Grupos, Usuários e Privilégios no PostgreSQL

(Usando PostgreSQL 10.6 no Linux Mint 19.1)

Para que um determinado usuário tenha acesso a certo banco de dados, de um certo IP usando um certo método de autenticação precisamos efetuar várias configurações em:

- postgresql.conf
- pg_hba.conf (banco, user, ip/rede, método)
- Grant e revoke

Por conta disso este é um recursos trabalhosos do PostgreSQL e que gera muitas dúvidas. Isso me motivou a criar este guia bem detalhado.

Criarei 6 grupos cada um com um perfil diferente em termos de privilégios

super - com todos os privilégios em todos os bancso do SGBD
admin - com todos os privilégios apenas sobre o banco_um, mas sem privilégio de criar o banco, que será criado por um usuário do grupo super
devel - com todos os privilégios apenas sobre o esquema_um do banco banco_um, mas somente localmente
devel_remote - com todos os privilégios apenas sobre o esquema_um do banco banco_um remotamente (de outros IPs)
manager - com todos os privilégios mas apenas na tabela tabela_um do esquema_um do banco_um
user - apenas com privilégio de consultar/select a tabela_um do esquema_um do banco_um

Estes nomes não são bons, pois não representam uma realidade. É bom quando usamos nomes que representam a realidade de uma enpresa ou organização.

Grupos: grp_super, grp_admin, grp_engenharia, grp_comtabilidade, etc
Usuários: user_super1, user_admin1, user_engenharia1, user_contabilidade1, etc
ou
Usuários: user_joao.brigido, user_pedro.carlos, etc.
No caso, sempre precisarmos anotar a relação de usuários e suas funções/privilégios.

Observações importantes:

- Esquema public - todo banco criado no PostgreSQL tem um esquema chamado public, que dá acesso a qualquer usuário e inclusive permite que eles criem objetos no mesmo. Idealmente deve ser removido, para um melhor controle.

pg_hba.conf original no PostgreSQL 10.6 no Linux Mint 19.1:
l
/etc/postgresql/10/main/pg_hba.conf

Criando um backup do original:
sudo cp /etc/postgresql/10/main/pg_hba.conf /etc/postgresql/10/main/pg_hba.conf.BAK

```
local   all             postgres                                peer

# TYPE  DATABASE        USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
local   all             all                                     peer
# IPv4 local connections:
host    all             all             127.0.0.1/32            md5
# IPv6 local connections:
host    all             all             ::1/128                 md5
# Allow replication connections from localhost, by a user with the
# replication privilege.
local   replication     all                                     peer
host    replication     all             127.0.0.1/32            md5
host    replication     all             ::1/128                 md5
```

Acesso Local com o postgres

Após instalara o PostgreSQL não temos acesso aos bancos usando o usuário default, que é o postgresl. Para isso precisamos conceder uma senha ao mesmo e alterar o método de conexão no pg_hba.conf. Veja:

Veja na primeira linha que é dado ao usuário postgres acesso usando o método peer. Este método concede acesso para um usuário do sistema operacional com o mesmo nome (postgres). Então alteramos o método para **md5** e concedemos uma senha ao mesmo assim:

```
sudo su
su – postgres
psql
alter role postgres password 'postgres';
\q

exit
service postgresql restart
```

Agora podemos acessar localmente os bancos do PostgreSQL com postgres, postgres.

Resetando a senha do super user postgres

Caso perca o acesso ao SGBD com o postgres, tendo errado a senha ou por outro motivo, então edite o pg_hba.conf, mude a linha com postgres de md5 para trust. Então reinicie o serviço do PostgreSQL. Acesse novamente e mude a senha:

sudo su
su – postgres
psql
alter role postgres password 'postgres';
\q

exit
service postgresql restart

Acesso remoto
Para conceder acesso remoto e mesmo local do banco para certos usuários usando certo IP é feito no pg_hba.conf, especificamente na linha:

host all all 127.0.0.1/32 md5

Onde devemos indicar o banco, o usuário, o IP e o método de acesso.

- O parâmetro INHERIT é o default e não precisa ser indicado

- Quando criamos um usuárua com "create role", por default o PostgreSQL não concede direito de logar. Se quizermos este direito usemos o alter role.

- Todos os privilégios devem ser concedidos ao grupo e não diretamente ao usuário, assim como também para remover devemos remover do grupo. Os usuários membros do grupos herdarão e assim é realmente mais coerente.

- Alguns poucos privilégios não são herdados, como é o caso de LOGIN e SUPERUSER.

Criar um grupo de super usuários chamado "super"

sudo su
su - postgres
psql
CREATE ROLE super WITH LOGIN SUPERUSER PASSWORD 'super';

Listar usuários
\du
ou
select usename from pg_user;
Sair do psql
\q
Tentar logar como super

psql -U super

Erro: psql: FATAL: Peer authentication f ailed for user "super"
Precisamos configurar o pg_hba.conf para que usuários locais usem md5 e não peer, nesta linha

local   all   all   peer

Mudar para
local   all   all   md5

e reiniciar o serviço do PostgreSQL.

Tentar novamente
su - postgres
psql -U super
Password for user super:
psql: FATAL: database "super" does not exist

Reclama que o banco super não existe. Ele tenta conectar com um banco igual ao nome do usuário.

Conectar ao banco postgres
psql -U super -d postgres

Agora conectou

Criar usuário super1 pertencente ao grupo super usando o próprio usuário super
\q

psql -U super -d postgres
create role super1 in role super password 'super1';

\dg
Veja que super1 mesmo pertencendo ao grupo/role super, não pode fazer login

Corrija isso com:
alter role super1 login; -- O WITH é opcional
\dg
Agora ele pode fazer login. Então testemos:
\q

psql -U super1 -d postgres
\du

Consegue logar, mas veja que ele não é super usuário (o prompt é este =>), mesmo tendo sido criado no grupo do super (Member of) não herdou este privilégio.

Então usemos o super para transformá-lo em super usuário
\q
psql -U super -d postgres

alter role super1 superuser;
\dg

Agora ele é super user.

\q
psql -U super1 -d postgres
\du
Agora sim, ele é super usuário, com todos os poderes (prompt =#, ao invés de =>).

Agora vou conectar através do adminer.php via web usando o super1, que é um cliente para vários SGBDs, inclusive PostgreSQL:

http://localhost/adminer.php

System - PostgreSQL
Server - localhost
Username - super1
Password - super1

Conecta com sucesso. Inclusive criei e removi um banco teste.

Veja no pg_hba que não existe nenhum comando liberando acesso remoto, nenhum IP, exceto o 127.0.0.1, portanto o super1 não deve acessar remotamente de outro IP, mesmo que o outro computador esteja na mesma rede deste. Vamos testar. Estou em uma rede WIFI e vou testar em outra máquina da mesma.

Tentei acessar esta máquina (192.168.25.11 da máquina 192.168.25.10). Não consegue conectar, conexão recusada. Pergunta:

"O server está rodando no host 192.168.25.11 e aceita conexão tipo TCP/IP na porta 5432?"

Realmente não, só aceita cnexão socket unix (via psql no terminal) ou via localhost/127.0.0.1.

Alerta: é uma iniciativa arriscada em termos de segurança liberar o acesso remoto para um suer usuário. Para acesso remoto liberemos apenas usuários com privilégios restritos, apenas os privilégios necessários e simples como consultas ao banco. Evitemos algo como criar objetos e usuários, apenas manipular registros e ainda assim de forma otimizada, somente o que o usuário realmente precisa. Se podemos fazer assim, por que não fazer?

Obs.: por conta da segurança devemos evitar gerenciar o SGBD com o super usuário ou com um super usuário. É mais seguro para isso criar um usuário com algumas restrições como:
CREATE ROLE administrador WITH LOGIN CREATEROLE CREATEDB PASSWORD 'administrador';

Criar o segundo grupo, o admin
psql -U super1 -d postgres
create role admin with INHERIT LOGIN password 'admin';

\q
psql -U admin -d postgres

Consegue logar pelo psql

Veja pelo prompt (=>) que é usuário comum. Não é super usuário.

Tentemos conectar pelo adminer.

Consegue também. Mas tentei remover o banco testes e não consegui. Mas consegui criar uma tabela no esquema public.

Esquema public

Vale lembrar que qualquer usuário pode criar objetos no esquema public, por padrão.
Criei uma tabela num banco teste que havia no servidor.
Isso indica que não é interessante manter o esquema public nos bancos que são usados por equipes especialmente, mas não somente.
Eu gosto de remover o esquema public para evitar problemas e confusões.
Logado como super1 remover o esquema public do banco teste existente:

psql -U super1 -d postgres
\c testes
drop schema public cascade;

Uma alternativa é, ao invés de remover o esquema public, remover todos os privilégios para o público do esquema public:

revoke all privileges on database banco_um from public;

Tentei criar a tabela novamente com o admin mas agora não consegui.

Testes do super1, se realmente é super usuário: criação de banco e de usuário.

Criar um usuário apenas para teste e um banco também.

create role teste1;
create database teste1;
Criou os dois.

Agora remover:

drop role teste1;
drop database teste1;
Removeu ambos, o que caracteriza um super usuário.

Criar o usuário admin1 pertencendo ao grupo admin usando o usuário super1.
\q
psql -U super1 -d postgres

create role admin1 in role admin login password 'admin1';
\du

Remover usuário de grupo
revoke grupo from usuario;

Criar banco_um e tornar o grupo admin seu dono

Este usuário precisa ter todos os poderes sobre o banco_um. Vou criar o banco_um com o usuário super1 e dar todos os privilégios ao admin:
create database banco_um owner admin;
\q
psql -U admin1 -d banco_um

Conectou e pode criar uma tabela no esquema public.

create schema esquema1;
create table esquema1.tabela1(id int);
\d esquema1.tabela1

Também conecta pelo adminer e pode criar um esquema e removê-lo. Pode até apagar o banco_um, mas não apaga o testes.
Tentei criar novamente o banco_um mas ele não tem permissão.
Tentei criar uma role mas ele não tem permissão. Tá coerente.

Criar o grupo devel -- com todos os privilégios apenas sobre o esquema_um do banco banco_um, mas somente localmente
\q
psql -U super1 -d postgres
create role devel with login password 'devel';
\du
\c banco_um -- conectar ao banco um
create schema IF NOT EXISTS esquema_um AUTHORIZATION devel;

ou
alter schema esquema_um OWNER TO devel;

Listar os esquemas criados
\dn

Para excluir recursivamente:
drop schema cascade nome_esquema;

Nosso esquema está pronto para ser usado pelo usuário devel1.
Caso criemos a tabela sem usar o prefixo (esquema_um.) ela seria criada no esquema default, que é o public. Como removemos não será criada. Me parece um bom motivo para remover o esquema public.

Removendo o esquema public e tentando criar a tabela dará erro, dizendo que nenhum esquema foi selecionado, o que é bem melhor do que criar e não ser onde desejamos.
Uma alternativa é remover privilégios de remoção para o público (PUBLIC) no esquema public:

REVOKE CREATE ON SCHEMA public FROM PUBLIC;

Vou remover o esquema public do banco_um:

drop schema public;

Criar o usuário devel1 no grupo devel para gerenciar o esquema_um do banco_um.

create role devel1 in role devel login password 'devel1';
\du
\q
psql -U devel1 -d banco_um

Tentemos criar uma tabela com apenas um campo:

create table tabela1(id int);

Não consegue pois o esquema public foi removido e não especificamos o esquema_um nem o configuramos como o atual.

Tentemos assim:
create table esquema_um.tabela1(id int);

Logicamente ele cria.

select * from tabela1;
Reclama que tabela1 não existe.

select * from esquema_um.tabela1;
\d

Agora assim:

SET search_path TO esquema_um;
create table tabela2(id int);
select * from tabela2;

Também cria e consulta, como era de se esperar.

Agora um teste de conexão pelo localhost via web com o adminer.
Conectou sem problema. Cria tabela no esquema_um do banco_um e no esquema public de outros bancos mas não em outros esquemas.

Criar o grupo devel_remote - com todos os privilégios apenas sobre o esquema_umremoto do banco banco_um remotamente (de outros IPs)
\q
psql -U super1 -d postgres
create role devel_remote with login password 'devel_remote';
\du

\c banco_um
create schema esquema_umremoto authorization devel_remote;
ou

alter schema esquema_umremoto owner to devel_remote;

Agora precisamos permitir que devel_remote acesse o esquema_umremoto do banco_um remotamente, ou seja de outro computador/IP.

Mas antes vou criar um usuário no grupo devel_remote e dar a este usuário o poder de se conectar remotamente:

Criar o usuário devel_remote1 no grupo devel_remote

create role devel_remote1 in role devel_remote login password 'devel_remote1';
\du
\q
psql -U devel_remote1 -d banco_um
create table esquema_umremoto.tabela1(id int);

Criou

\d esquema_umremoto.tabela1

Testando conexão no adminer.
Conectou. Não pode mexer em outros esquemas mas pode criar tabela no esquema_umremoto, mas não conseguiu criar no esquema_um.

Habilitar o Acesso Remoto

Para permitir o acesso remoto configuramos o postgresql.conf e o pg_hba.conf:

No postgresql.conf mudar apenas listen_address:
\q
nano /etc/postgresql/10/main/postgresql.conf

Onde tem:

#listen_addresses = 'localhost' # what IP address(es) to listen on
Mudar para
listen_addresses = '*' # what IP address(es) to listen on

Por default ele somente permite acesso no localhost. Veja que a porta usada também está neste arquivo assim como muitas outras configurações.

nano /etc/postgresql/10/main/pg_hba.conf
exit
/etc/init.d/postgresql restart
su - postgres

Observe que a linha atual é:
host all all 127.0.0.1/32 md
Permite todos os usuários acessarem todos os bancos mas apenas localmente.

Vamos adicionar esta linha, logo abaixo da linha acima:

host banco_um devel_remote1 192.168.25.47/32 md5

Ficarão assim as duas linhas no pg_hba.conf:
host all all 127.0.0.1/32 md
host banco_um devel_remote1 192.168.25.47/32 md5

Quero que o usuário devel_remote1 possa acessar o banco_um do IP 192.168.25.47 usando autenticação md5.

Ao tentar conectar pelo computador com IP 192.168.25.47 recebo a mensagem:

"Connection refused. Is the server running os host 192.168.25.47 and accepting tcp/ip on port 5432."

Eu estava reiniciando ou recarregando os scripts com o comando
sudo service postgresql reload ou restart

Parece que não estava realizando o que deveria, então restartei o postgresql assim:
/etc/init.d/postgresql restart

Agora ele reclamou que o usuário devel_remote1 no host 192.168.25.47 não tem acesso ao banco postgres no pg_bha.conf.

Então ao criar a conexão com o PgAdmin na máquina remota usei o banco_um ao invés do postgres.

Conectou normalmente.

Liberando acesso remoto para toda uma rede no pg_hba.conf
192.168.25.0/24
\q

Criação do grupo manager - com todos os privilégios mas apenas na tabela tabela_um do esquema_um do banco_um

\q
psql -U super1 -d postgres
create role manager login password 'manager';
\du
\c banco_um
create table esquema_um.clientes(id serial primary key, nome char(50) not null, email char(50), endereco char(100));
\d esquema_um.clientes

Criar o usuário manager1 no grupo manager, que terá direitos somente de mexer na tabela clientes
create role manager1 in role manager login password 'manager1';
\du

Dar permissão total na tabela esquema_um.clientes para o usuário manager1.

\c banco um

Antes precisamos dar permissão de acesso ao ao esquema_um

GRANT USAGE ON SCHEMA esquema_um TO manager1;
GRANT ALL ON esquema_um.clientes TO manager1;

Testando:

\q
psql -U manager1 -d banco_um
\d esquema_um.clientes
Lista a estrutura da tabela

Vamos tentar inserir um registro na tabela clientes:

insert into esquema_um.clientes values (1, 'Ribamar FS', 'ribafs@gmail.com', 'Rua Vasco');

Inseriu sem problemas.

select * from esquema_um.clientes;

Algo importante é remover os privilégios default de todos os bancos para que usuários não autenticados não possam acessar.

Uma das opções do postgresql para isso é usar o comando REVOKE e outra é a exclusão do esquema public. Devemos usar ambas, de acordo com o caso.

Criar o grupo usuario - apenas com privilégio de consultar/select a tabela_um do esquema_um do banco_um

Não usei "user", pois é uma palavra reservada do postgresql, que não aceita em nomes de roles.

Assim ele somente acessará o esquema public dos bancos. Precisamos que acesse a tabela_um, do esquema_um, do banco_um.

Vamos criar a tabela_um, no esquema_um, no banco_um usando o super1:

\q
psql -U super1 -d banco_um
create role usuario with login password 'usuario';
\du
\c banco_um
create table esquema_um.tabela_um (codigo int primary key, nome char(50) not null, endereco char(100));

Conceder permissão de acesso ao esquema_um:

GRANT USAGE ON SCHEMA esquema_um TO usuario;

Agora dar privilégio somente de select nesta tabela. Nada de insert nem outros.

GRANT SELECT ON esquema_um.tabela_um TO usuario;
\q
psql -U usuario -d banco_um
select * from esquema_um.tabela_um;

Funcionou.

Agora tentarei inserir um registro:

insert into esquema_um.tabela_um values (1, 'Ribamar', 'Rua Vasoco');

Permissão negada. Sem a permissão na tabela_um, pois somente o privilégio SELECT foi concedido.

Criar o usuário usuario1 no grupo usuario

\q
psql -U super1 -d banco_um

create role usuario1 in role usuario login password 'usuario1';
\du

revoke all on schema esquema_um from usuario1;

Testando

\q
psql -U usuario1 -d banco_um
select * from esquema_um.tabela_um;

Consultou. Então não basta remover dele a permissão.

Agora para testar, vou remover o privilégio do grupo usuario de SELECT no tabela_um para ver.

\q
psql -U super1 -d banco_um
REVOKE ALL PRIVILEGES ON esquema_um.tabela_um FROM usuario;

\z esquema_um
\z esquema_um.tabela_um
Veja que o acesso é somente para super1.

Testemos:

\q
psql -U usuario1 -d banco_um
select * from esquema_um.tabela_um;

Sem acesso à tabela_um.

Vamos devolver seu privilégio de SELECT na tabela_um, mas para o grupo usuario, de quem usuario1 herda.

```
\q
psql -U super1 -d banco_um
grant select on esquema_um.tabela_um to usuario;

\q
psql -U usuario1 -d banco_um
select * from esquema_um.tabela_um;
```

Agora conseguiu.

Isso é bom, precisamos apenas setar as permissões para o grupo e todos os seus membros herdam as mesmas.

Permissões

rolename=xxxx -- privileges granted to a role
=xxxx -- privileges granted to PUBLIC

r -- SELECT ("read")
w -- UPDATE ("write")
a -- INSERT ("append")
d -- DELETE
D -- TRUNCATE
x -- REFERENCES
t -- TRIGGER
X -- EXECUTE
U -- USAGE
C -- CREATE
c -- CONNECT
T -- TEMPORARY
arwdDxt -- ALL PRIVILEGES (for tables, varies for other objects)
* -- grant option for preceding privilege

/yyyy -- role that granted this privilege

SET

psql
ALTER DATABASE test SET enable_indexscan TO off;

SET search_path TO esquema_um, public;

SET datestyle TO banco_um, dmy;

SET TIME ZONE 'America/Fortaleza';

RESET timezone;

SHOW

Mostrar parâmetros em tempo de execução

Mostrar estilo de dadas
SHOW DateStyle;

Mostrar todos os parâmetros do SGBD
SHOW ALL;

\h show

\h set

Dicas Extras:

Usuários são Globais em todo o Agrupamento de Bancos de Dados
Todos os usuários são globais para todo o agrupamento de bancos de dados. Um usuário pode ter acesso a qualquer banco de dados.

Superusuários
Não estão sujeitos à verificação de permissão. Tem direito de fazer o que bem entender em qualquer banco de dados. Somente um superusuário pode criar usuários. Para criar um superusuário usamos o comando:
create role nome_user createrole;

Para que um usuário tenha privilégio de criar bancos de dados devemos conceder assim:
create role nome_user createdb;

Grupos
São uma forma lógica de juntar usuários para facilitar o gerenciamento de privilágios. Neste caso concedemos ou revogamos privilégios para todo o grupo, que fica mais prático. Cruar um usuário em um certo grupo:
create role user in role grupo password 'user';

GRANT e REVOKE
O comando GRANT concede privilégios específicos para um objeto (tabela, visão, seqüência, banco de dados, função, linguagem procedural, esquema ou espaço de tabelas) para um ou mais usuários ou grupos de usuários. Estes privilégios são adicionados aos já concedidos, se existirem.

A palavra chave PUBLIC indica que os privilégios devem ser concedido para todos os usuários, inclusive aos que vierem a ser criados posteriormente.

Se for especificado WITH GRANT OPTION quem receber o privilégio poderá, por sua vez, conceder o privilégio a terceiros.

Os privilégios especiais do dono da tabela (ou seja, o direito de DROP (remover), GRANT (conceder), REVOKE (revogar), etc.) são sempre implícitos ao fato de ser o dono, não podendo ser concedidos ou revogados. Mas o dono da tabela pode decidir revogar seus próprios privilégios comuns como, por exemplo, tornando uma tabela somente para leitura para o próprio e para os outros.

Listar grupos:
select groname from pg_group;
ou
\dg

Donos dos Objetos
Quando um objeto do banco de dados é criado é atribuído um dono ao mesmo. O dono é o usuário que executou o comando de criação do objeto. Por padrão somente o dono pode fazer qualquer coisa com o objeto. Para que outros usuários tenham acesso ao objeto o dono precisa conceder os privilágios usando o comando GRANT.

- Para maior segurnça, sempre antes de conceder somente os privilégios necessários para um usuário sobre um objeto, remova todos os privilégios sobre o objeto do usuário.

Exemplo:

REVOKE ALL ON tabela FROM usuario;
GRANT PRIVILÉGIOS ON tabela TO usuario;

- Ao conceder privilégios sobre uma tabela que contém um campo do tipo serial, também precisamos conceder privilégios para a sequência gerada pelo serial.
O comando abaixo mostra a sequência:
\d

- Remover privilégios de acesso a um esquema para todos os usuários:
revoke create on schema public from public;

- Nenhum usuário tenha acesso a uma tabela:
revoke all on tabela from public;

- Funções e Gatilhos
As funções e os gatilhos permitem que usuários insiram código no servidor que outros usuários podem executar sem conhecer. A única proteção real é um controle rígido sobre quem pode definir funções. Estas funções podem burlar qualquer sistema de controle de acesso. As linguagens de função que permitem este tipo de acesso são consideradas "não confiáveis" (untrusted), e o PostgreSQL somente permite que superusuários criem funções escritas nestas linguagens.

Alguns Privilégios:

CREATE
Para bancos de dados, permite a criação de novos esquemas no banco de dados.

Para esquemas, permite a criação de novos objetos no esquema. Para mudar o nome de um objeto existente é necessário ser o dono do objeto e possuir este privilégio no esquema que o contém.

Para tablespaces, permite a criação de tabelas e índices no espaço de tabelas, e permite a criação de bancos de dados possuindo este espaço de tabelas como seu espaço de tabelas padrão.

CONNECT
Permite ao usuário se conectar ao banco de dados especificados, também serão verificadas as restrições impostas pelo postgresql.conf e pelo pg_hba.conf.

USAGE

Para as linguagens procedurais, permite o uso da linguagem especificada para criar funções nesta linguagem. Este é o único tipo de privilégio aplicável às linguagens procedurais.

Para os esquemas, permite acessar os objetos contidos no esquema especificado (assumindo que os privilégios requeridos para os próprios objetos estejam atendidos). Essencialmente, concede a quem recebe o direito de "procurar" por objetos dentro do esquema.

Para as sequências permite usar o nextval e currval

EXECUTE
Este é o único tipo de privilégio aplicável às funções

ALL PRIVILEGES

Concede todos os privilégios disponíveis de uma só vez. A palavra chave PRIVILEGES é opcional no PostgreSQL, embora seja requerida pelo SQL estrito.

Os privilégios aplicáveis a um determinado tipo de objeto variam de acordo com o tipo de objeto, como pode ser visto acima.

Exemplo de Uso de Usuários e Privilégios

Criação do usuário us_dnocs - super usuário para administrar o banco db_intranet

ssh ribamar@10.10.0.60

sudo su
su - postgres
psql

CREATE ROLE us_dnocs WITH LOGIN SUPERUSER PASSWORD 'abcd1029@';

Listar usuários
\du

Sair do psql
\q

Conectar ao banco postgres (precisamos indicar um banco, pois não existe o banco user_dnocs)

psql -U us_dnocs -d postgres

Agora ele pede a senha e consegue efetuar o login
Agora irei criar um grupo de usuários/role chamado "gr_intranet" que se destinará a criação dos usuários para cada esquema do db_intranet

Será criado com o usuário us_dnocs e não precisamos dar senha a ele, pois não faremos login, apenas com os que serão criados através dele

\q
psql -U us_dnocs -d postgres

create role gr_intranet WITH LOGIN;
\q

Criação do banco de dados db_intranet pelo usuário us_dnocs e tornando o grupo gr_intranet seu dono;

psql -U us_dnocs -d postgres
create database db_intranet owner gr_intranet;

\l

Acessar o db_intranet com gr_intranet

psql -U gr_intranet -d db_intranet
\q

Este user pode criar qualquer objeto no banco db_intranet.

Remover o esquema public do db_intranet para evitar confusões e tornar o mesmo mais seguro.

psql -U us_dnocs -d db_intranet
drop schema public;

Criar um usuário chamado us_testes para gerenciar o esquema sc_testes e ser seu dono. Este usuário pertencerá ao grupo gr_intranet e será criado pelo usuário us_dnocs

psql -U us_dnocs -d postgres

create role us_testes in role gr_intranet login password 'senhaforte';
\du
\q

Tornar o us_testes o dono do sc_testes

psql -U us_dnocs -d db_intranet

create schema sc_testes AUTHORIZATION us_testes;

GRANT ALL ON SCHEMA sc_testes TO us_testes;
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA sc_testes TO us_testes;
GRANT USAGE ON ALL SEQUENCES IN SCHEMA sc_testes TO us_testes;

Ficar atento, pois talvez seja necessário executar o último comando após importar o script.

Caso ao executar algum comando do cake para cadastrar algo no banco e aparecer a mensagem de que o banco não tem nenhuma tabela execute os dois últimos comandos acima.

Acessar o sc_testes com us_testes

psql -U us_testes -d db_intranet

Tornar o schema sc_testes o default.

Podemos criar objetos aqui mas especificando o esquema como prefixo. Ex:
create table sc_testes.tabela1(campo1 int);

SET search_path TO sc_testes;

Agora podemos criar e excluir sem especificar o esquema, pois estamos nele:

create testes.tabela1(campo1 int);

Liberando, como root, acesso para o servidor de arquivos, que está no IP 172.16.5.15

\q
exit
nano /var/lib/pgsql/9.4/data/pg_hba.conf

Precisamos ter uma linha assim:

host db_intranet all 172.16.5.15/32 md5

E reiniciar o postgresql

service postgresql-9.4 restart

Tornei meu usuário ribamar_sousa superuser para poder acessar todos os bancos do meu desktop.

alter role ribamar superuser;

Referências:
Documentação do PostgreSQL 8 em português - http://pgdocptbr.sourceforge.net/pg80/

http://pgdocptbr.sourceforge.net/pg80/app-psql.html
https://www.postgresql.org/docs/9.5/static/app-psql.html

http://pgdocptbr.sourceforge.net/pg80/sql-createuser.html
https://www.postgresql.org/docs/9.5/static/sql-createrole.html (Create Role aparece na versão 8.1)

http://pgdocptbr.sourceforge.net/pg80/sql-alterdatabase.html
https://www.postgresql.org/docs/9.5/static/sql-alterdatabase.html

http://pgdocptbr.sourceforge.net/pg80/sql-grant.html
https://www.postgresql.org/docs/9.5/static/sql-grant.html

http://pgdocptbr.sourceforge.net/pg80/sql-revoke.html
https://www.postgresql.org/docs/9.5/static/sql-revoke.html

http://pgdocptbr.sourceforge.net/pg80/sql-set.html
https://www.postgresql.org/docs/9.5/static/sql-set.html

http://pgdocptbr.sourceforge.net/pg80/sql-show.html
https://www.postgresql.org/docs/9.5/static/sql-show.html

Dicas Extras:

Usuários são Globais em todo o Agrupamento de Bancos de Dados
Todos os usuários são globais para todo o agrupamento de bancos de dados. Um usuário pode ter acesso a qualquer banco de dados.

Superusuários
Não estão sujeitos à verificação de permissão. Tem direito de fazer o que bem entender em qualquer banco de dados. Somente um superusuário pode criar usuários. Para criar um superusuário usamos o comando:
create role nome_user createrole;

Para que um usuário tenha privilégio de criar bancos de dados devemos conceder assim:
create role nome_user createdb;

Grupos
São uma forma lógica de juntar usuários para facilitar o gerenciamento de privilágios. Neste caso concedemos ou revogamos privilégios para todo o grupo, que fica mais prático. Cruar um usuário em um certo grupo:
create role user in role grupo password 'user';

GRANT e REVOKE
O comando GRANT concede privilégios específicos para um objeto (tabela, visão, seqüência, banco de dados, função, linguagem procedural, esquema ou espaço de tabelas) para um ou mais usuários ou grupos de usuários. Estes privilégios são adicionados aos já concedidos, se existirem.

A palavra chave PUBLIC indica que os privilégios devem ser concedido para todos os usuários, inclusive aos que vierem a ser criados posteriormente.

Se for especificado WITH GRANT OPTION quem receber o privilégio poderá, por sua vez, conceder o privilégio a terceiros.

Os privilégios especiais do dono da tabela (ou seja, o direito de DROP (remover), GRANT (conceder), REVOKE (revogar), etc.) são sempre implícitos ao fato de ser o dono, não podendo ser concedidos ou revogados. Mas o dono da tabela pode decidir revogar seus próprios privilégios comuns como, por exemplo, tornando uma tabela somente para leitura para o próprio e para os outros.

Listar grupos:
select groname from pg_group;
ou
\dg

Donos dos Objetos
Quando um objeto do banco de dados é criado é atribuído um dono ao mesmo. O dono é o usuário que executou o comando de criação do objeto. Por padrão somente o dono pode fazer qualquer coisa com o objeto. Para que outros usuários tenham acesso ao objeto o dono precisa conceder os privilágios usando o comando GRANT.

- Para maior segurnça, sempre antes de conceder somente os privilégios necessários para um usuário sobre um objeto, remova todos os privilégios sobre o objeto do usuário.

Exemplo:

REVOKE ALL ON tabela FROM usuario;
GRANT PRIVILÉGIOS ON tabela TO usuario;

- Ao conceder privilégios sobre uma tabela que contém um campo do tipo serial, também precisamos conceder privilégios para a sequência gerada pelo serial.
O comando abaixo mostra a sequência:
\d

- Remover privilégios de acesso a um esquema para todos os usuários:
revoke create on schema public from public;

- Nenhum usuário tenha acesso a uma tabela:
revoke all on tabela from public;

- Funções e Gatilhos
As funções e os gatilhos permitem que usuários insiram código no servidor que outros usuários podem executar sem conhecer. A única proteção real é um controle rígido sobre quem pode definir funções. Estas funções podem burlar qualquer sistema de controle de acesso. As linguagens de função que permitem este tipo de acesso são consideradas "não confiáveis" (untrusted), e o PostgreSQL somente permite que superusuários criem funções escritas nestas linguagens.

Alguns Privilégios:

CREATE

Para bancos de dados, permite a criação de novos esquemas no banco de dados.

Para esquemas, permite a criação de novos objetos no esquema. Para mudar o nome de um objeto existente é necessário ser o dono do objeto e possuir este privilégio no esquema que o contém.

Para tablespaces, permite a criação de tabelas e índices no espaço de tabelas, e permite a criação de bancos de dados possuindo este espaço de tabelas como seu espaço de tabelas padrão.

CONNECT
Permite ao usuário se conectar ao banco de dados especificados, também serão verificadas as restrições impostas pelo postgresql.conf e pelo pg_hba.conf.

USAGE

Para as linguagens procedurais, permite o uso da linguagem especificada para criar funções nesta linguagem. Este é o único tipo de privilégio aplicável às linguagens procedurais.

Para os esquemas, permite acessar os objetos contidos no esquema especificado (assumindo que os privilégios requeridos para os próprios objetos estejam atendidos). Essencialmente, concede a quem recebe o direito de "procurar" por objetos dentro do esquema.

Para as sequências permite usar o nextval e currval

EXECUTE
Este é o único tipo de privilégio aplicável às funções

ALL PRIVILEGES

Concede todos os privilégios disponíveis de uma só vez. A palavra chave PRIVILEGES é opcional no PostgreSQL, embora seja requerida pelo SQL estrito.

Os privilégios aplicáveis a um determinado tipo de objeto variam de acordo com o tipo de objeto, como pode ser visto acima.

Exemplo de Uso de Usuários e Privilégios

Criação do usuário us_dnocs - super usuário para administrar o banco db_intranet

ssh ribamar@10.10.0.60

sudo su
su - postgres
psql

CREATE ROLE us_dnocs WITH LOGIN SUPERUSER PASSWORD 'abcd1029@';

Listar usuários
\du

Sair do psql

\q

Conectar ao banco postgres (precisamos indicar um banco, pois não existe o banco user_dnocs)

psql -U us_dnocs -d postgres

Agora ele pede a senha e consegue efetuar o login
Agora irei criar um grupo de usuários/role chamado "gr_intranet" que se destinará a criação dos usuários para cada esquema do db_intranet

Será criado com o usuário us_dnocs e não precisamos dar senha a ele, pois não faremos login, apenas com os que serão criados através dele

\q

psql -U us_dnocs -d postgres

create role gr_intranet WITH LOGIN;
\q

Criação do banco de dados db_intranet pelo usuário us_dnocs e tornando o grupo gr_intranet seu dono;

psql -U us_dnocs -d postgres
create database db_intranet owner gr_intranet;

\l

Acessar o db_intranet com gr_intranet

psql -U gr_intranet -d db_intranet
\q

Este user pode criar qualquer objeto no banco db_intranet.

Remover o esquema public do db_intranet para evitar confusões e tornar o mesmo mais seguro.

psql -U us_dnocs -d db_intranet
drop schema public;

Criar um usuário chamado us_testes para gerenciar o esquema sc_testes e ser seu dono. Este usuário pertencerá ao grupo gr_intranet e será criado pelo usuário us_dnocs

psql -U us_dnocs -d postgres

create role us_testes in role gr_intranet login password 'senhaforte';
\du

\q

Tornar o us_testes o dono do sc_testes

psql -U us_dnocs -d db_intranet

create schema sc_testes AUTHORIZATION us_testes;

GRANT ALL ON SCHEMA sc_testes TO us_testes;
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA sc_testes TO us_testes;
GRANT USAGE ON ALL SEQUENCES IN SCHEMA sc_testes TO us_testes;

Ficar atento, pois talvez seja necessário executar o último comando após importar o script.

Caso ao executar algum comando do cake para cadastrar algo no banco e aparecer a mensagem de que o banco não tem nenhuma tabela execute os dois últimos comandos acima.

Acessar o sc_testes com us_testes

psql -U us_testes -d db_intranet

Tornar o schema sc_testes o default.

Podemos criar objetos aqui mas especificando o esquema como prefixo. Ex:
create table sc_testes.tabela1(campo1 int);

SET search_path TO sc_testes;

Agora podemos criar e excluir sem especificar o esquema, pois estamos nele:

create testes.tabela1(campo1 int);

Liberando, como root, acesso para o servidor de arquivos, que está no IP 172.16.5.15

\q
exit
nano /var/lib/pgsql/9.4/data/pg_hba.conf

Precisamos ter uma linha assim:

host db_intranet all 172.16.5.15/32 md5

E reiniciar o postgresql

service postgresql-9.4 restart

Tornei meu usuário ribamar_sousa superuser para poder acessar todos os bancos do meu desktop.

alter role ribamar superuser;

Obs: o conteúdo acima é o capítulo 50 do livro
https://ribafs.github.io/curriculo/livros/sgbd-postgresql.html