

Knex.js

Migrate to Postgresql database

<https://github.com/TechMaster/knex-migrate-tutor>

cuong@techmaster.vn

<http://techmaster.vn>

```
npm init
```

```
npm install --save knex pg
```

```
knex init
```

```
knex migrate:latest
```

```
knex migrate:rollback
```

```
knex-migrate up
```

```
knex-migrate down
```

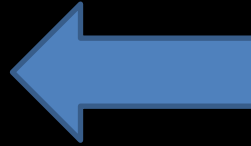
```
knex-migrate rollback
```

```
knex seed:make setup
```

```
knex seed:run
```

```
migrations
|
|--20170103120042_setup.js
|
|--20170103171359_step1.js

seeds
|
|--category.js
|
|--person.js
|
|--post.js
|
knexfile.js
package.json
```



Folder structure

```
module.exports = {
  development: {
    client: 'postgresql',
    connection: {
      host: '192.168.1.60',
      database: 'payroll',
      user: 'postgres',
      password: 'abc'
    },
  },
  production: {
    client: 'postgresql',
    connection: {
      database: 'my_db',
      user: 'username',
      password: 'password'
    },
    pool: {
      min: 2,
      max: 10
    },
    migrations: {
      tableName: 'knex_migrations'
    }
  }
};
```

`knex init`
creates file `knexfile.js`

Edit `knexfile.js` to
connect to your
Postgresql database

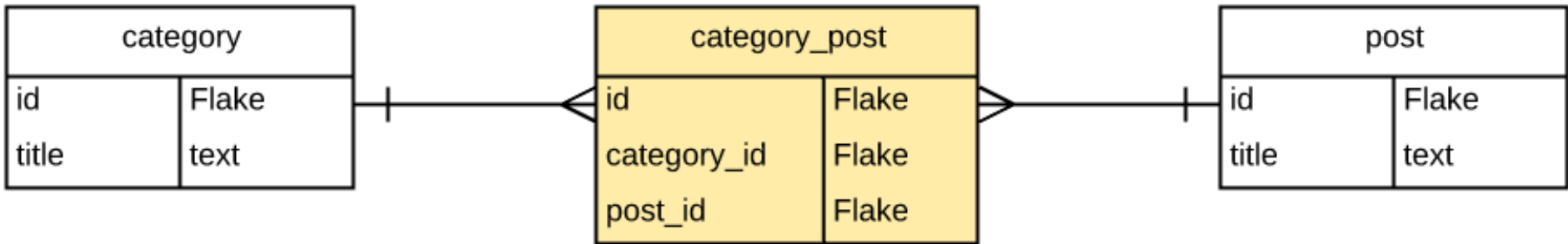


`knex migrate` sẽ ghi vào bảng `public.knex_migrations`

Migration file

- In folder migrations
- **up** create table và **down** drop table

```
exports.up = function(knex, Promise) {  
  return Promise.all([  
  
  ]);  
  
exports.down = function(knex, Promise) {  
  return Promise.all([  
  
  ]);  
};
```



Trường id sẽ được sinh bằng một function **util.id_generator()**

Bộ key (category_id, post_id) phải là duy nhất

```
knex.schema.withSchema('blog').createTable('post', function(table){
  table.bigInteger('id').primary().defaultTo(knex.raw('util.id_generator()'));
  table.text('title').comment("You can comment a column");
  table.comment('All posts will store here');
}),
```

```
knex.schema.withSchema('blog').createTable('category', function(table){
  table.bigInteger('id').primary().defaultTo(knex.raw('util.id_generator()'));
  table.text('title');
  table.comment('Category blog post');
}),
```

```
knex.schema.withSchema('blog').createTable('category_post', function(table){
  table.bigInteger('id').primary().defaultTo(knex.raw('util.id_generator()'));
  table.bigInteger('post_id').references('id').inTable('blog.post');
  table.bigInteger('category_id').references('id').inTable('blog.category');
  table.unique(['post_id', 'category_id']);
  table.comment('Intersection table')
})
```

Tạo bảng **post** trong schema **blog**

```
knex.schema.withSchema('blog').createTable('post',  
function(table){}
```


Add comment to table and column

```
knex.schema.withSchema('blog').createTable('category_post',  
function(table){  
    ...  
    table.comment('Intersection table')  
})
```

```
knex.schema.withSchema('blog').createTable('category_post', function(table){  
  table.bigInteger('post_id').references('id').inTable('blog.post');  
  table.bigInteger('category_id').references('id').inTable('blog.category');  
  table.unique(['post_id', 'category_id']);  
})
```

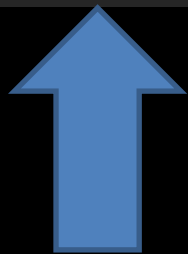


category_post	
id	Flake
category_id	Flake
post_id	Flake

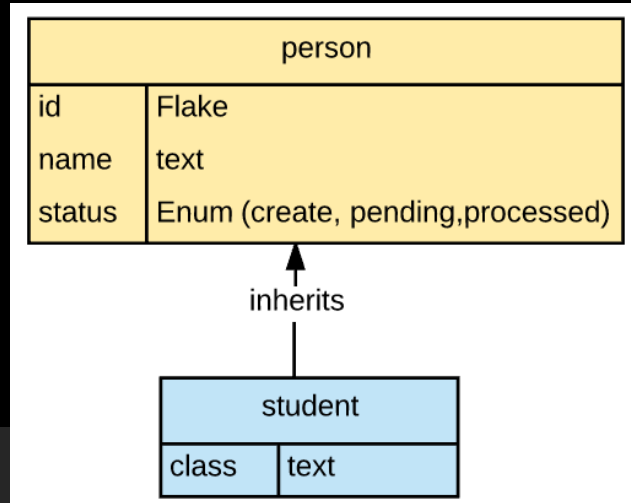
unique constraint for two columns (post_id , category_id)

Tạo một cột có dạng dữ liệu đặc biệt mà Knex không hỗ trợ. Ví dụ Postgresql array

```
table.specificType('authors', knex.raw('text[]'));
```



Tạo bảng kế thừa



```
exports.up = function(knex, Promise) {  
  return Promise.all([  
    knex.schema.withSchema('blog').createTable('person', function(table){  
      table.bigInteger('id').primary();  
      table.text('name');  
    }),  
  
    knex.schema.withSchema('blog').createTable('student', function(table){  
      table.inherits('blog.person');  
      table.text('class');  
    })  
  ]);  
};
```

knex chưa hỗ trợ tạo enum type thực thụ

<https://github.com/tgriesser/knex/issues/394>

```
knex.schema.createTable('person', function(table){  
  table.bigInteger('id').primary();  
  table.enum('status', ['create', 'pending', 'processed']);  
}),
```

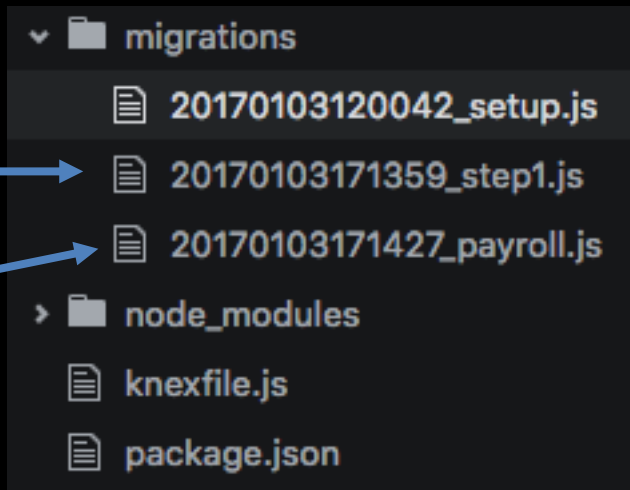
```
CONSTRAINT person_status_check CHECK (status = ANY  
(ARRAY['create'::text, 'pending'::text, 'processed'::text]))
```

Use raw to create Postgresql enumeration type

```
knex.schema.raw("CREATE TYPE blog.person_status AS  
ENUM ('create', 'pending', 'processed')"),  
  
table.specificType('status', 'blog.person_status')  
  
knex.schema.raw("DROP TYPE blog.person_status")
```

knex migrate:make

```
knex migrate:make step1  
knex migrate:make payroll
```



Roll back

```
knex migrate:rollback
```

```
Using environment: development
```

```
Batch 1 rolled back: 1 migrations
```




knex migrate chạy như thế nào

2 bảng ở public schema

- knex-migrations : lưu **id** các file migrate đã chạy (**up**)
- knex-migrations-lock

knex migrate:currentVersion trả về **id** của migrate cuối cùng chạy thành công. Nếu chưa chạy migrate nào thì trả về **none**

Xảy ra lỗi có thể xóa hết bản ghi trong knex-migrations rồi chạy lại từ đầu

knex-migrate

<https://github.com/sheerun/knex-migrate>

Migrate up / down từng bước một hoặc đến version cụ thể, chi tiết hơn tool chuẩn của knex

```
$ knex-migrate up                # migrate everything
$ knex-migrate up 20160905       # migrate upto given migration name
$ knex-migrate up --to 20160905  # the same as above
$ knex-migrate up --only 201609085 # migrate up single migration
$ knex-migrate down --to 0       # rollback all migrations
$ knex-migrate down              # rollback single migration
$ knex-migrate rollback          # rollback previous "up"
$ knex-migrate redo --verbose    # rollback and migrate everything
```



Tạo file sinh dữ liệu
`knex seed:make setup`

Chạy file sinh dữ liệu
`knex seed:run`

seed/setup.js

```
exports.seed = function(knex, Promise) {  
  // Xóa dữ liệu cũ  
  return knex('blog.category').del()  
    .then(function () {  
      return Promise.all([  
        // Chèn dữ liệu vào đây  
        knex('blog.category').insert({title: 'iOS'}),  
        knex('blog.category').insert({title: 'Linux'}),  
        knex('blog.category').insert({title: 'PHP'})  
      ]);  
    });  
};
```

Insert array data type

```
knex('blog.post').insert({title: 'iOS is fun',  
  authors: '{"Page", "Plant", "Jones", "Bonham"}' } ),
```



```
SELECT authors[1:3] from blog.post  
{"Plant", "Jones", "Bonham" }
```

Chèn dữ liệu enum

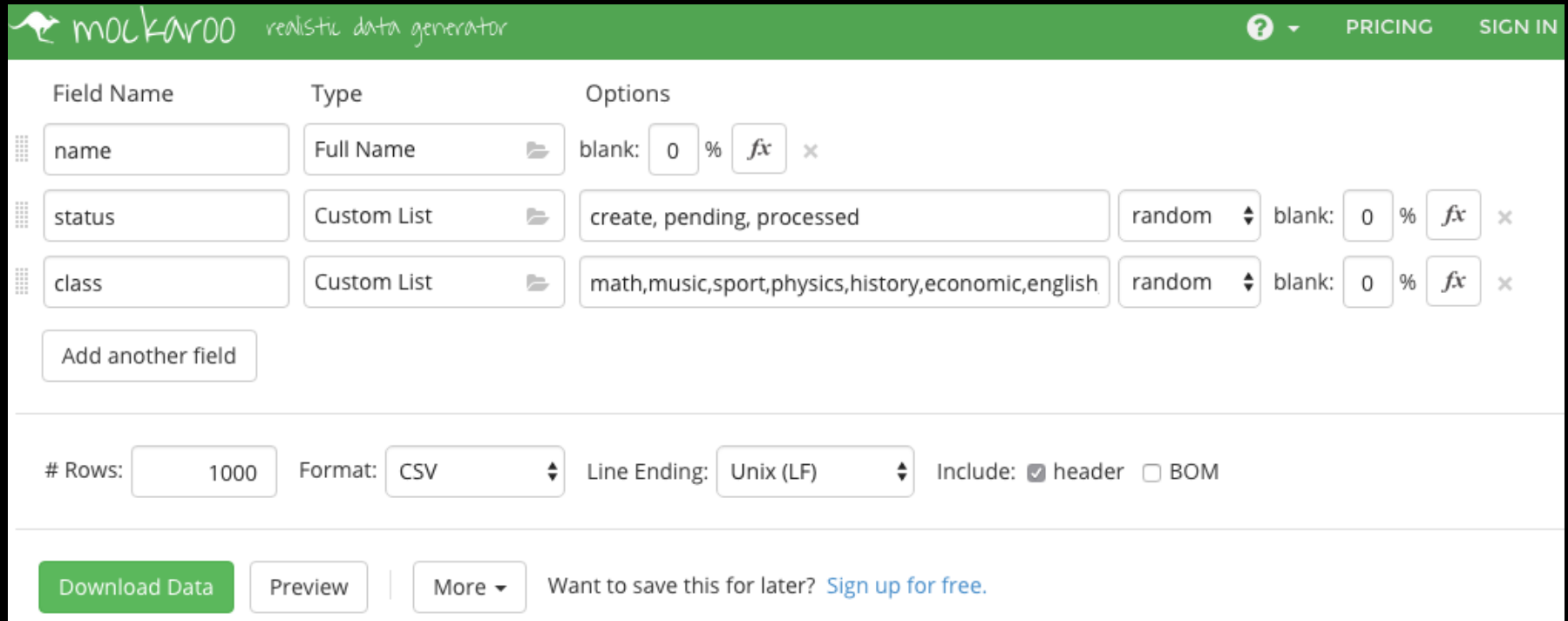
```
return knex('blog.person').del()
.then(function () {
  return Promise.all([
    knex('blog.person').insert({name: 'John', status: 'create'}),
    knex('blog.person').insert({name: 'Ben', status: 'pending'}),
    knex('blog.person').insert({name: 'Bill', status: 'processed'}),
    //hood is not item in blog.person_status
    //knex('blog.person').insert({name: 'Jake', status: 'hood'})
  ]);
});
```



Giá trị không có trong enum nếu chèn sẽ báo lỗi

Insert data from file

- Cannot insert manually > 50 records
- mockaroo.com free for <= 1000 records



The screenshot shows the Mockaroo website interface, a realistic data generator. The header is green with the Mockaroo logo and navigation links for PRICING and SIGN IN. The main form is white and contains three fields: 'name' (Full Name), 'status' (Custom List), and 'class' (Custom List). Each field has options for blank, random, and a formula (fx). The 'status' and 'class' fields have dropdown menus for random selection. Below the fields is a button 'Add another field'. At the bottom, there are settings for the number of rows (1000), format (CSV), line ending (Unix (LF)), and include options (header checked, BOM unchecked). A green 'Download Data' button is on the left, and a 'Preview' button is next to it. A 'More' dropdown is also present. A footer message says 'Want to save this for later? Sign up for free.'

Field Name	Type	Options
name	Full Name	blank: 0 % <input type="text" value="fx"/> x
status	Custom List	create, pending, processed random blank: 0 % <input type="text" value="fx"/> x
class	Custom List	math,music,sport,physics,history,economic,english random blank: 0 % <input type="text" value="fx"/> x

Add another field

Rows: 1000 Format: CSV Line Ending: Unix (LF) Include: ☒ header ☐ BOM

Download Data Preview More Want to save this for later? [Sign up for free.](#)

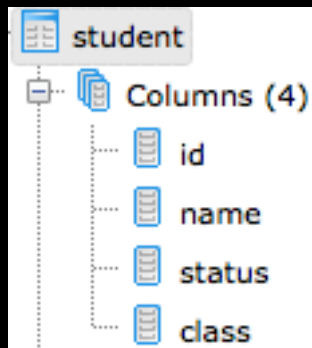
student.cvs generated by mockaroo

```
name,status,class
Lisa Reynolds,pending,music
Betty Lynch,pending,english
Mark Sims,pending,math
Paul Mason,create,english
Lillian Harrison,processed,sport
Eugene Crawford,pending,english
Irene Ramos,processed,chemistry
Julia Sanders,create,english
Michael Sanders,create,physics
Willie Romero,pending,physics
Patricia Montgomery,pending,sport
```


npm install --save-dev knex-seed-file

```
const path = require('path');
const seedFile = require('knex-seed-file');
exports.seed = function(knex, Promise) {
  return Promise.join(
    knex('blog.student').del(),
    seedFile(knex, path.resolve('./seeds/student.csv'),
    'blog.student', [
      'name',
      'status',
      'class'
    ], {
      columnSeparator: ',',
      ignoreFirstLine: true
    })
  );
};
```

Result !



id bigint	name text	status blog.pers...	class text
1420058186093692038	Betty Lynch	pending	english
1420058186093692039	Mark Sims	pending	math
1420058186102080649	Paul Mason	create	english
1420058186102080650	Lillian Harrison	processed	sport
1420058186102080648	Lisa Reynolds	pending	music
1420058186110469259	Eugene Crawford	pending	english
1420058186110469260	Irene Ramos	processed	chemistry
1420058186110469261	Julia Sanders	create	english
1420058186160800910	Michael Sanders	create	physics
1420058186160800911	Willie Romero	pending	physics
1420058186160800912	Patricia Montgomery	pending	sport