Contents

1. Introduction

This project is intended for refactoring of existing database for LibreHealth EHR application (https://github.com/LibreHealthIO/LibreEHR). It removes some of the tables which are not used in present application like icd9 code related tables, documents related tables, etc. Also it populates database with random data (as real as possible) which is used for testing database before actual run on real data. It can also be used while adding new feature before implementing that in main code base.

2. Old Schema v/s New Schema

There were flaws in old schema which is rectified in new one. Major improvements include:

- **Lack of Relationship :** In old schema, there is no proper relationship between tables and all the linking is maintained in code. In new schema, tables are properly linked with foreign keys.

- **Inappropriate Field Type :** In old schema some fields were inappropriate, like integer is used in place of boolean, varchar is used in place of json type. This is taken care of in new schema.

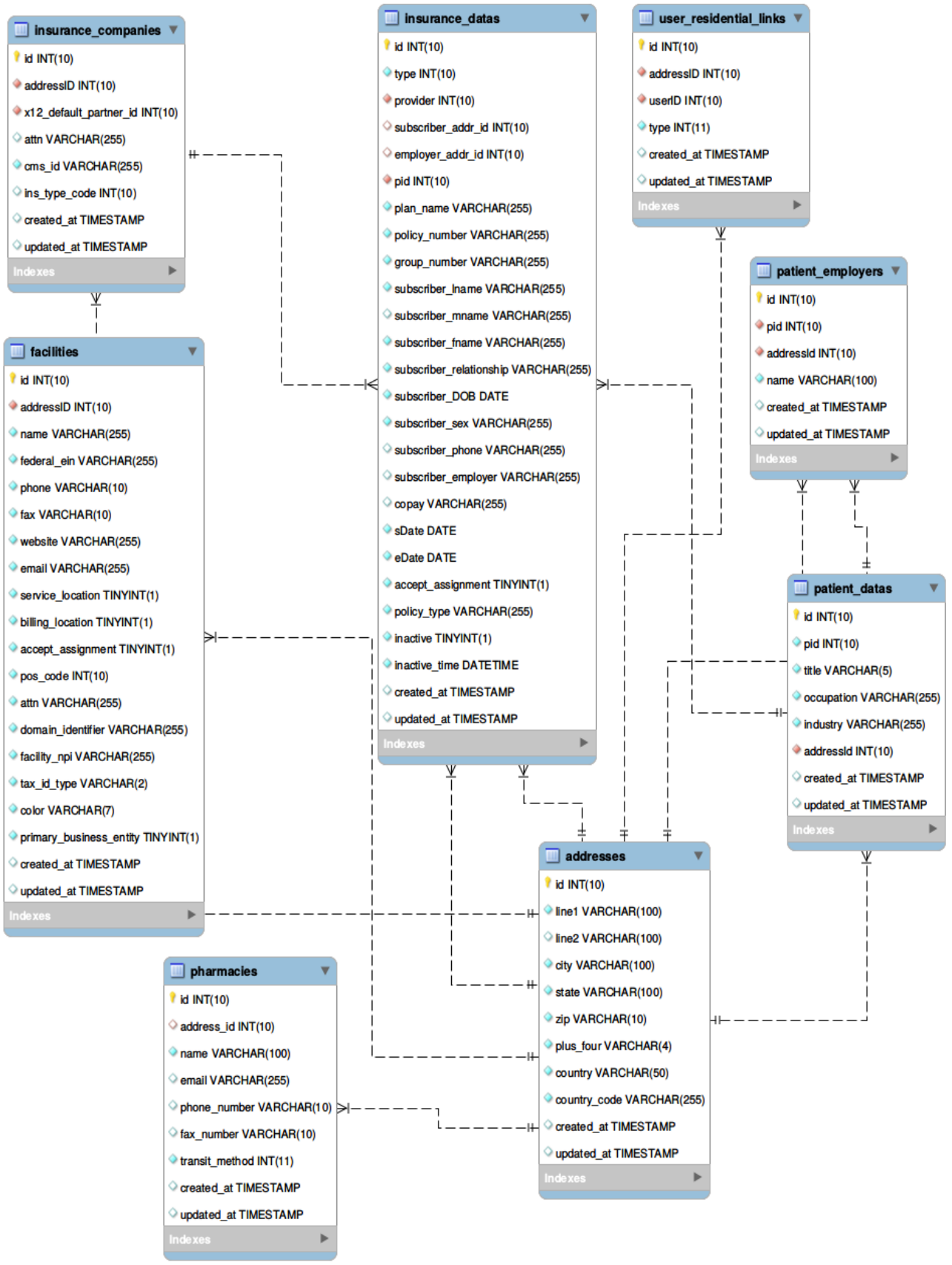
- **Lack of Normalization :** Old schema lacked normalization, i.e., tables were not normalized properly. This is taken care of in new schema and every table is broken to its lowest possible form and is properly linked.

- **Lack of Comments :** This may not be of much use as seen from technical perspective, but having table comment gives proper information of field and their purpose. This has been taken care of in new schema.

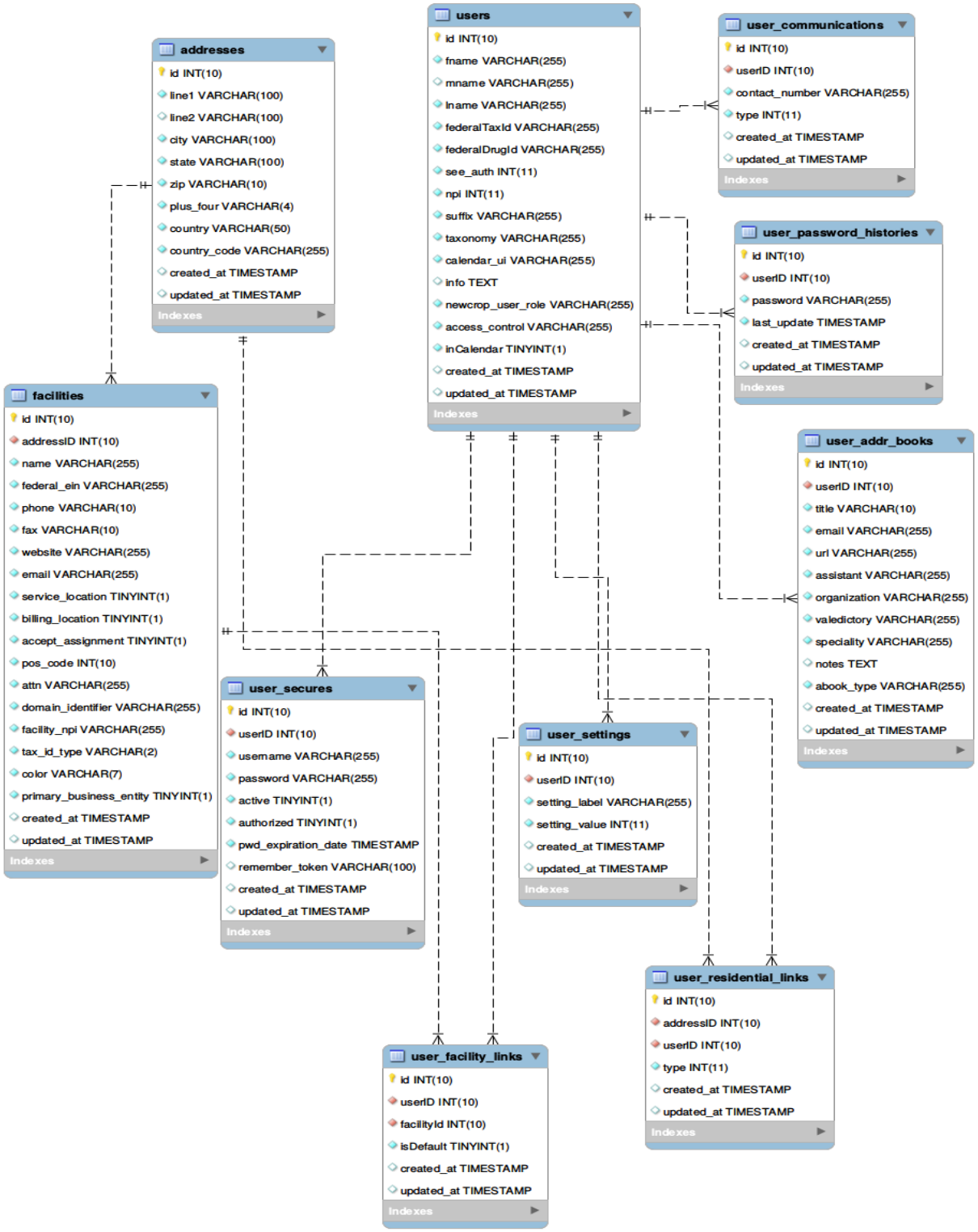
3. EER Diagrams and Table Description

EER Diagrams of important tables and their description are :

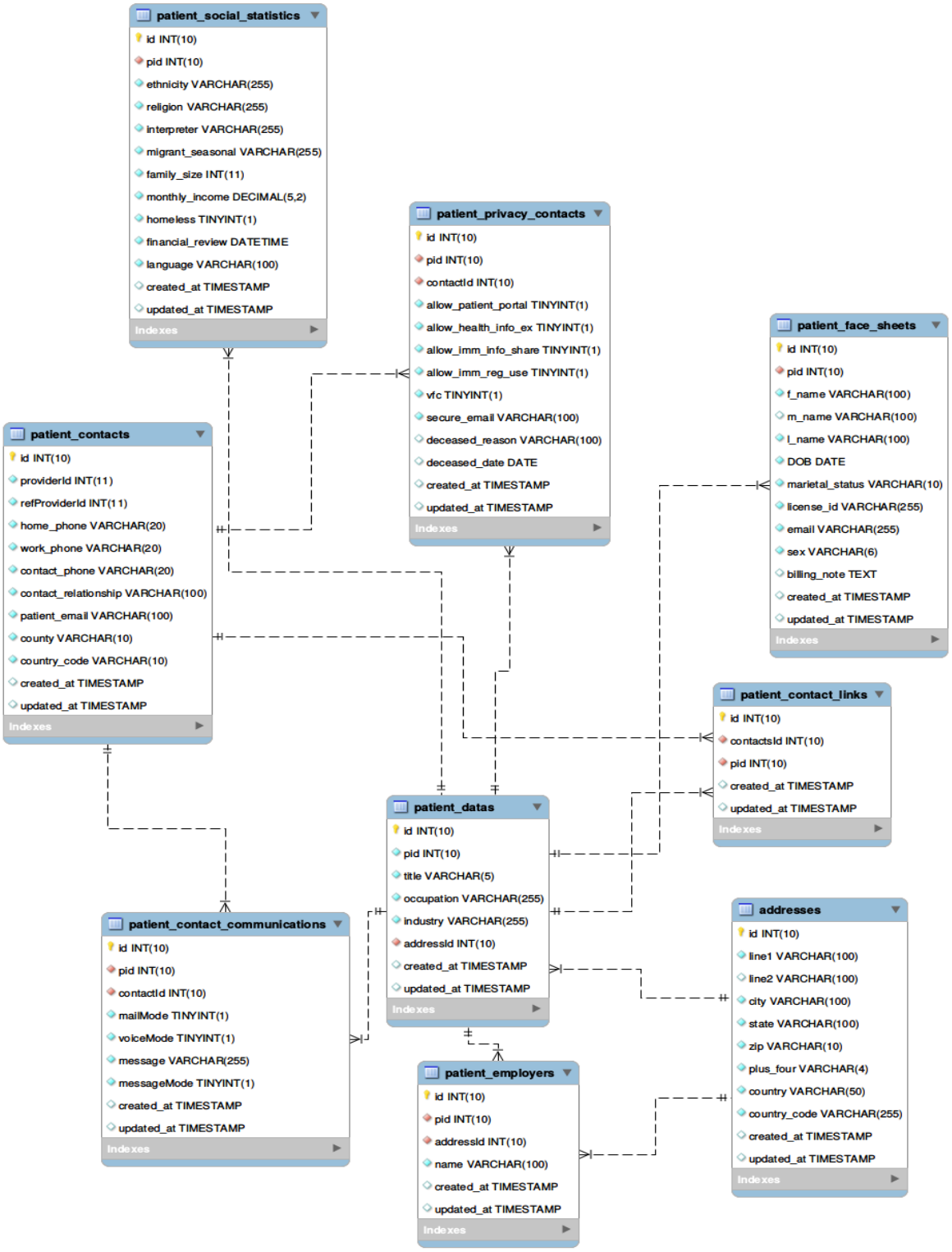
- **addresses :** It contains addresses of patients, users, insurance companies, facilities and employers. Its relationship with other tables can be shown as :


insurance_companies
id INT(10)
addressID INT(10)
x12_default_partner_id INT(10)
attn VARCHAR(255)
cms_id VARCHAR(255)
ins_type_code INT(10)
created_at TIMESTAMP
updated_at TIMESTAMP
Indexes
insurance_datas
id INT(10)
type INT(10)
provider INT(10)
subscriber_addr_id INT(10)
employer_addr_id INT(10)
pid INT(10)
plan_name VARCHAR(255)
policy_number VARCHAR(255)
group_number VARCHAR(255)
subscriber_lname VARCHAR(255)
subscriber_mname VARCHAR(255)
subscriber_fname VARCHAR(255)
subscriber_relationship VARCHAR(255)
subscriber_DOB DATE
subscriber_sex VARCHAR(255)
subscriber_phone VARCHAR(255)
subscriber_employer VARCHAR(255)
copay VARCHAR(255)
sDate DATE
eDate DATE
accept_assignment TINYINT(1)
policy_type VARCHAR(255)
inactive TINYINT(1)
inactive_time DATETIME
created_at TIMESTAMP
updated_at TIMESTAMP
Indexes
user_residential_links
id INT(10)
addressID INT(10)
userID INT(10)
type INT(11)
created_at TIMESTAMP
updated_at TIMESTAMP
Indexes
patient_employers
id INT(10)
pid INT(10)
addressId INT(10)
name VARCHAR(100)
created_at TIMESTAMP
updated_at TIMESTAMP
Indexes
facilities
id INT(10)
addressID INT(10)
name VARCHAR(255)
federal_ein VARCHAR(255)
phone VARCHAR(10)
fax VARCHAR(10)
website VARCHAR(255)
email VARCHAR(255)
service_location TINYINT(1)
billing_location TINYINT(1)
accept_assignment TINYINT(1)
pos_code INT(10)
attn VARCHAR(255)
domain_identifier VARCHAR(255)
facility_npi VARCHAR(255)
tax_id_type VARCHAR(2)
color VARCHAR(7)
primary_business_entity TINYINT(1)
created_at TIMESTAMP
updated_at TIMESTAMP
Indexes
patient_datas
id INT(10)
pid INT(10)
title VARCHAR(5)
occupation VARCHAR(255)
industry VARCHAR(255)
addressId INT(10)
created_at TIMESTAMP
updated_at TIMESTAMP
Indexes
addresses
id INT(10)
line1 VARCHAR(100)
line2 VARCHAR(100)
city VARCHAR(100)
state VARCHAR(100)
zip VARCHAR(10)
plus_four VARCHAR(4)
country VARCHAR(50)
country_code VARCHAR(255)
created_at TIMESTAMP
updated_at TIMESTAMP
Indexes
pharmacies
id INT(10)
address_id INT(10)
name VARCHAR(100)
email VARCHAR(255)
phone_number VARCHAR(10)
fax_number VARCHAR(10)
transit_method INT(11)
created_at TIMESTAMP
updated_at TIMESTAMP
Indexes

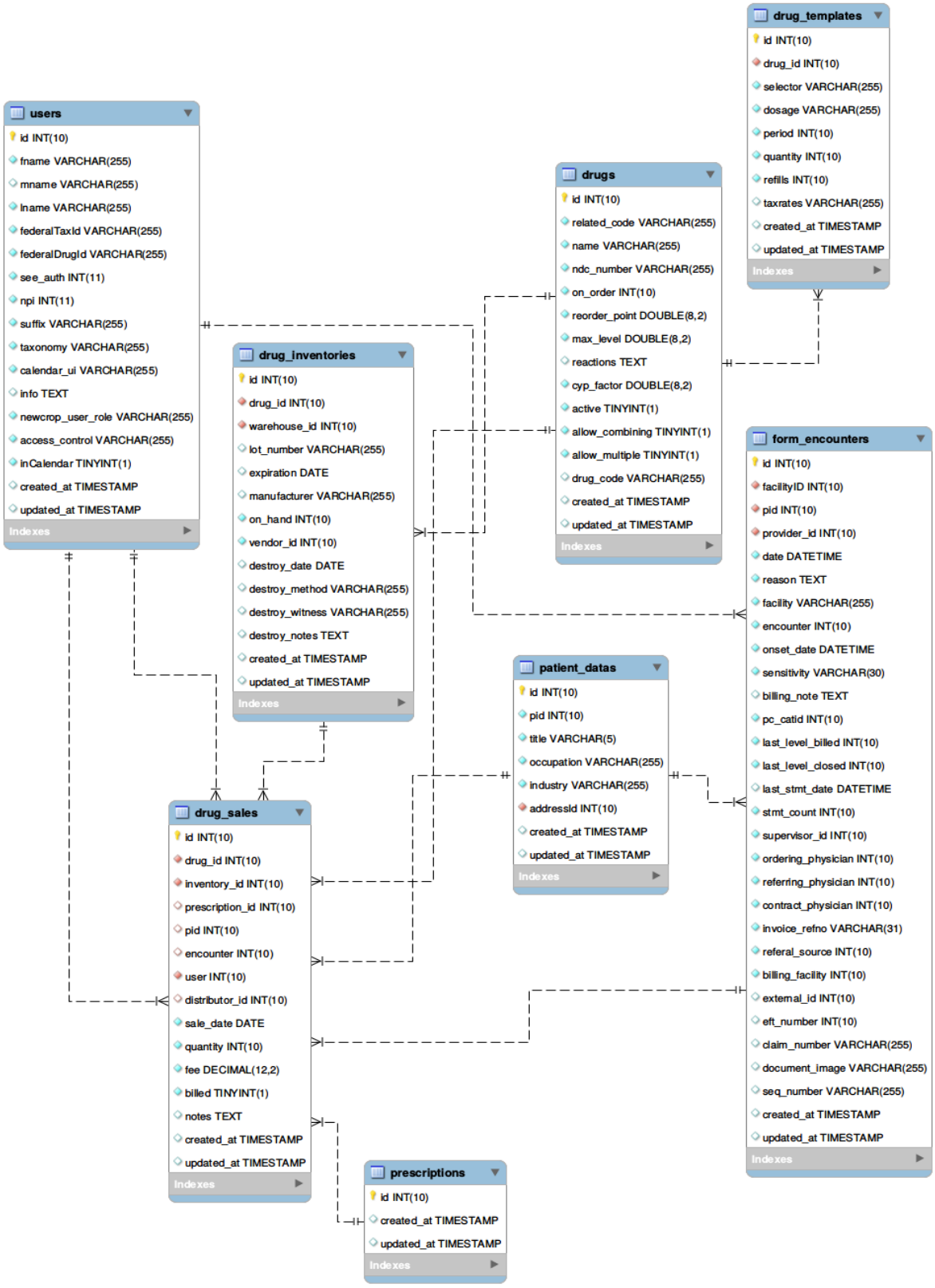
- **users :** In old schema, users table contain all related to user, but now it has been broken down into few smaller tables, to store information related to it only. For example, *user_residential_link* contains link of users with its residential info. Also users table is related to many other administrative tables, like facilities, documents, form_encounters, etc. EER diagram of user module can be shown as :



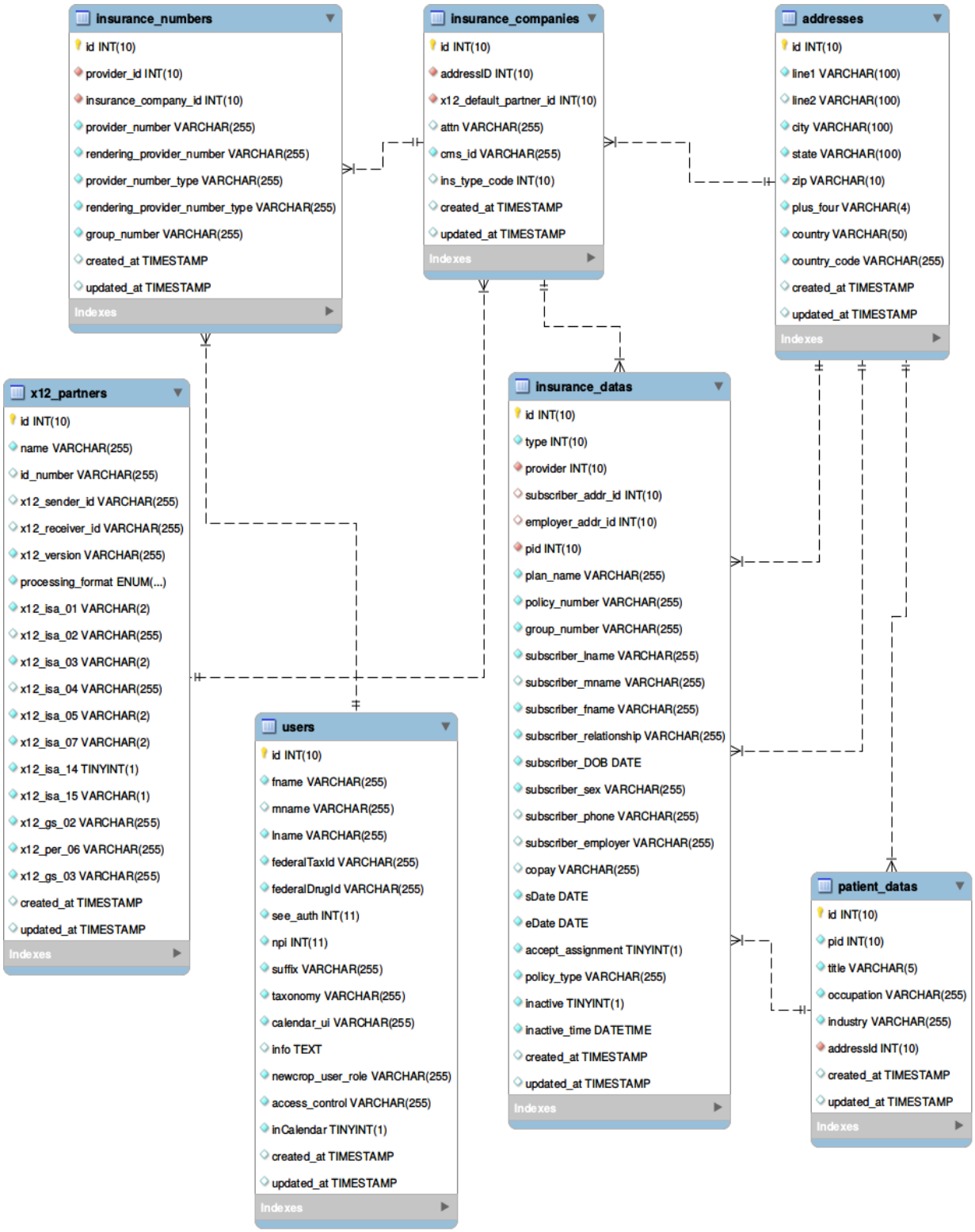
- **patients :** Patient is an important module in LibreHealth EHR application. It is linked with all patient related tables like, *form_encounters, forms_*, dated_reminders, doccuments, etc.* Its EER diagram can be shown as :



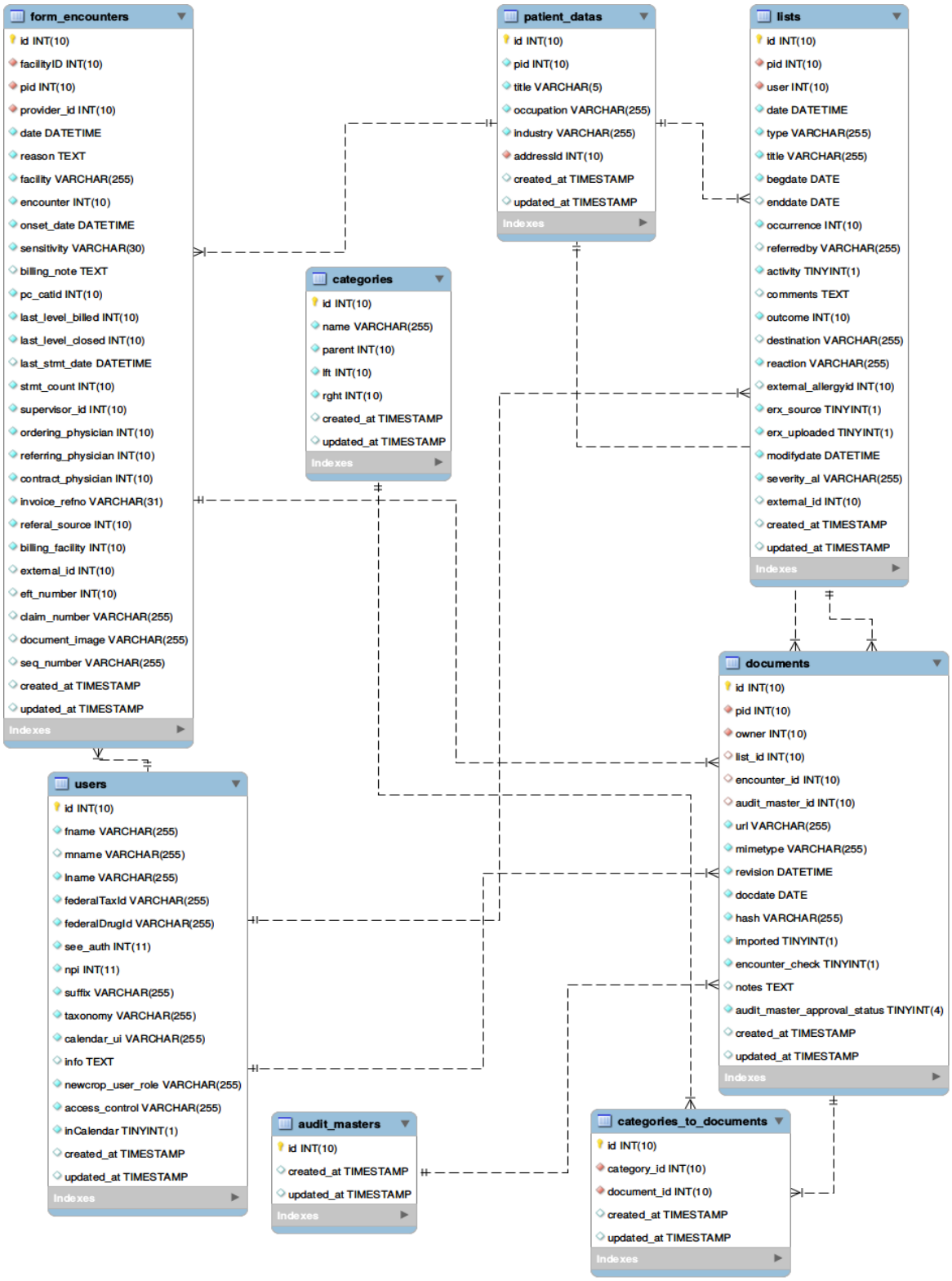
- **drugs :** This contains all the information related to drugs and inventory. It can be shown as :



- **insurance :** This contains all information related to insurance companies and insurance data. It can be shown as :



- **documents :** This contains related to patient documents. It is linked with categories. Its ER diagram is as follows :



These are main modules of EHR Application. For complete reference to tables, look at ***/LibreLaravel/database/migrations***. It contains uses of table, meaning of each column, and from UI, how it is related.

4. Seeding and Fake Data Generation

This is second part of project. This creates random data for tables and fill it. Laravel comes integrated with Faker, a library to fill tables with fake data. In Laravel, we need to write model factories to seed the database. There is alternative for this method but it creates one record at a time, while using model factories, bulk amount of data can be generated at one go.

4.1. Writing Model Factories : Model factory is located in ***/LibreLaravel/database/factories***. A simple example to write model factory is :

```php
/** @var \Illuminate\Database\Eloquent\Factory $factory */
$factory->define(App\Address::class, function (Faker\Generator $faker) {

    return [
        'line1' => $faker->streetName,
        'line2' => $faker->streetAddress,
        'city' => $faker->city,
        'state' => $faker->state,
        'zip' => $faker->postcode,
        'plus_four' => str_random(4),
        'country' => $faker->country,
        'country_code' => $faker->countryCode,
    ];
});
```

4.2. Writing Seeders : Seeders are located in ***/LibreLaravel/database/seeds.*** To create seed run command **: *php artisan make:seeder <Name Of Seeder Class>*.** It calls created model factory to created fake data. To seed table run command as : ***php artisan db:seed*** *or* ***php artisan db:seed --class=<Name of created seeder>.***

Example of address table seeder class is :

```php
<?php

use Illuminate\Database\Seeder;

class AddressTableSeeder extends Seeder
{
    /**
     * Run the database seeds.
     * Seed the Address Table.
     * @return void
     * @author Priyanshu Sinha <pksinha217@gmail.com>
     */

    public function run()
    {
        factory(App\Address::class, 25)->create();
    }
}
```

5. Making Table Schema : Migration

To generate table schema along with models, run command ***: php artisan make:model -m <TableNameInSingularForm>.*** It will create migration file with timestamp prefixed with it. This is so because when migrations are executed, the file which is created first will be migrated first. It is not advisable to alter with name of migration file, else it will not work in proper manner. This is in accordance with schema design, as we first create the base tables, and then add on to it.

Example of migration file is :

```php
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateAddressesTable extends Migration
{
    /**
     * Run the migrations.
     * Create addresses table.
     * It will store the addresse.
     * @author Priyanshu Sinha <pksinha217@gmail.com>
     * @return void
     */
    public function up()
    {
        Schema::create('addresses', function (Blueprint $table) {
            $table->increments('id')->comment = "Autoincrement Primary Key.";
            $table->string('line1', 100)->comment = "Line 1 of address.";
            $table->string('line2', 100)->nullable()->comment = "Line 2 of address. Optional";
            $table->string('city', 100)->comment = "city";
            $table->string('state', 100)->comment = "state";
            $table->string('zip', 10)->comment = "zip code";
            $table->string('plus_four', 4)->comment = "plus four code. US specific thing";
            $table->string('country', 50)->comment = "Country";
            $table->string('country_code')->comment = "Country Code";
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('addresses');
    }
}
```

To execute the created migration run command : ***php artisan migrate.*** Sometimes we need to make changes in schema, for this run command : ***php artisan migrate:refresh.*** There may be need of dumping autoload files, so for this run command : ***composer dump-autoload.***

More about model factory , seeds and migration with various cases can be found in source code.

6. Conclusion

This project is intended to refactor all tables. But some tables are still not refactored like *payment related tables, rules related tables and acl related tables.* This needs to be done as it required UI revamping, and accordingly its table structure will be designed later.

This project introduces laravel (just a beginning), and it can be extended further to have the whole application on Laravel Framework. As the application grows new schema can be generated accordingly. It is easy to maintain as we don't have to write raw query for table creation.

Future work includes working on models to have real query which will be followed by UI.