## Migrations

**Migrations** is used to keep track of changes to the database in the same way Git is used to keep track of the source code. With migrations the existing database can be transferred into another state and vice versa while those state transitions are saved in migration files. This describes how to get to the new state and how to revert the changes in order to get back to the old state.

The Sequelize Command Line Interface CLIis used for migrations and project bootstrapping.A Migration in Sequelize is a javascript file which exports two functions, up and down that dictate how to perform the migration and undo it. These functions can be called automatically by the CLI.

## Installing the CLI

To install the Sequelize CLI:

* Open the Terminal and type   
  npm install --save dev sequelize-cli
* To create an empty project:  
   npx sequelize-cli init  
  This will create following folders
* config, contains config file, which tells CLI how to connect with database
* models, contains all models for your project
* migrations, contains all migration files
* seeders, contains all seed files

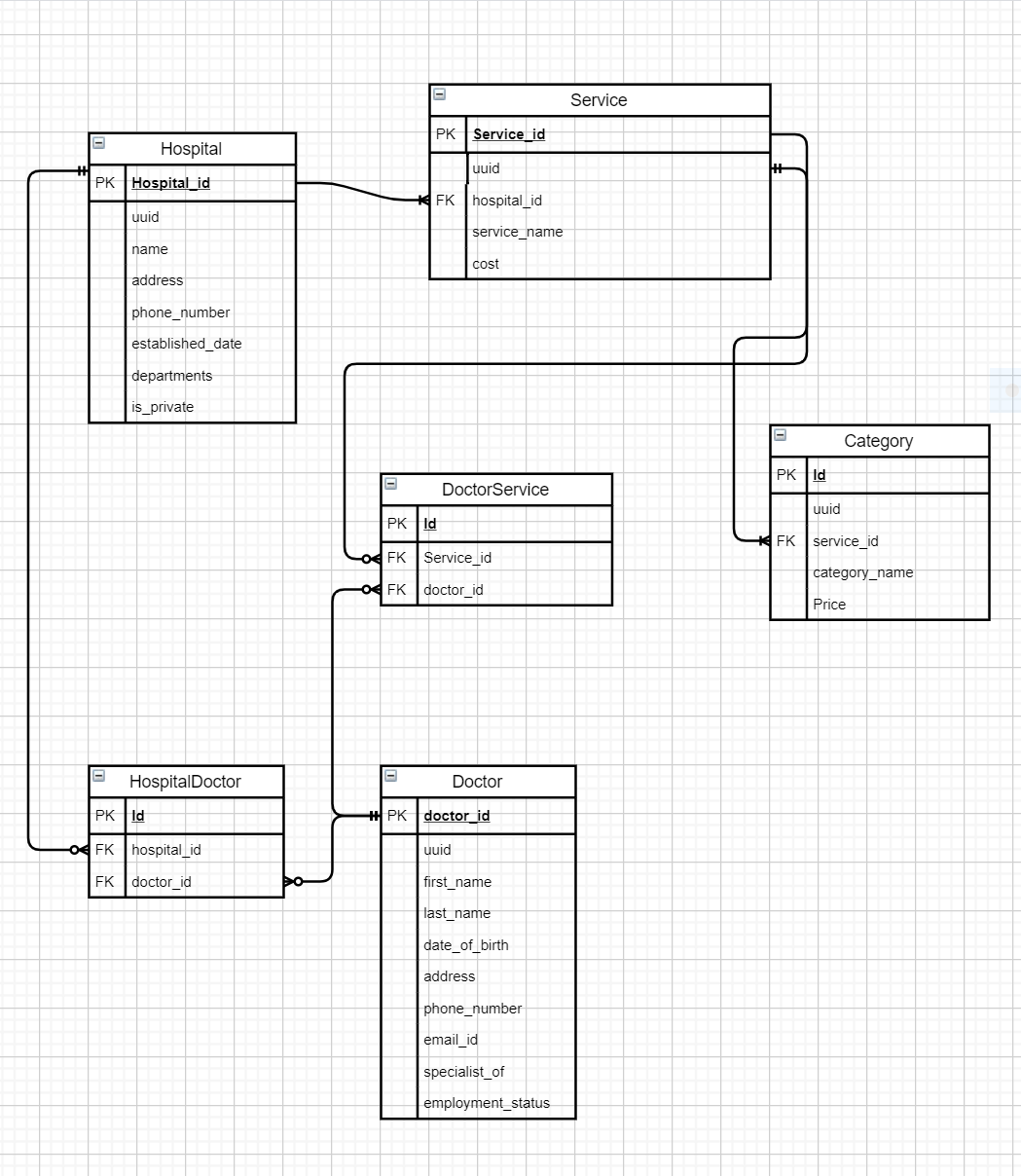
To instruct the CLI how to connect to the database you need to open the default config file config/config.json and edit the file. As Sequelize CLI assumes mysql by default, you need to change the content of the "dialect" option. To postgres added a password and the default port for postgres for database development, database production and database prod respectively as so.

**"username": "postgres"**

**"Password": "password"**

**"port" :5432**

**Data Model for Health Care System**

****

**Relationship between the tables**

One to many

* A Hospital offers many services
* A Service may have many categories

Many to Many

* A Hospital may have many Doctors
* A Doctor may be belong with many Hospitals
* A Particular Service may be offered by many Doctors
* A Doctor may perform various Services

## Creating the first Model and Migration for Hospital

Use the model:generate command. This command requires two options:

* name: the name of the model;
* attributes: the list of model attributes.

Create a Model named Hospital

npx sequelize-cli model:generate --name Hospital--attributes Name:string,phoneNumber:string,establishedDate:date,isPrivate:boolean

This will:

* create a model file Hospitals in models folder;
* create a migration file with a name like XXXXXXXXXXXXXX-create-Hospitals.js in the migrations folder.

You can edit the skeleton migration file to add or edit the attributes in the table or change the datatype for the attributes.

**Note:** *CLI automatically creates an id attribute which is auto-incremented and also the primary key for the table.*

'use strict';

module.exports = {

up: (queryInterface, Sequelize) => {

return queryInterface.createTable(**'Hospitals'**, {

id: {

allowNull: false,

autoIncrement: true,

primaryKey: true,

type: Sequelize.INTEGER

},

name: {

type: Sequelize.STRING

},

address: {

type: Sequelize.STRING

},

phone\_number: {

type: Sequelize.INTEGER

},

established\_date: {

type: Sequelize.DATE

},

is\_private: {

type: Sequelize.BOOLEAN

},

createdAt: {

allowNull: false,

type: Sequelize.DATE

},

updatedAt: {

allowNull: false,

type: Sequelize.DATE

}

});

},

down: (queryInterface, Sequelize) => {

return queryInterface.dropTable(**'Hospitals'**);

}

};

As the Hospital table itself does not contain any foreign key relationships in it so the associations need not be defined in the model.

Now, to actually create the table Hospital in the database the model has to be migrated for which the following command is used

npx sequelize-cli db:migrate

This command will execute these steps:

* will create a table called SequelizeMeta in database which is used to record which migrations have run on the current database
* to look for any migration files which haven't run yet, check the SequelizeMeta table.
* creates a table called Hospitals with all columns as specified in its migration file.

To undo or revert the migration ie: delete the table recently created ,

The command db:migrate:undo, is used . This reverts the most recent migration.

To go back to the initial state db:migrate:undo:all command is used to undo all migrations .

To revert back to a specific migration a name is passed in the option

i:e db:migrate:undo --20200512131209-create-hospital.js

### 

## Creating the first Seed

To create a seed file which will add values to the **Hospital** table, Run the command

npx sequelize-cli seed:generate --name demo-Hospital

This command will create a seed file in the seeders folder. File name will look something like XXXXXXXXXXXXXX-demo-Hospital.js. It follows the same up / down semantics as the migration files.Now we should edit this file to insert a demo-Hospital to Hospital table.

"use strict";

module.exports = {

up: (queryInterface, Sequelize) => {

return (

queryInterface.bulkInsert(

"Hospitals",

[

name: "Patan Hospital ",

address: "Lalitpur",

phone\_number: 4565667,

established\_date: new Date(),

is\_private: true,

createdAt: new Date(),

updatedAt: new Date()

} ,

{

name: "City Hospital ",

address: "Bharatpur",

phone\_number: 4565667,

established\_date: new Date(),

is\_private: true,

createdAt: new Date(),

updatedAt: new Date()

},

],

{}

)

);

},

down: (queryInterface, Sequelize) => {

return queryInterface.bulkDelete("Hospitals", null, {});

},

};

The model and migration file for the Service is created in a similar fashion to that of Hospital. But since the Service table also contains foreign key relationship between Service and the Hospital table. The model.js file for service has to be edited such that the one to many relationship is defined.

'use strict';

module.exports = (sequelize, DataTypes) => {

const Service = sequelize.define('Services', {

hospital\_id: DataTypes.INTEGER,

service\_name: DataTypes.STRING,

cost: DataTypes.FLOAT

}, {});

Service.associate = function(models) {

// associations can be defined here

models.Hospital.hasMany(Service);

Service.belongsTo(models.Hospital)

};

return Service;

};

In the last step, since the seed file has already been created but it's still not committed to the database. A simple command has to be run.

npx sequelize-cli db:seed:all

To run a specific seed however for example of the hospital.

npx sequelize-cli db:seed --seed 20200512131810-Hospital

**Note:**running the db:seed:all *command is not recommended after multiple tables have been created as this will lead to multiple copies of the same data being stored in the database.*

## Undoing Seeds

To undo the seed and delete the data from the database and revert back to its original state the following commands are used

For undoing the most recent seed:

npx sequelize-cli db:seed:undo

For undoing a specific seed:

npx sequelize-cli db:seed:undo --seed name-of-seed-as-in-data

For undoing all seeds:

npx sequelize-cli db:seed:undo:all

**To add more columns to a Table**

For example to add two new columns to the Services table as description and departments**.**

* Generate a new migration file using the command

npx sequelize-cli migration:generate --name addcolumService

* Edit the migration file, if two or more column are to be added use transaction to ensure that all instructions are successfully executed or rolled back in case of failure

**"use strict";**

**module.exports = {**

**up: (queryInterface, Sequelize) => {**

**return queryInterface.sequelize.transaction((t) => {**

**return Promise.all([**

**queryInterface.addColumn(**

**"Services",**

**"service\_name",**

**{**

**type: Sequelize.DataTypes.STRING,**

**},**

**{ transaction: t }**

**),**

**queryInterface.addColumn(**

**"Services",**

**"cost",**

**{**

**type: Sequelize.DataTypes.FLOAT,**

**},**

**{ transaction: t }**

**),**

**]);**

**});**

**},**

**down: (queryInterface, Sequelize) => {**

**return queryInterface.sequelize.transaction((t) => {**

**return Promise.all([**

**queryInterface.removeColumn("Services", "service\_name", {**

**transaction: t,**

**}),**

**queryInterface.removeColumn("Services", "cost", { transaction: t }),**

**]);**

**});**

**},**

**};**

* **To migrate the changes in the database use the command**npx sequelize-cli db:migrate

**To update an existing column in a Table**

For example, To update the “descriptions” column in the Hospital Table to “website”

* Generate a new migration file using the command

npx sequelize-cli migration:generate --name updatecolumnHospital

* Edit the newly created migration file to update the column by manipulating queryInterface

'use strict';

module.exports = {

up: (queryInterface, Sequelize) => {

return queryInterface.renameColumn('Hospitals', 'descriptions', 'website');

},

down: (queryInterface, Sequelize) => {

return queryInterface.renameColumn('Hospitals', 'descriptions', 'website');

}

};

**Note:** *Update the model file for Hospital to add the updated column name and remove the column before the update.*

'use strict';

module.exports = (sequelize, DataTypes) => {

const Hospital = sequelize.define('Hospitals', {

name: DataTypes.STRING,

address: DataTypes.STRING,

phone\_number: DataTypes.INTEGER,

established\_date: DataTypes.DATE,

is\_private: DataTypes.BOOLEAN,

website: DataTypes.STRING,

departments: DataTypes.STRING,

}, {});

Hospital.associate = function(models) {

};

return Hospital;

};

* **To migrate the changes in the database use the command**npx sequelize-cli db:migrate

In a similar manner the table for Doctor, DoctorService, HospitalService and Category was also created as per the requirement of the Data Model.

The different list of commands forCLI are:

npx sequelize-cli db:migrate:status List the status of all migrations

npx sequelize-cli db:migrate:undo Reverts a migration

npx sequelize-cli db:migrate:undo:all Revert all migrations seed

npx sequelize-cli db:seed Run specific seeder

npx sequelize-cli db:seed:undo Deletes data from the database

npx sequelize-cli db:seed:all Run every seeder

npx sequelize-cli db:seed:undo:all Deletes data from the database

npx sequelize-cli db:create Create database specified by configuration

npx sequelize-cli db:drop Drop database specified by configuration

npx sequelize-cli init Initializes project

npx sequelize-cli init:config Initializes configuration

npx sequelize-cli init:migrations Initializes migrations

npx sequelize-cli init:models Initializes models

npx sequelize-cli init:seeders Initializes seeders

npx sequelize-cli migration:generate Generates a new migration file

[aliases: migration:create] npx sequelize-cli model:generate Generates a model and its migration [aliases: model:create] npx sequelize-cli seed:generate Generates a