Desenvolvimento de Serviços e APIs

Faculdade Senac Pelotas

Escola de Tecnologia da Informação

Prof. Edécio Fernando Iepsen

#Relações muitos-para-muitos

Link direto para relacionamentos muitos-para-muitos

As associações Muitos-para-Muitos conectam uma origem com vários destinos, enquanto todos esses destinos podem, por sua vez, ser conectados a outras origens além da primeira.

Isso não pode ser representado pela adição de uma chave estrangeira a uma das tabelas, como os outros relacionamentos fizeram. Em vez disso, o conceito de um modelo de junção é usado. Este será um modelo extra (e uma tabela extra no banco de dados) que terá duas colunas de chave estrangeira e manterá o controle das associações. Às vezes, a tabela de junção também é chamada de *tabela de junção* ou *através de tabela* .

Para este exemplo, vamos considerar os modelos `Movie` e `Actor`. Um ator pode ter participado de muitos filmes e um filme teve muitos atores envolvidos em sua produção. Será chamada a tabela de junção que fará o acompanhamento das associações `ActorMovies`, que conterá as chaves estrangeiras `movieId` e `actorId`.

#Implementação

Link direto para Implementação

A principal maneira de fazer isso no Sequelize é a seguinte:

```
const Movie = sequelize.define('Movie', { name: DataTypes.STRING });
const Actor = sequelize.define('Actor', { name: DataTypes.STRING });
Movie.belongsToMany(Actor, { through: 'ActorMovies' });
Actor.belongsToMany(Movie, { through: 'ActorMovies' });
```

Como uma string foi fornecida na `through` opção de `belongsToMany` chamada, o Sequelize criará automaticamente o `ActorMovies` modelo que atuará como o modelo de junção. Por exemplo, no PostgreSQL:

```
CREATE TABLE IF NOT EXISTS "ActorMovies" (
  "createdAt" TIMESTAMP WITH TIME ZONE NOT NULL,
  "updatedAt" TIMESTAMP WITH TIME ZONE NOT NULL,
  "MovieId" INTEGER REFERENCES "Movies" ("id") ON DELETE CASCADE ON UPDATE CASCADE,
  "ActorId" INTEGER REFERENCES "Actors" ("id") ON DELETE CASCADE ON UPDATE CASCADE,
  PRIMARY KEY ("MovieId","ActorId")
);
```

Em vez de uma string, também há suporte para passar um modelo diretamente e, nesse caso, o modelo fornecido será usado como o modelo de junção (e nenhum modelo será criado automaticamente). Por exemplo:

```
const Movie = sequelize.define('Movie', { name: DataTypes.STRING });
const Actor = sequelize.define('Actor', { name: DataTypes.STRING });
const ActorMovies = sequelize.define('ActorMovies', {
  MovieId: {
    type: DataTypes.INTEGER,
    references: {
      model: Movie, // 'Movies' would also work
      key: 'id'
    }
  },
  ActorId: {
    type: DataTypes.INTEGER,
    references: {
      model: Actor, // 'Actors' would also work
      key: 'id'
    }
  }
});
Movie.belongsToMany(Actor, { through: ActorMovies });
Actor.belongsToMany(Movie, { through: ActorMovies });
```

Revenda AVENIDA

AVENIDA é uma famosa revenda de veículos.

Dr. Edecio, fundador e Diretor Geral da empresa, solicitou um sistema que permita cadastrar **Carros**, **Clientes**, **Usuários** e **Propostas**.

Obs.:

- Os usuários (**admins**) podem cadastrar os **carros**.
- Os **clientes** podem fazer **propostas** para os carros.