3) Entendendo o Sistema de Autenticação

3.1) Conhecendo os Métodos de autenticação do pg_hba.conf
3.2) Arquivo de senhas pgpass.conf
3.3) Entendendo e manipulando o arquivo pg_ident.conf
3.4) Configurando o ambiente do PostgreSQL

Informações Básicas sobre Redes

IPs, Redes, Subredes e Máscaras

Sendo um IP - 10.0.0.1/32
Sendo uma sub-rede - 10.0.0.0/24

10.0.0.7/32 ou 10.0.0.7/255.255.255.255

189.41.12.0/24 ou 189.41.12.0/255.255.255.0

* Classe A (8 bits de rede) : 255.0.0.0
* Classe B (16 bits de rede): 255.255.0.0
* Classe C (24 bits de rede): 255.255.255.0

Tabela sub-rede IPv4

Notação CIDR	Máscara	Nº IPs
/0	0.0.0.0	4.294.967.296
/8	255.0.0.0	16.777.216
/16	255.255.0.0	65.536
/24	255.255.255.0	256
/25	255.255.255.128	128
/26	255.255.255.192	64
/27	255.255.255.224	32
/28	255.255.255.240	16
/29	255.255.255.248	8
/30	255.255.255.252	4
/32	255.255.255.255	1

3.1) pg_hba.conf no PostgreSQL 8.3

Quando um aplicativo cliente se conecta ao servidor de banco de dados especifica o nome de usuário do PostgreSQL a ser usado na conexão, de forma semelhante à feita pelo usuário para acessar o sistema operacional Unix. Dentro do ambiente SQL, o nome de usuário do banco de dados determina os privilégios de acesso aos objetos do banco de dados. Portanto, é essencial controlar como os usuários de banco de dados podem se conectar.

A *autenticação* é o processo pelo qual o servidor de banco de dados estabelece a identidade do cliente e, por extensão, determina se o aplicativo cliente (ou o usuário executando o aplicativo cliente) tem permissão para se conectar com o nome de usuário que foi informado.

Observar que não se aborda auenticação para usuários locais, que no caso teria uma linha iniciando com 'local'.

A versão atual do PostgreSQL usa o arquivo:

C:\Documents and Settings\ribafs\Application Data\postgresql\pgpass.conf

Explicando os campos

TYPE pode ser conexão do tipo local ou host
DATABASE pode ser "all", "sameuser", "samerole", um nome de banco, ou

uma lista separada por vírgula

USER pode ser "all", um nome de usuário, um nome de grupo prefixado com "+", ou

uma lista separada por vírgula. Nos campos USER e DATABASE podemos escrever um nome de arquivo prefixado com "@" para incluir um arquivo.

CIDR-ADDRESS especificica os hosts, que podem ser um IP e a máscara CIDR, que é um inteiro entre 0 e 32 para IPV4 ou 128 para IPV6, que especifica o número de significância da máscara. Alternativamente podemos inserir um IP e a máscara em coluna separada para especificar um conjunto de hosts.

METHOD pode ser "trust", "reject", "md5", "crypt", "password", "gss", "sspi", "krb5", "ident", "pam" ou "ldap". Note que "password" envia senhas em texto claro; "md5" deve ser preferida desde que envia senhas criptografadas.

OPTION é o mapa ident ou o nome do serviço PAM, dependendo do METHOD.

Obs.: Nomes de bancos e de usuários contendo espaços, vírgulas, aspas e outros caracteres especiais devem vir entre aspas duplas.

Após alterar este arquivo pode usar o "pg_ctl -D data reload" para atualizar as alterações.

Trecho principal do arquivo pg_hba.conf padrão do PostgreSQL 8.3 no Windows

```
# TYPE  DATABASE    USER        CIDR-ADDRESS          METHOD    [OPTION]

# IPv4 local connections:

host        all             all          127.0.0.1/32          md5
# IPv6 local connections:
#host       all             all          ::1/128               md5
```

Trecho principal do arquivo pg_hba.conf padrão do PostgreSQL 8.3 no Linux/Ubuntu 7.10

```
# TYPE  DATABASE    USER        CIDR-ADDRESS          METHOD    [OPTION]
```

```
# "local" is for Unix domain socket connections only
local      all             all                                     ident           sameuser
# IPv4 local connections:
host       all             all          127.0.0.1/32               md5
# IPv6 local connections:
host       all             all           ::1/128                   md5
```

Alguns exemplos

```
# Permitir qualquer usuário do sistema local se conectar a qualquer banco
# de dados sob qualquer nome de usuário utilizando os soquetes do domínio
# Unix (o padrão para conexões locais).
#
# TYPE  DATABASE    USER       CIDR-ADDRESS         METHOD
local   all         all                             trust

# A mesma coisa utilizando conexões locais TCP/IP retornantes (loopback).
#
# TYPE  DATABASE    USER       CIDR-ADDRESS         METHOD
host    all         all        127.0.0.1/32         trust

# O mesmo que o exemplo anterior mas utilizando uma coluna em separado para
# máscara de rede.
#
# TYPE  DATABASE    USER       IP-ADDRESS    IP-MASK           METHOD
host    all         all        127.0.0.1     255.255.255.255   trust

# Permitir qualquer usuário de qualquer hospedeiro com endereço de IP 192.168.93.x
# se conectar ao banco de dados "template1" com o mesmo nome de usuário que "ident"
# informa para a conexão (normalmente o nome de usuário do Unix).
#
# TYPE  DATABASE    USER       CIDR-ADDRESS         METHOD
host    template1   all        192.168.93.0/24      ident sameuser

# Permitir o usuário do hospedeiro 192.168.12.10 se conectar ao banco de dados
# "template1" se a senha do usuário for fornecida corretamente.
#
# TYPE  DATABASE    USER       CIDR-ADDRESS         METHOD
host    template1   all        192.168.12.10/32     md5

# Na ausência das linhas "host" precedentes, estas duas linhas rejeitam todas
# as conexões oriundas de 192.168.54.1 (uma vez que esta entrada será
# correspondida primeiro), mas permite conexões Kerberos V de qualquer ponto
# da Internet. A máscara zero significa que não é considerado nenhum bit do
# endereço de IP do hospedeiro e, portanto, corresponde a qualquer hospedeiro.
#
```

```
# TYPE  DATABASE    USER        CIDR-ADDRESS          METHOD
host    all         all         192.168.54.1/32       reject
host    all         all         0.0.0.0/0             krb5

# Permite os usuários dos hospedeiros 192.168.x.x se conectarem a qualquer
# banco de dados se passarem na verificação de "ident". Se, por exemplo, "ident"
# informar que o usuário é "oliveira" e este requerer se conectar como o usuário
# do PostgreSQL "guest1", a conexão será permitida se houver uma entrada
# em pg_ident.conf para o mapa "omicron" informando que "oliveira" pode se
# conectar como "guest1".
#
# TYPE  DATABASE    USER        CIDR-ADDRESS          METHOD
host    all         all         192.168.0.0/16        ident omicron

# Se as linhas abaixo forem as únicas três linhas para conexão local, vão
# permitir os usuários locais se conectarem somente aos seus próprios bancos de
# dados (bancos de dados com o mesmo nome que seus nomes de usuário), exceto
# para os administradores e membros do grupo "suporte" que podem se conectar a
# todos os bancos de dados. O arquivo $PGDATA/admins contém a lista de nomes de
# usuários. A senha é requerida em todos os casos.
#
# TYPE  DATABASE    USER        CIDR-ADDRESS          METHOD

local   sameuser    all                               md5
local   all         @admins                           md5
local   all         +suporte                          md5

# As duas últimas linhas acima podem ser combinadas em uma única linha:
local   all         @admins,+suporte                  md5

# A coluna banco de dados também pode utilizar listas e nomes de arquivos,
# mas não grupos:
local   db1,db2,@demodbs  all                         md5
```

Detalhes em: http://pgdocptbr.sourceforge.net/pg80/client-authentication.html

Criando uma lista de usuários para controlar o acesso ao banco de dados..

É possível definir no pg_hba.conf para ir buscar os usuarios que tem acesso ao PostgreSQL em algum arquivo.

Imagine tendo que criar centenas de usuários, então perceberá a utilizade deste recurso.

No caso segue os seguintes passos:

- Sugestão: Criar um arquivo "users" (sem extensão).
- Inserir alguns usuários de teste (Ex: alice)
- Salvar o arquivo no mesmo diretório do pg_hba.conf, no caso do ubuntu em

/etc/postgresql/8.3/main ,

não esqueçendo de fornecer permissões do arquivo para o usuario postgres

- Comentar as configurações originais (caso ocorra algum imprevisto, sempre é bom ter as configurações iniciais).
- E informar as configurações escolhidas. No caso foi:

```
# "local" is for Unix domain socket connections only
#local         all     all             ident sameuser
local           all     @users          md5
```

- Salvar o pg_hba.conf.
- Reiniciar o serviço do PostgreSQL

Colaboração do Paulo Alceu (aluno da primeira turma do curso DBA PostgreSQL)

3.2) Arquivo de senhas pgpass.conf

O arquivo `.pgpass`, armazenado na pasta base (home) do usuário, é um arquivo que contém senhas a serem utilizadas se a conexão requisitar uma senha (e a senha não tiver sido especificada de outra maneira). No Microsoft Windows o arquivo se chama `%APPDATA %\postgresql\pgpass.conf` (onde `%APPDATA%` se refere ao subdiretório de dados do aplicativo no perfil do usuário).

Este arquivo deve conter linhas com o seguinte formato:

```
host:porta:banco:usuario:senha
```

Caso pretenda que o usuário 'copia' tenha acesso sem senha ao host '10.0.0.5' na porta 5432 e a todos os bancos, adicionar a linha abaixo ao arquivo pgpass.conf (windows) ou .pgpass (linux):

10.0.0.5:5432:*:copia:

Os quatro primeiros valores podem ser um literal, ou * para corresponder a qualquer coisa. É utilizada a senha da primeira linha que corresponder aos parâmetros da conexão corrente (portanto, as entradas mais específicas devem ser colocadas primeiro quando são utilizados curingas). Se a entrada precisar conter os caracteres : ou \, estes caracteres devem receber o escape de \.

O arquivo `.pgpass` não deve permitir o acesso por todos os usuários ou grupos; isto é conseguido pelo comando chmod 0600 ~/.pgpass. Se as permissões forem mais rígidas que esta, o arquivo será ignorado (Entretanto, atualmente as permissões não são verificadas no Microsoft Windows).

Detalhes em: http://pgdocptbr.sourceforge.net/pg80/libpq-pgpass.html e http://www.postgresql.org/docs/8.3/interactive/libpq-pgpass.html

3.3) Entendendo e manipulando o arquivo pg_ident.conf

O método de autenticação ident funciona obtendo o nome de usuário do sistema operacional do cliente, e determinando os nomes de usuário do PostgreSQL permitidos utilizando um arquivo de mapa que lista os pares de nomes de usuários correspondentes permitidos.

sameuser é o usuário padrão no ident.conf (significa o mesmo user do sistema operacional).

Este script de autenticação é utilizado nos UNIX, pois no Windows o usuário não é usuário com direito a login.
Exemplo de entrada no arquivo pg_hba.conf:

```
host    all     all     10.0.0.7/32     ident           mapateste
```

Todos os usuários da máquina 10.0.0.7 terão acesso a todos os bancos do PostgreSQL. Os usuários desta máquina serão mapeados pelo mapa ident mapateste. No caso teremos os seguintes registros no pg_ident.conf:

```
mapateste       joao    joaozinho
mapateste       mich    michael
mapateste       dani    daniel
```

Quando o usuário joao (em 10.0.0.7) se conecta remotamente ao servidor do PostgreSQL ele será mapeado para o usuário joaozinho, que é uma conta do sistema no servidor. Os demais de forma similar.
Caso precise usar este arquivo e ele não exista, crie no diretório sugerido acima, com as entradas desejadas.

Mais detalhes em:
http://www.postgresql.org.br/Documenta%C3%A7%C3%A3o?action=AttachFile&do=get&target=postgresql.conf.pdf
http://pgdocptbr.sourceforge.net/pg80/auth-methods.html#AUTH-IDENT
http://www.postgresql.org/docs/8.3/interactive/client-authentication.html
Livro PostgreSQL 8 for Windows de Richard Blum, capítulo 3.

Alerta Ao consultar a documentação fique atento para a versão do PostgreSQL que estás utilizando. Fortemente recomendada a consulta ao script postgresql.conf e sua documentação (comentários).

Exercícios

1) Avaliando o desempenho de índices

Cenário:
Banco de dados de CEPs com 633.401 registros cep_brasil.
tabela cep_full sem índice
tabela cep_full_index com índice

Criando o banco com codificação win1252:

```
postgres=# create database cep_brasil encoding 'win1252';

postgres=# \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)
);
```

Importanto do CSV (http://tudoemum.ribafs.net/includes/cep_brasil.sql.bz2):
```
\copy cep_full from E:\Enviar\cep_brasil_unique.csv
```

Essa é interessante para quem não conhece. Crio uma segunda tabela
tendo como base uma consulta sobre outra tabela, já importando todos
os registros:
```
create table cep_full_index as select * from cep_full;
```

Alterar a tabela cep_full_index adicionando índice:
```
alter table cep_full_index add constraint cep_pk primary key (cep);
```

Atualizando as estatísticas:
```
vacuum analyze;
```

Pegando o plano de consulta da primeira consulta:
```
explain select logradouro from cep_full where cep='60420440';
                        QUERY PLAN
-------------------------------------------------------------
 Seq Scan on cep_full  (cost=0.00..33254.51 rows=1 width=71)
  Filter: (cep = '60420440'::bpchar)
(2 rows)
```

O plano da segunda:
```
explain select logradouro from cep_full_index where cep='60420440';
                              QUERY PLAN
-----------------------------------------------------------------------------
 Index Scan using cep_pk on cep_full_index  (cost=0.00..8.37 rows=1 width=71)
  Index Cond: (cep = '60420440'::bpchar)
(2 rows)
```

Isso foi interessante, pois numa tabela grande gasta um tempo muito pequeno. Como eu estava fazendo, usando o \timing, a consulta demorava muito.

[1] – Adaptação de dica recebida do Roberto Mello (na lista pgbr-geral).

2) Acesso Remoto

- Visualizar os scripts originais: postgresql.conf, pg_hba.conf e o pg_ident.conf do micro servidor (do professor) do PostgreSQL
- Tentar conectar ao PostgreSQL numa máquina remota (outro micro da sala), usando o PGAdmin e o psql
- Observar as mensagens de erro
- Criar um novo usuário no micro cliente
- Liberar no postgresql.conf, no pg_hba.conf apenas para o usuário 'postgres', para o banco 'dbapg', para o IP 192.168.x.y
- Tentar novamente
- Liberar para toda a rede interna

Aproveitar para fazer um tour pelo PGAdmin.
Observe que as tablespaces e as roles ficam fora dos bancos.

3) Instalar o VirtualBox e nele o pg_live.
Observe que como no Windows, o pg_live traz a última versão do PGAdmin (1.82), como também traz o PostgreSQL 8.3 e o PHP rodando, além de outras ferramentas úteis.

Novo site de apoio ao curso

Para maior privacidade coloquei apenas para usuários registrados.

- Acesse http://ribafs.net
- Fazer seu registro, caso ainda não seja registrado e confirme o e-mail que receber.
- Faça login e acesse a seção Artigos – DBA PostgreSQL

Aqui estarei adicionando notícias, dicas e outros assuntos ligados direta ou indiretamente ao PostgreSQL.

3.4) Configurando o ambiente do PostgreSQL

Através do postgresql.conf e de comandos SQL 'SET'

Através do postgresql.conf

Lembrar que as linhas iniciadas com # estão comentadas e que todos os parâmetros comentados são o padrão. Caso queira alterar algum deixe comentado e copie a linha logo abaixo descomentada e com o valor desejado.

Alguns parâmetros importantes

#data_directory = 'ConfigDir'

Caso queira alterar o data_diretory para outro diretório:
- copie o diretório atual ou mova-o para o novo diretório
- altere este parâmetro no postgresql.conf
- recarregue o serviço

#hba_file = 'ConfigDir/pg_hba.conf'

#listen_addresses = 'localhost'
Por default o PostgreSQL somente dá acesso para usuários locais, pela console.
Para dar permissão para acesso pelo PGAdmin, mesmo local e a outros hosts não locais devemos entrar com seu IP ou nome neste parâmetro.
Entrar com '*' para dar acesso a qualquer host.
Para uma lista de IPs separar por vírgula, assim: '10.0.0.7,10.0.0.8,10.0.0.9'.
Obs.: Mudanças requerem restartar serviço.

#port = 5432

#ssl = off

Uso de conexões seguras através do openssl. Mudanças requerem restartar serviço.
Para aumentar a segurança criptografando as comunicações cliente/servidor, o PostgreSQL possui suporte nativo para conexões SSL. É necessário que o OpenSSL esteja instalado tanto no cliente quanto no servidor, e que o suporte no PostgreSQL tenha sido ativado em tempo de instalação.
Detalhes em: http://pgdocptbr.sourceforge.net/pg80/ssl-tcp.html e
http://www.postgresql.org/docs/8.3/interactive/ssl-tcp.html

#password_encryption = on
Quando é especificada uma senha em CREATE USER ou ALTER USER, sem que seja escrito ENCRYPTED ou UNENCRYPTED, este parâmetro determina se a senha deve ser criptografada. Altamente recomendado o uso de senhas criptografadas.

max_connections = 100
Másimo de conexões simultâneas que serão atendidas pelo servidor. Manter o mais baixo possível, tendo em vista que cada conexão ativa requer recursos do hardware.
Mudanças requerem restartar serviço

autovacuum = on
habilita o subprocesso autovacuum
Para ativar, obrigatoriamente ative também ***track_counts (na versão 8.3)***

#autovacuum_naptime = 1min
Tempo entre as execuções do vacuum

#datestyle = 'iso, mdy'
Formato/estilo da data
Para ter a data no formato do Brasil (dd/mm/yyyy) altere para: datestyle = 'sql, dmy'

#client_encoding = latin1
Na verdade, os padrões escolhidos pelo utilitário initdb são escritos no arquivo de configuração `postgresql.conf` apenas para servirem como padrão quando o servidor é inicializado. Se forem removidas as atribuições presentes no arquivo `postgresql.conf`, o servidor herdará as definições do ambiente de execução.

#lc_messages = 'C'
lc_messages = 'pt_BR' # Mensagens de erro em português do Brasil

Obs.: Alguns parâmetros do postgresql.conf podem ser alterados pelo comando "SET" do SQL e também pelo comando "ALTER".

Configurações Através do comando 'SET' do SQL
Através do comando SET podemos alterar as configurações do PostgreSQL.
O comando SET muda os parâmetros de configuração de tempo de execução. Muitos parâmetros de tempo de execução podem ser mudados dinamicamente pelo comando SET (Mas alguns requerem privilégio de superusuário para serem mudados, e outros não podem ser mudados após o servidor

ou a sessão iniciar). O comando SET afeta apenas os valores utilizados na sessão atual.

Para exibir o formato de data atual do PostgreSQL:
SHOW DATESTYLE;

Execute a consulta:
SELECT CURRENT_DATE;

Para alterar o estilo da data na seção atual para o formato dd-mm-yyyy:
SET DATESTYLE TO 'postgresql,dmy';

Execute novamente a consulta:
SELECT CURRENT_DATE;

Para alterar o estilo da data na seção atual para o formato dd/mm/yyyy:
SET DATESTYLE TO 'sql,dmy';

Execute novamente a consulta:
SELECT CURRENT_DATE;

SET DATESTYLE TO ISO;
SELECT CURRENT_TIMESTAMP;

SET DATESTYLE TO PostgreSQL;
SELECT CURRENT_DATE;

SET DATESTYLE TO US;
SELECT CURRENT_DATE;

SET DATESTYLE TO NONEUROPEAN, GERMAN;
SELECT CURRENT_DATE;

SET DATESTYLE TO EUROPEAN;
SELECT CURRENT_DATE;

Para exibir todas as configurações:
SHOW ALL;

Também podemos Configurar um banco para Aceitar datas no formato praticado no Brasil:
CREATE DATABASE bancobrasil;
ALTER DATABASE SET DATESTYLE TO 'SQL,DMY';

Também pode ser atribuído juntamente com o Usuário:
ALTER ROLE nomeuser SET DATESTYLE TO SQL, DMY;

Alerta: para bancos que já estejam em uso, geralmente fica inviável uma alteração permanente

(postgresql.conf), caso se esteja tratando as datas. Exemplo: ao consultar um campo data, a aplicação espera um formato e vai estar outro. A não ser que se altere consultas SQL existentes nos aplicativos.

ALTER DATABASE cep_brasil SET client_encoding=win1252;

ISO-8859-15 é o Latin9

SHOW ENABLE_SEQSCAN;
SET ENABLE_SEQSCAN TO OFF;

Mais detalhes em:
http://pgdocptbr.sourceforge.net/pg80/sql-set.html e
http://www.postgresql.org/docs/8.3/interactive/sql-set.html