# Permissões em banco de dados Postgresql

Para conceder permissões específicas de acesso (como `INSERT`, `UPDATE`, `DELETE` e `SELECT`) a um usuário em um banco de dados PostgreSQL, você pode usar o comando `GRANT`.

Aqui está o passo a passo:

**Acesse o PostgreSQL**: Primeiro, acesse o banco de dados PostgreSQL com um usuário que tenha privilégios suficientes para conceder permissões:

```
psql -U seu_usuario -d nome_do_banco
```

**Conceder permissões**: Use o comando `GRANT` para definir as permissões de acordo com as operações que você quer permitir. Aqui estão exemplos para conceder permissões em uma tabela específica:

```
-- Concede permissão de SELECT (consulta) na tabela 'nome_tabela'
GRANT SELECT ON nome_tabela TO nome_usuario;

-- Concede permissão de INSERT (inserção de dados) na tabela
'nome_tabela'
GRANT INSERT ON nome_tabela TO nome_usuario;

-- Concede permissão de UPDATE (atualização de dados) na tabela
'nome_tabela'
GRANT UPDATE ON nome_tabela TO nome_usuario;

-- Concede permissão de DELETE (exclusão de dados) na tabela
'nome_tabela'
GRANT DELETE ON nome_tabela TO nome_usuario;
```

Se você quiser conceder todas essas permissões de uma vez, você pode usar:

```
GRANT SELECT, INSERT, UPDATE, DELETE ON nome_tabela TO nome_usuario;
```

**Verificar as permissões** (opcional): Para verificar as permissões que um usuário tem em uma tabela, você pode usar a consulta:

```
\dp nome_tabela
```

Essas permissões podem ser aplicadas tanto em tabelas individuais quanto em várias tabelas, dependendo do seu cenário.

No PostgreSQL é possível criar grupos de usuários usando o conceito de *roles* (papéis). Um *role* pode ter permissões específicas, e você pode associar usuários a essa *role* para que herdem suas permissões.

**Criar um Grupo de Permissões** (ou *Role*): Primeiro, crie uma *role* que servirá como grupo de permissões. Por exemplo, vamos criar uma *role* chamada `grupo_usuarios`:

```
CREATE ROLE grupo_usuarios;
```

**Definir Permissões para o Grupo**: Conceda as permissões desejadas ao grupo de usuários na(s) tabela(s) específica(s). Aqui, concedemos as permissões `SELECT`, `INSERT`, `UPDATE`, e `DELETE` em uma tabela chamada `nome_tabela`:

```
GRANT SELECT, INSERT, UPDATE, DELETE ON nome_tabela TO grupo_usuarios;
```

**Criar Usuários e Associá-los ao Grupo**: Agora, crie os usuários e associe-os ao grupo usando o comando `GRANT` para que herdem as permissões do grupo.
Exemplo:

```
-- Criando um usuário
CREATE USER usuario1 WITH PASSWORD 'senha_usuario1';
CREATE USER usuario2 WITH PASSWORD 'senha_usuario2';

-- Associando os usuários ao grupo
GRANT grupo_usuarios TO usuario1;
GRANT grupo_usuarios TO usuario2;
```

*Verificar usuarios no pgAdmin*

```
SELECT rolname, rolsuper, rolinherit, rolcreaterole, rolcreatedb,
rolcanlogin FROM pg_roles;
```

**Verificar Herança de Permissões**: Por padrão, os usuários herdam as permissões dos grupos aos quais pertencem. Caso você queira que o usuário execute apenas as permissões que herda dos grupos, sem permissões adicionais diretas, não é necessário definir permissões adicionais para o usuário.

**Revisar Permissões**: Para verificar se o grupo e os usuários têm as permissões desejadas, use o comando `\dp nome_tabela` no PostgreSQL para listar as permissões na tabela.

Com essa estrutura, basta adicionar novos usuários ao grupo `grupo_usuarios`, e eles automaticamente herdarão todas as permissões configuradas para esse grupo.

## Exercícios de Permissões no PostgreSQL

### 1. Criação e Configuração Básica de Usuários e *Roles*

1. Crie um usuário chamado `usuario_leitura` com permissão apenas para ler (SELECT) dados de uma tabela chamada `clientes`.
2. Crie um usuário chamado `usuario_escrita` com permissões de leitura e escrita (SELECT, INSERT, UPDATE) na tabela `clientes`.
3. Crie uma *role* chamada `grupo_vendas` e conceda a ela permissões de leitura e escrita nas tabelas `produtos` e `pedidos`.

4. Associe os usuários `usuario_leitura` e `usuario_escrita` ao grupo `grupo_vendas` e verifique se ambos herdaram as permissões concedidas ao grupo.

**2. Praticando Permissões Específicas em Tabelas**

1. Conceda ao usuário `usuario_analista` apenas permissão de leitura (SELECT) na tabela `relatorios`.
2. Crie uma tabela chamada `financeiro` e conceda ao grupo `grupo_financeiro` permissão para ler e atualizar dados, mas não para inserir ou excluir registros.
3. Remova a permissão de atualização (UPDATE) da tabela `clientes` para o usuário `usuario_escrita` e verifique se ele ainda consegue atualizar registros.
4. Conceda permissões de inserção e atualização (INSERT e UPDATE) a um usuário chamado `usuario_editor` apenas em colunas específicas de uma tabela chamada `inventario`.

**3. Exercícios com Schemas e Controle de Acesso Amplo**

1. Crie um novo esquema chamado `financeiro` e conceda permissão ao grupo `grupo_financeiro` para criar e gerenciar tabelas dentro dele.
2. Dê permissão de leitura em todas as tabelas do schema `public` a um usuário chamado `usuario_consulta`.
3. Crie um usuário `admin_local` e conceda a ele permissões de administração completas (como `CREATE`, `DROP`, e `ALTER`) apenas no schema `local`.

**4. Testes e Revisão de Permissões**

1. Use o comando `\dp` no `psql` para revisar todas as permissões da tabela `clientes` e verifique as permissões de cada usuário e grupo.
2. Revogue a permissão de deleção (DELETE) na tabela `produtos` de todos os usuários que a possuem e tente realizar uma exclusão com um desses usuários para confirmar que a permissão foi removida.
3. Experimente remover um usuário do grupo `grupo_vendas` e verifique se ele perde as permissões herdadas do grupo.
4. Faça uma consulta para verificar todos os usuários e suas permissões em todas as tabelas, usando o `information_schema.role_table_grants`.