Ministério da Educação
Secretaria de Educação Profissional e Tecnológica
Instituto Federal Catarinense - Campus Sombrio
3° Ano de Informática para Internet

ORESTE EDUARDO MACHADO DOS SANTOS

Mapeamento Objeto-Relacional (ORM) com Sequelize: Fundamentos, Aplicações e Comparativos

Sombrio
2025

Sumário

1. Introdução

No desenvolvimento de aplicações modernas, a persistência de dados é uma das camadas mais críticas. O Mapeamento Objeto-Relacional (ORM) surge como uma abordagem para facilitar a comunicação entre o paradigma orientado a objetos e os bancos de dados relacionais. Este trabalho visa apresentar os conceitos fundamentais de ORM, explorando a biblioteca Sequelize — amplamente utilizada no ecossistema Node.js — e fornecendo uma análise crítica sobre suas vantagens, limitações e comparações com outras ferramentas.

2. Fundamentos dos ORMs

2.1 O QUE É UM ORM?

Um ORM (Object-Relational Mapping) é uma técnica que permite que objetos da programação sejam mapeados diretamente para tabelas de um banco de dados relacional. Ele fornece uma camada de abstração, permitindo manipular dados do banco usando a linguagem do código (como JavaScript), sem escrever diretamente comandos SQL.

Paradigma da Impedância Objeto-Relacional: Esse paradigma se refere às dificuldades e incompatibilidades entre os conceitos da programação orientada a objetos (classes, herança, encapsulamento) e os bancos de dados relacionais (tabelas, colunas, chaves). O ORM busca mitigar essas diferenças. Além disso, existe também o registro ativo : cada linha em uma tabela SQL é considerada uma instância da classe Cliente e o mapeador de dados : separa completamente o seu domínio da camada de persistência. Isso significa que nenhum dos seus objetos de domínio sabe nada sobre o banco de dados. Em vez disso, precisamos usar um serviço completamente diferente, conhecido como Gerenciador de Entidades, para persistir a instância no banco de dados.

2.2 COMO FUNCIONA UM ORM?

Quando utilizamos um ORM, por exemplo Usuário.findAll(), o código é traduzido para uma consulta SQL (SELECT * FROM usuários;) que será executada no banco de dados. Após a execução, os resultados retornam como objetos manipuláveis no código.

Diagrama Fluxograma:

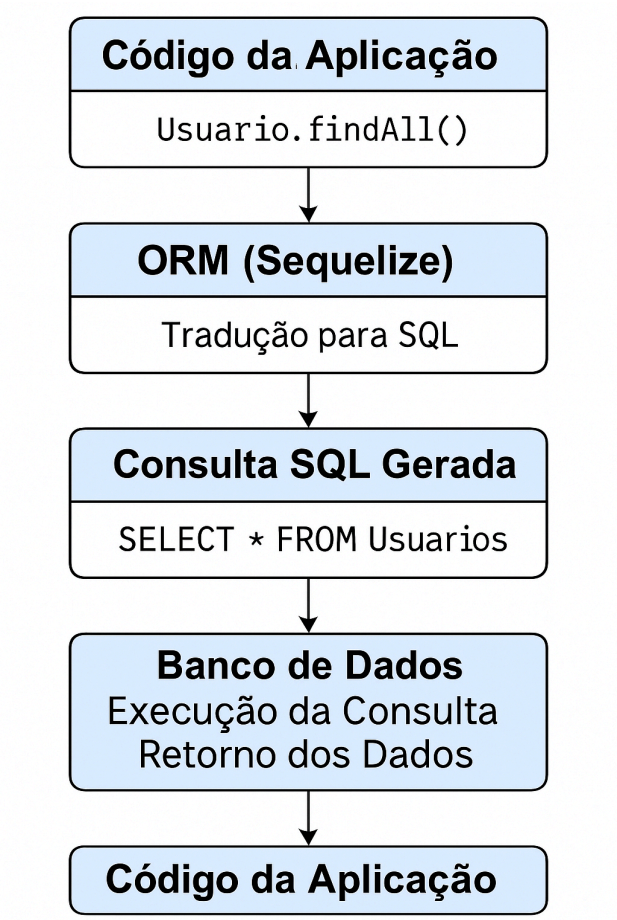

Imagem 1 - Fluxograma de funcionamento de ORM

3. Sequelize em Detalhes

3.1 O QUE É O SEQUELIZE?

Sequelize é um ORM baseado em JavaScript que facilita a interação com bancos de dados relacionais como PostgreSQL, MySQL, SQLite e MSSQL. Ele permite definir modelos, realizar associações e executar queries usando JS moderno.

Dialetos Suportados:

- PostgreSQL
- MySQL

- MariaDB
- SQLite
- Microsoft SQL Server

3.2 QUERIES: COMPARAÇÃO SQL VS. SEQUELIZE

Operação	SQL	Sequelize
Buscar por todos	SELECT * FROM usuarios;	Usuário.findAll()
Buscar por ID	SELECT * FROM usuarios WHERE id = 1;	usuário.findByPk(1)
Com condição	SELECT * FROM usuarios WHERE idade > 18;	Usuario.findAll({ where: { idade: { [Op.gt]: 18 } } })
Join	SELECT * FROM produtos INNER JOIN categorias ON ...	Produto.findAll({ include: Categoria })

4. Tópicos Avançados e Boas Práticas

4.1 MIGRATIONS

O comando sequelize.sync({ force: true }) apaga todas as tabelas e recria, o que é perigoso na produção.
Migrations resolvem isso criando arquivos que versionam mudanças no banco.
Fluxo:

1. Criar: sequelize migration:generate --name criar-produtos
2. Aplicar: sequelize db:migrate
3. Reverter: sequelize db:migrate:undo

4.2 TRANSAÇÕES

Transações garantem que múltiplas operações sejam atômicas — ou todas ocorrem, ou nenhuma.

5. Análise Crítica e Comparativa

5.1 Vantagens e Desvantagens

Vantagens:

- Abstração do SQL, maior produtividade.
- Portabilidade entre bancos diferentes.
- Estruturação e organização dos dados com associações.

Desvantagens:

- Curva de aprendizado.
- Pode gerar queries ineficientes.
- Dificuldade em consultas complexas.

5.2 Quando NÃO usar ORM?

- Aplicações com altíssima performance exigida.
- Projetos que exigem queries SQL extremamente específicas.
- Ambientes de banco são muito complexos.

5.3 COMPARATIVO: SEQUELIZE VS. PRISMA

Critério	Sequelize	Prisma
Tipo	ORM	ORM
Linguagem Principal	JavaScript	TypeScript
Definição de Schema	Código JS	Arquivo Prisma (.prisma)
Facilidade	Alta curva inicial, flexível	Interface amigável, documentação robusta

6. Conclusão

O ORM é uma solução eficaz para a integração entre objetos do código e tabelas de banco de dados. Sequelize, como um dos ORMs mais populares para Node.js, oferece flexibilidade, produtividade e suporte a múltiplos bancos. No entanto, sua adoção deve ser cuidadosamente avaliada com base nos requisitos do projeto.

7. Referências

SEQUELIZE. *Sequelize ORM Documentation*. Disponível em: https://sequelize.org. Acesso em: 8 jul. 2025.

POSTGRESQL GLOBAL DEVELOPMENT GROUP. *PostgreSQL: The World's Most Advanced Open Source Relational Database*. Disponível em: https://www.postgresql.org/docs/. Acesso em: 8 jul. 2025.

FOWLER, Martin. *Object-Relational Impedance Mismatch*. Disponível em: https://martinfowler.com/bliki/OrmHate.html. Acesso em: 8 jul. 2025.

MEDIUM. *Comparing Prisma and Sequelize: Key Differences for Node.js ORM*. Medium, 2024. Disponível em: https://medium.com/search?q=prisma%20vs%20sequelize. Acesso em: 8 jul. 2025.