>
                      </xs:element>
                    </xs:sequence>
                  </xs:complexType>
                </xs:element>
              - <xs:element name="PARAMS">
                - <xs:complexType>
                    <xs:attribute name="CHANGE_LOG" type="xs:string" use="required" />
                  </xs:complexType>
                </xs:element>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        - <xs:element name="ROWDATA">
          - <xs:complexType>
            - <xs:sequence>
              - <xs:element maxOccurs="unbounded" name="ROW">
                - <xs:complexType>
                    <xs:attribute name="RowState" type="xs:unsignedByte" use="required" />
                    <xs:attribute name="Data" type="xs:unsignedInt" use="optional" />
                    <xs:attribute name="Programador" type="xs:string" use="optional" />
                    <xs:attribute name="SQL" type="xs:string" use="optional" />
                  </xs:complexType>
                </xs:element>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
        <xs:attribute name="Version" type="xs:decimal" use="required" />
      </xs:complexType>
    </xs:element>
  </xs:schema>
```

Figura 21: Estrutura do arquivo XML que contém as instruções SQLs.

- Filtro de Datas: No dbExpress, existe um tipo de campo TSQLTimeStampField, o qual é mapeado para o tipo de dados estampa de tempo (*timestamp*) que todos os SGDB aqui utilizados suportam. Esse tipo de campo é uma representação simples baseada em registro de uma data ou hora. Um campo de estampa de tempo, pode converter valores de data e hora padrão automaticamente, usando a propriedade *AsDateTime*, podendo fazer conversões personalizadas e realizar mais manipulações de estampas de tempo por meio de rotinas fornecidas na

unidade *SqlTimSt*, incluindo funções como *DateTimeToSQLTimeStamp*, *SQLTimestampToStr* e *VarSQLTimeStampCreate*. Segundo Cantù (2003), é definida da seguinte forma:

```
TSQLTimeStam = packed record
        Year : Smallint;
        Month : Word;
        Day : Word;
        Hour : Word;
        Minute : Word;
        Second : Word;
        Fractions : Word;
end;
```

Além das alterações principais demonstradas acima, houveram outras divergências que tiveram que ser tratadas para o correto funcionamento do sistema SistranNet. Na Tabela 23, encontra-se o problema principal, a descrição e a sua solução.

Tabela 23: Divergências e solução utilizadas para instruções SQLs.

Problema	**Descrição**	**Solução encontrada**
Função SQL *CASE WHEN*	(CASE WHEN CP.Fornecedor IS NULL THEN CP.DadoFornecedor ELSE Cli.RazaoSocial END) Fornecedor	Nesse caso, como a função funciona para testar se o primeiro campo for nulo, retorna o o segundo campo, pode ser facilmente substituído pela função coalesce.
Função SQL *CASE WHEN (II)*	SELECT CASE WHEN MAX(EntradaSaida.Nota) > MAX(NotaFiscalDeServico.NotaFiscal) THEN MAX(EntradaSaida.Nota) WHEN MAX(NotaFiscalDeServico.NotaFiscal) > MAX(EntradaSaida.Nota) THEN MAX(NotaFiscalDeServico.NotaFiscal) ELSE MAX(EntradaSaida.Nota) END AS Nota FROM	Para o PostgreSQL, essa instrução SQL não funcionou, por gerar uma exceção de case com a função max, sendo assim foi necessário selecionar os dois campos e testa-los com o próprio Borland Delphi.
Função SQL *DatePart*	DatePart(Year,Cnh.DataDeEmissao) as Ano, DatePart(Month, Cnh.DataDeEmissao) as Mes, DatePart(Day, Cnh.DataDeEmissao) as Dia From tabela Order by ….	Como a função DatePart não exist no PostgreSQL, foi modificado o código para utilizar a função Day, Month e Year, descritas no Anexo E.
Função SQL *Top*	Select Top from tabela	Em todos os registros encontrados pelos fontes, todos se referiam a seleção de apenas 1 registro, para isso foi modificado a ordem dos campos para selecionar a ordem ao contrário, do maior para o menor. Em visto que os registro selecionados são poucos, não interferiu no desempenho do sistema.
Função SQL *GetDate()*	UPDATE Pedido set EmailEnviado=GetDate()	A função GetDate pode ser criada no PostgreSQL, mas para manter o padrão de SQL no sistema, foi modificado para o filtro ser setado pelo próprio Delphi.

Função SQL *If Not Exists(*	IF NOT EXISTS (SELECT ControlePedido) INSERT....	Nesse caso foi retirado o primeiro select para uma outra instrução SQL, e caso o resultado obtido for vazio, ai sim é feio o insert.
Função SQL Coalesce(Max (	SELECT Coalesce(MAX(NotaFiscalDeServico.NotaFiscal)),0)	Conforme descrito na Função SQL *CASE WHEN (II), o PostgreSQL não suporta esse tipo de operação, para isso é necessário utilizar a instrução CAST, para modificar o tipo do campo selecionado.*
Função SQL RTrim e LTrim	Select rtrim(Fantasia) from Cliente	Para manter o padrão, foi retirado esse tipo de instrução, para futuramente facilitar para os próximos SGDBs.
Dbo	Select * from dbo.tabela	Retirado em algumas instruções a cláusula dbo, que é específica para o SQL Server, assim como a public para o PostgreSQL, utilizada para setar o esquema selecionado.
Função SQL isNull	Select isnull(Fantasia, RazaoSocial)	Função que pode ter duas soluções: 1) Criar a função isNull 2) Modificar para a função coalesce, que tem o mesmo significado.
Função SQL Len	Where Len(Pro.CnpjCpf)=14	Sempre que a função len foi utilizada, foi para verificar se o campo código era um CNPJ ou CPF, pelo tamanho de caracteres, o que foi substituído pela seguinte instrução: Pro.CnpjCpf not like "/"

5.4 Considerações Finais

Nessa primeira parte da execução do plano de projeto, pode-se perceber a importância da utilização de ferramentas para a execução desses casos de teste. Tanto para a conversão como para a migração. A demora maior foi na execução do caso de teste referente as alterações nos fontes do sistema SistranNet, devida a grande quantidade de arquivos no código fonte (mais de mil). Após essa alteração, chega a hora de realizar os testes no sistema, para validar essas alterações e a conclusão do trabalho.

No próximo capítulo , agora que o banco de dados já está pronto para o PostgreSQL (convertido, migrado e com os dados transferidos), é apresentado os testes realizados no sistema e no banco de dados , bem como os resultados adquiridos com a execução dos testes.

6 ESTUDO DE CASO II – TESTES DO ERP SISTRANNET NO SGDB POSTGRESQL E SGDB FIREBIRD

A qualidade do *software* é hoje uma das maiores preocupações de qualquer empresa de tecnologia envolvida com o desenvolvimento de software. E consiste na análise dinâmica do produto, ou seja, na sua execução com o objetivo de provocar a falha nesse produto, contribuindo para uma futura detecção de defeitos através de um processo de depuração e, consequentemente, o aumento da confiança de que ele esteja correto. Com base nesse contexto, cabe após todas as alterações feitas nos fontes do sistema SistranNet, testar essas alterações, em ambos os SGDBs que aqui são trabalhados. Para isso, verificamos seus detalhes no caso de teste representado pela Tabela 24.

Tabela 24 : Caso de Teste das Alterações no Sistema ERP.

Caso de Teste:	Testar ERP
Objetivo Atingido:	Validado o sistema ERP após alteração no *driver* de acesso ao SGDB para o PostgreSQL
Descrição breve:	O propósito desse caso de teste é descrever e validar as atividades envolvidas no acesso do sistema SistranNet no esquema criado no PostgreSQL. Possíveis erros de sintaxe e problemas na execução de operações no SGDB, deve-se verificar o bom funcionamento no sistema (cadastros, consultas e relatórios), fazendo seu fluxo normal de ações (inserção, exclusão e consultas), buscando por erros.
Condições de Execução:	Utilizar a ferramenta que execute a ação de verificação nas operações básicas no sistema: Inclusão, Exclusão e Alteração. *The Delphi Unit Test Expert.*
Pré-condições:	Banco de dados com a estrutura correta do PostgreSQL concluída.
Pontos de Observações	Observar erros de sintaxe de SQL e observar principalmente se os dados foram gravados no banco de dados.
Resultados esperados:	Todo o sistema funcionando corretamente no PostgreSQL.

Os primeiros testes realizados no sistema foram testes manuais, que efetuam aplicações básicas no sistema como novos cadastros, exclusões de registros, consultas e geração de relatórios, onde o sistema é executado e as informações são inseridas manualmente, verificando se os resultados esperados foram alcançados. No entanto, esta abordagem é lenta e muito ineficaz, por isso se é necessária a utilização de uma ferramenta com um processo automatizado, buscando assim testar o máximo possível dos requisitos do *software*.

O objetivo desse caso de teste é apresentar os testes projetados e realizados no sistema, essa estratégia de teste foi realizada utilizando a ferramenta *DUnit* no ambiente de desenvolvimento *RAD Studio 2009* da *Embarcadero (http://www.embarcadero.com.br)*. Com essa ferramenta pode-se planejar e construir testes para um trecho ou módulo do sistema, com isso, os testes podem serem feitos de forma incremental (Daibert *at al*, 2008).

A ferramenta foi configurada com base no artigo "Teste Unitários de Software"(Daibert *at al*, 2008). Importante destacar que a ferramente *DUnit* é independente do ambiente de desenvolvimento, pois elas carregam a biblioteca compilada e executa os testes configurados.

Nas seções 6.1 a 6.5 serão descritos os testes de Desempenho, Sistema, Validação, Estresse e Segurança. Por fim, a seção 6.6 faz uma conclusão final ao capítulo.

6.1 Testes de Desempenho

Os testes de desempenho mostram valores reais quando o sistema está em execução, pois a muitos fatores que determinam o tempo de resposta do sistema: servidor, rede, SGDB, etc. O caso de teste referente ao desempenho do sistema pode ser visto na Tabela 25.

Tabela 25 : Caso de Teste referente ao teste de Desempenho.

Caso de Teste:	Teste de Desempenho
Objetivo Atingido:	Validado a utilização dos recursos do sistema rigorosamente.
Descrição breve:	O propósito desse caso de teste é descrever e validar as atividades envolvidas na verificação da quantidade de carga que o desempenho do sistema suporta até que se torne inaceitável.
Condições de Execução:	Utilizar a ferramenta que execute a ação de verificação do desempenho do sistema. SQL Manager 2007, PgAdminIII, Borland Delphi 7, SistranNet.
Pré-condições:	Banco de dados com a estrutura correta do PostgreSQL concluída, dados migrados e alterações no SistranNet efetuadas.
Pontos de Observações	Observar os resultados de ambos os bancos de dados.
Resultados	Dados para a documentação dos testes.

esperados:	

O teste de desempenho tem um fator determinante que é a quantidade de dados, seja em todo o banco de dados, ou seja apenas em uma tabela. Com a ajuda da ferramenta *SQL Manager 2007 for PostgreSQL*, foi possível realizar um gráfico, que pode ser verificado na Figura 22, com as tabelas que contém mais registros, e essas tabelas serão a base dos testes.

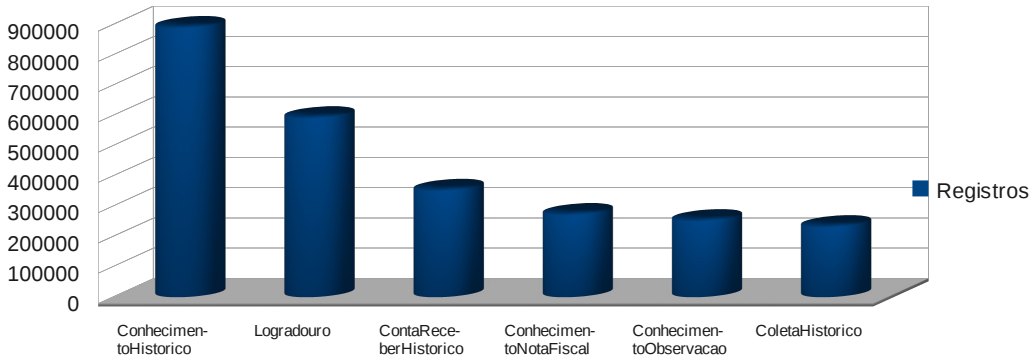


Figura 22: Tabelas com os maiores números de registros no banco de dados.

Cada consulta SQL será executada primeiramente utilizando apenas s*elect * from nome_da_tabela*, depois será feita novamente a mesma consulta porém utilizando a cláusula *order by*, a qual ordena o resultado da consulta, sempre como base será utilizada o campo texto de maior tamanho dentro da tabela.

As ferramentas utilizadas foram nativas dos SGDBs, para o SQL Server, foi utilizada a sua ferramenta *Microsoft SQL Server Management Studio* versão 10.0.1600.22. Para o PostgreSQL a ferramenta utilizada foi o *PgAdmin III* versão no Linux, visto que a documentação do PostgreSQL 8.2 diz que seu desempenho é melhor nesse sistema operacional. Para o Firebird foi utilizada a ferramenta de licença grátis *FlameRobin 0.8.6.1652 Unicode for Linux* essa instalada no Ubuntu 8.10. Com a ajuda de cada ferramenta de administração de ambos os SGDBs, temos resultados prontos após cada consulta.

A primeira tabela consultada em ambos os bancos de dados foi a maior de todas: ConhecimentoHistorico, após a tabela de Conhecimento e ContaReceberHistorico. A tabela de Conhecimento é uma tabela que tem em média 200.000 registros, não

aparecendo no gráfico da Figura 22, porém por ser uma tabela muito importante, merece um tratamento especial, em visto que é a tabela mais importante do sistema.

Na próxima seção é escrito como foi gerado o retorno dos testes em ambas as ferramentas, descrevendo cada item resultado pela ferramenta utilizada. Após, na seção 6.1.2 a 6.1.4 é apresentado a execução em três tabelas do sistema: ConhecimentoHistorico, Conhecimento e ContaReceberHistorico. E por fim, um relatório com os resultados atingidos.

6.1.1 Explicação do Retorno do SGDB

O retorno da análise realizado pelo Microsoft SQL Server 2008, é descrito conforme sua documentação *online* da seguinte forma (MSDN, 2009):

- Linhas afetadas pela Inserir, Excluir e Atualizar instruções: O número de linhas que foram afetadas por instruções Inserir, Excluir ou Atualizar que foram executadas como parte de sua consulta.

- Linhas recuperadas por instruções Inserir, Excluir e Atualizar: O número de linhas que foram recuperados por instruções Inserir, Excluir ou Atualizar que foram executadas como resultado de sua consulta.

- Número de instruções Selecionar: O número de instruções Selecionar que foram executadas através da conexão como parte de sua execução da consulta. Este número inclui instruções de FETCH para recuperar linhas de cursores.

- Linhas retornadas por instruções Selecionar: O número de linhas que foram Selecionadas como parte de sua execução da consulta. Esse número reflete todas as linhas geradas por instruções SQL, mesmo aqueles que não foram realmente consumidos pelo chamador (se, por exemplo, você pode cancelar execução).Esse número também inclui instruções de FETCH para recuperar linhas de cursores.

- Número de transações: O número de transações do usuário que foram iniciados como parte de sua execução da consulta, incluindo reversões.

- Buffers recebidos de o servidor: O número de dados tabulares fluxo de TDS (pacotes recebidos pelo cliente do servidor de banco de dados durante a execução da consulta.

- Pacotes de protocolo TDS enviados pelo cliente: O número de pacotes de protocolo TDS que o cliente enviado ao servidor de banco de dados durante a execução da consulta. Comandos grandes podem exigir Múltiplo buffers. Por exemplo, se um comando grande é enviado ao servidor e requer seis pacotes, o número de percursos circulares de servidor é incrementado por um, e o número de pacotes TDS que o cliente enviou é incrementado por seis.

- Pacotes de protocolo TDS enviados pelo servidor: O número de pacotes de protocolo TDS que o servidor enviado ao cliente.

- Bytes enviados de cliente: O número de *bytes* que o cliente enviado para SQL Servidor durante a execução da consulta.

- Número de percursos circulares de servidor: O número de vezes que a conexão enviada comandos para o servidor e recebido uma Responder como parte da execução da consulta.

- Tempo no servidor responde de espera: A quantidade cumulativa de tempo (em milissegundos) que o cliente gasto enquanto ele esperou o servidor para responder.

- Tempo de execução total: O cumulativa de tempo (em milissegundos) que o cliente gastou processar enquanto a consulta foi executada, incluindo o tempo que o cliente gasto aguardando respostas de servidor, bem como o tempo gasto executando o código.

- Tempo de processamento do cliente: A quantidade cumulativa de tempo que o cliente gastam executando o código enquanto a consulta foi executada.

O PostgreSQL concebe um plano de consulta para cada consulta solicitada. A escolha do plano correto, correspondendo à estrutura da consulta e às propriedades dos

dados, é absolutamente crítico para o bom desempenho. Para toda consulta pode ser utilizado o comando EXPLAIN, para ver o plano criado pelo sistema. O retorno realizado pelo PgAdminIII para o PostgreSQL, é descrito por sua documentação online da seguinte forma:

- Custo de partida estimado: O tempo gasto antes de poder começar a varrer a saída como, por exemplo, o tempo para fazer a classificação em um nó de classificação.

- Custo total estimado: Se todas as linhas fossem buscadas, oque pode não acontecer: uma consulta contendo a cláusula LIMIT para antes de gastar o custo total, por exemplo.

- Número de linhas de saída estimado para este nó do plano: Somente se for executado até o fim.

- Largura média estimada das linhas de saída deste nó do plano: Em *bytes*.

6.1.2 Tabela ConhecimentoHistorico

A tabela ConhecimentoHistorico guarda o histórico completo do conhecimento de carga, o qual é a entrega que a transportadora faz para seus clientes. Atualmente se encontra com mais de 888000 registros, na Tabela 26 mostra exemplo de possíveis dados da tabela.

Tabela 26 : Exemplo de dados obtidos na tabela ConhecimentoHistorico.

	Filial...	Controle	Seq...	Historico	Interno
1	01	000000000001	001	09/04/2007 07:42:51 - Conhecimento aguardando ...	NAO
2	01	000000000001	002	10/04/2007 16:58:25 - Conhecimento em entrega ...	NAO
3	01	000000000001	003	10/04/2007 16:58:55 - Conhecimento em entrega ...	NAO
4	01	000000000001	004	10/04/2007 16:59:44 - Conhecimento em entrega ...	NAO
5	01	000000000002	001	09/04/2007 07:56:22 - Conhecimento aguardando ...	NAO

Na Tabela 27 e Tabela 28, segue análise de retorno da seleção na tabela ConhecimentoHistorico. Primeiramente no SQL Server.

Tabela 27. Análise da tabela ConhecimentoHistorico no Microsoft SQLServer.

	Avaliação 2	Avaliação 1	Média
Tempo de Execução do Cliente	17:10:09	17:09:17	
Estatísticas do Perfil da Consulta			
Número de instruções INSERT, DELETE e UPDATE	0 →	0 →	0.0000
Linhas afetadas pelas instruções INSERT, DELETE ou U...	0 →	0 →	0.0000
Número de instruções SELECT	2 →	2 →	2.0000
Linhas retornadas pelas instruções SELECT	888339 →	888339 →	888339.0000
Número de transações	0 →	0 →	0.0000
Estatísticas de Rede			
Numero de viagens de ida e volta ao servidor	3 →	3 →	3.0000
Pacotes TDS enviados do cliente	3 →	3 →	3.0000
Pacotes TDS recebidos do servidor	23743 ↑	23742 →	23742.5000
Bytes enviados do cliente	286 ↑	248 →	267.0000
Bytes recebidos do servidor	9.724156E... ↑	9.723903E... →	97240300.0000
Estatísticas de Tempo			
Tempo de processamento do cliente	6763 ↓	6958 →	6860.5000
Tempo total de execução	12248 ↑	6974 →	9611.0000
Tempo de espera em respostas do servidor	5485 ↑	16 →	2750.5000

Na Tabela 28 a análise da tabela ConhecimentoHistórico no PostgreSQL.

Tabela 28. Análise da tabela ConhecimentoHistórico no PostGreSQL.

Custo	**Colunas**	**Tam.**	**Tempo**	**Observação**
Primeira avaliação, com a cláusula SQL *order by*:				
0.00..23501.38	888338	111	2615.142 ms	Seq Scan on conhecimentohistorico
266126.50 268347.35	888338	111	39306.262.. 47025.064	Sort
Segunda avaliação, sem a cláusula SQL *order by*:				
0.00..23501.38	888338	111	48261.639 ms	Seq Scan on conhecimentohistorico
Terceira avaliação, com a cláusula SQL *order by* e com um índice criado				
0.00..26036.38	888338	127	363062.50.. 365283.35	Índice criado.

E por fim, na Figura 23 mostra-se a comparação de medida de tempo de processamento entre os dois SGDBs, onde mostra a velocidade do PostgreSQL fazendo

uma simples consulta ao banco de dados, mas também mostra a sua larga demora ao realizar a mesma consulta, porém com uma ordenação definida.

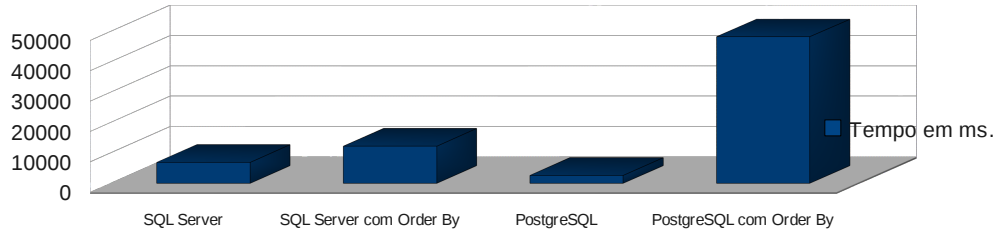


Figura 23 : Comparação entre os dois SGDBs.

6.1.3 Tabela Conhecimento

A tabela Conhecimento é a tabela com mais campos em todo os banco de dados, e ela armazena todos os dados do conhecimento de carga, o qual é a entrega que a transportadora faz para seus clientes. Atualmente se encontra com mais de 211000 registros, na Tabela 29 mostra exemplo de possíveis dados da tabela.

Tabela 29 : Exemplo de dados na tabela de conhecimento.

	Filial...	Controle	Servico	Empresa	FilialOrigem	FilialDestino	Serie...	Conhecimento
1	01	000000000001	01	01	01	01	UNI	00063087
2	01	000000000002	01	01	01	01	UNI	00063088
3	01	000000000003	01	01	01	04	UNI	00063090
4	01	000000000004	01	01	01	01	UNI	00063089
5	01	000000000005	01	01	01	01	UNI	00063091

Na Tabela 30 e Tabela 31, segue análise de retorno da seleção na tabela Conhecimento. Primeiramente no SQL Server. Na Tabela 31 a análise da tabela Conhecimento no PostgreSQL.

Tabela 30. Análise da tabela Conhecimento no Microsoft SQLServer.

	Avaliação 2	Avaliação 1	Média
Tempo de Execução do Cliente	17:21:52	17:19:12	
Estatísticas do Perfil da Consulta			
Número de instruções INSERT, DELETE e UPDATE	0 →	0 →	0.0000
Linhas afetadas pelas instruções INSERT, DELETE ou U...	0 →	0 →	0.0000
Número de instruções SELECT	2 →	2 →	2.0000
Linhas retomadas pelas instruções SELECT	211020 →	211020 →	211020.0000
Número de transações	0 →	0 →	0.0000
Estatísticas de Rede			
Numero de viagens de ida e volta ao servidor	3 →	3 →	3.0000
Pacotes TDS enviados do cliente	3 →	3 →	3.0000
Pacotes TDS recebidos do servidor	35145 ↑	35139 →	35142.0000
Bytes enviados do cliente	312 ↑	230 →	271.0000
Bytes recebidos do servidor	1.439423E... ↑	1.439196E... →	143930900.0000
Estatísticas de Tempo			
Tempo de processamento do cliente	27887 ↑	11811 →	19849.0000
Tempo total de execução	46870 ↑	11812 →	29341.0000
Tempo de espera em respostas do servidor	18983 ↑	1 →	9492.0000

Tabela 31. Análise da tabela Conhecimento no PostgreSQL.

Custo	**Colunas**	**Tam.**	**Tempo**	**Observação**
Primeira avaliação, com a cláusula SQL *order by*:				
0.00..18310.19	211019	980	2204.419 ms	Seq Scan on conhecimento
308533.17.. 309060.72	211019	980	17078.330.. 21308.947	Sort Disk: 120512kB
Primeira avaliação, sem a cláusula SQL *order by*:				
0.00..18310.19	211019	980	21643.276 ms	Seq Scan on conhecimento
Terceira avaliação, com a cláusula SQL *order by* e com um índice criado				
0.00..22873.19	211019	1106	475979.67.. 476507.22	Índice criado.

E por fim, na Figura 24 mostra-se a comparação de tempo entre os dois SGDBs, onde mostra uma demora muito maior do Microsoft SQL Server, em comparação a tabela anterior, ConhecimentoHistorico, isso devida a estrutura das tabelas, onde na tabela de Conhecimento muitos tipos de dados estão presentes.

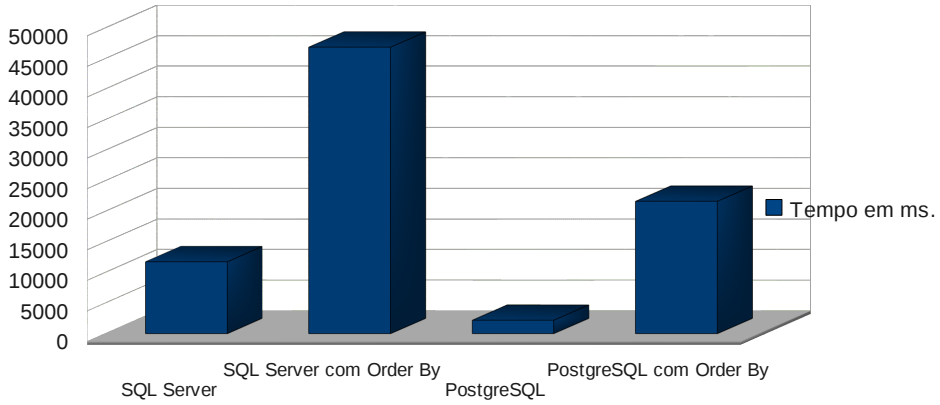


Figura 24 : Comparação entre os dois SGDBs.

6.1.4 Tabela ContaReceberHistorico

A tabela ContaReceberHistorico é a tabela que armazena os dados referente as faturas que a transportadora emite para seus clientes. Atualmente se encontra com mais de 888000 registros, na Tabela 32 mostra exemplo de possíveis dados da tabela.

Tabela 32 : Exemplo de dados na tabela de ContaReceberHistorico.

Em...	Fatura	Serie...	Linha	Observacao	Sistema	UltimaAlteracao
	00000001	F01	001	Retorno Banco ENTRADA CONFIRMADA ...	SIM	2009-08-31 ...
	00000002	F01	001	EM 08/09/2009 17:28:47 FOI ALTERADO ...	SIM	2009-09-08 ...
	00000003	F01	001	EM 08/09/2009 17:37:48 FOI ALTERADO ...	SIM	2009-09-08 ...
	00000004	F01	001	EM 09/09/2009 08:13:51 FOI ALTERADO ...	SIM	2009-09-08 ...
	00000005	F01	001	EM 09/09/2009 08:13:30 FOI ALTERADO ...	SIM	2009-09-08 ...

Na Tabela 33 e Tabela 34, segue análise de retorno da seleção na tabela ContaReceberHistorico. Primeiramente no SQL Server. Na Tabela 34 a análise da tabela ContaReceberHistorico no PostgreSQL.

Tabela 33. Análise da tabela ContaReceberHistorico no Microsoft SQLServer.

	Avaliação 2	Avaliação 1	Média
Tempo de Execução do Cliente	18:36:45	18:34:36	
Estatísticas do Perfil da Consulta			
Número de instruções INSERT, DELETE e UPDATE	0 →	0 →	0.0000
Linhas afetadas pelas instruções INSERT, DELETE ou U...	0 →	0 →	0.0000
Número de instruções SELECT	2 →	2 →	2.0000
Linhas retornadas pelas instruções SELECT	342293 →	342293 →	342293.0000
Número de transações	0 →	0 →	0.0000
Estatísticas de Rede			
Numero de viagens de ida e volta ao servidor	3 →	3 →	3.0000
Pacotes TDS enviados do cliente	3 →	3 →	3.0000
Pacotes TDS recebidos do servidor	9771 ↑	9770 →	9770.5000
Bytes enviados do cliente	288 ↑	246 →	267.0000
Bytes recebidos do servidor	4.001196E... ↑	4.0009E+07 →	40010480.0000
Estatísticas de Tempo			
Tempo de processamento do cliente	2933 ↓	17109 →	10021.0000
Tempo total de execução	7173 ↓	17566 →	12369.5000
Tempo de espera em respostas do servidor	4240 ↑	457 →	2348.5000

Tabela 34 . Análise da tabela ContaReceberHistorico no PostgreSQL.

Custo	**Colunas**	**Tam.**	**Tempo**	**Observação**
Primeira avaliação, com a cláusula SQL *order by*:				
0.00..9253.44	342144	108	1426.581 ms	Seq Scan on contareceberhis.
79301.73.. 80157.09	342144	108	17078.330.. 21308.947	Disk:40800kB
Primeira avaliação, sem a cláusula SQL *order by*:				
0.00..9253.44	342144	108	32470.244 ms	Seq Scan on contareceberhistorico
Terceira avaliação, com a cláusula SQL *order by* e com um índice criado				
0.00.. 10427.93	342144	126	104082.89.. 104938.62	Índice criado.

E por fim, na Figura 25 mostra-se a comparação de tempo entre os dois SGDBs, onde da mesma forma que a tabela de Conhecimeto, o Microsoft SQL Server teve uma demora no acesso aos dados.

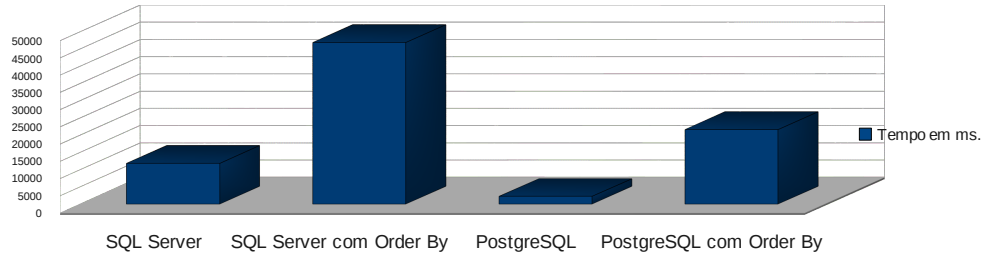


Figura 25 : Comparação entre os dois SGDBs.

6.1.5 Relatórios

Para confirmar a tese de Pressman (2007) onde o melhor teste de desempenho é o executado utilizando o sistema em tempo real, foi retirado relatórios e realizado consultas no sistema onde mostram a diferença de tempo de retorno. Todos os testes foram realizados em rede, visto que a segunda camada da aplicação servidora, não roda no sistema operacional Linux, por se tratar da utilização de DLLs.

A Figura 26 mostra o gráfico de relatórios utilizado no sistema e o tempo de retorno em ambos os SGDBs, onde mostra em que em todos o sistema se comportou melhor com o SGDB Microsoft SQL Server. Possível causa essa pela segunda camada do sistema também rodar na plataforma Microsoft Windows. Ficando como trabalhos futuros, a migração da segunda camada do sistema para Linux, tentando verificar assim se o acesso utilizando o PostgreSQL no sistema operacional Linux será mais rápido que o seu concorrente Microsoft SQL Server, rodando no sistema operacional Microsoft Windows.

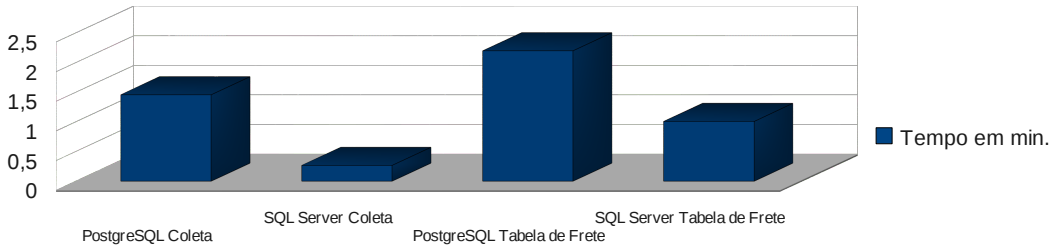


Figura 26: Gráfico de tempo em segundos de dois relatórios executados pelo sistema SistranNet.

6.2 Testes de Sistema

Na empresa, caso haja inconsistências no *software*, o analista de sistemas é acionado e volta ao ciclo de vida do teste. O produto é encaminhado novamente à fase de implementação para que sejam executadas as correções necessárias. Havendo aprovação dos testes, a versão é liberada e publicada no ambiente de produção. Este trabalho apresenta a automação de testes funcionais para componentes de *software*. A importância de iniciar os teste funcionais cada vez mais cedo, para minimizar o impacto da descoberta de erros em fases mais avançadas de projeto. Este trabalho teve como objetivo possibilitar a verificação das propriedades dos componentes e empacotar os artefatos e os resultados dos testes

Para garantir a execução correta do sistema, foi criado um caso de teste que pode ser verificado na Tabela 35, onde tem como funcionalidade descrever eventuais exceções que venham a acontecer durante a execução desse trabalho.

Tabela 35: Caso de Teste referente ao teste de Sistema.

Caso de Teste:	Teste de Sistema
Objetivo:	Validar a todo o sistema baseado em computador.
Descrição breve:	O propósito desse caso de teste é descrever e validar as atividades envolvidas nos testes de todo o círculo do software em si, toda a estrutura, componentes e elementos que interagem com o mesmo.
Condições de Execução:	Utilizar a ferramenta que execute a ação de verificação do sistema em sí. *The Delphi Unit Test Expert.*
Pré-condições:	Banco de dados com a estrutura correta do PostgreSQL concluída, dados migrados e alterações no SistranNet efetuadas.
Pontos de Observações	Observar firewall e acesso externo a rede do servidor.
Resultados esperados:	Dados para a documentação dos testes.

Um dos primeiros problemas encontrados foi a de localizar um *driver* de acesso *free* para o SGDB PostgreSQL, em Delphi. Existem apenas duas opções utilizando o dbExpress: VitaVoom ou DBX, ambos ferramentas pagas, que possuem uma opção free, porém com limitações.

Após a decisão de usar o *driver* da VitaVoom por já conhecimento da ferramenta, outro problema foi em relação do banco de dados do PostgreSQL no Ubuntu Linux. O problema aconteceu no idioma dos dois *softwares*. A documentação

do PostgreSQL diz que para o português Brasil deve-se utilizar o *collate* Latin1, porém, a restauração do *backup* do Postgresql instalado no Microsoft Windows só funcionou corretamente após ter sido criado o banco de dados com o *collate* SQL_ASCII. Mesmo assim, ao abrir o banco de dados pela ferramenta PGAdmin III, ele exibe um alerta sobre o *encoding* do banco de dados, veja na mensagem abaixo:

Database encoding

The database SistranNet is created to store data using the SQL_ASCII encoding. This encoding is defined for 7 bit characters only; the meaning of characters with the 8th bit set (non-ASCII characters 127-255) is not defined. Consequently, it is not possible for the server to convert the data to other encodings.

If you're storing non-ASCII data in the database, you're strongly encouraged to use a proper database encoding representing your locale character set to take benefit from the automatic conversion to different client encodings when needed. If you store non-ASCII data in an SQL_ASCII database, you may encounter weird characters written to or read from the database, caused by code conversion problems. This may cause you a lot of headache when accessing the database using different client programs and drivers.

For most installations, Unicode (UTF8) encoding will provide the most flexible capabilities.

Quanto ao acesso externo e *firewall*, para que o sistema possa funcionar, a porta 211 deve estar liberada, só assim o sistema funciona. Já as portas 5432 do PostgreSQL e 1433 do SQL Server, podem estar bloqueadas, pois o acesso ao sistema funciona apenas pelo *Borland Socket Server*, que gerencia a segunda camada do aplicativo. Sendo assim, a única forma de conseguir acessar o sistema externamente, é tendo a senha de algum usuário.

Na seção 6.5, novos testes são realizados e que também agregam ao teste de sistema, como por exemplo testes de rede.

6.3 Testes de Validação

Como o sistema é muito grande em número de arquivos de fontes, é quase impossível verificar todas as funções e procedimentos do sistema, tornando assim um trabalho lento e passível a erros, para isso foi novamente utilizado a ferramenta *DUnit* do ambiente de desenvolvimento *Embarcadero RAD Studio 2009*. Os detalhes do caso de teste de validação pode ser visto na Tabela 36.

Tabela 36 : Caso de Teste referente ao teste de Validação.

Caso de Teste:	Teste de Validação
Objetivo Atingido:	Validado o sistema de forma que funcione de maneira correta nos novos SGDBs.
Descrição breve:	O propósito desse caso de teste é descrever e validar as atividades envolvidas na verificação dos requisitos funcionais do sistema, se

	estão corretos e funcionarão de maneira esperada.
Condições de Execução:	Utilizar a ferramenta que execute a ação de verificação dos requisitos do sistema. DUnit.
Pré-condições:	Banco de dados com a estrutura correta do PostgreSQL econcluídas, dados migrados e alterações no SistranNet efetuadas.
Pontos de Observações	Observar a situação do sistema em um dia de produção normal no cliente.
Resultados esperados:	Dados para a documentação dos testes.

A execução dos testes pode ser vista na Figura 27, onde mostra que os quatros testes executados, sendo o primeiro a parte de passagem dos parâmetros. Ferramenta muito importante, pois ajudou bastante nos teste para a validação do sistema, principalmente na parte servidora da aplicação, onde esses testes foram executados.

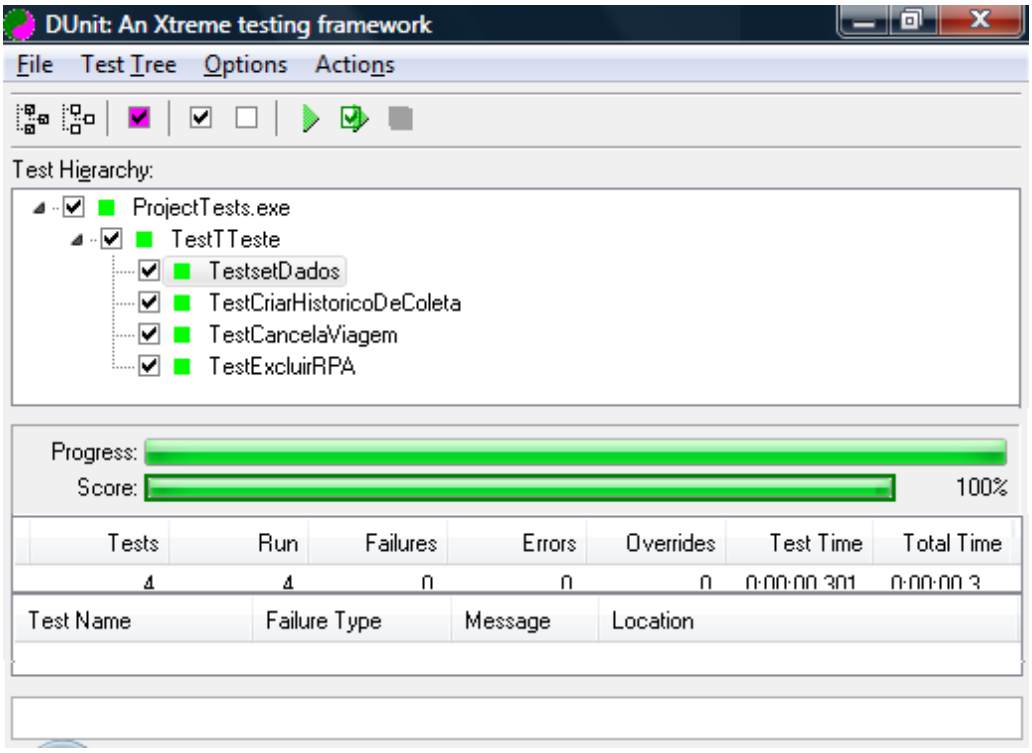


Figura 27: Execução da ferramenta DUnit.

Como mostra a Figura 27, a execução dos testes foram concluídas sem problemas, ficando assim como trabalhos futuros, aprofundar a técnica de testes utilizando a ferramenta, pois é possível também testar a ordem de entrada dos campos, testes em classes separadas ao sistema, etc.

6.4 Testes de Estresse

O teste de estresse, pode ser feito após o teste de validação estar concluído. A Tabela 37 demonstra o caso de teste para esse processo, que será melhor descrito ao decorrer desse subcapítulo.

Tabela 37: Caso de Teste referente ao teste de Estresse.

Caso de Teste:	Teste de Estresse
Objetivo Atingido:	Validado o sistema verificando circunstâncias que podem acontecer com uma combinação inesperada de eventos.
Descrição breve:	O propósito desse caso de teste é descrever e validar as atividades envolvidas no confronto do sistema com situações anormais, verificando que o sistema não cause perdas e danos em vez de entrar em colapso.
Condições de Execução:	Utilizar a ferramenta que execute a ação de estressar o sistema. DUnit.
Pré-condições:	Banco de dados com a estrutura correta do PostgreSQL concluídas, dados migrados e alterações no SistranNet efetuadas.
Pontos de Observações	Observar possíveis falhas e registros que poderão ser perdidos com o teste.
Resultados esperados:	Dados para a documentação dos testes.

O teste de estresse foi divido em duas formas, ambos executando diversas alterações tanto no banco de dados como na segunda camada da aplicação, a camada servidora. Todos os valores trafegados para realizar essas modificações nos registros, foram gerados sequencialmente, através da função do Delphi *Random* (valores inteiros) e a função *GeraTexto*, que foi criada e pode ser vista na Figura 28.

```
30  function GeraTexto : String;
31  var
32    i: Byte;
33    s: string;
34    intMAX_PW_LEN : integer;
35  begin
36    s := 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789';
37    intMAX_PW_LEN := Random(999);
38
39    Result := '';
40    for i := 0 to intMAX_PW_LEN-1 do
41      Result := Result + s[Random(Length(s)-1)+1];
42  end;
```

Figura 28 Função criada para gerar *strings* randômicas.

De forma a exemplificar o uso da ferramenta de teste unitário, é realizado um estudo de caso utilizando o ambiente de desenvolvimento Embarcadero RAD Studio 2009. Para isso foi criado uma classe chama TTeste com atributos de parâmetros e também métodos que serão executados pela ferramenta e retornando o sucesso ou a mensagem de erro.

Esta classe possui um método chamado *setDados*, responsável por atribuir os valores dos atributos da classe, respeitando assim o princípio de encapsulamento. Sempre que for necessário executar algum teste, seta-se os valores e espera o retorno da execução.

Na Figura 29 a 31 apresenta-se a codificação da *unit UTeste*, mostrando a classe criada, depois disso apenas executar a ferramenta *uTeste* e verificar o retorno, verificando a validação do sistema utilizando o novo SGDB.

```
unit UTeste;

interface

uses
  DBClient, SConnect;

type
   TTeste = class
   private
      p1,p2,p3,p4 : String;
   public
      procedure setDados(pp1,pp2,pp3,pp4 : String);
      function CriarHistoricoDeColeta : boolean;
      function CancelaViagem : boolean;
      function ExcluirRPA : boolean;
      constructor Create;
   end;

var
     ConnectionBroker: TConnectionBroker;
     SocketConnection: TSocketConnection;

implementation

{ TTeste }

constructor TTeste.Create;
begin
   Inherited Create;
end;

function TTeste.CancelaViagem: boolean;
begin
```

Figura 29 : Primeira parte da codificação da unit UTeste.

Na Figura 29, pode ser visto a primeira parte do código fonte da ferramenta. Nela está declarado as variáveis que irão fazer a conexão com o banco de dados e também o cabeçalho das funções que servirão como base dos testes realizados.

A Figura 30 e 31 mostram a execução de cada função da ferramenta, todas elas tem as suas primeiras dez linhas iguais, onde criam a conexão com o banco de dados. É passado cada parâmetro ao componente de conexão:

- ServerGuid: Endereço da conexão dentro do computador.
- ServerName: Nome do servidor da segunda camada que será conectado.
- Host: Endereço do computador que vai ser conectado.
- Port: Número da porta que será conectado.

```
function TTeste.CancelaViagem: boolean;
begin
 SocketConnection := TSocketConnection.create(nil);
 SocketConnection.Connected := False;
 SocketConnection.ServerGUID := '{0C82B89A-3584-42F3-8CE2-1BFE7777606D}';
 SocketConnection.ServerName := 'ServidorOperacional.Server_Operacional';
 SocketConnection.Host := 'localhost';
 SocketConnection.Port := 211;
 SocketConnection.Connected := True;
 ConnectionBroker := TConnectionBroker.create(nil);
 ConnectionBroker.Connection := SocketConnection;
 try
    ConnectionBroker.AppServer.CancelarViagem(self.p1,self.p2);
    result := true;
 except result := False; end;
end;

function TTeste.CriarHistoricoDeColeta: boolean;
begin
 SocketConnection := TSocketConnection.create(nil);
 SocketConnection.Connected := False;
 SocketConnection.ServerGUID := '{A3E28CFD-1BC4-489A-B288-4B8B685DC9F9}';
 SocketConnection.ServerName := 'ServidorColeta.Server_Coleta';
 SocketConnection.Host := 'localhost';
 SocketConnection.Port := 211;
 SocketConnection.Connected := True;
 ConnectionBroker := TConnectionBroker.create(nil);
 ConnectionBroker.Connection := SocketConnection;
 try
    ConnectionBroker.AppServer.CriarHistoricoDeColeta(self.p1,self.p2,
    self.p3,self.p4);
    result := true;
 except result := False;  end;
 end;
```

Figura 30: Segunda parte da codificação da unit UTeste.

Após ser criada a conexão é enviado para o servidor o nome da função e os parâmetros da execução. Como os valores são randômicos, pode muitas vezes obter sucesso, como também falha na execução.

```
procedure TTeste.setDados(pp1, pp2, pp3, pp4: String);
begin
   self.p1 := pp1;
   self.p2 := pp2;
   self.p3 := pp3;
   self.p4 := pp4;
end;

function TTeste.ExcluirRPA: boolean;
begin
     SocketConnection := TSocketConnection.create(nil);
     SocketConnection.Connected := False;
     SocketConnection.ServerGUID :=
     '{0C82B89A-3584-42F3-8CE2-1BFE7777606D}';
     SocketConnection.ServerName :=
     'ServidorOperacional.Server_Operacional';
     SocketConnection.Host := 'localhost';
     SocketConnection.Port := 211;
     SocketConnection.Connected := True;

     ConnectionBroker := TConnectionBroker.create(nil);
     ConnectionBroker.Connection := SocketConnection;

     try
        ConnectionBroker.AppServer.ExcluirRPA(self.p1,self.p2);
        result := true;
     except
        result := False;
     end;
end;

end.
```

Figura 31: Última parte da codificação da *unit* Uteste.

Nas próximas seções, é descrito a execução do teste de estresse de duas formas, a primeira é executando manualmente os testes e na segunda, o teste é executado de forma paralela através de *threads*.

6.4.1 Programas Separados

O primeiro teste foi realizando com 10000 processos sequências, dentre esses processos incluí-se inserções, deleções e alterações nos registros do banco de dados. A

lógica do programa, é executar essa sequência de operações e gravar um histórico para que possa ser analisado esse retorno (Figura 32).

```
44  procedure TForm1.Button1Click(Sender: TObject);
45  var
46   T : TTeste;
47   i : integer;
48  begin
49   application.ProcessMessages;
50   T := TTeste.create;
51   for I := 0 to 1000000 do
52   begin
53  ⟶   if t.criarhistoricodecoleta(formatfloat('00',random(5) + 1),
54  ⟶   ⟶formatfloat('000000000000',random(999999999999) + 1),'WILIAN',
55  ⟶   ⟶geratexto) then
56  ⟶   ⟶   memo1.lines.add(formatfloat('000000000000',i) + ' ok')
57  ⟶   else
58  ⟶   ⟶   memo1.lines.add(formatfloat('000000000000',i) + ' erro');
59  ⟶   if T.CancelaViagem(FormatFloat('00',Random(5) + 1),
60  ⟶   ⟶   FormatFloat('000000000000',Random(999999999999) + 1)) then
61  ⟶   ⟶   memo1.lines.add(formatfloat('000000000000',i) + ' ok')
62  ⟶   else
63  ⟶   ⟶   memo1.lines.add(formatfloat('000000000000',i) + ' erro');
64  ⟶   if T.ExcluirRPA(FormatFloat('00',Random(5) + 1),
65  ⟶   ⟶   FormatFloat('000000000000',Random(999999999999) + 1)) then
66  ⟶   ⟶   memo1.lines.add(formatfloat('000000000000',i) + ' ok')
67  ⟶   else
68  ⟶   ⟶   memo1.lines.add(formatfloat('000000000000',i) + ' erro');
69  ⟶   application.processmessages;
70   end;
71  end;
```

Figura 32: Código fonte do teste de estresse sequencial.

Analisando o código fonte, nota-se o uso constante da função r*andom*. Na primeira execução, na parte de criar histórico de coleta, ela é usada para gerar o valor para o primeiro parâmetro que é a filial, gerando valores de 1 a 5, pois no banco de dados de teste só existiam 5 filiais cadastradas. O segundo parâmetro refere-se ao número de controle da Coleta, o qual é gerado um número aleatório de 12 caracteres. O terceiro parâmetro o nome do usuário e por fim, o texto a ser gravado num campo *varchar* de 100 caracteres, gerado aleatoriamente pela função GeraTexto.

Importante salientar, que quando o sistema foi executado pela terceira vez, todos travaram, devido ao acesso concorrente, ficando como projeto futuro, a melhoria nos acessos concorrentes do sistema.

6.4.2 Threads

O segundo teste de estressar o sistema foi realizado através de *threads*. As quais foram executadas conforme a Figura 33, a qual demonstra a sua execução da mesma forma que o a seção anterior, porém, como é executada paralelamente, é necessário executar a função *Synchronize*, a qual sincroniza a execução com o formulário que contém barras de progressos e campos de textos.

```
36  procedure TThreadTeste.Execute;
37  var
38  T : TTeste;
39  i : integer;
40  begin
41  Priority := tpLower;
42  showmessage(Progress.Name);
43  T := TTeste.create;
44  for I := 0 to 1000000 do
45  begin
46    Synchronize(incrementa);
47    if t.criarhistoricodecoleta(formatfloat('00',random(5) + 1),
48    ⟶formatfloat('000000000000',random(999999999999) + 1), 'WILIAN',
49    ⟶Form1.geratexto) then
50    ⟶  Form1.memo1.lines.add(formatfloat('000000000000',i) + ' ok')
51    else
52    ⟶  Form1.memo1.lines.add(formatfloat('000000000000',i) + ' erro');
53    if T.CancelaViagem(FormatFloat('00',Random(5) + 1),
54    ⟶⟶FormatFloat('000000000000',Random(999999999999) + 1)) then
55    ⟶  Form1.memo1.lines.add(formatfloat('000000000000',i) + ' ok')
56    else
57    ⟶  Form1.memo1.lines.add(formatfloat('000000000000',i) + ' erro');
58    if T.ExcluirRPA(FormatFloat('00',Random(5) + 1),
59    ⟶⟶FormatFloat('000000000000',Random(999999999999) + 1)) then
60    ⟶  Form1.memo1.lines.add(formatfloat('000000000000',i) + ' ok')
61    else
62    ⟶  Form1.memo1.lines.add(formatfloat('000000000000',i) + ' erro');
63    application.processmessages;
64  end;
65  end;
```

Figura 33: Código fonte da execução da *thread*,

O resultado dessa execução, pode ser visto na Figura 34, onde demonstra a execução paralelamente de duas *threads*, a qual foi onde o sistema executou corretamente, a partir da terceira execução, ocorreu travamento, da mesma forma que a execução de programas separados, porém com as mesmas funções, vistos no item 5.8.1.

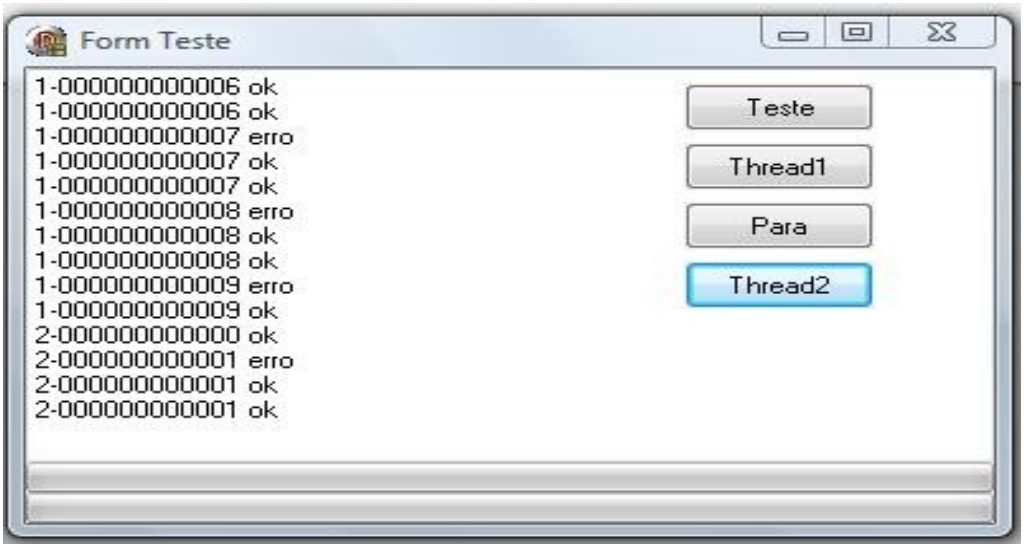


Figura 34 : Resultado execução *thread*.

Os resultados vistos na Figura 34, “erro” ou “ok” devem-se ao fato de valores randômicos terem sido enviados ao servidor pela função, assim, por exemplo, se algum registro não é encontrado, retorna a constante “erro” como negação, mas caso consiga executar a função, seja por exemplo uma inserção, retorna o “ok” como sucesso da execução.

Como conclusão do teste de estresse, pode-se dizer que o sistema precisa um bom tratamento de acessos concorrentes, uma dica seria a migração do Borland Delphi 7 para o RAD Studio 2009, onde a nova versão do D*ataSnap* foi reformulada e obteve várias novas funções que suprem essa necessidade do sistema.

6.5 Testes de Segurança

Os testes de segurança executados no sistema SistranNet foram variados, alguns com ferramentas e outros utilizando o próprio sistema. Na Tabela 38 pode-se verificar o caso de teste realizado para esse processo.

Tabela 38 : Caso de Teste referente ao teste de Segurança.

Caso de Teste:	Teste de Segurança
Objetivo Atingido:	Validado os mecanismos de segurança embutidos no sistema.
Descrição breve:	O propósito desse caso de teste é descrever e validar as atividades envolvidas no teste de segurança, deve-se evitar que acessos indevidos possam acessar principalmente os dados, os quais são as informações mais importante do software.

Condições de Execução:	Utilizar a ferramenta que execute a ação e verificação de segurança do sistema. Wireshark.
Pré-condições:	Banco de dados com a estrutura correta do PostgreSQL concluída, dados migrados e alterações no SistranNet efetuadas.
Pontos de Observações	Observar erros que possam a acontecer com quedas de energia, queda da rede, etc.
Resultados esperados:	Dados para a documentação dos testes.

A seguir serão descritos categorias de testes, sua execução e caso necessite, a sua solução. Esses testes foram divididos em quatro etapas, a primeira demonstra os testes realizados quanto ao acesso a rede, o próximo é uma validação e checagem quanto a consistência dos dados. As duas últimas etapas, representam o acesso aos dados, tanto de forma concorrente como de forma manual e monousuário.

6.5.1 Acesso e Rede

Foi realizado dois testes que obtiveram os mesmos resultados, o primeiro teste foi desligado o servidor e o segundo teste foi encerrado a segunda camada.

O primeiro teste foi realizado durante a execução de um relatório que demora em média 5 a 6 minutos, após uns 2 minutos de execução foi desligado bruscamente o servidor do banco de dados, no sistema o que aconteceu foi o seguinte erro: *[DBNETLIB][ConnectionWrite (send()).]General network error. Check your network documentation*. Após isso, o sistema teve que ser encerrado para que voltasse a funcionar, sem perda de dados.

O segundo teste foi durante a inserção de dados no cadastro de clientes, e antes de confirmar o novo cadastro, foi retirado o cabo de rede, ao clicar no botão salvar e gravar as alterações foi gerado um erro que pode ser visto na Figura 35. Como o sistema é controlado por transações, os dados não foram gravados, e, nem os registros detalhes foram gravados no banco, visto que para uma transação funcionar, todos os dados devem ser aplicados.

Figura 35: Erro ao salvar um novo registro com erro de conexão na rede.

6.5.2 Consistência do Banco de Dados

Para teste da consistência dos dados no novo SGDB, foi realizado um teste simples porém funcional, foi aberto diversos cadastros e telas de lançamentos e incluído registros que forçassem esse controle de integridade dos dados. Na Figura 36, pode-se verificar um desses testes, onde é cadastrado um novo cliente com uma cidade errada, sendo gerada a exceção no cadastro.

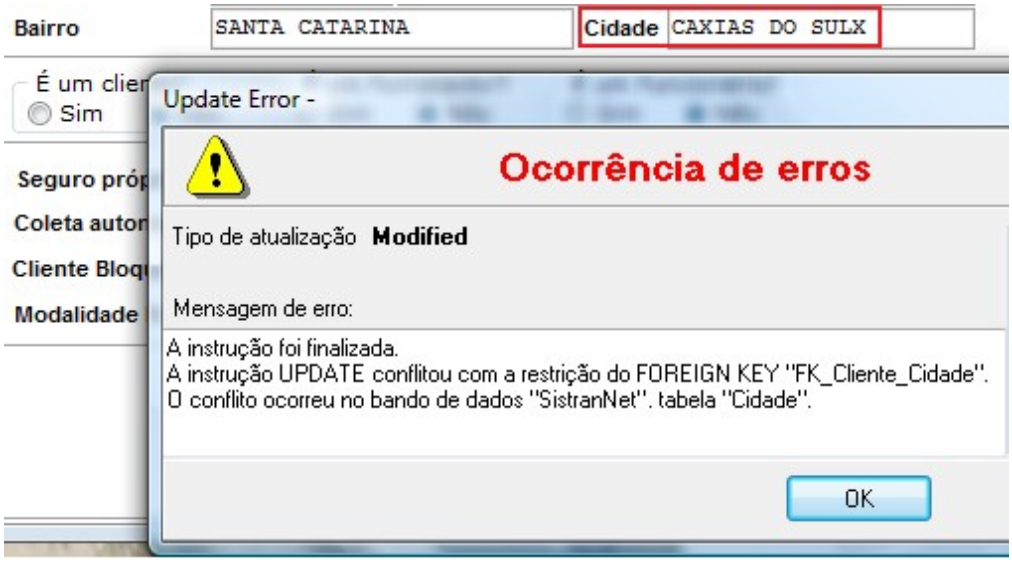


Figura 36 : Ocorrência de erro na consistência dos dados.

6.5.3 Acesso concorrente

A fim de verificar anomalias causadas por acesso concorrentemente aos dados, foi remetido relatórios com o mesmo propósito, porém em módulos diferentes, em alguns casos foi gerado até as mesmas SQLs, pois, alguns relatórios são acessados por módulos diferentes, exemplo módulo operacional e módulo gerencial. Os relatórios resultaram todos com os mesmos valores. Verificando controle transacional que o

componente de acesso ao banco de dados dbExpress possui.

6.5.4 Acesso aos Dados

Com a ajuda da ferramenta *free* WireShark versão 1.2.2 que pode ser baixada pelo site http://www.wireshark.org/download.html, foi possível analisar todos os pacotes de dados que trafegam entre o computador servidor e cliente com a execução do sistema.

Alguns pontos devem ser destacados, visto que esse sub-tópico refere-se a segurança das informações presente no sistema:

1. Todos os dados puderam ser interceptados pela ferramenta, ficando como futuros projetos a criptografia do tráfego das informações (Figura 37).

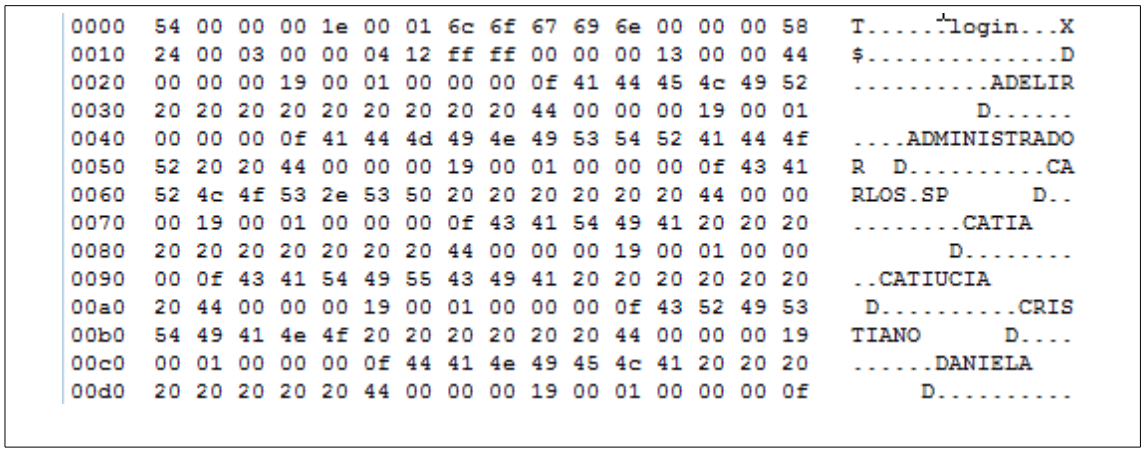


Figura 37 : Nome dos usuários que puderam ser interceptados pela ferramenta.

2. Além dos dados, as SQLs realizadas tanto para consulta como inserção também puderam ser analisadas.

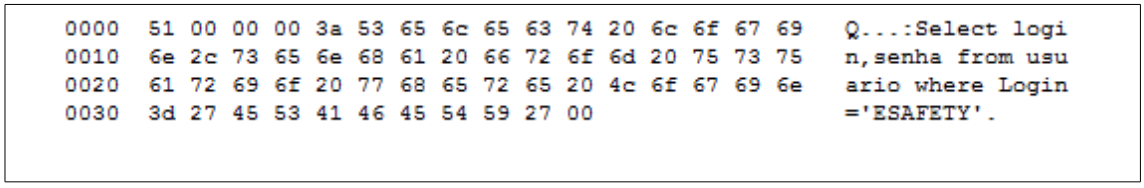
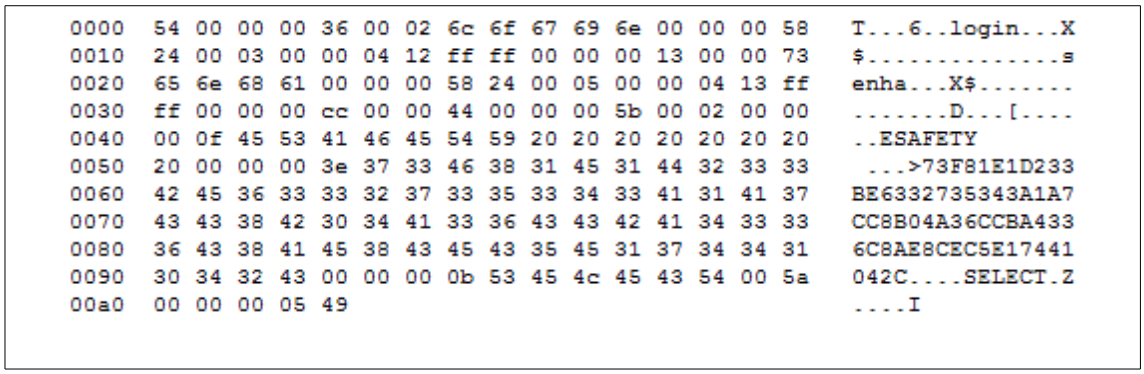


Figura 38 : Exemplo de SQL interceptada pela ferramenta.

3. Como os dados foram vistos, o teste crucial foi o teste das senhas, onde é necessário testar a senha do usuário. Com sorte o sistema faz a criptografia da senha automaticamente, nota-se isso na Figura 39.



```
0000  54 00 00 00 36 00 02 6c 6f 67 69 6e 00 00 00 58   T...6..login...X
0010  24 00 03 00 00 04 12 ff ff 00 00 00 13 00 00 73   $..............s
0020  65 6e 68 61 00 00 00 58 24 00 05 00 00 04 13 ff   enha...X$.......
0030  ff 00 00 00 cc 00 00 44 00 00 00 5b 00 02 00 00   .......D...[....
0040  00 0f 45 53 41 46 45 54 59 20 20 20 20 20 20 20   ..ESAFETY
0050  20 00 00 00 3e 37 33 46 38 31 45 31 44 32 33 33    ...>73F81E1D233
0060  42 45 36 33 33 32 37 33 35 33 34 33 41 31 41 37   BE6332735343A1A7
0070  43 43 38 42 30 34 41 33 36 43 43 42 41 34 33 33   CC8B04A36CCBA433
0080  36 43 38 41 45 38 43 45 43 35 45 31 37 34 34 31   6C8AE8CEC5E17441
0090  30 34 32 43 00 00 00 0b 53 45 4c 45 43 54 00 5a   042C....SELECT.Z
00a0  00 00 00 05 49                                    ....I
```

Figura 39 : Senha trafegando criptograficamente pela rede.

4. Um último teste realizado, foi capturando os pacotes no lado do servidor Linux, onde nele, uma segurança a mais foi detectada, nele não é possível verificar os valores contido nos pacotes, apenas o número de linhas e o tamanho da mesma. Esse resultado pode ser verificado na Figura 40.

```
No.     Time        Source                Destination           Protocol Info
 7 1.707990    192.168.0.69          192.168.0.62          PGSQL    >Q

Frame 7 (146 bytes on wire, 146 bytes captured)
Ethernet II, Src: AsustekC_a7:b3:3d (00:22:15:a7:b3:3d), Dst: IntelCor_6a:1d:0b
(00:1c:bf:6a:1d:0b)
Internet Protocol, Src: 192.168.0.69 (192.168.0.69), Dst: 192.168.0.62
(192.168.0.62)
Transmission Control Protocol, Src Port: smart-lm (1608), Dst Port: postgresql
(5432), Seq: 1, Ack: 1, Len: 92
PostgreSQL
 Type: Simple query
 Length: 91
 Query: Select Login from USUARIO where Filial='01' or Filial='00' or Nivel='9'
Order By Login

No.     Time        Source                Destination           Protocol Info
 8 1.708833    192.168.0.62          192.168.0.69          PGSQL
<T/D/D/D/D/D/D/D/D/D/D/D/D/D/D/D/D/D/D/D/D/D/D/D/D/D/D/D/D/D/D/D/D/D/D/D/C/Z

Frame 8 (935 bytes on wire, 935 bytes captured)
Ethernet II, Src: IntelCor_6a:1d:0b (00:1c:bf:6a:1d:0b), Dst: AsustekC_a7:b3:3d
(00:22:15:a7:b3:3d)
Internet Protocol, Src: 192.168.0.62 (192.168.0.62), Dst: 192.168.0.69
(192.168.0.69)
Transmission Control Protocol, Src Port: postgresql (5432), Dst Port: smart-lm
(1608), Seq: 1, Ack: 93, Len: 881

PostgreSQL                    PostgreSQL                    PostgreSQL
    Type: Row                     Type: Data row                Type: Data row
```

```
description
    Length: 30
    Columns: 1
                    Length: 25
                    Columns: 1
                                    Length: 25
                                    Columns: 1

PostgreSQL
    Type: Data row
    Length: 25
    Columns: 1
PostgreSQL
    Type: Data row
    Length: 25
    Columns: 1
PostgreSQL
    Type: Data row
    Length: 25
    Columns: 1
```

Figura 40 : Resultado da verificação dos pacotes no servidor Linux Ubuntu 8.04.

Com os resultados obtidos no teste de segurança, concluí-se que é de extrema urgência a aplicação de criptografia nos dados do sistema. Alguns SGDBs já possuem essa funcionalidade, ficando também para trabalhos futuros esse controle.

Para trabalhos futuros propõe-se que seja realizada a aplicação da proposta em um estudo de caso que possa ser realizado por completo, contemplando também a execução do projeto. Após analisar os efeitos causados por esta abordagem e seu impacto sobre o projeto. Como trabalho futuro também é sugerido a documentação do sistema.

6.6 Considerações Finais

Com a execução de cada caso de teste, pode-se provar a funcionalidade do sistema, a melhor conclusão sobre sua velocidade e concorrência, e, também o melhor conhecimento sobre o controle dos erros e exceções geradas pelo sistema.

Os testes tinham como objetivo medir primeiramente o desempenho, um retorno melhor do SGDB quando solicitado, tanto por alguma ferramenta de gerenciamento quanto pelo próprio sistema. Os testes de sistema e validação foram importantes para validar as alterações realizadas nos fontes no sistema e por fim o teste de segurança e estresse ajudaram na parte principalmente de acesso e concorrência.

Como resultados era esperado uma melhor resposta do SGDB PostgreSQL, principalmente utilizando o sistema, e também uma melhor resposta do sistema quando a acessos concorrentes. Nos outros, teste de sistema, validação e segurança os resultados esperados, foram os resultados atingidos.

7 CONCLUSÃO

Pode-se concluir que o trabalho conseguiu atingir o seu objetivo, identificando o problema e desenvolvendo a solução: a execução do sistema independente da base de dados selecionada.

Algumas funcionalidades necessárias podem ter sido passadas despercebidas pelo projeto, mas como o sistema está em constante execução e teste, outros desenvolvedores poderão trabalhar em cima do código-fonte e corrigir alguma inconsistência caso necessário. Algumas outras funcionalidades não serão implementadas agora por não haver tempo hábil para o desenvolvimento, e por também não tratar de pendências de grande importância, como por exemplo o Módulo Logística e EDI, os quais são módulos que .serão rescritos num futuro próximo.

Referente a migração dos dados, pode-se concluir que o espaço de disco utilizado pelo Microsoft SQL Server é maior que os demais SGDBs, e que o tempo de migração o Firebird leva uma pequena vantagem em relação ao PostgreSQL. Uma grande ajuda é em relação as ferramentas, todas obtiveram sucesso na migração, ficando apenas como uma dica, que elas respeitem as regras de chaves estrangeiras.

As alterações nos fontes do sistema SistranNet foi a mais custosa, devido ao número de arquivos de código fontes. Como o projeto foi realizado por diversos programadores e não tinha um padrão definido, diversas divergências foram encontradas, sendo necessário outros testes serem executados, para evitar erros no sistema.

Os benefícios trazidos pelo desempenho eficaz e eficiente das atividades de teste são produtos de *software* mais confiáveis e de maior qualidade, e pode-se afirmar isso com a ajuda de ferramentas de testes. Elas ajudaram muito nessa tarefa, tendo um retorno bastante valioso, quanto a tempo e erros. Uma vasta opções de ferramentas de licenças livres estão presentes no mercado, mas muitas deixam a desejar em relação a configurações, sejam elas por pré-requisitos, configuração de execução ou forma de retorno.

Com a tecnologia que temos a nossa disposição, é perfeitamente possível fazer testes precisos que possam garantir isso, pois tem-se uma vasta gama de ferramentas para realização de testes, inclusive muitas delas gratuitas. Também temos vários tipos de SGBDs mais ou menos robustos, gratuitos ou pagos. Certamente vale a pena investir um tempo maior com testes na fase de desenvolvimento para não correr riscos após a implantação.

Em relação as contribuições, neste trabalho foi possível verificar que a tarefa de teste de sistema pode ser feito de uma forma mais ágil sem comprometer o projeto. Assim a abordagem proposta considera importante a participação do testador de *software* no projeto, o envolvimento e empenho da equipe e principalmente a entrega de um produto de maior qualidade, sempre visando manter o prazo e os custos do projeto.

O teste de desempenho é um item que merece melhor destaque entre todos. Nos testes realizados, o SGDB que originou o sistema SQL Server da Microsoft, teve um melhor resultado que o PostgreSQL, em virtude de utilizar o mesmo sistema operacional. Como trabalho futuro, sugere-se criar a segunda camada da aplicação multiplataforma, para realizar novamente testes e verificar a velocidade do SGDB PostgreSQL.

Com os resultados obtidos até então, vislumbra-se a possibilidade de cumprir o plano de trabalho por completo, atacando todas as divergências e resolvendo-as sem maiores problemas.

Na próxima seção, é demonstrado um breve apanhado sobre o gerenciamento do projeto aqui executado, uma explicação sobre o tempo, horas trabalhadas e uma média do custo atingido.

7.1 Gerenciamento do Projeto

Para o projeto aqui levantado, a atividade de gerenciar o tempo e custo também faz parte desse processo, principalmente de planejamento para a entrega do trabalho. Para auxiliar, foi utilizada os casos de uso para auxiliar nessas estimativas para a definição do cronograma do projeto, com estimativa em horas e custo hora de cada atividade. Essas informações são acrescentadas com o custo de hora trabalhada, estimada em R$ 10,00.

7.1.1 Gerenciamento de tempo

Nos projetos de desenvolvimento de software o tempo é cada vez mais reduzido, e normalmente o tempo é fator crítico. Por esse motivo é vital para o projeto gerenciá-lo, isso evita atrasos no projeto e gera históricos para projetos futuros. É importante também citar que o atraso do projeto causa transtornos. Além disso, do ponto de vista dos métodos ágeis, os incrementos devem ser planejados de modo que o cliente receba entregas frequentes do software (PMBOK, 2004).

7.1.2 Gerenciamento de custo

É muito ligado ao tempo do projeto está o custo. Os projetos possuem um orçamento estipulado inicialmente que deve ser mantido. Esse custo está ligado com o tempo de duração do projeto e deve ser planejado para não gerar estouros no orçamento. O custo de um projeto também é crítico, pois o mesmo não deve terminar gerando prejuízo para a organização, ao contrário, deve ser planejado e controlado para que ao final se mantenha no custo estipulado (PMBOK, 2004).

O PMBOK (PMBOK, 2004), sugere as seguintes atividades para o gerenciamento de custos:

- Estimativa de custos: É uma aproximação dos custos de cada atividade, considerando os seus recursos. Os custos com materiais e equipamentos também devem ser estimados. Deve ser levado em conta ainda possíveis variações de custos e de riscos. A estimativa deve identificar e considerar diversas alternativas de custos. A exatidão desta estimativa para cada atividade pode ser verificada de acordo com o andamento do projeto.

- Orçamentação: Este processo soma os custos individuais estimados de cada atividade, a fim de gerar os custos totais do projeto e, com isso, possibilitar a avaliação de desempenho do projeto.

- Controle de custos: controlar periodicamente os custos, identificando possíveis variações e/ou estouros nos custos iniciais. Monitorar todas as mudanças ocorridas nos custos inicialmente estimados identificando suas causas e avaliando seu impacto.

7.1.3 Resultados Estimados

Como saída desses estudos, foi gerada a estimativa de custos de cada atividade, bem como o custo total do projeto além de cronogramas mais confiáveis, ficando assim mais fácil de gerenciar o projeto.

Todos os gráficos e relatórios aqui apresentados, foram realizados utilizando a ferramenta da Microsoft, Office Project 2007.

Primeiramente foi realizado o levantamento de horas trabalhadas [pr quinzena, acumulando um total estipulado em 250 horas (Figura 41), após, com base nas horas trabalhadas e no custo da hora trabalhada estimado em R$ 10,00, um gráfico de custo por horas trabalhadas (Figura 42).

O custo de R$ 10,00 pela hora trabalhada, foi retirado com base no valor do Sindicato das Empresas de Informática do Estado do Rio Grande do Sul, onde o salário de um analista de sistemas está estipulado em R$1.874,51 mais benefícios, conforme convenção realizada em novembro de 2008.

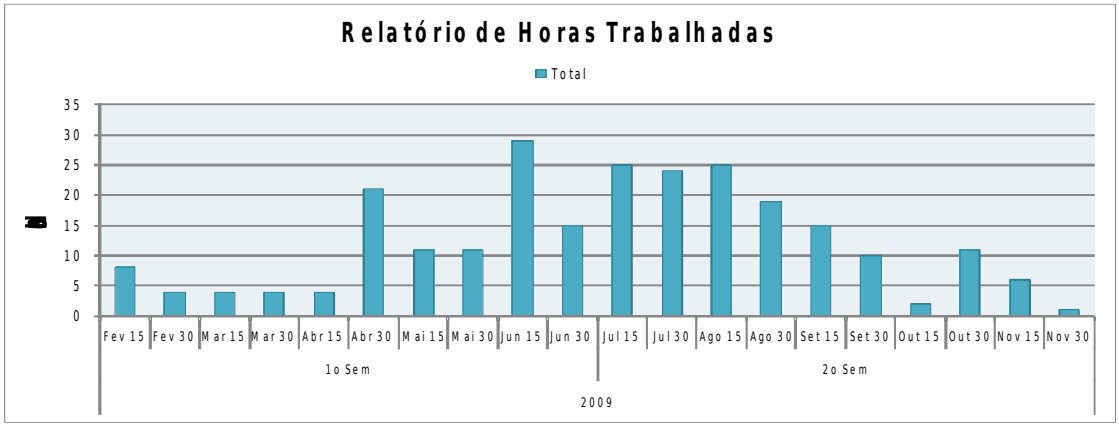


Figura 41 :Levantamento de horas trabalhadas

Abaixo gráfico do custo estimado do trabalho:

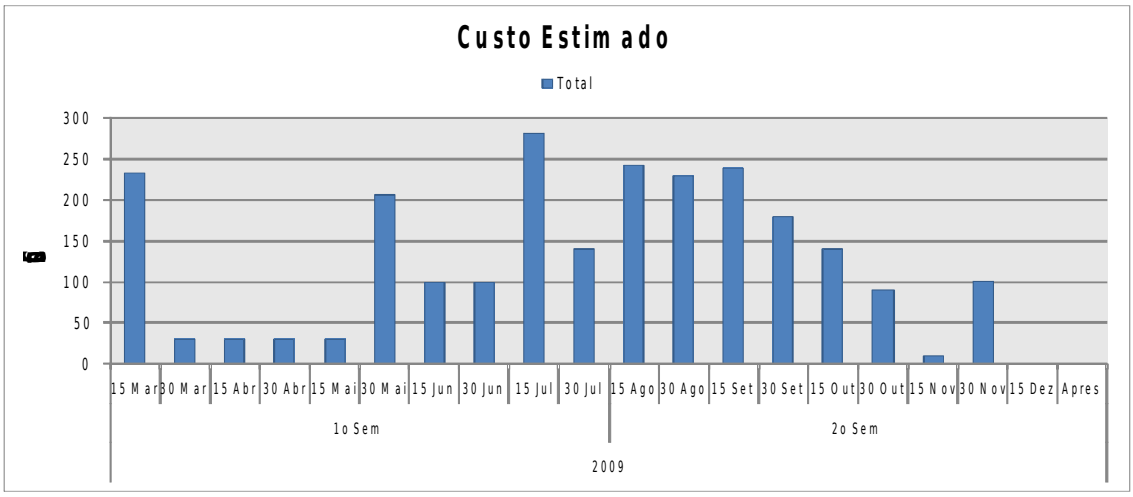


Figura 42 : Levantamento do custo do projeto

Com base nos dois levantamentos, horas trabalhadas e custo da hora, é possível realizar um gráfico mostrando o comparativo entre o custo e as horas trabalhadas (Figura 43).

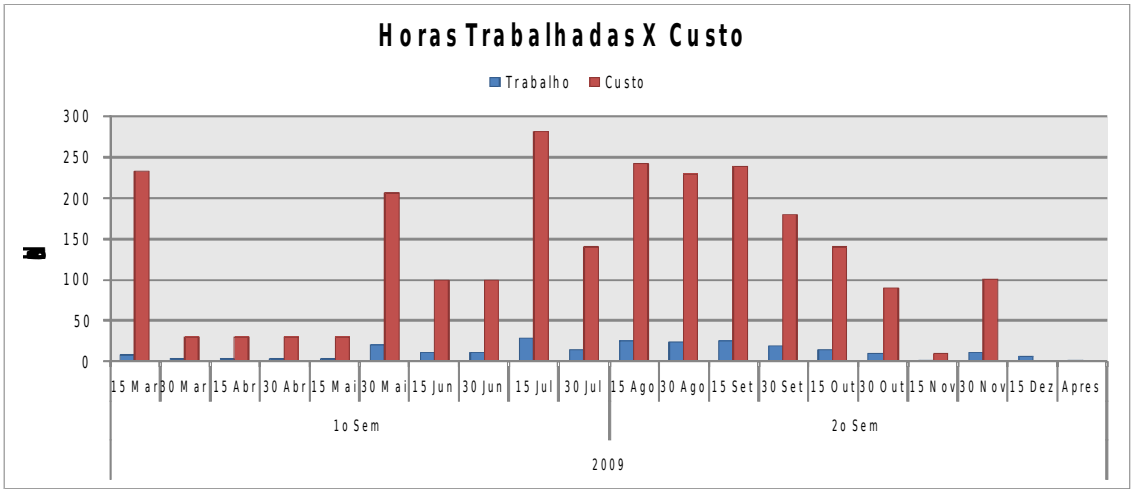


Figura 43: Comparativo entre as horas trabalhadas e o custo total.

E por fim, um relatório mostrando o custo acumulado por, ao final do projeto, estimado em aproximadamente R$ 2.400,00, que pode ser verificado na Figura 44.

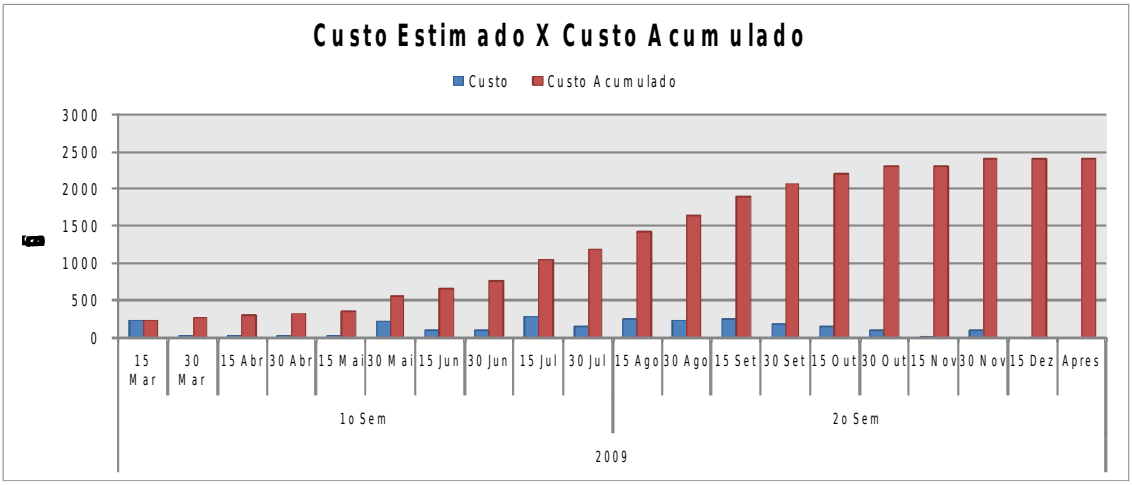


Figura 44 : Relatório de custo acumulado.

7.2 Trabalhos Futuros

A execução dos testes trouxe uma segurança muito maior para a empresa, e também alguns itens para serem melhores analisados para alterações no futuro, entre eles destaca-se:

- Uma melhor forma de controle concorrente aos dados, visto isso na seção 6.5, onde os teste de concorrência deixaram muito a desejar;

- Uma melhor forma de planejamento de padrão de *software* para que trabalhos futuros realizados na aplicação, mantenham-se em melhor ordem e estrutura;

- Modificar o sistema para multi-plataforma;

- Utilizar uma melhor criptografia dos dados, visto na seção 6.5.4, que pode-se facilmente obter senhas e valores com programas de leitura de redes.

REFERÊNCIAS BIBLIOGRAFICAS

ANSI, American National Standart Institute. **SQL 2003.** Disponível em <http://www.ansi.org/>. Acesso em Jun/2009.

ANSI-SQL 92. **Database Language SQL, ANSI/ISO/IEC International Standard (IS)**, ISO/IEC 9075:1992

ANSI-SQL 99. **Database Language SQL – Part 2: Foundation (SQL/Foundation), ANSI/ISO/IEC International Standard (IS)**, ISO/IEC 9075:1999

APACHE, Software Foundation. **JMETER User's Manual.** Disponível em <http://jakarta.apache.org/jmeter/usermanual/index.html>. Acesso em Mai/2009.

BATTISTI, Júlio. **SQL Server 2005 Administração e Desenvolvimento Curso Completo**. Rio de Janeiro: Axcel Books, 2005.

CANTÚ, Marco. **Dominando o Delphi 7 : a bíblia** Tradução Kátia Aparecida Roque; Revisão técnica Álvaro Rodrigues. São Paulo: Persona Makron Books, 2003.

CLUBE DELPHI. **Aplicações Multi Banco.** Ano 6 e Edição 75.– São Paulo. 2006.

DATE, C J. **Introdução a sistemas de banco de dados.** Tradução de Daniel Vieira. – Rio de Janeiro : Elsevier, 2003 – 2ª Reimpressão.

ELMASRI, Ramez & NAVATHE, Shmkant B.. **SISTEMA DE BANCO DE DADOS**. São Paulo : Addison Wesley, 2005.

ESAFETY SISTEMAS E TECNOLOGIA. Fonte : < http://www.esafety.com.br > acessado em Fev/2009.

FERRAÇA, Maycon. **MIGRATE DATABASES - Migrador Genérico de Bancos de Dados Relacionais.** 2008. 126f. Monografia (Conclusão do Curso) – Universidade de Caxias do Sul, Centro de Ciências Exatas e Tecnologia, Caxias do Sul – RS.

GUSTAFSON, David A. **Teoria e problemas de engenharia de software**. Trad. Fernanda Cláudia Alves Campos. - Porto Alegre : Bookman, 2003.

MICROSOFT SQL SERVER 2005. Fonte: <http://www.microsoft.com/brasil/servidores/sql/2005/default.mspx > acessado em Fev/2009.

MSDN, **Microsoft Developer Network**. Fonte: <http://msdn.microsoft.com/pt-br/library/aa833205.aspx> acessado em Out/2009.

PEREIRA NETO, Álvaro. **POSTGRESQL: TÉCNICAS AVANÇADAS: VERSÕES OPEN SOURCE: SOLUÇÕES PARA DESENVOLVEDORES E ADMINISTRADORES DE BANCOS DE DADOS.** São Paulo: Érica, 2003.

PETERS, James F. **Engenharia de Software / James F Peters, Witold Pedrycz ;** Tradução de Ana Patrícia Garcia. - Rio de Janeiro : Campus, 2001.

PMBOK. [S.l.: s.n.], 2004. **Um guia do conjunto de conhecimentos em Gerenciamento de Projetos** (Guia PMBOK).

POSTGRESQL, Global Development Group. Fonte < http://sourceforge.net/ projects/pgdocptbr > acessado em Fev/2009.

PRESSMAN, Roger S. **Engenharia de software**. Tradução José Carlos Barbosa dos Santos. – São Paulo : Pearson Education do Brasil. 2007.

SHAPIRO, Jeffrey. **SQL Server 2000 completo e total.** Tradução: Aldair José Coelho Corrêa da Silva / Flavia Barktevicius Cruz, Revisão técnica: Flavio Gomes Ferreira. São Paulo: Makrin Books Ltda, 2002.

SILBERSCHATZ, Abraham; KORTH, Henri F. & SUDARSHAN, S. **Sistema de Banco de Dados.** São Paulo : MAKRON Books, 1999.

SOMMERVILLE, Ian. **Engenharia de Software.** Tradução André Maurício de Andrade Ribeiro ; revisão técnica Kechi Hirama. – São Paulo : Addison Wesley, 2003 – 6ª edição.

SQL MAGAZINE. **Bancos de Dados Free.** Ano 3 e Edição 37.– São Paulo. 2006.

OPENSOURCETESTING. **OPENSOURCETESTING – open source software testing tools, news and discussion**, disponível em: <http://opensourcetesting.org/>. Acesso em Jun/2009.

TIGRIS. **TIGRIS - Open Source Software Engineering Tools**, disponível em: <http://tigris.org/>. Acesso em Jun/2009.

ANEXO A : MIGRAÇÃO DO BANCO DE DADOS

Após a instalação do *software* ESF Database Convert Standard Edition, versão 5.9.66 e da criação do banco de dados no PostgreSQL, a conversão é efetuada. Ao iniciar o *software*, a tela de apresentação é mostrada. Para iniciar a conversão o *software* pede a origem, o destino e as tabelas que irão ser selecionadas para a conversão. As figuras abaixo demonstram a execução:

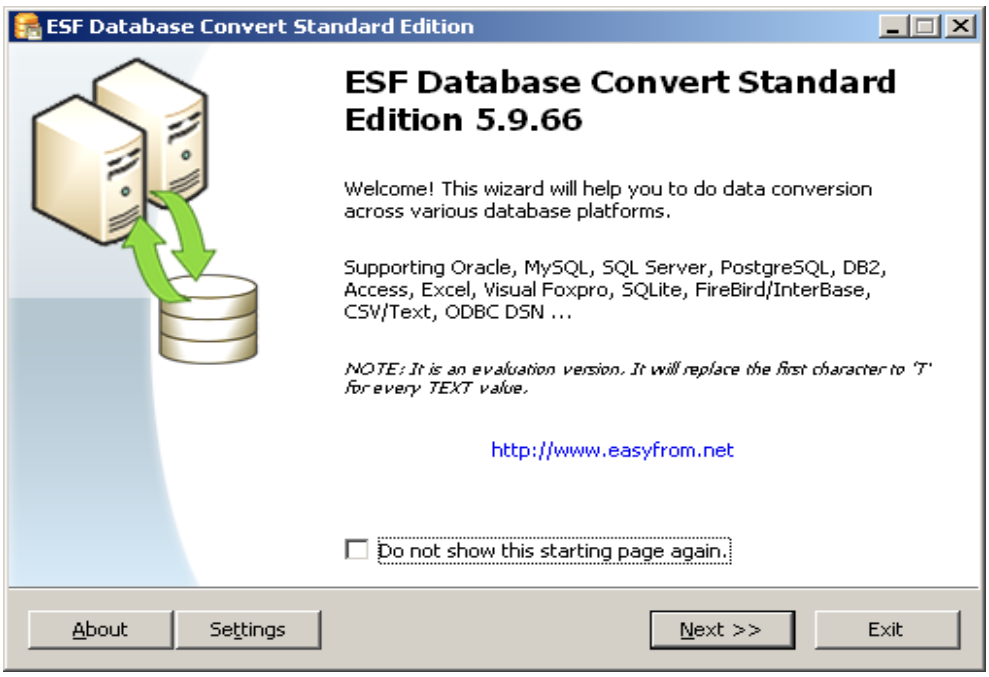


Figura 45: Tela de abertura do ESF Database Convert.

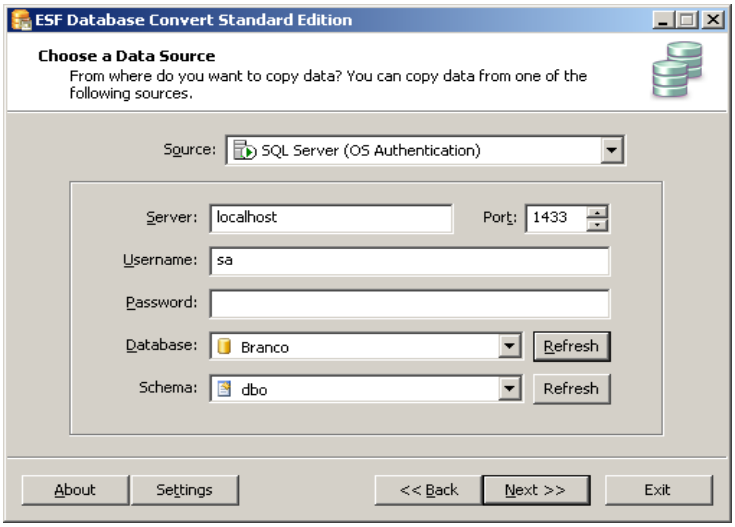


Figura 46 : Selecionando a origem, no caso o banco de dados do SQL Server

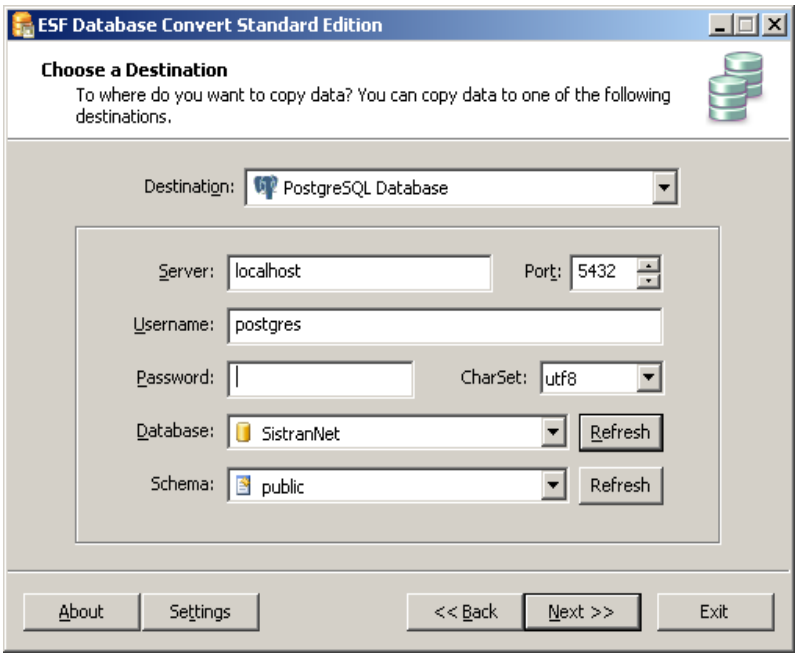


Figura 47 : Selecionando o destino, no caso o banco de dados do PostgreSQL.

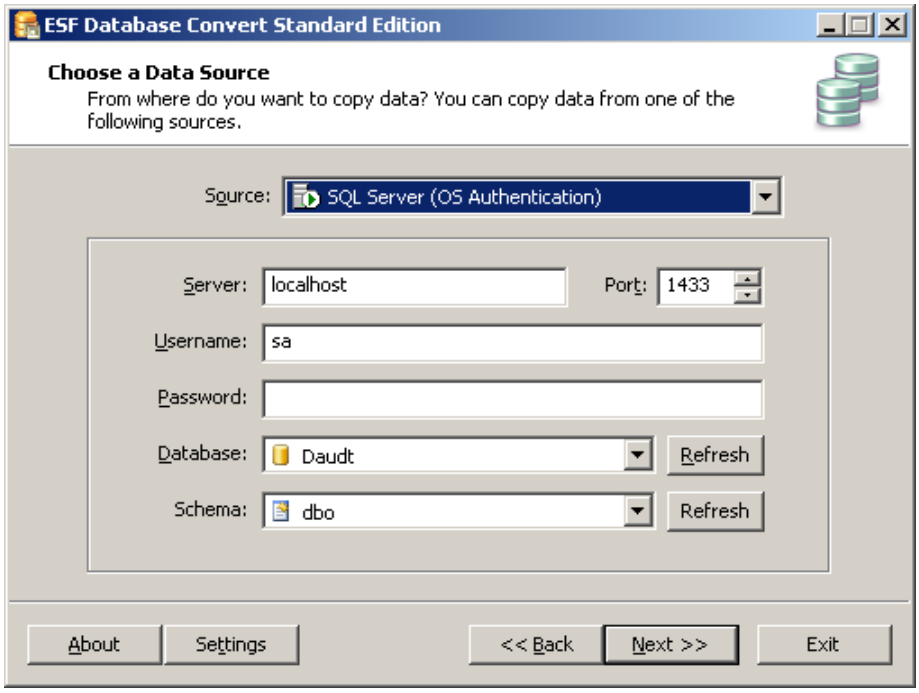


Figura 48 : Selecionando a origem, no caso o banco de dados do SQL Server

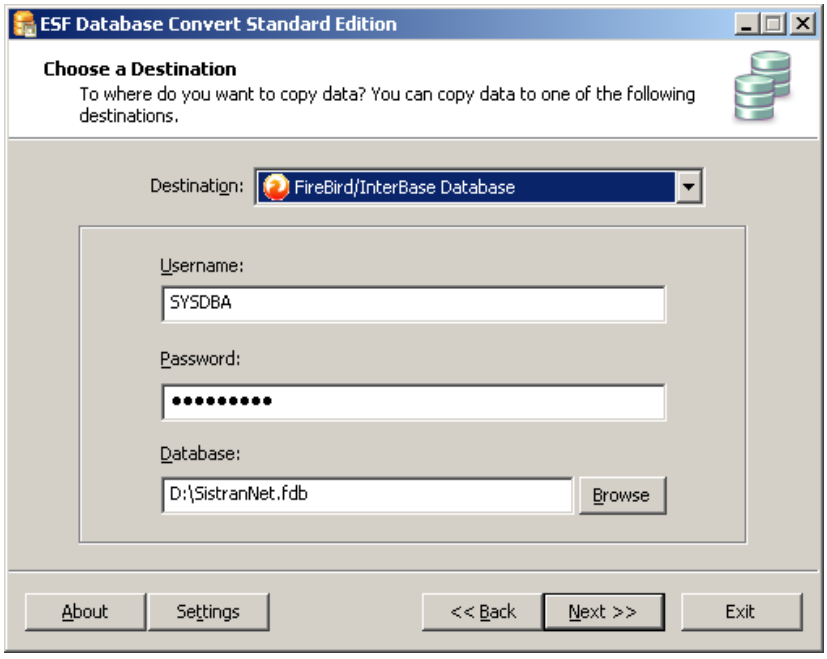


Figura 49 : Selecionando o destino, no caso o banco de dados do Firebird.

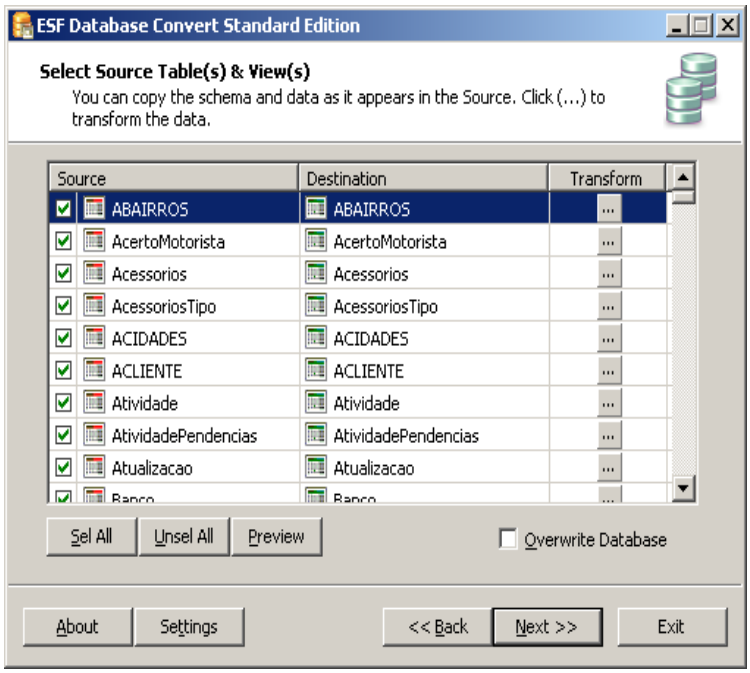


Figura 50 : Seleção das tabelas a importar.

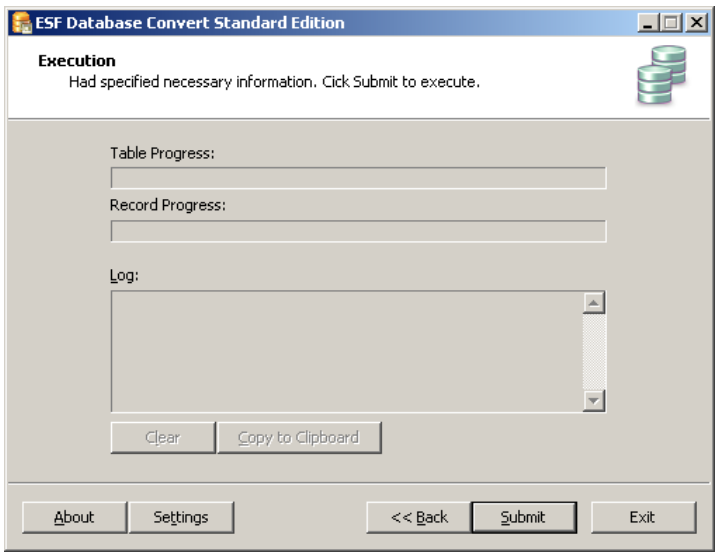


Figura 51: Tela para a conversão dos dados.

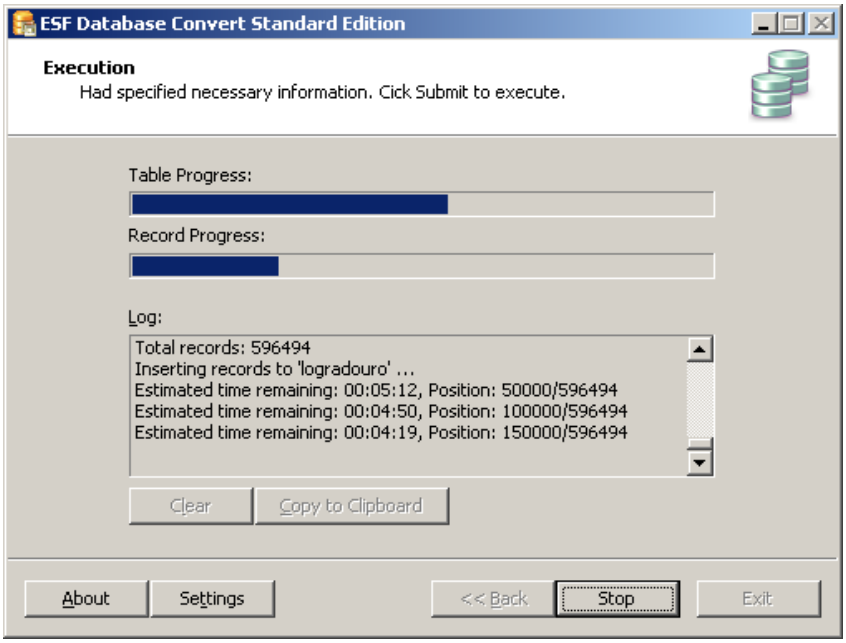


Figura 52 : Conversão em andamento.

Interessante mencionar que após clicar no botão para executar a conversão, como o software é Trial, ele exibe a mensagem dizendo que para a conversão, em todos os campos textos (varchar, char, etc.) o software adiciona um caractere ‘T’ na sua frente. Como a importância não é a migração dos dados, não será um empecilho para o trabalho.

Para título de curiosidade, abaixo segue o resultado obtido da migração da tabela de Conhecimentos, a qual tem uma grande diversidade de tipo de campos. A Figura 53 representa a tabela no PostgreSQL, a seguinte representa a mesma tabela no Firebird e por fim, a Figura 55, no SQL Server.

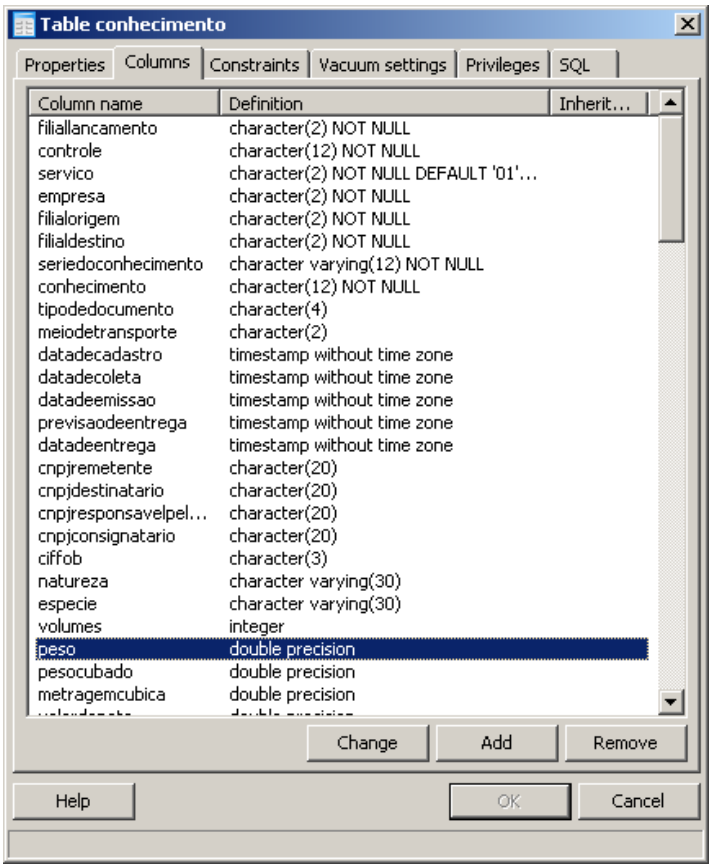


Figura 53: Estrutura da Tabela de Conhecimentos no PostgreSQL.

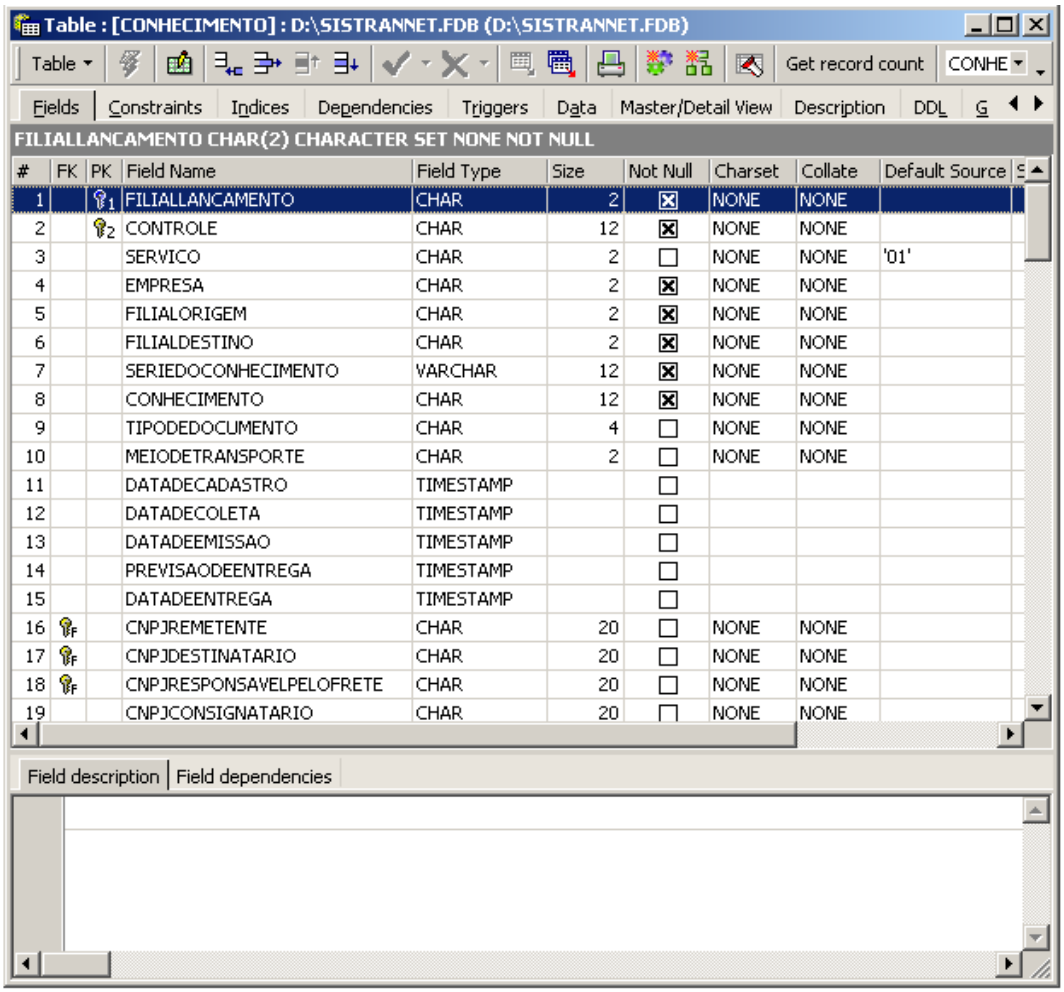


Figura 54 : Estrutura da Tabela de Conhecimentos no Firebird.

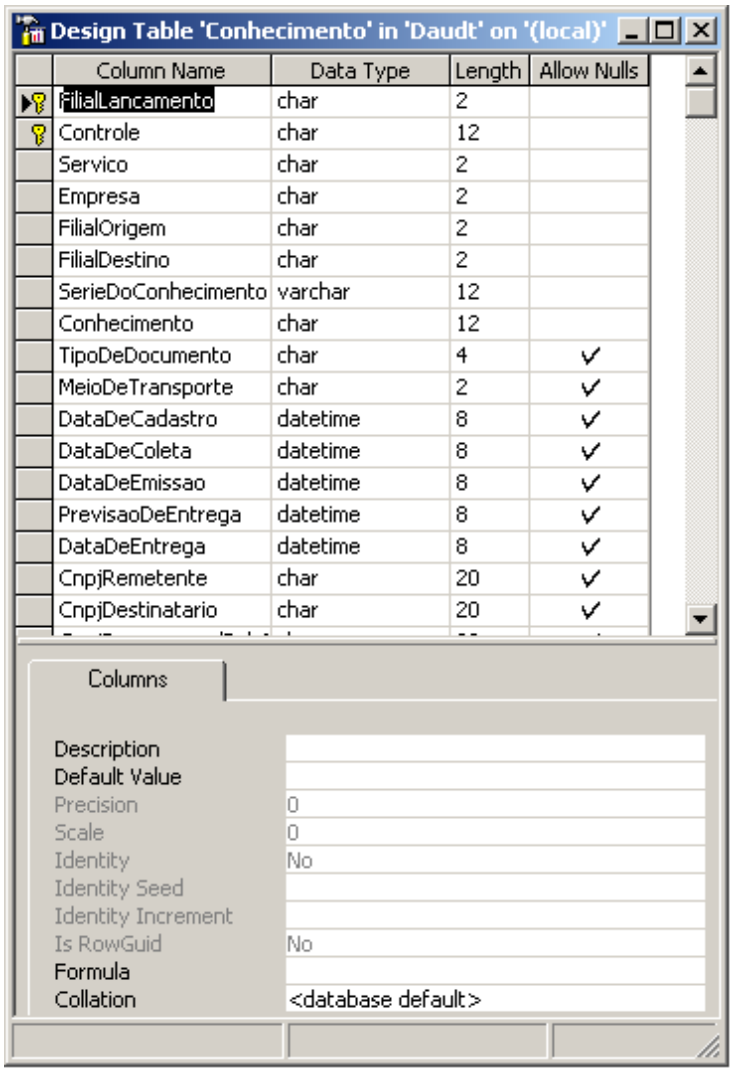


Figura 55 : Estrutura da tabela de Conhecimentos no SQLServer.

Continuando a demonstrar o resultado e seguindo no exemplo da tabela de conhecimentos, abaixo é demonstrado os principais campos da tabela e o *script* para a criação do banco nos 3 SGDBs.

- PostgreSQL:

```
CREATE TABLE conhecimento
(
  filiallancamento character(2) NOT NULL,
  controle character(12) NOT NULL,
  servico character(2) NOT NULL DEFAULT '01'::bpchar,
  empresa character(2) NOT NULL,
  filialorigem character(2) NOT NULL,
  filialdestino character(2) NOT NULL,
  seriedoconhecimento character varying(12) NOT NULL,
  conhecimento character(12) NOT NULL,
  tipodedocumento character(4),
  meiodetransporte character(2),
  datadecadastro timestamp without time zone,
  datadecoleta timestamp without time zone,
  datadeemissao timestamp without time zone,
  previsaodeentrega timestamp without time zone,
  datadeentrega timestamp without time zone,
  cnpjremetente character(20),
  cnpjdestinatario character(20),
  cnpjresponsavelpelofrete character(20),
  cnpjconsignatario character(20),
  ciffob character(3),
 …….....
  CONSTRAINT pk_conhecimento PRIMARY KEY (filiallancamento, controle),
  CONSTRAINT fk_conhecimentodest_cliente FOREIGN KEY (cnpjdestinatario)
      REFERENCES cliente (cnpj) MATCH SIMPLE ON UPDATE NO ACTION ON DELETE CASCADE,
  CONSTRAINT fk_conhecimentoreme_cliente FOREIGN KEY (cnpjremetente)
      REFERENCES cliente (cnpj) MATCH SIMPLE ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT fk_conhecimentorespcliente FOREIGN KEY (cnpjresponsavelpelofrete)
      REFERENCES cliente (cnpj) MATCH SIMPLE ON UPDATE NO ACTION ON DELETE NO ACTION
) WITHOUT OIDS;
ALTER TABLE conhecimento OWNER TO postgres;
```

- Firebird:

```
CREATE TABLE CONHECIMENTO (
    FILIALLANCAMENTO           CHAR(2) NOT NULL,
    CONTROLE                   CHAR(12) NOT NULL,
    SERVICO                    CHAR(2) DEFAULT '01',
    EMPRESA                    CHAR(2) NOT NULL,
    FILIALORIGEM               CHAR(2) NOT NULL,
    FILIALDESTINO              CHAR(2) NOT NULL,
    SERIEDOCONHECIMENTO        VARCHAR(12) NOT NULL,
    CONHECIMENTO               CHAR(12) NOT NULL,
    TIPODEDOCUMENTO            CHAR(4),
    MEIODETRANSPORTE           CHAR(2),
    DATADECADASTRO             TIMESTAMP,
    DATADECOLETA               TIMESTAMP,
    DATADEEMISSAO              TIMESTAMP,
    PREVISAODEENTREGA          TIMESTAMP,
    DATADEENTREGA              TIMESTAMP,
    CNPJREMETENTE              CHAR(20),
    CNPJDESTINATARIO           CHAR(20),
    CNPJRESPONSAVELPELOFRETE   CHAR(20),
    CNPJCONSIGNATARIO          CHAR(20),
    CIFFOB                     CHAR(3)
…..............................
);

ALTER TABLE CONHECIMENTO ADD CONSTRAINT PK_CONHECIMENTO PRIMARY KEY (FILIALLANCAMENTO,
CONTROLE);
```

- SQL Server:

```
CREATE TABLE [dbo].[Conhecimento] (
        [FilialLancamento] [char] (2) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
        [Controle] [char] (12) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
        [Servico] [char] (2) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
        [Empresa] [char] (2) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
        [FilialOrigem] [char] (2) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
        [FilialDestino] [char] (2) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
        [SerieDoConhecimento]  [varchar]  (12)  COLLATE  SQL_Latin1_General_CP1_CI_AS  NOT
NULL ,
        [Conhecimento] [char] (12) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
        [TipoDeDocumento] [char] (4) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
        [MeioDeTransporte] [char] (2) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
        [DataDeCadastro] [datetime] NULL ,
        [DataDeColeta] [datetime] NULL ,
        [DataDeEmissao] [datetime] NULL ,
        [PrevisaoDeEntrega] [datetime] NULL ,
        [DataDeEntrega] [datetime] NULL ,
        [CnpjRemetente] [char] (20) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
        [CnpjDestinatario] [char] (20) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
        [CnpjResponsavelPeloFrete]  [char]  (20)  COLLATE  SQL_Latin1_General_CP1_CI_AS
NULL ,
        [CnpjConsignatario] [char] (20) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
        [CifFob] [char] (3) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
        …..............
        [Motorista] [char] (20) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
        [DocumentoSAPBASF] [varchar] (15) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
        [RotaGrupo] [varchar] (10) COLLATE SQL_Latin1_General_CP1_CI_AS NULL
) ON [PRIMARY]
GO
```

Durante a conversão foi acompanhado o desempenho do computador de teste. Veja nas duas imagens abaixo:

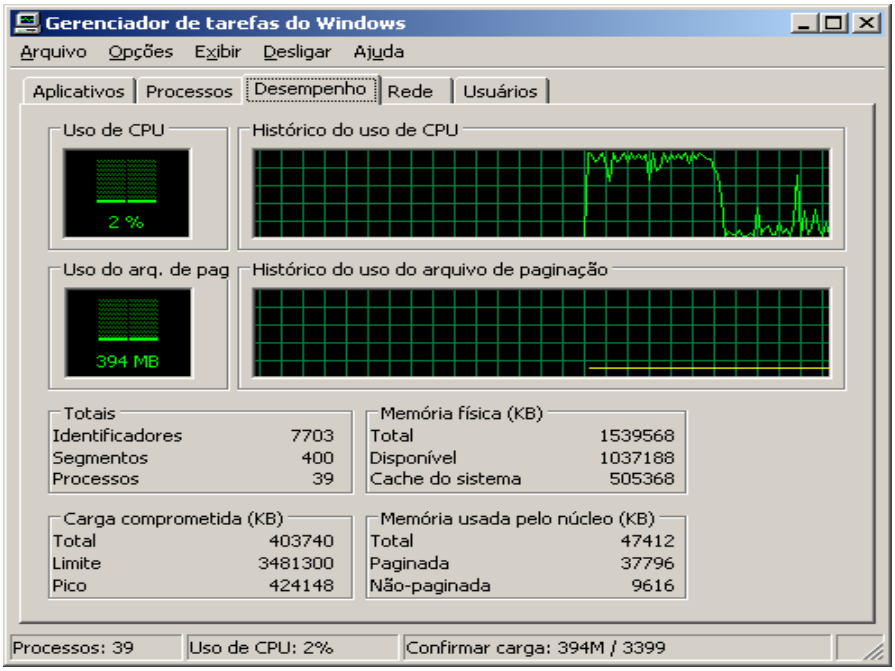


Figura 56: Acompanhamento do processamento do computador durante a conversão do banco de dados para o Firebird.

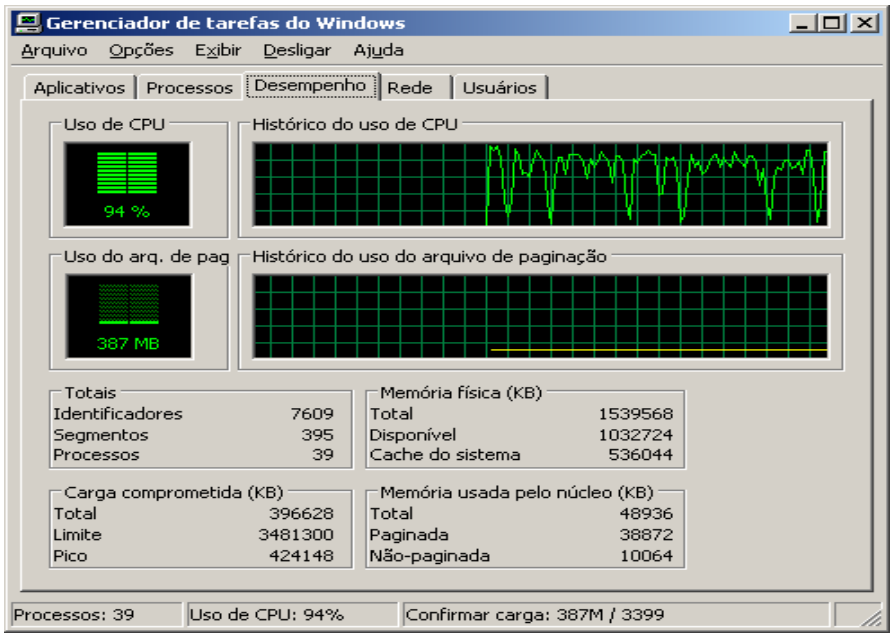


Figura 57 : Acompanhamento do processamento do computador durante a conversão do banco de dados para o PostgreSQL.

ANEXO B : CONFIGURAÇÃO ODBC

O software de conversão utilizado foi Trial, o qual não permite a migração dos dados. Portanto, foi utilizada a ferramenta grátis Migrate Database (Ferraça, 2006). Para acessar os bancos de dados, é necessária a configuração das fontes de dados ODBC.

Para o acesso aos bancos Firebird e PostgreSQL, foi necessário baixar o software para instalação do driver ODBC.

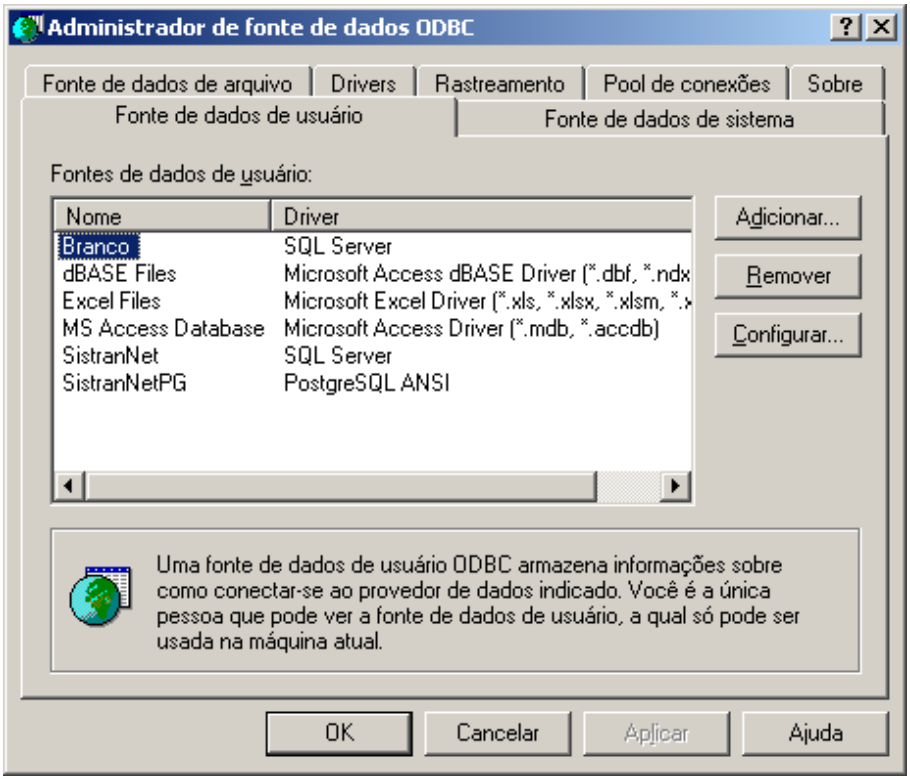


Figura 58: Tela inicial do Administrador de fonte de de dados ODBC do Windows.

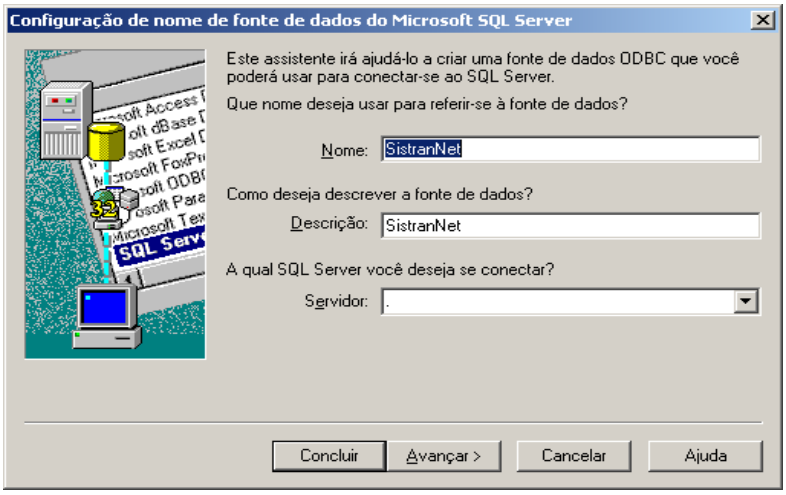


Figura 59 : Início da configuração de acesso do banco de dados SQL Server, escolha do servidor do SQL Server. O ponto no campo significa que o acesso vai ser local, no mesmo computador.

Abaixo é mostrada a configuração do ODBC para acesso ao SQL Server. O *driver* para acesso é instalado automaticamente quando instalado o SQL Server no computador.

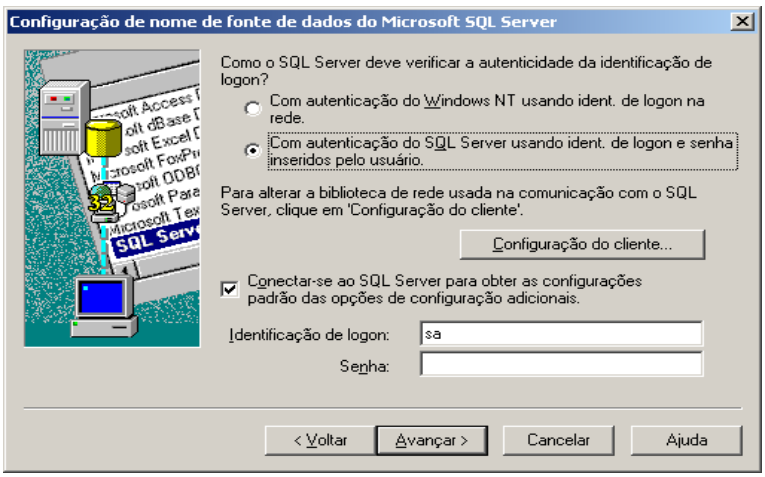


Figura 60 : Configuração da autenticidade da identificação do logon.

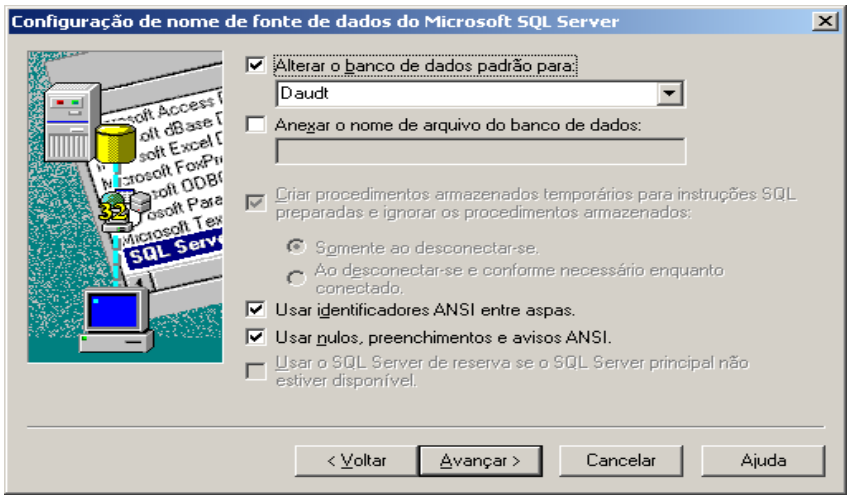


Figura 61 : Escolha do banco de dados para o acesso do ODBC.

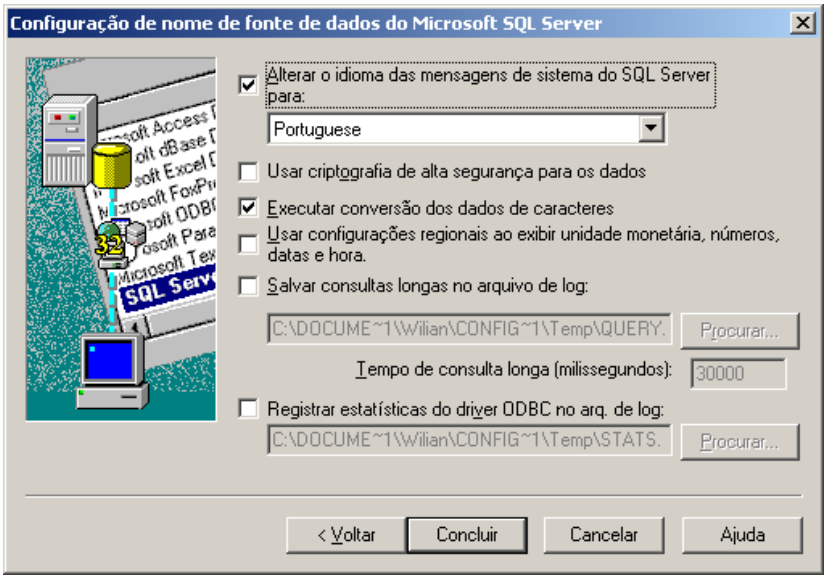


Figura 62 : A escolha do idioma das mensagens para o português, os demais campos deixados como padrão.

Abaixo é mostrada a configuração do ODBC para acesso ao Firebird. O *driver* pode ser baixado em http://www.Firebird.org/index.php?op=devel&sub=odbc. A versão utilizada foi a de 21 de outubro de 2008.

A configuração do Firebird é bastante simples. Escolhe-se o nome do data source, a descrição, o local do database. Seta-se, por fim, o nome do usuário e senha e o Data Source está criado.

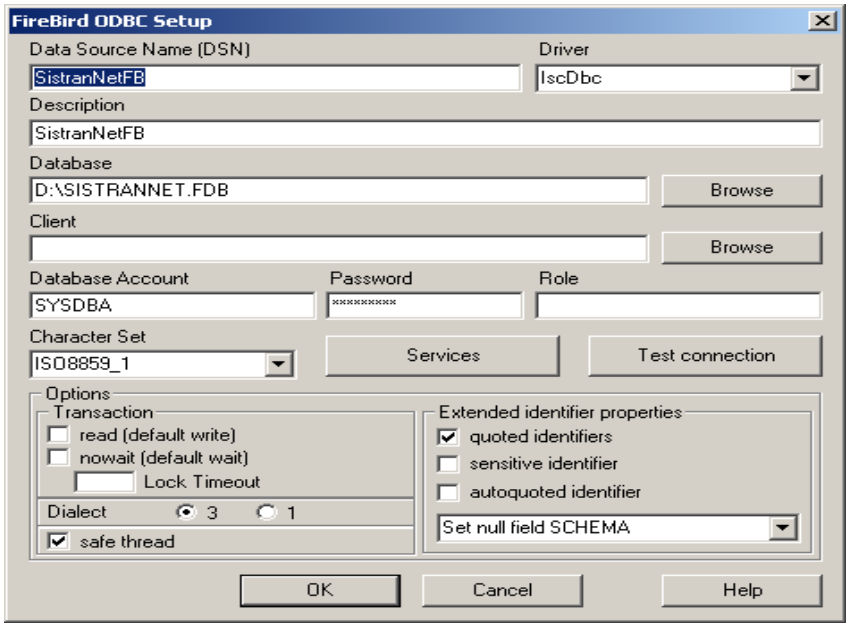


Figura 63 : Tela de configuração do *driver* ODBC para o Firebird.

Abaixo é mostrada a configuração do ODBC para acesso ao PostgreSQL. O *driver* pode ser baixado em http://www.postgresql.org/ftp/odbc/versions/msi/. A versão utilizada foi a 08.02.0105.

A configuração do PostgreSQL é simples como a do Firebird. Digita o nome do D*ata Source*, a descrição, o nome do *database*, o nome do servidor, a porta de conexão (5432 por padrão). Por fim, seta-se, o nome do usuário e senha e o *Data Source* está criado.

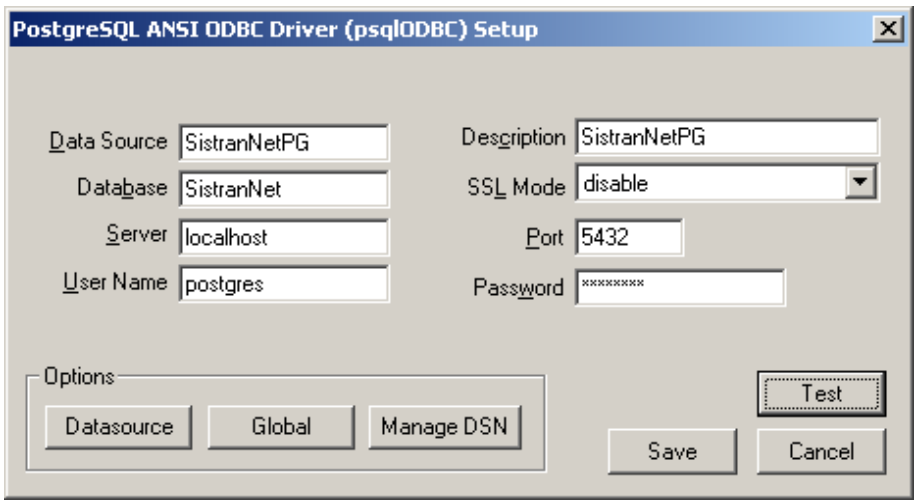


Figura 64 : Tela de configuração do driver ODBC para o PostgreSQL.

ANEXO C : MIGRAÇÃO DOS DADOS

Para a migração dos dados, foi utilizada a ferramenta *free* Migrate Database. Primeiramente conecta-se a origem e o destino, conforme Figura 65 e 66, para depois realizar a execução da conversão. Após finalizada, o software gera um *log* com todas as informações de erros que aconteceram.

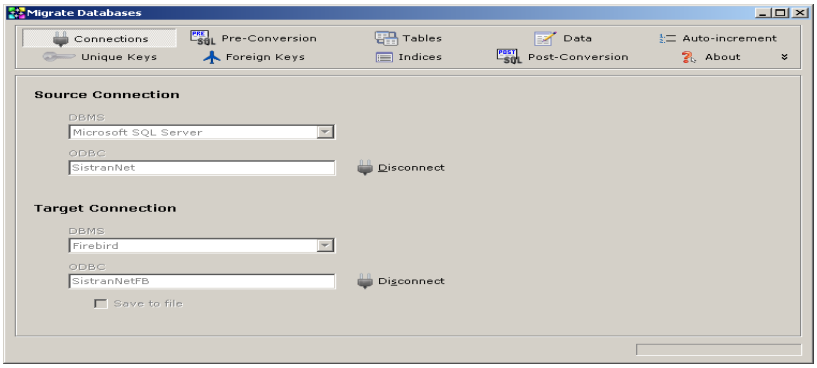


Figura 65 : Conexão SQL Server para Firebird.

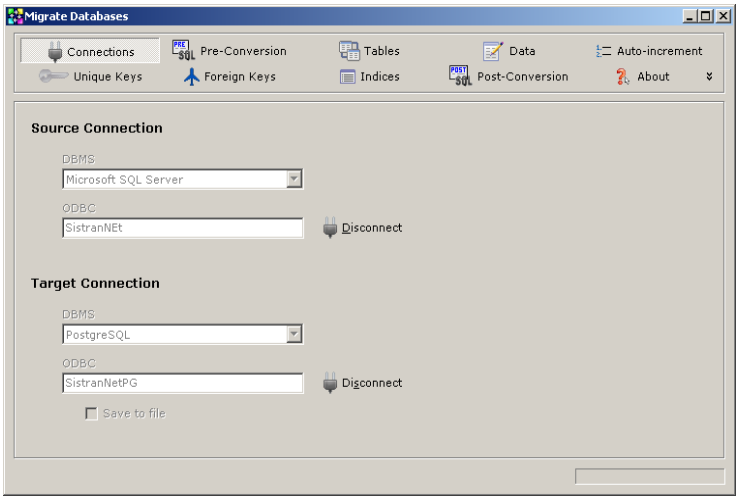


Figura 66 : Conexão SQL Server para PostgreSQL.

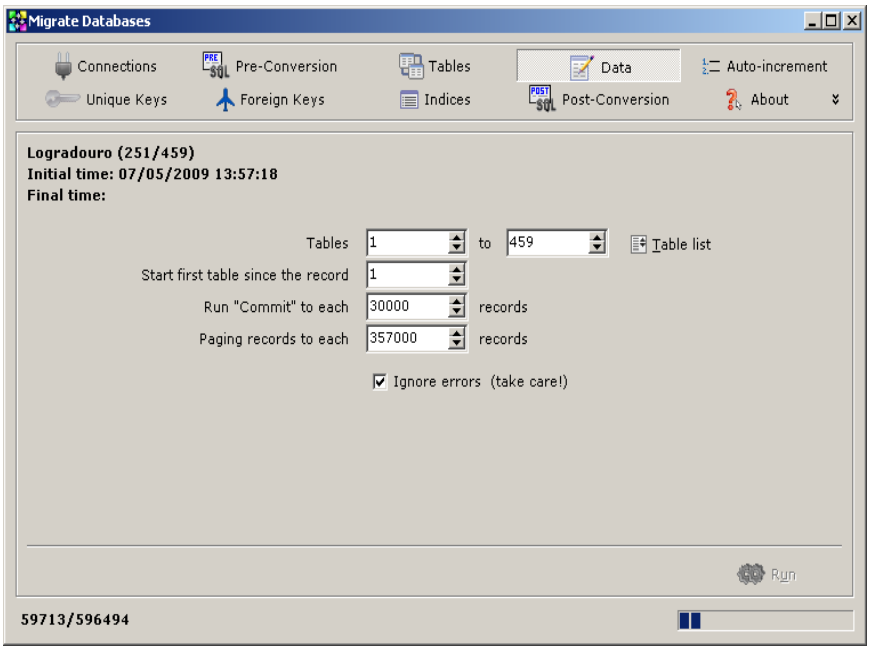


Figura 67 : Migração dos dados em andamento.

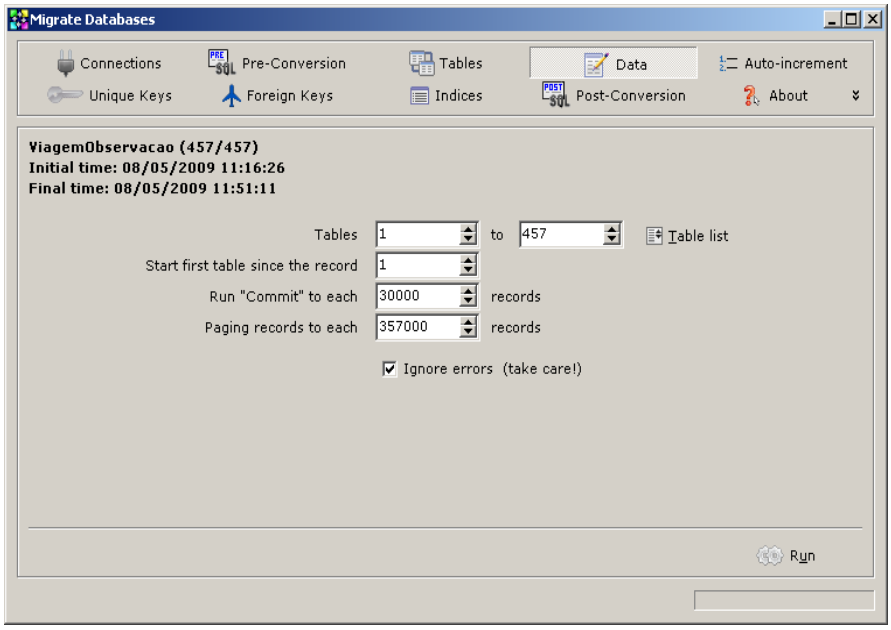


Figura 68 : Migração concluída

ANEXO D : SGDBS

Um sistema de banco de dados é basicamente apenas um sistema computadorizado de manutenção de registros. O banco de dados, por si só, pode ser considerado como o equivalente eletrônico de um armário de arquivamento; ou seja, ele é um repositório para uma coleção de arquivos de dados computadorizados. Os usuários de um sistema podem realizar (ou melhor, solicitar que o sistema realize) diversas operações envolvendo tais arquivos (DATE, 2004).

Segundo Date (2004) um SGDB possui diversas vantagens em relação aos métodos comuns de acesso a arquivos, por exemplo a sua densidade, velocidade, atualidade e proteção. Essas vantagens, são ainda maiores quando o ambiente for de multiusuários, no qual o banco de dados será muito maior e mais complexo, tornando-o muito mais útil.

A tecnologia de banco de dados cliente/servidor é mais amplamente usada em aplicações de regras de negócios. Sempre e em qualquer situação que você necessitar de um repositório de dados no qual armazenar, recuperar, consultar e analisar informações. Essas situações podem ocorrer no gerenciamento do relacionamento com o cliente (CRM), no planejamento de recursos das empresas (ERP), no suporte de marketing e vendas (do simples gerenciamento do contato ao sofisticado gerenciamento de lista), no suporte técnico (uma função discutível do CRM), na produção, nos serviços financeiros, na pesquisa e assim por diante. Os bancos de dados não armazenam eles mesmos os dados. Em vez disso, os dados são armazenados em estruturas tabuladas e essas tabelas são armazenadas dentro dos limites do arquivo do banco de dados, em várias estruturas sofisticadas de arquivos (SHAPIRO, 2002).

Geralmente, todos os *softwares* existentes no mundo necessitam de um lugar para armazenar seus dados. Esses dados possuem uma vida útil longa, podendo ficar anos ou décadas armazenados. No entanto eles estão sendo constantemente atualizados e, além disso, são constantemente requisitados pelas aplicações que se conectam ao banco.

Esses fatores mostram a importância que um banco de dados tem para uma aplicação, um indivíduo ou organização. As informações em questão podem ser qualquer coisa que tenha algum significado ao indivíduo ou à organização a que o sistema deve servir – ou seja, qualquer coisa que seja necessária para auxiliar no processo geral das atividades desse individuo ou dessa organização (DATE, 2004).

Como consequência dessa importância, alguns componentes devem interagir corretamente para que um sistema de banco de dados funcione corretamente, conforme Date (2004), são eles: dados, *hardware*, *software* e usuários.

Na próxima seção serão descritas as funcionalidades de um SGDB, o acesso, a integridade, a disponibilidade dos dados e uma visão sobre a linguagem padrão para acessar os dados (SQL).

D.1 O que é SGDB?

Entre o banco de dados físico e os usuários do sistema existe uma camada de software, conhecida como Sistema Gerenciador de Banco de Dados (SGBD). Todas as requisições de acesso ao banco de dados são tratadas pelo SGBD (DATE, 2004)

Partindo de um nível conceitual, o SGDB é o suporte para todas as tabelas, índices, restrições de integridade e outros objetos e propriedades usados para gerenciar os dados (SHAPIRO, 2002)

Segundo DATE (2004) um SGDB funciona conceitualmente da seguinte forma:

1. O DBA faz um pedido de acesso usando uma determinada sub linguagem de dados (geralmente SQL);
2. O SGBD intercepta o pedido e o analisa;
3. O SGBD, por sua vez, inspeciona o esquema externo para esse usuário, o mapeamento externo/conceitual correspondente, o esquema conceitual, o mapeamento conceitual/interno e a definição do banco de dados armazenado;
4. O SGBD executa as operações necessárias sobre o banco de dados armazenado.

O mapeamento conceitual/interno define a correspondência entre a visão conceitual e o banco de dados armazenado. Ele especifica o modo como os registros e campos conceituais são representados no nível interno. O Mapeamento externo/conceitual define a correspondência entre uma visão externa especifica e a visão conceitual. As diferenças existentes nesse mapeamento são semelhantes a existentes no mapeamento interno, permitindo que campos possam ter diferentes tipos de dados, nomes de campos possam ser alterados, vários campos conceituais possam ser combinados em um único campo externo (virtual), qualquer grupo de usuário possa compartilhar urna determinada visão externa (DATE,2004).

Como desvantagens de utilização de um SGDB, podemos citar o alto investimento

inicial na compra de *software* e *hardware* adicionais, generalidades essas que um SGBD fornece na definição e processamento de dados, também a sobrecarga na provisão de controle de segurança, controle de concorrência, recuperação e integração de funções.

Conforme Shapiro(2002) as principais vantagens de um SGBD são:a) Acesso simultâneo aos dados; b) Integridade dos dados; c) Disponibilidade dos dados. Essas vantagens serão explicadas a seguir:

D.1.1 Acesso simultâneo aos dados

O acesso simultâneo e o compartilhamento de bancos de dados entre muito usuários e processos, é, desse modo, uma consideração importante, se não a mais importante, quando você estiver projetando e criando um software ou aplicativos de bancos de dados. Nos bancos de dados cliente/servidor, somente o *engine* do servidor de bancos de dados controla o acesso aos dados, executa as consultas, gerencia o banco de dados e as solicitações de cada cliente e mantém o servidor do banco de dados operando da melhor maneira possível (SHAPIRO, 2002).

Um dos critérios de classificação de um SGDB refere-se ao número de usuários que podem usa o sistema concorrentemente. Um SGDB será monousuário se somente um usuário de cada vez puder usar o sistema, e será multiusuário se muitos usuários puderem usá-lo concorrentemente (ELMASRI & NAVATHE, 2005).

D.1.2 Integridade dos dados

Em um SGDB relacional, normalmente existirão muitas relações, e as tuplas dessas relações estão relacionadas de várias maneiras. Há muitas restrições para os valores reais em um estado do banco de dados (ELMASRI & NAVATHE, 2005).

Um banco de dados cliente/servidor, é muito mais comprometido em assegurar a validade dos dados, além de assegurar que o que entrar no banco de dados será ao aceitável quando possível. Quanto maiores se tornam os bancos de dados e mais complexos os modelos de dados, mais vulneráveis são os dados. Portanto, é vital usar tecnologia que foi projetada especialmente para garantir absolutamente o cumprimento das regras de integridade quando necessárias, principalmente onde sua vida e a de outros depender dela (SHAPIRO, 2003).

D.1.3 Disponibilidade dos dados

Não se passa um segundo em nossas vidas sem que os dados não sejam críticos. Está se tornando a cada dia mais difícil viver em nosso mundo conectado sem constantemente acessar bancos de dados e informações precisas. (SHAPIRO,2003) Os bancos de dados se tornaram componentes essenciais no cotidiano da sociedade moderna. No decorrer do dia, a maioria de nós se depara com atividades que envolvem alguma interação com os bancos de dados (ELMASRI & NAVATHE, 2005).

As empresas, pessoas processos, a própria vida, tudo dependa da disponibilidade dos dados diariamente. Enquanto tanto o acesso aos dados quanto sua integridade podem eles mesmos ser julgados componentes da disponibilidade, o mais importante é estar *on-line*, estar lá quando precisamos; responder em um período de tempo razoável (SHAPIRO, 2003).

Um SGDB deve permite que diversos usuários acessem o banco de dados ao mesmo tempo. Isso é essencial se os dados para várias aplicações estão integrados e mantidos em um único banco dedados (ELMASRI & NAVATHE, 2005).

D.2 SQL

O SQL (*Structured Query Language*) é a linguagem padrão para acessar dados num sistema gerenciador de banco de dados relacional. Ela foi criada originalmente no início dos anos 80, pela IBM. Em meados de 1908, o ANSI (*American Nacional Standards Institute* – Instituto Americano de Padrões) iniciou seus trabalhos de desenvolvimento para a padronização de uma linguagem para bancos de dados relacionais. O ANSI e a ISSO publicaram a primeira padronização do SQL em 1986 e 1987 respectivamente, e evoluiu conforme tabela abaixo: (PEREIRA NETO, 2003).

Tabela 39 : Evolução e padronização da Linguagem SQL.
Fonte: (PEREIRA NETO, 2003) e (ELMASRI & NAVATHE, 2005).

Padrão	**Ano**	**Evolução da Padronização SQL**
SQL/86	De 1986	Recebeu integralmente o dialeto SQL proposto e criado pela IBM;
SQL/89	De 1990	Adição de integridade referencial (relacionamento entre tabelas) e ainda o *Embedded* SQL (códigos embutidos SQL para pré-compilação) para utilização com linguagens C,

		ADA, COBOL, etc;
SQL/92	De 1992	Inclusão de E*mbedded* SQL, catálogos de sistema, esquemas, domínios, ampliação do conjunto de tipos de dados e conversões de tipos, chamadas de *cast*;
SQL/99	De 1999	Extensão das capacidade relacionais (como as *triggers* – procedimentos internos ao banco de dados relacionais disparados por eventos) e suporte a características de orientação a objetos (como os tipos de dados definidos pelo usuário e ainda a herança). Este padrão , como exceção dos *triggers*, ainda é muito pouco implementado pelos gerenciados de bancos de dados relacionais da atualidade;
SQL/03	De 2003	Introduz características relacionadas ao XML, sequências padronizadas e colunas com valores de auto-generalização (inclusive colunas-identidade).

ANEXO E : FUNÇÕES SQLS

A proposta do trabalho era em utilizar o máximo recurso dos componentes do Borland Delphi 7 e também o padrão de SQL. Porém algumas funções são tão específicas que para solucionar esses problemas sem perder a qualidade do *software*, deve-se recorrer a algumas funções.

No projeto, apenas duas consistências em alguns casos não puderam ser tratadas. A primeira refere-se a concatenação de *strings* nas SQLs, no Microsoft SQL Server usa-se o sinal de mais (+) para unir esses valores, já no PostgreSQL usa-se o *pipe* (|), para isso foi utilizado a função do SQL Server *StrCat* que une duas *strings*, isso resolveu o problema no SQL Server, porém no PostgreSQL foi necessário criar essa função, para solucionar o problema.

A segunda questão foi quanto as SQLs de extração de dias, meses e anos com agrupamento. Como o sistema inicialmente foi criado utilizando o SGDB SQL Server, foi mantido esse padrão, e apenas criado as funções necessárias no PostgreSQL, nesse caso as funções *day*, *month* e *year*.

Na Figura 67, é demonstrado o *script* de criação dessas funções no SGDB PostgreSQL. Essas funções foram retiradas do site http://powergres.sraoss.co.jp/s/ja/tech/plus/experience/vol11/samples/mssqlcomp.sql mantido pelo grupo Japonês PowerGres.

```
-- PowerGres

-- This file emulates some common MS SQL Server functions in PostgreSQL

-- maps getdate() to now()
create or replace function getdate() returns timestamptz as '
begin
return now();
end;
' language 'plpgsql';

-- maps isnull() to coalesce()
create or replace function isnull(anyelement, anyelement) returns anyelement as '
begin
return coalesce($1,$2);
end;
' language 'plpgsql' immutable;

-- This allows the use of "+" when joining strings.
create or replace function strcat(text, text) returns text as '
begin
return $1 || $2;
end;
' language 'plpgsql' immutable;
create operator + (procedure = strcat, leftarg = text, rightarg = text);

-- these simulate day, month, and year functions in T-SQL
-- day function for each date/time type.
```

```
create or replace function day(timestamptz) returns int as '
begin
return extract(day from $1);
end;
' language 'plpgsql' immutable;

create or replace function day(timestamp) returns int as '
begin
return extract(day from $1);
end;
' language 'plpgsql' immutable;

-- month function for each date/time type.
create or replace function month(timestamptz) returns int as '
begin
return extract(month from $1);
end;
' language 'plpgsql' immutable;

create or replace function month(timestamp) returns int as '
begin
return extract(month from $1);
end;
' language 'plpgsql' immutable;

-- year function for each date/time type.
create or replace function year(timestamptz) returns int as '
begin
return extract(year from $1);
end;
' language 'plpgsql' immutable;

create or replace function year(timestamp) returns int as '
begin
return extract(year from $1);
end;
' language 'plpgsql' immutable;

-- emulate ms sql string functions.
create or replace function space(integer) returns text as '
begin
return repeat('' '', $1);
end;
' language 'plpgsql' immutable;

create or replace function charindex(text, text) returns int as '
begin
return position($1 in $2);
end;
' language 'plpgsql';

create or replace function len(text) returns int as '
begin
return char_length($1);
end;
' language 'plpgsql';

create or replace function left(text, int) returns text as '
begin
return substr($1, 0, $2);
end;
' language 'plpgsql';

-- a function to allow mailing. It is currently a wrapper that allows
-- this functionality to be added later.
-- create or replace function xp_sendmail(tofield text, message text, subject text) returns int as '
-- declare
-- begin
-- return 0;
-- end;
```

```
-- ' language 'plpgsql';

-- these functions provide casts from timestamps to ints (# of days). Must be created as super-user.
create or replace function mscomp_int4(interval) returns int4 as '
begin
return extract(day from $1);
end;
' language 'plpgsql' immutable;

create cast (interval as int4) with function mscomp_int4(interval);
create or replace function mscomp_int4(timestamptz) returns int4 as '
begin
return mscomp_int4($1 - ''1/1/1900'');
end;
' language 'plpgsql' immutable;

create cast (timestamptz as int4) with function mscomp_int4
(timestamptz);
create or replace function mscomp_int4(timestamp) returns int4 as '
begin
return mscomp_int4($1 - ''1/1/1900'');
end;
' language 'plpgsql' immutable;

create cast (timestamp as int4) with function mscomp_int4(timestamp);

create or replace function mscomp_float(interval) returns float as '
begin
return (extract(epoch from $1) / 86400);
end;
' language 'plpgsql' immutable;

create cast (interval as float) with function mscomp_float(interval);

create or replace function mscomp_float(timestamptz) returns float as '
begin
return mscomp_float($1 - ''1/1/1900'');
end;
' language 'plpgsql' immutable;

create cast (timestamptz as float) with function mscomp_float(timestamptz);

create or replace function mscomp_float(timestamp) returns float as '
begin
return mscomp_float($1 - ''1/1/1900'');
end;
' language 'plpgsql' immutable;

create cast (timestamp as float) with function mscomp_float(timestamp);
```

Figura 69 : Funções do SQL Server para o PostgreSQL.