Introduction
Getting Started
Core Concepts
Model Basics
Model Instances
Model Querying - Basics
Model Querying - Finders
Getters, Setters & Virtuals
Validations & Constraints
Raw Queries
Associations
Paranoid
Advanced Association Concepts
Eager Loading
Creating with Associations
Advanced M:N Associations
Association Scopes
Polymorphic Associations
Other topics
Other Data Types
Using sequelize in AWS Lambda
Connection Pool

Version: v6 - stable

Creating with Associations

BelongsTo / HasMany / HasOne association
BelongsTo association with an alias
HasMany / BelongsToMany association

An instance can be created with nested association in one step, provided all elements are new.

In contrast, performing updates and deletions involving nested objects is currently not possible. For that, you will have to perform each separate action explicitly.

BelongsTo / HasMany / HasOne association

Consider the following models:

```
class Product extends Model {}
Product.init({
  title: Sequelize.STRING
}, { sequelize, modelName: 'product' });
class User extends Model {}
User.init({
  firstName: Sequelize.STRING,
  lastName: Sequelize.STRING
}, { sequelize, modelName: 'user' });
class Address extends Model {}
Address.init({
  type: DataTypes.STRING,
  line1: Sequelize.STRING,
  line2: Sequelize.STRING,
  city: Sequelize.STRING,
  state: Sequelize.STRING,
  zip: Sequelize.STRING,
}, { sequelize, modelName: 'address' });

// We save the return values of the association setup calls to use them later
Product.User = Product.belongsTo(User);
User.Addresses = User.hasMany(Address);
// Also works for `hasOne`
```

A new `Product`, `User`, and one or more `Address` can be created in one step in the following way:

```
return Product.create({
  title: 'Chair',
  user: {
    firstName: 'Mick',
    lastName: 'Broadstone',
    addresses: [{
      type: 'home',
      line1: '100 Main St.',
      city: 'Austin',
      state: 'TX',
      zip: '78704'
    }]
  }
}, {
  include: [{
    association: Product.User,
    include: [ User.Addresses ]
  }]
});
```

Observe the usage of the `include` option in the `Product.create` call. That is necessary for Sequelize to understand what you are trying to create along with the association.

Note: here, our user model is called `user`, with a lowercase `u` - This means that the property in the object should also be `user`. If the name given to `sequelize.define` was `User`, the key in the object should also be `User`. Likewise for `addresses`, except it's pluralized being a `hasMany` association.

BelongsTo association with an alias

The previous example can be extended to support an association alias.

```
const Creator = Product.belongsTo(User, { as: 'creator' });

return Product.create({
  title: 'Chair',
  creator: {
    firstName: 'Matt',
    lastName: 'Hansen'
  }
}, {
  include: [ Creator ]
});
```

HasMany / BelongsToMany association

Let's introduce the ability to associate a product with many tags. Setting up the models could look like:

```
class Tag extends Model {}
Tag.init({
  name: Sequelize.STRING
}, { sequelize, modelName: 'tag' });

Product.hasMany(Tag);
// Also works for `belongsToMany`.
```

Now we can create a product with multiple tags in the following way:

```
Product.create({
  id: 1,
  title: 'Chair',
  tags: [
    { name: 'Alpha'},
    { name: 'Beta'}
  ]
}, {
  include: [ Tag ]
})
```

And, we can modify this example to support an alias as well:

```
const Categories = Product.hasMany(Tag, { as: 'categories' });

Product.create({
  id: 1,
  title: 'Chair',
  categories: [
    { id: 1, name: 'Alpha' },
    { id: 2, name: 'Beta' }
  ]
}, {
  include: [{
    association: Categories,
    as: 'categories'
  }]
})
```

Edit this page

*Last updated on **Oct 13, 2022** by **Rik Smale***

Previous
« Eager Loading

Next
Advanced M:N Associations »