Instituto Federal Catarinense
Campus Sombrio
Curso Técnico em Informática para Internet

TRABALHO DE DESENVOLVIMENTO WEB 3

Desvendando o Mundo dos ORMs com Sequelize

Maria Clara Flores Steckert

Turma: 3ª série B

Sombrio

Julho, 2025.

SUMÁRIO

INTRODUÇÃO

No desenvolvimento de aplicações modernas, a comunicação entre código e banco de dados é um aspecto fundamental. Para facilitar essa integração, surgiram os ORMs — ferramentas que automatizam a tradução entre objetos na programação e tabelas em bancos relacionais. Este trabalho tem como foco explorar esse conceito por meio do **Sequelize**, um dos ORMs mais populares para Node.js.

2. Fundamentos dos ORMs

O que é um ORM?

ORM (Object-Relational Mapping) é uma técnica que permite interagir com bancos de dados relacionais utilizando objetos da linguagem de programação, em vez de comandos SQL puros.

O problema da Impedância Objeto-Relacional

O **Object-Relational Impedance Mismatch** refere-se às diferenças conceituais entre os paradigmas de orientação a objetos (usados na programação) e o modelo relacional (usado nos bancos de dados). ORMs tentam reduzir essa fricção.

Como funciona um ORM?

```
flowchart LR
App["App: Usuario.findAll()"] --> ORM["ORM Sequelize"]
ORM --> SQL["SELECT * FROM Usuarios;"]
SQL --> DB["Banco de Dados"]
DB --> ORM
ORM --> App["App recebe dados em objeto"]
```

O ORM atua como uma camada de **abstração**, convertendo objetos em consultas SQL e resultados em objetos JS.

3. Sequelize em Detalhes

O que é o Sequelize?

O Sequelize é um ORM baseado em Promises para Node.js, que suporta diversos bancos como PostgreSQL, MySQL, MariaDB, SQLite e MSSQL. Ele permite definir modelos de dados e interagir com o banco usando JavaScript.

Models e Associações

Exemplo de Model:

```
// models/produto.js
```

```
module.exports = (sequelize, DataTypes) => {
  const Produto = sequelize.define('Produto', {
    id: {
      type: DataTypes.INTEGER,
      primaryKey: true,
      autoIncrement: true,
    },
    nome: DataTypes.STRING,
    preco: DataTypes.DECIMAL,
  });
  return Produto;
};
```

Tipos de associação:

- hasOne: Um-para-Um
- hasMany: Um-para-Muitos
- belongsTo: Pertence a outro
- belongsToMany: Muitos-para-Muitos

```
// Exemplo hasMany
Usuario.hasMany(Post);
Post.belongsTo(Usuario);
```

Consultas com Sequelize x SQL

Operação	SQL	Sequelize
Buscar todos	SELECT * FROM Usuarios	Usuario.findAll()
Buscar por ID	SELECT * FROM Usuarios WHERE id=1	Usuario.findByPk(1)
Condição WHERE	SELECT * FROM Usuarios WHERE ativo=1	Usuario.findAll({ where: { ativo: 1 } })
JOIN	SELECT * FROM Usuarios u JOIN Posts p ON u.id = p.userId	Usuario.findAll({ include: Post })

4. Tópicos Avançados e Boas Práticas

Migrations

sequelize.sync({ force: true }) **apaga e recria as tabelas**, o que em produção pode causar perda de dados.

Migrations permitem versionar a estrutura do banco:

npx sequelize-cli migration:generate --name create-produto

npx sequelize-cli db:migrate

npx sequelize-cli db:migrate:undo

Transações

Transações são blocos atômicos de operações — tudo ocorre ou nada é salvo. Isso garante integridade.

```
const t = await sequelize.transaction();

try {
  await Produto.create({ nome: 'Arado', preco: 100 }, { transaction: t });
  await Estoque.decrement('quantidade', { where: { produtoId: 1 }, transaction: t });
  await t.commit();
} catch (err) {
  await t.rollback();
}
```

5. Análise Crítica e Comparativa

Vantagens do ORM

1. **Produtividade:** Menos código SQL e mais foco na lógica.

2. **Independência do banco:** Facilita trocar o SGDB sem reescrever tudo.
3. **Validações e segurança embutidas.**

Desvantagens

1. **Performance:** ORMs podem gerar queries ineficientes.
2. **Curva de aprendizado.**
3. **Abstração que esconde o controle fino do SQL.**

Quando NÃO usar ORM?

- Aplicações que exigem performance extrema.
- Projetos onde o controle sobre as queries é essencial.
- Bancos não-relacionais.

Comparativo: Sequelize vs Knex.js

Critério	**Sequelize**	**Knex.js**
Tipo	ORM	Query Builder
Linguagem	JavaScript	JavaScript
Definição de Schema	Via modelos (classes)	Via código SQL-like
Facilidade de uso	Alta para iniciantes	Mais técnico

CONCLUSÃO

O Sequelize oferece uma poderosa abstração para lidar com bancos relacionais usando JavaScript. Como todo ORM, traz vantagens em produtividade e organização, mas também exige atenção à performance e estrutura de dados. Compreender seu funcionamento é essencial para desenvolvedores que buscam equilíbrio entre facilidade e controle.

REFERÊNCIAS BIBLIOGRÁFICAS

Documentação oficial do Sequelize: https://sequelize.org

Artigo sobre ORM:

https://www.redhat.com/en/topics/data/what-is-an-object-relational-mapping-orm

MDN Web Docs

YouTube: Programador BR, Código Fonte TV

Stack Overflow