*8. До наявної no-sql DB необхідно розробити структуру бази даних з використанням SQL (PostgreSQL) для чатів, використовуючи наявну БД як еталон. Необхідно мінімізувати наслідки після міграції сервера з no-sql на sql (наприклад, імена стовпців мають бути однаковими за можливості). Зміни, які впливатимуть на таблицю користувачів, необхідно модифікувати строго за допомогою SQL(ALTER TABLE users..., etc...). Результати роботи необхідно залити на GIT у вигляді окремого SQL файлу. Надати схеми UML у вигляді скріншота з усіма можливими відношеннями (не забути прив’язати таблицю юзерів)*  
  
  
При перетворенні існуючої бази даних Mongo на PostgreSQL для підтримки реляційної цілісності необхідно додати до існуючої реляційної бази сутності, відповідні no-sql колекцій документів: Conversations, Messages, Catalogs

та створити звязки

між користувачами (Users) та чатами (Conversations) багато-до-багатьох

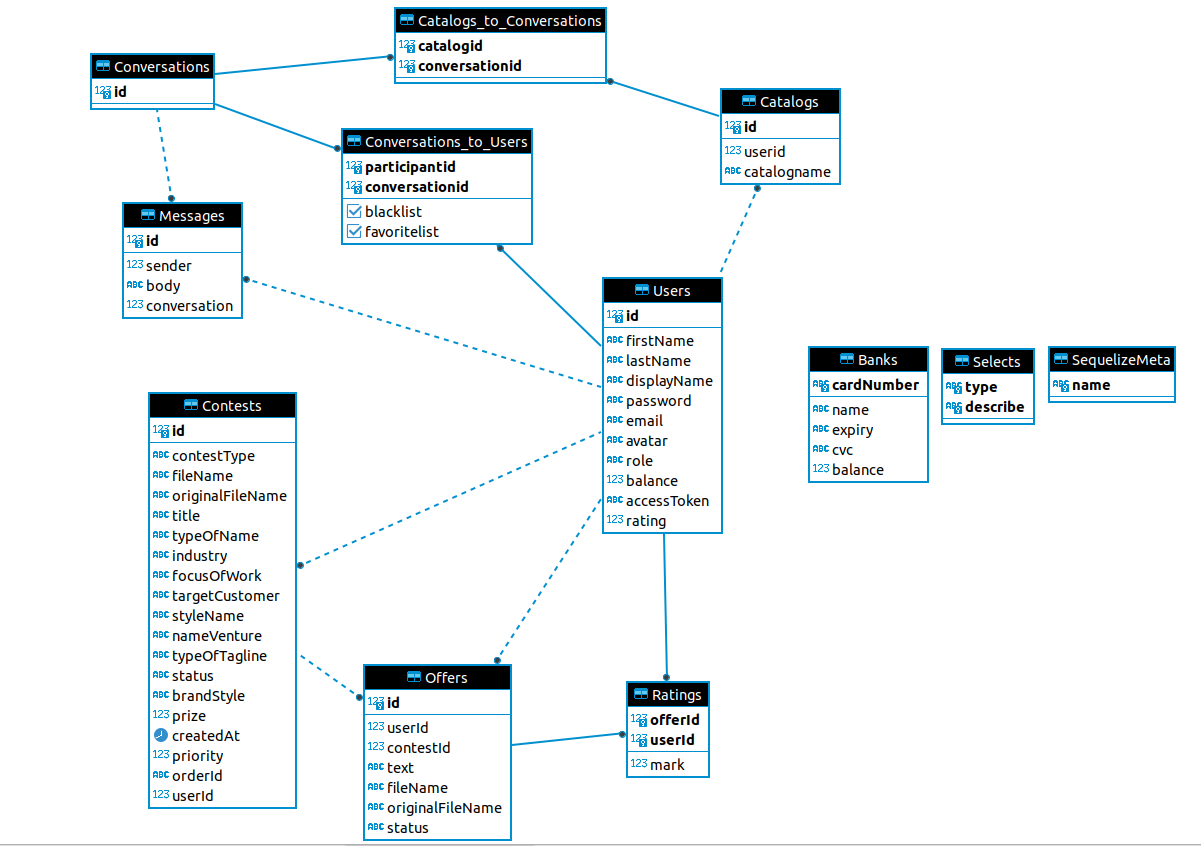
між користувачами (Users) та каталогами (Catalogs) один-до-багатьох,

між користувачами (Users) та повідомленнями (Messages) один-до-багатьох,

між чатами (Conversations) та повідомленнями (Messages) один-до-багатьох,

між чатами (Conversations) та каталогами (Catalogs) багато -до-багатьох

Таким чином, реляційна база даних після додавання сутностей, що відповідають за «чати» має виглядати наступним чином



І відповідний код SQL

CREATE TABLE "Conversations"(

"id" serial PRIMARY KEY

);

CREATE TABLE "Conversations\_to\_Users" (

participantId integer NOT NULL REFERENCES "Users"(id),

conversationId integer NOT NULL REFERENCES "Conversations"(id),

blackList boolean NOT NULL,

favoriteList boolean NOT NULL,

PRIMARY KEY (participantId, conversationId)

);

CREATE TABLE "Messages"(

"id" serial PRIMARY KEY,

sender integer NOT NULL REFERENCES "Users"(id),

body varchar(200) NOT NULL,

conversation integer NOT NULL REFERENCES "Conversations"(id)

);

CREATE TABLE "Catalogs"(

"id" serial PRIMARY KEY,

userId integer NOT NULL REFERENCES "Users"(id),

catalogName varchar(200) NOT NULL

);

CREATE TABLE "Catalogs\_to\_Conversations" (

catalogId integer NOT NULL REFERENCES "Catalogs"(id),

conversationId integer NOT NULL REFERENCES "Conversations"(id),

PRIMARY KEY (catalogId, conversationId)

);

В цьому випадку при розробці Sequelize моделей та міграцій необхідно створювати модель «Conversations\_to\_Users», оскільки відповідна сутність містить додаткові поля – blackList, favoriteList  
а для «Catalogs\_to\_Conversations» потрібна буде лише міграція

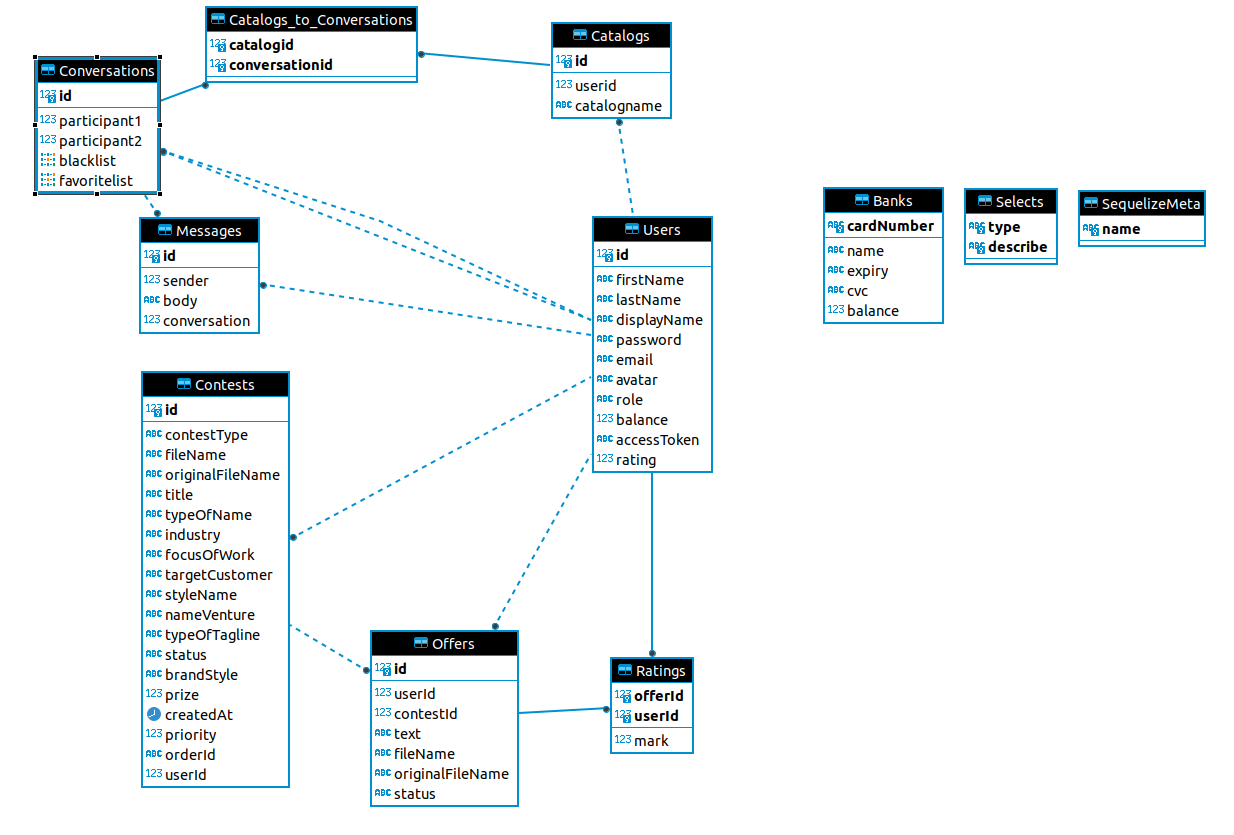
**АЛЕ**

Оскільки в нашому застосунку в чаті завжди лише два користувачі, я вирішила не створювати додаткову сутність

Conversations\_to\_Users, залишивши майже не змінною структуру no-sql «Conversations», прописавши учасників чатів як два посилання на сутність «Users» - participant1, participant2

Це значно полегшило роботу зі списками «улюблених» та «блокованих» користувачів, полегшило структуру та вкладеність запитів, і завдяки цьому я практично не переробляла супутню логіку на клієнті, просто привівши відповіді сервера до тих, які вже «навчився бачити» клієнт  
  
Тож, при реалізації пункту завдання 15 -   
Було додано лише 3 моделі та 4 міграції до існуючої Sequelize структури

UML модель відповідної схеми бази даних виглядатиме наступним чином



І вигляд SQL – запитів для створення відповідних сутностей в базах даних має наступний вигляд

CREATE TABLE "Conversations"(

"id" serial PRIMARY KEY,

participant1 integer NOT NULL REFERENCES "Users"(id),

participant2 integer NOT NULL REFERENCES "Users"(id),

blackList boolean[] NOT NULL,

favoriteList boolean[] NOT NULL

);

CREATE TABLE "Messages"(

"id" serial PRIMARY KEY,

sender integer NOT NULL REFERENCES "Users"(id),

body varchar(200) NOT NULL,

conversation integer NOT NULL REFERENCES "Conversations"(id)

);

CREATE TABLE "Catalogs"(

"id" serial PRIMARY KEY,

userId integer NOT NULL REFERENCES "Users"(id),

catalogName varchar(200) NOT NULL

);

CREATE TABLE "Catalogs\_to\_Conversations" (

catalogId integer NOT NULL REFERENCES "Catalogs"(id),

chatId integer NOT NULL REFERENCES "Conversations"(id)

PRIMARY KEY (catalogId, conversationId)

);