PUC Minas

Plataforma Node.js

Acesso a Bancos de Dados

Prof. Rommel Vieira Carneiro

Acesso a Bancos de Dados com Node.js

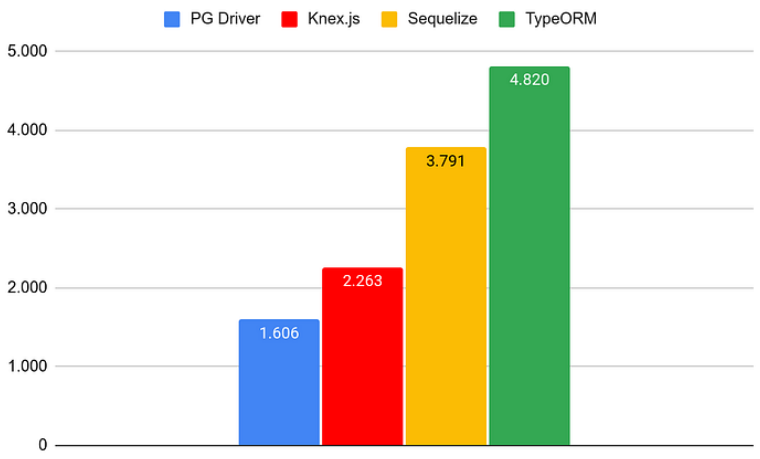
Alternativas para acesso a Bancos de Dados

- Drivers Nativos (PG, MySql, etc)
- Object-Relational Mapping (ORM) frameworks
- Object Data Modeling (ODM) frameworks
- SQL Query Builder

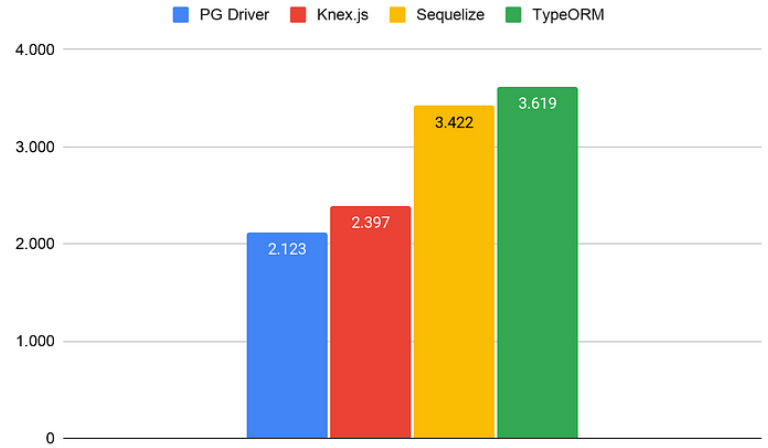
Frameworks & Bibliotecas para Node.js

Comparativo de Alternativas para DBs Relacionais

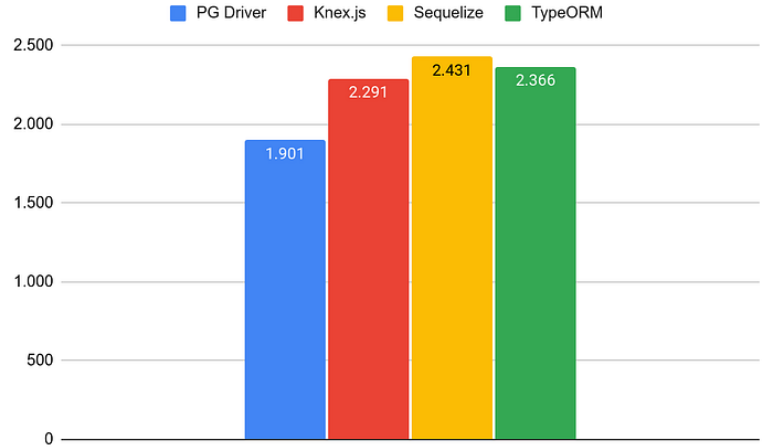
Driver Nativo (pg) x Knex.js x Sequelize x TypeORM



300 inserts



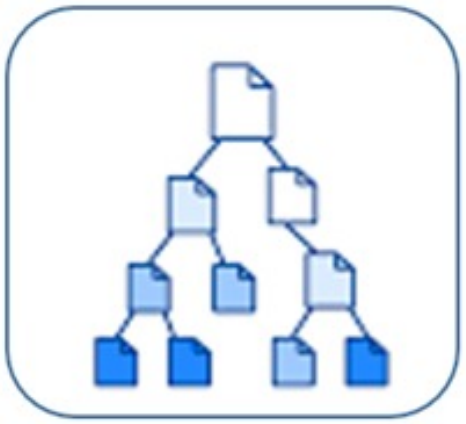
300 updates



300 deletes

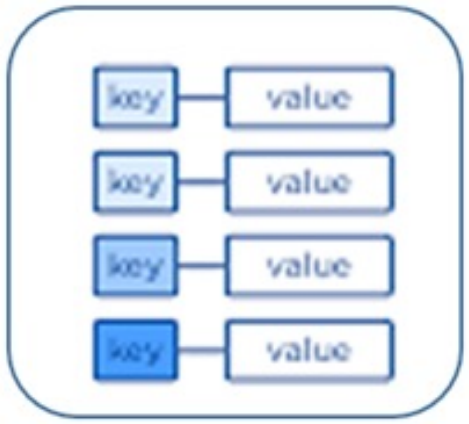
Fonte: PG Driver vs Knex.js vs Sequelize vs TypeORM (Welington Fidelis)

Tipos de NoSQL Databases



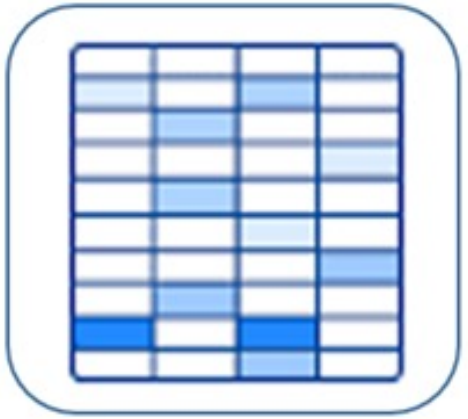
DOCUMENT STORE

Dados e metadados armazenados hierarquicamente em documentos baseados em JSON



KEY VALUE STORE

Dados representados por uma coleção de pares chave-valor (o mais simples)



WIDE-COLUMN STORE

Dados relacionados são armazenados com um conjunto de pares de chave-valor aninhados em uma única coluna.

GRAPH STORE

Dados são armazenados em uma estrutura de grafo (nodos, arestas e propriedades)

Fonte: Relational vs. NoSQL data (Microsoft Learn)

Bancos Relacionais *vs* Bancos NoSQL

Bancos de Dados Relacionais	x	Bancos de Dados NoSQL
Modelo estruturado, organizado em tabelas	**Estrutura de Dados**	Flexíveis na estrutura de dados com modelos de dados diferentes
Escalam verticalmente, adicionando mais poder de processamento ao servidor	**Escalabilidade**	Escalam horizontalmente, usando mais servidores para lidar com o aumento da carga
Propriedades ACID (Atomicidade, Consistência, Isolamento, Durabilidade), garantem confiabilidade em transações	**Transações ACID**	Nem todos suportam propriedades ACID. Alguns oferecem consistência eventual
SQL (Structured Query Language) para consultas	**Consultas**	Variedade de linguagens de consulta, dependendo do sistema específico
Representar relações entre dados.	**Relacionamentos**	Relacionamentos são difíceis de representar em alguns bancos
A consistência dos dados é crucial (sistemas bancários, reservas)	**Uso**	Escalabilidade e a flexibilidade são mais importantes do que a consistência estrita (redes sociais ou big data)

Bancos Relacionais *vs* Bancos NoSQL

Casos de uso típicos

Bancos de Dados Relacionais	Bancos de Dados NoSQL
• Content management systems (CMS)	• Gestão de Logística e Ativos
• Customer relationship management (CRM)	• Gerenciamento de inventário e catálogo
• Enterprise resource planning (ERP)	• Gerenciamento digital e de mídia
• Data warehouses	• Personalização, recomendações e experiência do cliente
• Análises Financeiras	• Internet das coisas (IoT) e dados de sensores
• Identity management	• Serviços financeiros e pagamentos
• Dados Geoespaciais	• Detecção de fraude e autenticação de identidade
• Catálogos de Product	• Sistemas de mensagens

Fonte: NoSQL Vs Relational Databases (Boltic)

Knex Query Builder

- SQL Query Builder para Node.js
- Compatível com diversos bancos de dados
  → PostgreSQL, SQL Server, MySQL, SQLite3, Oracle
- Abordagem via Promises e Callbacks
- Oferece os seguintes recursos
  - Controle de transações
  - Pool de Conexões
  - Streaming Queries
  - Pacote de testes
- Referências
  - Knexjs Org
  - Knex CheatSheet

Knex – Instalação e inicialização

Utilize o npm para instalação dos pacotes para o Knex e o npx para executar a inicialização do Knex que cria o arquivo **`knexfile.js`**

```
# Instalação do pacote do Knex no projeto
$ npm install knex --save

# Instalação da biblioteca de Banco de Dados
$ npm install pg
$ npm install mysql
$ npm install sqlite3
$ npm install oracledb

# Inicializa knexfile.js (arquivo de configuração)
$ npx knex init
```

Knex – Instalação e inicialização

Execute o comando de inicialização do Knex: **`npx knex init`**

knexfile.js

```
module.exports = {
  development: {
      client: 'sqlite3',
      connection: { filename: './db.sqlite3' }
  },

  production: {
      client: 'pg',
      connection: {
         database: 'db',
         user: 'user',
         password: 'pass'
      },
      pool: { min: 2, max: 10 }
  }
};
```

Knex – Conexão com o BD

Configuração de conexão via objeto de parâmetros

```
const knex = require('knex')({
      client: 'mysql',
      connection: {
          host : 'servidor.com',
          port : 3306,
          user : 'user',
          password : 'senha',
          database : 'database'
      },
      pool: { min: 0, max: 7 }
  });
```

Knex – Conexão com o BD

Configuração de conexão via string de conexão (literal e variável de ambiente)

```
const knex = require ('knex') ({
    client: 'pg',
    connection: 'postgres://user:senha@servidor:5432/database'
});
```

```
const knex = require ('knex') ({
    client: 'pg',
    connection: process.env.DATABASE_URL
});
```

Knex – Comandos básicos – SELECT

Selecionar dados a partir de uma tabela

```
...

  knex
     .select ('*')
     .from ('produto')
     .where ( { marca: 'Helmanns' } )
      .limit (10)

...
```

Knex – Comandos básicos – SELECT

Selecionar dados com filtro (where) a partir de uma tabela e devolver resultado em uma API com middleware Express.

```
...

app.get (endpoint + 'produtos', function (req, res) {
    knex
      .select ('*')
      .from ('produto')
      .where ( { marca: 'Helmanns' } )
      .then ( produtos => res.status(200).json (produtos) )
})

...
```

Knex – Comandos básicos – INSERT

Inserção de múltiplos registros com retorno dos IDs gerados automaticamente

```
...

 knex('users')
  .insert([
    { name: 'Starsky' },
    { name: 'Hutch' }
  ], ['id'])

...
```

Knex – Comandos básicos – UPDATE

Alteração de registro

```
...

knex('users')
  .where({ id: 2 })
  .update({ name: 'Homer' })

...
```

Knex – Comandos básicos – DELETE

Exclusão de registros

```
...

knex('users')
  .where({ id: 2 })
  .del()

...
```

Knex – Migrations

Processo de evolução do banco de dados

- Evolutionary Database Design (Martin Fowler)
- Migração de banco de dados (Google)
- What Are Database Migrations? | Database Migrations in Node
- Knex Migrations

Knex – Migrations


```
# Cria uma migração
npx knex migrate:make migration_name
npx knex migrate:make migration_name --env production

# Executa scripts de migração
npx knex migrate:latest [--env production] # Executa até mais novo
npx knex migrate:up                        # Executa próximo script

# Desfaz scripts de migração
npx knex migrate:rollback [--env production] # Desfaz tudo
npx knex migrate:down                        # Desfaz último script

# Lista migrações já executadas
npx knex migrate:list
```

Knex – Migrations – Configuração

Configurações para especificar o diretório das migrações e a tabela de controle de migrações no banco de dados.

knexfile.js

```
module.exports = {
  development: {
    client: 'sqlite3',
    connection: { filename: './db.sqlite3' } ,
    migrations: {
      directory: './db/migrations',   // Diretório para migrações
      tableName: 'knex_migrations'    // Tabela de controle de migrações
    },
    seeds: {
      directory: './db/seeds',        // Diretório para arquivos de carga
    }
  },
};
```

Knex – Migrations

Exemplo de arquivo de migração com rotina up e down

```
exports.up = function(knex) {
  return knex.schema.createTable("produtos", tbl => {
    tbl.increments ('id') ;
    tbl.text ("descricao", 255).unique ().notNullable();
    tbl.text ("marca", 128).notNullable();
    tbl.decimal ("valor").notNullable();
  });
};

exports.down = function(knex) {
  return knex.schema.dropTableIfExists ("produtos")
};
```

Knex – Migrations


```
exports.up = function(knex) {
  return knex.schema.createTable("usuarios", tbl => {
    tbl.increments ('id');
    tbl.text ("nome", 255).unique().notNullable();
    tbl.text ("login", 100).unique().notNullable();
    tbl.text ("email", 100).notNullable();
    tbl.text ("senha", 100).notNullable();
    tbl.text ("roles", 200).notNullable();
  });
};

exports.down = function(knex) {
  return knex.schema.dropTableIfExists ("usuarios")
};
```

Knex – Migrations (Seed CLI)

Seed é uma API do Knex para realizar carga no banco de dados.

```
# Gera arquivo para carga de dados → Cria pasta seeds
npx knex seed:make seed_name

# Executa carga de dados (todos ou arquivo específico)
npx knex seed:run
npx knex seed:run --specific=filename.js
```

Knex – Migrations (Seed CLI)

Exemplo de arquivo de carga (seed)

```
exports.seed = async function(knex) {
  // Deletes ALL existing entries
  await knex('produtos').del()
  await knex('produtos').insert([
    {id: 1, descricao: 'Cream Cracker', marca: 'Aymmoré', valor: 3.5},
    {id: 2, descricao: 'Cerveja', marca: 'Spaten', valor: 10.99},
    {id: 3, descricao: 'Filé Mignon', marca: 'Friboi', valor: 78.99},
    {id: 4, descricao: 'Refrigerante', marca: 'Coca-Cola', valor: 15.50}
  ]);
};
```

Obrigado!

Prof. Rommel Vieira Carneiro