Associations

Patrick Schroeder
SOFTWARE ENGINEER

Overview

Associations explained

Types of associations

Methods for building associations

Express routes

Associations Explained

Associations

Associations form relationships between tables. These relationships are used to create joins.

Joins merge data.

Users Table

Users	
Id	integer
name	string
email	string
password	string

Posts Table

Posts	
Id	integer
UserId	integer
title	string
content	text

```
[
 {
   “id”: 1,
   “title”: ‘Post name’,
   “content”: ‘Post content’,
   “UserId”: 2
   “User”: {
       “name”: ‘Joe’
       …(other attributes)
   }
 }
]
```

◂ Data from Post row

◂ Data from User row

```
// adds UserId column to Posts table
Post.belongsTo(User);


Post.findById('1', {
    include: [User]
})
```

Post	
Id	integer
UserId	integer

Creating Association

1. **Define association between models.**

2. **Add include property with value as associated model.**

```
Post.findById('1', {
   include: [{
      model: User
      attributes: ['name']         // specify attributes here
  }]
})
```

Creating Association

1. Define association between models.

2. Add include property with value as associated model.

Add belongsTo() Association

Foreign Keys

Posts Table

Posts	
Id	integer
UserId	integer
title	string
content	text

Foreign key → UserId

```
// foreignKey of userId added to Post table
Post.belongsTo(User, { foreignKey: 'userId'});
```

Foreign Key

Used to link two tables together.

Model Alias

Alias

Renames the model when it’s used as an association.

```
[
 {
    “id”: 1,
    “title”: ‘Post name’,
    “content”: ‘Post content’,
    “UserId”: 2
    “UserRef”: {
        “name”: ‘Joe’
        …(other attributes)
  }
 }
]
```

```
Post.belongsTo(User, { as: 'UserRef', foreignKey: 'userId'});
Post.findById('1', {
    include: [{
        model: User, as: 'UserRef'
   }]
})
```

Alias

Used to rename model with associations.

Types of Associations

Types of Associations

One-to-One

One-to-Many

Many-to-Many

Association Comparison

One-to-One	One-to-Many	Many-to-Many
belongsTo() or hasOne()	hasMany()	belongsToMany()
Single item retrieved	Array of items retrieved	Join table
		Array of items retrieved
`User.hasOne(Post);`	`User.hasMany(Post);`	`User.belongsToMany(Post)` `Post.belongsToMany(User)`

One-to-many Association

```
// foreignKey = PostId in Comments table

Post.hasMany(Comment, { as: 'All_Comments'});


Post.findById('1', {
    include: Comment, as: 'All_Comments'
})
```

One-to-many Association

1. Define association between models.

2. Add include property with value as associated model.

```
{
    “id”: 1,
    “title”: ‘Post name’,
    “content”: ‘Post content’
    “All_Comments”: [{
        “the_comment”: ‘first’
     }, {
        “the_comment”: ‘hi’
     }, {
        “the_comment”: ‘sup yo’
    }]
 }
```

◀ Single post

◀ Array of comments

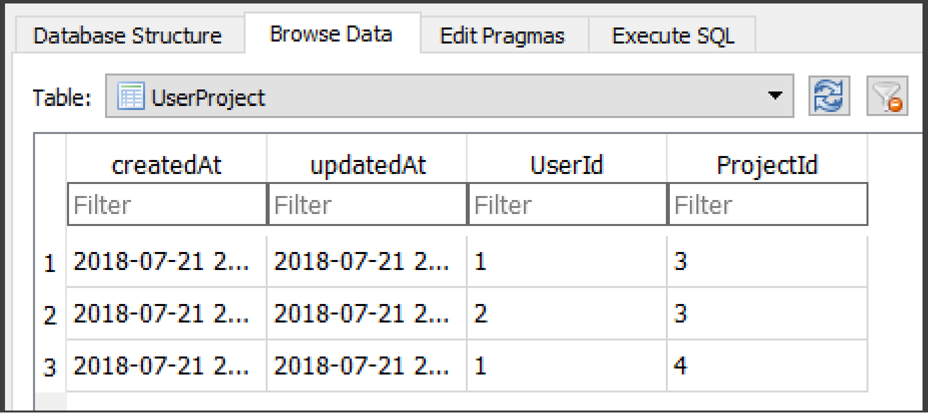
Many-to-many Association

```
// foreignKey = ProjectId and UserId in UserProject table

User.belongsToMany(Project, { as: 'Tasks', through: 'UserProject' });

Project.belongsToMany(User, { as: 'Workers', through: 'UserProject' });
```

```
User.findById('1', {
    include: Project, as: 'Tasks'
})
```



	createdAt	updatedAt	UserId	ProjectId
	Filter	Filter	Filter	Filter
1	2018-07-21 2...	2018-07-21 2...	1	3
2	2018-07-21 2...	2018-07-21 2...	2	3
3	2018-07-21 2...	2018-07-21 2...	1	4

Many-to-many Association

1. **Define association between 2 tables.**

2. **Add include property with value as associated model.**

```
// foreignKey = ProjectId and UserId in UserProject table
User.belongsToMany(Project, { as: 'Tasks', through: 'UserProject' });
Project.belongsToMany(User, { as: 'Workers', through: 'UserProject' });


User.findById('1', {
    include: Project, as: 'Tasks', attributes: ['name']
})
```

Many-to-Many Association

1. **Define association between 2 tables.**
2. **Add include property with value as associated model.**
3. **Add in optional attributes for either model.**

```
{
   “id”: 1,
   “name”: ‘Hakeem’,
   “email”: ‘1234@email.com’
   “Tasks”: [{
      “name”: ‘project 1’
     }, {
      “name”: ‘project 4’
     }
   }]
 }
```

◀ User

◀ Array of projects

Getters / Setters

Methods used perform CRUD on members of association.

Getters / Setters

set

add

get

remove

Getters / Setters

set

- setWorkers([])

add

- addWorkers()

get

- getWorkers()

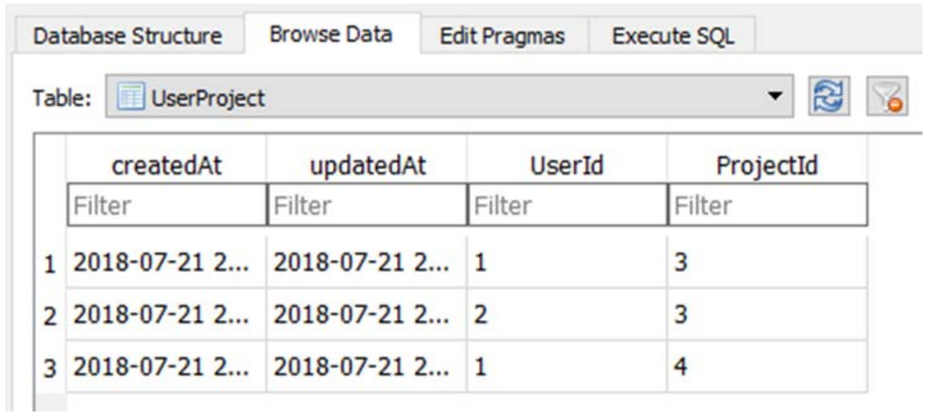
remove

- removeWorkers()

```
Project.create({
    name: 'project name'
    }).then((project) => {
     project.setWorkers([1, 2]);
  })

app.put('/addWorker', (req, res)
 => { Project.findById(4)
              .then((project) => {
    project.addWorkers(5);
  })
```

◀ Set a new Worker



Database Structure | Browse Data | Edit Pragmas | Execute SQL

Table: UserProject

	createdAt	updatedAt	UserId	ProjectId
1	2018-07-21 2...	2018-07-21 2...	1	3
2	2018-07-21 2...	2018-07-21 2...	2	3
3	2018-07-21 2...	2018-07-21 2...	1	4

◀ Add a Worker

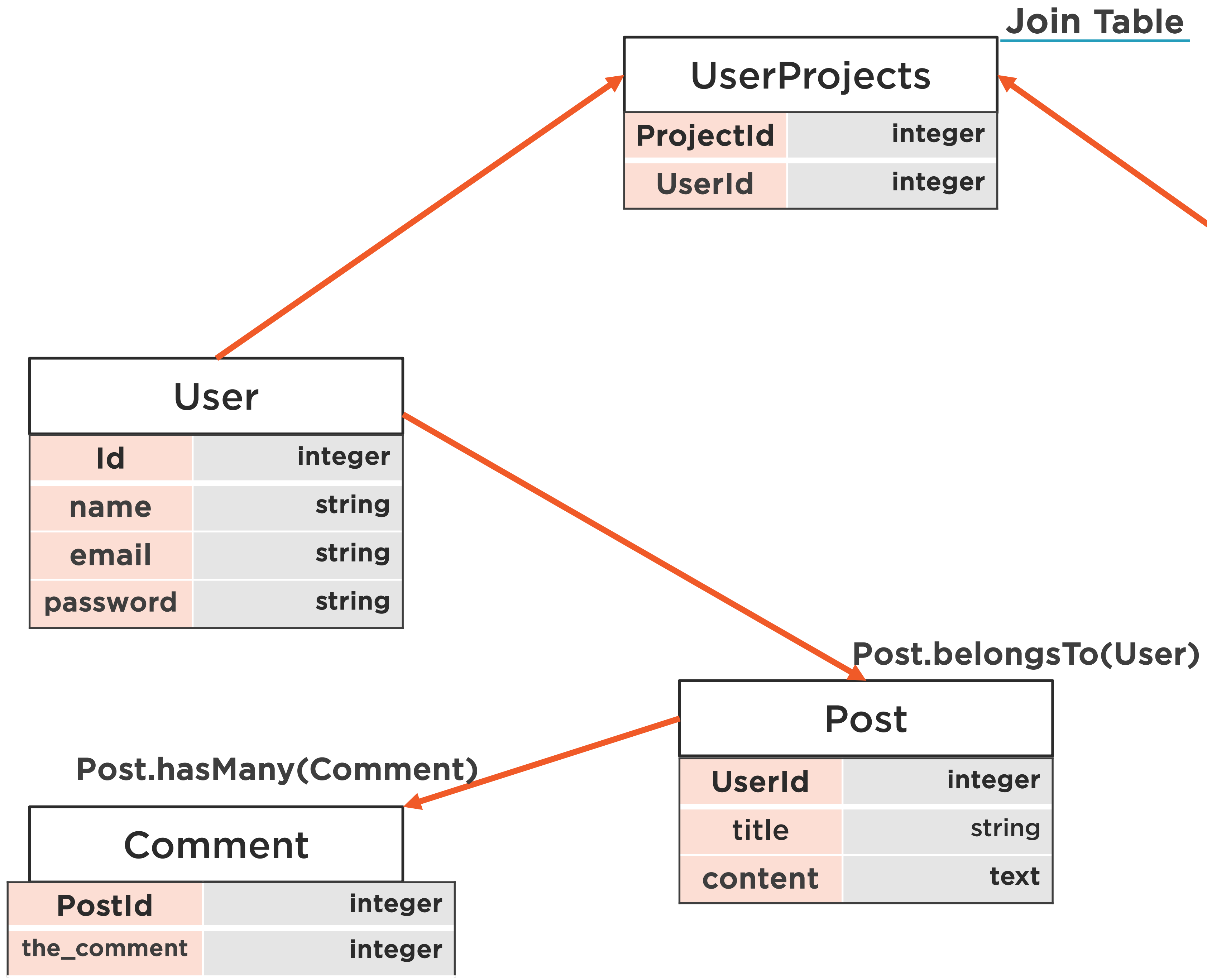
Summary

Associations

- belongsTo, hasOne
- hasMany
- belongsToMany

Foreign key

Test response


Join Table
UserProjects
ProjectId integer
UserId integer
User
Id integer
name string
email string
password string
Project
Id integer
title string
Post.belongsTo(User)
Post
UserId integer
title string
content text
Post.hasMany(Comment)
Comment
PostId integer
the_comment integer