UNIVERSIDADE FEDERAL DE ITAJUBÁ

Banco de Dados II

COM 231

Segurança em Banco de Dados
Usuários e Concessão de Privilégios
Aula 3

Vanessa Cristina Oliveira de Souza

Segurança

- Os dados armazenados no banco de dados precisam ser protegidos contra:
  - acessos não-autorizados
  - destruição ou alteração intencional
  - introdução ou alteração de inconsistência

Ameaças aos bancos de dados

- Perda de Integridade
  - A integridade é perdida se mudanças não autorizadas forem feitas nos dados por atos intencionais ou acidentais.
- Perda de Disponibilidade
  - Se um usuário ou um programa perde o acesso aos dados.
- Perda de Confidencialidade
  - A exposição não autorizada de um dado pode resultar em perda de confiança pública, constrangimento ou ação legal contra a organização mantenedora dos dados.

Ameaças aos bancos de dados

Segurança e informação são valores inseparáveis em computação!

Segurança X Integridade

- Segurança
  - Refere-se a segurança contra acessos maldosos.
    - Autorização
    - Visões
- Integridade
  - Refere-se ao ato de evitar a perda acidental de consistência.
    - Integridade referencial
    - Escalonamentos
    - Transações
    - Recuperação

Violações de Integridade

- Quebras durante o processamento de transações.
- Anomalias causadas por acesso concorrente ao banco de dados.
- Anomalias causadas pela distribuição de dados sobre diversos computadores.
- Um erro lógico que viola a suposição de que as transações preservam as restrições de consistência do banco de dados.

Violações de Segurança

- Leitura não-autorizada de dados (roubo de informação).
- Modificação não-autorizada de dados.
- Destruição não-autorizada de dados.

Medidas de Segurança

- A fim de proteger o banco de dados, medidas de segurança precisam ser tomadas em diversos níveis:
  - Físico
  - Humano
  - Sistema Operacional
  - Sistema de Banco de Dados
- A segurança de todos os níveis precisa ser mantida a fim de garantir a segurança do banco de dados.
- Uma fraqueza a um nível baixo de segurança (físico ou humano) permite contornar-se medidas de segurança de alto nível (banco de dados).

Medidas de Segurança

- Nível Físico
  - O local ou locais onde os sistemas de computador estão localizados precisam estar fisicamente protegidos contra assaltos ou intrusos.

Medidas de Segurança

- Bom fornecimento de energia
  - Instalação elétrica dedicada e balanceada;
  - No-breaks redundantes com carga compatível e bateria não vencida;
  - Geradores com carga compatível;
- Bom acondicionamento
  - Ar condicionado suficiente e redundante;
  - Boa acomodação (racks), bons gabinetes;
  - Segurança contra incêndio e desastres naturais;
- Equipe
  - Monitoramento constante dos sistemas;
- Backup

Medidas de Segurança

- Nível Humano
  - Os usuários devem ser cautelosamente autorizados para reduzir a chance de qualquer usuário dar acesso a um intruso em troca de suborno ou outros favores.

Medidas de Segurança

- Nível de Sistema Operacional
  - A fraqueza na segurança do sistema operacional pode servir como um meio para acesso não-autorizado ao banco de dados.
  - Uma vez que quase todos os sistemas de banco de dados permitem o acesso remoto através de terminais ou redes, a segurança no nível do *software* dentro do sistema operacional é tão importante quanto no nível físico.

Medidas de Segurança

- Nível de Sistema de Banco de Dados
  - Alguns usuários de banco de dados podem estar autorizados a fazer o acesso apenas a uma porção limitada do banco de dados.
  - A outros usuários pode ser permitida a formulação de consultas, mas proibida a modificação de dados.
  - É responsabilidade do sistema de banco de dados assegurar que essas restrições não sejam violadas.

Autenticação

Autenticação

- 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.
  - Acontece antes da conexão propriamente dita

Autenticação no PostgreSQL

- O *PostgreSQL* possui arquivos de configuração que afetam questões diversas que vão de segurança da informação à performance do banco.

- O arquivo **pg_hba.conf** é o principal arquivo de controle de autenticação de usuários no servidor.
  - host-based authentication
  - \PostgreSQL\9.3\data\ pg_hba.conf
  - Controla que máquinas terão acesso ao PostgreSQL e a autenticação dessas máquinas clientes (sem autenticação ou através de outras formas, trust, md5, crypt, etc)

Autenticação no PostgreSQL

- O arquivo **pg_hba.conf** é o principal arquivo de controle de autenticação de usuários no servidor.
- Este arquivo controla:
  - Quais hosts têm permissão de conectar
  - Como os clientes são autenticados
  - Nomes dos usuários que podem usar
  - Quais bancos eles podem acessar
- O formato básico do registro possui os campos
  - TYPE, DATABASE, USER, CIDR-ADDRESS e METHOD.

Autenticação no PostgreSQL

- O formato básico do registro possui os campos :
  - TYPE
  - Diz respeito ao tipo de conexão a qual a regra será aplicada: **local, host, hostssl e hostnossl**.
  - As conexões podem ser locais (local) ou externas (host), sendo possível determinar configurações que se apliquem apenas à conexões seguras (hostssl) ou justamente àquelas não seguras (hostnossl).
    - Conexão via TCP/IP -> Host -> localhost
    - IPV6 -> ::1/128

Autenticação no PostgreSQL

- O formato básico do registro possui os campos :
  - DATABASE
  - Define o nome do banco ao qual se refere o registro
  - ALL – todos os bancos

Autenticação no PostgreSQL

- O formato básico do registro possui os campos :
  - USER
  - Define os usuários a quem se aplica a regra, que pode ser o próprio usuário, um grupo ou uma lista.
  - indicar o nome do grupo precedido de um + no pg_hba.conf

```
TYPE        DATABASE        USER
host        producao        +admins
```

Autenticação no PostgreSQL

- O formato básico do registro possui os campos :
  - CIDR-ADDRESS
  - No campo CIDR (*Classless Inter-Domain Routing*) são inseridos o IPs que devem ser habilitados e a máscara de rede.
  - Não preencher este campo para registros do tipo "local",

Autenticação no PostgreSQL

- O formato básico do registro possui os campos :
  - METHOD
  - Define o método de autenticação
    - trust - é a forma menos segura; não será solicitado sequer senha e o usuário poderá acessar o banco como qualquer outro usuário, inclusive postgres.
    - password - requer o password do usuário que está tentando se conectar. A senha não vem criptografada.
    - md5 - requer o password do usuário que está tentando se conectar. A senha vem criptografada.
    - ident - o usuário é obtido a partir do sistema operacional cliente e esse dado é que é passado ao PostgreSQL.

Autenticação no PostgreSQL

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD

# 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.
#host    replication     postgres        127.0.0.1/32            md5
#host    replication     postgres        ::1/128                 md5
```

UNIVERSIDADE FEDERAL DE ITAJUBÁ

Criação de Usuários

Usuários

- Todo **agrupamento de bancos de dados** possui um **conjunto de usuários** de banco de dados.

- Existem cinco tipos de usuários de banco de dados, segundo o modo como o qual interagem com o sistema:
  - DBA
  - Programadores de aplicativos
  - Usuários de alto nível
  - Usuários especializados
  - Usuários ingênuos

Ameaças aos bancos de dados

Posição	2013 - Principais Ameaças	2015 - Principais Ameças
1	Privilégios excessivos ou esquecidos.	Privilégios excessivos ou esquecidos.
2	Abuso de privilégio	Abuso de privilégio
3	SQL Injection	Input Injection
4	Malware	Malware
5	Auditoria fraca	Auditoria fraca
6	Exposição de mídia de storage	Exposição de mídia de storage
7	Exploração de vulnerabilidades e configurações fracas de banco de dados	Exploração de vulnerabilidades e configurações fracas de banco de dados
8	Dados sensíveis sem politicas de segurança.	Dados sensíveis sem politicas de segurança.
9	DoS - Negação de Serviço	DoS - Negação de Serviço
10	Pouca experiência dos profissionais na área de segurança.	Pouca experiência dos profissionais na área de segurança.

https://www.perallis.com/news/10-principais-causas-de-ataques-a-banco-de-dados

Segurança a nível de SGBD

1
- Criação de Usuários
- Quem? Perfil? PQ?
- Senha

2
- Autenticação
- Local/ Host
- Método de autenticação

3
- Concessão de Privilégios
- Quem? Perfil? PQ?
- Sem *Grant Option*

Criação de Usuários em PostgreSQL

- Para criar um usuário deve ser utilizado o comando SQL CREATE USER:

CREATE USER nome_do_usuário;

- Cria um usuário sem senha:

CREATE USER ‘nome_do_usuário’ WITH PASSWORD ‘senha’;

- Cria um usuário com senha:

Criação de Usuários em PostgreSQL

- Para criar um usuário deve ser utilizado o comando SQL CREATE USER:

```
CREATE USER nome [ [ WITH ] opção [ ... ] ]

onde opção pode ser:

      SYSID id_do_usuário
    | CREATEDB | NOCREATEDB
    | CREATEUSER | NOCREATEUSER
    | IN GROUP nome_do_grupo [, ...]
    | [ ENCRYPTED | UNENCRYPTED ] PASSWORD 'senha'
    | VALID UNTIL 'data_e_hora'
```

Criação de Usuários em PostgreSQL

- Para criar um usuário deve ser utilizado o comando SQL CREATE USER:

```
CREATE USER manuel WITH PASSWORD 'jw8s0F4';
```

```
CREATE USER miriam WITH PASSWORD 'jw8s0F4' VALID UNTIL '2005-01-01';
```

```
CREATE USER manuel WITH PASSWORD 'jw8s0F4' CREATEDB;
```

SELECT * FROM pg_user;

Criação de Usuários em MySQL

- Outros Exemplos:
  - CREATE USER ‘nome_do_usuário’@‘localhost’ IDENTIFIED BY senha;
  - CREATE USER ‘nome_do_usuário’@‘unifei.edu.br’ IDENTIFIED BY senha;
  - CREATE USER ‘nome_do_usuário’@‘%.unifei.edu.br’ IDENTIFIED BY senha;

Exercícios

- Criar um usuário com senha;
- Logar como o usuário criado;
- Fazer select em alguma tabela;
- Apagar o usuário criado;

Criação de Usuários

- Os usuários criados não possuem nenhum privilégio no banco. Apenas podem conectar no servidor.
  - Exceto superusuários

UNIVERSIDADE FEDERAL DE ITAJUBÁ

Concessão de Privilégios

Autorização de Acesso aos Dados

- Um usuário pode ter diversas formas de autorização a partes do banco de dados:
  - Autorização leitura
  - Autorização inserção
  - Autorização atualização
  - Autorização eliminação

- Todas, nenhuma ou uma combinação desses tipos de autorização pode ser concedida a um usuário.

Autorização de Esquema

- Além da autorização de acesso aos dados, pode ser concedidas autorizações para modificar o esquema do banco de dados:
  - Autorização índice
  - Autorização recursos
  - Autorização alteração
  - Autorização remoção

- Todas, nenhuma ou uma combinação desses tipos de autorização pode ser concedida a um usuário.

UNIVERSIDADE FEDERAL DE ITAJUBÁ

Concessão de Privilégios

O Comando GRANT

- O comando GRANT permite aos administradores do sistema criar usuários e conceder direitos aos usuários.

O Comando GRANT

- Existem quatro níveis de privilégios:
  - Nível Global
    - Aplicam privilégios para todos os bancos de dados em um determinado servidor.
  - Nível de Banco de Dados
    - Privilégios de bancos de dados aplicam-se a todas as tabelas em um determinado banco de dados.

O Comando GRANT

- Existem quatro níveis de privilégios:
  - Nível de Tabela
    - Privilégios de tabelas aplicam-se a todas as colunas em uma determinada tabela.
  - Nível de Coluna
    - Privilégios de colunas aplicam-se a uma única coluna em uma determinada tabela.

Sintaxe do Comando GRANT

GRANT <privilégios (colunas)>

ON <item>

TO <usuário>

(WITH GRANT OPTION)

Sintaxe do Comando GRANT

```
GRANT { { SELECT | INSERT | UPDATE | DELETE | REFERENCES | TRIGGER }
    [,...] | ALL [ PRIVILEGES ] }
    ON [ TABLE ] tablename [, ...]
    TO { [ GROUP ] rolename | PUBLIC } [, ...] [ WITH GRANT OPTION ]

GRANT { { USAGE | SELECT | UPDATE }
    [,...] | ALL [ PRIVILEGES ] }
    ON SEQUENCE sequencename [, ...]
    TO { [ GROUP ] rolename | PUBLIC } [, ...] [ WITH GRANT OPTION ]

GRANT { { CREATE | CONNECT | TEMPORARY | TEMP } [,...] | ALL [ PRIVILEGES ] }
    ON DATABASE dbname [, ...]
    TO { [ GROUP ] rolename | PUBLIC } [, ...] [ WITH GRANT OPTION ]

GRANT { EXECUTE | ALL [ PRIVILEGES ] }
    ON FUNCTION funcname ( [ [ argmode ] [ argname ] argtype [, ...] ] ) [, ...]
    TO { [ GROUP ] rolename | PUBLIC } [, ...] [ WITH GRANT OPTION ]

GRANT { USAGE | ALL [ PRIVILEGES ] }
    ON LANGUAGE langname [, ...]
    TO { [ GROUP ] rolename | PUBLIC } [, ...] [ WITH GRANT OPTION ]

GRANT { { CREATE | USAGE } [,...] | ALL [ PRIVILEGES ] }
    ON SCHEMA schemaname [, ...]
    TO { [ GROUP ] rolename | PUBLIC } [, ...] [ WITH GRANT OPTION ]

GRANT { CREATE | ALL [ PRIVILEGES ] }
    ON TABLESPACE tablespacename [, ...]
    TO { [ GROUP ] rolename | PUBLIC } [, ...] [ WITH GRANT OPTION ]

GRANT role [, ...] TO rolename [, ...] [ WITH ADMIN OPTION ]
```

http://www.postgresql.org/docs/8.3/static/sql-grant.html

Exemplos Comando GRANT

- Nível de Banco de Dados

```
GRANT ALL ON DATABASE northwind TO vanessa;
```

  - CREATE, CONNECT and TEMPORARY

- Nível de Schema

```
GRANT ALL ON SCHEMA northwind TO vanessa;
```

- Nível de Tabelas

```
GRANT ALL ON TABLE northwind.customers TO vanessa;
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA northwind TO vanessa;
```

Exemplos Comando GRANT

- Nível de Colunas

```
GRANT SELECT (col1), UPDATE (col1) ON mytable TO miriam_rw;
```

select * from information_schema.role_table_grants where grantee='vanessa';

Exemplos Comando GRANT

- Verificar permissões de um usuário

  select * from information_schema.role_table_grants where grantee='vanessa';

- No plsql – verificar permissões de uma tabela

```
=> \dp mytable
                                Access privileges
 Schema |  Name   | Type  |   Access privileges   | Column access privileges
--------+---------+-------+-----------------------+--------------------------
 public | mytable | table | miriam=arwdDxt/miriam | col1:
                          : =r/miriam             :   miriam_rw=rw/miriam
                          : admin=arw/miriam
```

Sintaxe do Comando GRANT

GRANT <privilégios (colunas)>

ON <item>

TO <usuário>

(WITH GRANT OPTION)

- Se especificado, o usuário pode conceder seus privilégios a outros usuários.

O comando REVOKE

- O comando REVOKE permite aos administradores do sistema remover usuários e privilégios dos usuários.

Sintaxe do comando REVOKE

REVOKE <privilégios (colunas)>

ON item

FROM usuario

Exemplos Comando REVOKE

- **Privilégios a nível de Coluna.**

- Remover todos os privilégios do usuário “someuser” sobre a tabela “minhaTabela” do banco “MeuBanco”;

  - REVOKE ALL ON meuBanco.minhaTabela FROM someuser;

Exemplos Comando REVOKE

- REVOKE ALL ON DATABASE northwind FROM vanessa;

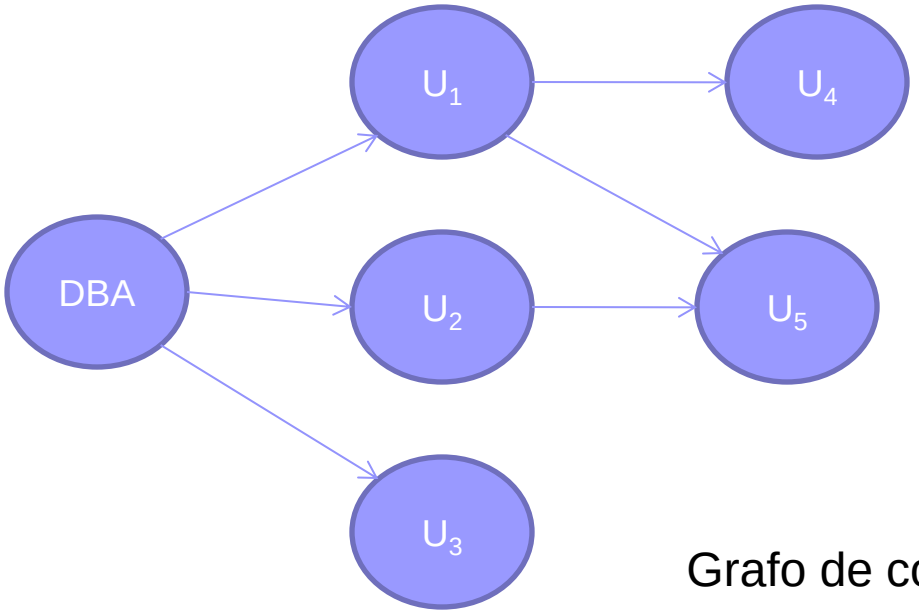
Concessão de Autorização

- Um usuário que tem concedida alguma forma de autoridade pode passar esta autoridade para outros usuários.

- Cuidados precisam ser tomados para assegurar que tal autorização possa ser revogada em momento futuro por quem a concedeu.

Concessão de Autorização

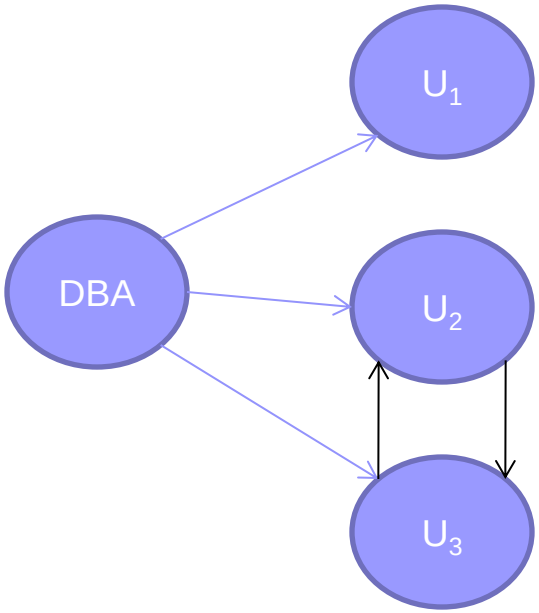
- Um usuário que tem concedida alguma forma de autoridade pode passar esta autoridade para outros usuários.



Grafo de concessão de autorização

Forma de burlar uma revogação de autorização

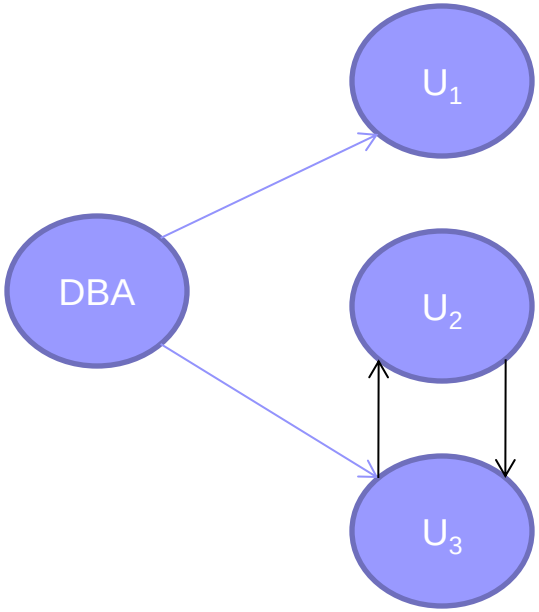
- O DBA concede autorização para os usuário 1, 2 e 3.
- O usuário 2 repassa seus direitos para o usuário 3.
- O usuário 3 repassa seus direitos para o usuário 2.



Grafo de concessão de autorização

Forma de burlar uma revogação de autorização

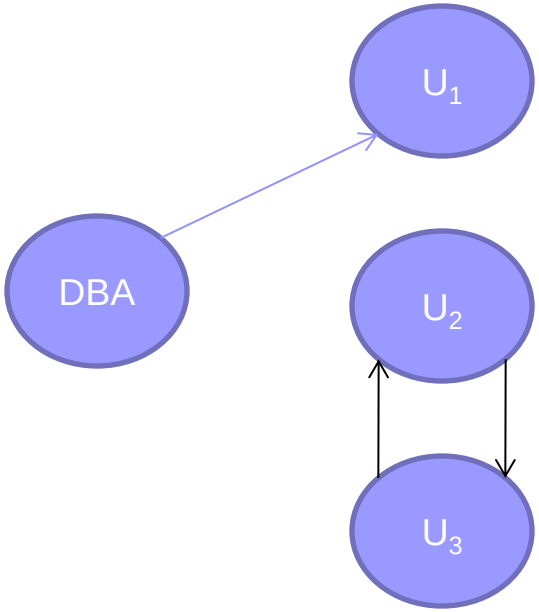
- O DBA revoga autorização para os usuário 2.
- O usuário 2 continua tendo os direitos concedidos pelo usuário 3.



Grafo de concessão de autorização

Forma de burlar uma revogação de autorização

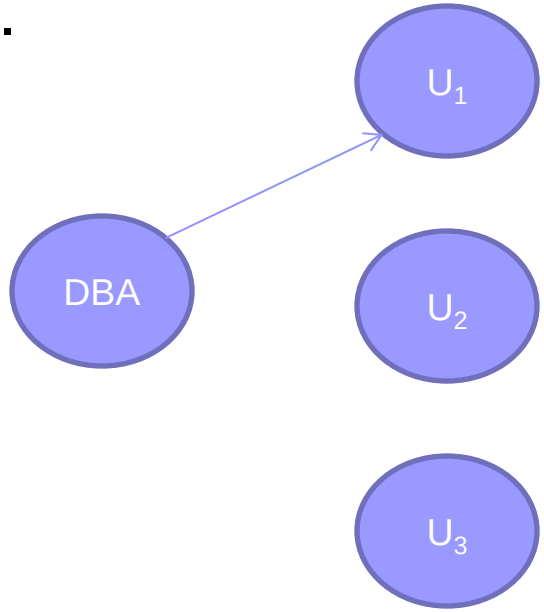
- O DBA revoga autorização para os usuário 3.
- O usuário 3 continua tendo os direitos concedidos pelo usuário 2.



Grado de concessão de autorização

Forma de burlar uma revogação de autorização

- Para evitar problemas como esse, requere-se que todas as arestas num grafo de autorização sejam parte de algum caminho originado no administrador do banco de dados.



Grafo de concessão de autorização

Exemplo

Usuário Postgres

```
CREATE USER user3 WITH PASSWORD 'senha';
GRANT ALL ON SCHEMA northwind TO user3 WITH GRANT OPTION;
GRANT SELECT, INSERT ON table northwind.customers TO user3 WITH GRANT OPTION;
GRANT SELECT, INSERT ON table northwind.orders TO user3 WITH GRANT OPTION;
```

```
38  select * from information_schema.role_table_grants where grantee='user3';
```

Data Output | Explain | Messages | Query History

	grantor character varying	grantee character varying	table_catalog character varying	table_schema character varying	table_name character varying	privilege_type character varying	is_grantable character varying (3)
1	postgres	user3	northwind	northwind	orders	INSERT	YES
2	postgres	user3	northwind	northwind	orders	SELECT	YES
3	postgres	user3	northwind	northwind	customers	INSERT	YES
4	postgres	user3	northwind	northwind	customers	SELECT	YES

Exemplo

Usuário Postgres

```
CREATE USER user4 WITH PASSWORD 'senha';
GRANT ALL ON SCHEMA northwind TO user4 WITH GRANT OPTION;
GRANT SELECT ON table northwind.customers TO user4 WITH GRANT OPTION;
```

```
38  select * from information_schema.role_table_grants where grantee='user4';
```

Data Output | Explain | Messages | Query History

	grantor character varying	grantee character varying	table_catalog character varying	table_schema character varying	table_name character varying	privilege_type character varying	is_grantable character varying (3)
1	postgres	user4	northwind	northwind	customers	SELECT	YES

Exemplo

Usuário user3

```
GRANT INSERT ON table northwind.orders TO user4 WITH GRANT OPTION;
```

```
40  select * from information_schema.role_table_grants where grantee='user4';
```

Data Output | Explain | Messages | Query History

	grantor character varying	grantee character varying	table_catalog character varying	table_schema character varying	table_name character varying	privilege_type character varying	is_grantable character vary
1	postgres	user4	northwind	northwind	customers	SELECT	YES
2	user3	user4	northwind	northwind	customers	INSERT	YES

Exemplo

Usuário Postgres

```
REVOKE ALL PRIVILEGES ON table northwind.customers FROM user4;
```

```
40  select * from information_schema.role_table_grants where grantee='user4';
```

Data Output | Explain | Messages | Query History

	grantor character varying	grantee character varying	table_catalog character varying	table_schema character varying	table_name character varying	privilege_type character varying	is_grantable character var
1	user3	user4	northwind	northwind	customers	INSERT	YES

Roles

- É importante que os usuários criados no banco pertençam a um grupo e os privilégios sejam dados ao grupo, e não para um usuário especificamente.
  - Evita que usuários do mesmo ‘tipo’ tenham permissões diferentes
  - Facilita a gestão dos usuários no banco
- Nos SGBDs atuais, a implementação se dá por meio do uso de ***roles***.

Roles

- Roles
  - *The concept of roles subsumes the concepts of "users" and "groups".*
  - *In PostgreSQL versions before 8.1, users and groups were distinct kinds of entities, but now there are only roles.*
  - *Any role can act as a user, a group, or both.*

Roles

- Exercício:
  - Criar uma role chamada ‘teste’;
  - Criar um usuário chamado ‘user1’ (+senha) e adicioná-lo à role teste.
  - Dar privilégios pra role de select na tabela Orders.

Roles

- Exercício:

```
CREATE ROLE teste;
CREATE USER user1 WITH PASSWORD 'senha' IN ROLE teste;
GRANT ALL ON SCHEMA northwind TO teste;
GRANT SELECT ON northwind.orders TO teste;
```

```
postgres=# \du
                                   Lista de roles
 Nome da role |                         Atributos                                  | Membro de
--------------+--------------------------------------------------------------------+-----------
 postgres     | Super-usußrio, Cria role, Cria BD, ReplicaþÒo, Ignora RLS           | {}
 teste        | NÒo pode efetuar login                                             | {}
 user1        |                                                                    | {teste}
```

Roles

- Exercício:
  - Conectar-se ao servidor com o usuário user1 e avaliar;

REVOKE

- Exercícios:
  - Remover o usuário ‘user1’ do banco;
  - Remover a role ‘teste’ do banco;

View + Grant

View x Grant

- O Grant garante um corte vertical na tabela, permitindo ao usuário manipular, no mínimo, uma coluna de uma tabela.

- Para permitir que um usuário tenha apenas acesso a um conjunto específico de registros, é preciso combinar a view com o grant.
  - Cria-se uma view
  - Concede ao usuário acesso apenas àquela view

View x Grant

- Exemplo:
  - Crie um novo grupo de usuários chamado ‘vendedoresMexico’;
  - Dê permissão de schema para esse grupo
  - Crie uma view chamada mexico sobre a tabela northwind.customers, filtrando por país (country like ‘Mexico’)
  - Dê permissão de select, insert, update e delete para o grupo vendedoresMexico sobre a view criada.
  - Crie o usuário vendedor1
  - Teste os privilégios para o vendedor1

Coisas importantes!

- Não confunda Integridade com Segurança!
- Usuários de aplicação não devem ser usuários de banco!!!!
- A definição dos usuários do banco e seus respectivos privilégios deve estar presente na documentação do banco

Segurança a nível de SGBD

1
- Criação de Usuários
- Quem? Perfil? PQ?
- Senha

2
- Autenticação
- Local/ Host
- Método de autenticação

3
- Concessão de Privilégios
- Quem? Perfil? PQ?
- Sem *Grant Option*

Para Casa

- Ler os itens 23.1, 23.2 e 23.3 do Elmasri e Navathe – 4ª Edição