## Creating in bulk

const captains = await Captain.bulkCreate([

{ name: 'Jack Sparrow' },

{ name: 'Davy Jones' }

]);

console.log(captains[0].id); // 1 // (or another auto-generated value)

validation option : lỗi nếu create obj với field không tồn tại trong model

const Foo = sequelize.define('foo', {

bar: {

type: DataTypes.TEXT,

validate: {

len: [4, 6]

}

}

});

No Error 🡪 Tạo bth

await Foo.bulkCreate([

{ name: 'abc123' },

{ name: 'name too long' }

]);

Err 🡪 Tạo không bình thường

await Foo.bulkCreate([

{ name: 'abc123' },

{ name: 'name too long' }

], { validate: true });

* Only field

await User.bulkCreate([

{ username: 'foo' },

{ username: 'bar', admin: true }

], { fields: ['username'] });

// Neither foo nor bar are admins.

## ORDER

Options : ASC, DESC, NULLS FIRST

Subtask.findAll({

order: [

['title', 'DESC'],

sequelize.fn('max', sequelize.col('age')),

[sequelize.fn('max', sequelize.col('age')), 'DESC'],

// Will order by otherfunction(`col1`, 12, 'lalala') DESC

[sequelize.fn('otherfunction', sequelize.col('col1'), 12, 'lalala'), 'DESC'],

With model

[Task, 'createdAt', 'DESC'],

[Task, Project, 'createdAt', 'DESC'],

['Task', 'createdAt', 'DESC'],

['Task', 'Project', 'createdAt', 'DESC'],

[Subtask.associations.Task, 'createdAt', 'DESC'],

[Subtask.associations.Task, Task.associations.Project, 'createdAt', 'DESC'],

[{model: Task, as: 'Task'}, 'createdAt', 'DESC'],

[{model: Task, as: 'Task'}, {model: Project, as: 'Project'}, 'createdAt', 'DESC']

],

// Will order by max age descending

order: sequelize.literal('max(age) DESC'),

// Will order by max age ascending assuming ascending is the default order when direction is omitted

order: sequelize.fn('max', sequelize.col('age')),

// Will order by age **ascending** assuming ascending is the **default** order **when** **direction is** **omitted**

order: sequelize.col('age'),

// Will order randomly based on the dialect (instead of fn('RAND') or fn('RANDOM'))

order: sequelize.random()

});

Foo.findOne({

order: [

// will return `name`

['name'],

// will return `username` DESC

['username', 'DESC'],

// will return max(`age`)

sequelize.fn('max', sequelize.col('age')),

// will return max(`age`) DESC

[sequelize.fn('max', sequelize.col('age')), 'DESC'],

// will return otherfunction(`col1`, 12, 'lalala') DESC

[sequelize.fn('otherfunction', sequelize.col('col1'), 12, 'lalala'), 'DESC'],

// will return otherfunction(awesomefunction(`col`)) DESC, This nesting is potentially infinite!

[sequelize.fn('otherfunction', sequelize.fn('awesomefunction', sequelize.col('col'))), 'DESC']

]

});

* A call to Sequelize.fn (which will generate a function call in SQL)

## Group

Project.findAll({ group: 'name' });

// yields 'GROUP BY name'

## Pagination

// Fetch 10 instances/rows

Project.findAll({ limit: 10 });

// Skip 8 instances/rows

Project.findAll({ offset: 8 });

// Skip 5 instances and fetch the 5 after that

Project.findAll({ offset: 5, limit: 5 });

## Count

Count models

console.log(`There are ${await Project.count()} projects`);

Count with condition

const amount = await Project.count({

where: {

id: {

[Op.gt]: 25

}

}

});

console.log(`There are ${amount} projects with an id greater than 25`);

## Max, min & sum

await User.max('age'); // 40

await User.max('age', { where: { age: { [Op.lt]: 20 } } }); // 10

await User.min('age'); // 5

await User.min('age', { where: { age: { [Op.gt]: 5 } } }); // 10

await User.sum('age'); // 55

await User.sum('age', { where: { age: { [Op.gt]: 5 } } }); // 50