|  |  |
| --- | --- |
| 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 *ІП-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)

|  |  |  |  |
| --- | --- | --- | --- |
| 1. Строк подання студентом роботи | | | *27.12.2021* |
| 1. Вихідні дані до роботи | | *Тексти програмного коду(Додаток А)* | |
|  | | | |
|  | | | |
|  | | | |
|  | | | |
|  | | | |
|  | | | |
| 1. 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* | | | |
|  | | | |
|  | | | |
|  | | | |
|  | | | |
| 1. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень) | | | |
| *ER-діаграма, приклади виконання SQL скриптів, реляційна схема бази даних* | | | |
|  | | | |
|  | | | |

|  |  |
| --- | --- |
| 1. Дата видачі завдання | *31.10.2021* |

|  |  |  |  |
| --- | --- | --- | --- |
| № 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* |  |

Календарний план

|  |  |  |  |
| --- | --- | --- | --- |
| **Студент** |  |  | *Галько М. В.* |

(підпис) (прізвище та ініціали)

|  |  |  |  |
| --- | --- | --- | --- |
| **Керівник роботи** |  |  | *Лебідь С. О.* |

(підпис) (прізвище та ініціали)

Зміст

[Вступ 4](#_Toc90817691)

[1 Аналіз предметної області 5](#_Toc90817692)

[2 Постановка завдання 9](#_Toc90817693)

[3 Концептуальна модель бази даних 10](#_Toc90817694)

[3.1 Побудова ER-діаграми 10](#_Toc90817695)

[3.2 Відношення між об’єктами БД 11](#_Toc90817696)

[4 Логічна модель бази даних 14](#_Toc90817697)

[4.1 Схеми таблиць БД 14](#_Toc90817698)

[4.2 Схема БД 17](#_Toc90817699)

[5 Реалізація бази даних 18](#_Toc90817700)

[5.1 Створення таблиць 18](#_Toc90817701)

[5.2 Забезпечення цілісності даних **Ошибка! Закладка не определена.**](#_Toc90817702)

[5.3 Створення обмежень **Ошибка! Закладка не определена.**](#_Toc90817703)

[5.4 Створення користувачів **Ошибка! Закладка не определена.**](#_Toc90817704)

[5.5 Імпорт даних у БД 24](#_Toc90817705)

[5.6 Створення запитів 26](#_Toc90817706)

[5.7 Створення процедур 35](#_Toc90817707)

[5.8 Оптимізація запитів 41](#_Toc90817708)

[Висновки 45](#_Toc90817709)

[Перелік посилань 46](#_Toc90817710)

[Додаток А 47](#_Toc90817711)

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 даний 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 – 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.
* Mockaroo – допоміжний сайт для генерації даних для їх майбутнього завантаження у створену базу даних.

# 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;
* Унікальне номер розташування ділянки;
* Унікальний номер типу використання ділянки;
* Унікальний номер інженерних комунікацій;

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;
* Назва 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;
* 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;
* Унікальний номер ресурсу;
* Назва;
* Податок за ресурс;

Сутність **Інженерна комунікація**:

* Унікальний номер;
* 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. Номера 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 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. Відображання 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. Відображання 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. Відображання 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.

# 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.

# Концептуальна модель бази даних

## Побудова ER-діаграми

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

Назви сутностей та їх призначення

|  |  |
| --- | --- |
| **Сутність** | **Набір атрибутів** |
| Ділянка (Land) | * ID * ID власника * ID розташування * ID типу використання * ID інженерних комунікацій |
| Фізична особа  (Natural) | * ID * Ім’я * Прізвище * Дата народження * Номер телефону |
| Юридична особа  (Legal) | * ID * Назва установи * ID фізичного представника |
| Розташування (Location) | * ID * Адреса * Податок |
| Тип використання ділянки (UsageType) | * ID * Назва типу * Податок |
| Об’єкт (Object) | * ID * ID ділянки * Широта нижнього лівого кута * Широта верхнього правого кута * Висота нижнього лівого кута |

Продовження таблиці 3.1

|  |  |
| --- | --- |
| **Назва** | **Обумовленість** |
| Об’єкт (Object) | * Висота верхнього правого кута |
| Ресурс (Resource) | * ID * ID об’єкту * Назва * Податок |
| Інженерна комунікація (Utility) | * ID * 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 |

## 

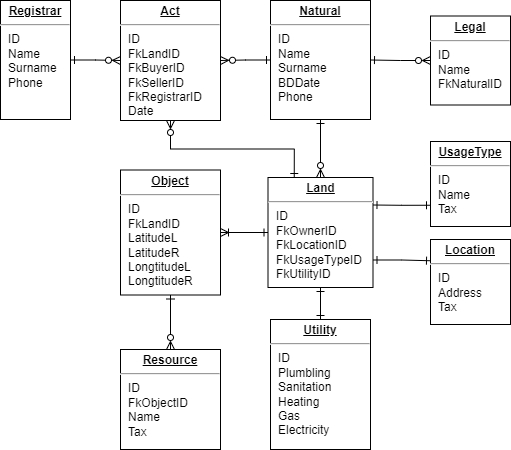
## Відношення між об’єктами БД

Зв’язки між сутностями наведено у таблиці 3.2

Зв’язки між сутностями

|  |  |  |
| --- | --- | --- |
| **Перша сутність** | **Друга сутність** | **Зв’язок** |
| Land | UsageType | 1:1 (fk\_usageType) |
| 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\_object) |
| Natural | Land | 1 : many (fk\_natural) |
| Natural | Legal | 1 : many (fk\_natural) |
| Natural | Act | 1 : many (fk\_natural) |
| Registrar | Act | 1 : many (fk\_registrar) |

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



* + - * 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.

# Логічна модель бази даних

## Схеми таблиць бази даних

Схема таблиці Land

|  |  |
| --- | --- |
| **Атрибут** | **Тип даних** |
| (PK) ID | INT |
| (FK) IdOwner | INT |
| (FK) IdLocation | INT |
| (FK) IdUsageType | INT |
| (FK) IdUtility | INT |

Схема таблиці Natural

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

Схема таблиці Location

|  |  |
| --- | --- |
| **Атрибут** | **Тип даних** |
| (PK) ID | INT |
| Address | VARCHAR(50) |
| Tax | FLOAT |

Схема таблиці UsageType

|  |  |
| --- | --- |
| **Атрибут** | **Тип даних** |
| (PK) ID | INT |
| Name | VARCHAR(50) |

Продовження таблиці 4.4

|  |  |
| --- | --- |
| Tax | FLOAT |

Схема таблиці Object

|  |  |
| --- | --- |
| **Атрибут** | **Тип даних** |
| (PK) ID | INT |
| (FK) IdLand | INT |
| LatitudeL | FLOAT |
| LatitudeR | FLOAT |
| LongtitudeU | FLOAT |
| LongtitudeD | FLOAT |

Схема таблиці Resource.

|  |  |
| --- | --- |
| **Атрибут** | **Тип даних** |
| (PK) ID | INT |
| (FK) IdObject | INT |
| Name | VARCHAR(50) |
| TAX | FLOAT |

Схема таблиці Utility

|  |  |
| --- | --- |
| **Атрибут** | **Тип даних** |
| (PK) ID | INT |
| (FK) IdLand | INT |
| Plumbing | BIT |
| Sanitation | BIT |
| Heating | BIT |
| Gas | BIT |
| Electricity | BIT |

Схема таблиці Legal

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

Схема таблиці Act

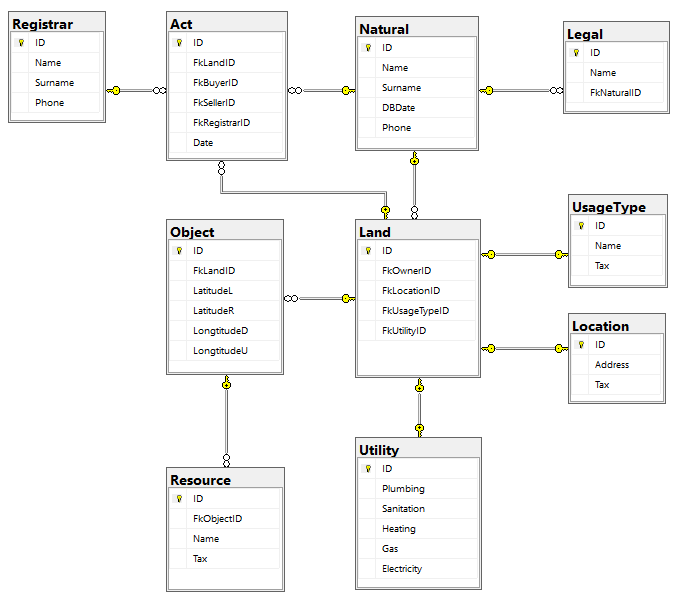
|  |  |
| --- | --- |
| **Атрибут** | **Тип даних** |
| (PK) ID | INT |
| (FK) IdLand | INT |
| (FK) IdBuyer | INT |
| (FK) IdSeller | INT |
| (FK) IdRegistrar | INT |
| Date | DATE |

Схема таблиці Registrar

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

## Схема БД

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



* + - * 1. Схема бази даних

# Реалізація бази даних

## Створення таблиць

Для реалізації БД в середовищі MS SQL Server, треба створити базу даний, 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 float default(0) not null

)

go

alter table UsageType

add constraint CH\_UsageTypeTax check (Tax >= 0)

go

--------------------------------------------------------------------------

create table Location (

ID int identity(1,1) primary key,

Address varchar(50) not null unique,

Tax float default(0) not null

)

go

alter table Location

add constraint CH\_LocationTax 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\_NaturalName

check (Name not like '%[^A-Za-z'' -]%')

go

alter table Natural

add constraint CH\_NaturalSurname

check (Surname not like '%[^A-Za-z'' -]%')

go

alter table Natural

add constraint CH\_NaturalDBDate

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

alter table Natural

add constraint CH\_DBDate check (DATEDIFF(yy, DBDate, GETDATE()) >= 18)

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\_RegistrarName

check (Name not like '%[^A-Za-z'' -]%')

go

alter table Registrar

add constraint CH\_RegistrarSurname

check (Surname not like '%[^A-Za-z'' -]%')

go

alter table Registrar

add constraint CH\_RegistrarPhone

check (Phone like '\_\_\_-\_\_\_-\_\_\_\_' and Phone not like '%[a-Z]%')

go

--------------------------------------------------------------------------

create table Utility (

ID int identity(1,1) primary key,

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

--------------------------------------------------------------------------

create table Legal (

ID int identity(1,1) primary key,

Name varchar(50) not null,

FkNaturalID int not null foreign key references Natural(ID)

on update cascade

on delete cascade

)

go

--------------------------------------------------------------------------

create table Land (

ID int identity(1,1) primary key,

FkOwnerID int not null foreign key

references Natural(ID)

on update cascade

on delete cascade,

FkLocationID int not null unique foreign key

references Location(ID)

on update cascade

on delete cascade,

FkUsageTypeID int unique foreign key

references UsageType(ID)

on update cascade

on delete set null,

FkUtilityID int null unique foreign key

references Utility(ID)

on update cascade

on delete set null

)

go

--------------------------------------------------------------------------

create table Object (

ID int identity(1,1) primary key,

FkLandID int not null foreign key

references Land(ID)

on update cascade

on delete cascade,

LatitudeL float not null,

LatitudeR float not null,

LongtitudeD float not null,

LongtitudeU float not null

)

go

alter table Object

add constraint CH\_Latitude check (LatitudeL < LatitudeR)

go

alter table Object

add constraint CH\_Longtitude check (LongtitudeD < LongtitudeU)

go

--------------------------------------------------------------------------

create table Act (

ID int identity(1,1) primary key,

FkLandID int foreign key

references Land(ID)

on update cascade

on delete set null,

FkBuyerID int not null foreign key

references Natural(ID),

FkSellerID int not null foreign key

references Natural(ID),

FkRegistrarID int not null foreign key

references Registrar(ID)

on update cascade,

Date date default(getdate()) not null

)

go

alter table Act

add constraint CH\_ActNaturals check (FkBuyerID != FkSellerID)

go

alter table Act

add constraint CH\_ActDate check (Date <= GETDATE())

go

--------------------------------------------------------------------------

create table Resource (

ID int identity(1,1) primary key,

FkObjectID int not null foreign key

references Object(ID)

on update cascade

on delete cascade,

Name varchar(50) not null,

Tax float DEFAULT(0) not null

)

go

alter table Resource

add constraint CH\_ResourceTax check (Tax >= 0)

go

# 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 (рис. 6.2). Інший же полягає у імпортуванні даних через файли форматів .csv або .json (рис. 6.3). Звичайно, для великого обсягу даних піддійте другий спосіб. Але оскільки дана робота більш демонструє по своїй суті уміння опрацьовувати дані, то загальний об’єм даних не буде більшим за 100 запитів. Однак для загального прикладу використаємо обидва способи для усіх файлів (рис. 6.1).

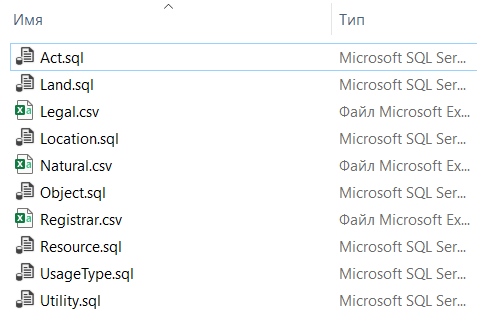


Рисунок 6.1 – sql та csv файли з даними таблиць

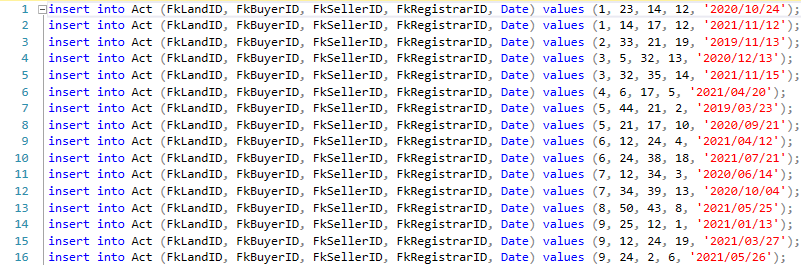


Рисунок 6.2 - Скрипти для вставки даних у БД

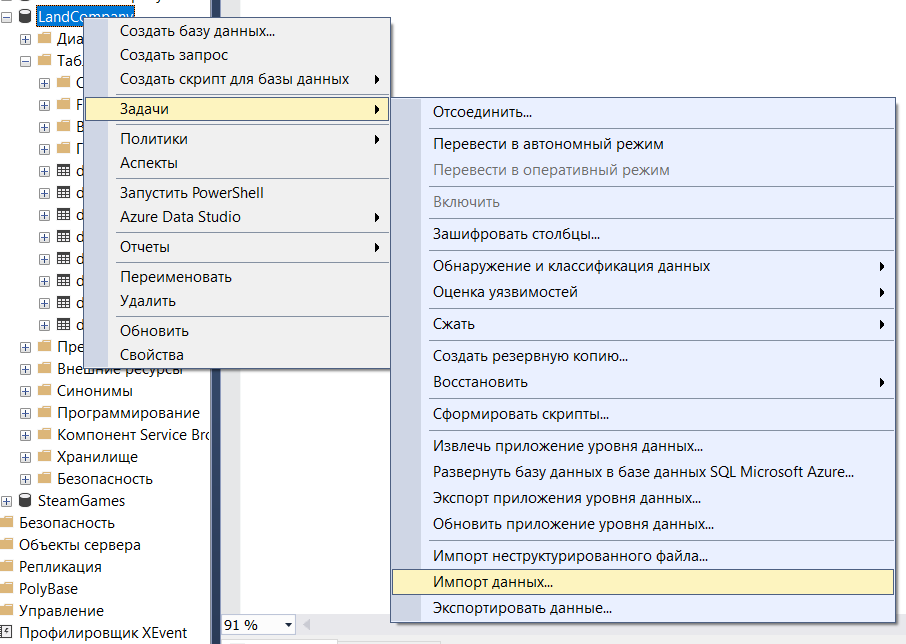


Рисунок 6.3 – функція імпорту даних через файл

Як видно з рисунка 6.1 усі файли окрім: Legal, Natural, Registrar – у форматі SQL, тобто мають у собі скрипти для завантаження даних, подібно рисунку 6.2.

Інші ж - у форматі .csv, а отже були використані як джерело (Flat file source) при імпорті даних (рис. 6.4) у таблиці.

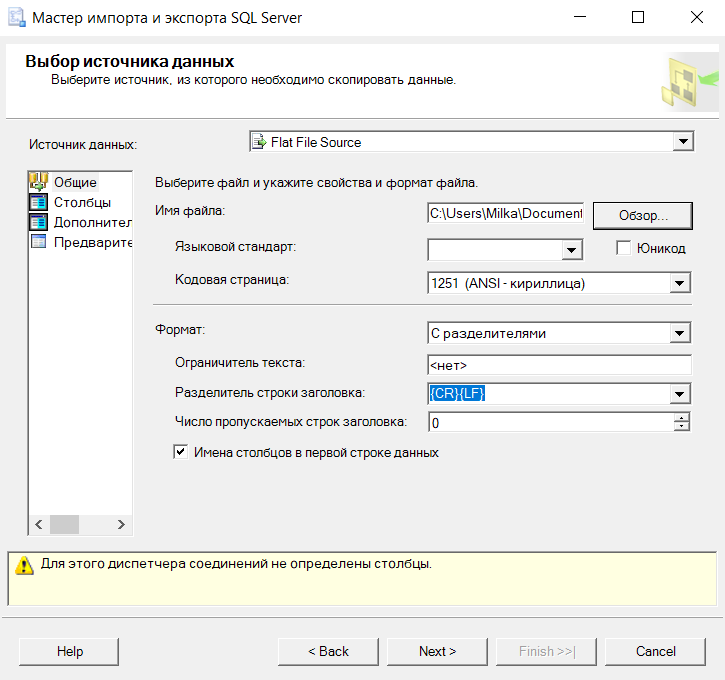


Рисунок 6.4 – обрання Flat File Source

Далі продовжуємо натискати «Next» поки не потрапимо на вікно зображене на рисунку 6.5, де обираємо SQL Server Native Client 11.0.

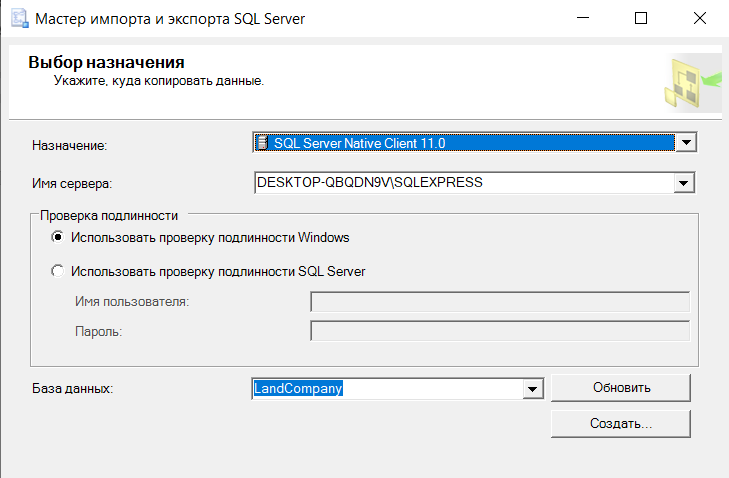


Рисунок 6.5 – обрання SQL Server Native Client 11.0

Фінальним етапом є натискання кнопки «Finish», де після завантаження даних відповідно відображається чи був імпорт успішним у вигляді статусів (рис.6.6). У разі успіху усі дані вже є у таблицях. В іншому випадку необхідно читати повідомлення про помилку, аби зрозуміти в чому полягає несумісність даних у файлі із таблицею. Самі ж дані не були завантаженні при невдачі.

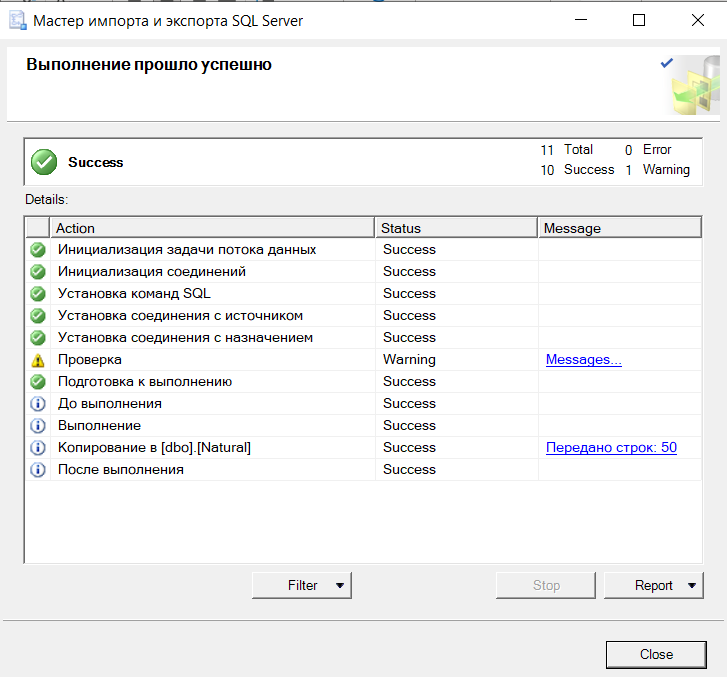


Рисунок 6.6 – вікно статусів імпорту даних

# 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:

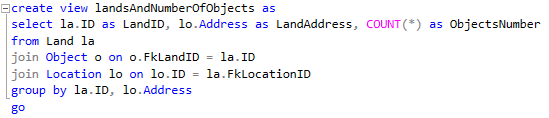


Рисунок 7.1 – Скрипт для відображення загальної кількості об’єктів для кожної з ділянок

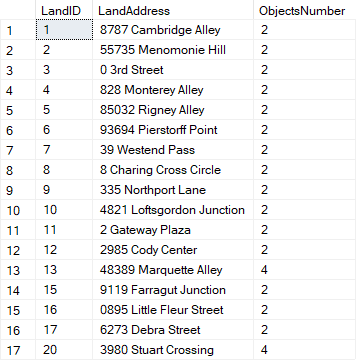


Рисунок 7.2 – Результат запиту

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

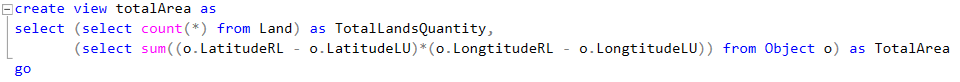


Рисунок 7.3 – Скрипт для відображення загальної площі усіх ділянок, що є у базі та їх кількості



Рисунок 7.4 – Результат запиту

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

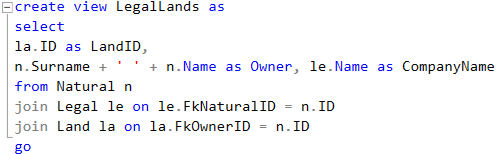


Рисунок 7.5 – Скрипт для відображення ділянок, власниками яких є юридичні особи



Рисунок 7.6 – Результат запиту

1. Запит для відображання ділянок, власниками яких є фізичні особи:

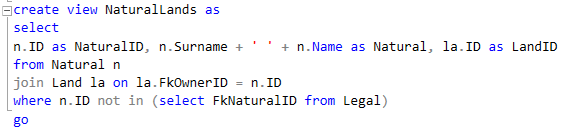


Рисунок 7.7 – Скрипт для відображення ділянок, власниками яких є фізичні особи

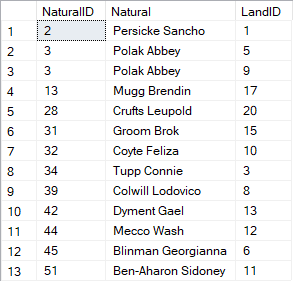


Рисунок 7.8 – Результат запиту

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

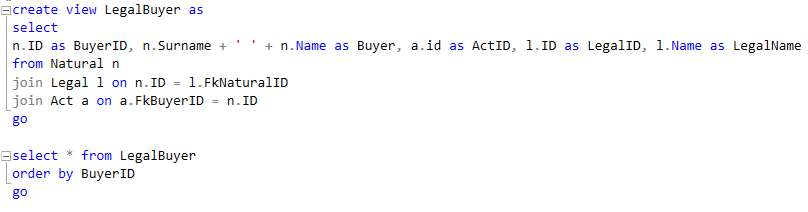


Рисунок 7.9 – Скрипт для відображення актів, де покупець був юридична особа

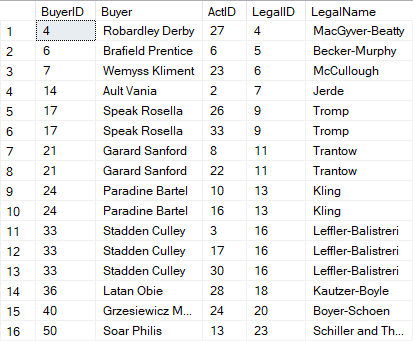


Рисунок 7.10 – Результат запиту

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

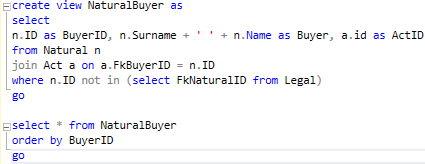


Рисунок 7.11 – Скрипт для відображення актів, де продавець був фізична особа

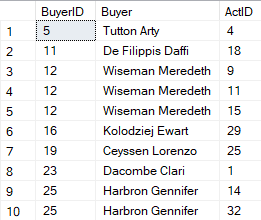


Рисунок 7.12 – Результат запиту

1. Запит на відображення затверджених актів за останній рік:

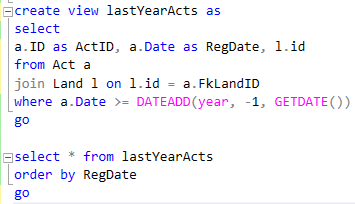


Рисунок 7.13 – Скрипт для відображення затверджених актів за останній рік

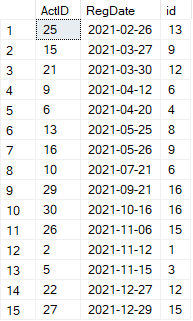


Рисунок 7.14 – Результат запиту

1. Запит на відображення ресурсів певного власника ділянки:

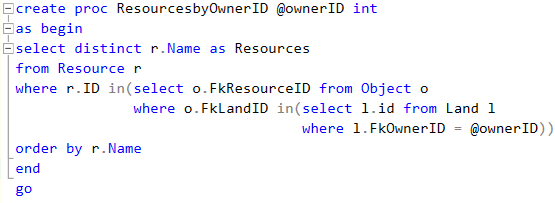


Рисунок 7.15 – Скрипт для відображення ресурсів певного власника ділянки

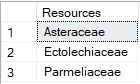


Рисунок 7.16 – Результат запиту

1. Запит на відображення ділянок без інженерних комунікацій:

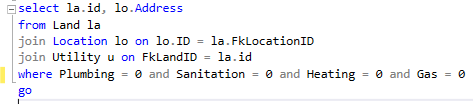


Рисунок 7.17 – Скрипт для відображення ділянок без інженерних комунікацій

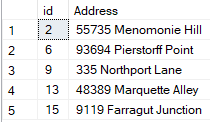


Рисунок 7.18 – Результат запиту

1. Запит на відображання осіб з найбільшою площею ділянки:

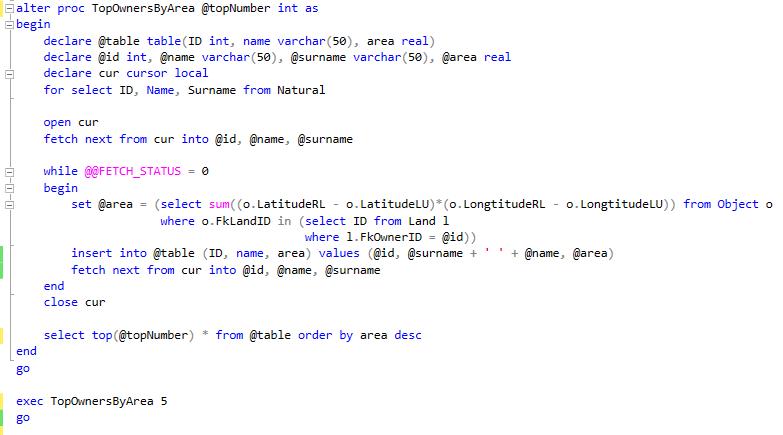


Рисунок 7.19 – Скрипт для відображення осіб з найбільшою площею ділянки:



Рисунок 7.20 – Результат запиту

1. Запит на відображення усіх ділянок із каналізацією:

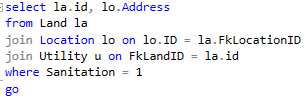


Рисунок 7.23 – Скрипт для відображення усіх ділянок із каналізацією

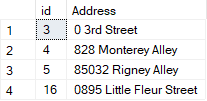


Рисунок 7.24 – Результат запиту

1. Запит на відображення історії купівлі-продажу ділянки:

Рисунок 7.21 – Скрипт для відображення

Рисунок 7.22 – Результат запиту

1. Запит на відображення ділянок, де є опалення:

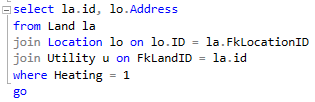


Рисунок 7.25 – Скрипт для відображення ділянок, де є опалення

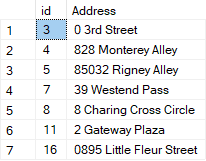


Рисунок 7.26 – Результат запиту

1. Запит на відображення власників ділянки із електроенергією:

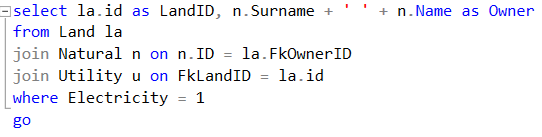


Рисунок 7.27 – Скрипт для відображення власників ділянки із електроенергією

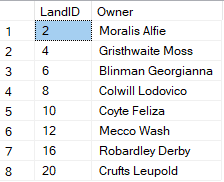


Рисунок 7.28 – Результат запиту

1. Запит на відображення кількості ділянок із газом:

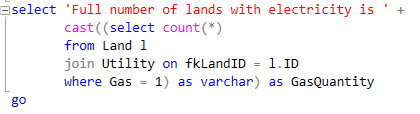


Рисунок 7.29 – Скрипт для відображення кількості ділянок із газом



Рисунок 7.30 – Результат запиту

1. Запит на відображання 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

1. Запит на відображення актів затвердженими реєстратором:

Рисунок 7.33 – Скрипт для відображення актів затвердженими реєстратором

Рисунок 7.34 – Результат запиту

1. Запит на відображення суми вартості ділянок фізичних осіб:

Рисунок 7.35 – Скрипт для відображення суми вартості ділянок фізичних осіб

Рисунок 7.36 – Результат запиту

1. 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

1. Запит на відображення кількості актів проведених із ділянкою:

Рисунок 7.39 – Скрипт для відображення кількості актів проведених із ділянкою

Рисунок 7.40 – Результат запиту

1. Клієнт повинен бути повнолітнім.
2. Дата заключення акта не може бути пізнішою за сьогоднішню дату.
3. Дати вказуються в часовому поясі UTC+0.
4. Податок не може бути від'ємним.
5. Усі номера телефонів унікальні.
6. Довгота лівого верхнього кута повинна бути меншою за довготу правого нижнього.
7. Широта лівого верхнього кута повинна бути меншою за довготу правого нижнього.

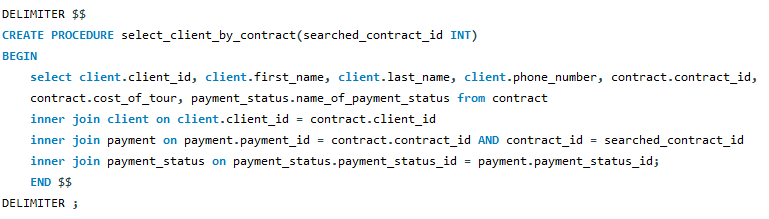
Для реалізації корисності БД необхідно розробити наступні запити та реалізувати їх на мові SQL:

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

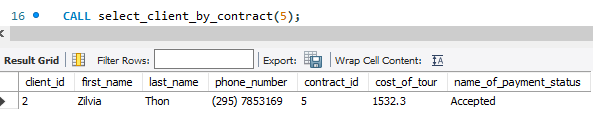
## Створення процедур

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

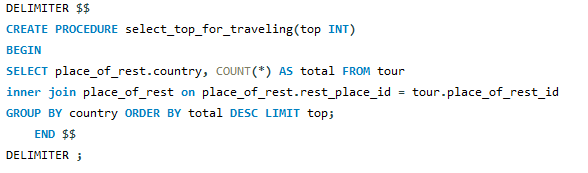
1. Обрати клієнта по id певного контракту(рис 5.53). Приклад виклику та роботи процедури представлено на рисунку 5.54.

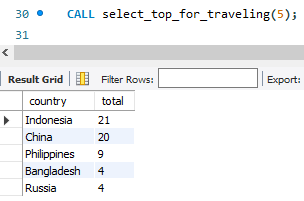


* + - * 1. Скрипт процедури для обрання клієнта по id контракту



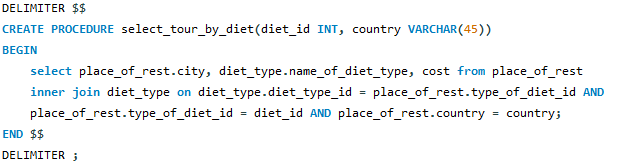
* + - * 1. Приклад виклику та роботи процедури для обрання клієнта по id контракту рівному 5

1. Вивести топ – n найпопулярніших країн для мандрівки(рис 5.55) Приклад виклику та роботи процедури представлено на рисунку 5.56.
   * + - 1. Вивести топ – n найпопулярніших країн

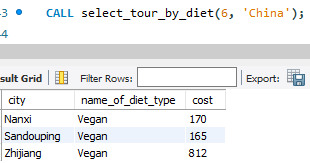


* + - * 1. Приклад виклику та роботи процедури для виведення топ – 5 найпопулярніших країн

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

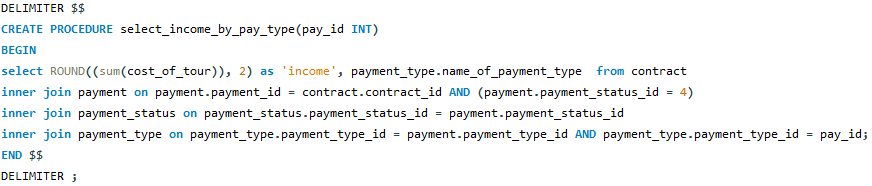


* + - * 1. Обрати всі місця відпочинку з певною дієтою та певною країною

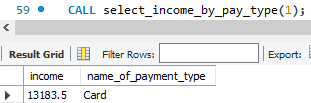


* + - * 1. Приклад виклику та роботи процедури для обрання турів з веганською дієтою у Китаї

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

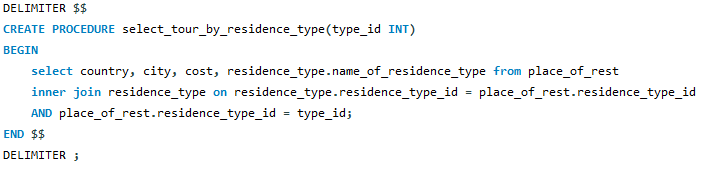


* + - * 1. Порахувати дохід за певним видом розрахунку

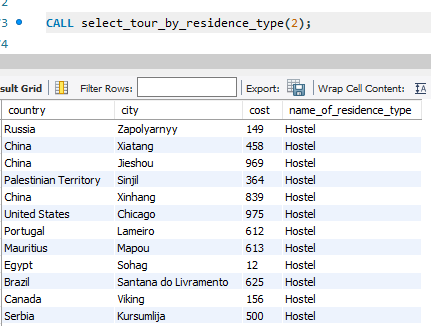


* + - * 1. Приклад виклику та роботи процедури: порахувати дохід за видом розрахунку Картка

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

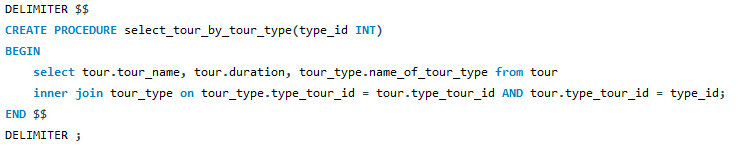


* + - * 1. Обрати всі місця відпочинку з певним типом

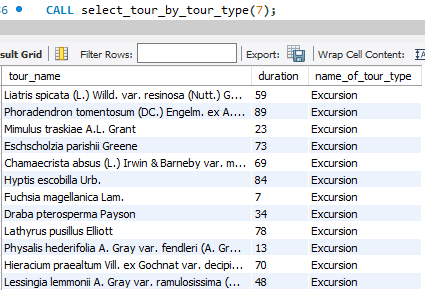


* + - * 1. Обрати всі місця відпочинку з типом Хостел

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

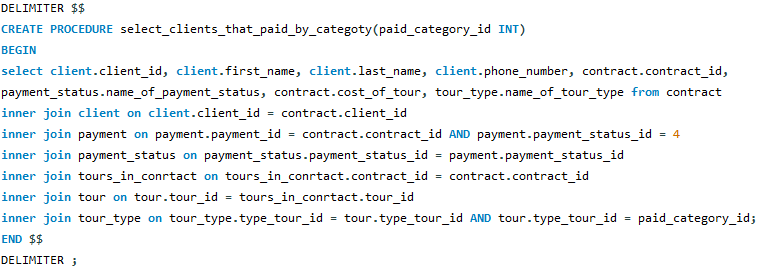


* + - * 1. Обрати всі тури з певним типом туру

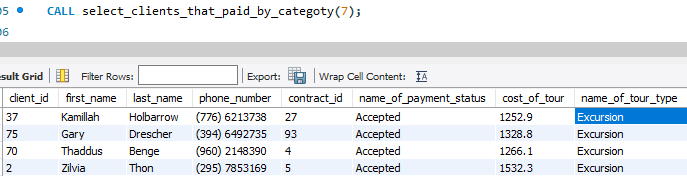


* + - * 1. Приклад виклику та роботи процедури обрання всіх турів з типом туру Екскурсія

1. 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

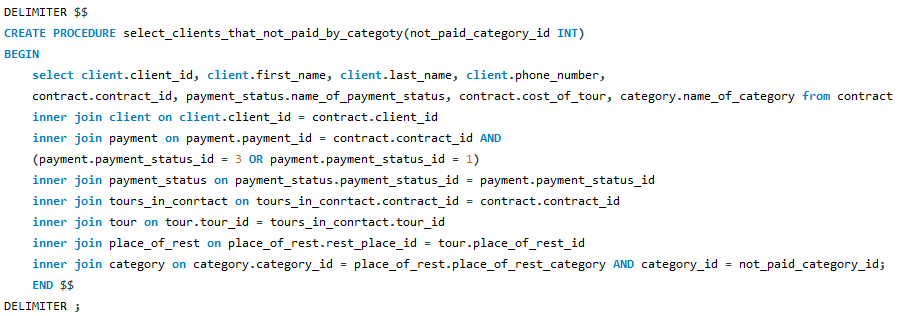


* + - * 1. Обрати всіх клієнтів, що обрали відрочинок з певним типом категорії та сплатили за нього

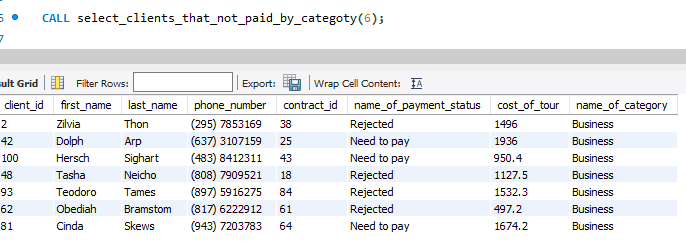


* + - * 1. Приклад виклику та роботи процедури обрання всіх турів з типом туру та сплатили за нього

1. 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

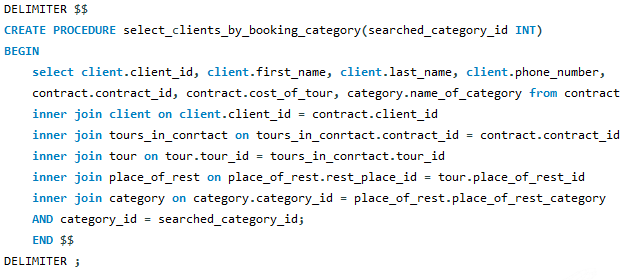


* + - * 1. Обрати всі місця відпочинку з певним типом категорії, що не були сплачені

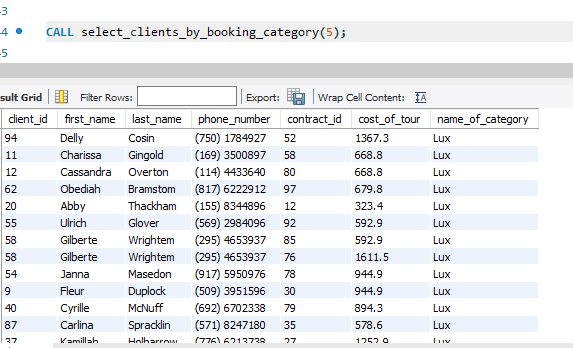


* + - * 1. Приклад виклику та роботи процедури обрання всіх місць відпочинку з типом категорії Бізнес, що не були сплачені

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



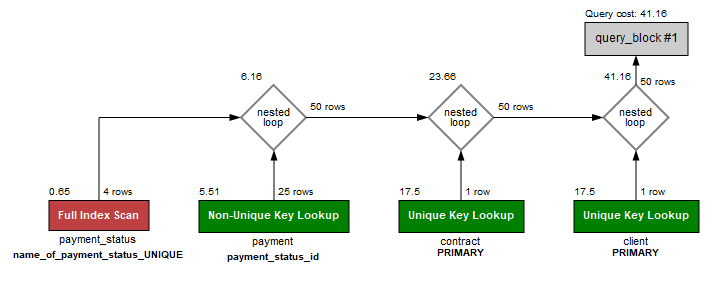
* + - * 1. Обрати всіх клієнтів що бронювали номери певної категорії



* + - * 1. Приклад виклику та роботи процедури обрання всіх клієнтів що бронювали номери категорії Люкс

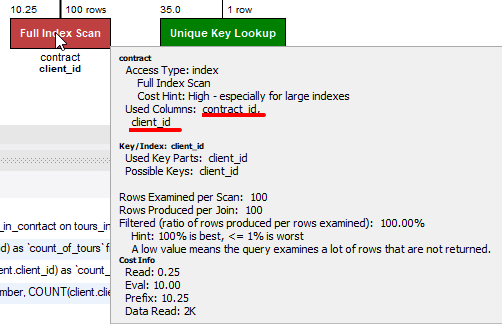
## Оптимізація запитів

Для оптимізації запитів буде використано можливість створення індексів у MySQL. Індекси використовуються для швидкого вилучення даних із бази даних. Користувачі не бачать індексів, вони просто використовуються для прискорення пошуків/запитів. Індекси 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)



* + - * 1. 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)



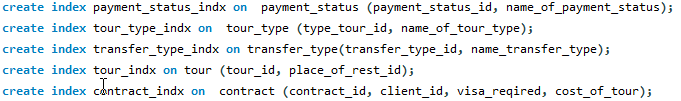
* + - * 1. Додаткова інформація про запит

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:

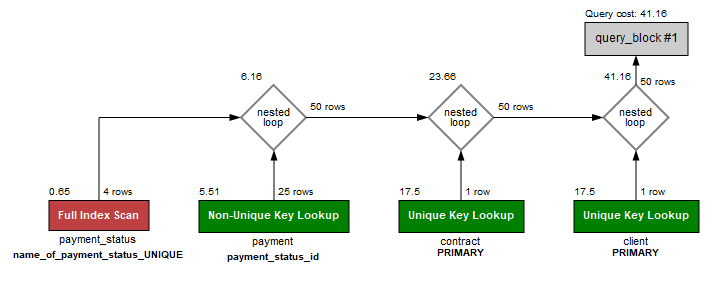
1. Таблиця payment\_status, колонки payment\_status\_id, name\_of\_payment\_status
2. Таблиця tour\_type, колонки type\_tour\_id, name\_of\_tour\_type
3. Таблиця transfer\_type, колонки transfer\_type\_id, name\_transfer\_type
4. Таблиця tour, колонки tour\_id, place\_of\_rest\_id
5. Таблиця contract,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

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

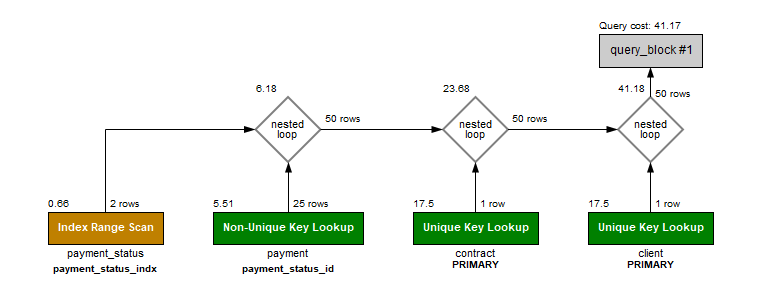


* + - * 1. Скрипт для створення індексів

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.



* + - * 1. Схема запиту до створення індексів



* + - * 1. 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
2. Організація баз даних: практичний курс: Навч. посіб. для студ. / А. Ю. Берко, О. М. Верес; Нац. ун-т «Львів. політехніка». — Л., 2003. — 149 c.
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 строения баз данных. 8 -е изд. / Д.Кренке. –

СПб.: Питер, 2003. – 800 с.

1. MySQL Workbench – [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/MySQL\_Workbench

Додаток А Тексти програмного коду

*студента групи ІП-02 II курсу*

*Трофимова Д.О.*

(Обсяг програми (документа), арк., Кб)

*34 арк, 66,6 Кб*

(Вид носія даних)

*HDD*

(Найменування програми (документа))

*Тексти програмного коду програмного забезпечення*

*Підтримка діяльності туристичної компанії*

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\_reqired` 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\_conrtact` (

`tour\_in\_conrtact\_id` int NOT NULL AUTO\_INCREMENT,

`tour\_id` int NOT NULL,

`contract\_id` int NOT NULL,

PRIMARY KEY (`tour\_in\_conrtact\_id`)

);

1. Обмеження(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;

1. 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'@'%';

1. Тригер для перевірки дат прибуття та відбуття на коректність (Файл «depatrure 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

1. 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

1. 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\_conrtact on tours\_in\_conrtact.contract\_id = contract.contract\_id

inner join tour on tour.tour\_id = tours\_in\_conrtact.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 кліентів і іх 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\_reqired = 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\_conrtact on tours\_in\_conrtact.contract\_id = contract.contract\_id

inner join tour on tour.tour\_id = tours\_in\_conrtact.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 бронюівали 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\_conrtact on tours\_in\_conrtact.contract\_id = contract.contract\_id

inner join tour on tour.tour\_id = tours\_in\_conrtact.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 бранювали 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\_conrtact on tours\_in\_conrtact.contract\_id = contract.contract\_id

inner join tour on tour.tour\_id = tours\_in\_conrtact.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\_conrtact on tours\_in\_conrtact.contract\_id = contract.contract\_id

inner join tour on tour.tour\_id = tours\_in\_conrtact.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;

1. 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-n 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\_categoty;

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

DELIMITER $$

CREATE PROCEDURE select\_clients\_that\_paid\_by\_categoty(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\_conrtact on tours\_in\_conrtact.contract\_id = contract.contract\_id

inner join tour on tour.tour\_id = tours\_in\_conrtact.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\_categoty(7);

DROP PROCEDURE IF EXISTS select\_clients\_that\_not\_paid\_by\_categoty;

-- 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\_categoty(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\_conrtact on tours\_in\_conrtact.contract\_id = contract.contract\_id

inner join tour on tour.tour\_id = tours\_in\_conrtact.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\_categoty(6);

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

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

DELIMITER $$

CREATE PROCEDURE select\_сlients\_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\_conrtact on tours\_in\_conrtact.contract\_id = contract.contract\_id

inner join tour on tour.tour\_id = tours\_in\_conrtact.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\_сlients\_by\_booking\_category(5);

1. Оптимізація запитів (Файл «optimization.sql»)

create index payment\_status\_indx on payment\_status (payment\_status\_id, name\_of\_payment\_status);

create index tour\_type\_indx on tour\_type (type\_tour\_id, name\_of\_tour\_type);

create index transfer\_type\_indx on transfer\_type(transfer\_type\_id, name\_transfer\_type);

create index tour\_indx on tour (tour\_id, place\_of\_rest\_id);

create index contract\_indx on contract (contract\_id, client\_id, visa\_reqired, cost\_of\_tour);

drop index payment\_status\_indx 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;