

Tutorial — Seus primeiros passos no pgAdmin

Aula 01 · Projeto de Banco de Dados · PostgreSQL + pgAdmin 4

Ao final deste tutorial você terá o pgAdmin aberto, um banco criado, as tabelas montadas e as primeiras consultas rodando. É a base de tudo o que vem no semestre, e o ambiente que você vai usar até dezembro.

Você pode seguir estes passos durante a aula e refazê-los em casa quantas vezes quiser. Se faltou ao encontro, dá para acompanhar tudo por aqui, do começo ao fim.

1. Abrindo o pgAdmin

O pgAdmin já está instalado na máquina do laboratório. Procure por **pgAdmin 4** no menu e abra.

A abertura demora alguns segundos na primeira vez, e a janela pode ficar em branco por um instante antes de a interface aparecer. É normal: ele está iniciando os componentes internos. Não clique várias vezes no ícone achando que não abriu, senão você acaba com duas cópias abertas.

A senha mestra

Na primeira abertura ele pede uma **senha mestra** (*master password*). Ela não é a senha do banco: é só para o pgAdmin guardar as senhas dos servidores que você cadastrar.

Escolha algo que você lembre e **anote**. Se esquecer, vai precisar recadastrar tudo depois.

No laboratório, se a máquina for compartilhada e o pgAdmin já tiver senha mestra de outra turma, chame o professor em vez de tentar adivinhar.

2. Conectando no servidor

Do lado esquerdo há uma árvore. Se já existir um servidor listado ali — normalmente chamado **PostgreSQL 16** ou parecido —, clique nele e pule para o passo 3. Ele vai pedir a senha do usuário `postgres`.

Se não existir nenhum servidor, cadastre:

1. clique com o botão direito em **Servers** → **Register** → **Server**;
2. na aba **General**, campo **Name**: escreva `Local` (é só um apelido, escolha o que quiser);
3. na aba **Connection**, preencha:

Campo	Valor
Host name/address	localhost
Port	5432
Maintenance database	postgres
Username	postgres
Password	a senha definida na instalação

4. marque **Save password** para não digitar toda vez;
5. clique em **Save**.

Se der erro de conexão

A mensagem costuma ser *"could not connect to server"*. Ela significa que o pgAdmin abriu, mas não encontrou nenhum PostgreSQL rodando — são dois programas diferentes, e o pgAdmin sozinho não é um banco de dados, é só o painel que conversa com um.

Chame o professor. Não perca a aula tentando resolver sozinho.

3. Criando o seu banco

Com o servidor conectado, a árvore abre e mostra **Databases**.

1. clique com o botão direito em **Databases** → **Create** → **Database...**;
2. em **Database**, escreva `biblioteca`;
3. clique em **Save**.

Ele aparece na árvore. Pronto: você acabou de fazer, pela interface, o que o comando abaixo faz por SQL. Os dois caminhos levam ao mesmo lugar, e daqui em diante vamos usar o segundo.

```
CREATE DATABASE biblioteca;
```

Use um banco só para a aula. Não crie tabelas dentro do banco `postgres`, que é o banco de manutenção do servidor. Misturar seus testes com ele dá confusão depois.

4. A janela onde você escreve SQL

Clique no banco `biblioteca` para selecioná-lo e abra a ferramenta de consulta:

- pelo menu: **Tools** → **Query Tool**;
- ou pelo atalho: botão direito no banco → **Query Tool**;
- ou pelo teclado: **Alt** + **Shift** + **Q**.

A janela tem três partes: o editor em cima, os resultados embaixo, e a barra de botões no topo.

Confira em qual banco você está. O nome aparece na aba da janela. Escrever SQL na janela errada e não entender por que a tabela "sumiu" é o tropeço mais comum do primeiro dia.

Executando

O que fazer	Como
Executar tudo o que está no editor	F5 ou o botão ►
Executar só um trecho	selecione o trecho com o mouse e aperte F5
Limpar o editor	Ctrl + Shift + K

O segundo é o mais útil da lista. Você vai escrever vários comandos na mesma janela; selecionar só o que quer rodar evita reexecutar tudo desde o começo a cada teste.

Seu primeiro comando

Digite e execute:

```
SELECT version();
```

A resposta traz a versão do PostgreSQL que está rodando na sua máquina. Anote esse número: na aula 02 você vai instalar um servidor numa máquina virtual e comparar.

5. Carregando o arquivo da aula

Se você se perdeu no meio da aula, chegou atrasado ou quer refazer em casa, existe o arquivo `banco-aula01.sql`, que cria tudo de uma vez.

Na janela de consulta:

1. clique no ícone de **pasta** na barra de botões (ou **Ctrl** + **O**);
2. escolha o arquivo `banco-aula01.sql`;
3. o conteúdo aparece no editor;
4. aperte **F5**.

O último comando do arquivo é uma conferência. Se tudo correu bem, ele devolve:

autores	livros	livros_sem_isbn	autores_sem_livro
6	7	1	1

Aqueles dois últimos números são de propósito, e as consultas da aula dependem deles.

Só carregue o arquivo se precisar. Digitar os comandos é lento e é exatamente por isso que funciona: você aprende a escrever SQL escrevendo SQL, não colando.

6. Agora o SQL

Daqui em diante, siga o **roteiro da aula**, que traz os comandos comentados um a um, da criação da tabela até `GROUP BY`. Este tutorial cobriu só o ambiente.

Duas coisas que vão acontecer e são normais:

Erro em vermelho. Leia a mensagem antes de chamar alguém: o PostgreSQL costuma dizer exatamente o que está errado e em que posição. `relation "autor" does not exist` significa que a tabela não foi criada, ou que você está no banco errado.

Comando que não devolve linhas. `CREATE TABLE` e `INSERT` não mostram tabela de resultado, só uma mensagem de sucesso na aba **Messages**. Não é erro.

7. Salvando o seu trabalho

Isto é o que você entrega, então faça desde o começo.

Mantenha uma janela de consulta com tudo o que você escreveu, na ordem, e salve num arquivo:

1. `Ctrl + S` na janela de consulta;
2. nome do arquivo: `aula01-seusobrenome.sql` ;
3. salve na sua pasta, não na Área de Trabalho da máquina do laboratório.

Vá salvando de tempos em tempos, não só no fim. Máquina de laboratório reinicia, trava e é usada por outra turma depois de você.

Leve o arquivo com você. Pendrive, e-mail para você mesmo, Drive, o que preferir. Na aula 02 você vai rodar esse mesmo script dentro de uma máquina virtual Linux, e ele precisa existir.

8. Se a sua máquina só tiver SQLite

Plano B, caso o PostgreSQL não conecte e não dê para resolver na hora. O SQLite não precisa de servidor: o banco inteiro é um arquivo.

No terminal:

```
sqlite3 biblioteca.db
```

Você entra num prompt onde digita SQL direto. Para sair, `.quit`.

A maior parte do roteiro funciona igual, mas quatro coisas mudam, e vale saber quais:

No PostgreSQL	No SQLite
<code>serial</code>	<code>INTEGER PRIMARY KEY AUTOINCREMENT</code>
<code>text</code> , <code>varchar(n)</code>	<code>TEXT</code> — o limite não é imposto
<code>numeric(10,2)</code>	não existe tipo decimal exato
Chave estrangeira sempre verificada	desligada por padrão

A última é a que mais importa para esta aula. Para o SQLite recusar um `INSERT` que aponta para registro inexistente, você precisa ligar a verificação **em toda conexão**:

```
PRAGMA foreign_keys = ON;
```

Sem essa linha, a demonstração da parte 4 do roteiro simplesmente não acontece: o dado errado entra sem reclamação nenhuma.

Isso não é defeito, é escolha de projeto: o SQLite foi feito para ser pequeno e embarcado, e por muito tempo nem tinha chave estrangeira. A comparação entre os dois volta na **aula 05**, quando estudarmos como cada um guarda os dados em disco.

Antes de sair da sala

- ☐ o banco `biblioteca` existe e você consegue abrir a janela de consulta nele
- ☐ as tabelas `autor` e `livro` foram criadas
- ☐ você viu pelo menos um erro de restrição acontecer, de propósito
- ☐ o desafio está respondido, ou o que deu tempo dele
- ☐ o arquivo `aula01-seusobrenome.sql` está salvo **e enviado no Classroom**
- ☐ você tem uma cópia do arquivo fora da máquina do laboratório