UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA

CURSO CIÊNCIA DA COMPUTAÇÃO
TURMA 98.2

ADMINISTRAÇÃO DE SGBD:
UM ESTUDO DE CASO

Autor

Rafael Donato Azevedo Loureiro (rdal@cin.ufpe.br)

Orientador

Prof. Fernando da Fonseca de Souza, PhD.

Recife, Agosto 2005

Agradecimentos

A André Lima, Eduardo Oliveira e Ivo Frazão da Cemicro Informática, pela disponibilização de informações relevantes à realização deste trabalho, bem como ao apreço e amizade dedicados a mim;

Ao meu orientador, Professor Fernando Fonseca, pelo apoio e orientação nesta monografia;

A Sabrina, pelo amor, paciência e carinho com que tem me tratado diariamente;

Aos meus pais, que torcem pelo meu crescimento pessoal e são responsáveis pela minha vida, a qual sou grato todos os dias.

Administração de SGBD: um estudo de caso

Resumo

Será abordado neste trabalho de graduação, algumas técnicas de administração para garantir estabilidade, segurança e performance ao sistema de gerenciamento de banco de dados (SGBD) PostgreSQL. Serão discutidas técnicas de melhora de performance de hardware, escolha do melhor sistema de arquivos para rodar o SGBD, e será falado também sobre os arquivos de configuração do SGBD, como configurar esses arquivos de modo a conseguir um bom resultado final. Será feito também um estudo comparativo de algumas ferramentas de administração do SGBD, destacando as vantagens e desvantagens de cada uma.

DBMS administration: a study case

Abstract

This graduation project talks about some techniques of administration to guarantee stability, security and performance to the PostgreSQL Database Management System (DBMS). Techniques of hardware performance improvement, choice of the best file systems to have the DBMS running fine, and also the configuration files of the DBMS, in order to obtain a good final result. A comparative study of some administration tools of the DBMS will be made, detaching the advantages and disadvantages of each one.

ÍNDICE

Lista de Figuras

1 Introdução

Os dados corporativos são os maiores patrimônios de uma organização, já que, sem eles, muitas atividades relacionadas com os negócios deixam de ser realizadas. Como eles são armazenados em equipamentos de informática e manipulados por software de banco de dados, a administração do SGBD (Sistema de Gerenciamento de Banco de Dados) é de suma importância para as organizações.

Software livre é um conceito de extrema importância no mundo da computação. De forma básica, quando um software é livre, significa que seu código-fonte está disponível para qualquer um e você pode alterá-lo para adequá-lo às suas necessidades, sem ter de pagar.

O PostgreSQL é um sistema gerenciador de banco de dados orientado a objetos, de livre distribuição e código-fonte aberto. Por ser um banco de dados livre com inúmeros recursos, ele está se firmando no mercado de informática e a procura por profissionais que dominem esta tecnologia é grande.

Assim, este trabalho abordará diversos assuntos relacionados à Administração do Sistema de Gerenciamento de Banco de Dados PostgreSQL[15], tais como boas práticas, testes de performance, melhora de performance de hardware, comparação entre algumas ferramentas de

administração, entre outros.

O capítulo 2 deste documento aborda a administração de SGBD e descreve três das principais ferramentas para administração do SGBD PostgreSQL. Serão discutidos os prós e contras de cada ferramenta, destacando as funcionalidades de cada uma e por fim a indicação de uma ferramenta na qual se possa obter um maior ganho de produtividade e praticidade, mostrando minuciosamente algumas de suas funcionalidades e como utilizá-las.

No capítulo 3 serão descritos testes de performance no SGBD. Primeiramente sem nenhuma otimização e depois será aplicada uma série de otimizações no sistema, de modo a demonstrar o ganho de performance obtido pelo SGBD após as melhorias.

No Capítulo 4 será mostrado como as otimizações tratadas no Capítulo 3 são feitas no próprio hardware da máquina e como as escolhas iniciais das partições do servidor podem influir na performance de um servidor de banco de dados.

No Capítulo 5 serão apresentadas as conclusões e sugestões de trabalhos futuros e, por fim, serão descritas as referências bibliográficas utilizadas.

2 Administração de SGBD

Administrar um SGBD é, de maneira simplista, instalar, configurar, monitorar e solucionar problemas do sistema. Esmiuçando este conceito, um Administrador de Banco de Dados tem as seguintes responsabilidades:

- Projeto lógico do banco de dados;
- Definição de verificação de segurança e integridade;
- Decisão de como os dados são representados na base de dados armazenada;
- Projeto físico da base de dados;
- Definição de procedimentos de recuperação;
- Monitoração do desempenho;
- Contato com usuários para averiguação de disponibilidade dos dados por eles requisitados e ajuda na determinação e resolução de problemas;
- Ajustes apropriados à medida que ocorram mudanças de requisitos

Para facilitar a execução das tarefas do administrador, existe uma série de ferramentas de administração que podem auxiliar em muito, aumentando a produtividade e garantindo a segurança e a consistência dos SGBD. Essas ferramentas provêm muitas funcionalidades de alteração

nos bancos de dados, geração de relatórios, exportação de dados, scripts de *backup* que seria um lapso da parte do administrador não optar pelo uso de alguma dessas ferramentas. A seguir serão mostrados alguns detalhes de algumas das mais populares ferramentas para administração do SGBD PostgreSQL: phpPgAdmin[6], EMS PostgreSQL Manager[20] e Pgadmin[14].

2.1 Ferramenta phpPgAdmin

O phpPgAdmin é considerado completo, como se pode ver por suas funcionalidades:

- Administra múltiplos servidores;
- Oferece suporte a PostgreSQL 7.0.x, 7.1.x, 7.2.x, 7.3.x, 7.4.x, 8.0.x;
- Gerencia todos os aspectos sobre:
  - Usuários e grupos;
  - Bases de dados;
  - Esquemas;
  - Tabelas, índices, restrições, *triggers*, regras e privilégios;
  - *Views*, seqüências e funções;
  - Objetos avançados;
  - Relatórios.

- Fácil Manipulação de dados:
    - Acessar tabela, *views* e relatórios;
- Executa comandos SQL arbitrários como Select, insert, update e delete;
- Executa *dump* dos dados em vários formatos: SQL, COPY, XML,XHTML, CSV, Tabbed e pg_dump;
- Importa scripts SQL, dados COPY, XML, CSV e Tabbed;
- Excelente suporte de línguas
    - Disponível em 26 línguas;
    - Sem conflitos. Edita dados em russo usando a interface em japonês, por exemplo;
- Fácil de instalar e configurar.

O programa é um software livre e pode-se redistribuir e/ou modificá-lo sob os termos da GNU General Public License[7], assim como foi publicado pela Free Software Foundation [24].

Recomendação inicial de instalação: PHP 4.1 [25] ou mais novo, porém 4.3 em diante oferece mais funcionalidades. A recomendação de hardware não é relevante uma vez que o servidor PHP já deva estar instalado na máquina, e quem instalou seguiu as recomendações mínimas para instalação do serviço PHP.

As restrições do phpPgAdmin se resumem às próprias restrições da web, cada ação carrega uma nova tela, sempre é necessário submeter

botões para editar dados, entre outras. A Figura 2.1 apresenta a tela principal do sistema.

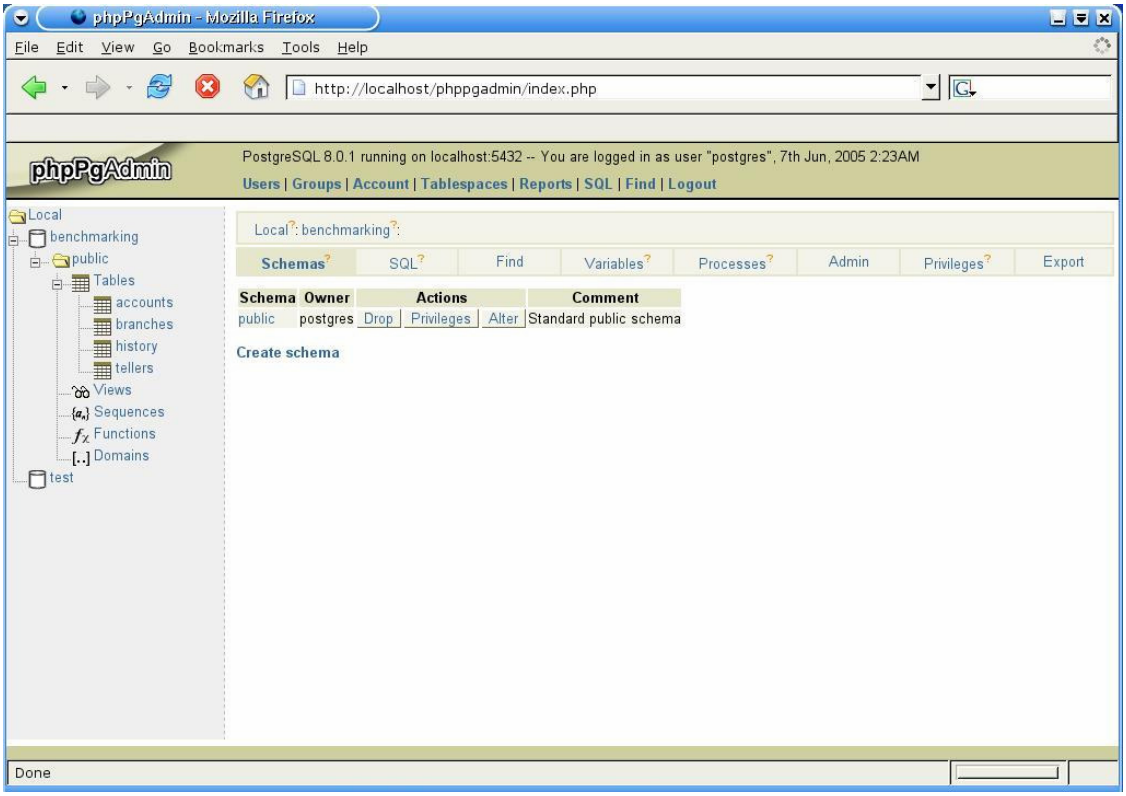

Figura 2.1 – Tela de phpPgAdmin

2.2 Ferramenta EMS PostgreSQL Manager

Também é considerada uma das melhores ferramentas para administração do PostgreSQL. Dá suporte completo ao PostgreSQL versão 8.0 ou acima, gerenciamento e navegação rápidos no SGBD, fácil gerenciamento de todos os objetos PostgreSQL, ferramentas avançadas de manipulação de dados, gerenciamento de segurança efetivo, visual

excelente e ferramentas de texto para construção de consultas, além de habilidades de exportação e importação de dados. A edição de dados é outro ponto forte nessa ferramenta que exibe todas as informações necessárias ao administrador e possui vários formatos de filtro. Detalhando essas características tem-se:

2.2.1 Suporte completo ao PostgreSQL versão 8.0 ou acima

- Mudar tipo das colunas;
- Suporte a Tablespaces;
- Suporte a nome de argumentos em funções;
- Suporte a *dollar-quoted strings;*

2.2.2 Interface gráfica

- Suporte aos esquemas visuais do Office 11 e Windows XP;
- Sistema *wizard* bem descrito;
- *Explorer* melhorado do banco de dados para facilitar o gerenciamento de todos os objetos do PostgreSQL;
- Barra de ferramentas para navegar entre as janelas facilmente;
- Barras de navegação com boas funcionalidades na maioria das janelas;
- Possibilidade de salvar todas as configurações do programa;
- Barras de ferramentas customizáveis para todas as janelas do programa;

- Interface personalizável completa do programa;
- Módulos de opções visuais poderosos;
- Interface localizadora do programa.

2.2.3 Gerenciamento e navegação rápidos no banco de dados

- Criar/destruir bancos e esquemas;
- Gerenciamento das tabelas e de todos os seus sub-objetos: campos, índices, *triggers*, regras, verificações e chaves estrangeiras;
- Gerenciamento de *views*, funções e seqüências;
- Gerenciamento de tipos, agregados, operadores, linguagens;
- Gerenciamento de regras de tabelas e de *views*;
- Duplicação de todos os objetos do SGBD;
- Renomear todos os objetos do SGBD;
- Função poderosa de *debbuging* PL/pgSQL;
- Visualização de dependências entre objetos do SGBD.

2.2.4 Ferramentas avançadas de manipulação de dados

- Visualizador/editor BLOB poderosos com diversos tipos de visualização de dados BLOB;
- Variedade de ferramentas úteis como agrupamento de dados, busca rápida e filtragem;

- Máximos detalhes de visualização para trabalhar com duas tabelas relacionadas entre si ao mesmo tempo;
- Visualização de tabelas e cartões para ver os dados do jeito que o usuário quiser;
- Capacidade de copiar e colar registros selecionados;
- Sistema de impressão avançado;
- Exportar dados para *SQL script* como *INSERT statement*.

2.2.5 Gerenciamento de segurança efetivo

- Gerenciador poderoso de usuários e grupos para administrar usuários e privilégios;
- Gerenciador de GRANT mostrando todos os objetos grant do SGBD em um *form* de um *grid*;

2.2.6 Visual excelente e ferramentas de texto

- *Query builder* visual, possibilitando ao administrador construir consultas complicadas sem ser necessário nenhum conhecimento da sintaxe SQL;
- Múltiplos editores SQL com *auto-complete* para códigos e *syntax highlight*;
- Larga execução de scripts (Editor de script SQL).

2.2.7 Criação de relatório usando o *report construction wizard*

- *Create Report Wizard* possibilita criar simples relatórios em alguns cliques do mouse;
- Gerenciamento de relatórios assim como se eles fossem objetos do SGBD: acessando relatórios diretamente da árvore do *DB Explorer*;

2.2.8 Impressionante capacidade de importação e exportação de dados

- Exportação de dados para os formatos mais populares: MS Excel, MS Word, HTML, PDF, TXT, CSV, DBF, XML, entre ouros;
- Exportação de dados para MS Access;
- Importação de dados a partir de MS Excel, DBF, TXT e CSV;
- Importação de dados a partir do MS Access;
- *Wizards* para copiar dados de/para arquivos em servidor.

2.2.9 Poderoso modelador visual de banco de dados

- Possibilita criar, editar e destruir tabelas e campos de tabelas, estabelece *links* entre tabelas visualmente;
- Engenharia reversa;
- Salvar como imagem ou imprimir diagrama do esquema de banco
- de dados.

2.2.10 Facilidade no uso de *wizards* que executam tarefas da manutenção do PostgreSQL

- Analisando tabelas;
- Executando *vacuum* (ver seção 3.1);
- Executando *cluster* nas tabelas;
- Reindexando tabelas;
- Visualizador de *logs* do servidor.

2.2.11 Outras ferramentas úteis que tornam o trabalho com o servidor PostgreSQL o mais fácil possível

- Extração de metadados de arquivos texto;
- Poderoso módulo de impressão de metadados para criação de relatórios de matadados personalizáveis;
- Modelagem de relatórios para construir poderosos relatórios visualmente;
- Monitor SQL;
- *Wizard* para relatório HTML para criar um relatório HTML detalhado sobre o SGBD rapidamente

2.2.12 Outras características úteis

- Disponível para Windows e Linux;
- Templates de teclado;

- Lista *to-do*;
- Ferramenta de gerenciamento externo.

Uma característica que essa ferramenta não oferece é a capacidade de criação de diagramas sem alterar o banco de dados correpondente. Outra limitação diz respeito à geração de *backups*:a ferramenta EMS não leva em consideração a ordem de criação de chaves estrangeiras, podendo gerar problemas na hora de restaurar o *backup*.

O EMS é uma ferramenta comercial e de código fechado. Na figura 2.2 é mostrado um *screenshot* da ferramenta EMS.

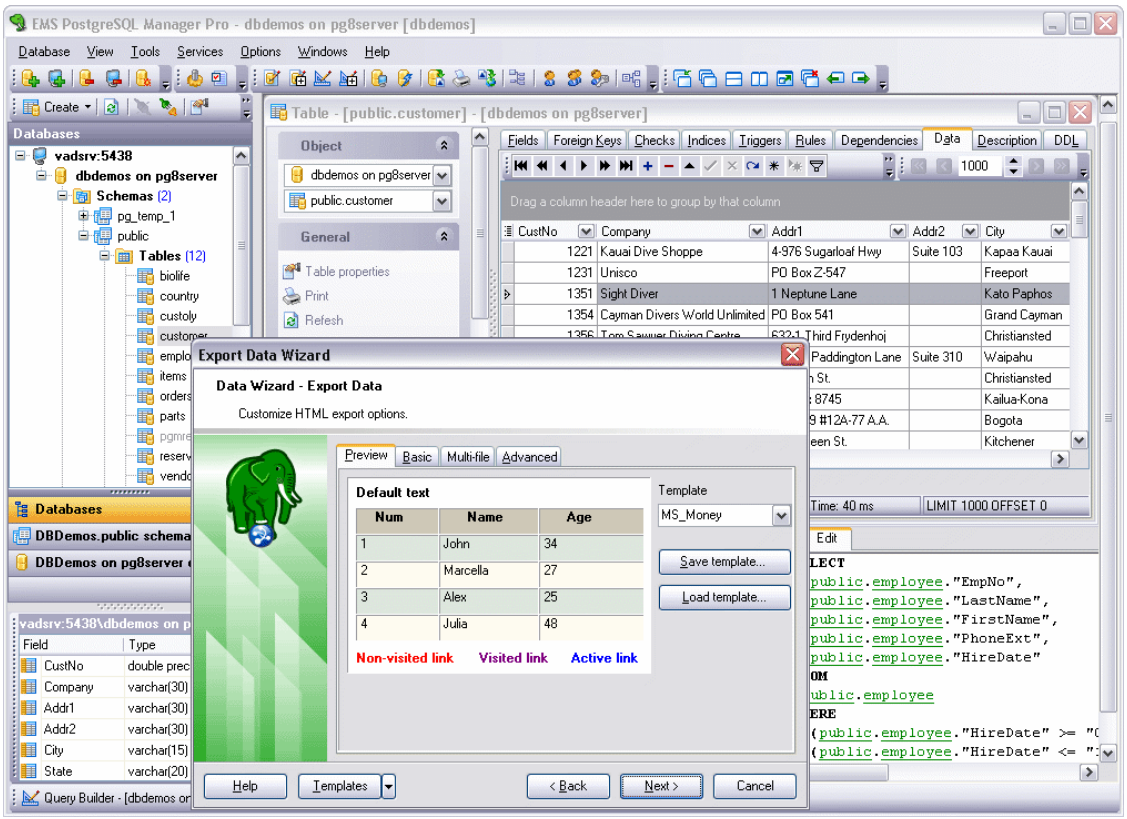

Figura 2.2 – Tela do EMS PostgreSQL Manager

2.3 Ferramenta PGAdmin

O PGAdmin é uma ferramenta software livre com o código disponível para download[14]. Ela oferece suporte multiplataforma (MS Windows 2000 e XP, GNU/Linux e FreeBSD) e versões para MacOsX e SunOS estão sendo desenvolvidas. Oferece suporte às versões mais recentes do PostgreSQL, possui acesso a todos os objetos PostgreSQL, além de acesso nativo PostgreSQL. É considerada uma ferramenta limitada, pois o PGAdmin não permite filtros, não se pode alterar os dados no mesmo lugar em que eles são exibidos, e abre muitas janelas desnecessariamente. A ferramenta não oferece geração de diagramas e também não gera *scripts* de *backup*. Outra parte fraca dessa ferramenta é a exportação de dados. Abaixo um maior detalhamento das características dessa ferramenta:

2.3.1 Multiplataforma

- MS Windows 2000 & XP;
- GNU/Linux;
- FreeBSD.

2.3.2 Desenvolvidos para as versões mais recentes do PostgreSQL

- PostgreSQL 7.3;

- PostgreSQL 7.4;
- PostgreSQL 8.0.

2.3.3 Help On-line

- Documentação PostgreSQL (on-line e off-line);
- Guia de comandos SQL.

2.3.4 Interface multilíngüe

- A interface de usuário do PgAdmin III é traduzida para mais de 30 línguas.

2.3.5 Acesso a dados

- Acesso nativo ao PostgreSQL (ODBC não é necessário);
- Poderosa ferramenta de consulta com *syntax highlight*;
- *Datagrid* bastante rápido para consulta e inserção de dados.

2.3.6 Acesso a todos os objetos PostgreSQL

- Objetos são mostrados com sua definição SQL
    - Agregados
    - *Casts*

- Colunas
- Restrições
- Conversões
- Bases de dados
- Domínios
- Funções
- Grupos
- Índices
- *Tablespaces*
- Linguagens *server-side* (como PLpgsql, PLpython, PLperl, etc...)
- Classes operadoras
- Operadores
- Servidores PostgreSQL
- Regras
- Esquemas
- Seqüências
- Tabelas
- Funções de *trigger*
- Tipos
- Usuários
- *Views*

2.3.7 Suporte Multibyte

- PgAdmin III dá suporte à maioria dos *server-side encodings* do PostgreSQL

    - SQL_ASCII
    - EUC_JP, EUC_CN, EUC_KR, EUC_TW
    - JOHAB
    - LATIN1, LATIN2, LATIN3, LATIN4, LATIN5, LATIN6, LATIN7, LATIN8, LATIN9, LATIN10
    - ISO_8859_5, ISO_8859_6, ISO_8859_7, ISO_8859_8
    - UNICODE UTF-8
    - MULE_INTERNAL
    - KOI8
    - WIN
    - ALT
    - WIN1256
    - TCVN
    - WIN874

    A figura 2.3 mostra um *screenshot* na ferramenta PgAdmin.

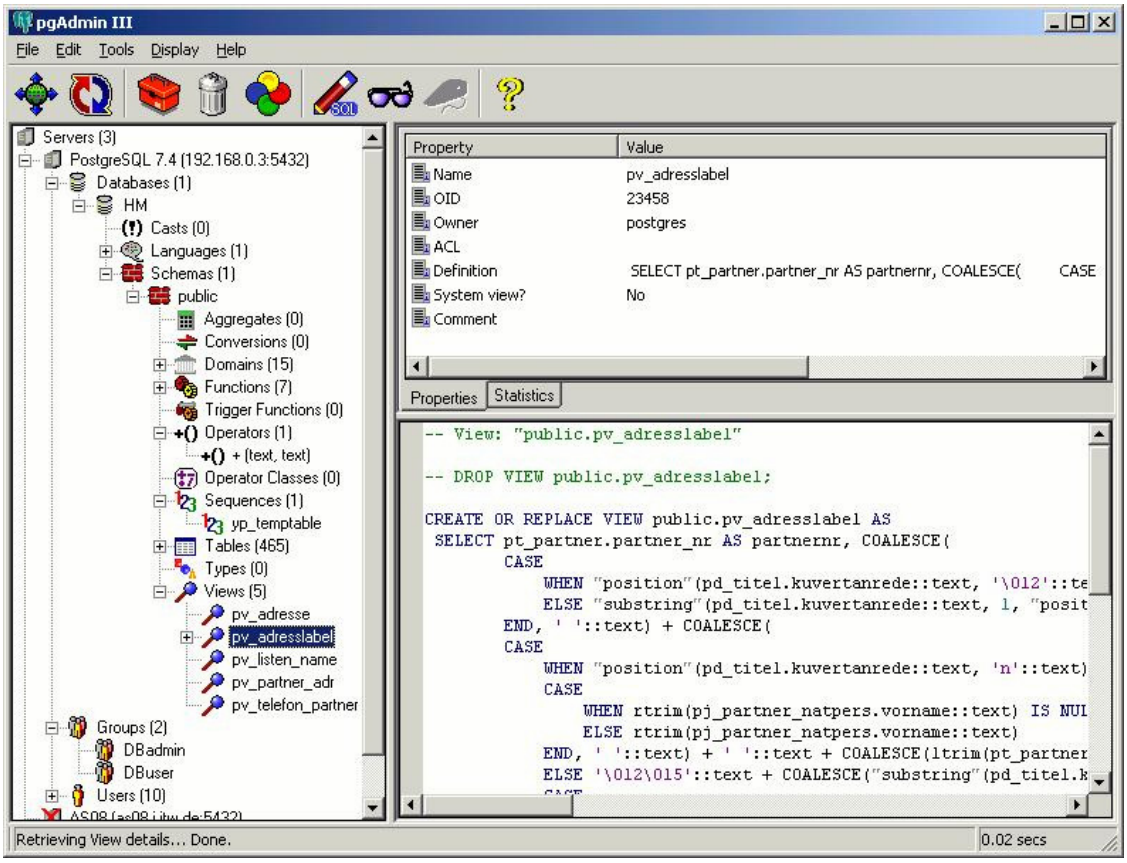

Figura 2.3 – Tela do PgAdmin

2.4 Quadro Comparativo

O quadro 2.1 apresenta um estudo comparativo entre as ferramentas phpPgAdmin, EMS PostgreSQL Manager, e PgAdmin, salientando os pontos positivos e nagativos de cada uma.

Quadro 2.1 - Comparativo entre as ferramentas de administração de SGBD PostgreSQL

	phpPgAdmin	EMS	PgAdmin III
Administra Múltiplos Servidores	Sim	Sim	Sim
Oferece suporte a Tablespaces	Sim (a partir da versão 3.5)	Sim	Não
Auto-complete e hints	Não	Sim	Sim
Barras de ferramentas customizáveis para todas as janelas do programa	Não	Sim	Não
Grátis	Sim	Não	Sim
Open-source	Sim	Não	Sim
Vários formatos de filtro	Não	Sim	Não
Geração de scripts de backup	Não	Sim	Não
Importação e exportação de dados em vários formatos	Sim	Sim	Não
Suporte a várias línguas	Sim	Sim	Sim
Multiplataforma	Sim	Sim	Sim
Geração de relatórios	Sim	Sim	Não
Visualizador/editor BLOB	Não	Sim	Não

Suporte ao PostgreSQL 8.0.x	Sim	Sim	Sim (a partir da versão 1.2.x)
Executa o vacuum	Sim	Sim	Sim

2.5 A escolha: EMS PostgreSQL Manager

Como foi visto no quadro comparativo acima, a ferramenta **EMS PostgreSQL Manager** é a ferramenta que oferece mais possibilidades ao administrador do SGBD. Faz tudo o que as outras fazem e ainda oferece várias funcionalidades que as outras não oferecem. Embora seja uma ferramenta paga, esta é a ferramenta mais recomendável para a administração de um SGBD PostgreSQL. Serão detalhadas abaixo algumas das principais funcionalidades dessa ferramenta.

2.5.1 Cadastrando uma Base de Dados

Para cadastrar um novo banco, clicar no menu Database, ou pressionar *Shift+Alt+R*, depois especificar o *host* no qual se quer conectar, a porta a ser usada na conexão, o usuário e senha. Então escolhe qual banco se quer cadastrar e qual o *alias* que se quer atribuir a ele. A figura 2.4 mostra como acessar o menu “Cadastrar database”.

Depois é mostrado na figura 2.5 o formulário que deverá ser preenchidas as informações necessárias sobre o novo banco, tais como *host*, porta, nome de usuário, senha, entre outros.

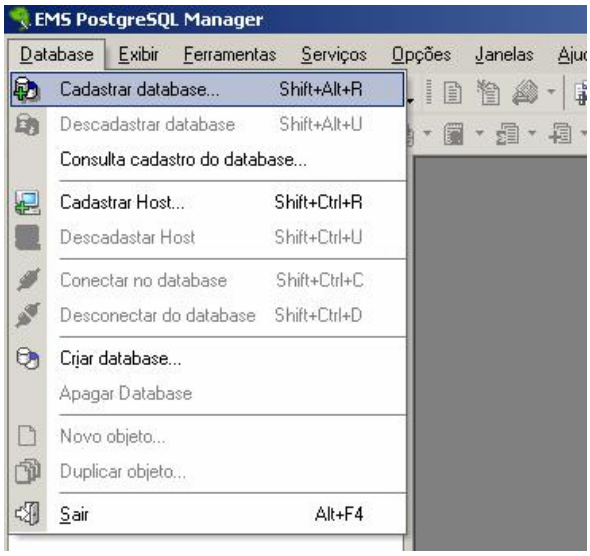

Figura 2.4 – Menu no EMS, escolha para cadastrar um novo banco

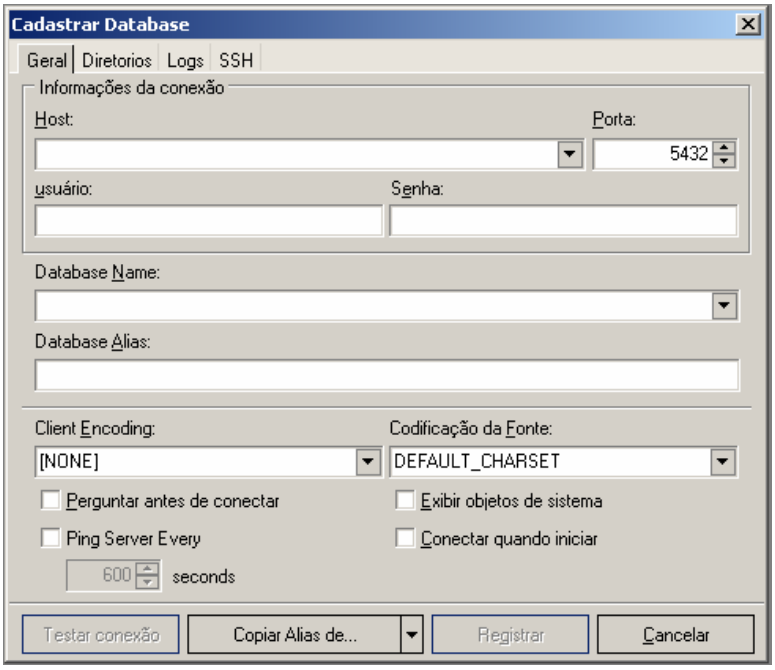

Figura 0.5 – Tela de cadastro com as informações do novo banco

2.5.2 Selecionar uma tabela

Para selecionar uma tabela, basta clicar com o mouse em cima de seu nome no menu do lado esquerdo, como mostrado na figura 2.6, que suas propriedades irão aparecer.

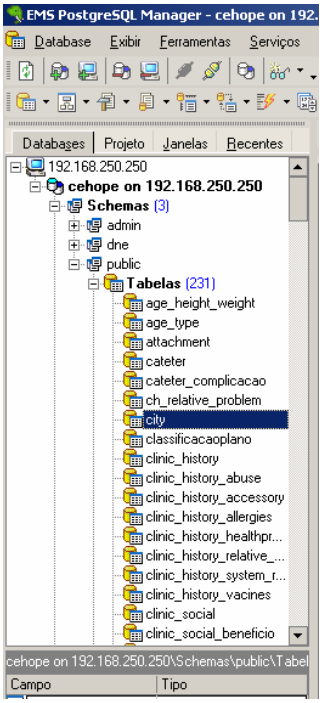

Figura 2.6 – Tabelas existentes no banco de dados

2.5.3 Aplicação de filtros

Uma forma de aplicação de filtros é com a tabela, que se deseja aplicar o filtro, selecionada, selecionar a aba Dados, e clicar no símbolo do funil, (figura 2.7), e então aplicar os filtros desejados como mostra a figura 2.8. Depois de clicar em OK, os valores filtrados irão aparecer no *grid*, (figura 2.9).

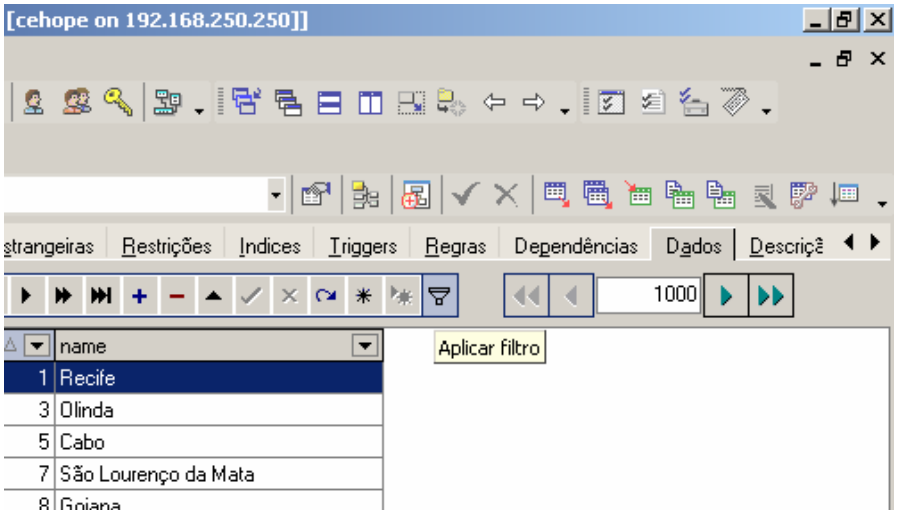

Figura 2.7 – Botão em forma de funil para aplicar filtro

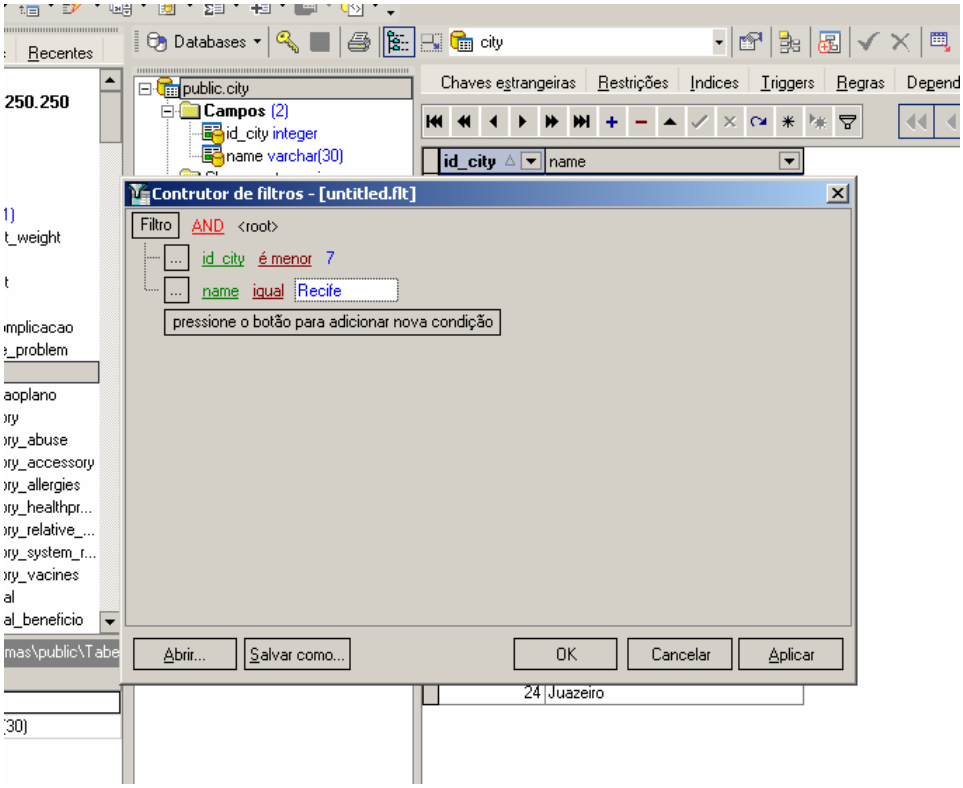

Figura 2.8 – Escolha dos filtros

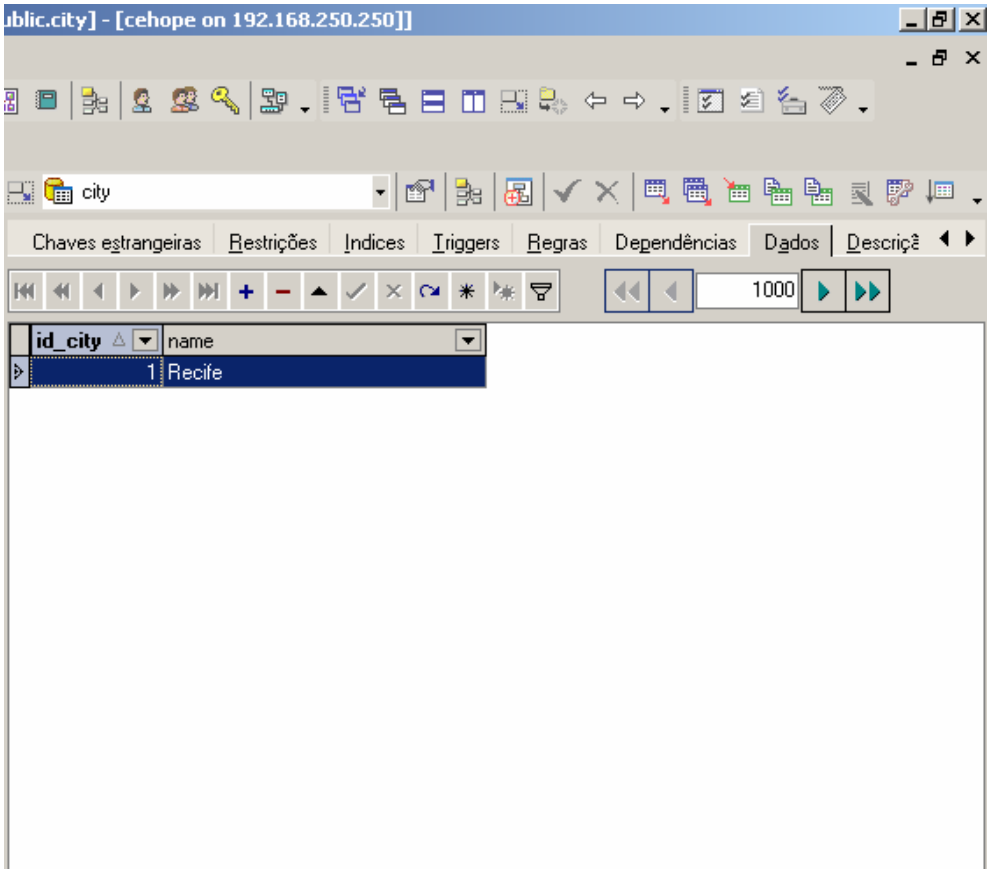

Figura 2.9 – Dados filtrados

2.5.4 Criação de relacionamentos

Para criar um diagrama visual de banco de dados, clicar em Ferramentas, mostrado na figura 2.10, e depois em *Visual Database Designer*. Com as tabelas aparecendo do lado direito como mostra a figura 2.11, basta arrastar as tabelas desejadas para dentro do espaço reservado para criar o diagrama.

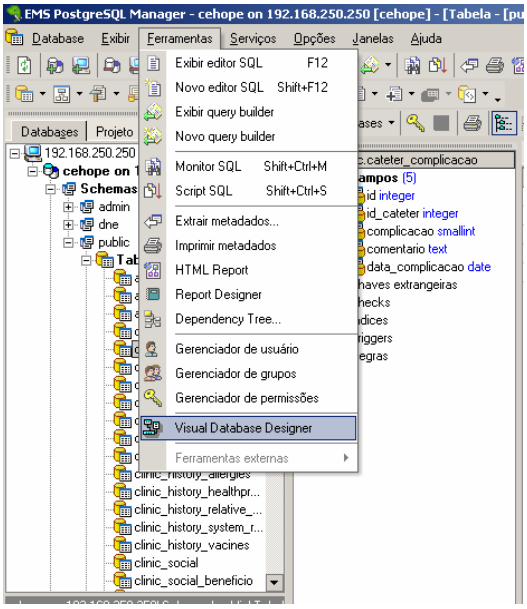

Figura 2.10 – Menu para acessar o Visual Database Designer

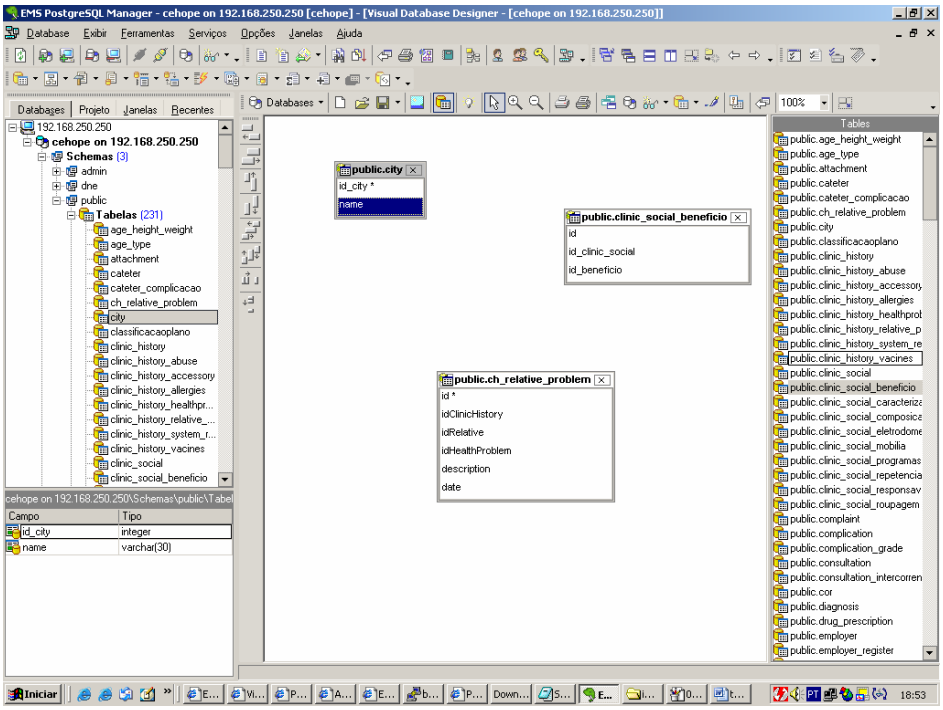

Figura 2.11 – Entidades de relacionamentos

2.5.5 Query Builder

Para fazer consultas ou alterações diretamente no banco através de linha de comando, pode-se usar o *query builder*. Para isso é necessário clicar em Ferramentas e depois em Exibir editor SQL (figura 2.12). O *query builder* possui *Syntax Highlighting* e *Autocomplete*, mostrado na figura 2.13 abaixo.

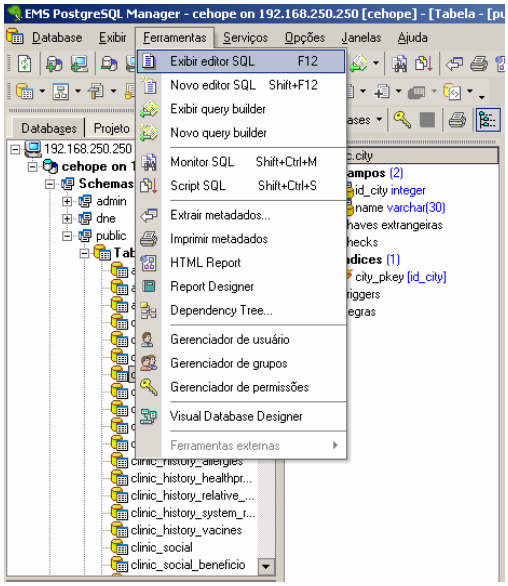

Figura 2.12 – Menu do editor SQL

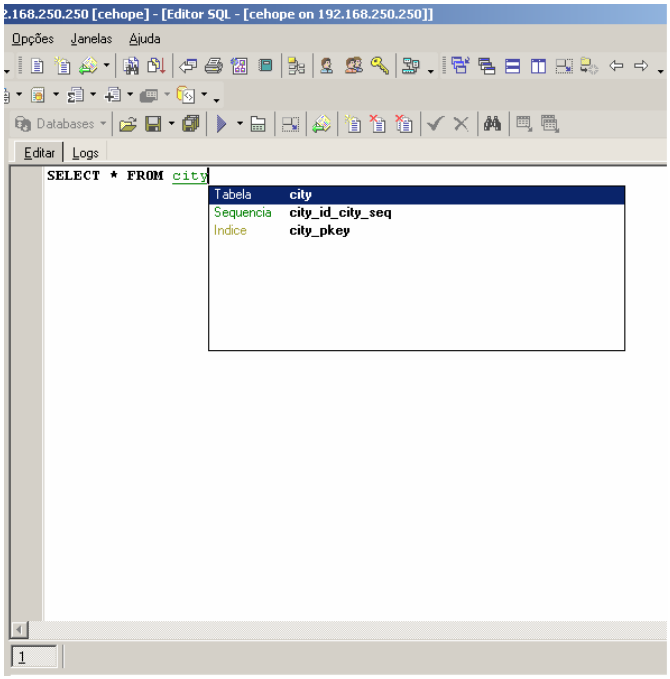

Figura 2.13 – Syntax Highlighting e Autocomplete no query builder

2.5.6 Criar nova tabela

Para criar uma nova tabela usando a ferramenta EMS PostgreSQL Manager, basta clicar com o botão direito em Tabelas (figura 2.14) dentro de *public* ou teclar *Ctrl+N*, tendo o ícone Tabelas selecionado. Depois se dá o nome à tabela e define-se o proprietário da mesma (figura 2.15). Há também a possibilidade de definir se essa nova tabela irá herdar de alguma já existente no banco de dados. Então depois se seleciona a aba *fields* para especificar como ficarão os campos desta tabela (figura 2.16).

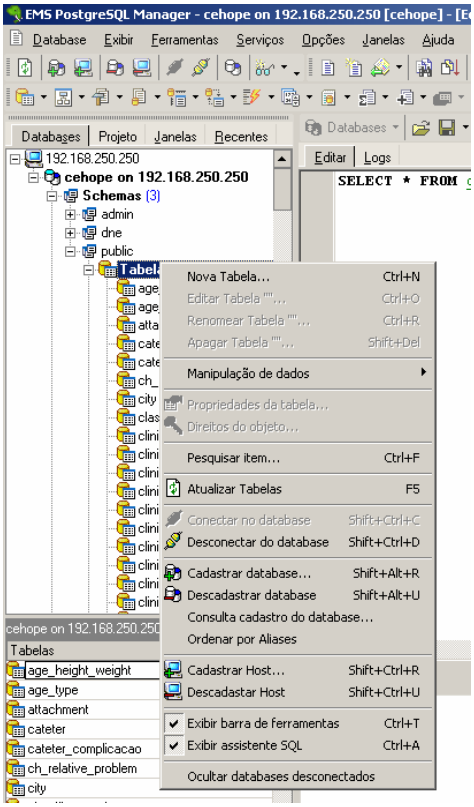

Figura 2.14 – Menu de criação de uma nova tabela

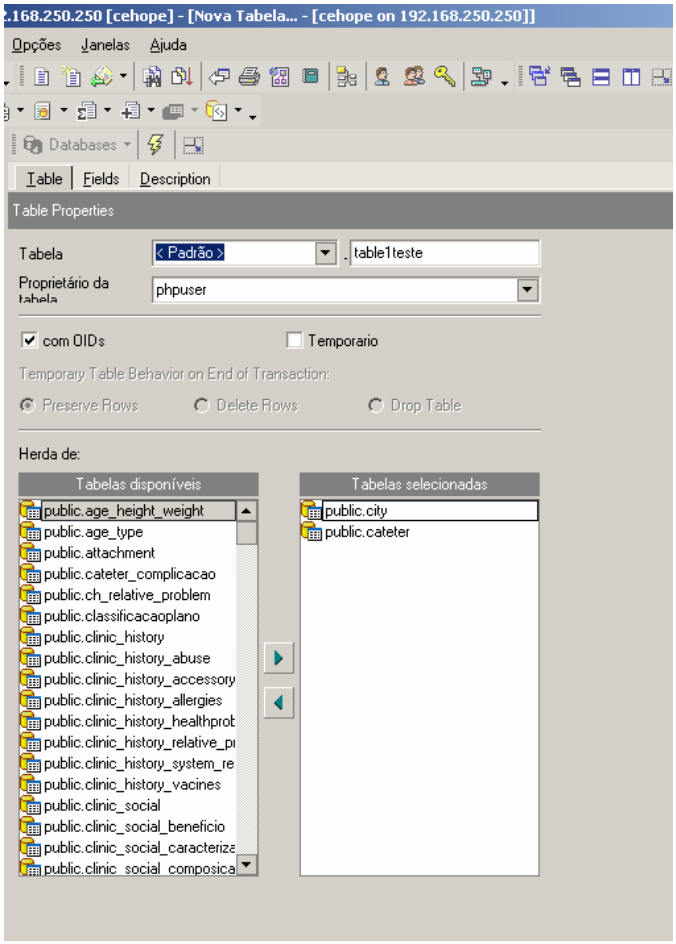

Figura 2.15 – Atributos da nova tabela, herança.

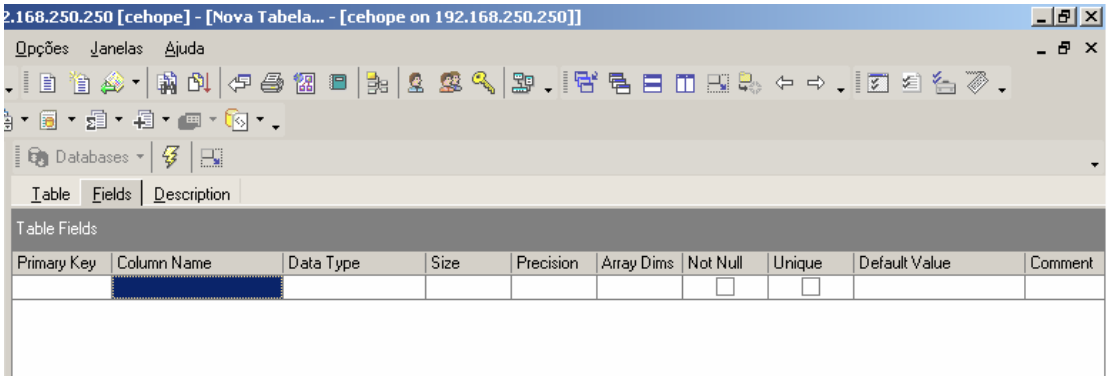

Figura 2.16 – Nova Tabela

2.5.7 Criar uma Base de Dados

Para criar um novo banco, basta clicar com o botão direito na conexão e selecionar Novo Database (figura 2.17), depois é necessário especificar um *host* e a porta, juntamente com um usuário com privilégio de criação de banco e sua senha e clicar em Próximo (figura 2.18). Depois é necessário informar o nome do novo banco, e ainda é possível especificar algumas propriedades como mostrado na figura 2.19, e depois clicar em criar.

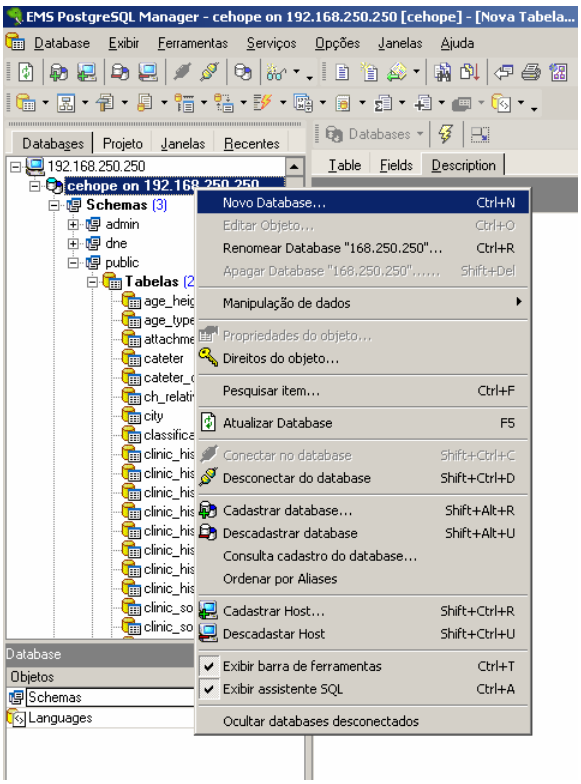

Figura 2.17 – Menu para criar nova base de dados

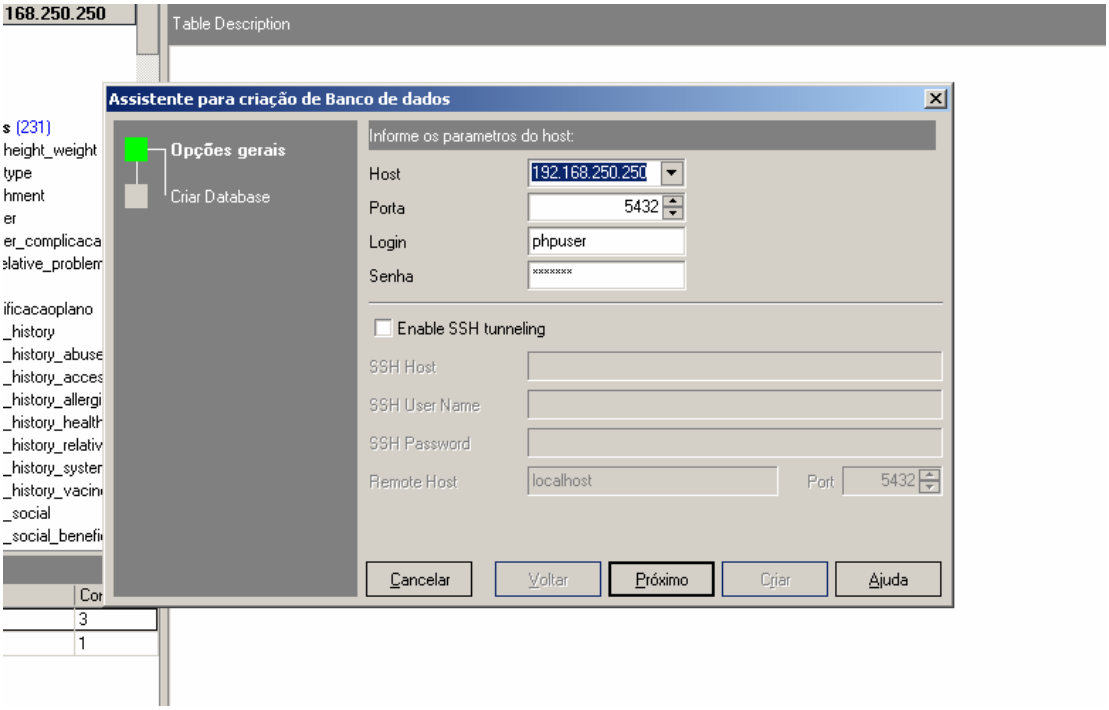

Figura 2.18 – Informações sobre o usuário, senha, porta e *host* para criar a nova base

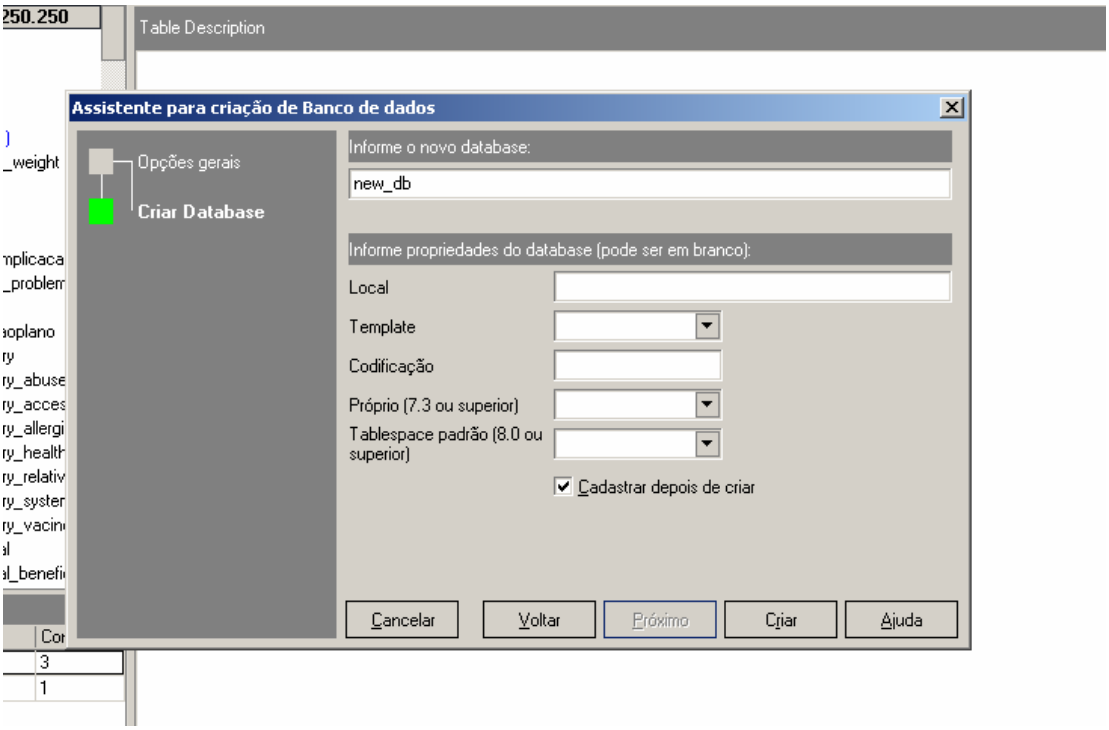

Figura 2.19 – Propriedades do novo banco

2.5.8 Inserindo e removendo um registro

Clicar no sinal "+" no menu acima possibilitará a inserção de um novo registro. Para excluir um registro, basta selecionar a linha e clicar no sinal "-" (figura 2.20).

id_city	name
1	Recife
3	Olinda
5	Cabo
7	São Lourenço da Mata
8	Goiana
9	Moreno
10	Bezerros
11	Caruaru
12	Petrolina
13	Gravatá
14	Pombos
15	Arapiraca
16	Paulista
17	Serra Talhada
18	Triunfo
19	Bonito
20	Brejo da Madre de Deus
21	Tamandaré
22	Santa Cruz
23	Paulo Afonso
27	Nova cidade
24	Juazeiro

Figura 2.20 – Valores contidos em uma tabela

2.5.9 Alterar campos de uma tabela

Para alterar os campos de uma tabela, seleciona-se a tabela que se deseja alterar e com o botão direito clica-se em editar campo (figura 2.21), para abrir um *form* com suas propriedades e alterar o que for necessário no banco de dados (figura 2.22).

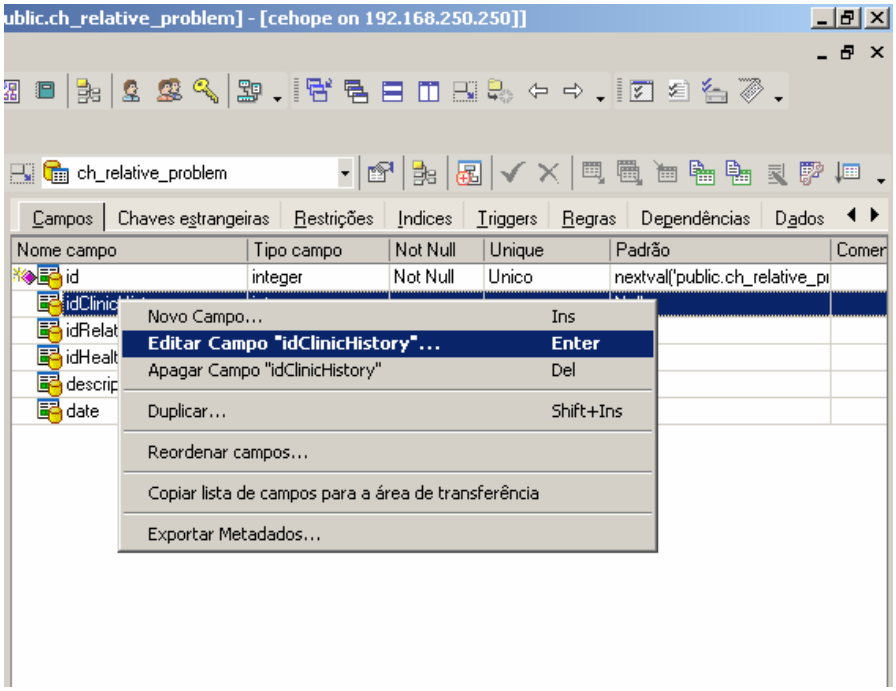

Figura 2.21 – Menu para edição de um campo

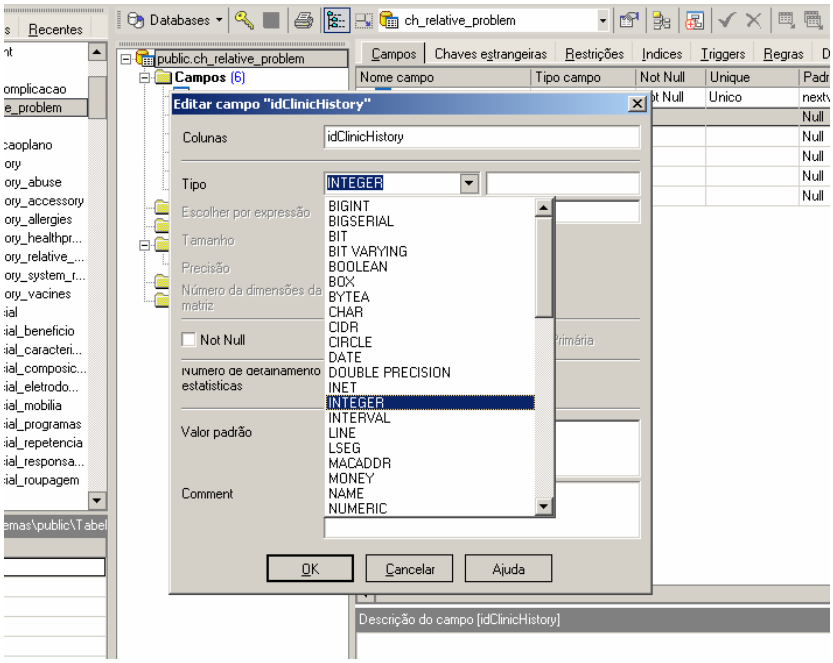

Figura 2.22 – Escolha do tipo da coluna

2.5.10 Adicionar um novo campo

Para adicionar um novo campo à tabela clica-se com o botão direito e depois se escolhe Novo Campo (figura 2.23). Depois basta especificar o nome do novo campo e suas propriedades tais como tipo e valor padrão, entre outras mostradas na figura 2.24.

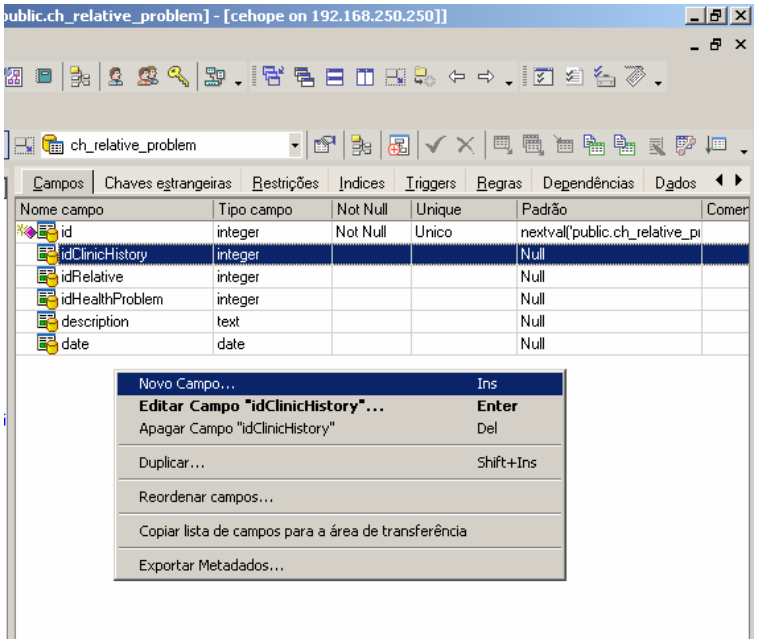

Figura 2.23 – Menu para adição de um novo campo na tabela

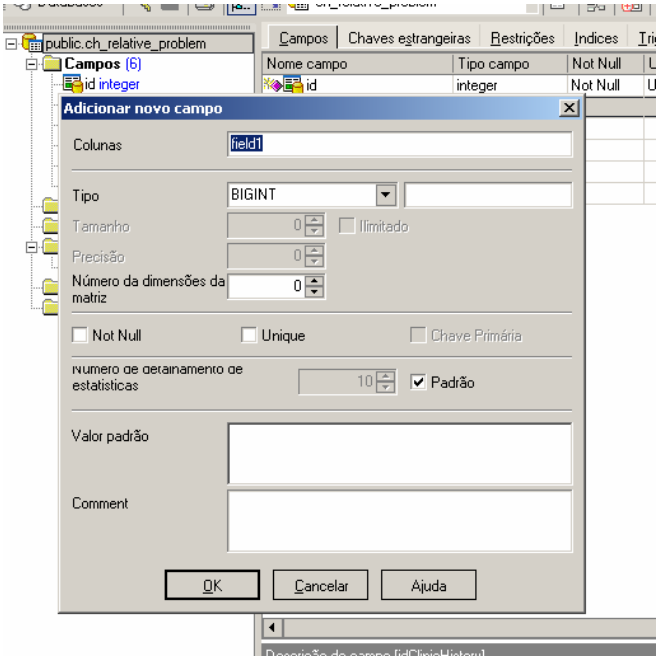

Figura 2.24 – Escolha do tipo e de outras características do novo campo

2.5.11 Executar o vacuum

Foi visto anteriormente que faz parte das boas práticas do administrador do SGBD executar periodicamente o comando **Vacuum** para obter uma melhor performance nas consultas aos bancos de dados. Para executar o Vacuum, usando a ferramenta EMS PostgreSQL Manager, é necessário clicar no menu Serviços e depois escolher a opção Vacuum tabelas (figura 2.25). Depois se define o *host* e o banco no qual se deseja efetuar o Vacuum, além de alguns parâmetros extras, como mostra a figura 2.26. Depois as tabelas que serão submetidas ao vacuum são selecionadas (figura 2.27) e por fim o comando é executado clicando-se em Executar (figura 2.28). Se o parâmetro Exibir relatório detalhado (VERBOSE) for selecionado na figura 2.26, um log da execução será gerado e mostrado na última tela do Vacuum. O comando Vacuum será abordado novamente no capítulo 3.

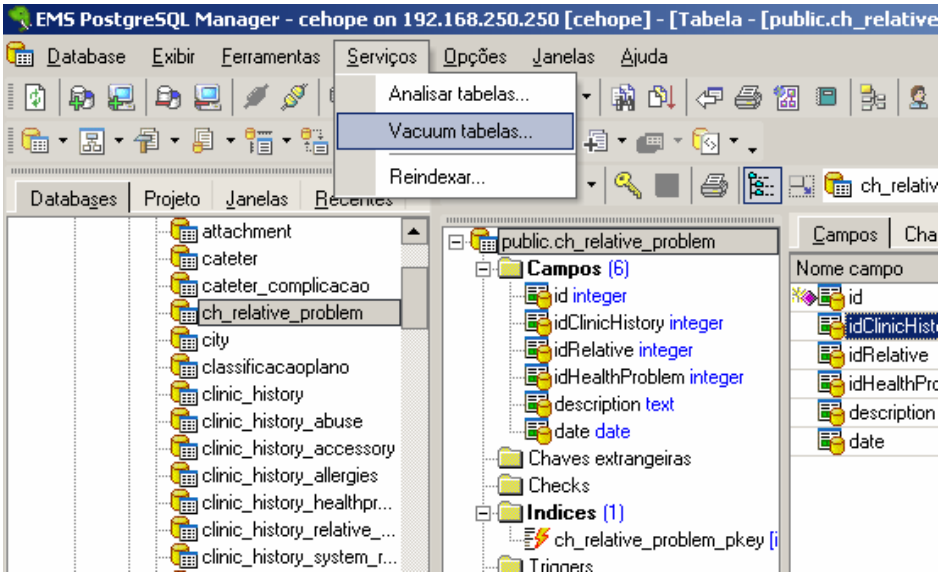

Figura 2.25 – Menu do Vacuum

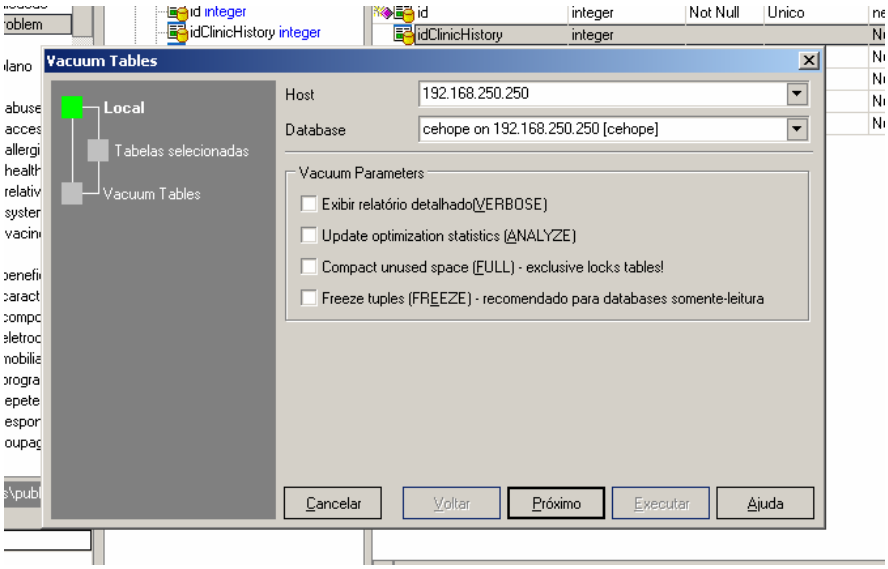

Figura 2.26 – Tela de personalização da execução do Vacuum

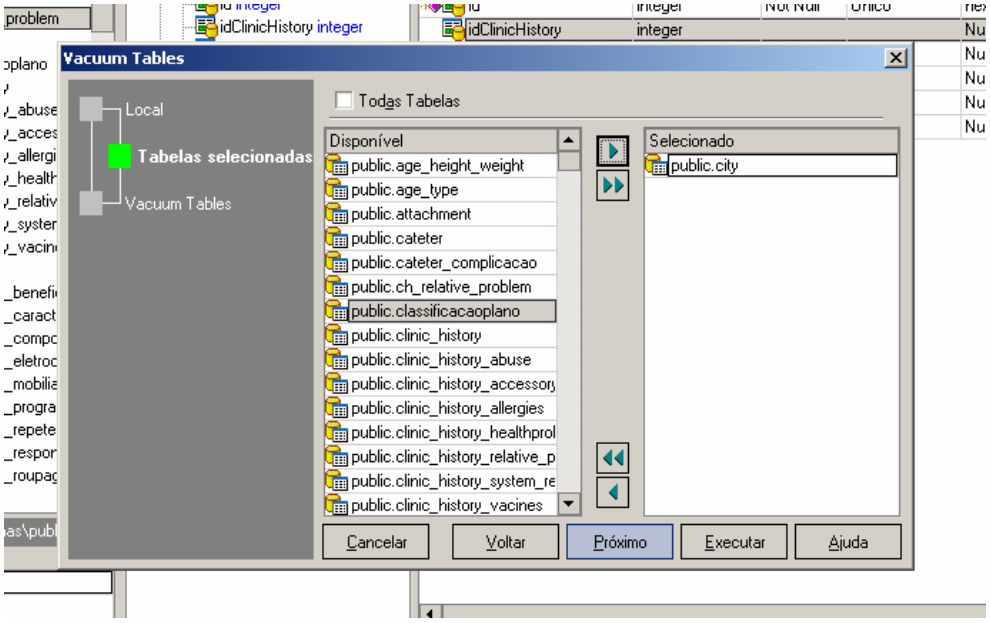

Figura 2.27 – Escolha das tabelas para executar o Vacuum

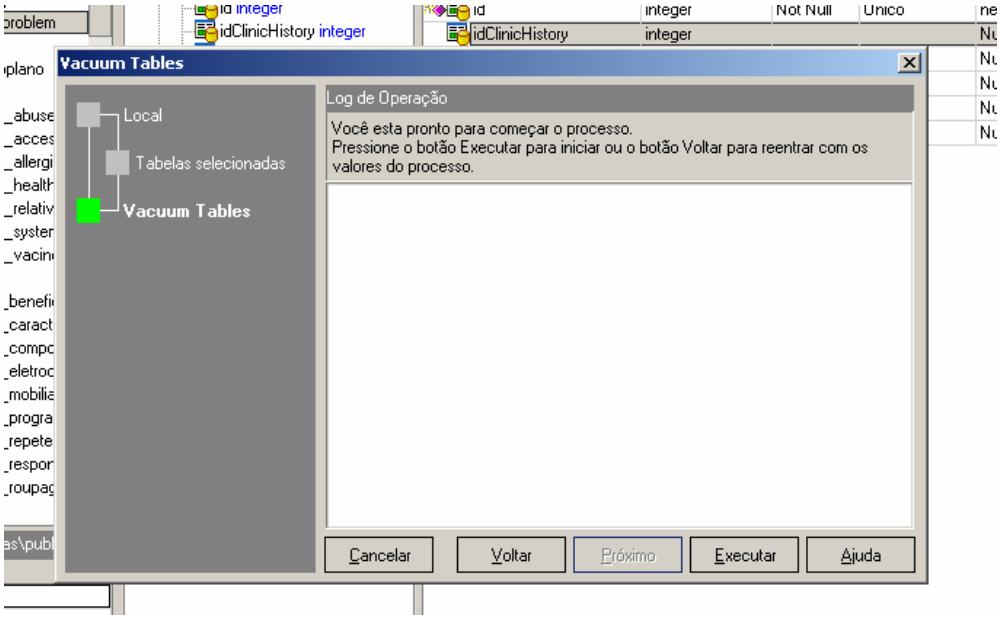

Figura 2.28 – Executar o Vacuum

2.5.12 Mudar permissões das tabelas

Para mudar as permissões de uma tabela, basta clicar com o botão direito em alguma tabela e selecionar a opção Permissão para a tabela (figura 2.29). Um quadro como na figura 2.30 será mostrado e então o usuário poderá adicionar ou remover permissões de um usuário para a tabela selecionada.

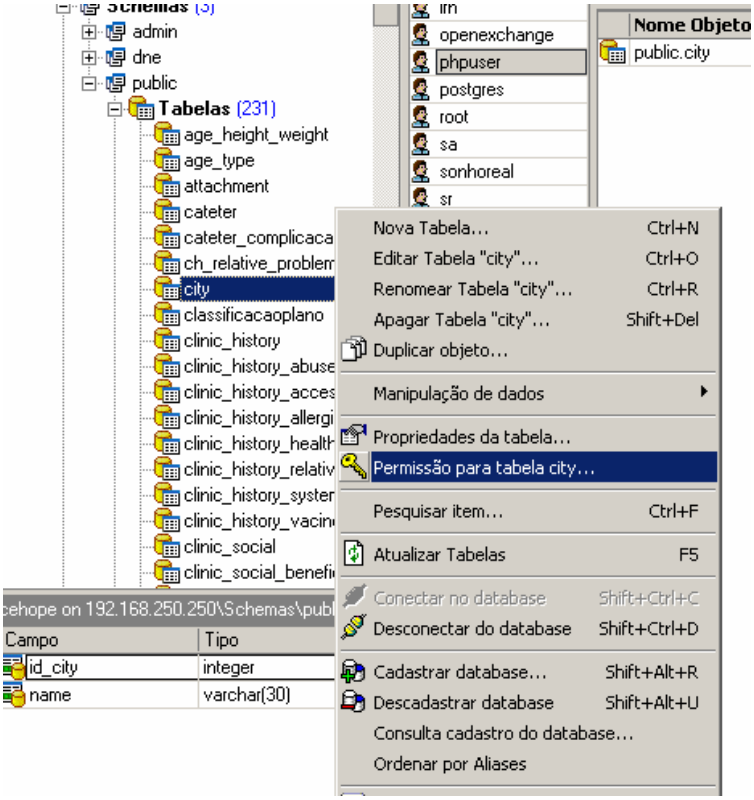

Figura 2.29 – Menu para alterar permissão de tabela

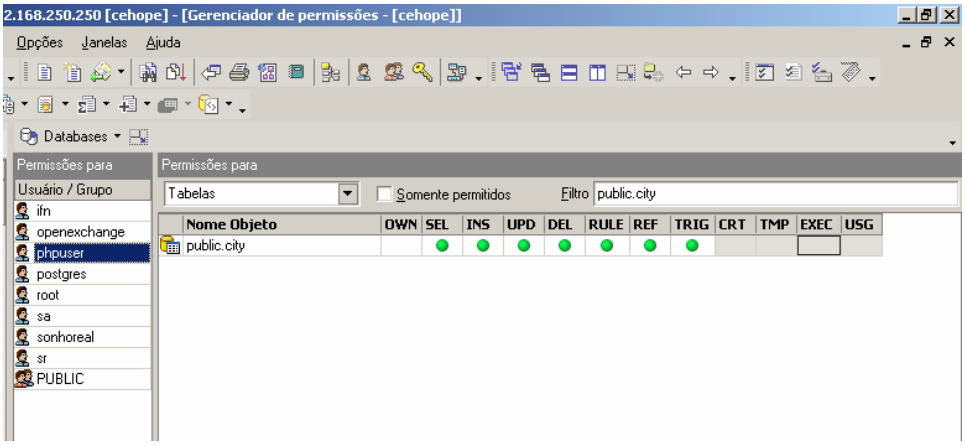

Figura 2.30 – Exibição das permissões da tabela para cada usuário

2.5.13 Analisar tabelas

Também foi visto anteriormente como funciona o Analize no PostgreSQL. Pode-se executar o Analize utilizando a ferramenta EMS PostgreSQL Manager da seguinte maneira: Selecionar o menu Serviços e escolher o item Analisar tabelas (figura 2.31). Assim como no Vacuum, é necessário escolher o *host*, o banco (figura 2.32) e as tabelas (figura 2.33) nas quais se deseja executar a análise. Ao final da execução um log será gerado (figura 2.34) caso o parâmetro Exibir relatório detalhado (VERBOSE) seja marcado.

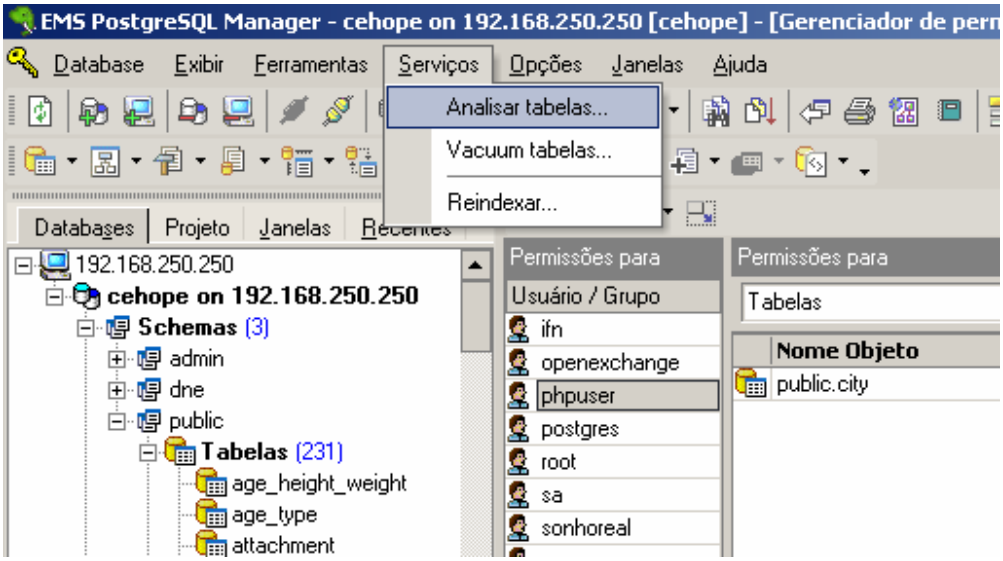

Figura 2.31 – Menu da análise de tabelas

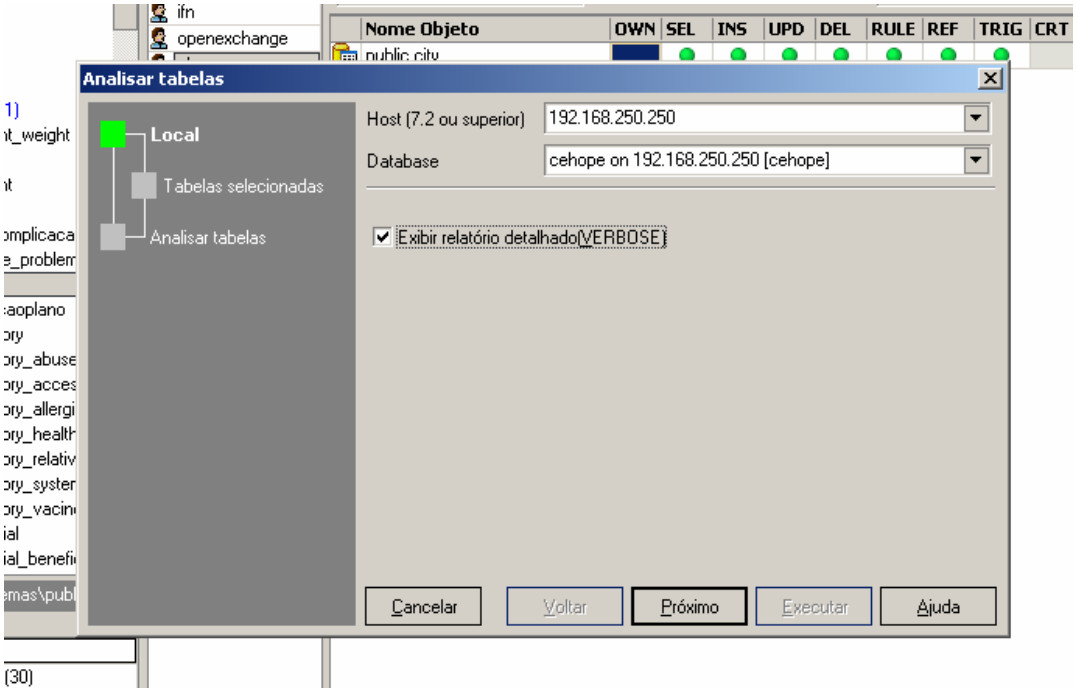

Figura 2.32 – Personalização da análise

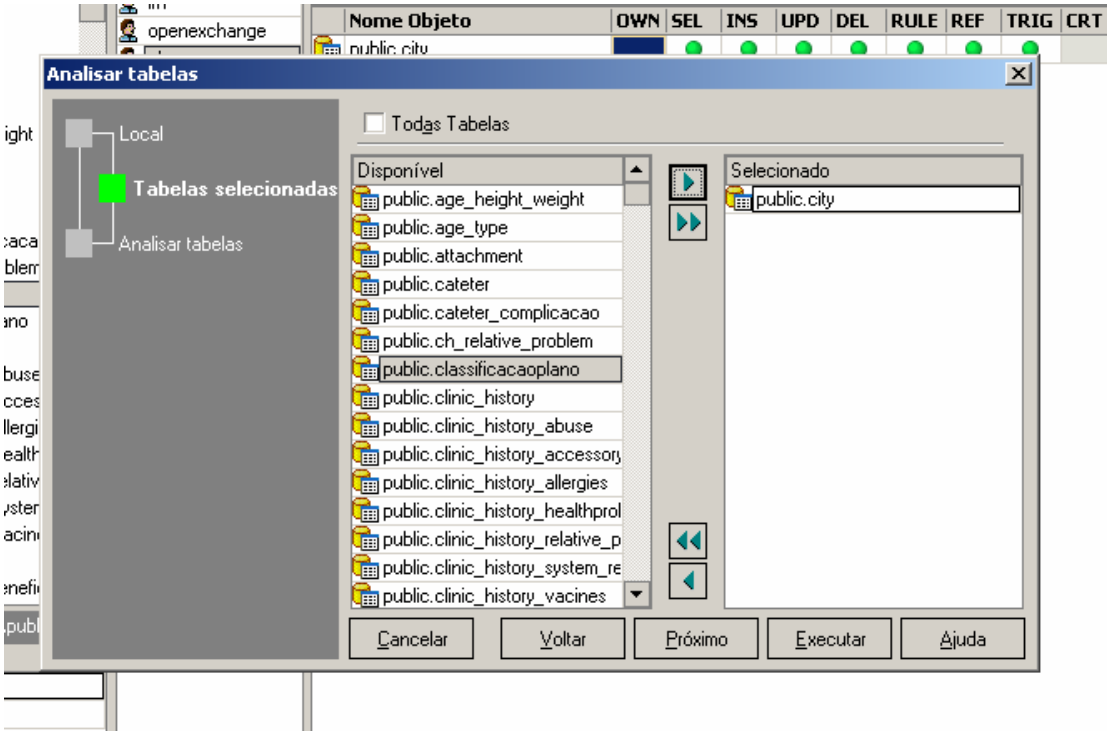

Figura 2.33 – Escolha das tabelas para análise

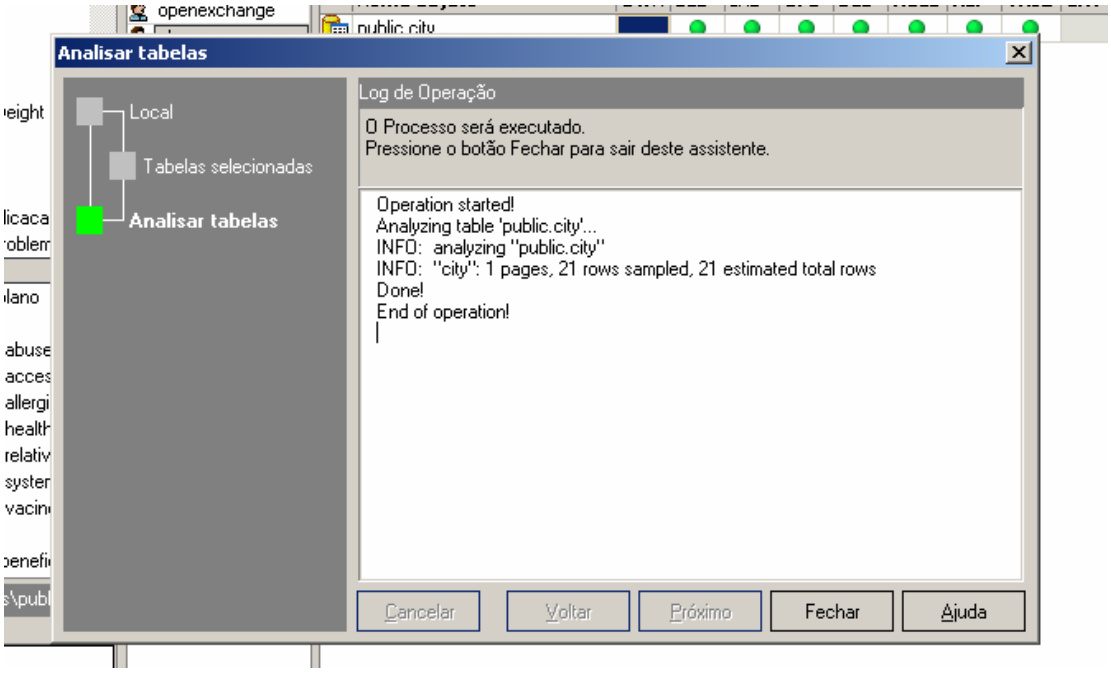

Figura 2.34 – Finalização da análise

2.5.14 Criar função de agregação

Uma função de agregação é uma função que opera sobre um conjunto de valores (tipicamente uma coluna de cada linha que corresponde à condição da consulta), e retorna um único valor calculado a partir destes valores. As funções de agregação típicas são **sum**, **count** e **max**. Para criar uma função de agregação clica-se no botão que contém a letra sigma, símbolo de somatório (figura 2.35), ou clique no meu Database e escolha Novo objeto, e então escolha Aggregate. Depois a tela de criação da função de agregação é mostrada na figura 2.36, onde os parâmetros da criação da função de agregação devem ser preenchidos.

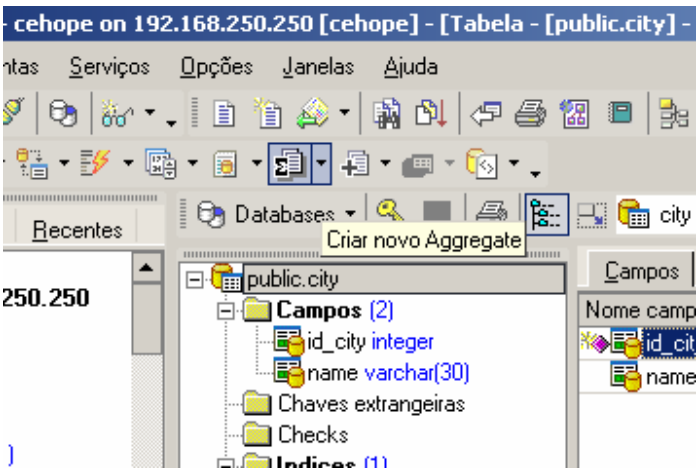

Figura 2.35 – Botão para criar função de agregação

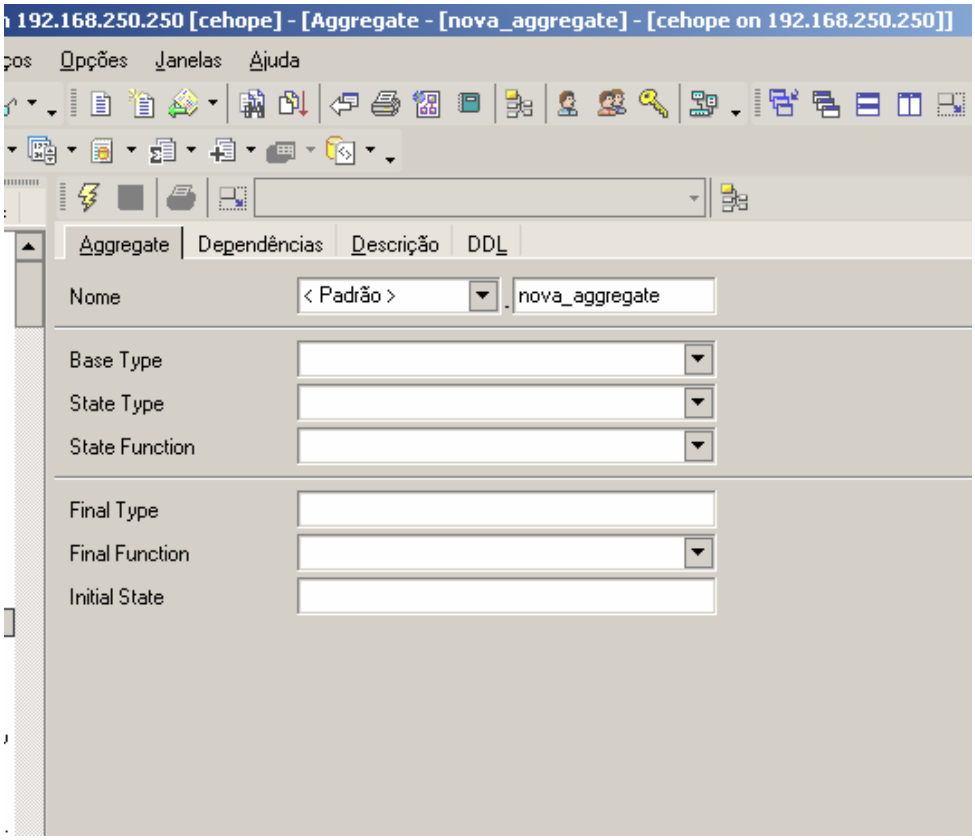

Figura 2.36 – Personalização da função de agregação

2.5.15 Exportar dados

Uma das grandes qualidades do EMS é a exportação de dados. É possível exportar dados em vários formatos como planilha do Excel, DBF, XML, entre outros. Para exportar dados de uma tabela, clica-se com o botão direito na tabela em questão, depois se seleciona o menu Manipulação de dados e por fim clica-se em Exportar dados (figura 2.37). Depois é preciso escolher um formato e um nome para o arquivo de exportação (figura 2.38).

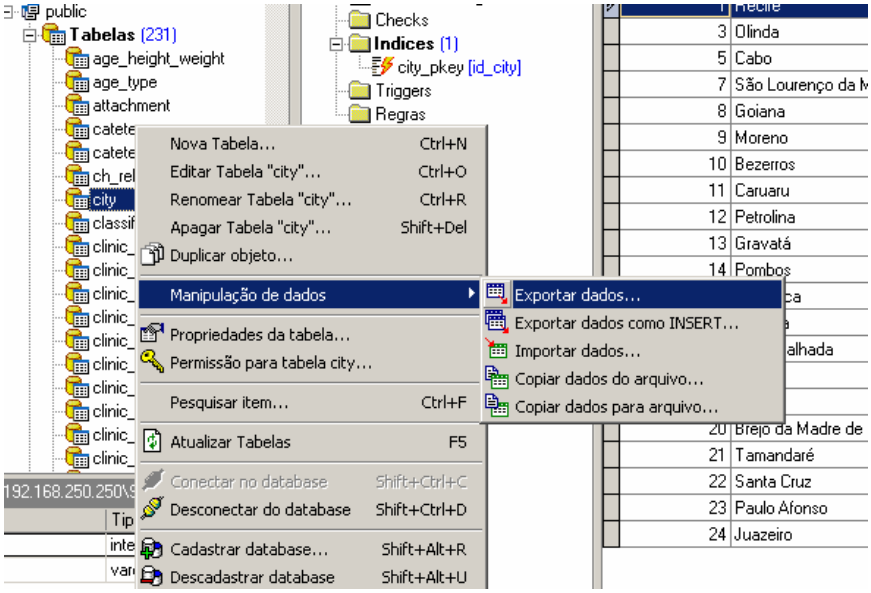

Figura 2.37 – Menu para exportação de dados

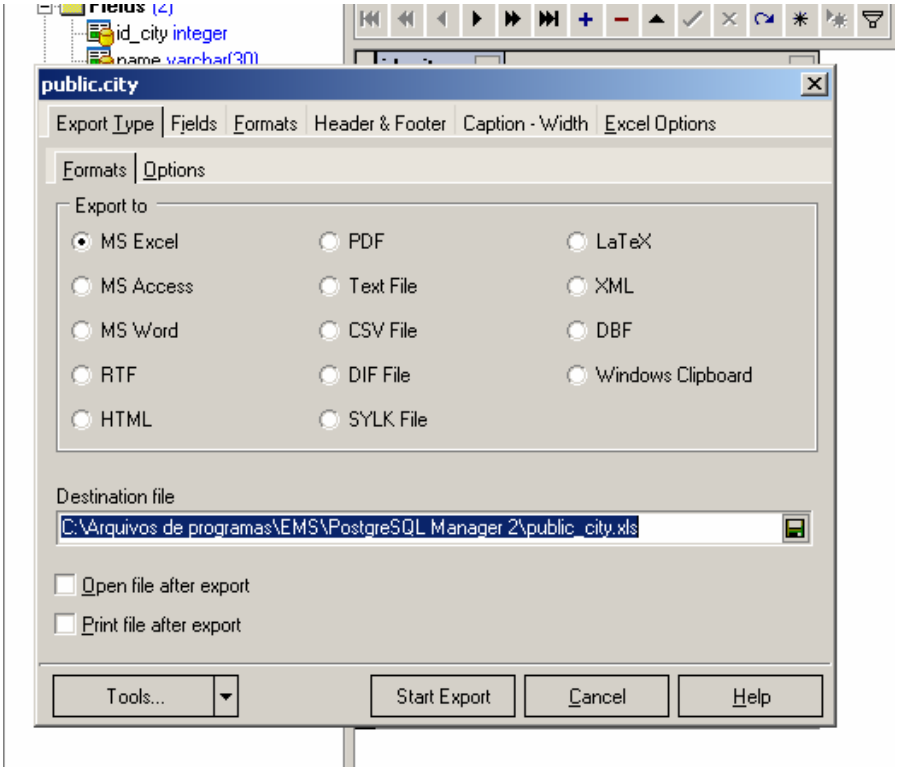

Figura 2.38 – Escolha do formato para exportação

3 Sintonia Fina de SGBD

Este capítulo aborda um das principais ações de administração de SGBD que é a sintonia fina do sistema (*database tuning*)[26]. O *database tuning* é a atividade de fazer uma aplicação de banco de dados rodar mais rápido. “Mais rápido” geralmente quer dizer rendimento elevado, embora possa significar um tempo de resposta mais baixo para aplicações de tempo crítico.

Serão mostrados testes de performance feitos em um servidor Linux[27], primeiramente sem nenhuma otimização. Em seguida os testes serão repetidos após a utilização de tuning em hardware (disco rígido). O tuning em disco será tratado no próximo capítulo.

3.1 O Comando Vacuum

O comando VACUUM recupera a área de armazenamento ocupada pelas tuplas excluídas. Na operação normal do PostgreSQL as tuplas excluídas (DELETE), ou tornadas obsoletas devido a uma atualização (UPDATE), não são fisicamente removidas da tabela, permanecendo presentes até o comando VACUUM ser executado. Portanto, é necessário executar o VACUUM periodicamente, especialmente em tabelas freqüentemente atualizadas. Sem nenhum parâmetro, o VACUUM processa todas as tabelas do banco de dados corrente. Com um parâmetro, o VACUUM processa somente esta tabela. É possível ainda fornecer uma lista de nomes de colunas e, neste caso, somente as

estatísticas para estas colunas são atualizadas.

O comando VACUUM ANALYZE executa o comando VACUUM e depois o comando ANALYZE para cada tabela selecionada. Esta é uma forma de combinação adequada para os scripts das rotinas de manutenção.

O comando ANALYZE coleta estatísticas relativas ao conteúdo das tabelas do PostgreSQL, armazenando os resultados na tabela do sistema *pg_statistic*. Posteriormente, o Otimizador de Consultas utiliza estas estatísticas para auxiliar na determinação do plano de execução mais eficiente para as consultas.

O comando VACUUM (sem o *FULL*) simplesmente recupera o espaço tornando-o disponível para ser reutilizado. Esta forma do comando pode operar em paralelo com a leitura e escrita normal, porque não requer o bloqueio exclusivo da tabela[26]. O VACUUM FULL executa um processamento mais extenso, incluindo a movimentação das tuplas através de blocos para tentar compactar a tabela para o menor número de blocos de disco. Esta forma é muito mais lenta e requer o bloqueio exclusivo de cada tabela para processá-la.

3.2 Testes de Performance

Nesta seção serão descritos os testes de performance realizados, mostrando a necessidade de otimização no disco rígido. Para tanto será utilizada a ferramenta descrita a seguir.

3.2.1 PGBench

O pgbench[1] é um programa para rodar um teste *benchmark* "TCP-B". Pgbench é uma aplicação cliente do PostgreSQL e roda com o PostgreSQL apenas. Ele executa várias pequenas e simples transações incluindo SELECT, UPDATE e INSERT, e depois calcula o número de transações completadas com sucesso em um segundo (*transactions per second - tps*).

Alguns testes foram executados usando uma tabela de 10Mb e um computador Athlon 2.8, 512Mb de RAM, HD padrão IDE, rodando Linux Gentoo sobre o KDE 3.3. Primeiro considerou-se apenas um cliente rodando 10 transações, os resultados obtidos são mostrados na Figura 3.1 .

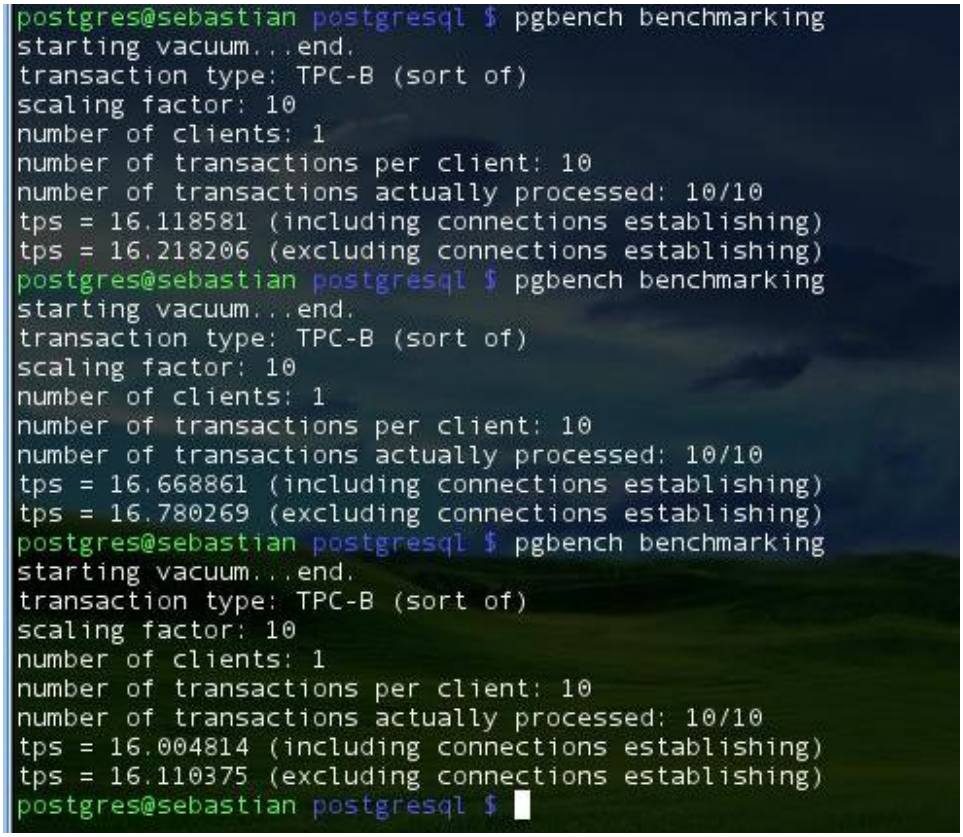

Figura 3.1 – Testes de performance com 1 cliente e 10 transações por cliente

Depois considerou-se 20 clientes, cada um rodando 100 transações, com os resultados mostrados na Figura 3.2 .

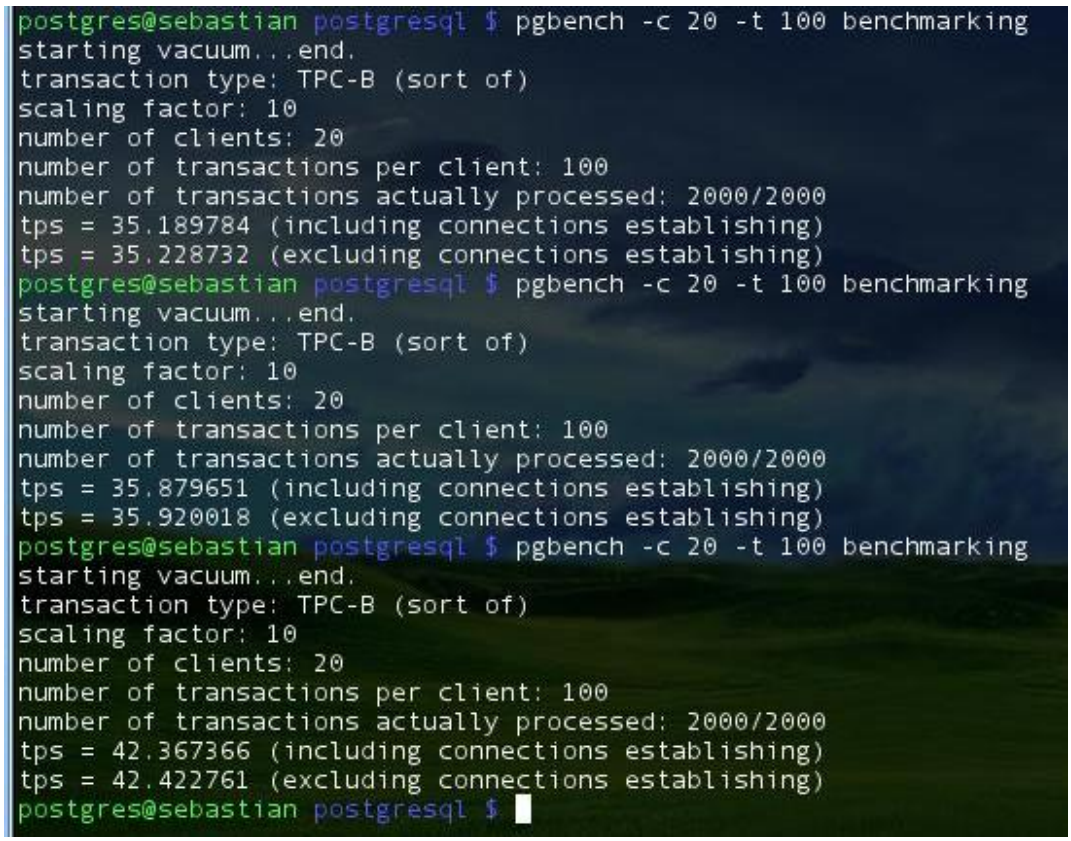

Figura 3.2 – Testes de performance com 20 clientes e 100 transações por cliente

Então se usando um processo de *tuning* de HD padrão IDE (será mostrado mais abaixo como fazer o processo de tuning no HD IDE), foram alteradas algumas *flags* nele de modo a executar acessos a disco mais rapidamente. O primeiro teste é executado novamente aqui, agora tendo sido efetuado o tunning no HD. Os resultados aparecem na Figura 3.3 .

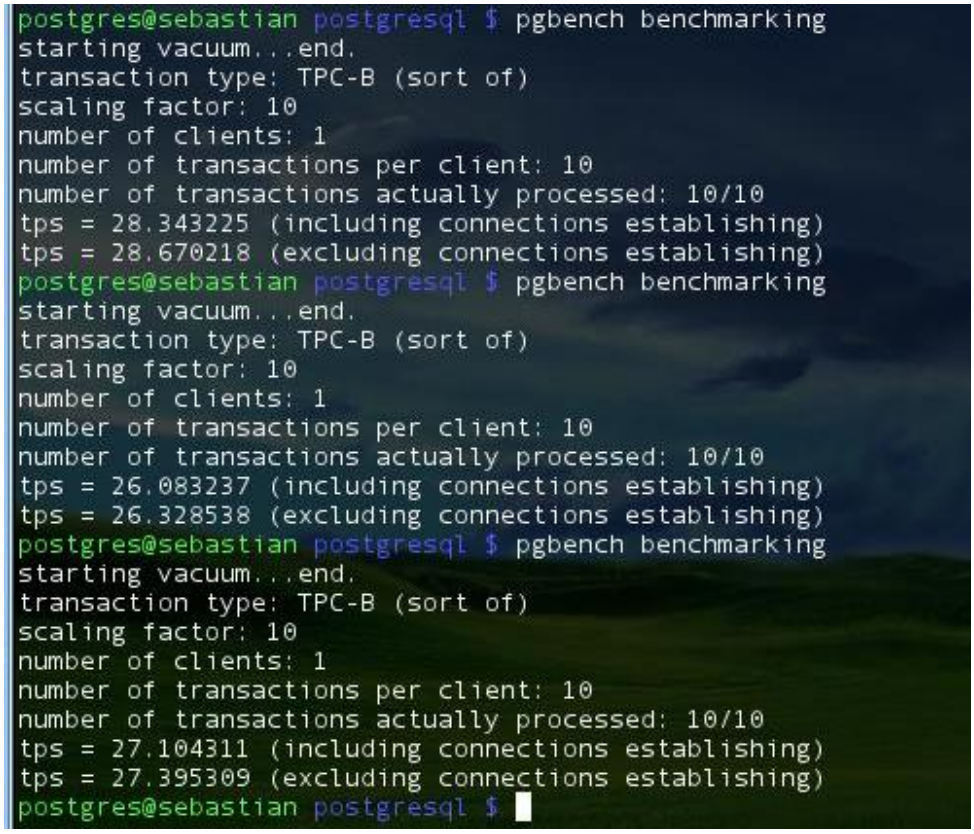

Figura 3.3 – Testes de performance com 1 cliente e 10 transações por cliente com tunning de HD

O segundo teste com mais clientes e transações por clientes é também executado novamente, tendo sido efetuado o *tunning* no HD, com os resultados mostrados na Figura 3.4 .

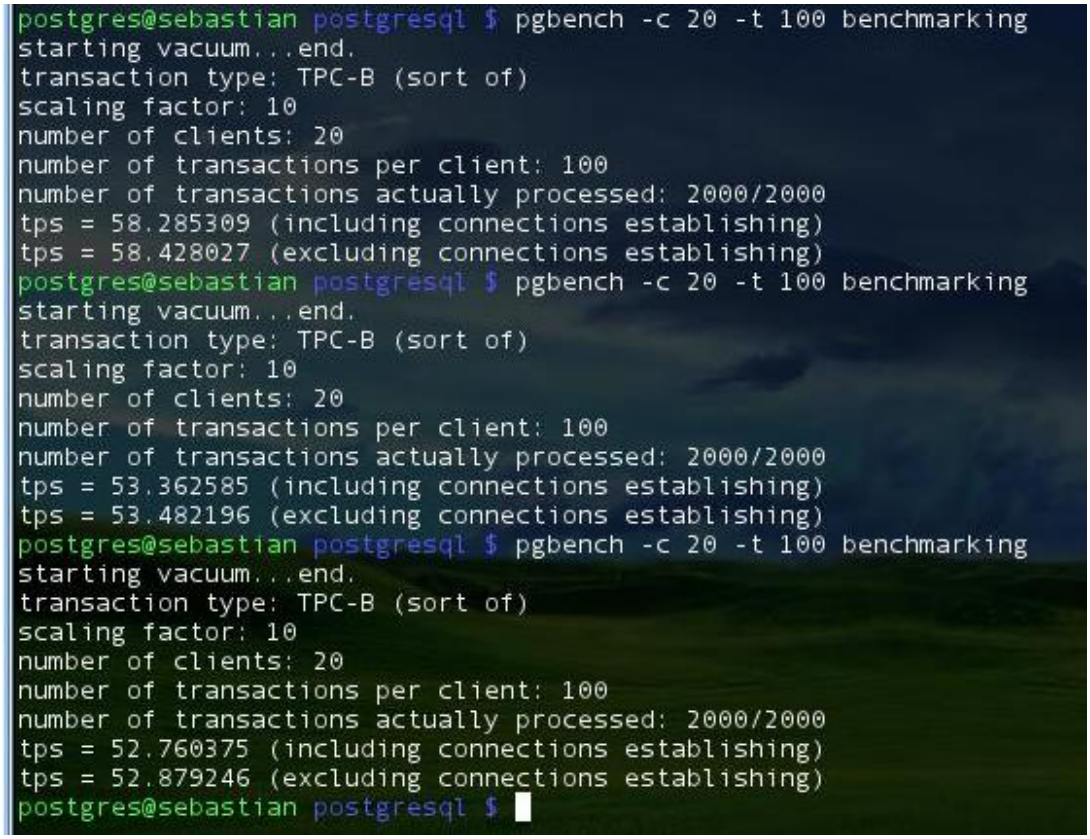

Figura 3.4 – Testes de performance com 20 clientes e 100 transações por cliente com tunning de HD

3.2.2 Vantagens do processo de tunning

Como foi possível perceber, o processo de tuning fez com que mais transações por segundo (tps) fossem processadas e com isso houve um sensível ganho de performance. A figura 3.5 mostra um gráfico comparativo entre os testes com um cliente rodando 10 transações, e a figura 3.6 mostra um gráfico comparativo entre os testes com 20 clientes rodando 100 transações cada.

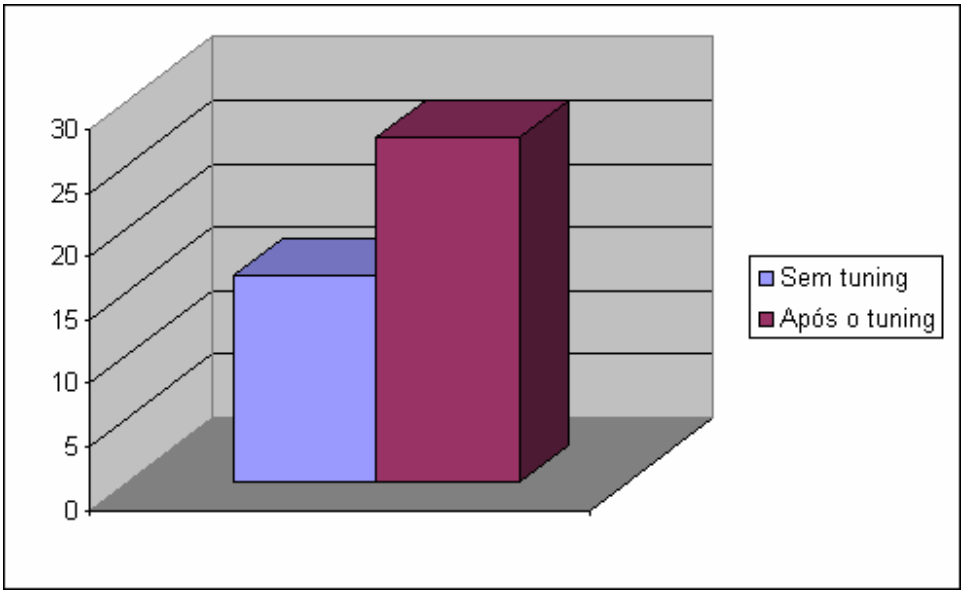

Figura 3.5 – Comparativo das tps em um hardware sem tuning e após o tuning com apenas um cliente rodando 10 transações

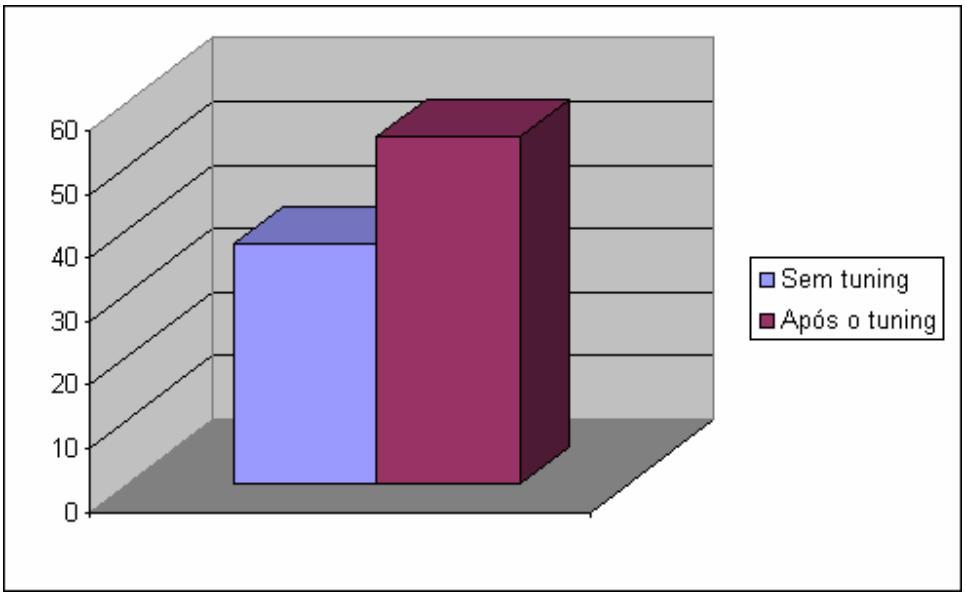

Figura 3.6 – Comparativo das tps em um hardware sem tuning e após o tuning com 20 clientes rodando 100 transações cada

4 Otimizando um Servidor Linux para SGBD

Para se ter um Servidor Linux otimizado, antes de mais nada, precisa-se saber qual a real necessidade e quais as características de cada sistema de arquivos disponível. A seguir serão abordadas as principais características dos principais sistemas de arquivos do Linux.

4.1 Características dos sistemas

XFS: Sistema de arquivos muito usado no IRIX (um tipo de Unix) e mantido pela SGI – Silicon Graphics [5]. É considerado um dos melhores, se não o melhor, sistema de arquivos para SGBD. Já vem como padrão nos Linux com kernel versão 2.6, e em algumas distribuições que utilizam o kernel 2.4.

ReiserFS: Sistema de arquivos desenvolvido por Hans Reiser e mantido pela Namesys [12]. Atualmente na versão 4 (experimental). Segundo o site da Namesys, é o sistema de arquivos mais veloz já desenvolvido, no entanto não se pôde encontrar um comparativo oficial que comprove isso. A maioria das distribuições ainda traz a versão 3 do Reiser.

JFS: De propriedade da IBM [10], foi desenvolvido para rodar nos UNIX que a IBM vendia. Após a adoção do Linux pela IBM, foi portado para trabalhar também nesse SO. É extremamente rápido e tem a

vantagem de trabalhar com uma quantidade de dados muito superior aos demais sistemas de arquivos. O JFS perdeu um pouco de credibilidade no início devido a *bugs* e instabilidade, porém encontra-se estável atualmente.

Ext2: Sistema de arquivos [2] padrão na maioria das distribuições com kernel 2.2. É o mais rápido por ser extremamente simples e sem grandes recursos. O grande problema do ext2 é a falta de confiabilidade, pois é muito sensível à instabilidade na máquina (travamentos por exemplo). É recomendado para partições que manipulam dados temporários, tais como /var/log, /tmp e o diretório de log de transações.

Ext3: Sistema de arquivos [4] padrão na série atual do kernel. É o sucessor do ext2 e bem mais seguro, pois oferece o *Journaling* (também chamado de *Loggin*), mas peca no que diz respeito à velocidade.

4.2 Qual sistema de arquivos usar?

Segundo o site fsbench [3], o XFS e o JFS têm seu desempenho muito próximo ao Etx2, apesar de muitos utilizarem o XFS por considerá-lo um pouco mais rápido que o JFS. Para quem utiliza o Red hat 9[18] e não quer atualizar o kernel para trabalhar com o XFS, é sugerido utilizar o JFS, que é o padrão nessa versão da distribuição. Já para quem usa o Fedora Core[28], Slackware[29] e o Conectiva[30], terá disponíveis as vantagens do XFS sem necessidade de atualização do kernel. Não é

aconselhável o uso do ReiserFS v4 em ambiente de produção por ainda estar em testes.

Considerando somente os sistema de arquivos com versão estável, é aconselhável ficar com o XFS, pois é seguro, veloz e permite uma administração flexível das partições formatadas com ele.

4.3 Determinando o tamanho das partições para BD

Na prática é difícil estipular qual vai ser o tamanho ideal de uma partição do BD, estima-se que uma base de dados cresça em média 3 Mb por dia e no final de um mês ter-se-á um acréscimo de 90 Mb, e no final de um ano, um aumento de 1080 Mb na base, ou seja, pouco mais de 1 Gb. Porém, não se pode ignorar no cálculo fatores como o crescimento do cliente, eventuais falhas, *backups* diários, entre outros. Se não se tem uma noção prévia de qual seria o crescimento estimado do banco, pode-se deixar uma partição de 10 Gb para o BD ou usar o LVM.

O LVM (Logical Volume Manager) possibilita o aumento do tamanho de uma partição sem perda de dados. Ou seja, no caso da partição do BD, não haveria necessidade de remover o banco, recriar a partição, recriar o banco, e assim por diante. O LVM cria volumes lógicos, onde, por exemplo, dois ou mais HD podem ser visualizados como se fossem um único dispositivo. O espaço total de armazenamento desse volume lógico seria a soma dos tamanhos de cada HD. A desvantagem do LVM é que o

desempenho da unidade lógica não é o mesmo se comparado com o acesso direto a um dispositivo físico, apesar de que dependendo do hardware envolvido, a perda de performance é imperceptível.

É possível utilizar o LVM para implementar um RAID 0[1]. No entanto, não é sua função principal, e em uma situação como essa, a perda de um HD implicaria na perda total dos dados do volume. É recomendável utilizar o LVM em conjunto com o RAID, preferencialmente via hardware.

Na figura 4.1 seguem os passos para criação de um LVM de forma direta e prática.

- Parâmetros utilizados no **fdisk**:
  - **N:** Nova partição;
  - **T:** Define qual o tipo de partição (LVM é o código 8e);
  - **W:** Salva a alteração e sai do **fdisk**.
- Outros comandos utilizados durante o processo:
  - **vgscan -** Varre todos os discos à procura de VGs (Volume Groups) e monta os arquivos /etc/lvmtab e /etc/lvmtab.d/* que são o banco de dados para todos os outros comandos do lvm;

1- Na organização RAID-0 os HD são concatenados de modo a formar um único dispositivo, cujo volume total é igual à soma dos volumes de cada HD envolvido.

- **pvcreate** – Inicializa um volume físico para ser usado pelo LVM. O volume físico pode ser uma partição de disco, o disco inteiro, um meta-device ou um arquivo de *loopback*;
- **pvdisplay** – Permite visualizar os atributos (tamanho por exemplo) de um ou mais volumes físicos;
- **vgcreate** –: Cria o Volume Group (VG – "disco virtual") utilizando o volume físico criado anteriormente pelo pvcreate;
- **lvcreate** –Cria um novo Logical Volume (Volume Lógico) em um Volume Group já existente. Os parâmetros -L e -n determinam o tamanho e o nome do volume lógico, respectivamente;
- **mkfs** – Constrói um sistema de arquivos no volume lógico recém criado;
- **mount** – Anexa um sistema de arquivos de um dispositivo a um diretório.

Sequência de criação de um dispositivo LVM (sem definição de RAID 0)

```
Manipulação da partição que iremos criar para o LVM, criaremos 6Gb para o LVM:

root@davi:/# fdisk /dev/hda

Command (m for help): n
First cylinder (754-2434, default 754): (ENTER)
Using default value 754
Last cylinder or +size or +sizeM or +sizeK (754-2434, default 2434): +6144M
Using default value 2434

Command (m for help): t
Partition number (1-8):
Hex code (type L to list codes): 8e
Changed system type of partition 8 to 8e (Linux LVM)
Command (m for help):w

Usando o VgScan pela primeira vez para criar o banco de dados para o LVM:
root@davi:/# vgscan

Usando o pvcreate para a criação de um volume físico para o /dev/hda8:
root@davi:/#  pvcreate /dev/hda8

Usando o pvdisplay para certificarmos que realmente o LVM atribui o /dev/hda8 como utilizá-
vel:
root@davi:/# pvdisplay /dev/hda8
pvdisplay -- "/dev/hda8" is a new physical volume of 6.0 GB

Usando o vgcreate para a criação do Disco Virtual chamado "concat":
root@davi:/# vgcreate concat /dev/hda8

Usando o lvcreate para a criação de um volume lógico de 900M chamado "backup" dentro do
disco virtual "concat":
root@davi:/# lvcreate -L 900M -n backup concat

Usando o mkfs.ext2 para a criação do sistema de arquivos para o volume "backup":
root@davi:/# mkfs.ext2 /dev/concat/backup

Criando a pasta para acessar o volume "backup"
root@davi:/# mkdir /mnt/backup

Usando o mount para a montagem deste sistema de arquivos:
root@davi:/# mount /dev/concat/backup /mnt/backup

Para termos certeza que montou:
root@davi:/# mount
root@dblinux:/# mount
/dev/hda1 on / type reiserfs (rw)
proc on /proc type proc (rw)
/dev/hda3 on /tmp type ext2 (rw)
/dev/hda5 on /var type ext2 (rw)
/dev/hda6 on /usr type reiserfs (rw)
/dev/hda7 on /usr/local type reiserfs (rw)
/dev/concat/backup on /mnt/backup  type ext2 (rw)
```

Figura 4.1 – Passos para criação de um LVM

4.4 Tuning de HD padrão IDE

Será usado o utilitário HDPARM para agilizar o acesso a discos IDE no Linux. Esse utilitário habilita algumas opções de acesso ao disco e pode melhorar em até 50% a velocidade de leitura e gravação de dados. Logicamente, a utilização do HDPARM deve ser feita com cuidado, pois parâmetros errados podem fazer a máquina reiniciar, travar ou perder dados.

O HDPARM apenas ativa no sistema operacional os *flags* de manipulação de HD. O HDPARM funciona para todos os HD padrão IDE, lembrando que algumas opções só devem ser configuradas caso o HD específico dê suporte à funcionalidade configurada por ela. O único risco da utilização desta ferramenta é o de se habilitar um *flag* cujo HD em questão não ofereça suporte, e com isso corromper os dados.

Inicialmente, será executado um teste para saber o tempo e taxa de transferência de dados inicial (sem qualquer otimização) para servir de comparação durante a configuração dos parâmetros do HDPARM. Para isso é executado o comando HDPARM passando como parâmetro "-Tt" e o dispositivo /dev/hda, como mostrado na figura 4.2.

Retorno inicial do hdparm com a configuração padrão do HD.

```
[root@dblinux root]# hdparm -Tt /dev/hda
/dev/hda:
 Timing buffer-cache reads:   128 MB in  0.72 seconds =175.34 MB/sec
 Timing buffered disk reads:  64 MB in  2.22 seconds = 28.70 MB/sec
```

Figura 4.2 – Teste inicial

Para saber se está havendo ganho de performance durante as próximas configurações, compara-se os valores das taxas de transferência do teste inicial com os valores obtidos a cada nova configuração. Se o valor da taxa de transferência aumentar, houve ganho de performance.

O resultado indicado por "*Timing buffer-cache reads*" mede a leitura da *cache*, e é uma boa referência para detectar a eficiência do conjunto IDE/Memória/CPU. Já o resultado mostrado por "*Timing buffered disk reads*" ou "Tranferência do disco para buffer" mede a taxa de leitura do disco propriamente dito.

O próximo passo é habilitar o suporte a 32 bits de transferência de dados entre o barramento PCI e o controlador de disco, e em seguida verificar se houve ganho de desempenho. Praticamente todos os *chipsets*[2] modernos dão suporte ao modo 3 (transferência 32bits sincronizada). O teste é mostrado na figura 4.3.

2- O chipset é o principal componente da placa mãe, justamente quem determina a maior parte dos seus recursos e desempenho.[9]

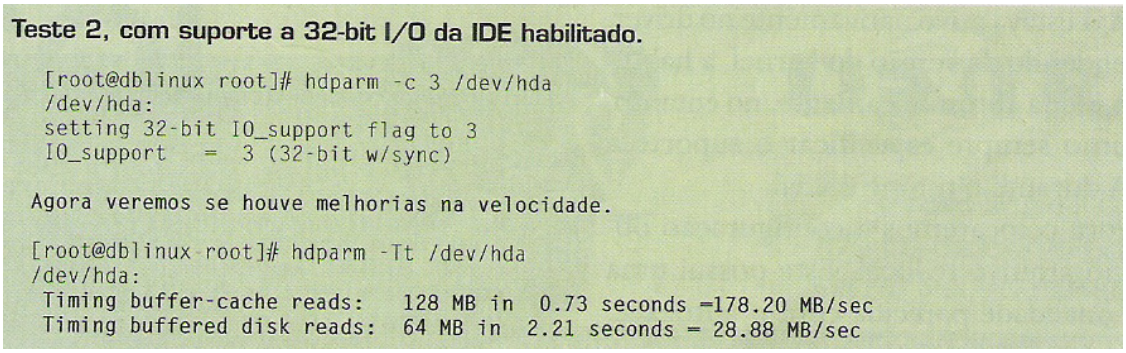

Figura 4.3 – Habilitar o suporte a 32 bits de transferência de dados

Observa-se o ganho de desempenho: *Timing buffer-cache* pulou de 175.34 MB/sec para 178.20 MB/sec e *Timing buffered disk reads* de 28.70 MB/sec para 28.88 MB/sec.

O próximo passo será habilitar a transferência de múltiplos setores (*multcount*) e verificar novamente se houve ganho de desempenho. O teste é mostrado na figura 4.4.

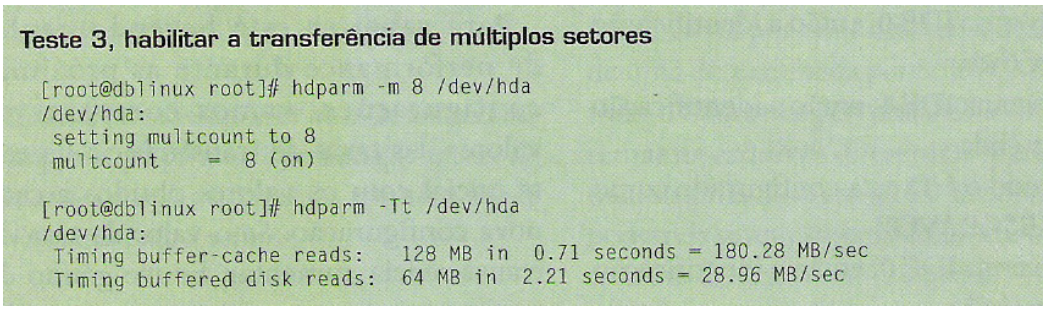

Figura 4.4 – Habilitar a transferência de múltiplos setores

O *multcount* diz qual é a quantidade máxima de setores no disco que pode ser lido de uma vez. Configura-se para o valor máximo ao qual o HD em questão oferece suporte. Para saber o valor máximo que o HD oferece suporte, roda-se o HDPARM com o parâmetro "-i" e verifica-se o valor de *MaxMultSect*.

O próximo passo será habilitar a leitura seqüencial adiantada de setores de disco. o aumenta a velocidade de leitura de arquivos grandes, mas pode prejudicar o desempenho com arquivos pequenos. Na dúvida é recomendável deixar o valor igual ao do *multcount* para arquivos grandes. O teste é mostrado na figura 4.5.

Teste 4, habilitar a leitura adiantada de 32 setores.

```
[root@dblinux root]# hdparm -a 32 /dev/hda
/dev/hda:
setting fs readahead to 32
readahead    = 32 (on)
[root@dblinux root]# hdparm -Tt /dev/hda
/dev/hda:
Timing buffer-cache reads:   128 MB in  0.70 seconds =182.86 MB/sec
Timing buffered disk reads:  64 MB in  2.21 seconds = 28.96 MB/sec
```

Figura 4.5 – Habilitar leitura adiantada

A seguir será habilitado o uso do DMA[3], como na Figura 4.6 .

Teste 5, habilitar o uso do DMA.

```
[root@dblinux root]# hdparm -d 1 /dev/hda
/dev/hda:
 using_dma    =  1 (on)

[root@dblinux root]# hdparm -Tt /dev/hda
/dev/hda:
 Timing buffer-cache reads:   128 MB in  0.70 seconds =182.86 MB/sec
 Timing buffered disk reads:  64 MB in  2.21 seconds = 28.96 MB/sec
```

Figura 4.6 – Habilitar o uso do DMA

3- Abreviação de direct memory access, uma técnica para transferir dados da memória principal para um dispositivo sem passar pelo processador

Observa-se pelo resultado do teste da figura 4.6 que, neste caso, não houve ganho de velocidade. Isso indica que o acesso a DMA já estava ativo nativamente no driver. Dependendo da versão do kernel, a habilitação é feita automaticamente, no entanto, é bom costume sempre especificar o suporte a DMA durante o *tunning* oficial.

A seguir será colocada a configuração obtida no arquivo *rc.local*. Este possui uma funcionalidade parecida com o *autoexec.bat* do MS-DOS[31]. Isso é necessário para que a configuração seja executada a cada boot do Linux. Veja a figura 4.7.

Adicionar a configuração adquirida durante os testes para serem executadas no boot do Linux

Agora colocaremos no rc.local. Este arquivo é um tipo de AUTOEXEC.BAT do MS-DOS no Linux.

```
[root@dblinux root]# cat /etc/rc.d/rc.local
#!/bin/sh
#
# This script will be executed *after* all the other init scripts.
# You can put your own initialization stuff in here if you don't
# want to do the full Sys V style init stuff.
touch /var/lock/subsys/local
/sbin/hdparm  -c 3 -m 8 -a 32 -d 1 /dev/hda
```

Figura 4.7 – Exemplo de um rc.local para configurar o HD a cada boot da máquina

Os HD do tipo SCSI possuem uma placa controladora específica, onde é possível realizar alguns ajustes de configuração diretamente na placa, por isso não existem utilitários genéricos de *tunning* para HD SCSI.

4.5 Ajustando o I/O da máquina

Através do utilitário **elvtune** será ajustado o kernel do sistema operacional, fazendo-o trabalhar melhor com a leitura e a gravação de dados no dispositivo físico. Ele reorganiza os pedidos de leitura/escrita ao disco de acordo com seus setores, reduzindo assim os movimentos da cabeça HD e, portanto, o tempo de acesso.

Para os Linux com o kernel 2.6, o **elvtune** foi depreciado a favor do **sysfs**, apesar dele ainda funcionar com essa versão do kernel. Basicamente os parâmetros utilizados são:

-r: Qual é a latência em blocos para leitura;

-w: Qual é a latência em blocos para gravação.

O algoritmo de ajuste de I/O do kernel utiliza uma fila de requisição de I/O em função do setor do disco. O **elvtune** ajusta os valores deste algoritmo de forma que se possa obter o melhor resultado possível dentro de uma determinada finalidade. Para quem estiver usando o kernel padrão do Red Hat 9 (versão 2.4.22)[32], os valores mostrados abaixo são bastante interessantes na utilização com SGBD:

```
/sbin/elvtune -r 64 -w 8192 /dev/hda
```

Os parâmetros configuram a latência de leitura para 64 blocos, e

de gravação para 8192. Estes valores são recomendados para se trabalhar com kernel 2.4.22, mas já foi também utilizado com sucesso no kernel 2.6.5. É necessário editar novamente o arquivo *rc.local* e acrescentar a seguinte linha:

```
/sbin/elvtune -r 64 -w 8192 /dev/hda
```

Nessa seção foi mostrada uma noção básica de configuração do sistema operacional Linux, visando obter um maior desempenho com servidores de banco de dados. Vimos que não existe uma regra única para configuração, e por isso é necessário entender como o Linux trabalha. A configuração correta das partições, escolha do sistema de arquivos ideal e *tunning* do HD são essenciais para que se ganhe eficiência no SGBD.

5 Conclusão

Nesse trabalho foram abordados alguns procedimentos de como administrar um SGBD PostgreSQL de modo a obter uma maior satisfação dos envolvidos (Usuário final, administrador de SGBD, gerente da empresa por exemplo). Foi discutido o uso de algumas ferramentas de administração e chegou-se à conclusão de que a **EMS PostgreSQL Manager**, apesar de ser uma ferramenta paga, oferece todo suporte necessário à administração do SGBD. Foram vistas em detalhes algumas das suas principais funções e como utilizá-las. Também foram abordadas técnicas de tunning de HD padrão IDE, e foi visto, através de testes de performance, o ganho obtido com algumas otimizações feitas no servidor de banco de dados. Também foi discutido qual seria a melhor escolha para sistema de arquivos de modo a obter um acesso mais rápido aos dados do banco sem esquecer da consistência dos dados e segurança em relação a falhas.

Como trabalho futuro, sugere-se um estudo semelhante com sistemas de gerenciamento de banco de dados líderes de mercado, como o Oracle[33], SQL Server[34] e DB2[35].

Referências

[1]ftp://ftp.sra.co.jp/pub/cmd/postgres/pgbench/pgbench-1.1.tar.gz

(Arquivo README dentro do pacote)

[2]http://e2fsprogs.sourceforge.net/ext2.html

[3]http://fsbench.netnation.com

[4]http://olstrans.sourceforge.net/release/OLS2000-ext3/OLS2000-ext3.html

[5]http://oss.sgi.com/projects/xfs/

[6]http://phppgadmin.sourceforge.net/

[7]http://switch.dl.sourceforge.net/sourceforge/phppgadmin/phpPgAdmin-3.5.3.tar.gz

(Arquivo de licença desse pacote)

[8]http://techdocs.postgresql.org/guides/GUITools

[9]http://www.guiadohardware.net/artigos/020/

[10]http://www.ibm.com/link/oss.software.ibm.com/redirect.shtml/jfs/

[11]http://www.lozano.eti.br/palestras/tunning-pgsql.pdf

[12]http://www.namesys.com

[13]http://www.opensource.org/licenses/gpl-license.php

[14]http://www.pgadmin.org/

[15]http://www.postgresql.org

[16]http://www.postgresql.org.br/referencia/sql-analyze.html

[17]http://www.postgresql.org.br/referencia/sql-vacuum.html

[18]http://www.redhat.com/

[19]http://www.sqlmagazine.com.br/Colunistas/Methanias/04_AdministracaoBD.asp

[20]http://www.sqlmanager.net/products/postgresql/manager

[21]http://www.sqlmanager.net/products/postgresql/manager/features

[22]http://www.vivaolinux.com.br/artigos/verArtigo.php?codigo=2544

[23]SQL Magazine, Edição 12, Ano 1

[24]http://www.gnu.org/

[25]http://www.php.net/

[26] A. Silberschatz, H. Korth, S. Sudarshan: Database System Concepts, 4th Edition 2002.

[27]http://www.linux.org/

[28]http://fedora.redhat.com/

[29]http://www.slackware.com/

[30]http://www.conectiva.com.br/

[31]http://www.computerhope.com/msdos.htm

[32]http://www.redhat.com/

[33]http://www.oracle.com/

[34]http://www.microsoft.com/sql/

[35]http://www-306.ibm.com/software/data/db2/