

Relaciones 1:N

Índice

1. .associate
2. .belongsTo()
3. .hasMany()

“

Las relaciones en Sequelize existen para optimizar la obtención de datos en una consulta a la base de datos.



”

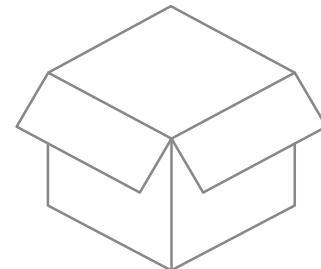
1

.associate

.associate

Las relaciones se aplican dentro de nuestro modelo de tabla. Luego de definirlo con el método `.define()`, debemos llamar a la variable creada y utilizar la propiedad `.associate` para definir nuestras relaciones.

`.associate` debe almacenar una función anónima que recibe un solo parámetro. Este será **un objeto que contiene todos los modelos**, pudiendo acceder así a cada uno de ellos.



.associate

```
const Pelicula = sequelize.define(alias, cols, config);

{}

Pelicula.associate = function(modelos){
    // Relación
}
```

Definimos una función en la propiedad **.associate** de nuestra **variable que representa al modelo** (Pelicula).

2 | .belongsTo()

.belongsTo()

Lo utilizamos para decir que un registro puede estar asociado a uno o más de otra tabla. En este caso, la relación sería de 1:N.

Para definir la relación, **.belongsTo()** recibe dos parámetros. El primero es el modelo con el que queremos relacionarlo (llamándolo a través del parámetro que contiene nuestros modelos) y el segundo es un objeto donde debemos detallar la relación.

.belongsTo()

```
Pelicula.associate = function(modelos){  
  Pelicula.belongsTo(modelos.Generos, {  
    as: "generos",  
    foreignKey: "genre_id"  
  });  
}
```

Asignamos un alias con el que llamaremos luego a la relación.

.belongsTo()

```
Pelicula.associate = function(modelos){  
  Pelicula.belongsTo(modelos.Generos, {  
    as: "generos",  
    foreignKey: "genre_id"  
  });  
}
```

Aclaramos la `foreignKey` donde se relacionan ambas tablas.

3 | .hasMany()

.hasMany()

Lo utilizamos para decir que uno o más registros pueden estar asociados a uno solo de otra tabla. En este caso, la relación sería de N:1.

Para definir la relación, **.hasMany()** recibe dos parámetros. El primero es el modelo con el que queremos relacionarlo (llamándolo a través del parámetro que contiene nuestros modelos) y el segundo es un objeto donde debemos detallar la relación.

.hasMany()

```
Genero.associate = function(modelos){  
  GenerohasMany(modelos.Pelicula, {  
    as: "peliculas",  
    foreignKey: "genre_id"  
  });  
}
```

Asignamos un alias con el que llamaremos luego a la relación.

.hasMany()

```
Genero.associate = function(modelos){  
  GenerohasMany(modelos.Pelicula, {  
    as: "peliculas",  
    {}  
    foreignKey: "genre_id"  
  });  
}
```

Aclaramos la foreignKey donde se relacionan ambas tablas.

¡ATENCIÓN!

Siempre que creamos una relación desde un modelo, debemos generar la misma desde el otro con quien está relacionado. Si no lo hacemos, Sequelize no reconoce la asociación.



Documentación



Para saber más podemos acceder a la documentación oficial de Sequelize haciendo clic en el siguiente: [link](#).

DigitalHouse>
Coding School