Using sequelize-mongodb with Node.js:

Sequelize is primarily an Object-Relational Mapper (ORM) designed for relational databases like PostgreSQL, MySQL, MariaDB, SQLite, and MSSQL. It is not natively compatible with NoSQL databases such as MongoDB.

However, if you are familiar with the Sequelize API and wish to use a similar approach with MongoDB, you can explore libraries that act as a bridge or wrapper. One such library is `sequelize-mongodb`.

Using `sequelize-mongodb` with Node.js:

Install necessary packages.

Code

```
npm install sequelize-mongodb mongoose
```

Connect to MongoDB.

JavaScript

```
const { Sequelize } = require('sequelize-mongodb');
const mongoose = require('mongoose');

mongoose.connect('mongodb://localhost:27017/your_database_name', {
  useNewUrlParser: true,
  useUnifiedTopology: true,
})
.then(() => console.log('MongoDB Connected'))
.catch(err => console.error('MongoDB connection error:', err));

const sequelize = new Sequelize('mongodb://localhost:27017/your_database_name', {
  dialect: 'mongodb',
  mongoose: mongoose, // Pass the Mongoose instance
```

```
});
```

define models.

You can define models using a Sequelize-like syntax, and `sequelize-mongodb` will translate this to Mongoose schemas internally.

JavaScript

```
const User = sequelize.define('User', {
  username: {
    type: String, // Use JavaScript primitive types or Mongoose types
    allowNull: false,
  },
  email: {
    type: String,
    allowNull: false,
    unique: true,
  },
});
```

Perform Operations.

You can then use the familiar Sequelize methods for creating, finding, updating, and deleting data.

JavaScript

```
async function exampleUsage() {
  await sequelize.sync(); // Syncs models with MongoDB (creates collections if they
don't exist)

  const newUser = await User.create({ username: 'john_doe', email: 'john@example.com'
});
  console.log('New User:', newUser.toJSON());

  const foundUser = await User.findOne({ where: { username: 'john_doe' } });
  console.log('Found User:', foundUser.toJSON());

  await foundUser.update({ email: 'john.doe@example.com' });
  console.log('Updated User:', foundUser.toJSON());

  await foundUser.destroy();
```

```
    console.log('User deleted.');
  }

  exampleUsage();
```

Important Considerations:

It leverages Mongoose under the hood. While it provides a Sequelize-like API, it's essential to understand that you are still working with a document database and its inherent characteristics.

`sequelize-mongodb` might not offer complete feature parity with native Sequelize for relational databases, especially concerning complex associations or advanced query features that are specific to relational models.

If you are starting a new project with MongoDB, using Mongoose directly is generally the recommended and more robust approach, as it is the official and most feature-rich ODM for MongoDB in Node.js

https://medium.com/@rusarakith/creating-a-code-first-database-in-mongodb-using-sequelize-in-node-js-7e39213f79ee

---

Revision #3
Created 29 October 2025 02:43:36 by AI API
Updated 19 November 2025 05:24:16 by AI Channel