Módulo II

2) Entendendo e trabalhando com CLUSTERS

1 - Iniciando e parando o serviço do PostgreSQL
2 - Entendendo os Tablespaces
3 - Criação de bancos de dados
4 - Removendo bancos de dados
5 - Bancos de dados de template/modelo
6 - Entendendo o layout físico dos bancos de dados

A palavra "cluster" pode significar várias coisas diferentes dependendo do contexto:

1) Existe o cluster de tabelas:
http://www.postgresql.org/docs/8.3/static/sql-cluster.html ou
http://www.postgresql.org/docs/8.3/static/app-clusterdb.html

2) Existe o cluster do postgresql, que consiste em todos os arquivos que compõem um conjunto de bancos de dados administrados por uma instância do postgresql que sobe em uma porta só dele. Este cluster é criado pelo initdb:
http://www.postgresql.org/docs/8.3/static/app-initdb.html

3) Existe o conceito de cluster de bancos de dados que são várias instâncias de bancos de dados rodando em máquinas diferentes e se comportando como se fossem uma só. O PostgreSQL não possui uma implantação deste tipo.

4) Existe do pg_cluster que é uma implementação de replicação e é parecido com um cluster, mas na verdade é uma replicação:
http://pgfoundry.org/projects/pgcluster/

Para ver a diferença entre cluster e replicação, veja:
http://www.midstorm.org/~telles/2007/08/24/cluster-replicacao/

Cluster de Versões
1) Os pacotes do Debian fazem isso automaticamente, de forma limpa.
Você instala o 8.1 e 8.2 na mesma máquina e ele sobe cada um em uma porta diferente. Vale a pena conferir para ver como ele organiza as coisas, particularmente a estrutura de diretórios é o init.d.

(Fábio Telles na lista pgbr-geral.)

Instalação do PostgreSQ-8.3 através dos repositórios do Ubuntu 7.10

Para instalar o Servidor, acesse um terminal e digite:
sudo apt-get install postgresql-8.3 (com este pacote serão instalados o server, o client e o common)
sudo apt-get install postgresql-contrib-8.3
sudo apt-get install postgresql-doc-8.3 (ficará em /usr/share/doc/postgresql-doc-8.3/html/index.html)

Em máquinas que somente precisam ter o cliente instalado use:
sudo apt-get install postgresql-client-8.3
sudo apt-get install postgresql-doc-8.3
sudo apt-get install pgadmin3

Após instalar o 8.3 repita os passos para instalar também a versão 8.2, de forma que fiquemos com dois clusters instalados, um da versão 8.3 e outro da versão 8.2. O Ubuntu gerencia bem isso e de forma transparente. Geralmente o primeiro instalado fica na porta 5432 e o segunda na 5433.

Após instalar o pgadmin execute o script abaixo para habilitar algumas funções úteis:
sudo -u postgres psql -d postgres < /usr/share/postgresql/8.3/contrib/adminpack.sql

Para acessar a console do psql:

Para acessar o 8.3 use
sudo -u postgres psql

Para acessar o 8.2 use
sudo -u postgres psql -p 5433

Ou mais adequadamente com:
sudo -u postgres /usr/lib/postgresql/8.2/bin/psql -p 5433

Observe que não será solicitada a senha do postgres. Isso devido ao método de autenticação usado por padrão no /etc/postgresql/8.3/main/pg_hba.conf, que identifica o usuário do sistema operacional, sem senha.
Na primeira vez que acessar altere a senha do super-usuário:

ALTER ROLE postgres WITH ENCRYPTED PASSWORD 'postgres';
Para oferecer mais segurança no acesso remoto, já que localmente não requer senha.

Para constatar os dois serviços no ar abra um terminal execute:

ps ax | grep post

Diretórios quando instalado pelos repositórios:

- Diretório base (com bancos e outros) - /var/lib/postgresql/8.3/main
- Arquivos de configuração (pg_hba.conf, pg_ident.conf e postgresql.conf) - /etc/postgresql/8.3/main
- Diretório de gerenciameto dos clusters - /etc/postgresql-common
- Diretório dos binários - /usr/lib/postgresql/8.3/bin

Exercício

Importar o banco de dados de CEP existente no CD ou em:
http://tudoemum.ribafs.net/includes/cep_brasil.sql.bz2

Faça o download e copie para a pasta /home/aluno
Descompacte:

Abra o gerenciador de arquivos (locais – pasta pessoal). Clicando com o botão direito e extrair aqui.

Este banco de CEPs (do Brasil), não é completo nem está atualizado mas é uma boa base de testes e contém os CEPs das grandes cidades e capitais.

- Mude as permissões do arquivo cep_brasil_unique.csv para que o usuário postgres tenha permissão de acessá-lo:
sudo chown postgres:postgres cep_brasil_unique.csv

Experimente criar o banco na versão 8.3 com codificação latin1.
Ele não criará, informando incompatibilidade com as configurações do servidor/sistema operacional.
Para ter compatibilidade com latin1 no 8.3 devemos criar o clustar passando a codificação pelo comando initdb.

```
sudo -u postgres psql
create database cep_brasil with encoding 'latin1';

\c cep_brasil
create table cep_full (
        cep char(8),
        tipo char(72),
        logradouro char(70),
        bairro char(72),
        municipio char(60),
        uf char(2)
);
```

Execute os comandos abaixo e veja as informações:
\l
Depois:
\d

Importar o script de ceps para a tabela que acabamos de criar:
\copy cep_full from /home/ribafs/cep_brasil_unique.csv

Então, para ativar o cronômetro, execute:
\timing

Faça uma consulta que retorne apenas o registro com o CEP da sua rua:
select logradouro from cep_full where cep = '60420440';

Dual Core 2.2, 2GB RAM e Ubuntu 7.10 gastou 6711,204 ms.
Centrino 1.86 1GB de RAM e Ubuntu 7.10 gastou 8476.877 ms

Caso repita a consulta, levará apenas 514 ms na última máquima, pois está no cache.

Agora adicione uma chave primária na tabela:
ALTER TABLE cep_full add constraint cep_pk primary key (cep);

Então repita a mesma consulta anterior e veja a diferença de desempenho por conta do índice adicionado. Tenha o cuidado de executar o comando ANALYZE antes da consulta.

Aqui gastou somente 1.135ms.

1 - Iniciando e parando o serviço do PostgreSQL

Parar:
sudo /etc/init.d/postgresql-8.3 stop

Iniciar:
sudo /etc/init.d/postgresql-8.3 start

Reiniciar:
sudo /etc/init.d/postgresql-8.3 restart

Para o 8.2 é de forma semelhante.

Observação
Estamos usando o script oferecido pela distribuição ou então aquele dos contribs que acompanha os fontes.

Manualmente:

sudo /usr/local/pgsql/bin/pg_ctl -D /usr/local/pgsql/data start

sudo /usr/local/pgsql/bin/pg_ctl -D /usr/local/pgsql/data stop

sudo /usr/local/pgsql/bin/pg_ctl -D /usr/local/pgsql/data restart

2 - Entendendo os Tablespaces

Como o nome sugere, um espaço destinado a armazenar tabelas. Pode ser criado para proteção/segurança de algumas tabelas (que ficam em localização diferente do restante dos bancos) como para balancear carga e também para otimizar dezempenho de apenas algumas tabelas ou bancos que requerem maior desempenho.

Utilizando espaços de tabelas, o administrador pode controlar a organização em disco da instalação do PostgreSQL. É útil pelo menos de duas maneiras:

Primeira: se a partição ou volume onde o agrupamento foi inicializado ficar sem espaço, e não puder ser estendido, pode ser criado um espaço de tabelas em uma partição diferente e utilizado até que o sistema possa ser reconfigurado.

Segunda: os espaços de tabelas permitem que o administrador utilize seu conhecimento do padrão de utilização dos objetos de banco de dados para otimizar o desempenho. Por exemplo, um índice muito utilizado pode ser colocado em um disco muito rápido com alta disponibilidade, como uma unidade de estado sólido. [4] Ao mesmo tempo, uma tabela armazenando dados históricos raramente utilizados, ou que seu desempenho não seja crítico, pode ser armazenada em um sistema de disco mais barato e mais lento.

Para definir um espaço de tabelas é utilizado o comando CREATE TABLESPACE como, por exemplo:

```
CREATE TABLESPACE nometablespace [ OWNER nomedono ] LOCATION 'diretório';

CREATE TABLESPACE area_veloz LOCATION '/mnt/sda1/postgresql/data';

Ao criar um banco, uma tabela ou um índice podemos escolher a qual
tablespace pertencerá, usando o parâmetro tablespace.
```

Criar Novo Cluster

Caso sinta necessidade pode criar outros clusters, especialmente indicado para grupos de tabelas com muito acesso.

O comando para criar um novo cluster é:

\h create tablespace

Comando: CREATE TABLESPACE
Descrição: define uma nova tablespace

Sintaxe:
CREATE TABLESPACE nome_tablespace [OWNER usuário] LOCATION 'diretório'

Exemplo:

CREATE TABLESPACE ncluster OWNER usuário LOCATION '/usr/local/pgsql/nc';
CREATE TABLESPACE ncluster [OWNER postgres] LOCATION 'c:\\ncluster';

Importante: O diretório deve estar vazio e pertencer ao usuário.

Criando um banco no novo cluster:
CREATE DATABASE bdcluster TABLESPACE = ncluster;

Obs: Podem existir numa mesma máquina vários agrupamentos de bancos de dados (cluster) gerenciados por um mesmo ou por diferentes postmasters.
Se usando tablespace o gerenciamento será de um mesmo postmaster, se inicializados por outro initdb será por outro.

Setar o Tablespace default:
SET default_tablespace = tablespace1;

Listar os Tablespaces existentes:
\db
SELECT spcname FROM pg_tablespace;

Na documentação oficial em português do PG 8.0:
http://pgdocptbr.sourceforge.net/pg80/manage-ag-tablespaces.html
http://pgdocptbr.sourceforge.net/pg80/sql-createtablespace.html
E diversos outros capítulos fazem referência.
Na documentação do 8.3 em inglês:
http://www.postgresql.org/docs/8.3/interactive/manage-ag-tablespaces.html
http://www.postgresql.org/docs/8.3/interactive/sql-createtablespace.html
http://www.postgresql.org/docs/8.3/interactive/sql-droptablespace.html
http://www.postgresql.org/docs/8.3/interactive/sql-altertablespace.html
O Capítulo 5 do Livro Dominando o PostgreSQL aborda os tablespaces.

3 - Criação de bancos de dados

No prompt de comando:
createdb -p numeroporta -h nomehost -E LATIN1 -e nomebanco

Mais Detalhes: http://www.postgresql.org/docs/8.3/interactive/app-createdb.html

Como SQL dentro do psql ou em outro cliente:
CREATE DATABASE banco OWNER dono TABLESPACE nometablespace ENCODING 'LATIN1';

Mais Detalhes em: http://www.postgresql.org/docs/8.3/interactive/sql-createdatabase.html

4 - Removendo bancos de dados

DROP DATABASE nomebanco;
Este comando não pode ser desfeito nem executado dentro de uma transação.

Mais detalhes: http://www.postgresql.org/docs/8.3/interactive/sql-dropdatabase.html

dropdb -U usuario nomebanco
Detalhes em: http://www.postgresql.org/docs/8.3/interactive/app-dropdb.html

5 - Bancos de dados de template/modelo

Na verdade o comando CREATE DATABASE funciona copiando um banco de dados existente. Por padrão, copia o banco de dados padrão do sistema chamado template1. Portanto, este banco de dados é o "modelo" a partir do qual os novos bancos de dados são criados. Se forem adicionados objetos ao template1, estes objetos serão copiados nos próximos bancos de dados de usuário criados. Este comportamento permite modificar localmente o conjunto padrão de objetos nos bancos de dados. Por exemplo, se for instalada a linguagem procedural PL/pgSQL em template1, esta se tornará automaticamente disponível em todos os próximos bancos de dados dos usuários sem que precise ser feito qualquer procedimento adicional na criação dos bancos de dados.

Existe um segundo banco de dados padrão do sistema chamado template0. Este banco de dados contém os mesmos dados contidos inicialmente em template1, ou seja, contém somente os objetos padrão pré-definidos pela versão do PostgreSQL. **O banco de dados template0 nunca deve ser modificado após a execução do utilitário initdb**. ***Instruindo o comando CREATE DATABASE para copiar template0 em vez de template1, pode ser criado um banco de dados de usuário "intacto", não contendo nenhuma adição feita ao***

banco de dados template1 da instalação local. É particularmente útil ao se restaurar uma cópia de segurança feita por pg_dump: o script da cópia de segurança deve ser restaurado em um banco de dados intocado, para garantir a recriação do conteúdo correto da cópia de segurança do banco de dados, sem conflito com as adições que podem estar presentes em template1.

Para criar um banco de dados copiando template0 deve ser utilizado:

CREATE DATABASE nome_do_banco_de_dados TEMPLATE template0;
a partir do ambiente SQL, ou

createdb -T template0 nome_do_banco_de_dados
a partir do interpretador de comandos.

É possível criar bancos de dados modelo adicionais e, na verdade, pode ser copiado qualquer banco de dados do agrupamento especificando seu nome como modelo no comando CREATE DATABASE. Entretanto, é importante compreender que não há intenção (ainda) que este seja um mecanismo tipo "COPY DATABASE" de uso geral. Em particular, é essencial que o banco de dados de origem esteja inativo (nenhuma transação em andamento alterando dados) durante a operação de cópia. O comando CREATE DATABASE verifica se nenhuma sessão (além da própria) está conectada ao banco de dados de origem no início da operação, mas não garante que não possa haver alteração durante a execução da cópia, resultando em um banco de dados copiado inconsistente.

Portanto, recomenda-se que os bancos de dados utilizados como modelo sejam tratados como somente para leitura.

Após preparar um banco de dados modelo, ou fazer alguma mudança em um deles, é recomendado executar o comando VACUUM FREEZE ou VACUUM FULL FREEZE neste banco de dados. Se for feito quando não houver nenhuma outra transação aberta no mesmo banco de dados, é garantido que todas as linhas no banco de dados serão "congeladas" e não estarão sujeitas a problemas de recomeço do ID de transação. Isto é particularmente importante em um banco de dados que terá datallowconn definido como falso, uma vez que não será possível executar a rotina de manutenção VACUUM neste banco de dados. Para obter informações adicionais deve ser consultada a Seção 21.1.3 do manual oficial em português do Brasil em:
http://pgdocptbr.sourceforge.net/pg80/maintenance.html#VACUUM-FOR-WRAPAROUND .

Nota: Os bancos de dados template1 e template0 não possuem qualquer status especial além do fato do nome template1 ser o nome padrão para banco de dados de origem do

comando CREATE DATABASE, e além de ser o banco de dados padrão para se conectar utilizado por vários programas, como o createdb. Por exemplo, template1 pode ser removido e recriado a partir de template0 sem qualquer efeito prejudicial. Esta forma de agir pode ser aconselhável se forem adicionadas, por descuido, coisas inúteis ao template1. Na versão 8 apareceu um novo template, que é uma cópia do template1, é o **postgres,** este a partir de então é o template default.

Recriação do banco de dados template1

Neste exemplo o banco de dados template1 é recriado. Deve ser observado na seqüência de comandos utilizada que não é possível remover o banco de dados template1 conectado ao mesmo, e enquanto este banco de dados estiver marcado como modelo no catálogo do sistema pg_database.

Para recriar o banco de dados template1 é necessário: se conectar a outro banco de dados (teste neste exemplo); atualizar o catálogo pg_database para que o banco de dados template1 não fique marcado como um banco de dados modelo; remover e criar o banco de dados template1; conectar ao banco de dados template1; executar os comandos VACUUM FULL e VACUUM FREEZE; atualizar o catálogo do sistema pg_database para que o banco de dados template1 volte a ficar marcado como um banco de dados modelo.

Abaixo está mostrada a seqüência de comandos utilizada:

```
template1=#  DROP DATABASE template1;
ERRO:  não é possível remover o banco de dados aberto atualmente
template1=#  \c teste
Conectado ao banco de dados "teste".
teste=#  DROP DATABASE template1;
ERRO:  não é possível remover um banco de dados modelo
teste=#  UPDATE pg_database SET datistemplate=false WHERE datname='template1';
UPDATE 1
teste=#  DROP DATABASE template1;
DROP DATABASE
teste=#  CREATE DATABASE template1 TEMPLATE template0 ENCODING 'latin1';
CREATE DATABASE
teste=#  \c template1
Conectado ao banco de dados "template1".
template1=#  VACUUM FULL;
VACUUM
template1=#  VACUUM FREEZE;
VACUUM
template1=#  UPDATE pg_database SET datistemplate=true WHERE datname='template1';
UPDATE 1
```

Mais detalhes em: http://pgdocptbr.sourceforge.net/pg80/manage-ag-templatedbs.html

6 - Entendendo o layout físico dos bancos de dados

Formato de armazenamento no nível de arquivos e diretórios.
Todos os dados necessários para um agrupamento de bancos de dados são armazenados dentro do diretório de dados do agrupamento. Podem existir na mesma máquina vários agrupamentos, gerenciados por diferentes postmaster.

Quando instalamos pelos fontes e não alteramos o default ficam em /usr/local/pgsql/data.
O diretório data contém vários subdiretórios e arquivos de controle, conforme mostrado na tabela abaixo. Além destes itens requeridos, os arquivos de configuração do agrupamento postgresql.conf, pg_hba.conf e pg_ident.conf são tradicionalmente armazenados no data (embora a partir da versão 8.0 do PostgreSQL seja possível mantê-los em qualquer outro lugar).

Conteúdo de diretório data (lembre: algumas distribuições usam outro diretório)

Item	Descrição
PG_VERSION	Arquivo contendo o número de versão principal do PostgreSQL
base	Subdiretório contendo subdiretórios por banco de dados
global	Subdiretório contendo tabelas para todo o agrupamento, como pg_database
pg_clog	Subdiretório contendo dados sobre status de efetivação de transação
pg_subtrans	Subdiretório contendo dados sobre status de subtransação
pg_tblspc	Subdiretório contendo vínculos simbólicos para espaços de tabelas
pg_xlog	Subdiretório contendo os arquivos do WAL (registro prévio da escrita)
postmaster.opts	Arquivo contendo as opções de linha de comando com as quais o postmaster foi inicializado da última vez
postmaster.pid	Arquivo de bloqueio contendo o PID corrente do postmaster, e o ID do segmento de memória compartilhada (não mais presente após o postmaster ser parado)

Em caso de problema ao tentar iniciar o PostgreSQL remova o postmaster.pid.
Para cada banco de dados do agrupamento existe um subdiretório dentro de PGDATA/base, com nome correspondente ao OID do banco de dados em pg_database. Este subdiretório é o local padrão para os arquivos do banco de dados; em particular, os catálogos do sistema do banco de dados são armazenados neste subdiretório.
Cada tabela e índice é armazenado em um arquivo separado.
Mais detalhes em: http://pgdocptbr.sourceforge.net/pg80/storage.html e em http://www.postgresql.org/docs/8.3/interactive/storage-file-layout.html