Programação Web Back-end

Banco de dados relacionais

Adriano Rivolli

rivolli@utfpr.edu.br

Universidade Tecnológica Federal do Paraná (UTFPR)
Câmpus Cornélio Procópio
Departamento de Computação

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Conteúdo

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Sequelize

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Sequelize

- https://sequelize.org/
- ORM para Postgres, MySQL, MariaDB, SQLite e Microsoft SQL Server
- Baseado em **Promises**
- Suporta transações, relações, carregamento de dados *lazy*, replicações de leitura

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Instalação

- Instalação:
  `npm install sequelize`
- Driver do banco:
  - `npm install pg pg-hstore`
  - `npm install mysql2`
  - `npm install mariadb`
  - `npm install sqlite3`
  - `npm install tedious` *Microsoft SQL Server*

Conexão com o banco de dados

```
const Sequelize = require("sequelize")
```

- Opção 1

```
const sequelize = new
Sequelize("postgres://user:pass@host:5432/db")
```

- Opção 2

```
const sequelize = new Sequelize("database", "username",
"password", {host: "localhost", dialect: "opção"})
```

- Dialetos: **mysql**, **mariadb**, **postgres**, **mssql**

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Testando a conexão

```
try {
  await sequelize.authenticate();
  console.log('Connection has been established successfully.');
} catch (error) {
  console.error('Unable to connect to the database:', error);
}
```

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Definição do modelo

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Modelo

- Uma abstração que representa uma tabela do banco de dados
- Define o nome da tabela, as colunas e seus tipos
- Sintaxe:
  `sequelize.define(modelName, attributes, options)`

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Exemplo de criação de um modelo

```
const { Sequelize, DataTypes } = require('sequelize');
const sequelize = new Sequelize('sqlite::memory:');

const User = sequelize.define('User', {
  // Model attributes are defined here
  firstName: {
    type: DataTypes.STRING,
    allowNull: false
  },
  lastName: {
    type: DataTypes.STRING
    // allowNull defaults to true
  }
}, {
  // Other model options go here
});
```

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Tipos de dados

- String `DataTypes.STRING`, `DataTypes.STRING(1234)`, `DataTypes.TEXT`, ...
- Booleano `DataTypes.BOOLEAN`
- Números `DataTypes.INTEGER`, `DataTypes.FLOAT`, `DataTypes.DOUBLE`, `DataTypes.DECIMAL(10, 2)`, ...
- Data `DataTypes.DATE` e `DataTypes.DATEONLY`

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Opções de definição das colunas

- type
- allowNull
- defaultValue
- unique
- primaryKey
- autoIncrement
- comment

Chave estrangeira

- 1:N

```
Model.belongsTo(OtherModel)
```

Irá criar uma chave estrangeira na tabela ***Model***

```
Model.hasMany(OtherModel)
```

Irá criar uma chave estrangeira na tabela ***OtherModel***

- N:N

```
Model.belongsToMany(OtherModel, { through:  "ModelOtherModel"})
OtherModel.belongsToMany(Model, { through:  "ModelOtherModel"})
```

Irá criar uma nova tabela ***MotherOtherModel*** *com ambas as chaves*

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Sincronização

- `Model.sync()`

  Cria a tabela se ela não existe

- `Model.sync({ force: true })`

  Cria a tabela, ee a tabela existe apaga primeiro

- `Model.sync({ alter: true })`

  Se necessário realiza alterações na tabela para estar em conformidade com o modelo

- `sequelize.sync()`

  Sincroniza todos os modelos registrados

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

CRUD

Inserção (1 registro)

- Método **salvar**

```
const obj = Model.build({ col1:  "val1", ...  })

await obj.save()
```

- Método **create**

```
const obj = await Model.create({ col1:  "val1", ...  })
```

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Inserção (múltiplos objetos)

- O método **bulkCreate** é similar ao método **create** porém recebe múltiplos objetos

```
const objs = await Model.bulkCreate([
 { col1:  "val11", col2:  "val21", ...  },
 { col1:  "val12", col2:  "val22", ...  },
 ...  ])
```

Alteração (1 objeto)

- Método **salvar**

```
obj.col1 = "abc"
obj.set({col2:  "v2", col3:  "v3"})

await obj.save()
```

- Método **update**

```
await obj.update({ col1:  "v1"})
```

Não salva outras modificações do objeto

- Descartar alterações

```
await obj.reload()
```

Inserção (múltiplos objetos)

- É possível realizar uma alteração sem carregar o objeto préviamente

```
await Model.update({ col1:  "val1", col2:  "val2"}, {
 where:  { ...  }

 })
```

Exclusão

- 1 objeto:

```
await obj.destroy()
```

- Múltiplos objetos:

```
await Model.destroy({
 where:  { ...  }

 })
```

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Consultas

- Buscar pela chave primária

```
const obj = await Model.findByPk(id)
```

- Listar todos os registros

```
const list = await Model.findAll()
```

- Condições simples

```
await Model.findOne({where: { cod: 123 }})
await Model.findAll({where: { uf: "PR"}})
```

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Operadores

```
const { Op } = require("sequelize");
Post.findAll({
  where: {
    [Op.and]: [{ a: 5 }, { b: 6 }],           // (a = 5) AND (b = 6)
    [Op.or]: [{ a: 5 }, { b: 6 }],            // (a = 5) OR (b = 6)
    someAttribute: {
      // Basics
      [Op.eq]: 3,                             // = 3
      [Op.ne]: 20,                            // != 20
      [Op.is]: null,                          // IS NULL
      [Op.not]: true,                         // IS NOT TRUE
      [Op.or]: [5, 6],                        // (someAttribute = 5) OR (someAttribute = 6)

      // Number comparisons
      [Op.gt]: 6,                             // > 6
      [Op.gte]: 6,                            // >= 6
      [Op.lt]: 10,                            // < 10
      [Op.lte]: 10,                           // <= 10
      [Op.between]: [6, 10],                  // BETWEEN 6 AND 10
      [Op.notBetween]: [11, 15],              // NOT BETWEEN 11 AND 15

      [Op.in]: [1, 2],                        // IN [1, 2]
      [Op.notIn]: [1, 2],                     // NOT IN [1, 2]

      [Op.like]: '%hat',                      // LIKE '%hat'
      [Op.notLike]: '%hat',                   // NOT LIKE '%hat'
      [Op.startsWith]: 'hat',                 // LIKE 'hat%'
      [Op.endsWith]: 'hat',                   // LIKE '%hat'
      [Op.substring]: 'hat',                  // LIKE '%hat%'
      [Op.iLike]: '%hat',                     // ILIKE '%hat' (case insensitive) (PG only)
      [Op.notILike]: '%hat',                  // NOT ILIKE '%hat'  (PG only)
      [Op.regexp]: '^[h|a|t]',                // REGEXP/~ '^[h|a|t]' (MySQL/PG only)
      [Op.notRegexp]: '^[h|a|t]',             // NOT REGEXP/!~ '^[h|a|t]' (MySQL/PG only)
      [Op.iRegexp]: '^[h|a|t]',               // ~* '^[h|a|t]' (PG only)
      [Op.notIRegexp]: '^[h|a|t]',            // !~* '^[h|a|t]' (PG only)

    }
  }
});
```

Joins

- Acessando a informação (sem carregar previamente)

```
let autor = await livro.getAutor()
```

- Realizando um único Join

```
let obj = await LivroModel.findAll({include: { model:
AutorModel, as: 'Escritor' } })

console.log(obj.Escritor)
```

- Realizando múltiplos Join

```
await LivroModel.findAll({include: [AutorModel, EditoraModel] })
```

- Condições na junção

```
await LivroModel.findAll({include: { model: AutorModel, where:
{...} } })
```

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Exemplo prático