

Get started

Open in app



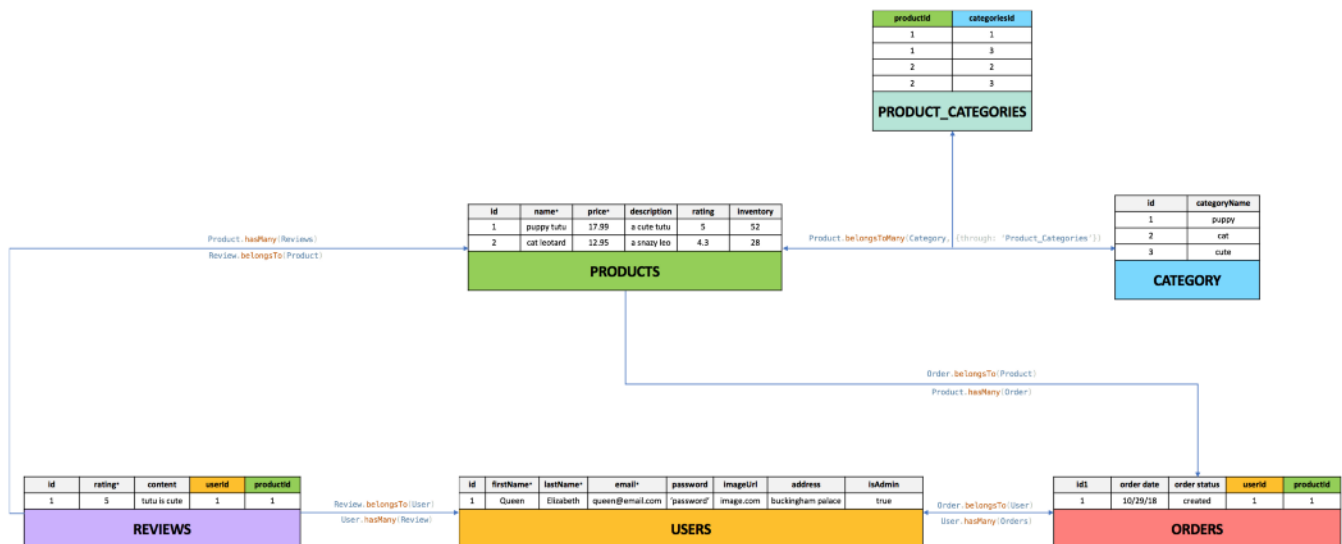
## Julianne Crawford

53 Followers · About Follow

# Sequelize Associations & Magic Methods



Julianne Crawford Dec 1, 2018 · 3 min read



## Objective

The objective of this article is to provide more understandable (visual) documentation on Sequelize associations and magic methods. If you're new to Sequelize and/or relational databases, take a look at these articles ([here](#) and [here](#)) for some background information.

## Motivation

[Get started](#)[Open in app](#)

pages of deterring and dense documentation. When I was first learning Sequelize, I struggled to understand model associations and their respective “magic methods”.

## Associations & Magic Methods

An association in Sequelize is a relationship between two models (a source and target model). Creating an association generally involves adding a foreign key to one model which creates a reference to the other. There are several Sequelize associations, including *hasOne*, *belongsTo*, *hasMany*, and *belongsToMany*. In terms of implementation, the key difference between these associations is in which model (the source or the target) the foreign key resides. I’ve made a series of visualization tools (see below) to help differentiate between these associations.

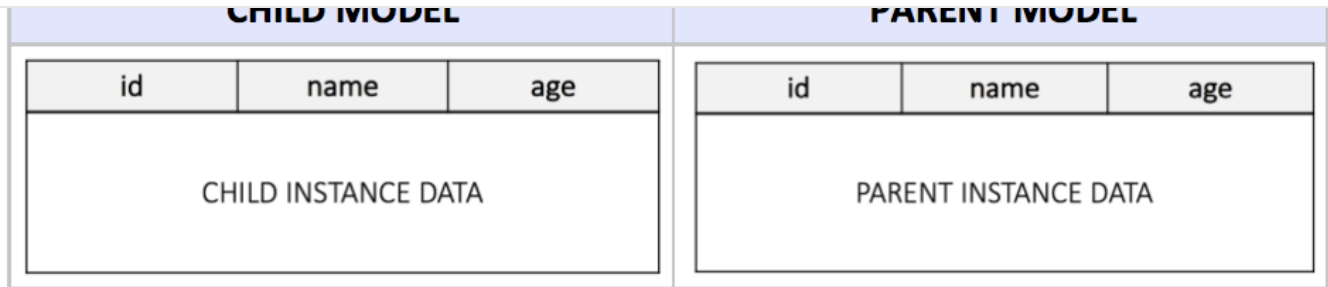
“Magic methods” are simply methods that are generated by defining Sequelize associations. As their name implies, the capabilities they invoke are quite magical. The visualizations below also outline which magic methods a given association will create. Alternatively, another way to easily see what magic methods a model has access to, is to use:

```
Object.keys(ModelName.prototype)
```

## Starting Point

### Base Models

For the purpose of this demonstration, I will begin with two base models — a parent and a child model. The tables below show how the 4 Sequelize associations mentioned above affect these base models, and what magic methods are created as a result.

[Get started](#)[Open in app](#)

## One-To-One

SYNTAX	CHILD MODEL	PARENT MODEL	MAGIC METHODS														
<p>SOURCE MODEL foreign key from target model is added to source model</p> <p>TARGET MODEL</p> <p><code>Child.belongsTo(Parent)</code></p>	<table><tr><th>id</th><th>name</th><th>age</th><th>parentId</th></tr><tr><td colspan="4">CHILD INSTANCE DATA</td></tr></table>	id	name	age	parentId	CHILD INSTANCE DATA				<table><tr><th>id</th><th>name</th><th>age</th></tr><tr><td colspan="3">PARENT INSTANCE DATA</td></tr></table>	id	name	age	PARENT INSTANCE DATA			<pre>child.getParent() child.setParent(parent) child.createParent(parent)</pre>
id	name	age	parentId														
CHILD INSTANCE DATA																	
id	name	age															
PARENT INSTANCE DATA																	
<p>SOURCE MODEL foreign key from target model is added to source model</p> <p>TARGET MODEL</p> <p>adds <b>favoriteParentId</b> to the source model rather than <b>parentId</b></p> <p><code>Child.belongsTo(Parent, {as: 'favoriteParent'})</code></p> <p>AS optional: where as has been defined the following string will be used in place of the target model name</p>	<table><tr><th>id</th><th>name</th><th>age</th><th>favoriteParentId</th></tr><tr><td colspan="4">CHILD INSTANCE DATA</td></tr></table>	id	name	age	favoriteParentId	CHILD INSTANCE DATA				<table><tr><th>id</th><th>name</th><th>age</th></tr><tr><td colspan="3">PARENT INSTANCE DATA</td></tr></table>	id	name	age	PARENT INSTANCE DATA			<pre>child.getFavoriteParent() child.setFavoriteParent(parent) child.createFavoriteParent(parent)</pre>
id	name	age	favoriteParentId														
CHILD INSTANCE DATA																	
id	name	age															
PARENT INSTANCE DATA																	
<p>SOURCE MODEL foreign key from target model is added to source model</p> <p>TARGET MODEL</p> <p>uses <b>favoriteParentId</b> for the accessor methods rather than <b>parentId</b></p> <p>adds <b>foreignId</b> to the target model rather than <b>parentId</b></p> <p><code>Child.belongsTo(Parent, {as: 'favoriteParent', foreignKey: 'foreignId'})</code></p> <p>AS optional: where as has been defined the following string will be used in place of the target model name</p> <p>FOREIGN-KEY optional: this option overwrites the default foreign key <b>parentId</b> to the following string</p>	<table><tr><th>id</th><th>name</th><th>age</th><th>foreignId</th></tr><tr><td colspan="4">CHILD INSTANCE DATA</td></tr></table>	id	name	age	foreignId	CHILD INSTANCE DATA				<table><tr><th>id</th><th>name</th><th>age</th></tr><tr><td colspan="3">PARENT INSTANCE DATA</td></tr></table>	id	name	age	PARENT INSTANCE DATA			<pre>child.getFavoriteParent() child.setFavoriteParent(parent) child.createFavoriteParent(parent)</pre>
id	name	age	foreignId														
CHILD INSTANCE DATA																	
id	name	age															
PARENT INSTANCE DATA																	

### One-to-One Association: Belongs To

## Has One

Associations where the foreign key exists on the **target model**

<p>SOURCE MODEL</p> <p>TARGET MODEL foreign key from source model is added to target model</p> <p>↓</p> <p>↓</p> <p><code>Parent.hasOne(Child)</code></p>	<table><tr><th>id</th><th>name</th><th>age</th><th>parentId</th></tr><tr><td colspan="4">CHILD INSTANCE DATA</td></tr></table>	id	name	age	parentId	CHILD INSTANCE DATA				<table><tr><th>id</th><th>name</th><th>age</th></tr><tr><td colspan="3">PARENT INSTANCE DATA</td></tr></table>	id	name	age	PARENT INSTANCE DATA			<pre>parent.getChild() parent.setChild(child) parent.createChild(child)</pre>
id	name	age	parentId														
CHILD INSTANCE DATA																	
id	name	age															
PARENT INSTANCE DATA																	
<p>SOURCE MODEL</p> <p>TARGET MODEL foreign key from source model is added to target model</p> <p>↓</p> <p>↓</p> <p><code>Parent.hasOne(Child, {as: 'OnlyChild'})</code></p> <p>adds <b>OnlyChild</b> to the target model rather than <b>child</b></p> <p>↑</p> <p>AS</p> <p><i>optional:</i> where <b>as</b> has been defined the following string will be used in place of the target model name</p>	<table><tr><th>id</th><th>name</th><th>age</th><th>onlyChildId</th></tr><tr><td colspan="4">CHILD INSTANCE DATA</td></tr></table>	id	name	age	onlyChildId	CHILD INSTANCE DATA				<table><tr><th>id</th><th>name</th><th>age</th></tr><tr><td colspan="3">PARENT INSTANCE DATA</td></tr></table>	id	name	age	PARENT INSTANCE DATA			<pre>parent.getOnlyChild() parent.setOnlyChild(child) parent.createOnlyChild(child)</pre>
id	name	age	onlyChildId														
CHILD INSTANCE DATA																	
id	name	age															
PARENT INSTANCE DATA																	
<p>SOURCE MODEL</p> <p>TARGET MODEL foreign key from source model is added to target model</p> <p>↓</p> <p>↓</p> <p><code>Parent.hasOne(Child, {as: 'OnlyChild', foreign-key: 'motherId'})</code></p> <p>uses <b>OnlyChild</b> for the accessor methods rather than <b>child</b></p> <p>adds <b>motherId</b> to the target model rather than <b>parentId</b></p> <p>↑</p> <p>AS</p> <p><i>optional:</i> where <b>as</b> has been defined the following string will be used in place of the target model name for the accessor "magic" methods</p> <p>↑</p> <p>FOREIGN-KEY</p> <p><i>optional:</i> this option overwrites the default foreign key <b>parentId</b> to the following string</p>	<table><tr><th>id</th><th>name</th><th>age</th><th>motherId</th></tr><tr><td colspan="4">CHILD INSTANCE DATA</td></tr></table>	id	name	age	motherId	CHILD INSTANCE DATA				<table><tr><th>id</th><th>name</th><th>age</th></tr><tr><td colspan="3">PARENT INSTANCE DATA</td></tr></table>	id	name	age	PARENT INSTANCE DATA			<pre>parent.getOnlyChild() parent.setOnlyChild(child) parent.createOnlyChild(child)</pre>
id	name	age	motherId														
CHILD INSTANCE DATA																	
id	name	age															
PARENT INSTANCE DATA																	

One-to-One Association: Has One

## One-To-Many

One-to-Many associations connect one source with multiple targets. The targets however, are connected to exactly one specific source.

### Has Many

[Get started](#)[Open in app](#)

<p>SOURCE MODEL</p> <p>TARGET MODEL foreign key from source model is added to target model</p> <p>↓ ↓</p> <pre>Parent.hasMany(Child)</pre>	<table><tr><th>id</th><th>name</th><th>age</th><th>parentid</th></tr><tr><td colspan="4">CHILD INSTANCE DATA</td></tr></table>	id	name	age	parentid	CHILD INSTANCE DATA				<table><tr><th>id</th><th>name</th><th>age</th></tr><tr><td colspan="3">PARENT INSTANCE DATA</td></tr></table>	id	name	age	PARENT INSTANCE DATA			<pre>parent.getChildren() parent.countChildren() parent.hasChild() parent.hasChildren() parent.setChildren(children) parent.addChild(child) parent.addChildren(children) parent.removeChild(child) parent.removeChildren(children) parent.createChild(child)</pre>
id	name	age	parentid														
CHILD INSTANCE DATA																	
id	name	age															
PARENT INSTANCE DATA																	
<p>SOURCE MODEL</p> <p>TARGET MODEL foreign key from source model is added to target model</p> <p>↓ ↓</p> <p>uses <b>offspring</b> for the accessor methods rather than <b>child</b></p> <pre>Parent.hasMany(Child, {as: 'offspring'})</pre> <p>↑ AS</p> <p><i>optional:</i> where <b>as</b> has been defined the following string will be used in place of the target model name for the accessor "magic" methods</p>	<table><tr><th>id</th><th>name</th><th>age</th><th>parentid</th></tr><tr><td colspan="4">CHILD INSTANCE DATA</td></tr></table>	id	name	age	parentid	CHILD INSTANCE DATA				<table><tr><th>id</th><th>name</th><th>age</th></tr><tr><td colspan="3">PARENT INSTANCE DATA</td></tr></table>	id	name	age	PARENT INSTANCE DATA			<pre>parent.getOffspring() parent.countOffspring() parent.hasOffspring() parent.setOffspring(children) parent.addOffspring(children) parent.removeOffspring(child) parent.createOffspring(child)</pre>
id	name	age	parentid														
CHILD INSTANCE DATA																	
id	name	age															
PARENT INSTANCE DATA																	
<p>SOURCE MODEL</p> <p>TARGET MODEL foreign key from source model is added to target model</p> <p>↓ ↓</p> <p>uses <b>offspring</b> for the accessor methods rather than <b>child</b></p> <p>adds <b>foreignKey</b> to the target model rather than <b>parentid</b></p> <pre>Parent.hasMany(Child, {as: 'offspring', foreign-key: 'foreignId'})</pre> <p>↑ AS</p> <p><i>optional:</i> where <b>as</b> has been defined the following string will be used in place of the target model name for the accessor "magic" methods</p> <p>FOREIGN-KEY</p> <p><i>optional:</i> this option overwrites the default foreign key <b>parentid</b> to the following string</p>	<table><tr><th>id</th><th>name</th><th>age</th><th>foreignid</th></tr><tr><td colspan="4">CHILD INSTANCE DATA</td></tr></table>	id	name	age	foreignid	CHILD INSTANCE DATA				<table><tr><th>id</th><th>name</th><th>age</th></tr><tr><td colspan="3">PARENT INSTANCE DATA</td></tr></table>	id	name	age	PARENT INSTANCE DATA			<pre>parent.getOffspring() parent.countOffspring() parent.hasOffspring() parent.setOffspring(children) parent.addOffspring(children) parent.removeOffspring(child) parent.createOffspring(child)</pre>
id	name	age	foreignid														
CHILD INSTANCE DATA																	
id	name	age															
PARENT INSTANCE DATA																	

## One-to-Many Association — Has M

## Many-To-Many

Many-to-Many associations connect one source with many targets. The targets can also be connected to many sources.

## Belongs To Many

Unlike the 3 previous associations, belongsToMany associations create a new joint table with the foreign keys of the target and source models.

## Conclusion

[Get started](#)[Open in app](#)

differences, Sequelize is extremely powerful in establishing relational databases and querying data.

[Sequelize](#)[Relational Databases](#)[Association](#)[Magic](#)[JavaScript](#)[About](#) [Help](#) [Legal](#)

Get the Medium app

