Usuários e Permissões

Aula 16 — Quem pode ver e mudar o quê

UniSenac Centro Universitário RS
Senac
Fecomércio
IFEP
Sesc
Sindicatos Empresariais

O mapa de hoje

- **Bloco 1** — por que ninguém deveria usar o `postgres` no dia a dia
- **Bloco 2** — `CREATE ROLE`, `GRANT` e `REVOKE`
- **Bloco 3** — privilégio mínimo, views como proteção e LGPD
- **Bloco 4** — mão na massa: criar uma role restrita e **testar** que ela é restrita

> Este é o **Indicador 6** da competência: *operacionalizar permissões de acesso*. Não basta saber o conceito — hoje vocês configuram de verdade.

Retomando a Aula 15

- Quais funções do CRUD precisam de `commit()`?
- A view guarda os dados ou a consulta?
- Qual dos três usos de view tem a ver com **segurança**?

Bloco 1

Por que não usar o `postgres` para tudo

O que vocês vêm fazendo até agora

Desde a Aula 10, todo mundo conecta como `postgres` — o superusuário.

- Pode criar e apagar **qualquer** banco
- Pode apagar **qualquer** tabela
- Pode ler **qualquer** dado, de qualquer cliente
- Não existe nada que ele não possa fazer

> Isso é aceitável num banco de estudo, na sua máquina. Em qualquer outro lugar, é um problema sério.

⚠ O que dá errado

- O atendente **apaga sem querer** a tabela de clientes — e podia
- O estagiário **vê o telefone e o CPF** de todo mundo — e não precisava
- Um sistema invadido vira **acesso total**, não acesso parcial
- Ninguém sabe **quem** fez o quê: todo mundo é o mesmo usuário

> Lembram do **Problema 4** da Aula 01 — "quem apagou a linha 200?". Com um login só para todo mundo, a resposta continua sendo *"não dá para saber"*.

Role — usuário e grupo na mesma palavra

GLOSSÁRIO

`role`	papel — um usuário **ou** um grupo de permissões
`LOGIN`	pode se conectar ao banco
`superuser`	superusuário — pode tudo
`privilege`	privilégio — uma permissão específica

No PostgreSQL, **usuário e grupo são a mesma coisa**: uma *role*. Uma role com `LOGIN` funciona como usuário; sem `LOGIN`, funciona como grupo.

Checagem do Bloco 1

Na locadora trabalham a dona, duas atendentes e um estagiário de estoque. Hoje os quatro entram como `postgres`.

- Quem precisa poder **apagar** um empréstimo? E quem não pode?
- O estagiário roda `DROP TABLE emprestimo` por engano. O que o banco faz hoje?
- Depois do estrago, o que você **não** consegue descobrir?



`forms.gle/1CwPu1MuMXkLfXnW7`

Intervalo

10 minutos

Bloco 2

CREATE ROLE, GRANT e REVOKE

Criar a role

```
CREATE ROLE atendente WITH LOGIN PASSWORD 'senha_atendente';
```

GLOSSÁRIO

`CREATE ROLE`	criar papel
`WITH LOGIN`	com permissão de conectar
`PASSWORD`	senha

Uma role recém-criada **não pode fazer nada** — nem ver uma tabela. Tudo o que ela puder fazer, você vai conceder explicitamente.

GRANT — conceder

```
GRANT CONNECT ON DATABASE locadora TO atendente;
GRANT USAGE   ON SCHEMA public     TO atendente;

GRANT SELECT           ON jogo, cliente, plataforma TO atendente;
GRANT SELECT, INSERT   ON emprestimo, item_emprestimo TO atendente;
```

GLOSSÁRIO

GRANT	conceder
ON	sobre (qual objeto)
TO	para (qual role)

Os privilégios que existem

Privilégio	Permite
`SELECT`	ler as linhas
`INSERT`	inserir linhas novas
`UPDATE`	alterar linhas existentes
`DELETE`	apagar linhas
`ALL PRIVILEGES`	todos os acima

Repare que são exatamente as **4 letras do CRUD** da Aula 15. Permissão é definida *por verbo* e *por tabela* — o atendente pode inserir empréstimo, mas não pode apagar jogo.

REVOKE — retirar

```
REVOKE INSERT ON emprestimo FROM atendente;
```

Conceder é fácil; **lembrar de retirar** é o que falha na vida real. Pessoa muda de função, sai da empresa — e a permissão continua lá. É assim que vazamento acontece.

Role de grupo — concedendo para vários de uma vez

```
CREATE ROLE equipe_balcao;                          -- sem LOGIN: é um grupo
GRANT SELECT ON jogo, plataforma, jogos_plataforma TO equipe_balcao;

CREATE ROLE ana   WITH LOGIN PASSWORD '...';
CREATE ROLE bruno WITH LOGIN PASSWORD '...';

GRANT equipe_balcao TO ana, bruno;                  -- os dois herdam as permissões
```

Contrata mais um atendente? Um `GRANT equipe_balcao TO fulano` resolve. Mudou a regra do balcão? Muda no grupo, e todos mudam junto.

Onde ver as permissões no pgAdmin

1. Clique com o botão direito na tabela → **Properties…**
2. Abra a aba **Security**
3. A lista mostra **quem** tem **qual** privilégio
4. Dá para conceder e revogar por aqui também, sem escrever SQL

> Use a tela para **conferir**. Para **configurar**, prefira o SQL — ele fica no arquivo `.sql` do grupo, versionado e repetível.

Checagem do Bloco 2

Escreva os comandos:

- Criar a role `estagiario`, que pode conectar
- Deixar ela **só ler** a tabela `jogo`
- Tirar dela a permissão de ler `cliente`



`forms.gle/12EK4Mz5XAwF4mLZA`

Intervalo

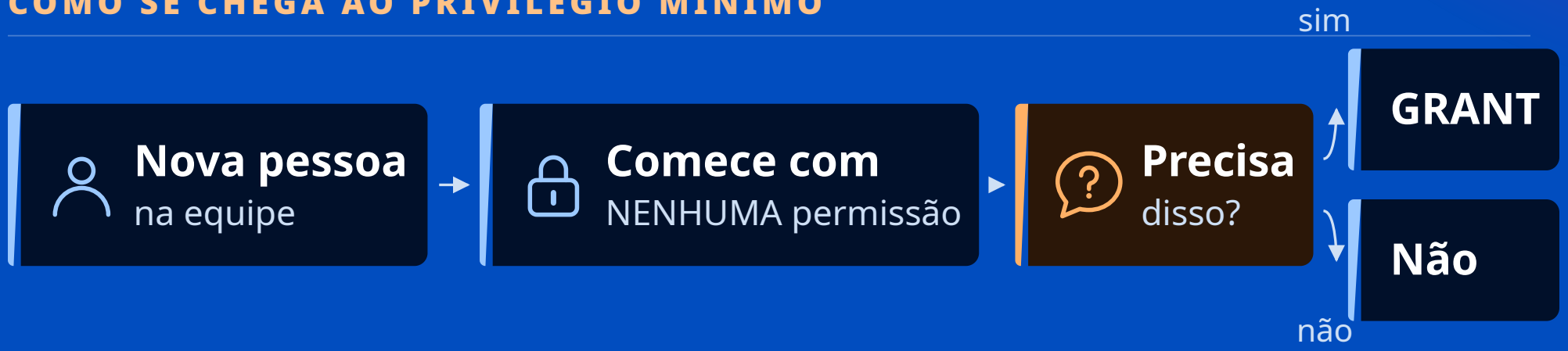
10 minutos

Bloco 3

Privilégio mínimo, views e dados pessoais

O princípio do privilégio mínimo

COMO SE CHEGA AO PRIVILÉGIO MÍNIMO



Volte à pergunta para cada tabela e cada verbo — a permissão se concede uma de cada vez.

Comece com **nenhuma** permissão e conceda só o que a pessoa precisa para trabalhar. Nunca o contrário.

Desenhando as permissões do negócio



Agora a matriz do seu negócio

Antes de escrever `GRANT`, responda: **quem** precisa fazer **o quê** em **qual** tabela?

- Três papéis do **seu** minimundo, um por linha
- As tabelas do seu modelo, uma por coluna
- Em cada cruzamento, o verbo — `SELECT`, `INSERT`, `UPDATE`, `DELETE` — ou nada

> Permissão que você não consegue justificar em voz alta é permissão que não se concede.

View como proteção

```
CREATE VIEW catalogo_publico AS
SELECT titulo, genero, ano_lancamento FROM jogo;

GRANT SELECT ON catalogo_publico TO estagiario;
-- e NENHUM grant na tabela jogo
```

O estagiário consulta o catálogo e **nunca** enxerga o preço de custo, nem o dado de cliente nenhum. É o **nível de visões** da Aula 02 — prometido lá, entregue aqui.

Dados pessoais — não é só boa prática

- Nome, telefone, CPF e e-mail de cliente são **dados pessoais**
- A **LGPD** (Lei 13.709/2018) exige que o acesso seja **limitado ao necessário**
- Vazamento gera multa para a empresa e dano para a pessoa

> Privilégio mínimo deixou de ser "caprichoso" e virou **exigência legal**. Quem trabalha com banco de dados responde por isso.

⚠ Erro comum — testar a permissão com o próprio `postgres`

> Conceder `GRANT` e continuar conectado como `postgres` não prova nada: o superusuário passa por cima de qualquer restrição. **É preciso desconectar e conectar como a role nova.**

No pgAdmin: crie uma **segunda conexão de servidor**, com o usuário da role e a senha dela. Assim dá para alternar entre as duas.

Checagem do Bloco 3

O estagiário precisa ver **título e gênero** dos jogos. Nada mais.

- Escreva os comandos: a view e o `GRANT`
- Ele consegue rodar `SELECT preco_diaria FROM jogo`? Por quê?
- Como você **prova** que ele não consegue?



forms.gle/S4xyAMfKpKd68QHZ7

Intervalo

10 minutos

MÃO NA MASSA

Permissões no banco do projeto

1. No papel, desenhem a **matriz**: quem precisa fazer o quê em qual tabela
2. Criem uma role `atendente` (ou equivalente do tema de vocês)
3. Concedam `CONNECT`, `USAGE` e só os privilégios necessários
4. Criem uma **view** sem dados pessoais e uma role que só enxerga ela
5. Abram uma **segunda conexão** no pgAdmin com a role nova
6. **Testem:** o que ela deve poder, funciona? O que ela não deve, é recusado?
7. Anotem a mensagem de erro do acesso negado

MÃO NA MASSA

Lembrete de entrega

GLOSSÁRIO

`grupoNN-aulaNN-artefato.<ext>`	`grupo03-aula16-permissoes.sql`
	`grupo03-aula16-matriz.pdf`

No `.sql`, incluam um **comentário** com a mensagem de erro que apareceu quando a role tentou o que não podia. É a prova de que funcionou.

MÃO NA MASSA

Enquanto vocês configuram

O que eu vou conferir:

- A matriz foi desenhada **antes** dos comandos?
- `CONNECT` e `USAGE` foram concedidos? (o esquecimento nº 1)
- Os privilégios são os **mínimos** necessários, ou foi `ALL PRIVILEGES`?
- Existe uma **view** protegendo dado pessoal?
- Vocês **conectaram** com a role nova e **viram** o acesso negado?

Terminou? Usem `REVOKE` para tirar uma permissão e confirmem que a role perdeu o acesso.

MÃO NA MASSA

Devolutiva — passo de mesa em mesa

Eu passo em cada grupo. Deixem na tela:

- A role restrita fazendo o que **pode**
- A mesma role sendo **recusada** no que não pode, com a mensagem lida em voz alta

> A recusa é a evidência do indicador. Todo grupo precisa me mostrar as duas coisas, mesmo que rápido.

PARA CASA · OPCIONAL

Antes da próxima aula

Testar em casa: crie uma role no seu banco pessoal que só possa ler uma tabela. Conecte com ela e tente um `INSERT`.

Tarefa opcional: revisem a matriz de permissões do grupo e perguntem, para cada `GRANT`: *"alguém consegue trabalhar sem isso?"*. Se a resposta for sim, façam `REVOKE`.

Para treinar: `locadora-exercicios-16-permissoes.sql`, na página da aula — testa tudo com `SET ROLE`, sem precisar de outra conexão.

Para ver acontecer: demonstrações da Aula 16 — entrar como cada pessoa da locadora e tentar as mesmas quatro coisas — e oito perguntas com resposta na hora. No site da UC, na página da aula.

Para a próxima aula

As permissões protegem contra quem **não deveria** mexer. **E contra o disco que queima? E contra o `DELETE` sem `WHERE` do administrador?**

UniSenac Centro Universitário RS
Senac
Fecomércio
IFEP
Sesc
Sindicatos Empresariais