НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

(повна назва інституту/факультету)

КАФЕДРА інформатики та програмної інженерії

(повна назва кафедри)

**КУРСОВА РОБОТА**

з дисципліни «Бази даних»

(назва дисципліни)

на тему:\_\_\_\_\_База даних агенції нерухомості \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студента (ки) \_\_2\_\_ курсу \_ІП-15\_\_\_\_\_ групи

спеціальності 121 «Інженерія програмного забезпечення»

\_Мєшкова Андрія Ігоровича\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище та ініціали)

Керівник \_Марченко Олена Іванівна\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_ Оцінка ECTS \_\_\_\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

Київ – 2022 рік

**Національний технічний університет України**

**«Київський політехнічний інститут імені Ігоря Сікорського»**

Факультет Інформатики та обчислювальної техніки

(повна назва)

Кафедра Інформатики та програмної інженерії

(повна назва)

Дисципліна Бази даних

Курс \_\_2\_\_\_ Група \_\_ІП-15\_\_ Семестр \_\_\_3\_\_

# **ЗАВДАННЯ**

**НА КУРСОВУ РОБОТУ СТУДЕНТУ**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Мєшкова Андрія Ігоровича\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище, ім’я, по батькові)

1. Тема роботи\_\_\_\_\_ База даних агенції нерухомості\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

керівник роботи Марченко Олена Іванівна \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

2. Строк здачі студентом закінченої роботи «14» січня 2023 р.

3. Вихідні дані до роботи Завдання на розробку бази даних для агенства нерухомості

4. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці)

1) Аналіз предметного середовища

2) Побудова ER-моделі

3) Побудова реляційної схеми з ER-моделі

4) Створення бази даних, у форматі обраної системи управління базою даних

5) Створення користувачів бази даних

6) Імпорт даних з використанням засобів СУБД в створену базу даних

7) Створення мовою SQL запитів

8) Оптимізація роботи запитів

5. Перелік графічного матеріалу ( з точним зазначенням обов’язкових креслень)

6. Дата видачі завдання 08.11.2022

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва етапів виконання курсового проекту | Строк виконання етапів проекту | Примітка |
| 1 | Аналіз предметного середовища |  |  |
| 2 | Побудова ER-моделі |  |  |
| 3 | Побудова реляційної схеми з ER-моделі |  |  |
| 4 | Створення бази даних, у форматі обраної системи управління базою даних |  |  |
| 5 | Створення користувачів бази даних |  |  |
| 6 | Імпорт даних з використанням засобів СУБД в створену базу даних |  |  |
| 7 | Створення мовою SQL запитів |  |  |
| 8 | Оптимізація роботи запитів |  |  |
| 9 | Оформлення пояснювальної записки |  |  |
| 10 | Захист курсової роботи |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Студент \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_**Мєшков А. І**\_\_\_\_\_\_\_**

(підпис ) (прізвище та ініціали)

**Керівник роботи \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_**Марченко О.І.**\_\_\_\_\_\_\_\_**

(підпис ) (прізвище та ініціали)

ЗМІСТ

[**ЗАВДАННЯ** 2](#_Toc124553413)

[ВСТУП 5](#_Toc124553414)

[1 ОПИС ПРЕДМЕТНОГО СЕРЕДОВИЩА 6](#_Toc124553415)

[2 ПОСТАНОВКА ЗАВДАННЯ 8](#_Toc124553416)

[3 ПРОЕКТУВАННЯ БАЗИ ДАНИХ 10](#_Toc124553417)

[3.1 Побудова ER-моделі 10](#_Toc124553418)

[3.2 Даталогічна модель бази даних 13](#_Toc124553419)

[4 РЕАЛІЗАЦІЯ БАЗИ ДАНИХ 18](#_Toc124553420)

[5 РОБОТА З БАЗОЮ ДАНИХ 31](#_Toc124553421)

[ВИСНОВКИ 47](#_Toc124553422)

[ПЕРЕЛІК ПОСИЛАНЬ 48](#_Toc124553423)

[Додаток А Тексти SQL скриптів та результати їх роботи 49](#_Toc124553424)

[Додаток Б файли з даними, що були імпортовані в базу даних 66](#_Toc124553425)

# ВСТУП

На даний час у зв'язку з переходом на ринкову економіку, з'явилася потреба у створенні інформаційних систем, які для малого бізнесу дозволяють автоматизувати більшість інформаційних потоків, збільшити продуктивність та ефективність праці на підприємстві, тим самим домогтися необхідних конкурентних переваг у всіх галузях діяльності. На сьогодні агентство нерухомості мають затребуваність через різні міграції населення, яке шукає собі нове постійне житло або місце для бізнесу.

Метою курсової роботи є створення бази даних агентства нерухомості.

Для розв’язання задачі було розроблено базу даних, реалізовано її на мові MySQL за допомогою середовища MySQL Workbench.

# ОПИС ПРЕДМЕТНОГО СЕРЕДОВИЩА

Процес роботи сфери послуг включає в себе збереження певних даних. Клієнт, який шукає нерухомість звертається до агенції чи шукає оголошення на спільних інтернет порталах, листівках. Для подальшої праці з агентством клієнт залишає дані[2]:

1. ПІБ
2. Контактні дані – електронна адреса
3. Контактний телефон
4. Номер паспорту

Клієнт шукає оголошення, яке виставляє власник нерухомості та реєструє в системі тільки контактні дані та ПІБ.

Агенція повинна розмістити каталог оголошень на порталі нерухомості, як і інші агенції. Кожне майно записуються з параметрами[3]:

1. Точна адреса - вулиця
2. Глобальна адреса – місто
3. Почтовий індекс
4. Поверх(у багатоповерховому будинку, приватний будинок має 1)
5. Площа(у м2)
6. Кількість кімнат

Обираючи майно, для підписання договору клієнт знову звертається до агентства. Оголошення нерухомості розміщується на порталі різними агенціями. Кожне агентство має:

1. Назву
2. Адресу офісу
3. Ареал обслуги(країна, регіон, місто)
4. Контакти

Для прямого контакту для кожного клієнта призначається агент, який допоможе підписати договір. При зверненні до посередників з метою купівлі, продажу нерухомості, ріелтори нерідко наполягають на тому, щоб клієнт підписав з ними ексклюзивний контракт. У тексті цього документа зазвичай прописана конкретна сума, яку потрібно буде виплатити ріелтору за надання його послуг. Починається «переписка» агенції і клієнта за нерухомість, яку знайшов ріелтор[4]:

1. Дата переписки
2. Винагорода
3. URL(місце переписки, портал, сайт з оголошеннями, сайт агентства)
4. Статус оголошення(відкрите за замовчуванням)

Оброблюється пропозиція, підписується договір. У цьому договорі має бути зазначено: [1]:

1. Умови співпраці
2. Вартість оплати;
3. Термін дії договору

Після підписання договору оголошення повинно закритися, змінити статус «Прийнято».

У випадку оренди з клієнт підписує контракт з власником, зазначено атрибут Податок, або Депозит. Для виплати оренди створюється окрема сутність, де описується термін дії та виплати за місяць.

Дана система передбачає перевірку типу введених даних. Під час роботи система надає доступ з різними можливостями агенції/агенту, клієнту під час реєстрації особи та продавцю/власнику нерухомості для розміщення оголошення. Також передбачається збереження усіх даних необхідних для налагодження зв’язку між сторонами договору, і поліпшення роботи агенції.

Перевірка коректності інформації у базі даних може перевірятися зовнішніми джерелами, що не передбачені у даній роботі.

# ПОСТАНОВКА ЗАВДАННЯ

Для виконання поставленої мети потрібно виконати наступні завдання: побудувати ER-модель, побудувати реляційну схему, створити базу даних за допомогою обраної СУБД, створити користувачів бази даних, реалізувавши розроблену багато користувальницьку модель, врахувати збереження цілісності даних, імпортувати дані, оптимізувати роботу запитів.

Побудована база даних має містити не менше 10 таблиць, 20 DML запитів, містити збережені представлення, функції і процедури. Має містити тригери та генератори.

Процес роботи агенції нерухомості передбачає взаємодію 10 сутностей:

1. Клієнт
2. Майно
3. Агенція
4. Агент
5. Каталог
6. Пропозиція
7. Запит
8. Власник майна
9. Контракт оренди
10. Оренда

Клієнт – це одна особа, що заключає один або декілька договорів з агентством.

Майно – оголошення нерухомості для продажу або здачі в оренду.

Агенція – офіс підприємства чи одне підприємство серед декількох у випадку спільної Системи купівлі/продажу/оренди нерухомості.

Агент – особа прикріплена агенцією до клієнта для надання послуг, підписання договору.

Каталог – переписка клієнта з агентом, робота посередника для пошуку майна, особистий контракт з ріелтором, плата за послуги агенції.

Пропозиція – обробка, підписання договору на купівлю/продаж нерухомості.

Запит – запит клієнта на обране оголошення. Сутність-посередник між клієнтом і майном.

Власник майна – особа, яка дає оголошення на купівлю/оренду.

Контракт оренди – підписання контракту оренду між клієнтом і власником нерухомості.

Оренда – умови оренди: термін дії, оплата.

У системі існує три типи користувачів: Агент, Клієнт, Власник майна.

Варто передбачити перевірку вірності введеної дати народження перед записом нового або оновленням існуючого клієнта, агента, власника майна в базу даних

В системі має існувати набір тригерів видалення, додавання, оновлення даних. Після підписання контракту оренди чи продажу майна, оголошення з даною нерухомістю повинно бути закрите.

Для роботи користувачів передбачається створення представлень та процедур.

# ПРОЕКТУВАННЯ БАЗИ ДАНИХ

## 3.1 Побудова ER-моделі

Визначимо необхідний набір атрибутів для кожної сутності. Таблиця 3.1

Таблиця 3.1 - Таблиця сутностей

|  |  |
| --- | --- |
| Сутність | Опис сутності |
| Client | Містить інформацію про клієнта: імʼя, паспортні дані, контактний телефон, адресу електронної скриньки. |
| Property | Містить інформацію про майно: розташування(вулиця, місто, почтовий індекс), номер поверху, площа, кількість кімнат, тип нерухомості(оренда чи продаж), ключ власника. |
| Agency | Містить інформацію про офіс агенції: назва, адреса, ареал послуг, контакти. |
| Agent | Містить інформацію про агента – робітника агенції: ключ прикріпленого клієнта, ключ агенції. |
| Listing | Містить інформацію про роботу агента: ключі агента, клієнта, оголошення нерухомості, дата початку роботи, винагорода агента, джерело, статус замовлення. |
| Offer | Містить інформацію про пропозицію – договір купівлі: ключі власника, листування агента з клієнтом, термін дії договору(початок та кінець), ціна, валюта, умови продажу, статус замовлення. |
| Inquiry | Містить інформацію про запит, відповідь на оголошення: ключі клієнта та майна. |
| PropertyOwner | Містить інформацію про власника майна: імʼя, номер телефону. |
| Contract | Містить інформацію про контракт оренди: ідентифікатори власника та лістинга, податок. |
| Rent | Містить інформацію про умови оренди: термін дії оренди(початок, та кінець), помісячна ціна, ідентифікатор контракту. |

Згідно з сутностями були сформовано наступні зв’язки. Таблиця 3.2

Таблиця 3.2 - Таблиця зв’язків

|  |  |
| --- | --- |
| Сутність | Звʼязок |
| Property | Багато до 1 з PropertyOwner |
| Agent | Багато до 1 з Agency  Багато до 1 з Client |
| Listing | Багато до 1 з Agent  Багато до 1 з Client  1 до 1 з Property |
| Offer | Багато до 1 з PropertyOwner  1 до 1 з Listing |
| Inquiry | Багато до 1 з Client  Багато до 1 з Property |
| Contract | Багато до 1 з PropertyOwner  1 до 1 з Listing |
| Rent | 1 до 1 з Contract |

На рисунку 3.1 наведено спроектовану ER-модель за нотацією Баркера.

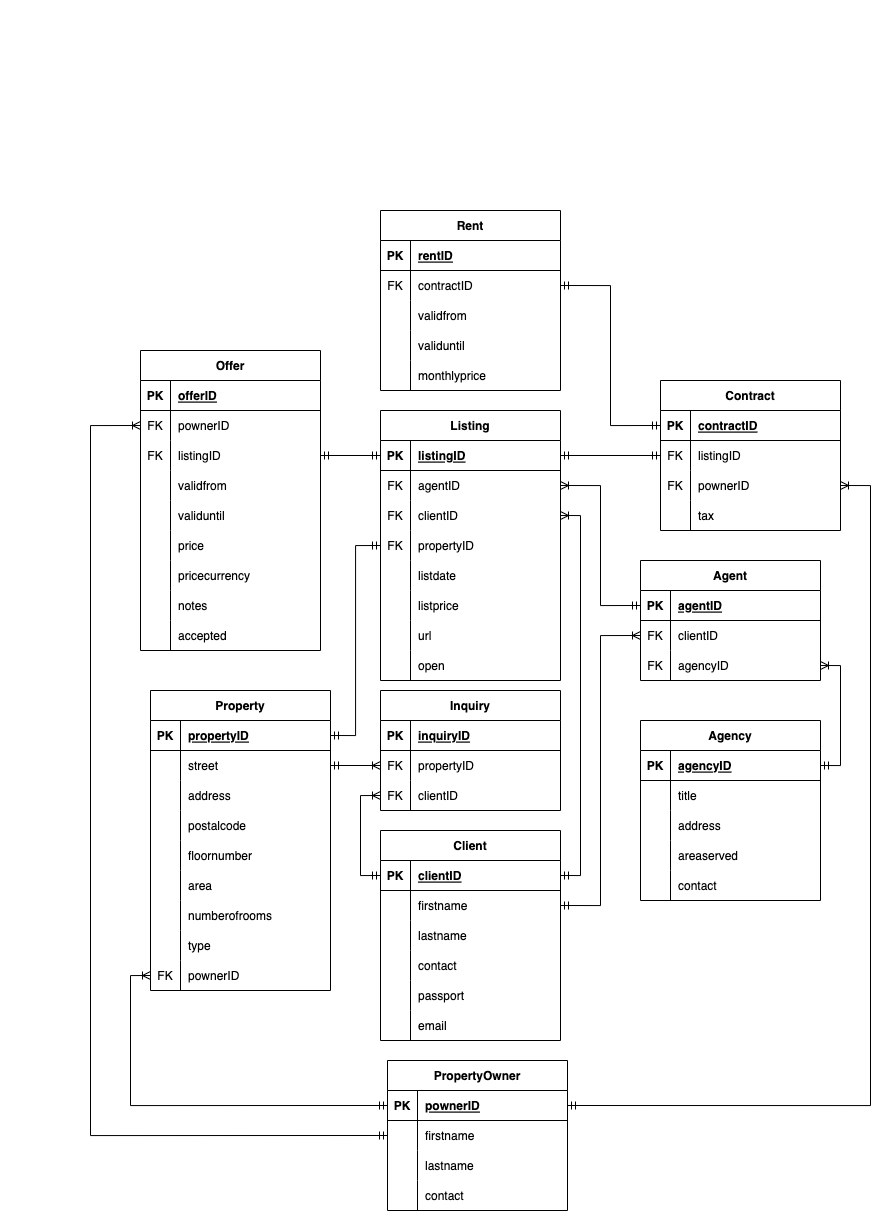


Рисунок 3.1 - ER діаграма

## 3.2 Даталогічна модель бази даних

Для побудови даталогічної моделі бази даних необхідно визначити типи даних для усіх атрибутів таблиць, зазначених в побудованій ER-моделі.

Відповідно моделей бази даних було розроблено таблиці. Опис таблиць представлено у таблицях 3.3-3.12

Таблиця 3.3 - Опис таблиці Client

|  |  |  |  |
| --- | --- | --- | --- |
| Назва таблиці | | | |
| Client | | | |
| Ключ | Імʼя атрибуту | Тип даних | Розмір |
| PK | clientID | INT | - |
| - | firstname | VARCHAR | 45 |
| - | lastname | VARCHAR | 45 |
| UQ | contact | VARCHAR | 15 |
| UQ | passport | VARCHAR | 9 |
| UQ | email | VARCHAR | 255 |

Таблиця 3.4 - Опис таблиці Property

|  |  |  |  |
| --- | --- | --- | --- |
| Назва таблиці | | | |
| Property | | | |
| Ключ | Імʼя атрибуту | Тип даних | Розмір |
| PK | propertyID | INT | - |
| - | street | VARCHAR | 45 |
| - | address | VARCHAR | 45 |
| - | postalcode | VARCHAR | 10 |
| - | floornumber | INT | - |
| - | area | VARCHAR | 10 |
| - | numberofrooms | INT | - |
| - | type | VARCHAR | 4 |
| FK | pownerID | INT | - |

Таблиця 3.5 - Опис таблиці Agency

|  |  |  |  |
| --- | --- | --- | --- |
| Назва таблиці | | | |
| Agency | | | |
| Ключ | Імʼя атрибуту | Тип даних | Розмір |
| PK | agencyID | INT | - |
| - | title | VARCHAR | 45 |
| - | address | VARCHAR | 45 |
| - | areaserved | VARCHAR | 45 |
| - | contact | VARCHAR | 15 |

Таблиця 3.6 - Опис таблиці Agent

|  |  |  |  |
| --- | --- | --- | --- |
| Назва таблиці | | | |
| Agent | | | |
| Ключ | Імʼя атрибуту | Тип даних | Розмір |
| PK | agentID | INT | - |
| FK | agencyID | INT | - |
| FK | clientID | INT | - |

Таблиця 3.7 - Опис таблиці Listing

|  |  |  |  |
| --- | --- | --- | --- |
| Назва таблиці | | | |
| Listing | | | |
| Ключ | Імʼя атрибуту | Тип даних | Розмір |
| PK | listingID | INT | - |
| FK | agentID | INT | - |
| FK | clientID | INT | - |
| FK | propertyID | INT | - |
| - | listdate | DATETIME | - |
| - | listprice | FLOAT | - |
| - | url | VARCHAR | 255 |
| - | open | TINYINT | - |

Таблиця 3.8 - Опис таблиці Offer

|  |  |  |  |
| --- | --- | --- | --- |
| Назва таблиці | | | |
| Offer | | | |
| Ключ | Імʼя атрибуту | Тип даних | Розмір |
| PK | offerID | INT | - |
| FK | pownerID | INT | - |
| FK | listingID | INT | - |
| - | validfrom | DATE | - |
| - | validuntil | DATETIME | - |
| - | price | FLOAT | - |
| - | pricecurency | VARCHAR | 3 |
| - | note | VARCHAR | 255 |
| - | accepted | TINYINT | - |

Таблиця 3.9 - Опис таблиці Inquiry

|  |  |  |  |
| --- | --- | --- | --- |
| Назва таблиці | | | |
| Inquiry | | | |
| Ключ | Імʼя атрибуту | Тип даних | Розмір |
| PK | inquiryID | INT | - |
| FK | propertyID | INT | - |
| FK | clientID | INT | - |

Таблиця 3.10 - Опис таблиці PropertyOwner

|  |  |  |  |
| --- | --- | --- | --- |
| Назва таблиці | | | |
| PropertyOwner | | | |
| Ключ | Імʼя атрибуту | Тип даних | Розмір |
| PK | pownerID | INT | - |
| - | firstname | VARCHAR | 45 |
| - | lastname | VARCHAR | 45 |
| UQ | contact | VARCHAR | 15 |

Таблиця 3.11 - Опис таблиці Contract

|  |  |  |  |
| --- | --- | --- | --- |
| Назва таблиці | | | |
| Contract | | | |
| Ключ | Імʼя атрибуту | Тип даних | Розмір |
| PK | contractID | INT | - |
| FK | pownerID | INT | - |
| FK | listingID | INT | - |
| - | tax | FLOAT | - |

Таблиця 3.12 - Опис таблиці Rent

|  |  |  |  |
| --- | --- | --- | --- |
| Назва таблиці | | | |
| Rent | | | |
| Ключ | Імʼя атрибуту | Тип даних | Розмір |
| PK | rentID | INT | - |
| FK | contactID | INT | - |
| - | validfrom | DATETIME | - |
| - | validuntil | DATETIME | - |
| - | monthlyprice | FLOAT | - |

На рисунку 3.2 наведено схематичну реалізацію даталогічної моделі бази даних з відображення звʼязків між таблицями.

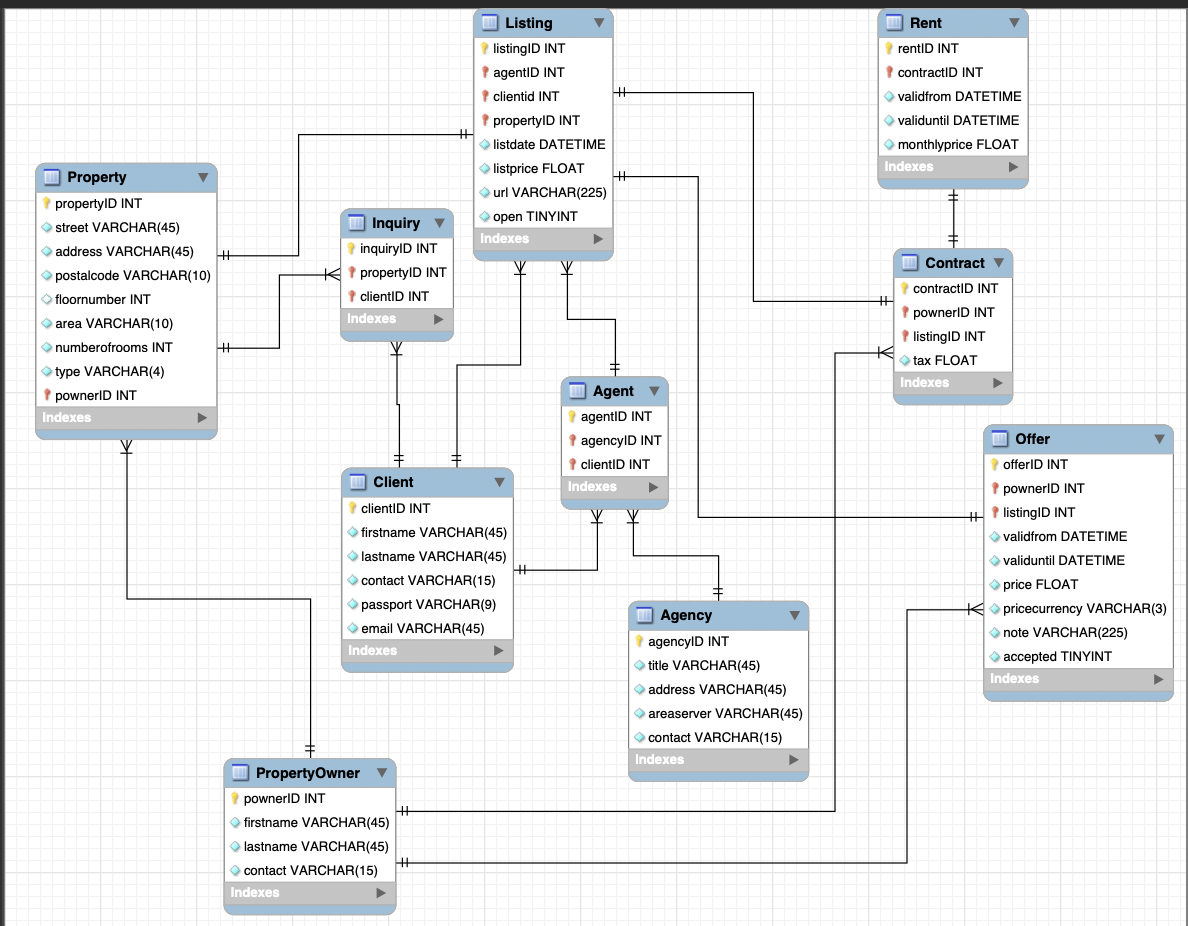


Рисунок 3.2 Даталогічна модель БД

Отже, на основі побудованої схеми сутність-звʼязок було розроблено даталогічну модель базу даних. Результуюча логічна схема бази даних представлена у вигляді таблиць із зазначенням їх атрибутів, типів даних та обраних первинних ключів.

# РЕАЛІЗАЦІЯ БАЗИ ДАНИХ

Програмну реалізацію виконано з використанням системи управління базою даних MySQL. Причиною вибору даної СУБД стала гнучкість роботи з нею, наявність великого інструментарію, швидкодія та популярність серед користувачів операційної системи macOS.

Особливістю даної СУБД є відсутність генераторів. Замість цього деякі атрибути будуть використовувати ауто-інкрементацію.

На основі моделі бази даних з Розділу 3 написано скрипти для створення таблиць.

DROP SCHEMA IF EXISTS `RealEstateAgency`;

CREATE SCHEMA IF NOT EXISTS `RealEstateAgency`;

USE `RealEstateAgency` ;

DROP TABLE IF EXISTS `RealEstateAgency`.`Client`;

CREATE TABLE IF NOT EXISTS `RealEstateAgency`.`Client` (

`clientID` INT NOT NULL AUTO\_INCREMENT,

`firstname` VARCHAR(45) NOT NULL,

`lastname` VARCHAR(45) NOT NULL,

`contact` VARCHAR(15) NOT NULL,

`passport` VARCHAR(9) NOT NULL,

`email` VARCHAR(45) NOT NULL,

PRIMARY KEY (`clientID`),

UNIQUE INDEX `passport\_UNIQUE` (`passport` ASC) VISIBLE,

UNIQUE INDEX `email\_UNIQUE` (`email` ASC) VISIBLE,

UNIQUE INDEX `contact\_UNIQUE` (`contact` ASC) VISIBLE)

ENGINE = InnoDB;

DROP TABLE IF EXISTS `RealEstateAgency`.`PropertyOwner`;

CREATE TABLE IF NOT EXISTS `RealEstateAgency`.`PropertyOwner` (

`pownerID` INT NOT NULL AUTO\_INCREMENT,

`firstname` VARCHAR(45) NOT NULL,

`lastname` VARCHAR(45) NOT NULL,

`contact` VARCHAR(15) NOT NULL,

PRIMARY KEY (`pownerID`),

UNIQUE INDEX `contact\_UNIQUE` (`contact` ASC) VISIBLE)

ENGINE = InnoDB;

DROP TABLE IF EXISTS `RealEstateAgency`.`Property`;

CREATE TABLE IF NOT EXISTS `RealEstateAgency`.`Property` (

`propertyID` INT NOT NULL AUTO\_INCREMENT,

`street` VARCHAR(45) NOT NULL,

`address` VARCHAR(45) NOT NULL,

`postalcode` VARCHAR(10) NOT NULL,

`floornumber` INT NULL,

`area` VARCHAR(10) DEFAULT 'm^2',

`numberofrooms` INT NOT NULL,

`type` VARCHAR(4) NOT NULL,

`pownerID` INT NOT NULL,

PRIMARY KEY (`propertyID`, `pownerID`),

INDEX `prop-pow\_idx` (`pownerID` ASC) VISIBLE,

CONSTRAINT `prop-pow`

FOREIGN KEY (`pownerID`)

REFERENCES `RealEstateAgency`.`PropertyOwner` (`pownerID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

DROP TABLE IF EXISTS `RealEstateAgency`.`Agency`;

CREATE TABLE IF NOT EXISTS `RealEstateAgency`.`Agency` (

`agencyID` INT NOT NULL AUTO\_INCREMENT,

`title` VARCHAR(45) NOT NULL,

`address` VARCHAR(45) NOT NULL,

`areaserver` VARCHAR(45) NOT NULL,

`contact` VARCHAR(15) NOT NULL,

PRIMARY KEY (`agencyID`))

ENGINE = InnoDB;

DROP TABLE IF EXISTS `RealEstateAgency`.`Agent`;

CREATE TABLE IF NOT EXISTS `RealEstateAgency`.`Agent` (

`agentID` INT NOT NULL AUTO\_INCREMENT,

`agencyID` INT NOT NULL,

`clientID` INT NOT NULL,

PRIMARY KEY (`agentID`, `agencyID`, `clientID`),

INDEX `agent-cl\_idx` (`clientID` ASC) VISIBLE,

INDEX `agent-agency\_idx` (`agencyID` ASC) VISIBLE,

CONSTRAINT `agent-agency`

FOREIGN KEY (`agencyID`)

REFERENCES `RealEstateAgency`.`Agency` (`agencyID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `agent-cl`

FOREIGN KEY (`clientID`)

REFERENCES `RealEstateAgency`.`Client` (`clientID`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

DROP TABLE IF EXISTS `RealEstateAgency`.`Listing`;

CREATE TABLE IF NOT EXISTS `RealEstateAgency`.`Listing` (

`listingID` INT NOT NULL AUTO\_INCREMENT,

`agentID` INT NOT NULL,

`clientid` INT NOT NULL,

`propertyID` INT NOT NULL,

`listdate` DATETIME NOT NULL,

`listprice` FLOAT NOT NULL,

`url` VARCHAR(225) NOT NULL,

`open` TINYINT DEFAULT TRUE,

PRIMARY KEY (`listingID`, `agentID`, `propertyID`, `clientid`),

INDEX `list-agent\_idx` (`agentID` ASC) VISIBLE,

INDEX `list-cl\_idx` (`clientid` ASC) VISIBLE,

INDEX `list-prop\_idx` (`propertyID` ASC) VISIBLE,

CONSTRAINT `list-agent`

FOREIGN KEY (`agentID`)

REFERENCES `RealEstateAgency`.`Agent` (`agentID`)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `list-cl`

FOREIGN KEY (`clientid`)

REFERENCES `RealEstateAgency`.`Client` (`clientID`)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `list-prop`

FOREIGN KEY (`propertyID`)

REFERENCES `RealEstateAgency`.`Property` (`propertyID`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

DROP TABLE IF EXISTS `RealEstateAgency`.`Offer`;

CREATE TABLE IF NOT EXISTS `RealEstateAgency`.`Offer` (

`offerID` INT NOT NULL AUTO\_INCREMENT,

`pownerID` INT NOT NULL,

`listingID` INT NOT NULL,

`validfrom` DATE NOT NULL,

`validuntil` DATE NOT NULL,

`price` FLOAT NOT NULL,

`pricecurrency` VARCHAR(3) NOT NULL,

`note` VARCHAR(225) NOT NULL,

`accepted` TINYINT DEFAULT FALSE,

PRIMARY KEY (`offerID`, `listingID`, `pownerID`),

INDEX `off-pow\_idx` (`pownerID` ASC) VISIBLE,

INDEX `off-list\_idx` (`listingID` ASC) VISIBLE,

CONSTRAINT `off-pow`

FOREIGN KEY (`pownerID`)

REFERENCES `RealEstateAgency`.`PropertyOwner` (`pownerID`)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `off-list`

FOREIGN KEY (`listingID`)

REFERENCES `RealEstateAgency`.`Listing` (`listingID`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

DROP TABLE IF EXISTS `RealEstateAgency`.`Inquiry`;

CREATE TABLE IF NOT EXISTS `RealEstateAgency`.`Inquiry` (

`inquiryID` INT NOT NULL AUTO\_INCREMENT,

`propertyID` INT NOT NULL,

`clientID` INT NOT NULL,

PRIMARY KEY (`inquiryID`, `clientID`, `propertyID`),

INDEX `in-prop\_idx` (`propertyID` ASC) VISIBLE,

INDEX `in-cl\_idx` (`clientID` ASC) VISIBLE,

CONSTRAINT `in-prop`

FOREIGN KEY (`propertyID`)

REFERENCES `RealEstateAgency`.`Property` (`propertyID`)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `in-cl`

FOREIGN KEY (`clientID`)

REFERENCES `RealEstateAgency`.`Client` (`clientID`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

DROP TABLE IF EXISTS `RealEstateAgency`.`Contract`;

CREATE TABLE IF NOT EXISTS `RealEstateAgency`.`Contract` (

`contractID` INT NOT NULL AUTO\_INCREMENT,

`pownerID` INT NOT NULL,

`listingID` INT NOT NULL,

`tax` FLOAT NOT NULL,

PRIMARY KEY (`contractID`, `listingID`, `pownerID`),

INDEX `con-pow\_idx` (`pownerID` ASC) VISIBLE,

INDEX `con-list\_idx` (`listingID` ASC) VISIBLE,

CONSTRAINT `con-pow`

FOREIGN KEY (`pownerID`)

REFERENCES `RealEstateAgency`.`PropertyOwner` (`pownerID`)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `con-list`

FOREIGN KEY (`listingID`)

REFERENCES `RealEstateAgency`.`Listing` (`listingID`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

DROP TABLE IF EXISTS `RealEstateAgency`.`Rent`;

CREATE TABLE IF NOT EXISTS `RealEstateAgency`.`Rent` (

`rentID` INT NOT NULL AUTO\_INCREMENT,

`contractID` INT NOT NULL,

`validfrom` DATE NOT NULL,

`validuntil` DATE NOT NULL,

`monthlyprice` FLOAT NOT NULL,

PRIMARY KEY (`rentID`, `contractID`),

INDEX `rent-con\_idx` (`contractID` ASC) VISIBLE,

CONSTRAINT `rent-con`

FOREIGN KEY (`contractID`)

REFERENCES `RealEstateAgency`.`Contract` (`contractID`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

DROP FUN

DELIMITER $$

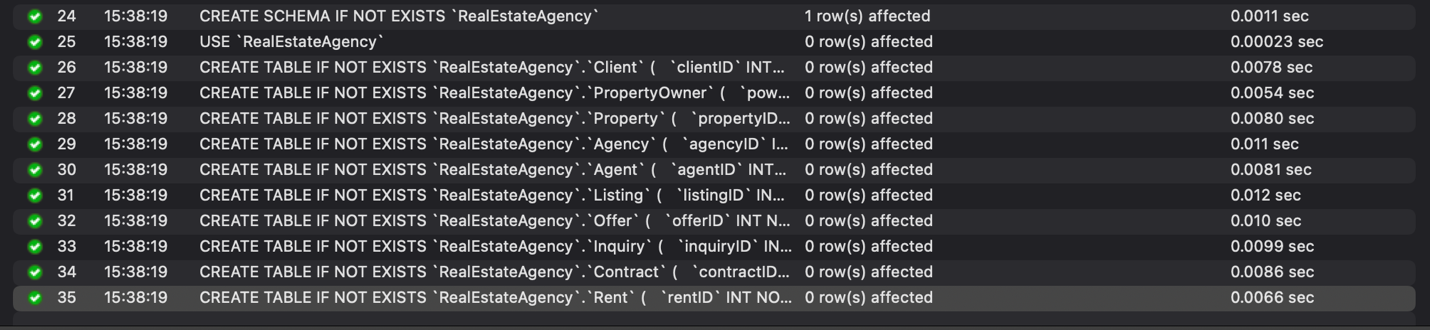


Рисунок 4.1 Створення таблиць

Для обмеження було створено тригери для перевірка деяких атрибутів перед вставкою у базу даних. Також є обмеження на тип дозволених символів. Далі надано скрипти для збереження цілісності даних.

DELIMITER $$

CREATE FUNCTION dates(date1 DATETIME, date2 DATETIME)

RETURNS INT DETERMINISTIC

BEGIN

RETURN YEAR(date2) - YEAR(date1) + (DATE\_FORMAT(date2, '%m%d') > DATE\_FORMAT(date1, '%m%d'));

END$$

CREATE TRIGGER add\_email BEFORE INSERT ON Client

FOR EACH ROW

BEGIN

IF NOT NEW.email NOT LIKE '[A-Z0-9.\_%-]+@[A-Z0-9.-]+\.[A-Z]{2,4}$' THEN

SET NEW.email = NULL;

END IF;

END $$

CREATE TRIGGER add\_area BEFORE INSERT ON Property

FOR EACH ROW

BEGIN

IF NEW.area NOT LIKE '%[^0-9]%' THEN

SET NEW.area = CONCAT(NEW.area,'m^2');

END IF;

END $$

CREATE TRIGGER add\_valid\_of BEFORE INSERT ON Offer

FOR EACH ROW

BEGIN

IF dates(NEW.validfrom, NEW.validuntil) < 0 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Incorrect data...';

END IF;

END $$

CREATE TRIGGER add\_valid\_r BEFORE INSERT ON Rent

FOR EACH ROW

BEGIN

IF dates(NEW.validfrom, NEW.validuntil) < 0 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Incorrect data...';

END IF;

END $$

CREATE TRIGGER add\_email\_UP BEFORE UPDATE ON Client

FOR EACH ROW

BEGIN

IF NEW.email NOT LIKE '[A-Z0-9.\_%-]+@[A-Z0-9.-]+\.[A-Z]{2,4}$' THEN

SET NEW.email = NULL;

END IF;

END $$

CREATE TRIGGER add\_area\_UP BEFORE UPDATE ON Property

FOR EACH ROW

BEGIN

IF NEW.area NOT LIKE '%[^0-9]%' THEN

SET NEW.area = CONCAT(NEW.area,'m^2');

END IF;

END $$

CREATE TRIGGER add\_valid\_of\_UP BEFORE UPDATE ON Offer

FOR EACH ROW

BEGIN

IF dates(NEW.validfrom, NEW.validuntil) <= 0 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Incorrect data...';

END IF;

END $$

CREATE TRIGGER add\_valid\_r\_UP BEFORE UPDATE ON Rent

FOR EACH ROW

BEGIN

IF dates(NEW.validfrom, NEW.validuntil) <= 0 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Incorrect data...';

END IF;

END $$

CREATE TRIGGER add\_valid\_O\_UP AFTER UPDATE ON Offer

FOR EACH ROW

BEGIN

DECLARE id INT;

SELECT propertyID INTO id FROM Listing WHERE listingID = NEW.listingID;

IF NEW.accepted = TRUE THEN

DELETE FROM Property WHERE propertyID=id;

END IF;

END $$

CREATE TRIGGER add\_valid\_L\_UP AFTER UPDATE ON Listing

FOR EACH ROW

BEGIN

IF NEW.open = TRUE THEN

DELETE FROM Listing WHERE (propertyID=NEW.propertyID AND listingID=NEW.listingID);

END IF;

END $$

DELIMITER ;

ALTER TABLE Client ADD CONSTRAINT c\_clname CHECK ((firstname NOT LIKE '%[^A-Z]%') AND (lastname NOT LIKE '%[^A-Z]%'));

ALTER TABLE PropertyOwner ADD CONSTRAINT c\_powname CHECK ((firstname NOT LIKE '%[^A-Z]%') AND (lastname NOT LIKE '%[^A-Z]%'));

ALTER TABLE Client ADD CONSTRAINT c\_clpas CHECK ((passport NOT LIKE '%[^0-9]%') AND (LENGTH(passport)=9));

ALTER TABLE Client ADD CONSTRAINT c\_clnum CHECK (contact NOT LIKE '%[^0-9+]%');

ALTER TABLE PropertyOwner ADD CONSTRAINT c\_pownum CHECK (contact NOT LIKE '%[^0-9+]%');

ALTER TABLE Offer ADD CONSTRAINT c\_prcur CHECK (pricecurrency NOT LIKE '%[^A-Z]%');

ALTER TABLE Property ADD CONSTRAINT c\_type CHECK ((type = 'sale') OR(type = 'rent'));

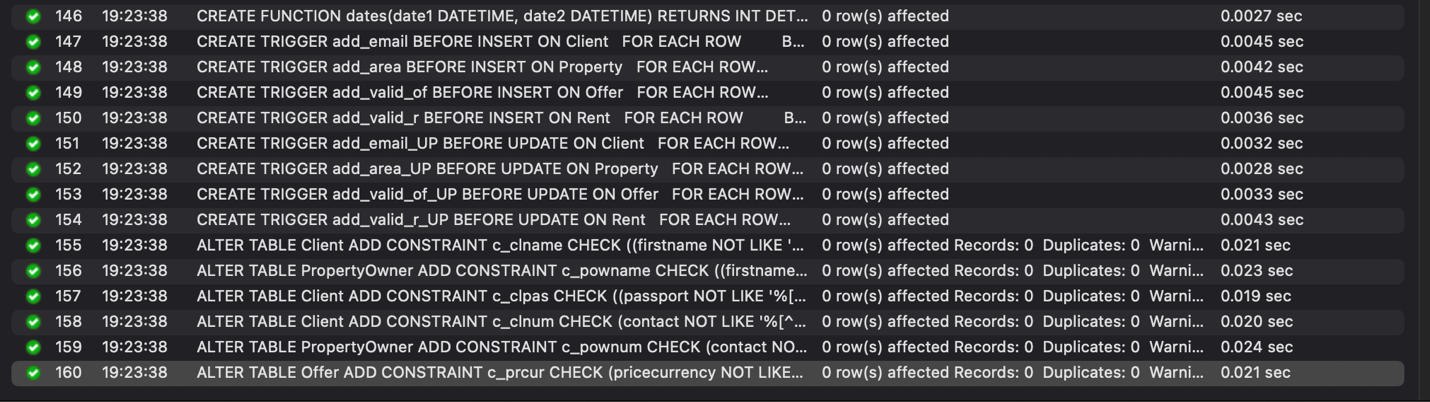


Рисунок 4.2 Створення тригерів та інших обмежень

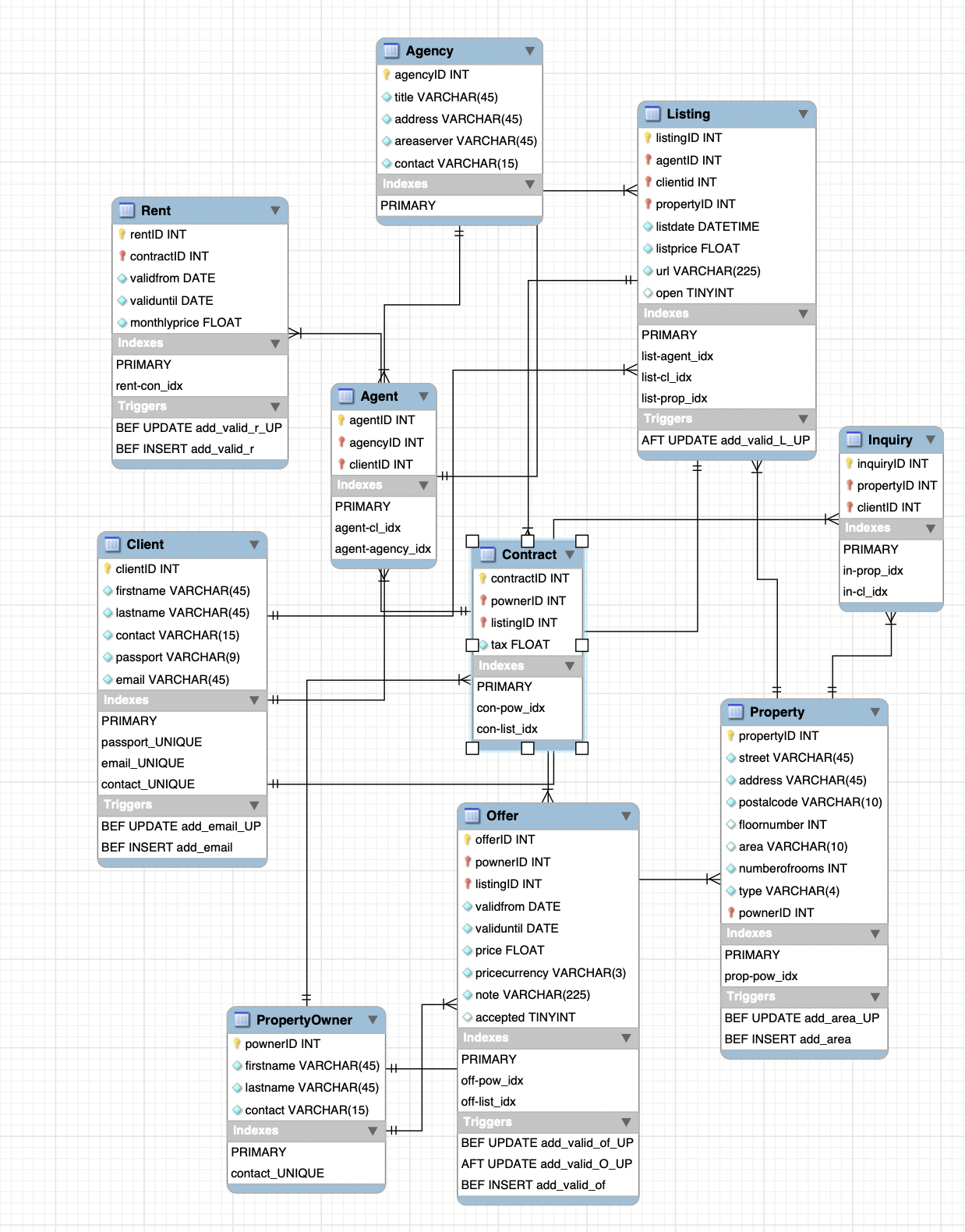


Рисунок 4.3 Згенерована схема БД

Отже, в результаті роботи було створено SQL скрипт для побудови бази даних.

# РОБОТА З БАЗОЮ ДАНИХ

Для роботи з базою даних потрібно імпортувати тестові дані, вони надані у Додатку Б. Під час роботи створені 20 DML запитів з використанням представлень та процедур. Після запитів показана робота з ролями користувачів.

Запит 1. Статистика, клієнти які шукали майно в оренду з площею.

CREATE

ALGORITHM = UNDEFINED

DEFINER = 'root'@'localhost'

SQL SECURITY DEFINER

VIEW rent\_inquiry AS

SELECT

Client.firstname AS Name,

Client.lastname AS Surname,

Property.type AS Type,

Property.area AS Area

FROM

((Client JOIN Inquiry ON((Inquiry.clientID = Client.clientID )))

JOIN Property ON((Property.propertyID = Inquiry.propertyID)))

WHERE

Property.type = 'rent'

GROUP BY Client.firstname, Client.lastname, Property.type, Property.area

ORDER BY Client.firstname, Client.lastname, Property.type

Запит:

SELECT \* FROM RealEstateAgency.rent\_inquiry;



Рисунок 5.1 Результат виконання запиту

Запит 2. Перевірка користувачів, що звʼязалися з агенцією.

CREATE

ALGORITHM = UNDEFINED

DEFINER = 'root'@'localhost'

SQL SECURITY DEFINER

VIEW agent\_agency AS

SELECT

Client.firstname AS Name,

Client.lastname AS Surname,

Agency.title AS Agency

FROM

(Agency INNER JOIN Agent ON (Agency.agencyID = Agent.agencyID)

INNER JOIN Client ON(Client.clientID = Agent.clientID))

WHERE Agency.title != 'title'

GROUP BY Client.firstname,Client.lastname, Agency.title

ORDER BY Client.firstname,Client.lastname

Запит:

SELECT \* FROM RealEstateAgency.agent\_agency;



Рисунок 5.2 Результат виконання запиту

Запит 3. Знаходження власників нерухомості з майном в «Херсоні».

CREATE

ALGORITHM = UNDEFINED

DEFINER = 'root'@'localhost'

SQL SECURITY DEFINER

VIEW kherson AS

SELECT

CONCAT(PropertyOwner.firstname, " ", PropertyOwner.lastname) AS Name

FROM

(PropertyOwner JOIN Property ON (PropertyOwner.pownerID = Property.pownerID))

WHERE Property.address = 'Kherson'

GROUP BY Name

ORDER BY Name

Запит:

SELECT \* FROM RealEstateAgency.kherson;

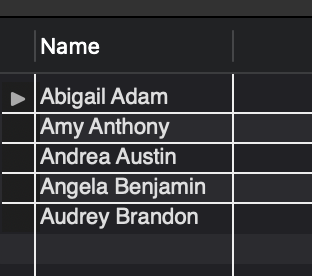


Рисунок 5.3 Результат виконання запиту

Запит 4. Перевірка домовленостей клієнтів з агенцією за роком.

DELIMITER $$

CREATE DEFINER='root'@'localhost' PROCEDURE findClientDate(searched VARCHAR(4))

BEGIN

SELECT CONCAT(Client.firstname, " ", Client.lastname) AS Client, Client.passport AS Passport, Listing.listprice AS Price

FROM Client

JOIN Listing ON Client.clientID = Listing.clientID

WHERE YEAR(Listing.listdate) = searched;

END$$

DELIMITER ;

Запит:

call RealEstateAgency.findClientDate('2002');

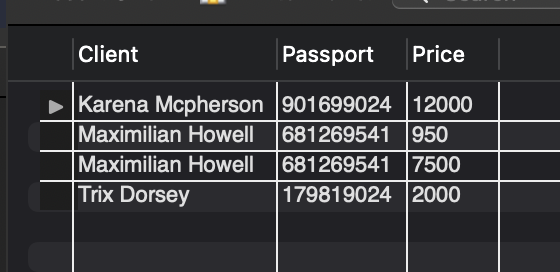


Рисунок 5.4 Результат виконання запиту

Запит 5. Шукати клієнтів, які купили нерухомість за роком.

DELIMITER $$

CREATE DEFINER='root'@'localhost' PROCEDURE findFromYear(searched YEAR)

BEGIN

SELECT CONCAT(Client.firstname, " ", Client.lastname) AS Client, Property.street AS address, CONCAT(Property.postalcode,", ",Property.address) AS PostalCode

FROM Client

JOIN Inquiry ON Inquiry.clientID = Client.clientID

JOIN Property ON Property.propertyID = Inquiry.propertyID

JOIN Offer ON Offer.pownerID = Property.pownerID

WHERE YEAR(Offer.validfrom) = searched;

END$$

DELIMITER ;

Запит:

call RealEstateAgency. findFromYear('2002');

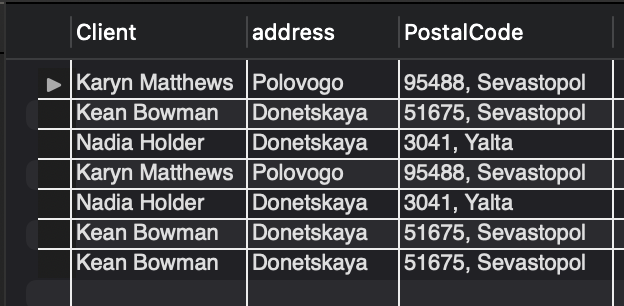


Рисунок 5.5 Результат виконання запиту

Запит 6. Клієнти які заплатили за послуги агента менше 7000.

SELECT CONCAT (Client.firstname, " ", Client.lastname) AS Name, Client.contact, Listing.listdate, Listing.listprice FROM Client INNER JOIN Listing ON Client.clientID = Listing.clientID AND Listing.listprice < 7000;

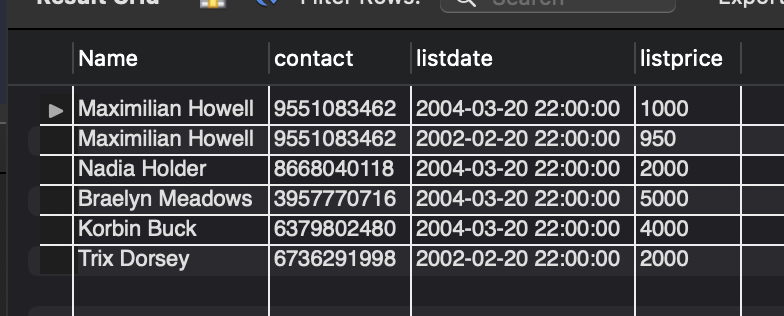


Рисунок 5.6 Результат виконання запиту

Запит 7. Інформація клієнта та продавця майна за купівлею.

SELECT Client.\*, PropertyOwner.\* FROM RealEstateAgency.Client INNER JOIN RealEstateAgency.Listing ON Client.clientID = Listing.clientID INNER JOIN RealEstateAgency.Offer ON Listing.listingID = Offer.listingID INNER JOIN RealEstateAgency.PropertyOwner ON PropertyOwner.pownerID = Offer.pownerID ORDER BY Client.clientID;

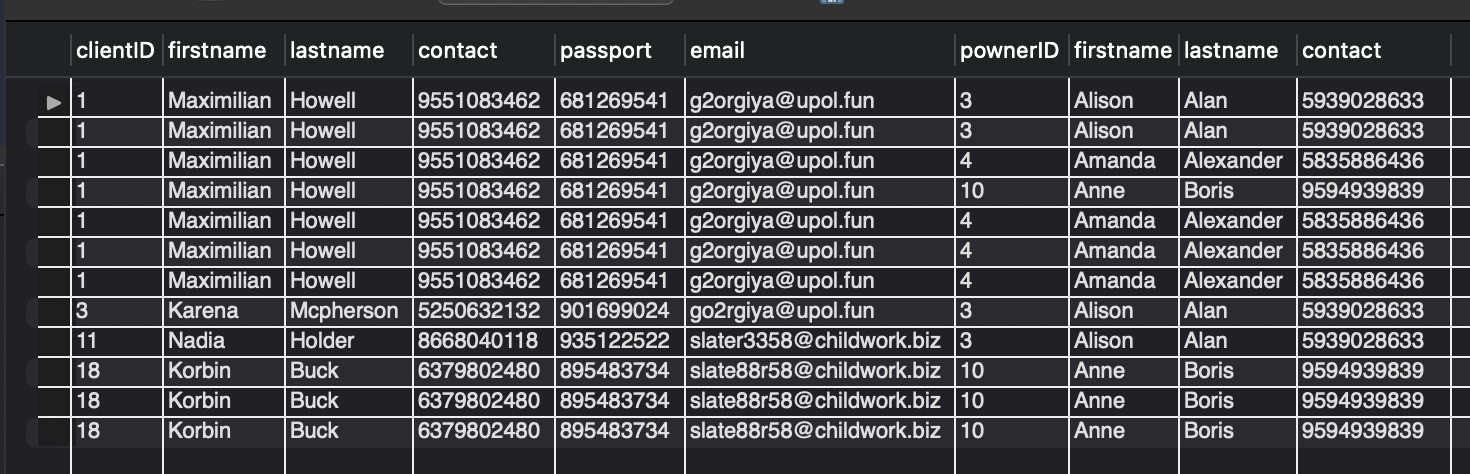


Рисунок 5.7 Результат виконання запиту

Запит 8. Агент який здійснив процес оренди.

SELECT Agent.agentID, Agency.title, Agency.contact, Listing.listdate FROM RealEstateAgency.Agency INNER JOIN RealEstateAgency.Agent ON Agency.agencyID = Agent.agencyID INNER JOIN RealEstateAgency.Listing ON Listing.agentID = Agent.agentID INNER JOIN RealEstateAgency.Contract ON Contract.listingID = Listing.listingID INNER JOIN RealEstateAgency.Rent ON Rent.contractID = Contract.contractID;

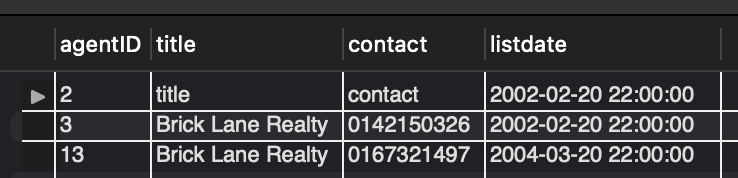


Рисунок 5.8 Результат виконання запиту

Запит 9. Кількість обговорень у кожного клієнта.

SELECT Client.clientID, COUNT(Listing.listingID) AS Sum\_list FROM RealEstateAgency.Client RIGHT JOIN RealEstateAgency.Listing ON Client.clientID = Listing.clientID GROUP BY Client.clientID ORDER BY Client.clientID;

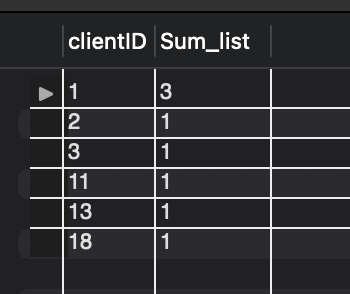


Рисунок 5.9 Результат виконання запиту

Запит 10. Кількість майна у кожного власника нерухомості.

SELECT CONCAT(PropertyOwner.firstname, " ",PropertyOwner.lastname) AS Owner, COUNT(Property.propertyID) AS All\_properties FROM RealEstateAgency.PropertyOwner RIGHT JOIN RealEstateAgency.Property ON PropertyOwner.pownerID = Property.pownerID GROUP BY Owner ORDER BY Owner;



Рисунок 5.10 Результат виконання запиту

Запит 11. Оборот коштів – сума, яку отримали власники при продажі майна та процес від спільної суми.

SELECT CONCAT(PropertyOwner.firstname, " ",PropertyOwner.LAstname) AS Name, SUM(Offer.price) AS Sum, SUM(Offer.price)/(SELECT SUM(Offer.price) FROM RealEstateAgency.Offer) AS "% of Total"FROM RealEstateAgency.PropertyOwner INNER JOIN RealEstateAgency.Offer ON PropertyOwner.pownerID = Offer.pownerID GROUP BY Name ORDER BY Name;



Рисунок 5.11 Результат виконання запиту

Запит 12. Сторони контракту оренди, що почав свою дію.

SELECT PropertyOwner.\*, Client.\* FROM RealEstateAgency.PropertyOwner INNER JOIN RealEstateAgency.Contract ON Contract.pownerID = PropertyOwner.pownerID INNER JOIN RealEstateAgency.Rent ON Rent.contractID = Contract.contractID INNER JOIN Listing ON Contract.listingID = Listing.listingID INNER JOIN Client ON Listing.clientID = Client.clientID WHERE Rent.validfrom <= CURDATE() ORDER BY PropertyOwner.pownerID;

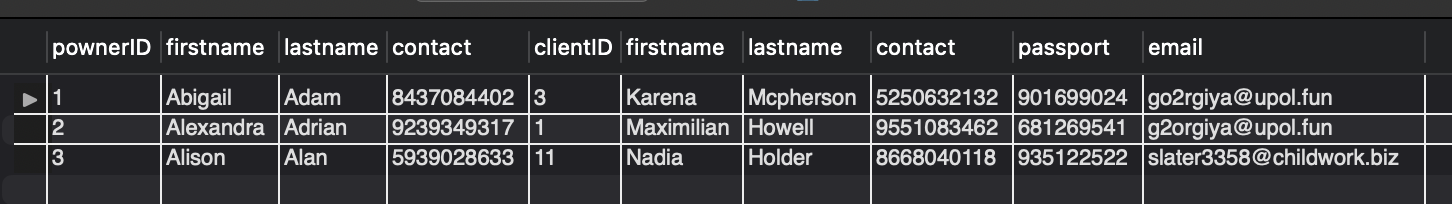


Рисунок 5.12 Результат виконання запиту

Запит 13. Агенти, чия заробітна плата більше середньої усіх працівників.

SELECT Agency.\*, Agent.agentID FROM RealEstateAgency.Agency INNER JOIN RealEstateAgency.Agent ON Agent.agencyID = Agency.agencyID INNER JOIN RealEstateAgency.Listing ON Listing.agentID = Agent.agentID WHERE listprice > (SELECT AVG(Listing.listprice) FROM RealEstateAgency.Listing) ORDER BY Agency.agencyID;

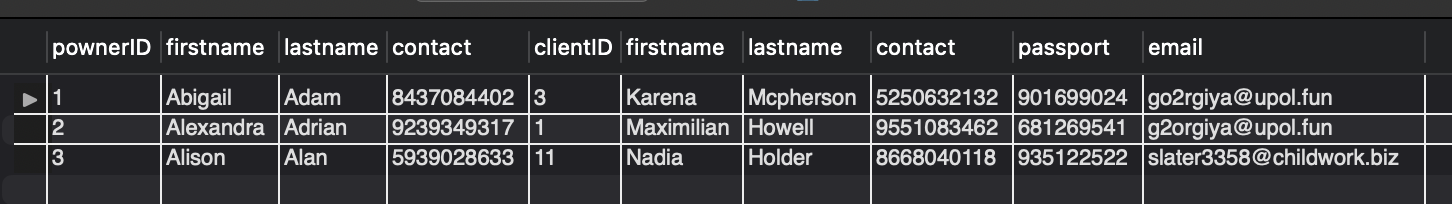


Рисунок 5.13 Результат виконання запиту

Запит 14. Агенція з більшою середньою заробітною платою.

SELECT Agency.title, AVG(Listing.listprice) AS AveragePrice FROM RealEstateAgency.Agency INNER JOIN RealEstateAgency.Agent ON Agent.agencyID = Agency.agencyID INNER JOIN RealEstateAgency.Listing ON Listing.agentID = Agent.agentID GROUP BY Agency.title HAVING AveragePrice>=ALL (SELECT AVG(Listing.listprice) FROM Listing GROUP BY Listing.listprice);

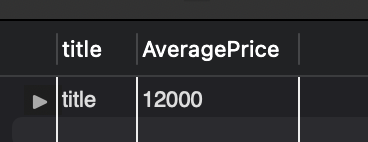


Рисунок 5.14 Результат виконання запиту

Запит 15. Клієнти, які не писали агенту, але ввійшли у Систему.

SELECT Client.\* FROM RealEstateAgency.Client WHERE NOT EXISTS (SELECT \* FROM RealEstateAgency.Listing WHERE Listing.clientID = RealEstateAgency.Client.clientID);

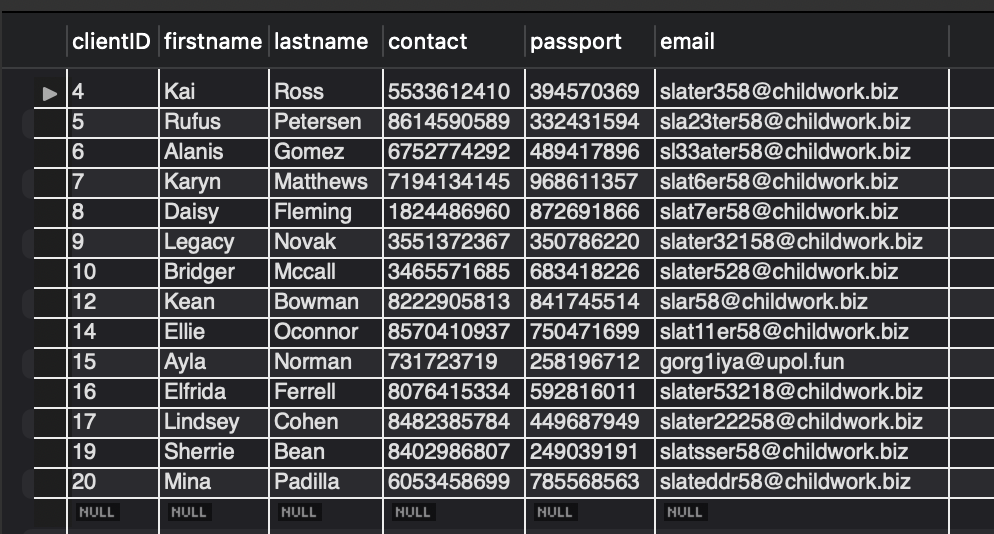


Рисунок 5.15 Результат виконання запиту

Запит 16. Умови кожного контракту оренди.

SELECT Contract.contractID, dates(Rent.validfrom, Rent.validuntil) AS Years, Rent.monthlyprice AS MonthlyPrice, Rent.monthlyprice+Contract.tax AS FirstMonthPrice FROM RealEstateAgency.Contract INNER JOIN RealEstateAgency.Rent ON Contract.contractID = Rent.contractID AND dates(Rent.validfrom, Rent.validuntil) != 0 ORDER BY Contract.contractID;

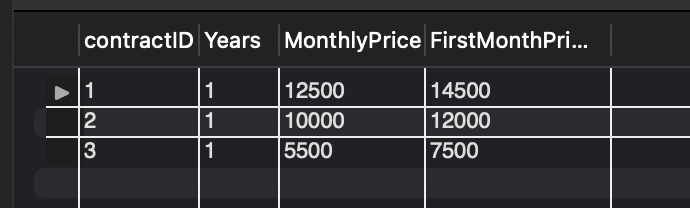


Рисунок 5.16 Результат виконання запиту

Запит 17. Кількість запросів на кожне оголошення.

SELECT Property.propertyID, COUNT(Inquiry.inquiryID) AS Count FROM RealEstateAgency.Inquiry INNER JOIN RealEstateAgency.Property ON Property.propertyID = Inquiry.propertyID GROUP BY Property.propertyID ORDER BY Property.propertyID;

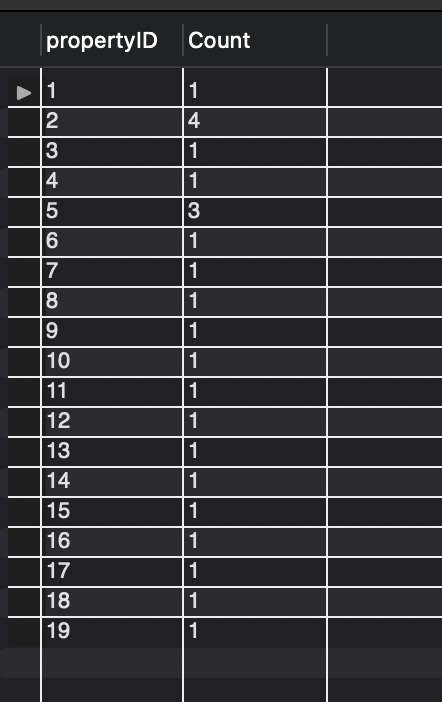


Рисунок 5.17 Результат виконання запиту

Запит 18. Знаходження клієнтів потрібного агенства.

DELIMITER $$

CREATE DEFINER='root'@'localhost' PROCEDURE findClients(ID INT)

SELECT firstname, lastname, email, contact FROM RealEstateAgency.Client JOIN RealEstateAgency.Agent ON RealEstateAgency.Client.clientID = RealEstateAgency.Agent.clientID WHERE RealEstateAgency.Agent.agencyID = ID;

END$$

DELIMITER ;

Запит:

call RealEstateAgency.findClients(2);

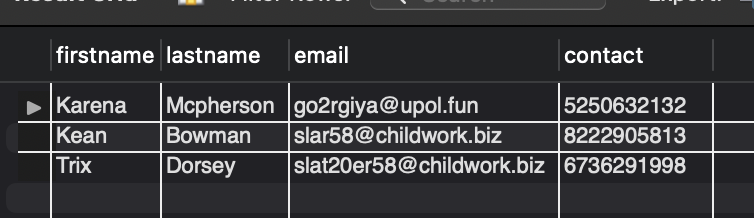


Рисунок 5.18 Результат виконання запиту

Запит 19. Подивитися нерухомість, чий почтовий код починається з «6».

SELECT CONCAT(firstname," ", lastname) AS Owner, Property.\* FROM RealEstateAgency.PropertyOwner JOIN RealEstateAgency.Property ON RealEstateAgency.PropertyOwner.pownerID = RealEstateAgency.Property.pownerID WHERE RealEstateAgency.Property.postalcode LIKE '6%';

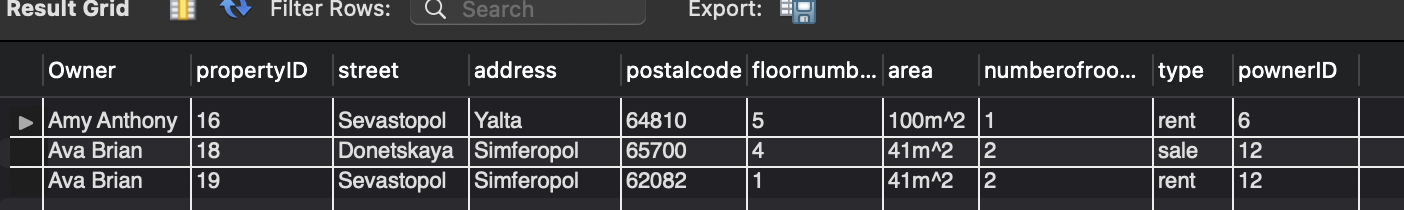


Рисунок 5.19 Результат виконання запиту

Запит 20. Знайти першого клієнта у черзі офісу агенства за номером офісу.

SELECT CONCAT(firstname, " ", lastname) AS Client, contact FROM RealEstateAgency.Client WHERE clientID = (SELECT clientID FROM RealEstateAgency.Agent WHERE agencyID = (SELECT agencyID FROM RealEstateAgency.Agency WHERE Agency.contact = '0142150326'));

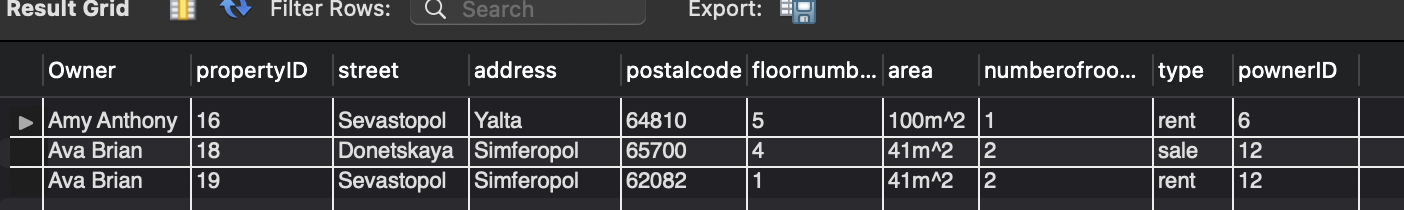
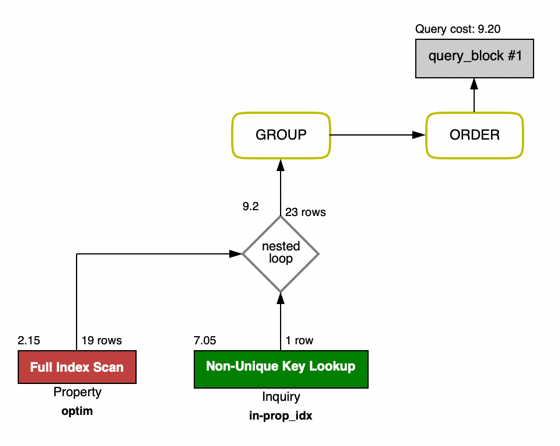


Рисунок 5.20 Результат виконання запиту

Окремим пунктом курсового проекту була оптимізація та представлені схеми запиту до і після її втілення:

Існує наступний запит:

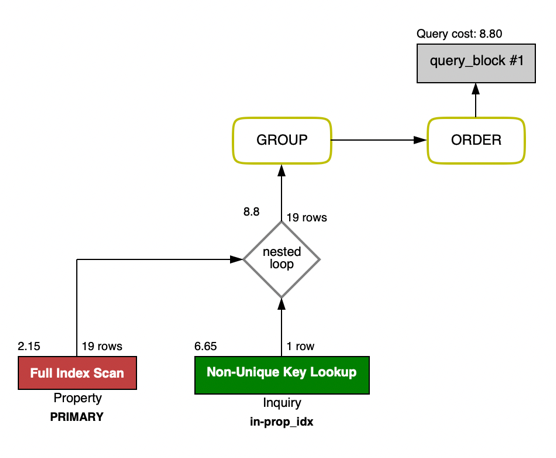
SELECT Property.propertyID ,COUNT(Inquiry.inquiryID) AS Count FROM RealEstateAgency.Inquiry INNER JOIN RealEstateAgency.Property ON Property.propertyID = Inquiry.propertyID GROUP BY Property.propertyID ORDER BY Property.propertyID;





CREATE INDEX optim ON Property(propertyID);

SELECT Property.propertyID ,COUNT(Inquiry.inquiryID) AS Count FROM RealEstateAgency.Inquiry INNER JOIN RealEstateAgency.Property ON Property.propertyID = Inquiry.propertyID GROUP BY Property.propertyID ORDER BY Property.propertyID;





Отже, бачимо, що до оптимізації тривалість роботи 0,003 c, а після 0,00066 с.

Щодо багатокористувацької моделі, то у базі даних було створено 3 основних користувачі: root, client та owner, на основі яких була показана взаємодія різних прав доступу до БД, виходячи з логічних міркувань:

1. Суперкористувач(root) є адміністратором системи і має повний функціонал по відношенню до усіх сутностей системи. Цю роль приймає агент.

2. Клієнт має право тільки знаходити важливу для нього інформацію, але не вносити свої зміни до сутностей, окрім реєстрації – додавання своїх даних у базу.

3. Власник має право тільки знаходити важливу для нього інформацію, єдині достопні можливості: реєстрація та створення оголошень про свою нерухомість – додавання своїх даних у базу.

Нижче наведено приклади синтаксису додавання користувачів системи, зміни їх прав доступу та роботи запитів для різних користувачів:

DROP USER IF EXISTS 'client'@'localhost';

CREATE USER 'client'@'localhost' IDENTIFIED BY '1234';

GRANT SELECT ON RealEstateAgency.\* TO 'client'@'localhost';

GRANT INSERT, UPDATE ON TABLE RealEstateAgency.Property TO 'client'@'localhost';

DROP USER IF EXISTS 'owner'@'localhost';

CREATE USER 'owner'@'localhost' IDENTIFIED BY '1234';

GRANT SELECT ON RealEstateAgency.\* TO 'owner'@'localhost';

GRANT INSERT, UPDATE ON TABLE RealEstateAgency.Property TO 'owner'@'localhost';

GRANT INSERT, UPDATE ON TABLE RealEstateAgency.PropertyOwner TO 'owner'@'localhost';

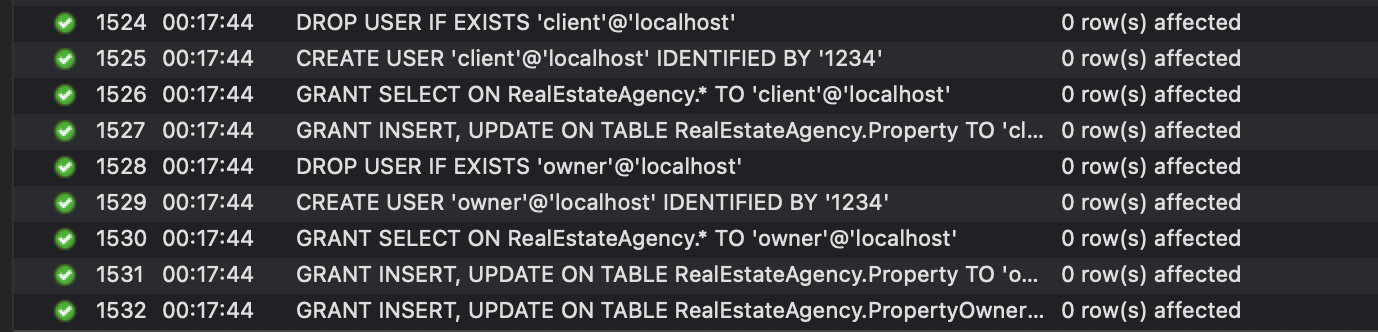


Рисунок 5.21 Результат надання прав

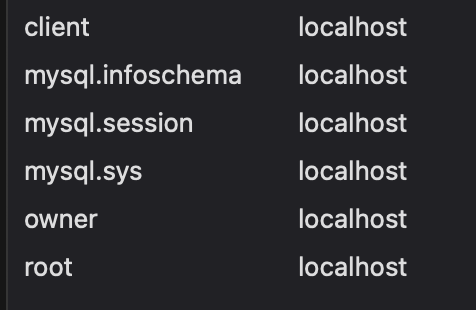


Рисунок 5.21 Список користувачів

Користувачів створено і права надано.

# ВИСНОВКИ

У ході виконання розглядуваного курсового проекту, мною було розроблено базу даних «Агенції нерухомості». Першим кроком створення БД було ознайомлення з предметною областю та виконання підготовчих етапів: розробка концептуальної моделі БД(виділення сутностей, прелімінарне відтворення зв’язків між ними та втілення напрацювань у ER-діаграмі, декларування бізнес-правил та обмежень, які далі будуть присутні у реалізації БД), розробка логічної моделі БД(побудова таблиць) та безпосередня реалізація БД програмно у середовищі MySQL Workbench на мові MySQL. Було створено 10 таблиць з обмеженнями для збереження цілісності даних. Щодо специфіки завдання курсової роботи, то було застосовано тригери, генератори(автоінкремент), представлення, збережені процедури, функції, як мінімум 20 DML-запитів типу SELECT.

У майбутньому планується підключення створеної БД до системи управління базою даних(БД).

Як наслідок, реалізована база даних підтримує функціонал на запитах, який у подальшому можна використати для СУБД, адже новостворена БД уможливлює коректне додавання, обробку та виведення затребуваної інформації.

# ПЕРЕЛІК ПОСИЛАНЬ

1. Стаття. Як працювати з агентством нерухомості: корисні рекомендації. URL:

https://dim.ctd.com.ua/statti/yak-pratsiuvaty-z-ahentstvom-nerukhomosti-korysni-rekomendatsii

1. Портал нерухомості, УкрХата. URL:

<https://ukrxata.info/>

1. Сайт нерухомості, DOM.RIA. URL:

<https://dom.ria.com/uk/>

1. Стаття. Скільки беруть ріелтори за свої послуги. URL: <https://ocenka.ua/skilky-berut-rieltory-za-svoyi-poslugy/>
2. Про ріелторську діяльність Президент України; Указ від 27.06.1999 № 733/99. URL:

<https://web.archive.org/web/20160815172435/http://zakon2.rada.gov.ua/laws/show/733/99>

1. Стаття. В чем разница между брокером по недвижимости и агентством? URL:

<https://strojka-veka.info/v-chem-raznica-mezhdu-brokerom-po-nedvizhimosti-i-agentstvom/>

1. Стаття. Estate agents and the IPI. URL:

<https://web.archive.org/web/20160701154905/http://www.immoweb.be/en/Page.cfm?Page=Agencies_IPI.htm>

Додаток А Тексти SQL скриптів та  
результати їх роботи

(Вид носія даних)

*Віртуальний носій*

*студента групи ІП-15 2 курсу*

*Мєшкова А.І.*

DROP SCHEMA IF EXISTS `RealEstateAgency`;

CREATE SCHEMA `RealEstateAgency`;

USE `RealEstateAgency` ;

DROP TABLE IF EXISTS `RealEstateAgency`.`Client`;

CREATE TABLE IF NOT EXISTS `RealEstateAgency`.`Client` (

`clientID` INT NOT NULL AUTO\_INCREMENT,

`firstname` VARCHAR(45) NOT NULL,

`lastname` VARCHAR(45) NOT NULL,

`contact` VARCHAR(15) NOT NULL,

`passport` VARCHAR(9) NOT NULL,

`email` VARCHAR(45) NOT NULL,

PRIMARY KEY (`clientID`),

UNIQUE INDEX `passport\_UNIQUE` (`passport` ASC) VISIBLE,

UNIQUE INDEX `email\_UNIQUE` (`email` ASC) VISIBLE,

UNIQUE INDEX `contact\_UNIQUE` (`contact` ASC) VISIBLE)

ENGINE = InnoDB;

DROP TABLE IF EXISTS `RealEstateAgency`.`PropertyOwner`;

CREATE TABLE IF NOT EXISTS `RealEstateAgency`.`PropertyOwner` (

`pownerID` INT NOT NULL AUTO\_INCREMENT,

`firstname` VARCHAR(45) NOT NULL,

`lastname` VARCHAR(45) NOT NULL,

`contact` VARCHAR(15) NOT NULL,

PRIMARY KEY (`pownerID`),

UNIQUE INDEX `contact\_UNIQUE` (`contact` ASC) VISIBLE)

ENGINE = InnoDB;

DROP TABLE IF EXISTS `RealEstateAgency`.`Property`;

CREATE TABLE IF NOT EXISTS `RealEstateAgency`.`Property` (

`propertyID` INT NOT NULL AUTO\_INCREMENT,

`street` VARCHAR(45) NOT NULL,

`address` VARCHAR(45) NOT NULL,

`postalcode` VARCHAR(10) NOT NULL,

`floornumber` INT NULL,

`area` VARCHAR(10) DEFAULT 'm^2',

`numberofrooms` INT NOT NULL,

`type` VARCHAR(4) NOT NULL,

`pownerID` INT NOT NULL,

PRIMARY KEY (`propertyID`, `pownerID`),

INDEX `prop-pow\_idx` (`pownerID` ASC) VISIBLE,

CONSTRAINT `prop-pow`

FOREIGN KEY (`pownerID`)

REFERENCES `RealEstateAgency`.`PropertyOwner` (`pownerID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

DROP TABLE IF EXISTS `RealEstateAgency`.`Agency`;

CREATE TABLE IF NOT EXISTS `RealEstateAgency`.`Agency` (

`agencyID` INT NOT NULL AUTO\_INCREMENT,

`title` VARCHAR(45) NOT NULL,

`address` VARCHAR(45) NOT NULL,

`areaserver` VARCHAR(45) NOT NULL,

`contact` VARCHAR(15) NOT NULL,

PRIMARY KEY (`agencyID`))

ENGINE = InnoDB;

DROP TABLE IF EXISTS `RealEstateAgency`.`Agent`;

CREATE TABLE IF NOT EXISTS `RealEstateAgency`.`Agent` (

`agentID` INT NOT NULL AUTO\_INCREMENT,

`agencyID` INT NOT NULL,

`clientID` INT NOT NULL,

PRIMARY KEY (`agentID`, `agencyID`, `clientID`),

INDEX `agent-cl\_idx` (`clientID` ASC) VISIBLE,

INDEX `agent-agency\_idx` (`agencyID` ASC) VISIBLE,

CONSTRAINT `agent-agency`

FOREIGN KEY (`agencyID`)

REFERENCES `RealEstateAgency`.`Agency` (`agencyID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `agent-cl`

FOREIGN KEY (`clientID`)

REFERENCES `RealEstateAgency`.`Client` (`clientID`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

DROP TABLE IF EXISTS `RealEstateAgency`.`Listing`;

CREATE TABLE IF NOT EXISTS `RealEstateAgency`.`Listing` (

`listingID` INT NOT NULL AUTO\_INCREMENT,

`agentID` INT NOT NULL,

`clientid` INT NOT NULL,

`propertyID` INT NOT NULL,

`listdate` DATETIME NOT NULL,

`listprice` FLOAT NOT NULL,

`url` VARCHAR(225) NOT NULL,

`open` TINYINT DEFAULT TRUE,

PRIMARY KEY (`listingID`, `agentID`, `propertyID`, `clientid`),

INDEX `list-agent\_idx` (`agentID` ASC) VISIBLE,

INDEX `list-cl\_idx` (`clientid` ASC) VISIBLE,

INDEX `list-prop\_idx` (`propertyID` ASC) VISIBLE,

CONSTRAINT `list-agent`

FOREIGN KEY (`agentID`)

REFERENCES `RealEstateAgency`.`Agent` (`agentID`)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `list-cl`

FOREIGN KEY (`clientid`)

REFERENCES `RealEstateAgency`.`Client` (`clientID`)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `list-prop`

FOREIGN KEY (`propertyID`)

REFERENCES `RealEstateAgency`.`Property` (`propertyID`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

DROP TABLE IF EXISTS `RealEstateAgency`.`Offer`;

CREATE TABLE IF NOT EXISTS `RealEstateAgency`.`Offer` (

`offerID` INT NOT NULL AUTO\_INCREMENT,

`pownerID` INT NOT NULL,

`listingID` INT NOT NULL,

`validfrom` DATE NOT NULL,

`validuntil` DATE NOT NULL,

`price` FLOAT NOT NULL,

`pricecurrency` VARCHAR(3) NOT NULL,

`note` VARCHAR(225) NOT NULL,

`accepted` TINYINT DEFAULT FALSE,

PRIMARY KEY (`offerID`, `listingID`, `pownerID`),

INDEX `off-pow\_idx` (`pownerID` ASC) VISIBLE,

INDEX `off-list\_idx` (`listingID` ASC) VISIBLE,

CONSTRAINT `off-pow`

FOREIGN KEY (`pownerID`)

REFERENCES `RealEstateAgency`.`PropertyOwner` (`pownerID`)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `off-list`

FOREIGN KEY (`listingID`)

REFERENCES `RealEstateAgency`.`Listing` (`listingID`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

DROP TABLE IF EXISTS `RealEstateAgency`.`Inquiry`;

CREATE TABLE IF NOT EXISTS `RealEstateAgency`.`Inquiry` (

`inquiryID` INT NOT NULL AUTO\_INCREMENT,

`propertyID` INT NOT NULL,

`clientID` INT NOT NULL,

PRIMARY KEY (`inquiryID`, `clientID`, `propertyID`),

INDEX `in-prop\_idx` (`propertyID` ASC) VISIBLE,

INDEX `in-cl\_idx` (`clientID` ASC) VISIBLE,

CONSTRAINT `in-prop`

FOREIGN KEY (`propertyID`)

REFERENCES `RealEstateAgency`.`Property` (`propertyID`)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `in-cl`

FOREIGN KEY (`clientID`)

REFERENCES `RealEstateAgency`.`Client` (`clientID`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

DROP TABLE IF EXISTS `RealEstateAgency`.`Contract`;

CREATE TABLE IF NOT EXISTS `RealEstateAgency`.`Contract` (

`contractID` INT NOT NULL AUTO\_INCREMENT,

`pownerID` INT NOT NULL,

`listingID` INT NOT NULL,

`tax` FLOAT NOT NULL,

PRIMARY KEY (`contractID`, `listingID`, `pownerID`),

INDEX `con-pow\_idx` (`pownerID` ASC) VISIBLE,

INDEX `con-list\_idx` (`listingID` ASC) VISIBLE,

CONSTRAINT `con-pow`

FOREIGN KEY (`pownerID`)

REFERENCES `RealEstateAgency`.`PropertyOwner` (`pownerID`)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `con-list`

FOREIGN KEY (`listingID`)

REFERENCES `RealEstateAgency`.`Listing` (`listingID`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

DROP TABLE IF EXISTS `RealEstateAgency`.`Rent`;

CREATE TABLE IF NOT EXISTS `RealEstateAgency`.`Rent` (

`rentID` INT NOT NULL AUTO\_INCREMENT,

`contractID` INT NOT NULL,

`validfrom` DATE NOT NULL,

`validuntil` DATE NOT NULL,

`monthlyprice` FLOAT NOT NULL,

PRIMARY KEY (`rentID`, `contractID`),

INDEX `rent-con\_idx` (`contractID` ASC) VISIBLE,

CONSTRAINT `rent-con`

FOREIGN KEY (`contractID`)

REFERENCES `RealEstateAgency`.`Contract` (`contractID`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

DROP FUN

DELIMITER $$

DELIMITER $$

CREATE FUNCTION dates(date1 DATETIME, date2 DATETIME)

RETURNS INT DETERMINISTIC

BEGIN

RETURN YEAR(date2) - YEAR(date1) + (DATE\_FORMAT(date2, '%m%d') > DATE\_FORMAT(date1, '%m%d'));

END$$

CREATE TRIGGER add\_email BEFORE INSERT ON Client

FOR EACH ROW

BEGIN

IF NOT NEW.email NOT LIKE '[A-Z0-9.\_%-]+@[A-Z0-9.-]+\.[A-Z]{2,4}$' THEN

SET NEW.email = NULL;

END IF;

END $$

CREATE TRIGGER add\_area BEFORE INSERT ON Property

FOR EACH ROW

BEGIN

IF NEW.area NOT LIKE '%[^0-9]%' THEN

SET NEW.area = CONCAT(NEW.area,'m^2');

END IF;

END $$

CREATE TRIGGER add\_valid\_of BEFORE INSERT ON Offer

FOR EACH ROW

BEGIN

IF dates(NEW.validfrom, NEW.validuntil) < 0 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Incorrect data...';

END IF;

END $$

CREATE TRIGGER add\_valid\_r BEFORE INSERT ON Rent

FOR EACH ROW

BEGIN

IF dates(NEW.validfrom, NEW.validuntil) < 0 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Incorrect data...';

END IF;

END $$

CREATE TRIGGER add\_email\_UP BEFORE UPDATE ON Client

FOR EACH ROW

BEGIN

IF NEW.email NOT LIKE '[A-Z0-9.\_%-]+@[A-Z0-9.-]+\.[A-Z]{2,4}$' THEN

SET NEW.email = NULL;

END IF;

END $$

CREATE TRIGGER add\_area\_UP BEFORE UPDATE ON Property

FOR EACH ROW

BEGIN

IF NEW.area NOT LIKE '%[^0-9]%' THEN

SET NEW.area = CONCAT(NEW.area,'m^2');

END IF;

END $$

CREATE TRIGGER add\_valid\_of\_UP BEFORE UPDATE ON Offer

FOR EACH ROW

BEGIN

IF dates(NEW.validfrom, NEW.validuntil) <= 0 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Incorrect data...';

END IF;

END $$

CREATE TRIGGER add\_valid\_r\_UP BEFORE UPDATE ON Rent

FOR EACH ROW

BEGIN

IF dates(NEW.validfrom, NEW.validuntil) <= 0 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Incorrect data...';

END IF;

END $$

CREATE TRIGGER add\_valid\_O\_UP AFTER UPDATE ON Offer

FOR EACH ROW

BEGIN

DECLARE id INT;

SELECT propertyID INTO id FROM Listing WHERE listingID = NEW.listingID;

IF NEW.accepted = TRUE THEN

DELETE FROM Property WHERE propertyID=id;

END IF;

END $$

CREATE TRIGGER add\_valid\_L\_UP AFTER UPDATE ON Listing

FOR EACH ROW

BEGIN

IF NEW.open = TRUE THEN

DELETE FROM Listing WHERE (propertyID=NEW.propertyID AND listingID=NEW.listingID);

END IF;

END $$

DELIMITER ;

ALTER TABLE Client ADD CONSTRAINT c\_clname CHECK ((firstname NOT LIKE '%[^A-Z]%') AND (lastname NOT LIKE '%[^A-Z]%'));

ALTER TABLE PropertyOwner ADD CONSTRAINT c\_powname CHECK ((firstname NOT LIKE '%[^A-Z]%') AND (lastname NOT LIKE '%[^A-Z]%'));

ALTER TABLE Client ADD CONSTRAINT c\_clpas CHECK ((passport NOT LIKE '%[^0-9]%') AND (LENGTH(passport)=9));

ALTER TABLE Client ADD CONSTRAINT c\_clnum CHECK (contact NOT LIKE '%[^0-9+]%');

ALTER TABLE PropertyOwner ADD CONSTRAINT c\_pownum CHECK (contact NOT LIKE '%[^0-9+]%');

ALTER TABLE Offer ADD CONSTRAINT c\_prcur CHECK (pricecurrency NOT LIKE '%[^A-Z]%');

ALTER TABLE Property ADD CONSTRAINT c\_type CHECK ((type = 'sale') OR(type = 'rent'));

USE RealEstateAgency;

CREATE

ALGORITHM = UNDEFINED

DEFINER = 'root'@'localhost'

SQL SECURITY DEFINER

VIEW rent\_inquiry AS

SELECT

Client.firstname AS Name,

Client.lastname AS Surname,

Property.type AS Type,

Property.area AS Area

FROM

((Client JOIN Inquiry ON((Inquiry.clientID = Client.clientID )))

JOIN Property ON((Property.propertyID = Inquiry.propertyID)))

WHERE

Property.type = 'rent'

GROUP BY Client.firstname, Client.lastname, Property.area

ORDER BY Client.firstname, Client.lastname, Property.area

CREATE

ALGORITHM = UNDEFINED

DEFINER = 'root'@'localhost'

SQL SECURITY DEFINER

VIEW agent\_agency AS

SELECT

Client.firstname AS Name,

Client.lastname AS Surname,

Agency.title AS Agency

FROM

(Agency INNER JOIN Agent ON (Agency.agencyID = Agent.agencyID)

INNER JOIN Client ON(Client.clientID = Agent.clientID))

WHERE Agency.title != 'title'

GROUP BY Client.firstname,Client.lastname, Agency.title

ORDER BY Client.firstname,Client.lastname

CREATE

ALGORITHM = UNDEFINED

DEFINER = 'root'@'localhost'

SQL SECURITY DEFINER

VIEW kherson AS

SELECT

CONCAT(PropertyOwner.firstname, " ", PropertyOwner.lastname) AS Name

FROM

(PropertyOwner JOIN Property ON (PropertyOwner.pownerID = Property.pownerID))

WHERE Property.address = 'Kherson'

GROUP BY Name

ORDER BY Name

DELIMITER $$

CREATE DEFINER='root'@'localhost' PROCEDURE findClientDate(searched VARCHAR(4))

BEGIN

SELECT CONCAT(Client.firstname, " ", Client.lastname) AS Client, Client.passport AS Passport, Listing.listprice AS Price

FROM Client

JOIN Listing ON Client.clientID = Listing.clientID

WHERE YEAR(Listing.listdate) = searched;

END$$

DELIMITER ;

DELIMITER $$

CREATE DEFINER='root'@'localhost' PROCEDURE findFromYear(searched YEAR)

BEGIN

SELECT CONCAT(Client.firstname, " ", Client.lastname) AS Client, Property.street AS address, CONCAT(Property.postalcode,", ",Property.address) AS PostalCode

FROM Client

JOIN Inquiry ON Inquiry.clientID = Client.clientID

JOIN Property ON Property.propertyID = Inquiry.propertyID

JOIN Offer ON Offer.pownerID = Property.pownerID

WHERE YEAR(Offer.validfrom) = searched;

END$$

DELIMITER ;

SELECT CONCAT (Client.firstname, " ", Client.lastname) AS Name, Client.contact, Listing.listdate, Listing.listprice FROM Client INNER JOIN Listing ON Client.clientID = Listing.clientID AND Listing.listprice < 7000;

SELECT Client.\*, PropertyOwner.\* FROM RealEstateAgency.Client INNER JOIN RealEstateAgency.Listing ON Client.clientID = Listing.clientID INNER JOIN RealEstateAgency.Offer ON Listing.listingID = Offer.listingID INNER JOIN RealEstateAgency.PropertyOwner ON PropertyOwner.pownerID = Offer.pownerID ORDER BY Client.clientID;

SELECT Agent.agentID, Agency.title, Agency.contact, Listing.listdate FROM RealEstateAgency.Agency INNER JOIN RealEstateAgency.Agent ON Agency.agencyID = Agent.agencyID INNER JOIN RealEstateAgency.Listing ON Listing.agentID = Agent.agentID INNER JOIN RealEstateAgency.Contract ON Contract.listingID = Listing.listingID INNER JOIN RealEstateAgency.Rent ON Rent.contractID = Contract.contractID;

SELECT Client.clientID, COUNT(Listing.listingID) AS Sum\_list FROM RealEstateAgency.Client RIGHT JOIN RealEstateAgency.Listing ON Client.clientID = Listing.clientID GROUP BY Client.clientID ORDER BY Client.clientID;

SELECT CONCAT(PropertyOwner.firstname, " ",PropertyOwner.lastname) AS Owner, COUNT(Property.propertyID) AS All\_properties FROM RealEstateAgency.PropertyOwner RIGHT JOIN RealEstateAgency.Property ON PropertyOwner.pownerID = Property.pownerID GROUP BY Owner ORDER BY Owner;

SELECT CONCAT(PropertyOwner.firstname, " ",PropertyOwner.LAstname) AS Name, SUM(Offer.price) AS Sum, SUM(Offer.price)/(SELECT SUM(Offer.price) FROM RealEstateAgency.Offer) AS "% of Total"FROM RealEstateAgency.PropertyOwner INNER JOIN RealEstateAgency.Offer ON PropertyOwner.pownerID = Offer.pownerID GROUP BY Name ORDER BY Name;

SELECT PropertyOwner.\*, Client.\* FROM RealEstateAgency.PropertyOwner INNER JOIN RealEstateAgency.Contract ON Contract.pownerID = PropertyOwner.pownerID INNER JOIN RealEstateAgency.Rent ON Rent.contractID = Contract.contractID INNER JOIN Listing ON Contract.listingID = Listing.listingID INNER JOIN Client ON Listing.clientID = Client.clientID WHERE Rent.validfrom <= CURDATE() ORDER BY PropertyOwner.pownerID;

SELECT Agency.\*, Agent.agentID FROM RealEstateAgency.Agency INNER JOIN RealEstateAgency.Agent ON Agent.agencyID = Agency.agencyID INNER JOIN RealEstateAgency.Listing ON Listing.agentID = Agent.agentID WHERE listprice > (SELECT AVG(Listing.listprice) FROM RealEstateAgency.Listing) ORDER BY Agency.agencyID;

SELECT Agency.title, AVG(Listing.listprice) AS AveragePrice FROM RealEstateAgency.Agency INNER JOIN RealEstateAgency.Agent ON Agent.agencyID = Agency.agencyID INNER JOIN RealEstateAgency.Listing ON Listing.agentID = Agent.agentID GROUP BY Agency.title HAVING AveragePrice>=ALL (SELECT AVG(Listing.listprice) FROM Listing GROUP BY Listing.listprice);

SELECT Client.\* FROM RealEstateAgency.Client WHERE NOT EXISTS (SELECT \* FROM RealEstateAgency.Listing WHERE Listing.clientID = RealEstateAgency.Client.clientID);

SELECT Contract.contractID, dates(Rent.validfrom, Rent.validuntil) AS Years, Rent.monthlyprice AS MonthlyPrice, Rent.monthlyprice+Contract.tax AS FirstMonthPrice FROM RealEstateAgency.Contract INNER JOIN RealEstateAgency.Rent ON Contract.contractID = Rent.contractID AND dates(Rent.validfrom, Rent.validuntil) != 0 ORDER BY Contract.contractID;

SELECT Property.propertyID ,COUNT(Inquiry.inquiryID) AS Count FROM RealEstateAgency.Inquiry INNER JOIN RealEstateAgency.Property ON Property.propertyID = Inquiry.propertyID GROUP BY Property.propertyID ORDER BY Property.propertyID;

DELIMITER $$

CREATE DEFINER='root'@'localhost' PROCEDURE findClients(ID INT)

SELECT firstname, lastname, email, contact FROM RealEstateAgency.Client JOIN RealEstateAgency.Agent ON RealEstateAgency.Client.clientID = RealEstateAgency.Agent.clientID WHERE RealEstateAgency.Agent.agencyID = ID;

END$$

DELIMITER ;

SELECT CONCAT(firstname," ", lastname) AS Owner, Property.\* FROM RealEstateAgency.PropertyOwner JOIN RealEstateAgency.Property ON RealEstateAgency.PropertyOwner.pownerID = RealEstateAgency.Property.pownerID WHERE RealEstateAgency.Property.postalcode LIKE '6%';

SELECT CONCAT(firstname, " ", lastname) AS Client, contact FROM RealEstateAgency.Client WHERE clientID = (SELECT clientID FROM RealEstateAgency.Agent WHERE agencyID = (SELECT agencyID FROM RealEstateAgency.Agency WHERE Agency.contact = '0142150326'));

DROP USER IF EXISTS 'client'@'localhost';

CREATE USER 'client'@'localhost' IDENTIFIED BY '1234';

GRANT SELECT ON RealEstateAgency.\* TO 'client'@'localhost';

GRANT INSERT, UPDATE ON TABLE RealEstateAgency.Property TO 'client'@'localhost';

DROP USER IF EXISTS 'owner'@'localhost';

CREATE USER 'owner'@'localhost' IDENTIFIED BY '1234';

GRANT SELECT ON RealEstateAgency.\* TO 'owner'@'localhost';

GRANT INSERT, UPDATE ON TABLE RealEstateAgency.Property TO 'owner'@'localhost';

GRANT INSERT, UPDATE ON TABLE RealEstateAgency.PropertyOwner TO 'owner'@'localhost';

Додаток Б файли з даними, що   
були імпортовані в базу даних

(Вид носія даних)

*Віртуальний носій*

*студента групи ІП-15 2 курсу*

*Мєшкова А.І.*

CSV файли з даними, що були імпортовані в базу даних при виконанні курсової роботи доступні за посиланням:

<https://drive.google.com/drive/folders/1JRvcngKZ4BzAb08QDUYyfg1Z4QUDMmah?usp=sharing>