Acesso a banco de dados

Programação para Internet I
IFB - Tecnologia em Sistemas para Internet

Marx Gomes van der Linden

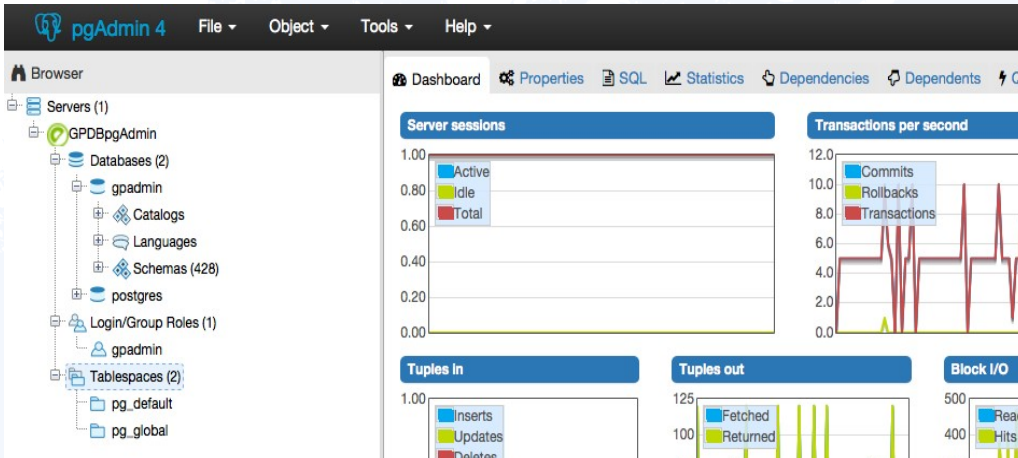
SQL

- *Structured Query Language*
- Linguagem-padrão para manipulação de informações em bancos de dados.
  - Definições de dados
  - Consultas
  - Modificações, Inserções, Exclusões
  - Controle de Acesso

Qual sistema utilizar?

- O Node.js tem suporte a praticamente todos os sistemas de bancos de dados relacionais em uso atualmente:
  - MS SQL Server
  - MySQL
  - Oracle
  - PostgreSQL
  - SQLite
  - etc..

- Software livre
- Disponível para Windows, Linux, macOS, entre outros
- Implementação quase completa de todos os padrões SQL
- Utilitário gráfico: pgAdmin



Arquitetura do nosso sistema

Cliente (Front end)

Servidor (Back end)

Banco de dados

Acessando pelo Node.js

- É possível acessar qualquer sistema de banco de dados por meio de uma aplicação Node.js
- Para acessar um banco PostgreSQL, usa-se o

```
npm install pg --save
```

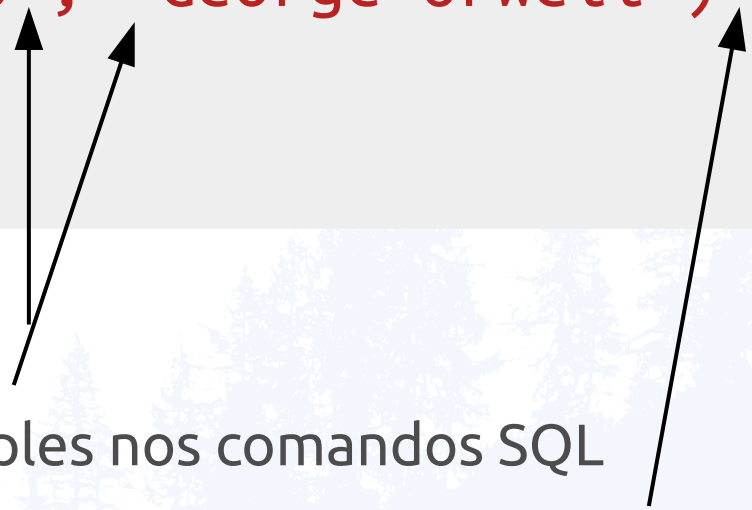
Configurando e conectando

```
const pg = require('pg');

const client = new pg.Client(
   {
        user: nome_do_usuario,
        host: 'localhost',
        database: banco_de_dados,
        password: senha,
       port: 5432,
   }
);

client.connect();
```

Método `query(...)`

- Para enviar comandos SQL ao banco de dados basta usar o método `query` no objeto client:

```
client.query(
"INSERT INTO livro (titulo, autor) VALUES
('A Revolução Dos Bichos', 'George Orwell')"
);
```

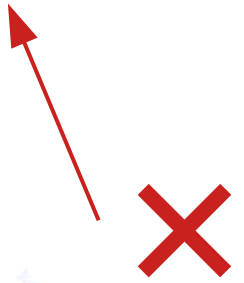


- Atenção às aspas!
  - » PostgreSQL requer aspas simples nos comandos SQL
  - » É mais fácil usar aspas duplas nas strings do próprio JavaScript

Utilizando parâmetros

- Não é recomendável montar comandos SQL manualmente com concatenação ou interpolação de variáveis:

```
let titulo = 'A Revolução Dos Bichos';
let nomeDoAutor = 'George Orwell';

client.query("INSERT INTO livro (titulo, autor)
VALUES (${titulo}, '${nomeDoAutor}')")
```



- Não fazer isso! Falha de segurança!

Utilizando parâmetros (maneira correta)

```
let nomeDoAutor = 'George Orwell';

client.query(
  {
    text: "INSERT INTO livro (titulo, autor) VALUES ($1, $2)",
    values: ['A Revolução Dos Bichos', nomeDoAutor]
  }
);
```

Obtendo dados da tabela (comando SELECT)

- Para obter ler o retorno de um comando SELECT, use o método then após o query:

```
client.query('SELECT * FROM livro')
.then(
    function(ret){
        console.log(ret.rows);
    }
);
```

Array de objetos

Obtendo dados da tabela (comando SELECT)

```
client.query('SELECT * FROM livro')
.then(
    function(ret){
        for(linha of ret.rows){
            console.log("Título:", linha.titulo);
            console.log("Autor:", linha.autor);
            console.log("Número de páginas:", linha.npags);
            console.log();
        }
    }
);
```

Construindo uma API

- Para construir uma API de acesso a banco de dados, basta combinar o `express` com o `pg`
- A API deve retornar como JSON os dados lidos do banco de dados através de um comenado SQL

Estrutura básica

```
const express = require("express");
const bodyParser = require("body-parser");
const cors = require("cors");

const pg = require('pg');
const app = express();

app.use(cors());
app.use(bodyParser.json());

const client = new pg.Client({
    //...
});

client.connect();

//...

app.listen(3000, () => {
    console.log('Servidor web funcionando');
}) ;
```

Gerando JSON a partir de uma consulta

- A maneira mais simples de gerar o JSON é diretamente a partir de **`ret.rows`**:

```
app.get("/livros", function(req, res) {
    client.query('SELECT * FROM livro')
        .then(
            function (ret) {
                res.json(ret.rows);
            }
        )
});
```

Personalizando o resultado

```
app.get("/livros", function(req, res) {
    client.query('SELECT * FROM livro')
        .then(
            function (ret) {
                let array = [];

                for(let livro of ret.rows){
                    array.push({
                            nomeDoAutor: livro.autor,
                            titulo: livro.titulo

                        }
                    )
                }

                res.json({
                    status: 'OK',
                    numeroDeResultados: array.length,
                    resultados: array
                });
            }
        )
});
```

```
app.get('/livro/:id', function(req, res) {
    client.query(
        {
            text: 'SELECT * FROM livro WHERE id = $1',
            values: [req.params.id]
        }
    )
        .then(
            function (ret) {
                console.log(req.params.id, ret.rows);
                let livro = ret.rows[0];
                res.json(
                    {
                        autor: livro.autor,
                        titulo: livro.titulo,
                        numeroDePaginas: livro.npags
                    }
                );
            }
        )
});
```

Usando parâmetros

```
app.post('/livro', function(req, res) {
  client.query(
    {
     text:
      'INSERT INTO livro (autor, titulo, npags) VALUES
($1, $2, $3)',
     values:
       [req.body.autor, req.body.titulo,
req.body.numeroDePaginas]
    }
  )
        .then(
            function (ret) {
                res.json(
                    {
                        status: 'OK',
                        dadosEnviados: req.body
                    }
                )
            }
        );
});
```

Inserindo dados