5) Manutenção do Servidor do PostgreSQL e dos Bancos

5.1) Instalando o PostgreSQL no Windows
5.2) Instalando o PostgreSQL no Linux
5.3) Backup Lógico e restauração de bancos de dados
5.4) Backup e restauração física offline
5.5) Usando o Vacuum e Analyze
5.6) Atualizando e migrando entre versões
5.7) Manutenção no arquivo de logs (registro)

5.1) Instalando o PostgreSQL no Windows

Abordado no módulo 1.

5.2) Instalando o PostgreSQL no Linux

Obs.: Veja a aula extra para instalação no Linux Mandriva e no FreeBSD.

Instalando pelos binários dos Repositórios da Distribuição

Atualmente todas as grandes distribuições Linux tem o PostgreSQL em seus repositórios e algumas trazem a última versão ou a penúltima, como o Slackware e o Ubuntu.

Instalaremos em uma distribuição filha da distribuição Debian, no nosso caso a Ubuntu.
Caso queira instalar em uma distribuição Linux com outra origem procure pelo repositório da mesma e pelos programas de instalação.

Aqui adotarei o apt-get para a instalação (outra opção é o Synaptic).

Instalando pelo terminal:

sudo apt-get install postgresql-8.3

Com isso teremos a versão 8.3 instalada, configurada e o servidor no ar pronto para ser usado.

Onde ficam os arquivos:

Diretório de dados - /var/lib/postgresql/8.3/main/base
Diretório do pg_hba.conf e do postgresql.conf - /etc/postgresql/8.3/main
autovacuum.conf - /etc/postgresql-common
pg_dump, pg_dumpall, psql e outros binários - /usr/bin

Testando via console

sudo -u postgres psql

Caso queira ter acesso à linha de comando como usuário postgres, conceda uma senha:

sudo passwd postgres

Instalação Através dos Fontes

Numa instalação através dos fontes, fazemos o download do código fonte do PostgreSQL do site oficial, então configuramos e compilamos.

Faremos uma instalação do PostgreSQL 8.3.1 no Linux Ubuntu 7.10.

Pré-requisitos para instalação do PostgreSQL num UNIX em geral:

make do GNU (gmake ou make)
compilador C, preferido GCC mais recente
gzip
biblioteca readline (para psql)
gettext (para NLS)
kerberos, openssl e pam (opcionais, para autenticação)

Pré-requisitos para a instalação com os fontes no Ubuntu:

sudo apt-get install build-essential
sudo apt-get install libreadline5-dev
sudo apt-get install zlib1g-dev
sudo apt-get install gettext

Site de Pacotes do Ubuntu

Caso alguma não seja encontrada nos repositórios, faça o download do site de pacotes do Ununtu, que fica aqui: http://packages.ubuntu.com/

Sobre os pacotes .deb

Os pacotes deste site devem ser rigorosamente para a versão do Ubuntu que você está usando (7.10 é a versão atual) e para a arquitetura do seu hardware (geralmente i386).
São pacotes .deb para distribuições Debian ou oriundas, no caso, específicos para o Ubuntu.

Instalar pacotes .deb com:

sudo dpkg -i nomepacote.deb

Ou simplesmente abrindo o gerenciador de arquivos (nautilus) e um duplo clique sobre o arquivo.

Obs.: estes pacotes podem mudar de nome devido ao aparecimento de novas versões.

Download dos fontes do PostgreSQL

http://www.postgresql.org/ftp/source/
Faça o download da versão mais recente no formato tar.gz (fique à vontade para outro formato).

Descompactando

Acesse a console e descompacte em /usr/local/src com:
sudo tar zxpvf postgresql-8.3.0.tar.gz -C /usr/local/src

Acessando os fontes
cd /usr/local/src/postgresql-8.3.0

Caso execute o comando 'ls' verá os arquivos fontes prontos para serem convertidos em binários.

Verificando se existem as bibliotecas necessárias e configurando para a conpilação:

sudo ./configure

Compilando (este demora um pouco)

sudo make

Instalando o PostgreSQL

sudo make install

Criando o grupo (caso ainda não exista)

sudo groupadd postgres

Criando o usuário, adicionando ao grupo criado e setando sua home

sudo useradd -g postgres -d /usr/local/pgsql postgres

Criando o diretório do cluster padrão

sudo mkdir /usr/local/pgsql/data

Tornando o usuário postgres o dono do cluster

sudo chown postgres:postgres /usr/local/pgsql/data

Atribindo uma senha ao usuário (super-usuário) postgres

sudo passwd postgres

Alternando do usuário atual para o usuário postgres

su – postgres

Criando o cluster, os scripts de configuração, os logs, etc

bin/initdb -D /usr/local/pgsql/data

Iniciando manualmente o servidor

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

Criando um banco de testes

bin/createdb teste

Acessando a console interativa do PostgreSQL (psql)

bin/psql teste

Obs.: Isso criará um cluster com os locales e codificação padrão do SO e da nossa versão atual do PostgreSQL.

No nosso caso, PostgreSQL 8.3.1 e Ubuntu 7.1 em português do Brasil teremos algo assim:

pt_BR.UTF-8, ou seja, locales pt_BR (português do Brasil) e codificação de caracteres UTF-8.

Confira executando o meta-comando, que Lista os bancos:

\l

Serão listados: template0, tempalte1, postgres e teste. Todos com codificação UTF-8.

A codificação UTF-8 é uma codificação que atende as exigênciasde uma empresa ou instituição globalizada ou que tem possibilidades de crescer e manter alguma informação diferente das suportadas pelo nosso latin1 (que neste aspecto está obsoleto).

Além de trazer seus bancos, por default em UTF-0, esta versão do PostgreSQL não mais dá suporte à criação de bancos com a codificação LATIN1, que é o ISO-8859-1.

Mas se mesmo assim desejar criar bancos com codificação LATIN1. Nesta etapa da instalação do PostgreSQL podemos criar um segundo ou vários clusters com suporte a LATIN1.

Copiando o Script de Inicialização (torna mais prático iniciar e parar o servidor)

sudo cp /usr/local/src/postgresql-8.3.0/contrib/start-script/linux /etc/init.d/postgresql-8.3.0

Tornar executável

sudo chmod u+x /etc/init.d/postgresql-8.3.0

postmaster ou pg_ctl – iniciam o processo do servidor responsável por escutar por pedidos de conexão.

5.3) Backup Lógico e restauração de bancos de dados

pg_dump – faz o backup de um banco de dados do PostgreSQL em um arquivo de script (texto puro) ou de outro tipo (tar ou outro).

pg_dump opções banco

São feitas cópias de segurança consistentes, mesmo que o banco de dados esteja sendo utilizado ao mesmo tempo. O pg_dump não bloqueia os outros usuários que estão acessando o banco de dados (leitura ou escrita).

As cópias de segurança podem ser feitas no formato de *script (p)*, *customizado (c)* ou *tar (t)*.
O formato personalizado, por padrão é compactado.
Para restaurar a partir de scripts em texto devemos usar o comando psql.
Para restaurar scripts feitos com formato personalizado (-Fc) e tar (-Ft) devemos usar o pg_restore.
Por default customizado é compactado.

Algumas Opções

-a (backup somente dos dados)

-b (incluir objetos grandes no backup)

-c (incluir comandos para remover – DROP, os objetos do banco antes de criá-lo)

-C (criar o banco e conectar ao mesmo)

-d (salvar os registros usando o comando INSERT ao invés do COPY). Reduz o desempenho. Somente se necessário (migração de SGBDs por exemplo).

-D (salvar os registros usando o comando INSERT ao invés do COPY explicitando os nomes das colunas).

-E codificação (passar uma codificação no backup)

-f arquivo (gerar backup para um arquivo)

-F formato (formato de saída. p – para texto plano, que é o padrão, c – para sustom e t – para tar)

-i (ignorar a diferença de versão entre o pg_dump e o servidor)

-n esquema (backup apenas do esquema citado)

-N esquema (no backup NÃO incluir o esquema citado)

-s (backup somente a estrutura do banco, sem os dados)

-t tabela (salvar somente a tabela, sequência ou visão citada)

-T tabela (NÃO realizar o backup da referida tabela)

--disable-triggers (aplica-se somente quando salvando só os dados. Para desativar as triggers enquanto os dados são carregados).

-h host (host ou IP)

-p porta (porta)

-U usuário (usuário)

-W (forçar solicitação de senha)

Mais detalhes em:
http://pgdocptbr.sourceforge.net/pg82/app-pgdump.html

Observações

Se o agrupamento de bancos de dados tiver alguma adição local ao banco de dados template1, deve-se ter o cuidado de restaurar a saída do pg_dump em um banco de dados totalmente vazio; senão, poderão acontecer erros devido a definições duplicadas dos objetos adicionados. Para criar um banco de dados vazio, sem nenhuma adição local, deve-se fazê-lo a partir de template0, e não de template1 como, por exemplo:

```
CREATE DATABASE banco WITH TEMPLATE template0;
```

O backup no formato tar tem limitação para cada tabela, que devem ter no máximo 8GB no formato texto puro. O total dos objetos não tem limite de tamanho mas seus objetos sim.

É aconselhável executar o ANALYZE após restaurar de uma cópia de segurança para garantir um bom desempenho.

O pg_dump também pode ler bancos de dados de um PostgreSQL mais antigo, entretanto geralmente não consegue ler bancos de dados de um PostgreSQL mais recente.

Exemplos

Para salvar o banco de dados chamado meu_bd em um arquivo de script SQL:

```
pg_dump -U postgres meu_bd > bd.sql
```

Para restaurar este script no banco de dados (recém criado) chamado novo_bd:

```
dropdb novo_bd
createdb novo_bd
psql -U postgres -d novo_bd -f bd.sql
```

Para efetuar backup do banco de dados no script com formato customizado:

```
pg_dump -U postgres -Fc meu_bd > bd.dump
```

Para recarregar esta cópia de segurança no banco de dados (recém criado) chamado novo_bd:

```
pg_restore -U postgres -d novo_bd bd.dump
```

Para salvar uma única tabela chamada minha_tabela:

```
pg_dump -U postgres -t minha_tabela meu_bd > bd.sql
```

Para salvar todas as tabelas cujos nomes começam por emp no esquema detroit, exceto a tabela chamada empregado_log:

```
pg_dump -U postgres -t 'detroit.emp*' -T detroit.empregado_log meu_bd > bd.sql
```

Para salvar todos os esquemas cujos nomes começam por leste ou oeste e terminam por gsm, excluindo todos os esquemas cujos nomes contenham a palavra teste:

```
pg_dump -U postgres -n 'leste*gsm' -n 'oeste*gsm' -N '*teste*' meu_bd > bd.sql
```

A mesma coisa utilizando a notação de expressão regular para unificar as chaves:

```
pg_dump -U postgres -n '(leste|oeste)*gsm' -N '*teste*' meu_bd > bd.sql
```

Para salvar todos os objetos do banco de dados, exceto as tabelas cujos nomes começam por ts_:

```
pg_dump -U postgres -T 'ts_*' meu_bd > bd.sql
```

Backup dos usuários e grupos e depois de todos os bancos:

pg_dumpall -g
pg_dump -Fc para cada banco

http://www.pgadmin.org/docs/1.6/backup.html

`pg_dumpall`

Salva todos os bancos de um agrupamento de bancos de dados do PostgreSQL em um único arquivo de script.

pg_dumpall opções

O pg_dumpall também salva os objetos globais, comuns a todos os bancos de dados (O pg_dump não salva estes objetos). Atualmente são incluídas informações sobre os usuários do banco de dados e grupos, e permissões de acesso aplicadas aos bancos de dados como um todo.

Necessário conectar como um superusuário para poder gerar uma cópia de segurança completa. Também serão necessários privilégios de superusuário para executar o *script* produzido, para poder adicionar usuários e grupos, e para poder criar os bancos de dados.
O pg_dumpall precisa conectar várias vezes ao servidor PostgreSQL (uma vez para cada banco de dados). Se for utilizada autenticação por senha, provavelmente será solicitada a senha cada uma destas vezes. Neste caso é conveniente existir o arquivo `~/.pgpass ou pgpass.conf(win)`

Algumas Opções

-a (realizar backup somente dos dados)

-c (insere comandos para remover os bancos de dados – DROP antes dos comandos para criá-los)

-d (realiza o backup inserindo comandos INSERT ao invés do COPY). Fica lenta.

-D (backup usando INSERT ao invés de COPY e explicitando os nomes de campos)

-g (salva somente os objetos globais do SGBD, ou seja, usuários e grupos)

e várias outras semelhante ao comando pg_dump.

Mais opções em: http://pgdocptbr.sourceforge.net/pg82/app-pg-dumpall.html

Exemplos

pg_dumpall -U postgres > todos.sql

Restaurando:

psql -U postgres -f todos.sql banco

Aqui o banco pode ser qualquer um, pois o script gerado pelo pg_dumpall contém os comandos para a criação dos bancos e conexão aos mesmos, de acordo com o original.

Backup Full Compactado (Linux)

pg_dumpall -U postgres | gzip > full.gz

cat full.gz | gunzip | psql template1

Descompactar e fazer o restore em um só comando:

gunzip -c backup.tar.gz | pg_restore -d banco

ou

cat backup.tar.gz | gunzip | pg_restore -d banco

(o cat envia um stream do arquivo para o gunzip que passa para o pg_restore)

Backup remoto de um banco:

pg_dump -h hostremoto -d nomebanco | psql -h hostlocal -d banco

Backup em multivolumes (volumes de 200MB):

pg_dump nomebanco | split -m 200 nomearquivo

m para 1Mega, k para 1K, b para 512bytes

Importando backup de versão anterior do PostgreSQL

Instala-se a nova versão com porta diferente (ex.: 5433) e conectar ambos

pg_dumpall -p 5432 | psql -d template1 -p 5433

No Windows

pg_dump -f D:\MYDB_BCP -Fc -x -h localhost -U postgres MYDB

========== SCRIPT BAT ==========

```
@echo off
rem (Nome do Usuário do banco para realizar o backup)
SET PGUSER=postgres
rem (Senha do usuário acima)
SET PGPASSWORD=1234
rem (Indo para a raiz do disco)
C:
rem (Selecionando a pasta onde será realizada o backup)
chdir C:\Sistemas\backup_postgresql
rem (banco.sql é o nome que definir para o meu backup
rem (Deletando o backup existente, só por precaução)
del banco.sql
echo "Aguarde, realizando o backup do Banco de Dados"
pg_dump -i -U postgres -b -o -f "C:\Sistemas\backup_postgresql\banco%Date%.sql" banco
rem (sair da tela depois do backup)
exit
```

Por default o psql continua caso encontre um erro, mas podemos configurá-lo para parar caso encontre um:

\set ON_ERROR_STOP

Script de Backup no Windows

========backup.bat=Formato: banco_dd-mm-aaaa_hh-mm=====

for /f "tokens=1,2,3,4 delims=/ " %%a in ('DATE /T') do set Data=%%b-%%c-%%d

for /f "tokens=1 delims=: " %%h in ('time /T') do set hour=%%h

for /f "tokens=2 delims=: " %%m in ('time /T') do set minutes=%%m

for /f "tokens=3 delims=: " %%a in ('time /T') do set ampm=%%a

set Hora=%hour%-%minutes%-%ampm%

pg_dump -U postgres controle_estoque -p 5222 -f "controle_estoque_%Data%_%Hora%.sql"

pg_restore - restaura um banco de dados do PostgreSQL a partir de um arquivo criado pelo pg_dump no formato custom (-Fc) ou tar (-Ft)

Importante: no Windows o pg_dumpall pedirá a senha para cada banco existente.

No pgpass.conf colocar * no campo bancodedados ou então faça uma cópia da linha para todos os bancos, inclusive tempalte0, template1 e postgres.
Ou não use pg_dumpall mas pg_dump apenas para os bancos desejados.

Na lista pgsql-general - http://archives.postgresql.org/pgsql-general/2005-06/msg01279.php

pg_restore

Objetivo - reconstruir o banco de dados no estado em que este se encontrava quando foi feito o backup. Para garantir a integridade dos dados, o banco a ser restaurado deve estar limpo, um banco criado recentemente e sem nenhum objeto ou registro.

-a (restaura somente os dados)
-c (exclui os objetos dos bancos antes de criar para restaurar)

-C (cria o banco antes de rstaurar)

-d banco (restaura dirtamente neste banco indicado). Caso esta opção não seja explicitada, a restauração será para o banco original do backup.

-e (encerra caso encontre erro. exit_on_error)

-f arquivo (rstaura deste arquivo)

-i (ignorar diferença de versões entre o pg_restore e o servidor)

-L lista (restaura somente os objetos presentes no arquivo lista e na ordem)

-n esquema (restaura somente o esquema especificado)

-P função (restaura somente esta função)

-s (restaura somente o esquema do banco, não os dados)

-t tabela (restaura somente a definição e/ou dados da tabela)

-T gatilho (restaura somente este gatilho)

-W (força a solicitação da senha)

-1 (a restauração ocorre em uma única transação. Tudo ocorrerá bem ou nada será salvo)

-h host

-p porta

-U usuário

Mais opções em: http://pgdocptbr.sourceforge.net/pg82/app-pgrestore.html

Para criar um banco de dados vazio, sem nenhuma adição local, deve-se fazê-lo partir de template0, e não de template1 como, por exemplo:

```
CREATE DATABASE banco WITH TEMPLATE template0;
```

Uma vez restaurado, é aconselhável executar o comando ANALYZE em todas as tabelas restauradas para que o otimizador possua estatísticas úteis.

Exemplos

Banco de dados meu_bd em um arquivo de cópia de segurança no formato personalizado:

```
pg_dump -U postgres -Fc meu_bd > db.dump
```

Para remover o banco de dados e recriá-lo a partir da cópia de segurança:

```
dropdb -U postgres meu_bd
pg_restore -U postgres -C -d postgres db.dump
```

O banco de dados especificado na chave -d pode ser qualquer banco de dados existente no agrupamento; o pg_restore somente utiliza este banco de dados para emitir o comando CREATE DATABASE para meu_bd. Com a chave -C, os dados são sempre restaurados no banco de dados cujo nome aparece no arquivo de cópia de segurança.

Para recarregar a cópia de segurança em um banco de dados novo chamado bd_novo:

```
createdb -U postgres -T template0 bd_novo
pg_restore -U postgres -d bd_novo db.dump
```

Deve ser observado que não foi utilizada a chave -C, e sim conectado diretamente ao banco de dados a ser restaurado. Também deve ser observado que o novo banco de dados foi clonado a partir de template0 e não de template1, para garantir que esteja inicialmente vazio.

pg_dumpall has lot of disadvantages compared to pg_dump -Fc, I don't see the point why one would want that.

What I'd recommend is to use pg_dumpall -g and pg_dump -Fc on each DB. Then get a copy of rdiff-backup for windows to create nice incremental backups. Wrap those 3 things in a cmd script and use the task scheduler to run it in a given interval.

Mais detalhes em:

http://www.postgresql.org/docs/8.3/interactive/backup.html

http://pgdocptbr.sourceforge.net/pg80/backup.html

5.4) Backup Físico offline (sistema de arquivos)

Uma estratégia alternativa para fazer cópia de segurança, é copiar diretamente os arquivos que o PostgreSQL usa para armazenar os dados dos bancos de dados.

Pode ser utilizada a forma preferida para fazer as cópias de segurança usuais dos arquivos do sistema como, por exemplo:

```
tar -cf copia_de_seguranca.tar /usr/local/pgsql/data
```

Entretanto, existem duas restrições que fazem com que este método seja impraticável ou, pelo menos, inferior ao pg_dump:

1. O servidor de banco de dados *deve* estar parado para que se possa obter uma cópia de segurança utilizável. Meias medidas, como impedir todas as conexões, não funcionam (principalmente porque o tar, e as ferramentas semelhantes, não capturam um instantâneo atômico do estado do sistema de arquivos em um determinado ponto no tempo). As informações sobre como parar o servidor podem ser encontradas na Seção 16.6. É desnecessário dizer que também é necessário parar o servidor antes de restaurar os dados.
2. Caso tenha se aprofundado nos detalhes da organização do sistema de arquivos do banco de dados, poderá estar tentado a fazer cópias de segurança ou restauração de apenas algumas determinadas tabelas ou bancos de dados a partir de seus respectivos arquivos ou diretórios. Isto *não* funciona, porque as informações contidas nestes arquivos possuem apenas meia verdade. A outra metade está nos arquivos de registro de efetivação `pg_clog/*`, que contêm o status de efetivação de todas as transações. O arquivo da tabela somente possui utilidade com esta informação. É claro que também não é possível restaurar apenas uma tabela e seus dados associados em `pg_clog`, porque isto torna todas as outras tabelas do agrupamento de bancos de dados inúteis. Portanto, as cópias de segurança do sistema de arquivos somente funcionam para a restauração completa de todo o agrupamento de bancos de dados.

Deve ser observado que a cópia de segurança do sistema de arquivos não será necessariamente menor que a do Método SQL-dump. Ao contrário, é mais provável que seja maior; por exemplo, o pg_dump não necessita fazer cópia de segurança dos índices, mas apenas dos comandos para recriá-los.

Mais detalhes em:

http://pgdocptbr.sourceforge.net/pg80/backup-file.html

http://www.postgresql.org/docs/8.3/interactive/backup-file.html

5.5) Usando o Vacuum e Analyze

Uso do Vacuum

VACUUM
Vacuum - limpa e opcionalmente analisa um banco de dados. Recupera a área de armazenamento ocupada pelos registros excluídas. Na operação normal do PostgreSQL os registros excluídos, ou tornados obsoletos por causa de uma atualização, não são fisicamente removidos da tabela; permanecem presentes até o comando VACUUM ser executado. Portanto, é necessário executar o comando VACUUM periodicamente, especialmente em tabelas freqüentemente atualizadas.

Sem nenhum parâmetro, o comando VACUUM processa todas as tabelas do banco de dados corrente. Com um parâmetro, o comando VACUUM processa somente esta tabela.
O comando VACUUM ANALYZE executa o VACUUM e depois o ANALYZE para cada tabela selecionada. Esta é uma forma de combinação útil para scripts de rotinas de manutenção. Para obter mais detalhes sobre o seu processamento deve ser consultado o comando ANALYZE.
O comando VACUUM simples (sem o FULL) apenas recupera o espaço, tornando-o disponível para ser reutilizado. Esta forma do comando pode operar em paralelo com a leitura e escrita normal da tabela, porque não é obtido um bloqueio exclusivo. O VACUUM FULL executa um processamento mais extenso, incluindo a movimentação das tuplas entre blocos para tentar compactar a tabela no menor número de blocos de disco possível. Esta forma é muito mais lenta, e requer o bloqueio exclusivo de cada tabela enquanto está sendo processada.

Parâmetros

FULL
Seleciona uma limpeza "completa", que pode recuperar mais espaço, mas é muito mais demorada e bloqueia a tabela no modo exclusivo.

FREEZE
Seleciona um "congelamento" agressivo das tuplas. Especificar FREEZE é equivalente a realizar o VACUUM com o parâmetro vacuum_freeze_min_age definido como zero. A opção FREEZE está em obsolescência e será removida em uma versão futura; em vez de utilizar esta opção deve ser definido o parâmetro vacuum_freeze_min_age. (adicionar ao postgresql.conf).
vacuum_freeze_min_age (integer)

Specifies the cutoff age (in transactions) that VACUUM should use to decide whether to replace transaction IDs with FrozenXID while scanning a table. The default is 100000000 (100 million). Although users can set this value anywhere from zero to 1000000000, VACUUM will silently limit the effective value to half the value of autovacuum_freeze_max_age, so that there is not an unreasonably short time between forced autovacuums. For more information see Seção 22.1.3.

A convenient way to examine this information is to execute queries such as SELECT relname, age(relfrozenxid) FROM pg_class WHERE relkind = 'r';
SELECT datname, age(datfrozenxid) FROM pg_database;

VERBOSE
Mostra, para cada tabela, um relatório detalhado da atividade de limpeza.

ANALYZE
Atualiza as estatísticas utilizadas pelo planejador para determinar o modo mais eficiente de executar um comando.

tabela
O nome (opcionalmente qualificado pelo esquema) da tabela específica a ser limpa. Por padrão todas as tabelas do banco de dados corrente.

coluna
O nome da coluna específica a ser analisada. Por padrão todas as colunas.

Executando o Vacuum Manualmente

Para uma tabela
VACUUM ANALYZE tabela;
VACUUM VERBOSE ANALYZE clientes;

Para todo um banco
\c nomebanco
VACUUM FULL ANALYZE;

Encontrar as 5 maiores tabelas e índices
SELECT relname, relpages FROM pg_class ORDER BY relpages DESC LIMIT 5;

Ativando o daemon do auto-vacuum
Iniciando na versão 8.1 é um processo opcional do servidor, chamado de autovacuum daemon, cujo uso é para automatizar a execução dos comandos VACUUM e ANALYZE.

Roda periodicamente e checa o uso em baixo nível do coletor de estatísticas.
Só pode ser usado se stats_start_collector e stats_row_level forem alterados para true.

Veja detalhes na aula 5 do módulo 2 (Monitorando as Atividades do Servidor do PostgreSQL).

Por default será executado a casa 60 segundos. Para alterar descomente e mude a linha:
autovacuum_naptime = 60

Autovacuum

Assim que você entra em produção no 8.0, você vai querer fazer um plano de manutenção incluindo VACUUMs e ANALYZEs. Se seus bancos de dados envolvem um fluxo contínuo de escrita de dados, mas não requer a maciças cargas e apagamentos de dados ou freqüentes reinícios, isto

significa que você deve configurar o pg_autovacuum. Isto é melhor que agendar vaccuns porque:

* Tabelas sofrem o vaccum baseados nas suas atividades, excluindo tabelas que apenas sofrem leituras.

* A freqüência dos vaccums cresce automaticamente com o crescimento da atividade no banco de dados.

* É mais fácil calcular o mapa de espaço livre e evitar o inchaço do banco de dados.

Configurando o autovacuum requer a fácil compilação de um módulo do diretório contrib/pg_autovacuum da fonte do seu PostgreSQL (usuários Windows devem procurar o autovacuum incluído no pacote do instalador). Você liga as estatísticas de configuração detalhadas no README. Então você inicia o autovacuum depois de o PostgreSQL ser iniciado como um processo separado; ele será desligado automaticamente quando o PostgreSQL desligar.

As configurações padrões do autovacuum são muito conservadores, imagino, e são mais indicadas para bancos de dados muito pequenos. Eu geralmente uso algo mais agressivo como:

-D -v 400 -V 0.4 -a 100 -A 0.3

Isto irá rodar o vaccum nas tabelas após 400 linhas + 40% da tabela ser atualizada ou apagada e irá rodar o analyze após 100 linhas + 30% da tabelas sofres inserções, atualizações ou ser apagada. As configurações acima também me permitem configurar o meu max_fsm_pages para 50% das páginas de dados com a confiança de que este número não será subestimado gerando um inchaço no banco de dados. Nós atualmente estamos testando várias configurações na OSDL e teremos mais informações em breve.

Note que você também pode usar o autovacuum para configurar opções de atraso ao invés de configura-lo no PostgreSQL.conf. O atraso no Vaccum pode ser de vital importância em sistemas que tem tabelas e índices grandes; em último caso pode parar uma operação importante.

Existem infelizmente um par de limitações sérias para o autovacuum no 8.0 que serão eliminadas em versões futuras:

* Não tem memória de longa duração: autovacuum esquece toda a sua atividade quando você reinicia o banco de dados. Então se você reinicia regularmente, você deve realizar um VACUUM ANALYZE em todo o banco de dados imediatamente antes ou depois.

* Preste atenção em quanto o servidor está ocupado: há planos de checar a carga do servidor antes de realizar o vacuum, mas não é uma facilidade corrente. Então se você tem picos de carga extremos, o autovacuum não é para você.

Fábio Telles em - http://www.midstorm.org/~telles/2006/10/

5.6) Manutenção no arquivo de logs (registro)

Para saber a localização dos arquivos de logs confira o script postgresql.conf:

É uma boa prática, periodicamente, remover os arquivos mais antigos. Se será feito backup dos mesmos ou não, depende da importância dos mesmos.

O diretório pg_xlog contém os arquivos de logs das transações. Tamanhos fixos de 16MB.

O diretório pg_clog guarda o status das transações.

O diretório pg_subtrans contém o status das subtransações

No postgresql.conf temos:

#log_directory = 'pg_log'

1- Instalando o PostgreSQL no Windows
2- Instalando o PostgreSQL no Linux
3- Atualizando e migrando entre versões
4- Recuperando fisicamente os bancos

Instalações (itens 1 e 2), vide módulo 1 e aula 2 do módulo2.

5.7) – Atualizando e Migrando entre Versões

Existem dois tipos de atualização entre versões no PostgreSQL. Uma entre as versões menores e entre as maiores:

menores: com 3 dígitos (8.1.2, 8.1.4, etc)
maiores: com 2 dígitos (8.0, 8.1, 8.3, etc)

As menores sempre têm formato de armazenamento compatível e são chamadas compatíveis.

Exemplo: Tinhamos a versão 8.3.0. Foi lançada hoje a versão 8.3.1. Estas são compatíveis e permitem uma atualização física.

Vejamos o que diz o script upgrade.bat do PostgreSQL 8.3.1-1 no Windows:
"Você precisa ter o PostgreSQL 8.3.x instalado de um instalador MSI para usar este atualizador.

Nas "Release Notes":
http://www.postgresql.org/docs/8.3/interactive/release-8-3-1.html

"E.1.1. Migration to Version 8.3.1

A dump/restore is not required for those running 8.3.X. However, you might need to REINDEX indexes on textual columns after updating, if you are affected by the Windows locale issue described below."

Acostume-se a olhar pois eles sempre informam qual o procedimento a
ser adotado para aquela versão.

Osvaldo na lista pgbr-geral.

Atualização entre versões menores de uma mesma versão maior:

No Windows, para atualizar da 8.1.2 para 8.1.3 podemos usar o upgrade.bat. Também alerta que se o servidor ou qualquer cliente do PostgreSQL estiver rodando, será requerido um reboot.

Atualização entre versões maiores

Para atualizar entre versões 8.1 e 8.2 ou entre 8.2 e 8.3.
Por precaução pare o serviço antigo e mantenha o antigo em outro diretório.
Exportar todos os dados usando pg_dumpall
Reinstalar a nova versão
Importar os dados para a nova versão

Recuperando fisicamente o PostgreSQL

Em caso de falha do sistema e que não mais se tenha acesso ao servidor, podemos recuperar o sistema fisicamente, copiando todo o cluster.

- De um servidor com o mesmo SO, mesmo sistema de arquivos, mesma versão do PostgreSQL instalada, com os mesmos parâmetros de configuração.
- Substitua o cluster inteiro do novo servidor pelo cluster do seu backup. Verifique antes, é claro, se a estrutura de diretórios do seu backup está semelhante à do seu backup.

Obs.: Faça a cópia com o servidor parado!!!

Dica do Fábio Telles na lista pgbr-geral.

Migrações

É recomendada a utilização dos programas pg_dump e pg_dumpall da nova versão do PostgreSQL, para tirar vantagem das melhorias que podem ter sido introduzidas nestes programas.

O menor tempo de parada pode ser obtido instalando o novo servidor em um diretório diferente, e executando tanto o servidor novo quanto o antigo em paralelo, em portas diferentes. Depois pode ser executado algo como

```
pg_dumpall -p 5432 | psql -d template1 -p 6543

Exemplo através dos fontes (Linux):

pg_dumpall > backup
pg_ctl stop
mv /usr/local/pgsql /usr/local/pgsql.old
cd ~/postgresql-8.0.0
gmake install
initdb -D /usr/local/pgsql/data
postmaster -D /usr/local/pgsql/data
psql template1 < backup
```

Migrando entre Versões

Caso você tenha uma versão que não seja 8.1.x e esteja querendo instalar a 8.1.4, então precisa fazer um backup dos seus dados e restaurar logo após a instalação como sugerido em seguida. Será assumido que sua instalação foi em:
/usr/local/pgsql e seus dados no sub diretório data. Caso contrário faça os devidos ajustes.

1 – Atenção para que seus bancos não estejam recebendo atualização durante o backup. Se preciso proíba acesso no pg_hba.conf.
2 – Efetuando backup:
pg_dumpall > bancos.sql .Para preservar os OIDs use a opção -o no pg_dumpall.
3 – Pare o servidor
pg_ctl stop ou outro comando

Caso queira instalar a nova versão no mesmo diretório da anterior
mv /usr/local/pgsql /usr/local/pgsql.old
Então instale a nova versão, crie o diretório de dados e einicie o novo servidor.
/usr/local/pgsql/bin/initdb -D /usr/local/pgsql/data
/usr/local/pgsql/bin/postmaster -D /usr/local/pgsql/data

Finalmente, restore seus dados usando o novo servidor com:
/usr/local/pgsql/bin/psql -d postgres -f bancos.sql

Mais detalhes:
http://pgdocptbr.sourceforge.net/pg80/install-upgrading.html
http://pgdocptbr.sourceforge.net/pg80/migration.html
http://pgdocptbr.sourceforge.net/pg80/migration.html
http://www.postgresql.org/docs/8.3/interactive/install-upgrading.html
http://www.postgresql.org/docs/8.3/interactive/migration.html

Texto de discussão na lista pgbr-geral:

> Salve Pessoal,

Opa!

> Estou pensando em atualizar a versão do meu banco para a 8.3.
> Alguma recomendação especial? algum how to a indicar?
> Obrigado.

Kra, não há muitos mistérios. Siga:

1. Faça um backup (com o pg_dump)

2. pare o serviço da versão instalada e instale a nova (por enquanto
não desinstale a antiga...)
3. restaure o backup na versão nova.
4. faça testes e tunning e todas as configurações que são necessárias
5. se tudo deu certo até aqui, desinstale a anterior e corra pro abraço!

Prezado Marco,

Recomendações:

* Revise todas as consultas que montam textos, seja unindo com constantes,
seja concatenando campos texto:
*** Coloque um CAST explícito em todas elas. ***

* Revise todas as comparações entre campos e de campos com constantes.
Todas
as comparações devem ter rigorosamente o mesmo tipo de dado:
*** Coloque um CAST explícito em todas elas. ***

* Se tiver stored procedures, revise se os tipos de dados dos parâmetros
batem
rigorosamente com o que estiver declarado na stored procedure:
*** Coloque um CAST explícito em todas elas. ***

* Se em algum lugar você vir uma mensagem de erro ao comparar alguma coisa:
*** Coloque um CAST explícito em todas elas. ***

* Cuidado com tabelas temporárias, o Postgres 8.3 não deixa você
excluí-las. Caso tenha algum caso desses, você pode precisar usar tabelas
reais e controlar sua limpeza manualmente.

* Prepare-se para sofrer muito e estourar seu cronograma, pois essa migração
é cheia de surpresas. Teste cada consulta de cada script e cada programa com
todas as alternativas imagináveis. Quanto mais cuidadoso for nisso, menos
erros vão escapar.

* Quando seu chefe reclamar que está demorando para migrar, lembre-o de que a
diferença de desempenho da versão 7 para a 8 é enorme e que, ao contrário
da versão 7, a versão 8 não perde de lavada do SQL Server 2000 quando você
começa a escrever consultas mais complexas em tabelas que não sejam
pequenas. Não medi ainda para saber se empata ou ganha, mas pelo menos a
diferença não é mais tão absurda.

Para finalizar: Boa sorte!

Mozart Hasse

Apenas para acrescentar, como se trata de uma versão '7' para a '8' é importante que voce verifique as releases notes da 8.3, pois muitas coisas mudaram e isso pode gerar um impacto em determinados pontos de sua aplicação.

Conversões implicitas, tsearch2 entre outros, sofreram mudanças desde a versão 7.4...

Use sempre um ambiente de teste e realmente teste tudo o que puder, antes de colocar isso em produção.

Em muitas empresas existe a figura do Testador utilize-o para checar se sua aplicação não sofreu danos na lógica das regras de negócio.
--
[]s
Dickson S. Guedes

Tem mais uma:

Como as versões 8.X passaram a gravar usuários e grupos na mesma tabela. Elas não permitem mais a criação de um usuário com nome igual ao de um grupo.

Marco Aurélio