## Lazy Loading

const Ship = sequelize.define('ship', {

name: DataTypes.TEXT

});

const Captain = sequelize.define('captain', {…});

Captain.hasOne(Ship);

Ship.belongsTo(Captain);

const awesomeCaptain = await Captain.findByPK(1);

const hisShip = await awesomeCaptain.getShip();

console.log('Ship Name:', hisShip.name);

## Eager Loading

* Sử dụng ‘include’

const awesomeCaptain = await Captain.findOne({

where: {

name: "Jack Sparrow"

},

include: Ship

});

console.log('Ship Name:', awesomeCaptain.ship.name);

Khi nested fetch object thì Sqlize tự động object hóa data của association

[{

"name": "A Task",

"id": 1,

"userId": 1,

"user": {

"name": "John Doe",

"id": 1

}

}]

tasks[0].user instanceof User 🡪 true

## Fetching Alias – Association

Nếu quan hệ được defined theo kiểu alias 🡪 phải sử dụng alias để eagerFetch

const users = await User.findAll({

include: { model: Tool, as: 'Instruments' }

});

User.findAll({ include: 'Instruments' }); // Also works

User.findAll({ include: { association: 'Instruments' } }); // Also works

[{

"name": "John Doe",

"id": 1,

"Instruments": [{

"name": "Scissor",

"id": 1,

"userId": 1

}]

}]

## Inner Join with [required: true]

Mặc định là Sqlize sẽ sử dụng OUTER JOIN

outerJoin 🡪 innerJoin

User.findAll({

include: {

model: Task,

required: true

}

});

Và sqlize tự động activate required: true khi mình sử dụng filter cho association

User.findAll({

include: {

model: Tool,

as: 'Instruments'

where: {

size: {

[Op.ne]: 'small'

}

}

}

});

Chính vì ‘where’ nằm ở trong ‘include’ nên sqlize tự động enable required: true

SELECT

`user`.`id`,

`user`.`name`,

`Instruments`.`id` AS `Instruments.id`,

`Instruments`.`name` AS `Instruments.name`,

FROM `users` AS `user`

INNER JOIN `tools` AS `Instruments` ON

`user`.`id` = `Instruments`.`userId` AND

`Instruments`.`size` != 'small';

## Complex ‘WHERE’ at top-level

Thay vì mình phải đặt ‘WHERE’ bên trong include 🡪 mình có thể đặt nó ở top-level bằng :

'$nested.column$'

Đặt WHERE ở đây sẽ giúp ta tạo ‘required: false’ query

User.findAll({

where: {

'$Instruments.size$': { [Op.ne]: 'small' }

},

include: [{

model: Tool,

as: 'Instruments'

}]

});

SELECT

`user`.`id`,

`user`.`name`,

`Instruments`.`id` AS `Instruments.id`,

`Instruments`.`name` AS `Instruments.name`,

...

FROM `users` AS `user`

LEFT OUTER JOIN `tools` AS `Instruments` ON

`user`.`id` = `Instruments`.`userId`

WHERE `Instruments`.`size` != 'small';

Đôi khi có thể rất deep : $some.super.deeply.nested.column$

Và ta cũng thể làm điều tương tự (LEFT OUTER JOIN – required: false) bằng Inner WHERE

await User.findAll({

include: {

model: Tool,

as: 'Instruments',

where: {

size: { [Op.ne]: 'small' }

},

required: false

}

});

Kết quả :

-- Top-level where, with default `required: false`

SELECT [...] FROM `users` AS `user`

LEFT OUTER JOIN `tools` AS `Instruments` ON

`user`.`id` = `Instruments`.`userId`

WHERE `Instruments`.`size` != 'small';

## Right OUTER Join

User.findAll({

include: [{

model: Task,

right: true // will create a right join

}]

});

User.findAll({

include: [{

model: Tool,

where: { name: { [Op.ne]: 'empty trash' } },

required: false

right: true // will create a right join

}]

});

Khi sử dụng 🡪 không được enable

* ‘required: true’
* inner WHERE without ‘required: false’

## Mulitple Eager Loading

Foo.findAll({

include: [

{

model: Bar,

required: true

},

{

model: Baz,

where: /\* ... \*/

},

Qux // Shorthand syntax for { model: Qux } also works here

]

})

## Nested Eager Loading

const users = await User.findAll({

include: {

model: Tool,

as: 'Instruments',

include: {

model: Teacher,

include: [ /\* etc \*/ ]

}

}

});

## Include Everything

// Fetch all models associated with User

User.findAll({ include: { all: true }});

// Fetch all models associated with User and their nested associations (recursively)

User.findAll({ include: { all: true, nested: true }});

## Include Soft Deleted

User.findAll({

include: [{

model: Tool,

as: 'Instruments',

where: { size: { [Op.ne]: 'small' } },

paranoid: false

}]

});

## Order Associations (top-level)

Company.findAll({

include: Division,

order: [

[Division, 'name', 'ASC']

]

});

Company.findAll({

// If the include uses an alias...

include: { model: Division, as: 'Div' },

order: [

[{ model: Division, as: 'Div' }, 'name', 'DESC']

]

});

Company.findAll({

// If we have includes nested in several levels...

include: {

model: Division,

include: Department

},

order: [

[Division, Department, 'name', 'DESC']

]

});

## Order Association (nested-level)

Nằm trong include

// This only works for `separate: true` (which in turn

// only works for has-many relationships).

User.findAll({

include: {

model: Post,

separate: true,

order: [

['createdAt', 'DESC']

]

}

});

## findAndCountAll

User.findAndCountAll({

include: [

{ model: Profile, required: true }

],

limit: 3

});

Eager Load – N\_N

Foo.belongsToMany(Bar, { through: 'Foo\_Bar' });

Bar.belongsToMany(Foo, { through: 'Foo\_Bar' });

const foo = await Foo.create({ name: 'foo' });

const bar = await Bar.create({ name: 'bar' });

await foo.addBar(bar);

const fetchedFoo = Foo.findOne({ include: Bar });

{

"id": 1,

"name": "foo",

"Bars": [

{

"id": 1,

"name": "bar",

"Foo\_Bar": {

"FooId": 1,

"BarId": 1

}

}

]

}

Đối với many to many, Sqlize tự động thêm Junction\_Model vào “Foo\_Bar”

Custom ~ field trả về của Junction\_Model

Foo.findAll({

include: [{

model: Bar,

through: {

attributes: [/\* list the wanted attributes here \*/]

attributes: [] // hoặc không trả về gì cả

}

}]

});

Kết quả khi không trả về gì cả

{

"id": 1,

"name": "foo",

"Bars": [

{

"id": 1,

"name": "bar"

}

]

}

Sử dụng Filter cho Junction\_Model

User.findAll({

include: [{

model: Project,

through: {

where: {

// Here, `enable` is a column present at the junction table

enable: {

[Op.eq]: ‘ton\_tai’

}

}

}

}]

});