Knex.js

Migrate to Postgresql database

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

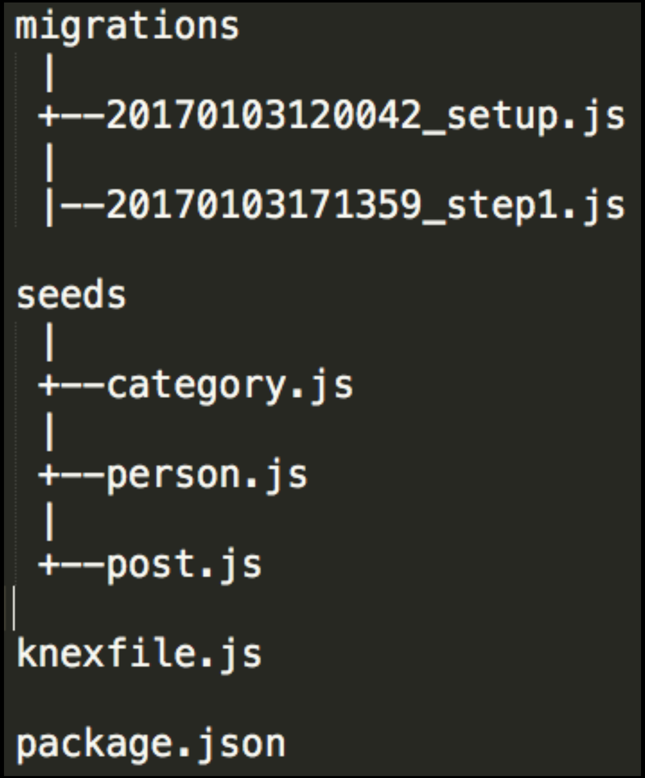
cuong@techmaster.vn
http://techmaster.vn

Installation

```
npm init
npm install --save knex pg
npm install –g knex-migrate
npm install -D knex-seed-file
knex init
```

Migration

```
knex migrate:latest
knex migrate:rollback
knex migrate:currentVersion
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 will write public.knex_migrations

Migration file

- In folder migrations
- up create tables và down drop tables

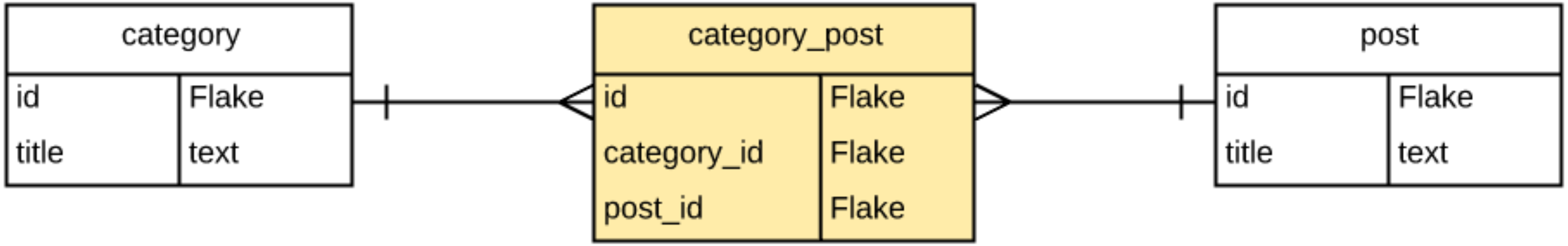
```
exports.up = function(knex, Promise) {
  return Promise.all([

};

exports.down = function(knex, Promise) {
  return Promise.all([

  ]);
};
```

Sample database schema. Many : many relationship



Primary key id type BigInt will be generated by a function **`util.id_generator()`**
Call this kind of primary is Flake (Twitter Snow Flake, see
http://rob.conery.io/2014/05/29/a-better-id-generator-for-postgresql)

Category_post is intersection table where composite key (category_id, post_id) must be unique

```
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')
   })
```

Create table post in schema blog

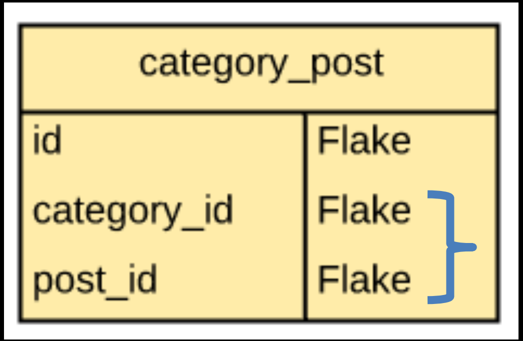
```
knex.schema.withSchema('blog').createTable('post',
function(table){}
```

Always use schema to divide hundreds of tables into manageable schema

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']);
})
```



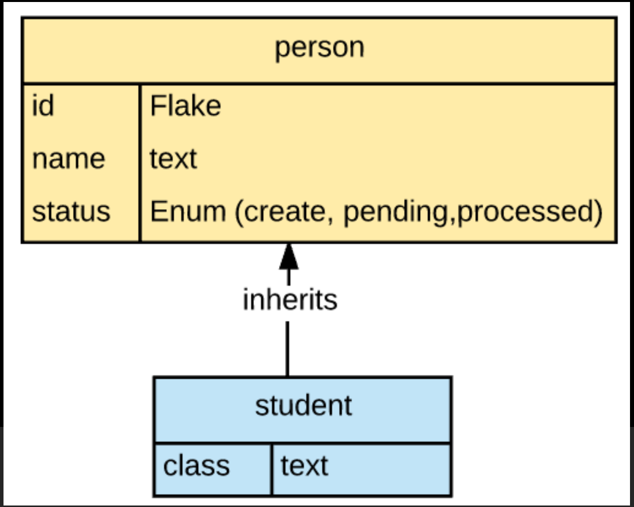
category_post	
id	Flake
category_id	Flake
post_id	Flake

unique constraint for two columns (post_id , category_id)

Use specificType and knex.raw to create column if its type is not natively supported by Knex

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

student inherits from person



```
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 does not support true enumeration type

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

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

knex creates constrain check

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

Use raw to create Postgresql enumeration type

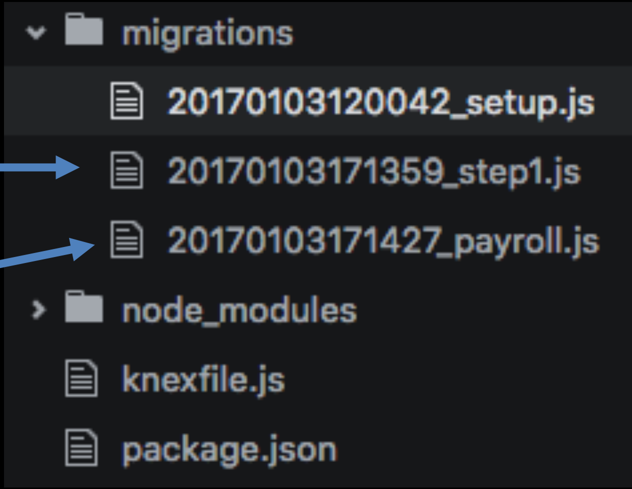
```
knex.schema.raw("CREATE TYPE blog.person_status AS
ENUM ('create', 'pending', 'processed')"),

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
```



```
migrations
    20170103120042_setup.js
    20170103171359_step1.js
    20170103171427_payroll.js
node_modules
knexfile.js
package.json
```

Roll back

```
knex migrate:rollback
Using environment: development
Batch 1 rolled back: 1 migrations
```

knex migrate

In public schema

- knex-migrations : store id of already run migration files (up)
- knex-migrations-lock

`knex migrate:currentVersion` returns id of last successful migration, returns none if no migration has run yet

When migration error happens, you cannot migrate nor rollback, remove all records in knex-migrations and clean database

knex-migrate

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

Migrate up / down each step or to specific version . It is better thanh default knex migrate cli. Should install knex-migrate globally as command

```
$ knex-migrate up                     # migrate everytings
$ 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
```

Create file prepare data

```
knex seed:make setup
```

Run to insert sample data

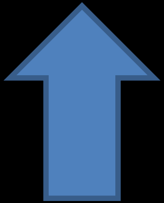
```
knex seed:run
```

seed/setup.js

```
exports.seed = function(knex, Promise) {
  // Delete old data
  return knex('blog.category').del()
    .then(function () {
      return Promise.all([
        // Insert sample data
        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"}
```

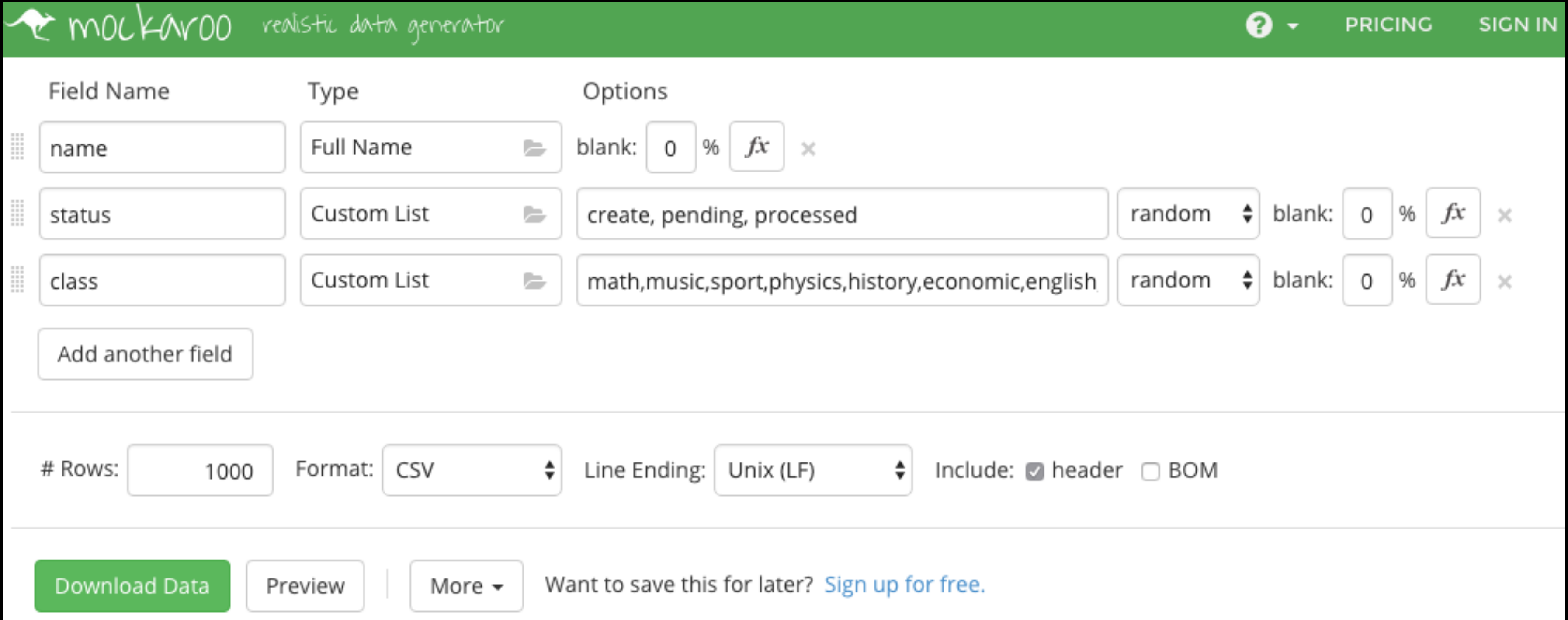
Insert enum data

```
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'})
 ]);
});
```

Value not in enum will raise error when insert

Insert data from file

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


mockaroo realistic data generator
PRICING
SIGN IN

Field Name	Type	Options
name	Full Name	blank: 0 %
status	Custom List	create, pending, processed; random; blank: 0 %
class	Custom List	math,music,sport,physics,history,economic,english; random; blank: 0 %

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.

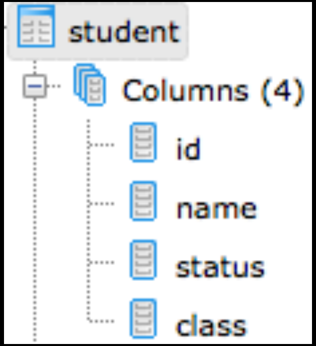
student.cvs generated by mockaroo

```
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
```

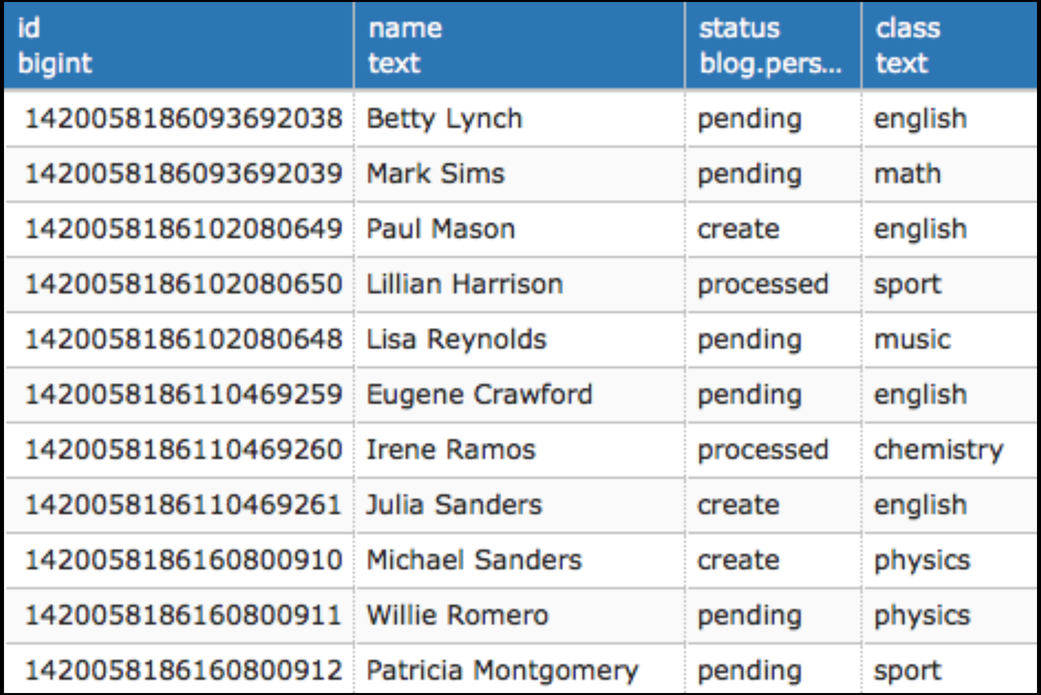
npm install --save-dev knex-seed-file

```
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 !



student
Columns (4)
id
name
status
class



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