

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 для ведення реєстру земельних ділянок та їх власників з  
можливістю побудови топологічної карти

Студента 2 курсу групи ІП-01  
спеціальності 121 «Інженерія програмного  
забезпечення»

Галько М.В.

\_\_\_\_\_  
(прізвище та ініціали)

Керівник Лебідь С. О.

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

Національна шкала \_\_\_\_\_

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

Члени комісії

Лебідь С. О.

\_\_\_\_\_  
(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-2021 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 ІІІ-01 Семестр 1

**ЗАВДАННЯ  
НА КУРСОВУ РОБОТУ СТУДЕНТУ**

*Галько Мілі Вячеславівні*

(прізвище, ім'я, по батькові)

1. Тема роботи База даних для ведення реєстру земельних ділянок та їх власників з можливістю побудови топологічної карти

керівник роботи Лебідь С. О.

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

2. Строк подання студентом роботи 27.12.2021

3. Вихідні дані до роботи Тексти програмного коду(Додаток А)

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Вступ, аналіз предметної області, постановка завдання, концептуальна модель бази даних, логічна модель бази даних, реалізація бази даних, висновки, перелік посилань

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) ER-діаграма, приклади виконання SQL скриптів, реляційна схема бази даних

6. Дата видачі завдання 31.10.2021

### КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання курсового проекту	Строк виконання етапів проекту	Примітка
1	Вивчення літератури	02.11.2021	
2	Аналіз предметного середовища	05.11.2021	
3	Побудова ER-діаграми	12.11.2021	
4	Побудова реляційної схеми	14.11.2021	
5	Створення бази даних	19.11.2021	
6	Створення користувачів бази даних	25.11.2021	
7	Імпорт даних з використанням засобів СУБД	02.12.2021	
8	Створення запитів до розробленої БД	13.12.2021	
9	Оптимізація роботи запитів	20.12.2021	

**Студент**

*Галько М. В.*

\_\_\_\_\_  
(підпис)

\_\_\_\_\_  
(прізвище та ініціали)

**Керівник роботи**

*Лебідь С. О.*

\_\_\_\_\_  
(підпис)

\_\_\_\_\_  
(прізвище та ініціали)

## ЗМІСТ

Вступ.....	4
1 Аналіз предметної області .....	5
2 Постановка завдання.....	9
3 Концептуальна модель бази даних.....	10
3.1 Побудова ER-діаграми.....	10
3.2 Відношення між об'єктами БД.....	11
4 Логічна модель бази даних.....	14
4.1 Схеми таблиць БД.....	14
4.2 Схеми БД.....	17
5 Реалізація бази даних.....	18
5.1 Створення таблиць.....	18
5.2 Забезпечення цілісності даних... <b>Ошибка! Закладка не определена.</b>	
5.3 Створення обмежень..... <b>Ошибка! Закладка не определена.</b>	
5.4 Створення користувачів ..... <b>Ошибка! Закладка не определена.</b>	
5.5 Імпорт даних у БД.....	24
5.6 Створення запитів .....	26
5.7 Створення процедур .....	35
5.8 Оптимізація запитів .....	41
Висновки .....	45
Перелік посилань.....	46
Додаток А.....	47

## Вступ

На сьогодні є дуже актуальним ведення реєстру земельних ділянок, а саме тема власництва, продажу чи привласнення. Через об'єм даних виникає потреба у зручному збереженні інформації для комфортного надання послуг у даній сфері. Таким чином база даних допоможе у структуризації даної інформації, і є оптимальним варіантом для використання, оскільки є не тільки зручною але й більш швидким варіантом для пошуку, редагування чи видалення інформації. В ході курсової роботи буде реалізована саме така база даних.

Для реалізації поставленої задачі будуть використані наступні інструменти:

- [app.diagrams.net](http://app.diagrams.net) – для створення діаграм. Цей сервіс було обрано, оскільки він повністю безкоштовний, доступний онлайн, має у собі багато інструментів та різноманіття діаграм, та простий в освоєнні;
- Microsoft SQL Server - було обрано в якості системи управління базами даних, оскільки була розглянута у курсі «Бази даних», і практично досліджена мною.
- Moscaooo – допоміжний сайт для генерації даних для їх майбутнього завантаження у створену базу даних.

## 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Необхідно розробити базу даних для ведення реєстру земельних ділянок та їх власників. Відповідно, основною сутністю є Ділянка. Вона має наступні атрибути:

- Унікальний номер;
- Унікальний номер власника ділянки;
- Унікальний номер розташування ділянки;
- Унікальний номер типу використання ділянки;

Власником ділянки може бути звичайна людина (Фізична особа) або ж юридична особа, яку представляє фізичка. В залежності від сутності визначимо атрибути для:

Фізичної особи:

- Унікальний номер;
- Ім'я;
- Прізвище;
- Дата народження;
- Номер телефону;

Юридичної особи:

- Унікальний номер;
- Унікальний номер установи;
- Унікальний номер фізичної особи;

Оскільки для даної бази даних головною сутністю є Ділянка, то для сутності Розташування є важливими лише такі атрибути:

- Унікальний номер;
- Адреса;
- Розмір податку;

Така ж логіка і у формуванні атрибутів для типу використання ділянки:

- Унікальний номер;
- Назва типу;

- Податок;

Оскільки до реалізації також відноситься можливість побудови топологічної карти, то потребуємо сутність Об'єкт для утримання інформації про типографічні дані про прямокутники(куски) ділянки, атрибутами якого є:

- Унікальний номер;
- Унікальний номер ділянки;
- Унікальний номер ресурсу;
- Широта для верхнього лівого кута;
- Висота для верхнього лівого кута;
- Широта для верхнього правого кута;
- Висота для верхнього правого кута;

Для майбутніх покупців ділянок є важливою інформація про ресурси та інженерну комунікацію ділянки. Отже визначимо сутності Ресурси та Інженерна комунікація. Для сутності Ресурси визначимо такі атрибути:

- Унікальний номер;
- Назва;
- Податок за ресурс;

Сутність Інженерна комунікація:

- Унікальний номер;
- Унікальний номер ділянки;
- Водопровід;
- Каналізація;
- Опалення;
- Газ;
- Електроенергія;

Процес купівлі-продажу буде виконаний через реєстратора. Отже зазначимо сутність Реєстратор із атрибутами:

- Унікальний номер;
- Ім'я;

- Прізвище;
- Номер телефону;

Оскільки кількість купівлі-продажу є об'ємною, то створимо сутність Акт прийому передачі, що буде мати у собі атрибути:

- Унікальний номер;
- Унікальний номер ділянки;
- Унікальний номер покупця;
- Унікальний номер продавця;
- Унікальний номер реєстратора;
- Дата заключення акту;

Також варто передбачити наступні обмеження системи:

1. Клієнт повинен бути повнолітнім.
2. Дата заключення акта не може бути пізнішою за сьогоднішню дату.
3. Податок не може бути від'ємним.
4. Усі номери телефонів унікальні.
5. Довгота лівого верхнього кута повинна бути меншою за довготу правого нижнього.
6. Широта лівого верхнього кута повинна бути меншою за довготу правого нижнього.
7. Покупець та продавець у акті не може бути однією й тою самою людиною.

Для реалізації корисності БД необхідно розробити наступні запити та реалізувати їх на мові SQL:

1. Відображення кількості об'єктів з яких складається ділянки.
2. Відображення загальної площі усіх ділянок, що є у базі та їх кількості.
3. Відображення ділянок, власниками яких є юридичні особи.
4. Відображення ділянок, власниками яких є фізичні особи.
5. Відображення актів, де покупець був юридична особа.



6. Відображення актів, де продавець був фізична особа.
7. Відображення затверджених актів за останній рік.
8. Відображення ресурсів певного власника ділянки.
9. Відображення ділянок без інженерних комунікацій.
10. Відображення осіб з найбільшою площею ділянки.
11. Відображення історії купівлі-продажу ділянки.
12. Відображення усіх ділянок із каналізацією.
13. Відображення ділянок, де є опалення.
14. Відображення власників ділянки із електроенергією.
15. Відображення кількості ділянок із газом.
16. Відображення ділянок у яких був лише 1 власник.
17. Відображення актів затвердженими реєстратором.
18. Відображення суми вартості ділянок фізичних осіб.
19. Відображення номеру телефону продавця та його ділянки.
20. Відображення кількості актів проведених із ділянкою.

Отже, в цьому розділі ми визначили основні вимоги до бази даних, обмеження нашої системи, а також основні запити, що потребують реалізації.

## 2 ПОСТАНОВКА ЗАВДАННЯ

Необхідно спроектувати базу даних для ведення реєстру земельних ділянок. Для цього необхідно виконати наступні кроки:

- побудувати ER-діаграму та схему БД;
- створити та заповнити БД записами;
- розробити скрипти та запити на мові SQL;
- оптимізувати роботу скриптів для мінімізації часу.

Також необхідними умовами є використання тригерів, генераторів та представлень.

Кількість таблиць, які використовуються у кожному із запитів має бути не менше двох, а загальна кількість сутностей – не менше 10.

Таким чином, було визначено основні кроки та вимоги для проектування бази даних.

### 3 КОНЦЕПТУАЛЬНА МОДЕЛЬ БАЗИ ДАНИХ

#### 3.1 Побудова ER-діаграми

Дана БД має 12 сутностей. Їх назви та призначення наведено у таблиці 3.1

Таблиця 3.1 – Назви сутностей та їх призначення

Сутність	Набір атрибутів
Ділянка (Land)	<ul style="list-style-type: none"> <li>- ID</li> <li>- ID власника</li> <li>- ID розташування</li> <li>- ID типу використання</li> </ul>
Фізична особа (Natural)	<ul style="list-style-type: none"> <li>- ID</li> <li>- Ім'я</li> <li>- Прізвище</li> <li>- Дата народження</li> <li>- Номер телефону</li> </ul>
Юридична особа (Legal)	<ul style="list-style-type: none"> <li>- ID</li> <li>- ID установи</li> <li>- ID фізичного представника</li> </ul>
Розташування (Location)	<ul style="list-style-type: none"> <li>- ID</li> <li>- Адреса</li> <li>- Податок</li> </ul>
Тип використання ділянки (UsageType)	<ul style="list-style-type: none"> <li>- ID</li> <li>- Назва типу</li> <li>- Податок</li> </ul>
Об'єкт (Object)	<ul style="list-style-type: none"> <li>- ID</li> <li>- ID ділянки</li> <li>- ID ресурсу</li> <li>- Широта нижнього лівого кута</li> <li>- Широта верхнього правого кута</li> <li>- Висота нижнього лівого кута</li> </ul>

Продовження таблиці 3.1

Назва	Обумовленість
Об'єкт (Object)	- Висота верхнього правого кута
Ресурс (Resource)	- ID - Назва - Податок
Інженерна комунікація (Utility)	- ID - ID ділянки - Водопровід - Каналізація - Опалення - Газ - Електроенергія
Реєстратор (Registrar)	- ID - Ім'я - Прізвище - Номер телефону
Акт прийому-передачі (Act)	- ID - ID ділянки - ID покупця - ID продавця - ID реєстратора - Дата заключення акту

### 3.2 Відношення між об'єктами БД

Зв'язки між сутностями наведено у таблиці 3.2

Таблиця 3.2 – Зв'язки між сутностями

Перша сутність	Друга сутність	Зв'язок
Land	UsageType	1:1 (fk_usageType)

Продовження таблиці 3.2

Перша сутність	Друга сутність	Зв'язок
Land	Location	1:1 (fk_location)
Land	Utility	1:1 (fk_utility)
Land	Object	1 : many (fk_land)
Land	Act	1 : many (fk_land)
Resource	Object	1 : many (fk_resource)
Natural	Land	1 : many (fk_natural)
Natural	Legal	1:1 (fk_natural)
Natural	Act	1 : many (fk_natural)
Registrar	Act	1 : many (fk_registrar)

Створена ER-діаграма на основі відношень між об'єктами бази даних та таблиці сутностей зображена на рис. 3.1

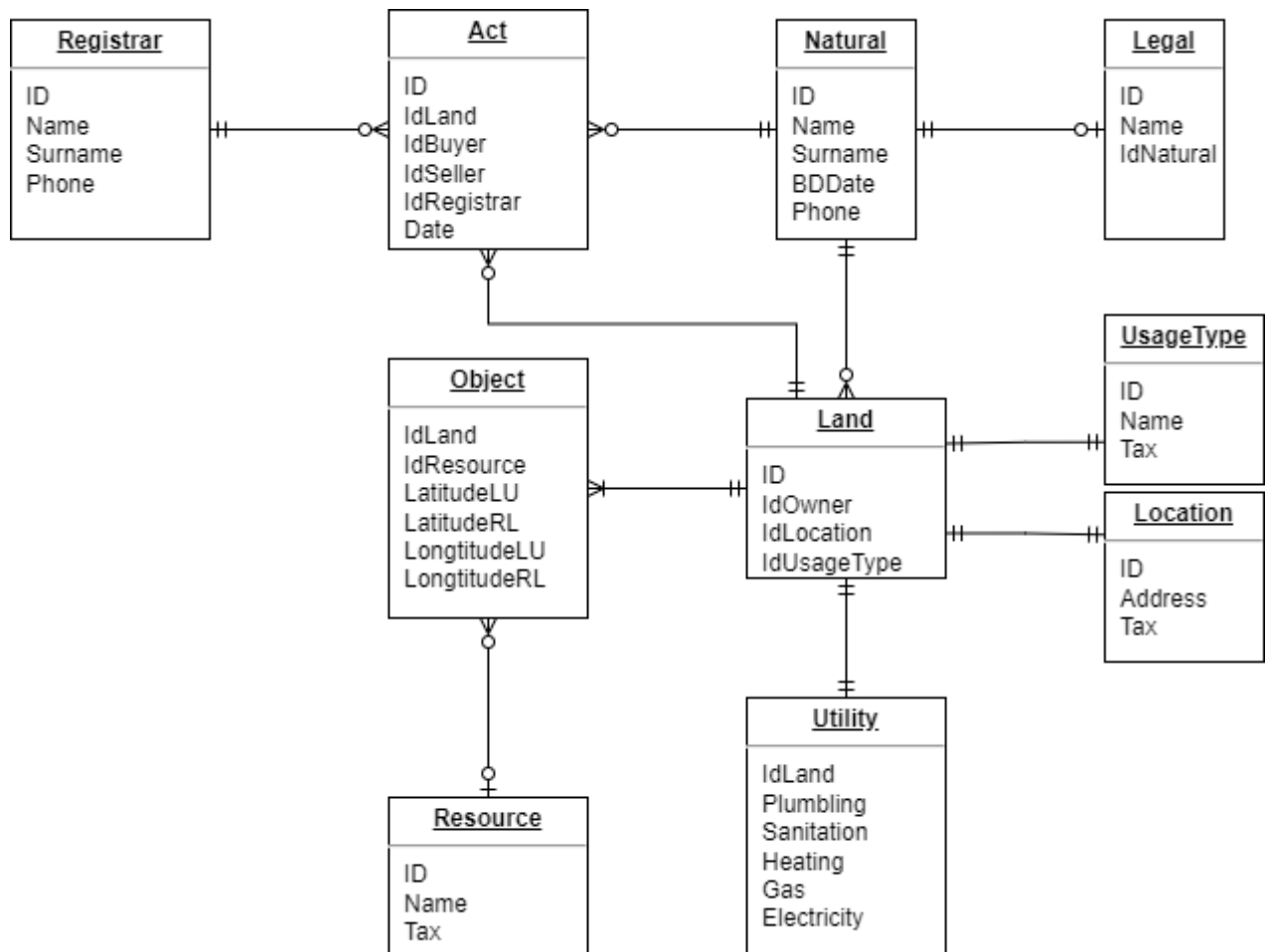


Рисунок 3.1 – ER-діаграма

В ході визначення основних сутностей, їх атрибутів та зв'язків між ними були реалізовані таблиця сутностей, таблиця відношень сутностей та ER-діаграма. Далі дана визначена модель є базою для створення бази даних.

## 4 ЛОГІЧНА МОДЕЛЬ БАЗИ ДАНИХ

### 4.1 Схеми таблиць бази даних

Таблиця 4.1 – Схеми таблиці Land

Атрибут	Тип даних
(PK) ID	INT
(FK) IdOwner	INT
(FK) IdLocation	INT
(FK) IdUsageType	INT

Таблиця 4.2 – Схеми таблиці Natural

Атрибут	Тип даних
(PK) ID	INT
Name	VARCHAR(20)
Surname	VARCHAR(20)
BDDate	DATE
Phone	VARCHAR(15)

Таблиця 4.3 – Схеми таблиці Location

Атрибут	Тип даних
(PK) ID	INT
Address	VARCHAR(50)
Tax	REAL

Таблиця 4.4 – Схеми таблиці UsageType

Атрибут	Тип даних
(PK) ID	INT
Name	VARCHAR(50)
Tax	REAL

Таблиця 4.5 – Схема таблиці Object

Атрибут	Тип даних
(FK) IdLand	INT
(FK) IdResource	INT
LatitudeLU	REAL
LatitudeRL	REAL
LongitudeLU	REAL
LongitudeRL	REAL

Таблиця 4.6 – Схема таблиці Resource.

Атрибут	Тип даних
(PK) ID	INT
Name	VARCHAR(50)
TAX	REAL

Таблиця 4.7 – Схема таблиці Utility

Атрибут	Тип даних
(FK) IdLand	INT
Plumbing	BIT
Sanitation	BIT
Heating	BIT
Gas	BIT
Electricity	BIT

Таблиця 4.8 – Схема таблиці Legal

Атрибут	Тип даних
(PK) ID	INT
Name	VARCHAR(50)
(FK) IdNatural	INT



Таблиця 4.9 – Схема таблиці Act

Атрибут	Тип даних
(PK) ID	INT
(FK) IdLand	INT
(FK) IdBuyer	INT
(FK) IdSeller	INT
(FK) IdRegistrar	INT
Date	DATE

Таблиця 4.10 – Схема таблиці Registrar

Атрибут	Тип даних
(PK) ID	INT
Name	VARCHAR(20)
Surname	VARCHAR(20)
Phone	VARCHAR(15)

## 4.2 Схема БД

Після створення схем таблиць, базуючись на ER-моделі та схемах таблиць можна спроектувати схему бази даних (рис 4.1).

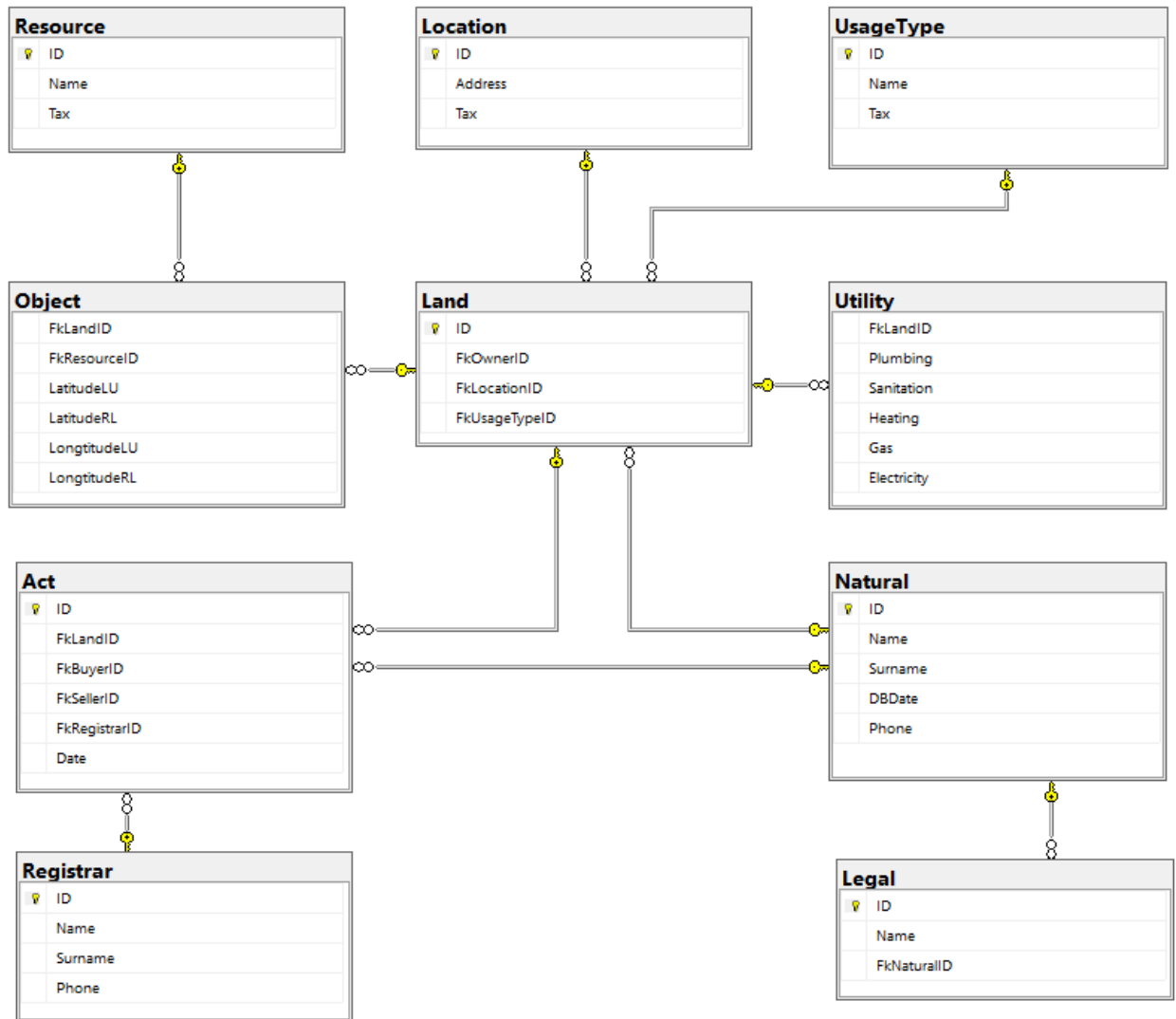


Рисунок 4.1 – Схема бази даних

## 5 РЕАЛІЗАЦІЯ БАЗИ ДАНИХ

### 5.1 Створення таблиць

Для реалізації БД в середовищі MS SQL Server, треба створити базу даних, а також таблиці. Для створення таблиць, потрібно написати спеціальний скрипт, де буде вказано назва таблиці, її поля, типи полів. Скрипт для створення бази даних та її таблиць:

```
CREATE DATABASE LandCompany;  
GO  
USE master;  
USE LandCompany;  
GO
```

```
CREATE TABLE UsageType (  
    ID INT IDENTITY(1,1) PRIMARY KEY,  
    Name VARCHAR(50) NOT NULL,  
    Tax REAL DEFAULT(0) NOT NULL  
)  
GO
```

```
ALTER TABLE UsageType  
ADD CONSTRAINT CH_UsageType_Tax CHECK (Tax >= 0)  
GO
```

```
CREATE TABLE Location (  
    ID INT IDENTITY(1,1) PRIMARY KEY,  
    Address VARCHAR(50) NOT NULL UNIQUE,  
    Tax REAL DEFAULT(0) NOT NULL  
)  
GO
```

```
ALTER TABLE Location  
ADD CONSTRAINT CH_Location_Tax CHECK (Tax >= 0)  
GO
```

```
CREATE TABLE Resource (  
    ID INT IDENTITY(1,1) PRIMARY KEY,  
    Name VARCHAR(50) NULL,  
    Tax REAL DEFAULT(0) NOT NULL  
)
```

GO

```
ALTER TABLE Resource
ADD CONSTRAINT CH_Resource_Tax CHECK (Tax >= 0)
GO
```

```
CREATE TABLE Natural (
    ID INT IDENTITY(1,1) PRIMARY KEY,
    Name VARCHAR(20) NOT NULL,
    Surname VARCHAR(20) NOT NULL,
    DBDate DATE NOT NULL,
    Phone VARCHAR(15) NOT NULL UNIQUE
)
GO
```

```
ALTER TABLE Natural
ADD CONSTRAINT CH_Natural_Name
CHECK (Name NOT LIKE '%[^A-Za-z" -]%' )
GO
```

```
ALTER TABLE Natural
ADD CONSTRAINT CH_Natural_Surname
CHECK (Surname NOT LIKE '%[^A-Za-z" -]%' )
GO
```

```
ALTER TABLE Natural
ADD CONSTRAINT CH_Natural_DBDate
CHECK (DBDate LIKE '____-__-__' AND DBDate NOT LIKE '%[a-Z]%' )
GO
```

```
ALTER TABLE Natural
ADD CONSTRAINT CH_Natural_Phone
CHECK (Phone LIKE '____-____-____' AND Phone NOT LIKE '%[a-Z]%' )
GO
```

```
CREATE TABLE Registrar (
    ID INT IDENTITY(1,1) PRIMARY KEY,
    Name VARCHAR(20) NOT NULL,
    Surname VARCHAR(20) NOT NULL,
    Phone VARCHAR(15) NOT NULL UNIQUE
)
```

GO

```
ALTER TABLE Registrar
ADD CONSTRAINT CH_Registrar_Name
CHECK (Name NOT LIKE '%[^A-Za-z" -]%)
GO
```

```
ALTER TABLE Registrar
ADD CONSTRAINT CH_Registrar_Surname
CHECK (Surname NOT LIKE '%[^A-Za-z" -]%)
GO
```

```
ALTER TABLE Registrar
ADD CONSTRAINT CH_Registrar_Phone
CHECK (Phone LIKE '____-____-____' AND Phone NOT LIKE '%[a-Z]%)
GO
```

```
CREATE TABLE Legal (
    ID INT IDENTITY(1,1) PRIMARY KEY,
    Name VARCHAR(50) NOT NULL,
    FkNaturalID INT NOT NULL
)
GO
```

```
ALTER TABLE Legal
WITH CHECK ADD CONSTRAINT FK_Natural_ID FOREIGN KEY(FkNaturalID)
REFERENCES Natural(ID)
ON UPDATE CASCADE
ON DELETE CASCADE
GO
```

```
CREATE TABLE Land (
    ID INT IDENTITY(1,1) PRIMARY KEY,
    FkOwnerID INT NULL,
    FkLocationID INT NOT NULL,
    FkUsageTypeID INT NULL
)
GO
```

```
ALTER TABLE Land
WITH CHECK ADD CONSTRAINT FK_Owner_ID FOREIGN KEY(FkOwnerID)
```

```
REFERENCES Natural(ID)
ON UPDATE CASCADE
GO
```

```
ALTER TABLE Land
WITH CHECK ADD CONSTRAINT FK_Location_ID FOREIGN KEY(FkLocationID)
REFERENCES Location(ID)
ON UPDATE CASCADE
ON DELETE CASCADE
GO
```

```
ALTER TABLE Land
WITH CHECK ADD CONSTRAINT FK_UsageType_ID FOREIGN KEY(FkUsageTypeID)
REFERENCES UsageType(ID)
ON UPDATE CASCADE
ON DELETE SET NULL
GO
```

```
CREATE TABLE Utility (
    FkLandID INT NULL,
    Plumbing BIT DEFAULT(0) NOT NULL,
    Sanitation BIT DEFAULT(0) NOT NULL,
    Heating BIT DEFAULT(0) NOT NULL,
    Gas BIT DEFAULT(0) NOT NULL,
    Electricity BIT DEFAULT(0) NOT NULL
)
GO
```

```
ALTER TABLE Utility
WITH CHECK ADD CONSTRAINT FK_Land_ID FOREIGN KEY(FkLandID)
REFERENCES Land(ID)
ON UPDATE CASCADE
ON DELETE SET NULL
GO
```

```
CREATE TABLE Object (
    FkLandID INT NOT NULL,
    FkResourceID INT NULL,
    LatitudeLU REAL NOT NULL,
    LatitudeRL REAL NOT NULL,
    LongitudeLU REAL NOT NULL,
```

```

        LongitudeRL REAL NOT NULL
    )
GO

```

```

ALTER TABLE Object
WITH CHECK ADD CONSTRAINT FK_Object_Land_ID FOREIGN KEY(FkLandID)
REFERENCES Land(ID)
ON UPDATE CASCADE
ON DELETE CASCADE
GO

```

```

ALTER TABLE Object
WITH CHECK ADD CONSTRAINT FK_Resource_ID FOREIGN KEY(FkResourceID)
REFERENCES Resource(ID)
ON UPDATE CASCADE
ON DELETE SET NULL
GO

```

```

ALTER TABLE Object
ADD CONSTRAINT CH_Latitude CHECK (LatitudeLU < LatitudeRL)
GO

```

```

ALTER TABLE Object
ADD CONSTRAINT CH_Longitude CHECK (LongitudeLU < LongitudeRL)
GO

```

```

CREATE TABLE Act (
    ID INT IDENTITY(1,1) PRIMARY KEY,
    FkLandID INT NOT NULL,
    FkBuyerID INT NOT NULL,
    FkSellerID INT NOT NULL,
    FkRegistrarID INT NOT NULL,
    Date DATE NOT NULL
)
GO

```

```

ALTER TABLE Act
WITH CHECK ADD CONSTRAINT FK_Act_Land_ID FOREIGN KEY(FkLandID)
REFERENCES Land(ID)
ON UPDATE CASCADE
GO

```

```
ALTER TABLE Act
WITH CHECK ADD CONSTRAINT FK_Buyer_ID FOREIGN KEY(FkBuyerID)
REFERENCES Natural(ID)
ON DELETE NO ACTION
GO
```

```
ALTER TABLE Act
WITH CHECK ADD CONSTRAINT FK_Seller_ID FOREIGN KEY(FkSellerID)
REFERENCES Natural(ID)
ON DELETE NO ACTION
GO
```

```
ALTER TABLE Act
WITH CHECK ADD CONSTRAINT FK_Registrar_ID FOREIGN KEY(FkRegistrarID)
REFERENCES Registrar(ID)
ON UPDATE CASCADE
ON DELETE NO ACTION
GO
```



## 6 ІМПОРТ ДАНИХ У БД

Імпортування даних у СУБД Microsoft SQL Server реалізується декількома способами. Один із них - безпосереднє додавання даних через команду insert SQL у вигляді скриптів. Інший же полягає у імпортуванні даних через файли форматів .csv або .json. Звичайно, для великого обсягу даних підійде другий спосіб. Але оскільки дана робота більш демонструє по своїй суті – уміння опрацьовування дані, то загальний об'єм даних не буде більшим за 100 запитів. Однак для загального прикладу використаємо обидва способи.

Имя	Тип
Act.sql	Microsoft SQL Ser...
Land.sql	Microsoft SQL Ser...
Legal.csv	Файл Microsoft Ex...
Location.sql	Microsoft SQL Ser...
Natural.csv	Файл Microsoft Ex...
Object.sql	Microsoft SQL Ser...
Registrar.csv	Файл Microsoft Ex...
Resource.sql	Microsoft SQL Ser...
UsageType.sql	Microsoft SQL Ser...
Utility.sql	Microsoft SQL Ser...

Рисунок 6.1 – sql та csv файли з даними таблиць

```

1 insert into Act (FkLandID, FkBuyerID, FkSellerID, FkRegistrarID, Date) values (1, 23, 14, 12, '2020/10/24');
2 insert into Act (FkLandID, FkBuyerID, FkSellerID, FkRegistrarID, Date) values (1, 14, 17, 12, '2021/11/12');
3 insert into Act (FkLandID, FkBuyerID, FkSellerID, FkRegistrarID, Date) values (2, 33, 21, 19, '2019/11/13');
4 insert into Act (FkLandID, FkBuyerID, FkSellerID, FkRegistrarID, Date) values (3, 5, 32, 13, '2020/12/13');
5 insert into Act (FkLandID, FkBuyerID, FkSellerID, FkRegistrarID, Date) values (3, 32, 35, 14, '2021/11/15');
6 insert into Act (FkLandID, FkBuyerID, FkSellerID, FkRegistrarID, Date) values (4, 6, 17, 5, '2021/04/20');
7 insert into Act (FkLandID, FkBuyerID, FkSellerID, FkRegistrarID, Date) values (5, 44, 21, 2, '2019/03/23');
8 insert into Act (FkLandID, FkBuyerID, FkSellerID, FkRegistrarID, Date) values (5, 21, 17, 10, '2020/09/21');
9 insert into Act (FkLandID, FkBuyerID, FkSellerID, FkRegistrarID, Date) values (6, 12, 24, 4, '2021/04/12');
10 insert into Act (FkLandID, FkBuyerID, FkSellerID, FkRegistrarID, Date) values (6, 24, 38, 18, '2021/07/21');
11 insert into Act (FkLandID, FkBuyerID, FkSellerID, FkRegistrarID, Date) values (7, 12, 34, 3, '2020/06/14');
12 insert into Act (FkLandID, FkBuyerID, FkSellerID, FkRegistrarID, Date) values (7, 34, 39, 13, '2020/10/04');
13 insert into Act (FkLandID, FkBuyerID, FkSellerID, FkRegistrarID, Date) values (8, 50, 43, 8, '2021/05/25');
14 insert into Act (FkLandID, FkBuyerID, FkSellerID, FkRegistrarID, Date) values (9, 25, 12, 1, '2021/01/13');
15 insert into Act (FkLandID, FkBuyerID, FkSellerID, FkRegistrarID, Date) values (9, 12, 24, 19, '2021/03/27');
16 insert into Act (FkLandID, FkBuyerID, FkSellerID, FkRegistrarID, Date) values (9, 24, 2, 6, '2021/05/26');

```

Рисунок 6.2 - Скрипти для вставки даних у БД

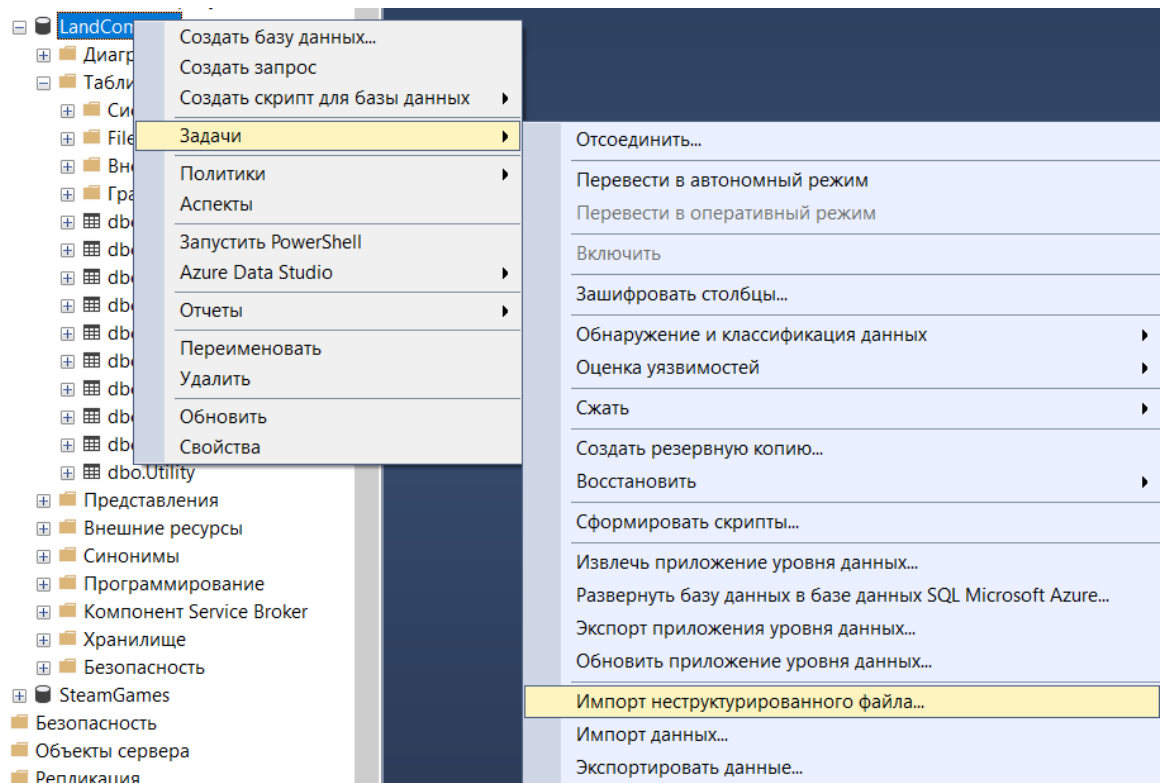


Рисунок 6.3 – функція імпорту даних через файл

Як видно з рисунка 6.1 файли: Legal, Natural, Registrar у форматі .csv, а отже були використані як джерело (Flat file source) при імпорті даних (рис. 6.3) у таблиці. Інші ж файли мають у собі скрипт із вхідними даними, подібно рисунку 6.2.

## 7 СТВОРЕННЯ ЗАПИТІВ

Для підтримки роботи реєстру земельних ділянок, а саме його бази даних, необхідно розробити запити, які вже були зазначені у розділі «Аналіз предметної області». Спершу будемо зазначати функцію запита, далі його розроблений скрипт з подальшим прикладом використання.

1. Запит для відображення загальної кількості об'єктів (складова ділянки) для кожної з ділянок:

```
create view landsAndNumberOfObjects as
select la.ID as LandID, lo.Address as LandAddress, COUNT(*) as ObjectsNumber
from Land la
join Object o on o.FkLandID = la.ID
join Location lo on lo.ID = la.FkLocationID
group by la.ID, lo.Address
go
```

Рисунок 7.1 – Скрипт для відображення загальної кількості об'єктів для кожної з ділянок

	LandID	LandAddress	ObjectsNumber
1	1	8787 Cambridge Alley	2
2	2	55735 Menomonie Hill	2
3	3	0 3rd Street	2
4	4	828 Monterey Alley	2
5	5	85032 Rigney Alley	2
6	6	93694 Pierstorff Point	2
7	7	39 Westend Pass	2
8	8	8 Charing Cross Circle	2
9	9	335 Northport Lane	2
10	10	4821 Loftsgordon Junction	2
11	11	2 Gateway Plaza	2
12	12	2985 Cody Center	2
13	13	48389 Marquette Alley	4
14	15	9119 Farragut Junction	2
15	16	0895 Little Fleur Street	2
16	17	6273 Debra Street	2
17	20	3980 Stuart Crossing	4

Рисунок 7.2 – Результат запиту

2. Запит для відображення загальної площі усіх ділянок, що є у базі та їх кількості:

```

create view totalArea as
select (select count(*) from Land) as TotalLandsQuantity,
       (select sum((o.LatitudeRL - o.LatitudeLU)*(o.LongitudeRL - o.LongitudeLU)) from Object o) as TotalArea
go

```

Рисунок 7.3 – Скрипт для відображення загальної площі усіх ділянок, що є у базі та їх кількості

	TotalLandsQuantity	TotalArea
1	17	1638,00182938576

Рисунок 7.4 – Результат запиту

3. Запит для відображення ділянок, власниками яких є юридичні особи:

```

create view Legallands as
select
  la.ID as LandID,
  n.Surname + ' ' + n.Name as Owner, le.Name as CompanyName
from Natural n
join Legal le on le.FkNaturalID = n.ID
join Land la on la.FkOwnerID = n.ID
go

```

Рисунок 7.5 – Скрипт для відображення ділянок, власниками яких є юридичні особи

	LandID	Owner	CompanyName
1	16	Robardley Derby	MacGyver-Beatty
2	4	Gristhwaite Moss	Kshlerin LLC
3	2	Moralis Alfie	Spinka
4	7	Bockett Arri	Osinski and Skiles

Рисунок 7.6 – Результат запиту

4. Запит для відображення ділянок, власниками яких є фізичні особи:

```

create view NaturalLands as
select
  n.ID as NaturalID, n.Surname + ' ' + n.Name as Natural, la.ID as LandID
from Natural n
join Land la on la.FkOwnerID = n.ID
where n.ID not in (select FkNaturalID from Legal)
go

```

Рисунок 7.7 – Скрипт для відображення ділянок, власниками яких є фізичні особи

	NaturalID	Natural	LandID
1	2	Persicke Sancho	1
2	3	Polak Abbey	5
3	3	Polak Abbey	9
4	13	Mugg Brendin	17
5	28	Crufts Leupold	20
6	31	Groom Brok	15
7	32	Coyte Feliza	10
8	34	Tupp Connie	3
9	39	Colwill Lodovico	8
10	42	Dyment Gael	13
11	44	Mecco Wash	12
12	45	Blinman Georgianna	6
13	51	Ben-Aharon Sidoney	11

Рисунок 7.8 – Результат запиту

5. Запит для відображення актів, де покупець був юридична особа:

```

create view LegalBuyer as
select
n.ID as BuyerID, n.Surname + ' ' + n.Name as Buyer, a.id as ActID, l.ID as LegalID, l.Name as LegalName
from Natural n
join Legal l on n.ID = l.FkNaturalID
join Act a on a.FkBuyerID = n.ID
go

select * from LegalBuyer
order by BuyerID
go

```

Рисунок 7.9 – Скрипт для відображення актів, де покупець був юридична особа

	BuyerID	Buyer	ActID	LegalID	LegalName
1	4	Robardley Derby	27	4	MacGyver-Beatty
2	6	Brafield Prentice	6	5	Becker-Murphy
3	7	Wemyss Kliment	23	6	McCullough
4	14	Ault Vania	2	7	Jerde
5	17	Speak Rosella	26	9	Tromp
6	17	Speak Rosella	33	9	Tromp
7	21	Garard Sanford	8	11	Trantow
8	21	Garard Sanford	22	11	Trantow
9	24	Paradine Bartel	10	13	Kling
10	24	Paradine Bartel	16	13	Kling
11	33	Stadden Culley	3	16	Leffler-Balistreri
12	33	Stadden Culley	17	16	Leffler-Balistreri
13	33	Stadden Culley	30	16	Leffler-Balistreri
14	36	Latan Obie	28	18	Kautzer-Boyle
15	40	Grzesiewicz M...	24	20	Boyer-Schoen
16	50	Soar Philis	13	23	Schiller and Th...

Рисунок 7.10 – Результат запиту

6. Запит на відображення актів, де продавець був фізична особа:

```

create view NaturalBuyer as
select
n.ID as BuyerID, n.Surname + ' ' + n.Name as Buyer, a.id as ActID
from Natural n
join Act a on a.FkBuyerID = n.ID
where n.ID not in (select FkNaturalID from Legal)
go

select * from NaturalBuyer
order by BuyerID
go

```

Рисунок 7.11 – Скрипт для відображення актів, де продавець був фізична особа

	BuyerID	Buyer	ActID
1	5	Tutton Arty	4
2	11	De Filippis Daffi	18
3	12	Wiseman Meredith	9
4	12	Wiseman Meredith	11
5	12	Wiseman Meredith	15
6	16	Kolodziej Ewart	29
7	19	Ceyssen Lorenzo	25
8	23	Dacombe Clari	1
9	25	Harbron Gennifer	14
10	25	Harbron Gennifer	32

Рисунок 7.12 – Результат запиту

7. Запит на відображення затверджених актів за останній рік:

```

create view lastYearActs as
select
a.ID as ActID, a.Date as RegDate, l.id
from Act a
join Land l on l.id = a.FkLandID
where a.Date >= DATEADD(year, -1, GETDATE())
go

select * from lastYearActs
order by RegDate
go

```

Рисунок 7.13 – Скрипт для відображення затверджених актів за останній рік

	ActID	RegDate	id
1	25	2021-02-26	13
2	15	2021-03-27	9
3	21	2021-03-30	12
4	9	2021-04-12	6
5	6	2021-04-20	4
6	13	2021-05-25	8
7	16	2021-05-26	9
8	10	2021-07-21	6
9	29	2021-09-21	16
10	30	2021-10-16	16
11	26	2021-11-06	15
12	2	2021-11-12	1
13	5	2021-11-15	3
14	22	2021-12-27	12
15	27	2021-12-29	15

Рисунок 7.14 – Результат запиту

8. Запит на відображення ресурсів певного власника ділянки:

```

create proc ResourcesbyOwnerID @ownerID int
as begin
select distinct r.Name as Resources
from Resource r
where r.ID in(select o.FkResourceID from Object o
              where o.FkLandID in(select l.id from Land l
                                   where l.FkOwnerID = @ownerID))
order by r.Name
end
go

```

Рисунок 7.15 – Скрипт для відображення ресурсів певного власника

ділянки

	Resources
1	Asteraceae
2	Ectolechiaceae
3	Parmeliaceae

Рисунок 7.16 – Результат запиту

9. Запит на відображення ділянок без інженерних комунікацій:

```

select la.id, lo.Address
from Land la
join Location lo on lo.ID = la.FkLocationID
join Utility u on FkLandID = la.id
where Plumbing = 0 and Sanitation = 0 and Heating = 0 and Gas = 0
go

```

Рисунок 7.17 – Скрипт для відображення ділянок без інженерних комунікацій

	id	Address
1	2	55735 Menomonie Hill
2	6	93694 Pierstorff Point
3	9	335 Northport Lane
4	13	48389 Marquette Alley
5	15	9119 Farragut Junction

Рисунок 7.18 – Результат запиту

#### 10. Запит на відображення осіб з найбільшою площею ділянки:

```

alter proc TopOwnersByArea @topNumber int as
begin
    declare @table table(ID int, name varchar(50), area real)
    declare @id int, @name varchar(50), @surname varchar(50), @area real
    declare cur cursor local
    for select ID, Name, Surname from Natural

    open cur
    fetch next from cur into @id, @name, @surname

    while @@FETCH_STATUS = 0
    begin
        set @area = (select sum((o.LatitudeRL - o.LatitudeLU)*(o.LongitudeRL - o.LongitudeLU)) from Object o
                     where o.FkLandID in (select ID from Land l
                     where l.FkOwnerID = @id))

        insert into @table (ID, name, area) values (@id, @surname + ' ' + @name, @area)
        fetch next from cur into @id, @name, @surname
    end
    close cur

    select top(@topNumber) * from @table order by area desc
end
go

exec TopOwnersByArea 5
go

```

Рисунок 7.19 – Скрипт для відображення осіб з найбільшою площею ділянки:

	ID	name	area
1	45	Blinman Georgianna	710,1069
2	28	Crufts Leupold	173,1978
3	3	Polak Abbey	147,1335
4	46	Bockett Arri	105,9729
5	34	Tupp Connie	86,13688



Рисунок 7.20 – Результат запиту

11. Запит на відображення усіх ділянок із каналізацією:

```
select la.id, lo.Address
from Land la
join Location lo on lo.ID = la.FkLocationID
join Utility u on FkLandID = la.id
where Sanitation = 1
go
```

Рисунок 7.23 – Скрипт для відображення усіх ділянок із каналізацією

	id	Address
1	3	0 3rd Street
2	4	828 Monterey Alley
3	5	85032 Rigney Alley
4	16	0895 Little Fleur Street

Рисунок 7.24 – Результат запиту

12. Запит на відображення історії купівлі-продажу ділянки:

Рисунок 7.21 – Скрипт для відображення

Рисунок 7.22 – Результат запиту

13. Запит на відображення ділянок, де є опалення:

```
select la.id, lo.Address
from Land la
join Location lo on lo.ID = la.FkLocationID
join Utility u on FkLandID = la.id
where Heating = 1
go
```

Рисунок 7.25 – Скрипт для відображення ділянок, де є опалення

	id	Address
1	3	0 3rd Street
2	4	828 Monterey Alley
3	5	85032 Rigney Alley
4	7	39 Westend Pass
5	8	8 Charing Cross Circle
6	11	2 Gateway Plaza
7	16	0895 Little Fleur Street

Рисунок 7.26 – Результат запиту

14. Запит на відображення власників ділянки із електроенергією:

```
select la.id as LandID, n.Surname + ' ' + n.Name as Owner
from Land la
join Natural n on n.ID = la.FkOwnerID
join Utility u on FkLandID = la.id
where Electricity = 1
go
```

Рисунок 7.27 – Скрипт для відображення власників ділянки із електроенергією

	LandID	Owner
1	2	Moralis Alfie
2	4	Grithwaite Moss
3	6	Blinman Georgianna
4	8	Colwill Lodovico
5	10	Coyte Feliza
6	12	Mecco Wash
7	16	Robardley Derby
8	20	Crufts Leupold

Рисунок 7.28 – Результат запиту

15. Запит на відображення кількості ділянок із газом:

```
select 'Full number of lands with electricity is ' +
cast((select count(*)
from Land l
join Utility on fkLandID = l.ID
where Gas = 1) as varchar) as GasQuantity
go
```

Рисунок 7.29 – Скрипт для відображення кількості ділянок із газом

	GasQuantity
1	Full number of lands with electricity is 7

Рисунок 7.30 – Результат запиту

16. Запит на відображення ділянок у яких був лише 1 власник:

Рисунок 7.31 – Скрипт для відображення ділянок у яких був лише 1 власник

Рисунок 7.32 – Результат запиту

17. Запит на відображення актів затвердженими реєстратором:

Рисунок 7.33 – Скрипт для відображення актів затвердженими  
реєстратором

Рисунок 7.34 – Результат запиту

18. Запит на відображення суми вартості ділянок фізичних осіб:

Рисунок 7.35 – Скрипт для відображення суми вартості ділянок  
фізичних осіб

Рисунок 7.36 – Результат запиту

19. Запит на відображення номеру телефону продавця та його ділянки:

Рисунок 7.37 – Скрипт для відображення номеру телефону продавця та  
його ділянки

Рисунок 7.38 – Результат запиту

20. Запит на відображення кількості актів проведених із ділянкою:

Рисунок 7.39 – Скрипт для відображення кількості актів проведених із  
ділянкою

Рисунок 7.40 – Результат запиту

8. Клієнт повинен бути повнолітнім.
9. Дата заключення акта не може бути пізнішою за сьогоднішню дату.
10. Дати вказуються в часовому поясі UTC+0.
11. Податок не може бути від'ємним.
12. Усі номери телефонів унікальні.
13. Довгота лівого верхнього кута повинна бути меншою за довготу правого нижнього.
14. Широта лівого верхнього кута повинна бути меншою за довготу правого нижнього.

Для реалізації корисності БД необхідно розробити наступні запити та реалізувати їх на мові SQL:

Отже, в цьому розділі ми визначили основні вимоги до бази даних, обмеження нашої системи, а також основні запити, що потребують реалізації.

## 7.1 Створення процедур

Для більш гнучких запитів при обранні певних категорій, типів, тощо було створено модифікації для деяких з вище наведених скриптів.

1. Обрати клієнта по id певного контракту(рис 5.53). Приклад виклику та роботи процедури представлено на рисунку 5.54.

```
DELIMITER $$
CREATE PROCEDURE select_client_by_contract(searched_contract_id INT)
BEGIN
    select client.client_id, client.first_name, client.last_name, client.phone_number, contract.contract_id,
    contract.cost_of_tour, payment_status.name_of_payment_status from contract
    inner join client on client.client_id = contract.client_id
    inner join payment on payment.payment_id = contract.contract_id AND contract_id = searched_contract_id
    inner join payment_status on payment_status.payment_status_id = payment.payment_status_id;
END $$
DELIMITER ;
```

Рисунок 7.1 – Скрипт процедури для обрання клієнта по id контракту

16 • `CALL select_client_by_contract(5);`

client_id	first_name	last_name	phone_number	contract_id	cost_of_tour	name_of_payment_status
2	Zilvia	Thon	(295) 7853169	5	1532.3	Accepted

Рисунок 7.2 – Приклад виклику та роботи процедури для обрання клієнта по id контракту рівному 5

2. Вивести топ – n найпопулярніших країн для мандрівки(рис 5.55) Приклад виклику та роботи процедури представлено на рисунку 5.56.

```
DELIMITER $$
CREATE PROCEDURE select_top_for_traveling(top INT)
BEGIN
SELECT place_of_rest.country, COUNT(*) AS total FROM tour
inner join place_of_rest on place_of_rest.rest_place_id = tour.place_of_rest_id
GROUP BY country ORDER BY total DESC LIMIT top;
END $$
DELIMITER ;
```

Рисунок 7.3 – Вивести топ – n найпопулярніших країн

30 • `CALL select_top_for_traveling(5);`  
31

country	total
Indonesia	21
China	20
Philippines	9
Bangladesh	4
Russia	4

Рисунок 7.4 – Приклад виклику та роботи процедури для виведення топ – 5 найпопулярніших країн

3. Обрати всі місця відпочинку з певною дієтою та певною країною (рис 5.37) Приклад виклику та роботи процедури представлено на рисунку 5.58

```

DELIMITER $$
CREATE PROCEDURE select_tour_by_diet(diet_id INT, country VARCHAR(45))
BEGIN
    select place_of_rest.city, diet_type.name_of_diet_type, cost from place_of_rest
    inner join diet_type on diet_type.diet_type_id = place_of_rest.type_of_diet_id AND
    place_of_rest.type_of_diet_id = diet_id AND place_of_rest.country = country;
END $$
DELIMITER ;

```

Рисунок 7.5 – Обрати всі місця відпочинку з певною дієтою та певною країною

3 • CALL select\_tour\_by\_diet(6, 'China');  
4

city	name_of_diet_type	cost
Nanxi	Vegan	170
Sandouping	Vegan	165
Zhijiang	Vegan	812

Рисунок 7.6 – Приклад виклику та роботи процедури для обрання турів з веганською дієтою у Китаї

4. Порахувати дохід за певним видом розрахунку (рис 5.59). Приклад виклику та роботи процедури представлено на рисунку 5.60

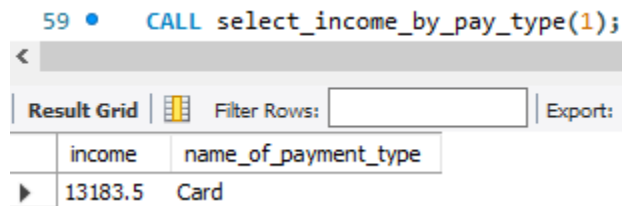
```

DELIMITER $$
CREATE PROCEDURE select_income_by_pay_type(pay_id INT)
BEGIN
    select ROUND((sum(cost_of_tour)), 2) as 'income', payment_type.name_of_payment_type from contract
    inner join payment on payment.payment_id = contract.contract_id AND (payment.payment_status_id = 4)
    inner join payment_status on payment_status.payment_status_id = payment.payment_status_id
    inner join payment_type on payment_type.payment_type_id = payment.payment_type_id AND payment_type.payment_type_id = pay_id;
END $$
DELIMITER ;

```

Рисунок 7.7 – Порахувати дохід за певним видом розрахунку

59 • `CALL select_income_by_pay_type(1);`



income	name_of_payment_type
13183.5	Card

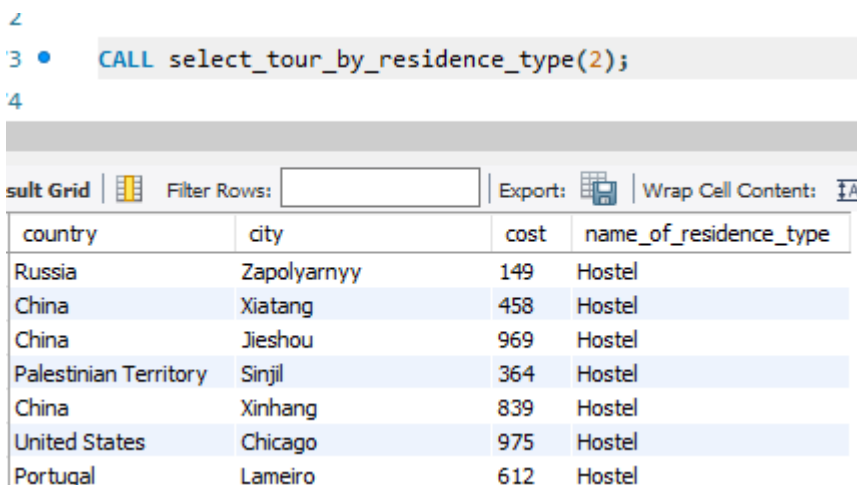
Рисунок 7.8 – Приклад виклику та роботи процедури: порахувати дохід за видом розрахунку Картка

5. Обрати всі місця відпочинку з певним типом (рис 5.61). Приклад виклику та роботи процедури представлено на рисунку 5.62

```
DELIMITER $$
CREATE PROCEDURE select_tour_by_residence_type(type_id INT)
BEGIN
    select country, city, cost, residence_type.name_of_residence_type from place_of_rest
    inner join residence_type on residence_type.residence_type_id = place_of_rest.residence_type_id
    AND place_of_rest.residence_type_id = type_id;
END $$
DELIMITER ;
```

Рисунок 7.9 – Обрати всі місця відпочинку з певним типом

3 • `CALL select_tour_by_residence_type(2);`



country	city	cost	name_of_residence_type
Russia	Zapolyarnyy	149	Hostel
China	Xiatang	458	Hostel
China	Jieshou	969	Hostel
Palestinian Territory	Sinjil	364	Hostel
China	Xinhang	839	Hostel
United States	Chicago	975	Hostel
Portugal	Lameiro	612	Hostel

Рисунок 7.10 – Обрати всі місця відпочинку з типом Хостел

6. Обрати всі тури з певним типом туру(рис 5.63) Приклад виклику та роботи процедури представлено на рисунку 5.64

```
DELIMITER $$
CREATE PROCEDURE select_tour_by_tour_type(type_id INT)
BEGIN
    select tour.tour_name, tour.duration, tour_type.name_of_tour_type from tour
    inner join tour_type on tour_type.type_tour_id = tour.type_tour_id AND tour.type_tour_id = type_id;
END $$
DELIMITER ;
```

Рисунок 7.11 – Обрати всі тури з певним типом туру

6 • **CALL** select\_tour\_by\_tour\_type(7);

tour_name	duration	name_of_tour_type
<i>Liatis spicata</i> (L.) Willd. var. <i>resinosa</i> (Nutt.) G...	59	Excursion
<i>Phoradendron tomentosum</i> (DC.) Engelm. ex A....	89	Excursion
<i>Mimulus traskiae</i> A.L. Grant	23	Excursion
<i>Eschscholzia parishii</i> Greene	73	Excursion
<i>Chamaecrista absus</i> (L.) Irwin & Barneby var. m...	69	Excursion
<i>Hyptis escobilla</i> Urb.	84	Excursion
<i>Fuchsia magellanica</i> Lam.	7	Excursion
<i>Draba pterosperma</i> Payson	34	Excursion
<i>Lathyrus pusillus</i> Elliott	78	Excursion
<i>Physalis hederifolia</i> A. Gray var. <i>fendleri</i> (A. Gr...	13	Excursion

Рисунок 7.12 – Приклад виклику та роботи процедури обрання всіх турів з типом туру Експедиція

7. Обрати всіх клієнтів, що обрали відпочинок з певним типом категорії та сплатили за нього(рис 5.65). Приклад виклику та роботи процедури представлено на рисунку 5.66

```
DELIMITER $$
CREATE PROCEDURE select_clients_that_paid_by_category(paid_category_id INT)
BEGIN
select client.client_id, client.first_name, client.last_name, client.phone_number, contract.contract_id,
payment_status.name_of_payment_status, contract.cost_of_tour, tour_type.name_of_tour_type from contract
inner join client on client.client_id = contract.client_id
inner join payment on payment.payment_id = contract.contract_id AND payment.payment_status_id = 4
inner join payment_status on payment_status.payment_status_id = payment.payment_status_id
inner join tours_in_contract on tours_in_contract.contract_id = contract.contract_id
inner join tour on tour.tour_id = tours_in_contract.tour_id
inner join tour_type on tour_type.type_tour_id = tour.type_tour_id AND tour.type_tour_id = paid_category_id;
END $$
DELIMITER ;
```

Рисунок 7.13 – Обрати всіх клієнтів, що обрали відпочинок з певним типом категорії та сплатили за нього

15 • **CALL** select\_clients\_that\_paid\_by\_category(7);

16

client_id	first_name	last_name	phone_number	contract_id	name_of_payment_status	cost_of_tour	name_of_tour_type
37	Kamillah	Holbarrow	(776) 6213738	27	Accepted	1252.9	Excursion
75	Gary	Drescher	(394) 6492735	93	Accepted	1328.8	Excursion
70	Thaddus	Benge	(960) 2148390	4	Accepted	1266.1	Excursion
2	Zilvia	Thon	(295) 7853169	5	Accepted	1532.3	Excursion

Рисунок 7.14 – Приклад виклику та роботи процедури обрання всіх турів з типом туру та сплатили за нього



8. Обрати всі місця відпочинку з певним типом категорії, що не були сплачені (рис 5.67). Приклад виклику та роботи процедури представлено на рисунку 5.68

```
DELIMITER $$
CREATE PROCEDURE select_clients_that_not_paid_by_category(not_paid_category_id INT)
BEGIN
    select client.client_id, client.first_name, client.last_name, client.phone_number,
    contract.contract_id, payment_status.name_of_payment_status, contract.cost_of_tour, category.name_of_category from contract
    inner join client on client.client_id = contract.client_id
    inner join payment on payment.payment_id = contract.contract_id AND
    (payment.payment_status_id = 3 OR payment.payment_status_id = 1)
    inner join payment_status on payment_status.payment_status_id = payment.payment_status_id
    inner join tours_in_contract on tours_in_contract.contract_id = contract.contract_id
    inner join tour on tour.tour_id = tours_in_contract.tour_id
    inner join place_of_rest on place_of_rest.rest_place_id = tour.place_of_rest_id
    inner join category on category.category_id = place_of_rest.place_of_rest_category AND category_id = not_paid_category_id;
END $$
DELIMITER ;
```

Рисунок 7.15 – Обрати всі місця відпочинку з певним типом категорії, що не були сплачені

```
5 • CALL select_clients_that_not_paid_by_category(6);
```

```
7
```

client_id	first_name	last_name	phone_number	contract_id	name_of_payment_status	cost_of_tour	name_of_category
2	Zilvia	Thon	(295) 7853169	38	Rejected	1496	Business
42	Dolph	Arp	(637) 3107159	25	Need to pay	1936	Business
100	Hersch	Sighart	(483) 8412311	43	Need to pay	950.4	Business
48	Tasha	Neicho	(808) 7909521	18	Rejected	1127.5	Business
93	Teodoro	Tames	(897) 5916275	84	Rejected	1532.3	Business
62	Obediah	Bramstom	(817) 6222912	61	Rejected	497.2	Business
81	Cinda	Skews	(943) 7203783	64	Need to pay	1674.2	Business

Рисунок 7.16 – Приклад виклику та роботи процедури обрання всіх місць відпочинку з типом категорії Бізнес, що не були сплачені

9. Обрати всіх клієнтів що бронювали номери певної категорії(рис 5.69). Приклад виклику та роботи процедури представлено на рисунку 5.70

```

DELIMITER $$
CREATE PROCEDURE select_clients_by_booking_category(searched_category_id INT)
BEGIN
    select client.client_id, client.first_name, client.last_name, client.phone_number,
    contract.contract_id, contract.cost_of_tour, category.name_of_category from contract
    inner join client on client.client_id = contract.client_id
    inner join tours_in_contract on tours_in_contract.contract_id = contract.contract_id
    inner join tour on tour.tour_id = tours_in_contract.tour_id
    inner join place_of_rest on place_of_rest.rest_place_id = tour.place_of_rest_id
    inner join category on category.category_id = place_of_rest.place_of_rest_category
    AND category_id = searched_category_id;
END $$
DELIMITER ;

```

Рисунок 7.17 – Обрати всіх клієнтів що бронювали номери певної категорії

3

4 • **CALL** select\_clients\_by\_booking\_category(5);

5

client_id	first_name	last_name	phone_number	contract_id	cost_of_tour	name_of_category
94	Delly	Cosin	(750) 1784927	52	1367.3	Lux
11	Charissa	Gingold	(169) 3500897	58	668.8	Lux
12	Cassandra	Overton	(114) 4433640	80	668.8	Lux
62	Obediah	Bramstom	(817) 6222912	97	679.8	Lux
20	Abby	Thackham	(155) 8344896	12	323.4	Lux
55	Ulrich	Glover	(569) 2984096	92	592.9	Lux
58	Gilberte	Wrightem	(295) 4653937	85	592.9	Lux
58	Gilberte	Wrightem	(295) 4653937	76	1611.5	Lux
54	Janna	Masedon	(917) 5950976	78	944.9	Lux
9	Fleur	Duplock	(509) 3951596	30	944.9	Lux
40	Cyrille	McNuff	(692) 6702338	79	894.3	Lux
87	Carlina	Spracklin	(571) 8247180	35	578.6	Lux
37	Kamillah	Holbarrow	(776) 6213738	27	1252.9	Lux

Рисунок 7.18 – Приклад виклику та роботи процедури обрання всіх клієнтів що бронювали номери категорії Люкс

## 7.2 Оптимізація запитів

Для оптимізації запитів буде використано можливість створення індексів у MySQL. Індеси використовуються для швидкого вилучення даних із бази даних. Користувачі не бачать індексів, вони просто використовуються для прискорення пошуків/запитів. Індеси дозволяють оптимізувати роботу з базою – дані сортуються і СУБД не доводиться шукати значення серед усіх записів. На

практиці це може скоротити час обробки запитів на кілька порядків. Проаналізувавши схеми запитів, я знайшов ті таблиці та їх поля, де присутній Full index scan.(рис. 5.71)

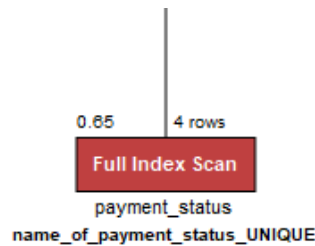


Рисунок 7.19 – Частина схеми запиту з Full index scan

При наведенні на червоний прямокутник, з'являється додаткова інформація про запит, де також вказуються колонки, що використовуються в цьому запиті(рис. 5.72, червоним підкреслено назви колонок)

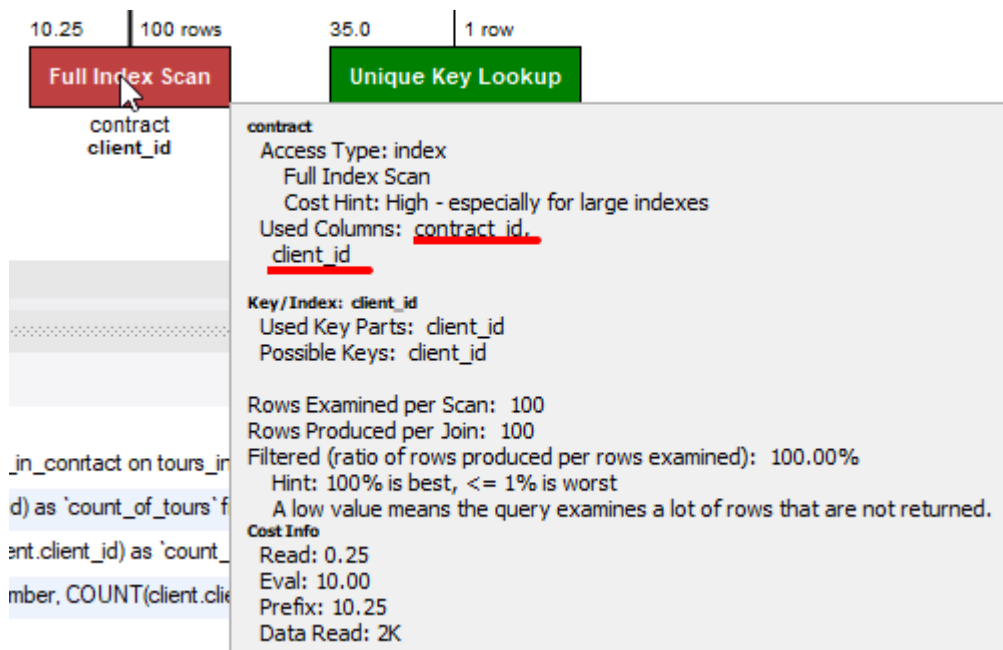


Рисунок 7.20 – Додаткова інформація про запит

Для прикладу роботи оптимізації запитів, я покажу схеми деяких запитів до та після оптимізації.

Таким чином для створення індексів було обрано наступні таблиці та їх колонки:

1. Таблиця payment\_status, колонки payment\_status\_id, name\_of\_payment\_status
2. Таблиця tour\_type, колонки type\_tour\_id, name\_of\_tour\_type
3. Таблиця transfer\_type, колонки transfer\_type\_id, name\_transfer\_type
4. Таблиця tour, колонки tour\_id, place\_of\_rest\_id
5. Таблиця contract, колонки contract\_id, client\_id, visa\_required, cost\_of\_tour

На рисунку 5.73 наведено скрипт для створення індексів у наведених вище таблицях та їх колонках.

```
create index payment_status_idx on payment_status (payment_status_id, name_of_payment_status);
create index tour_type_idx on tour_type (type_tour_id, name_of_tour_type);
create index transfer_type_idx on transfer_type (transfer_type_id, name_transfer_type);
create index tour_idx on tour (tour_id, place_of_rest_id);
create index contract_idx on contract (contract_id, client_id, visa_required, cost_of_tour);
```

Рисунок 7.21 – Скрипт для створення індексів

Приклад роботи запиту обрання тих клієнтів, що не здійснили оплату туру.

Рисунок 5.74 – схема запиту до створення індексів, рисунок 5.75 – схема запиту після створення індексів.

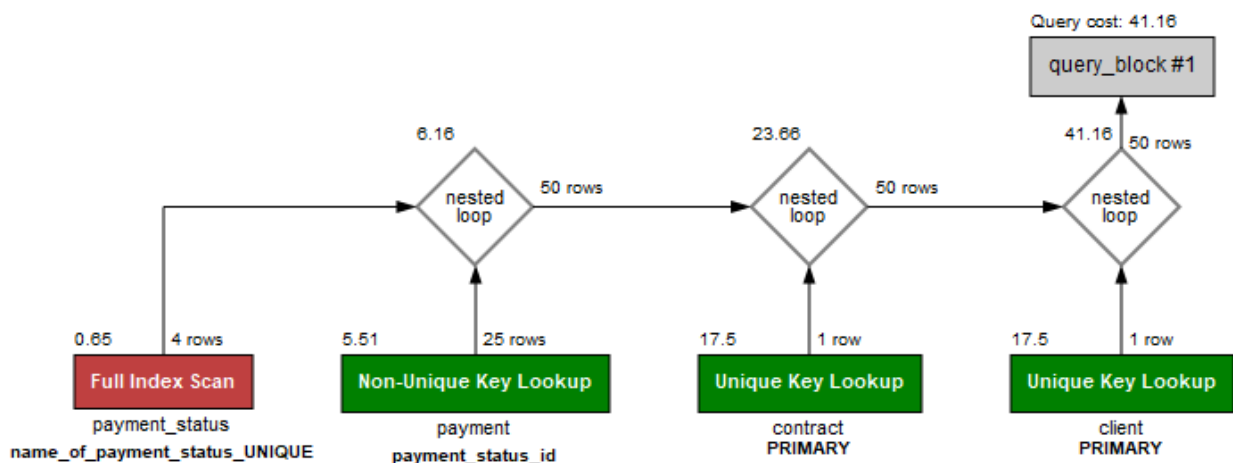


Рисунок 7.22 – Схема запиту до створення індексів

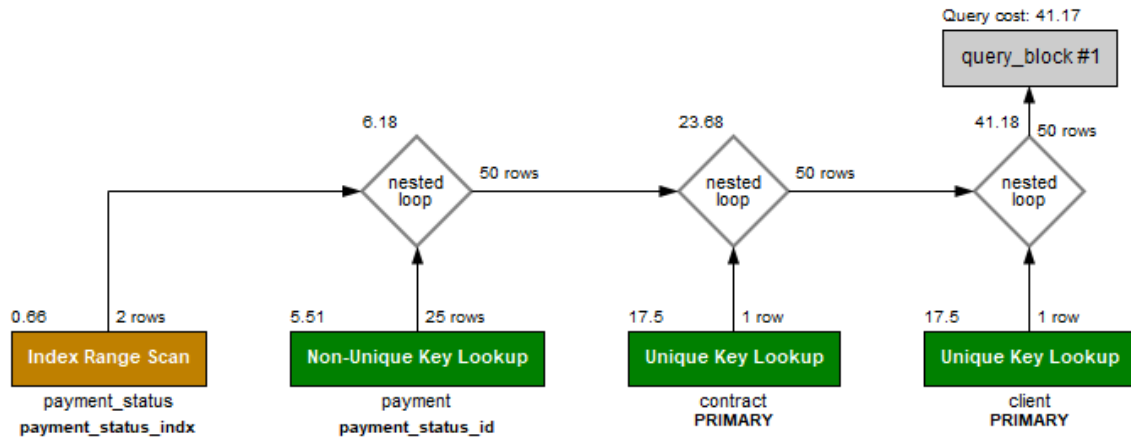


Рисунок 7.23 – Схема запиту після створення індексів

В ході цього розділу було таблиці. Для них було забезпечено збереження цілісності даних то накладено певні обмеження. Було розроблено запити для реалізації зазначених вимог. Деякі з запитів було переписано в процедури для більш зручного використання. Наприкінці було реалізовано оптимізацію запитів шляхом додавання індексів.

## Висновки

Таким чином, в результаті написання курсової роботи було створено базу даних для підтримки діяльності туристичної компанії. Головною задачею було реалізувати базу даних, що буде відповідати певним вимогам.

Мною було проведено дослідження предметної області, де я досконало розібрався з тим, які моделі БД можуть використовувати туристичні компанії, та створив свою модель.

Перед реалізацією бази даних, я визначив основні сутності, описав їх. Після цього мною було розроблено ER-модель бази даних, яку я в подальшому використовував як основу для створення схеми БД.

Для написання бази даних, було обрано програмне забезпечення MySQL workbench, як сучасний інструмент для розробки баз даних. Це програмне забезпечення має зручний інтерфейс та корисні вбудовані засоби для роботи з базою даних.

Мною було створено таблиці, додано до них обмеження та правила для збереження цілісності даних. Далі було додано ролі та користувачів БД, реалізовано основні скрипти для підтримки діяльності туристичної компанії. Кожен з скриптів було переірено на коректність роботи. Далі для більш зручного використання скриптів, деякі з них було перетворено на процедури. Для прискорення роботи та оптимізації запитів, було додано індекси для деяких з таблиць та полів.

### Перелік посилань

1. База даних – [Електронний ресурс]. – Режим доступу: [https://uk.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0\\_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85](https://uk.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85)
2. Організація баз даних: практичний курс: Навч. посіб. для студ. / А. Ю. Берко, О. М. Верес; Нац. ун-т «Львів. політехніка». — Л., 2003. — 149 с.
3. Аткинсон, Леон. MySQL. Библиотека профессионала.: Пер. с англ. — М.: Изд. дом "Вильямс", 2002. — 624 с
4. Мартин Грабер. SQL. — К.: Изд-во “Лори”, 2003. — 644 с
5. Теория и практика по строению баз данных. 8 -е изд. / Д.Кренке. — СПб.: Питер, 2003. — 800 с.
6. MySQL Workbench – [Електронний ресурс]. – Режим доступу: [https://en.wikipedia.org/wiki/MySQL\\_Workbench](https://en.wikipedia.org/wiki/MySQL_Workbench)

## Додаток А Тексти програмного коду

*Тексти програмного коду програмного забезпечення*

*Підтримка діяльності туристичної компанії*

---

(Найменування програми (документа))

*HDD*

---

(Вид носія даних)

*34 арк, 66,6 Кб*

---

(Обсяг програми (документа), арк.,

*студента групи ІП-02 II курсу*

*Трофимова Д.О.*



1. Створення таблиць(Файл«create tables.sql»)

```
CREATE SCHEMA touristagency;
```

```
CREATE TABLE `diet_type` (  
  `diet_type_id` int NOT NULL AUTO_INCREMENT,  
  `name_of_diet_type` varchar(45) NOT NULL,  
  PRIMARY KEY (`diet_type_id`)  
);
```

```
CREATE TABLE `category` (  
  `category_id` int NOT NULL,  
  `name_of_category` varchar(45) NOT NULL,  
  PRIMARY KEY (`category_id`)  
);
```

```
CREATE TABLE `payment_status` (  
  `payment_status_id` int NOT NULL AUTO_INCREMENT,  
  `name_of_payment_status` varchar(30) NOT NULL,  
  PRIMARY KEY (`payment_status_id`)  
);
```

```
CREATE TABLE `payment_type` (  
  `payment_type_id` int NOT NULL AUTO_INCREMENT,  
  `name_of_payment_type` varchar(30) NOT NULL,  
  PRIMARY KEY (`payment_type_id`)  
);
```

```
CREATE TABLE `residence_type` (  
  `residence_type_id` int NOT NULL,  
  `name_of_residence_type` varchar(45) NOT NULL,
```

```
PRIMARY KEY (`residence_type_id`)  
);
```

```
CREATE TABLE `transfer_type` (  
  `transfer_type_id` int NOT NULL AUTO_INCREMENT,  
  `name_transfer_type` varchar(45) NOT NULL,  
  PRIMARY KEY (`transfer_type_id`)  
);
```

```
CREATE TABLE `tour_type` (  
  `type_tour_id` int NOT NULL AUTO_INCREMENT,  
  `name_of_tour_type` varchar(30) NOT NULL,  
  PRIMARY KEY (`type_tour_id`)  
);
```

```
CREATE TABLE `client` (  
  `client_id` int NOT NULL AUTO_INCREMENT,  
  `first_name` varchar(30) NOT NULL,  
  `patronymic` varchar(30) DEFAULT NULL,  
  `last_name` varchar(30) NOT NULL,  
  `address` varchar(80) NOT NULL,  
  `date_of_birth` date NOT NULL,  
  `passport` varchar(10) NOT NULL,  
  `phone_number` varchar(20) NOT NULL,  
  PRIMARY KEY (`client_id`)  
);
```

```
CREATE TABLE `payment` (  
  `payment_id` int NOT NULL AUTO_INCREMENT,  
  `payment_type_id` int NOT NULL,
```

```

`payment_status_id` int NOT NULL,
PRIMARY KEY (`payment_id`)
);

```

```

CREATE TABLE `transfer` (
  `transfer_id` int NOT NULL AUTO_INCREMENT,
  `transfer_type_id` int NOT NULL,
  `departure_country` varchar(45) NOT NULL,
  `country_of_arrival` varchar(45) NOT NULL,
  `departure_time` datetime NOT NULL,
  `arrival_time` datetime NOT NULL,
  `cost` double NOT NULL,
  PRIMARY KEY (`transfer_id`)
);

```

```

CREATE TABLE `contract` (
  `contract_id` int NOT NULL AUTO_INCREMENT,
  `client_id` int NOT NULL,
  `cost_of_tour` double NOT NULL,
  `payment_id` int NOT NULL,
  `visa_required` tinyint NOT NULL,
  PRIMARY KEY (`contract_id`)
);

```

```

CREATE TABLE `place_of_rest` (
  `rest_place_id` int NOT NULL AUTO_INCREMENT,
  `residence_type_id` int NOT NULL,
  `place_of_rest_category` int NOT NULL,
  `country` varchar(45) NOT NULL,

```

```

`city` varchar(45) NOT NULL,
`cost` double NOT NULL,
`duration` varchar(45) NOT NULL,
`type_of_diet_id` int NOT NULL,
`transfer_to_residence_id` int NOT NULL,
PRIMARY KEY (`rest_place_id`)
);

```

```

CREATE TABLE `tour` (
  `tour_id` int NOT NULL AUTO_INCREMENT,
  `type_tour_id` int NOT NULL,
  `place_of_rest_id` int NOT NULL,
  `transfer_id` int NOT NULL,
  `tour_name` varchar(100) NOT NULL,
  `duration` varchar(45) NOT NULL,
  PRIMARY KEY (`tour_id`)
);

```

```

CREATE TABLE `tours_in_contract` (
  `tour_in_contract_id` int NOT NULL AUTO_INCREMENT,
  `tour_id` int NOT NULL,
  `contract_id` int NOT NULL,
  PRIMARY KEY (`tour_in_contract_id`)
);

```

## 2. Ограничения(Файл «constraints.sql»)

```

alter table diet_type add UNIQUE KEY `diet_type_id_UNIQUE`
(`diet_type_id`);
alter table diet_type add UNIQUE KEY `name_of_diet_type_UNIQUE`
(`name_of_diet_type`);

```

```
alter table category add UNIQUE KEY `name_of_category_UNIQUE`
(`name_of_category`);
```

```
alter table payment_status add UNIQUE KEY `payment_status_id_UNIQUE`
(`payment_status_id`);
```

```
alter table payment_status add UNIQUE KEY
`name_of_payment_status_UNIQUE` (`name_of_payment_status`);
```

```
alter table payment_type add UNIQUE KEY `payment_type_id_UNIQUE`
(`payment_type_id`);
```

```
alter table payment_type add UNIQUE KEY
`name_of_payment_type_UNIQUE` (`name_of_payment_type`);
```

```
alter table residence_type add UNIQUE KEY
`name_of_residence_type_UNIQUE` (`name_of_residence_type`);
```

```
alter table residence_type add UNIQUE KEY `residence_type_id_UNIQUE`
(`residence_type_id`);
```

```
alter table transfer_type add UNIQUE KEY `transfer_type_id_UNIQUE`
(`transfer_type_id`);
```

```
alter table transfer_type add UNIQUE KEY `name_transfer_type_UNIQUE`
(`name_transfer_type`);
```

```
alter table tour_type add UNIQUE KEY `type_tour_id_UNIQUE`
(`type_tour_id`);
```

```
alter table tour_type add UNIQUE KEY `name_of_tour_type_UNIQUE`
(`name_of_tour_type`);
```

```
alter table client add UNIQUE KEY `client_id_UNIQUE` (`client_id`);
```

```
alter table client add UNIQUE KEY `phone_number_UNIQUE`
(`phone_number`);
```

```
alter table client add UNIQUE KEY `passport_UNIQUE` (`passport`);
```

```
alter table client add CONSTRAINT `check_date_of_birth` CHECK ((YEAR
(`date_of_birth`) > cast((((1901 - 1) - 1) as date))));
```

```
alter table payment add UNIQUE KEY `payment_id_UNIQUE`
(`payment_id`);
```

```
alter table payment add CONSTRAINT `payment_ibfk_1` FOREIGN KEY
(`payment_type_id`) REFERENCES `payment_type` (`payment_type_id`) ON
DELETE RESTRICT ON UPDATE CASCADE;
```

```
alter table payment add CONSTRAINT `payment_ibfk_2` FOREIGN KEY
(`payment_status_id`) REFERENCES `payment_status` (`payment_status_id`) ON
DELETE RESTRICT ON UPDATE CASCADE;
```

```
alter table transfer add UNIQUE KEY `transfer_id_UNIQUE` (`transfer_id`);
```

```
alter table transfer add CONSTRAINT `transfer_ibfk_1` FOREIGN KEY
(`transfer_type_id`) REFERENCES `transfer_type` (`transfer_type_id`) ON DELETE
RESTRICT ON UPDATE CASCADE;
```

```
alter table transfer add CONSTRAINT `check_cost1` CHECK ((`cost` > 0));
```

```
alter table contract add UNIQUE KEY `contract_id_UNIQUE` (`contract_id`);
```

```
alter table contract add CONSTRAINT `contract_ibfk_1` FOREIGN KEY
(`payment_id`) REFERENCES `payment` (`payment_id`) ON DELETE RESTRICT
ON UPDATE CASCADE;
```

```
alter table contract add CONSTRAINT `contract_ibfk_2` FOREIGN KEY
(`client_id`) REFERENCES `client` (`client_id`) ON DELETE RESTRICT ON
UPDATE CASCADE;
```

```
alter table contract add CONSTRAINT `check_cost2` CHECK ((`cost_of_tour`
> 0));
```

```
alter table place_of_rest add UNIQUE KEY `(PK)rest_place_id_UNIQUE`
(`rest_place_id`);
```

```
alter table place_of_rest add CONSTRAINT `place_of_rest_ibfk_1` FOREIGN
KEY (`residence_type_id`) REFERENCES `residence_type` (`residence_type_id`)
ON DELETE RESTRICT ON UPDATE CASCADE;
```

```
alter table place_of_rest add CONSTRAINT `place_of_rest_ibfk_2` FOREIGN
KEY (`place_of_rest_category`) REFERENCES `category` (`category_id`) ON
DELETE RESTRICT ON UPDATE CASCADE;
```

```
alter table place_of_rest add CONSTRAINT `place_of_rest_ibfk_3` FOREIGN
KEY (`transfer_to_residence_id`) REFERENCES `transfer_type` (`transfer_type_id`)
ON DELETE RESTRICT ON UPDATE CASCADE;
```

```
alter table place_of_rest add CONSTRAINT `place_of_rest_ibfk_4` FOREIGN
KEY (`type_of_diet_id`) REFERENCES `diet_type` (`diet_type_id`) ON DELETE
RESTRICT ON UPDATE CASCADE;
```

```
alter table place_of_rest add CONSTRAINT `check_cost` CHECK ((`cost` >
0));
```

```
alter table place_of_rest add CONSTRAINT `check_rest_duration` CHECK
((`duration` > 0));
```

```
alter table tour add UNIQUE KEY `tour_id_UNIQUE` (`tour_id`);
```

```
alter table tour add UNIQUE KEY `tour_name_UNIQUE` (`tour_name`);
```

```
alter table tour add CONSTRAINT `tour_ibfk_1` FOREIGN KEY
(`type_tour_id`) REFERENCES `tour_type` (`type_tour_id`) ON DELETE
RESTRICT ON UPDATE CASCADE;
```

```
alter table tour add CONSTRAINT `tour_ibfk_2` FOREIGN KEY
(`transfer_id`) REFERENCES `transfer` (`transfer_id`) ON DELETE RESTRICT ON
UPDATE CASCADE;
```

```
alter table tour add CONSTRAINT `tour_ibfk_3` FOREIGN KEY
(`place_of_rest_id`) REFERENCES `place_of_rest` (`rest_place_id`) ON DELETE
RESTRICT ON UPDATE CASCADE;
```

```
alter table tour add CONSTRAINT `check_tour_duration` CHECK ((`duration`
> 0));
```

```
alter table tours_in_conrtact add UNIQUE KEY
`tour_in_conrtact_id_UNIQUE` (`tour_in_conrtact_id`);
```

```
alter table tours_in_conrtact add CONSTRAINT `tours_in_conrtact_ibfk_1`
FOREIGN KEY (`tour_id`) REFERENCES `tour` (`tour_id`) ON DELETE
RESTRICT ON UPDATE CASCADE;
```

```
alter table tours_in_conrtact add CONSTRAINT `tours_in_conrtact_ibfk_2`
FOREIGN KEY (`contract_id`) REFERENCES `contract` (`contract_id`) ON
DELETE RESTRICT ON UPDATE CASCADE;
```

### 3. Створення ролей(Файл «create roles.sql»)

```
create role 'admin'@'%';
```

```
grant select, insert, update, delete on touristagency.* to 'admin'@'%';
```

```
create role 'worker'@'%';
```

```
grant select on touristagency.* to 'worker'@'%';
```

```
grant select, insert, update on touristagency.client to 'worker'@'%';
```

```
grant select, insert, update on touristagency.contract to 'worker'@'%';
```

```
grant select, insert, update on touristagency.tour to 'worker'@'%';
```

```
grant select, insert, update on touristagency.transfer to 'worker'@'%';
```

```
grant select, insert, update on touristagency.payment to 'worker'@'%';
```

```
grant select, insert, update on touristagency.tours_in_conrtact to 'worker'@'%';
```

```
create role 'client'@'%';
```

```
grant select on touristagency.tour to 'client'@'%';
```



```
grant select on touristagency.category to 'client'@'%';
grant select on touristagency.diet_type to 'client'@'%';
grant select on touristagency.place_of_rest to 'client'@'%';
grant select on touristagency.residence_type to 'client'@'%';
grant select on touristagency.tour_type to 'client'@'%';
grant select on touristagency.transfer to 'client'@'%';
grant select on touristagency.transfer_type to 'client'@'%';
```

```
create user 'test_admin'@'%' identified by 'tuSM644F9y';
grant 'admin' to 'test_admin'@'%';
```

```
create user 'test_worker'@'%' identified by 'bu23z9XVS7';
grant 'worker' to 'test_worker'@'%';
```

```
create user 'test_client'@'%' identified by 'i92j9EYMe9';
grant 'client' to 'test_client'@'%';
```

4. Тригер для перевірки дат прибуття та відбуття на коректність (Файл «departure arrival dates trigger.sql»)

```
CREATE DEFINER=`Danylo`@`%` TRIGGER `transfer_BEFORE_INSERT`
BEFORE INSERT ON `transfer` FOR EACH ROW BEGIN
IF (NEW.departure_time <= CURDATE() OR NEW.arrival_time <=
CURDATE())
THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Invalid date';
END IF;
END
```

5. Тригер для перевірки віку улієнта на коректність (Файл «age trigger.sql»)

```

CREATE DEFINER=`Danylo`@`%` TRIGGER `client_BEFORE_INSERT`
BEFORE INSERT ON `client` FOR EACH ROW BEGIN
DECLARE currAge INT;
DECLARE currDate DATE;
SET currDate=NEW.date_of_birth;
SET currAge=DATEDIFF(curdate(), currDate) / 365.25;
IF currAge<18 THEN signal sqlstate '45000'
SET MESSAGE_TEXT = 'Client must be 18 years or older';
END IF;
END

```

#### 6. Скрипти типу Select(Файл «selects.sql»)

use touristagency;

-- 0. Оновити дані по вартості у контракті

```
create view view1 AS
```

```
select contract.contract_id, place_of_rest.cost as `rest_cost` from place_of_rest
inner join contract
```

```
inner join tours_in_conrtact on tours_in_conrtact.contract_id =
contract.contract_id
```

```
inner join tour on tours_in_conrtact.tour_id = tour.tour_id AND
place_of_rest_id = place_of_rest.rest_place_id;
```

```
select * from view1;
```

```
create view view2 AS
```

```
select contract.contract_id, transfer.cost as `transfer_cost` from transfer
inner join contract
```

```
inner join tours_in_conrtact on tours_in_conrtact.contract_id =
contract.contract_id
```

```
inner join tour on tours_in_conrtact.tour_id = tour.tour_id AND tour.transfer_id
= transfer.transfer_id;
```

```
select * from view2;
```

```
update contract
```

```
inner join view1 on view1.contract_id = contract.contract_id
```

```
inner join view2 on view2.contract_id = contract.contract_id
```

```
set      contract.cost_of_tour      =      ROUND(((view1.rest_cost      +
view2.transfer_cost)*1.1), 2) ;
```

```
drop view view1;
```

```
drop view view2;
```

```
-- 1. Обрати всі доступні тури, їх тривалість, тип та країну.
```

```
select tour.tour_id, tour_name, tour.duration, tour_type.name_of_tour_type,
place_of_rest.country from tour
```

```
inner join tour_type on tour_type.type_tour_id = tour.type_tour_id
```

```
inner join place_of_rest on place_of_rest.rest_place_id = tour.place_of_rest_id;
```

```
-- 2. Обрати тих клієнтів, що не здійснили оплату туру
```

```
select client.client_id, client.first_name, client.last_name, client.phone_number,
contract.contract_id,
```

```
payment_status.name_of_payment_status, contract.cost_of_tour from contract
```

```
inner join client on client.client_id = contract.client_id
```

```
inner join payment on payment.payment_id = contract.contract_id AND
```

```
(payment.payment_status_id = 3 OR payment.payment_status_id = 1)
```

```
inner join payment_status on payment_status.payment_status_id =
payment.payment_status_id;
```

```
-- 3. Обрати контракти та клієнтів, що замовляли цей контракт
```

```
select client.client_id, client.first_name, client.last_name, client.phone_number,
contract.contract_id,
```

```

contract.cost_of_tour, payment_status.name_of_payment_status from contract
inner join client on client.client_id = contract.client_id
inner join payment on payment.payment_id = contract.contract_id
inner join payment_status on payment_status.payment_status_id =
payment.payment_status_id;

```

-- 4. Обрати всі тури клієнтів

```

select tour.tour_id, tour.tour_name, payment_status.name_of_payment_status,
client.client_id, client.first_name, client.last_name, client.phone_number from
contract
inner join client on client.client_id = contract.client_id
inner join payment on payment.payment_id = contract.contract_id
inner join payment_status on payment_status.payment_status_id =
payment.payment_status_id
inner join tours_in_contract on tours_in_contract.contract_id =
contract.contract_id
inner join tour on tour.tour_id = tours_in_contract.tour_id;

```

-- 5. Обрати дати та країни вильоту та прильоту турів

```

select      tour.tour_id,      tour.tour_name,      transfer.arrival_time,
transfer.departure_time,
transfer.country_of_arrival,      transfer.departure_country,
transfer_type.name_transfer_type from tour
inner join transfer on transfer.transfer_id = tour.transfer_id
inner join transfer_type on transfer_type.transfer_type_id =
transfer.transfer_type_id;

```

-- 6. Обрахувати прибуток компанії з проданих турів

-- округлення до 2 знаків, множимо на 0.0909 так як це покаже приблизний прибуток з початковою націнкою 10%

```
select ROUND((sum(cost_of_tour)*0.0909), 2) as 'income' from contract
inner join payment on payment.payment_id = contract.contract_id AND
(payment.payment_status_id = 4)
inner join payment_status on payment_status.payment_status_id =
payment.payment_status_id;
```

-- 7. Обрати всі апартаменти з типом Люкс

```
select country, city, cost, category.name_of_category from place_of_rest
inner join category on place_of_rest.place_of_rest_category =
category.category_id AND category.category_id = 5;
```

-- 8. Обрати суму грошей, що клієнти тільки мають сплатити

```
select ROUND(sum(contract.cost_of_tour), 2) as `need to pay sum` from
contract
inner join payment on payment.payment_id = contract.contract_id AND
payment.payment_status_id = 1
inner join payment_status on payment_status.payment_status_id =
payment.payment_status_id;
```

-- 9. Вивести топ три найпопулярніших країни для турів

```
SELECT place_of_rest.country, COUNT(*) AS total FROM tour
inner join place_of_rest on place_of_rest.rest_place_id = tour.place_of_rest_id
GROUP BY country ORDER BY total DESC LIMIT 3;
```

-- 10. Обрати всі міста для відпочинку що відповідають халяльній дієті в Україні

```
select place_of_rest.city, diet_type.name_of_diet_type, cost from place_of_rest
inner join diet_type on diet_type.diet_type_id = place_of_rest.type_of_diet_id
AND place_of_rest.type_of_diet_id = 7 AND place_of_rest.country = 'Ukraine';
```

-- 11. Порахувати дохід за безготівковим розрахунком

```
select ROUND((sum(cost_of_tour)), 2) as 'income' from contract
inner join payment on payment.payment_id = contract.contract_id AND
(payment.payment_status_id = 4)
inner join payment_status on payment_status.payment_status_id =
payment.payment_status_id
inner join payment_type on payment_type.payment_type_id =
payment.payment_type_id AND payment_type.payment_type_id = 1;
```

-- 12. Обрати всіх клієнтів і їх контракти, що потребували допомоги в оформленні візи

```
select client.first_name, client.last_name, client.phone_number, client.passport,
contract.contract_id from contract
inner join client on client.client_id = contract.client_id AND
contract.visa_required = true;
```

-- 13. Обрати всі тури, де трансфер буде не літаком чи гелікоптером(клієнт - аерофоб)

```
select tour.tour_name, tour.duration, transfer_type.name_transfer_type from
tour
inner join transfer on transfer.transfer_id = tour.transfer_id
inner join transfer_type on transfer_type.transfer_type_id =
transfer.transfer_type_id
```

AND NOT transfer.transfer\_type\_id = 5 AND NOT transfer.transfer\_type\_id = 4;

-- 14. Обрати всі тури же тип місця відпочинку - кемпінг

```
select country, city, cost, residence_type.name_of_residence_type from
place_of_rest
inner join residence_type on residence_type.residence_type_id =
place_of_rest.residence_type_id
AND place_of_rest.residence_type_id = 6;
```

-- 15. Обрати всі тури що мають тип Зимній

```
select tour.tour_name, tour.duration, tour_type.name_of_tour_type from tour
inner join tour_type on tour_type.type_tour_id = tour.type_tour_id AND
tour.type_tour_id = 5;
```

-- 16. Обрати всіх клієнтів, що обрали зимній тур та сплатили за нього

```
select client.client_id, client.first_name, client.last_name, client.phone_number,
contract.contract_id,
payment_status.name_of_payment_status, contract.cost_of_tour from contract
inner join client on client.client_id = contract.client_id
inner join payment on payment.payment_id = contract.contract_id AND
payment.payment_status_id = 4
inner join payment_status on payment_status.payment_status_id =
payment.payment_status_id
inner join tours_in_contract on tours_in_contract.contract_id =
contract.contract_id
inner join tour on tour.tour_id = tours_in_contract.tour_id
inner join tour_type on tour_type.type_tour_id = tour.type_tour_id AND
tour.type_tour_id = 5;
```

-- 17. Обрати клієнтів, що бронювали люкс, але не сплатили

```
select client.client_id, client.first_name, client.last_name, client.phone_number,
contract.contract_id,
payment_status.name_of_payment_status, contract.cost_of_tour from contract
inner join client on client.client_id = contract.client_id
inner join payment on payment.payment_id = contract.contract_id AND
(payment.payment_status_id = 3 OR payment.payment_status_id = 1)
inner join payment_status on payment_status.payment_status_id =
payment.payment_status_id
inner join tours_in_contract on tours_in_contract.contract_id =
contract.contract_id
inner join tour on tour.tour_id = tours_in_contract.tour_id
inner join place_of_rest on place_of_rest.rest_place_id = tour.place_of_rest_id
inner join category on category.category_id =
place_of_rest.place_of_rest_category AND category_id = 5;
```

-- 18. Обрати всіх клієнтів що бронювали бізнес-номери

```
select client.client_id, client.first_name, client.last_name, client.phone_number,
contract.contract_id,
contract.cost_of_tour, category.name_of_category from contract
inner join client on client.client_id = contract.client_id
inner join tours_in_contract on tours_in_contract.contract_id =
contract.contract_id
inner join tour on tour.tour_id = tours_in_contract.tour_id
inner join place_of_rest on place_of_rest.rest_place_id = tour.place_of_rest_id
inner join category on category.category_id =
place_of_rest.place_of_rest_category AND category_id = 6;
```



-- 19. Обрати найдешевший тур

```
select tour.tour_name, contract.cost_of_tour from contract
inner join tours_in_contract on tours_in_contract.contract_id =
contract.contract_id
inner join tour on tour.tour_id = tours_in_contract.tour_id
GROUP BY cost_of_tour ORDER BY cost_of_tour ASC LIMIT 1;
```

-- 20. Обрати клієнта що зробив найбільше замовлень

```
select client.client_id, client.first_name, client.last_name,
client.phone_number, COUNT(client.client_id) as `count_of_tours` from client
inner join contract on client.client_id = contract.client_id
GROUP BY client.first_name ORDER BY count_of_tours DESC LIMIT 1;
```

7. Процедури(Файл «procedures.sql»)

```
DROP PROCEDURE IF EXISTS select_client_by_contract;
```

-- аналог 2 запиту, обрати клієнта що замовляв певний контракт

```
DELIMITER $$
```

```
CREATE PROCEDURE select_client_by_contract(searched_contract_id INT)
```

```
BEGIN
```

```
select client.client_id, client.first_name, client.last_name, client.phone_number,
contract.contract_id,
```

```
contract.cost_of_tour, payment_status.name_of_payment_status from
contract
```

```
inner join client on client.client_id = contract.client_id
```

```
inner join payment on payment.payment_id = contract.contract_id AND
contract_id = searched_contract_id
```

```
inner join payment_status on payment_status.payment_status_id =
payment.payment_status_id;
```

END \$\$

DELIMITER ;

CALL select\_client\_by\_contract(5);

DROP PROCEDURE IF EXISTS select\_top\_for\_traveling;

-- аналог 9 запиту, вивести топ-п найпопулярніших країн для мандрівки

DELIMITER \$\$

CREATE PROCEDURE select\_top\_for\_traveling(top INT)

BEGIN

SELECT place\_of\_rest.country, COUNT(\*) AS total FROM tour

inner join place\_of\_rest on place\_of\_rest.rest\_place\_id = tour.place\_of\_rest\_id

GROUP BY country ORDER BY total DESC LIMIT top;

END \$\$

DELIMITER ;

CALL select\_top\_for\_traveling(5);

DROP PROCEDURE IF EXISTS select\_tour\_by\_diet;

-- аналог 10 запиту, обрати всі місця відпочинку з певною дієтою та певною країною(значення в рамках [1,8])

DELIMITER \$\$

CREATE PROCEDURE select\_tour\_by\_diet(diet\_id INT, country VARCHAR(45))

BEGIN

select place\_of\_rest.city, diet\_type.name\_of\_diet\_type, cost from place\_of\_rest

inner join diet\_type on diet\_type.diet\_type\_id = place\_of\_rest.type\_of\_diet\_id

AND

```
place_of_rest.type_of_diet_id = diet_id AND place_of_rest.country =
country;
```

```
END $$
```

```
DELIMITER ;
```

```
CALL select_tour_by_diet(6, 'China');
```

```
DROP PROCEDURE IF EXISTS select_income_by_pay_type;
```

```
-- 11. Порахувати дохід за певним розрахунком(значення в рамках [1,3])
```

```
DELIMITER $$
```

```
CREATE PROCEDURE select_income_by_pay_type(pay_id INT)
```

```
BEGIN
```

```
select      ROUND((sum(cost_of_tour)),      2)      as      'income',
payment_type.name_of_payment_type from contract
```

```
inner join payment on payment.payment_id = contract.contract_id AND
(payment.payment_status_id = 4)
```

```
inner join payment_status on payment_status.payment_status_id =
payment.payment_status_id
```

```
inner join payment_type on payment_type.payment_type_id =
payment.payment_type_id AND payment_type.payment_type_id = pay_id;
```

```
END $$
```

```
DELIMITER ;
```

```
CALL select_income_by_pay_type(1);
```

```
DROP PROCEDURE IF EXISTS select_tour_by_residence_type;
```

-- аналог 14 запиту, обрати всі місця відпочинку з певним типом категорії  
(значення в рамках [1,6])

DELIMITER \$\$

CREATE PROCEDURE select\_tour\_by\_residence\_type(type\_id INT)

BEGIN

select country, city, cost, residence\_type.name\_of\_residence\_type from  
place\_of\_rest

inner join residence\_type on residence\_type.residence\_type\_id =  
place\_of\_rest.residence\_type\_id

AND place\_of\_rest.residence\_type\_id = type\_id;

END \$\$

DELIMITER ;

CALL select\_tour\_by\_residence\_type(2);

DROP PROCEDURE IF EXISTS select\_tour\_by\_tour\_type;

-- аналог 15 запиту, обрати всі тури з певним типом туру (значення в  
рамках [1,7])

DELIMITER \$\$

CREATE PROCEDURE select\_tour\_by\_tour\_type(type\_id INT)

BEGIN

select tour.tour\_name, tour.duration, tour\_type.name\_of\_tour\_type from tour

inner join tour\_type on tour\_type.type\_tour\_id = tour.type\_tour\_id AND  
tour.type\_tour\_id = type\_id;

END \$\$

DELIMITER ;

CALL select\_tour\_by\_tour\_type(7);

```

DROP PROCEDURE IF EXISTS select_clients_that_paid_by_category;

-- аналог 16 запиту, обрати всіх клієнтів, що обрали відрочинки з певним
типом категорії та сплатили за нього (значення в рамках [1,7])

DELIMITER $$

CREATE PROCEDURE
select_clients_that_paid_by_category(paid_category_id INT)
BEGIN
    select client.client_id, client.first_name, client.last_name, client.phone_number,
    contract.contract_id,
    payment_status.name_of_payment_status, contract.cost_of_tour,
    tour_type.name_of_tour_type from contract
    inner join client on client.client_id = contract.client_id
    inner join payment on payment.payment_id = contract.contract_id AND
    payment.payment_status_id = 4
    inner join payment_status on payment_status.payment_status_id =
    payment.payment_status_id
    inner join tours_in_contract on tours_in_contract.contract_id =
    contract.contract_id
    inner join tour on tour.tour_id = tours_in_contract.tour_id
    inner join tour_type on tour_type.type_tour_id = tour.type_tour_id AND
    tour.type_tour_id = paid_category_id;
END $$

DELIMITER ;

CALL select_clients_that_paid_by_category(7);

DROP PROCEDURE IF EXISTS select_clients_that_not_paid_by_category;

```

-- аналог 17 запиту, обрати всі місця відпочинку з певним типом категорії, що не були сплачені (значення в рамках [1,7])

```

DELIMITER $$

CREATE PROCEDURE
select_clients_that_not_paid_by_category(not_paid_category_id INT)
BEGIN
    select client.client_id, client.first_name, client.last_name, client.phone_number,
           contract.contract_id, payment_status.name_of_payment_status,
           contract.cost_of_tour, category.name_of_category from contract
    inner join client on client.client_id = contract.client_id
    inner join payment on payment.payment_id = contract.contract_id AND
        (payment.payment_status_id = 3 OR payment.payment_status_id = 1)
    inner join payment_status on payment_status.payment_status_id =
payment.payment_status_id
    inner join tours_in_contract on tours_in_contract.contract_id =
contract.contract_id
    inner join tour on tour.tour_id = tours_in_contract.tour_id
    inner join place_of_rest on place_of_rest.rest_place_id = tour.place_of_rest_id
    inner join category on category.category_id =
place_of_rest.place_of_rest_category AND category_id = not_paid_category_id;

END $$

DELIMITER ;

```

CALL select\_clients\_that\_not\_paid\_by\_category(6);

DROP PROCEDURE IF EXISTS select\_clients\_by\_booking\_category;

-- аналог 18 запиту, обрати всіх клієнтів що бронювали номери певної категорії (значення в рамках [1,6])

```

DELIMITER $$

```

CREATE

PROCEDURE

```

select_clients_by_booking_category(searched_category_id INT)
BEGIN
    select client.client_id, client.first_name, client.last_name, client.phone_number,
           contract.contract_id, contract.cost_of_tour, category.name_of_category from
contract
    inner join client on client.client_id = contract.client_id
    inner join tours_in_contract on tours_in_contract.contract_id =
contract.contract_id
    inner join tour on tour.tour_id = tours_in_contract.tour_id
    inner join place_of_rest on place_of_rest.rest_place_id = tour.place_of_rest_id
    inner join category on category.category_id =
place_of_rest.place_of_rest_category
    AND category_id = searched_category_id;
END $$
DELIMITER ;

```

CALL select\_clients\_by\_booking\_category(5);

#### 8. Оптимізація запитів (Файл «optimization.sql»)

```

create index payment_status_idx on payment_status (payment_status_id,
name_of_payment_status);
create index tour_type_idx on tour_type (type_tour_id, name_of_tour_type);
create index transfer_type_idx on transfer_type(transfer_type_id,
name_transfer_type);
create index tour_idx on tour (tour_id, place_of_rest_id);
create index contract_idx on contract (contract_id, client_id, visa_required,
cost_of_tour);

drop index payment_status_idx on payment_status;

```

```
drop index tour_type_indx on tour_type ;  
drop index transfer_type_indx on transfer_type;  
drop index tour_indx on tour;  
drop index contract_indx on contract;
```